Occlusion Modeling by Tracking Multiple Objects*

Christian Schmaltz[1], Bodo Rosenhahn[2], Thomas Brox[3], Joachim Weickert[1], Daniel Cremers[3], Lennart Wietzke[4], and Gerald Sommer[4]

[1] Mathematical Image Analysis Group, Faculty of Mathematics and Computer Science, Building E1.1, Saarland University, 66041 Saarbrücken, Germany
{schmaltz,weickert}@mia.uni-saarland.de
[2] Max-Planck Institute for Informatics, 66123 Saarbrücken, Germany
rosenhahn@mpi-sb.mpg.de
[3] Department of Computer Science,University of Bonn, 53117 Bonn, Germany
{brox,dcremers}@cs.uni-bonn.de
[4] Institute of Computer Science, Christian-Albrecht-University, 24098 Kiel, Germany
{lw,gs}@ks.informatik.uni-kiel.de

Abstract. This article introduces a technique for region-based pose tracking of multiple objects. Our algorithm uses surface models of the objects to be tracked and at least one calibrated camera view, but does not require color, texture, or other additional properties of the objects. By optimizing a joint energy defined on the pose parameters of all objects, the proposed algorithm can explicitly handle occlusions between different objects. Tracking results in simulated as well as real world scenes demonstrate the effects of occlusion and how they are handled by the proposed method.

1 Introduction

This article deals with 2-D–3-D pose tracking of multiple objects, which is the task to pursuit the 3-D positions and orientations of known 3-D object models from a 2-D image data stream [7]. Pose tracking has a wide range of applications, e.g. self localization and object grasping in robotics, or camera calibration. Although the initial work of Lowe [10] was published more than a quarter of a century ago, pose tracking is still a challenging problem, especially in scenes with cluttered backgrounds, partial occlusions, noise, or changing illumination.

A problem similar to pose tracking is pose estimation. The difference is that there is usually no initial pose given in pose estimation, but only a single pose must be estimated. In this article, we will concentrate on pose tracking and not on pose estimation. Thus, the problem to find the necessary approximate model pose for the first frame will not be discussed.

A lot of different approaches for pose tracking have been considered [6]. A common idea is to use feature matching. The features used range from points [1] over lines [3] to more complex features such as vertices, t-junctions, cusps, three-tangent junctions, limb

* We acknowledge funding by the German Research Foundation under the projects We 2602/5-1 and SO 320/4-2, and the Max-Planck Center for Visual Computing and Communication.

and edge injections, and curvature L-junctions [9]. Drummond and Cipolla used edge-detection to achieve real-time tracking of articulated object with their iterative algorithm [5]. In [2], Agarwal and Triggs describe learning-based methods that use regression for human pose tracking. Another learning based approach was proposed by Taycher et al. in [16], in which an undirected conditional random field is used. Moreover, methods based on neural networks [17] have been introduced. Another possible approach to pose tracking is to match a surface model of the tracked object to the object region seen in the images. In doing so, the computation of this region yields a typical segmentation problem. It has been suggested to optimize a coupled formulation of both problems and to solve simultaneously for both the contour and the pose parameters via level sets [4]. In [13], it was proposed to estimate the 3-D pose parameters by minimizing an energy function directly defined on the images, i.e. without using segmentation as an intermediate step. In the present paper, we build upon this framework.

Most works on 3D tracking concentrate on a single object that used to be fully visible in the image. Usually, the techniques run into severe problems when objects occlude each other. In the present work, we deal with such scenes that contain multiple, partially occluding objects, and show that the corresponding problems can be avoided, if the occlusions are explicitly modeled in the tracking framework. Some related works on multiple object tracking are those in [8], where particle filters and a Gibbs sampler are employed for 2-D tracking of a changing number of objects. The same problem is solved in [15] with a Rao-Blackwellized sequential Monte Carlo method. As both works state only a 2-D tracking in the image domain, they are very restricted in handling mutual occlusions.

Our paper is organized as follows: In the following section, we will briefly review the basics of pose estimation from 2-D–3-D point correspondences. After that, an approach for pose tracking of single objects is described in Section 3, followed by an explanation how the algorithm can be extended to yield improved results by tracking several objects. Experimental results are presented in Section 5. Section 6 concludes with a summary.

2 Pose Estimation from 2-D–3-D Point Correspondences

This section introduces basic concepts and notation and briefly describes the point-based pose estimation algorithm in [12]. The main idea is to use 2-D–3-D point correspondences (x_i, q_i), i.e. 3-D points x_i on the object model, which are visible as 2-D points q_i in an image, to find the rigid motion of the object. Section 3 shows how such point correspondences are obtained with our method.

2.1 Rigid Motion and Twists

A rigid body motion in 3-D, i.e. an isomorphism that preserves orientation and distances, can be represented as $m(x) := Rx + t$, where $t \in \mathbb{R}^3$ is a translation vector and $R \in SO(3)$ is a rotation matrix with $SO(3) := \{R \in \mathbb{R}^{3\times 3} : \det(R) = 1\}$. By means of homogeneous coordinates, we can write m as a 4×4 matrix M:

$$m((x_1, x_2, x_3)^T) = M(x_1, x_2, x_3, 1)^T = \begin{pmatrix} R_{3\times 3} & t_{3\times 1} \\ 0_{1\times 3} & 1 \end{pmatrix} x \, . \tag{1}$$

Rigid motions are of interest to us, since a rigid body can only perform a rigid motion. The set of all rigid motions is called the *Lie group* $SE(3)$. To every Lie group there is an associated Lie algebra, whose underlying vector space is the tangent space of the Lie group evaluated at the origin. The Lie algebras associated with $SO(3)$ and $SE(3)$ are $so(3) := \{A \in \mathbb{R}^{3\times 3} | A^T = -A\}$, and $se(3) := \{(v,\omega) | v \in \mathbb{R}^3, \omega \in so(3)\}$, respectively. Since elements of $se(3)$ can be converted to $SE(3)$ and vice versa, we can represent rigid motions as elements of $se(3)$. Such elements are called *twists*. This is advantageous since a twist has only six parameters while an element of $SE(3)$ has twelve. Both have six degrees of freedom, though.

Since elements of $so(3)$ and $se(3)$ can be written both as vectors $\omega = (\omega_1, \omega_2, \omega_3)$, $\xi = (\omega_1, \omega_2, \omega_3, v_1, v_2, v_3)$ and as matrices,

$$\hat{\omega} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \in so(3), \qquad \hat{\xi} = \begin{pmatrix} \hat{\omega} & v \\ 0_{3\times 1} & 0 \end{pmatrix} \in se(3) \, , \tag{2}$$

we distinguish these two ways of representing elements by a hat sign. Thus, the matrix $\hat{\xi}$ and the vector ξ are always two different representations of the same element. A twist $\xi \in se(3)$ can be converted to an element of the Lie group $M \in SE(3)$ by the exponential function $\exp(\hat{\xi}) = M$. This exponential can be computed efficiently with the Rodriguez formula. For further details we refer to [11].

2.2 From 2-D–3-D Point Correspondences to a Linear Least Squares Problem

Let (q,x) be a 2-D–3-D point correspondence, i.e. let $x \in \mathbb{R}^4$ be a point in homogeneous coordinates on the 3-D silhouette of the object model and $q \in \mathbb{R}^2$ its position in the image. Furthermore, let $L = (n,m)$ be the Plücker line [14] through q and the corresponding camera origin. The distance of any point a to the line L given in Plücker form can be computed by using the cross product: $\|a \times n - m\|$, i.e., $a \in L$ if and only if $\|a \times n - m\| = 0$.

Our goal is to find a twist ξ such that the transformed points $\exp(\hat{\xi})x_i$ are close to the corresponding lines L_i. Linearizing the exponential function $\exp(\hat{\xi}) = \sum_{k=0}^{\infty} \frac{\hat{\xi}^k}{k!} \approx I + \hat{\xi}$ (where I is the identity matrix), we like to minimize with respect to ξ:

$$\sum_i \left\| \left(\exp\left(\hat{\xi}\right) x_i \right)_{3\times 1} \times n_i - m_i \right\|^2 \approx \sum_i \left\| \left(\left(I + \hat{\xi} \right) x_i \right)_{3\times 1} \times n_i - m_i \right\|^2 \to \min, \tag{3}$$

where the function $\cdot_{3\times 1} : \mathbb{R}^4 \mapsto \mathbb{R}^3$ removes the last entry, which is 1.

Evaluation yields three linear equations of rank two for each correspondence (q_i, x_i). Thus, three correspondences are sufficient to obtain a unique solution of the six parameters of the twist. Usually, there are far more point correspondences and one obtains a least squares problem, which can be solved efficiently with the Householder algorithm. Since the twist ξ only corresponds to the pose change it is usually rather small, which justifies the linearization. In order to also allow for larger motions, we iterate this minimization process. This comes down to a variant of the Gauss-Newton method.

3 Region-Based Model Fitting

A lot of existing contour-based pose estimation algorithms expect an explicit contour to establish correspondences between contour points and points on the model surface. This involves a matching of the projected surface and the contour. Here we avoid explicit computations of contours and contour matching. Instead, we stick to [13] and seek to adapt the pose parameters in such a way that the projections of the surface optimally split all images into the object and the background region. For simplicity, we will first review this setting for a single rigid object. The extension to multiple objects, which is the main focus of this paper, will be explained later in Section 4.

3.1 Energy Model

Like in a segmentation task, we seek an optimal partitioning of the image domain Ω. This can be expressed as minimization of the energy function

$$E(\xi) = -\int_{\Omega} \big(P(\xi,q)\log p_1 + (1-P(\xi,q))\log p_2\big)dq\ , \tag{4}$$

where the function $P : \mathbb{R}^6 \times \Omega \ni (\xi,q) \mapsto \{0,1\}$ is 1 if and only if the surface of the 3-D model with pose ξ projects to the point q in the image plane. P splits the image domain into two parts, in each of which different feature distributions are expected. These distributions are modeled by *probability density functions (pdf)* p_1 and p_2. Such pdfs also occur in variational segmentation methods [4], where a functional similar to this function is sought to be minimized. However, while in variational segmentation algorithms the partitioning is represented by a contour, i.e. a function, (4) implies only six optimization variables. Moreover, there is no need for a regularization of the object boundary, which can reduce the accuracy of tracking [13]. In order to model the image features by pdfs, we first have to decide which features should be modeled. For the experiments presented later, we have used the color in CIELAB color space.

Since the two pdfs p_1 and p_2 are unknown, we must assume an underlying model to estimate them. We track objects with uniform appearance by means of a non-parametric Parzen density and object with a varying appearance with a local Gaussian distribution [4]. Since there is not enough data available to accurately estimate a multi-dimensional pdf, we consider the separate feature channels to be independent. Thus, the total probability density function is the product of the single channel densities. As soon as the estimated pose changes, and thus the induced partitioning, p_1 and p_2 are recomputed.

3.2 Minimization

Since $E(\xi)$ in (4) is a multi-dimensional function on an open domain, we know from basic calculus that $\nabla E(\xi)$ must vanish at a minimum of $E(\xi)$. However, this nonlinear equation system is far too complex to be solved directly. Hence, we make use of a gradient descent that should result in the desired pose that minimizes $E(\xi)$ locally. In order to compute the gradient of $E(\xi)$, we assume that the function P is differentiable. Then we get:

$$\nabla E(\xi) = -\int_{\Omega} (\nabla P(\xi,q)(\log p_1 - \log p_2))\, dq\ . \tag{5}$$

Thus, the energy function (4) is minimized by moving each point on the contour of the projected model to the direction indicated by the gradient ∇P. This movement is transfered to corresponding 3-D points on the surface model by using the framework from Section 2. In this way, we estimate the rigid body motion necessary to change the 2-D silhouette in such a way that different features are separated more clearly.

More precisely, we create 2-D–3-D point correspondences (q_i,x_i) by projecting silhouette points x_i, using the current pose ξ, to the image plane where they yield q_i. Each image point q_i obtained in this way which seems to belong to the object region – i.e. those points for which $p_1(q_i)$ is greater than $p_2(q_i)$ – will be moved in outward normal direction to a new point q_i'. Points where $p_1(q_i) < p_2(q_i)$ holds will be shifted into the opposite direction to q_i', respectively. In order to compute the normal direction ∇P, we use Sobel operators. Experimental results indicate that the length $l := \|q_1 - q_2\|$ of the shift vector should be set to a constant depending on the sequence, since the results from experiments with varying l were inferior to those obtained with a constant l.

The 2-D–3-D point correspondences (q_i',x_i) obtained in this way are used in the point based pose tracking algorithm explained above to get a new pose. This forms one optimization step. This step is iterated until the pose changes induced by the force vectors will start to mutually cancel each other. We stop iterating when the average pose change after up to three iterations is smaller than a given threshold. Before changing frames in an image sequence, we predict the object's pose in the new frame by linearly extrapolating the results from the two previous frames. This prediction is very simple and fast, but leads to improved results in case of fast moving objects.

4 Extension to Multiple Objects

The tracking algorithm presented in the last section works fine if there is only a single object in the scene. In this section, we discuss possible problems that can occur as soon as there is more than one object to be tracked and how the tracking framework can be extended in order to deal with such scenes.

4.1 Uncoupled Tracking of Multiple Objects

The basic idea when tracking n objects simultaneously is as follows: Instead of minimizing the energy function (4), which depends on only one pose, the goal is to minimize an energy function depending on the poses of all objects $\xi_1, \ldots, \xi_n$, i.e.

$$E(\xi_1,\ldots,\xi_n) = -\sum_{i=1}^{n} \int_{\Omega} \big(P_i(\xi_i,q)\log p_{i,1} + (1-P(\xi_i,q))\log p_{i,2}\big)dq \,. \tag{6}$$

This function can be minimized in basically the same way as in the single object case: After projecting every object to the image plane, 2-D–3-D point correspondences are gathered along the 2-D silhouette of each object. These correspondences are adapted depending on the pdfs for the inside and outside regions and used to estimate a new pose. Once the movement of one object is below the requested threshold, the iterations on this object can be stopped.

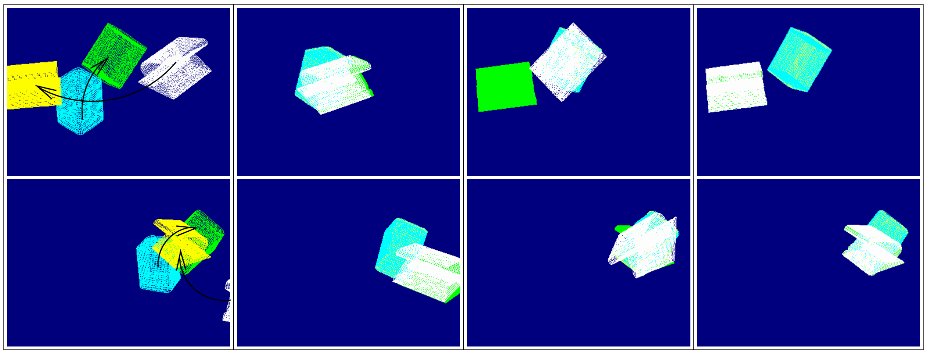

Fig. 1. Leftmost: Here, the 3-D movement which the objects perform is illustrated in the two available views; The puncher moves from the white to the yellow pose while the tea box moves from the cyan to the green pose. The arrows indicate the directions in which the two objects move. **Left:** Tracking result for frame 9 with uncoupled tracking. This is the first frame in which the estimated pose of the puncher is imprecise due to the occlusions. **Right:** Tracking result for frame 17 with uncoupled tracking. As explained in Section 4.1, the estimated pose of the puncher is close to the yellow tea box and is thus incorrect. **Rightmost:** Tracking result for frame 17 with the proposed coupled algorithm. It can be seen that the estimated pose is far better when using the proposed algorithm. **Top:** View 1, **Bottom:** View 2.

One problem that can occur, though, is that one object might occlude a large portion of another object. Although the algorithm can deal with occlusions up to a certain extend, it must fail if too much of the object to be tracked is occluded in the image(s).

To understand the problem, consider Figure 1. In this simulation with 20 frames, the projected model of a tea box (yellow) moves from left to right while the projected model of a puncher (green) moves from right to left. Both objects also rotate slowly. The projections of the models overlap in frames 5 to 16 (first view) and from frame 7 until the end of the sequence (second view). Since the models actually penetrate each other, the projection of the puncher is in front of the projected tea box in some places while it is the other way round in other places.

Since the objects are obviously clearly separated from each other as well as from the background, and since two views are available, this scene should be very easy to track. As can be seen in this figure, the green puncher is not tracked correctly with the current algorithm. This happens because most of the puncher is occluded from the yellow tea box for some frames, i.e. large parts of the puncher region contain yellow pixels. Consequently, the information of the puncher being mainly yellow is included into the pdfs. The algorithm tries to follow the motion of this "mostly yellow" puncher, in fact following the occluding yellow tea box.

4.2 Coupling the Tracking

To solve the problem described in the last section, we change the energy function (6) in such a way that each image point is considered as inside the object region for at most one point. To achieve this, we define $O_i(\xi_i, q)$ as the set of all 3-D points on the object model of the ith object in the pose ξ_i that are projected to the image point q. Furthermore, for the normal Euclidean metric d, let $d_i(\xi_i, q) := d(O_i(\xi_i, q), C)$ be the minimal distance from the camera origin C to a 3-D point in the set $O_i(\xi_i, q)$, i.e.

$$d_i(\xi_i, q) := \min_{x \in O_i(\xi_i, q)} \{d(x, C)\} \ . \tag{7}$$

Finally, define

$$v_i(\xi_1, \ldots, \xi_n, q) = \begin{cases} 1 \text{ if } d_i(\xi_i, q) = \min_{j \in \{1,\ldots,n\}} \{d_j(\xi_j, q)\} \ , \\ 0 \text{ else} \ . \end{cases} \tag{8}$$

Then, the integral to be minimized is:

$$\begin{aligned} E(\xi_1, \ldots, \xi_n) = -\sum_{i=1}^{n} \int_{\Omega} \quad & \big[v_i(\xi_1, \ldots, \xi_n, q) P_i(\xi_i, q) \log p_{i,1} \\ & + (1 - v_i(\xi_1, \ldots, \xi_n, q) P(\xi_i, q)) \log p_{i,2}\big] dq \ . \end{aligned} \tag{9}$$

In other words, the function $P_i(\xi_i, q)$ is multiplied by a visibility indicator function $v_i(\xi_1, \ldots, \xi_n, q)$, which is 1 if there is no point closer to the camera origin on a different object that is also projected to q, and 0 else. Note that this is a more complex setting than simply stating that one object is in front of another, because the objects can also partially occlude each other.

Algorithmically, this means that those parts of the projected objects which are occluded by another object are discarded in the calculation of the object interior. This results in different pdfs, a different silhouette and thus different 2-D–3-D point correspondences.

Additionally, instead of using all points on the new silhouette for pose estimation, only those points which are on the 2-D silhouette before and after omitting the occluded model parts are used. This is advantageous because, although such points are on the visible silhouette of the projected model, the corresponding 3-D points are not on the 2-D model silhouette as seen from the camera.

The reason why those points are not used might get clearer when looking at the idea behind the pose tracking algorithm: Every contour point "votes" for the direction in the image in which the projected model should move to get closer to the contour of the actual object seen in the image. Thus, the point would benefit from moving the projected model into that direction. However, the points that will be omitted would not benefit: Such a point is either below the other object (if it is moved in outward normal direction) or in the object interior (if it is moved in inward normal direction) after any amount of movement. In both cases, it is not a silhouette point any more, and cannot be used for object tracking anymore.

In contrast to the uncoupled case, every object must be tracked until every object movement is below the requested threshold. This is necessary because every part of the integral depends on all poses, which is not the case for uncoupled tracking.

Although it is possible to choose different parameters for each object (e.g. a different parameter l, a different threshold, other image features etc.), this vastly increases the number of parameters. For the experiments presented here, all parameters are equal for all tracked objects.

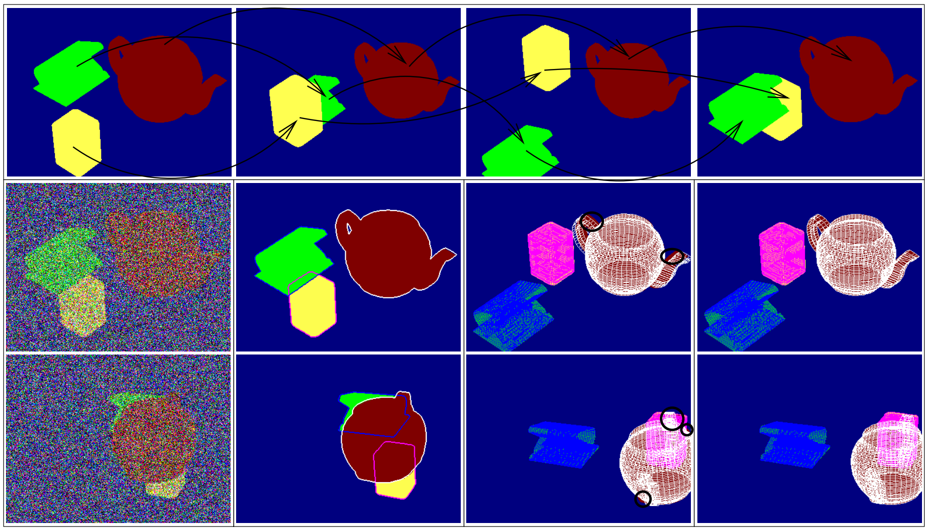

Fig. 2. First row: Four input images (frames 8, 15, 22 and 29) of one of the views. The arrows indicate the movement of the objects. **Leftmost:** Input views for frame 9, including the independent Gaussian noise with a standard deviation of 256. **Left:** Contours of the tracking results of frame 9, shown in images without noise. Note the multiple occlusions. **Right:** Pose results for frame 26 of this sequence. Again, the noise that was added for the pose tracking was removed for the presentation of the results. The black ellipses show areas where it is possible to see that the tracking of the teapot is not optimal due to the noise. **Rightmost:** Pose results obtained without noise in frame 26. **Middle row:** View 1, **Last row:** View 2.

5 Experiments

In this section, we show several tracking results for different objects obtained with the proposed algorithm.

Since the scene shown in Figure 1 is very simple, we present another simulated scene (cf. Figure 2) that was degraded with uncorrelated Gaussian noise with a standard deviation of 256. This time, an additional third object (a teapot projected in dark red) must be tracked. All objects move in a circle with radius 7cm around a certain point with a speed of one full rotation every 25 frames. Since the yellow tea box and the green puncher circle around the same center, the tea box occludes the puncher in some frames while it is the other way round in other frames. The red teapot performs only a slight movement when seen from the first view and a strong movement as seen from the second view, which further complicates the tracking. As can be seen, simple simulated scenes in which the objects are clearly distinguished can be tracked even with a high amount of noise and in the presence of several occlusions.

In Figure 3, tracking results for a real world stereo scene are shown. In this scene, the objects to be tracked are built from Lego Duplo® bricks. The object built with

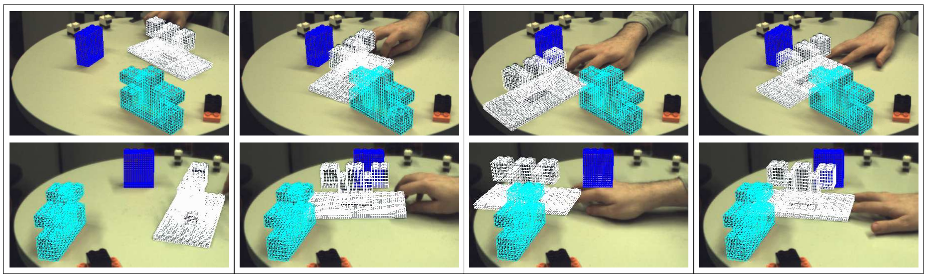

Fig. 3. From Left to Right: Tracking results for frame 10, 90, 140 and 170 of three Lego Duplo objects. (cropped) **Top:** View 1, **Bottom:** View 2.

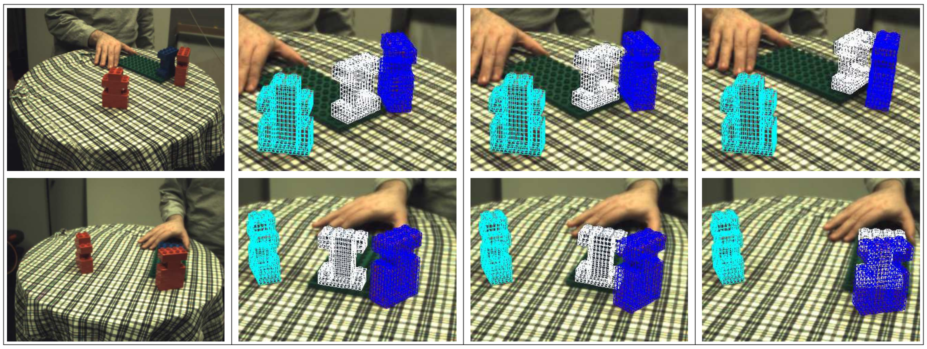

Fig. 4. From Left to Right: Input images for frame 110 and tracking results of the three Lego Duplo objects for the frames 50, 80 and 110 (cropped). **Top:** View 1, **Bottom:** View 2. Note that the blue object is nearly completely occluded in the second view of frame 110.

blue, light green, and dark green bricks moves between the two objects build from red, yellow, and ocher bricks. Thus, it both occludes and is occluded. As can be seen, all three objects have been tracked simultaneously with the proposed algorithm.

Figure 4 shows tracking results for another stereo sequence in which three different objects have been tracked. Again, the objects have been made from Lego Duplo bricks. One of the objects is made from blue bricks while the other two are made from red bricks. Although the blue object is nearly completely occluded in the second view, the tracking results with the new algorithm are still good.

6 Summary

We have presented a region based method for coupled pose tracking of multiple objects that can handle an arbitrary number of objects. In particular, the simultaneous 3-D tracking of multiple objects allows to model mutual occlusions of these objects explicitly. We introduced a visibility function for this purpose. This way, even cases where two objects partially occlude each other are handled correctly. We presented tracking results for simulated as well as real world scenes to demonstrate that the proposed algorithm is able to track different objects in different scenes.

References

1. Abidi, M.A., Chandra, T.: Pose estimation for camera calibration and landmark tracking. In: Proc. International Conf. Robotics and Automation, Cincinnati, May 1990, vol. 1, pp. 420–426 (1990)
2. Agarwal, A., Triggs, B.: Recovering 3D human pose from monocular images. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(1) (January 2006)
3. Beveridge, J.: Local Search Algorithms for Geometric Object Recognition: Optimal Correspondence and Pose. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst (May 1993)
4. Brox, T., Rosenhahn, B., Weickert, J.: Three-dimensional shape knowledge for joint image segmentation and pose estimation. In: Kropatsch, W.G., Sablatnig, R., Hanbury, A. (eds.) Pattern Recognition. LNCS, vol. 3663, pp. 109–116. Springer, Heidelberg (2005)
5. Drummond, T., Cipolla, R.: Real-time tracking of multiple articulated structures in multiple views. In: Vernon, D. (ed.) ECCV 2000. LNCS, vol. 1843, pp. 20–36. Springer, Heidelberg (2000)
6. Goddard, J.: Pose And Motion Estimation From Vision Using Dual Quaternion-Based Extended Kalman Filtering. PhD thesis, Imaging, Robotics, and Intelligent Systems Laboratory, University of Tennessee, Knoxville-College of Engineering, Tennessee (1997)
7. Grimson, W., Lozano–Perez, T., Huttenlocher, D.: Object Recognition by Computer: The Role of Geometric Constraints. MIT Press, Baton (1990)
8. Hue, C., Cadre, J.L., Perez, P.: Tracking multiple objects with particle filtering. IEEE Trans. on Aerospace and Electronic Systems 38(3), 791–812 (2002)
9. Kriegman, D., Vijayakumar, B., Ponce, J.: Constraints for recognizing and locating curved 3D objects from monocular image features. In: Sandini, G. (ed.) ECCV 1992. LNCS, vol. 588, pp. 829–833. Springer, Heidelberg (1992)
10. Lowe, D.: Solving for the parameters of object models from image descriptions. In: Proc. ARPA Image Understanding Workshop, College Park, April 1980, pp. 121–127 (1980)
11. Murray, R.M., Li, Z., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC Press, Boca Raton (1994)
12. Rosenhahn, B., Sommer, G.: Adaptive pose estimation for different corresponding entities. In: Van Gool, L. (ed.) Pattern Recognition. LNCS, vol. 2449, pp. 265–273. Springer, Heidelberg (2002)
13. Schmaltz, C., Rosenhahn, B., Brox, T., Cremers, D., Weickert, J., Wietzke, L., Sommer, G.: Region-based pose tracking. In: Martí, J., Benedí, J.M., Mendonça, A.M., Serrat, J. (eds.) Pattern Recognition and Image Analysis, Girona, Spain, June 2007. LNCS, vol. 4478, pp. 56–63. Springer, Heidelberg (2007)
14. Shevlin, F.: Analysis of orientation problems using Plucker lines. In: Proc. 14th International Conference on Pattern Recognition, Washington, DC, USA, vol. 1, pp. 685–689. IEEE Computer Society Press, Los Alamitos (1998)
15. Särkkä, S., Vehtari, A., Lampinen, J.: Rao-Blackwellized Monte Carlo data association for multiple target tracking. In: Proc. 7th International Conference on Information Fusion, vol. 1, pp. 583–590 (2004)

16. Taycher, L., Shakhnarovich, G., Demirdjian, D., Darrell, T.: Conditional random people: Tracking humans with CRFs and grid filters. In: Proc. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, New York, NY, December 2006, pp. 222–229. IEEE Computer Society Press, New York (2006)
17. Winkler, S., Wunsch, P., Hirzinger, G.: A feature map approach to real-time 3-D object pose estimation from single 2-D perspective views. In: Paulus, E., Wahl, F. (eds.) Mustererkennung 1997 (Proc. DAGM), pp. 129–136. Springer, Berlin (1997)