EZCam: WYSWYG Camera Manipulator for Path Design

Cheng-Chi Li, Yu-Chi Lai, *Member, IEEE*, Nai-Sheng Syu, Hong-Nian Guo, Dobromir Todorov, and Chih-Yuan Yao, *Member, IEEE*

***Abstract*— With advances in the movie industry, composite interactions and complex visual effects require us to shoot at the designed part of a scene for immersion. Traditionally, the director of photography plans a camera path by recursively reviewing and commenting path-planning rendered results. Since the adjust–render–review process is not immediate and interactive, miscommunications happen to make the process ineffective and time consuming. Therefore, this paper proposes a What-You-See-What-You-Get camera path reviewing system for the director to interactively instruct and design camera paths. Our system consists of a camera handle, a parameter control board, and a camera tracking box with mutually perpendicular marker planes. When manipulating the handle, the attached camera captures markers on visible planes with selected parameters to adjust the world rendering view. The director can directly examine results to give immediate comments and feedbacks on transformation and parameter adjustment in order to achieve effective communication and reduce the reviewing time. Finally, we conduct a set of qualitative and quantitative evaluations to show that our system is robust and efficient and can provide means to give interactive and immediate instructions for effective communication and efficiency enhancement during path design.**

***Index Terms*— Camera path design, camera transformation manipulator, marker-based camera tracking.**

I. Introduction

GOOD camera motion can bring more immersive experience to an audience and, traditionally, a cinematographer or a director of photography (DP) must plan and adjust camera shooting paths on a trolley based on his/her experience to create desired photographic and visual effects. However, with advances in technologies, there are more complex actions, interactions, and special effects inside a shot, which brings difficulties to get a right camera path in the first trial. In other words, path adjustment results in time-consuming and expensive reshooting. As shown in Fig. 1, computer graphics can relieve reshooting expenses by building a virtual scene along with camera paths to simulate the shot. However, it is still a time-consuming task requiring several adjust–render–review iterations because the gaps between director's expectation and animators' interpretation of directing instructions may result in unsatisfied camera motion and cause extra iterations. There are generally two issues during the adjust–render–review process.

1) The number of iterations depends on the experience of animators and adequate communication between animators and the director.
2) The director cannot review the rendered results in the mean time of commenting and instructing, this may induce a gap between initial instructions and later expected results as time goes by, and the gap results in extra unnecessary iterations.

Therefore, this paper proposes an interactive, intuitive, and robust manipulation and review system to create and modify a camera path and immediately review the result for effective communication, portability, and efficiency.

When motion capture techniques become mature, camera transformation can be estimated by attaching passive sensors on the real movie camera to manipulate the shooting view [1]. This also allows the director to directly review and adjust the camera path in a similar spirit to our design. However, there are several limitations in this method.

1) The motion capture equipment is expensive, and small studios and education institutes cannot afford it.
2) Its portability is limited because the technique requires a large space and a complex calibration process to have proper operations.
3) Manipulation requires operating a cumbersome movie camera, and the movement inside the captured space is strenuous.

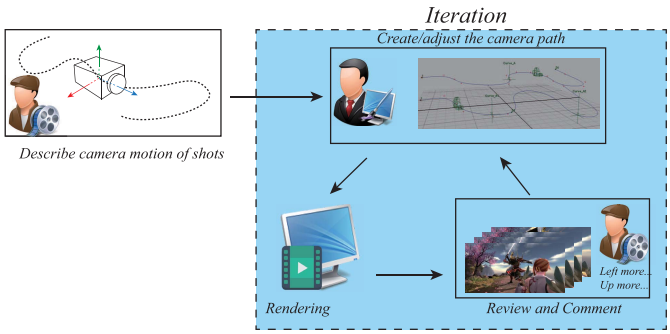


Fig. 1. Traditional computer graphics-based camera path design pipeline consists of director's path description, animators' path creation, rendering with designed paths, and director's commenting.

Manuscript received May 10, 2015; revised September 20, 2015 and December 4, 2015; accepted January 26, 2016. Date of publication March 16, 2016; date of current version August 2, 2017. This work was supported by the National Science Council of Taiwan under Grant NSC 103-2221-E-011-114-MY2, Grant NSC 104-2221-E-011-029-MY3, and Grant NSC 103-2218-E-011-014. This paper was recommended by Associate Editor Q. Tian. *(Cheng-Chi Li and Yu-Chi Lai contributed equally to this work.) (Corresponding author: Chih-Yuan Yao.)*

C.-C. Li, Y.-C. Lai, N.-S. Syu, H.-N. Guo, and C.-Y. Yao are with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan (e-mail: cyuan.yao@gmail.com).

D. Todorov is with Next Media, Inc., Taipei 114, Taiwan.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2016.2543018

Therefore, this paper proposes a low-cost, portable, and real-time interactive reviewing system based on marker recognition and tracking [2]. Furthermore, with advance in computer vision (CV), camera tracking can be done with structure from motion (SfM) techniques based on feature tracking such as parallel tracking and mapping (PTAM) [3] or photographic tracking such as semidirect visual odometry (SVO) [4]. Although these techniques do not require designed hardware, camera manipulation in path design consists of transient large transformations to induce tracking failure as shown in our evaluations. NCam [5] is developed for camera tracking, and according to our observation, NCam uses two fisheye lenses to capture the world and estimates the rough structures based on feature-based matching for tracking. It has the following limitations.

1) Feature extraction and matching are computationally costly.
2) Fisheye lenses are more expensive than a webcam.
3) Although it can work without markers, it still requires an environment of enough features.

Our hardware is cheaper and available everywhere. Furthermore, NCam technical staffs have visited Next Media Animation Japan Studio and used Easy Camera (EZCam), and they are impressed with our tracking results. Qualcomm Vuforia [6] is another commercially available augmented reality (AR) system to track camera motion by embedding square ring features into the 3D world. It tracks these features for the camera transformation. According to our evaluation, our system can more stably and robustly track camera in higher precision because Vuforia does not take advantage of the multiplane setting, and perspective artifacts may affect its precision. We overcome the robustness, portability, and efficiency issues of these methods with a knocked-down and portable tracking box consisting of perpendicular marker planes along with a camera handle for transformation manipulation. Our system directly collects the manipulated transformation along with camera parameters selected on our designed control board to render the view on a tablet and monitor for direct and immediate review.

After implementing the system, we have designed qualitative experiments to evaluate the tracking ability and precision of our system and a quantitative experiment to evaluate the manipulation ability and robustness on two testing scenes when comparing with: 1) two vision-based tracking methods, PTAM [3] and SVO [4]; 2) two depth-sensor-based tracking method, KinectFusion [21] and Difodo [20]; 3) two accelerometer-gyroscope-based tracking methods, attitude and heading reference system (AHRS) [19] and Cardboard [20]; and 4) commercially available Vuforia [6]. Furthermore, we have also conducted two user studies to evaluate the usefulness in improving design efficiency and communication effectiveness. The results show that our designed system is cheap, portable, and robust and can help construct effective communication to reduce the number of review iterations and the amount of review time.

This paper makes the following contributions.

1) We propose a What-You-See-What-You-Get (WYSWYG) review system to design camera paths for movie shooting to remove the gap between the director instructions and animator understanding for reduction of extra iterations.
2) We design camera tracking boxes of patterned marker planes to ensure robust and effective manipulation of the camera transformation and tracking efficiency. Due to the rapidness and robustness of marker recognition, we can estimate the view for a real-time renderer to display the results for discussing and adjusting the camera path in a real-time and face-to-face manner.
3) We have designed three types of knocked-down elements for portability and easy construction of the tracking box.
4) We provide a parameter control board to select and tune camera parameters for more possible photographic effects.

II. Related Works

Camera tracking has long been the research topics for decades because it is important for understanding the scenes and adding objects into the virtual world. This paper focuses on tracking camera motion to manipulate the virtual camera, and thus, the following discusses only those directly related to this research.

A. Camera Tracking

Vision-based camera tracking is important for AR, CV, and computer graphics. Fundamentally, they can be categorized as two fields: 1) marker-based tracking and 2) markerless tracking. Marker-based tracking depends on identification of designed patterns such as ARToolKit [7], ARTag [8], C^2 Tags [9], Pi-Tag [10], and error-correcting grid codes [2] to estimate the camera pose and attached plane transformation. Marker-based tracking methods are generally fast and robust. Therefore, our system chooses error-correcting grid codes for its efficiency, precision, and robustness. Furthermore, our system can also use other marker patterns, but our system requires a large number of different markers, which may be over the limit of other marker design algorithms.

Marker-less tracking simultaneously recovers camera poses and scene structures from a video and are classified into three categories.

1) Feature-based methods [3], [11]–[14] extract sparse salient image features such as corners and edges from each frame, match them across successive frames with invariant feature descriptors, and estimate camera transformations and scene structures. The success of these methods depends on the number of detected features and the robustness of descriptor matching. Therefore, these methods may lack of robustness when the shot regions lack of features. In additional, feature detection and tracking across frames are computationally intensive. Our system avoid these two issues with marker-based tracking.
2) Direct methods [15]–[18] estimate camera transformations and scene structures directly based on intensities and their derivatives in each frame. This can avoid failure of feature detection and tracking. However, the

robustness is still an issue and photometric tracking is still computationally intensive. We avoid this by using pattern-based tracking.

3) Semidirective methods [4] combine both feature-based and direct methods for better efficiency and robustness. However, as our evaluation shows, camera manipulation in path design is generally fast and large to cause tracking failure, and we choose pattern-based tracking to ensure robustness.

Accelerometer-gyroscope-based methods such as AHRS [19] and Google Cardboard [20] track manipulation by accumulating the orientation and acceleration provided by gyroscopes and accelerometers. As a result, the precision of these algorithms highly depends on data precision, but commercially available microelectromechanical systems (MEMS) gyroscopes and accelerometers are imprecise to robustly track manipulation. Depth-based tracking methods such as KinectFusion [21] and Difodo [22] require enough depth variations and details to track camera movement in consecutive frames, and thus, tracking spots are limited. For example, our flat box surfaces are not suitable. Their precision is lower than our system. Furthermore, the price for a depth sensor is still much more expensive (U.S. $100) than a webcam (U.S. $30). During camera manipulation, an operator can manipulate the handle to generate a motion over limit of camera shooting and algorithmic tracking abilities. Tracking may fail, but they should recover tracking immediately after abnormal conditions are removed. Both types of methods are not equipped with the ability to recover their tracking without another calibration process. Our algorithm overcomes the precision, robustness, and stability issues by using maker-based tracking mechanism.

B. Augmented Reality Applications

AR creates an environment where computer-generated information is superimposed onto the user's view of a real world. For example, Wang [23] uses markers to simulate construction sites for a proper site setting. MacIntyre *et al.* [24] navigate through the virtual world of an interactive Web site using markers. Gee *et al.* [25] place markers on the ground to identify the location of the outdoor scene. These methods rely on marker detection and tracking to place extra information and contents onto the screen. We adopt a similar concept to track camera motion for manipulating the virtual worldview to build an imagination of camera shooting for discussion foundation and direct communication to remove the understanding gaps between the director and animators.

III. Overview

We develop EZCam to provide direct communication and immediate feedback based on marker recognition for direct camera manipulation during the design phase. There are mainly two components, transformation manipulator and parameter control board, in our system. The manipulator is designed for tracking the camera transformation and consists of a camera handle with a webcam, a light for providing enough lighting for precise marker tracking, and a tripod head for easily manipulating translation and rotation

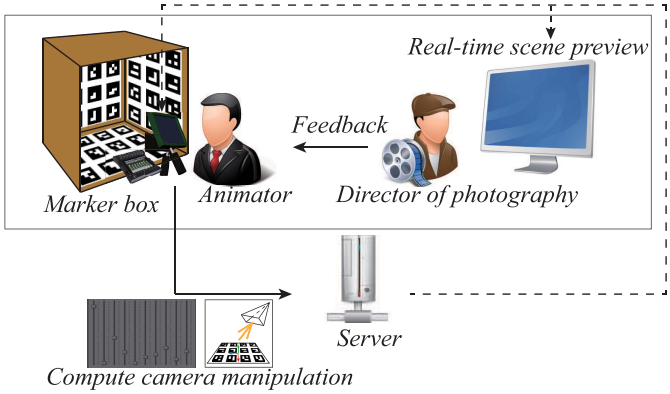


Fig. 2. Animators can manipulate the camera handle and adjust parameter sliders to control the worldview, our system estimates camera transformations through marker tracking and renders the world with selected parameters, and the director can comment and give out instructions to adjust camera manipulation directly and immediately.

and an auxiliary camera tracking box with perpendicular marker planes. As shown in Fig. 2, an operator manipulates the camera handle under director's instructions, and the webcam captures attached markers for estimating the transformation of marker planes. Our system combines multiple plane-camera transformations with weights computed based on the number of recognized markers to remove camera jittering. Furthermore, we also apply a double exponential smoothing filter for removal of hand-shaking artifacts. By analyzing the world-to-view matrix, we can easily estimate the camera transformation related to the visible-marker planes in world space. At the same time, the operator can also adjust the shooting parameters such as depth of field, focal length, and scale of each dimension using the control board of sliders for the total control over all camera parameters. After collecting view manipulation information, the server renders the virtual world accordingly for review. Since manipulation and rendering is in real time, the director and animators can directly examine the result and exchange modification opinions to reduce the number of iterations. Initially, there is no existing path, the captured per-frame transformations and parameters are used to construct an initial path with little cost. Later, since the design process is iterative, our system incrementally adjust the camera transformation and relative parameters based on the previously constructed/modified path.

IV. Implementation

This section first describes our camera tracking box and handle and our real-to-virtual pose estimation algorithm based on marker detection and recognition for analysis of the world-to-view transformation. Finally, we describe the mechanism of our designed control board for external camera parameters and its plug-in mechanism.

A. Portable Camera Pose Tracking Box

As shown in Fig. 3, there are three configurations designed for different purposes: 1) 5-plane inward; 2) 5-plane outward; and 3) 13-plane panoramic tracking configurations.

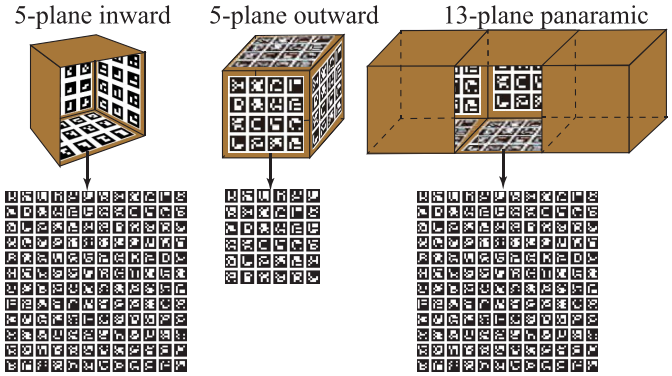


Fig. 3. Top three configurations: inward, outward, and panoramic tracking configurations, respectively, for different manipulation operations. Bottom: exemplar marker sheets used for each configuration.

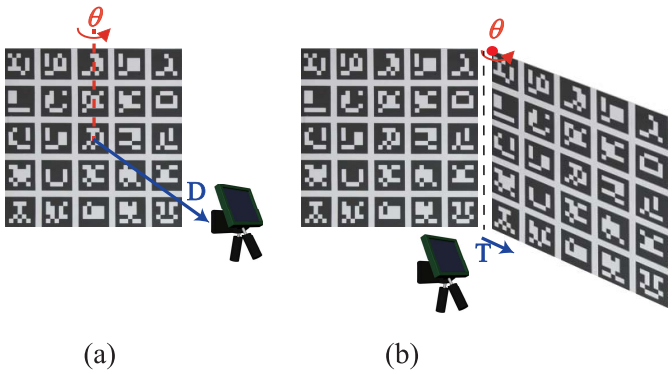


Fig. 4. (a) Experimental setting for determining recognition distance and orientation ranges. We set the camera shoot at the center of the plane and move it in and out until tracking fails. Furthermore, we also rotate the plane along the central axis until tracking fails. (b) Experimental setting for determining the registration error. We choose several shooting positions and orientations to capture both planes, and then we physically measure and photographically estimate the location and orientation of the second plane based on the front plane for determining the registration error. Later, we translate and rotate the second plane for further analysis of registration errors in different configurations.

The 5-plane inward configuration consists of five mutual perpendicular planes with a set of designed markers attached to the inward faces, and the five planes correspond to five operation directions: 1) front; 2) up; 3) down; 4) left; and 5) right. Similarly, the 5-plane outward configuration uses a similar concept except that the markers are placed in the outward faces instead of inward faces and the five planes correspond to five operation directions: 1) down; 2) front; 3) back; 4) left; and 5) right. The 13-plane panoramic configuration consists of three box units where both ends have a 5-plane inward unit, and the center uses three planes to bridge the end units. We design the 5-plane inward and outward configurations for easy construction but still maintaining enough camera manipulation variety. Generally, the 5-plane inward configuration is enough for most manipulation along one side but has a limitation in rotating backward. Therefore, we design the 5-plane outward configuration for those shots of concentric rotation around an object (i.e., orbit camera). The 13-plane configuration puts focus on both forward and backward tracking and manipulation, but we can easily extend it to track the left, right, and top movements and manipulation by replacing the left, right, and top transition faces with extra tracking boxes.

After setting up the configuration, we must determine the size of the markers. After discussing with a cameraman [26] and testing the camera manipulation, we find that the operation range should be in the range 25–100 cm. Therefore, we designed an experiment as shown in Fig. 4(a) to examine the distance and orientation ranges for a chosen marker size. We determine to choose the marker size of 6.5×6.5 cm^2 because they have the stable tracking distance between 20 and 150 cm. In addition, the stable detection orientation range is about 60°. Larger markers allow for a farther tracking range but must have the camera further away from the plane to capture enough markers for tracking because tracking requires four full markers for perspective-n-points (PnP) inversion. Smaller markers allow for closer tracking but shrink the tracking range and induce more perspective distortions. When using more markers, tracking is more precise, but efficiency is affected because the computational cost of a PnP inversion grows with the size of tracking points. Fundamentally, the size of the box is chosen while considering the following three aspects.

1) When testing manipulation inside a box with the selected marker size, we find that if the box size is smaller than $1 \times 1 \times 1$ m^3 and there is not enough space for translation inside the box. In order to get enough translation movement inside the virtual world, we need to scale the manipulated translation with a larger constant to induce instability.
2) The larger the box is, the more freedom we have for operation, but the less portable it becomes. We select the length of each side to be 113 cm for the 5-plane inward configuration to have enough space for operation but still have proper portability.
3) We want to have an enough number of markers in each plane for proper tracking and the selected number in each side is 12.

For the outward tracking box, we use the same marker size for the same operation range. In addition, since the box is used to track concentric camera movement, the $1 \times 1 \times 1$ m^3 box is too large to walk around for proper operations. According to our test, we choose a size of $0.5 \times 0.5 \times 0.5$ m^3.

Adapted from error-corrected grid codes [2], each marker consists of 7×7 grids whose outer border cells are black and used for easy detection. Therefore, there are 2^{25} possible keywords which are abundant to have distinct code words for all markers with proper error detection. Furthermore, each row encodes 2 bits of information for error detection and rotation invariant, and thus, there are a total of 1024 IDs available from 0 to 1023. To summarize, we observe that there are several tradeoffs. First, the size of the tracking box affects the operation size and the amount of allowable camera manipulation. When the box is larger, the allowable manipulation range is larger, but its portability becomes smaller. Second, the size of the marker is also a tradeoff between image recognition precision and the distance of detection because when the

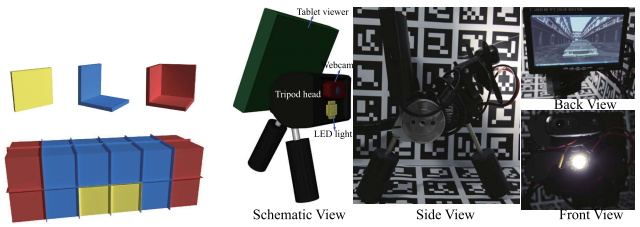


Fig. 5. (a) Top three fundamental elements: plane-shaped (yellow), L-shaped (blue), and C-shaped (red) for our tracking box. Bottom: the element composition of the 13-plane configuration. (b) Our handle of a webcam, a light source, a tablet, and a tripod head for camera position and direction manipulation.

marker is large, the distance required to fully spot the marker is larger, and the shortest operation distance is larger, but the precision of correct detection and pose estimation is higher. Furthermore, the tracking box must be portable, steady, and solid. As shown in Fig. 5(a), we decompose our box into three types of fundamental units: 1) plane-shaped; 2) L-shaped; and 3) C-shaped. We connect these units together with steel brackets or clippers. When decomposing into fundamental units, we can stack them together for portability. After designing the camera pose tracking box, Section IV-C gives the details of camera estimation based on the marker recognition.

B. Camera Handle Design

Fig. 5(b) shows the schematic design and snapshots of our camera handle. Because camera operators generally operate heavy photographic apparatuses and a Web camera is too light for comfortable grip [26], we use a tripod as the foundation of our camera handle. Although a tripod head is designed for professional cameras and camcorders, these shooting devices are generally not designed for live video stream connection to computers. Therefore, we mount a Web camera (Logitech portable webcam C950) on the top using clay because of its cheapness and easiness to connect with computers when comparing with those professional cameras. Since camera motion in movies is generally smooth and steady, we assume gradual camera manipulation to set the webcam in the frame rate of 30 frames/s with a resolution of 640×480. As shown in Section V, our tracking algorithm is efficient for a higher frame rate and resolution if users require tracking fast manipulation. The webcam communicates with the computer via a Universal Serial Bus (USB) port. Although we have already got as many recognized markers as possible to reduce jittering artifacts, there exists motion blur when the captured interval is long or the user moves camera quickly in a short period of time. Motion blur induces tracking failure and inaccuracy into our algorithm, and thus, we reduce this artifact by turning off the autolighting function and using as short an exposure period as possible at a value of 9000 in the Logitech user interface. Low capture time can reduce motion blur, but the captured frames are too dark to have correct recognition, and we correct this problem by attaching an extra LED lighting unit of 10W COB 820-900LM 6000-6500K cool-white LED chip on the tripod to enlighten the captured markers. After the server receives the captured frame, it uses the captured marker planes along with its markers to estimate the camera pose described in the following section. Although the camera handle captures manipulation to render results onto a review screen for the director and operator, it is inconvenient for the operator to manipulate the handle in one direction and check the visual feedback in another direction. We directly add a tablet onto the camera handle to real-time show the rendering results from the server when manipulating the virtual camera inside the scene.

C. Marker-Based Camera Transformation Estimation

AR applications depend on estimation of the precise camera transformation with/without markers, and we choose a marker-based estimation method adapted from error-correcting grid codes [2] because of their robustness and efficiency. Different from solving the occlusion problem in AR applications, our system focuses on robustly estimating the camera transformation to real-time manipulate the view. The following summarizes relative camera transformation estimation to a single marker plane with those visible markers. Then, we also give the details of combining multiple plane-camera transformations from all-visible-marker planes at the end of this section. Our system recognizes markers and estimates the relative transformation to the camera in the following steps.

1) *Camera Calibration:* Different cameras have their own intrinsic properties such as focal length and principle points. These parameters are important for determining the transformation from world space to screen space, and thus, it is necessary to correctly estimate them before tracking. In addition, the real-world camera deviates from the pinhole model, and it is important to take radial and tangential distortions into account. We express the camera model as

$$s\overline{m} = \mathbf{A}[\mathbf{R}|\overline{t}]\overline{M} \tag{1}$$

where s is a scale factor, $\overline{m}$ is the image-space coordinate, $\mathbf{A}$ is the intrinsic matrix of a camera, $\mathbf{R}$ and $\overline{t}$ represent the rotation and the translation of the camera, and $\overline{M}$ is the world-space coordinate. Actually, it can be regarded as a perspective transformation. Before any computation, our system runs a camera calibration process to estimate $\mathbf{A}$ [27].

2) *Marker Extraction:* Estimating camera transformation based on markers of designed patterns is generally efficient without the need of expensive feature detection such as scale-invariant feature transform [28] and speeded up robust features [29]. We fundamentally follow the steps proposed in [2] and summarize as follows. When capturing a frame, our system locally thresholds and segments the frame in order to detect the rectangular contours as our marker candidates. For each marker candidate, we apply perspective warping to get its top–bottom view and decompose it onto a regular grid of black and white components to discard those without black borders. Then, our system assigns each encoding element with 0 or 1 depending on the majority of pixel values inside the element. We use the coding of the candidate to identify its ID along with the four corners of the marker. Our system stores the identification and the

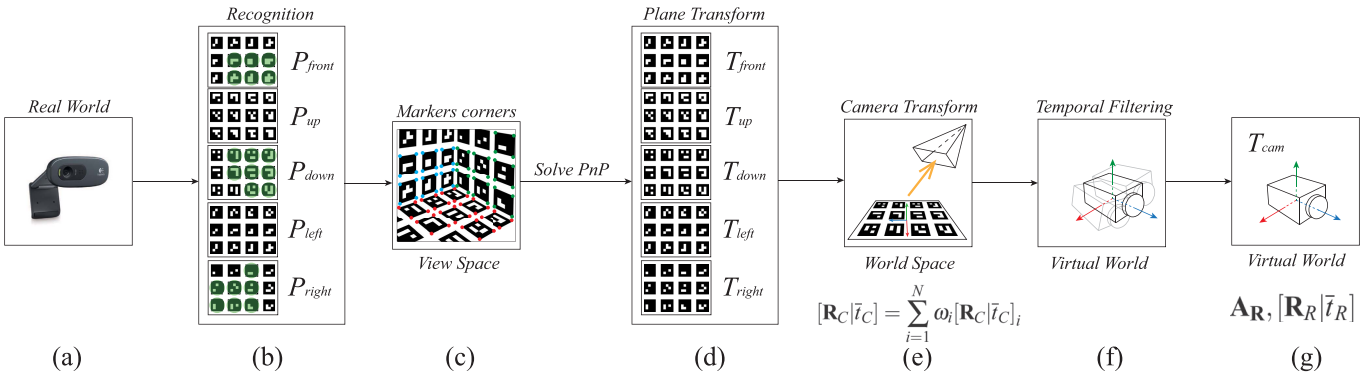


Fig. 6. (a) Webcam first takes a shot of the camera tracking box. (b) Our system identifies visible markers based on the encoding information. (c) We use corners of visible markers on the same plane to estimate the camera transformation according to the global position and orientation of the located plane. (d) and (e) Then, our system combines the transformation from different planes according to weights based on the number of visible markers of the plane. (f) We filter the overall camera transformation with a temporal filter. (g) Our system uses the camera transformation along with selected parameters to manipulate the shot of the virtual world.

four corners in a specified order for latter transformation estimation.

3) *Marker Transformation Estimation in Camera Coordinate:* After extracting recognized markers, the remaining problem is to estimate the relative relationship between the marker and camera, i.e., determine the orientation and translation of each marker in camera space. We can formulate the transformation estimation as a PnP problem [30] along with the constraints of detected corners to minimize the reprojection errors of all N corresponding points

$$[\tilde{\mathbf{R}}|\tilde{t}] = \underset{[\mathbf{R}|\bar{t}]}{\operatorname{argmin}} \sum_{i=1}^{N} r_i^2([\mathbf{R}|\bar{\mathbf{t}}]) \tag{2}$$

where the residual function, $r_i(\mathbf{R}|\bar{\mathbf{t}})$, denotes the reprojection error of the ith projection corresponding pair, $[\tilde{\mathbf{R}}|\tilde{t}]$ is the camera transformation including orientation which is represented as quaternion and translation, and N represents the number of 2D–3D correspondences. We can solve the problem with direct linear transformation (DLT) [31]. Therefore, our system takes four corner points of the recognized marker in model space, and their projected image locations along with camera intrinsic properties to estimate the object transformation in the camera coordinate.

4) *Plane Transformation Estimation in Camera Coordinate:* The recognized marker ID can determine the global location and orientation of the marker, $[\mathbf{R}_M|\bar{t}_M]$. We can estimate the camera location and orientation in the following manner:

$$\begin{aligned} [\tilde{\mathbf{R}}|\tilde{t}][\mathbf{R}_C|\bar{t}_C] &= [\mathbf{R}_M|\bar{t}_M] \\ [\mathbf{R}_C|\bar{t}_C] &= [\tilde{\mathbf{R}}|\tilde{t}]^{-1}[\mathbf{R}_M|\bar{t}_M] \end{aligned} \tag{3}$$

where $[\mathbf{R}_C|\bar{t}_C]$ is the world-space camera location and orientation. After solving $[\tilde{\mathbf{R}}|\tilde{t}]$ and recognizing the marker, we can easily estimate the camera transformation.

It is easy to estimate the camera transformation from the detection of a single marker. However, the usage of a single marker may result in a few artifacts including undesired movement in camera manipulation such as jittering, perspective distortion, and estimation error. Therefore, we put an array of markers on a plane and use all visible planes to relieve these issues and use multiple visible markers and their recognized corners instead of using a single recognized marker. Fig. 6(c) shows an example of recognized markers for the front, down, and right planes for the 5-plane inward configuration. First, our system identifies all visible markers and their corresponding ID in the view. Second, our system records those markers belonging to the same plane in a list along with their four corners. All recorded corners and their corresponding 2D image coordinates are used as constraints to determine the location and orientation of the marker plane by 2.

The perspective distortion from a single plane estimation becomes serious when the camera direction is almost perpendicular to the plane normal. We solve this issue by using multiple mutually perpendicular planes instead of one marker plane and choosing a plane that is closely perpendicular to the view direction, i.e., the camera can see a large number of markers.

When estimating the camera transformation, we have two different choices: 1) plane-based visible-marker estimation and 2) all-visible-marker estimation. The precision of all-visible-marker estimation is easily affected by those detected markers whose tilting angle to the image plane is large because of its large perspective distortions in its detected marker locations.

In addition, plane-based estimation allows us to consider each face independently for possibility of self-calibration to reduce the setting time for the tracking box. When building the tracking box, it is hard to maintain the relative position and orientation among planes. In addition, it is also hard to measure the location and orientation of each marker. Therefore, we develop a process to register the relative location and orientation of planes to the center and orientation of a reference plane. In our system, we first select the center of

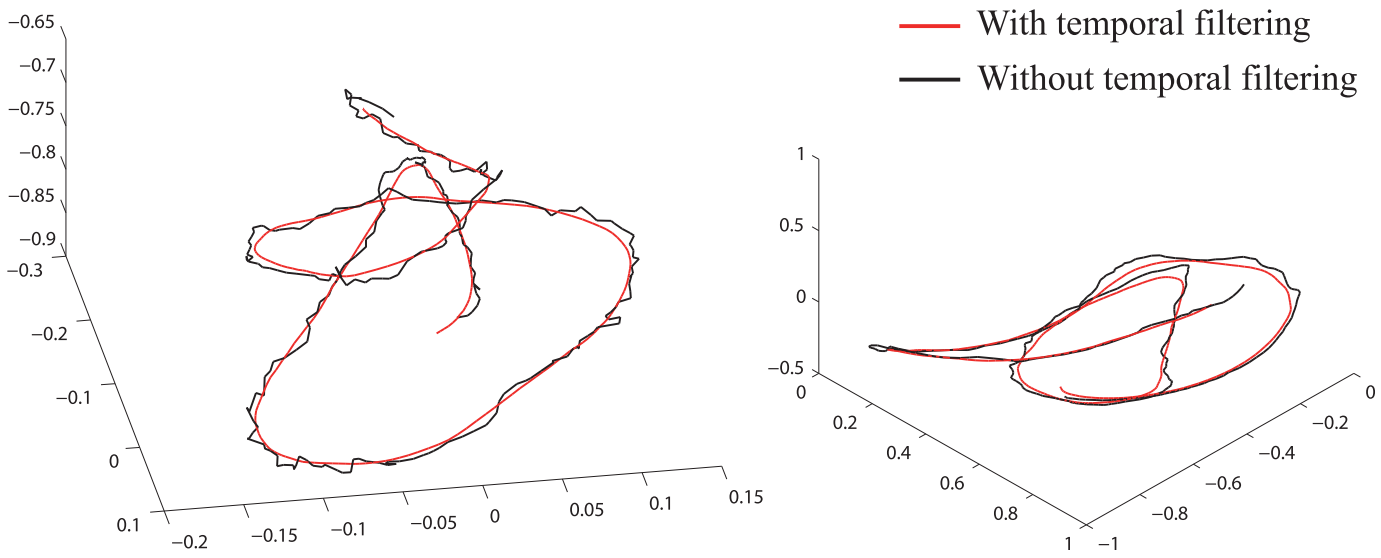


Fig. 7. Comparison of camera tracking in location (left) in the box coordinate and orientation (right) in quaternion, respectively, with and without the temporal filter.

the front face as the origin and its normal as the z-axis, and its right direction when facing as the x-axis to construct our world coordinate. During registration, our camera shoots at the registered and front planes for estimating its relative location and orientation along with all markers' location and orientation. This process allows us to print out marker sheets and attach them to the plane with misalignment and misorientation without affecting the tracking results. This process needs only to be done once before manipulation because generally the setting and configuration should not change during the operation. Since the process is so simple that when the configuration changes, we can easily restart manipulation by simple registration. After designing registration, we would like to understand the precision of registration. Therefore, we have conducted an experiment as shown in Fig. 4(b) to measure the registration errors of the plane when comparing with the physical measurement. We find that the mean estimation error is within 5 mm.

Generally, this strategy works well but there are jittering artifacts when switching the estimation plane. We remove these jittering artifacts by combing the transformation estimation from visible planes in the following manner. First, we use planes whose number of visible markers over a user chosen threshold to estimate the camera transformation. In other words, when the number of recognized markers from a plane is small, its estimation is not taken into account because estimation is error prone. Second, our system computes the number of markers in each estimated plane and their sum. Ratios between these numbers are computed as weights (ω_i) to represent effectiveness of their transformation estimation. Finally, we estimate the camera pose as the weighting average of all valid camera transformations

$$[\mathbf{R}_C|\overline{t}_C] = \sum_{i=1}^{N} \omega_i [\mathbf{R}_C|\overline{t}_C]_i \tag{4}$$

where i is the index of the plane. This weighting average can relieve the jittering artifacts caused by plane switching, but it cannot stabilize the camera caused by unconscious shaking of the operating hand. The following section describes our stabilization method for removal of hand-shaking artifacts.

D. Double Exponential Temporal Smoothing

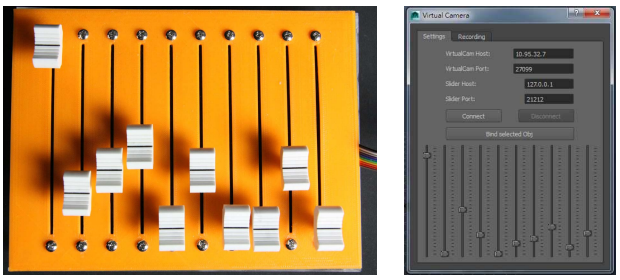
Stabilized camera paths are important for comfortable final rendering results. Although we have designed a camera handle with a proper weight to reduce hand-shaking artifacts and used multiple planes to prevent jittering artifacts when switching the reference plane, there are still shaking artifacts. We use a temporal filter to further relieve shaking artifacts. Instead of a complex Kalman filter, our system adapts the double exponential smoothing technique [32] which uses an exponentially decayed weighting for previous samples in a sequence based on a user specific parameter, $\alpha \in [0, 1]$, to smooth the camera transformation. The Nth camera transformation, $[\mathbf{R}_C|\overline{T}_C]^N$, can be estimated from the $(N-1)$th sample, $[\mathbf{R}_C|\overline{T}_C]^{N-1}$ as

$$[\mathbf{R}_C|\overline{T}_C]^N = \frac{2-\alpha}{1-\alpha}[\mathbf{R}_0|\overline{T}_0]^N - \frac{1}{1-\alpha}[\mathbf{R}_1|\overline{T}_1]^N \tag{5}$$

where $[\mathbf{R}_0|\overline{T}_0]^N$ and $[\mathbf{R}_1|\overline{T}_1]^N$ are two maintained prediction sequences

$$\begin{aligned}[\mathbf{R}_0|\overline{T}_0]^N &= \alpha[\mathbf{R}_C|\overline{T}_C]^{N-1} + (1-\alpha)[\mathbf{R}_0|\overline{T}_0]^{N-1} \\ [\mathbf{R}_1|\overline{T}_1]^N &= \alpha[\mathbf{R}_0|\overline{T}_0]^{N-1} + (1-\alpha)[\mathbf{R}_1|\overline{T}_1]^{N-1}.\end{aligned} \tag{6}$$

After using the smooth filter, the estimated camera path is smooth without jittering artifacts. Fig. 7 shows the comparison of camera manipulation paths with and without use of the filter, and the temporal filter can successfully remove the hand-shaking artifacts.



(a) (b)

Fig. 8. (a) Snapshot of the control board which consists of sliders for tuning the value of the plug-in camera parameters. (b) Plug-in parameter control panel in Maya.

E. Parameter Control Board

Our designed camera handle and tracking box can provide direct camera transformation manipulation, but they have limited manipulation dimensions due to the available physical space. In order to relieve this limitation, we design a control board to change the manipulation scale in each dimension for scene-specific and task-specific adjustment. In addition, certain shooting parameters such as focal length and field of view (FOV) are almost required to adjust during the shooting process. Therefore, the main goal of the control board is to provide the physical control of important camera parameters in the renderer for more photographic effects. When observing the manipulation manner of these parameters provided by the renderer, their values are generally controlled by a digital slider that is similar to the volume control in a sound track recorder. Therefore, we design our control board to have several sliding bars to linearly control the value of these parameters as shown in Fig. 8(a). The sliding bars are potentiometers whose resistance varies with the position of the bar and an AD converter reads the resistance of each slider and converts the analog signal into a digital one. The signal is transmitted to the server via a USB port. Later, we design a plug-in application to bind the camera object in Maya/Unity3D to connect with the desired control camera parameters, as shown in Fig. 8(b). The following are several plug-in parameters.

1) *Focal Length:* It is common to change the focusing part in a scene by tuning the focal length. For example, we can first focus on the pitcher ready to pitch and change the focal length to focus on the batter ready to swing his bat.
2) *FOV:* FOV generally decides the viewing angles and the amount of contents.
3) *Scale of Each Dimension:* Although the camera pose can be manipulated by the camera handle and tracking box, the linear mapping between the tracking and rendering cameras may result in inefficient control. Therefore, it is practical to give different scaling magnitudes in each dimension to achieve the desired controlling effects. For example, when we set scale of x dimension larger, the camera moves faster along the x-direction and this can cause slow/fast motion effect. Two men fight (the camera shoots with a normal scale) and then one of them shows the gun and shoots (the camera shoots with a larger scale).

Although some parameters can change the result from estimated pose information (scale of each dimension), the computation is relatively fast and the overhead can almost be ignored. In addition, the camera is first manipulated with the estimated camera pose estimated from the tracking box and then the plug-in parameters are applied later to give the final rendering results.

F. Renderer

After estimating the manipulation camera pose and setting the control parameters, the server computes and feeds the information as the camera extrinsic parameters (containing a 3×3 rotation matrix and a 3×1 translation vector) to its destined renderer. Currently, we choose three different rendering mechanics: 1) Maya; 2) Unreal Development Kit (UDK); and 3) Unity3D. For Maya and Unity3D, the camera transformation and parameters are directly fed into the application in a frame-by-frame manner to manipulate the camera of the active window as

$$\begin{aligned} \mathbf{A}_R &= \mathbf{A}_P(f, \text{fov}, \ldots) \\ [\mathbf{R}_R|\bar{t}_R] &= \mathbf{S}_P[\mathbf{R}_C|\bar{t}_C][\mathbf{R}_O|\bar{t}_O] \end{aligned} \tag{7}$$

where $\mathbf{A_P}$ is the adjusted intrinsic parameters, $\mathbf{S}_P$ is the scale motion in each dimension collected from the control board, and $[\mathbf{R}_O|\bar{t}_O]$ is the virtual camera transformation before manipulation. Although Maya and Unity3D can provide interesting rendering results, they are still not close to the realistic results. In order to have more realistic preview, the Maya scene and camera information can be directly feed to UDK for more powerful Unreal Engine for special lighting effects.

V. Qualitative and Quantitative Evaluations

After system design, we would like to understand the robustness and effectiveness of our camera manipulation method and have conducted several experiments to qualitatively and quantitatively evaluate its performance. At the same time, we have also conducted user studies to show that EZCam provides effective communication and enhances setting efficiency during the path design process against the Maya built-in camera tool. This section gives the details of the evaluations and user study.

A. Path Design Results

We have used our EZCam system to design camera paths for three animation scenes, katana, chain reaction, and speed drive 4 in Hong Kong, and the results are shown in Fig. 9. Details of implementation and manipulation results can be found in the supplemental Web site.[1] As shown, our system can easily manipulate the camera direction and orientation to create the desired camera path to shoot at the desired path.

[1]Web site address: graphics.csie.ntust.edu.tw/pub/EZCam/main.html.

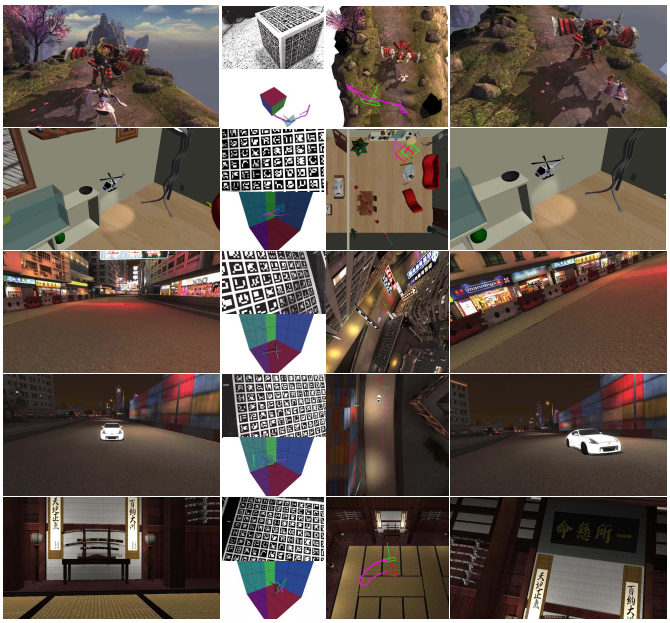
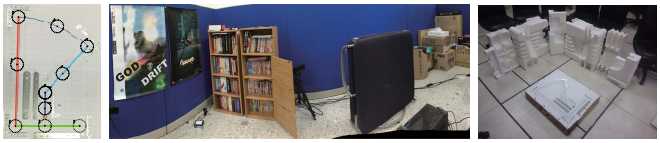


Fig. 9. Camera manipulation results for katana, chain reaction, speed drive 4 in Hong Kong, speed drive 4 in Hong Kong, and Japanese room. First column: the original camera view. Top of the second column: the captured frame of the camera tracking box. Bottom of the second column: the estimated camera transformation inside the box. Third column: the view of the scene with the original and manipulated camera transformations. Green marks the original camera transformations and red marks the manipulated camera transformations. Final column: the manipulation results.



(a) (b) (c)

Fig. 10. (a) Rail board for moving the camera handle on a designed path. (b) We have designed a color-feature-tracking corner by attaching two posters on its left side and placing two bookshelves with a large number of books to ensure abundant tracking feature points. (c) We have designed a depth-feature-tracking corner in our lab by placing and stacking several diffuse Styrofoam protectors whose surface is noneven for enough tracking depth details.

B. Qualitative Evaluation

The camera transformation estimation ability determines the effectiveness and robustness in camera manipulation. Therefore, after designing EZCam, we would like to understand its precision in estimating camera transformation. We have designed a camera path board for this purpose as shown in Fig. 10(a). Since we have manually moved the handle, the speed was not constant, and thus, we have only done numerical analysis at selected key points, $\overline{C}_1, \ldots, \overline{C}_6$. Then, we have designed a path consisting of different movement types including panning, zooming, circling, and rotating by connecting these key points. In order to avoid undesired deviations, we have placed steel brackets to set up the moving rail and perpendicular brackets at the corner to ensure rotation. In addition, the webcam is not mounted on the center of the camera handle, and we have used the configuration in Fig. 10(a) to estimate its position and orientation offset for computing the position and orientation at each key point. After designing the board, we have put it at a distance of 28 cm in front of the front plane for the 5-plane inward configuration. During the experiment, we have the camera handle station at the key points for a few seconds and then use the mean of all estimated transformation as our estimated transformation. SfM tracking algorithms may provide more flexible tracking mechanics, and thus, we have chosen two available state-of-art tracking algorithms, PTAM [3], and SVO [4] for comparison. The recorded video has been plugged into EZCam, PTAM, and SVO for estimating the precision of tracking as shown in our supplemental Web site.[1] First, the instant manipulation is generally faster and larger than their tracking limit. Second, PTAM heavily depends on feature detection and tracking, but the detected features from markers have a similar signature to induce tracking failure. Generally, these methods are designed to track camera poses in regular environments such as a corner in a lab. Therefore, we have designed a color-feature-tracking corner in our lab using two posters and two bookshelves as shown in Fig. 10(b) to ensure abundant tracking features for SfM algorithms. The tracking board has been placed at a location in front of the bookshelves with a distance of 28 cm for comparison on the designed path. The tracking results are shown in Fig. 11(b) and both methods still lose their tracking. Since detected features are distinct to have robust feature tracking, SVO performs much better but still loses their tracking because of the fast instant camera movement.

Accelerometer-gyroscope-based methods such as AHRS [19] and Google Cardboard [20] track manipulation based on the information provided by gyroscopes and accelerometers. Therefore, we have developed a motion tracking unit consists of an STMicroelectronics L3GD20 gyroscope and an STMicroelectronics LIS3DH accelerometer. Table I and Fig. 11(b) show the tracking results, and commercially available MEMS gyroscopes and accelerometers are generally too imprecise to provide proper manipulation abilities required for camera path planning. Only those sensors inside airplanes and satellites have precision high enough to get tracking precise, but they are really expensive and still require other types of sensors to continuously calibrate their tracking for proper operation. Furthermore, during the experiment process, large manipulation magnitude happens to make integration of orientation and acceleration become unstable to lose tracking. These algorithms would require a calibration process to restart the tracking. In other words, they cannot recover from tracking loss. Our algorithm uses encoding markers to recover tracking when failure happens.

Depth-based tracking methods such as KinectFusion [21] and Difodo [22] require enough depth details to track camera movement in consecutive frames, and thus, we have designed a depth-feature-tracking corner in our lab using several Styrofoam protectors as shown in Fig. 10(c) to ensure abundant tracking features for depth-based tracking algorithms. Table I and Fig. 11(c) show the tracking results, and Difodo has better precision than KinectFusion does. However, these depth-based tracking methods compute the camera transformation based on the relative depth information in consecutive frames, and thus,

TABLE I

TRACKING ESTIMATION RESULTS AT THE KEY POINTS OF THE DESIGNED PATH FOR EZCam, VUFORIA [6], PTAM, SVO, ARHS [19], GOOGLE CARDBOARD [20], KINECT FUSION [21], AND DIFODO [22]. WHEN TRACKING WITH OUR EZCam SYSTEM, WE HAVE USED FOUR SETS OF MARKER SIZES INCLUDING 4.5, 5.5, 6.5, AND 7.5 cm.

	C1	C2	C3	C4	C5	C6
(Pos.x, Pos.y, Ori.)	(-30.0, 0., 0.)	(-30.5, 50.5, 0.)	(5.40, 41.4,-0.714)	(-14.2, 18.7, -0.714)	(-8.90, 19.3, -1.57)	(-8.90, 12.2, -1.57)
EZCam (4.5)	(0.000, 0.013)	(0.100, 0.019)	(1.928, 0.028)	(1.041, 0.018)	(1.100, 0.046)	(1.000, 0.025)
EZCam (5.5)	(0.361, 0.029)	(0.283, 0.054)	(0.780, 0.042)	(0.498, 0.031)	(1.000, 0.075)	(0.200, 0.065)
EZCam (6.5)	(0.100, 0.002)	(0.300, 0.030)	(0.922, 0.004)	(0.899, 0.015)	(0.608, 0.089)	(0.000, 0.073)
EZCam (7.5)	(0.283, 0.014)	(0.100, 0.011)	(0.710, 0.001)	(0.718, 0.008)	(0.412, 0.097)	(0.000, 0.098)
EZCam (All)	(2.802, 0.007)	(0.316, 0.008)	(1.542, 0.064)	(1.433, 0.027)	(1.432, 0.081)	(1.780, 0.105)
EZCam (No. Filter)	(0.200, 0.018)	(0.412, 0.002)	(0.616, 0.050)	(0.866, 0.032)	(0.316, 0.024)	(0.608, 0.014)
EZCam (Kalman)	(0.224, 0.018)	(0.412, 0.002)	(0.616, 0.014)	(0.866, 0.006)	(0.316, 0.084)	(0.510, 0.087)
EZCam (ARToolKit)	(0.384, 0.000)	(1.272, 0.000)	-	-	-	-
EZCam (DLS)	(0.300, 0.015)	(0.412, 0.004)	(0.660, 0.047)	(0.992, 0.037)	(0.447, 0.031)	(0.632, 0.021)
EZCam (EPnP)	(1.414, 0.008)	(0.640, 0.012)	-	-	(1.616, 0.082)	(1.972, 0.104)
EZCam (UPnP)	(0.223, 0.011)	-	-	-	-	-
Vuforia	(0.540, 0.006)	(9.025, 0.016)	(8.734, 0.025)	(6.588, 0.039)	-	-
PTAM	(0.855, 0.043)	(0.907, 0.047)	-	-	-	-
SVO	(0.816, 0.025)	(2.198, 0.040)	(9.031, 0.171)	(8.804, 0.103)	-	-
ARHS	-	-	-	-	-	-
Cardboard	-	-	-	-	-	-
Kinect	(2.594, 0.010)	(1.775, 0.025)	-	-	-	-
Difodo	(0.156, 0.033)	(2.435, 0.018)	(0.842, 0.321)	(0.171, 0.335)	(2.896, 0.449)	(3.726, 0.403)

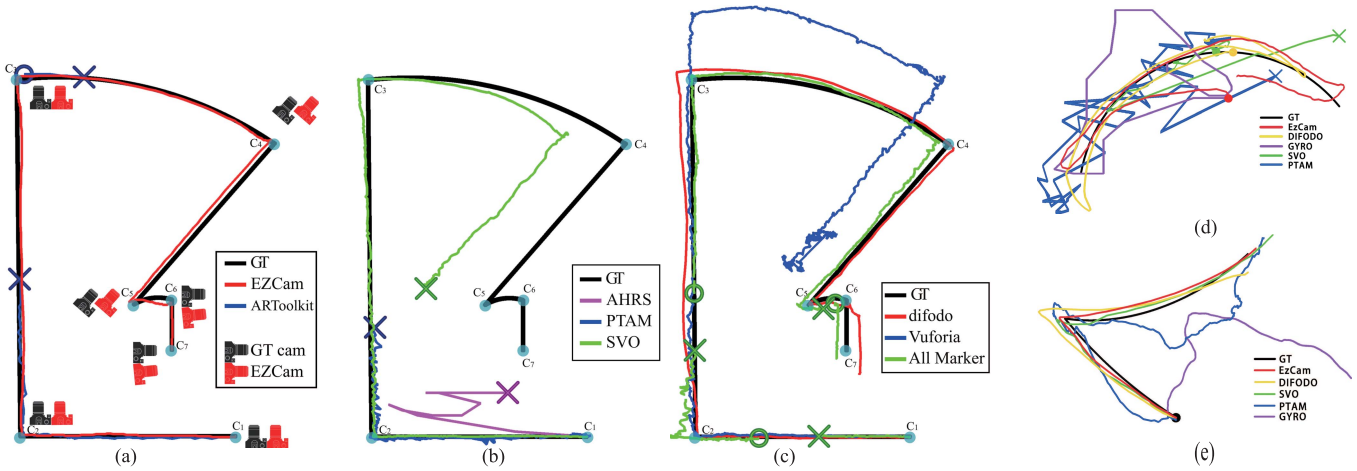


Fig. 11. (a) Camera tracking results using EZCam with ArUco marker detector (red) and with ARToolKit marker detector (blue) along with the 5-plane inward tracking box. Black marks the designed path. Along the path, black camera icons mark the shooting orientations of the designed path and red icons mark the tracking orientations of EZCam with the double exponential smoothing filter. Cyan solid dots illustrate the stop position for numerical measurement. Crosses of a corresponding track color illustrate the failure tracking locations and dots of a corresponding track color illustrate the restarting tracking locations. (b) Camera pose tracking results using AHRS (red), PTAM in the color-feature-tracking corner (blue), and SVO in the color-feature-tracking corner (green). (c) Camera tracking results using Difodo in the depth-feature corner (red), Vuforia in the tracking box (blue), and EZCam with all-visible-marker PnP inversion in the tracking box (green), respectively. (d) and (e) Camera tracking results in the room and racing scenes, respectively, using EZCam in 5-plane inward box (red), PTAM in the color-feature-tracking corner (blue), SVO in the color-feature-tracking corner (green), Difodo in the depth-feature-tracking corner (yellow), and AHRS (purple). Dots of a corresponding track color illustrate the starting locations and crosses of a corresponding track color illustrate the failure tracking locations.

when the motion of the camera is too large to track, these algorithms require a calibration process to restart the tracking. In other words, they cannot recover from loss camera tracking as Difodo unless they reconstruct rough 3D structures for calibration. KinectFusion uses voxels to represent the world, but the camera tracking for camera path planning is generally outward. Thus, the memory requirement is too large to handle by our system with 64-GB main memory to cause failure of tracking in the result. Both methods' tracking precision is still not as good as ours.

We also conduct a comparison with Qualcomm Vuforia [6]. We have placed its total available 512 patterns onto the surface of our 5-plane inward tracking box. Table I and Fig. 11(c) show the tracking results, and its precision is lower than ours. Because it can provide only 512 different encoding markers, there are not enough makers for precise tracking. In addition, their tracking mechanism is not as precise as our chosen scheme. In addition, we also measured the average operation time for different stages, marker recognition, position inversion, and smoothing, which are 6.91, 0.614, and 0.003 ms, and the average tracking time of EZCam, PTAM, SVO, ARHS, Cardboard, Difodo, KinectFusion, and Vuforia, which are 7.93, 8.58, 2.50, 0.00174, 7.45, 77.6, 4.36, and 9.66 ms, respectively.

We would like to understand whether the marker size affects the tracking ability. Therefore, we have designed another three

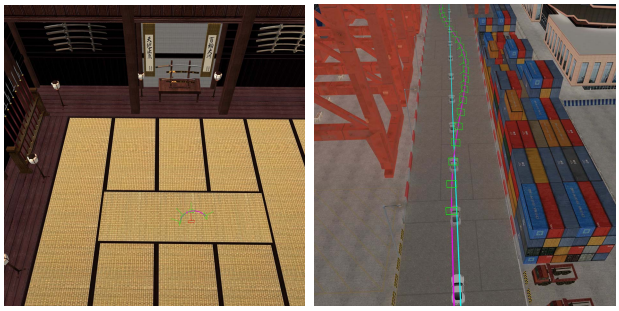
marker sets with a marker size of 4.5 × 4.5, 5.5 × 5.5, and 7.5×7.5 cm^2. We have placed these sets on the inward tracking planes and tested their tracking precision using the same designed path. The precision analysis is also listed in Table I. We can find that when using smaller one, the tracking precision is higher when the camera is near the tracking planes, but the tracking range is smaller. When using marker recognition, the tracking can be free from scale calibration which is generally needed for vision-based tracking. The marker size determines the range of tracking, and the number of visible markers determines the tracking precision.

We would like to understand whether the all-visible-marker PnP inversion and our plane-based visible-marker PnP inversion affect the tracking precision and stability. Therefore, we use both inversion methods for experiments on the track, and the results are shown in Table I and Fig. 11(c). When doing PnP inversion with all visible markers at once, its estimation precision is easily affected by outliers whose detected location is seriously disturbed by noises and artifacts. This is especially serious when parts of the captured markers lie on a plane whose tilting angle to the image plane is large. In other words, this large tilting angle induces large perspective distortions into detected marker locations. We have added random sample consensus to relieve the disturbance of these outliers, but the precision is still not good. Our system uses weighting average of plane-based PnP inversion results to further improve estimation precision. We also measured their inversion time of all marker and image plane as 0.816 and 0.984 ms, respectively.

We also want to know the difference between the Kalman filter and the double exponential smoothing filter as shown in Table I. We also measured the smoothing time of double exponential smoothing and Kalman as 2.55 and 1.02 μs, respectively. The double exponential smoothing filter has comparative smoothing results with comparative computational cost when comparing against the Kalman filter. However, to implement double exponential smoothing is simpler.

We have replaced our markers with those generated by ARToolKit [7] as shown in Table I and Fig. 11(a). It shows that other marker patterns can be easily plugged into EZCam with our plane-based estimation method for camera tracking. However, our chosen marker pattern can have more correctly identified markers in each frame (averaging 90.1% for ours and 73.4% for ARToolKit on visible markers) for more precise tracking. We also measured the recognition time of error-corrected grid codes and ARToolKit as 6.91 and 1.26 ms, respectively.

Finally, we would like to know whether different PnP inversion algorithms including DLT [31], direct least square (DLS) [33], Efficient Perspective-n-Point (EPnP) [34], and Unified Perspective-n-Point (UPnP) [35] affect the tracking results as shown in Table I. All algorithms are available in OpenCV 3.0 [36], and we get extra implementation of UPnP from [35]. We also measured their inversion time of DLT, DLS, EPnP, and UPnP as 0.614, 36.087, 0.183, and 0.241 ms, respectively. Currently, our camera transformation estimation is based on DLT to solve PnP inversion, and there are less than 500 2D–3D matching points involved. Under this problem size, DLT can get a solution within 1 ms with a precision under 1 cm which is already satisfied with our tracking purpose. When plugging in DLS, the computational cost is much higher than DLT with a limited improvement in estimation precision. Since our plane-based tracking configurations have all features lying on the same plane, while viewing the plane from a large angle, perspective distortions induce serious deviations on the EPnP inversions. Moreover, our configuration has the UPnP inversion generate 3–4 solutions in our experiment. Generally, the first solution can better describe the camera motion, and thus, our system uses it for camera tracking. However, perspective distortions also cause serious deviations on its estimated results. Furthermore, the UPnP-estimated camera motion is wiggling and less stable. Since our plane-based tracking configuration generates sufficient artificial markers for accurate and robust tracking, it is a good choice for our system. However, more efficient and precise PnP inversion algorithms are still welcome to get more precise results in a shorter period of time. Due to length limitation, we listed complete experimental data in our supplemental Web site.[1]



(a) (b)

Fig. 12. (a) and (b) Scenes, Japanese room and Hong Kong Port, used for studying the efficiency enhancement of our proposed tool where blue and purple mark the original and desired camera paths, respectively.

C. Quantitative Manipulation Evaluation

Since the goal of this system is to manipulate camera transformation for camera path design, we have designed two scenes, room and racing, along with their respective camera manipulation scenario for quantitatively evaluating the manipulation ability of our system when comparing with PTAM, SVO, AHRS, and Difodo. The first scene is a room with walls and doors in the Japanese style, and the original camera is still to look at the door of the room. The scenario simulates that the hero wakes up to check around the room as shown in Fig. 12(a). The desired camera motion is as follows.

1) The hero rotates his head to the left and the top of the room for examination.
2) The hero rotates his head to the right and bottom of the room for examination.
3) The hero moves his head back to examine the front of the room.

The second scene is a racing scene in the port of Hong Kong, and the animator ties the original camera on the back of the hero car. The scenario simulates that the camera puts focus on the hero car, and the camera requires us to move around

TABLE II

FIRST PART: MEANS AND STANDARD DEVIATIONS OF THE TIME TO OBTAIN THE PROPER CAMERA POSES FOR TASKS 1 AND 2 IN THE USEFULNESS STUDY OF THE PROPOSED TOOL. SECOND PART: MEANS AND STANDARD DEVIATIONS OF THE NUMBER OF OPERATIONS AND THE TIME TO OBTAIN THE PROPER STEREOSCOPIC SETTING FOR TASKS 1 AND 2 IN THE USEFULNESS STUDY OF THE PROPOSED TOOL

	Efficiency		Effective Communication			
	Time (s)		# of Ops		Time (s)	
	Mean	Std	Mean	Std	Mean	Std
Room(Maya)	843	697	4.389	2.118	1220	612
Room(Ours)	81.1	58.4	1.222	0.428	50.7	29.8
Racing(Maya)	1100	541	3.389	1.614	1110	410
Racing(Ours)	96.8	56.2	1.556	0.616	62.7	29.0

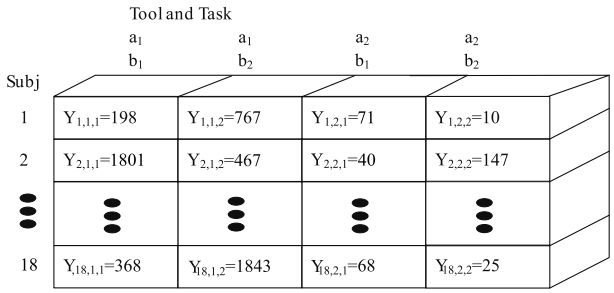


Fig. 13. Data arrangement for RBF-22 of the randomized block factorial design (RBFD). $Y_{i,j,k}$ denotes a score in one of the $i = 1, \ldots, n$ blocks for subjects, $j = 1, \ldots, p$ for tool types, and $k = 1, \ldots, q$ for scenes. Since there are two tools, the Maya built-in and our proposed tools, and two scenes, there are a total of four combinations for each subject.

to generate excitement and nervousness of racing as shown in Fig. 12(b). The desired camera motion is as follows.

1) When the hero car moves, the shooting camera looks at the front of the car and moves close to emphasize this car.
2) The camera puts the hero car in the center of the view and then moves along a curve similar to the car front window to the left until the entire left side of the body is within the view.
3) The camera moves to the right along a curve similar to the car front window to see the right side of the body.

After designing the scenarios, we apply EZCam to manipulate the camera pose to achieve the desired effects. At the same time, we also apply PTAM, SVO, AHRS, and Difodo to manipulate our camera. All the results are shown in Fig. 11(d) and (e). Our method can effectively and robustly set up the desired camera pose for both scenarios. Our method can create a path more close to the desired one when comparing with other methods.

D. Quantitative User Study

We have conducted user studies to verify the usefulness of our proposed tool with the physical setting shown in Fig. 14(c). We used a Sharp LC-40W5T (1920×1080 pixels, 400-cd/m^2 brightness, 240-Hz refresh rate, and 90-cm screen width) for reviewing the manipulation and rendering results. Before our study, participants were asked to have a 5-min training section to get familiar with EZCam.

The first study focuses on efficiency enhancement for path design. We asked subjects to set up a camera path as the one described in the quantitative experiment described in the previous section. Two tools were used to complete the tasks of setting camera paths for these two scenarios. One is the Maya built-in camera manipulation tool, and the other is the proposed tool. The instructions were given at the beginning of this section, and subjects used the tools to set up the camera paths for the two scenes. To avoid the studying effect, the instructor counter-balance chose the order of using these two tools for each subject. During the study, we recorded how much time (only including operation but not rendering and evaluation) the subject spent for obtaining the desired camera paths. In total, 18 subjects participated in our experiments with normal or corrected-to-normal vision. Their ages range from 22 to 37 years; and two are females and 15 are males. In addition, all of them have at least two years of experiences using Maya and are familiar with the tools provided by Maya. We listed the means and standard deviations of the operation time for room and racing in Table II. With the proposed tool, the time to get the proper setting was roughly reduced to 15% compared with Maya's built-in tool. This paper aims at evaluating efficiency enhancement of our proposed EZCam, and differences among the subjects who have different genders and ages, which are called nuisance variables, may make a significant contribution to error variances and thereby affect the judgment. Therefore, we used a randomized block factorial design (RBFD) [37] to employ a blocking procedure to assign the levels of nuisance variations randomly to the experimental units for distribution of known and unsuspected variation sources among the units over the entire experiment in order to avoid the affection of just one or a limited number of affected factors. We collected and listed the data in the format as shown in Fig. 13, and the complete data of this paper are provided in the supplemental Web site.[1] The procedure involves 18 blocks of 2×2 homogeneous experimental units, where 18 blocks correspond to the subjects and 2×2 units correspond to two tools and two tasks, respectively. In Fig. 13(b), $Y_{i,j,k}$ is the score set for the time required to finish the task. According to [37], we can formally express the expectation of $Y_{i,j,k}$ with a mixed model for type Random Blocked Factorial (RBF)-pq design as

$$Y_{ij} = \mu + \alpha_j + \beta_k + (\alpha\beta)_{j,k} + \pi_i + \varepsilon_{i,j,k}$$

where μ denotes the overall population mean, α_j denotes the effect of the jth tool, β_k denotes the effect of the kth task, $(\alpha\beta)_{j,k}$ denotes the joint effect of the jth tool and the kth task, π_i denotes the effect of the ith subject, and $\varepsilon_{i,j,k}$ denotes the experimental error. We can derive the single and overall deviation from experimental data in a similar manner as discussed in [37]. Finally, we computed their $F_{1,68} = 71.6$ ($p < .05$) for operation time on the tool types according to [37, Table 9.5-1]. Therefore, the tool is a significant factor for the operation time. Based on the mean operation time, the proposed tool does enhance the efficiency of the camera path design.

The second study focuses on effective communication of EZCam for path design. We asked subjects to set up a camera

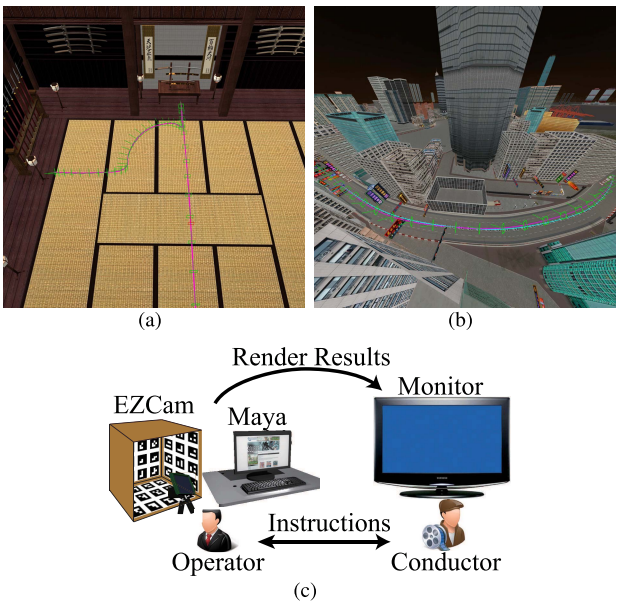


Fig. 14. (a) and (b) Scenes, Japanese room and Hong Kong Port, used for studying the effectiveness in communication of our proposed tool where blue and purple mark the original and desired camera paths, respectively. (c) Subject sat in front of the 5-plane inward tracking box for operating EZCam and next to a computer for operating Maya. He/she was asked to set up the camera path using the Maya built-in tool and our EZCam tool. The scene was rendered based on the camera poses and displayed on the review monitor. An instructor sat next to the subject and gave instructions according to the rendered frame.

path for two scenes as purple paths in Fig. 14(a) and (b) with our selected setting requirement. We have designed the camera operations for the room scene for simulation of walking into the room and then walking around to examine the environment as follows.

1) The camera moves from the door to the interior of the room.
2) The camera keeps moving forward to observe the katana rack until only the katanas are inside the view.
3) The camera moves backward and turn left to check the name cards.
4) When turning left, the camera must clearly see the names on the cards.

We have designed the camera operations for the racing scene to simulate looking out of the car window as follows.

1) The camera stays inside the front seat to look out of the window.
2) When the right of the view spots the signboard of extreme speedy motor, the camera turns to check the signboard.
3) After checking, the camera turns back to look forward.
4) When the left of the view spots the signboard of Chinese Study School, the camera turns to check the board.
5) After checking, the camera turns back to look forward.
6) When the top of the view spots the signboard of Synthesized Perfume Drug Store, the camera turns to check the board.

For this study, subjects were asked to use the same two tools as the previous one. The built-in tool of Maya was operated as follows.

1) Subjects were asked to switch to the worldview to examine the scene and the camera position.
2) The instructor gave the key setting requirement.
3) Subjects were instructed to add keyframes for the camera when he/she thought necessary.
4) Subjects were instructed to adjust the camera location and orientation on the deviated parts of camera motion.
5) The result was rendered to show on the TV. The instructor reviewed it and gave out instructions to help subjects adjust the parameters.
6) Steps 2–4 were repeated until reaching the stopping criteria.

EZCam was operated in the following manner.

1) Subjects were asked to switch to the worldview to examine the scene and the camera position.
2) The instructor gave the key setting requirement.
3) Subjects were instructed to run the animation and use the EZCam system to manipulate the camera pose for each frame.
4) The instructor gave certain comments during the setting process to help them set the proper camera path.
5) The result was rendered to show on the TV. The instructor reviewed it and gave out instructions to help subjects adjust the parameters.
6) Steps 2–4 were repeated until reaching the stopping criteria.

To avoid the studying effect, the instructor counter-balance chose the order of using these two tools for each subject. During the study, we recorded how many operations and how much time (only including operation but not rendering and evaluation) the subject spent for obtaining the desired camera path. We listed the means and standard deviations of the number of operations and the time for room and racing in Table II. With the proposed tool, both the number of operations and the time to the proper setting were roughly reduced to 33% and 15%, respectively, compared with Maya's built-in tool. Similarly, we also used a RBFD [37] to avoid the affection of these limited nuisance factors and compute their $F_{1,68} = 162$ and 58.8 ($p < .05$) for the operation time and numbers, respectively. Therefore, our tool is a significant factor for path design. Based on the scores, our tool can provide direct instruction to reduce the number of operations and improve the efficiency of the camera path design process. Due to the limitation of space, we listed the complete user study statistics in our supplemental Web site.[1] In addition, we also interviewed subjects for their opinions about EZCam. Certain subjects reflected that it is like the joystick requiring to construct a operation metaphor and once constructed, EZCam provides intuitive manipulation means to design camera paths.

VI. Conclusion

This paper proposes an interactive, low-cost, and real-time camera path design system that provides direct communication between animators and the director to avoid misunderstanding and reduce the number of revision iterations. We design a handle and several tracking configurations along with real-time transformation estimation for different types of camera manipulation. In addition, we also design a control board to plug in the renderer for controlling other parameters

for desired photographic effects. Our system is not without limitations and there are a few future research directions. First, to operate our system requests two persons, one manipulates the camera and the other adjusts the control board. They may have different interpretations of DP's instructions, which may induce more revision time. Thus, we would like to integrate control sliders onto the handle similar to a video camera. Second, marker recognition depends on image quality, and when camera movement is fast, motion blur may result in estimation artifacts. We would like to incorporate motion deblurring into our system to overcome this issue.

ACKNOWLEDGMENT

The authors would like to thank M.-T. Tsai for box assembly elements and the participants for user studies.

REFERENCES

[1] F. Gaspar, R. Bastos, and M. Sales, "Accurate infrared tracking system for immersive virtual environments," *Int. J. Creative Interfaces Comput. Graph.*, vol. 2, no. 2, pp. 49–73, Jul. 2011.

[2] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognit.*, vol. 47, no. 6, pp. 2280–2292, 2014.

[3] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality (ISMAR)*, Nara, Japan, Nov. 2007, pp. 225–234.

[4] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May/Jun. 2014, pp. 15–22.

[5] (2015). *Ncam: AR/VR Realtime Camera Tracking*. [Online]. Available: http://www.ncam-tech.com/

[6] (2015). *Qualcomm Vuforia Developer Portal*. [Online]. Available: https://developer.vuforia.com/

[7] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," in *Proc. 2nd IEEE ACM Int. Workshop Augmented Reality (IWAR)*, Oct. 1999, pp. 85–94.

[8] M. Fiala, "Designing highly reliable fiducial markers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 7, pp. 1317–1324, Jul. 2010.

[9] L. Calvet, P. Gurdjos, and V. Charvillat, "Camera tracking based on circular point factorization," in *Proc. 21st ICPR*, Nov. 2012, pp. 2128–2131.

[10] F. Bergamasco, A. Albarelli, and A. Torsello, "Image-space marker detection and recognition using projective invariants," in *Proc. Int. Conf. 3D Imag., Modeling, Process., Visualizat. Transmiss. (3DIMPVT)*, May 2011, pp. 381–388.

[11] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2010, pp. 21–28.

[12] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Real-time detection and tracking for augmented reality on mobile phones," *IEEE Trans. Vis. Comput. Graphics*, vol. 16, no. 3, pp. 355–368, May/Jun. 2010.

[13] L. Kneip, M. Chli, and R. Siegwart, "Robust real-time visual odometry with a single camera and an IMU," in *Proc. Brit. Mach. Vis. Conf.*, 2011, pp. 16.1–16.11.

[14] S. Weiss *et al.*, "Monocular vision for long-term micro aerial vehicle state estimation: A compendium," *J. Field Robot.*, vol. 30, no. 5, pp. 803–831, 2013.

[15] M. Irani and P. Anandan, "About direct methods," in *Proc. Int. Workshop Vis. Algorithms, Theory Pract. (ICCV)*, 2000, pp. 267–277.

[16] A. I. Comport, E. Malis, and P. Rives, "Real-time quadrifocal visual odometry," *Int. J. Robot. Res.*, vol. 29, nos. 2–3, pp. 245–266, Feb. 2010.

[17] T. Tykkälä, C. Audras, and A. I. Comport, "Direct iterative closest point for real-time visual odometry," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCV Workshops)*, Nov. 2011, pp. 2050–2056.

[18] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Proc. IEEE ICRA*, May 2013, pp. 3748–3754.

[19] S. O. H. Madgwick, A. J. L. Harrison, and A. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," in *Proc. IEEE Int. Conf. Rehabil. Robot. (ICORR)*, Jun./Jul. 2011, pp. 1–7. [Online]. Available: http://dx.doi.org/10.1109/ICORR.2011.5975346

[20] (2015). *Google Cardboard*. [Online]. Available: https://www.google.com/get/cardboard/

[21] S. Izadi *et al.*, "KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera," in *Proc. 24th Annu. ACM Symp. User Interface Softw. Technol. (UIST)*, 2011, pp. 559–568.

[22] M. Jaimez and J. González-Jiménez, "Fast visual odometry for 3-D range sensors," *IEEE Trans. Robot.*, vol. 31, no. 4, pp. 809–822, Aug. 2015.

[23] X. Wang, "Using augmented reality to plan virtual construction worksite," *Int. J. Adv. Robot. Syst.*, vol. 4, no. 4, pp. 501–512, 2007.

[24] B. MacIntyre, A. Hill, H. Rouzati, M. Gandy, and B. Davidson, "The Argon AR Web Browser and standards-based AR application environment," in *Proc. 10th IEEE Int. Symp. Mixed Augmented Reality (ISMAR)*, Oct. 2011, pp. 65–74.

[25] A. P. Gee, M. Webb, J. Escamilla-Ambrosio, W. Mayol-Cuevas, and A. Calway, "A topometric system for wide area augmented reality," *Comput. Graph.*, vol. 35, no. 4, pp. 854–868, 2011.

[26] J.-W. Kuo, private communication, Aug. 2015.

[27] D. C. Brown, "Decentering distortion of lenses," *Photogramm. Eng.*, vol. 32, no. 3, pp. 444–462, May 1966.

[28] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[29] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, 2008.

[30] S. Li, C. Xu, and M. Xie, "A robust $O(n)$ solution to the perspective-n-point problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1444–1450, Jul. 2012.

[31] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3D," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 835–846, 2006.

[32] J. J. LaViola, "Double exponential smoothing: An alternative to Kalman filter-based predictive tracking," in *Proc. Workshop Virtual Environ. (EGVE)*, 2003, pp. 199–206.

[33] J. A. Hesch and S. I. Roumeliotis, "A direct least-squares (DLS) method for P*n*P," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Nov. 2011, pp. 383–390.

[34] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EP*n*P: An accurate $O(n)$ solution to the P*n*P problem," *Int. J. Comput. Vis.*, vol. 81, no. 2, pp. 155–166, 2009.

[35] L. Kneip, H. Li, and Y. Seo, "UPnP: An optimal $O(n)$ solution to the absolute pose problem with universal applicability," in *Computer Vision* (Lecture Notes in Computer Science), vol. 8689, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Heidelberg, German: Springer, 2014, pp. 127–142.

[36] (2015). *OpenCV*. [Online]. Available: http://opencv.org/

[37] R. E. Kirk, *Experimental Design*, 2nd ed. Pacific Grove, CA, USA: Brooks/Cole, 1982.

Cheng-Chi Li received the B.S. degree from the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, in 2014, where he is currently working toward the M.S. degree with the Department of Computer Science and Information Engineering.

His research interests include computer graphics, vision, and parallel computing.

Yu-Chi Lai (M'14) received the B.S. degree from the Electrical Engineering Department, National Taiwan University, Taipei, Taiwan, in 1996; the M.S. and Ph.D. degrees in electrical and computer engineering from University of Wisconsin–Madison, Madison, WI, USA, in 2003 and 2009, respectively; and the second M.S. and Ph.D. degrees in computer science, in 2004 and 2010, respectively.

He is an Associate Professor with National Taiwan University of Science and Technology, Taipei. His research interests include graphics, vision, and multimedia.

Nai-Sheng Syu received the B.S. degree from the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, in 2014, where he is currently working toward the M.S. degree with the Department of Computer Science and Information Engineering.

His research interests include computer graphics, vision, and image processing.

Hong-Nian Guo received the B.S. degree from the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, in 2015, where he is currently working toward the M.S. degree with the Department of Computer Science and Information Engineering.

His research interests include computer graphics, vision, and nonphotorealistic rendering.

Dobromir Todorov received the B.S. degree in computer systems and technologies from Technical University of Varna, Varna, Bulgaria, in 2009, and the M.S. degree from the Computer Science Department, National Taiwan University of Science and Technology, Taipei, Taiwan, in 2013.

He is a Senior Software Engineer with Next Media, Inc., Taipei. His research interests include graphics, visual effect, and game design.

Chih-Yuan Yao (M'08) received the M.S. and Ph.D. degrees in computer science and information engineering from National Cheng Kung University, Tainan, Taiwan, in 2003 and 2010, respectively.

He is an Assistant Professor with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan. His research interests include computer graphics, mesh processing and modeling, and nonphotorealistic rendering.