Real-time Object Recognition with Pose Initialization for Large-scale Standalone Mobile Augmented Reality

Suwon Lee[1*]
[1] Department of Computer Science and The Research Institute of Natural Science, Gyeongsang National University
501 Jinjudae-ro, Jinju-si, Gyeongsangnam-do, Republic of Korea
[e-mail: leesuwon@gnu.ac.kr]
*Corresponding author: Suwon Lee

Received May 20, 2020; revised August 10, 2020; accepted September 24, 2020; published October 31, 2020

Abstract

Mobile devices such as smartphones are very attractive targets for augmented reality (AR) services, but their limited resources make it difficult to increase the number of objects to be recognized. When the recognition process is scaled to a large number of objects, it typically requires significant computation time and memory. Therefore, most large-scale mobile AR systems rely on a server to outsource recognition process to a high-performance PC, but this limits the scenarios available in the AR services. As a part of realizing large-scale standalone mobile AR, this paper presents a solution to the problem of accuracy, memory, and speed for large-scale object recognition. To this end, we design our own basic feature and realize spatial locality, selective feature extraction, rough pose estimation, and selective feature matching. Experiments are performed to verify the appropriateness of the proposed method for realizing large-scale standalone mobile AR in terms of efficiency and accuracy.

Keywords: Real-time object recognition, mobile augmented reality, large-scale object recognition, standalone augmented reality, real-time feature matching

http://doi.org/10.3837/tiis.2020.10.010 **ISSN : 1976-7277**

1. Introduction

The popularization of mobile devices such as smartphones has increased the expectation for realizing augmented reality (AR) while simultaneously addressing resource constraints. Portability and accessibility, which are the most important advantages of mobile devices, enable various scenarios to be provided in the service dimension of AR applications for mobile devices. However, mobile devices have limited memory capacity and high computational complexity, which limits the number of objects that can be recognized.

Generally, as the number of objects to be recognized increases, the required memory capacity and computation amount increase proportionally. One solution to this problem is to leverage the power of a server. Most of the operations and memory utilization can be performed on the server, and then, the result can be transmitted to the client mobile device. However, this method necessarily requires network communication, and the service quality depends on the state of the network connection of the user's mobile device, the server status of the service provider, and the number of simultaneous users at a given time. In addition, this approach interferes with immersion, which is one of the most important factors in AR. To establish a sense of immersion, when a user's point of view is focused on an object, the virtual content related to the object should immediately be augmented. To realize this, it is necessary to process every frame coming through the camera built in the user's mobile device, but it is practically impossible to transmit every frame to the server. In most cases, only the frame desired by the user is transmitted to the server, which requires the user to interact with the screen, for example, to interfere with the immersion. Therefore, it is very important to develop a large-scale mobile AR system in which the entire processing is performed on a mobile device without the assistance of a server.

In this paper, we call such a system a large-scale standalone mobile AR system and propose efficient and robust recognition methods to realize it. The proposed methods include basic feature design, spatial locality, selective feature extraction, approximate pose estimation, and selective feature matching. Compared to the existing standalone mobile AR system designed for a single object, the number of recognition objects has been increased to 2,000 at the expense of only an 11% reduction in the recognition rate. In addition, it has been found that real-time recognition of 2,000 objects uses only 36 MB of memory on the mobile device. We have also shown that the proposed method is robust to a variety of environmental changes, including cluttered backgrounds.

We organized the remainder of this paper as follows: Section 2 describes the AR systems related to this study and Section 3 proposes the basic features to be used in the learning and recognition phase, and Section 4 and 5 propose the learning and recognition phase respectively. Section 6 presents the experimental results for the proposed method, finally concludes in Chapter 7, and seeks future research directions.

2. Related Work

Most AR systems recognize a specific object based on a method of matching keypoints detected in different images. This is because the keypoint-based method is robust to changes such as scale, view point, and illumination or partial occlusion [1].

One of the most important properties of a keypoint is repeatability. Repeatability refers to the property that is repeatedly detected at the same location even if an object is photographed under different viewpoints or different lighting. A representative keypoint detection method is using difference of Gaussian proposed by Lowe [2]. Lowe detected a keypoint with high repeatability for scale change by selecting local extrema as a keypoint in a difference of Gaussian pyramid approximating the Laplace of Gaussian pyramid. However, the method requires a large amount of computation, and as an alternative, a Feature from Accelerated Segment Test (FAST) [3] has been proposed. FAST detects keypoints by simply comparing the brightness values of surrounding pixels, and is being used as an alternative in applications where computation speed is important due to its high computation speed.

There have been two main approaches to match the detected keypoints. The first approach is to create a high-dimensional vector in an affine-invariant manner for the local image patch surrounding the keypoint. Typical examples include Scale Invariant Feature Transform (SIFT) [2] and Speeded Up Robust Features [4], and these are called keypoint descriptors. Among them, SIFT is regarded as one of the best performing keypoint descriptors [5]. However, calculating these descriptors includes gradient calculation, scale selection, rotation correction, and intensity normalization processes that require large amounts of computation. Also, because the keypoint matching based on descriptors basically depends on the nearest neighbor search, even if the search space is reduced using an efficient data structure such as a tree structure or hashing, a large amount of computation is required. As an alternative, binary descriptors which compare the brightness values of surrounding pixels and describe the result in a bit string to generate features have been proposed. Typical examples include Binary Robust Independent Elementary Feature (BRIEF) [6], Oriented FAST and Rotated BRIEF (ORB) [7], Binary Robust Invariant Scalable Keypoints [8], and Fast Retina Keypoint [9]. They have many advantages in terms of speed, because the processes that required a large amount of computation are omitted or reduced. In addition, due to the nature of the binary feature, where one meaningful value is stored in a single bit within a feature, high-speed matching is possible using a Hamming distance, which enables fast calculation when keypoints are matched.

The second approach for keypoint matching recasts the problem as a statistical classification problem. In short, the set of all possible appearances of each keypoint is considered a class, and is used to train a classifier. In the offline learning, for all extracted keypoints, various variants of the local image patch surrounding the keypoint are set as a class to learn the classifier. By shifting the computational burden from runtime to an offline training phase, this approach can achieve real-time keypoint matching. A representative method is Random Forest (RF) [10] proposed by Lepetit. RF is composed of several weak classifiers in a tree structure, and a tree learns the posterior probability distribution for keypoint matching. The internal node of the tree serves to divide the image space of the local image patch to reach the leaf node. In the leaf node, the posterior probability is calculated by using the class label of the local image patch. RF enabled fast and robust matching online at the expense of offline learning time and memory usage. After this, random ferns [11] has been proposed to simplify the structure and learn the class conditional probability, and studies have been conducted to reduce the unrealistic memory usage of the RF and random ferns [12,13].

The first standalone mobile AR system was developed by [14]. They proposed PhonySIFT, PhonyFerns, and PatchTracker, which have excellent performance in terms of computation, and showed that it is possible to recognize and track a single object on mobile devices.

Efforts to increase the number of recognition targets were first made on a personal computer (PC). In [15], Cho et al. proposed the generic random forest (GRF) to implement an AR book

with up to 200 pages on a PC. The GRF maximizes the reusability of the existing RF so that both object recognition and keypoint matching can be performed simultaneously. However, the GRF requires a huge memory just like the RF. The amount of memory required to learn one page is approximately 10 MB; hence, they limited their research to a PC.

To realize large-scale mobile AR, server–client-based systems have been proposed. The server performs object recognition on the image received from the client and transmits the recognition result back to the client. The recognition result includes the initial posture of the object, and the client tracks the object using the recognition result. In [16], Jung et al. implemented a server–client-based large-scale mobile AR system for 10,000 objects using vocabulary trees [17] as a recognizer. Because it is impossible to transmit all the real-time images from the client to the server, only the frame desired by the user is transmitted to the server. This requires additional interaction from the user and leads to the inhibition of immersion.

3. Basic Feature Design

We consider an approach based on keypoint matching to realize initial pose estimation simultaneously with object recognition. The approach facilitates the pose estimation of an object through the geometric relationship of the matched keypoints. In particular, considering the limited memory of mobile devices, we focus on a local feature-based method that includes keypoint detection [18] and keypoint description [5].

In this paper, a keypoint p is defined as a combination of four attributes, such as position attribute **x**, intensity attribute h, orientation attribute o, scale attribute s as follows:

$$\mathrm{p} = (\mathbf{x}, \mathrm{h}, \mathrm{o}, s),\ \mathbf{x} = (x, y) \in R^2,\ \mathrm{h} \in R,\ \mathrm{o} \in [0, 360],\ s \in \{0, \cdots, N_s - 1\}, \tag{1}$$

In general, keypoints are detected on the scale space so that the scale attribute can be assigned to the keypoints. The scale attribute is robust to scale changes when keypoints are described. Because detecting keypoints in the scale space results in high computational complexity, we approximate the scale space by building an image pyramid of N_s levels, in which each level is scaled down by a factor of S compared to the previous one. The choice of N_s and S is a trade-off between robustness to scale changes of the object and the computation time. In our implementation, we use the experimentally chosen values of N_s = 8 and S = 1.2.

We then detect keypoints using the FAST corner detector [3], which is known to be one of the fastest detectors at each level of the image pyramid. Because keypoints are detected on the image pyramid, the keypoints have a scale attribute s as well as a position attribute **x**. The Harris response [19] is then calculated for the keypoints, and an intensity attribute h is assigned. When more than a predetermined number of keypoints are detected, the number of keypoints is limited by preferentially selecting keypoints with high h. Thereafter, to assign rotation-invariant properties to the keypoints, an orientation attribute o is assigned to each keypoint using the intensity centroid, as described in [20]. In the recognition step, the orientation attribute is assigned immediately before the keypoint is described as the feature vector. This is to maximize the speed gain by reducing unnecessary operations through selective feature extraction. Finally, a keypoint p is defined as a combination of four attributes as follows:

To extract feature vectors from the detected keypoints, the SIFT descriptor [2], which is known to be robust to illumination and 3D viewpoint changes, is used. Owing to speed and memory considerations, we extract a 36-dimensional vector, instead of a 128-dimensional vector, by adopting 3 × 3 subregions with four gradient bins each. In addition, to maximize

memory efficiency, we represent each element via 8-bit values by quantizing the 4-byte floating-point value of each element to the [0, 255] range. From this, we have the additional advantage of being able to accelerate the distance computation between two features by calculating all combinations of $(a-b)^2$ in advance. For example, using this descriptor, the squared L2 distance of two feature vectors can be calculated through only 36 addition operations.

4. Offline Training Step

For the given model images $\mathrm{I} = \{I_0, \cdots, I_{N_I-1}\}$, feature vectors $F_i = \{\mathrm{f}_i^0, \cdots, \mathrm{f}_i^{N_\mathrm{f}-1}\}$ are extracted from each model image I_i. In this process, keypoints $P_i = \{\mathrm{p}_i^0, \cdots, \mathrm{p}_i^{N_\mathrm{p}-1}\}$ are also detected ($N_\mathrm{p} = N_\mathrm{f}$). The choice of N_f is a trade-off between accuracy and efficiency. The larger the value of N_f, the higher is the accuracy, but the computation time and the required memory capacity also increase. In our implementation, we use the experimentally chosen value of N_f = 500.

When feature vectors are extracted, it is recorded in a file so that it can be read and used in the recognition step. A total of 36 bytes are required to store one feature f. Therefore, 18 kB are required for one image and 18 MB for 1000 images, which is reasonable for mobile devices.

The local feature-based method used in this research has another advantage in that training is very fast. It takes approximately 30s to train 1000 images on a mobile device. This speed is considerably faster than that achieved by classifier-based methods such as RF and random ferns [11], which take 1 min to train one image. In addition, the local feature-based method can easily train additional images because the training process is performed only for the additional images independently of the previously trained images. On the other hand, the GRF must retrain the classifier with the previously learned images to train additional images. This is greatly inconvenient and burdensome for the AR service provider.

The total number of extracted features N_f increases in proportion to the number of objects N_O (e.g., N_F = 1M when N_F = 500 and N_O = 2000). N_F represents the size of the space to be searched when the query feature performs matching in the recognition step, and increasing N_F makes real-time matching impossible (e.g., over 10s when N_F = 1M). For such a large search space, the approximate nearest neighbor (ANN) search is employed to realize real-time matching. The objective of using the ANN search is to trade off some amount of precision during the search for the sake of a substantial reduction in the query time (e.g., 100 times faster with 95% precision).

For the ANN search, we use randomized kd-trees [21] and the best-bin-first [22] algorithm. In a randomized tree, the splitting dimension of a node is randomly chosen from among a set of the dimensions with highest variance and the split value is also randomly chosen using a point close to the median. Because of the randomness, trees are built to divide different feature spaces. Combining these trees creates partitions with overlapping feature spaces and mitigates quantization effects. A new data point falls to a leaf node of each tree, and the distances to the discriminating boundaries are recorded in a single priority queue for all trees. In the best-bin-first manner, the most promising branch from all trees is chosen and keep adding unseen nodes into the priority queue. An approximate answer can be returned without significantly increasing the search time by stopping further search after traversing a certain number of the nearest bins.

The randomized kd-trees consist of N_T trees that are constructed differently to complement each other. While traversing trees, the best-bin-first algorithm searches the space in the order

of the bins closest to the query location and stops the search after checking the first N_V nearest neighbor candidates. In our implementation, we use the experimentally chosen values of N_T = 4 and N_V = 32.

(a) (b) (c)

Fig. 1. (a) In real word images, a number of keypoints are detected in parts other than the object of interest, such as the background; (b) Subregion strategy based on spatial locality; (c) Uniformly selected keypoints.

5. Online Recognition Step

Recognition includes the process of initial pose calculation together with the identification of the object in the image captured from the camera. If the object is recognized, it is possible to track the object from the subsequent image using the calculated initial pose. The tracking performance is irrelevant to the scale of the number of objects because the tracking is performed on a specific object. On the other hand, the recognition performance is very sensitive to the scale of the number of objects. The following subsections describe schemes for solving the accuracy and speed problems that occur in the recognition step when the scale of the number of objects becomes large.

5.1 Spatial Locality

In the recognition step, because no information is available on the location of the object of interest in the image, keypoints must be detected in the entire area of the image. Accordingly, a number of keypoints are detected in parts other than the object of interest, such as the background. **Fig. 1 (a)** shows the location of the object of interest in the image and the detected keypoints. The indiscriminate use of such detected keypoints leads to a decrease in recognition accuracy. To resolve the problem, we introduce spatial locality, which is the assumption that an object projected on to an image is spatially connected. Based on the spatial locality, we first divide the query image I_q into a m × n gird of N_r subregions $R = \{\mathrm{r}_0, \cdots, \mathrm{r}_{N_\mathrm{r}-1}\}$, as shown in **Fig. 1 (b)**. We then detect keypoints $P_q = \{\mathrm{p}_q^0, \cdots, \mathrm{p}_q^{N_\mathrm{p}-1}\}$ from I_q and assign the membership information of the subregions as an additional attribute to the keypoints. The spatial locality handles the cluttered background problem, while accelerating recognition through selective feature extraction in the following process.

5.2 Candidate Estimation

To accelerate recognition, we first estimate the candidate at a high speed using only a small number of features. For this, keypoints $P_q' = \{\mathrm{p}'^0_q, \cdots, \mathrm{p}'^{N_{\mathrm{p}'}-1}_q\}$ are selected in the order of the highest h among P_q ($P_q' \subseteq P_q$and $N_{\mathrm{p}'} \leq N_\mathrm{p}$). When selecting keypoints, $N_{\mathrm{p}'}/N_\mathrm{r}$ keypoints are allocated to be selected for each subregion so that the keypoints P_q' are uniformly distributed. **Fig. 1 (c)** shows that the same number of keypoints are selected per subregion when N_p = 200

and $N_{\mathrm{p}'} = 40$. We then extract feature vectors $F_q' = \{\mathrm{f}'^{0}_{q}, \cdots, \mathrm{f}'^{N_{\mathrm{f}'}-1}_{q}\}$ to be used for candidate estimation from P_q'. Because the unselected keypoints are not to be extracted as feature vectors, the orientation attribute is not assigned. The closer $N_{\mathrm{p}'}$ is to N_{p}, the higher is the accuracy but so is the computation time. In our implementation, we use the experimentally chosen value of $N_{\mathrm{p}'} = 100$, which compromises between accuracy and computation time.

The extracted feature vectors F_q' perform feature matching to the N_{F} feature vectors of the feature space built from the model images I. The matching is speeded up by the ANN searcher trained in the offline training step. For a feature vector f_q', the ANN searcher returns the index i of the matched feature vector with the distance d as follows:

$$\mathrm{i} = (i, j) = \mathrm{index}\left(\mathrm{ANN}\left(\mathrm{f}_q'\right)\right), i = 0, \cdots, N_O - 1, j = 0, \cdots, N_{\mathrm{f}} - 1 \tag{2}$$

$$d = \mathrm{distance}\left(\mathrm{f}_q', \mathrm{ANN}\left(\mathrm{f}_q'\right)\right). \tag{3}$$

Based on the matching result, the score of the support as a candidate for each object is calculated as follows:

$$\mathrm{Score}_{\mathrm{f}_q'}(O_i) = \begin{cases} \exp -\dfrac{d^2}{2\sigma^2} & \text{if } \mathrm{ANN}\left(\mathrm{f}_q'\right) \in F_i \\ 0 & \text{otherwise} \end{cases}, i = 0, \cdots, N_O - 1. \tag{4}$$

To take advantage of the spatial locality, the score obtained in (4) is accumulated for each subregion as follows:

$$\mathrm{Score}_{\mathrm{r}}(O_i) = \sum_{\mathrm{f}_q' \in \mathrm{r}} \mathrm{Score}_{\mathrm{f}_q'}(O_i), i = 0, \cdots, N_O - 1. \tag{5}$$

For a subregion r, the model image having the highest score according to (5) is estimated as a candidate of the subregion as follows:

$$\mathrm{Candidate}(\mathrm{r}) = arg\max_{O_i} \mathrm{Score}_{\mathrm{r}}(O_i), i = 0, \cdots, N_O - 1. \tag{6}$$

After estimating the candidates for subregions, adjacent subregions having the same candidate are connected and called locally connected regions (LCRs). As shown in **Fig. 2**, an LCR roughly segments the object region. Because there may be several LCRs that estimate different candidates, the candidate estimated by the LCR having the highest score is determined as the final candidate O_c of I_q as follows:

$$O_c = \mathrm{Candidate}\left(I_q\right) = arg\max_{O_i} \mathrm{Score}_{I_q}(O_i), i = 0, \cdots, N_O - 1, \tag{7}$$

$$\text{where } \mathrm{Score}_{I_q}(O_i) = \sum_{\mathrm{r} \in LCR_{O_i}} \mathrm{Score}_{\mathrm{r}}(O_i), i = 0, \cdots, N_O - 1. \tag{8}$$

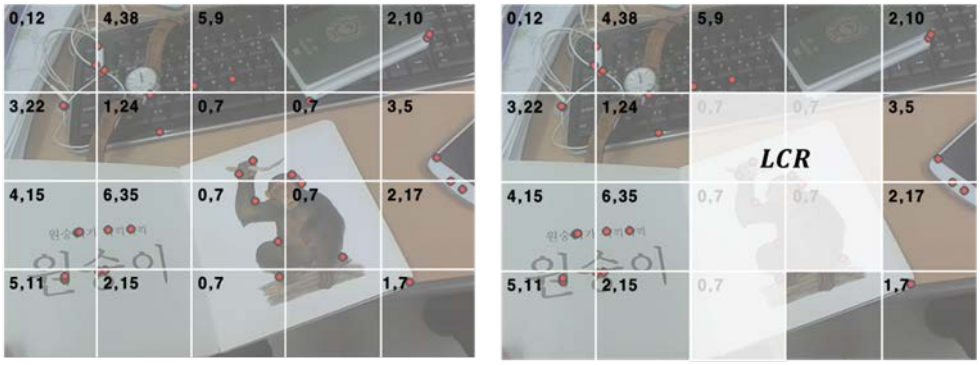


Fig. 2. (a) Estimation of regional candidate; (b) Locally connected regions.

5.3 Rough Pose Estimation

A single feature correspondence contains geometric transformations generated by the attributes of the corresponding keypoints. These transformations includes translations Δx and Δy, rotation Δo, and scale Δs, as shown in **Fig. 3**. By using the transformations to estimate the rough pose of the candidate, it is possible to speed up the verification process of the subsequent candidate. For this, we introduce the generalized Hough transform (GHT) [23], which generalizes the Hough transform to detect arbitrary shapes. The GHT allows each correspondence to vote for the bins of the corresponding transformation and accumulates these votes in a binned voting space, as shown in **Fig. 4** [24]. We use the transform combination corresponding to the most-voted bin as a rough pose.

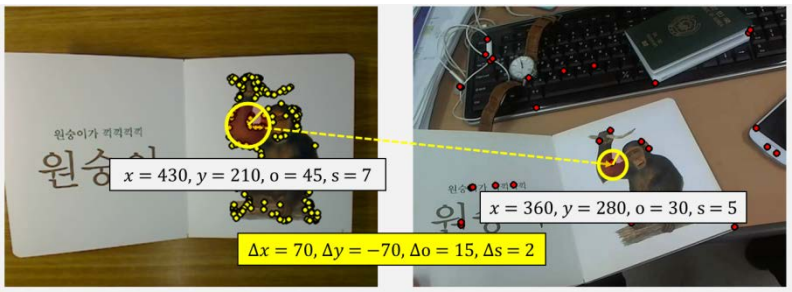


Fig. 3. A geometric transformation of a single feature correspondence.

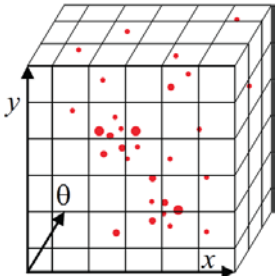


Fig. 4. An example of three-dimensional GHT voting space [24].

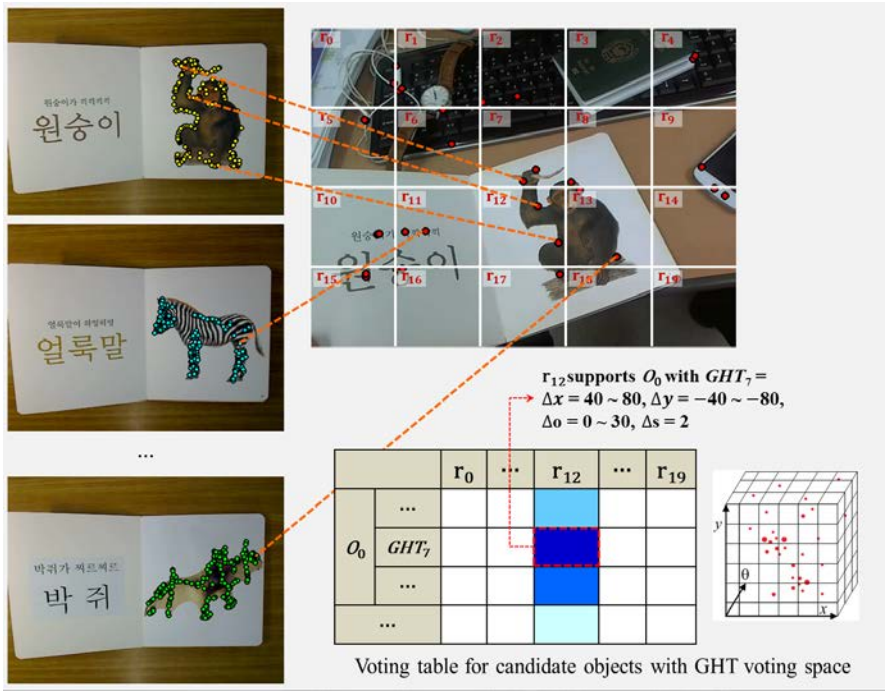


Fig. 5. Estimation of regional candidate and rough pose.

The voting is performed during the estimation process of the previous candidate. All correspondences vote in the GHT voting space of the candidates they support when the score for the candidate is calculated. The process of accumulating the results for each subregion and forming the LCR is performed simultaneously with the candidate estimation. Therefore, subregions that estimate the same candidate as the same rough pose form the LCR. **Fig. 5** illustrates this process. In our research, we define a four-dimensional GHT voting space for Δx, Δy, Δo, and Δs and set the range of each transformation to 40, 40, 30, and 1, respectively. We use a dynamic hash map for implementation because the voting space is very sparse in practice.

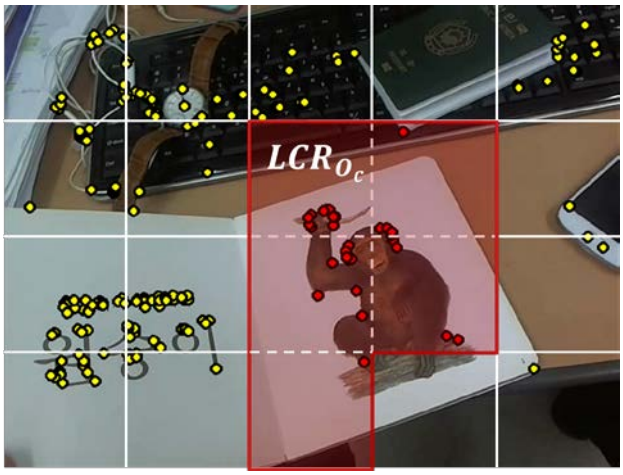


Fig. 6. Locally connected regional keypoints.

5.4 Candidate Verification and Pose Initialization

A verification process is performed to determine whether to recognize or reject the final candidate O_c while calculating the initial pose. Because LCR_{O_c} roughly segments O_c, we first select locally connected regional keypoints $P_{q,LCR_{O_c}}$, which belong to LCR_{O_c} among P_q, as shown in **Fig. 6**. We then extract locally connected regional feature vectors $F_{q,LCR_{O_c}}$ from $P_{q,LCR_{O_c}}$.

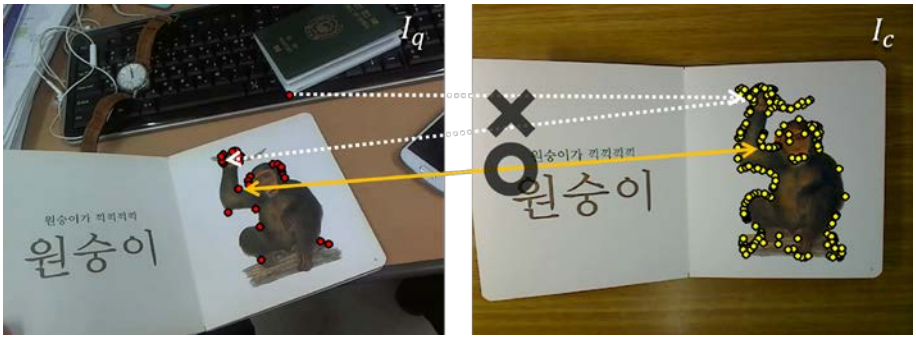


Fig. 7. An example of bidirectional matching.

The 3D pose calculation of O_c is based on feature matching between $F_{q,LCR_{O_c}}$ and F_c, where F_c is the feature vector extracted from I_c, which is model image of O_c. Because LCR_{O_c} still contains features extracted from the background, bidirectional matching is performed as a means of minimizing false matching, as shown in **Fig. 7**. For the distance metric, we use the squared L2 distance, which facilitates the acceleration of computation because of the property of our basic feature, as mentioned in Section 3. The matched pairs establish matching correspondences.

Fig. 8. Accelerating bidirectional matching using the estimated rough pose.

The process is further accelerated using the estimated rough pose. Because a feature pair that does not satisfy the estimated rough pose is not likely to be a matching correspondence, many operations can be reduced. For this reduction, a transformation defined by the GHT is obtained for all the pairs between $F_{q,LCR_{O_C}}$ and F_c, and bidirectional matching is performed only between the pairs having the same transform as the estimated rough pose. As shown in **Fig. 8**, a majority of the features are excluded for feature matching in both the directions. If no feature exists that has the same transformation as the estimated rough pose, the feature vector extraction process is also skipped.

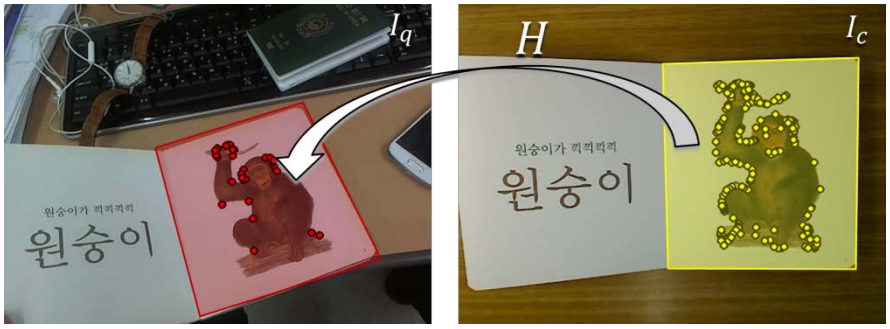


Fig. 9. An example of 2D transformation by using a homography.

Bidirectional matching minimizes falsely matched pairs called outliers but does not completely eliminate them. They must be removed because they obscure the judgment of the final recognition and cause errors in the calculation of the initial pose. For outlier removal, we use progressive sample consensus (PROSAC) [25], which is one of the random sample concensus (RANSAC) [26]-based algorithms. These algorithms are best suited for the purpose of this research because they perform not only outlier removal but also homography computation, which should be performed before the initial pose calculation. A homography is the most general model that can explain two-dimensional image transformation. **Fig. 9** shows the area where the model image is transformed into a query image by using a homography. Because PROSAC is used, matching correspondences consisting of only inliers and the homography calculated from them are returned.

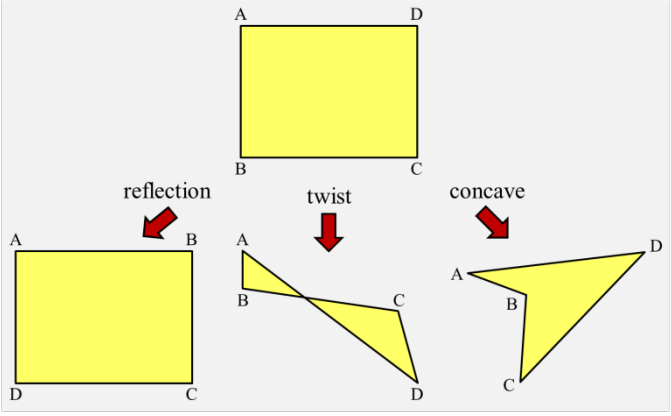


Fig. 10. Three examples of invalid homographies.

Finally, matching correspondences and the homography are evaluated to determine the final recognition. The candidate is recognized only when the number of inliers is higher than a

certain threshold and the homography is valid. Examples of invalid homographies include reflected, twisted, and concave shapes, as shown in **Fig. 10**. If it is judged that the candidate is recognized, the initial six-degrees-of-freedom pose of the recognized object is obtained by homography decomposition.

Table 1. Each set of Vienna dataset contains a different number of frames and different environmental changes.

Set	Environmental changes	# of frames
Simple	Smooth camera path	602
Occlusion	Partial occlusion of user interaction	1134
Tilt	Strong tilt	782
Fast movement	Strong blur caused by fast camera movement	928
Loss of target	Pointing camera away from and back to target	601

6. Experimental Results

6.1 Environments

We evaluate the performance of the proposed method using the Vienna dataset [14], which consists of five sets of images obtained from a single satellite image; each set contains a different number of frames and different environmental changes as shown in **Fig. 11** [14] and **Table 1**. The dataset contains a variety of environmental changes but excludes cluttered backgrounds, which are one of the most important environmental changes commonly encountered in AR scenarios. Thus, we create a Clutter set consisting of 500 frames with several textured cards placed around the target image, as shown in **Fig. 12**. We call the dataset containing the Clutter set as the Vienna+ dataset to distinguish it from the existing Vienna dataset. In addition, we add images of the UKBench dataset [17] to act as distractors when recognizing the target image, as shown in **Fig. 13**. In the following discussion, the number of distractors represents the number of objects to be recognized.

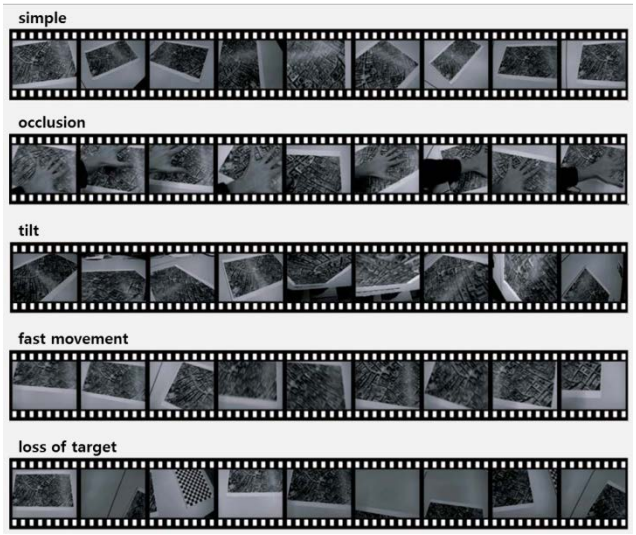


Fig. 11. Sample frames from each set of Vienna dataset [14].

Fig. 12. Clutter set created to consider cluttered backgrounds.

The performance is evaluated in terms of efficiency and accuracy. Efficiency measures the memory usage and the average time required to process one frame. Accuracy refers to the success rate of recognition and measures the ratio of the number of frames that are successfully recognized to the total number of frames. A recognition is considered to be successful if a valid homography is calculated with more than 15 inliers for the final candidate. All the experiments are performed on a Samsung Galaxy S5, a smartphone launched in 2014.

Fig. 13. Target image with distractor images.

6.2 Feature Comparison

To compare the performance of the feature itself, the recognition performance is evaluated using only the target image on the Vienna dataset. Our proposed feature is compared with the RF (a representative of a statistical classifier), ORB (a representative of binary features), and PhonySIFT (a representative of the mobile version of SIFT).

Table 2. Accuracy comparison of the features.

	Simple	Occlusion	Tilt	Fast movement	Loss of target	Total
Random Forest	0.80 (486/602)	0.77 (874/1134)	0.45 (355/782)	0.60 (565/928)	0.55 (336/601)	0.64 (2616/4047)
ORB	0.93 (565/602)	0.53 (608/1134)	0.37 (296/782)	0.74 (695/928)	0.54 (328/601)	0.61 (2492/4047)
PhonySIFT	0.96 (581/602)	0.82 (936/1134)	0.46 (363/782)	0.51 (482/928)	0.53 (323/601)	0.66 (2685/4047)
QPhonySIFT	0.94 (567/602)	0.75 (857/1134)	0.44 (349/782)	0.42 (397/928)	0.52 (317/601)	0.61 (2487/4047)

Table 2 lists the accuracy of the features for each set. The RF shows uniform performance. ORB is robust to blur (fast movement) but is vulnerable to tilt and occlusion, which are characteristics of binary features. On the other hand, PhonySIFT and proposed the feature are vulnerable to blur but robust to tilt and occlusion. Because of these results, we design the basic feature based on SIFT rather than binary features. In AR scenarios, image blur occurs more frequently in the tracking process where camera movement is fast, and occlusion and viewpoint changes occur more frequently in the recognition process. Overall, the proposed feature shows a recognition rate that is 5% lower than that of PhonySIFT. This lower rate is the result of the loss of information caused by quantization, which is performed to maximize efficiency.

Table 3. Computation time and memory usage comparison of the features.

	Computation time for each operation (ms)					Memory(kB)
	Detection	Description	Matching	PROSAC	Total	
Random Forest	5.1	-	11.2	2.9	19.2	10240
ORB	5.1	5.6	4.1	3.0	17.8	16
PhonySIFT	5.1	6.6	5.6	2.9	20.2	72
QPhonySIFT	4.9	6.6	2.5	3.0	17.0	18

Table 3 lists the efficiency of the features. Keypoint detection and PROSAC are used in all the features in the same manner, and keypoint description is omitted for the RF. All the features guarantee real-time performance, but ORB and the proposed feature have better performance in terms of computation time. The proposed feature saves approximately 3.1 ms of computation time compared to PhonySIFT because of the speed gain from the matching process. The RF requires more than 10 MB of memory to train one object. The other three features are excellent in terms of memory usage, but ORB and the proposed feature are the best. The memory efficiency of the proposed feature is four times that of PhonySIFT. In terms of the overall efficiency, ORB and the proposed feature are the best choices.

Table 4. Results of the baseline method according to the number of distractors.

	# of distractors					
	50	100	200	500	1000	2000
Success rate of recognition	0.58	0.53	0.48	0.39	0.29	0.21
# of inliers at recognition success	34.9	31.7	28.0	24.7	21.9	18.6
Success rate of candidate	0.73	0.72	0.72	0.71	0.69	0.66
Computation time (ms)	28.5	31.3	40.4	45.8	51.7	61.5

6.3 Baseline Method

To show the superiority of the proposed method, the baseline method is designed and the results are presented. The online recognition steps of the baseline method are as follows:

- Extract features from the query image (no consideration of spatial locality).
- Match the features to the feature space built from model images using the ANN searcher.
- Estimate the object corresponding to the most matching model image as a candidate.
- Verify the candidate by running PROSAC with only the results of the ANN searcher.

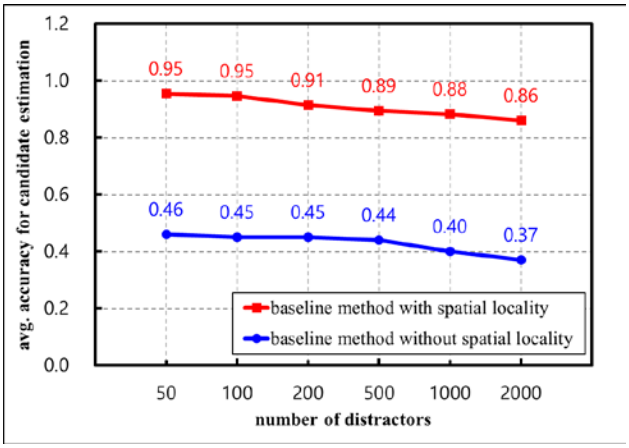


Fig. 14. Success rate of candidate estimation according to the number of distractors on the Clutter set.

Table 4 shows the results of the baseline method according to the number of distractors. As the number of distractors increases, the average number of inliers decreases at the same rate at which the recognition rate decreases. On the other hand, the success rate of a candidate is not greatly influenced by the number of distractors, and good results are obtained. This indicates that even though the number of distractors increases, the number of features matching the feature extracted from the target image is the greatest, although the number is significantly reduced. This implies that more sophisticated matching is needed after the candidate estimation. The computation time is also greatly influenced by the number of distractors. This is attributable to the overhead that occurs in the tree search because the depth of the tree constituting the ANN searcher becomes larger as the feature space becomes larger.

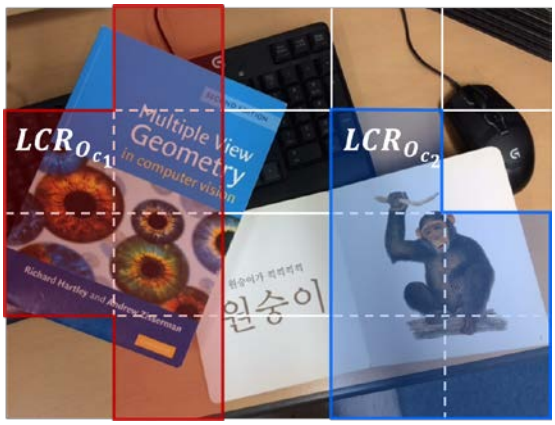


Fig. 15. Simultaneously recognizing multiple objects in an image.

6.4 Effect of Spatial Locality

To investigate the effect of spatial locality, we compare the baseline method with and without applying spatial locality to the Clutter set. **Fig. 14** shows the success rate of candidate estimation according to the number of distractors on the Clutter set. When spatial locality is not applied, it cannot overcome the cluttered background and shows severe performance degradation in candidate estimation. This is because many features extracted from the background influences the candidate estimation. On the other hand, when spatial locality is applied, it maintains high performance in the cluttered background.

The proposed spatial locality also opens the possibility of simultaneously recognizing multiple objects in an image. As shown in **Fig. 15**, two or more LCRs can exist in a query image because the candidate is estimated for each subregion based on spatial locality. If the candidate is verified independently for each LCR, we can recognize multiple objects simultaneously.

6.5 Scalability of Proposed Method

By performing experiments on the scalability of different aspects of the proposed method, including spatial locality, selective feature extraction, rough pose estimation, and selective feature matching, we verify the appropriateness of the proposed method for realizing large-scale standalone mobile AR.

Fig. 16 shows the accuracy of the proposed method according to the number of distractors. In contrast to the baseline method, which shows a degraded recognition rate as the number of distractors increases, the proposed method shows a stable recognition rate. The recognition rate of the proposed method decreases by only 5% as the number of distractors increases from 50 to 2,000. Compared to [14] (**Table 3**, PhonySIFT), which considers only a single object, it

is an encouraging achievement to extend the number of recognition objects to 2,000 at the expense of only an 11% reduction in the recognition rate.

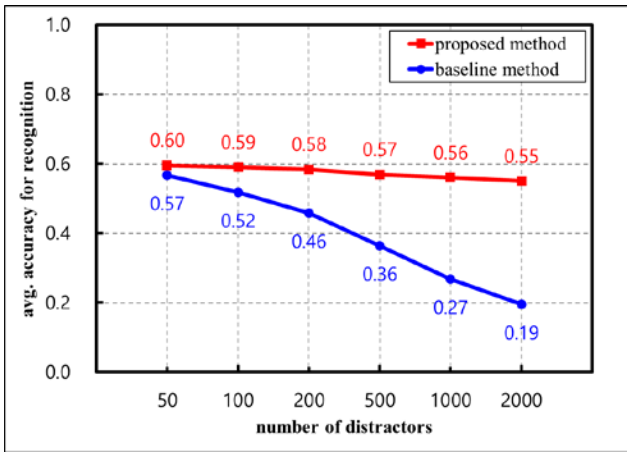


Fig. 16. Accuracy comparison between the proposed method and the baseline method according to the number of distractors.

Fig. 17 shows the efficiency of the proposed method according to the number of distractors. In the case of the computation time, the results of the baseline method are also visualized for comparison. Memory usage increases in proportion to the number of distractors, but memory consumption is only 36 MB when the number of distractors is 2,000 because of the efficient memory usage of the proposed basic feature. This is reasonable even considering mobile devices. As the number of distractors increases, the computation time is affected by the overhead caused by the ANN searcher, but the proposed method shows much improvement compared to the baseline method. This is because of the use of efficient techniques such as selective feature extraction and selective feature matching based on spatial locality. As the number of distractors increases from 50 to 2,000, additional operations that consume 8.4ms are required, but it is confirmed that the real-time performance is still maintained.

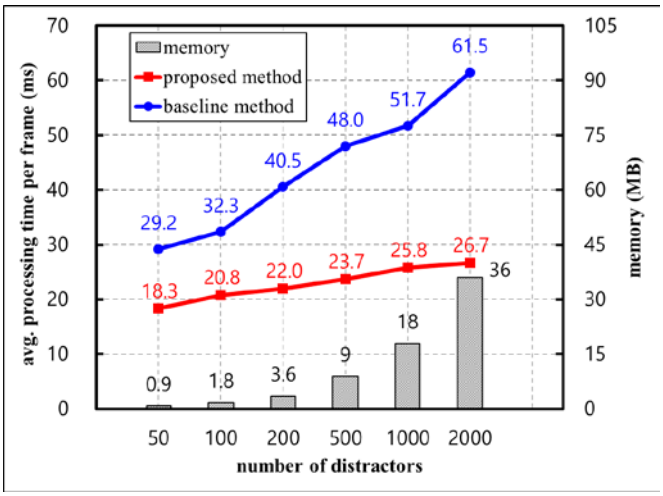


Fig. 17. Computation time comparison between the proposed method and the baseline method and memory usage of the proposed method according to the number of distractors.

7. Conclusion

In this paper, we have proposed a method for achieving real-time recognition with pose initialization to realize large-scale standalone mobile AR. The characteristics of the proposed method include basic feature design, spatial locality, selective feature extraction, rough pose estimation, and selective feature matching. Compared to the existing standalone mobile AR system, which recognizes only one object, we could extend the number of recognition objects by 2,000 at the expense of only an 11% reduction in the recognition rate. In addition, we have shown that the real-time recognition of 2,000 objects uses only 36 MB of memory on mobile devices. We have also shown that the proposed method is robust to various environmental changes including cluttered environments.

The proposed method provides a solution to mobile AR services that target a large number of objects, alleviates cost problems encountered in existing server–client-based systems, and maximizes the immersion of AR services. It is expected to be used not only in the diversification of scenarios provided by AR services but also in computer vision applications that recognize many specific objects in real time.

Experiments have shown that real-time operation can be performed on 2,000 objects, but this does not completely solve the scalability problem. This is attributable to the overhead incurred in the tree search because the depth of the randomized kd-trees constructed for the ANN searcher becomes larger as the feature space becomes larger. To solve this problem, additional research should be conducted to maximize the efficiency of the ANN searcher. In the proposed method, feature extraction in the training step is performed independently for each model image. If additional research is conducted toward extracting features of model images while considering the distribution of the feature space, it is expected that the accuracy can be further improved by reducing false matching occurring in the recognition step.

Acknowledgment

This work was supported by the the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (NRF-2018R1C1B5046098).

References

[1] Leptit, V., Fua, P., “Monocular Model-Based 3D Tracking of Rigid Objects: A Survey,” *Computer Graphics and Vision*, 1(1), 1-89, 2005. Article (CrossRef Link).

[2] Lowe, D. G., “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, 60, 91-110, 2004. Article (CrossRef Link).

[3] Rosten, E., Drummond, T., “Machine learning for high-speed corner detection,” in *Proc. of the 2006 European Conference on Computer Vision (ECCV), Graz, Austria*, pp. 430-443, 7-9, May 2006. Article (CrossRef Link).

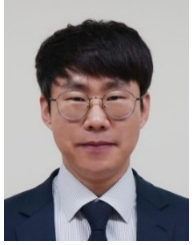
[4] Bay, H., Tuytelaars, T., Van Gool, L., “SURF:Speeded Up Robust Features,” in *Proc. of the European Conference on Computer Vision (ECCV), Graz, Austria*, pp. 404-417, 7-9, May 2006. Article (CrossRef Link).

[5] Mikolajczyk, K., Schmid, C., “A performance evaluation of local descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10), 1615-1630, 2005. Article (CrossRef Link).

[6] Calonder, M., Leptit, V., Strecha, C., Fua, P., “BRIEF: Binary robust independent elementary features,” in *Proc. of the European Conference on Computer Vision (ECCV), Crete, Greece*, pp. 778-792, 5-11, Sep 2010. Article (CrossRef Link).

[7] Rublee, E., Rabaud, V., Konolige, K., "ORB: an efficient alternative to SIFT or SURF," in *Proc. of the International Conference on Computer Vision (ICCV), Barcelona, Spain*, pp. 2564-2571, 6-13, Nov 2011. Article (CrossRef Link).
[8] Leutenegger, S., Chli, M., Siegwart, R. Y., "BRISK: Binary robust invariant scalable keypoints," in *Proc. of the International Conference on Computer Vision (ICCV), Barcelona, Spain*, pp. 2548-2555, 6-13, Nov 2011. Article (CrossRef Link).
[9] Alahi, A., Ortiz, R., Vandergheynst, P., "FREAK: Fast retina keypoint," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA*, pp. 510-517, 16-21, June 2012. Article (CrossRef Link).
[10] Leptit, V., Fua, P., "Keypoint recognition using randomized trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9), 1465-1479, 2006. Article (CrossRef Link).
[11] Ozuysal, M., Calonder, M., Leptit, V., "Fast keypoint recognition using random ferns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3), 448-461, 2010. Article (CrossRef Link).
[12] Lee, S., Lee, S. W., Chae, Y. N., Yang, H. S., "Lightweight Random Ferns Using Binary Representation," in *Proc. of the International Conference on Pattern Recognition (ICPR), Tsukuba, Japan*, pp. 1342-1345, 11-15, Nov 2012. Article (CrossRef Link).
[13] Lee, S., Yang, H. S., "Lightweight generic random ferns for multi-target augmented reality on mobile devices," *IET Electronics Letters*, 49(13), 800-802, 2013. Article (CrossRef Link).
[14] Wagner, D., Mulloni, A., Schmalstieg, D., "Real-time detection and tracking for augmented reality on mobile phones," *IEEE Transactions on Visualization and Computer Graphics*, 16, 355-368, 2010. Article (CrossRef Link).
[15] Cho, K., Jung, J., Lee, S. W., Lim, S. O., Yang, H. S., "Real-time recognition and tracking for augmented reality books," *Computer Animation and Virtual Worlds*, 22(6), 529-541, 2011. Article (CrossRef Link).
[16] Jung, J., Ha, J., Lee, S. W., Rojas, F. A., Yang, H. S., "Efficient mobile AR technology using scalable recognition and tracking based on server-client model," *Computers & Graphics*, 36, 131-139, 2012. Article (CrossRef Link).
[17] Nister, D., Stewenius, H., "Scalable recognition with a vocabulary tree," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), New York, NY, USA*, pp. 2161-2168, 17-22, June 2006. Article (CrossRef Link).
[18] Tuytelaars, T., Mikolajczyk, K., "Local invariant feature detectors: a survey," *IEEE Transactions on Visualization and Computer Graphics*, 3, 177-280, 2008. Article (CrossRef Link).
[19] Harris, C., Stephens, M., "A combined corner and edge detector," in *Proc. of the 1988 Alvey Vision Conference (AVC), Manchester, UK*, pp. 23.1-23.6, 1988. Article (CrossRef Link).
[20] P. L. Rosin, "Measuring corner properties," *Computer Vision and Image Understanding*, 73, 291-307, 1999. Article (CrossRef Link).
[21] Silpa-Anan, C., Hartley, R., "Optimised kd-trees for fast image descriptor matching," in *Proc. of the 2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Anchorage, AL*, USA, pp. 1-8, 23-28, June 2008. Article (CrossRef Link).
[22] Beis, J., Lowe, D., "Shape indexing using approximate nearest-neighbour search in high dimensional spaces," in *Proc. of the 1997 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Juan, Puerto Rico*, pp. 1000-1006, 17-19, June 1997. Article (CrossRef Link).
[23] Ballard, D., "Generalizing the Hough Transform to detect arbitrary shapes," *Pattern Recognition*, 13, 111-122, 1981. Article (CrossRef Link).
[24] Leibe, B., Schindler, K; Van Gool, L., "Coupled object detection and tracking from static cameras and moving vehicles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20, 1683-1698, 2008. Article (CrossRef Link).
[25] Chum, O., Matas, J., "Matching with PROSAC-progressive sample consensus," in *Proc. of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, USA*, pp. 220-226, 20-26, June 2005. Article (CrossRef Link).

[26] Fischler, M. A., Bolles, R. C., “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, 24(6), 381-395, 1981. Article (CrossRef Link).



Suwon Lee is an assistant professor of Dept. of Computer Science at Gyeongsang National University. He received his B.S. degree in Computer Engineering at Kyungpook National University, and his M.S. and Ph.D. degrees in Computer Science at KAIST. His research interests include computer vision, machine learning, augmented reality and human computer interaction.