An Effective Similar Object Searching System for Complex Satellite Images

Jung-Bum Kim, Chin-Wan Chung and Yongkwon Kim
Division of Computer Science
KAIST
Daejeon, Korea
{jbkim, yongkwon}@islab.kaist.ac.kr, chungcw@cs.kaist.ac.kr,

Deok-Hwan Kim
School of Electronic Engineering
Inha University
Incheon, Korea
deokhwan@inha.ac.kr

Seok-Lyong Lee
School of Industrial and Information Engineering
Hankuk University of Foreign Studies
Yongin-si, Gyunggi-do, Korea
sllee@hufs.ac.kr

***Abstract*— As the amount of satellite images in the database increases, the demand for searching similar satellite images also increases. The content-based image retrieval (CBIR) is one of approaches for finding images containing similar objects from an image database. To apply CBIR to satellite images, image segmentation is necessary to separate the shape of an object from an image. However, due to several properties of satellite images, image segmentation is very difficult. Therefore, it is difficult to get a good result for satellite images by using CBIR.**

In this paper, we propose a new approach which doesn't require image segmentation to search similar images. We use the SIFT keypoint descriptor, which doesn't require image segmentation, as a shape descriptor. However, the basic SIFT matching needs to handle a large number of keypoint descriptors. In the proposed approach, we set a pruning circle to find similar objects from an image. Then, we prune some SIFT keypoint descriptors which are not located on similar objects. A similarity is assigned to an image based on the remaining SIFT keypoint descriptors. We also propose a way to use an index for scalability of the proposed approach. Experimental results show that the proposed approach searches images containing a similar object effectively in the case of satellite images.

Keywords-Satellite images; CBIR; SIFT; Similarity search

I. INTRODUCTION

Image data such as satellite images are continuously produced. As the amount of image data increases, the demand for the retrieval of similar images also increases. To efficiently find similar images from a large collection of images, a technique for retrieval of similar images is necessary. The use of annotations is one way to find similar images. However, due to the subjectivity of annotations, there are some limitations. To overcome the limitation of annotations, we have to search similar images based on their contents such as color, shape, and texture. In the content-based image retrieval (CBIR), similar images can be found based on the contents of an image itself. Up to now, there are many researches about CBIR [2], [12], [8].

Although we can use CBIR to search similar images, it is difficult to get a good result in case of searching images which are very complex and do not have color information such as satellite images. The reason is as follows. Due to the lack of color information, we should use other image contents such as shape and texture. To use shape information instead of color information, image segmentation needs to be applied first to separate objects from an image. However, due to the complexity of images, image segmentation produces poor results. Even though shape feature extraction techniques such as [5], [11] are used, it is difficult to get a good retrieval result from complex satellite images due to the difficulty of the image segmentation. Since we cannot successfully extract shape information of an image when the result of image segmentation is poor, it is difficult to search similar images successfully. The purpose of this paper is to find images containing similar objects from satellite image databases. Therefore, we propose a new approach which doesn't require image segmentation to search similar satellite images.

To retrieve similar images without image segmentation, a feature extraction technique which doesn't require image segmentation is necessary. One of good feature extraction techniques which doesn't require image segmentation is SIFT (Scale-Invariant Feature Transform) [10]. In SIFT, when it extracts features, SIFT first finds keypoints which are invariant to various changes in some aspects such as scale, rotation and viewpoint. It helps finding the same objects despite their various scales, rotations or viewpoints. Then, it extracts features from the region around keypoints. Therefore, SIFT doesn't require image segmentation. In [10], a basic matching approach is proposed to decide whether there are objects in another image the same as an object in a certain image. In this paper, we propose a new approach to apply the SIFT keypoint descriptor to retrieve similar objects from a satellite image database. In the proposed approach, we set a pruning circle based on a query image to find similar objects from an image. Then, we prune some SIFT keypoint descriptors which are not

located on similar objects. We assign a similarity value to an image based on the remaining SIFT keypoint descriptors. We also propose a way to use an index for scalability of the proposed approach.

Experimental results show the proposed approach searches images containing a similar object more effectively in case of satellite images than the approach using the image segmentation and shape feature [5], [11] or the approach using the basic SIFT matching.

A preliminary result of this research was reported in [9]. However, the result only showed simple algorithms, and the feasibility of the concept. This paper improved algorithms and included efficiency consideration.

The remainder of this paper is organized as follows. In Section 2, the proposed approach, an effective similar object searching, is presented. In Section 3, we compare the experimental result of the proposed approach with those of the others by using the dataset consisting of satellite images. Finally, in Section 4, we conclude this paper.

II. Similar Object Searching for Satellite Images

To search similar objects, we intend to assign a similarity to an image based on its SIFT keypoint descriptors. For objects which are not the same but similar, the keypoints similar to a query keypoints can't be located on comparable objects – a query keypiont is a keypoint in a query image. For this case, to map a query keypoint to the keypoint located on a comparable object, we prune the keypoints which are similar but not located on a comparable object. By pruning some non-relevant keypoints, the result of the retrieval can be enhanced.

Figure 1 shows how to calculate a similarity of an image. In the figure, the image on the left side is a query image and the image on the right side is an image in a database.

When a query is issued, *m* keypoints most similar to each query keypoint are found from an image in a database as shown in Figure 1. We assume that *m* is 3 in this example. Let each query keypoint be □, Δ , and ○ respectively. A keypoint in a database image with the same shape -□, △, and ○ in the figure - can be considered as the keypoint similar to the corresponding query keypoint. The numbers represent the similarity ranks of keypoints. Therefore, □1 represents the keypoint most similar to the query keypoint □, and □2 represents the 2nd most similar keypoint, similarly. In Figure 1, □1 is located on a comparable object but △1 and ○1 are not located on a comparable object. If we calculate a similarity of the image on the right side based on □1, △1, and ○1, △1 and ○1 which are not located on a comparable object can affect the similarity of the image. To exclude these non-relevant keypoints, we prune keypoints which are not considered as located on a comparable object. In Figure 1, △1 and ○1 can be pruned because they are scattered. Since SIFT only uses the region around a keypoint to extract features, there can be a similar keypoint not located on a comparable object such as △1 and ○1. After pruning some keypoints, each query keypoint is mapped to the most similar keypoint among

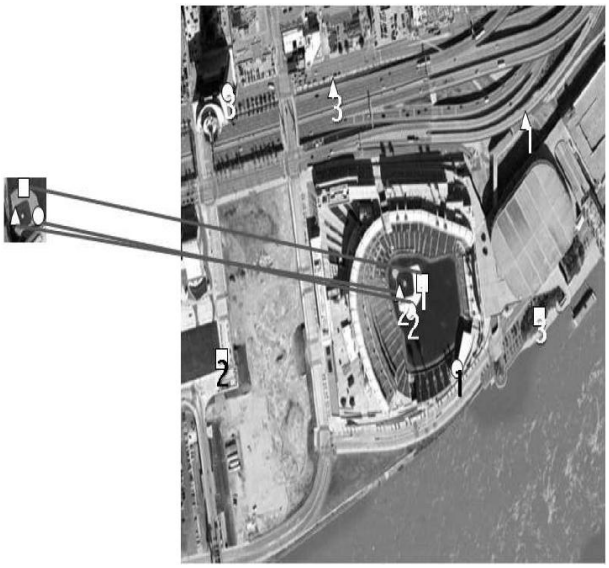


Figure 1 An example for calculating a similarity of an image

TABLE 1 Notations and Definitions

Q	the set of query keypoints
q_i	i th query keypoint in Q
I	an image in databases
C_i	the closest set of q_i
C_i'	the closest set without the the farthest keypoint
r_i	the representative keypoint of C_i
$RemainedSet$	the set of keypoint to be remained after pruning
$\|.\|$	cardinality of a set
$D(.,.)$	distance between two elements or two sets
m	$\|C_i\|$

the remained keypoints. Then, each query keypoint is mapped to the keypoint on a comparable object. The distance of an image in a database is calculated based on the mapped keypoints. Therefore, by pruning some non-relevant keypoints, we expect to get a more effective retrieval result. In this paper, we use the Euclidean distance when calculating the distances between keypoints.

The procedure of the proposed approach can be summarized as follows:

1) Pruning circle setting
  a) Finding a closest set of each query keypoint
  b) Keypoint pruning
  c) Representative keypoint setting
2) Setting a distance of an image
3) Image ranking

A. *Pruning circle setting*

As a keypoint not located in a similar object can be matched to a query keypoint, we need to prune that kind of keypoint. A pruning circle is a basis for the keypoint pruning. Therefore, we should set a pruning circle. A pruning circle is defined as a circumcircle of a query image. If there are more than *k* keypoints in a pruning circle, we do not prune these keypoints because we consider these keypoints are located on a comparable object. By using a pruning circle, keypoint pruning can be rotation-invariant. The diameter of a pruning circle is

This research was supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (grant number 2010-0000863) and in part by Mid-career Researcher Program through NRF grant funded by the MEST (2009-0083862).

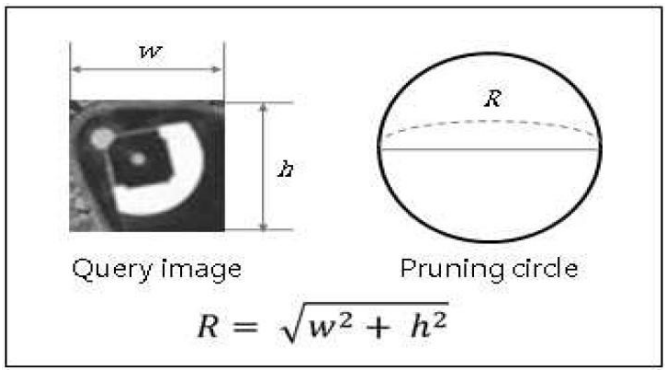


Figure 2. Pruning circle setting

defined as the length of diagonals of the query image. Figure 2 shows the process of setting a pruning circle.

k is the number to determine whether keypoints in a pruning circle are going to be pruned. If there are more than *k* keypoints in a pruning circle, keypoints in the pruning circle are excluded in the keypoint pruning. *k* can be set by taking a parameter which should be an integer. In SIFT basic matching, 3 keypoints matched correct are considered sufficient for reliable matching. Therefore, when a user doesn't provide a parameter, *k* is set to 3 by default.

B. Setting a distance of an image

After we set a pruning circle, we set a distance of an image. To set a distance of an image, we should find keypoints most similar to each query keypoint. Then, we eliminate keypoints not located on a similar object based on a pruning circle and map each query keypoint to a keypoint remained after the previous elimination. A distance of an image is determined using the mapped keypoints.

In basic SIFT matching, SIFT finds the most similar keypoint and the 2nd most similar keypoint to each query keypoint whereas we find multiple similar keypoints for each query keypoint. These multiple similar keypoints are called the Closest set. The definition of the Closest set is as follows:

Definition 2.1: for a given number *m*, the Closest set C_i of the query keypoint q_i is the set of the *m* most similar keypoints in an image *I* to q_i.

By calculating the distances between each query keypoint and all keypoints in a database image, we can find the Closest set of each query keypoint. For finding Closest sets, it is important to select *m* which is the number of keypoints in a Closest set. *m* can be calculated as follows.

We first calculate the probability that there are more than *k* points in a certain pruning circle when we randomly place *n* points in an image. Even though the probability is very small, if a certain pruning circle contains more than *k* keypoints, there can be similar objects in the region related to the pruning circle. Therefore, we calculate *n* so that the probability becomes small, then calculate *m* based on *n* calculated.

A random variable *X* can be defined as follows:

X: the number of points in a certain pruning circle when *n* points are placed randomly in an image.

p is the probability that one point placed randomly is contained by a certain pruning circle. *p* can be calculated as follows:

$$p = \frac{size\ of\ pruning\ circle}{database\ image\ width \times database\ image\ height}$$

P(X=x) represents the probability that there are exactly *x* keypoints in a certain pruning circle when *n* points are placed randomly in an image. *P(X=x)* is calculated as follows:

$$P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}, x = 0, 1, ..., n$$

Therefore, $X \sim b(n, p)$, the random variable X follows the binomial distribution. The probability that there are more than *k* points in a certain pruning circle when we randomly place *n* points in an image can be represented by $P(X \geq k)$. $P(X \geq k)$ can be calculated using the binomial distribution.

As mentioned, to calculate *m*, we have to calculate *n* so that $P(X \geq k)$ becomes small. To calculate proper *n*, we take a very small probability value, $P\tau$, as a parameter. Then we calculate the largest *n* such that $P(X \geq k) \leq P\tau$. Let the largest *n* calculated be n_1. *m* can be calculated as follows:

$$m = \lceil \frac{n_1}{|Q|} \rceil + 1$$

Since there can be n_1 points in an image, we can find the number of keypoints which can be contained by a Closest set, by dividing n_1 by the number of query keypoints. The reason why we add '1' is to prepare the case that all keypoints in a Closest set are pruned. The added keypoint is excluded in the keypoint pruning. Let the set of keypoints in C_i excluding the added keypoint be C'_i. Therefore, in the keypoint pruning, only keypoints in C'_i are considered. From now on, the keypoints related to the keypoint pruning are regared as the keypoints in C'_i.

After finding the Closest set of each query keypoint, the keypoint pruning is conducted. In the keypoint pruning, to map each query keypoint to a keypoint on a comparable object, we prune some keypoints which are not considered as located on a comparable object.

As mentioned, if a certain pruning circle contains more that *k* keypoints, the keypoints in the pruning circle are not pruned. For the keypoint pruning, we have to find a pruning circle containing more than *k* keypoints. Keypoint pruning algorithm in Table II can find the keypoints in a pruning circle containing more than *k* keypoints. Keypoint pruning algorithm first finds a candidate region for each keypoint. A candidate region is the circular region centered on each keypoint with the radius same as the diameter of a pruning circle. A candidate region is the region in which all possible pruning circles containing the center keypoint can exist. Then, the algorithm sorts the keypoints in the candidate region by the distance from the center keypoint in non-decreasing order. Then, it finds combinations consisting of *k-1* keypoints among the keypoints in a candidate region except for the center keypoint. The reason

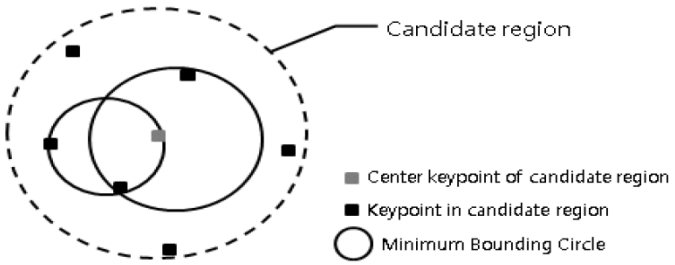


Figure 3. A candidate region and minimum bounding circles

why it sorts is to list the keypoints in each combination by the distance from the center keypoint in non-decreasing order. The algorithm then calculates a minimum bounding circle for the keypoints in each combination and the center keypoint. When a minimum bounding circle calculated is not larger than a pruning circle, the center keypoint must be contained in a pruning circle containing more that *k* keypoints. Therefore, the center keypoint must not be pruned and be included in *RemainedSet*. *RemainedSet* includes the keypoint which must not be pruned.

In Figure 3, we can see a candidate region and minimum bounding circles consisting of the center keypoint and the *k-1* keypoints in a combination.

Keypoint pruning algorithm should find minimum bounding circles consisting of the center keypoint and each combination to decide whether the center keypoint is contained in a pruning circle containing more than *k* keypoints. However, like the branch and bound approach, for a combination trivially larger than a pruning circle, it is not necessary to calculate a minimum bounding circle. Therefore, it does not calculate a minimum bounding circle in that case. As a result, the algorithm can be more efficient.

To calculate a minimum bounding circle, an MBC, we use the Smallest Enclosing Balls of Points source code [7] based on the algorithm in [4].

In Lemma 1, we prove the correctness of the keypoint pruning algorithm in Table II.

Lemma 1: Keypoint pruning algorithm finds all keypoints in a pruning circle containing more than *k* keypoints.

Proof: Assume that keypoint pruning algorithm can't find a certain keypoint k_1 in a pruning circle containing more than *k* keypoints. In this case, there are more than *k* keypoints in the candidate region centered on k_1 and pruning circles containing more than *k* keypoints including k_1.

Keypoint pruning algorithm first finds a candidate region for each keypoint in Closest sets. Then, it finds all combinations with the *k* keypoints including k_1 in the candidate region. Then, it finds an MBC which consists of the keypoints in a combination and is not larger than a pruning circle. Since there is a pruning circle containing more than *k* keypoints including k_1 in the candidate region, there must be an MBC consisting of *k* keypoints in a combination. Therefore, k_1 becomes included in *RemainedSet*. That is, Keypoint pruning algorithm can find k_1.

TABLE II KEYPOINT PRUNING ALGORITHM

```
Algorithm KeypointPruning
1. for each query keypoint q_i in Q
2.   for each keypoint in C_i'
3.     calculate a candidate region centered on the corresponding keypoint.
4.     cand_region := a candidate region calculated.
5.     sort keypoints in cand_region by the distance from the center keypoint
6.     construct combinations of k − 1 keypoints out of keypoints
       in cand_region.
7.     for each combination
8.       calculate an MBC contating all keypoints
         in the corresponding combination.
9.       if the MBC calculated is smaller than Pruning circle then
10.        assign the center keypoints to RemainedSet.
11.        break.
12.      else
13.        for each keypoint in the corresponding combination
             sorted by the distance from the center keypoint
14.          MBC_sofar := calculate an MBC including keypoints so far.
15.          if MBC_sofar is larger than Pruning circle then
16.            key_found := the corresponding keypoint.
17.            skip the combinations including key_found
               and keypoints farther than key_found.
18.            break.
19.          end if
20.        end for
21.      end if
22.    end for
23.  end for
24. end for
25. prune keypoints which are not included in RemainedSet.
```

Since the assumption is contradicted, Lemma 1 becomes proved. □

Keypoint pruning algorithm in Table II performs line 3 ~ 22 for all keypoints in C_i. Therefore, the algorithm performs line 3 ~ 22 for the keypoints which are already in *RemainedSet*. By eliminating these unnecessary operations, improved keypoint pruning algorithm in Table III makes the keypoint pruning more efficient. In addition, the algorithm in Table II calculates the MBC for each combination. As calculating an MBC is a computationally expensive task, the algorithm in Table III uses some condition to avoid this expensive task. To avoid calculating MBCs, the algorithm in Table III calculates the distance of each keypoint in a combination from the center keypoint. Then, if the distance of a certain keypoint is larger than the diameter of a pruning circle, the algorithm doesn't perform operation in line 16 ~ 23 for the combinations including the keypoint with the distance larger than a pruning circle and keypoints farther than the keypoint. Therefore, the algorithm makes the keypoint pruning more efficient.

After the keypoint pruning, a Closest set consists of the keypoints in *RemainedSet* and the keypoints not in *RemainedSet*. A representative keypoint is a keypoint mapped to a query keypoint. In Figure 1, a representative keypoint of the query keypoint △ is the keypoint △2. The definition of a representative keypoint is as follows:

Definition 2.2: A representative keypoint r_i of a query keypoint q_i is the keypoint in C_i and *RemainedSet* with the smallest distance from q_i.

A representative keypoint is a keypoint in the Closest set of the query keypoint after the keypoint pruning, which is most similar to the query keypoint.

TABLE III IMPROVED KEYPOINT PRUNING ALGORITHM

```
Algorithm ImprovedKeypointPruning
1. for each query keypoint q_i in Q
2.     for each keypoint in C'_i
3.         if the corresponding keypoint is in RemainedSet then
4.             continue.
5.         end if
6.         calculate a candidate region centered on the corresponding keypoint.
7.         cand_region := a candidate region calculated.
8.         sort keypoints in cand_region by the distance from the center keypoint
9.         construct combinations of k − 1 keypoints out of keypoints in cand_region.
10.        for each combination
11.            calculate an MBC contating all keypoints in the corresponding combination.
12.            if the MBC calculated is smaller than Pruning circle then
13.                assign all keypoints in the MBC calculated to RemainedSet.
14.                break.
15.            else
16.                for each keypoint in the corresponding combination
                       sorted by the distance from the center keypoint
17.                    dist_sofar := calculate the largest distance of a pair of keypoints so far.
18.                    if dist_sofar is larger than the diameter of Pruning circle then
19.                        key_found := the corresponding keypoint.
20.                        skip the combinations including key_found and keypoints farther than key_found.
21.                        break.
22.                    end if
23.                end for
24.            end if
25.        end for
26.    end for
27. end for
25. prune keypoints which are not included in RemainedSet.
```

A representative keypoint r_i of a query keypoint q_i can be found as follows:

$$r_i = \min_{keypoint \in (C_i \cap RemainedSet)} (D(q_i, keypoint))$$

After finding a representative keypoint for each query keypoint, we should set the distance of an image. The distance of an image is defined as the average of the distances between each query keypoint and its representative keypoint.

$$D(Q, I) = \frac{1}{|Q|} \sum_{i=0}^{|Q|} D(q_i, r_i)$$

C. *Image ranking*

After setting the distances of images, we rank images according to their distances in non-decreasing order. Then, we provide *topK* images with higher ranks. *topK* is the number of images to be retrieved as a result.

III. EXPERIMENTS

We use the system with Pentium D 2.80 GHz CPU and 2.00 GB RAM and Windows XP OS. The platform used is Visual Studio .NET. The dataset used consists of 327 satellite images. The satellite images are captured from Google Earth [6] and include 22 baseball stadium images and 30 American football stadium images. Our purpose is to find those stadium images from the dataset. The size of the images in the dataset is about 520 pixels by 410 pixels. Query images are the images only containing a baseball stadium or an American football stadium. The sizes of the objects in a query image are similar to those in images in the dataset.

In experiments, we compare the proposed approach with the basic SIFT matching [10] and the Angular Radial Transform (ART) [5], [11] approaches. In the basic SIFT matching approach, if a query keypoint is matched to a keypoint in a database, the approach can find only one similar image. Therefore, we let the approach match a query keypoint to a keypoint in each image in a database so that we can match multiple images. In the approach using ART, we apply region-growing based image segmentation to each image to separate objects then extract shape features using ART from the result of image segmentation. The approach using ART is a kind of CBIRs using image segmentation. As, according to the papers such as [13], [1], ART is considered a good region-based shape descriptor and is designated as the MPEG-7 standard, we select ART as a shape feature extraction technique.

In the figures showing experimental results, 'Proposed' stands for the proposed approach, 'Basic SIFT Matching' stands for the approach using basic SIFT matching, and 'Seg+ART' stands for the approach using ART and image segmentation.

In this experiment, we query 8 images containing only an Amreican football stadium. Then, we compare the average of results of 8 queries. The size of a query image is about 50 pixels X 100 pixels. Figure 4 shows the results of 8 queries. In the figure, the average precision and recall of top10, 20, 30, 40 queries are shown. Parameters used are $k=5$ and $P\tau = 0.001$. The average precision and recall of top30 queries is about 0.45 and 0.45, respectively.

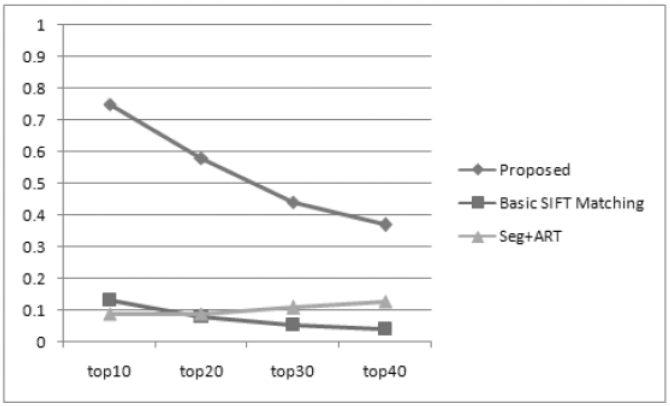
Figure 5 shows the result of retrieval by our approach. The upper-left image is the query image. The images on the bottom of the query image are the result images. In the result image, 'Rank' specifies the rank of the image. Lower rank means higher similarity. 'Distance' specifies the similarity of the image. The character in the bottom of 'Rank' is the name of the image. The image with the name including 'football' is a relevant image. As Shown in Figure 5, the proposed approach can successfully retrieve the images with a football stadium for the football stadium query.

IV. CONCLUSION

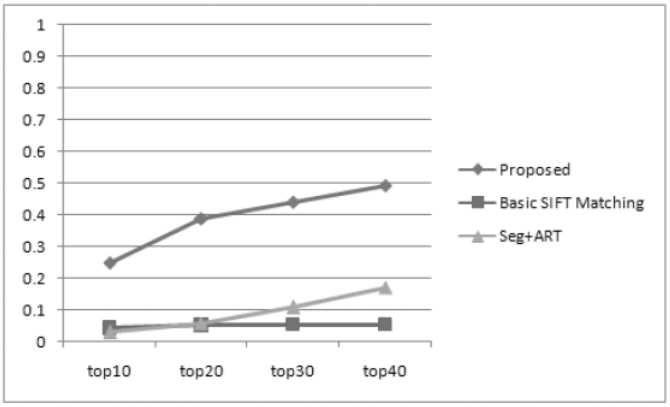
The purpose of this paper is to find the images containing similar objects from satellite image databases.

Although we can use the content-based image retrieval technique to search similar images, it is difficult to get a good result in case of searching images which are very complex and are difficult to use color information such as satellite images. The reason is that, as shape information is appropriate for satellite images, we should apply image segmentation to separate objects from the images. However, due to the complexity of satellite images, image segmentation is very difficult. We cannot successfully extract shape information from objects when the result of image segmentation is poor.

In this paper, we proposed a new approach to search similar images without image segmentation for the purpose. To do that, we use the SIFT keypoint descriptor, which doesn't require image segmentation, as a shape descriptor. We propose the approach which assigns a similarity to an images based on its SIFT keypoint descriptors. In the proposed approach, first, we set a pruning circle based on a query image. Then, for assigning a similarity to an image, we find closest sets for each query keypoint based on SIFT keypoint descriptors and then do the keypoint pruning. Finally, we set a representative keypoint of each query keypoint then assign a similarity to an image. As we eliminate the keypoint which is not considered as located on a comparable object by doing the keypoint pruning, the result of retrieval can be improved.



(a) Precision



(b) Recall

Figure 4. Precision and Recall for American football stadium queries

For efficient retrieval, we propose the approach using an index to avoid the situation where we should assign a similarity to all images in a database.

Experimental results show the proposed approach searches images containing a similar object more effectively in case of satellite images than the two other approaches.

ACKNOWLEDGMENT

This research was supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (grant number 2010-0000863) and in part by Mid-career Researcher Program through NRF grant funded by the MEST (2009-0083862).

REFERENCES

[1] John P. Eakins, K. Jonathan Riley, and Jonathan D. Edwards. Shape feature matching for trademark image retrieval. In CIVR 2003, 28-38, 2003.

Figure 5. The retrieval result of the football stadium query

[2] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Tanker. Query by image and video content: The QBIC system. IEEE Computer, 28(9):23-32, 1995.

[3] Jun Jie Foo and Ranjan Sinha. Pruning sift for scalable near-duplicate image matching. In Eighteenth Australasian Database Conference (ADC 2007), 63:63-71, 2007.

[4] Bernd Gartner. Fast and robust smallest enclosing balls. In Proceedings of the 7th Annual European Symposium on Algorithms, 325-338, 1999.

[5] ISO/IEC. 15938-3/FDIS, Information technology - Multimedia content description interface - Part 3 Visual, 2001.

[6] Google Earth. http://earth.google.com.

[7] Miniball Code. http://www.inf.ethz.ch/personal/gaertner/miniball.html.

[8] Tom Huang, Sharad Mehrotra, and Kannan Ramchandran. Multimedia analysis and retrieval system (MARS) project. In Proceedings of the Data Processing Clinic, 1996.

[9] J.B. Kim, C.W. Chung, D.H. Kim, S.H. Kim, and S.L. Lee. Similar satellite image search using sift. Journal of Korean Institute of Information Scientists and Engineers: Databases (in Korean), 35(5):379-390, 2008.

[10] David G. Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2):91-110, 2004.

[11] B.S. Manjunath, P. Salembier and T. Sikora. (2002) Introduction to MPEG-7 : Multimedia Content Description Interface, Wiley, 2002.

[12] John R. Smith and Shinh-Fu Chang. VisualSEEk: a fully automated content-based image query system. In Proceedings of the fourth ACM international conference on Multimedia, 87-98, 1996.

[13] R.C. Veltkamp, and L.J. Latecki. Properties and performance of shape similarity measures. In Proceedings of IFCS 2006 Conference: Data Science and Classification, 2006.