Visual Tracking

Examensarbete utfört i
vid Tekniska högskolan vid Linköpings universitet
av

Martin Danelljan

LiTH-ISY-EX--13/4736--SE

Handledare: Fahad Khan
ISY, Linköpings universitet

Examinator: Michael Felsberg
ISY, Linköpings universitet

Linköping, 12 december 2013
Visuell följning

Visual tracking is a classical computer vision problem with many important applications in areas such as robotics, surveillance and driver assistance. The task is to follow a target in an image sequence. The target can be any object of interest, for example a human, a car or a football. Humans perform accurate visual tracking with little effort, while it remains a difficult computer vision problem. It imposes major challenges, such as appearance changes, occlusions and background clutter. Visual tracking is thus an open research topic, but significant progress has been made in the last few years.

The first part of this thesis explores generic tracking, where nothing is known about the target except for its initial location in the sequence. A specific family of generic trackers that exploit the FFT for faster tracking-by-detection is studied. Among these, the CSK tracker have recently shown obtain competitive performance at extraordinary low computational costs. Three contributions are made to this type of trackers. Firstly, a new method for learning the target appearance is proposed and shown to outperform the original method. Secondly, different color descriptors are investigated for the tracking purpose. Evaluations show that the best descriptor greatly improves the tracking performance. Thirdly, an adaptive dimensionality reduction technique is proposed, which adaptively chooses the most important feature combinations to use. This technique significantly reduces the computational cost of the tracking task. Extensive evaluations show that the proposed tracker outperform state-of-the-art methods in literature, while operating at several times higher frame rate.

In the second part of this thesis, the proposed generic tracking method is applied to human tracking in surveillance applications. A causal framework is constructed, that automatically detects and tracks humans in the scene. The system fuses information from generic tracking and state-of-the-art object detection in a Bayesian filtering framework. In addition, the system incorporates the identification and tracking of specific human parts to achieve better robustness and performance. Tracking results are demonstrated on a real-world benchmark sequence.

Nyckelord

Tracking, Computer Vision, Person Tracking, Object Detection, Deformable Parts Model, Rao-Blackwellized Particle Filter, Color Names
Abstract

Visual tracking is a classical computer vision problem with many important applications in areas such as robotics, surveillance and driver assistance. The task is to follow a target in an image sequence. The target can be any object of interest, for example a human, a car or a football. Humans perform accurate visual tracking with little effort, while it remains a difficult computer vision problem. It imposes major challenges, such as appearance changes, occlusions and background clutter. Visual tracking is thus an open research topic, but significant progress has been made in the last few years.

The first part of this thesis explores generic tracking, where nothing is known about the target except for its initial location in the sequence. A specific family of generic trackers that exploit the FFT for faster tracking-by-detection is studied. Among these, the CSK tracker have recently shown obtain competitive performance at extraordinary low computational costs. Three contributions are made to this type of trackers. Firstly, a new method for learning the target appearance is proposed and shown to outperform the original method. Secondly, different color descriptors are investigated for the tracking purpose. Evaluations show that the best descriptor greatly improves the tracking performance. Thirdly, an adaptive dimensionality reduction technique is proposed, which adaptively chooses the most important feature combinations to use. This technique significantly reduces the computational cost of the tracking task. Extensive evaluations show that the proposed tracker outperform state-of-the-art methods in literature, while operating at several times higher frame rate.

In the second part of this thesis, the proposed generic tracking method is applied to human tracking in surveillance applications. A causal framework is constructed, that automatically detects and tracks humans in the scene. The system fuses information from generic tracking and state-of-the-art object detection in a Bayesian filtering framework. In addition, the system incorporates the identification and tracking of specific human parts to achieve better robustness and performance. Tracking results are demonstrated on a real-world benchmark sequence.
Acknowledgments

I want to thank my supervisor Fahad Khan and examiner Michael Felsberg.

Further, I want to thank everyone that have contributed with constructive comments and discussions. I thank Klas Nordberg for discussing various parts of the theory with me. I thank Zoran Sjanic for the long and late discussions about Bayesian filtering methods.

Finally, I thank Giulia Meneghetti for helping me set up all the computers I needed.

Linköping, Januari 2014
Martin Danelljan
# Contents

Notation xi

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>A Brief Introduction to Visual Tracking</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1</td>
<td>Introducing Circulant Tracking by Detection with Kernels</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>Thesis Overview</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1</td>
<td>Problem Formulation</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2</td>
<td>Motivation</td>
<td>5</td>
</tr>
<tr>
<td>1.2.3</td>
<td>Approaches and Results</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td>Thesis Outline</td>
<td>6</td>
</tr>
</tbody>
</table>

## I Generic Tracking

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Circulant Tracking by Detection</td>
<td>9</td>
</tr>
<tr>
<td>2.1</td>
<td>The MOSSE Tracker</td>
<td>9</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Detection</td>
<td>9</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Training</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>The CSK Tracker</td>
<td>11</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Training with a Single Image</td>
<td>11</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Detection</td>
<td>12</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Multidimensional Feature Maps</td>
<td>12</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Kernel Functions</td>
<td>12</td>
</tr>
<tr>
<td>2.3</td>
<td>Robust Appearance Learning</td>
<td>13</td>
</tr>
<tr>
<td>2.4</td>
<td>Details</td>
<td>14</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Parameters</td>
<td>14</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Windowing</td>
<td>15</td>
</tr>
<tr>
<td>2.4.3</td>
<td>Feature Value Normalization</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Color Features for Tracking</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>Evaluated Color Features</td>
<td>17</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Color Names</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>Incorporating Color into Tracking</td>
<td>19</td>
</tr>
</tbody>
</table>
4 Adaptive Dimensionality Reduction
4.1 Principal Component Analysis
4.2 The Theory Behind the Proposed Approach
4.2.1 The Data Term
4.2.2 The Smoothness Term
4.2.3 The Total Cost Function
4.3 Details of the Proposed Approach

5 Evaluation
5.1 Evaluation Methodology
5.1.1 Evaluation Metrics
5.1.2 Dataset
5.1.3 Trackers and Parameters
5.2 Circulant Structure Trackers Evaluation
5.2.1 Grayscale Experiment
5.2.2 Grayscale and Color Names Experiment
5.2.3 Experiments with Other Color Features
5.3 Color Evaluation
5.3.1 Results
5.3.2 Discussion
5.4 Adaptive Dimensionality Reduction Evaluation
5.4.1 Number of Feature Dimensions
5.4.2 Final Performance
5.5 State-of-the-Art Evaluation
5.6 Conclusions and Future Work

II Category Object Tracking

6 Tracking Model
6.1 System Overview
6.2 Object Model
6.2.1 Object Motion Model
6.2.2 Part Deformations and Motion
6.2.3 The Complete Transition Model
6.3 The Measurement Model
6.3.1 The Image Likelihood
6.3.2 The Model Likelihood
6.4 Applying the Rao-Blackwellized Particle Filter to the Model
6.4.1 The Transition Model
6.4.2 The Measurement Update for the Non-Linear States
6.4.3 The Measurement Update for the Linear States

7 Object Detection
7.1 Object Detection with Discriminatively Trained Part Based Models
Bibliography
## Notation

### Sets

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbb{Z} )</td>
<td>The set of integers</td>
</tr>
<tr>
<td>( \mathbb{R} )</td>
<td>The set of real numbers</td>
</tr>
<tr>
<td>( \mathbb{C} )</td>
<td>The set of complex numbers</td>
</tr>
<tr>
<td>( \ell_p(M, N) )</td>
<td>The set of all functions ( f : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R} ) with period ( (M, N) )</td>
</tr>
<tr>
<td>( \ell_p^D(M, N) )</td>
<td>The set of all functions ( f : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R}^D ) with period ( (M, N) )</td>
</tr>
</tbody>
</table>

### Functions and Operators

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \langle \cdot, \cdot \rangle )</td>
<td>Inner product</td>
</tr>
<tr>
<td>( | \cdot | )</td>
<td>( L^2 )-norm</td>
</tr>
<tr>
<td>(</td>
<td>\cdot</td>
</tr>
<tr>
<td>( \ast )</td>
<td>Convolution</td>
</tr>
<tr>
<td>( \ast )</td>
<td>Correlation</td>
</tr>
<tr>
<td>( \mathcal{F} )</td>
<td>The Discrete Fourier Transform on ( \ell_p(M, N) )</td>
</tr>
<tr>
<td>( \mathcal{F}^{-1} )</td>
<td>The Inverse Discrete Fourier Transform on ( \ell_p(M, N) )</td>
</tr>
<tr>
<td>( \tau_{m,n} )</td>
<td>The shift operator ( (\tau_{m,n}f)(k,l) = f(k - m, l - n) )</td>
</tr>
<tr>
<td>( \kappa(f, g) )</td>
<td>Kernel on some space ( \mathcal{X} ), with ( f, g \in \mathcal{X} )</td>
</tr>
<tr>
<td>( \overline{x} )</td>
<td>Complex conjugate of ( x \in \mathbb{C} )</td>
</tr>
<tr>
<td>( p(x) )</td>
<td>Probability of ( x )</td>
</tr>
<tr>
<td>( \mathcal{N}(\mu, C) )</td>
<td>Gaussian distribution with mean ( \mu ) and covariance matrix ( C )</td>
</tr>
</tbody>
</table>
Visual tracking can be defined as the problem of estimating the trajectory of one or multiple objects in an image sequence. It is an important and classical computer vision problem, which has received much research attention over the last decades. Visual tracking has many important applications. It often acts as a part in higher level systems, e.g. automated surveillance, gesture recognition and robotic navigation. It is generally a challenging problem for numerous reasons, including fast motion, background clutter, occlusions and appearance changes.

This thesis explores various aspects of visual tracking. The rest of of this chapter is organized as follows. Section 1.1 gives a brief introduction to the visual tracking field. Section 1.2 contains the thesis problem formulation and motivation together with a brief overview of the contributions and results. Section 1.3 describes the outline of the rest of the thesis.

1.1 A Brief Introduction to Visual Tracking

A survey of visual tracking methods lies far outside the scope of this document. The visual tracking field is extremely diverse, and it does not exist anything close to a unified theory. This section introduces some concepts in visual tracking that are related to the work in this thesis. See [48] for a more complete (but rather outdated) survey.

Visual tracking methods largely depend on the application and the amount of available prior information. There are for example applications where the camera and background are known to be static. In such cases one can employ background subtraction techniques to detect moving targets. In many cases the problem is to track certain kinds or classes of objects, e.g. humans or cars. The general appearance of objects in the class can thus be used as prior information. This type of tracking problems are sometimes referred to
as category tracking, as the task is to track a category of objects. Many visual tracking methods deal with the case where only the initial target position in a sequence is known. This case is often called generic tracking.

Visual tracking of an object can in its simplest form be divided into two parts.

1. Modelling.
2. Tracking.

The first step constructs a model of the object. This model can include various degrees of a priori information and it can be updated in each new frame with new information about the object. The model should include some representation of the object appearance. The appearance can for example be modelled using the object shape, as templates, histograms or by parametric representations of distributions. An often essential part of the modelling is the choice of image features. Popular choices are intensity, color and edge features. Models can also include information about the expected object motion.

The second step deals with how to use the model to find the position of the object in the next frame. Methods applied in this step are highly dependent on the used model. A very simple example of a model is to use the pixel values of a rectangular image area around the object. A simple way of tracking with this model is by correlating the new frame with this image patch and finding the maximum response.

A popular trend in how to solve this modelling/tracking problem is to apply tracking by detection. This is a quite loose categorization of a kind of visual tracking algorithms.
Tracking by detection essentially means that some kind of machine learning technique is used to train a discriminative classifier of the object appearance. The tracking step is then done by classifying image regions to find the most probable location of the object.

The second part of this thesis investigates automatic category tracking. This problem includes automatic detection and tracking of all objects in the scene that is of a certain class. Such tracking is visualized in figure 1.1. This problem contains additional challenges compared to generic tracking. However, there are known information about the object class, that can be exploited to achieve more robust visual tracking. The question is then how to fuse this information into a visual tracking framework.

Visual tracking techniques can also be categorized by which object properties or states that are estimated. A visual tracker should be able to track the location of the object in the image. The location is often described with a two-dimensional image coordinate, i.e. two states. However, more general transformations than just pure translations can be considered. Many trackers attempt to also track the scale (i.e. size) of the object in the image. This adds another one or two states to be estimated, depending on if uniform or non-uniform scale is used. The orientation of the object can also be added as a state to be estimated. Even more general image transformations such as affine transformations or homographies can be considered. However, in general a state can be any property of interest, e.g. complex shape, locations of object parts and appearance modes.

1.1.1 Introducing Circulant Tracking by Detection with Kernels

This section gives a brief introduction to the CSK tracker [21], which is of importance to this thesis. It is studied in detail in chapter 2, but is introduced here since it is included in the problem formulation of this thesis. As mentioned earlier, methods that apply tracking by detection are becoming increasingly more popular. Most of these use some kind of sparse sampling strategy to harvest training examples for a discriminative classifier. The processing of each sample independently requires much computational effort when it comes to feature extraction, training and detection. It is clear that a set of samples contains redundant information if they are sampled with an overlap. The CSK exploits this redundancy for much faster computation.

The CSK tracker relies on a kernelized least squares classifier [7]. The task is to classify patches of an image region as the object or background. For simplicity, one-dimensional signals are considered here. Let $z$ be a sample from such a signal, i.e. $z \in \mathbb{R}^M$. Let $\phi: \mathbb{R}^M \mapsto H$ be a mapping from $\mathbb{R}^M$ to some Hilbert space $H$ [28]. Let $f$ be a linear classifier in $H$ given by (1.1), where $v \in H$.

$$f(z) = \langle \phi(z), v \rangle$$

(1.1)

The classifier is trained using a sample $x \in \mathbb{R}^M$ from the region around the object. The set of training examples consists of all cyclic shifts $x_m, m \in \{0, \ldots, M - 1\}$ of $x$. The classifier is derived using regularized least squares, which means minimizing the loss function (1.2), where $y \in \mathbb{R}^M$ contains the label for each example and $\lambda$ is a regularization


\[ \epsilon = \sum_{m=0}^{M-1} |f(x_m) - y_m|^2 + \lambda \|v\|^2 \quad (1.2) \]

The \( \psi \) that minimizes (1.2) can be written as a linear combination of the mapped training examples, \( \psi = \sum_k \alpha_k \phi(x_k) \). By using the kernel trick, a closed form solution can be derived. This is given by (1.3). See [7] for a complete derivation.

\[ \alpha = (K + \lambda I)^{-1} y \quad (1.3) \]

\( K \) is the kernel matrix, with the elements \( K_{ij} = \kappa(x_i, x_j) \). \( \kappa \) is the kernel function that defines the inner product in \( H \), \( \kappa(x, z) = \langle \phi(x), \phi(z) \rangle \), \( \forall x, z \in \mathbb{R}^M \). The classification of an example \( z \in \mathbb{R}^M \) is done using (1.4).

\[ \hat{y} = \sum_{m=0}^{M-1} \alpha_m \kappa(z, x_m) \quad (1.4) \]

The result generalizes to images. Let \( x \) be a grayscale image patch of the target. Define the kernelized correlation as \( u_x(m, n) = \kappa(x_m, x_n) \), where \( x_m, n \) are cyclic shifts of \( x \). Again, \( y(m, n) \) are the corresponding labels. Let capital letters denote the Discrete Fourier Transform (DFT) [16] of the respective two-dimensional signals. It can be shown that the Fourier transformed coefficients \( A \) of the kernelized least squares classifier can be calculated using (1.5), if the classifier is trained on the single image patch \( x \).

\[ A = \frac{Y}{U_x + \lambda} \quad (1.5) \]

The classification of all cyclic shifts of a grayscale image patch \( z \) can be written as a convolution, which is a product in the Fourier domain. The classification score \( \hat{y} \) of the image region \( z \) is computed with (1.6), where we have defined \( u_z(m, n) = \kappa(z_m, n, x) \).

\[ \hat{y} = \mathcal{F}^{-1}\{AU_z\} \quad (1.6) \]

A notable feature of this tracker is that most computations can be done using the Fast Fourier Transform (FFT) [16]. Thereby exceptionally low computational cost compared to most other visual trackers is obtained.

### 1.2 Thesis Overview

This section contains an overview of this thesis. The problem formulation is stated in section 1.2.1. Section 1.2.2 describes the motivation behind the thesis. Section 1.2.3 briefly presents the approaches, contributions and results.

#### 1.2.1 Problem Formulation

The goal of this master thesis is to research the area of visual tracking, with the focus on generic tracking and automatic category tracking. This shall be done through the following scheme.
1. Study the CSK tracker [21] and investigate how it can be improved. The goal is specifically to improve it with color features and new appearance learning methods.

2. Use the acquired knowledge to build a framework for causal automatic category tracking of humans in surveillance scenes. The main goal is to investigate how deformable parts models can be used in such a framework, to be able to track defined parts of a human along with the human itself.

1.2.2 Motivation

In the recent benchmark evaluation of generic visual trackers by Wu et al [46], the CSK tracker was shown to provide the highest speed among the top 10 trackers. Because of the simplicity of this approach, it holds the potential of being improved even further. The goal is to achieve state-of-the-art tracking performance at faster than real-time frame rates. Such a tracker has many interesting applications, for example in robotics, where the computational cost often is a major limiting factor. One other interesting example is scenarios where it is desired to track a large number of targets in real-time, for example in automated surveillance of crowded scenes.

Many generic trackers including the CSK, only rely on image intensity information and thus discard all color information present in the images. Although, the use of color information has proven successful in related computer vision areas, such as object detection and recognition [41, 26, 49, 42, 25], it has not been thoroughly investigated for tracking purposes yet. Changing the feature representation in a generic tracking framework often requires modifications of other parts in the framework as well. It is therefore necessary to look at the whole framework to avoid suboptimal ad hoc solutions.

In many major applications of visual tracking, the task is to track certain classes of objects, often humans. Safety systems in cars and automated surveillance are examples of such applications. Many existing category tracking frameworks in literature use pure data association of object detections, thus discarding most of the temporal information. Many recent frameworks also use global optimization over time windows in the sequence, thus disregarding the causality requirement which is existent in most practical applications. These are the motivations behind creating an automatic category tracking framework that is causal and thoroughly exploits the temporal dimension to achieve robust and high precision tracking.

Recently the deformable parts model detector [15] has been used in category tracking of humans. However, this has not yet been attempted in a causal framework. By jointly tracking an object and a set parts, a more correct deformable model can be applied that may increase accuracy and robustness. This might especially increase the robustness to partial occlusions, that is a common problem. Furthermore, part locations and trajectories is of interest in action detection and recognition, which are computer vision topics with related applications.

1.2.3 Approaches and Results

Three contributions are made to the CSK tracker. Firstly an appearance learning method is derived, which significantly improves the robustness and performance of this tracker.
evaluations the proposed method, named RCSK, is shown to outperform the original one when using multidimensional feature maps (e.g. color). Secondly an extensive evaluation of different color representations was done. This evaluation shows that color improves the tracking performance significantly and that the Color Names descriptor [43] is the best choice. Thirdly, an adaptive dimensionality reduction technique is proposed to reduce the feature dimensionality, thereby achieving a significant speed boost with a negligible effect on the tracking performance. This technique adaptively chooses the most important combinations of features.

Comprehensive evaluations are done to validate the performance gains of the proposed improvements. These include a comparison between a large number of different color representations for tracking. Lastly, the proposed generic visual tracker methods are compared to existing state-of-the-art methods in literature in an extensive evaluation. The proposed method is shown to outperform the existing methods, while operating at many times higher frame rates.

The second part of the thesis deals with the second goal in the problem formulation. A category tracking framework was built that combines generic tracking with object detection in a causal probabilistic framework with deformable part models. Specifically, the derived RCSK tracker was used in combination with the deformable parts model detector [15]. The Rao-Blackwellized Particle Filter [37] was used in the filtering step to achieve scalability in the number of parts in the model. The framework was applied to automatic tracking of multiple humans in surveillance scenes. The tracking results are demonstrated on a real-world benchmark sequence. Figure 1.1 illustrates the output of the proposed tracker framework.

1.3 Thesis Outline

This thesis report is organized into two parts. The first part is dedicated to generic tracking. Chapter 2 discusses the family of circulant structure trackers, including the CSK tracker introduced in section 1.1.1. The proposed appearance learning scheme for these trackers is derived in section 2.3. Chapter 3 discusses different color features for tracking and how the proposed tracker is extended with color information. In chapter 4, the proposed adaptive dimensionality reduction technique is derived and integrated to the tracking framework. The evaluations, results and conclusions from the first part of the thesis is presented in chapter 5. This includes the extensive comparison with state-of-the-art methods from literature.

The second part of this report considers the category tracking problem. Chapter 6 gives an overview of the system and presents the model on which it is built. Chapter 7 describes in detail how DPM object detector is used. Additional details are discussed in chapter 8, including how the generic tracking method derived in the first part of this thesis is incorporated. Finally, the results are discussed in chapter 9.

The appendix contains two parts. Appendix A summarizes the Bayesian filtering problem and most importantly describes the RBPF. Appendix B contains mathematical proofs and derivations of the most important results.
Part I

Generic Tracking
Circulant Tracking by Detection

A standard correlation filter is a simple and straightforward visual tracking approach. Much research over the last decades have aimed at producing more robust filters. Most recently the Minimum Output Sum of Squared Error (MOSSE) filter [8] was proposed. It performs comparably to state-of-the-art trackers, but at hundreds of FPS. In [21], this approach was formulated as a tracking-by-detection problem and kernels was introduced into the framework. The resulting CSK tracker was presented briefly in section 1.1.1. This chapter starts with a detailed presentation of the MOSSE and CSK trackers. In section 2.3 a new learning scheme for these kinds of trackers is proposed.

2.1 The MOSSE Tracker

The key to fast correlation filters is to avoid computing the correlations in the spatial domain, but instead exploiting the $O(P \ln(P))$ complexity of the FFT. However, this assumes a periodic extension of the local image patch. Obviously this assumption is a very harsh approximation of reality. However, since the background is of much lesser importance, this approximation can be seen as valid if the tracked object is centered enough in the local image patch.

2.1.1 Detection

The goal in visual tracking is to find the location of the object in each new frame. Initially only monochromatic images are considered, or more generally two dimensional, discrete and scalar valued signals, i.e. functions $\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R}$. To avoid special notation for circular convolution and correlation, it is always assumed that a signal is extended periodically. The set of all periodic functions $f : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R}$ with period $M$ in the first argument and period $N$ in the second argument is denoted $\ell_p(M, N)$. The periodicity means that $f(m + M, n + N) = f(m, n), \forall m, n \in \mathbb{Z}$. 
Let $z \in \ell_p(M,N)$ be the periodic extension of an image patch of size $M \times N$. $h \in \ell_p(M,N)$ is a correlation filter that has been trained on the appearance of a specific object. The correlation result at the image patch $z$ can be calculated using (2.1). The position of the target can then be estimated as the location of the maximum correlation output.

$$\hat{y} = h \ast z = \mathcal{F}^{-1}\{HZ\}$$  \hspace{1cm} (2.1)

Capital letters denote the DFT of the corresponding signals. The second equality in (2.1) follows from the correlation property of the DFT. The next section deals with how to train the correlation filter $h$.

### 2.1.2 Training

First consider the simplest case. Given an image patch $x \in \ell_p(M,N)$ that is centred at the object of interest, the task is to find the correlation filter $h \in \ell_p(M,N)$ that gives the output $y \in \ell_p(M,N)$ if correlated with $x$. $y$ can simply be a Kronecker delta centered at the target, but it proves to be more robust to use a smooth function, e.g. a sampled Gaussian. The goal is to find a $h$ that satisfies $h \ast x = y$. If all frequencies in $x$ contain non-zero energy there is a unique solution given by $H = \sum X$.

In practice it is important to be able to train the filter using multiple image samples $x^1, \ldots, x^J$. These samples can originate from different frames. Let $y^1, \ldots, y^J$ be their corresponding desired output functions (or label functions). $h$ is found by minimizing:

$$\epsilon = \sum_{j=1}^J \beta_j \|h \ast x^j - y^j \|^2 + \lambda \|h\|^2$$  \hspace{1cm} (2.2)

Here $\beta_1, \ldots, \beta_J$ are weight parameters for the corresponding samples and $\lambda$ is a regularization parameter. The filter that minimizes (2.2) is given in (2.3). See [8] for the derivation.

$$H = \frac{\sum_{j=1}^J \beta_j Y^j X^j}{\sum_{j=1}^J \beta_j X^j X^j + \lambda}$$  \hspace{1cm} (2.3)

Equation 2.3 suggests updating the numerator $H_N$ and denominator $H_D$ of $H$ in each new frame using a learning parameter $\gamma$. If $H^{t-1} = \frac{H^{t-1}_N}{H^{t-1}_D} + \lambda$ is the filter updated in frame $t - 1$ and $x^t, y^t$ are the new sample and desired output function from frame $t$, then the filter is updated as in (2.4). This is the core part of the MOSSE tracking algorithm in [8].

$$H_N^t = (1 - \gamma)H_N^{t-1} + \gamma Y^t X^t$$  \hspace{1cm} (2.4a)

$$H_D^t = (1 - \gamma)H_D^{t-1} + \gamma X^t X^t$$  \hspace{1cm} (2.4b)

$$H^t = \frac{H_N^t}{H_D^t + \lambda}$$  \hspace{1cm} (2.4c)

This update scheme results in the weights given in (2.5).

$$\beta_j = \begin{cases} (1 - \gamma)^{t-1} & , j = 1 \\ \gamma(1 - \gamma)^{t-j} & , j = 2, \ldots, t \end{cases}$$  \hspace{1cm} (2.5)
2.2 The CSK Tracker

This section discusses the CSK tracker, which was briefly presented in section 1.1.1. The CSK tracker can be obtained by extending the MOSSE tracker with a non-linear kernel. This extension is accomplished by introducing a mapping \( \phi : \mathcal{X} \mapsto H \) from the signal space \( \mathcal{X} \) to some Hilbert space \( H \) and exploiting the kernel trick [7]. The result is also generalized to vector valued signals \( f : \mathbb{Z} \times \mathbb{Z} \mapsto \mathbb{R}^D \), to handle multiple features. The set of such periodic signals is denoted \( \ell_p^D(M, N) \). The individual components of \( f \) are denoted \( f^d, d \in \{1, \ldots, D\} \), where \( f^d \in \ell_p^2(M, N) \).

2.2.1 Training with a Single Image

Let \( \langle \cdot, \cdot \rangle \) be the standard inner product in \( \ell_p^2(M, N) \). Let \( x_{m,n} = \tau_{-m,-n}x \) be the result of shifting \( x \in \ell_p^2(M, N) \) \( m \) and \( n \) steps, so that \( x_{m,n}(k,l) = x(k+m, l+n) \). Note that \( h \star x(m, n) = \langle x_{m,n}, h \rangle \). The cost function in (2.2) can for the case of a single training image \( (J = 1) \) be written as in (2.6).

\[
\epsilon = \sum_{m,n} \left( \langle x_{m,n}, h \rangle - y(m, n) \right)^2 + \lambda \langle h, h \rangle \tag{2.6}
\]

The sum in (2.6) is taken over a single period. The equation can be further generalized by considering the mapped examples \( \phi(x_{m,n}) \). The decision boundary is obtained by minimizing (2.7) over \( v \in H \).

\[
\epsilon = \sum_{m,n} \left( \langle \phi(x_{m,n}), v \rangle - y(m, n) \right)^2 + \lambda \langle v, v \rangle \tag{2.7}
\]

Observe that this is the cost function for regularized least squares classification with kernels. A well known result from classification is that the \( v \) that minimizes (2.7) is in the subspace spanned by the vectors \( (\phi(x_{m,n}))_{m,n} \). This result is easy to show in this case by decomposing any \( v \) to \( v = v_\parallel + v_\perp \), where \( v_\parallel \) is in this subspace and \( v_\perp \) is orthogonal to it. The result can be written as in (2.8) for some scalars \( a(m, n) \).

\[
v = \sum_{m,n} a(m,n) \phi(x_{m,n}) \tag{2.8}
\]

The inner product in \( H \) is defined by the kernel function \( \kappa(f,g) = \langle \phi(f), \phi(g) \rangle, \forall f, g \in \mathcal{X} \). The coefficients \( a(m,n) \) are found by minimizing (2.9), where we have transformed (2.7) by expressing \( v \) using (2.8) and used the definition of the kernel function.

\[
\epsilon = \sum_{m,n} \left( \sum_{k,l} a(k,l) \kappa(x_{m,n}, x_{k,l}) - y(m, n) \right)^2 + \lambda \sum_{m,n} a(m,n) \sum_{k,l} a(k,l) \kappa(x_{m,n}, x_{k,l}) \tag{2.9}
\]

A closed form solution to (2.9) can be derived under the assumption of a shift invariant kernel. The concept of shift invariant kernels is defined in section 2.2.4. The coefficients \( a \) can be extended periodically to an element in \( \ell_p^2(M, N) \). The \( a \) that minimizes (2.9)
is given in (2.10). A derivation using circulant matrices can be found in [21], but it is also proved in section B.1.2 for a more general case. Here we have defined the function $u_x(m,n) = \kappa(x_{m,n}, x)$. It is clear that $u_x \in \ell_p(M,N)$.

$$A = \mathcal{F}\{a\} = \frac{Y}{U_x + \lambda} \quad (2.10)$$

This is the same result as in (1.5), which the original CSK tracker [21] builds upon.

### 2.2.2 Detection

The calculation of the detection results of the image patch $z$ is similar to (2.1). Here, $x$ is the learnt appearance of the object and $A$ is the DFT of the learnt coefficients. By defining $u_z(m,n) = \kappa(z_{m,n}, x)$, the output can be computed using (2.11).

$$\hat{y} = \mathcal{F}^{-1}\{AU_z\} \quad (2.11)$$

### 2.2.3 Multidimensional Feature Maps

The equations 2.10 and 2.11 can be used for any feature dimensionality. The task is just to define a shift invariant kernel function that can be used for multidimensional features. One example of such a kernel is the standard inner product in $\ell^D_p(M,N)$, i.e. $\kappa(f,g) = \langle f, g \rangle$. Let $x^d$ denote feature layer $d \in \{1, \ldots, D\}$ of $x$. The training and detection in this case can be derived from equations 2.10 and 2.11. The result is given in (2.12). This is essentially the MOSSE tracker for multidimensional features, trained on a single image.

$$H^d = Y \frac{X_d^d}{\sum_{d=1}^D X_d^d} + \lambda \quad (2.12a)$$

$$\hat{y} = \mathcal{F}^{-1}\left\{ \sum_{d=1}^D H^d Z^d \right\} \quad (2.12b)$$

### 2.2.4 Kernel Functions

The kernel function is a mapping $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that is symmetric and positive definite. $\mathcal{X}$ is the sample space, i.e. $\ell^D_p(M,N)$. The kernel function needs to be shift invariant for the equations 2.10 and 2.11 to be valid. This section contains the definition of a shift invariant kernel from [21] and theorems that need to be stated regarding this property.

#### 2.1 Definition (Shift Invariant Kernel).

A shift invariant kernel is a valid kernel $\kappa$ on $\ell^D_p(M,N)$ that satisfies

$$\kappa(f,g) = \kappa(\tau_{m,n}f, \tau_{m,n}g), \forall m, n \in \mathbb{Z}, \forall f, g \in \ell^D_p(M,N) \quad (2.13)$$

#### 2.2 Proposition. Let $\kappa$ be the inner product kernel in (2.14), where $k : \mathbb{R} \to \mathbb{R}$.

$$\kappa(f,g) = k(\langle f, g \rangle), \forall f, g \in \ell^D_p(M,N) \quad (2.14)$$
Then $\kappa$ is a shift invariant kernel. Further, the following relation holds.

$$
\kappa(\tau_{m,n}f, g) = k \left( \mathcal{F}^{-1} \left\{ \sum_{d=1}^{D} F^d G^d \right\} (m, n) \right), \forall m, n \in \mathbb{Z} \tag{2.15}
$$

\section{2.3 Robust Appearance Learning}

This section contains the description of my proposed extension of the CSK learning approach in (2.10) to support training with multiple images. It can also be seen as an extension of the MOSSE tracker to multiple features if a linear kernel is used. The result is a more robust learning scheme of the tracking model, which is shown to outperform the learning scheme of the CSK [21] in chapter 5. The tracker that is proposed in this section is therefore referred as Robust CSK or RCSK.

The CSK tracker learns its tracking model by computing $A$ using (2.10) for each new frame independently. It then applies an ad hoc method of updating the classifier coefficients by linear interpolation between the new coefficients $A$ and the previous ones $A^{t-1}$ using: $A^t = (1 - \gamma)A^{t-1} + \gamma A$, where $\gamma$ is a learning rate parameter. Modifying the cost function to include training with multiple images is not as straightforward as with the MOSSE-tracker for grayscale images. This is due to the fact that the kernel function is non-linear in general. The equivalent to (2.2) in the CSK case would be to minimize:

$$
\epsilon = \sum_{j=1}^{J} \sum_{m,n} \beta_j \left( \langle \phi(x^j_{m,n}), v \rangle - y^j(m, n) \right)^2 + \lambda \langle v, v \rangle \tag{2.18}
$$

However, the solution $v = \sum_{j=1}^{J} \sum_{m,n} a^j (m, n) \phi(x^j_{m,n})$ involves computing a set of coefficients $a^j$ for each training image $x^j$. This requires an evaluation of all pairs of kernel outputs $u^j_x(m, n) = \kappa(x^j_{m,n}, x^j)$. All $A^j$ can then be computed by solving $NM$ number of $J \times J$ linear systems. This is obviously highly impractical in a real-time setting if the number of images $J$ is more than only a few. To keep the simplicity and speed of the MOSSE tracker, it is thus necessary to find some approximation of the solution to (2.18). Specifically, the appearance model should only contain one set of classifier coefficients $a$ to simplify learning and detection.
This can be accomplished by restricting the solution so that the coefficients \( a \) are the same for all images. This is expressed as the cost function in (2.19).

\[
\epsilon = \sum_{j=1}^{J} \beta_j \left( \sum_{m,n} |\langle \phi(x_{m,n}^j), v^j \rangle - y^j(m,n) |^2 + \lambda \langle v^j, v^j \rangle \right) \tag{2.19a}
\]

where, \( v^j = \sum_{k,l} a(k,l) \phi(x_{k,l}^j) \) \tag{2.19b}

The \( a \) that minimizes (2.19) is given in (2.20), where we have set \( u_j^x(m,n) = \kappa(x_{m,n}^j, x^j) \).

\[
A = \frac{\sum_{j=1}^{J} \beta_j Y^j U_x^j}{\sum_{j=1}^{J} \beta_j U_x^j(U_x^j + \lambda)} \tag{2.20}
\]

See section B.1.2 for the derivation. The object patch appearance \( \hat{x}^t \) is updated using the same learning parameter \( \gamma \). The final update rule is given in (2.21).

\[
A_N^t = (1 - \gamma) A_N^{t-1} + \gamma Y^t U_x^t \tag{2.21a}
\]

\[
A_D^t = (1 - \gamma) A_D^{t-1} + \gamma U_x^t(U_x^t + \lambda) \tag{2.21b}
\]

\[
A^t = \frac{A_N^t}{A_D^t} \tag{2.21c}
\]

\[
\hat{x}^t = (1 - \gamma) \hat{x}^t + \gamma x^t \tag{2.21d}
\]

The resulting weights \( \beta_j \) will be the same as in (2.5). See algorithm 2.1 for the complete pseudo code of the proposed RCSK tracker.

### 2.4 Details

This section discusses various details of the proposed tracker algorithm, including parameters and necessary preprocessing steps for feature extraction.

#### 2.4.1 Parameters

The label function \( y \) is as in [8, 21] set to the Gaussian function in (2.22). The standard deviation is proportional to the given target size \( s = (s_1, s_2) \), with a constant \( \sigma_y \). Since a constant label function \( y \) is used, its transform \( Y^t = Y = \mathcal{F}\{y\} \) can be precomputed.

\[
y(m,n) = \exp \left( -\frac{1}{2\sigma_y^2s_1s_2} \left( \left( \frac{m - M}{2} \right)^2 + \left( \frac{n - N}{2} \right)^2 \right) \right), \quad \text{for} \quad m \in \{0, \ldots, M-1\}, \ n \in \{0, \ldots, N-1\} \tag{2.22}
\]

The kernel \( \kappa \) is set to a Gaussian with a variance proportional to the dimensionality of the patches, with a constant \( \sigma^2_{\kappa} \). The kernel used in [21] is given in (2.23).

\[
\kappa(f,g) = \exp \left( -\frac{1}{\sigma^2_{\kappa}MN} \|f - g\|^2 \right) \tag{2.23}
\]
The proposed RCSK tracker.

**Input:**
- Sequence of frames: \( \{ I^1, \ldots, I^T \} \)
- Target position in the first frame: \( p_1 \)
- Target size: \( s \)
- Window function: \( w \)
- Parameters: \( \gamma, \lambda, \eta, \sigma \)

**Output:**
- Estimated target position in each frame: \( \{ p_1, \ldots, p_T \} \)

**Initialization:**
1. Construct label function \( y \) using (2.22) and set \( Y = \mathcal{F}\{y\} \)
2. Extract \( x^1 \) from \( I^1 \) at \( p_1 \)
3. Calculate \( u^1_x(m, n) = \kappa(x_{m,n}^1, x^1) \) using (2.15) or (2.17)
4. Initialize: \( A^1_N = Y U^1_x, A^1_D = U^1_x(U^1_x + \lambda), A^1 = A^1_N / A^1_D, \hat{x}^1 = x^1 \)

5. **for** \( t = 2 : T \) **do**
   - **Detection:**
     6. Extract \( z^t \) from \( I^t \) at \( p_{t-1} \)
     7. Calculate \( u^t_x(m, n) = \kappa(z_{m,n}^t, \hat{x}^{t-1}) \) using (2.15) or (2.17)
     8. Calculate correlation output: \( \hat{y}^t = \mathcal{F}^{-1}\{A^{t-1} U^t_x\} \)
     9. Calculate the new position \( p_t = \arg\max_p \hat{y}(p) \)
   - **Training:**
     10. Extract \( x^t \) from \( I^t \) at \( p_t \)
     11. Calculate \( u^t_x(m, n) = \kappa(x_{m,n}^t, x^t) \) using (2.15) or (2.17)
     12. Update the tracker using (2.21)
   **end for**

A padding parameter \( \eta \) decides the amount of background contained in the patches, so that \( (M, N) = (1 + \eta)s \). The regularization parameter \( \lambda \) can be set to almost zero in most cases if the proposed learning is used. But since the effect of this parameter proved to be negligible for small values, it is set to the same value as in [21] for a fair comparison. The optimal setting of the learning rate \( \gamma \) is highly dependant on the sequence, though a compromise can often be found if the same value is used for many sequences (as in the evaluations). The complete set of parameters and default values is presented in table 2.1. The default values are the ones suggested by [21].

### 2.4.2 Windowing

As noted earlier, the periodic assumption is the key to be able to exploit the FFT in the computations. However, this assumption introduces discontinuities at the edges.\(^2\) A common technique from signal processing to overcome this problem is windowing, where

\(^2\)Continuity is not defined for functions with discrete domains. However, we can think of the domain as continuous for a moment, i.e. as before the signal was sampled
<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma$</td>
<td>0.075</td>
<td>Learning rate.</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>0.01</td>
<td>Regularization parameter.</td>
</tr>
<tr>
<td>$\eta$</td>
<td>1.0</td>
<td>Amount of background included in the extracted patches.</td>
</tr>
<tr>
<td>$\sigma_y$</td>
<td>1/16</td>
<td>Standard deviation of the label function $y$.</td>
</tr>
<tr>
<td>$\sigma_\kappa$</td>
<td>0.2</td>
<td>Standard deviation of the gaussian kernel function $\kappa$.</td>
</tr>
</tbody>
</table>

**Table 2.1:** The parameters for the RCSK and CSK tracker.

the extracted sample is multiplied by a window function. [21] suggests a Hann window, defined in (2.24).

$$w(m, n) = \sin^2 \left( \frac{\pi m}{M-1} \right) \sin^2 \left( \frac{\pi n}{N-1} \right)$$ (2.24)

In the detection stage of the tracking algorithm, an image patch $z$ is extracted from the new frame. However, it is not likely that the object is centred in the patch. This means that the window function distorts the object appearance. This effect becomes greater the further away from the center of the patch the object is located. This means that the windowing also effects the tracking performance in a negative way. The simplest ways to counter this effect is to iterate the detection step in the algorithm, where each new sample is extracted from the previously estimated position in each iteration. Although this often increases the accuracy of the tracker, it significantly increases the computational time. It can also make the tracking more unstable. Another option is to predict the position of the object in the next frame in a more sophisticated way, instead of just assuming constant position. This can be done by applying a Kalman filter on a constant velocity or acceleration model.

### 2.4.3 Feature Value Normalization

For image intensity features, [8, 21] suggest normalizing the values to the range $[-0.5, 0.5]$. The reason for this is to minimize the amount of distortion induced by the windowing operation discussed in the previous section. The idea is to remove as much of the inherent bias in the feature values as possible by subtracting some *a priori* mean feature value. The same methodology can be applied to other kinds of features.

One way of eliminating the need of choosing the normalization of each feature, is to automatically learn a normalization constant (that is subtracted from the feature value) based on the specific image sequence or even the specific frame. This however, has to be done with care to avoid corrupting the learnt appearance and classifier coefficients. A method for adaptively selecting the normalization constant based on the weighted average feature values was tried, but no significant performance gain was observed compared to using the ad-hoc *a priori* mean feature values. So it was not investigated further.

A special feature normalization scheme for features with a probabilistic representation (e.g. histograms) is presented in section 3.2.1.
Most state-of-the-art trackers either rely on intensity or texture information [19, 50, 24, 13, 38], including the CSK and MOSSE trackers discussed in the previous chapter. While significant progress has been made to visual tracking, the use of color information has been limited to simple color space transformations [35, 31, 10, 32, 11]. However, sophisticated color features have shown to significantly improve the performance of object recognition and detection [41, 26, 49, 42, 25]. This motivates an investigation of how color information should be used in visual tracking.

Exploiting color information for visual tracking is a difficult challenge. Color measurements can vary significantly over an image sequence due to variations in illuminant, shadows, shading, specularities, camera and object geometry. Robustness with respect to these factors have been studied in color imaging, and successfully applied to image classification [41, 26], and action recognition [27]. This chapter presents the color features that are evaluated in section 5.3 and discusses how they are incorporated into the family of circulant structure trackers presented in chapter 2.

### 3.1 Evaluated Color Features

In this section, 11 color representations are presented briefly. These are evaluated in section 5.3 with the proposed tracking framework. Each color representation uses a mapping from local RGB-values to a color space of some dimension. All color features evaluated here except Opponent-Angle and SO use pixelwise mappings from one RGB-value to a color value.

**RGB**: As a baseline, the standard 3-channel RGB color space is used.

**LAB**: The 3-dimensional LAB color space is perceptually uniform, meaning that colors at
equal distance are also perceptually considered to be equally far apart. The \( L \)-component approximates the human perception of lightness.

**YCbCr**: YCbCr contains a luminance component \( Y \) and two chrominance components \( Cb \) and \( Cr \) which encodes the blue- and red-difference respectively. The representation is approximately perceptually uniform. It is commonly used in image compression algorithms.

\( rg \): The \( rg \) [17] color channels are computed as \( (r, g) = \left( \frac{R}{R+G+B}, \frac{G}{R+G+B} \right) \). They are invariant with respect to shadow and shading effects.

**HSV**: In the HSV color space \( V \) encodes the lightness as the maximum RGB-value, \( H \) is the hue and \( S \) is the saturation, which corresponds to the purity of the color. \( H \) and \( S \) are invariant to shadow-shading. The hue \( H \) is additionally invariant for specularities.

**Opponent**: The opponent color space is an orthonormal transformation of the RGB-color space, given by (3.1).

\[
\begin{pmatrix}
O_1 \\
O_2 \\
O_3
\end{pmatrix} = \begin{pmatrix}
\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\
\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\
\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}}
\end{pmatrix} \begin{pmatrix}
R \\
G \\
B
\end{pmatrix}.
\]  

(3.1)

This representation is invariant with respect to specularities.

\( C \): The \( C \) color representation [41] adds photometric invariants with respect to shadow-shading to the opponent descriptor by normalizing with the intensity. This is done according to (3.2).

\[
C = \left( \frac{O_1}{O_3}, \frac{O_2}{O_3}, O_3 \right)^T
\]

(3.2)

**HUE**: The hue is a 36-dimensional histogram representation [42] of \( H = \arctan \left( \frac{O_1}{O_2} \right) \). The contribution to the hue histogram is weighted with the saturation \( S = \sqrt{O_1^2 + O_2^2} \) to counter the instabilities of the hue representation. This representation is invariant to shadow-shading and specularities.

**Opp-Angle**: The Opp-Angle is a 36-dimensional histogram representation [42] based on spatial derivatives of the opponent channels. The histogram is constructed using (3.3).

\[
\text{ang}_x^{O} = \arctan \left( \frac{O_{1x}}{O_{2x}} \right),
\]

(3.3)

The subscript \( x \) denotes the spatial derivative. This representation is invariant to specularities, shadow-shading, blur and a constant offset.

**SO**: SO is a biologically inspired descriptor of Zhang et al. [49]. This color representation is based on center surround filters on the opponent color channels.

**Color Names**: See section 3.1.1.
3.1.1 Color Names

The Color Names descriptor is explained in more detail, since it proved to be the best choice in the evaluation in section 5.3. It is therefore used in the proposed version of the tracker and in part two of this thesis. Color names (CN), are linguistic color labels assigned by humans to represent colors in the world. In a linguistic study performed by Berlin and Kay [6], it was concluded that the English language contains eleven basic color terms: black, blue, brown, grey, green, orange, pink, purple, red, white and yellow. In the field of computer vision, color naming is an operation that associates RGB observations with linguistic color labels. In this thesis, the mapping provided by [43] is used. Each RGB value is mapped to a probabilistic 11 dimensional color representation, which sums up to 1. For each pixel, the color name values represent the probabilities that the pixel should be assigned to the above mentioned colors. Figure 3.1 visualizes the color name descriptor in a real-world tracking example.

The color names mapping is automatically learned from images retrieved by the Google Image search. 100 example images per color were used in the training stage. The provided mapping is a lookup table from \(32^3 = 32768\) uniformly sampled RGB values to the 11 color name probabilities.

A difference from the other color descriptors mentioned in section 3.1 is that the color names encodes achromatic colors, such as white, gray and black. This means that it does not aim towards full photometric invariance, but rather towards discriminative power.

3.2 Incorporating Color into Tracking

In section 2.2.3 is was noted that the kernel formulation of the CSK and RCSK tracker makes it easy to extend the tracking algorithm to multidimensional features, such as color features. By using a linear kernel in these trackers, they can also be seen as different extensions of the MOSSE tracker to multidimensional features. The windowing operation discussed in section 2.4.2, is applied to every feature-layer separately after the feature extraction step, which in this case is a color space transformation followed by a feature normalization.

3.2.1 Color Feature Normalization

The feature normalization step, as described in section 2.4.3, is an important and non-trivial task to be addressed. For all color descriptors in section 3.1 with a non-probabilistic representation (i.e. all except HUE, Opp-Angle and Color Names), the normalization is done by centring the range of each feature value. This means that the range of the feature values are symmetric around zero. This is motivated by assuming uniform and independent feature value probabilities. However, the independence assumption is not valid for the high-dimensional color descriptors. For these descriptors it is more correct to normalize the representation so that the expected sum over the feature values is zero. For color names, this means subtracting each feature bin with \(\frac{1}{11}\).

A specific attribute of the family of trackers explained in chapter 2, including the proposed RCSK, opens up an interesting alternative normalization scheme that can be used...
with color names. It can in fact be used for any feature representation that sums up to some constant value. Color names contain only 10 degrees of freedom for this reason. The color name values lie in a 10-dimensional hyper plane in the feature space. This plane is orthogonal to the vector $(1, 1, \ldots, 1)^T$. The color name values can be centered by changing the feature space basis to an orthonormal basis chosen so that the last basis vector is orthogonal to this plane. However, since the last coordinate in the new basis is constant (and thus contains no information) it can be discarded. The feature dimensionality is thus reduced from 11 to 10 when this normalization scheme is used. This has a positive effect on the computational cost of the trackers, by reducing the number of necessary FFT-computations and memory accesses in each frame.

The nature of the trackers explained in chapter 2, makes them invariant to the choice of basis to be used in the normalization step. This comes from the fact that the inner products and $L^2$-norms that are used in the kernel computations, are invariant under unitary transformations of the feature values. This property is discussed further in section 4.2.2. To minimize the computational cost of this feature normalization step, a new lookup table was constructed that maps RGB-values directly to the 10-dimensional normalized color name values. In later chapters, these normalized color names is referred to as just color names. This means that this normalization scheme was always employed for color names in the experiments of chapter 5.
Adaptive Dimensionality Reduction

The time complexity of the proposed tracker in algorithm 2.1 scales linearly with the number of features. To overcome this problem, an adaptive dimensionality reduction technique is proposed in this chapter. This technique reduces the number of feature dimensions without any significant loss in tracking performance. The dimensionality reduction is based on Principal Component Analysis (PCA), which is described in section 4.1. Section 4.2 presents the theory behind the proposed approach. Section 4.3 contains implementation details and pseudo code of the approach and how it is applied to the trackers discussed in chapter 2.

4.1 Principal Component Analysis

PCA\(^1\) \([30]\) is a standard way of performing dimensionality reduction. It is done by computing an orthonormal basis for the linear subspace of a given dimension that holds the largest portion of the total variance in the dataset. The basis vectors are aligned so that the projections onto this basis are pairwise uncorrelated. From a geometric perspective, PCA returns an orthonormal basis for the subspace that minimizes the average squared \(L^2\)-error between a set of centered\(^2\) data points and its projections onto this subspace. This is formulated in (4.1).

\[
\begin{align*}
\min & \quad \varepsilon = \frac{1}{N} \sum_{i=1}^{N} \| x_i - BB^T x_i \|^2 \\
\text{subject to} & \quad B^T B = I
\end{align*}
\]

\(^{1}\)PCA is also known as the Discrete Karhunen-Loève Transform.

\(^{2}\)Here “centered” refers to that the average value has been subtracted from the data.
\( x_i \in \mathbb{R}^n \) are the centered data points and \( B \) is a \( n \times m \) dimensional matrix that contains the orthonormal basis vectors of the subspace in its columns. It can be shown that this optimization problem is equivalent to maximizing (4.2) under the same constraint. The covariance matrix \( C \) is defined as \( C = \frac{1}{N} \sum_i x_i x_i^T \).

\[
V = \text{tr}(B^T C B)
\]  

(4.2)

The PCA-solution to this problem is to choose the columns of \( B \) as the normalized eigenvectors of \( C \) that correspond to the largest eigenvalues (see [30] for the proof). It should be mentioned that any orthonormal basis to the subspace spanned by these eigenvectors is a solution to the optimization problem (4.1).

### 4.2 The Theory Behind the Proposed Approach

The proposed dimensionality reduction in this section is a mapping to a linear subspace of the feature space. This subspace is defined by an orthonormal basis. Let \( B_t \) denote the matrix containing the orthonormal basis vectors of this subspace as columns. Assume that the feature map of the appearance \( \hat{x}^t \in \ell_{D_1}^p(M,N) \) at frame \( t \) has \( D_1 \) features and that the desired feature dimensionality is \( D_2 \). \( B_t \) should thus be a \( D_1 \times D_2 \) matrix. The projection to the feature subspace is done by the linear mapping \( \tilde{x}^t(m,n) = B_t^T \hat{x}^t(m,n) \), where \( \tilde{x}^t \) is the compressed feature map. This section presents a method of computing the subspace basis \( B_t \) to be used in the dimensionality reduction.

#### 4.2.1 The Data Term

The original feature map \( \hat{x}^t \) of the learnt patch appearance can be optimally reconstructed (in \( L^2 \)-sense) as \( B_t \hat{x}^t = B_t B_t^T \hat{x}^t \). An optimal projection matrix can be found by minimizing the reconstruction error of the appearance in (4.3).

\[
\min \varepsilon_{\text{data}}^t = \frac{1}{MN} \sum_{m,n} \| \hat{x}^t(m,n) - B_t B_t^T \hat{x}^t(m,n) \|^2
\]  

(4.3a)

subject to \( B_t^T B_t = I \)  

(4.3b)

Equation 4.3a can be seen as a data term since it only regards the current object appearance. The expression can be simplified to (4.4) by introducing the data matrix \( X_t \) which contains all pixel values of \( \hat{x}^t \), such that there is a column for each pixel and a row for each feature. \( X_t \) thus has the dimensions \( D_1 \times MN \). The second equality follows from the properties of the *Frobenius* norm and the *trace* operator. The covariance matrix \( C_t \) is defined by \( C_t = \frac{1}{MN} X_t X_t^T \).

\[
\varepsilon_{\text{data}}^t = \frac{1}{MN} \| X_t - B_t B_t^T X_t \|_F^2 = \text{tr}(C_t) - \text{tr}(B_t^T C_t B_t)
\]  

(4.4)

#### 4.2.2 The Smoothness Term

The projection matrix must be able to adapt to changes in the target and background appearance. Otherwise it would likely become outdated and the tracker would deteriorate
over time since valuable information is lost in the feature compression. However, the projection matrix must also take the already learnt appearance into account. If it changes too drastically, the already learnt classifier coefficients $A_{t-1}$ become irrelevant since they were computed with a seemingly different set of features. The changes in the projection matrix must thus be slow enough for the already learnt model to remain valid.

To obtain smooth variations in the projection matrix, a smoothness term is added to the optimization problem. This term adds a cost if there is any change in the subspace spanned by the column vectors in the new projection matrix compared to the earlier subspaces. This is motivated by studying the transformations between these subspaces. Let $B_t$ be the ON-basis for the new subspace and $B_j$ for some earlier subspace ($j < t$) of the same dimension. The optimal transformation from the older to the new subspace is given by $P = B_t^T B_j$. It can be shown that the matrix $P$ is unitary if and only if the column vectors in $B_t$ and $B_j$ span the same subspace. One can easily verify that the point wise transformation by a unitary matrix corresponds to a unitary operator $U$ on $\ell_p^D(M,N)$. Lastly, one can see that inner product kernels and radial basis function kernels are invariant to unitary transforms, i.e. $\kappa(Uf,Ug) = \kappa(f,g)$. The kernel output is thus invariant under changes in the projection matrix as long as the spanned subspace stays the same. A cost should only be added if the subspace itself is changed. Equation 4.5 accomplishes this.

$$
\varepsilon_j^{\text{smooth}} = \sum_{k=1}^{D_2} \lambda_k \left\| b_j^{(k)} - B_t B_j^T b_j^{(k)} \right\|^2 \tag{4.5}
$$

$b_j^{(k)}$ is column vector $k$ in $B_j$. The positive constants $\lambda_j^{(k)}$ are used to weight the importance of each basis vector $b_j^{(k)}$. Equation 4.5 minimizes the squared $L^2$-distance of the error when projecting the old basis vectors onto the new subspace. The cost becomes zero if the two subspaces are the same (even if $B_t$ and $B_j$ are not) and is at a maximum if the subspaces are orthogonal. By defining the diagonal matrix $\Lambda_j$ with the weights along the diagonal $[\Lambda_j]_{k,k} = \lambda_j^{(k)}$, this expression can be rewritten to (4.6).

$$
\varepsilon_j^{\text{smooth}} = \text{tr}(\Lambda_j) - \text{tr}(B_t^T B_j \Lambda_j B_j^T B_t) \tag{4.6}
$$

### 4.2.3 The Total Cost Function

Assume that the tracker is currently on frame number $t$. Let $\hat{x}^t$ be the learnt feature map of the object appearance. The goal is to find the optimal projection matrix $B_t$ for the current frame. The set of previously computed projection matrices $\{B_1, \ldots, B_{t-1}\}$ are given. $B_t$ is found by minimizing (4.7), under the constraint $B_t^T B_t = I$.

$$
\varepsilon_t^{\text{tot}} = \alpha_t \varepsilon_t^{\text{data}} + \sum_{j=1}^{t-1} \alpha_j \varepsilon_j^{\text{smooth}}
$$

$$
= \alpha_t \left( \text{tr}(C_t) - \text{tr}(B_t^T C_t B_t) \right) + \sum_{j=1}^{t-1} \alpha_j \left( \text{tr}(\Lambda_j) - \text{tr}(B_t^T B_j \Lambda_j B_j^T B_t) \right) \tag{4.7}
$$

This cost function is the weighted sum of the data term (4.4) and the smoothness term in
(4.6) for each previous projection matrix $B_j$. $\alpha_j$ are importance weights. Equation 4.7 can be reformulated to the equivalent maximization problem (4.8) by exploiting the linearity of the trace-function.

$$V_{tot} = \text{tr} \left( B_t^T \left( \alpha_tC_t + \sum_{j=1}^{t-1} \alpha_jB_j\Lambda_jB_j^T \right) B_t \right) \quad (4.8)$$

By comparing this expression to the PCA-formulation (4.2) one can see that this optimization problem can be solved using the PCA methodology with the covariance matrix $R_t$ defined in (4.9). It can be verified that $R_t$ indeed is symmetric and positive definite.

$$R_t = \alpha_tC_t + \sum_{j=1}^{t-1} \alpha_jB_j\Lambda_jB_j^T \quad (4.9)$$

The columns in $B_t$ is thus chosen as the $D_2$ normalized eigenvectors of $R_t$ that corresponds to the largest eigenvalues.

### 4.3 Details of the Proposed Approach

The adaptive PCA algorithm described above requires a way of choosing the weights $\alpha_j$ and $\Lambda_j$. $\alpha_j$ control the relative importance of the current appearance and the previously computed subspaces. These are set by using an appropriate learning rate parameter $\mu$ that acts in the same way as the learning rate $\gamma$ for the appearance learning. Setting $\mu = 1$ corresponds to only using the current learnt appearance in the calculation of the projection matrix. $\mu = 0$ is the same as computing the projection matrix once in the first frame and then letting it be fixed for the entire sequence. The value was experimentally tuned to $\mu = 0.1$ for the linear kernel case and $\mu = 0.15$ for the non-linear kernel case.

The diagonal in $\Lambda_j$ contains the importance weights for each basis vector in the previously computed projection matrix $B_j$. These are set to the eigenvalues of the corresponding basis vectors in $B_j$. This makes sense since the score function (4.8) equals the sum of these eigenvalues. Each eigenvalue can thus be interpreted as the score for its corresponding basis vector in $B_t$. In a probabilistic interpretation, the eigenvalues are the variances for each component in the new basis. Since PCA uses variance as the measure of importance, it is natural to weight each component (basis vector) with its variance. The term $B_j\Lambda_jB_j^T$ then becomes the “reconstructed” covariance matrix of rank $D_2$, i.e. the covariance of the reconstructed appearance using the projections in image $j$. Equation 4.9 is thus a weighted sum of image covariances.

Algorithm 4.1 provides the full pseudo code for the computation of the projection matrix. The mean feature values do not contain information about the structure and should therefore be subtracted from the data before computing the projection matrix. Including the mean in the PCA computation affects the projection matrix to conserve the mean in the projected features, rather that maximizing the variance which is related to image structure.

Algorithm 4.2 provides the full pseudo code for the proposed RCSK tracker with adaptive
Algorithm 4.1 Adaptive projection matrix computation.

**Input:**
- Frame number: $t$
- Learned object appearance: $\hat{x}^t$
- Previous covariance matrix: $Q_{t-1}$
- Parameters: $\mu$, $D_2$

**Output:**
- Projection matrix: $B_t$
- Current covariance matrix: $Q_t$

1. Calculate mean $\bar{x}^t = \frac{1}{MN} \sum_{m,n} \hat{x}^t(m, n)$
2. Calculate covariance $C_t = \frac{1}{MN} \sum_{m,n} (\hat{x}^t(m, n) - \bar{x}^t)(\hat{x}^t(m, n) - \bar{x}^t)^T$
3. **if** $t = 1$ **then**
   4. Set $R_t = C_t$
5. **else**
   6. Set $R_t = (1 - \mu)Q_{t-1} + \mu C_t$
7. **end if**
8. Do EVD $R_t = E_t S_t E_t^T$, the eigenvalues in $S_t$ are in descending order
9. Set $B_t$ to the first $D_2$ columns in $E_t$
10. **if** $t = 1$ **then**
11. Set $Q_t = B_t \Lambda_t B_t^T$
12. **else**
13. Set $Q_t = (1 - \mu)Q_{t-1} + \mu B_t \Lambda_t B_t^T$
14. **end if**

Dimensionality reduction. Note that the windowing of the feature map is always done after the projection onto the new reduced feature space. It is not a part of the feature extraction as in algorithm 2.1. The reason is that windowing adds spatial correlation between the pixels, which contradicts the independence and stationarity assumptions used in the PCA.
Algorithm 4.2 Proposed RCSK tracker with dimensionality reduction.

**Input:**
- Sequence of frames: \( \{ I^1, \ldots, I^T \} \)
- Target position in the first frame: \( p_1 \)
- Target size: \( s \)
- Window function: \( w \)
- Parameters: \( \gamma, \lambda, \eta, \sigma, \kappa, \mu, D_2 \)

**Output:**
- Estimated target position in each frame: \( \{ p_1, \ldots, p_T \} \)

Initialization:
1. Construct label function \( y \) using (2.22) and set \( Y = \mathcal{F}\{y\} \)
2. Extract \( x^1 \) from \( I^1 \) at \( p_1 \)
3. Initialize \( \hat{x}^1 = x^1 \)
4. Calculate \( B_1 \) and \( Q_1 \) using algorithm 4.1
5. Project features and apply window: \( \tilde{x}^1(m, n) = w(m, n)B_1^T x^1(m, n) \)
6. Calculate \( u_x^1(m, n) = \kappa(\tilde{x}_{m,n}^1, \hat{x}^1) \) using (2.15) or (2.17)
7. Initialize: \( A^1_N = YU^1_x, A^1_D = U^1_x(U^1_x + \lambda), A^1 = A^1_N/A^1_D, \tilde{x}^1 = \hat{x}^1 \)

8. for \( t = 2 : T \) do

Detection:
9. Extract \( z^t \) from \( I^t \) at \( p_{t-1} \)
10. Project features and apply window: \( \tilde{z}^t(m, n) = w(m, n)B_t^T z^t(m, n) \)
11. Calculate \( u_x^t(m, n) = \kappa(\tilde{z}_{m,n}^t, \tilde{x}^{t-1}) \) using (2.15) or (2.17)
12. Calculate correlation output: \( \hat{y}^t = \mathcal{F}^{-1}\{A^{t-1} U_{z}^{t}\} \)
13. Calculate the new position \( p_t = \arg\max_p \hat{y}(p) \)

Training:
14. Extract \( x^t \) from \( I^t \) at \( p_t \)
15. Update appearance \( \hat{x}^t \) using (2.21d)
16. Calculate \( B_t \) and \( Q_t \) using algorithm 4.1
17. Project features and apply window: \( \tilde{x}^t(m, n) = w(m, n)B_t^T x^t(m, n) \)
18. Calculate \( u_x^t(m, n) = \kappa(\tilde{x}_{m,n}^t, \tilde{x}^t) \) using (2.15) or (2.17)
19. Update the tracker using (2.21a), (2.21b) and (2.21c)
20. Calculate projected appearance: \( \tilde{x}^t(m, n) = w(m, n)B_t^T \hat{x}^t(m, n) \)
21. end for
This chapter contains evaluations, results, discussions and conclusions related to the first part of this thesis. Section 5.1 describes the evaluation methodology, including evaluation metrics and datasets. In section 5.2, a comparison is made between the trackers presented in chapter 2. The color features discussed in chapter 3 are evaluated in section 5.3. The effect of the dimensionality reduction technique proposed in chapter 4 is investigated in section 5.4. The best performing proposed tracker versions is then compared to state-of-the-art methods in an extensive evaluation in section 5.5. Lastly, section 5.6 presents some general conclusions and discussions about possible directions of future work.

5.1 Evaluation Methodology

The methods were evaluated using the protocol and code recently provided by Wu et al. [46]. The evaluation code was modified with some bug fixes and some added functionality. It employs the most commonly used scheme for evaluating causal generic trackers on image sequences with ground-truth target locations. The tracker is initialized in the first frame, with the known target location. In the subsequent frames, the tracker is used to estimate the locations of the target. Only information from all the previous and the current frame may be exploited by the tracker when estimating a target location. The estimated trajectory is then compared with the ground truth locations using different evaluation metrics.

All evaluations were performed on a desktop computer with an Intel Xenon 2 core 2.66 GHz CPU with 16 GB of RAM.

---

1The sequences together with the ground-truth and matlab code are available at: https://sites.google.com/site/trackerbenchmark/benchmarks/v10
5.1.1 Evaluation Metrics

The trackers were evaluated using three evaluation metrics commonly used in literature. The first is average center location error (CLE), which is the average $L^2$-distance (in pixels) between the estimated and ground truth center locations of the target over the sequence. The second metric is distance precision (DP), which is the relative number of frames where the estimated center location is within a certain distance threshold $d$ from the ground truth center location. The third metric is overlap precision (OP), defined as the relative number of frames where the overlap between the estimated and ground truth bounding box exceeds a certain threshold $b$. Bounding box overlap is commonly measured using the PASCAL criterion. For an image sequence, the three measures are calculated using (5.1). The ground truth and estimated center locations are denoted $p_t$ and $\hat{p}_t$ respectively, where $t$ is the frame number. Similarly, $B_t$ and $\hat{B}_t$ denote the ground truth and estimated bounding boxes of the target. A bounding box is here defined as the set of pixels covered by the rectangular area. $N$ is the number of frames in the sequence.

\[
\text{CLE} = \frac{1}{N} \sum_{t=1}^{N} \|\hat{p}_t - p_t\| \tag{5.1a}
\]

\[
\text{DP}(d) = \frac{1}{N} \left| \{t : \|\hat{p}_t - p_t\| \leq d\} \right|, \quad d \geq 0 \tag{5.1b}
\]

\[
\text{OP}(b) = \frac{1}{N} \left| \left\{ t : \frac{\hat{B}_t \cap B_t}{\hat{B}_t \cup B_t} \geq b \right\} \right|, \quad 0 \leq b \leq 1 \tag{5.1c}
\]

There is not much agreement in the literature on what performance measures to use for comparing visual trackers. DP and CLE only regard the estimated center location, which is desirable when evaluating trackers that do not estimate scale. OP also takes the estimated scale into account. This metric was only used in the state-of-the-art evaluation (see section 5.5), to give a more complete and fair comparison to trackers that estimate scale variations.

Results of both CLE and DP are reported in the tables. The numeric values of distance precision are reported at a threshold of 20 pixels [21, 38, 46], i.e. DP(20). It is motivated to use both these metrics since they contain complementary information. CLE has the drawback of being unstable at tracker failures. DP is robust to failures, but it does not include any information about the accuracy of the tracker within 20 pixels. For robustness reasons, the per-video results are summarized using the median results over the whole dataset.

Some authors [21, 46] suggest the usage of precision and success plots, where distance and overlap precision respectively are plotted over a range of thresholds. This says much more about the performance, but makes the task of comparing different methods harder. In this chapter, precision and success plots are used to compare the overall performance of different trackers. The average precision values over the set of sequences were used. In both types of plots, a ranking score is computed to simplify the interpretation of the results. In the precision plots, the DP-value at 20 pixels is used. The area-under-the-curve (AUC) is used as the ranking score in the success plots. The ranking scores are displayed in brackets next to the tracker names in the legend of each plot. See [46] for more details.
5.1.2 Dataset

The benchmark evaluation of [46] includes a dataset of 50 image sequences. 35 of these are color sequences. Additionally, another 6 benchmark color sequences was added to the dataset, namely: Kitesurf, Shirt, Surfer, Board, Stone and Panda. The resulting set of 41 color sequences was used for all evaluations in this chapter, except in the initial comparison between MOSSE, CSK and RCSK on grayscale sequences. In this case, the full set of 56 sequences was used.

The sequences pose many challenging situations. [46] actually provides an annotation of their 50 sequences with 11 different attributes, which explain the challenges encountered in each sequence. The different attributes are: motion blur, illumination changes, scale variation, heavy occlusions, in-plane and out-of-plane rotations, deformation, out of view, background clutter and low resolution. This is used to make attribute based comparisons, which can show interesting strengths and weaknesses of different trackers.

5.1.3 Trackers and Parameters

All trackers are evaluated using the same parameters over the whole dataset. This requirement is commonly used to prevent over-tuning. For the trackers explained in chapter 2, including the proposed RCSK, the standard parameters suggested by [21] are used, including the same Gaussian kernel. The parameters and used values are displayed in Table 2.1. The kernel bandwidth $\sigma_k$, does not have any effect for the MOSSE tracker, or when a linear kernel is used for either RCSK or CSK. The dimensionality reduction learning rate $\mu$ is set to 0.1 for linear and 0.15 for non-linear kernels, when used with the RCSK.

The code for the proposed tracker versions was implemented in Matlab (no mex-functions were used). For the original CSK tracker, the Matlab-code provided by the authors was used but modified to support multidimensional feature maps. The evaluated MOSSE tracker was implemented in Matlab as well. Note that this is not exactly the same tracker as proposed in [8], but rather a simplification of it. There are no random initial examples and no failure detection (see [8]). The implementation of the MOSSE tracker that was evaluated here is similar to the implementation of the CSK and RCSK. The only differences are that it uses other equations for learning and updating the model, as well as calculating the tracking scores. Interestingly it was shown in [21] that a simple implementation of the core tracking functionality of the MOSSE tracker, i.e. (2.4) and (2.1), resulted in a better tracker than using the code provided by the authors.

In the state-of-the-art evaluation of section 5.5, the code or binaries for the compared trackers were either obtained from [46] or from the authors. They are used with the suggested default parameters.

5.2 Circulant Structure Trackers Evaluation

This section contains the comparison between different variants in the family of circulant structure trackers presented in chapter 2. This includes the proposed variant RCSK. Three experiments were done. Firstly RCSK, CSK [21] and MOSSE [8] were compared for grayscale features. Secondly the trackers were compared for multidimensional features,
specifically grayscale together with color names. Five trackers were evaluated in that experiment. The proposed robust learning scheme was used with both a linear and a Gaussian kernel. The methods are called RCS and RCSK respectively. Similarly, the learning scheme of [21] was used with a linear and a Gaussian kernel, called CS and CSK respectively. These were compared with a straightforward generalization of the MOSSE tracker to multidimensional features. In the third experiment, RCSK was compared with CSK for all the color features mentioned in section 3.1.

### 5.2.1 Grayscale Experiment

This experiment compares the RCSK with CSK and MOSSE when just using grayscale features. For this reason all the 56 sequences in the dataset are employed. The results are presented in figure 5.1. The same grayscale features as applied in the original CSK code provided by the authors was used for all trackers. For color sequences, these are computed using the rgb2gray-function in Matlab. The precision plot show a slight advantage for the linear kernel (i.e. the MOSSE tracker). The CSK and RCSK performs similarly.

Qualitatively, it can be seen that although the linear kernel provides better results in average, it has stability issues in situations with significant illuminations changes. This is most clearly seen in the shaking and skating1 sequences. In shaking there is drastically increasing back-light for a few frames. Figure 5.2 shows the frame where MOSSE fails and lose track of its target. The strong back-light and blooming in frame 59 compared to the previous frame completely corrupts the score function of the MOSSE tracker (figure 5.2b), while the other two trackers are able to track through this frame robustly (figure 5.2d and 5.2f). MOSSE fails in a similar way in skating1 when the target (an ice skater) moves from an illuminated area in the scene to a much darker area. Also in this case CSK and RCSK manage to track the target through those frames.
5.2 Circulant Structure Trackers Evaluation

(a) MOSSE, frame 58.  
(b) MOSSE, frame 59.  
(c) CSK, frame 58.  
(d) CSK, frame 59.  
(e) RCSK, frame 58.  
(f) RCSK, frame 59.

**Figure 5.2:** Frames 58 and 59 from the shaking sequence. The score function is shown in red and the tracking output bounding box in green. The tracked target in this sequence is the head of the guitarist. MOSSE fails in frame 59 (figure 5.2b) where the back-light from the spotlight in the background has increased significantly compared to the previous frame (figure 5.2a). The kernelized versions CSK (figure 5.2c and 5.2d) and RCSK (figure 5.2e and 5.2f) are able to track through these frames robustly.

5.2.2 Grayscale and Color Names Experiment

This experiment is similar to the one described in the previous section. The difference is that in addition to grayscale features, color names was used as well. The motivation for using color names here is that it is shown to be the best performing evaluated color descriptor in section 5.3. The evaluations in this experiment were done on the set of 41 color sequences.

Five trackers were evaluated in this experiment, of which two are versions of the proposed
Figure 5.3: Comparison between RCS, RCSK, CS, CSK and MOSSE for combined grayscale and color names features.

This means that RCS and RCSK use algorithm 2.1 with different kinds of kernels. By “linear kernel” it is meant that the standard inner product is applied as a kernel $\kappa(f, g) = \langle f, g \rangle$, as in the MOSSE tracker.

The results from the experiments are shown in figure 5.3. The best results were obtained by using the robust learning scheme, i.e. RCS and RCSK. It can clearly be seen that the learning scheme of [21] (CSK and CS) is suboptimal in this case. Further, there is a slight advantage in using a linear kernel over a Gaussian kernel.
5.3 Color Evaluation

In section 5.2 it was shown that the proposed learning scheme generally performs better for varying color features. The RCSK was therefore chosen for evaluating the different color representations. All color features in section 3.1 were included. These are also compared with using intensity features alone. As described in section 5.2.3, these intensity features (obtained by rgb2gray) were concatenated with the color representations with no inherent intensity channel (RGB, rg, HUE, Opp-Angle, SO and CN).\(^2\)

5.3.1 Results

The results from the experiment are shown in figure 5.5. Color names obtains the best results in general, with significantly better median CLE and DP values and better average distance precision in the precision plot. The Opp-Angle (AOpp) descriptor is the clear

\(^2\)It was noted that using the inherent intensity component, e.g. the L-component in LAB, gives better results than changing it to the usual intensity features (rgb2gray).

---

**Figure 5.4:** Comparison of original update scheme (CSK) with the proposed learning method (RCSK) using median distance precision (in percent).

---
Evaluation

Figure 5.5: Color evaluation results. The performance of the RCSK tracker was evaluated for the color features discussed in section 3.1 and intensity alone.

second best choice for high precision. These two color descriptors are then followed by the set of descriptors with different photometric invariances, namely HSV, LAB, HUE, C, RG and YCbCr. The simple opponent color transformation and the standard RGB representation give slight increase in performance compared to only intensity. Lastly, it can seen that the SO descriptor is not well suited for tracking purpose.

It should be noted that the Opponent-Angle (AOpp) descriptor encodes shape information as well as color. It is thus a powerful descriptor in scenarios where shape is more discriminative that color. The attribute based results are summarized in figures 5.6 and 5.7. AOpp outperform the other color descriptors in sequences with significant background clutter, while struggling in motion blur. AOpp and HSV perform better than color names in illumination variation due to more photometric invariance. CN proves to be the most robust descriptor at occlusions and blur. This evaluation show that some descriptors contain complementarity information, which indicates that even more powerful descriptors might be found by combining them in sophisticated ways, together with shape information.

5.3.2 Discussion

Three probable reasons for the success of color names over the other color descriptors can be identified. Firstly, it uses a continous representation in the sense that common distance measures (e.g. the Euclidian distance) makes sense, in the way that similar colors are close
Figure 5.6: Precision plots showing the results of the attribute-based evaluation of different color features with the RCSK. The plots display the distance precision for the evaluated attributes fast motion, background clutter, motion blur, deformation, illumination variation and in-plane rotation. The value appearing in the title denotes the number of videos associated with the respective attribute. The average distance precision at 20 pixels is displayed in the legends.
Figure 5.7: Precision plots showing the results of the attribute-based evaluation of different color features with the RCSK. The plots display the distance precision for the evaluated attributes: low resolution, occlusion, in-plane rotation, out of view, and scale variation. The value appearing in the title denotes the number of videos associated with the respective attribute. The average distance precision at 20 pixels is displayed in the legends.
to each other. This is also true for LAB, which is considered to be perceptually uniform, meaning that Euclidian distances in the colorspace reflects perceptual similarities. However, the hue component H in HSV should be interpreted as the angle in a cylindrical coordinate system, meaning that its maximum and minimum H-value correspond to the same hue. Secondly, color names is a probabilistic representation. The update scheme of the appearance template in (2.21d) can thus be interpreted as a statistical update of the color probabilities. However, for non probabilistic representations such as Lab, the template update of (2.21d) may produce colors that have not occurred in the examples. This increases the sensitivity of the tracker towards background clutter and occlusions. The third reason is that the color name representation is trained to categorize colors based on how humans do by language. Intuitively, this is a discriminative way of selecting basic colors.

Besides performance, two other attributes are important, namely compactness and computational cost. A feature descriptor should contain a minimal number of dimensions in relation to its discriminative power to reduce computation and memory costs. Here, compact color space representations such as HSV have a clear advantage. However, with the dimensionality reduction technique introduced in chapter 4, color names can be reduced to 2 dimensions without any significant performance loss. The major drawback of AOpp is its large computational cost, since it requires computations of derivatives among other things. The color names representation only requires the computation of indices for the lookup table, which can be done with simple integer arithmetic. The next step is just a memory access, which is fast for processors with reasonably sized caches. The median tracking frame rate is in fact over 10 times higher for color names compared to AOpp, while similar to HSV.

In conclusion it has been demonstrated that color information has the potential to drastically increase the tracking performance. However, the choice of color descriptor is crucial. Simple representations such as RGB and opponent give only a negligible improvement compared to using only grayscale features. The standard photometric invariant descriptors are significantly better suited for the tracking task. However, color names is the best choice for the RCSK. Not only because it gives the best overall results, it is also surprisingly inexpensive to compute.

5.4 Adaptive Dimensionality Reduction Evaluation

This section evaluates the impact of the dimensionality reduction technique presented in chapter 4. This is a general technique, that can be applied for any types of features for the RCSK tracker. However, since color names was shown to be the generally best color descriptor for the RCSK in section 5.3, it was only applied to that descriptor in these experiments. The goal is thus to create a more compact color name representation, without any significant performance loss.

The evaluated tracker versions that included the dimensionality reduction was implemented as algorithm 4.2. The color names features are compressed independently. This proved to give better results than to compress the intensity channel together with the color names. The learning rate parameter $\mu$ for the dimensionality reduction is set to
5.4.1 Number of Feature Dimensions

The RCSK (i.e with a Gaussian kernel) was used to evaluate the impact of the number of reduced dimensions. The normalized color names have $D_1 = 10$ dimensions. In this experiment they were compressed to $D_2 = 1, 2, \ldots, 9$ dimensions using algorithm 4.2. This is compared to no compression at all, i.e. RCSK with intensity and color names (I+CN) and using zero dimensions, i.e. RCSK with only intensity.

The results are displayed in figure 5.8. The compressed feature representation is named I+CN$D_2$ where $D_2$ is the compressed color name dimension. From the results of this experiment, it is clear that no significant gain is obtained by using more than two dimensions. However, using only one dimension gives inferior results, while hardly increasing the tracker speed. Thus, CN2 is chosen for the final representation. Note that CN needs 0.1 for RCS (linear kernel) and 0.15 for RCSK (Gaussian kernel). Two experiments are presented in this section. The first experiment investigates the impact of the number of output feature dimensions. The second experiment evaluates the effect of the dimensionality reduction on RCS and RCSK from section 5.2.2 with color names. Computational cost and performance is of equal interest in these experiments.
5.5 State-of-the-Art Evaluation

This section presents an extensive evaluation of the proposed trackers and state-of-the-art methods that exist in literature. Two proposed versions are compared to the existing methods, namely RCS with intensity and color names (RCS CN) and RCS with intensity and compressed color names (RCS CN2). The “I” in the naming convention is dropped. The Gaussian kernel versions are omitted since they proved to be inferior in section 5.4.2.

5.4.2 Final Performance

In this experiment the RCS and RCSK with intensity and color names (I+CN) (as in section 5.2.2) are compared with the respective trackers when color names are compressed to 2 dimensions (I+CN2), which was found to be optimal in the previous experiment. The results are shown in figure 5.9. The performance loss is minor in the precision plot (average distance precision). But the speed gain is 45% for RCS and 33% for RCSK, which is significant.

Figure 5.9: Comparison between the color names and compressed color names for RCS and RCSK.

to be compressed to 6 or fewer dimensions to overcome the computational overhead introduced by the dimensionality reduction. In general this depends on the target size though.

5.5 State-of-the-Art Evaluation

Table of the median CLE (in pixels), DP (in percent) and frame rate (in FPS) over all the sequences. The two best results are displayed in red and blue fonts respectively.

<table>
<thead>
<tr>
<th></th>
<th>RCS I+CN</th>
<th>RCS I+CN2</th>
<th>RCSK I+CN</th>
<th>RCSK I+CN2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median CLE</td>
<td>13.3</td>
<td>15.3</td>
<td>13.8</td>
<td>14.3</td>
</tr>
<tr>
<td>Median DP</td>
<td>83.8</td>
<td>75.7</td>
<td>81.4</td>
<td>79.3</td>
</tr>
<tr>
<td>Median FPS</td>
<td>94.1</td>
<td>136</td>
<td>78.9</td>
<td>105</td>
</tr>
</tbody>
</table>

(a) Precision plot.
The proposed methods are compared with 20 trackers that exist in literature. These are: CT [50], TLD [24], DFT [38], EDFT [14], ASLA [23], L1APG [4], CSK [21], SCM [52], LOT [32], CPF [35], CXT [13], Frag [1], IVT [36], ORIA [45], MTT [51], BSBT [40], MIL [3], Struck [19], LSHT [20] and LSST [44]. These methods include the top four performing trackers in the recent benchmark evaluation [46], namely Struck, SCM, TLD and CXT. Also ASLA, CSK, DFT and L1APG were among the top trackers in this evaluation. EDFT, LSHT and LSST are recent trackers in the literature, that were not included in the benchmark evaluation. Again, all 41 color sequences are used. The results are presented using both precision plots (distance precision) and success plots (overlap precision).

The overall results are presented in figure 5.10. The two proposed trackers outperform or perform favorably compared to the other evaluated methods in all evaluation metrics. Struck is the best performing method of the compared existing trackers. It uses powerful learning methods, namely a kernelized structured output support vector machine. So it is noteworthy that it is outperformed by tracker using a simpler learning method, namely a modified least squared classifier. It should also be noted that neither Struck nor RCS estimate scale variations. For this reason, ASLA and SCM which use an affine tracking model, obtain better overlap precisions at high overlap thresholds \( b > 0.7 \). However the robustness of these trackers seems to be far less than those of RCS and Struck.

The computational cost is an other important aspect of this evaluation. It should be noted that the RCS versions run at an order of magnitude higher frame rate than Struck (which is a C++ implementation). RCS CN2 runs at the second highest frame rate in median, only 10\% below CSK. Also CPF and CT obtain notable frame rates, but provide inferior performance. ASLA and SCM perform rather well in this evaluation, but they are not feasible for real-time applications.

The precision plots of the attribute based results are shown in figures 5.11 and 5.12. The corresponding success plots are shown in figures 5.13 and 5.14. The proposed trackers perform favourably in most of these attributes. The results are especially good in attributes that are related to appearance changes, namely motion blur, deformation, illumination variation, in-plane-rotation, occlusion and out-of-plane-rotation. The reason for this is probably a combination between robust learning and robust features. Struck performs better in sequences with fast-motion. This is related to the negative effects of the windowing operation discussed in section 2.4.2. It should be mentioned that ASLA and SCM naturally have an advantage in sequences with large scale variations since they are able to estimate this property.

5.6 Conclusions and Future Work

In this chapter it has been shown that the proposed RCS and RCSK trackers perform favorably to state-of-the-art trackers on a large number of benchmark sequences. Especially, their performance and speed combined is unmatched among the evaluated trackers. The framework is however still quite simple, so further improvements are expected to be

\(^3\)Unfortunately, code or binaries could not be obtained for all top 10 trackers from this comparison.
5.6 Conclusions and Future Work

There are a number of issues that could be addressed for future development. In section 5.3 it was noted that it should be possible to combine color and shape descriptors to obtain better discrimination of the target. The possibility of feature selection also comes in here, which would provide a way of selecting the most discriminative feature combinations. This is related to the dimensionality reduction presented in chapter 4. The current method basically uses the amount of structure or variance to determine the importance of different feature combinations. One other option would be to do feature selec-
tion/reduction based on supervised techniques, to aim at selecting the most discriminative feature combinations.

One factor, that is limiting in some situations, is that the RCS and RCSK trackers do not estimate scale. This limitation should be addressed in future research, since it may give a significant overall performance gain as well. This is motivated by the success of ASLA and SCM in sequences with scale variation (see figure 5.12 and 5.14). As discussed in section 8.1.2, the tracker model can be rescaled accurately, without basically any cost, since it can be stored entirely in the Fourier-domain. This opens the possibility of applying a brute-force search in the scale dimension. But to preserve the low computational cost, more sophisticated techniques must probably be applied.

A few issues are apparent from the attribute based results in section 5.5. One of these is fast target motions. A few techniques might be used to address this problem. A first simple thing to try is motion prediction, as discussed in section 2.4.2. This problem is also related to general failure detection and re-detection, which is currently not present in the algorithm. As most generic trackers, the RCS and RCSK have a very local view of the scene, meaning that it only cares about the target and the surrounding background. In contrast, the TLD tracker applies a whole framework for the purpose of re-detection in failure cases. Failures also often occur at occlusions, which is a major challenge. Although, the RCS is shown to be fairly robust to partial occlusion and short full occlusions compared to other trackers, long-term occlusions is a major problem. To address these sorts of issues, I think that it is necessary for the framework to take multiple hypothesis into account, and to evaluate them over time using different tracking models.

Although, failure detection and handling may be important for the generic tracking scenario, it may not be desired when a tracker acts as a part of larger system, which often is the case in real-world applications. One example is people tracking, which is discussed in the second part of this thesis. In that case there are distinct information, provided by for example a person detector, which can be used for failure detection and re-detection. The trackers that are proposed here might be especially suited as parts of such larger systems, thanks to their speed and simplicity. Another interesting property is that the tracker outputs a dense set of confidence scores, which can be fused with confidences from other system parts.
5.6 Conclusions and Future Work

Figure 5.11: Precision plots showing the attribute-based results of the state-of-the-art evaluation. The plots display the distance precision for the evaluated attributes fast motion, background clutter, motion blur, deformation, illumination variation and in-plane rotation. The trackers proposed in this thesis are shown in bold font. For clarity, only the top 10 performing methods are displayed in the plots. The value appearing in the title denotes the number of videos associated with the respective attribute. The average distance precision at 20 pixels is displayed in the legends.
Figure 5.12: Precision plots showing the attribute-based results of the state-of-the-art evaluation. The plots display the distance precision for the evaluated attributes low resolution, occlusion, in-plane rotation, out of view and scale variation. The trackers proposed in this thesis are shown in bold font. For clarity, only the top 10 performing methods are displayed in the plots. The value appearing in the title denotes the number of videos associated with the respective attribute. The average distance precision at 20 pixels is displayed in the legends.
Figure 5.13: Success plots showing the attribute-based results of the state-of-the-art evaluation. The plots display the overlap precision for the evaluated attributes fast motion, background clutter, motion blur, deformation, illumination variation and in-plane rotation. The trackers proposed in this thesis are shown in bold font. For clarity, only the top 10 performing methods are displayed in the plots. The value appearing in the title denotes the number of videos associated with the respective attribute. The area under the curve is displayed in the legends.
Figure 5.14: Success plots showing the attribute-based results of the state-of-the-art evaluation. The plots display the overlap precision for the evaluated attributes: low resolution, occlusion, in-plane rotation, out of view and scale variation. The trackers proposed in this thesis are shown in bold font. For clarity, only the top 10 performing methods are displayed in the plots. The value appearing in the title denotes the number of videos associated with the respective attribute. The area under the curve is displayed in the legends.
Part II

Category Object Tracking
In many applications, the problem is to automatically track all objects of a certain category. Examples of these kinds of applications include automated surveillance and safety systems in cars. These kinds of problems contain additional challenges compared to generic tracking. The system needs to automatically detect new objects and track them throughout the scene. However, compared to generic tracking there are additional a priori information that can be exploited to improve the robustness and accuracy. This information consists of the general appearance of the object class, e.g. humans. If multiple objects are tracked simultaneously, then the interactions between these can be used to further improve the tracking.

This second part of the thesis describes the automatic category tracker that was developed and implemented. This chapter gives an overview of the complete framework and describes the object and observation model, on which the framework is built. Section 6.1 gives an overview of the framework. Section 6.2 describes how individual objects are modelled using deformable part models and dynamics. Section 6.3 describes the measurement model and section 6.4 describes how the objects are tracked by applying the Rao-Blackwellized Particle Filter (see section A.4) to the model.

### 6.1 System Overview

Most category object trackers exploit the class specific and object specific appearance along with a motion model. The implemented tracker however, also incorporates tracking of specific object parts into the framework. One motivation for this is that such information can be of importance in e.g. action recognition. Consider for example tracking a human along with its hands and feet. Actions such as “walking” and “playing guitar” can then potentially be detected. However, main motivation for incorporating part tracking is
to improve the tracking of the object itself.

A short overview of the system is as follows. Images from the sequence are processed through two different system parts. The object detector produces dense score functions of the class specific object and part appearances. The appearance tracking produces score functions of the object and part specific appearances, based on the learnt appearance of the object and parts. These two sources of information are combined in a Bayesian filtering step, together with the dynamic model of the object. The results from the filtering are used to estimate the new location of the object in the image. The estimation is then used to update the learnt appearances of the object and parts. Tracked objects interact in two ways within the framework. Location estimates of all objects and parts are used to detect inter-object occlusions. This information is then used in the filtering step and in the appearance learning. The current location estimates are also used together with the object detections to find and initialize new objects in the scene and to remove false objects.

6.2 Object Model

This section presents the dynamic object model, which is formulated as a state space model.

6.2.1 Object Motion Model

A popular assumption in visual tracking is a constant velocity motion model [22, 2, 9]. If \( z \) is the object position in Cartesian coordinates (usually one, two or three dimensional), then the constant velocity model can be expressed as in (6.1).

\[
\begin{align*}
\dot{z}(t) &= v(t) \\
\dot{v}(t) &= w^v(t)
\end{align*}
\]  

The first equation in the model defines \( v \) as the velocity of the object. \( w^v \) is noise, which gives some flexibility in the model. Usually \( w^v \) is assumed to be white and Gaussian. A discretization of this model is given in (6.2), where \( T \) is the sample time. This discretization is done using Zero Order Hold [18], where the noise \( w^v \) is basically assumed to be constant during the sample interval.

\[
\begin{align*}
z_{t+1} &= z_t + T v_t + \frac{T^2}{2} w^v_t \\
v_{t+1} &= v_t + T w^v_t
\end{align*}
\]  

The scale (size) \( s_t \) of the object is modelled as constant with some process noise. Such a model is valid in cases when relative motion in the direction of the optical axis is small or if the tracked objects are far away.

\[
s_{t+1} = s_t + s_t w^s_t
\]  

It is physically more correct to let the noise \( w^s_t \) model the relative change in scale. This is the motivation for scaling the noise with \( s_t \). Only uniform scale is regarded in this work, i.e. \( s_t \) is scalar.
6.2 Object Model

6.2.2 Part Deformations and Motion

The model of the part locations should include the deformation costs of the parts and a motion model. The state space model in (6.4) has been constructed only using the deformation costs.

\[ z_{j,t+1}^j = s_t a_j^j + s_t u_t^j \]  
\[ u_t^j \sim \mathcal{N}(0, D^j) \]  

Here, \( z_{j,t}^j \) is the relative position of part \( j \) at time \( t \), \( a_j^j \) is the modelled expected position of the part and \( u_t^j \) is noise modelling the uncertainty. Although this model validly describes the deformations of a part as discussed in section 7.2.3, it discards the history of the part motion given the scale, i.e. \( p(z_{t+1}^j | z_t^j, s_t) = p(z_{t+1}^j | s_t) \). The history is included by using a Markov model instead, based on a constant relative position assumption. This assumption is highly valid for parts that are more or less rigidly connected to the object, e.g. the head of a human. However, the model can also be tuned for moving parts by increasing the process noise \( w_t^{zj} \). The part deformations are considered in the observation likelihood, as discussed in section 6.3.

\[ z_{t+1}^j = z_t^j + s_t w_t^{zj} \]  

This thesis will only deal with static models of part locations. Although there is a possibility of using dynamic models, it is not be investigated in this work.

6.2.3 The Complete Transition Model

The complete transition model that is used in the proposed object tracker is given in (6.6). The number of states are \( 2N + 5 \), where \( N \) is the number of object parts. In this chapter \( x_t = (z_t^0, s_t, v_t, z_t^1, \ldots, z_t^N)^T \) is used to denote the object state at time \( t \).

\[ z_{t+1}^0 = z_t^0 + T v_t + s_t w_t^{z0} \]  
\[ s_{t+1} = s_t + s_t w_t^s \]  
\[ v_{t+1} = v_t + T s_t w_t^v \]  
\[ z_{t+1}^j = z_t^j + s_t w_t^{zj}, \quad j \in \{1, \ldots, N\} \]  
\[ w_t^{zj} \sim \mathcal{N}(0, Q^{zj}) \]  
\[ w_t^s \sim \mathcal{N}(0, Q^s) \]  
\[ w_t^v \sim \mathcal{N}(0, Q^v) \]  

The velocity state noise \( w_t^v \) that should appear in (6.6a) according to the discretization (6.2) of the constant velocity model, is discarded. The reason is that the Rao-Blackwellized Particle Filter (RBPF) (see section A.4) does not handle this well if the velocity is taken as a linear state and the position as non-linear. This noise is replaced with \( w_t^{z0} \), which is assumed to be independent of \( w_t^v \). However, this modification also adds the freedom to tune the model between constant velocity and random walk. The drawback of this modification is insignificant, since the goal is to estimate the position of the object and not the real velocity.
The process noises defined in (6.6) are assumed to be mutually uncorrelated. $Q^z$, $Q^s$ and $Q^v$ are general covariance matrices. The fact that the noises are scaled with $s_t$ is motivated by the pinhole camera model. Note that the model is non-linear and non-Gaussian. It is however linear and Gaussian conditioned on the scale $s_t$. It is thus applicable for the RBPF.

### 6.3 The Measurement Model

This section describes the measurement model that is used in the tracker. Earlier works (like [9]) have used detector confidences as likelihoods in filtering or optimization frameworks. The measurement model that used here is inspired by [22], but extended with the deformation model for part positions. The likelihood contains two basic factors, as described in (6.7). $I_t$ is the image that is measured at time $t$. The factor $p(I_t | x_t)$ thus describes how well the given state explains the measured image. $M$ is the deformation model, which is static. The second factor $p(M | x_t)$ describes how well the state fits in to the deformation model.

$$p(I_t, M | x_t) = p(I_t | x_t)p(M | x_t) \tag{6.7}$$

Independence is assumed between the image measurements and the deformations given the object state.

#### 6.3.1 The Image Likelihood

The first factor in (6.7) combines information from the class specific appearance of the object and parts with the appearance of individual objects. Object detections (see chapter 7) are used as class specific appearance. $p(\theta^t_j | z, s)$ denotes the probability of detecting the object if $j = 0$ and part otherwise, at position $z$ and scale $s$ in image $I_t$. Section 7.2.2 describes how this factor is computed. The object appearance is modelled in a similar way. Assume that there is a detector trained on the appearance of the specific object. $p(\varphi^t_j | z)$ denotes the probability of detection analogously to the object detection. Section 8.1.2 describes how this factor is computed using the proposed K-MOSSE tracker that is described in the first part of this thesis. The image likelihood is modelled by (6.8).

$$p(I_t | x_t) = p(\theta^t_0 | z_0^t, s_t)p(\varphi^t_0 | z_0^t) \prod_{j=1}^{N} p(\theta^t_j | z_0^t + z_j^t, s_t)p(\varphi^t_j | z_0^t + z_j^t) \tag{6.8}$$

The model assumes independence between the object and part detections conditioned on the state. This is intuitively a valid approximation in an occlusion free environment. However, if occlusions are considered, the detections are likely to be correlated. The model also assumes independence between the detections from the class appearance and object appearance detector conditioned on the state. It can be argued that this is a reasonable approximation if the detectors uses different features. This is not valid at occlusions though.
6.3.2 The Model Likelihood

The second factor in (6.7) exploits the information in the known deformable parts model. This factor is modelled by (6.9).

\[ p(M|x_t) = \prod_{j=1}^{N} p(a_j^{|z_t^j}, s_t) = \prod_{j=1}^{N} \mathcal{N}(a_j^j; \frac{z_t^j}{s_t}, D_j) \]  

(6.9)

\(a_j^j\) is the known mean position of the part and \(D_j\) is a covariance matrix describing the deformations. Section 7.2.3 describes how these values are computed.

6.4 Applying the Rao-Blackwellized Particle Filter to the Model

The Rao-Blackwellized Particle Filter (RBPF) is a Bayesian filtering algorithm that exploits linear-Gaussian substructures in the state space model. It uses a particle filter to approximate the set of non-linear states \(x_n^t\) and Kalman filters for the remaining states \(x_l^t\), which are linear and Gaussian conditioned on \(x_n^t\). The algorithm is described in section A.4. The state space model in (6.6) is only linear and Gaussian conditioned on the scale \(s_t\). So, this state has to be included in \(x_n^t\). All states except the velocity states \(v_t\) appear as non-Gaussian in the measurement model because of (6.8). This implies that only \(v_t\) truly can be included in \(x_l^t\). However, since the main goal is to approximate the object position and scale, we will assume a Gaussian approximation of the part measurements. So, the partitioning of the states is done as in (6.10).

\[ x_n^t = \begin{pmatrix} z_t^0 \\ s_t \end{pmatrix}, \quad x_l^t = \begin{pmatrix} v_t \\ z_t^1 \\ \vdots \\ z_t^N \end{pmatrix} \]  

(6.10)

6.4.1 The Transition Model

The state transition model in (6.6) with the partitioning of states in (6.10) is a special case of the RBPF model in (A.14). The functions and matrices in the RBPF model is identified in (6.11).

\[ f_n^t(x_n^t) = x_n^t, \quad f_l^t(x_n^t) = 0_{2N+2 \times 1} \]  

(6.11a)

\[ A_n^t(x_n^t) = \begin{pmatrix} TI_{2 \times 2} & 0_{2 \times 2N} \\ 0_{1 \times 2} & 0_{1 \times 2N} \end{pmatrix}, \quad A_l^t(x_n^t) = I_{2N+2 \times 2N+2} \]  

(6.11b)

\[ B_n^t(x_n^t) = s_tI_{3 \times 3}, \quad B_l^t(x_n^t) = s_t \begin{pmatrix} TI_{2 \times 2} & 0_{2 \times 2N} \\ 0_{2N \times 2} & I_{2N \times 2N} \end{pmatrix} \]  

(6.11c)
The covariances are given by (6.12).

\[
Q^n_t = \begin{pmatrix} Q^0_t & 0_{2 \times 1} \\ 0_{1 \times 2} & Q^s_t \end{pmatrix}
\]  \hspace{1cm} (6.12a)

\[
Q^l_t = \begin{pmatrix} Q^0_t & 0_{2 \times 2} & \cdots & 0_{2 \times 2} \\ 0_{2 \times 2} & Q^{z_0} & \cdots & \vdots \\ \vdots & \vdots & \ddots & 0_{2 \times 2} \\ 0_{2 \times 2} & \cdots & 0_{2 \times 2} & Q^{z_N} \end{pmatrix}
\]  \hspace{1cm} (6.12b)

\[
Q^{ln}_t = 0_{3 \times 2N+2}
\]  \hspace{1cm} (6.12c)

### 6.4.2 The Measurement Update for the Non-Linear States

The measurement model described in section 6.3 is clearly more general than the one in (A.14). This means that the RBPF can not be applied directly. The restricting measurement model in the RBPF comes from the fact that it has to be linear and Gaussian in \(x^l_t\) conditioned on \(x^n_t\). However, the standard particle filter described in section A.3 can handle general measurement models. The measurement model in section 6.3 can thus be used to update the particle filtered states \(x^n_t\). A Gaussian approximation of the measurement model is then used to update the linear states \(x^l_t\). This results in a variant of the RBPF that uses different measurement models for the two measurement updates in algorithm A.4. The full model is used in the particle filter measurement update and the linear-Gaussian approximation is used in the Kalman filter measurement update.

The particle filter measurement update in algorithm A.4 requires the probability density \(p(y_t|X^n_t, Y_{t-1})\). \(y_t\) is the measurement, which in our case is \(y_t = \{I_t, M\}\) and \(Y_t = \{y_1, \ldots, y_t\}\) denotes all measurement up to time \(t\). \(X^n_t = \{x^n_1, \ldots, x^n_t\}\) denotes the trajectory of non-linear states up to time \(t\). This probability is given in (6.13). See section B.2 for the proof.

\[
p(y_t|X^n_t, Y_{t-1}) = g_t(s_t)L^0_t(z^0_t, s_t)\prod_{j=1}^N \int_{\mathbb{R}^2} h^j_t(z,s_t)L^j_t(z^0_t + z, s_t) \, dz
\]  \hspace{1cm} (6.13)

The functions that appear here are defined in (6.14).

\[
L^j_t(z, s) = p(\theta^j_t|z, s)p(\varphi^j_t|z)
\]  \hspace{1cm} (6.14a)

\[
h^j_t(z, s) = \mathcal{N}\left(z; \mu^j_t(s), H^j_t(s)\right)
\]  \hspace{1cm} (6.14b)

\[
g_t(s) = \prod_{j=1}^N \mathcal{N}\left(\frac{1}{s^2}z^j_{t|t-1}; a^j, \frac{1}{s^2}p^j_{z|t-1} + D^j\right)
\]  \hspace{1cm} (6.14c)
The mean and covariance of \( h^j_t \) are defined in (6.15).

\[
\begin{align*}
\mu^j_t(s) &= H^j_t(s) \left( (P^j_{t|t-1})^{-1} z^j_{t|t-1} + \frac{1}{s} (D^j)^{-1} a^j \right) \\
H^j_t(s) &= \left( (P^j_{t|t-1})^{-1} + (s^2 D^j)^{-1} \right)^{-1}
\end{align*}
\]

(6.15a)

(6.15b)

\( \hat{z}^j_{t|t-1} \) and \( P^j_{t|t-1} \) are the predicted mean and covariance of state \( z^j_t \) given the trajectory \( X^n_t \) for the non-linear states. I.e. we have \( p(z^j_t | X^n_t, Y_{t-1}) = \mathcal{N}(z^j_t; \hat{z}^j_{t|t-1}, P^j_{t|t-1}) \). This means that \( \hat{z}^j_{t|t-1} \) and \( P^j_{t|t-1} \) depend on \( z^0_t \) and \( s_t \), even though it is not denoted explicitly to simplify the notation. In the RBPF algorithm, these are the predictions bound to the particle \( X^n,t \), when evaluating the likelihood for this particle.

The derived particle weighting function in (6.13) has an interesting interpretation. \( h^j_t \) acts as a filter that smooths the likelihood function for part \( j \). This reflects the uncertainty that is present in the predicted part locations. \( h^j_t \) also shifts the part likelihood so that it is evaluated in \( z^0_t + \mu^j_t(s_t) \). \( \mu^j_t(s_t) \) should be seen as the predicted relative part location that has been corrected by the deformation model. Equation 6.15 is recognized as the fusion formula of the two estimates \( \hat{z}^j_{t|t-1} \) and \( s_t a^j \) with covariances \( P^j_{t|t-1} \) and \( s_t^2 D^j \) of the part location. \( H^j_t(s_t) \) is the covariance of the new part location estimate. \( h^j_t \) actually is the probability density \( p(z^j_t | X^n_t, Y_{t-1}, a^j) \), which supports this interpretation.

The function \( g^j_t(s_t) \) is essentially a deformation cost, where the uncertainty in the part locations has been added to the deformation covariance. This factor is the model likelihood given the non-linear states \( p(M | X^n_t, Y_{t-1}) \).

### 6.4.3 The Measurement Update for the Linear States

The RBPF requires a measurement model that is linear and Gaussian conditioned on the non-linear states to update the linear ones. This is achieved by a Gaussian approximation of (6.8). The first two factors in this equation (i.e. the object and appearance detection) do not contain any information about the part locations given \( z^0_t \) and \( s_t \), so they can be excluded here. Since the part measurements are mutually independent, each part can be considered individually. Consider the Gaussian approximation in (6.16) of the image likelihoods for each part. The function \( L^j_t \) defined in (6.14a) is the product between the class appearance likelihood and object specific appearance likelihood of part \( j \).

\[
L^j_t(z, s) \approx k^j_t(s) \mathcal{N}(z; y^j_t(s), R^j_t(s))
\]

(6.16)

\( y^j_t(s) \) and \( R^j_t(s) \) are the mean and covariance of the approximation, that depends on the scale \( s \). The scale factor \( k^j_t(s) \) is unimportant and can be disregarded since the scale is given when performing the measurement update for the linear states. The method for obtaining this approximation and the validity of it is discussed in section 8.2.3. The image likelihood for part \( j \) can be written as in (6.17).

\[
L^j_t(z^0_t + z^j_t, s_t) \sim \mathcal{N}(y^j_t(s_t); z^0_t + z^j_t, R^j_t(s_t))
\]

(6.17)

Here \( \sim \) denotes approximately equal up to a scale factor (for a constant \( s_t \)). The argument
and mean of the normal distribution have been switched. This can be done thanks to the symmetry of the Gaussian function. $y^j_t(s_t)$ can be regarded as a measurement. The right hand side in (6.17) is then the likelihood $p(y^j_t(s_t)|x_t)$ of this measurement. This likelihood is equivalent to the measurement equation in (6.18a).

The deformation model likelihood in (6.9) is already Gaussian and does not have to be approximated. Since the deformation likelihood for the parts are independent, (6.9) is equivalent to the measurement equation in (6.18b) for each part.

$$y^j_t(s_t) = z^j_t + e^j_t(s_t), \quad e^j_t(s_t)|s_t \sim \mathcal{N}(0, R^j_t(s_t)) \quad (6.18a)$$

$$a^j = \frac{1}{s_t}z^j_t + d^j_t, \quad d^j_t \sim \mathcal{N}(0, D^j) \quad (6.18b)$$

The conditionally linear-Gaussian measurement model thus contain two measurements for each part location. The iterated measurement update in algorithm A.2 can easily be applied in the RBPF on this model, since all position measurements are mutually uncorrelated.
As described in chapter 1, category object tracking is the problem of tracking objects of a specific class, e.g. humans. This tracking problem contains additional a priori information that can be exploited. If the system is supposed to work completely automatically, it additionally has to detect objects of this category. Object detection is a well studied area in computer vision. The first section in this chapter briefly describes the object detector [15], which is used in my proposed object tracker. The remaining sections discuss how this detector is used in my framework.

7.1 Object Detection with Discriminatively Trained Part Based Models

The object detector [15] by Felzenszwalb et. al. has proved to be very successful, and it still achieves state of the art performance. It uses a model of deformable parts to describe the object. This section contains a brief description of this object detection framework. For more details, see [15]. This object detector will be referred to as the deformable part model (DPM) detector.

7.1.1 Histogram of Oriented Gradients

Dalal and Triggs introduced the histogram of oriented gradients (HOG) features in [12] and applied it to human detection in static images. The HOG feature map is created by first computing histograms of gradient orientations in a dense image grid of cells, which are typically $8 \times 8$ pixels. The histograms are constructed using soft assignment in neighbouring cells. This is then followed by a normalization step, where the histograms are normalized using the gradient energy from different neighbouring cells. The original HOG results in a 36-dimensional feature vector for each cell. The HOG features are
Object Detection typically computed at many different scales. Figure 7.1 visualizes the HOG feature map for an image region.

Dalal and Triggs trained a linear support vector machine (SVM) [7] on thousands of false and positive examples of humans, using HOG features. Object detection score at all locations and scales of an image can be computed using a sliding window search, i.e. by correlating the feature map of the image with the SVM-weights.

### 7.1.2 Detection with Deformable Part Models

The DPM detector [15] extends the work of Dalal and Triggs to use deformable part models. The detector uses a modification of HOG-features. But instead of just training a single template of SVM weights for the whole object, separate part templates are also trained for a set of object parts (e.g. head or feet for a human). Additionally, a deformable part model of the object is trained. This model includes the anchor position \( v_j = (v^x_j, v^y_j) \) and the quadratic deformation cost coefficients \( d_j = (d^1_j, d^2_j, d^3_j, d^4_j)^T \) for each part \( j \in \{1, \ldots, N\} \). Let \( G_j \) be the trained SVM weights (or filters) for each part, where \( j = 0 \) indicates the root filter that is trained on the whole object. Let \( H \) denote the HOG feature map for an image. The classification score from a filter at position \( p = (x, y) \) and scale \( s \) is calculated as in (7.1).\(^1\) Note that the part scores for an object detection at scale \( s \) are computed at half the scale, i.e. double the resolution.

\[
\zeta_j(p, s) = \begin{cases} 
G_0 \star H(p, s) & , j = 0 \\
G_j \star H(p, \frac{s}{2}) & , j \in \{1, \ldots, N\} 
\end{cases} \quad (7.1)
\]

The vector defined in (7.2) contains the linear and quadratic absolute displacements of a part from its anchor position.

\[
\Delta_i(p_0, p_j, s) = \begin{pmatrix} 
\frac{(x_j - x_0)}{s} - v^x_j \\
\frac{(y_j - y_0)}{s} - v^y_j \\
\frac{(x_j - x_0)}{s} - v^x_j \right)^2 \\
\frac{(y_j - y_0)}{s} - v^y_j \right)^2
\end{pmatrix} \quad (7.2)
\]

The deformation cost for part \( j \) is computed as \( d_j^T \Delta_j(p_0, p_j, s) \). Given a trained model and SVM weights, the DPM detector calculates the final object detection score as in (7.3). Here \( b \) is just a constant.

\[
\zeta(p_0, s) = \max_{p_1, \ldots, p_N} \left( \sum_{j=0}^{N} \zeta_j(p_j, s) - \sum_{j=1}^{N} d_j^T \Delta_i(p_0, p_j, s) + b \right) \quad (7.3)
\]

\(^1\)Correlation (\( \star \)) is generalized to vector valued functions by correlating each feature layer individually and summing the results at each position.
7.1 Object Detection with Discriminatively Trained Part Based Models

(a) Frame 50 from the Town Centre sequence.

(b) Visualization of HOG features from a part of the image in figure 7.1a containing the two persons dressed in black near the center of the image. The HOG features has been calculated at scale $s = 2^{7/10} \approx 1.62$, i.e. the image is down-sampled with a factor $1/s$ first.

**Figure 7.1:** Visualization of the HOG features of the frame in figure 7.1a. In figure 7.1b, the magnitude of each orientation bin in a cell is visualized by the intensity of the line with the corresponding orientation.
7 Object Detection

(a) Root filter SVM weights.  (b) Part filter SVM weights.  (c) Part placements and deformation costs.

Figure 7.2: Visualization of the INRIA-person model, which is trained on the INRIA [12] dataset. The model contains two components that are reflections of each other along the vertical axis. The SVM weight for each orientation bin is visualized as in figure 7.2a and 7.1b. The magnitude of the deformation cost functions are visualized in figure 7.2c. The model contains eight parts.

7.1.3 Training the Detector

The classifier in (7.3) can be formulated as in (7.4).

\[
    f_w(x) = \max_{z \in Z(x)} w^T \Phi(x, z) \tag{7.4}
\]

Here, \(w\) is the vector of classifier weights, that in this case includes the filter weights \(G_j\) and the deformation coefficients \(d_j\). \(x\) is the example to be classified and \(Z(x)\) is the set of possible latent values for \(x\). The part positions are latent in this case. \(\Phi(x, z)\) is the extracted feature vector for the particular example \(x\) and part configuration \(z\). In [15] this classifier is trained using supervised learning with latent SVM. This is done by minimizing the objective function in (7.5). \(x_i\) denotes the examples and \(y_i \in \{-1, 1\}\) are the corresponding labels.

\[
    L_D(w) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i f_w(x_i)) \tag{7.5}
\]

Here, \(C\) is a regularization parameter. The optimization problem (7.5) is non-convex. However, a strong local optima can be found by exploiting the semi-convexity of this function. The optimization iterates between finding the optimal part placements for the positive examples given \(w\) and optimizing over \(w\) given these part placements. The second step can be shown to be a convex optimization problem. For further details on this training procedure, see [15]. My proposed human tracker uses the human detector that is pretrained on the INRIA [12] dataset. The trained model is visualized in figure 7.2.
7.2 Object Detection in Tracking

One of the goals in this part of the thesis is to fuse the information from the object and part detections with appearance tracking in a probabilistic framework. This section discusses how object detection can be used in a tracking framework.

7.2.1 Ways of Exploiting Object Detections in Tracking

The popular way of using object detections in tracking is to select a sparse set of detections as observations of the object state. This can then be used in for example a Bayesian filtering framework. Though many reduce the problem to data association of a set of sparse detections [2, 39]. Such a set of detections can be obtained by simply thresholding the dense detection scores. However, the thresholding discards large amounts of information returned by the object detector.

An obvious possibility is to use the whole detection score as a confidence map or likelihood of an object being present at a specific location. Breitenstein et. al. [9] pointed out that the detector score from the original HOG person detector [12] is too poor to use as a confidence map in most cases. They countered this fact by mostly relying on thresholded detections in their filtering framework and only trusting the detector confidence in certain specific cases. The DPM detector returns much more distinct detection confidences. Izzadinia et. al. [22] successfully exploited these confidences to track both humans and their parts in a non-causal framework based on graph optimization.

My work exploits the dense detection scores for both the object and the separate parts. The scores obtained from the root-filter are not used explicitly, but they of course contribute to the final object detection scores. The confidences returned by the full detector are of much higher quality than the confidences computed from the part filters. As the original HOG detector, the part confidences are just the output from a linear classifier. These outputs also suffer from the fact that the appearance of for example a shoulder is not that discriminative. However, these flaws are countered by jointly tracking the parts and the object itself.

7.2.2 Converting Detection Scores to Likelihoods

The detection scores are computed at a cell level resolution. The cells are not overlapping in the standard version of the DPM detector. This means that the resolution of the detection score at scale $s$ is $\frac{1}{8s}$ times the resolution of the original image for the object detections and twice for the part detections. However, it is more practical to use pixel-dense scores. These are obtained by interpolating the detection scores with splines. The effect is illustrated in figure 7.3.

To be able to use the detection scores in a probabilistic framework, the scores are transformed to values that can be interpreted at probabilities. In chapter 6 the detection scores are incorporated in the tracker as likelihoods $p(\theta_j | z, s)$. $\theta_j$ is a binary stochastic variable that indicates the detection of the object ($j = 0$) or part ($j > 0$). $p(\theta_j | z, s)$ should then be interpreted as how likely it is to detect the object or part at position $z$ and scale $s$. In [22] detection scores are transformed to probabilities using sigmoid functions. A Sigmoid function is some kind of smooth step function. There exist many proposed variants
of such functions. The type used here is given in (7.6).

\[
\psi(t) = \frac{1}{1 + e^{-\alpha(t-\beta)}}
\]

(7.6)

The parameters \( \alpha > 0 \) and \( \beta \in \mathbb{R} \) needs to be tuned for each object and part detector individually. Let \( \hat{\hat{\zeta}}_j(z, s) \) denote the interpolated detection score at position \( x \) and scale \( s \). These scores are simply mapped through a sigmoid function using (7.7), to obtain the corresponding likelihoods. The effect of this step is illustrated in figure 7.3.

\[
p(\theta^j|z, s) = \psi_j\left(\hat{\hat{\zeta}}_j(z, s)\right) = \frac{1}{1 + \exp\left(-\alpha_j(\hat{\hat{\zeta}}_j(z, s) - \beta_j)\right)}
\]

(7.7)

The sigmoid parameters are tuned based on gathered statistics of the confidence values. The cumulative distribution functions \( F_j \) of the detection scores over an image are defined in (7.8). Here \( x = (z, s) \) and \( X \) is the set of all positions and scales in the image. In practice many images can be used. \( F_j \) are approximated by computing histograms of detection scores over a set of images and computing their cumulative sums.

\[
F_j(\xi) = \begin{cases} 
\frac{1}{|X|} \left| \{x \in X : \zeta(x) \leq \xi\} \right|, & j = 0 \\
\frac{1}{|X|} \left| \{x \in X : \zeta_j(x) \leq \xi\} \right|, & j \in \{1, \ldots, N\}
\end{cases}
\]

(7.8)

Using the definition of precision \( P_j(\xi) \) and recall \( R_j(\xi) \) at a detection threshold \( \xi \), the equality in (7.9) can be derived. Here \( T_j \) is the relative number of locations \( x \) in the image that contain the specified object.

\[
F_j(\xi) = 1 - T_j \frac{R_j(\xi)}{P_j(\xi)}
\]

(7.9)

One way of tuning the parameters in (7.7) would be to specify the desired recall rate at the two thresholds \( \lambda_1 \) and \( \lambda_2 \), where \( x \) is considered a detection if and only if \( p(\theta^j|x) \geq \lambda_k \). The corresponding detection score thresholds \( \xi_j^{(1)} \) and \( \xi_j^{(2)} \) can then be obtained if the recall rates are known. This gives the equation system (7.10).

\[
\begin{align*}
\psi_j\left(\xi_j^{(1)}\right) &= \lambda_1 \\
\psi_j\left(\xi_j^{(2)}\right) &= \lambda_2
\end{align*}
\]

(7.10)

The solution is given in (7.11).

\[
\begin{align*}
\alpha_j &= \frac{1}{\xi_j^{(1)} - \xi_j^{(2)}} \left( \ln\left(\frac{1}{\lambda_2} - 1\right) - \ln\left(\frac{1}{\lambda_1} - 1\right) \right) \\
\beta_j &= \xi_j^{(1)} + \frac{1}{\alpha_j} \ln\left(\frac{1}{\lambda_1} - 1\right)
\end{align*}
\]

(7.11a, 7.11b)

This approach is problematic since a labelled dataset is needed to estimate the recall \( R_j(\xi) \). A simple method that does not require labelled data was used in this work. For the object
(a) The DPM human detector output at scale \( s = 2^{7} \approx 1.62 \) for the image displayed in figure 7.1a.

(b) Spline interpolation of the detector scores in figure 7.3a to obtain pixel-dense scores.

(c) The final likelihood \( p(\theta|z, s = 2^{7}) \) computed from the detector scores in figure 7.3b, using (7.7) with the computed values for \( \alpha_0 \) and \( \beta_0 \).

Figure 7.3: Visualization of the human detector output (figure 7.3a) from the image in figure 7.1a. Figure 7.3b and 7.3c show how these detections are transformed to pixel-dense likelihoods. Note that the high values corresponds to humans in figure 7.1a with similar height in pixels. Also note how apparent the peaks are in figure 7.3c compared to figure 7.3b.
detector, the desired fraction $\Lambda_k$ of detections in the images at threshold $\lambda_k$ was tuned by visual inspection of the resulting $p(\theta^0|x)$ and by performance evaluation. This gives $\xi_0^{(k)}$ from $F_0(\xi_0^{(k)}) = 1 - \Lambda_k$. A valid approximation is that $T_j = T$ if occlusions in the scene are rare. By comparing the detection scores of the part detectors with the object detector it is possible to get a very coarse approximation of the precision of the part detectors relative the object detector at a “high enough” recall rate. If the precision of the part detectors are assumed to be a factor $c_j$ worse at the desired recall rates, then $\xi_j^{(k)}$ can be found from $F_j(\xi_j^{(k)}) = 1 - \Lambda_k/c_j$ for $j > 0$. The sigmoid parameters are then computed using (7.11).

This method provides a more intuitive way of tuning the sigmoid parameters by letting the user select $\lambda_k$ and then $\Lambda_k$ based on that. These parameters have intuitive meanings, while it can be difficult to select $\alpha_j$ and $\beta_j$ directly.

In the proposed human tracker, the sigmoid parameters were tuned using the detection scores from the first 20 frames in the Town Centre sequence [5]. The thresholds were were chosen as $\lambda_1 = 0.5$ and $\lambda_2 = 0.1$. The desired fraction of detections for the object detector was set to $\Lambda_1 = 10^{-4}$ and $\Lambda_2 = 10^{-2}$. The precision factor was set to the same for all part detectors $c_j = 0.1$. Not much effort was invested in to tune these values, since the proposed tracker proved to be quite insensitive to variations in the mentioned parameters.

7.2.3 Converting Deformation Costs to Probabilities

The deformable parts model was fitted in to a probabilistic framework by converting the deformation costs in the DPM detector to probabilities. To simplify later steps in the modelling, Gaussian probabilities are used. The deformation probabilities are parametrized as in (7.12). As in chapter 6, the part number is denoted with index $j$, $z^j$ is the position of the part relative to the object location and $s$ is the scale of the object. $a^j$ is the mean position of the part relative to the object and $D^j$ is a $2 \times 2$ covariance matrix.

$$p(z^j|s) = \mathcal{N}(z^j; sa^j, s^2 D^j)$$ (7.12)

The straight forward way to estimate $a^j$ and $D^j$ is to do a Maximum Likelihood estimation given a set of images with known object and part locations and scales. However, a more ad hoc solution was done because of the absence of such a dataset. The proposed solution exploits the pre-trained anchor positions $v_j$ and deformation coefficients $d_j$ from the DPM detector. The mean position $a^j$ is set to the $(x, y)^T$ that minimizes the deformation cost function in (7.13). $(x, y)^T$ is the relative part position normalized with the scale of the object.

$$f_j(x, y) = d_1^j(x - v_x^j) + d_2^j(y - v_y^j) + d_3^j(x - v_x^j)^2 + d_4^j(y - v_y^j)^2$$ (7.13)

The solution, which is given in (7.14) is easily obtained by setting the gradient to zero $\nabla f_j = 0$.

$$a^j = \left( \begin{array}{c} v_x^j - \frac{d_1^j}{2d_3^j} \\ v_y^j - \frac{d_2^j}{2d_4^j} \end{array} \right)$$ (7.14)
The covariance matrix is set as in (7.15).

\[
D^j = \begin{pmatrix} 1/d_3^j & 0 \\ 0 & 1/d_4^j \end{pmatrix}
\]

(7.15)

The motivation for this is that \( p(z^1, \ldots, z^n | s) \) then can be written as in (7.16). The model assumes independence between the part locations given the scale. Note the similarities of the argument of the exponential function and the deformation cost in (7.3). The only difference is that the anchor position has been corrected with the linear deformation terms instead of including them as extra terms.

\[
p(z^1, \ldots, z^n | s) = \prod_{j=1}^{n} p(z^j | s) = \\
= \frac{1}{(2\pi s^2)^n} \sqrt{\prod_{j=1}^{n} d_3^j d_4^j} \exp \left( -\frac{1}{2} \sum_{j=1}^{n} d_3^j \left( \frac{z_j^x}{s} - a_j^x \right)^2 + d_4^j \left( \frac{z_j^y}{s} - a_j^y \right)^2 \right)
\]

(7.16)
This chapter presents and discusses the most important details of the constructed framework for human tracking in surveillance scenes. Section 8.1 discusses the computation of the appearance likelihood and how the proposed RCSK tracker is incorporated into the framework. Section 8.2 contains the details of how the RBPF is used in the filtering step. Section 8.3 presents some details related to occlusion handling and how objects are added and removed.

8.1 The Appearance Likelihood

The image measurement model in (6.8) uses two kinds of likelihoods. \( p(\varphi_t | z, s) \) models the class specific appearance and is discussed in section 7.2.2. The second factor \( p(\varphi_t | z) \) models the object specific appearance and does not take the object class into account. This section describes how this likelihood factor is computed using the generic tracking methods described in the first part of this thesis.

8.1.1 Motivation

It can be argued that the object specific appearance is irrelevant information if the application does not require identities of the tracked objects. However, this is not true in practice since the object detections are never perfect. The motivation for including appearance based tracking in the model can be summarized in these two points.

- To help keeping the object identities.
- To counter the imperfections of the object detections.

The first point is clear. If the individual appearance of the objects are modelled, then association between frames is simpler. The second point comes from the fact that object
detections in still images does not regard the temporal dimension. In section 7.2.2 it was noted that the object detections of human parts are of poor quality, i.e. with a low precision rate. A good generic tracker does not suffer from this problem, since it regards the learnt appearance of the object in the previous frames.

Generic trackers are often very accurate in tracking an object from frame to frame. However, common problems among generic trackers are long term drift and that failure often results in loosing track of its target completely. But object detections in still images does not suffer from these flaws. The motivation for combining appearance tracking and object detections is thus that it has the potential to give accurate frame to frame tracking while being robust to drift and failure.

For the application in this thesis, a generic tracker with the following properties was desired.

1. Simple.
2. Fast.
3. Exploits color information.
4. Outputs pixel-dense confidences.

Some generic trackers, like the Tracking Learning Detection framework [24] uses complex appearance models and methods for failure detection and redetection of the target. Such properties are not needed in this application, since it is handled by the implemented filtering framework and the fusion with object detections. This is the motivation behind the first point in the list.

Although the proposed framework does not aim for real time in its current state, it still has to be sufficiently fast to make testing and evaluation practical. The object detections can be precomputed, but this is not the case for the generic tracker output since it uses information from previous frames. In the surveillance sequence, which the framework is evaluated on, as many as 30 humans can appear in the scene at the same time. Since the human is tracked along with 8 defined parts, almost 300 image regions are tracked simultaneously. The second point in the list thus becomes clear.

The used object detector that is described in section 7.1 only uses edge information. It is thus intuitive to use color as appearance information to complement this. In the case of human tracking, the color of cloths is a very discriminative feature to separate individuals. This explains the need of the third property in the list. The last property comes as a result of the image measurement model in (6.8). It assumes that the appearance likelihood can be evaluated in a sufficiently dense set of locations.

The RCSK tracker, discussed in the first part of the thesis, turns out to have all these properties. This tracker is used to compute the appearance likelihoods as described in the next section.

8.1.2 Integration of the RCSK Tracker

A variant of the proposed RCSK tracker in algorithm 2.1 is used to compute the appearance likelihoods \( p(\varphi_t^I | z) \). In each frame where the object or part is not occluded, the
tracker is updated using the image patch around the estimated location ($\hat{z}_{0t}^t$ for the object and $\hat{z}_{jt}^t + \hat{z}_{jt}^t$ for part $j$). The size of the patch is determined by the estimated scale $\hat{s}_{t|t}$ of the object.

When processing a new frame, the tracking scores (correlation output) for each unoccluded object and part is computed in an area around their previously estimated locations in the image. These score values can equivalently be seen as detections scores from an object detector trained on the specific object or part appearance. Analogously to the detection scores from the DPM detector in chapter 7, these score values are mapped through a sigmoid function to obtain a probability interpretation. The same kind of sigmoid function (7.6) was used in this case. The parameters were tuned to $\alpha = 6$ and $\beta = 0.5$.

To get a more accurate appearance tracking, the spatial size of the RCSK appearance model (i.e. $A_{tN}^t$, $A_{tD}^t$ and $\hat{s}_t^t$ in (2.21)) needs to be set according to the current estimated object scale $\hat{s}_{t|t}$. The transformed model coefficients $A_{tN}^t$ and $A_{tD}^t$ are resized by either padding with zeros in the highest frequencies or removing the highest frequencies to get the appropriate size. This corresponds to an interpolation of the coefficients in the spatial domain. The learnt appearance is resized in a similar way, by either zero padding or removing the edges. This simple scheme of resizing the trackers proved to be very robust, even in quite extreme cases.

### 8.2 Rao-Blackwellized Particle Filtering

This section describes the practical details of how the RBPF is applied to the model described in chapter 6. The time update and the resampling step in the RBPF is performed as described in algorithm A.4. Some approximations are needed in the measurement updates though to make the RBPF applicable to the model proposed in chapter 6. The standard prior proposal distribution $q(x_{t+1}|X_t, Y_{t+1}) = p(x_{t+1}|X_t^n, Y_t^n)$ was chosen.

#### 8.2.1 Parameters and Initialization

When a new object is added, its states are initialized using the information from the corresponding detection. The initial object position, scale and velocity for each particle are drawn from the prior Gaussian distribution. The mean position and scale are set as the detection values. The mean velocity is set to zero. The initial part positions are for all particles set to the ones obtained from the initial detection.

All prior and process covariances are parametrized and tuned coarsely by hand.

#### 8.2.2 The Particle Filter Measurement Update

Theoretically (6.13) should be used to update the particle weights in the RBPF. In practice, the likelihood functions $L_j^t$ are defined over a discrete domain, i.e. at each pixel location and at some discrete scales. The integration must thus be approximated by a sum. The Gaussian filter $h_j^t$ needs to be sampled at a pixel dense grid. However, since the covariance $H_j^t$ of $h_j^t$ depends on the object scale $s$ and the predicted part location covariance $P_{z_j^{t-1}}$, it is also depends on the particle number. This means that the correlation in (6.13) should be computed with a different filter for each particle.
To reduce the computational cost of the measurement update, the covariance $H^j_t(s)$ defined in (6.15b) is in each time step approximated by a diagonal covariance matrix $\tilde{H}^j_t$ that is independent of the particle number. The likelihood (6.13) can be approximately calculated using (8.1).

$$p(y_t | X^n_t, Y_{t-1}) = g_t(s_t) L^0_t(z^0_t, s_t) \prod_{j=1}^{N} (\tilde{h}^j_t * L^j_t(\cdot, s_t)) \left( z^0_t + \mu^j_t(s_t) \right)$$  \hspace{1cm} (8.1)

Here, the approximative diffusion filter $\tilde{h}^j_t$ defined in (8.2) is independent of the particle number.

$$\tilde{h}^j_t(z) = \mathcal{N}(z; 0, \tilde{H}^j_t)$$  \hspace{1cm} (8.2)

Notice that the effect of $\mu^j_t(s_t)$ (defined in (6.15a)) has been moved to the computation of the point where the correlation result is evaluated. This is obtained by a simple change of variables in the integral in (6.13). The correlation output in (8.1) is hence evaluated in the deformation-corrected predicted part location $z^0_t + \mu^j_t(s_t)$.

The diagonal elements of $\tilde{H}^j_t$ are computed as mean of $H^j_t$ over all particles. The non-diagonal elements are set to zero. This approximation enables $\tilde{h}^j_t$ to be separated into two one-dimensional Gaussian filters, which further significantly reduces the computational cost. The approximation of $H^j_t$ is motivated by the fact that the exact amount of smoothing generated by the filter is of much lesser importance than the mean value $\mu^j_t$, which decides where the correlation result is evaluated. Further, the precision in the estimation of $P_{z^j_t|t-1}$ and thereby also $H^j_t$ is questionable. As described in section 7.2.3, the deformation covariance $D^j$ is set to a diagonal matrix. $P_{z^j_t|t-1}$ is almost diagonal in most cases since the process noise of the part locations are diagonal. The diagonal approximation of $H^j_t$ is thus also motivated.

In the evaluation of (6.13), the object and part positions for each particle are rounded to the nearest pixel. The likelihood is linearly interpolated in the scale dimension.

In evaluations it was observed that background clutter is a major problem for some human part detectors. This clutter occasionally deteriorated the total likelihood function for a part, thereby opposing the assumption of a major mode in the likelihood, which is necessary for the Gaussian approximation to be valid. This model error resulted in a noticeable reduction in tracking performance of the object itself when severe background clutter was apparent. This was typically the case for the lower body parts. The reason is that the feet and legs move relative to the object itself, which makes them harder to track due to appearance changes and self occlusions. Additionally the lower body parts are much more commonly occluded by background structures and have a less discriminative appearance. A significant improvement in the performance was noticed if only a subset of parts was used in the measurement update for the non-linear states (6.13). Only the upper body parts are thus used in (6.13) for the human tracking application.
The Kalman Filter Measurement Update

In section 6.4.3 a Gaussian approximation of the likelihood functions for the object parts is assumed. This is necessary to be able to apply the RBPF to the proposed tracking model. The mean $y^j_t(s)$ of the Gaussian approximation is selected as the maximum value of the likelihood function $L^j_t(z,s)$ in the region. The covariance $R^j_t(s)$ is set to the covariance of the likelihood function after normalizing it so that it sums to one. $y^j_t(s)$ and $R^j_t(s)$ are calculated for each scale level. In the Kalman filter measurement update of the RBPF, each particle can be updated using the measurement that corresponds to the scale that is closest to the scale estimation given by that particle. In practice, it turned out to be better to update all particles with the measurement that corresponds to the estimated scale $\hat{s}_t$ of the object.

Gaussian approximations reasonable if the actual probability is uni-modal. Since the labelling function of the RCSK tracker is Gaussian, this generates a roughly Gaussian shaped score function in most “easy” cases. In this case “easy” means sufficiently small translations and appearance changes between frames. This most often the case in surveillance scenes. The part detection scores are ideally uni-modal in a neighbourhood of the expected part location. If other objects are close, then several modes may exist. Though, this is in most cases handled by the fusion with the appearance likelihood, which should only have one mode at the target. The problem is that the part detectors for most human parts suffer from a very low precision rate due to background clutter. The trained SVM-detectors used for human parts by the DPM human detector, are small and detect at a quite low resolution. They often give high classification scores to certain basic shapes or patterns that can be common in background structures. For example, a spot or line on the ground can look suspiciously similar to a foot. When the tracking is affected by this kind of clutter, the Gaussian assumption is often violated, which might cause the tracking of a specific part to fail. However, since many parts are tracked jointly, the tracking of the object itself is robust to if a minority of the part trackers fail.

The iterated measurement update in algorithm A.2 was applied to the Kalman filter measurement update in the RBPF, since all position measurements are uncorrelated. This avoids inverting large matrices (up to $4N \times 4N$) in the computation of the Kalman gain in (A.16d) for each particle. Instead, the iterated measurement update only requires several $2 \times 2$ and $1 \times 1$ matrices to be inverted for each particle, which is considerably faster. The iterated measurement update was also applied to the Kalman filter time update, since it contains an “extra” measurement update with uncorrelated measurements, when using the model in (6.6) (see section A.4).

Estimation

Like the usual particle filter, the RBPF only returns an approximation of the posterior distribution and not a point estimate of the state. In chapter 1 it is stated that visual tracking is the problem of estimating the trajectory of the object in the image, i.e. the position or state of the object in each frame. It is thus necessary to find a point estimate of the state. This can be done in many ways.

The two most common methods for obtaining a point estimate were tried: minimum variance (MV) and maximum a posteriori (MAP). These are described in section A.4.2.
The performance difference proved to be insignificant. The MV estimate however gives smoother trajectories, which are visually more appealing. This method was therefore used in the final version.

### 8.3 Further Details

This section discusses the additional system parts that were needed to build a complete automatic human tracking framework.

#### 8.3.1 Adding and Removing Objects

An automatic object tracker requires automatic ways of detecting new objects and removing false tracked objects. Although these are important and non-trivial tasks, they were not in the focus of this thesis, so simple methods were employed. However, these methods proved to be quite effective.

To find new objects in the scene, all detections over a certain threshold are gathered in each frame. This set of detections is then reduced in a number of steps. Firstly, detections too close to the image borders are removed. This is then followed by two steps of non-maximum suppression. If the bounding boxes of two new detections are overlapping more than a certain threshold, then the one with the smallest score is removed. The second step compares the overlap between the remaining boxes and the bounding boxes of the existing objects in the scene. A new detection is removed if has a too large overlap with any of the existing objects. The remaining detections after this step are considered to be newly detected objects. These are initialized with the position, scale and part locations given by the detection. The overlap measure used in these cases is given in (8.3). \( B_1 \) and \( B_2 \) are two bounding boxes.

\[
\text{overlap}(B_1, B_2) = \max \left( \frac{\text{area}(B_1 \cap B_2)}{\text{area}(B_1)}, \frac{\text{area}(B_1 \cap B_2)}{\text{area}(B_2)} \right)
\]  

An existing tracked object is removed if any of the following requirements are fulfilled.

1. The object is too far outside the image.
2. The object is too large or too small, so that it is outside the range of scales used by the detector.
3. The object has been fully occluded for too many frames.
4. The object is not significantly occluded but too dissimilar to the object class, which is indicated by a too low detection score over the last few frames.

#### 8.3.2 Occlusion Detection and Handling

Advanced techniques for occlusion detection were not investigated in this thesis work. Rather, a very simple but effective method for detecting inter object occlusions was adopted from [47]. If two tracked objects overlap, then the object with the bounding box that has the highest lower y-coordinate is considered to be the occluded one. This results from the assumption that objects that are further away from the camera are higher up
in the image. This is true for example if the objects are moving on a ground plane which is tilted towards the camera, but not necessarily planar. The assumption holds for most surveillance videos. The parts of the occluded object that have a large enough overlap with the occluding object are considered to be occluded. Parts that are far enough outside the image borders are also considered to be occluded. No system is used for detecting occlusions from scene objects or other non-tracked objects. Figure 8.1 visualizes the effect of the inter object occlusion detection.

Occluded parts are not used in the measurement updates of the RBPF. They are not tracked and their appearance models are not updated. If enough parts of the object are occluded, then the object itself is considered to be too occluded to utilize the appearance tracking and detection scores for the whole object in the measurement update of the RBPF. In this case, the likelihoods $p(\theta^0_t|z, s)$ and $p(\varphi^0_t|z)$ are set to uniform distributions. Further, the appearance of the whole object is not updated. For the application to human tracking, it was chosen so that the object itself is considered occluded if any of the upper body parts

Figure 8.1: Three scenarios with significant inter object occlusions. Only the human part boxes that are considered to be non-occluded by the framework are displayed.
are occluded.

The time update in the RBPF is not affected by the occlusions. So, occluded part locations are predicted by the model and covariance is added to the predictions, which reflects the added uncertainty by only predicting. If all object parts are occluded then no measurements of the object location are available. The time update in the RBPF will then give predictions of the object location using the constant velocity motion model described in section 6.2.3.
This chapter presents the results of the constructed category tracker, when applied to human tracking in surveillance scenes. Section 9.1 presents the qualitative results. Section 9.2 discusses the method, results and potential future work. The final conclusions are summarized in section 9.3.

9.1 Results

The framework was implemented in Matlab and tested on a desktop computer with an Intel Xenon 2 core 2.66 GHz CPU with 16 GB RAM. The number of particles in the RBPF was set to 1000. Very little time was spent on tuning the large number of parameters. The Town Centre sequence provided by [5] was used for testing and evaluations. This sequence consists of 7501 frames in 1920 × 1080 resolution at 25 fps. The scene is a busy town centre street. Figure 9.1 visualizes the bounding boxes of the tracked objects and the estimated object trajectories for every 100:th frame in the first 1000 frames. Figure 9.2 displays the part-trajectories of a few selected objects. In the latter two images, the trajectories are disrupted by inter object occlusions. In the last image, the person is successfully tracked even though only the head of the person is visible for a long period of time.

Figure 9.1 and 9.2 show reasonable object and part trajectories. The system is able to track most humans through the entire scene. But there are some disrupted tracks, mostly due to imperfections in the object detector. It can also be seen that the DPM detector, while powerful, also gives some obvious false detections. Two of these are persistent, as they are triggered by background structures. One of these is the mannequin in the shop-window to the left and the other one occurs in the lower right corner.
Figure 9.1: Tracking results for frame number 100, 200, ..., 1000 in the Town Centre sequence. The center location trajectories are displayed for humans that are tracked in the specific frame.
Figure 9.2: Estimated human part trajectories of some selected objects. Note that many trajectories in the two latter cases are disrupted by inter object occlusions.
9.2 Discussion and Future Work

As discussed in section 1.2.2, most research in category tracking is focused on pure data association of detections and global optimization with non-causal assumptions. The motivation behind the presented work is to incorporate object detections in more sophisticated ways while avoiding the non-causal assumption. However, real-time frame rates are necessary for online applications. In the current Matlab implementation, the computational time for the system with the object detector excluded is between 1 and 3 seconds per frame, depending on the number of present targets. But since particle filters and FFTs are parallelizable, real-time frame rates can potentially be obtained in a GPU (graphics processing unit) implementation.

The object detection scores were precomputed using the Matlab/mex code provided by [15]. This took approximately four days for the entire Town Centre sequence (7501 frames and using 58 different scales). However, recent works [33, 34] have achieved close to real-time frame rates (10 fps) for a GPU implementation of the DPM detector, by exploiting coarse-to-fine search strategies. In the constructed framework, it would presumably be enough to use object detection measurements in every few frames and rely solely on the generic tracking results between these frames.

The main weakness of the proposed model is the Gaussian approximation of the part-likelihoods, as it does not model cluttered detections well. Figure 9.3 visualizes the detection likelihoods for the human and all parts, at a certain frame and scale. There are three strong human detections in this scale and three corresponding strong head detections are visible. However, almost no clear detections exist for the hips and legs. The feet likelihoods suffer from much clutter in many areas of the image. The straightforward way of handling this problem is to add the parts states that are most affected by clutter to the non-linear states in the RBPF. But this would also require an exponential increase in the number of particles to achieve the same theoretical accuracy of the posterior distribution, due to the curse of dimensionality. Another option is to approximate the part likelihoods in (6.16) with a mixture of Gaussians and use a mode parameter state to distinguish between the different hypotheses. The mode state is most simply included as a non-linear state. But since it is discrete, it would not require such a large increase in particles.

Much research has been invested in improving DPM detector framework. For example, [25] increased the performance in many object categories by combining color names (section 3.1.1) with HOG at the feature level. This could alleviate the problem with false detections and other imperfections of the detector. The problems with occlusions caused by non-tracked objects, e.g. background structures could be handled further by incorporating an occlusion model. The work of [39] uses part detections scores to determine partial occlusions.

Reidentification of objects that are lost, at for example occlusions, is an important task if object identities are of interest. This could potentially be done with the appearance models applied by the RCSK tracker. Otherwise, separate appearance models could be trained for this purpose.
Figure 9.3: Detection likelihoods $p(\theta_j | z, s)$ (figure 9.3b to 9.3j) at scale $s = 2^{12}$ computed on frame 130 (figure 9.3a) in the Town Centre sequence.
9.3 Conclusions

In this second part of the thesis, a system for category tracking is presented. The main novelties are the fusion of generic tracking with object detection scores from DPM in a causal probabilistic framework and that the RBPF is used in the filtering step. Encouraging results are demonstrated when applied to human tracking in a real-world surveillance sequence. The causal nature and real-time potential of this system make it attractive for online applications. Additionally, the estimated trajectories of human parts could be used by other systems for action detection and recognition.
Appendix
This chapter contains a brief presentation of Bayesian filtering theory. The chapter starts with the general theory and solution. The rest of this chapter presents various parts of the theory that is used in the proposed category object tracker. Section A.4 contains the algorithm and details of the Rao-Blackwellized particle filter, which is of major importance for the proposed tracker.

A.1 The General Case

Consider the general first order hidden Markov model in (A.1). The state of the system at time $t \in \mathbb{N}$ is denoted $x_t \in \mathbb{R}^n$. $y_t \in \mathbb{R}^{m_t}$ is the measurements given at $t$.

$$
\begin{align*}
    p(x_{t+1}|x_t) \quad \text{(A.1a)} \\
    p(y_t|x_t) \quad \text{(A.1b)}
\end{align*}
$$

The state transition density $p(x_{t+1}|x_t)$ model the dynamics of the system. The likelihood $p(y_t|x_t)$ model how likely it is to receive a certain measurement given the state of the system. The goal is to estimate the posterior probability $p(x_t|Y_t)$, where $Y_t = \{y_1, \ldots, y_t\}$ is the set of all measurements observed so far. The posterior is the probability distribution of the state given all information (measurements) available at that time instance.
A.1.1 General Bayesian Solution

The general solution of the Bayesian filtering problem of the model in (A.1) is given by the recursion formula in (A.2).

\[
p(x_t|Y_t) = \frac{p(y_t|x_t)p(x_t|Y_{t-1})}{p(y_t|Y_{t-1})} \quad \text{(A.2a)}
\]

\[
p(x_{t+1}|Y_t) = \int_{\mathbb{R}^n} p(x_{t+1}|x_t)p(x_t|Y_{t}) \, dx_t \quad \text{(A.2b)}
\]

The normalization factor in (A.2a) can be expressed as in (A.3).

\[
p(y_t|Y_{t-1}) = \int_{\mathbb{R}^n} p(y_t|x_t)p(x_t|Y_{t-1}) \, dx_t \quad \text{(A.3)}
\]

Equation A.2a is often called the measurement update, since the posterior is updated with the new information contained in the measurement \(y_t\). Equation A.2b is often called the time update, since it predicts the posterior in the next time instance using the modelled dynamics. The recursion can be initialized with \(p(x_1|Y_0) = p(x_1)\). The recursion formula in (A.2) can be derived by applying Bayes’ theorem and marginalization, along with the Markov properties of the model in (A.1).

In practice the recursive Bayesian solution can only be applied in some special cases where finite dimensional parametrizations of the densities exists. Section A.2 discusses such a case. Otherwise some sort of finite dimensional approximation of the densities are needed. Sections A.3 and A.4 discuss such methods.

A.1.2 Estimation

At each time instance, an estimation of the state \(x_t\) given the measurements \(Y_t\) can be calculated by using e.g. the minimum variance (MV) or maximum a posteriori (MAP) estimate.

\[
\hat{x}_{t|t}^{\text{MV}} = \int_{\mathbb{R}^n} x_t p(x_t|Y_t) \, dx_t \quad \text{(A.4a)}
\]

\[
\hat{x}_{t|t}^{\text{MAP}} = \arg \max_{x_t} p(x_t|Y_t) \quad \text{(A.4b)}
\]

A.2 The Kalman Filter

The linear Gaussian model is one of the cases where an analytic solution to (A.2) exists. Such a model is given by (A.5). It is a special case of the general model in (A.1), with
p(x_{t+1}|x_t) = \mathcal{N}(x_{t+1}; A_t x_t, B_t Q_t B_t^T) \text{ and } p(y_t|x_t) = \mathcal{N}(y_t; C_t x_t, R_t).

\begin{align*}
x_{t+1} &= A_t x_t + B_t v_t \\
y_t &= C_t x_t + e_t \\
v_t &\sim \mathcal{N}(0, Q_t) \\
e_t &\sim \mathcal{N}(0, R_t) \\
x_1 &\sim \mathcal{N}(\hat{x}_{1|0}, P_{1|0})
\end{align*}

(A.5a)

(A.5b)

(A.5c)

(A.5d)

(A.5e)

\(v_t\) and \(e_t\) are white. \(v_t\) is called process noise and \(e_t\) is measurement noise. \(A_t, B_t\) and \(C_t\) are matrices of appropriate dimensions.\(^1\) The notation \(\hat{x}_{t|k}\) and \(P_{t|k}\) denotes the estimation of the state mean and covariance respectively at time \(t\), given all measurements up to time \(k\).

### A.2.1 Algorithm

The measurement update of the Kalman filter is given in (A.6). \(K_t\) is the Kalman gain, that needs to be computed for each time instance.

\begin{align*}
\hat{x}_{t|t} &= \hat{x}_{t|t-1} + K_t (y_t - C_t \hat{x}_{t|t-1}) \\
P_{t|t} &= P_{t|t-1} - K_t C_t P_{t|t-1} \\
K_t &= P_{t|t-1} C_t^T (C_t P_{t|t-1} C_t^T + R_t)^{-1}
\end{align*}

(A.6a)

(A.6b)

(A.6c)

The time update is given in (A.7).

\begin{align*}
\hat{x}_{t+1|t} &= A_t \hat{x}_{t|t} \\
P_{t+1|t} &= A_t P_{t|t} A_t^T + B_t Q_t B_t
\end{align*}

(A.7a)

(A.7b)

The complete algorithm is given in algorithm A.1. For further details and derivation of the Kalman filter, see [18].

**Algorithm A.1** Kalman Filter Update at time \(t\)

**Input:**

Matrices: \(A_t, B_t, C_t, Q_t\) and \(R_t\)
Measurments: \(y_t\)
Prediction\(^a\) at time \(t\): \(\hat{x}_{t|t-1}\) and \(P_{t|t-1}\)

**Output:**

Estimation at time \(t\): \(\hat{x}_{t|t}\) and \(P_{t|t}\)
Prediction at time \(t + 1\): \(\hat{x}_{t+1|t}\) and \(P_{t+1|t}\)

2. Time update using (A.7).

\(^a\)The prediction is given by the model at the first iteration (i.e. \(t = 1\)) and by the previous iteration otherwise.

\(^1\)Note that the dimensions can vary dynamically since the number of measurements (i.e. dimension of \(y_t\)) can change.
A.2.2 Iterated Measurement Update

If \( y_t \) consists of several independent measurements, then the iterated measurement update can be used instead to reduce the computational cost. Let \( y_t = (y^1_t, \ldots, y^M_t)^T \), where \( y^i_t \) are the uncorrelated measurements.\(^2\) This implies that the measurement noise covariance \( R_t \) is block diagonal, with the non-zero blocks \( R^i_t = \text{Cov}(y^i_t|x_t) \). Also let \( C^i_t \) be the rows of the measurement matrix \( C_t \) associated with \( y^i_t \). The measurement update in algorithm A.1 can then be done using algorithm A.2.

Algorithm A.2 Iterated Measurement Update at time \( t \)

Input:
Matrices: \( C_t \) and \( R_t \)
Measurements: \( y_t = (y^1_t, \ldots, y^M_t)^T \)
Prediction at time \( t-1 \): \( \hat{x}_{t|t-1} \) and \( P_{t|t-1} \)

Output:
Estimation at time \( t \): \( \hat{x}_{t|t} \) and \( P_{t|t} \)

1: Set \( \hat{x}^0_t = \hat{x}_{t|t-1} \) and \( P^0_t = P_{t|t-1} \)
2: for \( i = 1, \ldots, M \) do
3: \[ K^i_t = P^{i-1}_t(C^i_t)^T(C^i_tP^{i-1}_t(C^i_t)^T + R^i_t)^{-1} \]
4: \[ \hat{x}^i_t = \hat{x}^{i-1}_t + K^i_t(y^i_t - C^i_t\hat{x}^{i-1}_t) \]
5: \[ P^i_t = P^{i-1}_t - K^i_tC^i_tP^{i-1}_t \]
6: end for
7: Set \( \hat{x}_{t|t} = \hat{x}^M_t \) and \( P_{t|t} = P^M_t \)

The major gain in using the iterated measurement update instead of (A.6) is that the matrix inversions are computed for smaller matrices. This can be used to radically increase the computational speed of the marginalized particle filter, where thousands of measurement updates (one for each particle) needs to be computed at each time instance.

A.3 The Particle Filter

Approximative methods are necessary in more general cases than the one in (A.5). The particle filter approximates the posterior over an adaptive grid. In contrast to the form of Bayesian filtering described in (A.2), the particle filter attempts to estimate the posterior of the whole trajectory \( X_t = \{x_1, \ldots, x_t\} \), i.e. \( p(X_t|Y_t) \). The filtering posterior \( p(x_t|Y_t) \) is obtained by marginalizing over the earlier states \( X_{t-1} \).

A.3.1 Algorithm

The complete algorithm is stated in algorithm A.3. It uses a proposal density \( q(x_{t+1}|x_t, y_{t+1}) \) to sample the new particle grid \( \{x^i_{t+1}\}_{i=1}^N \) given the previous grid \( \{x^i_t\}_{i=1}^N \) and the next measurement \( y_{t+1} \). The simplest and most common choice of proposal density is the prior

\(^2\)Note that \( y^i_t \) does not has to be scalar.
\( q(x_{t+1} | x_t, y_{t+1}) = p(x_{t+1} | x_t) \). Although many different proposal densities can be used depending on the application.

The resampling step in the particle filter is needed to avoid sample depletion. It is necessary to discard particles with too low weight that do not contribute significantly to the approximation of the posterior. It is important to note that even though the particle filter returns an estimate of the posterior for the whole trajectory, it is only accurate (assuming enough particles) for the last few states, because of the depletion problem.

**Algorithm A.3** Particle Filter Update at time \( t \)

**Input:**
- Number of particles: \( N \)
- Particles and predicted weights at time \( t \): \( \{ X_i^t \}_{i=1}^N, \{ w_i^t \}_{i=1}^N \)
- Proposal distribution: \( q(x_{t+1} | x_t, y_{t+1}) \)

**Output:**
- Particle weights at time \( t \): \( \{ w_i^t | t \}_{i=1}^N \)
- Particles and predicted weights at time \( t + 1 \): \( \{ X_{i+1}^t \}_{i=1}^N, \{ w_{i+1}^t | t \}_{i=1}^N \)

**Measurement update:**
1. Calculate new weights
   \[
   \tilde{w}_i^t = w_i^{t-1} p(y_t | x_i^t) \quad (A.8)
   \]
2. Normalize the weights
   \[
   w_i^t = \frac{\tilde{w}_i^t}{\sum_{i=1}^N \tilde{w}_i^t} \quad (A.9)
   \]

**Resampling:**
3. Sample \( N \) particles with replacement from the set \( \{ x_i^t \}_{i=1}^N \) with the probabilities \( \{ w_i^t | t \}_{i=1}^N \).
4. Set the weights to \( w_i^t | t = 1/N. \)

**Time update:**
5. Generate new particles using the proposal distribution
   \[ x_{i+1}^t \sim q(x_{t+1} | x_i^t, y_{t+1}) \quad (A.10) \]
6. Compute the new predicted weights
   \[
   w_{i+1}^t | t = \frac{p(x_{i+1}^t | x_i^t)}{q(x_{i+1}^t | x_i^t, y_{t+1})} w_i^t \quad (A.11)
   \]

\(^a\)In the first iteration \((t = 1)\) the particles are sampled using \( x_1^1 \sim p_{x_1} \), that is given by the model. The initial weights are set to \( w_0^1 = 1/N. \)

\(^b\)The resampling is optional in each iteration. It can be done only when needed, indicated by some measure for depletion (see [18]).

### A.3.2 Estimation

The filtering posterior is approximated by (A.12). The approximations are most commonly done before the resampling. However, it can be done after as well if enough parti-
A Bayesian Filtering

\[ \hat{p}(x_t|Y_t) = \sum_{i=1}^{N} w^i_{t} \delta(x_t - x^i_t) \]  

(A.12)

A point estimate of the state can be obtained by using this approximation in (A.4a). This results in the MV approximation given in (A.13). An approximation of the MAP estimate can be obtained by simply choosing the particle \( x^i_t \) with the highest weight \( w^i_{t} \)

\[ \hat{x}^{MV}_{t|t} = \sum_{i=1}^{N} w^i_{t} x^i_t \]  

(A.13)

A.4 The Rao-Blackwellized Particle Filter

The particle filter is in most cases unpractical for state spaces with more than a few dimensions. This is due to the fact that the number of particles has to grow exponentially with the number of states to keep the accuracy of the estimations. The Rao-Blackwellized particle filter (RBPF) [37] exploits linear-Gaussian substructures in the state space to reduce the dimensionality of the particle approximation. The rest of the states are approximated using Kalman filters. In this section the state space model in (A.14) is considered. In this model the state vector has been partitioned as \( x_t = (x^n_t, x^l_t)^T \). The partitions are called the non-linear and linear states respectively.

\[ x^n_{t+1} = f^n_t(x^n_t) + A^n_t(x^n_t)x^l_t + B^n_t(x^n_t)v^n_t \]  

(A.14a)

\[ x^l_{t+1} = f^l_t(x^n_t) + A^l_t(x^n_t)x^l_t + B^l_t(x^n_t)v^l_t \]  

(A.14b)

\[ y_t = h_t(x^n_t) + C_t(x^n_t)x^l_t + e_t \]  

(A.14c)

\[ \begin{pmatrix} v^n_t \\ v^l_t \end{pmatrix} \sim \mathcal{N}(0, Q_t), \quad Q_t = \begin{pmatrix} Q^n_t & Q^{ln}_t \\ (Q^{ln}_t)^T & Q^l_t \end{pmatrix} \]  

(A.14d)

\[ e_t \sim \mathcal{N}(0, R_t) \]  

(A.14e)

\[ x^n_1 \sim p_{x^n_1} \]  

(A.14f)

\[ x^l_1 \sim \mathcal{N}(\hat{x}^l_0, P_0) \]  

(A.14g)

\( f^n_t, f^l_t \) and \( h_t \) are vector valued functions of \( x^n_t \). \( A^n_t, A^l_t, B^n_t, B^l_t \) and \( C_t \) are matrices of appropriate dimensions that depend on \( x^n_t \). Note that this model is linear in \( x^l_t \) conditioned on \( x^n_t \).

A.4.1 Algorithm

The RBPF uses the factorization in (A.15).

\[ p(X^n_t, x^l_t|Y_t) = p(x^l_t|X^n_t, Y_t)p(X^n_t|Y_t) \]  

(A.15)

The first factor is the linear states conditioned on the non linear ones and the measurements. Given a particle approximation of \( X^n_t \), an optimal approximation of this factor can be derived. The solution is to run a Kalman filter for each particle \( X^n_{t,i} \). The
A.4 The Rao-Blackwellized Particle Filter

The time update of the linear states is given in (A.17).
\[
p(x_t^l|X_t^{n,i}, Y_t) = \mathcal{N}(x_t^l; \hat{x}_t^l, P_t^l) \quad \text{(A.16a)}
\]
\[
\hat{x}_t^l = \hat{x}_{t|t-1}^l + K_t(y_t - h_t - C_t\hat{x}_{t|t-1}^l)
\]
\[
P_{t|t} = P_{t|t-1} - K_tC_tP_{t|t-1} \quad \text{(A.16c)}
\]
\[
K_t = P_{t|t-1}C_t^T(C_tP_{t|t-1}C_t^T + R_t)^{-1} \quad \text{(A.16d)}
\]

The measurement update of the linear states is given in (A.16).
\[
p(x_t^l|X_t^n, Y_t) = \mathcal{N}(x_t^l; \hat{x}_t^l, P_t^l) \quad \text{(A.16a)}
\]
\[
\hat{x}_t^l = \hat{x}_{t|t|t-1}^l + \tilde{K}_t(z_t - A_t^n\hat{x}_{t|t-1}^l)
\]
\[
P_{t|t} = P_{t|t-1} - A_t^n\tilde{K}_tC_{t|t-1}^T(C_{t|t-1}C_{t|t-1}^T + R_t)^{-1} \quad \text{(A.16d)}
\]
\[
\tilde{K}_t = A_t^n\tilde{K}_tC_{t|t-1}(C_{t|t-1}C_{t|t-1}^T + R_t)^{-1} \quad \text{(A.16d)}
\]

The time update of the linear states is given in (A.17).
\[
p(x_{t+1}^l|X_{t+1}^n, Y_t) = \mathcal{N}(x_{t+1}^l; \hat{x}_{t+1}^l, P_{t+1|t}) \quad \text{(A.17a)}
\]
\[
\hat{x}_{t+1}^l = \hat{x}_{t+1|t}^l + \bar{B}_t^l(z_{t+1} - A_t^n\hat{x}_{t+1|t}^l)
\]
\[
P_{t+1|t} = \bar{A}_t^lP_{t|t}(\bar{A}_t^l)^T + B_t^l\tilde{Q}_t^l(B_t^l)^{-1} - L_tA_t^nP_{t|t}(\bar{A}_t^l)^T
\]
\[
L_t = \bar{A}_t^lP_{t|t}(A_t^n)^T(A_t^nP_{t|t}(A_t^n)^T + B_t^l\tilde{Q}_t^l(B_t^l)^{-1})^{-1} \quad \text{(A.17c)}
\]
\[
z_t = x_{t+1}^n - f_t^n \quad \text{(A.17b)}
\]

The bar-denoted matrices are defined in (A.18). However, if the process noises \( v_t^n \) and \( v_t^l \) are uncorrelated, i.e. \( Q_t^n = 0 \), then \( \bar{A}_t^l = A_t^l, \bar{B}_t^l = 0 \) and \( \tilde{Q}_t^l = Q_t^l \).
\[
\bar{A}_t^l = A_t^l - B_t^lA_t^n \quad \text{(A.18a)}
\]
\[
\bar{B}_t^l = B_t^l(Q_t^n)^{-1}(G_t^nQ_t^n)^{-1} \quad \text{(A.18b)}
\]
\[
\tilde{Q}_t^l = Q_t^l - (Q_t^n)^{-1}(Q_t^n)^{-1}Q_t^n \quad \text{(A.18c)}
\]

Note that (A.17) contains a measurement update using \( z_t \), followed by the ordinary Kalman filter time update. This extra measurement update in necessary to include the information from the time update in the particle filter. Equation A.14a acts as the measurement equation in this case.

The second factor in (A.15) can be factorized as in (A.19). This factorization is used for the particle approximation of the non-linear states.
\[
p(X_t^n|Y_t) = \frac{p(y_t|X_t^n, Y_{t-1})p(x_t^n|X_{t-1}^n, Y_{t-1})}{p(y_t|Y_{t-1})}p(X_{t-1}^n|Y_{t-1}) \quad \text{(A.19)}
\]

The distributions for the likelihood and prior factors are given in (A.20). The linear states
can in these cases be seen as extra measurement noise and process noise respectively.

\[
p(y_t | X^n_t, Y_{t-1}) = \mathcal{N} \left( y_t; h_t + C_t \hat{x}_{t+1|t}, C_t P_{t|t-1} C_t^T + R_t \right)
\] (A.20a)

\[
p(x^n_{t+1} | X^n_t, Y_t) = \mathcal{N} \left( x^n_{t+1}; f^n_t + A^n_t \hat{x}^l_{t|t}, A^n_t P_{t|t-1} (A^n_t)^T + B^n_t Q^n_t (B^n_t)^T \right)
\] (A.20b)

The complete algorithm is given in algorithm A.4. As the particle filter, it uses a general proposal distribution \( q( x_{t+1} | X_t, Y_{t+1} ) \). The most common example of a proposal distribution is the prior in (A.20b). See [29] for details on proposal distributions, and a more detailed description of the RBPF. Also see [37] for proofs and details on some special cases.

### A.4.2 Estimation

The filtering posterior and point estimates of the non-linear states can be calculated in the same way as for the particle filter in section A.3.2. The posterior of the linear states can be approximated as in (A.25), by a mixture of Gaussians.

\[
\hat{p}(x^l_t | Y_t) = \sum_{i=1}^{N} w_{t|t}^i \mathcal{N} \left( x^l_t; \hat{x}^{l|i}_{t|t}, P^{l|i}_{t|t} \right)
\] (A.25)

The MAP estimate of the linear states can be approximated with the \( \hat{x}^{l|i}_{t|t} \) that corresponds to the highest weight \( w_{t|t}^i \). The minimum variance (MV) estimate is calculated using (A.26).

\[
\hat{x}^l_{t|t} = \sum_{i=1}^{N} w_{t|t}^i \hat{x}^{l|i}_{t|t}
\] (A.26a)

\[
\hat{P}^l_{t|t} = \sum_{i=1}^{N} w_{t|t}^i \left( P^i_{t|t} + (\hat{x}^{l|i}_{t|t} - \hat{x}^l_{t|t})(\hat{x}^{l|i}_{t|t} - \hat{x}^l_{t|t})^T \right)
\] (A.26b)

Note that the measurement update of the linear states in algorithm A.4 is placed after the resampling step. It can thus be more convenient to extract all estimates after step 6 in the algorithm.
Algorithm A.4 Rao-Blackwellized Particle Filter Update at time $t$

**Input:**
- Number of particles: $N$
- Particles and predicted weights at time $t$: \( \{X_{t}^{n,i}\}_{i=1}^{N}, \{w_{t+1|i}^{i}\}_{i=1}^{N} \)
- Predicted linear states at time $t$: \( \{\hat{x}_{t+1|i}^{i}\}_{i=1}^{N}, \{P_{t+1|i}^{i}\}_{i=1}^{N} \)
- Proposal distribution: \( q(x_{t+1}^{n}, Y_{t+1}) \)

**Output:**
- Particle weights at time $t$: \( \{w_{t|i}^{i}\}_{i=1}^{N} \)
- Particles and predicted weights at time $t+1$: \( \{X_{t+1}^{n,i}\}_{i=1}^{N}, \{w_{t+1|i}^{i}\}_{i=1}^{N} \)
- Estimated linear states at time $t$: \( \{\hat{x}_{t|i}^{i}\}_{i=1}^{N}, \{P_{t|i}^{i}\}_{i=1}^{N} \)
- Predicted linear states at time $t+1$: \( \{\hat{x}_{t+1|i}^{i}\}_{i=1}^{N}, \{P_{t+1|i}^{i}\}_{i=1}^{N} \)

**Particle filter measurement update:**
1: Calculate new weights using (A.20a)
   \[ \tilde{w}_{t|i}^{i} = w_{t|i}^{i} p(y_{t}|X_{t}^{n,i}, Y_{t-1}) \]  \hspace{1cm} (A.21)
2: Normalize the weights
   \[ w_{t|i}^{i} = \frac{\tilde{w}_{t|i}^{i}}{\sum_{i=1}^{N} \tilde{w}_{t|i}^{i}} \]  \hspace{1cm} (A.22)
3: Sample a set indices $J_{t} = \{j_{k}\}_{k=1}^{N}$ with the probabilities \( p(j_{k}^{i} = i) = w_{t|i}^{i}, \forall i, k. \)
4: Set $X_{t}^{n,k} = X_{t}^{n,j_{k}}, \hat{x}_{t|i}^{k} = \hat{x}_{t|i}^{j_{k}}$ and $P_{t|i}^{k} = P_{t|i}^{j_{k}}$ for $k = 1, \ldots, N.$
5: Set the weights to \( w_{t|i}^{k} = 1/N. \)

**Kalman filter measurement update:**
6: Kalman filter measurement update for each particle $X_{t}^{n,i}$ using (A.16).

**Particle filter time update:**
7: Generate new particles using the proposal distribution
   \[ x_{t+1|i}^{n,i} \sim q(x_{t+1}^{n,i}|X_{t}^{n,i}, Y_{t+1}) \]  \hspace{1cm} (A.23)
8: Compute the new predicted weights using (A.20b)
   \[ w_{t+1|i}^{i} = \frac{w_{t|i}^{i} p(x_{t+1|i}^{n,i}|X_{t}^{n,i}, Y_{t})}{q(x_{t+1|i}^{n,i}|X_{t}^{n,i}, Y_{t+1})} \]  \hspace{1cm} (A.24)

**Kalman filter time update:**
9: Kalman filter time update for each particle $X_{t}^{n,i}$ using (A.17).

---

\(^{a}\) In the first iteration ($t = 1$) the particles are sampled using $x_{1|i}^{n,i} \sim p_{x_{1|i}^{n,i}}$. The initial weights are set to $w_{1|i}^{0} = 1/N.$

\(^{b}\) In the first iteration ($t = 1$) the linear states are initialized as $\hat{x}_{0}^{i} = \hat{x}_{0}$ and $P_{0|i}^{i} = P_{0}$ for all $i$.

\(^{c}\) The resampling is optional in each iteration (see algorithm A.3).
B

Proofs and Derivations

B.1 Derivation of the RCSK Tracker Algorithm

B.1.1 Kernel Function Proofs

This section proves the propositions in section 2.2.4. This is done by first showing the following result.

B.1 Lemma. The inner product on $\ell_p^D(M, N)$ is a shift invariant kernel.

Proof: The inner product is clearly a valid kernel function. Using the definition of the standard scalar product in $\ell_p^D(M, N)$ and the periodicity of $f$ and $g$ we get.

$$\langle \tau_{m,n}f, \tau_{m,n}g \rangle = \sum_{d=1}^D \sum_{k,l} f^d(k - m, l - n)g^d(k - m, l - n)$$

$$= \sum_{d=1}^D \sum_{r,s} f^d(r, s)g^d(r, s) = \langle f, g \rangle$$

(B.1)

Since $f, g, m, n$ were arbitrary, this is valid for all $f, g \in \ell_p^D(M, N)$ and $m, n \in \mathbb{Z}$. □

The results in section 2.2.4 can now be shown using this basic lemma.

Proof of proposition 2.2: The shift invariance follows directly from (2.14) and lemma B.1. Equation 2.15 follows from the correlation property and linearity of the DFT (see [16]).
\[ \kappa(\tau_{-m,-n} f, g) = k \left( \sum_{d=1}^{D} \sum_{k,l} f^d(k + m, l + n) g^d(k, l) \right) \]
\[ = k \left( \sum_{d=1}^{D} g * f(m, n) \right) = k \left( \mathcal{F}^{-1} \left\{ \sum_{d=1}^{D} F^d G^d \right\} \right)(m, n) \quad (B.2) \]

**Proof of proposition 2.3:** Equation 2.16 can be expanded as:
\[ \kappa(f, g) = k \left( \|f - g\|^2 \right) = k \left( \langle f - g, f - g \rangle \right) \]
\[ = k \left( \langle f, f \rangle + \langle g, g \rangle - 2 \langle f, g \rangle \right) = k \left( \|f\|^2 + \|g\|^2 - 2 \langle f, g \rangle \right) \quad (B.3) \]
The shift invariance now follows from lemma B.1. The proof of (2.17) is similar to the one of (2.15), using (B.3) and applying lemma B.1 to get \( \|\tau_{-m,-n} f\|^2 = \|f\|^2 \).  

**B.1.2 Derivation of the Robust Appearance Learning Scheme**

This section derives that \( A = F \{ a \} \) in (2.20) is the minimum of the cost function in (2.19). The cost function can be rewritten to (B.4), by inserting \( v^j \) in (2.19b) to (2.19a)
\[ \epsilon = \sum_{j=1}^{J} \beta_j \left( \sum_{m,n} \left( \sum_{k,l} a(k,l) \kappa(x^j_{m,n}, x^j_{k,l}) - y^j(m,n) \right)^2 + \lambda \sum_{m,n} a(m,n) \sum_{k,l} a(k,l) \kappa(x^j_{m,n}, x^j_{k,l}) \right) \quad (B.4) \]

This function is clearly convex in \( a \), since the squared \( L^2 \)-norm of an affine transformation is convex and \( \epsilon \) is a sum of such functions. The global minimum can thus be found by finding a stationary point. The derivative with respect to \( a(r, s) \) is computed in (B.5).
\[ \frac{\partial \epsilon}{\partial(a(r, s))} = \]
\[ = 2 \sum_{j=1}^{J} \beta_j \sum_{m,n} \kappa(x^j_{m,n}, x^j_{r,s}) \left( \sum_{k,l} a(k,l) \kappa(x^j_{m,n}, x^j_{k,l}) - y^j(m,n) + \lambda a(m,n) \right) \]
\[ = 2 \sum_{j=1}^{J} \beta_j \sum_{m,n} \kappa(x^j_{r-m,s-n}, x^j) \left( \sum_{k,l} a(k,l) \kappa(x^j_{m-k,n-l}, x^j) - y^j(m,n) + \lambda a(m,n) \right) \quad (B.5) \]

Here we have used the symmetric property of the kernel function, i.e. \( \kappa(x, z) = \kappa(z, x) \) and the shift invariance that is defined in definition 2.1. We define the function \( u^j_x \in \ell_p(M, N) \) in (B.6).
\[ u^j_x(m,n) = \kappa(x^j_{m,n}, x^j) \quad (B.6) \]
Using this definition, the derivative in (B.5) can be expressed as:

\[
\frac{1}{2} \frac{\partial \epsilon}{\partial (a(r, s))} = \sum_{j=1}^{J} \beta_j \sum_{m,n} u_x^j (r - m, s - n) \left( \sum_{k,l} a(k,l) u_x^j (m - k, n - l) - y^j (m, n) + \lambda a(m,n) \right)
\]

\[
= \sum_{j=1}^{J} \beta_j \sum_{m,n} u_x^j (r - m, s - n) \left( a \ast u_x^j (m, n) - y^j (m, n) + \lambda a(m,n) \right)
\]

\[
= \sum_{j=1}^{J} \beta_j u_x^j \ast (a \ast u_x^j - y^j + \lambda a) (r, s) \tag{B.7}
\]

By setting these derivatives to zero we get.

\[
\frac{\partial \epsilon}{\partial (a(r, s))} = 0, \quad \forall r, s \in \mathbb{Z}
\]

\[
\iff \sum_{j=1}^{J} \beta_j u_x^j \ast (a \ast u_x^j - y^j + \lambda a) = 0
\]

\[
\iff \mathcal{F} \left\{ \sum_{j=1}^{J} \beta_j u_x^j \ast (a \ast u_x^j - y^j + \lambda a) \right\} = 0
\]

\[
\iff \sum_{j=1}^{J} \beta_j U_x^j (U_x^j A - Y^j + \lambda A) = 0
\]

\[
\iff A \sum_{j=1}^{J} \beta_j U_x^j (U_x^j + \lambda) - \sum_{j=1}^{J} \beta_j Y^j U_x^j = 0
\]

\[
\iff A = \frac{\sum_{j=1}^{J} \beta_j Y^j U_x^j}{\sum_{j=1}^{J} \beta_j U_x^j (U_x^j + \lambda)} \tag{B.8}
\]

Here we have used that the DFT is linear and invertible, along with the convolution property (see [16]). The last equivalence assumes that all frequency components in the denominator are zero. This completes the derivation of (2.20).

### B.2 Proof of Equation 6.13

This section proves (6.13), that is used to update the particle weights in the RBPF.
B.2.1 Proof of Uncorrelated Parts

This section proves that (B.9) holds, when the RBPF is used on the described model.

\[ p(x_t^i | X_t^n, Y_{t-1}) = \mathcal{N}(v_t; \hat{v}_{t|t-1}, P_{t|t-1}^v) \prod_{j=1}^{N} \mathcal{N}(z_t^j; \hat{z}_{t|t-1}^j, P_{t|t-1}^z) \]  

(B.9)

From the RBPF we have that \( p(x_t^i | X_t^n, Y_{t-1}) = \mathcal{N}(x_t^i; \hat{x}_{t|t-1}^i, P_{t|t-1}^x) \) (see [37] for proof). So, it only has to be proven that \( P_{t|t-1} \) is block diagonal with the \( 2 \times 2 \) blocks \( P^z_{t|t-1}, P^z_{t|t-1}, \ldots, P^z_{t|t-1} \). From the model, this is true for \( t = 1 \). Now assume that it is also true for a \( t \leq 1 \).

We first prove that \( P_{t|t} \) is \( 2 \times 2 \)-block diagonal using the iterated measurement update in algorithm A.2. First let \( \{y_1^t, \ldots, y_N^t\} \) be the independent position measurements from the image likelihood. We have \( C^i_t = (0_{2 \times 2}, I_{2 \times 2}, 0_{2 \times 2(N-t)}) \). Assume that \( P_{t|t-1}^i \) in algorithm A.2 is \( 2 \times 2 \)-block diagonal. Using the algorithm it is easy to verify that \( K_t^i C^i_t P_{t|t-1}^i \) is only non-zero in diagonal block number \( i \). \( P_{t|t}^i \) is thus also block diagonal. It follows that \( P_{t|t}^N \) is block diagonal since \( P_{t|t}^0 = P_{t|t-1}^N \) is block diagonal from the assumption. Now let \( \{y_1^{N+1}, \ldots, y_2^{2N}\} \) be the independent position measurements from the deformation likelihood. The only difference now is that \( C^i_t \) is multiplied with a scalar, so the same holds here. \( P_{t|t} = P_{t|t}^N \) is thus block diagonal.

Now consider the time update in (A.17). Using the model in (6.11) it is easy to verify that \( L_t = (L_t^1, 0_{2 \times 2N})^T \) for some \( 2 \times 2 \) matrix \( L_t^1 \). This implies that the last term in (A.17c) is only non-zero in the first diagonal block. The first two terms in (A.17c) are also clearly block diagonal since \( Q_t^1 \) in (6.12) is \( 2 \times 2 \)-block diagonal. This implies that \( P_{t+1|t} \) is \( 2 \times 2 \)-block diagonal and that the initial statement is valid for \( t + 1 \). This proves by induction that (B.9) is valid for all \( t \).

B.2.2 Derivation of the Weight Update

To simplify notation, the likelihood function in (B.10) is defined.

\[ L_t^i(z, s) = p(\theta^i_t | z) p(\varphi^i_t | z) \]  

(B.10)

The measurement model from section 6.3 can then be written as in (B.11).

\[ p(y_t | x_t) = L_t^0(z_t^0, s_t) \prod_{j=1}^{N} \mathcal{N}(z_t^j; \hat{z}_{t|t}^j, D^j) \]

(B.11)

Let \( l \) be the number of linear states, in our case \( l = 2(N+1) \). It follows that.

\[ p(y_t | X_t^n, Y_{t-1}) = \int_{\mathbb{R}^l} p(y_t | x_t^l | X_t^n, Y_{t-1}) \, dx_t^l \]

\[ = \int_{\mathbb{R}^l} p(y_t | x_t^l | X_t^n, Y_{t-1}) p(x_t^l | X_t^n, Y_{t-1}) \, dx_t^l \]

\[ = \int_{\mathbb{R}^l} p(y_t | x_t) p(x_t | X_t^n, Y_{t-1}) \, dx_t^l \]  

(B.12)
The last step follows from the Markov property of the model. Using (B.9) and (B.11) in (B.12) gives.
\[
p(y_t | X^a_t, Y_{t-1}) = L^0_t(z^0_t, s_t) \prod_{j=1}^{N} \int_{\mathbb{R}^2} N(z^j_t; z^j_{t|t-1}, P^z_{t|t-1}) \cdot N(P^{z^j} \left| a^j, \frac{z^j_t}{s_t}, D^j \right) L^j_t(z^0_t + z^j_t, s_t) dz^j_t
\]
(B.13)

Notice that the linear state \( v_t \) has been marginalized away. We will now consider the product of the two Gaussian functions inside the integral. The indices are skipped to simplify the notation.
\[
N(z; \hat{z}, P)N(a; \frac{z}{s}, D) = \frac{1}{(2\pi)^2 \sqrt{\det(P) \det(D)}} \cdot \exp \left( -\frac{1}{2} \left( (z - \hat{z})^T P^{-1} (z - \hat{z}) + \left( a - \frac{z}{s} \right)^T D^{-1} \left( a - \frac{z}{s} \right) \right) \right)
\]
(B.14)

We define \( a_s = sa \) and \( D_s = s^2 D \) to simplify the equations. The exponent \( V \) can then be written as follows.
\[
V = (z - \hat{z})^T P^{-1} (z - \hat{z}) + (a_s - z)^T D^{-1}_s (a_s - z)
= z^T (P^{-1} + D^{-1}_s) z - 2z^T (P^{-1} \hat{z} + D^{-1}_s a_s) + \hat{z}^T P^{-1} \hat{z} + a_s^T D^{-1}_s a_s
\]
(B.15)

We define the quantities in (B.16).
\[
H = (P^{-1} + D^{-1}_s)^{-1} \quad \text{(B.16a)}
\mu = H (P^{-1} \hat{z} + D^{-1}_s a_s) \quad \text{(B.16b)}
\]

It is easy to check that (B.15) can be rewritten to (B.17).
\[
V = (z - \mu)^T H^{-1} (z - \mu) + \hat{z}^T P^{-1} \hat{z} + a_s^T D^{-1}_s a_s - \mu^T H^{-1} \mu =: V_1 + V_2
\]
(B.17)

\( V_2 \) is the part of \( V \) that is independent of \( z \).
\[
V_2 = \hat{z}^T (P^{-1} - P^{-1} H P^{-1}) \hat{z} + a_s^T (D^{-1}_s - D^{-1}_s H D^{-1}_s) a_s - 2a_s^T D^{-1}_s H P^{-1} \hat{z}
\]
(B.18)

The matrix inversion lemma is given in (B.19), where \( A \) and \( C \) are invertible matrices.
\[
(A - BCD)^{-1} = A^{-1} + A^{-1} B (C^{-1} - DA^{-1} B)^{-1} DA^{-1}
\]
(B.19)
Using this lemma, we get.
\[
(P^{-1} - P^{-1}HP^{-1})^{-1} = P + (H^{-1} - P^{-1})^{-1} \\
= P + (P^{-1} + D_s^{-1} - P^{-1})^{-1} = P + D_s \tag{B.20}
\]

And similarly.
\[
(D_s^{-1} - D_s^{-1}HD_s^{-1})^{-1} = P + D_s \tag{B.21}
\]

Additionally we have that.
\[
D_s^{-1}HP^{-1} = (P(P^{-1} + D_s^{-1})D_s)^{-1} = (P + D_s)^{-1} \tag{B.22}
\]

Equation B.18 can thus be simplified as.
\[
V_2 = (\hat{z} - a_s)^T(P + D_s)^{-1}(\hat{z} - a_s) = \left(\frac{\hat{z}}{s} - a\right)^T \left(\frac{1}{s^2}P + D\right)^{-1} \left(\frac{\hat{z}}{s} - a\right) \tag{B.23}
\]

Finally we note that (B.22) implies (B.24), where we have used the properties of matrix determinant.
\[
\frac{1}{\det(P + D_s)} = \frac{\det(H)}{s^4 \det(P) \det(D)} \tag{B.24}
\]

Using (B.17), (B.23) and (B.24) in (B.14) gives.
\[
\mathcal{N}(z; \hat{z}, P)\mathcal{N}(a; \frac{\hat{z}}{s}, D) = \mathcal{N}(z; \mu, H)\mathcal{N}\left(\frac{\hat{z}}{s}; a, \frac{1}{s^2}P + D\right) \tag{B.25}
\]

Using this result in (B.13) and moving the factor that is independent of $z_i^j$ out of the integral gives (6.13) with the definitions in (6.14) and (6.15).


[38] Laura Sevilla-Lara and Erik Learned-Miller. Distribution fields for tracking. In *IEEE


Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for his/her own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/

© Martin Danelljan