

Camera tracking by online learning of keypoint arrangements using LLAH in augmented reality applications

Hideaki Uchiyama · Hideo Saito · Myriam Servières · Guillaume Moreau

Received: 20 November 2009 / Accepted: 23 September 2010 / Published online: 7 October 2010
© Springer-Verlag London Limited 2010

Abstract We propose a camera-tracking method by on-line learning of keypoint arrangements in augmented reality applications. As target objects, we deal with intersection maps from GIS and text documents, which are not dealt with by the popular SIFT and SURF descriptors. For keypoint matching by keypoint arrangement, we use locally likely arrangement hashing (LLAH), in which the descriptors of the arrangement in a viewpoint are not invariant to the wide range of viewpoints because the arrangement is changeable with respect to viewpoints. In order to solve this problem, we propose online learning of descriptors using new configurations of keypoints at new viewpoints. The proposed method allows keypoint matching to proceed under new viewpoints. We evaluate the performance and robustness of our tracking method using view changes.

Keywords LLAH · Feature descriptor · Camera tracking · Augmented reality

1 Introduction

Camera tracking remains an open and active fundamental problem in the computer vision domain. In particular, augmented reality (AR) applications need real-time processing and robust camera tracking in order to place virtual objects in an actual scene. To meet the requirement, several approaches have been proposed.

Fiducial markers have been widely developed since a long time, and they have been used in many AR frameworks (Kato and Billinghurst 1999; Fiala 2005; Wagner et al. 2008a, b). In fact, these marker systems are used in practical applications in companies and industries (Pentenrieder et al. 2007).

Recently, the focus of research has been shifting toward the use of natural features from actual environments such as edges and feature points, because fiducial markers may not be available (for instance, because of installation permission in outdoor environments, size-based issues, and application constraints). For example, edge-based approaches can be found in model-based tracking (Drummond and Cipolla 1999), initialization of tracking (Kotake et al. 2007) and visual SLAM (Klein and Murray 2008) using boundaries of a room and the rims of a nontextured object. The keypoint (feature point) matching-based approach is also becoming common, owing to the development of local descriptors such as SIFT (Lowe 2004). In addition, the computational complexity of this approach is drastically decreasing and allows for the implementation of the method on a mobile device with a low-speed CPU and less memory, as in Phony SIFT (Wagner et al. 2008a, b).

Even though these remarkable developments on local descriptors have already been achieved, these approaches cannot be applied to our target objects such as maps that include only intersections as simple circular dots

H. Uchiyama (✉) · H. Saito
Keio University, 3-14-1 Hiyoshi, Kohoku-ku 223-8522, Japan
e-mail: uchiyama@hvrl.ics.keio.ac.jp

H. Saito
e-mail: saito@hvrl.ics.keio.ac.jp

M. Servières · G. Moreau
Ecole Centrale de Nantes - CERMA IRSTV, Rue de la Noë,
BP 92101, 44321 Nantes Cedex 3, France
e-mail: myriam.servieres@ec-nantes.fr

G. Moreau
e-mail: guillaume.moreau@ec-nantes.fr

(Uchiyama et al. 2008, 2009) and text documents having locally repetitive patterns (Uchiyama and Saito 2009), because rich textured objects are necessary for the descriptors. Instead of the local patch-based descriptors, the local arrangement of keypoints is shown as a distinctive descriptor in such cases.

In this study, we describe a camera-tracking method for our target objects, which cannot be tracked by traditional methods. We use LLAH (Nakai et al. 2005) to describe the local arrangement of keypoints. Because the local arrangement might be modified when the viewpoint changes, we propose a method for learning the new configuration of keypoints at new viewpoints, in order to handle a large range of viewpoint changes.

The rest of this paper is arranged as follows. The next section describes related studies regarding keypoint matching. In particular, the details of LLAH are highlighted in Sect. 3, because LLAH is an important component in our method. Sect. 4 explains our main contribution to the learning process with LLAH. Sect. 5 demonstrates the performance and robustness of our method. Sect. 6 discusses our conclusions and future work.

2 Related works

The entire process of keypoint matching can be divided into three parts: extraction, description, and matching.

For keypoint extraction, Harris corner (Harris and Stephens 1988) and FAST corner (Rosten and Drummond 2006) have been proposed for extracting keypoints that have a different appearance from their neighboring pixels. These methods can be applied to multiscale images to take into account scale changes. There have been several approaches for extracting scale-invariant feature points, such as the difference of Gaussians (Lowe 2004), gradient locations and orientation histograms (Mikolajczyk and Schmid 2005), and basic Hessian-matrix approximation (Bay et al. 2008).

A keypoint descriptor is a high dimensional vector computed from the local neighbor region of the keypoint in order to construct a discriminative power. Descriptors such as SIFT (Lowe 2004) and SURF (Bay et al. 2008) have been designed to be invariant to illumination, scale, rotation, and translation changes. Because they require high computational power, several attempts have been made to accelerate the computation of such descriptors. In particular, it is important to run at interactive frame rates in order to provide real-time applications and interaction with the user in AR systems. Among them, Sinha et al. (2006) has implemented SIFT on a GPU in order to use parallel processing. Wagner et al. (2008a, b) has proposed Phony SIFT, which is a mobile phone version of SIFT that

removes some computational costs related to keypoint extraction and descriptor computation.

The matching of descriptors can be addressed as a nearest neighbor searching problem between high dimensional vectors. Approximate nearest neighbor is a searching method based on kd-trees and box-decomposition trees (Arya et al. 1998). Because a distance computation is performed for the comparison between two vectors, the retrieval cost depends on the dimension of the vector. Locality sensitive hashing (LSH) is another approximate searching method based on probabilistic dimension reduction with a hash scheme (Datar et al. 2004). The computational cost of LSH is always $O(1)$, but the nearest neighbor points might not be found. The design of the hash function remains an important issue in order to efficiently store data, which requires having as few collisions as possible in the hash table. Nister and Stewenius have proposed a recursive k-means tree as a vocabulary tree for quick retrieval (Nister and Stewenius 2006). Lepetit et al. have treated the matching of descriptors as a classification problem (Lepetit et al. 2004).

Local descriptors are well suited to match keypoints with rich texture patterns. In contrast, these descriptors cannot be applied to our targets, such as intersection maps and text documents. For example, the local textures in the intersection maps are the same because the texture is only composed of identical circular dots. In this case, descriptors such as SIFT and SURF do not work well because local areas do not have enough discriminative power to be distinct from other areas. In addition, in text documents, local textures are almost identical and cannot be described by SIFT and SURF. Instead of local patch-based descriptors, we promote the use of descriptors that consider the geometrical relationship between keypoints, which has already been proposed in studies of document image retrieval (Hull et al. 2007; Nakai et al. 2005).

Hull et al. have proposed to use the horizontal connectivity of word lengths as a descriptor (Hull et al. 2007). Word length refers to the number of characters and is linked with the previous and next word lengths. Because word lengths are very sensitive to viewpoint changes, this descriptor is valid only when a user captures an image where the camera is orthogonal to the document and close enough to the paper. Text lines must also be parallel to the lower side of the image.

Nakai et al. (2005) have proposed keypoint matching using the local arrangement of keypoints for document image retrieval, called LLAH. The objective is to quickly find a document relevant to a query image from a database containing numerous documents. LLAH is an improved method of geometric hashing (GH) (Lamdan and Wolfson 1988) in terms of memory use and computational cost. Because the computational cost of GH considering

perspective distortion is $O(N^5)$, where N is the number of keypoints in a query, it is difficult to apply it to real-time applications, such as AR systems. To solve the computational cost problem, LLAH focuses only on local geometry with neighbor keypoints. However, the descriptors of local arrangement in LLAH are invariant to the narrow range of viewpoints because the arrangement is changeable with respect to viewpoints. In order to handle a large range of viewpoints, we merge the online learning of the new configuration of keypoints into LLAH.

3 Descriptors in LLAH

In this section, we explain the descriptors in LLAH because our method is mainly based on these descriptors (Nakai et al. 2006).

In Fig. 1, x is an example target keypoint. First, the n nearest neighbor points around x are selected as **abcdefg** ($n = 7$). The order to select the n points should be defined beforehand. For example, we select from **a** in a counter-clockwise fashion based on the reference axes, as illustrated in Fig. 1.

Next, m points out of n points are selected as **abcde** ($m = 5$). From these m points, one descriptor is computed. Because a descriptor is computed on each combination, a keypoint has ${}_n C_m = \frac{n!}{m!(n-m)!}$ descriptors.

From m points, l points are selected for computing a geometrical invariant. As the invariant, Nakai et al. selected a cross ratio as a perspective invariant (Nakai et al. 2005) and a ratio of two triangles as an affine invariant (Nakai et al. 2006). They concluded that the affine invariant was better because the perspective invariant was not stably computed. Even though the perspective invariant is utilized, the available range of keypoint matching is almost same as that by the affine invariant. In this work, we select the affine invariant ($l = 4$).

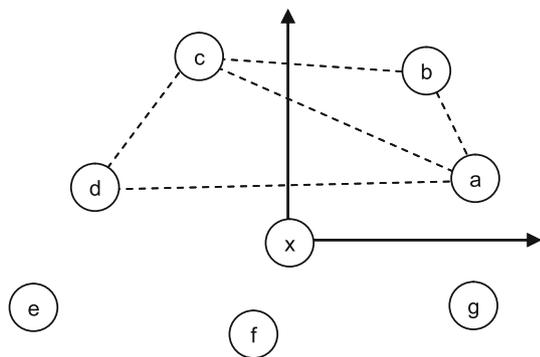


Fig. 1 Descriptors in LLAH. The descriptors of a keypoint are computed from the combination of the ratio of two triangles

In Fig. 1, four points are selected as **abcd** to compute a ratio of two triangles. For a hashing scheme, the value of the ratio is quantized into an index using a distribution histogram created in prior experiments (Nakai et al. 2005). The histogram is segmented into the number of quantization level to assign an integer number at each segment. Because the number of the combination to select four points is equivalent to the dimension of the descriptor, the dimension is ${}_m C_4$.

For quick retrieval, a hash scheme is adopted. The descriptor is converted into an index using following equation:

$$\text{Index} = \left(\sum_{i=0}^{{}_m C_l - 1} r_{(i)} k^i \right) \bmod H_{\text{size}} \quad (1)$$

where $r_{(i)}$ ($i = 0, 1, \dots, {}_m C_l - 1$) are quantized values of geometrical invariants, k is the quantization level, and H_{size} is the pre-defined hash size. As a result, each keypoint is stored in a hash table as (Index, Document ID + Keypoint ID). The table has been pre-defined to a large size to access the element of each index by $O(1)$. In addition, a keypoint database is prepared to store a 2D coordinate of each keypoint as (Document ID + Keypoint ID, 2D coordinate). In our implementation, a document ID and a keypoint ID are represented by a 32 bit ID by assigning 16 bits to both IDs.

In the hash table, a collision such that the same index is computed for different keypoints may occur. Because the discriminative power of such an index is considered to be too low, a keypoint is not stored at the index.

4 Proposed method

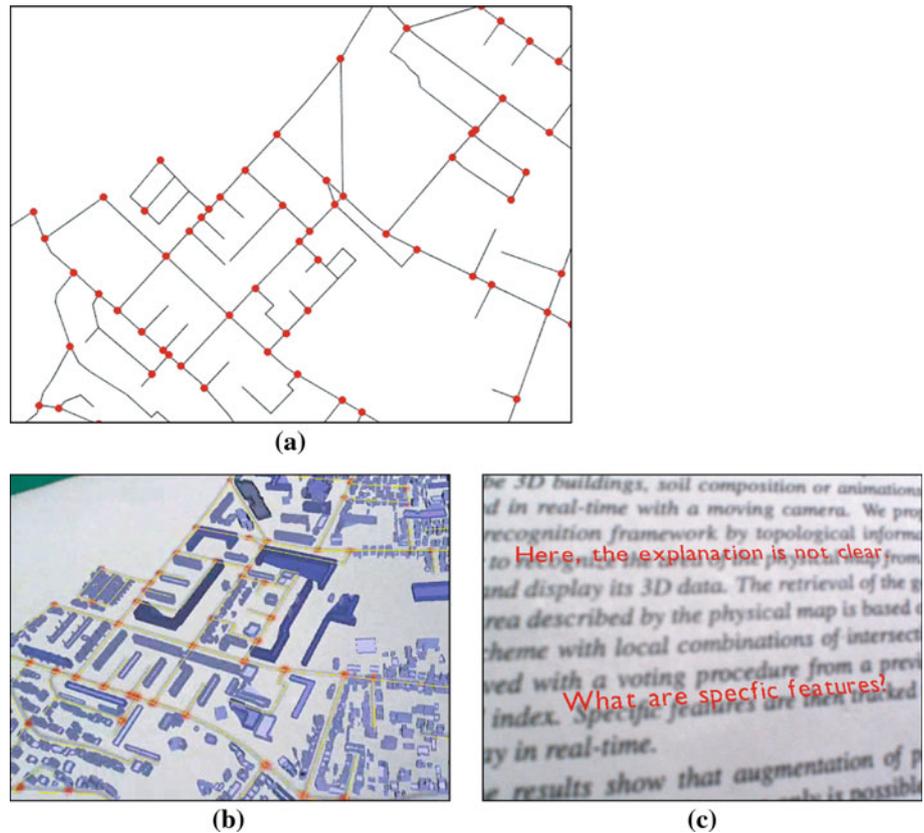
4.1 Target objects and their keypoints

Our target objects for augmented reality applications are intersection maps (Uchiyama et al. 2008, 2009) and text documents (Uchiyama and Saito 2009). Figure 2a is an example of intersection maps generated from GIS, and Fig. 2b is the visualization of 3D data on GIS. This application was developed to achieve a novel geo-visualization by relating GIS and paper maps. Also, Fig. 2c illustrates a virtual annotation system implemented as augmented reality on a document.

In these applications, local patch-based tracking methods do not work because local texture patterns are not sufficiently distinctive. For this reason, we sought another approach using the local arrangement of keypoints as a descriptor.

For keypoint extraction, the intersections are extracted by color extraction because the intersections have a specific

Fig. 2 Target objects.
a An example of intersection maps. **b** 3D building visualization on the intersection map. **c** Virtual annotation on a document



color, such as red. From text documents, word regions are extracted using adaptive thresholding in the same way as (Nakai et al. 2006). A keypoint is the center of each region. Compared to normal textures, local texture patterns are similar, but keypoints are stably extracted.

4.2 Initialization

In our applications, a camera pose with respect to a 2D printed paper (map or document) is tracked. Before pose tracking, an initial pose is estimated as an initialization. Because the initialization needs an initial hash table (descriptor database) and keypoint database, these databases are prepared as follows.

To create an intersection map, the 2D distribution of intersections is exported from GIS. The distribution is equivalent to the top view of a map. By dealing with intersections as keypoints, the initial databases are created. For a text document, a document image is prepared from a digital document, such as a PDF. The image is also regarded as the top view of the document. By extracting keypoints from the image, the initial databases are created.

Because the initial databases are equivalent to the databases from a top view, a camera needs to be set at the top view for the initialization.

The process is the same as that of document image retrieval by (Nakai et al. 2006). In the initialization, keypoints are extracted from a captured image (keypoint extraction), and their indices (descriptors) are computed (descriptor computation). For each keypoint, a histogram of keypoint IDs is generated by retrieving keypoint IDs from the indices in the initial hash table. By selecting a peak of the histogram, each keypoint in the captured image has a corresponding keypoint in the keypoint database. In order to estimate a camera pose, a homography is computed from the correspondences, because the paper is set on a plane. Because there are outliers in the correspondences, we use RANSAC (Fischler and Bolles 1981) to remove outliers and compute a refined homography.

4.3 Online learning process

A tilted camera pose cannot be computed from the initial hash table because the keypoint matching by LLAH fails. In order to achieve wide base-line keypoint tracking, the online learning of descriptors is proposed.

The flowchart of pose tracking is illustrated in Fig. 3. From keypoint extraction to pose estimation, the process is the same as the initialization. After pose estimation, the process moves to the descriptor update step. Because the

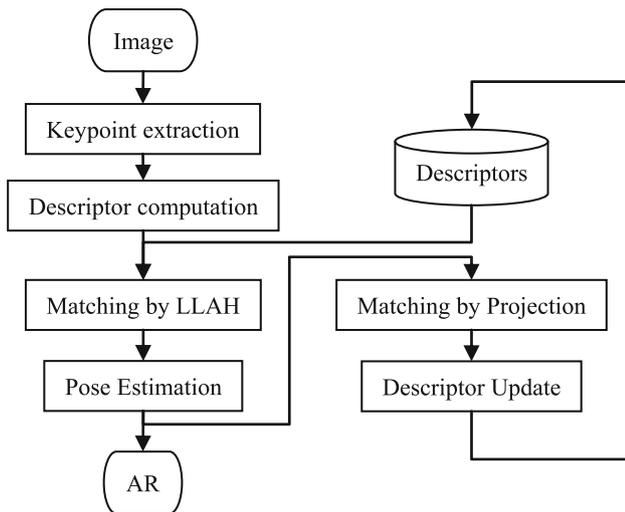


Fig. 3 Online learning process. As well as the initialization, a camera pose is estimated by matching by LLAH. After pose estimation, the keypoints in the keypoint database are projected onto the captured image to find correspondences in the image. If a correspondence is established, a descriptor update is performed

camera pose is computed, keypoints in the keypoint database can be projected onto the image. By computing the distance between a projected keypoint and keypoints in the image, some of the keypoints in the image can get corresponding points in the keypoint database as matching by projection. For each keypoint for which correspondence is established, a descriptor update is performed.

4.4 Neighbor keypoints’ selection

As described in Sect. 3, the neighbors of each keypoint are necessary in order to compute the descriptors. If we compute the distances for all keypoints from a keypoint, the computational cost is $O(N^2)$, where N is the number of keypoints. The computation of all possible distances would imply larger computational costs. To limit the computational costs in the neighbor keypoints’ selection, we limit the searching of candidates for distance computation to limited neighbor areas, in order to search neighbor keypoints efficiently.

As a pre-processing phase, the captured image is divided into square regions by segmenting them at a uniform interval, as illustrated in Fig. 4. When the keypoints are extracted in the captured image, we compute the region to which each keypoint belongs. In addition, each region maintains the list of keypoints included in that region.

When we search the neighbor keypoints of a target keypoint, we collect potential candidates from the surrounding regions. For example, if a target keypoint belongs to **a** in Fig. 4, the candidates are extracted from **abcdefghi**. If the number of candidates is less than n in Sect. 3, we

		j	k	l	m	n	
		y	b	c	d	o	
		x	i	a	e	p	
		w	h	g	f	q	
		v	u	t	s	r	

Fig. 4 Neighbor points selection. The image region is divided into square regions beforehand. If a keypoint is extracted in **a**, candidates of neighbor points are collected from **a** to **i**. If they are not sufficient, the candidates are collected from **j** to **y**

collect more candidates from more surrounding regions. When the number is more than n , the neighbor points are selected from the candidates by computing each distance.

4.5 Matching by projection

In the pose estimation, we compute homography **H** as:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \sim \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2}$$

where (X, Y) is a keypoint in the keypoint database and (x, y) is a keypoint in the image.

After the homography computation with RANSAC, we can obtain two types of outliers, as follows:

- Volatile keypoints by the instability of the keypoint extraction or motion blur.
- Keypoints stored in the keypoint database, but their descriptors are changed because of the narrow range invariance of the descriptors.

We ignore the former outliers because they are not useful for keypoint matching. For the latter outliers, a descriptor update is performed in order to keep these outliers as inliers.

When the homography is successfully computed, keypoints in the database can be projected onto the captured image by using inverse homography as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim \mathbf{H}^{-1} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{3}$$

where (x', y') is a projected keypoint from the database. For each outlier in the image $\mathbf{x} = (x, y)$, we compute the distances between each projected keypoint $\mathbf{x}' = (x', y')$ to find the nearest one. If the distance $|\mathbf{x} - \mathbf{x}'|$ is less than a threshold (usually two pixels), the outlier is matched with the nearest projected keypoint to assign the keypoint ID of the projected keypoint.

4.6 Descriptor update

For all keypoints extracted from the image, nC_m indices have already been computed in the descriptor computation. If a keypoint is an outlier, NULL is stored at some of the indices, because the indices (descriptors) were never computed. Because NULL means an empty index, we can update this index.

In Fig. 5, some of the computed indices have NULL. For these indices, we fill NULL with a keypoint ID computed from matching by projection. The updated indices can be utilized after the next frame or later, so that outliers in the current frame become inliers in matching by LLAH. If there is a keypoint ID in the indices, the update is not performed.

The update is performed using a threshold for the number of the inliers after RANSAC-based homography computation. If there are sufficient inliers in a frame, the update is not necessary for next frame. In Sect. 5.2, the influence of a threshold for the hash table is discussed.

Even though we insert keypoint IDs in the hash table, the size of the hash table does not change, because it has been pre-defined to a large size to access each index by $O(1)$. The pre-processing phase usually leads to many empty indices. In order to use those empty indices effectively, we insert keypoint IDs into the empty indices. If an index has already had a keypoint ID, another keypoint ID is not inserted. This computational cost is not affected by the number of inserted indices because of the property of a hash scheme.

This update helps the re-initialization of a camera pose when camera tracking fails. If there is no descriptor update, a camera should be set at the top view for the initialization, as described in Sect. 4.2. With the update, a camera pose can be re-initialized by setting back the camera on the

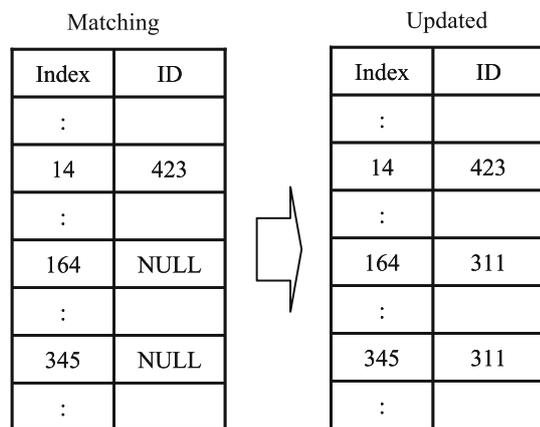


Fig. 5 Update of LLAH. A keypoint has the indices 14, 164, and 345. If the keypoint gets 311 as a keypoint ID from matching by projection, the keypoint ID is inserted into 164 and 345. For 14, the update is not performed

camera trajectory including the top view and last camera pose.

4.7 Parallel processing

In order to develop AR applications, we have to reduce computational costs as much as possible for real-time processing. We thus have to use computer resources effectively. Because we have two processing units in Intel Core 2 Duo architecture, it is important to assign the same load to each processing unit. By measuring each processing time, the proper assignment is determined.

The average processing times were measured from an experiment by using 100 input images, as described in Table 1. In this case, the parameters for LLAH were as follows: $n = 6$, $m = 5$, and $l = 4$.

The costs for keypoint extraction, descriptor computation, and matching by descriptor are influenced by the number of keypoints captured in the image. Because the number of extracted keypoints usually varied between 500 and 600, the times in Table 1 are averages, but also include a variation range. When there are many outliers, the homography computation takes time, especially to get a refined homography. We have limited the number of iterations in RANSAC to 500, thus the maximum time was 22 ms. When there are mostly inliers, this computation takes approximately 2 ms.

Given those results, we have divided our process into two parts. The first thread takes care of “Capture an image” and “Keypoint extraction” while the other does the remaining tasks.

For this parallel processing, we need two memory spaces to store keypoints. First, one memory has keypoints at the t frame extracted by the first thread. The other memory has keypoints at the $t - 1$ frame for the processes of the second thread. After all processes are finished in each thread, both threads are synchronized to copy the memory of the first thread to that of the second thread. If both threads work in the same way, the total cost per frame will be about 30 ms.

Table 1 Processing time

Process	ms
Image capture	10
Keypoints extraction	20 ± 3
Descriptor computation and matching by descriptor	23 ± 4
Homography computation	2 (+20)
Matching by projection	2
Descriptor update	2

We measured each processing time from 100 images. Processes are assigned to each thread depending on the result

5 Experimental results

5.1 Influence of the parameters in LLAH for keypoint matching

Because Nakai et al. (2005, 2006) evaluated only the accuracy of document image retrieval, we evaluate the influence of parameters in LLAH for keypoint matching by LLAH. In this experiment, we prepared a white paper with 100 black circular dots distributed randomly to eliminate the influence of the instability in the keypoint extraction. The initial hash table and keypoint database are prepared from a top view image.

In the parameters of LLAH, we evaluate the influence by two parameters: n and m , which mainly affect the computational costs and the accuracy. The other parameters were optimized to achieve the best result as $l = 4$, $k = 32$ and $H_{\text{size}} = 2^{15} - 1$. In this experiment, we tested the following combinations: $(n, m) = (5, 5), (6, 5), (7, 5), (8, 5),$ and $(7, 6)$.

For each combination, we apply matching by LLAH to the same video capturing the paper from a top view to an inclined view around the center of the paper. We compute the angle between the vector from the center of a document to a camera position and the document plane using the tracking with the descriptor update at every frame as described in Sect. 4.2. The number of inliers after the RANSAC-based homography computation is counted as illustrated in Fig. 6.

First, we investigated the influence of n as $(n, m) = (5, 5), (6, 5), (7, 5),$ and $(8, 5)$. In these cases, the descriptor dimension is ${}_5C_4 = 5$. As illustrated in Fig. 6, $n = 5$ got the least number of inliers. As n increased from 6 to 8, the number of inliers increased because the number of descriptors for a keypoint increased. However, the computational cost is increased as ${}_5C_5 = 1, {}_6C_5 = 6, {}_7C_5 = 21$ and ${}_8C_5 = 56$ in matching by LLAH. Because

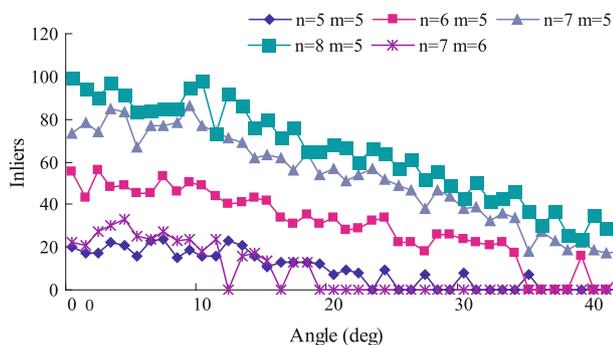


Fig. 6 The number of inliers after the RANSAC-based homography computation. The combinations of $(n, m) = (5, 5), (6, 5), (7, 5), (8, 5),$ and $(7, 6)$ are tested to check the relationship between inliers and angles

this is a trade-off, we have to select parameters depending on computer resources and accuracy requirements.

Afterward, we have examined the influence of the descriptor dimension. We compared these two combinations: $(n, m) = (7, 5)$ and $(7, 6)$ for which we have the following descriptor dimensions: ${}_5C_4 = 5$ and ${}_6C_4 = 15$. As illustrated in Fig. 6, the result of $(n, m) = (7, 6)$ was worse than that of $(n, m) = (7, 5)$ because the discriminative power was too much.

5.2 Behavior of descriptor update

In this section, we investigate the behavior of the descriptor update for camera tracking. Because the descriptor update is performed depending on a threshold as described in Sect. 4.6, the influence of a threshold is investigated.

We tested those cases: no update, update when the number of inliers is less than 20, less than 40, and update at every frame. They are applied to the video utilized in Sect. 5.1. The LLAH parameters are as follows: $n = 6, m = 5, l = 4, k = 32,$ and $H_{\text{size}} = 2^{15} - 1$.

The results are illustrated in Fig. 7. The graph of less than 20 is overlapped with the one with no update from 0 degrees to 27 degrees. Camera tracking does not fail with the update at every frame and update when the number of inliers is less than 40. If the number of inliers is less than 20, a camera pose could not be estimated at some viewpoints. No update could track a camera pose up to 34 degrees. Compared to no update, the descriptor update contributes to camera tracking for a large range of viewpoint changes.

Next, we investigated the behavior of the number of indices in the hash table as illustrated in Fig. 8. The results naturally depend on the number of the descriptor update times. However, collisions in the hash table often happen as the number of updated indices increases. In an application, update of every frame may be acceptable when we

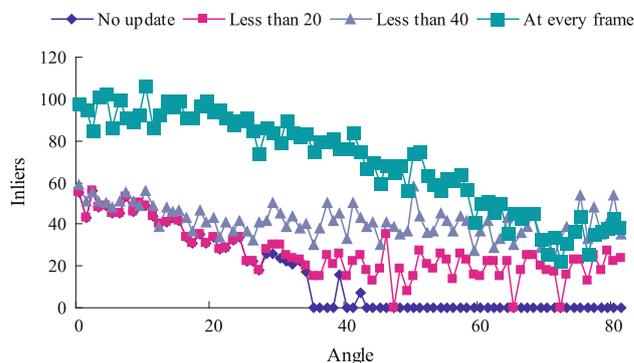


Fig. 7 Descriptor update with respect to a threshold. In case of no update, a camera could be tracked up to 34°. In other cases, a camera pose is tracked

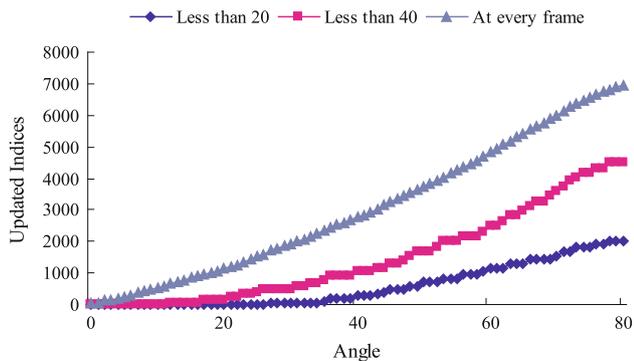


Fig. 8 Updated indices. The number of updated indices depends on the number of descriptor update times

use only one document. If we use many documents, it is important to effectively update descriptors in order to add descriptors in each document. As discussed in (Nakai et al. 2005), the appropriate parameters can be selected from several experiments and system environment.

5.3 Comparison with SURF

For intersection maps, local descriptors do not obviously work because the maps have circular dots and their local textures are same. In addition, these descriptors do not work well for text documents because of their repetitive patterns. To prove that the proposed tracking method is superior than these descriptors for documents, both tracking results on a document are compared. Because SIFT cannot run in real-time for AR systems, SURF implemented on OpenCV (Willowgarage 2010) is selected for the comparison.

In tracking by SURF, a document image is prepared and printed on a A4 paper. The image resolution is selected as 672×950 because this made the best result compared to other resolutions. For each captured image, correspondences in the document image are established by the descriptors in SURF, which can be regarded as matching between the top view and the captured image. In our tracking method, a document image is tracked by descriptor update at every frame. We applied both methods to a video capturing from the top view to the inclined view around the center of a document. For each case, the number of inliers after RANSAC-based homography computation is counted as illustrated in Fig. 9.

In case of SURF, the matching failed after 32 degrees because SURF descriptors are not invariant to the perspective distortion. In contrast, our proposed tracking succeeded at every frame and could estimate the angle of each captured image in Fig. 9. Because our proposed method is a framework for tracking by the descriptor

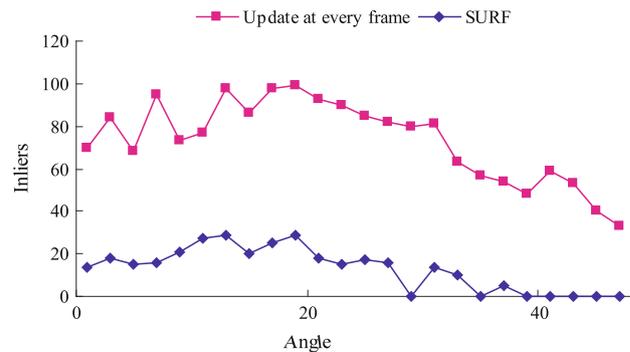


Fig. 9 Comparison with SURF. The number of inliers is compared in each method. Matching by SURF failed at 32 degrees

update, the descriptors in our tracking can be replaced with SURF.

6 Conclusions and future work

We proposed a camera-tracking method based on learning the local arrangements of keypoints for intersection maps and text documents. To describe the arrangements, we use LLAH used in the studies of document image retrieval. Because the descriptors in LLAH are not invariant to a wide range of viewpoints, we proposed a dynamic learning process of the descriptors, called tracking by the descriptor update.

In the updating process, the keypoints in the keypoint database are projected onto a captured image to establish the correspondences between the keypoints in the image and the projected keypoints as matching by projection. For each established correspondence, we insert the keypoint ID into the indices. From the experiment, the descriptor update contributes to a wide range of camera tracking.

In the future work, we have to handle a collision problem in a hash table efficiently. Because the purpose of the proposed method is to track a paper, an intersection map, or a document, the collision did not occur much. If we handle multiple papers, the collisions happen many times, and the structure of the hash table will be a list at each index. In addition, we will develop natural keypoint matching by the local arrangement of keypoints. The local arrangement may help the matching by local descriptors. The keypoint matching method will be utilized in various applications, such as SLAM.

Acknowledgments We thank Dr. Julien Pilet for the discussion. This work was supported in part by Grant-in-Aid for JSPS Fellows and a Grant-in-Aid for the Global Center of Excellence for high-Level Global Cooperation for Leading-Edge Platform on Access Spaces from the Ministry of Education, Culture, Sport, Science, and Technology in Japan.

References

- Arya S, Mount DM, Netanyahu NS, Silverman R, Wu A (1998) An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J ACM* 45:891–923
- Bay H, Ess A, Tuytelaars T, Gool LV (2008) SURF: speeded up robust features. *Comput Vis Image Underst* 110:346–359
- Datar M, Indyk P, Immorlica N, Mirrokni VS (2004) Locality-sensitive hashing scheme based on p-stable distributions. In: *Proceedings of SCG*, pp 253–262
- Drummond T, Cipolla R (1999) Real-time tracking of complex structures with on-line camera calibration. In: *Proceedings of BMVC*, pp 574–583
- Fiala M (2005) Artag, a fiducial marker system using digital techniques. In: *Proc. CVPR*, pp 590–596
- Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun ACM* 24:381–395
- Harris C, Stephens M (1988) A combined corner and edge detector. In: *Proceedings of AVC*, pp 147–151
- Hull J, Erol B, Graham J, Ke Q, Kishi H, Moraleda J, Van Olst D (2007) Paper-based augmented reality. In: *Proceedings of ICAT*, pp 205–209
- Kato H, Billinghurst M (1999) Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In: *Proceedings of IWAR*
- Klein G, Murray D (2008) Improving the agility of keyframe-based SLAM. In: *Proceedings of ECCV*, pp 802–815
- Kotake D, Satoh K, Uchiyama S, Yamamoto H (2007) A fast initialization method for edge-based registration using an inclination constraint. In: *Proceedings of ISMAR*, pp 239–248
- Lamdan Y, Wolfson H (1988) Geometric hashing: a general and efficient model-based recognition scheme. In: *Proceedings of ICCV*, pp 238–249
- Lepetit V, Pilet J, Fua P (2004) Point matching as a classification problem for fast and robust object pose estimation. In: *Proceedings of CVPR*, pp 244–250
- Lowe DG (2004) Distinctive image features from scale-invariant keypoints. *IJCV* 60:91–110
- Mikolajczyk K, Schmid C (2005) A performance evaluation of local descriptors. *PAMI* 27:1615–1630
- Nakai T, Kise K, Iwamura M (2005) Hashing with local combinations of feature points and its application to camera-based document image retrieval: Retrieval in 0.14 second from 10,000 pages. In: *Proceedings of CBDAR*, pp 87–94
- Nakai T, Kise K, Iwamura M (2006) Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval. In: *Proceedings of DAS*, pp 541–552
- Nister D, Stewenius H (2006) Scalable recognition with a vocabulary tree. In: *Proceedings of CVPR*, pp 2161–2168
- Pentenrieder K, Bade C, Doil F, Meier P (2007) Augmented reality-based factory planning—an application tailored to industrial needs. In: *Proceedings of ISMAR*, pp 1–9
- Rosten E, Drummond T (2006) Machine learning for high speed corner detection. In: *Proceedings of ECCV*, pp 430–443
- Sinha S, Frahm J, Pollefeys M, Genc Y (2006) GPU-based video feature tracking and matching. In: *Proceedings of EDGE*
- Uchiyama H, Saito H (2009) Augmenting text document by on-line learning of local arrangement of keypoints. In: *Proceedings of ISMAR*, pp 95–98
- Uchiyama H, Saito H, Servières M, Moreau G (2008) AR representation system for 3D GIS based on camera pose estimation using distribution of intersections. In: *Proceedings of ICAT*, pp 218–225
- Uchiyama H, Saito H, Servières M, Moreau G (2009) AR GIS on a physical map based on map image retrieval using LLAH tracking. In: *Proceedings of MVA*, pp 382–385
- Wagner D, Langlotz T, Schmalstieg D (2008a) Robust and unobtrusive marker tracking on mobile phones. In: *Proceedings of ISMAR*, pp 121–124
- Wagner D, Reitmayr G, Mulloni A, Drummond T, Schmalstieg D (2008b) Pose tracking from natural features on mobile phones. In: *Proceedings of ISMAR*, pp 125–134
- Willowgarage (2010) OpenCV 2.0. <http://opencv.willowgarage.com/wiki/>