Integrating Virtual Objects into Real Images for Augmented Reality

Chu-Song Chen[†], Yi-Ping Hung[†], Shen-Wen Shih[*], Chen-Chiung Hsieh[*], Chen-Yuan Tang[†], Chih-Guo Yu[†], You-Chung Cheng[†]

[†] Institute of Information Science
Academia Sinica
Taipei, Taiwan

[*] Dept. Computer Sci. and Info. Eng.
National Chi-Nan University
Nan-Tou, Taiwan

[*] Multimedia Laboratory
Institute for Information Industry
Taipei, Taiwan

hung@iis.sinica.edu.tw

1. ABSTRACT

A systematic approach for integrating virtual objects into real images is developed in this paper. We propose the *P3P-ICP* method to solve the camera pose estimation problem. A robust tracking method is developed via the combination of the *LMedS* technique and the *P3P-ICP* method. With the 3D models and the robust tracking methods, we can determine the camera poses associated with each frame in the image sequence. Knowing the camera poses for each image frame, we can then integrate virtual objects into a video segment.

1.1 Keywords

Augmented reality, computer graphics, computer vision.

2. INTRODUCTION

A fundamental problem in augmented reality (AR) is to integrate virtual objects into real images. There are three major concerns in the integration task for generating realistic scenes:

(1) *Geometrical consistency*: If we freely move the virtual object in a scene image, geometrically correct integration should be generated during the free motion. For example, the sizes of the virtual objects have to be correctly generated, and the occlusion effects have also to be faultlessly presented.

(2) *Motion consistency*: If we integrate a virtual object into an image sequence, the motion of the virtual object should be consistent with that of the objects contained in the scene images.

(3) *Photometry consistency*: The shadows of the virtual objects have to be corrected generated.

Both (1) and (2) require the estimations of the camera positions and orientations with respect to the reference objects contained in the scene; while the problem of (3) requires further considerations about the lighting conditions of the environment. In this paper, we focus on the problems of (1) and (2). Nevertheless, our approach also has great potential to be generalized for solving the problem of (3) because the important objects contained in the scene can be reconstructed.

We divided the work into three parts: a *semi-automatic modeling* part, a *model-based re-calibration and tracking* part, and an *integration* part. The semi-automatic modeling part is an off-line process. A user can reconstruct the objects in the scene via a human-machine interaction interface directly from some *reference images*. The reference images can be either some of the images contained in the image sequence into which a virtual object will be integrated, or some stereo image pairs off-line captured. The purpose of this part is to reconstruct the important objects (which may be the foregrounds in integration or may be used for building an object coordinate system) by using as small as possible number of reference images. The model-based camera pose estimation and tracking part is operated based on the knowledge that a rough 3D shape of the important object models contained in the scene is given. However, the position and orientation of the model with respect to the scene image is still unknown. In model-based camera pose estimation, we try to solve the problem of finding the pose of the camera by giving a set of correspondences of 3D points and 2D features in an image. This problem is also referred to as the *PnP* problem in the vision community [7][4]. In this paper, we proposed a new approach – the

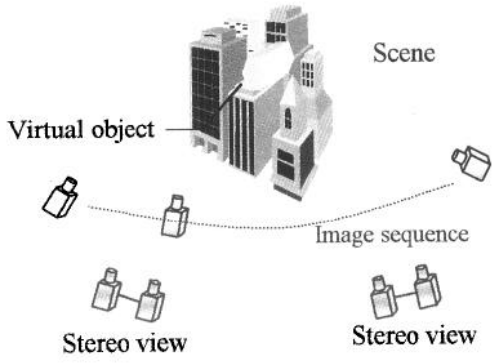


Figure 1: An image sequence taken from a scene into which virtual objects are to be integrated. The stereo views are used for building the 3D models of the important objects contained in the scene.

P3P-ICP approach, to solve this problem in a reliable way. In model-based tracking, temporal registrations of the model and the features in the images should be found. In this paper, we developed a model-based tracking strategy combining the *P3P-ICP* approach and the *LMedS* robust estimator [16]. Via the model-based re-calibration and tracking procedures, the camera poses with respect to the object coordinate system can be estimated, and hence a virtual object can be integrated with consistent geometry and motion if an initial pose of the object was given. In the integration part, to handle the shading and the occlusion of the virtual objects is a difficult problem. An advantage of our 3D modeling approach is that, by creating an equivalent 3D scene of the image into which virtual objects will be integrated, the shading and occlusion effects can be easily handled using an existing graph software (e.g., *OpenGL*).

Recently, another class of AR approaches which use *affine* or *projective* spaces for the integration of virtual objects (e.g., [9][17]) were proposed. They are referred to as *calibration free* approaches. In our opinion, advantages of these approaches include the allowance of less priori information and easy for implementation. However, they also suffer from some drawbacks. For example, these approaches require users to manually draw some points of the virtual objects in the first two frames of an image sequence (which can indeed be treated as a human-aided calibration). Hence, the quality of integration may depend on the accuracy of the human draw to some extent. Also, some photometry effects (such as shadows) are difficult to be handled because, to human imagination, virtual objects are basically characterized in the Euclidean space. In this paper, we use a calibration and 3D modeling approach, which allows users to perform integration naturally and directly.

This paper is organized as follows: Section 3 introduces the 3D object modeler used in this paper. Section 4 introduces the camera pose estimation method. Section 5 describes the model-based tracker developed for AR in this paper. Section 6 introduces the integration and graphics module. Some experimental results are given in Section 7. Finally, Section 8 gives some conclusions and discussion.

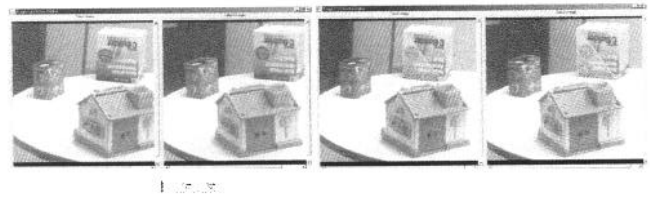


Figure 2: An example of some operations of our 3D object modeler. (a) The editing of the corresponding points. The epi-polar line is shown in the right image as a good hint, when the operator selected a point in the left image. (b) Connecting some points to form a polygon.

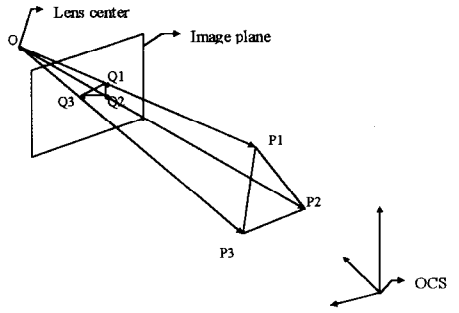
3. 3D OBJECT MODELER

The input of this modeler is a set of *stereo views* and their camera parameters (including both the intrinsic parameters and the extrinsic parameters between the left and right images of a stereo view). In our work, the stereo views can be either some reference frames of an image sequence, or some newly captured images of the same scene, as shown in Figure 1. In the former case, the camera parameters can be obtained via camera calibration [1]; while in the later case, the camera parameters can be computed using self-calibration techniques [11][14]. The stereo views should allow us to reconstruct all the important objects contained in the scene in the Euclidean space. An operator can edit the mesh structure of objects by interactively providing the computer the following information:

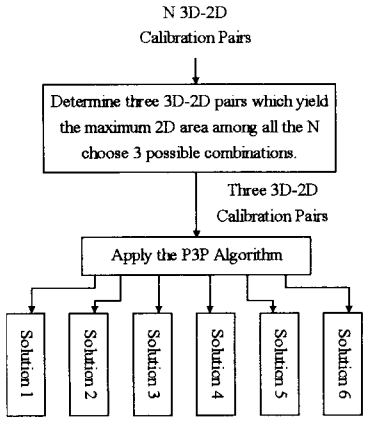
1. The *stereo-correspondence* information: An operator can select an image point in the left image, and specify its correspondence in the right image.
2. The *structural* information: An operator can *specify* that which image points should be connected to form a polygon.

After the specification of the stereo-correspondence information, the 3D coordinates of each pair of image points can be computed. These points can then be connected to form a 3D mesh since the structural information has also been given. The Arun et al. method [1] was used to compute the rigid-transformation when the 3D data obtained from multiple stereo views has to be integrated. Figure 2 shows an example of the operations of our 3D modeler. Practically, we usually edit a *rough* 3D model of the scene, that is, only sparse 3D points were reconstructed. It is because that a rough model is enough

[1] In this work, the Shih et, al. method [18] was used for calibration.



(a)



(b)

Figure 3: (a) The *P3P* problem. (b) Our method for solving the *P3P* problem.

for the application of AR.

4. RE-ESTIMATION OF CAMERA POSE

If the 3D model of the scene has been reconstructed, it can be used for re-estimation of the camera pose if the camera is used to grab the same scene from different poses. Given a set of 3D points and their corresponding image points, to find the least-mean-square rigid transformation by which the projection of the transformed 3D points coincide with their 2D image points, was called the P*n*P problem [7] if *n* point correspondences are used. Closed form solutions have been formulated if three feature points are considered [4]. However, if more point correspondences are used for camera pose estimations, closed form solutions do not exist. Also notable among pose computations are the method proposed by Lowe [12], Yuan [21] and Dementhon and Davis [5]. Both the Low and Yuan approaches used the Newton-Raphson method, thus a drawback is that an approximate pose has to be provided. An additional drawback is that, at each iteration step, the computation of the pseudoinverse matrix of a Jacobian is very computationally expensive. The Dementhon and Davis approach [5] assumed that the camera model is scaled orthographic and finds the rigid transformation by solving a linear system, and then used a POSIT procedure to iteratively refine the result. In this paper, we proposed the *P3P-ICP* approach to solve the P*n*P problem. In our approach, the *P3P* problem is solved for three selected seed points firstly via its close form solution. Then, the *iterative closest point* (ICP) algorithm is used to refine the estimated poses in an iterative way.

4.1 Estimation by solving the P3P problem

In the *P3P-ICP* approach, three *seed points* have to be selected. Here, we simply select three points that the triangle formed by them has the *largest area* because the larger the spread of the seed points, the smaller the affection of the 2D positioning error. Referred to Figure 3(a), assume that there are three points, P_1, P_2, P_3, whose coordinates are given with the *object coordinate system* (OCS). If their projections with respect to the *camera coordinate system* (CCS) are Q_1, Q_2, Q_3,, then the coordinates of P_1, P_2, P_3, with CCS are $\alpha_1\overrightarrow{OQ_1}$, $\alpha_2\overrightarrow{OQ_2}$, $\alpha_3\overrightarrow{OQ_3}$, respectively. If α_1, α_2, α_3 are computed, then finding the transformation between CCS and OCS (i.e,, the camera pose) is reduced to finding the least mean square rigid-transformation between two sets of 3D points, and can be computed via the Arun *et al,*. method [1]. From Figure 3(a), three equations can be listed as follows:

$$\left\|\alpha_1\overrightarrow{OQ_1} - \alpha_2\overrightarrow{OQ_2}\right\|^2 = \left\|P_1 - P_2\right\|^2, \quad (1)$$

$$\left\|\alpha_1\overrightarrow{OQ_1} - \alpha_3\overrightarrow{OQ_3}\right\|^2 = \left\|P_1 - P_3\right\|^2, \quad (2)$$

$$\left\|\alpha_2\overrightarrow{OQ_2} - \alpha_3\overrightarrow{OQ_3}\right\|^2 = \left\|P_2 - P_3\right\|^2, \quad (3)$$

There are three variables α_1, α_2, α_3, and three degree-2 nonlinear equations (1), (2), (3). They can be solved using the method given in [4]. Since six solutions can be solved with (1), (2), and (3) but only one of them is correct, we can use other 3D-2D correspondences to verify that which one is correct. Here, each of the six solutions are used as initial estimate of the *ICP* method described below, and the solution with the least residue error is served as the solution of the *P3P-ICP* approach.

4.2 Refinement using the ICP approach

Most nonlinear optimization methods require the estimation of gradients of the error function in the parametric space. Because the gradient computation is easily affected by noise, the obtained solution is easily trapped in a local minimum. In our work, the *ICP* method [3] is used, which requires no estimation of the gradients and can also converge to a solution with minimal residual error. Each iteration of the *ICP* algorithm contains two steps: (1) establishing point correspondences by finding the closest neighboring points, and (2) computing the rigid motion via a least-square error measure. In the Wunsch and Hirzinger approach [20], the *ICP* method was also used to register CAD models to images. However, in their

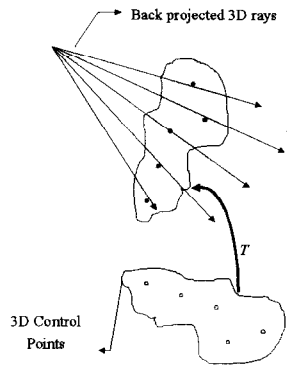


Figure 4: Using the ICP method to refine the camera pose for solving the PnP problem.

problem formulation, the correspondences between the 3D model points and 2D image points are assumed to be unknown in advance. On the other hand, in our work, the 3D-2D correspondences are available in advance, and an initial solution has also been given using the method described in Section 4.1. This additional constraint can be used to increase the reliability of our system. In the following, a method similar to that given in [20] is developed for the refinement of the initial solution obtained by solving the P3P problem.

Assume that $\{M_1, M_2, ..., M_N\}$ is a set of model points and $\{m_1, m_2, ..., m_N\}$ are their corresponding 2D image points, respectively. Initially, let T_0 be the transformation between the OCS and the CCS of the first frame.

1. Apply T_0 for the transformation, $M^*_i = T_0 M_i$.
2. In the 3D line $L(m_i)$, finding the closest point (namely, M'_i) of M^*_i, where $L(m_i)$ is the back projected 3D ray as shown in Figure 4.
3. Using the Arun et, al. method [1] to find the rigid-transformation T_1 that minimizing $e = \Sigma_i \|M'_i - T_1 M^*_i\|^2$.
4. If T_1 and the identity matrix are close enough, stop; Else, $T_0 \leftarrow T_1 T_0$, and go to 1.

According to our experience, the combination of the *P3P* algorithm and the *ICP* approach is a very reliable and robust method. Even the initial pose is quite different from the correct one, the *P3P-ICP* approach can successfully converge to the right solution without trapped in local minimum.

5. MODEL-BASED TRACKER

In our work, the *P3P-ICP* method was used to compute the camera pose of the first frame of an image sequence [2]. Once the camera pose of the first frame was computed, that of the other frames can be obtained using the model-based tracker. Hence, the model-based tracker can avoid intensive manual work of the associations of 3D-2D correspondences by automatically tracking the feature points of the 3D model and re-estimating the poses of the camera for each frames of the sequence. In the past, some methods [8][10] assumed a velocity model *in priori* and a Kalman filter was used to better predict the position of the image feature. Lowe [12] used a probabilistic criterion to guide the search for the best match, and numerical minimization is used to determine object rotation and translation. Unfortunately the use of a velocity model imposes regularity constraints on the camera motion. In [15], the indexing of aspect graph is used for tracking, and the aspect tables have to be constructed off line manually. Simon and Berger [2] used a robust estimation of the view point as well as the parts of the tracked image features that actually correspond to the model. The robust estimator adopted was the *M-estimator* in their approach.

In this work, we combine the *LMedS*, technique and the P3P-ICP method to obtain reliable tracking results. Our approach can deal with the case that features may disappear and re-appear in an image sequence. The tracking algorithm includes a hypotheses-generation stage and a verification stage. Assume that there are N model features in the 3D space, namely, $P_1, P_2, ..., P_N$, and their 2D corresponding points in current image are $p_1, p_2, ..., p_N$, respectively. In the hypotheses-generation stage, hypothesized matches of $p_1, p_2, ..., p_N$ in the next frame are found. Here, the *least squared-error block-matching* result is served as the hypothesized match of each feature. Denote the hypothesized matches found for $p_1, p_2, ..., p_N$ in the next frame to be $p'_1, p'_2, ..., p'_N$, respectively. Then, we used the *LMedS* technique to find the *most consensus class of matches* among all of the hypothesized matches. The *most consensus camera motion* (*MCCM*) is referred to as the rigid-transformation corresponding to the most consensus class of matches. The procedure of finding the *MCCM* is listed below.

Algorithm 1

1. Repeat K random trials (i.e., *iteration* = 1, 2, ..., K). Do steps 1.1-1.6:
   - 1.1 Randomly select three image points (p'_i, p'_j, p'_k) using the bucketing technique [22].
   - 1.2 Apply the 3D-2D correspondences (P_i, p'_i), (P_j, p'_j), (P_k, p'_k) to compute the object pose using the *P3P-ICP* approach. Assume that a rigid transformation $T_{i,j,k}$ is found.
   - 1.3 Apply $T_{i,j,k}$ to transform all of the N model features, $P_1, P_2, ..., P_N$ to new 3D positions $P''_1, P''_2, ..., P''_N$, where $P''_h = T_{i,j,k} P_h$, $h=1, 2, ..., N$.
   - 1.4 Compute the residue error $e_h = \|p'_h - p''_h\|$ for $h=1, 2, ..., N$, where p''_h is the projection of P''_h, onto the image plane.

[2] The initial correspondences (i.e. the correspondences between the model points and the 2D feature points of the first frame) were provided manually.

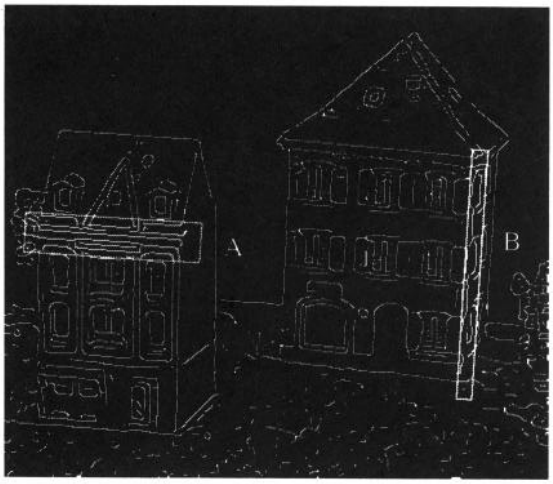


Figure 5: The edge image (canny edge) using a threshold. In regions A and B, there are full of edges. It is not easy to segment line features in these two regions.

1.5 Obtain the median residue error e_{mid} by sorting all e_h, $h=1, 2, ..., N$.

1.6 Record that $E(iteration) = e_{mid}$ and $T(iteration) = T_{i,j,k}$.

2 Let $E(t)$ be the least one among $\{E(iteration) \mid iteration = 1, 2, ..., K\}$. Let $T=T(t)$.

3 Re-compute T using the 3D-2D correspondences whose residues are smaller than $E(t)$, then T is the *MCCM*.

Algorithm 1 is a general approach which can deal with the model-based tracking problems where only 3D points are used to specify the model. In our experience, this algorithm had good performance if the object motion in the image sequence is translational. However, if there is some rotational motion in the image sequence, the correspondence may not be correctly found using block matching for only point features. Hence, we proposed another algorithm (*i.e.*, Algorithm 2) to overcome this problem. In Algorithm 2, line features in the wire frame model are used for the generation of more robust hypotheses. In the past, Deriche and Faugeras [6] and Zhang and Faugeras [22] have proposed line tracking methods which extracts line features by linking edge maps at first, and then tracking line features in the image sequence. Their methods can be referred to as a "linking followed by matching" approach. However, because extraction of line features contains feature grouping, clustering and fitting, it is not easy to segment line features in an image, especially, in an image containing rich textured objects. For example, Figure 5 shows an edge image obtained by using the canny edge detector. One can see that there are full of edges in regions A and B. In fact, it is difficult to perform correct edge linking or segmentation in these regions.

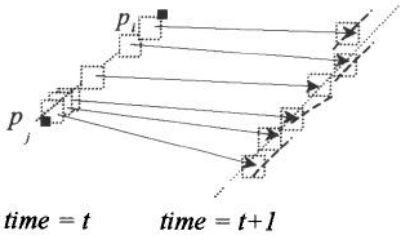


Figure 6: Finding the hypothesized matches using the masked block matching.

In this paper, instead of linking or grouping edge pixels in advance, we treat each neighborhood region of the model line points as a textured block, and find hypothesized line points in the next frame by performing block matching in the original image (rather than the edge map). As shown in Figure 6, consider lines p_i and p_j in the given model, we uniformly sample this line to a few points. Then, for each sampled point, find the corresponding point in the next frame using masked block matching [3]. Finally, we use the *LMedS* method to fit these corresponding points in a robust way in the next frame. The major advantage of our approach is that it can be applied not only for tracking line features with homogeneous neighborhood, but also for tracking line features with textured neighborhood. Notice that due to the aperture problem, the corresponding blocks found by block matching may not be correct. For example, some blocks may "shift" along the image line as shown in Figure 6. However, since the positions of the matched blocks are still lying on a line if the matching error is caused by aperture problem, the correct line can still be found by fitting those matched points. Hence, such a "matching followed by linking" strategy has better performance that those using a "linking followed by matching" strategy. The procedure of finding the *MCCM* by using the line features is listed below.

Algorithm 2

1. For each line segment of the wire-frame model

1.1 Assume that p_i and p_j are two end points of this line segment in the current frame.

1.2 Sample the line interval between p_i and p_j uniformly, and obtain points s_i, $i = 1, 2, N_s$.

1.3 For each s_i, find the hypothesized corresponding point, s'_i, in the next frame using the masked block matching.

1.4 Perform robust line fitting for $\{s'_i \mid i = 1, 2, N_s\}$ in

If we project the tracked 3D model to the current frame, the region covered by the projection is referred to as the *masked region*. Then, for each block across the boundary of the masked region, we give larger weights for the pixels in the masked region and smaller weights for the pixels not in the masked region. This method is similar to that used in [13], and is referred to as the *masked block matching* in this paper.

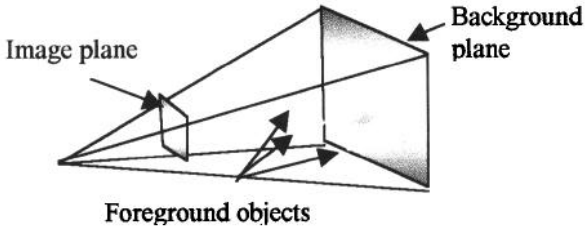


Figure 7: Equivalent 3D scene.

the next frame using the LMedS technique.

2. Find the corner points of the wire-frame model by computing the intersection points of joined lines. Let $p'_1, p'_2, ..., p'_N$ be all of the corner points.
3. Apply **Algorithm 1** to obtain the *MCCM*, *T*.

A similar method proposed by Armstrong and Zisserman [2] also used "matching followed by linking" strategy. Their method first selected a few control points from a primitive (*e.g.*, line), and then a 1D search is carried out perpendicular to the projected outline to find the strongest image edge in the next frame for each control point. Nevertheless, such a strategy may also suffer from wrong matching of the control points which have rich textured neighbor region because there may be too many strong edges in the neighborhood.

The tracking module in our work was designed to achieve temporal registration on long sequences without human interaction. However, automatic tracking of objects contained in an image sequence remains a very difficult problem. Hence, appropriate human interaction is necessary for correcting the tracking results. In our work, an operator can verify the tracking result by viewing the alignment of the model and the image sequence, and can modify the 3D-2D correspondences of any images and then re-start a *P3P-ICP* and tracking process from this image. We are now developing a more intelligent human-machine interaction interface which can verify its own performance and send alarms of the problematic images to the operator.

6. INTEGRATION AND GRAPHICS

To integrate a virtual object into real images, we separate the work into two parts:

1. Building an interactive interface allowing a user to dynamically control and move a virtual object in a static scene with respect to the object coordinate system.
2. If an initial integration has been given via the interface, a virtual object can then be integrated into a sequence automatically following the rigid-transformation generated by the tracking and re-calibration processes.

The integration module was designed that the existing graphic software can be directly applied. It is because that the shading, texture-mapping, and occlusion processing tools are all well-prepared in these software, the rendering of the virtual objects can be faster and easier. We created an *equivalent 3D scene* (E3DS) of the static scene, as shown in Figure 7. With this, an equivalent image (which is the same as the scene image) can then be produced by mapping the textures onto the E3DS and observed it from the same view.

7. EXPERIMENTAL RESULTS

In the first experiment we integrate a virtual object into a static scene. In particular, we apply our system for the integration of a virtual object into a stereo image pair. Figure 8(a) is the left image of a stereo image pair into which virtual objects are to be integrated. Using the 3D modeler introduced in Section 3, some important objects contained in the scene can be reconstructed via stereo vision, as shown in Figure 8(b). Figure 8(c) shows the reconstructed model observed from a slightly different view. The reconstructed object model can then be served as the foreground objects for building the E3DS (equivalent 3D scene). After the construction of the E3DS, a user can then freely move a virtual object (a teapot) on the table contained in the scene because we have also compute the plane equation of the table by using a specified triangle during modeling. Figure 8(d) and Figure 8(e) show two different integration results when we freely move the teapot on the table plane. Since the partial-occlusion and the changes of sizes effects can be correctly presented during the human-controlled moving of the teapot, the user can interact with the real world in a natural way.

In the next experiment, we integrate a virtual object into an image sequence containing a moving truck. First, a 3D model of the truck was created using the 3D object modeler from a stereo image pair, as shown in Figure 9(a). Then, the camera pose of the first frame of the image sequence was re-estimated based on the reconstructed model using the *P3P-ICP* approach. After that, the 3D model can be re-projected onto the image plane of the first frame. The other images contained in the sequence were tracked via the robust tracking process (Algorithm 1 was used in this experiment), and part of the tracking results are shown in Figure 9(b) by aligning the moving model with the images. It can be observed that good temporal registrations of the model and the images can be obtained via the tracking process. Finally, Figure 9(c) shows the integration of a virtual box on the top of the truck. To demonstrate the usefulness of our tracking algorithm for dealing with the partial-occlusion case, Figure 10(a) shows part of a sequence containing a moving truck which is occluded by a foreground object (a tree). In this sequence, model features could be disappear and re-appear. However, using our tracking algorithm, the 3D model of the truck can be successfully tracked in 3D space, as shown in Figure 10(b). Figure 10(c) shows the integration results.

In the above sequences, the motion type is roughly translational. In the last experiment, a moving camera was used to grab a sequence of a static scene. The motion contained in this sequence includes not only translation but also rotation. Since it is not easy to track a rotational motion using only point features, we adopted Algorithm 2 for visual tracking in this experiment, which can keep tracking the wire-frame model successfully. Figure 11 shows the integration results of this sequence.

8. CONCLUSIONS AND DISCUSSION

In this paper, we successfully developed a systematic approach which can integrate virtual objects into real images. Many useful modules in 3D computer vision, including 3D object reconstruction, model-based camera pose estimation and model-based tracking, were appropriately integrated in our approach. The 3D modeling approach has the advantage that the virtual can be integrated by human in a direct and natural way, and the geometrical consistency and the partial-occlusion effect can be easily handled by creating an equivalent 3D scene. We proposed a new model-based calibration method – the *P3P-ICP* method, which can solve the general *PnP* problem. A new robust model-based tracking method was also developed via the combination of the *LMedS* technique and the *P3P-ICP* approach. Our method requires no assumptions of motion models, and the most consensus camera motion can be found.

9. ACKNOWLEDGMENTS

This work was partly sponsored by MOEA and supported in part by Institute for Information Industry, R. O. C.

10. REFERENCES

[1] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-Square Fitting of Two 3-D Point Set," *IEEE Trans. PAMI*, Vol. 9, pp. 698-700, 1987.

[2] M. O. Berger and G. Simon, "Robust Image Composition Algorithms for Augmented Reality," *Proc. Asia Conf. Computer Vision*, pp. 360-367, 1998.

[3] P. J. Besl and N. McKay, "A Method for Registration of 3-D Shapes," *IEEE Trans. PAMI*, Vol. 14, pp. 239-256, 1992.

[4] D. DeMenthon and L. S. Davis, "Exact and Approximate Solutions of the Perspective-Three-Point Problem," *IEEE Trans. PAMI*, Vol. 14, pp. 1100-1105, 1992.

[5] D. F. Dementhon and L. S. Davis, "Model-Based Object Pose in 25 Line of Code," *International Journal of Computer Vision*, 15, pp. 123-141, 1995.

[6] R. Deriche and O. Faugeras, "Tracking Line Segments," *Image and Vision Computing*, Vol. 8, pp. 261-270, 1990.

[7] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Catography," *Communications of ACM*, Vol. 24, pp. 381-395, 1981.

[8] D. Gennery, "Visual Tracking of Known Three Dimensional Objects," *Intl. Journal of Comp. Vision*, 7(3), pp. 243-270, 1992.

[9] K. N. Kutulakos, J. R. Vallino, "Calibration-Free Augmented Reality," *IEEE Trans. Visualization and Computer Graphics*, Vol. 4, pp. 3-20, 1998.

[10] D. Koller et, al., "Automated Camera Calibration and 3D Egomotion Estimation for Augmented Reality Applications," *Proc. International Conference on Computer Analysis of Images and Patterns*, Kiel, Germany, pp. 199-205, 1997.

[11] Q-T. Luong and O. D. Faugeras, "Self-Calibration of A Moving Camera from Point Correspondences and Fundamental Matrix," *Intl. Journal of Comp. Vision*, 22(3), 261-289, 1997.

[12] D. G. Lowe, "Robust Model-based Motion Tracking Through the Integration of Search and Estimation," *Intl. Journal of Comp. Vision*, 8:2, pp. 113-122, 1992.

[13] T. Mitsunaga, T. Yokoyama, and T. Totsuka, "AutoKey: Human Assisted Key Extraction," *Proc. SIGGRAPH*, Los Angeles, USA, pp. 265-272, 1995.

[14] M. Pollefeys, R. Koch, and L. V. Gool, "Self-Calibration and Metric Reconstruction in spite of Varying and Unknown Internal Camera Parameters," *Proc. International Conference on Computer Vision*, Bombay India, pp. 90-95, 1998.

[15] S. Ravela, et, al., "Tracking Object Motion Across Aspect Changes for Augmented Reality," *Proc. ARPA Image Understanding Workshop*, pp. 1345-1352, August, 1996.

[16] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection.* Wiley, New York, 1987.

[17] Y. Seo, et al., "Video Augmentation by Image-Based Rendering under The Perspective Camera Model," *Proc. International Conference on Pattern Recognition*, Australia, pp. 1694-1996, 1998.

[18] S. W. Shih, Y. P. Hung, and W. S. Lin, "Accurate Linear Technique for Camera Calibration Considering Lens Distortion by Solving an Eigenvalue Problem," *Optical Engineering*, Vol. 32, pp. 138-149, 1993.

[19] G. Simon and M. O. Berger, "A Two-Stage Robust Statistical Method for Temporal Registration from Features of Various Type," *Proc. International Conference on Computer Vision*, Bombay, India, pp. 261-266, 1998.

[20] P. Wunsch and G. Hirzinger, "Registration of CAD-Models to Images by Iterative Inverse Perspective Matching," *Proc. International Conference on Pattern Recognition*, pp. 78-83, 1996.

[21] J. C. Yuan, "A General Photogrammetric Method for Determining Object Position and Orientation," *IEEE Trans. Robotics and Automation*, Vol. 5, pp. 129-142,

1989.

[22] Z. Zhang and O. Faugeras, *3D Dynamic Scene Analysis, A Stereo Based Approach*, Springer-Verlag, Berlin Heidelberg, 1992.

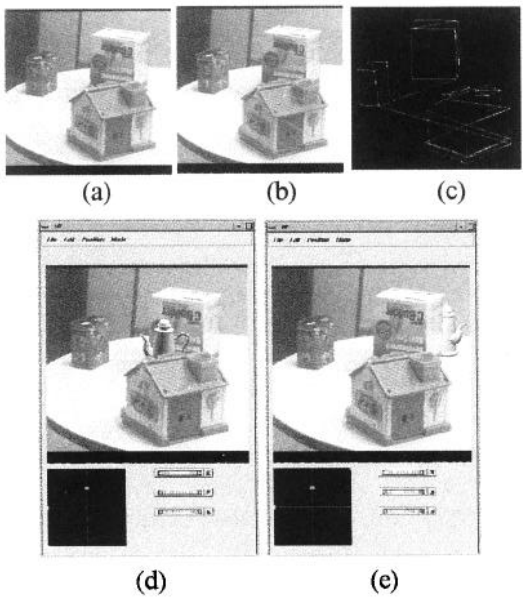


Figure 8: (a) The left image of a stereo pair into which virtual objects are to be integrated. (b) The 3D model reconstructed using the 3D object modeler. (c) The reconstructed 3D model observed from a different view point. (d) (e) show the freely integration results from two different object poses.

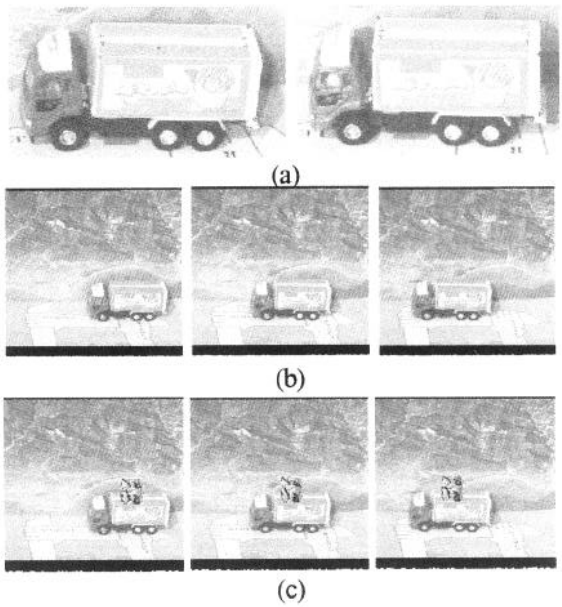


Figure 9: (a) The stereo image pair used to build a rough 3D model of the truck. (b) The tracking result of the sequence containing a moving truck. (c) The integration of a virtual object (a textured block) into this sequence.

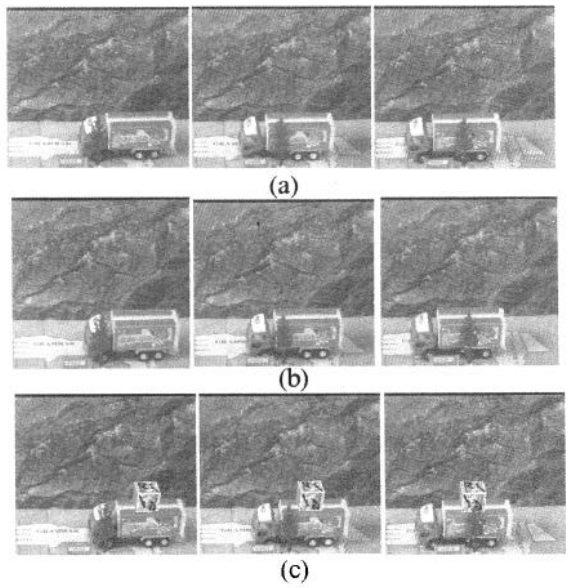


Figure 10: (a) A sequence containing a moving truck and a foreground object (a tree). (b) The tracking results. (c) The integration results.

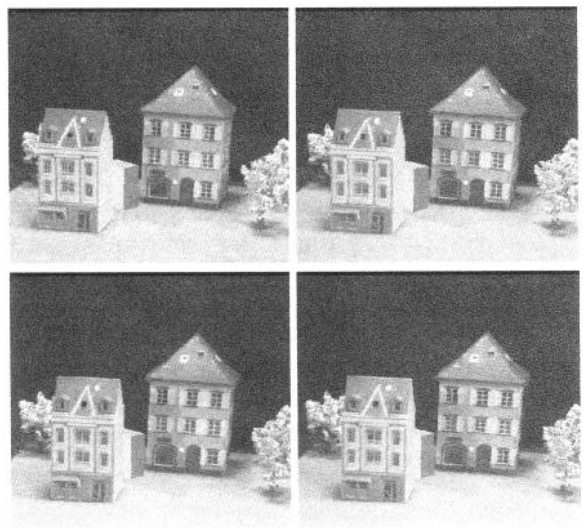


Figure 11: The integration results of the image sequence captured with a moving camera.