Maximal Clique Enumeration in Finding Near Neighbourhoods*

Christopher J. Henry and Sheela Ramanna

University of Winnipeg, Department of Computer Science,
Winnipeg, Manitoba R3B 2E9, Canada
{ch.henry,s.ramanna}@uwinnipeg.ca

Abstract. The problem considered in this article stems from the observation that practical applications of near set theory require efficient determination of all the tolerance classes containing objects from the union of two disjoints sets. Near set theory consists in extracting perceptually relevant information from groups of objects based on their descriptions. Tolerance classes are sets where all the pairs of objects within a set must satisfy the tolerance relation and the set is maximal with respect to inclusion. Finding such classes is a computationally complex problem, especially in the case of large data sets or sets of objects with similar features. The contributions of this article are the observation that the problem of finding tolerance classes is equivalent to the MCE problem, empirical evidence verifying the conjecture from [15] that the extra perceptual information obtained by finding all tolerance classes on a set of objects obtained from a pair of images improves the CBIR results when using the tolerance nearness measure, and a new application of MCE to CBIR.

Keywords: Near sets, maximal clique enumeration, tolerance near sets, tolerance space, tolerance relation, pre-class, nearness measure, CBIR.

1 Introduction

The problem considered in this article is one of finding all the tolerance classes on a set of objects. In the proposed application to content-based image retrieval (CBIR) [36], classes in image covers determined by a tolerance relation provide the content used in CBIR and a feature-based tolerance space solution to detecting and measuring similarities in digital images. Specifically, the tolerance classes represent the extracted perceptual information which is used in quantizing the nearness of sets. The notion of nearness in mathematics and the more general notion of resemblance that is a dominant part of CBIR can be traced back to J.H. Poincaré [32]. Our approach stems from a recent extension of J.H. Poincaré's

* This research has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) grants 194376 and 418413. Also, special thanks to Tariq Alusaifeer for recognizing the problem of finding tolerance classes is equivalent to maximal clique enumeration.

representative spaces, tolerance spaces [40,37,31] and near sets introduced by J.F. Peters in 2007 [26,27], and elaborated in [28,30,7,29,11,39,9,33].

Tolerance classes are sets where all the pairs of objects within a set must satisfy the tolerance relation and the set is maximal with respect to inclusion. Finding such classes is a computationally complex problem, especially in CBIR involving sets of objects with similar features [11,13,15,12,10,14]. Previous work into finding tolerance classes was based on the observation that all tolerance classes containing an object are a subset of the neighbourhood of that object [11,13]. Reported algorithms include a serial approach for finding most tolerance classes using the Fast Library for Approximate Nearest Neighbours (FLANN) [11,13], and a parallel computing approach for finding all tolerance classes using NVIDIA's Compute Unified Device Architecture (CUDA) Graphics Processing Unit (GPU) [15]. This article presents a new solution to the problem of finding tolerance classes by observing that this problem can be mapped to the Maximal Clique Enumeration (MCE) problem. Consequently, the classes can be found using an algorithm with reduced complexity based in graph theory. In addition, the MCE approach to performing CBIR introduced in this article is compared with the well known Earth Movers Distance and Integrated Region Matching in [16]. Finally, the parallel approach is not considered in this article since the runtimes of the serial and MCE algorithms are more than 10 times faster.

The article is organized as follows: First, Section 2 introduces tolerance classes by way of near set theory, providing the context in which tolerance classes are used in this article. Next, previously reported algorithms for find tolerance classes are given in Section 3. Section 4 provides a brief review of the problem of MCE. Then, a discussion on the multitreaded implementation of each algorithm is given in Section 5. Section 6 defines tolerance near sets and presents the nearness measure used to perform CBIR. Finally, Section 7 presents the results and discussion. The contributions of this article are the observation that the problem of finding tolerance classes is equivalent to the MCE problem, empirical evidence verifying the conjecture from [15] that the extra perceptual information obtained by finding all tolerance classes on a set of objects obtained from a pair of images improves the CBIR results when using the tolerance nearness measure, and a new application of MCE to CBIR.

2 Tolerance Classes

Disjoint sets containing objects with similar descriptions are near sets. Similarity is determined quantitatively via some description of the objects. Near set theory provides a formal basis for identifying, comparing, and measuring resemblance of objects based on their descriptions, *i.e.* based on the features that describe the objects. The discovery of near sets begins with identifying feature vectors for describing and discerning affinities between sample objects. Objects that have, in some degree, affinities in their features are considered *perceptually near* each other. Groups of these objects, extracted from the disjoint sets, provide information and reveal patterns of interest.

Tolerance near sets are near sets defined by a description-based tolerance relation. Tolerance relations provide a view of the world without transitivity [37]. Consequently, tolerance near sets provide a formal foundation for *almost solutions*, solutions that are valid within some approximation, which is required for real world problems and applications [37]. In other words, tolerance near sets provide a basis for a quantitative approach for evaluating the similarity of objects without requiring object descriptions to be exact.

Let us begin with defining the content of the sets. All sets in near set theory consist of perceptual objects, which is anything in the physical world with characteristics observable to the senses such that they can be measured and are knowable to the mind. A feature characterizes some aspect of the makeup of a perceptual object. A probe function is a real-valued function representing a feature of a perceptual object [26]. In the context of near set theory, objects in our visual field are always presented with respect to the selected probe functions, which is in keeping with the approach to pattern recognition suggested by M. Pavel [23] where the features of an object are quantified by probe functions. In other words, probe functions are used to measure characteristics of visual objects and similarities among perceptual objects.

A perceptual system is a set of perceptual objects, together with a set of probe functions, *i.*e. a perceptual system $\langle O, \mathbb{F} \rangle$ consists of a non-empty set O of sample perceptual objects and a non-empty set $\mathbb{F}$ of real-valued functions $\phi \in \mathbb{F}$ such that $\phi : O \to \mathbb{R}$ [30]. The notion of a perceptual system admits a wide variety of different interpretations that result from the selection of sample perceptual objects contained in a particular sample space O. Two examples of perceptual systems are: a set of images together with a set of image processing probe functions, or a set of results from a web query together with some measures (probe functions) indicating, *e.g.*, relevancy or distance (*i.e.* geographical or conceptual distance) between web sources. The description of a perceptual object within a perceptual system can be defined as follows. Let $\langle O, \mathbb{F} \rangle$ be a perceptual system, and let $\mathcal{B} \subseteq \mathbb{F}$ be a set of probe functions. Then, the description of a perceptual object $x \in O$ is a feature vector given by

$$\boldsymbol{\phi}_{\mathcal{B}}(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_i(x), \ldots, \phi_l(x)),$$

where l is the length of the vector $\boldsymbol{\phi}_{\mathcal{B}}$, and each $\phi_i(x)$ in $\boldsymbol{\phi}_{\mathcal{B}}(x)$ is a probe function value that is part of the description of the object $x \in O$. Note, the idea of a feature space is implicitly introduced along with the definition of object description. An object description is the same as a feature vector as described in traditional pattern classification [5], yet different from the signature of an object defined in [24] (due to the use of features instead of attributes[1]). The description of an object can be considered a point in an l-dimensional Euclidean space $\mathbb{R}^l$ called a feature space. Thus, the relationship between objects is discovered in a feature space that is determined by the probe functions in $\mathcal{B}$.

Formally, a tolerance space can be defined as follows [40,37,31]. Let O be a set of sample perceptual objects, and let ξ be a binary relation (called a tolerance

[1] See, [25,27,39] for a discussion on the difference between features and attributes.

relation) on X ($\xi \subset X \times X$) that is reflexive (for all $x \in X$, $x\xi x$) and symmetric (for all $x, y \in X$, if $x\xi y$, then $y\xi x$) but transitivity of ξ is not required. Then a tolerance space is defined as $\langle X, \xi \rangle$. Considering the tolerance space definition, a specific tolerance relation [28,29] (see [7,8] for applications in image analysis) is given as follows. Let $\langle O, \mathbb{F} \rangle$ be a perceptual system and let $\varepsilon \in \mathbb{R}_0^+$. For every $\mathcal{B} \subseteq \mathbb{F}$, the perceptual tolerance relation $\cong_{\mathcal{B},\varepsilon}$ is defined by:

$$\cong_{\mathcal{B},\varepsilon} = \{(x, y) \in O \times O : \parallel \phi(x) - \phi(y) \parallel_2 \leq \varepsilon\},$$

where $\parallel \cdot \parallel_2$ is the L^2 norm.

Finally, the algorithms presented in Section 3 are based on the propositions involving neighbourhoods and tolerance classes. Formally, these concepts are defined as follows. Let $\langle O, \mathbb{F} \rangle$ be a perceptual system and let $x \in O$. For a set $\mathcal{B} \subseteq \mathbb{F}$ and $\varepsilon \in \mathbb{R}_0^+$, a neighbourhood is defined as

$$N(x) = \{y \in O : x \cong_{\mathcal{B},\varepsilon} y\}.$$

Note, all objects satisfy the tolerance relation with a single object in a neighbourhood. In contrast, all the pairs of objects within a pre-class must satisfy the tolerance relation. Thus, let $\langle O, \mathbb{F} \rangle$ be a perceptual system. For $\mathcal{B} \subseteq \mathbb{F}$ and $\varepsilon \in \mathbb{R}_0^+$, a set $X \subseteq O$ is a pre-class iff $x \cong_{\mathcal{B},\varepsilon} y$ for any pair $x, y \in X$. Similarly, a maximal pre-class with respect to inclusion is called a tolerance class.

3 Neighbourhood-Based Algorithms

The serial approach [11,13] and the parallel approach [15] to finding tolerance classes are both based on the propositions (proved in [11]) that all tolerance classes containing $x \in O$ are subsets of the neighbourhood of x, $N(x)$, and that tolerance classes are formed from the query points of successive neighbourhoods, *i*.e. from finding neighbourhoods within neighbourhoods. An illustrative example of the propositions central to these algorithms is given in Fig. 1, Fig. 1(a) gives a tolerance class from within a neighbourhood, Fig. 1(b) shows $N(20)$ obtained using only objects from $N(1)$, and Fig. 1(c) shows successive neighbourhoods using the objects within grey region as query points.

The serial approach attempted to mitigate the computational complexity of finding tolerance classes by using FLANN searches to find neighbourhoods, as well as a simple heuristic to reduce runtime. As a result, the serial approach produced found most (but not all) tolerance classes (see, *e.g.* [11]). Algorithm 1 gives the serial approach to finding tolerance classes, where *compsub* is list of the objects along the search path (*i.e.*, the objects in the grey region of Fig. 1(c)), and *cand* is a list objects that are not in *compsub* but satisfy the tolerance relation with every object in *compsub*. Notice the similarity of this approach to Algorithm 2, a similarity that was discovered independently of the body of literature devoted to the MCE problem [2,4]. Note, the variable names in Algorithm 1 were introduced here to maintain notational consistency with the algorithm reported in [34]. Lastly, Algorithm 1 produces duplicate classes. Consequently, at

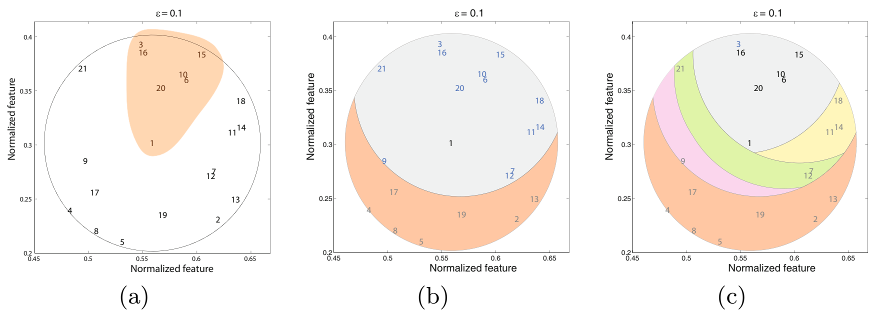


Fig. 1. Algorithm foundational ideas: a) Neighbourhood $N(1)$ in 2D feature space and tolerance class shown in orange, b) $N(20)$ found using only objects from $N(1)$, and c) series of successive neighbourhoods leading to the tolerance class depicted in (a), *i.e.* $N(3) \subset N(16) \subset N(15) \subset N(6) \subset N(10) \subset N(20)$.

Algorithm 1: Serial algorithm for finding tolerance classes

```
Input  : Set of objects $O$
Output: Set of tolerance classes $H_{\cong_{\mathcal{B},\varepsilon}}(O)$
1 for $x \in O$ do
2     for $y \in N(x)$ do
3         $compsub \leftarrow \{x, y\}$;
4         $cand \leftarrow$ All objects in $N(x)$ that satisfy $\cong_{\mathcal{B},\varepsilon}$ with $y$;
5         GenerateRemaining($y$, $compsub$, $cand$);
```

Procedure GenerateRemaining(y, *compsub*, *cand*)

```
1 if $cand = \{\}$ then
2     Output $compsub$
3 else
4     $new\_y \leftarrow$ Object in $N(y)$ that is closest to $y$;
5     $new\_cand \leftarrow$ All objects in $N(y)$ that satisfy $\cong_{\mathcal{B},\varepsilon}$ with $new\_y$;
6     $new\_cs \leftarrow compsub \cup new\_y$;
7     GenerateRemaining($new\_y$, $new\_cs$, $new\_cand$);
```

some point in the algorithm, the duplicate classes must be removed. In the case of Algorithm 1, this step is performed after the for loop in line 1 has completed.

Next, the parallel approach depends on CUDA GPU programming [21,22]. Briefly, a GPU consists of hundreds of cores (processors) and these cores are organized into groups call streaming multiprocessors that are capable of running code that is called the kernel. The abstraction used to execute this code is called a thread. To make full use of the GPU's stream processors one must generate 1000s of threads for execution. The parallel algorithm reported in [15] consists of the following three stages: Finding object neighbourhoods, finding pseudo neighbourhoods, deleting duplicate classes and subsets. While the first and last step are self explanatory, the process of finding pseudo neighbourhoods consists of initializing sets as neighbourhoods and, during each iteration of the loop, these sets approach tolerance classes. This structure facilated the GPU implementation and combined the *compsub* and *cand* sets described above, where the loop iterator indicates the boundary between the two. Note, only the second stage is executed on the GPU.

While this algorithm finds all the classes, it suffers from prohibitive runtimes. As reported, using $\varepsilon = 0.15$, the runtime for a single pair of images is 10 seconds. To generate the results in this article, tolerance classes need to be generated for 405,450 image pairs, giving a runtime of almost 47 days. As a result, the parallel approach is not considered in this article since the runtimes of the serial and MCE algorithms are 500 and 100 ms, respectively. Finally, note, a GPU approach to MCE is reported in [17], however, Jenkins, *et al.* report an inability to provide good performance against MCE algorithms that are non-GPU based.

4 Maximal Clique Enumeration Algorithm

The Maximal Clique Enumeration (MCE) problem consists of finding all maximal cliques among an undirected graph. Briefly, let $G = (V, E)$ denote an undirected graph, where V is a set of vertices and E is set of edges that connect pairs of distinct vertices from V. A clique is a set of vertices where each pair of vertices in the clique is connected by an edge in E. A maximal clique in G is a clique whose vertices are not all contained in some larger clique, *i.e.* there is no other vertex that is connected to all the vertices in the clique by edges in E.

The first serial algorithm for MCE was developed by Harary and Ross [6,2]. Since then, two main approaches have been established to solve the MCE problem [4], namely the greedy approach reported by Bron-Kerbosh [3] (and concurrent discovery by E. Akkoyunlu [1]), and output-sensitive approaches such as those in [38,19]. The implementation of the MCE algorithm used to generate the results in this paper is a modification of the one reported in [34], which is scalable and parallel version of the the Bron-Kerbosh approach. The Bron-Kerbosh algorithm is given in Algorithm 2 (again, we are using the same notation as in [34]). The general idea is to use a tree structure to find all maximal cliques, where each call to *CliqueEnumerate* creates a new child node. Each node in the tree consists of four items: the current vertex used to make decisions (*cur_v*), a list of vertices consisting of the (non-maximal) clique formed up to the current node in the tree (*compsub*), a list of potential vertices that are connected to every vertex in *compsub* (*cand*), and a list of vertices that are connected to every vertex in *compsub*, but, if followed, constitute a redundant path in the search tree. Notice, in terms of the neighbourhood-based algorithms, that, at any given level in the tree, *new_cand* is the neighbourhood of *cur_v* using only objects in the list *cand*. Also, similar to the neighbourhood-based approach, both algorithms stop when there are no candidates left to process. Finally, a connection predicate is necessary for the repeated decisions on whether edges exist between vertices [34].

Options include: A linear search of a linked list of adjacent vertices, a lookup using an adjacency bit matrix, and a lookup using a hash table of edges. Since the number of objects generated from each image pair is small (456), the adjacency matrix was used since it is the fastest. The adjacency matrix was constructed by creating a $|V| \times |V|$ matrix, where a 1 (resp. 0) at position i, j represents the existence (non-existence) of an edge between the vertices v_i and v_j.

Algorithm 2: The BK algorithm

```
Input  : A graph $G$ with vertex $V$ and edge set $E$
Output: MCE for graph $G$
1 $compsub \leftarrow \{\}$;
2 $cand \leftarrow V$;
3 $not \leftarrow \{\}$;
4 CliqueEnumerate($compsub$, $cand$, $not$);
```

5 Multithreading Approach

As was mentioned, [34] presents a scalable and parallel, multi-threaded approach to solving the MCE problem. The algorithm is parallel in two different aspects. First, their algorithm generates multiple processes that communicate using the

Procedure CliqueEnumerate(*compsub*, *cand*, *not*)

```
1  if $cand = \{\}$ then
2      if $not = \{\}$ then
3          Output $compsub$
4  else
5      $fixp \leftarrow$ The vertex in $cand$ that is connected to the greatest number of other
       vertices in $cand$;
6      $cur\_v \leftarrow fixp$;
7      while $cur\_v \neq NULL$ do
8          $new\_not \leftarrow$ All vertices in $not$ that are connected to $cur\_v$;
9          $new\_cand \leftarrow$ All vertices in $cand$ that are connected to $cur\_v$;
10         $new\_cs \leftarrow compsub \cup cur\_v$;
11         CliqueEnumerate($new\_cs$, $new\_cand$, $new\_not$);
12         $not \leftarrow not \cup cur\_v$;
13         $cand \leftarrow cand \setminus cur\_v$;
14         if there is a vertex $v$ in $cand$ that is not connected to $fixp$ then
15             $cur\_v \leftarrow v$;
16         else
17             $cur\_v \leftarrow NULL$;
```

Message Passing Interface (MPI), allowing their algorithm to run on a wide variety of parallel and networked computers. Second, each process generates multiple threads. To simplify the implementation, our results were generated using a single process with multiple threads since the amount of objects obtained from a pair of images in our experiments is 456, compared to the test sets used by Schmidt *et. al* in which the number of objects (vertices) range from 3,472 to 193,568. In fact, both the MCE and neighbourhood-based algorithms used a multi-thread approach to obtain results. The neighbourhood-based approach consisted of creating a stack of object to be processed. Then, each thread pops an object from the stack and finds all the tolerance classes containing that object. Thus, in the multi-threaded algorithm each thread runs an instance of Algorithm 1, except x is obtained from the stack in line 1 (rather than looping through all the objects in O). The MCE algorithm also uses a stack of structure. In this case, it contains the nodes in the tree and each thread process a single node at a time. The modified version of the algorithm in [34] is given in Algorithm 3.

Algorithm 3: The Multi-threaded BK algorithm

```
  Input  : A graph $G$ with vertex $V$ and edge set $E$
  Output: MCE for graph $G$
1 for $i = 0; i <$ num_threads; $i$++ do
2     Spawn thread $T_i$;
3     Have $T_i$ run MCliqueEnumerate();
4 Wait for threads to finish processing;
```

6 Quantifying Nearness

The following two definitions enunciate the fundamental notion of nearness between two sets and provide the foundation for applying near set theory to the problem of CBIR.

Definition 1 Tolerance Nearness Relation [28,29]. *Let $\langle O, \mathbb{F} \rangle$ be a perceptual system and let $X, Y \subseteq O, \varepsilon \in \mathbb{R}_0^+$. A set X is near to a set Y within the perceptual system $\langle O, \mathbb{F} \rangle$ ($X \underline{\underline{\bowtie}}_{\mathbb{F}} Y$) iff there exists $x \in X$ and $y \in Y$ and there is $\mathcal{B} \subseteq \mathbb{F}$ such that $x \cong_{\mathcal{B},\varepsilon} y$.*

Definition 2 Tolerance Near Sets [28,29]. *Let $\langle O, \mathbb{F} \rangle$ be a perceptual system and let $\varepsilon \in \mathbb{R}_0^+, \mathcal{B} \subseteq \mathbb{F}$. Further, let $X, Y \subseteq O$, denote disjoint sets with coverings determined by the tolerance relation $\cong_{\mathcal{B},\varepsilon}$, and let $H_{\cong_{\mathcal{B},\varepsilon}}(X), H_{\cong_{\mathcal{B},\varepsilon}}(Y)$ denote the set of tolerance classes for X, Y, respectively. Sets X, Y are* tolerance near sets *iff there are tolerance classes $A \in H_{\cong_{\mathcal{B},\varepsilon}}(X), B \in H_{\cong_{\mathcal{B},\varepsilon}}(Y)$ such that $A \underline{\underline{\bowtie}}_{\mathbb{F}} B$.*

Procedure MCliqueEnumerate

```
1  foreach vertex $v_i$ assigned to the thread do
2      $cp \leftarrow$ New candidate path node structure for $v_i$;
3      for $v_j \in V$ do
4          if connected$(v_i, v_j)$ then
5              if $i < j$ then
6                  Vertex $v_j$ is in $cp$'s cand list;
7              else
8                  Vertex $v_j$ is in $cp$'s not list;
9      Push $cp$ onto shared stack;
10 while shared stack is not empty do
11     $cur \leftarrow$ Pop a candidate path node structure from stack;
12     if $cur$'s cand and not lists are empty then
13         Output $cur$'s compsub
14     else
15         Generate all $cur$'s children (create child nodes and push onto stack);
```

Observe that two sets $X, Y \subseteq O$ are tolerance near sets, if they satisfy the tolerance nearness relation.

The tolerance nearness measure was created out of a need to determine the degree that near sets resemble each other, a need which arose during the application of near set theory to the practical applications of image correspondence (see, *e.g.* [7,11]). The tolerance nearness measure between two sets X, Y is based on the idea that tolerance classes formed from objects in the union $Z = X \cup Y$ should be evenly divided among X and Y if these sets are similar, where similarity is always determined with respect to the selected probe functions. The tolerance nearness measure is defined as follows. Let $\langle O, \mathbb{F} \rangle$ be a perceptual system, with $\varepsilon \in \mathbb{R}_0^+$, and $\mathcal{B} \subseteq \mathbb{F}$. Furthermore, let X and Y be two disjoint sets and let $Z = X \cup Y$. Then a tolerance nearness measure between two sets is given by

$$tNM_{\cong_{\mathcal{B},\varepsilon}}(X,Y) = \\ 1 - \left(\sum_{C \in H_{\cong_{\mathcal{B},\varepsilon}}(Z)} |C| \right)^{-1} \cdot \sum_{C \in H_{\cong_{\mathcal{B},\varepsilon}}(Z)} |C| \frac{\min(|C \cap X|, |[C \cap Y|)}{\max(|C \cap X|, |C \cap Y|)}. \quad (1)$$

Finally, new measures inspired by the tNM have been reported in [35,20]. A systematic comparison of the tNM and these measures is outside the scope of this paper and is left for future work.

7 Results and Discussion

The algorithms presented here are compared using CBIR, where the goal is to retrieve images from databases based on the content of an image rather than on some semantic string or keywords associated with the image. The content of the image is determined by functions that characterize features such as colour, texture, shape of objects, and edges. In our approach to CBIR, a search entails analysis of content, based on the tNM nearness measure (see, *e.g.* [11]) between a query image and test image. Moreover, the nearness measure on tolerance classes of objects derived from two perspective images provides a quantitative approach for accessing the similarity of images. To generate our results, the SIMPLIcity image database [18], a database of images containing 10 categories with 100 images in each category was used (shown in Fig. 2).

The results were generated by partitioning the images into subimages, where each subimage was considered as an object in the near set sense, *i.e.* each subimage is a perceptual object, and each object description consists of the values obtained from image processing techniques on the subimage. This technique of partitioning an image, and assigning feature vectors to each subimage is an approach that has also been traditionally used in CBIR. Formally, an RGB

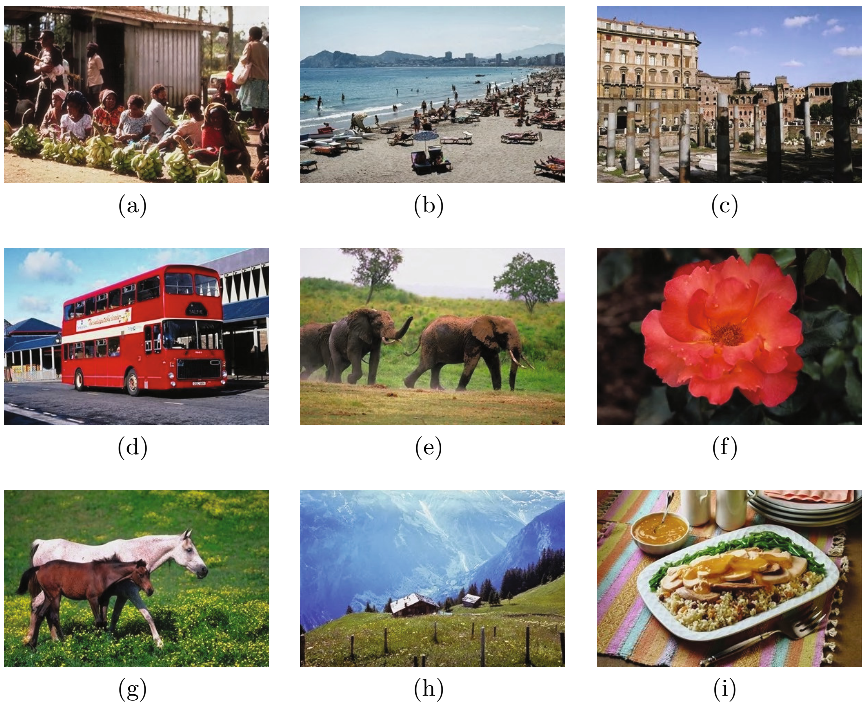


Fig. 2. Examples of each category of images. (a) - (d) Categories 0 - 3, and (e) - (i) categories 5 - 9.

image is defined as $f = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_T\}$, where $\mathbf{p}_i = (c, r, R, G, B)^{\mathrm{T}}$, $c \in [1, M]$, $r \in [1, N]$, $R, G, B \in [0, 255]$, and M, N respectively denote the width and height of the image and $M \times N = T$. Further, define a square subimage as $f_i \subset f$ such that $f_i \cap f_j = \{\}$ for $i \neq j$ and $f_1 \cup f_2 \ldots \cup f_s = f$, where s is the number of subimages in f. Next, O can be defined as the set of all subimages, *i.e.*, $O = \{f_1, \ldots, f_s\}$, and $\mathbb{F}$ is a set of image processing descriptors or functions that operate on images. Then, the nearness of two images can be discovered by partitioning each of the images into subimages and letting these represent objects in a perceptual system, *i.e*, let the sets X and Y represent the two images to be compared where each set consists of the subimages obtained by partitioning the images. Then, the set of all objects in this perceptual system is given by $Z = X \cup Y$.

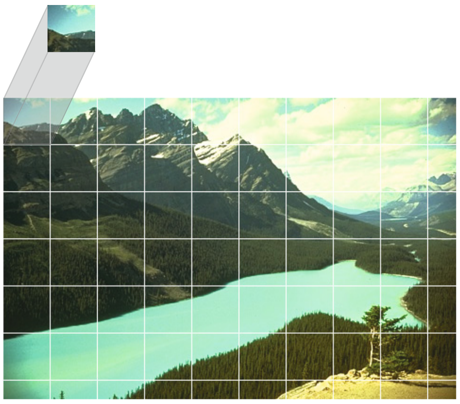


Fig. 3. Example demonstrating the application of near set theory to images, namely the image is partitioned into subimages where each subimage is considered a perceptual object, and object descriptions are the results of image processing techniques on the subimage

The results in this article were obtained using a subimage size of 20×20 (resulting in 456 objects per image pair) and the 18 features used in [11], namely 4 texture features obtained from the grey-level co-occurrence matrix of a subimage, the first and second moments of u and v in the CIELUV colour space, an edge based feature, and the Zernike moments of order 4, excluding $\widetilde{A}_{00}$. Moreover, the results are presented using precision vs. recall plots, where the idea is to find tNM values between each pair of images in the database. Then, the measure values are sorted in ascending order, and the smallest value represents the results of the first query, the second value the results of the second query, *etc.* Precision/recall plots are the common metric for evaluating CBIR systems where precision and recall are defined as

$$\text{precision} = \frac{|\{\text{relevant images}\} \cap \{\text{retrieved images}\}|}{|\{\text{retrieved images}\}}\,,$$

and

$$\text{recall} = \frac{|\{\text{relevant images}\} \cap \{\text{retrieved images}\}|}{|\{\text{relevant images}\}|}.$$

In the ideal case, all images from the same category would be retrieved before any images from other categories. In this case, precision would be 100% until recall reached 100%, at which point precision would drop to # of images in query category / # of images in the database. As a result, our final value of precision will be ∼11% since we used 9 categories each containing 100 images. Note, only 9 categories were used since the category of images shown in Fig. 4 are easy to retrieve and their inclusion in the test would only increase the runtime of the experiment.

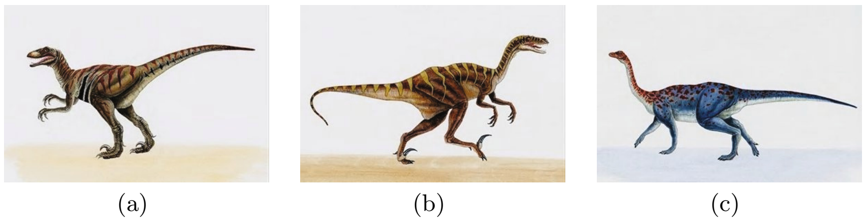


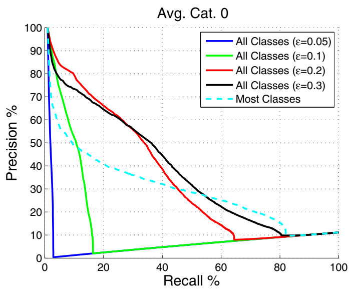
Fig. 4. Examples of images from category 4

The results are presented in Fig. 5 - 17, where the average precision vs. recall plots are given in Fig. 5 & 7, the precision vs. recall results of the best query image are given in Fig. 6 & 8, and Fig. 9 - 17 are the top 40 retrieved images from the best search in each category[2]. These plots present a comparison of the two approaches described in Sections 3 & 4 respectively: neighbourhood-based (most tolerance classes) vs. MCE (all tolerance classes.) In the case of the neighbourhood-based algorithm, only results for $\varepsilon = 0.2$ are given since it was reported in [11] that this value produces the best results that are achievable with reasonable runtime. In other words, the optimal value of ε for the neighbourhood based algorithm on this test and feature set may be greater than $\varepsilon = 0.2$, but, due to prohibitive runtimes, these experiments were not performed. Recall, in any given application (regardless of the distance metric), there is always an optimal ε when performing experiments using the perceptual tolerance relation [11]. For instance, a value of $\varepsilon = 0$ produces little or no pairs of objects that satisfy the perceptual tolerance relation, and a value of $\varepsilon = \sqrt{l}$, means that all pairs of objects satisfy the tolerance relation[3]. Consequently, ε should be selected such that the objects that are relatively[4] close in feature space satisfy the tolerance

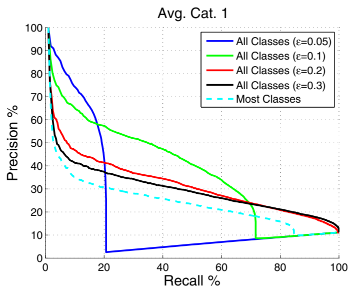
[2] The query image is in the top left position, where the images are ranked from the top down, then left to right.

[3] For normalized feature values, the largest distance between two objects occurs in the interval $[0, \sqrt{l}]$, where l is the length of the feature vectors.

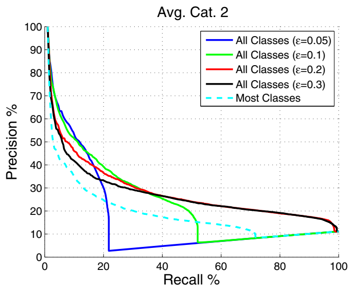
[4] Here, distance of "objects that are relatively close" will be determined by the application.



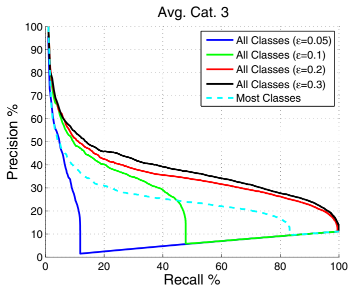
(a)



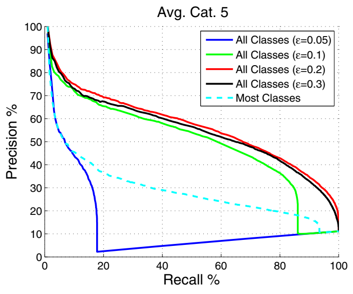
(b)



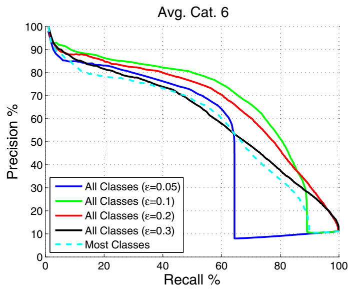
(c)



(d)



(e)

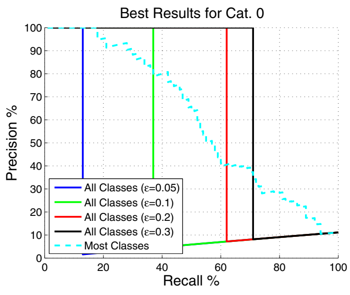


(f)

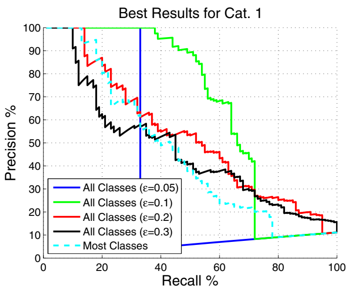
Fig. 5. Average precision versus recall plots grouped by category. (a) - (f) Categories 0 - 6 (excluding category 4).

relation, and the rest of the pairs of objects do not. The selection of ε is straightforward when a metric is available for measuring the success of the experiment. Thus, if runtime were not an issue, the value of ε should be selected based on the best result of the evaluation metric, which, in the context of CBIR, is the best results in terms of precision vs. recall.

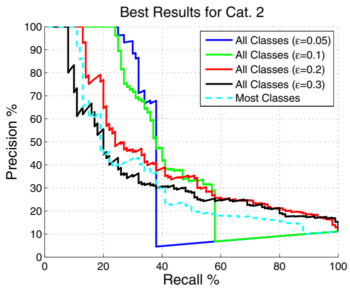
Next, the following presents some observations of the reported results. First, notice that some of the curves have a sharp point of inflection (see, *e.g.*, $\varepsilon = 0.05$ at 20% recall in Fig. 5(b)). These points represents the location at which the remaining tNM values for a particular query become zero. In the case of Fig. 5 & 7,



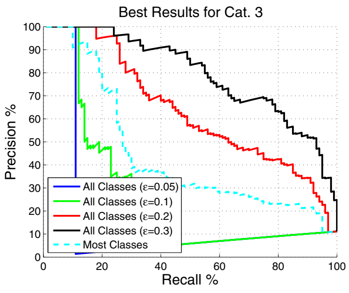
(a)



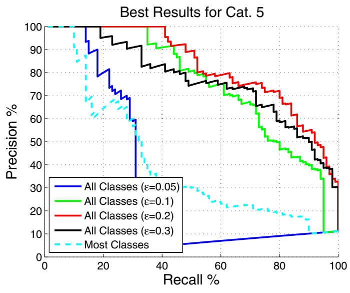
(b)



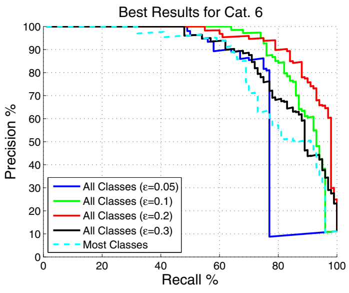
(c)



(d)



(e)

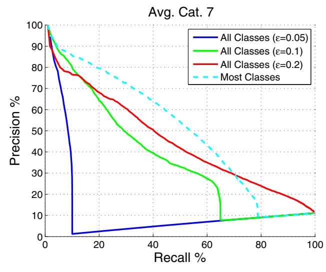


(f)

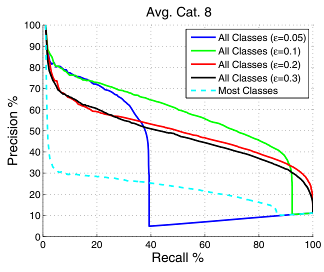
Fig. 6. Best precision versus recall plots grouped by category. (a) - (f) Categories 0 - 6 (excluding category 4).

these points represent the location at which all query images in the category produce a tNM value of zero. In order to provide this clear demarcation, any images from the same category as the query image that produced a tNM value of zero were ranked last in the search[5]. Next, results are not reported for $\varepsilon = 0.3$ for images from category 7 (see, *e.g.* Fig. 2(g)), since the runtime was too large for some of the images in this category. For instance, some image pairs produced

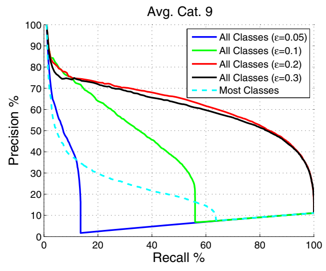
[5] This was not the case in [11], which accounts for some small discrepencies in the plots of this article near the end of the curve.



(a)

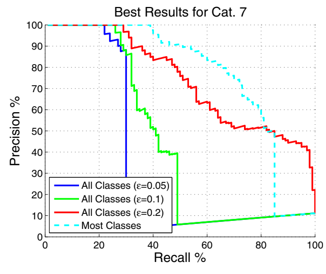


(b)

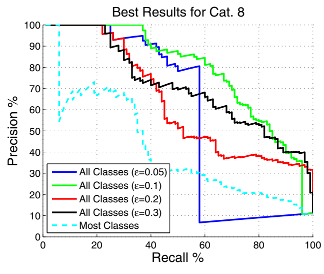


(c)

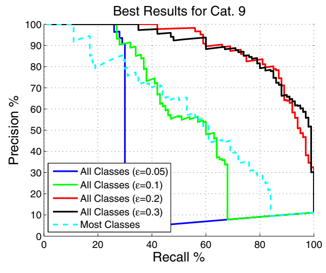
Fig. 7. Average precision versus recall plots grouped by category. (a) - (c) Categories 7 - 9 (excluding category 4).



(a)



(b)



(c)

Fig. 8. Best precision versus recall plots grouped by category. (a) - (c) Categories 7 - 9 (excluding category 4).

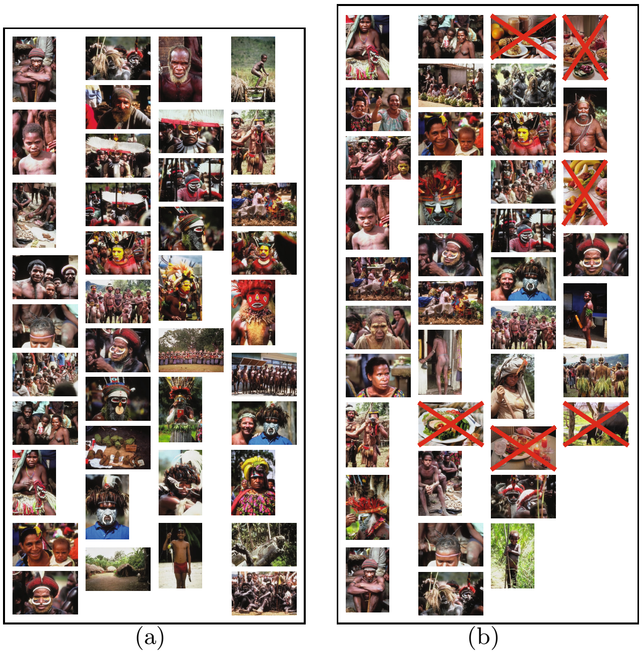


Fig. 9. Top 40 retrieved images using $\varepsilon = 0.2$ for category 0. (a) Results obtained using all classes, and (b) results from using most classes.

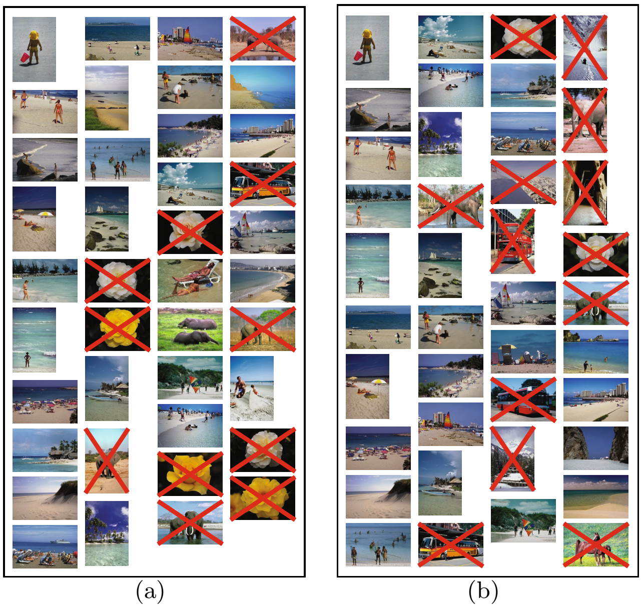


Fig. 10. Top 40 retrieved images using $\varepsilon = 0.2$ for category 1. (a) Results obtained using all classes, and (b) results from using most classes.

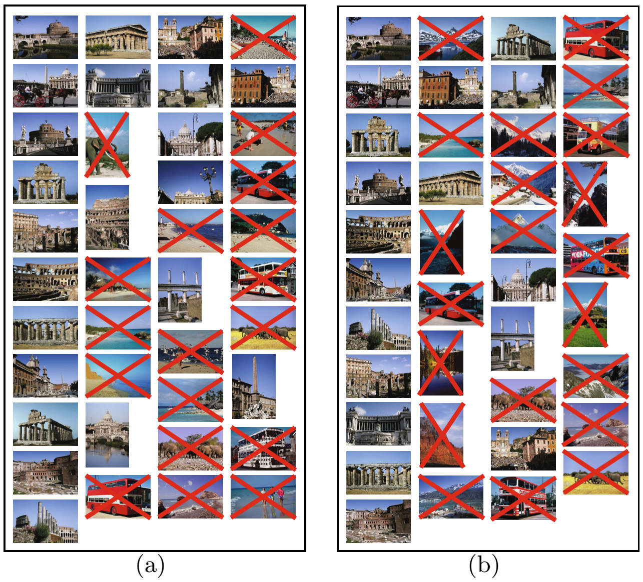


Fig. 11. Top 40 retrieved images using $\varepsilon = 0.2$ for category 2. (a) Results obtained using all classes, and (b) results from using most classes.

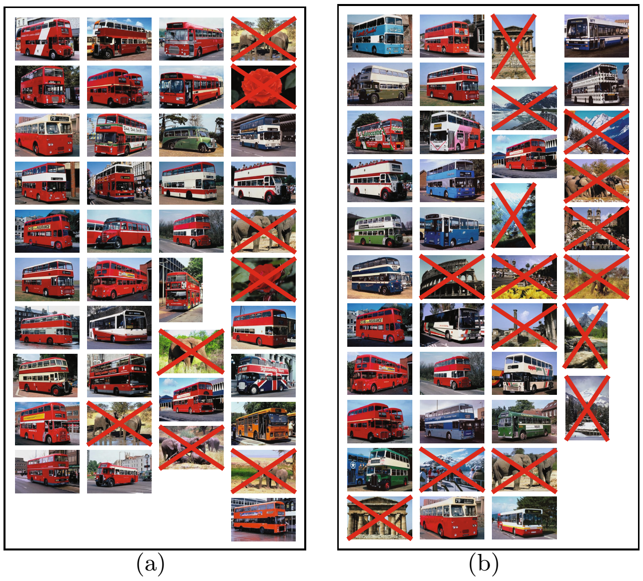


Fig. 12. Top 40 retrieved images using $\varepsilon = 0.2$ for category 3. (a) Results obtained using all classes, and (b) results from using most classes.

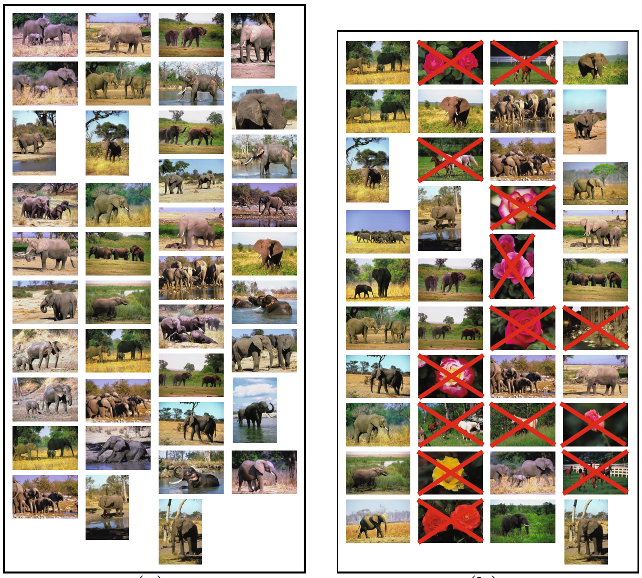

(a) (b)

Fig. 13. Top 40 retrieved images using $\varepsilon = 0.2$ for category 5. (a) Results obtained using all classes, and (b) results from using most classes.

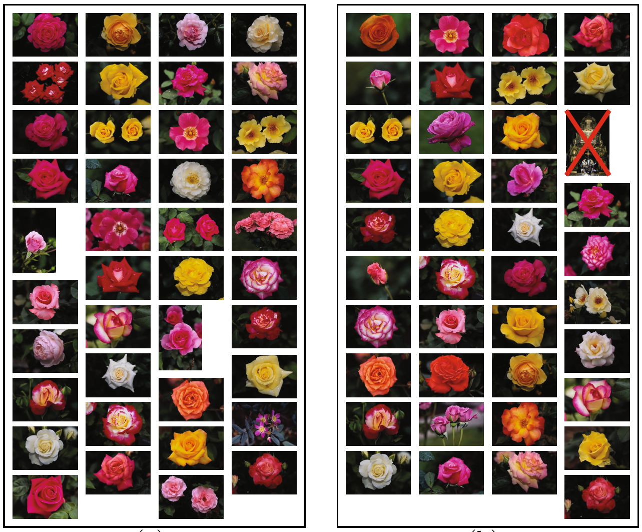

(a) (b)

Fig. 14. Top 40 retrieved images using $\varepsilon = 0.2$ for category 6. (a) Results obtained using all classes, and (b) results from using most classes.

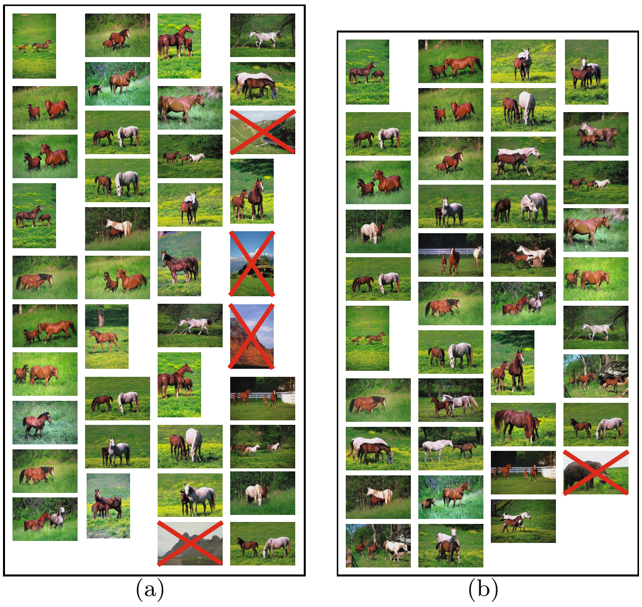


Fig. 15. Top 40 retrieved images using $\varepsilon = 0.2$ for category 7. (a) Results obtained using all classes, and (b) results from using most classes.

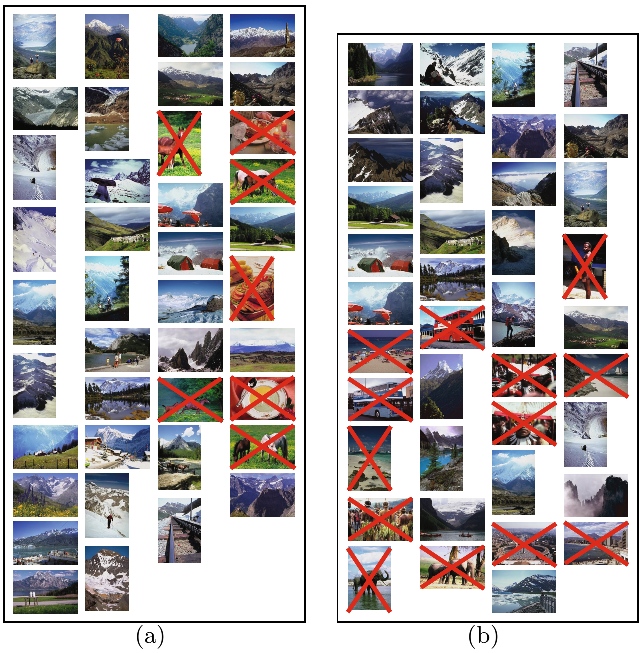


Fig. 16. Top 40 retrieved images using $\varepsilon = 0.2$ for category 8. (a) Results obtained using all classes, and (b) results from using most classes.

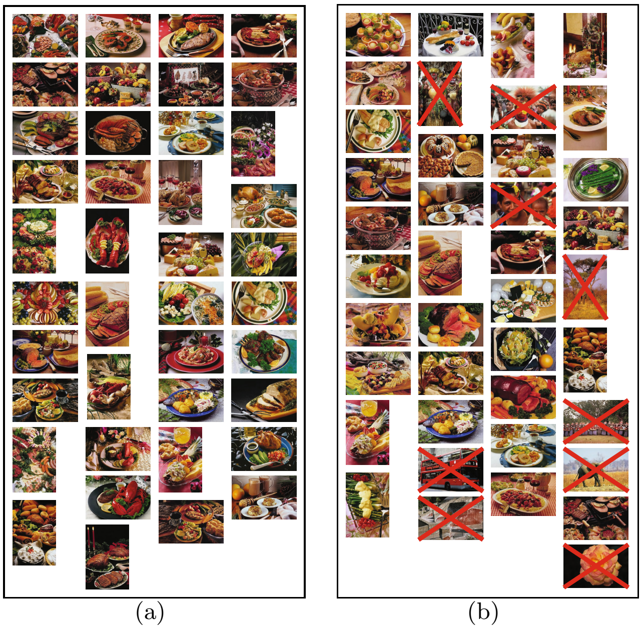


Fig. 17. Top 40 retrieved images using $\varepsilon = 0.2$ for category 9. (a) Results obtained using all classes, and (b) results from using most classes.

in excess of 700,000 tolerance classes (on only 456 objects) and had runtimes of over 2 hours. Finally, for $\varepsilon = 0.2$, the MCE approach significantly outperforms the neighbourhood-based approach in all categories except category 7, which is due to the value of ε. We conjecture the extra tolerance classes produced at $\varepsilon \geq 0.3$ would increase the performance due to the addition of more perceptual information in calculating tNM. This conjecture is substantiated by the results of every other category in which the extra tolerance classes produced better precision vs. recall values. While, the results of the neighbourhood-based approah may also increase with ε, the result from the other categories demonstrate the additional information obtained using all the classes will produce better results.

8 Conclusion

This article presents results within the context of CBIR, where perceptual information within the framework of near set theory is used to discern affinities between pairs of images. Specifically, perceptually relevant information was extracted from a set objects formed from pairs of images, where each object has an associated object description. It is the information contained in these feature vectors that is used to extract perceptual information represented by the discovered tolerance classes. The conjecture that the use of all tolerance classes in a covering of image pairs *increases* the perceptual information available to make decisions on nearness leading to an *improvement* of precision and recall was substantiated by the results presented here. This article also demonstrates that discovery of all tolerance classes is equivalent to the MCE problem. Finally, this article presents a new application of MCE.

References

1. Akkoyunlu, E.A.: The enumeration of maximal cliques of large graphs. SIAM Journal on Computing 2(1), 1–6 (1973)
2. Bomze, I., Budinich, M., Pardalos, P., Pelillo, M.: The maximum clique problem. In: Du, D.Z., Pardalos, P.M. (eds.) Handbook of Combinatorial Optimization, vol. 4. Kluwer (1999)
3. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. Communications of the ACM 16(9), 575–577 (1973)
4. Cazals, F., Karande, C.: A note on the problem of reporting maximal cliques. Theoretical Computer Science 407(1), 564–568 (2008)
5. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. Wiley (2001)
6. Harary, F., Ross, I.C.: A procedure for clique detection using the group matrix. Sociometry 20(3), 205–215 (1957)
7. Hassanien, A.E., Abraham, A., Peters, J.F., Schaefer, G., Henry, C.: Rough sets and near sets in medical imaging: A review. IEEE Transactions on Information Technology in Biomedicine 13(6), 955–968 (2009)
8. Henry, C.: Near set Evaluation And Recognition (NEAR) system. In: Pal, S.K., Peters, J.F. (eds.) Rough Fuzzy Analysis Foundations and Applications, pp. 7-1–7-22. CRC Press, Taylor & Francis Group (2010), `http://wren.ee.umanitoba.ca`
9. Henry, C., Peters, J.F.: Perception-based image classification. International Journal of Intelligent Computing and Cybernetics 3(3), 410–430 (2010), Emerald Literati Network 2011 Award for Excellence
10. Henry, C., Peters, J.F.: Arthritic hand-finger movement similarity measuresments: Tolerance near set approach. Computational and Mathematical Methods in Medicine, article ID 569898, 14 pp (2011)
11. Henry, C.J.: Near Sets: Theory and Applications. Ph.D. thesis (2010), `https://mspace.lib.umanitoba.ca/handle/1993/4267`
12. Henry, C.J.: Neighbourhoods, classes and near sets. Applied Mathematical Sciences 5(35), 1727–1732 (2011)
13. Henry, C.J.: Perceptual Indiscernibility, Rough Sets, Descriptively Near Sets, and Image Analysis. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets XV. LNCS, vol. 7255, pp. 41–121. Springer, Heidelberg (2012)
14. Henry, C.J., Peters, J.F.: Neighbourhood-based vision systems. Cybernetics and Systems 42(1), 33–44 (2011)
15. Henry, C.J., Ramanna, S.: Parallel Computation in Finding Near Neighbourhoods. In: Yao, J., Ramanna, S., Wang, G., Suraj, Z. (eds.) RSKT 2011. LNCS, vol. 6954, pp. 523–532. Springer, Heidelberg (2011)
16. Henry, C.J., Ramanna, S.: Signature-based perceptual nearness. Application of near sets to image retrieval. Mathematics in Computer Science p. 21 (submitted, 2012)
17. Jenkins, J., Arkatkar, I., Owens, J.D., Choudhary, A., Samatova, N.F.: Lessons learned from exploring the backtracking paradigm on the GPU. In: Proceedings of the 17th International Conference on Parallel Processing, vol. II, pp. 425–437 (2011)
18. Li, J., Wang, J.Z.: Automatic linguistic indexing of pictures by a statistical modeling approach. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(9), 1075–1088 (2003)

19. Makino, K., Uno, T.: New Algorithms for Enumerating All Maximal Cliques. In: Hagerup, T., Katajainen, J. (eds.) SWAT 2004. LNCS, vol. 3111, pp. 260–272. Springer, Heidelberg (2004)
20. Meghdadi, A.H.: Fuzzy Tolerance Neighborhood Approach to Image Similarity in Content-based Image Retrieval. Ph.D. thesis (2012)
21. NVIDIA: NVIDIA CUDA programming guide v3.0 (2010), `http://docs.nvidia.com/cuda/index.html`
22. Patel, S.J.: Applied parallel programming (2010), `http://courses.engr.illinois.edu/ece498/al/`
23. Pavel, M.: Fundamentals of Pattern Recognition. Marcel Dekker, Inc., NY (1993)
24. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information Sciences 177, 3–27 (2007)
25. Peters, J.F.: Classification of objects by means of features. In: Proceedings of the IEEE Symposium Series on Foundations of Computational Intelligence (IEEE SCCI 2007), pp. 1–8 (2007)
26. Peters, J.F.: Near sets. General theory about nearness of objects. Applied Mathematical Sciences 1(53), 2609–2629 (2007)
27. Peters, J.F.: Near sets. Special theory about nearness of objects. Fundamenta Informaticae 75(1-4), 407–433 (2007)
28. Peters, J.F.: Tolerance near sets and image correspondence. International Journal of Bio-Inspired Computation 1(4), 239–245 (2009)
29. Peters, J.F.: Corrigenda and addenda: Tolerance near sets and image correspondence. International Journal of Bio-Inspired Computation 2(5), 310–318 (2010)
30. Peters, J.F., Wasilewski, P.: Foundations of near sets. Info. Sci. 179(18), 3091–3109 (2009)
31. Peters, J.F., Wasilewski, P.: Tolerance spaces: Origins, theoretical aspects and applications. Information Sciences 195, 211–225 (2012)
32. Poincaré, H.: L'espace et la géomètrie. Revue de métaphysique et de morale 3, 631–646 (1895)
33. Ramanna, S., Meghdadi, A.H., Peters, J.F.: Nature-inspired framework for measuring image resemblance: A near rough set approach. Theoretical Computer Science 412(42), 5926–5938 (2011), doi:10.1016/j.tcs.2011.05.044
34. Schmidt, M.C., Samatova, N.F., Thomas, K., Byung-Hoon, P.: A scalable, parallel algorithm for maximal clique enumeration. Journal of Parallel and Distributed Computing 69, 417–428 (2009)
35. Shahfar, S.: Near Images: A Tolerance Based Approach to Image Similarity and Its Robustness to Noise and Lightening. M.Sc. thesis (2011)
36. Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-based image retrieval at the end of the early years. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(12), 1349–1380 (2000)
37. Sossinsky, A.B.: Tolerance space theory and some applications. Acta Applicandae Mathematicae: An International Survey Journal on Applying Mathematics and Mathematical Applications 5(2), 137–167 (1986)
38. Tsukiyama, S., Ide, M., Ariyoshi, H., Shirakawa, I.: A new algorithm for generating all the maximal independent sets. SIAM Journal on Computing 6, 505–517 (1977)
39. Wolski, M.: Perception and classification. A Note on near sets and rough sets. Fundamenta Informaticae 101, 143–155 (2010)
40. Zeeman, E.C.: The topology of the brain and the visual perception. In: Fort, K.M. (ed.) Topoloy of 3-manifolds and Selected Topices, pp. 240–256. Prentice Hall, New Jersey (1965)