AUTOMATIC GUIDANCE FOR INDOOR CAR PARKING USING AUGMENTED-REALITY AND OMNI-VISION TECHNIQUES[†]

[1]*Jair Chen* (陳頡) *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: pplinlin2@gmail.com, whtsai@cis.nctu.edu.tw

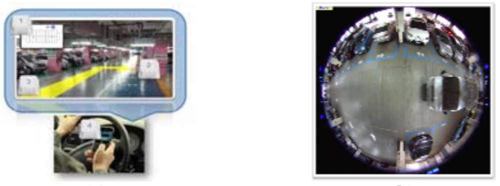
ABSTRACT

An augmented reality (AR)-based car parking guidance system is proposed to help a driver to save parking time, which has the functions of finding empty parking spaces automatically, planning an optimal path to an empty space, and displaying the path in an AR way on a mobile device to guide the driver to reach to the selected space. To detect and track a car driven in the parking lot, a car localization method by uses of 3D car-shape bounding boxes as well as the constraint information of the car speed and turning curvature is proposed. To find empty parking spaces in the parking lot, a dynamic environment learning technique and a parking-space detection method based on the use of 3D bounding boxes again are proposed. For path planning, the Dijsktra algorithm which yields a shortest path from the current car location to a selected empty parking space is employed. To guide the driver, also proposed is an integrated method for generating a perspective-view image from a fisheye image, on which the planned path is augmented. Finally, good experimental results showing the feasibility of the proposed system are also included.

Keywords: *automatic guidance, augmented reality, car tracking, empty parking-space detection, fisheye image, mobile device*.

1. INTRODUCTION

Finding an empty space to park a car in a parking lot is often a problem, especially in a crowded metropolitan area. A feasible solution is to design a method to conduct the functions of finding empty parking spaces *automatically* and guiding the driver to a selected space in an *augmented reality* (*AR*) way. The function may be implemented as an APP which is installed on a mobile device like an iPhone or an iPad. And the AR information may be designed to be the shape of a directed arrow representing a path leading to a destination parking space. The mobile device may be held by the driver or a passenger in the car, or even by a user outside the car before the car is driven into the parking lot, as illustrated in Fig. 1. To keep safe driving, the mobile device may also be affixed firmly in front of the driver for convenient observation. By following the augmented guiding arrow on the mobile-device screen, the driver can drive the car to reach the selected empty parking space. A system with these capabilities will be of great help to a driver to save parking time. In this study, it is desired to design such an automatic system for empty parking-space finding and AR-based driving guidance in a parking lot.



(a) (b)

Fig. 1: Proposed AR-based guidance system for car parking. (a) An illustration. (b) An image of a parking lot used in proposed system.

There are more and more researches about vehicle guidance systems using various techniques recently. These systems adopted different types of car positioning and AR techniques to meet the requirements for vehicle navigation. About the car positioning technique, Chen and Tsai [1] proposed a park-area navigation guidance system using circular sidewalk landmarks in the park; and by using the omni-image taken with an omni-camera placed on top of the vehicle, the vehicle position is computed as the top-view distance from the circular-shape center to optical center of the omni-camera. Wei and Tsai [2] proposed a system for the similar purpose by detecting line features on along-path buildings in omni-images and compute the vehicle position by line-feature matching using a longest common subsequence algorithm. Betke and Gurvits [3] proposed a car localization method by identifying landmarks in car surrounds to find their locations on an environment map. Wu and Tsai [4] proposed a method of using various combinations of line features to localize the vehicle. Recently, Kurz, et al. [5] introduced the first publicly available test dataset with ground truth values for outdoor handheld camera localization. About the AR technique, Hsieh and Tsai [6] proposed an indoor navigation system displaying the AR navigation information with real-world images taken by the camera on the mobile device. Kim and Dey [7] proposed the idea of AR windshield displaying.

About parking-space detection technique, Blumer et al. [8] detected vacant parking spaces by a single perspective camera using edge and color information and constructed the regions of parking spaces manually. Lixia and Dalin [9] determined the vacancy of a parking space based on image segmentation and local binary patterns,

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

providing that the pixels of the parking space are known in advance. And Shih and Tsai [10] proposed a method for the same purpose by detecting lines in omni-images directly using an elliptical-curve fitting technique.

In this study, we try to develop an AR-based guidance system with the above-mentioned functions for use by a car driver or a passenger after entering a parking lot, which has the following capabilities: 1) locating the car in a parking lot to inform the driver of his/her current position; 2) finding empty parking spaces automatically from omni-images acquired with fisheye cameras on the ceiling; 3) showing the found parking spaces on a map displayed on the mobile device for the user to select manually, or for the system to choose automatically; 4) planning a shortest navigation path to guide the driver to the selected space; 5) displaying the planned path on a mobile device in an AR manner (showing augmented arrows leading to the found parking space), by which the driver/user can inspect and drive accordingly to reach the destination space.

The details of the proposed system will be introduced in the remainder of this paper, with the configuration of the proposed system and the system process being described in Section 2, the process of "learning" an parking-lot environment presented in Section 3, a newly-proposed car detection method proposed in Section 4, a parking-space detection method and a path planning process for guidance described in Section 5, an AR-based method for generating a perspective-view image for displaying on a mobile-device screen proposed in Section 6, and some experimental results presented in Section 7, followed by some conclusions in the last section.

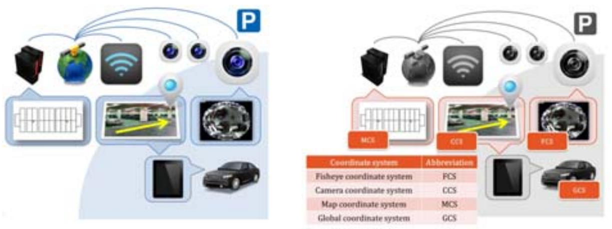
2. SYSTEM DESIGN AND PROCESSES

2.1 System Design

As shown in Fig. 2(a), at first we affix a number of fisheye cameras on the ceiling of the parking lot where the proposed system is to be applied. The cameras are used to acquire images of the parking-lot environment, from which *augmented images* displayed on the mobile-device screen is constructed. Also, the system is constructed to be of a client-server structure. The server, which is located at a remote site and connected to the fisheye cameras through an Ethernet, is designed to conduct the works of planning a path from the car location to a selected empty parking space and sending the navigation information to the client which is run on the user-held mobile device. When a driver enters the parking lot, the client is connected to the server through the network, and starts to receive the navigation information from the server and display augmented images on the mobile device.

2.2 Learning Process

The software operations of the proposed system include two processes – *learning* and *navigation*. The first goal of the learning process is to establish an *environment map*, which includes the background knowledge such as locations of the parking spaces and paths in the parking lot required for the navigation process. The second goal is to construct space-mapping tables for use in generating the augmented image displayed on the mobile device. Especially, as shown in Fig. 2(b), four coordinate systems are used in this study and three space-mapping tables are constructed to transform image points between different coordinate systems to generate perspective-view images of the parking lot environment from which augmented images for displays on the mobile device are created.



Coordinate system	Abbreviation
Fisheye coordinate system	FCS
Camera coordinate system	CCS
Map coordinate system	MCS
Global coordinate system	GCS

(a) (b)

Fig. 2: Hardware architecture of proposed system.

2.3 Navigation Process

There are four main tasks in the navigation process, namely, 1) detection of empty parking spaces, 2) tracking of cars, 3) path planning, and 4) generation of augmented images for AR-based for car-parking guidance. In the first task, environment images are acquired by use of all the fisheye cameras and all empty parking spaces are detected for selection by the system or by the user (the driver or any passenger). In the second task, acquired fisheye images are analyzed to detect empty parking spaces and track cars in the images. The third task is path planning, by which a suitable path is generated to guide the driver from the current car location to the selected parking space. The fourth task is AR-based guidance for car parking, in which a perspective-view image with the navigation path augmented on it, namely, an augmented image, is generated and displayed. The approach adopted in this study for this purpose is to generate the augmented image using the fisheye image without using the camera on the mobile device to take images.

3. CONSTRUCTING SPACE-MAPPING TABLE

3.1 Mapping between Fisheye and Top-view Image

To detect cars in the parking lot, we need top-view images of the parking lot. For this, we conduct an image-space mapping from the *fisheye-image coordinate system* (*FCS*) to the *map coordinate system* (*MCS*), by which we can obtain a top-view image of a part of the parking lot environment from a fisheye image, i.e., we regard the obtained top-view image as part of the environment map. The entire process of space mapping can be illustrated by Fig. 3. Firstly, we take an image of a *calibration cylinder* with a grid pattern (shown on the right-hand of Fig. 3), and find the corner points of it in the image. Secondly, a space-mapping table T_F is created using a geometric model. Finally, a transformation from the fisheye image into the top-view image can be derived by looking up the space-mapping table T_F.

More specifically, a calibrate cylinder as just mentioned is constructed at first, which is shown in Fig. 4(a). Inside it is attached a chessboard-like grid pattern whose squares are of the size of $2{\times}2$ cm^2. The circles formed by the grids on the bottom are designed to have increasing radii from inside to outside. Secondly, the cylinder is placed under each of the fisheye cameras, and an image of it is taken with the fisheye camera, which is named a *space-mapping image*. An example is shown in Fig. 4(b). Thirdly, a corner detector is applied to obtain the corner points of the squares in the image, as shown as the yellow points in Fig. 4(c). Subsequently, a space-mapping function, described as a table T_F, is derived, by which we can compute the corresponding location on the top-view image of each corner point in the space-mapping image.

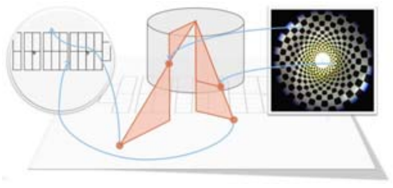
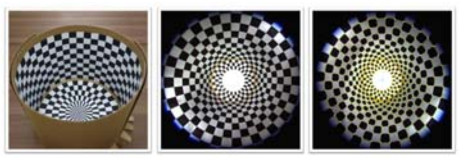
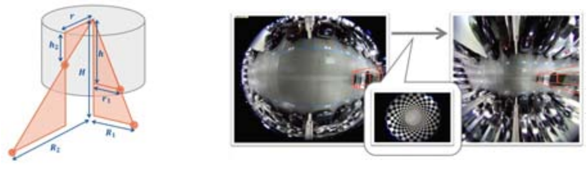


Fig. 3: An illustration of image space mapping between fisheye image and top-view image.


(a) (b) (c)
Fig. 4: An illustration of the space-mapping cylinder.

In more detail, an illustration of the space-mapping function is shown in Fig. 5(a) where the value r and h are the radius and height of the cylinder, respectively; H is the height of the fisheye camera with respect to the floor; r_1 is the distance of a corner point P_{cb} on the bottom of the cylinder to the center of the cylinder; h_2 is the distance of a corner point P_{cs} on the "vertical" side of the cylinder to the top of the cylinder which is an input known value. The output variables are the two distances R_1 and R_2 of the points P_{cb}' and P_{cs}' in the top-view image corresponding to P_{cb} and P_{cs}, respectively, as shown in the figure. According to the similar-triangle principle, we have $R_1/r_1 = H/h$ and $R_2/r = H/h_2$, leading to the results of $R_1 = (H/h)\times r_1$ and $R_2 = (H/h_2)\times r$. For non-corner points, the values of R_1 and R_2 are computed by interpolation. As a result, every point in the space-mapping image is mapped to a point in the top-view image using the formulas R_1 and R_2. This mapping can be represented as a table which is just Table T_F mentioned previously. Subsequently, given a fisheye image, we can then transform it into a top-view image conveniently by table-lookup using T_F. An experimental result of conducting such a mapping is shown in Fig. 5(b).


(a) (b)
Fig. 5: Mapping from fisheye images to top-view images. (a) An illustration of notations involved in the mapping. (b) An experimental result.

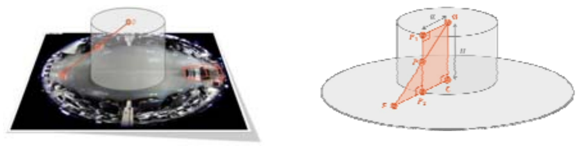
It is not difficult to figure out that the previously-described mapping is reversible so that we may construct a *backward space-mapping table* T_R from T_F, which can then be used to map a top-view image back into a fisheye image by table-lookup.

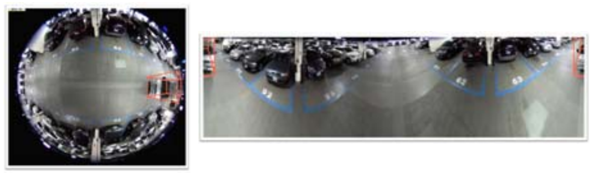
3.2 Forward Mapping from Fisheye Image to Panorama Image

To generate the previously-mentioned augmented image for displaying on the mobile-device screen, as mentioned previously a forward space-mapping method by table-lookup is proposed to transform a fisheye image into a panorama image. The detail is described now. The involved concept is illustrated by Fig. 6. At first, we imagine to put the fisheye image on the ground and create a virtual cylinder at the location of the fisheye camera which is named point O. For each pixel P on the panorama image to be constructed, we project it onto the cylinder in the direction of the vector $\overrightarrow{OP}$ to get a projection point F on the ground. We then assign the RGB values of point F to be those of point P. Once all the pixels on the cylinder are processed, transformation of the fisheye image onto the desired panorama image is completed. More specifically, according to the principle of similar triangles, we have the equality: $(FP_2 + R)/H = R/P_1P$, or equivalently, $FP_2 + R = R\times H/P_1P$. Accordingly, the position of point F can be derived by the following way:

$$F = C + (FP_2 + R)\overrightarrow{e_r} = C + (R \times H / P_1P)\overrightarrow{e_r} \qquad (1)$$

where $\overrightarrow{e_r} = \overrightarrow{OP_1}/\|\overrightarrow{OP_1}\|$, and F and C are regarded as vectors. An experimental result is shown in Fig. 7.


(a) (b)
Fig. 6: The cylinder model used in space mapping from the fisheye image to the panorama image. (a) Illustration of the mapping scheme. (b) Involved notations.


(a) (b)
Fig. 7: An experimental result of forward mapping from a fisheye image to a panorama image. (a) Input image. (b) Output image.

4. CAR DETECTION AND TRACKING BY DOWN-LOOKING FISHEYE CAMERAS

Three methods have been developed sequentially in this study to detect and track cars in the parking lot using fisheye image. Each method is an improvement of the precedent one, as described subsequently.

4.1 Car Localization by Use of 3D Bounding Boxes

A common technique in object detection is *frame differencing*, which, however, results in inaccuracy when the involved images are acquired by fisheye cameras.

There are two reasons for this. The first is geometric distortion found in fisheye images. To overcome this, mapping of fisheye images into top-view images as discussed above is adopted. The second reason is that 3D information is not used in frame differencing because the images are treated as 2D in the process. To remedy this, we introduce a type of 3D box shape, called *3D bounding box*, and use it to "bind" the car shape found in the fisheye image so that the car in the image can be detected more effectively, as described the algorithm below.

Algorithm 1. Car localization by 3D bounding box.

Input: a fisheye image I and a background image B both acquired by an identical fisheye camera.

Output: a parking-lot graph with a detected car located on it.

Steps.

1. (*Frame differencing and component labeling*) Compute the difference between input images, I and B, to create a frame-difference image I_F, find large connected components in I_F as possible car shapes, and compute the location of each component (as the average of all the points in the component).
2. (*Car location approximation*) Transform each found component with location (i, j) in the fisheye image into a component in the top-view image with location (x, y) in the environment map using the forward space-mapping table T_F as illustrated in Fig. 8(a); and regard (x, y) as the approximate location of a *car candidate*.
3. (*Finding car candidates*) Create a 2×2 chessboard (with four grid points) centered at the approximate location of each car candidate on the environment map, and put a bounding box on each grid point of the chessboard, as illustrated in Fig. 8(b), resulting in four bounding boxes.
4. (*Selecting an optimal car candidate*) Select an *optimal* car candidate from all the four car candidates in the following way: 4.1) initialize two *backward space-mapping tables* T_{R1} and T_{R2}, where T_{R1} is the table for mapping the bottom part of the car, and T_{R2} is the table for mapping the ceiling part of the car; 4.2) regard each of the four bounding box, B, in the environment map to consist of eight points P_1~P_8 with P_1~P_4 forming the *top*, B_{top}, of the box and P_5~P_8 forming the *bottom*, B_{bottom}; 4.3) regard bounding box B as a shape in the top-view image and transform it into the fisheye image by transforming B_{top} and B_{bottom} using tables T_{R1} and T_{R2}, respectively and regarding the result as a 3D polygonal *solid* shape G; 4.4) superimpose G on the frame difference image I_F and count the number C of overlapping pixels; and 4.5) select the car candidate among the four candidate ones with the maximal number of overlapping pixels as the *optimal car candidate*, as illustrated in Fig. 9, which is then taken to be the *detected car*.
5. Draw the location (x, y) of the detected car on the parking-lot graph O as output.

4.2 Car Detection by Prediction Using Knowledge of Car Movement

The previous method localizes a car by the use of 3D bounding boxes. However, the trajectory of the computed car locations is not only discontinuous but also vibrating as found in our experimental results. The reason is that only the fisheye image is used as input. To improve the result, it is found in this study some knowledge of the physical limitation on the vehicle's movement may be utilized, as discussed in the following.

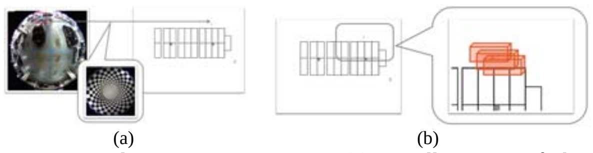


Fig. 8: Car location approximation. (a) An illustration of the process. (b) An illustration of finding car candidates using the bounding box.

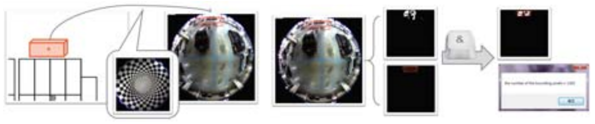


Fig. 9: An illustration of selecting an optimal car candidate.

The first limitation is the upper bound of the car speed, that is, the driver cannot drive too fast in the parking lot. Accordingly, we can make a limit which the car speed cannot exceed. For example, thirty kilometers per hour is a reasonable speed limit for driving in parking lots. The second limitation is the minimum *radius of curvature* through which the vehicle can turn. That is, when the driver turns right, the car trajectory cannot be a sharp 90^o angle; instead, it should be a smooth curve. Furthermore, when the driver turns the steering wheel to the end and moves the car forward for a while, the trajectory should be a circle; the location of the car should not appear in this circle; and the radius of this circle is just the *minimum radius of curvature* found in the car turning trajectory. This knowledge is used in this study for improving the last method for car localization. As a summary, the movement of the car is restricted by the upper bound of the car speed and the minimum radius of curvature in car turning. As found in this study, these restrictions cause the reachable range of the car movement *within an image processing cycle* (i.e., during the time duration between acquisitions of two image frames) to form an area with the shape of a *gingko leaf* as shown in Fig. 10(a), called a *gingko-shaped prediction area* in this study, as discussed in the following.

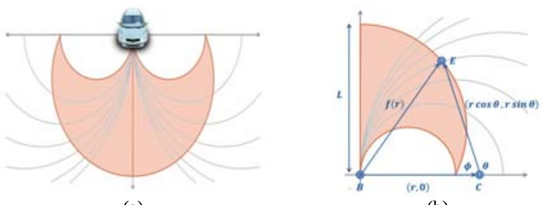
At first, we derive the equation of the boundary of this area as a proof of its shape — a gingko, which helps us to decide whether an input point is within this area or not. Referring to Fig. 10(b) for the notations, let r be the *radius of curvature* by which a car is driven to turn, and let $f(r)$ specify the corresponding limitation of how far the car can move forward in an image processing cycle. In other words, $f(r)$ specifies the scope of the gingko-shaped prediction area. Let L be the maximum distance that a car can move in an image processing cycle, which may be decided in advance by the upper bound of the car speed and the frame rate of the camera. Then, $f(r)$, when regarded as a vector, may derived to be

$$\begin{aligned} f(r) &= \overrightarrow{BC} + \overrightarrow{CE} \\ &= (r, 0) + (r\cos\theta, r\sin\theta) \end{aligned}$$

$$= (r(\cos\theta + 1), r\sin\theta) \quad (2)$$

where $\theta = \pi - \phi = \pi - [L/(2\pi \times r)] \times 2\pi = \pi - (L/r)$ so that (2) becomes

$$f(r) = (r(1+\cos(\pi - \frac{L}{r})), r\sin(\pi - \frac{L}{r}))$$
$$= (r(1-\cos(L/r)), r\sin(L/r)) . \quad (3)$$



(a) (b)

Fig. 10: Computation of gingko-shaped prediction area. (a) The limitation of car moving resulting in a gingko-shaped area. (b) An illustration of notations involved in proposed method.

Now, given an input point P on the environment map with coordinates (x, y), we want to check if it is in the gingko-shaped prediction area. For the answer to be positive, it must be true that P can be specified by the function $f(r)$ with a certain radius r as illustrated in Fig. 10, or equivalently, P must be on a circle with its center C being $(r, 0)$ and its radius being r which is the distance from point P to C. That is, we have the equality $r = \|(x, y) - (r, 0)\|$ which may be solved to get $r = (x^2 + y^2)/(2x)$. Then, with $r_{\min}$ denoting the minimum radius of curvature of car turning and ϕ denoting the angle between vectors $\overrightarrow{CB}$ and $\overrightarrow{CP}$, it is not difficult to figure out that if r satisfies the two conditions $r > r_{\min}$ and $r\phi < L$, then the input point P is in the gingko-shaped area. If this is true, then we can regard P to be reasonable for use as the approximate location of the car on the environment map. Otherwise, we change the approximate location by adding the distance of $L/2$ (unit of the orientation vector) to the current location of P and take the result as the location of a suitable candidate for the car. This approximate car location is then taken as input to perform Algorithm 1, starting from Step 2. By the way, it is noted that the meanings of the above two conditions are: (1) $r > r_{\min}$ means that the radius of car-turning curvature is in the reasonable range; (2) $r\phi < L$ means that the distance for which the car moves forward in an image processing cycle is smaller than L.

4.3 Car Tracking by Uses of Track Continuity and Virtual Fences

The method of car detection by prediction as described in the previous section yields a more precise location of the vehicle. However, the trajectory is not smooth, either, as seen in the results of the experiments we have conducted. The reason causing this problem is that the continuity of the car orientation has not yet been taken into consideration. If a car turns to the left in the first frame, it cannot turn to the right immediately in the next frame. In other words, dramatic changes in car turning should not happen. The mentioned selection of car candidates should be modified so as to avoid this problem. Specifically, given an input car location P at map coordinates (x, y), P must be on a trajectory within the gingko-shaped prediction area as shown in Fig. 10, and the trajectory is a portion of a circle with equation $(x - r)^2 + y^2 = r^2$. Taking a differentiation of the equation leads to $2(x - r) + 2y(dy/dx) = 0$ which may then be solved to get the orientation of point P as $dy/dx = (r - x)/y$. This restriction on the orientation of the car location may be utilized in a modified version of Step 4 of Algorithm 1 to make the selection of the optimal car candidate to be more effective, as is done in this study.

Furthermore, to have a start point of detection and tracking, a *3D virtual fence* is used in this study. Specifically, in order to detect a car appearing in the fisheye images for the first time after it enters the parking lot, the 3D virtual fence is implemented essentially as a 3D bounding box like the one described previously in Sec. 4.1 for detecting cars. After being detected, a car will then be tracked continuously by the system.

Some experimental results of applying various modified versions of Algorithm 1 taking into considerations of the above-discussed issues and their solutions are shown in Fig. 11. The upper-left image shows the output car trajectory which results from frame differencing only, the middle-left image shows the output trajectory of proposed car localization using 3D bounding boxes, and the lower-left image shows the output trajectory of proposed car detection by prediction. The right-side image is the output of proposed car tracking by uses of track continuity and virtual fences. As can be seen from the figure, the precision of the detection and tracking process gets higher and higher as the three remedy methods are applied sequentially, resulting in a smooth car trajectory finally.

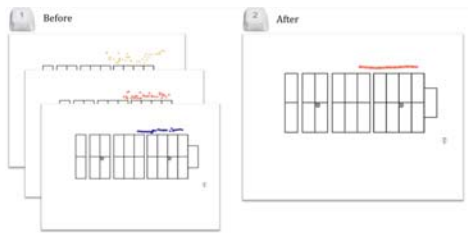


Fig. 11: Experimental results of the three proposed methods with a comparison of them.

5. PARKING-SPACE DETECTION AND PATH PLANNING

5.1 Parking-space Detection

Background subtraction is commonly used as the first step of detecting objects in an image, but it has two disadvantages if a *static* background is used. The first is that it requires all the objects (cars in this study) that we want to detect *not* to appear in the background image. However, this is not easy to satisfy by a parking lot usually full of parked cars. The second disadvantage is that the background may change during the car detection process. For example, the brightness of a scene in the parking lot may not be identical from morning to afternoon. Therefore, it is impractical to use only a static background which cannot vary when the surrounding condition is changing. To avoid these disadvantages of using a static image as the background, a dynamic environment learning technique is proposed for use in empty parking-space detection, as described in the following algorithm.

Algorithm 2. Dynamic environment learning and empty parking-space detection.

Input: a fisheye image I and a list L_{location} keeping the locations of the parking spaces in a parking lot.

Output: an environment map O with the empty parking spaces marked.

Steps.

1. (*Initialization*) Assign the input image I as the initial environment background image B and create a list L_{empty} for recording the emptiness condition of each parking space in the parking lot.
2. (*Dynamic environment learning*) Acquire a new image I' of the environment with each fisheye camera, and analyze the image content to check whether the environment covered by the camera (including the car spaces and the lighting condition) has changed too much; if so, update the background image B to be I'.
3. (*Detection by use of 3D bounding boxes*) For each parking space S in the list L_{location}, check in I' whether S is empty or not currently; if so, update the emptiness condition of the corresponding element in an *emptiness-condition list* L_{empty}.
4. Check each parking space in list L_{empty}, and if it is empty, then mark it up on the environment map O.
5. Repeat Steps 2~4 until the system is shut down.

The main part of the proposed dynamic environment learning technique as described above is updating of the background image B in every image processing cycle. This is necessary if the environment has changed too much. For example, when a car is parked in a parking space or when it is driven away later, the image should be updated. Fig. 12 shows an example of applying the proposed process of empty parking-space detection using the 3D bounding box.

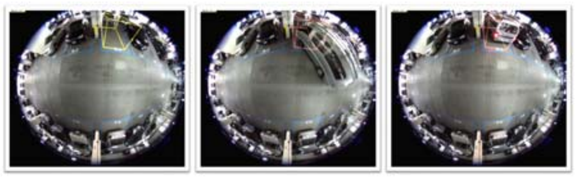


Fig. 12: An example of empty parking-space detection.

5.2 Path Planning for Parking Guidance

By the proposed system described so far, not only the locations of the car and the empty parking spaces have been obtained but also the locations of the paths in the parking lot are assigned in advance. Therefore, the most suitable method for planning a navigation path for an in-lot driver to follow is the *single-source shortest path* algorithm. With no doubt, the most well-known shortest path algorithm is the Dijkstra algorithm. The navigation path from the entry of the parking lot to a selected empty parking-space can be obtained efficiently by applying the Dijkstra algorithm, as is conducted in this study.

6. AR-BASED GUIDANCE FOR CAR PARKING

6.1 Generation of Parking Guidance Image

In this study, the planned path is drawn on a perspective-view image which looks like the scene seen by the driver through the front window from his/her viewpoint. An easy way to achieve this is to draw the navigation path on an image taken with the camera built in the mobile device. However, requiring the driver to take images with the mobile device will endanger him/her while driving. In order to generate a perspective-view image without using the mobile-device camera, the only image that we can use is the fisheye image captured with the fisheye camera affixed on the parking-lot ceiling. In this study, a two-stage method is proposed to achieve this goal. The first stage is *navigation path rendering* and the second is *perspective-view image generation*.

In the first stage, the navigation path is drawn onto the fisheye image. For this, a fisheye image captured with a camera with its scope covering a moving car is mapped into a top-view image I_{T} using the forward mapping table T_{F} mentioned in Sec. 3. Then, the navigation path generated by the path planning algorithm mentioned in Section 5 is drawn onto I_{T} to create a new top-view image I_{T}'. Finally, I_{T}' is mapped back into a fisheye image I_{F} using the backward mapping table T_{R} mentioned in Sec. 3 as well. And in the second stage, a perspective-view image is created from I_{F} by mapping I_{F} into a panorama image I_{P}, which is then cut properly to result in a perspective-view image I_{v} as output whose sightline is the same as that of the driver. The image I_{v} is finally displayed on the mobile-device screen for the driver or a passenger to inspect. An illustration of the above steps of the proposed AR-based guidance process for car parking is shown in Fig. 13, and some details of the process are described in the following.

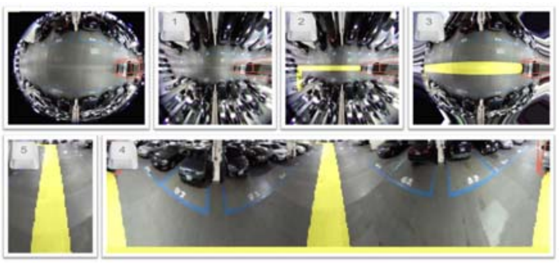


Fig. 13: An illustration of the guidance image generation process.

6.2 Perspective-view Image Generation

In the above perspective-view image generation process, the driver's viewpoint should be decided. For this, at first it is noted that a panorama image can be constructed in the form of a virtual cylinder from a fisheye image, and we can get the perspective-view image via this cylinder. This concept, mentioned before in Sec. 3.2, is illustrated in Fig. 14(a) from the need of computing the driver's viewpoint. In order to simplify the problem, we transform the cylinder model into a circle model as illustrated in Fig. 14(b). The circle model is the top-view of the cylinder model. Every 3D sightline of the driver in the cylinder model will form a 2D sightline in the circle model. The mathematical property of the 2D sightline is introduced in detail in the following to derive the driver's viewpoint.

Firstly, some notations are introduced. As illustrated by Fig. 15(a) which shows the circle model, R is the radius of the panorama image cylinder, O is the location of the fisheye camera, C is the location of the car, and $\overrightarrow{L_C}$ is the location vector of the car such that $\overrightarrow{L_C} = \overrightarrow{OC}$. Now, the driver's view may be seen to be contributed by many sightlines which are shown in orange color in Fig. 15(a). One line is taken without loss of generality and shown in Fig. 15(b) where $\overrightarrow{O_C}$ is the directional vector of the sightline, A is the intersection point of the sightline and the panorama image, and θ is the azimuth angle of the vector $\overrightarrow{OA}$.

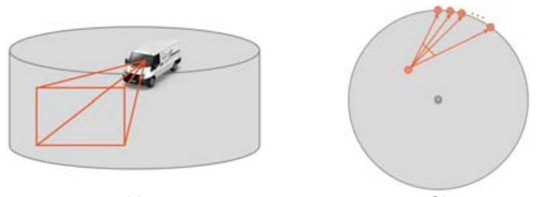

(a) (b)

Fig. 14: An illustration of perspective-view image generation.

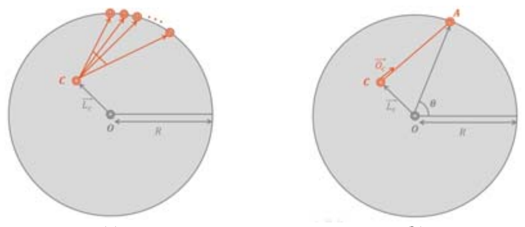

(a) (b)

Fig. 15: The notations used in the circle model for deriving the driver's viewpoint.

The values of the variables R and O are known in advance. Given the input C which is the location of the car and the vector $\overrightarrow{O_C}$ which specifies the orientation of the car, we want to decide the value of the azimuth angle θ of the vector $\overrightarrow{OA}$, which defines the direction of the viewpoint A of the driver from the location O of the camera. At first, suppose $\overrightarrow{OA} = \overrightarrow{L_c} + n\overrightarrow{O_c}$ where n is a value to be determined, and let the length of this vector be $\|\overrightarrow{OA}\| = R$. Then, it is the equation $\|\overrightarrow{L_c} + n\overrightarrow{O_c}\| = R$ that we should solve to get n. By taking the squares of both sides of the equation, we can get the following equality:

$$\|\overrightarrow{O_c}\|^2 n^2 + 2n\overrightarrow{L_c}\cdot\overrightarrow{O_c} + \|\overrightarrow{L_c}\|^2 - R^2 = 0\,.$$

There are two possible solutions to n. One is positive, and the other negative. We choose the positive one because it means the right direction of the sightline, i.e., we take n to be

$$n = (1/\|\overrightarrow{O_c}\|^2)(-\overrightarrow{L_c}\cdot\overrightarrow{O_c} + \sqrt{(\overrightarrow{L_c}\cdot\overrightarrow{O_c})^2 - \|\overrightarrow{O_c}\|^2(\|\overrightarrow{L_c}\|^2 - R^2)})\ ,(4)$$

where $\overrightarrow{L_c}$ and $\overrightarrow{O_c}$ may be computed by

$$\overrightarrow{L_c} = L_x\overrightarrow{e_x} + L_y\overrightarrow{e_y}\ ;\ \ \overrightarrow{O_c} = O_x\overrightarrow{e_x} + O_y\overrightarrow{e_y}\ ; \qquad (5)$$

with L_x and L_y being the x and y components of vector $\overrightarrow{L_C}$, respectively; O_x and O_y being the x and y components of vector $\overrightarrow{O_C}$, respectively, and $\overrightarrow{e_x}$ and $\overrightarrow{e_y}$ are the unit vectors of the directions of the x- and y-axis, respectively. Furthermore, we have

$$\overrightarrow{OA} = (R\cos\theta)\overrightarrow{e_x} + (R\sin\theta)\overrightarrow{e_y}\ . \qquad (6)$$

On the other hand, we have $\overrightarrow{OA} = \overrightarrow{L_c} + n\overrightarrow{O_c}$ which may be rewritten as

$$\overrightarrow{OA} = \overrightarrow{L_c} + n\overrightarrow{O_c} = (L_x + nO_x)\overrightarrow{e_x} + (L_y + nO_y)\overrightarrow{e_y}\ . \qquad (7)$$

Comparing (6) and (7), we can get two solutions for θ, namely, $\theta_x = \cos^{-1}[(L_x + nO_x)/R]$ and $\theta_y = \sin^{-1}[(L_y + nO_y)/R]$. Now, since the range of $\cos^{-1}$ is from 0 to π and the range of $\sin^{-1}$ is from $-\pi/2$ to $\pi/2$, it is possible that $\theta_x \neq \theta_y$. Therefore, we create a table, Table 1, in the following for determining a unique value for θ from θ_x and θ_y according to the properties of the quadrants.

Table 1. A truth table for computing azimuth angle θ of vector $\overrightarrow{OA}$.

$\theta_x > \pi/2$	$\theta_y > 0$	θ
True	True	θ_x
True	False	θ_y
False	False	$2\pi - \theta_x$
False	True	$2\pi + \theta_y$

7. EXPERIMENTAL RESULTS

Some experimental results of applying the proposed system are shown here. The experimental environment is a parking lot in National Chiao Tung University. There are 25 parking spaces there, and four down-looking fisheye cameras are installed on the ceiling, mainly above the main paths in the parking lot.

The left-side image in each of Figs. 16(a) through (g) is the car tracking result based on the use of car track continuity which was described in Sec. 4. The yellow 3D bounding box is the 3D virtual fence, and the red 3D bounding box binds the detected car. And the right-side images of Figs. 16(a) through (g) are the corresponding locations and the trajectories of the detected car on the environment map with the planned path also shown. The results show that a car was driven into the parking lot, and moved toward an empty parking space. Then, it was parked into an empty parking space, the one with No. 63; and the space on the map turns into gray, meaning that it is occupied. The corresponding AR-based guidance image is shown as Fig. 17. From these images, it can be seen that the proposed method works effectively.

8. CONCLUSIONS

An AR-based guidance system for car parking in a parking lot using multiple down-looking omni-cameras has been proposed. Several techniques have been proposed to implement the system as summarized in the following. 1) An integrated method for car detection, localization, tracking based on the uses of the 3D bounding box and the gingko-shaped prediction area has been proposed for finding the precise location of the vehicle. The car location yielded by the method is drawn onto a top-view map of the parking lot, on which the driver (or a passenger) may inspect to know where his/her car is located. 2) A new method for parking-space detection based on the use of the 3D bounding boxe has been proposed, by which the system can point out the positions of empty parking spaces for the driver to choose or for the system to specify automatically. 3) A new method for path planning in the parking lot has been proposed, by which the system can plan a suitable navigation path from the current car location to the selected empty parking space. 4) An integrated method has been proposed as well for generating a perspective-view image, on which the selected navigation path can be augmented. The method is based on several stages of transformations starting from a fisheye image. By following the navigation path augmented in the perspective-view image which is then shown on the mobile-device screen, the driver can be guided to reach the selected empty parking space. Good experimental results reveal the feasibility of the proposed system.

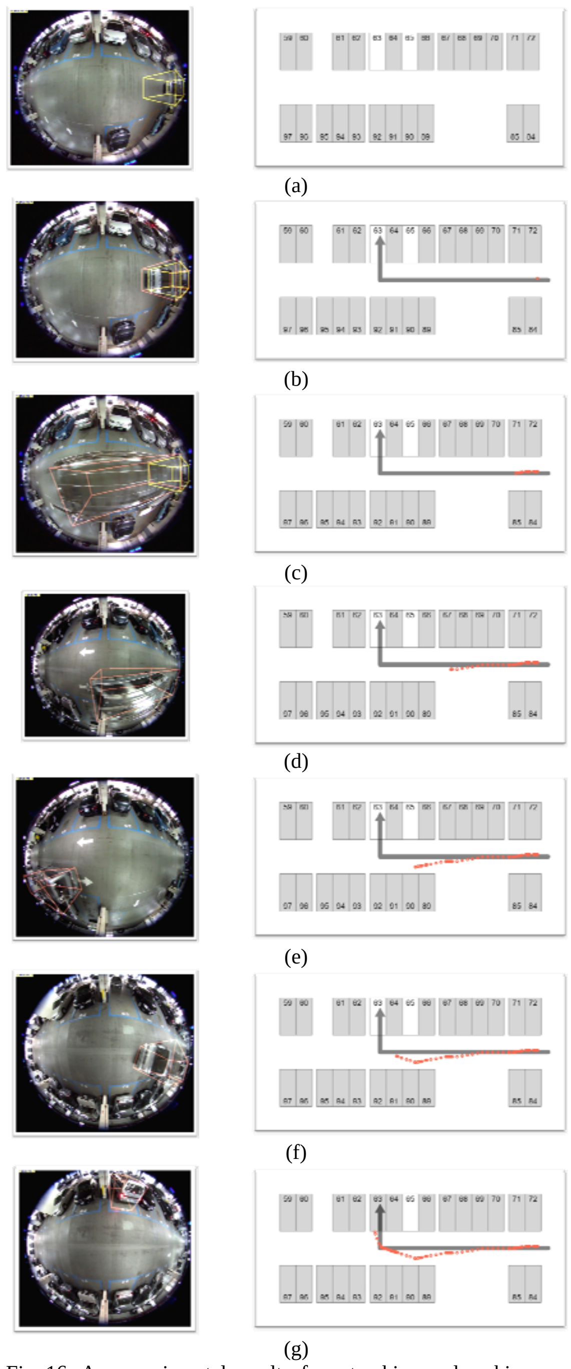


Fig. 16: An experimental result of car tracking and parking-space detection

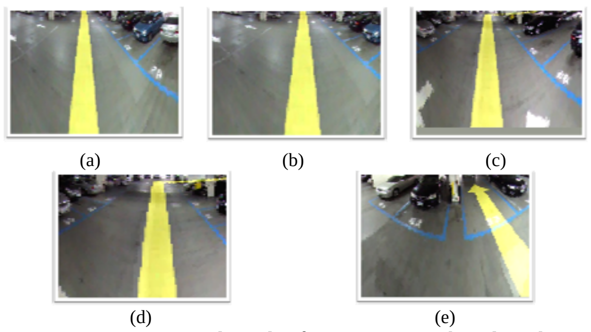


Fig. 17: An experimental result of generating AR-based guidance image for car parking.

9. REFERENCES

[1] B. C. Chen and W. H. Tsai, “A study on tour guidance by car driving in park areas using augmented reality and omni-vision techniques,” *Proc. of 2012 Conf. on CVGIP*, Nantou, Taiwan, Aug. 2012.

[2] Y. C. Wei, B. S. P. Lin and W. H. Tsai, “Augmented reality-based in-vehicle tour guidance in park areas by vertical-line features in omni-images,” *Proc. of 2013 Nat’l Computer Symp. - Workshop on Video & Image Analysis, Taichung*, Taiwan, Dec. 2013.

[3] M. Betke and L. Gurvits, “Mobile robot localization using landmarks,” *IEEE Trans. on Robotics & Automation*, Vol. 13, No.2, pp. 251-263, April 1997.

[4] C. J. Wu, “New Localization and Image Adjustment Techniques Using Omni-Cameras for Autonomous Vehicle Applications,” *Ph. D. Dissertation,* Inst. of CSE, Nat’l Chiao Tung Univ., Hsinchu, Taiwan, 2009.

[5] D. Kurz, P. G. Meier, A. Plopski, and G. Klinker, “An outdoor ground truth evaluation dataset for sensor-aided visual handheld camera localization,” *Proc. of Int’l Symp. on Mixed & Augmented Reality*, Rio de Janeiro, Brazil, pp. 263-264, May 2013.

[6] M. Y. Hsieh and W. H. Tsai, “A study on indoor navigation by augmented reality and down-looking omni-vision techniques using mobile devices,” *Pro. of Conf. on CVGIP*, Nantou, Taiwan, Aug. 2012.

[7] S. J. Kim and A. K. Dey, “Simulated augmented reality windshield display as a cognitive mapping aid for elder driver navigation,” *Proc. of SIGCHI Conf. on Human Factors in Computing Systems*, Boston, MA, USA, pp. 133-142, April 2009.

[8] K. Blumer, H. R. Halaseh, M. U. Arsan, H. Dong, and N. Mavridis, “Cost-effective single-camera multi-car parking monitoring and vacancy detection towards real-world parking statistics and real-time reporting,” *Neural Information Processing - Lecture Notes in Computer Science (LNCS)*, Vol. 7667, pp. 506-515, 2012.

[9] W. Lixia and J. Dalin, “A Method of Parking Space Detection based on Image Segmentation and LBP,” *Proc. of 4th IEEE Int’l Conf. on Multimedia Information Networking & Security*, Nanjing, Jiangsu, China, pp. 229-232, 2012.

[10] S. E. Shih and W. H. Tsai (2014). "A convenient vision-based system for automatic detection of parking spaces in indoor parking lots using wide-angle cameras," *IEEE Trans. on Vehicular Technology*, accepted and to appear.