Visual Tracking Using Stereo Images

Carl Dehlin
Master of Science Thesis in Electrical Engineering

Visual Tracking Using Stereo Images

Carl Dehlin

LiTH-ISY-EX–18/5181–SE

Supervisor:  
Gustav Häger  
isy, Linköpings universitet

Elisabeth Schold Linnér  
Unibap AB

Examiner:  
Michael Felsberg  
isy, Linköpings universitet

Computer Vision Laboratory  
Department of Electrical Engineering  
Linköping University  
SE-581 83 Linköping, Sweden

Copyright © 2018 Carl Dehlin
Abstract

Visual tracking concerns the problem of following an arbitrary object in a video sequence. In this thesis, we examine how to use stereo images to extend existing visual tracking algorithms, which methods exist to obtain information from stereo images, and how the results change as the parameters to each tracker vary. For this purpose, four abstract approaches are identified, with five distinct implementations. Each tracker implementation is an extension of a baseline algorithm, MOSSE. The free parameters of each model are optimized with respect to two different evaluation strategies called NOR- and WIR-tests, and four different objective functions, which are then fixed when comparing the models against each other. The results are created on single target tracks extracted from the KITTI tracking dataset, and the optimization results show that none of the objective functions are sensitive to the exposed parameters under the joint selection of model and dataset. The evaluation results also shows that none of the extensions improve the results of the baseline tracker.
Acknowledgments

I want to thank Unibap AB for giving me the opportunity to do this thesis at their premises. Especially I want to thank my supervisor at the company, Elisabeth Schold Linnér, for her commitment into this thesis, for taking her time to read the endless number of pages for so many times, providing excellent feedback and in general being the torch that lights the path in the darkest weeks during writing this document. This thesis would not be what it is without her support.

I also want to thank my supervisor at the university, Gustav Häger.

Lastly I want to thank my examiner, Michael Felsberg, for taking his time to read the thesis, and for providing invaluable constructive feedback.

Linköping, November 2018

Carl Dehlin
# Contents

Notation xi

1 Introduction 1
  1.1 Motivation .................................................. 1
  1.2 Purpose ....................................................... 1
  1.3 Problem Formulation ........................................... 1
  1.4 Delimitations .................................................. 2
  1.5 A note on notation ............................................ 2
    1.5.1 Vectors, matrices, and tensors .......................... 2

2 Background 3
  2.1 Definitions and fundamental concepts ........................ 3
  2.2 Related work .................................................. 4
    2.2.1 Historical contributions ................................ 5
    2.2.2 Discriminative Correlation Filters (DCF) .............. 12
    2.2.3 Deep Learning ........................................... 19
    2.2.4 Tracking Using Depth Information ........................ 21
  2.3 Disparity estimation ........................................... 24
  2.4 Datasets ....................................................... 25

3 Method 31
  3.1 Approach to the problem ..................................... 31
    3.1.1 Preliminaries ............................................ 31
    3.1.2 Problem identification .................................. 31
    3.1.3 Possible solutions ...................................... 31
  3.2 Optimization .................................................. 38
  3.3 Evaluation ..................................................... 38
  3.4 Experimental Setup .......................................... 41
    3.4.1 Dataset .................................................. 41
    3.4.2 Evaluation .............................................. 41

4 Results 43
5 Analysis and discussion
  5.1 Analysis of the results ........................................ 51
  5.1.1 OpenCV ......................................................... 52
  5.2 Discussion ....................................................... 52
  5.2.1 Analysis of the proposed methods .......................... 52
  5.2.2 Critique against the method ................................. 54

6 Conclusion and future work
  6.1 Summary .......................................................... 55
  6.2 Contributions .................................................... 55
  6.3 Future work ....................................................... 56
    6.3.1 Optimization ............................................... 56
    6.3.2 Disparity .................................................... 56
    6.3.3 Capacity of linear and non-linear tracker models ...... 56

A Algorithms
  A.1 Creating the citation graph .................................... 59

B Derivations
  B.1 Multi-channel MOSSE ............................................ 61

C Figures .............................................................. 63

D Tables ..................................................................... 111

Bibliography ................................................................ 121

Index .......................................................................... 127
Notation

The notation used in this report is summarized below. See section “A note on notation” in the introduction chapter for an elaborate explanation of the notation used in this report.

Typesetting

<table>
<thead>
<tr>
<th>Notation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>A scalar variable</td>
</tr>
<tr>
<td>$\mathbf{x}$</td>
<td>A geometrically interpretable vector variable</td>
</tr>
<tr>
<td>$X$</td>
<td>A matrix variable</td>
</tr>
<tr>
<td>$\mathbf{x}$</td>
<td>A tensor variable</td>
</tr>
<tr>
<td>$\mathbf{X}$</td>
<td>A tensor variable reinterpreted as a matrix in two indices</td>
</tr>
<tr>
<td>$\mathbb{X}$</td>
<td>A set variable</td>
</tr>
<tr>
<td>$x_i$</td>
<td>Variable indexing</td>
</tr>
<tr>
<td>$x^{ij...}$</td>
<td>Variable subscription</td>
</tr>
<tr>
<td>$x^i$</td>
<td>Variable multi-index subscription</td>
</tr>
<tr>
<td>$x(u)$</td>
<td>Signal evaluation through interpolation</td>
</tr>
<tr>
<td>$f(x)$</td>
<td>Continuous function evaluation of a signal</td>
</tr>
<tr>
<td>$f(x_0, \ldots, x_n)$</td>
<td>Continuous function evaluation of multiple signals</td>
</tr>
<tr>
<td>$f(x_0, \ldots ; y_0, \ldots)$</td>
<td>Variable argument grouping</td>
</tr>
</tbody>
</table>
## Default variable usage

<table>
<thead>
<tr>
<th>Notation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a, c$</td>
<td>Coefficient</td>
</tr>
<tr>
<td>$b$</td>
<td>Bias</td>
</tr>
<tr>
<td>$d$</td>
<td>Metric</td>
</tr>
<tr>
<td>$e$</td>
<td>Unit vector</td>
</tr>
<tr>
<td>$f, g, h$</td>
<td>functions</td>
</tr>
<tr>
<td>$i, j, k, l, m, n$</td>
<td>Index variables</td>
</tr>
<tr>
<td>$p$</td>
<td>Probability</td>
</tr>
<tr>
<td>$q, r, s, t, u, v$</td>
<td>Geometric entities</td>
</tr>
<tr>
<td>$x$</td>
<td>Input variable</td>
</tr>
<tr>
<td>$y$</td>
<td>Output variable</td>
</tr>
<tr>
<td>$z$</td>
<td>Auxiliary variable, Latent variable</td>
</tr>
<tr>
<td>$w$</td>
<td>Model weight, Image template</td>
</tr>
<tr>
<td>$P$</td>
<td>Linear projection mapping</td>
</tr>
<tr>
<td>$T, U, V$</td>
<td>Linear mappings</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Learning rate</td>
</tr>
<tr>
<td>$\beta, \gamma$</td>
<td>Sequence weight</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Kronecker delta, step function, indicator function</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>Residual, error</td>
</tr>
<tr>
<td>$\eta$</td>
<td>Noise</td>
</tr>
<tr>
<td>$\theta$</td>
<td>Parameter</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Regularization parameter</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Mean</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Standard deviation, Sigmoid function</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Threshold parameter</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Robust statistic, robust function</td>
</tr>
<tr>
<td>$\varphi, \phi$</td>
<td>Non-linear mapping</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Frequency</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>Set of parameters</td>
</tr>
<tr>
<td>$\sum$</td>
<td>Summation, covariance</td>
</tr>
<tr>
<td>$\prod$</td>
<td>Product</td>
</tr>
</tbody>
</table>

## Sets and spaces

<table>
<thead>
<tr>
<th>Notation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>The empty set</td>
</tr>
<tr>
<td>$\mathbb{N}$</td>
<td>The set of natural numbers</td>
</tr>
<tr>
<td>$\mathbb{Z}$</td>
<td>The set of integer numbers</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>$l_2$</td>
<td>The space of square summable sequences</td>
</tr>
<tr>
<td>$L_2$</td>
<td>The space of square integrable functions</td>
</tr>
<tr>
<td>Notation</td>
<td>Explanation</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>$\langle x, y \rangle$</td>
<td>Inner product</td>
</tr>
<tr>
<td>$</td>
<td>x</td>
</tr>
<tr>
<td>$|x|_p$</td>
<td>P-norm</td>
</tr>
<tr>
<td>$|x|$</td>
<td>2-norm</td>
</tr>
<tr>
<td>$\bar{x}$</td>
<td>Element-wise conjugate</td>
</tr>
<tr>
<td>$x^T$</td>
<td>Transpose</td>
</tr>
<tr>
<td>$x^H$</td>
<td>Conjugate transpose</td>
</tr>
<tr>
<td>$\propto$</td>
<td>Proportional to</td>
</tr>
<tr>
<td>$x_\perp$</td>
<td>Perpendicular (given by context)</td>
</tr>
<tr>
<td>$x_\parallel$</td>
<td>Parallel (given by context)</td>
</tr>
<tr>
<td>$x^*$</td>
<td>Optimal variable</td>
</tr>
<tr>
<td>$x'$</td>
<td>Predicted/modified/corrected variable</td>
</tr>
<tr>
<td>$\tilde{x}$</td>
<td>Interpolated variable</td>
</tr>
<tr>
<td>$(x_k)$</td>
<td>Concatenation of variables $x_k$ over index $k$</td>
</tr>
<tr>
<td>${x_k}$</td>
<td>Set of variables $x_k$ over index $k$</td>
</tr>
<tr>
<td>$y := x$</td>
<td>$y$ defined as $x$</td>
</tr>
<tr>
<td>$\mathcal{F}$</td>
<td>Fourier transform</td>
</tr>
<tr>
<td>$\mathcal{F}[x]$</td>
<td>Fourier transform of $x$</td>
</tr>
<tr>
<td>$\mathcal{F}<a href="%5Comega">x</a>$</td>
<td>Fourier transform of $x$ evaluated at $\omega$</td>
</tr>
<tr>
<td>$\hat{x}$</td>
<td>Fourier transform of $x$</td>
</tr>
<tr>
<td>$\hat{x}(\omega)$</td>
<td>Fourier transform of $x$ evaluated at $\omega$</td>
</tr>
<tr>
<td>$L$</td>
<td>Likelihood</td>
</tr>
<tr>
<td>$\log L$</td>
<td>Log-likelihood</td>
</tr>
<tr>
<td>$\mathcal{N}$</td>
<td>Gaussian distribution</td>
</tr>
<tr>
<td>diag($x$)</td>
<td>Diagonal matrix with \ldots along the diagonal</td>
</tr>
<tr>
<td>svd($x$)</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Explanation</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>SSD</td>
<td>Sum of Squares Deviations</td>
</tr>
<tr>
<td>SAD</td>
<td>Sum of Absolute Deviations</td>
</tr>
<tr>
<td>NCC</td>
<td>Normalized Cross Correlation</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>DCF</td>
<td>Discriminative Correlation Filter</td>
</tr>
<tr>
<td>KCF</td>
<td>Kernelized Correlation Filter</td>
</tr>
<tr>
<td>EMD</td>
<td>Earth Movers Distance</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>MIL</td>
<td>Multiple Instance Learning</td>
</tr>
<tr>
<td>GMM</td>
<td>Gaussian Mixture Model</td>
</tr>
<tr>
<td>HOG</td>
<td>Histogram of Oriented Gradients</td>
</tr>
<tr>
<td>ICP</td>
<td>Iterative Closest Point</td>
</tr>
<tr>
<td>LBP</td>
<td>Local Binary Pattern</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman Filter</td>
</tr>
<tr>
<td>IMU</td>
<td>Inertial Measurement Unit</td>
</tr>
<tr>
<td>IMCMC</td>
<td>Interactive Monte Chain Monte Carlo</td>
</tr>
<tr>
<td>NOR</td>
<td>No Reset</td>
</tr>
<tr>
<td>WIR</td>
<td>With Reset</td>
</tr>
<tr>
<td>BM</td>
<td>Block Matching</td>
</tr>
<tr>
<td>SGM</td>
<td>Semi Global Matching</td>
</tr>
<tr>
<td>IPRF</td>
<td>Image Plane Response Fusion</td>
</tr>
<tr>
<td>MVL</td>
<td>Multiple View Learning</td>
</tr>
</tbody>
</table>
1.1 Motivation

Visual object tracking has many applications. Examples are in the context of surveillance, where a potentially dangerous entity is spotted by a camera and needs to be followed in real time. Running a visual tracking algorithm can then efficiently find the entity by processing huge amounts of video data, a task that would otherwise be overwhelming for a human being to perform in real time.

Another application is in the context of robotics where tracking can be used to close the sensor-actuator loop, with low latency and predictable execution times for the algorithms being used. An example is a robot arm that is used to manipulate irregularly moving objects.

1.2 Purpose

The purpose of this thesis is to examine the impact of using a stereo camera on single target visual object tracking. A lot of research has been done using RGB-images, but very little has been written on how to utilize the depth information acquired from various depth sensors. This thesis aims to contribute to this specific sub-field of visual tracking, and to show potential paths for further research in the area.

1.3 Problem Formulation

- What methods exist for obtaining depth information from stereo images?
• How can this depth information be incorporated into visual tracking algorithms?
• How do the results change as the parameters to each tracker vary?

1.4 Delimitations

Because of limited amount of time this thesis will only examine stereo images as the source for obtaining depth information. Information from other sensors such as structured light or laser scanners will not be considered. For the same reasons, all algorithms in this thesis will be based on one selected method, MOSSE (see section 2.2.2 for an introduction).

1.5 A note on notation

A summary of the notation used in this thesis is given in the Notation preface. Different types of variables and functions are separated through typesetting, as given in table Typesetting. The default meaning of a variable is given in table Default variable usage. Commonly used sets and spaces are given in table Sets and spaces and special operators and functions are given in table Operators and functions. Finally abbreviations are given in table Abbreviations.

In order to keep the notation general and compact, images in a stereo pair will be referred to as views. This is to show that the approaches introduces in this thesis can be extended to multiple images.

1.5.1 Vectors, matrices, and tensors

In this thesis regularly spaced multi-way arrays will be referred to as tensors, that is, tensors are not objects in the formal way used in e.g. physics, where tensors are mathematical objects that should be invariant under some differential transformations. This is considered convention in the field of machine learning, which is why this convention is also adapted here.

Often tensor variables can be reinterpreted as vector variables. This is done by throwing away the shape information about the tensor $\mathbf{x}^i = \mathbf{x}^{i_1,\ldots,i_n}$ and treating it as a variable with only one mode $\mathbf{x}^i$. This is sometimes done implicitly when the context does not require the spatial information to be kept, for example, a training set of examples will often be denoted as $X = (\mathbf{x}_1,\ldots,\mathbf{x}_n)$, where the samples $\mathbf{x}_k$ are vectorized and concatenated into the matrix $X$. This convention will in many cases keep the notation compact and clean.
2
Background

2.1 Definitions and fundamental concepts

Visual object tracking concerns the computer vision problem of tracking objects in image sequences. In it’s most general terms it can be defined as follows

**Definition 2.1 (Visual Object Tracking).** Given an image sequence \( \{x_i\} \) and an initial annotated frame containing an image \( x_0 \) and an initial state \( s_0 \), predict the states \( \{s_i\}_{i>0} \) for the following frames \( \{x_i\}_{i>0} \).

Definition 2.1 neither specifies any error function to minimize the prediction error over nor the nature of the states. It is however very common to use only the image plane bounding box as annotation and mean bounding box overlap as error measure for the predictions [39, 55, 57]. Prediction means that any algorithm solving the problem should in every instance only depend on the initial state and the outputs from previous time steps,

\[
s'_t = f(x_t, x_{t-1}, \ldots, x_0; s'_{t-1}, \ldots, s'_{1}, s_0; \theta),
\]

where \( s'_k \) are predicted states and \( \theta \) are the free parameters of the tracker. A tracker is said to be *Markovian* if the output prediction \( s_t \) at time \( t \) only depends on the input \( x_t \), an accumulated state variable \( w_{t-1} \) from the previous time step, and the free parameters \( \theta \).
\[ s'_t = f(x_t, w_{t-1}, \theta), \]
\[ w_t = g(x_{t-1}, s'_{t-1}, w_{t-1}, \theta). \]  

The bounding box overlap is defined as the intersection over union (IoU) between two boxes \( r \) and \( s \) as

\[ \text{IoU}(r, s) = \frac{r \cap s}{r \cup s}. \]

The definition of the bounding box overlap is visualized in figure 2.1.

![Figure 2.1: Bounding box overlap](a) Bounding box intersection (b) Bounding box union

A special case of the visual tracking problem is when only a single target is to be tracked. This together with the previous paragraph leads us to the definition of the problem that is being tackled in this thesis.

**Definition 2.2 (Single Target Visual Object Tracking).** Given an image sequence \( \{x_i\} \) and an initial frame \( x_0 \) annotated with the bounding box \( s_0 \) of a single target object, predict bounding boxes \( s'_i \) such that the mean bounding box overlap \( s \propto \sum_i \text{IoU}(s'_i, s_i) \) is maximized, where \( s_i \) are the true bounding boxes.

Maximizing the bounding box overlap directly is however non-trivial, and often a surrogate function is used in it’s place, that is, a optimizable score is maximized and hopefully this solution will also provide a good solution for maximizing the original score.

## 2.2 Related work

This section presents related work in the field of visual tracking. The section is divided into three subsections to distinguish historical, modern and the most recent state of the art algorithms. This section tries to present everything in
2.2 Related work

chronological order of development as well as grouping topics together by the nature of the solution to the problem. A visualization of articles published in the field and how these refer to each other can be seen in figure 2.2. The graph has been calculated by looking at the references in each article, and pruning any edges between two articles if one can get from one to the other by traversing some other path. Details are described in appendix A.

2.2.1 Historical contributions

This section presents some historical developments in the area. Many algorithms rely on using the patch extracted from the bounding box in the initial frame or use some appearance model that is incrementally updated in each frame.

Image Registration and the Lukas Kanade Algorithm

The first variants of target tracking algorithms performed some kind of image registration. Image registration is the problem of aligning one image $x$ with another image $x'$, where the images are assumed to contain some common content. The problem is solved by posing a loss function $\epsilon(v)$ over some parameterized image alignment $v(u, \theta)$, where the alignment $v$ depends both on pixel position $u$ and the parameters $\theta$. The error is then integrated over all pixel positions to give the final error $\epsilon'$, and the functional dependence on the parameters $\theta$ are made explicit, as shown in equation (2.2).

$$
\epsilon(\theta) = \int_{u} \epsilon(x(u + v(u, \theta)), x'(u)) du. \quad (2.2)
$$

Examples of image alignments can be anything from translations, rotations, skews or even arbitrary warps. Some common error functions are the Sum of Squared Differences (ssd), Sum of Absolute Differences (sad), and the negative Normalized Cross Correlation (ncc) [44]. Lucas and Kanade [44] were the first to propose (at the time) an algorithm for performing the minimization in reasonable time. They proposed to perform a Taylor expansion of the error around the current estimate of the parameters $\theta$, and then solve the optimization problem iteratively.

Using the ssd error gives a particularly nice expression for updating the parameters. In the context of a target tracker where $x$ and $x'$ are the image and a template patch respectively, the method is then often referred to as the Lukas-Kanade-Tracker.

EigenTracking

Black and Jepson [10] introduced a tracking model based on a linear subspace representation of images. A linear subspace is learned from a set of images of an object. The images should be taken such that the subspace represents the different modes of the target such as pose and illumination conditions. During the tracking, an error measure between the possible image patches and the linear subspace is minimized. Given a set of training examples $A = (x_k)$, a Singular Value Decompo-
Figure 2.2: Citation graph
2.2 Related work

The matrix $U$ defines a projection operator onto the subspace spanned by the training examples. Then an error measure is defined for new samples $x$ as the norm between the sample and its projection onto the subspace defined by $U$, $\epsilon(x) = \|x - Ux\|^2$. However, this measure is not very representative for images in between the training samples (since the space generated by the natural transformations of an image is not linear), therefore a robust error metric is proposed. Using the squashing function $\rho(x, \sigma) = \frac{x^2}{\sigma^2 + x^2}$, the robust error metric $\epsilon(x) = \sum \rho(x - Ux, \sigma)$ is used. The error is minimized using gradient descent with a continuation method, which means that the smoothness parameter $\sigma$ is set high at first and then gradually made smaller.

Also the authors introduce the concept of parametric model fitting into tracking. They propose an affine warp model $x_{t+1}(u) = x_t(u + v(u, \theta))$, where $v(u, \theta) = \theta_0 + \theta_1 u$. This is combined with the robust error metric to produce the final objective function

$$\epsilon(\theta) = \sum_u \rho(x_t(u + v(u, \theta))) - x_{t+1}(u, \sigma),$$

which also can be found in literature for robust estimation of optical flow [9]. The error function is minimized using a coarse to fine optimization method using a multi resolution image pyramid.

**Incremental Visual Tracking (IVT)**

Lim et al. [42] extends the approach introduced by Black and Jepson [10] with a probabilistic interpretation together with a sequential inference model that can be updated online. They propose a factorized generative Gaussian model on the form $p(x | z) = p_\perp(x | z)p_\parallel(x | z)$ where $x$ is the observed image patch, $z = (i, j, s, a, \theta, \gamma)$ is the latent target state consisting of position $i, j$, scale $s$, aspect ratio $a$, rotation angle $\theta$ and skew $\gamma$. $p_\perp$ and $p_\parallel$ are the probability distributions generated from the distance-to-subspace $d_\perp$, and the distance-within-subspace $d_\parallel$ respectively. Letting $X_n = (x_1, \ldots, x_n)$ denote the set of observation seen so far and $U\Sigma V^T = \text{svd}(X_n)$ the singular value decomposition of the observations, the probability distributions are defined as

$$p_\perp(x | z) = \mathcal{N}(x | \mu, UU^T + \lambda I),$$
$$p_\parallel(x | z) = \mathcal{N}(x | \mu, U\Sigma^{-2}U^T),$$

where $\mathcal{N}$ is the normal distribution and $\lambda$ is the Gaussian noise variance. A Gaussian motion model is used

$$p(z_t | z_{t-1}) = \mathcal{N}(z_t | z_{t-1}, \Theta),$$

where $\Theta = \text{diag}(\sigma_i^2, \sigma_j^2, \sigma_s^2, \sigma_a^2, \sigma_\theta^2, \sigma_\gamma^2)$ is a diagonal matrix consisting of the latent state variable variances, leading to the Bayesian inference model.
\[ p(z_t \mid x_t) \propto P(x_t \mid z_t) \int p(z_t \mid z_{t-1})p(z_{t-1} \mid X_{t-1})dz_{t-1}. \]

Visual Tracking Decomposition

Kwon and Lee [40] extend the Bayesian inference model described in section 2.2.1. with a composite model for the observation and motion models. Given a set of feature extractors \( F = \{ f_k \} \), a sequence of images \( X_t = (x_t, x_{t-1}, \ldots) \), the latent state \( z_t \) at each instance is estimated, where the authors use position and scale as latent variables in their experiment. The observations are defined as the concatenation of each feature extractor applied to each image \( y_t = (f_k(x_t))_k := f(x_t) \).

The observation and motion models are then decomposed into a mixture model as

\[
p(y_t \mid x_t) = \sum_i v_i p_i(y_t \mid x_t),
\]
\[
p(z_t \mid z_{t-1}) = \sum_j w_j p_j(z_t \mid z_{t-1}),
\]

where the components \( p_i \) are tractable functions of its inputs.

The composite model is based on a mixture of features and every component uses a subset of the feature pool \( M_t = F(X_t) = \{ f_i(x_j) \} \), that is, the set of every feature extractor \( f_i \in F \) applied to every image \( x_j \in X_t \). Selecting subsets \( M_{t,i} \subset M_t \) the observation model components are defined as

\[
p_i(y_t \mid z_t) = e^{-\lambda d(y_t, M_{t,i})},
\]

and the motion model components are defined as being Gaussian around the current estimate of the latent variables

\[
p_j(z_t \mid z_{t-1}) = \mathcal{N}(z_{t-1}, \sigma_j).
\]

It is worth noting that the approach is general and does not rely on a specific representation of the image sequence.

The authors suggest the usage of the sum of diffusion distances for \( d \)

\[
d(y_t, M_{t,i}) = \sum_j d(y_t, M_{t,i}^j),
\]

and setting the feature subsets \( M_{t,i} \) to the sparse principal components of the feature pool \( M_t \). Letting \( M_t = (f(x_j)) \) be the matrix of vectorized features from
the pool of features $M_t$ up to time $t$, the sparse principal components are found by solving the convex optimization problem

$$\begin{align*}
\text{maximize} & \quad c^T M_t c - \rho \|c\|_0^2 \\
\text{subject to} & \quad \|c\|_2 = 1
\end{align*}$$

where $\rho$ is a regularization parameter, and then sets $M_{t,k}$ to the set features corresponding to the columns of $M_t$ where the components of the vector $c^j_k \neq 0$.

The posterior is approximated with a Interactive Monte Chain Monte Carlo algorithm (IMCMC).

**Fragments-based Tracking (FragTrack)**

Adam et al. [1] use a generative model based on a parts based model. Each part is represented as a histogram over the pixel values from the patch defined by the part. The tracking is performed by first defining a distance measure between image patches $x_i$ and a set of template patches $w_j$, giving rise to a distance map

$$D_{ij} = d(x_i, w_j).$$

The distance measure is taken to be some distance measure between histograms. Metrics used can be anything from simple element-wise measures such as SSD / SAD to more complicated measures such as Earth Movers Distance (EMD).

The final score is aggregated over a set of template patches $w_j$. In order to make the tracking more invariant against occlusions this is done in a robust fashion using a threshold function $\tau(x) = \min(x, t)$, where $t$ is some predefined parameter.

giving the aggregated cost at each location

$$D_i' = \sum_j \tau(D_{ij}).$$

The tracking is then performed in each frame by

$$i^* = \arg\min_i D_i'.$$

If taking the set of image patches $x_i$ consist of the set of translations of some image patch $x$, then the optimal index $i$ can be translated into optimal translation coordinates $x, y$.

The approach is also extended to tracking scale simply by adding a image $x_i$ corresponding to different scales.
A problem with adding a scale parameter is that the use of histograms introduces ambiguities into the objective function when the target is being occluded. Instead of absorbing the occluded regions as outliers in the robustness function, the tracker compensates the occlusion by changing scale, estimating occluded objects being reduced in size.

**Support Vector Tracking**

Avidan [2] introduces a discriminative model for tracking. The tracking is performed by maximization of a classifier score instead of minimization of subspace projection error as described in 2.2.1. This is motivated by the observation that real world objects are not effectively represented by linear subspace models. The proposed method is based on a Support Vector Machine (svm) that is trained offline on a classification dataset. The svm is then used for calculating a score function

\[ s(u) = \sum_i y_i \alpha_i \varphi(x(u), z_i) + b, \]

where \( z_i \) are the support vectors, \( y_i \) their sign, \( \alpha_i \) their Lagrange multipliers, \( \varphi \) a kernel function, \( b \) the bias and \( x(u) \) the input image patch extracted at translation \( u = (u_x, u_y) \).

The tracking is done by finding the translation that maximizes the score.

Setting the partial derivatives \( \frac{\partial s(u)}{\partial u} = 0 \) gives a set of, depending on the kernel function \( \varphi \) used, linear or non-linear equations. By doing a first order Taylor expansion around the current estimate of the translation the equations are solved iteratively until convergence. This is the same method applied by the Lukas Kanade feature point tracker and optical flow estimators. Worth noting is that this method produces as class dependent tracker. For further reading about svm, see Christopher [16].

**Online Boosting**

Grabner et al. [26, 27] introduce the concept of boosting into the tracking community. They update the sample weights by doing a single pass through the training samples using only weak classifiers so that a strong classifier can be updated in real time. See [22] for more information on boosting.

**Semi-Supervised Boosting**

Grabner et al. [28] take the concept of Semi-Supervised Boosting (SemiBoost) Mallapragada et al. [46] into the tracking community. The algorithm can be used for tracking by modifying the weight update scheme in an online fashion as in Grabner et al. [26, 27].

**Multiple Instance Learning**

Babenko et al. [4] introduces the concept of Multiple Instance Learning (MIL) into the tracking community. The idea behind multiple instance learning is based on an
observation that object localization is ambiguous: there is no exact target location and trying to learn from exactly annotated data introduces structural errors. In the MIL framework samples are put into bags and an objective function is defined over bag labels. Letting $X_i = \{x_{ij}\}$ denote a bag of samples $x_{ij}$ drawn from some distribution $p_x$, the bag label is defined as $y_i = \max_j y_{ij}$. This has the property that if any single instance $x_{ij}$ is positive in a bag $X_i$ the corresponding bag label $y_i$ is also positive.

The latent (unknown) samples $\{(y_{ij}, x_{ij})\}$ are replaced by the bag samples $\{(y_i, X_i)\}$ and the corresponding log-likelihood function $L = \sum_i \log p(y_{ij} | x_{ij})$ is replaced by the bag-log-likelihood $L = \sum_i \log p_i(y_i | X_i)$.

In order to optimize over the bag-log-likelihood, models for the instance class distribution $p(y_{ij} | x_{ij})$ and the bag class distribution $p(y_i | X_i)$ are needed.

Babenko et al. [4] use boosted Haar-feature classifiers $h$ for modeling the instance class distribution

$$p(y | x) = \sigma(H(x)),$$

$$H(x) = \sum_k h_k(x),$$

where $\sigma(y) = \frac{1}{1+\exp(-y)}$ is the sigmoid function and $h_k(x)$ are simple decision stumps based on Haar features.

Babenko et al. [4] used a noise-or model for the bag class distribution

$$p(y_i | X_i) = 1 - \prod_j \left(1 - p(y_i | x_{ij})\right).$$

In each frame, a set of positive examples $X^+$ and negative examples $X^-$ are extracted around the current estimate of the target position $u$ as

$$X^+ = \{x(v) | \|v - u\| < s\},$$

$$X^- = \{x(v) | s < \|v - u\| < t\}.$$

The set $X^-$ can grow quite large so a random subset is sampled in its place.

**Struck: Structured Output Tracking with Kernels**

Hare et al. [30] propose changing the general form of the objective function used in the framework for discriminative tracking. Letting $f$ be the scoring function, $u$ be the position of the target in the image and $S$ the search area of translations between two consecutive frames, traditionally the objective function takes the form

$$u^* = \arg\max_{u \in S} f(x(u)).$$
The authors change this to include a translation explicitly in the scoring function so it can be used for training

\[ u^* = \arg\max_{u \in S} f(x, u). \]

This is then cast into a SVM formulation.

**Tracking-Learning-Detection (TLD)**

Kalal et al. [35] propose a general framework for tackling the long term tracking problem, that is, to re-detect a target that has left the scene or a full occlusion has occurred. The tracker is used for short term update of the target state, while the detector is used for re-detection when tracking begins to drift or fail completely. The training is done by a heuristic bootstrapping scheme, the proposed architecture is called “PN-learning” because of the unsupervised separation of training examples into “Positive” and “Negative” sets. The separation is done by assuming that there is some kind of “expert” or “oracle” algorithm that can classify samples as positive or negative. The authors provide theoretical limits on how much the performance can be increased depending on the error rates of the experts.

**Compressive Tracking**

Zhang and hsuan Yang [59] uses a sparse measurement model for tracking. The input image is represented by an integral image which is sparsely sampled by a random projection matrix with 4 non-zero values per row, which corresponds to a generalized Haar model. The features are used in a naive Bayes classifier to give the tracking score at each location. The method is motivated by that sparse box sampling of an image corresponds to a fragment based model, which could be more robust against appearance changes due to occlusions, which holistic models struggle with.

**2.2.2 Discriminative Correlation Filters (DCF)**

This section presents the modern approach to visual object tracking. The ideas presented here lie as the basis for the research performed until the writing of this thesis.

**Minimum Output Sum Of Squared Error (MOSSE)**

Bolme et al. [11] re-introduce the correlation concept into the visual tracking community. The authors propose a highly effective method for designing linear filters that can localize arbitrary objects in real time. Given an input image \( x \) and a desired tracker output \( y \), a linear filter \( w \) is optimized by minimizing the objective \( \epsilon \)
\[ \epsilon = \| w \ast x - y \|^2 = \| \hat{w} \cdot \hat{x} - \hat{y} \|^2, \]

where \( \hat{w} \) and \( \hat{x} \) are the Discrete Fourier Transform (DFT) of \( w \) and \( x \) respectively. Taking the derivative of \( \epsilon \) gives the optimal filter

\[ \hat{w} = \frac{\hat{y}}{\hat{x}}, \]

and the authors propose to take the average of this if multiple training samples are present

\[ \bar{w}_\mu = \frac{1}{N} \sum \hat{y}_i \hat{x}_i. \]

Bolme et al. [12] later posed the optimization directly with multiple samples

\[ \epsilon = \sum_i \| w \ast x_i - y_i \|^2 = \| \hat{w} \cdot \hat{x}_i - \hat{y}_i \|^2. \]

Taking the derivative with respect to \( \hat{w} \) gives

\[
\left( \sum_{i=0}^{k} \hat{x}_i \cdot \hat{x}_i \right) \bar{w}_k = \left( \sum_{i=0}^{k} \hat{x}_i \cdot \hat{y}_i \right),
\]

giving the filter solution

\[
\bar{w}_k = \frac{\sum_{i=0}^{k} \hat{y}_i \cdot \hat{x}_i}{\sum_{i=0}^{k} \hat{x}_i \cdot \hat{x}_i} = \frac{b_k}{a_k}, \tag{2.3}
\]

where some small constant usually is added in the denominator in order to not divide by zero. However, since the filter is linear the capacity diminishes as the filter is updated. To combat this, the samples are weighted proportional to their age. In order to keep the update equations recursive, as well as keeping the statistics of the loss constant, the filter is updated recursively as
\[ a_k = \alpha a_k + (1 - \alpha) \hat{x} \cdot \bar{x}, \]
\[ b_k = \alpha b_k + (1 - \alpha) \hat{x} \cdot \bar{y}, \]
\[ \hat{w} = \frac{b_k}{a_k}, \]

which corresponds to the loss function
\[ \epsilon_k = (1 - \alpha) \| \hat{w}_k \cdot \hat{x}_k - \hat{y}_k \| + \alpha \epsilon_{k-1}. \]

Taking the expectation we have

\[ \mathbb{E}(\epsilon_k) = (1 - \alpha) \mathbb{E}\left\| \hat{w}_k \cdot \hat{x}_k - \hat{y}_k \right\| + \alpha \mathbb{E}(\epsilon_{k-1}) \]
\[ = (1 - \alpha) \mathbb{E}\left( |x_k^2| \cdot |\hat{w}_k^2| - \hat{w}_k \cdot \hat{x}_k \cdot \bar{y}_k - \bar{w}_k \cdot \bar{x}_k \cdot \hat{y}_k + |\bar{y}_k|^2 \right) + \alpha \mathbb{E}(\epsilon_{k-1}) \]
\[ = (1 - \alpha) \left( C_{x}^{z}\hat{w}_k^2 - C_{y}^{z}\bar{w}_k - C_{x}^{y}\hat{w}_k + C_{y}^{y} \right) + \alpha \mathbb{E}(\epsilon_{k-1}), \]

where we defined
\[ C_{cd...}^{ab...} = \mathbb{E}(ab...cd). \]

Evaluating the expectation for the first frame
\[ \mathbb{E}(\epsilon_0) = C_{x}^{z}\hat{w}_k^2 - C_{y}^{z}\bar{w}_k - C_{x}^{y}\hat{w}_k + C_{y}^{y}, \]
shows that when inducing over \( k \) that the expected loss is a quadratic function in the weights \( \hat{w}_k \) (that is, does not depend on the expected input/outputs over time).

The filter solution given by equation (2.3), being the Minimum Output Sum of Squared Errors (MOSSE) is considered being a baseline solution in modern visual tracking. The optimization has interesting interpretations, one is in accordance with previous work such as Babenko et al. [4], solving this linear optimization problem is equivalent to extracting shifted samples around the target and training a linear classifier on all of these. This was before considered intractable and sub-sampling the shifted samples was a solution to this, as done by Babenko et al. [4]. The key for solving the problem is using the DFT for diagonalizing the problem, that is, turning it into an element-wise optimization problem.

**Kernelized Correlation Filter (KCF)**

Henriques et al. [32] extend the DCF tracking framework by combining the effectiveness of training the tracker in the frequency domain with kernel methods,
calling it the Kernelized Correlation Filter (KCF).

In general, the ridge regression problem

$$\epsilon = \sum_i \| w^T x_i - y_i \|^2 + \lambda \| w \|^2,$$

has the solution

$$w = (X^H X + \lambda I)^{-1} X^H y,$$

where $X = (x_i)$ is the matrix created from stacking the vectorizations of training samples and $\lambda$ is the regularization parameter for the ridge regression problem.

The solution can be expressed as a combination of the training samples, known as the Representer Theorem [52].

$$w = \sum_i a_i \varphi(x_i),$$

where $a_i$ are scalar coefficients and the function $\varphi$ is chosen such that inner products in feature space can be expressed through a kernel function $\langle \varphi(x), \varphi(x') \rangle = \phi(x, x')$. The output $y$ of the kernelized filter when applied to a new input $x'$ can then be expressed as

$$y = f(x') = \langle w, \varphi(x') \rangle = \sum_i a_i \langle \varphi(x_i), \varphi(x') \rangle = \sum_i a_i \phi(x_i, x').$$

Optimizing for $a_i$ under the assumption that the matrix $\Phi_{ij} = \phi(x_i, x_j)$ is circulant, gives the solution in the Fourier domain as

$$\hat{a} = \frac{\hat{y}}{\hat{k}_{xx} + \lambda},$$

where $\hat{k}_{xx} = \mathcal{F}[\phi(x_0, x_i)]$ are the components the Fourier transform of the auto-correlation $x \ast x$ of $x$. Expressing the solution in terms of $a$ is called the solution in the dual domain.

Detection is then performed as

$$\hat{y}' = \hat{f}(x') = \hat{a} \cdot \hat{k}_{xz} = \frac{\hat{y}}{\hat{k}_{xx} + \lambda} \hat{k}_{xz},$$
for some new input $x'$. 

**Scale Estimation**

Li and Zhu [41] extend the KCF tracker with scale estimation. They form a scale pyramid on a predefined set of scales $s = (s^{-N}, s^{1-N}, \ldots, s^{N-1}, s^N)$. The scale is selected by taking the scale with the highest filter response. Danelljan et al. [18] extend the scale estimation approach by training a DCF on the scale pyramid. When optimizing the scale filter the spatial and channel dimensions are treated as a single high dimensional feature space. The different scales of the image is then treated as training samples for the optimization problem. The solution and updating rules are similar to the standard DCF equations.

**Spatial Regularized Correlation Filters (SRDCF)**

Danelljan et al. [20] tackle the problem that the learned filter fits to both the target object and the background. This can be mitigated by regularizing the spatial extent of the filter with a per pixel regularizer $m$ as

$$
\epsilon = \sum_k \alpha_k \left\| \sum_l w_l^i * x_k^i - y_k \right\|^2 + \sum_l \left\| m \cdot w_l^i \right\|^2
$$

$$
= \sum_k \alpha_k \left\| \sum_l \hat{w}_l^i \cdot \hat{x}_k^i - \hat{y}_k \right\|^2 + \frac{1}{MN} \sum_l \left\| \hat{m} \cdot \hat{w}_l^i \right\|^2,
$$

where $M$ and $N$ are the number of spatial coordinates and in each mode. The regularizer $m$ is set such that pixels close to the target center affects the objective more than pixels further away. The drawback of this method is that the effectiveness of the optimization is reduced since the problem can no longer be diagonalized due to the spatial regularization term. The authors propose an optimization scheme based on matrix factorizations that can be updated in a sequential manner. This together with enforcing $\hat{m}$ to be sparse leads to an efficient update scheme. This can be done by letting $m$ correspond to a sinusoidal, such that it equals 1 at the target and 0 at the borders of the search area.

**Staple: Sum of Template And Pixel-wise LEarners**

Bertinetto et al. [5] combine DCF with histograms to give some invariance to spatial deformations. They fit a linear model to the local histogram around the target as

$$
\epsilon = \sum_i (\langle w, f_i \rangle - y_i)^2 = \sum_j \frac{N_j(O)}{|O|} (w^j - 1)^2 + \frac{N_j(B)}{|B|} (w^j)^2,
$$

where $f_i$ are the histogram features and $y_i$ the target responses for sample $i$, and $w$ is the regression parameters, $O$ is the set of foreground pixels, $B$ is the set of background pixels, $\frac{N_j(O)}{|O|}$ and $\frac{N_j(B)}{|B|}$ is the fraction of foreground and background pixels where feature $j$ is non-zero.
The solution to the regression problem is

\[ w^j = \frac{p_j(O)}{p_j(O) + p_j(B) + \lambda}. \]

The model parameters are updated online as

\[ p_t(O) = (1 - \alpha)p_{t-1}(O) + \alpha p'_t(O), \]
\[ p_t(B) = (1 - \alpha)p_{t-1}(B) + \alpha p'_t(B), \]

where \( p'(\cdot) = N_j(\cdot)/|\cdot| \) is the estimated probability for the current frame.

**Context Aware DCF**

Mueller et al. [48] incorporates context information into the standard DCF tracking framework. The loss function is extended by regularizing the filter response for patches outside the object. The loss function for a single image is

\[ \epsilon = \| w \ast x_0 - y \|^2 + \lambda_1 \| w \|^2 + \lambda_2 \sum_{i=1}^{K} \| w \ast x_i \|^2, \]

where \( x_0 \) is the image patch covering the target and \( x_1, \ldots, x_K \) are patches covering background.

The solution in the **primal** domain is

\[ \hat{w} = \frac{\hat{x}_0 \cdot \hat{y}}{\hat{x}_0 \cdot \hat{x}_0 + \lambda_1 + \lambda_2 \sum \hat{x}_i \cdot \hat{x}_i}. \]

The sampling strategy can be anything from uniformly random to selectively by penalizing high responses that are far away spatially from the target or to give low response on other targets for multi target tracking applications.

**Target Response Adaptation**

Bibi et al. [8] target the problem that circular shifts do not correspond to actual image translations, which is the fundamental assumption behind the usage of the **Transform Trick**. This is done by letting the target response be optimized jointly with the filter by minimizing the loss

\[ \epsilon = \| w \ast x - y \|^2 + \lambda_1 \| w \|^2 + \lambda_2 \| y - y_0 \|^2, \]

where \( y_0 \) is the a priori response that needs to be designed.
Subspace Projection

Danelljan et al. [19] implements the idea introduced by Henriques et al. [32] by using color-name features $\phi(x_k)$ as the basis for the solution of the filter $w = \sum_i \alpha_i \phi(x_i)$.

They also extend the KCF solution to multiple images by posing the loss function

$$\epsilon = \sum_t \beta_t \left\| w \ast \phi(x_t) - y_t \right\| + \lambda \left\| w \right\|^2.$$

The color-names feature extractor converts each pixel in the image $x_t$ into a 11-dimensional feature vector $f_t = \phi(x_t)$. The solution in the dual domain becomes

$$\hat{a}_t = \frac{\sum_t \beta_t \tilde{y}_t \cdot \hat{f}_t}{\sum_t \beta_t \hat{f}_t \cdot (\hat{f}_t + \lambda)}.$$

The feature vector is constrained to sum to 1, $\sum_i f^i_t = 1$, so the representation is redundant. The authors further compresses the feature vector using an online PCA algorithm. Computing the principal components online gives a set of basis vectors (when concatenated can be represented by a matrix) $B_k = (b_j)$ that spans some subspace to the original vector space. The current appearance model is compressed by minimizing the auxiliary cost function $\eta^{tot}$ as

$$\eta^{tot}_k = \alpha_k \eta^{data}_k + \sum_i \alpha_i \eta_i^{smooth},$$

where

$$\eta^{data}_k = \left\| x_k - B_k B^T_k x_k \right\|^2,$$

which corresponds to the PCA on the latest image. The smoothness terms makes the subspace adapt to previous samples by penalizing the projection matrix $B_k$ if the columns do not span the same subspace as previous projection matrices $B_i, i < k$. Letting $b_{ij}$ be the j:th basis vector of the calculated subspace $B_i$ in the i:th frame, the smoothness term is calculated as

$$\eta_i^{smooth} = \sum_i \gamma_{ij} \left\| b_{ij} - B_k B^T_k b_{ij} \right\|^2,$$

where $\gamma_{ij}$ are weights for each basis vector in each subspace.
The equations is solved recursively by tracking the eigen-value decomposition of the covariance matrix of the feature vectors.

2.2 Related work

2.2.3 Deep Learning

Recently the machine learning sub-field called deep learning (DL) has shown to solve many problems in various areas. Visual tracking comes as no exception to this and deep learning has been integrated into the DCF-framework.

The most prominent application of deep learning in the context of visual tracking is the usage of convolutional networks (CNN) as a feature extractor which is then used in a DCF.

Hierarchical Convolutional Features

Ma et al. [45] introduce deep convolutional networks into the tracking community. They realize that the deep layers are more discriminative at the cost of spatial resolution, and utilize this relationship to track the target in a coarse to fine fashion, beginning the search at the deepest layer, then use the maximum response location as the center of the search area in the previous layer, and so forth.

Siamese networks (SiamDCF)

Bertinetto et al. [6] apply the concept of similarity learning using deep convolutional siamese networks. Siamese networks is the concept of learning a deep feature extractor $\varphi(\cdot | w)$ such that the similarity between a target template $x$ and an input image $z$ is calculated as

$$ f(x, z | w) = \varphi(x | w) * \varphi(z | w) + b. $$

The parameters are optimized to minimize the average of the per image logistic loss function $l(x, z, y, w)$ over some external dataset $D$

$$ l(x, z, y, w) = \sum_i \log \left( 1 + \exp \left( -y_i f(x^i, z^i | w) \right) \right), $$

$$ L(w) = \frac{1}{|D|} \sum_{x, z, y \in D} l(x, z, y, w). $$

The patch is extracted from an image $x'$ from the same sequence as the search image $x$ such that the images are at most $N$ frames apart, for some parameter $N$. $y$ is an ideal Gaussian response centered on the current target and $w$ the network weights.

Continuous Convolutional Operators for Tracking (CCOT)

Danelljan et al. [21] extended the DCF-framework to learning filters in the continuous domain. This provides sub-pixel accuracy as well as natural integration of multi-resolution feature maps into a single score map without explicit re-sampling.
Given a feature map \( \mathbf{x} \) they define an interpolated variable \( \tilde{x} \) for each channel \( l \) as

\[
\tilde{x}^l(t) = \sum_i x^{l,i} f_i(t - \frac{T}{N_l} i),
\]

where they define the interpolation kernels in terms of the cubic spline \( g(t) \) as

\[
f_i(t) = g\left(\frac{N_l}{T} (t - \frac{T}{2N_l})\right).
\]

The goal is to learn a set of continuous filters \( \tilde{w}^1, \ldots, \tilde{w}^L \) such that the continuous score map

\[
\tilde{y}'_i = \sum_l \tilde{w}^l \ast \tilde{x}_i,
\]

minimizes the error

\[
\epsilon(w) = \sum_i \alpha_i \left\| \tilde{y}'_i - \tilde{y}_i \right\|_{L^2}^2 + \sum_l \left\| \tilde{m} \ast \tilde{w}^l \right\|_{L^2}^2.
\]

Note that the error is defined in the continuous domain of square-integrable functions \( L^2 \). The score is expanded into its Fourier coefficients

\[
\hat{f}^k_l = \frac{1}{N_l} \exp\left(-i \frac{\pi}{N_l} \right) \hat{g}\left(\frac{k}{N_l}\right),
\]

\[
\hat{\tilde{x}} = \hat{x} \hat{f},
\]

\[
\hat{\tilde{y}}' = \sum_l \tilde{w}^l \hat{x} = \sum_l \hat{w} \hat{x} \hat{f}.
\]

By Parseval’s Theorem, the error function can now be expressed as a norm in the space of discrete square-summable sequences \( l^2 \):

\[
\epsilon(w) = \sum_i \alpha_i \left\| \sum_l \hat{w} \hat{x} f_i - \hat{y}_i \right\|_{l^2}^2 + \sum_l \left\| \hat{m} \ast \hat{w}^l \right\|_{l^2}^2.
\]

For practical use, the Fourier coefficients are truncated to some finite number and are then optimized online as in previous work.

During tracking the target is first localized on a grid, which corresponds to the evaluation strategies from previous work. Then a sub-pixel refinement is done by
performing gradient descent with Newton’s method. This can be done efficiently since the gradient and Hessian can be analytically derived in terms of the Fourier coefficients.

**Efficient Convolutional Operators (ECO)**

Finally, Danelljan et al. [17] incorporate the idea of joint optimization of a projection and a convolutional operator introduced by Danelljan et al. [19] into the more recent approach using deep features and continuous convolution.

Also, they reintroduce generative models into visual object tracking by the observation that the error function

$$\epsilon(w) = \sum_i \alpha_i \| \tilde{y}_i' - \tilde{y}_j \|^2_{L^2} + \sum_l \| \tilde{m}\tilde{w}' \|^2_{L^2},$$

is only an approximation of the *true* error

$$\epsilon(w) = \mathbb{E} \| \tilde{y}' - \tilde{y} \|^2_{L^2} + \sum_l \| \tilde{m}\tilde{w}' \|^2_{L^2},$$

under the assumption that $p(x, y) = \sum_i \alpha_i \delta_{x_i, y}(x, y)$. This can be improved upon by estimating the actual joint distribution $p(x, y)$. If all target responses $y = y_0$ are equal then the joint distribution can be factorized into $p(x, y) = p(x) \delta_{y_0}(y)$. The image distribution $p(x)$ is modeled as a Gaussian Mixture Model (GMM)

$$p(x) = \sum_l a_l N(x | \mu_l, I).$$

The error function is then approximated in terms if the Gaussian mixture as

$$\epsilon(w) \approx \sum_l a_l \| \tilde{\mu}_l - \tilde{y}_0 \|^2_{L^2} + \sum_l \| \tilde{m}\tilde{w}' \|^2_{L^2}.$$

**2.2.4 Tracking Using Depth Information**

So far all trackers have worked with the same data type: Single stream of RGB-images. This can be extended to include other sources of information, such as depth. Common approaches to obtaining depth information include active methods such as structured light or passive methods such as using a stereo camera.

Images with an extra feature channel where each pixel corresponds to depth will be referred to as RGBD-images.
Princeton Tracking Benchmark

One tracking benchmark with RGBD-images is the Princeton Tracking Benchmark by Song and Xiao [55]. In connection with the benchmark the authors introduces some baseline algorithms for performing general object tracking on RGBD data. The authors propose to utilize Histogram of Oriented Gradients (hog) features on both color and depth together with a SVM for discriminative tracking. They also propose to use point cloud features, which consist of calculating the following attributes quantized for a set of cubic cells around the estimated target position

- Color-names histogram
- 3D-shape
- Number of points

The features can then be tracked either in 2D using optical flow or in 3D using Iterative Closest Point (icp). Occlusion handling is suggested to be done by examining the depth histogram.

Local Depth Patterns (LDPStruck)

Awwad et al. [3] apply Local Binary Patterns (LBP) to the depth image (and depth image only), called Local Depth Patterns (LDP). They use the Struck tracker by Hare et al. [30] as baseline with LDP features as input to the SVM. Occlusion is handled by keeping track of a moving average $\mu_t$ of the depth of the target. Occlusion is flagged if the current depth $d_t$ differs from the average by more than some threshold $|d_t - \mu_t| > \tau$. The average depth is only updated if there is no occlusion.

Occlusion Aware Particle Filter (OAPF)

Meshgi et al. [47] use a probabilistic observation model in a particle filtering framework

$$p(x_t | z_t) = \prod_i p_i(x_t | z_t) \propto \prod_i \exp \left( -\lambda_i d \left( \varphi_i(x_t), w_{it} \right) \right),$$

where $d$ is some metric, $\varphi_i$ are feature extractors and $w_{it}$ are template models. They try to estimate the state $z = (i, j, s_i, s_j, c)$ where $i, j, s_i, s_j$ is the bounding box image coordinates and $c$ is a binary variable indicating occlusion.

The template models $w_i^t$ are moving averages of the extracted features $f_{it} = \varphi_i(x_t)$.

Depth Scaling Kernelized Correlation Filters (DSKCF)

Camplani et al. [14] model the depth component of the target as a 1-dimensional distribution. Initially, the depth values in the search are put into a histogram
for which the K-means algorithm together with connected components analysis is applied to segment objects using only depth values. Pixels corresponding to the cluster with smallest mean is considered to belong to the target. After the initialization step the object is tracked using the cluster mean $\mu_{\text{obj}}$ and $\sigma_{\text{obj}}$.

The authors also introduce a method for changing the image patch size during tracking by a re-sampling strategy.

Occlusion is detected if a certain percentage of the pixels corresponding to the target is occluded and the detection score is small. During occlusion the occluding object is estimated and tracked instead, while searching for the original object around the occluding object.

**3D Part-Based Sparse Tracker (3DT)**

Bibi et al. [7] use a part based point cloud model for tracking objects. They estimate a 6-dimensional target state, the pose.

The estimation is done with a particle filter, each particle is described by a cuboid containing a part of the point cloud. A 13-dimensional feature vector is extracted from the point cloud inside the cuboid corresponding to the particle, 10 color-names and 3 3D-shape features which are then modeled as a sparse combination of dictionary items.

2D-Optical flow (Horn-Schunck) is used to give a crude estimate of the target location in the next frame. This can be seen as an image plane motion model.

Occlusion is detected by monitoring the number of points in each particle. If the number of points, after compensating by the inverse distance to the camera, falls below some threshold, that particle is considered occluded. When the target is in an occluded state the motion model is set to constant.

**Bayesian Filtering for Stereo Vision**

Given detections in a stereo pair a 3-dimensional state can be estimated through some geometrical estimation method, e.g. triangulation. Given the 3-dimensional state of the object being tracked, target tracking methods such as the Kalman filter or the particle filter can be applied. The relationship between the 3-dimensional state and the 2-dimensional image plane state of the target is however non-linear, and several approaches exists on how to attack the problem. Shtark and Gurfil [53] approximate the non-linear equations by letting the noise distribution be Gaussian conditioned on the current distance to target. Then a Kalman filter is used as usual on the estimated 3-dimensional state. Sinisterra et al. [54] linearize the equations with a first order Taylor expansion around the current position of the target, which is used in an Extended Kalman Filter (EKF). Ošep et al. [50] filter the detections by back-projecting the bounding box into 3D-space, the depth coordinate is estimated through the depth image obtained from the stereo pair. The 2D-state and 3D-state are then jointly tracked using a coupled state space model.
2.3 Disparity estimation

Given a rectified image pair one can obtain an estimation of depth. This is done by first estimating the so called disparity in each pixel, which is distance between the pixel in one view and the corresponding pixel in the other view. The depth is then calculated from the disparity $d$, the focal length $f$ and baseline length $B$ as

$$D = \frac{Bf}{d}.$$ 

According to Scharstein and Szeliski [51], any method that estimates disparity between two images consists of the following steps

- Matching cost computation
- Cost aggregation
- Disparity computation / optimization
- Disparity refinement

Most methods focus on improving the matching cost computation, as that seems to be the most crucial factor in the estimation.

The most traditional method is the Block Matching algorithm (BM). The origin of this method is unclear and seems to be considered as general knowledge in the community. The algorithm consists of looping over the pixels in one view, extracting a patch around that pixel and then calculating a match cost against all other pixels in the other view.

The fact that the images are rectified simplifies the search to the same row, making the search feasible. Also, some hard constraints are often incorporated such as minimum and maximum disparity, which corresponds to limiting the looping over columns in the other view, making the search completely local.

The problem with the basic algorithm is that the local computations are not globally coherent, making the disparity map very noisy. Also the algorithm can not handle occlusions, which render some points in one view not having a corresponding point in the other. Global methods such as graph cuts have been used to address these issues [36][13].

Hirschmuller [33] introduce Semi Global Matching (SGM) as a method to compute global constraints locally, resulting in a very efficient algorithm with reasonable results.

Zbontar and LeCun [58] introduce deep learning as a method of computing similarity between image patches. The learning is cast as a classification problem of determining if one patch corresponds to another. The results outperforms other hand crafted metrics.

Chang and Chen [15] use deep learning to learn disparity estimation end to end. The architecture is quite complex, utilizing many deep learning techniques such
as spatial pyramid pooling [31] and stacked hourglass [49]. By feeding each view through a CNN in a siamese network fashion, then through the spatial pyramid pooling and stacked hourglass, they regress the disparity in each pixel.

2.4 Datasets

Traditional single target tracking datasets are the Visual Object Tracking challenge (VOT) [39], Object Tracking Benchmark (OTB) [57], Template Color 128 (TC-128), ALOV300++, UAV123. For multiple target tracking there is the Multiple Object Tracking challenge (MOT). Examples of datasets for visual tracking containing depth information are the Princeton Tracking Benchmark (PTB) [55], KITTI [25] and RGB-D People Dataset [43, 56]. PTB contains 100 sequences taken with a Microsoft Kinect RGBD camera. However only 5 of these sequences have public annotations, so the dataset is of quite limited use. RGB-D People Dataset is a small dataset containing only three sequences taken at the same time with three RGBD-cameras (Microsoft Kinect). KITTI is a big dataset containing sequences taken from a stereo rig mounted on a vehicle. The dataset contains 21 sequences from different locations, and in each sequence there are a varying number of targets annotated, in total there are 917 number of tracks in the whole dataset. The number of tracks per sequence can be seen in table 2.1.

The distribution of sequence lengths together with lower and upper percentiles are shown in figure 2.3

Some example images with annotations are shown in figure 2.4 to figure 2.11. Together with the annotated images there is data from other sensors such as IMU and Lidar. The sensor layout can be seen in figure 2.12.
Table 2.1: Number of tracks per sequences for the KITTI dataset

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Number of tracks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>98</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>41</td>
</tr>
<tr>
<td>5</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>63</td>
</tr>
<tr>
<td>8</td>
<td>28</td>
</tr>
<tr>
<td>9</td>
<td>89</td>
</tr>
<tr>
<td>10</td>
<td>28</td>
</tr>
<tr>
<td>11</td>
<td>60</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>68</td>
</tr>
<tr>
<td>14</td>
<td>17</td>
</tr>
<tr>
<td>15</td>
<td>26</td>
</tr>
<tr>
<td>16</td>
<td>28</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
</tr>
<tr>
<td>18</td>
<td>21</td>
</tr>
<tr>
<td>19</td>
<td>106</td>
</tr>
<tr>
<td>20</td>
<td>134</td>
</tr>
</tbody>
</table>
2.4 Datasets

Figure 2.3: Sequence length distribution in the KITTI tracking dataset, shown in blue. The lower 10% and upper 90% percentiles are marked in red.

Figure 2.4: First frame from a car tracking sequence
Figure 2.5: A later frame from a car tracking sequence

Figure 2.6: First frame from a bicycle tracking sequence

Figure 2.7: A later frame from a bicycle tracking sequence

Figure 2.8: First frame from a pedestrian tracking sequence
2.4 Datasets

**Figure 2.9:** First frame from a car tracking sequence where the car is initially hidden behind a bush

**Figure 2.10:** First frame from a car tracking sequence where the car is initially hidden behind another car

**Figure 2.11:** First frame from a truck tracking sequence
Figure 2.12: A schematic of the sensor layout on the car that was used when recording the data for KITTI [24].
3.1 Approach to the problem

This section explains the approach to the problem attacked by this thesis.

3.1.1 Preliminaries

As mentioned in section 1.4, a single tracker is selected as the baseline. Because of its simplicity and influence on current state of the art algorithms the MOSSE tracker is used. The results are compared with various tracker implementations from OpenCV.

3.1.2 Problem identification

Running two independent MOSSE trackers on each camera in the stereo pair from the KITTI dataset and then naively triangulate the relative world position of the tracked object gives the plot shown in figure 3.1. The triangulation error is normalized to lie between 0 and 1 for visualization purposes (the box overlap lies between 0 and 1 by definition). The plots clearly demonstrates the relationship between the error in the image plane and the triangulation error. This indicates that if one can reduce the triangulation error then the bounding box overlap should also increase.

3.1.3 Possible solutions

This section presents a set of blueprints of how a stereo vision tracker can be implemented. The baseline is of course to run the trackers independently on each view. One first approach to combining these is to apply sensor fusion methods on the detections. This combined information can then be used to guide the trackers
Figure 3.1: Plots showing the relation between the image plane bounding box overlap and the triangulated position error when running the MOSSE tracker on the KITTI dataset.
in each view. Another way is to utilize stereo vision constraints on the trackers, such as the fact that a detection in the left image should always be to the right of the detection in the right image, or that the detections should be on the same row in a rectified image pair. A third variant is to use the stereo pair to create a disparity map, which can be used to calculate depth statistics to guide the trackers, or by appending the disparity map as an extra image channel. The variants are visualized in figure 3.2. Implementations for each variant are further described below.

\[\text{(a) Independent} \quad \text{(b) Exchange of 2D-information} \quad \text{(c) Exchange of 3D-information} \quad \text{(d) 3D information based} \]

\textbf{Figure 3.2: Meta architectures for stereo vision tracker algorithms}
MOSSE (Baseline)

The baseline tracker will be a variant of the MOSSE tracker introduced in section 2.2.2. The tracker is extended to handle multi-channel images by optimizing the filter independently per channel,

\[
\hat{w}_k^l = \sum_{i=0}^{k} \hat{y}_k^l \cdot \hat{x}_i^l \sum_{i=0}^{k} \hat{x}_i^l \cdot \hat{x}_i^l,
\]

and the response is then calculated by taking the sum over channels,

\[
\hat{y}' = \sum_l \hat{w}_k^l \cdot \hat{x}_l.
\] (3.1)

The application of equation (3.1) is visualized in figure figure 3.3 and figure 3.4 when done in the spatial and in the Fourier domain respectively.

![Figure 3.3: A visual representation of the operation of the MOSSE filter.](image)

![Figure 3.4: A visual representation of the operation of the MOSSE filter in the Fourier domain.](image)

Image plane response fusion (Exchange of 2D-information)

Given responses \(y_k\) from different views, how can these be fused to give a combined response \(y^*\)? Interpreting the responses as un-normalized probabilities we can derive equations of how to do this.
Given inputs $x_k$, outputs $y_k$, the probability map $p_k$ for each view separately (independent of each other) becomes

$$p_k = \frac{y_k - \min y_k}{\sum_i (y_k^i - \min_j y_{k}^j)}.$$  

Letting $i_1 = (i_1^u, i_1^v)$ and $i_2 = (i_2^u, i_2^v)$ be the multi-indices for each view (in a stereo pair) containing the coordinates in the image plane, the joint probability model can be factored as

$$p(i_1, i_2 | x_1, x_2) = p(i_1 | i_2, x_1, x_2)p(i_2 | x_1, x_2).$$

Assuming that the output in one view is independent of the input in the other view, the expression can be simplified to

$$p(i_1 | i_2, x_1)p(i_2 | x_2) = p(i_1 | i_2) p(i_2 | x_2).$$

Marginalizing over the other view gives

$$p'_1 = p(i_1 | x_1, x_2) = \sum_{i_2} T_{12}^{i_1} p_2^{i_2}.$$

The result is that we can transfer the response in one view to the other by fixing the model for $T_{12}$. For a rectified image pair, a reasonable model is to assume a separable model in the image coordinates

$$T_{12}^{i_1 i_2} = p(i_1 | i_2) = p(i_1^u, i_1^v | i_2^u, i_2^v) = p(i_1^u | i_2^u) p(i_1^v | i_2^v).$$

Assuming that the trackers estimate position as the true position plus Gaussian noise, the estimates is also Gaussian in the $y$-direction of the image.

$$p_v(i_1 | i_2) \propto \exp \left( \frac{|i_1^v - i_2^v|^2}{2\sigma^2} \right).$$

Also, the disparity $i_1^u - i_2^u$ has a hard constraint in one direction, so the conditional probability in this direction should correspond to a step function

$$p_u(i_1 | i_2) \propto \delta(i_1^u - i_2^u \leq 0).$$
Since the conditional output probabilities \( p_u \) and \( p_v \) only depend on the differences between the output \( i_1 \) and \( i_2 \), the probability maps can be represented with kernels \( k_{u}^{i_1-i_2} = p_u(i_1 | i_2) \) and \( k_{v}^{i_1-i_2} = p_v(i_1 | i_2) \) such that the transfer operation can be efficiently computed with convolutions.

\[
(p'_1)_{i_1} = p_{i_1}^{i_2} \sum_{i_2} T_{i_2}^{i_1,i_2} p_{i_2}^{i_2} = p_{i_1}^{i_2} \sum_{i_2} p_u(i_1 | i_2) p_v(i_1 | i_2) p_{i_2}^{i_2}
\]

\[
= p_{i_1}^{i_2} \sum_{i_2} k_{u}^{i_1-i_2} k_{v}^{i_1-i_2} p_{i_2}^{i_2} = p_{i_1}^{i_2} \sum_{i_2} \sum_{i_2} k_{u}^{i_1-i_2} k_{v}^{i_1-i_2} p_{i_2}^{i_2},
\]

which is identified with a convolution of the response in the second view and the two filter kernels, followed by element-wise multiplication with the first view.

\[
p' = p_1(k_u * (k_v * p_2)).
\]

The method is abbreviated as IPRF.

**Multiple View Learning (Exchange of 2D-information)**

Another approach for utilizing the information from multiple views is to update the trackers for each camera with the result of the tracker output from the other camera. The result is that each tracker is trained on twice the amount of data. Instead of optimizing two separate DCF-equations independently

\[
\epsilon_1 = \sum_i \alpha_i \left\| w_1 * x_{1,i} - y_{1,i} \right\|^2,
\]

\[
\epsilon_2 = \sum_i \alpha_i \left\| w_2 * x_{2,i} - y_{2,i} \right\|^2,
\]

they are optimized with respect to each other as

\[
\epsilon'_1 = \sum_i \left( \alpha_i \left\| w_1 * x_{1,i} - y_{1,i} \right\|^2 + \beta_i \left\| w_1 * x_{2,i} - y_{2,i} \right\|^2 \right),
\]

\[
\epsilon'_2 = \sum_i \left( \alpha_i \left\| w_2 * x_{1,i} - y_{1,i} \right\|^2 + \beta_i \left\| w_2 * x_{2,i} - y_{2,i} \right\|^2 \right).
\]

The method is abbreviated as MVL.

**Reprojection of Triangulated Coordinates (Exchange of 3D-information)**

A simple approach to explicitly use the relation between the stereo cameras is to first create three dimensional information and then use this information to update the two dimensional state of the trackers, for example, triangulation followed by reprojection. In each frame the box center of the detections from each tracker
is triangulated to give a three dimensional point. This point is then reprojected in into each view, and the difference between the original middle point and the reprojected point is used as a correction to translate the box in each view.

**Depth as color (3D-information based)**

If per pixel depth information is available, a third approach is to append this as a fourth channel to existing RGB-data. Per pixel depth information can be created either through some active method (structural lightning) or passive method (block matching between rectified stereo images). Appending the depth data as a color channel one can apply any existing tracker algorithm to this new data.

**Multi-channel MOSSE (Independent)**

The DCF solution for multiple channels described in section 3.1.3 are naive in the sense that the DCF-equations are solved for each channel independently and then summed together at the end. Galoogahi et al. [23] show that solving the equations exactly leads to a per pixel matrix inversion. While risking being inefficient, they still show improved tracking results. However, because the DCF-loss function is recursively defined, the matrix inversion can be done recursively too, greatly improving the efficiency of the exact solution of the multi-channel DCF-equations. The caveat is that one has to sacrifice constant regularization for a regularization that diminishes over time. This is because holding the regularizer constant would make recursive formulation of the matrix inversion impossible (up to the knowledge of the author).

The original update equations (MOSSE)

\[
\begin{align*}
a_t &= \alpha a_{t-1} + (1 - \alpha)\hat{x}_t \hat{x}_t, \\
b_t &= \alpha b_{t-1} + (1 - \alpha)\hat{x}_t \cdot \hat{y}_t, \\
\hat{w}_t &= \frac{b_t}{a_t + \lambda},
\end{align*}
\]

are replaced by

\[
\begin{align*}
A_0 &= \hat{x}_0 \hat{x}_0^H + \lambda I, \\
A_t^{-1} &= \frac{1}{\alpha} \left( A_{t-1}^{-1} \hat{x}_t \hat{x}_t^H A_{t-1}^{-1} - \beta \frac{A_{t-1}^{-1} \hat{x}_t \hat{x}_t^H A_{t-1}^{-1}}{1 + \beta \hat{x}_t^H A_{t-1} \hat{x}_t} \right), \\
b_t &= \alpha_t b_{t-1} + (1 - \alpha_t)\hat{x}_t \cdot \hat{y}_t, \\
\hat{w}_t &= A_t^{-1} b_t, \\
\beta &= \frac{1 - \alpha}{\alpha}. 
\end{align*}
\]  

The derivation of equation (3.3) is given in appendix B.
3.2 Optimization

Tracking algorithms often have some tunable parameters. This can include for example a learning rate when updating some filter. Black box optimization of the parameters can be done in several ways, the two simplest being grid search and random search. For this thesis grid search was used for one dimensional parameters and random search for higher dimensional parameters. The quality of the optimization is assessed by looking at the distribution of optimal parameters when then data selection varies. By sub-sampling the dataset multiple times one can get an estimate of how often each choice of parameters is optimal. This procedure is known as the bootstrap, and the optimal parameter is given by the “bootstrap posterior” maximum. The details are outlined in algorithm 1.

Algorithm 1 Statistical Bootstrap

```python
1: procedure BootstrapOptimize(tracker, dataset, metric, numIter)
2:     results = run(tracker, dataset)
3:     scores = metric(results)
4:     histogram = Dict()
5:     iter = 0
6:     while iter < numSamples do
7:         iter += 1
8:         resampledScores = Resample(scores)
9:         optimalParam = argmax resampledScores
10:        histogram[optimalParam] += 1
11:     end while
12:     return argmax histogram
13: end procedure
```

3.3 Evaluation

In order to compare different tracker algorithms various common metrics are used. The most popular metric used in large scale tracker evaluations is the bounding box overlap [38, 55, 57]. This metric do however not provide any information when the tracker loses the target [38]. To overcome this the precision plot is commonly used. The precision plot captures information about the successfullness of the tracker at various thresholds on the required tracker performance. Metrics that outputs higher values for better performance of the evaluated algorithm are referred to as scores, while metrics that outputs lower values for better performance are referred to as losses. The precision plot can be created for any bounded metric score.

Let \( \mathcal{X} = \{X_i\} \) be a dataset consisting of annotated sequences \( X_i \). Letting \( s_{ij} = f(x_{ij}, y_{ij}) \) be the metric score of the tracker output \( y_{ij} \) in sequence \( i \) and frame \( j \), various aggregated statistics can be calculated. Given a dataset aggregation function \( A_X \) and a sequence aggregation function \( A_{X_i} \), we define a metric over the
3.3 Evaluation

whole dataset with respect to the metric \( f \)

\[
F(\mathcal{X}) = A_{\mathcal{X}} \left( A_{\mathcal{X}}(s_{ij}) \right),
\]

and similarly we define the precision with respect to the metric \( f \) and threshold \( \tau \) as

\[
P(\tau) = A_{\mathcal{X}} \left( A_{\mathcal{X}}(s_{ij}) > \tau \right). \tag{3.4}
\]

The most commonly precision metric used is the mean average precision, which is obtained by setting \( A_{\mathcal{X}}(f) = \frac{1}{|\mathcal{X}|} \sum_i f_i \) and \( A_{\mathcal{X}}(f) = \frac{1}{|\mathcal{X}|} \sum_j f_j \), or inserted into equation (3.4)

\[
P(\tau) = \frac{1}{|\mathcal{X}|} \sum_i \frac{1}{|\mathcal{X}_i|} \sum_j \left( f(x_{ij}, y_{ij}) > \tau \right),
\]

where \( |\mathcal{X}| \) is the number of sequences in the dataset and \( |\mathcal{X}_i| \) is the number of frames in sequence \( i \). From the mean average precision one can obtain plots by varying the threshold. Using the bounding box overlap as metric the resulting precision plot is commonly known as the success plot [57]. The precision plot gives wider insight into the behavior of the tracker. A scalar metric might not distinguish between trackers that are very precise for a short number of frames before losing the target or trackers that quickly lose a tight bounding box around the target but are able to maintain an acceptable output for a longer time. Scalar metrics can be created from the precision either by fixing the threshold \( \tau \) or by integration. The resulting scalar metric is called the accuracy with respect to the metric \( f \).

A note should be made about the expected average overlap measure introduced in [37]. The measure is there defined as (in the notation used above)

\[
F(\mathcal{X}) = \frac{1}{N_{hi} - N_{lo}} \sum_{k=N_{lo}}^{N_{hi}} \frac{1}{N_k} \sum_{\mathcal{X} \in \mathcal{X} : |\mathcal{X}| = k} \frac{1}{k} \sum_{x,y \in \mathcal{X}} f(x, y),
\]

where \( N_k = |\{\mathcal{X} \in \mathcal{X} : |\mathcal{X}| = k\}| \) and \( N_{lo} \) and \( N_{hi} \) are calculated by taking the first values left and right of the mode of the distribution of sequence lengths where the left and right points have approximately the same probability and the CDF covers 50% probability between the points. What this metric actually does is to compensate for the skewed distribution of sequence lengths, such that the value corresponds to the value one would have if sequences where sampled such that the distribution of sequence lengths is uniform. In this thesis we consider the distribution of sequence lengths to be a property of the dataset and not of the metric.

**Probabilistic objective for visual tracking**

The objective function used in visual tracking benchmarks is most commonly some variant of the bounding box overlap metric. Intuition tells us that the bounding box overlap is a good metric to measure the performance of a computer vision algo-
rithm, but when looking closer one begins to ask oneself, what does the bounding box overlap really represent, mathematically?

One problem with the box overlap metric is that it is hard to relate to any statistical measure. A general approach in modern machine learning is to create a statistical model for the problem, that is, to define the likelihood function for the dataset given the parameters of the tracker. One statistical model of a sequence \( \{x_k\} \) with annotated states \( \{s_k\} \), predicted states \( \{s'_k\} \) and a tracker with parameters \( \theta \) is

\[
L(\theta) = p(s'_n, \ldots, s'_1 \mid s_n, \ldots, s_1; x_n, \ldots, x_1; w_{n-1}, \ldots, w_1; \theta).
\]

Assuming that the tracker is Markovian (see equation (2.1)), the model can be factorized as

\[
L(\theta) = \prod_k p(s'_k \mid s_k, x_k, w_{k-1}; \theta).
\]

Taking the negative logarithm, one obtains a model on a standard statistical form that is more common to work with.

\[
L(\theta) = -\sum \log p(s'_k \mid s_k, x_k, w_{k-1}; \theta) = -\sum \log p(s'_k \mid s_k; \theta),
\]

where the dependence on the inputs \( x_k \) and internal tracker state \( w_k \) have been discarded since they are fixed by the dataset and by design of the tracker respectively. If one wants the loss to be less sensitive to variations in sequence length, one can take the mean instead of the sum. This corresponds to the exotic likelihood-function

\[
L(\theta) = \left( \prod_{k=1}^{n} p(s'_k \mid s_k, x_k, w_{k-1}; \theta) \right)^{\frac{1}{n}}.
\]

Posing the problem in this form could potentially bring deeper insights to tracking algorithms, as well as opening up for more rigorous approaches to the problem. The only thing that needs to be specified is the probability measure \( p(s'_k \mid s_k; \theta) \). For example, one such metric can be the bounding box overlap, which is a valid probability since it lies between 0 and 1. We refer to the resulting loss metric as the box log-likelihood.

**Testing strategies**

As noted in [38], if considering short term tracker algorithms, that is, if one expects there to be a loss of the target at some point during tracking, then measures averaged over the whole image sequence are biased estimators of the performance of the tracker during the periods when it has not lost the target yet. To compensate...
for this an evaluation scheme is proposed where the tracker is reset when it drifts completely of the ground truth target. Then, in order to not initialize the tracker in a bad state, $N_{\text{skip}}$ frames are skipped before resetting the tracker. The authors propose to set $N_{\text{skip}} = 5$. Evaluation schemes relying on resetting the tracker are be referred to as wir-tests (\textit{With Reset}), while letting the tracker run over the whole sequence no matter if it has lost the target or not are referred to as nor-tests (\textit{NO Reset}).

### 3.4 Experimental Setup

#### 3.4.1 Dataset

PTB dataset could be used for experimenting with color as depth, but the size of the dataset prevents it’s usage in a quantitative setting. The only public dataset that contains many sequences from a stereo camera is, up to the knowledge of the author, the KITTI dataset.

The dataset is prepared by performing some pre-processing steps.
- Separate the annotations from the original dataset into separate tracks.
- Calculate bounding boxes from the point cloud annotations by reprojection into each camera.
- Calculate disparity maps using block matching (can be exchanged with some other of algorithm).
- Remove sequences of length below the 10:th percentile or above the 90:th percentile of the dataset.

The disparity images are created by using the classical block matching algorithm.

#### 3.4.2 Evaluation

Each tracker is evaluated by first optimizing the parameters using the procedure described in algorithm 1 or by direct optimization of the objective function evaluated over the whole dataset for a selected set of parameters (on a grid or uniformly random). Then the trackers are run on the whole dataset and the outputs are saved for later analysis. Using the output from each tracker various statistics are calculated, such as the mean average box overlaps, success plots and box log-likelihoods. Also, the distribution of each score/loss is analyzed and visualized by looking at the percentiles of the distribution of the per frame metrics.
The results from the parameter optimization are visualized through a distribution plot, which is a plot where the percentiles of the score/loss is visualized using a color-map. Here, the colors go from red to yellow to green and blue as the percentile move from the median to the tails of the distribution. The results are created for $A_X \in \{\text{Mean, Median}\}$ and for $A_X \in \{\text{Mean, Sum}\}$ as described in section 3.3. The results from the optimization of the various trackers when running the nor-test are shown in figure C.1 to figure C.24. The respective plots when running the wir-test are shown in figure C.25 to figure C.48. The histograms obtained from calculating the optimal parameters for resampled sets of sequences are given in figure C.49 to figure C.60 for the nor-tests and in figure C.61 to figure C.72 for the wir-tests respectively. For visualization purposes, plots are not given for trackers with more than one parameter. Tables with the optimized parameters for each tracker and evaluation type are given in table D.1 to table D.26.

The total number of failures for each tracker when running the wir test is shown in figure 4.1.

nor success plots for each tracker are given in figure 4.2 and figure 4.3 when taking the optimal parameters from direct minimization of the objective and maximization of the histogram bootstrap respectively. The corresponding plots for the wir tests are given in figure 4.4 and figure 4.3. The baseline implementation is compared to various trackers implemented in OpenCV in figure 4.6.

In the plots, MultiChannel refers to the method that solves the MOSSE equations exactly, while MultiChannelOpt refers to the recursive formulation described in section 3.1.3.
Figure 4.1: Number of failures in the dataset per parameter value.
Figure 4.2: Precision plot using the optimized parameters from direct optimization when running the NOR test
Figure 4.3: Precision plot using the optimized parameters from the histograms when running the NOR test.
Figure 4.4: Precision plot using the optimized parameters from direct optimization when running the WIR test.
Figure 4.5: Precision plot using the optimized parameters from the histograms when running the WIR test.
Figure 4.6: Success plots from the NOR test for OpenCV trackers and the baseline implementation of MOSSE.
5.1 Analysis of the results

As can be seen in figures 4.2 to 4.5, none of the proposed extensions seems to improve upon the baseline. As an exception to this is Reprojection that greatly improves on the WIR tests. This is however due to a lot higher number of failures, as can be seen in figure 4.1.

Looking at the distribution plots in figure C.1 to figure C.48 we can see that none of the objectives are sensitive to the parameters, except for Reprojection tracker where there seems to be a consistent peak in the objective, as can be seen in figure C.9 to figure C.12 and figure C.33 to figure C.36.

The histogram plots shown in figure C.49 to figure C.72 strengthen the observation that the objectives are not sensitive to the parameters. Resampling the dataset many times and picking the best parameter in each run, the optimal parameter seems to be quite random for many of the trackers. Exception to this can be found for some combinations of evaluation strategy, tracker and metric, as can be seen in figure C.49, figure C.50 and figure C.62 for Reprojection, figure C.68 for RGBD. Figure C.58, figure C.57 and figure C.70 for MultiChannel, figure C.72 for MultiChannelOpt. It is unclear why it occurs for these combinations of evaluation strategy, tracker and metric and not for other combinations.

One interesting observation is that the distributions are all heavy tailed against the lower score/higher loss regions. This is most evident in the plots for the WIR-tests for the log-likelihood loss, where the mean is almost completely dominated by outliers (see figure C.25 to figure C.47).

Also, as can be seen in figure C.71, the optimal decay for MultiChannelOpt is
1 or close to 1, which indicates that the regularization is more important than the learning rate (since setting the decay $\alpha = 1$ is the only way to not make the regularization vanish).

Another interesting observation is that the relative performance between the trackers is inverted between the NOR and WIR testing strategies. This contradicts the design principle behind the WIR-test which is to compensate for evaluation bias, not the relative ordering between the results.

Finally, the results for multiple channels are not consistent with previous results shown by Galoogahi et al. [23]. This might be because not all the parameters are optimized.

### 5.1.1 OpenCV

The plot shown in figure 4.6 clearly indicates that the tracker implementations in OpenCV differ from the one in this thesis. It is unclear why this is the case, but can be because of an erroneous implementation of MOSSE in OpenCV. This makes the comparison of the methods tested in this thesis against other methods unfair, so the results have to be discarded in the analysis.

### 5.2 Discussion

#### 5.2.1 Analysis of the proposed methods

Possible reasons why the stereo vision extensions of MOSSE do not improve the results are given below

- The assumptions done in the model are not valid
- The results were not obtained with optimal parameters for the corresponding method
- The depth information contained too many errors
- The annotations are not reasonable

Further discussion about each method is given below.

**IPRF**

In the case of fusing image plane responses (after reinterpreting them as un-normalized probabilities), a conditional independence assumption is made to simplify the calculations. This turned the probability transfer into an element-wise multiplication between the probability maps in each view, after filtering the other view with two simple, separable filter kernels. The kernels are low pass (the frequency spectrum of a Gaussian is a Gaussian and the spectrum of a block filter is a sinc), and this might be one reason why the method does not improve the performance. The motivation for this is that low pass filtering the response blurs the maximum of the filter response, effectively decreasing the responsiveness of the tracker, which results in a higher drift.
Also, the model assumes that the trackers always provide an estimate of the target position plus Gaussian noise. If the tracker drifts, this is no longer true and the fusion will probably just amplify the errors.

**MVL**

Incorporating information from one view into the other through modifying the objective function decreases the overall performance. Looking at the success plot one can clearly observe how the performance decreases rapidly with higher thresholds. The conclusion from this is that the information gained from the one view is more noisy than informative for the other view. The recursive formulation in equation (3.2) does not expose any parameter that controls how much information that is incorporated from the one view into the other, just how fast the filter should learn from the other view. Exposing such a parameter would allow the optimization to set that parameter to zero, effectively turning the method equivalent to the baseline implementation. Therefor the optimization results are more sparse when compared to the other trackers. However, since the objectives have shown to be extremely flat it is quite unlikely that this is the reason why the method perform so much worse than the other methods.

**Reprojection**

Triangulating the box midpoint followed by reprojection clearly decrease the performance. This can be explained by the fact that the middle point of the box in each view are not necessarily corresponding points (except for the first frame). As the trackers drift, the correspondence decreases fast as well. Triangulation of non-corresponding points is meaningless, and reprojecting meaningless points gives erroneous updates of the trackers in each view. A better approach would instead be to use a multiple parts model where multiple points are tracked and matched between the views.

**Disparity based**

Using disparity as a 4:th channel seems to increase the performance slightly for the optimized parameters when running the WIR test but not on the NOR test, when compared to using no disparity. One cause that the performance is not increased might be because the disparity channel is created using a classical block matching algorithm. The information in this channel might contain too many errors when used in conjunction with the MOSSE tracker.

**Multi-Channel MOSSE**

Solving the multi-channel MOSSE equations using matrix inversion decreases the performance both when done exactly and approximately. The conclusion from this is that the channels are almost independent of each other, so that solving the equations independently for each channel and followed by summation is almost equivalent to exact matrix inversion. Also, since other parameters such as the
regularization $\lambda$ is not optimized so that the results might be for sub-optimal parameters.

**KITTI dataset**

Looking at some sample images from the dataset in figure 2.4 to figure 2.11 one can see how the object to track is hidden in the first frame. It is unreasonable to expect that a visual tracker should be able to understand that it is the hidden object that should be tracked, and not the occluding one. This makes all the trackers automatically perform a lot worse than they would have with reasonable annotations, the conclusion is that the results are highly biased towards the negative. This can only be mitigated through the cumbersome work of re-labelling the whole dataset.

**5.2.2 Critique against the method**

The code for evaluating different methods is written in Python. Common benchmark frameworks such as the VOT toolkit [39] was not used since this requires a proprietary Matlab [34] license to run. Therefore, if this project was to be done again, it would sacrifice standalone software for convenience, and use existing frameworks for evaluating the trackers. Although this would make the software have non-free dependencies, it would reduce development time while also making the results more comparable.
6 Conclusion and future work

6.1 Summary

The problem formulation (chapter 1) of this thesis is re-stated below

- What methods exist for obtaining depth information from stereo images?
- How can this depth information be incorporated into visual tracking algorithms?
- How do the results change as the parameters to each tracker vary?

In chapter 3 four abstract approaches are identified (Independent, Exchange of 2D-information, Exchange of 3D-information and 3D-information based) which constitutes the answer to the first question. Five distinct implementations are proposed (Baseline, IPRF, MVL, Reprojection and RGBD) which provides the answer to the second question. Finally, the optimization results shows that the none objective functions are not sensitive to the exposed parameters under the joint selection of models and the selected dataset KITTI. This is shown by a statistical bootstrap procedure, where looking at the histogram over the optimized parameters as the samples from the dataset varies reveals the randomness in the optimization. This observation answers the final question.

6.2 Contributions

The contributions of this thesis is summarized below.

- Tested tracking on video sequences taken with stereo camera
- Used optimization of parameters and showed problems related to that
• Applied the matrix inversion lemma to efficiently solve the multi channel MOSSE equations approximately

6.3 Future work

During the writing of this thesis many thoughts regarding the problem of visual tracking have passed by. Due to time limitations or because many of these thoughts are out of scope of this thesis, they have not been followed up but instead saved for the future to tackle. Observe that many of the ideas in this section are at an early stage, and many comes without references or any deeper motivation.

6.3.1 Optimization

In this thesis, the optimization was never done on more than two parameters. If a tracker exposes more parameters, then grid/random search quickly becomes unfeasible. More intelligent search methods such as Bayesian optimization remains to be tested.

6.3.2 Disparity

Recent state of the art algorithms for creating disparity images show impressive results when compared to the classical algorithms used in this thesis. It still remains to test if using higher quality disparities obtained from these methods increases the performance when compared to using plain RGB images.

6.3.3 Capacity of linear and non-linear tracker models

Linear DCF models all share the property that the accumulated state $w_k$ is approximately a linear combination of the inputs. If the target undergoes too many non-linear transformations, the capacity of the model will be exhausted and the target is no longer ensured to be tracked correctly. Non-linear pre-processing such as CNN resolves this by making the tracker invariant to many transformations, increasing the time it takes for the final layer DCF to reach full capacity. We also know from research on CNN that they extract semantically meaningful features, which increases the discriminativity of the DCF, that is, the filter can distinguish between target and non-target better than using raw input features. One question is, how much of the performance gain in CNN-based DCF-trackers are due to better features and how much is due to higher model capacity?
Appendix
A.1 Creating the citation graph
Algorithm 2 Citation graph algorithm

1: procedure GETDESCENDANTS(node)
2:     descendants = Set()
3:     for child in Children(node) do
4:         descendants.add(child)
5:         descendants.add(GETDESCENDANTS(child))
6:     end for
7:     return descendants
8: end procedure

9: procedure SIMPLIFYTOPOLOGY(graph)
10:     for node in graph do
11:         descendants = Set()
12:         for child in Children(node) do
13:             descendants.add(GETDESCENDANTS(child))
14:         end for
15:         children(node) = children(node) - descendants
16:     end for
17:     return graph
18: end procedure

20: procedure CREATECITEGRAPH(bibliography)
21:     graph = Graph()
22:     for article in bibliography do
23:         if article not in graph then
24:             graph.node(article)
25:         end if
26:         for citation in Citations(article) do
27:             if citation not in graph then
28:                 graph.node(citation)
29:             end if
30:             graph.edge(article, citation)
31:         end for
32:     end for
33:     return SIMPLIFYTOPOLOGY(graph)
34: end procedure
B.1 Multi-channel MOSSE

Below we derive the recursive update scheme of the per pixel matrix inverses. Starting with the original loss function

\[
\epsilon = \sum_t \alpha_t \left\| \sum_k w^k \cdot x^k_t - y_t \right\|^2 = \sum_t \alpha_t \left\| \sum_k \hat{w}^k \hat{x}^k_t - \hat{y}_t \right\|^2
\]

and taking the derivative with respect to the weights for channel \( k \), \( w^k \)

\[
\frac{\partial \epsilon}{\partial w^k} \propto \sum_t \alpha_t \left( \hat{x}_t^l \sum_k \hat{x}^k_t \cdot \hat{w}^k - \hat{x}_t^l \cdot \hat{y}_t \right)
\]

Setting \( \frac{\partial \epsilon}{\partial w^k} \) to zero yields

\[
\sum_t \alpha_t \hat{x}_t^l \sum_k \hat{x}^k_t \cdot \hat{w}^k = \sum_t \alpha_t \hat{x}_t^l \cdot \hat{y}_t
\]

Putting it into vector form over channels gives that

\[
\sum_t \alpha_t \hat{x}_t \hat{x}_t^H \hat{w} = \sum_t \alpha_t \hat{x}_t \cdot \hat{y}_t
\]
Setting $A_T = \sum_t^T \alpha_t \hat{x}_t \hat{x}_t^H$ and $b_T = \sum_t^T \alpha_t \hat{x}_t \cdot \bar{y}_t$ we can rewrite the equation as

\[
\left(\alpha A_{T-1} + (1 - \alpha) \hat{x}_T \hat{x}_T^H\right)\bar{w} = \alpha b_{T-1} + (1 - \alpha) \hat{x}_T \cdot \bar{y}_T
\]

\[
\alpha \left( A_{T-1} + \frac{1 - \alpha}{\alpha} \hat{x}_T \hat{x}_T^H \right)\bar{w} = \alpha b_{T-1} + (1 - \alpha) \hat{x}_T \cdot \bar{y}_T
\]

Using the matrix inversion lemma [29] (per pixel)

\[
(A + bc^H)^{-1} = A^{-1} - \frac{A^{-1}bc^H A^{-1}}{1 + c^H Ab}
\]

we can efficiently update the filter as

\[
A_0 = \hat{x}_0 \hat{x}_0^H + \lambda I
\]

\[
A_T^{-1} = \frac{1}{\alpha} \left( A_{T-1}^{-1} - \beta \frac{A_{T-1}^{-1} \hat{x}_T \hat{x}_T^H A_{T-1}^{-1}}{1 + \beta \hat{x}_T^H A_{T-1}^{-1} \hat{x}_T} \right)
\]

\[
b_T = \alpha b_{T-1} + (1 - \alpha) \hat{x}_T \cdot \bar{y}_T
\]

\[
\bar{w}_T = A_T^{-1} b_T
\]

\[
\beta = \frac{1 - \alpha}{\alpha}
\]

where the matrix multiplications are performed per pixel.
Figure C.49: Histogram plots for the box-log-likelihood metric for tracker Baseline when running the NOR test.
Figure C.1: Plot of the mean box log likelihood for various parameter values for tracker Baseline when running the NOR test.

Figure C.2: Plot of the mean box overlap for various parameter values for tracker Baseline when running the NOR test.
**Figure C.3:** Plot of the sum box log likelihood for various parameter values for tracker Baseline when running the NOR test.

**Figure C.4:** Plot of the sum box overlap for various parameter values for tracker Baseline when running the NOR test.
Figure C.5: Plot of the mean box log likelihood for various parameter values for tracker IPRF when running the NOR test.

Figure C.6: Plot of the mean box overlap for various parameter values for tracker IPRF when running the NOR test.
Figure C.7: Plot of the sum box log likelihood for various parameter values for tracker IPRF when running the NOR test.

Figure C.8: Plot of the sum box overlap for various parameter values for tracker IPRF when running the NOR test.
Figure C.9: Plot of the mean box log likelihood for various parameter values for tracker Reprojection when running the NOR test.

Figure C.10: Plot of the mean box overlap for various parameter values for tracker Reprojection when running the NOR test.
Figure C.11: Plot of the sum box log likelihood for various parameter values for tracker Reprojection when running the NOR test.

Figure C.12: Plot of the sum box overlap for various parameter values for tracker Reprojection when running the NOR test.
Figure C.13: Plot of the mean box log likelihood for various parameter values for tracker RGBD when running the NOR test.

Figure C.14: Plot of the mean box overlap for various parameter values for tracker RGBD when running the NOR test.
**Figure C.15:** Plot of the sum box log likelihood for various parameter values for tracker RGBD when running the NOR test.

**Figure C.16:** Plot of the sum box overlap for various parameter values for tracker RGBD when running the NOR test.
Figure C.17: Plot of the mean box log likelihood for various parameter values for tracker MultiChannel when running the NOR test.

Figure C.18: Plot of the mean box overlap for various parameter values for tracker MultiChannel when running the NOR test.
Figure C.19: Plot of the sum box log likelihood for various parameter values for tracker MultiChannel when running the NOR test.

Figure C.20: Plot of the sum box overlap for various parameter values for tracker MultiChannel when running the NOR test.
Figure C.21: Plot of the mean box log likelihood for various parameter values for tracker MultiChannelOpt when running the NOR test.

Figure C.22: Plot of the mean box overlap for various parameter values for tracker MultiChannelOpt when running the NOR test.
Figure C.23: Plot of the sum box log likelihood for various parameter values for tracker MultiChannelOpt when running the NOR test.

Figure C.24: Plot of the sum box overlap for various parameter values for tracker MultiChannelOpt when running the NOR test.
Figure C.25: Plot of the mean box log likelihood for various parameter values for tracker Baseline when running the WIR test.

Figure C.26: Plot of the mean box overlap for various parameter values for tracker Baseline when running the WIR test.
Figure C.27: Plot of the sum box log likelihood for various parameter values for tracker Baseline when running the WIR test.

Figure C.28: Plot of the sum box overlap for various parameter values for tracker Baseline when running the WIR test.
Figure C.29: Plot of the mean box log likelihood for various parameter values for tracker IPRF when running the WIR test.

Figure C.30: Plot of the mean box overlap for various parameter values for tracker IPRF when running the WIR test.
Figure C.31: Plot of the sum box log likelihood for various parameter values for tracker IPRF when running the WIR test.

Figure C.32: Plot of the sum box overlap for various parameter values for tracker IPRF when running the WIR test.
Figure C.33: Plot of the mean box log likelihood for various parameter values for tracker Reprojection when running the WIR test.

Figure C.34: Plot of the mean box overlap for various parameter values for tracker Reprojection when running the WIR test.
**Figure C.35:** Plot of the sum box log likelihood for various parameter values for tracker Reprojection when running the WIR test.

**Figure C.36:** Plot of the sum box overlap for various parameter values for tracker Reprojection when running the WIR test.
Figure C.37: Plot of the mean box log likelihood for various parameter values for tracker RGBD when running the WIR test.

Figure C.38: Plot of the mean box overlap for various parameter values for tracker RGBD when running the WIR test.
Figure C.39: Plot of the sum box log likelihood for various parameter values for tracker RGBD when running the WIR test.

Figure C.40: Plot of the sum box overlap for various parameter values for tracker RGBD when running the WIR test.
Figure C.41: Plot of the mean box log likelihood for various parameter values for tracker MultiChannel when running the WIR test.

Figure C.42: Plot of the mean box overlap for various parameter values for tracker MultiChannel when running the WIR test.
Figure C.43: Plot of the sum box log likelihood for various parameter values for tracker MultiChannel when running the WIR test.

Figure C.44: Plot of the sum box overlap for various parameter values for tracker MultiChannel when running the WIR test.
Figure C.45: Plot of the mean box log likelihood for various parameter values for tracker MultiChannelOpt when running the WIR test.

Figure C.46: Plot of the mean box overlap for various parameter values for tracker MultiChannelOpt when running the WIR test.
Figure C.47: Plot of the sum box log likelihood for various parameter values for tracker MultiChannelOpt when running the WIR test.

Figure C.48: Plot of the sum box overlap for various parameter values for tracker MultiChannelOpt when running the WIR test.
Figure C.50: Histogram plots for the box-overlap metric for tracker Baseline when running the NOR test.
Figure C.51: Histogram plots for the box-log-likelihood metric for tracker IPRF when running the NOR test.
Figure C.52: Histogram plots for the box-overlap metric for tracker IPRF when running the NOR test.
Figure C.53: Histogram plots for the box-log-likelihood metric for tracker Reprojection when running the NOR test.
Figure C.54: Histogram plots for the box-overlap metric for tracker Reprojection when running the NOR test.
Figure C.55: Histogram plots for the box-log-likelihood metric for tracker RGBD when running the NOR test.
Figure C.56: Histogram plots for the box-overlap metric for tracker RGBD when running the NOR test.
Figure C.57: Histogram plots for the box-log-likelihood metric for tracker MultiChannel when running the NOR test.
Figure C.58: Histogram plots for the box-overlap metric for tracker Multi-Channel when running the NOR test.
Figure C.59: Histogram plots for the box-log-likelihood metric for tracker MultiChannelOpt when running the NOR test.
Figure C.60: Histogram plots for the box-overlap metric for tracker Multi-ChannelOpt when running the NOR test.
Figure C.61: Histogram plots for the box-log-likelihood metric for tracker Baseline when running the WIR test.
Figure C.62: Histogram plots for the box-overlap metric for tracker Baseline when running the WIR test.
Figure C.63: Histogram plots for the box-log-likelihood metric for tracker IPRF when running the WIR test.
Figure C.64: Histogram plots for the box-overlap metric for tracker IPRF when running the WIR test.
Figure C.65: Histogram plots for the box-log-likelihood metric for tracker Reprojection when running the WIR test.
Figure C.66: Histogram plots for the box-overlap metric for tracker Reprojection when running the WIR test.
Figure C.67: Histogram plots for the box-log-likelihood metric for tracker RGBD when running the WIR test.
Figure C.68: Histogram plots for the box-overlap metric for tracker RGBD when running the WIR test.
Figure C.69: Histogram plots for the box-log-likelihood metric for tracker MultiChannel when running the WIR test.
Figure C.70: Histogram plots for the box-overlap metric for tracker Multi-Channel when running the WIR test.
Figure C.71: Histogram plots for the box-log-likelihood metric for tracker MultiChannelOpt when running the WIR test.
Figure C.72: Histogram plots for the box-overlap metric for tracker Multi-ChannelOpt when running the WIR test.
Table D.1: Optimal parameters for tracker Baseline for each objective function when running the NOR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>24.96</td>
<td>0.68</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.28</td>
<td>0.81</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>1071.86</td>
<td>0.72</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>10.23</td>
<td>0.81</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>17.29</td>
<td>0.72</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.29</td>
<td>0.75</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>593.77</td>
<td>0.72</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>8.81</td>
<td>0.79</td>
</tr>
</tbody>
</table>
Table D.2: Optimal parameters for tracker IPRF for each objective function when running the NOR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>27.90</td>
<td>0.50</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.26</td>
<td>0.68</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>1175.06</td>
<td>0.63</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>9.40</td>
<td>0.68</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>22.63</td>
<td>0.76</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.27</td>
<td>0.68</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>747.77</td>
<td>0.77</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>8.24</td>
<td>0.64</td>
</tr>
</tbody>
</table>

Table D.3: Optimal parameters for tracker Reprojection for each objective function when running the NOR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>45.86</td>
<td>0.98</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.20</td>
<td>0.97</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>1912.48</td>
<td>0.99</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>6.92</td>
<td>0.99</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>49.17</td>
<td>0.97</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.18</td>
<td>0.96</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>1420.33</td>
<td>0.99</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>5.88</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Table D.4: Optimal parameters for tracker MVL for each objective function when running the NOR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>alpha</th>
<th>beta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>30.27</td>
<td>0.07</td>
<td>0.85</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.20</td>
<td>0.28</td>
<td>0.77</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>1273.35</td>
<td>0.32</td>
<td>0.88</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>7.29</td>
<td>0.74</td>
<td>0.92</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>26.74</td>
<td>0.41</td>
<td>0.77</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.18</td>
<td>0.15</td>
<td>0.48</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>825.93</td>
<td>0.43</td>
<td>0.04</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>5.53</td>
<td>0.75</td>
<td>0.06</td>
</tr>
</tbody>
</table>
**Table D.5:** Optimal parameters for tracker RGBD for each objective function when running the NOR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>27.08</td>
<td>0.43</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.27</td>
<td>0.81</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>1156.99</td>
<td>0.43</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>9.91</td>
<td>0.81</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>22.10</td>
<td>0.62</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.27</td>
<td>0.81</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>741.86</td>
<td>0.53</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>8.48</td>
<td>0.59</td>
</tr>
</tbody>
</table>

**Table D.6:** Optimal parameters for tracker MultiChannel for each objective function when running the NOR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>25.72</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.26</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>1126.42</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>9.66</td>
<td>0.98</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>20.75</td>
<td>0.00</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.28</td>
<td>0.00</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>610.79</td>
<td>0.00</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>8.50</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**Table D.7:** Optimal parameters for tracker MultiChannelOpt for each objective function when running the NOR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>27.43</td>
<td>0.82</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.26</td>
<td>0.98</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>1164.09</td>
<td>0.91</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>9.66</td>
<td>0.98</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>21.68</td>
<td>0.96</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.26</td>
<td>0.98</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>750.54</td>
<td>0.98</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>8.00</td>
<td>0.95</td>
</tr>
</tbody>
</table>
**Table D.8:** Optimal parameters for tracker Baseline for each objective function when running the WIR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>5.46</td>
<td>0.75</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.40</td>
<td>0.84</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>195.40</td>
<td>0.75</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>13.54</td>
<td>0.88</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>1.62</td>
<td>0.75</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.39</td>
<td>0.89</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>66.00</td>
<td>0.78</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>11.05</td>
<td>0.80</td>
</tr>
</tbody>
</table>

**Table D.9:** Optimal parameters for tracker IPRF for each objective function when running the WIR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>9.23</td>
<td>0.52</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.36</td>
<td>0.86</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>292.14</td>
<td>0.51</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>12.26</td>
<td>0.71</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>1.79</td>
<td>0.53</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.37</td>
<td>0.87</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>84.00</td>
<td>0.53</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>10.20</td>
<td>0.39</td>
</tr>
</tbody>
</table>

**Table D.10:** Optimal parameters for tracker Reprojection for each objective function when running the WIR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>3.63</td>
<td>0.16</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.57</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>30.37</td>
<td>0.07</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>7.64</td>
<td>0.97</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>0.80</td>
<td>0.00</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.59</td>
<td>0.01</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>6.04</td>
<td>0.00</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>6.52</td>
<td>0.97</td>
</tr>
</tbody>
</table>
**Table D.11:** Optimal parameters for tracker MVL for each objective function when running the WIR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Obective value</th>
<th>alpha</th>
<th>beta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>6.52</td>
<td>0.90</td>
<td>0.07</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.31</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>197.06</td>
<td>0.99</td>
<td>0.17</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>10.15</td>
<td>0.53</td>
<td>0.51</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>2.23</td>
<td>0.70</td>
<td>0.29</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.31</td>
<td>0.99</td>
<td>0.17</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>90.46</td>
<td>0.70</td>
<td>0.29</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>8.23</td>
<td>0.46</td>
<td>0.48</td>
</tr>
</tbody>
</table>

**Table D.12:** Optimal parameters for tracker RGBD for each objective function when running the WIR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Obective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>5.57</td>
<td>0.78</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.41</td>
<td>0.91</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>190.31</td>
<td>0.78</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>13.73</td>
<td>0.91</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>1.62</td>
<td>0.81</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.40</td>
<td>0.92</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>66.44</td>
<td>0.82</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>11.24</td>
<td>0.91</td>
</tr>
</tbody>
</table>

**Table D.13:** Optimal parameters for tracker MultiChannel for each objective function when running the WIR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Obective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>6.18</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.39</td>
<td>1.00</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>216.81</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>13.10</td>
<td>0.81</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>1.72</td>
<td>0.73</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.38</td>
<td>0.98</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>74.86</td>
<td>0.73</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>10.77</td>
<td>0.79</td>
</tr>
</tbody>
</table>
Table D.14: Optimal parameters for tracker MultiChannelOpt for each objective function when running the WIR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>6.60</td>
<td>0.91</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.40</td>
<td>1.00</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>229.51</td>
<td>0.91</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>13.21</td>
<td>0.98</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>1.72</td>
<td>0.91</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.39</td>
<td>1.00</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>73.78</td>
<td>0.91</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>10.81</td>
<td>0.98</td>
</tr>
</tbody>
</table>

Table D.15: Optimal parameters for tracker Baseline for each objective function when running the NOR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>24.56</td>
<td>0.68</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.27</td>
<td>0.81</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>1160.34</td>
<td>0.68</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>10.39</td>
<td>0.86</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>19.90</td>
<td>0.72</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.29</td>
<td>0.75</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>547.10</td>
<td>0.72</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>8.90</td>
<td>0.76</td>
</tr>
</tbody>
</table>

Table D.16: Optimal parameters for tracker IPRF for each objective function when running the NOR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>27.97</td>
<td>0.50</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.25</td>
<td>0.68</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>1174.72</td>
<td>0.63</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>9.46</td>
<td>0.68</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>22.21</td>
<td>0.76</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.27</td>
<td>0.68</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>696.97</td>
<td>0.77</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>8.85</td>
<td>0.62</td>
</tr>
</tbody>
</table>
Table D.17: Optimal parameters for tracker Reprojection for each objective function when running the NOR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Obective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>45.25</td>
<td>0.97</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.19</td>
<td>0.97</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>1943.17</td>
<td>0.99</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>7.07</td>
<td>0.99</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>48.57</td>
<td>0.97</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.17</td>
<td>0.97</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>1480.20</td>
<td>0.99</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>6.02</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Table D.18: Optimal parameters for tracker RGBD for each objective function when running the NOR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Obective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>27.91</td>
<td>0.16</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.26</td>
<td>0.81</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>1167.37</td>
<td>0.43</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>10.22</td>
<td>0.81</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>22.79</td>
<td>0.54</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.27</td>
<td>0.81</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>816.73</td>
<td>0.53</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>8.11</td>
<td>0.66</td>
</tr>
</tbody>
</table>

Table D.19: Optimal parameters for tracker MultiChannel for each objective function when running the NOR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Obective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>25.18</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.27</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>1091.68</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>9.58</td>
<td>0.98</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>21.50</td>
<td>0.00</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.27</td>
<td>0.00</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>583.68</td>
<td>0.00</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>8.33</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Table D.20: Optimal parameters for tracker MultiChannelOpt for each objective function when running the NOR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>25.39</td>
<td>0.82</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.26</td>
<td>0.98</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>1196.28</td>
<td>0.91</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>9.91</td>
<td>0.98</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>23.04</td>
<td>0.96</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.26</td>
<td>0.98</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>738.97</td>
<td>0.98</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>8.14</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Table D.21: Optimal parameters for tracker Baseline for each objective function when running the WIR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>5.30</td>
<td>0.75</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.40</td>
<td>0.84</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>201.17</td>
<td>0.75</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>13.72</td>
<td>0.88</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>1.63</td>
<td>0.75</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.39</td>
<td>0.89</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>68.63</td>
<td>0.86</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>11.12</td>
<td>0.81</td>
</tr>
</tbody>
</table>

Table D.22: Optimal parameters for tracker IPRF for each objective function when running the WIR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>8.69</td>
<td>0.59</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.36</td>
<td>0.86</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>293.80</td>
<td>0.47</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>12.31</td>
<td>0.71</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>1.73</td>
<td>0.53</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.37</td>
<td>0.87</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>100.18</td>
<td>0.53</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>10.24</td>
<td>0.39</td>
</tr>
</tbody>
</table>
Table D.23: Optimal parameters for tracker Reprojection for each objective function when running the WIR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>3.59</td>
<td>0.16</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.56</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>26.78</td>
<td>0.02</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>7.76</td>
<td>0.97</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>0.82</td>
<td>0.00</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.60</td>
<td>0.01</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>6.14</td>
<td>0.00</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>6.65</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Table D.24: Optimal parameters for tracker RGBD for each objective function when running the WIR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>5.80</td>
<td>0.83</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.40</td>
<td>0.91</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>178.70</td>
<td>0.78</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>13.90</td>
<td>0.91</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>1.58</td>
<td>0.81</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.40</td>
<td>0.91</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>69.20</td>
<td>0.82</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>11.10</td>
<td>0.91</td>
</tr>
</tbody>
</table>

Table D.25: Optimal parameters for tracker MultiChannel for each objective function when running the WIR test

<table>
<thead>
<tr>
<th>Objective function</th>
<th>Objective value</th>
<th>decay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>6.08</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.38</td>
<td>1.00</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>232.56</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>13.33</td>
<td>0.81</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>1.69</td>
<td>0.73</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.38</td>
<td>0.98</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>76.04</td>
<td>0.73</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>10.76</td>
<td>0.79</td>
</tr>
<tr>
<td>Objective function</td>
<td>Objective value</td>
<td>decay</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-----------------</td>
<td>-------</td>
</tr>
<tr>
<td>Mean of mean of box log likelihood</td>
<td>6.48</td>
<td>0.91</td>
</tr>
<tr>
<td>Mean of mean of box overlap</td>
<td>0.39</td>
<td>1.00</td>
</tr>
<tr>
<td>Mean of sum of box log likelihood</td>
<td>219.47</td>
<td>0.91</td>
</tr>
<tr>
<td>Mean of sum of box overlap</td>
<td>12.92</td>
<td>0.98</td>
</tr>
<tr>
<td>Median of mean of box log likelihood</td>
<td>1.81</td>
<td>0.97</td>
</tr>
<tr>
<td>Median of mean of box overlap</td>
<td>0.39</td>
<td>1.00</td>
</tr>
<tr>
<td>Median of sum of box log likelihood</td>
<td>82.29</td>
<td>0.91</td>
</tr>
<tr>
<td>Median of sum of box overlap</td>
<td>11.01</td>
<td>0.99</td>
</tr>
</tbody>
</table>

*Table D.26: Optimal parameters for tracker MultiChannelOpt for each objective function when running the WIR test*


[19] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost van de


[52] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized repre-


Index

BM  abbreviation, xii
DCF  abbreviation, xii
DEFINITIONS
  Single Target Visual Object Tracking, 4
  Visual Tracking, 3
EKF  abbreviation, xii
EMD  abbreviation, xii
GMM  abbreviation, xii
HOG  abbreviation, xii
ICP  abbreviation, xii
IMCMC abbreviation, xii
IMU  abbreviation, xii
IPRF abbreviation, xii
KCF  abbreviation, xii
LBP  abbreviation, xii
MIL  abbreviation, xii
MVL  abbreviation, xii
NCC  abbreviation, xii
NOR  abbreviation, xii
SAD  abbreviation, xii
SGM  abbreviation, xii
SSD  abbreviation, xii
SVD  abbreviation, xii
SVM  abbreviation, xii
WIR  abbreviation, xii