Stable Pose Tracking from a Planar Target with an Analytical Motion Model in Real-time Applications

Po-Chen Wu #1, Yao-Hung Tsai *2, Shao-Yi Chien #3

# *Media IC and System Lab*
Graduate Institute of Electronics Engineering
National Taiwan University
1 pcwu@media.ee.ntu.edu.tw
3 sychien@ntu.edu.tw

* *Department of Electrical Engineering*
National Taiwan University
2 b99901152@ntu.edu.tw

***Abstract*—Object pose tracking from a camera is a well-developed method in computer vision. In theory, the pose can be determined uniquely from a calibrated camera. However, in practice, most real-time pose estimation algorithms experience pose ambiguity. We consider that pose ambiguity, i.e., the detection of two distinct local minima according to an error function, is caused by a geometric illusion. In this case, both ambiguous poses are plausible, but we cannot select the pose with the minimum error as the final pose. Thus, we developed a real-time algorithm for correct pose estimation for a planar target object using an analytical motion model. Our experimental results showed that the proposed algorithm effectively reduced the effects of pose jumping and pose jittering. To the best of our knowledge, this is the first approach to address the pose ambiguity problem using an analytical motion model in real-time applications.**

I. Introduction

The objective of pose estimation is to calculate the position and orientation of a target object from a calibrated camera. Augmented reality (AR) [1], where synthetic objects are inserted into a real scene in real-time, is a prime candidate system for pose estimation. After obtaining the pose computed using geometric information, the system can render computer-generated images (CGI) according to the pose on the display. For example, ARToolkit [2] is a system that is used widely with AR applications. The target object in AR systems is usually the planar fiducial marker, which is used frequently for navigation and localization.

The information available for solving the pose estimation problem is usually a set of point correspondences, which comprise a 3D reference point expressed in object coordinates and its 2D projection expressed in image coordinates [3], [4]. Using the object-space collinearity error, Lu et al. [4] derived an iterative algorithm that computed the orthogonal rotation matrices directly. Instead of using the iterative algorithm, Ansar et al. [5] developed a framework that generated a set of linear solutions to the pose estimation problem, and the algorithm was applicable to points and lines. These online pose

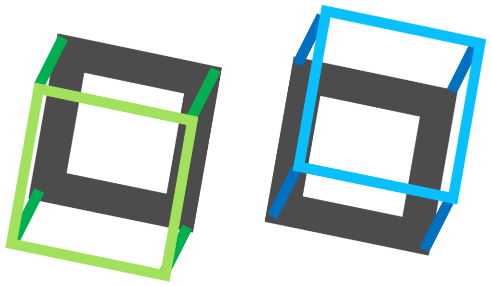
Fig. 1. Illustration of pose ambiguity, which is a geometric illusion. There appears to be more than one 3D geometrical explanation based on the same perspective-projected marker on the image plane.

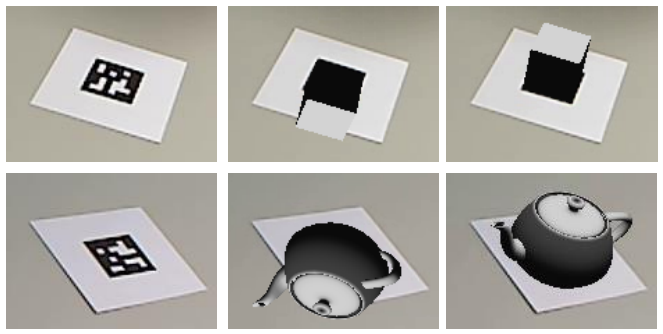
Fig. 2. Pose estimation results. The images in the first column are the original images. The images in the second column are CGIs, where the poses were estimated using state-of-the-art algorithms. The images in the third column were obtained using our algorithm.

estimation methods determined a unique pose in each frame without considering the pose ambiguity problem.

Pose ambiguity is the main cause of pose jumping, as shown in Fig. 1. According to our experience, several state-of-the-art pose algorithms suffer from pose jumping. These pose ambiguity problems have been discussed in previous studies [6], [7], [8]. Oberkampf et al. [6] provided a straightforward interpretation for orthographic projection. They developed an algorithm for planar targets, which used scaled orthographic projection during each iteration step. Schweighofer et al. [7]

extended this method to address the general case of perspective projection and developed an algorithm that obtained a unique solution for pose estimation. However, the problem of pose jumping still persists occasionally with these algorithms, as shown in Fig. 2. In addition, the problem of pose jittering also bothers users due to the noisy images. Wu et al. [8] attempted to determine the correct pose with an empirical motion model, but it would lose the accuracy of estimated pose without analyzing the motion model based on the ground truth.

Thus, to reduce the effects of pose jumping effectively, we developed an algorithm that uses an analytical motion model to obtain the pose of the target object. The motion model is updated using a Kalman filter [9], which provides an efficient computational method for estimating the true poses by computing the weighted average of the measured pose and the predicted pose from the motion model. Based on our observations, one of the two ambiguous poses with distinct local minima according to an error function is the correct pose. Therefore, after obtaining the two ambiguous poses in each iteration, the pose that is most similar to the predicted pose is selected. If the predicted pose is realistic, it is almost guaranteed that the pose selected is the correct one.

The main contributions of this study are as follows.

1) We can address the problem of pose jumping, because we can select the correct pose from two ambiguous poses using the analytical motion model.
2) The effects of pose jittering are reduced by the Kalman filter. We can estimate the pose that tends to be closer to the true pose than the measured pose. The sequences of estimated poses are also much smoother because the poses are much more consistent with the previous ones.
3) This is the first work to attempt pose estimation combined with an analytical motion model. If the target object is not detected in some frames for long sequences, we can simply use the pose predicted by the motion model as the final pose to prevent discontinuities in the sequence of poses.

The remainder of this article is organized as follows. First, we describe the formulation of the pose estimation problem in detail in Section II. We explain pose ambiguity and describe a method to obtain the two poses with local minima according to an error function in Section III. In Section IV, we describe the details of our stable pose estimation algorithm. In Section V, we present the results using our proposed pose estimation algorithm and compare its performance with other competitive pose estimation algorithms. Our conclusions are given in Section VI.

II. Problem Formulation

The main problem of camera pose estimation is determining the six degrees of freedom, which are parameterized based on the orientation and position of the target object with respect to a calibrated camera (with known internal parameters), as shown in Fig. 3. Given a set of noncollinear 3D coordinates of reference points $\mathbf{p}_i = (x_i, y_i, z_i)^t, i = 1, ..., n, n \geq 3$ expressed as object-space coordinates and a set of camera-space

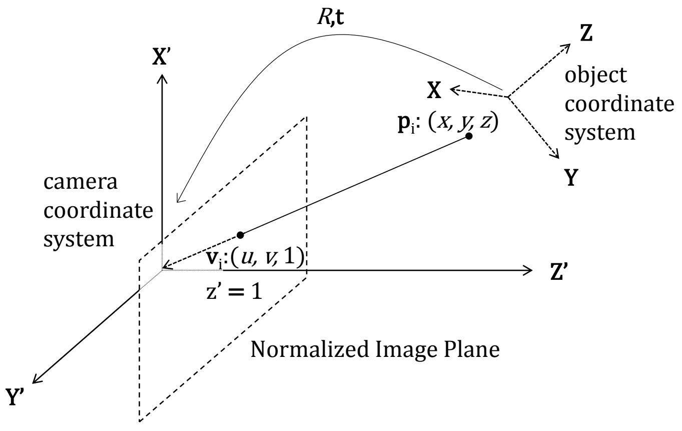

Fig. 3. Coordinate systems between a camera and target objects for addressing the pose estimation problem.

coordinates $\mathbf{q}_i = (x'_i, y'_i, z'_i)^t$, the transformation between them can be formulated as

$$\mathbf{q}_i = R\mathbf{p}_i + \mathbf{t}, \tag{1}$$

where

$$R = \begin{pmatrix} \mathbf{r}_1^t \\ \mathbf{r}_2^t \\ \mathbf{r}_3^t \end{pmatrix} \in SO(3) \quad \text{and} \quad \mathbf{t} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \in R(3) \tag{2}$$

are the rotation matrix and translation vector, respectively.

We use the *normalized image plane located* at $z' = 1$ as the camera reference frame. In this normalized image plane, we define the image point $\mathbf{v}_i = (u_i, v_i, 1)^t$ as the projection of $\mathbf{p}_i$ on it. In the idealized pinhole camera model, $\mathbf{v}_i$, $\mathbf{q}_i$, and the center of projection are collinear. We can express this relationship using the following equation:

$$u_i = \frac{\mathbf{r}_1^t\mathbf{p}_i + t_x}{\mathbf{r}_3^t\mathbf{p}_i + t_z}, \quad v_i = \frac{\mathbf{r}_2^t\mathbf{p}_i + t_y}{\mathbf{r}_3^t\mathbf{p}_i + t_z}. \tag{3}$$

or

$$\mathbf{v}_i = \frac{1}{\mathbf{r}_3^t\mathbf{p}_i + t_z}(R\mathbf{p}_i + \mathbf{t}), \tag{4}$$

which is known as the collinearity equation in photogrammetry. Given the observed image points $\hat{\mathbf{v}}_i = (\hat{u}_i, \hat{v}_i, 1)^t$, the pose estimation algorithm needs to determine values for R and $\mathbf{t}$ that minimize an appropriate error function. In principle, there are two possible error functions. The first is the *image-space error*, which was used by [3] and [6],

$$E_{is}(R, \mathbf{t}) = \sum_{i=1}^{n} \left[(\hat{u}_i - \tfrac{\mathbf{r}_1^t\mathbf{p}_i + t_x}{\mathbf{r}_3^t\mathbf{p}_i + t_z})^2 + (\hat{v}_i - \tfrac{\mathbf{r}_2^t\mathbf{p}_i + t_y}{\mathbf{r}_3^t\mathbf{p}_i + t_z})^2 \right] \tag{5}$$

whereas the second is the $object - space\ error$, which was used by [4] and [7]:

$$E_{os}(R, \mathbf{t}) = \sum_{i=1}^{n} \left\| (I - \hat{V}_i)(R\mathbf{p}_i + \mathbf{t}) \right\|^2 \tag{6}$$

where

$$\hat{V}_i = \frac{\hat{\mathbf{v}}_i\hat{\mathbf{v}}_i^t}{\hat{\mathbf{v}}_i^t\hat{\mathbf{v}}_i} \tag{7}$$

is the line-of-sight projection matrix, which is applied to a scene point and projects the point orthogonally onto the line of sight defined by the image point $\hat{\mathbf{v}}_i$. In our study, we use the *object-space error* as the error function.

III. Pose Ambiguity Interpretation

Pose ambiguity describes situations where the error function has several local minima for a given configuration. Pose ambiguity is caused by the low accuracy of reference point extraction, which is almost inevitable in general cases. Fig. 1 shows the illustration of pose ambiguity.

Most recent pose estimation algorithms that operate in real time experience pose ambiguity, as shown in Fig. 2. Schweighofer et al. [7] found that when coplanar points $\mathbf{p}_i = (p_{i_x}, p_{i_y}, 0)$ are viewed by a perspective camera, the algorithm typically derives two distinct minima, according to E_{is} and E_{os}. We derive the two poses with the minima of E_{os} using the method proposed in [7].

IV. Stable Pose Estimation Algorithm

After obtaining the poses with the local minima, some previous methods have determined the final pose using the lowest error E_{os} [7]. Unfortunately, these methods still experienced pose ambiguity even when selecting the optimal solution of E_{os}. Indeed, the correct pose $\hat{P}$ need not be the pose with the lowest error. Based on our experimental evidence, we determined that the second pose is the correct one when pose jumping occurs. The results shown in Fig. 4 agree with our assumption: the two poses with local minima sometimes interchange and one of them is correct. Based on these observations, we develop our *Stable Pose Estimation Algorithm*. During each time step, the system generates a predicted pose $\tilde{P}$ based on a motion model. This motion model simulates the orientation of the pose in real conditions and is updated by the Kalman filter in each time step. Of two candidates, the pose that is more similar to $\tilde{P}$ is selected as the correct pose $\hat{P}$. The final pose is the weighted average of the predicted pose $\tilde{P}$ and measured pose $\hat{P}$.

A. Motion Model

Assuming that the motion model of the pose rotation about three axes (X, Y, and Z) is identical, we address only the case where rotation occurs about the X-axis in the remainder of this paper. The cases with rotation about the Y-axis and Z-axis are the same.

To estimate the next rotation angle using a motion model, the motion model needs to maintain the current angle and angular velocity. The angle and angular velocity are described by the linear state space $\mathbf{x}_k = [x \ \dot{x}]^t$, where $\dot{x}$ is the angular velocity. We assume that between the $(k-1)$ and k time step the system undergoes a constant angular acceleration of a_k, which is normally distributed with the mean 0 and standard deviation σ_a for Δt seconds. Based on Newton's laws of motion we conclude that

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{G}a_k, \tag{8}$$

where

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{G} = \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix}. \tag{9}$$

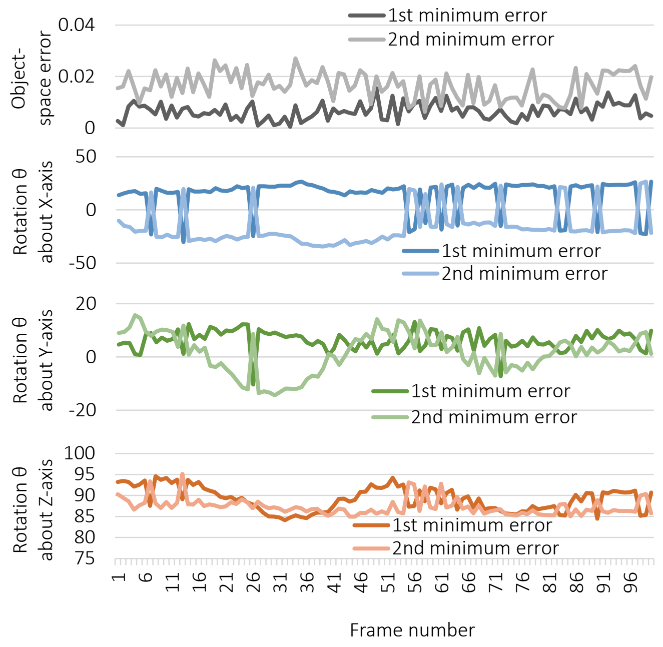

Fig. 4. Object-space errors E_{os} in a video sequence with a planar target. The pose with the lower error E_{os} (the darker plot) is the final pose in each frame. The rotation angle about the X-axis, Y-axis, and Z-axis of the poses with the minimum error E_{os} changes dramatically during some frames.

We rewrite (8) in another form

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{w}_k, \tag{10}$$

where

$$\mathbf{w}_k \sim N(0, \mathbf{Q}) \quad \text{and} \quad \mathbf{Q} = \mathbf{G}\mathbf{G}^t\sigma_a^2 = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \sigma_a^2. \tag{11}$$

For each time step, we measure the rotation angle about the X-axis. Let us assume that the measurement noise $\mathbf{v}_k$ is also normally distributed with the mean 0 and standard deviation σ_z:

$$\mathbf{m}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \tag{12}$$

where

$$\mathbf{H} = [1 \ 0] \quad \text{and} \quad \mathbf{v}_k \sim N(0, \mathbf{R}), \ \mathbf{R} = \mathrm{E}[\mathbf{v}_k\mathbf{v}_k^t] = \sigma_z^2. \tag{13}$$

B. Parameter Analysis

We need the ground truth of the rotation angles about the X-, Y-, and Z-axis to measure the standard deviation of the analytical motion model. We use the program Blender [10] to calculate the rotation matrix of the moving camera relative to the tracking marker to obtain reliable data, which are referred to as the ground truth. To facilitate more precise camera tracking, we place the checkerboard pattern around the tracking marker, as shown in Fig. 5. To avoid the effect of pose jumping, the measured pose should be the correct one from the two ambiguous poses. We consider that the measured poses will have a higher error variance when the area of the tracking marker is smaller in a frame. Fig. 6 shows the differences in the rotation angles about the X-, Y-, and Z-axis relative to the area of the tracking marker, and it verifies our assumption.

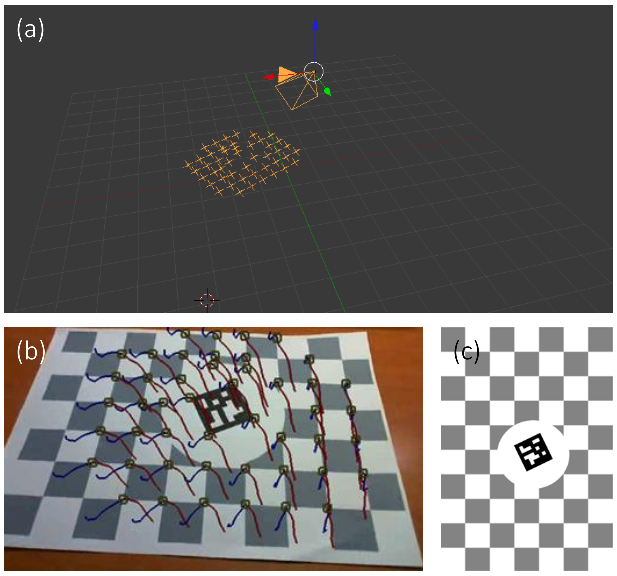

Fig. 5. We obtained the ground truth for the camera pose using Blender [10]. (a) The yellow crosses on the plane are the feature points of the tracking marker and the chessboard pattern. (b) Feature points on the marker and the chessboard pattern. The blue line and the red line denote the tracking path of the features. Blender uses these tracking paths to solve the extrinsic parameters of the camera. (c) The chessboard-based marker.

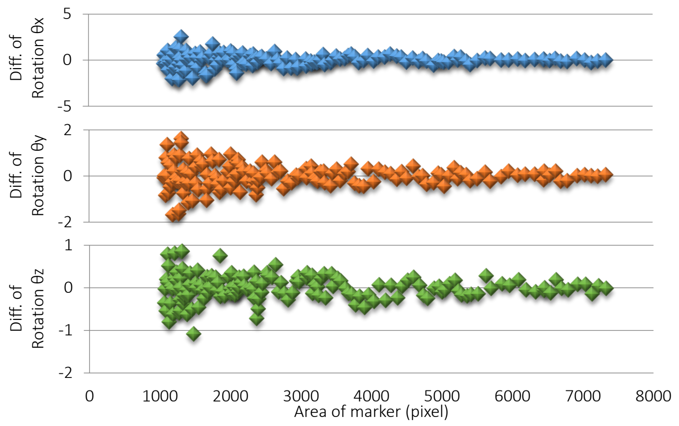

Fig. 6. The differences in rotation angles about the X-axis, Y-axis, and Z-axis of the ground truth poses and the poses generated using our proposed algorithm without the motion model relative to the area of the marker.

Fig. 7 shows the relationship between the standard deviation of the rotation angles about the X-, Y-, and Z-axis relative to the area of the marker. We consider that the standard deviation of the rotation parameter has an inverse power law (IPL) relationship to the area of the marker, and the existence of the regression lines in Fig. 7 verifies our assumption to some extent. We do not know how the camera moves in every condition so it is impossible to analyze the standard deviation σ_a of the angular acceleration of a_k. Thus, we simply use $\sigma_a = 20$.

C. Prediction and Updating Using the Kalman Filter

The operation of the two phases of the Kalman filter, “Predict” and “Update,” are shown in Fig. 8. Due to pose

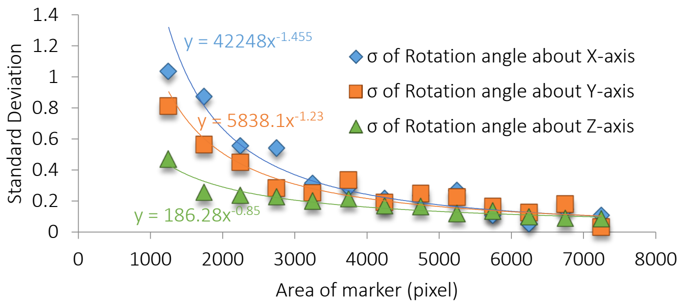

Fig. 7. The relationship between the standard deviation of the rotation angles about the X-, Y-, and Z-axis relative to the area of the marker.

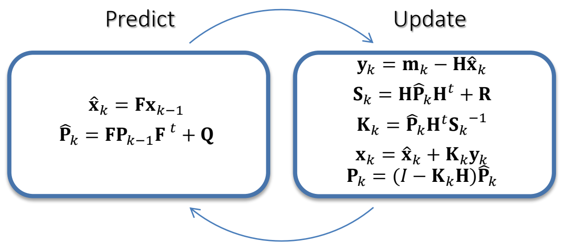
Predict
Update
$\hat{\mathbf{x}}_k = \mathbf{F}\mathbf{x}_{k-1}$
$\hat{\mathbf{P}}_k = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^t + \mathbf{Q}$
$\mathbf{y}_k = \mathbf{m}_k - \mathbf{H}\hat{\mathbf{x}}_k$
$\mathbf{S}_k = \mathbf{H}\hat{\mathbf{P}}_k\mathbf{H}^t + \mathbf{R}$
$\mathbf{K}_k = \hat{\mathbf{P}}_k\mathbf{H}^t\mathbf{S}_k^{-1}$
$\mathbf{x}_k = \hat{\mathbf{x}}_k + \mathbf{K}_k\mathbf{y}_k$
$\mathbf{P}_k = (I - \mathbf{K}_k\mathbf{H})\hat{\mathbf{P}}_k$

Fig. 8. Detailed operation of the two phases of the Kalman filter, “Predict” and “Update.”

ambiguity, two pose measurements, $\hat{\mathbf{m}}_{k1}$ and $\hat{\mathbf{m}}_{k2}$, are obtained in reality at each time step. Assuming that the a priori state estimate $\hat{\mathbf{x}}_k$ is authentic, the measurement that is most consistent with $\hat{\mathbf{x}}_k$ is regarded as the only measurement $\mathbf{m}_k$. After the Kalman filter operations, a new a posteriori state estimate $\mathbf{x}_k$ is obtained, which can be used during the next recursion.

D. Initial Conditions

To guarantee that the state estimate is reliable during each time step, we need to ensure that the state estimate is correct at the beginning. It is assumed that the difference between the first and second minimum object-space errors E_{os} is usually smaller when pose jumping occurs. This can be verified based on Fig. 4 to some extent. To obtain the error difference threshold value, we recorded the error differences in poses using numerous marker-based video sequences, as shown in Fig. 9. The result showed that pose jumping did not occur when the error difference was larger than 0.015. Thus, we set the error difference threshold to 0.015.

Based on this characteristic, we select the pose with the first minimum object-space error E_{os} as the first measurement $\mathbf{m}_0$ if the error difference between the two poses is greater than the threshold. If we cannot find a suitable pose during the first ten frames, we determine the first measurement $\mathbf{m}_0$ based on a vote. The first measured poses are much more reliable after this operation. After obtaining the first reliable measured pose, the state estimate $\mathbf{x}_k$ at each time step is similar to the previous one (this is a distinguishing feature of the Kalman filter) and all of them are reliable to some degree. In our experiments, this was the crucial factor for avoiding pose jumping.

After determining the first measurement $\mathbf{m}_0$, the state estimate and estimate covariance matrix, $\mathbf{x}_0$ and $\mathbf{P}_0$, respectively,

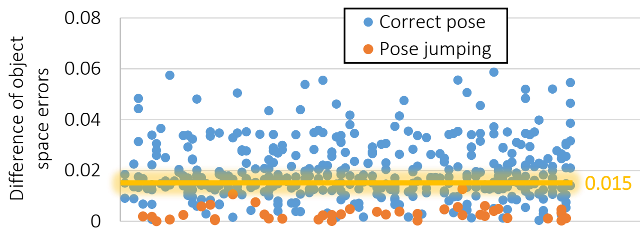

Fig. 9. Differences between the first and second minimum object-space errors E_{os}. We set the threshold to 0.015. If the error difference is larger than the threshold, we can claim the pose with the first minimum object-space errors E_{os} is the correct pose.

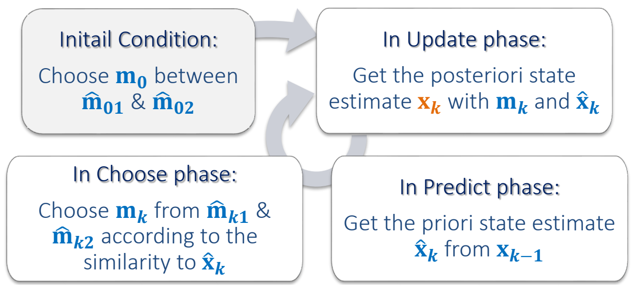
Initail Condition:
Choose $\mathbf{m_0}$ between $\hat{\mathbf{m}}_{01}$ & $\hat{\mathbf{m}}_{02}$
In Update phase:
Get the posteriori state estimate $\mathbf{x}_k$ with $\mathbf{m}_k$ and $\hat{\mathbf{x}}_k$
In Choose phase:
Choose $\mathbf{m}_k$ from $\hat{\mathbf{m}}_{k1}$ & $\hat{\mathbf{m}}_{k2}$ according to the similarity to $\hat{\mathbf{x}}_k$
In Predict phase:
Get the priori state estimate $\hat{\mathbf{x}}_k$ from $\mathbf{x}_{k-1}$

Fig. 10. System flow of the stable pose estimation algorithm.

were initialized as follows:

$$\mathbf{x}_0 = \begin{bmatrix} \mathbf{m}_0 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{P}_0 = \begin{bmatrix} L & 0 \\ 0 & L \end{bmatrix}, \tag{14}$$

where L is a value determined by the variance of the initial state. A higher L means that the initial state estimate is very unreliable and the true value tends to be closer to the measurement values. In this case, the state estimate is set to the same value as the measurement. In this study, we set $L = 10$ as our initial condition.

Fig. 10 shows the process flow of our proposed stable pose estimation algorithm. Finally, we use the first element of $\mathbf{x}_k$ as the output value of the pose estimation.

V. Experimental Results

In this section, we discuss the effects of different parameter settings and present the pose estimation results. Various video sequences of markers with random rotation angles from the camera were used as the test data. According to the marker pattern in the database described by [11], we determined the set of point correspondences between the object space and image plane as $\mathbf{p}_i$ and $\hat{\mathbf{v}}_i$ in Section II. Next, we calculated the pose of the planar marker from the camera using the set of point correspondences with the proposed algorithm and other state-of-the-art algorithms.

The process was performed using a PC with an Intel Core i7-2600K (3.40 GHz) processor and 16 GB of memory. In our current implementation in C++, the proposed system could process a 640*480 pixel image (without rendering 3D computer-generated images) in approximately 0.79 ms. The camera (webcam) used by our system was a Microsoft LifeCam Cinema and we obtained the intrinsic camera parameters using the OpenCV library.

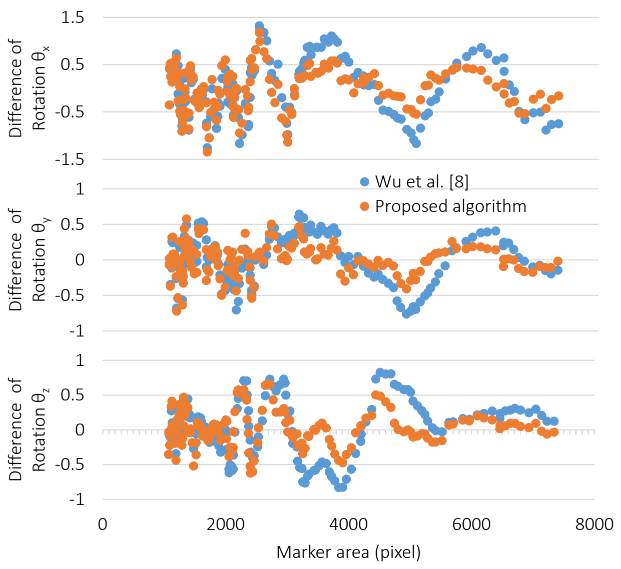

Fig. 11. The differences in the rotation angles about the X-, Y-, and Z-axis relative to the area of the tracking marker.

Mean of absolute difference	Wu et al. [8]	Proposed algorithm
Rotation θ about X-axis	0.497	0.365
Rotation θ about Y-axis	0.269	0.181
Rotation θ about Z-axis	0.322	0.192

TABLE I. Mean absolute differences between the ground truth and another pose estimation method for the rotation angles about the X-, Y-, and Z-axis.

A. Parameter Settings

We recorded various video sequences using a chessboard-based marker at random distances from the camera and we calculated the camera poses in these video sequences using different parameter settings. Figure 11 shows the differences in the rotation angles about the X-, Y-, and Z-axis between the ground truth poses and other methods relative to the area. The differences between the ground truth poses and those with the proposed algorithm appeared to be smaller with the correct parameter settings. An overall comparison is shown in Table I. It shows that our proposed algorithm outperformed [8] because it included the analytical motion model.

B. Pose Estimation Results Comparison

We compared the results using different pose estimation algorithms by recorded various video sequences of a marker placed at random rotation angles from the camera. Fig. 12 compares the proposed pose estimation results with the results obtained using state-of-the-art algorithms. In each condition and at every time step, our algorithm provided a high stability solution for real-time pose estimation. The first row in Fig. 12 shows the original images with a fiducial marker. The second and third rows show the pose estimation results with CGIs based on Kato et al. [2], where the latter was implemented using history functions. The fourth row shows the pose estimation results based on Schweighofer et al. [7]. The final row

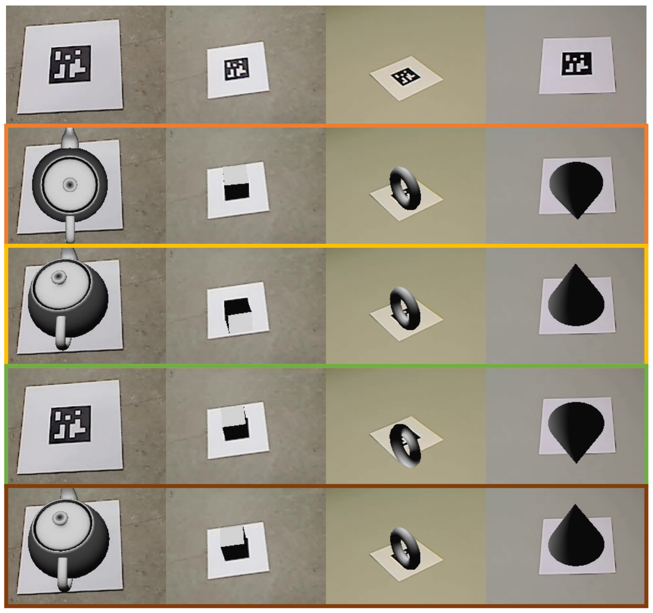

Fig. 12. Pose estimation results comparison. The first row shows the original images with a marker. The second to fourth rows show the results obtained using other algorithms, while the fifth row shows the results obtained using our proposed algorithm.

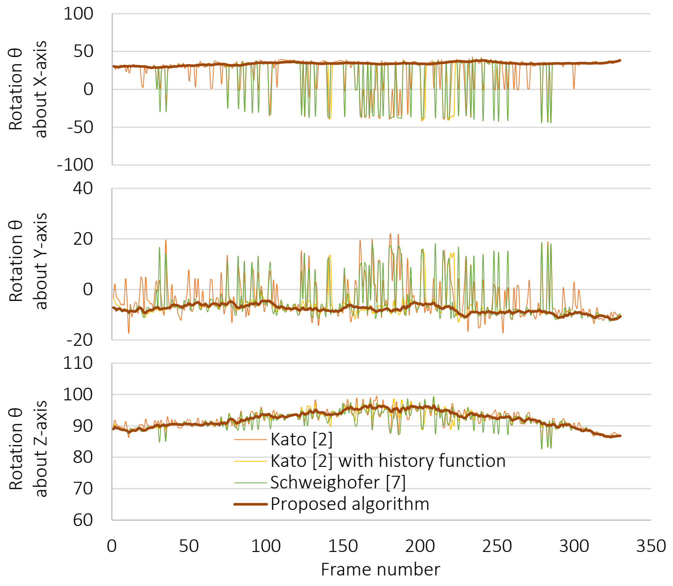

Fig. 13. Comparison of the rotation angles about the X-axis, Y-axis, and Z-axis for various poses.

shows the results obtained using our proposed algorithm. Even at a low resolution and with noisy images, our method obtained pose sequences without pose jumping, which were markedly better than the pose estimation results obtained using other algorithms.

Fig. 13 shows the rotation angle of the marker relative to the camera. When the pose jumped during the video sequences, the rotation angle varied dramatically. The most obvious examples are shown in the first and second charts in Fig. 13. Our proposed algorithm avoided pose jumping in most cases. Furthermore, much more stable poses were generated using the analytical motion model. Pose jittering implies that the difference in the values are unsettled during video sequences and that they fluctuate around 0. Fig. 13 shows that the pose sequences obtained using our algorithm were much more stable and there was a smaller difference between frames. We also applied temporal filters to the other methods to reduce the effects of pose jittering but the final pose was still affected badly by ambiguous poses nearby.

VI. Conclusion

In this study, we proposed a stable pose estimation algorithm based on an analytical motion model, which is suitable for real-time applications. Our proposed motion modeling method can be used with our proposed algorithm and other pose estimation algorithms. This method reduces the effects of pose jittering dramatically. The correct pose can be selected from two candidate ambiguous poses using the motion model, so the problem of pose jumping is overcome effectively.

To the best of our knowledge, this is the first study to combine a pose estimation algorithm with an analytical motion model. Several pose estimation applications are processed during video formatting, so we cannot estimate the pose simply by considering the information from one frame. The implementation of the Kalman filter in the analytical motion model means that the pose derived during each time step is more consistent with the previous pose. The users of these applications will feel more comfortable with the much smoother pose sequences obtained with our method.

References

[1] R. Azuma, “A survey of augmented reality,” *Presence: Teleoperators and Virtual Environments*, vol. 7, pp. 355–385, 1997.

[2] H. Kato and M. Billinghurst, “Marker tracking and hmd calibration for a video-based augmented reality conferencing system,” in *Augmented Reality, 1999. (IWAR ’99) Proceedings. 2nd IEEE and ACM International Workshop on*, 1999, pp. 85–94.

[3] H. Araujo, R. L. Carceroni, and C. M. Brown, “A fully projective formulation to improve the accuracy of lowes pose-estimation algorithm,” *Computer Vision and Image Understanding*, vol. 70, pp. 227–238, 1998.

[4] C.-P. Lu, G. D. Hager, and E. Mjolsness, “Fast and globally convergent pose estimation from video images,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 6, pp. 610–622, Jun. 2000. [Online]. Available: http://dx.doi.org/10.1109/34.862199

[5] A. Ansar and K. Daniilidis, “Linear pose estimation from points or lines,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 578 – 589, May 2003.

[6] D. Oberkampf, D. F. DeMenthon, and L. S. Davis, “Iterative pose estimation using coplanar feature points,” *Computer Vision and Image Understanding*, vol. 63, no. 3, pp. 495 – 511, 1996.

[7] G. Schweighofer and A. Pinz, “Robust pose estimation from a planar target,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 2024–2030, Dec. 2006.

[8] P.-C. Wu, J.-H. Lai, J.-L. Wu, and S.-Y. Chien, “Stable pose estimation with a motion model in real-time application,” *2012 IEEE International Conference on Multimedia and Expo*, vol. 0, pp. 314–319, 2012.

[9] G. Welch and G. Bishop, “An introduction to the kalman filter,” *Technical Report TR 95-041, University of North Carolina, Department of Computer Science*, 1995.

[10] Blender, “A free and open-source 3d computer graphics software product.” http://www.blender.org/, accessed March 29, 2014.

[11] D. Wagner and D. Schmalstieg, “Artoolkitplus for pose tracking on mobile devices,” in *Proceedings of 12th Computer Vision Winter Workshop (CVWW’07)*, 2007, pp. 139–146.