Toward Augmenting Everything: Detecting and Tracking Geometrical Features on Planar Objects

Hideaki Uchiyama[*]
INRIA Rennes Bretagne-Atlantique

Eric Marchand[†]
Université de Rennes 1, IRISA,
INRIA Rennes Bretagne-Atlantique

Abstract

This paper presents an approach for detecting and tracking various types of planar objects with geometrical features. We combine traditional keypoint detectors with Locally Likely Arrangement Hashing (LLAH) [21] for geometrical feature based keypoint matching. Because the stability of keypoint extraction affects the accuracy of the keypoint matching, we set the criteria of keypoint selection on keypoint response and the distance between keypoints. In order to produce robustness to scale changes, we build a non-uniform image pyramid according to keypoint distribution at each scale. In the experiments, we evaluate the applicability of traditional keypoint detectors with LLAH for the detection. We also compare our approach with SURF and finally demonstrate that it is possible to detect and track different types of textures including colorful pictures, binary fiducial markers and handwritings.

Index Terms: H.5.1 [INFORMATION INTERFACES AND PRESENTATION (e.g., HCI)]: Multimedia Information Systems—Artificial, augmented, and virtual realities; I.4.8 [IMAGE PROCESSING AND COMPUTER VISION]: Scene Analysis—Tracking

1 Introduction

6DoF object detection and tracking are basically two important technologies for vision based augmented reality. Object retrieval (also known as recognition or identification) is recently incorporated in the detection such that an object is captured in an image, retrieved from a database and its initial pose is simultaneously computed [24]. Once an object is detected, it is tracked from the next frame by tracking keypoints [31] or minimizing the difference between two consecutive images under blurring [23] and non-linear illumination change [8]. This type of tracking is named tracking by tracking. Detecting an object at every frame is another type of tracking named tracking by matching [22].

Generally, object detection is based on local texture descriptors such as SIFT [14]. Because the orientation histogram based keypoint descriptor in SIFT is considered as discriminative enough for large-scale keypoint matching, several attempts to approximate and simplify SIFT have been actively investigated in recent years [17, 31, 5, 28]. Normally, keypoint matching with those methods works well for rich textures that have the variety of intensity in a local patch such as Figure 1(a). However, binary textures such as Figure 1(b), (c) and (d) do not have enough variety of intensity and are hardly detected by those methods. Especially, it is almost impossible to detect random dot markers in Figure 1(e) because local texture at each dot is exactly same. This means that a wide variety of textures cannot be detected with only local texture based descriptors. To address this issue and develop an all-in-one solution, we need to drastically shift the approach for keypoint matching.

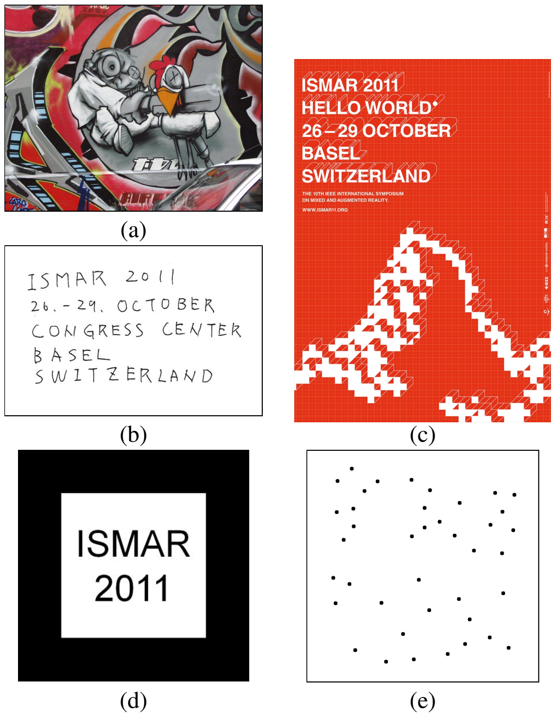


Figure 1: Our objective is to detect and track various types of textures such as (a) graffiti [16], (b) handwriting, (c) poster for ISMAR 2011, (d) ARToolKit marker [13] and (e) random dot marker [29] with only one method.

In this paper, we propose an approach for detecting and tracking various types of planar objects with geometrical features. Compared with traditional methods, we do not use local texture based descriptors. Instead, we employ geometrical relationship between keypoints as a descriptor. Whereas this type of method was not much discussed in the community of computer vision and augmented reality, similar methods were proposed for document image retrieval [21, 18]. Because those methods were dedicated to retrieving only text document images, we generalize them for common object retrieval.

In our approach, keypoints are first extracted with a traditional keypoint detector. Next, we select stable keypoints by considering keypoint response and the distance between keypoints. After selecting keypoints, we compute geometrical descriptors for each keypoint and find correspondences between an input image and a database with LLAH [21]. From the next frame, we perform keypoint matching between two consecutive frames to track objects with LLAH. To improve the robustness for scale changes, we build a non-uniform image pyramid according to the change of keypoint

[*]e-mail: Hideaki.Uchiyama@inria.fr
[†]e-mail: Eric.Marchand@irisa.fr

distribution at each scale.

To the present, geometrical feature based keypoint matching has not been much discussed yet. By investigating and evaluating such approach, we contribute to various kinds of researches based on keypoint matching. Especially, the approach has a possibility to relax the limitation of target objects for augmented reality applications toward "augmenting everything".

The remain of the paper is organized as follows. In the next section, we review traditional keypoint detectors and local descriptors. Section 3 briefly describes LLAH based keypoint matching and tracking for text documents because it is a key technique in our approach. Section 4 explains the detailed parts of detecting and tracking geometrical features. In Section 5, we quantitatively evaluate the robustness of our approach by comparing with SURF [3]. In Section 7, we show augmentation results for various types of textures.

2 Related Works

Detecting keypoints is an initial step in various researches and has been investigated for a long time. Harris corner detector is one of the most popular methods that computes a corner score from the determinant and trace of autocorrelation matrix [12]. FAST corner detector selects a pixel that has higher or lower value than neighbors with a machine learning technique [25]. Mair, et al. improved FAST by building an optimal decision tree [15]. These detectors can be incorporated onto an image pyramid such that keypoints are detected in a scale invariant manner. The keypoint detector in SIFT is based on building a scale space with Differences of Gaussians (DoG) computed by subtracting blurred images [14]. In this method, a point with a local extrema on the space is selected as a keypoint. Another type of scale space is constructed by resizing a filter on a single scale in the following methods. In SURF, Hessian is approximated by box filters and its response is efficiently computed from integral images [3]. In CenSurE, Laplacian filters with the different sizes of octagons and boxes are computed from integral images as well [2]. In CSDD, a central foreground region is compared with its surrounding by computing Laplacian of Gaussian (LoG) with IIR filters [7].

For keypoint description, the recent major approach is the approximation and simplification of SIFT [14]. Mikolajczyk and Schmid [17] proposed spatial sampling of gradient using a log-polar location grid named GLOH compared with a Cartesian grid in SIFT. For mobile devices, Wagner, et al. used smaller size of a sampling grid and fewer quantization level of orientations to simplify SIFT as Phony SIFT [31]. In CHoG, a tree coding technique is used to quantize and compress a histogram of gradient into a low bit rate feature [5]. For an orientation invariant descriptor, Takacs, et al. developed RIFF that performed radial gradient transform for gradient binning [28]. For other types of descriptors, hundreds of binary features in a local patch are computed for a random fern based classification [22] and BRIFF [4].

Due to the development of various keypoint detectors and descriptors, their stability has quantitatively been evaluated from various aspects such as rotation and illumination changes. In 2000, Schmid, et al. reported that Harris was better than other detectors in various cases [26]. For keypoint matching on 3D objects, Hessian-affine and Harris-affine detectors with the descriptor in SIFT outperformed other combinations [19]. Gauglitz, et al. evaluated all combinations of state-of-the-art keypoint detectors and descriptors with several parameters for visual tracking [11]. They stated that it was difficult to select a single best method in all cases because each detector has advantages and disadvantages in a situation.

Some of approaches incorporated geometrical features into local texture based descriptors. Chum and Matas proposed normalization of a local region computed from local affine frames on the outer boundaries of MSER with the scheme of geometric hashing [6]. Forssén and Lowe simply constructed pair descriptors with nearest neighbor keypoints [10].

3 Detecting and Tracking Documents using LLAH

LLAH is a method for retrieving document images using geometrical features [21]. It was extended to a method for tracking documents for augmented reality applications [30]. Also, this technique was applied to the development of random dot markers [29]. Because our approach in this paper is crucially related with these works, we briefly review the procedure to assist the understanding of overall parts of our method.

3.1 Descriptors

We explain the procedure to compute descriptors with Figure 2. For each keypoint, $x'y'$ coordinate system is set and n nearest neighbor points are selected in anti-clockwise direction on the coordinate system ($n = 7$ in the figure). The order of the selection is pre-defined such that we start from **a** (the point closest to x' at the first quadrant). Second, m points are selected from n points to compute one descriptor. This means that the number of descriptors for a keypoint is ${}_nC_m$. For m points, 4 points are selected to compute the ratio of two triangles. By computing all combinations, ${}_mC_4$ dimensional descriptor is constructed. The descriptor is finally quantized into one dimension as index by a hash scheme. Note that these descriptors are not rotation-invariant because the starting point **a** varies when capturing these points from different viewpoints.

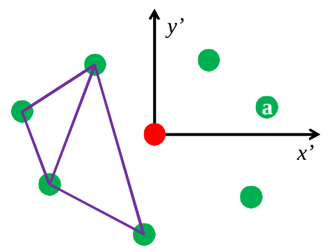


Figure 2: Descriptors in LLAH. A descriptor is the set of the ratios of two triangles. For each keypoint, several descriptors are computed from neighbor points.

3.2 Detection

First, the descriptors (indices) are computed for each keypoint on a reference and stored into an inverted file as (index, keypoint IDs). To achieve rotation-invariant matching, the starting point **a** is set at all n neighbor points for computing the descriptors. Thus, the total number of computed descriptors for a keypoint is $n \cdot {}_nC_m$.

In the keypoint matching, ${}_nC_m$ descriptors are computed and keypoint IDs are retrieved from the inverted file to make a histogram of keypoint IDs versus its retrieval frequency. A keypoint ID that got the maximum frequency is selected as matched. To remove outliers, geometrical verification is performed such that RANSAC based homography is computed for planar objects.

3.3 Tracking

LLAH based keypoint matching is also applied to matching between two consecutive frame for keypoint tracking [30, 29]. This means that the detection and tracking of keypoints are performed with the same computation.

After geometrical verification in the detection, each keypoint on the reference is projected onto the input image to find the closest keypoint in the image. This process can increase correspondences, refine the homography and effectively work because the number of tracked keypoints decreases during moving a camera. To track keypoints, we perform keypoint matching between the keypoints obtained from the projection and those in the next frame.

4 Detecting and Tracking Geometrical Features

In this section, we explain the details of detecting and tracking geometrical features that work well for various types of textures.

4.1 Keypoint Extraction

We first tried applying document image retrieval using LLAH [21] to typical image retrieval. However, it was far from working well because of the instability of keypoint extraction on normal textured patterns. In that method, binarization by adaptive thresholding is used as a word region detector. From a text document image, it can extract word regions and then the center of each region is computed as a keypoint. When we applied it to other textured patterns, it could not stably segment the patterns into comprehensible regions as illustrated in Figure 3. It turned out that the binarization by adaptive thresholding only worked well for extracting word regions in text documents.

As described in Section 2, various keypoint detectors have already been developed in the literature. Therefore, we decided to use them and select stable keypoints from the extracted keypoints because the same neighbors of a keypoint should be extracted from different views for keypoint matching. In the next section, we will explain how to select stable keypoints.

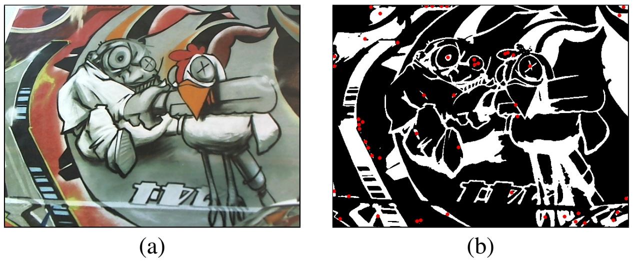


Figure 3: Applying word region detector [21] to Figure 1(a). Binarization by adaptive thresholding was applied to (a) an input image. (b) Few keypoints were extracted from segmented regions because most of the extracted regions were connected.

4.2 Keypoint Selection

Traditional keypoint detectors return the image coordinate of each keypoint with several parameters such as size, scale, angle and response. Some of these parameters might be useful as selection criteria. Size and scale are not meaningful because many keypoints have the same value. Though angle is variant for each keypoint, it is unlikely that a specific angle is better than others. Therefore, we select keypoint response to represent the quality of a keypoint as in [27] and assume that a keypoint with higher response can have higher repeatability. However, it is difficult to set a threshold for keypoint response because the response value depends on textured patterns. Instead, we set a threshold for the maximum number of extracted keypoints selected from the strongest response as first criterion. In traditional keypoint detectors, Harris [12], FAST [25], Fast-Hessian in SURF [3] and CenSurE [2] provide keypoint response. In the evaluations, we compared the accuracy of the detection by all keypoint detectors with LLAH.

The distance between keypoints is also important criterion. When the neighbor points are too close, the ratio of two triangles cannot correctly be computed because the points are almost on a line or the triangles are small as illustrated in Figure 4(a). To compute geometrical features, keypoints should sparsely be distributed. Thus, we set a threshold for the minimum distance between keypoints as second criterion.

The overall procedure for selecting keypoints is as follows. We first sort all keypoints in the descending order of keypoint response. When we select one keypoint from the sorted keypoints, we compute the distance with the nearest neighbor in already-selected keypoints. If the distance is less than a threshold, it is discarded. When the total number of selected keypoints is over than a threshold, the selection is finished. An example of the selection is illustrated in Figure 4. Because these two thresholds affect the result of object detection, we investigate the influence in the evaluations.

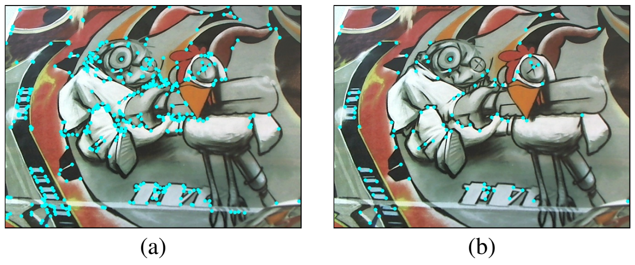


Figure 4: Keypoint selection. FAST was applied to Figure 3(a). (a) 427 keypoints were extracted in total. (b) 100 keypoints were extracted by setting the minimum distance as 10 pixels and the maximum number of keypoints as 100.

4.3 Building Image Pyramid

To produce the robustness for scale changes, we take an image pyramid based approach. Usually, an image pyramid is built by resizing images with a fixed scaling such that the image of each pyramid level is generated by scaling down with a factor of $1/\sqrt{2}$ from the previous level as in [31]. However, such image pyramid was not efficient for our approach because keypoint distribution in two consecutive scales were quite different depending on textures. In this case, it is better to check the middle scale between two scales. Therefore, we develop a method for building an image pyramid with adaptively selected levels.

We first build an image pyramid with small scale changes, and then select appropriate level as illustrated in Figure 5. Full resolution corresponds to level 0 on the pyramid. From the next level, keypoint matching is performed with the previously selected level. Because the transformation is known, we can consider that a correspondence is an inlier when the reprojection error is less than a threshold (set as 3 pixels). Then, we simply compute an inlier rate from the matching result as

$$\text{Inlier rate} = \frac{\text{Number of inliers}}{\text{Number of retrieved keypoints}}. \tag{1}$$

Note that the denominator of this equation is not the number of extracted keypoints because a keypoint does not alway have a corresponding point with LLAH [21]. We remove such keypoints from the extracted keypoints and use only the remaining as retrieved keypoints.

When an inlier rate is lower than a threshold, the level is selected for our image pyramid. For example, we set a threshold for acceptable inlier rate as 20% in Figure 5. We perform keypoint matching between level 1 and level 0, and skip level 1 from the matching result because the inlier rate is higher than the threshold. When level 3 is matched with level 0, level 3 is selected for our pyramid because of its inlier rate. Then, level 4 is matched with level 3 to judge whether level 4 is put into our pyramid or not.

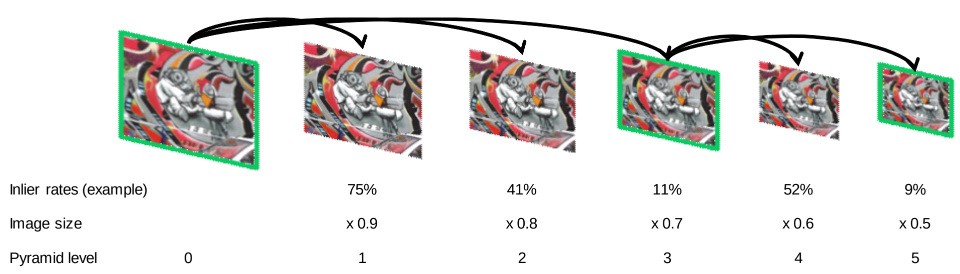


Figure 5: Building a non-uniform image pyramid. The image is first resized with a factor of $1/10$ compared with a full resolution. A pyramid level is selected for our image pyramid in the case that the inlier rate of keypoint matching with the previously selected pyramid level is lower than a threshold. When we set the threshold for the inlier rate as 20%, the selected pyramid levels were 0, 3 and 5 in this figure. The selected levels vary depending on keypoint detectors, two thresholds in the keypoint selection and target textures.

The result of selected levels depends on target textures, keypoint detectors and two thresholds in the keypoint selection. For example, Figure 1(a) with 600×480 pixels had there levels at 600×480, 480×384 and 360×288 whereas Figure 1(e) with 400×400 pixels had one level at 400×400 under the same condition as described in Section 7. This means that the image pyramid using a fixed scaling is not efficient.

4.4 Detection and Tracking

Detecting and tracking planar objects are based on a method described in Section 3. We incorporate the procedure for extracting and selecting stable keypoints on an image pyramid into the method.

At the beginning, we build our image pyramid of reference images to reduce computational cost in the on-line detection and tracking. We label each selected level on the pyramid as an individual reference and compute descriptors in the rotation invariant manner for a keypoint database.

For an input image in the detection, we compute the descriptors only on full resolution and find correspondences on the references. From the result of keypoint matching, we have the candidates of detected references with their pyramid level. This means that a reference image can be detected on different levels. By computing RANSAC [9] based homography for each candidate, we finally select a reference at a level.

Once a reference is detected, it is tracked using the reference with a detected level. The keypoints on the reference are projected onto the input image to increase correspondences. In the next frame, keypoints extracted in the image were matched with the keypoints obtained from the projection. When tracking fails, we detect a reference in the database again.

5 Evaluations with Our Dataset

We first focused on the evaluations of object detection to clarify the applicability of traditional keypoint detectors with LLAH. As described in Section 3, the geometrical descriptors in LLAH should theoretically be rotation invariant. To evaluate the invariance, we prepared our dataset images captured under large rotations.

5.1 Overview

In order to capture the whole part of each texture with a rotating camera, we used a camera mounted on a robot arm to precisely control the rotation angle. We initially set the arm perpendicular to textured patterns in Figure 1 on a table and rotated it at 5 degrees increments through 90 degrees with centering around a principal point of the camera as illustrated in Figure 6. The resolution of captured grayscale images was 640×480.

We performed keypoint matching between the first frame and others, and counted inliers by computing a reprojection error with the ground truth of a rotation matrix obtained from the robot odometry. A keypoint is an inlier when the error is less than 3 pixels. The evaluation criterion was the inlier rate defined in Equation 1. Note that we did not build our image pyramid because the distance between a camera and a texture was fixed. We discussed the influence in a single scale in this section. Regarding the parameters in LLAH [21], we set $n = 7$ and $m = 5$ for all experiments.

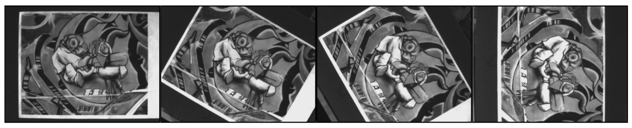


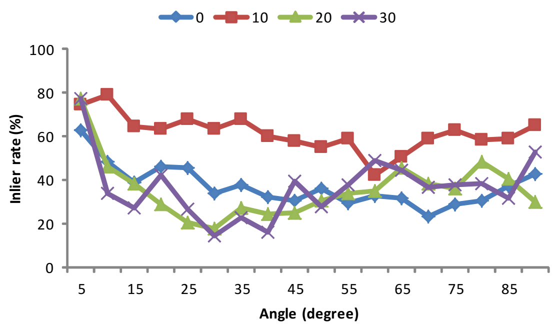
Figure 6: Example of images. A camera equipped on an arm robot was used to control the rotaiton angles precisely. Grayscale images were captured at 5 degrees increments through 90 degrees. The resolution of the captured images was 640×480.

5.2 Comparison between Keypoint Detectors

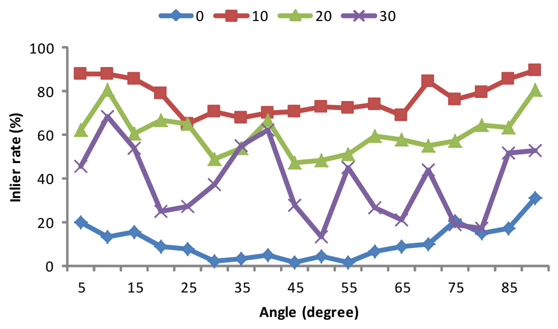
We first compared the inlier rates computed with each traditional keypoint detector under large rotations. For all texture images in Figure 1, we tested our approach with Harris [12], FAST [25], Fast-Hessian in SURF [3] and STAR that was improved version of CenSurE [2] implemented in OpenCV [1]. We used default parameters in OpenCV for all detectors. For two thresholds in the keypoint selection, we set the maximum number of keypoints as 200 and the minimum distance between keypoints as 10 pixels. As a comparison with a traditional local texture descriptor, we selected SURF [3] for keypoint extraction and description and FLANN [20] for keypoint matching implemented in OpenCV. We limited the maximum number of extracted keypoints to 200 by selecting from the strongest response to set up the same condition as in our approach. The inlier rate for SURF was computed by dividing the number of inliers by the number of extracted keypoints because FLANN always returned the nearest point for each keypoint.

In Figure 7, we showed the results of the inlier rates at each angle for all texture images. Harris outperformed others in most of the cases including rich texture and binary texture. This means that keypoint extraction with rotation invariance in Harris provided better performance than others in terms of stability. Especially, the keypoints on the handwriting were stably extracted as illustrated in Figure 7(b). Because the inlier rates are around 50%, RANSAC [9] or other robust estimators can remove outliers when computing homography.

For the ISMAR 2011 pattern in Figure 1(c), the inlier rate of Harris was lower than the rate of Harris in other textures as illustrated

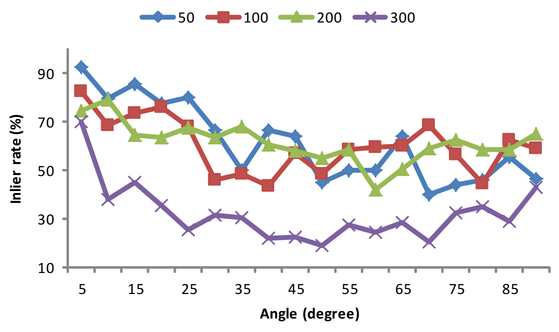


(a)

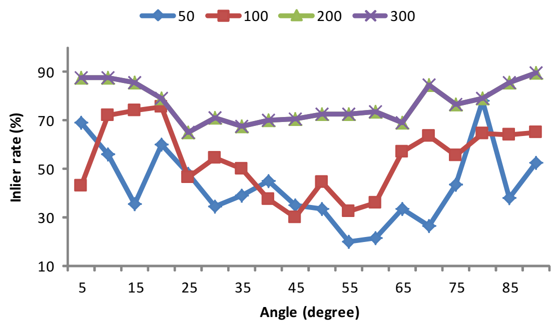


(b)

Figure 8: Thresholds for the minimum distance between keypoints (pixel). (a) and (b) correspond to (a) and (b) in Figure 1. We tested the threshold from 0 pixels to 30 pixels. In both cases, 10 pixels got fairly good results.



(a)



(b)

Figure 9: Thresholds for the maximum number of extracted keypoints. (a) and (b) correspond to (a) and (b) in Figure 1. We set the threshold from 50 to 300. In (b), the lines of 200 and 300 were overlapped. It is difficult to state that more or fewer number is better.

in Figure 7(c). This is because there were a lot of similar corners extracted from squares in the lower part of this texture. When we limited the number of extracted keypoints, all corners could not always be extracted and some of them were unstably extracted.

From Figure 7(e), it was obviously proved that a local texture based approach could not detect random dot markers because the local texture was exactly same. Therefore, an geometrical feature based approach is definitely required to detect various kinds of textures with only one method.

5.3 Thresholds in Keypoint Selection

In this section, we investigated the influence of two thresholds for inlier rates with Harris corner detector for Figure 1(a) and (b).

5.3.1 Minimum Distance between Keypoints

In this experiment, we fixed the threshold for the maximum number of extracted keypoints as 200. As illustrated in Figure 8, we tested the minimum distance between keypoints from 0 pixels to 30 pixels. From Figure 8(a) and (b), the result at 10 pixels commonly outperformed others. For other cases, the results were not stable. The influence of the threshold clearly appeared in Figure 8(b). When the threshold was 0 pixels, the inlier rates were around 0 at many angles because many keypoints densely appeared together. As a result, geometrical features for such keypoints were not discriminative. To avoid detecting dense keypoints, it is necessary to set up a threshold for the minimum distance between keypoints.

5.3.2 Maximum Number of Extracted Keypoints

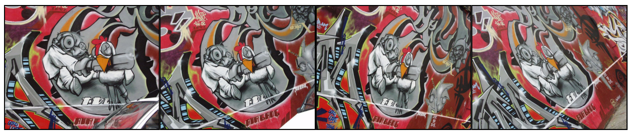
In this experiment, we fixed the threshold for the minimum distance between keypoints as 10 pixels. As illustrated in Figure 9, we tested the maximum number of extracted keypoints from 50 to 300. In Figure 9(b), the line charts of 200 and 300 were overlapped because extracted keypoints in both cases were same. In Figure 9(a), the worst result appeared at 300. In contrast, the best result appeared at 300 in Figure 9(b). Similarly, fairly good result appeared at 50 in Figure 9(a) whereas the same threshold was worst in Figure 9(b). From this result, it is hard to state that more or fewer number of keypoints is better or not. From various experiments, we confirmed that setting 200 on the threshold had provided better performance.

6 Evaluations with Mikolajczyk's Dataset

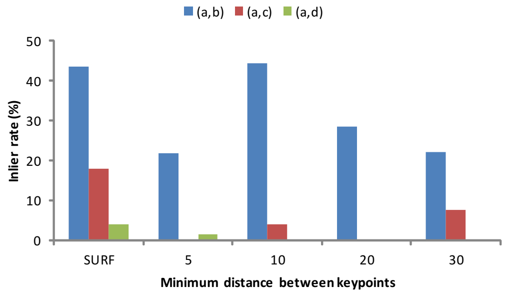
In this section, we evaluated the influence of viewpoint changes for the detection with two datasets in Mikolajczyk framework [16, 17] as illustrated in Figure 10 and 11.

6.1 Overview

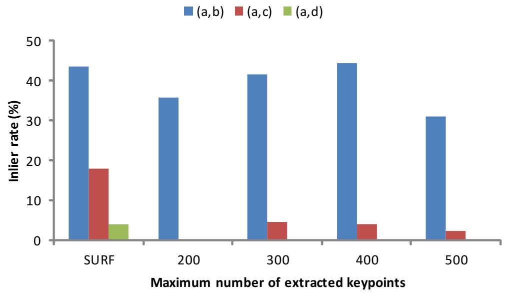
As in Section 5.3, we investigated the influence of two thresholds. In Figure 10(e) and Figure 11(e), the inlier rate with several thresholds for the maximum number of extracted keypoints was computed by fixing the minimum distance between keypoints as 10 pixels. In Figure 10(f) and Figure 11(f), the inlier rate with several thresholds for the minimum distance between keypoints was computed by fixing the maximum number of extracted keypoints as 400. In each image set in Figure 10 and 11, (a) was used as a reference with an image pyramid and compared with other three images from (b) to (d). We compared our approach using Harris corner detector with SURF.



(a) (b) (c) (d)

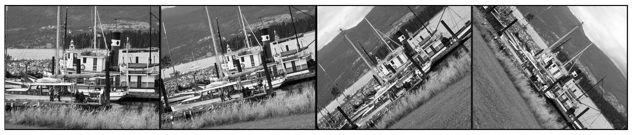


(e)

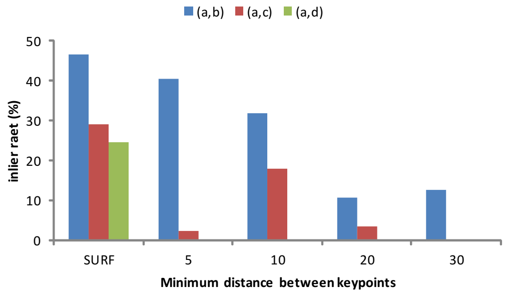


(f)

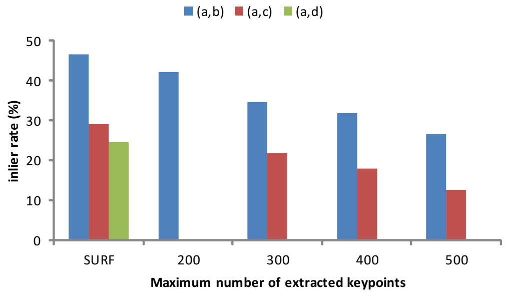
Figure 10: View angle change. (e) The thresholds for the minimum distance between keypoints from 5 to 30 pixels were tested with three image pairs: (a,b), (a,c) and (a,d). (f) The thresholds for the maximum number of extracted keypoints from 200 to 500 were tested with the same image pairs as (e).



(a) (b) (c) (d)



(e)



(f)

Figure 11: Zoom and rotation change. The experimental condition of (e) and (f) was the same as that of Figure 10 (e) and (f).

6.2 Evaluation

Figure 10 included view angle changes in perspective views. In Figure 10(e) that illustrated the inlier rates with different thresholds for the minimum distance between keypoints, the result at 10 pixels was quite better than others for the pair (a,b). For other pairs, our approach did not achieve enough inlier rate for camera pose estimation. In Figure 10(f) that illustrated the inlier rates with different thresholds for the maximum number of extracted keypoints, there were few differences among the results of four thresholds. Compared with SURF, our approach with Harris corner detector was not invariant to viewpoint changes such that the inlier rates computed from our approach were not sufficient in the pair (a,c) while SURF still achieved around 20%.

Figure 11 included zoom and rotation changes in a 3D outdoor scene. Compared with the previous experiment, the results for the pair (a,c) were better because the viewpoint change in this experiment was limited to zoom and rotation. For other pairs, the result was almost same as that in the previous experiment.

6.3 Discussion

In order to establish correct correspondences in our approach, we need to extract both same keypoints and same neighbor keypoints. As illustrated in Figure 11(e), the inlier rate decreased when the maximum number of extracted keypoints increased. Because different keypoints might be included in the neighbor keypoints in different views if there are too many keypoints, the number of extracted keypoints should be limited.

Also, the matching result was sensitively influenced by the minimum distance between keypoints. From both experiments in Figure 10 and 11, it is hard to find the best threshold because the best threshold depended on texture patterns. Because the density of extracted keypoints varies depending on an image size and content, it is necessary to set up the threshold considering such elements.

Let us finally note that we only evaluated the detection part of our method in this section. Such detection process has only to be considered for the initialization and when the tracking process failed. It has also to be noted that, the complexity of the detection does not increase dramatically with the number of reference image in the database. Therefore, in a real application process, various reference images are considered. They can also be synthesized (as in [22] to take into account affine or perspective transformation). Such a process greatly improves the detection process.

7 Augmentation Results

Figure 12 represented the results of the detection on different scales and the tracking under perspective views. We first stored the references from all texture images, and then captured one of the textures for the augmentation. For two thresholds in the keypoint selection, we set the minimum distance between keypoints as 10 pixels and the maximum number of extracted keypoints as 200. The first row in Figure 12 represented the result of keypoint extraction for the reference images. The resolution of the texture images and that of their pyramid are described in Table 1.

Table 1: Image resolution of the textures in Figure 1 and their pyramid (pixel). The selected levels for the pyramid varied according to textures.

Texture	Resolution	Pyramid
(a)	640×480	640×480, 480×384, 360×288
(b)	640×480	640×480, 448×336
(c)	353×500	353×500, 282×400, 247×350
(d)	400×400	400×400, 240×240
(e)	400×400	400×400

Table 2: Average computational costs (msec).

Extraction	Description	Detection	Tracking
15	10	5	2

As illustrated in Figure 12, the movement of a camera included scale changes and perspective views. Object detection was done with difference scales. The robustness of random dot markers to scale changes was apparently higher than others even when one level on the image pyramid was generated because the dots were stably extracted from several viewpoints. For others, the range of camera movement during the tracking was almost same. Regarding computational costs, a frame rate was around 30 frames per second on a typical laptop (Intel i7 CPU 2.67GHz with 4GB RAM) for 640×480 pixels as described in Table 2. By using our approach, it is possible to detect and track rich texture, handwriting, ARToolKit markers and random dot markers.

8 Conclusion

We proposed an approach for detecting and tracking various types of textures using geometrical feature based keypoint matching. We combined traditional keypoint detectors with LLAH originally designed for retrieving text documents. Because the stability of keypoint extraction was required, we selected stable keypoints according to keypoint response and the distance between keypoints. We also built a non-uniform image pyramid by matching between neighboring scales to adaptively select the representative scales.

In the evaluations, we investigated the inlier rates in the detection for several image sets. Because the result was influenced by both the minimum distance between keypoints and the maximum number of extracted keypoints, we need to set up two thresholds considering an image size and content. To be invariant to viewpoint changes, one possible approach is to prepare synthetic images of various different viewpoints such as [22] and compute descriptors on them because the number of registered images did not much influence the retrieval costs due to a hashing scheme as reported in [21].

In the further works, we need to improve object tracking because traditional trackers are more robust to illumination and scale change [8]. One solution is to merge such trackers into our approach such that object detection is performed by our method and the tracker is selected depending on the detected object. The integration of a local texture descriptor with geometrical descriptors in LLAH will be also an interesting topic.

References

[1] http://opencv.willowgarage.com.

[2] M. Agrawal, K. Konolige, and M. R. Blas. CenSurE: Center surround extremas for realtime feature detection and matching. In *ECCV*, pages 102–115, 2008.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *CVIU*, 110:346–359, 2008.

[4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. In *ECCV*, pages 778–792, 2010.

[5] V. Chandrasekhar, G. Takacs, D. M. Chen, S. S. Tsai, R. Grzeszczuk, and B. Girod. CHoG: Compressed histogram of gradients a low bitrate feature descriptor. In *CVPR*, pages 2504–2511, 2009.

[6] O. Chum and J. Matas. Geometric hashing with local affine frames. In *CVPR*, pages 879–884, 2006.

[7] R. T. Collins and W. Ge. CSDD features: Center-surround distribution distance for feature extraction and matching. In *ECCV*, pages 140–153, 2008.

[8] A. Dame and E. Marchand. Accurate real-time tracking using mutual information. In *ISMAR*, pages 47–56, 2010.

[9] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *C. of ACM*, 24:381–395, 1981.

[10] P.-E. Forssén and D. Lowe. Shape descriptors for maximally stable extremal regions. In *CVPR*, pages 778–792, 2007.

[11] S. Gauglitz, T. Höllerer, and M. Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *IJCV*, 2011.

[12] C. Harris and M. Stephens. A combined corner and edge detector. In *AVC*, pages 147–151, 1988.

[13] H. Kato and M. Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *IWAR*, pages 85–94, 1999.

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004.

[15] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *ECCV*, pages 183–196, 2010.

[16] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 60:63–86, 2004.

[17] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *TPAMI*, 27:1615–1630, 2005.

[18] J. Moraleda and J. J. Hull. Toward massive scalability in image matching. In *ICPR*, pages 3424–3427, 2010.

[19] P. Moreels and P. Perona. Evaluation of features detectors and descriptors based on 3D objects. *IJCV*, 73:263–284, 2007.

[20] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISSAPP*, pages 331–340, 2009.

[21] T. Nakai, K. Kise, and M. Iwamura. Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval. In *DAS*, pages 541–552, 2006.

[22] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *TPAMI*, 32:448–461, 2010.

[23] Y. Park, V. Lepetit, and W. Woo. ESM-Blur: Handling & rendering blur in 3D tracking and augmentation. In *ISMAR*, pages 163–166, 2009.

[24] J. Pilet and H. Saito. Virtually augmenting hundreds of real pictures: An approach based on learning, retrieval, and tracking. In *IEEEVR*, pages 71–78, 2010.

[25] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *TPAMI*, 32:105–119, 2010.

[26] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *IJCV*, 37:151–172, 2000.

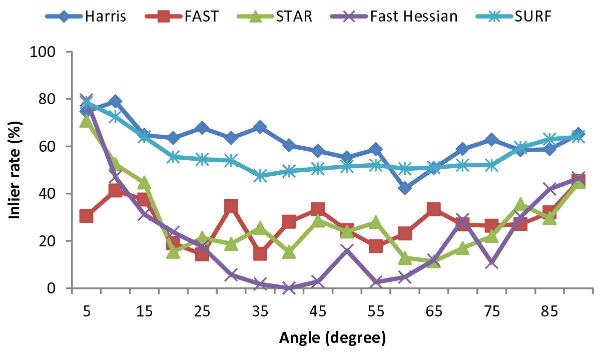
[27] J. Shi and C. Tomasi. Good features to track. In *CVPR*, pages 593–600, 1994.

[28] G. Takacs, V. Chandrasekhar, S. S. Tsai, D. M. Chen, R. Grzeszczuk, and B. Girod. Unified real-time tracking and recognition with rotation-invariant fast features. In *CVPR*, pages 934–941, 2010.

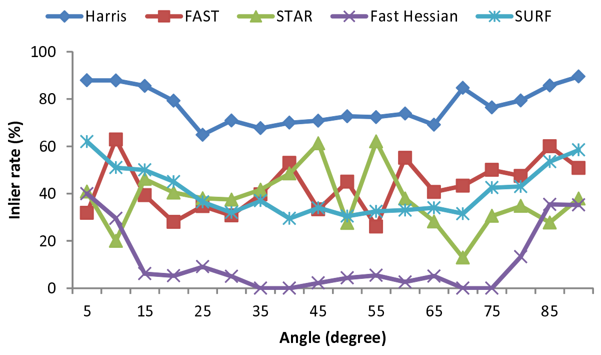
[29] H. Uchiyama and H. Saito. Random dot markers. In *IEEEVR*, pages 35–38, 2011.

[30] H. Uchiyama, H. Saito, M. Serviерés, and G. Moreau. Camera tracking by online learning of keypoint arrangements using LLAH in augmented reality applications. *VR*, 2010.

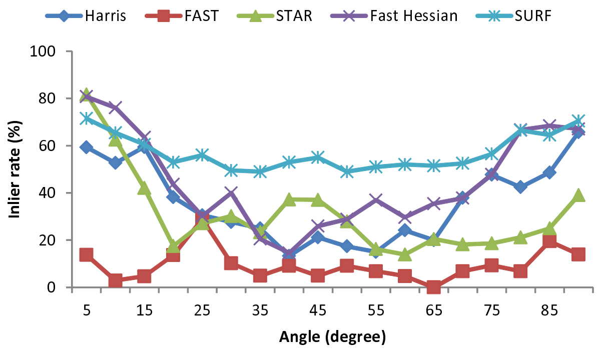
[31] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *ISMAR*, pages 125–134, 2008.



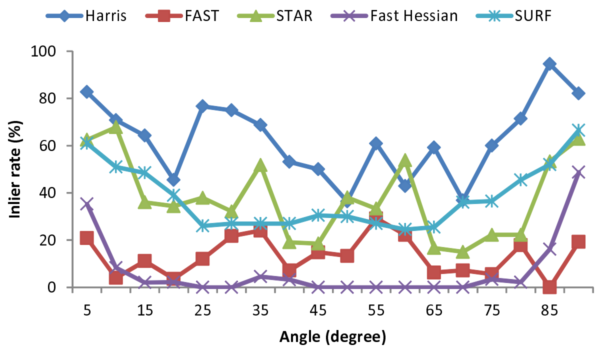
(a)



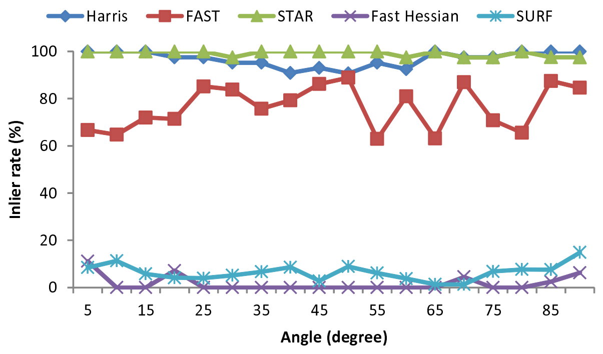
(b)



(c)



(d)



(e)

Figure 7: Comparison between keypoint detectors. (a)-(e) correspond to (a)-(e) in Figure 1. In most of the cases, Harris outperformed others. This means that the stability of keypoint extraction with rotation invariance in Harris was more superior than others. For random dot markers, it was obviously proved that keypoint matching with local texture descriptors failed. To deal with various types of textures, a geometrical feature based approach should be integrated.

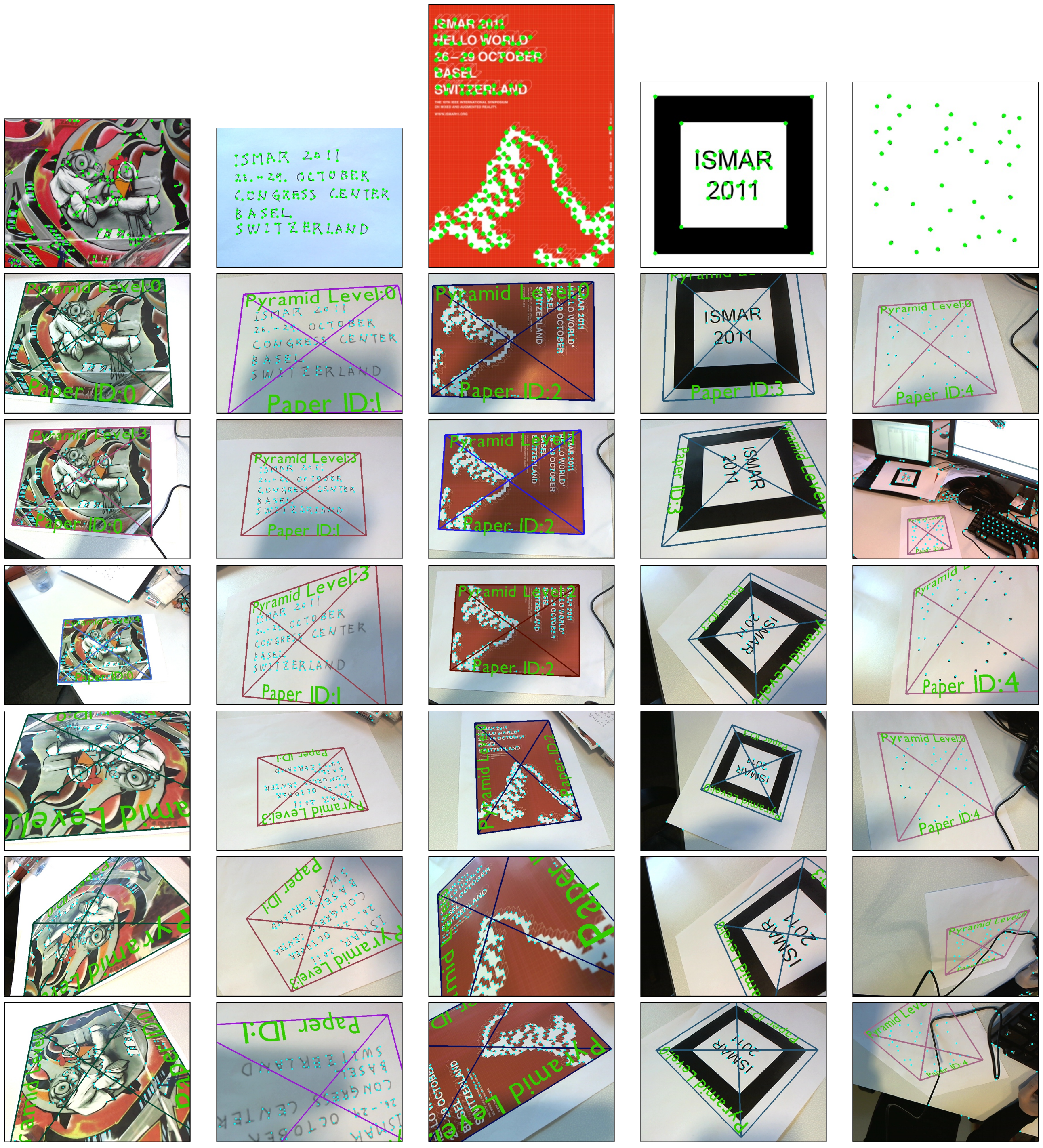


Figure 12: Augmentation results from detecting and tracking planar images. (a)-(e) correspond to (a)-(e) in Figure 1. Images at the first row are the results of keypoint extraction with Harris on the reference images. As a result of detection and tracking, we overlaid a paper ID and a pyramid level on the images. The range of camera movement for random dot markers was widest.