Universität des Saarlandes

Fachrichtung 6.1 – Mathematik

Preprint Nr. 277

Region Based Pose Tracking with Occlusions using 3D Models

Christian Schmaltz, Bodo Rosenhahn,

Thomas Brox and Joachim Weickert

Saarbrücken 2011

Fachrichtung 6.1 – Mathematik
Universität des Saarlandes

Preprint No. 277
submitted: January 3, 2011

Region Based Pose Tracking with Occlusions using 3D Models

Christian Schmaltz
Saarland University
Department of Mathematics and Computer Science
Mathematical Image Analysis Group
Campus E1.1, 66041 Saarbrücken, Germany
schmaltz@mia.uni-saarland.de

Bodo Rosenhahn
Leibniz University of Hannover
Faculty of Electrical Engineering and Computer Science
Appelstr. 9A, 30167 Hannover, Germany
rosenhahn@tnt.uni-hannover.de

Thomas Brox
Albert-Ludwigs-University Freiburg
Department of Computer Science
Computer Vision Lab
79110 Freiburg, Germany
brox@informatik.uni-freiburg.de

Joachim Weickert
Saarland University
Department of Mathematics and Computer Science
Mathematical Image Analysis Group
Campus E1.1, 66041 Saarbrücken, Germany
weickert@mia.uni-saarland.de

Edited by
FR 6.1 – Mathematik
Universität des Saarlandes
Postfach 15 11 50
66041 Saarbrücken
Germany

Fax:	+ 49 681 302 4443
e-Mail:	preprint@math.uni-sb.de
WWW:	http://www.math.uni-sb.de/

Region Based 3-D Pose Tracking with Occlusions

Christian Schmaltz   Bodo Rosenhahn   Thomas Brox
Joachim Weickert

January 3, 2011

Abstract

Despite great progress achieved in 3-D pose tracking during the past years, occlusions and self-occlusions are still an open issue. This is particularly true in silhouette-based tracking where even visible parts cannot be tracked as long as they do not affect the object silhouette. Multiple cameras or motion priors can overcome this problem. However, multiple cameras or appropriate training data are not always readily available. We propose a framework in which the pose of 3-D models is found by minimising the 2-D projection error through minimisation of an energy function depending on the pose parameters. This framework makes it possible to handle occlusions and self-occlusions by tracking multiple objects and object parts simultaneously. Therefore, each part is described by its own image region each of which is modeled by one probability density function. This allows to deal with occlusions explicitly, which includes self-occlusions between different parts of the same object as well as occlusions between different objects. The results we present for simulations and real-world scenes demonstrate the improvements achieved in monocular and multi-camera settings. These improvements are substantiated by quantitative evaluations, e.g. based on the HumanEVA benchmark.

1 Introduction

Following the 3-D position, orientation and, if present, the articulations of an object in a video is necessary for many applications ranging from self localisation and object grasping in robotics, body language interpretation, human computer interaction, traffic or security surveillance, character animation, analysis of athletes up to content-based video retrieval. For a detailed overview of the field, we refer to the surveys by Gavrila [1], Forsyth *et al.* [2], Moeslund [3], and Poppe [4].

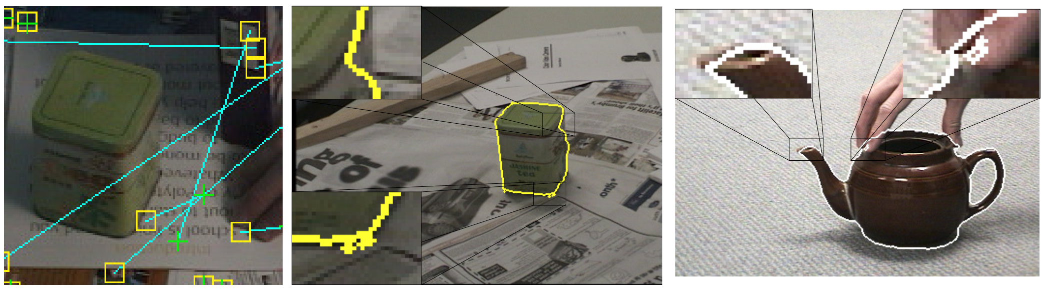

Figure 1: Examples for problems that can occur in feature based and in variational segmentation algorithms: **Left**: SIFT features [12] found in two consecutive frames are depicted. The green crosses denote the positions of SIFT features found in the frame shown, while the yellow boxes show the corresponding feature position in the last frame. Here, the tea box was to be tracked. As can be seen, no SIFT feature has been found that is on the tea box for these two frames, making a tracking with SIFT-features infeasible. See Gall *et al.* [13] for further details. **Middle**: A part of the segmentation is concave although the object to be tracked is convex, and an undesired split up into several connected components. **Right**: The spout is not accurately segmented due to oversmoothing. Furthermore, holes generated in the segmentation are not always correct. From [14].

This task, which is called pose tracking, is bound to some kind of feature matching between entities of the 3-D model and their corresponding features in the image. Such features can be points [5], lines [6, 7], or more complex features such as vertices, T-junctions, cusps, three-tangent junctions, limb and edge injections, and curvature L-junctions [8]. Drummond and Cipolla used edge-detection to achieve real-time tracking of articulated objects [9]. In [10], a combination of two tracking algorithms that use edges and texture information, respectively, are used to achieve improved results. In [11], the pose is predicted using pixel displacements and improved afterwards by using image-based cues such as silhouettes.
Techniques building on keypoint tracking are usually fast, but they also share some common problems. First of all, a sufficient number of common keypoints must be detected in two frames, which is not always the case; see the left image in Figure 1. Moreover, when not matching all images to a common keyframe, e.g. the first frame in the sequence, tracking errors accumulate and result in an undesired drift.
Drift is avoided in detection-based tracking approaches, where a description of an object model is assumed and the instance of this model in the image is sought to be detected in each frame. Reliable detection is a very hard problem. Thus, various assumptions that simplify the problem are commonly applied. Many approaches assume a static background in order to detect the object silhouette with

background subtraction, e.g. [15]. Other approaches assume sufficiently many discriminative features on the object to detect it in a reliable manner [16]. Sun *et al.* learn different object parts from a number of view points to recognise the view point of new images [17]. Ottlik and Nagel segment the optic flow to initialise the position of cars in inner-city sequences [18]. Ramanan *et al.* rely on detection only in frames where it is reliable and follow a tracking approach for the frames in between [19]. Rosenhahn *et al.* assumes a good initialisation of the pose of a 3-D model in one frame and limited motion between frames to update the silhouette and the pose parameters in an iterative approach, where the object model serves as a shape prior for the segmentation [20]. While the requirement of an initial pose and limited motion resemble a classical tracking approach, the matching of the surface model to the silhouette in the image avoids drift and involves the silhouette of the object as a general descriptor (see [21]).
If we are only interested in the pose of the object and not in its contour, the drawback of the technique in [20] is that it solves a much harder problem than actually necessary. In particular, it estimates an intermediate contour in an infinite-dimensional space only to derive a $6+n$-dimensional pose vector, where n is the number of articulations. The many degrees of freedom increase the computation time, decrease the robustness, and lead to inaccuracies in the contour; see Figure 1. Building upon the ideas we presented in an earlier conference paper [14], we propose an energy function that involves optimisation only in a finite low-dimensional space. This method also uses silhouettes as features for tracking. However, instead of separately estimating 2-D segmentation and 3-D pose parameters, we directly estimate 3-D pose parameters by minimising the projection error of a given 3-D model in the respective 2-D images. Thus, the contours obtained with our algorithm are by construction consistent with the object model with the estimated 3-D pose. Since we only need to estimate a small number of pose parameters instead of an infinite-dimensional level set function, the runtime of the algorithm significantly decreases compared to the algorithm described in [20]. Nevertheless, we can use the same general statistical representation of regions. Estimating a separate silhouette may be advantageous if the provided surface model does not fit the tracked object. However, as shown in [22], there are ways to express the typical variations of a surface model in a parametric form. As we do not have access to a rich surface database, we stick to fixed surface models from a 3-D scanner, but the approach would also work with parametric adaptive surfaces. While we restrict the model here to shape deformations at predefined joints (see Section 3.6), the model and optimisation scheme could be extended to include more general shape variations in a similar way as recently shown in [23]. Note that the 3-D object models used here do not include any appearance information (such as colour or texture information).
One reason why contemporary solutions to the tracking task are not yet satisfac-

tory is that there are a lot of issues that make pose tracking challenging. Especially partial occlusions are a big problem for many pose trackers. As a second contribution of this paper, we show how to deal with partial occlusions between different objects by making use of the estimated 3-D positions in order to model occlusions explicitly. To this end, we minimise an energy function jointly defined over multiple objects. In a similar manner, self-occlusions of articulated objects can be handled by using object models consisting of multiple components. This tackles a typical shortcoming of silhouette-based methods, where even visible parts cannot be tracked if their presence is not seen in the object silhouette.

The approach in [24] can handle occlusions between an unknown number of objects. However, the objects must be very clearly distinguishable from each other and from the background. In [25], images from a camera array are aggregated, and the resulting image was used for successfully tracking despite occlusions. In these approaches, tracking is only performed in 2-D, though. In [26], a multi-hypothesis approach to track multiple occluded persons was proposed. It relies on human appearance models and ground plane homographies and tracks the position at which the person is standing. In [27], several occlusion layers are used to detect occlusions with static occluders. Those layers are generated by rendering via computer graphics. Grabner *et al.* propose to handle occlusions by using supporters found in the visual context around the tracked object [28].

Another way to deal with occlusion is by using motion priors (see [29]). As such a prior incorporates information on the most likely continuation of a motion pattern, it principally allows to track even objects that are completely occluded. However, this is only possible if there are only few possible motion patterns, and if the object to be tracked approximately follows the pattern as stated in the prior. Consequently, the results depend on the choice and quality of the motion priors. Furthermore, tracking results are biased towards the priors, which is undesired in applications in which derivations from the usual motion shall be found. This is the case e.g. in medical applications or when analysing the motion of athletes. The method presented in this paper allows the incorporation of motion priors, but they are only required to deal with *full* occlusions. In all our experiments we did not use any prior data from motion databases. Nevertheless, our tracking approach yields results that surpass those of most other state of the art algorithms, as we will demonstrate using the HumanEVA-II benchmark.

This article is organised as follows: In Section 2 we review some basic mathematical concepts and pose estimation from point correspondences needed for our region-based approach introduced in Section 3. The method is extended to multiple objects in 4 and to multiple component models in Section 5. In Section 6, both approaches will be combined such that several objects with multiple components can be tracked. Experiments presented after each chapter illustrate the achieved improvements. Section 7 concludes the paper with a summary.

2 Basics

This section reviews the concept of representing rigid motions as twists, the concept of Plücker lines and the basic point-based pose tracking algorithm used later. An isomorphism that preserves orientation and distances is called rigid motion. Such isomorphisms are of interest to us, since a rigid body can only perform a rigid motion. Any rigid body motion in 3-D can be represented as $m(x) := Rx + t$, with a translation vector $t \in \mathbb{R}^3$ and a rotation matrix $R \in SO(3)$ with $SO(3) := \{R \in \mathbb{R}^{3\times 3} : \det(R) = 1\}$. Using homogeneous coordinates, one can also write m as a 4×4 matrix M:

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

The set containing all matrices of this form is the so-called *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 $se(3) := \{(\nu, \omega) | \nu \in \mathbb{R}^3, \omega \in so(3)\}$, respectively. The elements of $se(3)$ are called *twists*. Elements of a Lie group can be converted to elements of the corresponding Lie algebra, and vice versa. In particular, a rigid motion can be written as a twist. Since a rigid motion given as element of $SE(3)$ has twelve parameters while a twist has six, it makes sense to prefer estimating twists instead of rigid motions given as matrix. Moreover, when solving for the pose parameters in Section 2.2, a twist can easily be linearised.
Since elements of $so(3)$ and $se(3)$ can be written either as vectors $\omega = (\omega_1, \omega_2, \omega_3)$, $\xi = (\omega_1, \omega_2, \omega_3, \nu_1, \nu_2, \nu_3)$ or as matrices

$$\hat{\omega} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \in so(3), \tag{2}$$

$$\hat{\xi} = \begin{pmatrix} \hat{\omega} & \nu \\ 0_{1\times 3} & 0 \end{pmatrix} \in se(3), \tag{3}$$

we distinguish these two ways of representing elements by a hat sign. A twist ξ can be converted to an element of the Lie group $M \in SE(3)$ by the exponential function $\exp(\hat{\xi}) = M$, which can be computed efficiently with the Rodriguez formula (see [30]).

2.1 Plücker forms

3-D lines can be represented in different ways. Here, we use the Plücker form from [31] to represent lines. In the context of this article, we consider projection

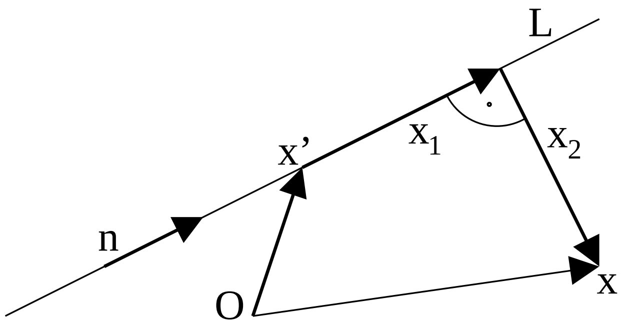

Figure 2: Identifiers used in the proof that the distance between a point x and a Plücker line $L = (n, m)$ is given by $\|x \times n - m\|$.

rays, i.e. lines containing all points which are projected to a certain image point. Consequently, these lines always go through this image point and the camera centre.
A line in Plücker form $L = (n, m)$ is given by a normalised vector n (pointing in the direction of the line) and a momentum $m := x' \times n$ for a point x' on the line. The distance of a point x to such a line $L = (n, m)$ can easily be computed as $\|x \times n - m\|$. To understand this, we write x as sum of x', a vector $x_1 = \lambda n$ parallel to n and a vector x_2 perpendicular to n, i.e. $x = x' + x_1 + x_2$ (see Figure 2) and compute

$$\begin{aligned}\|x \times n - m\| &= \|(x' + \lambda n + x_2) \times n - m\| \\ &= \|\underbrace{x' \times n}_{=m} + \lambda \underbrace{n \times n}_{=0} + x_2 \times n - m\| \\ &= \|x_2 \times n\| = \|x_2\| \;.\end{aligned} \tag{4}$$

2.2 Pose estimation with 2-D–3-D point correspondences

The region based pose tracking approach in this paper uses point correspondences as well as the point based pose estimation algorithm described in [20] to implement a gradient descent. Let (x, X) be a 2-D–3-D point correspondence, i.e. let $X \in \mathbb{R}^4$ be a point on the 3-D silhouette of the object model in homogeneous coordinates and $x \in \mathbb{R}^2$ its position in the image. Furthermore, let $L = (n, m)$ be the Plücker line through x and the respective camera origin. We will explain in Section 3 how such point correspondences emerge from a gradient descent.
In order to track the object, we need to find a twist ξ that maps the transformed model points $\exp(\hat{\xi})X_i$ as close as possible to the corresponding Plücker lines L_i:

$$\arg\min_{\xi} \sum_i \left\| \left(\exp\left(\hat{\xi}\right) X_i \right)_{3\times 1} \times n_i - m_i \right\|^2 \tag{5}$$

where the function $\cdot_{3\times1} : \mathbb{R}^4 \mapsto \mathbb{R}^3$ removes the last entry, which is 1. This results in a system of equations that is hard to solve due to the exponential function. However, since the twist ξ corresponds to the pose change, it is rather “small”. As in [9], we can linearise the exponential function by the first order Taylor expansion

$$\exp(\hat{\xi}) = \sum_{k=0}^{\infty} \frac{\hat{\xi}^k}{k!} \approx I + \hat{\xi} \tag{6}$$

with an identity matrix I without introducing a large error. Then, we get

$$\arg\min_{\xi} \sum_i \left\| \left(\exp\left(\hat{\xi}\right) X_i \right)_{3\times1} \times n_i - m_i \right\|^2 \tag{7}$$

$$\approx \arg\min_{\xi} \sum_i \left\| \left(\left(I + \hat{\xi}\right) X_i \right)_{3\times1} \times n_i - m_i \right\|^2 . \tag{8}$$

To solve this least squares problem, the cross product is evaluated, resulting in three linear equations of rank two for each correspondence (x_i, X_i). Thus, three non-colinear correspondences are sufficient to obtain a unique solution of the six parameters in the twist. Since correspondences are usually not accurate, however, it is advantageous to consider more correspondences. Thus, one obtains a least squares problem, which can be solved efficiently with standard methods, e.g. the Householder algorithm [32]. In order to further minimise the error introduced by the Taylor expansion, we iterate this minimisation process.

3 Region-based Model Fitting

Previous approaches usually try to match a contour obtained by segmentation to the projected model surface. Since segmentation and matching can be time consuming and erroneous, our idea is to avoid both the explicit contour computation as well as the matching step. Instead, we directly optimise the pose parameters such that all images are optimally partitioned into an object and a background region by the projected surface model. To simplify the description, we explain the algorithm for a sequence created with a single camera. The extension to multiple views is straightforward, though.

3.1 Energy model

To find the set of pose parameters that splits the image domain Ω into a foreground region Ω_{in} and a background region Ω_{out}, we minimise the energy function

$$E(\xi) = -\int_{\Omega} \left(P_{\xi}(x) \log p_{\text{in}} + \left(1 - P_{\xi}(x)\right) \log p_{\text{out}} \right) \mathrm{d}x \tag{9}$$

where $P_\xi(x) := P(\xi, x) \in (\mathbb{R}^6 \times \Omega \mapsto \{0,1\})$ is an indicator function for the projected object surface, i.e., it is 1 if and only if the surface of the 3-D model with pose ξ projects to the point x in the image plane. The functions $p_{\text{in}}, p_{\text{out}} : F \times \Omega \mapsto \mathbb{R}$ are two probability density functions (abbreviated as "PDF") that model the different feature distributions. Both take the features F of the image (such as colour or texture) and an image point as input and return the probability that the features at the point x occur inside or outside the object region, respectively. To make the formulas more readable, the arguments of the PDFs are omitted in the equations. Similar PDFs appear in [20] for estimating the contour. In contrast, the energy in (9) does not require the estimation of an intermediate contour but only the optimisation of the six pose parameters. This simplifies the estimation considerably. Also the length constraint on the contour used in [20] is no longer required.

Good and straightforward choices for the input features are the image intensity (for grey-scale images), or the colour in CIELAB colour space [33] (for colour images). These features are used in every experiment shown here. Additionally, we employ the texture feature space in [34] to improve tracking results for objects with a more complex appearance (see Table 2). Any other features which can be densely computed in the image, such as Gabor filters, can also be used.

To keep the model tractable, we assume that all pixels and feature channels are independently distributed. That is, the probability functions $p_{\text{in/out}}$ are given by the product

$$p_{\text{in/out}} = \prod_{i=1}^{N} p^i_{\text{in/out}}, \tag{10}$$

where N is the number of feature channels and p^i_{in} and p^i_{out} are the estimated PDFs of the i-th feature channel for the inside or outside region, respectively. This is also one reason why we use the CIELAB colour space, as it separates the different channels well. If the models appearance is not too complex, we model the PDFs with a non-parametric Parzen density

$$p^i_{\text{in/out}}(F^i(x)) = \frac{1}{|\Omega_{\text{in/out}}|} \int_{\Omega_{\text{in/out}}} K_\sigma(F^i(x) - F^i(y))\,\mathrm{d}y, \tag{11}$$

where $F^i(x)$ is the i-th feature at the image position x, and $K_\sigma(z) = \frac{1}{\sqrt{2\pi}\sigma} \exp(\frac{z^2}{2\sigma^2})$ is a Gaussian kernel with standard deviation $\sigma = \sqrt{30}$, approximated by three box filters of width 11. For more complex appearances, we employ a local Gaussian distribution as introduced in [20] and very common in recent segmentation approaches, e. g. [35] or [36]:

$$p^i_{\text{in/out}}(F^i(x), x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{(F^i(x) - \mu^i_{\text{in/out}}(x))^2}{2\sigma^i_{\text{in/out}}(x)^2} \right), \tag{12}$$

where the local means and standard deviations are computed locally as

$$\mu^i_{\text{in/out}}(x) = \frac{\int_{\Omega_{\text{in/out}}} K_\sigma(y-x)F^i(y)\mathrm{d}y}{\int_{\Omega_{\text{in/out}}} K_\sigma(y-x)\mathrm{d}y}, \tag{13}$$

$$\sigma^i_{\text{in/out}}(x) = \frac{\int_{\Omega_{\text{in/out}}} K_\sigma(y-x)(F^i(y)-\mu^i_{\text{in/out}}(x))^2\mathrm{d}y}{\int_{\Omega_{\text{in/out}}} K_\sigma(y-x)\mathrm{d}y}. \tag{14}$$

3.2 Minimisation

The gradient of the energy function (9) with respect to the pose parameters reads:

$$\begin{aligned} -\nabla E(\xi) = &\int_\Omega (\nabla P_\xi(x)\log p_{\text{in}} - \nabla P_\xi(x)\log p_{\text{out}}) \\ &+ \big(P_\xi(x)\nabla(\log p_{\text{in}}) + (1-P_\xi(x))\nabla(\log p_{\text{out}})\big)\mathrm{d}x \\ \approx &\int_\Omega \big(\nabla P_\xi(x)(\log p_{\text{in}} - \log p_{\text{out}})\big)\,\mathrm{d}x\,. \end{aligned} \tag{15}$$

The approximation neglects the dependency of the pose parameters on the PDFs. Both are actually not independent, since changing the pose usually leads to a different object region in which the PDFs are estimated. There are several works that discussed the use of the complete shape gradient in the scope of segmentation, e.g. [37]. Our own experiments did not reveal a practical advantage, such as considerably faster convergence, and iterations are much faster when using the approximation. The approximation is motivated by the fact that the PDFs usually change very slowly when changing the pose parameters. In our implementation, we use Sobel operators (see [38]) to approximate ∇P.
Since $\nabla E(\xi)$ is zero unless we are at the 2-D silhouette c of the projected object, we can rewrite Equation (15) as

$$-\nabla E(\xi) \approx \int_c \big(\nabla P_\xi(c(s))(\log p_{\text{in}} - \log p_{\text{out}})\big)\,\mathrm{d}s, \tag{16}$$

where s is the arc-length parameterisation of c. In the discrete setting, only a small number of object points is projected onto the 2-D silhouette. Denoting the set of these points by O_S, we have

$$-\nabla E(\xi) \approx \sum_{x_s \in O_S} \nabla P_\xi(x_s)(\log p_{\text{in}} - \log p_{\text{out}})\,. \tag{17}$$

Interpreting (17), the energy function (9) is minimised by moving each contour point along the direction indicated by the gradient ∇P. Speed and sign of this

```
For each frame:
    Extrapolate new pose from previous poses and compute image features
    Iterate
        − Project 3D object model onto image plane
        − Generate PDFs for interior/exterior of projected model (Section 3.1)
        − Adapt 2D−3D point correspondences $(x_i, X_i)$ to $(x'_i, X_i)$ (Section 3.2)
        − Construct projection rays from $(x'_i, X_i)$ (Section 2.1)
        − Generate and solve system of equations to get new pose (Section 2.2)
```

Figure 3: Overview over the steps done by the basic pose tracking algorithm.

motion are given by the log-likelihood ratio $\log\left(\frac{p_{\text{in}}}{p_{\text{out}}}\right)$. Employing the methodology from Section 2.2, this motion can be transferred to corresponding 3-D points on the surface model. Thus, the gradient vectors created this way are reprojected back to the 3-D space in order to find the most consistent 3-D rigid body motion that corresponds to the gradient in the image. Vice-versa, the estimated rigid body motion changes the 2-D silhouette such that the features are separated more clearly.

In practical implementation, we create 2-D–3-D point correspondences (x_i, X_i) by projecting silhouette points X_i using the current pose ξ to the image plane where they yield x_i. If the PDF for the inside region evaluated at x_i, i.e. $p_{\text{in}}(x_i)$, exceeds the corresponding function value for the outside region ($p_{\text{out}}(x_i)$), we suppose that x_i belongs to the object region. Thus, x_i will be moved in outward normal direction to a new point x'_i. Conversely, points where $p_{\text{in}}(x_i) < p_{\text{out}}(x_i)$ holds will be shifted into the opposite direction.

According to the gradient (17) the shift vector should have a length of $l := \|\log p_{\text{in}} - \log p_{\text{out}}\|$. We noticed that setting l to a fixed value tends to work better. We believe that this is because in the optimum the silhouette will usually be close to an image boundary. If the model does not exactly fit the object, parts of the contour will still be in the wrong region and induce large gradient vectors. Note that the absolute value of the difference of the logarithms states how likely it is that a point belongs to a certain region, which does usually not correspond to the distance of the point to the object boundary, and thus to the optimal length of the shift vector. Moreover, image boundaries are often blurred and lead to pixel values that do not fit well to either PDF. Capturing only the sign of the log-likelihood ratio can be regarded as a robust variant of the gradient, where points along the silhouette all have equal weights when voting for the direction of movement. Especially in case of unexpected occlusions, a constant l helps to reduce the influence of points assigned to the wrong region. We tested various other methods to adapt the length of the gradient vector, but results were usually inferior. An example result where l was not fixed is shown in the last image in Figure 10. Note that the optimal value of l depends on various factors such as the acceleration of the object, the type of movement, and the amount of silhouette points. Thus, it is currently a free

parameter in our tracker. All experiments shown use values between 0.1 and 2.
A gradient descent in the pose parameters very similar to the one we first proposed in [14] has recently been presented in [39]. The gradient is derived in 3-D space, which yields an additional multiplicative factor depending on the curvature of the object. In their implementation, however, this factor was neglected. Moreover, they use the exact gradient without discarding the difference $\|\log p_{\text{in}} - \log p_{\text{out}}\|$. While this makes the optimisation theoretically more sound, their approach would probably also benefit from discarding this factor.
After each gradient descent step in the pose parameters, the PDFs in the object and background region are recomputed. In order to have better initial pose parameters - which includes a region that better fits the object in the image - we predict the object's pose in a successive frame by linearly extrapolating the pose from the previous two frames. This is also beneficial when linearising the exponential function. A more sophisticated approach, e.g. using a physical simulation as in [40], could be used here as well. However, even if the initial pose estimated for a frame is quite bad, the optimisation will usually find the correct pose, as we will show later in an experiment. Figure 3 depicts an overview of the algorithm.

3.3 Illustrative example

Figure 4 illustrates the adaptation of the probability densities. The image in this example shows the contour of the pose estimated by our prediction step. PDFs are generated using the Parzen model for each feature channel and for the object and background region of the projected model. The algorithm then tries to optimise the pose in order to separate these PDFs. The three figures show the PDFs generated from the initial pose guess (red lines) and at the end of the pose estimation step (green lines) for the three channels used (i.e. the luminance and the two colour channels in the CIELAB colour system) for the interior (thick lines) and exterior (thin lines) of the puncher shown. It can be seen that the initial pose contains parts of the cyan mouse pad, which has values around (51, -15, -5) in CIELAB colour space, and of the orange measuring tape, whose channels are approximately (55,40,55). This is also clearly visible in the PDFs (thick red lines). After pose tracking, the projected pose extends almost entirely over the white puncher in the image. As a consequence, the peak in the luminance around 53 – as well as those in the colour channels around 35 and 50, respectively – have disappeared (thick green lines). Also note that the difference between the PDFs after pose tracking (green lines) is greater than before pose tracking (red lines). The difference between the PDFs for the outside regions before and after pose tracking is very small because only a small fraction of the outside area has changed. The evolution of the pose is shown in Figure 5.
Figure 6 illustrates how the force vectors affect the pose. Figure 6b depicts the

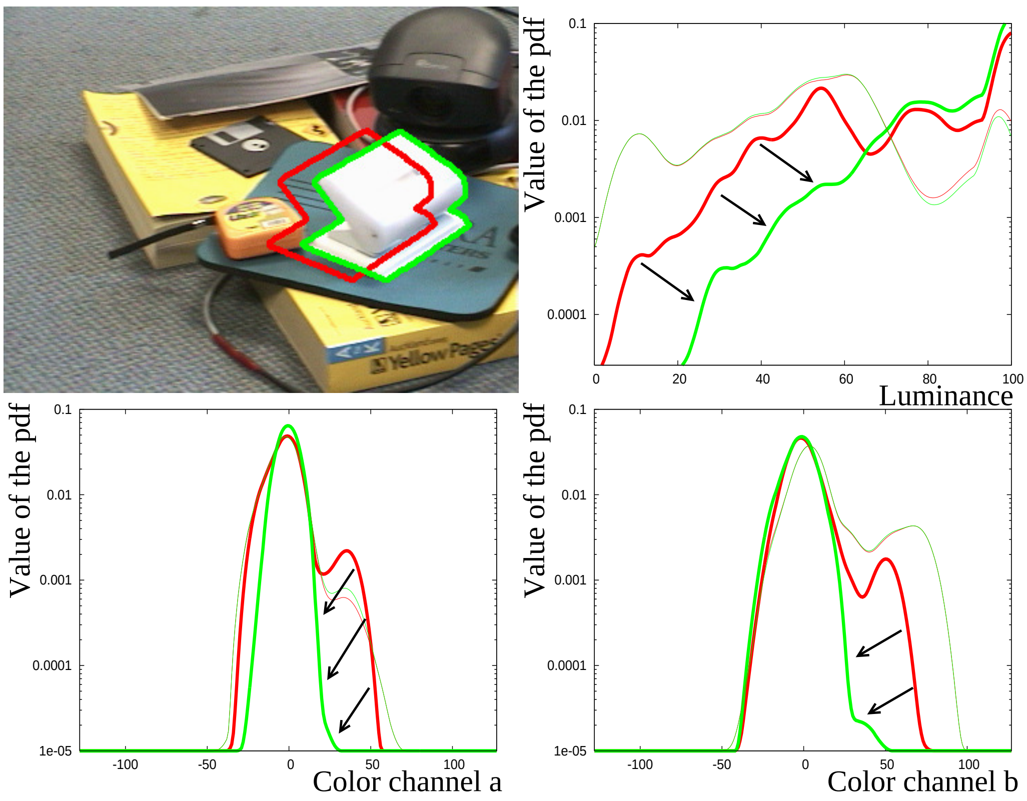

Figure 4: **Upper left corner**: Contour of an initial pose guess (red) and the contour after tracking (green). **Upper right corner**: PDFs generated for the luminance channel. **Lower two figures**: PDFs generated for the colour channels. **Red lines:** PDFs before pose estimation. **Green lines:** PDFs after pose estimation is finished. **Thick lines:** PDFs for the interior of the object. **Thin lines:** PDFs for the exterior of the object. The black arrows indicate how the PDFs evolved. Note that, as explained in the text, the PDFs for the background region changed only marginally.

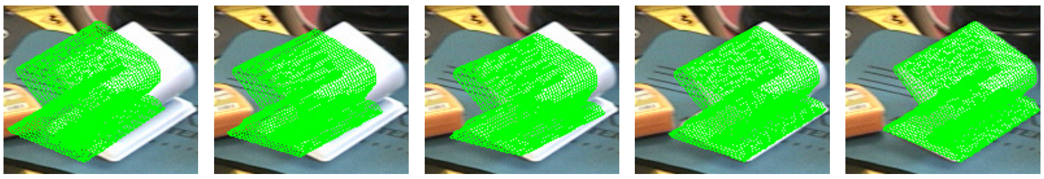

Figure 5: **From left to right:** Initial guess, and pose estimated after 10, 30, 50, and 70 iterations (magnified).

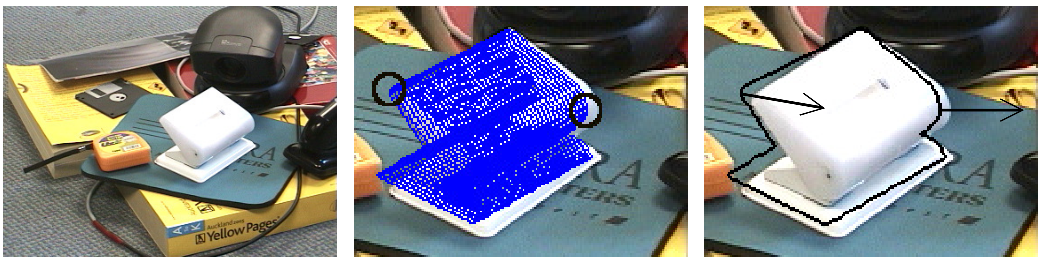

Figure 6: **From left to right:** (a) Input image. The puncher is to be tracked. (b) Projection of the model in an inaccurate pose onto the image (magnified). The two marked points are the points referenced to in Section 3.3. (c) The 2-D contour of the projection (magnified). The arrows show into which directions these points should move in our algorithm. From [14].

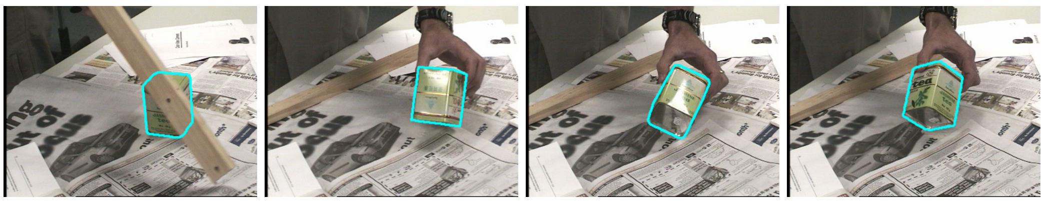

Figure 7: **From left to right**: Tracking results for frames 121, 214, 269 and 274 in one of the two available views of a tea box sequence. Only the silhouettes of the tracking results are shown such that the orientation of the tea box and the specular highlights can be seen.

surface model corresponding to the puncher which has been projected in an inaccurate pose onto the input image (Figure 6a). Figure 6c shows the boundary between the interior and exterior of the projected model. Since most of the interior is white, the white point marked by the circle on the right side of the image fits to the statistical model of the object region better than to that of the background. Thus, it is shifted away from the object, i.e. to the right. The other marked point, which is cyan, better fits the PDF of the background and is thus shifted towards the interior of the object, and thus also roughly to the right.
If the estimated pose is close to the optimal one, the pose changes induced by the force vectors will mutually cancel out. Thus, the process is iterated until the average pose change after up to three iterations is smaller than a threshold T. The optimal threshold depends on the sequence and the parameter l. For example, the farther the object is from the camera, the larger T should be.

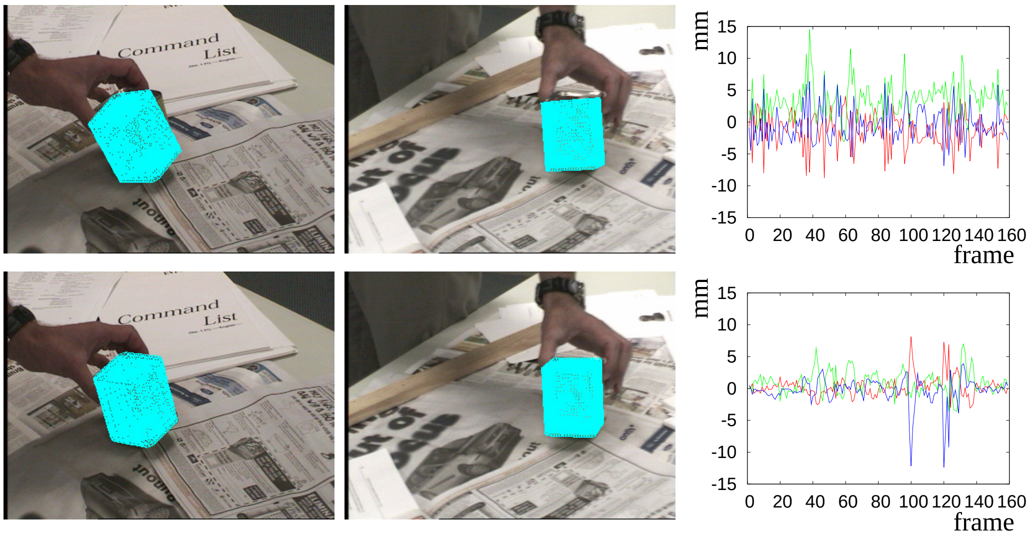

Figure 8: Illustration of the quality/speed tradeoff of the pose tracking algorithm. **From left to right**: Tracking results for frame 220 in both cameras and change of the three translation parameters in the first 160 frames in which the static tea box was partially occluded. **Upper row**: Fast but imprecise results. **Lower row**: Precise but slower results. In addition to being imprecise, the fast results also oscillate stronger.

3.4 Experiments

The experiment in Figure 7 demonstrates the abilities of the basic tracker. A tea box is tracked in a stereo setup. Despite partial occlusions, complete rotations and specular highlights, tracking worked very well. As in all other experiments, we use a given pose in the first frame (usually created by hand) as initialisation.
In Figure 8 we highlight the quality-speed tradeoff by comparing two variants of the tracking algorithm: a fast version where only intensity and colour are considered, the time step size in the gradient descent is larger, and a less restrictive stopping criterion is used (upper row), and a more accurate version where we also incorporate the texture features from [34] (lower row). As can be clearly seen in the diagrams, the version with texture features and finer time steps is much more precise as errors are mostly below 3mm. On the other hand, the result in the upper row was computed nearly an order of magnitude faster, as indicated in Table 2. In many applications the fast result may already be sufficiently precise. The worst tracking error of the fast tracker appeared in frame 220, which is the one depicted in Figure 8.

3.5 Reusing old probability densities

The extrapolated pose used as initial guess for a new frame is often inaccurate. Therefore, the PDFs based on this pose might differ strongly from the PDFs corresponding to the correct object position in the image. This can lead to a slower convergence rate or even to an unpleasant local optimum far from the sought solution.

One way to deal with this problem is to replace the simple extrapolation step by a more sophisticated method. For example, a 2-D pose tracking or an optical flow algorithm can be incorporated to improve the initial guess in a new frame as done in [41].

As an alternative to such rather complex approaches, we propose to diminish the problem of inaccurate PDFs at the beginning of a new frame by reusing the PDF estimates from the previous frame. The inherent assumption is that the appearance of the object and background, or more precisely their description by a PDF, changes only marginally each frame. This assumption is usually well satisfied, though care has to be taken in case of the local Gaussian model. Consequently, we recompute the probability density functions only after the estimation of the object position is completed, i.e. once per frame. Since this requires significantly fewer PDF estimations, it also leads to a speedup of the algorithm.

Keeping the PDFs from the previous frame has another advantage. Since the PDFs p_{in} and p_{out} are constant for every iteration, the derivatives $\nabla(\log p_{\text{in}})$ and $\nabla(\log p_{\text{out}})$ are zero, and Equation (15) is not an approximation anymore.

As mentioned above, the situation is a bit more complex when locally varying densities are employed. If the object (and possibly the background) moves between two frames, the positional information of the local densities from the previous frame are no longer valid. There are two possible approximations. The PDFs for a 2-D–3-D point correspondence (x, X) can be evaluated either: a) at x, or b) at the position to which X was projected in the previous frame. This second alternative is not available in pose tracking algorithms that use explicit segmentation, since there is no real 3-D information incorporated into the segmentation step.

Both approximations have advantages and disadvantages. Method (a) results in better approximations of static backgrounds. However, the background is often not static. Moreover, since it ignores the motion of the object, the approximation of the object model is often better when using method (b). Furthermore, method (b) has slightly higher computational costs and memory requirements, as more information from the previous frame must be stored and evaluated. In practise, however, both methods yield very similar results and the additional costs of method (b) are negligible. Since the results are similar, we used method (a) because it is a bit faster in our implementation.

Figure 9 shows five frames of a monocular sequence with a wooden toy giraffe.

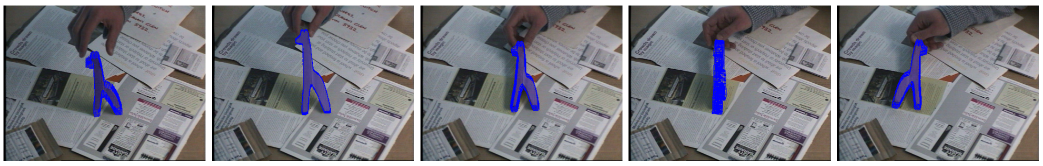

Figure 9: Estimated pose for five frames of a colour sequence with a wooden giraffe. **From left to right:** Frames 32, 48, 64, 74, and 84. Since the surface model consists of a single closed point grid, it is possible to look through the projected pose.

Since only one view is available, and since the appearance of the wooden giraffe is rather complex, tracking is quite hard in this sequence. Still, the projection of the estimated model fits the object in the image well. For this sequence, as well as for all experiments described from now on, we used the probability density functions from the previous frame as described before.
Figure 10 shows an example of a scene with fast movements and strong accelerations. Although only one view was available the white puncher was tracked even though it moved with more than 50 pixels per frame. Moreover, the algorithm was able to deal with accelerations of more than 70 pixels / frame / frame. With the proposed algorithm, 171 frames have been successfully tracked. The best result we achieved without reusing PDFs is a successful tracking of 25 frames, after which tracking failed completely. An example result obtained with a non-constant l is also shown in this figure.
In Figure 11 it is illustrated that the proposed algorithm can also handle movements towards the camera even if only one view is available. Furthermore, there is a second aeroplane visible in the background with an appearance very similar to that of the aeroplane being tracked. Nevertheless, tracking was successful. In the very last frame, in which only a very small part of the aeroplane is visible, tracking results still look good but are in fact quite bad, as can be seen in the last image and the plot in Figure 11.

3.6 Articulated objects

This section explains an extension of the model that allows to handle articulated objects by employing kinematic chains as introduced in [42]. Kinematic chains are a system of rigid bodies connected by joints, whereby each joint has only one degree of freedom. Since every joint angle is an additional parameter that must be estimated, and due to self-occlusions, pose tracking becomes far more challenging. Incorporating kinematic chains, the function P – and thus the energy function (9) – does not only depend on the twist ξ but also on the joint angles in

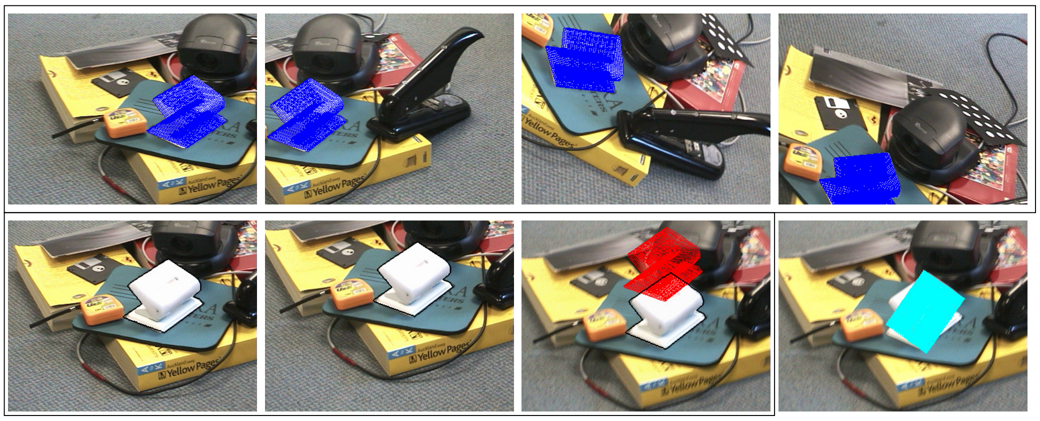

Figure 10: In the upper row, the estimated poses for the frames 10, 16, 134, and 142 of a monocular sequence are shown. The last row shows the estimated contours for the frames 101, 102 and 103 in black, followed by the tracking result in frame 103 when the length of the shift vector l is not set to a fixed value in turquoise. Note the fast movement between frame 10 and 16, the fact that the object partially left the image around frame 142 and the pretty sudden turnaround in the frames 101 to 103. Also note the motion blur visible in frame 103. The red pose shown in the image to frame 103 is the pose estimated by our initial prediction step. Although it is initially far away from the correct pose, our algorithm is able to successfully track the puncher if l is fixed. The position of the puncher estimated by our algorithm is indicated by the black contour. Note that only the camera moves while the scene itself is static.

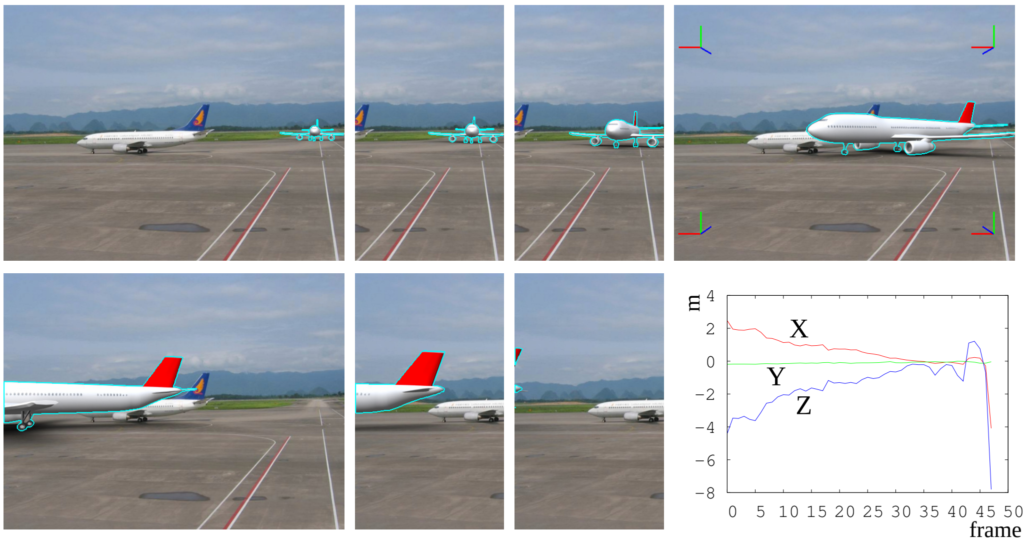

Figure 11: Tracking results for the frames 0, 10, 20, 30, 40, 44, and 47 of a monocular, synthetic scene with a moving aeroplane, illustrated as cyan contours in the original images. Four images are cropped, and some lines illustrating the coordinate system were added in frame 30. The plot shows the translational tracking errors per frame in meter. Although there is another aeroplane with similar appearance in the background, the tracking results look good. According to the quantitative evaluation, there is some inaccuracy due to depth ambiguity, and the last frame was tracked badly due to the fact that most of the aeroplane was not visible. Input sequence generated and kindly provided by Richard Steffen from the University of North Carolina at Chapel Hill.

the chain. Apart from the additional parameters there is no change in the energy. A joint can be modelled by a twist of the form $\theta_i\xi_i$ with unknown joint angle $\theta_i \in \mathbb{R}$ and known joint axis ξ_i (the rotation axis is a part of the model representation, and there is no translation). The complete pose that must be estimated for a model with n joints is given by $\chi = (\xi, \theta_1, \ldots, \theta_n) \in \mathbb{R}^{6+n}$.
The equation of a point X_i behind the j-th joint then has to satisfy

$$\left(\exp(\hat{\xi})\exp(\theta_1\hat{\xi}_1)\cdots\exp(\theta_j\hat{\xi}_j)X_i\right)_{3\times 1} \times n_i - m_i = 0 \qquad (18)$$

in order to lie on the Plücker line $L = (n_i, m_i)$. After all exponential functions are linearised in the same way as in Section 2.2, one such constraint results in three rank two equations in the six pose parameters and the joint angles.
A problem typically arising with kinematic chains is that there might be no 2-D silhouette points for a rigid body in the chain, either because it is completely surrounded by other parts of the object (e.g. a small hand in front of a big torso) or because it is occluded. In both cases, the angle of the corresponding joint cannot be estimated. This problem can be alleviated by including prior knowledge on joint angle configurations; see for instance [43] for further details of this optional extension. However, since a prior is never as good as a reliable measurement, particularly when there is little training data, we rather propose to improve the silhouette model. This will be done in Section 5, where we better exploit the available information provided by the image.
Tracking results with an articulated object model are depicted in Figure 12. For this scene, grey-scale images from four cameras as well as prior information were used. Apart from smaller problems due to the fact that a human cannot be perfectly modelled by a kinematic chain, the only tracking inaccuracies occur at the feet. This is due to the feet being white while the rest of the object is black.

4 Simultaneous Tracking of Multiple Objects

As shown above, the basic pose tracking approach can handle partial occlusions in some situations. However, once the percentage of the occluded area is too big, tracking will fail. This is because a large part of the initial estimate of the object region covers the background and thus, the estimated PDF for that region will be very inaccurate[1]. As a consequence, the estimated PDFs cannot be used to distinguish between interior and exterior anymore.

[1]When reusing the PDFs from the previous frame, this effect is postponed to the successive frame. The problem is not solved, though.

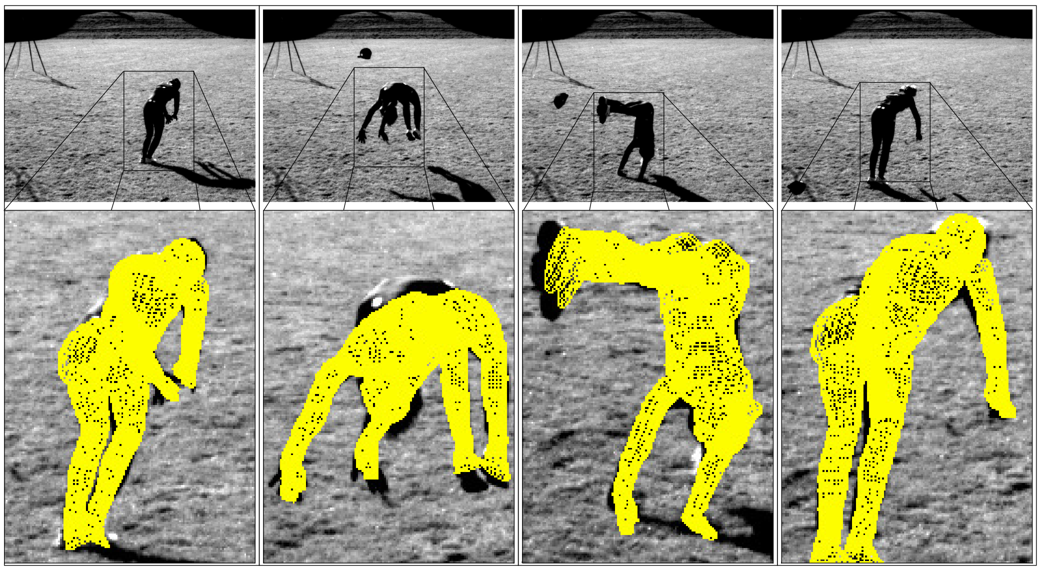

Figure 12: A sequence where a person turns two backflips. Only one out of four camera views is shown. The left image (frame 50) shows the start of the second flip. The two images in the middle show the frames 96 and 148, and in the image on the right (frame 190) the backflip is completed. The bottom row shows the estimated pose in zoomed images.

In this section, we show how to handle occlusions that are due to multiple objects by a coupled tracking of all involved objects. The basic idea was presented in a preliminary conference paper [44].

4.1 Basics of multiple object tracking

In order to track multiple objects, we assume that a model and a pose initialisation is available for each object. The goal is to find the poses of all objects. That is, if N is the number of objects to be tracked, the algorithm should find the pose χ_i of each object i. For the methodology we present here, it is not important if we track rigid or articulated objects, or both. Hence, we will always write χ_i for the pose parameters even though χ_i could be simplified to ξ_i in case of rigid objects.
The simplest way to track multiple objects with the approach introduced in the last section is to estimate PDFs for the interior and exterior of each object, and to minimise an energy function that is the sum of energy functions of the form (9):

$$\begin{aligned} &E(\chi_1,\dots,\chi_N) \\ &= -\sum_{i=1}^{N}\int_{\Omega}\Big(P_{\chi_i,i}(x)\log p_{i,\mathrm{in}} + \big(1-P_{\chi_i,i}(x)\big)\log p_{i,\mathrm{out}}\Big)\mathrm{d}x. \end{aligned} \tag{19}$$

Here, $P_{\chi_i,i}$ is a projection function $\Omega \mapsto \{0,1\}$ which projects the i-th object with pose χ_i to the image plane. Minimising this energy function can be done in basically the same way as with the old energy function: Each object is projected, PDFs are estimated, the 2-D parts of the 2-D–3-D point correspondences are adapted according to the gradient derived from the corresponding PDFs, and a system of equations is solved for each object. As soon as the pose update of one object is small, its position can be considered as final and only the poses of the remaining objects are searched. This approach treats all objects independently. As soon as two objects partially occlude each other, however, the independence assumption is no longer satisfied.

4.2 Occluding objects

In order to understand the problem of occluding objects, consider the illustrative example in Figure 13(a) in which a green puncher and a yellow box is to be tracked. Suppose the 3-D pose of both objects is already correctly estimated. When the green puncher is projected onto the image plane, it splits the image into the background region (blue region in Figure 13(b)) and foreground region (green and yellow region in Figure 13(b)). Clearly, the foreground region not only includes the green region, but also a considerable area of the yellow tea box. As a consequence, the PDF estimated for the interior of the puncher has two modes:

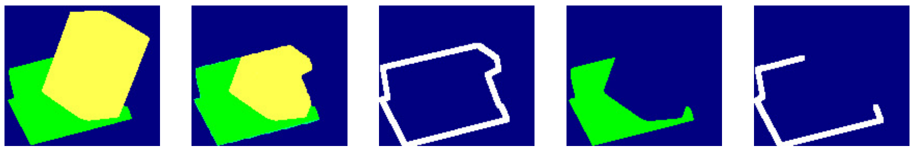

Figure 13: Illustration of two problems occurring when one of the tracked objects occludes an other, and how our algorithm solves them by utilising the visibility functions explained in Section 4: Suppose the initial poses of the two objects to be tracked in the artificial input image (a) are correct. The area (b) used for PDF estimation and silhouette (c) used without visibility functions are suboptimal. When using the visibility functions, the area (d) and silhouette (e) are more reasonable.

one from the green and one from the yellow area. Therefore, yellow parts of the image are falsely believed to belong to the interior of the puncher. Those silhouette points which are yellow (the silhouette is shown in Figure 13(c)) consequently generate gradient vectors that point towards the outside of the object. The tracked puncher will get stuck in the yellow area and tracking fails.

4.3 Coupled tracking

In order to prevent problems due to occlusions, it is necessary to make sure that each image point is only assigned to the visible object. Hence we introduce a visibility function [45] v_i for each object i. Let $O_i(\chi_i, x)$ be the set of those point on the i-th object model with pose χ_i that are projected onto the image point x. Next we define

$$d_i(\chi_i, x) := d(O_i(\chi_i, x), C) = \min_{y \in O_i(\chi_i, x)} \{d(y, C)\} \tag{20}$$

as the minimal distance of the set O_i to the camera origin C. Finally, the visibility functions are given by

$$v_i(\chi_1, \ldots, \chi_n, x) = \begin{cases} 1 & \text{if } d_i(\chi_i, x) = \min\limits_{j \in \{1,\ldots,n\}} \{d_j(\chi_j, x)\}, \\ 0 & \text{else} \end{cases} . \tag{21}$$

That is, a point is considered to be visible if there is no point of another object closer to the camera origin projecting to the same image point. This local visibility testing is clearly superior to approaches which simply assume *all* parts of one object to be in front of *all* parts of another.
The arguments of the visibility functions – the pose of all involved objects and the image point – will be omitted to keep the formulas readable.

In [44], the energy function

$$E(\chi_1,\ldots,\chi_N) \tag{22}$$
$$= -\sum_{i=1}^{N}\int_{\Omega}\Big(v_i P_{\chi_i,i}(x)\log p_{i,\text{in}} + \big(1 - v_i P_{\chi_i,i}(x)\big)\log p_{i,\text{out}}\Big)\mathrm{d}x.$$

was used for tracking multiple objects: When multiplying each projection function $P_{\chi_i,i}$ with the corresponding visibility function v_i, occluded points are not considered as being inside the interior of the object region. In Figure 13(a), only the green part of Figure 13(d) is therefore used to estimate the PDF of the interior region of the green object. Consequently, the PDF is perfectly accurate here.
However, the PDFs estimated for the outside of each object region still overlap and are thus inaccurate. Thus, we propose to use a single background region here instead of one outside region per object.
The necessary functions $v_0(\chi_1,\ldots,\chi_N,x)$ and $P_0(\chi_1,\ldots,\chi_N,x)$ for the background region are given by $v_0(\chi_1,\ldots,\chi_N,x) := \prod_{i=1}^{N}(1-v_i)$ (the background is visible if no other object can be seen) and $P_0(\chi_1,\ldots,\chi_N,x) := 1$ (ignoring visibility, the background covers the whole image), while the energy function is:

$$E(\chi_1,\ldots,\chi_N) = -\sum_{i=0}^{N}\int_{\Omega}\Big(v_i P_{\chi_i,i}(x)\log p_i\Big)\mathrm{d}x. \tag{23}$$

The minimisation is basically the same as in the single object case. For each object, a silhouette is found by projecting the object, and the point correspondences on the silhouette are adapted by comparing the PDFs of the two regions next to the considered silhouette point, i.e. the PDF of the region in which the point lies and the next region in outwards normal direction. The visibility functions can be computed efficiently by projecting the objects with openGL.
In addition to the improved way to estimate the PDFs, we can further benefit from the explicit modelling of occluding objects by considering only true silhouette points and neglecting occluding contours, as illustrated in Figure 13(d) and (e).
It is advantageous to normalise the lengths of the force vectors depending on the number of correspondences c_i available for each object i. Thus, we linearly scale the length l of the force vectors depending on c_i in each iteration step: For a correspondence on object i, the length of the force vector is set to $\frac{c_i}{c_m}$, where $c_m = \max_i c_i$. This is beneficial since, when only few correspondences are available, each force vector has a larger influence on the pose. Thus, the step size should be smaller to prevent oscillations.
Figure 14 shows a scene in which two Lego Duplo® objects were successfully tracked. Despite the cluttered backgrounds and occlusions in both views, both objects were tracked accurately.

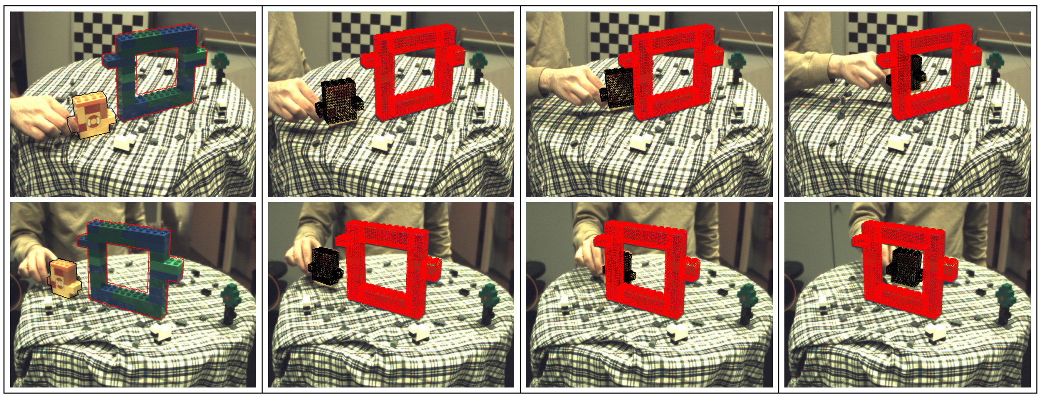

Figure 14: **From left to right:** Four frames (15, 45, 75, and 105) of a sequence with two objects. **Top:** View 1, **Bottom:** View 2. Despite simultaneous occlusions in both frames, the tracking was successful.

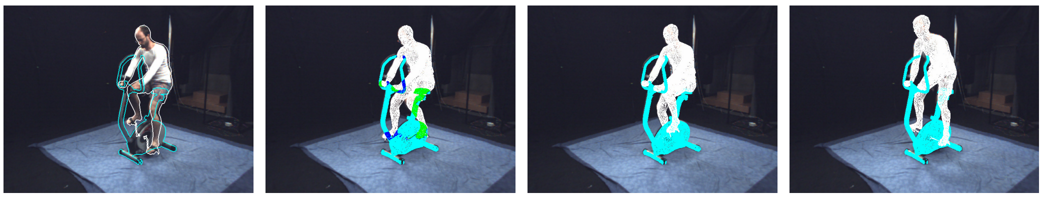

Figure 15: **From left to right:** Four frames (210, 240, 270, and 510) of a sequence with a bike and a cyclist. Only one of the four views is shown. It can be seen that the tracker can handle mutual occlusions. These occlusions are illustrated for frame 240: The parts where the cyclists occludes the bike are shown in green while it is vice versa for those parts shown in blue.

The Figure 15 shows an example in which there are mutual occlusions: The bike partially occludes the cyclist and vice versa in all views throughout the complete sequence. For this sequence, we used the soft constraints from [46] that encourage the hands to stay close to the handlebar and the feet to perform a circular trajectory.

5 Objects With Multiple Components

As long as all objects to be tracked are rigid, the only problematic occlusions occur because one object occludes another, since self-occlusions are usually unproblematic. Once kinematic chains are tracked, there can be self occlusions that result in ambiguities, though. An example for such a situation is shown in Figure 17. For some frames, the right forearm is completely inside the silhouette of the person. As a result, an approach using a single silhouette can only guess the joint angles of the forearm from the prior distribution. Even in case of a learned prior, there remains a lot of uncertainty in the forearm estimate although the arm is clearly visible in the image. We will now present an extension of the tracking approach in order to make better use of the image data in such situations.

5.1 Energy Function using Multiple Internal Regions

We follow the idea discussed in [47], which is to assign different rigid parts of the articulated object to be tracked into one of several components. Then, each component has a separate PDF and is projected separately into the image plane, yielding an increased number of silhouettes for tracking. Note the difference between “part” and “component” used throughout this section: “Part” refers to a rigid part of the kinematic chain, while “component” is a set of one or more parts with similar appearance. For the moment, assume that a good splitting into l different components $M_i, i = 1, \ldots, l$ is already given. We will explain in Section 5.2 how the number and composition of the components can be found.
As in the case of multiple objects, visibility functions should be defined that account for occlusions. Thus, we similarly define $O^j(\chi, x)$ as the set of those points on the j-th component M_j with pose χ that are projected onto the image point x, and

$$d^j(\chi, x) := d(O^j(\chi, x), C) = \min_{y \in O^j(\chi, x)} \{d(y, C)\} \tag{24}$$

as the minimal distance of the set O^j to the camera origin C. The visibility function for the multiple internal regions case are then given by

$$v^j(\chi, x) := 1 \quad \text{if} \quad d^j(\chi, x) = \min_{j \in \{1, \ldots, l\}} \{d^j(\chi, x)\}. \tag{25}$$

Defining the visibility and projection functions for the background as $v_0(\chi,x) := \prod_{j=1}^{l}\left(1-v^j(\chi,x)\right)$ (the background is visible if no other object can be seen) and $P_0(\chi,x) := 1$ (ignoring visibility, the background covers the whole image) leads to the following energy function:

$$E(\chi) = -\sum_{j=0}^{l}\int_{\Omega}\left(v^j(\chi,x)P_{\chi}^{j}(x)\log p^j\right)\mathrm{d}x\,. \tag{26}$$

There is another potential problem that can be solved by this approach: Some parts of the object might look too similar to the background such that including this part degrades the results. For example, this can be the case when tracking someone standing in front of a large white background while wearing a white T-shirt. In this case, white pixels are more likely in the background. Consequently, the tracker tries to reduce the number of white points in the object region, even those of the white T-shirt. This challenge can be easily handled in our approach. All that must be done is to exclude the corresponding parts of the object from all regions M_i. Then it is automatically assigned to the background region. If the removed parts are not at one end of the kinematic chain, this works without any problems since the approach does not require that each part of the object is in one component. Furthermore, it is also possible that a component contains parts that are not connected, e.g. both arms but not the torso.

5.2 Automatic component generation

The tracking approach using multiple internal regions requires that the appearances of the different components can be distinguished. Thus, we assume that parts with similar appearance should be in the same component.
To automatically split an articulated object model, we start by putting each rigid part of the articulated object into a component of its own. If there are significant appearance differences in rigid parts, these parts should also be separated into several components. We have done this manually for the experiment shown in Figure 20. However, it should also be possible to automate this step, e.g. by segmenting the 3-D model based on its appearance visible in the image.
Next, each component is projected into the image plane, and a PDF for each region is estimated. Then, the two components whose PDFs are most similar are combined. After computing the PDF for the new, combined region, these steps are repeated until the difference between all PDFs exceeds a threshold α.
For this algorithm, it is necessary to compare two PDFs. From the several possible ways how to compare PDFs, the Jenson-Shannon divergence [48], which is a symmetric and smooth variant of the Kullback-Leibler divergence [49], works

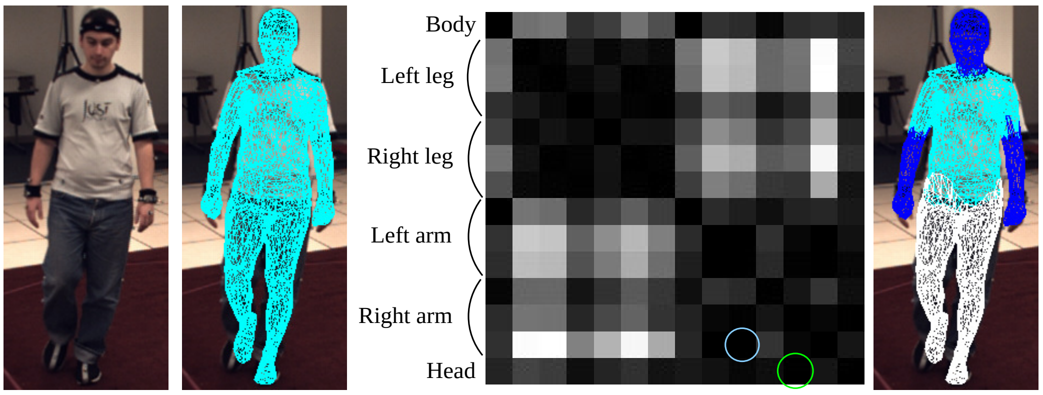

Figure 16: Result of the automatic splitting explained in Section 5.2. **Leftmost:** Input image (cropped).**Left:** Object in initial pose. **Right:** Similarity matrix of the first step. The darker a dot, the more similar two parts are. The circles show the two region pairs which are merged first: Head and lower right arm (green), followed by the two hands (blue). **Rightmost:** Final splitting suggested with splitting threshold in the interval $[0.15, 0.33]$.

best in our experiments. It is given by

$$JSD(p,q) := \frac{J(p,M)+J(q,M)}{2}, \tag{27}$$

where p and q are the PDFs to be compared, M is equal to $\frac{p+q}{2}$, and where J is the Kullback-Leibler divergence

$$J(p,q) := \sum_i p(i)\log\frac{p(i)}{q(i)}. \tag{28}$$

5.3 Experiments

Figure 16 demonstrates the automatic splitting algorithm. Note that upper arms and torso are clustered in the same component, because both have the appearance of the shirt.

In Figure 17 we show tracking results of a monocular scene in which internal regions and the splitting by the automatic splitting algorithm with threshold $\alpha = 0.25$ is used. As can be seen, the tracking works quite well in this monocular sequence, despite the fact that one arm is completely inside the silhouette of the whole object.

Figure 18 shows more results using internal components obtained with $\alpha = 0.25$. The sequence in this experiment is Sequence S4 of the HumanEva-II database [50], for which four cameras are available. The advantage of this benchmark is that it is

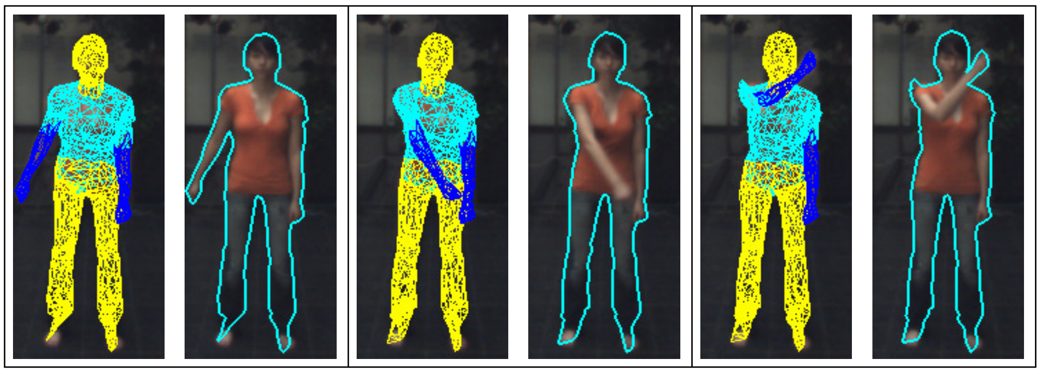

Figure 17: Tracking results for a monocular scene. **From left to right**: Pose and silhouette estimated in frames 10, 38 and 54. Although no information about the right arm is included in the silhouette, the arm is correctly tracked.

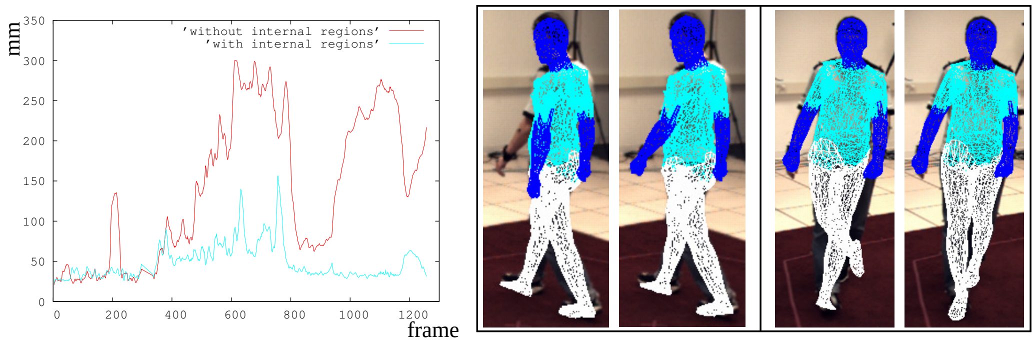

Figure 18: Tracking results for Sequence S4 of HumanEva II. The leftmost image shows the tracking error with and without internal regions. The left image in each block shows the tracking result without using internal regions, the right when using internal regions. The same colours have been used to allow an easy comparison of the results. Without using multiple components, tracking can be wrong due to occlusions (middle images, frame 50), or because the estimated PDFs are inaccurate (rightmost images, frame 200). Only one of the four available views is shown. Images have been cropped.

possible to receive the average tracking error in millimetre per frame via an online interface at Brown university, i.e. a quantitative evaluation is possible. The sequence illustrates that multiple component models can also be useful in multiple camera settings. It is not surprising that the arms are better tracked around frame 50 (see middle images in Figure 18). However, there are also improvements that look surprising at first glance: when using a model with a single component, the two legs crossed around frame 200 (see right images in Figure 18). When using the split model, more accurate PDFs are estimated. As a result, the crossing of legs disappears.
Later in the sequence, tracking without using internal regions fails as indicated by the error plot in Figure 18. Sample frames for the whole sequence, which includes three different types of movement, are shown in Figure 19. The bottom row also shows the frame with the largest tracking error.
Table 1 shows the average tracking error obtained with our method when using a single component, and when using multiple components for the whole sequence, i.e. until frame 1257. It can be seen that the average tracking error is much smaller using the multi-component model. Moreover, these results show that our tracking approach yield very accurate results: In [51], an average tracking error of 80mm is reported for the same sequence. However, only the first 150 frames were used, as tracking became less robust and started failing afterwards. For these frames, we obtain an average error of 32.13mm. Brubaker *et al.* achieve a mean error of 54mm using only two of the four cameras [52]. Their approach only handles the lower body part, as a physical model of walking is used. Thus, only the walking part of the sequence is evaluated (frames 15 to 350). We obtain an average error of 33.4mm in these frames.
Gall *et al.* proposed to perform a global optimisation based on interacting simulated annealing, followed by a local optimisation [53]. Such a global optimisation step prevents tracking failures from propagating into the next frame, resulting in an average error of 32.01mm for the whole sequence. Such an approach can also be incorporated as first step into the tracking approach used here, but would approximately increase the run time of our algorithm by a factor of five.

6 Simultaneous Tracking of Multi-Part Objects

The two methods described in the last two sections, i.e., tracking multiple objects and tracking an object with multiple internal regions, can be combined into a single energy function in a straightforward manner. By minimising this function, multiple object with multiple internal regions can be tracked.

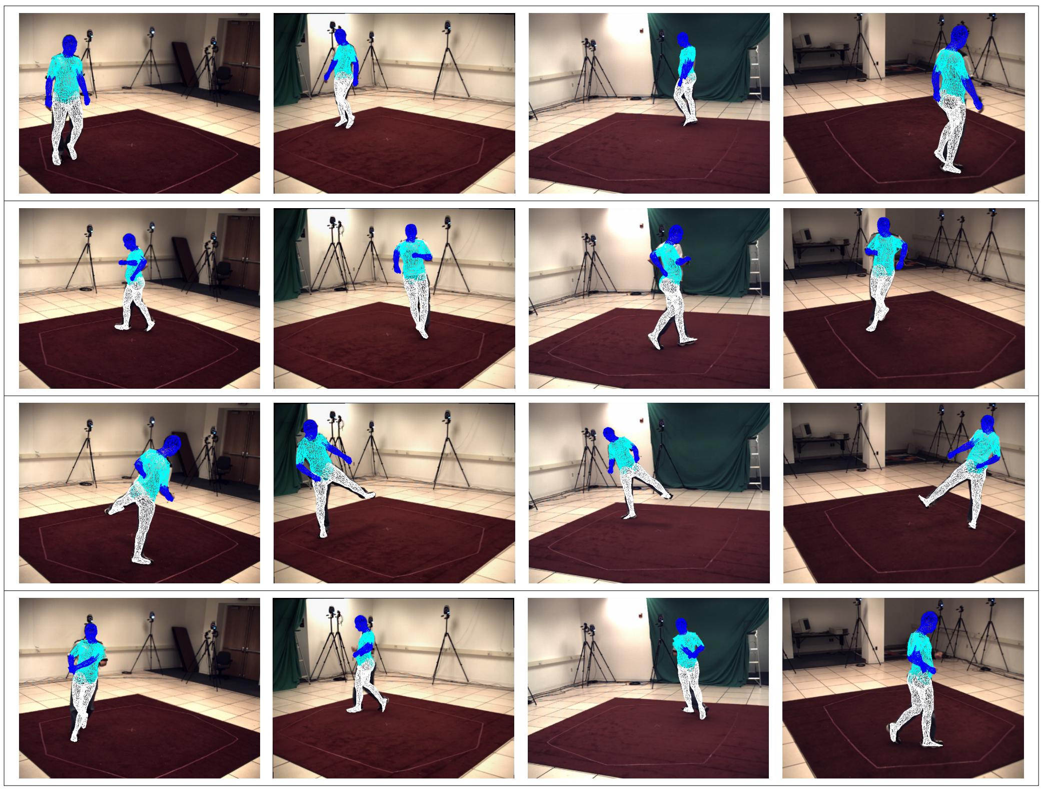

Figure 19: Each row shows tracking results in the four camera views for sequence S4 of the HumanEva-II benchmark. **From top to bottom:** A frame in the walking part (frame 222), a frame in the jogging part (frame 444), a frame in the “balancing” part (frame 888), and the frame with the worst results (frame 757).

Table 1: Some statistics of the tracking error with and without using internal regions for the sequence shown in Figure 18. Given are the average and standard deviation of the tracking error and the maximal tracking error. All numbers are in millimetres.

Model	Avg. error	Std. deviation	Max. error
Multi-part object	48.87	21.94	156.53
Single-part object	142.48	89.82	300.30

6.1 Tracking Multiple Multi-Part Objects

To track n objects M_i ($1 \leq i \leq n$) with unknown poses χ_i, where object i consists of l_i parts $M_i^1, \ldots, M_i^{l_i}$(as explained above), we use functions similar to those used before: We define $P_i^j(\chi_i)$ as projection function of the component M_i^j, $p_i^j := p_i^j(\chi_1, \ldots \chi_n, x)$ as the PDF of M_i^j, and $O_i^j(\chi_i, x)$ as the set of those point on M_i^j that are projected onto the image point x. Furthermore, we set

$$d_i^j(\chi_i, x) := d(O_i^j(\chi_i, x), C) = \min_{y \in O_i^j(\chi, x)} \{d(y, C)\} \tag{29}$$

as the minimal distance of O_i^j to the camera origin C, and

$$v_i^j(\chi_i, x) := 1 \text{ if } d_i^j(\chi, x) = \min_{i \in \{1, \ldots N\}} \min_{j \in \{1, \ldots, l_i\}} \{d_i^j(\chi_i, x)\} \tag{30}$$

as visibility function of the component M_i^j.
Then the function to be minimised reads:

$$E(\chi_1, \ldots \chi_N) = -\sum_{i=0}^{N} \sum_{j=1}^{l_i} \int_\Omega \left[v_i^j P_{\chi_i, i}^j(x) \log p_i^j\right] \mathrm{d}x \tag{31}$$

For the background we set $l_0 = 1$, since it has only a single component. Furthermore, we set $v_0^1(\chi_1, \ldots, \chi_n, q) = \prod_{i=1}^n \prod_{j=1}^{l_i} \left(1 - v_i^j(\chi_1, \ldots, \chi_n, q)\right)$ (again, the background is visible if no other object is visible) and $P_0^1(\chi_1, \ldots, \chi_n, q) = 1$ in this case. Minimisation of this energy function is performed in a similar way as minimising the energy function (26). The only difference is that the poses of all objects are optimised simultaneously, like in (23).

6.2 Experiments

In Figure 20 we track another scene with three Lego Duplo objects that was first used in [44]. Two of those objects are red, and the third consists of a large green

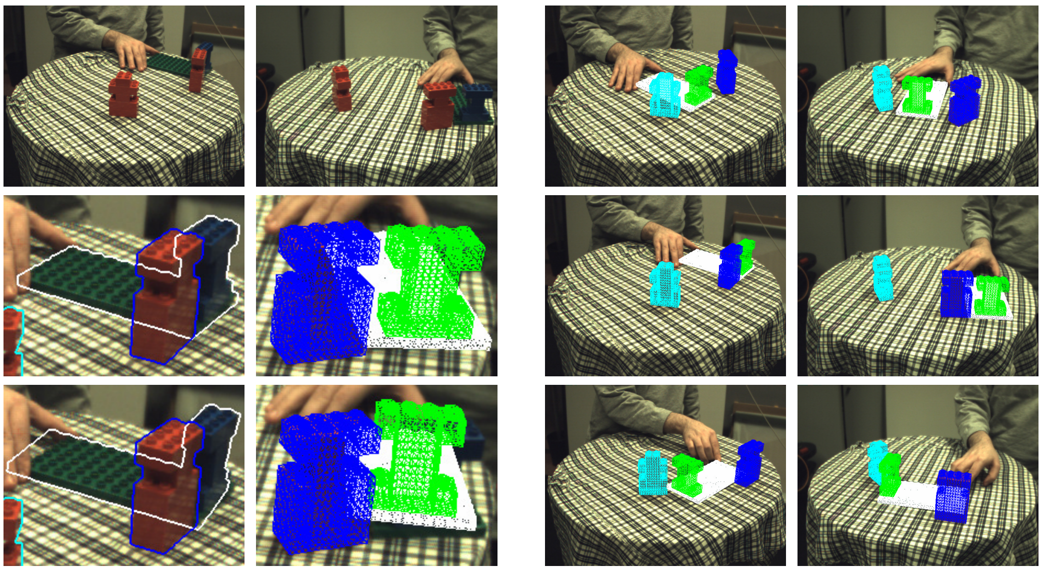

Figure 20: Relevance of using multiple internal regions. **Left two columns, from top to bottom**: The two input images for frame 139, result with internal regions (magnified), and with a single region (magnified) Only the silhouette is shown for one view to make the details better visible. **Right two columns, from top to bottom**: Tracking results in frame 20, 190, and 350 using internal regions.

board with blue Duplos attached to it. We track this multiple-object sequence once without internal regions, and once after splitting the last object into a green and a blue part. Without using internal regions, tracking is quite inaccurate for some frames. Trying to track the blue/green object without tracking the red objects fails completely. These experiments clearly shows that using internal regions is also advantageous when tracking rigid objects.

7 Summary

In this paper, we have presented a region-based pose tracking algorithm that can incorporate various statistical models for object and background region without explicitly estimating a contour in the image. We showed that this is advantageous, since it leads to a significant speedup compared to other region based methods that require costly segmentation and contour matching steps. Furthermore, results are often better than those achieved with alternating segmentation and pose estimation steps. Moreover, we demonstrated that this framework can be extended to kinematic chains in order to track human motion. We also proposed to reuse probability density functions from previous frames. This has led to substantial

Table 2: In this table, several pieces of information are given for each image sequence shown in this paper: the number of views (Views), the degrees of freedom per object (Unknowns), the size of each image (Size), the average number of iterations necessary (Iterations), the average amount of real time (or wall clock time) required per tracked frame (RTime), the average amount of processor time required per tracked frame (PTime), the number of frames tracked (Frames), if information from the textures space proposed in [34] was used (Texture), if probability density functions from the last frame were used (PDFs), and the figure(s) in this article in which results are shown (Figure(s)). All times are from computations on an Intel Pentium 4 with 3.2 GHz. Since the wall clock time is influenced by other running processes and delays due to accessing a hard drive, we also included the processor time unaffected by these problems.

Sequence	Views	Unknowns	Size	Iterations	RTime	PTime	Frames	Texture	PDFs	Figure(s)
Tea box	2	6	384×288	12.7	5.05s	4.75s	395	yes	new	7, 8
Tea box 2	2	6	384×288	4.5	0.79s	0.51s	395	no	new	8
Giraffe	1	6	384×288	280.0	6.94s	6.63s	84	yes	old	9
Puncher	1	6	376×284	50.2	1.04s	0.92s	171	no	old	10
Aeroplane	1	6	640×480	39.4	13.79s	13.43s	48	no	old	11
Flip	4	27	500×380	59.7	19.81s	18.75s	260	no	old	12
Duplo2	2	6/6	640×480	36.7	24.77s	24.22s	110	yes	old	14
Bike	4	6/28	640×480	23.2	42.86s	42.17s	550	no	old	15
Arm	1	30	640×480	155.7	24.09s	23.42s	59	no	old	17
HumanEva	4	28	656×490	18.3	15.48s	14.63s	1257	no	old	18,19
Duplo3	2	6/6/6	640×480	12.8	4.89s	4.27s	380	no	old	20

speedups and increased the robustness of the algorithm in case of fast object motions with high accelerations.
In order to deal with partial occlusions in tracking, we suggested a generative approach, which tracks all objects simultaneously. Thanks to the depth reasoning in this approach, occlusions can be dealt with, which leads to significantly improved performance than when treating occlusions merely as model noise. The same concept can deal with self-occlusions that have been a nuisance in silhouette-based approaches. We also showed a way how to generate reasonable multi-component object models from a single model automatically. The methodology was evaluated in various settings and the experiments revealed a clear improvement in performance due to the occlusion reasoning.
Table 2 summarises all the presented experiments. As can be seen the number of required iterations varies strongly from sequence to sequence: More iterations are needed if the prediction of the object's pose for the next frame is bad. Furthermore, rotating object (parts) require more iterations, especially in monocular setting. Similarly, the computation time per frame also varies strongly. In all experiments except *tea box 2*, we optimised the parameters independently for each sequence to get optimal results. Thus the runtimes are often quite high. This is due to the tradeoff between quality and speed mentioned before. Note that real-time performance was obtained with an algorithm similar to our basic tracker in [54] by using CUDA. Thus, one can expect significant lower runtimes by performing processing on the GPU using NVIDIA's CUDA framework.
Even if partial occlusions are handled, there are still situations in which our approach is likely to fail, e. g. if one (part of an) object can barely be distinguished from the surrounding background, or if the projection of the initial pose guess does not overlap with the correct object region due to strong accelerations. This is the topic of our ongoing research. Similarly, we plan to estimate the parameters of our algorithm (the length of the shift vector l, the threshold T, and possibly even the models underlying the PDF, as well as the image features used) automatically in the future.

Acknowledgements

Our research was partly funded by the Deutsche Forschungsgemeinschaft under project We2605/5-2. This is gratefully acknowledged.

References

[1] D. M. Gavrila. The visual analysis of human movement: a survey. *Computer Vision and Image Understanding*, 73(1):82–98, January 1999.

[2] David A. Forsyth, Okan Arikan, Leslie Ikemoto, James O'Brien, and Deva Ramanan. Computational studies of human motion: part 1, tracking and motion synthesis. *Foundations and Trends in Computer Graphics and Vision*, 1(2–3):77–254, 2005.

[3] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *International Journal of Computer Vision*, 104(2):90–126, November 2006.

[4] Ronald Poppe. Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding*, 108(1-2):4–18, October 2007.

[5] M. A. Abidi and T. Chandra. Pose estimation for camera calibration and landmark tracking. In *Proc. International Conf. Robotics and Automation*, volume 1, pages 420–426, Cincinnati, May 1990.

[6] David G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(2):441–450, May 1991.

[7] J.R. Beveridge. *Local Search Algorithms for Geometric Object Recognition: Optimal Correspondence and Pose*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, May 1993.

[8] D.J. Kriegman, B. Vijayakumar, and J. Ponce. Constraints for recognizing and locating curved 3D objects from monocular image features. In G. Sandini, editor, *Computer Vision – ECCV '92*, volume 588 of *Lecture Notes in Computer Science*, pages 829–833, Berlin, 1992. Springer.

[9] T. Drummond and R. Cipolla. Real-time tracking of multiple articulated structures in multiple views. In D. Vernon, editor, *Computer Vision – ECCV 2000, Part II*, volume 1843 of *Lecture Notes in Computer Science*, pages 20–36, Berlin, 2000. Springer.

[10] M. Pressigout and E. Marchand. Real-time 3d model-based tracking: Combining edge and texture information. In *IEEE Int. Conf. on Robotics and Automation, ICRA'06*, pages 2726–2731, Orlando, Florida, May 2006.

[11] A. Sundaresan and R. Chellappa. Multicamera tracking of articulated human motion using shape and motion cues. *IEEE Transactions on Image Processing*, 18(9):2114–2126, 2009.

[12] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[13] J. Gall, B. Rosenhahn, and H.-P. Seidel. Drift-free tracking of rigid and articulated objects. In *Proc. 2008 IEEE Computer Society Conference of Computer Vision and Pattern Recognition*, New York, NY, USA, June 2008. IEEE Computer Society Press.

[14] C. Schmaltz, B. Rosenhahn, T. Brox, D. Cremers, J. Weickert, L. Wietzke, and G. Sommer. Region-based pose tracking. In J. Martí, J. M. Benedí, A. M. Mendonça, and J. Serrat, editors, *Pattern Recognition and Image Analysis*, volume 4478 of *Lecture Notes in Computer Science*, pages 56–63, Girona, Spain, June 2007. Springer.

[15] A. Agarwal and B. Triggs. Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1), January 2006.

[16] M. Özuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature harvesting for tracking-by-detection. In *Computer Vision – ECCV 2006, Part III*, volume 3953 of *Lecture Notes in Computer Science*, pages 592–605. Springer, Graz, Austria, 2006.

[17] Min Sun, Hao Su, Silvio Savarese, and Li Fei-Fei. A multi-view probabilistic model for 3D object classes. In *Proc. 2009 IEEE Computer Society Conference of Computer Vision and Pattern Recognition (CVPR)*, pages 1247–1254, New York, NY, USA, June 2009. IEEE Computer Society Press.

[18] A. Ottlik and H.-H. Nagel. Initialization of model-based vehicle tracking in video sequences of inner-city intersections. *International Journal of Computer Vision*, 80(2):211–225, 2008.

[19] Deva Ramanan, David A. Forsyth, and Andrew Zisserman. Tracking people by learning their appearance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):65–81, January 2007.

[20] B. Rosenhahn, T. Brox, and J. Weickert. Three-dimensional shape knowledge for joint image segmentation and pose tracking. *International Journal of Computer Vision*, 73(3):243–262, July 2007.

[21] B. Rosenhahn and G. Sommer. Adaptive pose estimation for different corresponding entities. In L. Van Gool, editor, *Pattern Recognition*, volume 2449 of *Lecture Notes in Computer Science*, pages 265–273, Berlin, September 2004. Springer.

[22] Alexandru O. Bălan and Michael J. Black. The naked truth: Estimating body shape under clothing. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008, Part II*, volume 5303 of *Lecture Notes in Computer Science*, pages 15–29. Springer, 2008.

[23] Romeil Sandhu, Samuel Dambreville, Anthony Yezzi, and Allen Tannenbaum. Non-rigid 2D-3D pose estimation and 2D image segmentation. In *Proc. 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 786–793, Washington, DC, USA, June 2009. IEEE Computer Society Press.

[24] Yan Huang and Irfan Essa. Tracking multiple objects through occlusions. In *Proc. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1051–1058, New York, 2005. IEEE Computer Society Press.

[25] N. Joshi, S. Avidan, W. Matusik, and D.J. Kriegman. Synthetic aperture tracking: Tracking through occlusions. In *Proc. Eleventh International Conference on Computer Vision*. IEEE Computer Society Press, October 2007.

[26] K. Kim and L.S. Davis. Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision – ECCV 2006, Part II*, volume 3953 of *Lecture Notes in Computer Science*, pages 98–109, Berlin, May 2006. Springer.

[27] V. Ablavsky, A. Thangali, and S. Sclaroff. Layered graphical models for tracking partially-occluded objects. In *Proc. 2008 IEEE Computer Society Conference of Computer Vision and Pattern Recognition (CVPR)*, New York, NY, USA, June 2008. IEEE Computer Society Press.

[28] H. Grabner, J. G. Matas, L.J. Van Gool, and P. Cattin. Tracking the invisible: Learning where the object might be. In *Proc. 2010 IEEE Computer Society Conference of Computer Vision and Pattern Recognition (CVPR)*, pages 1285–1292, New York, 2010. IEEE Computer Society Press.

[29] D. Ormoneit, H. Sidenbladh, M. J. Black, and T. Hastie. Learning and tracking cyclic human motion. In Todd K. Leen, Thomas G. Dietterich, and

Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 894–900. The MIT Press, 2001.

[30] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, 1994.

[31] F. Shevlin. Analysis of orientation problems using Plucker lines. In *Proc. 14th International Conference on Pattern Recognition*, volume 1, pages 685–689, Washington, DC, USA, August 1998. IEEE Computer Society Press.

[32] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer, 1980.

[33] ISO/CIE. CIE colorimetry – part 4: 1976 L*a*b* colour space. ISO 11664-4:2008(E)/CIE S 014-4/E:2007, 2007.

[34] T. Brox and J. Weickert. A TV flow based local scale estimate and its application to texture discrimination. *Journal of Visual Communication and Image Representation*, 17(5):1053–1073, October 2006.

[35] B. Mory, R. Ardon, and J.P. Thiran. Variational segmentation using fuzzy region competition and local non-parametric probability density functions. In *Proc. Eleventh International Conference on Computer Vision*. IEEE Computer Society Press, October 2007.

[36] S. Lankton and A. Tannenbaum. Localizing region-based active contours. *IEEE Transactions on Image Processing*, 17(11):2029–2039, November 2008.

[37] G. Aubert, M. Marlaud, O. Faugeras, and S. Jehan-Besson. Image segmentation using active contours: calculus of variations or shape gradients? *SIAM Journal on Applied Mathematics*, 63(6):2128–2154, 2003.

[38] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison–Wesley, Reading, second edition, 2002.

[39] Samuel Dambreville, Romeil Sandhu, Anthony Yezzi, and Allen Tannenbaum. Robust 3D pose estimation and efficient 2D region-based segmentation from a 3D shape prior. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008, Part II*, volume 5303 of *Lecture Notes in Computer Science*, pages 169–182. Springer, 2008.

[40] Marek Vondrak, Leonid Sigal, and Odest Chadwicke Jenkins. Physical simulation for probabilistic motion tracking. In *Proc. 2008 IEEE Computer*

Society Conference of Computer Vision and Pattern Recognition (CVPR), New York, NY, USA, June 2008. IEEE Computer Society Press.

[41] T. Brox, B. Rosenhahn, D. Cremers, and H.-P. Seidel. High accuracy optical flow serves 3-D pose tracking: exploiting contour and flow based constraints. In A. Leonardis, H. Bischof, and A. Pinz, editors, *European Conference on Computer Vision (ECCV)*, volume 3952 of *Lecture Notes in Computer Science*, pages 98–111, Graz, Austria, May 2006. Springer.

[42] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proc. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8–15, Santa Barbara, CA, 1998.

[43] T. Brox, B. Rosenhahn, U. Kersting, and Daniel Cremers. Nonparametric density estimation for human pose tracking. In K. Franke, K. Müller, B. Nickolay, and R. Schäfer, editors, *Pattern Recognition*, volume 4174 of *Lecture Notes in Computer Science*, pages 546–555, Berlin, 2006. Springer.

[44] C. Schmaltz, B. Rosenhahn, T. Brox, J. Weickert, D. Cremers, L. Wietzke, and G. Sommer. Occlusion modeling by tracking multiple objects. In F. Hambrecht, C. Schnörr, and B. Jähne, editors, *Pattern Recognition*, volume 4713 of *Lecture Notes in Computer Science*, pages 173–183, Berlin, September 2007. Springer.

[45] Erik B. Sudderth, Michael I. Mandel, William T. Freeman, and Alan S. Willsky. Distributed occlusion reasoning for tracking with nonparametric belief propagation. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*, 17, pages 1369–1376, Cambridge, MA, 2005. MIT Press.

[46] B. Rosenhahn, C. Schmaltz, T. Brox, J. Weickert, D. Cremers, and H.-P. Seidel. Markerless motion capture of man-machine interaction. In *Proc. 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, June 2008. IEEE Computer Society Press.

[47] C. Schmaltz, B. Rosenhahn, T. Brox, J. Weickert, L. Wietzke, and G. Sommer. Dealing with self-occlusion in region based motion capture by means of internal regions. In F. J. Perales and R. B. Fisher, editors, *Articulated Motion and Deformable Objects*, volume 5098 of *Lecture Notes in Computer Science*, pages 102–111, Berlin, Heidelberg, July 2008. Springer.

[48] A. K. C. Wong and M. You. Entropy and distance of random graphs with application of structural pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(5):599–609, May 1985.

[49] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[50] L. Sigal, A.O. Balan, and M.J. Black. HUMANEVA: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87(1/2):4–27, March 2010.

[51] S. Corazza, L. Mündermann, E. Gambaretto, G. Ferrigno, and T. P. Andriacchi. Markerless motion capture through visual hull, articulated ICP and subject specific model generation. *International Journal of Computer Vision*, 87(1/2):156–169, March 2010.

[52] M. A. Brubaker, D. J. Fleet, and A. Hertzmann. Physics-based person tracking using the anthropomorphic walker. *International Journal of Computer Vision*, 87(1/2):140–155, March 2010.

[53] J. Gall, B. Rosenhahn, T. Brox, and H.-P. Seidel. Optimization and filtering for human motion capture. *International Journal of Computer Vision*, 87(1/2):75–92, March 2010.

[54] V. A. Prisacariu and I. D. Reid. PWP3D: Real-time segmentation and tracking of 3D objects. In *Proceedings of the 20th British Machine Vision Conference*, September 2009.