# A STUDY ON INDOOR NAVIGATION BY AUGMENTED REALITY AND DOWN-LOOKING OMNI-VISION TECHNIQUES USING MOBILE DEVICES

[1]*Meng-Yuan Hsieh* (謝孟原) *and* [2]*Wen-Hsiang Tsai* (蔡文祥)

[1]Institute of Multimedia Engineering
[2]Department of Computer Science
National Chiao Tung University, Hsinchu, Taiwan
Emails: among0424@gmail.com, whtsai@cis.nctu.edu.tw

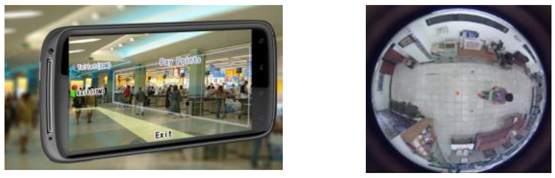
## ABSTRACT

An indoor navigation system based on augmented reality (AR) and computer vision techniques by the use of a mobile device like an HTC Flyer or an iPad is proposed. An indoor vision infra-structure is set up by attaching fisheye cameras on the ceiling of an indoor navigation environment. For human localization, a vision-based localization technique is proposed, which analyzes images captured from the fisheye cameras, and detects human activities in the environment. Furthermore, three techniques are integrated together to conduct human orientation detection effectively. A path planning technique for use to generate a path from a spot to a selected destination via the use of an environment map is also proposed. Finally, the navigation information is overlaid onto the image shown on the mobile device to provide an AR navigation interface. Good experimental results are also presented to show the feasibility of the proposed system.

***Keywords:*** *indoor navigation, augmented reality, human localization.*

## 1. INTRODUCTION

When people visit new places or complicated indoor environments, there usually needs a navigation system to guide them to desired destinations. Also, with the advance of mobile devices like iPhones and iPads, it is convenient to use them as display tools for implementing such navigation systems. In this study, it is desired to design an indoor navigation system based on *augmented reality* (AR) and *computer vision* (CV) techniques using mobile devices as display tools, as illustrated by Fig. 1.



(a) (b)

Fig. 1: Proposed AR-based indoor navigation system using computer vision techniques

In recent years, more and more researches have been conducted to satisfy the demand of indoor environment navigation for various applications. Lukianto, et al. [1] proposed an indoor navigation system for use on a smart phone, which is based on an inertial navigation system (INS) and provides the position, speed, and orientation of the user. Ozdenizci, et al. [2] proposed a near field communication (NFC) based system, which detects a user's position by touching NFC tags with a smart phone. Besides the sensor-based navigation systems just mentioned, several image-based systems using image processing techniques have also been proposed. The most common technique used in such systems is *marker-based navigation* [3, 4], in which a user must orient a phone camera to "look" at a marker for the system to recognize it and know accordingly where the user is located. Many other systems also use image-based techniques for AR. Werner et al. [5] proposed a method to detect human positions in indoor environments by image processing using a distance estimation algorithm and the camera built in a mobile device. Hile and Borriello [6] developed an indoor navigation system that can find the camera pose by detecting a landmark appearing in the image captured by a phone camera, matching it with previously-learned landmarks, and overlaying relevant information onto the image.

We propose in this study a new AR navigation system using a mobile device for indoor environments. The image of tour sites visited in the navigation process is captured by the built-in camera installed on the mobile device held by the user. The tour sites in the image are *augmented* by their respective names before being displayed on the screen of the mobile device, as shown in Fig. 1(a). Such a type of AR is accomplished by human localization implemented by analyzing images taken from fisheye cameras affixed on the ceiling in the navigation environment as well as the magnetic field information at the visited spot under the assumption that there is only one user in the environment as shown in Fig. 1(b). At first, a sufficient number of fisheye cameras are installed on the ceiling in the environment. Then, an environment map model is created, which includes the information to be used in the navigation stage. Also, while the images captured with the cameras are analyzed by a server system to detect the user's location, the user's orientation is detected by integrating three different techniques. A path planning technique for indoor navigation is proposed as well, which is based on the use of a floor plan of the indoor environment. Specifically, the image of the floor plan is analyzed to localize walkable regions in the environment, and the resulting information is then used to plan a collision-avoiding path for each navigation session.

Finally, the server system sends the obtained navigation information to the client, which includes the name and distance of the tour sites in sight of the user and the planned navigation path. Such navigation information then is overlaid onto the real tour sites shown in the current image taken of the environment by the camera built in the device. In other words, the device displays the navigation information in an AR way.

In summary, the goal of this study is to develop an indoor navigation system with the following capabilities: (1) working in indoor environments, and being able to detect a user’s location and orientation; (2) integrating real images with virtual augmentations to provide users convenient navigation information; (3) planning a proper path from a user’s location to a desired destination, and changing the planned path dynamically when the user moves to a location not in the path.

The details of the proposed AR-based indoor navigation system will be introduced in the following, with the configuration of the proposed system and the system process described in Section 2, the learning of an indoor environment in Section 3, the user localization and orientation detection methods in Section 4, the path planning technique in Section 5, the proposed AR techniques in Section 6, and some experimental results in Section 7, followed by conclusions in the last section.

## 2. IDEAS OF PROPOSED METHODS AND SYSTEM DESIGN

### 2.1 System Design

As shown in Fig. 2, the proposed system is constructed to be of a client-server structure. The server system runs on a centralized computer, and is connected to the fisheye cameras on the ceiling through a local area network. The server sends the navigation information to the client system running on the user’s mobile device. On the other hand, when the user enters the environment, the client system on the user’s mobile device is connected to the server system through a network and receives relevant information from the server.

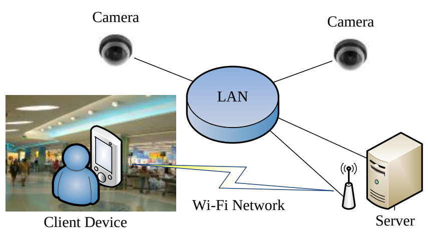


Fig. 2: Architecture of proposed system.

### 2.2 Learning Process

The system operations include two stages — a *learning* process and a *navigation* process. The goal of the learning process is to establish an *environment map*, which includes information about the tour sites, cameras, magnetic fields, and obstacle orientations in the environment. After the map is established, an obstacle analysis process is conducted, by which the proposed path planning algorithm can determine how to avoid the obstacles. Next, the system cameras, including the server-side fisheye cameras and the client-side on-device camera, are calibrated to map space or image points between different coordinate systems. Furthermore, the system detects the user’s orientation by aid of an orientation sensor built in the mobile device, which measures the azimuth angle of the device by detecting the changes and disturbances in the magnetic field around the currently-visited spot. However, the magnetic field is always interfered by the structural steel elements in buildings, so the magnetic field does not have an identical distribution at every location. To learn the magnetic field in the environment to solve this problem, we establish an *azimuth map*, which then can be used for human orientation detection.

### 2.3 Navigation Process

In the navigation process, the server accesses the images captured by the cameras in each navigation cycle, and analyzes them to detect the user’s location and orientation. It then sends the user’s location, orientation, and the information of nearby tour sites to the user’s client system, which then uses them to “augment” the image acquired of the scene of the currently-visit spot and display the result on the mobile device screen. Furthermore, when the user wants to reach a certain destination, the server will plan a path from the user’s location to the destination and display it on the screen of the mobile device.

In short, when a user enters an environment monitored by the proposed system, the client system on the user’s mobile device receives relevant information from the server and then displays the information on the screen of the device for viewing by the user.

## 3. LEARNING OF ENVIRONMENTS

### 3.1 Construction of Environment Map

The environment map is created from an image of the digital floor plan drawing, which we call the *floor plan image*, like that shown in Fig. 3(a). The image is a grayscale bitmap in which “walkable” regions are drawn with white pixels. The environment map includes the information of the cameras, tour sites, obstacles, and magnetic fields. The camera information includes the position of each fisheye camera and its height. The camera height is utilized in the transformation between the *image coordinate system* (ICS) and the *global* *coordinate system* (GCS). A tour site is a place or object of interest. The information of a tour site includes its name, its location in the GCS, and its *range* which represents the visible region of the site and is specified as a vertical plane described by a height and a width.

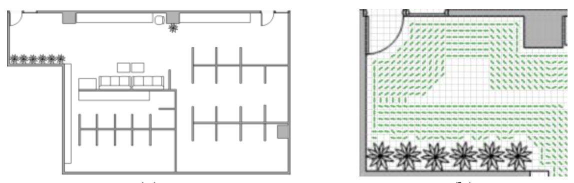


Fig. 3: Construction of environment map and avoidance map. (a) Floor plan image of experimental environment. (b) Part of constructed obstacle avoidance map (shown by green arrows).

### 3.2 Obstacle Analysis

In the process of obstacle analysis, we separate the walkable region from the other part of the image by a connected component labeling algorithm. Then, we apply edge detection to the walkable region and compute the orientation of each obstacle as the direction of the edge of the obstacle. Then, according to the following concept: when one encounters an obstacle, he/she avoid it by going left or right, we define the *avoidance direction* at each spot as the vector perpendicular to the obstacle orientation. In this way, we can construct an *obstacle avoidance map*, like the one shown in Fig. 3(b), which is finally used for necessary obstacle avoidance in the path planning process (described later).

### 3.3 Learning of Magnetic Field Information

One way we propose to determine the human orientation is to use the readings of the orientation sensor built in the mobile device, which measures the azimuth angle of the device with respective to the by detecting changes in the magnetic field around the currently-visited spot. However, according to our experimental experience, the measured azimuth values are not sufficiently stable for the navigation application due to indoor interferences from various sources.

To overcome this difficulty, in the proposed magnetic field learning process we measure the azimuth values of *four pre-specified directions*, denoted by vectors (1, 0), (0, 1), (−1, 0), and (0, −1), in the environment map at each selected spot in the environment. Data obtained at these spots are used to construct an *azimuth map*, which then are used for human orientation detection by an interpolation technique (details described later).

### 3.4 Fisheye Camera Calibration and Ground Point Location Mapping

For fisheye camera calibration, we propose a space-mapping method modified from Chen and Tsai [7] for transformations between the GCS and the ICS. At first, we construct a “concave” *calibration box B* and set up a *calibration coordinate system* (*CACS*) as shown in Fig. 4(a). Each corner point on the “inner” faces of the box is called a *calibration point*, who 3D coordinates in the CACS are pre-measured. Next, we affix a fisheye camera to a ceiling spot above the box and looking downward to capture a *calibration image I*. Then, we can obtain a mapping table from the ICS to the CACS by finding all calibration points in *I* and mapping the image coordinates to the pre-measured 3D coordinates of all the calibration points.

In order to perform ground point location mapping, at first, the mapped calibration points of the *vertical* boards of the calibration box *B* are projected onto the *x*-*y* plane of the CACS as shown in Fig. 4(b). The relation between the coordinates ($C_x$, $C_y$, $C_z$) of each calibration point *C* in the CACS and the coordinates ($C_x'$, $C_y'$, 0) of the corresponding projection point $C'$ on the *x*-*y* plane, according to the principle of similar triangles, may be described by $H_c/C_z = C_y'/(C_y' - C_y) = C_x'/(C_x' - C_x)$ which may be simplified to get

$$C_x' = (H_c \times C_x)/(H_c - C_z);\; C_y' = (H_c \times C_y)/(H_c - C_z) \quad (1)$$

where $H_c$ is the *z*-coordinate of the camera affixed above the calibration box *B* in the CACS.

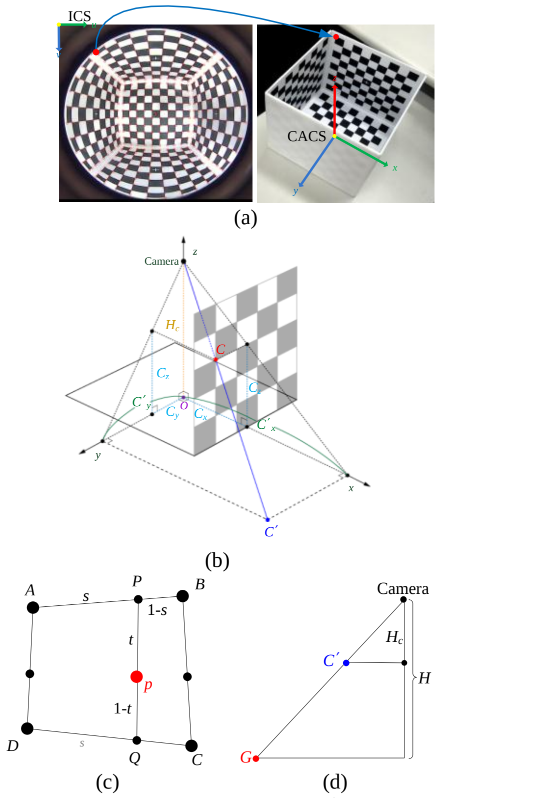


Fig. 4: Mapping between the ICS and the CACS of a calibration point. (a) The calibration box and the CACS. (b) Projection of a point in CACS on the (*x*, *y*) plane. (c) Calculating the coordinates of a point between calibration points by bilinear interpolation. (d) Projection of calibration point on ground, where *G* is the projection point of *C'*, and *H* is the height of the camera affixed on the ceiling.

So far, we have a mapping from the ICS to the CACS just for the calibration points only. For other points between them, we apply a bilinear interpolation scheme to get the corresponding mappings from the ICS to the CACS. Three major steps are conducted for this aim. First, as shown in Fig. 4(c), using the image coordinates of the calibration points $A$, $B$, $C$, and $D$ at the four corners of a grid, and those of a point $p$ within the grid, the following equations are set up according to the bilinear interpolation principle:

$$P = A + (B - A)s = A(1-s) + Bs\,;$$
$$Q = D + (C - D)s = D(1-s) + Cs\,;$$
$$p = P + (Q - P)t = P(1-t) + Qt\,,$$

where each of the notations $A$ through $D$, $P$ and $Q$, and $p$ is regarded as a 2D vector of image coordinates of the corresponding image points (e.g., $p = [p_u,\ p_v]^T$). The last equation above for $p$ may be transformed into the following form using the first two ones:

$$p = (1-s)A + st(A - D + C - B) - t(A - D) \qquad (2)$$

which represents

$$p_u = (1-s)A_u + st(A_u - D_u + C_u - B_u) - t(A_u - D_u);$$
$$p_v = (1-s)A_v + st(A_v - D_v + C_v - B_v) - t(A_v - D_v).$$

The two equations above may be solved to obtain the value of the parameter $t$ as

$$t = \left(-b \pm \sqrt{b^2 - 4ac}\right) / 2a$$

where

$$a = (A - D) \otimes (C - B);$$
$$b = (p - A) \otimes (A - D + C - B) + (A - D) \otimes (B - A);$$
$$c = (p - A) \otimes (B - A)$$

with $\otimes$ means the cross product operator (e.g., $[U_u, U_v]^T \otimes [V_u, V_v]^T = U_u \times V_v - U_v \times V_u$) Then, $s$ may be solved from (2) using the solution for $t$ above. Finally, we can substitute the corresponding CACS coordinates for $A$, $B$, $C$, and $D$ and the values of $s$ and $t$ into Eq. (2) to obtain the CACS coordinates of $p$.

At last, we want to transform the calibration points further from the CACS to the GCS to create a mapping from the ICS to the GCS for use in human localization (described later). As shown in Fig. 4(d), each fisheye camera is affixed at the ceiling at a height of $H$, and the $x$-$y$ plane of the CACS is parallel to the ground plane. Let $G$ denote the projected ground point of a point $C'$ on the $x$-$y$ plane of the CACS with coordinates ($C_x'$, $C_y'$, 0) in the CACS. Because by the principle of similar triangles, we have $d = (H \times H_c)/d_c$. Therefore, the GCS coordinates ($G_x$, $G_y$) of $G$ can be derived to be:

$$\begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \dfrac{H \times H_c}{C_x'} \\ \dfrac{H \times H_c}{C_y'} \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \end{bmatrix} \qquad (2)$$

where $p_x$ and $p_y$ are the coordinates specifying the location of the camera in the GCS, and $\theta$ is the angle between the CACS axis and the GCS axis.

## 3.5 Mobile-Device Camera Calibration

Besides the fisheye camera, the camera on the mobile device must also be calibrated in order to get a projection matrix, by which points in the GCS can be projected onto the mobile device screen. For this, a perspective camera model is used to represent the camera on the mobile device; and according to perspective projection properties [8], a 3D space point at coordinates ($p_x$, $p_y$, $p_z$) in a symmetric view frustum (as shown in Fig. 5) of the *camera coordinate system* (CCS) can be transformed to be an image point in the ICS with coordinates ($q_u$, $q_v$) described by:

$$\begin{pmatrix} p_x' \\ p_y' \\ p_z' \\ p_w' \end{pmatrix} = \begin{pmatrix} \frac{h}{w}\cot\frac{\alpha}{2} & 0 & 0 & 0 \\ 0 & \cot\frac{\alpha}{2} & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix}; \qquad (3)$$

$$\begin{pmatrix} q_u \\ q_v \\ q_z \end{pmatrix} = \begin{pmatrix} w & 0 & 0 \\ 0 & h & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.5 & 0 & 0 & 0.5 \\ 0 & -0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \end{pmatrix} \cdot \begin{pmatrix} p_x'/p_w' \\ p_y'/p_w' \\ p_z'/p_w' \\ 1 \end{pmatrix} \qquad (4)$$

where $w$ is the image width in the unit of pixel, and $h$ is the image height. The values $n$ and $f$ of the projection matrix in Eq. (3), which are respectively the nearest and the farthest visible distance of the view frustum, do not affect the coordinates in the ICS. Actually, it is used to represent the relationship of the distance of a point with respect to the viewpoint. Therefore, we can specify the values of both $n$ and $f$ arbitrarily. Finally, we have only one unknown variable $\alpha$, which specifies the angle of the field of view in the $y$ direction.

We propose in this study a simple method to estimate the value of $\alpha$. We direct the camera to face toward the calibration board, which is drawn of the form of a grid pattern or others as long as it can help us measure the region. Then, we try to use the camera to "observe" the region in such a way that the region exactly fills the image. Accordingly, we can obtain the maximum visible height $H$ by counting the grids or scale drawn on the calibration board. The distance $D$ of the calibration board to the camera is assumed to be known in advance, so the angle $\alpha$ of the field of view in the $y$ direction can be obtained from the equation $\tan(\alpha/2) = (H/2)/D$, leading to the solution:

$$\alpha = 2\left(\tan^{-1}(H/2D)\right). \qquad (5)$$

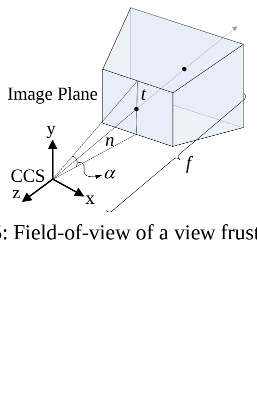


Fig. 5: Field-of-view of a view frustum.

## 4. HUMAN LOCALIZATION BY COMPUTER VISION TECHNIQUES

### 4.1 Human Location Detection

The first step of the proposed human location detection scheme is background/foreground separation. For this, we capture a background image before running the system. Then, when a user enters the environment, he/she will be considered as part of the foreground region. We obtain the human body region by finding the connected components in the foreground image, which is obtained by subtracting the background image from the acquired image. In order to find the foot point of the human in the region, we use a property of the fisheye camera, that is, with a fisheye camera affixed on the ceiling and looking downward at the ground, a space line which is perpendicular to the ground will appear in the omni-image taken by the camera as a *radial line* passing through the image center, as shown in Fig. 6. Accordingly, the human body axis will go through the image center because the user is standing on the ground during the navigation session. Consecutive, the foot point of a user can be obtained by finding the *nearest* point in the human body region of the user *to* the image center.

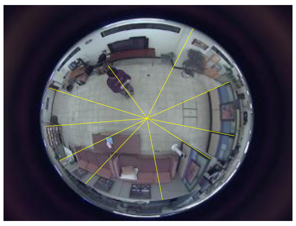


Fig. 6: Extended image lines of space lines which are perpendicular to the ground will pass through the image center.

More specifically, at first we find the nearest point to the image center in the human body region. Then, the point is projected to the line going through the human's body axis. Finally, the projected point is transformed into the GCS using the spatial transformation described in Section 3.4 to get the user's location. An example of the results is shown in Fig. 7.

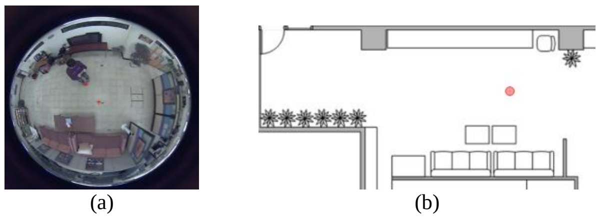


Fig. 7: Detected foot point of a human (shown as red circle). (a) Detected foot point in image (c) The foot point in the environment map.

### 4.2 Human Orientation Detection by Human Motions

We detect the user's location in the omni-image acquired in every navigation cycle, by which we can obtain a sequence of human locations. Then, we use the sequence to compute the human motion vectors. However, because the locations are detected by the aforementioned image-based technique, the path composed of these locations may be not smooth enough. To solve this problem, one way is to compute more reliable motion vectors by averaging all the motion vectors obtained within a certain time interval.

However, this averaging operation will undesirably delay the orientation decision process when the human is turning. We solve this problem by making a decision about a turning direction after multiple detections of an identical direction are achieved. As shown in Fig. 8, each *current* point is on the left-hand side of the *previous* motion vector, so we consider it as a point on a turning path, and then use only its previous motion vector as the user's current orientation.

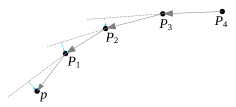


Fig. 8: A path of turning to the left where each human foot point is on the left-hand side of the previous motion vector.

### 4.3 Human Orientation Detection by Orientation Sensor on Mobile Device

When the user is not walking, we seek another way to detect the orientation, which is through the use of the orientation sensor on the mobile device. The orientation sensor measures the azimuth angle of the device by detecting changes and disturbances in a magnetic field. In the learning state, we have established an azimuth map for orientation detection. In the navigation stage, we choose the nearest learned spot to the user's location. Then, we obtain an azimuth from the orientation sensor and compute the user's orientation by the interpolation of the learned azimuth angles.

### 4.4 Human Orientation Detection by Color Edge Mark on Top of Mobile Device

The two orientation detection techniques described previously still have stability and precision problems, which cause failures in detecting the human orientation. Specifically, detection by human motions can only be used when the human moving, and detection by the orientation sensor is not stable enough due to magnetic interferences almost everywhere in indoor environments. We solve these problems by the idea of attaching a color edge mark on the top of the mobile device. The color of the color edge mark has high saturation and high lightness, so it can be segmented easily from the input omni-image. As shown in Fig. 9, we can see the green edge color edge mark clearly in the omni-image.

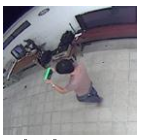
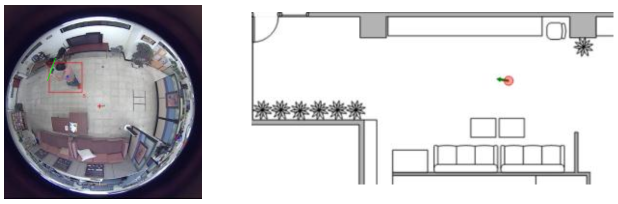


Fig. 9: The color edge mark (the green strip) in the omni-image.

Specifically, the color edge mark becomes a strip shape in the omni-image, so we can apply a line approximation scheme to the detected color edge mark.

Then, we transform the two end points of the approximating line segment into the GCS using the technique described Section 3.4. The transformed end points and the real end points of the color edge mark will be projected onto identical image points. Under the assumption that the user holds the device horizontally, the color edge mark becomes parallel to the ground. Therefore, we can determine the orientation of the color edge mark by the vector of the two transformed end points. We will get two orientation vectors which are perpendicular to the direction vector of the color edge mark, and we have to determine which one is correct. Here we use the orientation vector detected by the technique described in Section 4.3 to make a decision by choosing the one which is closer to it. Fig. 10 shows an example of the results of orientation detections using the color edge mark.



(a) (b)

Fig. 10: Orientation detection by color edge mark. (a) The approximating line of the color edge mark (shown as a green line) (b) The detected orientation (shown as a green arrow).

### 4.5 Human Tracking

We assume that there is only one user in the indoor environment taken care of by the system. Therefore, if we detect a color edge mark in a foreground region, then we can consider it is the user's region. However, if the color edge mark is invisible in the image, we choose the largest region as the user's region as a solution to this problem.

## 5. PATH PLANNING FOR NAVIGATION

We use an *obstacle image* obtained in the learning stage to determine whether a planned path collides with any obstacle or not. If a path starting from the location of a user and ending at the location of a destination collides with an obstacle, we have to determine an immediately neighboring point to avoid the obstacle. Here we use the obstacle avoidance map constructed in the learning stage for this purpose. We follow the avoidance direction to find the immediately neighboring point at each spot to obtain a collision-free path starting from the user's location to the destination. More details are described in the following.

### 5.1 Obstacle Avoidance

Each element in the obstacle avoidance map represents two opposite avoidance directions. The planned path "walks" a block to reach a new spot at a time. We assign three avoidance blocks for each avoidance direction. As shown in Fig. 11, *each region of avoidance directions* (shown as semi-transparent regions) consists of three blocks, which include one *primary* avoidance block (shown as red regions) and the two neighborhoods, which are called *secondary* avoidance blocks (shown as blue regions).

We can find the next immediately neighboring point in the three avoidance blocks by the avoidance direction. If we want to reach a destination from the current position, we find the avoidance blocks of the position at first. For each avoidance block, we check whether it can be reached from the position. If a block can be reached, we add the center point, which is called *avoidance point*, of the block into a *resulting set*. The set contains all the reachable avoidance blocks finally, and at that time it is sorted by the *priorities* of the avoidance points. The priority of an avoidance point is assigned according to the distance from the avoidance point to the destination point. The avoidance point of a primary avoidance block has the highest priority. An avoidance point with a higher priority should be considered first as the next immediately neighboring point in the path finding process. The path finding process scheme will be described in the next section.

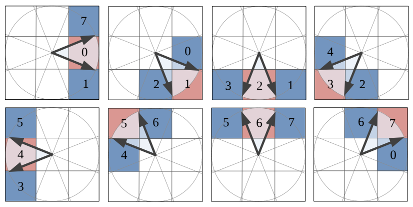


Fig. 11: Avoidance blocks of 8 avoidance directions. Each avoidance direction is assigned three blocks, which include one primary avoidance block (shown as red regions) and two secondary avoidance blocks (shown as blue regions).

### 5.2 Path Finding

When we want to find a path from the current position to a destination position, we find the avoidance points of the current position at first. Then we begin to check each avoidance point. We record it when we check an avoidance point, and we do not check the same avoidance point twice, meaning that a path does not visit an identical point twice. If an avoidance point has not been visited before, we try to find a path from this avoidance point to the destination. Therefore, the process can become recursive. However, if an avoidance point cannot reach the destination, we continue to try the next avoidance point. If there is no more avoidance point, we finish the finding process. The following algorithm always returns a path starting from an input start point and ending at a destination if a path is found; otherwise, it returns a flag indicating a failure.

**Algorithm 1: Path finding.**

**Input:** A start point $p$ in the map coordinate system (MCS), a final destination point $d$ in the MCS, and a set $S_w$ which includes the points we have visited before.

**Output:** A set of points consisting of a found path $S_{result}$, and a flag $f$ indicating whether a path is found.

**Steps.**

```
1. Initialize an empty set $S_{result}$ to store the output points.
2. Add $p$ into $S_{result}$ and $S_w$.
3. If $p$ can directly go to $d$ without colliding obstacles,
   add $d$ into $S_{result}$ and then go to Step 9.
4. Find the avoidance points by the technique described
   in Section 5.1, resulting in a set of avoidance points $S_a$.
5. Take out the first avoidance point $p_a$ in $S_a$.
6. If $p_a$ is contained in $S_w$, then
     if there still is any avoidance point in $S_a$, go to Step
     5; otherwise, set $f$ to fail and finish this algorithm.
7. Apply this algorithm recursively with the inputs $p_a$, $d$,
   and $S_w$, resulting in a set of points of a found path $S_{ad}$
   and a flag $f_a$.
8. If flag $f_a$ is a success, add each point of $S_{ad}$ into $S_{result}$;
   otherwise, go to Step 5.
9. Set $S_{result}$ as output and set $f$ to be a success.
```

### 5.3 Path Simplification

After a path is found, we begin to conduct a path simplification process, which consists of two parts: *redundant point elimination* and *distance reduction*. The goal of the *redundant point elimination* is to find two points which are non-connected and can instead be connected together, and then to remove the points between the two points; in other words, the goal is to find a "shortcut" between two points (like the red line shown in Fig. 12(a)). The goal of *distance reduction* is to find a "shortcut" between two line segments. As shown in Fig. 12(b), we can reduce the total length of the path by substituting $P_2$ with two new immediately neighboring points (shown as red points). Fig. 13 shows a result of the path simplification process.

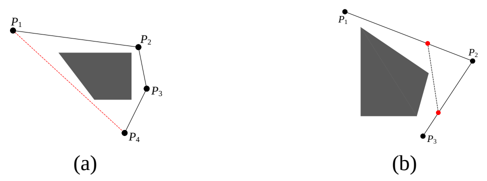


Fig. 12: Path simplification process. (a) Redundant point elimination: two redundant points $P_2$ and $P_3$ can be removed. (b) Distance reduction: path length can be reduced by substituting $P_2$ by a shortcut.

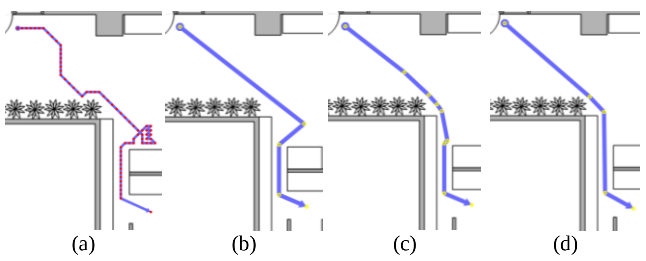


Fig. 13: Result of path simplification. (a) Result of path finding. (b) Result of applying redundant point elimination on (a). (c) Result of applying distance reduction on (b). (d) Final result.

### 5.4 Path Update

A user might not always follow a planned path in a navigation session. We have to update a planned path when a user walks away from the planned path. To update a planned path (as shown in Fig. 14(a) and (b)), we find the last reachable point (shown as red circles) of the planned path from the current point (shown as green circles).

After a path is updated, it contains only one new point, which is the current point of the user, and the remainders in the path are the points of the originally planned path Therefore, we have to check whether the resulting new line segments are of the simplest form or not. Therefore, we apply the path simplification process to the two line segments (as shown in Fig. 14(d)) and obtain a simplified path.

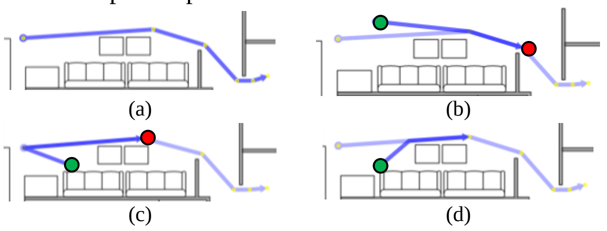


Fig. 14: Results of path update process. (a) Original planned path. (b) Updated path. (c) An updated path which is not of the simplest form. (d) Result of applying path simplification on path of (c).

## 6. AUGMENTED REALITY FOR NAVIGATION

We overlay navigation information onto the real images taken of the current scene to get augmented reality effects. Real images taken of the current scene will be called "*scene images*" subsequently, and scene images overlaid with navigation information will be called "*augmented images*." At first, we will describe the mapping between the real world scene and the displayed image on the screen of the mobile device. And then, we will describe the proposed method for displaying the navigation information.

### 6.1 View Mapping between Real World Scenes and Client Device Displays

In order to display the information in an AR way, the client system transforms the GCS coordinates onto a 2D screen plane. A point $p$ in the CCS can be transformed to be a point $q$ in the ICS by Eq. (3). Here we have to transform the coordinates from the GCS into the CCS at first. The transformation can be expressed by the following equation:

$$p = M_c \cdot [a_x\ a_y\ a_z\ 1]^T \tag{6}$$

where $a$ is a point in the GCS, $p$ is the transformed point of $a$ in the CCS, and $M_c$ is a transformation matrix. The transformation matrix can be expressed as follows:

$$M_c = \begin{pmatrix} right_x & up_x & -forward_x & -c_x \\ right_y & up_y & -forward_y & -c_y \\ right_z & up_z & -forward_z & -c_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7}$$

The left three columns *up*, *right*, and *forward* are actually the three orthonormal basis of the CCS. As shown in Fig. 15(a), *up* is the up direction of the camera, *right* is the right direction, and *forward* is the front direction. The rightmost column is the coordinates of the camera, which is actually the user's location detected by the server system. The *z*-coordinate of the camera position is a predefined parameter; in other words, the height of a camera is fixed to about the height of the eyes of an adult in the proposed system.

Therefore, we can determine the transformation matrix $M_c$ by finding the three vectors *up*, *right*, and *forward*. The *up* direction of the GCS is the +*z* direction, so the vector *up* is (0, 0, 1). Also, we assume that the camera orientation is in the same direction of the user's orientation. Therefore, the direction $d = [d_x\ d_y]^T$ of the camera on the *x*-*y* plane, which is the human orientation, can be obtained from the server system by the methods described in Sec. 4.2 through 4.4. On the other hand, as shown in Fig. 16, the camera is tilted for a pitch angle $\beta$, which can be obtained from the orientation sensor of the client device. Therefore, we can obtain the *z* direction of the camera by the following equation:

$$d' = [d_x\ d_y\ \sin\beta]^T. \tag{8}$$

And then *forward* can be obtained by normalizing *d'*. The last vector *right* can be obtained by:

$$r = forward \otimes up;\ right = r/|r|. \tag{9}$$

Finally, we correct the *up* by the cross product of *right* and *forward* to make the three vectors to be orthogonal. Now, we have obtained all the needed variables to perform the transformation described by (6) and (7).

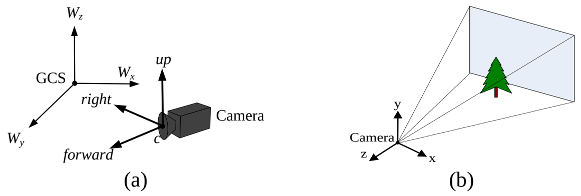


Fig. 15: A camera in the GCS and the CCS. (a) A camera in the GCS with three orthonormal vectors *up*, *right*, and *forward*. (b) The CCS.

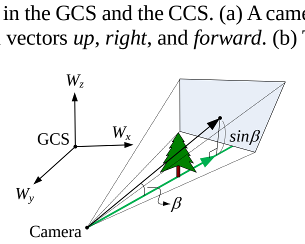


Fig. 16: Camera looks at a pitch angle $\beta$. The green line indicates a line on the horizontal plane.

### 6.2 Rendering for Displaying Names of Tour Sites

To create AR effects, we try to overlay the name and the distance of tour sites onto the corresponding objects in the scene image on the mobile device. As shown in Fig. 17(a), a tour site is defined by four parameters $\overline{f}$, *p*, *w*, and *h*. Then, the four corner points of the region of a tour site can be computed and transformed into the ICS. Then, as shown in Fig. 17(b), we clip the resulting coordinates to the range of the image size. Finally, the position of the display text can be obtained by finding the centroid of the four transformed points.

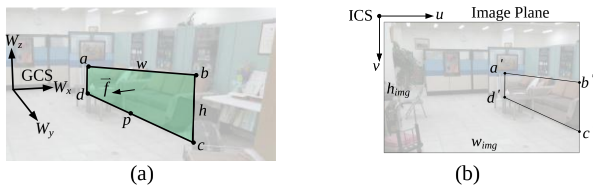


Fig. 17: Transformation of the GCS of a tour site into the ICS. (a) Parameters of a tour site. (b) After transforming into the ICS and clipping to the range of the image size.

### 6.3 Rendering for Displaying Navigation Path

It is decided in this study that only two line segments of a path will be displayed at a time on the mobile device screen. After a user receives a path sent from the server system, the first two line segments are what we are going to display. The displayed path is composed of thick line segments and an arrow. It is a 3D augmented object which can be rendered by the OpenGL API; therefore, we just have to compute the vertices of the geometric shape of the path.

## 7. EXPERIMENTAL RESULTS

The environment map of the environment where our experiments were conducted is shown in Fig. 18, which is an open space in a lab, including eight tour sites (shown as green regions) and two fisheye cameras (shown as blue circles). The first experimental result shown here is a series of images resulting from browsing the surrounding tour sites at a certain location in the environment. A user stood at the location as shown in Fig. 19(a), and the augmented images shown on the user's mobile device are shown in Fig. 19(b)-(f).

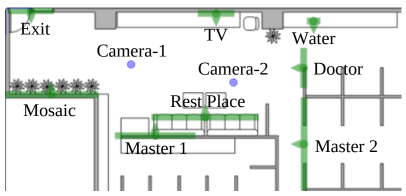


Fig. 18: The environment map of the experimental environment.

Next, we show a result of navigation according to a navigation path. A user stood at a location and searched the environment map for a tour site, and there appeared accordingly a yellow "stroke text" on the right-hand side of the bottom edge of the augmented image as shown in Fig. 20(a). The user could then understand that the destination is on the right rear, so the user began to turn to the right-hand side and saw the destination as well as the navigation path when he turned to the correct direction (as shown in Fig. 20(b)). Therefore, the user began to follow the navigation path to move. Fig. 20(c) and (d) show the two augmented images corresponding to two different locations in the user's path of movement.

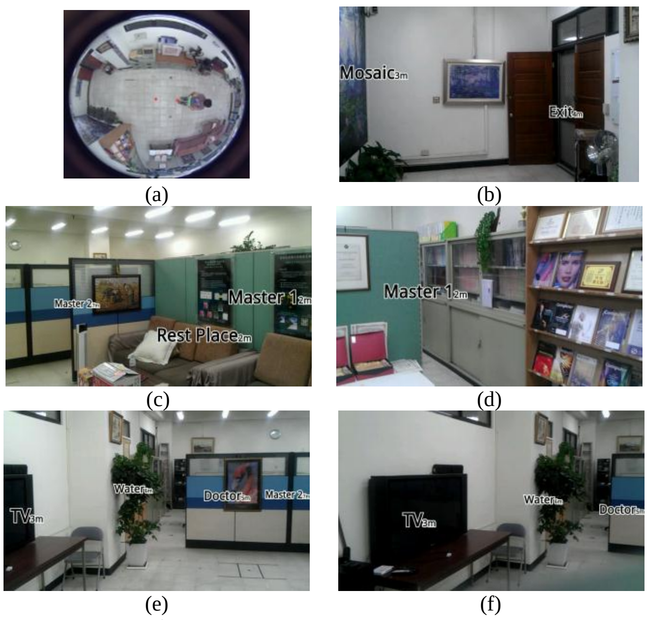


Fig. 19: A result of browsing tour sites at a certain location. (a) Place where the user stood at. (b)-(f) The series of resulting augmented images.

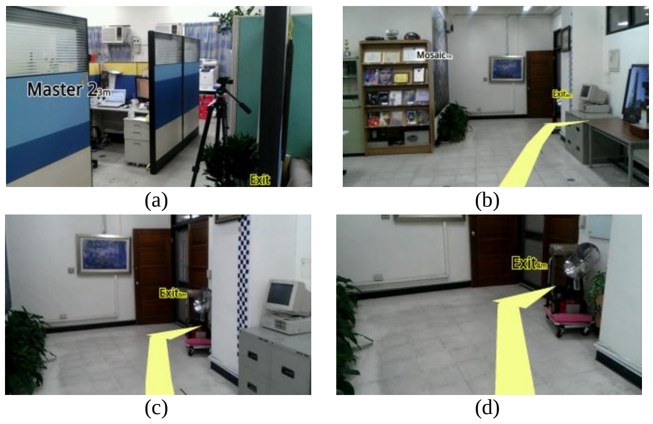


Fig. 20: A result of navigation following a planned path. (a) Augmented image shown when the user searched a tour site. (b) Augmented image seen by the user when facing the correct direction. (c)(d) Two augmented images corresponding to two different locations on the user's path.

## 8. CONCLUSIONS

In this study, an indoor navigation system constructed by augmented reality and down-looking omni-vision techniques using mobile devices has been proposed. To design such a system, several techniques have been proposed , including: 1) a modified method for point transformation from an omni-image into the global coordinate system; 2) a method for human localization in indoor environments; 3) a method for path planning for indoor environments; 4) a method for indoor AR navigation by overlaying tour site information on the real objects in scene images; 5) a method for indoor AR navigation by overlaying a navigation path on the floor in a scene image. The experimental results shown in the previous section have revealed the feasibility of the proposed system.

## REFERENCES

[1] C. Lukianto, C. Honniger, and H. Sternberg, "Pedestrian Smartphone-Based Indoor Navigation Using Ultra Portable Sensory Equipment," in *Proceedings of International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Zurich, Switzerland, 2010, pp. 1-5.

[2] B. Ozdenizci, K. Ok, V. Coskun, and M. N. Aydin, "Development of an Indoor Navigation System Using NFC Technology," in *Proceedings of Fourth International Conference on Information and Computing (ICIC)*, Phuket Island, Thailand, 2011, pp. 11-14.

[3] L. C. Huey, P. Sebastian, and M. Drieberg, "Augmented Reality Based Indoor Positioning Navigation Tool," in *Proceedings of IEEE Conference on Open Systems (ICOS)*, Langkawi, Malaysia, 2011, pp. 256 - 260.

[4] A. Mulloni, D. Wagner, D. Schmalsteig, and I. Barakonyi, "Indoor Positioning and Navigation with Camera Phones," *Pervasive Computing, IEEE*, vol. 8, pp. 22-31, 2009.

[5] M. Werner, M. Kessel, and C. Marouane, "Indoor positioning using smartphone camera," in *Proceedings of International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Guimaraes, Portugal, 2011, pp. 1-6.

[6] H. Hile and G. Borriello, "Positioning and Orientation in Indoor Environments Using Camera Phones," *IEEE Computer Graphics and Applications*, vol. 28, pp. 32-39, 2008.

[7] H. C. Chen and W. H. Tsai, "Optimal security patrolling by multiple vision-based autonomous vehicles with omni-monitoring from the ceiling," in *Proceedings of 2008 International Computer Symposium*, Taipei, Taiwan, Republic of China, 2008, pp. 196-201.

[8] T. Akenine-Moller, E. Haines, and N. Hoffman, "Pespective Projection," in *Real-Time Rendering*, Third Edition, T. Akenine-Moller, ed., 2008, pp. 92-97.