AUGMENTED REALITY-BASED TOUR GUIDANCE IN STREETS BY COMPUTER VISION TECHNIQUES USING SMART PHONES †

[1]*Song-Tsung Ho* (侯松圳), [2]*Yu-Shiuan Tsai* (蔡宇軒) *and* [3]*Wen-Hsiang Tsai* (蔡文祥)

[1]Institute of Multimedia Engineering
[2]Computer Vision Research Center
[3]Department of Computer Science
National Chiao Tung University, Hsinchu, Taiwan
Emails: c00203@hotmail.com, whtsai@cis.nctu.edu.tw

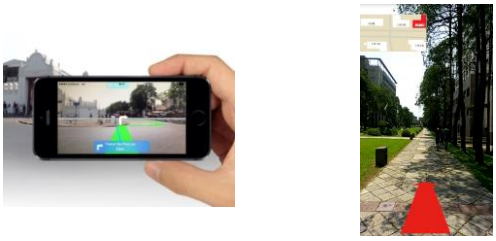
ABSTRACT

An augmented reality (AR)-based tour guidance system for use in streets by computer vision techniques is proposed. The system can guide a user to a goal building and display building information and walking directions in an AR manner on the screen of a user-held smart phone. During system learning, a tour guidance map is generated, which includes a top-view street map and the images and information of the buildings along the street to be visited. A method for building recognition by image matching using speeded up robust features (SURFs) is proposed. A user localization method is proposed then which is based on building recognition around the user and novel applications of the camera calibration techniques used in computer vision. Furthermore, a method for AR-based guidance using a shortest path generated by the Dijkstra algorithm is proposed. Finally, methods for AR-based building-information augmentation and guidance-arrow generation are proposed, which overlays related information on the user-held mobile-device screen to guide the user to walk to the destination. Good experimental results are also presented to show the feasibility of the proposed system for real applications.

Keywords: *augmented reality, tour guidance, mobile device, vanishing points, user localization.*

1. INTRODUCTION

The augmented reality (AR) technique can be used to enhance the real-world image with virtual objects or digital information. It becomes popular recently for uses in many applications because it provides an intuitive way for people to interact with the environment. One application is to develop an AR-based guidance system for touring in streets, which provides a user with the information of surrounding buildings and walking directions on mobile devices in an AR manner. In using such a kind of system during walking, AR-based guidance arrows are displayed on the mobile device continuously, which are more intuitive to follow than a conventional map while trying to reach a goal spot, as illustrated by Fig. 1. In this way, a user can get a complete guidance along a visited street without helps from human guides and maps. The goal of this study is to develop such a type of AR-based guidance system for touring in street areas using a smart phone.



(a) (b)

Fig. 1: Illustrations of AR-based tour guidance systems. (a) An example by MapFan [1]. (b) A snapshot of the screen of a smart phone used in the system proposed in this study.

About AR-related works, Reitmayr et al. [2] presented a model-based tracking method for AR applications in urban environments. In Grosch [3], panoramic images are used in an image-based method for navigation in a real environment. Narzt et al. [4] proposed an AR navigation system which improves the depiction of virtual objects in a real world to assist navigation. Krichenbauer et al. [5] designed an AR-based user interface for creating 3D models for movies and games.

About image-based AR techniques for tour guidance, robust features like SURF [6] are often utilized, as is done in this study. In addition, for AR-based guidance the most importance task is accurate user localization (also called user positioning). For this, we utilize the camera calibration technique proposed by Wang and Tsai [7] in this study. For tour guidance, Katz et al. [8] integrates an adapted geographic information system with different classes of objects useful for improving route selection and guidance. About the used equipment for AR, smart wearable devices are becoming popular, including Google Goggles [9], Kooaba [10], and Amazon Snaptell [11], which are used for identifying products. Furthermore, an application using a head-mounted device (HMD) was proposed by Kato et al. [12], which overlays virtual images on real objects like papers or whiteboards, etc.

To construct the proposed AR-based street-tour guidance system using computer vision techniques, *client-server* architecture is adopted, with a computer used as the server to perform user localization, and the smart phone used as the client that augments guidance and building information on its screen. The system is designed to have the following capabilities. 1) "Learning" street paths and building information organized into a tour

† This work was supported financially by NSC project No. 102-2221-E-009-106-.

[2] Wen-Hsiang Tsai is also with the Dept. of Information Communication, Asia University, Taichung, Taiwan 41354.

guidance map. 2) Detecting and matching building image features (SURFs). 3) Conducting user localization for tour guidance. 4) Augmenting building and guidance information the mobile-device screen. 5) Planning a shortest path to a destination building. 6) Showing step-by-step guidance arrows in realtime to lead the user to the destination.

The remainder of this paper is organized as follows. In Sec. 2, the configuration and processes of the proposed system are introduced. In Sec. 3, the learning process for constructing the tour guidance map is described. In Sec. 4, the proposed method for building recognition and user localization by image matching using SURFs is presented. In Sec. 5, the proposed methods for shortest-path and guidance-arrow generations for AR-based tour guidance are presented. In Sec. 6, some experimental results and discussions are included. Finally, conclusions and some suggestions for future works are given in Sec. 7.

2. SYSTEM DESIGN AND PROCESSES

2.1 System Design

When using the system to visit a street, a user holds a mobile device to receive AR-based guidance in the following way. At first, the user selects a destination building along the street listed in a guidance map as input. Next, he/she takes an image of a nearby building which then recognized by the system as a starting point of the visit. Then, a path from the start point to the destination building is planned by the system and displayed on the mobile device screen to guide the user to reach the destination.

To implement the proposed system, the adopted architecture is shown in Fig. 2, where the client device is a mobile device mentioned above which accesses a server through a wireless network. The reason why a server is used is to reduce the computation load of the client device so that realtime guidance can be accomplished. Through the network, the server receives images from the client device and conducts the works of user localization, path planning, etc., and returns the result to the client.

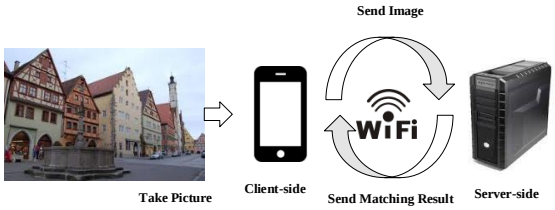


Fig. 2: The network architecture of the proposed system.

2.2 Learning Process

The learning process of the proposed system constructs mainly a *tour guidance map* for use during the tour guidance process. At first, a real-world map of the street area to be visited is drawn and associated with the information of buildings in the area. Also, the path to be followed in the tour guidance is saved as part of the map. Next, we walk on the path and stop in front of each building at a spot along the path at an appropriate distance from the building to start learning of the buildings around. The actions taken in the learning include acquiring a number of surrounding-building images by turning around 360^{o} at the spot and measuring manually the distances to the buildings. The resulting images and distance data are put into the tour guidance map finally.

After this learning step, we continue to walk along the path and repeat the actions in the front of the next along-path building until the path is traversed to its end. Furthermore, the features in each taken image are detected by an SURF extraction algorithm, and used to build a *feature table* organized in the form of a *k-d tree*.

2.3 Tour Guidance Process

In the tour guidance process, the user takes a building image in each visit step and sends it to the server by a mobile device. Then, the server detects the image features by the SURF extraction algorithm, and matches them against the above-mentioned feature table. From the matching result, we conduct user localization, including computation of the user's position and orientation, by *computer vision techniques using vanishing points*. Accordingly, the system augments relevant building information on the mobile-device screen. Or if the user is searching for a destination building, the system plans a path from the user position to the destination and shows the result on the mobile-device screen using step-by-step AR-based guidance arrows. The details will be described in subsequent sections.

3. LEARNING OF TOUR GUIDANCE MAP

3.1 Creation of Real-world Map

In this section, we introduce some methods we propose to construct the tour guidance map. The data associated with this map, as mentioned previously, includes: (1) the information about the location of each spot where we take building images; (2) the images of the buildings along the path; (3) the information of the buildings in the street area to be visited; and (4) the feature table of the building images for recognition.

At first, we get a 2D real-world map like the Google Map (see Fig. 3(a)) of the concerned tour area. Next, we associate related information with the map, which includes, an introduction to the street area, the name of each selected building, and so on. Finally, we save the map with data type ".jpeg" into the tour guidance map.

(a) (b)

Fig. 3: (a) The real-world map of the main part of the NCTU campus downloaded from the Google Map site. (b) The real-world map which is used in the process.

3.2 Creation and Association of Street-image Nodes

In this section, we present the method we propose for building street-image nodes as part of the tour guidance map for use in path planning as an algorithm described in the following.

Algorithm 1. Street-image node establishment.
Input: a selected path.
Output: street-image nodes along the path.

Steps.

1. Stand on the starting spot of a selected path in the street area.
2. Record the position of the spot in the real-world environment into the tour guidance map.
3. Turn around through the range of 360^o at the spot as illustrated by Fig. 4, and take an image every 45^o, resulting in eight images of the surrounding buildings.
4. Set up a street-image node whose data include the position recorded in Step 2 and the images taken in Step 3.
5. Walk to the next position by advancing 4 meters.
6. Repeat Steps 2 to 5 until the end of the selected path is reached.

With the above algorithm carried out, for the example shown in Fig. 3(b), we set up totally 78 street-image nodes and 624 images in the tour guidance map.

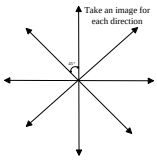


Fig. 4 Illustration of the method which we taken images.

3.3 Calibration between Street-image Nodes and Real-world Map

In this study, the street area for touring is measured in terms of length units in the real world. But the street image we take has its size represented in terms of the unit of pixel. Therefore, a calibration between the length unit used in real-world data and the pixel unit used in the taken image is necessary. As an example, the street area shown in Fig. 3(b) is 190 meters in width and 111 meters in height in the real world, and the *digital* map which we constructed and used by the proposed system is 700 pixels in width and 400 pixels in height. To calibrate the parameters between the real-world distance and the pixel, we conduct the following computation:

$$190_{meter} \div 700_{pixel} = 0.27_{meter/pixel} \quad (1)$$

which means that a pixel in the image is 0.27 meter in length in the real world. Also, the distance between every two path nodes is 4 meters in length as mentioned before, which may be converted into a value in terms of pixels by the following computation:

$$4_{meter} \div 0.27_{meter/pixel} \cong 15_{pixel} \quad (2)$$

which means the distance between two path nodes is 15 pixels in length in the digital map.

4. IMAGE RECOGNITION FOR USER LOCALIZATION

4.1 Building Recognition by SURFs

In the tour guidance process, the server receives images from the client at first. The images taken at the client side are of the resolution of 480 pixels in width, 640 pixels in height; and are transformed into gray-level images for reducing the transfer time to the server side. Because the speed of the image matching process depends on the number of used features, we have to decide the range of the number of features in the image which should be used for the matching process. According to our experimental experience, the more features we extract for matching, the better performance we can obtain, but the slower the resulting speed of matching. So, we decide that the number of features that the server side detects to be in the range of 500 to 1000 by experimental experience.

After extracting features from the acquired image, we match them against the pre-constructed feature table associated with the tour guidance map for the purpose of building recognition. We adopt a *k*-Nearest Neighbor (*k*NN) matching method and set *k* as 2 to accomplish the purpose of building recognition. We call this way of matching *2NN method*. Each feature extracted from the image received from the client side is given a *similarity distance* after it is compared with each of the features in the feature table. The similarity distances then are sorted in an ascending order. And if the smallest distance divided by the second smallest distance is larger than 1.5, we can say that the matching result of this feature is good, or we reject it and check the next feature. If the matching result between a pair of features, one in the input image and the other in the feature table, is good, we draw a line between them, meaning that the two features are similar. This procedure is repeated until all the features in the image received from the client side are processed.

We have conducted many experiments using the 2NN matching method. However, error analysis of the experimental results showed that the method is not good enough to deal with complicated outdoor scenes. An example is shown in Fig. 5, in which each feature-pair matching result is marked by two red circles. The left circle encloses a feature in the left image received from the client which we call the *query image*, and the right circle encloses the corresponding feature in the right image which is in the database, called the *training image*. The matching result of this feature pair is obviously erroneous. To solve this problem, we propose further a *structure matching* method, which takes the similarity between the structures of the features *both* in the query image and in the training image into consideration, not just dealing with the numerical similarity between the two feature sets in the two images.

(a) (b)

Fig. 5. The contrast image of the result after steps of algorithm 4.2(a) The original matching result. (b) The improvement result.

More specifically, in the proposed structure matching method, say, we select a feature matching pair from the matching result, including a *reference feature* r_1 in the query image and a corresponding reference feature

r_1' in the training image. A vector v_1 from r_1 to r_1' can then be defined. In this way, we can also define vectors v_2, v_3, ..., v_n for all the remaining matched feature pairs. All these vectors are regarded to originate from the origin of a coordinate system. Then, we compute the distances d_2, d_3, …, d_n of v_2, v_3, …, v_n with respect to v_1, respectively. If the matching of every feature pair is good, then all the vectors v_1 through v_n, when made to originate from the origin of the coordinate system, will point to the same direction and so overlap perfectly. Therefore, if the distance d_i between v_1 and any vector v_i with $n \geq i \geq 2$ is too large, we can say that v_i is distinct from v_1. Consequently, we may discard v_i because the vector is not similar to v_1. In other words, we discard the pairs which do not satisfy *structural similarity*.

It can be seen from the above process that the selection of the *reference vector* v_1 is crucial. How to select it will be presented in the following algorithm which describes the proposed structure matching method mentioned above.

Algorithm 2. Structure matching for improving matching accuracy.

Input: the feature matching pairs between two images.

Output: the feature matching pairs with more accuracy.

Steps.

1. Match the query image and the training image to get the matching-feature pairs.
2. Find the top five pairs with the smallest distances in all the matching pairs.
3. Average the corresponding vectors of the five pairs to get a reference vector v_1.
4. Compute the distance between vector v_1 and the next vector, and discard the pair if the distance of the two vectors is too large.
5. Repeat the above two steps to confirm that all the other pairs satisfy the structure relationship of the matching result.

4.2 Speeding up Feature Matching

In the previous section, we introduce the SURF-matching algorithm for the purpose of image recognition. When a huge number of image feature points in a database needs to be matched, the speed is usually slow. Specifically, the matching time grows in the order of $O(n^2)$ where n is the number of images in the database In order to solve this problem, we organize all the image features in the database into a k-d tree data structure by which the feature-matching time grows in the order of $O(\log_2 n)$, which is a great improvement. The complete steps implementing this technique are described as an algorithm below.

Algorithm 3. Speeding up feature matching.

Input: the features f_1, f_2, …, f_n of the input image I and a k-d tree T of all the image features of the database.

Output: the matching result of each f_i in I, $i = 1 \sim n$.

Steps.

1. Traverse the k-d tree T from its root node N_{root} which is assigned initially to be the feature $f_i = f_1$, and if the first axis value of f_1 is larger than the first axis value of T, then go to the right subtree of T; otherwise, go to the left subtree.
2. Repeat the traversing action of Step 1 until a leaf node N_{leaf} at the bottom of tree T is reached.
3. Set the leaf node N_{leaf} temporarily as the best matching result.
4. Check the whole path in T from N_{leaf} to N_{root} recursively, and update the best result to be the node which is the most similar to f_1.
5. Repeat Steps 1 through 4 with root node $N_{\text{root}} = f_{i+1}$ until a matching result is found for each feature of f_1, f_2, …, f_n.
6. Count the number of the features that have been matched successfully for each image in the database, and the image whose features have been matched with the largest frequency is the best result.

A series of experiments of image recognition have been conducted using the matching algorithms presented in the last two sections, and the results are shown in Table 1, from which we can see that using a k-d tree indeed can improve the matching speed up to 150 times faster though the resulting image recognition rate is lowered a little bit from 96.1% down to 94.8%.

Table 1. Results of image recognition rate by using SURFs.

	Number of test cases	Number of correct recognition	Recognition rate	Average matching time (sec)
Using a k-d tree	77	73	94.8%	1.7
Use no k-d tree	77	74	96.1%	256

4.3 Derivation of User's Position and Orientation Parameters by a Calibration Object

It is known that the *parallel edges* on a target object like the rectangular parallelepiped shown in Fig. 6 will appear to be *vanishing lines* in an image of the object, and that each pair of vanishing lines will extend to intersect at a *vanishing point*. The camera calibration method proposed by Wang and Tsai [7] uses the properties of such vanishing points to compute the camera's position and orientation in a world coordinate system. The method was found in this study to be applicable subtly to solve the user localization problem, including computations of the user's position and orientation.

In detail, let $P_0 \sim P_7$ be the eight vertices of a target object, a rectangular parallelepiped, shown in Fig. 6. Define two coordinate systems for use here, one being a world coordinate system (WCS) in which the object is located, and the other a camera coordinate system (CCS) built on a camera used to take images of the object. The origin of the WCS is defined to be vertex P_3 of the object with the X-axis going through edge $\overline{P_2P_3}$, the Y-axis through edge $\overline{P_3P_4}$, and the Z-axis through edge $\overline{P_3P_6}$. The world coordinates of the eight vertices of the object are known in advance by manual measurement.

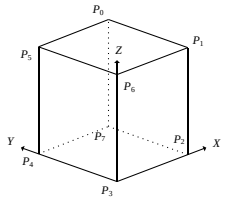


Fig. 6 The calibration target used in the calibration method adopted in this study for user localization.

The CCS is shown in Fig. 7 with the camera's lens center L as the origin. The V-axis of the CCS is the optical axis of the camera and the U-W plane is parallel to the image plane located at $V = f$, with f being the camera focal length. The U'-axis and the W'-axis, which are parallel to the U-axis and the W-axis, respectively, define the coordinates of any point in the image plane.

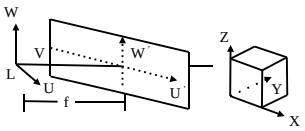


Fig. 7 Two coordinate systems: camera coordinate system UVW and world coordinate system XYZ.

We now define the camera parameters with respect to the WCS. Suppose that the camera lens center L is located at world coordinates (x_c, y_c, z_c), and the pan, tilt, and swing angles of the camera are θ, φ, ψ, respectively. Based on these parameters, two matrices, one for translation and the other for rotation, used in the world-to-camera coordinate transformation can be defined in the following way:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ x_c & y_c & z_c & 1 \end{bmatrix};$$

$$M = \begin{bmatrix} \cos\theta\cos\psi + \sin\theta\sin\phi\sin\psi & -\sin\theta\cos\phi & \sin\theta\sin\phi\cos\psi - \cos\theta\sin\psi & 0 \\ \sin\theta\cos\psi - \cos\theta\sin\phi\sin\psi & \cos\theta\cos\phi & -\cos\theta\sin\phi\cos\psi - \sin\theta\sin\psi & 0 \\ \cos\phi\sin\psi & \sin\phi & \cos\phi\cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (3)$$

For brevity of representation, matrix M is denoted alternatively as

$$M = \begin{bmatrix} A & D & G & 0 \\ B & E & H & 0 \\ C & F & I & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

where A through I are those in the corresponding entries of the matrix M in Eq. (3). Now, the coordinate transformation between the two coordinate systems, from the world coordinates (x, y, z) to the camera coordinates (v, u, w), can be written simply as

$$(v,\ u,\ w,\ 1) = (x,\ y,\ z,\ 1)\cdot T^{-1}\cdot M = \begin{pmatrix} A(x-x_c)+B(y-y_c)+C(z-z_c), \\ D(x-x_c)+E(y-y_c)+F(z-z_c), \\ G(x-x_c)+H(y-y_c)+I(z-z_c) \end{pmatrix}. \quad (5)$$

That is, we have

$$\begin{aligned} u &= D(x-x_c)+E(y-y_c)+F(z-z_c), \\ v &= A(x-x_c)+B(y-y_c)+C(z-z_c), \\ w &= G(x-x_c)+H(y-y_c)+I(z-z_c). \end{aligned} \quad (6)$$

So for any point p at coordinates (x, y, z) in the WCS, according to the imaging geometry of the pinhole camera model, the coordinates (u', w') of its corresponding projection point p' in the image can be computed as:

$$(u', w') = \left(\frac{f \cdot u}{v}, \frac{f \cdot w}{v} \right). \quad (7)$$

And the vanishing point $V_x = (u_x', w_x')$ in the x-direction can be computed according to Eqs. (6) and (7) with x approaching infinity in the following:

$$(u_x', w_x') = \begin{pmatrix} \lim\limits_{x\to\infty} f\dfrac{D(x-x_c)+E(y-y_c)+F(z-z_c)}{A(x-x_c)+B(y-y_c)+C(z-z_c)}, \\ \lim\limits_{x\to\infty} f\dfrac{G(x-x_c)+H(y-y_c)+I(z-z_c)}{A(x-x_c)+B(y-y_c)+C(z-z_c)} \end{pmatrix}^T = \left(f\frac{D}{A}, f\frac{G}{A} \right). \quad (8)$$

From matrix M defined in Eq. (3), the above equation can be reduced to be

$$(u_x', w_x') = \left(f\frac{-\sin\theta\cos\phi}{\cos\theta\cos\psi + \sin\theta\sin\phi\sin\psi}, f\frac{\sin\theta\sin\phi\cos\psi - \cos\theta\sin\psi}{\cos\theta\cos\psi + \sin\theta\sin\phi\sin\psi} \right). \quad (9)$$

Similarly, by Eqs. (6) and (7) with y approaching infinity, the vanishing point $V_y = (u_y', w_y')$ in the y-direction can be derived to be

$$(u_y', w_y') = \left(f\frac{\cos\theta\cos\phi}{\sin\theta\cos\psi - \cos\theta\sin\phi\sin\psi}, f\frac{-\cos\theta\sin\phi\cos\psi - \sin\theta\sin\psi}{\sin\theta\cos\psi - \cos\theta\sin\phi\sin\psi} \right). \quad (10)$$

And the coordinates of the vanishing point $V_z = (u_z', v_z')$ in the z-direction can be computed similarly to be

$$(u_z', w_z') = \left(f\frac{\sin\phi}{\cos\phi\sin\psi}, f\frac{\cos\phi\cos\psi}{\cos\phi\sin\psi} \right). \quad (11)$$

The equations derived above are for use in 3D applications, but for our study here we are dealing with 2-D cases because we may assume that the user takes images in a normal posture such that the camera held by the user is *vertical to ground* and *facing the target*. Accordingly, we can set both the tilt and swing angles φ and ψ of the camera to be 0. In accordance, the equations for computing the vanishing point $V_y = (u_y', w_y')$ in the y-direction derived previously in Eqs. (10) can be simplified to be

$$(u_y', w_y') = = \left(f\frac{\cos\theta}{\sin\theta},\ 0 \right). \quad (12)$$

By using Eq. (12) above, we can derive the pan angle θ, which may be considered as the user's orientation, to be

$$\theta = \operatorname{atan}\left(\frac{f}{u_y'}\right). \quad (13)$$

In addition, we nay regard the user's position as that of the camera lens center with world coordinates (x_c, y_c, z_c). Suppose that P_1 at coordinates (x_1, y_1, z_1) and P_2 at coordinates (x_2, y_2, z_2) are any two known points in the WCS and that their corresponding projection points in the image plane are P_1' at coordinates (u_1', w_1') and P_2' = (u_2', w_2'), respectively. By Eq. (8) we have

$$u_1' = f\frac{D(x_1-x_c)+E(y_1-y_c)+F(z_1-z_c)}{A(x_1-x_c)+B(y_1-y_c)+C(z_1-z_c)}; \quad (14)$$

$$w_1' = f\frac{G(x_1-x_c)+H(y_1-y_c)+I(z_1-z_c)}{A(x_1-x_c)+B(y_1-y_c)+C(z_1-z_c)}; \quad (15)$$

$$u_2' = f\frac{D(x_2-x_c)+E(y_2-y_c)+F(z_2-z_c)}{A(x_2-x_C)+B(y_2-y_c)+C(z_2-z_c)}; \quad (16)$$

$$w_2' = f\frac{G(x_2-x_c)+H(y_2-y_c)+I(z_2-z_c)}{A(x_2-x_C)+B(y_2-y_c)+C(z_2-z_c)}. \quad (17)$$

Any three of these four equations can be selected to derive analytic solutions for the camera position parameters x_c, y_c, and z_c. Here we take Eqs. (14), (15), and (16). The three equations can be transformed into

$$\begin{aligned} a_1x_c + b_1y_c + c_1z_c &= d_1; \\ a_2x_c + b_2y_c + c_2z_c &= d_2; \\ a_3x_c + b_3y_c + c_3z_c &= d_3 \end{aligned} \quad (18)$$

where

$$\begin{aligned} &a_1 = fD - u_1'A; && a_2 = fG - w_1'A; && a_3 = fD - u_2'A; \\ &b_1 = fE - u_1'B; && b_2 = fH - w_1'B; && b_3 = fE - u_2'B; \\ &c_1 = fF - u_1'C; && c_2 = fI - w_1'C; && c_3 = fF - u_2'C; \\ &d_1 = a_1x_1 + b_1y_1 + c_1z_1; && d_2 = a_2x_1 + b_2y_1 + c_2z_1; && d_3 = a_3x_2 + b_3y_2 + c_3z_2. \end{aligned}$$

Then, solving the three simultaneous linear equations in Eqs. (18), we get unique solutions for x_c, y_c, and z_c as:

$$x_c = \frac{\begin{vmatrix} d_1 & b_1 & c_1 \\ d_2 & b_2 & c_2 \\ d_3 & b_3 & c_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}}; y_c = \frac{\begin{vmatrix} a_1 & d_1 & c_1 \\ a_2 & d_2 & c_2 \\ a_3 & d_3 & c_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}}; z_c = \frac{\begin{vmatrix} a_1 & b_1 & d_1 \\ a_2 & b_2 & d_2 \\ a_3 & b_3 & d_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}}. \quad (19)$$

4.4 Derivation of User Position and Orientation

The above derivations of user orientation and position θ and (x_c, y_c, z_c) are based on the use of a calibration object which is a rectangular parallelepiped. It is found in this study that instead of using inconveniently a target object in the guidance process for user localization, the image matching result can be utilized as a substitute for the target object. Actually, the matching result can be processed further to find appropriate vanishing points on which the user localization process described in the last section can be applied.

Specifically, after matching the query image I_q with the training image I_t using SURFs, we discard the pairs which are not similar, and get two feature-point sets F_q and F_t, where each feature point in F_q has a corresponding point in F_t. Then, we choose *randomly* four pairs of corresponding feature points from F_q and F_t and use them to find a *projection* relationship from I_q to I_t by a homographic transformation defined by the following equation:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x_3' & -y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y_3' & -y_3y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x_4' & -y_4x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y_4' & -y_4y_4' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{bmatrix}. \quad (20)$$

where (x_i, y_i), i = 1~4 and (x_i', y_i'), i = 1~4 are the coordinates of the four chosen points p_i in F_q and those of the corresponding four chosen points p_i' in F_t; and h_{ij} are the elements of the homography matrix H specified as follows:

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}. \quad (21)$$

Eqs. (20) describes a linear system which may be solved to obtain the elements h_{ij} in matrix H. However, H so computed might not be good enough because the the four matching-point pairs (p_i, p_i') were chosen randomly. Therefore, we check the goodness of the computed H by the following way: (1) map the coordinates (x_j, y_j) of each feature point p_j in F_q by H in the following way:

$$\begin{bmatrix} w \cdot x_j'' \\ w \cdot y_j'' \\ w \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix} \quad (22)$$

to get a pair of coordinates (x_j'', y_j''); (2) compute the distance d_j between the computed coordinates (x_j'', y_j'') and the coordinates (x_j', y_j') of the feature point p_j' in F_t corresponding to p_j; (3) if d is smaller than a pre-selected threshold t, then we call this feature-point pair (p_j, p_j') an *inlier*; otherwise, an *outlier*; (4) if the number of inliers is larger than two times that of the outliers, then take the current H as a good homography matrix and exit, otherwise, take another set of 4 feature-point pairs from F_q and F_t again and repeat the above steps until done.

Next, to show the mapping specified by the computed homography matrix H clearly for visual inspection, we apply H to find the projection of the four corners of the query image I_q onto the training image I_t, and draw the result as a red or green quadrilateral like the three examples shown in Fig. 8. From the drawn red quadrilateral overlaid on the training image in each example, we can see that the four corner points of the query image appear respectively at the correct corresponding points in the training image. Furthermore, the upper and lower sides of the red quadrilateral, though appearing in the training image, actually are exactly the vanishing lines of the query image. Therefore, we compute the coordinates (u_y', v_y') of the intersection point of the two vanishing lines for use as a vanishing point which is finally used for user localization as described previously to get the orientation and position of the user as described by Eqs. (13) and (19).

(a) (b) (c)

Fig. 8 Three examples of projection from the query image (left) to the training image (right) drawn as a red or green quadrilateral.

5. AUGMENTED REALITY-BASED TECHNIQUES FOR BUILDING INTRODUCTION AND STREET GUIDANCE

5.1 Augmenting Names and Information of Buildings on User-view Images

In this section, we describe the method we propose to calculate the position of a concerned building appearing on the user's mobile-device screen according to the image matching result, and augment accordingly the corresponding building information at a correct position on the mobile-device screen.

Firstly, the position for building-information augmentation on the mobile-device screen is computed according to the drawn red quadrilateral mentioned previously. The top-left corner of the red quadrilateral is used as the anchor point for displaying the augmented information, as illustrated by the example shown in Fig. 9(a). And if the top-left corner of the red quadrilateral is out of the image, then the anchor point is taken to be just the top-left corner of the mobile-device screen, as

illustrated by the example shown in Fig. 9(b). Afterwards, we augment at the anchor point relevant building information, which includes the building's name, distance, and orientation, as illustrated by the two examples shown in Fig. 9. Of course, other more relevant information may also be shown.

(a) (b)

Fig. 9 Using top-left corner of quadrilateral as anchor point for information augmentation.

5.2 Path Planning and AR-based Guidance

The proposed AR-based street guidance process in concept is based on the use of a path planned by use of the Dijkstra algorithm from the user's current position to a selected destination which is a building along the street. More specifically, at first the user is asked to select a building to visit as the *destination node* N_d, and then to take an image of the nearest building around him/her in the street for user localization. Next, the user' current position yielded by the user localization process is taken as the *start node* N_s, and the Dijkstra algorithm is applied to plan a path from N_s to N_d. Then, AR-based street guidance is started in a *pedestrian dead reckoning* (PDR) fashion. That is, the user is guided to the next node N_i in the planned path and asked to take an image of the nearby building there for the system to conduct user localization to confirm his/her arrival at N_i correctly. The system then updates the user position as the current node and guides the user to the next node N_{i+1} in the planned path. Such a process is repeated until reaching the destination node. Guidance of the user in each step of this process is accomplished by showing a guidance arrow on the use-held mobile-device screen. More details are described in the following algorithm.

Algorithm 4. AR-based guidance by path planning using the Dijkstra algorithm.

Input: a user identified by the proposed system.

Output: guidance of the user to a destination node.

Steps.

1. Ask the user to select a building he/she wants to visit as the *destination node* N_d.
2. Ask the user to take an image of a nearby building for recognition to conduct user localization, and use the resulting user position as the start node N_s.
3. Find a shortest path P by the Dijkstra algorithm from N_s to N_d, and let the nodes in the path be denoted as $\{N_1, N_2, \ldots, N_n\}$ where $N_1 = N_s$ and $N_n = N_d$.
4. Set $i = 1$ and get the initial node $N_i = N_1 = N_s$ in P.
5. Guide the user from node N_i to the next node N_{i+1} in P by drawing an AR-based guidance arrow on the user's mobile-device screen pointing to a direction D_{arrow} computed in the following way:
   5.1) compute the *next-node direction* D_i from N_i to N_{i+1};
   5.2) take to be the reading of the electronic compass built in the user-held mobile device as the current user's direction D_{user}
   5.3) compute the guidance-arrow direction D_{arrow} in terms of D_i and D_{user} according to Algorithm 5 (described in the next section).
6. While the user is walking, update the direction D_{arrow} of the guidance arrow in the following way and display it continuously:
   6.1) update the value of D_{user} by checking the new reading of the electronic compass;
   6.2) compute a new direction D_{arrow}' for the guidance arrow in terms of the data of both the next-node direction D_i and the updated D_{user} according to Algorithm 5 (described in the next section).
7. Ask the user at the just-reached spot (not necessarily N_{i+1}) to take an image of a nearby building again for recognition to conduct user localization, and check the yielded user's position to perform one of the following four cases:
   7.1) if the user has not arrived at node N_{i+1}, go to Step 6;
   7.2) if the user has arrived at node N_{i+1}, then set $i = i + 1$ and go to Step 5;
   7.3) if the user has arrived at the destination node N_n, then go to Step 8;
   7.4) if the user has arrived at a node N_{other} other than N_{i+1} and N_n, then set N_{other} as a new start node N_s, and go to Step 3 to re-plan a new path and start a new guidance session.
8. Show the message "GOAL" as well as an introduction to the destination building on the mobile-device scree.

In the above algorithm, checking if the user has arrived at a certain node N_i is accomplished by checking if the user position is within a certain range of the node.

5.3 Generation of AR-based Guidance Arrow

In Steps 5 and 6 of Algorithm 4, a guidance arrow is generated and overlaid on the street-scene image acquired by the mobile-device camera and shown on the device screen to guide the user to the next node or to the destination in the planned path. The direction D_{arrow} of the arrow is computed is computed according to Algorithm 5 described in this section.

Algorithm 5. Generation of a guidance arrow.

Input: the current node N_i of the planned path P (which is also the user's current location), the next node N_{i+1} of P, and the user's direction D_{user} taken to be the electronic-compass reading value.

Output: a proper guidance arrow drawn on the device.

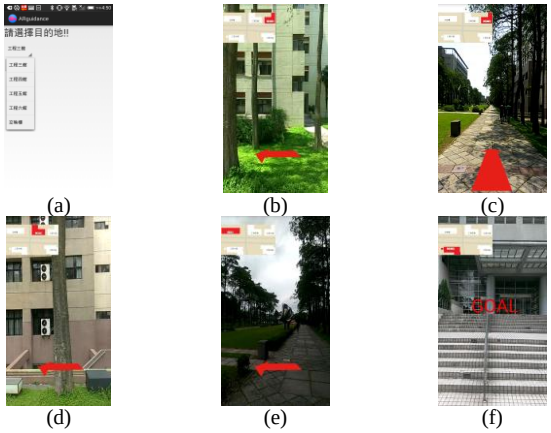
Steps.

1. Compute the direction D_i from N_i to N_{i+1}.
2. Select a proper guidance arrow according to D_i and D_{user} in the following way:
   2.1) if $-45^o \le (D_i - D_{user}) < 45^o$, select a forward arrow;
   2.2) if $45^o \le (D_i - D_{user}) < 135^o$, select a left-turn arrow;
   2.3) if $135^o \le (D_i - D_{user}) < 225^o$, select a backward arrow;
   2.4) if $225^o \le (D_i - D_{user}) < 315^o$, select a right-turn arrow.
3. Update the arrow to point to the correct direction and draw it on the user's mobile-device screen.

6. EXPERIMENTAL RESULTS

An area taken for conducting AR-based tour guidance experiments in this study is an avenue in the campus of

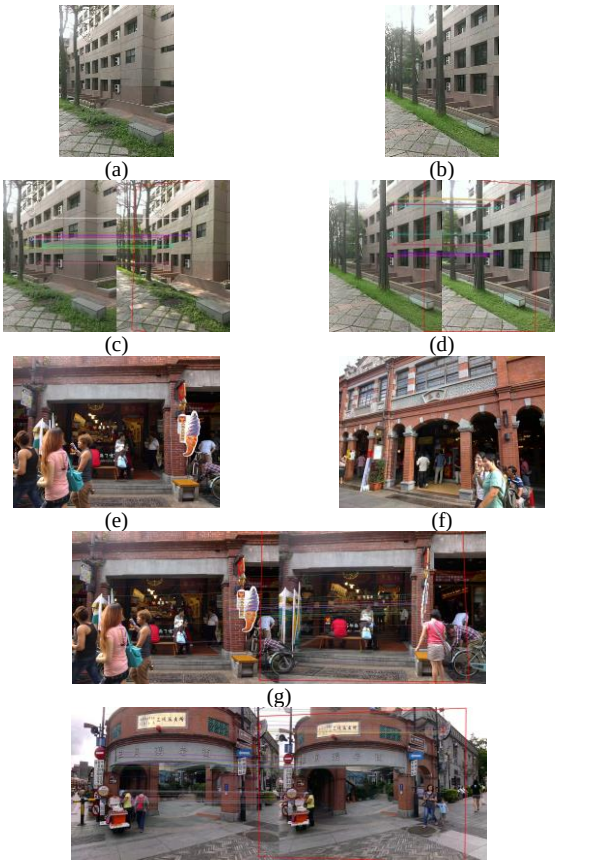
National Chiao Tung University with five large buildings alongside. An experimental result of the guidance process using the proposed system along a path in the tour area is shown in Fig. 10. The interface for selecting the destination building is shown in Fig. 10(a). Fig. 10(b) shows an arrow guiding the user to the next node. Fig. 10(c) shows an updated arrow when the user turned left. Figs. 10(d) and (e) are two results of guiding the user to the next nodes after updating the user's location. Fig. 10(f) shows the message "GOAL" augmented on the device screen when the user arrived at the destination.



(a) (b) (c) (d) (e) (f)

Fig. 10. An example of intermediate experimental results of a guidance process conducted by use of the proposed system.

Some more experimental results of image recognition by the method proposed in Sec. 4 are shown in Fig. 11. Figs. 11(a) and (b) are two input images which were taken by the user. They also appear in the left parts of Figs. 11(c) and (d), respectively. The right parts in Figs. 11(c) and (d) are the recognition results in which red quadrilateral rectangles are drawn to verify the correctness of the matchings. Two more examples are shown in Figs. (e) through (h).



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 11 Results of image recognition.

7. CONCLUSIONS

A system for AR-based tour guidance along streets has been proposed for use on a user-held mobile device. The system shows an augmented arrow in every guidance step on the mobile-device screen to lead the user to a goal building, and augment the building information on the screen for inspection. Techniques proposed to design such a system include a method for street-building recognition SURF matching; a method for speeding up feature matching using a k-d tree; a computer vision-based method for user localization; a method for path planning by the Dijkstra algorithm; and a method for creating guidance arrows in realtime to lead the user to the destination. The experimental results have revealed the feasibility of the proposed system for real applications.

Future studies may be directed to user localization with street images taken with tilt angles; use of other devices to implement the system, likes Google glass; providing a more convenient way to learn the environment map and an automatic way to construct the database; conducting experiments under different weather conditions and in various street scenes, etc.

REFERENCES

[1] MapFan.com. (2015, June). "MapFan AR Global," [online], available: https://www.mapfan.com/iphone /arg/

[2] G. Reitmayr and T. W. Drummond, "Going out: robust model-based tracking for outdoor augmented reality," *Proc. IEEE/ACM Int. Symp. on Mixed & Augmented Reality, 2006*, Santa Barbara, CA, pp. 109-118, Oct. 2006.

[3] T. Grosch, "PanoAR: Interactive Augmentation of Omni-Directional Images with Consistent Lighting," *Proc. CV/CG Collaboration Techniques & Applications*, INRIA Rocquencourt, France, pp. 25-34, 2005.

[4] W. Narzt, G. Pomberger, A. Ferscha, D. Kolb, R. Muller, J. Wieghardt, H. Hortner, and C. Lindinger, "Augmented reality navigation systems," *Universal Access in the Inform. Soc.*, Vol. 4, Issue 3, pp 177-187, March 2006.

[5] M. Krichenbauer, G. Yamamoto, T. Taketomi, C. Sandor, H. Kato, "Towards Augmented Reality User Interfaces in 3D Media Production," *Proc. IEEE/ACM Int. Symp. on Mixed & Augmented Reality, 2014*, Munich, Germany, pp. 23-28, Sept. 2014.

[6] Herbert Bay, T. Tuytelaars and L. Van Gool, "SURF: Speeded Up Robust Features," *Proc. 9th European Conf. on CV*, Graz, Austria, pp. 404-417, May 2006.

[7] L. L. Wang and W. H. Tsai, "Camera calibration by vanishing lines for 3D computer vision," *IEEE Trans. PAMI*, Vol. 13, No. 3, pp. 370-376, March 1991.

[8] B. Katz, et al. "Navig: Augmented reality guidance system for the visually impaired," *Virtual Reality*, Vol. 16, No. 4, pp. 253–269, June 2012.

[9] Google. (2015, March). "Google Goggles: use pictures to search the web," [online], available: http://www.google.com/mobile/goggles

[10] Kooaba. (2015, March). "Kooaba Visual Search: get instant product information," [online], available: http://www.kooaba.com.

[11] Amazon. (2015, March). "Amazon Remembers: create a visual list of products," [online], available: http://www.amazon.com/gp/ remembers.

[12] H. Kato and M. Billinghurst, "Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System," *Proc. IEEE & ACM Int. Workshop on Augmented Reality*, San Francisco, CA, pp. 85-64, Oct. 1999.