Identity Aware LBS

► Privacy Threats in Location-Based Services

Identity Unaware LBS

► Privacy Threats in Location-Based Services

iDistance Techniques

H.V. JAGADISH[1], BENG CHIN OOI[2], RUI ZHANG[3]
[1] Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA
[2] Department of Computer Science, National University of Singapore, Singapore, Singapore
[3] Department of Computer Science and Software Engineering, The University of Melbourne, Parkville, VIC, Australia

Synonyms

Query, nearest neighbor; Scan, sequential

Definition

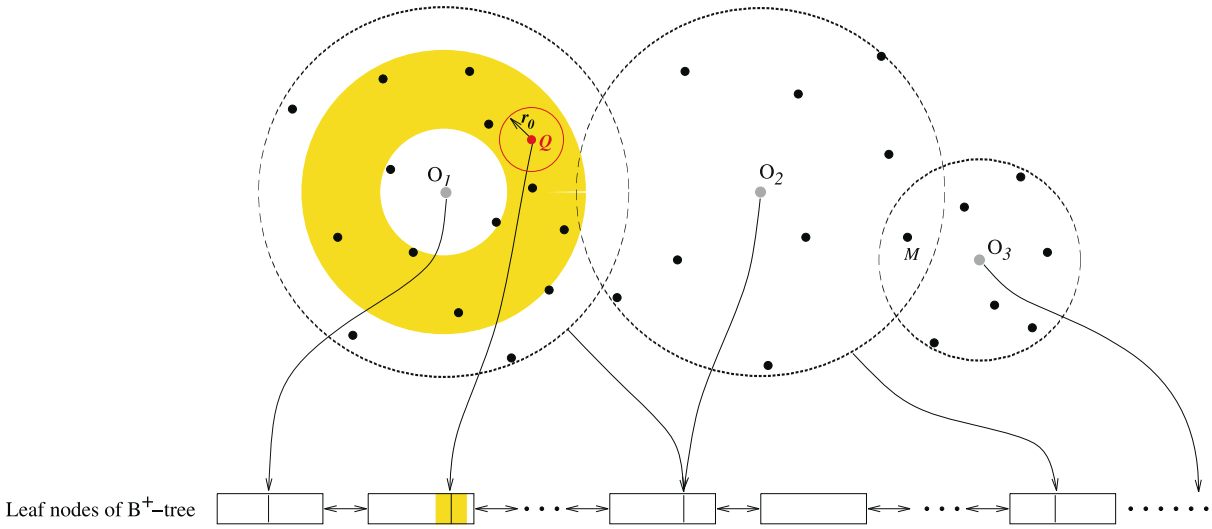
The iDistance is an indexing and query processing technique for k nearest neighbor (kNN) queries on point data in multi-dimensional metric spaces. The kNN query is one of the hardest problems on multi-dimensional data. It has been shown analytically and experimentally that any algorithm using hierarchical index structure based on either space- or data-partitioning is less efficient than the naive method of sequentially checking every data record (called the *sequential scan*) in high-dimensional spaces [4]. Some data distributions including the uniform distribution are particularly hard cases [1]. The iDistance is designed to process kNN queries in high-dimensional spaces efficiently and it is especially good for skewed data distributions, which usually occur in real-life data sets. For uniform data, the iDistance beats the sequential scan up to 30 dimensions as reported in [3]. Building the iDistance index has two steps. First, a number of reference points in the data space are chosen. There are various ways of choosing reference points. Using cluster centers as reference points is the most efficient way. Second, the distance between a data point and its closest reference point is calculated. This distance plus a scaling value is called the point's *iDistance*. By this means, points in a multi-dimensional space are mapped to one-dimensional values, and then a B^+-tree can be adopted to index the points using the iDistance as the key. A kNN search is mapped to a number of one-dimensional range searches, which can be processed efficiently on a B^+-tree. The iDistance technique can be viewed as a way of accelerating the sequential scan. Instead of scanning records from the beginning to the end of the data file, the iDistance starts the scan from spots where the nearest neighbors can be obtained early with a very high probability.

Historical Background

The iDistance was first proposed by Cui Yu, Beng Chin Ooi, Kian-Lee Tan and H. V. Jagadish in 2001 [5]. Later, together with Rui Zhang, they improved the technique and performed a more comprehensive study on it in 2005 [3].

Scientific Fundamentals

Figure 1 shows an example of how the iDistance works. The black dots are data points and the gray dots are reference points. The number of reference points is a tunable parameter, denoted by N_r. The recommended value for N_r is between 60 and 80. In this example, $N_r = 3$. At first, 3 cluster centers of the data points, O_1, O_2, O_3 are identified using a clustering algorithm, and these cluster centers are chosen as reference points. Each data point p is assigned to its closest reference point and hence all the data points are divided into 3 partitions P_1, P_2 and P_3. Let d_i be the distance between O_i and the farthest data point to O_i in P_i. Then the range of P_i is a sphere centered at O_i with the radius d_i. Some data points may be contained in the ranges of multiple partitions, such as M in the figure. Such a point only belongs to the partition whose reference point



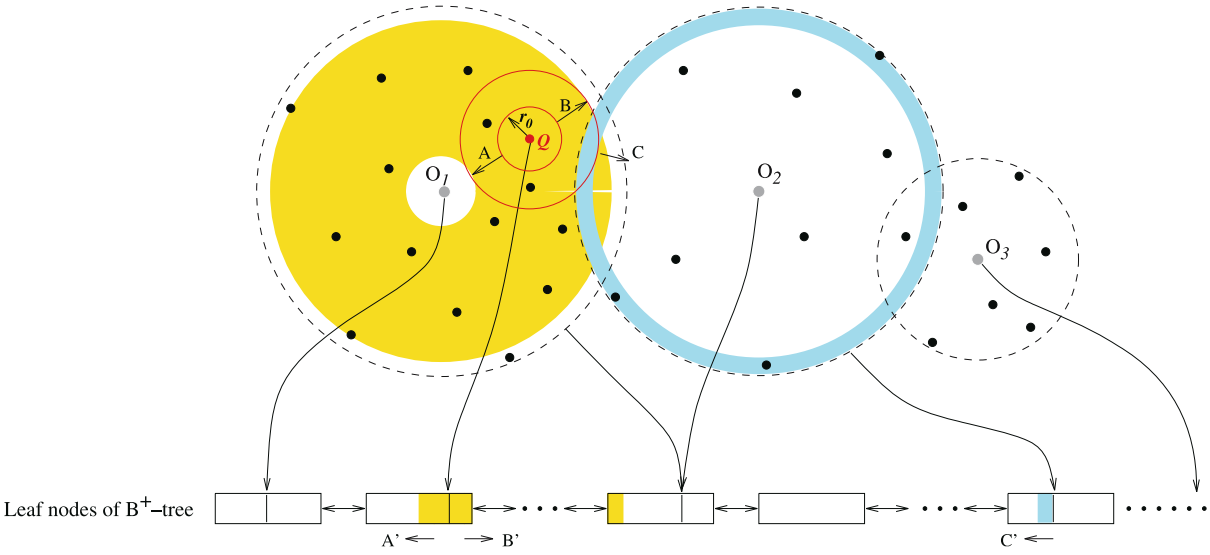
iDistance Techniques, Figure 1 KNN search of the iDistance

is closest to it. The closest reference point to M is O_3, so M belongs to P_3. Now the iDistance of a point p can be defined. Let P_i be the partition p belongs to. Then the iDistance of p

$$iDist(p) = dist(p, O_i) + i \cdot c\,,$$

where $dist(p_1,p_2)$ returns the distance between two points p_1 and p_2; i is the partition number and c is a constant used to stretch the data ranges. All points in partition P_i are mapped to the range $[i{\cdot}c, (i+1){\cdot}c)$. c is set sufficiently large to avoid the overlap between the iDistance ranges of different partitions. Typically, it should be larger than the length of the diagonal in the hyper-rectangular data space. Given the definition of the iDistance, building the index is simple. The data points are stored in the leaf nodes of a B$^+$-tree using their iDistances as keys. Each partition corresponds to a continuous range of entries in the leaf nodes as shown in Fig. 1. Insertions and deletions are performed the same way as in a usual B$^+$-tree. The kNN search algorithm based on the iDistance index is described in the next paragraph.

Like the strategy of many other kNN search algorithms, the iDistance kNN search algorithm begins by searching a small "sphere", and incrementally enlarges the search sphere until the kNN are found. See the query point Q in Fig. 1. Let r denote the search radius. The search starts by a sphere region centered at Q with a initial radius r_0. Then r is iteratively increased by a small amount Δr until the kNN are found. At every iteration, the enlarged portion of the search sphere corresponds to an iDistance range (or two) for any intersected partition. All the points with their iDistances in the range(s) are retrieved in this iteration. During the whole process of enlarging the query sphere, the k nearest points to Q retrieved so far are maintained as candidates. Let $p_{k\text{th}}$ be the kth nearest candidate point to Q. The algorithm terminates when $dist(p_{k\text{th}}, Q){\le}r$, which is the termination condition. At this moment, the k candidates are the actual kNN. For the example query Q in Fig. 1, its initial search sphere intersects partition P_1. In an intersected partition, there is a region that corresponds to the same iDistance range as the search sphere. This region is called the *mapped region* (of the query sphere), which is in the shape of an annulus around the reference point (the shaded region in P_1). All the points in the mapped region are retrieved. Note that all these points are stored continuously in the leaf nodes of the index, so the retrieval is a range query on the B$^+$-tree. As the current r is not greater than $p_{k\text{th}}$, it is increased by Δr and the search is continued. Figure 2 shows the enlarged search sphere and its mapped regions. Now the search sphere intersects P_2, too. The mapped region is a larger annulus in P_1 (the shaded region in P_1) and an annulus in P_2 (the shaded region in P_2). The enlarged portion of the annulus in P_1 consists of two smaller annulus regions, one expanding towards the center of P_1 (arrow A) and the other expanding outwards P_1 (arrow B). Each of the smaller annulus corresponds to a continuous range of points stored in the leaf nodes of the index and they adjoin the range retrieved last time. Therefore, the algorithm performs a backward range search (arrow A') and a forward range search (arrow B') to retrieve the new data points, starting from the boundaries of the last range retrieval. For P_2, the mapped region is just an annulus expanding towards the center of P_2 (arrow C) because Q lies outside of P_2. This corresponds to a range search backwards (arrow C') on the index. All the newly retrieved points are



iDistance Techniques, Figure 2 KNN search of the iDistance (continued)

compared with the maintained k candidates and always the k nearest ones to Q are maintained. After this, r continues to be increased iteratively until the termination condition is satisfied.

In the algorithm, r_0 and Δr are two other tunable parameters. r_0 can be set close to an estimation of the final search radius so that less iterations are needed to reach the final search radius, or r_0 can be simply set as 0. Δr is a small value so that not much unnecessary data points are retrieved. It can be set as 1% of an estimation of the final search radius. An implementation of the iDistance in C by Rui Zhang is available at http://www.csse.unimelb.edu.au/~rui/code.htm.

Key Applications

The iDistance technique is mainly designed for processing kNN queries in high-dimensional spaces. Typical applications are similarity search on multimedia data and data mining. Multimedia data such as images or videos are usually represented by features extracted from them including color, shape, texture, etc [2]. The number of features is large. Therefore, retrieving similar objects translates to kNN queries in high-dimensional spaces. In data mining or data warehouse, data records have large number of attributes. Finding similar objects also requires kNN search in high-dimensional spaces. In general, similarity search on any objects that can be represented by multi-dimensional vectors can take advantage of this technique.

Future Directions

The mapping function of the iDistance index relies on a fixed set of reference points. If the data distribution changes much, the kNN search performance may deteriorate. Then the iDistance index has to be rebuilt in order to keep the high performance. One open problem is the possibility to find a dynamic mapping function that can adapt to the change of data distributions automatically, so that the high performance is always kept without rebuilding the index.

Cross References

- ▶ Distance Metrics
- ▶ Indexing, High Dimensional
- ▶ Indexing of Moving Objects, B^x-Tree
- ▶ Nearest Neighbor Query
- ▶ Nearest Neighbors Problem

Recommended Reading

Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is nearest neighbors meaningful? In: Proceedings of International Conference on Database Theory (1999)

Faloutsos, C., Equitz, W., Flickner, M., Niblack, W., Petkovic, D., Barber, R.: Efficient and Effective Querying by Image Content. J. Intell. Inf. Syst. **3**(3), 231–262 (1994)

Jagadish, H.V., Ooi, B.C., Yu, C., Tan, K.-L., Zhang, R.: iDistance: An Adaptive B+-tree Based Indexing Method for Nearest Neighbor Search. ACM Trans. Data Base Syst. **30**(2), 364–397 (2005)

Weber, R., Schek, H., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: Proceedings of International Conference on Very Large Data Bases (1998)

Yu, C., Ooi, B.C., Tan, K.-L., Jagadish, H.: Indexing the distance: an efficient method to KNN processing. In: Proceedings of International Conference on Very Large Data Bases (2001)

Image

► Oracle Spatial, Raster Data

Image Acquisition

► Photogrammetric Sensors

Image Analysis

► Data Acquisition, Automation

Image Compression

LEI SHA
Department of Physiology and Biophysics,
Mayo Clinic, Rochester, MN, USA

Synonyms

Data compression; Lossless image compression; Lossy image compression

Definition

Image compression is the process of encoding digital image information using fewer bits than an unencoded representation would use through use of specific encoding schemes.

Historical Background

Uncompressed image data is large in file size. To store or transmit the uncompressed image requires considerable storage capacity and transmission bandwidth. For example, for a uncompressed color picture file of $6{,}000 \times 4{,}800$ pixel (good for a mere 10×8 in. print at 600 dpi), number of bytes required to store it is 85 MB. Despite rapid progress in mass-storage density, processor speeds, and digital communication system performance, demand for image storage capacity and image transmission bandwidth constantly outstrip the technical progress. Image compression is therefore essential for image storage and transmission in most cases. In geographic information systems (GIS), images are generally very large. A simple uncompressed world map (3×2.5 ft. in size, for example) will require more than 1 GB storage space. To retrieve the simple world map remotely over Internet with an Internet transmission speed of 1 Mbps, it would take more than 2 h. GIS images are generally much larger than a simple world map. Therefore, how to compress and store GIS images (as well as fast access/retrieval) is a hot research topic.

Scientific Fundamentals

Principle of Image Compression

Images can be compressed, because images have some degree of redundancy. For example, one of the common characteristics of most images is that the neighboring pixels are correlated and, therefore, the information is redundant among the neighboring pixels. In still images, there are two type of redundancy: spatial redundancy and spectral redundancy.

Spatial redundancy refers to the correlation between neighboring pixels. This is the redundancy due to patterning, or self–similarity within an image. Spectral redundancy is the redundancy occurring on the correlation between different color planes or spectral bands. The objective of general image compression is to reduce redundancy of the image data in order to be able to store or transmit data in an efficient form.

There are two classes of image compression methods to reduce the redundancy: lossless image compression and lossy image compression. In lossless compression, the reconstructed image from the compression is identical to the original image. However, lossless compression can only achieve a modest amount of compression because the the redundant information is retained. Lossy image compression can achieve a high compression rate but an image reconstructed following lossy compression is degraded relative to the original, because the redundant information is completely discarded.

Lossless Image Compression

Lossless image compression uses a class of data compression algorithms that allows the exact original data to be reconstructed from the compressed data. Lossless compression is used when it is important that the original and the decompressed data be identical, or when no assumption can be made on whether a certain deviation is uncritical. Some image file formats, notably PNG (Portable network graphics), use only lossless compression, while others like TIFF (Tagged image file format) and MNG (Multiple-image Network Graphics) may use either lossless or lossy methods.

Most lossless compression programs use two different kinds of algorithms: one that generates a statistical model for the input data, and another that maps the input data to bit strings. The two type of lossless compression algorithmsare statistical modeling algorithms (for text or text-like binary data) and encoding algorithms to produce bit sequences. Statistical modeling algorithms include the Burrows–Wheeler transform (block sorting preprocessing that makes compression more efficient), LZ77 (used by

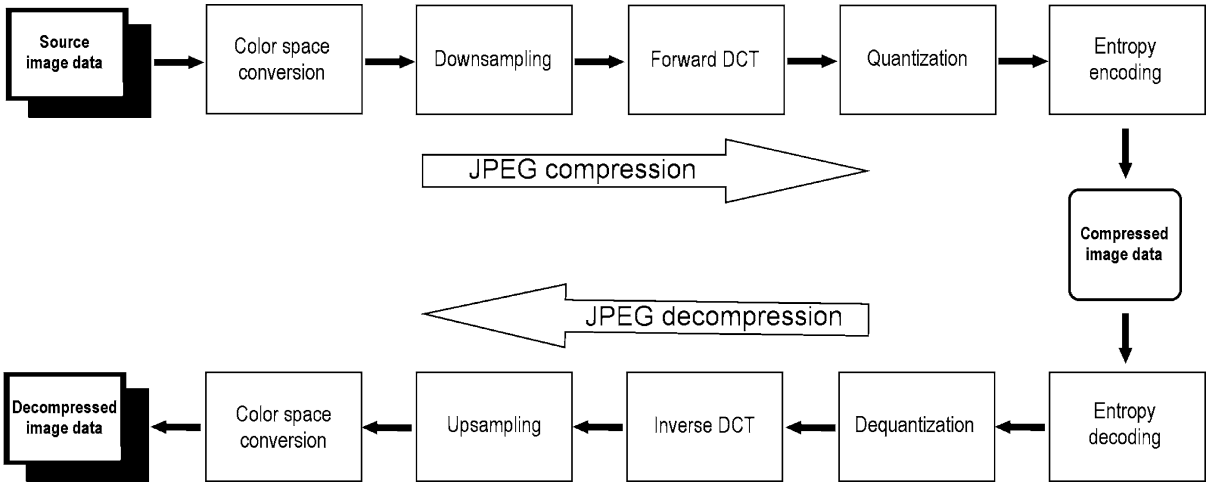


Image Compression, Figure 1 Compression and decompression flow in a popular lossy image compression. DCF (Discrete Cosine Transform)

DEFLATE) and LZW. Encoding algorithms include Huffman coding (also used by DEFLATE) and arithmetic coding.

Lossless data compression algorithms cannot guarantee compression of all input data sets. In other words, for any lossless data compression algorithm there will be an input data set that does not get smaller when processed by the algorithm.

Here are some popular lossless image compression format: adaptive binary optimization (ABO), GIF (Graphics Interchange Format, after lossy reduction of color depth), PNG, JPEG-LS (can be lossless/near-lossless), JPEG 2000 (includes lossless compression method), and TIFF (using Lempel-Ziv-Welch lossless compression).

Lossy Image Compression

A lossy compression method is one where compressing data and then decompressing it retrieves data that may well be different from the original, but is "close enough" to be useful in some way.

There are basically four lossy image compression methods:

- Transform coding: transform coding uses a transformation that exploits peculiar characteristics of the signal, increasing the performance of a scheme such as Huffman or arithmetic coding.
- Wavelet compression: wavelet compression is based on wavelet functions. It also adopts transform coding. The basic idea of this coding scheme is to process data at different scales of resolution. Wavelet compression has a very good scalability for macrostructure and microstructure as the compression is achieved using two versions of the picture as a different scaling of the same prototype function called the mother wavelet or wavelet basis.
- Vector quantization: this method is better known as the dictionary method. A dictionary is a collection of a small number of statistically relevant patterns. Every image is encoded by dividing it into blocks and assigning to each block the index of the closest codeword in the dictionary.
- Fractal compression: the basic idea of fractal compression is "self-vector-quantization," where an image block is encoded by applying a simple transformation to one of the blocks previously encoded. Algorithms based on fractals have very good performance and high compression ratios (32:1 is not unusual), but their use can be limited by the extensive computation required.

Some popular lossy image compression formats are: compression via fractal compression, JPEG, JPEG2000, JPEG's successor format that uses wavelets, wavelet compression, Cartesian perceptual compression, DjVu, and ICER (Incremental cost-effectiveness ratio).

Figure 1 is an example of compression and decompression flow in a popular lossy image compression (JPEG).

The advantage of lossy methods over lossless methods is that in some cases a lossy method can produce a much smaller compressed file than any known lossless method, while still meeting the requirements of the application. Lossy methods are most often used for compressing images. Lossily compressed still images are often compressed to a tenth of their original size.

When a user acquires a lossily compressed file, (for example, to reduce download time) the retrieved file can be quite different to the original at the bit level while being indistinguishable to the human eye for most practical purposes. Many lossy image compression methods focus on the

idiosyncrasies of the human anatomy, taking into account, for example, that the human eye can see only certain frequencies of light. Although lossy image compression can achieve a very compact compression, high image compression can cause some artifacts.

Four types of artifacts may occur: blocking, blurring, ringing effect, and texture deviation. Between lossless image compressions and the varying degrees of lossy image compressions, there has to be some balance/trade-off for any particular mission. When considering using an image compression method, scalability of the compressed image using the image compression method is also very important. Especially, in the GIS field, scalability is important for viewing an image with different views, which provide variable quality access to databases.

Key Applications

In the GIS field, both lossless and lossy image compression methods are used depending on the differing demands of the various projects. JBIG(Joint Bi-level Image experts Group), for example, is a format using a lossless compression of a bilevel image. It can be used for coding grey scale and color GIS images with limited numbers of bits per pixel. It can offer between 20 and 80% improvement in compression (about 20 to 1 over the original uncompressed digital bit map). Formats based on JBIG can be used for large GIS binary images in digital spatial libraries.

However, in some projects, a high image compression rate and more information are needed. High resolution satellite images with more geometry and information content, for example, require a compression rate well above a factor of 10, and lossy image compression methods become necessary. In this case, JPEG/JPEG2000 or other image compression methods based on wavelet compression are widely used.

With many researchers' contributions to the spatial image compression technique, the spatial image compression field is rapidly moving forward. The following are a few new developments on spatial image compression research. Eugene Ageenko and Pasi Franti [9] proposed a compression method based on JBIG aimed at compression of large binary images in digital spatial libraries. The simulation results of their image compression method showed that their proposed method allows dense tiling of large images down to 50×50 pixels versus the 350×350 pixels that is possible with JBIG without sacrificing the compression performance. It will allow partial decompression of large images far more efficiently than if JBIG was applied. For clusters larger than 200×200 pixels, the method improves JBIG by about 20%. Renato Pajarola and Peter Widmayer [6,8] developed an image compression method for spatial search with the consideration of efficiency on decompression and retrieval of spatial images. The compression algorithm they proposed is oriented toward spatial clustering and permits decompression from local information alone. They demonstrated experimentally that the Hibert compression ratio typically is competitive with well known compression algorithms such as lossless JPEG or CALIC (Context-based Adaptive Lossless Image Coding). They implemented an experimental image database that provides spatial access to compressed images and can be used as a texture server to a real-time terrain visualization system. Hassan Ghassemian [11] developed an onboard satellite image compression method by object-feature extraction. Ghassemian stated that "recent developments in sensor technology make possible Earth observational remote sensing systems with high spectral resolution and data dimensionality. As a result, the flow of data from satellite-borne sensors to earth stations is likely to increase to an enormous rate." Therefore, "a new onboard unsupervised feature extraction method that reduces the complexity and costs associated with the analysis of multispectral images and the data transmission, storage, archival and distribution"was investigated. He developed an algorithm to reduce data redundancy by an unsupervised object-feature extraction process. His results showed an average compression of more than 25, the classification performance wasimproved for all classes, and the CPU time required for classification reduced by a factor of more than 25. Jiri Komzak and Pavel Slavik [5] in their paper described a general adaptive tool usable for compression and decompression of different types of data in GIS. Their approach makes it possible to easily configure compressor and decompressor and use different compression methods. There is also a very important possibility using their approach of optionally turning on and off lossy transitions during the compression process, and in such a way as to change the data loss even for each symbol. Jochen Schiewe [10] in his paper, and Ryuji Matsuoka and colleagues [7] in their paper took time comparing the effects of lossy data compression techniques on the geometry and information content of satellite imagery. They compared different image compression methods and outlined the advantages and disadvantages of a few popular methods. For example, Matsuoka concluded that, on comparison of lossy JPEG compression and lossy JPEG 2000 compression in performance, it was confirmed that lossy JPEG 2000 compression is superior to lossy JPEG compression in color features. However, lossy JPEG 2000 compression does not necessarily provide an image of good quality in texture features. Their research provided very useful information for GIS designers to use in picking image compression formats in their system.

As in the digital image era, the development of image compression techniques in GIS is progressing at a rapid pace.

Future Directions

The development of image compression technique in GIS is toward more efficient algorithms with high compression ratios, low magnitudes of the error introduced by the encoding (lossy image compression), and fast and scalable coding and retrieval.

Cross References

► Map Generalization

Recommended Reading

Wikipedia. Image compression. http://en.wikipedia.org/wiki/Image_compression

Saha, S.: Image compression—from DCT to wavelets: a review. http://www.acm.org/crossroads/xrds6-3/sahaimgcoding.html

Motta, G., Rizzo, F., Storer, J. A.: Digital image compression. In: Ralston, A., Reilly, E.D., Hemmendinger, D. (eds.) The Encyclopedia of Computer Science, 4th edn. Grove Dictionaries (2000)

GIS Standards. http://fortboise.org/maps/GISstandards.html

Komzak, J., Slavik, P.: Architecture of system for configurable GIS data compression. http://kmi.open.ac.uk/people/jiri/publ/wscg2002.pdf

Pajarola, R., Widmayer, P.: Spatial indexing into compressed raster images: how to answer range queries without decompression. http://www.ifi.unizh.ch/vmml/admin/upload/MMDBMS96.pdf

Matsuoka, R., Sone, M., Fukue, K., Cho, K., Shimoda, H.: Quantitative analysis of image quality of lossy compression images. http://www.isprs.org/istanbul2004/comm3/papers/348.pdf

Pajarola, R., Widmayer, P.: An image compression method for spatial search. IEEE Trans. Image Process. **9**, (2000)

Ageenko, E.I., Frnti, P.: Compression of large binary images in digital spatial libraries. Comput. Graphics **24**, 91–98 (2000)

Schiewe, J.: Effect of lossy data compression techniques of geometry and information content of satellite imagery. IAPRS. **32**, 540–544 (1998)

Hassan Ghassemain. On-board satellite image compressiaon by object-feature wxtraction. International Atchieves of Photogrammertry Remote Sensing and Spatial Information Sciences.**35**, 820–825 (2004)

Image Mining, Spatial

SUDHANSHU SEKHAR PANDA
Department of Science, Engineering, and Technology, Gainesville State College, Gainesville, GA, USA

Synonyms

Visual data mining; Image pattern; Object recognition; Association rules: image indexing and retrieval; Feature extraction

Definition

Image mining is synonymous to data mining concept. It is important to first understand the data mining concept prior to image mining. Data mining is a set of techniques used in an automated approach to exhaustively explore and establish relationships in very large datasets. It is the process of analyzing large sets of domain-specific data and subsequently extracting information and knowledge in a form of new relationships, patterns, or clusters for the decision-making process [1]. Data mining applications are of three-level application architecture. These layers include applications, approaches, and algorithms and models [2]. The approaches of data mining are association, sequence-based analysis, clustering, estimation, classification, etc. Algorithms and models are then developed based on the dataset type to perform the data mining. At that point, the ill-data points are extracted from the dataset. Similarly, image mining is a set of tools and techniques to explore images in an automated approach to extract semantically meaningful information (knowledge) from them. A single image is a collection of a set of data. Therefore, image mining techniques are far more complicated than the data mining techniques, but data mining tools and techniques could be replicated for the purpose of image mining. However, there should not be any misnomer that image mining is just a simple extension of data mining applications or just a pattern recognition process [3].

Historical Background

Images are abundant in the present multimedia age. There has been tremendous growth in significantly large and detailed image databases due to the advances in image acquisition and storage technology [4]. The World Wide Web is also regarded as the largest global image repository. Every day, a large number of image data are being generated in the form of satellite images, aerial images, medical images, and digital photographs. These images, if analyzed with proper tools and techniques, can reveal useful information to the users. However, there is general agreement that sufficient tools are not available for analysis of images [3]. Effective identification of features in the images and their proper extraction are some of the important problems associated with image analysis. One of the difficult tasks is to know the image domain and obtain a priori knowledge of what information is required from the image. Therefore, complete automation of an image mining process cannot be performed. Image mining deals with the extraction of implicit knowledge, image data relationship, or other patterns not explicitly stored in the image databases. It is an interdisciplinary endeavor that essential-

ly draws upon expertise in computer vision, image processing, image retrieval, data mining, machine learning, database, and artificial intelligence [5].

Scientific Fundamentals

Issues in Image Mining

Image mining, by definition deals with the extraction of image patterns from a group of images. Image mining is not exactly similar to low-level computer vision and image processing techniques. The computer vision and image processing techniques are applied to understand and/or extract specific features from a single image where the objective of image mining is to discover image patterns in a set of collected images [3]. As mentioned earlier, image mining is not just another data mining technique.

Difference Between Image Mining and Data Mining

In a relational database, the data are well defined. While analyzing a relational database of annual precipitation and runoff, a value of 15 cm rainfall in May, 2006 has its own absolute meaning. It suggests that there is 15 cm of rainfall in May, 2006. But in an image database (which deals with mostly the digital gray values), a gray value of 15 representing a moist soil in one image could be 25 in another image based on all other pixels in the image, i. e., if the second image is taken with a brighter light condition than the first one, all digital gray values will end up in the second image as higher numbers. Therefore, image mining deals with relative values, while data mining always deals with absolute values.

Image mining always deals with spatial issues as images represent spatial references. Satellite images and aerial photographs are representing spatial locations of earth. Similarly, when analyzing a tumor in an image of a body part, the tumor position is analyzed on a spatial basis, too, i. e., gauging how far is the tumor from the lungs, intestine, or brain center, etc. In case of a digital image of a person, the spatial position of different features should be analyzed. But in case of data mining, spatial consideration may not be required. Consequently, image miners try to overcome the spatial issue problem by extracting position-independent features from images before attempting to mine useful patterns from the images [3].

Often image mining deals with multiple interpretations of images of same location (Fig. 1). For example, temporal changes of the land use/land cover of an area need to be analyzed differently than the plain data mining or feature extraction techniques, to extract the change information in images. Therefore, traditional data mining techniques of associating a pattern to a class interpretation may not work with image mining. A new set of algorithms is needed to cater to discover useful patterns from images [3].

Image Mining Techniques

Image mining may use data from a image databases (Fig. 1). Basically, images are stored with some description against a particular image. Again images are nothing but some intensity values, which figure the image in terms of color, shape, texture, etc. The mining task is based on using such information contained in the images [6]. The following are some of the image mining techniques frequently used by image miners:

Image Mining, Spatial, Figure 1 Set of aerial images (part of image database) of a quarter size crop field used for crop yield analysis using image mining techniques. **a** NW29 quarter aerial image in August 1997. **b** W29 quarter aerial image in August1997 (few days apart). **c** W29 quarter aerial image in August 1998 (similar time of the month as first one). Note: The images quality and patterns in images change as a factor of difference in image acquisition set up and other environmental factor

1. Image acquisition
2. Image enhancement and restoration
3. Object recognition
4. Image indexing and retrieval
5. Image segmentation and feature extraction
6. Representation and description
7. Association rules mining

Among these tasks, image indexing and retrieval and image segmentation and extraction are two of the principal tasks in image mining. They are categorized as content-based techniques [7]. Some of the other image retrieval techniques are categorized under description based retrieval techniques.

1) ***Image acquisition***: In image mining, the understanding of the image acquisition setup is very much essential. As described earlier, similar spatial images vary in look with different image acquisition setups, such as lighting intensity, use of sensor type, and radiometric factor (in case of satellite and aerial images), etc. While comparing a set of similar spatial images, it is necessary to bring them into similar image acquisition setup. Color calibration is a technique that could be employed for this. For satellite images, radiometric correction techniques can be used. ENVI software (ITT Visual Information Solutions, Boulder, CO) has a good technique for radiometric corrections of satellite images.
2) ***Image enhancement and restoration***: Image enhancement and restoration is one of the simplest image processing techniques [8]. This technique is used to highlight only the important area of interest in an image and ignoring irrelevant areas. Contrast enhancement is an example of image enhancement and restoration. Log transformations, power-law transformations, piecewise-linear transformation, histogram equalization, image subtraction, image averaging, smoothing by different spatial filters, sharpening by different spatial filters, Fourier transformation, noise reduction using different filtering techniques, geometric transformations, etc., are some of the algorithms used for image enhancement and restoration. Books [8,9,10] on image processing will provide with the algorithms and working procedure for all these techniques.
3) ***Object recognition***: Object recognition is one of the major tasks in image mining. With the help of a known object models, an object recognition system finds objects in the real world from an image. Automatic machine learning and meaningful information extraction can be comprehended by training (supervised training) the machine with some known objects [3]. Machine learning algorithms could be accessed from several neural networks books [11]. The object recognition model is initially provided with a set of objects and labels (those are expected to be found in the tested image) to become trained and the model for object recognition assigns correct labels to regions, or a set of regions, in the image. Models of known objects and labels are usually provided by human (teacher for the model).

   In general, an object recognition module consists of four components, such as 1) model database, 2) feature detector, 3) hypothesizer, and 4) hypothesis verifier. The model database contains all the models known to the system that became trained. The models also contain several features or labels that describe the objects in the image being tested. The detected image primitive features in the gray scale (pixel) level help the hypothesizer to assign likelihood to the objects in the image. Finally, the verifier uses the models to verify the hypothesis, refine the object likelihood, and label the features in the tested image [3,5]. Thus the correct objects are recognized by the machine learning model. Bonet [12] designed a system that processes an image into a set of "characteristic maps" to locate a particular known object in an image or set of images.
4) ***Image indexing and retrieval***: Image mining is incomplete without image retrieval process. Images can be retrieved based on features, such as color, texture, shape, size, or the spatial location of image elements; logical features like objects of a given type or individual objects or persons; or abstract attributes or features. Three query schemas for image retrieval are described by Rick Kazman and Kominek [13]. They include query by associate attributes, query by description, and query by image content. In recent times, the content-based image retrieval techniques are used most [14]. Content-based image retrieval is based on three processes, such as, visual information extraction, image indexing and retrieval system application [15]. IBM's QBIC system [16] is perhaps the best known of all image content retrieval systems in commercial application. The system offers image retrieval by individual parameter or any combination of color, texture, shape, or text keyword along with R-tree indexing feature to improve search efficiency [3]. There are many other software commercially available for image indexing and retrieval. A simple web search with "image indexing software" can provide insight to them.
5) ***Image segmentation and feature extraction***: Image segmentation and feature extraction are closely similar to image indexing and retrieval process. Image segmentation can be performed on a single image or a batch of images based on batch processing. Segmentation procedures partition an image into constituent parts or objects. They are mostly feature-based, i. e., by analyz-

ing image features such as color, texture, shape, and size, etc. Intelligently segmenting an image by content is an important way to mine valuable information from large image collection. Several image segmentation techniques are used now a day. New techniques are evolving with the advancement in computer programming. Some of the image segmentation techniques are ISODATA, K-means, Fuzzy C-means, region growing, self organizing map (SOM) neural classification, Bayesian classifier based hierarchical classification, WARD-minimum variance method, maximum likelihood, etc. They are mostly classified into two main categories, such as supervised and unsupervised. Unsupervised segmentation techniques are widely pursued because it does not need any a priori knowledge regarding the image under segmentation. However, the better the segmentation algorithm, there is a good probability of correct feature extraction. Image segmentation in multimedia is a priority now. They are although similar to the normal segmentation procedures used in remote sensing image analysis but work differently. Zaiane and Han [4] have developed MM-Classifier, a classification module embedded in the MultiMedia Miner which classifies multimedia data, including images. Wang and Li [17] have proposed a new classifier technique, IBCOW (image-based classification of objectionable websites) to classify websites based on objectionable or benign image content.

Artificial intelligence has a big role in image segmentation now. Artificial neural network (ANN) and self-organizing maps are being used to accurately segment images for several applications [18].

6) ***Representation and description***: In image mining, these steps always follow the output of segmentation stage. The output of segmentation is usually the raw pixel data constituting either boundary of a region or all the points of a region. Converting these output data to a form suitable for computer processing is essential [8]. The decision taken to represent the classified pixels as boundary pixels or pixels of a region is the representation stage. Finally, description of region based on real world completes the image classification process in image mining.
7) ***Association rule mining***: There is a two steps approach in a typical association rule mining algorithm. The first step finds all large item sets that meet the minimum support constraint while the second step generates rules from all the large item sets that satisfy the minimum confidence constraint [3]. Association rule mining generated rules works well in image mining than some user-defined thresholds [3]. Association rule mining is used in data mining to uncover interesting trends, patterns, and rules in large datasets. Association rule mining has found its application in image mining consisting of large image databases [19,20]. The first approach in association rule mining in image mining is to mine from large collections of images alone and the second approach is to mine from the combined collections of images and associated alphanumeric data [20]. Simon Fraser University has developed a prototype called Multimedia Miner, and one of its major modules is called MM-Associator, which uses a three-dimensional visualization to explicitly display the associations [4].

Key Applications

Image mining is a recent phenomenon. It is still in its infancy. It has lot of scope in multimedia research.

World Wide Web

Web image mining is a very new concept. With the advent of Google, Yahoo, AltaVista, and other web search engines, it is possible to search articles and images with text-based search method. But searching images from World Wide Web using a visual approach is a challenge. However, web image mining techniques are progressive steps towards that [21].

Multimedia

As discussed in the previous text, multimedia has the largest application of image mining. They deal with huge databases of images, animations, audios, videos, and other text databases. Retrieving information from these databases is getting easier with the use of several image mining software [22]. Several other software programs developed for this purpose are mentioned in the main text.

Healthcare

Modern health care generates huge amounts of digital data, including an overabundance of images. They include SPECT images, DNA micro-array data, ECG signals, clinical measurements, such as blood pressure and cholesterol levels, and the description of the diagnosis given by the physician interpretation of all these data. With the advent of image mining techniques, most of their processes are becoming automated.

Engineering and Construction Industry

Architectural designs using AutoCAD generates huge amount of digital data. Visual data generated in engineering and construction industries are 3D-models, build-

ing component databases, images of the environment, text descriptions and drawings from the initial and evolved design requirements, financial and personnel data, etc. They could profit from image mining.

Geotechnology

It is obvious that the geotechnology industry representing GIS, remote sensing, and GPS application deals with highest amount of image databases. These databases are generated in any field of application, such as agriculture, forestry, environmental science, geosciences, to name a few. Image mining is a premium tool in geotechnology field. VisiMine (Insightful Corporation, Seattle, WA) is an image mining software (a search engine) used for analyzing image databases. It is designed for satellite imagery and aerial photos analysis. Visimine provides a comprehensive workbench for image information mining by integrating state-of-the-art statistics, data mining, and image processing to extract information and locate images from potentially huge databases.

Police and Security

The organizations dealing with security are swamped with image databases. Image mining techniques are having a major use in their activities.

Future Directions

As image mining is still in its infancy, there is a huge scope for its development. Future research should emphasize on these few factors: 1) design powerful query language (a la GIS query expression) to access image data from image database, 2) explore new visual discovery algorithm to locate unique characteristics present in image data, and 3) devise automated image mining techniques based on content based image retrieval techniques.

Cross References

► Co-location Patterns, Algorithms
► Gaussian Process Models in Spatial Data Mining

Recommended Reading

1. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, San Francisco, CA (2001)
2. Moxon, B.: Defining Data Mining: The Hows and Whys of Data Mining, and How it Differs from Other Analytical Techniques. DBMS Data Warehouse Supplement, Miller Freeman, Inc., San Francisco, CA (1996)
3. Zhang, J., Hsu, W., Lee, M.L.: An information-driven framework for image mining. In: Proceedings of 12th International Conference on Database and Expert Systems Applications (DEXA), Munich, Germany (2001)
4. Zaiane, O.R., Han, J.W.: Mining multimedia data. CASCON'98: Meeting of Minds, pp 83–96, Toronto, Canada, November 1998
5. Burl, M.C.: Mining for image content. In Systemics, Cybernetics, and Informatics / Information Systems: Analysis and Synthesis, Orlando, FL (July 1999)
6. Jain, L. C., Ghosh, A.:Evolutionary Computation in Data Mining. Springer, New York, NY (2005)
7. Rui, Y., Huang, T., Chang, S.: Image retrieval: current techniques, promising directions and open issues. J. Vis. Commun. Image Represent. **10**(4): 39–62 (1999)
8. Gonzalez, R. C., Woods, R. E.: Digital Image Processing. 2nd edn. Pearson Education, New Delhi, India (2002)
9. Jain, A.K.: Fundamentals of Digital Image Processing. Prentice Hall, New York, NY (1998)
10. Pitas, I.: Digital Image Processing Algorithms. Prentice Hall, New York, NY (1993)
11. Haykin S.: Neural Networks a Comprehensive Foundation. Prentice Hall Inc., New York, NY (1999)
12. Bonet, J.S.D.: Image Reprocessing for Rapid Selection in "Pay attention mode". MIT Press, Boston, MA (2000)
13. Kazman, R., Kominek, J.: Information organization in multimedia resources. Proceedings of the 11th annual international conference on Systems documentation, pp149–162 (1993)
14. Foschi, P. G., Kollipakkam, D., Liu, H., Mandvikar, A.: Feature extraction for image mining (2007) www.public.asu.edu/~huanliu/papers/mis02.pdf. Accessed on 01.10.07
15. Rui, Y., Huang, T.S.: Image retrieval: past, present and future. Invited paper in: International Symposium on Multimedia Information Processing, Taipei, Taiwan, December 11–13, 1997
16. Niblack, W., Barber, R.: The QBIC project: querying images by content using color, texture and shape. In: Proc. SPIE Storage and Retrieval for Image and Video Databases, February 1994
17. Wang, J.Z., Li, J.:System for screening objectionable images using daubechies' wavelets and color histograms. Interactive Distributed Multimedia Systems and Telecommunication Services, Proceedings of the Fourth European Workshop (IDMS'97) (1997)
18. Panda, S.: Data mining application in production management of crop. Ph.D. Dissertation, North Dakota State University, Fargo, North Dakota, USA (2003)
19. Megalooikonomou, V., Davataikos, C., Herskovits, E. H.: Mining lesion-deficit associations in a brain image database. KDD, San Diego, CA, USA (1999)
20. Ordonez, C., Omiecinski, E.: Discovering association rules based on image content. In: Proceedings of the IEEE Advances in Digital Libraries Conference (ADL'99) (1999)
21. Yanai, K.: Web image mining toward generic image recognition. In: Proceedings of the Twelfth International World Wide Web Conference (2003)
22. Missaoui, R., Sarifuddin, M., Vaillancourt, J.: Similarity measures for efficient content-based image retrieval. In: IEE Proceedings – Visual Image Signal Process. 152: 6, December 2005

Image Pair

► Photogrammetric Methods

Image Pattern

► Image Mining, Spatial

Image Station

► Intergraph: Real Time Operational Geospatial Applications

Image Triplet

► Photogrammetric Methods

Imagery Conflation

► Conflation of Geospatial Data

Immediate Response Zone

► Emergency Evacuations, Transportation Networks

Imprecision and Spatial Uncertainty

MICHAEL F. GOODCHILD
Department of Geography, University of California Santa Barbara, Santa Barbara, CA, USA

Synonyms

Quality, spatial data; Accuracy, spatial; Accuracy, map; Error propagation

Definition

Spatial uncertainty is defined as the difference between the contents of a spatial database and the corresponding phenomena in the real world. Because all contents of spatial databases are representations of the real world, it is inevitable that differences will exist between them and the real phenomena that they purport to represent. Spatial databases are compiled by processes that include approximation, measurement error, and generalization through the omission of detail. Many spatial databases are based on definitions of terms, classes, and values that are vague, such that two observers may interpret them in different ways. All of these effects fall under the general term of spatial uncertainty, since they leave the user of a spatial database uncertain about what will be found in the real world. Numerous other terms are partially synonymous with spatial uncertainty. Data quality is often used in the context of metadata, and describes the measures and assessments that are intended by data producers to characterize known uncertainties. Vagueness, imprecision, and inaccuracy all imply specific conceptual frameworks, ranging from fuzzy and rough sets to traditional theories of scientific measurement error, and whether or not it is implied that some true value exists in the real world that can be compared to the value stored in the database.

Historical Background

Very early interest in these topics can be found in the literature of stochastic geometry (Kendall, 1961), which applies concepts of probability theory to geometric structures. An early paper by Frolov and Maling (1969) analyzed the uncertainties present in finite-resolution raster representations, and derived confidence limits on measures such as area, motivated in part by the common practice of estimating measures of irregular patches by counting grid cells. Maling's analysis established connections between the spatial resolution of the overlaid raster of cells and confidence limits on area estimates. Maling's book (Maling, 1989) was a seminal venture into the application of statistical methods to maps, and helped to stimulate interest in the topic of spatial uncertainty. The growth of geographic information systems (GIS) provided the final impetus, and led to the first research initiative of the new US National Center for Geographic Information and Analysis in 1988, on the topic of accuracy in spatial databases (Goodchild and Gopal, 1989).

The notion that spatial databases could be treated through the application of classical theories of measurement error soon proved too limiting, however. The definitions of types that are used in the compilation of maps of soil class, vegetation cover class, or land use are clearly open to interpretation, and such maps must be regarded as to some degree subjective and outside the normal bounds of scientific replicability. Concepts of fuzzy and rough sets were explored by researchers interested in these issues (Fisher and Unwin, 2005). While the definition of a given class may be vague, it is nevertheless helpful to think about degrees of membership in the class. For example, researchers interested in developing plain-language interfaces to GIS found that prepositions such as "near" had vague meanings that could be represented more formally through membership functions. This approach resonated well with the move in the early 1990s to introduce theories of linguistics and cognition into GIS research.

By the end of the 1990s the literature on spatial uncertainty had grown to include several distinct theoretical frameworks, including geostatistics, fuzzy sets, rough sets, and spatial statistics. Zhang and Goodchild (2002) published a synthesis, framed within the fundamental dichotomy between discrete objects and continuous fields that under-

lies much of GIScience. Research continues, particularly on such topics as spatial uncertainty in digital terrain data.

Scientific Fundamentals

In the classical theory of measurement, an observed value z' is distorted from its true value z by a series of random effects. If these effects are additive, the distortion $\delta z = z' - z$ is expected to follow a Gaussian distribution, and each observed measurement is interpreted as a sample drawn from that distribution. The mean of the distribution is termed the bias or systematic error, and the root mean square of δz is termed the standard error. The standard deviation of δz with respect to its own mean is often termed precision, and a biased measurement device is thus said to be possibly precise but not accurate. However, precision can also refer to the number of numerical digits used to report a measurement, and imprecision is used in several ways in the literature on spatial uncertainty.

This analysis extends readily to measurement of position in two or three dimensions, and thus to measurements made by such technologies as the Global Positioning System (GPS), where the multivariate Gaussian distribution is widely used to characterize positional uncertainty. Measurement errors in the two horizontal dimensions are commonly found to have equal variance, but errors in the vertical dimension typically have very different variance; and measurement errors in all three dimensions are commonly found to be uncorrelated.

This classical theory has been developed extensively within the discipline of surveying, under the rubric of adjustment theory, in order to understand the effects that errors in raw measurements may have on the inferred locations of items of interest. For example, errors in the measurement of bearing, elevation, and range will translate into errors in the inferred positions of the objects of the survey. Complications arise when closed loops are surveyed in the interests of reducing errors, which must then be allocated around the loop in a process known as adjustment. This body of theory has not had much influence on spatial databases, however, outside of the domain of traditional surveying.

Any spatial database will consist of large numbers of measurements. For example, a remotely sensed image may contain millions of pixels, each containing several measurements of surface reflectance. Although measurements made by simple devices such as thermometers can reasonably be assumed to have statistically independent errors, this is almost never true of data compiled across geographic space. Instead, strong and mostly positive correlations are observed between data values that are close together in space. These correlations may be induced by the production process, when many data values inherit the errors in a smaller number of nearby measurements through various forms of interpolation, or through the measurements themselves, which are distorted by effects that operate across areas of space. Such correlations are generally known as spatial dependence or spatial autocorrelation.

This tendency turns out to be quite useful. For example, consider a curved segment of a street, recorded in a spatial database as a sequence of coordinate pairs. Assume a measurement error of 10 m, not unreasonable in today's street centerline databases. If each point was independently disturbed by 10 m, the result would be impossibly and unacceptably erratic, and the segment's length as determined from the database would be severely overestimated. Instead, positive correlation between nearby errors ensures that the general shape of the street will be preserved, even though its position is disturbed. Similar arguments apply to the preservation of slopes in disturbed elevation models, and to many other examples of spatial data.

Several authors have drawn attention to an apparent paradox that follows from this argument. Consider a straight line, such as a straight segment of a street or property boundary, and suppose that the endpoints are disturbed by measurement error. If the disturbances are independent with known distributions, standard errors can be computed at any point along the line; and are found to be in general smaller away from the endpoints. If the disturbances have perfect positive correlation then standard errors are constant along the line; if they have identical and independent distributions then standard error is least at the midpoint where it is equal to 0.707 times the endpoint standard error; and if errors have perfect negative correlation then standard error will drop to zero at one intermediate point. Kyriakidis and Goodchild (2006) have generalized this problem to several other instances of linear interpolation. In practice, however, the straight line may itself be a fiction, and deviations of the truth from the straight line will tend to rise away from the endpoints, more than compensating for this effect.

Geostatistics (Goovaerts, 1997) provides a comprehensive theoretical framework for modeling such spatial autocorrelation of errors. Variances between nearby errors are expected to increase monotonically up to a distance known as the range, beyond which there is no further increase. The variance at this range is termed the sill, and corresponds to the absolute error of the database; however relative error is less over distances shorter than the range, and near zero over very short distances. Mathematical functions provide models of the monotonic increase of variance with distance.

Such models provide a convenient and powerful basis for exploring the effects of errors in applications such as ter-

rain databases. Just as one might simulate the effects of error by adding independent samples from a Gaussian distribution to an observed value, so the effects of error in such databases can be simulated by adding realizations from random field models with suitable spatial covariances. In such cases, however, and because of the strong spatial dependences present in virtually all spatial data, it is the entire database that must be simulated in each realization of the random process, not its individual measurements; and samples from the stochastic process are entire maps, not simple measurements. Such simulations have proven very useful in visualizing the effects of spatially autocorrelated errors in spatial databases, and in exploring the propagation of such errors during GIS analysis. Several studies have demonstrated the use of geostatistical techniques such as conditional simulation to provide models of error in spatial databases.

Progress has been made in modeling the ways in which uncertainties propagate through GIS operations based on this theoretical framework. Although simple queries may refer only to a single point, and require knowledge only of that point's marginal distribution of uncertainty, other operations such as the measurement of area, distance, slope, or direction require knowledge of joint distributions and thus covariances. Heuvelink (1998) has developed a comprehensive framework for the propagation of uncertainty, using both analytic and numeric methods, including Taylor series approximations.

Such approaches are fundamentally limited by their insistence on the existence of a truth that is distorted by measurement. They fit well with applications in terrain modeling, and the positional accuracy of well-defined features such as roads, but poorly to applications involving classifications of soil, vegetation cover, or land use. But progress has been made in analyzing these latter types of database using the theoretical frameworks of fuzzy and rough sets. Briefly, such frameworks suppose that although the exact nature of a class A may remain unknown, it is still possible to measure membership $m(\mathrm{A})$ in the class. Zhu et al. (1996) have shown how maps of membership can be useful in characterizing inherently vague phenomena, and Woodcock and Gopal (2000) have shown how such maps can be useful in managing forests. Fisher and Unwin (2005) have explored more advanced versions of these simple frameworks. Fundamentally, however, and despite the simplicity and intuitive appeal of these approaches, the question remains: if A cannot be defined, how is it possible to believe that $m(\mathrm{A})$ can be measured? Moreover, it has proven difficult to represent the fundamental spatial dependence properties of spatial data within these frameworks, so while marginal properties can be analyzed with some success, the joint properties that underlie many forms of GIS analysis remain the preserve of statistical methods and of frameworks such as geostatistics and spatial statistics.

Key Applications

The literature on imprecision and spatial uncertainty now encompasses virtually all types of spatial data. As noted earlier, the literature on uncertainty in terrain data is voluminous. Several authors have demonstrated the use of representations of uncertainty in spatial decision support (e. g., Aerts, Goodchild, and Heuvelink, 2003), and have discussed the many sources of uncertainty in such applications. Interesting methods have been devised for visualizing uncertainty, including animation (Ehlschlaeger, Shortridge, and Goodchild, 1997). To date, however, the implementation of these methods in GIS software remains limited. Duckham (2002) and Heuvelink (2005) have described efforts to build error-aware systems, and data quality is now an important element of metadata. But the mainstream GIS products continue to report the results of calculations to far more decimal places than are justified by any assessment of accuracy, and to draw lines whose positions are uncertain using line widths that are in no way representative of that uncertainty. Indeed, GIS practice seems still to be largely driven by the belief that accuracy is a function of computation, not representation, and that the last uncertainties were removed from maps many decades ago.

Future Directions

Uncertainty has been described as the Achilles' Heel of GIS (Goodchild, 1998): the dark secret that once exposed, perhaps through the arguments of clever lawyers, will bring down the entire house of cards. While this sounds extreme, it is certainly true that the results of GIS analysis are often presented as far more accurate than they really are. As GIS moves more and more into the realm of prediction and forecasting, the dangers of failing to deal with uncertainty are likely to become more and more pressing. At the same time the accuracy of databases is steadily improving, as more accurate measurements become available. Nevertheless there is an enormous legacy of less accurate data that is sure to continue to find application for many years to come.

While much has been learned over the past two decades about the nature of spatial uncertainty, a large proportion of the literature remains comparatively inaccessible, due to the complexity of its mathematics. Some progress has been made in making the work more accessible, through visualization and through the comparatively straightforward methods of Monte Carlo simulation. In time, such approaches will result in greater awareness of what is pos-

sible, and greater adoption of these methods within the wider community.

Progress is also needed on the construction of suitable data models for error-sensitive spatial databases. The simple expedient of adding values representing uncertainty to the entire database in its metadata, or to individual objects as additional attributes, fails to capture all of the complexity of spatial uncertainty, particularly its essential spatial dependence. Goodchild (2004) has argued that this problem is profound, stemming from the complex structures of spatial dependence; that it presents fundamental and expensive barriers to any attempt to improve spatial databases through partial correction and update; and that it can only be addressed by a radical restructuring of spatial databases around the concept of measurement, in what he terms a measurement-based GIS. In practice, the almost universal use of absolute coordinates to define position in spatial databases ensures that any information about spatial dependence, and the processes used to compute or compile such positions, will have been lost at some point during the production process.

Cross References

► Statistical Descriptions of Spatial Patterns

Recommended Reading

1. Aerts, J.C.J.H., Goodchild, M.F., Heuvelink, G.B.M.: Accounting for spatial uncertainty in optimization with spatial decision support systems. Trans. GIS **7**(2), 211–230 (2003)
2. Duckham, M.: A user-oriented perspective of error-sensitive GIS development. Trans. GIS **6**(2), 179–194 (2002)
3. Ehlschlaeger, C.R., Shortridge, A.M., Goodchild, M.F.: Visualizing spatial data uncertainty using animation. Comput. Geosci. **23**(4), 387–395 (1997)
4. Fisher, P.F., Unwin, D.J. (eds.): Re-Presenting GIS. Wiley, Hoboken, NJ (2005)
5. Frolov, Y.S., Maling, D.H., The accuracy of area measurement by point counting techniques. Cartogr. J. **1**, 21–35 (1969)
6. Goodchild, M.F.: Uncertainty: the Achilles heel of GIS? Geo. Info. Syst. November, 50–52 (1998)
7. Goodchild, M.F.: A general framework for error analysis in measurement-based GIS. J. Geogr. Syst. **6**(4), 323–324 (2004)
8. Goodchild, M.F., Gopal, S.: Accuracy of Spatial Databases. Taylor and Francis, New York (1989)
9. Goovaerts, P.: Geostatistics for Natural Resources Evaluation. Oxford, New York (1997)
10. Heuvelink, G.B.M.: Error Propagation in Environmental Modelling with GIS. Taylor and Francis, London (1998)
11. Heuvelink, G.B.M.: Handling spatial uncertainty in GIS: development of the data uncertainty engine. Instituto Geografico Portugues, Estoril (Portugal) (2005)
12. Kendall, M.G.: A Course in the Geometry of *n* Dimensions. Hafner, New York (1961)
13. Kyriakidis, P., Goodchild, M.F.: On the prediction error variance of three common spatial interpolation schemes. Int. J. Geogr. Inf. Sci. **20**(8), 823–856 (2006)
14. Maling, D.H., Measurement from Maps: Principles and Methods of Cartometry. Pergamon, New York (1989)
15. Woodcock, C.E., Gopal, S.: Fuzzy set theory and thematic maps: accuracy assessment and area estimation. Int. J. Geogr. Inf. Sci. **14**(2), 153–172 (2000)
16. Zhang, J.-X., Goodchild, M.F.: Uncertainty in Geographical Information. Taylor and Francis, New York (2002)
17. Zhu, A.X., Band, L.E., Dutton, B., Nimlos, T.: Automated soil inference under fuzzy logic. Eco. Mod. **90**(2), 123–145 (1996)

Incident Management System

► Emergency Evacuation Plan Maintenance

Index Lifetime

► Indexing Schemes for Multi-dimensional Moving Objects

Index, MVR-Tree

► Indexing Spatio-temporal Archives

Index, R-Tree

► Indexing Spatio-temporal Archives

Index Structures, Extensible

MARIOS HADJIELEFTHERIOU[1], ERIK HOEL[2], VASSILIS J. TSOTRAS[3]
[1] AT&T Labs Inc., Florham Park, NJ, USA
[2] ESRI, Redlands, CA, USA
[3] Computer Science Department, University of California Riverside, Riverside, CA, USA

Synonyms

Indexing framework, spatial/spatiotemporal; Indexing API, spatial/spatiotemporal; Library, software

Definition

SaIL (SpAtial Index Library) [15] is an extensible application programming framework that enables easy integration of spatial and spatio-temporal index structures into existing applications. SaIL focuses mainly on design issues and techniques for providing an application programming interface generic enough to support user defined data types, customizable spatial queries, and a broad range

of spatial and spatio-temporal index structures, in a way that does not compromise functionality, extensibility and, primarily, ease of use.

Historical Background

A plethora of GIS and other applications are related with spatial, spatio-temporal and, generally, multi-dimensional data. Typically, such applications have to manage millions of objects with diverse spatial characteristics. Examples include mapping applications that have to visualize numerous layers and hundreds of thousands of features [8], astronomical applications that index millions of images [20], and traffic analysis and surveillance applications that track thousands of vehicles. The utility of spatial indexing techniques for such applications has been well recognized—complex spatial queries can be answered efficiently only with the use of spatial index structures (for instance nearest neighbor queries). Consequently, many indexing techniques aiming to solve disparate problems and optimized for diverse types of spatial queries have appeared lately in the literature [4,11]. As a result, each technique has specific advantages and disadvantages that make it suitable for different application domains and types of data. Therefore, the task of selecting an appropriate access method, depending on particular application needs, is a rather challenging problem. A spatial index library that can combine a wide range of indexing techniques under a common application programming interface can thus prove to be a valuable tool, since it will enable efficient application integration of a variety of structures in a consistent and straightforward way.

The major difficulty with designing such a tool is that most index structures have a wide range of distinctive characteristics, that are difficult to compromise under a common framework. For example, some structures employ data partitioning while others use space partitioning, some have rectangular node types while others have spherical node types, some are balanced while other are not, some are used only for indexing points while others are better for rectangles, lines, or polygons. Another important issue is that the index structures have to provide functionality for exploiting the semantics of application-specific data types, through easy customization while making sure that meaningful queries can still be formulated for these types. Moreover, it is crucial to adopt a common programming interface in order to promote reusability, easier maintenance and code familiarity, especially for large application projects where many developers are involved. The framework should capture the most important design characteristics, common to most structures, into a concise set of interfaces. This will help developers concentrate on other aspects of the client applications promoting, in this manner, faster and easier implementation. The interfaces should be easily extensible in order to address future needs without necessitating revisions to client code. These fundamental requirements make the design of a *generic* spatial index framework a challenging task. Even though there is a substantial volume of work on spatial index structures and their properties, little work has appeared that addresses design and implementation issues.

The most relevant spatial index library to SaIL is the eXtensible and fleXible Library (XXL) [7]. XXL offers both low-level and high-level components for development and integration of spatial index structures like cursors, access to raw disk, a query optimizer, etc. Even though XXL is a superset of SaIL, it differs in three respects: First, SaIL offers a very concise, straightforward interface for querying arbitrary index structures in a uniform manner. In contrast, XXL querying interfaces are index specific (apart for join and aggregation queries). Second, SaIL offers more generic querying capabilities. Despite the fact that XXL can support a variety of advanced spatial queries (with the use of a framework for generalizing an incremental best-first search query strategy), nonconventional user defined queries have to be implemented by hand requiring modifications in all affected index structures. In contrast, SaIL offers an intuitive interface, utilizing well known design patterns, for formulating novel queries without having to revise the library in any way. Finally, SaIL provides the capability to customize query behavior during execution with the use of standardized design patterns. In contrast, in XXL this capability has to be supported explicitly by all index structure implementations.

GiST (for *Generalized Search Tree* [16]) is also relevant to this work. GiST is a framework that generalizes a height balanced, single rooted search tree with variable fanout. In essence, GiST is a *parameterized* tree that can be customized with user defined data types and user defined functions on these types which help guide the structural and searching behavior of the tree. Each node in the tree consists of a set of predicate/pointer pairs. Pointers are used to link the node to children nodes or data entries. Predicates are the user defined data types stored in the tree. The user, apart from choosing a predicate domain (e. g., the set of natural numbers, rectangles on a unit square universe, etc.), must also implement a number of methods (i. e., *consistent*, *union*, *penalty* and *pickSplit*) which are used internally by GiST to control the behavior of the tree. By using a simple interface, GiST can support a wide variety of search trees and their corresponding querying capabilities, including B-trees [5] and R-trees [13]. In order to support an even wider variety of structures, Aoki [1] proposed three additions to GiST. The GiST interface was

augmented with multiple predicate support, which is useful for storing meta-data in the tree nodes. The tree traversal interface was also improved, so that the user can define complex search strategies. Finally, support for divergence control was added, so that a predicate contained in a parent node need not correspond to an accurate description of its subtree (for example an R-tree with relaxed parent MBRs that do not tightly enclose their children). Other extensions have also been proposed that modify the internal structure of GiST so that it can support very specialized indices, for instance, the TPR-tree [18] for moving objects. Aref and Ilyas proposed the SP-GiST framework [2] which provides a novel set of external interfaces and original design, for furnishing a generalized index structure that can be customized to support a large class of spatial indices with diverse structural and behavioral characteristics. The generality of SP-GiST allows the realization of space and data driven partitioning, balanced and unbalanced structures. SP-GiST can support k-D-trees [3], tries [6,10], quadtrees [9,19] and their variants, among others.

The design of SaIL is orthogonal to XXL, GiST, SP-GiST and their variants. These libraries address the implementation issues behind new access methods by removing for the developer the burden of writing structural maintenance code. SaIL does not aim to simplify the development process of the index structures per se, but more importantly, the development of the applications that use them. *In that respect, it can be used in combination with all other index structure developer frameworks*. Employing SaIL is an easy process that requires the implementation of simple adapter classes that will make index structures developed with XXL, GiST, and SP-GiST compliant to the interfaces of SaIL, conflating these two conflicting design viewpoints. Ostensibly, existing libraries can be used for simplifying client code development as well—and share, indeed, some similarities with SaIL (especially in the interfaces for inserting and deleting elements from the indices)—but given that they are not targeted primarily for that purpose, they are not easily extensible (without the subsequent need for client code revisions), they do not promote transparent usage of diverse indices (since they use index specific interfaces), and they cannot be easily used in tandem with any other index library (existing or not). Finally, SaIL is the first library that introduces functionality for supporting temporal and spatio-temporal indices, through specialized capabilities that all other libraries are lacking.

Scientific Fundamentals

The Architecture of SaIL

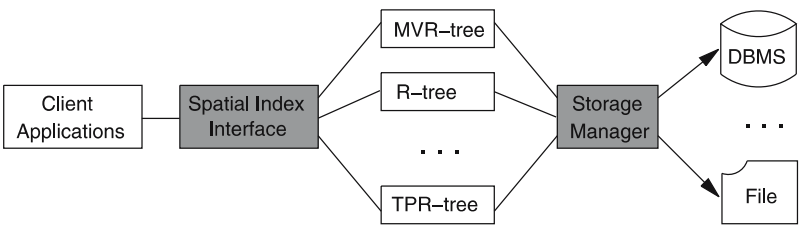
The SaIL framework consists of four components:

1. The core library toolkit which provides basic functionality. For example, generic types like variants, implementation of utility classes like property sets, an exception class hierarchy tailored for spatial indices, etc.
2. The storage manager toolkit for supporting diverse storage requirements. This container is useful for decoupling the index structure implementation from the actual storage system, enabling proper encapsulation.
3. The spatial index interface which is a set of abstractions of the most common spatial index operations and related types. For example, interfaces for nodes, leafs, spatial data types, spatio-temporal queries, and more.
4. The concrete implementations of various spatial index structures. For example, variants of the R-tree, the MVR-tree, the TPR-tree, etc.

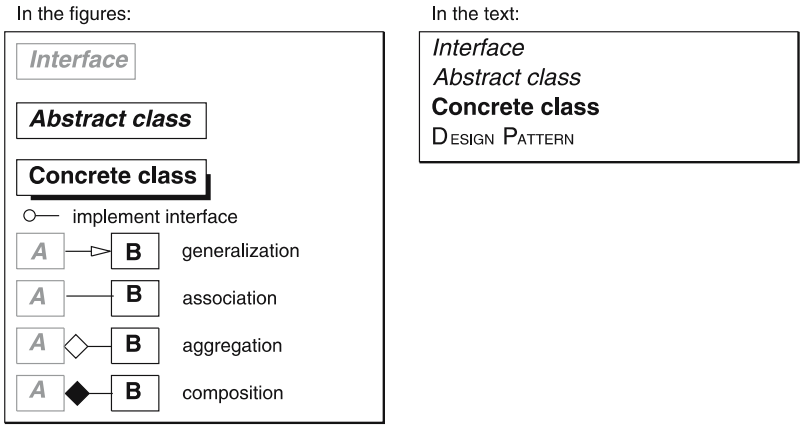
SaIL is designed to sit in-between the client application and the access methods that need to be interfaced by the client code. A simple architectural diagram is shown in Fig. 1. This section presents the most important concepts behind SaIL's design decisions using various examples. Figure 2 summarizes the UML notation used in the text and diagrams. When referring to a specific design pattern the definitions of Gamma et al. are used [12]. For convenience, some design pattern descriptions (quoted from [12]) are listed in Table 1.

The Core Toolkit The abstract and concrete classes offered by the core toolkit are shown in Fig. 3 (note that this and the remaining figures adopt the diagrammatic notation shown in Fig. 2). This toolkit addresses very simple but essential needs for any generic framework.

It provides a **Variant** type for representing a variety of different primitive types (like integers, floats, character arrays, etc.), which is necessary for avoiding hard coding



Index Structures, Extensible, Figure 1
Architectural diagram of SaIL



Index Structures, Extensible, Figure 2
Notation used in the paper

Index Structures, Extensible, Table 1 Design Pattern Descriptions

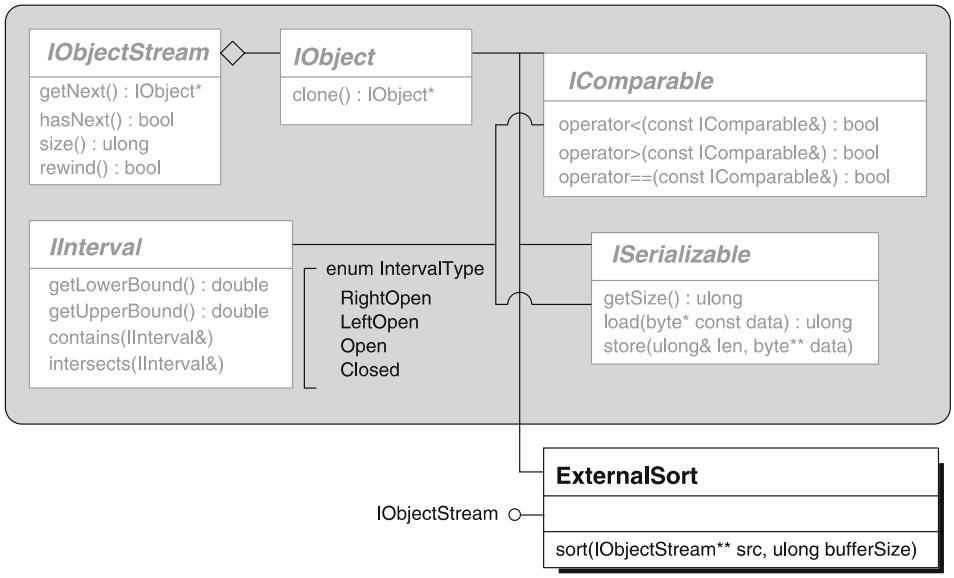
NAME	Description
MEMENTO	Without violating encapsulation, capture and externalize an object's internal state so that the object can be restored to this state later
PROXY	Provide a surrogate or place holder for another object to control access to it
COMPOSITE	Composite lets clients treat individual objects and compositions of objects uniformly
FACADE	Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use
VISITOR	Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates
STRATEGY	Define a family of algorithms, encapsulate each one, and make them interchangeable
COMMAND	Encapsulate a request as an object, thereby letting you parameterize clients with different requests

specific primitive types in interface definitions that might need to be modified at a later time. It offers a **PropertySet**, or a collection of <*PropertyName, Value*> pairs. Property sets are useful for passing an indeterminate number of parameters to a method, even after the interfaces have been defined, without the need to extend them. For example, adding and handling the initialization of new properties to an object can be achieved without modifying its constructor, if a **PropertySet** is used as one of the constructor's arguments.

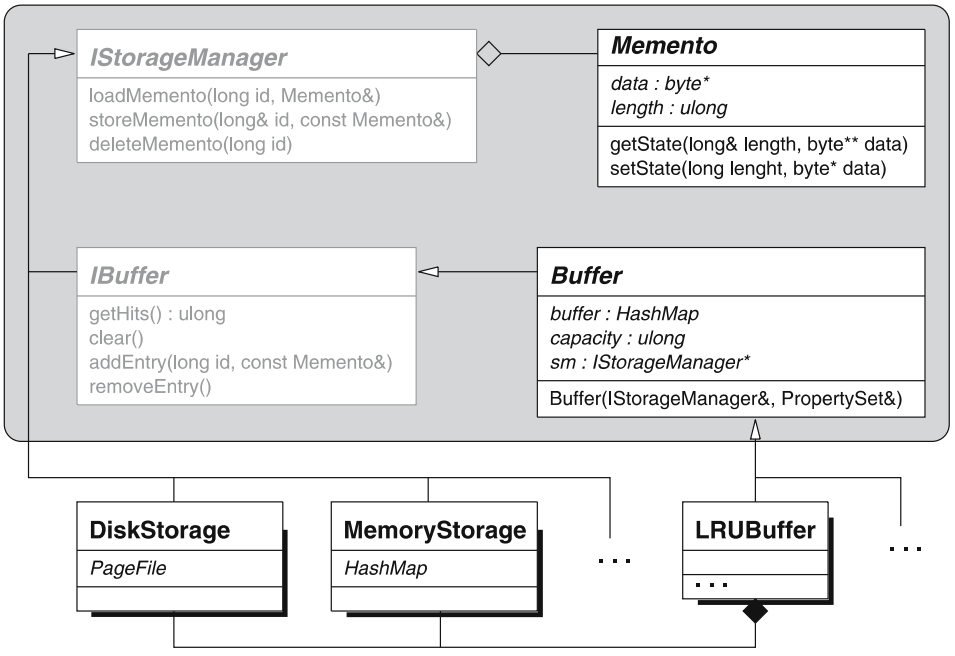
An index specific *Exception* class hierarchy is provided which helps promote the use of exception handling in client code, in a structured manner. Some basic interfaces representing essential object properties are also defined. For example interfaces for serializing and comparing objects, defining streams/iterators on object collections, etc. Other helper classes are provided as well, representing open/closed intervals, random number generators, resource usage utilities, etc. (some utilities are left out of this figure).

An important utility class provided by the core toolkit is **ExternalSort**. External sorting is needed for sorting very large relations that cannot be stored in main memory, an operation that is essential for bulk loading an index structure. By using the *IObject*, *IComparable* and *ISerializable* interfaces, **ExternalSort** is a generic sorting utility that can sort entries of any user defined data type in a very robust and straightforward manner. The caller needs to define an object stream that provides the sorter with objects of type *IObject* (using the *IObjectStream* interface), in random sequence. Then, **ExternalSort** iterates over the input objects is sorted order, as implied by a user provided comparator implementation. Sorting is accomplished by using temporary files on secondary storage. Conveniently, **ExternalSort** implements the *IObjectStream* interface itself and, thus, can be used as a PROXY in place of an *IObjectStream*, with the only difference that the stream appears to be in sorted order.

The Storage Manager Toolkit A critical part of spatial indexing tools is the storage manager, which should be versatile, very efficient and provide loose coupling. Clients that must persist entities to secondary storage should be unaware of the underlying mechanisms, in order to achieve proper encapsulation. Persistence could be over the network, on a disk drive, in a relational table, etc. All mediums should be treated uniformly in client code, in order to promote flexibility and facilitate the improvement of stor-



Index Structures, Extensible, Figure 3 The Core Toolkit



Index Structures, Extensible, Figure 4 The Storage Manager Toolkit

age management services as the system evolves, without the need to update the client code.

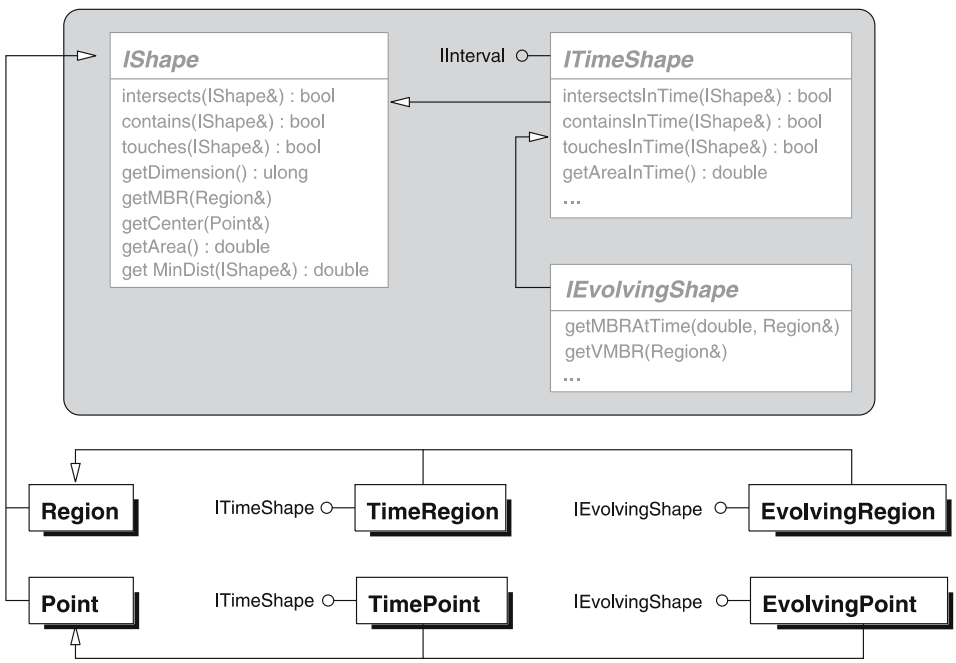
The storage manager toolkit is shown in Fig. 4. The key abstraction is a MEMENTO pattern that allows loose coupling between the objects that are persisted and the concrete implementation of the actual storage manager. An object that wants to store itself has to instantiate a concrete subclass of Memento that accurately represents its state. Then, it can pass this instance to a component supporting the *IStorageManager* interface, which will return

an identifier that can be used to retrieve the object's state at a later time. Two concrete implementations of the *IStorageManager* interface are already provided. A main memory manager that uses a hash table and a disk based page file.

This architecture allows multiple layers of storage management facilities to be streamlined one after the other. A simple example is the need to store data over a network. A simple adapter class can be implemented that sends the data over the network with the proper format expected by the remote computer that uses a second adapter class to persist the data. Another example is the implementation of a relational based index structure. An adapter can be embedded on an existing implementation, converting the data to be persisted into appropriate SQL statements that store them in a relational table (e. g., as BLOBs).

The *IBuffer* interface provides basic buffering functionality. It declares the two most important operations of a buffer: adding and removing an entry. The *Buffer* abstract class provides a default implementation for the major operations of a buffer component. For convenience, a concrete implementation of a Least Recently Used buffering policy is already provided. Using the *IBuffer* interface is straightforward; it acts as a proxy between an actual storage manager and a client, buffering entries as it sees fit. The client instantiates a concrete buffer class, provides the buffer with a reference to the actual storage manager and finally associates the index structure (or any other component that will use the storage manager) with a reference to the buffer. Conveniently, the callers are unaware that buffering is taking place by assuming that a direct interface with the actual storage manager is used. This architecture provides sufficient flexibility to alter buffering policies at runtime, add new policies transparently, etc.

The Spatial Index Interface Spatial access methods are used for indexing complex spatial objects with varying shapes. In order to make the interfaces generic it is essential to have a basic shape abstraction that can also represent composite shapes and other decorations (meta-data like z-ordering, insertion time, etc.). The *IShape* COMPOSITE pattern (Fig. 5) is defined as an interface that all index structures should use to decouple their implementation from actual concrete shapes. For example, inserting convex polygons into an R-tree [13] can be accomplished by calling the *IShape.getMBR* method to obtain the minimum bounding region of the polygon. As long as the user defined polygon class returns a proper MBR representation, the R-tree remains unaware of the actual details of the polygon class, internally—all it needs to be aware of is the *IShape.getMBR* contract. Complex shapes and combinations of shapes can be represented by composing a number of *IShapes* under one class and, hence, can be handled in a uniform manner as all other classes of type *IShape*. Various methods useful for comparing *IShapes* (like *contains*, *intersects*, etc.) are also specified. In addition, the *IShape* interface can handle shapes with arbitrary dimensionality, so that multi-dimensional indices can be supported.



Index Structures, Extensible, Figure 5 The shape interfaces

The *ITimeShape* interface extends *IShape* with methods for comparing shapes with temporal predicates. An *ITimeShape* has temporal extents, acquired by implementing the *IInterval* interface. This abstraction is useful for handling spatio-temporal indices, where the indexed objects are associated with insertion and deletion times, as well as for specifying spatial queries with temporal restrictions.

Furthermore, to cover special cases where shapes evolve over time, the *IEvolvingShape* interface is provided. An evolving shape can be represented by associating a velocity vector to every vertex of the representation and, in addition, it can be approximated by an evolving MBR. Special methods must also be implemented for computing intersections, area, volume, and other characteristics over time. The *IEvolvingShape* interface is derived from *ITimeShape* and, thus, such shapes can be associated with temporal extents during which the evolution is taking place.

Concrete implementations of basic shapes are provided for convenience. These shapes provide basic functionality, like computing intersections, areas, temporal relations, etc.

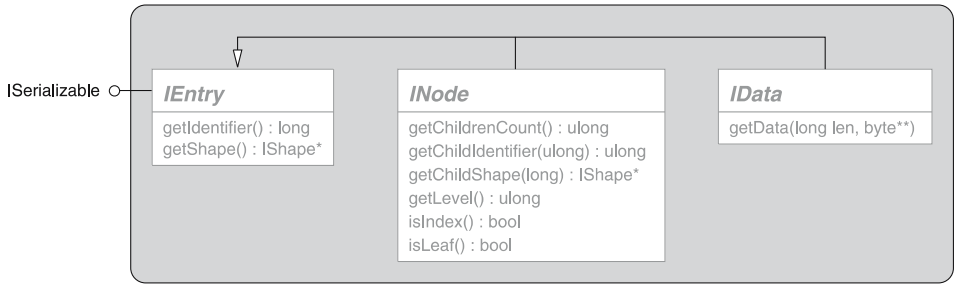
An essential capability of a generic index manipulation framework is to provide a sound set of index elements (leaf and index nodes, data elements, etc.) that enable consistent manipulation of diverse access methods from client code. For example, querying functions should return iterators (i. e., enumerations or cursors) over well-defined data elements, irrespective of what kind of structures they operate on. In general, a hierarchical index structure is composed of a number of nodes and a number of data entries. Every node has a specific shape, identifier and level. Nodes are further divided into index nodes and leaves. Data entries are either simple pointers to the actual data representations on disk (a secondary index), or the actual data themselves (a primary index). The data elements can be either exact shape representations or more generic meta-data associated with a specific index entry. These concepts are represented by using the following hierarchy: *IEntry* is the most basic interface for a spatial index entry; its basic members are an identifier and a shape. *INode* (that inherits from *IEntry*) represents a generic tree node; its basic members are the number of children, the tree level, and a property specifying if it is an index node or a leaf. The *IData* interface represents a data element and contains the meta-data associated with the entry or a pointer to the real data.

The core of the spatial index interface is the *ISpatialIndex* FACADE pattern. All index structures should implement *ISpatialIndex* (apart from their own custom methods), which abstracts the most common index operations. This interface is as generic as possible. All methods take *IShapes* as arguments and every shape is associated with a user defined identifier that enables convenient referencing of objects. Below each method is described in more detail.

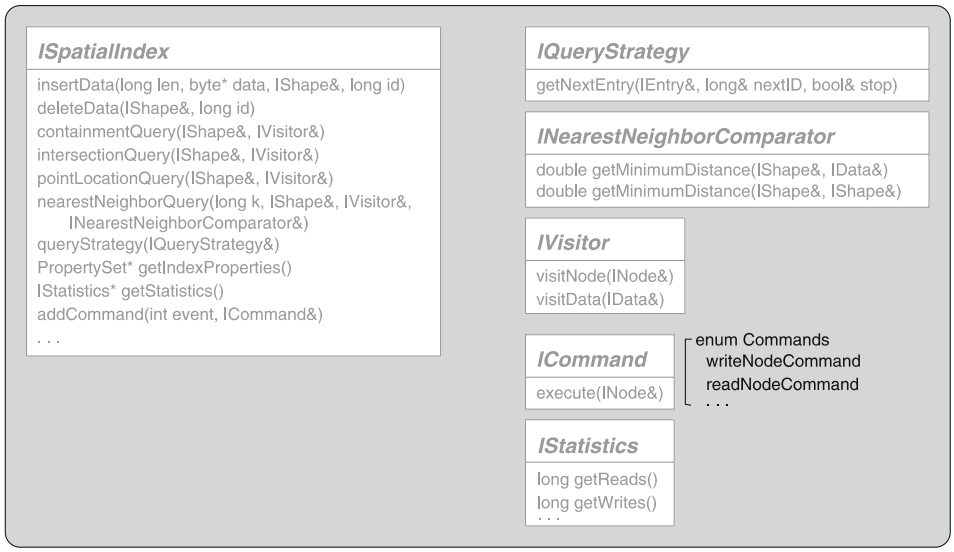
The *insertData* method is used for inserting new entries into an index. It accepts the data object to be inserted as an *IShape*—an interface that can be used as a simple decorator over the actual object implementation. Meta-data can also be stored along with the object as byte arrays (useful for implementing primary indices). The *deleteData* method locates and deletes an object already contained in an index. It accepts the *IShape* to be deleted and its object identifier. (The *IShape* argument is necessary since spatial indices cluster objects according to their spatial characteristics and not their identifiers.)

The query methods take the query *IShape* as an argument. This simple interface is powerful enough to allow the developer to create customized queries. For example, suppose a circular range query on an R-tree is required. Internally, the R-tree range search algorithm decides if a node should be examined by calling the query *intersects* predicate on a candidate node MBR. Hence, it suffices to define a **Circle** class that implements the *intersects* function (specific for intersections between circles and MBRs) and call the *intersectionQuery* method with a **Circle** object as its argument. Since arbitrarily complex shapes can be defined with the *IShape* interface, the querying capabilities of the index structures are only limited by the ability of the developer to implement correctly the appropriate predicate functions, for the *IShape* objects used as arguments to the querying methods.

In order to provide advanced customization capabilities a VISITOR pattern is used. The *IVisitor* interface is a very powerful feature. The query caller can implement an appropriate visitor that executes user defined operations when index entries are accessed. For example, the visitor can ignore all index and leaf nodes and cache all visited data entries (essentially the answers to the query) for later processing (like an enumeration). Instead, it could process the answers interactively (like a cursor), terminating the search when desired (an interactive *k*-nearest neighbor query is a good example, where the user can stop the execution of the query after an adequate number of answers has been discovered). Another useful example of the VISITOR pattern is tallying the number of query I/Os. By counting the number of times each visit method has been called, it is possible to count the number of index nodes, leaf nodes, or "any" level nodes that were accessed for answering a query (the *visitNode* method returns an *INode* reference when called, which provides all necessary information on the specific node accessed, e. g., its level, shape, etc.). A tallying visitor is presented in Algorithm 1. A different example is visualizing the progress of the query. As the querying algorithm proceeds, the visitor can draw the



Index Structures, Extensible, Figure 6 The index element interface



Index Structures, Extensible, Figure 7 The generic spatial index interface

last accessed node or data element on the screen. An access method can support the *IVisitor* interface simply by guaranteeing that all query algorithms call the *visitNode* and *visitData* methods of *IVisitor*, every time a node or a data entry is accessed while searching the structure. Thus, supporting the *IVisitor* interface requires a very simple, inexpensive procedure.

The *IShape* and *IVisitor* interfaces enable consistent and straightforward query integration into client code, increasing readability and extensibility. New index structures can add specialized functionality by requesting decorated *IShape* objects (thus, without affecting the interfaces). The *IVisitor* interface allows existing visitor implementations to be reused for querying different types of access methods and users can customize visitors during runtime.

To illustrate the simplicity of supporting the *IVisitor* interface from the querying methods of a spatial index implementation, the actual implementation of a range query algorithm of a hierarchical structure that supports all of the aforementioned features is shown in Algorithm 2.

An even larger degree of customization is provided for the nearest neighbor query method. Since different applications use diverse distance measures to identify nearest neighbors (like the Euclidean distance, and others),

```
class MyVisitor : public IVisitor {
public:
  map<long, IShape*> answers;
  long nodeAccesses;

  MyVisitor() : nodeAccesses(0) {}

  public visitNode(INode* n) {
    nodeAccesses++;
  }

  public visitData(IData* d) {
    // add the answer to the list.
    answers[d.getIdentifier()] = d.getShape();
  }
}
```

Index Structures, Extensible, Algorithm 1 *IVisitor* example

the *nearestNeighborQuery* method accepts an *INearestNeighborComparator* object. By allowing the caller to provide a customized comparator, the default nearest neighbor algorithm implemented by the underlying structure can be used, obviating any application specific changes to the library. In reality, a nearest neighbor comparator is

```
void rangeQuery(const IShape& query, IVisitor& v) {
  stack<NodePtr> st;
  Node* root = readNode(m_rootID);

  if (root->m_children > 0 && query.intersects(root->m_nodeBoundary)) st.push(root);

  while (! st.empty()) {
    Node* n = st.top(); st.pop();

    if (n->isLeaf()) {
      v.visitNode(*n);

      for (unsigned long cChild = 0; cChild < n->m_children; cChild++) {
        if (query.intersects(n->m_childBoundary[cChild])) {
          v.visitData(n->m_childData[cChild]);
        }
      }
    } else {
      v.visitNode(*n);

      for (unsigned long cChild = 0; cChild < n->m_children; cChild++)
        if (query.intersects(n->m_childBoundary[cChild]))
          st.push(readNode(n->m_childIdentifier[cChild]));
    }
  }
}
```

Index Structures, Extensible, Algorithm 2 Range query method implementation that supports the *IVisitor* interface

essential, since in order to find the actual nearest neighbors of a query, the query has to be compared with each candidate's exact representation so that an exact distance can be computed. Since most spatial index structures store object approximations in place of the real objects (e. g., R-trees store MBRs), internally they make decisions based on approximate distance computations and, hence, cannot identify the exact nearest neighbors. One way to overcome this weakness, is to let the index load the actual objects from storage and compute the real distances only when appropriate. Albeit, this would break encapsulation since loading the actual objects implies that the index has knowledge about the object representations. Alternatively, the user can provide a nearest neighbor comparator that implements a method for comparing index approximations (e. g., MBRs) with the actual objects (e. g., polygons). Method *getMinimumDistance* is used for that purpose.

For implementing "exotic" queries, without the need to make internal modifications to the library, a STRATEGY pattern is proposed. Using the *queryStrategy* method the caller can fully guide the traversal order and the operations performed on a structure's basic elements allowing, in effect, the construction of custom querying algorithms on the fly. This technique uses an *IQueryStrategy* object for encapsulating the traversal algorithm. The index structure calls *IQueryStrategy.getNextEntry* by starting the traversal from a root and the *IQueryStrategy* object chooses which entry should be accessed and returned next. The traversal can be terminated when desired. As an example, assume that the user wants to visualize all the index levels of an R-tree. Either the R-tree implementation should provide a custom tree traversal method that returns all nodes one by one, or a query strategy can be defined for the same purpose (which can actually be reused as is, or maybe with slight modifications, for any other hierarchical structure). An example of a breadth-first node traversal algorithm is presented in Algorithm 3 (the example requires less than 15 lines of code). Many other possible uses of the query strategy pattern exist.

Another capability that should be provided by most index structures is allowing users to customize various index operations (usually by the use of call-back functions). The spatial index interface uses a COMMAND pattern for that purpose. It declares the *ICommand* interface—objects implementing *ICommand* encapsulate user parameterized requests that can be run on specific events, like customized alerts. All access methods should provide a number of queues, each one corresponding to different events that trigger each request. For example, assume that a new index structure is being implemented. The function that persists

```
class MyQueryStrategy : public IQueryStrategy {
  queue<long> ids;
public:
  void getNextEntry(IEntry& e, long& nextID, bool& stop) {
    // process the entry.
    ...

    // if it is an index entry and not a leaf
    // add its children to the queue.
    INode* n = dynamic_cast<INode*>(&e);
    if (n != 0 && ! n->isLeaf())
      for (long cChild = 0; cChild < n->getChildrenCount(); cChild++)
        ids.push(n->getChildIdentifier(cChild));

    stop = true;
    if (! ids.empty()) {
      // if queue not empty fetch the next entry.
      nextID = ids.front(); ids.pop();
      stop = false;
    }
  }
};
```

Index Structures, Extensible, Algorithm 3 Breadth-first traversal of index nodes

a node to storage can be augmented with an empty list of *ICommand* objects. Using the *addCommand* method the user can add arbitrary command objects to this list, that get executed whenever this function is called, by specifying an appropriate event number (an enumeration is provided for that purpose). Every time the function is called, it iterates through the *ICommand* objects in the list and calls their execute method. Another example is the need to track specific index entries and be able to raise alerts whenever they take part in splitting or merging operations, or they get relocated a new disk page. The COMMAND pattern promotes reusability, clarity, and ease of extensibility without the need of subclassing or modifying the spatial index implementations simply to customize a few internal operations as dictated by user needs.

Key Applications

GIS and other applications that are related with spatial, spatio-temporal and, generally, multi-dimensional data can benefit significantly by using the SaIL framework for incorporating spatial and spatio-temporal index structures into existing code. For example, mapping applications [8], astronomical applications [20], traffic analysis and surveillance applications. A sample implementation in C++ and Java can be downloaded freely from [14].

Future Directions

The extensions to the framework for providing new functionality and supporting novel data structures and applications, are limitless. Currently, the prototype implementation includes R-tree variants [13], the MVR-tree [17], and the TPR-tree [18]. Exploring what are the necessary modifications for adapting the library to work with a diverse number of index structures that are not based on R-trees is an interesting avenue for future work.

Cross References

- ▶ Index Structures, Extensible
- ▶ Indexing, High Dimensional
- ▶ Indexing, Hilbert R-tree, Spatial Indexing, Multimedia Indexing
- ▶ Indexing the Positions of Continuously Moving Objects
- ▶ Indexing Schemes for Multi-dimensional Moving Objects
- ▶ Indexing Spatio-temporal Archives
- ▶ Nearest Neighbor Queries in Network Databases
- ▶ Nearest Neighbor Query
- ▶ Nearest Neighbors Problem
- ▶ R*-tree
- ▶ R-Trees – A Dynamic Index Structure for Spatial Searching

Recommended Reading

1. Aoki., P.M.: Generalizing "search" in generalized search trees (extended abstract). In: ICDE, pp. 380–389 (1998)
2. Aref, W.G., Ilyas, I.F.: An extensible index for spatial databases. In: SSDBM, pp. 49–58 (2001)
3. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Communications of the ACM **18**(9):509–517 (1975)

4. Böm, C., Berchtold, S., Keim, D.A.: Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. ACM Computing Surveys **33**(3):322–373 (2001)
5. Comer, D.: The ubiquitous B-tree. ACM Computing Surveys **11**(2):121–137 (1979)
6. de la Briandais, R.: File searching using variable length keys. In: Proceedings of the Western Joint Computer Conference, pp. 295–298 (1959)
7. Van den Bercken, V., Blohsfeld, B., Dittrich, J., Krämer, J., Schäfer, T., Schneider, M., Seeger, B.: XXL – a library approach to supporting efficient implementations of advanced database queries. In: VLDB, pp. 39–48 (2001)
8. ESRI: ArcGIS. http://www.esri.com/software/arcgis/index.html
9. Finkel, R.A., Bentley, J.L.: Quad Trees, a data structure for retrieval on composite keys. Acta Informatica **4**(1):1–9 (1974)
10. Fredkin, E.: Trie memory. Communications of the ACM **3**(9):490–499 (1960)
11. Gaede, V., Günther, O.: Multidimensional access methods. ACM Computing Surveys **30**(2):170–231 (1998)
12. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series. Addison-Wesley, New York, NY (1995)
13. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: SIGMOD, pp. 47–57 (1984)
14. Hadjieleftheriou, M.: SaIL. http://spatialindexlib.sourceforge.net
15. Hadjieleftheriou, M., Hoel, E., Tsotras, V.J.: Sail: A spatial index library for efficient application integration. GeoInformatica **9**(4):367–389 (2005)
16. Hellerstein, J.M., Naughton, J.F., Pfeffer, A.: Generalized search trees for database systems. In: VLDB, pp. 562–573 (1995)
17. Kumar, A., Tsotras, V.J., Faloutsos, C.: Designing access methods for bitemporal databases. TKDE **10**(1):1–20 (1998)
18. Saltenis, S., Jensen, C.S., Leutenegger, S.T., Lopez, M.A.: Indexing the positions of continuously moving objects. SIGMOD Record **29**(2):331–342 (2000)
19. Samet, H.: The quadtree and related hierarchical data structures. ACM Computing Surveys **16**(2):187–260 (1984)
20. SDSS. SkyServer. http://skyserver.sdss.org/dr1/en/

Indexing and Mining Time Series Data

EAMONN KEOGH
Computer Science & Engineering Department,
University of California – Riverside, Riverside, CA, USA

Synonyms

Similarity search; Query-by-content; Distance measures; Temporal data; Spatio-temporal indexing; Temporal indexing

Definition

Time series data is ubiquitous; large volumes of time series data are routinely created in geological and meteorological domains. Although statisticians have worked with time series for more than a century, many of their techniques hold little utility for researchers working with massive time series databases (for reasons discussed below). There two major areas of research on time series databases, the efficient discovery of previously *known* patterns (indexing), and the discovery of previously *unknown* patterns (data mining). As a concrete example of the former a user may wish to "*Find examples of a sudden increase, followed by slow decrease in lake volume anywhere in North America*" [14]. Such a query could be expressed in natural language, however virtually all indexing systems assume the user will sketch a query shape. In contrast, data mining aims to discover *previously unknown* patterns. For example "*Find all approximately repeated weekly patterns of vehicular traffic volume*". Because the space of unknown patterns is much larger than the space of known patterns, it should be obvious that data mining is a more demanding task in terms of both computer time and human intervention/interpretation.

Below are the major tasks considered by the time series data mining community. Note that indexing is sometimes considered a special case of data mining, and many data mining algorithms use indexing as a subroutine.

- **Indexing** (Query by Content/Similarity Search): Given a query time series Q of length n, a user defined query time series C of length m ($m \ll n$), and some similarity/dissimilarity measure $D(Q_{[i:i+m]},C)$, find the most similar time series in database *DB* [2,5,9,14].
- **Clustering**: Find natural groupings of the time series in database *DB* under some similarity/dissimilarity measure $D(Q,C)$ [4,10,12,20].
- **Classification**: Given an unlabeled time series Q, assign it to one of two or more predefined classes [6,12].
- **Motif Discovery**: Given an unlabeled time series Q of length n, and user defined subsequence length of m ($m \ll n$), find the pair of subsequences, A and B, that minimize $D(A,B)$.
- **Prediction** (Forecasting): Given a time series Q containing n datapoints, predict the value at time $n+1$.
- **Association Detection:** Given two or more time series, find relationships between them. Such relationships may or may not be casual and may or may not exist for the entire duration of the time series [3].
- **Summarization:** Given a time series Q containing n datapoints where n is an extremely large number, create a (possibly graphic) approximation of Q which retains its essential features but fits on a single page, computer screen etc [8,17].
- **Anomaly Detection** (Interestingness/Novelty Detection)**:** Given a time series Q, assumed to be normal, and an unannotated time series R. Find all sections of R which contain anomalies or "surprising/interesting/unexpected" occurrences [7,11,16].

- **Segmentation**: Given a time series Q containing n datapoints, construct a model $\overline{Q}$, from K piecewise segments ($K \ll n$) such that $\overline{Q}$ closely approximates Q [12]. Segmentation can be use to find regions of similar behavior, or simply to reduce the dimensionality of the data (see Fig. 3).

Historical Background

The task of indexing time series data can be traced back to a classic paper by Faloutsos, Ranganathan and Manolopoulos [5]. This paper also introduces the Gemini framework, which remains the basic framework for virtually all indexing (and many data mining) algorithms for time series. Given that most interesting datasets are too large to fit in main memory, the basic idea of the Gemini framework is to approximate the data in main memory, approximately solve the problem at hand, and then make (hopefully few) accesses to the disk to confirm or adjust the solution. Given this, a natural question to ask is which method should be used to approximate the data in main memory? The original paper suggested the discrete Fourier transform, but since then a bewilderingly large number of alternatives have been proposed. The section on Time Series Representations below considers this matter in some detail.

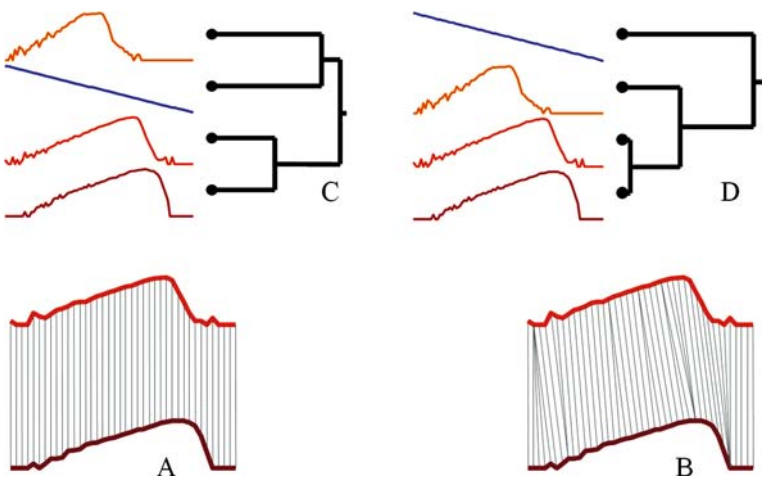
Scientific Fundamentals

Note that indexing, clustering and motif discovery make *explicit* use of a distance measure, and many approaches to classification, prediction, association detection, summarization and anomaly detection make *implicit* use of a distance measure. It is not surprising therefore, that the literature abounds with various distance measures, each with its proponents. Recently however, there has been an increasing understanding that the simple Euclidean distance is very difficult to beat in most domains [13]. One distance measure which *has* been shown to outperform Euclidean distance *some* datasets is Dynamic Time Warping (DTW). A visual intuition of both measures is shown in Fig. 1.

Unlike Euclidean distance, which does a non-adaptive point-to-point alignment, DTW tries to find a natural peak-to-peak, valley-to-valley alignment of the data.

It is interesting to note that with the exception of indexing, research into the tasks enumerated above predate not only the decade old interest in data mining, but computing itself. What then, are the essential differences between the classic, and the data mining versions of these problems? The key difference is simply one of size and scalability; time series data miners routinely encounter datasets that are gigabytes in size. As a simple motivating example, consider hierarchical clustering. The technique has a long history, and well-documented utility. If however, in order to hierarchically cluster a mere million items, it would be necessary to construct a matrix with 10^{12} cells, well beyond the abilities of the average computer for many years to come. A data mining approach to clustering time series, in contrast, must explicitly consider the scalability of the algorithm [10].

In addition to the large volume of data, it is often the case that each individual time series has a very high dimensionality [2]. Whereas classic algorithms assume a relatively low dimensionality (for example, a few measurements such as "height, weight, blood sugar etc"), time series data mining algorithms must be able to deal with dimensionalities in the hundreds and thousands. The problems created by high dimensional data are more than mere computation time considerations, the very meanings of normally intuitive terms such as "similar to" and "cluster forming" become unclear in high dimensional space. The reason is that as dimensionality increases, all objects become essentially equidistant to each other, and thus classification



Indexing and Mining Time Series Data, Figure 1
A The Euclidean distance computes the similarity of two time series by comparing the i^{th} point of one with the i^{th} point another. **B** Dynamic Time Warping in contrast, allows non-linear alignments. For most domains, the DTW clustering produced by DTW (**D**) will be more intuitive than the clustering produced by Euclidean Distance (**C**)

and clustering lose their meaning. This surprising result is known as the "curse of dimensionality" and has been the subject of extensive research [1]. The key insight that allows meaningful time series data mining is that although the actual dimensionality may be high, the *intrinsic* dimensionality is typically much lower. For this reason, virtually all time series data mining algorithms avoid operating on the original "raw" data, instead they consider some higher-level representation or abstraction of the data.

Time Series Representations

As noted above, time series datasets are typically very large, for example, just a few hours of weather data can require in excess of a gigabyte of storage. This is a problem because for almost all data mining tasks, most of the execution time spent by algorithm is used simply to move data from disk into main memory. This is acknowledged as the major bottleneck in data mining, because many naïve algorithms require multiple accesses of the data. As a simple example, imagine attempting to do k-means clustering of a dataset that does not fit into main memory. In this case, every iteration of the algorithm will require that data in main memory to be swapped. This will result in an algorithm that is thousands of times slower than the main memory case.

With this in mind, a generic framework for time series data mining has emerged. The basic idea can be summarized as follows

Indexing and Mining Time Series Data, Table 1 A generic time series data mining approach

1)	Create an approximation of the data, which will fit in main memory, yet retains the essential features of interest
2)	Approximately solve the problem at hand in main memory
3)	Make (hopefully very few) accesses to the original data on disk to confirm the solution obtained in Step 2, or to modify the solution so it agrees with the solution obtained on the original datax

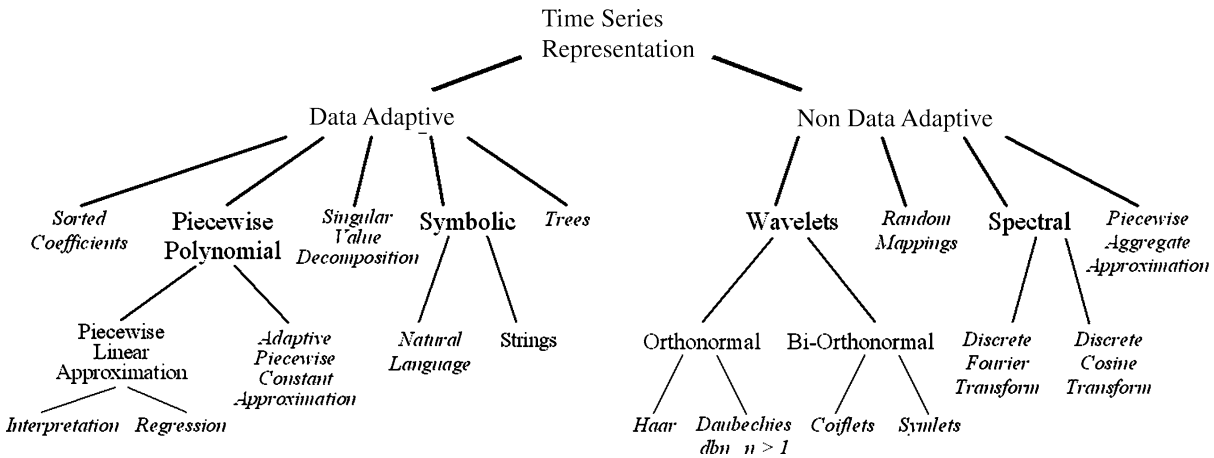
It should be clear that the utility of this framework depends heavily on the quality of the approximation created in Step 1. If the approximation is very faithful to the original data, then the solution obtained in main memory is likely to be the same, or very close to, the solution obtained on the original data. The handful of disk accesses made in Step 2 to confirm or slightly modify the solution will be inconsequential compared to the number of disks accesses required if the original data had been worked on. With this in mind, there has been a huge interest in approximate representation of time series. Figure 2 illustrates a hierarchy of every representation proposed in the literature.

To develop the reader's intuition about the various time series representations, Fig. 3 illustrates four of the most popular representations.

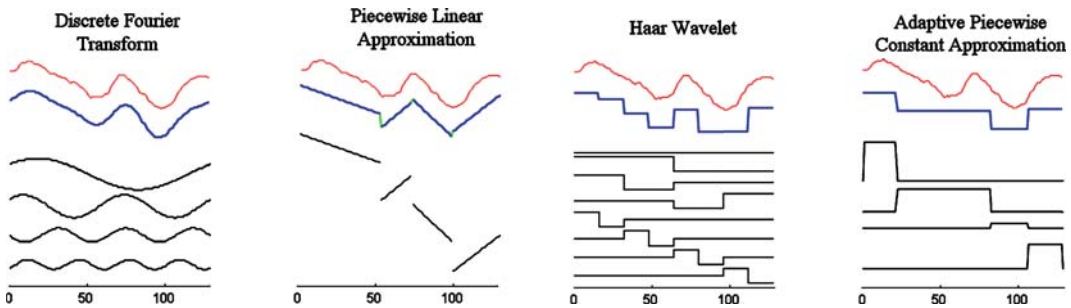
The Gemini framework discussed in the Historical Background section requires one special property of a time series representation in order to guarantee that it returns the true answer [5]. It must be the case that the distance function in the reduced dimensionality space underestimates the distance that would have be calculated in the original data space, the so called *Lower Bounding Lemma* [5,21]. This property has now been demonstrated for most representations, the exceptions being natural language, trees, random mappings and interpolation based Piecewise Linear Approximation.

Given the plethora of different representations, it is natural to ask which is best. Recall that the more faithful the approximation, the less clarification disks accesses will be needed to make in Step 3 of Table 1. In the example shown in Fig. 2, the discrete Fourier approach seems to model the original data the best, however it is easy to imagine other time series where another approach might work better. There have been many attempts to answer the question of which is the best representation, with proponents advocating their favorite technique [2,5,14,15]. The literature abounds with mutually contradictory statements such as "*Several wavelets outperform the … DFT*" [14], "*DFT-based and DWT-based techniques yield comparable results*" [18], "*Haar wavelets perform … better that DFT*" [9]. However an extensive empirical comparison on 50 diverse datasets suggests that while some datasets favor a particular approach, overall there is little difference between the various approaches in terms of their ability to approximate the data [13]. There are however, other important differences in the usability of each approach [2]. Below, some representative examples of strengths and weaknesses are considered.

The wavelet transform is often touted as an ideal representation for time series data mining, because the first few wavelet coefficients contain information about the overall shape of the sequence while the higher order coefficients contain information about localized trends [14,16]. This multiresolution property can be exploited by some algorithms, and contrasts with the Fourier representation in which every coefficient represents a contribution to the global trend [5,15]. However wavelets do have several drawbacks as a data mining representation. They are only defined for data whose length is an integer power of two. In contrast, the Piecewise Constant Approximation suggested by [19], has exactly the fidelity of resolution of as the Haar wavelet, but is defined for arbitrary length time series. In addition, it has several other useful properties such as the ability to support several different distance measures [19],



Indexing and Mining Time Series Data, Figure 2 A hierarchy of time series representations



Indexing and Mining Time Series Data, Figure 3 Four popular representations of time series. For each graphic is a raw time series of length 128. Below it is an approximation using 1/8 of the original space. In each case, the representation can be seen as a linear combination of basis functions. For example, the Discrete Fourier representation can be seen as a linear combination of the 4 sine/cosine waves shown in the bottom of the graphic

and the ability to be calculated in an incremental fashion as the data arrives [2]. Choosing the right representation for the task at hand is the key step in any time series data-mining endeavor. The points above only serve as a sample of the issues that must be addressed.

Key Applications

This volume has numerous detailed articles on both indexing and mining time series (or spatial time series) data. The reader should consult (indexing), High-dimensional Indexing, Indexing Schemes for Multi-Dimensional Moving Objects, Extensible Spatial and SpatioTemporal Index Structures; (data mining), Algorithms for Mining Co-location Patterns, Co-location Pattern Discovery, Correlation Queries in Spatial Time Series Data, Discovering Similar Trajectories Using A Pseudo-Metric Distance Function.

Future Directions

Most work in data mining assumed that the data was static, and the user thus had the ability to do batch processing. As the field is maturing there is an increasing understanding that in most real world situations the data continuously arrives. There is therefore an increasing effort to extend current algorithms in motif detection / novelty detection / clustering etc to the streaming data case. In many cases this forces us to abandon the hope of producing an exact answer; instead it is necessary to be content with some probabilistic guarantees.

Readings

The field of time series data mining is relatively new, and ever changing. Because of the length of journal publication delays, the most interesting and useful work tends to appear in top-tier conference proceedings. Interested readers are urged to consult the latest proceedings of the major conferences in the field. These include the ACM Knowledge Discovery in Data and Data Mining, IEEE International Conference on Data Mining and the IEEE International Conference on Data Engineering.

Cross References

► Approximation
► Indexing, High Dimensional

- ▶ Indexing Spatio-temporal Archives
- ▶ Nearest Neighbors Problem
- ▶ Patterns in Spatio-temporal Data
- ▶ Trajectories, Discovering Similar

Recommended Reading

1. Aggarwal, C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: Proceedings of the 8th International Conference on Database Theory, London, 6 January 2001, pp. 420–434
2. Chakrabarti, K., Keogh, E.J., Pazzani, M., Mehrotra, S.: Locally adaptive dimensionality reduction for indexing large time series databases. ACM Trans. Datab. Syst. **27**(2), 188–228 (2002)
3. Das, G., Lin, K., Mannila, H., Renganathan, G., Smyth, P.: Rule discovery from time series. In: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, New York, 27–31 August 1998, pp. 16–22
4. Debregeas, A., Hebrail, G.: Interactive interpretation of kohonen maps applied to curves. In: Proceedings of the 4th International Conference of Knowledge Discovery and Data Mining, New York, 27–31 August 1998, pp. 179–183
5. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Minneapolis, 25–27 May 1994, pp. 419–429
6. Geurts, P.: Pattern extraction for time series classification. In: Proceedings of Principles of Data Mining and Knowledge Discovery, 5th European Conference, Freiburg, Germany, 3–5 September 2001, pp. 115–127
7. Guralnik, V., Srivastava, J.: Event detection from time series data. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, 15–18 August 1999, pp. 33–42
8. Indyk, P., Koudas, N., Muthukrishnan, S.: Identifying representative trends in massive time series data sets using sketches. In: Proceedings of the 26th International Conference on Very Large Data Bases, Cairo, Egypt, 10–14 September 2000, pp. 363–372
9. Kahveci, T., Singh, A.: Variable length queries for time series data. In: Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, 2–6 April 2001, pp. 273–282
10. Kalpakis, K., Gada, D., Puttagunta, V.: Distance measures for effective clustering of ARIMA time-series. In: Proceedings of the IEEE International Conference on Data Mining, San Jose, 29 November–2 December 2001, pp. 273–280
11. Keogh, E., Lonardi, S., Chiu, W.: Finding Surprising Patterns in a Time Series Database In Linear Time and Space. In: The 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, 23–26 July 2002, pp. 550–556
12. Keogh, E., Pazzani, M.: An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, New York, 27–31 August 1998, pp. 239–241
13. Keogh, E., Kasetty, S.: On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In: The 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, 23–26 July 2002, pp. 102–111
14. Meyer, S.C.: Analysis of base flow trends in urban streams, northeastern Illinois, USA. Hydrogeol. J. **13**: 871–885 (2005)
15. Popivanov, I., Miller, R.J.: Similarity search over time series data using wavelets. In: Proceedings of the 18th International Conference on Data Engineering, San Jose, 26 February–1 March 2002, pp. 212–221
16. Rafiei, D., Mendelzon, A.O.: Efficient retrieval of similar time sequences using DFT. In: Proceedings of the 5th International Conference on Foundations of Data Organization and Algorithms, Kobe, Japan, 12–13 November 1998
17. Shahabi, C., Tian, X., Zhao, W.: TSA-tree: a wavelet based approach to improve the efficiency of multi-level surprise and trend queries. In: Proceedings of the 12th International Conference on Scientific and Statistical Database Management, Berlin, Germany, 26–28 July 2000, pp. 55–68
18. van Wijk, J.J., van Selow, E.: Cluster and calendar-based visualization of time series data. In: Proceedings 1999 IEEE Symposium on Information Visualization, 25–26 October 1999, pp 4–9. IEEE Computer Society
19. Wu, Y., Agrawal, D., El Abbadi, A.: A comparison of DFT and DWT based similarity search in time-series databases. In: Proceedings of the 9th ACM CIKM International Conference on Information and Knowledge Management. McLean, 6–11 November 2000, pp. 488–495
20. Xi, X., Keogh, E.J., Shelton, C.R., Li, W., Ratanamahatana, C.A.: Fast time series classification using numerosity reduction. In: Cohen, W.W., Moore, A. (eds.): Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, 25–29 June 2006, pp. 1033–1040. ACM 2006
21. Yi, B., Faloutsos, C.: Fast time sequence indexing for arbitrary lp norms. In: Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt, 10–14 September 2000, pp. 385–394

Indexing API, Spatial/Spatio-temporal

- ▶ Index Structures, Extensible

Indexing, BDual Tree

MAN LUNG YIU[1], YUFEI TAO[2], NIKOS MAMOULIS[3]

[1] Department of Computer Science, Aalborg University, Aalborg, Denmark

[2] Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, China

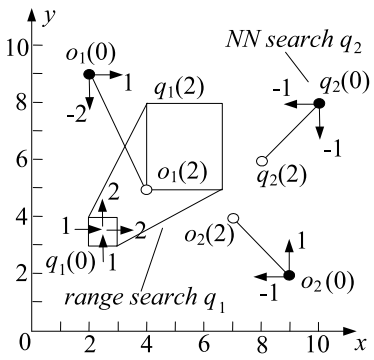
[3] Department of Computer Science, University of Hong Kong, Hong Kong, China

Synonyms

TPR-trees

Definition

The future location of a moving object is modeled as a linear function of time, consisting of its location at reference



Indexing, BDual Tree, Figure 1 Examples of spatiotemporal data and queries

time and its velocity. A centralized server manages the motion functions of all moving objects and an object issues an update to the server whenever its velocity changes. In Fig. 1, object o_1 is at location (2, 9) at time 0, and its velocities (represented with arrows) along the x- and y- dimensions are 1 and −2, respectively. A negative sign implies that the object is moving towards the negative direction of an axis.

A (predictive) *range query* returns the objects expected to appear (based on their motion parameters) in a moving rectangle q at some time within a future time interval qt. Figure 1 shows a query q_1 with $qt = [0, 2]$, whose extents at time 0 correspond to box $q_1(0)$. The left (right) edge of q_1 moves towards right at a velocity 1 (2), and the velocity of its upper (lower) boundary is 2 (1) on the y-dimension. Box $q_1(2)$ demonstrates the extents of q_1 at time 2. Notice that $q_1(2)$ has a larger size than $q_1(0)$ since the right (upper) edge of q_1 moves faster than the left (lower) one. The query result contains a single object o_1, whose location (4, 5) at time 2 falls in $q_1(2)$.

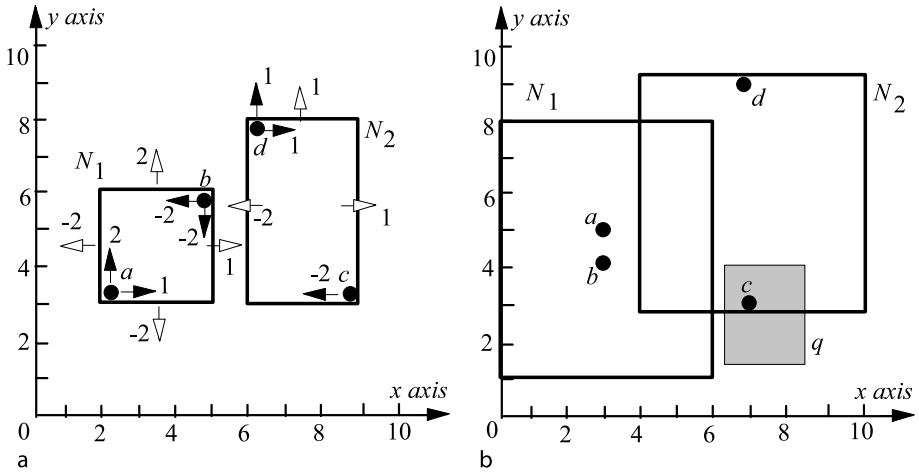
Various spatiotemporal indexes have been developed [3,4,5,6,7,9] to support efficiently (i) object updates, and (ii) query processing. However, they suffer from either large update cost or poor query performance. Motivated by this, another index called B^{dual}-tree [10] is designed to handle both updates and queries efficiently. B^{dual}-tree is a B^+-tree that indexes moving points according to their Hilbert values in the dual space defined by the location and velocity dimensions.

Historical Background

Existing predictive spatiotemporal structures can be classified into 3 categories: dual space indexes, time parameterized indexes, and space filling curve indexes.

The *Hough-X representation* of a 2D moving point o is a vector ($o.v[1]$, $o[1]$, $o.v[2]$, $o[2]$) where $o.v[i]$ is its velocity along dimension i ($1 \le i \le 2$), and $o[i]$ is its i-th coordinate at the (past) reference time t_{ref}. Patel et al. [6] propose STRIPES, to index the 4D Hough-X representation of 2D moving points by a PR bucket quadtree. Object updates are processed fast because only a single path of the tree needs to be accessed. However, a leaf node in a quadtree may contain an arbitrarily small number of entries, and hence, more pages need to be accessed to obtain the same number of results. Although [6] suggest a "half-page" storage scheme to alleviate the problem, the query performance of STRIPES may still be high.

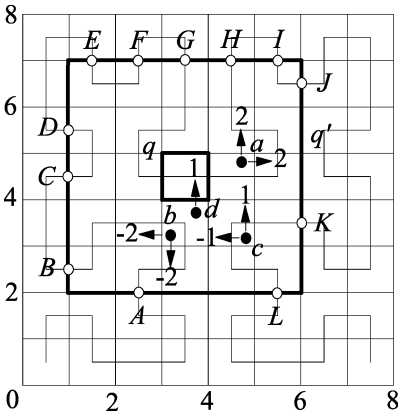
Saltenis et al. [7] propose the TPR-tree (later improved in [9]) that augments R-trees [2] with velocities to index moving objects. Figure 2a shows the locations of 4 objects at time 0, and the arrows indicate their movements. Figure 2b illustrates the object locations at time stamp 1. A node in the TPR-tree is represented as a *moving rectangle* (MOR), which includes (i) a SBox, a rectangle that tightly encloses the locations of the underlying objects at



Indexing, BDual Tree, Figure 2 A TPR-tree example. **a** SBox/VBox at time 0. **b** Node extents at time 1

time 0, and (ii) a VBox, a vector bounding their velocities. Observe that, in order to achieve good query performance, objects with similar motion parameters are grouped in the same TPR tree node. Consider a range query at time 1 whose search region q is the shaded rectangle in Fig. 2b. Since N_1 at time 1 does not intersect q, it does not contain any result, and can be pruned from further consideration. On the other hand, the query examines N_2, which contains the only qualifying object c. The MOR of a node grows with time such that it is guaranteed to enclose the locations of the underlying objects at any future time stamp, although it is not necessarily tight. Thus, during updates, MORs of nodes are tightened in order to optimize the performance of processing forthcoming queries. Due to the expensive node tightening triggered by object updates, TPR-tree is not feasible for real-life applications with frequent updates.

Jensen et al. [3] propose the B^x-tree, which consists of B^+-trees indexing the transformed 1D values of moving objects based on a space filling curve (e. g., Hilbert curve). Figure 3 shows an exemplary B^x-tree on 4 moving points. The location of an object at the reference time (e. g., 0) is mapped to a Hilbert value, which is indexed by a B^+-tree. Object updates are highly efficient by reusing the B^+ insertion/deletion procedures. Consider, for example, the small rectangle in Fig. 3 as a range query q at time stamp 1. First, q is expanded to another rectangle q' by the maximum object speed (i. e., 2) such that all actual results (i. e., any object intersecting q at time 1) are guaranteed to fall also in q' at time 0. Then, the region q' is decomposed into disjoint, consecutive Hilbert intervals and corresponding interval queries are executed on the B^+-tree for retrieving all points located inside q'. For each retrieved object, its actual location and velocity are verified against the original query q. Since expanding the query based on the maximum velocities of the entire dataset may lead to an excessively large number of false hits (e. g., objects a and b), the query performance of B^x-trees could be worser than that of TPR-trees.



Indexing, BDual Tree, Figure 3 A B^x-tree example

Scientific Fundamentals

With the exception of TPR-trees [7,9] (which optimize query performance by tightening nodes during updates), the query performance of the existing spatiotemporal indexes [3,6] degrades over time. In order to alleviate this problem, two trees (with different expiration time) are used for indexing objects alternatively. In the following, the above approach is adopted and two B^+-trees BT_1 and BT_2 are used alternatively for indexing objects. Interested readers may refer to [10] for details. Since BT_1 and BT_2 have the same structure, in the following, B^{dual}-tree is considered as a single tree.

A B^{dual}-tree has two parameters: (i) a *horizon* H, for deciding the farthest future time that can be efficiently queried, and (ii) a *reference time* T_{ref}, for converting moving points to their duals. A d-dimensional (in practice, $d=2$ or 3) moving point o is represented with

- a *reference time stamp* $o.t_{ref}$ (e. g., the last update time of o),
- its coordinates $o[1], o[2], \ldots, o[d]$ at time $o.t_{ref}$, and
- its current velocities $o.v[1], o.v[2], \ldots, o.v[d]$.

For example, object o_1 in Fig. 1 has reference time $o_1.t_{ref} = 0$, coordinates $o_1[1] = 2$, $o_1[2] = 9$, and velocities $o_1.v[1] = 1$, $o_1.v[2] = -2$. The vector $o(t) = (o[1](t), o[2](t), \ldots, o[d](t))$ is used to denote the location of o at a time stamp $t \geq o.t_{ref}$, where, for $1 \leq i \leq d$:

$$o[i](t) = o[i] + o.v[i] \cdot (t - o.t_{ref}) . \quad (1)$$

The dual representation of an object o is a $2d$-dimensional vector:

$$o^{dual} = (o[1](T_{ref}), \ldots, o[d](T_{ref}), o.v[1], \ldots, o.v[d]). \quad (2)$$

In other words, o^{dual} is a point in a $2d$-dimensional *dual space*, which contains d *location dimensions* $o[i](T_{ref})$ (for the first d components of o^{dual}) and d *velocity dimensions* $o.v[i]$. The dual space can be mapped to a 1D domain by any space filling curve. The Hilbert curve is chosen because it preserves the spatial locality better than other curves [3], achieving lower query cost.

Given the *resolution* λ (an integer), the dual space can be viewed as a *partitioning grid* of $2^{\lambda \cdot 2d}$ regular cells and each dimension has 2^{λ} divisions. Figure 4 depicts the dual space of 1D moving objects (i. e., $d=1$) at resolution level $\lambda = 3$. The number in each cell represents its Hilbert value, which can be computed by a standard algorithm [1].

Objects whose duals o^{dual} fall in the same cell have identical Hilbert values, which are indexed by a B^+-tree, called the B^{dual}-tree. In the tree, each leaf entry stores the detailed dual representation of an object (i. e., locations at T_{ref}, and velocities).

Updates

Each object o issues an update whenever its velocity changes. Let $o^{\text{dual}}_{\text{old}}$ ($o^{\text{dual}}_{\text{new}}$) be the old (new) dual representation of the object. $o^{\text{dual}}_{\text{old}}$ is removed from the tree and then $o^{\text{dual}}_{\text{new}}$ is inserted into the tree. An insertion/deletion is performed in the same way as a B^+-tree, by accessing $O(\log N)$ pages where N is the dataset cardinality.

Specifying a Range Query

A d-dimensional moving rectangle (MOR) r is captured by

- a *reference time stamp* $r.t_{\text{ref}}$,
- a *spatial box* (SBox), a $2d$-dimensional vector ($r_{\vdash}[1]$, $r_{\dashv}[1]$, …, $r_{\vdash}[d]$, $r_{\dashv}[d]$), where $[r_{\vdash}[i], r_{\dashv}[i]]$ is the i-th ($1 \le i \le d$) projection of r at time $r.t_{\text{ref}}$, and
- a *velocity box* (VBox), a $2d$-dimensional vector ($r.V_{\vdash}[1]$, $r.V_{\dashv}[1]$, …, $r.V_{\vdash}[d]$, $r.V_{\dashv}[d]$), where $r.V_{\vdash}[i]$ (or $r.V_{\dashv}[i]$) indicates the velocity of the left (or right) edge on the i-th dimension.

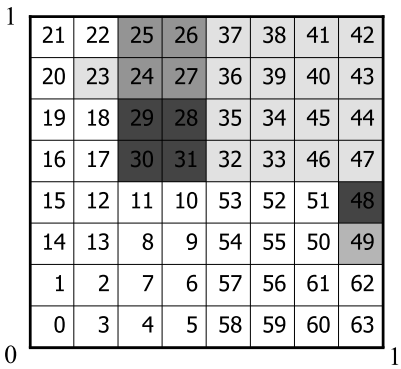
Denoting the spatial extents of r at a time stamp $t \ge r.t_{\text{ref}}$ as $r(t) = (r_{\vdash}[1](t), r_{\dashv}[1](t), \ldots, r_{\vdash}[d](t), r_{\dashv}[d](t))$, is expressed by:

$$r_{\vdash}[i](t) = r_{\vdash}[i] + r.V_{\vdash}[i] \cdot (t - r.t_{\text{ref}})$$
$$r_{\dashv}[i](t) = r_{\dashv}[i] + r.V_{\dashv}[i] \cdot (t - r.t_{\text{ref}}) .$$

A range query specifies a time interval $qt = [qt_{\vdash}, qt_{\dashv}]$, and an MOR q whose reference time is $qt_{\vdash}$. For instance, for the range search in Fig. 1, $qt = [0, 2]$, and the query q_1 is an MOR with reference time 0, SBox (2, 3, 3, 4), and VBox (1, 2, 1, 2). An object o satisfies q if $o(t)$ falls in $q(t)$ for some $t \in qt$.

The Dual Space

Next, the dual space will be studied in more detail. Observe that any cell c in the partitioning grid can be regarded as a d-dimensional MOR (moving rectangle) whose SBox (VBox) captures the projection of the cell on the location (velocity) dimensions of the dual space. Figure 4 shows an example where $d = 1$, and the dual space has $2d = 2$ dimensions. The partitioning grid contains $2^{3 \cdot 2} = 64$ cells (i. e., the resolution $\lambda = 3$), and the number in each cell is the Hilbert value (of any point inside). The cell 53, for example, has a 1D SBox [0.5, 0.625] (its projection on the horizontal axis) and a VBox [0.375, 0.5], assuming that all the dimensions have a domain [0, 1].



Indexing, BDual Tree, Figure 4 Hilbert range decomposition ($d = 1$, $\lambda = 3$)

Given a range query q (an MOR), objects in a cell c need to be inspected if and only if the MOR of c intersects q at some time within the query interval qt. For example, assume $T_{\text{ref}} = 0$ and let c be the cell in Fig. 4 with value 53. According to the SBox and VBox of c, the spatial extent of c at time 1 is $c(1) = [0.5 + 0.375, 0.625 + 0.5] = [0.875, 1.125]$. For a query with $q = [0.7, 0.8]$, $q.V = [0.1, 0.1]$, and $qt = [0, 1]$, all the objects with Hilbert value 53 must be examined because $q(1) = [0.8, 0.9]$ intersects $c(1)$; otherwise, some actual results may be missed. The intersection algorithm in [7] can be applied to determine whether two MORs intersect at some time within a specified time interval.

Hilbert Interval Decomposition

In fact, as a property of the B^+-tree, an intermediate entry e is associated with an interval $[e.h_{\vdash}, e.h_{\dashv})$, which contains the Hilbert values of all the objects in the subtree. $[e.h_{\vdash}, e.h_{\dashv})$ is referred as the *Hilbert interval* of e. Each integer in the interval corresponds to a cell in the partitioning grid. As mentioned before, each cell can be regarded as an MOR, and thus, e can be trivially decomposed into $e.h_{\dashv} - e.h_{\vdash}$ MORs. However, the number of these MORs can be $2^{\lambda \cdot 2d}$ in the worst case (i. e., all the cells in the grid), such that the resulting query algorithms incur expensive CPU cost.

The goal is to break $[e.h_{\vdash}, e.h_{\dashv})$ into several disjoint intervals, such that the union of the cells in each interval is a hyper-square in the dual space. Figure 5 presents a hierarchical algorithm for decomposing a Hilbert interval (for a non-leaf entry e) into MORs. [10] proves the correctness of the algorithm and stated that an interval is decomposed into at most $(4^d - 1) \cdot (2\lambda - 1)$ MORs. Since $d = 2$ or 3 in most real applications, the computational cost is essentially linear to the resolution λ. In practice, with the typical parameter values $d = 2$ and $\lambda = 10$, the decomposition technique generates at most $(4^d - 1) \cdot (2\lambda - 1) = 285$

```
Algorithm Decompose(a Hilbert Interval HI)
1.  S:=∅; // S will contain the decomposed perfect MORs eventually
2.  r0:= the MOR covering the entire dual space;
3.  ω0:= the interval of the Hilbert domain;
4.  L:={(r0, ω0)}; // L is a FIFO queue
5.  while (L is not empty)
6.     remove the first element (r, ω) of L; //r is a perfect MOR
7.     if (ω intersects HI) then
8.        if (ω is covered by HI) then
9.           add r to S;
10.       else if (the length of ω > 1) then
11.          divide r into 4^d identical perfect MORs;
12.          for each resulting MOR r' and its Hilbert interval ω'
13.             add (r', ω') to L;
14. return S;
```

Indexing, BDual Tree, Figure 5 Decomposing a Hilbert interval

MORs, which is much smaller than the number of MORs ($2^{\lambda 2d} = 1.1 \cdot 10^{12}$) produced by the trivial decomposition approach.

Figure 4 illustrates how the hierarchical decomposition algorithm works for the Hilbert interval [23, 49]. First, the 4×4 MORs of the dual space are considered: the intervals [0, 15], [16, 31], [32, 47], [48, 63]. A large MOR can be extracted from the interval [32, 47] because it is covered by [23, 49]. On the other hand, [23, 49] partially covers [16, 31]. Thus, the above technique is applied recursively on [16, 32] and consider its 2×2 MORs: the intervals [16, 19], [20, 23], [24, 27], [28, 31]. Now, two MORs can be extracted from the intervals [24, 27], [28, 31] because they are covered by [23, 49]. Again, [23, 49] partially covers [20, 23] so the 1×1 MORs of [20, 23] are considered and a MOR for [23,23] is extracted. Similarly, the other end of the interval [23, 49] can be decomposed into MORs for the intervals [48, 48], [49, 49].

In summary, the interval [23, 49] can be broken into 6 intervals [23, 23], [24, 27], [28, 31], [32, 47], [48, 48], [49, 49] satisfying the above condition. In particular, the cells in [23, 23], [24, 27], [32, 47] constitute 1×1, 2×2, and 4×4 squares, respectively. Each resulting square can be regarded as an MOR whose projection on a location/velocity dimension is identical to that of the square (e. g., [23, 49] can be associated with 6 MORs). Note that the actual number of MORs produced is usually much smaller than the upper bound $(4^d - 1) \cdot (2\lambda - 1)$ (e. g., the number 6 for the interval [23, 49] in Fig. 4 is much lower than the upper bound $(4^1 - 1) \cdot (2 \cdot 3 - 1) = 15$).

Query Processing

It is ready to discuss how to process a range query with a B^{dual}-tree. Let e be an intermediate entry, which can be decomposed into m MORs $r_1, r_2, \ldots, r_{\text{m}}$ ($m \leq (4^d - 1) \cdot (2\lambda - 1)$), by the algorithm of Fig. 5. Given a range query q, the subtree of e is pruned if no r_{i} ($1 \leq i \leq m$) intersects q during the query interval qt. The processing algorithm starts by checking, for each root entry e, whether any of its associated MORs intersects q during qt (in the same way as in TPR-trees [7]). If yes, the algorithm accesses the child node of e, carrying out the process recursively until a leaf node is reached. Then, detailed information of each object encountered is simply examined against the query predicate.

It is worth noticing that a B^{dual}-tree is as powerful as a TPR-tree in terms of the queries that can be supported. Intuitively, intermediate entries of both structures can be represented as MORs. Thus, an algorithm that applies to a TPR-tree can be adapted for the B^{dual}-tree. Adaptation is needed only because an entry of a TPR-tree has a single MOR, while that of a B^{dual}-tree corresponds to multiple ones. For instance, [10] discuss how B^{dual}-tree can be used to evaluate other more complex queries (e. g., predictive NN queries).

Key Applications

Range search is one of the most important operations in spatiotemporal databases. For instance, to perform flight control, an airport must continuously keep track of the aircrafts about to enter its vicinity in near future; this can be achieved by executing a range query periodically: "report the aircrafts that, in 10 minutes, will appear in the circle centering at the control tower with a radius of 20 miles". In a highway traffic monitoring system, on the other hand, a useful range query would "return the vehicles that are expected to enter Washington DC in 5 minutes". Besides being a useful stand-alone operator, range search is also the building block for complex retrieval tasks. For example, given a set of aircrafts, a *self distance join* would "find all pairs of aircrafts that will be within 50 miles from each other in 10 minutes". A strategy to process the join is to

issue a range query for each aircraft. Specifically, the query region is a moving circle, which always centers at the aircraft and has a radius of 50 miles.

Future Directions

Like B^x-tree, the B^{dual}-tree assumes that each moving object is a point moving with constant velocity. However, the above assumption may not hold for all real-life applications.

Moving Objects with Extent

In practice, a moving object may refer to an object with extent (e. g., rectangle, circle). The B^{dual}-tree indexes moving points by mapping them to points in the dual space. Thus, it may not be directly applicable for moving objects with extents. It remains an open question whether an effective mapping technique can be developed such that a moving object (with extent) can be mapped to a single point in a transformed space.

Non-linear Moving Objects

Also, in the B^{dual}-tree, a point is assumed to be moving linearly with constant velocity. For instance, if an object (e. g., car) accelerates/decelerates or moves with circular motion (e. g., along the curve in a race track), then the object has to issue a large number of updates to the server. A better spatiotemporal index not only reduces the effort spent by the server, but also allows the objects to save energy by reducing their update frequencies. [8] generalizes TPR-trees by modeling the motion of an object as a motion matrix instead of a linear motion function. It is interesting to study how B^{dual}-tree can be extended for indexing non-linear moving objects such that the number of updates can be reduced.

Cross References

► Indexing of Moving Objects, B^x-Tree
► Space-Filling Curves

References

1. Butz, A.R.: Alternative Algorithm for Hilbert's Space-Filling Curve. IEEE Trans. Comput. **C-20**(4), 424–426 (1971)
2. Guttman, A.: R-trees: A Dynamic Index Structure for Spatial Searching. In: Proc. of ACM Management of Data (SIGMOD) (1984)
3. Jensen, C.S., Lin, D., Ooi, B.C.: Query and Update Efficient B^+-Tree Based Indexing of Moving Objects. In: Proc. of Very Large Data Bases (VLDB) (2004)
4. Kollios, G., Gunopulos, D., Tsotras, V.J.: On Indexing Mobile Objects. In: Proc. of ACM Symposium on Principles of Database Systems (PODS) (1999)
5. Kollios, G., Papadopoulos, D., Gunopulos, D., Tsotras, V.J.: Indexing Mobile Objects Using Dual Transformations. The VLDB Journal **14**(2):238–256 (2005)
6. Patel, J.M., Chen, Y., Chakka, V.P.: STRIPES: An Efficient Index for Predicted Trajectories. In: Proc. of ACM Management of Data (SIGMOD) (2004)
7. Saltenis, S., Jensen, C.S., Leutenegger, S.T., Lopez, M.A.: Indexing the Positions of Continuously Moving Objects. In: Proc. of ACM Management of Data (SIGMOD) (2000)
8. Tao, Y., Faloutsos, C., Papadias, D., Liu, B.: Prediction and Indexing of Moving Objects with Unknown Motion Patterns. In: Proc. of ACM Management of Data (SIGMOD) (2004)
9. Tao, Y., Papadias, D., Sun, J.: The TPR^*-Tree: An Optimized Spatio-temporal Access Method for Predictive Queries. In: Proc. of Very Large Data Bases (VLDB) (2003)
10. Yiu, M.L., Tao, Y., Mamoulis, N.: The B^{dual}-Tree: Indexing Moving Objects by Space-Filling Curves in the Dual Space. The VLDB Journal, To Appear

Indexing Framework, Spatial/Spatio-temporal

► Index Structures, Extensible

Indexing, High Dimensional

MICHAEL GIBAS, HAKAN FERHATOSMANOGLU
Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA

Synonyms

Multi-dimensional indexing; Multi-dimensional access structures; Indexing, metric-space; Quantization; Time-series; VA-file; M-tree

Definition

High-dimensional indexing covers a number of techniques that are intended to offer faster access to high-dimensional data sets than a traditional sequential scan of the data itself. An index can provide more efficient query response by pruning the data search space. It is possible that increasing the dimensionality of an index can prune more search space at the cost of a larger index structure and more complicated index traversal. However, when the index dimensionality is too large, traditional multi-dimensional access structures are no longer effective. Techniques were developed that addressed the performance issues associated with high-dimensional indexes by 1) modifying existing techniques to expand the range of dimensionality for which they would be effective, 2) indexing objects based on characteristics of interest and 3) reading all data in much smaller quantized form.

Historical Background

As memory became less expensive, useful data sets became much larger in terms of both the raw size of the data and in terms of the number of attributes (dimensions) stored for each record. Additionally, data exploration tasks became more complicated. Complex objects were represented by *feature vectors*, a translation of the object into a series of attribute values. Objects were compared according to their similarity, a measure of closeness of one object's feature vector to another. Due to the size of the data sets it was too costly to read all records in the data set in order to answer a query. Due to the dimensionality of the data sets, the existing multi-dimensional access structures were not effective in efficiently answering queries. This led to the development of many techniques targeted toward addressing efficient query processing for large, high-dimensional data sets.

A data hierarchical multi-dimensional access structure, the R-tree, was first introduced in [8]. Many structures such as the X-tree [3] attempted to mitigate the performance issues associated with the R-tree at higher dimensionality. Other techniques were introduced [5] that index a metric abstraction of the data in order to efficiently answer a specific query type like nearest neighbor. Because of the severe performance degradation of multi-dimensional access structures at high-dimensionality, the VA-file was introduced [15] that essentially speeds up sequential scan.

Scientific Fundamentals

Typical Query Types

Index structure processing and effectiveness is partially dependent on query type. So first some common query types are discussed. A point query determines if a specific point exists in the database. A range query provides a set of points that exist within a given subspace. Similarity or Nearest Neighbor-type queries determine a set of points within the database that are 'closest' to the query point with respect to some distance function, such as Euclidean distance. Without indexing support, the point query can be answered by checking if any point matches the point query criteria. The range query is answered by determining which data objects have attribute values that fall within all the query criteria. The nearest neighbor query can be addressed by computing the distance function of each data object and keeping track of the closest ones.

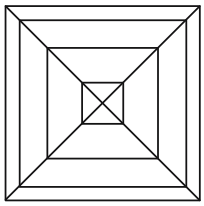
Multi-dimensional Access Structures

Multi-dimensional access structures were developed in order to take advantage of additional pruning that could take place in multiple dimensions that could not occur in a single dimension. A wide range of techniques are described in surveys about multi-dimensional indexing structures [4,7]. The structures typically fall into 2 categories, hierarchical data-partitioning structures and space partitioning structures. These structures can be used to answer the aforementioned common query types. Point queries can be answered by examining the data points falling within the partition(s) that contain the query point and checking for equality compared to the query point. Range queries are answered by examining the data points in all partitions that intersect with the space represented by the range and checking that they meet the query criteria. Nearest neighbor queries can be answered by keeping track of the best points examined thusfar and pruning those partitions that can not yield a better result.

These structures can work well when indexing a few dimensions, however they break down with respect to performance when the indexed dimensionality becomes too high. This is a phenomenon known as the curse of dimensionality. The data partitioning techniques break down because at higher dimensions, the hyper-shapes that could contain query matching objects increasingly overlap, and in order to find all potential query matches nearly the entire structure needs to be traversed. Non-overlapping space partitioning techniques break down because of increased size of the index structure at higher dimensions. As the dimensionality increases, the number of empty partitions increases exponentially.

Techniques have been introduced to mitigate the effect of the curse of dimensionality. Because data-hierarchical access structures degenerate at higher dimensions due to overlaps in the hyperspaces that are searched in order to find query matches, the X-tree [3] reduces these overlaps by allowing oversize nodes. This mitigates the effect of needing to search all the subspaces that overlap the query region at the cost of longer linear scan times of the supernodes. This structure can be effective for higher dimensions than other hierarchical data-partitioning structures, but still performs worse than sequential scan as the dimensionality becomes large enough.

The pyramid-technique [2] is a partitioning technique where the data space is subdivided into non-overlapping subspaces. Figure 1 shows a sample subdivision of data space in two dimensions. After dividing the space, the d dimensional space is transformed into a 1-dimensional space, which can be indexed using a B+-tree [1]. A range query is processed by examining data points in partitions that intersect with the query region. The technique does not suffer degraded performance as dimensionality increases when the query regions are not skewed. However, performance issues do arise when the query regions and data are skewed. There is also not a straightforward way to perform



Indexing, High Dimensional, Figure 1
Sample Pyramid-Technique Space Partitioning in 2-D

similarity type queries, since data points that are very close could be contained in partitions that are not close to each other.

Metric Space Indexing

Because many common queries for high-dimensional data are based on finding nearest neighbor or similar objects to some query object, metric space indexes were developed to provide such query support. The general idea is to index objects based on distances from parent substructures distances between objects and substructures and maintain these in a tree-like index. When a nearest neighbor query is performed, the tree is traversed by pruning paths that can not match the query.

An example of a metric space indexing structure is the M-tree. The M-Tree [5] is a balanced tree structure that associates data points to nodes based on the distance of the point from the node. A node consists of a spatial point, a covering radius, and a distance to the parent node. The M-tree is constructed by locating and inserting new points into the most suitable node. The most suitable node is one whose covering radius increases the least as a result of inserting the point. In the case that point lies in multiple potential nodes it is placed in the spatially closest node. Nearest neighbor queries are performed by traversing the structure using a branch-and-bound technique [12]. A queue of active subtrees that could contain matching points is maintained. Nodes are examined in order of how promising they are according to some heuristic and the distance of the current nearest neighbor is dynamically updated. When this distance is updated, it may be possible to prune candidate subtrees that could not contain a point that beats this distance.

These techniques can be effective for similarity searches using a given distance function. However, the index is only effective for the for the subset of attributes and distance function over which it was built. A new index will need to be constructed for each distance function or subset of attributes that are being queried.

Quantization Based Structures

Another class of techniques for high-dimensional data access is based on quantizing data. These techniques quantize the data into an approximate, but much smaller representation. Processing works by examining the approximated records and determining a set of records that could match the query.

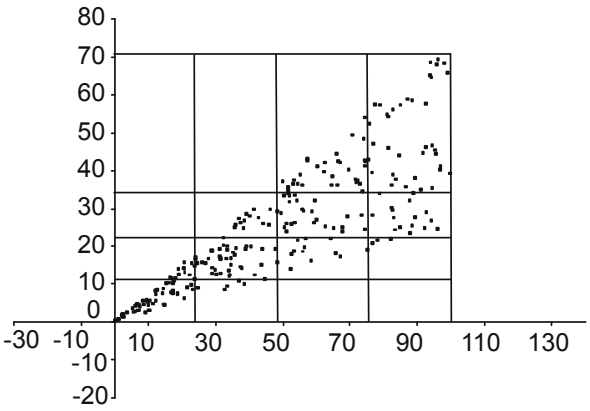
Indexing, High Dimensional, Table 1 Sample VA-File

Record ID	Attribute A	Attribute B	VA(A)	VA(B)
1	7.29	30.6	0000	0011
2	92.08	42.04	1011	0101
3	31.84	92.77	0011	1011
4	7.94	70.92	0000	1000
5	90.18	65.76	1011	1000
6	47.47	124.17	0101	1111
7	96.74	61.39	1100	0111
8	60.6	90.94	0111	1011
9	87.8	18.53	1010	0010
10	84.91	82.13	1010	1010
11	123.34	53.74	1111	0110
12	46.58	14.75	0101	0001
13	99.27	96.45	1100	1100
14	37.97	9.43	0100	0001
15	73.96	103.99	1001	1100
16	104.66	108.26	1101	1101
17	21.61	88.24	0010	1011
18	2.18	100.29	0000	1100
19	90.13	67.15	1011	1000
20	23.49	70.19	0010	1000

The VA-File (Vector Approximation) structure was the first such structure to follow this approach. Table 1 shows a simple example. Attributes A and B show the actual value for the object. Columns VA(A) and VA(B) show the vector approximation of the values. In this case the values are derived by using 4 bits for each attribute to divide the data space into equal width partitions.

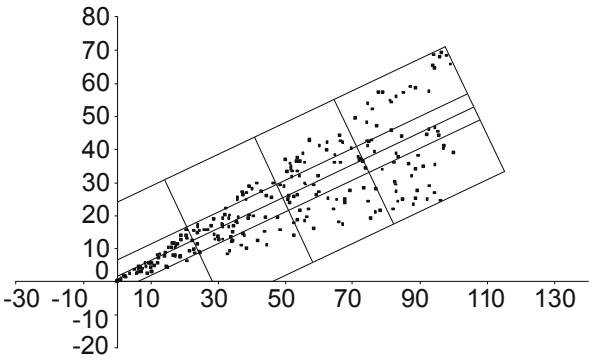
Point and range query processing works by first converting the query to cover the buckets that could answer the query. For example, if our query of interest was a range query where attribute A is between 37 and 47, and attribute B is between 10 and 20, the query would be converted to a VA query of VA(A) between 0100 and 0101 and VA(B) between 0001 and 0010. Then the VA-File would be traversed and any record's representation that intersected with the range queries approximated space would be identified. A second pass would go to disk to read the actual values and determine which of these records actually answered the query. In the sample query, objects 12 and 14 would meet the VA-file criteria with point 14 representing a false positive from the first pass that would be pruned in the second pass.

The effectiveness of the approximation associated with the VA-File is dependent on the resolution of data into distinct

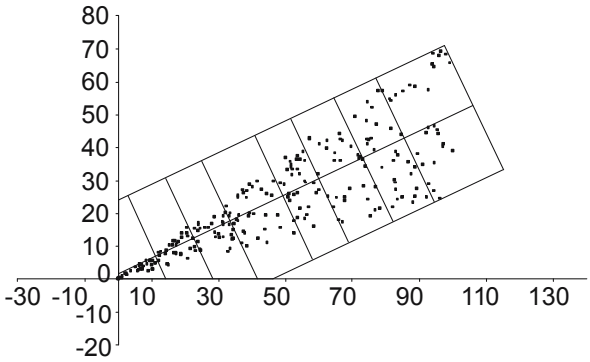


Indexing, High Dimensional, Figure 2 Sample VA-File Partitioning in 2-D

buckets. The VA-File does not take advantage of correlation between dimensions or the amount of energy associated with a given attribute. Figure 2 shows the cell representations for a sample VA-File. If two dimensions are highly correlated, there is little advantage with respect to pruning capability to index both dimensions. As well, if the attribute values for one dimension exhibit much more spread than the values of another dimension, it makes more sense to allocate more bits to differentiate the values in that dimension. The VA+-File [6] addresses both of these concerns by applying a quantizer that better differentiates data into meaningful cell representations. In the VA+-file, the data is first transformed using some technique to minimize the correlation between dimensions such as the Karhunen Loeve transform (KLT) [9,11]. Figure 3 displays a sample cell representation for 4 bit quantization for the same data shown for the VA-File. The budgeted space for a data object can also be better utilized by assigning bits to attributes non-uniformly. Figure 4 shows the same sample data set where 1 bit is assigned to the transformed y dimension and 3 bits are assigned to the transformed x-dimension. Another optimization that can be applied during VA+-file construction if appropriate for the data set is non-uniform quantization. Rather than using equi-populated, or equi-volumed splits, the cells can be split based on data clusters in each single dimension. Using Lloyd's algorithm [10] to determine the split points per dimension can help to keep data that is clustered in that dimension in the same bucket and improve the overall representation quality of the bucket.



Indexing, High Dimensional, Figure 3 Sample VA+-File Partitioning in 2-D, Uniform Bit Allocation



Indexing, High Dimensional, Figure 4 Sample VA+-File Partitioning in 2-D, Non-Uniform Bit Allocation

The advantage of these quantization-based structures is that performance does not get much worse at increased index dimensionality. However, a representation of each record must be visited, which can be time consuming if the number of records is very large or the quantized representation is not much smaller than the original data.

Key Applications

Characteristics of applications that can significantly benefit from high-dimensional indexing support include:

- data sets that are too large to read efficiently or to hold in memory
- data sets that cover many attributes
- applications that query objects based on many attributes (such as similarity queries or high-dimensional range queries)

Multimedia Databases

Multimedia databases are an application of interest with significant importance. Multimedia data is inherently large and is usually represented by a feature vector which describes the original data with a high number of dimensions. The similarity between two objects is defined by a distance function, e. g., Euclidean distance, between the

corresponding feature vectors. A popular query, that makes use of the feature vectors, is the similarity query. For example, in image databases, the user may pose a query asking for the images most similar to a given image. Similarity query with multi-dimensional data is usually implemented by finding the k closest feature vectors to the feature vector of query object, which is known as k-nearest neighbor, k-NN, query. A closely related query is the ε-range query, where all feature vectors that are within ε neighborhood of the query point q are retrieved.

Geographic Databases

Geographic data sets can include the geographic spatial locations as well as attributes associated with objects. Additionally, the data objects themselves can be represented by complex geometric shapes in multiple-dimensions. Indexing techniques need to be able to associate an object with respect to its spatial characteristics as well as its other data characteristics.

Scientific Databases

Scientific databases are a typical example of applications that are both high-dimensional and for which high-dimensional queries are common. Evolution of computing technology has allowed detailed modeling of physical, chemical, and biological phenomena. Some examples from DOE scientific applications include climate simulation and model development, computational biology, high energy and nuclear physics, and astrophysics. Many large data sets in scientific domains contain a large number of attributes that may be queried and analyzed, and therefore considered as high dimensional data. For example, High Energy Physics data in one DOE scientific application contains more than 500 attributes that describe the properties of the objects in experiment data [14]. Various types of queries, such as partial match query and range query, are executed on these large data sets to retrieve useful information for scientific discovery. Astrophysics data has many of the same characteristics as geographic data. Data objects have spatial location or spatial shape as well as other attributes.

Biological Databases

Biological Databases are another example of high-dimensional data sets that can benefit from high-dimensional search support. An increasing number of biological databases, such as bio-sequence databases and biomedical data warehouses, are available online and have been used by many researchers to generate new knowledge. Queries asking sequence similarity are widely used to capture interesting and useful information from these bio-sequence databases. For example, the similarity of subsequences of a genome data to a query sequence is used to predict some diseases in advance, or to find some functional or physical relation between different organisms.

Text Databases

Many applications for document databases involve finding similar documents with respect to their intended meaning. As opposed to finding exact matches for certain search terms (as is common for many internet search engines), users could benefit from finding a set of documents reflecting the same context. Much like the image example, a text document can be converted to a high-dimensional representation of its contents. Similarity, or semantic closeness, can then be approximated by comparing the feature vectors of documents [13].

Time Series Data

The time series domain is another fertile area that can benefit from high-dimensional indexing. Many applications compare the similarity of objects with respect to attributes that reflect measurements taken at time intervals. Effective indexing of the time-series data allows efficient data exploration to find correlations and cause and effect relationships between data objects. Financial/business analysis and environmental monitoring are two such application domains.

Future Directions

Future solutions will endeavor to enhance performance for each of the classes of current techniques, both for data sets in general and also targeting specific query types for specific applications.

Cross References

- ▶ Indexing, X-Tree
- ▶ Nearest Neighbor Query
- ▶ Pyramid Technique
- ▶ R-Trees – A Dynamic Index Structure for Spatial Searching

Recommended Reading

1. Bayer, R., McCreight, E.: Organization and maintenance of large ordered indexes. In: Software Pioneers - Contributions to software engineering, pp. 245–262 (2002)
2. Berchtold, S., Böhm, C., Kriegel, H.-P.: The pyramid-tree: Breaking the curse of dimensionality. In: Haas, L.M., Tiwary, A. (eds.) SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2–4, 1998, Seattle, Washington, USA, pp. 142–153. ACM Press, New York (1998)

3. Berchtold, S., Keim, D., Kriegel, H.-P.: The x-tree: An index structure for high-dimensional data. In: VLDB '96: Proceedings of the 22th International Conference on Very Large Data Bases, pp. 28–39. Morgan Kaufmann Publishers Inc, San Francisco, CA (1996)
4. Böhm, C., Berchtold, S., Keim, D.: Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. ACM Comput. Surv. **33**(3):322–373 (2001)
5. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An efficient access method for similarity search in metric spaces. In: VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases, pp. 426–435. Morgan Kaufmann Publishers Inc, San Francisco, CA (1997)
6. Ferhatosmanoglu, H., Tuncel, E., Agrawal, D., El Abbadi, A.: Vector approximation based indexing for non-uniform high dimensional data sets. In: CIKM '00: Proceedings of the ninth international conference on Information and knowledge management, pp. 202–209. ACM Press, New York, NY (2000)
7. Gaede, V., Günther, O.: Multidimensional access methods. ACM Comput. Surv. **30**(2):170–231 (1998)
8. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data, pp. 47–57. ACM Press, New York, NY (1984)
9. Karhunen, H.: Über lineare Methoden in der Wahrscheinlichkeitsrechnung. Ann. Acad. Science Fenn (1947)
10. Lloyd, S.P.: Least squares quantization in pcm. IEEE Transactions on Information Theory, **28**(2), 127–135 (1982)
11. Loeve, M.: Fonctions aleatoires de seconde ordre. Processus Stochastiques et Mouvement Brownien, Gauthier-Villars, Paris (1948)
12. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. In: SIGMOD (1995)
13. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**, 613–620 (1975)
14. SciDAC: Scientific data management center. http://sdm.lbl.gov/sdmcenter/ (2002)
15. Weber, R., Schek, H.-J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: VLDB (1998)

Indexing, Hilbert R-Tree, Spatial Indexing, Multimedia Indexing

IBRAHIM KAMEL
University of Sharjah, Sharjah, UAE

Synonyms

Database indexing; Multidimensional index; Spatial indexing; Multimedia indexing

Definition

Hilbert R-tree, an R-tree variant, is an index for multidimensional objects like lines, regions, 3-D objects, or high dimensional feature-based parametric objects. It can be thought of as an extension to B+-tree for multidimensional objects.

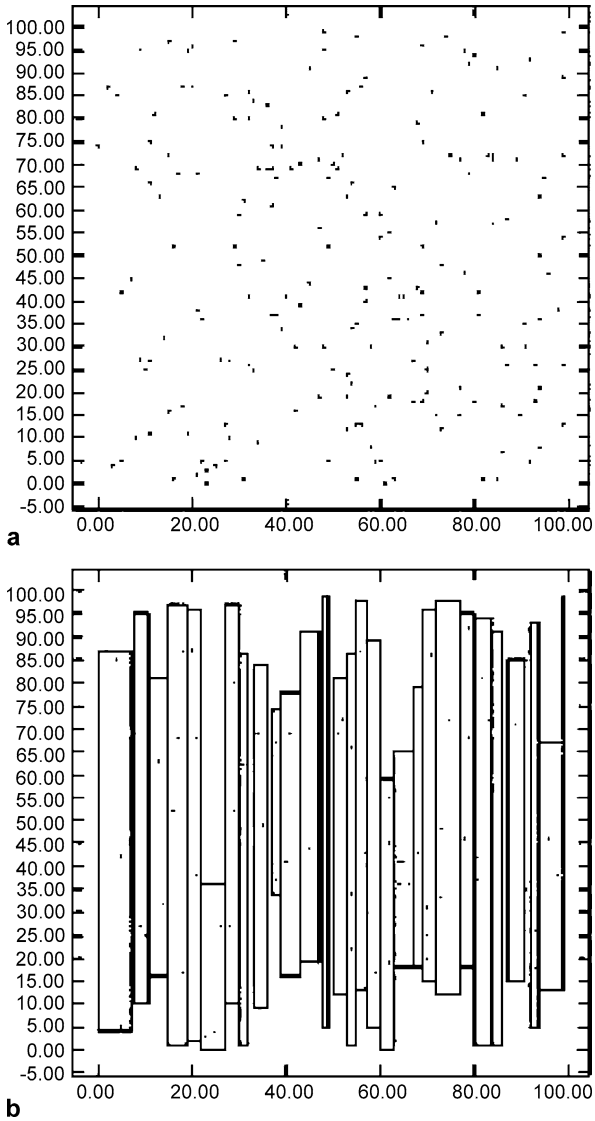
The performance of R-trees depends on the quality of the algorithm that clusters the data rectangles on a node. Hilbert R-trees use space filling curves, specifically the Hilbert curve, to impose a linear ordering on the data rectangles.

There are two types of Hilbert R-tree, one for static database and one for dynamic databases. In both cases, space filling curves and specifically the Hilbert curve are used to achieve better ordering of multidimensional objects in the node. This ordering has to be 'good' in the sense that it should group 'similar' data rectangles together to minimize the area and perimeter of the resulting minimum bounding rectangles (MBRs). Packed Hilbert R-trees are suitable for static databases in which updates are very rare or in which there are no updates at all.

The dynamic Hilbert R-tree is suitable for dynamic databases where insertions, deletions, or updates may occur in real time. Moreover, dynamic Hilbert R-trees employ a flexible deferred splitting mechanism to increase the space utilization. Every node has a well-defined set of sibling nodes. By adjusting the split policy, the Hilbert R-tree can achieve a degree of space utilization as high as is desired. This is done by proposing an ordering on the R-tree nodes. Hilbert R-tree sorts rectangles according to the Hilbert value of the center of the rectangles (i. e., MBR). Given the ordering, every node has a well-defined set of sibling nodes; thus, deferred splitting can be used. By adjusting the split policy, the Hilbert R-tree can achieve as high a utilization as desired. To the contrary, other R-tree variants have no control over the space utilization.

Historical Background

Although the following example is for a static environment, it explains the intuitive principals for good R-tree design. These principals are valid for both static and dynamic databases. Roussopoulos and Leifker proposed a method for building a packed R-tree that achieves almost 100% space utilization. The idea is to sort the data on the x or y coordinate of one of the corners of the rectangles. Sorting on any of the four coordinates gives similar results. In this discussion, points or rectangles are sorted on the x coordinate of the lower left corner of the rectangle. In the discussion below, the Roussopoulos and Leifker's method is referred to as the *lowx* packed R-tree. The sorted list of rectangles is scanned; successive rectangles are assigned to the same R-tree leaf node until that node is full; a new leaf node is then created and the scanning of the sorted list continues. Thus, the nodes of the resulting R-tree will be fully packed, with the possible exception of the last node at each level. Thus, the utilization is $\approx$100%. Higher levels of the tree are created in a similar way.



Indexing, Hilbert R-Tree, Spatial Indexing, Multimedia Indexing, Figure 1 **a** 200 points uniformly distributed. **b** MBR of nodes generated by the *lowx packed R-tree* algorithm

Figure 1 highlights the problem of the *lowx* packed R-tree. Figure 1**b** shows the leaf nodes of the R-tree that the *lowx* packing method will create for the points of Fig. 1**a**. The fact that the resulting father nodes cover little area explains why the *lowx* packed R-tree achieves excellent performance for point queries. However, the fact that the fathers have large perimeters, explains the degradation of performance for region queries. This is consistent with the analytical formulas for R-tree performance [7]. Intuitively, the packing algorithm should ideally assign nearby points to the same leaf node. Ignorance of the y- coordinate by the *lowx* packed R-tree tends to violate this empirical rule.

Scientific Fundamentals

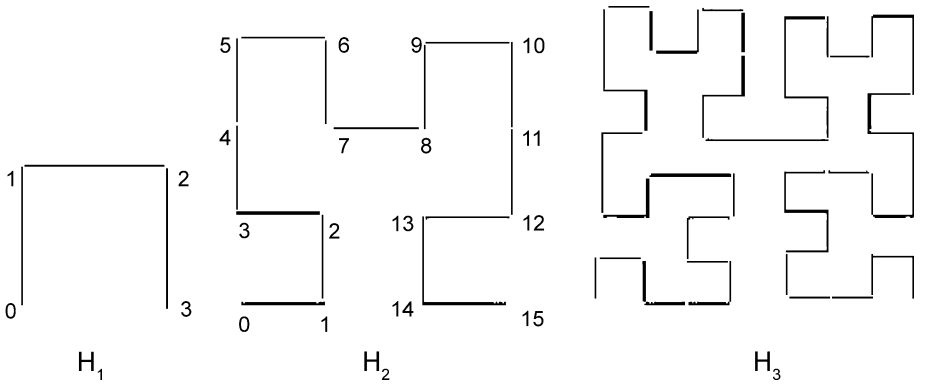
This section describes two variants of the Hilbert R-trees. The first index is suitable for the static database in which updates are very rare or in which there are no updates at all. The nodes of the resulting R-tree will be fully packed with the possible exception of the last node at each level. Thus, the space utilization is ≈100%; this structure is called a packed Hilbert R-tree. The second index supports insertions and deletions and is suitable for a dynamic environment, hence called the *Dynamic Hilbert R-tree*.

Packed Hilbert R-Trees

The following provides a brief introduction to the Hilbert curve. The basic Hilbert curve on a 2×2 grid, denoted by H_1, is shown in Fig. 2. To derive a curve of order i, each vertex of the basic curve is replaced by the curve of order $i-1$, which may be appropriately rotated and/or reflected. Figure 2 also shows the Hilbert curves of order two and three. When the order of the curve tends to infinity, like other space filling curves, the resulting curve is a *fractal* with a fractal dimension of two [5,7]. The Hilbert curve can be generalized for higher dimensionalities. Algorithms for drawing the two-dimensional curve of a given order can be found in [3,5]. An algorithm for higher dimensionalities is given in [2].

The path of a space filling curve imposes a linear ordering on the grid points; this path may be calculated by starting at one end of the curve and following the path to the other end. The actual coordinate values of each point can be calculated. However, for the Hilbert curve, this is much harder than, for example, the Z-order curve. Figure 2 shows one such ordering for a 4×4 grid (see curve H_2). For example, the point (0,0) on the H_2 curve has a Hilbert value of 0, while the point (1,1) has a Hilbert value of 2.

The Hilbert curve imposes a linear ordering on the data rectangles and then traverses the sorted list, assigning each set of C rectangles to a node in the R-tree. The final result is that the set of data rectangles on the same node will be close to each other in the linear ordering and most likely in the native space; thus, the resulting R-tree nodes will have smaller areas. Figure 2 illustrates the intuitive reasons why our Hilbert-based methods will result in good performance. The data is composed of points (the same points as given in Fig. 1). By grouping the points according to their Hilbert values, the MBRs of the resulting R-tree nodes tend to be small square-like rectangles. This indicates that the nodes will likely have a small area and small perimeters. Small area values result in good performance for point queries; small area and small perimeter values lead to good performance for larger queries.



Indexing, Hilbert R-Tree, Spatial Indexing, Multimedia Indexing, Figure 2 Hilbert curves of order 1, 2, and 3

Algorithm Hilbert-Pack:
(packs rectangles into an R-tree)
Step 1. Calculate the Hilbert value for each data rectangle
Step 2. Sort data rectangles on ascending Hilbert values
Step 3. */* Create leaf nodes (level l-0) */*
While (there are more rectangles)
generate a new R-tree node
assign the next C rectangles to this node
Step 4. */* Create nodes at higher level (l + 1) */*
While (there are > 1 nodes at level l)
sort nodes at level $l \geq 0$ on ascending
creation time
repeat Step 3

The assumption here is that the data are static or the frequency of modification is low. This is a simple heuristic for constructing an R-tree with 100% space utilization which at the same time will have as good of a response time as possible.

Dynamic Hilbert R-Trees

The performance of R-trees depends on the quality of the algorithm that clusters the data rectangles on a node. Hilbert R-trees use space-filling curves, specifically the Hilbert curve, to impose a linear ordering on the data rectangles. The Hilbert value of a rectangle is defined as the Hilbert value of its center.

Tree Structure

The Hilbert R-tree has the following structure. A leaf node contains at most C_l entries, each of the form *(R, obj_id)*, where C_l is the capacity of the leaf, R is the MBR of the real object (x_{low}, x_{high}, y_{low}, y_{high}) and *obj-id* is a pointer to the object description record. The main difference between the Hilbert R-tree and the R*-tree [1] is that non-leaf nodes also contain information about the LHVs. Thus, a non-leaf node in the Hilbert R-tree contains at most C_n entries of the form

$$(R, ptr, LHV)\,,$$

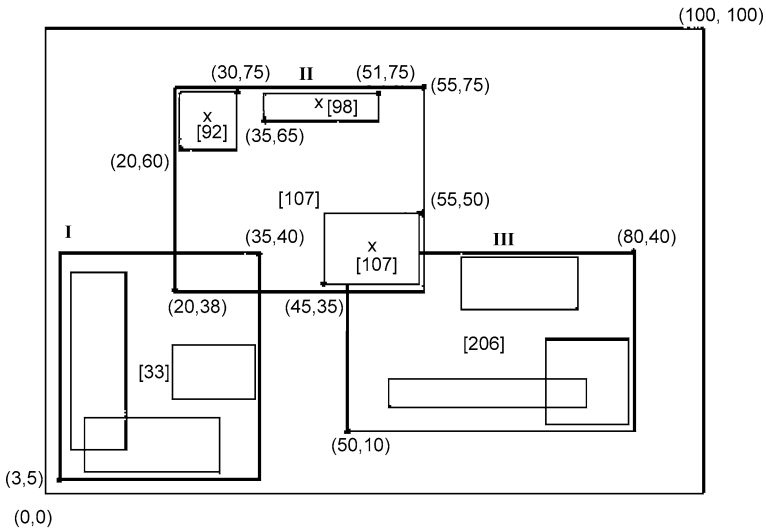
where C_n is the capacity of a non-leaf node, R is the MBR that encloses all the children of that node, ptr is a pointer to the child node, and LHV is the largest Hilbert value among the data rectangles enclosed by R. Notice that since the non-leaf node picks one of the Hilbert values of the children to be the value of its own LHV, there is no extra cost for calculating the Hilbert values of the MBR of non-leaf nodes. Figure 3 illustrates some rectangles organized in a Hilbert R-tree. The Hilbert values of the centers are the numbers near the 'x' symbols (shown only for the parent node 'II'). The LHV's are in [brackets]. Figure 4 shows how the tree of Fig. 3 is stored on the disk; the contents of the parent node 'II' are shown in more detail. Every data rectangle in node 'I' has a Hilbert value $v \leq 33$; similarly every rectangle in node 'II' has a Hilbert value greater than 33 and ≥ 107, etc.

A plain R-tree splits a node on overflow, creating two nodes from the original one. This policy is called a *1-to-2* splitting policy. It is possible also to defer the split, waiting until two nodes split into three. Note that this is similar to the B*-tree split policy. This method is referred to as the 2-to-3 splitting policy. In general, this can be extended to a s-to-$(s+1)$ splitting policy; where s is the order of the splitting policy. To implement the order-s splitting policy, the overflowing node tries to push some of its entries to one of its $s-1$ siblings; if all of them are full, then an s-to-$(s+1)$ split is required. The $s-1$ siblings are called the *cooperating siblings*.

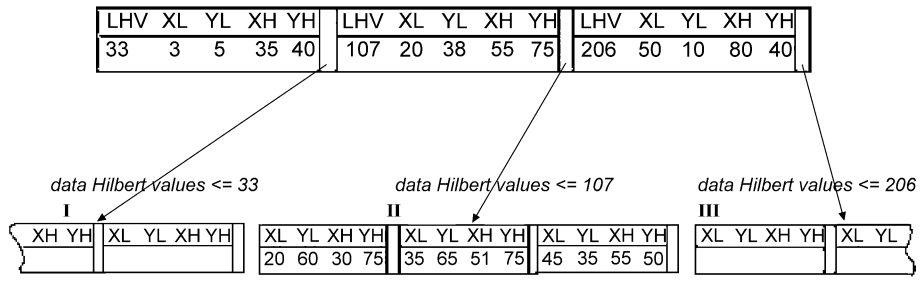
Next, the algorithms for searching, insertion, and overflow handling are described in detail.

Searching

The searching algorithm is similar to the one used in other R-tree variants. Starting from the root, it descends the tree and examines all nodes that intersect the query rectangle. At the leaf level, it reports all entries that intersect the query window w as qualified data items.



Indexing, Hilbert R-Tree, Spatial Indexing, Multimedia Indexing, Figure 3 Data rectangles organized in a Hilbert R-tree (Hilbert values and LHV's are in Brackets)



Indexing, Hilbert R-Tree, Spatial Indexing, Multimedia Indexing, Figure 4 The file structure for the Hilbert R-tree

Algorithm Search(node Root, rect w):
S1. *Search nonleaf nodes:*
Invoke Search for every entry whose MBR intersects the query window *w*.
S2. *Search leaf nodes:*
Report all entries that intersect the query window *w* as candidates.

Insertion

To insert a new rectangle *r* in the Hilbert R-tree, the Hilbert value h of the center of the new rectangle is used as a key. At each level, the node with the minimum LHV of all its siblings is chosen. When a leaf node is reached, the rectangle r is inserted in its correct order according to h. After a new rectangle is inserted in a leaf node N, **AdjustTree** is called to fix the MBR and LHV values in the upper-level nodes.

Algorithm Insert(node Root, rect r):
/* Inserts a new rectangle *r* in the Hilbert R-tree. *h* is the Hilbert value of the rectangle*/
I1. *Find the appropriate leaf node:*
Invoke **ChooseLeaf(r, h)** to select a leaf node L in which to place *r*.
I2. *Insert r in a leaf node L:*
If L has an empty slot, insert *r* in L in the appropriate place according to the Hilbert order and return.
If L is full, invoke **HandleOverflow(L,r)**, which will return new leaf if split was inevitable,
I3. *Propagate changes upward:*
Form a set S that contains L, its cooperating siblings and the new leaf (if any)
Invoke **AdjustTree(S)**.
I4. *Grow tree taller:*
If node split propagation caused the root to split, create a new root whose children are the two resulting nodes.

Algorithm ChooseLeaf(rect r, int h):
/* Returns the leaf node in which to place a new rectangle *r*. */
C1. *Initialize:*
Set *N* to be the root node.
C2. *Leaf check:*
If *N* is a leaf_return *N*.
C3. *Choose subtree:*
If *N* is a non-leaf node, choose the entry (R, ptr, LHV) with the minimum LHV value greater than *h*.
C4. *Descend until a leaf is reached:*
Set *N* to the node pointed by ptr and repeat from C2.

Algorithm AdjustTree(set S):
/* S is a set of nodes that contains the node being updated, its cooperating siblings (if overflow has occurred) and the newly created node NN (if split has occurred). The routine ascends from the leaf level towards the root, adjusting the MBR and LHV of nodes that cover the nodes in S. It propagates splits (if any) */
A1. If root level is reached, stop.
A2. *Propagate node split upward:*
Let N_p be the parent node of N.
If N has been split, let NN be the new node.
Insert NN in N_p in the correct order according to its Hilbert value if there is room. Otherwise, invoke HandleOverflow(N_p, NN).
If N_p is split, let PP be the new node.
A3. *Adjust the MBR's and LHV's in the parent level:*
Let P be the set of parent nodes for the nodes in S.
Adjust the corresponding MBR's and LHV's of the nodes in P appropriately.
A4. *Move up to next level:*
Let S become the set of parent nodes P, with NN = PP, if N_p was split.
repeat from A1.

Deletion

In the Hilbert R-tree there is no need to re-insert orphaned nodes whenever a father node underflows. Instead, keys can be borrowed from the siblings or the underflowing node is merged with its siblings. This is possible because the nodes have a clear ordering (according to Largest Hilbert Value, LHV); in contrast, in R-trees there is no such concept concerning sibling nodes. Notice that deletion operations require s cooperating siblings, while insertion operations require $s-1$ siblings.

Algorithm Delete(r):
D1. *Find the host leaf:*
Perform an exact match search to find the leaf node L that contains r.
D2. *Delete r :*
Remove r from node L.
D3. If L underflows
borrow some entries from s cooperating siblings.
if all the siblings are ready to underflow.
merge $s+1$ to s nodes,
adjust the resulting nodes.
D4. *Adjust MBR and LHV in parent levels.*
form a set S that contains L and its cooperating siblings (if underflow has occurred).
invoke **AdjustTree(S)**.

Overflow Handling

The overflow handling algorithm in the Hilbert R-tree treats the overflowing nodes either by moving some of the entries to one of the $s-1$ cooperating siblings or by splitting s nodes into $s+1$ nodes.

Algorithm HandleOverflow(node N, rect r):
/* return the new node if a split occurred. */
H1. Let ε be a set that contains all the entries from N and its $s-1$ cooperating siblings.
H2. Add r to ε.
H3. If at least one of the $s-1$ cooperating siblings is not full, distribute ε evenly among the s nodes according to Hilbert values.
H4. If all the s cooperating siblings are full, create a new node NN and distribute ε evenly among the $s+1$ nodes according to Hilbert values_return NN.

Key Applications

Hilbert R-tree is an index structure for multidimensional objects which commonly appear in Multimedia databases, Geographical Information Systems (GIS), and medical databases. For example, in multimedia databases, objects like images, voice, video, etc. need to be stored and retrieved. In GIS, maps contain multidimensional points, lines, and polygons, all of which are new data types.
Another example of such non-traditional data types can be found in medical databases which contain 3-dimensional brain scans (e. g., PET and MRI studies) For example, in these databases one of the common queries is "display the PET studies of 40-year old females that show high physiological activity inside the hippocampus." Temporal databases fit easily in the framework since time can be considered as one more dimension. Multidimensional objects appear even in traditional databases, for example, where a record with k attributes corresponds to a point in the k-space.

Future Directions

The performance of multi-dimensional indexes can be further improved through developing heuristics that produce smaller MBR for R-tree nodes. As it was shown in [7], the search performance of the R-trees improves by minimizing the perimeters and areas of the R-tree nodes. Grouping close-by multidimensional objects together in the same node would result in a better index.

Cross References

► Quadtree and Octree
► R*-tree

- ▶ R-Trees – A Dynamic Index Structure for Spatial Searching
- ▶ Space-Filling Curves

References

1. Beckmann, N., Kriegel, H., Schneider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles. In: Proc. of ACM SIGMOD, pp. 322–331. Atlantic City, NJ, May 1990
2. Bially, T.: Space-filling curves. Their generation and their application to bandwidth reduction. IEEE Trans. on Information Theory **15**(6), 658–664 (1969)
3. Griffiths, J.: An algorithm for displaying a class of space-filling curves. Software-Practice and Experience **16**(5), 403–411 (1986)
4. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: Proc. of ACM SIGMOD, pp. 47–57. Boston, MA, June 1984
5. Jagadish, H.: Linear clustering of objects with multiple attributes. In: Proc. of ACM SIGMOD Conf., pp. 332–342. Atlantic City, NJ, May 1990
6. Kamel, I., Faloutsos, C.: Parallel R-Trees. In: Proc. of ACM SIGMOD Conf., pp. 195–204. San Diego, CA, June 1992
7. Kamel, I., Faloutsos, C.: On Packing R-trees. In: Second International ACM Conference on Information and Knowledge Management (CIKM), pp. 490–499. Washington D.C., 1–5 Nov 1993
8. Kamel, I., Faloutsos, C.: Hilbert R-tree: An improved R-tree using fractals. In: Proc. of VLDB Conf., pp. 500–509. Santiago, Chile, September 1994
9. Koudas, N., Faloutsos, C., Kamel, I.: Declustering Spatial Databases on a Multi-Computer Architecture. In: International Conference on Extending Database Technology (EDBT), pp. 592–614. Avignon, France, 25–29 March 1996
10. Roussopoulos, N., Leifker, D.: Direct spatial search on pictorial databases using Packed R-trees. In: Proc. of ACM SIGMOD, pp. 17–31. Austin, TX, May 1985
11. Schroeder, M.: Fractals, Chaos, Power Laws: Minutes From an Infinite Paradise. W.H. Freeman and Company, NY (1991)
12. Sellis, T., Roussopoulos, N., Faloutsos, C.: The R+-Tree: a dynamic index for multi-dimensional objects. In: Proc. 13th International Conference on VLDB, pp. 507–518. England, September 1987

Indexing, Metric-Space

- ▶ Indexing, High Dimensional

Indexing, Mobile Object

- ▶ Mobile Object Indexing

Indexing Moving Objects

- ▶ Indexing the Positions of Continuously Moving Objects
- ▶ Mobile Object Indexing

Indexing Moving Points

- ▶ Indexing Schemes for Multi-dimensional Moving Objects

Indexing, Native Space

- ▶ Movement Patterns in Spatio-temporal Data

Indexing, Native-Space

- ▶ Indexing Spatio-temporal Archives

Indexing of Moving Objects, B^x-Tree

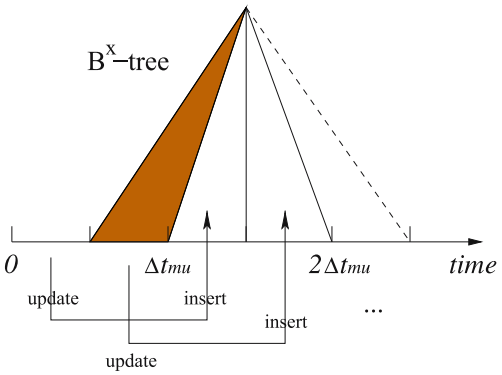
CHRISTIAN S. JENSEN[1], DAN LIN[2], BENG CHIN OOI[2]
[1] Department of Computer Science, Aalborg University, Aalborg, Denmark
[2] Department of Computer Science, National University of Singapore, Singapore, Singapore

Synonyms

B^x-tree; Moving objects; Linearization; Peano curve; Range query algorithm

Definition

The B^x-tree [1] is a query and update efficient B^+-tree-based index structure for moving objects which are represented as linear functions. The B^x-tree uses a linearization technique to exploit the volatility of the data values being indexed i. e., moving-object locations. Specifically, data values are first partitioned according to their update time and then linearized within the partitions according to a space-filling curve, e. g., the Peano or Hilbert curve. The resulting values are combined with their time partition information and then indexed by a single B^+-tree. Figure 1 shows an example of the B^x-tree with the number of index partitions equal to two within one maximum update interval Δt_{mu}. In this example, there are maximum of three partitions existing at the same time. After linearization, object locations inserted at time 0 are indexed in partition 1, object locations updated during time 0 to 0.5 Δt_{mu} are indexed in partition 2 and objects locations updated during time 0.5 Δt_{mu} to time Δt_{mu} are indexed in partition 3 (as indicated by arrows). As time elapses, repeatedly the first range expires (shaded area), and a new range is appended (dashed line). This use of rolling ranges enables the B^x-tree to handle time effectively.



Indexing of Moving Objects, B^x-Tree, Figure 1 An example of the B^x-tree

Historical Background

Traditional indexes for multidimensional databases, such as the R-tree [2] and its variants were, implicitly or explicitly, designed with the main objective of supporting efficient query processing as opposed to enabling efficient updates. This works well in applications where queries are relatively much more frequent than updates. However, applications involving the indexing of moving objects exhibit workloads characterized by heavy loads of updates in addition to frequent queries.

Several new index structures have been proposed for moving-object indexing. One may distinguish between indexing of the past positions versus indexing of the current and near-future positions of spatial objects. The B^x-tree belongs to the latter category.

Past positions of moving objects are typically approximated by polylines composed of line segments. It is possible to index line segments by R-trees, but the trajectory memberships of segments are not taken into account. In contrast to this, the spatio-temporal R-tree [3] attempts to also group segments according to their trajectory memberships, while also taking spatial locations into account. The trajectory-bundle tree [3] aims only for trajectory preservation, leaving other spatial properties aside. Another example of this category is the multi-version 3DR-tree [4], which combines multi-version B-trees and R-trees. Using partial persistence, multi-version B-trees guarantee time slice performance that is independent of the length of the history indexed.

The representations of the current and near-future positions of moving objects are quite different, as are the indexing challenges and solutions. Positions are represented as points (constant functions) or functions of time, typically linear functions. The Lazy Update R-tree [5] aims to reduce update cost by handling updates of objects that do not move outside their leaf-level MBRs specially, and a generalized approach to bottom-up update in R-trees has recently been examined [6].

Tayeb et al. [7] use PMR-quadtrees for indexing the future linear trajectories of one-dimensional moving points as line segments in (x, t)-space. The segments span the time interval that starts at the current time and extends some time into the future, after which time, a new tree must be built. Kollis et al. [8] employ dual transformation techniques which represent the position of an object moving in a d-dimensional space as a point in a $2d$-dimensional space. Their work is largely theoretical in nature. Based on a similar technique, Patel et al. [9] have most recently developed a practical indexing method, termed STRIPES, that supports efficient updates and queries at the cost of higher space requirements. Another representative indexes are the TPR-tree (time-parameterized R-tree) family of indexes (e. g., [10, 11]), which add the time parameters to bounding boxes in the traditional R-tree.

Scientific Fundamentals

Index Structure

The base structure of the B^x-tree is that of the B$^+$-tree. Thus, the internal nodes serve as a directory. Each internal node contains a pointer to its right sibling (the pointer is non-null if one exists). The leaf nodes contain the moving-object locations being indexed and corresponding index time.

To construct the B^x-tree, the key step is to map object locations to single-dimensional values. A space-filling curve is used for this purpose. Such a curve is a continuous path which visits every point in a discrete, multi-dimensional space exactly once and never crosses itself. These curves are effective in preserving proximity, meaning that points close in multidimensional space tend to be close in the one-dimensional space obtained by the curve. Current versions of the B^x-tree use the Peano curve (or Z-curve) and the Hilbert curve. Although other curves may be used, these two are expected to be particularly good according to analytical and empirical studies in [12]. In what follows, the value obtained from the space-filling curve is termed as the *x_value*.

An object location is given by $O = (\vec{x}, \vec{v})$, a position and a velocity, and an update time, or timestamp, t_u, where these values are valid. Note that the use of linear functions reduces the amount of updates to one third in comparison to constant functions. In a leaf-node entry, an object O updated at t_u is represented by a value $B^x value(O, t_u)$:

$$B^x value(O, t_u) = [index\_partition]_2 \oplus [x\_rep]_2 \qquad (1)$$

where *index_partition* is an index partition determined by the update time, *x_rep* is obtained using a space-filling curve, $[x]_2$ denotes the binary value of x, and $\oplus$ denotes concatenation.

If the timestamped object locations are indexed without differentiating them based on their timestamps, the proximity preserving property of the space-filling curve will be lost; and the index will also be ineffective in locating an object based on its *x_value*. To overcome such problems, the index is "partitioned" by placing entries in partitions based on their update time. More specifically, Δt_{mu} denotes the time duration that is the maximum duration in-between two updates of any object location. Then the time axis is partitioned into intervals of duration Δt_{mu}, and each such interval is sub-partitioned into n equal-length sub-intervals, termed *phases*. By mapping the update times in the same phase to the same so-called *label timestamp* and by using the label timestamps as prefixes of the representations of the object locations, index partitions are obtained, and the update times of updates determine the partitions they go to. In particular, an update with timestamp t_u is assigned a label timestamp $t_{lab} = \lceil t_u + \Delta t_{mu}/n \rceil_l$, where operation $\lceil x \rceil_l$ returns the nearest future label timestamp of x. For example, Fig. 1 shows a B^x-tree with $n = 2$. Objects with timestamp $t_u = 0$ obtain label timestamp $t_{lab} = 0.5\,\Delta t_{mu}$; objects with $0 < t_u \leq 0.5\Delta t_{mu}$ obtain label timestamp $t_{lab} = \Delta t_{mu}$; and so on. Next, for an object with label timestamp t_{lab}, its position at t_{lab} is computed according to its position and velocity at t_u. Then the space-filling curve is applied to this (future) position to obtain the second component of Eq. 1.

This mapping has two main advantages. First, it enables the tree to index object positions valid at different times, overcoming the limitation of the B$^+$-tree, which is only able to index a snapshot of all positions at the same time. Second, it reduces the update frequency compared to having to update the positions of all objects at each timestamp when only some of them need to be updated. The two components of the mapping function in Eq. 1 are consequently defined as follows:

$$index\_partition = (t_{lab}/(\Delta t_{mu}/n) - 1)\, mod(n + 1)$$

$$x\_rep = x\_value(\overrightarrow{x} + \overrightarrow{v} \cdot (t_{lab} - t_u))$$

With the transformation, the B^x-tree will contain data belonging to $n+1$ phases, each given by a *label timestamp* and corresponding to a time interval. The value of n needs to be carefully chosen since it affects query performance and storage space. A large n results in smaller enlargements of query windows (covered in the following subsection), but also results in more partitions and therefore a looser relationship among object locations. In addition, a large n yields a higher space overhead due to more internal nodes.

To exemplify, let $n = 2$, $\Delta t_{mu} = 120$, and assume a Peano curve of order 3 (i. e., the space domain is 8×8). Object positions $O_1 = ((7, 2), (-0.1, 0.05))$, $O_2 = ((0, 6), (0.2, -0.3))$, and $O_3 = ((1, 2), (0.1, 0.1))$ are inserted at times 0, 10, and 100, respectively. The B^x*value* for each object is calculated as follows.

Step 1: Calculate label timestamps and index partitions.

$$t^1_{lab} = \lceil (0 + 120/2) \rceil_l = 60,$$
$$index\_partition^1 = 0 = (00)_2$$
$$t^2_{lab} = \lceil (10 + 120/2) \rceil_l = 120,$$
$$index\_partition^2 = 1 = (01)_2$$
$$t^3_{lab} = \lceil (100 + 120/2) \rceil_l = 180,$$
$$index\_partition^3 = 2 = (10)_2$$

Step 2: Calculate positions x_1, x_2, and x_3 at t^1_{lab}, t^2_{lab}, and t^3_{lab}, respectively.

$$x'_1 = (1, 5)$$
$$x'_2 = (2, 3)$$
$$x'_3 = (4, 1)$$

Step 3: Calculate Z-values.

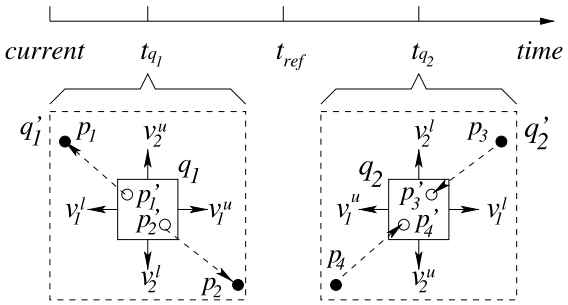
$$[Z\_value(x'_1)]_2 = (010011)_2$$
$$[Z\_value(x'_2)]_2 = (001101)_2$$
$$[Z\_value(x'_3)]_2 = (100001)_2$$

Range Query Algorithm

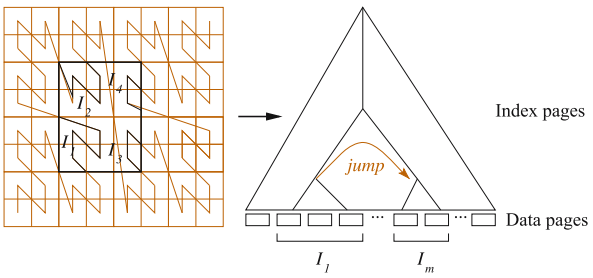
A range query retrieves all objects whose location falls within the rectangular range $q = ([qx^l_1, qx^u_1], [qx^l_2, qx^u_2])$ at time t_q not prior to the current time ("l" denotes lower bound, and "u" denotes upper bound).

A key challenge is to support predictive queries, i. e., queries that concern future times. Traditionally, indexes that use linear functions handle predictive queries by means of bounding box enlargement (e. g., the TPR-tree). Whereas, the B^x-tree uses query-window enlargement. Since the B^x-tree stores an object's location as of some time after its update time, the enlargement involves two cases: a location must either be brought back to an earlier time or forward to a later time. Consider the example in Fig. 2, where t_{ref} denotes the time when the locations of four moving objects are updated to their current value index, and where predictive queries q_1 and q_2 (solid rectangles) have time parameters t_{q1} and t_{q2}, respectively.

The figure shows the stored positions as solid dots and positions of the two first objects at t_{q1} and the positions



Indexing of Moving Objects, B^x-Tree, Figure 2 Query window enlargement



Indexing of Moving Objects, B^x-Tree, Figure 3 Jump in the index

of the two last at t_{q2} as circles. The two positions for each object are connected by an arrow. The relationship between the two positions for each object is $p_i' = p_i + \overrightarrow{v} \cdot (t_q - t_{ref})$. The first two of the four objects, thus, are in the result of the first query, and the last two objects are in the result of the second query. To obtain this result, query rectangle q_1 needs to be enlarged to q_1' (dashed). This is achieved by attaching maximum speeds to the sides of q_1 : v_1^l, v_2^l, v_1^u, andv_2^u. For example, v_1^u is obtained as the largest projection onto the x-axis of a velocity of an object in q_1'. For q_2, the enlargement speeds are computed similarly. For example, v_2^u is obtained by projecting all velocities of objects in q_2' onto the y-axis; v_2^u is then set to the largest speed multiplied by –1.

The enlargement of query $q = ([qx_1^l, qx_1^u], [qx_2^l, qx_2^u])$ is given by query $q' = ([eqx_1^l, eqx_1^u], [eqx_2^l, eqx_2^u])$:

$$eqx_i^l = \begin{cases} qx_i^l + v_i^l \cdot (t_{ref} - t_q) & \text{if } t_q < t_{ref} \\ qx_i^l + v_i^u \cdot (t_q - t_{ref}) & \text{otherwise} \end{cases} \quad (2)$$

$$eqx_i^u = \begin{cases} qx_i^u + v_i^u \cdot (t_{ref} - t_q) & \text{if } t_q < t_{ref} \\ qx_i^u + v_i^l \cdot (t_q - t_{ref}) & \text{otherwise} \end{cases} . \quad (3)$$

The implementation of the computation of enlargement speeds proceeds in two steps. They are first set to the maximum speeds of all objects, thus a preliminary q' is obtained. Then, with the aid of a two-dimensional histogram (e. g., a grid) that captures the maximum and minimum projections of velocities onto the axes of objects in each cell, the final enlargement speed in the area where the query window resides is obtained. Such a histogram can easily be maintained in main memory.

Next, the partitions of the B^x-tree are traversed to find objects falling in the enlarged query window q'. In each partition, the use of a space-filling curve means that a range query in the native, two-dimensional space becomes a set of range queries in the transformed, one-dimensional space (see Fig. 3); hence multiple traversals of the index result. These traversals are optimized by calculating the start and end points of the one-dimensional ranges and traverse the intervals by "jumping" in the index.

k Nearest Neighbor Query Algorithm

Assuming a set of $N>k$ objects and given a query object with position $q = (qx_1, qx_2)$, the k nearest neighbor query (kNN query) retrieves k objects for which no other objects are nearer to the query object at time t_q not prior to the current time.

This query is computed by iteratively performing range queries with an incrementally enlarged search region until k answers are obtained. First, a range R_{q1} centered at q with extension $r_q = D_k/k$ is constructed. D_k is the estimated distance between the query object and its k'th nearest neighbor; D_k can be estimated by the following equation [13]:

$$D_k = \frac{2}{\sqrt{\pi}} \left[1 - \sqrt{1 - \left(\frac{k}{N}\right)^{1/2}} \right] .$$

The range query with range R_{q1} at time t_q is computed, by enlarging it to a range R'_{q1} and proceeding as described in the previous section. If at least k objects are currently covered by R'_{q1} and are enclosed in the inscribed circle of R_{q1} at time t_q, the kNN algorithm returns the k nearest objects and then stops. It is safe to stop because all the objects that can possibly be in the result have been considered. Otherwise, R_{q1} is extended by r_q to obtain R_{q2} and an enlarged window R'_{q2}. This time, the region $R'_{q2} - R'_{q1}$ is searched and the neighbor list is adjusted accordingly. This process is repeated until we obtain an R_{qi} so that there are k objects within its inscribed circle.

Continuous Query Algorithm

The queries considered so far in this section may be considered as one-time queries: they run once and complete when a result has been returned. Intuitively, a continuous

query is a one-time query that is run at each point in time during a time interval. Further, a continuous query takes a *now*-relative time $now + \Delta t_q$ as a parameter instead of the fixed time t_q. The query then maintains the result of the corresponding one-time query at time $now + \Delta t_q$ from when the query is issued at time t_{issue} and until it is deactivated.

Such a query can be supported by a query q_e with time interval $[t_{issue} + \Delta t_q, t_{issue} + \Delta t_q + l]$ ("l" is a time interval) [14]. Query q_e can be computed by the algorithms presented previously, with relatively minor modifications: (i) use the end time of the time interval to perform forward enlargements, and use the start time of the time interval for backward enlargements; (ii) store the answer sets during the time interval. Then, from time t_{issue} to $t_{\text{issue}} + l$, the answer to q_l is maintained during update operations. At $t_{\text{issue}} + l$, a new query with time interval $[t_{issue} + \Delta t_q + l, t_{issue} + \Delta t_q + 2l]$ is computed.

A continuous range query during updates can be maintained by adding or removing the object from the answer set if the inserted or deleted object resides in the query window. Such operations only introduce CPU cost.

The maintenance of continuous *k*NN queries is somewhat more complex. Insertions also only introduce CPU cost: an inserted object is compared with the current answer set. Deletions of objects not in the answer set does not affect the query. However, if a deleted object is in the current answer set, the answer set is no longer valid. In this case, a new query with a time interval of length l at the time of the deletion is issued. If the deletion time is t_{del}, a query with time interval $[t_{del} + \Delta t_q, t_{del} + \Delta t_q + l]$ is triggered at t_{del}, and the answer set is maintained from t_{del} to $t_{del} + l$.

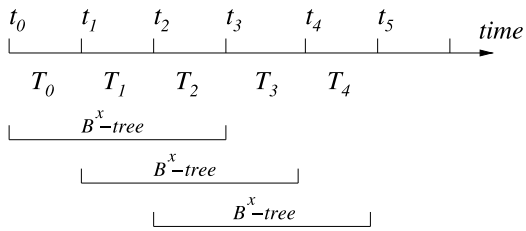
The choice of the "optimal" l value involves a trade-off between the cost of the computation of the query with the time interval and the cost of maintaining its result. On the one hand, a small l needs to be avoided as this entails frequent recomputations of queries, which involve a substantial I/O cost. On the other hand, a large l introduces a substantial cost: Although computing one or a few queries is cost effective in itself, the cost of maintaining the larger answer set must also be taken into account, which may generate additional I/Os on each update. Note that maintenance of continuous range queries incur only CPU cost. Thus, a range query with a relatively large l is computed such that l is bounded by $\Delta t_{mu} - \Delta t_q$ since the answer set obtained at t_{issue} is no longer valid at $t_{\text{issue}} + \Delta t_{mu}$. For the continuous *k*NN queries, l needs to be carefully chosen.

Update, Insertion, and Deletion

Given a new object, its index key is calculated according to Eq. 1, and then insert it into the B^x-tree as in the B$^+$-tree. To delete an object, an assumption is made that the positional information for the object used at its last insertion and the last insertion time are known. Then its index key is calculated and the same deletion algorithm as in the B$^+$-tree is employed. Therefore, the B^x-tree directly inherits the good properties of the B$^+$-tree, and efficient update performance is expected.

However, one should note that update in the B^x-tree does differ with respect to update in the B$^+$-tree. The B^x-tree only updates objects when their moving functions have been changed. This is realized by clustering updates during a certain period to one time point and maintaining several corresponding sub-trees. The total size of the three sub-trees is equal to that of one tree indexing all the objects.

In some applications, there may be some object positions that are updated relatively rarely. For example, most objects may be updated at least each 10 minutes, but a few objects are updated once a day. Instead of letting outliers force a large maximum update interval, a "maximum update interval" within which a high percentage of objects have been updated is used. Object positions that are not updated within this interval are "flushed" to a new partition using their positions at the label timestamp of the new partition. In the example shown in Fig. 4, suppose that some object positions in T_0 are not updated at the time when T_0 expires. At this time, these objects are moved to T_2. Although this introduces additional update cost, the (controllable) amortized cost is expected to be very small since outliers are rare. The forced movement of an object's position to a new partition does not cause any problem with respect to locating the object, since the new partition can be calculated based on the original update time. Likewise, the query efficiency is not affected.



Indexing of Moving Objects, B^x-Tree, Figure 4 B^x-tree evolution

Key Applications

With the advances in positioning technologies, such as GPS, and rapid developments of wireless communication devices, it is now possible to track continuously moving objects, such as vehicles, users of wireless devices and goods. The B^x-tree can be used in a number of emerg-

ing applications involving the monitoring and querying of large quantities of continuous variables, e. g., the positions of moving objects. In the following, some of these applications are discussed.

Location-Based Service

A traveller comes to a city that he is not familiar with. To start with, he sends his location by using his PDA or smart phone (equipped with GPS) to a local server that provides location-based services. Then the service provider can answer queries like "where is the nearest restaurant (or hotel)?" and can also help to dispatch a nearby taxi to the traveller.
A driver can also benefit from the location-based services. For example, he can ask for a nearest gas station or motel when he is driving.

Traffic Control

If the moving objects database stores information about locations of vehicles, it may be able to predict the possible congestion in near future. To avoid the congestion, the system can divert some vehicles to alternate routes in advance.
For air traffic control, the moving objects database system can retrieve all the aircrafts within a certain region and prevent a possible collision.

E-commerce

In these applications, stores send out advertisements or e-coupons to vehicles passing by or within the store region.

Digital Game

Another interesting example is location-based digital game where the positions of the mobile users play a central role. In such games, players need to locate their nearest neighbors to fulfill "tasks" such as "shooting" other close players via their mobile devices.

Battle Field

The moving object database technique is also very important in the military. With the help of the moving object database techniques, helicopters and tanks in the battlefield may be better positioned and mobilized to the maximum advantage.

RFID Application

Recently, applications using radio frequency identification (RFID) has received much interest. RFID enables data to be captured remotely via radio waves and stored on electronic tags embedded in their carriers. A reader (scanner) is then used to retrieve the information. In a hospital application, RFIDs are tagged to all patients, nurses and doctors, so that the system can keep a real-time tracking of their movements. If there is an emergency, nurses and doctors can be sent to the patients more quickly.

Future Directions

Several promising directions for future work exist. One could be the improvement of the range query performance in the B^x-tree since the current range query algorithm uses the strategy of enlarging query windows which may incur some redundant search. Also, the use of the B^x-tree for the processing of new kinds of queries can be considered. Another direction is the use of the B^x-tree for other continuous variables than the positions of mobile service users. Yet another direction is to apply the linearization technique to other index structures.

Cross References

► Indexing, BDual Tree
► Indexing the Positions of Continuously Moving Objects

Recommended Reading

1. Jensen, C.S., Lin, D., Ooi, B.C.: Query and update efficient B+-tree based indexing of moving objects. In: Proc. VLDB, pp. 768–779 (2004)
2. Guttman, A.: R-trees: A Dynamic Index Structure for Spatial Searching. In: Proc. ACM SIGMOD, pp. 47–57 (1984)
3. Pfoser, D., Jensen, C.S., Theodoridis, Y.: Novel approaches in query processing for moving objects. In: Proc. VLDB, pp. 395–406 (2000)
4. Tao, Y., Papadias, D.: MV3R-tree: a spatio-temporal access method for timestamp and interval queries. In: Proc. VLDB, pp. 431–440 (2001)
5. Kwon, D., Lee, S., Lee, S.: Indexing the current positions of moving objects using the lazy update R-tree. In: Proc. MDM, pp. 113–120 (2002)
6. Lee, M.L., Hsu, W., Jensen, C.S., Cui, B., Teo, K.L.: Supporting frequent updates in R-trees: a bottom-up approach. In: Proc. VLDB, pp. 608–619 (2003)
7. Tayeb, J., Ulusoy, O., Wolfson, O.: A quadtree based dynamic attribute indexing method. Computer J. **41**(3):185–200 (1998)
8. Kollios, G., Gunopulos, D., Tsotras, V.J.: On indexing mobile objects. In: Proc. PODS, pp. 261–272 (1999)
9. Patel, J.M., Chen Y., Chakka V.P.: STRIPES: An efficient index for predicted trajectories. In: Proc. ACM SIGMOD, (2004) (to appear)
10. Saltenis, S., Jensen, C.S., Leutenegger, S.T., Lopez, M.A.: Indexing the positions of continuously moving objects. In: Proc. ACM SIGMOD, pp. 331–342 (2000)
11. Tao, Y., Papadias, D., Sun, J.: The TPR*-tree: an optimized spatio-temporal access method for predictive queries. In: Proc. VLDB, pp. 790–801 (2003)

I

12. Moon, B., Jagadish, H.V., Faloutsos, C., Saltz J.H.: Analysis of the clustering properties of the hilbert space-filling curve. IEEE TKDE **13**(1):124–141 (2001)
13. Tao, Y., Zhang, J., Papadias, D., Mamoulis, N.: An efficient cost model for optimization of nearest neighbor search in low and medium dimensional spaces. IEEE TKDE **16**(10):1169–1184 (2004)
14. Benetis, R., Jensen, C.S., Karciauskas, G., Saltenis, S.: Nearest neighbor and reverse nearest neighbor queries for moving objects. In: Proc. IDEAS, pp. 44–53 (2002)

Indexing, Parametric Space

► Indexing Spatio-temporal Archives
► Movement Patterns in Spatio-temporal Data

Indexing, Query and Velocity-Constrained

SUNIL PRABHAKAR, DMITRI KALASHNIKOV, YUNI XIA
Department of Computer Sciences, Purdue University, West Lafayette, IN, USA

Synonyms

Spatio-temporal indexing; Continuous queries

Definition

Moving object environments are characterized by large numbers of moving objects and numerous concurrent continuous queries over these objects. Efficient evaluation of these queries in response to the movement of the objects is critical for supporting acceptable response times. In such environments the traditional approach of building an index on the objects (data) suffers from the need for frequent updates and thereby results in poor performance. In fact, a brute force, no-index strategy yields better performance in many cases. Neither the traditional approach, nor the brute force strategy achieve reasonable query processing times. The efficient and scalable evaluation of multiple continuous queries on moving objects can be achieved by leveraging two complimentary techniques: *Query Indexing* and *Velocity Constrained Indexing* (*VCI*). Query Indexing relies on i) incremental evaluation; ii) reversing the role of queries and data; and iii) exploiting the relative locations of objects and queries. VCI takes advantage of the maximum possible speed of objects in order to delay the expensive operation of updating an index to reflect the movement of objects. In contrast to techniques that require exact knowledge about the movement of the objects, VCI does not rely on such information. While Query Indexing outperforms VCI, it does not efficiently handle the arrival of new queries. Velocity constrained indexing, on the other hand, is unaffected by changes in queries. A combination of Query Indexing and Velocity Constrained Indexing enables the scalable execution of insertion and deletion of queries in addition to processing ongoing queries.

Historical Background

The importance of moving object environments is reflected in the significant body of work addressing issues such as indexing, uncertainty management, broadcasting, and models for spatio-temporal data. Several indexing techniques for moving objects have been proposed. These include indexes over the histories, or trajectories, of the positions of moving objects, or the current and anticipated future positions of the moving objects. Uncertainty in the positions of the objects is dealt with by controlling the update frequency where objects report their positions and velocity vectors when their actual positions deviate from what they have previously reported by some threshold. Tayeb et al. use quad-trees to index the trajectories of one-dimensional moving points. In one approach, moving objects and their velocities are mapped to points which are indexed using a kD-tree. Another indexes the past trajectories of moving objects treated as connected line segments. Yet another considers the management of collections of moving points in the plane by describing the current and expected positions of each point in the future [3]. They address how often to update the locations of the points to balance the costs of updates against imprecision in the point positions. The two techniques presented here appeared in [2].

Scientific Fundamentals

Continuous Query Processing

Location-based environments are characterized by large numbers of moving (and stationary) objects. To support these services it is necessary to execute efficiently several types of queries, including range queries, nearest-neighbor queries, density queries, etc. An important requirement in location-aware environments is the continuous evaluation of queries. Given the large numbers of queries and moving objects in such environments, and the need for a timely response for continuous queries, efficient and scalable query execution is paramount.
This discussion focuses on range queries. Range queries arise naturally in spatial applications such as a query that needs to keep track of, for example, the number of people that have entered a building. Range queries can also be useful as pre-processing tools for reducing the amount of

data that other queries, such as nearest-neighbor or density, need to process.

Model

Moving objects are represented as points, and queries are expressed as rectangular spatial regions. Therefore, given a collection of moving objects and a set of queries, the problem is to identify which objects lie within (i. e., are relevant to) which queries. Objects report their new locations periodically or when they have moved by a significant distance. Updates from different objects arrive continuously and asynchronously. The location of each object is saved in a file. Objects are required to report only their location, not velocity. There is no constraint on the movement of objects except that the maximum possible speed of each object is known and not exceeded (this is required only for Velocity Constrained Indexing).

Ideally, each query should be re-evaluated as soon as an object moves. However, this is impractical and may not even be necessary from the user's point of view. The continuous evaluation of queries takes place in a periodic fashion whereby the set of objects that are relevant to each continuous query are determined at fixed time intervals.

Limitations of Traditional Indexing

A natural choice for efficient evaluation of range queries is to build a spatial index on the objects. To determine which objects intersect each query, the queries are executed using this index. The use of the spatial index should avoid many unnecessary comparisons of queries against objects this approach should outperform the brute force approach. This is in agreement with conventional wisdom on indexing. In order to evaluate the answers correctly, it is necessary to keep the index updated with the latest positions of objects as they move. This represents a significant problem. Notice that for the purpose of evaluating continuous queries, only the current snapshot is relevant and not the historical movement of objects.

In [2], three alternatives for keeping an index updated are evaluated. Each of these gives very poor performance – even worse than the naive brute-force approach. The poor performance of the traditional approach of building an index on the data (i. e. the objects) can be traced to the following two problems: i) whenever *any* object moves, it becomes necessary to re-execute *all* queries; and ii) the cost of keeping the index updated is very high.

Query Indexing

The traditional approach of using an index on object locations to efficiently process queries for moving objects suffers from the need for constant updates to the index and re-evaluation of all queries whenever any object moves. These problems can be overcome by employing two key ideas:

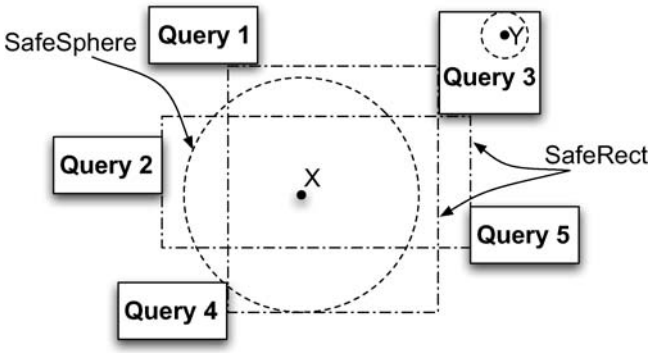
- *reversing* the role of data and queries, and
- *incremental* evaluation of continuous queries.

The notion of *safe regions* that exploit the relative location of objects and queries can further improve performance.

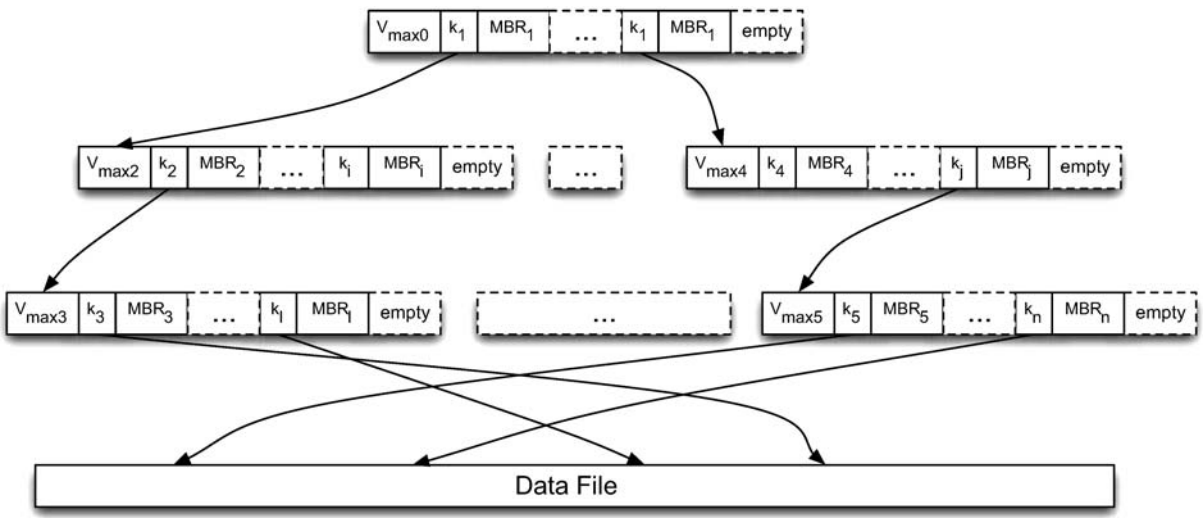
In treating the queries as data, a spatial index such as an R-tree is built on the queries instead of the customary index that is built on the objects (i. e. data). This structure is called a Query-Index or *Q-index*. To evaluate the intersection of objects and queries, each object is treated as a "query" on the Q-index (i. e., the moving objects are treated as queries in the traditional sense). Exchanging queries for data results in a situation where a larger number of queries (one for each object) is executed on a smaller index (the Q-index), as compared to an index on the objects. This is not necessarily advantageous by itself. However, since not all objects change their location at each time step, a large number of "queries" on the Q-index can be avoided by incrementally maintaining the result of the intersection of objects and queries.

Incremental evaluation is achieved as follows: upon creation of the Q-index, all objects are processed on the Q-index to determine the initial result. Following this, the query results are incrementally adjusted by considering the movement of objects. At each evaluation time step, only those objects that have moved since the last time step are processed, and adjust their relevance to queries accordingly. If most objects do not move during each time step, this can greatly reduce the number of times the Q-index is accessed. For objects that move, the Q-index improves the search performance as compared to a comparison against all queries.

Under traditional indexing, at each time step, it would be necessary to update the index on the objects and then evaluate each query on the modified index. This is independent of the movement of objects. With the "Queries as Data" or the Q-index approach, only the objects that have moved since the previous time step are evaluated against the Q-index. Building an index on the queries avoids the high cost of keeping an object index updated; incremental evaluation exploits the smaller numbers of objects that move in a single time step to avoid repeating unnecessary comparisons. Upon the arrival of a new query, it is necessary to compare the query with all the objects in order to initiate the incremental processing. Deletion of queries is easily handled by ignoring those queries. Further improvements in performance can be achieved by taking into account the relative locations of objects and queries or safe regions.



Indexing, Query and Velocity-Constrained, Figure 1 Examples of *Safe Regions*



Indexing, Query and Velocity-Constrained, Figure 2 Example of Velocity Constrained Index (VCI)

***Safe Regions*: Exploiting Query and Object Locations**

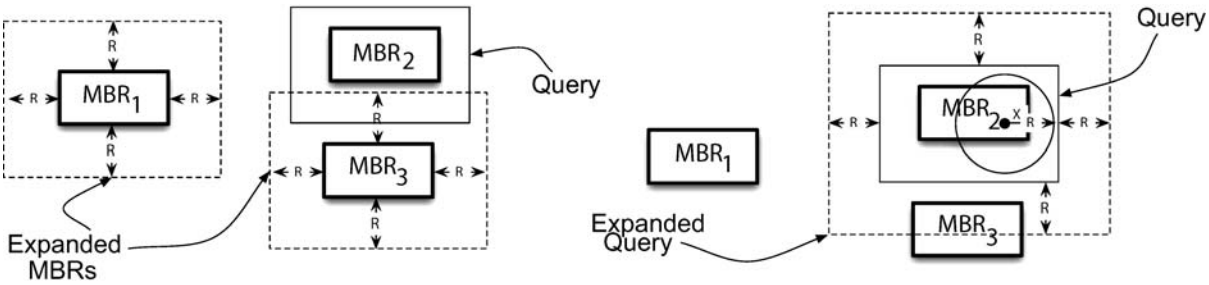
The relevance of an object with respect to a given query can only change if the object crosses a query boundary. Therefore, a region surrounding an object that does not cross any query boundary represents a region within which the object can move without affecting the result of any query. Such a region is termed a *Safe Region*. Two types of safe regions are maximal circles (sphere in general) and maximal rectangles centered at an object's current location. These are termed *SafeSphere* and *SafeRect* respectively. While there is only one maximal sphere for a given object, there can be multiple maximal rectangles.

Figure 1 shows examples of each type of *Safe Region* for two object locations, X and Y. Note that the union of two safe regions is also a safe region. If an object knows a safe region around it, it need not send updates of its movement to the server as long as it remains within the safe region. The safe region optimizations significantly reduce the need to test data points for relevance to queries if they are far from any query boundaries and move slowly. Using safe regions can significantly improve performance.

Velocity-Constrained Indexing

The technique of Velocity-Constrained Indexing (VCI) eliminates the need for continuous updates to an index on moving objects by relying on the notion of a maximum speed for each object. Under this model, the maximum possible speed of each object is known.

A VCI is a regular R-tree based index on moving objects with an additional field in each node: $v_{\max}$. This field stores the maximum allowed speed over all objects covered by that node in the index. The $v_{\max}$ entry for an internal node is simply the maximum of the $v_{\max}$ entries of its children. The $v_{\max}$ entry for a leaf node is the maximum allowed speed among the objects pointed to by the node. Figure 2 shows an example of a VCI. The $v_{\max}$ entry in each node is maintained in a manner similar to the Minimum Bounding



Indexing, Query and Velocity-Constrained, Figure 3 Query Processing with Velocity Constrained Index (VCI)

Rectangle (MBR) of each entry in the node, except that there is only one v_{max} entry per node as compared to an MBR per entry of the node. When a node is split, the v_{max} for each of the new nodes is copied from the original node. Consider a VCI that is constructed at time t_0. At this time it accurately reflects the locations of all objects. At a later time t, the same index does not accurately capture the correct locations of points since they may have moved arbitrarily. Normally the index needs to be updated to be correct. However, the v_{max} fields enable us to use this old index without updating it. We can safely assert that no point will have moved by a distance larger than $R = v_{max}(t - t_0)$. If each MBR is expanded by this amount in all directions, the expanded MBRs will correctly enclose all underlying objects. Therefore, in order to process a query at time t, the VCI created at time t_0 can be used without being updated, by simply comparing the query with expanded version of the MBRs saved in VCI. At the leaf level, each point object is replaced by a square region of side $2R$ for comparison with the query rectangle.[1]

An example of the use of the VCI is shown in Fig. 3a which shows how each of the MBRs in the same index node are expanded and compared with the query. The expanded MBR captures the worst-case possibility that an object that was at the boundary of the MBR at t_0 has moved out of the MBR region by the largest possible distance. Since a single v_{max} value is stored for all entries in the node, each MBR is expanded by the same distance, $R = v_{max}(t - t_0)$. If the expanded MBR intersects with the query, the corresponding child is searched. Thus to process a node all the MBRs stored in the node (except those that intersect without expansion) need to be expanded. Alternatively, a single expansion of the query by R could be performed and compared with the unexpanded MBRs. An MBR will intersect with the expanded query if and only if the same MBR after expansion intersects with the original query. Figure 3b shows the earlier example with query expansion. Expanding the query once per node saves some unnecessary computation.

The set of objects found to be in the range of the query based upon an old VCI is a superset, S', of the exact set of objects that are currently in the query's range. Clearly, there can be no false dismissals in this approach. In order to eliminate the false positives, it is necessary to determine the current positions of all objects in S'. This is achieved through a post-processing step. The current location of the object is retrieved from disk and compared with the query to determine the current matching. Note that it is not always necessary to determine the current location of each object that falls within the expanded query. From the position recorded in the leaf entry for an object, it can move by at most R. Thus its current location may be anywhere within a circle of radius R centered at the position recorded in the leaf. If this circle is entirely contained within the unexpanded query, there is no need to post-process this object for that query. Object X in Fig. 3b is an example of such a point.

To avoid performing an I/O operation for each object that matches each expanded query, it is important to handle the post-processing carefully. We can begin by first preprocessing all the queries on the index to identify the set of objects that need to be retrieved for any query. These objects are then retrieved only once and checked against all queries. This eliminates the need to retrieve the same object more than once. To avoid multiple retrievals of a page, the objects to be retrieved can first be sorted on page number. Alternatively, a clustered index can be built. Clustering may reduce the total number of pages to be retrieved. Clustering the index can improve the performance significantly.

Refresh and Rebuild

The amount of expansion needed during query evaluation depends upon two factors: the maximum speed v_{max} of the node, and the time that has elapsed since the index

[1] Note that it should actually be replaced by a circle, but the rectangle is easier to handle.

was created, $(t - t_0)$. Thus over time the MBRs get larger, encompassing more and more dead space, and may not be minimal. Consequently, as the index gets older its quality gets poorer. Therefore, it is necessary to *rebuild* the index periodically. This essentially resets the creation time, and generates an index reflecting the changed positions of the objects. Rebuilding is an expensive operation and cannot be performed too often. A cheaper alternative to rebuilding the index is to *refresh* it. Refreshing simply updates the locations of objects to the current values and adjusts the MBRs so that they are minimal. Following refresh, the index can be treated as though it has been rebuilt.

Refreshing can be achieved efficiently by performing a depth-first traversal of the index. For each entry in a leaf node the latest location of the object is retrieved (sequential I/O if the index is clustered). The new location is recorded in the leaf page entry. When all the entries in a leaf node are updated, the MBR for the node is computed and recorded it in the parent node. For directory nodes when all MBRs of its children have been adjusted, the overall MBR for the node is computed and recorded it in the parent. This is very efficient with depth-first traversal. Although refresh is more efficient than a rebuild, it suffers from not altering the structure of the index – it retains the earlier structure. If points have moved significantly, they may better fit under other nodes in the index. Thus there is a trade-off between the speed of refresh and the quality of the index. An effective solution is to apply several refreshes followed by a less frequent rebuild. In practice, refreshing works very well thereby avoiding the need for frequent, expensive rebuilds.

Performance

Detailed evaluation of the approaches can be found in [2]. Overall, the experiments show that for Query Indexing, *SafeRect* is the most effective in reducing evaluations. Q-Index is found to give the best performance compared to VCI, traditional indexing, and sequential scans. It is also robust across various scales (numbers of objects, queries), rates of movement, and density of objects versus queries. VCI, on the other hand, is more effective than traditional approaches for small numbers of queries. Moreover, the total cost of VCI approaches that of a sequential scan after some time. Clustering can extend the utility of the VCI index for a longer period of time. Eventually, refresh or rebuilds are necessary. Refreshes are much faster and very effective and thereby reduce the need for frequent rebuilds. VCI is not affected by how often objects move (unlike Q-Index). Thus the costs would not change even if all the objects were moving at each time instant. On the other hand, the VCI approach is very sensitive to the number of queries.

Combined Indexing Scheme

The results show that query indexing and safe region optimizations significantly outperform the traditional indexing approaches and also the VCI approach. These improvements in performance are achieved by eliminating the need to evaluate all objects at each time step through incremental evaluation. Thus they perform well when there is little change in the queries being evaluated. The deletion of queries can be easily handled simply by ignoring the deletion until the query can be removed from the Q-index. The deleted query may be unnecessarily reducing the safe region for some objects, but this does not lead to incorrect processing and the correct safe regions can be recomputed in a lazy manner without a significant impact on the overall costs.

The arrival of new queries, however, is expensive under the query indexing approach as each new query must initially be compared to every object. Therefore a sequential scan of the entire object file is needed at each time step that a new query is received. Furthermore, a new query potentially invalidates the safe regions rendering the optimizations ineffective until the safe regions are recomputed. The VCI approach, on the other hand, is unaffected by the arrival of new queries (only the total number of queries being processed through VCI is important). Therefore to achieve scalability under the insertion and deletion of queries *combined scheme* works best. Under this scheme, both a Q-Index and a Velocity Constrained Index are maintained. Continuous queries are evaluated incrementally using the Q-index and the *SafeRect* optimization. The Velocity Constrained Index is periodically refreshed, and less periodically rebuilt (e. g. when the refresh is ineffective in reducing the cost). New queries are processed using the VCI. At an appropriate time (e. g. when the number of queries being handled by VCI becomes large) all the queries being processed through VCI are transferred to the Query Index in a single step. As long as not too many new queries arrive at a given time, this solution offers scalable performance that is orders of magnitude better than the traditional approaches.

Key Applications

Mobile electronic devices that are able to connect to the Internet have become common. In addition, to being connected, many devices are also able to determine the geographical location of the device through the use of global positioning systems, and wireless and cellular telephone technologies. This combined ability enables new location-based services, including location and mobile commerce (L- and M-commerce). Current location-aware services allow proximity-based queries including map viewing and

navigation, driving directions, searches for hotels and restaurants, and weather and traffic information.

These technologies are the foundation for pervasive location-aware environments and services. Such services have the potential to improve the quality of life by adding location-awareness to virtually all objects of interest such as humans, cars, laptops, eyeglasses, canes, desktops, pets, wild animals, bicycles, and buildings. Applications can range from proximity-based queries on non-mobile objects, locating lost or stolen objects, tracing small children, helping the visually challenged to navigate, locate, and identify objects around them, and to automatically annotating objects online in a video or a camera shot. Another example of the importance of location information is the Enhanced 911 (E911) standard. The standard provides wireless users the same level of emergency 911 support as wireline callers.

Future Directions

Natural extensions of these techniques are to support other types of important continuous queries including nearest-neighbor and density queries. Query indexing can easily be extended for these types of queries too. More generally, there are many instances where it is necessary to efficiently maintain an index over data that is rapidly evolving. A primary example is sensor databases. For the applications too, query indexing can be an effective tool.

An important area for future research is the development of index structures that can handle frequent updates to data. Traditionally, index structures have been built with the assumption that updates are not as frequent as queries. Thus, the optimization decisions (for example, the split criteria for R-trees) are made with query performance in mind. However, for high-update environments, these decisions need to be revisited.

Cross References

► Continuous Queries in Spatio-temporal Databases
► Indexing the Positions of Continuously Moving Objects

Recommended Reading

1. Kollios, G., Gunopulos, D., Tsotras, V.J.: On indexing mobile objects. In: Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), June (1999)
2. Prabhakar, S., Xia, Y., Kalashnikov, D., Aref, W., Hambrusch, S.: Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects. IEEE Trans. Comput. **51**(10), 1124–1140 (2002)
3. Wolfson, Ouri, Xu, Bo, Chamberlain, Sam and Jiang, L.: Moving objects databases: Issues and solutions. In: Proceedings of the SSDBM Conf. pp. 111–122 (1998)

Indexing, R*-Tree

► R*-tree

Indexing Schemes for Multi-dimensional Moving Objects

KHALED ELBASSIONI[1], AMR ELMASRY[2], IBRAHIM KAMEL[3]
[1] Max-Planck-Institute for Computer Science, Saarbrücken, Germany
[2] Department of Computer Engineering and Systems, Alexandria University, Alexandria, Egypt
[3] University of Sharjah, Sharjah, UAE

Synonyms

Indexing moving points; Data-structures; Memory, external; Moving objects; Index lifetime; Disk page; R-tree; TPR-trees; STAR-tree; Dual space-time representation; MB-index

Definition

Indexing schemes are generally data structures that are used to keep track of moving objects whose positions are functions of time. Most of the work in this area is described in the *external memory model*, as the number of mobile objects in the database is assumed to be very large. The objective is to allow for efficient answering of queries that usually involve the current and the predicted future positions of the objects. The main challenges in this case, as compared to the case of static objects, are

(i) *Large number of updates:* in the static case, usually the number of queries is the dominating factor. With moving objects, storing the continuously changing locations directly in the database becomes infeasible, considering the prohibitively large number of updates.
(ii) *Queries about the future are allowed:* this requires that the future positions of the objects be predicted somehow based on the current information and the accuracy of this prediction affects the accuracy of the query results.
(iii) *Frequent optimization overhead:* due to the mobility of the objects, an index optimized to give good query performance on these objects at a certain time might not be good later on. This means that optimization has to be carried out more frequently.

One commonly used solution to (i) is to abstract each object's location as a function of time, which will be stored

in the database and update the database only when the function's parameters change. A typical solution to (ii) is to assume that objects are moving with constant speed, and issue an update only when their speeds change. More generally, sampling can be used to obtain the positions of each object at certain points in time, and interpolation/extrapolation can be used to predict the locations of the object in between the sample points or at some time in the future. To solve (iii), indexing mechanisms often employ the notion of an index *lifetime*, which is the time interval during which the index is designed to give good performance. At the end of this period, the index is rebuilt and the optimization procedure is again applied in order to guarantee good performance for the next period.

Scientific Fundamentals

Model

In the most general setting, consider a database $\mathcal{D}$ of N objects moving in d-dimensional space, and assume that the position of object $o \in \mathcal{D}$ at time t is given by $o.\mathbf{y}(t) = (o.y_1(t), \ldots, o.y_d(t)) \in \mathbb{R}^d$. Most of the known work assumes that each object moves along a *straight line*, justified by the fact that non-linear functions can be approximated accurately enough by connected straight line segments. In such a case, the position of the object is determined by its initial location $o.\mathbf{a} = o.\mathbf{y}(t_{\text{ref}})$, measured at some reference time t_{ref} (usually the time of last update), and its velocity vector $o.\mathbf{v} = (o.v_1, \ldots, o.v_d)$, i. e., $o.\mathbf{y}(t) = o.\mathbf{a} + o.\mathbf{v}(t - t_{\text{ref}})$, for $t \geq t_{\text{ref}}$.

Let B be the disk page size, i. e., the number of units of data that can be processed in a single I/O operation. In the standard external memory model of computation, the efficiency of an algorithm is measured in terms of the number of I/O's required to perform an operation. If N is the number of mobile objects in the database and K is the number of objects reported in a given query, then the minimum number of pages to store the database is $n \stackrel{\text{def}}{=} \lceil \frac{N}{B} \rceil$ and the minimum number of I/O's to report the answer is $k \stackrel{\text{def}}{=} \lceil \frac{K}{B} \rceil$. Thus, the time and space complexity of a given algorithm, under such a model, is measured in terms of parameters n and k.

Typical Queries

The following types of queries have been considered in the literature:

Q_1 *(Window query).* Given an axis-parallel hyper-rectangular query region determined by the two corner points $\mathbf{y}', \mathbf{y}'' \in \mathbb{R}^d$ and two time instants t', t'', report all objects $o \in \mathcal{D}$ which cross the region at some instant between t' and t'', i. e., for which $\mathbf{y}' \leq o.\mathbf{y}(t) \leq \mathbf{y}''$, for some $t' \leq t \leq t''$. When $t' = t''$, the query is sometimes called an *instantaneous* or *timeslice* query.

Q_2 *(Moving query).* Here, the query region is a function of time. For instance, consider a generalization of query Q_1, where the corner points $\mathbf{y}', \mathbf{y}''$ vary linearly with time, i. e., $\mathbf{y}'(t) = \mathbf{y}'_0 + v't$ and $\mathbf{y}''(t) = \mathbf{y}''_0 + v''t$, for some constant vectors $\mathbf{y}'_0, \mathbf{y}''_0, \mathbf{v}', \mathbf{v}'' \in \mathbb{R}^d$. The requirement is to report all objects $o \in \mathcal{D}$ for which $\mathbf{y}'(t) \leq o.\mathbf{y}(t) \leq \mathbf{y}''(t)$, for some time t in a specified interval $[t', t'']$.

Q_3 *(Proximity query).* Given a set of m moving objects $O_1, \ldots, O_m$, with motion functions $\mathbf{y}^i(t)$, for $i = 1, \ldots, m$, and two time instants t', t'', report all objects $o \in \mathcal{D}$ that will become within a distance δ_i from (i. e., will cross *the neighborhood* of) *some* object O_i, $i = 1, \ldots, m$, in the specified time interval, i. e., for which $d(o.\mathbf{y}(t), \mathbf{y}^i(t)) \leq \delta_i$ for *some* $i \in \{1, \ldots, m\}$, and some $t' \leq t \leq t''$, where $d(\cdot,\cdot)$ is the Euclidean distance.

Q_4 *(Approximate nearest neighbor query).* Given a point $p \in \mathbb{R}^d$, and a time instant t', report the δ-approximate nearest neighbor of p in $\mathcal{D}$ at time t', i. e., an object $o \in \mathcal{D}$, such that $d(o.\mathbf{y}(t'), p) \leq (1 + \delta) \min_{o' \in \mathcal{D}} d(o'.\mathbf{y}(t'), p)$. When $\delta = 0$, it is required to report the nearest neighbor exactly. One may also ask for the approximate (or exact) k-nearest neighbors of the given point p at t'.

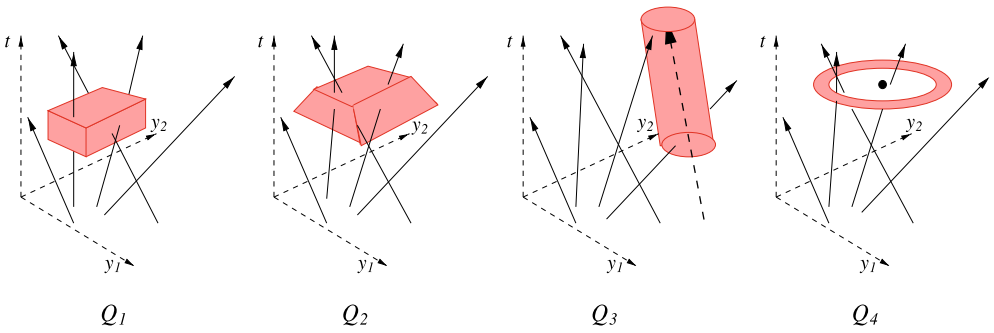
Figure 1 gives examples for these queries. It is assumed that all time instants t', t'' above are at least as large as the current time, i. e., the queries concern the current or future locations of the objects.

Indexing Techniques

Many indices have been recently proposed for indexing moving objects. These techniques can be generally classified into three categories according to the type of moving objects' representation on which they work. Examples of each are given here. Additionally, the reader is referred to [9] and the references therein for a more extensive survey.

1. Explicit Space-Time Representation This is the most straightforward representation. It does not assume linearity of the objects' trajectories. It works by viewing the trajectories of the objects as *static* objects in the $(d+1)$-dimensional space obtained by adding the time as an additional coordinate. Then, any spatial access method, such as quad-trees, R-trees or its variants, can be used to index these trajectories. There is a number of solutions proposed in the literature that belong to this category:

Quad-Tree Based Approach [13]: The 2-dimensional distance-time space (assuming 1-dimensional moving objects) is partitioned into four equal-sized quadrants, each



Indexing Schemes for Multi-dimensional Moving Objects, Figure 1 The four types of queries $Q_1, \ldots, Q_4$

of which is partitioned recursively as in the normal quadtree construction. However, in contrast to the static case, the information about a moving object is stored in every quadrant its trajectory crosses, and this continues recursively until a small number of objects, say at most B, is left in every quadrant. This clearly generalizes to higher dimensions. An obvious drawback is that an object is stored in the index many times, which implies both a large update overhead and a large storage requirement.

R-Tree and Variants A different approach is to approximate each trajectory by its minimum bounding rectangle (MBR), which is then indexed using an *R-tree*, or an R^*-*tree* (see e. g., [2]). However, an MBR assigns a trajectory a much larger area than it really occupies, resulting in what is called *large dead-space*. In addition, the data will be skewed since all trajectories have the same end time value. As a result, these indices usually suffer from excessive overlap between their nodes, eventually leading to poor query performance.

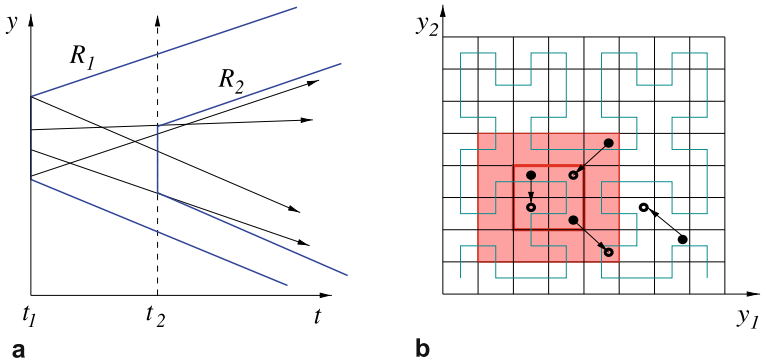
Furthermore, there is a problem in both of the above approaches: at the current time, the trajectories are assumed to extend to infinity. Thus, in order to be able to apply the above techniques, it is necessary to truncate the trajectories at some point in time. One solution is to partition the time dimension into "sessions" of a given length and keep a separate index for each session. However, this again introduces problems for updates and storage.

Obviously, these methods can be used to handle any query type for which spatial access methods are applicable [4,7,12,13,14].

2. Implicit Space-Time (Or Time Parameterized Space) Representation The objects' trajectories are represented as functions of time, which means, for instance, that only the speed and initial location vectors are stored in case of linear motion. The indexing is done with respect to the original spacial coordinates, but the index itself is made dependent on time.

Perhaps the most notable approaches in this category are the following:

Time Parameterized R-Trees (TPR Trees) of [14]: Here, the underlying structure is a usual R-tree, but one in which the minimum bounding rectangles are replaced by the so-called *conservative* bounding rectangles. Each such rectangle grows with time: at time t_{ref}, the rectangle is a minimum bounding rectangle for all moving objects enclosed inside it and continues to be a bounding rectangle for these objects (but not necessarily minimum) at any time $t \geq t_{\text{ref}}$. This is achieved by making the lower and upper bounds of the rectangle move, respectively, with the minimum and maximum speeds of the enclosed objects. Formally, assuming linear motion, a bounding rectangle enclosing a subset of moving objects $\mathcal{O}' \subseteq \mathcal{D}$ is determined by its lower and upper bounds, given respectively by $L(t) = \mathbf{a}_{\min} + \mathbf{v}_{\min}(t - t_{\text{ref}})$ and $U(t) = \mathbf{a}_{\max} + \mathbf{v}_{\max}(t - t_{\text{ref}})$, where the i-th components of $\mathbf{a}_{\min}$ and $\mathbf{v}_{\min}$ are given respectively by $\min\{o.y_i(t_{\text{ref}}) : o \in \mathcal{O}'\}$ and $\min\{o.v_i : o \in \mathcal{O}'\}$, for $i = 1, \ldots, d$, and $\mathbf{a}_{\max}$ and $\mathbf{v}_{\max}$ are defined similarly; see Fig. 3 for an illustration in the 1-dimensional case. As time passes, the parametrized rectangles may grow too much, thus affecting performance. To alleviate this problem, the rectangles are recomputed after a certain period of time, or when an object more enclosed in the rectangle is updated (this is called "*tightening*"); see Fig. 2a. Queries of types Q_1 and Q_2 can be answered by applying the standard R-tree query processing procedure. The difference is that, in this case, at each node of the tree, two time parameterized rectangles have to be checked for intersection. This can be done using a polyhedron-polyhedron intersection. However, a more direct procedure that takes advantage of the fact that these polyhedra are parameterized rectangles is given in [14]. Heuristics for packing the objects into



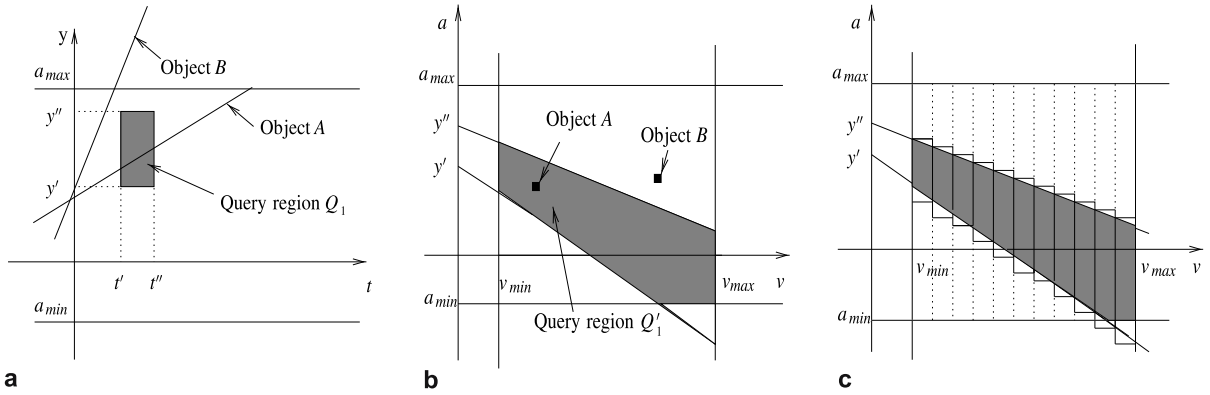
Indexing Schemes for Multi-dimensional Moving Objects, Figure 2 The TPR and B^x-trees: **a** The TPR-tree for the four objects at time t_1 and the tightened tree at t_2. **b** The B^x-tree: the small square represents the query region at the time the query is issued, while the large one represents the expanded query region (at t_{ref}); solid circles represent the locations of objects at t_{ref} and hollow circles represent the locations at the query time. The hollow circle at the bottom-right corner of the large square is a false hit

the nodes of a TPR-tree follow the corresponding ones for the regular R^*-tree [2], with the exception that the measure function (e. g., total volume of bounding rectangles, or overlap among them) is now a function of time whose *integration* over the lifetime of the index is to be minimized. In [11], an index called the *STAR-tree* was suggested to reduce the problem of frequent tightening by using some geometric ideas to compute *tighter* bounding rectangles than the conservative ones. This index is *self-adjusting* in the sense that no explicit tightening is needed (though at the cost of extra space requirement).

B^x-Tree and Its Variants Another approach used in [6] is to map the data into a 1-dimensional space and then use any 1-dimensional indexing structure, such as B^+-trees. In the *B^x-tree* approach, the time coordinate is partitioned into intervals, each of equal size to some upper bound Δ on the duration between two consecutive updates of any object's location. Each such interval is further partitioned into f phases for some integer $f > 0$. This way an object at most belongs to one of the phases at any given time. A B^+-tree is used to index the objects in each phase, where the keys are obtained by mapping the objects into a 1-dimensional space using a *space-filling curve*. More specifically, the d-dimensional data space is partitioned into a uniform grid whose cells are traversed using some space-filling curve; see Fig. 2b for an example using the *Hilbert* curve. This way each object is assigned a 1-dimensional key computed by concatenating its phase number and its space-filling-curve value, with the latter being computed using the location of the object at the start time t_{ref} of the phase. A query issued at time $t \geq t_{ref}$ is *expanded* conservatively using the minimum and maximum speeds of (all or a subset of) the objects to guarantee that all objects satisfying the query at the current time are reported. This may result in some *false positives* which can be removed in a post-processing step. Additionally, [6] suggested to reduce the number of false positives by maintaining a *histogram* of extremum speeds of objects in each grid cell. Depending on which extremum speeds are used for expanding the query range, different strategies can be obtained. For instance, in an iterative approach, one can start using the global extremum velocities to obtain an initial expanded region. Then the extremum velocities among the cells intersecting this region are used to get a refined expansion and so on. This technique has been used in [6] to answer queries of type Q_1, Q_2, and Q_4.

Achieving Logarithmic Query Time It should also be mentioned that if space is not an issue and is allowed to increase non-linearly with n, then logarithmic query time can be achieved; see, e. g., [7] and the *kinetic range trees* of [1].

3. Dual Space-Time Representation An alternative approach, assuming the objects are moving along linear trajectories, is to first map the objects into some *duality space* (see e. g., [5,7]) and then use some method to index the objects in this new space. The idea is to use a transformation under which the query answering problem reduces to a *simplex*, or more generally to a *polyhedral range searching* problem for which efficient indexing techniques are already known. One commonly used transformation is to map each d-dimensional object with velocity vector $\boldsymbol{v}$ and initial location $\boldsymbol{a}$ into a $2d$-dimensional point $(\boldsymbol{v}, \boldsymbol{a})$ (this is called the Hough-X transform in [5]). Given a query, say of type Q_1, the corresponding query region in the dual space can be obtained by eliminating the time variable t from the set of constraints defining Q_1 using, for example, the *Fourier-Motzkin elimination* method (see [3] for more details). For example, for the 1-dimensional version of query Q_1, under the existence of lower and upper bounds $v_{\min} \leq |v| \leq v_{\max}$ and $a_{\min} \leq a \leq a_{\max}$, the query region in the dual space becomes a (possibly) truncated trapezoid [7] (see Fig. 3). Hence, to retrieve all objects that pass between y' and y'' within the time interval $[t', t'']$, one needs to report all the points that lie inside the shaded trapezoid in Fig. 3b.



Indexing Schemes for Multi-dimensional Moving Objects, Figure 3 **a** Query Q_1 in native space. **b** Q_1 in Dual space. **c** Partitioning the dual query region for the MB-index

However, this does not generalize immediately to higher dimensions: if the above transform is used, query Q_1 transforms into a $2d$-dimensional region, bounded by a mix of linear and *quadratic* constraints [3]. This poses a problem since most currently known indexing techniques are designed to deal with polyhedral query regions, i. e., those that are described with linear constraints. One solution is to approximate the dual query region by the minimum bounding rectangle or by some other polyhedron, and add a post-processing step to filter out the objects not satisfying the query. Another solution, suggested in [3], is to use an alternative duality transformation which maps an object with velocity vector $\boldsymbol{v} = (v_1, \ldots, v_d)$ and initial location $\boldsymbol{a} = (a_1, \ldots, a_d)$ to a $2d$-dimensional point $(\boldsymbol{u}, \boldsymbol{w})$, where $u_i \stackrel{\text{def}}{=} 1/v_i$ and $w_i \stackrel{\text{def}}{=} a_i/v_i$, for $i = 1, \ldots, d$ (this is a variant of the so-called Hough-Y transform in [5]). For $v_i > 0$, this gives a one-to-one correspondence between (v_i, a_i) and (u_i, w_i). With such a transformation, query regions of type Q_1 and Q_2 become $2d$-dimensional polyhedra.

Indices With Proved Bounds on the Query Time The simplex range reporting problem has a rich background in the computational geometry literature. In particular, extending a known bound for main-memory range searching, [7] showed that simplex reporting in d-dimensions using only $O(n)$ disk blocks requires $\Omega(n^{1-1/d} + k)$ I/O's. In [8], an almost optimal main memory algorithm for simplex range searching is given using *partition trees*. Given a static set of N points in d-dimensions and an $O(N)$ memory space, this technique answers any simplex range query in $O(N^{1-1/\mathrm{d}+\varepsilon} + K)$ time after an $O(N \log N)$ preprocessing time for any constant $\varepsilon > 0$. This has been adapted to the external memory model in [1,7], showing that a set of N points in two dimensions can be preprocessed in $O(N \log_B n)$ I/O's using $O(n)$ blocks as an *external partition tree*. Using such a structure, simplex range queries can be answered in $O(n^{1/2+\varepsilon} + k)$ I/O's while insertions and deletions can each be supported in amortized expected $O(\log_B^2 n)$ I/O's.

Briefly, a partition tree is a structure that represents a set of points in the Euclidean space; the root of the tree represents the whole set of points and the children of each node represent a *balanced simplicial partitioning* of the points represented by the parent node. A balanced simplicial partitioning of a point set S is a set of pairs $\{(S_1, \Delta_1), \ldots, (S_r, \Delta_r)\}$, where $S_1, \ldots, S_r$ form a partition of S, $|S_i| \leq 2|S_j|$, for all $1 \leq i, j \leq r$, and each Δ_i is a simplex containing all the points in S_i. To guarantee efficient query performance, any straight line must cross at most $O(\sqrt{r})$ of the r simplices. Such partitioning can be constructed in $O(nr)$ I/O's [8]. In order to answer a simplex range query, one starts at the root. If a simplex lies entirely inside the query region, all the points in such a simplex are reported. If the simplex is not intersecting the query region, the simplex is discarded, otherwise the search is continued recursively with the children of the node representing this simplex. Combining the above duality transformation with partition trees, as suggested in [7], the 1-dimensional version of Q_1 (thereby mapped to a 2-dimensional dual space) can be answered in $O(n^{1/2+\varepsilon})$ I/O's using $O(n)$ disk blocks. For the d-dimensional version of Q_1, this technique would map the objects into a $2d$-dimensional dual space, and therefore answer such queries in $O(n^{1-1/2\mathrm{d}+\varepsilon} + k)$ I/O's. In [1], another variant was suggested and yielded asymptotically better bounds for small dimensions by using *multilevel partition trees*. In particular, the asymptotic query time remains $O(n^{1/2+\varepsilon} + k)$ I/O's (assuming constant dimension), but the constant of proportionality increases exponentially with dimension. However, the hidden constant in the query search time of these techniques becomes large if a small ε is chosen.

Spatial Access Methods For more practical techniques, a spatial access method [4] may be used in the dual space. Although all such methods were originally designed to address orthogonal range queries (i. e., queries expressed as multi-dimensional hyper-rectangles), most of them can be easily modified to handle non-orthogonal queries. Note that the use of R*-trees in the dual space is equivalent to using a TPR-tree in the native space without tightening. Another approach, called *STRIPES*, which uses quad-trees in the dual space was suggested in [10].

Approximating the Query Region Another practical approach is based on the idea of approximating the query region in the dual space. This may result in some of the points (called *false hits*) not lying inside the original query region being processed before being excluded. For the 1-dimensional case of Q_1, [7] suggested the use of a rectangle for approximating the query region. In order to reduce the number of false hits, the coordinates of the objects in the original space are measured from c different "observation points", and c-indices are maintained corresponding to these points. It follows that, using this technique, such queries can be answered using $O(\log_B n + f + k)$ I/O's, where f is the number of I/O's resulting from false hits. Assuming uniformly distributed data, f can be bounded by n/c on the average. However, this approach uses $O(cn)$ disk blocks. Under non-uniform distributions of the parameters of the objects, c has to be large in order to reduce the number of false hits. In the *MB-index* of [3], the query region is approximated by partitioning the d-dimensional dual space into disjoint rectangular regions and using a B^+-tree to index the points in each region. In more details, this can be done by fixing one dimension, say the d-th dimension, and splitting the data points in each of the other dimensions $i = 1, \cdots, d-1$, using s $(d-1)$-dimensional hyperplanes perpendicular to the i-th dimension. Given a query region Q in the native space, the index is searched by first transforming the query region into one or more polyhedra in the dual space and then intersecting each such polyhedron with the s^{d-1} hyper-rectangular regions. The intersection defines s^{d-1} *conservative* lower and upper bounds on the d-th dimension, (see Fig. 3c), each of which is then used to search the corresponding B^+-tree. Finally, the resulting candidate answers are scanned to eliminate the false hits. By selecting s and the boundaries of each region appropriately, the number of false hits can be bounded. In particular, the 1-dimensional case of Q_1 can be answered on the average, assuming velocities and initial locations of objects are statistically independent (but not necessarily uniform), in $O(\sqrt{n \log_B n} + k)$ I/O's, with an amortized $O(\log_B n)$ I/O's per update, using $O(n)$ disk blocks. This can be generalized to queries of type Q_2 and Q_3, and also for higher dimensional versions of Q_1.

Key Applications

Transportation Industry

Recent progress in wireless communication technologies and geographical positioning systems (GPS) has made it possible, and relatively cheap, to connect moving vehicles to company databases. Being able to answer queries about these moving vehicles is one of the main motivations to build such databases, and efficient indexing techniques are important. For instance, for a database representing the location of taxi-cabs, a typical query may be: retrieve all free cabs that are currently within one mile of a given location, or retrieve the k nearest cabs (Q_1 or Q_4 queries). For a trucking company database, a typical query may be: retrieve all trucks that will be within one mile of a given truck (which needs assistance) within a certain time period (type Q_3 query).

Air-Traffic Control and Digital Battlefield

With the recently increased issues of safety and security, it is desirable to monitor all commercial aircrafts flying across the country at any given time. A database with this information can be queried, for instance, for all aircrafts that will approach a given region at a given point in time. The region could be either static or moving, or it could be one or more other aircrafts. This gives rise to queries of types Q_1, Q_2 or Q_3. Similar queries may also arise in digital battlefields in military applications.

Mobile Communication Systems

The use of mobile phones, palmtop computers, and smart cards has drastically increased recently. These usually work in a client-server model in which the servers are fixed base stations and the clients are the mobile devices. From the server point of view, it is useful for improving the quality of service to be able to predict either the number or the set of mobile devices that will approach the neighborhood of a given base station in a given time period.

Future Directions

Future work can address extensions of the existing methods to other types of queries; for instance, proximity queries of type Q_3 in higher dimensions. More techniques handling non-linear motion are of great interest. From the theoretical point of view, lower and upper bounds on the indexing problem, for $d \geq 2$, are still open. Another direction is to design practical indices with theoretically proved upper bounds, on the average, as was done preliminarily in [1,3].

Cross References

► Indexing, Hilbert R-tree, Spatial Indexing, Multimedia Indexing

Recommended Reading

1. Agarwal, P.K., Arge, L., Erickson, J.: Indexing moving points. J. Comput. Syst. Sci. **66**(1), 207–243 (2003)
2. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles. SIGMOD '90: Proceedings of the 1990 ACM SIGMOD international conference on Management of data (New York, NY, USA), ACM Press, pp. 322–331 (1990)
3. Elbassioni, K., Elmasry, A., Kamel, I.: An indexing method for answering queries on moving objects. Distrib. Parallel Databases **17**(3), 215–249 (2005)
4. Gaede, V., Günther, O.: Multidimensional access methods-, ACM Comput. Surv. **30**(2), 170–231 (1998)
5. Jagadish, H.V.: On indexing line segments, VLDB '90: Proceedings of the 16th International Conference on Very Large Data Bases San Francisco, CA, USA, pp. 614–625, Morgan Kaufmann Publishers Inc. (1990)
6. Jensen, C.S., Tiešytye, D., Tradišauskas, N.: Robust B+-tree-based indexing of moving objects, MDM '06: Proceedings of the 7th International Conference on Mobile Data Management (MDM'06) Washington, DC, USA, IEEE Computer Society, p. 12 (2006)
7. Kollios, G., Gunopulos, D., Tsotras, V.J.: On indexing mobile objects. PODS '99: Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. pp. 261–272 ACM Press, New York, NY, USA (1999)
8. Matoušek, J.: Efficient partition trees. Discrete Comput. Geom. **8**(3), 315–334 (1992)
9. Mokbel, M.F., Ghanem, T.M., Aref, W.G.: Spatio-temporal access methods. IEEE Data Eng. Bull. **26**(2), 40–49 (2003)
10. Patel, J.M., Chen, Y., Chakka, V.P.: STRIPES: an efficient index for predicted trajectories. SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pp. 635–646. ACM Press, New York, NY, USA (2004)
11. Procopiuc, C.M., Agarwal, P.K., Har-Peled, S.: STAR-tree: An efficient self-adjusting index for moving objects, ALENEX '02: Revised Papers from the 4th International Workshop on Algorithm Engineering and Experiments, pp. 178–193. Springer-Verlag, London, UK (2002)
12. Salzberg, B., Tsotras, V.J.: Comparison of access methods for time-evolving data, ACM Comput. Surv. **31**(2), 158–221 (1999)
13. Tayeb, J., Ulusoy, Ö, Wolfson, O.: A quadtree-based dynamic attribute indexing method. Comput. J. **41**(3), 185–200 (1998)
14. Šaltenis, S., Jensen, C.S., Leutenegger, S.T., Lopez, M.A.: Indexing the positions of continuously moving objects. SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp. 331–342. ACM Press, New York, NY, USA (2000)

Indexing, Spatial

► Oracle Spatial, Geometries

Indexing Spatial Constraint Databases

PETER REVESZ
Department of Computer Science and Engineering, University of Nebraska Lincoln, Lincoln, NE, USA

Synonyms

Access structures for spatial constraint databases

Definition

Indexing structures are data structures used in computer science to store data. Indexing structures are closely associated with indexing methods or indexing algorithms that describe how to use the indexing structures correctly and efficiently to retrieve data, to insert data, or to modify or update data.

Main Text

Sometimes retrieval only means checking whether a given record is in the indexing structure or not. In this case only a "yes" or "no" answer needs to be returned. However, the retrieval of data is usually done on an associative basis, that is, a complex data with multiple components is to be retrieved based on one or more components that are given. For example, from an index structure of customer records, one may want to retrieve all those customer records which contain the street name "Apple Street." There are more sophisticated retrieval problems where instead of exact match between the records and the given information, a comparison or an approximate match is required. For example, retrieve the records of those customers who live within a mile from a specific store location, or retrieve the records of those customer(s) who live closest to the store location.

Sometimes the goal of retrieval is not simply to retrieve the records themselves but to retrieve some aggregate information or statistics of the relevant records. For example, one may be interested in the number of customers who live within a mile from a specific store location. This is called a *Count* query. Another example is finding the maximum number of airplanes that are within the airspace surrounding an airport at any given time. This is called a *Max-Count* query. When precise answers cannot be efficiently computed for Count or Max-Count aggregations, then it may be possible to develop algorithms that efficiently estimate their values.

When static spatial objects need to be indexed, then indexing spatial constraint databases can be done similarly to indexing spatial data or GIS data. However, for indexing

changing maps or moving objects, then the indexing structures are are more specialized. Cai and Revesz [2] extend the R-tree spatial data indexing structure to a parametric R-tree data structure that uses t as a temporal parameter to index moving objects. Revesz and Chen [3,5] describe count and max-count aggregation algorithms for moving objects. These have been extended with better estimation of Count and Max-Count in [1,4].

Cross References

► Constraint Databases, Spatial
► MLPQ Spatial Constraint Database System

Recommended Reading

1. Anderson, S.: Aggregation estimation for 2D moving points. 13th International Symposium on Temporal Representation and Reasoning, IEEE Press, Washington (2006)
2. Cai, M., Revesz, P.: Parametric R-Tree: An index structure for moving objects. 10th International Conference on Management of Data, pp. 57–64. McGraw-Hill, New Dehli (2000)
3. Chen, Y., Revesz, P.: Max-count aggregation estimation for moving points. 11th International Symposium on Temporal Representation and Reasoning,pp. 103–108. IEEE Press, Washington (2004)
4. Revesz, P.: Efficient rectangle indexing algorithms based on point dominance. 12th International Symposium on Temporal Representation and Reasoning, pp. 210–212. IEEE Press, Washington (2005)
5. Revesz, P., Chen, Y.: Efficient aggregation over moving objects. 10th International Symposium on Temporal Representation and Reasoning, pp. 118–127. IEEE Press, Washington (2003)

Indexing Spatio-temporal Archives

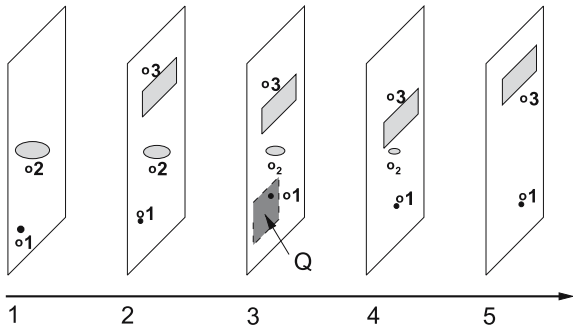
MARIOS HADJIELEFTHERIOU[1], GEORGE KOLLIOS[2], VASSILIS J. TSOTRAS[3], DIMITRIOS GUNOPULOS[3]
[1] AT&T Inc., Florham Park, NJ, USA
[2] Computer Science Department, Boston University, Boston, MA, USA
[3] Computer Science Department, University of California Riverside, Riverside, CA, USA

Synonyms

Spatio-temporal index structures; Moving objects; Lifetime; Evolution; Indexing, native-space; Indexing, parametric space; Index, R-tree; Index, MVR-tree

Definition

Consider a number of objects moving continuously on a 2-dimensional universe over some time interval. Given the complete archive of the spatio-temporal evolution of these objects, we would like to build appropriate index structures for answering range and nearest neighbor queries efficiently. For example: "Find all objects that appeared inside area S during time-interval $[t_1, t_2)$ " and "Find the object nearest to point P at time t".



Indexing Spatio-temporal Archives, Figure 1 A spatio-temporal evolution of moving objects

Consider the spatio-temporal evolution appearing in Fig. 1. The X and Y axes represent the plane while the T axis corresponds to the time dimension. At time 1 objects o_1 (a point) and o_2 (a region) appear. At time 2, object o_3 appears, while o_1 moves to a new position and o_2 shrinks. Object o_1 moves again at times 4 and 5; o_2 continues to shrink and disappears at time 5. Based on its behavior in the spatio-temporal evolution, each object is assigned a record with a *lifetime* interval $[t_i, t_j)$ according to when an object first appeared and disappeared. For example, the lifetime of o_2 is $L_2 = [1, 5)$. A spatio-temporal archive that stores the detailed evolution of the 2-dimensional universe consists of the complete update history of all the objects. A range query is also illustrated in the same figure: "Find all objects that appeared inside area Q at time 3"; only object o_1 satisfies this query. A variety of spatio-temporal and multi-dimensional index structures like the R-tree and its variants [11] can be used for indexing the spatio-temporal archive for the purpose of answering range and nearest neighbor queries. Nevertheless, given the special role that the temporal dimension plays in a spatio-temporal archive, it is possible to design methods and algorithms that can significantly improve upon existing approaches.

Historical Background

Due to the widespread use of sensor devices, mobile devices, video cameras, etc., large quantities of spatio-temporal data are produced everyday. Examples of applications that generate such data include intelligent transportation systems (monitoring cars moving on road networks), satellite and GIS analysis systems (evolution of forest boundaries, fires, weather phenomena), cellular network applications, video surveillance systems, and more.

Given the massive space requirements of spatio-temporal datasets it is crucial to develop methods that enable efficient spatio-temporal query processing. As a result, a number of new index methods for spatio-temporal data have been developed. They can be divided in two major categories: Approaches that optimize queries about the *future positions* of spatio-temporal objects (including continuous queries on the location of moving objects), and those that optimize *historical* queries, i. e., queries about past states of the spatio-temporal evolution. This article concentrates on historical queries.

Güting et al. [10] discussed the fundamental concepts of indexing spatio-temporal objects. Kollios et al. [15] presented methods for indexing the history of spatial objects that move with linear functions of time. The present article is an extension of this work for arbitrary movement functions. Porkaew et al. [22] proposed techniques for indexing moving points that follow trajectories that are combinations of piecewise functions. Two approaches were presented: The Native Space Indexing, where a 3-dimensional R-tree was used to index individual segments of the movement using one MBR per segment (what was referred to in this article as the piecewise approach), and the Parametric Space Indexing, which uses the coefficients of the movement functions of the trajectories to index the objects in a dual space (where only linear functions can be supported by this approach), augmented by the time-interval during which these movement parameters were valid, in order to be able to answer historical queries. A similar idea was used by Cai and Revesz [4]. The present work indexes the trajectories in the native space—being able to support arbitrary movement functions—and is clearly more robust than the piecewise approach in terms of selecting a query-efficiency vs. space tradeoff. Aggarwal and Agrawal [1] concentrated on nearest neighbor queries and presented a method for indexing trajectories with a special convex hull property in a way that guarantees query correctness in a parametric space. This approach is limited to specific classes of object trajectories and is targeted for nearest neighbor queries only. Pfoser et al. [21] introduced the TB-tree which is an indexing method for efficient execution of navigation and historical trajectory queries. TB-trees are optimized for trajectory preservation, targeting queries that need the complete trajectory information to be retrieved in order to be evaluated, in contrast to conventional range and nearest neighbor queries that need to acquire only partial information. Finally, Zhu et al. [27] proposed the Octagon-Prism tree which indexes trajectories by using octagon approximations. The Octagon-Prism tree is mostly related to the TB-tree.

A number of techniques for approximating 1-dimensional sequences have appeared in time-series research. Faloutsos et al. [9] presented a sliding window technique for grouping streaming values into sets of MBRs in order to approximate the time series and reduce the size of the representation. Keogh et al. [13] presented similar algorithms to the ones discussed here for segmenting time-series consisting of piecewise linear segments in an on-line fashion.

The problem of approximating piecewise linear 2-dimensional curves is of great importance in computer graphics and has received a lot of attention from the field of computational geometry as well, for at least the past thirty years. Kolesnikov [14] presents a concise analysis of all the algorithms that have been proposed in the past, including dynamic programming and greedy solutions like the ones discussed here. The pivotal work on this problem was introduced by Douglas and Peucker [6] and Pavlidis and Horovitz [20]. The work in [12] exploited these algorithms in 3-dimensional spaces and also introduced new algorithms for distributing a number of MBRs to a set of 3-dimensional curves.

Methods that can be used to index static spatio-temporal datasets include [5,15,18,23,24,25]. Most of these approaches are based either on the overlapping [3] or on the multi-version approach for transforming a spatial structure into a partially persistent one [2,7,17,26].

Scientific Fundamentals

Preliminaries

Indexing Alternatives for Spatio-temporal Archives The straightforward solution for this problem is to use any multi-dimensional index structure, like the R-tree and its variants [11]. An R-tree would approximate the whole spatio-temporal evolution of an object with one Minimum Bounding Region (MBR) that tightly encloses all the locations occupied by the object during its lifetime. While simple to deploy, simplistic MBR approximations introduce a lot of empty volume, since objects that have long lifetimes correspond to very large MBRs. This, in turn, introduces excessive overlapping between the index nodes of the tree and, therefore, leads to decreased query performance.

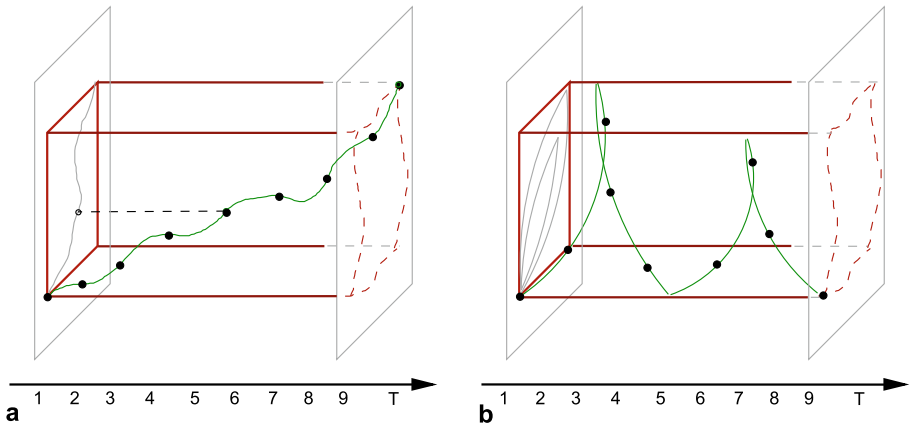
A better approach for indexing a spatio-temporal archive is to use a multi-version index, like the MVR-tree [8,17,26]. This index "logically" stores all the past states of the data evolution and allows updates only to the most recent state. The MVR-tree divides long-lived objects into smaller, better manageable intervals by introducing a number of object copies. A historical query is directed to the exact state acquired by the structure at the time that the query refers to; hence, the cost of answering the query is proportional only to the number of objects that the structure contained at that time. The MVR-tree is an improvement over the

straightforward R-tree approach, especially for short time-interval queries, since the individual parts of the MVR-tree corresponding to different versions of the data that will be accessed for answering a historical query are more compact than an R-tree that indexes the whole evolution. However, there is still space for substantial improvement. Object clustering performed by the multi-version structure is affected by both the length of the lifetime of the objects and their spatial properties. By using rough MBR approximations with the multi-version structure we take advantage of the temporal dimension of the data only and not their spatial characteristics. Given that the spatio-temporal archive is available before building the index, it is possible to further improve index quality by creating finer object approximations that take into account the spatial dimensions as well.

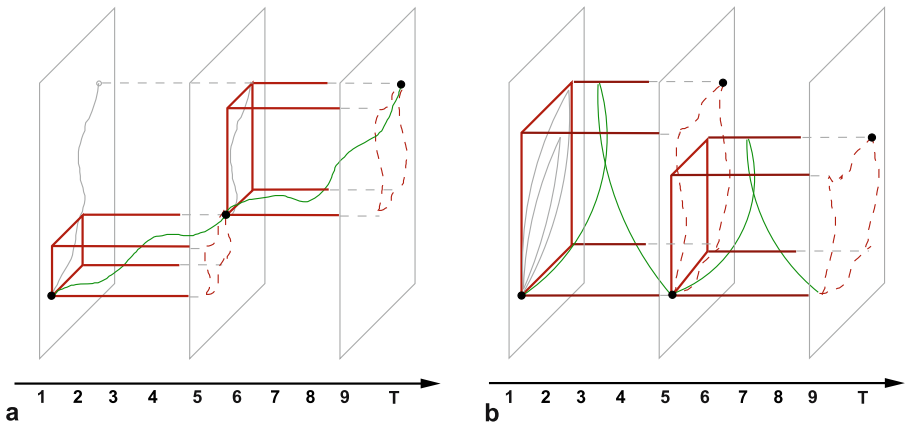
A number of algorithms can be applied on any spatio-temporal archive as a preprocessing step in order to improve index quality and, hence, query performance. These algorithms produce finer object approximations that limit empty volume and index node overlapping. Given N object trajectories the input to the algorithms are N MBRs, one per object trajectory. The output is K MBRs (typically $K = O(N)$) that approximate the original objects more accurately. Some of the N original MBRs may still be among the resulting K (for objects for which no better approximation was needed to be performed), while others will be split into sets of smaller, consecutive in time MBRs. The resulting MBRs will then be indexed with the aim of answering historical queries efficiently. Using these algorithms in combination with normal indexing techniques like the R-tree will not yield any improvements. Motivated by the query cost formula introduced by Pagel et al. [19] which states that the query performance of any MBR based index structure is directly proportional to the total volume, the total surface and the total number of indexed objects, it is argued that the proposed algorithms will benefit mostly the multi-version approach. This is due to the fact that introducing finer object approximations in the R-tree decreases the total volume of tree nodes but, at the same time, increases the total number of indexed objects, ripping off all benefits. On the other hand, in the MVR-tree the number of alive objects per time-instant remains constant independent of the number of MBRs used to approximate the objects, and this fact constitutes the biggest advantage of the multi-version approach.

Spatio-temporal Object Representation Object movements on a 2-dimensional plane can be modeled using various representations. Most applications represent a moving object as a collection of locations sampled every few seconds: The object movement is a set of tuples $\{\langle t_1, p_1\rangle, \ldots, \langle t_n, p_n\rangle\}$. Another common assumption is that objects follow linear or piecewise linear trajectories. An object movement is then represented by a set of tuples $\{\langle [t_1, t_i), f_{x_1}(t), f_{y_1}(t)\rangle, \ldots, \langle [t_j, t_n), f_{x_n}(t), f_{y_n}(t)\rangle\}$, where t_1 is the object insertion time, t_n is the object deletion time, $t_i, \ldots, t_j$ are the intermediate time instants when the movement of the object changes characteristics and $f_{x_i}, f_{y_i} : \mathbb{R} \rightarrow \mathbb{R}$ are the corresponding linear functions. In general, the time-interval between two object updates is arbitrary, meaning that for any two consecutive time-instants t_i, t_j the corresponding time-interval $[t_i, t_j]$ can have an arbitrary length. In the degenerate case, the object can be represented by one movement function per time-instant of its lifetime. If objects follow complex non-linear movements this approach becomes inefficient as it requires a large number of linear segments in order to accurately represent the movement. A better approach is to describe movements by using combinations of more general functions. All these approaches can be easily extended for higher dimensionality.

Regardless of the preferred representation, an object can always be perceived as the collection of the locations it occupied in space for every time-instant of its lifetime. This *exact representation* is clearly the most accurate description of the object's movement that can be attained but, at the same time, it has excessive space requirements. Nevertheless, it is amenable to substantial compression. Given a space constraint or a desired approximation accuracy, one can derive various approximations of a spatio-temporal object. On the one extreme, the object can be approximated with a single MBR—a 3-dimensional box whose height corresponds to the object's lifetime interval and its base to the tightest 2-dimensional MBR that encloses all locations occupied by the object (in the rest, this representation is referred to as the *single MBR approximation*). This simplistic approximation introduces unnecessary empty volume. For example, in Fig. 2a point starts at time 1 from the lower left corner of the plane and follows an irregular movement. It is evident that approximating the point's movement with a single MBR introduces too much empty volume (but the space requirements of this approximation are only the two endpoints of the single MBR). A similar example is shown in Fig. 2b where a point moves in circles. On the other extreme, the exact representation is equivalent to keeping one point per time instant of the object's lifetime (for a total of nine points in Fig. 2a and b). This representation yields the maximum reduction in empty volume, but also the maximum number of required points, increasing space consumption substantially. Note that these figures depict a moving point for simplicity. In general, objects that have extents that vary over time would require one



Indexing Spatio-temporal Archives, Figure 2 Single MBR approximations. **a** An irregular movement, **b** a circular movement



Indexing Spatio-temporal Archives, Figure 3 Multiple MBR approximations. **a** An irregular movement, **b** a circular movement

MBR, instead of one point, per time instant of their lifetime.

Alternatively, spatio-temporal objects can be represented using multiple MBRs per object, which is a compromise in terms of approximation quality and space requirements. For example, in Fig. 3a and b two smaller MBRs are used to represent each trajectory. It is obvious that the additional MBRs increase the representation's space requirements. It is also apparent that there are many different ways to decompose a trajectory into consecutive MBRs, each one yielding different reduction in empty volume. Given a trajectory and a space constraint (imposed as a limit on the total number of MBRs), an interesting problem is to find the set of MBRs that produces the best approximation possible, given some optimality criterion (for instance, the minimization of the total volume of the representation). Notice that, it is not clear by simple inspection which are the best two MBRs for optimally reducing the total volume of the representation of the object depicted in Fig. 3b. Finding the optimal K MBRs for a given object trajectory is a difficult problem. Moreover, assume that we are given a set of trajectories and a space constraint (as a total number of MBRs). Another interesting problem is how to decide which objects need to be approximated more accurately than others (i. e., which objects need to be approximated using a larger number of MBRs) given that there is an upper-bound on the number of MBRs that can be used overall for the whole set.

The Multi-Version R-tree This section presents the MVR-tree, a multi-version indexing structure based on R-trees. Since this structure is an essential component of the proposed techniques, it is presented here in detail. Given a set of K 3-dimensional MBRs, corresponding to a total of $N < K$ object trajectories, we would like to index the MBRs using the MVR-tree. The MBRs are perceived by the tree as a set of K insertions and K deletions on 2-dimensional MBRs, since the temporal dimension has a special meaning for this structure. Essentially, an MBR with projection S on the X–Y plane and a lifetime interval equal to $[t1, t2)$ on the temporal dimension represents a 2-dimensional MBR S that is inserted at time t_1 and marked as logically deleted at time t_2.

Consider the sequence of $2 \cdot K$ updates ordered by time. MBRs are inserted/deleted from the index following this order. Assume that an ephemeral R-tree is used to index

the MBR projections S that appeared at $t = 0$. At the next time-instant that an update occurs (a new MBR appears or an existing MBR disappears), this R-tree is updated by inserting/deleting the corresponding MBR. As time proceeds, the R-tree evolves. The MVR-tree conceptually stores the evolution of the above ephemeral R-tree over time. Instead of storing a separate R-tree per time-instant—which would result in excessive space overhead—the MVR-tree embeds all the states of the ephemeral R-tree evolution into a graph structure that has linear overhead to the number of updates in the evolution.

The MVR-tree is a directed acyclic graph of nodes. Moreover, it has multiple root nodes each of which is responsible for recording a consecutive part of the ephemeral R-tree evolution (each root splits the evolution into disjoint time-intervals). The root nodes can be accessed through a linear array whose entries contain a time-interval and a pointer to the tree that is responsible for that interval. Data records in the leaf nodes of an MVR-tree maintain the temporal evolution of the ephemeral R-tree data objects. Each data record is thus extended to include the lifetime of the object: *insertion-time* and *deletion-time*. Similarly, index records in the directory nodes of an MVR-tree maintain the evolution of the corresponding index records of the ephemeral R-tree and are also augmented with the same fields.

The MVR-tree is created incrementally following the update sequence of the evolution. Consider an update at time t. The MVR-tree is traversed to locate the target leaf node where the update must be applied. This step is carried out by taking into account the lifetime intervals of the index and leaf records encountered during the update. The search visits only the records whose lifetime fields contain t. After locating the target leaf node, an insertion adds the new data record with lifetime $[t, \infty)$, and a deletion updates the deletion-time of the corresponding data record from ∞ to t.

With the exception of root nodes, a node is called *alive* for all time instants that it contains at least $B \cdot P_v$ records that have not been marked as deleted, where $0 < P_v \leq 0.5$ and B is the node capacity. Otherwise, the node is considered *dead* and it cannot accommodate any more updates. This requirement enables clustering the objects that are alive at a given time instant in a small number of nodes, which in turn improves query I/O.

An update leads to a *structural change* if at least one new node is created. *Non-structural* are those updates which are handled within an existing node. An insertion triggers a structural change if the target leaf node already has B records. A deletion triggers a structural change if the target node ends up having less than $B \cdot P_v$ alive records. The former structural change is a *node overflow*, while the latter is a *weak version underflow* [2].

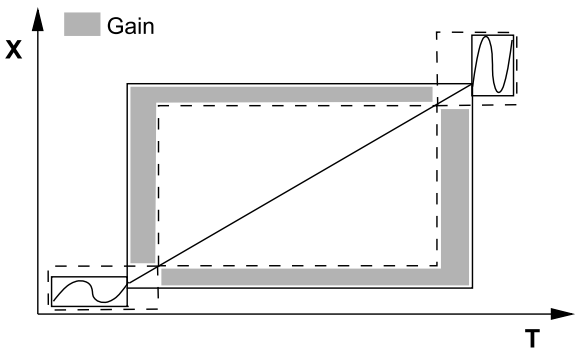
Node overflow and weak version underflow require special handling. Overflows cause a *split* on the target leaf node. Splitting a node A at time t is performed by creating a new node A' and copying all the alive records from A to A'. Now, node A can be considered dead after time t; any queries that refer to times later than t will be directed to node A' instead of A. To avoid having a structural change occurring too soon on node A', the number of alive entries that it contains when it is created should be in the range $[B \cdot P_{svu}, B \cdot P_{svo}]$, where P_{svu} and P_{svo} are predetermined constants. This allows a constant number of non-structural changes on A' before a new structural change occurs. If A' has more than $B \cdot P_{svo}$ alive records a *strong version overflow* occurs; the node will have to be *key-split* into two new nodes. A key-split does not take object lifetimes into account. Instead, it divides the entries according to their spatial characteristics (e. g., by using the R*-tree splitting algorithm which tries to minimize spatial overlap). On the other hand, if A' has less than $B \cdot P_{svu}$ alive records a *strong version underflow* occurs; the node will have to be merged with a sibling node before it can be incorporated into the structure (Kumar et al. [16] discuss various merging policies in detail).

Object Trajectory Approximation Algorithms

This section presents various algorithms for deciding how to approximate the objects contained in a spatio-temporal archive in order to reduce the overall empty volume, improve indexing quality and thus enhance query performance. Henceforth, the optimality of an approximation is considered only in terms of total volume reduction. The problem can be broken up into two parts:

1. Finding optimal object approximations: Given a spatio-temporal object and an upper limit on the number of MBRs that can be used to approximate it, find the set of consecutive MBRs that yield the representation with the minimum volume.
2. Reducing the overall volume of a dataset given a space constraint: Given a set of object trajectories and a space constraint (or, equivalently, a maximum number of MBRs) approximate each object with a number of MBRs such that the overall volume of the final set of MBRs is reduced.

Finding Optimal Object Approximations A simple method for approximating a spatio-temporal object using MBRs is to consider only the time instants when its movement/shape is updated (e. g., the functional boundaries) and partition the object along these points. Although, a few



Indexing Spatio-temporal Archives, Figure 4 Approximating the object using the three dashed MBRs yields more reduction in empty volume than the piecewise approach

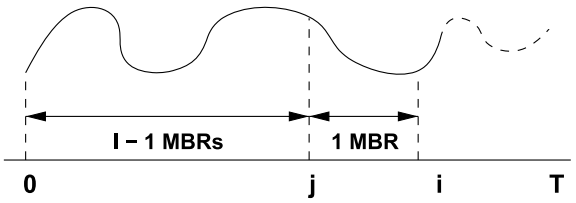
judiciously chosen MBRs are considerably more efficient in decreasing empty volume as shown in Fig. 4 with an example. Another way would be to let the MVR-tree split the object trajectories at the time instants when a leaf version split occurs. The resulting MBRs could be tightened directly after the split by looking at the raw trajectory data. Nevertheless, this technique would not yield the optimal per object approximations, neither can it be used to approximate some objects better than others. In addition, it has the added cost of reading the raw trajectory data multiple times during index creation. Therefore, more sophisticated approaches for finding optimal object approximations are essential.

Given a continuous time domain there are infinite number of points that can be considered as MBR boundaries. Although, practically, for most applications a minimum time granularity can be determined. Under this assumption, an appropriate granularity can be selected according to various object characteristics. The more agile an object is (i. e., the more complex the object movement is), the more detailed the time granularity that needs to be considered. On the other hand, objects that evolve very slowly can be accurately approximated using very few MBRs and thus a very coarse granularity in this case is sufficient. Of course, the more detailed the granularity, the more expensive the algorithms become. A good compromise between computational cost and approximation accuracy per object is application dependent. For ease of exposition in the rest it is assumed that a time granularity has already been decided for a given object. Two algorithms are presented that can be used to find the object approximation that minimizes the empty volume.

*An Optimal Algorithm (*DYNAMICSPLIT*)*

Given a spatio-temporal object O evolving during the interval $[t_0, t_n)$ that consists of n time instants (implied by the chosen time granularity) the goal is to find how to optimally partition the object using k MBRs, such that the final volume of the approximation is minimized. Let $V_l[t_i, t_j]$ be the volume of the representation corresponding to the part of the spatio-temporal object between time instants t_i and t_j after using l optimal MBRs. Then, the following holds:

$$V_l[t_0, t_i] = \min_{0 \le j < i} \{V_{l-1}[t_0, t_j] + V_1[t_j, t_i]\} .$$



Indexing Spatio-temporal Archives, Figure 5 Iteratively finding the best representation for interval $[t_0, t_i)$ using l MBRs

The formula is derived as follows: Suppose that the optimal solutions for partitioning the sub-intervals $[t_0, t_1), [t_0, t_2), \ldots, [t_0, t_{i-1})$ of the object using $l-1$ MBRs are already known. The goal is to find the optimal solution for partitioning interval $[t_0, t_i)$, using l MBRs. The algorithm sets the $(l-1)$-th MBR boundary on all possible positions $j \in [0, 1, \ldots, i-1]$, dividing the object movement into two parts: The optimal representation for interval $[t_0, t_j)$ using $l-1$ MBRs, which is already known by hypothesis, and interval $[t_j, t_i)$ using only one MBR (Fig. 5). The best solution overall is selected, which is the optimal partition of $[t_0, t_i)$ using l MBRs. These steps are applied iteratively until the required amount of MBRs k for the whole lifetime of the object $[t_0, t_n)$ is reached.

Using the above formula a dynamic programming algorithm that computes the optimal positions for k MBRs is obtained. In addition, the total volume of this approximation is computed (value $V_k[t_0, t_n]$).

Theorem 1. *Finding the optimal approximation of an object using k MBRs (i. e., the one that minimizes the total volume) can be achieved in $O(n^2k)$ time, where n is the number of discrete time instants in the object's lifetime.*

A total of nk values of the array $V_l[t_0, t_i]$ $(1 \le l \le k, 0 \le i \le n)$ have to be computed. Each value in the array is the minimum of at most n values, computed using the above formula. The volume $V_1[j, i]$ of the object between positions j and i can be precomputed for every run i and all values of j using $O(n)$ space and $O(n)$ time and thus does not affect the time complexity of the algorithm. □

*An Approximate Algorithm (*GREEDYSPLIT*)*

The DYNAMICSPLIT algorithm is quadratic to the number of discrete time instants in the lifetime of the object.

For objects that live for long time-intervals and for very detailed time granularities the above algorithm is not very efficient. A faster algorithm is based on a greedy strategy. The idea is to start with n consecutive MBRs (the exact representation with the given time granularity) and merge the MBRs in a greedy fashion (Algorithm 1). The running time of the algorithm is $O(n \lg n)$, due to the logarithmic overhead for updating the priority queue at step 2 of the algorithm. This algorithm gives, in general, sub-optimal solutions.

Algorithm 1 GREEDYSPLIT
A spatio-temporal object O as a sequence of n consecutive MBRs, one for each time-instant of the object's lifetime.

InputOutput: A set of k MBRs that represent O's movement.
1: For $0 \leq i < n$ compute the volume of the resulting MBR after merging O_i with O_{i+1}. Store the results in a priority queue.
2: Repeat $n - k$ times: Use the priority queue to merge the pair of consecutive MBRs that give the smallest increase in volume. Update the priority queue with the new (merged) MBR.

Reducing the Overall Volume of a Dataset Given a Space Constraint It is apparent that given a set of objects, in order to represent all of them accurately, some objects may require only few MBRs while others might need a much larger number. Thus, it makes sense to use varying numbers of MBRs per object, according to individual object evolution characteristics. This section discusses methods for approximating a set of N spatio-temporal objects using a given total number of MBRs K such that the total volume of the final object approximations is minimized.[1] In the following, we refer to this procedure as the *MBR assignment process*, implying that, given a set of K "non-materialized" MBRs, each MBR is assigned to a specific object iteratively, such that the volume of the object after being approximated with the extra MBR is minimized (assuming that all objects are initially assigned only one MBR).

*An Optimal Algorithm (*DYNAMICASSIGN*)*
Assuming that the objects are ordered from 1 to N, let $MTV_l[i]$ be the Minimum Total Volume consumed by the first i objects with l optimally assigned MBRs and $V_k[i]$ be the total volume for approximating the i-th object using k MBRs. The following observation holds:

$$MTV_l[i] = \min_{0 \leq k \leq l} \{MTV_{l-k}[i-1] + V_k[i]\}.$$

[1] Where K is implied by some space constraint, e. g., the available disk space.

Intuitively, the formula states that if it is known how to optimally assign up to $l - 1$ MBRs to $i - 1$ objects, it can be decided how to assign up to l MBRs to i objects by considering all possible combinations of assigning one extra MBR between the $i - 1$ objects and the new object. The idea is similar to the one explained for the DYNAMICSPLIT algorithm. A dynamic programming algorithm can be implemented with running time complexity $O(NK^2)$. To compute the optimal solution the algorithm needs to know the optimal approximations per object, which can be computed using the DYNAMICSPLIT algorithm. Hence, the following theorem holds:

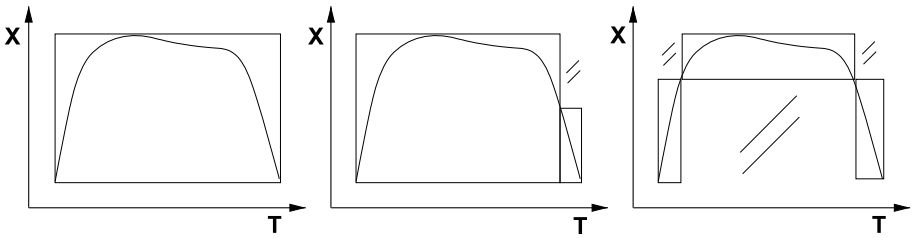
Theorem 2. *Optimally assigning K MBRs among N objects takes $O(NK^2)$ time.*
A total of NK values for array $MTV_l[i]$ ($0 \leq l \leq K$, $1 \leq i \leq N$) need to be computed, where each value is the minimum of at most $K + 1$ values for $0 \leq k \leq K$, in each iteration.□

*A Greedy Algorithm (*GREEDYASSIGN*)*
The DYNAMICASSIGN algorithm is quadratic to the number of MBRs, which makes it impractical for large K. For a faster, approximate solution a greedy strategy can be applied: Given the MBR assignments so far, find the object that if approximated using one extra MBR will yield the maximum possible global volume reduction. Then, assign the MBR to that object and continue iteratively until all MBRs have been assigned. The algorithm is shown in Algorithm 2. The complexity of step 2 of the algorithm is $O(K \lg N)$ (the cost of inserting an object K times in the priority queue) thus the complexity of the algorithm itself is $O(K \lg N)$ (since it is expected that $N < K$).

Algorithm 2 GREEDYASSIGN
A set of N spatio-temporal objects and a number K.

Proof. **InputOutput:** A near optimal minimum volume required to approximate all objects with K MBRs.
1: Assume that each object is represented initially using a single MBR. Find what the volume reduction $VR_{i,2}$ per object i would be when using 2 MBRs to approximate it. Store in a max priority queue according to $VR_{i,2}$.
2: For K iterations: Remove the top element i of the queue, with volume reduction $VR_{i,k}$. Assign the extra k-th MBR to i. Calculate the reduction$VR_{i,k+1}$ if one more MBR is used for i and reinsertit in the queue.

*An Improved Greedy Algorithm (*LAGREEDYASSIGN*)*
The result of the GREEDYASSIGN algorithm will not be optimal in the general case. One reason is the following: Consider an object that yields a small empty volume reduction when approximated using only two MBRs, while most of its empty volume is removed when three MBRs are used



Indexing Spatio-temporal Archives, Figure 6 Two MBRs yield almost no reduction in empty space, while three MBRs reduce it substantially

(an 1-dimensional example of such an object is shown in Fig. 6). Using the GREEDYASSIGN algorithm it is probable that this object will not be given the chance to be assigned any MBRs at all, because its first MBR allocation produces poor results and other objects will be selected instead. However, if the algorithm is allowed to consider more than one assigned MBRs per object per step, the probability of assigning more MBRs to this object increases. This observation gives the intuition on how the greedy strategy can be improved. During each iteration, instead of choosing the object that yields the largest volume reduction by assigning only one more MBR, the algorithm can *look ahead* and find the object that results in even bigger reduction if two, three, or more MBRs were assigned all at once.

For example, the look-ahead-2 algorithm works as follows (Algorithm 3): First, all MBRs are assigned using the GREEDYASSIGN algorithm as before. Then, one new priority queue PQ_1 is created which sorts the objects by the volume reduction offered by their last assigned MBR (if an object has been assigned k MBRs, the object is sorted according to the volume reduction yielded by the k-th MBR). The top of the queue is the *minimum reduction*. A second priority queue PQ_2 is also needed which sorts each object by the volume that would be reduced if it was assigned two more MBRs (if an object has been assigned k MBRs, the object is sorted according to the volume reduction produced by $k + 2$ MBRs). The top of the queue is the *maximum reduction*. If the volume reduction of the top element of PQ_2 is bigger than the sum of the reduction of the two top elements of PQ_1 combined, the splits are reassigned accordingly and the queues are updated. The same procedure continues until there are no more redistributions of MBRs. In essence, the algorithm tries to find two objects for which the combined reduction from their last assigned MBRs is less than the reduction obtained if a different, third object, is assigned two extra MBRs. The algorithm has the same worst case complexity as the greedy approach. Experimental results show that it achieves much better results for the small time penalty it entails.

Algorithm 3 LAGREEDYASSIGN

A set of spatio-temporal objects with cardinality N and a number K.

Proof. **InputOutput:** A set of spatio-temporal objects with cardinality N and a number K.

1: Allocate MBRs by calling the GREEDYASSIGN algorithm. PQ_1 is a min priority queue that sorts objects according to the reduction given by their last assigned MBR. PQ_2 is a max priority queue that sorts objects according to the reduction given if two extra MBRs are used per object.
2: Remove the top two elements from PQ_1, let O_1, O_2. Remove the top element from PQ_2, let O_3. Ensure that $O_1 \neq O_2 \neq O_3$, otherwise remove more objects from PQ_1. If the volume reduction for O_3 is larger than the combined reduction for O_1 and O_2, redistribute the MBRs and update the priority queues.
3: Repeat last step until there are no more redistributions of MBRs.

Key Applications

All applications that generate spatio-temporal data would benefit significantly from using advanced historical index structures. A few examples are intelligent transportation systems (monitoring cars moving on road networks), satellite and GIS analysis systems (evolution of forest boundaries, fires, weather phenomena), cellular network applications, and video surveillance systems.

Future Directions

Future work could concentrate on investigating the online version of the problem. Given a stream of object updates, we would like to maintain a historical spatio-temporal structure efficiently given that the data is made available incrementally and a pre-processing step is not feasible. In particular, it would be interesting to explore how dynamically evolving object update distributions can be detected in order to develop appropriate buffering policies that can be used for pre-processing the data within the buffer, before being inserted into the historical index structure.

Cross References

- ▶ Indexing of Moving Objects, B^x-Tree
- ▶ Indexing the Positions of Continuously Moving Objects
- ▶ Indexing Schemes for Multi-dimensional Moving Objects

- ▶ Mobile Object Indexing
- ▶ Mobile Objects Databases
- ▶ Movement Patterns in Spatio-temporal Data
- ▶ Patterns in Spatio-temporal Data
- ▶ R*-tree
- ▶ R-Trees – A Dynamic Index Structure for Spatial Searching

Recommended Reading

1. Aggarwal, C.C., Agrawal, D.: On nearest neighbor indexing of nonlinear trajectories. In: PODS, pp. 252–259 (2003)
2. Becker, B., Gschwind, S., Ohler, T., Seeger, B., Widmayer, P.: An asymptotically optimal multiversion B-Tree. VLDB J. **5**(4), 264–275 (1996)
3. Burton, F., Kollias, J., Kollias, V., Matsakis, D.: Implementation of overlapping B-trees for time and space efficient representation of collection of similar files. Comput. J., **33**(3), 279–280 (1990)
4. Cai, M., Revesz, P.: Parametric R-tree: An index structure for moving objects. In: COMAD, pp. 57–64 (2000)
5. Chakka, V.P., Everspaugh, A., Patel, J.M.: Indexing large trajectory data sets with SETI. In: CIDR, pp. 164–175 (2003)
6. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitised line or its caricature. Can. Cartogr. **10**(2), 112–122 (1973)
7. Driscoll, J., Sarnak, N., Sleator, D., Tarjan, R.E.: Making data structures persistent. In: STOC (1986)
8. Driscoll, J.R., Sarnak, N., Sleator, D.D., Tarjan, R.E.: Making data structures persistent. J. Comput. Syst. Sci. **38**(1), 86–124 (1989)
9. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: SIGMOD, pp. 419–429 (1994)
10. Güting, R.H., Bhlen, M.H., Erwig, M., Jensen, C.S., Lorentzos, N.A., Schneider, M., Vazirgiannis, M.: A foundation for representing and querying moving objects. TODS **25**(1), 1–42 (2000)
11. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: SIGMOD, pp. 47–57 (1984)
12. Hadjieleftheriou, M., Kollios, G., Tsotras, V.J., Gunopulos, D.: Indexing spatiotemporal archives. VLDB J. **15**(2), 143–164 (2006)
13. Keogh, E.J., Chu, S., Hart, D., Pazzani, M.J.: An online algorithm for segmenting time series. In: ICDM, pp. 289–296 (2001)
14. Kolesnikov, A.: Efficient algorithms for vectorization and polygonal approximation. PhD thesis, University of Joensuu, Finland, (2003)
15. Kollios, G., Tsotras, V.J., Gunopulos, D., Delis, A., Hadjieleftheriou, M.: Indexing animated objects using spatiotemporal access methods. TKDE **13**(5), 758–777 (2001)
16. Kumar, A., Tsotras, V.J., Faloutsos, C.: Designing access methods for bitemporal databases. TKDE **10**(1), 1–20 (1998)
17. Lomet, D., Salzberg, B.: Access methods for multiversion data. In: SIGMOD, pp. 315–324 (1989)
18. Nascimento M., Silva, J.: Towards historical R-trees. In: SAC, pp. 235–240 (1998)
19. Pagel, B.-U., Six, H.-W., Toben, H., Widmayer, P.: Towards an analysis of range query performance in spatial data structures. In: PODS, pp. 214–221 (1993)
20. Pavlidis, T., Horovitz, S.L.: Segmentation of plane curves. IEEE Trans. Comput. **23**(8), 860–870 (1974)
21. Pfoser, D., Jensen, C.S., Theodoridis, Y.: Novel approaches in query processing for moving object trajectories. In: VLDB, pp. 395–406 (2000)
22. Porkaew, K., Lazaridis, I., Mehrotra, S.: Querying mobile objects in spatio-temporal databases. In: SSTD, pp. 59–78 (2001)
23. Tao, Y., Papadias, D.: MV3R-Tree: A spatio-temporal access method for timestamp and interval queries. In: VLDB, pp. 431–440 (2001)
24. Theodoridis, Y., Sellis, T., Papadopoulos, A., Manolopoulos, Y.: Specifications for efficient indexing in spatiotemporal databases. In: SSDBM, pp. 123–132 (1998)
25. Tzouramanis, T., Vassilakopoulos, M., Manolopoulos, Y.: Overlapping linear quadtrees and spatio-temporal query processing. Comput. J. **43**(3), 325–343 (2000)
26. Varman, P.J., Verma, R.M.: An efficient multiversion access structure. TKDE **9**(3), 391–409 (1997)
27. Zhu, H., Su, J., Ibarra, O.H.: Trajectory queries and octagons in moving object databases. In: CIKM, pp. 413–421 (2002)

Indexing the Positions of Continuously Moving Objects

SIMONAS ŠALTENIS
Department of Computer Science, Aalborg University, Aalborg, Denmark

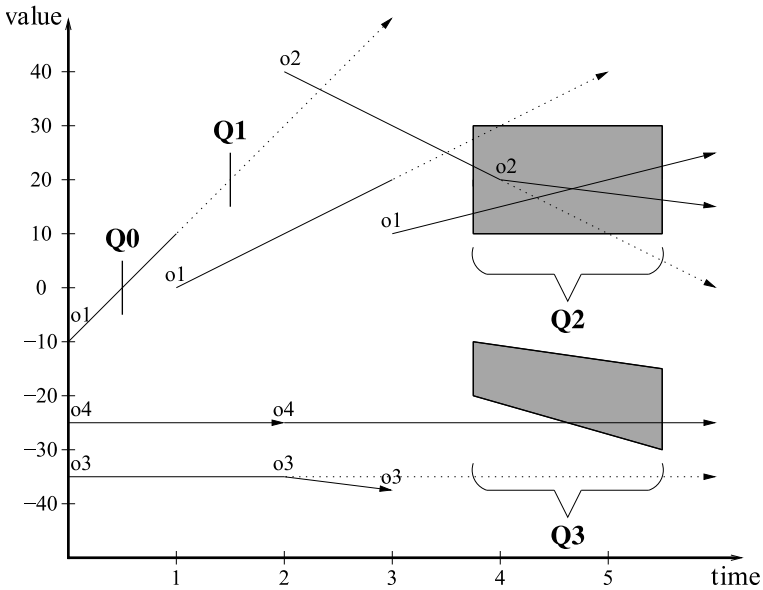
Synonyms

Indexing moving objects; Spatio-temporal indexing

Definition

Wireless communications and positioning technologies enable tracking of the changing positions of objects capable of continuous movement. Continuous movement poses new challenges to database technology. In conventional databases, data is assumed to remain constant unless it is explicitly modified. Capturing continuous movement under this assumption would entail either performing very frequent updates or recording outdated, inaccurate data, neither of which are attractive alternatives. Instead, rather than storing simple positions, functions of time that express the objects' changing positions are stored [21]. More specifically, linear functions are assumed. This entry describes indexing of the current and anticipated future positions of such objects with the focus on the TPR-tree [16].

Modeling the positions of moving objects as functions of time enables querying not only the current positions of objects but also tentative future predictions of these positions. The index should support queries that retrieve all points with positions within specified regions. It is possible to distinguish between three kinds of queries based on the regions they specify: *timeslice queries*, *window queries*, and *moving queries*. Figure 1 shows a set of trajectories



Indexing the Positions of Continuously Moving Objects, Figure 1 Query examples for one-dimensional data [16]

of one-dimensional moving objects and examples of the three types of queries. Here, queries *Q0* and *Q1* are time-slice queries, *Q2* is a window query, and *Q3* is a moving query.

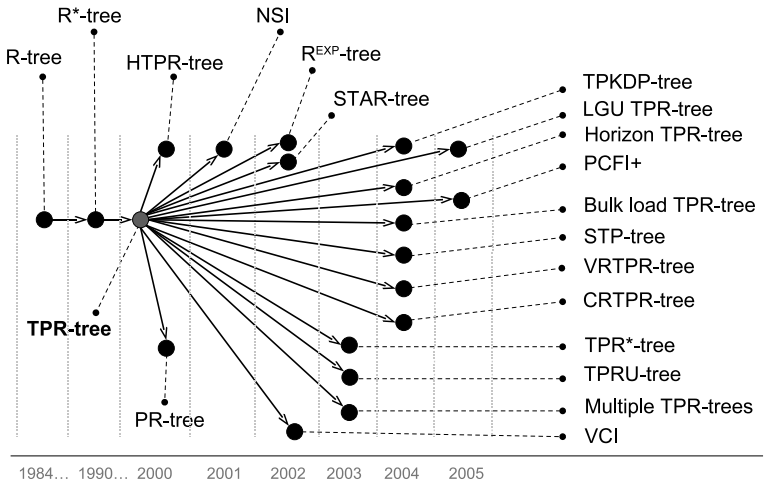
Similarly, with an additional input of a time point or a time interval, other types of spatial queries such as nearest neighbor queries or reverse nearest neighbor queries should be supported [4]. The index should also support insertions and deletions. An update is handled as a deletion followed by an insertion.

Historical Background

The concept of moving object databases and modeling of continuous movement as linear functions is rather recent. It was first introduced by Wolfson et al. in 1998 [21]. The first proposal for indexing such data was introduced by Tayeb et al. [19], who proposed using PMR-quadtrees [12] for indexing the future linear trajectories of one-dimensional moving point objects as line segments in (x, t)-space. Kollios et al. [8] also focuses mainly on one-dimensional data and employs the so-called *duality* data transformation where a line $x = x(t_{\text{ref}}) + v(t - t_{\text{ref}})$ is transformed to the point $((x(t_{\text{ref}}), v))$, enabling the use of regular spatial indices.

Later research focuses on two-dimensional and, in some cases, three-dimensional data. In general, the proposed indexing methods can be classified according to the space that they index, i. e., what view is taken on the indexed data. Assume the objects move in a d-dimensional space ($d = 1, 2, 3$). The first approach is to index future trajectories as lines in a $(d + 1)$-dimensional space. This approach is taken in the above-mentioned method by Tayeb et al. [19], but the method is difficult to extend to higher dimensions and the index has to be periodically rebuilt. The second approach is to map the trajectories to points in a higher-dimensional space which are then indexed. Queries must subsequently also be transformed to counter the data transformation. The above-mentioned duality transformation maps the d-dimensional linearly moving points into $2d$-dimensional static points [8]. In STRIPES [13], PR quadtrees [18] are used to index these $2d$-dimensional points. Yiu et al. instead proposed to use space filling curves to further transform $2d$-dimensional points into one-dimensional points that are then indexed by the B^+-tree. Finally, Agarwal et al. [1] combined the duality transformation with kinetic data structures [2]. The main idea of kinetic data structures is to schedule future events that update a data structure so that necessary invariants hold.

The third approach, sometimes referred to as indexing in the *primal* space, is to index data in its native, d-dimensional space, which is possible by parameterizing the index structure using velocity vectors and thus enabling the index to be "viewed" as of any future time. This absence of transformations yields quite intuitive indexing techniques. The Time Parameterized R-tree (TPR-tree) [16] is the main example of this approach. First proposed by Šaltenis et al. in 2000, the TPR-tree gave rise to a number of other access methods (see Fig. 2). For example, the R^{EXP}-tree extends the TPR-tree to index data with expiration times so that objects that do not update their positions are automatically removed from the index. The TPR*-tree [20] adds a number of heuristics to improve the query performance of the TPR-tree and to optimize it for a slightly different workload of queries than the one considered by the authors of the TPR-tree. The STAR-tree [15] modifies the TPR-tree



Indexing the Positions of Continuously Moving Objects, Figure 2 TPR-tree origins and the follow-up research

by introducing more complex time-parameterized bounding rectangles and making the index self-adjustable. Finally, the Velocity Constrained Indexing (VCI) [14] uses the regular R-tree [5] with an additional field of v_{max} added to each node. The v_{max} is used to expand bounding rectangles of the R-tree when future queries are asked.

The fourth approach is to index the objects' positions at some specific label timestamps and to extend a query according to the maximum speed of objects. B^x-tree [6,7] uses a combination of space-filling curves and B^+-trees to index the static positions of objects at label timestamps.

Scientific Fundamentals

Data and Queries

For an object moving in a d-dimensional space, the object's position at some time t is given by $\bar{x}(t) = (x_1(t), x_2(t), \ldots, x_d(t))$, where it is assumed that the times t are not before the current time. This position is modeled as a linear function of time which is specified by two parameters. The first is a position for the object at some specified time t_{ref}, $\bar{x}(t_{ref})$, which is called the reference position. The second parameter is a velocity vector for the object, $\bar{v} = (v_1, v_2, \ldots, v_d)$. Thus, $\bar{x}(t) = \bar{x}(t_{ref}) + \bar{v}(t - t_{ref})$.

Then, as shown in Fig. 1, *Timeslice query*, $Q = (R, t)$, retrieves points that will be inside the d-dimensional hyper-rectangle R at time t. *Window query*, $Q = (R, t^{\vdash}, t^{\dashv})$, retrieves points that will be inside the hyper-rectangle R sometime during time-interval $[t^{\vdash}, t^{\dashv}]$. *Moving query*, $Q = (R_1, R_2, t^{\vdash}, t^{\dashv})$, retrieves points with trajectories in $(\bar{x}, t)$-space crossing the $(d + 1)$-dimensional trapezoid obtained by connecting R_1 at time $t^{\vdash}$ to R_2 at time $t^{\dashv}$.

The Structure of the TPR-Tree

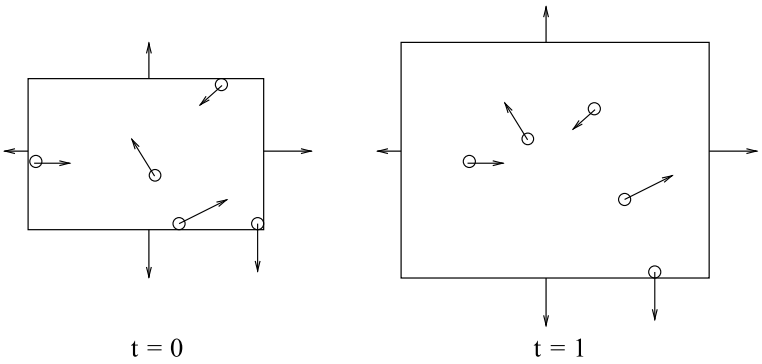
The TPR-tree indexes continuously moving points in one, two, or three dimensions. It employs the basic structure of the R-tree [5], which stores data in the leaves of a balanced index tree and each non-leaf index entry contains a minimum bounding rectangle (MBR) of all the data in the subtree pointed to by the entry. In contrast to the R-tree, both the indexed points and the bounding rectangles are augmented with velocity vectors in the TPR-tree. This way, bounding rectangles are time parameterized—they can be computed for different time points. Velocities are associated with the edges of bounding rectangles so that the enclosed moving objects, be they points or other rectangles, remain inside the bounding rectangles at all times in the future. More specifically, if a number of points p_i are bounded at time t, the spatial and velocity extents of a bounding rectangle along the x axis are computed as follows:

$$x^{\vdash}(t) = \min_i\{p_i.x(t)\}; \quad x^{\dashv}(t) = \max_i\{p_i.x(t)\};$$
$$v_x^{\vdash} = \min_i\{p_i.v_x\}; \quad v_x^{\dashv} = \max_i\{p_i.v_x\}.$$

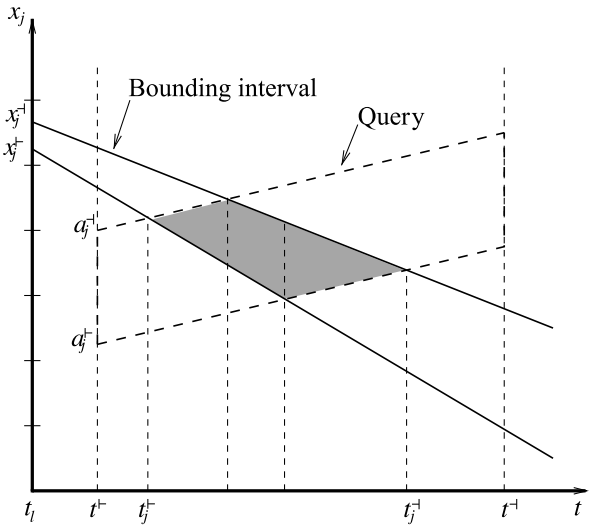
Figure 3 shows an example of the evolution of a bounding rectangle in the TPR-tree computed at $t = 0$. Note that, in contrast to R-trees, bounding rectangles in the TPR-tree are not minimum at all times. In most cases, they are minimum only at the time when they are computed. In a process called *tightening*, whenever an index node is modified during insertions or deletions, the node's time-parameterized bounding rectangle is recomputed, making it minimum at that time.

Querying the TPR-Tree

The TPR-tree can be interpreted as an R-tree for any specific time, t_q. This suggests that the algorithms that are based on the R-tree are easily "portable" to the TPR-tree. For example, answering a timeslice query proceeds as for the regular R-tree, the only difference being that all bound-



Indexing the Positions of Continuously Moving Objects, Figure 3 Example time-parameterized bounding rectangle [4]



Indexing the Positions of Continuously Moving Objects, Figure 4 Intersection of a bounding interval and a query [16]

ing rectangles are computed for the time t_q specified in the query before the intersection is checked.

To answer window queries and moving queries, the algorithm has to check if, in $(\bar{x}, t)$-space, the trapezoid of a query intersects with the trapezoid formed by the part of the trajectory of a bounding rectangle that is between the start and end times of the query. This can be checked using a simple algorithm [16]. Figure 4 demonstrates the intersection between a one-dimensional time parameterized bounding rectangle (interval) and a moving query.

Updating the TPR-Tree

An update of the moving object's position is modeled as a deletion of the old position followed by an insertion of a new position. The TPR-tree's update algorithms are based on the update algorithms of the R*-tree [3], which is an R-tree with improved update algorithms. The update algorithms of the TPR-tree differ from the corresponding algorithms of the R*-tree only in the heuristics that are used. The R*-tree uses heuristics that minimize certain functions, such as the area of a bounding rectangle, the intersection of two bounding rectangles, the margin of a bounding rectangle, and the distance between the centers of two bounding rectangles. The TPR-tree employs time-parameterized bounding rectangles which in turn means that the above-mentioned functions are time dependent, and their evolution in time should be considered. Specifically, given an objective function $A(t)$, the following integral should be minimized:

$$\int_{t_c}^{t_c+H} A(t)\,\mathrm{d}t\,,$$

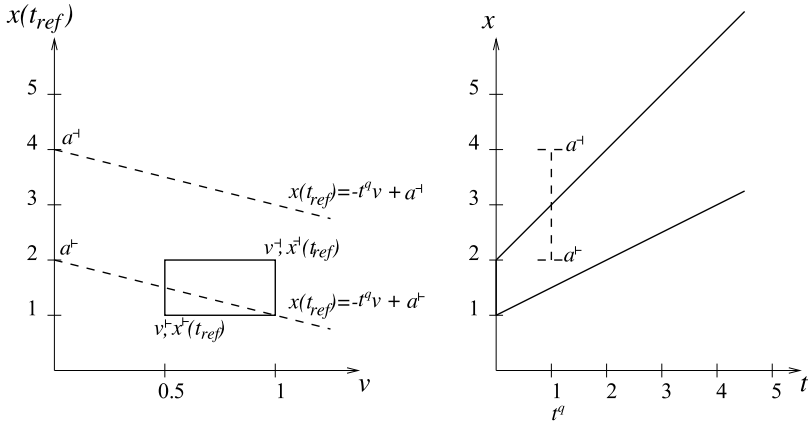
where t_c is the time when the heuristics is being applied and H is the so-called *horizon* parameter that determines how far into the future the queries will "see" the effects of the application of this heuristics. If $A(t)$ is area, the integral computes the area (volume) of the trapezoid that represents part of the trajectory of a bounding rectangle in $(\bar{x}, t)$-space (see Fig. 4).

Duality-Transformation Approach

The general approach of indexing moving points in their native space using a time-parameterized index structure such as the TPR-tree is very closely related to the duality transformation approach [8,13,23].

Considering one-dimensional data, the duality transformation transforms the linear trajectory of a moving point $x = x(t_{\mathrm{ref}}) + v(t - t_{\mathrm{ref}})$ in (x, t)-space into a point $(x(t_{\mathrm{ref}}), v)$, where t_{ref} is a chosen reference time. Then, queries are also transformed.

Bounding points $(x(t_{\mathrm{ref}}), v)$ in the dual space with a minimum bounding rectangle is equivalent to bounding them (as moving points) with a time-parameterized bounding interval computed at t_{ref}. Figure 5 shows the same bounding rectangle and query in $(x(t_{\mathrm{ref}}), v)$-space and in (x, t)-space.



Indexing the Positions of Continuously Moving Objects, Figure 5 Timeslice query (dashed) and bounding interval (solid) in dual ($x(t_{\text{ref}})$, v)-space and (x, t)-space

In spite of such equivalence of bounding rectangles in both approaches, the algorithms used in specific indexes might be quite different. A duality-transformation index may not even explicitly use minimum bounding rectangles [23]. Furthermore, while the heuristics of time-parameterized indexes consider the objects' positions at the time the heuristics are applied, the algorithms of duality-transformation approaches always use a pre-chosen constant t_{ref}. For this reason, duality-transformation approaches usually use two indexes, such that newer updates are put into the second index and, when the first index becomes empty, the indexes change roles [8].

Note that a sufficiently high update rate is crucial for both the time-parameterized indexes, such as the TPR-tree, and the indexes using the duality transformation. If the rate of updates is too low, the indexes suffer the degradation of query performance. The reasons for this are most obvious in the TPR-tree, where the degradation is caused by the expansion of time-parameterized bounding rectangles, resulting in more queries intersecting with a given bounding rectangle.

Key Applications

Online, Position-Aware People, Vehicles, and Other Objects

The rapid and continued advances in positioning systems, e. g., GPS, wireless communication technologies, and electronics in general, renders it increasingly feasible to track and record the changing positions of objects capable of continuous movement. Indexing of such positions is necessary in advanced Location-Based Services (LBS) such as location-based games, tourist-related services, safety-related services, and transport-related services (for example, fleet tracking).

Process Monitoring

Applications such as process monitoring do not depend on positioning technologies. In these, the position of a "moving point" could, for example, be a pair of temperature and pressure values at a specific sensor. A timeslice query would then retrieve all sensors with current measurements of temperature and pressure in given ranges.

Future Directions

Tracking continuous real-world phenomena inevitably involves a high rate of updates that have to be processed by the index. Very recent research provides a number of interesting ideas to speed-up processing of index updates [9,10,22]. Further research is needed to fully explore trade-offs between the update performance and the query performance or the query accuracy. Finally, main-memory indexing of such data could be explored to dramatically boost the performance of index updates.

How to handle the always-present uncertainty regarding the positions of objects has not been sufficiently explored in connection with indexing and warrants further study.

Cross References

- ▶ Indexing, BDual Tree
- ▶ Indexing of Moving Objects, B^x-Tree
- ▶ Indexing, Query and Velocity-Constrained
- ▶ Indexing Schemes for Multi-dimensional Moving Objects
- ▶ Indexing Spatio-temporal Archives
- ▶ Mobile Object Indexing
- ▶ Mobile Objects Databases
- ▶ Nearest Neighbor Query
- ▶ Queries in Spatio-temporal Databases, Time Parameterized
- ▶ R*-tree

Recommended Reading

1. Agarwal, P.K., Arge, L., Erickson, J.: Indexing Moving Points. In: Proceedings of the ACM PODS Symposium, pp. 175–186. Dallas, Texas, USA, 14–19 May 2000

2. Basch, J., Guibas, L.J., Hershberger, J.: Data structures for mobile data. In: Proceedings of ACM-SIAM SODA, pp. 747–756. New Orleans, Lousiana, USA, 5–7 Jan 1997
3. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-tree: An efficient and robust access method for points and rectangles. In: Proceedings of the ACM SIGMOD Conference, pp. 322–331. Brisbane, Queensland, Australia, 13–16 Aug 1990
4. Benetis, R., Jensen, C.S, Karčiauskas, G., Šaltenis, S.: Nearest and Reverse Nearest Neighbor Queries for Moving Objects. The VLDB Journal **15**(3), 229–249 (2006)
5. Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching. In: Proceedings of the ACM SIGMOD Conference, pp. 47–57. Boston, Massachusetts, USA, 18–21 June 1984
6. Jensen, C.S., Lin, D., Ooi, B.C.: Query and Update Efficient B^+-Tree Based Indexing of Moving Objects. In: Proceedings of the VLDB Conference, pp. 768–779. Toronto, Canada, 31 Aug–3 Sept 2004
7. Jensen, C.S., Tiešytė, D., Tradišauskas, N.: Robust B^+-Tree-Based Indexing of Moving Objects. In: Proceedings of the MDM Conference, p. 12, Nara, Japan, 9–13 May 2006
8. Kollios, G., Gunopulos, D., Tsotras, V.J.: On Indexing Mobile Objects. In: Proceedings of the ACM PODS Symposium, pp. 261–272. Philadelphia, Pennsylvania, USA, 31 May–2 June 1999
9. Lee, M.L., Hsu, W., Jensen, C.S., Cui, Teo, K.L.: Supporting Frequent Updates in R-Trees: A Bottom-Up Approach. In: Proceedings of the VLDB Conference, pp. 608–619. Berlin, Germany, 9–12 Sept 2003
10. Lin, B., Su, J.: Handling frequent updates of moving objects. In: Proceedings of ACM CIKM, pp. 493–500. Bremen, Germany, 31 Oct–5 Nov 2005
11. Mokbel, M.F., Ghanem, T.M., Aref, W.G.: Spatio-temporal Access Methods. IEEE Data Engineering Bulletin **26**(2), 40–49 (2003)
12. Nelson, R.C., Samet, H.: A Consistent Hierarchical Representation for Vector Data. In: Proceedings of the ACM SIGGRAPH Conference, pp. 197–206. Dallas Texas, USA, 18–22 Aug 1986
13. Patel, J.M., Chen, Y., Chakka, Y.P.: STRIPES: An Efficient Index for Predicted Trajectories. In: Proceedings of the ACM SIGMOD Conference, pp. 637–646. Paris, France, 13–18 June 2004
14. Prabhakar, S., Xia, Y., Kalashnikov, D.V., Aref, W.G., Hambrusch, S.E.: Query Indexing and Velocity Constrained Indexing: Scalable Techniques for Continuous Queries on Moving Objects. IEEE Transactions on Computers **51**(10), 1124–1140 (2002)
15. Procopiuc, C.M., Agarwal, P.K., and S. Har-Peled.: STAR-Tree: An Efficient Self-Adjusting Index for Moving Objects. In: Proceedings of the ALENEX Workshop, pp. 178–193, San Francisco, California, USA, 4–5 Jan 2002
16. Šaltenis, S., Jensen, C.S., Leutenegger, S.T., Lopez, M.A.: Indexing the Positions of Continuously Moving Objects. In: Proceedings of the ACM SIGMOD Conference, pp. 331–342. Dallas, Texas, USA, 14–19 May 2000
17. Šaltenis, S., Jensen, C.S.: Indexing of Moving Objects for Location-Based Services. In: Proceedings of ICDE, pp. 463–472. San Jose, California, USA, 26 Feb–1 Mar 2002
18. Samet, H.: The Quadtree and Related Hierarchical Data Structures. Computing Surveys **16**(2), 187–260 (1984)
19. Tayeb, J., Ulusoy, O., Wolfson, O.: A Quadtree Based Dynamic Attribute Indexing Method. The Computer Journal **41**(3), 185–200 (1998)
20. Tao, Y., Papadias, D., Sun, J.: The TPR*-Tree: An Optimized Spatio-temporal Access Method for Predictive Queries. In: Proceedings of the VLDB Conference, pp. 790–801. Berlin, Germany, 9–12 Sept 2003
21. Wolfson, O., Xu, B., Chamberlain, S., Jiang, L.: Moving Objects Databases: Issues and Solutions. In: Proceedings of the SSDBM Conference, pp. 111–122. Capri, Italy 1–3 July 1998
22. Xiong, X., Aref, W.G.: R-trees with Update Memos. In: Proceedings of ICDE, p. 22. Atlanta, Georgia, USA, 3–8 Apr 2006
23. Yiu, M. L., Tao, Y., Mamoulis, Y.: The B^{dual}-Tree: indexing moving objects by space filling curves in the dual space. To appear: VLDB Journal, p. 22 (in press)

Indexing Trajectories

► Movement Patterns in Spatio-temporal Data

Indexing, X-Tree

DANIEL KEIM[1], BENJAMIN BUSTOS[2], STEFAN BERCHTOLD[3], HANS-PETER KRIEGEL[4]
[1] Computer Science Institute, University of Konstanz, Konstanz, Germany
[2] Department of Computer Science, University of Chile, Santiago, Chile
[3] stb ag, Augsburg, Germany
[4] Institute for Computer Science, Ludwig-Maximilians-University Munich, Munich, Germany

Synonyms

Extended node tree; Access method, high-dimensional; Split, overlap-free; R-tree; Rectangle, minimum bounding; Rectangle, hyper-

Definition

The *X-tree* (eXtended node tree) [1] is a spatial access method [2] that supports efficient query processing for high-dimensional data. It supports not only point data but also extended spatial data. The X-tree provides *overlap-free split* whenever it is possible without allowing the tree to degenerate; otherwise, the X-tree uses extended variable size directory nodes, so-called supernodes. The X-tree may be seen as a hybrid of a linear array-like and a hierarchical R-tree-like directory.

Historical Background

The R-tree [3] and the R*-tree [4], spatial access methods with a hierarchically structured directory that use minimum bounding rectangles (MBRs) as page regions, have primarily been designed for the management of spatially extended, two-dimensional objects, but have also been used for high-dimensional point data. Empirical studies,

however, show a deteriorated performance of these spatial access methods for high-dimensional data. The major problem of these index structures in high-dimensional spaces is the overlap between MBRs. In contrast to low-dimensional spaces, there is only a few degrees of freedom for splits in the directory. In fact, in most situations, there is only a single good (overlap-free) split axis. An index structure that does not use this split axis will produce highly overlapping MBRs in the directory and thus show a deteriorated performance. Unfortunately, this specific split axis might lead to unbalanced partitions. In this case, a split should be avoided in order to prevent underfilled nodes.

It is well established that in low-dimensional spaces the most efficient organization of the directory is a hierarchical organization. The reason is that the selectivity in the directory is very high which means that, for example, for point queries, the number of required page accesses directly corresponds to the height of the tree. This, however, is only true if there is no overlap between directory rectangles which is very likely for low-dimensional data. It is also reasonable that for very high dimensionality a linear organization of the directory is more efficient. The reason is that due to the high overlap, most of the directory, if not the whole directory, has to be searched anyway. If the whole directory has to be searched, a linearly organized directory needs less space and may be read much faster from disk than a block-wise reading of the directory. For medium dimensionality, an efficient organization of the directory would probably be partially hierarchical and partially linear. The problem is to dynamically organize the tree such that portions of the data which would produce high overlap are organized linearly and those which can be organized hierarchically without too much overlap are dynamically organized in a hierarchical form.

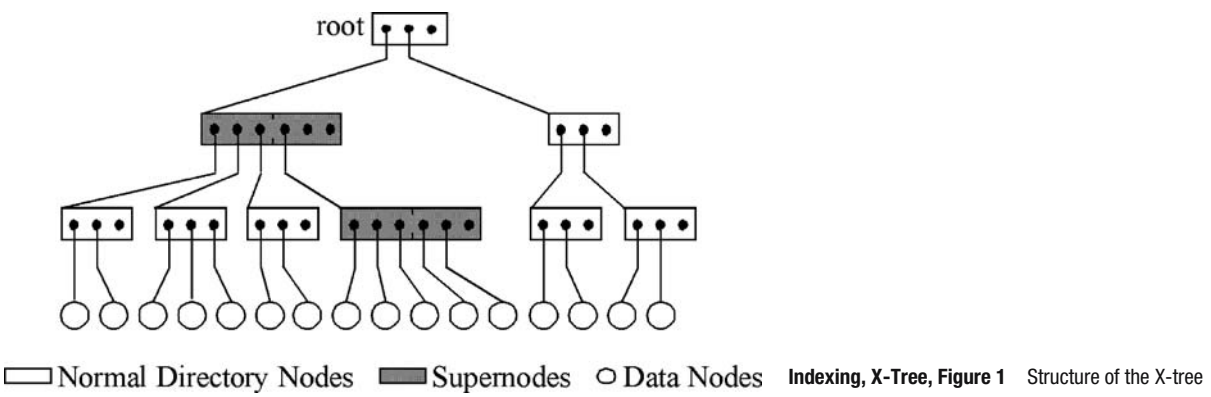
The X-tree is directly designed for the managment of high-dimensional objects and is based on the analysis of problems arising in high-dimensional data spaces. It extends the R*-tree by two concepts: overlap-free split according to a split history and supernodes with an enlarged page capacity. The algorithms used in the X-tree are designed to automatically organize the directory as hierarchically as possible, resulting in a very efficient hybrid organization of the directory.

Scientific Fundamentals

It has been experimentally observed that in high-dimensional spaces a portion of the data space covered by more than one MBR in an R*-tree quickly approaches the whole data space. This is due to the criteria used by the R*-tree to split nodes, which also aims to minimize the volume of the resulting MBRs. The high amount of overlap between MBRs means that, for any similarity query, at least two subtrees must be accessed in almost every directory node, thus reducing the efficiency of the index structure.

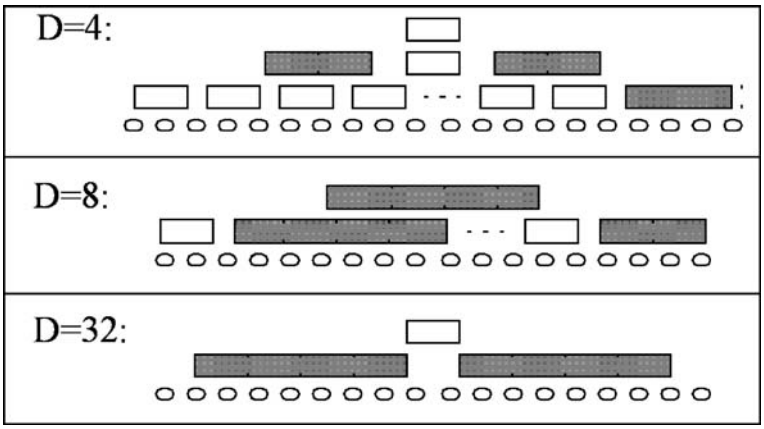
To avoid this problem, the X-tree maintains the history of data page splits of a node in a binary tree. The root of the *split history tree* contains the dimension where an overlap-free split is guaranteed (that is a dimension according to which all MBRs in the node have been previously split). When a directory node overflows, this dimension is used to perform the split. However, the overlap-free split may be unbalanced, i. e., one of the nodes may be almost full and the other one may be underfilled, thus decreasing the storage utilization in the directory. The X-tree does not split in this case, but instead creates a *supernode*. A supernode is basically an enlarged directory node which can store more entries than normal nodes. In this way, the unbalanced split is avoided and a good storage utilization is mantained at the cost of diminishing some of the discriminative power of the index.

The overall structure of the X-tree is presented in Fig. 1. The data nodes of the X-tree contain rectilinear MBRs together with pointers to the actual data objects, and the directory nodes contain MBRs together with pointers to sub-MBRs (see Fig. 2). The X-tree consists of three different types of nodes: data nodes, normal directory nodes, and supernodes. Supernodes are large directory nodes of vari-



Indexing, X-Tree, Figure 1 Structure of the X-tree

Indexing, X-Tree, Figure 2 Structure of a directory node in the X-tree



Indexing, X-Tree, Figure 3 Various shapes of the X-tree in different dimensions

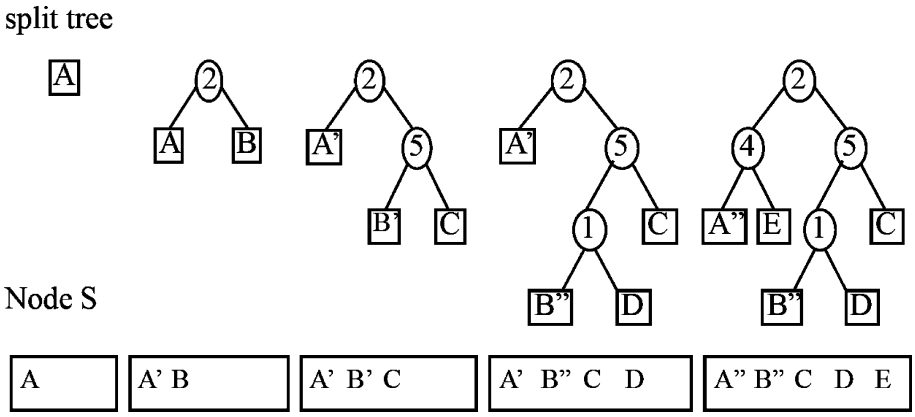
able size (a multiple of the usual block size). The basic goal of supernodes is to avoid splits in the directory that would result in an inefficient directory structure. The alternative to using larger node sizes are highly overlapping directory nodes which would require one to access most of the child nodes during the search process. This, however, is less efficient than linearly scanning the larger supernode.

Note that the X-tree is completely different from an R-tree with a larger block size since the X-tree only consists of larger nodes where actually necessary. As a result, the structure of the X-tree may be rather heterogeneous as indicated in Fig. 1. Due to the fact that the overlap is increasing with the dimension, the internal structure of the X-tree is also changing with increasing dimension. In Fig. 3, three examples of X-trees containing data of different dimensionality are shown. As expected, the number and size of supernodes increases with the dimension. For generating the examples, the block size has been artificially reduced to obtain a drawable fanout. Due to the increasing number and size of supernodes, the height of the X-tree is decreasing with increasing dimension.

The most important algorithm of the X-tree is the insertion algorithm. It determines the structure of the X-tree which is a suitable combination of a hierarchical and a linear structure. The main objective of the algorithm is to avoid splits which would produce overlap. The algorithm first determines the MBR in which to insert the data object and recursively calls the insertion algorithm to actually insert the data object into the corresponding node. If no split occurs in the recursive insert, only the size of the corresponding MBRs has to be updated. In case of a split of the subnode, however, an additional MBR has to be added to the current node which might cause an overflow of the node. In this case, the current node calls the split algorithm which first tries to find a split of the node based on the topological and geometric properties of the MBRs. Topological and geometric properties of the MBRs are, for example, dead-space partitioning, extension of MBRs, etc. The heuristics of the R*-tree [4] split algorithm are an example for a topological split to be used in this step. However, if the topological split results in high overlap, the split algorithm tries next to find an overlap-minimal split which can be determined based on the split history.

For determining an overlap-minimal split of a directory node, one has to find a partitioning of the MBRs in the node into two subsets such that the overlap of the minimum bounding hyperrectangles of the two sets is minimal. In case of point data, it is always possible to find an overlap-free split, but in general it is not possible to guarantee that the two sets are balanced, i. e., have about the same cardinality. It is an interesting observation that an overlap-free split is only possible if there is a dimension according to which all MBRs have been split since otherwise, at least one of the MBRs will span the full range of values in that dimension, resulting in some overlap.

For finding an overlap-free split, a dimension according to which all MBRs of a node S have been previously split has to be determined. The split history provides the necessary information, in particular the dimensions according to which an MBR has been split and which new MBRs have been created by this split. Since a split creates two new MBRs from one, the split history may be represented as a binary tree, called the split tree. Each leaf node of the split tree corresponds to an MBR in S. The internal nodes of the split tree correspond to MBRs which do not exist any more since they have been previously split into new MBRs. Internal nodes of the split tree are labeled by the split axis that has been used; leaf nodes are labeled by the MBR they are related to. All MBRs related to leaves in the left subtree of an internal node have lower values in the



Indexing, X-Tree, Figure 4
Example for the split history

split dimension of the node than the MBRs related to those in the right subtree.

Figure 4 shows an example for the split history of a node *S* and the respective split tree. The process starts with a single MBR *A* corresponding to a split tree which consists of only one leaf node labeled by *A*. For uniformly distributed data, *A* spans the full range of values in all dimensions. The split of *A* using dimension 2 as split axis produces new MBRs *A* and *B*. Note that *A* and *B* are disjointed because any point in MBR *A* has a lower coordinate value in dimension 2 than all points in MBR *B*. The split tree now has one internal node (marked with dimension 2) and two leaf nodes (*A* and *B*). Splitting MBR *B* using dimension 5 as a split axis creates the nodes *B* and *C*. After splitting *B* and *A* again, the situation depicted in the right most tree of Fig. 4 is reached, where *S* is completely filled with the MBRs *A*, *B*, *C*, *D*, and *E*.

One may find an overlap-free split if there is a dimension according to which all MBRs of *S* have been split. To obtain the information according to which dimensions an MBR *X* in *S* has been split, the split tree has to be traversed from the root node to the leaf that corresponds to *X*. For example, MBR *C* has been split according to dimensions 2 and 5 since the path from the root node to the leaf *C* is labeled with 2 and 5. Obviously, all MBRs of the split tree in Fig. 4 have been split according to dimension 2, the split axis used in the root of the split tree. In general, all MBRs in any split tree have one split dimension in common, namely the split axis used in the root node of the split tree.

The partitioning of the MBRs resulting from the overlap-minimal split, however, may result in underfilled nodes which is unacceptable since it leads to a degeneration of the tree and also deteriorates the space utilization. If the number of MBRs in one of the partitions is below a given threshold, the split algorithm terminates without providing a split. In this case, the current node is extended to become a supernode of twice the standard block size. If the same case occurs for an already existing supernode, the supernode is extended by one additional block. Obviously, supernodes are only created or extended if there is no possibility of finding a suitable hierarchical structure of the directory. If a supernode is created or extended, there may not be enough contiguous space on disk to sequentially store the supernode. In this case, the disk manager has to perform a local reorganization.

The algorithms to query the X-tree (point, range, and nearest neighbor queries) are similar to the algorithms used in the R*-tree since only minor changes are necessary in accessing supernodes. The delete and update operations are also simple modifications of the corresponding R*-tree algorithms. The only difference occurs in case of an underflow of a supernode. If the supernode consists of two blocks, it is converted to a normal directory node. Otherwise, that is if the supernode consists of more than two blocks, the size of the supernode is reduced by one block. The update operation can be seen as a combination of a delete and an insert operation and is therefore straightforward.

Key Applications

In many applications, indexing of high-dimensional data has become increasingly important. In multimedia databases, for example, the multimedia objects are usually mapped to feature vectors in some high-dimensional space and queries are processed against a database of those feature vectors [5]. This feature-based approach is taken in many other areas including CAD [6], 3D object databases [7], molecular biology (for the docking of molecules [8]), medicine [9], string matching and sequence alignment [10], and document retrieval [11]. Examples of feature vectors are color histograms [12], shape descriptors [13], Fourier vectors [14], text descriptors [15], etc. In some applications, the mapping process does not yield point objects, but extended spatial objects in a high-dimen-

sional space [16]. In many of the mentioned applications, the databases are very large and consist of millions of data objects with several tens to a few hundreds of dimensions.

Future Directions

The feature-based approach has several advantages compared to other approaches for implementing similarity searches. The extraction of features from the source data is usually fast and easily parametrizable, and metric functions for feature vectors, as the Minkowski distances, can also be efficiently computed. Novel approaches for computing feature vectors from a wide variety of unstructured data are proposed regularly. As in many practical applications where the dimensionality of the obtained feature vectors is high, the X-tree is a valuable tool to perform efficient similarity queries in spatial databases.

Cross References

- ► Indexing and Mining Time Series Data
- ► Nearest Neighbor Query
- ► R*-tree

Recommended Reading

1. Berchtold, S., Keim, D., Kriegel, H.P.: The X-tree: An index structure for high-dimensional data. In: Proc. 22th International Conference on Very Large Databases (VLDB'96), pp. 28–39. Morgan Kaufmann, San Francisco, CA, USA (1996)
2. Böhm, C., Berchtold, S., Keim, D.: Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. ACM Computing Surveys **33**(3), 322–373 (2001)
3. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: Proc. ACM International Conference on Management of Data (SIGMOD'84), pp 47–57. ACM Press, New York, NY, USA (1994)
4. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The R*-tree: An efficient and robust access method for points and rectangles. In: Proc. ACM International Conference on Management of Data (SIGMOD'90), pp. 322–331. ACM Press, New York, NY, USA (1990)
5. Faloutsos, C.: Searching Multimedia Databases by Content. Kluwer Academic Publishers, Norwell, MA, USA (1996)
6. Berchtold, B., Keim, D., Kriegel, H.-P.: Using extended feature objects for partial similarity retrieval. The VLDB Journal **6**(4), 333–348 (1997)
7. Bustos, B., Keim, D., Saupe, D., Schreck, T., Vranić, D.: Feature-based similarity search in 3D object databases. ACM Computing Surveys **37**(4), 345–387 (2005)
8. Shoichet, B.K., Bodian, D.L., Kuntz, I.D.: Molecular docking using shape descriptors. Journal of Computational Chemistry **13**(3), 380–397 (1992)
9. Keim, D.: Efficient geometry-based similarity search of 3D spatial databases. In: Proc. ACM International Conference on Management of Data (SIGMOD'99), pp. 419–430. ACM Press, New York, NY, USA (1999)
10. Altschul, S. F., Gish, W., Miller, W., Myers, E.W., Lipman D.J.: A basic local alignment search tool. Journal of Molecular Biology **215**(3), 403–410 (1990)
11. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA (1999)
12. Shawney, H., Hafner, J.: Efficient Color Histogram Indexing. In: Proc. International Conference on Image Processing, pp. 66–70. IEEE Computer Society, Los Alamitos, CA, USA (1994)
13. Jagadish, H.V.: A retrieval technique for similar shapes. In: Proc. ACM International Conference on Management of Data (SIGMOD'91), pp. 208–217. ACM Press, New York, NY, USA (1991)
14. Wallace, T., Wintz, P.: An efficient three-dimensional aircraft recognition algorithm using normalized fourier descriptors. Computer Graphics and Image Processing **13**, 99–126 (1980)
15. Kukich, K.: Techniques for automatically correcting words in text. ACM Computing Surveys **24**(4), 377–440 (1992)
16. Murase, H., Nayar, S.K.: Three-dimensional object recognition from appearance-parametric eigenspace method. Systems and Computers in Japan **26**(8), 45–54 (1995)

Indoor Geolocation

- ► Channel Modeling and Algorithms for Indoor Positioning

Indoor Localization

MOUSTAFA YOUSSEF
Department of Computer Science,
University of Maryland, College Park, MD, USA

Synonyms

Indoor location determination systems; Tracking; Reference, symbolic; Reference, coordinate based; Position, absolute; Position, relative; FCC 94-102; GPS; Time of flight; Arrival, Time of; Arrival, Angle of; Range Combining; Trilateration; Triangulation; Multilateration; Landmark proximity; WiMAX

Definition

Indoor localization refers to tracking objects in an indoor environment. This tracking can be either in 2-dimensions, 3-dimensions, or 2.5-dimensions. 2.5-dimensions refers to the case when the object position is tracked at discrete plans of the 3-dimensional space, rather than the entire continuum of the 3-dimensional space. For example, tracking a person in multiple 2-dimensional floor-plans in a 3-dimensional building can be considered a 2.5-dimensional tracking.

An indoor location determination system can report the estimated location as a symbolic reference, for exam-

ple, "the lounge", or as a coordinate-based reference. For the coordinate-based reference, the reported tracked-object position can be either relative or absolute. Relative positioning refers to the case where the returned position is relative to a reference point, for example, the x and y coordinates of the position relative to the origin of the map. On the other hand, absolute positioning refers to the case when the returned position is in absolute coordinates, such as the longitude, altitude, and height coordinates.

An indoor location determination system can be centralized or distributed. In a centralized implementation, all the computations are implemented on a centralized server, relieving the computational load from the energy-constrained mobile devices. For a distributed-implementation, the location estimation is performed at the mobile devices. This allows better scalability for the system and, for human tracking, it allows better control over privacy.

Historical Background

Location determination systems have been an active research area for many years. Since the 1970's, the Global Positioning System (GPS) has been a well known and widely used location determination system. However, the GPS requires a line-of-sight to the satellites and, hence, is not suitable for high-accuracy indoor localization.

Wide-area cellular based systems have been active in developing location determination systems for locating cellular users motivated by the FCC 94-102 order, mandating wireless E911. E911 refers to automatically locating cellular callers who dial the emergency 911 number equivalent to the wired 911 service.

A number of indoor location determination systems have been proposed over the years, including: infrared, ultrasonic, computer vision, and physical contact. All of these technologies share the requirement of specialized hardware, leading to more deployment and maintenance costs and poor scalability.

In the last few years, researchers have started looking at location determination systems that do not require any additional hardware. For example, in an 802.11 WLAN, the wireless card uses the signal strength returned form the access points to roam between different access points. This signal strength information is available to the application level. Therefore, a location determination system based on signal strength information in an 802.11 network, such as the *Horus* system [20], can be implemented without requiring any specialized hardware. This has the advantage of increasing the utility of the data network. A similar idea can be applied to FM radio signals to determine the location of an FM receiver [10] and also to Bluetooth networks [15].

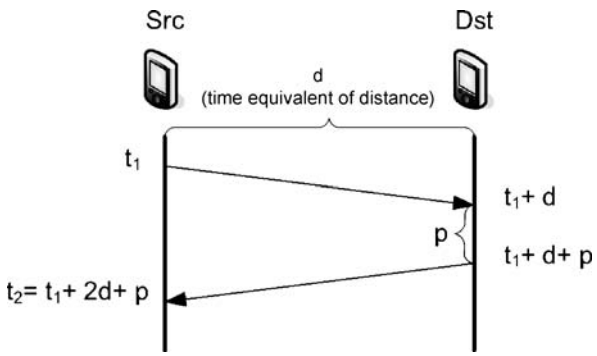
Scientific Fundamentals

The basic idea used in a location determination system is to measure a physical quantity that is proportional to distance and use the measured value to estimate the distance to a reference point. This process is called ranging. Once the distance is known to one or more reference points, the position of the tracked object can be determined. This process is called range-combining. All location determination systems use these two processes, ranging and range-combining, either explicitly or implicitly. For example, in the GPS system, the reference points are the satellites and the physical quantity used is the time it takes the signal to travel from the satellite to the GPS receiver. The more time it takes the signal to travel from the satellite to the GPS receiver, the larger the distance between them.

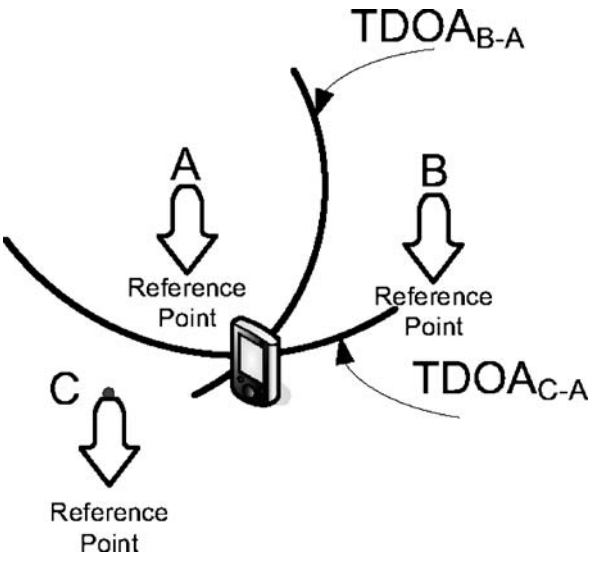
Ranging Techniques

Examples of the signals that can be used in ranging include: Time-of-flight, Angle-of-arrival, and Signal-strength.

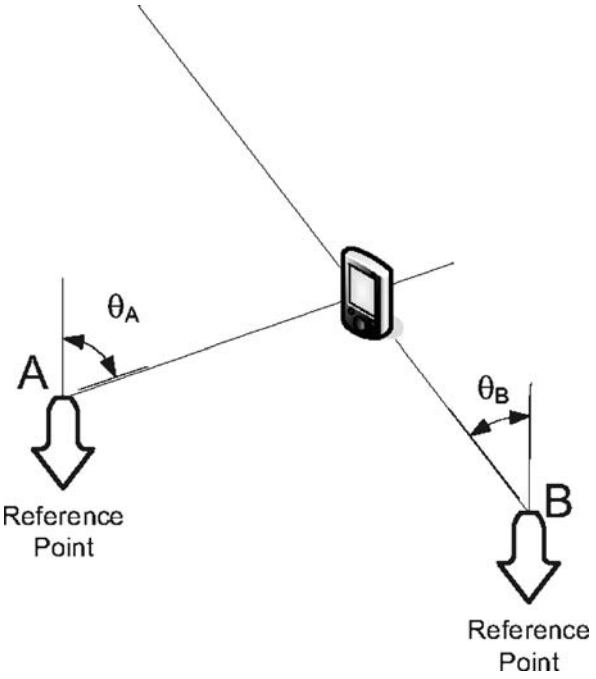
Time-of-flight based techniques depend on measuring the time the signal takes to travel from the transmitter to the receiver (called Time-of-Arrival or TOA), the difference of the arrival time at two or more receivers (called Time-Difference-of-Arrival or TDOA), or the time it takes two different signals to reach the receiver from the same transmitter. For example, the system in [13] presented a location technique based on TOA obtained from Round-Trip-Time measurements (Fig. 1) at data link level of the 802.11 protocol. In their system, the sender sends a frame and includes the send timestamp, t_1, in it. As soon as the receiver gets the frame, it sends it back. When the sender gets the frame, it gets a timestamp of the receive time, t_2. The time difference between sending and receiving the frame, $t_2 - t_1$, is used as an estimate of the distance between the



Indoor Localization, Figure 1 The echoing technique for estimating the time of flight. p is the processing delay at the destination. The estimated time is $\frac{t_2 - t_1}{2} = d + \frac{p}{2}$



Indoor Localization, Figure 2 TDOA systems use the principle that the transmitter location can be estimated by intersection of the hyperbolae of the constant differential TOA of the signal at two or more pairs of base stations



Indoor Localization, Figure 3 Based on the estimated AOA of a signal at two or more reference points, the location of the desired unit can be determined

sender and receiver. Note that the processing time established at the receiver affects the accuracy of the estimated distance.

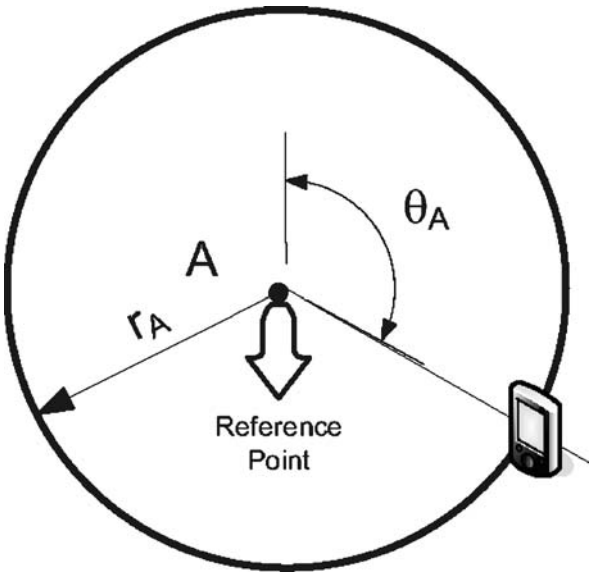
TDOA systems use the principle that the transmitter location can be estimated by the intersection of the hyperbolae of a constant differential TOA of the signal at two or more pairs of base stations (Fig. 2). The idea here is that a message transmitted from the sender is received by three or more receivers. Instead of keeping track of the absolute TOA of the signal, TDOA-based systems keep track of the difference of reception times of the transmitted message at the different receivers. Given two receivers' locations and a known TDOA, the locus of the sender location is a hyperboloid. For more than two receivers, the intersection of the hyperbolae associated with the TDOA of each pair of receivers provides the final transmitter's location.

Systems that use two different physical signals, e. g., the Cricket System [16], which uses ultrasound and RF signals, use one of the signals for synchronization and the other for the time estimation. The idea is that the speed of the ultrasonic signal is much lower than the speed of the RF signal. Therefore, when the sender transmits an RF signal followed by an ultrasound signal, the receiver can use the difference in time between the reception of the ultrasound signal and the RF signal as an estimate of the distance between the sender and the receiver since the time it takes the RF signal to reach the receiver is negligible compared to the time it takes the ultrasound signal to reach the same receiver.

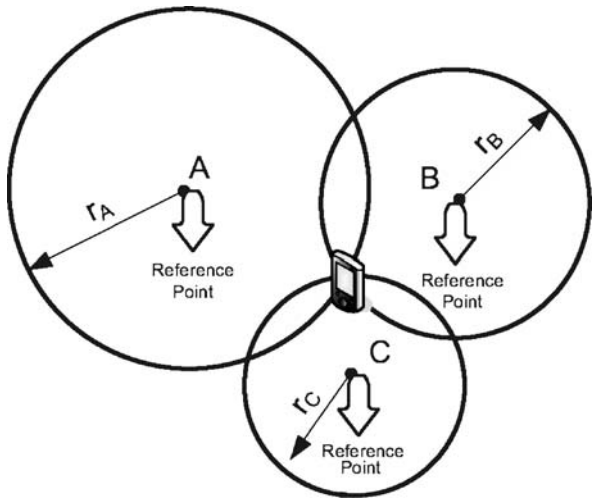
Angle-of-arrival (AOA) based techniques use antenna arrays to estimate the angle of arrival of the signal from the transmitter. The idea is to measure the difference of arrival time of the signal at individual elements of the array and use the delay to estimate the AOA. Based on the estimated AOA of a signal at two or more reference points, the location of the desired unit can be determined as the intersection of a number of lines (Fig. 3).

Signal-strength based techniques, e. g., the *Horus* system, use the signal strength received from a reference point as an estimate of how close the reference point is. For outdoor environments, the relation between signal strength and distance can be approximated by a logarithmic function. However, for indoor environments, this relation is very complex due to multiple phenomena, such as the multipath effect and signal diffraction, among others. Therefore, indoor location determination systems that use signal strength usually use a lookup table to store the relation between the signal strength and distance. This table has been called a "radio-map" in the literature.

An example of another possibility that implies an implicit measurement of a physical quantity is the Cell-ID based method. In **Cell-ID** based methods, e. g., RF-IDs, the location of the transmitter is taken to be the location of the nearest base station it is associated with.



Indoor Localization, Figure 4 Combining propagation time measurement with angle measurement to obtain the position estimate can be done by using only one reference point



Indoor Localization, Figure 5 Trilateration locates a node by calculating the intersection of three circles. If the ranges contain an error, the intersection of the three circles may not be a single point

Hybrid methods can be used that combine two or more of these techniques. For example, combining propagation time measurement with angle measurement to obtain the position estimate can be done by using only one reference point (Fig. 4).

To obtain these physical measurements, different underlying communication technologies can be used. This includes infrared, ultrasonic, radio frequency (RF), computer vision, and physical contact, among others.

Range Combining Techniques

Once the range information is gathered from a number of reference points, it needs to be combined to estimate the location. This process is called Range-combining. Trilateration, triangulation, multilateration, and landmark-proximity are the most common techniques for combining ranges.

Trilateration refers to locating a node by calculating the intersection of three circles (Fig. 5). Each circle is centered at a reference point with a radius equal to the estimated range between the reference point and the node. If the ranges contain errors, the intersection of the three circles may not be a single point.

Triangulation is used when the angle of the node, instead of the distance, is estimated, as in AOA methods. The nodes' positions are calculated in this case by using the trigonometry laws of sines and cosines. In this case, at least two angles are required.

In **multilateration**, the position is estimated from distances to three or more known nodes by minimizing the error between the estimated position and the actual position by solving a set of non-linear equations.

Proximity-based techniques are usually used when no range information is available. For example, the GPS-less system [6] employs a grid of beacon nodes with known locations; each unknown node sets its position to the centroid of the beacon locations it is connected to.

Key Applications

Indoor localization can be used in many applications, most notably in context-aware applications and enhancing network protocols.

Context-Aware Applications

The context of an application refers to the information that is part of its operating environment. Typically, this includes information such as location, activity of people, and the state of other devices. Algorithms and techniques that allow an application to be aware of the location of a device on a map of the environment are a prerequisite for many of these applications. Examples of location-aware applications [7,8,18] include location-sensitive content delivery, where tailored information is sent to the user based on his current location, direction finding, asset tracking, teleporting, robotics, and emergency notification.

Asset Tracking Location information can be used to track assets in indoor environments. For example, RF-IDs

have been widely used for tracking assets in military and civilian applications. Note that the technologies that can be used for asset tracking can also be used for tracking humans, such as in [19].

Direction Finding Another important application for indoor localization is to find the direction and route between two points. This is similar to the GPS-based navigation systems in cars today, however, it is applied to indoor environments. For example, in the Shopping Assistance system [3], the device can guide the shoppers through the store, provide details of items, help locate items, point out items on sale, do a comparative price analysis, and so forth. There is a privacy concern since the store maintains the customer profiles. As a consequence, customers are divided into two classes. The first class is the regular customers who shop anonymously without profiles in the store. The second class is the store customers who signed up with a store will get additional discounts in exchange for sacrificing their privacy.

Guided Tours In guided tour applications, e. g., [2], information can be displayed on a device carried by a user based on the device's current location. The user can also leave comments on an interactive map. This kind of tailored information enhances the user experience.

Call Forwarding In this application[17], based on the Active Badge System, the user location is tracked in a central server that is connected to the enterpriser phone system. Whenever a call arrives to a user who is not currently in his office, the call is automatically routed to the room the user is located based on his/her current location.

Teleporting The Teleporting System [4], developed at the Olivetti Research Laboratory (ORL), is a tool for experiencing obile applications The system allows users to dynamically change the display device from which their currently running applications are accessible. It operates within the X Window System and allows users to interact with their existing X applications at any X display within a building. The process of controlling the interface to the teleporting system comes from the use of an automatically maintained database of the location of equipment and people within the building.

Robotics Finding the location of robots in indoor environments is crucial in many applications. For example, the system in [11] uses radio frequency identification (RFID) for robot-assisted indoor navigation for the visually impaired. Robots equipped with RFIDs and laser range finders allow visually impaired individuals to navigate in unfamiliar indoor environments and interact with the robotic guide via speech, sound, and wearable keyboards.

Network Protocols Enhancements

The second class of applications for indoor location determination systems is enhancements for network protocols. This usually applies to sensor network applications for indoor environments. These enhancements include determining the location of an event, location-based routing, node identification, and node coverage.

Determining the Location of an Event Determining the location of an event is an important service that is particularly important in indoor sensor networks [5]. In indoor sensor networks, it is always important to record the location of an event whenever the event occurs. This highlights the importance of indoor location determination systems in such applications.

Location-Based Routing A number of location based routing protocols have been proposed for using the location information of the sender and receiver to achieve scalable routing protocols. For example, the GPSR protocol [9] exploits the correspondence between geographic position and connectivity in a wireless network by using the positions of nodes to make packet forwarding decisions compared to the standard routing protocols that use graph-theoretic notions of shortest paths and transitive reachability in order to find routes. GPSR uses greedy forwarding to forward packets to nodes that are always progressively closer to the destination. In regions of the network where such a greedy path does not exist (i. e., the only path requires that one move temporarily farther away from the destination), GPSR recovers by forwarding in perimeter mode, in which a packet traverses successively closer faces of a planar subgraph of the full radio network connectivity graph until reaching a node closer to the destination where greedy forwarding resumes.

Node ID The inspection of building structures, especially bridges, is currently made by visual inspection [12]. The few non-visual methodologies make use of wired sensor networks which are relatively expensive, vulnerable to damage, and time consuming to install. Recordings of structures during ambient vibrations and seismic disturbances are essential in determining the demand placed upon those structures. For structures in high seismic areas, information provided by monitoring structural responses will inevitably lead to a better scientific understanding of how structures behave in the nonlinear realm. Using structure monitoring sensor networks is vital in these environ-

ments. It is a challenge for such huge indoor sensor networks to determine the node IDs for a large number of randomly placed nodes. Location determination can be used as node identification in these environments where the node location is used as its ID.

Node Coverage One of the fundamental issues that arises in sensor networks is coverage. Coverage can be considered as the measure of quality of service of a sensor network [14]. For example, in a fire detection sensor network scenario, one may ask how well the network can observe a given area and what the chances are that a fire starting in a specific location will be detected in a given time frame. Furthermore, coverage formulations can try to find weak points in a sensor field and suggest future deployment or reconfiguration schemes for improving the overall quality of service. For the coverage problem, knowing the nodes' locations is essential for protocols that address this problem.

Future Directions

New different technologies, e. g., WiMax, are being developed that will allow larger transmission ranges and more accurate measurements of the physical quantities. This should allow more accurate and ubiquitous localization. As more accurate localization techniques are being introduces, a new set of applications are emerging to take advantage of these localization capabilities, including GPS-less city wide localization [1].

Cross References

- ► Channel Modeling and Algorithms for Indoor Positioning
- ► Indoor Positioning, Bayesian Methods
- ► Radio Frequency Identification (RFID)

Recommended Reading

1. http://www.skyhookwireless.com/
2. Abowd, G.D., Atkeson, C.G., Hong, J., Long, S., Kooper, R., Pinkerton, M.: Cyberguide: A mobile context-aware tour guide. Wireless Networks **3**(5), 421–433 (1997)
3. Asthana, A., Cravatts, M., Krzyzanowski, P.: An indoor wireless system for personalized shopping assistance. In: IEEE Workshop on Mobile Computing Systems and Applications, pp. 69–74. Santa Cruz, California (1994)
4. Bennett, F., Richardson, T., Harter, A.: Teleporting - making applications mobile. In: IEEE Workshop on Mobile Computing Systems and Applications, pp. 82–84. Santa Cruz, California (1994)
5. Brooks, A., Makarenko, A., Kaupp, T., Williams, S., Durrant-Whyte, H.: Implementation of an indoor active sensor network. In: Int. Symp. on Exp. Robotics, Singapore (2004)
6. Bulusu, N., Heidemann, J., Estrin, D.: GPS-less Low-cost Outdoor Localization for Very Small Devices. In: IEEE Personal Communications (2000)
7. Chen, G., Kotz, D.: A survey of context-aware mobile computing research. Technical Report Dartmouth Computer Science Technical Report TR2000–381 (2000)
8. Gellersen, H.-W., Schmidt, A., Beigl, M.: Adding some smartness to devices and everyday things. In: IEEE Workshop on Mobile Computing Systems and Applications, pp. 3–10. December (2000)
9. Karp, B., Kung, H.T.: Greedy perimeter stateless routing for wireless networks. In: Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000), pp. 243–254: Boston, MA (2004)
10. Krumm, J., Cermak, G., Horvitz, E.: Rightspot: A novel sense of location for a smart personal object. In: Fifth International Conference on Ubiquitous Computing, October (2003)
11. Kulyukin, V., Gharpure, C., Nicholson, J., Pavithran, S.: RFID in robot-assisted indoor navigation for the visually impaired. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004, pp. 1979–1984. (2004)
12. Lynch, J.P., Kiremidjian, A.S., Law, K.H., Kenny, T., Carryer, E.: Issues in wireless structural damage monitoring technologies. In: 3rd World Conference on Structural Control (WCSC), Como, Italy (2002)
13. Sebastiano, C., Ciurana, M., Barcelo,F.: Indoor tracking in wlan location with toa measurements. In: MobiWac 2006 (2006)
14. Meguerdichian, S., Koushanfar, F., Potkonjak, M., Srivastava, M.B.: Coverage problems in wireless ad-hoc sensor networks. In: INFOCOM, pp. 1380–1387 (2001)
15. Peddemors, A.J.H., Lankhorst, M.M., de Heer, J.: Presence, Location, and Instant Messaging in a Context-Aware Application Framework. Proc. of the Intl. Conf. on Mobile Data Management (MDM) **2574**, 325–330
16. Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The cricket location-support system. In: 6th ACM MOBICOM, Boston, MA, August (2000)
17. Want, R., Hopper, A., Falcão, V., Gibbons, J.: The active badge location system. ACM Transactions on Information Systems **10**(1), 91–102 (1992)
18. Want, R., Schilit, B.: Expanding the horizons of location-aware computing. IEEE Computer **34**(8), 31–34 (2001)
19. Want, R., Schilit, B.N., Adams, N.I., Gold, R., Petersen, K., Goldberg, D., Ellis, J.R., Weiser, M.: An overview of the parctab ubiquitous computing experiment. IEEE Personal Communications **2**(6), 28–43 (1995)
20. Youssef, M., Agrawala, A.: The Horus WLAN Location Determination System. In: Third International Conference on Mobile Systems, Applications, and Services (MobiSys 2005), June (2005)

Indoor Location Determination Systems

► Indoor Localization

Indoor Location Estimation

► Channel Modeling and Algorithms for Indoor Positioning

Indoor Position Estimation

► Channel Modeling and Algorithms for Indoor Positioning

Indoor Positioning

CLAUDE CARON[1],
DANIEL CHAMBERLAND-TREMBLAY[1],
CÉDRIC LAPIERRE[1], PIERRE HADAYA[3],
STÉPHANE ROCHE[2], MOKHTAR SAADA[1]
[1] GeoBusiness Group, Faculty of Administration, University of Sherbrooke, Sherbrooke, QC, Canada
[2] Center for Research in Geomatics, Pavillon Louis-Jacques-Casault, University of Laval, Québec, QC, Canada
[3] Department of Management and Technology, University of Québec at Montréal, Montréal, QC, Canada

Synonyms

Indoor positioning system; Microgeomatics; Real-time location services; Location-based services; Spatial statistical analysis; Smart buildings

Definition

Indoor positioning is a technique that provides the continuous real-time location of objects or people within a closed space through measurements [1]. It is primarily used in retail floors, warehouses, factories, and offices to monitor and track people, equipments, merchandise, etc. Contrary to self-positioning systems such as GPS, indoor positions are calculated on a distant server using the information transmitted by mobile tags [2]. Indoor positioning systems (IPS) may have different configurations. For example, tags may transmit movement information directly through a wireless network or may be read by scanners as they pass by. In the latter case, tags possess no processing capabilities and are unable to calculate their own position. Also, different radio frequencies can be used for indoor positioning: namely, ultrasound, a wireless LAN-based signal such as wireless fidelity (WiFi) or Bluetooth, and cellular network signals. Finally, to be completed, the process of indoor positioning usually requires data analysis (e. g., validation, spatial analysis, geostatistics, data mining) to extract patterns and trends as well as to provide accurate and timely knowledge to managers.

Historical Background

Dedicated IPS have emerged from the need to have accurate repeated location measurements of tangibles such as humans and equipments. Hightower and Borrielo [3] trace back the origin of these systems to the "Active Badge" project [4] which aimed at replacing the traditional "call and report back" pager approach to locate people.
Weiser [5,6] was one of the first to recognize the importance and potential of indoor location and viewed it as a central component of ubiquitous computing. In articulating his vision of the next generation computing, he identified three innovations that would transform computer science [7]:

- Computing devices will become embedded in everyday objects and places.
- Designers will develop intuitive, intelligent interfaces for computing devices to make them simple and unobtrusive for users.
- Communications networks will connect these devices together and will evolve to become available anywhere and anytime.

In time, development efforts in the field of indoor location technologies echoed these principles. Today, the field offers great opportunities for context-aware computing, whether ubiquitous, pervasive or mobile (for examples, please refer to Future Directions, below).
Indoor positioning also borrows from the great tradition of land surveying. Over the last 20 years, this field of research has embraced computer science and new technologies to transform itself into what is known nowadays as geographic information sciences (GISciences). The focus of GISciences has broadened from geocentric measurements (land registry, natural resources and public utilities management) to include novel application domains such as business (e. g., geomarketing, fleet management), communication (e. g., peer finder) and sports (e. g., geocaching, precision training). Evidently, information technology innovations such as wireless communication (e. g., Wi-Fi, WiMax), mobile devices [e. g., presonal digital assistants (PDAs), mobile phones] and new location techniques [e. g., assisted global positioning system (A-GPS) on mobile phones] are now important components of GISciences. Today, indoor positioning is able to take advantage of the topological analysis and mapping expertise found in GISciences.
Indoor positioning did not evolve from GISciences, however. Devices emerged from various technical fields. Indeed, Beresford [8] provides examples of mechanical, magnetic, acoustic, radio and microwave, optical, and inertial systems. Of these, radio-frequency-based systems, namely radio frequency identification (RFID), attracts

most attention among scholars and practitioners alike. New applications of RFID technologies are appearing rapidly and have profound impacts on processes and strategies of organizations (Batty 2004). RFID technology appeared in the 1940s, and was initially used mainly for military purposes [9] but in the 1990s, new standards were developed and they contributed to the deployment of the technology in numerous other contexts. In contrast to GPS, the RFID approach is foreign to the GISciences. It is rooted in the field of supply chain management (SCM) and viewed as a powerful replacement for barcode systems. The integration of RFID into spatial analysis and geostatistics has led the way to a new domain of expertise: "microgeomatics" [10]. Microgeomatics is the field of real-time tracking and spatial analysis of people and equipment motions in a closed space, such as merchandise in a warehouse (e. g., Walmart), customers in a retail store, airline luggage, family pets and medical equipments. Such applications are currently deployed in an increasing number of organizations [11], where important financial investments and operational adjustments are generally required. Indoor positioning capabilities play a highly strategic role for businesses. The best strategic fit appears where the appropriate identification and location data can be integrated to the management systems such as ERP, CRM and SCM.

Scientific Fundamentals

In order to properly understand the scientific fundamentals of indoor positioning, this section will highlight the telecommunication methods, describe the two dominant positioning methods, and) detail the positioning techniques adopted in the field.

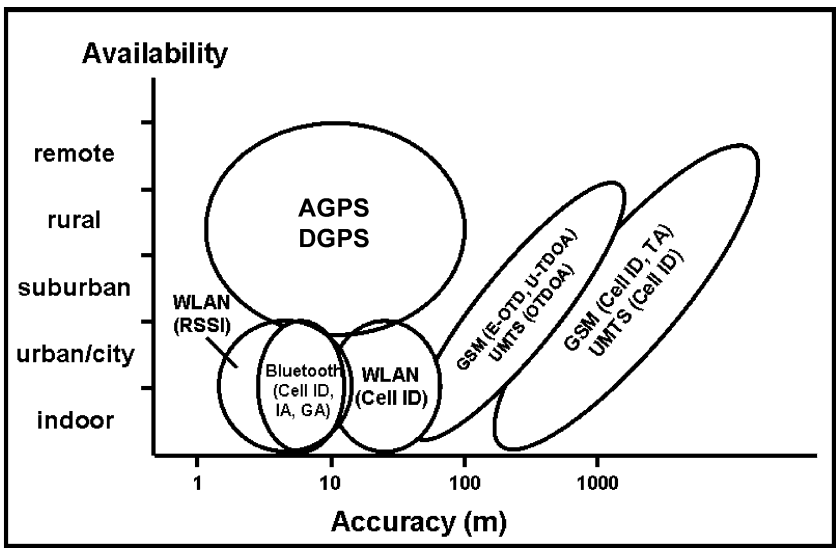
Telecommunication Methods

Various telecommunication methods can be used by location systems: radio frequencies [RFID/Wi-Fi/cellular/Bluetooth/ultrawideband (UWB)], infrared radiation, ultrasound, optics and electromagnetic fields to name a few [12,13]. The use of radio frequencies remains, however, the most popular telecommunication approach adopted for indoor positioning because of its ease of deployment and low cost. Even though field studies have demonstrated that any radio frequency (RF) can be used for indoor positioning, the choice of radio frequencies should vary according to the accuracy requirement and the positioning environment (Fig. 1). In the context of indoor systems, numerous frequencies can be used: Wi-Fi (IEEE 802.11), bluetooth (IEEE 802.15), wide-area cellular and GPS/UWB [12].

Two Dominant Indoor Positioning Methods

The RFID Method An RFID-based system is "an emerging technology intended to identify, track, and trace items automatically" [15]. This positioning method is claimed to add intelligence and minimize human intervention in the identification process by using electronic tags. An RFID application is comprised, at a minimum, of four components: tags, antennae, readers and software.

An RFID tag is a transponder usually placed on objects to identify their locations. It is made of a coil, an antenna and a microchip. One way of classifying tags is to divide them into passive and active tags. Passive tags are relatively cheap, have a short range of detection, are read-only, and are powered by remote antenna signals. In contrast, active tags [often called real-



Indoor Positioning, Figure 1 Accuracy per positioning environment [14]. *AGPS*, Assisted Global Positioning System; *DGPS*, Differential Global Positioning System; *WLAN*, Wireless Local Area Network; *Cell ID*, Cell Identification; *IA*, Incremental Algorithm; *GA*, Geometric Algorithm; *GSM*, Global System for Mobile communications; *E-OTD*, Enhanced Observed Time Difference; *U-TDOA*, Uplink Time Difference of Arival; *UMTS*, Universal Mobile Telecommunications System; *OTDOA*, Observed Time Difference of Arival; *TA*, Timing Advance based positioning; *RSSI*, Received Signal Strength Indicator

time location systems (RTLS)], are more expensive, have a longer read/write range, offer greater functionality and must be battery powered. The battery of the active tag allows for repeated and autonomous transmissions of radio waves used for positioning; its life span is usually up to 5 years.

Passive RFID tags are slowly replacing barcodes as they become less expensive. They provide automatic contactless capture of information and do not require line-of-sight to work. RFID tags, passive and active alike, are used to manage various processes (e. g., warehouse, logistics, access control) in numerous industries (e. g., agriculture, healthcare) [9].

In recent years, researchers have tackled the problem of real-time positioning using RFID passive and/or active tags. Two approaches have been developed [16]. The first one requires fixing the tags on objects inside a room and placing a short-range receiver on the person or object in motion. This technique is constrained by the size of the antenna and the power requirement, thus limiting receivers to short-range models. The second method requires installing powerful antennae in a room to create zones and attaching tags to the objects or persons in motion.

Antennae come in many shapes and forms, and present different technical characteristics. They vary in size from less than a square centimeter to several square meters. Ultra-high frequency (UHF) reader antennae can be classified as circular-polarized or linear-polarized antennae [15]. Circular-polarized antennae emit and receive radio waves from all directions, are less sensitive to transmitter-receiver orientation and work better "around corners". Linear-polarized antennae work in one particular direction but have a longer range.

Readers are used to query or read all the tags within their range in quick succession. In interrogation, the reader sends a signal to the tag and listens. In reading, active tags continually send out signals to the reader. To read passive tags, the reader sends them radio waves which energize the tags as they start broadcasting their data [15]. The range of passive tags increases every year. It was less than 1 m a few years ago and today it can be up to 10 m [12]. A reader's range rests on its transmission power, the size of its antenna, the size of the antenna of the tag and its relative orientation, as well as on he presence of metal structures in its surroundings [16].

The software components are responsible for the integration of an RFID system. In a typical setting, a software front-end component manages the readers and the antennae while a middleware component routes this information to servers running the backbone database applications [15].

The Wi-Fi Method Many working environments and shopping centers provide their users with wireless high-speed internet access. The infrastructure required to support these connections is affordable, easily deployed and user oriented. The wireless infrastructure transmits at 2.4 GHz and has a reach of 50–100 m depending on the environment and the transfer rate [12].

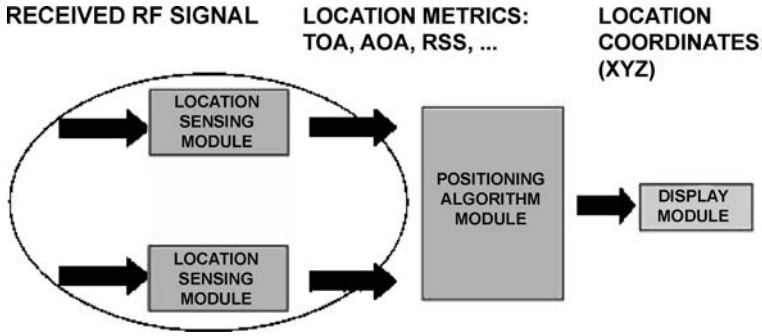
The Wi-Fi location method takes advantage of existing wireless infrastructures present in many buildings. However, in using radio-waves-based positioning, it is imperative to take into account the errors, distortions and inaccuracies tied to the propagation of the signal as obstacles are encountered and longer distances are covered. Indeed, according to experts, the propagation of the signal varies from site to site and cannot be modeled in a general fashion [13,17,18,19].

Reflection, diffraction and dispersion of radio waves are the main causes of measure distortion, signal loss and multipath effect. Movements of object, temperature variations and air displacement also account for random irregularities in the propagation of the signal causing positioning inaccuracy. The radio signal strength (RSS) metric is further affected by electronic devices operating on the WiFi band causing lost signal areas, noise and interferences [13,20]. This latter problem is important since the frequency is open to all manufacturers of microwaves, cordless phones and Bluetooth devices. Finally, the orientation of the person or object carrying the location device has an influence on the strength of the measured signals; the person or object can block, partially or entirely, the signal arriving at certain access points. For example, the human body, comprised mainly of water, absorbs waves at 2.4 GHz and has a direct impact on the reception of the Wi-Fi signal [20].

Positioning Techniques

According to Pahlavan et al. [17], the process of indoor positioning is initiated by the reception of the RF signal of a tag by sensing modules (antennae) (Fig. 2). These modules have a known and fixed location. They measure specific metrics of the tag transmission such as RSS, time of arrival (TOA) or angle of arrival (AOA) and relay the measurements to the positioning module. The measurements are then pooled and used to calculate the position of the tag. Finally, these positions are sent to be visualized by a user or can be stored in a database for archival and future analyses.

Hightower and Borriello [3,21] classify positioning techniques into four categories: geometric triangulation (including lateration and angulation methods), statistical analysis, proximity, and scene analysis.



Indoor Positioning, Figure 2 Indoor positioning process [17]. *TOA* Time of arrival, *AOA* angle of arrival, *RSS* radio signal strength, *RF* radio frequency

Geometric Technique This technique uses a geometric algorithm based on the trilateration method or the triangulation approach. In the trilateration method, a position is found at the intersection of the arcs defined around the network antennae by the distance of the detected object. The distance is computed based on the travel time of a signal from the source to the receiver (e. g., TOA/TDOA). This approach is based on a fixed and often costly infrastructure that guarantees the necessary synchronization and precision [17,22]. Distance can also be estimated with a correlation function between the signal attenuation (RSS) and the distance between the source and the receiver [22,23].
The triangulation approach is based on a known distance between two or more antennae and the corresponding angles of an incoming signal to position an object or a person. This approach requires the use of directional antennae and the knowledge of the distance between antennae on a line [22]. Measurements based on signal propagation depend upon a constant signal. Indoor environments seldom offer such stability; radio waves vary widely as they often collide with objects such as walls, furniture, equipments and other obstacles. Special infrastructures are used to circumvent this problem and obtain reliable metrics. Still, the geometric technique often yields coarse areas rather than exact point locations [17]. In such cases, the use of direct or iterative positioning algorithms by the system may help to gain better precision.

Statistical Analysis This technique, also called "empirical method," uses the "pattern matching algorithm" [13]. It relies on metrics such as RSS to create the "location fingerprint" of a tag [23]. Statistical analysis uses the RSS value directly for positioning and not as an input in a calculation as is the case with the geometric technique. According to Pahlavan et al. [17], every area in a building has a unique signature (RSS) tied to the signal propagation and location of antennae. During a preliminary benchmarking phase, the system records all the signatures of these areas in a database. These signatures are then used by different algorithms to acknowledge the position in operational mode [13]. Next, the receiver receives the signals from the antennae at a predetermined number of seconds and the information is sent back in the network. The server processes the signals in a filter that makes an average according to the delay determined (number of seconds) prior to going through the deterministic and probabilistic algorithms [14,17,18,19,22].
The advantage of this method lies in that the exact knowledge of the positions of the antennae is not required. The statistical analysis method is particularly appropriate for indoor environments as it can take into account the signal obstruction caused by different objects when calculating the location. However, the method suffers from the potentially long benchmarking process, especially for extensive indoor space coverage, which needs to be repeated every time the radio environment changes.

Proximity Technique This technique, also called "cell-ID based positioning" or "cell of origin", is based on the proximity of detectable objects to a point of reference[3,21]. It includes both indoor and outdoor positioning technologies that fall into three categories: (1) Physical contact to a sensor (e. g., pressure, capacity, movement), (2) inference from an identification system (e. g., credit card in a retail store), and (3) connection to a network access point of known radius of action (e. g., traditional cellular phone location). More specifically in the latter category, the position of the source of a signal corresponds to the zone of the receiver (antenna) coverage. Thus, the precision of a location varies according to the size of the zone covered. The size of a detection zone can be adjusted by varying the strength of the signal of the receiver. By lowering the signal strength, zones become smaller and precision increases.
The proximity technique is suitable for a large number of applications such as manufacturing, access control, healthcare management, industrial maintenance, container tracking, computer hardware management and tracking of personnel.

Indoor Positioning, Table 1 Categories of indoor LBS applications and their characteristics

Service category	Example application	Characteristics	
		Telecommunication method	Indoor positioning method and technique
Infotainment services	In a ubiquitous tourism perspective, use urban Wi-Fi or cellular network to access information concerning relevant local events in the surroundings	Wi-Fi (WiMax), wide-area cellular and AGPS	Wi-Fi, proximity technique
Tracking services	In a business-to-business perspective, place and track automatically an order to a hardware supplier, by billing and managing warehouse inventory through RFID technology	Wi-Fi, WLAN	RFID, statistical analysis
Selective information dissemination	In an hospital, allow for the transmission of the patient file on PDA (personal digital assistant) only to nurses entering the patient's room; control the access of hospital restricted areas	Bluetooth, Wi-Fi	Wi-Fi, geometric technique, scene analysis
Location-based games	With Web-based games, propose a default setup (language, country, etc.) according to the location of the gamer (IP address, cellular zone)	Cellular,GPS/UWB, IP address	Wi-Fi, proximity technique
Emergency support services	Upon a fire alert, provide automatically the detailed building map on PDA to the firefighter, indicate life-threatening situations, and track him in risk areas (e.#8239;g., toxic products)	Wi-Fi, Wide-Area Cellular, AGPS	Wi-Fi, geometric technique
Location-sensitive billing	Automatic RFID tag reading for continuous flow through highway tolls, an invoice is sent periodically by mail	RFID/Wireless (Wi-Fi) or Bluetooth	RFID, proximity technique

Scene Analysis A scene analysis system identifies and locates users inside a building using special colored badges (visual tags) and video analysis. This technique requires the use of a network of cameras equipped to detect the tags in their video streams. The location of a tag can be detected by performing a triangulation when two or more cameras detect the same tag. This is possible because: (1) the size of the badges is fixed (2) the positions of the cameras are known, and (3) the angle of the cameras is known when detecting a tag [24].

A similar approach relies on image analyses for shape recognition. This technique has been used to detect and locate pieces of equipment and people, using their physical traits such as the shape of their face or the clothes they wear, in an assembly line.

Key Applications

Indoor positioning is a special case of the application field of location-based services (LBS). It has evolved to complement GPS positioning which is almost exclusively restricted to outdoor environments.

A LBS system is generally defined as any system providing a value-added service based on the location of a mobile device in a wireless environment. It can be viewed as a collection of intelligent agents capable of initiating a communication to obtain services based on the state or activity of a user or a device. The applications are based on the integration of wireless technology, computer science, positioning technologies and spatial data [25]. LBS applications deployed and used indoors are specifically considered indoor positioning applications.

Different configurations of LBS applications are possible. Some applications send information directly to the user when the person's position coincides with some predetermined or user-defined parameters. Other applications only work when manually activated by the user. Based on this differentiation criterion, three models of LBS applications are recognized:

- Pull-type or "user-requested" applications: the user requests a service at a point in time based on the user's location.
- Push-type or "triggered" applications: a service is provided automatically based on the location of a user as soon as a set of predefined conditions are satisfied. In this case, the system collects the position of the device to provide the service.
- Tracking applications: the system provides information about a mobile device (e. g., cellular phone) upon request from a user. Services based on this model combine the "push" and "pull" approaches.

Some authors also distinguish LBS applications according to whether the services are provided to a person or a device. In the case of people, the application uses the position of a user to provide a value-added service. For devices, it is the position of the device or the equipment

that triggers the application (e. g., the arrival of merchandise in a warehouse triggers an update of the inventory database).

Jacobsen [24] classifies LBS into six categories. Table 1 presents an example of some applications, as well as their particular characteristics for each of these six categories.

Given the infancy of the field, some indoor positioning applications mentioned in Table 1 are already considered mature, whereas others still are at the forefront of research and development. Numerous applications, such as identification and tracking of individuals through face recognition for security purposes or pathway analysis of customer behavior in shopping centers, are already available today.

Many specialists recognize the great potential of application development for indoor positioning. However, the increasing number of technological opportunities makes understanding the application market an ever more complex task. As indoor positioning technologies become common place, it will be easier to capture the market and identify customers' needs as well as application niches.

Future Directions

Despite all the advances in the field, the diffusion of indoor positioning is still limited considering its potential. An increasing number of research, development, and business projects will appear in the next few years. Future works will spread beyond technological boundaries into four major and complementary research streams: technology performance and evolution, organizational benefits, individual benefits, and people and society.

Technology Performance and Evolution

Indoor positioning technologies are benefiting from the ongoing research and development efforts in the fields of telecommunications (e. g., 3G) and RF analysis. Systems will gain better accuracy and precision and increase their range of operation as new approaches and algorithms are being developed (e. g., fuzzy logic, neural networks, hidden Markov) [17]. In addition, turnkey solutions will be commercialized as standards emerge and facilitate the integration of indoor positioning applications with corporate technology infrastructures. As microgeomatics and GISciences influence each other, new dedicated analytical tools for indoor information built on geographic information systems, online analytical processing and data mining will appear.

Organizational Benefits

Large firms will become more inclined to adopt IPSs as they will provide new seamlessly integrated functionalities to their existing information system infrastructure (e. g., ERP, CRM, SCM). The convergence will give rise to novel applications that will contribute to increase organizational performance. Without a doubt, the strategic advantage of indoor positioning will decrease as its technology and applications become ubiquitous in business processes. Benefits will be most important in inventory management, merchandise tracking, collaborative work, security in organizations, asset management and supply chain optimization.

Individual Benefits

The extent of adoption of IPSs by organizations largely depends on the benefits perceived by their employees. IPSs can improve employees' work and improve well-being: context-aware information retrieval, simplified automated management of rooms access in work facilities, decision support system for public safety (e. g., children at school), life-critical reduction in time of emergency response.

People and Society

Advances in indoor positioning must be mirrored by research spanning human (e. g., privacy), ethical, legal (e. g., access to personal data) and social issues. Indeed, the control or perception of control caused by IPSs can lead to diverse reactions from uneasiness to a plain refusal to adopt the technology.

Cross References

► Statistical Descriptions of Spatial Patterns

Recommended Reading

1. Dempsey, M.: The fundamentals of indoor positioning systems in healthcare. Biomed. Instrum. Technol. 293–297 (2003), (PDF version on the web: www.radianse.com/download/newsbit-2004.pdf, July 12th, 2007)
2. Zeimpekis, V., Giaglis, G. M., Lekakos, G.: A taxonomy of indoor and outdoor positioning techniques for mobile location services. J. ACM Special Interest Group Electron. Commerce **3**, 19–27 (2003)
3. Hightower, J., Borriello, G.: Location systems for ubiquitous computing. IEEE Comput. **34**, 57–66 (2001)
4. Want, R, Hopper, A., Falcão, V., Gibbons, J.: The Active Badge Location System. ACM Trans. Inf. Syst. **10**, pp. 91–102 (1992)
5. Weiser, M.:. The computer for the 21st century. *Sci. Am.* **265**, 94–104 (1991), July 12th, 2007
6. Weiser, M.: Some computer science issues in ubiquitous computing. Commun. ACM **36**, 74–84 (1993)
7. Gow, G.A.: Privacy and ubiquitous network societies. Paper presented at the ITU workshop on ubiquitous network societies, International Telecommunication Union, April 6–8, (2005) Geneva, Switzerland
8. Beresford, A.R.: Location privacy in ubiquitous computing. In: Technical Report No. 612 UCAM-CL-TR-612, University of Cambridge, Computer Laboratory, UK (2005)

9. Srivastava, L.: Ubiquitous network societies: the case of radio frequency identification. Paper presented at: the ITU workshop on ubiquitous network societies, International Telecommunication Union, April 6–8 2005
10. Perreault L., Lapierre C., Plante M., Bachy D., Caron C.: Micro-Geomatics: a new paradigm for mobile workforce management. GeoWorld **March**, 26–28 (2006)
11. O'Connor, M.C.: RFID takes attendance—and heat. RFID J. **February 16** (2005)
12. Muthukrishnan, K., Lijding, M., Havinga, P.: (2005). Towards smart surroundings: enabling techniques and technologies for localization. In: First International Workshop on Location- and Context-Awareness (LoCA 2005), Oberpfaffenhofen 2005, Faculty of Computer Science, University of Twente, Enschede, The Netherlands
13. Guvenc, I., Abdallah, C.T., Jordan, R., Dedeoglu, O.: (2003). Enhancements to RSS based indoor tracking systems using Kalman filtersIn: Global Signal Processing Expo and International Signal Processing Conference, Dallas, Texas, March 31–April 3 2003, Guvenc
14. Melnikov, H.: Open solution for location based services in WLAN environment. Master thesis, Tampere University of Technology (2004)
15. Asif, Z.: Integrating the supply chain with RFID: A technical and business analysis. Commun. Assoc. Inf. Syst. **15**, 393–427 (2005)
16. Anne, M., Crowley, J.L., Devin, V., Privat, G.: Localisation intra-bâtiment multitechnologies: RFID, Wi-Fi, vision. In: Proceedings of Ubimob 2005, Grenoble, France, 2005, Vol.120, pp. 29—35
17. Pahlavan, K., Li, X., Makela, J-P.: Indoor geolocation science and technology. IEEE Commun. Mag. **40**, 112–118 (2002)
18. Battiti, R., Brunato, M., Villani, A.:. Statistical learning for location fingerprinting in wireless LANs. In: *Technical Report DIT-02–0086*, University of Trento, Informatic and Telecommunication Department, Italy (2002)
19. Ladd, A.M., Bekris, K.E., Rudys, A., Marceau, G., Kavraki, L.E., Wallach, D.S.: (2002). Robotics-based location sensing using wireless ethernet. In: 8th ACM MOBICOM, Atlanta, Georgia, September 23-28 2002
20. Xiang, Z., Song, S., Chen, J., Wang, H., Huang, J., Gao, X.: A wireless LAN-based indoor positioning technology. IBM J. Res. Dev. **48**, 617–626 (2004), Armonk, New York, USA
21. Hightower, J., Borriello, G.: A survey and taxonomy of location systems for ubiquitous computing. In: Technical Report UW CSE 01–08–03, University of Washington, Washington (2001)
22. Kaemarungsi, K.: Design of indoor positioning systems based on location fingerprinting technique. PhD thesis, University of Pittsburgh (2005)
23. Prasithsangaree, P., Krishnamurthy P., Chrysanthis, P.: On indoor position location with wireless LANs. Paper presented at: The 13th IEEE international symposium on personal, indoor and mobile radio communications (PIMRC 2002), Lisbon, 2002.
24. Jacobsen, H.-A. Middleware for location-based services. In: Schiller, J., Voisard, A. (eds.). Location-Based Services. Morgan Kaufmann. Elsevier, pp. 83–114 (2004)
25. Coleman, K.: Convergence: GIS/Communications/Information Technologies. In: *Direction Mag.* **13/11/2004**. Available via: www.directionsmag.com (2004)
26. Batty P. 2004: Future Trends & the Spatial Industry – Part Two. Geospatial Solutions. Advanstar Communications, New York, September, pp. 32-35

Indoor Positioning, Bayesian Methods

JUE YANG, XINRONG LI
Department of Electrical Engineering,
University of North Texas, Denton, TX, USA

Synonyms

Geolocation; Localization; Location estimation; Bayesian estimation; Mobile robotics; Location tracking

Definition

Indoor positioning, generally speaking, is the technology through which the geospatial location coordinates of a number of mobile or stationary objects are determined in indoor environments. A typical indoor positioning system usually estimates the target object's location from observation data collected by a set of sensing devices or sensors. When the target object is stationary the location estimation problem is also referred to as localization problem. On the other hand, estimating the location of mobile target objects is known as target tracking.

Bayesian estimation methods are based on the Bayes' theorem, a well-known result in probability theory, which relates the conditional and marginal probability distributions of random variables. Bayesian approaches are fundamentally different from the classical approaches such as maximum-likelihood estimator and minimum-variance unbiased estimator in that the unknown parameters are assumed to be deterministic constants in classical approaches but random variables in Bayesian approaches. By assigning a probability density function (PDF) to the unknown parameters, a Bayesian optimal estimator can always be determined, such as the minimum mean square error (MMSE) estimator that is optimal on the average.

Historical Background

The concept of geolocation is not new; it has evolved over many years through successful design and application of various legacy localization and tracking systems such as sonar, radar, and the global positioning system (GPS). In recent years, location-awareness of mobile wireless devices has attracted great interest from researchers in both the academic and industrial communities [1]. For example, Federal Communications Commission (FCC) mandated that all wireless carriers need to implement wireless E911 service for mobile subscribers in phases starting from 1996; location-awareness has also become a major research topic in pervasive computing and sensor networks communities [2]. Location information for people

or mobile devices has tremendous potential for many innovative applications in indoor environments such as shopping centers, museums, office buildings, hospitals, and prisons. Consider that in office buildings users may often need to print to the nearest printer from laptop computers or get directions to a particular office on palmtop computers; in prison or intensive healthcare facilities the location of various personnel need to be monitored and logged continuously for management, safety and/or surveillance purposes [3,4].

The well-known GPS technology has proven to be extremely valuable in many military and civilian applications. However, GPS can only work reliably where the GPS receiver has clear unobstructed line-of-sight view of at least four NAVSTAR satellites. Building walls and other objects in and around building environments can easily scatter and attenuate the radio signals employed in most geolocation systems, which significantly degrades their performance. Thus, in recent years many alternative geolocation technologies have been proposed and studied for complex indoor environments, including enhanced GPS, location fingerprinting, superresolution time of arrival (TOA), ultra-wideband (UWB), radio-frequency identification (RFID), inertial navigation and dead reckoning, wireless local area network (WLAN)-based localization, Kalman filters, particle filters, etc. [1,4,5,6]. Among the alternatives, Bayesian methods are well suited for indoor positioning, since the Bayesian methodology provides a unified framework to conveniently incorporate prior knowledge, integrate multimodal location sensors, and exploit historical data through a recursive tracking process. The Bayesian estimation methods have gone through many years of development. For example, over the past decades, Kalman filters, one of the instantiations of Bayesian methodology, have been successfully employed in many navigation and tracking systems [7]. More recently, however, with the tremendous advancement in computational power other Bayesian methods such as particle filters have become increasingly popular in the research of positioning techniques [8,9,10]. In this article, a brief description of the scientific fundamentals of Bayesian methods and the key applications of Bayesian methods are presented within the context of indoor positioning applications.

Scientific Fundamentals

Principles of Bayesian Methods

General Framework Bayesian methods provide a rigorous framework for dynamic state estimation problems [7]. They are intended to probabilistically estimate the location of target objects from noisy observations [11]. In the general framework of Bayesian methods or Bayesian filters, the location of an object is represented by a vector random variable $\boldsymbol{x}$, which is often specified by the coordinates in two- or three-dimensional Cartesian space, and sometimes by even including pitch, roll, yaw, and linear and rotational velocities, depending on specific application requirements. Observations are the measurement data collected from a wide variety of sensors that are available while the type of sensors that are employed depends on the specific application scenarios. Based on the observations collected from n sensors, namely $z = \{z_1, z_2, \ldots, z_n\}$, Bayesian filters estimate the possible positions of the target, depicted by the posterior PDF of the vector random variable $\boldsymbol{x}$ conditioned upon all of the available observation data, that is, $p(x|z)$.

By applying the Bayes' theorem, it can be readily derived that [7]

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)} = \frac{p(z|x)p(x)}{\int p(z|x)p(x)\ \mathrm{d}x}, \tag{1}$$

which describes the general framework of Bayesian filters. The distribution function $p(x)$ is usually referred to as the prior PDF since it completely characterizes the initial or prior knowledge about the unknown parameter x before any observation data has been collected. The stage at which the prior distribution is established is known as the prediction stage. On the other hand, the conditional PDF $p(z|x)$ is usually referred to as the perception model, observation model, or likelihood function, which represents the likelihood of making the observation z given that the object is at the location $\boldsymbol{x}$. Within the context of Bayesian framework given in (1), the likelihood function can be seen to utilize the measurements to refine the prior knowledge of the unknown variable $\boldsymbol{x}$ to obtain the posterior distribution of the unknown variable. Thus, the stage of deriving the posterior distribution of unknown random variable from the likelihood function and the prior distribution as in (1) is often referred to as the update stage.

Recursive Bayesian Estimation In most of the tracking applications, it is often required to continuously track the location of a target object in real-time; that is, the location estimate of the target object is continuously updated as new observation data arrive. The recursive Bayesian estimation method was developed for such scenarios following the general framework of Bayesian filtering given in (1).

More specifically, from the Bayesian perspective the purpose of tracking is to determine the posterior distribution $p(x_k|z_{1:k})$ recursively, where x_k is the state at the time k and $z_{1:k} = \{z_1, z_2, \ldots, z_k\}$ is the ensemble observation data up to the time k [9]. It is assumed that the prior distribution $p(x_0) \equiv p(x_0|z_0)$ is available, where z_0 designates the case that no observation data is available. Then, the pos-

terior distribution can be obtained recursively according to the two stages introduced above. Suppose that the posterior distribution $p(x_{k-1}|z_{k-1})$ at the time $k-1$ is available. At the prediction stage, that is, before the observation at the time k arrives, the prior PDF of the state at the time k can be obtained via the Chapman–Kolmogorov equation [9]:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})\,\mathrm{d}x_{k-1}\,, \quad (2)$$

where $p(x_k|x_{k-1})$ is the state transition model describing the system dynamics. Typically, the state transition of a target object is defined as a Markov process of order one, so that $p(x_k|x_{k-1}, z_{1:k-1}) = p(x_k|x_{k-1})$, which is utilized in deriving (2). In addition, in defining the perception model, it is usually assumed that $p(z_k|x_k, z_{1:k-1}) = p(z_k|x_k)$. Thus, at the update stage, the posterior PDF at the time k can be derived by updating the prior using the new observation data z_k as in (1),

$$\begin{aligned} p(x_k|z_{1:k}) &= \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \\ &= \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{\int p(z_k|x_k)p(x_k|z_{1:k-1})\,\mathrm{d}x_k}\,. \end{aligned} \quad (3)$$

Statistical Modeling

The general framework introduced only provides an abstract approach for probabilistic localization and tracking. In practice, implementation of Bayesian filters requires the specification of the prior PDF $p(x)$ in localization or $p(x_0)$ in tracking, the perception model $p(z|x)$ or $p(z_k|x_k)$, and the system transition model $p(x_k|x_{k-1})$ in tracking applications. In this section, a briefly discussion is presented on how these distribution functions can be determined in practice.

Prior Model The prior distribution of the state variable $p(x_0)$ is typically derived from the prior knowledge about the target state that is available. The Bayesian framework in (1) provides a convenient way to incorporate the prior knowledge. In contrast, in classical estimation methods it is difficult to make use of any prior knowledge. It is a fundamental rule of estimation theory that the use of prior knowledge will lead to a more accurate estimator, and any prior knowledge when modeled in the Bayesian sense will improve the Bayesian estimator [7]. Thus, the choice of the prior distribution is critical in Bayesian methods. Usually, Gaussian prior PDF is employed in theoretical analysis due to its mathematical tractability. However, in actual implementation of Bayesian methods, any types of prior distributions can be easily incorporated using the particle filtering techniques (discussed in "Particle Filters"), which is a versatile approximate nonlinear Bayesian filtering technique based on sequential Monte Carlo simulations. In the absence of any prior knowledge, noninformative prior distribution is used such as the uniform distribution and the noninformative Gaussian distribution [7].

System Transition Model The state transition model $p(x_k|x_{k-1})$ is inferred from the knowledge of system dynamics such as the maneuvering direction, velocity, and acceleration of the target object in tracking applications. In tracking applications, the state transition is usually modeled as a Markov process of order one, that is [9],

$$x_k = f_k(x_{k-1}, v_{k-1}) \quad (4)$$

where $f_k(\cdot)$ is a function of the state at the time $k-1$, x_{k-1}, and the random perturbation noise v_{k-1}. As an example, consider the following simple model for a maneuvering target. Suppose a vehicle is traveling through a sensor field at a constant velocity, perturbed only by slight speed corrections due to terrain change and other unexpected causes. The perturbation to the velocity can be simply modeled as additive white Gaussian noise, i. e., $v_{k-1} \sim N(\mathbf{0}, Q_{k-1})$, so that the state transition is defined as,

$$x_k = Ax_{k-1} + Bv_{k-1}\,, \quad (5)$$

where

$$x_k = \begin{bmatrix} r_k \\ u_k \end{bmatrix}, \quad A = \begin{bmatrix} I & \tau \cdot I \\ \mathbf{0} & I \end{bmatrix}, \quad B = \begin{bmatrix} \mathbf{0} \\ I \end{bmatrix}, \quad (6)$$

and r_k and u_k are the location coordinates and velocity of the target object at the time k, respectively, and τ is the time interval between samples, while I and 0 are the identity matrix and the all-zero matrix of appropriate dimensions, respectively. Then, the state transition model $p(x_k|x_{k-1})$ can be readily derived from the known statistical distribution of v_{k-1}.

The perturbation to the velocity can also be modeled with a random acceleration; that is, the random noise, $v_{k-1} \sim N(\mathbf{0}, Q_{k-1})$, represents random acceleration of the target object at the time $k-1$, caused by slight speed corrections. For this new modeling strategy, the state transition model remains the same as in (5) and (6) while the system matrices A and B take a slightly different form as compared with (6), i. e.,

$$A = \begin{bmatrix} I & \tau \cdot I \\ \mathbf{0} & I \end{bmatrix}, \quad B = \begin{bmatrix} \tau^2/2 \cdot I \\ \tau \cdot I \end{bmatrix}. \quad (7)$$

Both models have been used extensively in the literature for tracking applications.

Perception Model The well-known GPS technology has proven to be extremely valuable in many practical applications. However, GPS cannot work reliably in indoor environments due to severe attenuation and scattering effects caused by building walls and other objects in and around building environments [1]. As a result, in recent years many alternative geolocation technologies have been proposed and studied for indoor environments. Different types of location sensors employ different types of sensing modalities, including but not limited to the proximity, TOA, time difference of arrival (TDOA), angle of arrival (AOA), and received signal strength (RSS) of radio frequency (RF) (either narrowband, wideband, or UWB), infrared, and/or acoustic signals [1,4,5,6]. In addition, due to the complex nature of the indoor environments, extensive measurement and modeling efforts (sometimes known as training) are needed before system deployment to derive accurate perception model of each one of the location sensors by accounting for the complex effects of environmental characteristics.

In tracking applications, it is normally assumed that the observation data from location sensors z_k, contaminated by sensor observation noise n_k, only depend on the target's current location coordinates, i. e., the state variable x_k; that is,

$$z_k = h_k(x_k, n_k) \tag{8}$$

where $h_k(\cdot)$ is usually a nonlinear function of the state x_k and the observation noise n_k. Then, the perception model $p(z_k|x_k)$ can be derived from (8). For instance, suppose an array of acoustic sensors are used to measure the received energy of an acoustic signal originated from the target object. If the object emits the signal isotropically with strength A, according to the acoustic wave propagation theory, the measured energy at the ith acoustic sensor can be determined from

$$z_{k,i} = \frac{A}{\|x_k - r_i\|^{\beta}} + n_{k,i}\,, \tag{9}$$

where r_i and $n_{k,i}$ are the location coordinates and the observation noise of the ith sensor, respectively. The parameter β describes the signal attenuation characteristics of the medium, through which the acoustic signal propagates, and it is typically determined from an extensive measurement campaign within the intended application environments. Then, the perception model $p(z_k|x_k)$ can be readily derived from (9) with a specific assumption or knowledge of the statistical distribution of the observation noise $n_{k,i}$.

Practical Implementation of Bayesian Filters

The basic framework presented in "Statistical Modeling" provides a conceptual solution to the Bayesian estimation problems. Closed-form optimal solutions do exist in a restrictive set of cases, but in general, due to the complexity of the state transition model, perception model, and/or the prior distribution in many practical applications, the optimal solution of Bayesian methods cannot be determined analytically in the closed form. In this section, several practical implementations of Bayesian filters are briefly presented.

Kalman Filters In the derivation of Kalman filters, it is assumed that the posterior density at every time step is Gaussian and, hence, parameterized by a mean and covariance [7,9,12]. Kalman filters are the optimal estimators in the Bayesian sense when the system dynamics $f_k(x_{k-1}, v_{k-1})$ and the observation model $h_k(x_k, n_k)$ are both linear functions of the state variable x_k and both v_{k-1} and n_k are zero-mean white Gaussian noises.

In some applications, however, $f_k(x_{k-1}, v_{k-1})$ and $h_k(x_k, n_k)$ are nonlinear functions. In such cases, the extended Kalman filter (EKF) can be applied, which is based on local linearization of the nonlinear functions, typically using the first-order Taylor series expansion [7]. Since EKF is based on dynamic linearization, it has no optimality properties and its performance significantly depends on the accuracy of the linearization. In addition, in the derivation of EKF the state transition and observation models are still assumed Gaussian. Therefore, the performance of EKF also significantly depends on the suitability of the Gaussian assumption; that is, if the true densities are significantly different from Gaussian distribution, such as multimodal or heavily skewed, the approximate nonlinear EKF will result in poor performance.

Besides the EKF, some numerical integration methods have also been developed to approximately implement Kalman filters for nonlinear problems, when the noises are additive and Gaussian. The most widely used approximate nonlinear Kalman filters include the Gauss–Hermite Kalman, unscented Kalman, and Monte Carlo Kalman filters [13]. In all of these approximate nonlinear methods, integrals are approximated by discrete finite sums and direct evaluation of the Jacobian matrices is avoided.

Grid-Based Methods When the state space is discrete and consists of a finite number of states $\{x_k^i, 1 \le i \le N_S\}$ at the time k, the integrals in (2) and (3) can be simplified to summation [9]. The grid-based methods are optimal when the state space is finite. When the state space is continuous, in many applications the grid-based methods can also be used as an approximation method by piecewise discretization of the state space and the distribution functions. The grid must be sufficiently dense to achieve good approxi-

mation to the continuous state space. Thus, the grid density depends directly on the performance requirements of specific applications. For example, for many indoor positioning applications, the indoor environment of interest can be typically tessellated into uniform (or nonuniform) grid cells of about 1 m in size. Then, the posterior probabilities at the center of the cell or the average probabilities within the cells are determined. The major advantage of the approximate grid-based methods lies in the fact that arbitrary distribution functions can be approximated through discretization while a disadvantage of such methods is that predefined grids may result in inadequate resolution in high probability density regions. The computational complexity and memory space cost of the grid-based methods increases exponentially with the number of state space dimensions. As a result, the approximate grid-based methods are usually applied in low-dimensional applications, such as estimating only the target's location and orientation.

In addition, many indoor environments provide a natural way to represent a person's location at a symbolic level such as the room or hallway the person is currently in. Therefore, in some applications, if only imprecise location information is required, another type of grid-based method can be employed based on nonmetric topological representation of the physical application environments. Such a method significantly reduces the computational complexity of the approximate grid-based methods.

Particle Filters Particle filters are a set of approximate Bayesian methods based on Monte Carlo simulations. Such methods provide a convenient and attractive approach to approximate the nonlinear and non-Gaussian posterior distributions. Unlike the grid-based methods, particle filters are flexible and can conveniently overcome the difficulties encountered in dealing with multivariate, nonstandard, and multimodal distributions through the use of importance sampling techniques. The particle filters are also widely known as sequential Monte Carlo filters, the condensation algorithm, bootstrap filtering, interacting particle approximations, and survival of the fittest [8].

The key idea of particle filters is to represent the posterior PDF $p(x_k|z_{1:k})$ by a set of m weighted samples $\{x_k^i, w_k^i\}_{i=1,\ldots,m}$ [8,9]. In this representation, each ${x_k}^i$ corresponds to a possible value of the state at the time k, which can be easily drawn from an importance density $q(x_k|z_{1:k})$ and ${w_k}^i$ are non-negative numerical coefficients known as importance weights that sum up to one. Thus, the posterior density can be readily approximated by:

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{m} w_k^i \delta(x_k - x_k^i)\,, \tag{10}$$

where

$$\begin{aligned} x_k^i &\sim q(x_k|z_{1:k})\,, \\ w_k^i &\propto \frac{p(x_k^i|z_{1:k})}{q(x_k^i|z_{1:k})}\,. \end{aligned} \tag{11}$$

In tracking applications, the weights can be determined recursively through sequential importance sampling (SIS) [9], i. e.,

$$w_k^i \propto w_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, z_k)}\,, \tag{12}$$

where $q(x_k^i|x_{k-1}^i, z_k)$ is the importance density for the transition model. Thus, with a SIS algorithm, the weights and support points are recursively updated as new observation data are received sequentially.

However, as the time step k increases, the distribution of the importance weights ${w_k}^i$ becomes more and more dispersed and skewed, which is known as the degeneracy problem [8,9]. The degeneracy problem can be practically solved by introducing an additional selection step, referred to as resampling, to eliminate the particles having low importance weights and multiply particles having high importance weights. A wide variety of particles filters have been developed in recent years, such as the auxiliary and regularized particle filters. When either the transition model or the observation model is linear, the Rao–Blackwell simplifications can be applied to the particle filters to reduce computational costs. A further and detailed discussion about particle filters can be found in [8] and many references therein.

Key Applications

In recent years, Bayesian estimation methods have been successfully employed in many indoor positioning and many other related applications. In this section, a brief review of several typical applications of Bayesian filters is presented within the context of indoor positioning applications.

Perception Model-Based Localization Indoors

The Bayesian methods presented in this article are well suited for many indoor localization and tracking applications. In indoor environments, extensive measurement campaigns can be carried out before system deployment so that a wide variety of prior knowledge can be derived from the premeasurements. The Bayesian framework provides a convenient way to incorporate and exploit the prior knowledge, which constitutes a major advantage of the Bayesian methods for indoor positioning applications as compared with the classical estimation methods.

As discussed earlier, extensive measurement and modeling effort is necessary in the Bayesian methods to derive accurate prior PDF and the perception model as well as the state transition model. In this section, the focus is on Bayesian localization techniques based on homogeneous sensors; localization using decision-level sensor fusion for heterogeneous sensors is presented in the next section. Once system models are established through premeasurement data, the posterior density can be readily derived following the general framework of Bayesian methods.

In [6], a radio-frequency (RF)-based system is presented, known as RADAR, for locating and tracking users inside buildings. The basic idea underlying RADAR is based on empirical perception models of the RSS measurements of the signals received from multiple base stations. With extensive premeasurement or training data collected in the training phase, a nearest-neighbor approach is applied to determine the location estimate of the target objects. Strictly speaking, such an approach is not a probabilistic method. However, it could be regarded as a special case within the general framework of Bayes estimation methods since the premeasurement data is actually employed to develop empirical perception models of the observed RSS data. The technique presented in [14] is very similar to RADAR, but the location estimation problem is addressed in a more general setting. An improvement introduced in [14] is that multiple observation vectors are collected at each specified location to provide more robust probabilistic estimation results. In [15], a more sophisticated and detailed perception model is developed, which is based on the parametric signal propagation pathloss model. The parameters in the pathloss model are typically environment specific, and thus can only be determined from extensive measurement data through linear regression analysis.

Localization Using Decision-Level Sensor Fusion

No matter how well the perception model is refined, a positioning system that is solely based on one sensing modality can only provide a limited estimation accuracy and reliability due to hardware limitations and adverse effects of application environments. In fact, the homogeneous localization techniques presented in the previous section do not fully exploit the potential of indoor applications. Inside a building, there are usually more than one existing infrastructure networks that can implicitly provide location information. For example, the wireless LAN infrastructure can work together with camera and sound surveillance systems to improve the location estimation accuracy. A robust and scalable probabilistic positioning method is presented in [16], which employs decision-level sensor fusion, to incorporate multimodal data obtained by many different types of location sensors.

In addition, three generic architectures are presented in [12] for sensor fusion systems, which only differ in terms of at which stage the fusion occurs. Decision-level sensor fusion occurs after subsystems have processed the raw observation data collected from their sensors and reported their decisions about the target's location. Each sensor module provides the entire system with a set of possible values (object locations) represented with a probability distribution. Then, the probability distributions are combined to compute a new probability distribution that represents the most likely location of the object. More specifically, following the general framework of Bayesian methods in (1), if the observation data from n sensors are $z = \{z_1, z_2, \ldots, z_n\}$,

$$p(x|z_1, z_2, \ldots, z_n) = \frac{p(z_1, z_2, \ldots, z_n|x)p(x)}{p(z_1, z_2, \ldots, z_n)}. \tag{13}$$

Then, knowing that $\{z_1, z_2, \ldots, z_n\}$ are measurement data from statistically independent sensors, it can be easily derived that:

$$p(x|z_1, z_2, \ldots, z_n) = \frac{p(z_1|x)p(z_2|x)\ldots p(z_n|x)p(x)}{p(z_1, z_2, \ldots, z_n)}. \tag{14}$$

It should be noted that the number of measurement data n is not a constant number; instead, its value varies from time to time, depending on the number of available sensors that can provide decision for the current estimation. This approach thus facilitates modular and extensible system architecture. There is no limit on the number and the types of sensors that can be employed using this method. With decision-level sensor fusion, when some of the sensors fail (or are shut off due to power saving efforts), the quality of localization is affected, but the system as a whole will continue remain functional.

Robot Tracking Using Particle Filters

The examples presented in the previous sections do not fully exploit the full strength of the Bayesian methods. The particle filters as well as other sequential Bayesian methods are best suited for location tracking problems as in many mobile robotics applications [8].

Location tracking is the most fundamental problem in many practical mobile robotics applications, and thus it has received tremendous attention in recent years. In such problems, the initial robot's location is represented by the

initial prior distribution $p(x_0)$ and the location is updated by tracking techniques based on sequentially collected sensor measurements. In order to implement the particle filters, three probability distributions need to be determined first, i. e., the prior PDF $p(x_0)$, the transition model or the motion model $p(x_k|x_{k-1}, U_{k-1})$ where U_{k-1} denotes control data or odometer data, and the perception model or the likelihood function $p(z_k|x_k)$. Then, the particle filters or other recursive Bayesian estimation methods that are presented previously in this article can be applied in a straightforward way. Detailed convergence property results of the particle filters and many improvement techniques can be found in [8].

Data Association for Multiple Target Tracking

Under the most general setup of a target tracking system, which is common in many applications, a varying number of moving targets need to be tracked continuously in a region from noisy observation data that are sampled at random intervals. Such a problem is widely known as multitarget tracking (MTT) [17,18,19,20]. In MTT, the association between the observation data and the targets are typically unknown so that MTT is much more complex than the single–target tracking problem. Existing MTT algorithms generally present two basic ingredients: an estimation algorithm and a data association method [20]. To address the data association problem in MTT, some assumptions are commonly made, for example, one measurement can originate from one target or from the clutter, one target can produce zero or one measurement at a time, and one target can produce zero or several measurements at one time. Over the years, many MTT methods have been proposed in the literature based on some or all of the preceding assumptions and they vary largely on the association methods that are employed.

The most successful multiple target tracking algorithm is the multiple hypothesis tracker (MHT) [17]. With MHT, each hypothesis associates past observations with a target and, as a new set of observation data arrives, a new set of hypotheses is formed from the previous hypotheses. Each hypothesis is scored by its posterior and the algorithm returns the hypothesis with the highest posterior as a solution. A different approach to the data association problem is the joint probabilistic data association filter (JPDAF) [18], which is a suboptimal single-stage approximation to the optimal Bayesian filter. The JPDAF assumes a fixed number of targets and is a sequential tracker in which the associations between the "known" targets and the latest observation data are made sequentially. In [19], the authors presented a Markov chain Monte Carlo data association (MCMCDA) method, which shares the same ideas of particle filtering. Like other particle filtering solution, MCMCDA requires relatively low memory consumption and outperforms other algorithms in extreme conditions where there are a large number of targets, false alarms, and missing observations. In the context of MTT, particle filters are also appealing, since as the association needs to be considered only at a given time iteration, the complexity of data association can be reduced [20].

Future Directions

From the examples discussed above, it is clear that in general the probabilistic Bayesian estimation methods have tremendous potential for indoor positioning systems. In particular, the particle filtering is a versatile approximate nonlinear Bayesian filtering technique based on sequential Monte Carlo simulations and it offers a simple yet powerful implementation of Bayesian methodology for sequential inference in nonlinear, non-Gaussian systems. However, in order to employ the Bayesian methods in complex indoor environments for localization and tracking applications, accurate system models need to be developed for a wide variety of location sensors that are suitable for indoor applications. On the other hand, due to the intractability of nonlinear, non-Gaussian problems, the performance of various methods designed for such problems are typically evaluated through computer simulations. However, the performance of a geolocation system is heavily affected by relative geometric structures formed by reference nodes and target nodes (a.k.a. geometric conditioning, characterized by geometric dilution of precision (GDOP)) as well as the severe adverse multipath and non-line-of-sight (NLOS) effects in indoor environments. Therefore, accurate models of indoor radio propagation channels and other indoor environmental characteristics need to be developed to enable convenient performance study of indoor positioning systems through simulations. In addition, more theoretical developments are necessary in both algorithm design and theoretical performance bounds study for the Bayesian methods for various specific indoor geolocation applications.

Cross References

► Indoor Positioning

Recommended Reading

1. Pahlavan, K., Li, X., Makela, J.: Indoor geolocation science and technology. IEEE Commun. Mag. **40**(2), 112–118 (2002)
2. Hightower, J., Borriello, G.: Location Systems for ubiquitous computing. Computer **34**(8), 57–66 (2001)
3. Christ, T.W., Godwin, P.A., Lavigne, R.E.: A prison guard duress alarm location system. In: Proceedings of the IEEE Interna-

tional Carnahan Conference on Security Technology, vol.1, pp. 106–116, Ottawa, Ont., Canada (1993)
4. Werb, J., Lanzl, C.: Designing a positioning system for finding things and people indoors. IEEE Spectrum **35**, 71–78 (1998)
5. Ladd, A.M., Bekris, K.E., Marceau, G., Rudys, A., Wallach, D.S., Kavraki, L.E.: Using Wireless Ethernet for Localization. IROS, Lausanne (2002)
6. Bahl, P., Padmanabhan, V.N.: RADAR: An in-building RF-based user location and tracking system. IEEE Infocom **2**, 775–784 (2000)
7. Kay, S.M.: Fundamentals of Statistical Signal Processing: Estimation Theory. Prentice Hall, Upper Saddle River, NY (1993)
8. Doucet, A., de Freitas, N., Gordon, N. (eds.): Sequential Monte Carlo in Practice. Springer-Verlag, New York, NY (2001)
9. Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Trans. Signal Process. **50**, 174–188 (2002)
10. Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U.: Particle filters for positioning, navigation and tracking. IEEE Trans. Signal Process. **50**, 425–437 (2002)
11. Fox, D., Hightower, J., Kauz, H., Liao, L., Patterson, D.: Bayesian Techniques for Location Estimation. In: Proceedings of the Workshop on Location-Aware Computing (UBICOMP Conference), Seattle, WA, October 2003
12. Hall, D.L.: Mathematical Techniques in Multisensor Data Fusion. Artech, Boston, MA (1992)
13. Cui, N., Hong, L., Layne, J.R. A comparison of nonlinear filtering approaches with an application to ground target tracking. Signal Process. **85**, 1469–1492 (2005)
14. Roos, T., Myllymaki, P., Tirri, H., Misikangas, P., Sievanen, J.: A Probabilistic Approach to WLAN User Location Estimation. Int. J. Wirel. Inf. Netw. **9**(3), 155–164 (2002)
15. Roos, T., Myllymäki, P., Tirri, H.: A statistical modeling approach to location estimation. IEEE Trans. Mobile Comput. **1**, 59–69 (2002)
16. Bohn, J., Vogt, H.: Robust Probabilistic Positioning Based on High-Level Sensor Fusion and Map Knowledge. Tech. Rep. 421. Institute for Pervasive Computing, Distributed Systems Group, Swiss Federal Institute of Technology (ETH), Zurich (2003)
17. Reid, D.B.: An algorithm for tracking multiple targets. IEEE Trans. Automat. Control **24**, 843–854 (1979)
18. Bar-Shalom, Y., Fortmann, T.E.: Tracking and data association. Mathematics in Science and Engineering Series 179. Academic, San Diego (1988)
19. Oh, S., Russell, S., Sastry, S.: Markov chain Monte Carlo data association for general multiple-target tracking problems. In: Proceedings of the 43rd IEEE Conference on Decision and Control, Paradise Island, Bahamas (2004)
20. Hue, C., Le Cadre, J.-P., Perez, P.: Posterior Cramer-Rao bounds for multi-target tracking. IEEE Trans. Aerospace Electron. Syst. **42**(1), 37–49 (2006)
21. Ramadurai, V., Sichitiu, M.L.: Localization in wireless sensor networks: a probabilistic approach. Proc. of the 2003 International Conference on Wireless Networks (ICWN2003), vol.1, pp. 275–281, Las Vegas, NV, Jan. (2003)

Indoor Positioning System

► Indoor Positioning

Indoor Positioning with Wireless Local Area Networks (WLAN)

A. KUSHKI, K.N. PLATANIOTIS, A.N. VENETSANOPOULOS
Multimedia Laboratory, University of Toronto, Toronto, ONT, Canada

Synonyms

WLAN localization; WLAN location estimation; WLAN location determination; WLAN geolocation; WLAN location identification; WLAN location discovery; Radiolocation; WLAN location sensing; Position location; Location based services

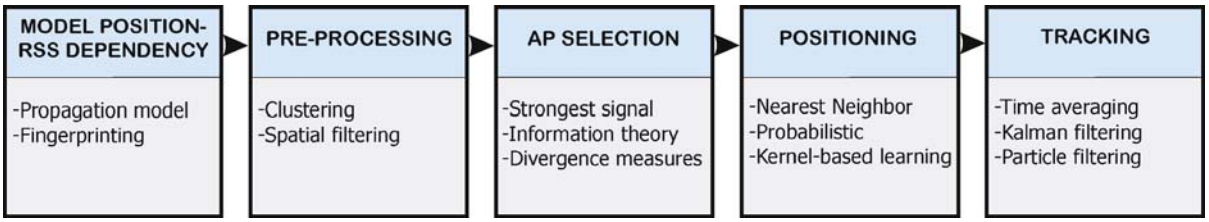
Definition

WLAN positioning refers to the process of locating mobile network devices, such as laptops or personal digital assistants, using a Wireless Local Area Network (WLAN) infrastructure. Positioning is carried out by exploiting the dependency between the location of a mobile device (MD) and characteristics of signals transmitted between the MD and a set of WLAN access points (APs). These characteristics are generally Time of Arrival (ToA), Time Difference of Arrival (TDoA), Angle of Arrival (AoA), and Received Signal Strength (RSS). RSS is the feature of choice in WLAN positioning systems as it can be obtained directly from Network Interface Cards (NIC) that are available on most handheld computers. This allows the implementation of positioning algorithms on top of existing WLAN infrastructures without the need for any additional hardware. The wide availability and ubiquitous coverage provided by WLANs makes this type of positioning a particularly cost effective solution for offering value-added location based services (LBS) in commercial and residential indoor environments.

With accuracies in the range of 1–5 meters, WLAN positioning systems can be used to infer location information in two or three dimensional Cartesian coordinates with the third dimension representing a floor number. Location information can either be reported relative to a predefined coordinate system or symbolically (e. g., room number, floor number, etc.). Positioning may be performed by the infrastructure (network-based, remote positioning) or by the MD (terminal-based, self-positioning). The former offers more computational power whereas terminal-based positioning enhances scalability and promotes privacy.

Historical Background

Cellular network infrastructures have served as a milieu for the development of positioning systems to deliver location-based emergency and commercial services, including loca-



Indoor Positioning with Wireless Local Area Networks (WLAN), Figure 1 Overview of steps involved in positioning

tion-sensitive billing and advertising, to their users since the introduction of the E-911 mandate by the U.S. Federal Communications Commission in 1996. Another important positioning system is the Global Positioning System (GPS), which offers highly accurate positioning capability and is used in outdoor navigation and LBS.

More recently, advances in wireless communication technology have led to user mobility within indoor networks, inspiring location-awareness and LBS in a wide range of personal and commercial applications geared toward indoor environments. Unfortunately, the positioning accuracy provided by existing cellular-based methods is not sufficient for such applications and coverage of the GPS system is limited in indoor environments.

With this in mind, various positioning systems have been proposed to address the problem of positioning specifically in indoor environments. These systems make use of a variety of technologies including proximity sensors, radio frequency (RF) and ultrasound badges, visual sensors, and WLAN radio signals, to carry out positioning.

Among the above methods, WLAN positioning systems have received special attention due to the fact that they can be cost-effectively implemented on top of existing and widely deployed WLANs. Similar to "cell of origin" methods in cellular systems, early work in WLAN positioning estimated the position of an MD as that of the access point with the strongest signal based on the premise that this AP is closest to the MD in the physical space. With the advent of the IEEE 802.11 based wireless networks, the first fine-grained WLAN positioning systems were introduced in the year 2000 (see, for example, the pioneering RADAR system [1]). Soon thereafter, WLAN positioning solutions were commercialized for use in applications such as asset tracking, resource management, and network security [2].

Scientific Fundamentals

The main technical challenge in WLAN positioning is the determination of the dependency between the received signal strength (RSS) and the location of an MD. As the signal travels through an ideal propagation medium (i. e., free-space), the received signal power falls off inversely proportional to the square of the distance between the receiver and transmitter. Thus, given the measurements of transmitted and received powers, the distance between the transmitter and the MD can be determined. Given three such distances, trilateration can be used to estimate the position of an MD. Unfortunately, in real environments, the propagation channel is much more complicated than the ideal scenario presented above, leading to a much more complex position-RSS dependency. The difficulties arise as a result of severe multipath and shadowing conditions as well as non-line-of-sight propagation (NLOS) caused by the presence of walls, humans, and other rigid objects. Moreover, the IEEE 802.11 WLAN operates on the license-free band of frequency 2.4 GHz which is the same as cordless phones, microwaves, BlueTooth devices, and the resonance frequency of water. This leads to time-varying interference from such devices and signal absorption by the human body, further complicating the propagation environment. To make matters worse, WLAN infrastructures are highly dynamic as access points can easily be moved or discarded, in contrast to their base-station counterparts in cellular systems which generally remain intact for long periods of time.

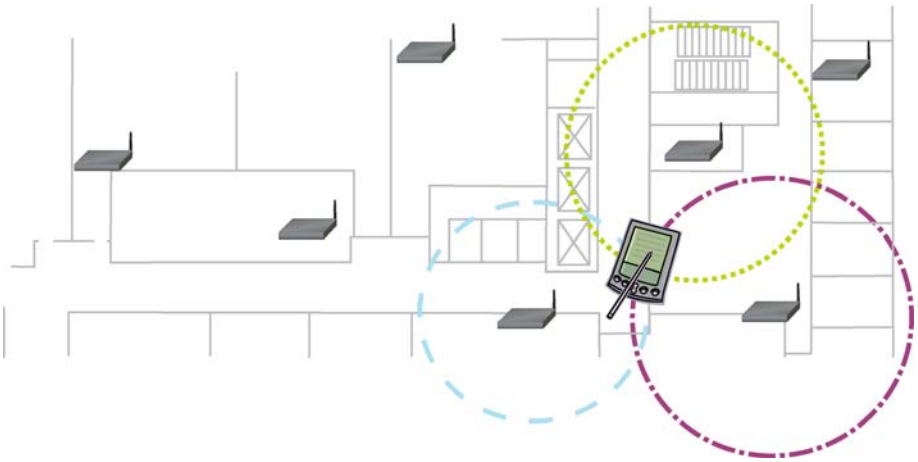
Figure 1 provides an outline of the steps involved in a positioning system. Each of the components are described in sections that follow.

Modeling of Rss-Position Dependency

Existing WLAN positioning techniques can be classified into model-based and fingerprinting approaches based on how the RSS-position dependency is determined. The approaches are discussed next.

Model-Based Methods The first class of methods aim to characterize the RSS-position dependency using theoretical models whose parameters are estimated based on training data [3]. Given an RSS measurement and this model, the distances from the MD to at least three APs are determined and trilateration is used to obtain the MD position as shown in Fig. 2.

A model for relating RSS and the distance to an AP can be constructed by considering the fact that in real envi-



Indoor Positioning with Wireless Local Area Networks (WLAN), Figure 2 Model-based WLAN positioning techniques use a propagation model to obtain distances to at least three APs and trilaterate the position of the mobile device

ronments, in addition to the distance traveled, two additional mechanisms contribute to variations in the propagation channel, namely, large scale and small scale fading. Large scale fading is due to path loss and shadowing effects. Path loss is related to dissipation of signal power over distances of 100–1000 meters. Shadowing is a result of reflection, absorption, and scattering caused by obstacles between the transmitter and receiver and occurs over distances proportional to the size of the objects [4]. Due to uncertainties in the nature and location of the blocking objects, the effects of shadowing are often characterized statistically. Specifically, a log-normal distribution is generally assumed for the ratio of transmit-to-receive power. The combined effects of path loss and shadowing can be expressed by the simplified model below [4]:

$$P_{\mathrm{r}}(\mathrm{dB}) = P_{\mathrm{t}}(\mathrm{dB}) + 10\log_{10} K - 10\gamma \log_{10}\left(\frac{d}{d_0}\right) - \psi(\mathrm{dB}). \quad (1)$$

In Eq. (1), P_{r} and P_{t} are the received and transmitted powers respectively, K is a constant relating to antenna and channel characteristics, d_0 is a reference distance for antenna far-field, and γ is the path loss exponent. Typical values of this parameter are $\gamma = 2$ for free-space and $2 \leq \gamma \leq 6$ for an office building with multiple floors. Finally, $\psi \sim \mathcal{N}(0, \sigma_\psi^2)$ reflects the effects of log-normal shadowing in the model.

Small scale fading is due to constructive and destructive addition from multiple signal paths (multipath) and happens over distances on the order of the carrier wavelength (for WLANs, $\lambda = \frac{c}{2.4\,\mathrm{GHz}} = 12.5\,\mathrm{cm}$). Small-scale fading effects can lead to Rayleigh or Rician distributions depending on the presence or absence of a line-of-sight, respectively [4].

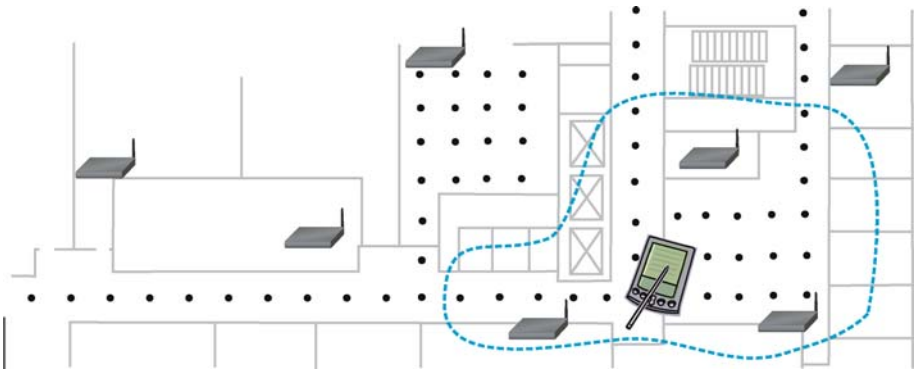
In indoor areas, materials for walls and floors, number of floors, layout of rooms, location of obstructing objects, and the size of each room have a significant effect on path loss. This makes it difficult to find a model applicable to general environments. Other limitations of model-based approaches include their dependence on prior topological information, assumption of isotropic RSS contours and invariance to receiver orientation [5].

Fingerprinting-Based Methods As an alternative to model-based methods, the RSS-position dependency can be characterized implicitly using a training-based method known as *location fingerprinting*. A location fingerprint is a vector $\mathbf{r}_i(\mathrm{t}) = [r_i{}^1(\mathrm{t}), \ldots r_i{}^L(\mathrm{t})]$ of RSS measurements from L APs at time t at spatial point $\mathbf{p}_i$. The symbol $\mathbf{p}_i = (x_i, y_i)$ or $\mathbf{p}_i = (x_i, y_i, z_i)$ represents a point in the two or three dimensional Cartesian coordinates.

Fingerprints are usually generated offline. This is done by collecting a set of measurements $\mathbf{r}_i(\mathrm{t})$, $t = 1, \ldots, n_i$, at a set of training locations $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ with the purpose of obtaining a sufficient representation of spatio-temporal RSS properties in the given environment. The database of these location fingerprints together with their respective coordinates is known as a *radio map*. The main challenges in the construction of the radio map include the placement and number of survey points as well as determination of the number of time samples needed for a sufficient representation at each point.

Preprocessing

Before the actual positioning is performed, some preprocessing may happen to reduce computational complexity and improve efficiency of the positioning algorithm. For example, [6] proposes two preprocessing methods. The first clusters the environment into grids that receive similar AP coverage and reduces the search space to a single cluster. The second involves an incremental trilateration



Indoor Positioning with Wireless Local Area Networks (WLAN), Figure 3
Fingerprinting-based WLAN positioning techniques estimate the position of the mobile device as a combination of training points whose fingerprint record best matches the observation RSS. The dashed line delineates the training points used to form the estimate

technique where APs are used consecutively to reduce the subset of candidate locations. With the objective of power efficiency in mind, the work of [7] proposes an offline clustering method based on K-means that considers the similarity of AP values in addition to the covering sets. Lastly, [8] proposes an online spatial filtering technique to dynamically exclude irrelevant survey points from positioning calculations. The online preprocessing techniques are advantageous to their offline counterparts in terms of resiliency to loss of APs as the set of APs used during the real operation of the system may be different than that used during training.

AP Selection

Although two-dimensional positioning can be carried out with as few as three APs, a mobile device may receive coverage from many more APs in large indoor environments with ubiquitous WLAN infrastructures. Clearly, using all available APs for positioning increases the computational complexity of the system. Moreover, depending on the relative distance of the MD and each AP and the topology of the environment in terms of obstacles causing NLOS propagation, correlated RSS readings may be received from subsets of APs, leading to biased estimates.

These problems motivate the design of an AP selection block whose function is to choose the best set of APs with the given minimum cardinality for positioning. The most commonly used selection scheme is to choose APs with the highest observation RSS to ensure coverage for survey points near the observation. Unfortunately, the time variance in RSS from an AP generally increases with its mean signal strength. In such cases, the observation may differ significantly from the training values and it becomes more difficult to distinguish neighboring points.

More recently, AP selection methods employing entropy-based techniques and divergence measures have been proposed in [7,8]. In particular, the work of [7] proposes the use of the Information Gain criterion to choose APs with the best discrimination ability across the survey points and experimentally demonstrates advantages over the traditional technique of the strongest APs. Because selection is performed during the offline training of the system and remains fixed for predetermined clusters of points, the method lacks flexibility in coping with loss of APs. An online and realtime selection technique is proposed in [8] to circumvent this problem. This method aims to select a set of APs with minimum correlation to reduce redundancy. Lastly, since distance calculations are performed on the AP set chosen by the selection component, it is important to consider the interplay among the selection strategy and distance measurement when designing these components [8].

Positioning

With reference to Fig. 1, the positioning step is initiated when a new RSS measurement from an MD is received during the online operation of the system. As previously mentioned, model-based techniques rely on trilateration for positioning. In contrast, fingerprinting-based methods compare the observation to fingerprints in the radio map and return a position estimate. In general, the estimate is a combination of survey points whose fingerprints most closely match the observation.

As shown in Fig. 3, the aim of positioning is to determine a position estimate as a function of the available survey points. That is, the goal is to find $\hat{\mathbf{p}} = h(\mathbf{p}_1, \ldots, \mathbf{p}_N)$ where $\hat{\mathbf{p}}$ denotes the position estimate and $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ is the set of survey points in the radio map. If $h(\cdot)$ is restricted to be a linear function of the survey points, the position estimate $\hat{\mathbf{p}}$ can be obtained as follows:

$$\hat{\mathbf{p}} = \frac{1}{K} \frac{\sum_{i=1}^{K} w_i \mathbf{p}_{(i)}}{\sum_{K=1}^{N} w_i}, \tag{2}$$

where w_i is inversely proportional to the distance between the fingerprints at $\mathbf{p}_{(i)}$ and the observation and the set

$\{\mathbf{p}_{(1)}, \ldots, \mathbf{p}_{(K)}\}$ denotes the ordering of survey points with respect to w_i.

There are three groups of positioning techniques in the existing literature, namely, deterministic, probabilistic, and kernel-based learning methods, based on how they determine the weights w_i. These are discussed in what follows.

In the simplest case, the survey points are ranked based on the Euclidean distance between the observation and the sample mean of the RSS training samples as in RADAR [1]. In this case, the position estimate is obtained as the average of the K nearest neighbors (KNN) and $w_1 = w_2 = \ldots = w_K$.

In the more general case, the weights w_i are determined as functions of the distance between the observation RSS and the training RSS record at each survey point. For example, the weights can be chosen to be inversely proportional to the Euclidean distance above. Despite its simplicity, however, the Euclidean distance may fail to deliver adequate performance in cases where the distribution of RSS training vectors included in the fingerprints are non-convex and multimodal. Such distributions arise frequently in indoor WLAN settings due to NLOS propagation and presence of users [5].

In Bayesian approaches, such as [6], the weights are directly proportional to the likelihood or posterior probabilities $p(\mathbf{r}|\mathbf{p}_i)$ and $p(\mathbf{p}_i|\mathbf{r})$. These probabilities can be estimated from the training data either parametrically, through the assumption of a specific form for the density (e. g., a Gaussian), or nonparametrically, using density estimates such as the histogram or the kernel density estimator (KDE) [2,9]. Using the probabilistic weights, the position estimate corresponds to the maximum likelihood (ML) or maximum a posteriori (MAP) estimate when $K = 1$, and the minimum mean square error estimate (MMSE) when $K = N$ [9].

Motivated by the complexity of RSS patterns in this Euclidean space, the work of [8] proposes a kernelized distance for the calculation of the distance between an RSS observation and the fingerprints. This method nonlinearly maps the original fingerprint data to a high dimensional feature space where the distribution of RSS training vectors is simplified and carries out distance calculations in such a space. The weights are then obtained as inner products in the kernel-defined feature space. The authors show that this position estimator can be interpreted as a multidimensional kernel regression as well as the MMSE estimator of position in the case where the empirical probability density estimate (epdf) is used as the prior $f(\mathbf{p})$.

Another kernel method explored in the context of WLAN positioning is the kernel support vector machine (SVM) for both classification and regression [10], aiming to find the best set of weights for interpolating the survey points to minimize the training error while controlling the model complexity. Both of the classification and regression problems are solved independently for each dimension in the physical space. In contrast to this, the work of [11] proposes a multidimensional vector regression. In particular, a nonlinear mapping between the signal and physical spaces is built by observing that the pairwise similarity in these two spaces should be consistent. The distance between an observation and the fingerprint record is obtained as the distance between their projections onto a set of canonical vectors determined through Kernel Canonical Correlation Analysis (KCCA) and used to determine the weights in (2).

As an alternative to the use of Equation (2), decision trees are used in [7] to determine the position estimate by classifying the incoming observation as coming from one of the survey points by a series of efficient tests. Such a scheme is effective in reducing complexity and, hence, improves power efficiency on mobile devices.

An important consideration in designing training-based methods, such as the aforementioned regression techniques, is resiliency to loss of APs in WLAN infrastructures. Because APs can easily be discarded or moved, the dimensionality of the RSS vector may well be different in the training and realtime operation of the system and cannot be assumed to be fixed [8].

Tracking

Human motion is generally not random but correlated over time. Therefore, at any point in time, past position estimates can be utilized to adjust the current estimate. Such dynamic tracking solutions may simply entail a running average of previous estimates [6] or rely on more sophisticated methods such as the Kalman filter [9], Markov-model based solutions [12], and the particle filter [13].

Tracking need not be limited to the use of past position information. Predictions of future locations can also be used to proactively adjust system parameters. For example, in [9], predictions are utilized to dynamically generate subsets of the radio map for use in positioning.

Key Applications

Since its conception, WLAN positioning has been used as an enabling tool for offering location-based services in indoor environments. Three examples of specific applications are outlined below.

Network Management & Security WLAN positioning solutions can be used to offer a variety of location sensitive network services such as resource allocation, traffic management, and asset tracking. In addition, such technology can promote network security by

introducing location-based authorization and authentication. Many such solutions are currently commercially available [2].

Information Delivery In order to provide mobile users with seamless and transparent access to information content, location-based personalization of information delivery is needed [14]. WLAN positioning is a particularly effective solution in such cases since the necessary infrastructure is widely available in both commercial and home settings.

Context-Awareness As location is an important piece of contextual information, WLAN positioning and tracking can offer an effective solution for context-aware applications that not only respond, but proactively anticipate user needs [15].

Future Directions

Wireless local area networks provide a cost-effective infrastructure for the implementation of positioning systems in indoor environments. As large urban cities turn into giant wireless "hotspots", it becomes essential to consider the design of inexpensive outdoor WLAN positioning methods. Such solutions must be distributed and scalable to support entire cities and power-efficient to allow implementation on a variety of mobile devices with different capabilities. Lastly, location information must be secured to prevent unauthorized usage.

Cross References

► Indoor Positioning
► Privacy Preservation of GPS Traces

Recommended Reading

1. Bahl, P., Padmanabhan, V.: RADAR: an in-building RF-based user location and tracking system. In: Proceedings of IEEE Infocom, Tel Aviv, Israel, vol. 2, pp. 775–784. IEEE, Piscataway, NJ (2000)
2. EKahau: [Online]. Available: http://www.ekahau.com/ (2006)
3. Singh, R., Macchi, L., Regazzoni, C., Plataniotis, K.: A statistical modelling based location determination method using fusion in WLAN. In: Proceedings of the International Workshop on Wireless Ad-hoc Networks, London, UK. (2005)
4. Goldsmith, A.: Wireless Communications. Cambridge University Press (2005)
5. Kaemarungsi, K., Krishnamurthy, P.: Properties of indoor received signal strength for WLAN location fingerprinting. In: Proceedings of the The First Annual Int. Conf. on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS), Boston, USA, pp. 14–23. IEEE, Piscataway, NJ (2004)
6. Yousief, M.: Horus: A WLAN-based indoor location determination system, Ph.D. dissertation, University of Maryland (2004)
7. Chen, Y., Yang, Q., Yin, J., Chai, X.: Power-efficient access-point selection for indoor location estimation. IEEE Trans. on Knowledge and Data Engineering **18**(7), 877–888 (2006)
8. Kushki, A., Plataniotis, K., Venetsanopoulos, A.: Kernel-based positioning in wireless local area networks. IEEE Trans. on Mobile Computing **6**(6), 689–705 (2007)
9. Kushki, A., Plataniotis, K., Venetsanopoulos, A.N.: Location tracking in wireless local area networks with adaptive radio maps. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Toulouse, France, vol. 5, pp. 741–744. IEEE, Piscataway, NJ (2006)
10. Brunato, M., Battiti, R.: Statistical learning theory for location finngerprinting in wireless lans. Computer Networks **47**(6), 825–845 (2005)
11. Pan, J., Kwok, J., Yang, Q., Chen, Y.: Multidimensional vector regression for accurate and low-cost location estimation in pervasive computing. IEEE Trans. on Knowledge and Data Engineering **18**(9), 1181–1193 (2006)
12. Haeberlen, A., Flannery, E., Ladd, A.M., Rudys, A., Wallach, D.S., Kavraki, L.E.: Practical robust localization over large-scale 802.11 wireless networks: In: Proceedings of the Tenth ACM Int. Conf. on Mobile Computing and Networking (MOBICOM), Philadelphia, PA, pp. 70–84 (2004)
13. Evennou, F., Marx, F., Novakov, E.: Map-aided indoor mobile positioning system using particlefilter. In: Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), New Orleans, USA, vol. 4, pp. 2490–2494. IEEE, Piscataway, NJ (2005)
14. Harroud, H., Ahmed, M., Karmouch, A.: Policy-driven personalized multimedia services for mobile users. IEEE Trans. on Mobile Computing **2**(1) 16–24 (2003)
15. Patterson, C., Muntz, R., Pancake, C.: Challenges in location-aware computing. IEEE Pervasive Computing **2**(2) 80–89 (2003)

Influence Diagrams

► Bayesian Network Integration with GIS

Influence Time

► Queries in Spatio-temporal Databases, Time Parameterized

Information Fusion

► Computing Fitness of Use of Geospatial Datasets

Information Integration

► Ontology-Based Geospatial Data Integration

Information Management Systems

► Pandemics, Detection and Management

Information Presentation, Dynamic

ZHEN WEN
IBM, T. J. Watson Research Center,
Hawthorne, NY, USA

Synonyms

Context-sensitive visualization; Context-aware presentation; Automatic graphics generation

Definition

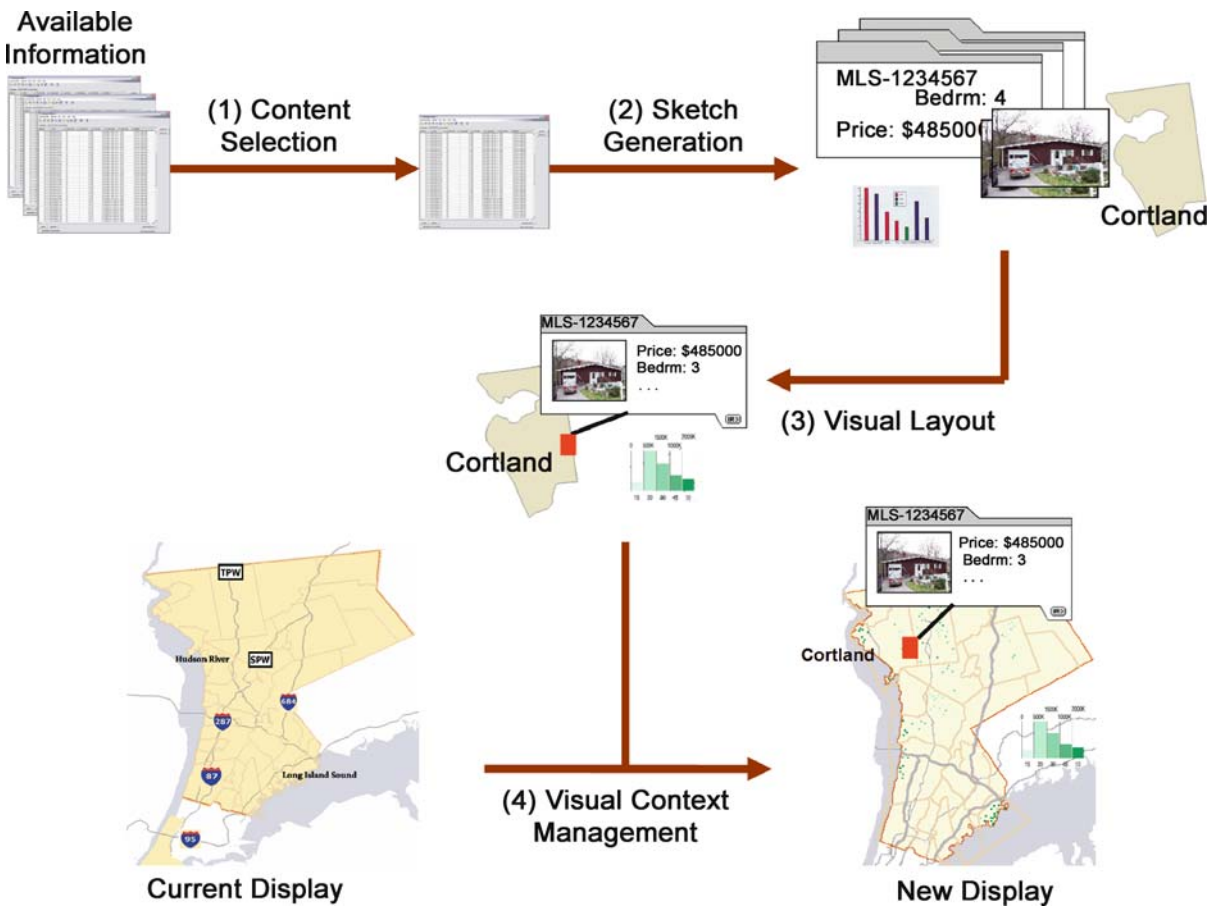
Many existing and emergent applications collect data containing geographic location. A large amount of such information from diverse sources is usually involved in investigative tasks (e. g., anti-terrorist intelligence analysis and business intelligence). Geographic information plays an important role in connecting scattered information in investigations. However, investigative tasks are highly dynamic processes where information changes can not be predicted. Therefore, traditional static geographic information presentation approaches are not sufficient for coherently presenting unanticipated information. To address this challenge, dynamic presentation approaches are proposed. Based on investigation contexts, the dynamic approaches use various metrics to dynamically measure the desirability of the information, the corresponding visualization and the visual scene updates. Then, appropriate information contents are selected and dynamically incorporated into existing visual display.

Historical Background

Investigative tasks, such as anti-terrorist intelligence analysis and business intelligence, require that analysts effectively distill a large amount of scattered, diverse information. Much of today's information collected by many existing and emergent applications contains geo-spatial locations. For example, most electronic transactions of daily life, such as purchasing goods by credit card or making phone calls, usually record geographic locations together with other attributes. For instance, telephone call records also include locations of communication endpoints, billing accounts, and sometimes cell phone zones. Therefore, geographic information can bridge spatially related information together into a coherent presentation which helps analysts understand and connect scattered information. For example, geographic information is essential for investigation of bio-terrorism [16] and military intelligence [9].
Traditionally, studies on geographic information presentation are mostly focused on creating maps for a pre-defined set of spatial information (e. g., cities and roads) [5]. In investigative tasks, however, analysts always need to interact with a highly dynamic set of geography related information because new information can keep coming in and may not be anticipated. For example, in a business intelligence application, analysts use geography related information to evaluate a specific real estate market. Analysts may start with an existing map to investigate the area. As the investigation unfolds, analysts may need to investigate various aspects (e. g., school districts, local business, transportation and etc.). Overlaying such related information on top of the map could assist analysts by using spatial cues to connect scattered information. Apparently it is not feasible to overlay all relevant information on a map since the information space is huge. Instead, a subset of information relevant to a current investigation focus should be presented on maps. Traditional geographic information presentation approaches [5,8] are insufficient for such purposes because three challenging issues are not addressed: (1) how to dynamically select un-anticipated information to be presented on maps; (2) how to dynamically layout unanticipated geographic information; and (3) how to transform maps as information changes to preserve an analysts' cognitive map. To address these challenges, dynamic geographic information presentation systems are demanded. Compared to traditional techniques, the dynamic presentation approaches offer two key advantages: (1) The dynamic approaches adapt the geographic information presentation as the investigation unfolds. Thus, analysts can be relieved from the burden of manually changing presentation configurations to view different spatial information during lengthy investigations; (2) The dynamic approaches ensure smooth transitions between maps containing different sets of information. Thus, analysts can better maintain their cognitive map of relevant information during investigations.

Related Work

Research in cartography [5] studied the principles of how to visualize geographic information. Cognitive scientists have also studied factors in order to facilitate information comprehension across multiple displays (e. g., maps) [12]. While these works provide empirical theories to draw upon, computational models are needed to realize these theories. In recent years, research has been initiated in the direction of supporting dynamic interaction scenarios such as those in investigative tasks.
Dynamic geographic information presentation is closely related to automated multimedia presentation systems [4,6]. These works automatically derive multimedia presentation based on user or task models. Since most of these systems support limited user interaction (e. g., [6]



Information Presentation, Dynamic, Figure 1 A pipeline for dynamic geographical information presentation

uses a predesigned menu), they often use a rule-based or schema-based approach to select content. In contrast, investigative tasks involve large and complex data sets, where developing an exhaustive set of rules or plans is impractical.

Researchers in the area of human-computer interaction and information visualization have exploited methods for dynamic overlaying information on maps. Bell et al. [3] use a greedy approach to dynamically layout labels of geographic locations during interactions which maintains a set of visual constraints, such as preventing object occlusion and ensuring visual continuity. Been et al. [1] proposed a dynamic map labeling method which is especially useful when users change map resolutions by zooming in and out. Kapler and Wright [10] presented a system called GeoTime to overlay temporal activities on maps.

Scientific Fundamentals

In a dynamic investigation, appropriate geographic information presentation can be created in four steps. First, a subset of geographic information content needs to be selected in context of the current investigation. Second, a visual sketch is created to describe the visual representations (i. e., visual encodings) of the selected information. Third, the selected information needs to be dynamically laid out on a map. Finally, a smooth transition between the previous map and the new map is derived. The four steps are illustrated in Fig. 1.

Content Selection

To provide a tailored presentation to diverse analyst queries introduced during an investigation, a geographic information presentation system must first decide the data content of the presentation. The goal of such content selection is to maximize the presentation desirability of selected information and minimize the presentation costs of the information.

To describe the presentation desirability of a set of information D, a function *desirability(D)* can be defined (for simplicity, all feature/metric values defined in this entry

are normalized to lie between [0, 1]). A set of metrics can be defined to measure *desirability(D)* using the relevance of information D to the current investigative focus. The relevance can be measured in multiple dimensions.

First, certain information is more relevant in a certain application domain than other information. For example, information important to a real-estate application can be obtained by analyzing real-estate web sites (e. g., www.realtor.com and www.century21.com) in order to study the importance of various housing dimensions (e. g., price and style). In particular, the *importance* of a dimension d, ($d \in D$) can be measured by its presentation prominence, including when it is presented (e. g., at the first click), where it is presented (e. g., at the top of a page), and how it is presented (e. g., in a bold face font). The final *importance* of a dimension is the average of its importance scores over many sites. Thus, domain relevance, *R(d, Domain)* can be defined as

$$R(d, Domain) = importance(d)\,. \tag{1}$$

Second, it is desirable to convey information relevant to analysis history, especially when current user requests are related to previous requests. For example, an analyst previously asked for the local average house price. Then, he wants to see the property tax information to understand where the heavily taxed areas are. Therefore, it is not effective to present tax information alone. Instead, previous price information should be presented together to assist the analyst to comprehend the relations. Let β denote a binary value such that $\beta = 1$ if the current query is a follow-up query and $\beta = 0$ if the current query starts a new context. Thus, history relevance can be defined as

$$R(d, History) = \beta \cdot \alpha^{t-t_0} \cdot desirability_{t-1}(d)\,, \tag{2}$$

where $0 \le \alpha \le 1$ controls how fast the previous information should decay in follow-up query scenarios; t is current time and t_0 is the time when d is first presented; $desirability_{t-1}(d)$ is the *desirability* of dimension d at previous query. By combining Eqs. 1 and 2, the desirability of a dimension d can be defined as

$$desirability(d) = \mu_1 \cdot R(d, Domain) + \mu_2 \cdot R(d, History)\,, \tag{3}$$

where μ_1 and μ_2 are weights of corresponding metrics.

Similarly, a cost function *cost(d)* can be defined to measure the presentation costs of information. One type of the presentation cost, for example, is the space cost which computes the pixels needed to convey one instance of dimension d in text or graphics. For example, the minimal space cost for displaying one house image is 100×100 pixels. Expert-made presentations (e. g., web pages) can be used to estimate the space costs required to depict a dimension (e. g., counting the minimal number of pixels needed to make a text string or an icon recognizable on a desktop).

By combining the desirability and cost functions defined above, an objective function can be define as

$$objective(D) = w_1 \cdot desirability(d) - w_2 \cdot cost(d)\,, \tag{4}$$

where $d \in D$, w_1 and w_2 are weights of the *desirability* and *cost* functions, respectively. The content selection can then be formulated as an optimization problem [13] whose objective is to maximize the desirability of all relevant metrics. A greedy algorithm [13] can be used to solve this optimization problem in real-time. Note that the metrics defined above are not intended to be a complete list of metrics. When needed, *desirability()* can also be easily extended to include other factors for dynamic presentation. For example, to adapt a presentation to different devices (e. g., PDA), a metric can also be introduced to measure how suitable the presentation is to the target device.

Sketch Generation

After the relevant information is selected, the corresponding visual encodings need to be decided. For example, use a circle to represent a city and use a information callout to show a house price. For traditional geographic information (e. g.,roads, cities, rivers), their visual encoding can be decided by existing map drawing approaches. For other types of customized information such as house information, their visual encodings can be automatically learned from an existing visualization corpus [14]. Specifically, each visualization example in the visualization corpus consists of annotations of its data and visual properties. Let tuple $\langle D_i, G_i \rangle$ denote the annotation of the i-th example in the corpus. Let D denote the data input of the dynamic presentation system. An appropriate visualization can be selected for D by

$$idx = argmin_i\{dist(D_i, D)\}\,, \tag{5}$$

where *idx* is the index of the best visualization example to select; $i = 1, \ldots, N$; and $\langle D_i, G_i \rangle$ is the i-th example in the corpus; *dist()* is a function defined to measure the difference between any two data annotations.

Visual Layout

To dynamically layout the visual encodings of the selected information, a set of visual layout constraints need to satisfied. For example, the occlusion of visual objects on a map should be avoided and spatial distribution of floating visual objects (e. g., callouts) should be balanced. In

traditional static map layout approaches, the set of constraints only need to be solved once for each configuration. In contrast, in dynamic scenarios such as investigations, the set of constrained visual objects will change as the underline information changes. Therefore, the visual constants need to be dynamically maintained. To maintain such visual constraints, greedy algorithms (e. g., [2]) or non-linear constraint solvers (e. g., [7]) can be adopted.

Visual Context Management

In lengthy investigation scenarios, visual display needs to be dynamically updated to effectively incorporate new information and remove old irrelevant information. This process is called *visual context management*. It is an important process to help analysts keep their cognitive map so that they can comprehend all relevant information as a coherent whole. Specifically, let S_t denote the visual scene at time t and let S' denote the new visual information at time $t+1$. Visual context management is a process that derives a set of visual transformations (e. g., visual update operators such as *add*) which updates an existing scene S_t to incorporate new information S'.

To achieve this goal, a set of *visual momentum* metrics can be introduced to measure the desirability of derived transformations. Visual momentum measures a user's ability to extract and integrate information across multiple displays. Since the amount of visual momentum is proportional to an analysts ability to comprehend information across consecutive displays [12], the visual momentum should be maximized when a visual presentation changes. Specifically, three key techniques from [12] are most applicable to the visual context management task. They are: (1) maximizing both semantic and visual overlaps of consecutive displays, (2) preserving perceptual landmarks, and (3) ensuring smooth visual transitions.

Maximizing Both Semantic and Visual Overlaps To measure visual overlap across consecutive displays, a presentation system can compute the average invariance of each visual object in S_t and its new state in S_{t+1}:

$$O_v(S_t, S_{t+1}) = \frac{1}{N}\sum_{i}^{N} inv(v_{i,t}, v_{i,t+1})\,. \tag{6}$$

Here, visual object $v_{i,t} \in S_t$, $v_{i,t+1} \in S_{t+1}$, and $v_{i,t+1} = op_i(v_{i,t})$, op_i is a visual update operator; N is the total number of visual objects in St; and *inv()* computes the invariance between two visual objects (e. g., color invariance, shape invariance).

Similarly, a semantic overlap metric can be defined to assess whether semantically related items remain together across displays. It computes the semantic relevance of S_t and S_{t+1}:

$$O_s(S_t, S_{t+1}) = \frac{1}{N^2}\sum_{i}^{N}\sum_{j}^{N} dist(d_i, d_j)\,, \tag{7}$$

where data objects d_i and d_j are encoded by $v_{i,t}$ and $v_{j,t+1}$, respectively, and *dist()* computes their semantic distance.

Preserving Perceptual Landmarks Perceptual landmarks are distinguishable features that anchor a visual context transition which in turn helps users to relate information in successive scenes. In a geographic information presentation system, geographic landmarks such as rivers and highways can be used to assist users to integrate information across successive displays. To preserve the maximal number of perceptual landmarks in a visual context, a metric can be defined to count the normalized number of landmarks in the context:

$$L(S_{t+1}) = \frac{L_{t+1}}{N}\,, \tag{8}$$

where L_{t+1} is the number of landmarks existing in visual context S_{t+1}, and N is the total number of visual objects in S_t.

Ensuring Smooth Transition Sudden changes in a scene prevents users from visually tracking the changes. As a result, the causal connection between an existing scene and a new scene may be lost. To ensure smooth transitions between successive displays, animation is often used to provide users with a powerful cue to interpret the changes. A metric can be defined to compute the average smoothness of applying a set of visual operators:

$$T(Op) = \frac{1}{K}\sum_{i}^{K} smoothness(op_i)\,, \tag{9}$$

where $Op = \{op_1, \ldots, op_K\}$ is the set of visual update operators used to obtain S_{t+1}. A function *smoothness()* is defined when implementing the operators to indicate how smooth the animation of the update is.

Visual Structuring Metrics In addition to maximizing visual momentum during a visual context transition, the structure of the visual scene also needs to be coherent after the transition. One example of a visual structuring issue is visual clutter. Visual clutter makes it difficult to comprehend a visual scene. Therefore, the visual clutter should be minimized after the integration of new information. Visual clutter can be measured using a set of factors:

$$\begin{aligned} C(S_{t+1}) = {} & \lambda_1 \cdot colorVariety(S_{t+1}) \\ & + \lambda_2 \cdot areaUsage(S_{t+1}) \\ & + \lambda_3 \cdot shapeComplexity(S_{t+1})\,. \end{aligned} \tag{10}$$

Here, weights $\lambda_1 = \lambda_2 = \lambda_3 = 0.33$, *colorVariety()* obtains the number of different colors used in S_{t+1}. Metric *areaUsage()* computes the normalized screen space occupied by S_{t+1}. Metric *shapeComplexity()* computes the average shape complexities of the shapes in S_{t+1}. To compute *shapeComplexity()*, a complexity value is assigned to each distinct shape type to indicate the cognitive effort needed to comprehend a shape. For example, a dot in a scattered plot requires much less effect to recognize than a text label.

By combining the metrics defined in Eqs. 6, 7, 8, 9, 10, an overall objective function can be defined to measure the desirability of a scene update.

$$reward(Op, S_t, S') = u_1 \cdot (O_v + O_s) + u_2 \cdot L + u_3 \cdot T + u_4 \cdot C \tag{11}$$

where u_1, u_2, u_3 and u_4 are weights of the metrics. The arguments of each metric are omitted for conciseness. Then, the visual context management problem is formulated into an optimization problem [11]. Note that this optimization could be a non-linear optimization because the defined metrics are not linear. To solve this optimization, either a greedy algorithm or an interactive non-linear algorithm can be used. Because the involved operators and visual objects are limited in [11], a simulated annealing algorithm is used where it still achieves real-time performance.

Key Applications

Investigative tasks prevail in many application domains. Geographic information are essential for many such tasks, such as location-based business intelligence and criminal investigation. Dynamic geographic information presentation is a valuable tool for these application scenarios.

Real-Estate Information Seeking

To evaluate the house prices and their trend in certain areas, a large amount of spatial information needs to be investigated. For example, local school districts affect buying decisions for people with children. The proximities to major business, cities and transportation impact house values. Even small roads and parks influence people's perception and thus the house prices. Dynamic geographic information presentation systems help analysts to view such related information integrated on maps. Moreover, when analysts switch their current investigation focus, (e.g, from schools near a house to parks near the house), the presentation can be adapted to the changing interests. For example, less relevant information (e. g., schools) may be removed due to limited space. However, the house is still preserved to assist analysts in comprehending the information in context.

A multi-modal conversation system called RIA has been developed by IBM T. J. Watson Research Center to assist users in exploring large and complex real-estate data sets [15]. RIA has implemented the content selection and visual context management components using Java on Windows. The real-estate data is subscribed from a multiple listing service, containing 2000+ houses, each with 128 attributes (e. g., price and style). RIA supports natural language- like queries where users can request for any combinations of attributes and related concepts (e. g., schools) in context. Based on the understanding of a user request, RIA automatically creates its response in two steps. First, RIA determines the data content of the response using content selection. Second, RIA designs the form of the response using suitable media and presentation techniques so that new information can be effectively integrated with the current visual scene.

It takes two steps to set up and use the content selection or visual context management method. First, static data and visual features used in metrics need to be defined (e. g., assigning data semantic categories). Building a simple data ontology helps to define these features. Second, feature-based metrics need to be formulated to model various constraints important to an application. For example, in a mobile application, device-dependent visual context management constraints can be modeled. To bootstrap the process and avoid tuning fallacy, it is recommended to start with simple settings. Thus far, RIA has used a simple data ontology, a set of basic visual operators, and equally weighted metrics to adequately handle diverse interaction situations in the real-estate applications.

Location-based Business Intelligence

In many business, location is a very important factor. Therefore, dynamic geographic information presentation could effectively help analysts evaluate this factor in lengthy business investigations. Here, examples are given on store location planning.

When a company wants to open a new store, it also needs to investigate many factors related to spatial information. For example, for a fast food company, it needs to understand the local traffic pattern, local business and attractions, and people living nearby. Again, dynamic geographic information presentation systems can integrate such related information on maps to help analysts comprehend it as a whole. Meanwhile, the presentations are tailored to analysts' current focus (e. g., traffic) to help them attend to their current information needs.

Anti-terrorist and Criminal Investigation

For anti-terrorist and criminal investigation, where an event transpires is of great importance. Many existing tools

try to incorporate maps as a component. However, other related information may often need to be viewed separately from the map. More over, a general purpose map may often contain information irrelevant to the current investigation. The irrelevant information can potentially be a distraction to analysts. Dynamic presentation systems could solve these two problems by integrating relevant information dynamically on a map based on current investigation contexts.

Public Health

To effectively prevent an epidemic outbreak, reported cases need to be tracked and analyzed in real-time. As new cases occur in different places, dynamic presentation systems can dynamically integrate customized case information on maps to assist such analysis.

Personal Web Research

With more and more information available on the Web, more people are using the Web as a personal research tool for tasks such as trip planning and relocation. In this type of research, users usually need to go through various information sources to find and evaluate related information. For example, for planning a trip, a user may go through airline, hotel, and local government information as well as reviews and opinions. Dynamic presentation systems provide a visual tool to connect scattered information spatially on-demand.

Future Directions

Software tools for investigation are receiving more and more attention because of increasingly overwhelming information. Geographic information is important to many investigative tasks that involve location related information. Dynamic geographic information presentation approaches are promising for investigative scenarios because investigations are highly dynamic.

Even though the dynamic information presentation concept has been studied for over 10 years, there are still many aspects to be improved upon in the future. For example, in many scenarios, such as text documents, information is unstructured. How to reliably assess the information relevance in dynamic investigations is still an ongoing problem.

Cross References

► Visual Momentum

Recommended Reading

1. Been, K., Daiches, E., Yap, C.: Dynamic map labeling. In: Proc. of IEEE Information Visualization (2006)
2. Bell, B., Feiner, S.: Dynamic space management for user interfaces. In: Proc. UIST '00, pp. 239–248 (2000)
3. Bell, B., Feiner, S., Hoellerer, T.: View management for virtual and augmented reality. In: Proc. of UIST, pp. 101–110 (2001)
4. Bordegoni, M., Faconti, G., Feiner, S., Maybury, M., Rist, T., Ruggieri, R., Trahanias, P., Wilson, M.: A standard reference model for intelligent multimedia presentation systems. Comp. Stand. Interf. **18**(6–7), 477–496 (1997)
5. Brewer, C.: Designing Better Maps. ESRI Press (2005)
6. Feiner, S., McKeown, K.: Automating the generation of coordinated multimedia. IEEE Computer **24**(10), 33–41 (1994)
7. Hosobe, H.: A modular geometric constraint solver for user interface applications. In: Proc. UIST '01, pp. 91–100. ACM (2001)
8. Imhoff, E.: Positioning names on maps. The American Cartographer **2**(2), 128–144 (1975)
9. Jones, P.M., et al.: Coraven: modeling and design of a multimedia intelligent infrastructure for collaborative intelligence analysis. IEEE Trans. Syst., Man, Cybernet. **1**, 914–919 (1998)
10. Kapler, T., Wright, W.: Geotime information visualization. In: InfoVis'04, pp. 25–32 (2004)
11. Wen, Z., Zhou, M., Aggarwal, V.: An optimization-based approach to dynamic visual context management. In: InfoVis'05, pp. 111–118 (2005)
12. Woods, D.D.: Visual momentum: A concept to improve the cognitive coupling of person and computer. Int. J. Man-Machine Stud. **21**, 229–44 (1984)
13. Zhou, M., Aggarwal, V.: An optimization-based approach to dynamic data content selection in intelligent multimedia interfaces. In: Proc. UIST '04, pp. 227–236 (2004)
14. Zhou, M., Chen, M.: Automated generation of graphic sketches by examples. In: IJCAI '03, pp. 65–71 (2003)
15. Zhou, M., et al.: Enabling context-sensitive information seeking. In: Intelligent User Interfaces, pp. 116–123 (2006)
16. Zubieta, J.C., Skinner, R., Dean, A.G.: Initiating informatics and gis support for a field investigation of bioterrorism: The new jersey anthrax experience. Int. J. Health Geograph. **2**(8). http://www.ij-healthgeographics.com/content/2/1/8 (2003)

Information Retrieval

► Internet-Based Spatial Information Retrieval
► Retrieval Algorithms, Spatial
► Skyline Queries

Information Services, Geography

BIN LI
Department of Geography, Central Michigan University, Mount Pleasant, MI, USA

Synonyms

GIService; GIS; Geographic information systems; Geographic information sciences; Services; Mash-ups; Location-based services

Definition

Geographic information is knowledge acquired through processing geographically referenced data. Geographic information services are (1) functionality provided by a software entity through its interfaces defined as named sets of operations and, (2) provisions of information generated from geospatial data. The development of geographic information services is closely related to distributed object technology and the Internet. It represents a major step to enable easy access to geographic information and geo-processing technology.

From software development perspectives, geographic information services represent a vertical domain of information services where many IT service components are combined to form information applications. GI-Service components are self-contained, self-describing, and reusable software objects that can be published, located, and invoked in a multiple address spaces and distributed environment. These components can be small or large and operate in different frameworks such as RPC, CORBA, DCOM, .NET, and Web Services.

Service Architecture

Different service infrastructures support different GI-Service architectures. On the desktop, DCOM/COM provides the framework for fine-grain GI-Services. On the Internet, DCOM/.NET, CORBA, J2EE, and Web Services support the distribution of GI-Services. Regardless the underlying framework, the logical architecture for distributed GI-Services has four tiers: human interaction services, user processing services, shared processing services, model/information management services (ISO19119). Different physical systems can be developed through different combinations of these tiers. For example, a thin client architecture would map human interaction services to the Web Browser, user processing and shared processing services to the Web Server and Application Server, model/information management services to the Data Server, resulting in a three-tier system. Alternatively, a thick client system can be developed by merging human interaction and user processing services into a stand-alone client, mapping shared processing to the Application Server. Google Earth is an example of a thick client.

Categories Based on the Open System Environment model of information services, the ISO19119 groups geographic information services into the following six categories:

1. Geographic human interaction services—presentation of information. Examples include Catalog viewer, Geographic viewer, and Geographic feature editor.
2. Geographic model/information management services—management of development, manipulation, and storage of metadata, conceptual schema, and data sets. Examples include WFS (Web Feature Services), WMS (Web Map Services), WCS (Web Coverage Services), and Catalog services.
3. Geographic workflow/task management services—provision of chain definition service and workflow enactment service.
4. Geographic processing services—provision of a variety of GIS functions including spatial, temporal, thematic, and metadata services.
5. Geographic communication services—typical examples are encoding (Geographic Markup Language, Keyhole Markup Language, ArcXML, and GeoTIFF) and compression of geographic data (JPEG2000, for example).
6. Geographic system management services—Examples include access control and usage rights management.

Components carrying out these services constitute the general architectural elements of geographic information services. Based on the service model, the Open Geospatial Consortium (OGC) and ISO have been working on a series of specifications/standards to guide the development of geographic information services.

OGC and Selective ISO Standards OGC and the ISO/TC211 are influential entities that develop specifications or standards of GI-Services. OGC is also a member of the W3C.

Encoding

1. Simple Features—Geometric object model of basic geospatial entities.
2. GML—ISO/CD 19136. Geographic Markup Language, as an XML-based standard format for network transfer of geospatial data.
3. Filter Encoding—XML expressions of queries. Styled Layer Descriptor—specification on symbolization of rendering of geospatial information.
4. Geographic Information Metadata—ISO19115 and ISO19139. Description of geospatial data collections and services.

Common Applications

1. Web Map Service—Display of maps in image formats.
2. Web Feature Service—Retrieval and update of geospatial data in GML format.
3. Web Coverage Service—Access to coverage data.
4. Web Map Context—Persistent representation of the state of a Web Map session for sharing and reuse.

5. Web Service Common—Common interfaces for all OGC Web Services.

Common Services

1. Catalogue Service—Publication and search of metadata.
2. Registry Service—Extension of the catalogue service using ebRIM.
3. Coordinate Transformation—Projection and transformation of geographic data.
4. Simple Feature Access—Access to geospatial data in different environments: SQL, CORBA, and OLE/COM.

Historical Background

The development of Geographic Information Services follows the path of mainstream information technology. In the early days (80s to mid-90s), like most computer software technology, Geographic Information System was a self-contained monolithic entity. Embedding GIS functionality in other software system or including external software into a GIS was a tremendous undertaking, which actually led to a specialty called system integration. Advances in distributed object technology in the mid to late 90s changed the landscape of software engineering. We first saw that object-orientation had become the dominate mode of programming and software design. Then came Microsoft's Object Linking and Embedding (OLE), the first OS level support for systematic interaction among software objects. With OLE, a Word document can contain an Excel document. As the scope of object services expands, OLE evolved into Component Object Model (COM), then Distributed Component Object Model (DCOM), and lately ".NET" that allows software units to interact through the Web. In parallel, the Object Management Group (OMG), an industrial consortium, developed the Common Object Request Broker Architecture (CORBA), which provides the reference model for the development of component technology such as JavaBeans, and other distributed object middlewares. Lastly, a lightweight framework for object services on the Web, "Web Services", has gained wide acceptance. These object standards specify how software units should be constructed, how they should be communicated with each other, and what types of common services the infrastructure must provide in order to assemble and execute component or object-based applications.

The GIS software industry and the academic community began to work with the general frameworks of distributed object technology. The OGC has been instrumental in formulating the reference specifications for developing service-oriented geo-spatial technology. Earlier efforts were in enabling interoperability among geo-spatial processing software programs. A reference model was established in 1996, followed by a series of abstract specifications for accessing spatial data. The OGC has since diligently followed the IT mainstream and guided the development of the GIS software industry with a series of service oriented specifications.

GIS software vendors have been quick in adapting to the object / component oriented framework, which can be illustrated with a brief product timeline from ESRI, the leader in the GIS software industry. ESRI released MapObjects, a collection of ActiveX components that can be assembled into application programs in 1996. In the next two years, the MapObjects Internet Map Server and the ArcView Internet Map Server both entered the market. In the subsequent year, the company released ArcInfo 8, marked a fundamental change with component technology with a comprehensive object model ArcObjects. The year 2000 represents a major push into "societal GIS" with the establishment of the geography network, a metadata service platform and major release of ArcIMS supporting multi-tier architecture and a workflow from map authoring to publication. In 2002, ESRI started "ArcWeb Services" to provide on-demand functionality and content with the Web Services framework. In 2006, along with a revolutionary wave of Web map services initiated by Google, Yahoo, and Microsoft, ESRI released ArcGIS 9.2 with a marketing as well as technical emphasis on service oriented architecture. ArcGIS Explorer, a thick client similar to Google Earth with more specialized GIS functions, maintains the relevance of the GIS software industry.

Scientific Fundamentals

Geographic information services are related to several layers of basic science. The first fundamental has to do with a general software model of Geographic Information Services. In the core of such a model is the classification of services, followed by the formal description of each service. Detail functions are further specified by the interfaces. Currently, object oriented design with Universal Model Language provides the common tools for developing software object models.

Because a key requirement in providing information services is message exchange, communication protocols are needed for describing service requirements and data contents. XML has showed wide acceptance as the language for development such protocols.

Ultimately, issues with service modeling, classification, and description, as well as service interoperations, will have to do with the ontology of geospatial information.

Ontology has been an active research area in Geographic Information Science. Most of the efforts, however, have not been directly affecting the development Geographic Information Services.

In the technical domain, architectural design and algorithm development are two key issues. Service oriented architecture is still under development. SOA is required to enable the consumption, composition, description, discovery, and publication of services, in a secure and reliable way. Algorithms must be redesigned to meet the efficiency expectations as there are potentially a large number of requests from the Web. Parallelism, once an active research area in GIS, is a promising approach to designing fast algorithms for geo-processing.

In the social science domain, there are several issues related to Geographic Information Services. A feasible business model is a prerequisite for providing sustainable Geographic Information Services. Establishing the legal and technical frameworks for handling liabilities is another infrastructural element. And finally, there are issues with usage rights of data and the protection of intellectual properties.

Key Applications

Location-Based Services

Location-based services are a unique category of GI-Services. They are distributed on wireless networks and often require real time processing. Client devices are often mobile phones and PDAs that have limited capabilities in processing, storage, and display. The types of services often include the following (OGC, 2005):

1. Directory service—Access online directory to find the nearest or specific place, product, or services.
2. Gateway service—Obtain position data from mobile devices through GMLC (Gateway Mobile Location Center) or MPC (Mobile Positioning Center).
3. Location utility service—Geocoding and reverse geocoding.
4. Presentation service—Graphic display of geo-information on mobile terminals.
5. Route service—Calculate the best route based on user specification.

Due to the specific characteristics of wireless networks, a special data encoding schema is developed. The ADT (Abstract Data Type) defines well-known data types and structure for location information through XML for Location Services (XLS).

Mobile Services

The number of mobile services of location information is rapidly rising. Below is a sample.

1. ESRI ArcPAD, ArcGIS Engine, ArcWeb Services
2. Google Mobile
3. MapInfo MapX Mobile
4. MapQuest Find Me (co-brand with Nextel)
5. Microsoft Windows Live Search for Mobiles
6. Microsoft MapPoint Location Service
7. Microsoft Virtual Earth for Mobiles
8. Tom Tom (navigation devices and services)
9. Webraska (GPS navigation software, SmartZone Geospatial Platform)

Desktop Applications

GI-Service components for desktop applications are often DCOM or JavaBean components. Each component exposes a set of interfaces as service contracts. Different components are glued together to form a desktop GIS application. ArcMap, a Windows GIS application, is built with ArcObjects, the core component library from ESRI. Other desktop applications, such as PowerPoint and MS Word, can also embed GIS components in their documents. Distributed object technology facilitated the transformation of GIS software systems from monolithic packages to a system of service components.

Network Applications

GI-Service components for Internet applications are often coarse grain, or large entities formed by smaller components such as those in ArcObjects. They are usually high level services combining a series of GIS functions and specific data. ESRI ArcWeb Services, for example, include five application services: mapping, place finder, address finder, route finder, and spatial query. A subscriber can use one or more of these services through a client program (ArcMap or ArcGIS Explorer, for example) or develop an application that combines the GI-services with other third party services.

Google, Yahoo, and Microsoft provide similar GI-services through the Internet. Google Maps, Yahoo Maps, and Microsoft Virtual Earth are all stand-alone Web applications. Meanwhile, each of them exposes a series of APIs that allow the "mash-up" of information from other Web applications. There are numerous of mash-ups developed everyday.

Software Products for Application

Access Control and Rights Management
CubeSERV Identity Management Server (CubeWerx)
Authoring
ArcIMS (ESRI)
FreeStyler (Galdos)
Application Servers

ArcGIS Server (ESRI)
ArcGIS Image Server (ESRI)
ArcIMS (Multiple protocol support, ESRI)
Catalina (WFS server, Galdos)
CubeSERV (WFS server, CubeWerx)
CubeSERV Cascading Server (WMS server, CubeWerx)
GeoServer (WFSserver, open source, Refraction Research)
OpenLayers (Map server, OSGeo)
RedSpider Web (WFS, WMS server, Ionic Software)
MapXtreme (MapInfo proprietary system)
MapGuide (Autodesk, WFS, WMS server)
MapServer (University of Minnesota, open source)
Catalog Servers
INdicio Catalog Server (Galdos)
CubeSERV Web Registry Server (CubeWerx)
ArcIMS Metadata Services (ESRI)
RedSpider Catalog (Ionic Software)
MapInfo Discovery (MapInfo, limited functions)
Clients
ArcCatalog (ESRI, metadata publishing and search)
ArcExplorer (ESRI)
ArcGIS Explorer (ESRI)
CubeXLOR (WMS client from CubeWerx)
Google Earth (Google)
Google Maps (Google)
Live Search (Microsoft)
MapBuilder (OSGeo)
NASA World Wind (NASA)
Yahoo Maps (Yahoo!)

Future Directions

The development of Geographic Information Services signifies a transition from monolithic software systems to interoperable components, from an emphasis in providing software tools to a focus on distributing information products, from a highly specialized technology mastered by the few to an information commodity consumed by the public, and from an information tool to a media of mass communication. The overwhelming popularity of online mapping services from Google, Yahoo, and Microsoft confirms that Geographic Information Services is becoming a ubiquitous information commodity.

With these easily accessible mapping services, people suddenly discovered their inherent interests in geography, locations, and maps, which are beyond finding places and getting directions. A piece of news is now attached with the geographic coordinates. Photos taken on a trip are linked to markers on the Web Map to be shared with friends and relatives. On these online maps, strangers meet, stories are shared, and communities are formed. Web maps are becoming the sand-box of the information society.

Hardware and software infrastructures will continue to improve, which will enhance the processing capability and quality of Geographic Information Services. The maturity of Service Oriented Architecture, for instance, will enable greater efficiency in authoring, chaining, publishing, and discovery of geographic information services. As a result, business and government entities will be able to integrate geographic information with other information in decision making more closely than ever before.

The availability of high quality data is likely to be a crucial condition for the development of Geographic Information Services. The issue is perhaps more in the institutional domain than in the technical one. Accessibility to high quality data inevitably contradict with privacy protection and even national security. Management of usage rights with geographic data is a challenge. The legal framework is still under development.

In the domain of academic research, development in analytical methods will play an important role in improving Geographic Information Services. Basic research in understanding earth processes and representing such knowledge as computational methods is the ultimate source of robust services. A better understanding in the spatial cognitive process, on the other hand, is increasingly important for effective interface design and information communication.

Cross References

- ▶ Catalogue Information Model
- ▶ Cyberinfrastructure for Spatial Data Integration
- ▶ Distributed Geospatial Computing (DGC)
- ▶ Geocollaboration
- ▶ Geography Markup Language (GML)
- ▶ Geospatial Authorizations, Efficient Enforcement
- ▶ Geospatial Semantic Integration
- ▶ Geospatial Semantic Web
- ▶ Grid, Geospatial
- ▶ Internet GIS
- ▶ Internet-Based Spatial Information Retrieval
- ▶ Location Intelligence
- ▶ Location-Based Services: Practices and Products
- ▶ Metadata and Interoperability, Geospatial
- ▶ OGC's Open Standards for Geospatial Interoperability
- ▶ Security Models, Geospatial
- ▶ Web Feature Service (WFS)
- ▶ Web Mapping and Web Cartography
- ▶ Web Services, Geospatial

Recommended Reading

1. OGC, OpenGIS Location Services (OpenLS): Core Services, OGC 05-016. http://portal.opengeospatial.org/files/?artifact_id=8836 (2005)
2. ISO 19119, Geographic Information—Services (2005)

3. Berre, A.J.: Geospatial Services. ISO/TC211 WG4 No. 43 (1996)
4. Buttenfield, B.P.: Looking Forward: Geographic Information Services and Libraries in the Future. Cartography and Geographic Information Systems, **25**(3), 161–173
5. Glass, G.: Web Services: Building Blocks for Distributed Systems, Prentice Hall, Upper Saddle River, NJ (2002)
6. Gunther, O., Muller, R.: From GISystems to GIServices: Spatial Computing on the Internet Marketplace. Proceedings of the International Conference on Interoperating Geographic Information Systems, Santa Barbara, CA, December 3–4 1997
7. Li, B.: Client-side Processing for Geographic Information Services. In: Geoinformatics for Global Change Studies and Sustainable Development, Manchu Li, et al. (eds.), Berkeley: CPGIS (2002)
8. Li, B.: A Component Perspective on Geographic Information Services, Cartography and Geographic Information Systems. **27**(1), 75–86 (2000)
9. Li, B., Zhang, L.: Distributed Spatial Catalog Service on the CORBA Object Bus. Geoinformatica. **4**(3), 253–269
10. OASIS, Reference Model for Service Oriented Architecture, version 1.0. http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf (2006)
11. OMG (Object Management Group), CORBATM/IIOPTM Specification 3.0.3. http://www.omg.org/technology/documents/formal/corba_iiop.htm
12. Peng, Z.R., Tsou, M.H.: Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Network. John Wiley & Sons, Inc., New Jersey (2003)
13. Tao, C.: Online GIServices. Journal of Geospatial Engineering. **3**(2), 135–143 (2001)
14. Tsou, M.H., Buttenfield, B.P.: A Dynamic Architecture for Distributed Geographic Information Services. Transactions in GIS. **6**(4), 355–381 (2002)
15. Tsou, M.H., Buttenfield, B.P.: An Agent-based, Global User Interface for Distributed Geographic Information Services. Proceedings 7th International Symposium on Spatial Data Handling, Vancouver, British Columbia, July, 1998. 603–612 (1998)
16. Tsou, M.H.: An Operational Metadata Framework for Searching, Indexing, and Retrieving Distributed Geographic Information Services on the Internet. In Geographic Information Science (GIScience 2002), Egenhofer, M., Mark, D. (eds.), Lecture Notes in Computer Science vol 2478, pp. 313–332. Springer-Verlag, Berlin (2002)
17. Zhuang, V., Wrazien, D., Wang, M.H., Huang, X.W.: Programming ASP.NET for ArcGIS Server, Thomson Delmar Learing Inc., Clifton Park, NY (2005)
18. W3C (World Wide Web Consortium), SOAP Version 1.2, http://www.w3.org/TR/soap

Information Theory

► Data Compression for Network GIS

Information Visualization

► Exploratory Visualization

Inservice

► Intergraph: Real Time Operational Geospatial Applications

Integration

► Smallworld Software Suite

Intelligence, Geospatial

DWIGHT LANIER
Institute for Environmental & Spatial Analysis, Gainesville State College, Gainesville, GA, USA

Synonyms

GEOINT; Geo-intelligence

Definition

Geospatial Intelligence (GEOINT) is defined as the exploitation and analysis of imagery and geospatial information to describe, assess, and visually depict physical features and geographically referenced activities on Earth [1]. As a specialized field within the much larger profession of intelligence, GEOINT provides action-oriented intelligence of a unique and powerful nature. By combining imagery, imagery intelligence (IMINT), and geospatial information with data collected by various other Intelligence Community (IC) components, or INTs (e. g., human intelligence (HUMINT), signal intelligence (SIGINT), etc.), a Common Operational Picture (COP) of an area of interest is developed. This approach provides context and clarity to decision makers such as war fighters, homeland security personnel, and national security policymakers [2]. The inherent spatiotemporal reference frame or geography of these data acts as the common link that fuses together what would be individual pieces of information into multi-dimensional visualizations that allow analysts to extract higher levels of understanding about the situation at hand. The planning, collection, processing, analysis, exploitation, and dissemination of information from these many sources is integrated at multiple spatial as well as temporal scales (e. g., from vast desert areas to dense urban centers, and from eras to seconds), and leveraged to make important decisions [3]. The versatility and capability of geographic information systems to effectively handle this approach to multi-source data integration and analysis is what allows GEOINT to provide a more powerful solution to intelligence professionals than would be possible if each piece of information were examined individually.

Historical Background

The fundamental history of geospatial intelligence can be traced as far back in time as any group has set out to perform reconnaissance and mapping for the purpose of fusing that information into an intelligence cycle that drives decision making (i. e., the exploration of the Louisiana Purchase by Lewis and Clarke). Geospatial intelligence, as defined here, has come a long way from the days of hand generated maps, but has always been a truly invaluable component of the intelligence cycle.

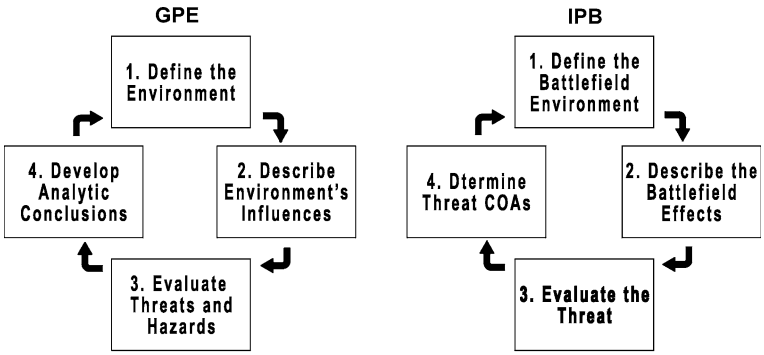
Commanders and leaders of all levels rely heavily upon knowledge of the geography of battlefields and of both foreign and domestic territories when considering their options and formulating their plans. With the advancement of the geospatial, imaging, and computer sciences (aka information technology) a more abundant source of information and improved analysis capability has found renewed strength in the decision making process. As aerial photography and photographic interpretation advanced, a wealth of information and intelligence was available about troop movements, enemy positions, and other issues and events key to national and global security. With the launch of satellites, a new era of intelligence gathering from space was ushered in. The advances of digital over film based transfer of images from satellite remote sensing platforms served to further shorten the response time to events. The tremendous boost in speed and processing capability of computers in recent times has allowed processing and analysis of geographic information (which, in its infancy was quite limited) to reach levels that finally began to unlock the potential of geographic information science. Maps that were once drawn by hand over a period of days can now be generated in a fraction of the time, and photographs that were once delayed by having to go through the delivery and exposure process now stream down as digital images to multiple users around the world in seconds. It is these events and developments, largely moved along by military necessity, that have molded what is now referred to as geospatial intelligence.

Scientific Fundamentals

Geospatial intelligence can be explored in further detail by examining the data it collects, what it does with that data on the processing side, and how the resulting products impact the intelligence community. To begin, an important distinction about the role of GEOINT in the intelligence cycle must be made. While other subject areas of the intelligence community have the ability to use GEOINT to develop their strategies and to find answers to intelligence problems, GEOINT also uses other forms of intelligence to ensure that the products it produces best match up with the overall picture. While there appears to be an overlap in the operations of GEOINT and other intelligence disciplines as far as intelligence fusion goes, it should be noted that the nature of their services is quite different. While GEOINT may bring together different components of multi-source data for the purpose of strengthening its geographic understanding and analysis of a situation, it is not to be confused with other intelligence disciplines whose sole function is to combine all sources of intelligence, including GEOINT products, and analyze them to solve intelligence problems [2].

One of the main strengths of GEOINT is that it is capable of exploiting the spatial and the temporal aspects of many sources of intelligence data. The three primary sources of data, as listed in the formal definition of GEOINT, are imagery, imagery intelligence, and geospatial information. These three components serve as the base upon which other products and data are extracted from or added to. The imagery component comes from national reconnaissance and intelligence platforms, such as commercial satellite, aerial, and unmanned aerial vehicles (UAVs). Types of imagery exploited by GEOINT cover the entire range of available remote sensing capabilities to include (but not to be limited to): panchromatic (visible), infrared, multi- and hyper- spectral, thermal, short and long wave (e. g., microwave), earth observation imagery, and other aerial platforms. Imagery collection is organized at both the local and national level with both managements typically being responsible for populating a national database [2]. Active sensor based information includes but is not limited to: Synthetic Aperture Radar (SAR), Light Detecting and Ranging (LIDAR), etc. Once imagery and other sensor information is collected and received, it is analyzed to extract valuable intelligence. The intelligence extraction step is performed by image interpreters through spectral and object-based classification techniques as well as varying forms of automated and manual feature extraction methods. The last data component, geographic information, is often a product of the first two steps. It does, however, also include other forms of data collected from surveying and mapping technologies, geodetic data, and other related products [1].

With the geographic data and imagery in hand, analysis is performed in a way that ensures that all angles of the situation are being examined. An example of geospatial intelligence preparation and exploitation is the Geospatial Intelligence Preparation of the Environment (GPE) methodology developed by the National Geospatial Intelligence Agency (NGA) (Fig. 1a). This methodology, based on the Joint Intelligence Preparation of the Battlespace (JIPB) (Fig. 1b) doctrine in use by America's military, has been modified such that it allows for the analysis and intro-



Intelligence, Geospatial, Figure 1 GPE compared with IPB [2]

duction of civilian and other non-military situations. This modified approach provides GEOINT with extreme flexibility in that the process can be used for disaster management, security planning for large events, emergency response, etc.

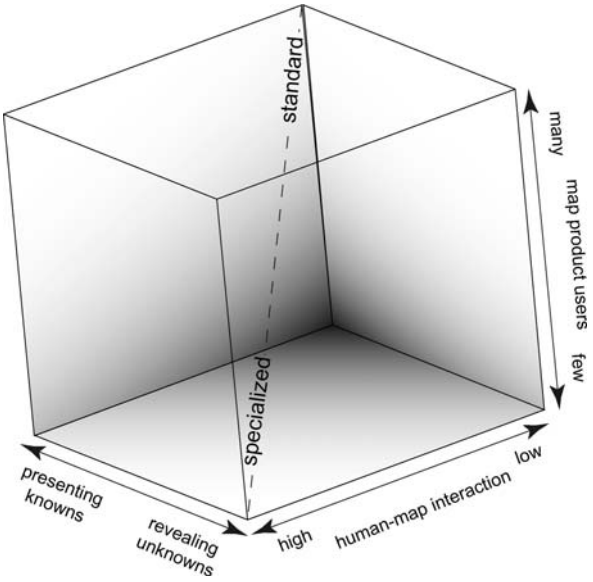
In keeping with this flexible nature, GPE is composed of four components that do not always have to occur in a strict linear fashion. The components represent key areas of the intelligence problem that provide a basic and systematic structure for evaluating any given situation and may be revisited as new information comes in. The components are:

1. **Define the Environment**: The first component of NGA's GPE methodology requires that the analyst define the environment. This first step uses basic information such as grid coordinates, vectors, latitude and longitude, altitude, natural and political boundaries, etc. to define the exact area of interest upon which the resulting GEOINT product will be built.
2. **Describe the Environment's Influence:** Once the base is established the analyst will describe the environment's influence by collecting all existing information on natural conditions, infrastructure, and culture in the area of interest. Any thing that can affect operations carried out in these regions must be considered, including: weather; transportation networks; language; cultural, political, ethnic, and religious factors and boundaries, etc.
3. **Evaluate Threats and Hazards:** In this component the analyst draws from the other INTs to evaluate threats and hazards in the area and layers this intelligence data onto the information base established in the first two components. This component includes data on the size and strength of the enemy, their tactics and doctrine, whether there is currently (or exists potential for) an insurgency in the area, any possible chemical/biological threats, etc.
4. **Develop Analytic Conclusions**: In this last component all information is integrated to develop predictive analytic conclusions. In this step the analyst attempts to determine the most likely course of action that the enemy or threat will take, and then to analyze the implications of those actions on the overall situation. During this process the analyst is constantly trying to increase his level of certainty over time by compiling and analyzing data and information until it reaches a level that allows him the confidence to raise a predictive alert or issue a report before the occurrence of an event [4].

The products created by GEOINT are wide ranging and vary in nature between what those in the field consider to be regular or standard products, and special or specialized products. Standard GEOINT products include such things as maps, charts, imagery, and digital raster and vector geospatial data. They usually support the largest population of users who do not have direct use for specialized products. Standard products are also typically of a planned and routine nature, and are designed to meet certain specifications. Specialized products are those that include the use of more advanced technologies, sensors, and often the variable of time. Specialized products at the same time allow the user to extract more detailed information in a more flexible, and sometimes more dynamic, format. Because they are often tailored to meet certain objectives, the number of users of specialized products is most often minimal.

It should be noted that not all products will fall neatly into just one of the following categories as shown in Fig. 2. Figure 2 shows a modified version of MacEachren's map use cube that helps to explain GEOINT products in general and to show the distinction between standard and specialized products.

In this conceptual space the use of a GEOINT product is shown to have three continua that form extremes in two corners but never any distinct boundaries within the space. The first continuum shows how the number of end users is typically a function of how specialized or how general the product is; with many users frequently requiring access to standard products and a smaller number of more



Intelligence, Geospatial, Figure 2 The conceptualized space of GEOINT product use (*modified from* [5])

advanced users requesting or having need for the specialized products. The second continuum defines the product in terms of the level of human to map interaction. A product that allows the user more control over changing the display properties, presentation, layers included, maps viewed, merging map displays, overlay and superposition of user data, etc. is a more specialized product than one in which all the variables are static and the user is limited in the changes they can make. The third continuum deals with the data that is being delivered to the user via the product. If the product focuses more on presenting only a few general data variables to the user with a certain message in mind (i. e. a topographic map), then the product is said to be of a more standard nature. On the other hand, products that effectively present multiple sources of information layered in such a way that the user can ask his/her own questions of the data and use it to analyze a wide array of situations are said to be more specialized. The two extremes of this conceptual space are products that provide the user with much interactivity and allow for the highest possible degree of analysis, and those that offer low amounts of interaction and simply present their data. The map product users continuum must be looked at carefully in this model. While it is true, in most cases, that more users will benefit from standard products that are simple in their presentation of data and are fairly static, a new age of dynamic products that share data rapidly across networks and allow users more and more levels of control over the presentation, display, and interaction with their data are beginning to influence a larger audience.

Key Applications

Because GEOINT allows for the layering of multiple sources of information to address any specific problem, the list of GEOINT applications is varied and extremely wide ranging. What follows are a few examples of GEOINT applications that can be further researched by visiting the NGA [6], UMC [7], and other websites listed in the recommended reading section.

Maps, Charts, Imagery

One of the most common applications of GEOINT is to build and provide paper and digital topographic, maritime, and aeronautical charts and graphics, as well as standard analyzed and unanalyzed imagery to operators across every organization.

3D and 4D Fly-Through

These 3 and 4 dimensional products are a great resource for providing commanders, planners, and other mission operators with a dynamic view of the battle space. A 3D virtual representation of the area of interest is constructed and the point of view of the observer is carried around the image along specified flight paths that are constructed to give the maximum amount of information about the layout and position of the area of interest. Other layers of information can be added to the 3D virtual world and then the whole thing can be enabled with the 4th dimension of time to recreate a given situation to enable more advanced analysis.

Data Dissemination and Fusion

With the large amount of data available, GIS provides critical infrastructure to support the collection, fusion, and rapid dissemination of geospatial and other data to a wide array of users operating around the globe.

Automated Feature Extraction

Methods are being refined to build accurate vector data of roads, building, etc. from high resolution commercial and non-commercial satellite imagery in an automated fashion. Building computer programs to do the brunt of the extraction process greatly reduces the number of man-hours involved.

Automated Target Recognition

This application focuses on using specific computer algorithms developed for a variety of sensor products to automatically detect, locate, and track the change in position of targets of interest such as vehicles, missile sites, tanks, etc.

I

Automated Scene Description

This application uses the target detection results from the ATR process and then goes one step further by providing descriptions of the contents of any scene. An example of this would be running ATR to locate a convoy moving through the desert and then having ASD provide higher-level description that the convoy is 5 miles SW of a certain facility.

Cross Country Mobility Studies

GIS is an exceptionally capable tool for analyzing the many variables that contribute to locating suitable mobility corridors across terrain. Combining variables like slope, soil composition, wet or dry conditions, land cover density and type, etc. the analyst can determine the best paths for a variety of vehicles.

Threat Range Study

These studies can be performed in a variety of ways with the same basic premise that the analyst is delineating the space between a sensitive object and a threat to that object. If the sensitive object is a boat in harbor, then the analyst will buffer out from the boat (with distance specified by the threat capability, e. g. explosives, etc.) to define the area that must be secured to prevent harm to the vessel.

3D Urban Scene Reconstruction

Constructing virtual recreations of urban areas provides extremely valuable support to operations. 3D views provide many benefits, among which is the ability to perform viewshed analysis to locate potentially good line of sight for sniper coverage of an advancing assault, or to inversely locate an enemy sniper position. 3D urban simulations are also being used to train soldiers by providing realistic situations with multidimensional environment.

Risk, Natural Hazard, and Homeland Security Management

When large scale disasters or hazardous events occur there is initially a lot of confusion and panic that must immediately be followed by coordination and purposeful action. In much the same way that action is coordinated in battle, GEOINT can provide structure for decision making, and rapid fusion and collaboration of data that is needed to handle a large scale domestic situation.

Future Directions

One of the most current and comprehensive plans to forward GEOINT was drafted as a set of challenges to NGA by a special committee formed by the Mapping Science Committee of the National Research Council of the National Academies [3]. The challenges set forth include advancements in data mining technology; sensor fusion; spatiotemporal database management systems; visualization; data sharing with coalition forces, partners, and communities at large, etc.

Cross References

- ▶ Change Detection
- ▶ Crime Mapping and Analysis
- ▶ Data Analysis, Spatial
- ▶ Homeland Security and Spatial Data Mining
- ▶ Image Mining, Spatial
- ▶ Patterns in Spatio-temporal Data
- ▶ Temporal GIS and Applications

Recommended Reading

1. Cornell Law School Legal Information Institute: http://www.law.cornell.edu/uscode/html/uscode10/usc_sec_10_00000467----000-.html (2006)
2. NGA (National Geospatial-Intelligence Agency): Geospatial Intelligence (GEOINT) Basic Doctrine: Publication 1–0 (2006) On-line at http://www.nga.mil/NGASiteContent/StaticFiles/OCR/geo_pub1.pdf
3. National Academies of Science. Priorities for GEOINT Research at the National Geospatial-Intelligence Agency. (2006) ISBN: 0–309-65862–4
4. MacEachren, A.M., Robinson, A., Hopper, S., Gardner, S., Murray, R., Gahegan, M., Hetzler, E.: Visualizing Geospatial Information Uncertainty: What We Know and What We Need to Know. Cartogr. Geogr. Inform. Syst., **32**(3), 139–160 (2005)
5. MacEachren A.M., Taylor, D.R.F. (eds.): Visualization in Modern Cartography. Elsevier, Oxford (1994)
6. National Geospatial-Intelligence Agency (NGA) in the United States of America. http://www.nga.mil/
7. Center for Geospatial Intelligence at the University of Missouri – Columbia (UMC). http://geoint.missouri.edu/
8. Defence Intelligence Collection Group (ICG) in the United Kingdom. http://www.mod.uk/DefenceInternet/AboutDefence/WhatWeDo/SecurityandIntelligence/DIS/ICG/
9. Defense Imagery and Geospatial Organization (DIGO) in Australia. http://www.defence.gov.au/digo/
10. Gordon, M.K.: NGA's Historian Reflects on First 10 Years, Pathfinder 4(5) Sept.–Oct. National Geospatial Intelligence Agency (2004)

Interaction, Space-Time

- ▶ Temporal GIS and Applications

Interdisciplinary

- ▶ Uncertainty, Modeling with Spatial and Temporal

Interesting Pattern

► Frequent Pattern

Interestingness Measures

► Co-location Patterns, Interestingness Measures

Intergraph: Real Time Operational Geospatial Applications

PETER BATTY, IGNACIO GUERRERO
Intergraph Corporation, Huntsville, AL, USA

Synonyms

Geomedia; G/Technology; Inservice; M&S Computing; Photogrammetry; Z/I imaging; Standards, geographic data object (GDO); TerraShare; Image station; Automated vehicle location (AVL); WiMAX

Definition

For many years, Intergraph has been one of the market leading commercial companies in geospatial technologies, with a broad range of products and a presence in many vertical markets. At the time of writing, Intergraph has two divisions: Process, Power and Marine (PP&M), which focuses on Plant Design and Engineering, and related technologies; and Security, Government and Infrastructure (SG&I), which sells a range of products and services leveraging geospatial technology, using a broad definition of that term.

The GeoMedia family of products is the most widely used of Intergraph's geospatial offerings, with applications in many industries including government, transportation, military, intelligence, utilities and communications. Intergraph also offers a suite of products specifically focused on the utility and telecommunications industry: G/Technology is focused on management of complex networks, and InService is used for outage management and workforce management. Intergraph has long been a market leader in the Public Safety market, providing emergency call taking systems – which many might not think of as "GIS", but which is fundamentally a geospatial application. Its products in this area have expanded to serve the growing market for emergency response and national security. Intergraph also provides a range of products for the capture, processing and distribution of imagery and terrain data, including cameras for aerial photography, the ImageStation family of products for processing, and TerraShare for distribution.

Intergraph believes that its combination of products and experience in both traditional geospatial applications, and in the real time mission critical space of public safety and security, position the company well in the growing market for real time geospatial applications.

Historical Background

Intergraph has its origins in the 1960s US space program that led to the successful Apollo 11 moon landing in 1969. At that time Jim Meadlock worked at IBM's Federal Systems Division and was responsible for integrating the computer operations for the Saturn Launch Vehicle. In February 1969 Meadlock formed M&S Computing along with Nancy Meadlock, Terry Schansman, Keith Schonrock and Bob Thurber. Jim Taylor, who would later become Intergraph's second President and CEO, joined M&S Computing in May of 1969 as employee number 6 and "honorary" founder. In 1980 the M&S Computing name was changed to Intergraph and Intergraph became a public company in 1981 trading on the NASDAQ market under the symbol INGR. Intergraph would remain a public company until November of 2006 when it was acquired by private investors.

In the 1970s, M&S Computing engaged in contracts with NASA and US Army to convert analog missile guidance systems to digital guidance systems. As part of this activity, M&S Computing developed graphic interfaces for displaying missile trajectories. Using this experience, M&S Computing developed an interactive graphics system integrating Tektronix terminals with a digitizer tablet from Summagraphics and a Digital Equipment PDP-11 minicomputer. The first interactive graphics application was a program for NASA used for printed circuit-board design. In this application, engineers were able to manipulate on-screen graphic objects representing the circuit-board components. Based on this interactive graphics technology, in 1974 M&S computing developed a mapping system for the City of Nashville. This represented the birth of GIS at Intergraph (then M&S Computing).

During the early M&S days other seminal GIS events were taking place at Harvard University. In 1964 Howard Fisher founded the Harvard Laboratory for Computer Graphics and Spatial Analysis. Fisher received a grant from the Ford Foundations to develop SYMAP, a program for mapping using a line printer as output device. Students and researchers at the Harvard Laboratory would later on influence greatly the GIS industry. Among them was David Sinton who joined Intergraph in 1979, and Jack Dangermond, who went on to found ESRI.

In the 1980s Intergraph graphics technology evolved into "turnkey" systems integrating hardware and soft-

ware. Intergraph adopted the successful Digital Equipment VAX/VMS minicomputer platform. This platform was augmented with specialized graphic processors. In this period Intergraph pioneered advances in software and hardware that brought interactive computer graphics to new levels. Intergraph introduced the Interactive Graphics Design Software (IGDS) that enabled persistent storage of graphics data in design files. In those days hard disks were not fast enough to drive display. One of Intergraph's most significant contributions was the development of a smart disk controller that was able to read graphics files (in DGN format) fast enough for interactive graphics. Intergraph systems delivered "Intergraph disks" configured at the factory for high performance. As base interactive graphics evolved so did number of CAD applications including GIS. During the 1980s Intergraph began important developments to build data capture and cartography systems. A contract with the Defense Mapping Agency (DMA) led to the development of new generation stereo plotter; Intermap Analytic (IMA). In this timeframe Intergraph also developed a variety of specialized scanners and high resolution plotters and film writers. Notable among these was the MapSetter series.

In the late 1980s and early 1990s Intergraph shifted from mini-computer based systems to UNIX workstations. Intergraph engineering workstations were powered by advanced 32-bit RISC processors of Intergraph design. This is the time when Intergraph microprocessor intellectual property was built which later led to the patent violation lawsuit filed against Intel filed in 1997. The Intel dispute eventually led Intergraph to exit the hardware business to focus solely on software and services. The Intergraph Intel litigation culminated with a settlement which recognized the value of Intergraph intellectual property.

During the late 1980s and early 1990s Intergraph developed robust GIS systems and applications. FRAMME, targeted for the Utilities industry, introduced advanced data modeling incorporating a rules based feature model and network model. TIGRIS introduced an innovative object-oriented native topological model. MGE introduced robust CAD-based GIS focused on data capture and spatial analysis. These systems incorporated advanced technologies using proprietary data storage and proprietary object systems as other systems did at the time. The base graphic platform used for FRAMME and MGE was Microstation, a system originating in IGDS developed by Bentley Systems.

Parallel with the evolution of GIS software, Intergraph continued developing specialized mapping systems, photogrammetry in particular. In 1988 Intergraph announced the formation of Z/I Imaging, a joint venture with Carl Zeiss to build state of the art end to end photogrammetry systems. Z/I Imaging developed analog and digital mapping cameras as well as processing software. Z/I imaging complemented the photogrammetry software portfolio with TerraShare for image management. In 2002 Intergraph acquired 100% ownership of Z/I Imaging.

In the mid to late 1990s, Intergraph embarked on the development from the ground up of new GIS technology that became the base for GeoMedia and G/Technology. Several tremendously important concepts emerged at that time. First it was recognized that GIS was a database (transactional) application rather than a document (or tile) based application as it was in previous generations. Secondly, it was recognized that GIS should leverage on open IT standards rather than rely on proprietary technology. Thus GeoMedia technology is based on standard databases (such as Oracle) and standard object technology which enables enhanced flexibility and extensibility. GeoMedia pioneered data access to multiple sources with no translation. This was realized by the creation of a specification (Geographic Data Objects – GDO) which provided a public API for data access. The fundamental change was the change from a format specification to an API specification. With the advent of the worldwide web, Intergraph leveraged the open standards based GeoMedia architecture to develop web GIS using objects in common with desktop applications.

Scientific Fundamentals

GeoMedia

GeoMedia is a broadly applicable suite of products for accessing, displaying, analyzing, and presenting spatial data. GeoMedia provides a full suite of powerful analysis tools, including attribute and spatial query, buffer zones, spatial overlays, and thematics. The GeoMedia suite of products consists of desktop products, web products and add-on applications for a variety of disciplines: terrain analysis, grid analysis, image processing, parcel management etc.

These products share a geospatial data access layer which is independent of the physical database storage. This capability, pioneered by GeoMedia, specifies a data access API or access protocol as opposed to physical storage. GeoMedia thus is able to implement "data servers" that connect a variety of physical storage sources: Oracle Locator database, ESRI shape files, MapInfo files, etc. These data sources are accessed directly in native mode with no translation. When connecting to an industry standard database like Oracle Locator, this approach results in automatic interoperability with other applications based on native Oracle Locator data storage.

Geospatial functionality is implemented as GeoMedia objects which expose a pubic API for customization and extensions. GeoMedia objects can be programmed to create extensions to functionality hosted by the GeoMedia desktop or to create web applications.
GeoMedia desktop applications offer a sophisticated event management and command system to support interactive graphic functions. A unique facility ("pipes") implements dynamic geospatial processing. Pipes are able to react in real time to database changes and re-compute analysis results. For example, if the user creates a buffer zone around a road feature and the road is edited, the buffer zone will immediately update itself. This real-time dynamic capability is complemented with a powerful expression evaluator system for attributes, geometry and style. GeoMedia web applications can be created from scratch using the public APIs or they can be generated using a publishing mechanism that derived the application look and feel from settings stored in the database.

G/Technology

There have historically been two approaches to geospatial applications in utilities and communications: one has been to apply generic geospatial products, and the other has been to develop specific geospatial products focused on the special needs of this market segment. Intergraph's G/Technology falls into the latter category, following in the tradition of other products such as IBM GFIS, which had its origins in the late seventies and was strong in the utility market in the eighties and early nineties; Smallworld, which came to prominence in the utility and communications market in the early to mid nineties, and Intergraph's own FRAMME product, which was the predecessor to G/Technology.
There are several issues specific to utilities and communication which have led to the development of products specifically focused on this market. One is the requirement to model complex networks. The traditional simple model of point, line and area features is not rich enough to effectively handle more complex network features such as switches and cabinets, which may have many connections at a single point, with complex logic to determine how items are connected. Ability to handle other types of relationships and compound features is also important for these types of systems. G/Technology has sophisticated data modeling capabilities which address all of these issues.
Utility and communications applications also present demanding challenges with regard to scalability, in terms of both database volumes and number of users. G/Technology utilizes an innovative caching technology, the Dynamic Data Cache (DDC), which enables the use of Oracle as the repository for all aspects of the data, both spatial and non-spatial, while also providing the performance benefit of a highly optimized graphical data format. Thirdly, managing many concurrent users involved in long transaction updates is important for large utilities and communications customers. G/Technology implements a highly scalable version management scheme within Oracle, which has been proven in production use with tens of thousands of versions.

TerraShare

TerraShare is an enterprise infrastructure for management of raster and elevation data, and earth imaging production. TerraShare integrates storage infrastructure with end-user production and exploitation tools, enabling individuals and organizations to address their geospatial data management, access, and distribution needs effectively. This unique solution addresses the complete geospatial raster and elevation information life cycle within the enterprise.

ImageStation and Aerial Cameras

Intergraph provides a comprehensive suite of digital photogrammetric software. The ImageStation product line includes applications for the entire production workflow. By combining Intergraph's data acquisition, data exploitation, and data distribution hardware and software systems, Intergraph provides a completely integrated, end-to-end photogrammetric workflow solution.
Intergraph's full line-up of products includes film and digital cameras, mission planning software, and flight management/sensor management systems. Intergraph's Z/I Imaging DMC (Digital Mapping Camera) is the industry's most innovative and precise turnkey digital camera system. Z/I Imaging partnered with Carl Zeiss to develop a unique lens design, minimizing distortion and maximizing resolution. The DMC supports aerial photogrammetric missions for the broadest range of mapping, geographic information systems (GIS), and remote sensing applications.

I/CAD and Security

I/CAD is Intergraph's solution for Computer Aided Dispatching, which was initially developed for the Public Safety market, to handle call taking and dispatching for police, fire and ambulance services, and which is a market leader in this space. This was one of the first applications to leverage early automated vehicle location (AVL) technologies. This platform is also used for dispatching in other markets, including automobile clubs, utilities (for both outage management and workforce management), and most recently in the growing market for security applications.

In the security market, the platform has been extended to handle integration with video cameras and other types of sensor, including access control and intrusion detection systems, and chemical, biological and radiation sensors. A key technical feature of I/CAD is its high availability capabilities, which provide redundancy in all areas of the system and have enabled many systems to run continuously for years with zero downtime.

Key Applications

Intergraph provides geospatial solutions to a wide range of industries, including commercial photogrammetry, transportation, public safety, security, local and central government, military and intelligence, utilities and communications. The following sections outline some typical applications in these areas.

National Security

Intergraph provides a range of applications to support national security, including antiterrorism and force protection initiatives. Many real-time inputs from sources such as video cameras, location tracking devices, access control and intrusion detection systems, chemical, radiation and biological sensors, and traffic sensors can be combined with other geospatial data to present a "common operating picture" to emergency responders and planners, which provides them with real time situational awareness. Applications handle "consequence management", handling complex responses plans to different situations. Intergraph solutions are employed in some of the most high profile security systems in the world.

Public Safety

Intergraph provides emergency call taking and response systems for police, fire, ambulance and other emergency response agencies. High performance and high availability are of course key aspects of such systems. Again, interoperability between multiple agencies is often an important element of emergency response systems. Intergraph also provides related applications, such as records management systems for recording and analyzing incidents, crimes and other information. Spatial analysis plays an important role in highlighting patterns in this data.

Government

Intergraph solutions are used in a wide variety of government applications, from public safety preparedness and response to land information, public works, and transportation management, spanning the complete life cycle of geospatial data – from creation to distribution. Intergraph solutions are used in managing spatial data infrastructures and creating eGovernment solutions, often leveraging OpenGIS standards which play an important role in these application areas.

Transportation

The transportation industry employs Intergraph geospatial solutions to ensure dissemination of accurate and timely information, keeping people and products moving safely and efficiently. From collecting information in the field to visualizing roadway assets and congestion on a screen, map, or via the Web, geospatial solutions are helping hundreds of state, provincial, and national governments solve their asset management and capital planning problems. Transportation operations management applications allow rail and transportation agencies better maintain accurate roadway and railway inventories to serve their constituents. These solutions can be integrated with the capabilities for incident management, command and control, and security, discussed previously, enabling personnel to see all transportation-related data in a single, common operating picture. This helps transit agencies, airports, and seaports simplify monitoring, and speed response to security-related issues.

Utilities and Communications

Managing assets, such as personnel, mobile resources, equipment, and facilities, can be one of the most time-consuming and labor-intensive aspects of any utility or communication organization. Managers require tools that streamline workflow processes to effectively meet operational requirements, budgetary constraints, and regulatory demands. Geospatial resource management applications allow organizations better manage and secure their resources – from service technicians, repair equipment, and emergency response vehicles to military field personnel and equipment – resulting in increased productivity and profitability, and improved customer satisfaction.

These infrastructure and resource management applications improve response time and improve customer service by enabling these organizations to use personnel effectively, eliminate redundant efforts, and manage assets easily and efficiently. Broad infrastructure networks of pipes, wires, cables, and fiber require management of many assets, including service centers, mobile work crews, vehicles, and facilities. By binding interrelated systems – geofacilities, outage, work, and mobile workforce management – information is instantly available across the enterprise. Through the integration of asset and maintenance data with the geospatial data of the network, these agencies and organizations can follow prioritized maintenance plans, dispatch mobile workers efficiently according to

location and availability, and keep an up-to-date inventory of assets – even across expansive geographic areas.

Military and Intelligence

Geospatial intelligence exploitation and production solutions help military and intelligence professionals meet their operational goals and enable data sharing across the enterprise and around the world. Production systems for the creation and maintenance of spatial data must meet rigorous specifications, such as those defined by national intelligence agencies. Again, a trend in this area is towards combining many sources of real time data into a geospatial common operating picture.

Future Directions

Wireless Networks, Location Tracking and Sensors

With wireless network access becoming increasingly pervasive, there will be growing use of this technology in geospatial applications. There are several technologies competing to provide this service, including cell phone networks, WiFi, WiMAX and satellite-based approaches for remote areas.

Location is of significant interest for mobile applications, and location tracking devices are progressing rapidly – GPS is becoming commoditized. In addition, local tracking technologies such as RFID and ultra wideband (UWB) are gaining momentum, which open new application areas for geospatial technologies. The number of non-spatial sensors is growing phenomenally, including video cameras, traffic sensors, etc., which require a geospatial context to make sense of them.

These sources, along with others, will provide huge amounts of geospatial data, enabling the creation of many new applications which provide new technical challenges. For many applications, self-updating databases will replace the current highly manual update processes. In a few years, the technology will exist to know where almost everything is, all the time. While this technology creates enormous opportunities, it also brings challenges in handling potential privacy issues.

Real-time Geospatial Applications

All of the factors mentioned previously are contributing to considerable growth in real-time operational geospatial applications. With the recent natural disasters and terrorist incidents, security and emergency response are major focus areas for this type of application. Transportation agencies and organizations are taking great strides to protect travelers, employees and assets. Major transportation authorities are installing infrastructure protection systems that use thousands of cameras, as well as intelligent video and sensor technologies, to monitor subway and rail systems, highways, bridges and tunnels. This type of system can provide a real-time common operational picture, which provides a detailed view of what is happening in the real world.

Geospatial Technology Enters Mainstream IT

Major IT vendors such as Google, Microsoft and Oracle all now provide significant geospatial technology. Many aspects of basic geospatial functionality are becoming commoditized, and increasingly large amounts of data are becoming freely or very cheaply available. This change removes what was previously the biggest single obstacle to broader adoption of geospatial technology: the prohibitive cost of each organization capturing and maintaining its own data. The IT market is finally reaching the point where geographic information is regarded as just another data type. This trend makes it increasingly hard to look at the geospatial market as a distinct entity, since geospatial elements are being introduced into so many different applications.

This infusion of technology, talent and new ideas is excellent for the geospatial industry, but it poses some questions for traditional geospatial vendors. Those who focus on providing vertical applications to solve specific business problems are better positioned to leverage these new developments as appropriate. Vendors who focus on horizontal platform solutions face more of a direct challenge from this change in the market.

Cross References

- ▶ Photogrammetric Applications
- ▶ Smallworld Software Suite

Recommended Reading

Prendergrasr, S.: Intergraph's Jim Meadlock Reflects on 30 Years of Graphics Technology. GeoWorld (October 1999), 54–56

Dunn, J.: Intergraph at the Heights. Business Alabama **18**(8), 19–24 (2003)

Chrisman, N.: Charting the Unknown: How Computer Mapping at Harvard Became GIS. ESRI Press (2006). ISBN 1-58948-118-6

Internet GIS

DAVID MORETZ

Department of Geography, University of Minnesota, Minneapolis, MN, USA

Synonyms

Web GIS; Distributed GIS

Definition

Internet GIS can be defined as network-based geographic information services that utilize both wired and wireless Internet to access and distribute geographic information, spatial analytical tools, and GIS web services. The basic concept of a distributed GIS is a major paradigm shift from the traditional desktop GIS model, which encouraged the use and development of proprietary and all-encompassing GIS software programs and data models. With Internet GIS, the underlying design concept is radically changed to shift the focus from proprietary architectures to standards-based components with specialized functions that can interface with other components and GIService providers. This shift has altered the landscape of GIS to foster the growth of many small Internet GIS service providers and specialized services.

Historical Background

The emergence of Internet GIS is significant, as it creates the ability to more easily share and disseminate GIS data, thereby reducing the amount of redundant data collection and creation. The ability to access GIServices over the Internet also expands the reach of GIS to organizations and governments that previously did not have the capacity, funds, and/or skillsets to implement full GIS capabilities previously. Finally, with the development of Internet GIS, applications are being developed that target less sophisticated GIS users and function to broaden the awareness of and provide practical applications that have mass appeal and provide useful benefits to everyday human activities. Examples of some of these applications include Google Earth, MapQuest, and OnStar.

Underlying the concept of Internet GIS is the issue of data publishing and distribution. Internet GIS implies the distributed nature of data and therefore allows data providers to maintain their data directly at the source, ensuring the timely updates and version controls that are necessary to maintain data integrity and accuracy. At the same time, Internet GIS enables service providers to remotely access the source data warehouses directly over the Internet without having to download and continually update large datasets. In order to support the evolving development and proliferation of Internet GIS, the fundamental framework of geographic information systems has to be altered to accommodate this shift. Included in the re-architecting of GIS is the issue of the supporting technology infrastructure, which includes network and software configurations to enable a Client/Server architecture. Other issues to be considered are the structure and nature of data and the necessity for data standards to enable integrated GISystems. And finally, with the introduction of Internet GIS, the landscape and functionality of the basic GIS applications change. Internet GIS introduces the concept of Web Mapping and distributed GIServices that enable specialization and integration with GISystems from around the world.

Scientific Fundamentals

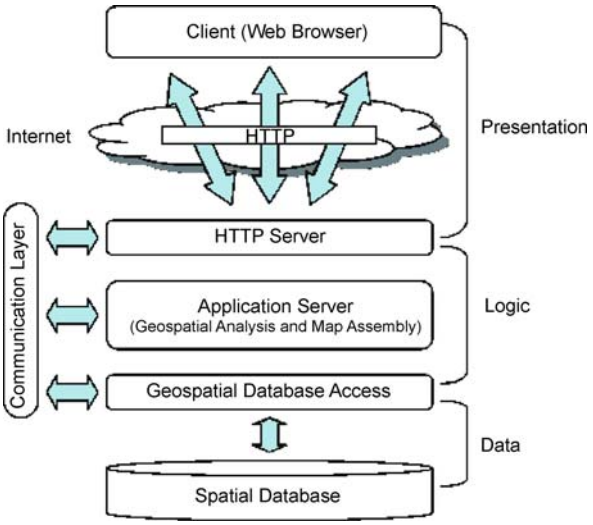
As the name implies, Internet GIS relies on the wired and wireless Internet for users to communicate or interact with spatial data storage and processing facilities. This contrasts with the concept of desktop GIS where the user interacts with spatial data and processing facilities via a GIS application and data stored locally or accessed over a LAN (Local Area Network).

This difference has significant implications in terms of how data is stored and processed, because the nature of a distributed environment is inherently more complex. Given the fact that spatial data is significantly larger than non-spatial data, the complexities of Internet GIS are magnified. Examples of some of the unique aspects of spatial data include the shear size of the data sets and the complex indexing and filtering of data needed to enable efficient transfer of data over a network. Spatial data sets often are measured in terms of Terrabytes instead of Megabytes, and relationships between the data are based on spatial components such as orientation to and distance from. The following sections describe some of the unique attributes of the Internet GIS Architecture.

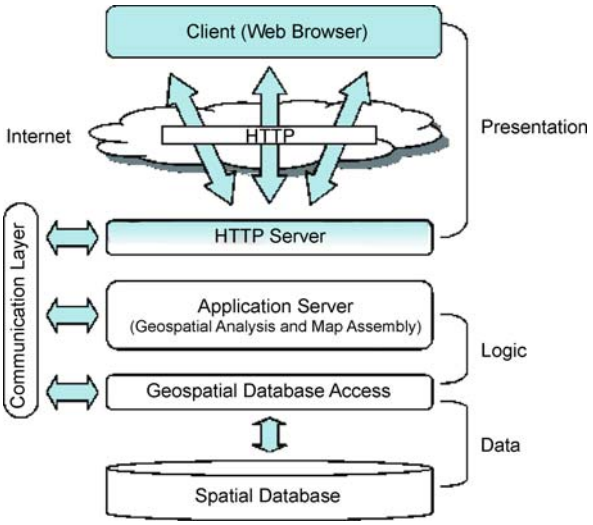
Internet GIS Architecture

Most GIS applications utilize the client/server model. The basic concept behind this computing model is that one element makes a request (the client) and another element fulfills that request (the server). An analogy can be made to a restaurant where a customer requests an entrée from the menu and the waiter returns with the desired entrée. The typical client/server application consists of three primary elements: presentation, logic, and data. The presentation represents the user interface, the logic represents the processing, and the data refers to the database or database management system. In the restaurant analogy, the presentation would be the menu, the logic would be the process of the waiter placing the correct order with the chef, and the data would be the actual entrée.

When this model is applied to Internet GIS, the simplistic view is of a three-tiered system in which the web browser represents the client, the web server represents the logic, and the database or database management system represents the data. For example, a person would use their web browser to enter the URL that contains the spatial data (or more practically, a map) that is desired. The web



Internet GIS, Figure 1 Basic presentation of Internet GIS architecture and breakdown of the three-tiered structure



Internet GIS, Figure 2 Internet GIS architecture showing the partition for distributed presentation

browser would then connect to the specified web server via the Internet to ask the server for that particular data. The web server would then transfer that request to the database management system. The database management system would then return the requested data to the web server. The web server would in turn reformat the data into a format that the web browser could understand, and then send the data back to the web browser in the form of HTML.

Conceptually, Internet GIS architectures typically consist of multiple layers. While the three-tiered structure tends to provide a high-level model for understanding the basic components of Internet GIS, the actual systems that implement the logic and data tiers of the architecture tend to be more complex.

Diagram 1 is a basic model for the architecture logic behind an Internet GIS.

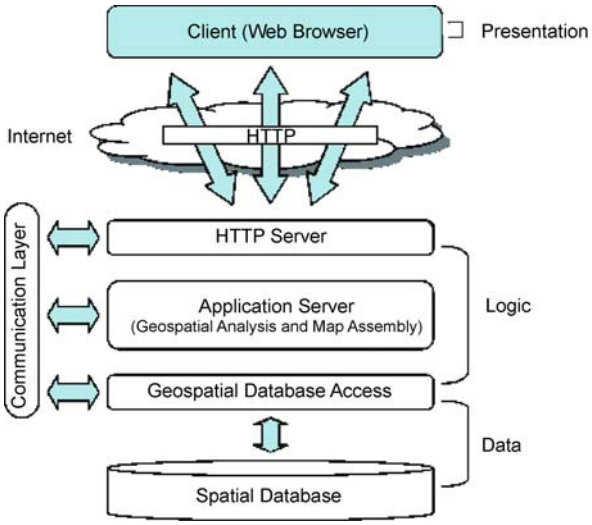
The capability and flexibility of Internet GIS becomes interesting when the components that make up the architecture are considered as individual components that can physically reside on any computer in the world. Internet GIS systems can access data warehouses directly without having to download and store separate datasets. Internet GIS systems can also integrate with other Internet GIS systems at the Logic tier to utilize distributed GIServices. Through APIs, it is possible to access geospatial analysis tools and processes that reside within other systems. These capabilities open up a whole host of possibilities for broad distribution and access to geospatial data and analysis.

While the three-tiered application elements – presentation, logic, data – represent clean partition points, complexities and greater functionality and flexibility can be gained by moving the partition points to the boundaries between the elements. Any combination of these partition points can and usually are implemented in Internet GIS depending on the desired application and infrastructure. The following is a description of the various partition points and their implications:

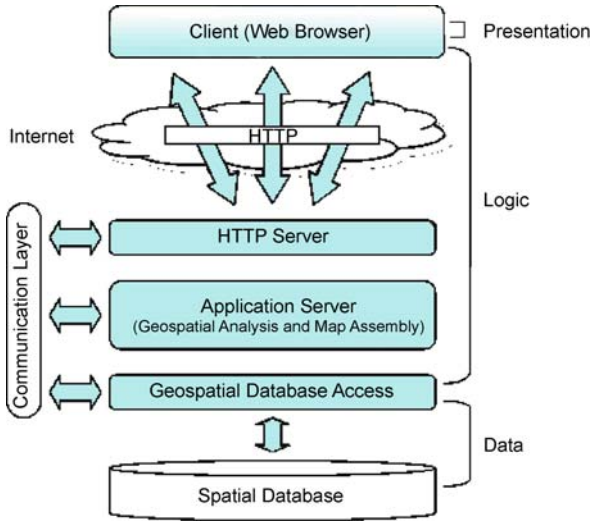
Distributed presentation *[partition point on presentation element]:* A very thin client configuration in which the data, logic, and even part of the client resides in the server environment. The client is responsible for presenting only a portion of the interface or a mirror of the server environment. A simple Web browser that has no plugins or applets and only renders HTML is another example of a distributed presentation.

Remote presentation *[partition point at boundary between presentation and logic]:* The entire presentation functionality is on the client side, while the logic and data are on the server side. An example is a server-based Internet GIS that uses CGI to process user requests on the server. The results are then presented to the web client.

Distributed function *[partition point on logic element]:* The distributed function partition splits the logic element between the client and the server and puts the presentation on the client machine. It allows the client to execute some functions while it sends other more complex functions to the server for processing. Frequently, Internet GIS that use Java applets or ActiveX controls fit into this category. Examples of the basic functions performed on the client machine include query, zoom and pan, while functions such as address matching and image analysis are performed on the server.



Internet GIS, Figure 3 Internet GIS architecture showing the partition for remote presentation



Internet GIS, Figure 4 Internet GIS architecture showing the partition for the distributed function

Remote data access *[partition point at boundary between logic and data]:* The remote data access partition puts presentation and application logic on a client that retrieves data from a remote database. This architecture is called a thick client, meaning the client is responsible for all logic operations. An example is an Internet GIS that uses SQL APIs to make calls directly to a relational database.

Distributed database *[partition point on data element]:* The distributed database partition splits the data management function between the client and one or more servers, while allocating the logic and presentation elements to the client.

Distributed GIS Standards

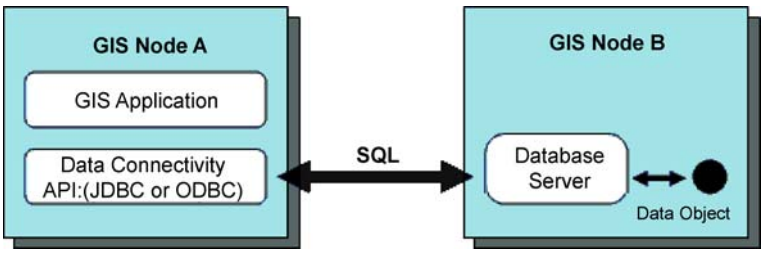
While Internet GIS creates the possibility of fully integrating geospatial data and geoprocessing resources into mainstream computing, the challenge resides in developing a widespread infrastructure of interoperable geoprocessing software and geodata products. To address this challenge, the Open GIS Consortium was founded by members of both industry and academia in 1994 to develop a framework for software developers to create software that enables users to access and process geographic data from a variety of sources across a generic computing interface within an open information technology foundation.

The framework created by the Open GIS Consortium is called the OpenGIS specification. The OpenGIS specification includes an abstract specification and a series of implementation specifications for various DCPs such as CORBA, OLE/COM, SQL, and Java. Developers use OpenGIS conformant interfaces to build distributed GIS-services, which include middleware, componentware, and applications.

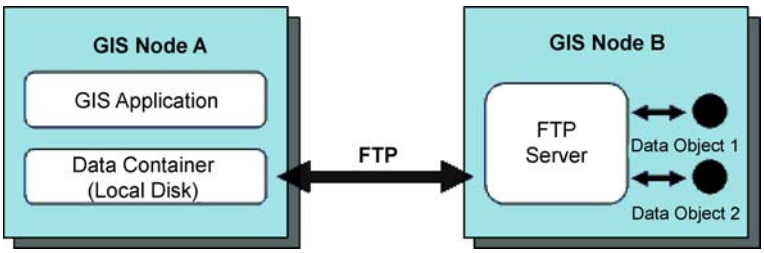
The OpenGIS specification is based on three conceptual models: the Open Geodata Model (OGM), OpenGIS services, and the Information Communities Model. The OGM specifies a common data model that uses object-based and conventional programming methods to digitally represent the earth and earth phenomena mathematically and conceptually. OpenGIS services defines the set of services needed to access and process the geodata defined in the OGM and to provide the capabilities to access, manage, manipulate, represent, and share the geodata with the GIS community. The Information Communities Model employs the OGM and OpenGIS services in a scheme to automatically translate data between different geographic feature lexicons, facilitating the collaborations among different GIS research domains and applications.

In order to successfully organize, maintain, and transfer data between organizations and systems, a geospatial metadata standard is essential. In the development of a distributed-network environment, the use of metadata plays a vital role for the interoperability of heterogeneous systems and data models. The conceptual model for geospatial metadata includes information regarding the description, history, and findings of the data. The description focuses on generic features of the data; the history refers to the derivation, update, and processing chronology of the datasets; and the findings consist of aspects such as precision, consistency and accuracy.

While there are several variations of metadata standards, the two most commonly used standards are the FGDC and ISO standards. Both standards require hundreds of fields to be filled out completely and focus on components such as identification, data, quality, spatial data organization, spa-



Internet GIS, Figure 5 Data connection representation for Internet GIS with remote data access



Internet GIS, Figure 6 Data connection representation for Internet GIS with a distributed database

tial reference, entity and attribute, and distributed information.

Geography Markup Language

To enable geodata interchange and interoperability over the Internet, a standard data structure is needed. Geography Markup Language (GML) has been designed to define this structure. GML is a markup language based on the XML standard to construct structured spatial and nonspatial information to enable data sharing and interchange over the Web. It offers a standard way to encode spatial features, feature properties, feature geometries, and the location of the feature geometries based on a standard data model.

GML supports interoperability among different data models by providing standard schemata and metadata to describe the geospatial data. When GML-coded geospatial data are transported, all the markup elements that describe the spatial and nonspatial features (geometry, spatial reference, etc.) are transported to the recipient. Based on the standard markup elements, the recipient knows exactly what each data component means and how to process that data so that nothing gets lost or distorted. The downside of GML, however, is the fact that the detailed description of GML feature elements cause the GML coded data to be very large. The large size of GML is currently a problem for transporting it over the Internet, but compression methods are being studied to try to reduce this current constraint.

Key Applications

Internet GIS has a wide range of applications and uses. Examples of these applications can be arranged into general groups such as data sharing and dissemination, data processing, and location-based services, and Intelligent Transportation Systems. The following is a list of some of these applications and a brief description regarding their use.

Data Sharing and Dissemination

Internet GIS is the ideal mechanism for data sharing and exchange over the Internet. Not only can simple raw data be distributed through FTP, but it can also be searched for and used as if it were local. For example, if a person has a need to obtain information about the Connecticut River, it is possible to access a GIS clearinghouse or data portal to find and download the necessary data. Because of the emerging data standards, this data is in a format that is usable for all users despite their chosen software application.

Online Data Processing

Traditionally, data analysis was only available to GIS professionals who had access to expensive and complicated GISystems. With the advent of the Internet and therefore Internet GIS, many data processing procedures are now available over the World Wide Web. Popular websites such as Mapquest and Yahoo! Maps in particular allow users to perform basic spatial network analysis such as shortest path queries (general driving directions from point A to point B) and buffers (all of the Chinese restaurants within 10 miles of point A). Also, commercial software such as Google Earth provides users with easy access to spatial data from all over the world.

Location-Based Services

Location-Based Services refer to a system that provides real-time data about a location and its surrounding areas. This technology is dependent on Internet GIS and mobile devices that allow users to access and interact with data at precise locations. Examples of some location-based services are access to real-time traffic information and driving directions from where the user is to locations of nearby businesses and attractions.

Intelligent Transportation Systems

Intelligent Transportation Systems link Internet GIS and real-time traffic information to provide and disseminate real-time travel information to end users. Several local governments have implemented ITS to enable users to plan travel itineraries based on real-time traffic conditions. An example of such a system is the one King County, Washington developed to improve access and usability of its public transportation system. This system provides predictive capabilities for users to find out when the next bus is coming, as well route planning functionality to determine the best way to reach a desired location.

Future Directions

The adoption, migration, and implementation of Internet GIS is filled with exciting possibilities. The hope is that increased accessibility to spatial data and analytical tools will enable scientists to build more realistic models to solve research problems and focus on the domain of the problems instead of the mechanisms of system implementation. The goal of Internet GIS is to encourage and enable the use of geographic information to make more informed decisions. While there has been much progress and research conducted so far, there is much more to do. Coordination and collaboration are needed to more fully develop a data infrastructure that enables easy searches for data and assurances of data quality and accuracy. More fully developed data standards are also so that the concept of seamless data integration can be realized. Finally, more sophisticated compression methods, indexing strategies and processing techniques are needed to successfully deal with the massive amounts of data and the taxing storage and transmission issues that result. The potential of Internet GIS is great and so is the hope that it can improve the quality of everyday human life.

Cross References

► Internet-Based Spatial Information Retrieval
► University of Minnesota (UMN) Map Server

Recommended Reading

Peng, Z.R.: Internet GIS: distributed geographic information services for the Internet and wireless networks. Peng, Z.R., Tsou, M.H. (eds) 1st edn., vol. 36. John Wiley & Sons, Hoboken, NJ, USA (2003)

Shekhar, S., et al.: WMS and GML based Interoperable Web Mapping System. GIS 01. Atlanta, Georgia (2001)

Shekhar, S., and Chawla, S.: Spatial Databases – A Tour. Horton, M. (ed). Pearson Education, Inc., Upper Saddle River, NJ, USA (2003)

Sonebraker, M., Rowe, L.: The Design of PostGRES, vol. 1. Association of Computing Machinery, Berkeley, CA, USA (1986)

Internet Mapping Service

► Web Feature Service (WFS) and Web Map Service (WMS)

Internet-Based Spatial Information Retrieval

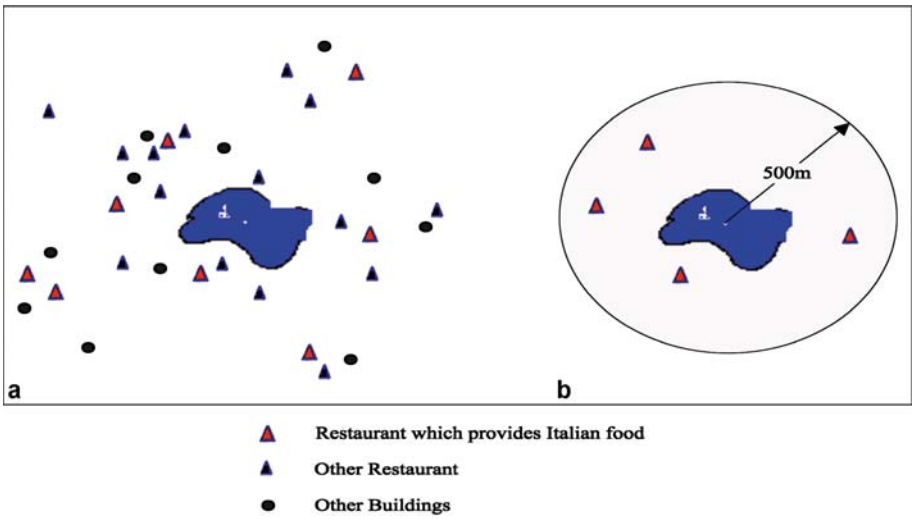
WENWEN LI[1,2], PHIL YANG[1], BIN ZHOU[1,3]
[1] Joint Center for Intelligent Spatial Computing, and Earth System and GeoInformation Sciences, College of Science, George Mason University, Fairfax, VA, USA
[2] Institute of Remote Sensing Applications, Chinese Academy of Sciences, Beijing, China
[3] Department of Electronic Engineering, Tsinghua University, Beijing, China

Synonyms

Information retrieval; Geographical information retrieval; Keyword Search; Text Search

Definition

Spatial information retrieval (SIR) refers to providing access to georeferenced sources by indexing, searching, retrieving, and browsing [1]. SIR is an interdisciplinary topic involving geospatial information science, data mining, computer networks, cognition, and cartography. An example of SIR query could be "Where are the Italian restaurants within 500 meters of a specific park?" Figure 1a illustrates the spatial distribution around the park: the shaded area represents the park; triangles represent restaurants; the red triangles represent those providing Italian food. To answer the question, spatial entities, such as restaurants and the park, are abstracted to a point described in $[x, y]$ coordinates. To execute the query, both an information retrieval algorithm and spatial relationship analysis are needed: (1) locate the area of interest where a circle has



Internet-Based Spatial Information Retrieval, Figure 1
A query example of spatial information retrieval

a radium of 500 m (Fig. 1b), (2) filter out restaurants from other objects (represented by the point in Fig. 1), (3) search the restaurant index to identify the keyword "restaurant types" with "Italian food", (4) send a list of restaurants, ranked by the distance to the park, back to users.

Spatial Data Sources

Spatial retrieval procedures may be categorized by the types of spatial data sources, such as a digital library and the world wide web.

SIR in Digital Library In general, a digital library stores a large volume of categorized information, for example, water resources below ground in a valley for flood analysis and monitoring. Researchers [2] proposed an automatic geospatial information processing system to provide fast access to a digital library. In such a system, Geographic place names (terms) and their attributes are extracted and identified based on a thesaurus and semantic information containing spatial relationships, such as "adjacent to a lake", "south of the river". Geographical coordinates are retrieved and probabilistic weights are assigned to the place names based on their occurrence in the thesaurus. Therefore, each term can be denoted as a three-dimensional (3D) object, dimensions [x, y] represent geographic coverage, and dimension Z represents weight of the term. Finally, all the terms extracted are denoted in 3D space to form a "skyline", where weights are summed when two terms have overlaps in geographical coverage. The geographic area where the peak of the "skyline" located will be indexed. Then, by applying the algorithm to all texts stored in the digital library, the entire index can be established to assist fast access.

SIR in the World Wide Web The context is complex in internet-based SIR because the world wide web (WWW) contains a huge amount of information. For example, as estimated by Danny Sullivan in 2005, Google indexed more than 9 billion documents in its crawler, where the web documents collection is built and maintained through crawling. Meanwhile, performance stood out as an issue for spatial indexing in such a large collection. To support spatial indexing in a large amount of documents, researchers [3] proposed a spatiotextual indexing algorithm with the help of geographical ontology. Each document containing place names is associated with one or more "footprints" (using coordinates to present a place) derived from ontology entries [3]. Then a three-step algorithm is applied: (1) all the documents are divided into several cells (S_i, $i = 1, 2 \ldots m$), based on their spatial distribution marked in the footprint; (2) the document sets ($D_j, j = 1, 2 \ldots n$) indexed by key words are intersected with each space cell (S_i) to form the spatiotextual index (S_i, D_j), because an index with this structure can be exploited by first searching for a textual term; then (3) the associated spatial index of documents is used to filter out those meeting the spatial constraints [3] so that the ambiguity of terms can be eliminated and query accuracy and performance is improved.

Historical Background

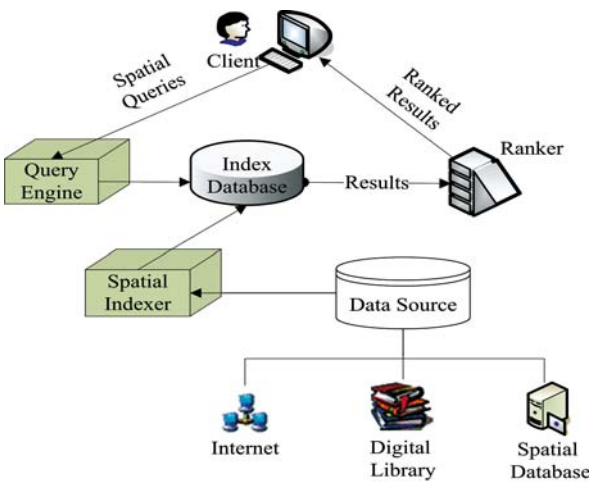
The history of SIR can be traced back to the year 850, when the first print book was created in China that changed the traditional mechanism of information storage. The second leap was in 1946, when the first electronic computer transformed data into a digital version. In 1950, the term "information retrieval" was first used by Vannevar Bush, and became popular [4]. In 1996, Ray R. Lar-

son coined "spatial information retrieval" in the context of a digital library service [1]. Many SIR models have been designed and applied to retrieve geospatial information [5,6].

Scientific Fundamentals

In general, an SIR system deals with spatial queries such as "What is here?" asking for place names, geographical features about a location, or "Where is it?" resulting in a reference in a map [7]. Five types of spatial queries are summarized [1] as: *point-in-polygon query*, which is a relatively precise query asking about the geographic information of a point (denoted by [*x*, *y*]) within a area (denoted by the polygon); *region query*, asking for any geographical element that is contained in, adjacent to or overlaps the region defined; *distance and buffer zone query* (Fig. 1) refers to finding spatial objects that are within a certain distance of an area; *path query*, which requires the presence of a network structure to do shortest-path or shortest-time planning; and *multimedia query*, which combines both georeference processing and nongeoreference processing, such as pattern recognition, in executing a query.

A general SIR model for answering previous spatial queries includes the data source, a spatial indexer, a query engine, and a ranker (Fig. 2). Spatial information is obtained from geospatial data. The data are in different formats, such as textual, numerical, graphical, and multimedia. A spatial indexer provides spatial indexing by extracting geographic locations in a text or mapping data to a term based on a certain geospatial ontology [8,9]. A query engine handles user requests [10]. To provide a better quality of service, a ranker is normally used to sort the results based on match level.



Internet-Based Spatial Information Retrieval, Figure 2 A general model for spatial information retrieval

Geospatial information sources for spatial retrieval are generally available from a digital library or the WWW, where large amounts of data are stored in a variety of formats, including basic text documents, airborne and satellite images, maps from specific geographic locations, and other forms. Therefore, it's of great importance to extract and index the spatial element from data sources to improve query efficiency [11,12]. Meanwhile, the dynamic, incoherent nature of the WWW makes it more difficult for a SIR system to gather information or make spatial indexing structures scalable and efficiently updatable. The solutions to these problems relying on probability theory and spatial reasoning as discussed in "Key Applications".

Key Applications

SIR is widely used in applications ranging from scientific research and government planning to daily life.

Earth and Planetary Research

Terabytes of imagery data are collected through satellite and airborne remote sensors every day [13].While providing valuable resources to earth system research, the imagery also complicates the management of the data. SIR could help to get the data with the appropriate spatial characteristics, time span, and format from a vast number of datasets. Project Sequoia [6] gives a successful example of earth scientists being able to retrieve information from tens of terabytes of spatial and geographical data.

Disaster and Disease Analysis

SIR can assist researchers and policy makers to extract emergency information, for example, the asset losses of cities during a river flooding [14,15].

Urban Transportation

SIR can be applied to urban transportation management by indexing and analyzing the transportation datasets.

Environment Protection

The US Environment Protection Agency (EPA) has its Aerometric Information Retrieval System and national Pesticide Information Retrieval System for viewing and researching the spatial distribution and trends for pollution or other environment destructions [16].

Traveling

SIR can be integrated in mobile devices to help travelers in guidance and route planning [1,17].

Future Directions

Any query related to location needs the support of SIR systems. Future SIR systems will be more intelligent and be able to answer questions given in natural language. Meanwhile, it will play a more important role in contemporary geospatial processing, especially in the context of the WWW.

Cross References

- ▶ Geospatial Semantic Integration
- ▶ Geospatial Semantic Web
- ▶ Geospatial Semantic Web: Applications
- ▶ Geospatial Semantic Web, Interoperability
- ▶ Geospatial Semantic Web: Personalisation
- ▶ Indexing, Hilbert R-tree, Spatial Indexing, Multimedia Indexing

Recommended Reading

1. Larson, R.R.: Geographical information retrieval and spatial browsing. In: Smith, L.C., Gluck, M. (eds.) Geographical Information Systems and Libraries: Patrons, Maps, and Spatial Information, pp. 81–124. Publications Office, Graduate School of Library & Information Science, University of Illinois at Urbana-Champaign, Champaign, IL, USA (1996)
2. Woodruff, A.G., Plaunt, C.: GIPSY: geo-referenced information processing system. J. Am. Soc. Inf. Sci. **45**, 645–655 (1994)
3. Jones, C.B., Abdelmoty, A.I., Finch, D., Fu, G., Vaid, S.: The SPIRIT spatial search engine: architecture, ontologies and spatial indexing. In: Freksa, C., Egenhofer, M.J., Miller, H.J. (eds.) Proceedings of the Third International Conference on Geographic Information Science, Adelphi, MD, USA, 2004. Lecture Notes in Computer Science vol. 3234, pp. 125–139. Springer, USA (2004)
4. Fielden, N.: History of Information Retrieval Systems & Increase of Information Over Time. In: Biennial California Academic & Research Librarians (CARL) Conference, Monterey, CA, US 10 May 2002
5. Jones, C.B.: Spatial information retrieval and geographical ontologies: an overview of the spirit project. In: Proceedings of ACM SIGIR conference on Reasearch and developement in information retrieval, Tampere, Finnland, 11 Aug 2002
6. Chen, J., Larson, R.R., Stonebraker, M.: Sequoia 2000 object browser. In: Digest of Papers, COMPCON Spring 1992. The 37th IEEE Computer Society International Conference, Los Alamitos, US (1992)
7. Oyvind, V.: Geographic Information Retrieval: An Overview. Internal Doctoral Conference, IDI, NTNU, Norway (2004)
8. McCurley, K.S.: Geospatial Mapping and Navigation of the Web. In: Proceedings of the 10th International Conference on the World Wide Web, Hong Kong, 1–5 May 2001
9. Abdelmoty, A.I., Smart, P.D., Jones, C.B., Fu, G., Finch, D.: A critical evaluation of ontology languages for geographical information retrieval. J. Visual Languages Comput. **16**, 331–358 (2005)
10. Arasu, A., Cho, J., Garcia-Molina, H., Paepcke, A., Raghavan, S.: Searching the web. ACM Trans. Internet Technol. **1**, 2–43 (2001)
11. Buyukokkten, O., Cho, J., Garcia-Molina, H., Gravano, L., Shivakumar, N.: Exploiting geographical location information of web pages. In: ACM SIGMOD Workshop on The Web and Databases, Philadelphia, Pennsylvania, USA (1999)
12. Mountain, D., MacFarlane, A.: Geographic information retrieval in a mobile environment: evaluating the needs of mobile individuals. J. Inf. Sci. (2007, in press)
13. Emerson, C.W., Quattrochi, D.A., Lam, N.S.: Spatial metadata for remote sensing imagery. In: Proceedings of the 4th Annual Earth Science Technology Conference (ESTC), Palo Alto, CA, USA (2004)
14. EM-DAT Emergency Disasters Database. http://www.em-dat.net/links/disasterdbs.html. Accesed 20 Aug 2007
15. Lu, W., Mannen, S., Sakamoto, M., Uchida, O., Doihara, T.: Integration of imageries in GIS for disaster prevention support system. In: Proceedings of the XXth ISPRS Congress, Istanbul, Turkey 2004
16. Air Release (AIRS/AFS). http://www.epa.gov/enviro/html/airs/index.html. Accesed 20 Aug 2007
17. Clough, P.: Extracting metadata for spatially-aware information retrieval on the internet. In: Proceedings of the 2005 Workshop on Geographic Information Retrieval, Bremen, Germany 2005
18. Beaujardiere, J.: Web Mapping Service, OGC Implémentation Spécification http://www.portal.opengoespatial.org/files/?artifact_id=14416(2004). Accesed 9 Aug 2007

Interoperability

- ▶ Geography Markup Language (GML)
- ▶ Geospatial Semantic Integration
- ▶ Metadata and Interoperability, Geospatial
- ▶ OGC's Open Standards for Geospatial Interoperability

Interoperability, Technical

- ▶ Metadata and Interoperability, Geospatial

Interoperability, XML Schema

- ▶ Metadata and Interoperability, Geospatial

Interpolation

- ▶ CrimeStat: A Spatial Statistical Program for the Analysis of Crime Incidents

Interpolation of Continous Geofields

► Geosensor Networks, Estimating Continuous Phenonena

Intertial Motion Unit (IMU)

► Evolution of Earth Observation

Inverse Distance Weighting

► Constraint Databases and Data Interpolation

ISO

► Geography Markup Language (GML)
► Spatio-temporal Query Languages

ISO 19115

► Metadata and Interoperability, Geospatial

ISO/IEC

► Oracle Spatial, Geometries

ISO/TC 211

► Application Schema
► deegree Free Software

Isometric Color Bands Displays

► Visualizing Constraint Data