Register Based on Large Scene for Augmented Reality System

Zhen-Wen Gui

Department of Information System, China Electronics Technology Group Corporation No. 7 Research Institute, China
whut_gzw@163.com

Abstract

Register is steadily gaining in importance due to the drive from various computer vision applications, such as augmented reality (AR), mobile computing, and human-machine interface. Efficient keypoint-based approachs are widely used in scene register. These approaches often model a scene as a collection of keypoints and associated descriptors, and then construct a set of correspondences between scene and image keypoints via descriptor matching. Finally, these correspondences are used as input to a robust geometric estimation algorithm such as RANSAC to find the transformation of the scene in the image. This paper focuses on designing a robust and flexible registration method for wide-area augmented reality applications. Firstly, we propose to partition the whole scene into several sub-maps according to the user's preference or the requirements of the AR applications instead of building a global map of the wide-area scene. Secondly a linear structured SVM classifier is used to perform scene learning online, which allows us to quickly adapt our model to a given environment. Finally, a hybrid tracking strategy is implemented by combining both wide and narrow baseline techniques. Some experiments have been conducted to demonstrate the validity of our methods.

Keywords: Augmented reality, Wide-area registration, Scene recognition

1 Introduction

Augmented reality (AR) is to add virtual objects to real environments, allowing computer-generated 3-D graphics or 2-D information to be overlaid on the real world in such a manner as to enhance people's understanding of the real word [1-2]. Registration between real and synthetic worlds is one of the major technological issues in order to create AR systems. As the user moves his/her head or viewpoint, the virtual objects must be properly aligned with the objects in the real world, or the coexistence of the virtual world and the real world will be compromised.

Keypoint-based method for scene register has become a cornerstone of modern computer vision, enabling great advances in areas such as AR and simultaneous localization and mapping (SLAM) [3]. These register approaches model an object as a set of keypoints which are matched independently in an input image. Robust estimation procedures based on RANSAC [4-6] are then used to determine geometrically consistent sets of matches which can be used to infer the presence and transformation of the scene.

The applications of keypoint-based include location recognition [7-10], autonomous robot navigation [11-13] and augmented reality [14-16]. Broadly speaking, there are two prevalent approaches that have been used to achieve registration for AR. The first addresses SLAM method, where the camera is localized within an unknown scene. The second method use the knowledge of a prior map or 3D scene model. Several recent methods fall in the second method [7-8, 14, 17]. And this renewed interest has been sparked by progress in structure from motion (SFM) [18-19], which makes it possible to easily reconstruct large scenes in great detail.

Despite the scalability of recent approaches [7-8, 17], real-time keypoint-based register in large environments remains a challenging problem. As the scene gets larger, recognizing unique identifiable landmarks becomes more challenging. In [7-8, 17], this difficulty is overcome by using sophisticated image features such as SIFT [20]. But these are too expensive to compute in real-time. On the other hand, some visual SLAM [15, 21-22] systems are real time, but their performance degrades in larger scenes, where map maintenance becomes progressively expensive. These techniques are also fragile if the camera moves too quickly, which makes them less attractive for persistently computing a precise camera pose over longer durations.

In this paper, we propose a new approach for continuously register within wide area. Our algorithm is real-time and runs over long periods with low fluctuations in the frame-rate by avoiding the need for scale-invariant keypoints on running. This is a key distinction from prior approaches [7-10], which rely on the scale-invariance of SIFT [20]. Keypoints (Binary robust invariant scalable keypoints) [23] from one frame are tracked in the consecutive frame by optical flow tracker, and are recovered by matching to

*Corresponding Author: Zhen-Wen Gui; E-mail: whut_gzw@163.com
DOI: 10.3966/160792642020012101010

candidate keypoints within a local search neighborhood [24] in the next frame when they are lost.However, matching features across different scales are important for reliable 2D-to-3D matching, and we address this requirement by computing redundant descriptors during offline processing.

In this paper, we propose to employ a unified framework for scene register such that overall detection performance can be optimized given a set of training images. Furthermore, because the linear structured SVM is used to perform learning which allows us to quickly adapt our model to a given environment. As a result, our algorithm maintained a good robustness and stability.

2 Related Work

SLAM and online SFM are two kinds of prevalent techniques that have been used to achieve wide-area registration for AR [25]. Visual SLAM systems have recently been used for realtime AR [26], by utilizing parallel threads for tracking and mapping (PTAM) [15], using multiple local maps [21] and performing fast relocalization through random ferns [22]. However, these approaches are susceptible to the problem of drift and error accumulation in the pose estimate, and existing solutions for fast relocalization do not scale to larger scenes. SFM is used to estimate 3D coordinates for the landmarks in recent work [7-8, 17]. The approaches scale well and some variants by introducing the GPU [7]. However, these methods address only the single-image registration problem, and are not fast enough for real-time registration from video.

Real-time registration approaches for AR on smart vehicles have also been recently proposed [16, 21, 27]. However, these approaches derive their speedup from tracking relatively fewer features which make them less suitable for continuous 6-DOF localization in larger scenes or over longer durations. An efficient approach for tracking and registration in video was proposed in [15, 28] employ online SFM to get denser and larger maps to improve tracking quality. The tracking techniques attempt to construct a global 3-D map that covers the whole observed scene to realize camera tracking in wide-area environments. However, a global map is often not getable due to the complexity of natural scene or the immaturity of mapping techniques.

Some research has been carried on scene learning and recognition problems for AR applications [29-31]. Lee and Hollerer [31] proposed to solve the online scene recognition problem by matching SIFT features between input frame and previously stored features directly. The scene has the maximum number of matched features among all scenes which is returned as the recognition result if the number of matched features exceeds a certain threshold. Klein and Murray [15, 28] proposed to facilitate the scene recognition processes by using keyframes. Each keyframe is represented by a descriptor which will be compared to the input frame's descriptor by using NCC to find the closest match to assistant the scene recognition processes. Experimental results indicate that the methods can deal with hundreds of keyframes in real-time. However, this is not enough to meet the requirement of our research since our system may contain thousands or even tens of thousands of keyframes. In our research, we proposed to deal with scene learning problem by using SVM classifiers. Each scene is represented by a predefined numbers of 3D points and local binary features detected from the ten keyframes. These local features will be used to train the multi-index table built in advance for online scene recognition. In this paper, we demonstrate that the tasks can be separated as individual threads to further improve tracking performance on scene learning and camera tracking.

3 Natural Features Detecting and Matching

This section introduces the natural features detecting and matching methods used in our system. In 2011, the BRISK [23] are found to be eminently suitable for low-power device because of its feature detection speed and memory efficiency. They are been chose for real-time AR systems in general.

Nearest neighbor (NN) search on binary codes has been widely applied in image recognition [24, 32], local features matching [33-34] and parameter estimation [35] recently. There has been growing interest in mapping image data onto compact binary codes for fast near neighbor search in scene recognition [32, 34, 36]. Because binary codes comparisons require just a small number of machine instructions; millions of binary codes can be compared to a query in less than a second. But the most compelling reason for binary codes which are used as direct indices (addresses) into a hash table, yielding a dramatic increase in search speed compared to an exhaustive linear scan as shown in [37]. In practice, using binary codes as hash indices is not necessarily efficient. To find near neighbors one needs to examine all hash table entries (or buckets) within some Hamming ball around the query. And the number of such buckets grows near-exponentially with the search radius. Later, in order to improve the search speed compared to linear scan, many researchers like Greene et. Al [37] and Mohammad et al. [38], who present an algorithm to search exact K-nearest neighbor on binary codes and implement a form of multi-index hashing, in which binary codes from the database are indexed m times into m different hash tables. Given a query code, entries that fall close to the query in at least one such substring are considered to be the neighbor candidates, which are then checked for validity using the entire

binary code to remove any non-r-neighbors presented in [38]. While promising, there are some disadvantages: The first one is large computational cost. Because the searching algorithm will be expensive to examine all buckets within r bits of a query even if the vast majority of the buckets in a full hash table will be empty with n binary codes of b bits by using their algorithm to build hash table, since $2^b \gg n$. The second disadvantage is the complexity of algorithm. Because it is not all items in the merged buckets that are r-neighbors of the query. As in actual query, it becomes complex when searching algorithm need to cull any candidate that is not a true r-neighbor. The final disadvantage is memory efficiency. The same binary codes are indexed m times into m different hash tables based on m disjoint binary substrings and have been stored repeatedly in hash table.

By considering the above-mentioned discussions, this paper focuses on the searching efficiency and scalability of binary code indexed when it runs on the smartphone. Firstly the binary ordinal is used as the index node to create multi-level index and sub-vector numbers (1, 2, ···, c) which are used as the keywords for inverted list as shown in Figure 1. Different from [39], this matching algorithm means that the retraining of the entire sampling descriptors are not needed when the number of the sampling images increase and only the training of the sub-vectors of the descriptors dynamically inserted to the corresponding list without changing the index. Finally, the index is used for the cluster center and radius R is used as the hamming threshold of the sub-vectors so as to all the sub-vectors of image descriptors are assigned to the corresponding linked list. Detailed steps are:

(1) Calculate hamming distances between the sub-vectors and the cluster center.

(2) When the hamming distance is less than or equal to the threshold R, descriptor ID and image ID of sub-vector are inserted into the inverted list according to the serial number of sub-vectors.

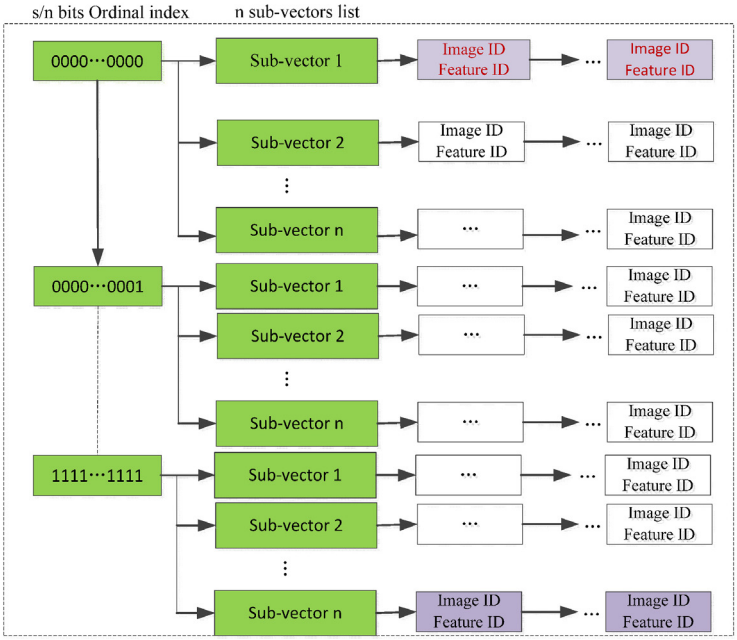


Figure 1. Ordinal multi-index list

In order to avoid search in the whole sampling datasets, the binary descriptor vector E is divided into c disjoint sub-vectors and the linking of these sub-vectors in sequence will form the original vector E. If the binary descriptor vector E comprising s bits is partitioned into n disjoints sub-vectors $E_1, E_2, \cdots, E_n$, each sub-vector length is $\lfloor s/n \rfloor$ or $\lceil s/n \rceil$bits. Based on the proposition in [39], if two binary descriptor vectors A and B differ by r bits or less, at least one of their n sub-vectors must differ by at most $\lceil r/n \rceil$ bits, i.e. when $\|A-B\|_H \le r$, there exists a sub-vector i, $1 \le i \le m$, as follows:

$$\|A_i \oplus B_i\|_H \le \left\lfloor \frac{r}{n} \right\rfloor \quad (1)$$

Proof of (1) follows straightforwardly from the Pigeonhole Principle. If Hamming distance between each of the m substrings is strictly greater than $\lfloor r/n \rfloor$, then $\|A-B\|_H$ must be larger than r and vice versa.

The significance of (1) arises from the fact that the sub-vector has only s/n bits, and the required search radius in each substring will reduce to $\lfloor r/n \rfloor$. For example, if A and B differ by 5 bits or less and n = 6, then at least one of the 5 sub-vectors must be identical. If they differ by at most 9 bits, they will differ by no more than 1 bit in at least one sub-vector, which means

that when searching a Hamming radius of 9 bits only a radius of 1 bit on each of 6 sub-vectors is searched. More generally, instead of examining L (s, r) ordinal tables, it suffices to examine $L\left(\lfloor s/n \rfloor, \lfloor r/n \rfloor\right)$ index nodes in each of n sub-vector ordinal tables.

One ordinal table is built for each of the n sub-vectors of the binary descriptors in a given dataset. For a query Q with sub-vectors $\{Q_i\}_{i=1}^{n}$, the i^{th} sub-vector ordinal table for index entries that are within a Hamming distance $\lfloor r/n \rfloor$ to Q_i is searched, thereby image and feature id is retrieved as a set of candidates which are denoted as $T_i(Q)$. According to the above-mentioned proposition, the union operation on the n sets $T(Q) = U_i T_i(Q)$ is used to obtain the nearest neighboring descriptor set of Q. The last step of the algorithm is to compute the Hamming distance between Q and each candidate in $T(Q)$, and retain the top 50 sample descriptors of the shortest distance which satisfy the hamming distance threshold.

4 Scene Building and Training

Many steps including scene reconstruction and reference image detection are completed in the offline stage which is foundation of the online registration process. This stage includes: a set of images are required to select by capturing from a different perspective views. These images are called key-frame which recording a variety of important known information for scenes. As shown in Figure 2, ten different perspective views are selected as a set of key frames.

Figure 2. 3D scene reconstruction

4.1 Scene Building

Before scene reconstruction, internal parameters of the camera need to be calibrated. Zhang [33] have proposed a calibration method which is widely applied in teaching and research region because it relatively easy to implement and has fewer restrictions by the environment. And the popular OPENCV (Open source computer vision library) is developed by Intel which includes the Zhang's method as the primarily calibration method. So, the Zhang's method is also used in this paper and the camera's internal parameters are kept unchanging for subsequent registration in the online tracking.

In this paper, the SFM algorithm is used to reconstruct three-dimensional structure of the scene as in [34] and capture 10 key-frames at different view point for every scene, as shown in Figure 3. The inherent difficulty in extracting suitable features from an image lies in balancing two competing goals: high quality description and low computation. BRISK algorithm [23] has a significant speed advantage in the key-point detection, description and matching as well as scale invariant to a significant extent which achieves high quality performance comparable to the state-of-the-art algorithms while dramatically reducing computational cost. So this approach can be applied to the object recognition and matching, 3D scene reconstruction and motion tracking in particular for tasks with hard real-time constraints or limited computation power.

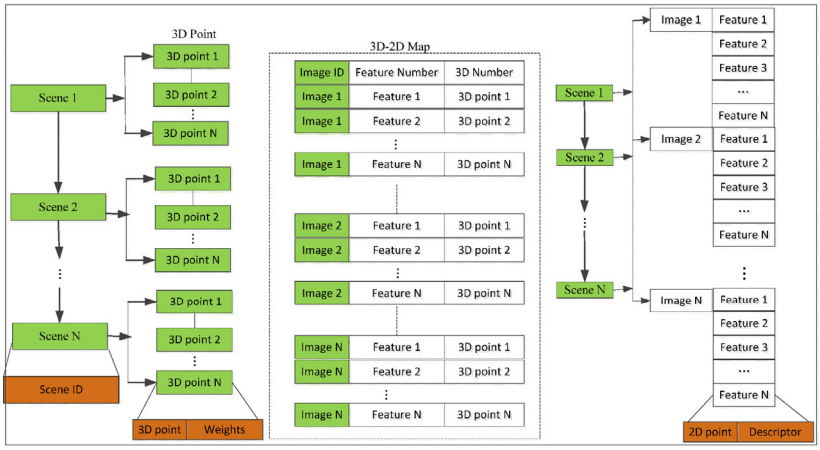


Figure 3. 2D-3D map table

Scene building is to get the data (key-frames and 3D features) that will be used in key-frames learning and real time registration. We use the camera phone to get the video of the scene in which virtual objects will be augmented, and then build the structure of this scene on a PC by using key-frames based SFM technique. Once the recognized scene structure is loaded, the next step is to initialize the camera for tracking use.

4.2 2D-3D Mapping Table Construction

Before feature tracking, the 2D-3D mapping table needs to be constructed in order that the 3D points of scene are fast found by the corresponding 2D features of key-frame on the same scene. And each 3D point may be corresponding to multiple 2D features on key-frames, so the mapping table only stores key-frame ID, 3D points and the corresponding 2D pointer in order to reduce memory in practical applications, such as Figure 3.

As illustrated in Figure 4, when the image ID and the two-dimensional (2D) feature sequence of image are known, the storage location of three-dimensional points can be obtained. So the three-dimensional point information including space coordinate, weights which can be accessed by the three-dimensional point location. The weights of 3D point will be obtained by SVM training in the next section. The 2D feature includes a two-dimensional coordinates in the image and a bit-string descriptor of length 512. After the mapping table has been established, all three-dimensional points and their corresponding feature descriptors are saved in files for online registration.

4.3 Training 3D Point Weights

In order to calculate the correct transformation between scene model and the current image, the correspondence of 2D feature and matching 3D points need to be ensured precision. The larger correct matching pairs, the more accurate the transformation between scene model and the current image. The common methods are used to choose the optimal transformation matrix P by calculating the highest score or counting max number of matching points to match up the transformation matrix as in (2). If these methods are directly used to implement, they are may be unfeasible for smartphones because all related transformation matrix is calculated with expensive computational cost. In this paper, firstly the 3D points are set an initial weights to calculate the highest score of transformation matrix. Secondly, the cycle times are set like PROSAC approach when the scoring values do not increase as the termination condition. This method avoid computing scores for all transformation matrix [40]. The detail implementation as equation (3), (4), (5), (6), $I=\{x_1, x_2 ..., x_k\}$ is provided for the 2D point coordinate of the image, $D=\{d_1, d_2 ..., d_k\}$ is the corresponding descriptors, $M=\{x_1, x_2 ..., x_k\}$ is the 3D points, $C=\{(X_j, x_k, s_{jk}) \mid X_j \in M, x_k \in I, s_{jk} \in R\}$ is the matching set for 2D points and 3D points, S_{jk} is their matching score, R is the set of scores.

$$P = \arg\max_{P' \in T} S(M, I, P') \tag{2}$$

Where M represents a scene model, I represents the scene image, P' is a transformation matrix between scene model and the image, T represents a collection of all the transformation matrices, S represents the score function, P represents the transformation matrix of the maximum value.

$$S(C, P) = \sum_{(X_j, x_k) \in C} E(\| x_k - P(X_j) \|_2 < \tau) \tag{3}$$

$$S_w(C, P) = \sum_{(X_j, x_k) \in C} E(\| x_k - P(X_j) \|_2 < \tau) =< w, L(C, P) > \tag{4}$$

$$L_j(C, P) = \begin{Bmatrix} d_k & \exists (X_j, x_k) \in C : \| x_k - P(X_j) \|_2 < \tau \\ 0 & \text{otherwise} \end{Bmatrix} \tag{5}$$

$$L(C,P)=[L_1(C,P),L_2(C,P)\ldots,L_J(C,P)]^T \ \mathrm{L}_j(1\le j\le J) \tag{6}$$

Where $w=[w_1,w_2\ldots,w_J]^T$ represent a collection of weight for the 3D point and P is chosen as best projection matrix when his score is maximum. The 10 sample images are acquired to learn, the more prominent points, the higher the performance of the set weights. $L(C, P)$ consists of a set of feature descriptors corresponding to 3D points.

The weights w of 3D point are obtained by training the samples $\{(I_i,P_i)\}_{i=1}^N$ as in [42]. They are used to distinguish the real transformation matrix P_i and the other alternative matrices by maximizing the score of these matrices as in (7).

$$\min_{w,\xi}\ \frac{\lambda}{2}\|w\|^2+\sum_{i=1}^N\xi_i \tag{7}$$

$$\forall \mathrm{i},\forall \mathrm{P}\neq \mathrm{P}_i:\delta S_w^i(P)\ge\Delta(P_i,P)-\xi_i$$
$$\delta S_w^i(P)=S_w(C_i,P)-S_w(C_i,P_i) \tag{8}$$

Where λ is balance factor, which keep the trade-off of the training accuracy and weight vector regularization. $\Delta(P_i,P)$ is the loss function which is used punish wrong choice P in place of P_i as (14) by using the difference of the matching points between the projection matrix P and P_i to pay penalties.

$$\Delta_I(P,P')=|S(C,T)-S(C,T')| \tag{9}$$

5 Online Learning and Tracking

The online ultimate goal is to achieve registration for tracking scenes. On this stage, we firstly need to load scene structure, mapping table, key-frame descriptors into memory that have been completed on offline stage. If these have already in memory, we would not have loaded. The main processes are described as follows:

Step (1) Get the current image, detect feature and describe the image;

Step (2) Recognize scene by searching multi-index list;

Step (3) Calculate the initial camera pose. When the current scene is identified, online learning is run to establish the corresponding relationship between image 2D points and scene 3D points for computing the initial pose matrix of the camera;

Step (4) Track the feature points of the subsequent frame by optical flow method. Online learning is still continued to carry out for the current image which establish the correspondence between the image 2D point and scene 3D point for the subsequent frame. So as to compute the current camera pose relative to initial position;

Step (5) Build registration matrix by assembling two pose matrix;

Step (6) Restore the lost feature or restart the scene recognition, when the number of tracking feature points is less than a threshold value.

5.1 Acquiring Current Images

The current images are obtained by an external camera and preprocessed to $320*240$ grayscale images and $320*240$ RGB images simultaneously. The grayscale images are used for scene recognition and registration. The RGB images are used to overlay virtual objects. The BRISK algorithm is run to extract features of the current image using the OpenCV2.4.7 library.

5.2 Scene Recognition

We now turn to the online recognition problems with the above modified matching method. The features are firstly extracted from the current frame. Then all features are rotated according to the orientations of BRISK descriptors to resist camera rotations and smoothed by Gaussian filter which reduce the influence of image noise. The transformed features are compared with each feature from the sample images to find candidate matching features based on Hamming distance of their feature vectors. A scene is identified by considering the sum of the matching points (subject to a threshold) between the current frame features $\{f_1,f_2,\cdots,f_N\}$ and the sample image features $\{e_1,e_2,\cdots,e_M\}$. The image with the maximum matching points in the sample library is selected to decide the scene category corresponding to current frame. The matching points of sample images L are counted as (10):

$$class(\{f_1,f_2,\cdots,f_N\})=\arg\max\sum_{i=1}^M\sum_{j=1}^N pairs_l<e_i,f_j> \quad l=1,\cdots,L \tag{10}$$

5.3 Scene Learning

In the tracking process, the algorithm of online scene learning is performed to update w weights to adapt to the changing environment. The main steps of the online learning process can be summed up as follows:

Firstly, the method of Section 4 is used to match the detected BRISK features between an active frame and previously stored features. The similar key frame is identified according to the matching points.

Secondly, we look up 2D- 3D mapping table to obtain the correspondence between the 3D point of the scene and the features of the current image according to relative information of the scene and the key frame;

Thirdly, the RANSAC algorithm is performed to remove the mismatching points for the current image

and the key frame. With the obtained RANSAC intermediate results, we calculate the score as Section 4 and select the matching point pairs with the maximum matching score to compute the three-dimensional registration matrix;

Finally, the weight w is subsequently updated to adapt to environmental changes. The weight updating algorithm [42] is applied to renew the w for each 3D point of the scene by computing a transform matrix between the current image and the 3D scene, With the transform matrix computed, we choose two transformation matrices with the highest score and the second highest, and use these scores to update the weights w, given as follows.

$$w_j^{t+1} \leftarrow (1-\eta_t\lambda)w_j^t + E(\max_{P\neq P_t}\{\Delta(P_t,P)-\delta S_w^i(P)\}>0)\eta_t\alpha_j^t$$
$$+E(u_j\in C_t^*)E(\max_{k'\neq k}\{1-<w_j,d_k-d_{k'}\}>0)\eta_t v\beta_j^t \quad \textbf{(11)}$$

$$\alpha_j^t = L_j(C_t,P_t) - L_j(C_t,\hat{P}) \quad \textbf{(12)}$$

$$\beta_j^t = d_k - d_{\hat{k}} \quad \textbf{(13)}$$

$$\hat{P} = \arg\max_{P\neq P_i}\{\Delta(P_i,P)-\delta S_w^i(P)\} \quad \textbf{(14)}$$

$$\hat{k} = \arg\max_{k'\neq k}\{1-<w_j,d_k-d_{k'}>\} \quad \textbf{(15)}$$

where j is 3D point number and t is the current frame number. w_j^t is the weight of the feature j for the current frame and w_j^{t+1} is the weight for the next frame. P_t represents the transformation matrix with the highest score for the current frame . $\eta_t = 1/\lambda t$ is the step size and λ is balance factor which is used to weigh training accuracy and weight vector regularization.

If the weight calculated algorithm [42] is directly used to update the weight w, all present transform matrices need to be computed to select the two matrices with the highest score and the second highest value for the current frame, which produce heavy computation.

In this paper, the algorithm is improved to reduce the computation cost for scoring by using the intermediate results of the RANSAC algorithm to calculate the maximum score and the second maximum score for updating w. The maximum score is approximately computed by the inliers and 3D points after the RANSAC algorithm performed as in (4). The second maximum score is obtained by the intermediate second largest value which is less than the approximate maximum score.

5.4 Pose Estimating

With the obtained mapping relationship for the current image features m_t and the 3D points M_t of corresponding scene , the internal parameters K of the camera K is known, we compute a more robust and stable camera pose $T=[R|t]$ by using PNP [41] algorithm, as follows:

$$m_t = \lambda K[R|t]M_t \quad \textbf{(16)}$$

Where R represents the rotation matrix, t is the translation matrix. Tukey M-estimation [42] algorithm is used to calculate the camera pose so as to improve the accuracy of parameter estimation and strengthen tolerance for error matching. The general M-estimator minimizes the residual $\min_T \sum_{i=1}^{n}\rho(r_i)$ for error function to obtain the optimal estimator of parameters. Where σ is the Tukey biweight objective function which can be computed as:

$$\left\{\begin{array}{l} if\ |x|\leq c, \sigma(x)=\frac{c^2}{6}[1-(1-(\frac{x}{6})^2)^3] \\ if\ |x|>c, \sigma(x)=\frac{c^2}{6} \end{array}\right\} \quad \textbf{(17)}$$

Where c is the standard deviation derived from all the residuals. The pose matrix T can be calculated by the iterative optimization method.

5.5 Augmented Display

Registration between real and synthetic worlds is one of the major technological issues in order to create AR systems. As the user moves his/her head or viewpoint, the virtual objects must be properly aligned with the objects in the real world, or the coexistence of the virtual world and the real world will be compromised. In order to achieve the accurate overlay for virtual objects, we need to obtain the transformation matrix T_{cp} between the current camera coordinate and the world coordinate. The transformation matrix T_{cp} can be obtained as

$$T_{cp} = T_{ck} \bullet T_{kp} \quad \textbf{(18)}$$

Where T_{ck} as the initial registration matrix between original camera coordinate and the world coordinate, T_{kp} is the transformation matrix between current camera position and the initial camera position.

With the obtained registration matrix, you can use OpenGL (Open Graphics Library) to achieve mapping for virtual objects to the surrounding environments which are superimposed onto the correct position.

5.6 Feature Tracking

In augmented reality registration process, if each pose computation needs to detect, describe and match points which spend a large of memory and time making the performance of algorithm deterioration. However, in practical applications, the user move relatively gentle and do not abruptly or violently shake

of the camera or move significantly the camera in a scene, so the pose change is very little between adjacent frames. The positions of the points are continuity between adjacent frames which can be efficiently estimated in the next frame and be used to quickly calculate the pose of the camera. The optical flow algorithm is used to predict the position for features between consecutive frames in this augmented reality system which improves the speed for feature tracking and reduce the time for repeatedly detecting and matching feature.

When the initialization has been done, the optical flow tracker is used to track features between consecutive frames, and then correspondence between feature points x_{t+1} with the 3D points X_{t+1} is established, finally the PNP algorithm is applied to calculate the camera pose.

Optical flow algorithm [43] uses temporal changes of pixel intensities in the sequence images and the correlation in the image sequences to determine the movement for pixel point, which based on the following assumptions: constant brightness between adjacent frames, small movement for object between adjacent frames. We use optical flow algorithm to calculate the movement for feature pixel point, given as follows:

$$I(x, y, t) = I(x + dx, y + dy, t + dy) \tag{19}$$

$I(x, y, t)$ represents time t with the gray value of the pixel (x, y). The Taylor series expansion as in followings:

$$I(x + dx, y + dy, t + dy) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt \tag{20}$$

Experimental results show that only a few dozen milliseconds are spend to calculate the feature point coordinates between adjacent frames by using optical flow algorithm. However, optical flow tracker suffered from losing features. This is especially true in the case of features going out of the field of view or occluded by users and some scene objects. Thus, the valid matches will become less and less during tracking process, which will finally result in the failure of registration.

In this paper, we firstly set the threshold T_1 to ensure the registration validity (herein T_1 is set to 30). When the tracking points are less than T_1 for the current frame, we need to restart extracting and matching features. If it is the failure for matching between the three consecutive frames and sample images, the user maybe has left the current scene into a new scene, so we will need to re-shoot key frames of a new scene for reconstruction. Secondly, we set the threshold T_2 to ensure the registration accuracy (herein T_2 is set to 40). If the number of tracking points of optical flow are less than T_2 (herein T_2 is set to 40), the following operations are performed to recover the lost features.

5.7 Lost Features Recovering

Optical flow algorithm is prone to produce drift problem when the camera moving for a long time that the tracking points become less and less. In order to ensure the accuracy of the registration, we perform the operation for lost features recovering when the tracking points are less than T_2. Duan [44] propose an approach to recover lost features by using the homography between the current and reference keyframes. The approach is firstly perform the image recognition to select three key frames belonging to of the same scene with the highest matching score, and secondly calculate the homography between three keyframes and the current frame, and then predict the position of lost features in the current frame by the homography, finally use the SSD (Sum of squared difference) method to find the matching features within $\pm$ 20 pixels surrounding the predicted position to recover the lost features.

The experiments [44] show that the approach can recover the feature points accurately, but also it introduces the amount of computation to perform re-image recognition, which influence the real-time performance for registration. Guan [26] proposes a method by using single keyframe and the projection matrix to recover lost features. To recover the lost features, the method firstly computes the candidate positions of the lost features on the current image using the calculated projection matrix. Then the patches of the lost features on the reference keyframes are transformed using the homography between the current and reference keyframes. Finally the correlation operations are performed with the features detected within ± 10 pixels surrounding the reprojection to recover the lost features. The recovered feature is the point that has the maximal NCC (normalized cross-correlation) score with the transformed patch on the reference image. The experiments [26] show that the method can recover the lost feature rapidly, but the NCC method itself is sensitive to illumination changes, partial occlusion or deformation and has poor robustness.

Therefore, in order to quickly recover the lost feature of the current frame, we use BRISK features to describe lost features producing lower computation cost. BRISK features can also improve the robustness of the recovery features to the environment which is invariance to light and rotation. At the same time, we do not perform image recognition in addition to the tracking failure on whole process.

The tracked points are firstly applied to compute the homography H_k^c between the current and reference key frames as in (21). Where $(x_k, y_k, 1)$ is a feature of the keyframe and $(x_c, y_c, 1)$ is a feature for the current frame, the positions of the lost features are estimated in

the current frame by the homography H_k^c. For example given the feature f in the keyframes, $(f_x, f_y, 1)$ is its 2D position which is lost in the current frame so that position are predicted to $(f_x^{'}, f_y^{'}, 1)$ in the current frame by the homography H_k^c. Secondly the BRISK features are extracted within ±10 pixels surrounding the $(f_x^{'}, f_y^{'}, 1)$ in the current frame. Then the detected feature points from the current image are matched with predicted feature. Finally, if selected feature point $f_{\min}$ is the nearest distance to f than the other features within ±10 pixels surrounding the f, the points will be recovered otherwise lost.

$$\begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = H_k^c \begin{bmatrix} x_k \\ y_k \\ 1 \end{bmatrix} \tag{21}$$

When the number of lost feature is larger, the recovery time is longer. In order to ensure real-time performance for registration algorithm, the maximal number of lost features is set. When the number reaches the maximal value (herein it is set to 80) we is no longer to recover lost features and continue to perform optical flow algorithm to track consecutive frames.

6 Experiments and Results

The proposed algorithm is implemented on smartphone with1.7GHz processor and 2G RAM. The software is written in JAVA and NDK C++ on Android 4.1 operating system with MicroSD (32G).

Experimental dataset include five scenes in UKBENCH dataset and eight outdoor scenes shot on campuses altogether 13 scenes, and the resolution of video frames are set to $320*240$. BRISK [23] algorithms are used to detect feature points and generate descriptors respectively, and then the descriptors are trained and stored to construct multi-index searching engine.

The number of features are detected and limited to not more than 400 in sample images. The virtual 3D model, 3DS MAX is used to render virtual objects and export OBJ file format to three-dimensional model. In experiment, the proposed algorithm is tested including register result, tracking accuracy and speed. Tracking accuracy is measured by the RMS (root mean square) errors (the square root of difference of actual pixel coordinates and the coordinates of the predicted point) [45]; tracking speed is measured by frame rate (frames can be handled per second).

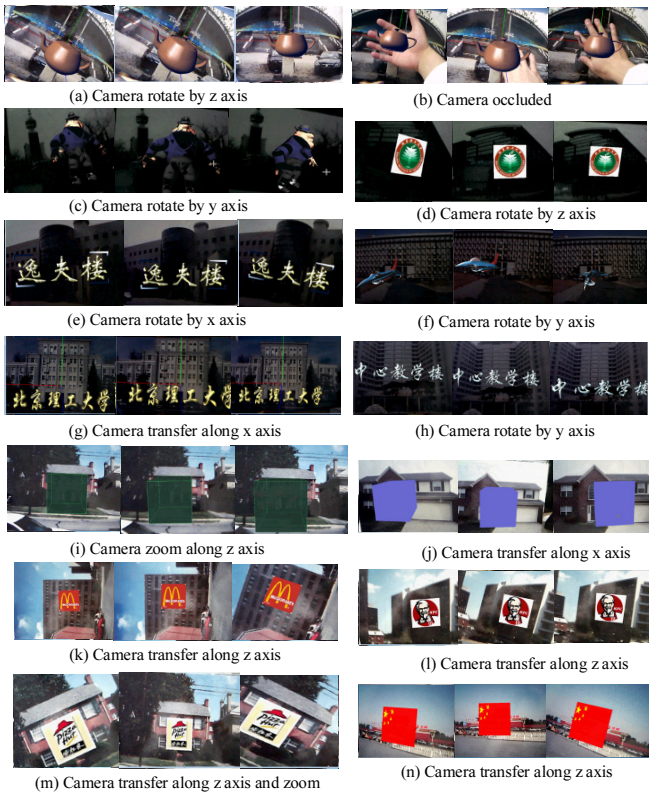
RMS error by is calculated as follows:

$$RMS = \sum_j \| PX_j - x_j \|^2 \tag{22}$$

Wherein, x_j represents a 2D feature positions on a frame, X_j means the corresponding 3D position of the point x_j, P indicates pose matrix of the camera corresponding to current frame.

6.1 Registration Result

The first experiment is taken to demonstrate the robustness of the proposed algorithm in different perspectives, different distances and different lighting conditions. As can be seen from Figure 4(a)-(f), under different natural environment, even in the dark part of the scene illumination or camera rotate along different axes, the proposed algorithm can be accurately complete the registration in real time. Left part of **Figure** 4 shows registration result of the campus scene, the right part shows registration result on scene of UKBENCH library. Figure 4(g)-(n) gives the results with changes of the volumes, viewing angles, and illumination, respectively. The proposed algorithm is able to successfully identify all the sub-scenes and switch between them automatically to complete exact registration.



(a) Camera rotate by z axis (b) Camera occluded
(c) Camera rotate by y axis (d) Camera rotate by z axis
(e) Camera rotate by x axis (f) Camera rotate by y axis
(g) Camera transfer along x axis (h) Camera rotate by y axis
(i) Camera zoom along z axis (j) Camera transfer along x axis
(k) Camera transfer along z axis (l) Camera transfer along z axis
(m) Camera transfer along z axis and zoom (n) Camera transfer along z axis

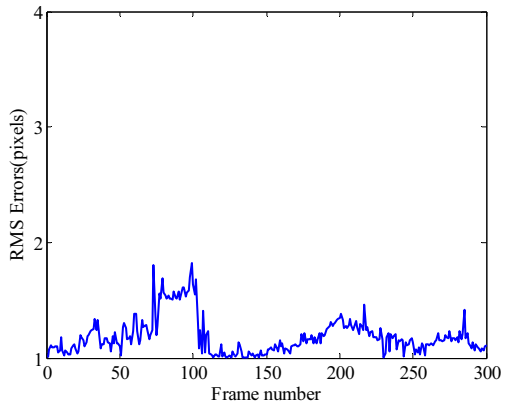
Figure 4. Tracking and register result

6.2 Registration Accuracy

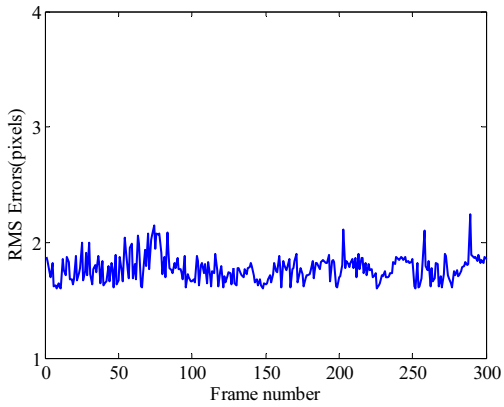
We continuously track 300 frames and use RMS errors to test the acsnrfkcfdgdcebnejy7curacy of registration. We are especially interested in the RMS errors under the circumstance of changes in rotations and zooming ratio. The movement patterns of camera are simulated by rotating along X, Y, Z axis or moving

from far to near the scene along the Z-axis to test the registration accuracy. Figure 5(a-c) gives the RMS errors of the proposed method when camera rotates along Z-axis from 0 to 60. The purpose is to simulate the case when users make large changes in view angles. Figure 5(d) gives the RMS errors when camera zooms. The purpose is to simulate the case when users move close to or far from the scene.

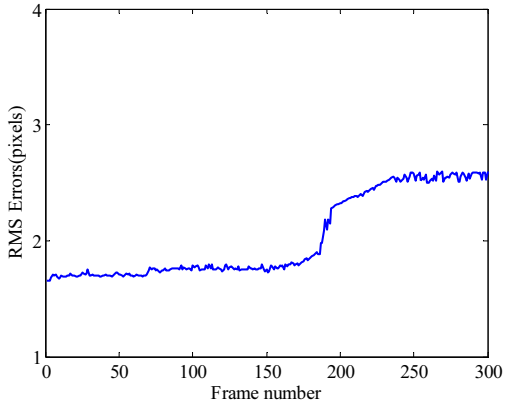
Figure 5(a) shows RMS errors are less than one pixel when the camera when the camera rotates 0-60 degrees along Z-axis. Figure 5(b) shows RMS errors are close to 2.5 pixels when the camera 0-40 degrees along Y-axis. Figure 5(c) shows RMS errors are close to 2.8 pixels when the camera rotates 0-40 along the X-axis. Figure 5(d) shows RMS errors are close to 2 pixels when moving users move close to or far from the scene. All the above errors are below 3 pixels which demonstrate the accuracy of the proposed method.



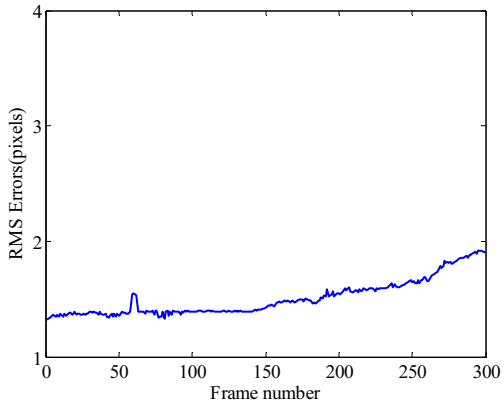
(a) Camera rotate 0-60° along z axis



(b) Camera rotate 0-40° along y axis



(c) Camera rotate 0-40° along x axis

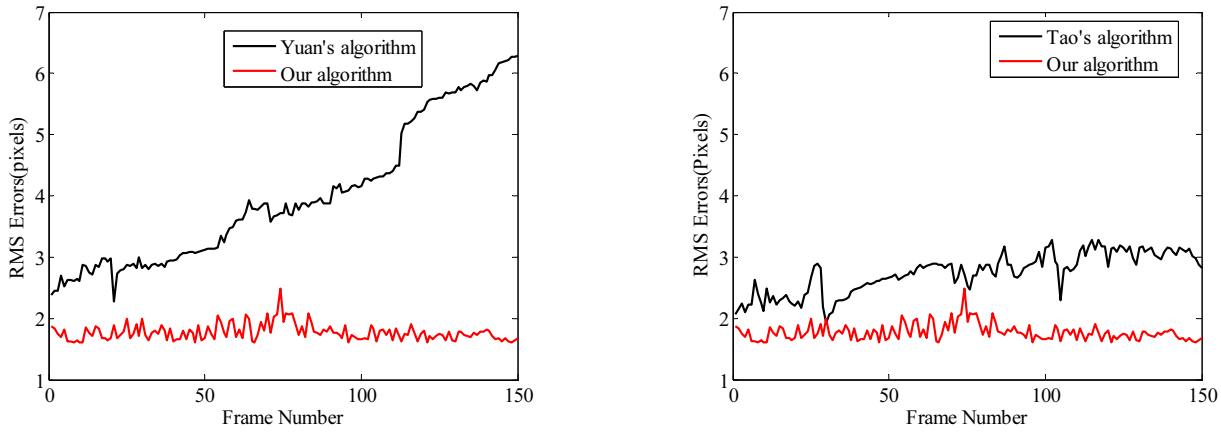


(d) Camera move along the z axis

Figure 5. RMS error for the proposed algorithm

We carry out two experiments to compare the RMS errors obtained using the proposed feature tracking method with other methods. The comparisons made are (1) comparison with the Yuan's KLT-based tracking method and (2) comparison with the Tao's random classification matching method.

In the first experiment, we compared the registration accuracy of the proposed approach with the method given by Yuan et al. [8]. In this experiment, we changed the camera view point angle from 0 (the camera's optical axis is orthogonal to the real scene) to 40 in the live video sequence. The recorded RMS errors are shown in Figure 6(a). We can see that the errors of the Yuan's method deteriorate over time. This is mainly due to the direct use of the NCC to recovery the lost features in tracking processes, whereas our method can always provide reasonable tracking results since the lost patches are warped by homography to take account of viewpoint changes between the patch's first observation and the current camera position. This experiment convincingly proves the validity of the proposed method based lost feature recovery method.

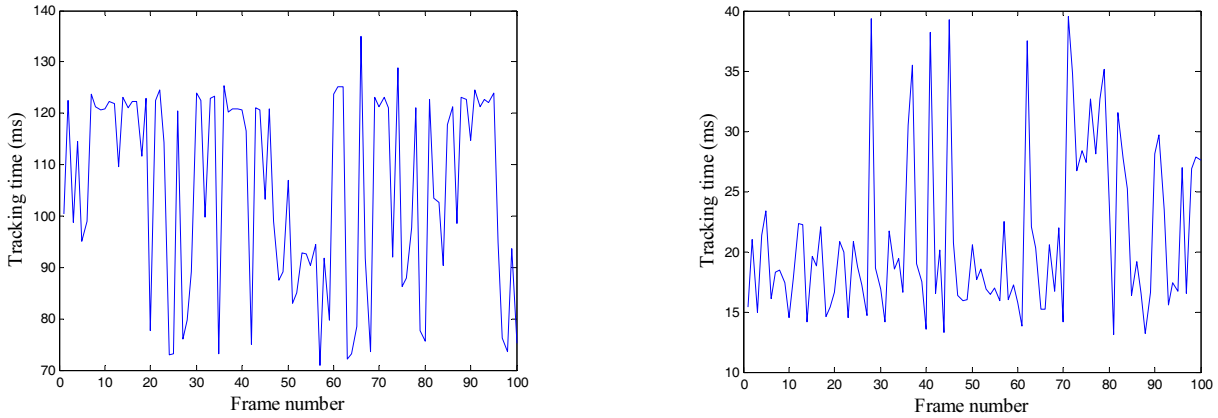


(a) Our approach vs yuan' algorithm

(b) Our approach vs Tao' algorithm

Figure 6. Our approach vs other methods

In the second experiment, we compare our method with the Tao's method [26] in which a sub-map baseline strategy is used to match features between the current and reference frames directly. Figure 7(b) gives the RMS errors recorded in this experiment. We can see that our method is more stable and accurate than the Tao's method, which really demonstrates the effectiveness of the proposed feature tracking method.



(a) Tracking time for matching approach

(b) Tracking time for optical flow approach

Figure 7. Time comparison of two approach

6.3 Timings

The second experiment is carried out to test and verify the tracking performance of our approach by evaluating the time-consumption for camera pose computation. In our approach, the camera pose is calculated by the tracked features of optical flow algorithm. The time cost of BSRIKS algorithm or optical flow algorithm will impact the overall performance. The processing time of two tracking methods are tested on continuous multi-frame images as shown in Figure 7. As can be seen in Figure 7(a), matching algorithm consumes time within 70ms to 140ms and in Figure 7(b), the time of optical flow tracking method is below 40ms.

Table 1 and Table 2 show the computation time of the various steps of the online registration process. As can be seen from Table1, it mainly consists of the following parts: Camera initialization and camera registration. Table 1 shows that the average scene recognition time is less than 70ms when the camera is initialized. Table 2 the registration time is less than 40ms and the frame rate is up to 25f/s can meet the requirements for real-time augmented reality applications when camera pose are tracked online.

Table 1. Initialization time

Camera Initialization	
Step	**Time (ms)**[a]
Feature points detecting(BRISK)	22.19
Scene recognition	48.65
Scene learning	1.07
Total	78

Table 2. Registration time

Camera registration	
Step	**Time (ms)**[a]
Feature tracking	2.4
Pose calculating	10.3
Lost feature recovering	25
Total	37.7

7 Conclusion

Register for augmented reality on smartphone is a challenging task. In this paper, several new approaches are proposed to improve the stability of the Register performance: a linear structured SVM classifier is used to perform scene learning online, which allows us to quickly adapt our model to a given environment; and the hybrid tracking strategy is implemented by combining both wide and narrow baseline techniques. The proposed approaches are very efficient and demonstrate excellent performance for large scene.

Acknowledgments

This research is supported by the National Science and Technology major projects under Grant No. 2018YFB1802401.

References

[1] L. Y. Duan, T. Guan, B. Yang, Registration Combining Wide and Narrow Baseline Feature Tracking Techniques for Markerless AR Systems, *Sensors*, Vol. 9, No. 12, pp. 10097-10116, December, 2009.

[2] Y. David, U. Efron, The Image Transceiver Device: Studies of Improved Physical Design, *Sensors*, Vol. 8, No. 7, pp. 4350-4364, July, 2008.

[3] S. Hare, A. Saffari, P. H. S. Torr, Efficient Online Structured Output Learning for Keypoint-Based Object Tracking, *IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, 2012, pp. 1894-1901.

[4] O. Chum, J. Matas, Matching with PROSAC- Progressive Sample Consensus, *Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, CA, USA, 2005, pp. 220-226.

[5] M. A. Fischler, R. C. Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, *Communications of the ACM*, Vol. 24, No. 6, pp. 381-395, June, 1981.

[6] P. H. S. Torr, A. Zisserman, MLESAC: A New Robust Estimator with Application to Estimating Image Geometry, *Computer Vision and Image Understanding*, Vol. 78, No. 1, pp. 138-156, April, 2000.

[7] A. Irschara, C. Zach, J.-M. Frahm, H. Bischof, From Structure-from-motion Point Clouds to Fast Location Recognition, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, 2009, pp. 2599-2606.

[8] Y. Li, N. Snavely, D. P. Huttenlocher, Location Recognition Using Prioritized Feature Matching, *European Conference on Computer Vision*, Heraklion, Crete, Greece, 2010, pp. 791-804.

[9] D. Robertson, R. Cipolla, An Image-based System for Urban Navigation, *15th Annual British Machine Vision Conference (BMVC)*, Kingston University, London, UK, 2004, pp. 819-828.

[10] G. Schindler, M. Brown, R. Szeliski, City-scale Location Recognition, *Computer Society Conference on Computer Vision and Pattern Recognition (ICRA)*, Minneapolis, MN, USA, 2007, pp. 1-7.

[11] L. Meier, P. Tanskanen, F. Fraundorfer, M. Pollefeys, PIXHAWK: A System for Autonomous Flight Using Onboard Computer Vision, *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 2992-2997.

[12] E. Royer, M. Lhuillier, M. Dhome, J.-M. Lavest, Monocular Vision for Mobile Robot Localization and Autonomous Navigation, *International Journal of Computer Vision*, Vol. 74, No. 3, pp. 237-260, September, 2007.

[13] M. Achtelik, M. Achtelik, S. Weiss, R. Siegwart, Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In- and Outdoor Environments, *IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 3056-3063.

[14] Z. Dong, G. F. Zhang, J. Y. Jia, H. J. Bao, Keyframe-based Real-time Camera Tracking, *International Conference on Computer Vision*, Kyoto, Japan, 2009, pp. 1538-1545.

[15] G. Klein, D. Murray, Parallel Tracking and Mapping for Small AR Workspaces, *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nara, Japan, 2007, pp. 225-234.

[16] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, D. Schmalstieg, Real-Time Detection and Tracking for Augmented Reality on Mobile Phones, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 16, No. 3, pp. 355-368, May-June 2010.

[17] T. Sattler, B. Leibe, L. Kobbelt, Fast Image-based Localization Using Direct 2d-to-3d Matching, *International Conference on Computer Vision (ICCV)*, Barcelona, Spain, 2011, pp. 667-674.

[18] Y. Jeong, D. Nister, D. Steedly, R. Szeliski, I.-S. Kweon, Pushing the Envelope of Modern Methods for Bundle Adjustment, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 34, No. 8, pp. 1605-1617, August, 2012.

[19] N. Snavely, S. M. Seitz, R. Szeliski, Modeling the World from Internet Photo Collections, *International Journal of Computer Vision*, Vol. 80, No. 2, pp. 189-210, November, 2008.

[20] D. G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, *International Journal of Computer Vision*, Vol. 60, No. 2, pp. 91-110, November, 2004.

[21] R. O. Castle, G. Klein, D. W. Murray, Wide-area Augmented Reality Using Camera Tracking and Mapping in Multiple Regions, *Computer Vision and Image Understanding*, Vol. 115, No. 6, pp. 854-867, June, 2011.

[22] B. Williams, G. Klein, I. Reid, Real-Time SLAM Relocalisation, *11th International Conference on Computer Vision*, Rio de Janeiro, Brazil, 2007, pp. 1-8.

[23] S. Leutenegger, M. Chli, R. Y. Siegwart, BRISK: Binary Robust Invariant Scalable Keypoints, *2011 International Conference on Computer Vision*, Barcelona, Spain, 2011, pp. 2548-2555.

[24] D. Ta, W. Chen, N. Gelfand, K. Pulli, SURFTrac: Efficient Tracking and Continuous Object Recognition Using Local Feature Descriptors, *IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, 2009, pp. 2937-2944.

[25] T. Guan, C. Wang, Registration Based on Scene Recognition and Natural Features Tracking Techniques for Wide-Area Augmented Reality Systems, *IEEE Transactions on Multimedia*, Vol. 11, No. 8, pp. 1393-1406, December, 2009.

[26] A. J. Davison, I. D. Reid, N. D. Molton, O. Stasse, MonoSLAM: Real-Time Single Camera SLAM, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 6, pp. 1052-1067, June, 2007.

[27] C. Arth, D. Wagner, M. Klopschitz, A. Irschara, D. Schmalstieg, Wide Area Localization on Mobile Phones, *8th IEEE International Symposium on Mixed and Augmented Reality*, Orlando, FL, USA, 2009, pp. 73-82.

[28] G. Klein, D. W. Murray, Improving the Agility of Keyframe-based SLAM, *10th European Conference on Computer Vision*, Marseille, France, 2008, pp. 802-815.

[29] T. Guan, L. Duan, Y. Chen, J. Yu, Fast Scene Recognition and Camera Relocalisation for Wide Area Augmented Reality Systems, *Sensors*, Vol. 10, No. 6, pp. 6017-6043, June, 2010.

[30] T. Guan, L. Duan, J. Yu, Y. Chen, X. Zhang, Real-Time Camera Pose Estimation for Wide-Area Augmented Reality Applications, *IEEE Computer Graphics and Applications*, Vol. 31, No. 3, pp. 56-68, May-June, 2011.

[31] T. Lee, T. Hollerer, Multithreaded Hybrid Feature Tracking for Markerless Augmented Reality, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 15, No. 3, pp. 355-368, May-June, 2009.

[32] A. Alahi, R. Ortiz, P. Vandergheynst, FREAK: Fast Retina Keypoint, *IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, 2012, pp. 510-517.

[33] Z. Y. Zhang, A Flexible New Technique for Camera Calibration, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 11, pp. 1330-1334, November, 2000.

[34] B. Delabarre, E. Marchand, Camera Localization Using Mutual Information-based Multiplane Tracking, *IEEE International Conference on Intelligent Robots and Systems*, Tokyo, Japan, 2013, pp. 1620-1625.

[35] H. Bay, A. Ess, T. Tuytelaars, L. V. Gool, Speeded-up Robust Features (SURF), *Computer Vision and Image Understanding*, Vol. 110, No. 3, pp. 346-359, June, 2008.

[36] E. Rosten, T. Drummond, Machine Learning for High-speed Corner Detection, *European Conference on Computer Vision*, Graz, Austria, 2006, pp. 430-443.

[37] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: An Efficient Alternative to SIFT or SURF, *International Conference on Computer Vision*, Barcelona, Spain, 2011, pp. 2564-2571.

[38] M. Calonder, V. Lepetit, C. Strecha, P. Fua, Brief: Binary Robust Independent Elementary Features, *European Conference on Computer Vision*, Heraklion, Crete, Greece, 2010, pp. 778-792.

[39] Z. Gui, Register Based on Efficient Scene Learning and Keypoint Matching for Augmented Reality System, *IEEE International Conference on Image, Vision and Computing*, Portsmouth, UK, 2016, pp. 79-85.

[40] I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, Large Margin Methods for Structured and Interdependent Output Variables, *Journal of Machine Learning Research*, Vol. 6, No. 5, pp. 1453-1484, September, 2005.

[41] C. Lu, G. Hager, E. Mjolsness, Fast and Globally Convergent Pose Estimation from Video Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 6, pp. 610-622, June, 2000.

[42] G. Medioni, S. B. Kang, *Emerging Topics in Computer Vision*, Prentice Hall, 2004.

[43] P. Weinzaepfel, J. Revaud, Z. Harchaoui, C. Schmid, DeepFlow: Large Displacement Optical Flow with Deep Matching, *IEEE International Conference on Computer Vision*, Sydney, NSW, Australia, 2013, pp. 1385-1392.

[44] L. Duan, T. Guan, Y. Luo, Wide Area Registration on Camera Phones for Mobile Augmented Reality Applications, *Sensor Review*, Vol. 33, No. 3, pp. 209-219, June, 2013.

[45] M. L. Yuan, S. K. Ong, A. Y. C. Nee, Registration Using Natural Features for Augmented Reality Systems, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 12, No. 4, pp. 569-580, July-August, 2006.

Biography

Zhen-Wen Gui receieved the B.S. at Hunan University of Science and Technology of China, Xiangtan, China, the M.S at Wuhan University of Technology of China, Wuhan, China and Ph.D. at Beijing Institute of Technology, Beijing, China. He is currently with Department of Information System, China Electronics Technology Group Corporation NO. 7 Research Institute, China. His research interest spans Digital Image Processing, Computer Vision.