Jochen Schmidt, Ingo Scholz, and Heinrich Niemann

Placing Arbitrary Objects in a Real Scene Using a Color Cube for Pose Estimation

appeared in:
Pattern Recognition, 23rd DAGM Symposium, Lecture Notes in Computer Science
2191
München, Germany
p. 421–428
2001

Placing Arbitrary Objects in a Real Scene Using a Color Cube for Pose Estimation

Jochen Schmidt, Ingo Scholz*, and Heinrich Niemann

Lehrstuhl für Mustererkennung, Universität Erlangen-Nürnberg
Martensstr. 3, 91058 Erlangen, Germany
jschmidt@informatik.uni-erlangen.de,
http://www5.informatik.uni-erlangen.de

Abstract We describe an Augmented Reality system using the corners of a color cube for camera calibration. In the augmented image the cube is replaced by a computer generated virtual object. The cube is localized in an image by the CSC color segmentation algorithm. The camera projection matrix is estimated with a linear method that is followed by a nonlinear refinement step. Because of possible missclassifications of the segmented color regions and the minimum number of point correspondences used for calibration, the estimated pose of the cube may be very erroneous for some frames; therefore we perform outlier detection and treatment for rendering the virtual object in an acceptable manner.
Keywords: Augmented Reality, camera calibration, color segmentation

1 Introduction

In this paper a system is introduced which is an application of Augmented Reality, a visual enhancement of real environments. Unlike many other applications in this field the system described here uses no independent tracking device, but follows a so-called vision-based approach, i. e. calibration information is derived solely from camera input. Many common vision-based systems [10] use some kind of calibration pattern or fiducials still visible in the scene even after augmentation, e. g. [8, 16]. Other methods require manually selected control points [2], but do not use fiducials. In our approach, a metal cube with a side length of 6 cm is used that is painted with a different color on each side such that its position and orientation can be determined unambiguously. A real scene can be augmented by rendering a virtual object in the same pose, thus replacing the cube. A possible application of this system is the usage of a head-mounted display for the visualization of three-dimensional objects that do not yet exist or only at distant locations. The system works without human intervention, and no explicit calibration step is required before using it. The main advantages of our approach comprise the usability in indoor and outdoor applications, easy interaction with the user, and the possibility of illumination estimation by exploiting the knowledge about the colors used on the cube.

* This work was partially funded by the German Science Foundation (DFG) under grant SFB 603/TP C2. Only the authors are responsible for the content.

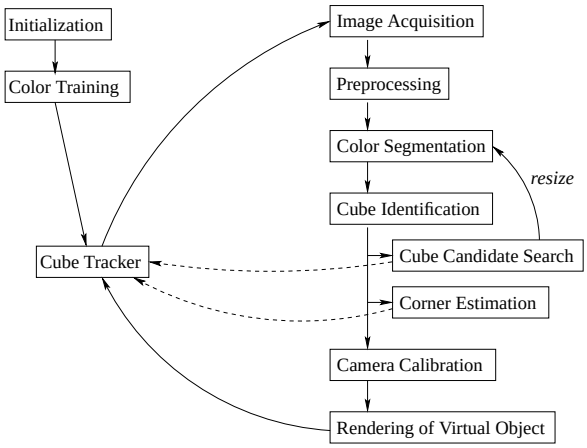

Figure 1. Overview of system structure with special focus on detection of cube corners.

2 System Overview

The first – and most time-consuming – task of the system is to detect the cube in the image and to determine its corners, thus establishing a correspondence between 2-D image and 3-D world points. In order to get a correct calibration of the camera, the identification of the corners must be as accurate as possible. Details on the method described here can be found in [15].

The system is basically structured as shown in Fig. 1. After an initialization phase and the training of the color classificator, the main loop of the cube tracker is started. This is the manager of the whole system: it contains general information, like the image size and previously known positions of the cube, and it executes the subsequent phases.

In each loop the tracker reads the current image and preprocesses it as described in Sect. 3. Color segmentation is done in two steps: first a version of the image that was reduced in size is segmented and checked for cube canditates. If one or more candidates are found, the corresponding image regions are segmented once more, this time in normal resolution. The color segmentation yields a number of uniformely colored regions. Cube candidates are identified out of these regions by applying a number of restrictions on sets of three or two regions. That is, their colors have to correspond to those of the cube and they must fulfill some geometric properties. If none of the region sets fits these qualities, the tracker discards this image and proceeds with the next one. After that, a set of one or more cube candidates remains, each of which is given a score depending on how well it fits the conditions above. The corner estimator then tries to find the corners of the cubes with the highest scores. If this step fails, the candidate is rejected and the next one is tried. If no candidate remains, the tracker continues with the next image as well.

With the corner points identified, it is now possible to do camera calibration. Details on that part are given in Sect. 4. The final step in the cycle is to render the virtual object that is to replace the cube in the image. From this point on the sequence starts again

with the acquisition of the next image frame. If a cube was recognized, its position is taken as a cue for where to start the search in the next image.

3 Finding the Cube

For locating the corners of the cube, two of its properties are used: its color and its geometry. In order to distinguish the cube from the background it was painted in luscious, matte colors. Each side received a unique color such that the system can identify the cube's correct orientation. Of course, in a natural scene it is very likely that other objects are colored similarly. Therefore the geometric appearance of the objects in consideration has to fit that of a cube as well. The input of the cube detector is a stream of RGB images. An enhancement of quality is reached by applying a Symmetric Nearest Neighbour (SNN) filter as suggested in [12] which reduces noise like a mean filter, but preserves edges. Each side of the cube is a region of almost uniform color, except for shadows cast onto it. By choosing matte colors the problem of highlights reflected from a light source was mostly eliminated. Therefore the first step in identifying the cube is a color segmentation of the image. The algorithm used for this is the Color Structure Code (CSC) as described in [12], which is based on an earlier method [5].

It is assumed that each visible side of the cube can be segmented as exactly one region, given that a coarse enough parameterization for the CSC is used. A division of a side happens only in extreme cases, e. g. if a shadow is partially cast over it. Thus each region in the image is classified for being a possible cube side or not, according to its mean color. The numerical classifier applied here uses two feature vectors, the RGB values of a region's color and its hue and saturation values, taken from HSV color space. The distance between a region's mean color and a cube side color class is calculated by the Mahalanobis distance measure. As the color segmentation is costly, the image is first shrunk to a fifth of its size and possible cubes are localized herein. The corresponding windows are cut out of the full-size image, where the cube detection is performed again. After color classification a set of regions remains containing all the regions that possibly are part of the cube. From these the ones truly belonging to the cube have to be filtered out. The subsequent processing of the image requires that at least two sides of the cube are visible – a criterion which reduces the number of cube side candidates, because the knowledge of adjacent colors on the cube can be exploited. In addition, the regions must have the correct appearance: their edges have to be straight lines, forming a parallelogram. Each of the remaining regions gets a score measuring the regions' affinity to the cube. Corners are determined for the regions with the highest scores. This is done by approximating the edges of each side with straight lines using the Hough transform. The intersections of these lines are taken as the corners of the side. In cases where lines are missing due to occlusion, the corners are approximated using information from neighboring sides or the original image in Cartesian coordinates.

The algorithms for camera calibration applied in the following sections need at least six or seven point correspondences, depending on the method. With one side of the cube visible only four points can be obtained, so that two or three sides have to be visible, providing six or seven point pairs respectively. In the case where only one side is visible, the method described in Sect. 4.4 can be applied.

4 Camera Calibration

Camera calibration is done in the following steps, which are described in detail afterwards: Computation of additional point correspondences (Sect. 4.1), linear calibration of all camera parameters (Sect. 4.2), maximum-likelihood estimation of the focal lengths (Sect. 4.2), nonlinear refinement of extrinsic camera parameters (Sect. 4.3), and test of the validity of the computed parameters (Sect. 4.4).

4.1 Computing Additional Point Correspondences

Either six or seven 2-D/3-D point correspondences are established by the algorithm described in the previous section. While six points are enough for one of the methods described below, at least seven are needed for applying the algorithm of Tsai.

Since we use an object of known shape we can compute one additional point correspondence for each side where all four corners have been detected by using the intersection of the two diagonals of a cube side. These additional points are easy to compute in the image and in 3-D and are valid correspondences because the perspective projection preserves intersections. Using projective geometry (see [4] for an introduction) the intersection $\boldsymbol{q}_\mathrm{S}$ of the diagonals in the image plane can be computed by

$$\boldsymbol{q}_\mathrm{S} = (\boldsymbol{q}_1 \times \boldsymbol{q}_3) \times (\boldsymbol{q}_2 \times \boldsymbol{q}_4) \quad , \tag{1}$$

where $\boldsymbol{q}_\mathrm{S}, \boldsymbol{q}_i$ $(i = 1, \ldots, 4)$ are 3-vectors representing image points in homogeneous coordinates, $\boldsymbol{q}_1$ is opposite to $\boldsymbol{q}_3$, $\boldsymbol{q}_2$ is opposite to $\boldsymbol{q}_4$.

4.2 Linear Calibration

For calibration we assume a perspective camera model. A homogeneous 3-D point $\boldsymbol{w}_i$ is projected onto a homogeneous 2-D point $\boldsymbol{q}_i$ in the image plane using the following equation:

$$\boldsymbol{q}_i = \boldsymbol{P}\boldsymbol{w}_i = \boldsymbol{K}\boldsymbol{R}^\mathrm{T}(\boldsymbol{I}_3 | -\boldsymbol{t})\boldsymbol{w}_i \quad , \tag{2}$$

where $\boldsymbol{K}$ is a 3×3 matrix containing the intrinsic parameters f_x, f_y, u_0, and v_0, $\boldsymbol{R}$ is a rotation matrix whose columns correspond to the axes of the camera coordinate system, $\boldsymbol{t}$ is a translation vector giving the position of the camera's optical center, and $\boldsymbol{I}_3$ is the 3×3 identity matrix.

After the previous steps there are enough point correspondences to apply a linear calibration method. For this purpose we use either the algorithm of Tsai that can be found in [18, 17], or the algorithm described in [17] for estimating the projection matrix. Radial distortions are neglected, the angle between the axes of the image reference frame is assumed to be $90°$. Tsai's method assumes that the principal point is known and the camera parameters are computed directly, in contrast to the second method used which estimates the projection matrix first and makes no assumptions about the principal point. Both methods require non-coplanar point correspondences and the orthogonalization of the resulting rotation matrix which can be done by applying a singular value decomposition (SVD) [11].

Since f_x, f_y are assumed to be constant over the whole sequence, we can get maximum-likelihood estimates $\hat{f}_\mathrm{x}$, $\hat{f}_\mathrm{y}$ of these parameters at frame t ($t = 1, 2, \ldots$) under the assumption of normally distributed, isotropic, and zero-mean noise by the following recursive equation (given here for $\hat{f}_\mathrm{x}$ only):

$$\hat{f}_{\mathrm{x},t} = \frac{1}{t+1}\left(t\hat{f}_{\mathrm{x},t-1} + \tilde{f}_{\mathrm{x},t}\right) = \hat{f}_{\mathrm{x},t-1} + \frac{1}{t+1}\left(\tilde{f}_{\mathrm{x},t} - \hat{f}_{\mathrm{x},t-1}\right) \quad , \tag{3}$$

where $\tilde{f}_{\mathrm{x},t}$ is the result of the linear calibration at time step t. For $t = 0$ the initialization $\hat{f}_{\mathrm{x},0} = \tilde{f}_{\mathrm{x},0}$ is used.

4.3 Nonlinear Refinement

Since we use only slightly more points than the minimum number required for calibration, nonlinear refinement of camera parameters with the linear estimation as initialization is absolutely essential. Optimization is done here for the extrinsic parameters only, while the intrinsic parameters from the previous maximum-likelihood estimation are used and held constant during nonlinear refinement. For this purpose the Gauss-Newton algorithm with Levenberg-Marquardt extension (see [3] for details) is utilized which computes a new estimate of a parameter vector $\boldsymbol{a}$ using a local parametrization $\Delta\boldsymbol{a}$ by $\hat{\boldsymbol{a}}_{k+1} = \hat{\boldsymbol{a}}_k + \Delta\boldsymbol{a}$ where

$$\Delta\boldsymbol{a} = -\left(\lambda\boldsymbol{I} + \boldsymbol{J}^\mathrm{T}\boldsymbol{J}\right)^{-1}\boldsymbol{J}^\mathrm{T}\,\boldsymbol{\epsilon}(\hat{\boldsymbol{a}}_k) \quad . \tag{4}$$

This method minimizes the mean square error $\boldsymbol{\epsilon}^\mathrm{T}\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a residual function that computes in our case the (non-squared) back-projection error between each image point (x_i, y_i) and the projection $\boldsymbol{q}_i = (q_{i\mathrm{x}}, q_{i\mathrm{y}}, q_{i\mathrm{w}})^\mathrm{T}$ of its corresponding 3-D point $\boldsymbol{w}_i$ obtained by equation (2):

$$\boldsymbol{\epsilon} = \left(x_1 - \frac{q_{1\mathrm{x}}}{q_{1\mathrm{w}}}, y_1 - \frac{q_{1\mathrm{y}}}{q_{1\mathrm{w}}}, \ldots, x_n - \frac{q_{n\mathrm{x}}}{q_{n\mathrm{w}}}, y_n - \frac{q_{n\mathrm{y}}}{q_{n\mathrm{w}}}\right)^\mathrm{T} \quad . \tag{5}$$

$\boldsymbol{J}$ is the Jacobian of the first derivatives of $\boldsymbol{\epsilon}$ evaluated at $\hat{\boldsymbol{a}}_k$: $\boldsymbol{J} = \frac{\partial\boldsymbol{\epsilon}}{\partial\boldsymbol{a}}(\hat{\boldsymbol{a}}_k)$. Since the matrix inversion in equation (4) may be numerically instable due to a nearly singular matrix $\boldsymbol{J}^\mathrm{T}\boldsymbol{J}$, the factor λ is introduced in the Levenberg-Marquardt algorithm and adapted during each iteration. One Levenberg-Marquardt iteration comprises the following actions: Computation of a parameter update using equation (4) as well as the resulting back-projection error, acceptance of the new parameters if the error is smaller than the error after the last iteration and division of λ by a factor of 10, or rejection of the computed parameters and muliplication of λ by a factor of 10. Since the error may increase during one iteration due to instabilities in matrix inversion, the preceding steps are done until the new parameters yield a smaller error than at the end of the last iteration. The parameter vector $\boldsymbol{a}$ contains the 3 components of the translation $\boldsymbol{t}$ plus 3 components parametrizing the rotation matrix $\boldsymbol{R}$, which has 9 elements but only 3 degrees of freedom (DOF). A numerically stable parametrization should be used, i. e. either the axis/angle representation or quaternions which are both a fair parametrization

of rotations in the sense of [6], while Euler angles are not. A rotation matrix can be represented by a 3-vector $\boldsymbol{r} = (r_1, r_2, r_3)^{\mathrm{T}}$ giving the direction of the rotation axis with 2 DOF plus the rotation angle θ encoded as the norm of $\boldsymbol{r}$. The corresponding rotation matrix $\boldsymbol{R}$ can be calculated by Rodrigues' formula [4]:

$$\boldsymbol{R} = \boldsymbol{I}_3 + \frac{\sin\theta}{\theta} \begin{pmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{pmatrix} + \frac{1-\cos\theta}{\theta^2} \begin{pmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{pmatrix}^2 \quad . \tag{6}$$

When using quaternions for nonlinear optimization it is necessary to consider that a quaternion representing a rotation has 4 elements but only 3 DOF, since it must be normalized to 1. The Levenberg-Marquardt algorithm cannot deal with constraints on the parameters and it must be guaranteed that the norm of a quaternion is always 1 during optimization. In order to accomplish this goal a quaternion parametrization at the operating point using only 3 elements was introduced in [13, 14].

4.4 Detection and Treatment of Outliers

For different reasons the virtual object in the resulting augmented image may be rendered in a completely different pose than the cube. Most of the time this is not due to calibration errors or badly localized cube corners, but to missclassification of the colors painted on the sides of the cube. Additionally there are images where no cube could be found at all. Both cases lead to visually inacceptable results and hence have to be dealt with in an appropriate way. For detection of a non-valid pose, we use thresholds for change in rotation (measured in distance between vectors in axis/angle representation) and translation with respect to the last valid frame.

The easiest solution is to keep the pose of the last valid frame. This is acceptable when only a few frames have to be dropped. Otherwise a prediction of the cube's movement would yield a better result for the human observer. For this purpose we use linear prediction, a technique classically applied in speech recognition [9]. Linear prediction estimates the n-th value of a sequence of discrete values g_j using a combination of the preceding k values as follows:

$$\hat{g}_n = -\sum_{i=1}^{k} \alpha_i g_{n-i} \quad . \tag{7}$$

Having a long enough sequence, one can estimate the unknowns α_i from a linear system of equations using e. g. the SVD which minimizes the mean square error. The α_i are in turn used for predicting new values of the sequence. In our case we predict the elements of the translation vector and of the rotation matrix in axis/angle representation.

5 Experiments

In our experiments we used images of size 360×288 pixels. Two results of augmentation are shown in Fig. 2. More images and video-sequences are available at [1].

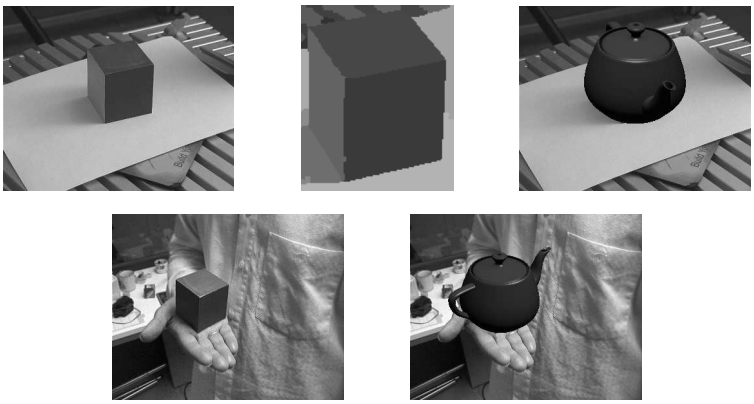

Figure 2. Two examples of augmentation: Original (top left), CSC segmented cut-out (top middle), and augmented image (top right) on a turntable; original (bottom left) and augmented image (bottom right) of cube held in hand.

We found that the part most prone to errors is the color classification due to different lighting conditions. Missclassification of a cube side leads to wrong 2-D/3-D assignment and thus to unusable calibration results. The color classifier was trained using 577 to 885 samples per color. The recognition rate for the whole cube ranges from 74% - 93%, depending on the illumination of the scene. The mean detection accuracy (measured by hand) of the cube's corners is about 1.3 pixels. If Tsai's calibration method is to be used, additional point correspondences must be computed when only six points have been detected which leads to fairly good visual results. When using the other method, simulations showed that calibration can be done accurately enough even with only six point correspondences, while computation of additional points yields a worse back-projection error and worse camera parameters.

The system is still working off-line, but since we have a real-time system in mind we want to give an impression of the computation times needed for one frame on a 800 MHz Pentium III. Using the Windows program of [12], the CSC took 40 to 130 msec for two passes, one on the reduced size image and one on the cube cut-out, the given time depending on the number of possible cube-regions found in the reduced-size image. Additional 150 to 250 msec are needed for color classification of the segmented regions and computing the corners. The time needed for calibration is dominated by nonlinear optimization; depending on the number of Levenberg-Marquardt iterations done, computation time varied from 10 msec (5 iterations) to about 30 msec (20 iterations). OpenGL rendering takes additional 30 msec.

6 Conclusion and Future Work

In this contribution we present an Augmented Reality system using a color segmentation approach for localizing a metal cube in an image which can be replaced by an arbitrary computer-generated object. Camera calibration with a minimum number of

point correspondences works well and is fast enough for real-time applications even when nonlinear refinement is done. Topics for further improvement are the speed and accuracy of cube localization and corner detection. We also want to consider illumination estimation using information on the cube's colors in the future.

References

1. `http://www5.informatik.uni-erlangen.de/~ar`.
2. C.-S. Chen, C.-K. Yu, and Y.-P. Hung. New calibration-free approach for augmented reality based on parameterized cuboid structure. In ICCV 99 [7], pages 30–37.
3. J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, 1983.
4. Oliver Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, MA, 1993.
5. G. Hartmann. Recognition of hierarchically encoded images by technical and biological systems. *Biological Cybernetics*, 57:73–84, 1987.
6. J. Hornegger and C. Tomasi. Representation issues in the ML estimation of camera motion. In ICCV 99 [7], pages 640–647.
7. *Proceedings of the 7^{th} International Conference on Computer Vision (ICCV)*, Corfu, September 1999. IEEE Computer Society Press.
8. D. Koller, G. Klinker, E. Rose, D. Breen, R. Whitaker, and M. Tuceryan. Automated camera calibration and 3D egomotion estimation for augmented reality applications. In *Computer Analysis of Images and Patterns (CAIP)*, pages 199–206, Kiel, September 1997. Springer.
9. J. D. Markel and A. H. Gray Jr. *Linear Prediction of Speech*, volume 12 of *Communications and Cybernetics*. Springer Verlag, Berlin, Heidelberg, New York, 1976.
10. Y. Ohta and H. Tamura, editors. *Mixed Reality – Merging Real and Virtual Worlds*. Springer-Verlag, Berlin, 1999.
11. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 2nd edition, 1992.
12. V. Rehrmann and L. Priese. Fast and robust segmentation of natural color scenes. In *Proceedings of the 3^{rd} Asian Conference on Computer Vision*, volume 1, pages 598–606, HongKong, January 1998.
13. J. Schmidt. Erarbeitung geeigneter Optimierungskriterien zur Berechnung von Kameraparametern und Szenengeometrie aus Bildfolgen. Diplomarbeit, Lehrstuhl für Mustererkennung, Universität Erlangen-Nürnberg, 2000.
14. J. Schmidt and H. Niemann. Using quaternions for parametrizing 3–D rotations in unconstrained nonlinear optimization. In T. Ertl, B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2001*, Stuttgart, Germany, November 2001. Submitted.
15. I. Scholz. Augmented Reality: A System for the Visualization of Virtual Objects Using a Head-mounted Display by Localization of a Real Object of Known Geometry and Color. Diplomarbeit, Lehrstuhl für Mustererkennung, Universität Erlangen-Nürnberg, 2000.
16. Y. Seo and K. Sang Hong. Calibration-free augmented reality in perspective. *IEEE Transactions on Visualization and Computer Graphics*, 6(4):346–359, 2000.
17. E. Trucco and A. Verri. *Introductory Techniques for 3–D Computer Vision*. Prentice Hall, New York, 1998.
18. 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*, Ra-3(3):323–344, August 1987.