Local Feature Descriptor Indexing for Image Matching and Object Detection in Real-Time Applications

K. Halavataya*

Faculty of Radiophysics and Computer Technologies, Belarusian State University, Minsk, 220030 Belarus

**e-mail: katerina-golovataya@yandex.ru*

Abstract—Local feature extraction and description algorithms can be used to address a large variety of computer vision problems, including pattern recognition, frame stitching and 3D reconstruction. One of the most computationally expensive and time-consuming stages of any method based on local features is keypoint matching. This paper discusses possible ways to optimize matching for frame alignment and object detection applications using Vantage-Point tree indexing to optimize pairwise keypoint matching, and image keypoint graph indexing to optimize pose estimation.

Keywords: computer vision, local feature descriptor, keypoint matching, Vantage-Point trees

DOI: 10.1134/S105466182001006X

INTRODUCTION

Intelligent image analysis algorithms usually rely on extracting meaningful and relevant information from the image. Individual pixel brightness values are often insufficient for any kind of processing, since valuable characteristics of an image are usually associated with a presence or absence of a certain object that can occupy an arbitrary number of pixels, have a different lighting condition, rotation or real to pixel span size ratio (dpi). Thus, most of the analysis is usually performed with some kind of alternate representation of an image that includes all the relevant information while ignoring or remaining invariant to the modifications caused by typical transforms that change individual pixel brightness values, but do not alter the semantics of an image in processing context.

One of the popular ways of creating such a representation is feature extraction. By itself, feature extraction refers to a process of creating some vector-based representation that contains relevant characteristics of an image in the form of features. There are two main types of image feature extraction algorithms – global feature extractors and local feature extractors. Global feature extractors produce a feature vector for the whole image (globally) and are generally used to describe an entire image in some way that can be used as input for supervised or unsupervised machine learning, sometimes also providing a metric for comparing feature vectors of different images. Local feature extraction refers to a process of choosing a subset of points of an image that are somewhat representative of its contents and is generally divided into two separate sub-problems – keypoint detection and keypoint description [1].

One of the key applications of keypoint descriptors and detectors is performing a full-image pairwise keypoint matching – i.e. detecting a set of keypoints on both images, calculating their respective descriptor values, and then finding a suitable pair keypoint on one image for every keypoint detected on the other. Because the number of keypoints across both images can be quite high, full pairwise matching is quite computationally expensive, which directly translates to slower execution. Some problems, like real-time object detection, may heavily rely on a specific computational complexity of algorithms used to solve them, and in such cases, it's necessary to establish a set of optimizations for keypoint matching.

For a singular point of reference image keypoint matching with target image is, essentially, a nearest-neighbor search problem, where the distance between a pair of keypoints is determined by their specific descriptor implementation. The aim of this paper is to research and evaluate different keypoint indexing methods, namely VP-trees and k-d trees, that would allow to perform a single-point match with logarithmic asymptotic complexity instead of linear, in turn reducing multiple point matching complexity to quasilinear, down from quadratic.

Received September 28, 2019; revised October 13, 2019; accepted October 21, 2019

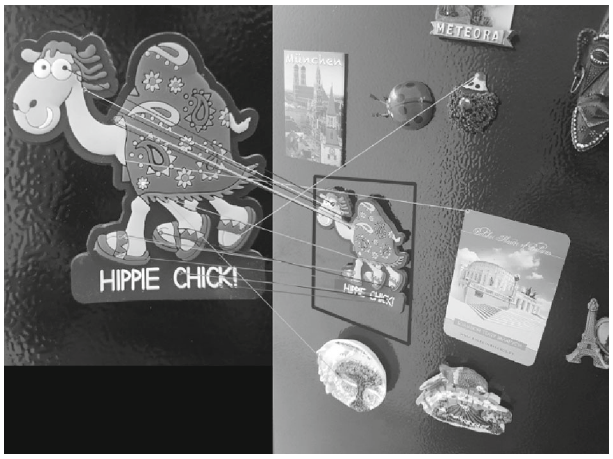

Fig. 1. Specific object detection and homography reconstruction in a scene using keypoint matching.

LOCAL IMAGE FEATURE EXTRACTORS AND DESCRIPTORS

Local keypoint detector is an algorithm that, given an image I of width w and height h produces an unordered set of coordinates $\{(x_i, y_i)\}$ of some of the points that are present on that image:

$$e(I) : \mathbb{M}(w,h) \rightarrow \{(x_i, y_i)\} \subset \overline{(1,w)} \times \overline{(1,h)}. \qquad (1)$$

A keypoint detector is essentially a per-pixel filter that determines whether a particular point is representative enough of a unique characteristic of the image region.

While there is no universal definition as to what constitutes a good keypoint, most of the algorithms rely on detecting sharp corners, edges, smaller blobs of similar color, etc. Nevertheless, detection is assumed to be invariant of typical transforms that individual image objects might undergo. For instance, if a detector Algorithm 1 correctly detects a keypoint at the corner of an object frame, it is generally expected to detect that same corner of the object even if the object itself is moved, rotated, skewed, optically distorted, has different lighting conditions etc.

Keypoint descriptor is an algorithm that produces a unique description vector of a point found by keypoint detector. Some algorithms operate independently and can produce a descriptor for any arbitrary point of the image, while others operate only with a specific detector output. Generally, a descriptor can be expressed as a projection of an image and individual point coordinates into an arbitrary metric space D:

$$d(I,x,y) : \left(\mathbb{M}(w,h) \times \overline{(1,w)} \times \overline{(1,h)}\right) \rightarrow D. \qquad (2)$$

Since descriptor projection in (2) is a metric space, an appropriate square measure should be defined for it:

$$m_d(d_1,d_2) : D^2 \rightarrow R. \qquad (3)$$

The measure (3) can be used to determine a distance between different descriptor vectors d_1, d_2. A keypoint descriptor should be able to produce descriptor values in such a way that descriptors uniquely describe the spatial properties of an individual point and typical object transformations produce a descriptor that is similar, as defined by distance (3).

Individual descriptor values can be thought of as local feature vectors of an image. Unique combinations of relatively distant descriptor vectors can be concatenated into a single global feature vector.

The typical application of keypoint detectors and descriptors is keypoint matching. Given a sufficiently stable detection and description algorithm, it is usually possible to analyze a reference image by detecting and describing its keypoints and then find the same points on another image. For instance, if a known object is used as a reference image, it should be possible to track the location of this object's keypoints on a different image. This can be used to perform specific object detection and correlate the location of points of the same scene across the images depicting the scene from various angles in order to stitch them together (a method widely used in panorama photo reconstruction) or to perform depth estimation and 3D reconstruction.

Popular detector and descriptor implementations include SIFT and SURF histogram detectors and descriptors, FAST detector, BRIEF and FREAK binary descriptors, and ORB – a combination of oriented FAST detector and learned BRIEF binary descriptor [1–3].

An example of keypoint-based specific object detection using ORB is presented on Fig. 1. Lines across the images denote keypoints that matched with each other because their specific descriptor values were similar across the images.

One of the most common distortions when searching for reference objects on a target scene are perspective transforms; thus, a sufficiently stable detector and descriptor must be invariant to rotation and scale of its surrounding areas.

When several points of the reference image match, it is possible to evaluate their respective positioning on the image where the object is detected and produce a homography from the reference image in order to determine exact object position. For our example scene, reconstructed homography bounding box for reference image is displayed as outline on the scene.

One of the main limitations of such an approach to object is the fact that it's not able to generalize – it can only perform exact object detection given its reference image, but cannot use specific object features to form more abstract or general representations of a broader categories, like "book" or "cat".

KEYPOINT MATCHING COMPLEXITY

Keypoint matching across two images is usually performed as the following series of steps:

(1) Extract and describe keypoints on reference image,

(2) Extract and describe keypoints on target image,

(3) For every keypoint on reference image, find a corresponding keypoint on target image using their descriptors in distance (3).

The third step presumes using exhaustive search. For every keypoint of reference image, it is necessary to iterate over every keypoint on the target image, i.e. compare every possible pair of keypoints from reference and target image, in order to locate the point with minimum distance. The number of pairs is generally much larger than the number of individual points – a phenomenon similar to the birthday paradox. The number of all possible pairs asymptotically increases as a square of the number of keypoints detected; that is, in a big-O notation computational complexity of comparing all possible pairs of n keypoints is $O(cn^2)$, where c is the complexity of individual comparison.

Since keypoint comparison is the most used operation, some algorithms optimize the process by introducing a more efficient comparison algorithm, i.e. reducing the complexity c. A class of keypoint descriptors called binary descriptors specifically address the individual comparison complexity problem by using bit strings (fixed-length sequences of only 2 possible values – 0 and 1) as feature vectors. This allows to use Hamming distance – the number of non-matching bits across two bit strings – as distance (3), that can be calculated efficiently using a small number of CPU cycles, as opposed to, for instance, Euclidean distance (L2-norm), that requires calculation of a square root [2, 3]. However, as big-O notation implies, computational complexity of comparison is a constant factor and doesn't depend on the number of points, while asymptotically the running time of an exhaustive search will greatly increase as the number of points increases.

For most applications, it is typically desirable to operate with quite a small subset of keypoints that only contain relevant features. For instance, in our example on Fig. 1 there are around 10 keypoint matches total across both images, each representing a characteristic and unique feature point. In such cases, keypoint matching performance is not much of an issue, and, for some cases, optimizing it for lower number of keypoints may actually increase matching time. However, when using finer-grained features and lower threshold values for keypoint detection, the number of keypoints to match across images may be significant, and case asymptotic complexity of $O(n^2)$ means that performance degrades quite fast as number of points increases.

VANTAGE-POINT TREE INDEXING

The proposed way to enhance keypoint matching performance is usage of indexing techniques. In general, indexing refers to structuring the data in some way that makes searching more computationally efficient than complete traversal. For example, for strict equality search with a partial order relation it's possible to use binary search trees. Each node in binary search tree contains an element and two subtree references, one containing only elements that go before the node element in the partial order relation, and the other containing only elements that go after. When performing a direct comparison search it's possible to compare target node with the next node using partial order relation and choose one of the two subtrees based on this comparison. If the data is distributed evenly across the tree branches (i.e. if the tree is balanced), each comparison effectively reduces the amount of data needed to be processed by half, i.e. the complexity of an individual search is $O(\log n)$, as opposed to $O(n)$ of exhaustive search. The downside of such a search optimization is the need to build the balanced tree every for every target dataset that will be used for comparison.

However, when matching keypoints across images, it's required to find not an exact match, but instead a keypoint that is closest to the target one based on a specific distance. Direct implementation of binary search trees does not allow to use distance measure for comparison, since it can be only implemented based on equality relation for direct match and partial order relation for subtree selection. Therefore, a different structuring algorithm must be used to index the keypoint descriptors based on their relative distance. There are two well-known data structuring algorithms that aim to solve this problem – k-d trees and vantage-point trees.

K-d trees are a generalization of binary search trees to k-dimensional space that separate it into subspaces based on hyperplane division along alternating axes through median point, effectively halving the search space. Hyperplane division produces subspaces within which two points can have a set maximum distance between each other. However, when performing nearest neighbor search, it is usually necessary to scan several subspaces, since target and reference points can, starting from a certain division factor, be placed in multiple different subspaces, since the division uses square regions. This means nearest neighbor search

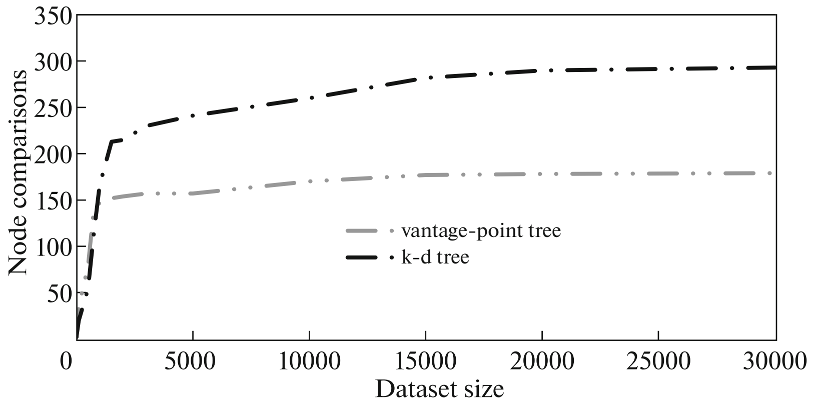

Fig. 2. Comparison of vantage-point tree and k-d tree node comparison counts for different number of points.

performance in k-d trees degrades for higher-dimensional vectors, which is the case for binary descriptors – the typical ones usually have the length of up to 512 bits (64 bytes).

Vantage-point trees create a division based on hyperspheres, with distances defined by an arbitrary metric. Each subtree has a vantage point as its node, a specific metric threshold value and two subtrees. The first subtree contains nodes for which distance from vantage point is smaller than node threshold (i.e. all points that are within hypersphere with threshold as its radius), and the second subtree contains remaining points. Vantage point and metric threshold are selected in such a way as to partition the target space into subspaces with roughly equal amount of points. During vantage point construction it is possible to record the lowest encountered node distance (the metric threshold for tree leaf nodes), so for nearest neighbor search it is necessary to only scan subspaces that may contain points as far as the lowest node distance from the target point; in best-case scenario it will require a single subspace scan [4].

Using arbitrary metric has an advantage over k-d trees in the fact that hypersphere subspace division will produce lower number of subspaces to scan for finer-grained nodes than hyperplane division, which makes this algorithm perform significantly better on high-dimensional spaces. To asses running time for k-d trees and vantage-point trees for nearest neighbor search, node comparison count was evaluated on a sample keypoint matching problem using ORB 32-byte (256 bit) descriptor. The results of the comparison are presented on Fig. 2. The analysis shows that vantage-point tree performs a little worse for small sample sizes (up to 500 points), but scales much better when dataset size increases.

For keypoint matching problem, it is possible to use distance measure (3) as a metric for vantage-point tree indexing, provided it is a well-formed metric. A metric, as opposed to a distance measure, must satisfy 4 conditions: non-negativity, identity of indiscernible values, symmetry and triangle inequality. Non-negativity means that metric value for any pair of input vectors is always greater than or equal to zero. Indiscernible value identity means that distance between equal values, as determined by the metric, is always zero, and that zero distance as determined by the metric means that input values are equal. Symmetry means that order of the values in metric arguments doesn't matter and always produces the same distance. Finally, triangle inequality states that distance between a pair of points x and y is less than or equal the sum of distances from x to any other point z and from that point z to y, i.e. $m_d(x,y) \leq m_d(x,z) + m_d(z,y)$. Some popular distance measures used in machine learning applications, like squared Euclidean distance, cannot be used for vantage-point trees, since they are not a well-formed metric and do not satisfy some of those conditions; for instance, triangle inequality is not satisfied by squared Euclidean. Hamming distance, used to compare binary descriptors, is a well-formed metric and can be used to produce vantage-point trees.

For keypoint matching problem, it is possible to use distance measure (3) as a metric for vantage-point tree indexing, provided it is a well-formed metric. A metric, as opposed to a distance measure, must satisfy 4 conditions: non-negativity, identity of indiscernible values, symmetry and triangle inequality. Non-negativity means that metric value for any pair of input vectors is always greater than or equal to zero. Indiscernible value identity means that distance between equal values, as determined by the metric, is always zero, and that zero distance as determined by the metric means that input values are equal. Symmetry means that order of the values in metric arguments doesn't matter and always produces the same distance. Finally, triangle inequality states that distance between a pair of points x and y is less than or equal the sum of distances from x to any other point z and from that point z to y, i.e. $m_d(x,y) \leq m_d(x,z) +$

$m_d(z, y)$. Some popular distance measures used in machine learning applications, like squared Euclidean distance, cannot be used for vantage-point trees, since they are not a well-formed metric and do not satisfy some of those conditions; for instance, triangle inequality is not satisfied by squared Euclidean. Hamming distance, used to compare binary descriptors, is a well-formed metric and can be used to produce vantage-point trees.

Vantage point nearest neighbor search has an $O(\log n)$ complexity with respect to the number of points to compare with, which means that keypoint matching across images has a complexity of $O(n \log n)$, which is significantly faster than $O(n^2)$ for full exhaustive search matching and scales much better with increase of keypoints to compare, for instance, when using images of higher resolution. One of the disadvantages of such an approach is the fact that vantage-point tree construction based on a set of keypoints from one of the images is itself a $O(n \log n)$ complexity task, i.e. indexing an image incurs an overhead for index calculation [4]. This makes such indexing suitable for matching a single reference images with multiple others, as is the case, for instance, for specific object detection tasks.

IMAGE INDEXING USING KEYPOINT GRAPH

When performing specific object search across multiple images it may also be necessary to introduce a measure of similarity across different image types. Moreover, object detection based on keypoint relative positioning and their respective descriptor values allows for independent template-based object description that can be used to index specific images. An appropriate representation of keypoint structure can be used not only to detect specific objects, but also to implement image similarity measure, image indexing and similar image lookups.

The proposed method for keypoint-based image indexing uses the representation of detected image keypoints as a graph. The vertices of the graph are detected keypoints, while arcs are established between keypoints based on their mutual position and distance. Two vantage-point trees are created to index image points. The first tree is used to evaluate point position and is created using point coordinates on the image as values and Euclidean distance as distance metric. The second tree is created based on keypoint descriptor distance, as discussed previously. To establish the arcs in keypoint graph, positional vantage-point tree is traversed post-order, and sequential traversed nodes are connected with an edge. The first established edges are placed between keypoints that are situated close to each other on the image, while the several final edges will correspond to point cloud centers. These edges can be used to cluster the points based on their position.

After initial graph composition, it is incrementally simplified by replacing pairs of points that have similar descriptor values, as determined by descriptor distance vantage-point tree, with a single point with averaged-out descriptor value, using post-order traversal of descriptor distance tree, but only for points where an arc exists in composed keypoint graph; points where no arc exists that are spatially close enough to each other are instead connected with an arc. This eliminates groups of similar keypoints that describe the same feature.

The remaining graph is then partitioned into several subgraphs using Kernighan-Lin algorithm until a sufficient simplicity is achieved. Graph vertices retain their respective descriptor values [4].

A graph isomorphism can be approximated to compare one simplified graph to another. For sufficiently simplified graphs (10–15 vertices) direct node permutation comparison can be performed to determine their similarity, that can be further quantified by aggregate corresponding keypoints distance. This sum of distances can be used, in turn, as distance measure between two graphs; two keypoint graphs reconstructed from a single image using the presented algorithm can be used to evaluate respective image pair similarity. The limitation of proposed method is its requirement to calculate descriptor distance vantage-point tree; however, in common object detection tasks based on keypoint correspondence, keypoint vantage-point tree may already be available as a side-effect of real time detection optimization.

CONCLUSIONS

Local feature extractors and descriptors can be used in various important computer vision problems, including object detection and image matching. The paper discussed the way to optimize keypoint matching of detector and descriptor algorithms by building an index of described keypoints. K-d trees and vantage-point trees were evaluated and compared; vantage-point trees were shown to have a better performance on high-dimensional descriptors, like most of the binary descriptors. Resulting keypoint indices can be used to speed up pairwise keypoint matching, requiring only logarithmic time to find a match, as opposed to linear-time full scan.

Created vantage-point trees can be further utilized to create a spatially-aligned keypoint representation in the form of keypoint graph, that combines keypoint relative positioning information with their specific descriptor values. Simplified keypoint graphs can be used to index individual images and to determine image similarity.

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

1. S. Krig, *Computer Vision Metrics*: *Survey, Taxonomy, and Analysis* (Apress, Berkeley, CA, 2014).
2. D. G. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vision **60** (2), 91–110 (2004).
3. E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. 2011 IEEE Int. Conf. on Computer Vision (ICCV 2011)* (Barcelona, Spain, 2011), IEEE, pp. 2564-2571.
4. P. N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," in Proc. *4th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA'93*) (Austin, TX, USA, 1993) (SIAM, Philadelphia, PA, 1993), pp. 311–321.

Halavataya Katsiaryna. Date of birth: 21.07.1993. Education: bachelor honors degree (grad. 2015), master's degree (grad. 2016), currently PhD student. Affiliation: Belarusian State University, Faculty of Radiophysics and Computer Technologies, Intelligent Systems dept.: Position: Sr. lecturer. Area of research: Computer vision, 3D reconstruction from images, machine learning and deep learning, computer graphics, virtual and augmented reality. Number of publications: 30 articles and conference proceedings materials.