Learning Dynamic Shape Models for Bayesian Tracking

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von
Dipl.-Inf. Jan Giebel
aus Georgsmarienhütte

Mannheim, 2005

Dekan: Prof. Dr. M. Krause, Universität Mannheim
Referent: Prof. Dr. C. Schnörr, Universität Mannheim
Korreferent: Prof. Dr. D.M. Gavrila, Universität Amsterdam

Tag der mündlichen Prüfung: 25. Oktober 2005

Abstract

This Thesis addresses appearance-based techniques for modelling complex deformable objects in order to support a detection and tracking process. As example application it considers pedestrian tracking from a moving vehicle.

The a-priory knowledge about the object class is captured by a novel probabilistic spatio-temporal shape representation, which consists of a set of distinct linear subspace models and handles continuous as well as discontinuous shape changes. An un-supervised learning algorithm is presented, which fully automatically derives the proposed representation from training sequences of closed contours. In contrast to previous work, no prior feature correspondences are required.

The 2D representation is applied to improve the performance of matching systems that correlate using shape templates. The basic idea involves extending an existing set of training shapes with generated "virtual" shapes, in order to improve the representational capability. In the experiments ROC curves demonstrate that generating "virtual" shapes by sampling the linear subspace models of the proposed representation increases the detection rate while simultaneously reducing the false alarm rate of a correlation-based object detection system.

A framework for multi-cue 3d object tracking is introduced, which approximates optimal Bayesian tracking by a particle filter and combines the three cues shape, texture and depth, in its observation density function. The framework integrates an independently operating object detection system by means of importance sampling. In the experiments the approach is evaluated on the challenging topic of pedestrian tracking from a moving vehicle in synthetic, rural, and complex urban environments. It is shown on thousands of images, that the proposed approach improves upon a cutting edge pedestrian protection system, that was designed for the European community funded project PROTECTOR.

Zusammenfassung

Diese Dissertation befasst sich mit ansichtsbasierten Methoden zur Modellierung deformierbarer Objekte, um Detektions- und Trackingaufgaben zu unterstützen. Die behandelten Methoden werden zum Erkennen und Verfolgen von Fußgängern in der realen Welt anhand von Videobildern aus einem Fahrzeug eingesetzt.

Das a-priori Wissen über die Objektklasse wird von einem neuen raum-/zeitlichen Konturmodell repräsentiert, das aus einer Menge linearer Unterraummodelle besteht und sowohl kontinuierliche als auch diskontinuierliche Konturänderungen abbilden kann. Ein unüberwachtes Lernverfahren wird vorgestellt, welches die Repräsentation automatisch aus Trainingssequenzen geschlossener Konturen bildet. Im Gegensatz zu den in der Literatur üblichen Methoden, setzt der vorgestellte Ansatz keine Punktkorrespondenzen zwischen den Konturpunkten zum Lernen voraus.

Die zweidimensionale Repräsentation eignet sich dazu, Detektionssysteme zu verbessern, die mittels Korrelation Konturen vergleichen. Die zugrundeliegende Idee besteht darin, eine vorliegende Trainingsmenge mit "virtuellen" Beispielen anzureichern, um ihre Aussagekraft zu verbessern. In den Experimenten wird anhand einer ROC Kurve gezeigt, daß das Generieren von "virtuellen" Beispielen anhand der linearen Unterräume des vorgestellten Formmodelles die Detektionsrate eines korrelationsbasierten Erkennungssystems erhöht, währen gleichzeitig die Rate der Fehlerkennungen sinkt.

Ein Verfahren zum Verfolgen von Objekten im dreidimensionalen Raum wird vorgestellt, welches die optimale Bayesische Lösung mit einem Parikelfilter approximiert und sowohl Form-, Textur- als auch Stereomerkmale der Bilder berücksichtigt. Ferner werden Messungen eines unabhängigen Objekterkennungssystemes als sogenannte "Importance" Funktion integriert. In den Experimenten wird der Ansatz auf dem herausfordernden Gebiet der Fußgängererkennung aus einem Fahrzeug in Test-, ländlichen und komplexen urbanen Umgebungen evaluiert. Anhand tausender Bilder wird gezeigt, daß der verfolgte Ansatz ein Fußgängererkennungssystem, das im Rahmen des Europäischen PROTECTOR Projektes entwickelt wurde, verbessert.

Acknowledgments

At this point I would like to thank all the people who supported my dissertation. First of all I would like to express my gratitude to my supervisors, professor D.M. Gavrila and professor C. Schnörr. Professor Gavrila for kindly supporting this work in many ways. Without his helpful suggestions, precise reviews, motivation, and introduction to secrets of Chamfer matching, this work would have been impossible. Professor Schnörr for taking the supervision of the dissertation and his kind overall support.

I am also very grateful to my colleagues and friends at the image understanding department of DaimlerChrysler research in Ulm.

Contents

List of Figures

1 Introduction

1.1 Motivation and Overview

For many interesting computer vision tasks there are no explicit prior models available, which capture object appearance and motion. This is typically the case for complex non-rigid objects under unrestricted viewpoints and/or under changing illumination conditions. This Thesis deals with general statistical spatio-temporal representations that can be learned from examples and are suitable to support object detection and tracking tasks. See Figure 1.1.

Many standard pattern classification techniques (e.g. for dimensionality reduction or clustering) require a single feature space in which all training samples are embedded. However, for complex objects with a high variance in appearance, it is often unrealistic to find correspondences between all features among the complete training set. In this work an integrated registration and clustering approach is presented, which only registers similar shapes into common feature spaces. The shapes of the automatically derived spaces are described by linear subspace models or "Point Distribution Models" (PDMs). A discrete first order Markov model controls the temporal transitions between them. The spatio-temporal representation is called "Dynamic Point Distribution Models" (DPDMs) throughout this work.

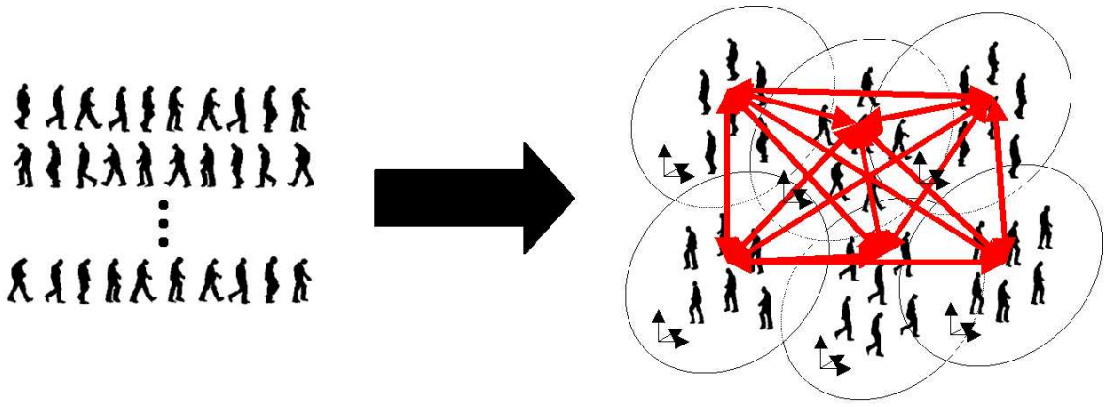

Figure 1.1: Learning "Dynamic Point Distribution Models". A set of linear subspace models with temporal transition probabilities between them is derived from training sequences.

For such object representations it is important that the training sets adequately cover the distribution of object appearance. This is in particular the case for exemplar-based representations, which naturally do not generalize. Yet the ability to collect training samples is limited for many applications and often involves time consuming manual labeling. In this work an automatic method for extending training sets of (unregistered) shapes with generated "virtual" shapes is presented, so that the extended training sets fill representational gaps in the target distribution. Refer Figure 1.2. This is achieved

by drawing samples from the learned DPDMs in a probabilistic manner.

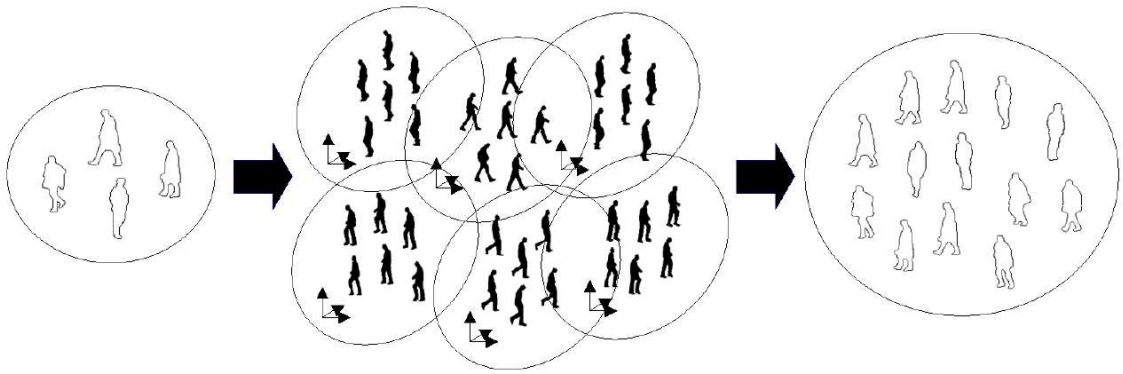

Figure 1.2: Virtual sample generation. A set of "Point Distribution Models" is learned and re-sampled to obtain "virtual" shapes. They fill representational gaps in the target distribution.

The main application of DPDMs is probabilistic object tracking. See Figure 1.3. While pure detection systems perform a search in each image of a sequence separately, tracking systems integrate their results over time. They are therefore more reliable and robust, especially in case of occlusions or heavy cluttered scenes. Object trajectories can be calculated to identify typical situations or gestures. Tracking systems typically iterate a "prediction" and a "correction" step. The first predicts the location and configuration of all tracked objects according to their trajectories and motion models. The second corrects the predictions with the actual measurements. To increase the reliability of the proposed tracking system, the three visual cues shape, texture, and stereo are combined in the measurement step in this work.

1.2 Applications

The ideas and methods developed in this Thesis can be applied to a wide variety of practical applications from various domains. They are particularly suited (but not limited) for applications with the following characteristics:

- The objects of interest have a complex structure and undergo significant appearance changes over time.
- No explicit models of appearance and motion exist for them. But a set of example sequences is available or can be generated to train such models.
- For object detection and tracking, simpler background subtraction techniques do not apply due to uncontrolled, cluttered environments.

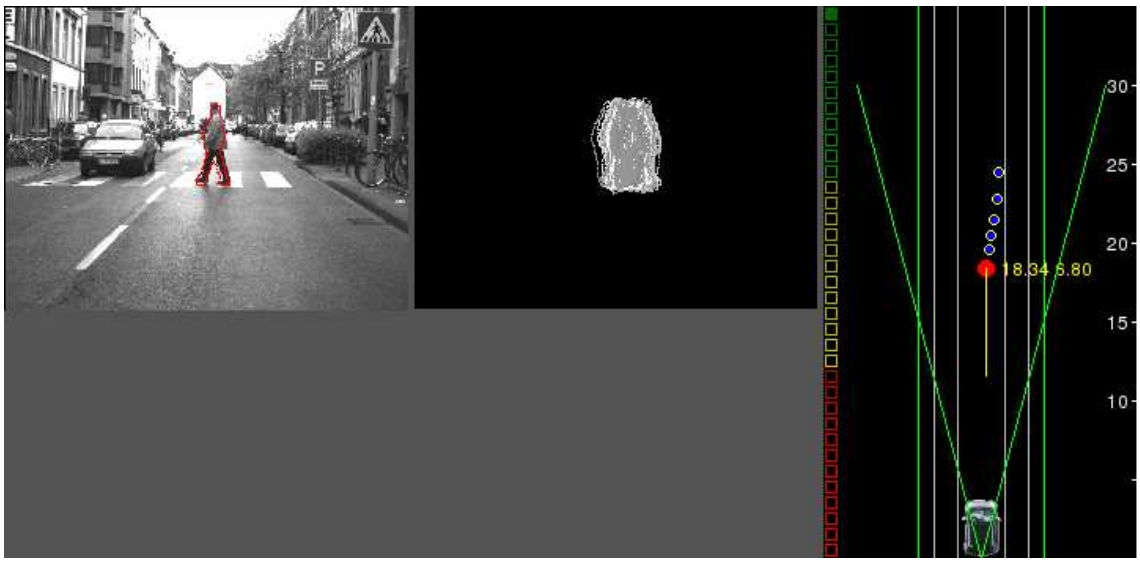

Figure 1.3: Object tracking. A pedestrian is tracked in urban environment. The actual solution of the tracker is outlined in the left image. In the middle, several object hypothesis are plotted in the image plane. The right image shows a top view of the scene. The pedestrian trajectory is represented by blue dots, while the actual solution is indicated by the red dot.

Some examples are listed below.

Medicine In the area of medicine images from various different sensors (ultrasound, radioscopy, computer tomography, positron emission tomography, etc.) have to be processed to improve diagnosis and treatment of physicians. Motion compensation in coronary angiography [42], tracking the non-rigid motion of the heart [106] are only two examples of medical applications.

Surveillance Reliable pedestrian tracking is a precondition for many surveillance tasks and has been considered for indoor [18] and outdoor [15, 20] environments using single [15, 20, 71] or multiple camera systems [18].

Robotics In robotics advanced tracking techniques are for example used for obstacle avoidance [105] and grasping certain objects. In cases, in which robots have to deal with complex, flexible objects, the techniques discussed in this work are of interest.

Driver assistance and safety systems To improve the comfort and safety of drivers, several assistance and safety systems have been developed, which all contain tracking components involving flexible models. Vehicle following systems for high speeds on highways (intelligent cruise control) and low speeds in traffic jams (stop and

go) have been considered. In so called "platooning" applications several vehicles automatically follow a leading one.

Gesture recognition The problem of gesture recognition has been studies by many researchers [8, 102, 126] . It is preliminary for many higher level applications, like person identification or advanced human computer interfaces.

Augmented reality Computer vision systems, which augment real world images with synthetic objects, can benefit from the advanced tracking capabilities discussed in this work. For example, if synthetic non-rigid objects have to be replaced by artificial figures, the target locations in the image have to be robustly tracked.

Visions Newer trends include for example interactive rooms [32], in which persons are tracked and several actions are initiated automatically.

This work deals with vision-based driver assistance systems for pedestrian protection. The aim is to develop intelligent sensor systems, which detect potentially dangerous situations involving pedestrians ahead of time, in order to either warn the driver, or, if no such time remains, initiate appropriate emergency measures (e.g. braking). See Figure 1.4 and the experiments in Chapter 5.

1.3 Related Work

1.3.1 Spatial and Spatio-Temporal Representations

Several representations have been proposed to cover the distribution of object appearance and motion, for example in the area of "Looking at People" [1, 56, 107]. The quality of such a representations is mainly determined by their accuracy and specificity. They should contain all valid appearance and motion and simultaneously exclude all invalid.

In general such models may be defined in 2D [6, 9, 22, 24, 54, 61, 110] or 3D [33, 36, 57]. In computer graphics often explicit 3D models are applied to render photo realistic images [47]. They often involve thousands of parameters (for example the position of polygon points). In computer vision however, the hard and ill posed problem of recovering 3D information from 2D images has to be solved. To speed up computation, 3D models of limited detail are mostly considered, which typically consist of a skeleton and a flesh model. For real time systems, 2D approaches are often favored, which typically involve only a few parameters. More efficient matching techniques exist to compare model instances with images, since no time consuming 3D to 2D projection is involved. Furthermore learning 2D models is much easier than deriving 3D models from examples.

Representations may be parametric [6, 33, 36, 54, 57] or non-parametric [60, 144]. Non-parametric exemplar-based representations cover the distribution of object appearance by a set of samples. They are accurate and easy to extend but have the disadvantage

Figure 1.4: Pedestrian tracking from a moving vehicle. The topic is extremely challenging because of the moving camera, the wide range of possible object appearances, the unknown number of targets, cluttered backgrounds, and hard real time constraints.

that they do not generalize and they are not compact, since all model instances have to be stored in a database. Various parametric representations have been designed for computer vision tasks ranging from hand crafted deformable templates to complex statistical models. These models generalize object appearance in the "valid" parameter space. However, invalid parameter combinations might occur and reduce model specificity.

1.3.2 Bayesian Tracking

In computer vision, tracking methods establish correspondence between image structures in consecutive frames. The structures may be simple features (e.g. corners or edges) [136], contours (e.g. splines) [12], regions (e.g. blobs) [86], or more complex objects described by appearance and/or motion models [10, 22].

In a typical tracking system, the parameters of interest (e.g. position, velocity, object

configuration) are collected in a state vector, which is estimated recursively time. See Figure 1.5. Starting with the actual state estimate, a dynamical model is applied to predict the state vector in time. From the predicted state parameters, object appearance in the image space can be synthesized. A measurement model is then applied to correct the prediction with the actual observation, yielding the new state estimate.

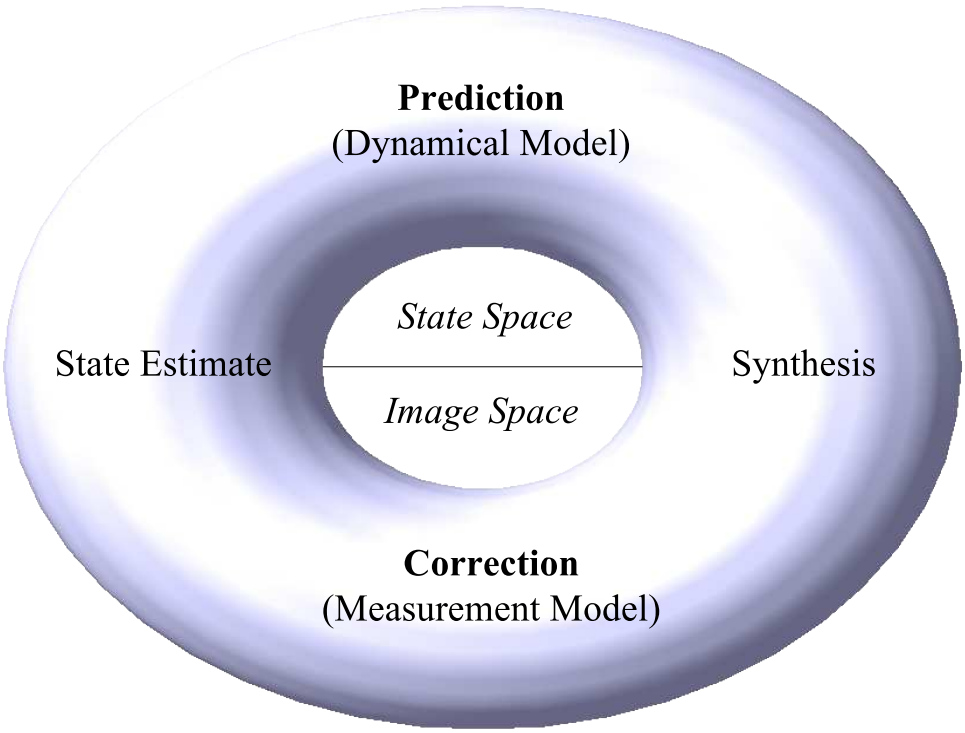

Figure 1.5: Tracking Loop. Starting with the actual state estimate, a dynamical model is used to predict the state in time. From the state vector, object appearance in the image space can be synthesized. A measurement model is applied to correct the prediction with the actual observation, yielding the new state estimate.

Tracking can be stated deterministically as optimization or probabilistically as estimation problem and can be solved by various techniques such as simulated annealing, dynamic programming [120], the simplex algorithm [119], genetic algorithms, mean shift approaches [21], level set approaches [134], least squared methods, Kalman filters [91, 104], sequential Monte Carlo methods [39]. Probabilistic approaches propagate complete probability density functions over the state space in time. Depending on whether they can process one or more modes in the involved probability density functions, they

are called uni- or multi-modal.

In this work recursive Bayesian tracking techniques are considered [3]. Especially particle filters are of interest, because of their great ease and flexibility in approximating complex probability density functions (pdfs), and dealing with a wide range of dynamical and measurement models. Their multi-modal nature makes them particularly suited for object tracking in cluttered environments, where uni-modal techniques might get stuck and loose track.

1.3.3 Tracking in Literature

In Appendix A a selection of recent literature on vision-based tracking systems is listed. Although the listing comprises about 90 documents, it is far from being complete. The vast amount of documents available reflects the enormous interest in the computer vision community in this domain. Most publications are taken from the "IEEE Transactions on Pattern Analysis and Machine Intelligence", the "International Journal of Computer Vision", the proceedings of the "European Conference on Computer Vision", the"Computer Vision and Pattern Recognition Conference", and the "International Conference on Computer Vision".

Several criteria could be defined to classify research in tracking. In the Appendix, it is classified whether it involves deformable objects, includes probabilistic or learned components, tracking is done in a closed loop (tracking results of one time step are used as prior information for the next), multiple features are considered, multiple modes are maintained, and whether it runs in real time. In addition the application area is given.

1.4 Contribution and Organization

This work presents a general spatio-temporal shape representation. It consists of a set of linear subspace models, which capture object appearance. Due to this "locally linear" approach an improved model specificity compared to standard single subspace models can be expected. The temporal transition probabilities between these models are stored in a discrete Markov model. See Figure 1.1. The model is general, since it can be learned from example sequences of closed contours. It captures continuous shape changes, corresponding to smooth parameter changes within a linear subspace model, as well as discontinuous, corresponding to model transitions.

The representation is introduced in Chapter 3 and builds upon the locally linear shape representation of [74]. In contrast to previous work, it can be fully automatically learned from training data. No prior feature space is required nor established over the entire training set.

A particle filtering framework for multi-cue 3D object tracking is presented in Chapter 4. It is able to cope with "mixed" discrete/continuous state spaces, which allows a joint estimate of the discrete subspace membership and the continuous shape parameters of

the proposed spatio-temporal representation. In contrast to previous work, object position and velocity are tracked directly in 3D space which simplifies occlusion reasoning and the assignment of measurements to tracks. The framework integrates the three visual cues shape, texture, and stereo for improved robustness. Texture is modeled by means of histograms similar to [111, 115]. but it is not based on circular/rectangular region primitives. Instead, the detailed shape information is used to derive appropriate object masks for texture extraction. Finally, the tracking framework integrates an independently operating object detection system by means of importance sampling.

In the experiments presented in Chapter 5, the spatio-temporal representation is trained on thousands of pedestrian shapes. It will be demonstrated that extending the training set by random samples from the trained representation will produce an extended training set, which facilitates superior matching performance, measured by a ROC curve using a similarity measure based on distance transforms.

The tracking framework is evaluated on the challenging application of pedestrian tracking from a moving vehicle. It is shown empirically on sequences derived from hours of driving in urban environment, that a system based on the new particle filtering framework improves an existing pedestrian protection system, that was designed for the European community funded project PROTECTOR.

2 Shapes in Motion - Background

2.1 Spatial and Spatio-Temporal Representations

A large variety of shape-based representations have been designed to describe object appearance and motion for image understanding applications. Detailed overviews can be found in well elaborated textbooks [139]. In the following only those approaches, which form the basis of this Thesis are briefly reviewed.

2.1.1 Contour Description

From a practical point of view, the most convenient way to represent contours is to enumerate their u and v pixel coordinates dependent on the boundary length l

$$c(l) = \begin{cases} u(l) \\ v(l) \end{cases} \tag{2.1}$$

In this work the N pixel coordinates of a contour are collected in a $2N$ dimensional shape vector

$$\vec{s} = (u(0), v(0), u(1), v(1), ..., u(N-1), v(N-1)) \tag{2.2}$$

Several further descriptions have been considered to represent single contours. Often, they are decomposed into segments, described by a polynomials [47]. In the simplest case, polynomials of degree one lead to polygonal curve approximations. (Uniform, non-rational) B-splines are quite popular in computer graphics applications, because of their smoothness properties. They are are an example for piecewise third degree polynomials. Each segment is described by four control points. To achieve the C^2 continuity, each segment contains the last three control points of the previous one. The control points of B-splines define the convex hull of the curve, but the control points are in general not interpolated. Hermite-splines are another example for piecewise third degree polynomials [47, 52]. Each segment is also described by four control points. In contrast to B-splines they have an intuitive meaning, which is for example advantageous for interactive delineation tasks. They define the start and end points of the segments and their tangents. C^1 continuity is achieved, when each segment contains the last two control points of the previous one. While the start and end points are interpolated, the convex hull can not easily be derived from the control points.

Fourier representations [88] have also been considered for contours. While the basis functions for splines are polynomials, Fourier approximations use sinusoidal basis functions. Arbitrary contours can be approximated in any desired detail, given a sufficient number of basis functions. Refer Section 3.1.1.

2.1.2 Registration

Automatic shape registration methods bring the points of two shapes into correspondence. A review of previous work shows that a typical procedure follows a similar

succession of steps: shape decomposition, feature selection, point correspondence and finally, alignment.

The first step, shape decomposition, involves determining control ("landmark") points along a contour and breaking the shape into corresponding segments. One way of doing this is to determine the locations of minima and maxima of the curvature function along the object's contour. The curvature function is computed by convolving the edge direction function with the first derivative of a Gaussian. The variance parameter σ of the Gaussian function determines the smoothing of a shape, see for example [99, 109, 122]. A different method for determining control points is described in [159], where points are removed according to a "criticality" measure based on the area of three successive points.

Once shape segments have been determined, the next step involves selecting features that are transformation invariant (e.g. translation, rotation and scale). These features are usually selected based on an approximation of the shape segments; most common are straight-line (i.e. polygonal) [43, 61, 77, 99] and Fourier approximations [65]. Similarity measures are typically based on length ratios and angles between successive segments for the polygonal case (e.g. [99]) and weighted Euclidean metrics on the low-order Fourier coefficients, for the Fourier approximations (e.g. [65]).

At this point, correspondence between the control points of two shapes can be established by means of either combinatoric approaches or sequential pattern matching techniques. Each correspondence is evaluated using match measures on the local features discussed earlier. Combinatoric approaches [43, 99] select iteratively a (e.g. minimal) set of initial correspondences and use various greedy methods to assign the remaining points. The advantage of this approach is that the overall goodness of a match can be based on all contour points simultaneously. Additional effort is necessary, though, to account for ordering constraints. Sequential pattern matching techniques, on the other hand, inherently enforce ordering constraints. Dynamic programming is widely used [61, 65], because can efficiently solve these problems recursively by storing the solutions of re-occurring sub-problems.

Finally, after correspondence between control points has been established (and possibly, between interpolated points), alignment with respect to similarity transformation (e.g. scale, rotation, translation) is achieved by a least-squares fit [24]. The remaining non-rigid shape variation between multiple registered shapes are statistically encoded with the methods outlined in the next section. The above techniques for shape registration have important applications for partial curve matching in object recognition tasks [61, 65, 99].

In previous work [54] a comparative study on different registration technique concludes, that a segmentation technique based on the maxima of the curvature function, combined with a similarity measure derived from the Fourier coefficients, yields the best results of the methods under consideration. These techniques build the basis for the approach followed in this work.

2.1.3 Contour Classes and Shape Variability

To treat the problem of object variability several representations have been proposed to cover distributions of object appearance and motion, which serve as prior knowledge for various computer vision tasks. The quality of such representations is mainly determined by their *accuracy*, *specificity*, *compactness*, and *simplicity*. A model is accurate, if it contains all valid examples. It is specific, when it excludes all invalid ones. The compactness is determined by the number of parameters, the simplicity by their intuitive meaning and ease of use.

Exemplar-based Representations

The easiest way to represent contour classes is store all possible instances into a common database. Such exemplar-based representations have for example been considered in [53, 87, 144]. Exemplar-based modeling has the advantage, that the training set does not have to be embedded into a common vector-space, which typically means manual labeling or the development of complex automatic registration procedures [6, 43, 77].

In Gavrila [53] contours are structured in a hierarchy according to their similarity. For object detection tasks, it is then no longer necessary to match each contour at all possible image locations. Instead matching is done in a coarse to fine approach over the hierarchy to speed up computation. Those exemplar-based models, which are not embedded into a vector-space, do not permit the use of standard probabilistic learning algorithms for example principle component analysis or k-means and matching is mostly restricted to pairwise dissimilarities, which is for example the case for the "chamfer" [78] and the "shuffle" [144] distance. To overcome these problems, a probabilistic "Metric Mixture" model for those exemplar-based representations was recently introduced by Toyama and Blake [144] and applied to tracking in a stochastic framework.

Parametric Representations

Various shape-based parametric models have been designed for computer vision tasks. Dynamic templates [155], snakes [92], articulated models, Fourier series approximations [142], finite element models [130], and statistical models [24, 64], are only some examples. Parametric modeling offers the possibility to use standard probabilistic learning techniques, to estimate distributions over the parameter space, such as PCA or k-means. Continuous shape changes can then easily be modeled by smooth variations of the parameters in the "valid" parameter space.

Compact low-dimensional object parameterizations can be obtained by *linear subspace* techniques (Figure 2.1) based on shape [24], texture [7, 148], motion [10], or combinations [6, 22] and are widely used in the computer vision community. However, these methods have some limitations. One concerns the global linearity assumption: nonlinear object deformations have to be approximated by linear combinations of the eigenvectors. Therefore linear subspace models are not the most compact representations for

objects undergoing complex (non-linear) deformations. They are also not specific, since implausible instances can be generated, when invalid weights for the eigenvectors are used.

To treat the problem of linearity, Bregler and Omohundro [17] developed a method to learn nonlinear surfaces from examples. The object manifold is approximated by the union of a set of *locally-linear* (Figure 2.1) surface patches. A drawback of this approach is, that feature correspondences over the entire training set have to be established for learning. Heap and Hogg used this idea to build their "Hierarchical Point Distribution Model" [73], which they extended to deal with discontinuous changes in object appearance for tracking [74].

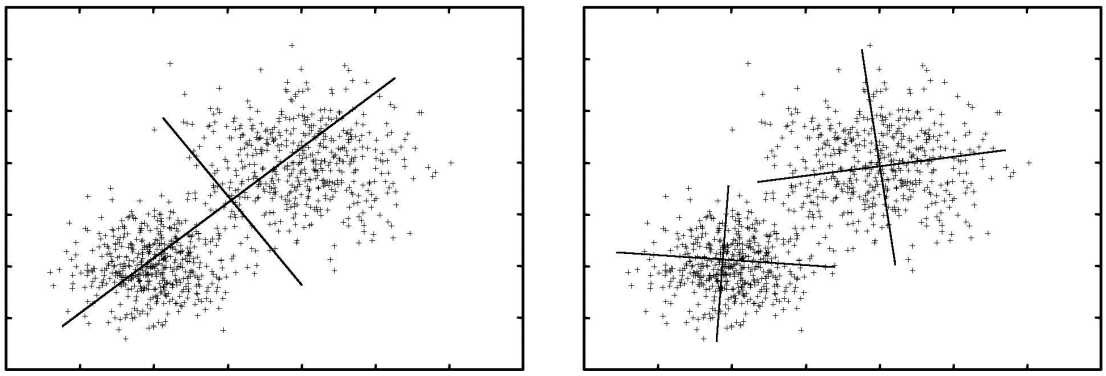

Figure 2.1: Feature spaces: linear and locally-linear feature spaces.

Probabilistically, appearance is often assumed to be multivariate Gaussian distributed in feature spaces, which is a useful approximation for a wide variety of applications. However, complex non-linear variation caused by object deformation or changes in viewpoint, may require more detailed probability density estimates to improve model specificity, if captured by a linear model. See Figure 2.2. In [23, 108] for example, a weighted mixture of Gaussians is fitted via the Expectation Maximization (EM) algorithm to the training data, which can in principal model arbitrarily complex densities. A different approach based on kernel methods [26, 132] considers a Gaussian distribution after a non-linear mapping of the training data to a potentially higher dimensional feature space. This allows to model complex highly non-linear densities in the input space.

2.1.4 Motion Models

To capture object dynamics, a rich variety of stochastic models were proposed. Overviews can for example be found in [4, 12].

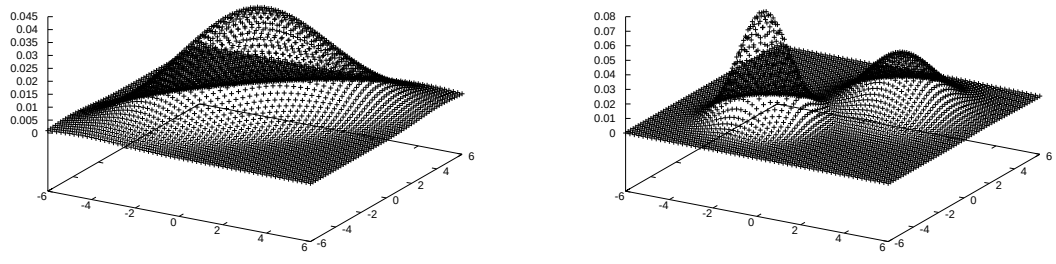

Figure 2.2: Probabilistic modeling: a single and a mixture of two multivariate Gaussians fitted via the expectation maximization algorithm to the two-dimensional data of Figure 2.1.

Stochastic Dynamical Models

In the context of contour tracking discrete time parametric linear dynamic models are often favored for computational reasons.

For *continuous* state spaces, auto-regressive processes (ARP), which are linear Gaussian Markov models, are probably the most popular dynamical models for contours [12, 74, 101]. They capture a rich variety of motion, are easy to implement, efficient to calculate, and learnable from examples. The state x at time t is modeled by an nth order ARP as a linear combination of the n previous states with additional noise

$$\vec{x}_t = \sum_{i=1}^{n} A_i \vec{x}_{t-i} + \vec{c} + B\vec{w}_t, \tag{2.3}$$

where A_i, B, and $\vec{c}$ are user defined or learned dynamical parameter and w_t is the $d \times 1$ stochastic process.

For *discrete* state spaces, hidden Markov models are frequently applied to computer vision tasks [120, 139]. A dynamic system described by a discrete Markov model may be in one of a finite number of states. According to the Markov assumption, the probability of being in a certain state at a certain time only depends on the previous n states. n is called the order of the Markov model.

Complex motion is often decomposed into several motion classes for which simpler models can be applied [4, 16, 84, 110, 114]. (Consider the complex human motion: different motion classes for walking, swimming, dancing, and creeping could be defined.) A common way is to consider auto-regressive processes for the individual motion classes, while class transitions are captured by discrete Markov models. Such models have for example been used in combination with dynamic Bayesian networks [114], Kalman [4] and particle filters [84, 110].

Unfortunately, it is not always straightforward to find distinct motion classes for complex objects and the computational complexity of learning them from examples is reported to be problematic [110].

2.2 Bayesian Tracking

Bayesian techniques are frequently applied to visual tracking. They provide a sound mathematical foundation for the derivation of (posterior) probability density functions (pdf) in dynamical systems. The evolution of the pdf can in principle be calculated recursively by optimal Bayesian filtering. Analytical solutions for the optimal Bayesian filtering problem are known only for certain special cases, including Kalman and hidden Markov model filtering. For others, approximate techniques have been developed, such as extended Kalman [4], particle [12, 39], and unscented Kalman filters [90, 149].

An overview of probabilistic tracking algorithms is given in Table 2.1. They are structured into uni-/multimodal approaches and whether they are parametric/non-parametric. While *uni-modal* techniques consider only one mode of the probability density function, *multi-modal* techniques maintain several of them over time. Multiple modes in the probability density function may for example arise due to ambiguities caused by insufficient measurements or multiple targets. In the first case, it is desirable to track these modes till the ambiguity can be resolved in time. In case of multiple targets, all modes should accurately be maintained.

Probabilistic estimation techniques may be parametric or non-parametric. *Parametric* techniques estimate the parameters of a distribution of known type, for example the mean and covariance of Gaussian or the parameters of a mixture of Gaussians (MOG). *Non-parametric* methods on the other hand offer the possibility to approximate distributions of any type by a set of samples.

The notation for Bayesian tracking systems followed in this Thesis is based on [12] and [39]. It is summarized in Table 2.2.

2.2.1 Bayesian Filtering Fundamentals

The goal of recursive Bayesian filtering is to estimate the complete a posteriori probability density $p(\vec{x}_t|Z_t)$ at time t of the state $\vec{x}_t \in \mathcal{S}^{d_x}$ of a dynamic system, which is observed by noisy measurements $\vec{z}_t \in \mathcal{S}^{d_m}$.

The state vector is assumed to evolve according to a system model

$$\vec{x}_t = f_t(\vec{x}_{t-1}, \vec{v}_{t-1}) \tag{2.4}$$

a function $f_{t-1} : \mathcal{S}^{d_x} \times \mathcal{S}^{d_v} \rightarrow \mathcal{S}^{d_x}$ of the previous state $\vec{x}_{t-1}$ and the process noise $\vec{v}_{t-1} \in \mathcal{S}^{d_v}$ of a known distribution Q_t.

The measurements

$$\vec{z}_t = h_t(\vec{x}_t, \vec{n}_t) \tag{2.5}$$

	parametric estimation	**non-parametric estimation**
unimodal	Kalman Filter (KF [4, 91]) Extended Kalman Filter (EKF [4]) Unscented Kalman Filter (UKF [90])	
multi-modal	Mixture of Gaussians (MOG [19])	Particle Filtering (PF [12, 39, 82]) Extended Kalman Particle Filter (EKPF [149]) Unscented Particle Filter (UPF [149]) Mixture of Particle Filters (MOPF [150])

Table 2.1: Probabilistic tracking techniques

are a function $h_t : \mathcal{S}^{d_x} \times \mathcal{S}^{d_n} \rightarrow \mathcal{S}^{d_m}$ of the state $\vec{x}_t$ and measurement noise $\vec{n}_t \in \mathcal{S}^{d_n}$ of known distribution R_t.

Recursive Bayesian tracking involves a prediction step based on the system model (Equation 2.4) and a correction step based on the measurement model (Equation 2.5).

In the prediction step the Chapman-Kolmogorov equation calculates the prior $p(\vec{x}_t|Z_{t-1})$ at time t

$$p(\vec{x}_t|Z_{t-1}) = \int p(\vec{x}_t|\vec{x}_{t-1})p(\vec{x}_{t-1}|Z_{t-1})d\vec{x}_{t-1} \tag{2.6}$$

using the process $p(\vec{x}_t|\vec{x}_{t-1})$ and previous a posteriori density $p(\vec{x}_{t-1}|Z_{t-1})$. Equation 2.6 uses the fact, that the system model (Equation 2.4) forms a Markov process of order one, and the assumption that the state transitions are independent given the observations $p(\vec{x}_t|\vec{x}_{t-1}, Z_{t-1}) = p(\vec{x}_t|\vec{x}_{t-1})$.

In the correction step, the new measurement $\vec{z}_t$ is used to update the prior via Bayes' rule to obtain the desired a posteriori density of the current state

$$p(\vec{x}_t|Z_t) = \frac{p(\vec{z}_t|\vec{x}_t)p(\vec{x}_t|Z_{t-1})}{p(\vec{z}_t|Z_{t-1})} \tag{2.7}$$

The normalizing constant

$$p(\vec{z}_t|Z_{t-1}) = \int p(\vec{z}_t|\vec{x}_t)p(\vec{x}_t|Z_{t-1})d\vec{x}_t \tag{2.8}$$

depends on the observation density $p(\vec{z}_t|\vec{x}_t)$, which is defined by the measurement model (2.5) and the a priory density $p(\vec{x}_t|Z_{t-1})$.

t	discrete time
$\vec{x}_t$	state vector
X_t	set of all measurement vectors from 0 up to time t
$\vec{v}_t$	process noise of distribution Q_t
$f_t(\vec{x}_t, \vec{v}_t)$	system model
$\vec{z}_t$	measurement vector
Z_t	set of all measurement vectors from 1 up to time t
$\vec{n}_t$	measurement noise of distribution R_t
$h_t(\vec{x}_t, \vec{n}_t)$	measurement model
$p(\vec{x}_t \vert Z_{t-1})$	a priory density
$p(\vec{x}_t \vert Z_t)$	a posteriori density
$p(\vec{x}_t \vert \vec{x}_{t-1})$	the process density
$p(\vec{z}_t \vert \vec{x}_t)$	the observation density
$s_t^{(n)}$	nth sample
$\pi_t^{(n)}$	nth sample weight
$\{s_t^{(n)}, \pi_t^{(n)}\}$	sample set

Table 2.2: Notations for Bayesian tracking systems

Equation 2.6 and 2.7 form the basis for the optimal Bayesian solution to propagate posterior probability density function in time. In general it cannot be evaluated analytically. Therefore several approximate algorithms have been developed.

2.2.2 Particle Filtering

In particular particle filters have become widespread in the computer vision community, because of their great ease and flexibility in approximating complex pdfs, and dealing with a wide range of dynamical and measurement models. Their multi-modal nature makes them particularly suited for object tracking in cluttered environments, where unimodal techniques might get stuck and lose track. Several particle filtering algorithms have been proposed in literature. A detailed overview can be found in [3].

Particle filters approximate the probability density function by a set of weighted particles or samples $\{s_t^{(n)}, \pi_t^{(n)}\}, \quad n = \{1, .., N\}$, where $s_t^{(n)}$ is the nth sample at time t with weight $\pi_t^{(n)}$. See Figure 2.3. In principle, particles could be placed at regular intervals in state space. For high dimensional state spaces however, this is not practical for computational reasons, since the required number of samples grows exponentially with the number dimensions. Therefore techniques have been developed, which concentrate the samples at the modes of the posterior during filtering.

In this Thesis particle filtering is performed by variants of the *condensation* [82] algorithm (Figure 2.4), which is briefly reviewed in the following. Since it approximates

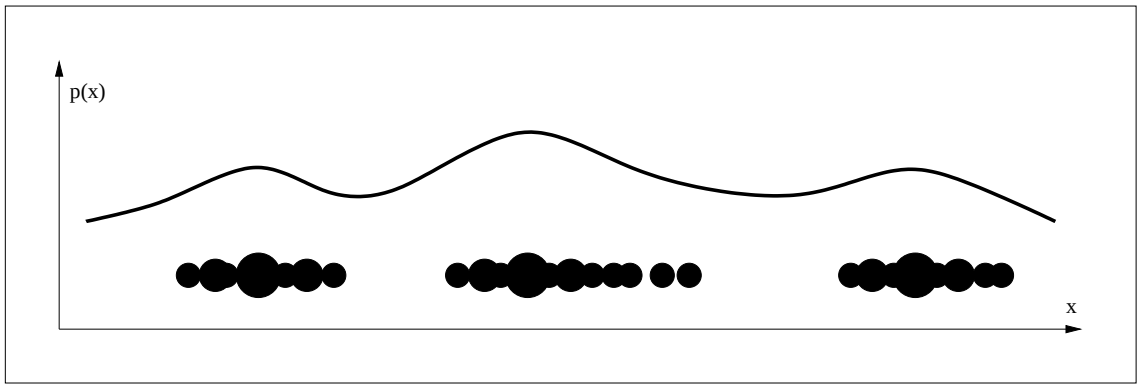

Figure 2.3: Discrete approximation of a probability density function by a set of particles or samples.

the (conceptual) optimal Bayesian solution for the filtering problem, it can also be subdivided into a prediction and correction step.

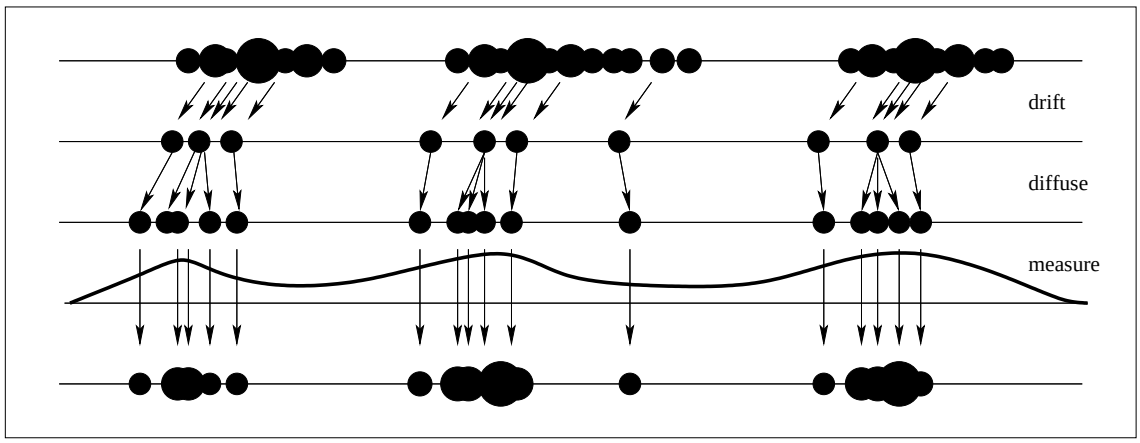

Figure 2.4: Propagating sample sets: at each iteration samples are drawn proportional to their weights, are drifted according to the deterministic component of the dynamical model and diffused according to the stochastic component. Finally they are re-weighted by the actual measurements. Figure adapted from Isard and Blake [82].

In the prediction step (Equation 2.6), the system dynamics is applied to each element of the sample set to approximate the a-priori pdf

$$s_t^{(n)} = f_{t-1}(s_{t-1}^{(n)}, v_{t-1}^{(n)}) \tag{2.9}$$

where $v_{t-1}^{(n)}$ is drawn from the known process noise Q_{t-1}.

From the prior sample set $\{s_{t-1}^{(n)}, \pi_{t-1}^{(n)},\ n \in \{1,..,N\}\}$ at time $t-1$ the nth sample $s_t^{(n)}$ with weight $\pi_t^{(n)}$ at time t is derived to yield the posterior sample set $\{s_t^{(n)}, \pi_t^{(n)},\ n \in \{1,..,N\}\}$.	
Select	a sample j of the prior population with probability $\pi_{t-1}^{(j)}$ and insert it into the new population $s_t'^{(n)} = s_{t-1}^{(j)}$
Predict	by sampling from $p(\vec{x_t}\|\vec{x_{t-1}} = s_t'^{(n)})$ to find $s_t^{(n)}$
Weight	according to the observation density $\pi_t^{(n)} = p(\vec{z_t}\|\vec{x_t} = s_t^{(n)})$
Finally, the weights are normalized such that $\sum_n \pi_t^{(n)} = 1$.	

Table 2.3: A particle filtering algorithm consists of three steps: sample selection, prediction, and weighting.

The correction step (Equation 2.7) involves the multiplication of the two probability density functions $p(\vec{x}_t|Z_t) \propto p(\vec{z}_t|\vec{x}_t)p(\vec{x}_t|Z_{t-1})$ by a technique called "factored sampling" [66]. It assumes, that the set of samples is randomly chosen from $p(\vec{x}_t|Z_{t-1})$. The new sample set is obtained by choosing sample $s^{(k)}$ with probability

$$\pi^{(k)} = \frac{p(\vec{z}_t|\vec{x}_t = s_t^{(k)})}{\sum_{i=1}^{N} p(\vec{z}_t|\vec{x}_t = s_t^{(i)})} \tag{2.10}$$

If the number of samples $N \to \infty$, their distribution should tend to $p(\vec{x}_t|Z_t)$. A simple particle filter algorithm is summarized in Table 2.3.

A theoretical treatment of particle filters can be found in [39]. A survey on convergence properties "for practitioners" on particle filtering methods is given in [27]. The authors conclude, that under weak assumptions (almost) sure convergence of the empirical distributions towards the optimal ones can be ensured.

Several extensions have been proposed to the early particle filter techniques, e.g. dealing with discrete/continuous state spaces. In [84], Isard and Blake track bouncing balls (and drawing hands) with several "continuous" motion classes, while the class membership is estimated simultaneously by a discrete parameter. Heap and Hogg follow a similar approach in [74]. In their work, a discrete parameter determines one of several shape classes, which consist of several continuous valued parameters.

Due to their ability to represent multimodal probability densities, single particle filters should be able to track multiple targets. In practice however, they fail for many applications. One problem is, that the samples tend to concentrate at the most probable mode. As a consequence, less probable objects get lost. Different strategies for multiple objects have therefore been developed [86, 105, 115, 133, 150].

A simple strategy is to constantly add samples into lower probable areas of the state space. Unfortunately this is only practical for low dimensional estimation problems. Another approach, for example chosen by Isard and MacCormick [86], is to include

the number of objects into the state space and estimate it simultaneously with their configuration. The dimension of the state space is then varied (or masked) over time, depending on how many objects are visible. Vermaak, Doucet, and Pérez [150] maintain multi-modality by tracking non-parametric mixture of particle filters.

Multiple sources of information [83, 141] have been considered with techniques such as importance sampling [83] or democratic integration [141, 146], and have been used to combine visual cues such as edge and texture in a particle filter framework.

Isard and Blake [83] fuse blob and contour tracking with an algorithm they call *icondensation*. It integrates the statistical technique of importance sampling and a reinitialization strategy into the standard *condensation* algorithm. Importance sampling techniques have also been considered in [3, 149]. These techniques can be applied in cases, for which it is difficult to sample directly from the posterior density function, but it is possible to sample from a known "proposal" distribution $q(x_t|X_{t-1}, Z_t)$, which comes for example from a different sensor. For computational reasons it is often convenient to choose the transition prior $p(\vec{x}_t|\vec{x}_{t-1})$ as proposal distribution [39].

Democratic integration [141, 146] is another way to fuse several sources of information, which can be applied, when consensus between the individual cues is predominant and environmental changes only effect a minority of the cues. During tracking, individual weights for each cue and for their integration are adapted un-supervised.

Philomin, Duraiswami, and Davis studied the influence of different random number generators on particle filters [118]. They conclude, that the sampling error can be reduced by applying quasi-random instead of pseudo random number sequences.

Particle filters have been applied in combination with low-level [111, 115], high-level [74], exemplar-based [144], or mixed-level [84] object representations.

Recently two new filters have been introduced: the "extended Kalman particle filter" and the "unscented particle filter" [149]. Their proposal distributions are based on the EKF and UKF (cf. next section) respectively. A comparative study of the different filters on synthetic and a real world data can be found in [149].

2.2.3 Kalman Filtering

The Kalman filter [91, 104] calculates the optimal solution to Bayesian filtering for the special case, that the posterior, system, and measurement densities form Gaussians and can therefore be parameterized by means and covariances. Furthermore the system and measurement models are assumed to be linear functions of their arguments. Equation 2.4 and 2.5 can therefore be written in matrix notation

$$\vec{x}_t = F_t\vec{x}_{t-1} + \vec{v}_{t-1} \tag{2.11}$$

$$\vec{z}_t = H_t\vec{x}_t + \vec{n}_t \tag{2.12}$$

where F_t and H_t define the linear system and measurement models.

The Kalman filter time update Equations

$$\hat{x}_{t+1}^{-} = F_t \hat{x}_t \tag{2.13}$$

$$P_{t+1}^{-} = F_t P_t F_t^T + Q_t \tag{2.14}$$

and measurement update Equations

$$K_t = P_t^{-} H_t^T (H_t P_t^{-} H_t^T + R_t)^{-1} \tag{2.15}$$

$$\hat{x}_t = \hat{x}_t^{-} + K_t(z_t - H_t \hat{x}_t^{-}) \tag{2.16}$$

$$P_t = (I - K_t H_t) P_t^{-} \tag{2.17}$$

can then be derived from Equation 2.6 and 2.7.

The time update Equations predict the state $\hat{x}_t$ and covariance P_t estimates from time t to $t+1$. The result are the a priory estimate $\hat{x}_{t+1}^{-}$ and covariance P_{t+1}^{-}.

The measurement update begins with the calculation of the Kalman gain K_t (Equation 2.15). It weights the prediction error $z_t - H_t \hat{x}_t^{-}$ in Equation 2.16 and therefore controls the influence of the new measurement z_t versus the predicted state estimate $\hat{x}_t^{-}$ to obtain the a posteriori estimate $\hat{x}_t$. Finally the a posteriori co-variance is calculated via 2.17.

A special kind of Kalman filters are the α-β and α-β-γ filters [4]. They are steady-state Kalman filters with fixed precomputed gains for constant noise and plant models (constant velocity and constant acceleration respectively). The gains can be derived from the ratio of motion and observation uncertainties [4]. For objects described by vector $\vec{x}$ moving at constant velocity $\vec{v}$, constant measurement noise, and constant time intervals T, a simple α-β filter is described by the following equations for prediction

$$\vec{x}_{k+1}^{-} = \vec{x}_k + T\vec{v}_k \tag{2.18}$$

$$\vec{v}_{k+1}^{-} = \vec{v}_k \tag{2.19}$$

and update

$$\vec{x}_k = \vec{x}_k^{-} + \alpha(\vec{z}_k - \vec{x}_k^{-}) \tag{2.20}$$

$$\vec{v}_k = \vec{v}_k^{-} + \frac{\beta}{T}(\vec{z}_k - \vec{x}_k^{-}) \tag{2.21}$$

To overcome the linearity assumptions of the system and measurement models, the "extended Kalman filter" (EKF) has been designed. It linearizes the non-linear Equations 2.4 and 2.5 by their Taylor series expansions. Although higher order terms of the expansion have been considered for EKFs, usually only the first term is used, for the local linear approximation.

A different approach for non-linear system and measurement models has recently been proposed in [90]. Instead of approximating non-linear functions, the authors approximate Gaussian distributions to propagate them elegantly through the non-linearities. The underlying principle is the "unscented transform". It selects a set of (sigma) points

deterministically from a known Gaussian distribution, maps them according a non-linear function, and re-estimates the parameters of the Gaussian. Based on this principle the "unscented" Kalman filter (UKF) and "unscented" particle filter (UPF) have been developed.

3 Dynamic Point Distribution Models (DPDMs)

3.1 Learning the Spatio-Temporal Shape Representation

For many interesting object detection tasks there are no explicit prior models available to support a matching process. This is typically the case for the detection of complex non-rigid objects under unrestricted viewpoints and/or under changing illumination conditions. This section deals with methods for acquiring (i.e. "learning") spatio-temporal shape models from examples. See Figure 3.1.

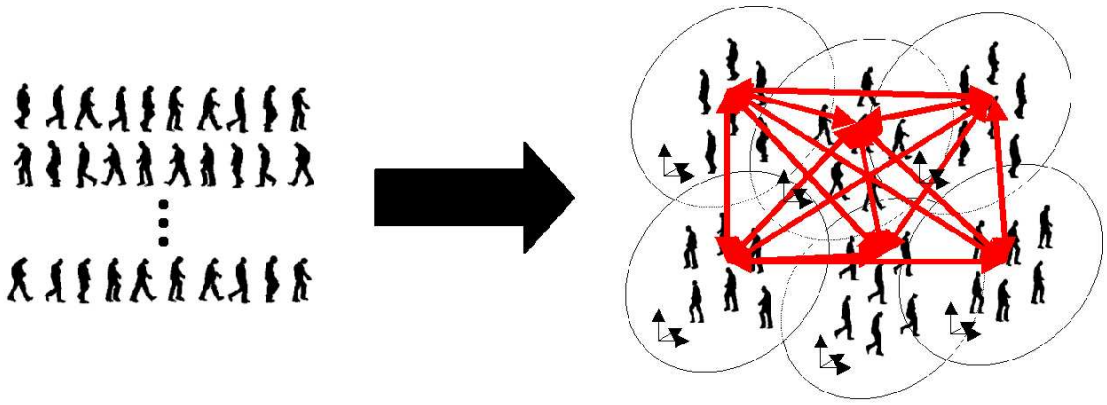

Figure 3.1: Learning Dynamic Point Distribution Models.

In Section 3.1.1 a technique for automatic shape registration is presented, which brings the points of two shapes into correspondence and aligns them with respect to similarity transforms. The registration procedure is integrated into a clustering algorithm, which partitions the training set into clusters of similar shapes. See Section 3.1.2. During clustering feature spaces are derived for each partition, which allows the derivation of compact shape parameterizations, e.g. "Point Distribution Models". To capture the temporal cluster transitions for tracking, their probabilities are stored in a Markov state transition matrix. (Section 3.1.3).

Since proposed shape representation consists of a set of "Point Distribution Models" with additional temporal information, it is called "Dynamic Point Distribution Model" or "DPDM" throughout this work.

3.1.1 Registration

Automatic shape registration methods bring the points of two shapes into correspondence and aligns them with respect to similarity transforms. The method described next, follows the typical succession of steps: shape segmentation, feature selection, point correspondence and finally, alignment.

Contour Segmentation

To divide shapes into segments the maxima of the curvature function are determined [99]. For twice differentiable contours parameterized by their boundary length (Equation 2.1) the curvature function is defined as

$$c(l) = \frac{\dot{u}\ddot{v} - \ddot{u}\dot{v}}{(\dot{u}^2 + \dot{v}^2)^{\frac{3}{2}}}. \tag{3.1}$$

where $\dot{u}$, $\dot{v}$, $\ddot{u}$, and $\ddot{v}$ are the first and second derivatives with respect to the arc length l respectively.

The curvature function is the first derivative of the "slope" function

$$s(l) = \frac{\dot{v}(l)}{\dot{u}(l)} \tag{3.2}$$

In practice Sobel operators are frequently applied to calculate the edge direction at each contour point. The gradient direction given by

$$\phi(l) = \tan^{-1} \frac{S_v(l)}{S_u(l)}, \tag{3.3}$$

where $S_v(l)$ and $S_u(l)$ are Sobel operators in the direction of u and v respectively.

A normalization step is performed to account for the "wraparound" of the arctangent function, which returns values in the range of $(-\pi, ..., \pi)$. Therefore the edge direction function is traced for any discontinuity greater than π or less than $-\pi$ and an offset of $\pm 2\pi$ is added accordingly.

The curvature function is calculated from the edge direction function by a convolution with the first derivative of a Gaussian $G'(s)$ with

$$G(s) = \exp(-\frac{s^2}{2\sigma^2}). \tag{3.4}$$

The Gaussian variance σ controls the smoothness of the resulting curve.

To determine the break points for segmentation, the curve is traced to find those extrema, whose absolute curvature value is above a user specified threshold. Figure 3.2 shows the curvature function for a plane shape (left) and the resulting break points (right).

The influence of the Gaussian σ-value is shown in Figure 3.3 for a plane and pedestrian shape. The higher the value the more local curvature extrema are "smoothed out" and therefore the number of break points is reduced.

Features for Shape Segments

Fourier descriptions are calculated to characterize boundary segments. The $N/2$ points $u(l)$ and $v(l)$ of each segment are followed forth and back and combined to a complex

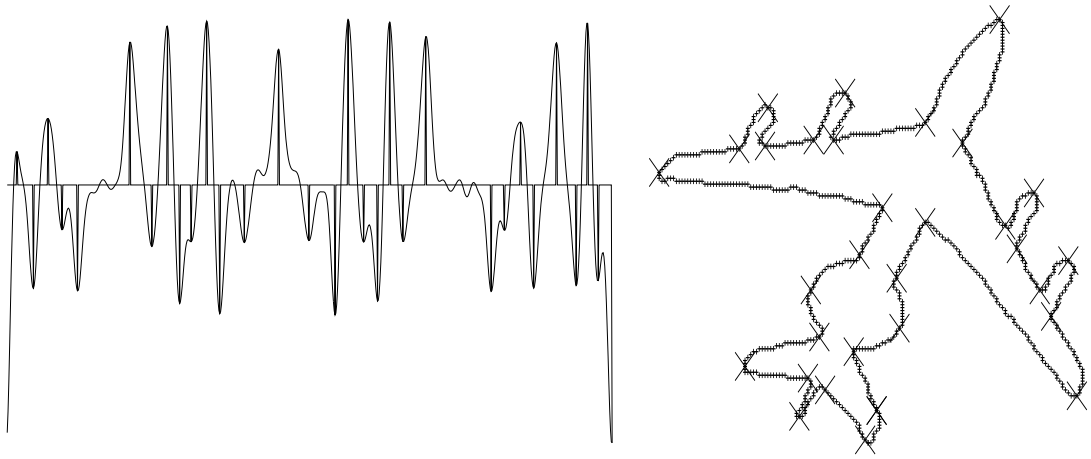

Figure 3.2: The extrema of the curvature function (left) define the break points for segmentation (right)

periodic curve $z(l) = u(l)+\mathbf{i}v(l)$, which can be approximated by a discrete Fourier series. The $k \in \{0, 1, ..., N-1\}$ coefficients

$$\bar{z}(k) = \frac{1}{N} \sum_{l=0}^{N-1} z(l) e^{\frac{-2\pi \mathbf{i} k l}{N}} \tag{3.5}$$

are called Fourier descriptors. From the descriptors, the original curve can be recovered without loss

$$z(l) = \sum_{k=0}^{N-1} \bar{z}(k) e^{\frac{2\pi \mathbf{i} k l}{N}} \tag{3.6}$$

The Fourier descriptors can be interpreted geometrically. The first one corresponds to the center of gravity, while the second one describes a circle [88]. Further descriptors add more and more detail to the curve. Depending on the required accuracy, their number can be reduced for compactness.

The Fourier coefficients are efficiently computed in $O(N \log_2 N)$ by the fast Fourier transform [119]. Typically only a few low order coefficients are considered for further computations to reduce the noise introduced by the pixel quantization and to speed up computation. Figure 3.4 illustrates the approximation for a varying number of Fourier coefficients.

To compare the segments of two contours, translation, scale, and rotation invariant Fourier descriptions are necessary. A pure *translation* only effects the first (complex) coefficient, which represents the center of gravity

$$\bar{z}(0) = \frac{1}{N} \sum_{l=0}^{N-1} z(l). \tag{3.7}$$

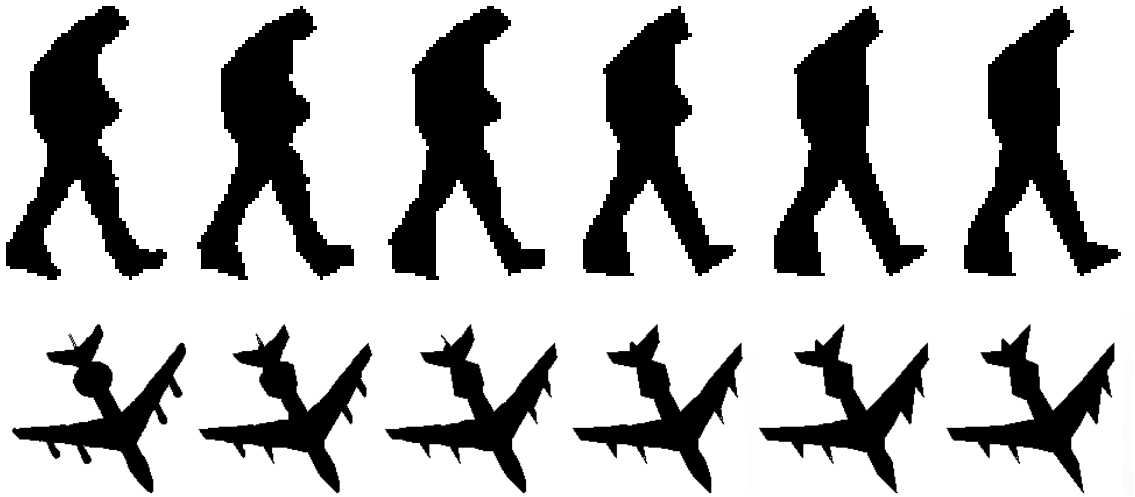

Figure 3.3: Polygonal approximation: the original shapes on the left are represented at multiple scales.

It is omitted to achieve translation invariance. If a contour is *scaled* with constant k, all coefficients are scaled with k as well. A normalization of all coefficients with the absolute value of $\bar{z}(1)$ yields scale invariance. According to the shift theorem of the Fourier transform a *rotation* of the curve about an angle ϕ causes a multiplication of each coefficient $\bar{z}(l)$ with $e^{il\phi}$. If all coefficients are related to the phase ϕ_1 of the second by reducing coefficient u by $u\phi_1$, all but the second are rotation invariant.

To measure the similarity between segments the Euclidian distance between the invariant Fourier coefficients is considered in this work. Given two segments s_1 and s_2 the distance is given by

$$d(s_1, s_2) = \sum_{k=0}^{M} \|\bar{s_1}(k) - \bar{s_2}(k)\|_2 \tag{3.8}$$

where $\|\|_2$ denotes the L2-norm over the invariant Fourier descriptors $\bar{s_1}(k)$ and $\bar{s_2}(k)$. In the experiments M is set to 1/16 of the number of segment points.

Establishing Correspondence

Given two contours segmented into N parts, there are $N!$ possibilities in theory to bring them into correspondence without additional prior knowledge. In case of closed contours, the ordering of the segments is known, so that the problem can optimally be solved in polynomial time by sequential pattern matching techniques, e.g. dynamic programming [65].

Dynamic programming searches the functions $i(k)$ and $j(k) \quad k \in \{1..N\}$, which correspond the features $w_{i(k)}$ of the segments of the first contour with features of the segments

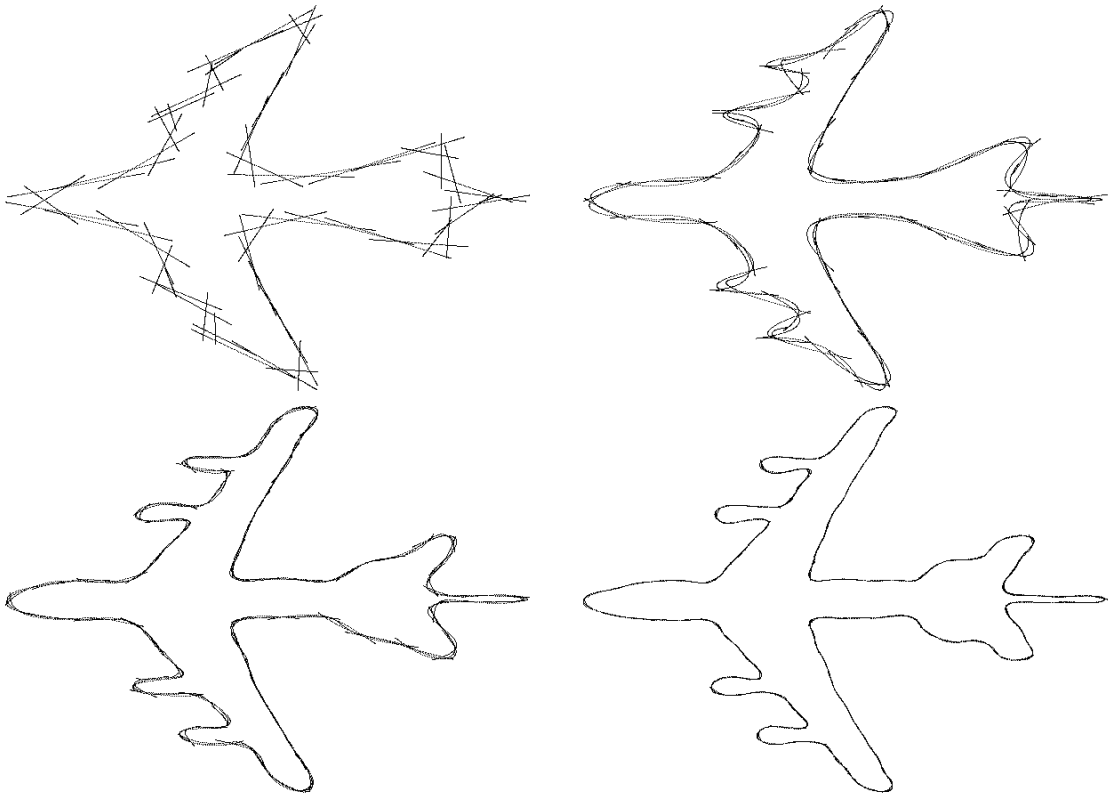

Figure 3.4: Piecewise Fourier approximation with a varying number of coefficients.

$w'_{j(k)}$ of the second one by minimizing the sum of differences

$$D_{min} = \min_{i(k),j(k)} \sum_{k=1}^{N} d(w_{i(k)}, w'_{j(k)}). \tag{3.9}$$

These functions map the segments to a common axis k. Segment $s_{i(k)}$ of the first corresponds to segment $s'_{j(k)}$ of the second contour. Caused by the ordering of the segments, both functions $i(k)$ and $j(k)$ are increasing strictly

$$i(k+1) \geq i(k) \quad j(k+1) \geq j(k) \tag{3.10}$$

Dynamic programming for given start segments proceeds as follows and utilizes three tables, exemplary shown in Table 3.1.

At first a complete table of distances $d(w_i, w'_j)$ between the segment features is calculated using the distance measure given by Equation 3.8. Based on the distances a table is constructed, which contains the minimum sum of differences to correspond segment i with j.

It is filled from left to right and bottom to top in the example. Typically the "partial cumulative distance"

$$s_{i,j} = d_{i,j} + \min \begin{cases} s_{i,j-1} \\ s_{i-1,j-1} \\ s_{i-1,j} \end{cases} \tag{3.11}$$

is the sum of the distance $d_{i,j}$ and the minimum of three preceeding partial cumulative distances $s_{i,j-1}, s_{i-1,j-1}$ and $s_{i-1,j}$. Some authors consider different partial solutions $s_{k,l}$ which might be useful to locally constrain the search or to cope with outliers in the table of distances [65].

Simultaneously a "path table" is constructed, which stores the direction of the previous optimal solution. In the example a one denotes from bottom left, a two from left, and a three from the bottom.

Once the tables are filled, the minimal cumulative distance to solve the correspondence problem is located in the upper right corner of the table of cumulative distances. The solution forms a path, which can easily be reconstructed by back tracing the path table. In the example the optimal path coincides with the diagonal, so that $i(k) = j(k) = k$.

4	0.072	0.039	0.070	0.042
3	0.041	0.070	0.039	0.073
2	0.073	0.039	0.070	0.041
1	0.041	0.070	0.039	0.073
i/j	1	2	3	4

(a)

4	0.228	0.189	0.189	0.161
3	0.155	0.150	0.119	0.192
2	0.114	0.080	0.151	0.191
1	0.041	0.111	0.150	0.223
i/j	1	2	3	4

(b)

4	3	3	3	1
3	3	3	1	2
2	3	1	2	1
1	1	2	2	2
i/j	1	2	3	4

(c)

Table 3.1: Dynamic programming: (a) table of distances (b) minimum cumulative sum of distances (c) path table

Since the optimal start segments $i(1)$ and $j(1)$ are unknown in advance, Equation 3.9 has to be computed N times. Thus the complexity is $O(n^3)$. The optimal start segments are chosen to minimize the cumulative sum of differences.

Figure 3.5 shows results for automatically determined correspondences between contours of pedestrians and planes.

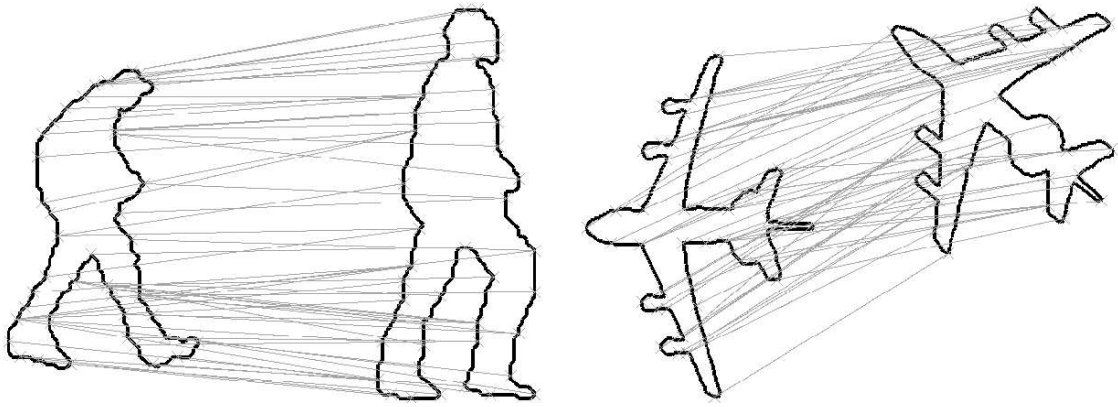

Figure 3.5: Automatically determined point correspondences between pedestrian and plane shapes.

Alignment

To match two contours, the points have to be aligned at first. If point correspondences are available, the optimal alignment parameters (e.g. rotation θ, scale s, translation $\vec{t}$) may be determined by a weighted least squared fit [24], where the weights for each correspondence are stored in a diagonal matrix W. (In the experiments, W equals an identity matrix.)

If $\vec{x}_i = (x_{i0}, y_{i0}, x_{i1}, y_{i1}, ..., x_{in}, y_{in})^T$ contains the ith shape vector and

$$T_{s,\theta}\begin{pmatrix} x_{ik} \\ y_{ik} \end{pmatrix} = \begin{pmatrix} (s\cos\theta)x_{ik} - (s\sin\theta)y_{ik} \\ (s\sin\theta)x_{ik} + (s\cos\theta)y_{ik} \end{pmatrix} = \begin{pmatrix} a_x x_{ik} - a_y y_{ik} \\ a_y x_{ik} + a_x y_{ik} \end{pmatrix} \tag{3.12}$$

denotes the transformation for each point, we seek the minimum of weighted squared differences

$$\min E = (\vec{x}_1 - T_{s,\theta}\vec{x}_2 - \vec{t})^T W (\vec{x}_1 - T_{s,\theta}\vec{x}_2 - \vec{t}) \tag{3.13}$$

with respect to rotation θ, scale s, and translation $\vec{t}$.

Differentiating with respect to each variable leads to a set of four linear equations

$$\begin{pmatrix} X_2 & -Y_2 & W & 0 \\ Y_2 & X_2 & 0 & W \\ Z & 0 & X_2 & Y_2 \\ 0 & Z & -Y_2 & X_2 \end{pmatrix} \begin{pmatrix} a_x \\ a_y \\ t_x \\ t_y \end{pmatrix} = \begin{pmatrix} X_1 \\ Y_1 \\ C_1 \\ C_2 \end{pmatrix} \tag{3.14}$$

where

$$X_i = \sum_{k=0}^{n-1} w_k x_{ik}$$

$$Y_i = \sum_{k=0}^{n-1} w_k y_{ik}$$

$$Z = \sum_{k=0}^{n-1} w_k (x_{2k}^2 + y_2 k^2)$$

$$W = \sum_{k=0}^{n-1} w_k$$

$$C_1 = \sum_{k=0}^{n-1} w_k (x_{1k} x_{2k} + y_{1k} y_{2k})$$

$$C_2 = \sum_{k=0}^{n-1} w_k (y_{1k} x_{2k} - x_{1k} y_{2k})$$

It can be solved using standard matrix methods, for example based on LU decomposition [119].

Figure 3.6 shows alignment results for pedestrian and plane shapes. The right shapes are aligned to the left. The results are shown in grey.

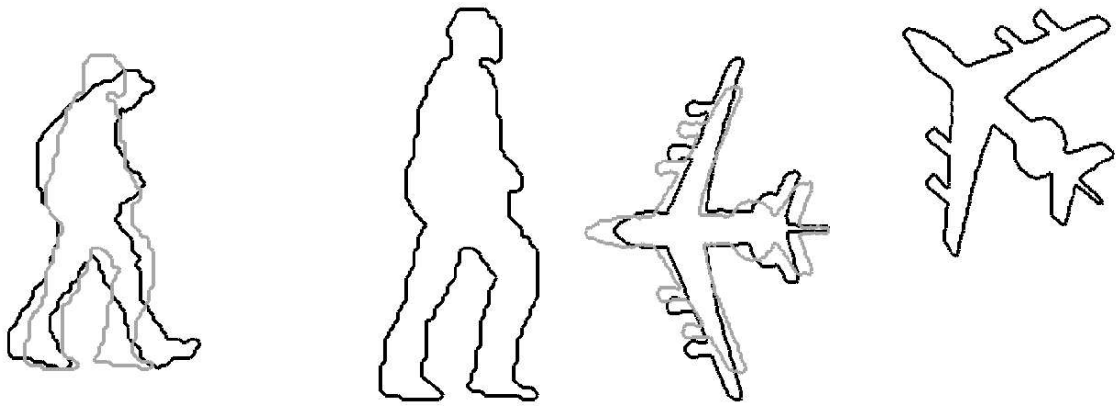

Figure 3.6: Alignment. The right shapes are aligned to the left. The resulting shapes are drawn in grey.

3.1.2 Contour Classes and Shape Variability

Registration establishes point correspondence between a pair of shapes. The straightforward way to account for N shapes is to embed all N shapes in a single feature vector space, based on the u and v locations of their corresponding points. The assumption that point correspondence can be established across all training shapes of a particular object class by means of automatic shape registration is, however, in many cases problematic. For example, none of the shape registration methods analyzed in [54] were able

to correctly register a pedestrian shape with the two feet apart with one with the feet together, without operator input or prior knowledge. A more general registration approach does not forcibly embed all N shapes into the same feature vector space. Instead, it combines shape clustering and registration, embedding only the (similar) shapes within a cluster into the same vector space. The resulting vector spaces allow the computation of compact linear subspace representations, known as "Point Distribution Models".

Integrated Registration and Clustering

A lot of algorithms for clustering exist in literature, which partition a set of shapes according to a similarity measure. (It may be based on vector spaces or pair-wise similarities.) Hierarchical methods automatically derive a tree structure, and can be subdivided into agglomerative and divisive approaches. While agglomerative approaches start with a huge amount of clusters (typically one cluster per shape) and join similar ones into larger, divisive methods begin with only a few large clusters (typically one) and successively divide them into smaller. Partitioning methods on the other hand, divide data sets into clusters with respect to a user supplied criterion, such as the separability of clusters.

In this work, a simple "one pass" k-means algorithm, which is able to handle continuously arriving data, is applied to cluster the shapes $\vec{s}_i$. The input of the proposed method is a sequence of unregistered closed contours, while the output is a set of clusters in which point correspondences are available.

The method has a computational complexity of $O(N(N-1)/2)$ in the worst case, when each shape builds a single cluster and N denotes the number of shapes. In the best case, only one comparison is necessary for each shape and the complexity reduces to $O(N)$.

The algorithm represents each cluster C_j by a prototype p_{C_j}. For continuously arriving data the prototypes may be updated according to

$$p_{C_j}^t = \begin{cases} p_{C_j}^{t-1} + a_t(\vec{s}_i - p_{C_j}^{t-1}), & if \quad \vec{s}_i \in C_j \\ p_{C_j}^{t-1} \quad otherwise \end{cases} \tag{3.15}$$

An example prototype of a class with three elements is illustrated in Figure 3.7.

The clustering algorithm proceeds as follows. At first, a single cluster is initialized with the first shape as prototype. Then iteratively a shape is chosen and compared to all existing prototypes. If the minimal distance falls below a user supplied threshold, the shape is assigned to the corresponding cluster and the mean is updated. Otherwise a new cluster is created with the shape as prototype.

The similarity measure is derived from the registration procedure described in the Section 3.1.1 and is defined to be the mean distance between the corresponding points of the shapes. This ensures, that only correctly registered shapes fall into the same cluster. The algorithm is summarized in the following.

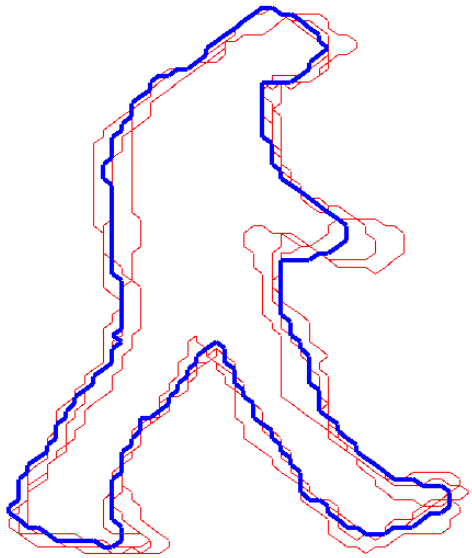

Figure 3.7: Mean shape computation. The dark shape is the mean of the three light shapes.

0. pick an initial shape $\vec{s}_1$ and add it to cluster C_1
as prototype:
$C_1 = \{\vec{s}_1\}$, $p_1 = \vec{s}_1$
while there are shapes left do
1. select one of remaining shapes: $\vec{s}_k$
2. compute mean alignment error $d(\vec{s}_k, p_i)$ from
element $\vec{s}_k$ to the existing prototypes p_i,
where i ranges over the number of clusters
created so far
3. Compute $d_{\min} = d(\vec{d}_k, p_j) = \min_i d(\vec{s}_k, p_i)$.
if $d(\vec{s}_k, p_j) > \theta$
then assign $\vec{s}_k$ to a new cluster C_{n+1}:
$C_{n+1} = \{\vec{s}_k\}, p_{n+1} = \vec{s}_k$
else assign $\vec{s}_k$ to existing cluster
$C_j = \{\vec{s}_{j1}, ...\vec{s}_{jn}\}$ and update its prototype:
$C_j = C_j \cup \{\vec{s}_k\}$
$p_j = \text{Mean}(\vec{s}_{j1}, ..., \vec{s}_{jn}, \vec{s}_k)$
end

Figure 3.8 shows some clustering results for a sequence of a pedestrian walking from the left to the right.

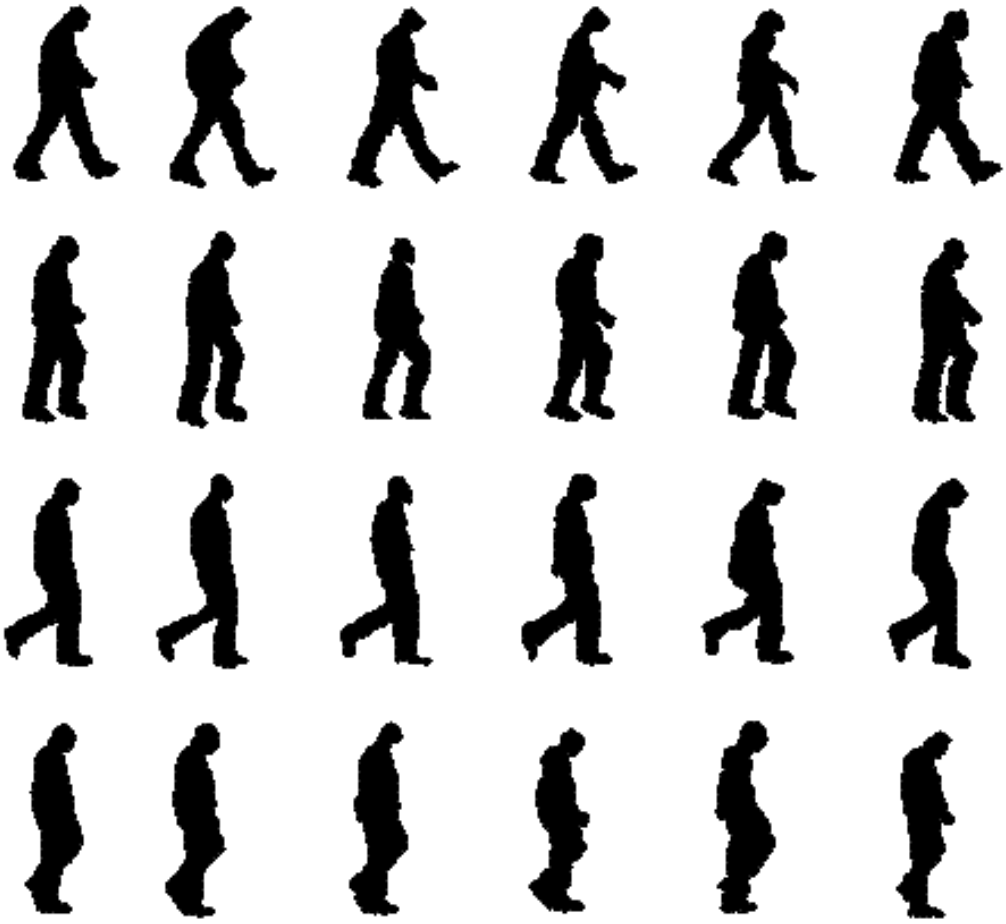

Figure 3.8: Clustering results of one sequence in which a pedestrian is walking from the left to the right.

Linear Subspace Models

A principal component analysis is applied in each cluster of registered shapes to obtain compact shape parameterizations known as "Point Distribution Models" (PDMs) [24].

From the N^c shape vectors of cluster c given by their u- and v-coordinates

$$\vec{s}_i^c = (u_{i,1}^c, v_{i,1}^c, u_{i,2}^c v_{i,2}^c, ..., u_{i,n^c}^c, v_{i,n^c}^c), \quad i \in \{1, 2, ..., N^c\} \tag{3.16}$$

the mean shape

$$\bar{\vec{s}}^c = \frac{1}{N^c} \sum_{i=1}^{N^c} \vec{s}_i^c \tag{3.17}$$

and covariance matrix

$$K^c = \frac{1}{N^c - 1} \sum_{i=1}^{N^c} (\vec{s}_i^{\,c} - \bar{\vec{s}}^{\,c})(\vec{s}_i^{\,c} - \bar{\vec{s}}^{\,c})^T \tag{3.18}$$

is derived.

Solving the eigensystem $K^c \vec{e}_j^{\,c} = \lambda_j^c \vec{e}_j^{\,c}$ one obtains the $2n^c$ orthonormal eigenvectors, corresponding to the "modes of variation".

The k^c most significant "variation vectors" $E^c = (\vec{e}_1^{\,c}, \vec{e}_2^{\,c}, ..., \vec{e}_{k^c}^{\,c})$, the ones with the highest eigenvalues λ_j^c, are chosen to explain a user specified proportion of total variance contained in the cluster.

Shapes are generated from the mean shape plus a weighted combination of the variation vectors

$$\vec{s}^{\,c} = \vec{\bar{s}}^{\,c} + E^c \vec{b}. \tag{3.19}$$

Figure 3.9 illustrates the changes in shape while varying the first mode of variation for two different clusters between ± 2 standard deviations σ.

Figure 3.9: Linear subspace models: varying the first mode of variation for two different clusters.

To ensure that the generated shapes remain similar to the training set, the weight vector $\vec{b}$ is constrained to lie in a hyperellipsoid about the subspace origin. Therefore $\vec{b}$ is scaled so that the weighted distance from the origin is less than a user-supplied threshold M_{max}

$$\sum_{l=1}^{k^c} \frac{b_l^{c2}}{\lambda_l^c} \leq M_{max}^2 \tag{3.20}$$

3.1.3 Dynamic Contours: Shape Transition Matrix

DPDMs model continuous shape changes by parameter variations within a PDM, while discontinuous changes correspond to PDM transitions.

To capture the temporal sequence of PDMs a Markov state transition matrix stores the transition probabilities $T_{i,j}$ from cluster i to j. They are automatically derived from the transition frequencies found in the training sequences. An extension for covering more complex temporal events (e.g. [45]) is conceivable and straightforward.

3.2 DPDM Snakes

In this Section Dynamic Point Distribution Models are applied to object segmentation. The method was originally examined in this Thesis to optimize the tracking framework described in chapter 4, but it can for example also be applied for interactive object labeling tasks.

The goal is to separate a modeled object from the background. This involves finding the optimal parameters

- scale s, rotation θ, and translation $\vec{t}$
- PDM membership c
- shape parameters $\vec{b}$

which bring the model into correspondence with the image structures.

An iterative scheme, which optimizes scale, rotation, translation, and the shape parameters has been proposed in [24] for a single Point Distribution Model. After an initial starting approximation is given by the user, three steps are repeated until no significant changes result. See Figure 3.10: at first the contour points are calculated from the actual parameters and a local displacement is derived for each, to better match it with the image features. From these local displacements the scale, rotation, translation, and the shape parameters adjustments are derived. Finally, limits on the shape parameters are applied, so that the resulting shapes remain similar to those in the training set.

In this work the method is generalized to multiple PDMs and a different way of determining the local displacements is considered. It is based on dynamic programming and uses a similarity measure, which considers the magnitude and direction of edges along the normals of the model boundary. Refer Section 3.2.2.

3.2.1 Determining the PDM-Membership

Two approaches to determine the optimal PDM membership are considered. The first matches the mean shapes of all PDMs at the initial image position and determines the best fitting one. The corresponding PDM is chosen for further parameter optimization.

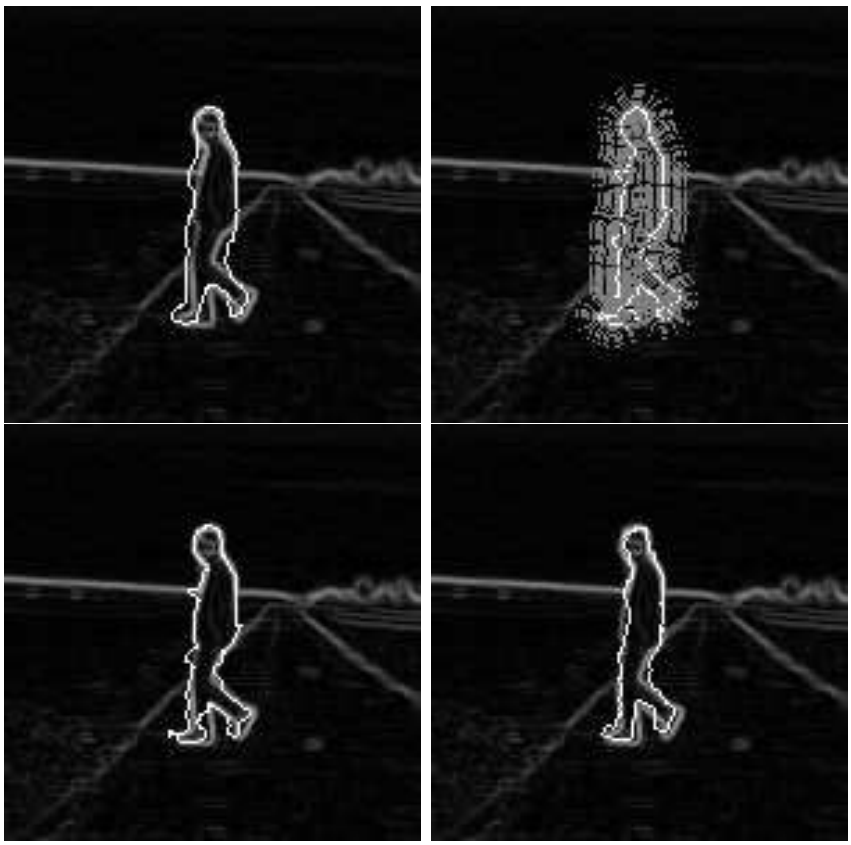

Figure 3.10: DPDM Snake: After initialization (upper left) edge features along the shape normals are registered with image features (upper right). From the resulting correspondences alignment parameters are derived and applied (lower left). Finally model constraints assure, that the shape remains similar to the training set (lower right).

The second starts a parameter optimization for each PDM. The membership is then chosen afterwards according to the match values of the resulting shapes.

The first method is obviously much faster since only a single iterative optimization procedure has to be processed. It can be applied, when the initialization is close to the correct shape. Otherwise the risk of selecting the wrong PDM is too high and the second approach is preferred.

(In the tracking framework, introduced in Chapter 4, the PDM membership is estimated by the particle filter.)

3.2.2 Registration of Shape and Image Features

Given an initial list of N boundary points $\vec{s} = (u(1), v(1), u(2), v(2), ..., u(N), v(N))$, the goal is to find a position for each point, which better matches it with the image features. To speed up computation, a common way is to constrain the search along the normals of the boundary points. While greedy approaches consider each point separately and move it to the strongest image feature, in this work, a global optimal solution is calculated by dynamic programming.

The algorithm maximizes the following energy over all contour points i

$$E_{\max \vec{s}} = \sum_{i=1}^{N} e(\vec{s}(i)) \tag{3.21}$$

where

$$e(\vec{s}(i)) = \begin{cases} |\nabla I(\vec{s}(i))| & if \quad \Delta\Phi < T \\ 0 & else \end{cases} \tag{3.22}$$

is the magnitude of the image gradient $|\nabla I(\vec{s}(i))|$, if the direction difference $\Delta\Phi$ between the contour normal and the image gradient is below a user defined threshold T. Otherwise $e(\vec{s}(i))$ equals 0. (It would also be possible to define $e(\vec{s}(i))$ in terms of a scalar product between the image gradient and shape normal.)

The optimal $\vec{s}_{opt}$, which maximizes E, is calculated by a standard dynamic programming formulation. As illustrated in Figure 3.11 a typical solution by dynamic programming results in a much smoother contour, compared to a simple greedy approach.

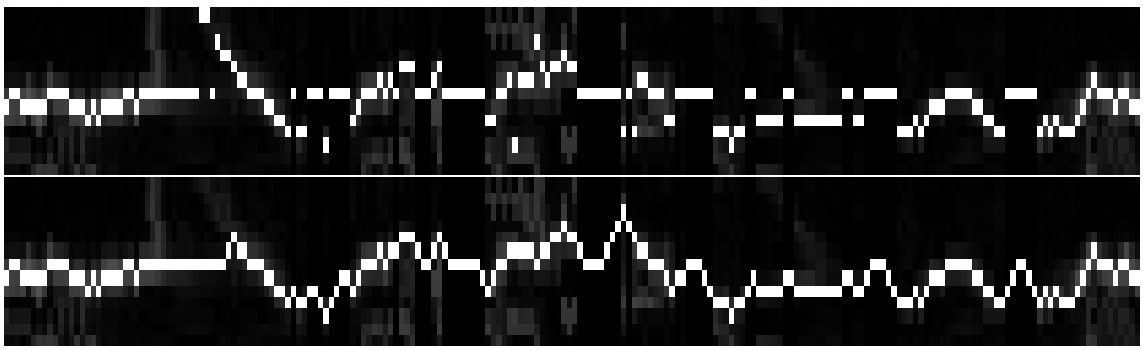

Figure 3.11: Registration of shape and image features: Each column represents the edge values under a shape normal. A greedy approach (upper image) assigns each boundary point to the strongest edge value. The solution is marked in white. Dynamic programming (lower image) calculates a global optimal solution.

3.2.3 Alignment and Shape Deformation

Once the point adjustments $d\vec{s} = \vec{s}_{opt} - \vec{s}$ have been determined, the next step is to find the pose and shape parameters, which bring the shape as close to the solution $\vec{s} + d\vec{s}$, while satisfying the shape constraints of the model. The first step is to calculate the translation $d\vec{t}$, scale $1 + ds$, and rotation $d\theta$ updates, which best map the points $\vec{s}$ to $\vec{s}_{opt}$ using the least squares fit described in Section 3.1.1.

The remaining residual adjustments have to be explained by shape deformations of the model. Since the shape

$$\vec{s} = T_{s,\theta}[\vec{s}_m] + \vec{t} \tag{3.23}$$

is defined in the image coordinate frame, it has to be transformed into the model coordinate frame at first. Of interest are the shape adjustments in the model coordinate frame $d\vec{s}_m$ in a way that

$$T_{s(1+ds),(\theta+d\theta)}[\vec{s}_m + d\vec{s}_m] + (\vec{t} + d\vec{t}) = (\vec{s} + d\vec{s}). \tag{3.24}$$

Thus

$$T_{s(1+ds),(\theta+d\theta)}[\vec{s}_m + d\vec{s}_m] = ((T_{s,\theta}[\vec{s}_m] + \vec{t}) + d\vec{s}) - (\vec{t} + d\vec{t}) \tag{3.25}$$

and since $T_{s,\theta}^{-1} = T_{s^{-1},-\theta}$ the adjustments in the local coordinate frame are

$$d\vec{s}_m = T_{(s(1+ds))^{-1},-(\theta+d\theta)}[T_{s,\theta}[\vec{s}_m] + d\vec{s} - d\vec{t}] - \vec{s}_m. \tag{3.26}$$

These local adjustments are projected into the subspace model to find model update parameters $d\vec{b}$, which are consistent with our model. Starting from Equation 3.19

$$\vec{s_m} = \bar{\vec{s}}_m + E\vec{b} \tag{3.27}$$

the goal is to find $d\vec{b}$ such that

$$\vec{s}_m + d\vec{s}_m \approx \bar{\vec{s}}_m + E(\vec{b} + d\vec{b}) \tag{3.28}$$

Subtracting Equation 3.27 from Equation 3.28 gives

$$d\vec{s}_m \approx Ed\vec{b} \tag{3.29}$$

and therefore

$$d\vec{b} \approx E^{-1}d\vec{s}_m. \tag{3.30}$$

At this point all parameter updates are available, which optimize the fit between the model and registered image structures.

$$\vec{t} \rightarrow \vec{t} + w_t d\vec{t} \tag{3.31}$$

$$\vec{s} \rightarrow \vec{s}(1 + w_s ds) \tag{3.32}$$

$$\theta \rightarrow \theta + w_\theta d\theta \tag{3.33}$$

$$\vec{b} \rightarrow \vec{b} + W_b d\vec{b} \tag{3.34}$$

They may be weighted by scalar factors w_t, w_s, w_θ and a diagonal matrix of weights W_b for the shape parameters $\vec{b}$.

An example of of the proposed method is given in Section 5.3.

3.3 Virtual Sample Generation

This section presents a technique for enlarging a set of training shapes with generated, "virtual" shapes, for improving representational capability. The method is general in the sense that it does not require prior feature correspondence among the shapes in the training set.

It is in particular interesting for shape-based approaches, which capture object appearance by a set of shape templates and which use correlation as basic tool for matching. The appeal of such systems lies in their robustness and generality. Correlation copes relatively well with missing data due to occlusion or incorrect segmentation. Their generality relates to the ease in dealing with a broad class of shapes, without requiring feature correspondences or parameterizations of the training shapes.

Among appearance-based systems using shape templates, those that correlate using distance transforms [5] have proven to be particularly successful. The smoothness of the resulting correlation measure with respect to shape perturbations and changes in transformation parameters enables the use of very efficient pruning and hierarchical techniques for matching [14, 78, 53].

An important prerequisite, however, for such appearance-based systems is that their training set adequately covers the object's shape distribution. All possible object instances should ideally have a dissimilarity value below the matching threshold with respect to existing templates. Yet the ability to collect training samples is for many applications limited by practical reasons. In this case, the generation of "virtual" samples can increase the performance of these applications.

3.3.1 Sampling DPDMs

The approach is summarized in Figure 3.12. From the original training set, represented as shape vectors, a DPDM is learned as described in Section 3.1. From each of the resulting Point Distribution Models shapes can be generated by sampling the weight vectors $\vec{b}$ according to a Gaussian distribution and projecting them to the original template representation using Equation 3.19.

Several parameters control the similarity between the training shapes and the generated "virtual" shapes. Refer Figure 3.13.

- **Number of PDMs:** The number of PDMs is determined by the clustering algorithm.
- **Number of eigenvectors:** The number of eigenvectors can be chosen to explain a user supplied proportion of total variance contained in the training data.
- **Gaussian Noise:** Gaussian noise is applied to determine the weight vectors $\vec{b}$.

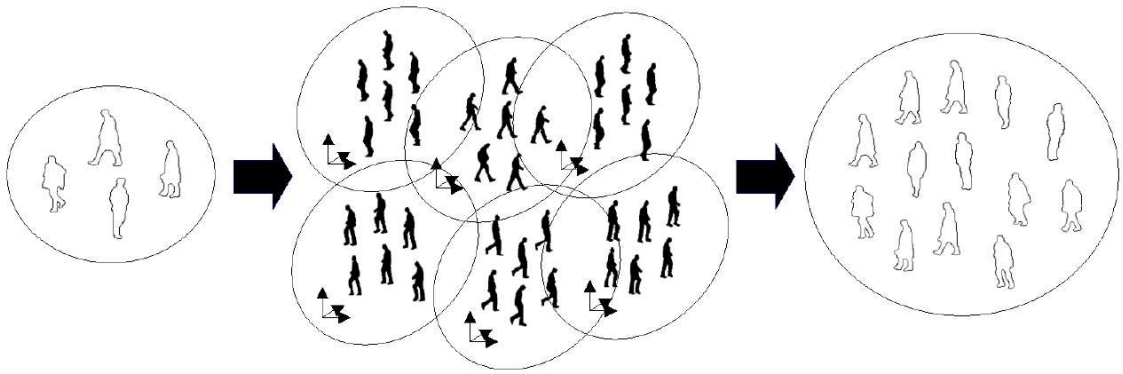

Figure 3.12: Extending training sets.

- **Mahalanobis threshold:** The weight vectors $\vec{b}$ are constrained to lie in a hyperellipsoid about the subspace origin by the Mahalanobis threshold. See Figure 3.13.

In the experiments in Section 5.4 it is shown, that extending a training set with generated shapes can result in an extended training set, which facilitates a superior matching performance, as measured by an ROC curve.

It is also possible to generate complete "virtual" shape sequences, by sampling the temporal PDM transitions according to the transition probabilities $T_{i,j}$, described in Section 3.1.3. Refer Figure 5.8.

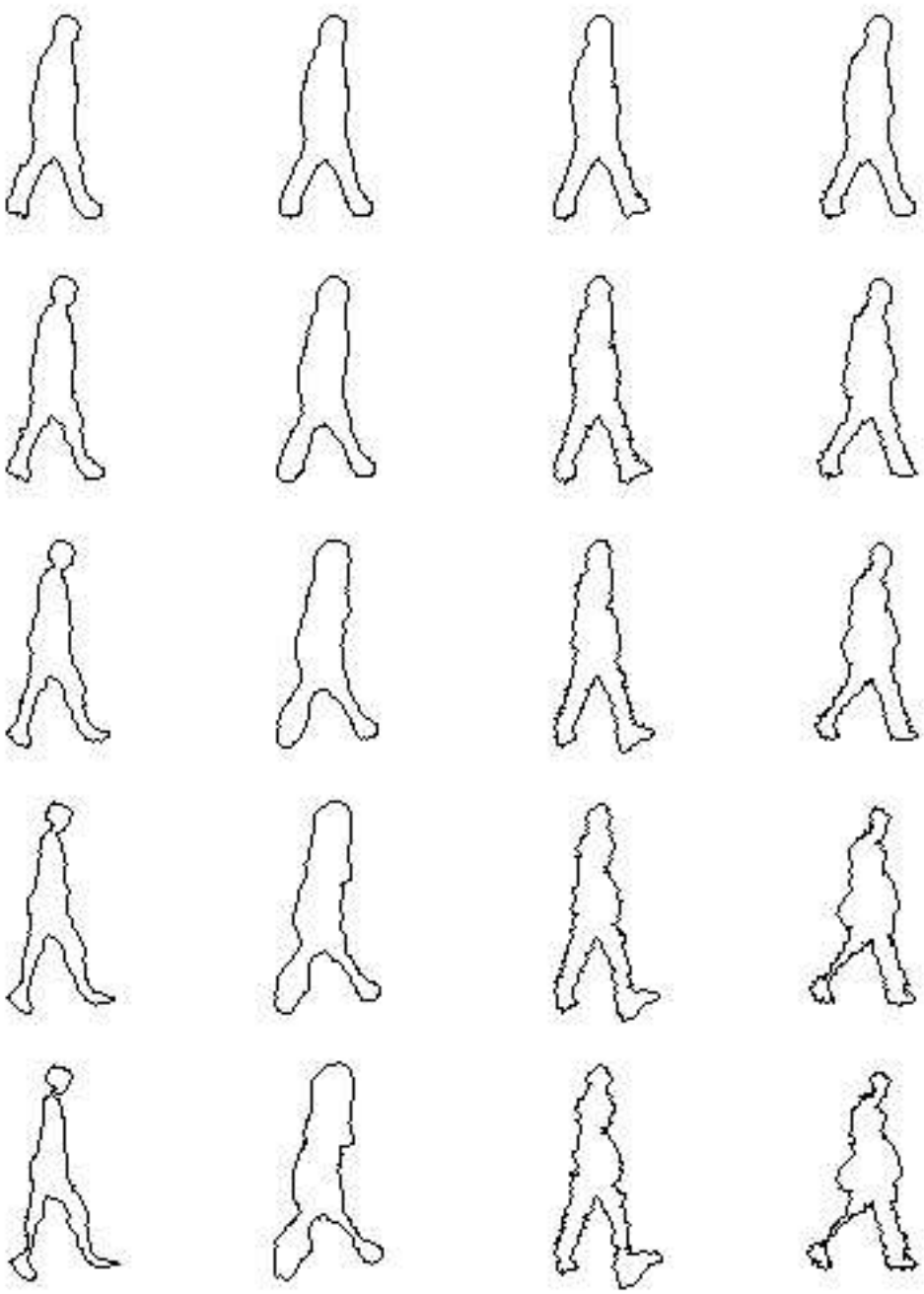

Figure 3.13: Virtual shapes. Several parameters control the similarity between the training shapes and the generated "virtual" shapes. In the example the Mahalanobis threshold is varied from the top row to the bottom leaving all other parameters unchanged.

4 Tracking DPDMs

Object tracking is a central theme in computer vision with applications ranging from surveillance to intelligent vehicles. The focus lies on tracking complex, deformable objects through cluttered environments, for those cases when simple segmentation techniques, such as background subtraction, are not applicable.

This section presents a Bayesian framework for integrated detection and tracking of non-rigid objects. Detections from an independent source of information are modeled as "mixture" of Gaussians and are integrated by two means: They control initialization and termination by a set of rules and serve as additional source of information for the active tracks.

To increase robustness three independent visual cues are combined. Object shape is used, since it is (quite) independent of the complex illumination conditions found in real world applications and efficient matching techniques exist to compare shapes with images [53]. Texture distributions are modeled as histograms [111, 115], which are particularly suitable for tracking since they are independent of object shape, invariant to rotation, scale, and translation, and easy to compute. Finally stereo measurements are integrated into the system.

Tracking proceeds directly in 3D-space, which allows a more natural incorporation of real-world knowledge (e.g. kinematical properties of objects) and simplifies reasoning about occlusion and data association.

The benefit of the integrated multi-cue tracking approach on pedestrian tracking from a moving vehicle is illustrated in the experiments.

4.1 Mixed Continuous/Discrete State Space

In this work particle filtering is applied to approximate optimal Bayesian tracking [12, 39] for a single target. The state vector

$$\vec{x} = (\Pi, \Sigma) \tag{4.1}$$

of a particle comprises the position and velocity

$$\Pi = (x, y, z, v_x, v_y, v_z) \tag{4.2}$$

in three dimensional space (a fixed object size is assumed), and the shape parameters

$$\Sigma = (c, \vec{b}) \tag{4.3}$$

introduced in the previous Section 3.1.

For tracking, the dynamics $p(\vec{x_{k+1}}|\vec{x_k} = s_k^i)$ and the conditional observation density $p(\vec{z_k}|\vec{x_k} = s_k^i)$ have to be specified, whereas s_k^i is the i^{th} sample at time k.

4.2 Decomposed Dynamics

Object dynamics is assumed independent for the two components Π and Σ of our state vector and is defined separately as follows.

During each sampling period T_k the position and velocity vector Π is assumed to evolve according to the following dynamic equation

$$\Pi_{k+1} = \begin{pmatrix} 1 & 0 & 0 & T_k & 0 & 0 \\ 0 & 1 & 0 & 0 & T_k & 0 \\ 0 & 0 & 1 & 0 & 0 & T_k \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \Pi_k + \Gamma_k \nu_k, \tag{4.4}$$

whereas ν_k is the user defined process noise, which has to be chosen to account for velocity changes during each sampling interval T_k, and Γ_k is the time dependent noise gain [4, 12].

The shape component $\Sigma = (c, \vec{b})$ is composed of a discrete parameter c_k modeling the cluster membership and the continuous valued weight vector $\vec{b}$. To deal with this "mixed" state the dynamics is decomposed as follows

$$p(\Sigma_{k+1}|\Sigma_k) = p(\vec{b}_{k+1}|c_{k+1}, \Sigma_k) p(c_{k+1}|\Sigma_k). \tag{4.5}$$

Assuming that the transition probabilities $T_{i,j}$ of our discrete Markov model are independent of the previous weight vector $\vec{b}_k$, the second part of Equation 4.5 reduces to

$$p(c_{k+1} = j | c_k = i, \vec{b}_k) = T_{i,j}(\vec{b}_k) = T_{i,j}. \tag{4.6}$$

For the continuous parameters two cases have to be considered: In case of $i = j$, when no PDM transition occurs,

$$p(\vec{b}_{k+1}|c_{k+1} = j, c_k = i, \vec{b}_k) = p_{i,j}(\vec{b}_{k+1}|\vec{b}_k) \tag{4.7}$$

is assumed to be a Gaussian random walk with user supplied variances. For $i \neq j$ the cluster is switched from i to j and the parameters $\vec{b}$ are assumed to be normally distributed about the mean shape of PDM j.

4.3 Multi-Cue Observation

Three cues are integrated in this work, which contribute to the particle weights: shape, texture, and stereo. Their distributions are assumed conditionally independent so that

$$p(\vec{z_k}|\vec{x_k} = s_k^i) = p_{shape}(\vec{z_k}|\vec{x_k} = s_k^i)\; p_{texture}(\vec{z_k}|\vec{x_k} = s_k^i)\; p_{stereo}(\vec{z_k}|\vec{x_k} = s_k^i). \tag{4.8}$$

Since the shape and texture similarity measures between the prediction and observation are defined in the image plane, the shape of each particle is generated using Equation 3.19. Its centroid coordinates u, v and the scale factor s are derived using a simple pinhole camera model with known (intrinsic/extrinsic) parameters from the 3D-coordinates x, y, z and the specified 3D object dimensions.

4.3.1 Shape

A method based on multi-feature distance transforms [53] is applied to measure the similarity between the predicted shapes and the observed image edges. See Figure 4.1. In contrast methods that directly match contour features with image features, matching based on distance transforms results smoother modes in the measurement density. Particles that are close to the correct solution, will therefore also get high weights.

As features the position and direction of edge elements are considered. Formally, if the image I is observed at time k and S is the shape of particle s_k^i

$$p_{shape}(\vec{z_k}|\vec{x_k} = s_k^i) \propto \exp(-\alpha_{shape}(\frac{1}{|S|}\sum_{s \in S} D_I(s))^2), \tag{4.9}$$

whereas $|S|$ denotes the number of features s in S, $D_I(s)$ is the distance of the closest feature in I to s, and α_{shape} is a user specified weight. In the experiments eight direction classes are considered.

4.3.2 Texture

The texture distribution over a region $\mathbf{R} = (u_1, v_1, u_2, v_2, ..., u_{n_R}, v_{n_R})$, given by its n_R u- and v-coordinates, is represented by a histogram $\theta_{\mathbf{R}} = \{\theta^r\}_{r=1,...,m}$, which is divided into m bins. It is calculated as follows

$$\theta_{\mathbf{R}}^r = \frac{1}{n_R}\sum_{i=1}^{n_R} \delta(h(u_i, v_i) - r), \tag{4.10}$$

whereas $h(u_i, v_i)$ assigns one of the m bins for the grey value at location u_i, v_i and δ is the Kronecker delta function.

To measure the similarity of two distributions $\theta_1 = \{\theta_1^r\}_{r=1,...,m}$ and $\theta_2 = \{\theta_2^r\}_{r=1,...,m}$ the Bhattacharyya coefficient was selected (amongst various possibilities [127]), which proved to be of value in combination with tracking [111, 115]

$$\rho(\theta_1, \theta_2) = \sum_{r=1}^{m} \sqrt{\theta_1^r \theta_2^r}. \tag{4.11}$$

$\rho(\theta_1, \theta_2)$ ranges from 0 to 1, with 1 indicating a perfect match. The Bhattacharyya distance $d(\theta_1, \theta_2) = \sqrt{1 - \rho(\theta_1, \theta_2)}$ can easily be calculated from the coefficient.

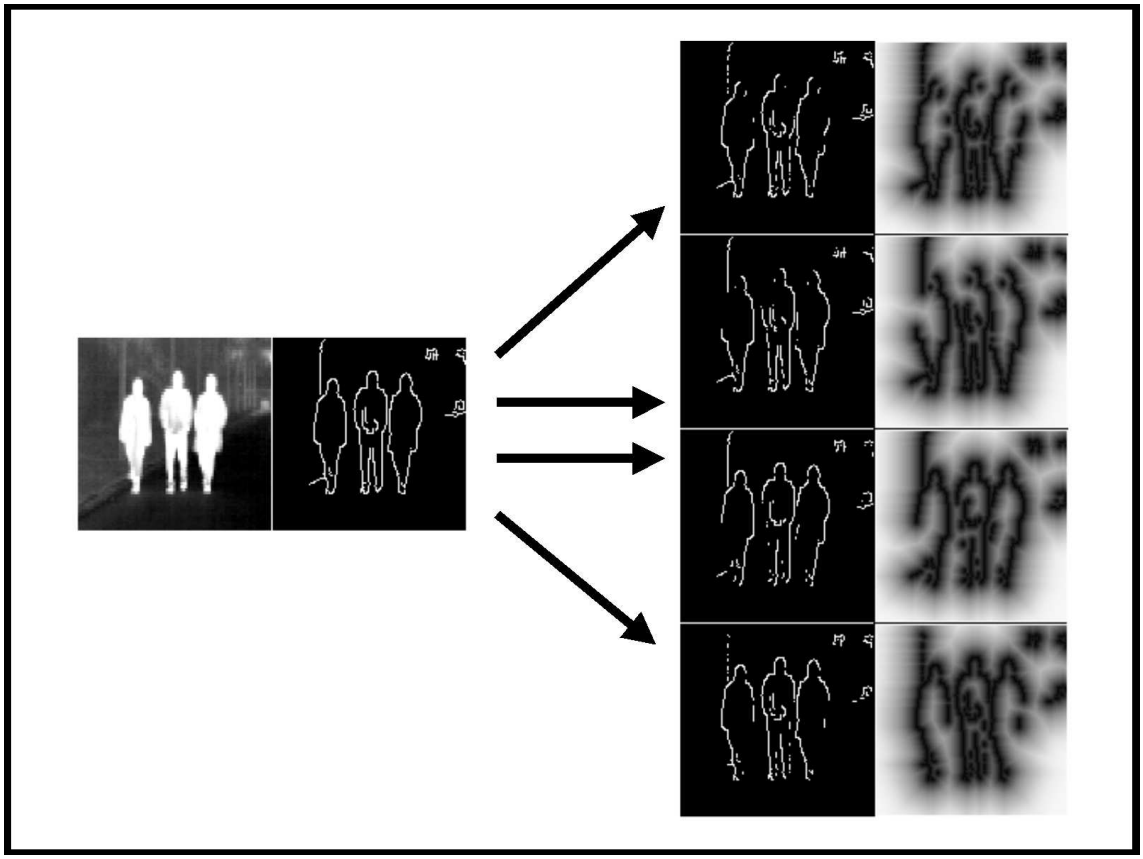

Figure 4.1: Multi-feature distance transform. In the example, features (e.g. edges) are subdivided into multiple (e.g. four) direction classes and a distance transform is applied for each. The distance transformed images contain the distance to the closest feature for each pixel. For matching, a direction class is determined for each shape point and they are correlated with the corresponding image.

For tracking, a reference distribution θ is calculated at track initialization, which is updated over time to compensate for small texture changes. As in [111] the update is done with the mean histogram $\bar{\theta}$ observed under the shape of all particles

$$\theta^r_{t+1} = \alpha\bar{\theta}^r_t + (1-\alpha)\theta^r_t. \tag{4.12}$$

The user specified parameter α controls the contribution of the previous reference and the observed mean histograms.

To summarize, $p_{texture}(\vec{z_k}|\vec{x_k} = s^i_k)$ is defined as

$$p_{texture}(\vec{z_k}|\vec{x_k} = s^i_k) \propto \exp(-\alpha_{texture} d^2(\omega, \theta)), \tag{4.13}$$

whereas $d(\omega, \theta)$ is the Bhattacharyya distance between the reference distribution θ and

the observed texture distribution ω under the shape of particle s_k^i. Like above, $\alpha_{texture}$ is a user defined weighting factor.

4.3.3 Stereo

A real-time stereo vision module generates a depth image I_{depth}, which contains the distance to certain feature points. See Figure 4.2. Details of the stereo algorithm are presented in [48], while a recent evaluation of different methodologies can be found in [131].

To measure the depth d_{stereo} of particle s_k^i the distance of the feature points under its shape are averaged. Given the predicted distance z of s_k^i and the measurement d_{stereo}

$$p_{stereo}(\vec{z_k}|\vec{x_k} = s_k^i) \propto \exp(-\alpha_{stereo}(d_{stereo} - z)^2), \tag{4.14}$$

whereas α_{stereo} is a weighting factor.

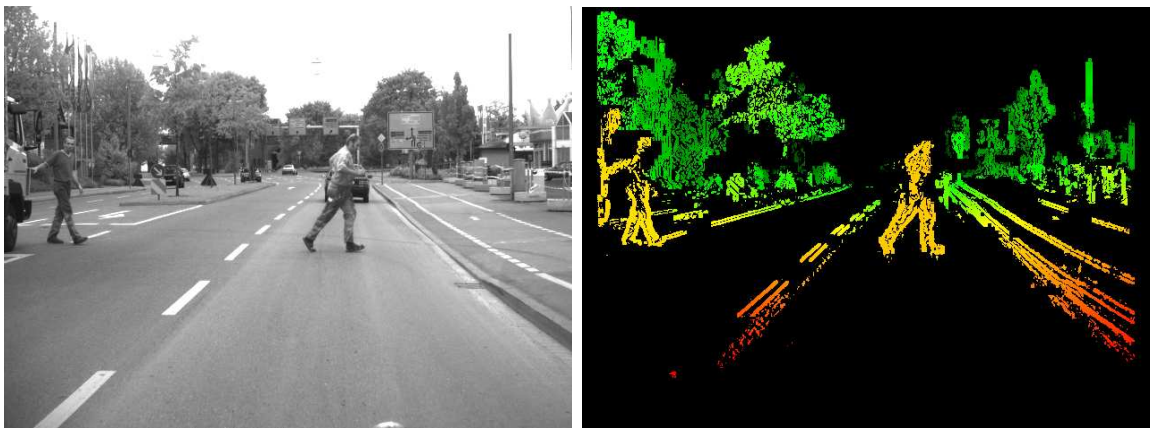

Figure 4.2: Stereo computation. From the original images a stereo algorithm calculates the depth at certain feature points. In the right image the depth information is coded in the color ranging from green (far) to red (near).

4.4 Integrated Detection and Tracking

Most multiple target tracking algorithms fall into one of two categories. The first initiates a single target tracker for each observed object. Several techniques exist to associate detections with active tracks and a variety of strategies for handling occlusions have been considered. The second category extends the state space of a single tracking algorithm to include components for each object.

In principal particle filters offer the possibility to estimate multimodal probability densities over state spaces, which contain multiple objects, each corresponding to one mode. In practice however, several problems arise. One is known as “sample depletion”

in literature and is illustrated in Figure 4.3. A particle filter is initialized with uniform distributed samples to approximate a bimodal probability density. After a short time most of the samples are attached to only one mode, although it has only a slightly higher probability. A bit later, the second mode completely vanishes. Another problem occurs,

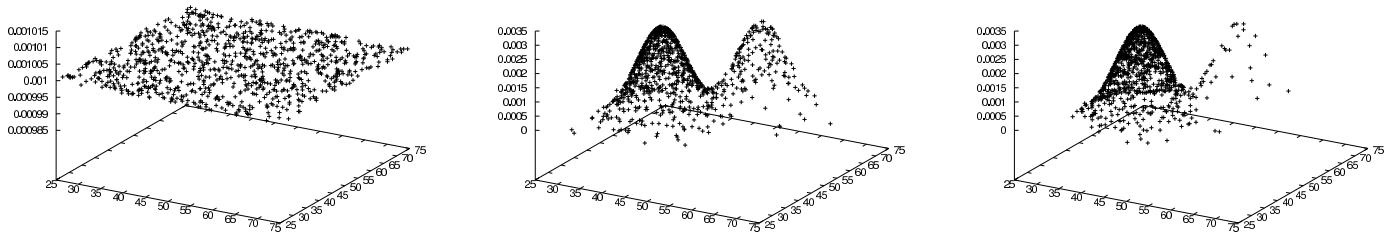

Figure 4.3: Multiple traget tracking. Sample depletion.

when the number of objects and their configuration have to be determined from the sample set. In other words, the modes of the probability density function have to be estimated, which is a non-trivial task for high dimensional state spaces and sparse set of samples.

Therefore this work follows the first strategy of initiating a single particle filter for each detected object. A particle filter is in one of the states *active* or *inactive*. An *active* track is either *visible* or *hidden*.

A detection system provides possible object locations, which are modeled as "mixture" of Gaussians, whereas one Gaussian corresponds to one detection. The mixture is exploited in two ways: As importance function for the particle filters and for the initialization and termination of tracks.

The following rules, which depend on the actual detections, observations, and geometric constraints control the evolution of tracks.

A track is initialized, if no mean state of an *active* track is in the 3σ-bound of a detection. To suppress spurious measurements it starts *hidden*. Initialization involves drawing the 3D position and velocity of the particles according to the Gaussian of the detection. (In this work, the detection system does not provide velocity estimation. Instead, it is assumed to be zero mean Gaussian distributed.) Since no shape information is provided by the detection system, the cluster membership and the continuous parameters are randomly assigned. After the first particle weighting the reference texture distribution is initialized with the mean histogram observed under the shape of all particles.

A track is *visible*, if it has at least t_1 associated detections, if the last associated detection is at most t_2 time steps old, and if the actual match values were better than user defined thresholds for the last t_3 times. Otherwise the track is *hidden*.

A track becomes *inactive*, if the prediction falls outside the detection area or image, if the actual match values (shape, texture, stereo) were worse than user specified thresholds

for t_4 successive times, or if a second track is tracking the same object. (If an object is in the 3σ bounds of two tracks, the shorter track is deleted.)

4.5 Sampling

For particle filtering an algorithm based on *icondensation* [83] is applied. It integrates the standard factored sampling technique of *condensation* [12], importance sampling, and sampling from a "reinitialization" prior probability density.

This allows us to integrate the mixture of Gaussians from the detection system as an importance function into the tracking framework. Like in [83], it is also used as a reinitialization prior, which gives us the possibility to draw samples independently of the past history using the Gaussian of the nearest detection.

A two step sampling approach is followed for the decomposed dynamics of the mixed discrete/continuous shape space [74, 84]. At first the cluster of our shape model is determined using the transition probabilities $T_{i,j}$ and afterwards the weight vector $\vec{b}$ is predicted according to the Gaussian assumptions described in Section 4.2.

4.6 Deterministic Sample Optimization

A well known disadvantage of particle filtering methods for object tracking is that they are typically much slower than deterministic local optimization techniques. The additional time is needed to calculate the evaluation function for each particle. To speed up the computation, it is therefore desirable, to take as few samples as possible. Unfortunately, fewer samples decrease the performance of the filters.

The idea followed in this work is to use fewer samples, but place them at higher probable locations in state space. Therefore a deterministic optimization technique is integrated into the particle filtering approach.

This is achieved by the iterative optimization technique presented in Section 3.2, which is applied to the best fitting f percent of the samples after the time update of the filter. Typically only three iterations are necessary to get satisfactory results. A comparison between a purely stochastic, deterministic, and hybrid shape tracker can be found in [74].

5 Experiments and Evaluation

5.1 Intelligent Vehicle Applications: Pedestrian Protection Systems

More than 150.000 pedestrians are injured yearly in traffic E.U.-wide, see Table 5.1. Of these, more than 6000 are killed. Pedestrian accidents represent the second largest source of traffic-related injuries and fatalities E.U.-wide, right after accidents involving car passengers. The legislative arm of the E.U. has meanwhile recognized the importance of the problem and is studying proposals for legislating maximum tolerated pedestrian leg- and head- "impact coefficients", in the unfortunate event of a pedestrian being hit frontally by a vehicle at 40 km/h.

	Killed	Injured	Total
Passenger Cars	22.502	995.026	1.017.528
Pedestrians	**6.049**	**155.151**	**161.200**
Mopeds	2.421	141.870	144.291
Bicycles	2.385	139.442	141.827
Motor Cycles	3.821	124.023	127.844
Other	4.559	121.816	126.375
Total	41.737	1.677.328	1.719.065

Table 5.1: Road Traffic Accidents 1997 - Figures for EU (Accident Source: UN-ECE)

The long-term goal is to develop vision-based vehicle safety systems which, if they cannot avoid these accidents altogether, at least minimize their severity by anticipating dangerous situations ahead of time and initiating appropriate protective measures. The pedestrian recognition application on-board vehicles is in particular challenging because of the moving camera, the wide range of possible pedestrian appearances and the cluttered backgrounds that apply. This is not to mention the hard real-time requirements and stringent performance criteria that need to be met for actual deployment of such safety systems.

Nevertheless prototype systems have been developed and installed into demonstrators. The demonstrator that was used for data collection and testing in the experiments is shown in Figure 5.1. It is equipped with a stereo camera, an industrial PC, a display, and a keyboard with integrated track ball.

The pair of horizontally aligned cameras deliver synchronized, grey-valued images in the visible spectrum. The camera characteristics are summarized in Table 5.2. Note, that the images are scaled down to 256×192 to speed up computation in the experiments. For efficient stereo computation, the cameras need to be aligned, so that their optical axis are parallel and points at infinite distance meet at the same image coordinates in both images.

Over a CAN bus card, the PC communicates with the vehicles communication system. For obstacle/pedestrian tracking systems the velocity and steering angle are typically

Figure 5.1: Demonstrator vehicle from the outside and inside.

camera parameter	value
focal length	0.012 meter
pixel size	0.00001 meter/pixel
image width	256 (640) pixel
image height	192 (480) pixel
baseline	0.3 meter
camera height	1.25 meter
pitch angle	0.024 radians

Table 5.2: Camera parameters.

read, since they can easily be used to improve the prediction stage of these systems. In modern vehicles like the Mercedes-Benz E-Class, it is also possible to control the vehicle speed over the CAN bus. Emergency braking in case of dangerous situations in front of the vehicle, can easily be initiated.

In the following, the experiments and results of the methodologies and algorithms examined in this Thesis to build and improve pedestrian protection systems are described.

5.2 DPDM Representation

For the experiments, a DPDM was trained from about 2500 pedestrian shapes. The shapes were scaled to a height of 80. Clustering was done with a threshold $\theta = 5.5$ (Section 3.1.2), which resulted 25 clusters. Due to the simple greedy clustering algorithm, several clusters with only a few elements appeared. They contained either exceptional or bad labeled pedestrian shapes. Therefore all clusters with less than 5 elements were removed for the tracking and snake experiments. Eight clusters remained. In the "virtual

sample generation" experiment, where the focus lies on the representation itself, the complete model with all clusters was examined.

The transition probabilities for the reduced "Dynamic Point Distribution Model" is illustrated in Figure 5.2. As expected, the diagonal elements of the transition matrix contain high values, so that there is always a high probability of staying in the same PDM during tracking. This is desirable, since each PDM transition causes a discontinuous change of the PDM parameters, because after a transition, a track is initialized at the origin of the target PDM.

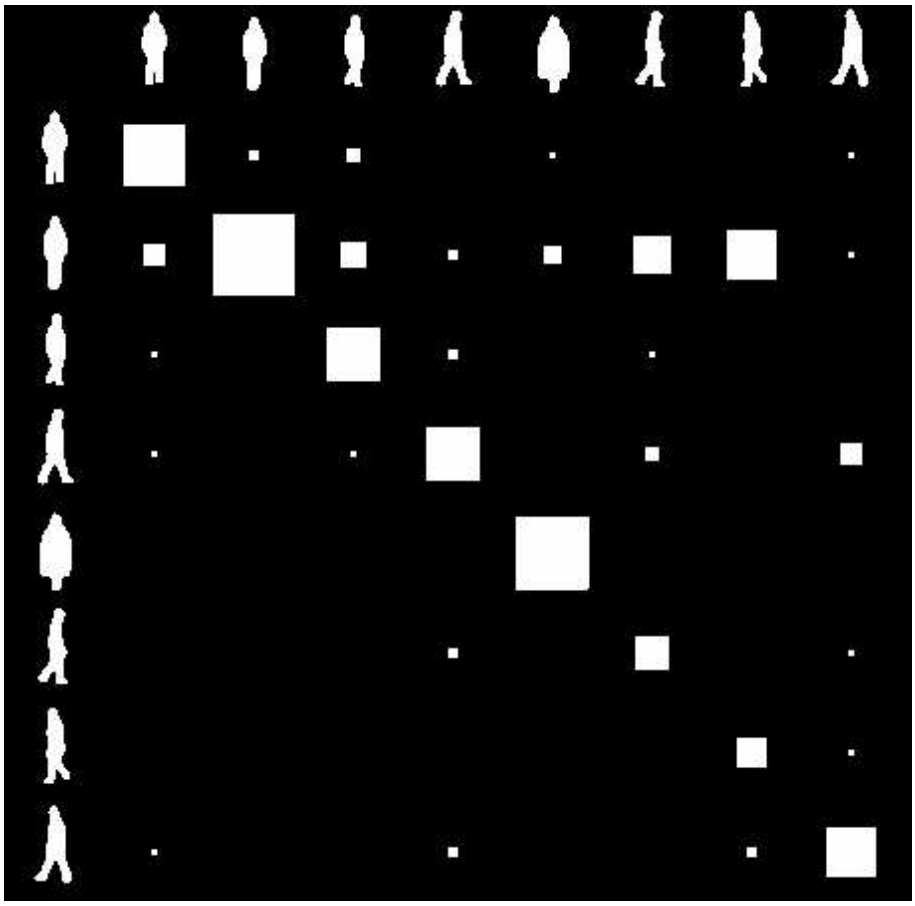

Figure 5.2: PDM transition matrix: The squares represent the transition probabilities from a PDM of column j to row i.

5.3 DPDM Snakes

DPDM Snakes can be applied to segment objects of a known class (e.g. pedestrians) from the background of an image, given an initial starting position, scale, and rotation.

For the methodology the reader is referred to Section 3.2. In the experiment shown in Figure 5.3 a DPDM model, is used to segment a pedestrian from the background.

The number of modes for the eight resulting PDMs was chosen to account for 99% of variance contained in the training data. During search the weight vectors $\vec{b}$ are scaled so that they remain within a hyperellipsoid with radius 2.5 about the PDM origins. This ensures, that the fitted shapes remain quite close to the ones found in the training set.

While the length of the training shapes was approximately 160 pixel, the length of the contour normals were set to 13. It has to be chosen in a way, that the image structures are within their reach. The initial contour is shown in white in the upper left image, while the resulting shape after 50 iterations is shown in the lower right.

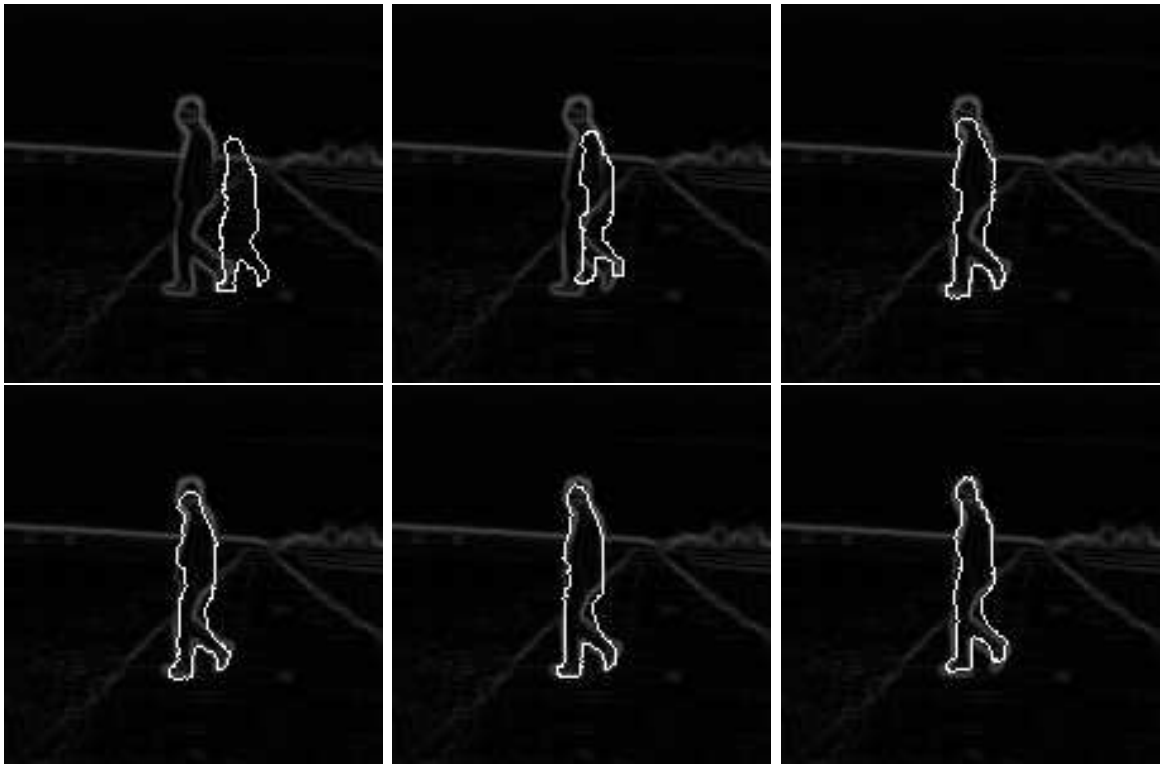

Figure 5.3: DPDM Snake example. About 50 iterations are necessary to match the contour to the image features, if only a rough starting position is given.

The algorithm runs in 0.5 seconds on a Intel(R) Pentium(R) 4 CPU with 2.40GHz with the parameters described above and if the optimization is done for each PDM. If the prediction (or output) from a tracker is used as initial snake position, typically only 3 to 7 iterations are required to optimize the fit. Since the computation time scales (approximately) linearly with the number of iterations, only 0.05 seconds are necessary in that case for each fit.

5.4 Virtual Sample Generation

In the following, experiments on automatically enlarging training sets with "virtual" samples is presented. (For the methodology the reader is referred to Section 3.3.) Generating virtual samples is not beneficial by itself. The purpose of the experiments is to demonstrate that an original set of training shapes can be extended in a way that it produces concrete advantages in terms of matching performance, offsetting inevitable drawbacks related to increased matching effort and higher memory requirements.

Earlier work showed the promise of a template matching system based on distance transforms (e.g. [55]). Therefore, correlation with distance transforms is chosen as basic matching tool for the experiments, in particular the chamfer distance [5].

Matching using chamfer distance in its most basic form is illustrated in Figure 5.4. It involves two binary images, a template (Figure 5.4 upper right) and an edge image (Figure 5.4 lower left) derived from the original scene image (Figure 5.4 upper left). Matching consists of applying the chamfer distance transform on the edge image, resulting in a distance image (Figure 5.4 lower right) which contains at each pixel an approximation of the Euclidean distance to the nearest feature pixel in the corresponding edge image. See Figure 5.4d, increasing distances are shown in lighter shades of grey. The template is then positioned over the distance image, and one considers the pixel values of the distance image which lie beneath the feature pixels of the template. The lower these distances are, the better the match. One typically averages the individual distance contributions to obtain an overall measure of match.

The data set contained roughly 5000 pedestrian shape templates, extracted from real images in a quite time-consuming manual labeling process. The data set was subdivided in a training and test set of about 2500 shape templates each, with no overlap between the two sets. The training set consisted only of the pedestrian templates. The test set contained, in addition, a total of 5000 noise or "garbage" templates, cropped image edge structures which did not contain pedestrians. These garbage template were selected from those image parts which showed already a good chamfer distance match with respect to the pedestrian templates (i.e. were not sampled randomly).

The original training set of 2500 shapes was extended using the methods described in Section 3.3. The number of modes was chosen to describe 95% of variance and the shapes were constrained to lie in a hyperellipsoid of radius 3. Re-sampling was done in a manner that increased the original training size by a factor of 6. The newly generated virtual samples replaced, rather than augmented, the original samples in the extended training set. Two cases are now considered, the test set against the original training set, and the test set against the extended training set.

Figure 5.5 shows the four relevant cumulative histograms of chamfer distances between the object classes in the training and test set, for the original and extended training set. The underlying distance distributions were computed by looping through the elements of the relevant object class in the test set and aggregating the minimum chamfer distances (i.e. the "best" match) with respect to the pedestrian templates in the training set.

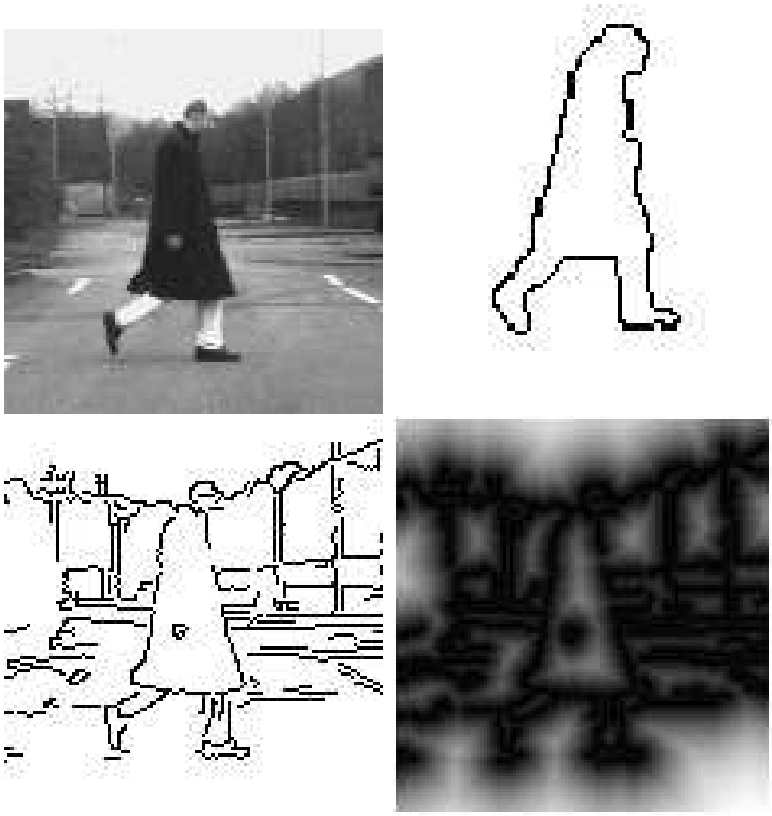

Figure 5.4: Chamfer Matching. (upper left) original image (upper right) template (lower left) edge image (lower right) DT image

From Figure 5.5 one observes that, the pre-selection notwithstanding, elements of the garbage class are more dissimilar to those of the pedestrian class, than the elements of the pedestrian class are among themselves (see dark grey versus white histograms and black versus light grey histograms, respectively). Furthermore, the distances from both the pedestrian and garbage test set to the training set decrease, when extending the latter. See Figure 5.6 for the (virtual) pedestrian-garbage template pairs with the lowest chamfer dissimilarity measure.

From the two cumulative histograms involving the original (or extended) training set, one derives the corresponding ROC curve, the rate of correct detections versus false positives. This is done by considering a particular chamfer distance threshold (x-axis of histogram) and identifying the fraction of pedestrian and garbage samples which have lower distance values (the detection and false positive rate, respectively). The resulting two curves are shown in Figure 5.7. The figure quite convincingly demonstrates the benefit of our virtual shape generation procedure; one observes that in a sizable

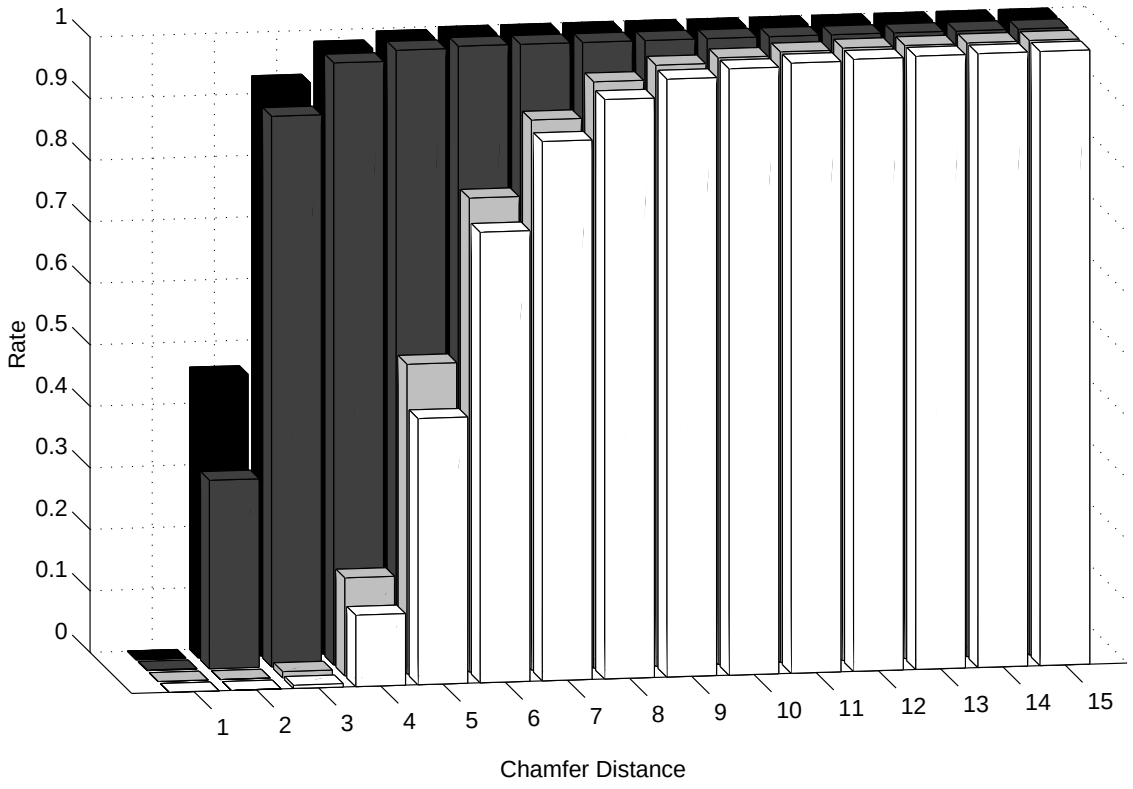

Figure 5.5: Cumulative histogram of chamfer distances. original pedestrian vs. pedestrian (dark grey), original pedestrian vs. garbage (white), extended pedestrian vs. pedestrian (black), extended pedestrian vs. garbage (light grey). First and second term refer to object classes in training and test set, respectively.

(and application-relevant) detection rate interval (0.75 - 0.90), the ROC involving the extended training set outperforms the one related to the original training set. For this interval, the false positive rate is reduced by at least factor 1.5 by equal detection rates. But Figure 5.7 also illustrates another important point, that by increasing the detection rate above a certain threshold (here at about 0.93) one no longer profits from a given virtual sample set. At a certain point, the associated chamfer distance thresholds become so large that any representational "gaps" in the pedestrian distribution are already covered by the original training samples. Increasing the distance thresholds further will now mainly increase the false positive rate. From Figure 5.7 one observes that this turning point occurs at a high false positive rate exceeding 0.015 for the current virtual sample set. On the other hand, one could try to sample the object shape distribution even more finely, to attempt to "squeeze" yet more performance out of the ROC curve.

5.5 Pedestrian Tracking

5.5.1 DPDM Tracking

To evaluate the tracker, experiments on pedestrian tracking from a moving vehicle is considered. The probability density density function for each track is approximated with 500 samples. 90 per cent of the samples are updated using standard factored sampling, while the remaining 10 per cent are updated by the importance function based on the detection system. To improve the speed and convergence of the purely stochastic particle filter, 10 per cent of the samples are deterministically optimized by the method described in chapter 3.2. Three iteration steps were applied with a normal length of 9.

A track is *visible*, if it has at least two associated detections, if the last associated detection is at most three time steps old, and if the actual match values were better than user defined thresholds for the last three times. Otherwise the track is *hidden*. A track becomes *inactive*, if the prediction falls outside the detection area or image, if the actual match values (shape, texture, stereo) were worse than user specified thresholds for four successive times, or if a second track is tracking the same object. (If an object is in the 3σ bounds of two tracks, the shorter track is deleted.)

Transitions

Figure 5.8 shows three random trajectories generated with the DPDM described in Section 5.2. The camera is moving at 5m/s towards the object in 3D-space, which is moving laterally with 1m/s. Each greyscale change corresponds to a PDM transition.

During tracking, the a-priori and a-posteriori probability of each PDM can be observed online, as shown in Figure 5.9. The size of the blue and yellow boxes indicate the a-priori and a-posteriori probability respectively. The more similar they are, the better the prediction.

Multi-Cue

To substantiate the visually observable improvement due to the integration of shape, texture, and stereo information, the position estimates of the system were compared

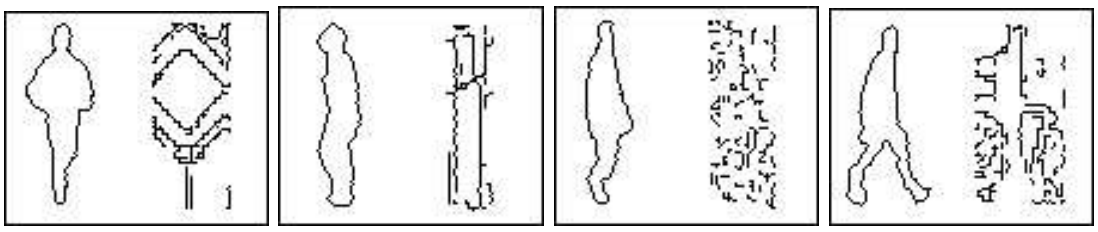

Figure 5.6: The most similar (virtual) pedestrian-garbage pairs in the data set.

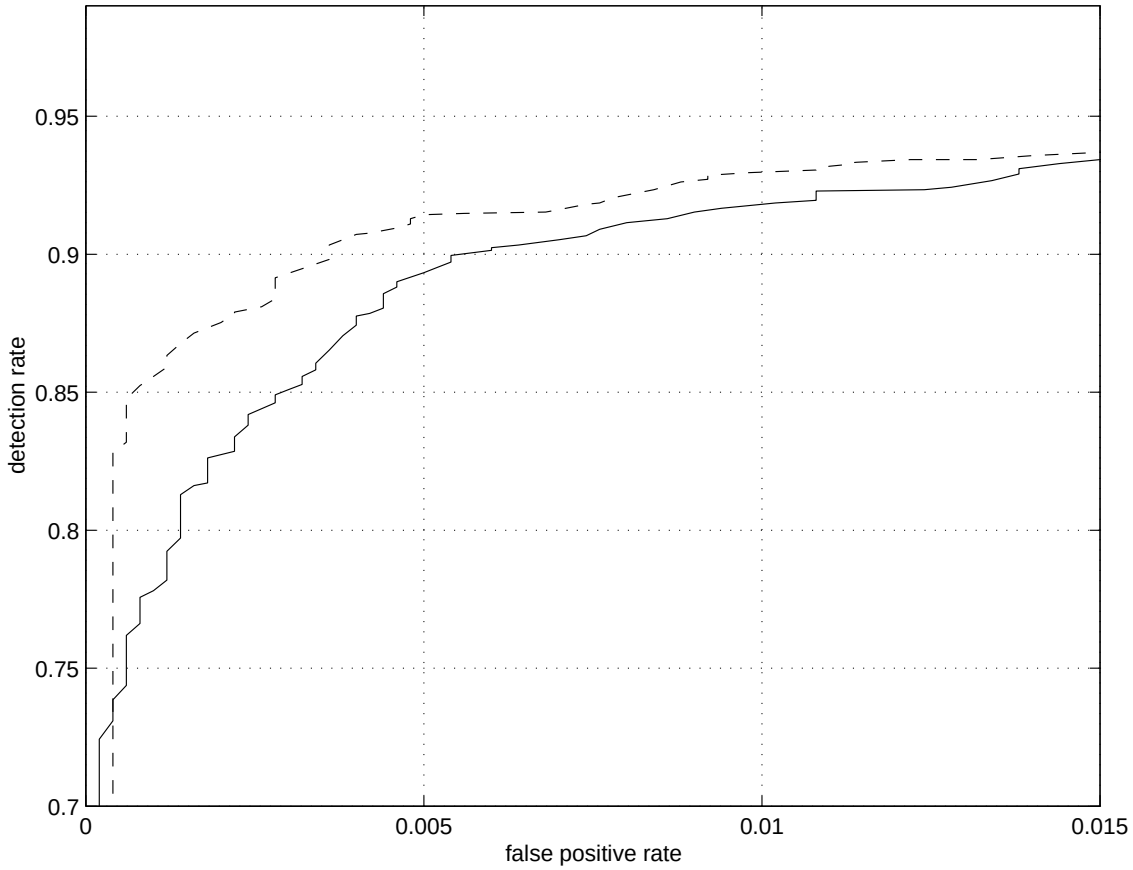

Figure 5.7: ROC performance for the original training set (solid line) and the extended training set (dotted line).

against ground truth.

The weighting factors α_{shape}, $\alpha_{texture}$, and α_{stereo} of Equation 4.9, 4.13, and 4.14 convert the distances between the measurements and tracks into similarities and simultaneously weight the individual cues. They were chosen in a way that tracking is possible with each cue on its own ($\alpha_{shape} = 1.0$, $\alpha_{texture} = 0.1$, and $\alpha_{stereo} = 2.0$). In future work, they may be learned and/or adapted in time.

Table 5.3 shows the results for the first sequence of Figure 5.13, which consists of 34 images.

As expected, the accuracy of the lateral position is higher than the longitudinal and an improved performance due to the integration of multiple cues can be observed.

Tracks

To track the pedestrians in Figure 5.10 and 5.11, a random population of samples was generated about a user supplied starting position. During tracking, the "modal" shape,

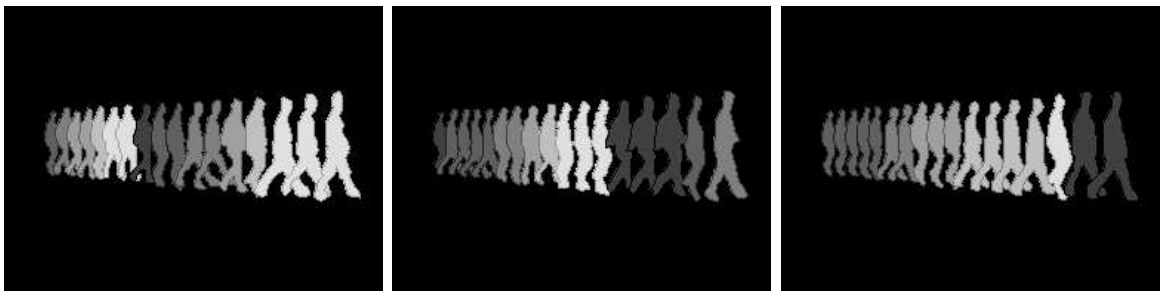

Figure 5.8: Predicting shape changes using Dynamic Point Distribution Models: Three random trajectories assuming that the camera is moving at 5m/s towards the object, which is moving laterally with 1m/s. Each greyscale change corresponds to a PDM transition.

cues	Euclidean distance	lateral error	depth error
edge	1.37m	0.085m	1.34m
edge + texture	1.28m	0.14m	1.25m
edge + texture + stereo	1.05m	0.11m	1.03m

Table 5.3: Average distances of the true and estimated pedestrian locations for a sequence of 34 images.

the one with the highest sample weight, is shown in black or white. Note that because of the discrete parameter in our state space and the different object parameterizations it is no longer possible to evaluate the mean properties of the posterior directly from the samples as in [81].

In Figure 5.10 the results of two different tracks are displayed. Although the scene is quite complex, the trackers correctly keep lock on the desired targets.

To illustrate the estimated probability density function in Figure 5.11 the complete sample set is projected onto the edge image and shown in black. The first part of the sequence is particularly difficult for the tracker, because of the low contrast between the upper part of the body and the background. One observes that the sample set splits while the pedestrian passes the bushes behind him. The ambiguity is solved, because the "correct" mode of the estimated probability function dominates according to the observation density in time.

5.5.2 Comparison

The DPDM tracker was integrated into the PROTECTOR system [60], which previously utilized a simple α-β bounding box tracker. In the following the PROTECTOR system is briefly reviewed and the results of both systems are compared.

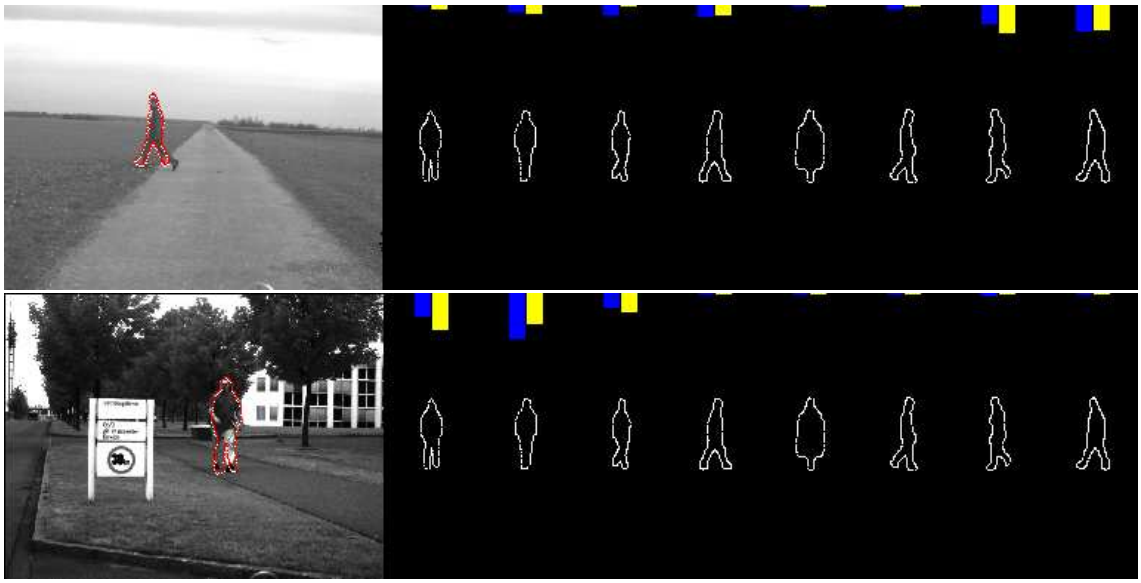

Figure 5.9: Tracking results. The blue box indicates the a-priori and the yellow the a-posteriori confidence of each cluster. The larger the box the higher the probability.

PROTECTOR System

The PROTECTOR system consists of several modules, see Figure 5.12.

- *Stereo pre-processing* performs obstacle detection and provides an initial area of interest. A depth map is computed in real-time by an hierarchical feature-based stereo algorithm [48]. The depth map is multiplexed into N different discrete depth ranges, which are subsequently scanned with windows related to minimum and maximum extents of pedestrians, taking into account the ground plane location at a particular depth range and appropriate tolerances. The locations where the number of depth features exceeds a percentage of the window area are added to the ROI point list of the template hierarchy of the Chamfer System.

- The *Chamfer System* [55] performs shape-based pedestrian detection: a hierarchy of pedestrian templates is matched with distance-transformed images in a tree traversal process. This method efficiently "locks onto" desired objects in a coarse-to-fine manner. A maximum chamfer distance is given as a threshold for each hierarchy level which determines whether child nodes are to be traversed, or whether a detection was made at the leaf level.

- *Texture classification* involves a neural network with local receptive fields [152] to verify the Chamfer System detections. An image patch extracted from the bounding box of a detection is scaled to a standard width and height and fed into

Figure 5.10: DPDM tracking.

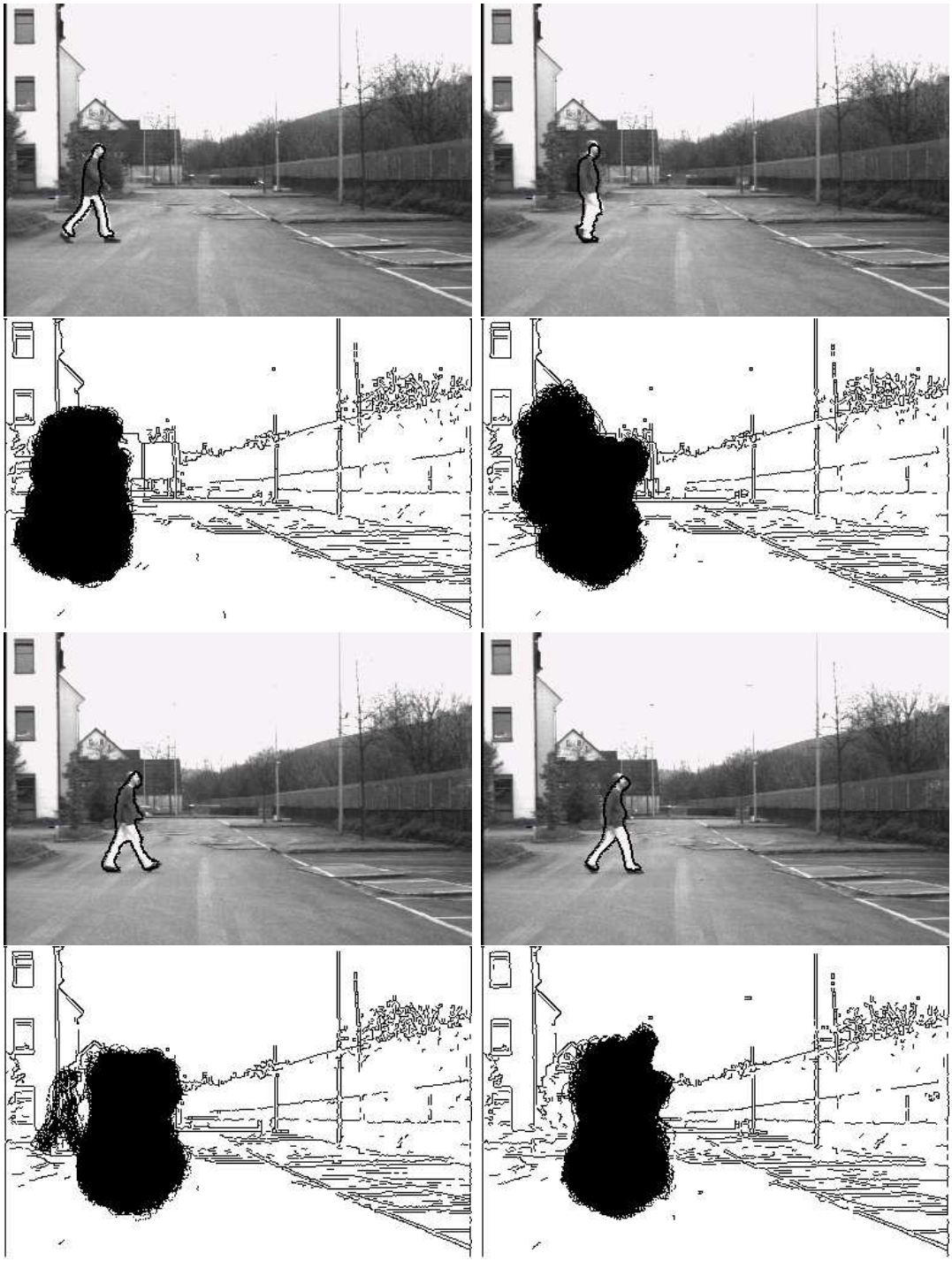

Figure 5.11: DPDM Tracking Results. To visualize the estimated probability density function of the tracker, the sample set is projected onto the edge image. Note that the sample set splits, while the pedestrian passes the bushes behind him. The ambiguity is solved because the correct mode of the probability density function dominates over time.

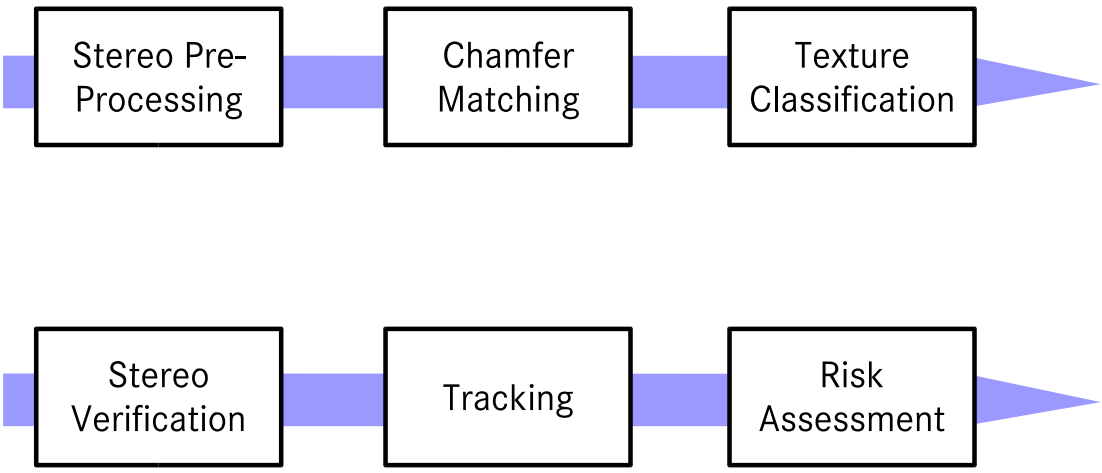

Figure 5.12: PROTECTOR system modules. Stereo preprocessing, Chamfer matching, texture classification, stereo verification, tracking, and risk assessment.

the neural network. Detections for which the output of the neural network is below a user-defined confidence are discarded.

- *Stereo verification* is a second verification approach to filter out false detections on the background. The shape template masks out background pixels for a dense cross-correlation between both stereo images within a certain disparity search range. A threshold is enforced on both height and spread of the resulting correlation function.

- *Tracking* consolidates detection results over time, discarding spurious detections; a rudimentary α-β tracker (refer Section 2.2.3) is used based on a 2.5 D bounding box object representation.

- Finally, the *Risk assessment and driver warning* module computes a risk level for each detected pedestrian based on its position and time-to-collision and issues an acoustical driver warning if it exceeds a certain limit.

DPDM System

In the DPDM system the α-β tracker is substituted by the DPDM tracker introduced in Chapter 4.

Results of the DPDM system are given in Figure 5.13 for urban, rural, and synthetic environments. In the original images (left column) the best sample of each active track

is shown in red. Whenever detections are observed, they are also represented there as grey boxes. The shapes of all particles, which approximate the posterior pdf, are drawn in the middle column. Finally, a top view of the scene can be viewed on the right. The past trajectory is represented by the blue dots while the current position estimate is marked red. The yellow text contains the actual distance and velocity estimates.

The tracking system was tested on a 2.4GHz standard workstation and needs about 300ms/frame for an active track, an image resolution of 256×196.

Results

The following definitions and test criteria were used for the comparison:

- An area from 10 to 25 m in front of the vehicles and 4 m to each side is defined as the *Sensor Coverage Area*. The performance of the system is evaluated in that area, although functionality might not cease immediately beyond these borders.
- *Localization Tolerance* is defined as a percentage of the distance, 10% in lateral and 30% in longitudinal direction.
- For *Data Assignment*, a "group rule" is employed, i.e. a system output is considered correct if there is at least one ground truth object within the localization tolerance of that object and vice versa.
- We distinguish two types of trajectories: those that have at least one entry matched ("B-class") and the higher-quality trajectories that have at least 50% of their entries matched ("A-class").
- Ground truth objects are distinguished by *Risk Level*. A pedestrian is considered "risky" if it is located within or moving towards the vehicle's trajectory. A second evaluation is made taking only "risky" pedestrians into account.

Performance is measured as *Sensitivity* vs. *Precision*, which are defined as

$$\begin{aligned} \text{Sensitivity} &= \frac{\text{Number of detected ground truth objects}}{\text{Number of all ground truth objects}} \\ \text{Precision} &= \frac{\text{Number of correct system outputs}}{\text{Number of all system outputs}} \end{aligned}$$

A sequence taken on a 24 min drive in urban environment was taken for the testing. It consist of 17390 images, and 370 instances of pedestrians originating from 29 pedestrians ("trajectories") within the sensor coverage area. Of them, 182 instances, corresponding to 18 pedestrians are considered risky.

Table 5.5.2 shows a comparison of the results. As can be seen, the multi-modal tracker outperforms the α-β one. There is a significant benefit at the A-class trajectory

performance, while there is only a small improvement of B-class trajectory sensitivity. This behavior is explained by the fact that the new tracker cannot track objects that have not been detected by the detection stage. But once a pedestrian is detected, tracking is more reliable compared to the α-β tracker.

	α-β tracker	DPDM tracker
All pedestrians		
A-class sensitivity	48.3%	58.6%
A-class precision	32.0%	38.6%
B-class sensitivity	65.5%	69.0%
B-class precision	33.3%	40.0%
"Risky" pedestrians		
A-class sensitivity	72.2%	77.8%
A-class precision	56.0%	57.7%
B-class sensitivity	77.8%	83.3%
B-class precision	56.0%	57.7%

Table 5.4: Comparison. Trajectory performance for the pedestrian protection system. The left column shows the results for the α-β tracker, the right column the results for the DPDM object tracker.

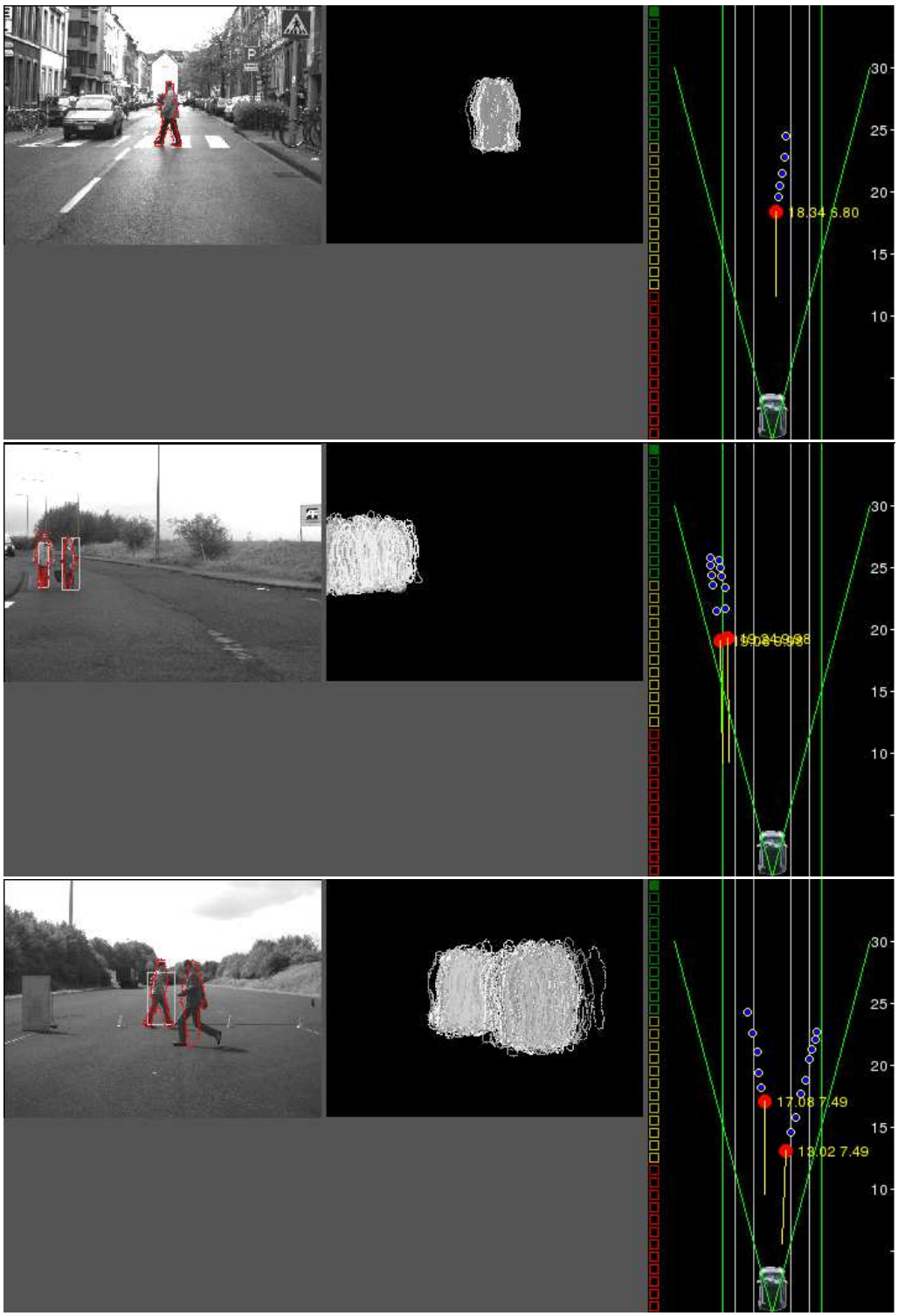

Figure 5.13: Tracking results: In the left images the best sample is shown for each track. In addition the detections are illustrated as boxes. The shapes of all particles, which approximate the posterior pdf, are drawn in the middle. Finally a top view of the scene can be viewed on the right.

6 Conclusions and Outlook

6.1 Conclusions

6.1.1 Dynamic Points Distribution Models

This Thesis presented a general spatio-temporal shape representation called "Dynamic Points Distribution Models" (DPDMs), which can automatically be learned from examples. A set of linear subspace models capture the distribution of object appearance, while the temporal transitions of these models – corresponding to discontinuous appearance changes – are covered by a discrete Markov model. Refer Chapter 3. Due to the improved accuracy of the "locally" linear approach and the ability to model continuous as well as discontinuous appearance changes, the representation is in particular suited for object detection and tracking tasks.

In the experiments, results on the challenging topic of pedestrian recognition from a moving vehicle were presented. A "Dynamic Points Distribution Models" was trained on a realistic data set including thousands of pedestrian shapes and applied to "Virtual Sample Generation" and Bayesian object tracking.

6.1.2 Virtual Sample Generation

A method for enlarging training sets with "virtual" samples was introduced. It involves deriving a parametric representation and a (re-)sampling step. For details, the reader is referred to Chapter 3.3.

Experiments underpin, that this approach is beneficial for a template-based object detection system in terms of improved ROC performance: Higher detection rates could be produced with the same amount of false alarms in the relevant parameter range.

6.1.3 Tracking

A general Bayesian framework for multi-cue 3D deformable object tracking was introduced in Chapter 4. Object shape was modeled by a Dynamic Point Distribution Model. Texture histograms and direct 3D measurements were integrated, to improve the robustness and versatility of the framework. It was presented how measurements from an independently operating detection system can be integrated into the particle filtering approach by means of importance sampling.

Large scale experiments showed, that the proposed framework is suitable for tracking pedestrians from a moving vehicle and improves upon a cutting edge pedestrian protections system, that was designed for the European community funded project PROTECTOR. Especially in difficult situations, in case of occlusions or heavy cluttered scenes, the ability of the particle filter to track multiple hypothesis is beneficial to keep the tracker on the desired targets. Experiments resulted, that the integration of multiple cues in the proposed way increases tracking accuracy.

6.2 Outlook

Future work on "Dynamic Point Distribution Models" includes a more detailed examination of the representation itself. An interesting question is how many clusters are optimal for a particular application. There is a trade off between model accuracy and model specificity. In the extreme case, when each training sample corresponds to a single cluster, the model is highly specific, since it cannot produce invalid instances, but will only be accurate, in the special case that the training set is complete. In the other extreme the training set is described by a single cluster. The model will be quite accurate but not specific, when invalid instances can be generated due to an oversized generalization of the model. Furthermore the number of eigenvectors for the subspace models and the Mahalanobis threshold discussed in Chapter 3 can be analyzed for particular applications.

The number of clusters also effects the temporal modeling. Remembering that the cluster transition matrix grows quadratic with the number subspaces, it becomes clear, that large databases are necessary to learn the transition probabilities for large number of subspaces from examples. This matter will even get worse, when higher order Markov models are considered, which can in principle capture object dynamics more accurately.

From the technical point of view more elaborated clustering approaches could be investigated to replace the simple greedy scheme applied in this work for computational reasons.

In terms of tracking, several areas could further be investigated. For example, the integration of more visual cues (e.g. optical flow) can be considered. Also single cues can be evaluated in several ways. At the moment texture is only assumed to stay constant over time. It is also possible to integrate prior texture knowledge of the objects of interest using standard pattern recognition techniques. Additional benefit can be expected by an adaptive weighting of the cues.

Further testing might include a stability study of the filter. This also involves the interesting question of how many samples are necessary for a particular application. Advanced data association strategies can be applied to associate detections with tracks to substitute the simple nearest neighbor rule, which is currently applied.

For the DPDM system, it might also be interesting to pass shape information of the detection system to the tracking framework. For this, a link between the non parametric shape representation of the Chamfer system and the parametric point distribution models needs to be established.

A Tracking in Literature

Ref	Def	Pro	Lea	Clo	MFe	MMo	Rea	Application
[114]	x	n	x	x		x		Body Tracking
[44]	x	p	x	x	x			Face Tracking
[18]	x	x	x	x	x		x	Multiple View Surveillance
[93]	x	p	x	x	x		x	Lip Tracking
[84]	x	n	x	x		x	x	Ball Tracking
[42]	x			x	x			Polyline Tracking
[74]	x	n	x	x		x	x	Hand Tracking
[89]	x	p		x	x		x	Body Tracking
[72]				x	x		x	Face Tracking
[154]				x	x			Rigid-Trackig
[38]		p		x		x	x	Arm Tracking
[108]	x	p	x					Probabilistic Learning
[97]	x	p	x	x				Face Tracking
[116]	x	p		x	x	x		Hand/Leg Tracking
[49]	x		x	x				Lip Tracking
[98]	x	p		x	x		x	Arm Tracking
[103]				x			x	3D-Rigid-Trackig
[15]		p	x					Surveillance
[2]		p		x			x	3D-Rigid-Trackig
[126]	x	p	x	x				Body Tracking
[20]				x	x		x	Surveillance
[75]	x							Blob Tracking
[80]	x			x	x		x	Children Tracking
[28]		p		x	x		x	Head Tracking
[69]	x			x			x	Ants Tracking
[147]								Feature Tracking
[106]	x	x		x	x			Heart Tracking
[32]	x	x	x	x		x	x	Active Face Tracking
[57]				x				Human Body Tracking
[40]				x			x	3D Tracking
[102]	x	n	x	x		x	x	Hand Tracking
[8]	x	n	x	x	x	x	x	Gesture Recognition
[70]	x			x	x		x	Surveillance
[85]	x	n	x	x		x	x	Hand Tracking
[34]	x	p		x			x	Face Tracking
[79]		p		x			x	Template Tracking
[37]			x	x				3D-Model Tracking

Table A.1: Aspects in literature I (Ref: reference, Def: deformable objects, Pro: probabilistic framework, Lea: learning, Clo: closed loop tracking, MFe: multiple features, MMo: multi modal, Rea: real time; p: parametric, n: non-parametric)

Ref	Def	Pro	Lea	Clo	MFe	MMo	Rea	Application
[140]		p		x	x		x	Agricultural Vehicle
[83]	x	n	x	x		x	x	Hand Tracking
[124]	x	n	x	x	x	x		Background Modeling
[118]	x	n	x	x		x	x	Human Body Tracking
[143]	x	p	x	x		x	x	Surveillance
[137]		n	x	x	x	x	x	Human Body Tracking
[113]	x			x				Cloud Tracking
[135]				x		x	x	Head/Hand Tracking
[156]				x				Cube Tracking
[158]	x			x	x		x	Face/Hand Tracking
[13]	x		x		x		x	Motion Recognition
[43]	x		x					Learning a Shape Model
[129]	x				x			Image Segmentation
[121]	x	p		x	x	x	x	Face/Arm Tracking
[110]	x	p/n	x	x		x	x	Hand Tracking
[30]	x		x					Periodic Motion Detection
[22]	x		x		x			Face Tracking
[18]	x	p		x	x		x	Surveillance (Multi-Camera)
[71]	x	p	x	x	x	x	x	Surveillance
[25]		p		x		x		Corner Tracking
[29]		p		x	x		n	3D-Structure from Motion
[11]	x	p		x			x	Hand Tracking
[94]		p		x	x		x	Road Traffic Scenes
[100]		p		x			x	3D-Motion Tracking
[157]				x			x	Egomotion Compensation/ FT
[76]	x	p		x			?	Articulated Objects (Lamp)
[123]		p		x			x	Active Tracking
[9]	x	p		x			x	Facial Expressions
[151]		p		x		x	x	Road Recognition
[7]	x	p	x	x			x	Texture Tracking
[67]		p		x	x	x	x	Road Traffic Scenes
[145]	x			x	x		x	General Framework
[50]	x		x	x			x	Lip-, Finger-tracking
[33]				x				Face Tracking
[31]					x		x	Face Deformation
[96]	x			x			x	Image Segmentation
[138]				x			x	Motion Segmentation

Table A.2: Aspects in literature II (Ref: reference, Def: deformable objects, Pro: probabilistic framework, Lea: learning, Clo: closed loop tracking, MFe: multiple features, MMo: multi modal, Rea: real time; p: parametric, n: non-parametric)

Ref	Def	Pro	Lea	Clo	MFe	MMo	Rea	Application
[95]		p		x		x	x	Moving Car
[41]	x	p	x	x				Road Traffic Scenes
[51]		p		x				Vehicle Tracking (Interactive)
[112]	x			x				Vehicle Tracking
[35]		p		x	x		x	Constraint Kalman Filter
[153]		p		x			x	Human Body Tracking
[68]	x	p		x	x		x	Face Tracking
[117]	x	p		x	x			Kalman Snakes
[128]	x	p		x	x			Structure from Motion
[125]	x		x		x			Face Modeling
[46]	x	x	x	x				Vehicle Tracking
[58]	x		x					Pedestrian Recognition
[62]	x	n	x	x		x	x	Pedestrian Tracking
[63]	x	n	x	x	x	x	x	Pedestrian Tracking
[59]	x		x		x		x	Pedestrian Tracking
[60]	x		x		x		x	Pedestrian Tracking

Table A.3: Aspects in literature III (Ref: reference, Def: deformable objects, Pro: probabilistic framework, Lea: learning, Clo: closed loop tracking, MFe: multiple features, MMo: multi modal, Rea: real time)

Bibliography

[1] J.K. Aggerwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428–440, 1999.

[2] M. Armstrong and A. Zisserman. Robust object tracking. In *Asian Conference on Computer Vision*, pages V(1)58–61, 1995.

[3] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, pages 100–117, 2001.

[4] Y. Bar-Shalom, X.R. Li, and T. Kirubarajan, editors. *Estimation with applications to tracking and navigation*. Wiley, 2001.

[5] H. Barrow. Parametric correspondence and chamfer matching: Two new techniques for image matching. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 659–663, 1977.

[6] A. Baumberg and D. Hogg. Learning flexible models from image sequences. *Proceedings of the European Conference on Computer Vision*, pages 299–308, 1994.

[7] M.J. Black and A.D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, January 1998.

[8] M.J. Black and A.D Jepson. A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. In *Proceedings of the European Conference on Computer Vision*, pages 909–924, 2001.

[9] M.J. Black and Y. Yacoob. Recognizing facial expressions in image sequences using local parameterized models of image motion. *International Journal of Computer Vision*, 25(1):23–48, October 1997.

[10] M.J. Black, Y. Yacoob, A.D. Jepson, and D.J. Fleet. Learning parameterized models of image motion. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages 561–567, 1997.

[11] A. Blake, R. Curwen, and A. Zisserman. A framework for spatiotemporal control in the tracking of visual contours. *International Journal of Computer Vision*, 11(2):127–145, October 1993.

[12] A. Blake and M. Isard. *Active Contours*. Springer, 1998.

[13] A.F. Bobick and J.W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):257–267, March 2001.

[14] B. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, November 1988.

[15] J. Boyd, J. Meloche, and Y. Vardi. Statistical tracking in video traffic surveillance. In *Proceedings of the International Conference on Computer Vision*, pages 163–168, 1999.

[16] C. Bregler. Learning and recognizing human dynamics in video sequences. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, 1997.

[17] C. Bregler and S.M. Omohundro. Surface learning with applications to lipreading. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 43–50. Morgan Kaufmann Publishers, Inc., 1994.

[18] Q. Cai and J.K. Aggarwal. Automatic tracking of human motion in indoor scenes across multiple synchronized video streams. In *Proceedings of the International Conference on Computer Vision*, pages 356–362, 1998.

[19] T.J. Cham and J.M. Rehg. A multiple hypothesis approach to figure tracking. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages II:23–25, 1999.

[20] I. Cohen and G. Medioni. Detecting and tracking moving objects in video surveillance. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages II:319–325, 1999.

[21] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages II:142–149, 2000.

[22] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–684, June 2001.

[23] T.F. Cootes and C.J. Taylor. A mixture model for representing shape variation. *Image and Vision Computing*, 17(8):567–574, 1999.

[24] T.F. Cootes, C.J. Taylor, D.C. Cooper, and J. Graham. Active-shape-models - their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, January 1995.

[25] I.J. Cox and S.L. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, February 1996.

[26] D. Cremers, T. Kohlberger, and C. Schnörr. Shape statistics in kernel space for variational image segmentation. 36(9):1929–1943, 2003.

[27] D. Crisan and A. Doucet. A survey of convergence results on particle filtering methods for practioners. *IEEE Transactions on signal processing*, 50(3):736–746, 2002.

[28] J.L. Crowley and F. Berard. Multi-modal tracking of faces for video communications. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages 640–645, 1997.

[29] J.L. Crowley, P. Stelmaszyk, T. Skordas, and P. Puget. Measurement and integration of 3-d structures by tracking edge lines. *International Journal of Computer Vision*, 8(1):29–52, July 1992.

[30] R. Cutler and L.S. Davis. Robust real-time periodic motion detection, analyse, and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):781–796, August 2000.

[31] T. Darrell, G.G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. *International Journal of Computer Vision*, 37(2):175–185, June 2000.

[32] T.J. Darrell, B. Moghaddam, and A.P. Pentland. Active face tracking and pose estimation in an interactive room. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages 67–72, 1996.

[33] D. de Carlo and D. Metaxas. Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision*, 38(2):99–127, July 2000.

[34] F. de la Torre, S. Gong, and S.J. McKenna. View-based adaptive affine tracking. In *Proceedings of the European Conference on Computer Vision*, 1998.

[35] J. Degeeter, H. van Brussel, J. Deschutter, and M. Decreton. A smoothly constrained kalman filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1171–1177, October 1997.

[36] Q. Delamarre and O.D. Faugeras. 3d articulated models and multiview tracking with physical forces. *Computer Vision and Image Understanding*, 81(3):328–357, March 2001.

[37] J. Denzler, B. Heigl, and H. Niemann. An efficient combination of 2-d and 3-d shape descriptions for contour-based tracking of moving objects. In *Proceedings of the European Conference on Computer Vision*, 1998.

[38] J. Deutscher, B. North, B. Bascle, and A. Blake. Tracking through singularities and discontinuities by random sampling. In *Proceedings of the International Conference on Computer Vision*, pages 1144–1149, 1999.

[39] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[40] T. Drummond and R. Cipolla. Real-time tracking of multiple articulated structures in multiple views. In *Proceedings of the European Conference on Computer Vision*, 2000.

[41] M.P. Dubuisson-Jolly, S. Lakshmanan, and A.K. Jain. Vehicle segmentation and classification using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(3):293–308, March 1996.

[42] M.P. Dubuisson-Jolly, C.C. Liang, and A. Gupta. Optimal polyline tracking for artery motion compensation in coronary angiography. In *Proceedings of the International Conference on Computer Vision*, pages 414–419, 1998.

[43] N. Duta, A.K. Jain, and M.P. Dubuisson-Jolley. Automatic construction of 2d shape models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(5):433–446, May 2001.

[44] G.J. Edwards, C.J. Taylor, and T.F. Cootes. Learning to identify and track faces in image sequences. In *Proceedings of the International Conference on Computer Vision*, pages 317–322, 1998.

[45] R.J. Elliot, L. Aggoun, and J.B. Moore. *Hidden Markov Models*. Springer, 2nd edition, 1997.

[46] J.M. Ferryman, A.D. Worral, and S.J. Maybank. Learning enhanced 3d models for vehicle tracking. *Proceedings of the British Machine Vision Conference*, 1998.

[47] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, editors. *Computer Graphics Principles and Practice*. Addison Wesley, 2. edition, 1990.

[48] U. Franke. Real-time stereo vision for urban traffic scene understanding. In *Proceedings IEEE International Conference on Intelligent Vehicles*, Detroit, USA, 2000.

[49] D. Freedman and M.S. Brandstein. A subset approach to contour tracking in clutter. In *Proceedings of the International Conference on Computer Vision*, pages 242–247, 1999.

[50] D. Freedman and M.S. Brandstein. Provably fast algorithms for contour tracking. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages I:139–144, 2000.

[51] W.F. Gardner and D.T. Lawton. Interactive model-based vehicle tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1115–1121, November 1996.

[52] D. M. Gavrila. Hermite deformable contours. In *Proceedings of the IEEE International Conference on Pattern Recognition*, pages 130–135, 1996.

[53] D. M. Gavrila. Pedestrian detection from a moving vehicle. In *Proceedings of the European Conference on Computer Vision*, pages 37–49, 2000.

[54] D. M. Gavrila, J. Giebel, and H. Neumann. Learning shape models from examples. In *Proceedings of the Deutsche Arbeitsgemeinschaft für Mustererkennung*, pages pp. 369–376, Munich, Germany, 2001.

[55] D. M. Gavrila and V. Philomin. Real-time object detection for "smart" vehicles. In *Proceedings of the International Conference on Computer Vision*, pages 87–93, Kerkyra, Greece, 1999.

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

[57] D.M. Gavrila and L.S. Davis. 3d model-based tracking of humans in action: A multi-view approach. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages 73–80, 1996.

[58] D.M. Gavrila and J. Giebel. Virtual sample generation for template-based shape matching. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages I:676–681, 2001.

[59] D.M. Gavrila and J. Giebel. Shape-based pedestrian detection and tracking. In *Proceedings IEEE International Conference on Intelligent Vehicles*, 2002.

[60] D.M. Gavrila, J. Giebel, and S. Munder. Vision-based pedestrian detection: The protector system. In *Proceedings IEEE International Conference on Intelligent Vehicles*, 2004.

[61] Y. Gdalyahu and D. Weinshall. Flexible syntatic matching of curves and its application to automatic hierarchical classification of silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1312–1328, December 1999.

[62] J. Giebel and D. M. Gavrila. Multimodal shape tracking with point distribution models. In *Proceedings of the Deutsche Arbeitsgemeinschaft für Mustererkennung*, Zurich, Switzerland, 2002.

[63] J. Giebel, D.M. Gavrila, and C. Schnoerr. A bayesian framework for multi-cue 3d object tracking. In *Proceedings of the European Conference on Computer Vision*, 2004.

[64] C. Goodall. Procrustes methods in the statistical analysis of shape. *Journal of the Royal Statistic Society*, 53(2):285–339, 1991.

[65] J.W. Gorman, O.R. Mitchell, and F.P. Kuhl. Partial shape recognition using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(2):1072–1079, March 1988.

[66] U. Grenander, Y. Chow, and D.M. Keenan, editors. *HANDS, A pattern theoretic study*. Springer, 1991.

[67] M. Haag and H.H. Nagel. Combination of edge element and optical flow estimates for 3d-model-based vehicle tracking in traffic image sequences. *International Journal of Computer Vision*, 35(3):1–25, December 1999.

[68] G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, October 1998.

[69] G. Halevy and D. Weinshall. Motion of disturbances: Detection and tracking of multi-body non-rigid motion. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages 897–902, 1997.

[70] I. Haritaoglu, D. Harwood, and L.S. Davis. W4s: A real-time system for detecting and tracking people in 2 1/2-d. In *Proceedings of the European Conference on Computer Vision*, 1998.

[71] I. Haritaoglu, D. Harwood, and L.S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, August 2000.

[72] M. Harville, A. Rahimi, T. Darrell, G.G. Gordon, and J. Woodfill. 3d pose tracking with linear depth and brightness constraints. In *Proceedings of the International Conference on Computer Vision*, pages 206–213, 1999.

[73] T. Heap and D. Hogg. Improving specificity in pdms using a hierarchical approach. In Adrian F. Clark, editor, *British Machine Vision Conference*, 1997.

[74] T. Heap and D. Hogg. Wormholes in shape space: Tracking through discontinuous changes in shape. In *Proceedings of the International Conference on Computer Vision*, pages 344–349, 1998.

[75] B. Heisele, U. Kressel, and W. Ritter. Tracking non-rigid moving objects based on color cluster flow. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages 257–260, 1997.

[76] Y. Hel-Or and M. Werman. Constraint fusion for recognition and localization of articulated objects. *International Journal of Computer Vision*, 19(1):5–28, July 1996.

[77] A. Hill, C.J. Taylor, and A.D. Brett. A framework for automatic landmark identification using a new method of nonrigid correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):241–251, 2000.

[78] D. Huttenlocher, G. Klanderman, and W.J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.

[79] S. Huwer and H. Niemann. 2-d-object tracking based on projection-histograms. In *Proceedings of the European Conference on Computer Vision*, 1998.

[80] S.S. Intille, J.W. Davis, and A.F. Bobick. Real time closed world tracking. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages 697–703, 1997.

[81] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings of the European Conference on Computer Vision*, pages I:343–356, 1996.

[82] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[83] M. Isard and A. Blake. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In *Proceedings of the European Conference on Computer Vision*, 1998.

[84] M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *Proceedings of the International Conference on Computer Vision*, pages 107–112, 1998.

[85] M. Isard and A. Blake. A smoothing filter for condensation. In *Proceedings of the European Conference on Computer Vision*, 1998.

[86] M. Isard and J. MacCormick. Bramble: A bayesian multiple-blob tracker. In *Proceedings of the International Conference on Computer Vision*, pages 34–41, 2001.

[87] D. W. Jacobs, D. Weinshall, and Y. Gdalyahu. Classification with nonmetric distances: Image retrieval and class representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):583–600, June 2000.

[88] B. Jaehne, editor. *Digitale Bildverarbeitung*. Springer, 4. edition, 1997.

[89] N. Jojic, M. Turk, and T. Huang. Tracking self-occluding articulated objects in dense disparity maps. In *Proceedings of the International Conference on Computer Vision*, pages 123–130, 1999.

[90] S. Julier and J. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, 1997.

[91] Emil Kalman, Rudolph. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[92] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, pages 321–331, 1988.

[93] R. Kaucic and A. Blake. Accurate, real-time, unadorned lip tracking. In *Proceedings of the International Conference on Computer Vision*, pages 370–375, 1998.

[94] D. Koller, K. Daniilidis, and H.H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257–281, June 1993.

[95] A. Kumar, Y. Bar-Shalom, and E. Oron. Precision tracking based on segmentation with optimal layering for imaging sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):182–188, February 1995.

[96] F. Leymarie and M.D. Levine. Tracking deformable objects in the plane using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):617–634, June 1993.

[97] S. Li and J. Lu. Modeling bayesian estimation for deformable contours. In *Proceedings of the International Conference on Computer Vision*, pages 991–996, 1999.

[98] M. Lin. Tracking articulated objects in real-time range image sequences. In *Proceedings of the International Conference on Computer Vision*, pages 648–653, 1999.

[99] H.-C. Liu and M. Srinath. Partial shape classification using contour matching in distance transformation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11):1072–1079, November 1990.

[100] D.G. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122, August 1992.

[101] J. MacCormick, editor. *Probabilistic modelling and stochastic algorithms for visual localisation and tracking*. PhD thesis, University of Oxford, 2000.

[102] J.P. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *Proceedings of the European Conference on Computer Vision*, 2000.

[103] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau. Robust real-time visual tracking using a 2d-3d model-based approach. In *Proceedings of the International Conference on Computer Vision*, pages 262–268, 1999.

[104] P. S. Maybeck. Stochastic models, estimation and control (vol. 1, 2, 3). *Academic Press*, 1979.

[105] E.B. Meier and F. Ade. Tracking multiple objects using the condensation algorithm. *Journal of Robotics and Autonomous Systems*, pages 93–105, 2001.

[106] F.G. Meyer, R.T. Constable, A.J. Sinusas, and J.S. Duncan. Dense non-rigid motion tracking from a sequence of velocity fields. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, page RP1, 1996.

[107] T.B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[108] B. Moghaddam and A. Pentland. Probabilistic visual learning for object detection. In *International Conference on Computer Vision*, pages 786–793, 1995.

[109] F. Mokhtarian and A. K. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):789–805, 1992.

[110] B. North, A. Blake, M Isard, and J. Rittscher. Learning and classification of complex dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):781–796, August 2000.

[111] K. Nummiaro, E. Koller-Meier, and L. Van Gool. Object tracking with an adaptive color-based particle filter. In *Proceedings of the Deutsche Arbeitsgemeinschaft für Mustererkennung*, Zurich, Switzerland, 2002.

[112] B. Olstad and A.H. Torp. Encoding of a-priori information in active contour models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):863–872, September 1996.

[113] C. Papin, P. Bouthemy, E. Memin, and G. Rochard. Tracking and characterization of highly deformable cloud structures. In *Proceedings of the European Conference on Computer Vision*, pages 428–442, 2000.

[114] V. Pavlovic, J.M. Rehg, T.J. Cham, and X. Murphy. A dynamic bayesian network approach to figure tracking using learned dynamic models. In *Proceedings of the International Conference on Computer Vision*, pages 94–101, 1999.

[115] P. Perez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *Proceedings of the European Conference on Computer Vision*, pages 661–675, 2002.

[116] N. Peterfreund. The pdaf based active contour. In *Proceedings of the International Conference on Computer Vision*, pages 227–233, 1999.

[117] N. Peterfreund. Robust tracking of position and velocity with kalman snakes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(6):564–569, June 1999.

[118] V. Philomin, R. Duraiswami, and L.S. Davis. Quasi-random sampling for condensation. In *Proceedings of the European Conference on Computer Vision*, pages 134–149, 2000.

[119] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C - The Art of Scientific Computing*. Cambridge University Press, 2. edition, 1992.

[120] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77(2), pages 257–285, 1989.

[121] C. Rasmussen and G.D. Hager. Probabilistic data association methods for tracking complex visual objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):560–0576, June 2001.

[122] A. Rattarangsi and R. T. Chin. Scale-based detection of corners of planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(4):430–449, 1992.

[123] I.D. Reid and D.W. Murray. Active tracking of foveated feature clusters using affine structure. *International Journal of Computer Vision*, 18(1):41–60, April 1996.

[124] J. Rittscher, J. Kato, S. Joga, and A. Blake. A probabilistic background model for tracking. In *Proceedings of the European Conference on Computer Vision*, 2000.

[125] S. Romdhani, S. Gong, and A. Psarrou. A multi-view nonlinear active shape model using kernel pca. *Proceedings of the British Machine Vision Conference*, 1999.

[126] R. Rosales and S. Sclaroff. 3d trajectory recovery for tracking multiple objects and trajectory-guided recognition of actions. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages II:117–123, 1999.

[127] Y. Rubner, J. Puzicha, C. Tomasi, and J.M. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. *Computer Vision and Image Understanding*, 84(1):25–43, 2001.

[128] G. Sandini and M. Tistarelli. Active tracking strategy for monocular depth inference over multiple frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):13–27, January 1990.

[129] S. Scarloff and L. Liu. Deformable shape detection and description via model-based region grouping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(5):475–489, May 2001.

[130] Stan Scarloff and A.P. Pentland. Modal matching for correspondance and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):545–561, June 1995.

[131] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 2002.

[132] B. Schoelkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.R. Mueller, G. Raetsch, and A.G. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.

[133] D. Schulz, W. Burgard, D. Fox, and A.B. Cremers. Tracking multiple moving objects with a mobile robot. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, 2001.

[134] A. Sethian, editor. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 2. edition, 1999.

[135] J. Sherrah and S. Gong. Tracking discontinuous motion using bayesian inference. In *Proceedings of the European Conference on Computer Vision*, 2000.

[136] Jianbo Shi and Carlo Tomasi. Good features to track. *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages 593–600, June 1994.

[137] H. Sidenbladh, M.J. Black, and D.J. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *Proceedings of the European Conference on Computer Vision*, 2000.

[138] S.M. Smith and J.M. Brady. Asset-2: Real-time motion segmentation and shape tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):814–820, August 1995.

[139] M. Sonka, V. Hlavac, and R. Boyle, editors. *Image Processing, Analysis and Machine Vision*. Brooks/Cole Publishing Company, 2. edition, 1999.

[140] B. Southall, J.A. Marchant, T. Hague, and B.F. Buxton. Model based tracking for navigation and segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 797–811, 1998.

[141] M. Spengler and B. Schiele. Towards robust multi-cue integration for visual tracking. *Machine, Vision and Applications*, 14:50–58, 2003.

[142] L.H. Staib and J.S. Duncan. Boundary finding with parametrically deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1061–1075, 1992.

[143] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, August 2000.

[144] K. Toyama and A. Blake. Probabilistic tracking with exemplars in a metric space. *International Journal of Computer Vision*, 48(1):9–19, 2002.

[145] K. Toyama and G.D. Hager. Incremental focus of attention for robust vision-based tracking. *International Journal of Computer Vision*, 35(1):45–63, November 1999.

[146] J. Triesch and C. von der Malsburg. Self-organized integration of adaptive visual cues for face tracking. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, Los Alamitos, 2000.

[147] H.T. Tsui, Z.Y. Zhang, and S.H. Kong. Feature tracking from an image sequence using geometric invariants. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages 244–249, 1997.

[148] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuro Science*, 3(1):71–86, 1991.

[149] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan, editors. *The unscented particle filter*. Technical Report CUED/F-INFENG/TR 380, Cambridge University Engineering Department, 2000.

[150] J. Vermaak, A. Doucet, and P. Perez. Maintaining multi-modality through mixture tracking. In *Proceedings of the International Conference on Computer Vision*, 2003.

[151] Kasprzak. W. and H. Niemann. Adaptive road recognition and ego-state tracking in the presence of obstacles. *International Journal of Computer Vision*, 28(1):5–26, 1998.

[152] C. Woehler and J. Anlauf. An adaptable time-delay neural-network algorithm for image sequence. 10(6):1531–1536, 1999.

[153] C.R. Wren, A. Azarbayejani, T.J. Darrell, and A.P. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.

[154] Y. Yacoob and L.S. Davis. Tracking rigid motion using a compact-structure constraint. In *Proceedings of the International Conference on Computer Vision*, pages 198–205, 1999.

[155] A.L. Yuille, D.S. Cohen, and P. Hallinan. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8:99–112, 1992.

[156] R. Zaritsky, N. Peterfreund, and N. Shimkin. Velocity-guided tracking of deformable contours in three dimensional space. In *Proceedings of the European Conference on Computer Vision*, 2000.

[157] Q. Zheng and R. Chellappa. Automatic feature point extraction and tracking in image sequences for arbitrary camera motion. *International Journal of Computer Vision*, 15(1-2):31–76, June 1995.

[158] Y. Zhong, A.K. Jain, and M.P. Dubuisson-Jolly. Object tracking using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):544–549, May 2000.

[159] P. Zhu and P. Chirlian. On critical point detection of digital shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):737–748, 1995.