Block matching based 2D-3D pose estimation

Bodo Rosenhahn, Harvey Ho and Reinhard Klette

Center for Imaging Technology and Robotics (CITR)
The University of Auckland
Private Bag 92019 Auckland
New Zealand
bros028@cs.auckland.ac.nz

abstract

This article presents an approach for 2D-3D pose estimation which relies on texture information on the surface mesh of an object model. The textured surface mesh is projected in a virtual image and a modified block matching algorithm is used to determine correspondences between midpoints of surface patches and locations in the image data. This is applied to 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 the potential of the chosen approach.

Keywords: 2D-3D pose estimation, block matching

1 Introduction and preliminary work

Pose estimation has been studied in computer vision since its beginnings. It is crucial for many computer and robot vision tasks. Object grasping, manipulation and recognition, or self-localization of mobile robots are typical examples for the use of pose estimation. For a definition of the pose problem, we quote [2]: *By pose we mean the transformation needed to map an object model from its inherent coordinate system into agreement with the sensory data.* This article addresses the *2D-3D pose estimation problem* 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 the object data with the image data. The basic scenario is visualized in Figure 1.

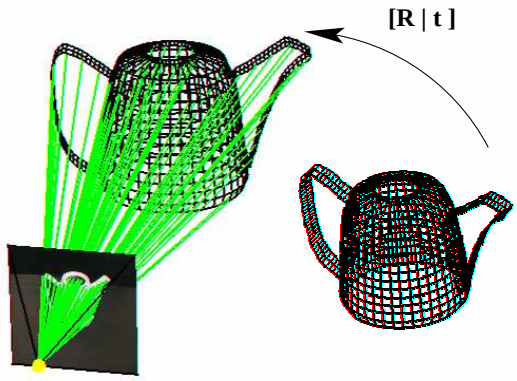

Figure 1: Basic sketch of the 2D-3D pose estimation problem: The task is to find such a rigid motion, defined by a rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$, which leads to a best fit between observed 2D image data and a predefined 3D object model.

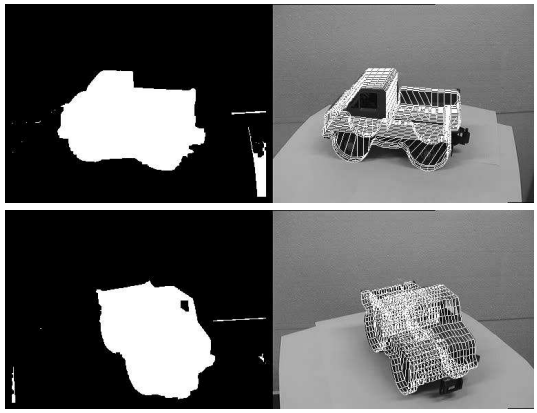

Figure 2: Example images taken from a silhouette based approach for pose estimation.

A crucial question for pose estimation is the object representation, and 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, 8] for overviews. Figure 2 shows example images taken from a silhouette based approach for pose

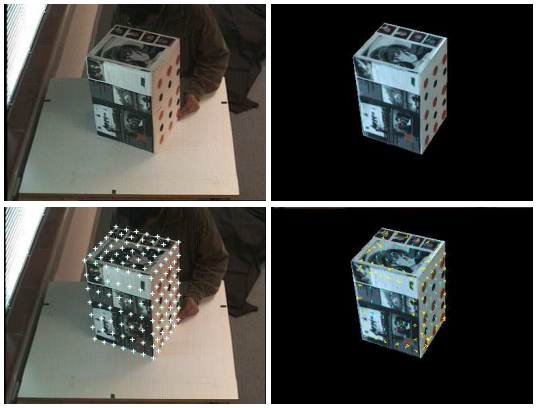

Figure 3: The principle for block matching based pose estimation

estimation of free-form surface models. As it can be seen, just the rim contour in the images is used as feature for pose estimation, and all internal structure information is lost. In this contribution we want to introduce a pose estimation algorithm which relies on texture information on rendered objects. The reason is that using texture information minimizes image pre-processing (e.g., corner extraction, silhouette estimation, etc.), and it allows for the use of as many correspondences as there are visible, in contrast to silhouette based pose estimation, which relies on the object rim. The basic idea is visualized in Figure 3: we assume the representation of a 3D object model as surface mesh and assume a texture mapped onto the object (upper right image). We further observe the represented object in an image of a calibrated camera (upper left image). 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 virtual projected object and the observed object (lower right). This leads to a set of 2D-2D correspondences by using the midpoints of the 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 block matching based pose estimation*. A pose result is shown in the lower left image of Figure 3 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 matching images and estimating a geometric transformation yielding to the closest similarity between images. Even 3D shape knowledge can be incorporated [3], but instead of minimizing within the space of image signals, we propose to estimate a local flow-field and apply a feature based pose estimation algorithm, since a faster (real-time) algorithm can be achieved this way. Before we explain the algorithm in detail, we want to point out that it is basically an approach which combines capabilities known from computer graphics within a computer vision application. Though these two disciplines are often treated separately, similar to [5] we propose a fusion of both disciplines which leads to advanced possibilities for various applications, such as pose estimation. We continue with an introduction to block matching and pose estimation. In the next section we introduce the 2D-3D block matching based pose estimation algorithm and explain the algorithmic steps in detail. Section three continues with first experiments on this approach and section four concludes the contribution with a brief discussion.

1.1 Block matching

Block matching (BM) is employed to measure similarities between two images, or portions of images, on a pixel-by-pixel basis [9]. 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 handle the scene. But no 3D information is entirely embedded in 2D-2D block matching, since only rectangular 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 a block inside of a search window follows some search patterns, which can be classified in two categories, Full Search and Step Search (e.g., Three-Step Search, Four-Step Search or New Three Step Search). During our research, we implemented Full Search, Three-Step search and the 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 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.

1.2 Pose estimation

To deal with geometric aspects of the pose problem, we use Clifford or geometric algebras [10] as mathematical language. Here we will not give a theoretical introduction into the concepts of Clifford algebras but want to point out a few properties which are important for this problem. The elements in geometric algebras are called multivectors which can be multiplied by using a geometric product. Euclidean, projective and conformal geometry [7] can all be expressed in geometric algebra without limiting their potentials. Geometric algebra also enables a coordinate-free and symbolic representation. To model the pose problem, we use conformal geometric algebra (CGA). CGA is build up on a conformal model (i.e., geometry on the sphere) which is coupled with a homogeneous model to deal with kinematics and projective geometry simultaneously. This enables us to deal with the Euclidean, kinematic and projective space in one framework in a unified manner and therefore to cope with the pose problem. Unknown rigid motions are expressed as *motors* which can be applied on different entities (e.g., points or lines) by the use of the geometric product. This leads to compact and easily interpretable equations. In equations we use the inner product $\cdot$, the outer product $\wedge$, the commutator, $\underline{\times}$ and anticommutator product $\overline{\times}$, which can be derived from the geometric product. Though we will also present equations formulated in conformal geometric algebra, we only explain these symbolically and want to refer to [8] for more detailed information.

For 2D-3D point based pose estimation, we are using constraint equations which compare 2D image points with 3D object points. To compare a 2D image point $\boldsymbol{x}$ with 3D object points $\underline{\boldsymbol{X}}$, the idea is to reconstruct from the image point a 3D projection ray, $\underline{\boldsymbol{L}}_x = \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})$ as a Plücker line [4]. The motor $\boldsymbol{M}$ is the exponential of a twist, Ψ, $\boldsymbol{M} = \exp(-\frac{\theta}{2}\Psi)$; it formalizes the unknown rigid motion as a screw motion [4]. The motor $\boldsymbol{M}$ is applied on the object point $\underline{\boldsymbol{X}}$ as a versor product, $\underline{\boldsymbol{X}}' = \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}$, where $\widetilde{\boldsymbol{M}}$ represents the reverse of $\boldsymbol{M}$. Then the rigidly transformed object point $\underline{\boldsymbol{X}}'$ is compared with the reconstructed line $\underline{\boldsymbol{L}}_x$ by minimizing the error vector between the point and the line. The representation of such a constraint equation in geometric algebra takes the form

$$(\boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}) \quad = \quad 0.$$

Note that we work with a 3D formalization of the pose problem. The constraint equations can be solved by linearization (i.e., solving the equations for the twist-parameters which generate the screw motion) and by applying the Rodrigues formula for reconstruction of the group action [4]. Iteration leads to a gradient descent method in 3D space. This is presented in [8] in more detail. There we also introduce similar equations to compare 3D points with 2D lines (3D planes), and 3D lines with 2D lines (3D planes). Pose estimation can be performed in real-time; we need 2ms to estimate a pose containing 100 point correspondences on a Linux 2GHz machine.

2 2D-3D Block matching

The basic flow chart is visualized in Figure 4. We

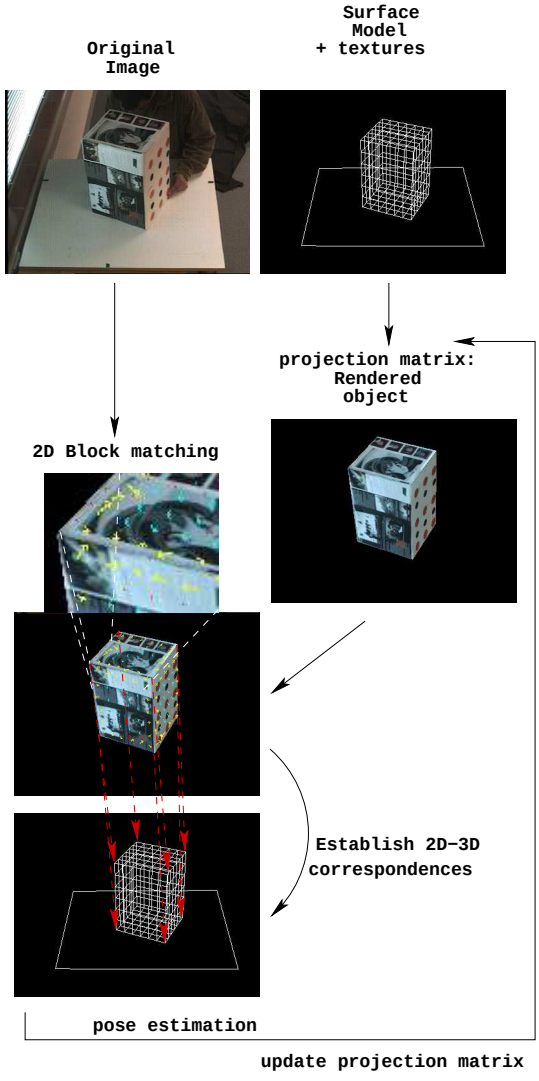

Figure 4: Flow chart of the algorithm.

assume an object model, given as surface mesh and equipped with texture information. Furthermore, we assume an image with 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

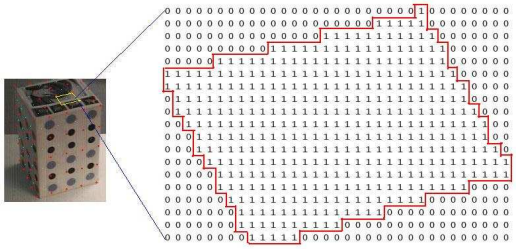

Figure 5: A deformed block mask.

shown in Figure 5. To model a perspective block, we use a rectangular box with a block-mask obtained through the so-called Ray-Crossing algorithm(e.g., introduced in [6]). Therefore we use the bounding box of each search pattern, where the 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 gain 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.

Note that, 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 are changing dynamically.

2.1 Optimizations

One problem during the comparison between the texture mapped object model and the image are the changing lighting conditions which result in the need for a color calibration, or on-line texture updating. We use a process of texture updating, which also allows that reflections on the object can 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 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 between matching in the image and world domain proves to be much more stable than using constant textures.

3 Experiments

We start our experiments with a simple scene. As 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. The camera has a distance of approximately $2m$ to the object and is grabbing the images with a resolution of 384×288 pixels. We implemented the sources in `C,C++` and use OpenGL for rendering and visualization.

The result of a first sequence, dealing with a translational movement of the object, is shown in Figure 6: the object is at the beginning 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. The frames are

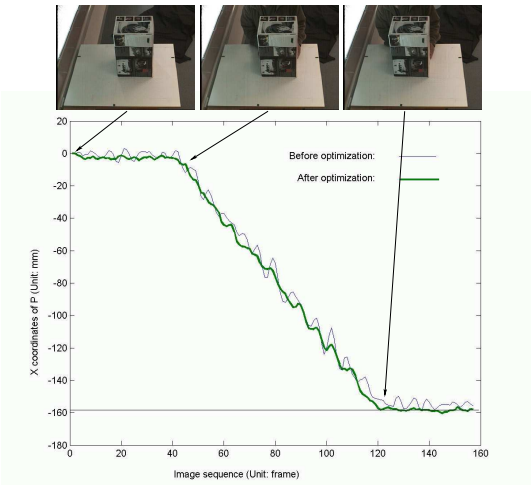

Figure 6: Translation along the x-axis.

grabbed from a static camera observing the object.

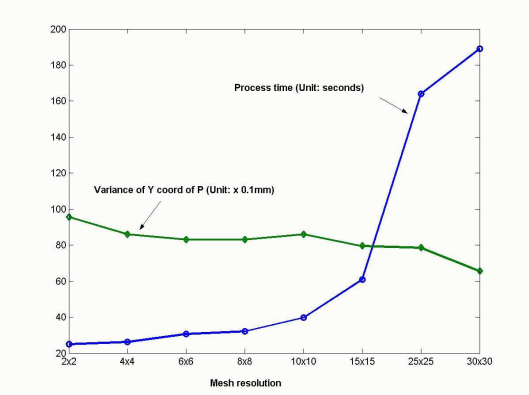

Figure 7: Variance versus computing time.

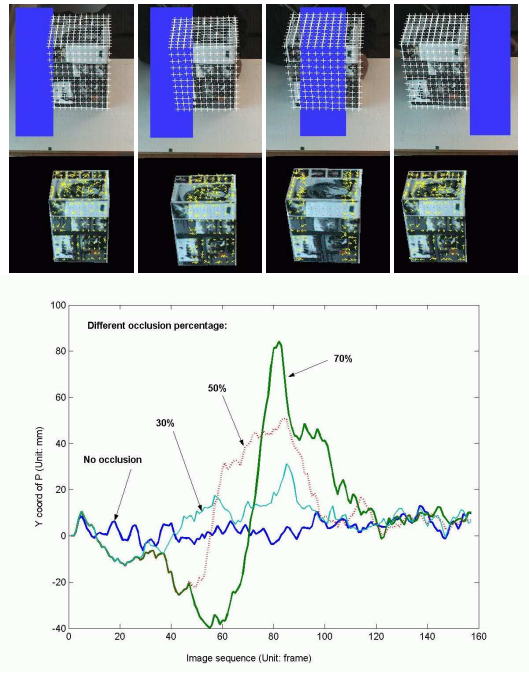

Figure 8: Different occlusions during an image sequence containing a non-moving object

The diagram in Figure 6 shows the frame number along the x-axis and the value of the x-coordinate in the world coordinate system along the y-axis. For the parts of the sequence where the object is not moving, the pose result should be a constant function as ground truth. The other frames should be nearly linear, but due to the manual motion of the object, it is not possible to gain a real ground truth. The estimations of the x-coordinate values during the sequence using the standard approach (without optimization) are shown as thin line in Figure 6. It can be seen that a small object motion can be observed, caused by errors during camera calibration, object measurement and block matching. It varies between $-4 \ldots 4$ mm in space. After applying our optimization methods, we gain a variance between $-2 \ldots 2$ mm in space (shown as thick line). Also the frequency of object motion is reduced.

We continue our experiments with an analysis based on mesh resolution: each side of the box is subdivided in $n \times n$ patches. Obviously, varying the mesh-resolution leads to different results: the more patches are used, the more correspondences for pose estimation can be established, but with increasing number of patches the computing time will increase, too. The aim is to find a suitable mesh resolution with tolerable processing time.

Therefore we analyze the number of patches on the surface grid and compare the computing time versus the accuracy of block matching and the pose result. As a scene, we use the translational movement between frame 40 and 120 of the sequence (with the non texture updating) shown in Figure 6. The results are shown in Figure 7.

On the sequence we apply different mesh sizes (sampled on the x-axis) and compare the coordinates of the pose result with the ground truth. The variance of the Y-coordinate is shown on the y-axis (in 0.1 mm) in the diagram. The second line shows the computation time for the different patch sizes (over the whole sequence). It can be seen, that a mesh resolution between 4×4 and 15×15 does not change the variance too much, but the computing time is increasing significantly beginning at 10×10. Therefore we use a mesh resolution of 6×6, or 8×8 patches for the experiments as in the previous experiment.

Figure 8 shows the stability of our approach with respect to 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 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 9 shows another image sequence during rotation of the object along the y-axis. It can be seen that we are able to handle the visible and non-visible surface patches adaptively. When the right-hand side of the object is coming into view

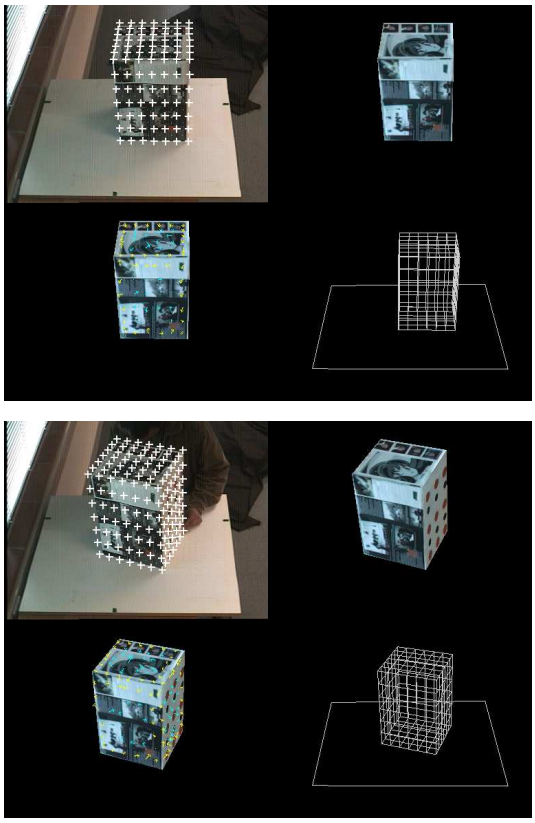

Figure 9: Rotation along y-axis. As can be seen, our algorithm is able to handle non-visible and visible surface patches, adaptively.

(and providing the algorithm with additional image patches), they are automatically taken into account. The white crosses in the upper left frames indicate the matching quality between the object and image data by projecting the corners of the 3D object model into the image.

4 Discussion

This contribution presents an approach for model based pose recovery, which is based on textures mapped onto the object. We adopt a standard block matching algorithm to a 2D-3D version and use correspondences from midpoints of surface patches to the image data as 2D-3D point information for pose estimation. We present experiments with rotational and translational movements, and analyze the pose quality for "more or less controlled" scenes. We performed experiments on various image sequences, including free motions, and achieved good pose results in the range of a few millimeters. For many tasks, like object grasping, this is sufficient, and we will continue our experiments with more complex objects and scenes.

Acknowledgments

This work has been supported by the EC Grant IST-2001-3422 (VISATEC) and by the DFG grants RO 2497/1-1, 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] 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.

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

[5] Netravali A.N., Salz J. Algorithms for estimation of three-dimensional motion. *AT&T Technical Journal*, Vol. 64, No.2, pp. 335-346, 1985.

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

[7] Perwass C. and Hildenbrand D. Aspects of Geometric Algebra in Euclidean, Projective and Conformal Space. An Introductory Tutorial. *Technical Report 0310, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2003.

[8] 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`

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

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