International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:11, No:2, 2017

Enhanced Planar Pattern Tracking for an Outdoor Augmented Reality System

L. Yu, W. K. Li, S. K. Ong, A. Y. C. Nee

***Abstract*—**In this paper, a scalable augmented reality framework for handheld devices is presented. The presented framework is enabled by using a server-client data communication structure, in which the search for tracking targets among a database of images is performed on the server-side while pixel-wise 3D tracking is performed on the client-side, which, in this case, is a handheld mobile device. Image search on the server-side adopts a residual-enhanced image descriptors representation that gives the framework a scalability property. The tracking algorithm on the client-side is based on a gravity-aligned feature descriptor which takes the advantage of a sensor-equipped mobile device and an optimized intensity-based image alignment approach that ensures the accuracy of 3D tracking. Automatic content streaming is achieved by using a key-frame selection algorithm, client working phase monitoring and standardized rules for content communication between the server and client. The recognition accuracy test performed on a standard dataset shows that the method adopted in the presented framework outperforms the Bag-of-Words (BoW) method that has been used in some of the previous systems. Experimental test conducted on a set of video sequences indicated the real-time performance of the tracking system with a frame rate at 15-30 frames per second. The presented framework is exposed to be functional in practical situations with a demonstration application on a campus walk-around.

***Keywords*—**Augmented reality framework, server-client model, vision-based tracking, image search.

I. Introduction

THE popularization of smartphones has brought up new research topics in various research fields. Mobile Augmented Reality (AR) evolves with the advent of powerful handheld devices. AR enhances a user’s perspective by overlaying virtual information on top of the camera view with accurate 3D registration. Early mobile AR systems are cumbersome, requiring a heavy backpack to carry the PC, sensors, display unit, batteries and other equipment with cable connections. One example is the early work of the “touring machine” created by Feiner et al. [1]. Compared with early wearable computing devices, the size and weight of current smartphones have been reduced dramatically although mobile devices still have their limitations on computational abilities. Another limitation of most mobile AR frameworks is the tracking scalability. Since tracking requires relatively intensive computation power, the number of tracking targets to be processed for each frame is extremely small. The recognition system on the other hand helps to solve the scalability issue. A framework, that has the capability to process tracking targets from a large database and facilitates a wide range of applications, is termed a scalable AR framework in this paper. Some of the research works focused on porting previous desktop-based recognition and tracking systems to mobile devices with algorithm modification and hardware acceleration. Wagner et al. [3] brought markerless tracking to mobile devices with modified classic descriptors. Guan et al. [2] proposed on-device mobile visual location recognition using vision and sensor integrated recognition algorithms. Their work demonstrates the feasibility of functional on-device recognition and tracking system. OpenGL ES provides the information displaying ability for mobile devices. As proposed in the concept of Augmented Reality 2.0 [4], widely deployable mobile AR experience can be achieved by combining user generated contents, geographical location services and various web services through web APIs. The proposed research is a practical implementation of the Augmented Reality 2.0 concept.

Recognition and tracking are the enabling technologies of a scalable AR framework. Image recognition and vision-based 3D tracking are fundamentally two different procedures, each of which attracts large amount of research effort in parallel. Research works in these two fields influence each other. For example, the classic vocabulary-tree based object recognition algorithm by Nister and Stewenius [5] and the randomized tree- based keypoint recognition algorithm for 3D tracking [6] use similar approaches based on a tree-structure for time and space efficiency. In a scalable AR framework, recognition of tracking targets among a large database introduces the scalability property to the system while on-device 3D tracking ensures tracking accuracy and efficiency.

Server-client structure based large-scale AR tracking systems have been discussed in some research works [7]-[9]. Gammeter et al. [7] proposed a server-client approach in which a very large database can be plugged into a mobile AR system. Takacs et al. [8] proposed a system for mobile phones that matches camera-phone images against a large database of location-tagged images using a robust image retrieval algorithm. In their work, feature matching is still performed on the mobile device. On the server side, the large database is preprocessed so that it is clustered into loxel-based feature stores, and a small cluster of the large database is communicated constantly from the server to the client. In this work, only image recognition is performed, and no real-time

L. Yu (Dr) was a PhD candidate at the Mechanical Engineering Department, National University of Singapore (e-mail: a0068412@u.nus.edu).
W. K. Li is a PhD candidate with the Integrative School for Science and Engineering at the National University of Singapore (e-mail: wenkail@u.nus.edu).
S. K. (Prof) Ong and A. Y. C. Nee (Prof) are with the Mechanical Engineering Department, National University of Singapore (e-mail: mpeongsk@nus.edu.sg, mpeneeyc@nus).

tracking of the camera pose is provided. Gammeter et al. [7] emphasized on the handling of a large database, whereas a simple 2D tracking is deployed for on-device tracking.

Jaewon et al. [9] equipped their system with recognition and 3D tracking. However, their system requires user input to trigger a tracking process. Both works only provide tracking solutions without the consideration of other important factors in a mobile AR system, such as content communication and representation.

Although there are some research works discussing scalable AR frameworks based on server-client data communication, there are few works on the research of combining image retrieval and image-based tracking. This paper demonstrates the feasibility of combining these two tasks in a single processing pipeline and presents an on-device 3D tracking and rendering framework that can achieve real-time and automatic virtual content augmentation. A scalable AR framework based on a server-client communication architecture is proposed. The search for tracking targets is realized by using the Vector of Locally Aggregated Descriptors (VLAD) descriptor generation [10] and the fast approximate nearest neighbors search in high-dimension [11], the tracking targets are fronto-parallel planar images based on a novel approach from the authors' previous works [12], [13]. Automatic streaming of content is achieved by continuously monitoring the system status. The recognition ability of the proposed framework is demonstrated to be robust and efficient based on the experiments conducted. The functionality of the framework in practical applications is demonstrated in a campus walk-around, with a pre-prepared database of planar surfaces collected around the campus and assigned virtual contents.

The main contributions of the paper are (1) enhanced recognition accuracy with VLAD, (2) integrating sensor feature matching and intensity-based optimization to improve tracking, and (3) real-time and automatic content streaming using key-frame auto-selection, client working phase monitor and AR content delivering.

The remaining of the paper is organized as follows. Section II discusses the proposed scalable AR framework. Section III describes the implementation of the server-side recognition approach, followed by the client-side tracking in Section IV. In Section V, the methods for detecting valid key-frames and monitoring state switching are discussed. Section VI gives a description of content management and delivery. Experimental results and application demonstration and evaluations are presented in Section VII. The conclusion of the work is given in Section VIII.

II. Proposed Scalable AR Framework

Building on top of the image search methods and the state-of-the-art 3D tracking algorithms, an optimized scalable mobile AR framework with automatic virtual content streaming is proposed in this research. Fig. 1 shows the structure of the proposed system. To enable the mobility of the system, the proposed framework is implemented on a smartphone with part of the computation work in the pipeline performed on a remote server. The communication between the server and the client involves key-frame upload and trackable target and content stream-in. Image search is performed on the server taking advantage of the high computational power of the server workstations, while tracking is performed locally on the mobile device to ensure real-time 3D tracking and rendering.

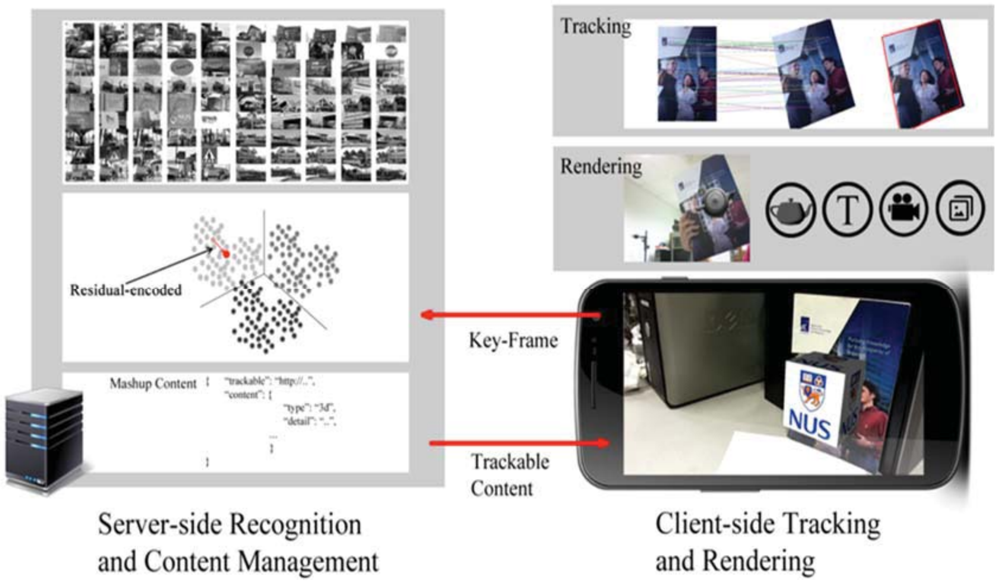


Fig. 1 Server-client framework for large-scale AR applications

Fig. 2 illustrates the roles of the different computer vision algorithms in the processing pipeline. Different levels of abstraction of visual information extracted from a single image perform different roles in the recognition, pose matching, and tracking processes. Three processing stages are performed to register and render the virtual information to an obtained camera frame correctly. Starting with the recognition process, the template image for tracking is selected from a large database. The feature points are sampled from the images and assigned with descriptors. The descriptors are further abstracted by using a feature quantization approach, e.g., the Bag-of-Words (BoW) and the residual-based feature approaches, and compared against a large database. In the detection stage, feature points are used directly with spatial information and matched against those in the template image. In the tracking stage, the pixel intensities instead of feature descriptors are used for a more accurate estimation. This can be explained as a coarse-to-fine matching process. At the start of the pipeline, the visual information is encoded and compressed so that this information can be compared against a large database efficiently. At the end of the pipeline, the pixel intensity and the raw information of the image are used directly for accurate 3D registration.

Without the accessibility to a wide range and meaningful virtual contents, the AR framework will have little significance for practical applications. Therefore, content delivery and rendering are important elements of an AR framework. There are some published works on the approaches for content management and delivery for AR frameworks based on the server-client structure. Langlotz et al. [14] introduced the Augmented Reality Markup Language (ARML) and MacIntyre et al. [15] introduced the KARML, which encloses the geographical information. Both approaches enclose the tracking information and rendering information on a single formatted file with pre-defined standards. In the proposed framework, a rendering engine is included in the system, which supports the rendering of 3D contents, text, video and images. Content communication between the server and client is enclosed in the JavaScript Object Notation (JSON) format.

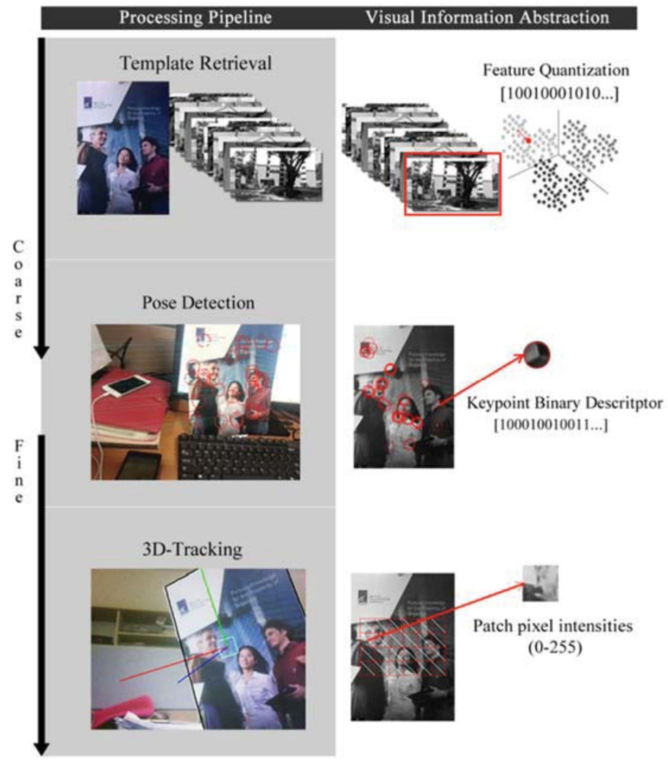


Fig. 2 Levels of Visual Information Abstraction in the Processing Pipeline

III. Server-Side Recognition

Feature quantization is a process that abstracts the visual information from the input image and represents its identifiable characteristics against a database. As a large database is to be dealt with, the feature quantization method to be selected is critical to the performance of the system. BoW [5] is a widely adopted approach to solve the image retrieval problem. The vocabulary tree structured BoW algorithm has been demonstrated to be practical in some scalable mobile AR systems [7], [9]. The vocabulary tree is built by using hierarchical k-means clustering. The tree is built up level by level up to a maximum number of L levels with K branches in a level. To quantize an input image, a histogram is generated in such a way that each feature descriptor extracted from the image is propagated down the vocabulary tree by comparing the descriptor vector to the K candidate cluster centers at each level and following the path of the closest one. The distance between their histograms quantizes the similarity between the images.

VLAD has been reported recently [10], and it uses a codebook generated using k-means clustering on the training dataset to quantize the image. Instead of calculating the histogram distribution of the feature descriptors in the codebook, the residuals to each cluster centers are accumulated. This approach results in a short descriptor for image representation. Evaluation on the performance of the algorithms has demonstrated that the VLAD approach out-performs the BoW approach for most of the test datasets. In this research, image descriptors are generated by using VLAD, which is used to quantize images for fast and accurate retrieval results. The feature descriptors used here are the Scale-invariant Feature Transform (SIFT) descriptors [16] as their effectiveness in visual recognition has been well tested. Feature matching in high-dimension is solved using the tool proposed in [10].

IV. Client-Side Tracking

For on-device tracking, the scheme is developed as follows. The sensor-aided feature matching approach is performed in the detection and re-detection steps when tracking is lost, while the intensity-based image registration method serves its role in continuous tracking of the translation and rotation of the camera. Feature-based tracking has its advantages as it is able to detect the tracking target with any initialization using a robust estimation algorithm, such as RANSAC [17]. With an initial estimation of the pose from keypoint detection and matching, 3D tracking can be achieved through image alignment algorithms. The image alignment process moves or transforms a template image to minimize the difference between the template and an input image. The detection-and-matching process described is able to provide a solution for 3D tracking.

A scale and rotation aware descriptor based on the combination of an efficient binary descriptor BRIEF [18] with sensor measurement from the handheld device is used in the proposed framework. The BRIEF descriptor is a bit vector created from the binary test responses of pixel pairs around certain feature points. Experiments in [18] have shown that an isotropic Gaussian distribution around the patch center gives the best match result. The BRIEF descriptor is robust to illumination changes and some degrees of affine transformation. However, it is not scale and rotational invariant. A sensor-aided feature algorithm [12], which integrates the sensor measurement with the BRIEF descriptor, is proposed and implemented to generate gravity-aligned descriptors in this research. The proposed feature detection and matching algorithm is able to deal with fronto-parallel planar surfaces which are common in the physical environment.

Intensity-based tracking can be achieved using an iterative optimization method. With a suitable parameterization of a small motion about the current camera pose, an optimization method can converge iteratively to the camera pose that gives the minimal image error between the reference and warped images. For tracking of planar surfaces, homography is used to model the perspective transformations due to camera motion. An example of a popular solver is the Gauss-Newton and Levenberg-Marquardt techniques. Efficient Second-order Minimization (ESM) is a more recent method that it has been observed to have a higher convergence rate [20]. In this research, the ESM solver is adopted and applied with a simple illumination model [11].

A. Sensor-Aided Detection

The BRIEF descriptor is constructed from a set of binary intensity tests on a smoothed image patch. The output of a single test is either 1 or 0 depending on the comparison result from the pair of test pixels. Therefore, the output is a string of bits. This bit string is used for matching based on a nearest-neighbor search with a metric of Hamming distance. The binary test is formulated as (1):

$$\boldsymbol{f}_{n,\theta}(\mathbf{p}) \coloneqq \sum_{1\le i\le n} 2^{i-1}\boldsymbol{\tau}(\boldsymbol{p};\ [\mathbf{G}_{\boldsymbol{\theta}}]_{2i-1}\,,\,[\mathbf{G}_{\boldsymbol{\theta}}]_{2i}) \qquad (1)$$

To encode the orientation information in the descriptor, the BRIEF test is steered to the orientation of the camera detected by the inertial sensors. As shown in Fig. 3, the lines in the pattern illustrate the test pair which two ends indicate the positions of the two test pixels and the patches are rotated together with the device. Let G represent the distribution of the test pixels, $G_\theta = R_\theta G$ is considered as the rotated test. Thus, the descriptor can be formulated as (2):

$$\boldsymbol{f}_{n,\theta}(\mathbf{p}) \coloneqq \sum_{1\le i\le n} 2^{i-1}\boldsymbol{\tau}(\boldsymbol{p};\ [\mathbf{G}_{\boldsymbol{\theta}}]_{2i-1}\,,\,[\mathbf{G}_{\boldsymbol{\theta}}]_{2i}) \qquad (2)$$

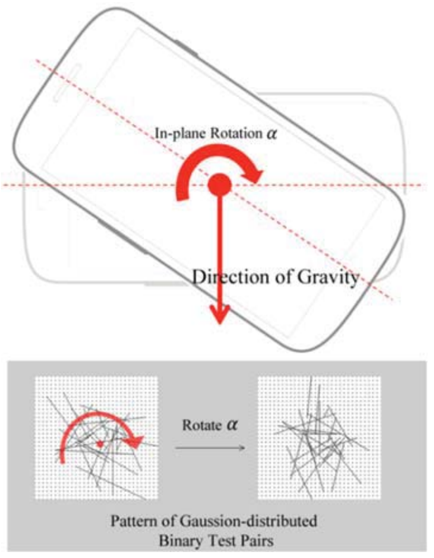

In-plane Rotation α
Direction of Gravity
Rotate α
Pattern of Gaussion-distributed Binary Test Pairs

Fig. 3 Gravity-aligned Test Pattern

In the implementation in this research, 256 test pairs are selected and applied on a 31 × 31 image patch. This steered BRIEF feature is demonstrated to be effective in feature detection and matching based on experiments and tests in the authors' previous work [13]. Using a FAST [19] keypoint detector, 300 points are detected for descriptor-matching. This choice of 300 keypoints ensures the system's efficiency without much sacrifice of the accuracy. The selection of the number of keypoints is discussed in Section VII (B).

B. 3D Tracking

The intensity-based method performs the optimization by minimizing the pixel intensity errors with the assumption that the intensity of the image remains constant in different situations. Since the lighting condition in the environment can strongly affect the illumination of the object being tracked, an illumination model is proposed.

The 3D tracking problem is modeled as follows. Let $\boldsymbol{I}^*$ denote the template image, and I is the frame image that needs to be warped with a transformation to the template. The pixel p_i from the frame image can be warped to the pixels p_i^* in the template image by a warping function $\mathcal{W}$. If the points in the template image form a planar surface, a homography transformation $H(\boldsymbol{x})$ can be used to represent the warping. The homography transformation contains eight parameters denoted by a vector x. For a set of n pixels $\boldsymbol{p}^*$ from a selected region of the template image, a vector $\boldsymbol{y}(\boldsymbol{x})$ of the image different from the template image and the warped frame image can be defined as (3):

$$\boldsymbol{y}(\boldsymbol{x}) = [y_1(\boldsymbol{x}) \;\; y_2(\boldsymbol{x}) \;\; \dots \;\; y_n(\boldsymbol{x})]^{\mathrm{T}} \tag{3}$$

where

$$y_i(\boldsymbol{x}) = \boldsymbol{I}\big(\mathcal{W}\langle\widehat{H}\rangle \circ \mathcal{W}\langle H(\boldsymbol{x})\rangle(p_i^*)\big) - \boldsymbol{I}^*(p_i^*) \tag{4}$$

In this formulation, the warping transformation is defined as an initial estimation $\mathcal{W}\langle H(\boldsymbol{x})\rangle$ composited with an incremental warping $\mathcal{W}\langle\widehat{H}\rangle$.

Given the initial estimated homography $H(\boldsymbol{x})$, the template image $\boldsymbol{I}^*$, the frame image $\boldsymbol{I}$, and the set of pixels from the selected region $\boldsymbol{p}^*$, the compositional incremental warping $\widehat{H}$ can be solved iteratively through minimizing the image difference $\boldsymbol{y}(\boldsymbol{x})$. To address this problem, the ESM is developed as a non-linear optimization procedure with a convergence rate similar to the second-order methods, but with an efficiency of the first-order methods through avoiding the repeated computation of the Hessians.

$\boldsymbol{x}_0$ is estimated and used to update the homography $\widehat{H}$ iteratively. Formulated with Lie Algebra, the homography update is obtained using the matrix exponential function as (5). Detailed formulation of ESM can be found in [20].

$$\widehat{H} \longleftarrow \widehat{H} e^{\sum_{k=1}^{8} x_k \boldsymbol{A}_k} \tag{5}$$

For a more efficient implementation, the patches for tracking are selected from the 24 grids that form the center portion of the template image. The top N grids with the highest intensity gradient are selected for tracking and the rest are eliminated. N is set between 5-10 for mobile devices. In Fig. 4, eight patches with the highest intensity gradient used for optimization are highlighted.

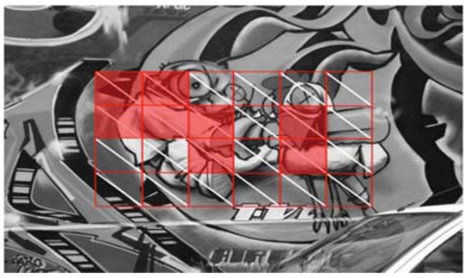


Fig. 4 Patch Grids for Image Alignment, eight patches with the highest intensity gradient are highlighted and used in the ESM optimization

The illumination model used is formulated as follows. Let m_j and d_j respectively be the mean and standard deviation of the pixel intensities in the sub-grid j in the warped image, and m_j^* and d_j^* be the corresponding values for the reference image. The modified pixel intensity $\boldsymbol{I}'(p_{i,j})$ is obtained by using the illumination model as (6):

$$\boldsymbol{I}'(p_{i,j}) = (d_j^*/d_j)\big(\boldsymbol{I}(p_{i,j}) - m_j\big) + m_j^* \tag{6}$$

This approach is feasible because the pose in the previous frame is close to the current one during tracking. It is assumed that the overall illumination of the sub-grid patches would not change too much unless there is a dramatic change of the lighting condition.

V. Automatic Key-Frame Querying and System State Monitoring

To ensure an auto-streaming of content rendering, a method of automatic key-frame querying is used. The simplest way to select key-frames automatically is to schedule a frame-query task at a fix rate, for example, every two seconds. In this case, the current frame in the stream is captured automatically and sent to the server for post-processing every two seconds. The problem with this implementation is that the quality of the frame to be selected cannot be guaranteed; therefore, some useless frames would be sent back which is a waste of the computational power and network bandwidth. The frames with low quality are typically blurred images induced by fast motion, and images with very few visual information or ground/sky images, which often do not contribute to object recognition or tracking as shown in Fig. 5.

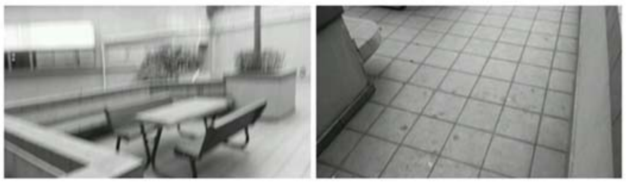


Fig. 5 Examples of invalid key-frames

To address the problem of incorrect orientation, sensor information is used. Since the target for tracking in the application scenario in this research is fronto-parallel planar surfaces, sensor information is used to determine whether the current orientation of the device is in the desired poses. When the angle is within a range of 45-90, the device is considered to be in a good orientation. One observation is that blurred images are poor querying images. The blur effect is mainly due to the fast motion of the device. One of the solutions to this problem is to monitor the sensor measurement and determine the fast-motion mode by calculating the change in the sensor outputs. However, this requires constant monitoring of the sensor measurement over the previous and current frames, and synchronization between the sensor measurement and the frames captured since they are processed in different threads with different frequencies.

In this research, a vision-based key-frame selection method is proposed. FAST feature points are extracted from each frame and descriptors are generated for each keypoint. The keypoints in the current frame are matched with the ones from the previous frame. For each matched pair of keypoints, the descriptor disparity and position displacement are compared. The thresholds are set such that if the descriptor similarity is within a threshold $t_d = 20$, and the displacement is within a threshold $t_p = 8$, the keypoint is considered to be matched. The total number of matched keypoints is a good indicator of the pace of movement of the camera, since when the camera moves too fast, most of the keypoints will not be matched. Two status of the camera movement are defined, namely, “in-focus” when the camera moves slowly tracking the target items and “out-of-focus” when the camera moves quickly. To handle the oscillating between the two statuses caused by small changes of the number of matched keypoints, two threshold values are used. When the matched feature number is greater than the lower-threshold, the device is considered to have entered the out-of-focus status, and when it is less than the higher-threshold, the device is considered to have entered the in-focus status. This approach stabilizes the monitoring process. Combining the criteria of orientation measurement and status monitoring, the device is considered to be ready to acquire key-frame only when it is in a good orientation and at the in-focus status. The automatic key-frame query scheme is modified by scheduling a frame-query task every two seconds, and querying the frame after checking the readiness of the device. If it is ready, the frame is captured for further processing, otherwise the frame is discarded.

The tracking system on the client side is monitored in three states; namely, searching, listening and tracking. Fig. 6 depicts the state transition. In the searching state, the system only executes the keypoint detector and filtering algorithm to select querying key-frames. After a key-frame has been sent for processing, it enters the listening state without any frame processing, and wait for a response from the server-side recognition. The recognition result is sent back within seconds; based on this result, the system will decide whether to return to the searching state or enter the tracking state. If negative matching is signaled, the system state goes back to the searching state, otherwise it proceeds to the on-device 3D tracking state. When tracking starts, the system first fetches the tracking and rendering information with a URL access provided by the searching result. Next, the 3D tracking algorithms are executed and the virtual information is rendered with the tracking results. It is possible for tracking to be lost for some time; however, as long as the user’s focus is still on the same target, tracking could be re-initialized with feature detection and matching. Detecting out-of-focus on a tracking target is enabled by a timer to monitor the tracking performance. If the tracking has been lost for a significant amount of time, it is assumed that the user has shifted his interest and the system returns to the searching state automatically.

VI. Content Management and Streaming

The proposed framework supports various contents to be rendered. The information communicated between the server and client is enclosed in a JSON file. Fig. 7 shows a sample code snippet of the JSON file. The tracking target is sent to the client as a scaled image which matches the key-frame that has been used for querying. The content to be rendered is specified with its type and source used for retrieving. Image, text, videos and 3D models are tagged with their pre-defined type tags. Since the downloading and importing of some large 3D models can cause a significant amount of overhead, an indication of successful tracking is first rendered to the user and the information is updated with the completion of model downloading.

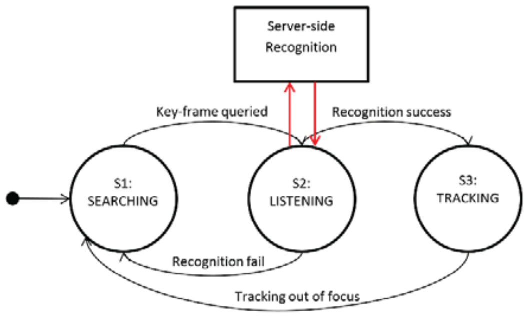
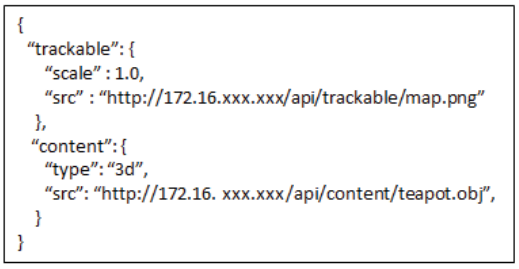


Fig. 6 State transitions of the automatic tracking system



```
{
  “trackable”: {
    “scale” : 1.0,
    “src” : “http://172.16.xxx.xxx/api/trackable/map.png”
  },
  “content”: {
    “type”: “3d”,
    “src”: “http://172.16. xxx.xxx/api/content/teapot.obj”,
  }
}
```

Fig. 7 Sample code snippet of the content to be delivered

VII. Experiments

For real-time tests, the server is configured with a 3.2 GHz quad core CPU and 16 GB of RAM in a 64 bit Linux System. The recognition system is built using the VLFeat [21] library. The client device is a Samsung mobile device with a quad-core 1.5 GHz CPU, 1 GB RAM and built-in accelerometers and gyroscopes. The tracking algorithms implemented in C++ and OpenCV [22] is used to assist in the basic image manipulation and OpenGL ES for 2D/3D information rendering.

A. Image Retrieval Performance Evaluation

The evaluation of the performance of the image retrieval pipeline is performed on a dataset that combines a sub set from the Stanford Mobile Visual Search (SMVS) dataset [21] and a dataset that is built by the authors from a set of images collected from a campus walk-around. The reasons that SMVS is selected are first it focuses on mobile visual search with query images taken from low- to high-end mobile phones; secondly, it focuses on near-duplicate image search over object recognition with identical objects contained in the database images and the query images. SMVS contains eight different categories of objects, from which three categories, namely CD covers, book covers and museum paintings are used in this experiment. The selected sets of data are collections that include mainly planar texture surfaces which are suitable tracking targets in the 3D tracking pipeline. An additional category is added to the test data which are a set of planar surfaces collected from around the campus, including sign boards, posters, maps, bus-stop stands, building facades, etc. For the categories of CD covers, book covers and museum paintings, four different query images are associated to a database image. The four different query images are from heterogeneous low and high-end camera phones. The campus data set also contains four query images for each reference image in the database. The query images are taken using the same device with changes in viewing angles and introduced noise, such as occlusion, blur and illumination variance. The test data set is summarized in Table I. A few samples of database images and query images are shown in Fig. 8.

TABLE I
SUMMARY OF TEST DATASET

	Reference Images	Query Images
Campus	70	280
Book Cover	100	400
CD Cover	100	400
Painting	100	400

Since a post-verification step follows the retrieval step, it is important to identify the number of top query results before a correct match can be found. Fig. 9 plots the retrieval precision against the number of images retrieved. Precision refers to the fraction (y-axis) of retrieved images that are correct matches of the ground truth. As shown in Fig. 9, the campus dataset is the easiest test set that exhibits the highest retrieval rate. The reason is that the images collected around the campus contain more distinctive visual information than the other three datasets, in which the book covers, CD covers and paintings have some similar visual characteristics among them in the same category. Another observation is that the successful retrieval rate goes above 80% for the CD covers and paintings, and above 95% for the book covers and campus dataset when the n value is increased to 5, which is an acceptable retrieval rate for practical applications.

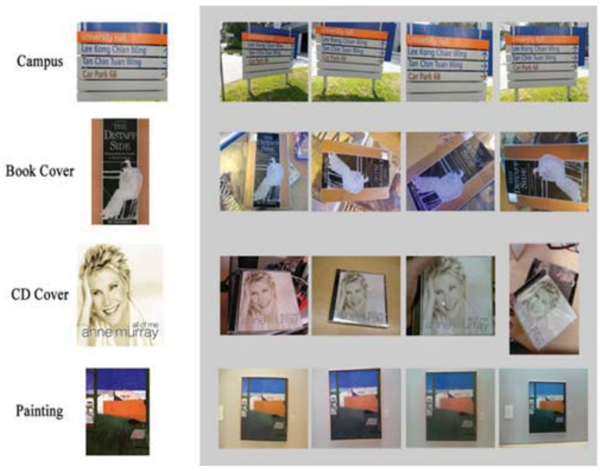


Fig. 8 Sample images from the test dataset

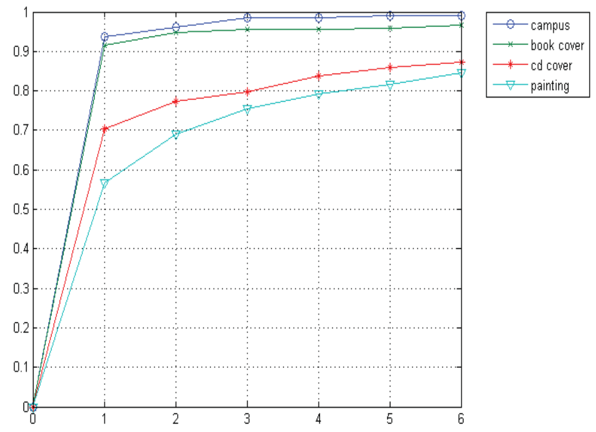


Fig. 9 Curves showing percentage (y-axis) of the ground truth query images that are in the top x percent (x-axis) of search results, plotted for four categories of the dataset

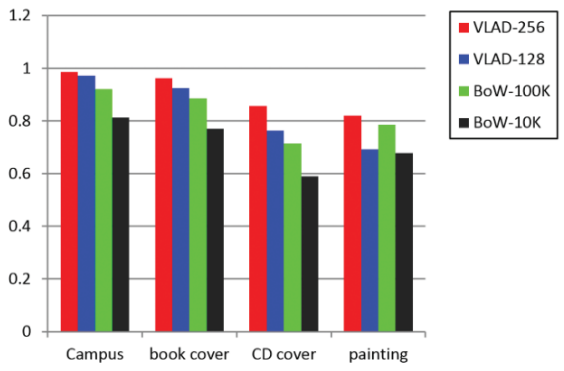


Fig. 10 Comparison of retrieval rate for VLAD and BoW evaluated on the test datasets

As illustrated in Fig. 9, the results based on top five queries provide more than 80% correct matching for all the categories of images. This threshold is selected in the experiment comparing the VLAD and the vocabulary tree that is used in [6], [8]. Fig. 10 shows the retrieval rate for each category of datasets based on a comparison of the two approaches. In the comparison experiment, VLAD with two different parameters, at k-means clusters $k = 256$ and $k = 128$ are used. The number of features from each image is chosen at 500. The vocabulary-tree structured BoW approach is evaluated with the number of leaves at 100k and 10k, and the parameters used are depth $L = 4$ and k-means center $k = 10$, and $L = 5$ and $k = 10$, respectively. As illustrated in Fig. 10, VLAD outperforms BoW for all the test datasets. With a finer clustering of codebook training, i.e. a larger number of k-means cluster centers, a better retrieval rate can be obtained.

B. Experimentation on Keypoint Detection and Matching

The number of keypoints used for matching is a critical parameter for the performance of the detection based on the proposed sensor-aided feature descriptor, since the efficiency of the algorithm depends strongly on the number of features used. In the detection pipeline, three steps are involved, namely feature detection, feature descriptor extraction and feature matching. In this process, matching is effected most strongly by the increase number of keypoints. The main reason is that brute-force search for matching has $O(n^2)$ time complexity. Therefore, matching time exhibits quadratic growth when the number of keypoints increases. To select an optimal number for feature matching, experiments are performed in a subset of sample images from a standard test dataset [22]. It contains four sets of test images, which are designed to capture the effects of viewpoint change or zooming/rotating. These four sets of images shown in Fig. 11 are resized to 640 in width for processing on devices and used to test the efficiency of the keypoint detection and matching algorithm.

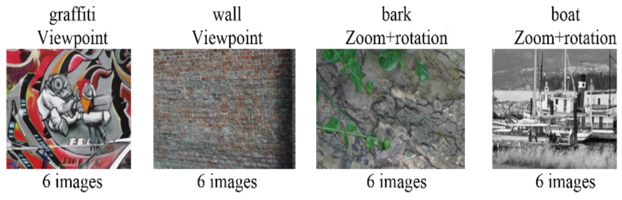


Fig. 11 Test image sets for keypoint detection experimentation

The average matching rate for the test dataset is plotted against the number of keypoints used for matching and shown in Fig. 12. The result of the pose estimation based on RANSAC that is performed on the matched keypoint pairs indicates as success or failure, is recorded for each test set in Fig. 13. It is observed that with increasing number of features used in matching, the processing time increases. Fig. 13 shows that pose estimation has a better performance as the number of keypoints increase. However, the increase in feature numbers does not contribute much to the successful estimation of the pose. Viewpoint change and zoom/rotation have stronger effects on the pose estimation result than the number of keypoints used. For the evaluation and demonstration in the following sections, the number of keypoints used for matching is selected at 300 for each scale.

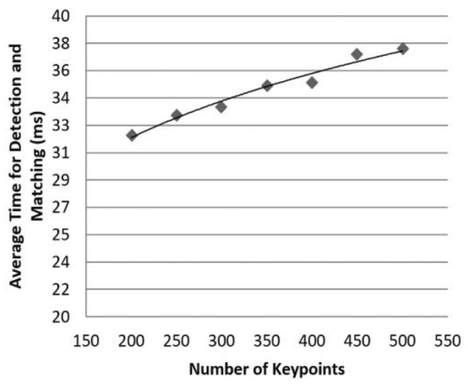


Fig. 12 Average processing time for feature detection, descriptor extraction and matching for four sets of images, plotted for the number of keypoints used in processing

C. 3D Tracking Performance Evaluation

Video sequences are provided to evaluate the performance of the proposed 3D tracking method. Four video sequences taken in an outdoor environment are used. The method to evaluate the robustness of the 3D tracking method is described next. The homography transformation is recorded for each camera frame if tracking is performed successfully. In the post-processing step, each frame is warped with the inverse of the recorded homography and compared with the target image using the normalized cross correlation (NCC). NCC is a good indicator of the similarity of two image patches. For frames that are not tracked, the value is set as 0. Video sequences (a) to (d) are taken during a campus walk-around. Video sequence (a) targets a reflective noticeboard while video sequence (b) targets a poster which contains rich visual features. Video sequences (c) and (d) are building facades that involve fast scaling and occlusion. The videos are taken by using the abovementioned mobile device without any control of the environment lighting; therefore, a practical application scenario is well captured in these videos. The video sequences are processed with the proposed intensity-based tracking method and a feature-based tracking method that is similar to the one in [3], where Wagner et al. proposed the Patch-Tracker with mobile marker-less tracking. Continuous tracking achieved using the Patch-Tracker is based on a search for known features at predicted locations in the input camera frames within a predefined search region and optimization on the re-projection errors. The tracking rate is calculated as the ratio of the successful tracked frames over the total of frames in the test video sequence and the number of successful tracked frames is selected when the NCC value is above a threshold of 0.2. Fig. 14 shows sample frames from each video sequence and the plot of the NCC that is post-calculated for each frame comparing the proposed method and the Patch-Tracker. Fig. 14 depicts the average NCC values for each sequence for the two methods. Fig. 15 shows the plot that compares the final tracking rate for the four video sequences.

From the plot of the NCC over frames in Fig. 14, the smoothness of the tracking can be observed to some extent. For video sequences (a), (c) and (d), ESM-tracking performs better than the Patch-Tracker. This observation is supported by the performance of the two trackers on the second half of video sequence (a), the middle part of video sequence (c) and last quarter of video sequence (d), in which the camera is experiencing rotation, zooming (scaling) and fast translational movement. ESM-tracking manages to provide continuous tracking for most of the time while the Patch-Tracker either experiences too much noise or loses the tracking completely (indicated by 0 NCC value). This conclusion is also supported in Fig. 15 (a), where the averaged NCC value for ESM-tracking is lower than that for the Patch-Tracker, and in Fig. 15 (b) which shows the final tracking rate for each sequence. The Patch-Tracker is comparable with ESM-tracking in terms of the performance for video sequence (b). The main reason is that the tracking target in video sequence (b) contains a relatively large number of distinctive keypoint features suitable for feature-based tracking. However, for the overall tracking rate for sequence (b), ESM-tracking outperforms the Patch-Tracker as indicated by Fig. 15 (b).

		200	250	300	350	400	450	500			200	250	300	350	400	450	500
	1->2	×	×	×	√	√	√	√		1->2	√	√	√	√	√	√	√
	1->3	×	×	×	×	×	×	×		1->3	√	√	√	√	√	√	√
bark	1->4	×	×	×	×	×	√	×	graffiti	1->4	×	×	√	√	√	√	√
	1->5	×	×	×	×	×	×	√		1->5	×	×	×	×	×	×	×
	1->6	×	×	×	×	×	×	×		1->6	×	×	×	×	×	×	×
	1->2	√	√	√	√	√	√	√		1->2	√	√	√	√	√	√	√
	1->3	√	√	√	√	√	√	√		1->3	√	√	√	√	√	√	√
boat	1->4	√	√	√	√	√	√	√	wall	1->4	×	×	×	×	√	√	√
	1->5	√	√	√	√	√	√	√		1->5	×	×	×	×	×	×	×
	1->6	×	×	×	×	×	×	×		1->6	×	×	×	×	×	×	×

Fig. 13 Pose estimation success (√) or fail (×) based on different numbers of keypoints used for matching, evaluated for four sets of test images

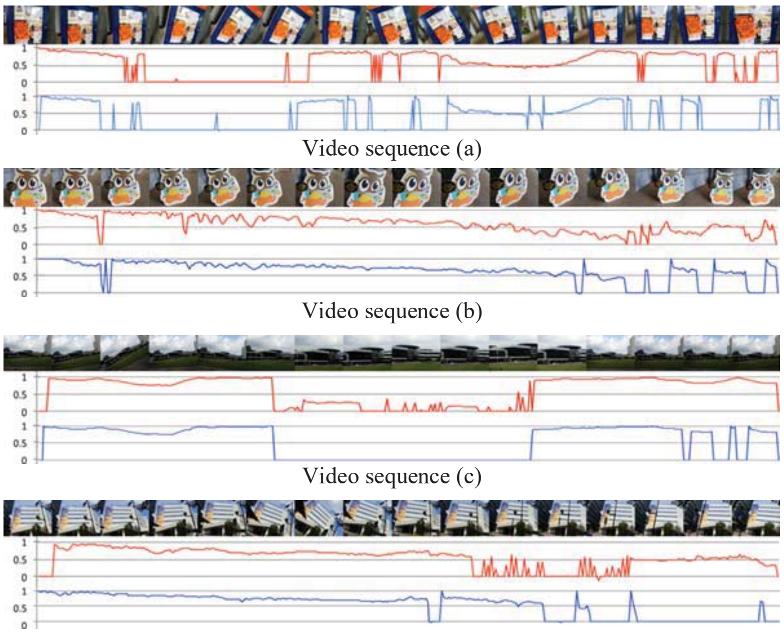
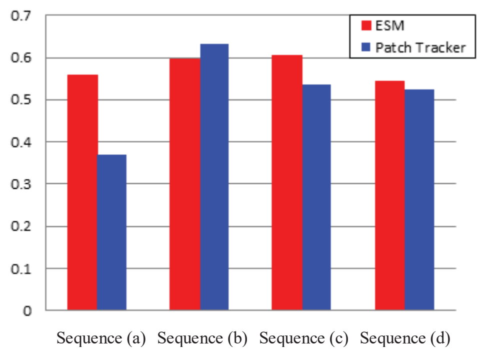
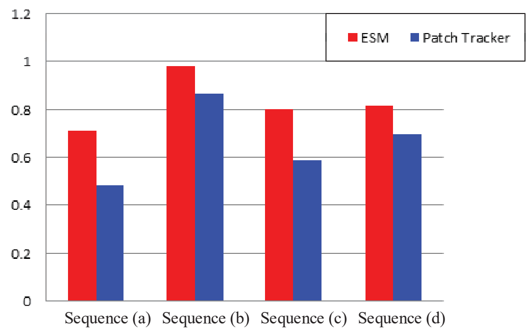


Fig. 14 Sample frames and NCC value plotted over frames. NCC based on ESM-tracking is plotted in red and NCC based on Patch-Tracker is plotted in blue for video sequences (a) to (d)



(a)



(b)

Fig. 15 (a) NCC averaged over frames (y-axis) comparison for ESM tracking and Patch-Tracker based on the test video sequences; (b) Tracking rate (y-axis) compared comparison for ESM tracking and Patch-Tracker based on the test video sequences

D. Key-frame Auto-selection Performance Evaluation

Another two video sequences are used to evaluate the performance of the proposed key-frame auto-selection method (Fig. 16). Video sequence (a) is taken in a laboratory setup environment in which the focus of the camera is switched among three objects. Video sequence (b) is taken in an outdoor environment with random movement of the camera. Fig. 16 shows sample frames from the clips and the feature of matched features are plotted against time. For video sequence (a), the intention of focus change is clear. The algorithm predicts the intention easily as the number of matched features is reduced to zero when the focus of the camera changes. Video sequence (b) exhibits random movement of the camera. It is observed for different scenarios, and the number of matched features remains around 50. Therefore, a single set of threshold values can be used for the system.

E. Application Demonstration and Efficiency Evaluation

To evaluate the performance of the proposed system in a practical scenario, the test samples are collected from a campus walk-around. 70 textured planar surfaces are used for preparing the database on the server-side. The experiment is performed by walking around the campus and monitoring the real-time performance with a monitor system running in the background.

For the real-time performance in a practical scenario, 3D tracking is evaluated based on a set of vertical planar surfaces collected from the campus walk-around. The frame size is selected as 640 × 480. To enable the scale-awareness of the feature tracking approach, a set of descriptors for the referenced image is computed on two scales with a downsizing factor of $\sqrt{2}$. Three hundred keypoints are used for keypoint matching in each scale, which has been determined empirically to be a good trade-off between the efficiency and the recognition performance. The sub-grid ESM tracking algorithm is based on 24 (6 in width and 4 in height) image patches in the center of the reference image. The top five grids with the highest intensity gradient are used for the optimization in order to achieve real-time efficiency. The speed for detection and tracking is summarized in Table II. The number of keypoints to be detected and matched strongly affects the processing time. In practice, the frame rate of the tracking system is maintained at 15 -30 fps most of the time.

The recognition on the server-side consumes up to 200 ms, and an average 50 ms information transfer overhead is observed under the network condition in the authors' campus. In total, it requires around 250 ms for the recognition process. However, the performance depends mainly on the on-device tracking efficiency, since recognition is not performed in every frame.

Fig. 17 shows a few sample frames from the application test. The system supports the rendering of images, text, 3D objects, and video streaming. The information is communicated between the client and server in the format of the pre-defined JSON content. With the key-frame auto-selection algorithm, this system creates a single application experience in which the only interaction required from the user is to hold the camera and explore. The proposed system is able to scale up to hundreds of images recognition and tracking with support from the server-side for image retrieval.

TABLE II
EFFICIENCY EVALUATION OF THE TRACKING SYSTEM

	Time in milliseconds			
	Mean	Standard Deviation	Min	Max
Detection	69.07	33.82	10.35	163.67
Tracking	32.33	18.11	2.68	101.68

VIII. CONCLUSION

In this paper, a scalable mobile AR framework is proposed; the feasibility is demonstrated with experimental tests on datasets collected from a practical situation. The residual-enhanced image descriptors representation adopted on the server-side gives the framework a scalability property. This approach is demonstrated to outperform the BoW method used in some other similar systems. The client-side takes the advantage of a sensor-equipped mobile device and an optimized intensity-based image alignment approach to ensure the accuracy of 3D tracking. The framework handles recognition, tracking, content delivery and rendering in an automatic processing pipeline, allowing the users to be immersed in an AR experience effortlessly. The system is proven to be functional in practical applications.

One of the limitations of the server-client AR framework is the 3D model information transfer. As 3D objects are quite large in bits, the transfer of this content often causes a large overhead on the performance of the system. 3D object streaming is a possible way to overcome the problem. There are a few further developments on the current framework. One of them is an authoring tool which could be integrated with the framework that enables user contribution of content. Automatic content update could be introduced. Besides, GPS information can also be considered to further scale up the tracking capability of the system.

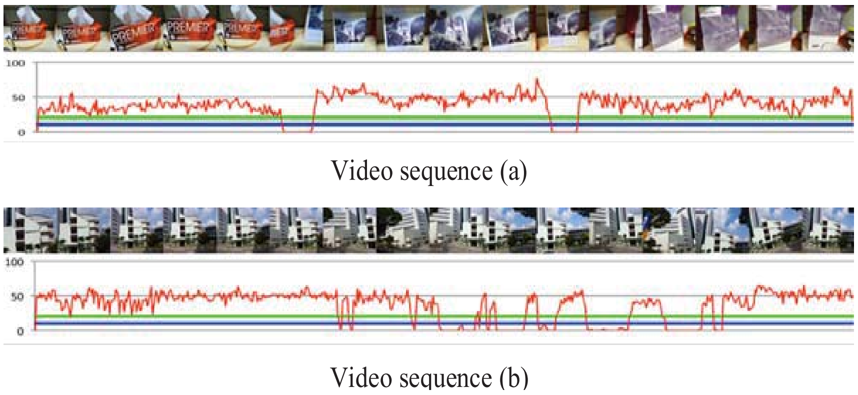


Fig. 16 Number of matched features plotted versus frames for video sequence (a) and (b)

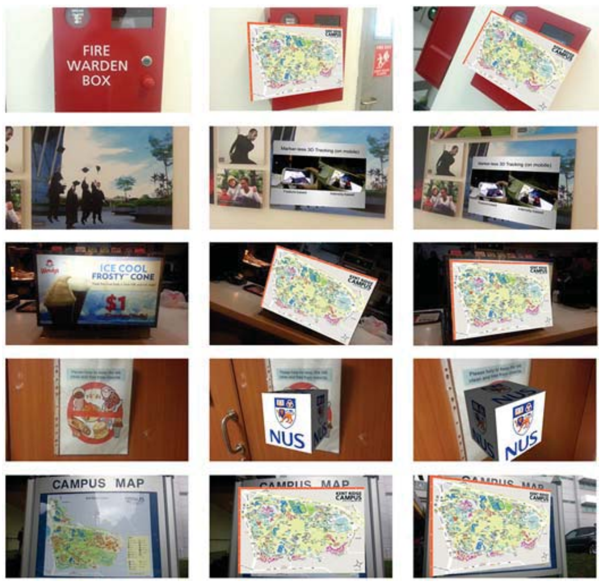


Fig. 17 Sample results from campus walk-around

REFERENCES

[1] S. Feiner, B. Macintype, T. Hollerer and T. Webster "A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment," In Proc. *International Symposium on Wearable Computers*, Cambridge, Massachusetts, 13-14 Oct 1997.

[2] T. Guan, Y. He, L. Duan, J. Gao, J. Yang and J. Yu, "Efficient Bag-of-Features Generation and Compression for On-Device Mobile Visual Location Recognition, " *IEEE MultiMedia*, 21(2):32-41, 2013.

[3] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond and D. Schmalstieg, "Real-time detection and tracking for augmented reality on mobile phone," *IEEE Transaction on Visualization and Computer Graphics* 16(3):355-368, 2010.

[4] D. Schmalstieg, T. Langlotz and M. Billinghurst, "Augmented Reality 2.0. Virtual Realities," *Vienna: Springer-Verlag/Wien*, pp 13-37, 2011.

[5] D. Nister, H. Stewenius, "Scalable Recognition with a Vocabulary Tree," In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, New York, USA, 17-22 June, 2006, pp. 2161-2168.

[6] V. Lepetit, P. Fua, "Keypoint recognition using randomized trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465-1479, 2006.

[7] S. Gammeter, A. Gassmann, L. Bossard, T. Quack and L. Van Gool "Server-side object recognition and client-side object tracking for mobile augmented reality," In *Proc. of 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, San Francisco, CA, 13-18 June, 2010, pp. 1-8.

[8] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli and B. Girod, "Outdoors augmented reality on mobile phone using Loxel-based visual feature organization," In *Proc. of 1st International ACM Conference on*

Multimedia Information Retrieval, Vancouver, BC, Canada, 30-31 August, 2008, pp. 427-434.

[9] H. Jaewon, C. Kyusung, F. A. Rojas and H. S. Yang, "Real-time scalable recognition and tracking based on the server-client model for mobile Augmented Reality," In *Proc. of 2011 IEEE International Symposium on VR Innovation*, Singapore, 19-20 March, 2011, pp. 267-272.

[10] H. Jegou, M. Douze, C. Schmid and P. Perez, "Aggregating local descriptors into a compact image representation," In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, USA, 13-18 June, 2010, pp. 3304-3311.

[11] M. Muja and D. G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration," In *Proc. of International Conference on Computer Vision Theory and Application*, Lisboa, Portugal, 5-8 Feb, 2009, pp. 331-340.

[12] W. T. Fong, L. Yu, S. K. Ong and A. Y. C. Nee, "Marker-less Computer Vision Tracking for Augmented Reality," In *Proc. of 25th Annual Conference on Computer Animation and Social Agents*, Singapore, 9-11 May, 2012, pp. 46-49.

[13] L. Yu, S. K. Ong and A. Y. C. Nee, "Inertial Sensor-aided Feature Detection and Tracking for Outdoor Augmented Reality Applications on Mobile Handheld Devices," In *Proc. of Computer Graphics International*, Hannover, Germany, 11-14 June, 2013, pp. 459-462.

[14] T. Langlotz, S. Mooslechner, S. Zollmann, C. Degendorfer, G. Reitmayr and D. Schmalstieg, "Sketching up the world: in situ authoring for mobile augmented reality." *Personal and Ubiquitous Computing*, 2012, 16(6):623-630.

[15] B. MacIntyre, A. Hill, H. Rouzati, M. Gandy and B. Davidson, "The Argon AR Web Browser and standards-based AR application environment," In *Proc. of 10th IEEE International Symposium on Mixed and Augmented Reality*, Basel, Switzerland, 26-29 Oct, 2011, pp. 65-74.

[16] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004, 60(2):91-110.

[17] M. A. Fischler, R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *1981 Communication of the ACM* 24(6):381-395.

[18] M. V. Calonder, V. Lepetit, C. Strecha and P. Fua, "BRIEF: Binary robust independent elementary features," In *Proc. of 11th European Conference on Computer Vision*, Heraklion, Greece, 5-11 Sep, 2010, pp. 778-792.

[19] E. Rosten, T. Drummond, "Machine learning for high-speed corner detection," In *Proc. of European Conference on Computer Vision*, Graz, Austria, 7-13 May, 2006, pp 430-443.

[20] S. Benhimane, E. Malis, "Homography-based 2D Visual Tracking and Servoing," *International Journal of Robotic Research*, 2007, 26(7):661-676.

[21] V. Chandrasekhar, D. Chen, S. Tsai, N. M. Cheung, H. Chen, G. Takacs, Y. Reznik, R. Vedantham, R. Grzezczuk, J. Bach and B. Girod, "The Stanford mobile visual search dataset," In *Proc. of the second annual ACM conference on Multimedia Systems*, San Jose, CA, 23-25 February, 2011, pp. 117-122.

[22] http://www.robots.ox.ac.uk/~vgg/research/affine. Accessed 9 Jan, 2015.