Universität des Saarlandes

Fachrichtung 6.1 – Mathematik

Preprint Nr. 196

Region-based Pose Tracking

Christian Schmaltz, Bodo Rosenhahn, Thomas Brox, Daniel Cremers, Joachim Weickert, Lennart Wietzke and Gerald Sommer

Saarbrücken 2007

Fachrichtung 6.1 – Mathematik
Universität des Saarlandes

Preprint No. 196
submitted: July 11, 2007

Region-based Pose Tracking

Christian Schmaltz
Saarland University
Department of Mathematics and Computer Science
66041 Saarbrücken, Germany
schmaltz@mia.uni-saarland.de

Bodo Rosenhahn
Max-Planck Institute for Informatics
66123 Saarbrücken, Germany
rosenhahn@mpi-sb.mpg.de

Thomas Brox
Department of Computer Science
University of Bonn, 53117 Bonn, Germany
brox@cs.uni-bonn.de

Daniel Cremers
Department of Computer Science
University of Bonn, 53117 Bonn, Germany
dcremers@cs.uni-bonn.de

Joachim Weickert
Saarland University
Department of Mathematics and Computer Science
66041 Saarbrücken, Germany
weickert@mia.uni-saarland.de

Lennart Wietzke
Institute of Computer Science
Christian-Albrecht-University, 24098 Kiel, Germany
lw@ks.informatik.uni-kiel.de

Gerald Sommer
Institute of Computer Science
Christian-Albrecht-University, 24098 Kiel, Germany
gs@ks.informatik.uni-kiel.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/

Abstract

In this article, we present a technique for region-based pose tracking that uses a surface model of the object to be tracked and at least one calibrated camera view. The proposed algorithm can follow the movement of rigid objects or kinematic chains without explicitly computing contours in the image. The aim to match the model surface to the contour of the object seen in the image is achieved by optimizing an energy with respect to the pose parameters. We explain how kinematic chains can be included, and how using knowledge from the previous frame can help to track objects that move with a high speed or accelerate strongly. We present experimental results for complex scenes observed with one, two, or four cameras.

Contents

1 Introduction

This article deals with 2-D–3-D *pose tracking*, which is the task to pursuit the 3-D position and orientation of a known 3-D object model from a 2-D image data stream [1]. Pose tracking is important in several applications, e.g. self localization and object grasping in robotics, or camera calibration. Although more than 25 years have passed since the initial work of Lowe [2], pose tracking is still a challenging problem. This is particularly true 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 is not discussed.

A lot of different approaches for pose tracking have been considered [3]. A common idea is to use feature matching for tracking. Such features range from points [4] over lines [5] to more complex features such as vertices, t-junctions, cusps, three-tangent junctions, limb and edge injections, and curvature L-junctions [6]. Drummond and Cipolla used edge-detection to achieve real-time tracking of articulated objects with an iterative algorithm [7]. In [8], Pressigout and Marchand combined two tracking algorithms that use edges and texture information, respectively, to achieve improved results.

In [9], Agarwal and Triggs describe learning-based methods that use regression for human pose tracking. Other learning-based approaches have been introduced by Taycher et al. [10], in which an undirected conditional random field is used, and by Sminchisescu et al. [11], in which generative models and recognition methods are combined. Moreover, methods that use neural networks [12] 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. Since the segmentation results have to fit to the model to be tracked, it makes sense to use prior knowledge about the shape of the region to be segmented, either by incorporating 2-D [13] or full 3-D information [14]. 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 iterative approaches in level sets [14] or graph cuts [15]. Although the coupled estimation of contours and pose parameters is beneficial compared to the uncoupled case, segmentation results are not always accurate, as shown in Figure 1.

The basic idea to our region based pose tracking algorithm has been in-

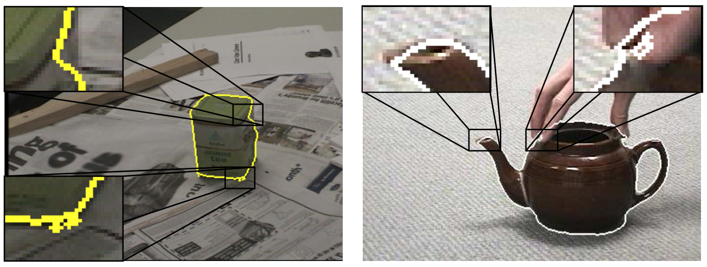


Figure 1: Examples for problems that can occur in variational segmentation algorithms: **Left**: A part of the segmentation is concave although the object to be tracked is convex, and an undesired split up into several connected components. **Right**: The handle is not accurately segmented due to over-smoothing. Furthermore, holes generated in the segmentation are not always correct. From [16].

troduced in a recent conference paper [16]. For the algorithm explained in this article, we utilize the statistical representation of regions described in [14]. We also use the point based pose estimation algorithm from that article. However, instead of separately estimating 2-D segmentation and 3-D pose parameters we directly estimate 3-D pose parameters by minimizing the projection error in the respective 2-D images. Thus, the segmentations obtained with our algorithm are by construction consistent with the 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 [14]. As a consequence, the proposed algorithm achieves nearly real-time performance for some scenes.
In addition to a more detailed description of the algorithm, the present paper extends [16] in several ways without loosing the advantages explained before. For example, the algorithm is extended from rigid objects to kinematic chains. Furthermore, it can now handle much faster accelerations due to the fact that we reuse knowledge from the previous frame.
Our paper is organized as follows: In the following section, we briefly review the basics of pose estimation from 2-D–3-D point correspondences. After that, our basic approach is described in Section 3, followed by several of extensions of the basic algorithm in Section 4. Experimental results are presented in Section 5. Section 6 concludes with a summary.

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

In order to understand the region-based pose tracking approach explained in Section 3, it is helpful to understand the point-based pose tracking approach introduced in [14]. Thus, we will explain this approach after introducing the necessary mathematical concepts.

2.1 Rigid Motion and Twists

An isomorphism that preserves orientation as well as distances is called rigid motion. Such isomorphisms are of interest to us, since a rigid body can only perform a rigid motion. Using a translation vector $t \in \mathbb{R}^3$ and a rotation matrix $R \in SO(3)$ with $SO(3) := \{R \in \mathbb{R}^{3\times3} : \det(R) = 1\}$, any rigid body motion in 3-D can be represented as $m(x) := Rx + t$. When using homogeneous coordinates, it is also possible to write m as a 4×4 matrix M:

$$\begin{aligned} m((x_1, x_2, x_3)^T) =& M(x_1, x_2, x_3, 1)^T \\ =& \begin{pmatrix} R_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1 \end{pmatrix} x \ . \end{aligned} \tag{1}$$

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 $so(3) := \{A \in \mathbb{R}^{3\times3} | A^T = -A\}$, 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. Of course, a rigid motion has always six degrees of freedom independent of its representation.
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

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

$$\hat{\xi} = \begin{pmatrix} \hat{\omega} & \nu \\ 0_{3\times1} & 0 \end{pmatrix} \qquad \in se(3) \ , \tag{3}$$

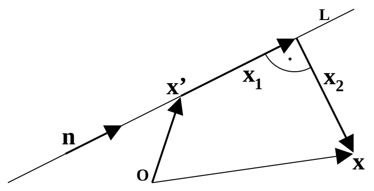


Figure 2: Identifiers used in the proof that the distance between a point x and a Plücker line $L = (n, m)$ is given by $\|x \times n - m\|$.

we distinguish these two ways of representing elements by a hat sign. Thus, the matrix ξ and the vector $\hat{\xi}$ are always two different representations of the same element. Note that a twist ξ can be converted to an element of the Lie group $M \in SE(3)$ by the exponential function $\exp(\hat{\xi}) = M$. Computing the exponential of a matrix is often difficult and time consuming. However, the exponential of twists can be computed efficiently with the Rodriguez formula. More details can be found in [17].

2.2 Plücker forms

As explained before we use lines passing through image points and through the optical center of a camera. Such 3-D lines can be represented in different ways. Here, we will use the Plücker form [18] to represent lines.
A line in Plücker form $L = (n, m)$ is given by a normalized 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 a line $L = (n, m)$ given in Plücker form 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). Then, we have

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

This equation also shows that $x \in L$ is equivalent to $\|x \times n - m\| = 0$. By repeating this calculation with $x_2 = 0$ and without taking the absolute values, we see that the momentum is independent of the choice of $x' \in L$.

2.3 Pose Estimation with 2-D–3-D Point Correspondences

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 an image. Furthermore, let $L = (n, m)$ be the Plücker line through q and the respective camera origin. In [14], such point correspondences are obtained using segmentation followed by a matching step. The details are omitted here since we will use a completely different method to obtain point correspondences, as will be explained in Section 3.
In order to track the object, we need to find a twist ξ that maps the points $\exp(\hat{\xi})x_i$ as close as possible to the corresponding Plücker lines L_i:

$$\arg\min_{\xi} \sum_i \left\| \left(\exp(\hat{\xi})x_i\right)_{3\times 1} \times n_i - m \right\| \tag{5}$$

where the function $\cdot_{3\times 1} : \mathbb{R}^4 \mapsto \mathbb{R}^3$ 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". Thus, we can linearize 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

$$\arg\min_{\xi} \sum_i \left\| \left(\exp\left(\hat{\xi}\right) x_i\right)_{3\times 1} \times n_i - m_i \right\|^2 \tag{7}$$

$$\approx \arg\min_{\xi} \sum_i \left\| \left(\left(I + \hat{\xi}\right) x_i\right)_{3\times 1} \times n_i - m_i \right\|^2 \quad . \tag{8}$$

By evaluating the cross product, we get 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 in the twist. However, there are typically several hundred point correspondences. Thus, one obtains a least squares problem, which can be solved efficiently with standard methods, e.g. the Householder algorithm [19]. In order to further minimize the error introduced by the Taylor expansion, we iterate this minimization process.

3 Region-based Model Fitting

Previous approaches incorporating point based pose estimation [14] usually try to match a contour obtained by segmentation to the projected model surface. Since segmentation and matching can be time consuming and inaccurate, our idea is to avoid both the explicit contour computation as well as the matching step. Instead, we try to find 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 will 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 background and a foreground region, we minimize the energy function

$$\begin{aligned} E(\xi) = \quad & -\textstyle\int_\Omega \big(P(\xi,q)\log p_1 \\ & +(1-P(\xi,q))\log p_2\big)dq \ , \end{aligned} \tag{9}$$

where the function $P(\xi,q) \in (\mathbb{R}^6 \times \Omega \mapsto \{0,1\})$ 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. The functions p_1 and p_2 are two probability density functions (abbreviated as "pdf" from now on) that model the different feature distributions. Such pdfs also occur in variational segmentation methods [14], 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, we only have six parameters that must be optimized. This simplifies the estimation considerably. For instance, the length constraint present in variational segmentation methods can be omitted here.
In order to model the image features by pdfs, we have to decide which features should be modeled. For the experiments presented later, we used the color in CIELAB color space (for color images) or the image intensity (for grayscale images). Additionally, the texture feature space proposed in [20] was used for objects with a complex appearance.
Since the two pdfs p_1 and p_2 are unknown, we must assume an underlying model to estimate them. We track objects with a uniform appearance with a non-parametric Parzen model and objects with a nonuniform appearance with a local Gaussian distribution [14]. However, since there is not enough data available to accurately estimate a multi-dimensional pdf, we consider the separate feature channels to be independent. Then, the total probability

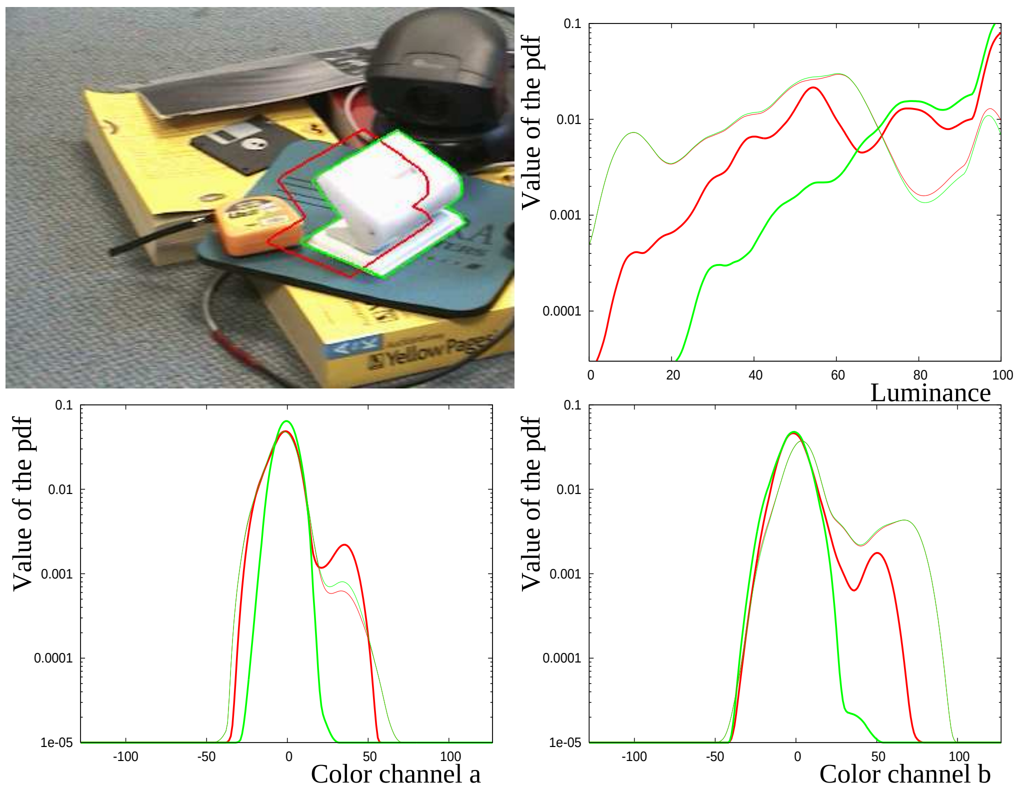


Figure 3: **Upper left corner**: Contour of an initial pose guess. **Upper right corner**: Pdfs generated for the luminance channel. **Lower two figures**: Pdfs generated for the color channels. **Red lines:** Pdfs before pose estimation. **Green lines:** Pdfs after pose estimation. **Thick lines:** Pdfs for the interior of the object. **Thin lines:** Pdfs for the exterior of the object.

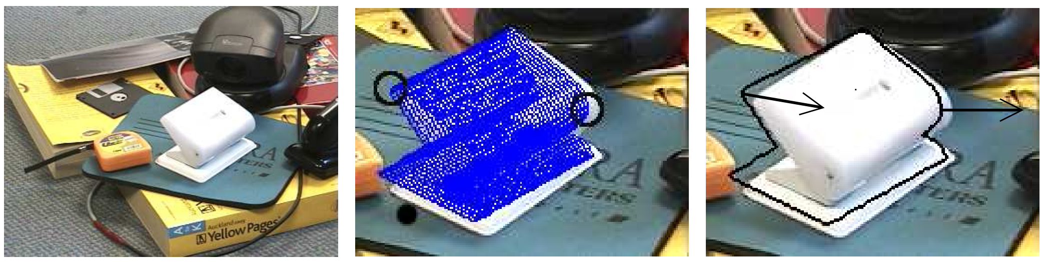
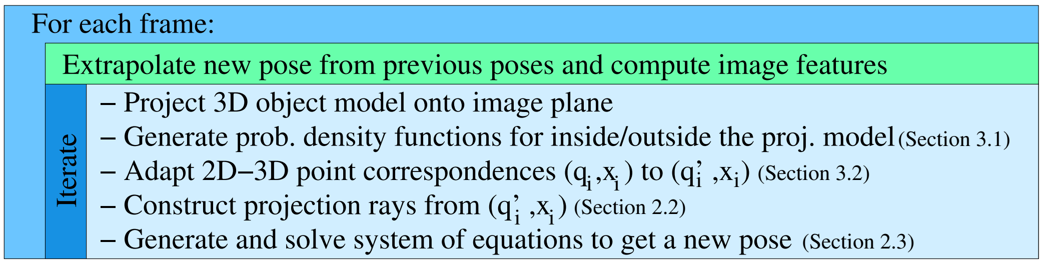


Figure 4: **From left to right:** (a) Input image. The puncher is to be tracked. (b) Projection of the model in an inaccurate pose onto the image (magnified). The two marked points are the points referenced to in Section 3.2 (c) The 2-D contour of the projection (magnified). The arrows show into which directions these points should move in our algorithm. From [16].

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 must be recomputed.
As an example, consider Figure 3. The image shown is this figure shows the contour of the pose estimated by our prediction step (see below). Here, pdfs are generated using the Parzen model for each feature channel and for the inside and outside region of the projected model. Then, our algorithm tries to find a pose which separates these pdfs. The three figures show these pdfs generated from the initial pose guess shown (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 color channels in the CIELAB color system) for the interior (thick lines) and exterior (thin lines) of the puncher shown. All pdfs have been smoothed with a box filter with width 11. It can be seen that the initial pose contains parts of the cyan mouse pad, which has values around (51, -15, -5) in CIELAB color 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 color 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 part of the outside area has changed.


For each frame:

Extrapolate new pose from previous poses and compute image features

Iterate

– Project 3D object model onto image plane

– Generate prob. density functions for inside/outside the proj. model (Section 3.1)

– Adapt 2D–3D point correspondences (q_i,x_i) to (q_i',x_i) (Section 3.2)

– Construct projection rays from (q_i',x_i) (Section 2.2)

– Generate and solve system of equations to get a new pose (Section 2.3)

Figure 5: Steps done by the proposed pose tracking algorithm.

3.2 Minimization

In order to minimize (9), we assume the function P to be differentiable and compute the gradient of E

$$\begin{aligned}\nabla E(\xi) &= -\int_\Omega (\nabla P(\xi,q)\log p_1 - \nabla P(\xi,q)\log p_2)dq \\ &= -\int_\Omega (\nabla P(\xi,q)(\log p_1 - \log p_2))\, dq\ . \end{aligned} \tag{10}$$

Thus, the energy function (9) 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. Thus, the gradient vectors created in this way are reprojected back to the 3-D space in order to estimate the movement. Therefore, we estimate the rigid body motion necessary to change the 2-D silhouette such 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. If the pdf for the inside region evaluated at q_i, i.e. $p_1(q_i)$, is exceeds the corresponding function value for the outside region $(p_2(q_i))$, we suppose that q_i belongs to the object region. Thus, q_i will be moved in outward normal direction to a new point q_i'. On the other side, points where $p_1(q_i) < p_2(q_i)$ holds will be shifted into the opposite direction. We use Sobel operators [21] to approximate the normal direction ∇P.
Although the gradient (10) suggest that the shift vector should have a length of $l := \|\log p_1 - \log p_2\|$, we noticed that setting l to a fixed value often yields better results. We believe that this is the case because the changes induced when using $l = \|\log p_1 - \log p_2\|$ are much smaller when the pose is already close to optimal. This explains why either the induced pose change is too

small, in which case the iteration steps are stopped because the algorithm assumes that it converged, or the correct changes induced are overruled by erroneous changes due to noise or model inaccuracies. More advanced methods for choosing l have been tested, but results were usually worse according to our experiments.
In order to better understand how these force vectors are computed, consider Figure 4. In Figure 4b, one can see the surface model corresponding to the puncher which has been projected in an inaccurate pose onto the input image (Figure 4a). Figure 4c depicts 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 approximately 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. The optimal threshold depends on the sequence and the parameter l. It is not adapted during one sequence at the moment.
Before changing frames in an image sequence, we predict the object's pose in the new frame by linearly extrapolating the results from the previous two frames. This also eases the possible problem introduced by linearizing the exponential function. However, even if the initial pose estimated for a frame is quite bad, the algorithm still has good chances to find the correct pose, as we will show in the experimental section. Figure 5 shows an overview of the algorithm.

4 Extensions

Now, we will introduce extensions that enable the algorithm to track more complex object models in more complicated scenes, e.g. in case of erratic object movements.

4.1 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

Table 1: Overview of the experiments presented.

Name	Views	Color	Size	Iterations	RTime	PTime	Frames	Texture	pdfs reused	Angles	Prior	Figure
Tea box	2	yes	384×288	12.0	4.74s	4.28s	395	yes	no	0	no	6, 7
Tea box fast	2	yes	384×288	3.59	1.09s	0.57s	395	yes	no	0	no	-
Giraffe	1	yes	384×288	280	6.94s	6.63s	84	yes	yes	0	no	8
Teapot	2	yes	376×284	3.56	1.03s	0.47s	347	no	yes	0	no	9
Puncher	1	yes	376×284	481	1.37s	1.15s	171	no	yes	0	no	10
Cart	4	no	500×380	17.1	4.24s	3.68s	464	no	yes	21	yes	11
Flip	4	no	500×380	59.7	19.8s	18.7s	260	no	yes	21	yes	12

In this table, several pieces of information are given for each image sequence used: the number of views (Views), if the images where color images (Color), the size of each image (Size), the average number of iterations necessary (Iterations), the average amount of real 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 [20] was used (Texture), if probability density functions from the last frame were used (pdfs reused), the number of joint angles in the object model (Angles), if prior information was available (Prior) and the figure(s) in this article in which results are shown (Figure). All times are from computations on an Intel Pentium 4 with 3.2 GHz.

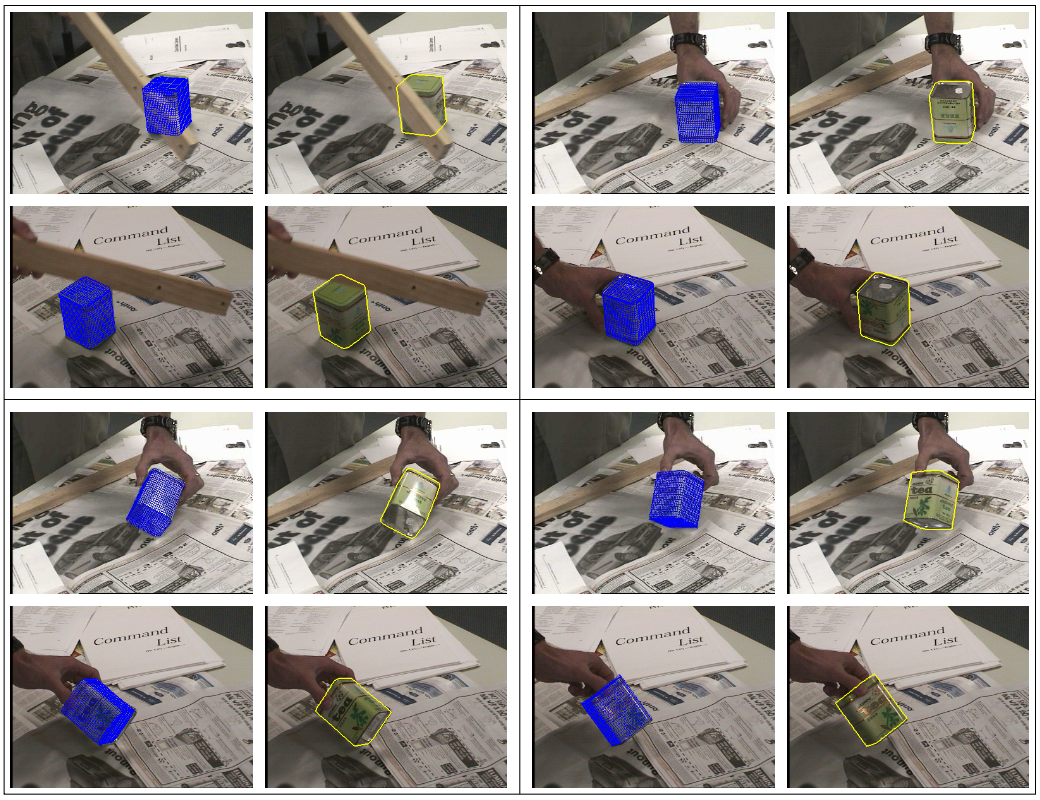


Figure 6: Results where a tea box was tracked with our algorithm. In each block, the computed pose (blue) and the contour (yellow) are shown for both views used for pose tracking. The scene contains partial occlusions (frame 97, top left), the tea box is turned upside down (frame 230, top right), there are specular reflections (frame 269, bottom left) and the box is turned around different axes simultaneously (frame 277, bottom right). From [16].

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 pose guess in a new frame [22].

Instead of using such a complicated and possibly time consuming approach, we propose another idea to diminish the problem of inaccurate pdfs at the beginning of new frames, namely to reuse pdfs from the previous frame for the first iterations in a new frame. Although it is possible to use the old pdfs only for a certain amount of iterations, the best results are often achieved when the old pdfs are used for all computations in the frame. In that case,

new probability density functions are only computed after the estimation of the object position is completed, i.e. once per frame. Since this requires significantly fewer pdf computations, it also leads to a speedup of the algorithm.
One question arises when evaluating pdfs from the previous frame: In which position in the old image should the pdfs be evaluated? Since evaluating the pdfs is only necessary at the 2-D position of point correspondences (q, x) in the proposed algorithm, the old pdfs can be evaluated either: a) at q, or b) at the 2-D position to which x had been projected in the previous frame. Note that this question only arises when using a local model [14]. Further note that the second alternative is not possible in pose tracking algorithms that use segmentation, since there is no real 3-D information incorporated into the segmentation step.
Both methods have advantages and disadvantages: Method (b) could lead to improved results because using the old point positions to evaluate the pdfs guarantees that the pdf of the object is evaluated at the right point on the object model. However, the pdf for the background might be better when using method (a), which also requires less time and less memory. In practise, however, both methods yield very similar results and the additional computational costs of method (a) are negligible.

4.2 Kinematic chains

Another extension enables the algorithm to handle kinematic chains. Kinematic chains are the name for a system of rigid bodies which are connected by joints [23]. Since every joint angle is an additional parameter that must be estimated, pose tracking becomes far more challenging this way. Incorporating kinematic chains, the function P – and thus the energy function (9) – not only depend on the twist ξ but also on the joint angles in the chain. However, there is no further change in the energy function. Thus the force vectors are still computed in the same way as before. In fact, the only change necessary to include kinematic chains into the algorithm is in how the position of the transformed 3-D model points – and thus the projected model points – are computed. Note that a joint angle can be modelled by a twist of the form $\theta\xi$ with unknown $\theta \in \mathbb{R}$ and known ξ (the rotation axis is a part of the model representation, and there is no translation). Thus, the equation of a point x_i on a jth joint has to satisfy

$$\left(\exp(\hat{\xi})\exp(\theta_1\hat{\xi}_1)\cdots\exp(\theta_j\hat{\xi}_j)x_i\right)_{3\times1} \times n_i - m_i = 0 \qquad (11)$$

in order to lie on the Plücker line $L = (n_i, m_i)$. After all exponential functions are linearized in the same way as in Section 2.3, one such constraint results in three linear equations in the six pose parameters and the joint angles.
A possible problem when using 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, seen from the front) or because it is occluded. In this case, the angle of the corresponding joint cannot be estimated from the system of equations because there is no information about it encoded in the system. This problem can be solved by including prior knowledge on the angles to which the angle configuration is drawn. We refer to [24] for further details. Examples for kinematic chains that have been tracked using such prior knowledge are shown in the next section.

5 Experiments

In this section, tracking results from very different scenes obtained with the proposed algorithm are presented.
In Figure 6, a tea box is to be tracked in a stereo sequence. In the first 160 frames of the sequence, the tea box is static but partially occluded by a wooden beam between the tea box and one or both cameras. Afterwards, the tea box is picked up and rotated. The frames 260 to 285 are especially challenging since the tea box is rotated around two axis simultaneously while specular highlights and reflections are visible on different parts of the tea box. The best results are achieved by setting l to 2.0 and choosing a threshold of 0.1mm for translational and 0.001 for the rotational parts of the twist.
Using this set of parameters and a C++ implementation, about 28.75 minutes of processor time are necessary to track the whole sequence on an Intel Pentium 4 with 3.2GHz. On average, 12 frames were tracked every minute. Most of this time (86%) was used for preprocessing (loading images, computing texture features, etc.) while only about a seventh part of the time was used for the actual pose tracking, for which an average of 12.03 iterations is needed. The two parameters (i.e. the threshold and l) have been chosen in such a way that the best results are obtained. As we will explain later in this section, faster but less accurate results are possible.
Figure 7 shows the time required per frame by our implementation for this scene. It can be observed that our algorithm is faster in "simple" situations, e.g. when nothing moves. Furthermore, we also show the changes in the translation and rotation parameters for the first 160 frames in this figure. With a standard deviation of 1.79 degrees and 0.83mm, our results are also

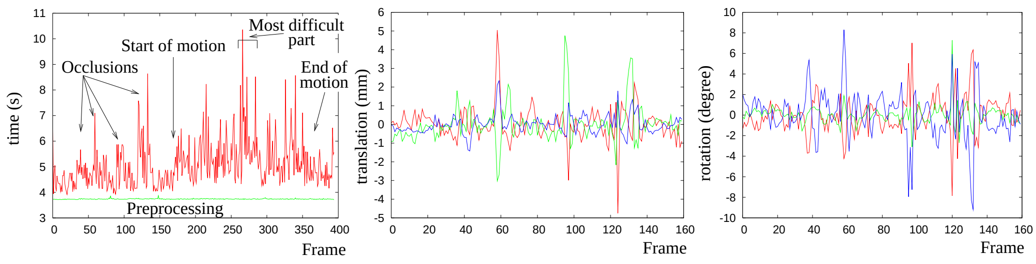


Figure 7: **Left:** Time needed for preprocessing (straight green line) and the total time used per frame (spiky red line) for the stereo image sequence shown in Figure 6. It can be seen that preprocessing requires most of the time. Furthermore, it is clearly visible that more time is needed in difficult situations. **Middle:** Changes of the three translation parameters in millimeters for the frames of the sequence in which the box is static and only occlusions occur. **Right:** Changes of the three Euler angles for the same frames, in degrees. Although there are some errors, the results are usually quite good. From [16].

quite good in this static part of the sequence.
Figure 8 shows three frames of a monocular sequence with a wooden toy giraffe. Since only one view is available, and since the appearance of the wooden giraffe is rather complicated, 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 (cf. Section 4.1). The results shown in Figure 8 are the only results presented in which the old pdfs have not been used for all computations of the next frame. Instead, new probability density functions are computed after 500 iterations.
When tracking objects that are clearly separated from the background, features from the texture space can be neglected and the local Gaussian model can be replaced by a global one. These changes noticeably decrease the runtime of our algorithm. For example, the teapot shown in Figure 9 has been tracked in a stereo sequence with more than one frame per second of processor time.
Ignoring texture information, increasing l to 6 and lowering the threshold, the tea box sequence shown in Figure 6 can be tracked (with slightly less accurate results) in less than 4 minutes ($\approx$ 104 frames per minute of processor time). This indicates that real-time processing with a region-based approach is feasible.
Figure 10 shows an example of a scene with fast movements and large accelerations. Although only one view was available, the white puncher was tracked

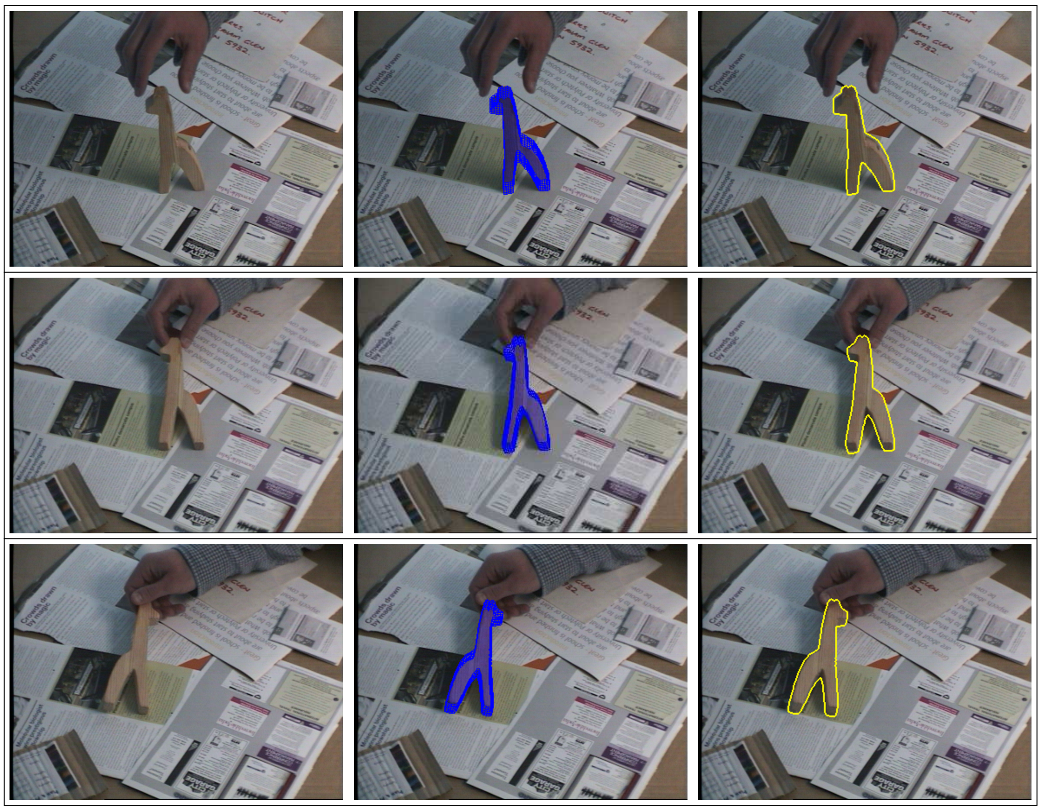


Figure 8: **From left to right:** Input image, estimated pose and extracted contour for three frames of a color sequence with a wooden giraffe. **Top**: Frame 32. **Middle**: Frame 64. **Bottom**: Frame 84. The surface model consists of a single closed point grid. Thus, it is possible to look through the projected pose. Note that this is irrelevant for contour-based pose estimation, where only silhouette points are needed.

even though it moved with more than 50 pixels per frame due to the built-in prediction of the next pose in our algorithm. Moreover, the algorithm was able to deal with accelerations of more than 70 pixels per frame.
Tracking results for scenes in which the model was a kinematic chain are shown in Figures 11 and 12. For these scenes, grey-scale images from four cameras were available, as well as prior information (see Section 4.2). Apart from smaller problems due to the fact that a human cannot be perfectly modelled by a kinematic chain, the only parts sometimes tracked inaccurately were the feet. This is most likely the case because the feet are white while the rest of the object is black. Note that the 'Cart'-sequence is tracked faster than the 'Flip'-sequence because the per-frame movement of the person is

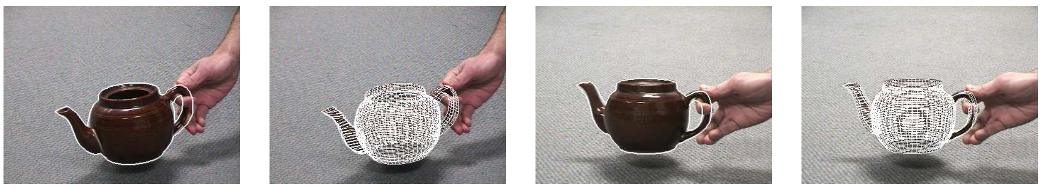


Figure 9: One frame of a stereo sequence in which a teapot has been tracked. Estimated contours and poses are shown in white. Scenes as easy as this are tracked much faster than complicated scenes.

slower there.

An overview of all experiments presented is shown in Table 1. In that table, the first sequence ("Tea box") corresponds to the best results we have achieved for the tea box sequence so far. The second ("Tea box fast") is a result for the same sequence with parameters that result in a low runtime but are sufficient to track the tea box over the whole sequence, as explained above. This shows that it is possible to significantly speed up the algorithm if less accurate results are sufficient.

Furthermore, it can be seen that more iterations are needed for monocular sequences, or if the rigid object is extended to a kinematic chain.

6 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. Another presented extension was to recycle probability density functions from previous frames. This has led to substantial speedups and increased the robustness of the algorithm in case of fast object motions with high accelerations. The algorithm was evaluated using scenes with changing illuminations, partial occlusions, cluttered backgrounds, specular highlights and several different models with and without joint angles.

References

[1] W.E.L. Grimson, T. Lozano–Perez, and D.P. Huttenlocher. *Object Recognition by Computer: The Role of Geometric Constraints.* MIT Press, 1990.

[2] D. Lowe. Solving for the parameters of object models from image descriptions. In *Proc. ARPA Image Understanding Workshop*, pages 121–127, College Park, April 1980.

[3] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89, October 2005.

[4] 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.

[5] 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.

[6] 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, May 1992. Springer.

[7] 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, June 2000. Springer.

[8] 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.

[9] A. Agarwal and B. Triggs. Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58, January 2006.

[10] D. Demirdjian L. Taycher, G. Shakhnarovich and T. Darrell. Conditional random people: Tracking humans with CRFs and grid filters. In *Proc. 2006 IEEE Computer Society Conference on Computer Vision*

and Pattern Recognition, pages 222–229, New York, NY, December 2006. IEEE Computer Society Press.

[11] C. Sminchisescu, A. Kanaujia, and D. Metaxas. Learning joint top-down and bottom-up processes for 3d visual inference. In *Proc. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1743–1752, New York, NY, USA, June 2006. IEEE Computer Society Press.

[12] S. Winkler, P. Wunsch, and G. Hirzinger. A feature map approach to real-time 3-D object pose estimation from single 2-D perspective views. In E. Paulus and F. Wahl, editors, *Mustererkennung 1997 (Proc. DAGM)*, pages 129–136, Berlin, September 1997. Springer.

[13] J. Kim, M. Çetin, and A. S. Willsky. Nonparametric shape priors for active contour-based image segmentation. In *European Signal Processing Conf. (EUSIPCO)*, Antalya, Turkey, September 2005.

[14] T. Brox, B. Rosenhahn, and J. Weickert. Three-dimensional shape knowledge for joint image segmentation and pose estimation. In W. Kropatsch, R. Sablatnig, and A. Hanbury, editors, *Pattern Recognition*, volume 3663 of *Lecture Notes in Computer Science*, pages 109–116, Berlin, August 2005. Springer.

[15] M. Bray, P. Kohli, and P. Torr. PoseCut: Simultaneous segmentation and 3D pose estimation of humans using dynamic graph-cuts. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision – ECCV 2006, Part II*, volume 3952 of *Lecture Notes in Computer Science*, pages 642–655, Berlin, May 2006. Springer.

[16] 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.

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

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

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

[20] 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.

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

[22] 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 *LNCS*, pages 98–111, Graz, Austria, May 2006. Springer.

[23] 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, June 1998. IEEE Computer Society Press.

[24] T. Brox, B. Rosenhahn, U. Kersting, and D. Cremers. Nonparametric density estimation for human pose tracking. In K. Franke et al., editor, *Pattern Recognition*, volume 4174 of *Lecture Notes in Computer Science*, pages 546–555, Berlin, September 2006. Springer.

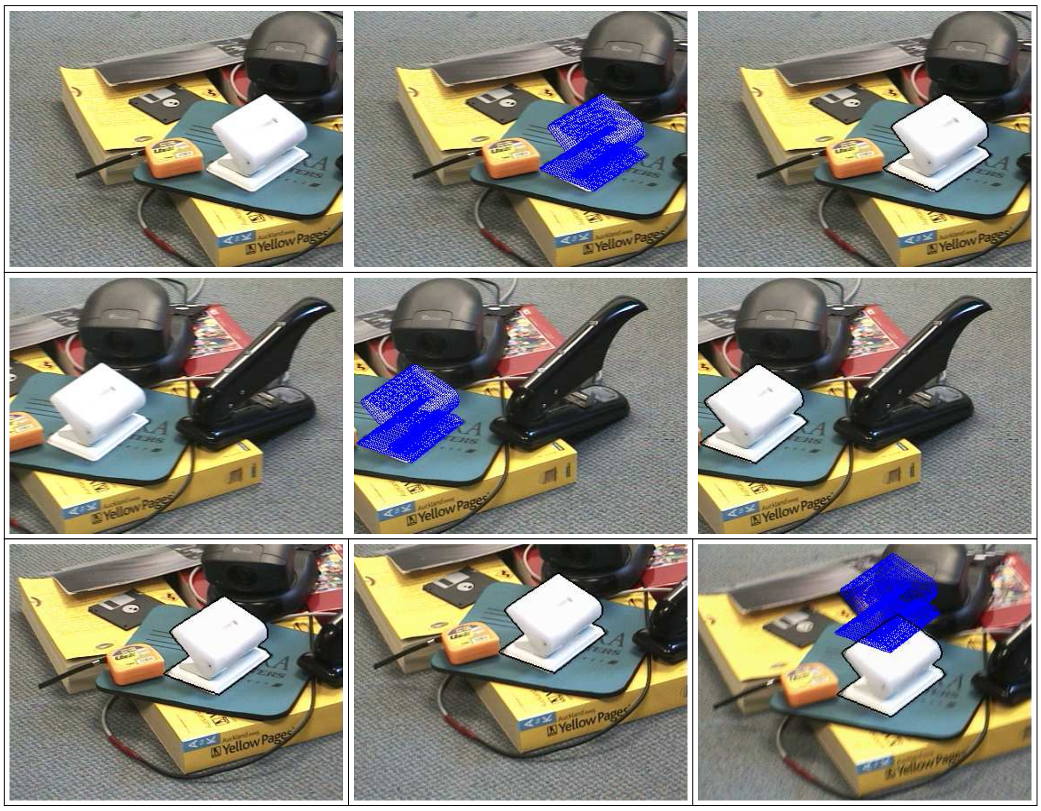


Figure 10: In the upper two rows, the input image (left), the estimated pose (middle) and the estimated contour (right) for frame 10 and 16 of a monocular sequence are shown. The last row shows the estimated contours for the frames 101, 102 and 103 in black. Note the fast movement between frame 10 and 16 and the pretty sudden turnaround in the frames 101 to 103. Also note the motion blur visible in frame 103. The blue 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. The position of the puncher estimated by our algorithm is indicated by the black contour.

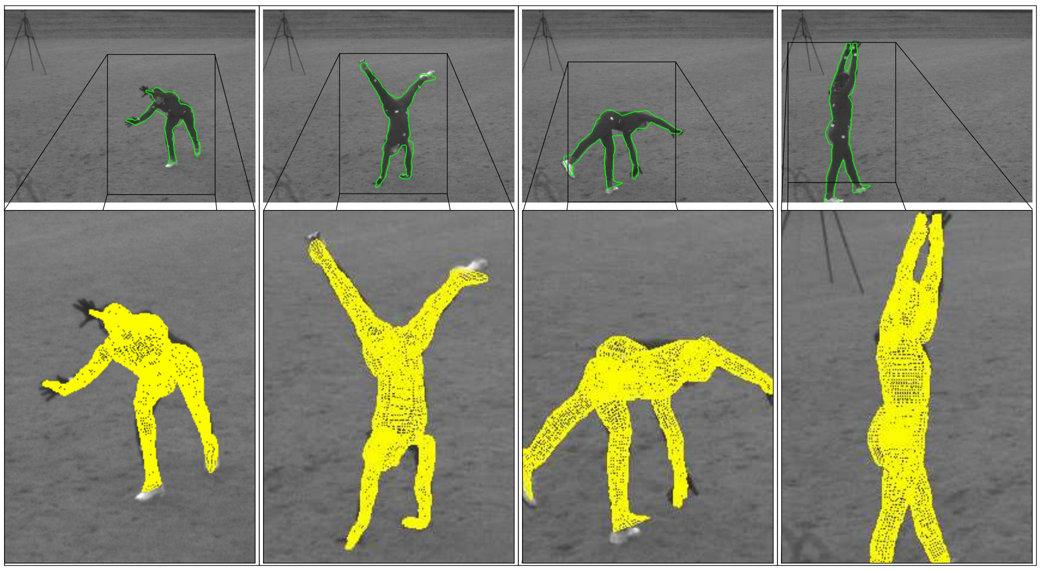


Figure 11: Here, tracking results of the frames 180 (left), 260 (middle) and 460 (right) of a person doing a cartwheel are presented. Only one of the four views used for pose tracking is shown here.

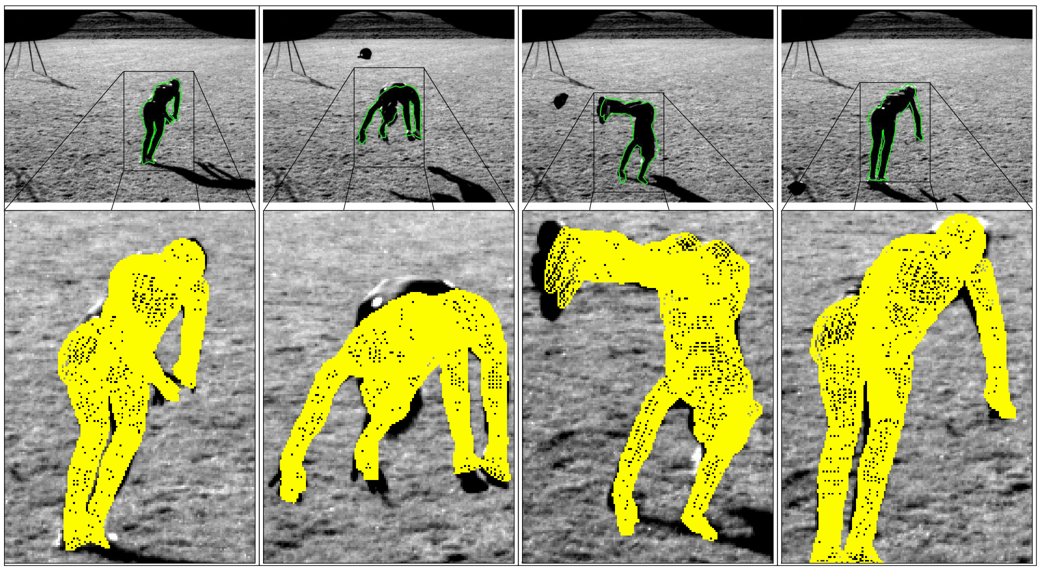


Figure 12: Another sequence from four views where the person turns two backflips. The first pair of images (left, frame 50) shows the start of the second flip. The four images in the middle show the frames 96 and 148, and in the images on the right (frame 190) the backflip is completed.