Localization and Mapping of Surveillance Cameras in City Map

Wee Kheng Leow
Dept. of Computer Science
National Univ. of Singapore
Computing 1
Singapore 117590, Singapore
leowwk@comp.nus.edu.sg

Cheng-Chieh Chiang
Dept. of Info. Tech.
Takming U. of Science & Tech.
No. 56, Sec. 1, Huanshan Rd.
Taipei 11451, Taiwan, R.O.C.
kevin@csie.ntnu.edu.tw

Yi-Ping Hung
Dept. Comp. Sci. & Info. Eng.
National Taiwan University
No. 1, Sec. 4, Roosevelt Road
Taipei 106, Taiwan, R.O.C.
hung@csie.ntu.edu.tw

ABSTRACT

Many large cities have installed surveillance cameras to monitor human activities for security purposes. An important surveillance application is to track the motion of an object of interest, e.g., a car or a human, using one or more cameras, and plot the motion path in a city map. To achieve this goal, it is necessary to localize the cameras in the city map and to determine the correspondence mappings between the positions in the city map and the camera views. Since the view of the city map is roughly orthogonal to the camera views, there are very few common features between the two views for a computer vision algorithm to correctly identify corresponding points automatically. This paper proposes a method for camera localization and position mapping that requires minimum user inputs. Given approximate corresponding points between the city map and a camera view identified by a user, the method computes the orientation and position of the camera in the city map, and determines the mapping between the positions in the city map and the camera view. The performance of the method is assessed in both quantitative tests and practical application. Quantitative test results show that the method is accurate and robust in camera localization and position mapping. Application test results are very encouraging, showing the usefulness of the method in real applications.

Categories and Subject Descriptors

I.4 [**Image Processing and Computer Vision**]: Applications

General Terms

Algorithms, Experimentation

Keywords

camera localization, position mapping, surveillance

1. INTRODUCTION

Surveillance cameras are becoming an important asset in homeland security. Many large cities have installed surveillance cameras along city streets to monitor and record human activities. The recorded videos provide valuable information to the police for solving crimes ranging from theft to terrorist activities. To use the information in the videos, it is necessary to sieve through the videos manually to identify the relevant video segments. A computer system for automatic analysis of videos will save a lot of manual effort.

An important surveillance application is to track the motion of an object of interest, e.g., a car or a human, using one or more cameras, and plot the motion path in a city map. To achieve this goal, it is necessary to localize the cameras in the city map and to determine the correspondence mappings between the positions in the city map and the camera views. The localization and mapping problems are non-trivial because a city map is a drawing of the top-down view, which is roughly orthogonal to the camera view (Fig. 1). Since the viewing angles of the city map and the camera view differ significantly, there are very few common features between the two views for a computer vision algorithm to correctly identify corresponding points automatically. This is true even if a satellite image of the city is available, let alone a city map that lacks distinct features.

Surveillance cameras are often stationary and are typically placed far apart such that their views do not overlap. This removes the possibility of using the method of multiple-view geometry to calibrate and localize the cameras. Even if the cameras can pan and tilt, thus permitting the use of multiple-view geometry, it is still a challenge to localize the cameras in the city map because the camera views are orthogonal to the view in the city map.

This paper proposes a method for camera localization and position mapping that requires minimal user inputs. The user simply identifies approximate corresponding 3D points in the city map and 2D points in the camera view. The method will then computes accurate camera orientation and position in the city map, and accurate mapping between the positions in the two views. It can obtain the best-fit solutions even though the user-specified correspondence is inaccurate. Both quantitative tests and practical application tests have been performed to assess the method's accuracy.

Quantitative test results show that the method is accurate and robust in camera localization and position mapping. Application test results are very encouraging, showing the usefulness of the method in real applications.

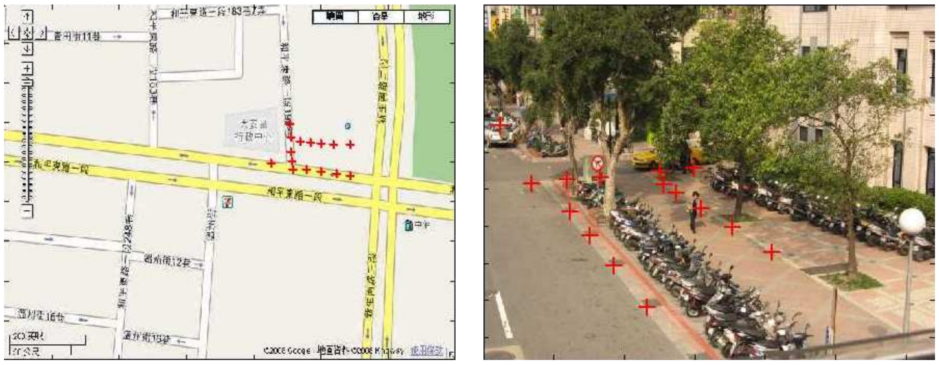

(a) (b)

Figure 1: Camera localization and position mapping. The view in (a) the city map is roughly orthogonal to (b) the camera view. Red crosses denote the approximate corresponding points.

2. RELATED WORK

Visual surveillance is a very active research area and many visual surveillance systems have been developed. For example, the W^4 system [8] is a real-time system that detects and tracks multiple people and monitors their activities in an outdoor environment. The Video Surveillance and Monitoring (VSAM) system [4] adopts a distributed network of active video cameras, and allows a single human operator to monitor human activities in a cluttered environment. Other multimedia surveillance systems are reviewed in [5]. A more detailed review of existing work related to visual surveillance is given in [12], which includes motion detection, object tracking, human detection, understanding of human behavior, and fusion of data from multiple cameras. A common theme in such systems is the detection, tracking, and monitoring of human activities using one or more video cameras. However, they do not transfer the tracked paths in the camera views into a city map, as is performed in our work.

There are several systems that are somewhat similar to our work but differ in important ways. The ViewFinder system [6] provides a 3D world model like GoogleEarth. It allows a user to indicate a 3D location in the world model, and displays the 2D photo at the specified location. Unlike our work, it does not map 3D locations in world model to 2D positions in the camera view. The system in [16] does not perform automatic placement of cameras. Instead, a visualization tool is provided for the user to manually position and orientate the cameras in the world model. The system in [15] proposes the use of a virtual environment simulator for evaluating the design of a visual surveillance system. It provides camera views as well as a top-down view of the environment under surveillance, but it is implemented for a virtual environment instead of a real environment.

Camera calibration is a necessary step in real applications of visual surveillance. A variety of camera calibration algorithms have been proposed [3, 9, 19, 21]. In our work, the Camera Calibration Toolbox for Matlab [3] is used to compute the intrinsic parameters of a camera.

Computation of the extrinsic parameters of a camera is often called the absolute orientation problem. That is, given two sets of corresponding 3D points, compute the rotation and translation between the two sets. Closed-form solutions of this problem are available [1, 10, 11, 20]. In contrast, our paper illustrates a method for computing a camera's extrinsic parameters with respect to a city map, given approximate correspondence between the 3D points in the city map and 2D points in the camera view.

Related to the localization of cameras in a city map is the localization of robots in a working environment. Typically, the robot localization problem seeks to determine the 2D orientation and position of the robot within a 2D environment, which is different from 3D localization of cameras in our work. Robot localization can be achieved using laser, sonar, or stereo sensor [14]. GPS is another practical technique and has been widely used in many applications. However, it has a localization error of about 10–20 meters [7].

Vision-based methods for robot localization use visual landmarks in the images (i.e., camera views) to perform robot localization [2, 13, 17]. The method in [2] assumes that the robots can identify visual landmarks and measure bearings relative to each other. The method in [13] uses both image features as well as a known 3D model of the environment. Known 2D-2D correspondences between camera views and 3D-2D correspondences between the 3D model and the camera views are used for camera localization. The method in [17] solves the localization problem by comparing the visual landmarks in the camera view with those in known database images. Minerva, a museum tour-guide robot, applies Monte Carlo method to compute 2D robot location based on sensor inputs [18]. In contrast, this paper illustrates a method that performs 3D camera localization using approximate corresponding points between a city map and a camera view.

3. OVERVIEW OF THE PROBLEMS

As discussed in Section 1, the view in a city map is roughly orthogonal to the camera view. There are very few common features between the two views for a computer vision algorithm to correctly identify corresponding points automatically. One way to establish correspondence is to manually locate the corresponding points in the city map and the camera view. These user inputs can be facilitated by a graphical user interface (GUI) that allows the user to select landmark points and measures the direction and distance between any two landmark points. User's knowledge of the environment can also help the user identify corresponding points.

With the GUI, a point $\mathbf{x}_i = (x_i, y_i)^\top$ in the camera view can be accurately located using subpixel algorithms because the user can clearly identify important landmarks or feature points in the image. On the other hand, it is impossible to accurately locate the corresponding 3D point in the city map. At best, the user can only locate its position (X_i, Y_i) approximately, and estimate the height Z_i. It is impractical for the user to physically measure the directions and distances of the landmark points in the actual environment.

There is another source of inherent inaccuracy in manual localization in a city map. To locate a point in the map, the user needs to judge relative distances from some landmarks in the map such as street lanes and corners. So, a map area of at least one city block needs to be displayed on the computer screen to include at least the four street corners of the city block. Assume that a city block of 100m×100m can be displayed on the computer screen at a resolution of 1000×1000 pixels. Then, the limit of accuracy of manual localization is 0.1m (i.e., one pixel). The localization uncertainly can be as large as 10 pixels, which is 1m.

In conclusion, although the correspondence between 3D positions in city map and 2D image positions are known, the 3D positions $\mathbf{X}_i = (X_i, Y_i, Z_i)^\top$ are approximate while the 2D image positions $\mathbf{x}_i$ are accurate. The challenge is to determine the camera's orientation and position, and the mapping between the positions in the city map and the camera view, given such approximate correspondence.

Let $\hat{\mathbf{x}}_i$ denote the homogeneous form of $\mathbf{x}_i$ such that $\hat{\mathbf{x}}_i = (x_i, y_i, 1)^\top$. Then, the the 3D position in city map is related to the 2D image position by the equation

$$\rho_i \hat{\mathbf{x}}_i = \mathbf{K}\,\mathbf{R}\,(\mathbf{X}_i - \mathbf{C}) \tag{1}$$

where ρ_i is a scaling parameter, $\mathbf{K}$ contains the camera's intrinsic parameters, and $\mathbf{R}$ and $\mathbf{C}$ are the camera rotation matrix and camera center in the world coordinate frame. In this paper, we assume that $\mathbf{K}$ is known because it can be computed using camera calibration algorithms [3, 21].

Now, we can formulate the camera localization problem:

Camera Localization

> Given n pairs of corresponding 3D positions $\mathbf{X}_i$ and 2D image positions $\mathbf{x}_i$, where $\mathbf{X}_i$ are approximate and $\mathbf{x}_i$ are accurate, and a known camera parameter matrix $\mathbf{K}$, determine the camera rotation matrix $\mathbf{R}$ and camera center $\mathbf{C}$ relative to the world coordinate frame of the city map.

The algorithms for camera localization and position mapping will be discussed in the following sections.

4. CAMERA LOCALIZATION

The 3D coordinates $\mathbf{X}_i$ of a 3D point in the world coordinate frame are related to its 3D coordinates $\mathbf{X}'_i$ in the camera coordinate frame by the equation

$$\mathbf{X}'_i = \mathbf{R}\,(\mathbf{X}_i - \mathbf{C}). \tag{2}$$

The projection line $\mathbf{P}_i$ from 3D point p_i to the image plane is given by

$$\mathbf{P}_i = \mathbf{K}^{-1}\hat{\mathbf{x}}_i. \tag{3}$$

It passes through the 2D image point $\hat{\mathbf{x}}_i$ and the camera center $\mathbf{C}$ (Fig. 2).

Suppose that the 3D points $\mathbf{X}_i$ can be accurately determined. In this case, if $\mathbf{R}$ and $\mathbf{C}$ are accurately estimated, $\mathbf{X}'_i$ would lie on $\mathbf{P}_i$. Otherwise, $\mathbf{X}'_i$ would be located at some distance from $\mathbf{P}_i$ (Fig. 2).

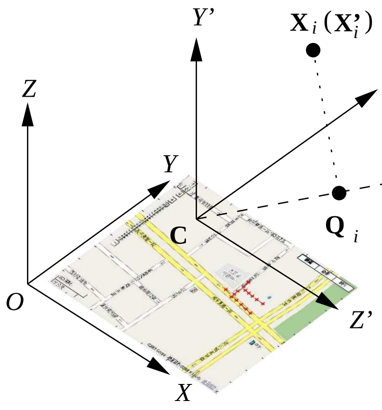


Figure 2: World coordinate frame (XYZ) and camera coordinate frame $(X'Y'Z')$. Ideally, the 3D point $\mathbf{X}_i$ should lie on its projection line $\mathbf{P}_i$ derived from its corresponding 2D image point $\mathbf{x}_i$.

In practice, the 3D points $\mathbf{X}_i$ can only be approximately determined. In this case, $\mathbf{X}'_i$ does not lie on $\mathbf{P}_i$ even if the correct $\mathbf{R}$ and $\mathbf{C}$ are used to compute $\mathbf{X}'_i$. Then, an iterative algorithm can be used to determine the best fitting $\mathbf{R}$ and $\mathbf{C}$ by iteratively moving the set of points $\mathbf{X}'_i$ onto their corresponding projection lines $\mathbf{P}_i$. This is the basis of our camera localization algorithm.

Camera Localization Algorithm

Initialize $\mathbf{R} = \mathbf{I}$ and $\mathbf{C} = \mathbf{0}$.
Repeat until convergence:

1. Compute 3D points in camera coordinate frame:
$$\mathbf{X}'_i = \mathbf{R}\,(\mathbf{X}_i - \mathbf{C}). \tag{4}$$
2. Compute projection lines:
$$\mathbf{P}_i = \mathbf{K}^{-1}\hat{\mathbf{x}}_i. \tag{5}$$
3. Compute ideal position $\mathbf{Q}_i$ of $\mathbf{X}'_i$:
The ideal position $\mathbf{Q}_i = \rho_i \mathbf{P}_i$ lies on its corresponding project line $\mathbf{P}_i$. That is, $\mathbf{Q}_i = \rho_i \mathbf{P}_i$ for an appropriate ρ_i. The value of ρ_i can be obtained by substituting Eq. 2 and 3 into Eq. 1 to obtain
$$\rho_i \mathbf{P}_i = \mathbf{X}'_i \tag{6}$$
$$\rho_i = \frac{\mathbf{P}_i^\top \mathbf{X}'_i}{\mathbf{P}_i^\top \mathbf{P}_i}. \tag{7}$$
Eq. 7 is the best-fit solution when $\rho_i \mathbf{P}_i$ is not exactly equal to $\mathbf{X}'_i$. In addition, $\rho_i \mathbf{P}_i$ is also the projection of $\mathbf{X}'_i$ on $\mathbf{P}_i$.
The mean distance of $\mathbf{X}'_i$ to its corresponding projection line $\mathbf{P}_i$ is given by the difference E_Q:
$$E_Q = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{Q}_i - \mathbf{X}'_i\|. \tag{8}$$
By iteratively minimizes E_Q, the algorithm updates $\mathbf{R}$ and $\mathbf{C}$ by moving the set of points $\mathbf{X}'_i$ onto their corresponding projection lines $\mathbf{P}_i$.

4. Compute best-fitting camera rotation matrix $\mathbf{R}$:
This step computes the best-fitting camera rotation matrix using the method in [11]:

(a) Remove translation:
Remove translational components of the 3D points by computing

$$\begin{aligned} \mathbf{r}_i &= \mathbf{X}_i - \bar{\mathbf{X}}, \\ \mathbf{r}'_i &= \mathbf{Q}_i - \bar{\mathbf{Q}} \end{aligned} \tag{9}$$

where

$$\begin{aligned} \bar{\mathbf{X}} &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{X}_i, \\ \bar{\mathbf{Q}} &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{Q}_i. \end{aligned} \tag{10}$$

(b) Perform eigen-decomposition of the point set: Form matrix M.
Form matrix M.

$$\mathbf{M} = \sum_{i=1}^{n} \mathbf{r}'_i \mathbf{r}_i^\top \tag{11}$$

and compute $\mathbf{U} = \mathbf{M}^\top \mathbf{M}$. Perform eigenvalue decomposition of $\mathbf{U}$ to obtain the eigenvectors $\mathbf{v}_j$ and eigenvalues λ_j, for $j = 1, 2, 3$. The matrix $\mathbf{U}$ is related to the eigenvectors and eigenvalues by

$$\mathbf{U} = \sum_{j=1}^{3} \lambda_j \mathbf{v}_j \mathbf{v}_j^\top. \tag{12}$$

(c) Compute the new camera rotation matrix $\mathbf{R}$:

$$\mathbf{R} = \mathbf{M}\,\mathbf{U}^{-1/2} \tag{13}$$

where

$$\mathbf{U}^{-1/2} = \sum_{j=1}^{3} \frac{1}{\sqrt{\lambda_j}} \mathbf{v}_j \mathbf{v}_j^\top. \tag{14}$$

5. Compute new camera center $\mathbf{C}$:
The new camera center $\mathbf{C}$ can be estimated by substituting the means of the point sets into Eq.2:

$$\mathbf{C} = \mathbf{R}^{-1}\left(\mathbf{R}\bar{\mathbf{X}} - \bar{\mathbf{Q}}\right). \tag{15}$$

Initially, when $\mathbf{R}$ and $\mathbf{C}$ are inaccurate, the rigid transformations $\mathbf{X}'_i$ of $\mathbf{X}_i$ are located at some distances from $\mathbf{P}_i$. The algorithm computes new $\mathbf{R}$ and $\mathbf{C}$ from $\mathbf{X}_i$ and $\mathbf{Q}_i$, and brings $\mathbf{X}'_i$ closer to $\mathbf{P}_i$. Therefore, as the algorithm iterates, $\mathbf{X}'_i$ will approach $\mathbf{P}_i$. If $\mathbf{X}_i$ are accurate, then $\mathbf{X}'_i$ will eventually fall on $\mathbf{P}_i$. Otherwise, $\mathbf{X}'_i$ will remain at some distance from $\mathbf{P}_i$ depending on the inaccuracy of $\mathbf{X}_i$. The average inaccuracy of $\mathbf{X}_i$ is given by E_Q (Eq. 8).

Our tests show that with $n = 3$, the algorithm in [11] sometimes produces a best-fit reflection matrix instead of a rotation matrix for $\mathbf{R}$. On the other hand, with $n \geq 4$, a rotation matrix is always produced.

At first glance, it may appear that the algorithm should somehow refine the positions of the 3D points $\mathbf{X}_i$ in the hope that more accurate $\mathbf{R}$ and $\mathbf{C}$ can be estimated. However, our experiments show that refining $\mathbf{X}_i$ without additional knowledge can cause $\mathbf{C}$ to move away from the true camera center even when both $\mathbf{R}$ and E_Q are reduced. That is, the algorithm can obtain a best-fit $\mathbf{R}$ even when $\mathbf{C}$ is inaccurate. Therefore, our current algorithm does not refine the positions of the 3D points.

5. POSITION MAPPING

After camera localization, the correspondence mapping between the positions in the city map and the camera view can be established. 3D-to-2D position mapping is straightforward. Given a 3D position $\mathbf{X}$ in the city map, compute $\mathbf{x} = (x, y)^\top$ as follows:

$$\begin{aligned} \begin{bmatrix} u \\ v \\ w \end{bmatrix} &= \mathbf{K}\,\mathbf{R}(\mathbf{X} - \mathbf{C}), \\ x &= \frac{u}{w}, \\ y &= \frac{v}{w}. \end{aligned} \tag{16}$$

The inverse 2D-to-3D position mapping is less trivial because multiple 3D positions can map to the same 2D image position. However, if the Z-coordinate of the 3D position is known, then it is possible to compute the 3D position given the corresponding 2D image position.

From Eq. 1, we obtain

$$\rho\,\hat{\mathbf{x}} = \mathbf{K}\,\mathbf{R}\,(\mathbf{X} - \mathbf{C}). \tag{17}$$

Rearranging Eq. 17 gives

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \rho\,\mathbf{R}^{-1}\mathbf{K}^{-1}\hat{\mathbf{x}} + \begin{bmatrix} C_X \\ C_Y \\ C_Z \end{bmatrix}. \tag{18}$$

Let $\mathbf{V} = (V_X, V_Y, V_Z)^\top$ denote $\mathbf{R}^{-1}\mathbf{K}^{-1}\hat{\mathbf{x}}$. Then, X and Y can be computed from the equations:

$$\begin{aligned} \rho &= \frac{Z - C_Z}{V_Z}, \\ X &= \rho V_X + C_X, \\ Y &= \rho V_Y + C_Y. \end{aligned} \tag{19}$$

6. QUANTITATIVE TESTS

6.1 Test Setup

To quantitatively evaluate the performance of the algorithms, it is necessary to generate synthetic test data with known ground truth. Typically, a surveillance camera is placed at a height of 2.5m or more, and is oriented slightly downward to capture the street area in front of it. So, the camera center was set at $C = (20, 20, 2.5)^\top$ in unit of m. The camera coordinate frame was placed such that it was pointing down towards the ground plane at an angle of 15°. Its X'-axis was aligned with the Y-axis of the world coordinate frame, and its Z'-axis was rotated downward by an angle of $\theta = 15°$ (Fig. 3). In this configuration, the rotation matrix $\mathbf{R}$ was given by

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \\ \cos\theta & 0 & \sin\theta \end{bmatrix}. \tag{20}$$

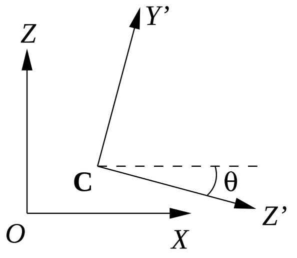


Figure 3: Ground truth camera coordinate frame in the quantitative tests. Camera's X'-axis and world's Y-axis are aligned and point into the page.

The angle θ was set at 15°. The camera parameter matrix $\mathbf{K}$ was obtained by applying the Camera Calibration Toolbox for Matlab [3] to a real camera, whose image size was set at 640×480 pixels.

A set of $n = 30$ 3D points $\mathbf{X}_i^*$ were randomly generated in front of the camera at a distance of at least 5m away and in a space of 50m×50m×5m. The maximum height was set at 5m because most interesting moving objects such as humans and cars do not exceed 5m in height. The corresponding 2D image points $\mathbf{x}_i^*$ were computed from $\mathbf{X}_i^*$ using Eq. 1. These data form the *ground truth* set $S^* = (\mathbf{X}_i^*, \mathbf{x}_i^*)$.

Noisy data sets were generated by adding random noise to the ground truth 3D positions. In data set $S(\mu) = (\mathbf{X}_i, \mathbf{x}_i^*)$, $\mathbf{X}_i = \mathbf{X}_i^* + \boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_i$ is the random noise term. The X- and Y-components of $\boldsymbol{\mu}_i$ fall in the range $[-\mu, +\mu]$ and the Z-component falls in the range $[-0.1\mu, +0.1\mu]$. $S(0) = S^*$. The noise level in the Z-direction is smaller than those in the X- and Y-direction because the vertical height can be more accurately estimated. For example, the actual height of a wall can be measured using a measuring tape.

Seven data sets $S(\mu)$ were used for camera localization tests, with noise level μ ranging from 0 to 1m. After camera localization, 3D-to-2D and 2D-to-3D mappings were performed to compute the mapping errors.

Five error measures were used to assess the performance of the algorithms:

- E_Q: Optimization quality
  This is the mean distance between $\mathbf{X}_i'$ and $\mathbf{Q}_i$. It measures how far away are the 3D points from their corresponding project lines. At convergence, E_Q should be close to 0. The unit of E_Q is m.

- E_C: Camera position error
  This is the distance between the computed and ground truth camera centers. The unit of E_C is m.

- E_R: Camera orientation error
  This is the root-mean-squared difference between the elements in the computed and ground truth rotation matrices. E_R has no unit.

- E_x: 3D-to-2D position mapping error
  This is the mean distance between computed and ground truth 2D image positions. The unit of E_x is pixel.

- E_X: 2D-to-3D position mapping error
  This is the mean distance between the computed and ground truth 3D positions. The unit of E_X is m.

6.2 Convergence

The data set $S(1)$ with noise level of 1m was used to test the convergence of the camera localization algorithm. This data set was used to show that the algorithm can converge efficiently even in the presence of large amounts of noise. The algorithm was executed for 100 iterations.

Figure 4 shows that the algorithm stabilizes very quickly after about 45 iterations. In particular, the optimization quality E_Q stabilized at about 0.4, and the errors in camera orientation and position E_R and E_C stabilized at about 0.0002 and 0.2 respectively.

The convergence of the algorithm goes through two phases. In the first phase, the optimization quality E_Q decreases by an order of magnitude over just the first few iterations (Fig. 4(a)). This rapid convergence is due to the computation of the best-fitting $\mathbf{R}$ in Step 4. That is, at each iteration of the algorithm, a linear least-square fit of $\mathbf{R}$ is computed using eigen-decomposition technique. If the 3D coordinates were accurate, the 3D points $\mathbf{X}_i$ and their coordinates $\mathbf{X}_i'$ in the camera frame would differ by only a rigid transformation defined by $\mathbf{R}$ and $\mathbf{C}$, and a single iteration of Step 4 would be sufficient to compute the correct $\mathbf{R}$ and $\mathbf{C}$.

In the second phase, the algorithm goes through a more gradual refinement process where E_Q decreases by another order of magnitude over about 40 iterations (Fig. 4(a)). This phase iteratively refines $\mathbf{R}$ and $\mathbf{C}$ by re-estimating the 3D coordinates $\mathbf{X}_i'$ of the 3D points and their ideal positions $\mathbf{Q}_i$ on the projection lines $\mathbf{P}_i$ (Steps 1–3). It allows $\mathbf{R}$ and $\mathbf{C}$ to be refined smoothly, which is evident in the smooth decrease of E_R and E_C (Fig. 4(b, c)).

6.3 Accuracy

The data set $S(\mu)$ with noise level μ ranging from 0 to 1m were used to test the accuracy of the algorithms. Figure 5 illustrates the test results. At low-noise levels, the errors in camera orientation E_R and position E_C are small. As a result, the mapping errors E_x and E_X are also small.

As the noise level increases, E_R remains practically zero at all noise levels tested. On the other hand, the error of camera center E_C grows with increasing noise level. That is, camera rotation $\mathbf{R}$ can be more accurately estimated than camera center $\mathbf{C}$. The 3D-to-2D mapping error E_x and 2D-to-3D mapping error E_X also grow with increasing noise level. These results suggest that mapping errors are influenced by E_C more than by E_R. Nevertheless, it is interesting to note that, while the growth of E_C with increasing noise level seems to be faster-than-linear, the growth of E_x and E_X remain linear. The slower growth in error could be the result of high accuracy in estimating camera rotation.

7. PRACTICAL APPLICATIONS

7.1 Test Setup

The city map and camera view shown in Fig. 6 were used for qualitative evaluation in practical application. The user manually marked 15 approximate corresponding points (Fig.6(a, b)) in the city map and the camera view. The points chosen were along road sides and at road corners. The user only needed to click the respective positions in the city map and the camera view in the GUI, which was accomplished fairly easily for a user who was familiar with the actual environment. The GUI automatically computed the

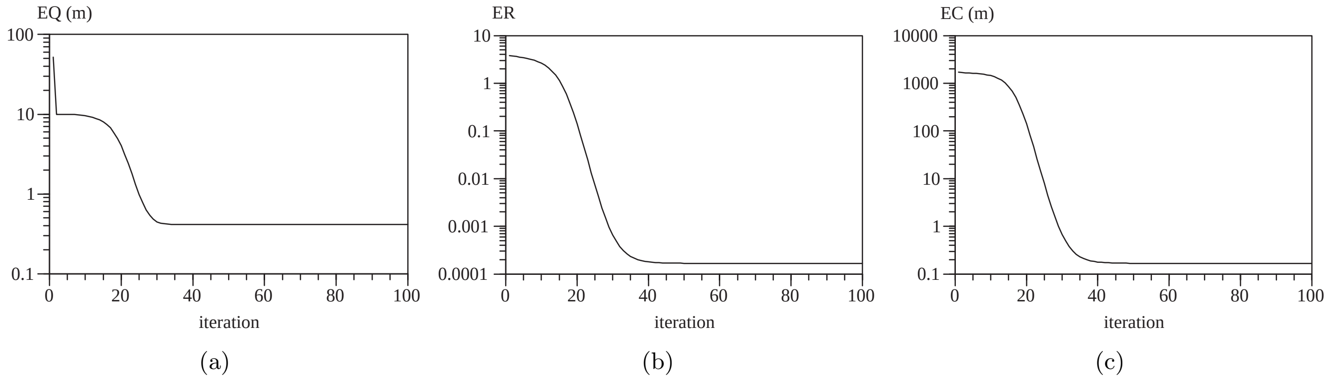


Figure 4: Convergence of camera localization algorithm. Convergence of (a) E_Q, (b) E_R, and (c) E_C.

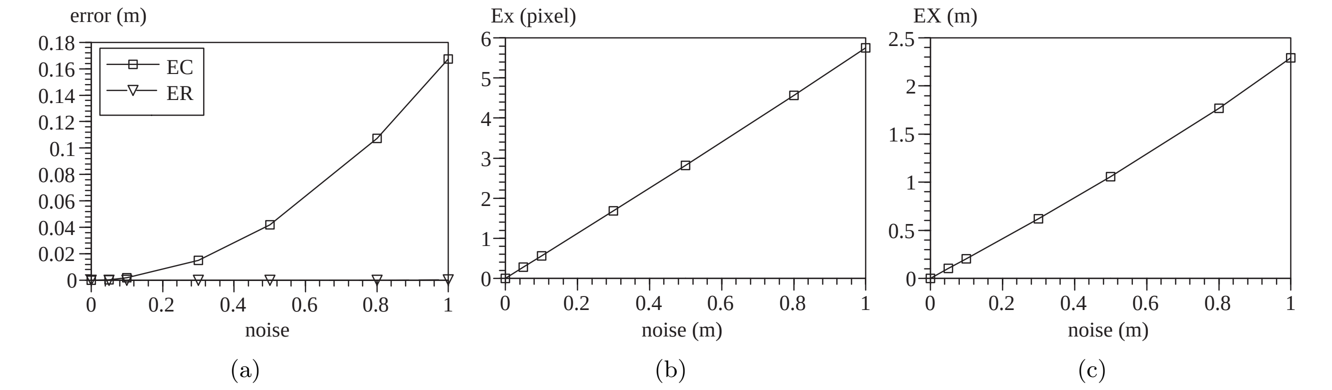


Figure 5: Test results at various noise levels. (a) Camera localization results. (b) 3D-to-2D position mapping error. (c) 2D-to-3D position mapping error.

pixel locations of the 2D points in the camera view and the approximate physical X- and Y-coordinates in the city map. The approximate Z-coordinates were entered by the user.

Most of the points lied on the ground and had Z-value of 0. The point on the top of the car had a non-zero Z value that was an estimate of the height of the car. Note that the area in the city map where the points were marked was relatively featureless. Therefore, the positions in the city map could not be accurately marked manually. The camera parameter matrix $\mathbf{K}$ was determined using the Camera Calibration Toolbox for Matlab [3].

Even though most test points have Z = 0, it does not imply that a 2D-to-2D mapping is performed in our algorithm. It only means that the points happen to fall onto a plane in 3D space. This plane is not parallel to the camera's image plane. The equations given in the algorithm still perform 3D-to-2D and 2D-to-3D mapping.

7.2 Accuracy

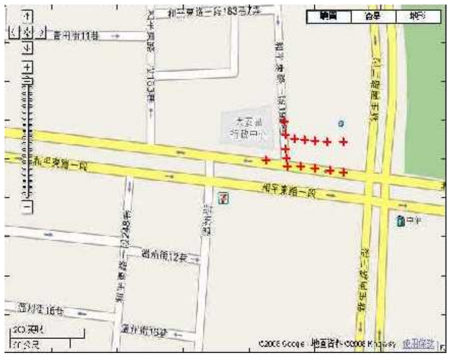
Given the manually marked corresponding points, the camera localization algorithm was executed. After camera localization, 3D-to-2D position mapping was performed using the 3D points in Fig. 6(a), and 2D-to-3D position mapping was performed using the 2D points in Fig. 6(b).

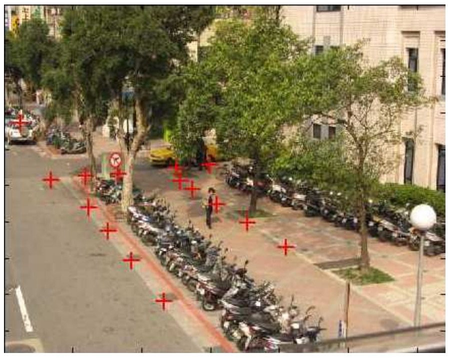
The camera's position estimated by the algorithm is indicated by the blue cross near the right edge of the image in Fig. 6(c). The mapped 2D points (Fig. 6(d)) computed by the position mapping algorithm are virtually identical to those in the manually marked points (Fig. 6(b)). On the other hand, the mapped 3D points (Fig. 6(c)) are displaced slightly from the positions of the manually marked points (Fig. 6(a)). Nevertheless, the relative positions of the points are preserved. These test results show that the algorithm can perform camera localization and position mapping reasonably accurately in real application.

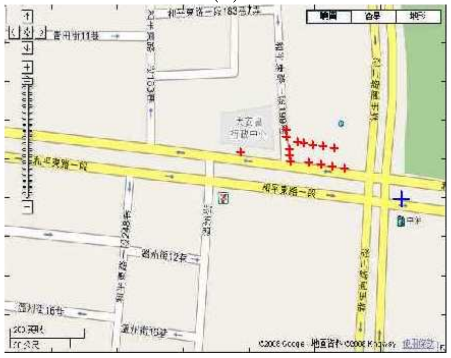
In the previous position mapping tests, the test data were the same as those used for camera localization. To evaluate the accuracy of the algorithm in mapping novel data points, two additional sets of tests were performed.

In the first set of tests, 3D points were manually marked in the city map to emulate the walking paths of different people. The Z-coordinates were all set at 0 to correspond to the positions of the people's feet. These 3D points were different from those used for camera localization. The 3D-to-2D mapping algorithm was executed to compute the corresponding 2D positions in the camera view. Figure 7 shows that the 2D positions in the camera view correspond very well to those in the city map. In particular, the relative positions of the points in the city map are preserved in the camera view.

In the second set of tests, 2D points were manually marked in the camera view to emulate the tracked paths of different people. These 2D points were different from those used for camera localization. The 2D-to-3D mapping algorithm was executed to compute the corresponding 3D points in the city map. It was assumed that the people's feet were tracked in the camera view. So, the Z-coordinates of the corresponding


(a)


(b)


(c)

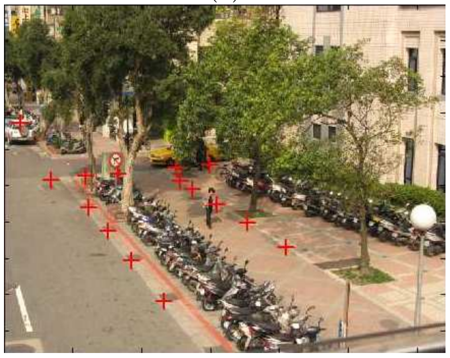

(d)

Figure 6: Mapping positions between city map and camera view. Corresponding points (red crosses) are manually marked in (a) city map and (b) camera view. (c) 2D-to-3D mapping results. Blue cross: recovered camera position. (d) 3D-to-2D mapping results.

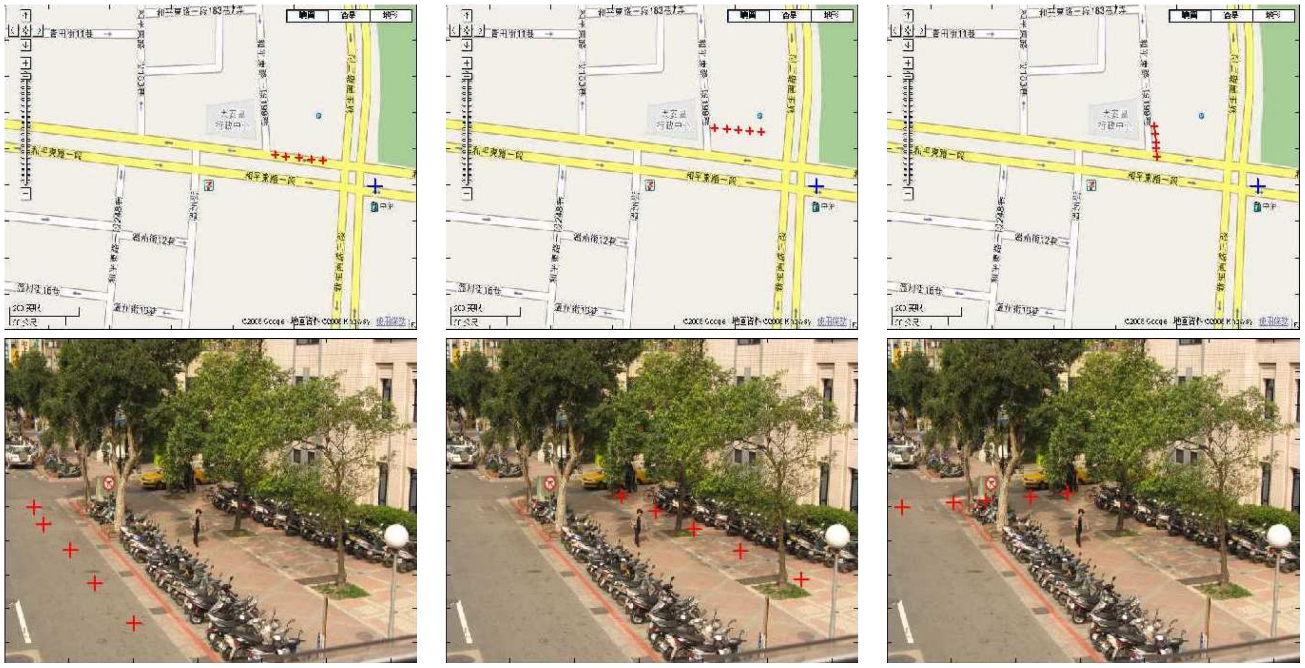


Figure 7: 3D-to-2D position mapping test results. (Top) Input 3D positions indicating human's walking paths. (Bottom) Mapped 2D positions in the camera view.

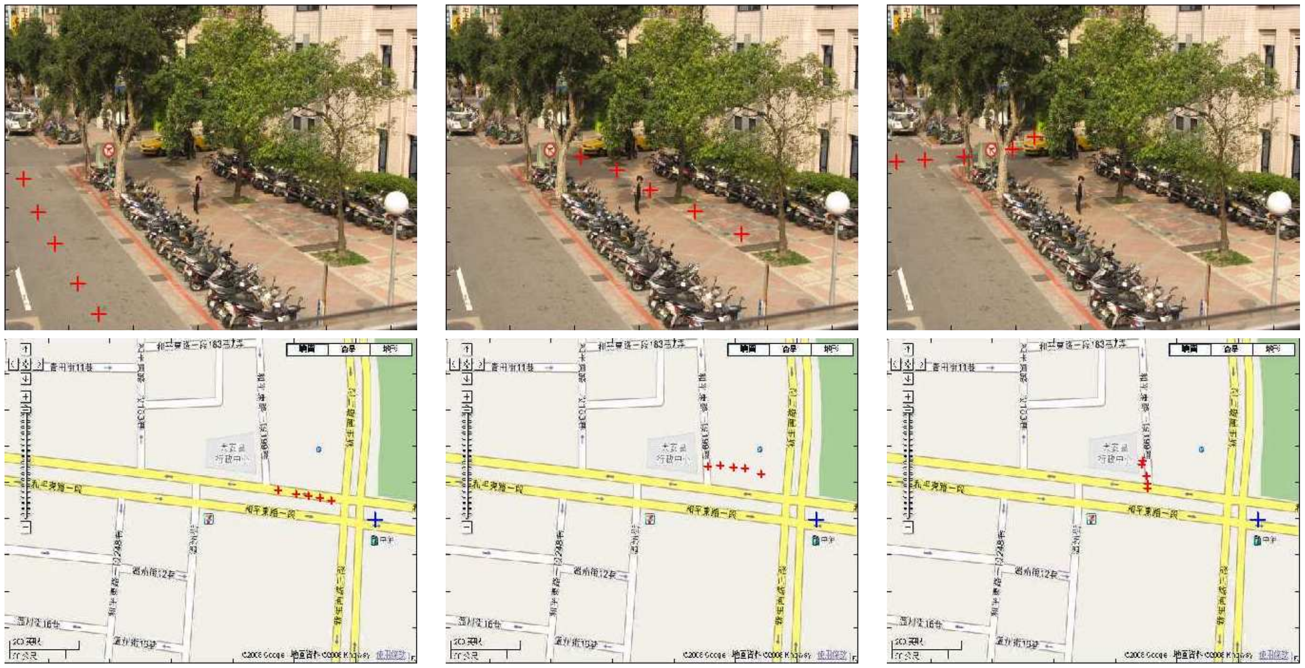


Figure 8: 2D-to-3D position mapping test results. (Top) Input 2D positions indicating human's walking paths. (Bottom) Mapped 3D positions in the city map.

3D points were all set to 0. Figure 8 shows that the 3D positions in the city map correspond very well to those in the camera view. In particular, the relative positions of the points in the camera view are preserved in the city map.

A plausible method of improving the algorithms' accuracy is to impose constraints on the points during camera localization. For example, the points selected for camera localization may fall on some straight lines. Our preliminary test results indicate that imposing linear constraints on the algorithm alone does not produce significant improvement because the algorithm converges very rapidly.

Another plausible method is to refine the coordinates of the 3D points after an application of the camera localization algorithm. The refined 3D points are constrained to lie on the same straight lines. After refinement, the camera localization algorithm is applied again. Unfortunately, this method does not necessarily produce more accurate results. In Fig. 9, the mapped points are constrained to lie on straight lines. But, a group of points on a straight line can be displaced from its actual location. The reason is that the 3D points can be moved quite far away from their actual positions even when they are constrained to lie on straight lines. Therefore, other forms of constraints are required if we wish to further improve the accuracy of the algorithms.

7.3 Consistency

To evaluate the consistency of the camera localization algorithm, it was re-evaluated on the same city map and camera view but with different user-selected approximate corresponding points (Fig. 10). The test results exhibit similar characteristics as those in the first test (Section 7.2, Fig. 6). In particular, the relative positions of the mapped points are preserved. The root-mean-squared difference between the camera rotation matrices $\mathbf{R}$ obtained in the first and second tests is 0.0309. The distance between the camera center $\mathbf{C}$ obtained in the two tests is 1.04m. These errors are reasonably small in practical application but larger than those obtained in the quantitative tests with synthetic data (Section 6.2). That is, the variation in user inputs and amount of noise are larger in practical applications.

In the next set of tests, 3D-to-2D and 2D-to-3D position mappings were performed on the same set of test data using the camera matrices $\mathbf{R}$ and centers $\mathbf{C}$ obtained in the first and second tests. Test results show that, due to the differences in the $\mathbf{R}$ and $\mathbf{C}$ obtained in the two tests, the same 3D points were mapped to slightly different 2D points, and vice versa (Fig. 11). Nevertheless, the points were consistently mapped and their relative positions were preserved.

8. CONCLUSION

This paper has presented a method for localizing cameras in a city map and mapping the positions between the city map and the camera view. The camera localization and position mapping problems are non-trivial because the view in the city map is roughly orthogonal to the camera view. Since the two views differ by a very large viewing angle, there are very few common features between them for a computer vision algorithm to identify corresponding points automatically.

The proposed method requires minimal user inputs. The user simply identifies approximate corresponding 3D points in the city map and 2D points in the camera view. Given the approximate correspondence, the method computes accurate camera orientation and position in the city map, and accurate mapping between the positions in the city map and camera view. It can obtain the best-fit solutions even though the user-specified correspondence is inaccurate.

Both quantitative tests and practical application test have been conducted to assess the performance of the algorithm.

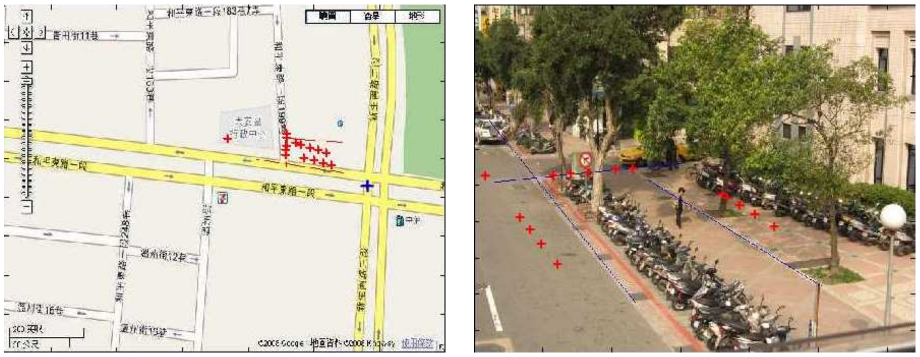

(a) (b)

Figure 9: Camera localization with linear constraints. (a) 2D-to-3D position mapping results and (b) 3D-to-2D position mapping results. Blue lines indicate the lines on which the points are supposed to lie.

Quantitative test results show that the camera localization algorithm can converge rapidly even in the presence of large amounts of inaccuracy in the 3D positions. Moreover, the algorithms are accurate and robust in camera localization and position mapping. In practical application tests, the 2D positions can be mapped very accurately after camera localization. 3D position mapping has some errors but their relative positions are still correct. By combining the camera localization results of multiple runs, one may be able to reduce the influence of noise and user inconsistency to further improve the accuracy of camera localization and position mapping. These test results show that the method proposed in the paper has great potential for camera localization and position mapping in practical applications.

Acknowledgment

This research is supported by National Science Council, Taiwan, under Grant NSC-97-2221-E-147-005.

9. REFERENCES

[1] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-D points sets. *IEEE Trans. PAMI*, 9(5):698–700, 1987.

[2] M. Betke and L. Gurvits. Mobile robot localization using landmarks. *IEEE Trans. on Robotics and Automation*, 13(2):251–262, 1997.

[3] J.-Y. Bouguet. Camera Calibration Toolbox for Matlab, www.vision.caltech.edu/ bouguetj/calib_doc/.

[4] R. Collins, A. Lipton, H. Fujiyoshi, and T. Kanade. Algorithms for cooperative multisensor surveillance. *Proc. of the IEEE*, 89(10):1456–1477, 2001.

[5] R. Cucchiara. Multimedia surveillance systems. In *Proc. ACM VSSN*, 2005.

[6] P. Debevec and H. Hoberman. Viewfinder. interactive.usc.edu/viewfinder/.

[7] Globsl positioning system, en.wikipedia.org/wiki/gps.

[8] I. Haritaoglu, D. Harwood, and L. S. Davis. W^4 real-time surveillance of people and their activities. *IEEE Trans. PAMI*, 22:809–830, 2000.

[9] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proc. IEEE CVPR*, 1997.

[10] Y. Hel-Or and M. Werman. Absolute orientation from uncertain data: A unified approach. In *Proc. IEEE CVPR*, pages 77–82, 1992.

[11] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour. Cloed-form solution of absolute orientation using orthonormal matrices. *J. Optical Society of America A*, 5(7):1127–1135, 1988.

[12] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Trans. on SMC, Part C*, 34(3):334–352, 2004.

[13] K. Josephson, M. Byrod, F. Kahl, and K. Astrom. Image-based localization using hybrid feature correspondences. In *Proc. IEEE CVPR*, 2007.

[14] P. Newman, J. Leonard, J. Neira, and J.Tardos. Explore and return: Experimental validation of real time concurrent mapping and localization. In *Proc. Int. Conf. Robotics and Auto.*, pages 1802–1809, 2002.

[15] F. Qureshi and D. Terzopoulos. Surveillance in virtual reality: System design and multicamera control. In *Proc. IEEE CVPR*, 2007.

[16] E. G. Rieffel, A. Girgensohn, D. Kimber, T. Chen, and Q. Liu. Geometric tools for multicamera surveillance systems. In *Porc. IEEE Int. Conf. on Distributed Smart Camera*, 2007.

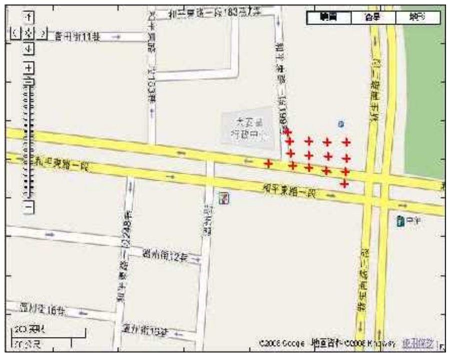
[17] S. Se, D. Lowe, and J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Trans. on Robotics*, 21(3):364–375, 2005.

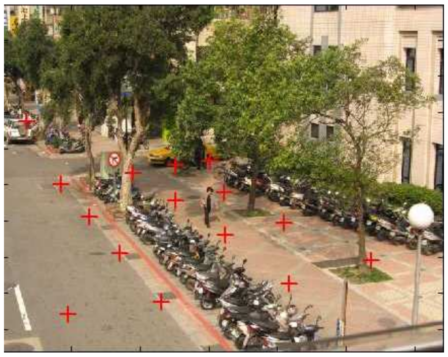
[18] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *Int. J. of Robotics Research*, 19(11):972–999, 2000.

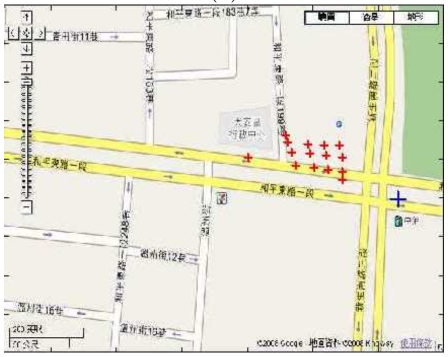
[19] R. Y. Tsai. A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE J. Robotics and Automation*, 3(4):323–344, 1987.

[20] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. PAMI*, 13(4), 1991.

[21] Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. PAMI*, 22(11):1330–1334, 2000.


(a)


(b)


(c)

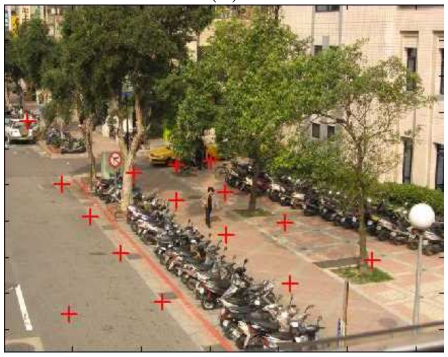

(d)

Figure 10: Mapping positions between city map and camera view. Corresponding points (red crosses) are manually marked in (a) city map and (b) camera view. (c) 2D-to-3D mapping results. Blue cross: recovered camera position. (d) 3D-to-2D mapping results.

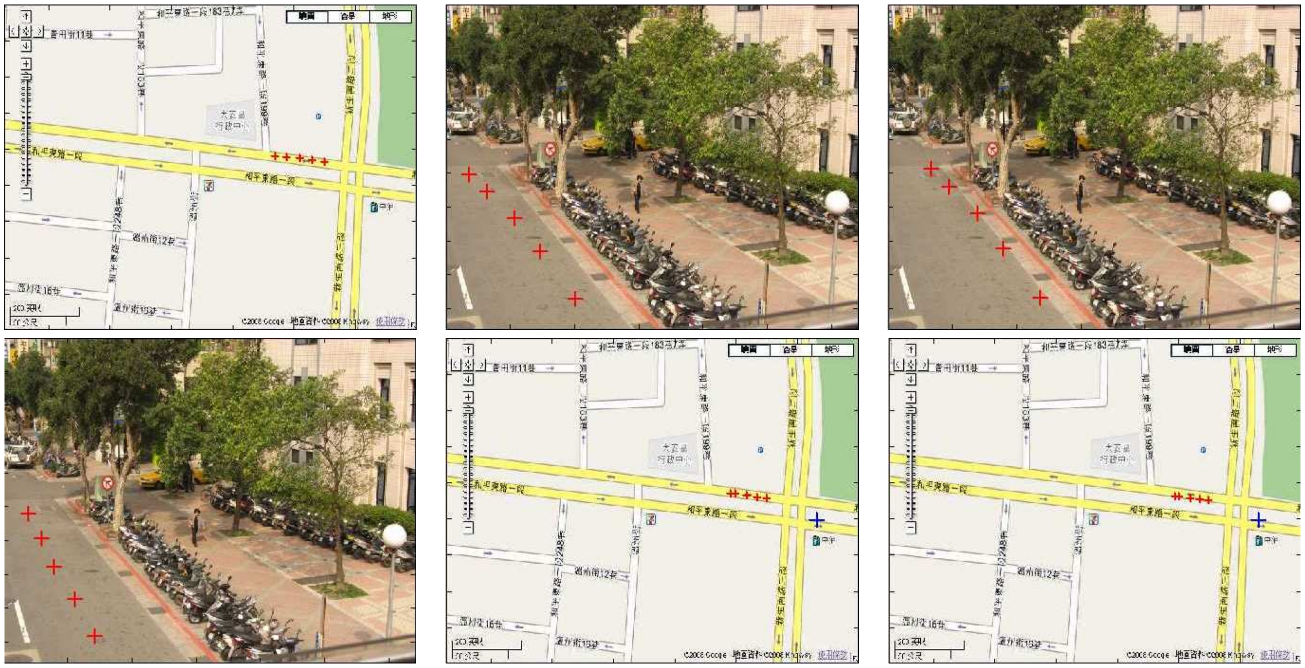

(a) (b) (c)

Figure 11: Comparison of position mappings. (Top row) 3D-to-2D mapping. (Bottom row) 2D-to-3D mapping. (a) Input points. (b, c) Mapping results using camera parameters obtained in, respectively, first and second tests.