All rights reserved

This work, in whole or in part, may not be copied (in any media), printed, translated, stored in a retrieval system, transmitted via the internet or other electronic means, except for "fair use" of brief quotations for academic instruction, criticism, or research purposes only. Commercial use of this material is completely prohibited.

©

ככל הזכויות שמורות

אין اللهיטיק (במהיה מלאכות), להדפיס, להטרם, ללאחסן בין ממון, להפיץ במעוף, chuüber אוrios 00ג ויריו ג'יקובס
Irwin and Joan Jacobs Graduate School

This work, in whole or in part, may not be copied (in any media), printed, translated, stored in a retrieval system, transmitted via the internet or other electronic means, except for "fair use" of brief quotations for academic instruction, criticism, or research purposes only. Commercial use of this material is completely prohibited.

©

ככל הזכויות שמורות

אין اللهיטיק (במהיה מלאכות), להדפיס, להטרם, ללאחסן בין ממון, להפיץ במעוף, chuüber אוrios 00ג ויריו ג'יקובס
Irwin and Joan Jacobs Graduate School
Visual Tracking in a General Context via Tracker Combination and Low-Level Cues

Ido Leichter
Visual Tracking in a General Context via Tracker Combination and Low-Level Cues

Research Thesis

In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Ido Leichter

Submitted to the Senate of the Technion - Israel Institute of Technology

Tevet 5768, Haifa, December 2007
The research thesis was done under the supervision of Prof. Ehud Rivlin and Prof. Michael Lindenbaum in the Department of Computer Science. The generous financial help of the Technion is gratefully acknowledged.

I would like to thank my supervisors, Prof. Ehud Rivlin and Prof. Micha Lindenbaum, for their encouragement throughout my Ph.D. research, for believing in me and giving me the freedom to pursue the research directions I was interested in within the topic of my thesis. I also thank them for funding my research during its last stages.

My time at the Technion would have been socially much less enjoyable without my student colleagues: Gregory Begelman, Gilad Ben-Avi, Gal Lavee, Ronen Lerner, Michael Pechuk, Roman Sandler, Octavian Soldea, Roman Talyansky, and others.
# Contents

Abstract

## 1 Introduction

1.1 State spaces .................................................................................................................. 3

1.1.1 State spaces for tracking an object’s location ......................................................... 3

1.1.2 State spaces for tracking an object’s location and scale in the image plane .......... 4

1.1.3 State spaces for tracking rigid objects ...................................................................... 4

1.1.4 State spaces for tracking articulated objects ......................................................... 5

1.1.5 State spaces for tracking curves enclosing objects’ silhouettes ............................. 5

1.1.6 Exemplar-based state spaces .................................................................................... 7

1.2 Dynamic models ............................................................................................................ 8

1.2.1 Zero-order temporal models .................................................................................... 9

1.2.2 Higher order temporal models .............................................................................. 10

1.2.3 General second-order linear difference equation ................................................... 11

1.2.4 Dynamic model in an exemplar-based tracker ...................................................... 11

1.2.5 Dynamic models for tracking multiple objects ..................................................... 11

1.2.6 Hybrid dynamic models ........................................................................................ 12

1.3 Appearance models and measurements ..................................................................... 12

1.3.1 2D image templates .............................................................................................. 12

1.3.2 Contour templates ................................................................................................ 13

1.3.3 Histogram templates ............................................................................................. 13

1.3.4 3D description-based appearance models ............................................................ 14

1.3.5 Appearance models based on template-spaces ..................................................... 14

1.3.6 On-line adapting appearance models ................................................................... 14

1.4 Filtering .......................................................................................................................... 15

1.4.1 The Kalman filter ................................................................................................ 15

1.4.2 The CONDENSATION filter ............................................................................... 16

## 2 Tracker Combination

2.1 Introduction .................................................................................................................... 21

2.2 Context .......................................................................................................................... 23

2.3 Combining explicit PDF-yielding trackers .................................................................. 24

2.3.1 Same state space .................................................................................................... 24
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.2 Different state spaces</td>
<td>27</td>
</tr>
<tr>
<td>2.4 Combining CONDENSATION-based trackers</td>
<td>29</td>
</tr>
<tr>
<td>2.4.1 Same state space</td>
<td>29</td>
</tr>
<tr>
<td>2.4.2 Different state spaces</td>
<td>30</td>
</tr>
<tr>
<td>2.5 Combining explicit PDF-yielding with CONDENSATION-based trackers</td>
<td>32</td>
</tr>
<tr>
<td>2.6 Combining trackers of different, related objects</td>
<td>33</td>
</tr>
<tr>
<td>2.7 Experiments</td>
<td>34</td>
</tr>
<tr>
<td>2.7.1 Experiment I: common object, explicit PDF-yielding trackers,</td>
<td></td>
</tr>
<tr>
<td>different state spaces</td>
<td>34</td>
</tr>
<tr>
<td>2.7.2 Experiment II: common object, CONDENSATION-based trackers,</td>
<td>37</td>
</tr>
<tr>
<td>common state space</td>
<td></td>
</tr>
<tr>
<td>2.7.3 Experiment III: common object, CONDENSATION-based trackers,</td>
<td>38</td>
</tr>
<tr>
<td>different state spaces</td>
<td></td>
</tr>
<tr>
<td>2.7.4 Experiments IV and V: separate objects</td>
<td>40</td>
</tr>
<tr>
<td>2.7.5 Experiment VI: separate objects, exemplar-based trackers</td>
<td>42</td>
</tr>
<tr>
<td>2.8 Conclusion and discussion</td>
<td>44</td>
</tr>
<tr>
<td>3 Tracking Object Bitmaps under Very General Conditions</td>
<td>46</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>46</td>
</tr>
<tr>
<td>3.2 Previous work</td>
<td>48</td>
</tr>
<tr>
<td>3.3 The Bittracker algorithm</td>
<td>50</td>
</tr>
<tr>
<td>3.3.1 Overview</td>
<td>50</td>
</tr>
<tr>
<td>3.3.2 The bitmap’s PDF</td>
<td>51</td>
</tr>
<tr>
<td>3.3.3 MAP bitmap estimation</td>
<td>57</td>
</tr>
<tr>
<td>3.3.4 Considering only target-potential pixels</td>
<td>59</td>
</tr>
<tr>
<td>3.3.5 Parameters and algorithm outline</td>
<td>60</td>
</tr>
<tr>
<td>3.4 Experiments</td>
<td>61</td>
</tr>
<tr>
<td>3.5 Conclusion</td>
<td>74</td>
</tr>
<tr>
<td>4 Tracking by Affine Kernel Transformations Using Color and Boundary</td>
<td>75</td>
</tr>
<tr>
<td>Cues</td>
<td></td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>75</td>
</tr>
<tr>
<td>4.2 Previous work</td>
<td>76</td>
</tr>
<tr>
<td>4.3 Affine kernel fitting</td>
<td>77</td>
</tr>
<tr>
<td>4.3.1 Color-related score component</td>
<td>78</td>
</tr>
<tr>
<td>4.3.2 Boundary-related score component</td>
<td>81</td>
</tr>
<tr>
<td>4.3.3 Score maximization</td>
<td>83</td>
</tr>
<tr>
<td>4.4 Experimental results</td>
<td>84</td>
</tr>
<tr>
<td>4.5 Conclusion</td>
<td>88</td>
</tr>
<tr>
<td>5 Mean Shift Tracking with Multiple Reference Color Histograms</td>
<td>90</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>90</td>
</tr>
<tr>
<td>5.2 The Mean Shift tracker</td>
<td>91</td>
</tr>
<tr>
<td>5.2.1 Target localization via Mean Shift</td>
<td>93</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Tracking the 2D location of a flag in the image plane. Taken from [22].
1.2 Augmenting the state space of 2D location with isotropic scaling. Taken from [26].
1.3 The experiment in [56]. Every fifth frame is shown.
1.4 The cube-tracking experiment in [4].
1.5 An illustration of the state space used to track the human body in [69].
1.6 An illustration of the state space used to track the human body in [32].
1.7 Tracking a leaf using the space of affine transformations of a hand-drawn template curve. Taken from [53].
1.8 Tracking a hand in the space of linear combinations of six hand-drawn template curves augmented with translation parameters. Taken from [53].
1.9 A subset of the exemplars used in a tracking experiment of [117].
1.10 Exemplar-based tracking of a person. Taken from [117].
1.11 The thick line is a hypothesized curve. The spines are curve normals along which high-contrast features are sought. Taken from [53].
1.12 A reference color histogram $q^*$ is gathered at time $t_0$. At time $t$ and for a hypothesized state $x$, the candidate color histogram $q_t(x)$ is gathered within the region $R(x)$. Taken from [96].
1.13 Before smoothing, multiple hypotheses can increase the variance of the state distribution (left) and shift the mean away from the object position (middle). After running the smoothing algorithm the estimated variance has dropped and the mean is now correctly positioned (right). The solid black line is the distribution mean, and the dotted white lines are high-scoring samples, where the width of the sample outline is proportional to its sample weight. Taken from [52].

2.1 Two separate, explicit PDF-yielding trackers. Both trackers track in the same state space.
2.2 Combining $T_1$ and $T_2$ of Figure 2.1: two explicit PDF-yielding trackers that use a common state space.
2.3 Two separate, explicit PDF-yielding trackers whose state spaces differ.
2.4 Combining $T_1$ and $T_2$ of Figure 2.3: two explicit PDF-yielding trackers using different state spaces.
2.5 Two separate, CONDENSATION-based trackers. Both trackers track in the same state space.
Combining $C_1$ and $C_2$ of Figure 2.5: two CONDENSATION-based trackers that use a common state space. 31

Combining two CONDENSATION-based trackers that use different state spaces. 32

A few frames of the sequence used in Experiment I, with the tracking results of the two separate trackers imposed (marked by boxes and dots). The template-based tracker failed already in the beginning due to an occlusion by a tree. Afterwards, the background subtraction-based tracker failed due to the proximity of the moving car to the tracked person. 35

Results of Experiment I using the composite tracker. The tracking succeeded until the person left the scene. 36

A few frames of the sequence used in Experiment II, with the tracking results of the edge-based tracker imposed (a representative set of samples from the sample-set is shown). Due to camouflage provided by another ball, a second, false hypothesis was generated, shunting aside the true hypothesis. 37

Results of Experiment II using the composite tracker. The tracker now overcame the camouflage provided by the other ball, as well as an occlusion caused by a third ball. 38

A few frames of the sequence used in Experiment III, with the tracking results of the edge-based tracker imposed. Due to many edges in the interior of the tracked object region, the tracker yielded poor results. 39

A few frames of the tracking results obtained in Experiment III using the reference colors-based tracker. The mean of the sample-set is shown. 39

Results of Experiment III using the composite tracker. The composite tracker overcame the distractions caused by the edges inside the tracked head’s region. 40

A few frames of the sequence used in Experiment IV, with the tracking results of the two separate trackers imposed (marked by dots). After the person rotated his head to his right, the right eye became occluded, causing an irrecoverable failure. 41

Results of Experiment IV using the composite tracker. The tracker now recovered from failures caused by the head rotations. 42

Results of Experiment VI using the two trackers separately. The two upper images in each frame are original test frames, the two middle images are the test frames fed to the trackers, and the two bottom images are the exemplars approximated by the trackers as most probable. (The exemplars were taken from another sequence.) The tracking succeeded during the undisturbed time periods, but failed whenever the image of the right eye was replaced by noise. 43
2.18 Results of Experiment VI after combining the two trackers. The two upper images in each frame are the original test frames, the two middle images are the test frames fed to the trackers, and the two bottom images are the exemplars approximated by the trackers as most probable. (The exemplars were taken from another sequence.) Now the tracking also succeeded when the image of the right-side eye was replaced by noise.

3.1 (a) and (b) constitute a consecutive pair of images in a video of a random-dot object moving in front of a random-dot background that is moving as well. The object in the two images is shown in (c) and (d).

3.2 The reduced \( I_t \) and the reduced \( I_{t-1} \). In practice, the bitmap is estimated in the reduced image and set to 0 outside of it.

3.3 Tracking in the Random-dot sequence. The estimated bitmap is shown in green, overlaid on top of intensity version images containing only the target.

3.4 Tracking results for the Random-dot sequence obtained by the mean shift tracker in [27]. The estimated location is shown on top of intensity version images containing only the target. The tracking failed quite early in the sequence.

3.5 (a) and (b) are two (nonconsecutive) images from a video of a randomly segmented object of gradually time-varying shape, segmentation and colors, moving in front of a randomly segmented background of gradually time-varying colors that is in motion as well. The object as it appears in the two images is shown in (c) and (d), respectively.

3.6 Tracking in the Random-segment sequence. The estimated bitmap is shown in green, overlaid on top of intensity version images containing only the target.

3.7 Tracking results for the Random-segment sequence obtained by the mean shift tracker in [27]. The estimated location is shown on top of intensity version images containing only the target. The tracking failed at the beginning of the sequence.

3.8 Tracking a rotating reel of cellotape filmed by a moving camera. The hole in the reel, which was not revealed at the beginning of the video, was revealed and marked correctly as the video progressed.

3.9 Tracking a man walking in a mall filmed by a moving camera. Note that Bittracker overcomes the zoom-in and zoom-out near the end of the sequence, as well as the partial occlusion at the end.

3.10 Tracking a woman walking in a mall as filmed by a moving camera. The tracking algorithm overcame lighting changes and long-term partial occlusions. Since the woman and the girl she takes by hand were adjacent and walking at similar velocity over an extended time period (beginning around frame #100), the girl and the woman were joined as the tracking proceeded.
3.11 Tracking results for the Woman-and-Child sequence obtained by the mean shift tracker in [27].

3.12 Tracking a cow in a running herd filmed by a moving camera. Although the tracked object underwent a severe partial occlusion, the tracking continued.

3.13 Tracking a lighter undergoing general motion and severe partial occlusions as filmed by a moving camera.

3.14 Tracking results for the Lighter sequence obtained by the mean shift tracker.

3.15 Tracking a rolling ball filmed by a moving camera. Note the occlusion caused by the toy.

3.16 BitTracker’s tracking results for the Camouflage sequence, constructed in [73] to show the limits of the tracking algorithm proposed there.

3.17 Tracking results for the Camouflage sequence obtained by the mean shift tracker. The images are ordered from left to right. The tracking succeeds as long as the target’s texture is different from nearby background, but as the target passes over a background region of similar texture, the tracker locks on a wrong image area.

3.18 Tracking results for the Camouflage sequence reported in [73]. (The images are taken from [73].)

4.1 The color-related normalized kernel \((d = 0.2)\).

4.2 The normalized kernels used for weighting the pixels in the color histograms of the target (a) and the background (b).

4.3 The boundary-related normalized kernel.

4.4 Results of the proposed tracker for Sequence I.

4.5 Results of the mean shift tracker [27] for Sequence I.

4.6 Results of the proposed tracker without updating the target’s color PDF for a sub-sequence of Sequence I. Since the target colors that are revealed to the camera change with time, the tracking fails. (Only small, magnified portions of the frames are shown so that the target colors will be clearly seen.)

4.7 Results of the proposed tracker for Sequence II.

4.8 Results of the mean shift tracker [27] for Sequence II.

4.9 Results of the proposed tracker for Sequence III.

4.10 Results of the proposed tracker for Sequence IV.

4.11 Results obtained using the proposed tracker without its boundary-related kernel \((\alpha = 0)\). It is evident that color edges play a crucial role in the proposed tracker.

5.1 Results of the Mean Shift tracker for Sequence I, where the reference color histogram was set in the first frame.
5.2 The two target views used for approximating the two reference histograms (a) \( \hat{q}_1 \) and (b) \( \hat{q}_2 \). These two reference histograms were used to set the final reference histogram \( q \) according to (5.9). The latter histogram is illustrated in (c) by an image with an identical color histogram.

5.3 Results of the Mean Shift tracker for Sequence I, where the reference color histogram was set using the two target views shown in Fig. 5.2.

5.4 Tracking results for Sequence I using the convex hull-based target model. The reference color histograms were set using the two target views shown in Fig. 5.2.

5.5 The two target views used for approximating the two reference histograms (a) \( \hat{q}_1 \) and (b) \( \hat{q}_2 \) for the convex hull-based target model in Sequence II.

5.6 Tracking results for Sequence II using the convex hull-based target model. The reference color histograms were set using the two target views shown in Fig. 5.5.

5.7 The four target views used for approximating the four reference histograms (a) \( \hat{q}_1 \), (b) \( \hat{q}_2 \), (c) \( \hat{q}_3 \) and (d) \( \hat{q}_4 \) for the convex hull-based target model in Sequence III.

5.8 Tracking results for Sequence III using the convex hull-based target model. The reference color histograms were set using the four target views shown in Fig. 5.7.

5.9 The two target views used for approximating the two reference histograms (a) \( \hat{q}_1 \) and (b) \( \hat{q}_2 \) for the convex hull-based target model in Sequence IV.

5.10 Tracking results for Sequence IV using the convex hull-based target model. The reference color histograms were set using the two target views shown in Fig. 5.9.

C.1 (b) The histogram of the \( f_{\Delta s} \) interval sizes computed for the object marked in (a) (the largest interval out of the four is taken per pixel). We see that a large portion of the \( f_{\Delta s} \) have small intervals and thus affect the bitmaps PDF.
Abstract

This research thesis addresses the problem of visual tracking in video in a general context using two approaches. The first approach consists of the combination of multiple trackers that use different features and thus have different failure modes. A general framework for combining visual trackers that propagate filtering distributions over time is proposed. The individual trackers may propagate the filtering distributions either explicitly, for example, via Kalman filtering, or by using sample-sets of the distributions, via particle filtering. The proposed framework enables the combination of trackers of different state spaces, and in many cases it allows treating the individual trackers nearly as “black boxes.” Another benefit of the proposed framework is that it may be applied as is to the combination of trackers that track different, albeit related, targets. The suggested framework was successfully tested using various state spaces and datasets.

The second approach consists of the employment of basic, low-level visual characteristics, which are typically valid. A tracker of object bitmaps (silhouettes) is proposed without using any prior information about the target or the scene. The low-level visual characteristics employed are (short-term) Constancy of Color - the color projected to the camera from a point on a surface is approximately similar in consecutive video frames; Spatial Motion Continuity - the optical flow of the vast majority of the pixels in an image region corresponding to an object is spatially continuous; Spatial Color Coherence - it is highly probable that adjacent pixels of similar color belong to the same object. The tracker relies only on these basic visual characteristics, which makes it applicable in a very general context. The proposed tracker works by approximating, in each video frame, a probability distribution function of the target’s bitmap and then estimating the maximum a posteriori bitmap. Many experiments were conducted to demonstrate that the tracker is able to track objects that undergo drastic appearance changes and that are filmed using an arbitrarily moving camera.

The usage of kernel-based trackers may also be included in the second approach. Two kernel-based trackers are proposed. The first tracker exploits the constancy of color and the presence of color edges along the target boundary. This tracker uses these two visual cues to affinely transform the kernel over time. In a sense, this work extends previous kernel-based trackers by incorporating the object boundary cue into the tracking process and by allowing the kernels to be affinely transformed instead of only translated and isotropically scaled. These two extensions make for more precise target localization. Moreover, a more accurately localized target facilitates safer updating of its reference color model, further enhancing the tracker’s robustness. The second kernel-based tracker enhances the Mean Shift tracker to use multiple color histograms obtained from different target views. This enhancement makes the Mean Shift tracker suitable for tracking objects whose colors that are revealed to the camera change over time. Both these kernel-based trackers were experimentally validated to cope with tracking scenarios for which traditional kernel-based trackers fail.
Chapter 1

Introduction

The Computer Vision research area is concerned with the extraction of information about the content of a scene from one or more images. The extracted information may, for example, be the three-dimensional structure of the scene or objects contained in it (e.g., stereo: three-dimensional descriptions from two or more views), the identity of the objects (e.g., face recognition), the class the imaged objects are classified into (e.g., character recognition) or the the class of the image itself (e.g., deciding whether an input image contains an urban or a natural scene), a meaningful partitioning of the image into image regions (segmentation), and the locations of specific kinds of targets in the image (e.g., face detection).

Often, the source of information is a video, i.e., an image time-sequence. Therefore, automated tracking of objects through an image sequence is essential to image analysis and is a fundamental problem in computer vision. In addition, there are a number of problem domains where a solution to the tracking task is desirable, such as traffic and security monitoring and video databases.

Given an image sequence, tracking an object consists of estimating its state in every frame in the sequence. The object’s sought state may be of different complexities, ranging from very simple (e.g., a 2D location in the image plane) to more complex (e.g., a parametric contour such as a B-Spline enclosing the object’s silhouette in the image plane, or the positions of the different parts composing an articulated object). However, in contrast to a sole detection task which is applied on each frame independently of the other frames, the task of tracking incorporates a dynamic model (referred also as stochastic dynamics or motion model) which relates the state between two or more successive frames. The integration of the dynamic model has two advantages over a sole detection algorithm. First, the amount of computations is reduced due to the reduced search space. Secondly, the states at adjacent frames bear additional information on the current state. This information is taken into account by the dynamic model, resulting in higher robustness.

A visual tracking algorithm is generally composed of the following components: a state space of the tracked state, a dynamic model of the tracked state which couples the states between (usually successive) frames, an appearance model of the target along with appropriate measurements for seeking this appearance in the images, and a filter-
ing process which combines all the above to propagate the state estimate through the image sequence. In the following we describe and discuss each of these components separately.

1.1 State spaces

Tracking an object may be performed in various degrees of detail and while using different types of states. The particular state space is chosen according to the needed degree of detail in one hand and the allowed computational complexity on the other hand – two aspects between which exists a tradeoff. Another factor that determines the particular state space used is whether the object is to be tracked in the 3D space or in the image plane. The type of the tracked object also has a crucial affect on the state space used: a detailed tracking of an articulated or deformable object may require a state space of higher dimension than that of a rigid object. Also affecting the state space is the degree of detail by which the object’s shape is known: consider the case of tracking a rigid object’s silhouette in the image plane. If the object’s shape is known, the object’s silhouette may be determined from its location and pose by projecting the object onto the image (assuming a calibrated camera), thus requiring a state space of six dimensions. However, if the object’s shape is unknown, a state space of much higher dimension might be required to accurately describe the object’s silhouette. An example of an opposite situation may be the tracking of humans only by their center of mass (e.g., in surveillance applications). In this case only a 2D or 3D location is tracked, although the human is an articulated object that requires a high-dimensional space for an accurate description.

Following we give the commonly used state spaces.

1.1.1 State spaces for tracking an object’s location

Probably the simplest type of state space used in tracking applications is the one constituting the mere object’s location. For example, the object’s location may be its center of mass in the image plane (e.g., [22] – see Figure 1.1), or its 2D location on the floor (part of the state used in [54]). The image region considered to contain the tracked object is many times referred to as a region of interest (ROI). When tracking only the object’s location in the image plane, the ROI is usually an ellipse or a rectangle of predetermined shape.

![Figure 1.1. Tracking the 2D location of a flag in the image plane. Taken from [22].](image-url)
1.1.2 State spaces for tracking an object’s location and scale in the image plane

The tracking of an object’s location in the image plane as in the previous case may be augmented by tracking its scale. The space may be augmented by a single dimension corresponding to isotropic scaling (e.g., [23, 26] – see Figure 1.2), or with a different scale in each axis. A rectangular ROI enclosing the object tightly, it termed as a **bounding box** (e.g., [74]). Sometimes, the space is augmented with another degree of freedom for rotation in the image plane (e.g., [56] – see Figure 1.3).

1.1.3 State spaces for tracking rigid objects

When a rigid object is tracked in space, its state is fully determined by its 3D location (three degrees of freedom) and pose (3D rotation). Thus, a state space of six dimensions
is used. [4] is an example for such a tracking application (see Figure 1.4).

![Figure 1.4. The cube-tracking experiment in [4]](image)

There are also cases of hybrid 2D-3D state spaces. For example, the 2D location of the object in the image plane may be tracked along with the rotation about one axis. For example, in [80, 124] people’s heads are tracked by location and scale in the image plane, along with the degree of horizontal rotation.

### 1.1.4 State spaces for tracking articulated objects

When the tracked object is articulated (e.g., [32, 69, 84]), i.e., composed of multiple links (segments) connected by means of joints, it may be desirable to track its detailed configuration. One common example is tracking of the human body. In [69] the human body is modelled as an object composed of 10 joints and 14 segments with a total of 25 degrees of freedom (see illustration in Figure 1.5). This 25-dimensional space was augmented with three dimensions for global translation, three dimensions for global rotation, and global scale. In [32] the human body is based on a kinematic chain consisting of 17 segments. Six degrees of freedom are given to base translation and rotation. The shoulder and hip joints are treated as sockets with three degrees of freedom, the clavicle joints are given two degrees of freedom (they are not allowed to rotate about their own axis) and the remaining joints are modeled as hinges requiring only one. This results in a model with 29 degrees of freedom. See Figure 1.6.

### 1.1.5 State spaces for tracking curves enclosing objects’ silhouettes

A physical object may be projected into an arbitrary region in an image plane, and many times we wish to track the object in the image plane. These cases may arise when the tracked object is deformable (e.g., a fish), when it is articulated but we want to avoid its physical modelling or we need only its approximated silhouette in the image plane, or even when it is rigid but of unknown shape.

The simplest case of a state space of this kind is the two-dimensional space of translations of a curve of a predetermined shape. Such a space is used in [81], where a vertical ellipse of fixed size is used to track people’s heads.

A more general kind of state spaces are those of parametric geometric transformations of a template curve. For example, in [53] a dancing girl’s head and a leaf were
Figure 1.5. An illustration of the state space used to track the human body in [69].

Figure 1.6. An illustration of the state space used to track the human body in [32].
Figure 1.7. Tracking a leaf using the space of affine transformations of a hand-drawn template curve. Taken from [53].

tracked using the six-dimensional state space of affine geometric transformations of a hand-drawn template curve. See illustration in Figure 1.7.

Another kind of state space used to track curves was originally suggested in [12], where a time-dependent parameterized image curve \( r(s, t) \) was tracked. The parametrization was in terms of B-splines, so \( r(s, t) = (B(s) \cdot Q_{x}(t), B(s) \cdot Q_{y}(t)) \) for \( 0 \leq s \leq L \), where \( B(s) \) is a vector \((B_{1}(s), \ldots, B_{N_{B}}(s))^{T}\) of B-spline basis functions, \( Q_{x} \) and \( Q_{y} \) are vectors of B-spline control point coordinates, and \( L \) is the number of spans. See Figure 1.11 for an illustration.

Image curves of articulated objects may be modeled also as linear combinations of \( n \) template curves, forming an \((n-1)\)-dimensional space. For example, in [53] a hand that was translating, rotating and flexing its fingers was tracked using a shape space constructed from six templates drawn around the hand in a fixed orientation and with the fingers and thumb in varying configurations. The six templates were combined linearly to form a five-dimensional space of deformations which were then added to the space of translations to form a seven-dimensional shape space. See Figure 1.8.

1.1.6 Exemplar-based state spaces

In [117] are introduced exemplar-based state spaces. A space of this kind is formed as a union of shape spaces, each constitutes the transformation space of a particular exemplar. Denoting the set of exemplars by \( \chi = \{\bar{x}_{k}, k = 1, \ldots, K\} \) and the transformation parameterized by the parameters \( \alpha \in A \) (\( A \) is the transformation space) by \( T_{\alpha} \), then a particular state is a pair \((\alpha, k)\) standing for the imaged object \( T_{\alpha} \bar{x}_{k} \).

One example of such a state space, given in [117], is the one formed by a set of exemplar image curves undergoing similarity transformations \((\alpha = (u, \theta, s), \) where \( u \) denotes the two-dimensional translation parameters, \( \theta \) denotes the rotation parameter, and \( s \) denotes the scale parameter). An experiment of tracking the curve enclosing
the silhouettes of walking people using a set of 30 exemplars without any transformations was performed in [117]. Figure 1.9 shows a subset of the used exemplars, and Figure 1.10 illustrates the tracking.

1.2 Dynamic models

As mentioned, the dynamic model is the component enabling to perform the tracking as an inter-frame process in which the states of different (usually consecutive) frames are related. The dynamic model models the probability distribution function (PDF) of the tracked state conditioned on the state in the previous frame. (Generally, the PDF may be conditioned on multiple previous frames.) This modeling of the relation between states of consecutive frames may be exploited to reduce the search space and enhance the tracking robustness.

A most general dynamic model models the state PDF at time \( t, x_t \), conditioned on all the states at all previous times. Equivalently, the dynamic model relates \( x_t \) to the previous states through a function \( f \) which is dependent also on a random, time-varying
noise $w_t$:

$$x_t = f(x_{t-1}, x_{t-2}, \ldots, w_t).$$

(1.1)

Sometimes the dynamic model is preset in an ad-hoc manner, whereas in other times its parameters may be learned from a training sequence. Following we give common dynamic models used.

### 1.2.1 Zero-order temporal models

Probably the simplest dynamic model is the one approximating the state PDF in the next time step as a Gaussian PDF (or some other PDF, say, uniform) with the previous state as the mean. For example, in [124] people’s heads are tracked in the space composed of translations in the image plane, scale, and horizontal pose. The dynamic model was set to

$$p(x_t) = N(x_{t-1}, \sigma_x^2)$$
$$p(y_t) = N(y_{t-1}, \sigma_y^2)$$
$$p(\theta_t) = N(\theta_{t-1}, \sigma_\theta^2)$$
$$s_t = s_{t-1} \cdot 1.2^k$$

with $k \sim N(0, \sigma_s^2)$

where $(x_t, y_t)$ is the translation, $\theta_t$ is the rotation and $s$ is the scale expressed in resolution. The standard deviations were set heuristically, but could have been learnt given a training sequence. This kind of model was also used in [54] for tracking people using a state of higher dimensionality.
1.2.2 Higher order temporal models

In addition to using the state at the previous time to predict the following state (as is performed in a zero-order temporal model), in an \( n \)-order temporal model the first \( n \) time derivatives of the state are used as well for the prediction, based on the Taylor expansion. The state PDF is set to be a Gaussian with this state prediction as its mean.

A temporal model of order higher than zero may be implemented either by augmenting the state vector with the parameters’ time-derivatives, or by conditioning the predicted state PDF on preceding states. For example, denoting the state vector by \( \mathbf{x}_t \), the first order prediction is (using Taylor expansion)
\[
\mathbf{x}_{t}^- = \mathbf{x}_{t-1} + (\Delta t) \dot{\mathbf{x}}_{t-1},
\]
where \( \Delta t \) is the time between two consecutive frames. Thus, augmenting the state vector with its derivative yields
\[
[\mathbf{x}_{t}^-; \dot{\mathbf{x}}_{t-1}] = \begin{bmatrix} 1 & (\Delta t) \ I \\ 0 & \ I \end{bmatrix} [\mathbf{x}_{t-1}; \dot{\mathbf{x}}_{t-1}].
\]

The corresponding PDF is
\[
p \left( \begin{bmatrix} \mathbf{x}_t \\ \dot{\mathbf{x}}_t \end{bmatrix} \middle| \begin{bmatrix} \mathbf{x}_{t-1} \\ \dot{\mathbf{x}}_{t-1} \end{bmatrix} \right) = \mathcal{N} \left( \begin{bmatrix} \mathbf{x}_{t}^-; \dot{\mathbf{x}}_{t-1} \end{bmatrix}, C \right),
\]
where the mean is as in (1.6) and \( C \) is the covariance matrix. Alternatively, \( \dot{\mathbf{x}}_{t-1} \) may be approximated as \( \frac{\mathbf{x}_{t-1} - \mathbf{x}_{t-2}}{\Delta t} \), yielding the state prediction
\[
\mathbf{x}_{t}^- = \mathbf{x}_{t-1} + (\Delta t) \dot{\mathbf{x}}_{t-1} = \mathbf{x}_{t-1} + (\Delta t) \frac{\mathbf{x}_{t-1} - \mathbf{x}_{t-2}}{\Delta t} = 2\mathbf{x}_{t-1} - \mathbf{x}_{t-2}.
\]
Adding normal noise to the state prediction yields the PDF
\[
p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}) = \mathcal{N}(2\mathbf{x}_{t-1} - \mathbf{x}_{t-2}, C).
\]

In a similar way, the second temporal model is
\[
[\mathbf{x}_{t}^-; \ddot{\mathbf{x}}_{t-1}] = \begin{bmatrix} 1 & (\Delta t) \ I & 0.5(\Delta t)^2 \ I \\ 0 & \ I & (\Delta t) \ I \\ 0 & 0 & \ I \end{bmatrix} [\mathbf{x}_{t-1}; \dot{\mathbf{x}}_{t-1}; \ddot{\mathbf{x}}_{t-1}],
\]
and the PDF obtained by adding normal noise is
\[
p \left( \begin{bmatrix} \mathbf{x}_t \\ \dot{\mathbf{x}}_t \\ \ddot{\mathbf{x}}_t \end{bmatrix} \middle| \begin{bmatrix} \mathbf{x}_{t-1} \\ \dot{\mathbf{x}}_{t-1} \\ \ddot{\mathbf{x}}_{t-1} \end{bmatrix} \right) = \mathcal{N} \left( \begin{bmatrix} \mathbf{x}_{t}^-; \dot{\mathbf{x}}_{t-1}; \ddot{\mathbf{x}}_{t-1} \end{bmatrix}, C \right),
\]
or alternatively,
\[
p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \mathbf{x}_{t-3}) = \mathcal{N}(2.5\mathbf{x}_{t-1} - 2\mathbf{x}_{t-2} + 0.5\mathbf{x}_{t-3}, C).
\]

For example, in [87] this motion model is used to track 2D locations of people in the image.
1.2.3 General second-order linear difference equation

A more object-adjusted dynamic model may be achieved by learning the parameters of a second order linear difference equation. Denoting the object state by $x_t$ and its mean by $\bar{x}$, the difference equation is

$$x_{t+2} - \bar{x} = A_0(x_t - \bar{x}) + A_1(x_{t+1} - \bar{x}) + B w_t,$$

where $w_t$ are vectors of independent normal random variables. Note that this is a generalization of the previous first order (constant velocity) motion model. The matrices $A_0$, $A_1$ and $B$ may be learned from a labeled training sequence by setting them to the maximum-likelihood ones. It can be shown that the likelihood maximization with respect to the matrices of the deterministic part ($A_0$ and $A_1$) may be performed independently of $B$ [13]. Thus, $A_0$ and $A_1$ may be found first as the least-squares solution, and then $B$ may be calculated as the covariance matrix of the errors.

Such a model is especially suitable for tracking motions containing oscillations such as in [53] or other well-behaved motions such as a walking person [117].

1.2.4 Dynamic model in an exemplar-based tracker

As described in Section 1.1.6, the state vector contains an index parameter which indicates a specific exemplar. Thus, the dynamic model has to account also for this index parameter. In [117] the dynamics of this parameter are modeled as a first-order Markov chain and independently of the transformation parameters. Thus, the dynamic model accounts for the index parameter using a $K \times K$ matrix of transition probabilities ($K$ denotes the number of exemplars). Entry $(i, j)$ in the matrix contains the probability of transforming from exemplar $i$ into exemplar $j$. These probabilities may be learned from the transitions histogram in the training sequence. It is known that normalized histograms are the maximum-likelihood PDF for the transitions.

1.2.5 Dynamic models for tracking multiple objects

There are tracking algorithms suited for tracking multiple objects (e.g., [70, 72, 75]). When the number of objects tracked is fixed, the composed state vector may simply be a concatenation of the single-object state vectors. In [72], where two objects are tracked simultaneously, the concatenated state vector is augmented by an additional binary parameter determining which of the two objects is closer to the camera. When the objects do not overlap in the image, this parameter may be flipped with some predetermined small probability.

However, for a time-varying number of objects the situation is more complicated. A simple way of handling a time-varying number of targets is performed in [124], where faces are detected and tracked in an image sequence: elimination of tracked faces was performed when some confidence measure was too low, and once in every few frames a face detection process was executed to detect appearing faces.
A more rigorous handling of a time-varying number of targets is done in the people tracker of [54]. There, the general state space is augmented by another parameter indicating the number of tracked objects. The general state PDF conditioned on the previous general state eliminates each person with a predetermined probability, and increments the number of people with some other predetermined probability. In the later case, the new person’s state is distributed according the steady-state distribution of a single person’s state.

1.2.6 Hybrid dynamic models

A dynamic model switching between different modes may also be accommodated (e.g., [51, 82, 116]). Such a dynamic model is appropriate when tracking objects that change their behavior significantly over time, like a ball rolling around and bumping into walls (one dynamic model is suitable at times the ball is rolling without interruptions, and another dynamic model when the ball hits a wall), or a hand drawing a picture of different textures (each texture is drawn under a specific type of dynamic model).

In [51] the general motion model consists of $N_S$ different case-specific dynamic models, between which it switches. This is accomplished by augmenting the state vector $x_t$ by another parameter $y_t \in \{1, \ldots, N_S\}$, indicating to which case-specific dynamic model the object obeys. The general object dynamics $p(x_t, y_t|x_{t-1}, y_{t-1})$ are decomposed into $T_{ij}$ – the probability that the state $x$ obeying dynamic model $i$ will be transformed into a state obeying dynamic model $j$, and $p_{ij}(x_t|x_{t-1})$ – the dynamics of the state $x$ given it is transformed from being a state obeying dynamic model $i$ into obeying dynamic model $j$.

1.3 Appearance models and measurements

The appearance model models how the tracked objects should appear in the images, or more generally, the implications on the image that arise due to a specific state of the target. Based on the appearance model, the tracked object in the image may be separated from the rest of the image. By performing appropriate measurements (or observations) in the image (or images when multiple cameras are used, e.g. [20, 21, 62]), the appearance model is used to give indication about the true target state, or assessing hypothesized states. That is, if the appearance in the image corresponding to some hypothesized state agrees with the appearance model, that hypothesized state is reinforced, and vice versa.

In the following we go through different types of appearance models with their corresponding measurements.

1.3.1 2D image templates

One of the simplest and most limited appearance model is the template-based one. In this appearance model, the target appearance in the image is modeled as a given,
constant template. Note that this appearance model is suitable only for rigid objects with very limited types of motion in space (the 3D motions whose projections in the image are well-approximated as translations in the image plane), or with approximately similar appearance from all different views (e.g., a Lambertian ball of homogeneous color). Note that this kind of model does not handle any other difficulties such as changing lightning conditions (temporally and spatially). The measurement used for this appearance model is simply some kind of distance from the template (e.g., SSD - Sum of Square Differences or L1 difference) or a measure of similarity to the template (e.g., correlation) of a candidate image region. This kind of appearance model was used, for example, in the surveillance system of [24], in [89] and in the exemplar-based mouth tracker of [117]. Linear intensity changes may be accounted for by using normalized correlation (with mean subtraction) instead of regular correlation.

This appearance model may be generalized to geometric transformations more general than mere 2D translations. For example, the state space may be extended to image plane rotations, similarity transformations and even planar affine transformations.

1.3.2 Contour templates

In contrast to the image template, the contour-based appearance model models only the 2D enclosing contour of the object and not the content inside it. There may be various types of measurements when using this appearance model. When the background is known, background subtraction may be performed to detect the pixels constituting the foreground object: A pixel of large difference (in absolute value) probably belongs to the foreground object and a pixel of low difference probably does not. Then, a contour enclosing the detected foreground pixels may be approximated. This kind of measurement was used in [24]. Another kind of measurement is spatial gradients. The assumption made when using this measurement is that the spatial gradients along the perimeter of the foreground object are high in magnitude. This kind of measurement is very popular in contour tracking [13, 18, 53]. See Section 1.4.2 for an implementation.

1.3.3 Histogram templates

A less strict version of the image template is the histogram template. This appearance model models only the color distribution of the imaged object. A benefit of this is robustness with respect to certain geometric distortions (e.g., rotation and scaling), but this benefit is gained in expense of invariance of the appearance model to these transformations, which disables including them in the state space. The measurements used with this appearance model are, of course, the calculation of color histograms in different image regions. The tracker in [96] is an example of using this appearance model. Another example is [49] where projection-histograms are modeling the object appearance in the image. A projection-histogram is the histogram of one row or column of pixels in the image region. The tracked object is represented by its projection-histograms on each of its rows and columns in the image.
Related to histogram-templates are templates of filter responses. For example, in [57] the phase components of filter responses are used to “describe” the object appearance.

1.3.4 3D description-based appearance models

When a 3D target description is given, the target’s appearance model is related to the description’s projection into the image. The description might be full (i.e., containing the 3D shape and colors as in [123]), or contain only the shape (as in [4], where the tracked object’s shape is described as a set of related line segments in space corresponding to the object edges). In the first case the raw image pixels may be used as the observations, where in the second case line segments in the image are sought. A less informative 3D description is used in [31], where an object silhouette is tracked using the object’s outer hull. The contour enclosing the silhouette was tracked using the measurements of spatial derivatives.

1.3.5 Appearance models based on template-spaces

In the previous image template-based appearance model there was a single, ideal template. The appearance model here is based on a space of image templates of the target.

In EigenTracking [11], the tracked object appearance in the image is modeled as varying inside a subspace spanned by some “principal” image templates, extracted by SVD from a set of given views of the object. Denote by $A$ the matrix obtained by “stacking” side-by-side the given images, each rearranged into a column vector by lexicographic order, and the SVD of $A$ by

$$A = U D V^T. \quad (1.10)$$

Then the $t$ left-most column vectors in $U$ (corresponding to the $t$ largest singular values) are the chosen “principal” image templates. Denoting these principal templates by $\{U_i\}_{i=1}^t$, an “ideal” object would then be

$$\sum_{i=1}^t c_i U_i \quad (1.11)$$

for any set of parameters $c_i$.

Another kind of template-space-based appearance model is used in Support Vector Tracking (SVT) [5]. In this tracking the target image space is separated from the space of background images using a Support Vector Machine. The measurements consist of the SVM score for the target.

1.3.6 On-line adapting appearance models

An alternative to using an appearance model which is valid during all the tracking process, an appearance model adapting in time may be used. This alternative is sometimes
inevitable, for example when no 3D information about an arbitrarily moving object is given. In such cases, the appearance has to be tracked simultaneously with the tracking of the required state. An example of such an appearance model is an adapting image template reported in [24]. In [100] histogram-templates of surface regions on a tracked human are adapting. Another example is [56], where the distributions of filter-responses modeling the tracked object appearance are changing on-line during the tracking process.

1.4 Filtering

When using a probabilistic framework, a PDF of the tracked state is propagated in time, and not merely a point estimate. In such cases, the mean state or the most probable state are taken for the final output of the tracker.

The filtering process is used to propagate the state PDF through the image sequence. This filtering is carried out by employing a filter. When \( s_{t+1} \) and the measurements at frame \( t \) are modeled as linear functions in \( s_t \) with additive Gaussian noises, a Kalman filter [60] is used to propagate the PDF of the tracked state. When these assumptions do not hold, the Extended Kalman filter (EKF) may be used to approximate the PDF of the tracked state. However, the EKF only approximates the correct PDF, and the approximations remain Gaussian. A usually better alternative to the EKF is employing a particle filter referred to as CONDENSATION [53] in the computer vision area. A CONDENSATION filter approximates the state PDF by a sample-set of state-weight pairs. This filter supports general distributions, and the approximated PDF tends to the exact one as the sample-set size is larger. In case the probability distributions are discrete and compact (or may be approximated as such), a General Distribution Filter [101] may be used instead. However, the General Distribution Filter is suited only for discrete state domains, and due to the quadratic complexity in the space size, the space has to be small in size and typically of very low dimensionality.

In the following are given the Kalman and the CONDENSATION filters, which are the filters used in the vast majority of visual tracking applications and are also the two used the research of this thesis.

1.4.1 The Kalman filter

The Kalman filter addresses the problem of estimating the state \( s \in \mathbb{R}^d \) of a discrete-time controlled process that is governed by the linear stochastic difference equation

\[
s_t = As_{t-1} + Bu_t + w_{t-1},
\]

(1.12)

with the measurement

\[
z_t = Hs_t + v_t.
\]

(1.13)

The random variables \( w_t \) and \( v_t \) represent the process and measurement noise, respectively. They are assumed to be independent of each other and independent of the noise
in previous times, and with Gaussian PDFs

\[ p(w_t) \sim N(0, Q_t), \quad p(v_t) \sim N(0, R_t). \quad (1.14) \]

The \( d \times d \) matrix \( A \) in the difference equation (1.12) relates the state at the previous time step, \( t - 1 \), to the state at the current step, \( t \), in the absence of either a driving function or process noise. The \( d \times l \) matrix, \( B \), relates the optional control input, \( u \in \mathbb{R}^l \), to the state \( s \). The \( m \times d \) matrix \( H \) in the measurement equation (1.13) relates the state to the measurement \( z_t \). Note that in practice \( A \) and \( H \) might change with each time step, but here we assume they are constant.

Given the above, the state PDF at all time steps, conditioned on all past measurements, is Gaussian. Its mean \( \hat{s}_t \) and covariance matrix \( P_t \) may be computed iteratively in each time step \( t \) by first incorporating the stochastic dynamics (1.12)

\[ \hat{s}_t^- = A\hat{s}_{t-1} + Bu_t \]
\[ P_t^- = AP_{t-1}A^T + Q, \]

and then updating according to the current measurement \( z_t \):

\[ K_t = P_t^- H^T \left( HP_t^- H^T + R \right)^{-1} \]
\[ \hat{s}_t = \hat{s}_t^- + K_t (z_t - H\hat{s}_t^-) \]
\[ P_t = (I - K_t H)P_t^- . \]

\( \hat{s}_t \) is then the best state estimate, with \( P_t \) being its error covariance matrix.

### 1.4.2 The Condensation filter

As described above, the Kalman filter is applicable only under the stochastic dynamics (1.12) and the measurement equation (1.13). When this model is not valid, or if the noise distributions are multimodal, the Kalman filter is not applicable. The CONDENSATION (Conditional density propagation) filter, on the other hand, is suited to arbitrary object dynamics, to arbitrary PDFs of the measurements conditioned on the object’s state, and to arbitrary process and measurement noises. Thus, it allows for general state PDFs, and in particular multi-modal.

When using the Condensation filter, the state PDF is not given explicitly, but is represented as a sample-set in the state space. The sample-set is updated iteratively for each new image in the input image sequence, in such a manner that corresponds to a sample-set randomly sampled from the state PDF in the new image. The sample-set representing the state PDF in image \( t \) in the sequence \( (t \in \mathbb{N}) \) consists of \( N \) couples \((s_t^{(i)}, \pi_t^{(i)})\), \( i = 1, 2, \ldots, N \), where the \( s \)'s are state-vectors and the \( \pi \)'s are corresponding weights, such that \( \sum_{i=1}^N \pi_t^{(i)} = 1 \). This sample-set represents the state PDF in image \( t \) in a manner that sampling randomly a state from the sample-set according to the respective weights is an approximated simulation of sampling a state from the exact state PDF in image \( t \). Thus, different statistics \( E[f(\cdot)] \) of the state PDF in image \( t \) may
be estimated by $E[f(s_t)] = \sum_{i=1}^{N} \pi_t^{(i)} f(s_t^{(i)})$, $s_t$ denoting a random variable of the true state at frame $t$. For example, the mean may be obtained using $f(x) = x$.

From computational efficiency reasons, the sample-set of the pairs $(s_t^{(i)}, \pi_t^{(i)})$ is augmented with the cumulative weights to yield an $N$-sample-set of triples $(s_t^{(i)}, \pi_t^{(i)}, c_t^{(i)})$, $i = 1, 2, \ldots, N$, where $c_t^{(i)} = \sum_{j=1}^{i} \pi_t^{(j)}$. These cumulative weights are used to aid in the random sampling from the sample-set according to the weights $\pi_t^{(i)}$s. Following is outlined one iteration of the CONDENSATION algorithm, propagating the “old” sample-set $\{(s_{t-1}^{(i)}, \pi_{t-1}^{(i)}, c_{t-1}^{(i)}), i = 1, \ldots, N\}$, corresponding to frame $t - 1$, into the “new” sample-set $\{(s_t^{(i)}, \pi_t^{(i)}, c_t^{(i)}), i = 1, \ldots, N\}$, corresponding to frame $t$:

1. **Select** $N$ new samples $s_t^{(i)}$, $i = 1, \ldots, N$, as follows:
   
   (a) Generate $N$ independent random numbers $r_i \in [0, 1]$, $i = 1, \ldots, N$, uniformly distributed.
   
   (b) Find for each $i = 1, \ldots, N$, by binary subdivision, the smallest $j_i$ for which $c_{t-1}^{(j_i)} \geq r_i$.
   
   (c) Set $s_t^{(i)} = s_{t-1}^{(j_i)}$.

2. **Predict** by sampling from $p(s_t|s_{t-1} = s_t^{(i)})$ to choose each $s_t^{(i)}$, $i = 1, \ldots, N$.

3. **Measure** and weight the new states in terms of the measured features $z_t$:

   $$\pi_t^{(i)} = p(z_t|s_t = s_t^{(i)}), \quad i = 1, \ldots, N.$$ 

   Then, normalize so that $\sum_i \pi_t^{(i)} = 1$ and store together with cumulative probabilities as $(s_t^{(i)}, \pi_t^{(i)}, c_t^{(i)})$ where

   $$c_t^{(0)} = 0,$$
   
   $$c_t^{(i)} = c_t^{(i-1)} + \pi_t^{(i)}.$$ 

The CONDENSATION complexity is $O(N \log N)$, where usually setting $N$ to a few hundreds of samples suffices (examples may be seen, e.g., in [53]). However, the tracker’s performance also depends on the state dimension and it is extremely difficult to get good results in the spaces of dimension greater than 10 [128].

**Applying CONDENSATION for visual tracking:** Applying the CONDENSATION filter requires the formulation of two models: the motion model $p(s_t|s_{t-1})$ and the observation model $p(z_t|s_t)$.

The motion model is usually modeled as a sum of a deterministic process and an unbiased noise: $s_t = f(s_{t-1}) + n$. Using such a model, the prediction phase in the CONDENSATION filter, consisting of sampling from $p(s_t|s_{t-1} = s_t^{(i)})$, may be carried out in a straightforward fashion: Calculate $f(s_t^{(i)})$ and add a random vector sampled from the noise distribution. The first part applied to the whole distribution (or all samples) is termed *deterministic drift* and the second part *stochastic diffusion* [53].
deterministic drift may, for example, be a first or second order temporal model. In such a case, the state parameters consist also their time derivatives of appropriate orders.

Modeling the observations consists of defining what are the measurements $z_t$, and their PDF conditioned on the tracked state, $p(z_t|s_t)$. Following are two examples of observation models used in CONDENSATION-based tracking:

1. In [53] contours enclosing an object in the image plane are tracked. The state domain includes the coordinates of B-splines’ control points or the affine transformation parameters of a hand-drawn contour template. In another case the state space is the vector space spanned by a number of contours. In all cases, edges are searched along $M$ lines normal to the hypothesized contour (see Figure 1.11), and the distance of the closest edge along each normal is measured. The conditional observation density is modeled as

$$ p(z|x) \propto \exp\left\{ -\sum_{m=1}^{M} \frac{1}{\sqrt{2\pi \sigma^2}} \min(d_{m}, \mu^2) \right\}, $$

$d_m$ being the distance of the closest edge along the $m$th normal, and $\mu$ and $\sigma$ constants. This PDF models the measurements as $\mathcal{M}$ i.i.d. (identical, independently distributed) zero-mean Gaussian random variables with variance $\sigma^2$. The bounding of the Gaussians from below accounts for the possibility of the lack of object edges due to low contrast with background.

2. In [96] a region inside a window (of any suitable shape) is tracked, allowing the window to undergo similarity transformations. The measurement is an HSV histogram $q(x_t)$ of the region inside the hypothesized window (see Figure 1.12), and the observation density is modeled as an exponential in the squared distance of the measured histogram from the reference histogram $q^*$:

$$ p(z|x) \propto \exp\left\{ -\lambda D^2(q^*, q(x)) \right\}, $$

where $D(q^*, q(x)) = \left[ 1 - \sum_{n=1}^{N} \sqrt{q^*(n)q(n;x)} \right]^{\frac{1}{2}}$, and $q^*(n)$ and $q(n;x)$ being the values of the histograms in bin $n$, $n = 1, \ldots, N$.

**Smoothing:** The sample-sets generated in the CONDENSATION filter represent the state distribution at each time $t$ conditioned on the past images $Z_t \equiv (z_1, \ldots, z_t)$. The distribution may occasionally be multi-modal, each mode corresponding to some hypothesis. As more images are processed, the false hypotheses decay and the state may be estimated reliably. However, if the whole image sequence is given in advance, the CONDENSATION filter may be modified to generate sample-sets representing the state distribution conditioned on the entire sequence rather than on only the preceding images. These distributions are conditioned on more information, and hence fewer false hypotheses are expected to arise. This results in fewer modes in the state distribution, explaining the term smoothing. A sequence-based smoothing algorithm given in [52] consists of storing an entire trajectory $S_t^{(n)} = (s_{t}^{(n,1)}, \ldots, s_{t}^{(n,t)})$ instead of only the state
Figure 1.11. The thick line is a hypothesized curve. The spines are curve normals along which high-contrast features are sought. Taken from [53].

Figure 1.12. A reference color histogram $q^*$ is gathered at time $t_0$. At time $t$ and for a hypothesized state $x$, the candidate color histogram $q_t(x)$ is gathered within the region $R(x)$. Taken from [96].
Figure 1.13. Before smoothing, multiple hypotheses can increase the variance of the state distribution (left) and shift the mean away from the object position (middle). After running the smoothing algorithm the estimated variance has dropped and the mean is now correctly positioned (right). The solid black line is the distribution mean, and the dotted white lines are high-scoring samples, where the width of the sample outline is proportional to its sample weight. Taken from [52].

$s_t^{(n)}$. Moments of the smoothed density $p(x_t | Z_t)$, can be estimated for $1 \leq \tau \leq t$ by computing the expectation

$$E[\phi(x_{\tau})|Z_t] \approx \sum_{n=1}^{N} \pi_t^{(n)} \phi(s_t^{(n,\tau)}).$$

Following is outlined one iteration of this smoothing algorithm, propagating the “old” sequence sample-set $\{(S_{t-1}^{(i)}, \pi_{t-1}^{(i)}), \ i = 1, \ldots, N\}$, corresponding to frame $t - 1$, into the “new” sequence sample-set $\{(S_t^{(i)}, \pi_t^{(i)}), \ i = 1, \ldots, N\}$, corresponding to frame $t$:

1. **Select** independently $N$ new base sequences $S_{t-1}^{(i)}$, $i = 1, \ldots, N$, with probability $\pi_{t-1}^{(i)}$. This can be done efficiently using cumulative distributions as before.

2. **Predict** by sampling from $p(x_t|x_{t-1} = s_{t-1}^{(i-1)})$ to choose each $s_t^{(i,\tau)}$, $i = 1, \ldots, N$.

3. **Measure** and weight the new states in terms of the measured features $z_t$:

$$\pi_t^{(i)} = p(z_t|x_t = s_t^{(i,\tau)}), \quad i = 1, \ldots, N.$$

Then set $S_t^{(i)} = S_{t-1}^{(i)} \cup s_t^{(i,\tau)}$, $i = 1, \ldots, N$.

Finally, normalize so that $\sum_i \pi_t^{(i)} = 1$.

A result of applying smoothing is demonstrated in Figure 1.13.
Chapter 2

Tracker Combination

2.1 Introduction

Over the past few years researchers have been investigating the enhancement of visual tracking performance by devising trackers that simultaneously make use of several different features. In some cases the feature combination is performed explicitly in a single tracker, usually in the observation phase, whereas in other cases the feature combination is performed implicitly by combining trackers that use different features.

Examples of the first kind may include [96] and [88], where better tracking performance was gained by using color histograms in several regions, each of a different part of the tracked object. Further robustness was achieved in [96] by also incorporating the histogram of the background as an additional feature. In [97] color is fused with either stereo sound, for tele-conferencing, or with motion, for surveillance with a still camera. In [125] three different color bands (RGB) were used together in the observation model. In [88] a particle filter, in which cascaded Adaboost detections [126] are used to generate the proposal distribution and color histograms are used for the likelihood, is used to track hockey players. In [54] the responses of different filters were used together to model the observation likelihood. Another related work is [22], where the tracker performs on-line switching between different features, according to their quality of discrimination between object and background pixels.

An example of the second kind of combination is [118], where multiple window-based trackers simultaneously track the same feature. Different situations in which miss-tracking occurs are categorized and dealt with by the appropriate trackers. In [40] the individual estimates of two tracking systems, based on the bounding-box and the 2D pattern of the targets, are combined to produce a global estimate for vehicle tracking. The combination is performed by utilizing instantaneous performance coefficients. In [105] two trackers, a region tracker and an edge tracker, are run in parallel. The two trackers, having complementary failure modes, correct each other based on their confidence measures. In [76] the results of two corner trackers, global correspondence and local relaxation, are merged using a classification-based approach and reliability attributes for each corner match. In [108] a system containing three co-operating detection and tracking modules is devised for tracking people in an indoor environment.
The three modules are an active shape tracker, a region tracker featuring region splitting and merging for multiple hypothesis matching, and a head detector to aid in the initialization of tracks. In [30] three different modules, which are guided by an integration module using the report of each module, are combined to track people. One module uses stereo data for range computation, another uses color data for skin color localization, and the third is a face pattern detection module. In [109] the so-called democratic integration scheme [119] was used to combine multiple cues for visual tracking. Two co-inference tracking algorithms, combining trackers of different modalities were developed in [131]. The first co-inference algorithm, combining importance sampling-based trackers, uses the state PDF of each tracker as the importance function for the other tracker. The second co-inference algorithm uses the state PDF of one tracker as the importance function for the other tracker, and then the former tracker uses the latter tracker’s state PDF to compute its own state PDF and observation model parameters through an EM framework.

We consider here a general methodology for the combination of synchronous trackers (synchronous in the sense that all trackers receive their frames at the same time instances.) The previous work described above focuses either on feature combination within the tracker, or on a combination of trackers carried out in a specific method tailored to the particular trackers considered. For example, the combination in [108] consists of running multiple trackers in parallel, where each one is devised for a specific sub-task and is dependent on the output of other algorithms for initialization or correction. The trackers described in [105] run in parallel, and switching between their outputs is done according to ad-hoc rules. Perhaps the closest work to our approach is [109], where multiple trackers are run in parallel to track the same object and provide 2D-location distribution estimates, which are combined heuristically by weighted sum.

Unlike previous work, the approach presented here is general and not specific to the particular separate trackers used. The separate trackers are treated as “black boxes”, and only their output, which may be modified before their propagation to the next time step, are used. Consequently, the developed framework is fairly general, making it suitable for combining a wide range of trackers in a simple manner. These trackers may even be of different state spaces of different dimensionality. Since only the separate trackers’ final estimates are used, a second gained advantage is that the combination may be performed in a distributed setting, where each separate tracker runs on a different site and uses different data, while exchanging only their final state PDF (Probability Density Function) estimates and avoiding the need to share the data, which is typically much larger in size (e.g., as in [14]). The suggested framework enables tracking enhancement not only by using several sources of data associated with one object, but also by tracking different, related objects (as in the context of the Constrained Joined Likelihood Filter (CJLF) in [99], where linked objects are tracked with constraints.) Although the simultaneous tracking of multiple objects is a widely approached topic (e.g., [72], [48], [107] and many other papers in [1]), it has been approached mainly in the context of independently moving objects.

We would like to stress the black box constraint to which our framework is subject,
which prevents the use of other data combination methods that are more exact (e.g., [8, 33, 54, 96, 97, 131].) In particular, in the general context of data combination, message passing schemes in the field of graphical models may be an elegant framework for the combination of different data (e.g., [9, 50, 110].)

We start by defining the context in Section 2.2. Then we describe methods to combine explicit PDF-yielding trackers in Section 2.3, beginning with the case of a common state space, and continuing with combining trackers using different state spaces. We show how to combine CONDENSATION-based trackers in Section 2.4. The combination of explicit PDF-yielding trackers with CONDENSATION-based ones is considered in Section 2.5. In Section 2.6 we show how the framework may be used to combine trackers of different, related objects. We continue by presenting experimental results in Section 3.4, and conclude with a summary and discussion in Section 5.6.

2.2 Context

We consider the following task: given two or more synchronous trackers (synchronous in the sense that all trackers receive their frames at the same time instances), we would like to combine their outputs into one estimate. The trackers may track a common state or different, albeit related, states. We assume the following regarding the separate trackers:

1. The trackers are black boxes; only their final estimates are observed and may be changed before their back feeding for propagation to the next time step. Note that due to this black box constraint, the inner states and all the modeling used by the trackers, which includes their observation processes and the tracked state’s dynamic model, are hidden.

2. The trackers provide a PDF estimate representation of the tracked state, sequentially for each image, and these PDF estimates may be modified before their propagation to the next time step. A specific tracker outputs either an explicit PDF, or a sample-set of it via CONDENSATION. Such trackers are very common. For example, any tracker using Kalman filtering explicitly provides a Gaussian PDF (namely, a mean vector and covariance matrix) of the tracked state (e.g., [12]). Other trackers employing a general discrete probability distribution for tracking are described in [101], [102] and [7]. Trackers providing samples from general PDFs using CONDENSATION are also widely used. Further on in the chapter, we distinguish between explicit PDF-yielding trackers and CONDENSATION-based trackers that provide sample-sets from the state PDF.

3. The trackers are conditionally independent, i.e., each tracker relies on features that, given the tracked state, are conditionally independent of the features used by the other trackers. That is, if at time \( t \) tracker \( T_i \) uses features \( z_i^t \) and tracker \( T_j \) uses features \( z_j^t \) \((i \neq j)\), then we assume that for every state at time \( t \), \( x_t \),

\[
p(z_i^t, z_j^t | x_t) = p(z_i^t | x_t) \cdot p(z_j^t | x_t).
\] (2.1)
While this assumption is indeed difficult to verify, there are many cases where it is true or approximately true. Moreover, tracker combination in general is most beneficial when the separate trackers use conditionally independent features in the first place. Therefore, such a conditional independence assumption has been made also in many other visual tracking-related papers (e.g., [30, 56, 88, 96, 97, 125]). Nevertheless, there are tracker combinations that do not rely on this assumption, such as [131].

For clearer presentation we will discuss combining two trackers, but the entire discussion and results may be easily generalized for an arbitrary number of trackers.

2.3 Combining explicit PDF-yielding trackers

2.3.1 Same state space

Let $T_1$ and $T_2$ be two conditionally independent trackers, tracking an object in a common state space. Denote by $z^i_t$ the features extracted by $T_i$ in the $t^{th}$ frame ($t = 0, 1, 2, \ldots$), and denote by $Z^i_t \triangleq \{z^i_t\}_{t=0}^t$ the features extracted by $T_i$ from all frames up to the $t^{th}$ frame. At time $t$ $p(x_{t-1}|Z_{t-1})$ – the tracked state PDF at time $t - 1$, and $I_t$ – the frame at time $t$, are available to each tracker as its input. Each tracker $T_i$ extracts from the frame its relevant features $z^i_t$ and outputs $p(x_t|Z^i_t)$ – the PDF of the tracked state at time $t$, given the features $z^i_t$ in the $t^{th}$ frame and all the previously extracted features as well. The overall system of the two separate trackers is illustrated in Figure 2.1.

Now, using the notation $Z_t \triangleq (Z^1_t, Z^2_t)$, let us derive $p(x_t|Z_t)$ – the state PDF of the combined system. In the derivation we apply Bayes’ rule [91] and assumption (2.1), assumed valid also when given past observations. So,
\begin{align}
p(x_t|z_t) &= p(x_t|z_t^1, z_t^2, Z_{t-1}) \\
&= \frac{p(x_t, z_t^1|z_{t-1})p(x_t|Z_{t-1})}{p(z_t^1, z_t^2|z_{t-1})} \\
&= p(z_t^1|x_t, Z_{t-1}) \cdot p(z_t^2|x_t, Z_{t-1}) \cdot \frac{p(x_t|z_{t-1})}{p(z_t^1|Z_{t-1})} \\
&= \frac{p(x_t|x_t^1, Z_{t-1})p(x_t|z_{t-1})}{p(x_t|Z_{t-1})} \\
&= \frac{p(x_t|x_t^1, Z_{t-1})p(x_t|z_{t-1})}{p(x_t|Z_{t-1})} \cdot \frac{p(x_t|Z_{t-1})}{p(z_t^1|Z_{t-1})} \\
&= \frac{p(x_t|x_t^1, Z_{t-1})p(x_t|z_{t-1})}{p(x_t|Z_{t-1})} (2.2)
\end{align}

This expression can be written as the product

\begin{equation}
p(x_t|Z_t) = k \cdot \frac{p(x_t|z_t^1, Z_{t-1})p(x_t|z_t^2, Z_{t-1})}{p(x_t|Z_{t-1})}, (2.3)
\end{equation}

where \( k \) is the normalization constant independent of \( x \), ensuring the PDF has a unit integral. By induction on time \( t \) beginning from \( t = 0 \), it is now easily seen that \( T_1 \) and \( T_2 \) may be combined by multiplying their output PDFs (which will become \( p(x_t|z_t^1, Z_{t-1}) \) and \( p(x_t|z_t^2, Z_{t-1}) \), respectively), dividing by the PDF \( p(x_t|Z_{t-1}) \), and scaling to have a unit integral.

The PDF \( p(x_t|Z_{t-1}) \) in the denominator of (2.3) is usually referred to as the prediction, or prior PDF – the PDF of the tracked state predicted for the time when the \( t \) – th frame is taken, but prior to the measurements in this frame. This PDF may be determined by the tracked object dynamics \( p(x_t|x_{t-1}) \) and the posterior at the previous time, \( p(x_{t-1}|Z_{t-1}) \). When combining the trackers, the optimal choice would be to estimate this PDF according to the tracked object dynamics in a separate module. Alternatively, it is common for the separate trackers to have a prediction phase in which the prior PDF is estimated (e.g., a Kalman filter or a general distribution filter [101]). If there is access to this already-computed PDF, it may be simply taken from there. However, this option cannot be accomplished under the black boxes constraint, where only \( p(x_t|z_t^i, Z_{t-1}) \) are available.

In order to obtain the combined estimate, the PDF in the denominator of (2.3) has to be set in a way that the black box constraint is satisfied, that is, without assuming any knowledge regarding the dynamic model. Clearly, in typical situations visual tracking may be performed even if the dynamic model is not exactly known. Since some dynamic model must be assumed in order to carry on with the tracking, usually a wide PDF is used as the dynamic model. We chose to make the worst case assumption that no knowledge regarding the tracked object dynamics is given in the combination. (The separate trackers, however, do use motion models.) This leads directly to the independence between the current state and previous observations. Thus, we set the prior in the denominator of (2.3) to a pseudo-prior that is constant in time,

\begin{equation}
p(x_t|Z_{t-1}) \equiv p(x), (2.4)
\end{equation}

which allows us to treat the separate trackers as black boxes. The combined system is illustrated in Figure 2.2.

Since we would also like to assume that any state that is contained in the image domain is of equal, or approximately equal, probability when no data is given, the
Figure 2.2. Combining $T_1$ and $T_2$ of Figure 2.1: two explicit PDF-yielding trackers that use a common state space.

The static PDF in the right-hand-side of (2.4) should be set to be uniform or very wide with respect to the image size.

Note that the exact combined state PDF (not restricted by the black box constraint and thus may use the true dynamic model) is proportional to the product of the prior and the observation likelihoods (see third line in (2.2)), where the former is moved to the denominator after manipulations. Thus, using a static, uniform PDF as the pseudo-prior in the denominator of (2.3) is equivalent to replacing the true prior by its square, which does not change the qualitative effect of the prior on the resulting state estimate: we multiply by a high value for the same states weighted higher by the true prior. We found that this inaccuracy did not come to fruition in our experiments.

Perhaps a more accurate alternative to assumption (2.4) would be to set the pseudo-prior in the denominator of (2.3) to a diffused version of the posterior at the previous time, $p(x_{t-1} | Z_{t-1})$. This corresponds to assuming some sort of a general dynamic model in the combination, e.g., a zero-mean Gaussian with a diagonal covariance matrix as the evolution PDF. This will, however, imply some violation of the black box constraint, and as we saw in our experiments, was not necessary.

Trackers $T_1$ and $T_2$ might, for example, be Kalman filters. In this case, if we approximate the pseudo-prior of (2.3) as Gaussian, then the combined PDF remains Gaussian $N(\mu, C)$, with the covariance matrix and mean being

$$C^{-1} \triangleq C_{1}^{-1} + C_{2}^{-1} - C_{3}^{-1},$$
$$\mu \triangleq C(C_{1}^{-1}\mu_{1} + C_{2}^{-1}\mu_{2} - C_{3}^{-1}\mu_{3}),$$

where $N(\mu_{1}, C_{1})$ and $N(\mu_{2}, C_{2})$ are the PDFs provided by the separate trackers, and $N(\mu_{3}, C_{3})$ is the prior PDF [33] (see alternative proof in the Appendix A.) Being Gaussian, the combined PDF may be fed back into $T_1$ and $T_2$.

Note that setting the pseudo-prior in the denominator of (2.3) to be of very high variances is realized here by setting the variances in $C_{3}$ to be very large. This ensures that $C$ may become non-positive only in the extreme case where both separate trackers
provide estimates of very high variances, that is, when both trackers have lost the target. Note also that if desired, the pseudo-prior in the denominator of (2.3) may be brought to the uniform distribution by letting $C_3$ approach the identity matrix scaled by a large factor. Then $C_3^{-1}$ approaches the zero matrix, making the combined PDF $N(\mu, C)$ independent of the pseudo-prior in the denominator of (2.3), as expected.

In case the probability distributions are discrete and compact (or may be approximated as such), $T_1$ and $T_2$ may be general distribution filters [101].

### 2.3.2 Different state spaces

Often we can construct conditionally independent trackers in different state spaces, and then would like to combine them to enhance the tracking performance. This is indeed possible (if the state spaces are related). In order to combine the trackers we require that a probability distribution on each state space, conditioned on the state in any of the other state spaces, is available. In other words, if we denote the state space of tracker $T_i$ by $S_i$, then for all $i \neq j$, the conditional PDF $p(x^j|\mathbf{z}^i)$ where $x^i \in S_i$ and $x^j \in S_j$ has to be given. Since this PDF will be used for translating the PDF from the source state space $S_i$ into the target state space $S_j$, we denote it by the special name *translator* and notate it as $p_{S_i \rightarrow S_j}(x^j|x^i)$.

Note that these translators mediate only between the state spaces and have no connection to the separate trackers. That is, when combining the trackers, the translators only deal with ‘what’ is being tracked and not ‘how’ it is tracked. Thus, in this respect their usage does not violate the black box constraint.

We turn now to obtain the combination of $T_1$ and $T_2$, two conditionally independent trackers, tracking an object in different state spaces with variables at time $t$ $x_t$ and $y_t$, respectively. Denote by $S_1$ and $S_2$ the corresponding spaces, and denote by $\mathbf{z}_1^t$ and $\mathbf{z}_2^t$ the features used in time $t$ by $T_1$ and $T_2$, respectively. The overall system of the two separate trackers is illustrated in Figure 2.3.

In order to combine the trackers’ estimations, each provided PDF has to be translated into a PDF in the state space of the other tracker. This may be accomplished given the translators: the PDFs of the state in one space given the state in the other space –
Figure 2.4. Combining $T_1$ and $T_2$ of Figure 2.3: two explicit PDF-yielding trackers using different state spaces.

$p_{S_2 \rightarrow S_1}(x|y)$ and $p_{S_1 \rightarrow S_2}(y|x)$ ($x \in S_1$, $y \in S_2$). By marginalization we can use these translators to compute $p(x_t|z_{t-1}^2)$ as

$$p(x_t|z_{t-1}^2) = \int_{S_2} p(y_t|z_{t-1}^2, Z_{t-1}) \ p_{S_2 \rightarrow S_1}(x_t|y_t) \ dy_t,$$  \hspace{1cm} (2.6)$$

and similarly for $p(y_t|z_{t-1}^2, Z_{t-1})$. Note that it is assumed in (2.6) that the translator is independent of the past measurements (because we assume no knowledge in the combination regarding the object dynamics) and with respect to measurements performed by the tracker of the source space. Once we have the translated PDFs, we can combine the trackers as in the case of the common state space. Observe that now the combined system has two outputs, one in each state space. If one of the spaces is contained in the other, the estimate in the latter, more detailed, space may be used for the final output display. In case neither of the space is contained in the other, the two estimates or a combination of their variables may be used for the final display. See Figure 2.4 for the combined system.

It should be noted that for consistency the pseudo-priors $p(x)$ and $p(y)$ of (2.4) should be approximated consistently with the translators:

$$p_{S_1 \rightarrow S_2}(y|x)p(x) = p_{S_2 \rightarrow S_1}(x|y)p(y).$$  \hspace{1cm} (2.7)$$

Then, if the final estimated state PDFs in the different state spaces have common variables, they will have identical marginal PDFs in the final, joint PDF estimates.
2.4 Combining CONDENSATION-based trackers

2.4.1 Same state space

Many trackers use CONDENSATION to propagate an approximated state PDF, represented by a sample. This approach is used when an analytic form of the PDF is unknown or unjustified. When using CONDENSATION, sample-sets of PDFs are given rather than explicit PDFs as in the previous cases. We use the same notations as in [53] and denote an \( N \)-sample-set of the tracked state PDF at time \( t \) by \( \{ s_i^{(n)}(t), \pi_i^{(n)}(t), n = 1, \ldots, N \} \), \( s_i^{(n)}(t) \) being the sampled states and \( \pi_i^{(n)}(t) \) the corresponding weights. Let \( C_1 \) and \( C_2 \) be two conditionally independent CONDENSATION-based trackers using the same state space, and denote their measurements in time \( t \) by \( z_1^t \) and \( z_2^t \), respectively.

The overall system of the two separate trackers is illustrated in Figure 2.5.

Not restricted by the black box constraint, the combination here may be performed simply by using one of the separate trackers and augmenting it with the measurements performed by the other trackers. That is, for each sample all measurements are performed, their corresponding pre-normalized weights are calculated and their product is taken as the final pre-normalized weight. The justification for the multiplication of the pre-normalized weights, which are the observation likelihoods, follows immediately form the assumption of the conditional independence between the observations.

Similarly to the combination of the explicit PDF-yielding trackers, in order to combine \( C_1 \) and \( C_2 \) under the black box constraint, we need to create sample-sets corresponding to the normalized ratio between the product of the two state PDFs represented by the two originally provided sample-sets and the pseudo-prior \( p_s(s) \) of Equation (2.4). To accomplish this, we propose to perform the following:

1. Multiply the weight \( \pi_i^{(n)}(t) \) of each sample \( (s_i^{(n)}(t), \pi_i^{(n)}(t)) \) by the (multivariate) kernel density estimate of the other state PDF in the state \( s_i^{(n)}(t) \);
2. Divide the weight of each sample by the pseudo-prior of (2.4) at the correspond-
ing state, \( p_s(s_i^{(n)}) \); and

3. Normalize the resulting sample-sets to unit total weight.

The kernel density estimate of the state PDF provided by \( C_i \) is

\[
p(x_t|z_t^i, Z_{t-1}) = \sum_{n=1}^{N_i} \pi_t^{(n)}(n) K\left(\frac{x_t - s_t^{(n)}}{h}\right),
\]

(2.8)

where \( K(s) \) is the kernel and \( h \) is the window radius. Various types of kernels may be used. Under certain conditions, the minimization of an average global error between the estimate and the true density yields the (multivariate) Epanechnikov kernel \[104\]

\[
K(s) = \begin{cases} 
  c \cdot (1 - ||s||^2) & ||s|| < 1, \\
  0 & \text{otherwise}
\end{cases}
\]

which was also used in all the relevant experiments in this chapter. Thus, the weight update of the sample-set provided by \( C_1 \) is carried out by the assignment

\[
\pi_t^{1(n)} \leftarrow \frac{\pi_t^{1(n)}}{p_b(s_t^{1(n)})} \cdot \sum_{j=1}^{N_2} \pi_t^{2(j)} K\left(\frac{s_t^{1(n)} - s_t^{2(j)}}{h}\right), \quad n = 1, \ldots, N_1, \quad (2.9)
\]

followed by normalization,

\[
k = \sum_{j=1}^{N_1} \pi_t^{1(j)}, \quad \pi_t^{1(n)} \leftarrow \frac{\pi_t^{1(n)}}{k}, \quad n = 1, \ldots, N_1. \quad (2.10)
\]

The weights in the sample-set provided by \( C_2 \) are similarly updated. If Gaussian kernels are used, the method suggested in \[110\] to multiply multiple kernel density estimates may be utilized. The combined system is illustrated in Figure 2.6.

### 2.4.2 Different state spaces

Consider now the combination of Condensation-based trackers corresponding to different state spaces. As in the explicit PDF-yielding case, we rely on translators between the state spaces; \( p_{S_i \rightarrow S_j}(x^i|x^j) \) are available for all \( i \neq j \).

Let \( C_1 \) and \( C_2 \) be two conditionally independent Condensation-based trackers, using different state spaces. Denote by \( S_1 \) and \( S_2 \) the corresponding spaces and denote the measurements performed in time \( t \) by \( z_t^1 \) and \( z_t^2 \), respectively. The illustration in Figure 2.5 given earlier for the case of a common state space may also be appropriate here, though one should bear in mind that the states in the sample-sets of \( C_1 \) are of space \( S_1 \) and the states in the sample-sets of \( C_2 \) are of space \( S_2 \). As in the previous case where both trackers used the same state space, here we need to create for each state space a sample-set corresponding to the normalized ratio between the product of the two state PDFs represented by the two provided sample-sets and the pseudo-prior
The state PDF approximation in the other direction is performed similarly. Now we can multiply the weight \( \pi_i(n) \) of each sample \((s_i(n), \pi_i(n))\) by the other tracker’s state PDF at this sample’s state \(s_i(n)\) in the same space, as in the case of the common state space. Thus, the weight update of the sample-set provided by \(C_1\) is carried out by the assignment

\[
\pi_i^{(n)} \leftarrow \frac{\pi_i^{(n)}}{p_{S_1}(s_i^{(n)})} \sum_{j=1}^{N_2} \pi_j^{(j)} \int_{S_2} K \left( \frac{y_t - s_j^{(j)}}{h} \right) p_{S_2 \rightarrow S_1}(s_j^{(j)} | y_t) \, dy_t, \quad n = 1, \ldots, N_1, \tag{2.12}
\]

followed by normalization (2.10). The weights in the sample-set provided by \(C_2\) are updated similarly. The obtained combination is illustrated in Figure 2.7.

It should be remembered that in the general context of Condensation-based tracker combinations, more exact methods can be used. For example, given functions \(F_j\) from one state space \(S_i\) to all other state spaces \(S_j\), we can use the tracker of state space \(S_i\) and augment its measurements by the ones that are performed by the other trackers and are related to the “translated” sampled states, \(F_j(s_i^{(n)})\). (For example,
there is a (straightforward) function relating a contour to a bounding box.) Then, the pre-normalized weights related to all measurements are computed and multiplied for each sample to give the final pre-normalized weights. As in the case of the common state space above, the justification for the multiplication of the pre-normalized weights follows immediately from the assumption of the conditional independence between the observations. Note that this method assumes that for the source state space index $i$ and for all other $j \neq i$ $p(z_j|x_i) = p(z_j|F_j(x_i))$. Notice that if the $F_j$s are bijections, no assumption is made, and if not, this assumption is very realistic; since naturally the observations made by any separate algorithm are dependent only on the object parameters sought by that specific algorithm. Another option that is not restricted by the black box constraint is [131], where neither conditional independency is assumed nor are translators needed, allowing the combination of trackers in unrelated state spaces, such as shape and color. These require, however, that the internal processes of the trackers be merged.

2.5 Combining explicit PDF-yielding with CONDENSATION-based trackers

We shall now consider the combination of an explicit PDF-yielding tracker with a CONDENSATION-based one. Similarly to the previous cases, the PDF provided by the explicit PDF-yielding tracker has to be multiplied by the PDF represented by the sample-set provided by the CONDENSATION-based tracker, and the weights of the samples provided by the CONDENSATION-based tracker have to be multiplied by the PDF.
provided by the explicit PDF-yielding tracker at their corresponding states (followed
by division by the pseudo-prior of (2.4) and normalization in both). The latter multi-
plication may be straightforwardly performed in the case when both trackers track in a
common state space. If the spaces are different, then the PDF should be translated into
the state space of the samples, using (2.6). The former multiplication, however, is more
problematic, since the sample-set has to be transformed into an explicit PDF of a pa-
rameterization suitable for combination with the other PDF. Therefore, (2.8) and (2.11)
cannot be used here. Instead, a PDF of the desired parameterization is estimated from
the sample-set. For example, when the explicit PDF-yielding tracker uses Kalman fil-
tering, the PDF estimated from the sample-set should be Gaussian. A Gaussian PDF is
parameterized by its mean vector \( \mu \) and covariance matrix \( C \). These may be estimated
from the sample-set \( \{ s_t^{(n)}, \pi_t^{(n)}, n = 1, \ldots, N \} \) as

\[
\mu = \sum_{n=1}^{N} \pi_t^{(n)} s_t^{(n)}, \quad C = \sum_{n=1}^{N} \pi_t^{(n)} (s_t^{(n)} - \mu)(s_t^{(n)} - \mu)^T.
\]

After estimating the Gaussian PDF from the sample-set, the former may be combined
with the Gaussian PDF provided by the Kalman-based tracker.

### 2.6 Combining trackers of different, related objects

In addition to the combination of trackers that track a common object, the aforemen-
tioned combination methods that combine trackers of different state spaces may also be
used for enhancing the performance of trackers that track different objects, as long as
the objects have some coupling between them. Being coupled, the state of one tracked
object bears information on the state of the other tracked object(s). Approximating the
coupling between each pair of tracked objects as a pair of PDFs of the object state con-
ditioned on the other object state, these PDFs may play the role of the translators in
the combination of trackers of different state spaces. Thus, the aforementioned combi-
nation methods for combining trackers of different state spaces may also be used here.
This shows that simultaneously tracking multiple, coupled objects may be treated as
tracking a common object in different state spaces. Note also that the conditional inde-
pendence assumption is usually satisfied here even if both trackers use the same kind
of features, since the features used by the trackers relate to different objects.

Note that as in the case of the common-object tracking, the removal of the black
box constraint here may enable more exact tracking, especially if a joint measurement
for all separate trackers is performed and a joint observation likelihood is evaluated.
This may lead to better target-measurement associations, prohibiting scenarios such
as multiple trackers locking onto the same feature (e.g., as in the Joint Probabilistic
Data Association Filter (JPDAF) [8]) and account also for depth ordering affecting the
objects’ appearances (e.g., as in the Joint Likelihood Filter (JLF) and CJLF in [99].)
2.7 Experiments

We performed six experiments, in four of which we used standard image sequences from the IEEE International Workshops on Performance Evaluation of Tracking and Surveillance (PETS) [2, 3]. In the other two we used self-made sequences.

The first three experiments tested the combination of trackers that track a common object. In the first experiment a pair of simple, explicit PDF-yielding trackers, tracking a person outdoors using different state spaces of different dimensionality, were combined. In the second experiment, two CONDENSATION-based trackers tracking a ball using a common state space were combined. The third experiment tested the combination of two CONDENSATION-based trackers that track a person’s head using different state spaces of different dimensionality.

The other three experiments tested the combination of trackers that track separate, related objects. In the fourth and fifth experiments two trackers tracking a person’s left and right eyes were combined. The fourth experiment used two explicit PDF-yielding trackers, whereas the fifth experiment used a CONDENSATION-based tracker for one eye and an explicit PDF-yielding one for the other eye. The sixth experiment tested the combination of two probabilistic exemplar-based trackers [117], tracking eye states, as explicit PDF-yielding trackers.

In all the experiments combining trackers in the same state space, the pseudo-priors of (2.4) were approximated as uniform. In the experiments involving translators, these state PDFs were set to be very wide while ensuring that (2.7) is satisfied or approximately satisfied.

All the experimental results validate the significantly improved performance of the composite tracker over the separate trackers. The wide variety of the experimental situations demonstrate the wide scope of the proposed framework.

2.7.1 Experiment I: common object, explicit PDF-yielding trackers, different state spaces

This experiment demonstrates the combination of two explicit PDF-yielding trackers, tracking a common object in different state spaces of different dimensionality. We tracked a walking person in an outdoor scene, using a 1:2 down-sampled gray-level version of the image sequence from the First IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS2000). Two simple trackers were implemented. The first tracker $T_1$ tracked the center of the person by a template search of part of his body (the area of the stomach and chest). The state space was composed of two parameters – the 2D center coordinates. The second tracker $T_2$ tracked the bounding box of the person using background subtraction. The state space was composed of four parameters: the center coordinates, the height and the width of the box. The two states were manually initialized and propagated using a Kalman filter. A few frames with the tracking results imposed are shown in Figure 2.8. The template-based tracker, giving the light dot in the images, failed already at the beginning due to an occlusion by a tree. Afterward, the background subtraction-based tracker failed due to the proximity
Figure 2.8. A few frames of the sequence used in Experiment I, with the tracking results of the two separate trackers imposed (marked by boxes and dots). The template-based tracker failed already in the beginning due to an occlusion by a tree. Afterwards, the background subtraction-based tracker failed due to the proximity of the moving car to the tracked person.

Next, the two trackers were combined using only their output (Section 2.3.2). Denote the two coordinates of the template center sought by the template-based tracker at time \( t \) by \( x_1^t, y_1^t \), and denote the center coordinates, width and height of the bounding box sought by the background subtraction-based tracker at time \( t \) by \( x_2^t, y_2^t, w_t, h_t \), respectively. The translator from the space of the second tracker to the space of the first was set to
\[
p_{S_2 \rightarrow S_1} (x_1^t, y_1^t | x_2^t, y_2^t, w_t, h_t) = \delta_{x_2^t,y_2^t} (x_1^t, y_1^t),
\]
yielding the translated PDF (using (2.6))
\[
p(x_1^t, y_1^t | z_2^t, Z_{t-1}) = p_{S_2 \rightarrow S_1} (x_2^t, y_2^t | z_2^t, Z_{t-1}).
\]
Thus, the PDF provided by the second tracker was translated into the space of the first by making the PDF of the template center equal to the marginal PDF of the bounding box center, and discarding the height and width parameters. Being a marginal PDF of a Gaussian PDF, this translated PDF remains Gaussian. For the other direction of translation, the translator was set to
\[
p_{S_1 \rightarrow S_2} (x_2^t, y_2^t, w_t, h_t | x_1^t, y_1^t) = \delta_{x_1^t,y_1^t} (x_2^t, y_2^t) \cdot N_{w_t,h_t} \left( \begin{pmatrix} 10 \\ 20 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix} \right),
\]
where \( N_{w_t,h_t} \) stands for a Gaussian distribution of \( w_t \) and \( h_t \) with mean and covariance matrix, in pixel units, as indicated. (The exact parameters of the distribution of \( w_t \) and \( h_t \) are not important, as long as they are mutually normal and of very large variances.) This yielded the translated PDF
\[
p(x_1^t, y_1^t, w_t, h_t | z_1^t, Z_{t-1}) = p_{S_1 \rightarrow S_2} (x_2^t, y_2^t | z_1^t, Z_{t-1}) \cdot N_{w_t,h_t} \left( \begin{pmatrix} 10 \\ 20 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix} \right).
\]
Thus, the PDF of the bounding box center was made equal to the PDF of the template center, and augmented with two additional, independent Gaussian random variables with very large variances, as the height and width. Being a product of Gaussian PDFs,
Figure 2.9. Results of Experiment I using the composite tracker. The tracking succeeded until the person left the scene.

this translated PDF also remains Gaussian. The pseudo-prior of (2.4) used for the center coordinates state space was uniform (a Gaussian of null inverse covariance matrix), and in the bounding box state space it was set to be uniform in the center coordinates and separably $N_{w_t,h_t} \left( \begin{pmatrix} 10 \\ 20 \\ 100 \\ 100 \end{pmatrix}, \begin{pmatrix} 0 \\ 100 \\ 0 \\ 100 \end{pmatrix} \right)$ in the width and height. The reader may verify that these choices satisfy (2.7). Since both the translated PDFs and the chosen pseudo-priors in the denominator of (2.3) are Gaussian, their combinations remain Gaussian (using (2.5)), which makes the feedback to the Kalman filters feasible. The composite tracker now overcame all the situations in which the separate trackers failed, and tracked the person successfully until he left the scene. Note that, as expected, the tracking does not seem to be degraded by the combination with the output of an unsuccessful tracker, since the degradation of the PDF provided by a successful tracker is small due to the implicit small weighting of the PDF of large covariances. A few representative frames are shown in Figure 2.9.

We used this experiment also to test the extent of inaccuracy in the combined state PDFs caused by setting the pseudo-priors in the denominator of (2.3) to the stationary wide PDFs used. We have compared a few of the these PDFs with their exact counterparts, which were calculated using the same motion model as in the separate trackers for the prior in the denominator of (2.3) (that is, when not constrained by the black box restriction.) The motion models used were first order temporal predictions summed with Gaussian noise, dimensions uncorrelated, with standard deviations of 1 pixel in each dimension. The differences between the exact PDFs and the approximate ones turned out to be minor (a fraction of a pixel in average, both in mean values and in standard deviations.)
2.7.2 Experiment II: common object, CONDENSATION-based trackers, common state space

In this experiment two CONDENSATION-based trackers, tracking a common object, were combined. We tracked a ball rolling around amidst an assortment of objects. The image sequence was a gray-level one, taken by a constantly moving camera. (To make the tracking harder, we did not rely on any consistency in the camera motion.) The 3D tracked state consisted of the circle enclosing the ball in the image plane (center and radius). The first tracker was CONDENSATION-based and used intensity edges as its observations: a hypothesized circle was weighted according to the amount of edges near the circle’s contour. A few frames with the tracking results imposed are shown in Figure 2.10 (a representative set of samples from the sample-set is shown). Due to camouflage provided by another ball, a second, false hypothesis was generated, shunting aside the true hypothesis.

Next, we combined the tracker with another CONDENSATION-based tracker, the observations of which were gray-level differences from a reference gray-level value of the tracked ball. This tracker, which was manually initialized as well, turned out to be very weak and lost track after just a few frames due to other areas in the image that had a similar gray-level. The trackers were then combined as described in Section 2.4.1. The composite tracker overcame the camouflage provided by the other ball, as well as an occlusion caused by a third ball (Figure 2.11).
2.7.3 Experiment III: common object, CONDENSATION-based trackers, different state spaces

Here we demonstrate the combination of two CONDENSATION-based trackers, tracking a common object in different state spaces of different dimensionality. The experiment consisted of tracking the head of a person taking part in a “smart meeting”. We used a 1:2 down-sampled version of an image sequence from the Fourth PETS Workshop (PETS-ICVS, scenario A, camera 2). The first tracker specified the head’s shape and pose as an ellipse (defined by five parameters: center coordinates, main and secondary axes, and angle). As in the previous experiment, the tracking was performed via a CONDENSATION-based tracker, using color edges as its measurements: a hypothesized ellipse was weighted according to the amount of edges near the ellipse’s contour. The initialization was performed manually. A few frames with the tracking results imposed are shown in Figure 2.12 (the mean of the sample-set is shown). Due to many edges in the interior of the tracked object region, the tracker yielded poor results.

In order to improve the tracking, we combined the last tracker with another CONDENSATION-based tracker that was manually initialized and tracked the vertical axis of the head (center coordinates and length). The measurements performed by this tracker were the deviations of the color at the top of the hypothesized vertical axis, the color at its bottom and the color below it from reference colors (the colors of the hair, skin and shirt, respectively). A few frames of the tracking results obtained by this tracker are shown in Figure 2.13. As before, the mean of the sample-set is shown. The trackers were combined as described in Section 2.4.2. Denote the tracked ellipse center, main axis length, secondary axis length and angle at time $t$ by $(x_1^t, y_1^t, a_1^t, b_1, \theta_1)$, and denote
Figure 2.12. A few frames of the sequence used in Experiment III, with the tracking results of the edge-based tracker imposed. Due to many edges in the interior of the tracked object region, the tracker yielded poor results.

Figure 2.13. A few frames of the tracking results obtained in Experiment III using the reference colors-based tracker. The mean of the sample-set is shown.
Figure 2.14. Results of Experiment III using the composite tracker. The composite tracker overcame the distractions caused by the edges inside the tracked head’s region.

the center and length of the tracked vertical axis of the head at time $t$ by $(x_1^t, y_1^t, a_1^t)$. The translator from the space of the ellipses to the space of the head’s vertical axis was set to $p_{S_1 \rightarrow S_2}(x_1^t, y_1^t, a_1^t \mid x_2^t, y_2^t, a_2^t) = \delta_{x_2^t, y_2^t, a_2^t}(x_1^t, y_1^t, a_1^t)$, and the translator in the opposite direction was set to $p_{S_2 \rightarrow S_1}(x_1^t, y_1^t, a_1^t \mid x_2^t, y_2^t, a_2^t) = \delta_{x_2^t, y_2^t, a_2^t}(x_1^t, y_1^t, a_1^t) \cdot U(b_t, \theta_t)$, where $U$ stands for a very widely supported uniform distribution. (This makes the integral in (2.12) proportional to $K \left( \frac{s_1^{(n)}(1:3) - s_2^{(j)}(j)}{h} \right)$, where $v(n:m)$ stands for the part of vector $v$ consisting of the $n^{th}$ through the $m^{th}$ elements.) The pseudo-priors of (2.4) in both state spaces were chosen to be uniform, which satisfies (2.7).

The composite tracker yielded better results, as may be seen in Figure 2.14. Now the composite tracker overcame the distractions caused by the edges inside the tracked head’s region.

### 2.7.4 Experiments IV and V: separate objects

In the fourth experiment we tested the combination of two explicit PDF-yielding trackers, tracking separate, albeit related objects. The experiment consisted of tracking the eye locations (in the image plane) of a person taking part in a “smart meeting”. We used a 1:2 down-sampled version of an image sequence from the Fourth PETS Workshop (PETS-ICVS, scenario A, camera 1). We first tracked each of the eyes independently, using a template-based tracker, aided by a background subtraction scheme to locate the left and right margins of the person’s head: A location of high correlation with the eye’s template was considered correct with high confidence only if the location also satisfied requirements regarding the position of the eye within the head boundary. The state
space of each tracker was 2D – the coordinates of the eye’s center. The state was initialized manually, and propagated using a Kalman filter. A few frames with the tracking results imposed are shown in Figure 2.15. Note that after the person rotated his head to his right, the tracker of the person’s right eye failed without recovering.

Since the two tracked states were coupled, their estimations could be combined to yield more robust trackers. We combined the two trackers using only the output as described in Section 2.3.2. The PDF provided by the tracker of the person’s left eye location \((x_{\text{left}}^t, y_{\text{left}}^t)\) was translated into a PDF estimating the location of the person’s right eye \((x_{\text{right}}^t, y_{\text{right}}^t)\), by setting the PDF of the location of the person’s right eye, conditioned on a location of the person’s left eye, to be a Gaussian, centered a few pixels (8 for this sequence) to the left (in the image plane) of the person’s left eye’s center. In other words, the translator was set to

\[
P_{S_{\text{left}} \rightarrow S_{\text{right}}} \left( x_{\text{right}}^t, y_{\text{right}}^t | x_{\text{left}}^t, y_{\text{left}}^t \right) = N \left( \left( x_{\text{left}}^t - 8, y_{\text{left}}^t \right), \left( 5, 0 \right) \right). \]

Consequently, the translated PDF obtained by substituting this PDF into (2.6) is Gaussian, since the translated PDF is a correlation between two Gaussian PDFs. The translation in the other direction was performed symmetrically, and the pseudo-priors of (2.4) for both eyes were set to be uniform (Gaussians of null inverse covariance matrix), which satisfies (2.7). As before, since the translated PDFs and the pseudo-priors in the denominator of (2.3) were Gaussian, both their combinations with the originally provided PDFs and the feedback to the Kalman filters were feasible. The composite tracker was now more robust, recovering from head rotations to both sides. A few representative frames are shown in Figure 2.16.

As in Experiment I, we also used this experiment to test the extent of inaccuracy in
the combined state PDFs caused by setting the pseudo-prior in the denominator of (2.3) to be uniform. We compared a few of the these PDFs to their exact counterparts, which were calculated using the same motion model as in the separate trackers for the prior in the denominator of (2.3) (that is, when not constrained by the black box restriction.) The motion model used was a first order temporal prediction summed with 2D Gaussian noise, dimensions uncorrelated, with standard deviations of 1 pixel in each axis. The differences between the exact PDFs and the approximate ones turned out to be minor (a fraction of a pixel in both expected values and standard deviations.)

The fifth experiment repeated the above after replacing the tracker of the left eye by a CONDENSATION-based tracker. The two trackers were combined using the same translators as in the previous experiment, but as described in Section 2.5. Similar results were obtained.

2.7.5 Experiment VI: separate objects, exemplar-based trackers

This experiment demonstrates the application of the framework to combine probabilistic exemplar-based trackers [117]. First, we implemented the tracker version that had been used for the mouth tracking in [117], and used two instances of it for separately tracking the states of a person’s left eye, $x^l_t$, and right eye, $x^r_t$. The two trackers picked the exemplars ($\{\tilde{x}^l_k\}_{k=1}^{32}$ and $\{\tilde{x}^r_k\}_{k=1}^{32}$ for the left and right eye, respectively) and learned the $\mathcal{M}^2$ kernel parameters and dynamics from two respective training sequences, taken simultaneously. The trained trackers were tested on two new test sequences, one of each eye, taken simultaneously as well. However, the tracker of the right eye was challenged by replacing a couple of sections of its test sequence with noise. The two trackers man-
Figure 2.17. Results of Experiment VI using the two trackers separately. The two upper images in each frame are original test frames, the two middle images are the test frames fed to the trackers, and the two bottom images are the exemplars approximated by the trackers as most probable. (The exemplars were taken from another sequence.) The tracking succeeded during the undisturbed time periods, but failed whenever the image of the right eye was replaced by noise.

Aged to track the eyes’ states by their exemplars, but as expected, the tracker of the right eye failed during the disturbed time periods. The results of a few representative frames are shown in Figure 2.17. The two upper images in each frame are original test frames, the two middle images are the test frames fed to the trackers, and the two bottom images are the exemplars approximated by the trackers as most probable. Normally, a person’s eyes move together in synchronization. This was the case in the training and test sequences here. Therefore, combining the two trackers can potentially overcome the disturbances, and the corresponding failures. Each exemplar-based tracker provided a probability distribution on its set of exemplars, thus making it feasible to combine the two trackers as two explicit PDF-yielding trackers. The set of exemplars of each tracker constitutes its state space. Since each tracker had a different set of exemplars, the combination was performed as in the case of different state spaces (Section 2.3.2). In order to translate the probability distribution from the left eye’s exemplars set to the right eye’s exemplars set, we set the translating probabilities as

\[
\Pr_{S_L \to S_R} (x_t^R = \tilde{x}_j^R | x_t^L = \tilde{x}_k^L) \propto p(I_R | x_t^R = \tilde{x}_j^R), \quad j, k = 1, 2, \ldots, 32,
\]

where \(I_R\) is the frame of the right eye that was taken simultaneously with \(\tilde{x}_k^L\) in the training sequence. The PDF \(p(I_R | x_t^R = \tilde{x}_j^R)\) was estimated according to the learnt \(M^2\) kernel parameters for the right eye. The translation of the probability distribution in
Figure 2.18. Results of Experiment VI after combining the two trackers. The two upper images in each frame are the original test frames, the two middle images are the test frames fed to the trackers, and the two bottom images are the exemplars approximated by the trackers as most probable. (The exemplars were taken from another sequence.) Now the tracking also succeeded when the image of the right-side eye was replaced by noise.

the opposite direction was similarly performed. Both pseudo-priors of (2.4) were set to be uniform over the set of exemplars. This setting was verified to approximately obey (2.7) in the sense that calculating one of the pseudo-priors of (2.4) in terms of the other three distributions in (2.7) yielded an approximately uniform distribution.

Using the same test sequence, we found that the cooperating trackers were powerful enough to overcome the disturbances (Figure 2.18). We would like to point out that these disturbances would not have been overcome by simply using a single instance of the above exemplar-based tracker, the exemplars of which are a spatial concatenation of the left and right eye images. The reason is that this single tracker would have been fed during the disturbances by “half” images. Since the tracker (as implemented in [117]) measured shuffle distances [117] between exemplars and test images, these “half” images would have been enormously distant from all exemplars.

2.8 Conclusion and discussion

A fairly general, probabilistic framework for combining synchronous trackers providing a PDF of the tracked state or a sample-set of it via CONDENSATION was developed. Its main advantage is in its treatment of the separate trackers as black boxes, using only the trackers’ output estimates, which may be modified before their propagation to the
next time step. Consequently, the framework may be used to combine a wide range of trackers, and the combination may be easily performed from the software aspect. The separate trackers may even be of different state spaces of different dimensionality. Thus, a tracker in a low-dimensional state space may be assisted by a tracker in a high-dimensional state space, and vice versa.

Another positive consequence of solely using the trackers’ final estimates is that the combination may be performed in a distributed setting, where each separate tracker runs on a different site and uses different data, while avoiding the need to share the data. Consider, for example, combining Kalman filters, where the only information that needs to be exchanged per time step is a mean vector and a covariance matrix, which is typically much smaller in size than the data itself.

In addition, the approach used here handles combinations of trackers that track a common object and combinations of trackers that track different, related objects, in the same framework. Thus, the suggested framework is also suitable for combining trackers that track separate, albeit related objects, thereby improving the tracking performance of each object.

The framework was successfully tested using various state spaces and datasets.

In addition to the conditions outlined in Section 2.2, it should be noted that the separate trackers’ output PDFs are assumed to be approximately correct in terms of the observations they make. In particular, it is assumed that in a case where the observations made by a tracker contain little information on the state, its likelihood function should be wide. In a case where, despite the lack of information in the observations made by a tracker, it outputs a low variance, miss-centered state PDF, the combination is likely to fail. In other words, the weaknesses of the separate trackers should not result from incorrect modeling of observation likelihoods or evolution models, but may be due to the temporal lack of information in their observations, leading to an indecisive state estimate (but correct in terms of the particular observations performed.) Note that this implies that in the case of combining CONDENSATION-based trackers, a situation of disjoint supported filtering distributions (leading to an undefined PDF) should rarely happen. A useful, related generalization that might be considered for further investigation is downweighting in the combination the PDFs provided by trackers that are observed to be unreliable over time. This might be achieved, for example, by raising the PDF provided by an unreliable tracker to a fractional power.

Another related but more difficult extension of the proposed framework is its generalization for combining asynchronous trackers, i.e., combining trackers where the frames received by the different trackers were not necessarily received at the exact same time instances. In this case, state PDFs have to be interpolated for time instances in-between the time instances at which the frames were taken, so that the combination will be performed using PDFs of states at the same time instance.
Chapter 3

Tracking Object Bitmaps under Very General Conditions

(patent pending)

3.1 Introduction

This chapter is concerned with visual tracking under the very general conditions of a possibly non-rigid target whose appearance may change drastically over time, a 3D scene, general camera motion, and no a priori information regarding the target or the scene except for the target’s bitmap in the first frame, used to initialize the tracker.

Much research has been done in the field of visual tracking, bearing fruit to an abundance of visual trackers, but most trackers were not intended for such a general context. The majority of existing trackers, in order to reduce computational load and enhance robustness, are restricted to some a priori known context. These trackers use some (possibly updatable [57]) appearance model or shape model for the tracked object in the images and they track in a low-dimensional state space of the target’s parameters (e.g., via CONDENSATION [53]). For example, even in [96], in which the tracker is known to be very robust to deformations and changes in appearance, two assumptions are made: that the color histogram of the target does not change very much over time and that the changes in the target’s 2D shape in the image may be approximated by scale alone. As long as the target obeys this model in terms of appearance and shape, the tracker will be robust. However, once the target ceases to obey the model, the tracking is likely to fail without recovery. This might be caused, for example, by heavy partial occlusions that cause drastic changes in the target’s histogram, or by shape deformations that are poorly modeled by scale alone.

The seminal results by Julesz have shown that humans are able to visually track objects merely by clustering regions of similar motion [59]. As an extreme case, consider images (a) and (b) in Figure 3.1. These two images constitute a consecutive pair of images in a video displaying a random-dot object moving in front of a random-dot background that is moving as well. Since the patterns on the object, on the background, and around the object’s enclosing contour are all alike, the object is indistinguishable from the background for the observer who is exposed to these images at nonconsecutive times. However, if these two images are presented one after the other in the same place,
as in a video, the observer is able to extract the object in the two images, shown in (c) and (d).

![Figure 3.1](image)

**Figure 3.1.** (a) and (b) constitute a consecutive pair of images in a video of a random-dot object moving in front of a random-dot background that is moving as well. The object in the two images is shown in (c) and (d).

On the basis of this observation, we propose an algorithm (*Bittracker*) for visual tracking that relies only on three basic visual characteristics:

- **(Short-term) Constancy of Color** – The color projected to the camera from a point on a surface is approximately similar in consecutive frames;

- **Spatial Motion Continuity** – The optical flow in the image region corresponding to an object is spatially piecewise-continuous. That is, the optical flow of the vast majority of the pixels in this area is spatially continuous;

- **Spatial Color Coherence** – It is highly probable that adjacent pixels of similar color belong to the same object.

The first two characteristics usually hold under a sufficiently high frame rate, and the third holds for natural images.

In order to track non-rigid objects of general shape and motion without prior knowledge of their shape, the tracker proposed in this chapter uses the state space of bitmaps to classify whether each pixel in the image belongs to the target (hence the name *Bittracker*). Note that this state space is even more general than the state space of non-parametric contours, since the former may also accommodate for holes in the target (although level set methods also allow topology changes). As no specific target-related, scene-related or camera motion-related assumptions are made, the resulting tracker is suitable for tracking under the aforementioned very general conditions. Moreover, a detailed state of the target is provided, making *Bittracker* suitable for tasks where precise target characterization is required. For example, it may be used to assist in the process of alpha matting by providing an estimation of the desired foreground object.
in the frames. Thus, only refinements of the object mask are required to be manually performed. Note, however, that a lower dimensional target state (e.g., a bounding box) may be easily extracted.

The tracker works by estimating in each frame the MAP (maximum a posteriori) bitmap of the target. The PDF (probability distribution function) of the target’s bitmap in the current frame is conditional on the current and previous frames, as well as on the bitmap in the previous frame. A lossless decomposition of the information in the image into color information and pixel location information allows color and motion to be treated separately and systematically for the construction of the PDF.

One important advantage of the proposed tracker is that the target’s bitmap PDF is marginalized over all possible motions per pixel. This is in contrast to other general-context trackers, which cling to a sole optical flow hypothesis. These trackers perform optical flow estimation—which is prone to error and is actually a harder, more general problem than the mere tracking of an object—or do not use the motion cue at all.

Another advantage of the proposed algorithm over other general-context trackers is that the target’s bitmap PDF is formulated directly at the pixel level (unlike image segments). Thus, the precursory confinement of the final solution to objects composed of preliminarily-computed image segments is avoided.

The rest of this chapter is organized as follows. Related work is reviewed in Section 3.2. Section 3.3 outlines the proposed algorithm. Experimental results are given in Section 3.4, and a summary is given in Section 5.6.

### 3.2 Previous work

Relevant previous work is mainly in the area of video segmentation. However, very few video segmentation algorithms are intended for the very general context discussed here. Most were developed in the context of a stationary camera (e.g., [66, 92, 127]) or under the assumption that the background has a global, parametric motion (e.g., affine [98] or projective [115, 121]). Recently, the last restriction was relaxed to a planar scene with parallax [61]. Other algorithms were constrained to track video objects modeled well by parametric shapes (e.g., active blobs [103]) or motion (e.g., translation [29], 2D rigid motion [115], affine [38, 93], projective [43], small 3D rigid motion [90] and normally distributed optical flow [63, 120]). These algorithms are suitable only for tracking rigid objects or specific preset types of deformations. The algorithm proposed in here, however, addresses the tracking of potentially non-rigid objects in 3D scenes from an arbitrarily moving camera, without prior knowledge other than the object’s bitmap in the first frame.

There are algorithms that address video segmentation and successfully track objects under general conditions as an aftereffect. That is, they do not perform explicit tracking in the sense of estimating a current state conditional on the previous one or on the previous frames. For example, in [106] each set of a few (five) consecutive frames is spatiotemporally segmented without considering the previous results (other than saving calculations). In [71] each frame is segmented into object/background without consid-
ering previous frames or previous classifications. (Furthermore, the classification requires a training phase, upon which the classification is performed, prohibiting major changes in the target’s appearance.) In the contour tracking performed in [55], an active contour is run in each frame separately, while the only information taken from previous frames is the previously estimated contour for initialization in the current frame. In this work, the state (target’s bitmap) is explicitly tracked by approximating a PDF of the current state, which is conditional on the previous state and frame, as well as on the current frame, and by estimating the MAP state. (Of course, conditioning the filtering distribution on the entire history of images would be more precise, but it seems that the very high dimensionality of the state space and the complexity of the state distribution render this impractical.)

Optical flow is an important cue for visually tracking objects, especially under general conditions. Most video segmentation algorithms make a point estimate of the optical flow, usually prior to segmentation (e.g., [29, 38, 43, 63, 79, 83, 90, 94, 120, 121]) and sometimes in conjunction with it (e.g., [93]). An exception is the work in [85] and [86], where motion segmentation is applied to consecutive image pairs by using a tensor voting approach, where each pixel may be assigned multiple flow vectors of equal priority. Since optical flow estimation is prone to error, other algorithms avoid it altogether (e.g., [55, 71, 73, 113, 134]), but these algorithms tend to fail when the target is in proximity to areas of similar texture, and may erroneously classify newly appearing regions with different textures. This is shown in an example in [73], where occlusions and newly appearing areas are prohibited due to the modeling of image domain relations as bijections. Another exception to the optical flow point-estimation is [106], where a motion profile vector that captures the probability distribution of image velocity is computed per pixel, and motion similarity of neighboring pixels is approximated from the resemblance of their motion profiles. In the work here, the optical flow is neither estimated as a single hypothesis nor discarded, but the bitmap’s PDF is constructed through a marginalization over all possible pixel motions (under a maximal flow assumption).

One class of video segmentation and tracking algorithms copes with general object shapes and motions in the context of an arbitrarily moving camera by tracking a non-parametric contour influenced by intensity/color edges (e.g., [113]) and motion edges (e.g., [83]). However, this kind of algorithm does not deal well with cluttered objects and partial occlusions, and may cling to irrelevant features in the face of color edges or additional moving edges in proximity to the tracked contour.

Many video segmentation and tracking algorithms perform spatial segmentation of each frame as a preprocessing step. The resulting segments of homogeneous color/intensity are then used as atomic regions composing objects (e.g., [29, 38, 93]). These algorithms also assign a parametric motion per segment. Rather than confining the final solution in a preprocessing step and making assumptions regarding the type of motion the segments undergo, the algorithm proposed here uses the aforementioned spatial color coherence characteristic and works directly at pixel level.

Two works closely related to this one are [28] and [64]. These two papers deal with foreground/background segmentation in videos. Although the camera in both is
required to be stationary and the input to the algorithm in the second work is enhanced by a synchronous stereo pair of videos, the probabilistic framework is these papers is similar to our approach in several ways. These will be discussed later in the chapter.

Also relevant is the approach in [58] and [36], which, like here, included the estimation of generic pixel masks to describe the shapes of moving objects. Each object’s appearance and mask are learned off-line in an unsupervised manner from an entire input sequence for the purpose of post-processing the input sequence itself. This approach is thus more suitable for batch analysis and editing of video shots. Less object variability is allowed in [58], where the color and transparency of each pixel in an object are modeled as being normally distributed. In [36], greater object variability is accounted for by modeling the template mean (color and transparency) as varying in a low-dimensional affine subspace, where the subspace coordinate is modeled as normally distributed as well. In both [58] and in [36], only simple geometric transformations to the objects and the background are allowed. (In fact, only translation was assumed in the experiments.)

3.3 The Bittracker algorithm

3.3.1 Overview

Every pixel in an image may be classified as belonging or not belonging to some particular object of interest according to the object projected to the pixel’s center point. Bittracker aims to classify each pixel in the movie frame whether it belongs to the target or not. Thus, the tracker’s state space is binary images, i.e., bitmaps. The tracker works by estimating the target’s bitmap at time $t$, given the movie frames at times $t$ and $t - 1$ and the estimate of the previous bitmap at time $t - 1$ ($t = 0, 1, 2, \ldots$). Thus, after being initialized by the target’s bitmap in the first frame (for $t = 0$), the tracker causally propagates the bitmap estimate in an iterative fashion. At each time $t$, the tracker approximates the PDF of the bitmap $X_t$,

$$P(X_t) = \Pr(X_t | I_{t-1}, I_t, X_{t-1}) \quad (3.1)$$

($I_t$ denotes the frame at time $t$), and then estimates the MAP bitmap by maximizing its PDF. The estimated MAP bitmap may then be used to estimate the target’s bitmap at time $t + 1$ in the same way, and so forth. Note that the initializing bitmap $X_0$ need not be exact, as the target’s bitmap may be self-corrected with time using the spatial motion continuity and spatial color coherence characteristics, which are incorporated in (3.1).

When the the bitmap tracking problem is formulated as in (3.1), the solution is targeted directly towards the sought bitmap. Thus, the commonly performed intermediate step of determining optical flow is avoided. This is an important advantage since computing optical flow is a harder, more general problem than estimating the bitmap (given the one in the previous frame).
3.3.2 The bitmap’s PDF

Modeling the bitmap’s PDF (3.1) is very complex. In order to simplify it we consider a discrete image \( I \) not in the usual sense of a matrix representing the pixel colors at the corresponding coordinates, but rather as a set of pixels with indices \( p = 1, 2, \ldots, |I| \) (\(|I|\) denotes the number of pixels in image \( I \)), each one having a particular color \( c^p \) and location \( l^p \) (coordinates). (The pixels are indexed arbitrarily, regardless of their location in the image. To remove any doubt, there is no connection between the indexing of the pixels in \( I \) and the indexing in \( I_{t-1} \). Specifically, if a pixel of index \( p \) in \( I \) and a pixel of index \( p' \) in \( I_{t-1} \) are such that \( p = p' \), it does not imply that the two pixels are related by their colors or locations.) Taking this alternative view, a discrete image \( I \) may be decomposed into the pair \( I = (C, L) \), where \( C = \{c^p\}_{p=1}^{|I|} \) and \( L = \{l^p\}_{p=1}^{|I|} \). Note that no information is lost because the image may be fully reconstructed from its decomposition. This enables us to decompose \( I_t \) into \( I_t = (C_t, L_t) = (\{c^p_t\}_{p=1}^{|I_t|}, \{l^p_t\}_{p=1}^{|I_t|}) \). The bitmap’s PDF (3.1) may now be written as

\[
P(X_t) = \Pr(X_t | I_{t-1}, C_t, L_t, X_{t-1}). \tag{3.2}
\]

We denote the Boolean random variable representing the bitmap’s value at pixel \( p \) in \( I_t \) by \( x^p_t \), which may receive one of the following values:

\[
x^p_t = \begin{cases} 1 & \text{Pixel } p \text{ in } I_t \text{ belongs to the target,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}
\]

Note that the notation \( X_t \) is the abbreviation of \( \{x^p_t\}_{p=1}^{|I_t|} \). The bitmap’s PDF (3.2) is modeled as a Gibbs distribution [39] with a potential consisting of functions of up to two bitmap bits,

\[
P(X_t) \propto \prod_{S \subseteq \{1, \ldots, |I_t|\}, |S| \leq 2} \exp \left\{ -V_S \left( \{x^p_t\}_{p \in S} \right) \right\},
\]

globally conditioned on the variables in the right-hand side of the conditioning line in (3.2). Such modeling, consisting of a multiplicand per pixel and per unordered pixel pair, is neither practical nor necessary. This PDF is later restricted to include multiplicands for all individual pixels and for a small subset of all unordered pixel pairs. That is, the bitmap’s PDF is considered as a conditional random field [68] with respect to some neighborhood system. To simplify the modeling, the bitmap’s PDF is factorized into a product of two simpler functions:

\[
P(X_t) \propto F_1(X_t; I_{t-1}, C_t, X_{t-1}) \cdot F_2(X_t; L_t, I_{t-1}, C_t, X_{t-1}). \tag{3.4}
\]

The first factor consists of all the multiplicands that are independent of the pixel locations \( L_t \). Thus, it may be modeled using only the constancy of color characteristic, without consideration of the spatial color coherence and motion continuity characteristics. The second factor consists of the rest of the multiplicands and is modeled in
light of the latter two visual characteristics. To emphasize that \( \mathcal{X}_t \) is the variable to be estimated where the rest of the variables are fixed and known, we often denote the two functions in the right-hand side of (3.4) by \( F_1 (\mathcal{X}_t) \) and \( F_2 (\mathcal{X}_t) \). As will be seen in what follows, these two components are easier to model due to the separation of the color information from the location information.

**Modeling \( F_1 (\mathcal{X}_t) \)**

Considering only \( \mathcal{L}_t \) as observations, the first factor in (3.4), \( F_1 (\mathcal{X}_t; \mathcal{I}_{t-1}, C_t, \mathcal{X}_{t-1}) \), is a “prior-like” part of the target’s bitmap PDF, which does not depend on the observations. With \( \mathcal{L}_t \) not given, there is no information on the motion from frame \( t - 1 \) to frame \( t \), and no information on the relative position between the pixels in \( \mathcal{I}_t \). Under these circumstances, the dependency of the bitmap’s bits on the pixels’ colors is much stronger than the dependency between the bits themselves. That is, the decision as to whether a pixel belongs to the target can be made mainly by examining its color with respect to the colors of already-classified pixels in the previous frame. Therefore, using only on the pixels’ colors, it is reasonable to associate with the product \( \text{pdf} \) of a single bit \( x_p^t \) inside the product in (3.5) may be marginalized over all the potential correspondences of \( \mathcal{I}_t \)’s pixel \( p \) to pixels \( p' \) in \( \mathcal{I}_{t-1} \), including the event of its correspondence to \( \text{none} \):

\[
f_1 (x_p^t) = \sum_{p' \in \mathcal{N}_t \cup \{\text{none}\}} \Pr (x_p^t | x_{p'}^t \in \mathcal{I}_{t-1}, C_t, \mathcal{X}_{t-1})
\]

where \( \mathcal{N}_t \) denotes the set \( \{1, 2, \ldots, |\mathcal{I}_t|\} \). Note that any hard decision about the optical flow is avoided when this marginalization is applied. Note that this marginalization is similar to the marginalization over all possible disparities along epipolar lines performed in [64].

We model the color of a pixel \( p \) in \( \mathcal{I}_t \) as normally distributed with mean equal to the color of the corresponding pixel \( p' \) in \( \mathcal{I}_{t-1} \) or as uniformly distributed for pixels corresponding to \( \text{none} \). This yields (after a detailed derivation described in Appendix B)

\[
f_1 (x_p^t) \propto (1 - P_{\text{none}}) \cdot \sum_{p' \in \mathcal{N}_t} \Pr (x_p^t | p \rightarrow p', x_{p'}^t) \cdot \frac{1}{|\mathcal{I}_{t-1}|} \cdot N_{c_{p'}} (c_p^t) + P_{\text{none}} \cdot \Pr (x_p^t | p \rightarrow \text{none}) \cdot U(c_p^t),
\]

52
where \( N_{\mu,C} \) is the Normal PDF of mean \( \mu \) and covariance matrix \( C \) (\( C \) is set to an identity matrix scaled by a variance reflecting the degree of color similarity assumed in the constancy of color characteristic), and \( U \) is the uniform PDF on the color space (RGB in our implementation). \( P_{\text{none}} \) is a preset constant that estimates the prior probability of having no corresponding pixel in the previous frame. (\( P_{\text{none}} \) is typically set to 0.1, but as explained in Appendix B, it has only minor influence on the tracker.)

We see that \( f_1(x_p^t) \) may be viewed as a mixture distribution with a component for having a corresponding pixel in the previous frame (with weight \( 1 - P_{\text{none}} \)) and a component for having no corresponding pixel (with weight \( P_{\text{none}} \)).

\[
\Pr \left( x_p^t | p \rightarrow p', x_{t-1}^{p'} \right) \]

is the probability distribution of the bitmap’s bit at a pixel \( p \), when its corresponding pixel in the previous frame, along with its estimated classification bit, are known. Since the MAP bitmap estimated for the previous frame may contain errors, we set this PDF to

\[
\Pr \left( x_p^t | p \rightarrow p', x_{t-1}^{p'} \right) = \begin{cases} 
P_{\text{correct}} & x_p^t = x_{t-1}^{p'}, \quad p' \in N_{t-1}, \\
1 - P_{\text{correct}} & x_p^t \neq x_{t-1}^{p'}, \end{cases} \tag{3.8}
\]

where \( P_{\text{correct}} \) is a preset constant as well. \( P_{\text{correct}} \), which is typically set to 0.9, approximates the prior probability of the estimated bitmap being correct for a pixel. Note that this parameter is equivalent to \( 1 - \nu \) in [28], as there \( \nu \) is the prior probability that a pixel is incorrectly classified.

\[
\Pr \left( x_p^t | p \rightarrow \text{none} \right) \]

is the prior probability distribution of the bitmap’s bit at a pixel \( p \) with no corresponding pixel in the previous frame. This probability distribution is set to

\[
\Pr \left( x_p^t | p \rightarrow \text{none} \right) = \begin{cases} 
P_{\text{target}} & x_p^t = 1 \\
1 - P_{\text{target}} & x_p^t = 0, \end{cases} \tag{3.9}
\]

where \( P_{\text{target}} \) is another preset constant that approximates the prior probability of a pixel, with no corresponding pixel in the previous frame, to belong to the target. (This constant is typically set to 0.4).

While the location information \( \mathcal{L}_t \) is not used at all for deriving (3.7) (as the conditioning is on \( \mathcal{C}_t \) only), in practice we calculate (3.7) with two modifications, using pixel location information in a limited way: First, instead of evaluating pixel correspondences by merely comparing the candidate pixel themselves, as is realized by the Gaussian component in (3.7), we compare small image patches (5 pixels in diameter) centered around the candidate pixels. This is accomplished by modifying the normal and uniform PDFs in Equation (3.7) to products of the color PDFs of the pixels in the patches (see Appendix B for details). This is done in order to make the pixel correspondence distributions less equivocal. (While comparing single pixels might be satisfactory, we found that comparing image patches yielded better results.) Second, we restrict the maximal size of optical flow to \( M \) pixels (in our implementation \( M = 6 \)), and thus compare only image patches that are distanced at most by \( M \) and sum over these correspondences only (137 potential correspondences per pixel), which reduces the number of computations.
We would like to note that, in contrast to an actual optical flow computation consisting of an intermediate step where the patch around each pixel in one image is compared to all nearby patches in the other image (e.g., as in [130] and [111]), the remainder of the optical flow computations are avoided when the marginalization method is used.

Modeling $F_2(\mathcal{X}_t)$

Considering only $\mathcal{L}_t$ as observations, the second factor in (3.4), $F_2(\mathcal{X}_t; \mathcal{L}_t, I_{t-1}, C_t, X_{t-1})$, is an “observation likelihood-like” part of the target’s bitmap PDF, where the pixels’ colors, as well as the previous frame with its corresponding bitmap, are known. Note that unless the image is observed, the pixel locations are random variables with unknown values. (The pixel locations should not be confused with the pixel indices, which are arbitrarily and deterministically ordered.) Given $I_{t-1}$ and $C_t$, PDFs of pixel correspondences between $I_t$ and $I_{t-1}$ are induced (as in $F_1(\mathcal{X}_t)$).

On the basis of these correspondence PDFs, $\mathcal{L}_t$ induces PDFs of optical flow between these two frames. By the spatial motion continuity characteristic, for an adjacent pair of pixels in a region belonging to a single object (where the optical flow is spatially continuous), the discrete optical flow is likely to be the same, and for an adjacent pair of pixels belonging to different objects, it is likely to differ. Thus, the likelihood of an unequal bit-assignment to similarly-moving adjacent pixels should be much lower than an equal bit-assignment, and vice versa for differently-moving adjacent pixels. By the spatial color coherence characteristic, the likelihood of an equal bit-assignment to similarly-colored adjacent pixels should be much higher than an unequal bit-assignment.

Taking this view and noting that $\mathcal{L}_t$ determines pixel adjacency in $I_t$ and pixel motion from time $t - 1$ to time $t$, we associate with $F_2(\mathcal{X}_t)$ multiplicands that depend on pairs of bitmap bits and assume interaction only between bits of pixels that turn out to be adjacent:

$$F_2(\mathcal{X}_t) \propto \prod_{\text{unordered pairs } p_1, p_2 \in \mathcal{N}_t \text{ of adjacent pixels in } I_t} f_2(x_{t}^{p_1}, x_{t}^{p_2})$$

(3.10)

This implies that all the other multiplicands of bit pairs are modeled as constants. We model multiplicands related to adjacent pixels using modeled probabilities of pixel adjacencies and coordinate differences

$$f_2(x_{t}^{p_1}, x_{t}^{p_2}) = \frac{\Pr(\text{adj}(p_1, p_2) \mid x_t^{p_1}, x_t^{p_2}, c_t^{p_1}, c_t^{p_2})}{f_{\text{adj}}(x_t^{p_1}, x_t^{p_2})} \cdot \frac{\Pr(\Delta_t(p_1, p_2) \mid \text{adj}(p_1, p_2), x_t^{p_1}, x_t^{p_2}, I_{t-1}, C_t, X_{t-1})}{f_{\Delta}(x_t^{p_1}, x_t^{p_2})}$$

(3.11)

where $\Delta_t(p_1, p_2) \triangleq l_t^{p_1} - l_t^{p_2}$ and $\text{adj}(p_1, p_2)$ is the event of pixels $p_1$ and $p_2$ being adjacent ($\|l_t^{p_1} - l_t^{p_2}\|_2 = 1$). Thus, the bitmap’s PDF is reduced to a Gibbs distribution with respect to the first-order neighborhood system [39].
We shall begin with the first multiplicand in the right-hand side of (3.11). By Bayes’ rule,

\[
f_{\text{adj}}(x_t^{p_1}, x_t^{p_2}) = p \left( e_t^{p_1}, e_t^{p_2} \mid x_t^{p_1}, x_t^{p_2}, \text{adj}(p_1, p_2) \right) \cdot \frac{\Pr \left( \text{adj}(p_1, p_2) \mid x_t^{p_1}, x_t^{p_2} \right)}{p \left( e_t^{p_1}, e_t^{p_2} \mid x_t^{p_1}, x_t^{p_2} \right)}. \tag{3.12}
\]

We assume no prior information on the object shape and on the object/non-object color distribution. Therefore, the influence of the bitmap bits on \(f_{\text{adj}}(x_t^{p_1}, x_t^{p_2})\) is dominated by the first multiplicand, and thus we approximate

\[
f_{\text{adj}}(x_t^{p_1}, x_t^{p_2}) \propto p \left( e_t^{p_1}, e_t^{p_2} \mid x_t^{p_1}, x_t^{p_2}, \text{adj}(p_1, p_2) \right). \tag{3.13}
\]

Applying the chain rule yields

\[
f_{\text{adj}}(x_t^{p_1}, x_t^{p_2}) \propto p \left( e_t^{p_1} \mid x_t^{p_1}, x_t^{p_2}, \text{adj}(p_1, p_2) \right) \cdot p \left( e_t^{p_2} \mid x_t^{p_1}, x_t^{p_2}, \text{adj}(p_1, p_2) \right). \tag{3.14}
\]

The first multiplicand on the right-hand side does not depend on the bitmap bits, which leaves only the second multiplicand, which we model as

\[
f_{\text{adj}}(x_t^{p_1}, x_t^{p_2}) \propto \begin{cases} U \left( e_t^{p_2} \right) + N_{c_t^{p_1}, C_{\text{adj}}} \left( e_t^{p_2} \right) & x_t^{p_1} = x_t^{p_2} \\ U \left( e_t^{p_2} \right) & x_t^{p_1} \neq x_t^{p_2} \end{cases}. \tag{3.15}
\]

This corresponds to modeling the colors of adjacent pixels as uniformly and independently distributed in the case that they belong to different objects. If these pixels belong to the same object, their color distribution is a mixture of a uniform distribution (corresponding to the case of belonging to different color segments) and a Gaussian in their color difference (corresponding to the case of belonging to the same segment of homogeneous color). \(C_{\text{adj}}\) is assigned an identity matrix scaled by a very small variance, reflecting the variance of the color differences between adjacent pixels belonging to a surface of homogeneous color. (In our implementation the variance was set to 0.01 for each RGB color channel, where the range of each color is \([0,1]\).) We see that for differently-colored adjacent pixels the likelihood is approximately similar for equal and unequal bit-assignments, and for similarly-colored adjacent pixels the likelihood is much higher for equal bit-assignments, which is in keeping with the spatial color coherence characteristic. Equation (3.15) may be used to compute the four likelihoods \(\{f_{\text{adj}}(x_t^{b_1} = b_1, x_t^{b_2} = b_2)\}_{b_1, b_2 \in \{0,1\}}\) (up to a scaling, which is unimportant). Note that the term \(f_{\text{adj}}(x_t^{p_1}, x_t^{p_2})\) causes a bias towards pixel class transitions where the color gradient is high. This effect is similar to the effect of certain corresponding terms in the energy functions in [28] and [64].

We turn now to the second multiplicand on the right-hand side of (3.11), \(f_{\Delta}(x_t^{p_1}, x_t^{p_2})\).
After a detailed derivation, which is given in Appendix C,

\[
\begin{align*}
\Delta (x_i^{p_1}, x_i^{p_2}) &= \left\{ \begin{array}{ll}
P_{\text{flow}_1} \cdot S_1 (x_i^{p_1}, x_i^{p_2}; p_1, p_2) + (1 - P_{\text{flow}_1}) \cdot S_2 (x_i^{p_1}, x_i^{p_2}; p_1, p_2) \\
+ 0.25 \cdot S_3 (x_i^{p_1}, x_i^{p_2}; p_1, p_2),
\end{array} \right.
\quad x_i^{p_1} = x_i^{p_2}, \\
(1 - P_{\text{flow}_2}) \cdot S_1 (x_i^{p_1}, x_i^{p_2}; p_1, p_2) + P_{\text{flow}_2} \cdot S_2 (x_i^{p_1}, x_i^{p_2}; p_1, p_2) \\
+ 0.25 \cdot S_3 (x_i^{p_1}, x_i^{p_2}; p_1, p_2),
\end{align*}
\]

where \( S_1 (x_i^{p_1}, x_i^{p_2}; p_1, p_2) \) is the probability that \( I_t \)'s pixels \( p_1 \) and \( p_2 \) have identical discrete optical flows, \( S_2 (x_i^{p_1}, x_i^{p_2}; p_1, p_2) \) is the probability that they have different discrete optical flows, and \( S_3 (x_i^{p_1}, x_i^{p_2}; p_1, p_2) \) is the probability that at least one of the two pixels has no corresponding pixel in the previous frame (and thus has no optical flow). All these probabilities are conditional on 1) the two pixels’ classification bits; 2) \( C_i \); and 3) the previous frame along with its estimated bitmap. (See Appendix C for the method used to estimate these probabilities.) \( P_{\text{flow}_1} \) is a predefined constant approximating the prior probability that two equally classified, adjacent pixels have similar discrete optical flows (given that the corresponding pixels exist). \( P_{\text{flow}_2} \) is another predefined constant approximating the prior probability that two unequally classified, adjacent pixels have different discrete optical flows. Both constants are typically assigned 0.99.

Examining (3.16), we see that the higher the probability of identical discrete optical flows, the higher the likelihood for \( x_i^{p_1} = x_i^{p_2} \), and vice versa for the probability of different discrete optical flows, conforming to the spatial motion continuity characteristic. When at least one of the pixels has no corresponding pixel in the previous frame, there is no preference for any bit assignments, since the optical flow is undefined.

The final bitmap PDF

The multiplicands in (3.5) and in (3.10) may be written as

\[
\begin{align*}
f_1 (x_i^p) &= c_1 (p, t) x_i^p + c_2 (p, t), \\
f_2 (x_i^{p_1}, x_i^{p_2}) &= c_3 (p_1, p_2, t) x_i^{p_1} x_i^{p_2} + c_4 (p_1, p_2, t) x_i^{p_1} + c_5 (p_1, p_2, t) x_i^{p_2} \\
&\quad + c_6 (p_1, p_2, t),
\end{align*}
\]

where

\[
\begin{align*}
c_1 (p, t) &= f_1 (x_i^p = 1) - f_1 (x_i^p = 0), \\
c_2 (p, t) &= f_1 (x_i^p = 0), \\
c_3 (p_1, p_2, t) &= f_2 (x_i^{p_1} = 1, x_i^{p_2} = 1) - f_2 (x_i^{p_1} = 1, x_i^{p_2} = 0) \\
&\quad - f_2 (x_i^{p_1} = 0, x_i^{p_2} = 1) + f_2 (x_i^{p_1} = 0, x_i^{p_2} = 0), \\
c_4 (p_1, p_2, t) &= f_2 (x_i^{p_1} = 1, x_i^{p_2} = 0) - f_2 (x_i^{p_1} = 0, x_i^{p_2} = 0), \\
c_5 (p_1, p_2, t) &= f_2 (x_i^{p_1} = 0, x_i^{p_2} = 1) - f_2 (x_i^{p_1} = 0, x_i^{p_2} = 0), \\
c_6 (p_1, p_2, t) &= f_2 (x_i^{p_1} = 0, x_i^{p_2} = 0).
\end{align*}
\]
Substituting (3.17) into (3.5) and (3.10), the bitmap’s PDF (3.4) is finally

\[ P(X_t) \propto \prod_{p=1}^{\lvert \mathcal{I}_t \rvert} \left[ c_1(p, t) x_t^p + c_2(p, t) \right] \cdot \prod_{\text{unordered pairs } p_1, p_2 \in \mathcal{N}_t} \left[ c_3(p_1, p_2, t) x_t^{p_1} x_t^{p_2} + c_4(p_1, p_2, t) x_t^{p_1} + c_5(p_1, p_2, t) x_t^{p_2} + c_6(p_1, p_2, t) \right]. \]  

(3.19)

### 3.3.3 MAP bitmap estimation

In order to estimate the MAP bitmap \( X_t^{\text{MAP}} \), (3.19) should be maximized:

\[ X_t^{\text{MAP}} = \arg \max_{X_t} P(X_t). \]  

(3.20)

Since the logarithm is a monotonically increasing function,

\[
X_t^{\text{MAP}} = \arg \max_{X_t} \ln \left( P(X_t) \right) = \arg \max_{X_t} \sum_{p=1}^{\lvert \mathcal{I}_t \rvert} \ln \left( c_1(p, t) x_t^p + c_2(p, t) \right) \\
+ \sum_{\text{unordered pairs } p_1, p_2 \in \mathcal{N}_t} \ln \left( c_3(p_1, p_2, t) x_t^{p_1} x_t^{p_2} + c_4(p_1, p_2, t) x_t^{p_1} + c_5(p_1, p_2, t) x_t^{p_2} + c_6(p_1, p_2, t) \right).
\]  

(3.21)
Due to the fact that the variables in the objective function are 0-1,

\[ \mathcal{X}_t^{\text{MAP}} = \arg\max_{\mathcal{X}_t} \sum_{p=1}^{[T_t]} \left[ \ln \left( \frac{c_4(p, t) + c_2(p, t)}{c_2(p, t)} \right) x_t^p + \ln c_2(p, t) \right] + \sum_{p_1, p_2 \in \mathcal{N}_t} \text{unordered pairs of adjacent pixels in } I_t \\
\ln \left( \frac{c_4(p_1, p_2, t) + c_6(p_1, p_2, t)}{c_4(p_1, p_2, t) + c_6(p_1, p_2, t)} \right) x_t^{p_1} x_t^{p_2} + \ln c_6(p_1, p_2, t) x_t^{p_2} \right] \]

= \arg\max_{\mathcal{X}_t} \sum_{p_1, p_2 \in \mathcal{N}_t} \text{unordered pairs of adjacent pixels in } I_t \\
\ln \left( \frac{f_2(x_t^{p_1} = 1, x_t^{p_2} = 1) \cdot f_2(x_t^{p_1} = 0, x_t^{p_2} = 0)}{f_2(x_t^{p_1} = 1, x_t^{p_2} = 0) \cdot f_2(x_t^{p_1} = 0, x_t^{p_2} = 1)} \right) x_t^{p_1} x_t^{p_2} + \ln \left( \frac{f_2(x_t^{p_1} = 0, x_t^{p_2} = 1)}{f_2(x_t^{p_1} = 0, x_t^{p_2} = 0)} \right) x_t^{p_2} \right] + \sum_{p=1}^{[T_t]} \ln \left( \frac{f_1(x_t^p = 1)}{f_1(x_t^p = 0)} \right) x_t^p. \tag{3.22} \]

After gathering common terms in the resulting polynomial, we obtain

\[ \mathcal{X}_t^{\text{MAP}} = \arg\max_{\mathcal{X}_t} \sum_{p_1, p_2 \in \mathcal{N}_t} \tilde{c}_1(p_1, p_2, t) x_t^{p_1} x_t^{p_2} + \sum_{p=1}^{[T_t]} \tilde{c}_2(p, t) x_t^p, \tag{3.23} \]

where \( \tilde{c}_1(p_1, p_2, t) \) and \( \tilde{c}_2(p, t) \) are the coefficients obtained after the common terms are gathered.

Unfortunately, maximizing quadratic pseudo-Boolean functions is NP-hard [15,65]. Thus, instead of maximizing the objective function in (3.23), we choose to replace each quadratic term \( \tilde{c}_1(p_1, p_2, t) x_t^{p_1} x_t^{p_2} \) with a negative coefficient by the term \( \frac{\tilde{c}_1(p_1, p_2, t)}{2} x_t^{p_1} + \frac{\tilde{c}_1(p_1, p_2, t)}{2} x_t^{p_2} \). The resulting objective function has only nonnegative coefficients for the quadratic terms, and therefore its maximization may be reduced into a maximum-flow problem [42]. (Note that formulating the resulting maximization problem as a minimization of a regular energy function in \( \mathcal{F}^2 \) [65], such as the one in [28], is straightforward.) Note that this heuristic modification to the objective function does not change.
the score contributed by quadratic terms corresponding to equal bit assignments (i.e., where both variables are assigned 1 or both variables are assigned 0). This modification only reduces the score for unequal bit assignments. In order to assess the quality of the approximation, let us denote the objective function in (3.23) by $Q$, the altered objective function by $\hat{Q}$, and its maximizer by $\mathcal{X}_t^{\text{MAP}}$. We then have the following lower bound for the approximation ratio $\alpha$:

$$\alpha \triangleq \frac{Q(\mathcal{X}_t^{\text{MAP}})}{Q(\mathcal{X}_t^{\text{MAP}})} \geq \frac{Q(\mathcal{X}_t^{\text{MAP}})}{\hat{Q}(\mathcal{X}_t^{\text{MAP}}) + D} \geq \frac{Q(\mathcal{X}_t^{\text{MAP}})}{\hat{Q}(\mathcal{X}_t^{\text{MAP}}) + D},$$

(3.24)

where $D = \frac{1}{2} \sum_{\text{adjacent } p_1, p_2 \text{ s.t. } |\tilde{c}_1(p_1, p_2, t)| < 0} |\tilde{c}_1(p_1, p_2, t)|$. Calculating this ratio bound in our experiments showed that the approximated bitmaps are no worse than a $\alpha$-approximations to the MAP bitmaps for $\alpha$ varying in the range between 0.6 and 0.95. Note that these ratios are only lower bounds to the exact ratios. (We could not compute the exact approximation ratios because the exact MAP solutions are inaccessible). We believe that, as implied by the experimental results, the exact ratios are much higher and show that, in practice, the approximated bitmaps are meaningful.

Occasionally the estimated MAP bitmap may contain extraneous small connected components. This may happen after a small patch is erroneously attached to the target (due to very similar color or motion) and then disconnected from it as a set of non-target pixels separating the target from this patch is correctly classified. (In another scenario, the target may actually split into more than one connected component. Note that the bitmap’s PDF does not assume any a priori topological information.) In this case, only the largest connected component in the estimated bitmap is maintained.

### 3.3.4 Considering only target-potential pixels

Since the optical flow between adjacent frames is assumed to be limited by a maximal size $M$, there is no need to solve (3.23) for all the pixels in $\mathcal{I}_t$. Instead, it is enough to solve only for the set of pixels with locations similar to the ones constituting the target in $\mathcal{I}_{t-1}$, dilated with a disc of radius equal to $M$ pixels, and set the bitmap to zero for all other pixels. In other words, $\mathcal{I}_t$ is reduced to contain only the pixels that might belong to the target (see the left-hand diagram of Figure 3.2). The set of pixels in $\mathcal{I}_{t-1}$ that may correspond to the pixels in the reduced $\mathcal{I}_t$ contains the set of pixels with locations similar to the ones in the reduced $\mathcal{I}_t$, dilated with the aforementioned disc. That is, the reduced $\mathcal{I}_{t-1}$ constitutes the target in $\mathcal{I}_{t-1}$, dilated twice with the aforementioned disc (see the right-hand diagram of Figure 3.2). Note that the reduced $\mathcal{I}_{t-1}$ is larger than the reduced $\mathcal{I}_t$, because the latter may include non-target pixels whose corresponding pixels in $\mathcal{I}_{t-1}$ are of locations outside the reduced $\mathcal{I}_t$. Note that changing the pixel-sets $\mathcal{I}_t$ and $\mathcal{I}_{t-1}$ to the corresponding reduced versions affects some normalization constants in the formulae.
Figure 3.2. The reduced $I_t$ and the reduced $I_{t-1}$. In practice, the bitmap is estimated in the reduced image and set to 0 outside of it.

### 3.3.5 Parameters and algorithm outline

As all five $P_*$ parameters are actually prior probabilities and the two others are simply color variances, all these parameters may be set according to their statistics, which may be learned from sequences labeled by the ground truth of each frame’s bitmap and (discrete) optical flow: $P_{\text{none}}$ may be set to the percentage of pixels having no corresponding pixel in the previous frame; $P_{\text{target}}$ may be set to the percentage of the pixels having no corresponding pixel in the previous frame that belongs to the target; $P_{\text{flow}}_1$ ($P_{\text{flow}}_2$) may be set to the percentage of the adjacent pixel pairs with equal (different) bit labeling that have similar (different) discrete optical flows; the variance in $C$ may be set to the variance of the color differences between corresponding pixels; the variance in $C_{\text{adj}}$ may be set to the variance of the color differences between adjacent pixels belonging to a segment of homogeneous color; and $P_{\text{correct}}$ may be adjusted to equal the percentage of correctly classified pixels returned by the tracker.

In our implementation, the noise variance in $C$ was learned from pairs of consecutive frames with known motion. The variance in $C_{\text{adj}}$ was set to that of the low-variance Gaussian component in the distribution of color differences between adjacent pixels. (This Gaussian component corresponds to adjacent pixel pairs belonging to segments of homogeneous color.) However, the five prior probabilities, which may be learned only from long, labeled sequences, might vary across videos. We have therefore set these five prior probabilities in an ad hoc fashion. Nonetheless, because they are prior probabilities of simple events with immediate meanings, setting them ad hoc to reasonable values was possible.

Experimenting with the values of these parameters showed that Bittracker has only minor sensitivity to them, as long as the values are reasonable. This low sensitivity characteristic is reinforced by the fact that all these parameters were set to the exact same values for all sequences, although obviously the true values of these prior probabilities varied across sequences. Note in particular that the effect of $P_{\text{target}}$ is restricted to “new” pixels (pixels that are revealed after being occluded and thus have no corresponding pixel in the previous frame). Therefore, this parameter may be set to zero, and the “new” pixel will be classified in the next frame (where it is no longer “new”) according to the spatial motion continuity characteristic. Note also that $P_{\text{none}}$ has only
minor influence on the tracker, as is explained in Appendix B.

We summarize Bittracker in the outline below. Note that some parts of the algorithm refer to equations given in the appendices B-C. This was done for the sake of readability.

3.4 Experiments

Bittracker was tested on several image sequences, the first two synthesized and the rest natural. All the experiments demonstrate the successful tracking of rigid and non-rigid targets moving in 3D scenes and filmed by an arbitrarily moving camera. As no prior knowledge is assumed regarding the scene or target, and the target’s shape and appearance undergo heavy changes over time (due to deformations, changes in viewing direction or lighting, or partial occlusions), a tracker of a more restricted context such as [96] or [27], which rely on the target’s histogram in the first frame, would not be suitable here. The unsuitability of the mean shift tracker in [27] for some of the sequences is demonstrated.

As the tracker was implemented in MATLAB® (except for a C++ implementation of [17] to solve the maximum-flow problem), the execution was slow. On a personal computer with a Pentium® IV 3GHz processor, the running time was several seconds per frame, depending on the target size.

In all experiments, the parameters were set to the values indicated before, and the tracking was manually initialized in the first frame. Although all the image sequences are colored, they are shown here as intensity images so that the estimated bitmaps, overlaid on top in green, will be clear.

Although all the initial bitmaps were set quite accurately, the tracker’s performance in the case of inaccurate initialization is demonstrated in any sub-sequence beginning in an intermediate frame where the estimated bitmap is inaccurate. An estimated inaccurate bitmap in a frame in the middle of a sequence may be regarded as an initial bitmap for the sub-sequence beginning from this frame. This is true because the tracking is independent of frames and bitmaps at times earlier than the previous frame.

Random-dot sequence

First we tested Bittracker on a random-dot object of gradually time-varying shape and colors moving in front of a random-dot background of gradually time-varying colors that is in motion as well. See Figure 3.1 for the first two frames ((a)-(b)) and the object in each of them ((c)-(d)). Figure 3.3 shows, for a number of frames, the estimated bitmap in green, overlaid on top of intensity version images containing only the target. The background was cut from these images to enable the comparison of the estimated bitmap to the target. It is evident that the tracking in this sequence is very accurate. Note that new object pixels and revealed background pixels are correctly classified, due to the spatial motion continuity characteristic. Such a detailed target localization would not have been possible by using a global feature such as the color histogram in [27], which depends on the entire target.
Input: $I_t, I_{t-1}, \mathcal{X}_{t-1}$.
Output: $\mathcal{X}_t$.

1. $I_t ←$ reduced $I_t; I_{t-1} ←$ reduced $I_{t-1}$.
2. For all pixels $p ∈ I_t$ compute the optical flow distribution $f^1(p'; p, t)$ using (B.5), as well as the two optical flow distributions $f^1_{\text{marginal}}(p', x^p_t; p)$ conditional on $x^p_t = 0$ and $x^p_t = 1$ using (C.11).
3. For each pixel $p ∈ I_t$ compute $f_1(x^p_t = 1)$ and $f_1(x^p_t = 0)$ using (B.8) and (B.9), respectively.
4. For each pair of adjacent pixels (4-neighborhood) $p_1, p_2 ∈ I_t$:
   (a) Compute $f_{\text{adj}}(x^{p_1}_t, x^{p_2}_t)$ for the four possible bit-assignments using (3.15).
   (b) Compute $S_3(x^{p_1}_t, x^{p_2}_t; p_1, p_2)$ for the four possible bit-assignments using (C.13).
   (c) Calculate the bounds on $S_1(x^{p_1}_t, x^{p_2}_t; p_1, p_2)$ for the four possible bit-assignments using (C.15).
   (d) Obtain the four intervals of $f_\Delta(x^{p_1}_t, x^{p_2}_t)$ by substituting $S_2(x^{p_1}_t, x^{p_2}_t; p_1, p_2)$ in (3.16) by the right-hand side of (C.16) and using the results from steps (b) and (c).
   (e) Set the four values of $f_\Delta(x^{p_1}_t, x^{p_2}_t)$ within the corresponding intervals obtained in (d) using Algorithm MINIMIZE.
   (f) Compute $f_2(x^{p_1}_t, x^{p_2}_t)$ for the four different bit-assignments by substituting the results from steps (a) and (e) in (3.11).
5. Obtain the objective function in the right-hand side of (3.22) using the results from steps 3 and 4(f), transform into canonical form (3.23), and replace each quadratic term $\tilde{c}_1(p_1, p_2, t)x^{p_1}_tx^{p_2}_t$ with a negative coefficient by the term $\frac{\tilde{c}_1(p_1, p_2, t)}{2}x^{p_1}_t + \frac{\tilde{c}_1(p_1, p_2, t)}{2}x^{p_2}_t$.
6. Find the bitmap $\mathcal{X}_t^{\text{MAP}}$ maximizing the objective function obtained in previous step as explained in 3.3.3.
7. $\mathcal{X}_t ← \mathcal{X}_t^{\text{MAP}}$ zero-padded into image size.

Summary of Bittracker
We repeated this experiment using the mean shift tracker in [27], which searches in each frame the location in the image whose histogram is most similar to the target’s histogram in the first frame. The tracking failed already in the beginning because the target’s colors change over time and the target and background histograms are quite similar. The tracking results for a few frames are shown in Figure 3.4. It should be noted that the scale adaptation was turned off in this experiment, as the target did not undergo any changes in scale. As it turned out, the tracking failed even under this simplifying condition.

Furthermore, in this sequence, the pixels added to the target during its enlargement phases were assigned colors independently of those of other target pixels. Therefore, trackers of non-parametric target shapes that do not use the motion cue ([73] or [134], for example) would not be suitable here either.

**Random-segment sequence**

Since the random-dot video contains a lot of texture, the optical flow may be estimated with high precision. To test Bittracker on a less favorable (by our algorithm) video, we used a randomly segmented object of gradually time-varying shape, segmentation and colors, moving in front of a randomly segmented background of gradually time-varying colors that is in motion as well. See Figure 3.5 for two sample images and the object as it appears in them. Tracking results are given in Figure 3.6, where the estimated bitmaps are shown in green, overlaid on top of intensity version images containing only the target. As in the random-dot experiment, the tracking here is accurate too. New object segments and revealed background segments are correctly classified due to the spatial motion continuity and the spatial color coherence characteristics.

The time-varying colors of this sequence caused the mean shift tracker in [27] to fail, as can be seen in Figure 3.7. As in the previous sequence, the scale adaptation was turned off, which did not help here either. Moreover, because the segments added to the
Figure 3.4. Tracking results for the Random-dot sequence obtained by the mean shift tracker in [27]. The estimated location is shown on top of intensity version images containing only the target. The tracking failed quite early in the sequence.

Figure 3.5. (a) and (b) are two (nonconsecutive) images from a video of a randomly segmented object of gradually time-varying shape, segmentation and colors, moving in front of a randomly segmented background of gradually time-varying colors that is in motion as well. The object as it appears in the two images is shown in (c) and (d), respectively.
target during its enlargement phases were assigned colors independently of the colors of other target segments, trackers of non-parametric target shapes that do not use the motion cue ([73] or [134], for example) would not be suitable here either.

**Cellotape sequence**

Here we tracked a rotating and moving reel of cellotape filmed by a moving camera. A few frames with the corresponding tracking results are shown in Figure 3.8. The hole in the reel, which was not revealed at the beginning of the video, was revealed and marked correctly as the video progressed. Note that this change in object topology could not have been coped with using a state space of object enclosing contours.

**Man-in-Mall sequence**

In this experiment we tracked a man walking in a mall, filmed by a moving camera. A few frames with the tracking results overlaid are shown in Figure 3.9. Note the zoom-in and zoom-out near the end of the sequence, and the partial occlusion at the end.

**Woman-and-Child sequence**

Here Bittracker was tested on a sequence of a woman walking in a mall, filmed by a moving camera. See Figure 3.10 for a few frames and the corresponding tracking results. Note that the tracking overcame lighting changes and long-term partial occlusions. Since the woman and the girl she takes by the hand were adjacent and walking at similar velocity over an extended time period (beginning around frame #100), the girl was joined to the woman in the tracking process.

As can be seen in Figure 3.11, the mean shift tracker in [27] yielded poor results for this sequence, for two reasons. First, the similarity between the histogram of the target (the woman) and the histogram of the target’s surroundings (the floor) caused the
Figure 3.7. Tracking results for the Random-segment sequence obtained by the mean shift tracker in [27]. The estimated location is shown on top of intensity version images containing only the target. The tracking failed at the beginning of the sequence.

Figure 3.8. Tracking a rotating reel of cellotape filmed by a moving camera. The hole in the reel, which was not revealed at the beginning of the video, was revealed and marked correctly as the video progressed.
Figure 3.9. Tracking a man walking in a mall filmed by a moving camera. Note that Bittracker overcomes the zoom-in and zoom-out near the end of the sequence, as well as the partial occlusion at the end.
the tracker to fail around frame #70. Second, the partial occlusion near the end of the sequence caused the target’s shape to deviate from the shape in the first image of the sequence in a manner that could not be captured by the tracker, which supports only changes in target scale.

Herd sequence

In this experiment Bittracker was tested on one cow running in a herd filmed by a moving camera. A few frames with the tracking results overlaid are shown in Figure 3.12. Note that the tracking overcame a severe partial occlusion.

Lighter sequence

In this sequence we tracked a lighter undergoing general motion, filmed by a moving camera. Figure 3.13 shows a few frames along with the tracking results. Note that the areas of the lighter that were previously occluded by other objects or by the lighter itself are correctly classified upon exposure.

This experiment was repeated using the mean shift tracker in [27]. The full rotation of the target, which is colored differently in each side, caused its histogram to change quite significantly throughout the sequence. In addition, while the target in the first frame was reasonably approximated by an ellipse of some axis ratio and of axes parallel to the image boundaries, such a shape approximation was too crude for the rest of the sequence, where the target’s shape changes are far from being of only isotropic
Figure 3.11. Tracking results for the Woman-and-Child sequence obtained by the mean shift tracker in [27].

Figure 3.12. Tracking a cow in a running herd filmed by a moving camera. Although the tracked object underwent a severe partial occlusion, the tracking continued.
Figure 3.13. Tracking a lighter undergoing general motion and severe partial occlusions as filmed by a moving camera.

scale (as is approximated by the mean shift tracker). Thus, although the target was not lost during the sequence, the tracking was very inaccurate, as is clearly demonstrated in Figure 3.14. Neither were the two aforementioned problems addressed in [23], which suggested an enhanced scale adaptation through the use of negative weights in the kernel.

**Ball sequence**

Here we tracked a ball, initially rolling in front of the moving camera, but then partially occluded by a toy. Results are shown in Figure 3.15. Note the correct classification of areas of the ball that appear during its roll behind the toy.

**Camouflage sequence**

In this last experiment we used the Camouflage sequence from [73]. This sequence was constructed in [73] by cutting a disc-like shape from the textured vegetation in the center of the image and pasting it so as to create apparent motion from the lower left corner to the upper right corner. This can be seen in Figure 3.16, showing Bittracker’s tracking results for this sequence. The results reported in [73] are shown in Figure 3.18. As the moving disc has similar texture to part of the background, distinguishing one from the other is impossible when the former is superimposed on the latter, unless the motion cue is used. The algorithm in [73] does not use motion at all, and thus fails in this sequence. The tracker in [134] is not appropriate for this sequence, for the same reason, which holds also for the mean shift tracker in [27] (with the scale adaptation turned off). The mean shift tracker’s results for this sequence are shown in Figure 3.17. Note that the target’s shape in this sequence is constant and circular (like the support of
Figure 3.14. Tracking results for the *Lighter* sequence obtained by the mean shift tracker.

Figure 3.15. Tracking a rolling ball filmed by a moving camera. Note the occlusion caused by the toy.
the Epanechnikov kernel used by the mean shift tracker) and of a constant histogram. Therefore, no enhancement of the mean shift tracker by updating the target’s color model or by adding degrees of freedom to the target’s shape will compensate for not using the motion cue. In contrast, the algorithm proposed here successfully tracks the moving disc throughout the sequence.

When the tracking results are examined, several types of imperfections are apparent. One type of imperfection is the (usually temporary) misclassification of small target regions as non-target due to specularities, discretization artifacts and sharp spatial lighting changes, which cause the constancy of color assumption to be violated. Nonetheless, most of these errors are corrected with time due to the spatial motion continuity and the spatial color coherence characteristics. A second type of imperfection is the misclassification of thin target or non-target branches, usually thinner than the diameter of the image patches used to estimate the pixel correspondences. Another type of imperfection might occur when a non-target area is exposed after being occluded by the target, and the color of the exposed area is very similar to the color of a nearby target area but very different from all nearby non-target areas. In such a scenario the exposed non-target area might be temporarily classified as belonging to the target. However, these errors are fixed after the erroneously classified non-target regions are disconnected from the ball, and thus eliminated by the single connected component constraint (Section 3.3.3).

Figure 3.16. Bittracker’s tracking results for the Camouflage sequence, constructed in [73] to show the limits of the tracking algorithm proposed there.
Figure 3.17. Tracking results for the Camouflage sequence obtained by the mean shift tracker. The images are ordered from left to right. The tracking succeeds as long as the target's texture is different from nearby background, but as the target passes over a background region of similar texture, the tracker locks on a wrong image area.

Figure 3.18. Tracking results for the Camouflage sequence reported in [73]. (The images are taken from [73].)
3.5 Conclusion

A novel algorithm for visual tracking under very general conditions was developed. The algorithm handles non-rigid targets whose appearance and shape in the image may change drastically, as well as general camera motion and 3D scenes. The tracking is conducted without any a priori target-related or scene-related information (except the target’s bitmap in the first frame, given for initialization).

The tracker works by maximizing in each frame a PDF of the target’s bitmap, formulated at pixel level through a lossless decomposition of the image information into color information and pixel-location information. This image decomposition allows color and motion to be treated separately and systematically. The tracker relies on only three basic visual characteristics: approximate constancy of color in consecutive frames (short-term constancy of color characteristic), spatial piecewise-continuity in the optical flow of pixels belonging to the same object (spatial motion continuity characteristic), and the belonging of similarly-colored adjacent pixels to the same object (spatial color coherence characteristic).

Rather than estimating optical flow by means of a point estimate, we construct the bitmap’s PDF by marginalizing over all possible pixel motions. This is an important advantage, as optical flow estimation is prone to error, and is actually a harder and more general problem than target tracking. A further advantage is that the target’s bitmap PDF is formulated directly at pixel level. Thus, the precursory confinement of the final solution to objects composed of preliminarily-computed image segments, as is common in video segmentation algorithms, is avoided. Experimental results demonstrate Bittracker’s robustness to general camera motion and major changes in object appearance caused by variations in pose, configuration and lighting, or by long-term partial occlusions.

Finally, we note that since the proposed tracker estimates the MAP bitmap conditional only on the previous frame and its corresponding bitmap estimate (and of course also on the current frame), a target that reappears after a total occlusion will not be recovered and the estimated target will remain null. In future work, we may try to use the estimated target locations prior to the total occlusion to build some target model that can be used for target detection.
Chapter 4

Tracking by Affine Kernel Transformations Using Color and Boundary Cues

4.1 Introduction

Kernel-based visual trackers (e.g., [27]) spatially fit a kernel (usually in location and in scale) over the image plane such that an objective function of the aggregate of image features under the kernel support is optimized. The spatial structure of features usually changes over time, so that abstaining from reliance on such a structure helps to make the tracker more robust to these changes. In kernel-based tracking, no attempt is made to find the exact boundary of the target, and all the pixels in the region where the kernel is fitted affect very few kernel parameters (usually only center and scale). Furthermore, no explicit relation between the kernel and the target shape is required, except that the kernel support be roughly where the target is located.

Although kernel-based trackers are robust, they usually transform the kernel only by translation and isotropic scaling. Such a coarse object localization may be of little significance when the change of target’s shape in the image is poorly approximated by these three parameters (consider a rotating stick or a partial occlusion of a significant portion of the target). More seriously, this coarse localization may render the target region such a small portion of the kernel support (or vice versa) that the target will be lost. In addition, a too coarse localization is a major obstacle to updating the target’s appearance model, though this is generally desired.

This chapter tackles the aforementioned problems by proposing a kernel-based tracker that tracks spatial affine transformations. Since affine localization is usually too refined to be identified well by color alone, a spatially aligned pair of kernels, one related to target and background colors and the other to the object boundary, are used in conjunction. The two corresponding spatially normalized kernels are rotationally symmetric with respect to all rotations, which reduces their affine fitting to 5 dimensions (instead of 6). As we demonstrate, the refinement of the target localization to affinities enables us to more safely update the reference color histograms of both the target and
the background, which further enhances the tracker’s robustness.

The chapter proceeds by describing previous work in Section 4.2, followed by a detailed description of the proposed affine kernel fitting tracker in Section 4.3. Section 4.4 presents experimental results and Section 4.5 summarizes the chapter.

### 4.2 Previous work

An early kernel-based visual tracker is the CAMSHIFT [18]. It tracked human faces by assigning each pixel a positive weight reflecting the incidence of the pixel color in a reference color histogram of the face, and finding the location of an axis-aligned rectangular window (the kernel) in which the total weight of the pixels in the window is maximal. The search for the window location was performed quickly via mean-shift iterations [37]. The window localization was repeated several times, each time after isotropically scaling the size of the window according to some function of the sum of the pixel weights inside it, until some heuristic criterion was met. A similar tracking procedure performing a single mean-shift iteration was developed in [77].

A kernel-based tracker of heads was proposed in [10]. In this work a uniform kernel of an elliptical support was used to exhaustively search (in a region of the state space centered at the predicted state) for the location and isotropic scale of the head by comparing the color histogram under the support of the candidate kernel with a reference color histogram. Since the ellipse approximated to the head’s shape in the image quite well, the score could also take into account the gradients along the candidate ellipse, with each gradient projected into the normal to the ellipse at the corresponding pixel. The kernel used in this tracker, as well as in the two aforementioned trackers, was uniform. As will be explained in Section 4.3.1, kernels that decrease from their center, like the ones used in the subsequent references, are better suited for tracking.

A kernel-based tracker minimizing a Bhattacharyya coefficient-based distance between the reference color distribution of the target and the target’s color distribution in the current frame was formulated in [27]. The search for the best kernel location was performed by the mean-shift procedure three times per frame: once with the kernel scale estimated in the previous frame, once with the scale enlarged, and once with the scale reduced. The scale that produced the smallest distance was chosen to be input to an IIR filter used to derive the new scale. Extensions that incorporate background information and Kalman prediction, as well as an application to face tracking, were also proposed. A method for choosing the correct scale for mean-shift blob tracking by adapting Lindeberg’s theory of feature scale selection was proposed in [23]. An extension that updates the reference color histogram by Kalman filtering each histogram bin, followed by hypothesis testing, was suggested in [95].

Kernel-based tracking that minimizes the Matusita distance between the feature distributions using a Newton-style method, as well as the extension to the use of multiple kernels, were developed in [45]. This last work implemented a location and scale tracker as well as a wand tracker. An extension to multiple kernel-tracking where the different trackers collaborate by utilizing the state constraints was proposed in [35].
Instead of fix-point estimation of kernel transformation parameters in the work above, a PDF (probability distribution function) of the transformation parameters (location and isotropic scale) was tracked in [96] via CONDENSATION [53]. This technique was enhanced by fusion with stereo sound or motion detection with a still camera in [97].

Tracking enhancement by using multiple cue modalities has also been performed in the past. In [97] the aforementioned color-based tracker [96] was enhanced by fusion with stereo sound or motion detection with a still camera. A particle filter, in which cascaded Adaboost detections were used to generate the proposal distribution and color histograms were used for the likelihood, was used to track hockey players in [88]. In [105] two trackers, a region tracker and an edge tracker, ran in parallel and performed mutual corrections based on their confidence measures. Another example are the co-inference algorithms, developed in [131] to combine trackers of different modalities.

Another related work is [34,133], where, rather than completely ignoring the spatial structure of the image features in the target, the spatial constraint is only relaxed. This is achieved by modeling the feature-spatial joint probability of a region by a multivariate kernel density estimation.

We would like to note that in addition to the target’s location and scale, the kernel-based trackers in [77] and [18] tried to estimate the vertical and horizontal scale of the target, and the latter also the rotation. However, the estimation was performed after the target localization was completed using a few heuristics. The tracker in [77] was used also in [78], where a mixture-of-Gaussians color model of the target was selectively adapted over time. Finally, we observe that affine object tracking was also performed in [135]. However, in contrast to the work here, the appearance model used there consisted of spatial structure through the use of the spatial-color representation of [34], no boundary-related cues were used, the (implicitly used) kernel was uniform, and no attempt was made to update the target’s appearance model. These have probably limited all the targets in the presented experiments to be rigid or nearly rigid and wholly visible.

4.3 Affine kernel fitting

Using the framework of kernel-based tracking, the tracker proposed here transforms, in each video frame $I(x)$, a spatially aligned pair of (spatially) normalized kernels $K^{\text{color}}(x)$ and $K^{\text{edge}}(x)$, color-related and object boundary-related, respectively. ($x = (x, y)^T$ denotes coordinates.) The transformation parameters $\hat{p}$ are estimated such that a certain scoring function is maximized:

$$\hat{p} = \arg \max_p S^{\text{color}}(I, K_p^{\text{color}}) + \alpha \cdot S^{\text{edge}}(I, K_p^{\text{edge}}),$$

where $S^{\text{color}}$ and $S^{\text{edge}}$ are the color-related and boundary-related score components, respectively; $K_p^{\text{color}}(x)$ and $K_p^{\text{edge}}(x)$ are the transformed kernels with transformation parameters $p$; and $\alpha$ is a parameter regulating the weight of the boundary-related score component.
In the proposed tracker, the geometric transformation applied on the normalized kernels is *affine*. In an affine transformation, the relation between a point \( x \) and its corresponding point \( x' \) in the transformed coordinates may be written as [46]:

\[
x' = R(\phi) D(\lambda_x, \lambda_y) R(\theta) x + t,
\]

where \( R(\gamma) = \begin{pmatrix} \cos \gamma & -\sin \gamma \\ \sin \gamma & \cos \gamma \end{pmatrix} \) is a \( \gamma \)-rotation matrix, \( D(\lambda_x, \lambda_y) = \begin{pmatrix} \lambda_x & 0 \\ 0 & \lambda_y \end{pmatrix} \) is a scaling/reflection matrix, and \( t = (t_x, t_y)^T \) is a translation vector. Now, since the normalized kernels used here are, as usual, rotationally symmetric with respect to all rotations, the second rotation of the coordinate system (corresponding to the multiplication by \( R(\phi) \) in (4.2)) and any reflection (corresponding to a negative \( \lambda_x \) or \( \lambda_y \)) may be omitted by a proper adjustment of the translation vector \( t \). Thus, the problem of estimating the 6 transformation parameters of the kernel pair is reduced to that of estimating the 5 parameters, \( p = (\lambda_x, \lambda_y, \theta, t_x, t_y) \), of the coordinate transformation

\[
T_p(x) = \begin{pmatrix} \lambda_x \cos \theta & -\lambda_x \sin \theta \\ \lambda_y \sin \theta & \lambda_y \cos \theta \end{pmatrix} x + (t_x, t_y)^T, \quad \lambda_x, \lambda_y > 0.
\]

Following the above, we define the transformed version of \( K_{color}(x) \) with transformation parameters \( p \) as

\[
K^\text{color}_p(x) = \lambda_x \lambda_y K_{color}(T_p(x))
\]

and similarly for \( K^\text{edge}_p(x) \). The kernel normalization term \( \lambda_x \lambda_y \), which is the Jacobian determinant of \( T_p \), guarantees that the volume of the transformed kernel will be constant for all transformations. This normalization term is the generalization of the normalization constant \( C_h \) in [27] to affinities. As will be evident in what follows, this normalization is required if the scoring function is to be unbiased with respect to scale.

It is easy to show that the level-sets of the normalized kernels, which are circular, are transformed into *elliptical* level-sets in the transformed kernels. In particular, when the transformation parameters of the transformed kernel are written as

\[
p(l_x, l_y, \theta, o_x, o_y) = \left(2l_x^{-1}, 2l_y^{-1}, -\theta, -2l_x^{-1}(o_x \cos \theta + o_y \sin \theta), 2l_y^{-1}(o_x \sin \theta - o_y \cos \theta)\right),
\]

the circular level-set of unit radius in the normalized kernel is transformed in the transformed kernel into an ellipse with axes of lengths \( l_x \) and \( l_y \), centered at \((o_x, o_y)\) and rotated by \( \theta \) about its center \((l_x\)-length axis parallel to the \( x \)-axis before the rotation).

### 4.3.1 Color-related score component

Denote the discrete color PDFs of the target and the background by \( p^{\text{tar}}(c) \) and \( p^{\text{bg}}(c) \) (\( c \) denotes color), respectively, and the color at pixel location \( x \) by \( I(x) \). When a pixel color is considered as being drawn either from the target’s color PDF or from the
background’s, the probability that a pixel location $x$ belongs to the target is higher than the probability it belongs to the background by

$$\frac{p_{\text{tar}}(I(x)) - p_{\text{bg}}(I(x))}{p_{\text{tar}}(I(x)) + p_{\text{bg}}(I(x))}$$

(assuming equal prior probabilities). This probability difference is used to assign each pixel location a weight

$$w_x = \max \left\{ \frac{p_{\text{tar}}(I(x)) - p_{\text{bg}}(I(x))}{p_{\text{tar}}(I(x)) + p_{\text{bg}}(I(x))}, 0 \right\}. \quad (4.6)$$

(See Section 4.3.1 for the approximated calculation of these color PDFs.) This weight is positive for pixels whose color is more prominent in the target’s color PDF than in the background’s, and the greater the difference in the color’s prominence, the higher the weight. Note that in practice there might be pixels of colors that are not present in either of the two color PDFs ($p_{\text{tar}}(c) = p_{\text{bg}}(c) = 0$), so the denominator in (4.6) should be summed with a small constant to circumvent the possibility of a division by zero.

The color-related score component for transformation parameters $p$ is set to

$$S_{\text{color}}(I, p) = \sum_x K_p^\text{color}(x) \cdot w_x. \quad (4.7)$$

This score has the same form as the objective function derived in [27] (with different weights), where the normalization term $C_h$ in [27] corresponds to the kernel normalization term $\lambda_x \lambda_y$ in (4.4). Note that without this kernel normalization, the color-related score component will grow constantly with the growth in scale.

Since the target is not exactly elliptical and its peripheral pixels are often affected by occlusions or interference from the background [27], the pixel weights (4.6) are expected to be smaller farther from the target center. As the authors of [27] did, we choose the Epanechnikov kernel (which is rotationally symmetric for all rotations and monotonically decreasing in the distance from its center). Then, in order to make the kernel more adaptive to increases in scale, we extend it symmetrically by negative weights. (Such an approach was also used in [23].) Thus, the color-related normalized kernel we use is

$$K_p^\text{color}(x) = \left(1 - \|x\|^2\right) \cdot 1_{\{v: \|v\| \leq 1\}}(x) - \frac{d^2 - (1 + d - \|x\|)^2}{\ln (d + 1) d^3} \cdot 1_{\{v: 1 \leq \|v\| \leq 1 + 2d\}}(x), \quad (4.8)$$

where $d$ is a parameter indicating half the width of the ring-shaped domain where the kernel is negative (see Fig. 4.1). Note that the positive and negative parts of the kernel are of equal volume. In all experiments we set $d = 0.1$.

$^11_A(x) : \mathcal{X} \rightarrow \{0, 1\}$ is the indicator function of subset $A \subseteq \mathcal{X}$. This function equals 1 for $x \in A$ and 0 for $x \notin A$. 

79
Color PDF approximation

The computation of the pixel weights (4.6) requires the discrete color PDFs of the target and the background. These two color PDFs are approximated using the previous video frame and the previously estimated transformation parameters of the kernel pair for that frame. This allows the color PDFs of the target and the background to gradually change over time – which is not possible when using only the approximation of the color PDFs as they appear in the first video frame. Although more sophisticated mechanisms for updating the target model have been proposed (e.g., [57, 78, 95]), the affine target localization achieved here allowed the use of this naive update of the color model. We would like to note that no experimental advantages resulted from using an IIR filter operating also on the histograms estimated before the previous frame.

The target’s color PDF is approximated using the color histogram in the elliptical region corresponding to the positive part of the estimated transformed color-related kernel. As previously explained, the peripheral pixels are the least reliable for estimating the target’s color histogram, and therefore the pixels’ influence on the approximated histogram should decrease with the increase in distance from the target (kernel) center. Thus, we use the Epanechnikov kernel (which was also used in the approximation of the target’s color histogram in [27]), spatially transformed according to the estimated transformation parameters $\hat{p}$ (see Fig. 4.2(a) for the normalized kernel). The approximated color histogram of the target is thus

$$h_{\text{tar}}(c) = C \sum_x \left( 1 - \|T_{\hat{p}}(x)\|^2 \right) \cdot \mathbf{1}_{\{\|v\| \leq 1\}} (T_{\hat{p}}(x)) \cdot \delta(I(x) - c) ,$$  

where $\delta$ is the Kronecker delta function, and $C$ is a scalar normalizing the histogram to unit sum.

Finally, because factors such as noise and illumination variance may cause the pixel colors to drift between consecutive frames, the target’s discrete color PDF $p_{\text{tar}}(c)$ is approximated as a smoothed version of the target’s color histogram $h_{\text{tar}}(c)$. When the
colors are considered as vectors of several color components (e.g., RGB), the discrete domain of the histogram has one dimension per color component (e.g., 3 for RGB). The smoothing is performed by convolving the color histogram with a 1D triangular kernel, normalized to unit sum, in each dimension of the color histogram.

The color quantization into bins performed in [27] may be viewed as a smoothing of the color histogram followed by its sampling. Under this view, that smoothing is not shift-invariant and uses an asymmetric kernel. The histogram smoothing performed here is actually a shift-invariant version of the previous (implicit) smoothing, where here the smoothing kernel is symmetric. This smoothing version simulates the color changes (that result from noise and illumination changes) more appropriately.

The background’s color PDF is approximated in the same manner as the target’s, using the pixels in a region around the target (similar to the region in [27]). On one hand, the closer the pixel is to the target, the less reliable it is for approximating the background’s color PDF. On the other hand, the farther the pixel is from the target, the less relevant it is for approximating the color PDF of the background surrounding the target. We therefore have to use a rotationally symmetric kernel of a ring-shaped support around the target, spatially transformed according to the estimated transformation parameters $\hat{p}$. The normalized kernel we use is (see illustration in Fig. 4.2(b)):

$$
1 - \left( \frac{4}{3} - 9 \| \mathbf{x} \| \right)^2 \cdot 1\{v:1\leq \|v\|\leq 5/3\}(\mathbf{x}).
$$

(4.10)

### 4.3.2 Boundary-related score component

Object boundaries are usually associated with high-magnitude gradients, which are perpendicular to the orientation of the projected object edges. Therefore, denoting by $I_c(\mathbf{x})$ the $c$-th color component at pixel location $\mathbf{x}$ (e.g., $c = 1, 2, 3$ for RGB), we set the
boundary-related score component for transformation parameters $p$ to $^2$

$$S_{\text{edge}}(I, p) = \sum_x K_{p}^{\text{edge}}(x) \sum_c \left\langle \nabla I_c(x), \nabla K_{p}^{\text{edge}}(x) \right\rangle$$  \hspace{1cm} (4.11)

(the caret denotes here normalization to unit length), where the boundary-related normalized kernel is

$$K_{\text{edge}}(x) = \left( 1 - 16 \left( 1 - \|x\| \right)^2 \right) \cdot 1_{\{v:3/4<\|v\|<5/4\}}(x). \hspace{1cm} (4.12)$$

Note that the first summation may consist of only pixel locations where the transformed boundary-related kernel is strictly positive. The boundary-related kernel was constructed such that its highest level-set coincides with the zero-crossing level-set of the color-related kernel. The kernel is illustrated in Fig. 4.3. This score component rewards for color gradients indicating the presence of edges that are in proximity to the boundary of the positive part (the zero-crossing level-set) of the transformed color-related kernel and of orientation similar to the boundary’s. The higher the gradient size, the higher the score; the closer the implied edge to this boundary and the closer this edge direction to the direction of the boundary tangent, the higher the score. This score component may be viewed as a kernel-based extension of the gradient module used in [10].

![Figure 4.3. The boundary-related normalized kernel.](image)

The gradient of the transformed boundary-related kernel, where the kernel is positive, is

$$\nabla K_p^{\text{edge}}(x) = k \cdot \begin{pmatrix} \lambda_x \cos \theta & \lambda_y \sin \theta \\ -\lambda_x \sin \theta & \lambda_y \cos \theta \end{pmatrix} T_p(x), \hspace{1cm} (4.13)$$

$^2$The term $\nabla K_p^{\text{edge}}(x)$ is used here as a means to indicate a perpendicular direction to the tangent of the elliptical level-sets of the kernel. Although such a direction exists at all locations where the kernel is positive, this term is undefined at locations corresponding to the highest level-set (as the gradient there is zero). However, this term is calculated only at locations on the pixel grid, so we neglect the possibility that a pixel location will correspond exactly to a point in this zero-area level-set.
where \( k \) is a scalar (depending on \( x \) and \( p \)), and therefore has no bearing on the calculation of the normalized gradient

\[
\nabla K_{\text{edge}}^p(x) = \frac{\nabla K_{\text{edge}}^p(x)}{\| \nabla K_{\text{edge}}^p(x) \|}. \tag{4.14}
\]

The image gradients \( \nabla I_c(x) \) at pixel locations are estimated using the simple discrete approximations

\[
\nabla I_c(x, y) \triangleq \frac{1}{2} \left( I_c(x + 1, y) - I_c(x - 1, y) \right) - \frac{1}{2} \left( I_c(x, y + 1) - I_c(x, y - 1) \right). \tag{4.15}
\]

Note that the boundary-related score component does not use an edge-related reference model such as an edge orientation histogram [132].

### 4.3.3 Score maximization

In order to estimate the transformation parameters of the kernel pair, the scoring function in (4.1) has to be maximized. After experimenting with several maximization methods, we chose a “coarse-to-fine” search as the best method for finding the global maximum of the scoring function. In this method the 5 parameters \( p_{\text{ellipse}} = (l_x, l_y, \theta, o_x, o_y) \) of the elliptical level-set corresponding to the circular level-set of unit radius in the normalized kernels are sought by a “coarse-to-fine” search in the 5-dimensional space of ellipse parameters: center \( (o_x, o_y) \), rotation \( \theta \), and lengths of horizontal and vertical (before rotation) axes \( l_x \) and \( l_y \), respectively. The search starts from the estimated ellipse in the previous frame, but it may of course start from a different ellipse if object dynamics is incorporated. The kernel transformation parameters \( p \) corresponding to the ellipse parameters \( p_{\text{ellipse}} \) are given in (4.5).

The exact maximization is performed as follows. First, the score is calculated for the estimated ellipse \( \hat{p}_{\text{ellipse}} \) in the previous video frame. Then, the score is calculated for the following 32 ellipses:

\[
\hat{p}_{\text{ellipse}} + (b_1 \cdot \Delta o_x, b_2 \cdot \Delta o_y, b_3 \cdot \Delta \theta, b_4 \cdot \Delta l_x, b_5 \cdot \Delta l_y), \quad b_i = \pm 1,
\]

where \( \Delta o_x = \Delta o_y = 0.5 \Delta l_x = 0.5 \Delta l_y = 4 \) pixels and \( \Delta \theta = 8^\circ \). If the best ellipse (yielding the highest score) out of these 32 ellipses is better than \( \hat{p}_{\text{ellipse}} \), then \( \hat{p}_{\text{ellipse}} \) is changed to the best ellipse and the process is repeated. If not, the search radius is reduced to half by halving the five \( \Delta s \), and the above process is repeated using the reduced \( \Delta s \). When the search radius becomes small enough, the search is terminated. In our implementation the search is terminated when \( \Delta o_x < 0.5 \) pixels, that is, after halving the search radius 4 times.

Experiments show that the search time may be reduced by first coarsely optimizing (until \( \Delta o_x < 1 \)) only the ellipse center and rotation parameters while keeping the axis lengths constant (and equal to the estimated ones in the previous frame), and only then optimizing all 5 parameters as described above. In practice, using this method
the score (4.1) will be typically calculated 200 to 600 times per frame. Note that the color PDFs of the target and the background, the pixel weights (4.6), and the image gradients (4.15) are calculated only once per frame. We would like to point out that gradient ascent was shown in our experiments to require fewer score computations per frame, but it occasionally converged at a local maximum.

**Discussion on CONDENSATION as an alternative:** An alternative to the above score maximization may be the tracking of a probability distribution function over the space of kernel transformation parameters via CONDENSATION. This would allow multi-modal distributions to be dealt with. However, using CONDENSATION here would make the update of the target’s color PDF too computationally heavy, since each particle (hypothesis) would have a different color PDF associated with its corresponding target. Moreover, even with a single, unchanged color PDF for the target, each particle would have a different color PDF for the background surrounding the target associated with it. This would require that the background’s color PDF and the pixel weights, which depend on it, be estimated separately for each particle. Finally, even if the last computational problem would be solved, experiments show (e.g. [53]) that using CONDENSATION for tracking a shape in the space of affinities requires 100-1200 particles, which is in the same order of magnitude as the number of score computations performed per frame in the maximization method above.

### 4.4 Experimental results

Experiments using four image sequences are discussed below. In all but one sequence the color space used was RGB with 128 equally spaced values in [0,1] in each color band. In Sequence II the color PDFs were estimated over the HSV color space (with the same dynamic range and discretization as in the RGB color space), since this allowed for better separation of target from background colors in this sequence. In all experiments with the proposed tracker, the support size of the 1D triangular kernel used to smooth the color histograms was 9. The boundary-related weight parameter $\alpha$ (Eq. (4.1)) was set in Sequences I-III to 1. Since the target’s shape (in the image) in Sequence IV is very close to an ellipse, better results for this sequence were obtained by raising $\alpha$ to 2. The initialization was manually performed in the first frame. The estimated ellipses are presented with each of their semiaxes enlarged by 3 pixels so that the target boundary will be clear.

**Sequence I:** The proposed tracker was tested on an image sequence, where both the target (a lighter) and the camera are arbitrarily translated and rotated. The target was successfully tracked (see Fig. 4.4).

The same experiment was repeated using the mean shift tracker [27]. Results are shown in Fig. 4.5. The shape of the target in the first frame is indeed approximated reasonably by an ellipse of some axis ratio and of axes parallel to the image boundaries. However, such a shape approximation is too crude for the rest of the sequence, as the target’s shape changes throughout the sequence are far from being only of isotropic scale. Moreover, since the target is colored differently on different sides, its color PDF
changes as it rotates. These two obstacles cause the poor performance of the mean shift tracker in this experiment.

To demonstrate the importance of updating the target’s color PDF, the proposed tracker was run again from the middle of this sequence, with the update mechanism of the target’s color PDF turned off. That is, the target’s color PDF, which is used to calculate the pixel weights (4.6), remained equal to the one estimated in the first frame from which the tracker was run. Due to the rotational motion of the target, the target’s colors in this frame are very distinct from those in the subsequent frames, which causes the tracking failure (see Fig. 4.6).
Figure 4.6. Results of the proposed tracker without updating the target's color PDF for a sub-sequence of Sequence I. Since the target colors that are revealed to the camera change with time, the tracking fails. (Only small, magnified portions of the frames are shown so that the target colors will be clearly seen.)

Figure 4.7. Results of the proposed tracker for Sequence II.

**Sequence II:** Here the proposed tracker was tested on a non-rigid target (two people walking in a mall), filmed by a moving camera. The tracking succeeded, as may be seen
in Fig. 4.7.

As in the previous sequence, the mean shift tracker [27] did not perform well here (see Fig. 4.8). The main reason for this is that the target shape changes from a vertical blob at the beginning of the sequence to a horizontal one near the end. As the mean shift tracker supports only isotropic scale changes, the former change in target shape could not be captured, which eventually caused the loss of the target.

![Frame 1](image1.jpg) ![Frame 70](image2.jpg) ![Frame 185](image3.jpg) ![Frame 200](image4.jpg)

**Figure 4.8.** Results of the mean shift tracker [27] for Sequence II.

**Sequence III:** In this sequence, filmed by a moving camera, a walking person is tracked by the proposed tracker. The tracking is robust until the person leaves the scene (see Fig. 4.9).

Note that the person’s legs were left out from the estimated target as the sequence progressed. This is caused by the significant changes in the target’s shape near the leg area. These shape changes are too great to be reasonably approximated by an affinity of the approximated ellipse of the first frame. A kernel transformation that would have included the person’s legs inside the positive part of the color-related kernel (and hence inside the ellipse), would have also included large background areas, which would have lowered the score too much.

**Sequence IV:** To test the proposed tracker on a target of topologically (in the image plane) changing shape, we used a sequence consisting of a roll of cellotape rotating in front of an arbitrarily moving camera. The successful tracking is shown in Fig. 4.10.

**Disregarding of Edges:** Lastly, in order to demonstrate the contribution of the boundary-related kernel to the tracking, we repeated the above experiments without it ($\alpha = 0$). Several results are presented in Fig. 4.11. These tracking failures are a strong indication of the importance of the boundary-related kernel in the proposed tracker.
4.5 Conclusion

A new kernel-based visual tracker was proposed. The proposed tracker is enhanced with respect to previous kernel-based trackers in two ways. First, in addition to the constancy of color exploited by other kernel-based trackers, this tracker exploits the presence of color edges along the target boundary. Second, it spatially adjusts the kernels using affine transformations instead of using merely translation and isotropic scaling, to which other kernel-based trackers are restricted. These two enhancements make the target localization more accurate. Moreover, the refined target localization facilitates safer updating of the target’s reference color PDF, further enhancing the tracker’s robustness.

The tracking is performed by estimating the affinity of a spatially aligned pair of kernels, one color-related and the other object boundary-related, such that a scoring function is maximized. As was shown, the rotational symmetry of the spatially normalized kernel pair reduces the (normally 6-dimensional) space of affinities into a 5-dimensional one.

Experiments using several challenging image sequences demonstrate the high capability of the proposed tracker and its advantage over other kernel-based trackers.
Figure 4.10. Results of the proposed tracker for Sequence IV.

Figure 4.11. Results obtained using the proposed tracker without its boundary-related kernel ($\alpha = 0$). It is evident that color edges play a crucial role in the proposed tracker.
Chapter 5

Mean Shift Tracking with Multiple Reference Color Histograms

5.1 Introduction

The target’s color histogram is widely used for visual tracking (e.g., [10, 23, 27, 95, 96]) and, as was shown in [26, 27], tracking using this feature may be performed very quickly via the Mean Shift procedure [25]. This chapter extends Comaniciu et al.’s tracker in [26, 27], which will be referred to here by its common name Mean Shift tracker.

The Mean Shift tracker works by searching in each frame for the location of an image region whose color histogram is closest to the reference color histogram of the target. The distance between two histograms is measured using their Bhattacharyya coefficient, and the search is performed by seeking the target location via Mean Shift iterations beginning from the target location estimated in the previous frame (the tracker is outlined in Section 5.2).

In the Mean Shift tracker, as well as in the other trackers cited previously, the reference color histogram is approximated according to a single view of the target, typically as it appears in the first frame of the sequence. Although using this method for obtaining the reference histogram proved to be very robust in many scenarios, it produces, in many cases, a poor representation of the target, which might result in poor tracking. More dangerously, the support of a reference histogram obtained by this method may become non-overlapping with the support of the target’s histogram as it appears in the sequence, usually resulting in target loss. Indeed, for many objects, any viewing direction may be replaced with a different viewing direction where all the object’s colors apparent in the latter view differ from those in the former. (An (unscrambled) Rubik Cube is an extreme example of such an object; each side is a different color, and three sides at most are visible from any viewing direction.) Major changes in the apparent colors of a target may also result from changes in the actual target’s colors, as when a person puts on or removes a piece of clothing.

Often, several different views of the target are available prior to the tracking, either from images that were previously acquired (e.g., [6, 11, 67, 122]) or when performing off-
line tracking (e.g., [19,112,114,129]). In these contexts, this work extends the the Mean Shift tracker to using multiple reference color histograms. At first we suggest a simple method to combine these histograms into a single histogram that is more appropriate for tracking the target. In order to enhance the tracking further, we then propose an extension to the Mean Shift tracker, where the convex hull of these histograms is used as the target model. That is, rather than searching for the image region whose color histogram is closest to a single reference histogram, we search for the image region by minimizing the distance of its color histogram from the convex hull of several reference histograms.

Many trackers have modeled the target’s 2D appearance as being time-varying within a subspace (e.g., [11, 44, 47, 67]). In these trackers, the search for the target (and possibly for additional transformation parameters) is performed by minimizing the distance of its appearance in the current frame from that subspace. This idea is applied here by modeling the target’s color histogram as being a time-varying linear combination of several reference histograms, under the restriction that the mixture coefficients are nonnegative and sum to unity (so that the linear combination will be a histogram mixture).

Time-varying histograms of colors (e.g., [78, 95]) or of other features such as filter responses (e.g., [57]) have been used for target modeling before. This work extends the Mean Shift tracker to use a target model consisting of a color histogram that is time-varying within the convex hull of several histograms acquired prior to the tracking process. A related work is [122], where the target model employed by the Mean Shift tracker was extended to use mixtures of histogram pairs. The tracking there was performed by alternating between target location optimization using one measure (the Bhattacharyya coefficient) and model parameter optimization using a second measure (an observation likelihood). Thus, unlike the work here (and the original Mean Shift tracker), no definite measure was minimized during the tracking, and there was no guaranteed stopping criterion for these optimization iterations.

Section 5.2 outlines the original Mean Shift tracker [27]. A simple method for combining multiple reference histograms into one is proposed in Section 5.3. Section 5.4 describes the proposed extension of the Mean Shift tracker to use the convex hull-based target model. Experimental results are described in Section 5.5, and a summary is given in Section 5.6.

5.2 The Mean Shift tracker

In this section we outline the Mean Shift tracker described in [27]. The notations used here are similar to those in [27], with minor modifications to suit the subsequent sections.

Let \( \hat{q} = \{\hat{q}_u\}_{u=1}^m \) be an \( m \)-bin reference color histogram of the target. The center point \( y \) of the image region in which the color histogram \( \hat{p}(y) = \{\hat{p}_u(y)\}_{u=1}^m \) is closest to \( \hat{q} \) is sought in each frame of the image sequence. The metric used in [27] for measuring the distance between the hist-

91
The image region used for calculating the color histogram is elliptical.\(^1\) The elliptical region is specified in one (typically the first) frame in the sequence, and the reference color histogram is calculated using this region. Then, with the ellipse motion approximated as translational, the center of this elliptical region is tracked. Note that approximating the motion of the ellipse (which encloses the tracked image region) as translational does not imply that the motion of the region itself is only translational. A simple mechanism for adapting also to isotropic scale changes of the region was proposed in [27]. This scale adaptation is carried out by running the Mean Shift tracker for three different scales – the scale estimated in the previous frame, an enlarged scale, and a reduced scale – and then choosing the one that produced the target location in which the color histogram is closest to the reference histogram. The chosen scale is then input into an IIR filter for producing the final scale. This mechanism operates on top of the Mean Shift tracker and it may be applied exactly in the same manner using the algorithms proposed here. Therefore, for simplicity, the target scale in this work is assumed to be fixed.

The pixel locations are normalized such that the specified reference ellipse will become the unit circle centered at the origin. (This is achieved by rotating, non-isotropically scaling, and translating the coordinate system of the image.) The normalized locations of the pixels inside the reference ellipse are denoted \(\{x^*_i\}_{i=1,...,n}\), and the reference color histogram is computed as

\[
\hat{q}_u = C \sum_{i=1}^{n} k(\|x^*_i\|^2) \delta \left[ b(x^*_i) - u \right],
\]

where \(b(x)\) is the bin number (1, \ldots, \(m\)) associated with the color at the pixel of normalized location \(x\), \(\delta\) is the Kronecker delta function, \(k(x)\) is a kernel profile that assigns smaller weights to pixels farther from the circle center, and \(C\) is a constant that normalizes the histogram to be of unit sum. Similarly, the normalized locations of the pixels inside an ellipse centered at a candidate normalized location \(y\) are denoted \(\{x_i\}_{i=1,...,n}\), and the color histogram in the corresponding region is thus

\[
\hat{p}_u(y) = C \sum_{i=1}^{n} k(\|y - x_i\|^2) \delta \left[ b(x_i) - u \right].
\]

It is assumed here that both elements of \(y\) correspond to integer numbers of pixels. Otherwise, \(n\) and \(C\) may fluctuate according to the exact inter-pixel location of the ellipse center.

\(^1\)The ellipse in [27] had axes parallel to the image border, but of course generalizing to account for rotated ellipses is performed by simply rotating the coordinate system of the image.
The Epanechnikov kernel was used in [27], and we use it here as well. The profile of this kernel is
\[
k(x) \propto \begin{cases} 
1 - x, & 0 \leq x \leq 1; \\
0, & x > 1.
\end{cases}
\] (5.5)

5.2.1 Target localization via Mean Shift

The goal in each frame of the sequence is to estimate the target translation \( \hat{y} \) that minimizes the distance \( d_{\hat{q}}(\hat{y}) \) between its corresponding histogram \( \hat{p}(\hat{y}) \) and the reference histogram \( \hat{q} \). This is equivalent to estimating the target translation \( \hat{y} \) that maximizes the Bhattacharyya coefficient (5.2) between these two histograms. Denote by \( \hat{y}_0 \) the estimated target location in the previous frame. Approximating the Bhattacharyya coefficient (5.2) in the current frame by its first-order Taylor expansion around the values \( \hat{p}_u(\hat{y}_0) \) and substituting (5.4) for \( \hat{p}(y) \) results in
\[
\rho(y) \approx C_{\hat{q}\hat{y}_0} + C \sum_{i=1}^{n} w_i k\left( \|y - x_i\|^2 \right),
\] (5.6)

where
\[
w_i = \frac{m}{\sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u(\hat{y}_0)}{\hat{p}_u(\hat{y}_0)}} \delta [b(x_i) - u]}
\] (5.7)

and \( C_{\hat{q}\hat{y}_0} \) is independent of \( y \).

The search for \( \hat{y} \) in the current frame starts at the estimated target location in the previous frame, i.e., initially \( \hat{y} \leftarrow \hat{y}_0 \). Now, the \( y \)-dependent term in the right-hand side of (5.6) may be viewed as a kernel density estimate computed with kernel profile \( k(x) \) at \( y \), with the samples \( x_i \) being weighted by \( w_i \) (5.7). As proved in [25], if the weights \( w_i \) are nonnegative and the kernel profile \( k(x) \) is monotonically non-increasing and convex (all these requirements are fulfilled (5.5,5.7)), then a higher density value is reached by shifting \( \hat{y} \) from \( \hat{y}_0 \) to the mean of the sample, which is weighted by the kernel whose profile is \( g(x) = -k'(x) \) and which is centered at \( \hat{y}_0 \).
\[
\hat{y} \leftarrow \frac{\sum_{i=1}^{n} x_i w_i g\left( \|\hat{y}_0 - x_i\|^2 \right)}{\sum_{i=1}^{n} w_i g\left( \|\hat{y}_0 - x_i\|^2 \right)}.
\] (5.8)
(An exception is a shift by zero, indicating that the estimated location is already at a density mode.) Moreover, iteratively repeating the Mean Shift (5.8), each time replacing \( \hat{y}_0 \) by \( \hat{y} \) calculated at the previous iteration, will result in the convergence of \( \hat{y} \) to a density mode. (Although rarely, it might happen that an application of (5.8) will decrease the true \( \rho(y) \), since (5.6) is only an approximation of it.)

In practice, the weights \( w_i \) (5.7) are recomputed after each Mean Shift iteration (5.8). Observe that for the Epanechnikov profile (5.5), the Mean Shift iteration (5.8) reduces to a simple weighted average.
5.3 Combining multiple histograms into one

Sometimes no view of the target yields a reasonable approximation of its circumferential color histogram. An extreme example is presented in Fig. 5.1. This figure shows the results of the Mean Shift tracker for Sequence I, where a Rubik Cube is tracked. The reference color histogram was set in the first frame, where the visible colors of the cube are orange, blue and yellow. As the cube rotates, different sets of colors become visible, causing the large deviations of the estimated locations from the target center. At Frame 475 none of the originally visible colors of the target remain visible, and so the tracking fails. Each side of the cube is a different color, and three sides at most may be visible from any viewing direction. Therefore, this problem could not be generally solved by using a different viewing of the target to model its reference color histogram.

One possible solution to the above problem is to stay with a single reference histogram, but set it using multiple color histograms obtained from different target views. Assume we are given $M$ views of the target and denote the color histograms visible from these views by $\hat{q}^v = \{\hat{q}^v_u\}_{u=1,...,m}$, $v = 1, \ldots, M$. Since any of these target views may appear during the tracking process, the maximal reference histogram’s distance from all the given histograms should be as small as possible. To this end, the following minimax optimization problem has to be solved for $\hat{q}$:

$$\begin{align*}
\text{minimize} \quad & \max_u \sqrt{1 - \rho [\hat{q}, \hat{q}^v]} \\
\text{subject to} \quad & \hat{q}_u \geq 0, \quad u = 1, \ldots, m \\
& \sum_{u=1}^{m} \hat{q}_u = 1.
\end{align*}$$

(5.9)
The solution to this problem is similar to that of the problem:

\[
\text{maximize } \min_v \sum_{u=1}^{m} \sqrt{\hat{q}_u q_u^v} \\
\text{subject to } \hat{q}_u \geq 0, \quad u = 1, \ldots, m \quad (5.10)
\]

\[
\sum_{u=1}^{m} \hat{q}_u = 1.
\]

The feasible set of this maximization problem is convex. Likewise, each term in the sum in the objective function is concave, and concavity of functions is preserved under addition and pointwise minimization, so that the whole objective function is concave. Therefore, (5.10) is a convex optimization problem, and thus can be solved efficiently [16].

To test the above method for setting the reference color histogram, the Mean Shift tracker was tested again on Sequence I, this time using the reference color histogram (5.9) obtained from two different views of the target \((M = 2)\). One view is the one used in the previous experiment (Frame 1 in the sequence), and the other view is from the opposite direction. These two views and an illustration of the computed final histogram \(\hat{q}\) are shown in Fig. 5.2. The tracking results are presented in Fig. 5.3. The improved accuracy with respect to Fig. 5.1 is clearly evident. Moreover, the tracking failure at Frame 475, where all the initially visible target colors were occluded, was resolved here.

![Figure 5.2](image-url)

**Figure 5.2.** The two target views used for approximating the two reference histograms (a) \(\hat{q}_1\) and (b) \(\hat{q}_2\). These two reference histograms were used to set the final reference histogram \(\hat{q}\) according to (5.9). The latter histogram is illustrated in (c) by an image with an identical color histogram.

Despite the higher accuracy and robustness achieved by this method in the sequence, a final reference model consisting of a single histogram is still a mediocre representation of the true, time-varying histogram of the target. An example of this may be seen in Frame 500 in Fig. 5.3. In this frame, an image region consisting of
three different colors, all present in the final reference histogram \( \hat{q} \), has a color histogram closer to \( \hat{q} \) than the histogram at the correct image region, which consists of only two colors, both present in \( q \).

The above method of setting the reference histogram has an additional drawback: the need to solve (although only once) the optimization problem (5.10) before the tracking begins. In this work we used the RGB color space, where each color band was equally divided into 8 bins. This yielded \( m = 8^3 = 512 \) color bins in all. Solving the convex optimization problem (5.10) with this number of variables took several seconds using \texttt{cvx} [41].

5.4 Convex hull-based target model

As different sides of the target face the camera, the target’s histogram changes. To accommodate for a time-varying target histogram, we propose to extend the reference target model used by the Mean Shift tracker to include the convex hull of multiple reference histograms obtained from different target views. That is, the target model is approximated as the mixture of \( M \) reference histograms

\[
\hat{q}(\alpha) = \sum_{v=1}^{M} \alpha_v \hat{q}^v, \quad \forall \alpha_v \geq 0, \quad \sum_{v=1}^{M} \alpha_v = 1,
\]

(5.11)

where the mixture proportions \( \alpha = \{\alpha_v\}_{v=1...M} \) vary with time.

Thus, the tracking process now consists of finding in each frame the target translation \( \hat{y} \) that minimizes the minimal distance

\[
\min_{\alpha} d_{\hat{q}(\alpha)}(\hat{y})
\]
between \( \hat{p}(\hat{y}) \) and the set of all histogram mixtures \( \hat{q}(\hat{\alpha}) = \{ \hat{q}_u(\hat{\alpha}) \}_{u=1}^m \). Although the previous Bhattacharyya-based distance (5.1)-(5.2) may be minimized here as well, tracking using the convex hull-based target model provided better experimental results by minimizing the sum of absolute difference (SAD),

\[
d_{\hat{q}(\hat{\alpha})}(y) = \sum_{u=1}^m |\hat{p}_u(y) - \hat{q}_u(\hat{\alpha})|.
\]  

(5.12)

As in (5.1)-(5.2), the measure (5.12) imposes a metric structure as well.

Like the search for the current \( \hat{y} \) in the original Mean Shift tracker (Section 5.2.1), the search for the current target location is performed here by the iterative reduction of (5.12) via the iterative modification of the target location and mixture proportions \( (\hat{y}, \hat{\alpha}) \), beginning from the target location and mixture proportions estimated in the previous frame, i.e., initially \( (\hat{y}, \hat{\alpha}) \leftarrow (\hat{y}_0, \hat{\alpha}_0) \). The minimization of (5.12) is performed by successive repetitions of the following two steps:

1. shifting the estimated target location \( \hat{y} \) such that (5.12) is reduced, while keeping the estimated mixture proportions \( \hat{\alpha} \) fixed; and

2. minimizing (5.12) with respect to the mixture proportions \( \hat{\alpha} \), while keeping the estimated target location \( \hat{y} \) fixed.

As in the reduction of the distance (5.1) in the original Mean Shift tracker, the reduction of the distance (5.12) in Step 1 above is performed via a Mean Shift iteration. Because the minimization problem in Step 2 is convex, it may be performed rapidly as well. These two steps are described in detail in the following two subsections.

Since (5.12) is bounded from below and is reduced in each of the two steps, the sequence of distances obtained by the successive application of these steps is guaranteed to converge. In practice, these two steps are repeated until the shift of the estimated target location is smaller than half a pixel (this criterion is very similar to the one in the original Mean Shift implementation [27]) and each of the mixture proportions is modified by less than 0.01. As in the original Mean Shift implementation, the above pair of steps usually has to be performed only a few times per frame until these two criteria are met.

### 5.4.1 Step 1: shifting \( \hat{y} \)

In this step, the distance \( d_{\hat{q}(\hat{\alpha})}(y) \) in (5.12) has to be reduced by shifting the estimated target location \( \hat{y} \) from the location obtained after the previous application of Step 1 (or the one obtained in the previous frame, if this is the first application of Step 1 in the current frame), while keeping the estimated mixture proportions \( \hat{\alpha} \) unchanged. As \( \hat{\alpha} \) is fixed in this step, the histogram mixture \( \hat{q}(\hat{\alpha}) \) will be denoted here in shorthand by \( \hat{q} \).

For notational compatibility with the previous Mean Shift derivation (Section 5.2.1), let us denote the estimated target location obtained after the previous application of Step 1 by \( \hat{y}_0 \).
Since the metric used here (Eq. (5.12)) is different from the one used in [27], the minimization procedure should be suitably adapted. Reducing $d_\hat{q}(y)$ is equivalent to increasing $-d_\hat{q}(y)$. Approximating $-d_\hat{q}(y)$ by its first order Taylor expansion around the values $\hat{p}_u(\hat{y}_0)$ yields\(^2\)

\[
-d_\hat{q}(y) \approx -d_\hat{q}(\hat{y}_0) + \left( \begin{array}{c} \text{sgn} (\hat{q}_1 - \hat{p}_1(\hat{y}_0)) \\ \vdots \\ \text{sgn} (\hat{q}_1 - \hat{p}_1(\hat{y}_0)) \end{array} \right)^T \left( \begin{array}{c} \hat{p}_1(y) - \hat{p}_1(\hat{y}_0) \\ \vdots \\ \hat{p}_m(y) - \hat{p}_m(\hat{y}_0) \end{array} \right) = C'_{q,\hat{y}_0} + \sum_{u=1}^{m} \hat{p}_u(y) \text{sgn} (\hat{q}_u - \hat{p}_u(\hat{y}_0)), \quad (5.13)
\]

where $C'_{q,\hat{y}_0}$ is independent of $y$ and $\text{sgn}(\cdot)$ is the sign function. Substituting (5.4) for each $\hat{p}_u(y)$ results in

\[
-d_\hat{q}(y) \approx C'_{q,\hat{y}_0} + C \sum_{i=1}^{n} w_i k (\|y - x_i\|^2), \quad (5.14)
\]

where

\[
w_i = \sum_{u=1}^{m} \text{sgn} (\hat{q}_u - \hat{p}_u(\hat{y}_0)) \delta [b(x_i) - u]. \quad (5.15)
\]

The $y$-dependent term in this approximation to the objective function $-d_\hat{q}(y)$ is of similar form as the one in (5.6), but with different pixel weights $w_i$. Since now the weights may be negative, $\hat{y}$ cannot be updated via the regular Mean Shift iteration (5.8), as it was in the maximization of the Bhattacharyya coefficient. However, shifting $\hat{y}$ according to

\[
\hat{y} \leftarrow \hat{y}_0 + \frac{\sum_{i=1}^{n} (x_i - \hat{y}_0) w_i g (\|\hat{y}_0 - x_i\|^2)}{\sum_{i=1}^{n} w_i g (\|\hat{y}_0 - x_i\|^2)} \quad (5.16)
\]

will increase the kernel density even when there are negative weights [23].

### 5.4.2 Step 2: optimizing $\hat{\alpha}$

In this step, the distance $d_{\hat{q}(\hat{\alpha})}(y)$ in (5.12) has to be minimized with respect to the mixture proportions $\hat{\alpha}$, while keeping the estimated target location $\hat{y}$ unchanged. As $\hat{y}$ is fixed in this step, the histogram at the estimated target location $\hat{p}(\hat{y})$ will be denoted here in shorthand by $\hat{p}$.

\(^2\)Since the absolute value function is not differentiable at 0, its derivative there is replaced by the mean of its derivatives from each side, which equals 0. As may be seen in the obtained expression for the objective function, terms resulting from differentiating the absolute value function at 0 are independent of $y$, and thus do not affect its optimization.
In order to minimize $d_{q(\hat{\alpha})}(\hat{y})$ with respect to $\hat{\alpha}$, the following problem has to be solved for $\hat{\alpha}$:

$$\minimize \sum_{u=1}^{m} \hat{p}_u - \sum_{v=1}^{M} \hat{\alpha}_v \hat{q}_v$$

subject to $\hat{\alpha}_v \geq 0, \quad v = 1, \ldots, M$ \hspace{1cm} (5.17)

$$\sum_{v=1}^{M} \hat{\alpha}_v = 1.$$

The feasible set of this minimization problem is convex. Likewise, it is easy to see that each term in the first sum in the objective function is convex, and since convexity of functions is preserved under addition, the whole objective function is convex as well. Therefore, (5.17) is a convex optimization problem, which may be solved quickly, especially since only a few mixture proportions need to be optimized (2-4 in the experiments) in practice.

### 5.5 Experimental results

Results of testing the Mean Shift tracker with the convex hull-based target model are presented for several sequences. All the targets tracked in the experiments were such that their color histogram could not be reasonably modeled from a single view. In all experiments the RGB color space was used. Each color band was equally divided into 8 bins, except for Sequence IV, where each color band had to be divided into 32 bins because the target’s colors were very similar to the background’s. The target locations in the first frame and in the reference images were manually marked.

**Experiment 1.** The Mean Shift tracker with the convex hull-based target model was first tested on Sequence I. This sequence was previously used to test the regular Mean Shift tracker with one target view and with two target views (Fig. 5.2(a,b)). These two target views were used for the convex hull-based target model here. Tracking results, along with the estimated mixture proportions (rounded to the thousandth), are presented in Figure 5.4. The improvement over the fixed target model (Fig. 5.1 and Fig. 5.3) is evident.

**Experiment 2.** Tracking the Rubik Cube in a different, longer image sequence (Sequence II) was successful. The target views used to approximate the reference histograms are shown in Fig. 5.5, and the results are shown in Fig. 5.6.

**Experiment 3.** In this experiment a greeting card, differently colored on each side, was tracked. Although the card is planar, four target views were used (Fig. 5.7) to account for the extreme lighting changes. The results are shown in Fig. 5.8.

**Experiment 4.** This experiment consisted of tracking a woman exiting an apartment (Sequence IV). The woman, wearing light-colored clothes, puts on a black coat before she exits the apartment. Thus, her appearance changes drastically. The target views used to approximate the reference histograms are shown in Fig. 5.9. As may be seen in Fig. 5.10, the tracking was successful.
\[ \hat{\alpha}_1 = 1 \quad \hat{\alpha}_2 = 0 \]
\[ \hat{\alpha}_1 = 0.722 \quad \hat{\alpha}_2 = 0.278 \]
\[ \hat{\alpha}_1 = 0.101 \quad \hat{\alpha}_2 = 0.899 \]
Frame 1  
Frame 100  
Frame 200

\[ \hat{\alpha}_1 = 0.134 \quad \hat{\alpha}_2 = 0.866 \]
\[ \hat{\alpha}_1 = 0 \quad \hat{\alpha}_2 = 1 \]
\[ \hat{\alpha}_1 = 0 \quad \hat{\alpha}_2 = 1 \]
Frame 405  
Frame 475  
Frame 500

**Figure 5.4.** Tracking results for Sequence I using the convex hull-based target model. The reference color histograms were set using the two target views shown in Fig. 5.2.

### 5.6 Conclusion

While the Mean Shift tracker [27] proved to be robust in many tracking scenarios, there are cases where no single view suffices to produce a reference color histogram appropriate for tracking the target. Examples of such targets include (unscrambled) Rubik Cubes, flat objects colored differently on each side, and people changing their clothes during the sequence.

This chapter presented a method for immunizing the Mean Shift tracker against the above problem by using multiple reference color histograms. These histograms are obtained from different target views or for different target states. A simple method for combining these histograms into a single histogram that is more appropriate for tracking the target was suggested. In order to enhance the tracking further, an extension to the Mean Shift tracker, where the convex hull of these histograms is used as the target model, was proposed. The extended Mean Shift tracker was experimentally verified in scenarios where the visible target colors changed drastically and rapidly during the sequence.

Finally, although both methods – that of using several target views to obtain a single reference histogram and that of using the convex-hull – were proposed in the context of the Mean Shift tracker, these may also be accommodated in other tracking frameworks such as CONDENSATION [53].
Figure 5.5. The two target views used for approximating the two reference histograms (a) \( \hat{q}_1 \) and (b) \( \hat{q}_2 \) for the convex hull-based target model in Sequence II.

\[
\hat{\alpha}_1 = 1 \quad \hat{\alpha}_2 = 0 \\
\hat{\alpha}_1 = 0.034 \quad \hat{\alpha}_2 = 0.966 \\
\hat{\alpha}_1 = 0.847 \quad \hat{\alpha}_2 = 0.153 \\
\hat{\alpha}_1 = 0.001 \quad \hat{\alpha}_2 = 0.999 \\
\hat{\alpha}_1 = 0.914 \quad \hat{\alpha}_2 = 0.086 \\
\hat{\alpha}_1 = 0 \quad \hat{\alpha}_2 = 1
\]

Figure 5.6. Tracking results for Sequence II using the convex hull-based target model. The reference color histograms were set using the two target views shown in Fig. 5.5.
**Figure 5.7.** The four target views used for approximating the four reference histograms (a) $\hat{q}_1$, (b) $\hat{q}_2$, (c) $\hat{q}_3$ and (d) $\hat{q}_4$ for the convex hull-based target model in Sequence III.

\[
\begin{align*}
\hat{\alpha}_1 &= 0.886 & \hat{\alpha}_2 &= 0.029 \\
\hat{\alpha}_3 &= 0 & \hat{\alpha}_4 &= 0.085 \\
\hat{\alpha}_1 &= 0.129 & \hat{\alpha}_2 &= 0.856 \\
\hat{\alpha}_3 &= 0 & \hat{\alpha}_4 &= 0.015
\end{align*}
\]

**Figure 5.8.** Tracking results for Sequence III using the convex hull-based target model. The reference color histograms were set using the four target views shown in Fig. 5.7.
Figure 5.9. The two target views used for approximating the two reference histograms (a) \( \hat{q}_1 \) and (b) \( \hat{q}_2 \) for the convex hull-based target model in Sequence IV.

\[
\hat{\alpha}_1 = 0.987 \quad \hat{\alpha}_2 = 0.013 \\
\hat{\alpha}_1 = 0.988 \quad \hat{\alpha}_2 = 0.012
\]

Frame 1 Frame 70 Frame 180

\[
\hat{\alpha}_1 = 0.608 \quad \hat{\alpha}_2 = 0.392 \\
\hat{\alpha}_1 = 0.015 \quad \hat{\alpha}_2 = 0.985 \\
\hat{\alpha}_1 = 0.253 \quad \hat{\alpha}_2 = 0.747
\]

Frame 210 Frame 290 Frame 420

Figure 5.10. Tracking results for Sequence IV using the convex hull-based target model. The reference color histograms were set using the two target views shown in Fig. 5.9.
Chapter 6

Conclusions

This thesis addressed the problem of visual tracking in a general context. The proposed methods may roughly be categorized into two approaches, one based on tracker combination and the other based on the employment of low-level visual characteristics.

In Chapter 2 we investigated the combination of synchronous visual trackers that use different features while treating the trackers as “black boxes”. That is, instead of fusing the usage of the different types of data as has been performed in previous work, the combination here is allowed to use only the trackers’ output estimates, which may be modified before their propagation to the next time step. We proposed a probabilistic framework for combining multiple synchronous trackers, where each separate tracker outputs a probability density function of the tracked state, sequentially for each image. The trackers may output either an explicit probability density function, or a sample-set of it via CONDENSATION. Unlike previous tracker combinations, the proposed framework is fairly general and allows the combination of any set of trackers of this kind, even in different state spaces of different dimensionality, under a few reasonable assumptions. The combination may consist of different trackers that track a common object, as well as trackers that track separate, albeit related objects, thus improving the tracking performance of each object. The benefits of merely using the final estimates of the separate trackers in the combination are twofold. Firstly, the framework for the combination is fairly general and may be easily used from the software aspects. Secondly, the combination may be performed in a distributed setting, where each separate tracker runs on a different site and uses different data, while avoiding the need to share the data. The suggested framework was successfully tested using various state-spaces and datasets, demonstrating that fusing the trackers’ final distribution estimates may indeed be applicable.

In Chapter 3 was developed a tracker for tracking object bitmaps under very general conditions: a possibly non-rigid target whose appearance may drastically change over time; general camera motion; a 3D scene; and no a priori information except initialization. The proposed tracker works by approximating, in each frame, a PDF of the target’s bitmap and then estimating the maximum a posteriori bitmap. The PDF is marginalized over all possible motions per pixel, thus avoiding the stage in which optical flow is determined. This is an advantage over other general-context trackers that do
not use the motion cue at all or rely on the error-prone calculation of optical flow. Using a Gibbs distribution with respect to the first-order neighborhood system yields a bitmap PDF whose maximization may be transformed into that of a quadratic pseudo-Boolean function, the maximum of which is approximated via a reduction to a maximum-flow problem. Many experiments were conducted to demonstrate that the tracker is able to track under the aforementioned general context.

Chapter 4 introduced a kernel-based visual tracker that exploits the constancy of color and the presence of color edges along the target boundary. The tracker estimates the best affinity of a spatially aligned pair of kernels, one of which is color-related and the other of which is object boundary-related. In a sense, this work extended previous kernel-based trackers by incorporating the object boundary cue into the tracking process and by allowing the kernels to be affinely transformed instead of only translated and isotropically scaled. These two extensions make for more precise target localization. Moreover, a more accurately localized target facilitates safer updating of its reference color model, further enhancing the tracker’s robustness. The improved tracking was demonstrated for several challenging image sequences.

Chapter 5 proposed an enhancement of the Mean Shift tracker to using multiple color histograms obtained from different target views. It first suggested a simple method to use these histograms for producing a single histogram that is more appropriate for tracking the target. Then, to enhance the tracking further, it proposed an extension to the Mean Shift tracker where the convex hull of these histograms is used as the target model. Experimental results demonstrated the higher robustness achieved for tracking targets whose visible colors change drastically and rapidly during the sequence.
Appendix A

Combining Gaussian PDFs

In the following we show that the combination of a Gaussian prior PDF with Gaussian posterior PDFs, estimated by conditionally independent trackers, remains Gaussian. We will prove it for two posterior PDFs and derive (2.5). The generalization to an arbitrary number of posterior PDFs is straightforward.

Let \( G_i(x) = \frac{1}{\sqrt{(2\pi)^n|C_i|}} e^{-\frac{1}{2}(x-\mu_i)^T C_i^{-1}(x-\mu_i)} \), \( i = 1, 2, 3 \), be Gaussian PDFs with means \( \mu_i \) and covariance matrices \( C_i \), where \( n \) is the dimensionality of the random variable. (We assume the Gaussian random variables are non-degenerate, i.e., their covariance matrices are positive definite. The proof can also be extended for degenerate distributions.) Let \( G_1 \) and \( G_2 \) be the posterior PDFs, and \( G_3 \) the prior PDF. From (2.3) the combined PDF is

\[
p(x) = k \cdot \frac{G_1(x) \cdot G_2(x)}{G_3(x)} \propto e^{-\frac{1}{2}\{ (x-\mu_1)^T C_1^{-1}(x-\mu_1) + (x-\mu_2)^T C_2^{-1}(x-\mu_2) - (x-\mu_3)^T C_3^{-1}(x-\mu_3) \}}.
\]

Rearranging terms in the exponent yields

\[
p(x) \propto e^{-\frac{1}{2}\{ x^T (C_1^{-1} + C_2^{-1} - C_3^{-1}) x - x^T (C_1^{-1} \mu_1 + C_2^{-1} \mu_2 - C_3^{-1} \mu_3) - (\mu_1^T C_1^{-1} + \mu_2^T C_2^{-1} - \mu_3^T C_3^{-1}) \}}.
\]

Let us define

\[
C^{-1} \triangleq C_1^{-1} + C_2^{-1} - C_3^{-1} \quad \text{and} \quad \mu \triangleq C(C_1^{-1} \mu_1 + C_2^{-1} \mu_2 - C_3^{-1} \mu_3). \tag{A.1}
\]

Note that \( C \) exists since, as will be later explained, \( C_1^{-1} + C_2^{-1} - C_3^{-1} \) is positive definite, making it nonsingular. Also, since the \( C_i \)'s are symmetric positive definite, so are the \( C_i^{-1} \)'s, making \( C \) and \( C^{-1} \) symmetric too. Now we may rewrite

\[
p(x) \propto e^{-\frac{1}{2}\{ x^T C^{-1} x - x^T C^{-1} \mu - \mu^T C^{-1} \}}.
\]

Adding and subtracting the constant \( \mu^T C^{-1} \mu \) in the exponent leads to

\[
p(x) \propto e^{-\frac{1}{2}(x-\mu)^T C^{-1}(x-\mu)}, \tag{A.2}
\]

106
showing that $p(x)$ is indeed a Gaussian distribution $N(\mu, C)$, with the mean and covariance matrix as defined in (A.1).

Finally, we make the observation that $C^{-1}$ (and thus also $C$) is positive definite, since otherwise the integral of $p(x)$ would be infinite, contradicting $p(x)$ being a PDF. In more detail, $C^{-1} > 0$, since otherwise there would be some $x' \neq 0_{N \times 1}$ for which the quadratic term in the exponent of $p(x)$ will not be negative. This will cause $p(x)$ to tend to a positive, possibly infinite, limit as $x \to \infty$ in the direction of $x'$, prohibiting $p(x)$ from being a PDF.
Appendix B

Derivation of $f_1(x_t^p)$

Continuing from Equation (3.6) by using the chain rule yields

$$f_1(x_t^p) = \sum_{p' \in \mathcal{N}_{t-1} \cup \{ \text{none} \}} \frac{Pr(p \rightarrow p'|I_{t-1}, C_t, \mathcal{X}_{t-1})}{f_1(p'; p, t)} \cdot \frac{Pr(x_t^p | p \rightarrow p', I_{t-1}, C_t, \mathcal{X}_{t-1})}{f_2(x_t^p; p')}.$$ (B.1)

The first multiplicand inside the sum of (B.1) is the probability that $I_t$’s pixel $p$ corresponds to pixel $p'$ in $I_{t-1}$ (or the probability that it corresponds to none) when considering only the pixel colors in $I_t$ and disregarding their exact placement in the frame. Using Bayes’ rule, we have

$$f_1^1(p'; p, t) \propto Pr(p \rightarrow p'|I_{t-1}, \mathcal{X}_{t-1}) \cdot Pr(x_t^p | p \rightarrow p', I_{t-1}, C_t, \mathcal{X}_{t-1}).$$ (B.2)

Since $L_t$ is not given, the prior on the potentially corresponding pixels $p' \in \mathcal{N}_{t-1}$ is uniform, and we set the prior probability of having no corresponding pixel in the previous frame to $P_{\text{none}}$. Subject to this and under the constancy of color assumption, for $p' \in \mathcal{N}_{t-1}$ we approximate the first multiplicand inside the sum of (B.1) as

$$f_1^1(p'; p, t) \propto \frac{1 - P_{\text{none}}}{|I_{t-1}|} \cdot N_{\mu, C}(c_t^p), \quad p' \in \mathcal{N}_{t-1},$$ (B.3)

where $N_{\mu, C}$ is the normal PDF of mean $\mu$ and covariance matrix $C$ that is set as a constant as well. ($C$ is set to an identity matrix scaled by a variance reflecting the degree of color similarity assumed in the constancy of color characteristic) For $p' = \text{none}$ we approximate

$$f_1^1(\text{none}; p, t) \propto P_{\text{none}} \cdot U(c_t^p),$$ (B.4)

where $U$ is the uniform PDF on the color-space (RGB in our implementation). Note that modeling $f_1^1(p'; p, t)$ in (B.3) as a multidimensional Gaussian causes this function (in $p'$) to be very highly peaked. This makes the tracker’s sensitivity to $P_{\text{none}}$ minor. In our implementation $P_{\text{none}}$ was set to 0.1.
In practice, in order to estimate the pixel correspondences more exactly, we compare small image patches (5 pixels in diameter) centered around the candidate pixels instead of merely comparing the candidate pixels themselves. This change is made by modifying the normal and uniform PDFs in Equations (B.3) and (B.4), respectively, to products of the color PDFs of the pixels in the patches. In addition, since pixels that are projections of different objects are likely to have different optical flows despite their adjacency in the image plane, we avoid comparing an image patch in $I_t$ to an image patch in $I_{t-1}$ that contains a mix of pixels $\hat{x}_{t-1}$ assigned $x_{t-1} = 1$ and $x_{t-1} = 0$. In such cases, we compare only the pixels in the patch that are assigned the same bitmap value as is assigned to the center pixel, which is the one the correspondence is sought for. We also restrict the maximal size of optical flow to $M$ pixels (in our implementation $M = 6$), and compare only image patches distanced at most by $M$, which reduces the number of computations. Thus, the sum in (B.1) is computed over a subset of feasible pixels in $I_{t-1}$ (137 pixels for $M = 6$) and none, which reduces computation time. We conclude for the first multiplicand inside the sum of (B.1):

$$f_1^1(p'; p, t) \propto \begin{cases} 
1 - P_{\text{none}} & p' \in D_{t-1}(p), \\
P_{\text{none}} \cdot U(c_t^p) & p' = \text{none}, 
\end{cases}$$

(B.5)

where $D_{t-1}(p) \triangleq \{ p' : \|p_{t-1}' - p_t\|_2 \leq M \}$ is the index-set of pixels in $I_{t-1}$ within a radius of $M$ pixels from pixel $p$, and $c_t^p$ is the vector of colors of every pixel composing the image patch for pixel $p$ in $I_t$, say in raster order. Since all the feasible cases for $p'$ are covered by $D_{t-1}(p) \cup \{\text{none}\}$, normalizing to a unit sum over $p' \in D_{t-1}(p) \cup \{\text{none}\}$ produces the estimated probabilities (although normalizing here is not necessary, as it will only scale $P(X_t)$, which does not change its maximizing bitmap).

The second multiplicand inside the sum of (B.1) is the PDF of the bitmap’s value at pixel $p$ in $I_t$, conditional on this pixel’s correspondence to pixel $p'$ in $I_{t-1}$, whose bitmap value is given. Since the MAP bitmap estimated for the previous frame may contain errors, we set this PDF to

$$f_1^2(x_t^p; p') = \begin{cases} 
P_{\text{correct}} & x_t^p = x_{t-1}', \\
1 - P_{\text{correct}} & x_t^p \neq x_{t-1}', 
\end{cases}$$

(B.6)

where $P_{\text{correct}}$ is a preset constant approximating the prior probability of the estimated bitmap being correct for a pixel. ($P_{\text{correct}}$ is typically set to 0.9.) For $p' = \text{none}$ we set this PDF to

$$f_1^2(x_t^p; \text{none}) = \begin{cases} 
P_{\text{target}} & x_t^p = 1, \\
1 - P_{\text{target}} & x_t^p = 0, 
\end{cases}$$

(B.7)

where $P_{\text{target}}$ is also a preset constant approximating the prior probability of a pixel, with no corresponding pixel in the previous frame, to belong to the target. ($P_{\text{target}}$ is typically set to 0.4.)

To conclude the steps for computing $f_1(x_t^p)$ for pixel $p$ in $I_t$, we first use Equation (B.5) to compute the probabilities $f_1^1(p'; p, t)$ for $p' \in D_{t-1}(p) \cup \{\text{none}\}$, that is,
the probabilities for pixel $p$’s different correspondences to pixels in $I_{t-1}$ (feasible sub-
ject to the maximal optical flow assumed), including the probability of having no cor-
responding pixel. Then, by substituting Equations (B.6) and (B.7) into Equation (B.1),
we derive

$$f_1 (x^p_t = 1) = P_{correct} \cdot \sum_{p' \in D_{t-1}(p) \cap \{q; x^q_{t-1} = 1\}} f_1 (p'; p, t)$$

$$+ (1 - P_{correct}) \cdot \sum_{p' \in D_{t-1}(p) \cap \{q; x^q_{t-1} = 0\}} f_1 (p'; p, t) + P_{target} \cdot f_1 (none; p, t), \quad (B.8)$$

and by complementing,

$$f_1 (x^p_t = 0) = 1 - f_1 (x^p_t = 1). \quad (B.9)$$
Appendix C

Derivation of \( f_{\Delta} (x_t^{p_1}, x_t^{p_2}) \)

In the following we shall derive and show how we compute \( f_{\Delta} (x_t^{p_1}, x_t^{p_2}) \), which equals \( \Pr \left( \Delta_t(p_1, p_2) \big| \text{adj}(p_1, p_2), x_t^{p_1}, x_t^{p_2}, \mathcal{I}_{t-1}, C_t, \mathcal{X}_{t-1} \right) \), where \( \Delta_t(p_1, p_2) \triangleq t_t^{p_1} - t_t^{p_2} \) and \( \text{adj}(p_1, p_2) \) is the event of pixels \( p_1 \) and \( p_2 \) being adjacent.

Marginalizing \( f_{\Delta} (x_t^{p_1}, x_t^{p_2}) \) over all the potential correspondences of pixels \( p_1 \) and \( p_2 \) to pixels in \( \mathcal{I}_{t-1} \), including the event of corresponding to none, and then applying the chain rule, yields

\[
f_{\Delta} (x_t^{p_1}, x_t^{p_2}) = \sum_{p_1', p_2' \in \mathcal{N}_{t-1} \cup \{\text{none}\}} \Pr \left( p_1 \rightarrow p_1', p_2 \rightarrow p_2' \big| \text{adj}(p_1, p_2), x_t^{p_1}, x_t^{p_2}, \mathcal{I}_{t-1}, C_t, \mathcal{X}_{t-1} \right) \frac{f_{\Delta} (p_1', p_2', x_t^{p_1}, x_t^{p_2}; p_1, p_2)}{f_{\Delta} (x_t^{p_1}, x_t^{p_2}; \Delta_t(p_1, p_2), p_1', p_2')} .
\]

(C.1)

The second multiplicand inside the sum of (C.1) is the likelihood of the relative position between adjacent pixels \( p_1 \) and \( p_2 \) in \( \mathcal{I}_t \), where the coordinates of their corresponding pixels in \( \mathcal{I}_{t-1} \), if any, are known (because the likelihood is conditional on the pixel correspondences and on \( \mathcal{I}_{t-1} \), which consists of \( \mathcal{L}_{t-1} \)). In accordance with the spatial motion continuity characteristic, we approximate this likelihood as is summarized in Table C.1. When \( x_t^{p_1} = x_t^{p_2} \) and both pixels have corresponding pixels in \( \mathcal{I}_{t-1} \), it is very likely that \( \Delta_t(p_1, p_2) = \Delta_{t-1}(p_1', p_2') \). The probability of this event is assigned \( P_{\text{flow}_1} \), which is a preset constant of typical value 0.99. The complementary event of \( \Delta_t(p_1, p_2) \neq \Delta_{t-1}(p_1', p_2') \) is thus assigned \( 1 - P_{\text{flow}_1} \). Equivalently, when \( x_t^{p_1} \neq x_t^{p_2} \) and both pixels have corresponding pixels in \( \mathcal{I}_{t-1} \), it is very likely that \( \Delta_t(p_1, p_2) \neq \Delta_{t-1}(p_1', p_2') \). The probability of this event is assigned \( P_{\text{flow}_2} \), which is a preset constant as well, with a typical value of 0.99. Complementing again yields that the event of \( \Delta_t(p_1, p_2) = \Delta_{t-1}(p_1', p_2') \) is \( 1 - P_{\text{flow}_2} \). When one or both of the pixels have no corresponding pixel in \( \mathcal{I}_{t-1} \), the spatial motion continuity characteristic is irrelevant and the four different values for \( \Delta_t(p_1, p_2) \) are assigned the same probability of 0.25.

111
Following the partitioning of the possibilities for $p'_1$ and $p'_2$ summarized in Table C.1, the sum in (C.1) may be split into three cases:

\[
\begin{align*}
\{ (p'_1, p'_2) &\in \mathcal{N}_{t-1}^2 \mid \Delta_{t-1}(p'_1, p'_2) = \Delta_t(p_1, p_2) \}, \\
\{ (p'_1, p'_2) &\in \mathcal{N}_{t-1}^2 \mid \Delta_{t-1}(p'_1, p'_2) \neq \Delta_t(p_1, p_2) \} \quad \text{and} \\
\{ (p'_1, p'_2) &\in (\mathcal{N}_{t-1} \cup \{\text{none}\})^2 \mid p'_1 = \text{none or } p'_2 = \text{none} \}. 
\end{align*}
\]

For $x^{p_1}_t = x^{p_2}_t$ Equation (C.1) becomes

\[
f_{\Delta}(x^{p_1}_t, x^{p_2}_t) = P_{\text{flow}_1} \cdot \sum_{p'_1, p'_2 \in \mathcal{N}_{t-1}} f^1_{\Delta}(p'_1, p'_2, x^{p_1}_t, x^{p_2}_t; p_1, p_2) \frac{S_1(x^{p_1}_t, x^{p_2}_t; p_1, p_2)}{\Delta_{t-1}(p'_1, p'_2) = \Delta_t(p_1, p_2)} \\
+ (1 - P_{\text{flow}_1}) \cdot \sum_{p'_1, p'_2 \in \mathcal{N}_{t-1}} f^1_{\Delta}(p'_1, p'_2, x^{p_1}_t, x^{p_2}_t; p_1, p_2) \frac{S_2(x^{p_1}_t, x^{p_2}_t; p_1, p_2)}{\Delta_{t-1}(p'_1, p'_2) \neq \Delta_t(p_1, p_2)} \\
+ 0.25 \cdot \sum_{p'_1, p'_2 \in \mathcal{N}_{t-1} \cup \{\text{none}\}} f^1_{\Delta}(p'_1, p'_2, x^{p_1}_t, x^{p_2}_t; p_1, p_2) \frac{S_3(x^{p_1}_t, x^{p_2}_t; p_1, p_2)}{p'_1 = \text{none or } p'_2 = \text{none or } p'_1 \neq x^{p_2}_t}
\]

and for $x^{p_1}_t \neq x^{p_2}_t$

\[
f_{\Delta}(x^{p_1}_t, x^{p_2}_t) = (1 - P_{\text{flow}_2}) \cdot S_1(x^{p_1}_t, x^{p_2}_t; p_1, p_2) + P_{\text{flow}_2} \cdot S_2(x^{p_1}_t, x^{p_2}_t; p_1, p_2) \\
+ 0.25 \cdot S_3(x^{p_1}_t, x^{p_2}_t; p_1, p_2), \quad x^{p_1}_t \neq x^{p_2}_t. \tag{C.3}
\]

<table>
<thead>
<tr>
<th>$p'_1$ none or $p'_2$ none</th>
<th>$p'_1, p'<em>2 \in \mathcal{N}</em>{t-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^{p'_1}_t = x^{p'_2}_t$</td>
<td>$\Delta_{t-1}(p'_1, p'_2) = \Delta_t(p_1, p_2)$</td>
</tr>
<tr>
<td>0.25</td>
<td>$P_{\text{flow}_1}$</td>
</tr>
<tr>
<td>$x^{p'_1}_t \neq x^{p'_2}_t$</td>
<td>0.25</td>
</tr>
</tbody>
</table>

The term inside the summations (which is the first multiplicand inside the sum of (C.1)) is the joint probability that $I_t$’s pixels $p_1$ and $p_2$ correspond to pixels $p'_1$ and $p'_2$ in $I_{t-1}$, respectively (or correspond to none). This term is similar to $f^1_1(p'; p, t)$, which is the correspondence distribution for a single pixel, although now the conditioning is also on the Boolean variables of $I_t$’s pixels. Calculating the sums in (C.2) and (C.3) for a single pair of pixels under one $(x^{p_1}_t, x^{p_2}_t)$-hypothesis (out of four different hypotheses) would require estimating this term for a number of cases that is quadratic in the size of the image region for searching corresponding pixels, which we find to be too computationally demanding. (For $M = 6$ pixels, the number of such
cases is \((137 + 1)^2 = 19,044\). To reduce the computational cost, we replace the exact calculation of the three sums by an estimate based on the marginal PDFs

\[
f_{\Delta \text{marginal}} (p', x'_t; p) \overset{\Delta}{=} \Pr \left( p \to p' \mid x_t, t \in \mathcal{I}_t-1, c_t, x_{t-1} \right), \quad p \in \mathcal{N}_t, \; p' \in \mathcal{N}_{t-1} \cup \{ \text{none} \},
\]

and obtain estimates for the four likelihoods \(f_\Delta (x_t^{p_1} = b_1, x_t^{p_2} = b_2) \) \((b_1, b_2 \in \{0, 1\})\).

In what follows, we show how \(f_{\Delta \text{marginal}} (p', x'_t; p)\) is calculated and used to obtain an estimate for \(f_\Delta (x_t^{p_1}, x_t^{p_2})\).

**Calculating** \(f_{\Delta \text{marginal}} (p', x'_t; p)\)

For \(p' \in \mathcal{N}_{t-1}\), marginalizing \(f_{\Delta \text{marginal}} (p', x'_t; p)\) over the correctness of \(x'_t\), followed by application of the chain rule yields

\[
f_{\Delta \text{marginal}} (p', x'_t; p) = \Pr \left( x'_t \text{ is correct} \right) \cdot \Pr \left( p \to p' \mid x'_t \text{ is correct}, x_t, t \in \mathcal{I}_t-1, c_t, x_{t-1} \right) + \Pr \left( x'_t \text{ is incorrect} \right) \cdot \Pr \left( p \to p' \mid x'_t \text{ is incorrect}, x_t, t \in \mathcal{I}_t-1, c_t, x_{t-1} \right),
\]

and using the predefined constant \(P_{\text{correct}}\) leads to

\[
f_{\Delta \text{marginal}} (p', x'_t; p) = P_{\text{correct}} \cdot \Pr \left( p \to p' \mid x'_t \text{ is correct}, x_t, t \in \mathcal{I}_t-1, c_t, x_{t-1} \right) + (1 - P_{\text{correct}}) \cdot \Pr \left( p \to p' \mid x'_t \text{ is incorrect}, x_t, t \in \mathcal{I}_t-1, c_t, x_{t-1} \right).
\]

This results in

\[
f_{\Delta \text{marginal}} (p', x'_t; p) = \begin{cases} 
P_{\text{correct}} \cdot \Pr \left( p \to p' \mid x'_t \text{ is correct}, x_t, t \in \mathcal{I}_t-1, c_t, x_{t-1} \right) & x_t = x'_t, \; p' \in \mathcal{N}_{t-1}, \\
(1 - P_{\text{correct}}) \cdot \Pr \left( p \to p' \mid x'_t \text{ is incorrect}, x_t, t \in \mathcal{I}_t-1, c_t, x_{t-1} \right) & x_t \neq x'_t,
\end{cases}
\]

By Bayes’ rule

\[
\Pr \left( p \to p' \mid x'_t \text{ is (in)correct}, x_t, t \in \mathcal{I}_t-1, c_t, x_{t-1} \right) \propto \Pr \left( p \to p' \mid x'_t \text{ is (in)correct}, x_t, t \in \mathcal{I}_t-1, x_{t-1} \right) \cdot \Pr \left( c_t \mid p \to p', x'_t \text{ is (in)correct}, x_t, t \in \mathcal{I}_t-1, x_{t-1} \right), \quad p' \in \mathcal{N}_{t-1}.
\]

113
Using the same approximations as in (B.3), we obtain

\[
\Pr \left( p \rightarrow p' \mid x_{t-1}' \text{ is correct, } X_t, T_{t-1}, C_t, X_{t-1} \right) \\
\propto \frac{1 - P_{\text{none}}}{|N_{t-1}| P_{\text{correct}} + |N_{t-1} \cap \{ q : x_{t-1}' = x_q^p \}| (1 - 2P_{\text{correct}})} \\
\times N_{c_{t-1}, C}(\theta^p_t), \quad x_t^p = x_{t-1}', \quad (C.7)
\]

and

\[
\Pr \left( p \rightarrow p' \mid x_{t-1}' \text{ is incorrect, } X_t, T_{t-1}, C_t, X_{t-1} \right) \\
\propto \frac{1 - P_{\text{none}}}{(|N_{t-1}| + 1) P_{\text{correct}} + |N_{t-1} \cap \{ q : x_{t-1}' = x_q^p \}| (1 - 2P_{\text{correct}})} \\
\times N_{c_{t-1}, C}(\theta^p_t), \quad x_t^p \neq x_{t-1}'. \quad (C.8)
\]

Substituting these into (C.5) gives

\[
f_{\Delta \text{marginal}}^1 (p', x_t^p ; p) \propto \begin{cases} 
P_{\text{correct}} \cdot A_e \cdot N_{c_{t-1}', C}(\theta^p_t) & x_t^p = x_{t-1}', \\
(1 - P_{\text{correct}}) \cdot A_\# \cdot N_{c_{t-1}', C}(\theta^p_t) & x_t^p \neq x_{t-1}', \quad p' \in N_{t-1}.
\end{cases}
\]

For \( p' = \text{none} \) the conditioning on \( X_t \) has no influence on \( f_{\Delta \text{marginal}}^1 (p', x_t^p ; p) \) and thus, as in (B.4),

\[
f_{\Delta \text{marginal}}^1 (\text{none}, x_t^p ; p) \propto P_{\text{none}} \cdot U(\theta^p_t). \quad (C.10)
\]

Due to the same considerations that lead to (B.5), we conclude for the marginal PDFs of the pixel correspondences that

\[
f_{\Delta \text{marginal}}^1 (p', x_t^p ; p) \\
\propto \begin{cases} 
P_{\text{correct}} \cdot A_e \cdot N_{c_{t-1}', C}(\theta^p_t) & p' \in D_{t-1}(p) \text{ and } x_t^p = x_{t-1}', \\
(1 - P_{\text{correct}}) \cdot A_\# \cdot N_{c_{t-1}', C}(\theta^p_t) & p' \in D_{t-1}(p) \text{ and } x_t^p \neq x_{t-1}', \quad (C.11)
P_{\text{none}} \cdot U(\theta^p_t) & p' = \text{none}.
\end{cases}
\]

Normalizing to a unit sum over the “generalized neighborhood” of \( p, D_{t-1}(p) \cup \{\text{none}\} \), produces the estimated probabilities.
Estimating $f_\Delta (x_{t}^{p_1}, x_{t}^{p_2})$

The third sum in (C.2) and (C.3), which is the probability that at least one of the two pixels has no corresponding pixel in the previous frame, is

$$S_3 (x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2) = \sum_{p_2' \in \mathcal{N}_{t-1} \cup \{\text{none}\}} f_{\Delta}^1 (\text{none}, p_2', x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2)$$

$$+ \sum_{p_1' \in \mathcal{N}_{t-1} \cup \{\text{none}\}} f_{\Delta}^1 (p_1', \text{none}, x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2) - f_{\Delta}^1 (\text{none}, \text{none}, x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2)$$

$$= f_{\Delta}^1_{\text{marginal}} (\text{none}, x_{t}^{p_1}; p_1) + f_{\Delta}^1_{\text{marginal}} (\text{none}, x_{t}^{p_2}; p_2)$$

$$- f_{\Delta}^1 (\text{none}, \text{none}, x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2), \quad (C.12)$$

and modeling the events $p_1' = \text{none}$ and $p_2' = \text{none}$ as independent, we obtain

$$S_3 (x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2) = f_{\Delta}^1_{\text{marginal}} (\text{none}, x_{t}^{p_1}; p_1) + f_{\Delta}^1_{\text{marginal}} (\text{none}, x_{t}^{p_2}; p_2)$$

$$- f_{\Delta}^1 (\text{none}, \text{none}, x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2). \quad (C.13)$$

Turning to $S_1 (x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2)$, which is the probability that the two pixels have identical discrete optical flows from the previous frame, and denoting

$$k (x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2) \triangleq 1 - f_{\Delta}^1_{\text{marginal}} (\text{none}, x_{t}^{p_1}; p_1) \cdot f_{\Delta}^1_{\text{marginal}} (\text{none}, x_{t}^{p_2}; p_2), \quad (C.14)$$

it is easy to verify the $S_1 (x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2)$ bounds

$$S_1 (x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2) \geq \sum_{p_1', p_2' \in \mathcal{N}_{t-1} \text{ such that} \Delta_{t-1} (p_1', p_2') = \Delta_t (p_1, p_2)} \max \left\{ 0, f_{\Delta}^1_{\text{marginal}} (p_1', x_{t}^{p_1}; p_1) \right\}$$

$$+ f_{\Delta}^1_{\text{marginal}} (p_2', x_{t}^{p_2}; p_2) - k (x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2) \} \right\}$$

$$S_1 (x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2) \leq \sum_{p_1', p_2' \in \mathcal{N}_{t-1} \text{ such that} \Delta_{t-1} (p_1', p_2') = \Delta_t (p_1, p_2)} \min \left\{ f_{\Delta}^1_{\text{marginal}} (p_1', x_{t}^{p_1}; p_1), \right\}$$

$$f_{\Delta}^1_{\text{marginal}} (p_2', x_{t}^{p_2}; p_2) \} \right\} \quad (C.15)$$

By complementing, the second sum in (C.2) and (C.3), which is the probability of having different discrete optical flows, is

$$S_2 (x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2) = 1 - S_3 (x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2) - S_1 (x_{t}^{p_1}, x_{t}^{p_2}; p_1, p_2) \quad (C.16)$$

Equations (C.13)-(C.16) induce immediate bounds on $f_\Delta (x_{t}^{p_1}, x_{t}^{p_2})$

$$\text{low} (x_{t}^{p_1}, x_{t}^{p_2}) \leq f_\Delta (x_{t}^{p_1}, x_{t}^{p_2}) \leq \text{up} (x_{t}^{p_1}, x_{t}^{p_2}) \quad (C.17)$$

Thus, for each unordered pair of adjacent pixels $p_1$ and $p_2$ in $\mathcal{I}_t$, there are the four intervals

$$\text{low} (x_{t}^{p_1} = b_1, x_{t}^{p_2} = b_2) \leq f_\Delta (x_{t}^{p_1} = b_1, x_{t}^{p_2} = b_2) \leq \text{up} (x_{t}^{p_1} = b_1, x_{t}^{p_2} = b_2), \quad b_i \in \{0, 1\}. \quad (C.18)$$
Avoiding additional computations, we take only these interval restrictions into account and set the four likelihoods \( f_\Delta(x_1^{p_1} = b_1, x_1^{p_2} = b_2) \) (\( b_1, b_2 \in \{0, 1\} \)), under these interval restrictions, to be as close to each other as possible so that the sum of their differences

\[
\frac{1}{2} \sum_{(b_1, b_2) \in \{0, 1\}^2} \sum_{(b_1', b_2') \in \{0, 1\}^2} |f_\Delta(x_1^{p_1} = b_1, x_1^{p_2} = b_2) - f_\Delta(x_1^{p_1} = b_1', x_1^{p_2} = b_2')| \tag{C.19}
\]

is minimized. (A minimization method is given next paragraph.) Note that if \( f_\Delta(x_1^{p_1}, x_1^{p_2}) \) is equal for all four possible bit-assignments, it will have no effect at all on the maximization of the bitmap PDF \((3.4)\), which is proportional to it. Qualitatively, by closely clustering the four values of \( f_\Delta(x_1^{p_1}, x_1^{p_2}) \), the effect on the bitmap’s PDF is minimized, while the interval restrictions are obeyed. Therefore, the larger the uncertainty (i.e., interval) in the values of \( f_\Delta(x_1^{p_1}, x_1^{p_2}) \), the less the effect of this component on the bitmap’s PDF. It is easily seen through \((C.15)\) that the more unequivocal the marginal optical flows \( f_{\Delta_{\text{marginal}}}(p_i', x_1^{p_2}; p_i) \), the smaller these uncertainties. A typical histogram of these interval sizes is given in Fig. C.1 (the largest interval out of the four is taken per pixel), showing that a large portion of the \( f_\Delta s \) have small intervals and thus significantly affect the bitmap’s PDF.

![Figure C.1.](image)

**Figure C.1.** (b) The histogram of the \( f_\Delta s \) interval sizes computed for the object marked in (a) (the largest interval out of the four is taken per pixel). We see that a large portion of the \( f_\Delta s \) have small intervals and thus affect the bitmap’s PDF.

The minimization of \((C.19)\) within the intervals of \((C.18)\) may be easily accomplished by algorithm MINIMIZE hereinafter. The setting provided by the algorithm is an optimal one, as proven in the following.

**Proof:** Since the sum of differences \((C.19)\) is continuous in the \( f_\Delta s \), we assume w.l.o.g. that the eight bounds are all different.

First, observe that in an optimal setting of the four values, each of them is set, within its interval, as close as possible to the median of the four. This is easy to see by supposing that one of the four values is not as close as possible to the median. Then obviously it can be brought closer and thus reduce the sum of differences \((C.19)\), which means that the setting is not optimal.
1. Sort the eight interval bounds

\[
\{\text{low} \left( x_{t_1}^{p_1} = b_1, x_{t_2}^{p_2} = b_2 \right) \}_{b_1, b_2 \in \{0,1\}} \cup \{\text{up} \left( x_{t_1}^{p_1} = b_1, x_{t_2}^{p_2} = b_2 \right) \}_{b_1, b_2 \in \{0,1\}}
\]

in ascending order.

2. Measure for each adjacent pair of bounds \textit{bound}_1 and \textit{bound}_2 (seven pairs) the sum of differences (C.19) obtained by setting each of the four \( f_\Delta (x_{t_1}^{p_1}, x_{t_2}^{p_2}) \)'s most closely to \( \text{bound}_1 + \text{bound}_2 \) within its interval.

3. Out of the seven settings, choose the setting of the \( f_\Delta (x_{t_1}^{p_1}, x_{t_2}^{p_2}) \)'s that had the smallest sum of differences.

**Algorithm Minimize**

Therefore, the search for an optimal setting may be performed, in principle, by going over all points \( v \) between the two extreme bounds, setting for each such point the four values as close as possible to it, and choosing a setting of the smallest sum of differences. We refer to such a point \( v \) as a “potential median” and to the four closest values as its corresponding values. Note that a “potential median” may not be the actual median of its four corresponding values. However, such “potential medians” may be discarded.

Now, note that a “potential median” \( v \) is the actual median of the corresponding four values only if it belongs to 4, 2, or 0 intervals. In the first and third cases, the sum of differences is equal for all “potential medians” in the range between the two bounds that are closest to \( v \). This is also true for the second case if \( v \) is the actual median of the four values. (The proof is straightforward.)

The correctness of the minimization algorithm now immediately follows. \( \square \)
Bibliography


Mean Shift algorithm is inspired by the concept of density clustering and is based on the estimation of probability density functions. Unlike other algorithms, Mean Shift does not require the specification of the number of clusters in advance. Instead, it iteratively moves each data point towards the local maximum of the data density. This makes Mean Shift particularly useful in applications where the number of clusters is unknown or varies over time.

The Mean Shift algorithm is particularly effective in applications such as image segmentation, where it can be used to identify regions of interest in an image. It is also widely used in computer vision tasks such as object tracking, where it can adapt to changes in the appearance of the target object.

In conclusion, Mean Shift is a powerful tool for density estimation and clustering. Its ability to adapt to changes in the data makes it a versatile tool for a wide range of applications. Further research is needed to explore the potential of Mean Shift in emerging areas such as big data and real-time processing.
תקציר

תחום מחקר הראייה הממוחשבת עוסק בחילוץ מידע על תכולת סצנה מתמונה אחת או יותר. המידע המחולץ עשוי להיות, למשל, המבנה התלת-ממדי של הסצנה או של גופים הכלולים בה (לדוג', סטריאו: תאור תלת-ממדי מתוך שתי תמונות או יותר), זהות הגופים (לדוג', זיהוי פרציפים), הקטגוריה锂电池 הגופים משתייכים (לדוג', זיהוי אותיות) או הקטגוריה של התמונה עצמה (למשל, החלטה האם התמונה הנה של סצנה עירונית או טבעית), חלוקה בעלת משמעות של התמונה לאזורים (סגמנטציה), ומיקומם של סוג מסוים של מטרות בתמונה (לדוג', גילוי פרציפים).

לעיתים קרובות, מקור המידע הנו סרט, דהיינו רצף תמונות. לכן, עקיבה ממוכנת אחר הגופים בסרט הנה חיונית לניתוח התמונות בו ומהווה בעיה בסיסית בראייה ממוחשבת. בנוסף, ישנם תחומים נוספים בהם פתרון בעיית העקיבה מאוד רצוי, כגון ניטור תנועה, ניטור לצרכי ביטחון וניהול ארכיונים של סרטים.

תזה זו עוסקת בבעיית העקיבה הויזואלית בסרטים ווידאו בהקשר רחב תוך שימוש בשתי גישות. הגישה הראשונה מציעה לשלב מספר עוקבים המשתמשים במאפיינים שונים ועל-כן בעל מצבי כשל שונים. מוצעת מתודולוגיה כללית לשילוב עוקבים ויזואליים מסונכרנים המוציאים (כפלט) התפלגות על מרחב המצבים. הגישה מתייחסת לעוקבים המעבירים את התפלגותן באופן מפורש, לדוגמה, על-ידי שימוש במסנן קלמן, או באמצעות מדגמים מההתפלגות, על-ידי שימוש במ瞥נים קלמון, או באמצעות המ😳线条 למטרת הפקת פרוצי שילוב של עוקבים ויזואליים(NUM) משולב לפי היכולת המיוחדת האורולוגית המ позвוצהصول ומאפשרת את הפקת פרוצי שילוב של עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקива באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיבה באמצעות עוקבים ויזואליים משולב בין פתרונות של עקיב
המחוקק עטשה ההנחיית פרופ' אהרון ריבלין ופרופ' מייכל לנדבאות בסקרולותה בצירוף הנחיות. אנ"י

מודה לכספית תמיכה בתקක"ת המחקר וההדרכה והשגת מודעות למחקר ול틋ותיו.
عقהבר ריוואליית בחקש רחבי באמצעות שליב
עקרבים וشبهים ברמות ברמות נמוכות

הはある משחר

לשמ מיילר חלקי של הדירישות לקיבלה החנור
דיקטור לפילוסופיה

עדו לייבנר

הוגש לסנט החכוני—it מכון טכנולוגי לישראל
טבת תשס"ח, ח"שת, דצמבר 2007
עקובה ויזואלית ב嘗שר רותב佈אמצעות שלוב
עקובה ורשים ברמה נמוכה נמקה
עדו לייטר
ערוד לייטר