Texture driven pose estimation

Bodo Rosenhahn, Harvey Ho and Reinhard Klette
Center for Imaging Technology and Robotics (CITR)
The University of Auckland
New Zealand
bros028@cs.auckland.ac.nz

Abstract

This article presents a 2D-3D pose estimation algorithm which relies on texture information on the surface mesh of an object model. The textured surface mesh is rendered in a virtual image and a modified block matching algorithm is applied to determine correspondences between midpoints of surface patches to points in an image. This is used in a point-based 2D-3D pose estimation algorithm to determine the pose and orientation of a 3D object with respect to given image data. We present experiments on various image sequences and show advantages of the chosen approach (e.g., in the context of varying backgrounds, noise or partial occlusions).

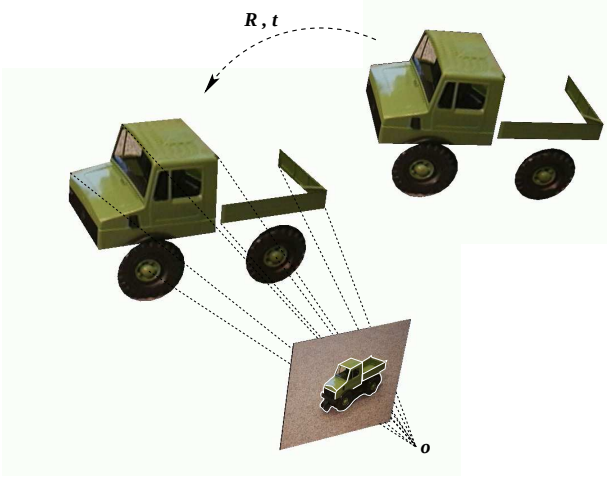

Figure 1. Basic sketch of the 2D-3D pose estimation problem: The task is to find a rigid body motion, defined by a rotation R and translation t, which leads to a best fit between observed 2D image data and a predefined 3D object model.

1 Introduction

Pose estimation has been studied in computer vision since its beginnings. It is crucial for many computer and robot vision tasks. This article addresses the *2D-3D pose estimation problem* [2] which is defined as follows: We assume an image of an object taken by a calibrated camera as 2D sensory data, and we assume a 3D representation of an object model. With 2D-3D pose estimation we calculate a rigid motion (i.e., containing both 3D rotation and 3D translation) which fits a particular object model with the image data. The basic scenario is visualized in Figure 1. A crucial question for pose estimation is the form of object representation. The literature deals with point and line based representations, kinematic chains, higher order curves or surfaces, up to free-form contours or free-form surfaces; see [1, 9] for overviews. Previous works [10] followed a silhouette-based approach for pose estimation of free-form surface models. Figure 2 shows examples of images. There, just the rim contour in an image is used as feature for pose estimation, and all internal structure information is ignored. In this contribution we want to introduce a pose estimation algorithm which relies on texture information of rendered objects. The reason is that using texture information eliminates image pre-processing (e.g., corner extraction or silhouette estimation), since no preprocessing will be needed at all. Instead we will work on the image itself. Furthermore, it allows for the use of as much image information as there is visible, in contrast to silhouette based pose estimation, which relies on the object rim. The price to be paid is a more complex object representation,

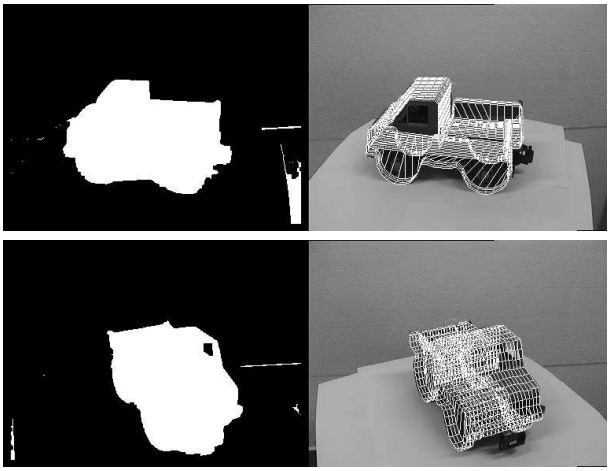

Figure 2. Examples of images taken for a silhouette-based approach for pose estimation.

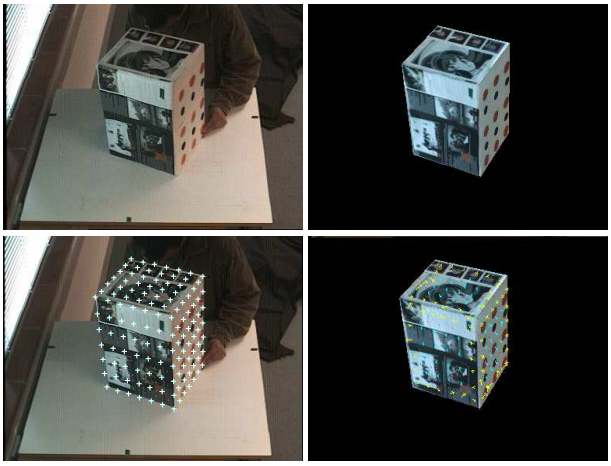

Figure 3. Illustration of texture-based pose estimation.

which not only requires a free-form surface mesh, but also textures and texture coordinates on the mesh. Figure 3 visualizes the basic idea: we assume the representation of a 3D object model as surface mesh and assume a texture mapped onto the object (upper right). We further observe the represented object in an image of a calibrated camera (upper left). Now the idea is to render the object with the projection matrix into a virtual image and to perform a 2D block matching between the virtually projected object and the observed object (lower right). This leads to a set of 2D-2D correspondences by using the midpoints of surface patches. Since we can further determine the midpoint of a 3D surface patch to its projected 2D surface patch, this leads to a set of 2D-3D point correspondences. We use this set of correspondences in a point-based pose estimation algorithm to determine the pose of the object model with respect to the image data. We call this procedure *2D-3D texture-based pose estimation*. A pose result (shown in the lower left image of Figure 3) is stated by overlaying the corners of the 3D surface mesh with the input image. The basic principle can be compared with a least-square correlation technique, widely used for image matching by estimating their closest similarity [3, 4]. Different 2D shapes such as squares or ellipses are common; instead here we work with perspectively projected surface patches as adaptive 2D shapes. Even 3D shape knowledge can be incorporated [5]. But instead of minimizing within a space of image signals by combining the optic flow constraint with a linearized rigid motion, we propose to estimate a local flow-field and apply a feature-based pose estimation algorithm separately. This allows for a faster (real-time) algorithm and local errors (due to noise or occlusion) can be handled more easily.

We continue in Section two with an introduction to block matching and pose estimation. In Section three we introduce the 2D-3D block matching based pose estimation algorithm and explain the algorithmic steps in detail. Section four continues with first experiments of this approach and Section five concludes with a brief discussion.

2 Foundations

2.1 Block matching

Block matching (BM) is employed to measure similarities between two images, or portions of images, on a pixel-by-pixel basis [12]. It is widely used in visual tracking, stereo vision and video compression applications. To compare two blocks (a block is a subarea of an image), two common criteria are the sum of square difference (SSD) and the sum of absolute difference (SAD). Although adopted for various computer vision tasks, block matching in 2D space has some weaknesses (e.g., in its inaccuracy in 3D rotation estimation): the block motion vector is computed relatively precisely for a translational movement, but if an object rotates in 3D, its blocks should be deformed with a perspective transformation to match the scene. But no 3D information is entirely embedded in 2D-2D block matching, since only rectangularly shaped blocks are commonly used. This leads to inaccuracies and therefore unsatisfactory matching results. One aspect of this contribution is to couple 3D model information with block matching to handle scenes with rotations, as well as partially or fully occluded surface patches.

Searching for a block inside of a search window follows some search patterns, which can be classified into two

categories, Full Search and Step Search (e.g., Three-Step Search, Four-Step Search or New Three Step Search). We implemented Full Search, Three-Step search and Four-Step Search. As the block matching accuracy is the primary requirement, we use Full Search as the main approach. It still leads to a fast algorithm, and we can process the images in 5-15[1] frames per second on a standard Linux PC. We use block matching to compare texture mapped patches on a surface model with 2D image data.

2.2 Pose estimation

We assume a set of point correspondences (X_i, x_i), with 4D (homogeneous) model points X_i and 3D (homogeneous) image points x_i. Each image point is reconstructed to a Plücker line $L_i = (n_i, m_i)$, with a (unit) direction n_i, and moment m_i [6]. The 3D rigid motion is represented as exponential form

$$M \quad = \quad \exp(\theta\hat{\xi}) = \exp\begin{pmatrix} \hat{\omega} & v \\ 0_{3\times 1} & 0 \end{pmatrix} \qquad (1)$$

where $\theta\hat{\xi}$ is the matrix representation of a twist $\xi = (\omega_1, \omega_2, \omega_3, v_1, v_2, v_3) \in se(3) = \{(v, \omega)|v \in \mathbb{R}^3, \omega \in so(3)\}$. A twist contains six parameters and can be scaled to $\theta\xi$ with a unit vector ω. To reconstruct a group action $M \in SE(3)$ from a given twist, the exponential function $\exp(\theta\hat{\xi}) = M \in SE(3)$ can be used. The parameter $\theta \in \mathbb{R}$ corresponds to the motion velocity (i.e., the rotation velocity and pitch). For varying θ, the motion can be identified as screw motion around an axis in space. This is also proven by Chasles' Theorem [6] from 1830. Indeed, evaluating the exponential of a matrix is non-trivial, but it can be calculated efficiently by using the Rodriguez formula [6],

$$\exp(\hat{\xi}\theta) = \begin{pmatrix} \exp(\theta\hat{\omega}) & (I - \exp(\hat{\omega}\theta))(\omega \times v) + \omega\omega^T v\theta \\ 0_{1\times 3} & 1 \end{pmatrix}$$

with $\exp(\theta\hat{\omega})$ computed by calculating

$$\exp(\theta\hat{\omega}) \quad = \quad I + \hat{\omega}\sin(\theta) + \hat{\omega}^2(1 - \cos(\theta)). \qquad (2)$$

Note that only sine and cosine functions of real numbers need to be computed.

For pose estimation we combine the reconstructed Plücker lines with the screw representation for rigid motions and apply a gradient descent method: Incidence of the transformed 3D point X_i with the 3D ray L_i can be expressed as

$$(\exp(\theta\hat{\xi})X_i)_{3\times 1} \times n_i - m_i \quad = \quad 0. \qquad (3)$$

Indeed, x is a homogeneous 4D vector, and after multiplication with the 4×4 matrix $\exp(\theta\hat{\xi})$ we neglect the homogeneous component (which is 1) to evaluate the cross product with n. Note, that this constraint equation expresses the perpendicular error vector between the Plücker line and the 3D point. The aim is to minimize this spatial error. Therefore we linearize the equation by using $\exp(\theta\hat{\xi}) = \sum_{k=0}^{\infty} \frac{(\theta\hat{\xi})^k}{k!} \approx I + \theta\hat{\xi}$, with I as identity matrix. This results in

$$((I + \theta\hat{\xi})x)_{3\times 1} \times n - m = 0, \qquad (4)$$

and can be reordered into an equation of the form $A\xi = b$. Collecting a set of such equations (each is of rank two) leads to an overdetermined system of equations, which can be solved using, for example, the Householder algorithm. The Rodriguez formula can be applied to reconstruct the group action M from the estimated twist ξ. Then, the 3D points can be transformed and the process is iterated until the gradient descent approach converges. In recent years, this technique has been extended to higher order curves, free-form contours and free-form surfaces, see [9, 10]. There, a silhouette-based method for pose estimation of free-form surface models is proposed.

3 2D-3D Block matching

We assume an object model, given as a surface mesh added with texture information. Furthermore, we assume an image of the visible object and a projection matrix, which relates the surface mesh to the camera. The projection matrix with respect to the image data should give us a tracking assumption, else the search space for block matching will be too large (e.g., in this case up to 10 pixels). Then the algorithm starts with rendering the object model in a virtual image by using the given projection matrix. After this we perform a 2D block matching from the rendered object model to the image data. Note that background is automatically eliminated, since we perform block matching from the virtual image to the given image, and not vice versa. Furthermore, we do not work with rectangular shaped 2D blocks, but with deformed search windows along the surface mesh. This is shown in Figure 4. To model a perspective block, we use a rectangular box with a block-mask

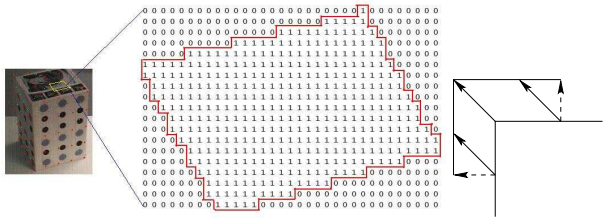

Figure 4. Left: A deformed block mask. Right: Illustration of the well-known aperture problem.

[1] Depending on the mesh resolution.

obtained through the Ray-Crossing algorithm (e.g., introduced in [7]). Therefore we use the bounding box of each search pattern, where height and width are calculated for each patch at each frame. Then the Ray-Crossing algorithm is applied to check whether a point is inside or outside the bounding box. If a point is inside the deformed block, its corresponding flag is set to 1, else to 0. When comparing the deformed block from one frame to the next frame, the mask is tested for each point on the block. If the flag is 1, a comparison is performed, else not. Note that we apply a standard 2D-2D block matching algorithm and only use an additional filter mask to obtain a deformed block. After the block matching, we determine from the 2D-2D correspondences the 2D-3D correspondences. This set of correspondences is used in our point-based pose estimation procedure, and we can use the estimated rigid body motion to update the projection matrix for the next frame.

When the box moves, its six faces can become visible or invisible dynamically. In other words, if a face is present in the first frame, it might be occluded in the second frame. Consequently, the number of correspondences is changing dynamically. Optic flow estimation procedures are often

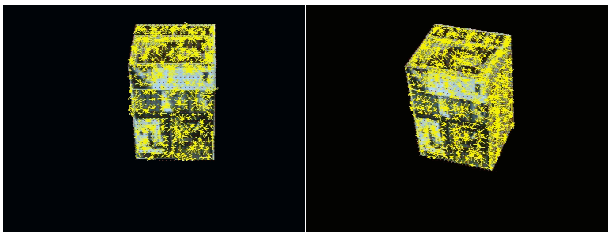

Figure 5. Block matching result from the virtual image to the real image. The aperture problem influences results.

sensitive to the aperture problem (visualized on the right in Figure 4): using only local search windows allows us to estimate the normal flow along a gradient (shown as dashed arrows), instead of the real flow. This also happens here during block matching, see Figure 5. But since all information is used at once for pose estimation, the normal flows contribute to the real object motion. This leads to stable and smooth pose results, although the local flow field is noisy.

A problem during the comparison between the texture mapped object model and the image are the changing lighting conditions which result in a need for color calibration, or on-line texture updating. We use a process of texture updating, which also allows reflections on an object to be handled. To achieve an on-line texture updating, we use the pose result of the last processed image frame and take the last frame for new texture coordinates on the object model. Matching in the image domain is not even stable with a

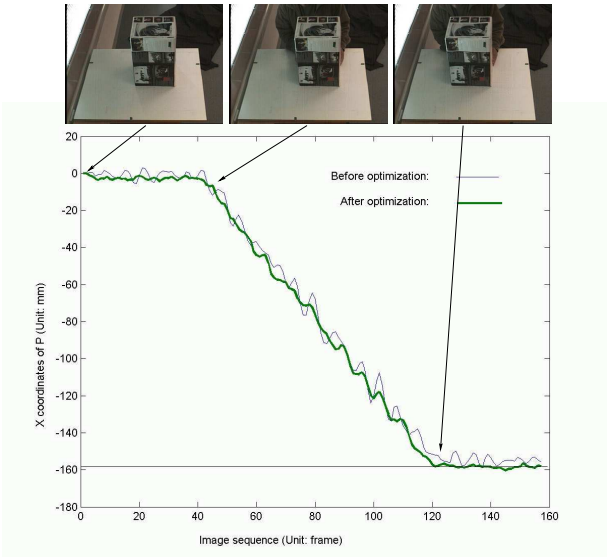

Figure 6. Translation along the x-axis.

static scene, since an error propagation at the object boundaries can occur during mapping the textures next to the object on the object grid. To avoid this problem, we use a hybrid method and perform texture updates just for the inner surface patches, whereas the boundary patches remain unchanged. This hybrid approach proves to be much more stable than using constant textures.

4 Experiments

We start our experiments with a simple scene. As an object model we use a box with six textured faces. The box has the height, width and depth of $405 \times 280 \times 255\, mm$. We calibrate the scene using Tsai-calibration [11]. The camera has a distance of approximately $2m$ to the object and takes images of resolution 384×288 pixels. We implemented in `C,C++` and use OpenGL for rendering and visualization.

A result of a first sequence, dealing with a translational movement of the object, is shown in Figure 6: at the beginning the object is not moving over 40 frames. Then the object is moved along the x-axis of the world coordinate system for $160mm$ (till frame 120), and then it remains constant till frame 160. The images above the diagram show a few examples of the sequence. Though the object is moved manually, the motion pattern is clearly visible in the diagram. Assuming that we perform an ideal motion, the error varies around $2mm$ in space. The two diagrams show results using constant textures (before optimization) and using dynamic texture updating (after optimization). It shows, that the results are stabilized.

Figure 7 shows examples from a rotation sequence. The algorithm can automatically decide which patches are cur-

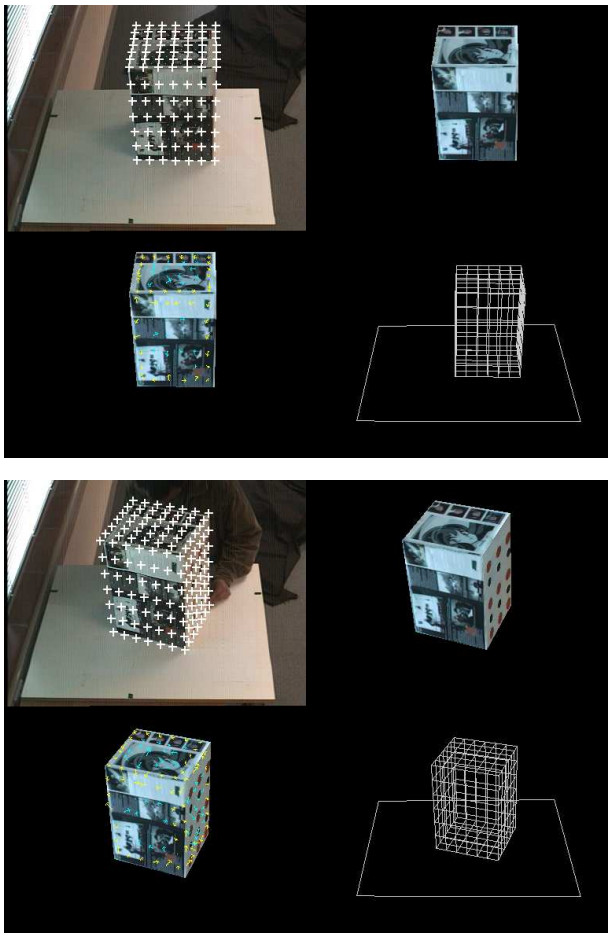

Figure 7. Rotation of the box model: Due to rotation, patches can become visible or occluded, leading to a dynamic number of correspondences.

rently visible or (self-)occluded and determine the number of used correspondences dynamically.

Figure 8 shows examples of changing backgrounds during pose estimation. Since 2D-3D block matching is always performed from the virtual image to the real image (and not vice versa) changing backgrounds do not influence the block matching results. Shadows are only effects on the objects which disturb the matching result. For occlusion handling, those correspondences of surface patches whose matching errors are too big are rejected during pose estimation: Figure 9 shows the stability of our approach with respect to artificially disturbed image data: for a sequence with a non-moving object we add a blue stripe over each frame and move it from the left to the right in the image. We further estimate the pose of the object during the sequence. The images on top show a few examples for a blue

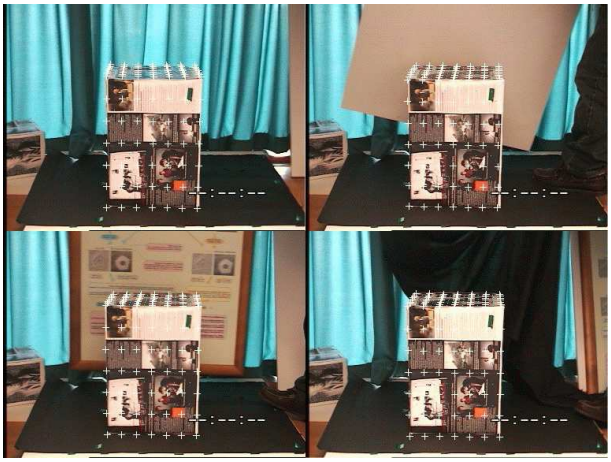

Figure 8. Pose results for changing backgrounds.

stripe leading to 70% occlusion. The diagram shows the frame number at the x-axes and the value of the estimated y-coordinate (in mm) at the y-axes during the sequence. The different curves show the y-coordinates for different stripe widths leading to 0%-70% occlusion of the object. Since the object is not moving in the sequence, the ground truth is a zero function, which is nearly given by the values for no occlusion. The more occlusion occurs, the more noisy are the results, but as can be seen, we are able to track an object successfully even with 70% occlusion. Figure 10 shows pose results from a second occlusion experiment: A person is moving the arms in front of a camera, while the object is moving along the x-axis.

Figure 11 shows the stability of the algorithm with respect to noisy image data. Therefore, we use the first sequence and add Gaussian noise to each pixel. Then the pose is estimated. The diagram shows the translation along the x-axis during the sequence for different noise levels. The image in the upper right corner visualizes the noise level of 0.6 which still allows us to track the object.

Another extension is to turn from a monocular approach to a stereo approach. This requires a block matching for each image separately. Then the sets of equations for both cameras are combined into one system of equations and solved simultaneously. The setup is shown in Figure 12 and a pose result of a cylinder model is shown in Figure 13.

5 Discussion

In this paper we present an approach for pose estimation of object models with textured surfaces. For this we fuse a point-based pose estimation algorithm with a perspective block-matching algorithm, which allows to match

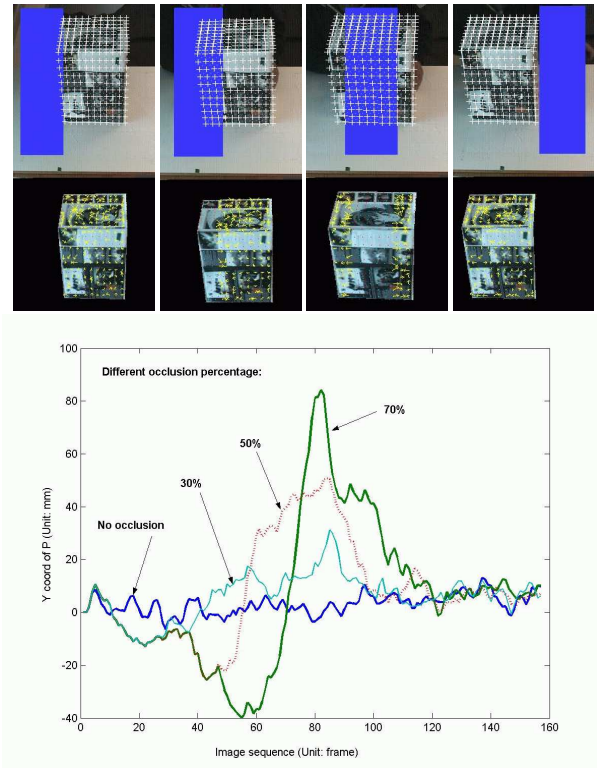

Figure 9. Different occlusions during an image sequence containing a static object.

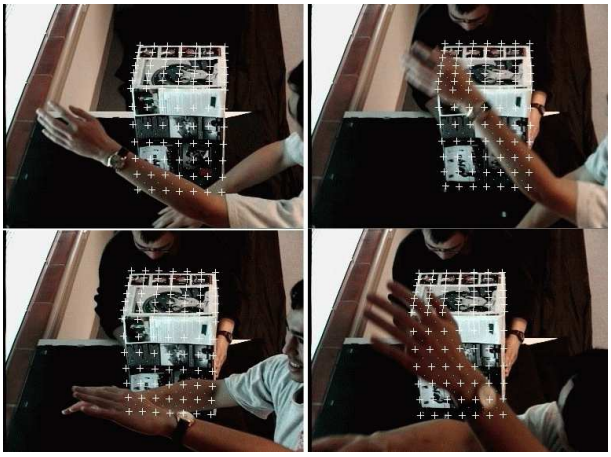

Figure 10. Pose results during occlusions

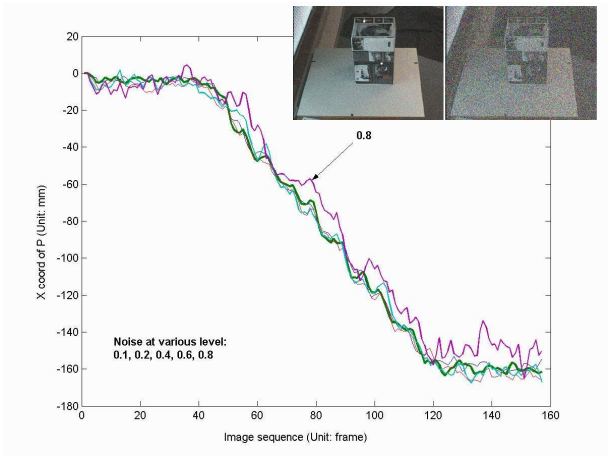

Figure 11. Artificial noise during pose estimation. The left image has a random noise factor of 0.2 and the right one of 0.6.

projected surface patches with image data. Several interesting properties and aspects can be summarized: Firstly, the fusion of a local (block) matching approach with a (global) pose estimation procedure allows to deal with the aperture problem. Though the motion field is not very accurate, the pose is stable since all available information is used simultaneously for pose estimation. Secondly, the approach uses all available information at once and can deal dynamically with changing views and faces which become visible and occluded during different frames. Thirdly, no extra image processing is required, such as silhouette extraction. To achieve this, a more complex model (with texture information) is required. To deal with changing lighting conditions, a remapping is performed during tracking by using the result of the previous frame. This turns out to be fast and stable. Fourthly, the approach is independent from changing backgrounds and can deal with partial occlusions during pose estimation by detecting and eliminating outliers. The whole algorithm can be seen as a successful interaction of computer graphics capabilities within a computer vision application. Though computer vision and computer graphics are often treated as two different disciplines, the interaction of both disciplines can be advantageous, as shown here.

Acknowledgments

This work has been supported by the DFG projects RO 2497/1-1 and RO 2497/1-2.

References

[1] Goddard J.S. Pose and Motion Estimation From Vision Using Dual Quaternion-Based Extended Kalman Filtering. *University of Tennessee, Knoxville*, Ph.D. Thesis, 1997.

[2] Grimson W. E. L. *Object Recognition by Computer*. The MIT Press, Cambridge, Massachusetts, 1990.

[3] Gruen A and Baltsavias E. Adaptive least squares correlation with geometrical constraints. Proceedings of *SPIE* (The society of photo-Optical Instrumentation Engineers), Vol. 595, pp. 72-82, 1985.

[4] Foerstner W. On feature based correspondence algorithm for image segmentation and least squares matching. *Int.*

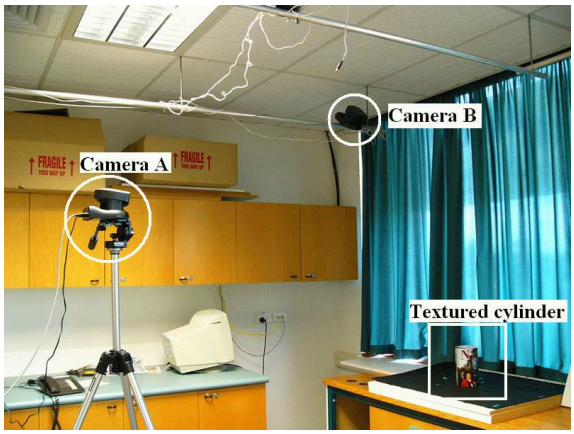

Figure 12. The stereo setup.

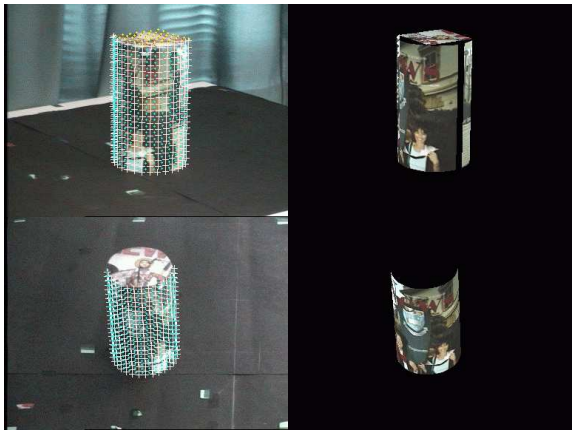

Figure 13. Pose results of a cylinder.

Archives of Photogrammetry and Remote Sensing, Vol 26, P3, Rovaniemi, pp. 150-166, 1986.

[5] Koch R. Dynamic 3D scene analysis through synthesis feedback control. *IEEE Pattern Analysis and Machine Intelligence*, Special issue on analysis and synthesis. Vol 15, No 6, pp. 556-568, June 1993.

[6] Murray R.M., Li Z. and Sastry S.S. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, Inc. Boca Raton, FL, USA, 1994.

[7] ORourke J. *Computational Geometry in C*. Cambridge University Press, Cambridge, UK, 1998.

[8] Rosenhahn B., Perwass C. and Sommer G. Pose estimation of free-form surface models. In *Pattern Recognition, 25th DAGM Symposium*, B. Michaelis and G. Krell (Eds.), Springer-Verlag, Berling Heidelberg, LNCS 2781, pp. 574-581, 2003.

[9] Rosenhahn B. Pose Estimation Revisited. *Technical Report 0308, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2003. Available at `http://www.ks.informatik.uni-kiel.de`

[10] Rosenhahn B. and Sommer G. Pose Estimation of Free-form Objects. *Proceedings of the European Conference on Computer Vision, ECCV '04*, Part I, T. Pajdla and J. Matas (Eds.), Springer-Verlag, Berling Heidelberg, LNCS 3021, pp. 414-427, Prague, 2004.

[11] Tsai R. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. 1986. E.g. Available at `http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/rgw/www/TsaiCode.html`. Last accessed at 6.4.2004.

[12] Shi Y. and Sun H. *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards.* CRC Press, Boca Raton, FL, USA, 1999.

[13] Sommer G., (ed.), *Geometric Computing with Clifford Algebra.* Springer, Berlin, 2001.

[14] Tarel J.-P., Civi H. and Cooper D.B. Pose estimation of free-form 3D objects without point matching using algebraic surface models. In *IEEE Workshop Model Based 3D Image Analysis*, IEEE CS Press, Los Alamitos, CA, pp. 13-21, 1998.