AN INTERACTIVE VISION-BASED TOOL FOR MODEL-BASED SCENE CALIBRATION OF AUGMENTED REALITY ENVIRONMENTS

Pierre A. J. Bayerl[1]

Gregory Baratoff

Pierre Bayerl
Abteilung Neuroinformatik,
Universität Ulm, Fakultät für Informatik
D-89069 Ulm, Germany
Pierre.Bayerl@neuro.informatik.uni-ulm.de

Gregory Baratoff
Virtual Reality Competence Center
DaimlerChrysler Research & Technology
89077 Ulm, Germany
Gregory.Baratoff@daimlerchrysler.com

ABSTRACT

Non-trivial Augmented Reality environments consist of a reconfigurable set of objects whose spatial relations need to be calibrated upon setup. Since manual measurement is tedious, time-consuming, and error-prone if not done carefully, fully or semi-automatic procedures are needed. We have developed a semi-automatic vision-based scene calibration tool which allows a user to perform this task efficiently and accurately. For each scene object to be calibrated, the user interactively specifies its approximate pose by aligning a 3D model of the object with its projection in one or several camera image(s) of the scene. Thereupon, the exact pose is automatically computed by a combination of computer graphics and computer vision techniques. The combined knowledge of the object model and of the initial pose estimate allows the system to significantly improve the feature matching and pose estimation processes. Because the latter is based on a robust estimation method, the system performs well even in cluttered environments where objects are partly occluded.

Keywords: augmented reality, calibration, computer vision

1 INTRODUCTION

In Augmented Reality (AR) environments, real scenes are augmented with virtual objects. This is achieved by overlaying a user's view onto the scene (e.g. perceived by means of a head-mounted display (HMD)), with properly aligned, computergraphically generated, images of the virtual objects. Many interesting AR applications such as interior decoration [Tamur99], architectural modelling [Klink99], and industrial maintenance [Klink99], become possible once virtual objects can be placed in fixed spatial relation to real ones (e.g. next to, in front of, on top of, etc.). For such spatially correct positioning to be possible, the user's pose (position and orientation) and that of relevant real objects must be known in a fixed reference frame.

Typical scenes we work with do not consist of a single rigid object, but of several distinct ones that can be configured in different ways. If we want to be able to place virtual objects relative to them, all of them have to be tracked. For those objects to which fiducial markers can be attached, tracking can easily be achieved, e.g. using a marker-tracking approach based on the ARToolkit software [Kato99]. In this paper, however, we focus on those objects to which fiducial markers can not be attached, be it because they are too small, because their surfaces do not permit markers to be attached, because the markers would obscure information of interest to the user, or because of aesthetic factors. For this class of objects and applications, we have opted for

[1]This work was performed at the VRCC, DaimlerChrysler Research & Technology, Ulm, Germany

a non-intrusive vision-based approach where the object's pose is computed from the locations of the object's natural features detected in camera images. In Section 2 we give an overview of our semi-automatic off-line procedure for determining the configuration (poses) of the known rigid objects which make up the stationary part of the scene. We call this process - as well as its result - *scene calibration.*

In order to track the pose of an object based on natural features, a model of the object is needed that (a) allows the visual appearance of the object's features to be predicted in a form that can be used for matching against image features, and that (b) contains the geometric information necessary for pose estimation. Currently, our system is based on polyhedral object descriptions extracted from CAD models. Furthermore, we have chosen the model's vertices as the natural features to be matched. In Section 3 we describe how the local appearance of a model vertex is synthesized with the polyhedral description, and how it is used to find the corresponding image position using a template-matching strategy.

Many real-time visual tracking systems [Harri92, Armst95, Drumm00] use edges as natural features mainly for performance reasons. In contrast, our choice of image templates as features was motivated by considerations of useability and extensibility. In the model-creation phase it is much easier for a user to select a vertex as a feature to be matched than to pick a point on an edge. Furthermore, the template-based approach allows textured regions to be selected as features. This is particularly interesting when one moves away from polyhedral descriptions to less geometrically structured and more appearance-based object models where natural surface texture plays a more important role.

In our system, the user is asked to initially specify an approximate pose for each object of interest. This interactive specification is a crucial aspect of our system, since besides reducing the search space of the feature matching stage, it also stabilizes the model fitting by providing an initial pose estimate not too far from the correct solution.

Once a correspondence between model and image features has been established, the object's pose is determined by fitting the model to the image features. The robust estimation method described in Section 4 allows the objects' poses to be determined even in cluttered environments where objects are partially occluded by each other or by unknown scene elements.

We have performed many experiments with both synthetic and real data to assess the accuracy of our method. The results are summarized in Section 5. Section 6 provides our conclusions and an outlook for future work.

2 SYSTEM OVERVIEW

In our system the user sees the real world through a video see-through HMD. Superimposed on this view are the projected models of the real objects to be calibrated. The model visualization is done using the OpenInventor library[2] , with the video being rendered by an OpenGL drawpixel command.

For each object, the user interactively specifies the approximate pose using a 3D input device such as the Magellan Spacemouse[3] by roughly aligning the projections of the object model and the real object in the camera image. This process can be repeated for images taken from different positions. Using this initial pose the system automatically computes the exact pose by a combination of computer graphics and computer vision techniques. The initial pose information is used

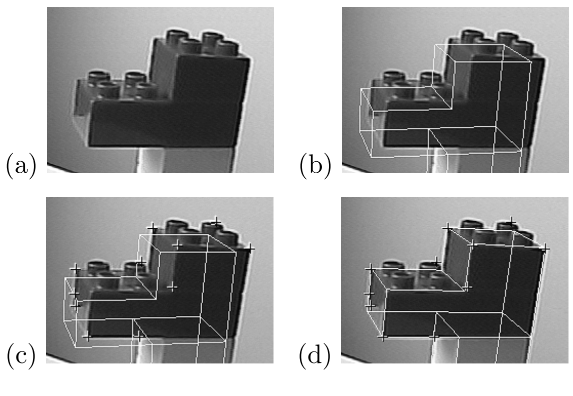


Figure 1: Typical calibration procedure : (a) presentation of original image (b) interactive specification of the initial pose by the user (model shown in wire-frame) (c) automatic feature extraction (detected features indicated by crosses) (d) automatic outlier rejection and pose estimation (wire-frame model "snaps" on to the real object).

to make a prediction of feature locations in the images and to synthesize the features themselves. Starting from the predicted locations the exact positions of the features in the images are refined. The obtained data are point correspondences between model points and feature points.

[2]oss.sgi.com/projects/inventor or www.tgs.com
[3]www.logicad3d.com/products/Classic.htm

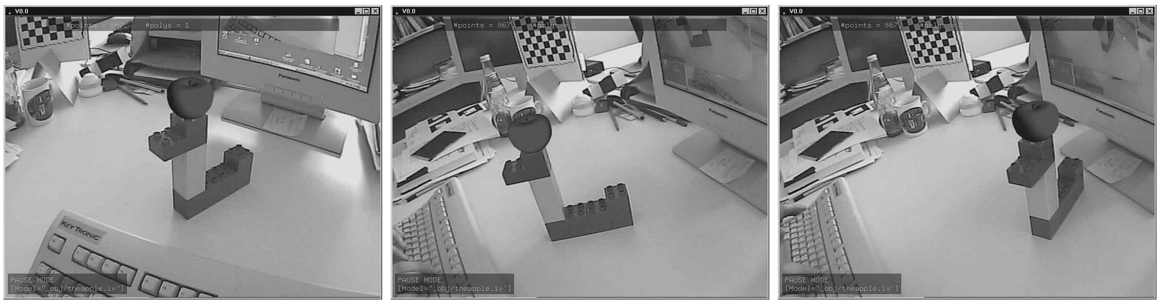


Figure 2: A virtual apple on the head of a real "lego duck" – from different viewpoints and with different positions of the duck

The next step is to fit the model of the object to the obtained data, which contains noise and also a significant percentage of outliers. A robust method proposed by Fischler and Bolles [Fisch81] for single images combined with a nonlinear optimization over feature locations in one or more images are used for this task. The calibration procedure is illustrated in Figure 1.

Once an object's pose has been determined by the system, it becomes possible to place virtual objects in spatial relation to it. In Figure 2 a virtual apple has been placed on the head of a real "lego duck" calibrated using the method presented above. The first two images show the augmented scene from two different viewpoints, showing the correct positioning of the apple in 3D relative to the duck. The third image shows the scene after the duck has been moved and recalibrated in its new position, with the virtual apple moving along.

3 FEATURE SYNTHESIS AND DETECTION

In this section we present a method to synthesize pose-dependent image features and to find them in the image. First of all, to compensate for light and shading effects and color variations, the gradient image of the gray-converted original image is used as search space instead of the original color image. Within this image a real object will look like a wire frame. With the given pose and the known model one can roughly determine the shape of the real object as a wire frame. For each visible corner of the model a small patch of the projection of the wire frame is generated. These corners are then used as features and the created patches are templates representing these features, respectively, in the gradient image. Then the normalized cross-correlation of the gradient image and the synthesized template is calculated. The location of the maximum value of the normalized cross-correlation yields a potential location of the feature within the image. The search area is restricted around the predicted position of the feature given by the initial pose.

Gradient image creation: The original image I is blurred with a 5×5 binomial filter to reduce noise. Then the gradient magnitude is computed from a horizontal and vertical sobel operator [Trucc98]. Afterwards, a threshold is applied to suppress small values of the gradient image. This eliminates small intensity variations in the original image due to texture or noise and yields a better approximation of the "wire frame"-representation.

Template creation: In order to synthesize the templates, one must first know the visible corners of the selected object with the given pose. To solve this problem, for each model point (or corner) one can consider the ray through the camera viewpoint and the corner itself. The model point is invisible if this ray intercepts any polygon of the model on the range between these two points. Otherwise the point is visible. For each visible corner, the visible parts of the edges have to be determined, starting from the corner. Several solutions for this problem statement can be found in standard computer graphics literature [Foley96]. We simplified the problem, in so far that if a point on an edge at a certain distance d from the tested model point is visible, the part of the edge between these two points is assumed to be visible, too. We have chosen $d \ll$ the mean edge length. Optimally d should be chosen in such a way that the projection of the line segment defined by d just fits into the template.

The projection of these edges are drawn into the template using the Bresenham line algorithm. The point z within the template represents the corner itself. One possibility is to place z at the center of the template. To prevent a waste of template space we optimize the position of z to achieve a better distribution of the edge projections in the template (see figure 3).

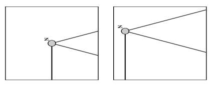


Figure 3: Centered and optimized corner template

Feature detection: To locate the features a normalized cross-correlation of the gradient image and the template is performed. The location of the feature in the image is supposed to be near the initial guess for the projected model points, deduced from the pose specified by the user. The actual location is obtained by seeking the maximum correlation value within a certain area around the initial guess for the feature position.

The resulting data are point correspondences from model points to feature points in the images for each visible corner in each image. In figure 4 the templates for a given model are superimposed on the detected locations in one image of the object.

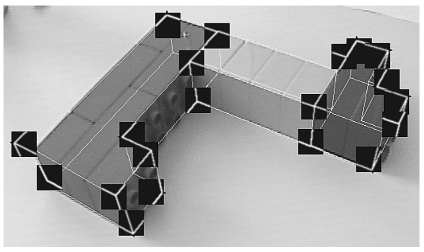


Figure 4: located templates

4 MODEL FITTING

Given the perspective projection of a certain number of points for a known model in 3D space and using a pin-hole model for the camera, it is possible to determine numerically or analytically the rigid transformation (rotation and translation) relative to the camera reference frame [Haral89, Haral91, Horau89, Madse97]. With three points there can be up to four analytical solutions in front of the viewpoint [Haral91]. With some more information (e.g. another point or a given approximate pose) one can filter out the correct solution. The problem with such a small number of points is the susceptibility to errors in the input data. Using numerical methods with more points ($N \gg 3$) gaussian noise can be cancelled, but outliers still remain as big error sources.

4.1 Identifying outliers

An effective method to handle outliers is the RANSAC algorithm proposed by Fischler and Bolles [Fisch81]. As an application they were fitting models on projective data, like the data obtained in the previous section. The algorithm is based on point samples of minimal size (three points), for which a fit with the model can be achieved. These point samples are randomly chosen from the results of the feature detection stage. In each iteration of the algorithm a new random sample is chosen and the possible poses are computed. For each fitted pose of each sample, those points, from the set of detected features, which match the transformed model within a certain error tolerance are counted. The sample with the largest number of points is regarded as the best. The set of points determined by this sample, the points matching the model with the fitted pose, can then be used to compute the real transformation of the object. As described in [Fisch81], the maximum number of iterations needed to obtain – with a certain probability – at least once a sample for which the resulting pose is "correct" can be estimated relative to the probability p of picking a "good" point (a point which is consistent with the "correct" pose). E.g., for $p = 0.5$, 75 iterations are needed to obtain with a probability of 0.9999 at least one "good" sample. The probability $1 - p$ can be seen as the portion of outliers among the point correspondences. It is worth noting that the number of iterations is unrelated to the size of the input data.

4.2 Fit using one image

Once a subset of the input data without outliers is determined, the pose is estimated using a nonlinear optimization based on the newton method as for example described in [Trucc98]. With this method the distance of the feature points to the projected points of the transformed model is minimized subject to the six DOFs (rotation and translation).

4.3 Fit using several images

To accomplish the fit over more than one image, first the external camera parameters, which represent the pose of the camera, have to be determined for each camera viewpoint from which the images were taken. Then, from each view the outliers are excluded using the RANSAC-algorithm. Afterwards, the remaining data points of all the images are fitted to the model with a modification of the nonlinear optimization used for the fit with one image as described above.

Determination of the external camera parameters: With more than one image a global reference frame is needed. With one image we implicitly used the center of projection and the camera orientation as reference frame. With more images we need to be able to deduce the pose of the camera from each image. For this purpose two different approaches where employed: (a) The use of the ARToolkit [Kato99], a real-time marker-based pose tracking software. But since scene calibration does not require real-time operation, we are able to use more accurate methods to determine the pose of the camera: (b) The use of a reference object which is modelled the same way as any other object in the system. This reference object has to be positioned for each view (using the RANSAC-algorithm and the nonlinear optimization) before calibrating any other object of the scene, to tell the system where the camera is located.

Modification of the nonlinear optimization: The method described in [Trucc98] is based on the difference $\mathbf{p}_i - \tilde{\mathbf{p}}_i$ of projections $\mathbf{p}_i = (x\,y)^T$ of the transformed model points $\mathbf{P}_i^M$ in camera coordinates taken from a single view (see equation 1 beneath) and the input data points $\tilde{\mathbf{p}}_i = (\tilde{x}\,\tilde{y})^T$ (the detected location of the features). Since - using a single image like in section 4.2 – the camera is located at the origin of the world coordinate system, the camera coordinates are the same as the world coordinates $\mathbf{P}_i^W$. The transformation is defined by six parameters, three for the translation $\mathbf{T} = (T_1\ T_2\ T_3)$ and three for the rotation R, represented by the Euler angles $\theta_1, \theta_2, \theta_3$.

$$\mathbf{P}_i^W = R\mathbf{P}_i^M + \mathbf{T} \tag{1}$$

Further the partial derivatives of the projected coordinates $\frac{\partial \mathbf{p_i} - \tilde{\mathbf{p}}_i}{\partial T_j}$, $\frac{\partial \mathbf{p_i} - \tilde{\mathbf{p}}_i}{\partial \theta_j}$ are needed for the nonlinear optimization.

The modification consists in integrating the external camera parameters into equation 1 and in working with the projections of all the different viewpoints. The external camera parameters are defined by an orientation R^k (three orthonormal base vectors defining a rotation matrix) and a translation C^k. With this information the transformed model points in camera coordinates with respect to camera k are computed as follows:

$$\mathbf{P}_i^k = R^k(R\mathbf{P}_i^M + \mathbf{T} - C^k) \tag{2}$$

Let the projection of $\mathbf{P}_i^k$ be $\mathbf{p}_i^k$ and the corresponding input point $\tilde{\mathbf{p}}_i^k$. Like in the original method, the difference of the projection and the input data $\mathbf{p}_i^k - \tilde{\mathbf{p}}_i^k$ has to be minimized, but now we are using the data of all images. For simplicity the derivatives needed for the nonlinear minimization of this difference were computed numerically as described in [Hartl00]. Finally, we replaced the Newton method by the more stable Levenberg-Marquard method[Hartl00].

4.4 Camera distortion

It is very important in the context of scene calibration using camera images to mention the effects of camera distortion. For the experimental results we used an off-the-shelf camera[4]. The distortion of such cameras is too big to get an accurate pose estimation (depending on the employed lens). So the detected feature locations were undistorted using a radial distortion model such as in [Tsai87].

5 RESULTS

In this section, results from simulations as well as from real experiments are presented. All the simulations had similar general conditions:

- screen size: 600×400 pixel
- gaussian noise added to the projection of the modelpoints: $\sigma = 2$ pixel
- model points are randomly generated within a cube of $10 \times 10 \times 10$ cm^3
- all the model points are always visible (no hiding polygons)
- the position of the camera relative to the object was always chosen to center the object on the screen and to fill approximately half the screen width
- <u>measured translation error</u>: the **distance** $|T_{fit} - T_{orig}|$ is used as error
- <u>measured rotation error</u>: computation of the angle and the axis of the rotation $R_{fit}^{-1} R_{orig}$ (see [Kanat93]), the **angle** is used as error

Nonlinear optimization: In order to determine the number of iterations needed by the nonlinear optimization and how many points correspondences at least should be used to get a small error, some simulations with synthetic data were made:

The behavior of the error of both, translation and rotation was observed as a function of the number of iterations used for the nonlinear optimization. The number of model points was 8. The initial pose had a constant error, for the rotation 15° on each axis, and for the translation about 2 cm on each axis. The results, averaged over 50 trials,

[4]Visual Pacific PC-605 with 3.0mm lens

showed that only about four iterations are needed to converge to a stable result. Furthermore The results pointed out that even though the theoretical minimum number of points is three (for a four-fold solution) or four (for a unique solution), reliable solutions under noise conditions are only obtained with five points or more.

In order to study the fit using more images, especially with respect to an error of the external parameters of the camera one more simulation was done:

This simulation performed a pose estimation using different numbers of cameras distributed on a segment of 45° using a model consisting of 5 points. Gaussian noise was added to the external parameters of the camera: $\sigma = 0, 0.25, 0.5, 1$ for the rotation (in degrees) and the translation (in cm) on each axis. The rotation error, averaged over 200 trials is displayed in figure 5. The

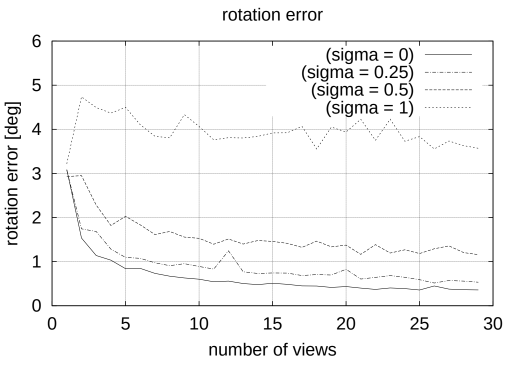


Figure 5: Effect of different errors of the external camera parameters using multiple views

translation error had a similar shape. One can see that when the errors in the external camera parameters are too large using more cameras does not increase the accuracy of the estimation. For smaller errors in the external parameters the the accuracy improves with an increasing number of views.

RANSAC: To test the implementation of the RANSAC-algorithm and to confirm the results of [Fisch81] the following simulation was carried out: A comparison of the pure nonlinear fit with the nonlinear fit in combination with the RANSAC algorithm for varying amounts of outliers.

- the model used in this simulation had 12 points
- the number of outliers varied from 0% to 90%
- the outliers were chosen equally distributed within a region of 70 × 70 pixels

As one can see in figure 6, the more outliers there are, the more iterations are needed. If enough iterations are performed, the fit supported by the RANSAC algorithm is much better than the fit without the detection of outliers. If too few iterations are performed or if there are too many outliers, the fit with the RANSAC-algorithm is worse, because then just a small subset of very bad points are used (see [Fisch81]) to estimate the pose.

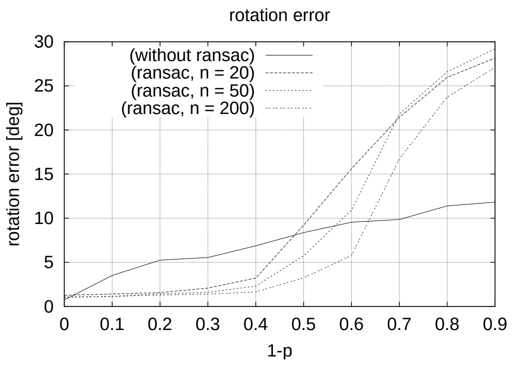


Figure 6: Pose estimation using the RANSAC-algorithm with different numbers of iterations n vs. pose estimation without any detection of outliers

In figure 7 an example of a scene calibration using one image is shown, with and without outlier rejection by the RANSAC algorithm. The scene is quite typical of our setups, with many corner features generated by objects in the background. The many mismatched features (18 of a total of 24 points, corresponding to 75% proportion of outliers) present a formidable fitting problem. It is clearly seen that a simple fitting method generates inacceptable results. In combination with the RANSAC algorithm, however, a visually correct solution is generated based on the remaining 6 correct model points.

6 CONCLUSION

We have presented a semi-automatic system for vision-based scene calibration that can be used to precalibrate a stationary but reconfigurable AR setup. By asking the user to interactively specify approximate object poses by simple visual alignment, the system can robustly compute the object's exact pose automatically using computer graphics and computer vision techniques. In fact, the system uses the known object model and the initial pose estimate provided by the user to threefold advantage : (1) to increase the robustness of feature matching by synthesizing ap-

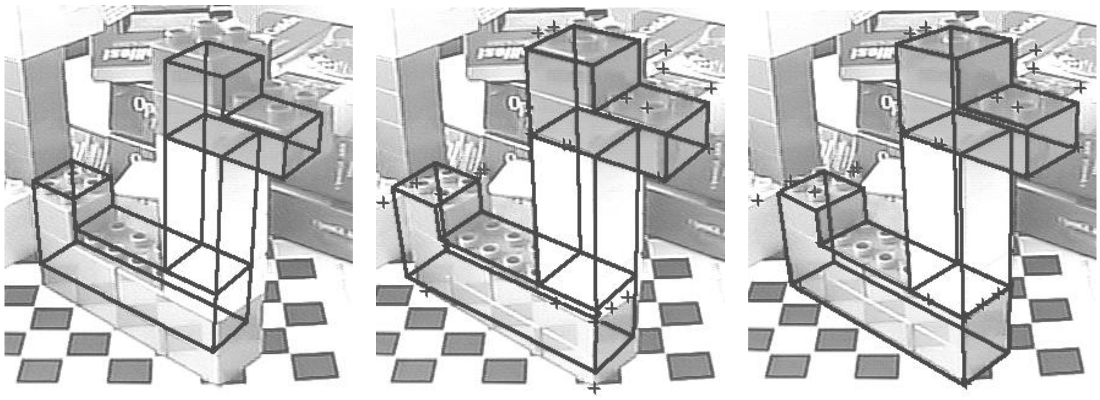
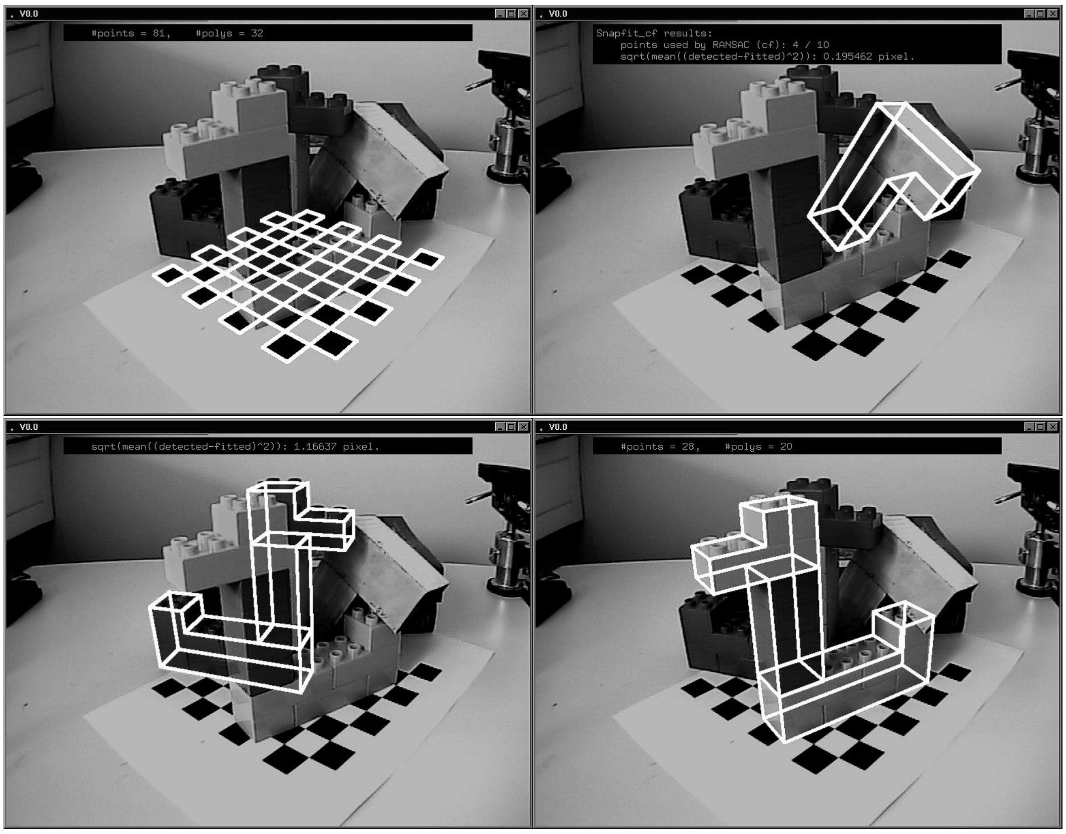


Figure 7: Comparison of the pose estimation with and without outlier detection, left: user given initial pose, center: pose estimation without outlier detection (all 24 detected point are used), right: pose estimation using the RANSAC algorithm to prevent outliers from being used (only 6 points are used)


```
#points = 81,    #polys = 32
```

```
Snapfit_cf results:
    points used by RANSAC (cf): 4 / 10
    sqrt(mean((detected-fitted)^2)): 0.195462 pixel.
```

```
sqrt(mean((detected-fitted)^2)): 1.16637 pixel.
```

```
#points = 28,    #polys = 20
```

Figure 8: Scene calibration for heavily cluttered scene consisting of two lego ducks, an L-shaped object and a chessboard pattern. Individual snapshots show overlaid poses in wire-frame for each object.

propriate view-dependent templates, (2) to limit the search space for each feature to a small window around its predicted location, and (3) to initialize the nonlinear optimization procedure for determining object pose from matched image features with a good starting value.

Our experiments showed that only about four iterations are necessary for the nonlinear pose estimation method to converge to the correct solution from the user-specified initial value. Furthermore, the RANSAC algorithm effectively deals with outliers, producing very good results even in cluttered scenes. Figure 8 shows a scene calibration produced using our system for a collection of 4 objects with significant mutual occlusion.

We currently use only simple polyhedral models. However, as mentioned in the introduction, the templated-based method can be easily extended to non-polyhedral models. For example, appearance-based models containing (natural) textures or more complicated geometric models (e.g. NURBS) can be incorporated into our system if they also allow the synthesis of visual templates of the model features. We are currently investigating more efficient and realistic feature models.

Although there is still much room for improvement concerning the subcomponents of our system, the overall system concept is very promising. By striking a good balance between user interaction and automatic processing, the system offers both ease-of-use and good accuracy of the results, achieving our two main requirements.

REFERENCES

[Armst95] M. Armstrong and A. Zisserman. Robust object tracking. In *Asian Conf. on Computer Vision*, volume I, pages 58–61, 1995.

[Drumm00] T. Drummond and R. Cipolla. Real-time tracking of complex structures for visual servoing. In *Proc. of the Vision Algorithms : Theory and Practice workshop, LNCS 1883*, pages 91–98, Corfu, Greece, 2000. Springer Verlag.

[Fisch81] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[Foley96] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics Principles and Practice.* Addison-Wesley Publishing Company, Reading, Massachusettes, 2nd edition, 1996.

[Haral89] R. Haralick, H. Joo, C. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1426–1446, 1989.

[Haral91] R. Haralick, C. Lee, K. Ottenberg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. *Proceedings of the Conference on Computer Vision and Pattern Recognition, Maui, Hawaii, USA*, pages 592–598, 1991.

[Harri92] C.J. Harris. Tracking with rigid models. In A. Blake and A. Yuille, editors, *Active Vision*, chapter 4, pages 59–73. MIT Press, Cambridge, MA, 1992.

[Hartl00] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, Cambridge, United Kingdom, 2000.

[Horau89] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics, and Image Processing*, 47:33–44, 1989.

[Kanat93] Kenichi Kanatani. *Geometric Computation for Machine Vision.* Clarendon Press, Oxford, 1993.

[Kato99] H. Kato and M. Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. *Proc. of 2nd Int. Workshop on Augmented Reality*, pages 85–94, 1999.

[Klink99] Gudrun Klinker, Didier Stricker, and Dirk Reiners. *Augmented Reality: A Balancing Act Between High Quality and Real-Time Constraints*, chapter 18, pages 325–346. In Ohta and Tamura [Ohta99], 1999.

[Madse97] C. Madsen. A comparative study of the robustness of two pose estimation techniques. *Machine Vision and Applications*, 9(5-6):291–303, 1997.

[Ohta99] Yuichi Ohta and Hideyuki Tamura, editors. *Mixed Reality.* Ohmsha, Ltd. & Springer Verlag, Berlin, 1999.

[Tamur99] Hideyuki Tamura, Hiroyuki Yamamoto, and Akihiro Katayama. *Steps Toward Seamless Mixed Reality*, chapter 4, pages 59–79. In Ohta and Tamura [Ohta99], 1999.

[Trucc98] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3D Computer Vision.* Prentice Hall, Upper Saddle River, New Jersey 07458, 10th edition, 1998.

[Tsai87] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, 1987.