Improvement of Retrieval Speed and Required Amount of Memory for Geometric Hashing by Combining Local Invariants

Masakazu Iwamura, Tomohiro Nakai, and Koichi Kise
Osaka Prefecture University, Japan
{masa,kise}@cs.osakafu-u.ac.jp, nakai@m.cs.osakafu-u.ac.jp

Abstract

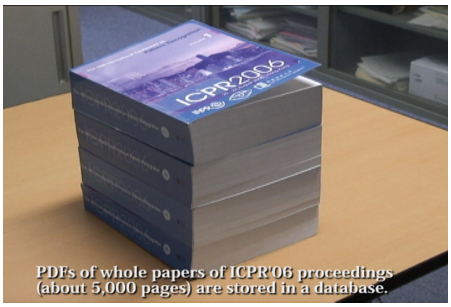
The geometric hashing (GH) is a well-known model-based object recognition technique with good properties both in retrieval speed and required amount of memory. However, it has a significant weak point; as the number of objects increases, both retrieval speed and required amount of memory increase in the cubic, fourth or higher order. Recently, a new technique "locally likely arrangement hashing (LLAH)" whose computational cost is a linear order has been proposed. The objective of the current paper is to reveal how LLAH improves the performance. By comparing GH and LLAH, we describe four primary factors of the performance improvement.

1 Introduction

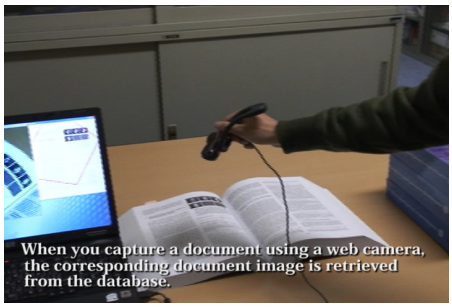
x $\boldsymbol{x}$ In the current paper, we discuss a problem to find the corresponding object from the database using feature points extracted from a query object. For the problem, rich descriptors such as SIFT descriptor [9] (a typical feature vector is 128 dimensions), PCA-SIFT descriptor [7] (typically 36 dimensions) and SURF descriptor [2] (typically 64 dimensions) are often used to describe the object. Since the descriptors are designed to be robust for noises such as change of intensity, rotation and scale, the corresponding object is retrieved by searching similar values of descriptors from the database. Popular searching methods include approximate nearest neighbor (ANN) [1], locality-sensitive hashing (LSH) [6, 3] and vocabulary tree [11]. However, distinctive rich descriptors are not always available. For example, a SIFT descriptor extracted from a texture type repeating pattern including a text region is not distinctive.

Thus, to the contrary to the rich descriptors, we discuss methods employing a poor descriptor whose feature is only *the location of the feature point*. This enables object recognition techniques to be widely applicable as mentioned below. However, this makes the problem difficult since retrieving the corresponding feature point in the database only with a single feature point is impossible. Focusing the arrangement of other feature points is necessarily required. Furthermore, if the query object image is taken from an oblique angle, it has undergone some geometric transformation. This makes the problem more difficult because the arrangement is no longer identical to the stored one in the database. Therefore, removing the effect of the geometric transformation is also required for precise retrieval.

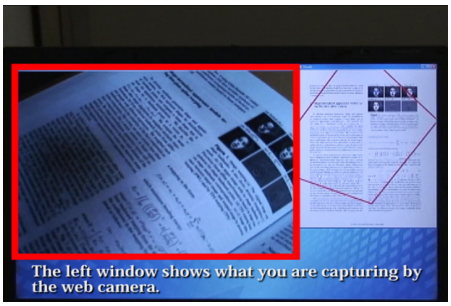
BMVC 2007 doi:10.5244/C.21.103



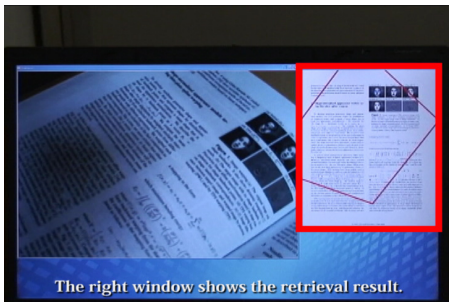
(a) Stored documents



(b) Search with a web camera



(c) Captured image in the left window



(d) Retrieval result in the right window

Figure 1: Snapshots of a real-time image retrieval system [5] using LLAH [10]. The system is invariant against rotation, scaling, perspective distortion, occlusion and curvature of a page. The system works at around 7Hz for 5,000 pages, and the clock does not depend on the size of the database so much. (a) stored documents in the database, 5,000 pages in this demo. (b) a search scene with a 1.3M pixels web camera. (c) a screen shot including a captured image in the left window. (d) a screen shot including a thumbnail of the retrieved page and the corresponding region to the captured image drawn in a red rectangle in the right window.

The simplest way to resolve the problem is an exhaust search which requires the computational cost of $O(N!)$ for a search of an object including N feature points under perspective distortion. The cost is reduced by the geometric hashing (GH) down to $O(N^5)$ introducing an idea of geometric invariance [8, 16]. GH is known to be a general model-based object recognition method and widely used not only in computer vision, but also in many other domains including bioinformatics [12, 16]. However, GH still requires much processing time and large amount of memory, especially for a large number of objects. Many variants of GH have been proposed and some of them reduce the cost using probabilistic approach such as random reduction of feature points. However, probabilistic reduction of computational cost cannot avoid reduction of retrieval accuracy because computational cost and retrieval accuracy are in a trade-off relationship [4]. Therefore, it is quite difficult to apply GH and its variants to practical applications requiring quick response.

For the problem, we have proposed a new technique “locally likely arrangement hash-

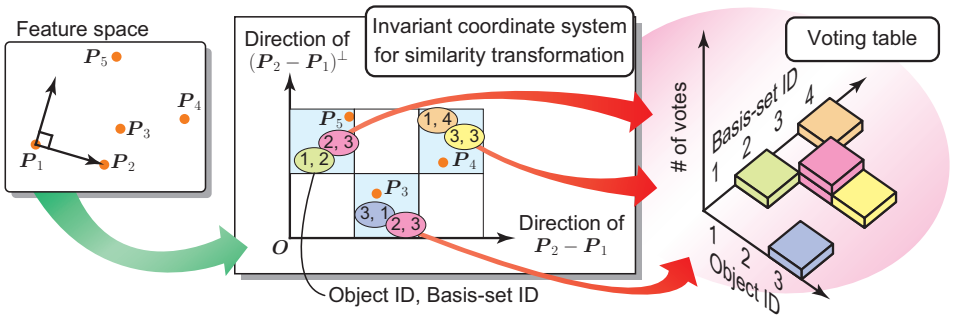


Figure 2: Retrieval process of the geometric hashing.

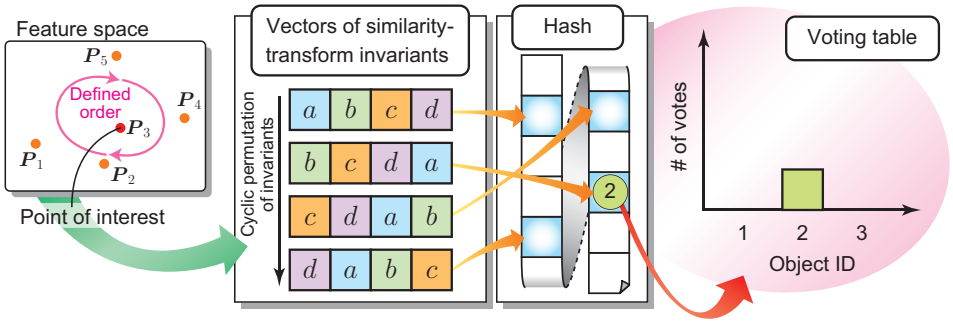


Figure 3: Retrieval process of LLAH. Invariants in the figure are $a = \frac{\|P_4 - P_1\|}{\|P_5 - P_1\|}$, $b = \frac{\|P_2 - P_5\|}{\|P_4 - P_5\|}$, $c = \frac{\|P_1 - P_4\|}{\|P_2 - P_4\|}$, and $d = \frac{\|P_5 - P_2\|}{\|P_1 - P_2\|}$, respectively.

ing (LLAH)" which outperforms GH in both processing time and required amount of memory [10]. LLAH requires the computational cost of $O(N)$ for a search. Due to its outstanding performance, we have applied it to a real-time image retrieval system [5] (see Fig. 1 for snapshots). In about 150 milliseconds, the system with 4GB memory can retrieve a corresponding image from 5,000 images, and find the corresponding region captured by a web camera. In spite of its outstanding performance, LLAH has not analyzed in detail. Therefore, the objectives of the current paper is to reveal where the outstanding performance of LLAH comes from. For the sake of the purpose, we compare GH and LLAH, and describe four primary factors of the performance improvement.

2 Geometric hashing and LLAH

2.1 Geometric hashing [8]

We explain the storage and retrieval processes of GH and LLAH in the case under a similarity transformation.

2.1.1 Storage process

GH describes the object which has undergone a certain geometric transformation using invariant coordinate systems. We explain the storage process of it. Though Fig. 2 is an illustration of the retrieval process, it will help understand the storage process because the most are common.

To begin with, feature vectors are extracted from the image of the object. Two of them are chosen, and a pair of bases is defined as shown in Fig. 2. In the figure, a basis $\boldsymbol{P}_2 - \boldsymbol{P}_1$ and its orthogonal one (denoted as $(\boldsymbol{P}_2 - \boldsymbol{P}_1)^{\perp}$) are defined. Then, the rest of the feature points ($\boldsymbol{P}_3$, $\boldsymbol{P}_4$ and $\boldsymbol{P}_5$) are projected to the invariant coordinate system spanned by the pair of bases ($\boldsymbol{P}_2 - \boldsymbol{P}_1$ and $(\boldsymbol{P}_2 - \boldsymbol{P}_1)^{\perp}$). The invariant coordinate system is divided (quantized) into subregions in advance. Thus, the object ID and a basis-set ID are stored into the each corresponding subregion.

The storage process finishes after the procedure above is carried out for all the invariant coordinate systems spanned by all the pairs of bases and for all the objects to be stored.

2.1.2 Retrieval process

We explain the retrieval process of GH with the illustration of Fig. 2. The initial phase of the retrieval process is almost the same as that of the storage one.

To begin with, feature vectors are extracts from the image of the object. Two of them are chosen, and a pair of bases is defined as shown in Fig. 2. Then, the rest of the feature points are projected to the invariant coordinate system spanned by the pair of bases. Each projected feature vector corresponds to a subregion of the invariant coordinate system. The votes for the corresponding pairs of the object ID and the basis-set ID are made.

The procedure above is carried out for all the invariant coordinate systems. The pair of the object ID and the basis-set ID with the highest vote determines the corresponding object. In the illustration of Fig. 2, the object #2 is retrieved because the combination of the object #2 and the basis-set #3 obtains the highest vote. The process can quit when the corresponding object is obviously determined.

2.2 LLAH [10]

2.2.1 Storage process

We explain the storage process of LLAH. Like Fig. 2, Fig. 3 is an illustration of the retrieval process. However, it will also help understand the storage process because the most are common.

After feature points are extracted, LLAH calculates feature vectors which describe the arrangements of the m nearest feature points of the feature point of interest. In Fig. 3, the point of interest is $\boldsymbol{P}_3$, and the $m\ (=4)$ nearest ones are $\boldsymbol{P}_1$, $\boldsymbol{P}_2$, $\boldsymbol{P}_4$, and $\boldsymbol{P}_5$. The feature vectors of $\boldsymbol{P}_3$ are calculated as follows. The m nearest points are ordered in clockwise rotation as $\boldsymbol{P}_5, \boldsymbol{P}_4, \boldsymbol{P}_2, \boldsymbol{P}_1, \boldsymbol{P}_5, \cdots$. With three points denoted as A, B and C, a similarity invariant $\frac{\mathrm{AC}}{\mathrm{AB}}$ is calculated, where AB stands for the line segment between A and B. Thus, a similarity invariant is calculated from a set of three points. By sliding the points to regard A, B and C in clockwise rotation, $\binom{m}{3}\ (=4)$ similarity invariants are calculated (i.e. a, b, c and d in Fig. 3). By combining $\binom{m}{3}$ invariants using clockwise

rotation, m vectors are created (i.e. $abcd$, $bcda$, $cdab$ and $dabc$ in Fig. 3). In the storage process, one of them is chosen arbitrarily. Then, a hash value is calculated from the chosen feature vector. Finally, the object ID is stored into the corresponding cell of the hash table in the hash value.

The storage process finishes after the procedure above is done selecting each feature point of all the objects to be stored as a point of interest.

For simplicity, in the above explanation, we omit an important factor of LLAH. Because LLAH creates the feature vectors by combining the information of coordinates of m neighbors (feature points), it is less robust than GH to the appearance and disappearance of feature points. In order to overcome the weakness, LLAH once chooses n ($\geq m$) neighbors, and then chooses m neighbors from the n neighbors. This makes it possible to perform a robust retrieval with a small increase of computational cost. This is described in Sec. 3.2.4.

2.2.2 Retrieval process

We explain the retrieval process of LLAH. The initial phase of the retrieval process is almost the same as that of the storage one.

After feature points are extracted, LLAH calculates feature vectors of the point of interest as in the storage process. With the illustration of Fig. 3, m vectors are created as in the storage process (i.e. $abcd$, $bcda$, $cdab$ and $dabc$). Though only one of them is used in the storage process, all of them are used in the retrieval process. Hash values are calculated from the feature vectors, and votes for the object IDs are made [1].

The procedure above is done selecting each feature point as a point of interest. The object ID with the highest vote determines the corresponding object. In the illustration of Fig. 3, one vote for the object #2 and three votes for *nothing* are made. Thus, the object #2 is retrieved. The vote for nothing is caused by an empty cell. The empty cells appear because a high-dimensional feature vector is employed to calculate a hash value described in Sec. 3.2.3.

Like GH, the process can quit when the corresponding object is obviously determined, for example, in the case that the difference between the highest vote and the second highest one is large.

3 Primary factors of the performance improvement of LLAH

Computational cost (processing time) and required amount of memory of LLAH decrease greatly compared to those of GH. In this section, we discuss four primary factors of it.

3.1 Problems of geometric hashing

Before discussing LLAH, we discuss the problems of GH first. In many practical situations, GH is unusable because it requires large amount of computation and memory. We

[1] Storing not only object IDs but also feature point IDs enables us to know the correspondence of feature points between the query and stored images with a slight increase of computational and memory resources. We have applied this to a real-time document image retrieval system [5].

consider three causes.

For the preparation, let N be the average number of feature points in an image. M be the number of stored images. Let b be the number of bases used in GH. b is determined by the class of invariants, i.e. $b = 2$ for a similarity transformation, $b = 3$ for an affine transformation, and $b = 4$ for a perspective transformation.

The first cause is that the computational cost based on the number of feature points (N). The original GH [8] requires the computational cost of $O(N^{b+1}M)$ in the storage process and $O(N^{b+1})$ in the retrieval process. Required amount of memory is the same as the required computational cost in the storage process. Therefore, even a modern computer cannot handle only several hundred points per image in practical time. Some variants of GH which reduce computational cost by thinning out feature points or basis-pairs to process probabilistically have been proposed. However, such probabilistic reduction of computational cost cannot avoid reduction of retrieval accuracy because computational cost and retrieval accuracy are in a trade-off relationship [4].

The second cause is hash collisions. The number of stored entries (sets of an object ID and a basis-set ID) in the hash table of GH is $N^{b+1}M$. If an invariant coordinate system is divided (quantized) into k bins per axis, k^b subregions per coordinate system exist. Therefore, the number of entries in a subregion is $N^{b+1}M/k^b$ in average. This is the same as the number of collisions. This means that one hash value causes as many as $N^{b+1}M/k^b$ votes! Since it is not easy to imagine how much $N^{b+1}M/k^b$ is, let us calculate it when $N = 100$, $M = 100$, $b = 3$ and $k = 10$. We can find the answer is 10^7! Even if k is changed to 100, the value is still as much as $10,000$. This is a major reason that GH cannot employ even moderate size of N and M in practice. A simple solution to avoid such enormous computation cost is to increase k. However, this also increases the possibility that a different bin is selected by a same feature vector by a noise because the hash table is divided finer. As the result, retrieval accuracy is also reduced.

The third cause is selection cost of the highest vote. GH has N^bM bins[2] in the voting table. In order to find the bin with the highest vote, all the bins have to be examined. Therefore, the computational cost of $O(N^bM)$ is required.

3.2 LLAH as the outcome of step-by-step improvements on the original geometric hashing

In order to reveal the relevance and difference between GH and LLAH, we regard LLAH as the outcome of step-by-step improvements on the original GH. That is, four improvements on the original GH derive LLAH and resolve the problems mentioned in Sec. 3.1.

3.2.1 Introduction of point of interest and m neighbors

The original GH uses all feature points in an image. We reduces the computational cost by reducing them. As mentioned in Sec 3.1, probabilistic reduction of feature points and basis-pairs cannot avoid retrieval accuracy. Thus, we define each feature point as a point of interest. Besides, only m neighbors of the point of interest are used for calculation. They reduce computational cost as much as $O(m^bNM)$ in the storage process and $O(m^bN)$ in the retrieval process.

[2] N^bM comes from the number of the object IDs (M) and the number of the basis-set IDs (N^b).

Note that the most important matter is that the same m points are obtained in the storage and retrieval processes. Obviously, if the image is taken from an oblique angle, retrieval accuracy can decrease due to change of m neighbors. This problem is discussed in Secs. 3.2.4.

3.2.2 Non-probabilistic reduction of invariants

We resolve the first problem mentioned in Sec. 3.1 by reducing the number of invariants. LLAH attaches great importance to select the same feature points to calculate invariants in reproducible manner. This enables to decreases computational cost without reducing retrieval accuracy. In order to realize it, we introduce clockwise order because

> clockwise order of feature points around the point of interest is invariant to geometric transformations.

By introducing the order, selectability of a sequence of feature points is greatly reduced. As an example, let's think about choosing two out of three points. Let A, B and C be the three points. Without introducing order, there are six choices such as AB, AC, BA, BC, CA and CB. However, by introducing an order like A $\rightarrow$ B $\rightarrow$ C which means AB, AC and BC are allowed to choice, but neither BA, CA nor CB is not. This decreases six choices to three. And, we can employ the same three choices with the order (reproducibility).

We explain the procedure of calculating invariants in *order-introduced* LLAH. Firstly, m neighbors of the point of interest are selected, and ordered in clockwise. Note that the beginning of the ordered points can be arbitrarily chosen because we handle only m points; Testing m possible beginning points increases the computational cost only m times and a constant growth of the cost is trivial compared to an exponential growth. Secondly, $b+1$ points are chosen form the m points keeping the order. Thirdly, an invariant is calculated with the arrangement of the $b+1$ points. Since a $(b+1)$-combination taken from the m points is $\binom{m}{b+1}$, $\binom{m}{b+1}$ invariants are calculated for a point of interest. The above procedure is repeated by selecting one of N points in an image as a point of interest. Thus, we have $\binom{m}{b+1}N$ invariants for an image. Finally, the computational cost in the storage process is $O\left(\binom{m}{b+1}NM\right)$. This is because $\binom{m}{b+1}N$ invariants are calculated for each of M objects. The cost in the retrieval process is $O\left(\binom{m}{b+1}mN\right)$. This is because $\binom{m}{b+1}N$ invariants are calculated for each of m beginning points.

For comparison, we explain how many invariants GH calculates. GH calculates invariants by projecting feature points to an invariant coordinate system since the coordinates on the invariant coordinate system are invariants. Since the dimensionality of the coordinate system is 2, two invariants for each point are calculated. Thus, the number of calculated invariants is given as

$$\text{(The number of basis-sets)} \times \text{(the number of projected points)} \times 2.$$

The number is the same order as the computational cost of a retrieval. That of the original GH is $O(N^b) \times (N-1) \times 2 = O(N^{b+1})$ and that of the method introduced in Sec. 3.2.1 is $O(Nm^{b-1}) \times (m-b+1) \times 2 = O(m^bN)$. This means that the computational cost of GH and LLAH is proportional to the number of calculated invariants. Therefore, we confirmed LLAH reduced the cost since LLAH reduced the number of calculated invariants.

3.2.3 Introduction of high-dimensional feature vector

We resolve the second problem mentioned in Sec. 3.1, which is the collision problem. The problem that many collisions occur in GH comes from low discrimination ability. As mentioned in Sec. 3.1, the number of stored entries is as many as $N^{b+1}M$, while the size of the hash table (the number of subregions) is only k^b. Thus, as a solution, LLAH enhances the discrimination ability by enlarging the size of the hash table. For the sake of that, the discrimination ability of invariants is also enhanced. Thus, LLAH combines $\binom{m}{b+1}$ invariants and creates a $\binom{m}{b+1}$-dimensional feature vector. The alignment is determined by use of clockwise order. Since each invariant is quantized into k discrete values, the size of the hash table is as much as $k^{\binom{m}{b+1}}$ at most.

We show an actual data that a large size hash table reduces collisions. In the case that LLAH is applied to a document image retrieval task[3], very sparse hash tables were obtained: only 2.95% of hash bins were nonempty for 1,000 pages (images), and 19.7% for 10,000 pages. In the nonempty bins, the average number of collisions was less than two.

Such a sparse hash table contributes to not only reduction of processing time, but also robust retrieval. As mentioned above, combining the information of feature points requires a complete match of $\binom{m}{b+1}$ discrete values of vector elements. This can cause failure of a vote because the probability two feature vectors match is much lower than the one that two elements of vectors match. Even so, if the hash table is sparse, most of wrong hash values do not harm due to empty bins.

As a side effect of combining invariants, we have resolved the third problem mentioned in Sec. 3.1. Due to high discrimination ability of a feature vector, the basis-set IDs are no longer required for discrimination. Thus, the form of the voting table has been changed into the one in Fig. 3. This reduces the computational cost of finding the bin with the highest vote from $O(N^bM)$ down to $O(M)$.

3.2.4 Robustness by using "m neighbors from n neighbors rule"

As mentioned before, a feature vector is weak to disturbances such as the appearance and disappearance of feature points, and change of skew angle of an image. In order to overcome the weakness, LLAH once chooses n ($\geq m$) neighbors, and then chooses m neighbors from the n neighbors. We call this "m neighbors from n neighbors rule". The rule creates $\binom{n}{m}$ possible choices of m neighbors. By employing all of them, $\binom{n}{m}$ times of feature vectors are created. This makes it possible to perform a robust retrieval because the probability that the same m feature points (i.e., a feature vector) are chosen in the storage process and the retrieval process is not 0 even if up to $n-m$ points are lost.

Introducing "m neighbors from n neighbors rule" changes the computational cost of LLAH to $O\left(\binom{n}{m}\binom{m}{b+1}NM\right)$ in the storage process, and $O\left(\binom{n}{m}\binom{m}{b+1}mN\right)$ in the retrieval process. Though the rule increases computational cost $\binom{n}{m}$ times, a good choice of n and m does not increase it so much. For example, in the case of $n=8$ and $m=7$, $\binom{8}{7}=8$, and in the case of $n=10$ and $m=8$, $\binom{10}{8}=45$.

[3] An affine invariant was used. The size of the hash table was approximately 2^{27}. $n=7$, $m=6$, $k=15$ were used for parameters.

4 Discussion

LLAH combines the information of feature points to calculate invariants. There are some methods which combines the information of feature points. In this section, we compare LLAH and them to clear the novelty of LLAH.

There are some improved methods of GH which employ other primitive features than a feature point: a line segment [15] and a chain of connected line segments [13]. The latter reduces the computational cost more; [13] achieved $O(N)$ for N line segments. However, a chain of connected line segments is not available for most target objects. The advantage of LLAH is ability to combine discontiguous feature points in a defined order.

For a robust match of local features, [14] introduces a geometric constraint such that angles between feature points should be in a range, though it does not use GH. Though the constraint improves discrimination ability, the angles change under a perspective and an affine transformation. To the contrary, the constraint of LLAH (i.e., order) is invariant even if the transformation is a perspective transformation.

5 Conclusion

In this paper, we described the important factors of improving performance of the geometric hashing (GH) in both retrieval speed and required amount of memory. For the sake of it, we compared GH and locally likely arrangement hashing (LLAH). Consequently, we obtained four primary factors: (1) introduction of point of interest and m neighbors, (2) non-probabilistic reduction of invariants, (3) introduction of high-dimensional feature vector, and (4) introduction of "m neighbors from n neighbors rule." The most important one is to use non-probabilistic selection of feature points because probabilistic one cannot escape from the trade-off relationship between computational cost and retrieval accuracy.

In another aspect, the contribution of this paper is the first detailed analysis of LLAH. We pointed out that LLAH breaks the thetrade-off relationship and resolves a collision problem of hashing.

Future work includes an evaluation of robustness of LLAH against disturbances such as the appearance and disappearance of feature points.

Acknowledgment

This research was supported in part by the Grant-in-Aid for Scientific Research (B) (19300062) from Japan Society for Promotion of Science.

References

[1] Sunil Arya, David M. Mount, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45(6):891–923, 1998.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *LNCS (Proc. ECCV'06)*, volume 3951, pages 404–417, May 2006.

[3] Mayur Datar, Piotr Indyk, Nicole Immorlica, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. 20th SCG*, pages 253–262, 2004.

[4] Michael Hoffman and Michael Lindenbaum. Some tradeoffs and a new algorithm for geometric hashing. In *Proc. ICPR'98*, pages 1700–1704, 1998.

[5] http://imlab.jp/LLAH/.

[6] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. 30th ACM STOC*, pages 604–613, 1998.

[7] Yan Ke and Rahul Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proc. CVPR'04*, volume 2, pages 506–513, 2004.

[8] Yehezkel Lamdan and Haim J. Wolfson. Geometric hashing: a general and efficient model-based recognition scheme. In *Proc. ICCV'88*, pages 238–249, 1988.

[9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Proc. ICCV'04*, 60(2):91–110, 2004.

[10] Tomohiro Nakai, Koichi Kise, and Masakazu Iwamura. Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval. In *LNCS (Proc. DAS'06)*, volume 3872, pages 541–552, February 2006.

[11] David Nistér and Henrik Stewénius. Scalable recognition with a vocabulary tree. In *Proc. CVPR'06*, volume 2, pages 2161–2168, 2006.

[12] Ruth Nussinov and Haim J. Wolfson. Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques. In *Proc. Nat'l Acad. Sci. U. S. A.*, volume 88, pages 10495–10499, 1991.

[13] Charles A. Rothwell, Andrew Zisserman, David A. Forsyth, and Joseph L. Mundy. Using projective invariants for constant time library indexing in model based vision. In *Proc. BMVC'91*, 1991.

[14] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE Trans. PAMI*, 19(5):530–535, May 1997.

[15] Franc C. D. Tsai. Robust affine invariant matching with application to line features. In *Proc. CVPR '93*, pages 393–399, June 1993.

[16] Haim J. Wolfson and Isidore Rigoutsos. Geometric hashing: an overview. *IEEE Comp. Sci. and Eng.*, 4(4):10–21, 1997.