RWTH AACHEN UNIVERSITY

Fakultät für Mathematik, Informatik und Naturwissenschaften
Lehrstuhl für Informatik VIII (Computergraphik, Computer Vision und Multimedia)
Mobile Multimedia Processing
Prof. Dr. Bastian Leibe

Diplomarbeit

Robust Visual Tracking using Level Sets

Esther Horbert
Matrikelnummer: 252518

Juni 2010

Erstgutachter: **Prof. Dr. Bastian Leibe**
Zweitgutachter: **Prof. Dr. Leif Kobbelt**

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mir bei der Erstellung dieser Arbeit geholfen haben:
Prof. Dr. Bastian Leibe, für die Möglichkeit, die vielen Ideen, Erklärungen und die fortwährende Unterstützung.
Prof. Dr. Leif Kobbelt, für die hilfreichen Anregungen.
Meinem Betreuer, Dennis Mitzel, für die beste Zusammenarbeit, die man sich wünschen kann, und dafür dass er immer für mich Zeit hatte.
Josephine Sullivan, dafür dass sie meine Begeisterung für Computer Vision geweckt hat.
Charles Bibby und Andreas Ess, die ihre Videosequenzen zur Verfügung gestellt haben.
Georgios Floros, Dennis Mitzel, Patrick Sudowe und Tobias Weyand für die Hilfe bei kleinen und größeren Problemen, fürs Korrekturlesen und für die tolle Arbeitsatmosphäre, durch die auch die anstrengendsten Phasen immernoch Spaß gemacht haben.
Und vor allem meinen Eltern Maria und Peter Horbert, die mich finanziell unterstützt haben und die immer für mich da sind und an mich glauben. Außerdem meiner Schwester Rebecca Horbert, die mich oft motiviert hat.
Vielen Dank!

I hereby affirm that I composed this work independently and used no other than the specified sources and tools and that I marked all quotes as such.

Hiermit versichere ich, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

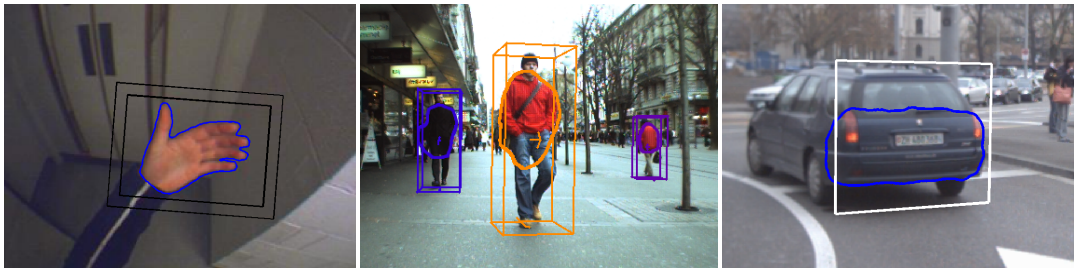
Aachen, December 9, 2011

(Esther Horbert)

Contents

Chapter 1

Introduction



Visual tracking has a large variety of potential applications in *e.g.* computer vision, human-computer interaction, mobile robotics, autonomous driving and driver assistance systems. In this work, we address the more specific problem of multi-object tracking from a moving platform, like *e.g.* a robot or a car. Realistic scenarios for this application pose challenges like multiple objects in crowded scenes with possible occlusions and cluttered background. Because of the moving camera, background modelling [SG99] is not applicable, which means that the objects of interest have to be located in the first place. This can be done by using visual object detectors [DWSP09] and consequently tracking-by-detection has become the dominant paradigm [OTdF$^+$04, WN07, ARS08, HWN08, LSV08, ELSV09]. In this framework an object detector is used in every frame and the resulting detections are associated to tracks for the individual detected objects. However, the detector cannot distinguish between different instances and data-association is complex since there are misaligned, false positive and false negative detections [DWSP09].

On the other hand, using only detections might be unnecessarily difficult. Level set segmentation and tracking approaches have made great progress in recent years [CRD07] and can track previously unseen objects without strong information about shape, appearance or motion [BR08]. These are region-based approaches and can be used to track specific objects over a potentially large number of video frames. Because they operate on the image-level they do not require further data association for generating tracklets (fragments of trajectories) and we will show that a level set tracker can

thus be used to provide further information to a high-level multi-hypothesis tracking-by-detection system.

If tracking-by-detection is to be used on vehicles there is another challenge: Vehicles can move much faster than pedestrians and often switch between acceleration and deceleration. Moreover they are non-holonomic, which means they can only move in the direction perpendicular to their wheel axes, yet this direction cannot be observed. It is thus much more difficult to model the motion of a vehicle, since detector bounding boxes offer only very little information. We will show how our level set tracking approach can be extended to incorporate a constrained homography, with which a vehicle's orientation can be estimated for every frame. These additional measurements make the trajectory estimation more robust and lead to smoother trajectories.

1.1 Related Work

Our level set tracker builds on the excellent work by [BR08], who have recently extended their work in order to track multiple objects by modelling the objects' relative depth [BR10]. A layered or 2.5D representation [NM90] has proven successful in representing the presence of multiple objects, which are possibly occluding each other. We also use a layered approach, however, we infer the objects' order from stereo depth maps or bounding box intersections with a ground plane instead of the image information. Given this sorting, we mask out occupied pixels for more distant objects. Apart from ordering the objects, they are tracked completely independently, *i.e.* without any probabilistic exclusion as for example in [MB99, BR10] or level sets with multiple regions as in [BW04].

Multi-person tracking from a mobile platform is a core capability for many applications in mobile robotics and autonomous vehicles [GM07]. While early approaches have been developed already some while ago for aerial scenarios [TSK02, KPB$^+$05], an application on ground-level poses significant additional difficulties, like occlusions and changing background. Robust multi-person tracking in such challenging situations has only recently become feasible by the development of powerful tracking-by-detection approaches [WN07, GM07, ARS08, LSV08, ELSV09]. Various strategies have been developed for solving the challenging data association problems encountered here. However, most of them regard only a single-layer tracker [OTdF$^+$04, GGB06, WN07, LSV08, ELSV09]. Most directly related to our approach are the multi-layer models of [TSK02, KPB$^+$05], which also initialize a number of low-level trackers to follow individual objects and later integrate their results in a high-level tracker. However, their frameworks are based on aerial scenarios, where adaptive background modelling is still feasible. [HWN08] also propose a hierarchical data association framework that links detection responses to form tracklets at an image level, before fusing the tracklets and integrating scene constraints at higher levels. Their approach is however targeted at a surveillance application with a static camera. [ARS08] extract position and articulation of the limbs and associate tracklets with the

use of a dynamical model and an appearance model. They focus on handling complex and long occlusions by use of these models. To our knowledge, ours is the first approach that integrates segmentation-based level-set trackers [BR08, CRD07] with a tracking-by-detection framework for street-level mobile tracking and is to be published in [MHLE10].

Tracking-by-detection approaches can also be applied to vehicles. However, for elongated objects with non-holonomic motion constraints, the raw detection bounding boxes often do not constrain the object motion sufficiently, making robust trajectory estimation difficult. Model-based tracking approaches try to obtain more information about the tracked objects by estimating their precise pose [KDN93, DT97]. They require a 3D model of the target object, which makes it hard to apply them for complex outdoor settings where many different objects can occur. The complexity can be reduced by limiting pose estimation to a planar region, for which efficient template-based tracking schemes can be used [CJDC02]. By decomposing the homography estimated from the template deformation, information about the 3D object motion can be obtained [BMR05], but this approach heavily relies on sufficient texture content inside the tracked region, which restricts it mainly to tracking fiducial regions. In the context of region-based tracking, little work has been done in order to incorporate dedicated 3D scene constraints. [FDP06] explore affine motion models in order to track multiple regions under 3D transformations. [BRW05] and [PR09b] propose different ways of combining level-set tracking with direct 3D pose estimation. Both assume a detailed 3D model of the target object to be available, which is not the case in our application. [SC09] propose a globally optimal approach for contour tracking which is also applied to an automotive scenario. However, this approach does not use knowledge about the geometric meaning of the changed contour. Our work on geometrically constrained level set tracking for automotive applications is to be published in [HMLE10].

1.2 Contributions

We make the following contributions: (1) We show how a level set segmentation and tracking framework can be automatically initialised with detector bounding boxes and track multiple objects in front of challenging backgrounds. (2) We integrate this level set tracker with a high-level tracking-by-detection system, such that they can overcome each other's weaknesses by closely cooperating, and achieve robust multi-person tracking performance. In contrast to existing tracking-by-detection systems our framework can continue tracking objects that are only partially visible and would not be found by a detector. (3) By extending our level set tracker to utilize a geometrically constrained homography, we can draw conclusions about the detailed 3D-movement of vehicles without using any model. This additional information can be used to improve the trajectories generated by a high-level tracker.

1.3 Outline

This work is structured as follows. Chapter 2 gives background information helpful for understanding this thesis. It introduces level set methods, structure-from-motion and stereo depth, and can be skipped if these are familiar. In Chapter 3 we introduce the level set segmentation and tracking framework by Bibby and Reid [BR08] and conduct a detailed analysis of the key parameters. We present several extensions to this approach in Chapter 4, among other things we show how to achieve higher robustness to automatic initialisation and how stereo depth information can be included. The next two chapters describe the integration with a tracking-by-detection system: Chapter 5 presents a multi-person tracking framework and Chapter 6 a geometrically constrained level set tracker for tracking vehicles. Both chapters contain a more detailed problem description and list of contributions. We conclude in Chapter 7. Appendix A contains derivations and Appendix B shows further result images.

Abbreviations

LS level set

HL high-level

SfM structure-from-motion

fps frames per second

Chapter 2

Background

In this chapter we briefly introduce several methods employed in this work. The segmentation and representation of a tracked shape is done using level set methods as presented in Section 2.1. Section 2.2 outlines how the camera's position in 3D coordinates and an estimation of the ground plane can be obtained with structure-from-motion and Section 2.3 explains the computation of depth maps from stereo video.

2.1 Level Sets

Level set methods were introduced by Osher and Sethian [OS88] and comprise numerical techniques for analysing and computing interface motion. The approach is very versatile and relevant to many disciplines, *e.g.* fluid mechanics, material science and computer vision [Set96].

A given boundary is assumed to move in a direction normal to itself (Fig. 2.1(a)) with a known speed function F, which may depend on factors like local properties, such as curvature and normal direction, global properties of the front or independent properties.

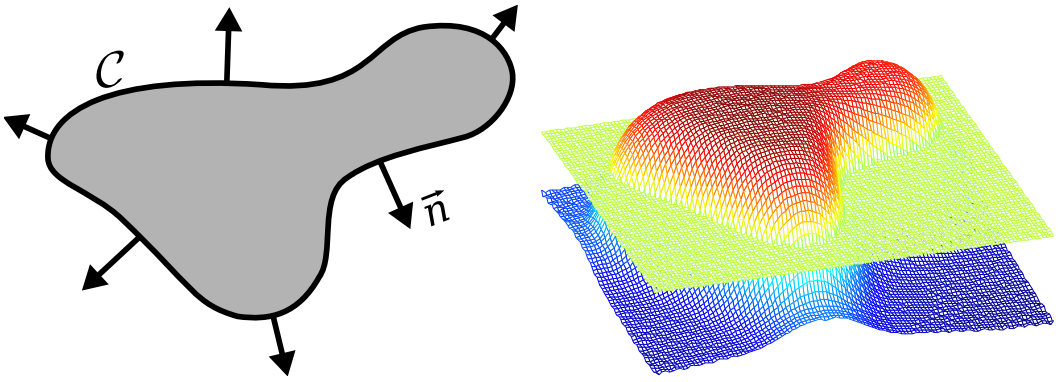
$$\frac{\partial \mathcal{C}}{\partial t} = F \cdot n \tag{2.1}$$

The main idea for representing this boundary is to embed it as the zero level set of a higher dimensional function $\Phi : \Omega \to \mathbb{R}$, the level set embedding function (Fig. 2.1(b)). Values of Φ are positive inside the contour and negative outside. In this implicit representation the contour is given by:

$$\mathcal{C} = \{x \in \Omega | \Phi(x) = 0\} \tag{2.2}$$

To evolve the contour a partial differential equation for the time-dependent embedding function Φ is derived. Since $\Phi(\mathcal{C}(t),t) = 0$ at all times,

$$\frac{d}{dt}\Phi(\mathcal{C}(t),t) = \nabla\Phi\frac{\partial \mathcal{C}}{\partial t} + \frac{\partial \Phi}{\partial t} = \nabla\Phi F \cdot n + \frac{\partial \Phi}{\partial t} = 0. \tag{2.3}$$



(a) (b)

Figure 2.1: (a) Curve propagation [Set96]; (b) Level set embedding function Φ and zero level.

With the normal $\vec{n} = \frac{\nabla\Phi}{|\nabla\Phi|}$ [Set96] the evolution equation for Φ is:

$$\frac{\partial\Phi}{\partial t} = -|\nabla\Phi|F. \tag{2.4}$$

Alternatively one can derive the level set equation from a variational formulation, by embedding a variational principle $E(\mathcal{C})$ on the contour via a variational principle $E(\Phi)$ on the level set function [CRD07]. The Euler-Lagrange equation which minimize this energy functional can be derived as:

$$\frac{\partial\Phi}{\partial t} = -\frac{\partial E(\Phi)}{\partial\Phi}. \tag{2.5}$$

Level set formulations have several desirable aspects. First, the evolving function Φ remains a function, even if the propagating contour changes topology, breaks, merges or forms sharp corners. Moreover Φ can be implemented using a discrete grid and substituting spatial and temporal derivatives by finite difference approximations. Finally, the intrinsic properties of the contour can easily be determined from the higher dimensional function Φ, *e.g.* the normal vector $\vec{n} = \frac{\nabla\Phi}{|\nabla\Phi|}$ and the curvature $\kappa = \mathrm{div}(\frac{\nabla\Phi}{|\nabla\Phi|})$.

There are a number of efficient schemes for computing level set evolution. In this work the narrow band approach is used: Computations are performed on the neighbourhood of the zero level set only, as opposed to all grid points, constituting a significant cost reduction [Set99].

In the context of visual tracking level set methods are used to model boundaries, *i.e.* an initial curve is evolved towards the boundaries of a target object to obtain a segmentation [OP96]. The evolved curve is a representation of the object's shape. We will present the probabilistic level set segmentation framework by Bibby and Reid [BR08] in detail in Chapter 3.

2.2 Structure-from-Motion

Structure-from-motion (SfM) is the process of deducing three-dimensional structure from video by analysing objects' motion over time. Given a pair of images, the 3D positions of points are computed by triangulation. SfM can be used on monocular or stereo video, however stereo video provides more information and can be used to extract scene knowledge even if the camera is moving very slowly or standing still [NNB06].

In chapters 5 and 6 we address the problem of tracking from a mobile platform, for which scene knowledge provides very useful information. The following two sections describe how the camera position and the ground plane can be estimated using SfM.

2.2.1 Visual Odometry

Visual odometry is the process of estimating the camera's 3D position and orientation using SfM. Knowing the camera's position allows reasoning about the tracked object's trajectories in the world coordinate system. The camera positions used in this work were obtained with the system by [ELSV09], which builds on previous work by [NNB04]. In short it works as follows: Point features, *e.g*. Harris corners, are detected and matched between the left and the right frame and used to triangulate 3D points. These features are tracked over time (in subsequent frames), *i.e*. matched with the previously obtained points, and new features are detected. Triples of these 3D points are used to generate hypotheses with the 3-point pose algorithm [Nis04]. The hypotheses are evaluated in a preemptive RANSAC framework and refined with windowed bundle adjustment. A Kalman filter is used to predict the next camera position for feature detection. In addition, independently moving objects are sorted out at an early stage and short occlusions of the tracked scene points are bridged. Older frames are discarded, along with the points without support in the examined frames.

Robust performance is guaranteed by failure detection mechanisms. In case of failure the Kalman filter prediction is used instead, all feature points are discarded and the system is restarted [ELSV08]. This can introduce drift, but a locally smooth trajectory is more important for our tracking application.

2.2.2 Ground Plane Estimation

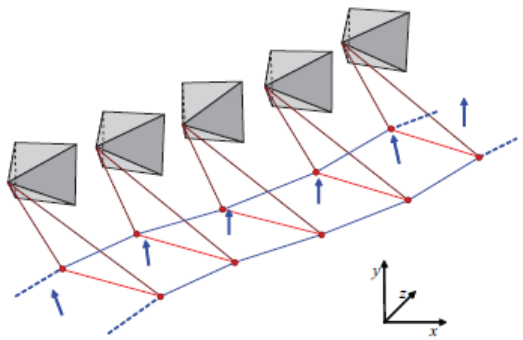
An estimated ground plane helps substantially in constraining object detection to meaningful locations [HEH06, LCC$^+$07]. It is defined for the current camera frame and consists of the normal vector $\mathbf{n}$ and the distance to the camera center d:

$$\pi = (\mathbf{n}, d). \tag{2.6}$$

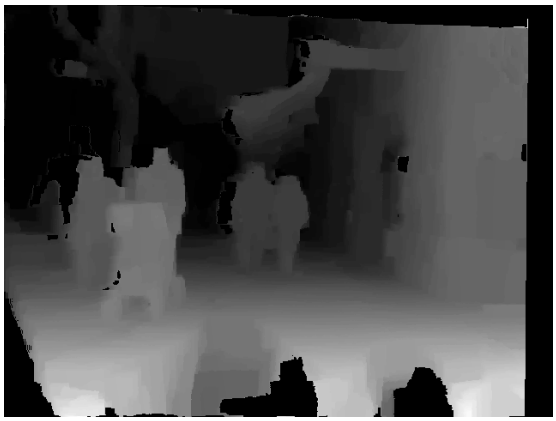
The ground plane estimates employed in this work were obtained from the camera positions (from visual odometry) and knowledge about the used camera setup (relative

position of the wheels to the camera). Fig. 2.2(a) shows a path of camera positions. The ground plane estimates are obtained by fitting trapezoidal patches to the four reconstructed wheel contact points of two adjacent frames and smoothing their normals over several frames [LSV08]. This method thus estimates the plane underneath the camera setup and can neither account for alternating incline nor describe multiple planes (*e.g.* stairs), however, it is suitable for mostly planar areas as in our applications.

The ground plane can also be estimated from depth maps using a global optimization framework [ELSV09].



(a) Ground plane estimation (taken from [LSV08])



(b) Example disparity map

Figure 2.2: (a) Ground plane estimation with camera positions. (b) Example for a disparity map. Black corresponds to areas for which no depth information could be determined.

2.3 Stereo Depth

Generating depth maps from stereo video is a well studied topic with many different methods [SS02]. The distance of points to the cameras is computed from the disparity between point correspondences in the two images. The challenge is producing dense depth maps, *i.e.* a depth estimation for every pixel. Undertextured or distant regions for example do not produce many point correspondences. Moreover since the two cameras are at a different angle there are areas only visible to one of them. Computing dense depth maps thus requires some form of interpolation to fill these regions. Our depth maps were obtained using convex optimization [ZNF09] and belief propagation [FH06], which enforces constraints like smoothness and finds an optimal solution.

Another restriction of depth maps arises from the discrete nature of the input image. The displacement between two corresponding feature points is used to compute a disparity map. The larger the distance the smaller the displacement. Since this disparity can only be measured in pixels, the resulting depth map contains only a finite

number of distance values. The interval between these values grows logarithmically with the distance from the camera.

Fig. 2.2(b) shows an example for a disparity map, where black corresponds to pixels for which no depth could be determined. Those are for example very distant structures or undertextured areas on the ground in front of the cameras. Moreover, since this depth map belongs to the left camera stream, there are black "shadows" on the left side of the persons and at the right image border, since those areas were not visible to the right camera.

We employ stereo depth maps for additional appearance information in our level set tracker in Section 4.4 and for consistency checks for multi-person and object tracking from mobile platforms in chapters 5 and 6.

Chapter 3

Level Set Segmentation and Tracking using Pixel-Wise Posteriors

This chapter presents the segmentation and tracking framework introduced by Bibby and Reid in [BR08], a probabilistic level set framework for visual tracking of previously unseen objects from a moving camera. It comprises segmentation, rigid registration between frames and shape adaptation, without using prior information about shape, appearance or movement. The target object's contour is represented by a level set and pixels are partitioned into foreground (inside the contour) and background pixels (outside the contour). The exact position of pixels is not regarded, resulting in viewpoint invariance. Appearance models are built from colour histograms and learned online. This method differs from previous ones in the use of pixel-wise posteriors instead of likelihoods and achieves high robustness and real-time performance.

The following sections will describe the generative model (Section 3.1) that is used to derive the segmentation (Section 3.2) and tracking (Section 3.3) frameworks. In Section 3.4 we give a detailed analysis of the effect of key parameters and in Section 3.5 we will show that our implementation yields results of comparable quality to the ones presented in [BR08]. We discuss our results in Section 3.6.

3.1 Generative Model

Fig. 3.1 shows the generative model used in this approach. The tracked object is described by its shape, its location and by foreground and background appearance models.

Shape The contour $\mathbf{C}$ is represented by the zero level $\mathbf{C} = \{\mathbf{x}|\Phi(\mathbf{x}) = 0\}$ of the level set embedding function $\Phi(\mathbf{x})$, where $\mathbf{x}$ denotes the pixel location. It segments the object into two regions, which are denoted foreground and background and are described by appearance models. This level set is evolved in order to maximize the accordance with the appearance models, while fulfilling certain constraints on the shape of the embedding function and of the contour.

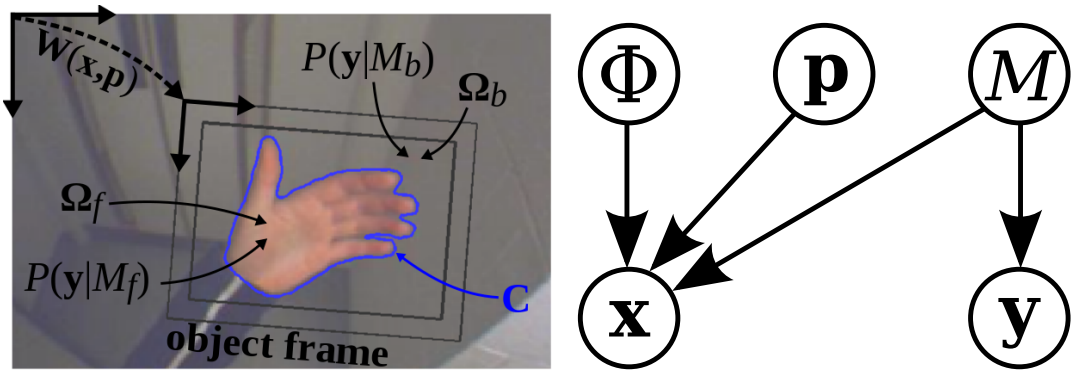


Figure 3.1: (Left): Representation of the object frame with the Contour $\mathbf{C}$, the foreground pixels Ω_f, the background pixels Ω_b, the foreground model $P(\mathbf{y}|M_f)$, the background model $P(\mathbf{y}|M_b)$ and the warp $\mathbf{W}(\mathbf{x};\mathbf{p})$. (Right): Generative model [BR08].

Location Segmentation and tracking are performed only on a part of the current image, *i.e.* an object frame, which contains the object to be tracked and some background. The position and scale of the object frame are described by the warp $\mathbf{W}(\mathbf{x},\mathbf{p})$ with parameters $\mathbf{p}$, which transforms the pixel coordinates w.r.t. the object frame into coordinates w.r.t. the original image coordinates. The tracking framework will perform a rigid registration, *i.e.* move the object frame on the image without changing the contour and find a new optimal position $\mathbf{p}$.

Appearance Models We use colour histograms with 32 bins per channel to compute the models $P(\mathbf{y}|M_f)$ and $P(\mathbf{y}|M_b)$: the probability of a colour $\mathbf{y}$ given the foreground M_f and the background M_b respectively.

The following notation will be used below:

- I: image
- $\mathbf{x}$: pixel coordinates inside object frame
- $\mathbf{y}$: colour of a pixel
- $W(x,\mathbf{p})$: warp with parameters $\mathbf{p}$
- $M = \{M_f, M_b\}$: foreground and background model parameters
- $P(\mathbf{y}|M_f)$: foreground model: probability of colour $\mathbf{y}$ given foreground
- $P(\mathbf{y}|M_b)$: background model: probability of colour $\mathbf{y}$ given background
- $\Phi(\mathbf{x})$: level set embedding function
- $\mathbf{C}$: contour of the object, represented by the zero level set $\mathbf{C} = \{\mathbf{x}|\Phi(\mathbf{x}) = 0\}$

- $\Omega = \{\Omega_f, \Omega_b\}$: pixels in the object frame partitioned into foreground and background
- $H_\varepsilon(z)$: smoothed Heaviside step function
- $\delta_\varepsilon(z)$: smoothed Dirac delta function

The joint distribution for a pixel according to the generative model in Fig. 3.1 is:

$$P(\mathbf{x}, \mathbf{y}, \Phi, \mathbf{p}, M) = P(\mathbf{x}|\Phi, \mathbf{p}, M)P(\mathbf{y}|M)P(M)P(\Phi)P(\mathbf{p}) \tag{3.1}$$

where the probability of the pixel location $\mathbf{x}$ depends on $\Phi, \mathbf{p}$ and M and the pixel value $\mathbf{y}$ depends on M, whereas $\Phi, \mathbf{p}$ and M are independent. To simplify the expressions in this section it is assumed without loss of generality that $\mathbf{W}(\mathbf{x}_i, \mathbf{p}) = \mathbf{x}_i$. See appendix A.1 for a detailed derivation of the following. A marginalisation over models M yields the pixel-wise posterior probability of the shape Φ and location $\mathbf{p}$ given a pixel $\mathbf{x}, \mathbf{y}$:

$$P(\Phi, \mathbf{p}|\mathbf{x}, \mathbf{y}) = \frac{1}{P(\mathbf{x})} \sum_{i=\{f,b\}} \Big\{ P(\mathbf{x}|\Phi, \mathbf{p}, M_i)P(M_i|\mathbf{y}) \Big\} P(\Phi)P(\mathbf{p}) \tag{3.2}$$

For the probability of shape Φ and location $\mathbf{p}$ given *all* the pixels in the object frame the single pixel posteriors are fused with a logarithmic opinion pool

$$P(\Phi, \mathbf{p}|\Omega) = \prod_{i=1}^{N} \sum_{M} \Big[P(\mathbf{x}|\Phi, \mathbf{p}, M_i)P(M_i|\mathbf{y}) \Big] P(\Phi)P(\mathbf{p}) \tag{3.3}$$

where $\frac{1}{P(\mathbf{x})}$ has been dropped since it is constant for all pixels and $P(\Phi, \mathbf{p}|\Omega)$ will only be maximized in the following. The probabilities in (3.3) are specified as follows: $P(\mathbf{x}_i|\Phi, \mathbf{p}, M)$, the probability of coordinates $\mathbf{x}_i$ given the contour Φ, the pose $\mathbf{p}$ and the foreground or background model, respectively, is given by:

$$P(\mathbf{x}_i|\Phi, \mathbf{p}, M_f) = \frac{H_\varepsilon(\Phi(\mathbf{x}_i))}{\eta_f}, \qquad P(\mathbf{x}_i|\Phi, \mathbf{p}, M_b) = \frac{1 - H_\varepsilon(\Phi(\mathbf{x}_i))}{\eta_b} \tag{3.4}$$

where H_ε is a smoothed Heaviside step function, *i.e.* one for coordinates inside the contour, zero for outside and a value between zero and one for coordinates with distance from the contour less than ε. In practice we clip H_ε to be slightly greater than zero and less than one, since it appears in the denominator. In our implementation:

$$H_\varepsilon(x) = \begin{cases} 1 \cdot 10^{-5}, & x < -\varepsilon \\ \frac{1}{2\varepsilon}x + \frac{1}{2\pi}\sin\left(\frac{\pi x}{\varepsilon}\right) + \frac{1}{2}, & -\varepsilon \leq x \leq \varepsilon \\ 1 - 1 \cdot 10^{-5}, & x > \varepsilon \end{cases} . \tag{3.5}$$

η_f and η_b correspond to the number of pixels in the foreground and background respectively:

$$N = \eta = \eta_f + \eta_b, \qquad \eta_f = \sum_{i=1}^{N} H_\varepsilon(\Phi(\mathbf{x}_i)), \qquad \eta_b = \sum_{i=1}^{N} 1 - H_\varepsilon(\Phi(\mathbf{x}_i)), \tag{3.6}$$

thus:

$$P(M_f) = \frac{\eta_f}{\eta}, \qquad P(M_b) = \frac{\eta_b}{\eta}. \tag{3.7}$$

When using level set methods it is numerically necessary to keep the level set embedding function close to a signed distance function during its evolution [OP96]. A signed distance function returns the distance from the contour with a positive sign inside the contour and a negative sign outside. After evolving the level set this property will typically not be fulfilled any more. One way of achieving numerically stable results is to re-initialize the level set function to a signed distance function according to the current contour after several iterations. Here a more elegant method is used: Since an embedding function Φ that is closer to a signed distance function is more desirable, it should be more probable. This can be done by introducing a term into the variational formulation that rewards a level set function that is closer to a signed distance function [LXGF05]. $P(\Phi)$ is a geometric prior:

$$P(\Phi) = \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp - \frac{(|\nabla\Phi(\mathbf{x}_i)| - 1)^2}{2\sigma^2} \tag{3.8}$$

which is a normal distribution with mean 1 and standard deviation σ, evaluated for the absolute value of the gradient of Φ. This means a gradient of 1, which corresponds to a signed distance function, receives the highest reward. The greater σ, the greater the width of the distribution and the smaller the punishment for a certain deviation from a signed distance function. $\frac{1}{\sigma^2}$ is the weight of the prior.
We denote

$$P_f = \frac{P(\mathbf{y}_i|M_f)}{\eta_f P(\mathbf{y}_i|M_f) + \eta_b P(\mathbf{y}_i|M_b)}, \quad P_b = \frac{P(\mathbf{y}_i|M_b)}{\eta_f P(\mathbf{y}_i|M_f) + \eta_b P(\mathbf{y}_i|M_b)} \tag{3.9}$$

and

$$P(\mathbf{x}_i|\Phi, \mathbf{p}, \mathbf{y}_i) = H_\varepsilon(\Phi(\mathbf{x}_i))P_f + (1 - H_\varepsilon(\Phi(\mathbf{x}_i)))P_b. \tag{3.10}$$

$P(\mathbf{y}_i|M_f)$ and $P(\mathbf{y}_i|M_b)$ are the appearance models, the probabilities of a colour given foreground or background, and are computed from a foreground and a background colour histogram. P_f and P_b are the probabilities of a pixel belonging to the foreground and background and are normalized such that $P_f + P_b = 1$.
Putting all of the above together (*c.f.* Appendix A.1) we arrive at

$$\log(P(\Phi, \mathbf{p}|\Omega)) \propto \sum_{i=1}^{N} \left\{ \log(P(\mathbf{x}_i|\Phi, \mathbf{p}, \mathbf{y}_i)) - \frac{(|\nabla\Phi(\mathbf{x}_i)| - 1)^2}{2\sigma^2} \right\} + N \log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) + \log(P(\mathbf{p})) \tag{3.11}$$

for the posterior.

3.2 Segmentation

To achieve a segmentation of the image into foreground and background pixels the probability of the level set function Φ and pose $\mathbf{p}$ given the pixels Ω, eq. (3.11) is to be maximized with respect to Φ, while $\mathbf{p}$ is constant. The first variation (Gateaux derivative) of eq. (3.11) by calculus of variation [Eva10] is (*c.f.* Appendix A.2):

$$\frac{\partial P(\Phi,\mathbf{p}|\Omega)}{\partial\Phi} = \frac{\delta_\varepsilon(\Phi)(P_f - P_b)}{P(\mathbf{x}|\Phi,\mathbf{p},\mathbf{y})} - \frac{1}{\sigma^2}\left[\nabla^2 - \operatorname{div}\left(\frac{\nabla\Phi}{|\nabla\Phi|}\right)\right] \tag{3.12}$$

where ∇^2 is the Laplacian operator and δ_ε the derivative of H_ε, a smoothed Dirac delta function:

$$\delta_\varepsilon = \begin{cases} 0, & |x| > \varepsilon \\ \frac{1}{2\varepsilon}\left[1+\cos\left(\frac{\pi x}{\varepsilon}\right)\right], & |x| \leq \varepsilon. \end{cases} \tag{3.13}$$

A maximum can now be found at $\frac{\partial P(\Phi,\mathbf{p}|\Omega)}{\partial\Phi} = 0$ with gradient ascent. Φ is initialized with a signed distance function of some initial contour, *e.g.* a bounding box and then evolved:

$$\Phi_{n+1} = \Phi_n + \tau\frac{\partial P(\Phi,\mathbf{p}|\Omega)}{\partial t} \tag{3.14}$$

with time step τ and gradient flow:

$$\frac{\partial P(\Phi,\mathbf{p}|\Omega)}{\partial t} = \frac{\partial P(\Phi,\mathbf{p}|\Omega)}{\partial\Phi}. \tag{3.15}$$

For stability $\frac{\tau}{\sigma^2} < 0.25$ must be satisfied [LXGF05]. In practice this is implemented using a numerical scheme on a discrete grid and the Laplacian is computed using the kernel

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{3.16}$$

Note that in the implementation the level set function is not continuous, but consists of a finite number of scalars. However the zero level set, *i.e.* the contour, still is continuous as the level set function does not necessarily contain zeros and needs to be interpolated to retrieve the contour.

In summary the level set function is first initialized with a bounding box around the object. When evolving it according to (3.14) its values will rise for pixels with a higher probability of belonging to the foreground and vice versa. In this way the contour evolves such that it encloses foreground pixels. After each iteration the foreground and background models are rebuilt with the newly obtained contour and thus refined. At the same time the second term in (3.12) will keep the level set embedding function in a numerically stable shape, which will mainly have the effect of keeping it smoother. The obtained contour is then tracked in the subsequent frames.

3.3 Tracking

Tracking is done by performing a rigid registration in every frame, *i.e.* by optimizing the pose $\mathbf{p}$ while keeping the contour Φ constant. Similar to inverse compositional image alignment [BM04], we aim at warping the next frame such that its content best fits the current contour (in the following we will use the terms *old* and *new* frame). By introducing a warp $\mathbf{W}(\mathbf{x},\mathbf{p})$ the object reference frame can be transformed with parameters $\mathbf{p}$. This leaves the level set embedding function untouched and only moves it to a position that matches the foreground and background models better. After registration the level set can be evolved to account for shape deformations and a drift correction keeps the borders of the object frame balanced and at a minimum distance. During tracking the foreground and background models M_f and M_b are not rebuilt in every frame, but instead slightly adapted during online learning.

3.3.1 Rigid Registration

Eq. (3.11) is now optimized w.r.t. the pose, so we can drop all terms that do not contain $\mathbf{p}$ as they are constant here:

$$\log(P(\Phi,\mathbf{p}|\Omega)) \propto \sum_{i=1}^{N}\left\{\log(P(\mathbf{x}_i|\Phi,\mathbf{p},\mathbf{y}_i))\right\} + \log(P(\mathbf{p})) + const. \tag{3.17}$$

The probability of a certain pose $P(\mathbf{p})$ is also dropped and instead handled by drift correction as will be described in the next section. We introduce $\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})$, which are the coordinates in the image that correspond to coordinates $\mathbf{x}_i$ in the object frame. $\Delta\mathbf{p}$ represents an incremental warp.

$$\log(P(\Phi,\mathbf{p}|\Omega)) \propto \sum_{i=1}^{N}\left\{\log(P(W(\mathbf{x}_i,\Delta\mathbf{p})|\Phi,\mathbf{p},\mathbf{y}_i))\right\} \tag{3.18}$$

Since all terms are probabilities, and thus positive, they can be expressed as squared square-roots and we can use Gauss-Newton for the optimization. We arrive at (see appendix A.3 for a detailed derivation):

$$\Delta\mathbf{p} = \left[\sum_{i=1}^{N}\frac{1}{2P(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})|\Phi,\mathbf{p},\mathbf{y}_i)}\left(\frac{P_f}{\sqrt{H_\varepsilon(\Phi(\mathbf{x}_i))}}+\frac{P_b}{\sqrt{1-H_\varepsilon(\Phi(\mathbf{x}_i))}}\right)\mathbf{J}^T\mathbf{J}\right]^{-1}\times \sum_{i=1}^{N}\frac{(P_f-P_b)\mathbf{J}^T}{P(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})|\Phi,\mathbf{p},\mathbf{y}_i)} \tag{3.19}$$

where

$$\mathbf{J} = \frac{\partial H_\varepsilon}{\partial\Phi}\frac{\partial\Phi}{\partial\mathbf{x}}\frac{\partial\mathbf{W}}{\partial\Delta\mathbf{p}} = \delta_\varepsilon(\Phi(\mathbf{x}_i))\nabla\Phi(\mathbf{x}_i)\frac{\partial\mathbf{W}}{\partial\Delta\mathbf{p}} \tag{3.20}$$

Note that $P(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})|\Phi,\mathbf{p},\mathbf{y}_i)$ still contains $\Delta\mathbf{p}$ while the rest of the terms does not. Although $\Delta\mathbf{p}$ is unknown at this point this is correct: We naturally need two images to

compute the warp between them, however eq. (3.19) does not reflect this in an obvious way. In a manner of speaking the object's points have been warped with $\Delta\mathbf{p}$ due to the object's (and camera's) movement in between the two frames. Thus the warp with $\Delta\mathbf{p}$ is already included in the image information of the new frame.
Given eq. (3.21) it is not possible to seek the warp from old to new image frame, since the pose of the contour in the new frame is unknown. The current object frame and contour in (3.21) correspond to the un-warped coordinates, thus it is necessary to switch the roles of old and new frame and compute the warp from new to old image. $\mathbf{W}(\mathbf{x}_i, \Delta\mathbf{p})$ can simply be inverted afterwards to obtain the tracking result.
In summary $P(\mathbf{W}(\mathbf{x}_i, \Delta\mathbf{p})|\Phi, \mathbf{p}, \mathbf{y}_i)$ refers to the old image and the rest of the terms refer to the new image:

$$\Delta\mathbf{p} = \left[\sum_{i=1}^{N} \frac{1}{2P(\mathbf{x}_i|\Phi,\mathbf{p},\mathbf{y}_i)^{old}} \left(\frac{P_f}{\sqrt{H_\varepsilon(\Phi(\mathbf{x}_i))}} + \frac{P_b}{\sqrt{1-H_\varepsilon(\Phi(\mathbf{x}_i))}}\right)\mathbf{J}^T\mathbf{J}\right]^{-1} \times \sum_{i=1}^{N} \frac{(P_f - P_b)\mathbf{J}^T}{P(\mathbf{x}_i|\Phi,\mathbf{p},\mathbf{y}_i)^{old}} \tag{3.21}$$

We apply a transformation with translation, scale and rotation which can be parametrised with

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}, \quad \mathbf{W}(\mathbf{x}_i, \mathbf{p}) = \begin{bmatrix} 1+p_1 & -p_2 & p_3 \\ p_2 & 1+p_1 & p_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}, \text{ with } \mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}. \tag{3.22}$$

If all parameters are zero the resultant warp is the identity. Then

$$\frac{\partial\mathbf{W}}{\partial\Delta\mathbf{p}} = \begin{pmatrix} \frac{\partial\mathbf{W}_x}{\partial\Delta\mathbf{p}_1} & \frac{\partial\mathbf{W}_x}{\partial\Delta\mathbf{p}_2} & \frac{\partial\mathbf{W}_x}{\partial\Delta\mathbf{p}_3} & \frac{\partial\mathbf{W}_x}{\partial\Delta\mathbf{p}_4} \\ \frac{\partial\mathbf{W}_y}{\partial\Delta\mathbf{p}_1} & \frac{\partial\mathbf{W}_y}{\partial\Delta\mathbf{p}_2} & \frac{\partial\mathbf{W}_y}{\partial\Delta\mathbf{p}_3} & \frac{\partial\mathbf{W}_y}{\partial\Delta\mathbf{p}_4} \end{pmatrix} = \begin{pmatrix} x_i & -y_i & 1 & 0 \\ y_i & x_i & 0 & 1 \end{pmatrix}. \tag{3.23}$$

The final pose $\mathbf{p}$ of the object frame in the new image is thus obtained in the following way:
Pre-compute:

1. Compute the object frame I^i_{obj} by warping the *old* image frame with $\mathbf{W}(\mathbf{x}_i, \mathbf{p})^{-1}$ (we use bilinear interpolation).
2. Compute $P(\mathbf{x}_i|\Phi, \mathbf{p}, \mathbf{y}_i)$ for I^i_{obj}.

Iterate:

3. Compute the object frame I^{i+1}_{obj} by warping the *new* image frame with $\mathbf{W}(\mathbf{x}_i, \mathbf{p})^{-1}$.
4. Compute $\Delta\mathbf{p}$ between I^i_{obj} and I^{i+1}_{obj} with (3.21).

5. Update the warp $\mathbf{W}(\mathbf{x}_i,\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x}_i,\mathbf{p}) \circ \mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})^{-1}$.

Until: the step size is small enough, $\|\Delta\mathbf{p}\| \leq \varepsilon_p$.

6. $\mathbf{W}(\mathbf{x}_i,\mathbf{p})$ is the warp for the new image frame and can be used to obtain the current contour.

The registration accounts for rigid movement but it does not incorporate shape deformations. We therefore evolve the level set for up to several iterations after registration in each frame in order to adapt the contour to a changing object, *i.e.* re-segment it.

3.3.2 Drift Correction

Re-segmentation has the side effect of moving the contour in relation to the object frame, it drifts inside its bounding box. At this point $P(\mathbf{p})$ from eq. (3.11) is revisited. We keep the margins between contour and box balanced and scale down or enlarge the box to keep the borders at a minimum of two and a maximum of four pixels in the foreground frame (a slightly smaller box inside the object frame).

3.3.3 Online Learning

The foreground and background models are appearance models built with the first segmentation. In the following steps we do not re-built the models but only slightly adapt them online in order to achieve higher robustness. We use a linear opinion pool and learning rates $\alpha_i, i = \{f,b\}$:

$$P_t(\mathbf{y}|M_i) = (1-\alpha_i)P_{t-1}(\mathbf{y}|M_i) + \alpha_i P_t(\mathbf{y}|M_i), \quad i = \{f,b\} \tag{3.24}$$

In our implementation we set $\alpha_f = 0.02$ and $\alpha_b = 0.025$.

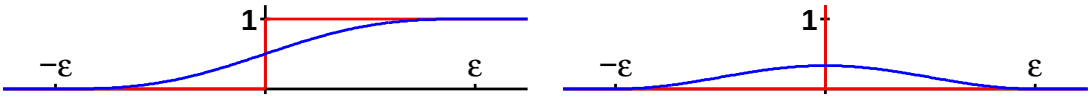
3.4 Parameters

In this section we detail the effect and interdependencies of key parameters. At the same time it will become clear how the above described methods work in practice. We provide parameter values that yield good performance, as we will show on several video sequences (see Section 3.5 for results with our chosen parameter set) and some example images illustrate how a changed parameter effects the result. Finally Section 3.4.11 gives a short summary.

3.4.1 ε Width of the Band around the Contour

The Heaviside step function H_ε and Dirac delta function δ_ε are used to determine the pixels in the foreground and background and on the contour $\mathbf{C}$ respectively. ε specifies

the degree of smoothing applied to both functions (see Fig. 3.2) and could be described as uncertainty on the contour: Instead of the exact contour, a band of width 2ε around the contour is employed and pixels located farther from the contour are assigned a lower weight. Incidentally, smoothing δ_ε is necessary since the image is discretized into pixels, whereas the contour is continuous.



(a) (b)

Figure 3.2: (a) (Red): Heaviside step function; (Blue): H_ε smoothed with $\varepsilon = 3$. (b) (Red): Dirac delta function; (Blue): δ_ε smoothed with $\varepsilon = 3$.

A greater ε has the effect of smoothing the contour and slowing down the evolution, thereby making it more robust to similar background. At the same time this leads to less accurate segmentations (Fig. 3.3), which in turn leads to less accurate appearance models. Moreover the wider the band, the more pixels are employed in the optimizations of segmentation and tracking and thus the higher the runtime. A lower ε on the other hand means that very few pixels are used, the contour is more unsteady and tracking is less robust to similar background (Fig. 3.4). We set $\varepsilon = 3$.

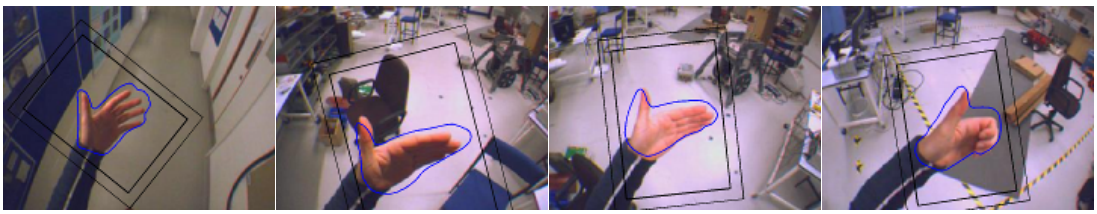


Figure 3.3: Result for $\varepsilon = 12$: The contour is smoothed so much it cannot encase the object in detail and the registration fails to register the hand accurately. The tracker still successfully tracks the hand through the whole sequence.

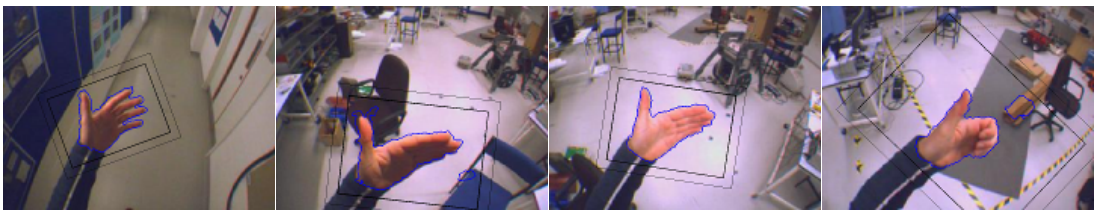


Figure 3.4: Result for $\varepsilon = 1.5$: The contour is not smooth and the tracker fails after 96% of the sequence while there are objects in the background that contain foreground colours.

3.4.2 τ Timestep

The level set Φ is evolved using a steepest-ascent algorithm with the timestep τ. A larger timestep results in faster evolution, and thus enables the contour to adapt to shape changes more quickly. At the same time this decreases robustness since background can also be enclosed very quickly, as can be seen in Fig. 3.5. We used $\tau = 1$ for our result sequences.

Figure 3.5: Result for $\tau = 2$: The segmentation adapts to shape changes quickly but is not robust to foreground colours in the background and fails after 37% of the sequence.

3.4.3 Number of Iterations

The level set is first evolved to obtain a segmentation of the target object. In this case the initialization usually consists of a bounding box which is only a rough estimate of the object's shape. In the subsequent frames, when the object is re-segmented, the contour is already similar to the current shape as shape changes only happen gradually. Consequently the number of iterations necessary for the first segmentation is significantly higher than for re-segmentation.

The number of iterations during re-segmentation is important for robustness, because a better contour can be tracked more easily and will keep the appearance models accurate. If it is chosen too low the contour may not be able to follow the object's shape deformations (Fig. 3.6). If it is chosen too high however the contour will easily enclose background pixels if they have the same colour as the tracked object (Fig. 3.7). Thus this parameter should be chosen depending on the application, *i.e.* how fast the target object can change its shape, and the number of frames per second.

We use $it_1 = \frac{600}{\tau}$ iterations for the initial segmentation and $it_2 = 1$ iteration for re-segmentation. The timestep τ effects the number of necessary iterations, since a

Figure 3.6: Result for $it_2 = 0$: The contour cannot adapt to shape changes and loses the object after 3% of the sequence, when its shape has become very different.

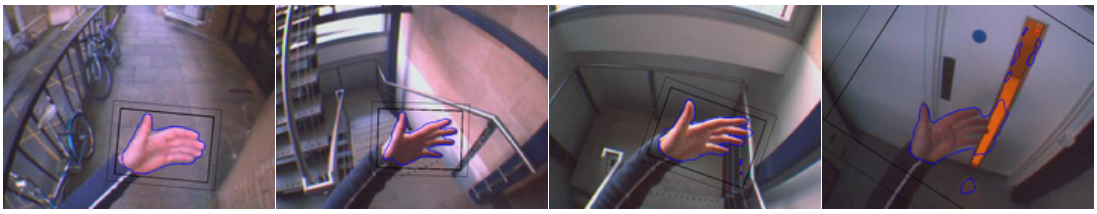


Figure 3.7: Result for $it_2 = 2$: The segmentation adapts to shape changes quickly but is not robust to background with similar colours like the object, as for $\tau = 2$ (*c.f.* Fig. 3.5). The tracker fails after 37% of the sequence.

larger step size results in faster evolution. Note that while the number of iterations is a discrete value, τ does not need to be an integer and can thus be used for achieving a segmentation that evolves more slowly, if necessary. Varying τ or it_2 has very similar effects, however since the contour moves with each iteration the two are not identical.

3.4.4 σ Weight of the Geometric Prior

σ was introduced by the geometric prior (3.8) and $\frac{1}{\sigma^2}$ denotes its relative weight in the optimization of the segmentation. It only has an effect on the level set during segmentation: the level set embedding function remains close to a signed distance function and thus numerically stable. [LXGF05] showed that $\frac{\tau}{\sigma^2} < 0.25$ must be satisfied. Its visible effects on the contour are minimal, however the contour is smoother and evolves more slowly when the geometric prior has a higher weight (Fig. 3.8). We use $\sigma = \sqrt{50}$.

3.4.5 $\min(P_f), \min(P_b)$ Minimum Probability

The probabilities of a pixel belonging to the foreground and background, P_f and P_b, are computed based on colour histograms and normalized so that $P_f + P_b = 1$. If a colour appears in the foreground but not in the background P_f will be zero and vice versa. For the computation of $\Delta\mathbf{p}$ with eq. (3.21) however P_f and P_b are used in the denominator. We therefore need to restrict P_f and P_b to a minimum value greater than zero.

For choosing this minimum value it is important to consider that a smaller value has a larger effect on the result, since P_f and P_b appear in the denominator. In practice each pixel contributes a small transformation to $\Delta\mathbf{p}$. Eq. (3.21) is evaluated for every point $\mathbf{x}$ in the band around the contour. Each pixel with a low foreground probability, but located in the foreground, contributes a warp that moves it towards the outside of the contour and vice versa. Thus $\min(P_f), \min(P_b)$ have influence on the tracking step size. They must obviously not be chosen too high, since foreground and background need to be distinguishable. If chosen smaller however the tracking step size will become larger. The following section will discuss the step size in detail.

There is another effect that can be achieved with $\min(P_f), \min(P_b)$: During tracking shape changes will have the effect that pixels very near the object's (real) contour

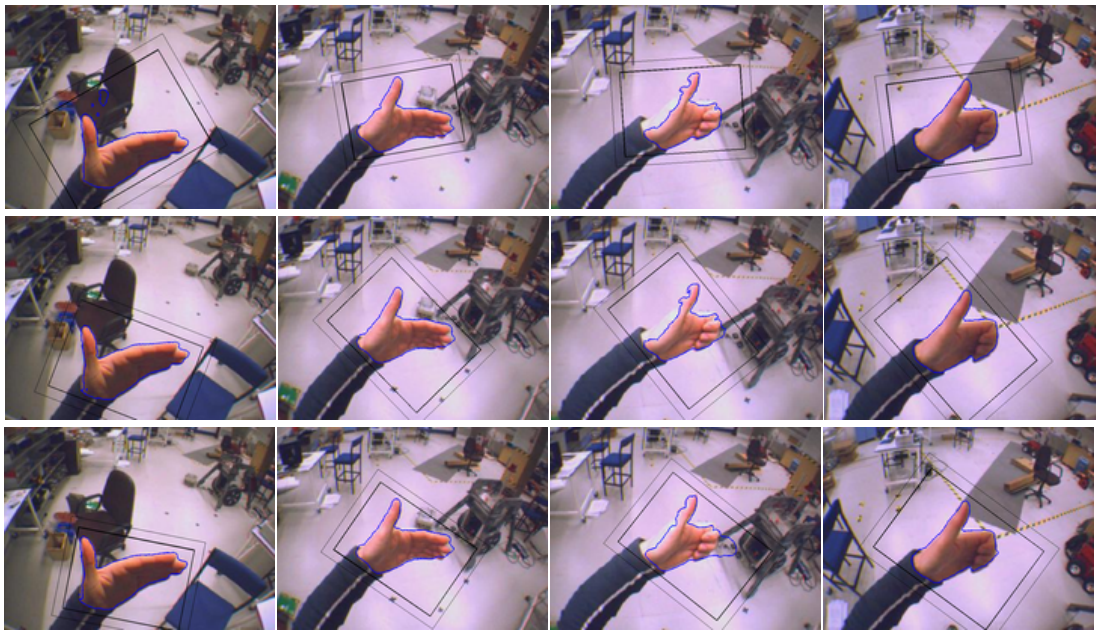


Figure 3.8: (Top): Result for $\sigma = \sqrt{25}$: the contour adapts quickly but is not robust to similar colours. (Middle): Result for $\sigma = \sqrt{50}$ (the value used for our results): the segmentation is both robust and quick. (Bottom): Result for $\sigma = \sqrt{75}$: the segmentation is robust, but slow. However, the hand is successfully tracked through the whole sequence in all three cases.

are inside the tracked contour and thus contribute their colour to the foreground model. Over a large number of frames this can cause the contour to grow and finally to enclose background pixels (Fig. 3.9). By setting $\min(P_b)$ a little lower than $\min(P_f)$ background pixels inside the contour will have a larger effect than foreground pixels outside, resulting in a small inward force that keeps the contour tighter around the tracked object and thus results in higher robustness. If the difference is too high however the contour typically shrinks gradually during lighting changes (Fig. 3.10).

In our implementation we use $\min(P_f) = 1 \cdot 10^{-4}$ and $\min(P_b) = 5 \cdot 10^{-5}$.

Figure 3.9: Result for $\min(P_f) = \min(P_b) = 10^{-4}$: The tracker works well while foreground and background are very different, but fails after 16% of the sequence, when it encounters background similar to the foreground.

Figure 3.10: Result for $\min(P_f) = 10^{-4}$, $\min(P_b) = 10^{-5}$: The tracker fails during lighting changes after 7% of the sequence.

3.4.6 ε_p and Normalization of $\Delta\mathbf{p}$

The tracking algorithm's step size is determined automatically: Each pixel contributes a warp and a lower probability P_f in the foreground or P_b in the background results in a larger step. Moreover the step size depends on the number of pixels contributing since it is not fixed but depends on the length of the contour and on ε. The algorithm converges when pixels' probabilities match their position and the step size has thus become very small, $\|\Delta\mathbf{p}\| \leq \varepsilon_p$.

A large step size can cause the object frame to bounce around the object and the tracking algorithm not to converge. A small step size on the other hand will naturally cause a slow convergence. We found that for some parameter sets it can be useful to reduce the step size in order to achieve better and faster convergence. To this end we normalize it depending on the number of used pixels:

$$\Delta\mathbf{p} \leftarrow f_p \frac{\Delta\mathbf{p}}{\sum_i \delta_\varepsilon(\Phi(\mathbf{x}_i))} \,. \tag{3.25}$$

The factor f_p retains a step size large enough to yield fast results. In this way we obtain a step size that is independent of the length of the contour and reduces it such that fast convergence can be achieved.

For $\varepsilon = 3$ we do not use the normalization and $\varepsilon_p = 0.08$. For $\varepsilon = 12$ we use $f_p = 20$, $\varepsilon_p = 0.05$ and $it_2 = 2$. We set the minimum number of tracking steps to 3 and the maximum to 50.

3.4.7 α_f, α_b Online Learning

The learning rates α_f and α_b specify the rate at which the foreground and background models are adapted, which is necessary for example when lighting conditions change. If they are too low the tracker cannot adapt and will lose the object (Fig. 3.11). On the other hand if they are too high the tracker can easily learn background colours while the contour is still adapting to a changing shape (Fig. 3.12).

Figure 3.11: Result for $\alpha_f = 0$ and $\alpha_b = 0$: Without online learning the tracker is not able to recover from the inexact initial segmentation and fails after 0.3% of the sequence.

Figure 3.12: Result for $\alpha_f = 0.2$ and $\alpha_b = 0.25$: After 27% of the sequence the tracker learns a background colour as foreground colour and fails.

3.4.8 Size of the Object Frame

The size of the object frame determines the scale at which the object is examined. If the object frame is too small an object can obviously not be segmented well. If it is larger the runtime, which depends on the number of pixels in the narrow band around the contour, rises as well. We use an object frame with approximately 8000 pixels, depending on the initial object frame's width-to-height ratio.

3.4.9 Number of Bins in Colour Histogram

The number of bins used for the colour histograms determines the number of colours that can be distinguished by the framework. Both a low (Fig. 3.13) and a high (Fig. 3.14) number of bins result in lower robustness, since similar colours cannot be separated or very similar colours are already separated, respectively. We use 32 bins per channel.

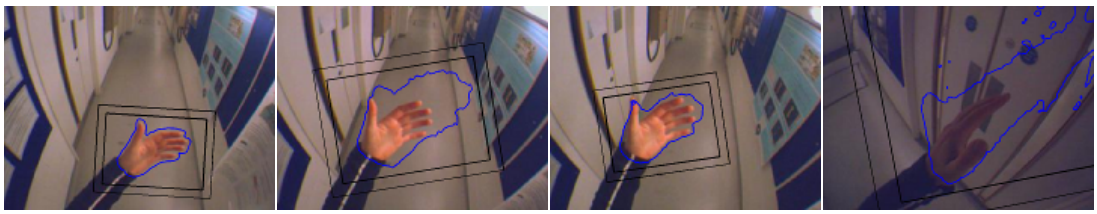


Figure 3.13: Result for $\#bins = 16$: The tracker has difficulties with background similar to the foreground, but is often able to recover before it fails after 53% of the sequence.

Figure 3.14: Result for $\#bins = 64$: The tracker is not able to handle lighting changes and fails after 6% of the sequence.

3.4.10 Colour Spaces

Colour spaces are mathematical representations of sets of colours. We compare three different colour spaces: RGB uses the three primary additive colours red, green and blue. YCbCr instead uses one value for intensity (Y) and two for colour (Cb,Cr), which has the advantage that only one value has to be modified in order to change the intensity [Jac01]. L*a*b* (or CIELAB) also has a value for intensity (L*) and two for colour (a*,b*), but unlike RGB and YCbCr it is colorimetric, *i.e.* colours that look the same to a human are encoded identically [GW08]. Fig. 3.15 shows that all three colour spaces yield very similar results with our implementation. For our results we use the L*a*b* colour space.

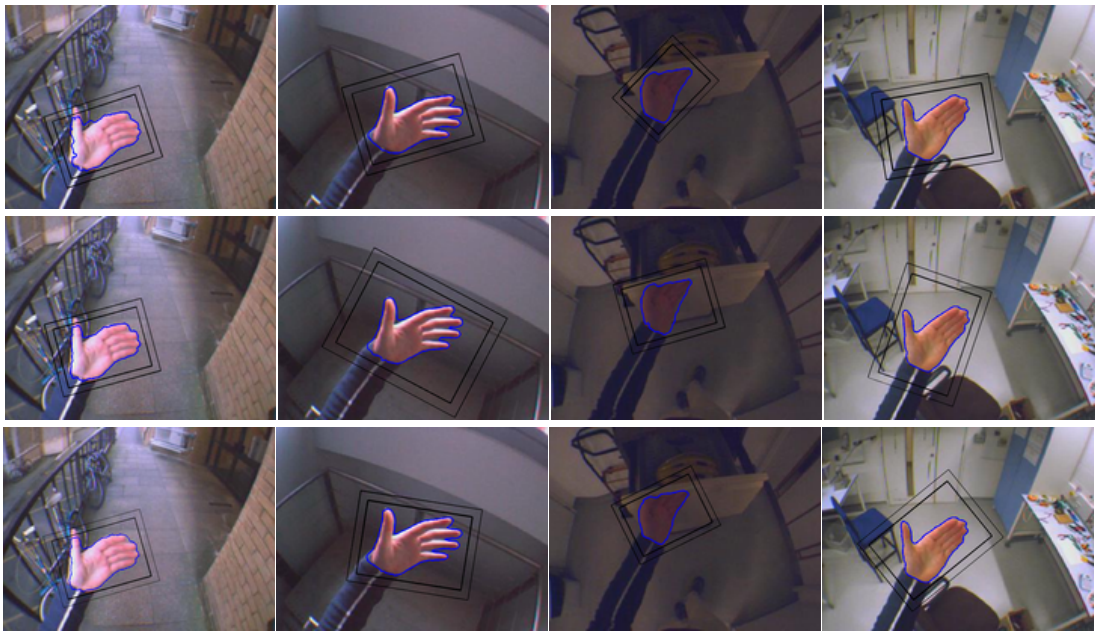


Figure 3.15: Results for different colour spaces: (Top): RGB; (Middle): YCbCr; (Bottom): L*a*b*. The tracker is successful for all three colour spaces.

Interestingly the tracking approach also works well without L*a*b*'s intensity channel, the robustness is only slightly decreased (Fig. 3.16). Using only the colour channels has the advantage of reducing the size of the colour histograms from 32^3 bins to 32^2 bins.

Figure 3.16: Result for L*a*b* without the intensity channel L* (and $\lambda = 1$, *c.f.* Section 4.2). The hand is tracked through the whole sequence successfully.

3.4.11 Overview

Table 3.1 gives a short overview of the parameters described above.

Table 3.1: Parameter set used to obtain the results in Section 3.5.

Parameter	*Suggested Value*	*Description, Effects (of increasing the parameter)*	*Interdependencies*
ε	3	Width of band around contour - Smooths contour - Slows down level set evolution - Increases runtime	τ, it_2, σ
τ	1	Segmentation step size - Speeds up evolution - Decreases accuracy of segmentation	$\varepsilon, it_2, \sigma$
it_2	1	# iterations for re-segmentation - Increases accuracy of segmentation - Decreases robustness to similar objects	$\varepsilon, \tau, \sigma$
σ	$\sqrt{50}$	Weight of geometric prior - Smooths contour - Slows down level set evolution	ε, τ, it_2
$min(P_f)$, $min(P_b)$	$1 \cdot 10^{-4}$, $5 \cdot 10^{-5}$	Minimum foreground and background probability - Decreases tracking step size - Inward force on contour	ε_p
ε_p	0.08	Tracking convergence - Decreases accuracy of registration - Decreases runtime	$min(P_f)$, $min(P_b)$
α_f, α_b	0.02, 0.025	Online learning rate - Increases robustness to lighting changes - Decreases robustness to similar objects	

3.5 Results

In this section we compare our qualitative results to the ones presented in [BR08]. We use two long and challenging video sequences generously provided by the authors of [BR08]. One is of a hand filmed from a head-mounted camera past a background with similar appearance to the object and moreover containing considerable lighting changes. It consists of 4660 frames (in the following called HAND). The second is of a person jumping around and contains fast movements and motion blur and consists of 1587 frames (in the following called JUMP). Both sequences were captured with approx. 25 fps.

Fig. 3.17 shows our results for the HAND sequence compared to the results by [BR08]. As can be seen our implementation sometimes includes background areas, which Bibby and Reid's implementation does not (13th frame). (We will improve our results with respect to foreground colours in the background in Section 4.2.) In return our contours do not shrink as much during lighting changes (3rd frame). Fig. 3.18 shows our results for the JUMP sequence. Our contours are less smooth, but more accurate. Overall both results are very similar, our implementation yields results of comparable quality to [BR08].

Please refer to appendix B for our tracking results for several other sequences.

3.6 Discussion

In this chapter we have introduced the combined segmentation and tracking framework by Bibby and Reid [BR08] and conducted a detailed analysis of the key parameters.

Only pixels in a narrow band of width 2ε around the contour (determined with δ_ε, a smoothed Dirac delta function) are used and an individual pixel's contribution to the optimization decreases with its distance from the contour. The width ε influences the speed of the level set evolution and the contour's smoothness, as well as the overall runtime. The target object is segmented by optimizing the segmentation's probability with a gradient-ascent algorithm where the timestep τ affects the speed of the level set evolution and the contour's accuracy. A geometric prior with weight $\frac{1}{\sqrt{\sigma}}$ keeps the level set embedding function in a numerically stable shape and also increases the contour's smoothness. Between frames the object's position is determined by a rigid registration, which is performed using a Gauss-Newton optimization. Its step size is determined automatically and the algorithm converges when the step size is small enough ($\Delta\mathbf{p} \leq \varepsilon_p$). The algorithm exhibits reliable convergence, but reducing the step size can improve, *i.e.* accelerate convergence for some parameter sets. A higher convergence criterion ε_p decreases the runtime.

The robustness of the tracking framework depends on a well-balanced parameter set. An accurate contour increases robustness, since it increases the accurateness of the appearance models. On the other hand a faster evolution or more re-segmentation iterations can decrease robustness, since background pixels with foreground colours

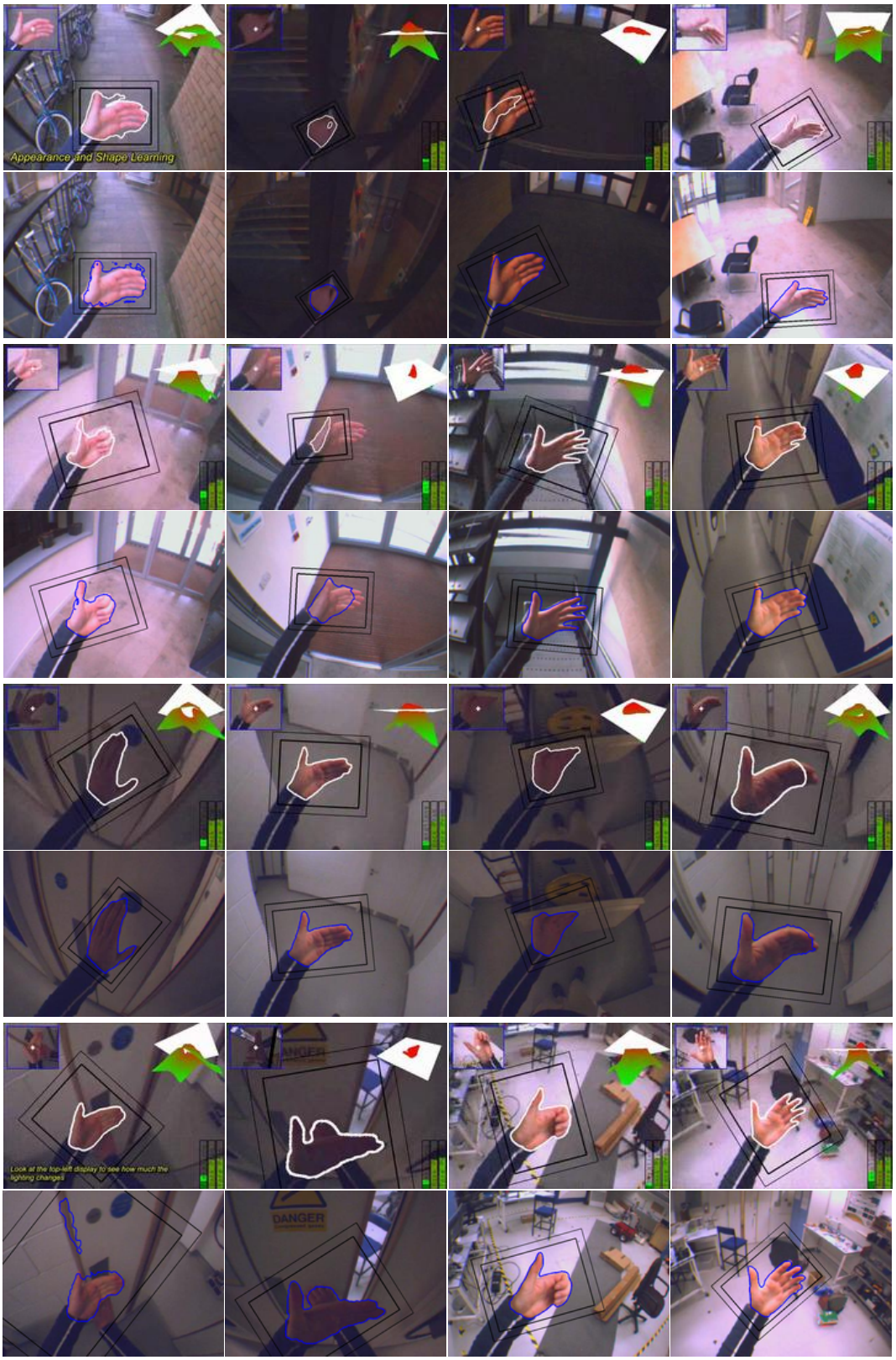


Figure 3.17: Result for seq. HAND (4660 frames). (Top rows): Results from [BR08] (contour in white); (Bottom rows): Our results (contour in blue).

Figure 3.18: Result for seq. JUMP (1587 frames). (Top rows): Results from [BR08] (contour in white); (Bottom rows): Our results (contour in blue).

can be enclosed more quickly. The runtime depends on the width of the narrow band and the convergence criterion.

Our results show that our implementation yields results of comparable quality to the ones presented in [BR08]. The approach is robust to fast movements of the target object, motion blur, sensor noise and a low frame rate. Challenges are lighting changes, background with colours similar to the object and inaccurate initializations, all are handled well. In the next chapter we will present several enhancements to the framework.

Chapter 4

Extensions

In this chapter we present some extensions we developed for the above described approach. We first examine the case of objects partially leaving the image frame in Section 4.1. Second, the segmentation approach depends highly on its initialization, since the appearance models are built iteratively during the first segmentation. If the initialization includes much background, the object might not be segmented well, since some background colours dominate the initialized foreground. In order to gain more robustness and enable an automatic initialization, *e.g.* with detector bounding boxes, we investigate the use of a smoothness constraint on the contour (Section 4.2). So far we used a transformation with translation, scale and rotation. We present several other transformations and their effect on the quality of the registration in Section 4.3. Furthermore we develop an approach for incorporating depth information into the appearance models (Section 4.4). In preparation for tracking pedestrians (*c.f.* Chapter 5) we show how the tracking of several objects simultaneously, including occlusion resolve, can be done (Section 4.5). Finally we discuss the presented extensions and possible future work in Section 4.6.

4.1 Tracking Partially Visible Objects

It frequently occurs that a tracked object is partially "occluded" by the image boundaries for some time during its movement. In this case the object frame will partially leave the image as well and the tracker has to deal with the missing image information.

A very simple solution is to set P_b high and P_f low, *i.e.* to effectively classify this area as background, in order to keep tracking only the visible part of the object. Although the tracking algorithm is robust to this sudden shape deformation, it has the disadvantage of losing all information about the object's size and shape. The contour is pushed onto the visible part of the object and shape adaptation will also adapt to the visible part and thus in time classify the image border as the object's border. When the object is completely visible again later the contour needs to recover.

Our method deals with partial visibility without losing shape information: We set

$P_b = P_f$ in areas without image information, which has the effect of not taking this area into account. For segmentation (eq. (3.12)) as well as tracking (eq. (3.21)) this is equivalent to setting $\delta_\varepsilon(\Phi)$ to zero as both equations contain the term $\delta_\varepsilon(\Phi)(P_f - P_b)$, which additionally results in computational savings. This is because the rigid registration operates only on a narrow band of pixels around the contour, which is determined by $\delta_\varepsilon(\Phi)$. In addition we also do not consider pixels in a margin of two pixels at the image boundaries. This results in tracking and re-segmenting only the visible part of the contour. Thus, if an object becomes completely visible again, the shape will fit immediately.

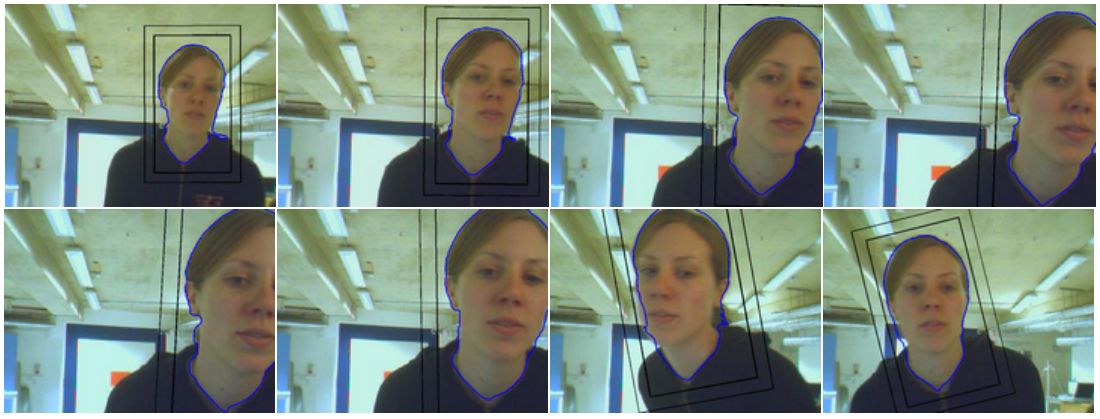


Figure 4.1: The tracked head is only partially visible for more than 30 frames, but the contour stays exact and does not need to recover.

4.2 Smoothness Constraint on the Contour

The tracking framework depends on a sufficiently accurate initial segmentation, which in turn depends on its initialization. Fig. 4.2 illustrates how vulnerable the segmentation framework is to the size and position of the initialization box. Moreover during tracking background areas that contain foreground colours are sometimes segmented

(a) Small initialization box (red), as was used in our results in Chapter 3.

(b) Big initialization box (red).

Figure 4.2: An initialisation box that is only slightly bigger than the hand yields a segmentation so inaccurate, that the tracker fails immediately.

(*c.f.* Fig. 3.17, 13th frame). Although our tracker is generally robust to such background areas and does not lose the object, the resulting contours and object frame positions in the meantime are not desirable.

To counteract this problem we add a constraint on the smoothness of the contour to our shape optimization (similar to ht length term in [LXGF05], but independent of edges in the image). The curvature of the level set is given by $\kappa = \mathrm{div}\frac{\nabla\Phi}{|\nabla\Phi|}$ [Set99]. We add a term that penalizes the curvature extremes of the contour (or more exactly the band around the contour) to the variational level set formulation. The optimization of the segmentation (eq. 3.12) thus becomes:

$$\frac{\partial P(\Phi,\mathbf{p}|\Omega)}{\partial\Phi} = \frac{\delta_\varepsilon(\Phi)(P_f - P_b)}{P(\mathbf{x}|\Phi,\mathbf{p},\mathbf{y})} - \frac{1}{\sigma^2}\left[\nabla^2(\Phi) - \mathrm{div}\left(\frac{\nabla\Phi}{|\nabla\Phi|}\right)\right] + \lambda\delta_\varepsilon(\Phi)\mathrm{div}\left(\frac{\nabla\Phi}{|\nabla\Phi|}\right) \quad (4.1)$$

where λ is the weight of the smoothness constraint.

Smoothing the contour, *i.e.* penalizing the length of the contour, slows down the expansion to new pixels as well as speeds up the exclusion of solitary foreground pixels without generally shrinking the contour. This is because the contour evolves more evenly. Fig. 4.3 shows that the adapted segmentation framework yields much smoother and more accurate initial contours and thereby alleviates the dependency on an advantageous initialization.

(a) Small initialization box (red). (b) Big initialization box (red).

Figure 4.3: With the smoothness constraint ($\lambda = 3$) the segmentation is able to overcome the background areas around the hand. The tracker is successful.

Moreover the smoothness constraint increases the robustness to similar background areas, while only slightly decreasing the contour's accurateness (Fig. 4.4). The *CAVIAR* sequence (part of the seq. *ThreePastShop1cor* from the caviar data set [Fis04], 850 frames), is particularly challenging since the right person is in front of a similar background in the first frame. The contour thus needs to leave this small background area when the person moves away from it. With smoothness constraint the contour cannot grow in direction of the background as fast and is successfully dragged away from it by the moving person (Fig. 4.5).

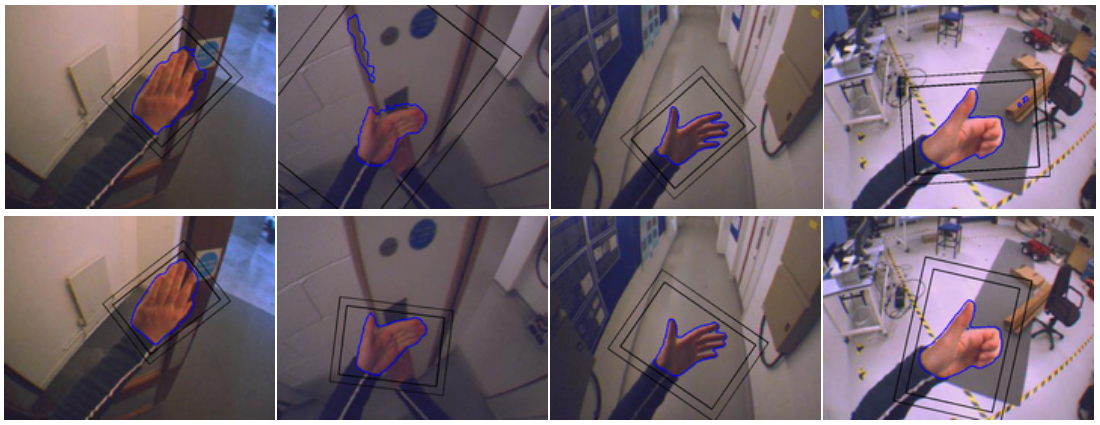


Figure 4.4: Result for seq. HAND: (Top): $\lambda = 0$; (Bottom): $\lambda = 3$. With the smoothness constraint the tracker is more robust to similar background areas. The contour is only slightly less accurate (3rd frame).

Figure 4.5: Results for seq. *CAVIAR*: (Top) $\lambda = 0$: Without the smoothness constraint the contours are more detailed and accurate, but over time the background or the middle person are incorporated into the contour. (Bottom) $\lambda = 3$: With the smoothness constraint both persons are tracked successfully through the whole sequence.

4.3 Transformations

So far we used a transformation containing translation, scale and rotation as in [BR08][1]. In principle any transformation that forms a group can be employed in our tracking framework. This means the only restriction in fact is that the transformation needs to be invertible (as the warp is inverted during the rigid registration). A transformation with more degrees of freedom can explain more complex transformations of the shape

[1] Not to confuse with the term *rigid transformation* even if [BR08] speak of *rigid registration*.

with the rigid registration. A transformation with fewer degrees of freedom in turn needs to explain these transformations with shape adaptation.

In this section we explore three further transformations: translation and scale (without rotation) in Section 4.3.1, affine transformations in Section 4.3.2 and projective transformations in Section 4.3.3. In Section 4.3.4 we compare all four transformations.

4.3.1 Translation, Scale

The transformation for translation and scale has three parameters: translation in x-direction, translation in y-direction and a (proportional) scale factor.

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}, \quad \mathbf{W}(\mathbf{x}_i, \mathbf{p}) = \begin{bmatrix} 1+p_1 & 0 & p_2 \\ 0 & 1+p_1 & p_3 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}, \text{ with } \mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}. \tag{4.2}$$

Thus

$$\frac{\partial \mathbf{W}}{\partial \Delta \mathbf{p}} = \begin{pmatrix} x_i & 1 & 0 \\ y_i & 0 & 1 \end{pmatrix}. \tag{4.3}$$

Fig. 4.6 shows results for tracking with translation and scale. The movement of the hand could often be described by a rotation of the object frame, without that possibility those movements have to be explained by shape adaptation. This often results in an inaccurate segmentation which decreases robustness. (More re-segmentation iterations again have the effect of enclosing similar background). If the target object's movement can be described well without rotation of the object frame on the other hand, this transformation is well suited. This is for example the case for pedestrians. When tracking pedestrians a rotated box might not even be useful for subsequent processing of the tracking result. Moreover since pedestrians are not able to perform a movement that would require a rotation of the object frame, this restriction can even increase robustness (see Fig. 4.7).

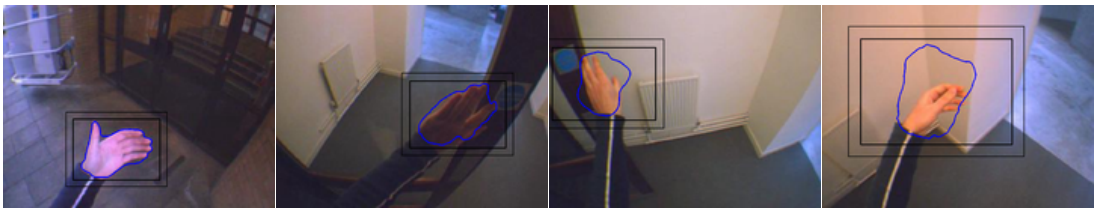


Figure 4.6: Result for tracking with translation and scale: The re-segmentation needs to cover all the perceived shape changes caused by the hand's movement. This results in a lower robustness and the hand is lost after 15% of the sequence.

Figure 4.7: Results for tracking with translation and scale: (Top) $\lambda = 0$: The contours are more robust to the background than with rotation, but still incorporate background pixels over time. (Bottom) $\lambda = 3$: The two persons are tracked successfully through the whole sequence as for translation, scale and rotation with $\lambda = 3$ (*c.f.* Fig. 4.5, B.3).

4.3.2 Affine Transformation

The affine transformation has six parameters which in combination result in translation, scale, shear (each in x and y-direction) and rotation. It preserves parallelism of lines.

$$\mathbf{p} = \begin{bmatrix} p_1 \\ \vdots \\ p_6 \end{bmatrix}, \quad \mathbf{W}(\mathbf{x}_i, \mathbf{p}) = \begin{bmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{4.4}$$

Thus

$$\frac{\partial \mathbf{W}}{\partial \Delta \mathbf{p}} = \begin{pmatrix} x_i & 0 & y_i & 0 & 1 & 0 \\ 0 & x_i & 0 & y_i & 0 & 1 \end{pmatrix}. \tag{4.5}$$

Fig. 4.8 illustrates how the contour and with it the object frame are transformed. Since an affine transformation has more degrees of freedom shape deformations can now partly be explained by the transformation, not only by re-segmentation. The amount of drift is increasing (affine transformations contain *e.g.* non-proportional scaling) and robustness is decreasing when the object frame is distorted. An improved drift correction algorithm could correct this, however the extension to an affine transformation did not improve the quality of the tracked contours in our experiments.

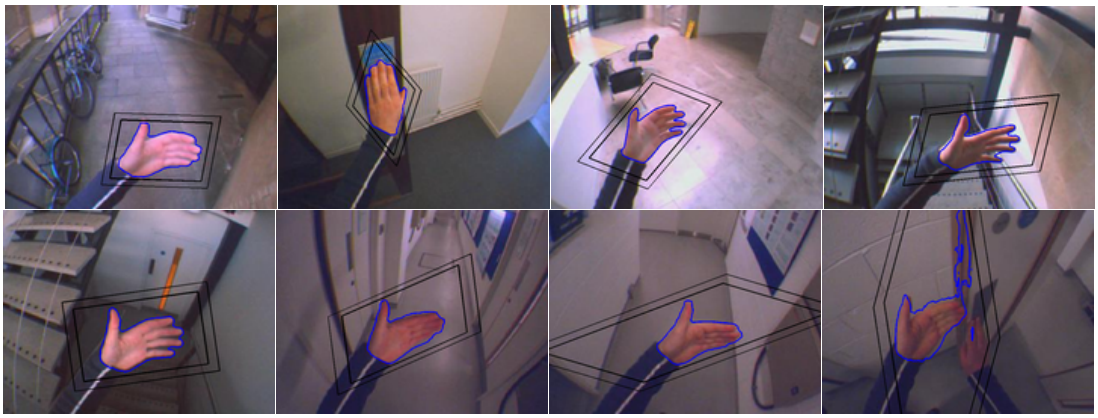


Figure 4.8: Result for tracking with an affine transformation: The contour is tracked well, but the object frame is distorted over time and includes a large amount of background pixels. The tracker fails after 76% of the sequence.

4.3.3 Projective Transformation

The projective transformation or homography has eight parameters, and corresponds to the changes in an image that are perceived when the viewpoint changes. It preserves lines.

$$\mathbf{p} = \begin{bmatrix} p_1 \\ \vdots \\ p_8 \end{bmatrix}, \quad \mathbf{W}(\mathbf{x}_i, \mathbf{p}) = \frac{1}{p_7 + p_8 + 1} \begin{bmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{4.6}$$

Thus

$$\frac{\partial \mathbf{W}}{\partial \Delta \mathbf{p}} = \begin{pmatrix} x_i & 0 & y_i & 0 & 1 & 0 & -x_i^2 & -x_i y_i \\ 0 & x_i & 0 & y_i & 0 & 1 & -x_i y_i & -y_i^2 \end{pmatrix} \tag{4.7}$$

where $p_1..p_8$ are set to 0, which can be assumed since the used images are already warped and we thus have $\Delta \mathbf{p} = 0$ at the beginning of each tracking step.

Fig. 4.9 shows results for tracking with a perspective transformation. Having even more degrees of freedom than an affine transformation can easily result in the contour being distorted so much that the homography degenerates and the object frame does not even remain a convex quadrilateral.

Figure 4.9: Result for tracking with a perspective transformation: The homography degenerates and the tracker fails after 3% of the sequence.

4.3.4 Comparison

Our results show that a transformation with fewer degrees of freedom is more robust, provided it can describe the object's possible movements on the image plane well enough. Transformations with translation and scale are well suited for tracking pedestrians, but not for tracking *e.g.* a hand or a head. The tracker using translation, scale and rotation performed well for all tested sequences. Affine or projective transformations can track the contour well but degenerate over time and are of no benefit for the quality of the contour compared to translation, scale and rotation.

In Chapter 6 we will show how a perspective transformation can be constrained such that only physically plausible transformations are allowed, while at the same time conclusions about the object's movement in 3D-space can be drawn.

4.4 Depth Information

Until now we only used colour for the foreground and background appearance models. We found that when tracking pedestrians this can yield rather unreliable segmentations since other people or background structures often contain similar colours. Moreover in this case the registration is often not able to follow the pedestrian's movement since although the person moved away from the contour the colours inside the contour still fit the appearance model. Fig. 4.10 (top) shows an example of a failed tracking attempt. The tracker is not able to deal with the shape deformation, shrinks and even slides off when the background colours are too similar.

We therefore extend the approach by also including stereo depth information (see Chapter 2 for information about stereo depth maps). For segmentation, we use the median depth of the foreground area. Unlike the colour distribution, the median depth will not stay the same during the following frames. For tracking, we therefore use a simple motion model which computes an expected distance range for each pedestrian according to the last median depth and a maximum velocity. Each depth value in the image is then assigned a probability according to a Gaussian distribution around the median depth or the expected depth, respectively. The two probabilities for colour and depth are individually normalized (*c.f.* (3.9)) and then merged:

$$P_i = (1-\alpha)P_{i,colour} + \alpha P_{i,depth}, \quad i \in \{f,b\}\ , \tag{4.8}$$

where the weighting factor α is set to 0.1 in all our experiments.

Fig. 4.10 (middle) shows that depth information can stabilize position and size of the tracked contour. However the tracker with depth information performed with variable success. Other persons near the tracked one, which are at the same distance, can increase the foreground probability in this area and "pull" the contour in direction of this other person and in this way decrease the tracker's robustness. Moreover stereo depth maps do not provide information for all pixels and are not reliable for objects very near the cameras. The object's boundaries thus fluctuate more than the boundaries

Figure 4.10: (Top): Result without depth information; (Middle): Result with depth information; (Bottom): Used disparity maps. The tracked pedestrian is dressed in black and white, the same as her background in this scene. This example clearly shows that depth information can stabilize position as well as size of the tracked contour.

in a colour image and hinder a rigid registration of a given contour in a depth map. We therefore assign only a small weight to the probabilities computed with depth information. This has the effect that colour is still the main criterion for tracking and depth information only provides hints for inconclusive pixels.

4.5 Tracking Multiple Objects Simultaneously

With a view to pedestrian tracking it is particularly interesting to be able to track multiple persons in the same video sequence. Since different pedestrians move independently we assign individual level sets and appearance models to each. Even if overlaps occur the level sets will not interact directly (as *e.g.* in [BW04]). The tracking framework is generally robust to partial occlusions, as long as the tracked object is still mostly visible. When depth information is available these partial occlusions can be resolved by registering the nearer objects first. The occupied pixels are then masked out such that they are not regarded by the remaining objects, as was described in Section 4.1 for the image borders.

4.6 Discussion

In this chapter we presented and evaluated several extensions to the original segmentation and tracking approach. First objects can continue to be tracked even if they partially leave the image frame by ignoring the part of the narrow band for which no image information is available. This has the additional advantage that the shape information is not lost if objects reappear. We then added a smoothness constraint to the segmentation formulation, which increases robustness to inexact initializations and to similar background areas. We compared several transformations and found that more complex transformations decrease robustness without increasing the quality of the tracked contour. A transformation with translation, scale and rotation performed well for all tested video sequences. For tracking pedestrians simply translation and scale is also adequate. Last we incorporated depth information into the appearance models to increase robustness to similar colours in the background. We found that the tracker with depth information performed with variable success, since objects near the target object and with the same distance can in turn decrease robustness. Multiple objects are tracked with individual levels sets and appearance models. Occlusions are then handled by registering the nearer objects first and ignoring the occupied pixels for the remaining objects, as for pixels outside the image frame. Chapter 5 describes our framework for tracking pedestrians.

Further enhancements could incorporate more information into the appearance model, e.g. texture or optical flow. In addition the convergence of the tracker not necessarily corresponds to successful registration of the object's position. Measures for the tracker's success could be used for a dynamic adaptation of the number of re-segmentation iterations and online learning rate to the current quality of the contour.

Chapter 5

Integration with Detection-Based Tracking

In this chapter, we address the problem of multi-person tracking with a camera mounted on top of a moving vehicle, *e.g*. a mobile robot. This task is very challenging, since multiple persons may appear or emerge from occlusions at every frame and need to be detected. Since background modelling [SG99] is no longer applicable in a mobile scenario, this is typically done using visual object detectors [DWSP09]. Consequently, tracking-by-detection has become the dominant paradigm for such applications [OTdF$^+$04, ARS08, LSV08, ELSV09, WN07, HWN08]. In this framework, a generic person detector is applied to every frame of the input video sequence, and the resulting detections are associated to tracks. This leads to challenging data association problems, since the detections may themselves be noisy, containing false positives and misaligned detection bounding boxes [DWSP09]. Several approaches have been proposed to address this issue by optimizing over a larger temporal window using model selection [LSV08], network flow optimization [ZN08], or hierarchical [HWN08] or MCMC data association [ZNW08].

Intuitively, this complex data association seems to be at least to some degree an overkill. Once we have detected a person in one frame, we know their appearance and should be able to use this information in order to disambiguate future data associations. This has been attempted by using person-specific color descriptors (*e.g*. [ARS08, LSV08, ELSV09]) or online-trained classifiers [GGB06]. The difficulty here is however that no precise segmentation is given – the detector bounding boxes contain many background pixels and the persons' limbs may undergo considerable articulations, causing the classifiers to drift. Another problem of tracking systems that only rely on detector input is that they will not work in situations where the detectors themselves fail, *e.g*. when a person gets too close to the camera and is partially occluded by the image borders. [ELSV09] explicitly point out those situations as failure cases of their approach.

We propose to address those problems by complementing the detection-based tracking framework with a robust image-level tracker [MHLE10], *i.e*. the level set tracker

introduced in chapters 3 and 4. In this integration, a high-level tracker only initializes new tracklets from object detections, while the frame-to-frame target following and data association is taken over by the image-based LS tracker. The resulting tracked target locations are then transmitted back to the high-level tracker, where they are integrated into 3D trajectories using physically plausible motion models.

This combination is made possible by the great progress LS segmentation and tracking approaches have made in recent years [CRD07]. The approaches now available can obtain robust tracking performance over long and challenging sequences, as was shown above. In addition, LS trackers can be efficiently implemented using narrow-band techniques, since they need to process only a small part of the image around the tracked contour. However, the targeted integration is far from trivial. The LS tracking framework has originally been developed for following individual targets over time and has mostly been evaluated for tasks where a manual initialization is given [CRD07, BR08]. Here, we need to automatically initialize a large number of tracklets from potentially inaccurate detections. In addition, we need to deal with overlaps and partial occlusions between multiple followed persons, as well as with tracker drift from changing lighting conditions and poor image contrast. Finally, we need to account for cases where a person gets fully occluded for a certain time and comes into view again a few frames later. In this chapter, we show how those challenges can be addressed by a careful interplay of the system components.

In detail, we make the following contributions: (1) We demonstrate how LS trackers can be integrated into a tracking-by-detection framework for robust multi-person tracking. In this integration scheme, the low-level image-based LS tracker and the high-level detection-based tracker are closely cooperating in order to overcome each other's weaknesses. (2) Our approach is based on the idea to track each individual pedestrian by an automatically initialized level set. We develop robust methods for performing this initialization from object detections and show how additional geometric constraints and consistency checks can be integrated into the image-based LS tracker. (3) The tracked person contours in each video frame are automatically converted to 3D world coordinates and are transmitted to the high-level tracker, which integrates the position evidence into a robust multi-hypothesis trajectory estimation approach making use of physical motion models. This high-level tracker is responsible for initializing new tracks, correcting the low-level tracker's predictions when drift occurs, and tracking person identities through occlusions. (4) We experimentally demonstrate that this proposed integration achieves robust multi-person tracking performance in challenging mobile scenarios. In particular, as our approach does not depend on continuous pedestrian detection, it can also continue tracking persons that are only partially visible. (5) Another interesting property of our proposed integration is that it does not require the object detector to be executed for every video frame. This is especially relevant for the deployment on mobile platforms, where real-time performance is crucial and computational resources are notoriously limited. We experimentally investigate at what intervals object detections are still required for robust system-level performance.

This chapter presents our proposed end-to-end tracking framework. First section

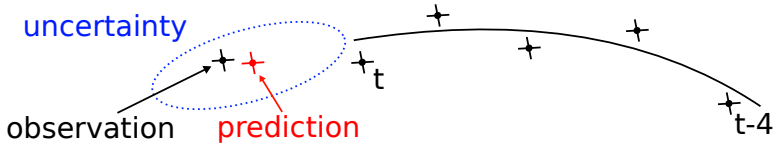


Figure 5.1: Illustration of the Kalman filter's steps: prediction, observation, update

5.1 introduces the basic algorithmic components for trajectory estimation. Section 5.2 then describes the details of the integration and section 5.3 presents experimental results.

5.1 Tracking-by-Detection

For the tracking-by-detection framework, we use a simplified version of the robust multi-hypothesis tracking framework by [LSV08]. In brief, it works as follows: Detected pedestrian locations are converted into 3D world coordinates using the current camera position from SfM together with an estimate of the scene's ground plane. These measurements are collected in a spacetime volume, where they are integrated into multiple competing trajectory hypotheses. The final hypothesis set is then obtained by applying model selection in every frame.

Trajectory Estimation. We model pedestrian trajectories by Kalman filters with a constant-velocity motion model on the ground plane, similar to [ELSV09]. When new observations become available in each frame, we first try to extend existing trajectory hypotheses by the new evidence (Fig. 5.1). In addition, we start a new trajectory hypothesis from each new detection and try to grow it by applying a Kalman filter backwards in time through the spacetime volume of past observations. This step allows us to recover lost tracks and bridge occlusions. As a consequence of this procedure, each detection may end up in several competing trajectory hypotheses.

Model Selection. For each frame, we try to find the subset of trajectory hypotheses that provides the best explanation for the collected observations. This is done by performing model selection in a Minimum Description Length framework, as in [LSV08]. A trajectory's score takes into account the likelihood of the assigned detections under its motion and appearance model (represented as an RGB color histogram). Trajectory hypotheses interact through penalties if they compete for the same detections or if their spacetime footprints overlap. For details of the mathematical formulation we refer to [LSV08].

Assigning Person Identities. As the model selection procedure may choose a different hypothesis set in each frame, a final step is required in order to assign consistent person IDs to the selected trajectories. This is done by maintaining a list of active tracks and assigning trajectories to them based on the overlap of their supporting observations.

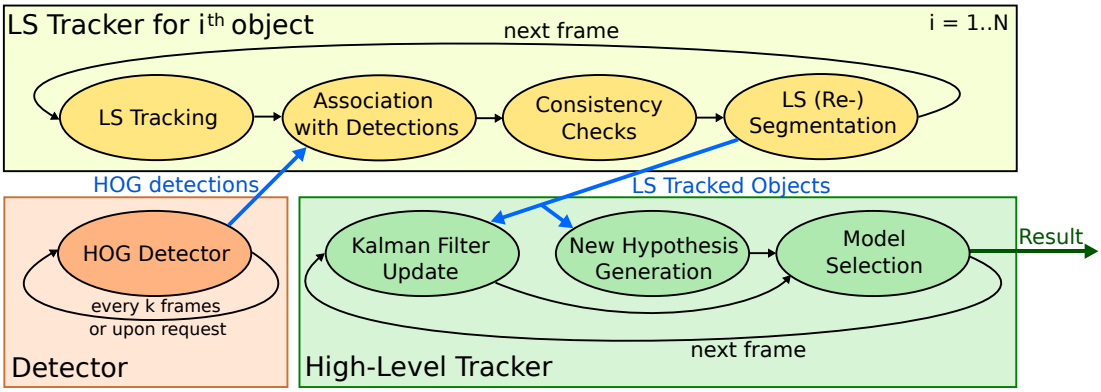


Figure 5.2: System-level view of our proposed end-to-end tracking framework.

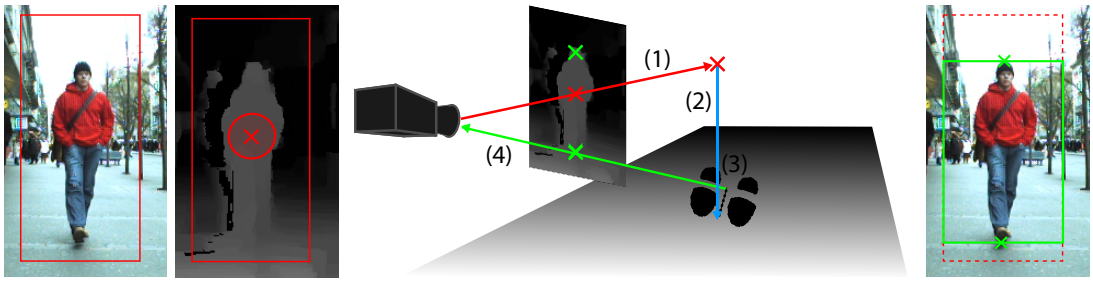
5.2 Integrated Tracking Framework

Fig. 5.2 shows a system-level overview of our proposed integrated tracking framework. The system is initialized by detections from a pedestrian detector. For each detected person, an independent LS tracker (a *tracklet*) is initialized, which follows this person's motion in the image space. The LS tracker is kept robust through a series of consistency checks and transmits the tracked person's bounding box to the high-level tracker after every frame. The high-level tracker in turn converts the bounding boxes into ground plane coordinates and integrates them into physically plausible trajectories using the model selection framework from the previous section. During regular operation, the object detector only needs to be activated in regular intervals in order to prevent existing tracklets from degenerating and to start new ones for newly appearing pedestrians. In addition, tracklets can request new detections when they become uncertain. Overall, this results in considerable computational savings, as we will show in Section 5.3.

As motivated earlier, the difficulty of the street-level mobile tracking task results in a number of non-trivial challenges, which we address by introducing consistency checks and carefully modelled interactions between the different components of the tracking framework. We now present the different stages of our combined tracking framework.

5.2.1 Setup

Similar to previous work on mobile pedestrian tracking [GM07, LSV08, ELSV09], we assume a setup consisting of a stereo camera rig mounted on a mobile platform. From this setup, we obtain structure-from-motion (SfM), stereo depth, and a ground plane estimate for every frame (see Chapter 2 for a more detailed explanation). All subsequent processing is then performed only on the left camera stream. That is, our approach would in principle also be applicable to a setup consisting of a single camera



(a) (b) (c) (d)

Figure 5.3: Depth-based bounding box correction: (a) original bounding box; (b) depth map; (c) correction procedure; (d) corrected bounding box (see text for details).

and a time-of-flight range sensor.

5.2.2 Object Detection

For pedestrian detection, we apply the widely used HOG detector [DT05] in the efficient *fastHOG* GPU implementation by [PR09a]. Detections that are inconsistent with the scene geometry are automatically filtered out by enforcing a ground plane corridor.

5.2.3 Depth-based Bounding Box Correction

Both the level set (re-)initialization and the high-level 3D trajectory integration require accurately aligned bounding boxes. In general, the HOG detector however yields detections that include a certain border area. Inaccuracies in the detector's scale estimate may additionally result in fluctuations of the box size. Similarly, the boxes provided by the LS tracker may drift due to articulations and shape changes of the level set contour and need to be corrected. We therefore apply the following correction procedure both to new detections and after each LS tracking step. Starting from the original bounding box (Fig. 5.3(a)), we first compute the median depth around the bounding box center (Fig. 5.3(b)). We then determine the corresponding 3D point using the camera calibration from SfM and project it onto the ground plane (Fig. 5.3(c), steps(1)+(2)). We add a fixed offset in the viewing direction in order to determine the person's central foot point, and finally project the resulting 3D point back to the image (Fig. 5.3(c), steps (3)+(4)). This determines the bottom line of the corrected bounding box. The top line is found by searching for the highest point inside the bounding box that is within 0.5m of the median depth (Fig. 5.3(d)). As a final step, we verify that the resulting bounding box aspect ratio is in the range $[\frac{1}{3}, \frac{2}{3}]$. Bounding boxes falling outside this range are rejected as likely false positives.

(a) (b) (c) (d) (e) (f)

Figure 5.4: Initialization of the LS tracker: (a) detection box (green), initial object frame (yellow), and initialization of the level set (magenta); (b,c) evolved level set after 40 and 150 iterations; (d) level set transferred to next frame; (e) after warping; (f) after shape adaptation (5 iterations).

5.2.4 Level Set Initialization

When started with a new detection, the LS tracker tries to segment the torso of the person inside the bounding box. To this end, a new level set embedding function is initialized with a rectangular box (see Fig. 5.4), and the level set segmentation is iterated for 150 steps. In the next frame, the obtained contour is tracked and the resulting warp is applied to the object frame and the associated detection box in order to obtain the new object position. Afterwards, the level set shape is adapted for 5 iterations. We limit ourselves to tracking the person's torso here, since this body part deforms only very little, requiring far fewer shape adaptation iterations than an attempt to track the full body. This both speeds up level set tracking and it improves robustness, since it effectively limits the amount of "bleeding" that can occur to similar-coloured background structures. In order to infer the person's full extent, we maintain the transformation $V(\mathbf{x}, \mathbf{p}')$ from the warped reference frame to the original bounding box.

5.2.5 Multi-Region Handling and Overlap Detection

When tracking several persons at once, each of the tracked contours is represented by its own level set. Even if there are overlaps, the level sets will not interact directly (as, *e.g.*, in [BW04]). Instead, we use the stereo depth in order to resolve overlaps. All tracked objects are sorted according to their distance from the camera and the closest object is updated first. All pixels belonging to the resulting segmentation are masked out, such that they cannot be used by the remaining objects.

This leaves us with some objects that are only partially visible, which is in fact the same case as a person leaving the image frame, and only the visible part of the contour is tracked. Objects are discarded if only a small part of the area inside the contour (50% for person-person occlusions, 20% for occlusions by image borders) remains visible.

5.2.6 Consistency Checks

For robust operation, it is necessary to check the consistency of the tracking results. An object could be occluded, leave the image frame, or be lost due to other reasons. This need not even have any effect on the convergence of the LS tracker, which might get stuck on some local image structure, resulting in a wrong track. We therefore perform the following checks in order to identify corrupted tracklets. (1) If the object is occluded, only pixels with background colours will remain and the shape will typically shrink massively within a few frames. If such a case is detected, the tracklet is terminated. (2) We keep track of the median depth inside the tracked contour and react if this value changes too fast. We distinguish two cases here: If the median depth decreases too fast, this indicates an occlusion by another object; if the depth increases too fast, the object was probably lost. This may happen if background and foreground are too similar and the depth information is not significant enough. We terminate the tracklets in both cases. (3) Finally, objects whose median depth does not fit their ground-plane distance are also discarded. Typically, a failed consistency check indicates a tracking failure and will result in a request for the detector to be activated in the next frame. An exception are cases where an occlusion is "explained" by the high-level tracker (as was described in the previous section), or when the object is close to the image boundary and is about to leave the image.

5.2.7 Requesting New Detections

New detections are requested in the following cases: (1) if a tracklet has not received an updated detection in the last k frames; (2) if a tracking failure cannot be explained by an occlusion or by the tracked person leaving the image; (3) if no request has been issued for k frames (*e.g.*, since no object is currently tracked). A tracklet will not request new detections if it is close to the image boundary, as the chance for finding a detection there would be small. If a tracklet receives no updated detection despite its request, it will repeat the request, but will continue to be tracked as long as it passes the consistency and depth correction checks.

5.2.8 Level Set Re-initialization

There are several reasons for the tracklet shape to degenerate over time. Typical examples are lighting changes or similar colours near the object, which can cause the shape to shrink during tracking (see Fig. 5.5) or to bleed out during shape adaptation. By periodically updating a tracklet bounding box with new detector bounding boxes, it is possible to identify degenerating shapes based on their size in relation to the bounding box. This is done by first performing the LS tracking step in order to adapt the contour to the new image and then matching the tracked location to new detector boxes. If the box overlap (measured by the intersection-over-union criterion) is above 0.5, the detection box is used to update the relationship $V(\mathbf{x}, \mathbf{p}')$ between box and warp. The

(a) (b) (c) (d) (e) (f)

Figure 5.5: Adaptation to lighting changes: (a-c) tracked shape becomes too small due to lighting changes; (d,e) level set re-initialization is triggered; (f) tracking can continue.

level set contour itself is only updated if its area gets too small or too large with respect to the updated box, or if 20% of its content lie outside the box. Thus, the tracklet integrity is maintained and an ID change is avoided (*c.f*. Fig. 5.5).

5.2.9 Integration of Tracklets by High-Level Tracker

The high-level tracker's task is to integrate the tracklet bounding boxes into physically plausible 3D trajectories. This is done by first creating an *observation* at each tracked person's 3D foot point and then associating this observation to trajectory hypotheses. The overall procedure is similar to the general tracking-by-detection framework described in Section 5.1. However, we make the following changes in order to account for the additional information provided by the LS tracker.

Since we already know the tracklet identity of each observation from the LS tracker, we can use this information in order to simplify data association. Thus, we first try to extend each existing trajectory hypothesis by searching for an observation matching the trajectory's currently followed tracklet ID in a gating area around the Kalman filter prediction. If such an observation can be found, it will directly be associated with the trajectory. Note that in this case, only the motion model is considered; the appearance is assumed to be correct due to the association performed by the LS tracker. In case no observation with the correct tracklet ID can be found, we try to find the best-matching observation under the trajectory's motion and appearance model (again within a gating area determined by the Kalman filter uncertainty). If such a new observation can be found, the trajectory takes on the new tracklet ID, thus connecting the two tracklets. This latter case can occur if the LS tracker diverges and fails the consistency checks (in which case the tracklet will be terminated), if the tracked bounding box is rejected by the depth correction (in which case the tracklet may persist for up to k frames and can be recovered), or if the tracked object is occluded or leaves the image.

In addition to the above, each observation is used to start a new trajectory hypothesis, which searches backwards in time in order to find a potentially better explanation

for the observed data. This makes it possible to automatically create tracks for newly appearing persons and to correct earlier tracking errors. The final set of accepted tracks is then obtained by performing model selection, as was described in Section 5.1.

5.2.10 Tracking through Occlusions

As motivated above, a main advantage of the image-based low-level tracker, compared to a pure tracking-by-detection approach, is that it simplifies data association, thus making it easier to integrate observed pedestrian locations into valid tracks. The image-based tracklet generation will however fail when the tracked person gets occluded, which often occurs in practice. This is a limitation of any image-based tracking approach. While strategies can be devised to cope with short-term occlusions at the image-level, they would make this component unnecessarily complex. In our approach, we instead address this issue by explicit occlusion handling on the high-level tracker's side. In order to bridge short-time occlusions, we keep potentially occluded trajectories alive for up to 15 frames and extrapolate their position on the ground plane using the Kalman filter. Since the Kalman filter's positional uncertainty grows with the absence of observations, the corresponding person can likely be associated to the predicted trajectory when reappearing from the occlusion.

In addition, the high-level tracker can predict person-person occlusions and reinitialize the image-based tracker when those are over. For this, we backproject the predicted 3D bounding box of each tracked person into the image and compute the bounding box overlap using the intersection-over-union criterion. If the overlap is larger than 0.5, then an occlusion is likely to occur. This information is stored together with the occluded trajectory and is transmitted to the corresponding LS tracklet, which will typically be terminated 1-2 frames later when the consistency check fails. When the corresponding object is predicted to become visible again a few frames later, the object detector is fired in order to recover the person with as little delay as possible[1]. This is similar in spirit to the *track-suspend-fail* strategy proposed in [KPB+05], but our approach extends the idea through the integration of the robust multi-hypothesis tracking framework.

Fig. 5.6 shows an example where this occlusion handling process is used in practice. Cued by the occlusion prediction and the failed depth consistency check, the LS tracklet is terminated in order to avoid degeneracies (which would be likely in this case due to the similar color distributions). On the high-level tracker's side, the trajectory is however extrapolated through the occlusion. As soon as the occluded person becomes visible again, the object detector is fired in order to initialize a new LS tracklet, which is correctly associated to the trajectory, maintaining the person's identity.

[1] It would be possible to keep the tracklet alive and just suspend it until the occlusion is finished. However, in our experiments the "safe termination" and subsequent new tracklet generation strategy proved to be more robust.

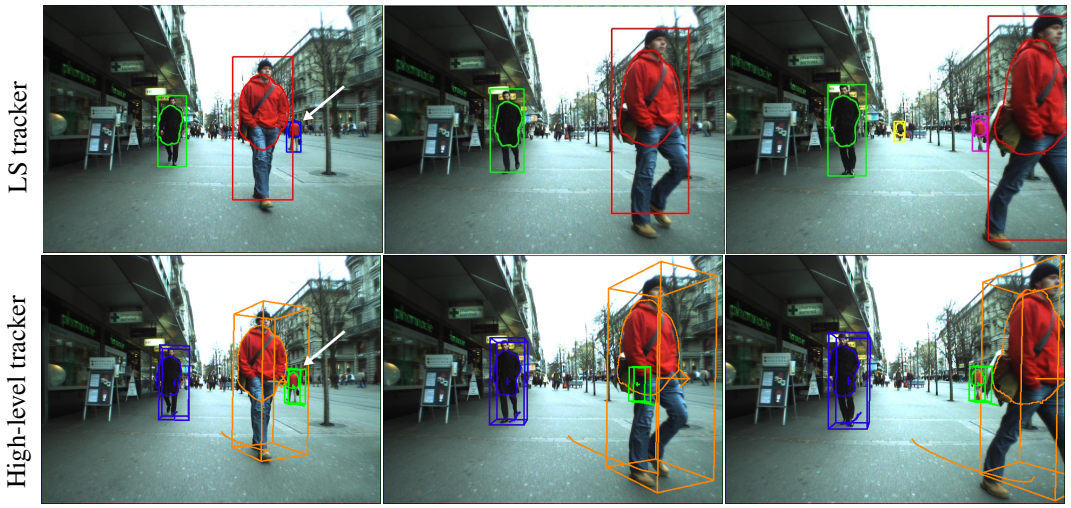


Figure 5.6: Example for the occlusion handling process: (Top): contours tracked by the LS tracker; (Bottom): output of the high-level tracker. When the distant person is temporarily occluded, its LS tracklet is terminated. As soon as the occlusion is over, a new tracklet is started. The high-level tracker connects both tracklets through the occlusion and maintains the person's identity.

5.3 Results

In this section we present experimental results for two datasets and discuss efficiency considerations.

5.3.1 Datasets

In order to evaluate our proposed approach, we applied it to two long and challenging sequences from the Zurich Mobile Pedestrian corpus generously provided by the authors of [ELSV09]. In detail, we used the sequences BAHNHOF (in the following: "Seq. A") and SUNNY DAY ("Seq. B") from this corpus. Both sequences were captured with a stereo rig at 13-14fps and 640×480 resolution. Seq. A (999 frames, containing 5193 annotated pedestrians with a height of at least 60 pixels) was taken on a crowded sidewalk on a clouded day. Seq. B (999 frames, 354 of which are annotated with 1867 annotations) was captured on a sunny day and contains strong illumination changes. Both sequences come with stereo depth maps, structure-from-motion localization, and ground plane estimates, which we use in this work. Similar to [ELSV09], we upscale all images to twice their original resolution before performing object detection in order to also detect pedestrians at larger distances. Using the upscaled images, fastHOG performed at 2-3fps, while it processed the original images at 10fps. In contrast to [ELSV09, LSV08], we however only use the left camera stream for detection and tracking, thus reducing the necessary processing effort. All system parameters were kept the same throughout both sequences.

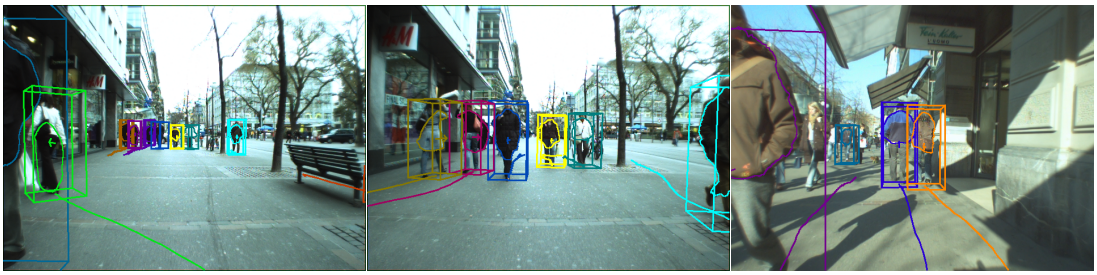


Figure 5.7: Examples demonstrating our approach's capability to continue tracking persons close to the camera and/or the image borders, where object detection is no longer applicable.

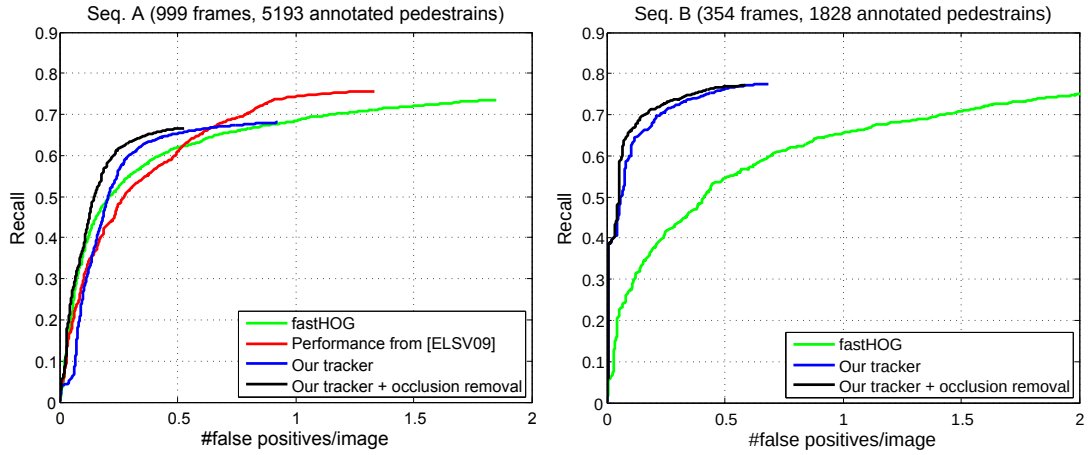


Figure 5.8: Quantitative tracking performance of our approach compared to different baselines.

5.3.2 Tracking Performance

Fig. 5.7 shows some qualitative results of our approach, demonstrating its capability to continue tracking persons who are close to the camera or partially occluded by the image boundaries. This is a fundamental advantage our proposed tracking framework can offer over pure tracking-by-detection approaches.

In order to obtain a quantitative assessment of our approach's performance, we adopt the evaluation criteria from [ELSV09] and measure the intersection-over-union of tracked person bounding boxes and annotations in every frame. We accept detections having an overlap greater than 0.5 as correct and report the results in terms of *recall vs. false positives per image* (fppi). Fig. 5.8 shows the resulting performance curves when we set the maximum re-initialization interval to $k = 5$ frames (in blue), together with the baseline of fastHOG (in green). As can be seen, our approach achieves good performance, reaching 65% and 76% recall at 0.5 fppi for Seq. A and Seq. B, respectively. As the bounding box criterion penalizes the tracker's property of predicting a person's location through occlusions (since those cases are not annotated in the test data), we additionally provide the performance curve when filtering out tracked

Table 5.1: Track-level evaluation according to the criteria by [WN07].

	Seq. A	Seq. B
# persons	67	53
mostly hit	58	40
part. tracked	2	4
mostly missed	7	9
false alarms	5	1
ID sw.	2	1
ID sw. (long occ.)	9	5

bounding boxes which are more than 50% occluded by other boxes (in black). This results in an additional improvement, as becomes visible by the increased precision.

For comparison, we also provide the performance curve reported by [ELSV09] on Seq. A, which is also based on HOG detections (shown in red, no such curve is available for Seq. B). This approach integrates detections from both camera streams and thus obtains a higher recall. Its performance should be compared to our blue curve, since no occlusion removal was performed in [ELSV09]. Still, it can be seen that our approach achieves better performance in the high-precision range, despite only using a single camera stream. This is a result of the better data association provided by the image-level tracklets.

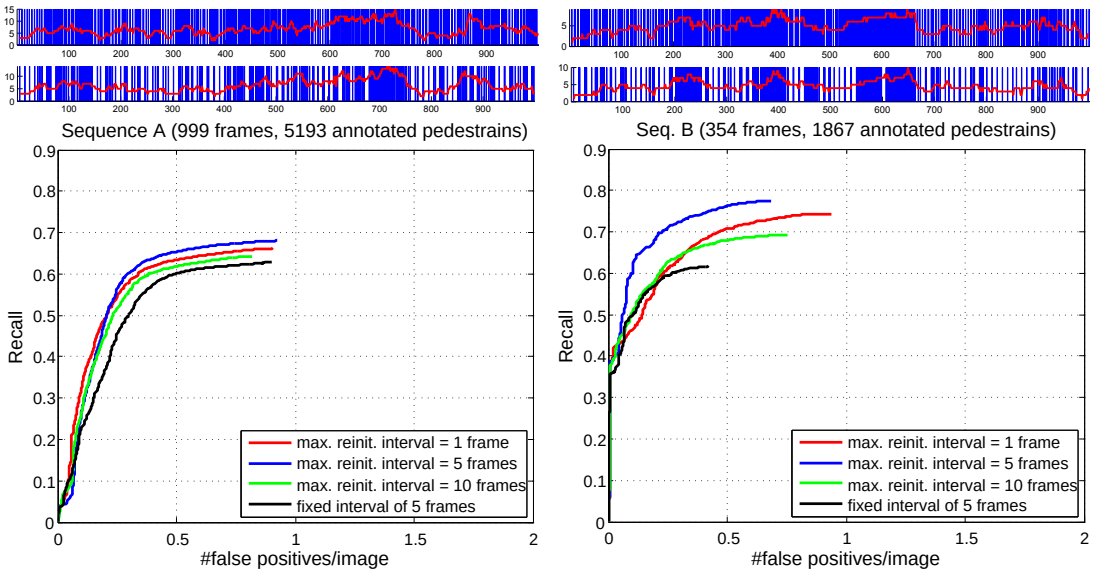
Table 5.1 reports a track-level evaluation according to the criteria by [WN07], showing that most pedestrians are correctly tracked and only few ID switches occur (except for those caused by very long occlusions). Fig. 5.9 shows typical results of our combined tracker for both test sequences and visualize the obtained level set contours. As can be seen, our system is able to track most of the visible pedestrians correctly in a very busy environment with many occlusions.

5.3.3 Efficiency Considerations

An additional motivation for our approach was to reduce the dependence on the costly pedestrian detector. Even though efficient GPU implementations are now available for HOG (*e.g.* [PR09a]), the obtainable framerate is still not sufficient for real-time operation in a pure tracking-by-detection context. In addition, the excessive power consumption of GPUs is a major restriction for their use in mobile robotics applications. In contrast, the level set tracking approach employed here can be very efficiently implemented on regular CPUs. [BR08] reports a framerate of 85Hz for tracking a single target of size 180×180 pixels in their implementation. In our application, we track targets at a lower resolution of 80×100 pixels and therefore expect even faster performance once our code is fully optimized.

An important consideration in this respect is how often the pedestrian detector

Figure 5.9: Example tracking results of our approach on Seq. A and B.



(a) Seq. A (b) Seq. B

Figure 5.10: (Top): Frequency of detector activations for both sequences for an interval of 5 (first) and 10 (second) frames. The red curve shows the number of tracked pedestrians; (Bottom): Tracking performance for the two test sequences when varying the maximum re-initialization interval.

needs to be activated for robust tracking performance. Remember that our approach lets the LS tracker request detections whenever required, but enforces a maximum re-initialization interval of k frames. Fig. 5.10 shows the effective frequency of detector activations when setting this interval to $k \in \{1,5,10\}$, together with the resulting tracking performance. As can be seen, a setting of $k = 5$ provides the best tracking quality and results in a detector activation on average every 1.66 frames. The red curve indicates that the low level tracker requests new detections more often when the number of trajectories rises. By increasing the maximum interval to 10 frames, the detector activation rate falls to once every 2.71 frames at a small loss in recall that is still comparable to [ELSV09] at 0.5 fppi. Considering that [ELSV09] performed detection in both camera streams, our approach thus requires 5.42 times fewer detector activations. Finally, we show the resulting performance when activating the detector at a fixed interval of 5 frames, without allowing additional requests. This results in an additional small drop in recall, but still yields good overall performance.

5.4 Discussion

We have presented an integrated framework for mobile street-level multi-person tracking. Our approach combines the advantages of a fast segmentation-based tracker for following individual persons with the robustness of a high-level multi-hypothesis tracking framework for performing longer-term data association. As our experiments

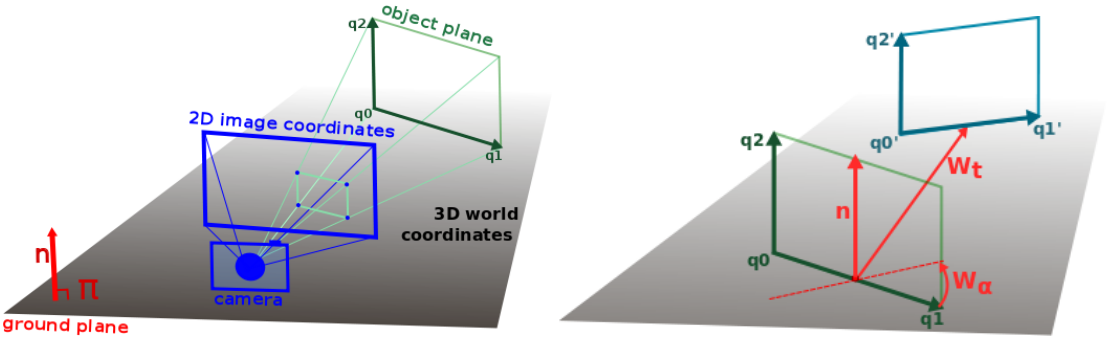
have shown, the approach reaches state-of-the-art performance, while requiring fewer detector evaluations than conventional tracking-by-detection approaches. Our results open several interesting research perspectives. The requested detector activations for tracklet re-initialization could be restricted to the tracklet's immediate neighbourhood, thus resulting in further speedups. In addition, the obtained level set segmentation could be a possible starting point for articulated tracking that we plan to explore in future work.

Chapter 6

Geometrically Constrained Level Set Tracking

Object tracking from a mobile platform is an important problem with many potential applications. Consequently, many different approaches have been applied to this problem in the past, including tracking-by-detection [BHD00, GM07, LSV08, ELSV09], model-based [KDN93, DT97], template-based [CJDC02, BMR05] and region-based [CRD07, SC09] methods. In this chapter we again combine our level set tracking framework with a tracking-by-detection framework. However we model the object's 3D movement already in the level set framework by allowing a homography as transformation in the warping step. As all appearance-based approaches, our level set tracker is restricted in the types of transformations it can robustly handle without additional knowledge about the expected motions. In this chapter we investigate the use of geometric constraints for improving level set tracking and show how geometric scene knowledge can be directly integrated into the level set warping step in order to constrain object motion. For this, we propose a constrained-homography transformation model that represents rigid motion on the ground plane. This model is targeted for tracking vehicles in an automotive scenario and takes advantage of an egomotion estimate obtained by structure-from-motion (SfM).

An advantage of our proposed approach, compared to pure 2D tracking, is that it restricts the deformation space to physically plausible rigid-body motions, thus increasing the robustness of the estimation step. In addition, the 3D transformation model allows us to directly infer the tracked object's detailed 3D motion and orientation at every time step. We show how this information can be used in a higher-level tracker, which models the vehicle dynamics in order to obtain smooth and physically correct trajectories. The additional measurements provided by our geometrically constrained level set tracker make the estimation more robust and lead to smoother trajectories. We demonstrate our approach on several video sequences for tracking cars on city roads under viewpoint changes.



(a) Coordinate Systems (b) Transformation Model

Figure 6.1: Visualization of the coordinate systems and the proposed transformation model used in our approach.

6.1 Geometric Transformation Model

In addition to the image based tracking approach as described in the previous chapters, we model the 3D position of a tracked object. This allows us to make assumptions about an object's movement by the projective distortions that arise on the 2D image. The gained information is then used by the high-level tracker as will be described later in Section 6.2. In this section we describe the different coordinate systems, the 3D transformation model and how the resulting transformation is used in the optimization during the rigid registration of the object's position.

6.1.1 Coordinate Systems

Fig. 6.1(a) shows the used coordinate systems. The image itself consists of a number of pixels with 2D coordinates. The colours of these pixels correspond to points in the world which were projected onto the image plane. The 3D coordinates of those points cannot however be inferred directly from one image without additional depth information. We use a ground plane, which was obtained with structure-from-motion (SfM), to estimate the base point of a detected object. We approximate the object to be a plane in world coordinates, which is orthogonal to the ground plane. This object plane can be described with a point $\mathbf{q}_0$ and two direction vectors $\mathbf{q}_1$ and $\mathbf{q}_2$.

$$\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix} = \mathbf{P}\mathbf{x}_w = \mathbf{P}\mathbf{Q}\mathbf{x}_o, \quad \text{with } \mathbf{Q} = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{x}_o = \begin{bmatrix} x_o \\ y_o \\ 1 \end{bmatrix}, \ \mathbf{x}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{6.1}$$

where $\mathbf{x}_o$ is a 2D point on the object plane and $\mathbf{x}_w$ are its corresponding world coordinates. The point $\mathbf{x}_i$ in the image that corresponds to this world point can be obtained by projection with the camera matrix $\mathbf{P}$.

6.1.2 3D Transformation Model

The level set tracking framework requires us to specify a family of warping transformations $\mathbf{W}$ that relate the previous object reference frame to the current one. In the following, we show how this warp can be used to incorporate scene knowledge by enforcing geometric constraints on the object motion.

Our target scenario is an automotive application where the goal is to track other vehicles' motions relative to our own vehicle. In this scenario, we can safely assume that the tracked object parts are approximately planar and that the target objects move rigidly on the ground plane. This means that their 3D shape will not change between two frames; only their position relative to the camera will change. The resulting projective distortions in the image can therefore be modelled by a homography. However, an unconstrained homography has too many degrees of freedom, which makes it hard to keep the tracking approach robust. Instead, we propose to use the available scene knowledge by modelling $\mathbf{W}$ as a *constrained homography* that requires fewer parameters and that can be estimated more robustly.

Fig. 6.1(b) illustrates our proposed transformation model. We represent object motion by a 3D homography, consisting of a rotation $\mathbf{W}_\alpha$ around an axis orthogonal to the ground plane and a translation $\mathbf{W}_t$ along the vector $\mathbf{t} = [t_x, t_y, t_z]^\mathsf{T}$. In order to compare the object points with the stored level set contour of the previous frame, we then project the object into the image using an estimated camera matrix $\mathbf{P}$ obtained by SfM. Finally, we compute a 2D homography $\mathbf{W}_{obj}$ which warps the content of the object window (defined by the projections of its four corner points) onto the level set reference frame. This is done using the DLT algorithm [HZ00].

$$\mathbf{W} = \mathbf{W}_{obj}\,\mathbf{P}\,\mathbf{W}_t\mathbf{W}_\alpha\mathbf{Q}\begin{bmatrix} x_o \\ y_o \\ 1 \end{bmatrix} \tag{6.2}$$

$\mathbf{W}_\alpha$ can be computed as a sequence of several transformations: a translation $\mathbf{T}_P$ moving the rotation axis into the origin; a rotation $\mathbf{R}_{xz}$ into the xz-plane; another rotation $\mathbf{R}_z$ onto the z-axis; and finally a rotation $\mathbf{R}_z(\alpha)$ about the z-axis with the desired angle α, followed by the inverse of the first three steps.

$$\mathbf{W}_\alpha = \mathbf{T}_P^{-1}\mathbf{R}_{xz}^{-1}\mathbf{R}_z^{-1}\mathbf{R}_z(\alpha)\mathbf{R}_z\mathbf{R}_{xz}\mathbf{T}_P, \qquad \mathbf{W}_t = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.3}$$

In the above formulation, we have assumed a general translation $\mathbf{W}_t$. In principle, this could be restricted further to only allow translations parallel to the ground plane. However, the estimated ground plane is not always completely accurate and in any case does not account for uneven ground, especially at farther distances. We have therefore found that allowing a small movement component in the direction of the ground plane normal is necessary to achieve robustness.

6.1.3 Optimizing for the Transformation

The tracking framework uses the Gauss-Newton method to optimize the warp between two image frames. This requires the Jacobian of the overall warp $\mathbf{W}$, which contains the partial derivatives of $\mathbf{W}$ with respect to the parameters α, t_x, t_y and t_z.

$$\frac{\partial \mathbf{W}}{\partial \Delta \mathbf{p}} \quad \text{with} \quad \Delta \mathbf{p} = \begin{bmatrix} \alpha & t_x & t_y & t_z \end{bmatrix}^\mathsf{T}$$

The parameters $\Delta\mathbf{p}$ available for optimization restrict the possible movements of the contour and the gradient $\frac{\partial \mathbf{W}}{\partial \Delta \mathbf{p}}$ indicates the effect a certain parameter value has on the position of the contour in the image. (6.4) is substituted into (3.21) and evaluated for every point $\mathbf{x}$ in the band around the contour which is determined by $\delta_\varepsilon(\Phi)$. In this way pixel locations with a low probability of belonging to the foreground contribute a warp towards the outside of the contour and vice versa. The algorithm has converged when the step size has become very small. We found it can be useful to normalize the step size depending on the number of examined pixels, *i.e.* reduce it, in order to achieve better convergence.

6.1.4 Final Tracking Algorithm

Putting the above steps together, we can summarize the proposed tracking algorithm as follows:

(a) Initialize the object position using *e.g.* a detection bounding box.

(b) Apply the level set segmentation for it_2 iterations.

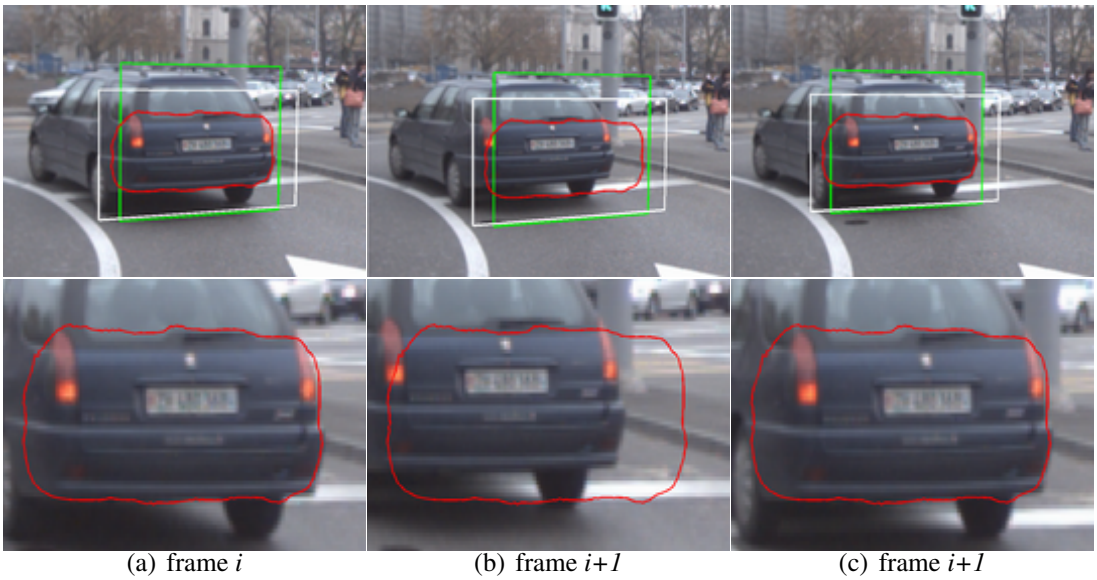
(c) Compute the object plane's world coordinates $\mathbf{Q}$ by projecting the detection box base points onto the ground plane.

(d) For the following frames: Track the object's shape, *i.e.* compute $\Delta\mathbf{p}$ for the warp between two images i and $i+1$:
Pre-compute:

1. Interpolate synthetic view I^i_{obj} for the old object frame (Fig. 6.2(a)) by warping the old image with $\mathbf{W}^i_{obj}$.
2. Compute $P(\mathbf{x}_i|\Phi, \mathbf{p}, \mathbf{y}_i)$ for I^i_{obj}.
3. Assume the object is still located at the same 3D position in the new frame to compute $\mathbf{W}^{i+1}_{obj}$.

Iterate:

4. Interpolate synthetic view I^{i+1}_{obj} for the new object frame (Fig. 6.2(b)) by warping the new image with $\mathbf{W}^{i+1}_{obj}$.



(a) frame i (b) frame $i+1$ (c) frame $i+1$

Figure 6.2: (a) 3D position and synthetic view of object frame. (b) Before warp: 3D position as in frame i. (c) After warp: tracked 3D position. Notice how the contour moved in 3D and its projection onto the image plane changed accordingly; its shape did not change.

5. Use eq. (3.21) with eq. (6.4) to find a set of parameters $\Delta\mathbf{p}$ such that the contour in I^{i+1}_{obj} better matches I^{i}_{obj}.
6. Use the inverse of the estimated homography $(\mathbf{W}_t\mathbf{W}_\alpha)^{-1}$ to warp the modelled 3D coordinates of the object and obtain a new 3D position estimate of the object in frame $i+1$ (Fig. 6.2(c)). (*E.g.* if image $i+1$ needs to be warped "closer" to the camera in order to look like image i, the object in fact moved to the back.)
7. Re-compute $\mathbf{W}^{i+1}_{obj}$ with the newly obtained 3D coordinates in order to obtain an improved synthetic view.

Until: the step size is small enough, $\|\Delta\mathbf{p}\| \leq \varepsilon_p$.

6. $(\mathbf{W}^{i+1}_{obj})^{-1}$ can be used to obtain the current contour.

The results of this procedure are a level set contour and a bounding box for each frame, as well as the estimated 3D position and orientation of the object.

6.2 Integration with a High-Level Tracker

The level set tracking approach described in the previous section can robustly follow an individual object over time. However, it requires an initialization to pick out objects of interest, and it does not incorporate a dynamic model to interpret the observed

motion. For this reason, we integrate it with a high-level tracker. In this integration, the task of the level set tracker is to generate independent *tracklets* for individual objects, which are then integrated into a consistent scene interpretation with physically plausible trajectories by the high-level tracker.

6.2.1 System Overview

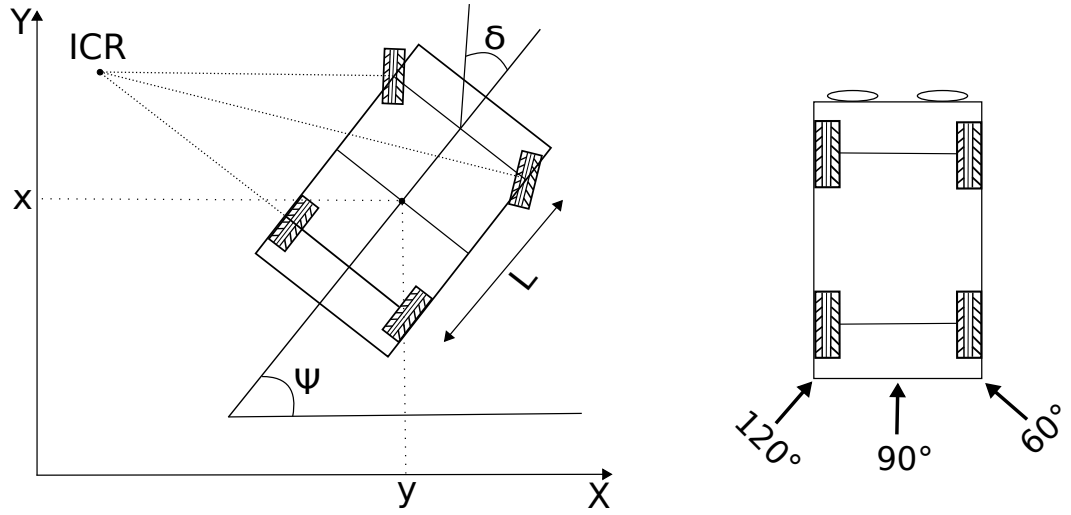
We apply a simplified version of the robust multi-hypothesis tracking framework by [LSV08]. Given an estimate of the current camera position and ground plane location from SfM, we collect detected vehicle positions on the ground plane over a temporal window. Those measurements are then connected to trajectory hypotheses using Extended Kalman Filters (EKFs). Each trajectory obtains a score, representing the likelihood of the assigned detections under the motion and appearance model (represented as an RGB color histogram). As a result, we obtain an overcomplete set of trajectory hypotheses, from which we select the best explanation for the observed measurements by applying model selection in every frame (Details can be found in [LSV08]).

6.2.2 Motion Model

For modeling the vehicle trajectories, we use an EKF with the Ackermann steering model (*e.g.* [MN87]), as shown in Fig. 6.3(a). This model incorporates a nonholonomic motion constraint, enforcing that the velocity vector is always perpendicular to the rear wheel axis. The state vector is given as $\mathbf{s}_t = [x_t, y_t, \psi_t, v_t, \delta_t, a_t]$, where (x, y) is the position of the car, ψ the heading angle, δ the steering angle, v the longitudinal velocity, and a the acceleration. The prediction step of the Kalman filter is then defined as follows:

$$\mathbf{s}_{t+1} = \begin{pmatrix} x_t + v_t\cos(\psi_t)\Delta t + \frac{1}{2}a_t\cos(\psi_t)\Delta t^2 \\ y_t + v_t\sin(\psi_t)\Delta t + \frac{1}{2}a_t\sin(\psi_t)\Delta t^2 \\ \psi_t + \frac{v_t}{L}\tan(\delta)\Delta t \\ v_t + a_t\Delta t \\ \delta_t \\ a_t \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ n_\delta \\ n_a \end{pmatrix} . \tag{6.4}$$

L is the distance between the rear and front wheel axes and is set to a value of 3.5m. Multi-vehicle tracking-by-detection using a similar motion model was demonstrated by [ELSV09] based on a battery of object detectors with discretised viewing angles. We use a similar coarse discretisation of the viewing angle with three separate, HOG-style [DT05] vehicle detectors in order to initialize our level set tracker (Fig. 6.3(b)). However, after this initialization, we only use the observations provided by the low-level tracker and integrate them into the motion model.



(a) (b)

Figure 6.3: (a) Ackermann steering model used for modelling the motion of the vehicle (assumes rolling without slippage). (b) Discretisation of the the detected viewing angles.

6.3 Results

In this section we present qualitative results for three sequences and compare it to a baseline approach.

We demonstrate our approach on three parts of a challenging sequence from the Zurich Mobile Car corpus, generously provided by the authors of [ELSV09], in the following called CITY. The sequence was captured using a stereo setup (13-14 fps and 640×480 resolution) mounted on top of a car. We use the SfM and ground plane estimates provided with this data set, but then restrict all further processing to just the left camera stream.

Fig. 6.4 shows qualitative results of our approach on three different test sequences, demonstrating its capability to accurately track vehicles under viewpoint changes. As can be seen here, the estimated vehicle orientation from the level set tracker supports the high-level tracker by providing 3D position and orientation estimates and hence in computing smooth vehicle trajectories.

Fig. 6.5 presents a comparison of our 3D estimation approach with the results of a baseline level set tracker using only a 2D (translation + scale) transformation model (as in [BR08]). As can be seen from those results, our approach achieves better tracking accuracy and manages to closely follow the target vehicles despite considerable viewpoint changes. In contrast, the 2D baseline method slips off the car, since the 3D position of the car's center is incorrectly estimated, resulting in a wrong trajectory.

Figure 6.4: Tracking results of our approach on three different test sequences. (Top rows): Tracked level set contour and 3D transformation; (Bottom rows): Integrated 3D estimation results of the high-level tracker. Please refer to Appendix B.3 for more result images.

Figure 6.5: Comparison with the results of a 2D baseline model. As can be seen, the lacking orientation estimate causes the high-level tracker to slip off the vehicles when the viewpoint changes, because the center of the vehicle cannot be estimated correctly.

6.4 Discussion

Our proposed approach has several advantages. Compared to a pure tracking-by-detection approach, the level set tracker yields much finer-grained measurements of the viewing angle at which the target vehicle is seen. In addition, the level set tracker can continue tracking objects even when they partially leave the image and the object detector would fail. Compared to a level set tracker with a simpler 2D transformation model (*i.e.*, just using translation and scale, without applying ground plane constraints), our proposed model has the advantage of being able to estimate not only the target vehicle's location, but also its current orientation. This orientation estimate is beneficial in two respects. It allows us to extrapolate from the tracked car trunk lo-

cation and infer the true object center, resulting in better position estimates (which is especially important for elongated objects such as cars). And it makes it possible to use the orientation as *observed quantity* in the motion model, resulting in better predictions. All of those factors contribute to yield robust tracking performance.

In conclusion, we have presented an approach for incorporating geometric scene constraints into the warping step of a level set tracker. Our approach allows to estimate both the location and orientation of the tracked object in 3D, while at the same time restricting the parameter space for more robust estimation. As we have shown, the estimation results can be used to improve the performance of a higher-level multi-hypothesis tracker integrating the measurements with vehicle dynamics into physically plausible trajectories. A possible extension could be to incorporate detections for different vehicle orientations, as well as stereo depth information, in order to initialize tracking also for other vehicle viewpoints.

Chapter 7

Conclusions

In conclusion, we have presented a segmentation and tracking framework, that can be automatically initialized and provide additional information about data association and objects' 3D motion to a tracking-by-detection system for mobile multi-object tracking.

Our level set segmentation and tracking framework is robust to automatic initialisation with detector bounding boxes through the use of a smoothness constraint on the contour. We showed how stereo depth information can be employed in addition to color histograms in the warping step to increase robustness to similar coloured background. We track multiple objects with independent level sets and resolve contour overlaps by registering the nearer objects first and masking out the used pixels for the remaining objects.

We have presented a framework for mobile street-level multi-person tracking consisting of our level set tracker integrated with a multi-hypothesis tracking framework. In this system the LS tracker is initialised with HOG detector bounding boxes and generates tracklets for the high-level tracker which takes over longer-term data association and bridges occlusions. We perform a number of consistency checks including the enforcement of geometric constraints on tracklets, *i.e*. a ground plane and consistent depth values, and reach state-of-the-art performance, while requiring fewer detector evaluations than conventional tracking-by-detection approaches. Moreover our LS tracker is able to continue tracking objects, that are near the camera and partially leaving the image frame and thus will not be found by a detector, by tracking only the visible part of the contour.

Furthermore, we have shown how geometric scene knowledge can be incorporated into our level-set tracking framework. By using a homography constrained to rigid motion on the ground plane in the tracker's warping step we are able to draw conclusions about the 3D movement of a tracked object without using shape priors. We demonstrate how an estimation of a vehicle's viewing angle can improve the performance of a higher-level multi-hypothesis tracker integrating the measurements with vehicle dynamics into physically plausible trajectories.

In future work we plan to explore the incorporation of optical flow and scene flow into the level set tracking framework to obtain more exact and more detailed informa-

tion about the 3D movement of tracked objects. Moreover we plan to extend the rigid registration to several level sets at once, in order to track several planes in an image, which intersect at a specific angle. In this way we could for example model two sides of a car, which would be approximated by two orthogonal planes, each with its own contour.

Appendix A

Derivations

A.1 Derivation of the Pixel-Wise Posteriors

The joint distribution for one pixel as in Fig. 3.1 is

$$P(\mathbf{x},\mathbf{y},\Phi,\mathbf{p},M) = P(\mathbf{x}|\Phi,\mathbf{p},M)P(\mathbf{y}|M)P(M)P(\Phi)P(\mathbf{p}) \tag{A.1}$$

$$P(\mathbf{x},\Phi,\mathbf{p},M|\mathbf{y})P(\mathbf{y}) = P(\mathbf{x}|\Phi,\mathbf{p},M)P(\mathbf{y}|M)P(M)P(\Phi)P(\mathbf{p}) \tag{A.2}$$

$$P(\mathbf{x},\Phi,\mathbf{p},M|\mathbf{y}) = P(\mathbf{x}|\Phi,\mathbf{p},M)P(M|\mathbf{y})P(\Phi)P(\mathbf{p}) \tag{A.3}$$

with

$$P(M_j|\mathbf{y}) = \frac{P(\mathbf{y}|M_j)P(M_j)}{P(\mathbf{y})} = \frac{P(\mathbf{y}|M_j)P(M_j)}{\sum_{i=\{f,b\}} P(y|M_i)P(M_i)}, \quad j=\{f,b\} \tag{A.4}$$

and marginalization over the models M yields the pixel-wise posterior probability of shape Φ and location $\mathbf{p}$ given a pixel $\{\mathbf{x},\mathbf{y}\}$:

$$P(\mathbf{x},\Phi,\mathbf{p},M_f|\mathbf{y}) + P(\mathbf{x},\Phi,\mathbf{p},M_b|\mathbf{y}) = P(x,\Phi,\mathbf{p}|\mathbf{y}) \tag{A.5}$$

$$= P(\Phi,\mathbf{p}|\mathbf{x},\mathbf{y})P(\mathbf{x}) \tag{A.6}$$

$$= \sum_{i=\{f,b\}} \Big\{ P(\mathbf{x}|\Phi,\mathbf{p},M_i)P(M_i|\mathbf{y}) \Big\} P(\Phi)P(\mathbf{p}) \tag{A.7}$$

follows

$$P(\Phi,\mathbf{p}|\mathbf{x},\mathbf{y}) = \frac{1}{P(\mathbf{x})} \sum_{i=\{f,b\}} \Big\{ P(\mathbf{x}|\Phi,\mathbf{p},M_i)P(M_i|\mathbf{y}) \Big\} P(\Phi)P(\mathbf{p}) \tag{A.8}$$

Fusing the pixel-wise posteriors with a logarithmic opinion pool and specifying

$$P(\mathbf{x}_i|\Phi,\mathbf{p},M_f) = \frac{H_\varepsilon(\Phi(\mathbf{x}_i))}{\eta_f}, \qquad P(\mathbf{x}_i|\Phi,\mathbf{p},M_b) = \frac{1-H_\varepsilon(\Phi(\mathbf{x}_i))}{\eta_b} \tag{A.9}$$

$$P(M_f) = \frac{\eta_f}{\eta}, \qquad P(M_b) = \frac{\eta_b}{\eta} \tag{A.10}$$

$$N = \eta = \eta_f + \eta_b, \qquad \eta_f = \sum_{i=1}^{N} H_\varepsilon(\Phi(\mathbf{x}_i)), \qquad \eta_b = \sum_{i=1}^{N} 1 - H_\varepsilon(\Phi(\mathbf{x}_i)) \tag{A.11}$$

yields

$$P(\Phi,\mathbf{p}|\Omega) = \prod_{i=1}^{N}\sum_{M}\Big[P(\mathbf{x}|\Phi,\mathbf{p},M_i)P(M_i|\mathbf{y})\Big]P(\Phi)P(\mathbf{p}) \tag{A.12}$$

$$= \prod_{i=1}^{N}\Big[P(\mathbf{x}_i|\Phi,\mathbf{p},M_f)P(M_f|\mathbf{y}_i) + P(\mathbf{x}_i|\Phi,\mathbf{p},M_b)P(M_b|\mathbf{y}_i)\Big]P(\Phi)P(\mathbf{p}) \tag{A.13}$$

$$= \prod_{i=1}^{N}\left[\frac{H_\varepsilon(\Phi(\mathbf{x}_i))}{\eta_f}P(M_f|\mathbf{y}_i) + \frac{1-H_\varepsilon(\Phi(\mathbf{x}_i))}{\eta_b}P(M_b|\mathbf{y}_i)\right]P(\Phi)P(\mathbf{p}) \tag{A.14}$$

$$= \prod_{i=1}^{N}\left[\frac{H_\varepsilon(\Phi(\mathbf{x}_i))P(\mathbf{y}_i|M_f)P(M_f)}{\eta_f\sum_M P(\mathbf{y}_i|M)P(M)} + \frac{(1-H_\varepsilon(\Phi(\mathbf{x}_i)))P(\mathbf{y}_i|M_b)P(M_b)}{\eta_b\sum_M P(\mathbf{y}_i|M)P(M)}\right]P(\Phi)P(\mathbf{p}) \tag{A.15}$$

$$= \prod_{i=1}^{N}\left[\frac{H_\varepsilon(\Phi(\mathbf{x}_i))P(\mathbf{y}_i|M_f)}{\eta\sum_M P(\mathbf{y}_i|M)P(M)} + \frac{(1-H_\varepsilon(\Phi(\mathbf{x}_i)))P(\mathbf{y}_i|M_b)}{\eta\sum_M P(\mathbf{y}_i|M)P(M)}\right]P(\Phi)P(\mathbf{p}) \tag{A.16}$$

$$= \prod_{i=1}^{N}\left[\frac{H_\varepsilon(\Phi(\mathbf{x}_i))P(\mathbf{y}_i|M_f) + (1-H_\varepsilon(\Phi(\mathbf{x}_i)))P(\mathbf{y}_i|M_b)}{\eta_f P(\mathbf{y}_i|M_f)P(M_f) + \eta_b P(\mathbf{y}_i|M_b)P(M_b)}\right]P(\Phi)P(\mathbf{p}) \tag{A.17}$$

$$= \prod_{i=1}^{N}\Big[H_\varepsilon(\Phi(\mathbf{x}_i))P_f + (1-H_\varepsilon(\Phi(\mathbf{x}_i)))P_b\Big]P(\Phi)P(\mathbf{p}) \tag{A.18}$$

$$= \prod_{i=1}^{N}P(\mathbf{x}_i|\Phi,\mathbf{p},\mathbf{y}_i)P(\Phi)P(\mathbf{p}) \tag{A.19}$$

where

$$P_f = \frac{P(\mathbf{y}_i|M_f)}{\eta_f P(\mathbf{y}_i|M_f) + \eta_b P(\mathbf{y}_i|M_b)}, \quad P_b = \frac{P(\mathbf{y}_i|M_b)}{\eta_f P(\mathbf{y}_i|M_f) + \eta_b P(\mathbf{y}_i|M_b)} \tag{A.20}$$

$$P(\mathbf{x}_i|\Phi,\mathbf{p},\mathbf{y}_i) = H_\varepsilon(\Phi(\mathbf{x}_i))P_f + (1-H_\varepsilon(\Phi(\mathbf{x}_i)))P_b \tag{A.21}$$

An embedding function Φ that is more like a signed distance function is more desirable, since numerically stable, thus should be made more probable, so $P(\Phi)$ is specified as a geometric prior that rewards a signed distance function:

$$P(\Phi) = \prod_{i=1}^{N}\frac{1}{\sigma\sqrt{2\pi}}\exp -\frac{(|\nabla\Phi(\mathbf{x}_i)|-1)^2}{2\sigma^2} \tag{A.22}$$

where σ is the weight of the prior.
Substituting (A.22) into (A.19) and taking logs gives the following expression for the

log posterior:

$$\log(P(\Phi,\mathbf{p}|\Omega)) \propto \sum_{i=1}^{N}\left\{\log(P(\mathbf{x}_i|\Phi,\mathbf{p},\mathbf{y}_i)) - \frac{(|\nabla\Phi(\mathbf{x}_i)|-1)^2}{2\sigma^2}\right\} + N\log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) + \log(P(\mathbf{p})) \tag{A.23}$$

A.2 Derivation of the Segmentation Framework

For segmentation we optimize (A.23) w.r.t Φ, so the last two terms can be dropped, the rest is then differentiated by calculus of variation [Eva10]:

$$\frac{\partial P(\Phi,\mathbf{p}|\Omega)}{\partial\Phi} = \frac{\delta_\varepsilon(\Phi)(P_f-P_b)}{P(\mathbf{x}|\Phi,\mathbf{p},\mathbf{y})} - \frac{1}{\sigma^2}\left[\nabla^2(\Phi) - \mathrm{div}\left(\frac{\nabla\Phi}{|\nabla\Phi|}\right)\right]. \tag{A.24}$$

A.3 Derivation of the Tracking Framework

In preparation for differentiation w.r.t $\mathbf{p}$ some terms in (A.23) can be dropped:

$$\log(P(\Phi,\mathbf{p}|\Omega)) \propto \sum_{i=1}^{N}\{\log(P(\mathbf{x}_i|\Phi,\mathbf{p},\mathbf{y}_i))\} + \log(P(\mathbf{p})) + const. \tag{A.25}$$

Now the warp $\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})$ is introduced into (A.25) and $P(\mathbf{p})$ dropped for the moment (this is handled with drift correction):

$$\log(P(\Phi,\mathbf{p}|\Omega)) \propto \sum_{i=1}^{N}\log\Big\{P(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})|\Phi,\mathbf{p},\mathbf{y}_i)\Big\} \tag{A.26}$$

$$= \sum_{i=1}^{N}\log\Big\{H_\varepsilon(\Phi(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})))P_f + (1-H_\varepsilon(\Phi(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p}))))P_b\Big\} \tag{A.27}$$

All terms are probabilities and therefore positive, *i.e.* can be written as sqared sqare-roots and used in a Gauss-Newton optimization. To this end each square-root is approximated with a first-order Taylor series.
With $h = H_\varepsilon(\Phi(\mathbf{x}_i))$:

$$H_\varepsilon(\Phi(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p}))) = \Big[\sqrt{H_\varepsilon(\Phi(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})))}\Big]^2 \approx \left[\sqrt{h} + \frac{1}{2\sqrt{h}}\mathbf{J}\Delta\mathbf{p}\right]^2 \tag{A.28}$$

and similarly

$$1-H_\varepsilon(\Phi(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p}))) = \Big[\sqrt{1-H_\varepsilon(\Phi(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})))}\Big]^2 \approx \left[\sqrt{1-h} - \frac{1}{2\sqrt{1-h}}\mathbf{J}\Delta\mathbf{p}\right]^2 \tag{A.29}$$

where

$$\mathbf{J} = \frac{\partial H_\varepsilon}{\partial \Phi}\frac{\partial \Phi}{\partial \mathbf{x}}\frac{\partial \mathbf{W}}{\partial \Delta \mathbf{p}} = \delta_\varepsilon(\Phi(\mathbf{x}_i))\nabla\Phi(\mathbf{x}_i)\frac{\partial \mathbf{W}}{\partial \Delta \mathbf{p}} \tag{A.30}$$

i.e.

$$\log(P(\Phi,\mathbf{p}|\Omega)) \propto \sum_{i=1}^{N}\log\left\{\left[\sqrt{h}+\frac{1}{2\sqrt{h}}\mathbf{J}\Delta\mathbf{p}\right]^2 P_f + \left[\sqrt{1-h}-\frac{1}{2\sqrt{1-h}}\mathbf{J}\Delta\mathbf{p}\right]^2 P_b\right\} \tag{A.31}$$

After differentiation:

$$\begin{aligned}
0 = \sum_{i=1}^{N}\frac{1}{P(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})|\Phi,\mathbf{p},\mathbf{y}_i)}\Bigg\{&2\left[\sqrt{h}+\frac{1}{2\sqrt{h}}\mathbf{J}\Delta\mathbf{p}\right]\frac{1}{2\sqrt{h}}\mathbf{J}P_f \\
&-2\left[\sqrt{1-h}-\frac{1}{2\sqrt{1-h}}\mathbf{J}\Delta\mathbf{p}\right]\frac{1}{2\sqrt{1-h}}\mathbf{J}P_b\Bigg\} && \text{(A.32)}\\
= \sum_{i=1}^{N}\frac{1}{P(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})|\Phi,\mathbf{p},\mathbf{y}_i)}&\left\{\left[\mathbf{J}^T+\frac{1}{2\sqrt{h}}\mathbf{J}^T\mathbf{J}\Delta\mathbf{p}\right]P_f-\left[\mathbf{J}^T-\frac{1}{2\sqrt{1-h}}\mathbf{J}^T\mathbf{J}\Delta\mathbf{p}\right]P_b\right\} && \text{(A.33)}\\
= \sum_{i=1}^{N}\frac{1}{P(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})|\Phi,\mathbf{p},\mathbf{y}_i)}&\left\{(P_f-P_b)\mathbf{J}^T+\left(\frac{P_f}{2\sqrt{h}}+\frac{P_b}{2\sqrt{1-h}}\right)\mathbf{J}^T\mathbf{J}\Delta\mathbf{p}\right\} && \text{(A.34)}\\
= \sum_{i=1}^{N}&\frac{(P_f-P_b)\mathbf{J}^T}{P(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})|\Phi,\mathbf{p},\mathbf{y}_i)} \\
+\sum_{i=1}^{N}&\frac{1}{P(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})|\Phi,\mathbf{p},\mathbf{y}_i)}\left(\frac{P_f}{2\sqrt{h}}+\frac{P_b}{2\sqrt{1-h}}\right)\mathbf{J}^T\mathbf{J}\Delta\mathbf{p} && \text{(A.35)}
\end{aligned}$$

It follows

$$\sum_{i=1}^{N}\frac{(P_f-P_b)\mathbf{J}^T}{P(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})|\Phi,\mathbf{p},\mathbf{y}_i)} = -\left[\sum_{i=1}^{N}\frac{1}{2P(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})|\Phi,\mathbf{p},\mathbf{y}_i)}\left(\frac{P_f}{\sqrt{h}}+\frac{P_b}{\sqrt{1-h}}\right)\mathbf{J}^T\mathbf{J}\right]\Delta\mathbf{p} \tag{A.36}$$

and after negation (Gauss-Newton is a minimization method, we want to maximize the probability):

$$\begin{aligned}
\Delta\mathbf{p} = &\left[\sum_{i=1}^{N}\frac{1}{2P(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})|\Phi,\mathbf{p},\mathbf{y}_i)}\left(\frac{P_f}{\sqrt{h}}+\frac{P_b}{\sqrt{1-h}}\right)\mathbf{J}^T\mathbf{J}\right]^{-1}\times \\
&\sum_{i=1}^{N}\frac{(P_f-P_b)\mathbf{J}^T}{P(\mathbf{W}(\mathbf{x}_i,\Delta\mathbf{p})|\Phi,\mathbf{p},\mathbf{y}_i)}
\end{aligned} \tag{A.37}$$

Appendix B

Results

B.1 Level Set Tracking

This section shows results for further video sequences for the level set tracker and demonstrates how the approach is able to handle challenges like topological changes (Fig. B.1), viewpoint changes (Fig. B.2), objects with several colors (Fig. B.3) and low quality video (Fig. B.4). Unless stated otherwise all sequences were obtained with the same parameter set, see table 3.1.

Figure B.1: Result for seq. MUG (2031 frames), generously provided by the authors of [BR08]. The level set representation is able to represent the contour of the mug, which contains a hole.

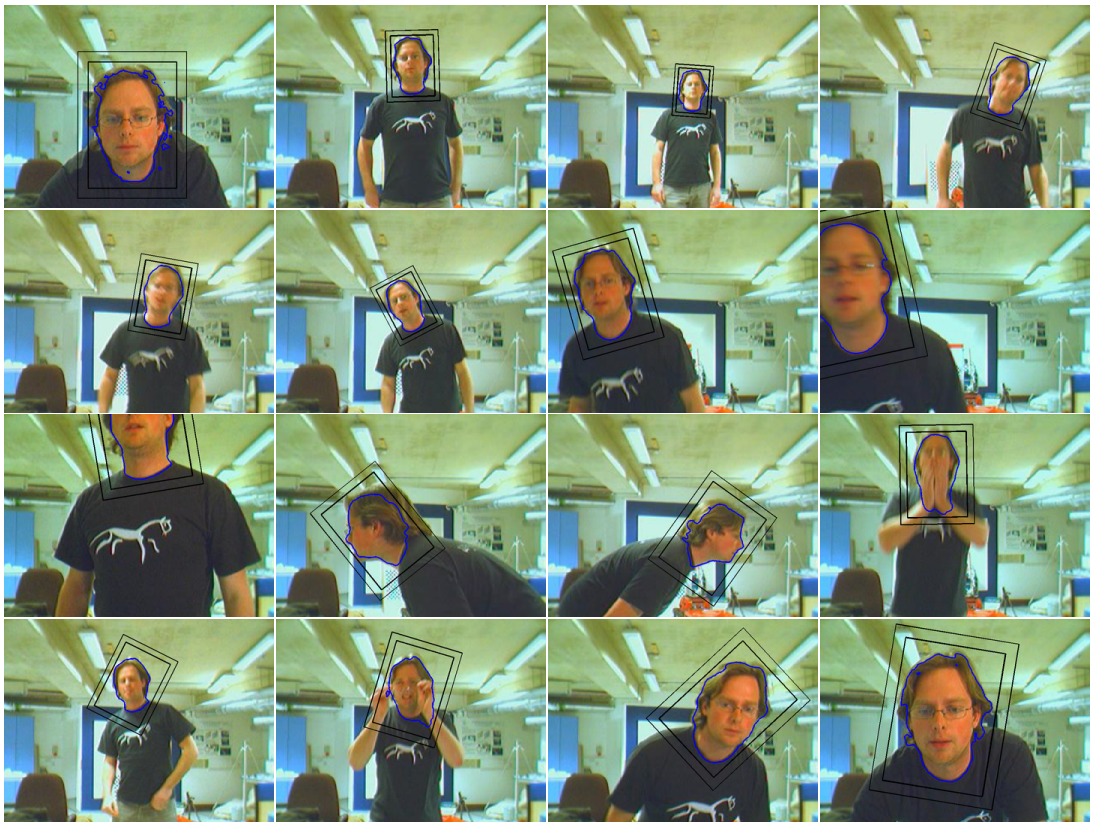


Figure B.2: Result for seq. JUMP2 (2011 frames), generously provided by the authors of [BR08]. The tracked head's viewpoint changes several times and it is sometimes occluded by the hands.

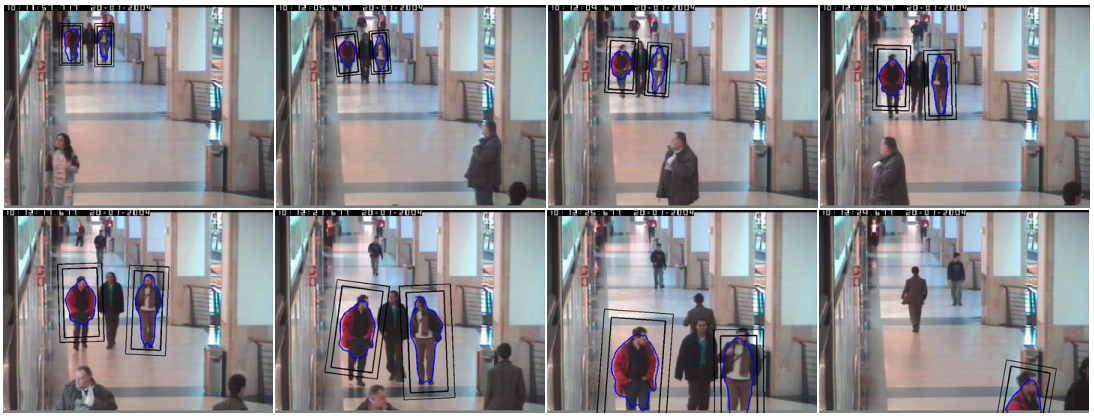


Figure B.3: Result for seq. CAVIAR (part of the seq. *ThreePastShop1cor* from the caviar data set [Fis04], 850 frames): The tracked persons appear at a large distance from the camera and wear clothes in multiple colors, which are similar to their neighbours and the background. The smoothness constraint with $\lambda = 3$ is necessary here to achieve robustness to these similar areas.

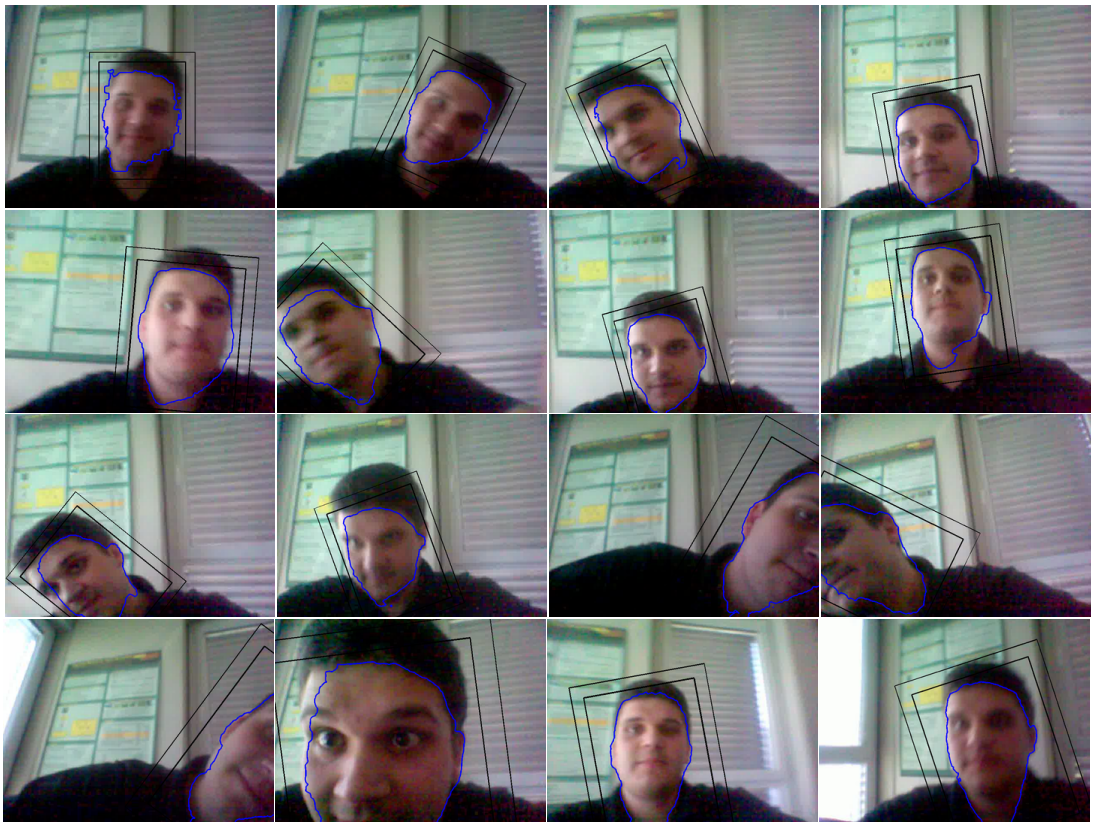


Figure B.4: Result for seq. CELL (644 frames), which was taken with the front-facing camera of a cellphone at a low resolution and frame rate (approx. 12 fps), containing sensor noise and with very fast movements.

B.2 Multi-Person Tracking

Fig. B.5 shows more results for Seq. A which includes many trees, often resulting in false positive detections. Seq. B contains many lighting changes, see Fig. B.6 for more results.

Figure B.5: Results for Seq. A (BAHNHOF). (Top rows): LS tracking results. (Bottom rows): HL tracking results. The LS tracker sometimes attempts to track false positives. Most of the time these are correctly sorted out by the HL tracker.

Figure B.6: Results for Seq. B (SUNNY DAY). (Top rows): LS tracking results. (Bottom rows): HL tracking results. The HL tracker is able to track persons over very long distances, as can be seen from the trajectories on the ground.

B.3 Geometrically Constrained Level Set Tracking

Figures B.7, B.8 and B.9 show more results for parts of Seq. CITY. We chose challenging sequences containing cars turning corners or being followed around a bend. Moreover cars in dark colours, as in the second and third sequence, are particularly difficult to track, since their colour is similar to the road.

Figure B.7: Result for Seq. CITY: (Top rows): Tracked contour and 3D transformation from the constrained homography level set tracker. (Bottom rows): Resulting trajectory from the multi-hypothesis tracker. The van is tracked successfully for 161 frames.

Figure B.8: Result for Seq. CITY: (Top rows): Tracked contour and 3D transformation from the constrained homography level set tracker. (Bottom rows): Resulting trajectory from the multi-hypothesis tracker. When the car mostly leaves the image frame the estimated angle is inaccurate and the trajectory drifts to the right, but recovers immediately when the car is completely visible again. The car is tracked successfully for 222 frames.

Figure B.9: Result for Seq. CITY: (Top rows): Tracked contour and 3D transformation from the constrained homography level set tracker. (Bottom rows): Resulting trajectory from the multi-hypothesis tracker. The LS tracker is vulnerable to lighting changes under the bridge, resulting in a wrong trajectory, but recovers and provides very accurate measurements when the tracked car turns the corner. The car is tracked successfully for 129 frames.

List of Figures

List of Tables

List of Video Sequences

We used the following video sequences for our experiments:

Generously provided by the authors of [BR08]:	
HAND	4660 frames.
JUMP	1587 frames.
JUMP2	2011 frames.
MUG	2031 frames.
CAVIAR	850 frames, part of the sequence *ThreePastShop1cor* from the caviar data set [Fis04].
CELL	644 frames, which was taken with the front-facing camera of a cell-phone.
From the Zurich Mobile Pedestrian corpus, generously provided by the authors of [ELSV09]:	
Seq. A	(BAHNHOF) 999 frames, captured with a stereo rig at 13-14fps and 640×480 resolution, containing 5193 annotated pedestrians.
Seq. B	(SUNNY DAY) 999 frames, captured with a stereo rig at 13-14fps and 640×480 resolution, 354 of which are annotated with 1867 annotations.
CITY	was captured using a stereo setup mounted on top of a car at 13-14 fps and 640×480 resolution.

Bibliography

[ARS08] M. Andriluka, S. Roth, and B. Schiele. People Tracking-by-Detection and People Detection-by-Tracking. In *CVPR*, 2008.

[BHD00] M. Betke, E. Haritaoglu, and L. S. Davis. Real-time multiple vehicle detection and tracking from a moving vehicle. *MVA*, 12:69–83, 2000.

[BM04] S. Baker and I. Matthews. Lucas-Kanade 20 Years On: A Unifying Framework. *IJCV*, 69:221–255, 2004.

[BMR05] S. Benhimane, E. Malis, and P. Rives. Vision-based Control for Car Platooning using Homography Decomposition. In *ICRA*, pages 2161–2166, 2005.

[BR08] C. Bibby and I. Reid. Robust Real-Time Visual Tracking using Pixel-Wise Posteriors. In *ECCV*, 2008.

[BR10] C. Bibby and I. Reid. Real-time Tracking of Multiple Occluding Objects using Level Sets. In *CVPR*, 2010.

[BRW05] T. Brox, B. Rosenhahn, and J. Weickert. Three-dimensional shape knowledge for joint image segmentation and pose estimation. In *DAGM*, 2005.

[BW04] T. Brox and J. Weickert. Level Set Based Image Segmentation with Multiple Regions. In *DAGM*, 2004.

[CJDC02] T. Chateau, F. Jurie, M. Dhome, and X. Clady. Real-Time Tracking Using Wavelet Representation. In *DAGM*, 2002.

[CRD07] D. Cremers, M. Rousson, and R. Deriche. A Review of Statistical Approaches to Level Set Segmentation Integrating Color, Texture, Motion and Shape. *IJCV*, 72:195–215, 2007.

[DT97] F. Dellaert and C. Thorpe. Robust Car Tracking using Kalman filtering and Bayesian templates. In *CITS*, 1997.

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[DWSP09] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian Detection: A Benchmark. In *CVPR*, 2009.

[ELSV08] A. Ess, B. Leibe, K. Schindler, and L. Van Gool. A Mobile Vision System for Robust Multi-Person Tracking. In *CVPR*, 2008.

[ELSV09] A. Ess, B. Leibe, K. Schindler, and L. Van Gool. Robust Multi-Person Tracking from a Mobile Platform. *PAMI*, 31(10):1831–1846, 2009.

[Eva10] L.C. Evans. *Partial Differential Equations*. AMS Bookstore, 2010.

[FDP06] M. Fussenegger, R. Deriche, and A. Pinz. Multiregion Level Set Tracking with Transformation Invariant Shape Priors. In *ACCV*, 2006.

[FH06] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Belief Propagation for Early Vision. *IJCV*, 70(1):41–54, October 2006.

[Fis04] R. B. Fisher. CAVIAR test case scenarios. `http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/`, 2004.

[GGB06] H. Grabner, M. Grabner, and H. Bischof. Real-Time Tracking via On-line Boosting. In *BMVC*, 2006.

[GM07] D. Gavrila and S. Munder. Multi-Cue Pedestrian Detection and Tracking from a Moving Vehicle. *IJCV*, 73(1):41–59, 2007.

[GW08] R. C. González and R. E. Woods. *Digital Image Processing*. Prentice Hall, 2008.

[HEH06] D. Hoiem, A.A. Efros, and M. Hebert. Putting Objects in Perspective. In *CVPR*, 2006.

[HMLE10] E. Horbert, D. Mitzel, B. Leibe, and A. Ess. Geometrically Constrained Level Set Tracking for Automotive Applications. In *DAGM (to appear)*, 2010.

[HWN08] C. Huang, B. Wu, and R. Nevatia. Robust Object Tracking by Hierarchical Association of Detection Responses. In *ECCV*, 2008.

[HZ00] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2000.

[Jac01] K. Jack. *Video Demystified: a Handbook for the Digital Engineer*. LLH Technology Pub., 2001.

[KDN93] D. Koller, K. Daniilidis, and H. Nagel. Model-Based Object Tracking in Monocular Image Sequences of Road Traffic Scenes. *IJCV*, 10(3), 1993.

[KPB+05] R. Kaucic, A.G. Perera, G. Brooksby, J. Kaufhold, and A. Hoogs. A Unified Framework for Tracking through Occlusions and Across Sensor Gaps. In *CVPR*, 2005.

[LCC+07] B. Leibe, N. Cornelis, K. Cornelis, , and L. Van Gool. Dynamic 3D Scene Analysis from a Moving Vehicle. In *CVPR*, 2007.

[LSV08] B. Leibe, K. Schindler, and L. Van Gool. Coupled Object Detection and Tracking from Static Cameras and Moving Vehicles. *PAMI*, 30(10):1683–1698, 2008.

[LXGF05] C. Li, C. Xu, C. Gui, and M.D. Fox. Level Set Evolution without Re-initialization: A New Variational Formulation. In *CVPR*, 2005.

[MB99] J. MacCormick and A. Blake. A Probabilistic Exclusion Principle for Tracking Multiple Objects. In *ICCV*, pages 572–578, 1999.

[MHLE10] D. Mitzel, E. Horbert, B. Leibe, and A. Ess. Multi-Person Tracking with Sparse Detection and Continuous Segmentation. In *ECCV (to appear)*, 2010.

[MN87] P.F. Muir and C.P. Neuman. Kinematic Modeling of Wheeled Mobile Robots. *JRS*, 4(2):281–340, 1987.

[Nis04] D. Nistér. A Minimal Solution to the Generalised 3-Point Pose Problem. In *CVPR (1)*, pages 560–567, 2004.

[NM90] M. Nitzberg and D.B. Mumford. The 2.1-D sketch. In *ICCV*, 1990.

[NNB04] D. Nistér, O. Naroditsky, and J.R. Bergen. Visual Odometry. In *CVPR (1)*, pages 652–659, 2004.

[NNB06] D. Nistér, O. Naroditsky, and J.R. Bergen. Visual Odometry for Ground Vehicle Applications. *J. Field Robotics*, 23(1), 2006.

[OP96] S. Osher and N. Paragios. *Geometric Level Set Methods in Imaging, Vision and Graphics*. Springer, 1996.

[OS88] S. Osher and J. Sethian. Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[OTdF+04] K. Okuma, A. Taleghani, N. de Freitas, J. Little, and D. Lowe. A Boosted Particle Filter: Multitarget Detection and Tracking. In *ECCV*, 2004.

[PR09a] V.A. Prisacariu and I.D. Reid. fastHOG – a Real-Time GPU Implementation of HOG. Technical Report 2310/09, Dept. of Engineering Science, University of Oxford, 2009.

[PR09b] V.A. Prisacariu and I.D. Reid. PWP3D: Real-time segmentation and tracking of 3D objects. 2009.

[SC09] T. Schoenemann and D. Cremers. A Combinatorial Solution for Model-based Image Segmentation and Real-time Tracking. *PAMI*, 2009.

[Set96] J.A. Sethian. *Level Set Methods*. Cambridge University Press, 1996.

[Set99] J.A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[SG99] C. Stauffer and W.E.L. Grimson. Adaptive Background Mixture Models for Realtime Tracking. In *CVPR*, 1999.

[SS02] D. Scharstein and R.S. Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *IJCV*, 47(1-3):7–42, apr 2002.

[TSK02] H. Tao, H.S. Sawhney, and R. Kumar. Object Tracking with Bayesian Estimation of Dynamic Layer Representations. *PAMI*, 24(1):75–89, 2002.

[WN07] B. Wu and R. Nevatia. Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet Part Detectors. *IJCV*, 75(2):247–266, 2007.

[ZN08] L. Zhang and R. Nevatia. Global Data Association for Multi-Object Tracking Using Network Flows. In *ECCV*, 2008.

[ZNF09] C. Zach, M. Niethammer, and J. M. Frahm. Continuous Maximal Flows and Wulff Shapes: Application to MRFs. In *CVPR*, pages 1911–1918, 2009.

[ZNW08] T. Zhao, R. Nevatia, and B. Wu. Segmentation and Tracking of Multiple Humans in Crowded Environments. *PAMI*, 30(7):1198–1211, 2008.