Incorporating Scene Depth in Discriminative Correlation Filters for Visual Tracking

John Stynsberg
Master of Science Thesis in Electrical Engineering

Incorporating Scene Depth in Discriminative Correlation Filters for Visual Tracking

John Stynsberg
LiTH-ISY-EX–18/5178–SE

Supervisor:  Abdelrahman Eldesokey
            ISY, Linköpings universitet
Erik Ringaby
            SICK IVP

Examiner:    Fahad Shahbaz Khan
            ISY, Linköpings universitet

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

Copyright © 2018 John Stynsberg
Abstract

Visual tracking is a computer vision problem where the task is to follow a target through a video sequence. Tracking has many important real-world applications in several fields such as autonomous vehicles and robot-vision. Since visual tracking does not assume any prior knowledge about the target, it faces different challenges such as occlusion, appearance change, background clutter and scale change.

In this thesis we try to improve the capabilities of tracking frameworks using discriminative correlation filters by incorporating scene depth information. We utilize scene depth information on three main levels. First, we use raw depth information to segment the target from its surroundings enabling occlusion detection and scale estimation. Second, we investigate different visual features calculated from depth data to decide which features are good at encoding geometric information available solely in depth data. Third, we investigate handling missing data in the depth maps using a modified version of the normalized convolution framework. Finally, we introduce a novel approach for parameter search using genetic algorithms to find the best hyperparameters for our tracking framework.

Experiments show that depth data can be used to estimate scale changes and handle occlusions. In addition, visual features calculated from depth are more representative if they were combined with color features. It is also shown that utilizing normalized convolution improves the overall performance in some cases. Lastly, the usage of genetic algorithms for hyperparameter search leads to accuracy gains as well as some insights on the performance of different components within the framework.
Acknowledgments

I thank my supervisors Erik and Abdelrahman for all their help when I would come along with my ill-posed questions. I’d also like to thank my examiner Fahad Khan.

I thank my fellow thesis workers: Kevin, Fredrik and Petter for enduring my ramblings and frequent outbursts.

I want to thank SICK IVP and their employees for allowing me to write my thesis at their office, while supplying me with a computer, workspace, office material, coffee and fruit.

Linköping, September 2018
John Stynsberg
# Contents

1 Introduction
- 1.1 Motivation .................................................. 1
- 1.2 Problem ...................................................... 2
- 1.3 Research Questions ......................................... 4
- 1.4 Limitations ................................................... 4
- 1.5 Outline ....................................................... 4

2 Background
- 2.1 Tracking ....................................................... 8
  - 2.1.1 DCF .................................................... 8
  - 2.1.2 SRDCF .................................................. 11
- 2.2 Visual Features ............................................. 15
  - 2.2.1 Deep Features .......................................... 15
  - 2.2.2 HOG & FHOG ........................................... 16
  - 2.2.3 Color Names ............................................ 21
  - 2.2.4 Depth Clusters .......................................... 22
  - 2.2.5 Jet-encoding ............................................ 23
- 2.3 Scene Depth ................................................. 23
  - 2.3.1 Handling Missing Data ................................ 23
- 2.4 Normalized Convolution .................................. 24
  - 2.4.1 Signal and Certainty ................................... 24
  - 2.4.2 Basis Functions and Applicability .................. 25
  - 2.4.3 Filter Responses using Normalized Convolution .... 25
- 2.5 Conjugate Gradient ....................................... 26
  - 2.5.1 The algorithm .......................................... 26
- 2.6 Bayesian Gaussian Mixtures ............................... 27
  - 2.6.1 Dirichlet process ...................................... 28
- 2.7 Kalman Filters ............................................. 30
  - 2.7.1 Stationary Kalman filter ............................. 30
- 2.8 Genetic Algorithms ....................................... 30
- 2.9 Related Work .............................................. 32
  - 2.9.1 Tracking in RGB and RGB-D ......................... 32
  - 2.9.2 Deep Learning in Tracking ......................... 33
1

Introduction

1.1 Motivation

Visual tracking is the problem of following a specific object of interest in a sequence of images or a video. Typically, the object is manually selected in the first frame and then the task is to follow this object through the consecutive frames. The importance of visual tracking is evident in many existing and emerging fields, such as autonomous vehicles, video-surveillance and robot vision. In biomedical image analysis, visual tracking can be used to provide statistics, for example, for white blood cells chasing bacteria.

One of the most essential papers on tracking is the 1994 paper by Shi and Tomasi, "Good Features to Track" [33], where they claimed: "Although tracking by itself is by and large a solved problem, ...". Shi and Tomasi argued that the areas that need development are the input to the trackers, not the trackers themselves. Today, after 25 years, visual trackers still suffer with difficult challenges, such as occlusion, lighting conditions, shape changes and camera pose.

Many trials have been made in the literature to address the aforementioned challenges. This includes building robust object models or incorporating auxiliary information in the object model. Scene depth started to grab attention in many computer vision tasks, showing great potential. In most standard vision tasks the only data available is the grayscale or color image representing the scene at hand. It's difficult then to develop solutions that are invariant to the lighting conditions, or that are able to differentiate between an object changing appearance and object being occluded by another. This is where scene depth can fill in the blanks. The scene depth is the image containing the distance from the camera to everything in the image. The depth data is impacted less by light changes compared to normal images, which means that even if the lighting in the scene changes drastically, the depth sensor would still provide the roughly the same in-
formation. By using the depth it should be easier to distinguish the cases where
the target undergoes an appearance change, a scale change or an occlusion.

However, scene depth is not faultless. Depth sensors are only able to output
estimated depth within a certain interval. For the Microsoft Kinect v2 this depth
range is from 0.4 meters to 8 meters. Outside of this range the data is called
missing data and the placeholder value is usually set to 0. This is a problem
for computer vision algorithms, because the value in these pixels is not 0 but
unknown. Depth sensors also output missing data where the depth estimation
failed, which depends on the estimation strategy used.

In this thesis, we investigate incorporating depth into existing discriminative
correlation filter (DCF) frameworks. This would remedy the issues of lighting
problems, occlusion detection and enable solutions such as allowing autonomous
vehicles to operate in a secure manner at all times of the day, in all types of
locations.

1.2 Problem

As mentioned before, visual tracking presents several challenges such as occlu-
sion and appearance change. Occlusion handling is a challenging problem be-
cause it is difficult to determine if the object is being occluded or is changing its
appearance. We use the depth information to determine if an occluding object is
present or not. Another issue is scale change, where it is difficult to accurately
estimate the target size through an appearance model. An example of a sequence
where scale estimation is necessary can be seen in Figure 1.1. The approaches in
literature use a search in scale-space [12], evaluating at which scale the learned
appearance model fits best. In this thesis the depth information allows for fast
segmentation of the target which in turn allows for a direct calculation of the
target size.

For this thesis, the focus will be on Discriminative Correlation Filters (DCF)
as an approach to visual tracking. More details will be provided in chapter 2 on
how these work, but they are in essence filters that are trained on-line to be able
to distinguish between the target and their surroundings. These filters depend
heavily on the input to be able to discriminate properly. The inputs are called
feature maps and finding good ones is crucial for the quality of the performance.
It is therefore interesting to see if there are good feature maps calculated from
the depth data that can serve as inputs to the tracker. These features would be
invariant to light changes and would, in that sense, be more robust than RGB
features.

The integration of scene depth is not limited to inputting data into the filters
however. The depth data contains spatial information that can't be as simply
extracted from the color image. Estimation of target size and occlusion estima-
tion are both problems that are very difficult in color images, but which can be
estimated in depth more easily. For this a modular extension can be made and
attached onto the existing tracking frameworks.
1.2 Problem

(a) Initial frame, ground truth green box is given around the target. In this case a post-it note attached to a thermos.

(b) An image from the middle of the sequence. The target has changed appearance from the initial frame.

(c) Here the appearance has been restored, but the scale has changed

Figure 1.1: A tracking sequence showing some of the difficulties with tracking.
A final element that will be considered in this thesis is the problem of missing data. When processing the scene depth, some pixels will not contain spatial information of the scene, but rather the missing data value, meaning that the sensor failed to measure the distance to that point. This needs to be handled in a different way than normal sensor data, as existing tracking algorithms don’t have a concept of "uncertain" or "missing" data values.

1.3 Research Questions

1. What type of input features are good when tracking in video sequences with both color and depth information?

2. Can we compensate for missing data in DCF-based tracking using normalized convolution?

3. Can we estimate the scale of the target using the scene depth?

4. Can occlusion be estimated and handled using the scene depth?

1.4 Limitations

The thesis had a time limit of roughly 6 months. The hardware resources available during this thesis was a computer with a Nvidia GeForce 1080. Although suitable for training smaller convolutional neural networks, training large networks on large amounts of data were not feasible. For datasets containing annotated videos, there was a possibility of recording and annotating videos myself. However, due to the availability online and the time constraint, this was deemed outside of the scope of the thesis.

The Computer Vision Laboratory at Linköping University have published the source code for all the modern state-of-the-art trackers that they’ve authored. These resources could not be used due to the office that this thesis was written at, did not own the necessary MathWorks licenses. Therefore, the trackers were implemented in Python from scratch.

The thesis only considers two different tracking frameworks: Discriminative Correlation Filters (DCF) [12] and Spatially Regularized Discriminative Correlation Filters (SRDCF) [9]. Due to the time limitation, it would have been too cumbersome to implement, test, experiment and evaluate more tracking frameworks.

1.5 Outline

The first chapter, called Background, will explain the theoretical background required to understand the work that is done, while also talking about what else is done in the academic field regarding visual tracking and deep learning both in RGB and RGB-D. The background chapter will start with a couple of sections
delving into the mathematics detailing DCF-strategies, feature extraction procedures and other concepts that will be needed for the understanding of the thesis. Afterwards, the related work section will start and there the reader can find an overview of what has been done in fields of RGB and RGB-D tracking, deep learning and its applications in visual tracking and finally deep Learning in RGB-D.

The third chapter will be the Method chapter. Here the approaches, which are used to answer the questions posed by this thesis, are presented. First the modifications done to the tracking frameworks will be explained, then the attention control section will show how normalized convolution was used to enable attention control for the filters. Afterwards the implementation of Gaussian mixture models (GMM) for scale-estimation and occlusion handling will be explained. Last is the section describing the design and training of the convolutional neural networks (CNN) used for extracting features.

The fourth chapter is the Experiments chapter. This chapter will go over some of the implementation details of which Python libraries that were used. This chapter also features a description of the datasets used both for CNN training and tracking evaluation and a presentation of the results in terms of raw statistics.

The Discussion chapter is the final chapter and in this chapter the results are interpreted and analyzed. After this a critical lens is put to the method of this thesis, by discussing what decisions were the most influential in terms of the results of the thesis. The discussion is finalized by a look into the future, talking about in which areas it would be interesting to do more research.
This chapter will introduce the background to this thesis. Sections 2.1 to 2.8 deal with the theory needed to understand the methods used to answer the research questions. After these theory sections, Section 2.9 will give an overview of the related work in the field. As an aid to understand each section’s role in the thesis as a whole, a system overview was created, see Figure 2.1.

First section 2.1 and section 2.2 will present the theory behind the tracking frameworks used and the visual features that are used as input to these frameworks. Afterwards section 2.3 and section 2.4 will present how scene depth information is obtained, the problem of missing data and how normalized convolution, an extension of convolution, can be used to handle uncertain data. In section 2.5, Conjugate Gradient is introduced. Conjugate Gradient is used to efficiently solve equations in one of the tracking frameworks. Next is section 2.6 and section 2.7 that describe Bayesian Gaussian Mixture Models and the Kalman filter. These are the tools that are used for scale estimation, occlusion detection and occlusion recovery. Last in the theory section is section 2.8 which presents genetic algorithms, which are used for hyperparameter search for the trackers. After these sections an overview of the related work in the field is described in section 2.9.
2.1 Tracking

This section will present the theory behind the two tracking frameworks used in this thesis, the multi-channel Discriminative Correlation Filters (DCF) and the Spatially Regularized Discriminative Correlation Filters (SRDCF).

2.1.1 DCF

Discriminative correlation filters is the name of a group of tracking frameworks used for tracking. The common denominator between these frameworks is that they train a filter online that discriminates between the target object and the background. Correlation filters were used to solve other computer vision problems during the 80’s and 90’s, but one of the first successful instances of employing these for tracking was the MOSSE tracker presented in [5] by Bolme et al. This filter is trained and updated on the greyscale image of the target in every frame. The big contribution of this filter was the idea to conduct both the training and inference of the filter in the frequency domain. One issue with MOSSE though, was the limitation of only being able to use single-channel features.

The simplest tracking framework utilized in this thesis is the multi-channel DCF, which allowed using multiple channels in DCF frameworks, proposed by Danelljan et al. in [12]. The DCF update proposed closely resembles the MOSSE [5] and the Dual correlation filter [21] with some minor modifications. As mentioned before, in MOSSE there is no support for multi-channel features and in
the dual correlation filter there is no support for using training samples of the target appearance from more than one time instance.

Assume a sample \( f \) corresponds to the image patch centered around the target. The sample \( f \) will be \( D \times N \times M \) dimensional, where \( D \) is the number of feature dimensions, \( N \) is the height and \( M \) is the width. For example, gray-scale images, would have \( D = 1 \) feature dimensions, RGB, would have \( D = 3 \) dimensions and the Histogram of Oriented Gradients (HOG) features discussed later in this thesis would have \( D = 31 \) feature dimensions. This sample is used to learn several 2-dimensional correlation filters, for each feature dimension, that is used to estimate the target translation. The feature channel \( d \) will be denoted as \( f^d, d \in \{1, \ldots, D\} \). The goal is to learn \( h \), consisting of one filter \( h^d \) per channel, that produces the desired correlation output \( g \). This \( g \) is typically a Gaussian function with standard deviations that scale with \( M \) and \( N \). The loss-function used is the \( L^2 \)-norm between the filter \( h \) applied to the sample \( f \) to the goal function \( g \).

\[
e(h) = \| g - \sum_{d}^{D} f^d \star h^d \|^2 + w \sum_{d}^{D} \| h^d \|^2 \tag{2.1}
\]

\( w \) is the uniform weight penalty on the filter coefficients and \( \star \) denotes circular correlation. This is a linear least squares problem that can be solved efficiently by transforming Equation 2.1 to the DFT domain using Parsevals formula. Then the filter that minimizes Equation 2.1 becomes:

\[
\hat{h}^d = \frac{\overline{\mathcal{F}} f^d \cdot \overline{\mathcal{F}} g}{\sum_{k=1}^{D} \overline{\mathcal{F}} f^k \cdot \overline{\mathcal{F}} g + w}
\]

here a hat \( \hat{h} \) above a letter denotes the DFT of the corresponding entity, the bar \( \overline{\ } \) denotes the complex conjugation, \( \cdot \) is the Hadamard-product (element-wise) and the division is also element-wise. For a single base sample \( f \) this formulation is the same as the dual correlation filter introduced in [21] by Henriques et al. They show that this is equivalent to training on all cyclic shifts of a base sample, this due to all cyclic matrices being made diagonal by the DFT matrix, the matrix which performs the DFT on a given vector. A complete derivation and explanation of this interaction can be found in [21]. The fact that our filter will be trained on all cyclic shifts of a sample carries with it positives and negatives. On the positive side, the learned filter has trained on all cyclic shifts of the sample, which resemble all translations of the sample within the image patch. But on the other side, the difference between cyclic shifts and true translation is the distortions along the edges of the sample. To mitigate this distortion, all features are masked with a windowing function to "darken" the edges of the samples. This is visualized for a simple 1-D case in Figure 2.2.
The base sample and the circulant matrix containing all cyclic shifts of the sample.

**Figure 2.2:** A visualization of a circulant matrix containing cyclic shifts for different base samples. The left image shows a case with no masking of the base sample. The right image shows a case with a masked base sample. In the masked case, any distortion resulting from the connection of the left and right edge, is smoothed.

As mentioned before, one limitation of the dual correlation filter is that it does not include training on more than one base sample. For one base sample, the resulting filter can be trained only using element-wise operations. This is due to the circulant sample matrix being made diagonal by Discrete Fourier Transform (DFT) matrix. But when the sample matrix is extended to contain multiple training samples, the property does not hold anymore. Rather, the DFT matrix will make the sample matrix block diagonal. However, by following the arguments of Danelljan in [12], a robust approximation can be made by combining the exact solution for the single sample with the update strategy from [5]. Let \( \{ f_t \}_{t=1}^{t_c} \) be the training samples taken at different times \( t \) up until time \( t_c \), the filter \( \hat{h}_t^d \) is created by updating the numerator \( \hat{a}_t^d \) and denominator \( \hat{b}_t \) at each time step as follows,

\[
\hat{a}_t^d = (1 - \lambda) \hat{a}_{t-1}^d + \lambda \overline{f}_t^d, \quad d = 1, \ldots, D \\
\hat{b}_t = (1 - \lambda) \hat{b}_{t-1} + \lambda \sum_{k=1}^{D} \overline{f}_t^k f_t^k
\]

(2.3)

\( 0 \leq \lambda \leq 1 \) denotes the learning rate of the filter, a low learning rate means that it will be robust to noise, while a high learning rate handles appearance changes better.
2.1 Tracking

Figure 2.3: A visualization of the training result of a DCF. To the left, the training sample patch. In the middle, the trained filter. To the right, the response produced by applying the filter to the training sample. The outline of the drawing can almost be seen in the trained filter.

To apply the learned filter in a new frame $t$, a sample $z_t$ is extracted from the predicted target location. The extraction is done in the same way as $f_t$ is extracted, which means that $z_t$ is also $D \times N \times M$ dimensional. The correlation scores are computed in the DFT domain

$$\hat{y}_t = \frac{\sum_d \hat{a}_{t-1}^d \hat{z}_t^d}{\hat{b}_{t-1} + \hat{w}}. \quad (2.4)$$

The target location is then obtained by finding the maximum of $y_t = F^{-1}\{\hat{y}_t\}$ where $F^{-1}$ denotes the inverse DFT. An example of this is shown in Figure 2.3.

One should note that this formulation only requires the features to be of same size, they do not need to come from the same source. HOG features with cell-size $4 \times 4$ can be combined with RGB features as long as the RGB image is downsampled, such that both feature types have the same size $M \times N$.

2.1.2 SRDCF

The motivation behind the Spatially Regularized Discriminative Correlation Filters (SRDCF) was to constrain the coefficients of the filters from DCF further. It would be desirable to extract larger image patches around the target, to become more robust against target/camera movements. Bigger patches would also allow for more negative samples which would help the classifier generalize. Unfortunately this is something that the standard DCF framework can’t handle. More background in the training samples tends to lead the filters to learn and track the background rather than the target.
Another issue with DCF that SRDCF attempts to solve is the periodicity distortion. Even though DCF tries to mitigate the effects of the distortion by windowing functions on the input patches, the issue still remains. If the filter is forced to "look away" from the background, then this would provide a more robust solution to this problem.

To incorporate all this in the loss function, first some terms must be defined. Let’s define \( S_h(f) \) to be the convolution response from applying filter \( h \) on feature \( f \) such as,

\[
S_h(f) = \sum_d f^d \star h^d. \tag{2.5}
\]

To enable this regularization the \( L^2 \)-error of DCF is generalized to Tikhonov regularization. Also \( \{\alpha_t\}_t^k \) is applied as a coefficient defining the weight of each sample at each time step \( t \) and where \( k \) is the current time instance. This corresponds to the learning rate applied in the DCF formulation,

\[
e(h) = \sum_{t=1}^k \alpha_t \| S_h(f_t) - g_t \|^2 + \sum_d \| w \cdot h^d \|^2. \tag{2.6}
\]

By applying Parseval’s theorem, we can minimize over the DFT coefficients of \( h \),

\[
\hat{e}(\hat{h}) = \sum_{t=1}^k \alpha_t \left\| \sum_d f^d_t \cdot \hat{h}^d - \hat{g}_t \right\|^2 + \sum_d \left\| \frac{\hat{w} \star \hat{h}^d}{MN} \right\|^2. \tag{2.7}
\]

A vectorization of Equation 2.7 gives

\[
\hat{e}(\hat{h}) = \sum_{t=1}^k \alpha_t \left\| \sum_d \text{D}(\hat{f}^d_t)\hat{h}^d - \hat{g}_t \right\|^2 + \sum_d \left\| \frac{C(\hat{w})}{MN} \hat{h}^d \right\|^2, \tag{2.8}
\]

where a vectorization of a matrix is denoted with boldface \( \text{vec}(\hat{h}) = \hat{h} \), \( \text{D}(a) \) is the diagonal matrix with \( a \) on its diagonal, i.e. \( \text{D}(a)b = a \cdot b \). \( \text{C}(a) \) is the circulant matrix containing cyclic permutations of \( a \) in its rows, thus it is equivalent to circular correlation \( \text{C}(a)b = a \star b \). Now let’s introduce the following matrices

\[
\hat{\delta}^d_t = \text{D}(\hat{f}^d_t) \tag{2.9}
\]

\[
\hat{c} = \frac{C(\hat{w})}{MN} \tag{2.10}
\]

\[
\hat{h} = \left( (\hat{h}^1)^H \ldots (\hat{h}^D)^H \right)^H \tag{2.11}
\]

\[
\hat{\delta}_t = \left( \hat{\delta}_t^1 \ldots \hat{\delta}_t^D \right). \tag{2.12}
\]
Finally $\hat{\gamma}$ is defined as a $DMN \times DMN$ block diagonal matrix containing $\hat{\epsilon}$ on each diagonal block. This allows us to rewrite Equation 2.8 into the following:

$$
\hat{\epsilon}(\tilde{h}) = \sum_{i=1}^{k} \alpha_i \|\hat{\delta}_i \tilde{h} - \hat{g}_i\|^2 + \|\hat{\gamma} \tilde{h}\|^2.
$$

Equation 2.13 is now minimized by solving the normal equations $\hat{a}_t \tilde{h} = \tilde{b}_t$

$$
\hat{a}_t = \sum_{t=1}^{k} \alpha_t \hat{\delta}_t^H \hat{\delta}_t + \hat{\gamma}^H \hat{\gamma}
$$

$$
\tilde{b}_t = \sum_{t=1}^{k} \alpha_t \hat{\delta}_t^H \hat{g}_t.
$$

One important optimization that is possible for Equation 2.14 and 2.15 is that the spatial-domain signals are all real-valued before their DFT which means that they will have Hermitian symmetry,

$$
g(x, y) \in \mathbb{R}, \forall x, y \in \mathbb{R}
$$

$$
G(u, v) = \mathcal{F}\{g(x, y)\}
$$

$$
G(u, v) = G^*(-u, -v).
$$

Here $\mathcal{F}$ denotes the Fourier transform. This means that it is possible to utilize this Hermitian symmetry to only make calculations for half the spectrum and utilize the symmetry to extrapolate the second half. The difference between the correlation filters when the restriction of the filter coefficients is applied can be seen in Figure 2.4.
(a) The weights of the DCF are spread over the entire image patch.

(b) SRDCF has its weights confined to the target

Figure 2.4: A comparison of the trained filter coefficients in DCF and SRDCF. SRDCF doesn’t “assign” coefficients to the background and is therefore more robust in theory.
2.2 Visual Features

This section will describe the theory behind the visual features used in this thesis. Visual features are calculated from input images to enhance certain properties of the image. The features presented are deep features, histogram of oriented gradients, color names, jet encoding and a novel feature introduced in this thesis, depth clusters.

2.2.1 Deep Features

Training convolutional neural networks for computer vision tasks has driven great advances in many disciplines. Conventional CNN architectures can usually be separated into a feature extraction part, consisting of convolutional, non-linearities and pooling layers, followed by a classification part, consisting of fully-connected layers.

Although it might seem that this feature extraction should by tightly connected to both the image task it was trained for and to the classifier it feeds information to, it has been shown this is not the case. The features that are retained from these extractors can be repurposed for other tasks. It has been shown that employing the features from these extractors to other classifiers in different computer vision problems outperform hand-crafted features [8], referring to features developed by researchers such as HOG features. In other words, networks trained for image classification on cat and dog images, will have tuned a feature extractor that can be used as a good starting point for a face detector. This phenomenon is called transfer learning.
The extraction of deep features is very simple. Let’s consider a simple feedforward network CNN where its feature extractor consists of convolutional, rectified linear unit (ReLU) and max-pooling layers as shown in Table 2.1. Let \( A_n \) be the activation from layer \( n \) and \( x \) be the input image. To acquire the activation at a certain layer one needs only to apply the composition, denoted as \( \circ \), of all layer functions \( F \) up to and including that layer:

\[
A_n = F_n(F_{n-1}(F_{n-2}(\ldots (F_2(F_1(x))\ldots))) = (F_n \circ F_{n-1} \circ F_{n-2} \ldots F_1)(x).
\]

(2.17)

In [8], best results were found from using activations from the ReLU layers of VGG-M, which for this example corresponds to \( A_2, A_5 \) and in general \( A_{2+3k} \) where \( k \) is the \( k \):th ReLU-layer.

Depending on which layer in the network these features are extracted, the features have different characteristics. In the early layers, the features tend to represent low-level features such as edges, flat surfaces and colors (if the input was a RGB-image). Close to the end of the feature extractor the features take on abstract interpretations. The activations are focused on more abstract things that consist of structures built on the previously mentioned low-level features. Here activations can be found on things such as faces, leaves and other things on that abstract level. These deeper features are more dependent on the dataset that the network was trained on, as opposed to the earlier layers which tend to behave more similarly across different vision tasks. Examples of these activations can be seen in Figure 2.5.

### 2.2.2 HOG & FHOG

Histogram of oriented gradients (HOG) are hand-crafted features created for the purpose of human detection by Dalal & Triggs in [7]. The features can be said to roughly represent edges in an image, but with more nuances. The HOG features were later augmented by Felzenszwalb et al. in [15] for object detection. The Felzenszwalb HOG (FHOG) has later been successfully used in multiple object detection and classification tasks including tracking frameworks such as the Dual CF [21], DSST [12] and in SRDCF [9]. The features are high dimensional, but a visualization of them can be seen in Figure 2.6.
### 2.2 Visual Features

(a) Activations from the first RGB ReLu-layer in a RGB-D classification CNN ($A_2$). The activations vaguely resemble blurring (2, 8), edge detections (4, 3, 6) and thresholding (0).

(b) Activations from the last layer ($A_{13}$) in the same network as above, after a fusion of depth and color channels. A lot of specialized layers don’t activate much (3, 4), others find sharp activations (6, 8) and some activate on the target/background/something in between (0, 1, 5, 7, 9)

**Figure 2.5:** A subset of activations from the early and late layers in a multistream RGB-D classification CNN. The activations were cherry-picked to demonstrate different characteristics of the features. This is necessary especially in the last layer, where most feature channels barely activate at all.
In short the procedure for calculating HOG features is the following:

**Result:** HOG features

```plaintext
for each pixel do
  Calculate the gradient;
end

Divide the image into cells;

```plaintext
for each cell do
  Create the gradient orientation bins between 0 - 180 degrees;
  Let each pixel vote on the orientation of the cell;
end

Group cells into blocks;

```plaintext
for each block do
  Normalize;
end
```

As described in [15], first gradients are calculated by using centered 1-D derivatives kernels $[-1, 0, 1]$ and its transpose. Let $\theta(x, y)$ be the orientation of the gradient and $r(x, y)$ be the magnitude. For color images the color channel with the greatest gradient norm is chosen as the pixel’s gradient vector. The gradient orientations are discretized using either a contrast sensitive ($B_1$) or an insensitive ($B_2$) definition.

\[
B_1 = \text{round} \left( \frac{p \cdot \theta(x, y)}{2\pi} \right) \mod p, \tag{2.18}
\]

\[
B_2 = \text{round} \left( \frac{p \cdot \theta(x, y)}{\pi} \right) \mod p. \tag{2.19}
\]

where $p$ indicates the number of orientation bins. From now on $B$ will be used to refer to either $B_1$ or $B_2$.

Let $b \in \{0, \ldots, p - 1\}$ range over the orientation bins. Lets define a pixel-level feature vector $F$ that defines a sparse histogram of gradient magnitudes at each pixel.

\[
F(x, y)_b = \begin{cases} 
  r(x, y), & \text{if } b = B(x, y) \\
  0, & \text{otherwise.} 
\end{cases} \tag{2.20}
\]

$F$ now contains the oriented edges over the entire image. For each pixel the closest gradient bin is chosen and the gradient magnitude is the strength of that edge. Let $C$ be a cell-based "soft binning" of $F$. Cell centers are placed at the "crossroads" between each pixel. Each pixel of $F$ will then contribute to the four cells around it using bilinear interpolation.

The next step is normalize the feature map $C$ to make the features less affected by local variations in illumination and foreground-background contrast. Note here that since the features are gradients, there’s no point in subtracting any bias. 4 different normalization factors $N_{\delta, \gamma}(i, j)$, with $\delta, \gamma \in \{-1, 1\}$, are used. These are calculated as follows
2.2 Visual Features

\[ N_{\delta,\gamma}(i, j) = \sqrt{\|C(i, j)\|^2 + \|C(i + \delta, j)\|^2 + \|C(i, j + \gamma)\|^2 + \|C(i + \delta, j + \gamma)\|^2}. \]  

(2.21)

The features are also truncated by \( T_\alpha(v) \) where \( v \) is a vector and the \( i \)-th element of \( T \) is the minimum of \( v_i \) and \( \alpha \). The HOG feature map \( H \) is finally the concatenation of the truncation of \( C \) with respect to each normalization factor,

\[
H(i, j) = \begin{cases} 
T_\alpha(C(i, j)/N_{-1,-1}(i, j)) \\
T_\alpha(C(i, j)/N_{+1,-1}(i, j)) \\
T_\alpha(C(i, j)/N_{-1,+1}(i, j)) \\
T_\alpha(C(i, j)/N_{+1,+1}(i, j)) 
\end{cases}.
\]  

(2.22)

Commonly used HOG features are defined by \( p = 9 \) contrast insensitive gradient orientations, a cell-size of \( k = 4 \) and truncation of \( \alpha = 0.2 \). This results in a \( 4 \times 9 = 36 \) dimensional feature vector. This is where the original HOG features end their formulation.

**Felzenszwalb HOG**

The big extension made by Felzenszwalb et al. \[15\] is the incorporation of both contrast sensitive and insensitive oriented edges. Note that these were both introduced by Dalal and Triggs \[7\], but were limited to using either one or the other. The sensitive edges are calculated with \( p = 18 \) and are sorted into bins to create another cell-level feature map \( D(i, j) \). \( C \) and \( D \) are then concatenated, normalized and truncated to obtain a \( 4 \times (9 + 18) = 4 \times 27 = 108 \) feature map, denoted as \( F \). To reduce the dimension of this feature map, Felzenszwalb et al. show that almost all of the information in the 36-dimensional HOG features are captured by 11 eigenvectors. They also show that these vectors can be approximated by simply projecting the features onto \( 9 + 4 = 13 \) vectors, 9 for the orientations and 4 for the normalizations. The projected features are then created by taking either a orientation and summing over normalizations or vice versa, taking a normalization and summing over orientations.

This same strategy is then applied to \( F \) creating a \( 4 + 18 + 9 = 31 \) dimensional feature map consisting of 9 dimensions representing contrast-insensitive orientations, 18 contrast-sensitive orientations and 4 representing the gradient energy in square blocks. FHOG can be calculated from both RGB and depth data, to distinguish between these FHOG features extracted from depth data are called DFHOG features.
Figure 2.6: Visualization of Felzenszwalb HOG features on both RGB and Depth data. The FHOG from the depth data has more of a geometric interpretation, showing clear edges and smooth surfaces. The RGB is harder to decipher, but a rough outline can be seen.
2.2 Visual Features

2.2.3 Color Names

Color names or color attributes are an RGB feature that represents linguistic labels of colors. From a study made by Berlin and Kay [1], it was concluded that the english language contained eleven basic color terms: black, blue, brown, grey, green, orange, pink, purple, red, white and yellow. In the work of van Weijer et al. [35], a probabilistic mapping was created to convert from the RGB-domain to this 11-dimensional color domain. The new 11-dimensional feature represents the probability that a certain RGB-value corresponds to a color name. This feature has been used to great success in RGB-tracking [12]. These features can be seen in Figure 2.7 and Figure 2.8.

*Figure 2.7: The original image and 11 channels of color names.*
2.2.4 Depth Clusters

Depth clusters is the name of a novel approach to finding features suitable for tracking in RGB-D data, based on clustering of the depth histogram. The initialization of the features is done in the first frame of a tracking sequence where the initial clustering by Gaussian Mixture Models is done. This will assign $N$ probability values to each pixel describing the probability of the pixel belonging to each of the $N$ clusters estimated. In the normal use-case, a pixel would be grouped in with the clusters for which it has the highest probability to belong to. In this case the probabilities are stored in a $W \times H \times N$ feature map. The clusters parameters are updated in each frame by running a couple of iterations of the fitting algorithm of choice. In this thesis, Bayesian Gaussian Mixture Models together with Variational Inference with Dirichlet Process prior was used for clustering, but in practice any clustering algorithm that can be iteratively updated could be used, such as K-means. An overview of Bayesian Gaussian Mixture Models will be done in section 2.6.

The inspiration for this method came from the color names subsection 2.2.3, since they are also a probabilistic mapping, that provides a robust mapping of the color space, making the filters less sensitive to small changes in color. The idea for the depth clusters was to similarly de-noise the depth data to make the tracking filters unperturbed by smaller movements in the z-axis.
2.2.5 Jet-encoding

The last feature presented is the jet color-encoding. This feature is a simple mapping from values in between \([0, 1]\) to RGB values. The mapping can be seen in Figure 2.9. In this thesis it is used to encode the depth image. The depth image is normalized with the highest value that the depth sensor can output and then encoded. The resulting feature has the same spatial resolution as the input image, but with 3 color channels.

There have been attempts to encode geometric information in the depth image when doing RGB-D deep learning [19], this has however been outperformed by simply jet-encoding the depth image [13]. The jet-encoding also enabled transfer learning from networks from the RGB-domain.

Figure 2.9: The jet colormap. 0 maps to blue and 1 maps to red.

2.3 Scene Depth

In computer vision, the scene depth can be projected onto a 2D image where every pixel contains either the estimated depth or a placeholder value for when the depth estimation failed. The resolution of this image and the cases where missing data is output are of course different from sensor to sensor, depending on their depth estimation method.

Two popular methods for depth estimations are structured light and time-of-flight (TOF). Microsoft Kinect v1 and Asus Xtion both use structured light. This means that they project a structured light pattern on the scene and then capture images of this pattern using an IR-camera. After this, the displacement of the structured light is used to estimate the depth. The Kinect v2 used TOF instead, which measures the time that light traveled from the camera until it reached the camera again. A complete analysis of how these measurements are done and when they fail is outside of the scope of this thesis. It’s only necessary to know that they do frequently output missing data, both for objects outside the depth range but also in certain difficult cases.

2.3.1 Handling Missing Data

Missing data is fundamentally different from normal data values. These pixels contain a placeholder value, which must not be confused for real data values. This presents problems for calculations of geometric features on the data. A method that aims to estimate the surface normals on the depth data might interpret the placeholder values as discontinuities in the data or as objects in their own. In the case of visual tracking using DCF, using the raw depth data could severely hurt the the stability of a filter if the learned object appearance is based on failures in the sensor.
The handling of missing data is an interesting problem however. There exist methods such as normalized convolution and normalized differential convolution which enable a certainty value to be assigned to every pixel. This could be used to focus the filters attention to the reliable pixels, as was shown by in West-ins Ph.D. thesis [38].

2.4 Normalized Convolution

Normalized convolution [26] is a method for signal analysis that accounts for the uncertainty of signal values while also enabling spatial control for otherwise spatially uninhibited analysis functions. The geometrical interpretation of normalized convolution is that it serves as a projection upon a subspace spanned by the analysis functions. The projection becomes a weighted least squares problem, where the weights are decided by the certainty of the signal together with the desired localization of the analysis function. The result is at each signal point a set of coordinate expansions, one for each analysis function.

This extension of convolution allows for applying filters while being aware of the certainty of the data in every pixel and compensating for this with an interpolation from the surrounding data. It also allows for control of every pixels influence on the filters output. In the context of tracking with DCF, this allows for reducing the impact of missing data.

2.4.1 Signal and Certainty

Let $f$ be the entire signal, while $f$ be the neighbourhood around a given point. Regardless of the dimensionality of the signal, $f$ can be represented by a column vector of size $n$. In the case of a $W \times H$ image patch with three color channels, it would be flattened to a $n = W \times H \times 3$ vector. Let $c$ denote the certainty in every element of $f$. The values of $c$ will all be non-negative, where 0 can be interpreted as complete uncertainty. Although there is no restriction on the upper bound of these values, the value of 1 has an intuitive representation as a value of complete certainty.

Together these two entities represent the signal data and the reliability of the signal data. The reason for this unreliability could be deficient sensors or artifacts from a pre-processing step. It can also be used to reduce border effects when operations such as convolution are applied together with zero-padding along the border of a signal. In the realm of RGB-D data, the certainty is intuitively applied to the missing data pixels, where the depth sensors failed to estimate the depth.
2.4 Normalized Convolution

2.4.2 Basis Functions and Applicability

The basis functions define the subspace in which we estimate a local model for the signal. Each base vector $b$ is a column vector of equal length to $f$ and $c$. The set of $m$ basis functions are stored in a $m \times n$ matrix $B$

$$
B = \begin{pmatrix}
| & | & | \\
\mathbf{b}_1 & \mathbf{b}_2 & \ldots & \mathbf{b}_m \\
| & | & |
\end{pmatrix}
$$

(2.23)

The applicability map $a$ can be interpreted as the certainty map, but for the basis functions. It will also be of length $n$ and define how much each basis function is weighted in the projection.

2.4.3 Filter Responses using Normalized Convolution

According to Ph.D thesis of Farnebäck [14], the equation for performing normalized convolution using the definitions above is

$$
\mathbf{r} = (\mathbf{B}^\dagger \mathbf{D}(a) \mathbf{D}(c) \mathbf{B})^{-1} (\mathbf{B}^\dagger \mathbf{D}(a) \mathbf{D}(c) \mathbf{f}),
$$

(2.24)

where $\mathbf{r}$ is response of the operation, $\ast$ denotes conjugate transpose and $\mathbf{D}(\mathbf{x})$ is the matrix with $\mathbf{x}$ on its diagonal.

For the understanding of this thesis, the complete formulation of normalized convolution is not needed. This because the implementation only uses the special case where the subspace defined by $\mathbf{B}$ only contains one base vector, $\mathbf{b}_1 = (1, 1, 1, \ldots, 1)^T$. This reduces the Equation 2.24 to

$$
\mathbf{r} = \left\langle \mathbf{a}, \mathbf{c} \cdot \mathbf{f} \right\rangle \\
\left\langle \mathbf{a}, \mathbf{c} \right\rangle
$$

(2.25)

where $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the scalar product between vectors $\mathbf{x}$ and $\mathbf{y}$ and $\cdot$ denotes element-wise multiplication.

Keep in mind that $\mathbf{f}$ is only an image patch of $\mathbf{f}$. Let us now similarly denote $\mathbf{r}$ to be the response over the entire image. If we now let $\mathbf{a}$ be invariant of the patch/position currently considered, this means that we can equivalently formulated this as a division between two cross-correlations. The expression then becomes

$$
\mathbf{r} = \frac{\mathbf{a} \ast \mathbf{c} \cdot \mathbf{f}}{\mathbf{a} \ast \mathbf{c}}
$$

(2.26)
2.5 Conjugate Gradient

Conjugate gradient (CG), as a method for solving linear systems was proposed by Hestenes and Stiefel in [22]. It is an algorithm for solving linear equation systems that can be described by symmetric and positive-definite matrices. It exists both as a numerical solution and an iterative method. The iterative method is useful when solving for large and sparse matrices that fulfill the aforementioned conditions. Consider for example \( Ax = b \) where \( A \) is a very large and sparse matrix, \( x \) and \( b \) are column vectors. When solving with the conjugate gradient, \( A \) never has to be realized in the computers memory. We can replace \( A \) with a function \( f \), such that \( f(x) = b \). This gives us an opportunity to ignore all the zeros in \( A \) and only compute the bare necessities which reduces the computational complexity.

2.5.1 The algorithm

Let \( Ax = b \) be the system where we want to find a good approximation for \( x \). The objective function that we aim to minimize is the quadratic equation

\[
  f(x) = \frac{1}{2}x^TAx - x^Tb
\]

which has an unique solution thanks to \( A \) being positive-definite and symmetric. Using an initial guess of \( x = x_0 \). We will now minimize the error by using the negative gradient direction at \( x_0 \)

\[
  p_0 = -\nabla f(x_0) = b - Ax
\]

where \( p_0 \) denotes the search direction. We will also make use of the residual (error) at each step \( k \)

\[
  r_k = b_k - Ax_k
\]

which also is computed by the negative gradient. Now we impose the requirement that all search directions \( p_k \) must be conjugate to each other with respect to \( A \), i.e. \( \{ p_i^T Ap_i = 0 \ \forall i, j \geq 0, i \neq j \} \).

We can now build the next search direction from all previous search directions and the current residual

\[
  p_k = r_k - \sum_{i<k} \frac{p_i^TA r_k}{p_i^T A p_i} p_i
\]

This type of constraint should be familiar to anyone that has built orthogonal basis using Gram-Schmidt orthonormalization:

\[
  u_k = v_k - \sum_{i<k} \frac{u_i^T v_k}{u_i^T u_i} u_i
\]

Iterating using the directions \( p_k \) and the optimal step length \( \alpha_k \) given by

\[
  x_{k+1} = x_k + \alpha_k p_k
\]

\[
  \alpha_k = \frac{p_k^T r_k}{p_k^T A p_k}
\]
will progressively give better estimates of the $x$ that minimizes our objective. It is practical here to apply some error tolerance to jump out of the solver when a good enough solution has been found. Conjugate gradient will converge in a maximum of $n$ steps, where $n$ is the dimensionality of the system, but later we will use CG to optimize filters. It’s not unreasonable for these filters to be of size $80 \times 80 \times 31 = 198400$ which might be too much when aiming for real-time performance. A comparison between gradient descent and conjugate gradient can be seen in Figure 2.10.

**Figure 2.10:** Visualization of CG and gradient descent, in a linear problem. The conjugate restriction allows for a quick solution. CG in black, gradient descent in blue.

### 2.6 Bayesian Gaussian Mixtures

Gaussian Mixture Models (GMM) are used to model groupings within a population. These models are described by a set of parameters that define their shape, position, weight and so forth.

In this thesis mixture models will be used for clustering of depth data. Clustering is when unlabeled data is sorted into groups based on similar characteristics. A very popular method for doing the estimation of these parameters is the maximum-likelihood method Expectation-Maximization (EM). However, in this thesis an extension of EM, called Variational Inference (VI) introduced by Jordan...
et al. [24], will be used.

This method maximizes a lower bound on model evidence instead of data-likelihood. This basically means that VI will integrate information from prior distributions when iterating. This does help to avoid singularities, that sometimes EM will struggle with.

The main reason for using VI using instead of EM is the inclusion of an extra hyper-parameter called weight concentration prior, from now on denoted as \( \nu_0 \). This parameter controls the weight applied to the Gaussians. A low \( \nu_0 \) will weight a few distributions more heavily, while a high value will let a larger amount of distributions be active.

VI together with a Dirichlet Process, introduced by [16], as a prior allows for setting a maximum value of clusters, where the amount of active clusters may or may not reach this maximum depending on the \( \nu_0 \) chosen. This is very useful for this thesis, where clustering will be used to estimate the distance and size of the target. If too many Gaussians become active, the target will be separated into parts and if too few are estimated then the target will blend in with other objects in the scene.

### 2.6.1 Dirichlet process

Dirichlet processes are processes which realizes probability distributions. It takes in a distribution \( G \) and a concentration parameter \( \alpha \) and creates a new distribution denoted as \( \text{DP}(G, \alpha) \). Samples can be extracted from this new distribution where with each sample drawn, there is a probability that the sample will be a new sample or an old one. For each time that an old sample is drawn, the probability of drawing it again increases. The concentration parameter \( \alpha \) controls the possibility of a new sample being drawn.

A more mathematical definition is that a Dirichlet processes probability distribution \( p \) is constructed from the process as

\[
p(x) = \sum_{k=1}^{\infty} \beta_k \delta_{x_k}(x)
\]

where the samples \( \{x_k\}_{k=1}^{\infty} \) are drawn from the given distribution \( G \), \( \delta_{x_k}(x) \) is the impulse that is zero everywhere except for \( \delta_{x_k}(x_k) = 1 \) and the weights \( \beta \) are repeatedly drawn from \( \text{Beta}(1, \alpha) \). In this thesis a Dirichlet process is used as a prior to the VI-algorithm. The process is used to estimate initial parameters of the clusters according to the \( \nu_0 \) and the maximum number of clusters specified. In Figure 2.11 a comparison is shown between a Gaussian Mixture estimated with EM and two Bayesian Gaussian Mixtures estimated with VI and a Dirichlet process prior. The two BGMs used different weight concentration priors, one with a high concentration and the other with a low concentration.
Figure 2.11: Comparison between Gaussian Mixtures and two Bayesian Mixtures estimated with different weight concentration priors. All mixtures were allowed 5 components. The GM used all 5, the BGM with a low $v_0$ used 4 and the BGM with high $v_0$ used 3 components.
2.7 Kalman Filters

Kalman filters is a recursive method that is used for prediction, smoothing and filtering. The filter uses a description of a dynamical linear model describing how old state vectors, measurements and noise affect the current state vector. The flow of the Kalman filter is to predict, measure and then update. Reminiscent of machine learning, the filter guesses a future state given the model it has, gets an estimate and updates it’s hidden variables to improve future estimates.

2.7.1 Stationary Kalman filter

The stationary Kalman filter can be formulated using the following state space model

\[
p_{k+1} = Fp_k + w_k \\
z_k = Hp_k + v_k \\
\text{Cov}(w_k) = Q, \quad \text{Cov}(v_k) = R
\]

where \(p\) denotes the state vector, \(z\) is the observation vector, the matrices \(F\) and \(H\) are state transition matrices and \(w\) and \(v\) are the process and measurement noise vectors.

\(p\) is an internal variable that the filter uses to store the components that describe the systems current state. For example, if we were to model a particle with constant velocity, the position and velocity would be candidates of variables to enter into the state vector. \(z\) contains the observable variables, in the example with the particle, say that we can only measure the position of the particle but not it’s speed. In that case \(z\) would only contain the position of the particle.

The matrices \(F\) and \(H\) model how the old state vectors affect new ones and how they relate to the observation vector. \(w\) and \(v\) represent how well the model represents the system and how accurate the measurements are.

In every time step the Kalman filter will try to predict the next time step, take in an observation and use the error between the prediction and measurement to update it’s hidden variables to improve future predictions. The hidden variables are the state vector and it’s covariance matrix.

In this thesis we model the targets position as a particle with constant velocity to be able to predict the targets movement during occlusion.

2.8 Genetic Algorithms

As described in Darell Whitley’s comprehensive genetic algorithm tutorial [39] the name genetic algorithms (GA) refers to a family of computational models inspired by evolution. GAs are mainly used as function optimizers, although the models can be applied to a wide array of problems. An implementation of a GA begins with a generation of a population of chromosomes. A chromosome in this
context is one attempt the solution. The chromosomes are then evaluated according to a fitness function, which puts a score on how well the chromosome solves the problem at hand. The fitness of a given chromosome is then used to determine its opportunity to reproduce into the next generation of the population. There is also an element of mutation, where some offspring change in ways that are not defined by what parents it had. This is to combat the problems of local minimums.

This is an iterative method that after generating and filtering population after population, will aim to improve at manifesting a type of chromosome that provides high fitness values. A visualization of this workflow is shown in Figure 2.12. This is all predicated on the user designing a fitting algorithm. There is no guarantee on the algorithm find a good solution if the problem formulation is ill-formed. Chromosome design, fitness function, reproduction and mutation
strategies all heavily play into the success of a genetic algorithm. In the case of this thesis, where genetic algorithms are used for hyperparameter search, the chromosomes are sets of parameters and the fitness function is based on how a tracker performs when it uses these parameters. During reproduction and mutation, the new parameters are created in different ways depending on if they are represented by continuous values or binary decisions. More on this topic in the Method chapter.

2.9 Related Work

In this section an overview of the work done in the fields of tracking and deep learning both in the RGB and RGB-D domains.

2.9.1 Tracking in RGB and RGB-D

In RGB-sequences, discriminative correlation filters have demonstrated huge success. The fundamental idea is to create a filter that produces a high response when applied on the target, but a low response when applied to background. The approaches discussed below seek to create filters that output a Gaussian with a peak around the center of the target object. They also update their filters online to adjust to object appearance changes.

In Bolme et al. [5] the MOSSE update scheme was introduced, where the filters were updating by minimizing the sum of squared errors between the output of the filters and a desired Gaussian output. This updating scheme also utilized the DFT domain to conduct updates using only point-wise operations. In Henriques et al. [21] KCF and DCF were introduced and here the updating strategy was generalized to multiple feature channels and it was shown that the usage of circulant matrices and kernel tricks could be used to update non-linear filters fast as the linear filters. Danelljan et al. extended the DCF strategy in several ways. The first was with scale-estimation [12] using scale space domain filters. In [9], weight penalties were applied on filter coefficients corresponding to the background, thus enabling training on image patches larger than the current target patch. In [10], the features and filters were transferred to a continuous domain to enable fusion between features of different resolution, this is especially useful for fusing deep features from different layers. In [17] Gladh et al. investigate the fusion of deep appearance and motion features.

In the RGB-D domain tracking, DCF/KCF is still the most common approach when using correlation filters. In [20], HOG features from both depth and color are used. They also used the depth data in several ways to aid the tracker to estimate scale changes and to detect occlusion. They also use it to efficiently segment the target from the surroundings. The currently best performing approach to RGB-D tracking are particle filters. On the Princeton Tracking Benchmark (PTB) [34], the top trackers are 3D-T [4] and OAPF [32]. 3D-T is a part-based sparse tracker where both the motion and appearance model are formulated in 3D. OAPF is a 2D particle filter was adopted in which the sampling variance of
2.9 Related Work

each individual particle changes according to its occlusion state, thus becoming an occlusion aware particle filter.

2.9.2 Deep Learning in Tracking

Deep learning has recently demonstrated a huge success in the field of visual tracking. In VOT 2015 3% of trackers used a CNN architecture for either a localization or for feature extraction [27]. This percentage rose to 20% and then 31% in the next two years [29] [28]. Approaches such as MDnet [37] and SiamFC [2] train their networks to output the location of the target. MDnet outputs a bounding box and SiamFC returns a score map where the position of maximum score is used as the estimated translation. The other approach for incorporating CNNs is to use them as feature extractors. In CFCF [18] a network is attached to a correlation filter based tracker, and the network is trained by backpropagating the tracking error to the CNN. In [8] features are extracted by passing the input image through the network and extracting the activations from the ReLu layers. This approach is also used in [10], but activations from different layers are transferred to the continuous domain to compensate for their different resolutions. In CCOT and DeepSRDCF deep features were taken from the VGG-M network introduced in [6], specifically chosen due to its small size. Deep networks tend to lose their spatial information [23].

2.9.3 Deep Learning in RGB-D

When applying CNNs to the RGB-D domain, it has been shown that taking other approaches than the naïve one, where the depth channel is simply treated as another color channel, yields good results. Particularly encoding the depth image in different ways to enable transfer learning from networks trained on RGB images. Gupta et al. [19] encode the horizontal disparity, height above ground and angle of a pixels local surface with the inferred gravity direction, they call this HHA-encoding. Eitel et al. [13] use the colors from a JET-encoding of the depth image and slot these into the red, green and blue channel and perform slightly better than the HHA-encoding on their object recognition task. When using depth data, missing data should be taken into consideration. In [40] a network is used to estimate surface normals and occlusion boundaries. Here, a binary mask indicating where no depth was estimated is supplemented with the depth image to help the network separate missing data from actual sensor values.
In this chapter, the methods used to answer the questions posed will be introduced. First the changes to the original SRDCF framework will be presented. After this, attention control is explained. Attention control is the method which attempts to divert the attention of the DCFs from the missing data. Section 3.3 and 3.4 are the sections which show how scale estimation, occulsion detection and recovery after target loss all can be done using the information from the scene depth. Section 3.5 describe the CNN architectures used for deep feature extraction. The last section is the hyperparameter search section. This section describes how genetic algorithms were used to find the set of optimal parameters, and at the same time providing insight into the quality of each tracker.

The two trackers presented in this thesis are called Depth Cluster Analysis Discriminative Correlation Filters (DCA-DCF) and Depth Cluster Analysis Spatially Regularized Discriminative Correlation Filters (DCA-SRDCFX). DCA-DCF is the tracking framework based on the multi-channel DCF with the modules presented in this chapter and similarly for DCA-SRDCFX. The addition of X to the name is to indicate that there were modifications of the SRDCF formulation, which are presented in the first section.

### 3.1 Modifications to SRDCF

Some modifications were done compared to the original SRDCF formulation. In the original paper the proposed solver for the equations in Equation 2.15 was a Gauss-Seidel solver. In the repository managed by the authors of SRDCF [9], the implementation of SRDCF currently uses a Preconditioned Conjugate Gradient solver. Therefore the decision was made to replace the Gauss-Seidel solver with a CG sparse solver. Another modification that is inspired by Danelljan et al. is that in their tracker ECO [11], they find that updating the filter is not necessary in
every frame. This due to target appearance rarely changing fast enough to make updates in every frame necessary. Therefore a hyperparameter controlling the update frequency is added.

In the original SRDCF formulation a unitary matrix $F$ is created that enforces Hermitian symmetry upon the filter to enable faster convergence. In the implementation used in this paper, this matrix is never created due to being replaced by the real-valued Fourier transform. This transform only computes the right-hand side frequency spectrum for real-valued signals. The left-hand side is then implicitly considered due to the prior knowledge. This reduced the computational load due to only half the amount of memory being used. Another difference is that in SRDCF, the first filter is trained with a starting point that is based on solving an equation system with the same sparse coefficients as the desired filter. This system would also be invertible and therefore solvable with a direct sparse solver. However, during the experimentation of this thesis, the matrix inversion became prohibitively slow to do. Therefore this starting point was not included.

### 3.2 Attention Control

The inclusion of attention control was inspired by the work of [38], due to the perfect fit of the normalized convolution formulation in the context of RGB-D tracking. There is a big issue concerning RGB-D tracking is the handling of the depth data and the missing data that comes with it.

However, the implementation of this into the DCF framework is not trivial. If we are to replace convolution with the normalized version in the DCF, one of two conditions is broken, depending on the interpretation. Either the applicability weights become negative, breaking the non-negative rule, or the subspace that is defined by the base vectors will not contain the DC-component. Using the second interpretation, we could add the DC-component to the base vectors to enable proper filtering and we move into normalized differential convolution. The issue here is that this will require a calculation of a fairly large matrix inverse compromising real-time performance. The other option is to enforce positive filter-coefficients in some capacity.

What was done in this thesis was investigate what happens if these requirements are relaxed. For every frame the trained filter is applied in standard DCF fashion on the masked feature map. The filter is also applied as a normalized convolution in the same way as in [38]. The peaks are compared and if they aren’t far away from each other, the normalized convolution response is chosen as the one that decides the estimated translation. If they differ a lot then the one that estimated the shortest translation is chosen.
3.3 Scale Estimation

To accomplish scale estimation, the target is separated from its surroundings by clustering using Bayesian Gaussian Mixture Models (BGMM). Its models are trained in the first frame and iterated upon in subsequent frames. In DS-KCF [20], k-means clustering is used to estimate a Gaussian around the target. This is very reliant on the hyperparameter k, which determines how many clusters to estimate. To avoid this issue in this thesis, BGMM are used instead, since with variational inference and the Dirichlet prior, the clustering has the flexibility to not use Gaussians it deems unnecessary to describe the data.

The clustering is done on the depth histogram of the target and its surroundings. A large patch is extracted, to give the clustering more training data to hopefully better distinguish between target and background. The training patch can be seen in Figure 3.1a and the resulting histogram in Figure 3.1b. A couple of initializations are done, to become robust against the randomness, and the initialization with the highest lower bound value on the likelihood is kept. The initializations are done by getting the weight concentration of each model from a Dirichlet process and the means and covariances from a K-means clustering. The estimated Gaussians can be seen in Figure 3.1b. Since this is the first frame it’s assumed that the target will not be under occlusion and thus the target will be the object in the middle of the initial bounding box. To be robust against noise, a small area is extracted around the center pixel and the most occurring cluster in this area is chosen as the target cluster. The application of the clustering to the depth image can be seen in Figure 3.1c. Depth segmentation is done by masking all pixels with a depth-value 2 standard deviations from the mean of the target cluster. A connected component analysis is done to find the blob that belongs to the target, this to separate the target from other objects at the same depth. The standard deviation of the depth image within this extracted mask is then saved as the target’s actual depth variation. This connected component analysis can be seen in Figure 3.1d.

The quality of the initial clustering is also checked by measuring the bounding box dimensions of the segmented target with the ones of the initial ground truth bounding box. If these differ too much, the clustering is considered a failure and is redone with adjusted initial parameters. The flowchart for the depth analysis in the first frame is shown in Figure 3.2.

In every subsequent frame, the patch around the estimated position target is extracted and clustering is updated with the old clustering as a starting point. The depth segmentation is then done and if no occlusion is detected, the measurements of the target will be updated. The update of the measurements is done by analyzing the standard deviation of depth within the mask and also the dimensions and the location of the centroid. If the changes are acceptable, the dimensions are updated and the tracker proceeds.
3 Method

(a) The training data depth patch for the first clustering

(b) The depth histogram of the training patch together with the estimated Gaussians. The Gaussian with the lowest distance is the target. The Gaussian with the most samples is the one corresponding to the background. Distance is in millimeters.

(c) The target patch labeled with the clustering. Black is the label for missing data. Here the cyan cluster is chosen as the target cluster

(d) Connected component analysis on the target cluster. It is here divided into a turquoise and a yellow blob. The turquoise blob is chosen as the target blob and its measurements are shown above the image.

Figure 3.1: Images visualizing parts of the depth analysis initialization.
3.4 Occlusion Detection and Handling

Two novel forms of occlusion detection are used in this thesis. One will be called active occlusion detection (AO) and the other will be called passive occlusion (PO) detection. Both use the same depth segmentation method as in the scale estimation. When these methods detect occlusion they both launch the same occlusion-handling method. The naming is the show the ordering of the analysis of the depth segmentation. In active occlusion the analysis is actively done before the application of the filter, as opposed to the passive version which passively waits until after the application of the filter.

The occlusion recovery method is designed to find possible recovery candidates for the tracker. This method is also used when the tracker loses its target. The method uses analysis of the depth data and a Kalman filter to find possible

---

**Figure 3.2:** A flowchart of the depth analysis initialization done for the first frame. If the dimension-check fails, the parameters are tuned and then the clustering is attempted again.

If these measurements increase too much then this might be a sign of the Gaussians trying to expand to incorporate another object under the target cluster. This is undesirable and preemptive measures are taken. The models are constrained by increasing the weight concentration prior to the Dirichlet process and a reinitialization is done. This forces another cluster to take over the object and hopefully preserves the cluster that corresponds to the target. If the newly initiated model provides a good segmentation of the target, the new model is kept, measurements are updated and the tracker proceeds. If all above fail, the measurements are not changed for that frame. In Figure 3.3 the flowchart for the depth analysis that enables scale estimation in an active occlusion context, which will be introduced in the forthcoming section.
Figure 3.3: The flowchart for the depth analysis that enables the scale estimation. It is here shown in an active occlusion context. The signature of Active Occlusion is that the depth analysis is done before the DCF application.
locations for the tracker to find its target. This is method is thoroughly described in subsection 3.4.4.

### 3.4.1 Active Occlusion

In short, AO detection will try to anticipate when an occlusion will happen. On a new frame, the occlusion detector will use depth segmentation from GMMs to detect any object in front of the target. If there is an object close to the camera, the tracker will still evaluate the patch with the filter, but if the response is poor, occlusion handling mode will be engaged.

Occlusion is detected by analyzing the size change of the target cluster. If the target cluster has shrunk too fast, a segmentation is done of everything in the image patch with a depth value smaller than that of the target. If this segmentation reveals a sufficiently large occluding object, an occlusion awareness flag is raised. This means that a tighter restriction is put on the response of the filter when it is evaluated. If the next tracker response is of poor quality the occlusion is confirmed and occlusion handling is engaged. A flowchart for this process can be seen in Figure 3.3. It should be noted that the AO detection flowchart is not a separate independent module from the depth analysis as they are tightly interwoven operations.

### 3.4.2 Passive Occlusion

PO first lets the filter calculate the response from the filter. It then validates the estimated position through depth segmentation to estimate whether the new position can be trusted. This is done by the same techniques used in AO. If the validation fails, the target loss algorithm is started. The motivation for passive occlusion is that the quality measures of the response are not always reliable. During initial testing the output from DCA-SRDCFX was of good quality even when the target was partly occluded. This throws a wrench into the machinery of AO, that depends on the output being very poor for occluding objects. PO therefore bypasses this check and puts the responsibility of occlusion detection entirely on the Kalman filter and the depth clustering. The flowchart for this is shown in Figure 3.4. Same as for AO detection, the process for PO can’t be shown without showing it’s interaction with the depth analysis.

### 3.4.3 Response Quality

Aside from occlusion, candidate search can be launched when the response from the tracker is very poor. A rolling average of the peak value of the response and a quality measure of the response is kept. If the quality measures drop way below these expected values, the tracker launches a candidate search to try to recover. The rolling average of the peak value is fairly simple

\[
R^0_{avg} = 0
\]

\[
R^t_{avg} = \lambda y(u_{max}, v_{max}) + (1 - \lambda)R^{t-1}_{avg}
\]  

(3.1)  

(3.2)
Figure 3.4: Depth analysis in the context of Passive Occlusion detection. The signature of Passive Occlusion is that the depth analysis is done after the application of the DCF application.
where \( R^k_{\text{avg}} \) denotes the rolling average at time \( k \), \((u_{\max}, v_{\max})\) is the coordinate in which the response \( y \) has its maximum value and \( \lambda \) denotes the weight applied to a new observation.

The quality measure is a bit more complicated. It was proposed by Bhat et al. [3] to weight multiple filter responses from different features. In this thesis however it is just used to determine the quality of a response.

Consider the function
\[
\Delta(u, v) = 1 - e^{-\kappa(u^2+v^2)},
\]
(3.3)
It’s a function which has it’s minimum at the origin and approaches it’s maximum of 1 as \((u^2 + v^2)\) increases depending on the value of \( \kappa \). This function will be used as a sort of weighting function in the quality measure.

The quality measure is as follows
\[
\xi = \min_{u^*,v^*} \{ \Xi(u^*,v^*) \} = \min_{u^*,v^*} \left\{ \frac{y(u_{\max},v_{\max}) - y(u^*,v^*)}{\Delta(u_{\max} - u^*, v_{\max} - v^*)} \right\}
\]
(3.4)
where \( \xi \) is the quality measure, \((u^*, v^*)\) are coordinates that minimize the target function \( \Xi \) and \( y \) is the response. The function aims to find the coordinates that reaches the highest value of \( y \), while also being far away from the global maximum in \((u_{\max}, v_{\max})\). \( \Xi \) approaches its maximum value of 1 when \((u^*, v^*) \rightarrow (u_{\max}, v_{\max})\). \( \Xi \) has low values if there exists other coordinates of similar values to the global maximum far way from the maximum. The quality measure \( \xi \) can then be used to describe the quality of the response. A quality measure \( \xi \) close to 1 indicates that the maximum is unique, while a low value shows that it isn’t.

A rolling average of the quality \( \xi \) is kept in the same way as the peak value of the response. The reasoning behind not choosing an arbitrary threshold beforehand is that both quality and peak value differ a lot from sequence to sequence.

### 3.4.4 Occlusion Handling/Candidate Search

The occlusion handler uses depth segmentation and a Kalman filter to output candidate regions that the tracker is evaluated upon. If the tracker outputs a response of high enough quality, the tracking will resume at that position. The candidate regions are extracted by searching at the predicted depth, predicted position, last-known depth and at the last-known position. The predicted positions are from the Kalman filter.

For the searches based on depth, the depth image is segmented by extracting all connected regions within a distance from the predicted and last-known depth. The connected regions are measured to verify that their size is big enough to contain the target and if they are, they are added to the candidate regions. For the searches based on position, the positions are verified by checking if there is anything near the position that has a depth that resembles either the predicted depth or the last known depth. If there is and it is of sufficient size, it is added to the candidate regions.
The filter is then applied to the candidate regions and the one with the highest acceptable response is chosen as the recovery position. If no acceptable region is found, the search is done again next frame.

### 3.4.5 Kalman Filter

In chapter 2 the Stationary Kalman filters were presented in a general sense. In this thesis the model for the system will be a constant velocity model

\[
\begin{align*}
p &= (x, y, z, \dot{x}, \dot{y}, \dot{z})^T \\
p_{k+1} &= Fp_k + w_k \\
z_k &= Hp_k + v_k
\end{align*}
\] (3.5)

\[
F = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\] (3.6)

\[
H = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\] (3.7)

where \( p \) denotes the state vector, \( F \) is the state transition matrix, \( z \) is the observation vector, \( H \) is the output transition matrix, \( w \) is the process noise and \( v \) is the measurement noise. Variables are denoted with subscript \( k \) to indicate the time step. The \( x, y, z \) in the state vector is the estimated centroid of the target and \( \dot{x}, \dot{y}, \dot{z} \) denotes the time derivative of these coordinates. The process and measurement covariance matrices are as follows:

\[
Q = \sigma_p \begin{bmatrix}
\frac{1}{4} & 0 & 0 & \frac{1}{2} & 0 & 0 \\
0 & \frac{1}{4} & 0 & 0 & \frac{1}{2} & 0 \\
0 & 0 & \frac{1}{4} & 0 & 0 & \frac{1}{2} \\
\frac{1}{2} & 0 & 0 & 1 & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 & 1 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 1
\end{bmatrix}
\] (3.8)

\[
R = \sigma_m I
\] (3.9)

where \( \sigma_p \) is a parameter that corresponds to the propagated error, \( \sigma_m \) corresponds to the measurement error and \( I \) denotes the identity matrix.
3.5 Deep Features

To be able to obtain deep features, a trained CNN for a problem in the RGB-D domain is required. This was acquired by training CNNs for classification on the cropped object images from the Washington RGB-D Object Dataset [30]. Two single modal networks and three multi-modal networks were tested. The names and short descriptions for the architectures can be seen in Table 3.1.

The networks that are trained on raw depth data are supplied with a missing data mask. The mask is a binary mask indicating which pixels that contain missing data. This mask is simply coupled with the depth data, making the input $W \times H \times 2$ dimensional. This is inspired by [40], where a network was trained for filling in missing data. The designs of the networks take heavy inspiration from VGG-M [6]. The reason being is that VGG-M retains spacial structure as the image is passed through and compressed in the network, which is a desirable characteristic for the features that will be extracted. The complete architectures can be seen in Figure 3.5 and Figure 3.6.

As mentioned in the subsection 2.9.3, using jet encoding has yielded better results for RGB-D deep learning than encoding other geometric information in the depth image. It also enables transfer learning, but this possibility is not used in this thesis.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StynsNet</td>
<td>Single-modal network on depth data</td>
</tr>
<tr>
<td>AusechaNet</td>
<td>Single-modal network on RGB data</td>
</tr>
<tr>
<td>FusNet</td>
<td>Multi-modal network that fuses RGB and Depth feature extraction</td>
</tr>
<tr>
<td>SumNet</td>
<td>Same as FusNet, but fuses with summation rather than concatenation</td>
</tr>
<tr>
<td>JetNet</td>
<td>Multi-modal network that fuses RGB and Jet-encoded depth extraction</td>
</tr>
</tbody>
</table>

Table 3.1: A table describing the different CNN architectures tried. The single-modal networks only use one type of input data, while the multi-modal ones use both color and depth information. All the architectures mimic the structure of VGG-M [6].
Figure 3.5: The architectures for the multi-modal networks, FusNet, SumNet and JetNet.
Figure 3.6: The architecture for the single-modal networks AusechaNet and StynsNet.
3.6 Hyperparameter Search

Due to the hyperparameters having a very large impact on the performance of the tracker, a parameter search was determined to be an important element to verify the quality of the strategies and modules employed. Strategies for tuning these parameters has not gotten a lot of traction in the literature and the norm seems to either grid or random search. The parameter search in this thesis was done in a novel way, using a genetic algorithm strategy where the chromosomes were defined as a set of parameters. Aside from the improvement of tracking performance, it also enables a new way of evaluating tracking modules by analyzing the population of chromosomes.

The fitness function was set to be the area under curve (AUC). This evaluation metric for trackers is introduced properly in subsection 4.3.1 in chapter 4. Reproduction was done by randomly picking parents from the top performers and using the parameter values from the parents as bounds for the value of the offspring. The offspring’s obtained value will thus be a random sample from the uniform distribution defined by these bounds. To not get stuck in local maxima a mutation event was allowed to occur. When an offspring is generated, there is a 2.5% chance that a property of the child will obtain a mutation. This parameter is then nudged randomly, independent of what bounds that were imposed by the parents. After a new generation of offspring were created, they were all evaluated and ranked on the same list as the parents. If the population size is 64, then this means that in this step we are evaluating a set of new chromosomes and checking if any perform well enough to be ranked in the top 64. These top 64 are now the pool of chromosomes that have a chance of reproducing into the next generation.

Hyperparameters were optimized for each of the handcrafted features that were presumed to need similar parameter sets. Such sets of features were the "FHOG-group", which trained the parameters for FHOG, DFHOG and a concatenation of both at the same time. The second grouping was the "color-group"-features which trained combinations of RGB, jet, colornames and depth clusters. The combinations were limited to one color feature and one depth feature. Due to hardware restrictions it was not practical to run these searches for deep features. Deep features were optimized through a grid search, guided by the results obtained from the search of the handcrafted features. These hyperparameter searches were done on the 5 validation videos in the PTB [34]. For DCA-SRDCFX, which comparatively to DCA-DCF is a lot slower to run, a calibration set was created. This calibration set consisted of 2 of the videos from the validation set, for which the hyperparameter search was ran for a couple hundred iterations before it was continued on all of the videos in the validation set. This enabled the search to weed out the worst chromosomes quicker and then fine-tune the search on the bigger set of videos. Since DCA-DCF was faster to evaluate, a calibration set was not created during its hyperparameter search.
The hyperparameters that were optimized for were:

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCA-DCF</td>
<td>Learning rate, Gaussian variance, search area shape, patch size, lambda regularization, process noise (Kalman), measurement noise (Kalman), feature set and attention control (enable yes/no)</td>
</tr>
<tr>
<td>DCA-SRDCFX</td>
<td>Learning rate, Gaussian variance, search area shape, patch size, process noise (Kalman), measurement noise (Kalman), active or passive occlusion, attention control, update frequency, regularization window parameters, feature set and update iterations.</td>
</tr>
</tbody>
</table>
In this chapter we describe implementation details and evaluation metrics. Then, quantitative result of our proposed methods will follow. In addition, a quantitative analysis is done on the attention control and occlusion handling modules. Lastly a qualitative analysis is done of the trackers, networks and the response quality measures.

As mentioned in chapter 3, the tracking frameworks will go under the names DCA-DCF (Depth Cluster Analysis Discriminative Correlation Filters) for the tracker using the DCF update and the other tracker using the modified SRDCF update will be called DCA-SRDCFx.

4.1 Implementation Details

The entire implementation was done in Python 3.6 utilizing the following libraries:

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numpy</td>
<td>General numeric computation</td>
</tr>
<tr>
<td>Pytorch</td>
<td>GPU-accelerated CNN training and pre-trained models</td>
</tr>
<tr>
<td>Scikit-image</td>
<td>Image processing</td>
</tr>
<tr>
<td>Scikit-learn</td>
<td>Gaussian mixture models</td>
</tr>
<tr>
<td>ImageIO</td>
<td>Image reading</td>
</tr>
<tr>
<td>Matplotlib</td>
<td>Plotting and color mapping</td>
</tr>
<tr>
<td>Scipy</td>
<td>Sparse matrix linear algebra and equation solvers</td>
</tr>
<tr>
<td>PyVision</td>
<td>Genetic Algorithms</td>
</tr>
</tbody>
</table>

FHOG features were extracted using the implementation done by Daniel Mataruana [31].
4.2 Datasets

This section will introduce the datasets used for the evaluation of trackers and training of CNNs.

4.2.1 CNN Dataset

For the training of the convolutional neural networks, the Washington RGB-D Object Dataset was used. The dataset contains several different subsets of images for different computer vision tasks. For the purpose of this thesis, the cropped RGB-D dataset is used.

The images in this dataset were captured by placing an object on a turntable and recording a video of the object spinning. The dataset contains 300 different instances categorized under 51 classes. The image are of different dimensions but all are roughly 100×100 pixels. The images are therefore preprocessed before the input to the network. Following the arguments of [13], simple up/downsampling of the depth image is ill-advised. Due to the existence of missing data and the spatial structure being important, an alternative method of resizing is used. The method scales the image to the desired size by copying the edge column/row and appending these to the image, until the desired size is obtained. This operation is similar to the clamp to edge operation used in computer graphics. The preprocessing step is shown in Figure 4.1. Note that this preprocessing was only done during the training of the CNNs. It was not applied in the tracking frameworks.

Figure 4.1: The preprocessing of depth images for the CNNs. This preprocessing aims to preserve the shape of the depth data.
4.2.2 Tracking Datasets

In comparison to the wide range of high-quality datasets available for RGB tracking, the RGB-D domain lacks in this regard. The dataset used is the Princeton Tracking Benchmark (PTB) [34] by Song and Xiao. They made an early step into RGB-D tracking providing several contributions to the field by introducing the dataset, providing several different trackers and setting benchmark standards. However the benchmark has serious issues with synchronization and registration between the RGB and depth image. Since the two images are unsynchronized, ambiguities arise. This is especially apparent during occlusion, when the target can be occluded in one channel, but not in the other. This is problematic both for the ground truth annotation, but also for any occlusion detection. The current best performing tracker on the PTB [4] uses a modified version of the evaluation dataset, which will be used in this thesis as well. This means that the issues mentioned here are present for the validation videos.

4.3 Evaluation Metrics

In this section the metrics used to evaluate the tracking frameworks, tracking modules and the CNNs are presented.

4.3.1 Tracking Evaluation

The two metrics which are used in the Princeton tracking benchmark [34] will be used. The first of which is center position error (CPE), which is the Euclidean distance between the centers of output bounding boxes and the ground truth. This metric can only be used as a qualitative measure from video to video. The issue with CPE is that it will be undefined whenever the target is entirely occluded. In the benchmark it is possible for the target to be under total occlusion, which results in the ground truth annotation for the bounding box being "no bounding box". While CPE can be used for qualitative analysis, we define P20 as a quantitative metric. P20 is the percentage of frames where the CPE is below 20 pixels. For occlusion cases, a mismatch between the occlusion flag of the tracker and ground truth is considered an error of CPE = \(\infty\) and a correct occlusion estimation is considered as CPE = 0.

To enable a more detailed error measurement the same metric is used as in the PASCAL VOC challenge. There a modified version of intersection over union (IOU) is used,

\[
    r_i = \begin{cases} 
        \frac{\text{area}(\text{BB}_{Ti} \cap \text{BB}_{Gi})}{\text{area}(\text{BB}_{Ti} \cup \text{BB}_{Gi})} & \text{if } \text{BB}_{Ti} \text{ and } \text{BB}_{Gi} \text{ exist} \\
        1 & \text{if neither exist.} \\
        -1 & \text{otherwise.} 
    \end{cases} 
\]  

where \(\text{BB}_{Ti}\) & \(\text{BB}_{Gi}\) denotes the bounding box from the tracker and the ground truth respectively. Here a minimum acceptable overlap area can be set \(r_t\) and we can calculate the average success rate for the tracker as follows:
Experiments

\[ R = \frac{1}{N} \sum_{i=1}^{N} u_i \quad \text{where } u_i = \begin{cases} 1 & \text{if } r_i > r_t \\ 0 & \text{otherwise} \end{cases} \tag{4.2} \]

where \( N \) is the number of frames in a given sequence. This success rate is usually concatenated into a curve that plots \( R \) against the threshold \( r_t \). This curve is summarized into a single number, that is the integral of the curve, called area under the curve (AUC).

### 4.3.2 Tracking Module Evaluation

It is very difficult to evaluate a single module or part of the tracking framework, but through the hyperparameter optimization with genetic algorithms it is observable in the population of chromosomes whether or not the chromosomes enabling the modules survive in the long run. It can also be determined if the module is both enabled and disabled in the population that it has no significant impact on the tracking. The evaluation of attention control and active occlusion (AO) or passive occlusion (PO) detection is done by analyzing the final population of the genetic algorithms. Attention control was evaluated both tracking frameworks, while AO/PO was only evaluated on DCA-SRDCFX.

For the evaluation of the response quality analysis, only a qualitative analysis of how the quality measures behave for both of the trackers during a sequence with occlusion and one with no occlusion.

### 4.3.3 CNN Evaluation

When training the networks, a random subset of images were chosen such that 30% of the images in every class was selected as a validation set that was not shown to the network during training. This validation set is what the network accuracy was measured on.

These error measurements give us confidence in whether or not the network is performing well for its original purpose. However this will not tell us much about the quality of the features in the context of tracking. A good classification score does not necessarily imply that the features will have enough discriminative power for the DCF to be able to discriminate between the tracking target and the rest of the scene.

For evaluation, analysis of the confusion matrix will be done. A given index \( i, j \) in the confusion matrix \( C \) contains the amount of times that the network predicted class \( j \) an image of a class \( i \). The number of times the network correctly classified a class will appear on the diagonal. The accuracy of the network can be calculated by the following formula

\[ \text{Accuracy} = \frac{\sum_{i=j} C_{i,j}}{\sum_{i,j} C_{i,j}}. \tag{4.3} \]

In other words, summation over the diagonal of \( C \) and dividing over the sum of the entire matrix \( C \). This measure is also known as the top-1 error. From
4.4 Quantitative Results

This section will present the quantitative results from the CNNs, tracking frameworks and lastly the tracking modules.

4.4.1 Convolutional Neural Networks

The confusion matrices for all the networks can be found in Appendix A. The accuracies are summarized in Table 4.1. The networks were trained with batch sizes of 32 and with the Adam method for optimization [25]. All were trained for 30 epochs.

All of the networks achieved close to a perfect score on the evaluation set. The fusion networks, SumNet, JetNet and FusNet, performed better than the single channel networks, AusechaNet and StynsNet. Comparing the confusion matrices for the single channel networks, it can be shown that the color network, AusechaNet, had almost all its misclassifications when classifying "caps" as "food bags". The worst performer was the depth network, StynsNet, which was more generally confused with errors all over the span of classes.

It's difficult to see any kind of pattern in the confusion matrices of the fusion networks. All of them perform well, with a few misclassifications spread across a few classes. These misses will be shown in the qualitative analysis.

4.4.2 Tracking

For each tracker, the top 4 performing hand-crafted feature sets and the best deep feature on the validation set are summarized in Table 4.2 and Table 4.3. The complete AUC and P20 curves for each feature set tested can be seen in the Appendix B. Only the results from the best known set of parameters will be presented here. The overall best result comes from DCA-DCF running on DFHOG.

<table>
<thead>
<tr>
<th>Net</th>
<th>Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>StynsNet</td>
<td>99.56 %</td>
</tr>
<tr>
<td>AusechaNet</td>
<td>99.94 %</td>
</tr>
<tr>
<td>FusNet</td>
<td>99.98 %</td>
</tr>
<tr>
<td>SumNet</td>
<td>99.98 %</td>
</tr>
<tr>
<td>JetNet</td>
<td>99.98 %</td>
</tr>
</tbody>
</table>

Table 4.1: The classification accuracies from the networks. AusechaNet performed the best of the single-modal networks and the multi-modals were basically equal in performance.

analyzing the confusion matrix it's possible to see which classes the network has a hard time differentiating between. From the top-1 error the overall performance of the network can be determined.
Experiments

<table>
<thead>
<tr>
<th>Tracker: DCA-DCF Feature set</th>
<th>Validation videos</th>
<th>Validation videos</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bear_front_zcup_move_1</td>
<td>child_no_1_zcup_move_1</td>
</tr>
<tr>
<td>dfhog AUC P20</td>
<td>74.5 84.6 57.3 65.9</td>
<td>81.3 100</td>
</tr>
<tr>
<td>dfhog+fhog AUC P20</td>
<td>73.2 80.8 56.9 68.9</td>
<td>77.9 100</td>
</tr>
<tr>
<td>cn+jet AUC P20</td>
<td>75.3 87.5 51.7 50.6</td>
<td>78.4 100</td>
</tr>
<tr>
<td>cn+gmm AUC P20</td>
<td>73.5 77.9 55.2 59.8</td>
<td>79.0 100</td>
</tr>
<tr>
<td>jetNet both AUC P20</td>
<td>42.1 47.7 59.2 92.0</td>
<td>71.2 98.1</td>
</tr>
<tr>
<td></td>
<td>face_occ new_ex_occ4</td>
<td>Average</td>
</tr>
<tr>
<td>dfhog</td>
<td>83.7 98.2 65.4 72.6</td>
<td>76.5 90.0</td>
</tr>
<tr>
<td>dfhog+fhog</td>
<td>83.9 97.9 62.1 66.7</td>
<td>74.9 89.2</td>
</tr>
<tr>
<td>cn+jet</td>
<td>84.9 98.7 57.3 74.5</td>
<td>74.9 88.9</td>
</tr>
<tr>
<td>cn+gmm</td>
<td>84.0 98.8 63.2 88.2</td>
<td>75.2 88.5</td>
</tr>
<tr>
<td>jetNet both</td>
<td>82.1 98.8 67.0 92.0</td>
<td>65.6 85.4</td>
</tr>
</tbody>
</table>

Table 4.2: Best performances from DCA-DCF on the validation videos together with the average over all validation videos. cn refers to color name features, gmm for depth clusters, jet for jet encoding and jetNet both refers to JetNet features from the first ReLU layer after the fusion. In bold is the best tracker on each sequence/overall score.

features. The best deep feature for both of the trackers were the features from the first ReLU layer after the fusion in JetNet.

The best performing feature set for each tracker is shown on the PTB evaluation dataset. The PTB evaluation dataset results can be seen in Table 4.4. The benchmark provides results for the different types of videos that exist in the evaluation set, as well as the overall AUC. The exact definition of each category can be found in [34]. The results are shown together with a couple of other trackers that provide good reference points. DSKCF_SHAPE is one of the best correlation filter based trackers on the PTB, KCF is the basic tracker from Henriques et al. [21], Dhog is a tracker from the creators of the benchmark [34] that uses a support vector machine based tracker. All these trackers use some combination FHOG features as input. DSKCF uses a concatenation of FHOG and DFHOG, KCF uses FHOG and Dhog uses DFHOG.

4.4.3 Tracking Modules

The evaluation of the viability of attention control and occlusion handling for DCA-SRDCFX was done through a quantitative analysis of the population of chromosomes generated in the hyperparameter search. For evaluation of the response quality measure, refer to subsection 4.5.2. The population size was set to 64 and the mutation rate was set to 2.5% for any individual parameter in the chromosome. The DCA-DCF searches generated around 1000 chromosomes. DCA-SRDCFX searches generated around 400 chromosomes on the calibration
### 4.4 Quantitative Results

#### Validation videos

**Tracker:**

<table>
<thead>
<tr>
<th>DCA-SRDCFX</th>
<th>Validation videos</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Feature set</strong></td>
<td><strong>bear_front</strong></td>
</tr>
<tr>
<td></td>
<td><strong>AUC</strong></td>
</tr>
<tr>
<td><strong>cn</strong></td>
<td>61.9</td>
</tr>
<tr>
<td><strong>dfhog+fhog</strong></td>
<td>68.7</td>
</tr>
<tr>
<td><strong>cn+jet</strong></td>
<td><strong>71.3</strong></td>
</tr>
<tr>
<td><strong>cn+gmm</strong></td>
<td>47.3</td>
</tr>
<tr>
<td><strong>jetNet both</strong></td>
<td>38.2</td>
</tr>
</tbody>
</table>

**Validation videos**

<table>
<thead>
<tr>
<th>Average</th>
<th>face_occ</th>
<th>new_ex_occ4</th>
<th><strong>AUC</strong></th>
<th><strong>P20</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cn</strong></td>
<td>81.8</td>
<td>97.9</td>
<td>52.4</td>
<td>25.4</td>
</tr>
<tr>
<td><strong>dfhog+fhog</strong></td>
<td>77.0</td>
<td>97.0</td>
<td><strong>52.7</strong></td>
<td><strong>43.1</strong></td>
</tr>
<tr>
<td><strong>cn+jet</strong></td>
<td>80.8</td>
<td>98.2</td>
<td>55.0</td>
<td>19.6</td>
</tr>
<tr>
<td><strong>cn+gmm</strong></td>
<td><strong>81.9</strong></td>
<td><strong>98.4</strong></td>
<td>54.0</td>
<td>13.7</td>
</tr>
<tr>
<td><strong>jetNet both</strong></td>
<td>37.9</td>
<td>7.27</td>
<td>19.1</td>
<td>13.73</td>
</tr>
</tbody>
</table>

**Table 4.3:** Best performances from DCA-SRDCFX on the validation videos together with the average over all validation videos. cn refers to color names features, gmm for depth clusters, jet for jet encoding and jetNet both refers to JetNet features from the first ReLU layer after the fusion. In bold is the best tracker on each sequence/overall score.

#### Results on the 95 evaluation videos in the PTB

**Tracker**

<table>
<thead>
<tr>
<th>Tracker</th>
<th>Target type</th>
<th>Target size</th>
<th>Speed</th>
<th>Occlusion</th>
<th>Motion type</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSKCF_SHAPE [20]</td>
<td>70.9</td>
<td>70.3</td>
<td>73.6</td>
<td>73.9</td>
<td>70.3</td>
<td>76.2</td>
</tr>
<tr>
<td>KCF [21] [20]</td>
<td>41.8</td>
<td>50.4</td>
<td>64.9</td>
<td>48.4</td>
<td>54.7</td>
<td>65.0</td>
</tr>
<tr>
<td>DCA-DCF dfhog+fhog (ours)</td>
<td>49.7</td>
<td>48.1</td>
<td>53.9</td>
<td>51.0</td>
<td>50.9</td>
<td>51.9</td>
</tr>
<tr>
<td>Dhog [34]</td>
<td>43.3</td>
<td>48.3</td>
<td>55.9</td>
<td>47.2</td>
<td>50.3</td>
<td>52.7</td>
</tr>
<tr>
<td>DCA-SRDCFX dfhog (ours)</td>
<td>33.1</td>
<td>47.3</td>
<td>41.7</td>
<td>42.6</td>
<td>36.5</td>
<td>47.0</td>
</tr>
</tbody>
</table>

**Table 4.4:** Results on the 95 evaluation videos in the PTB. DCA-DCF and DCA-SRDCFX are shown together with other comparable tracking frameworks evaluated on the PTB. The best performer in every category is shown in bold. The best performer is DSKCF [20].
Experiments

Nr of no att. cont. chromosomes 938, Nr of att. cont. chromosomes 62

Ranking

DCA-DCF fhog-set: All generated chromosomes ranked by AUC
Att. cont. disabled
Att. cont. enabled

Nr of no att. cont. chromosomes 818, Nr of att. cont. chromosomes 182

Ranking

DCA-DCF color-set: All generated chromosomes ranked by AUC
Att. cont. disabled
Att. cont. enabled

(a) All generated chromosomes in the DCA-DCF FHOG-group search divided by usage of attention control.

(b) All generated chromosomes in the DCA-DCF color-group search divided by usage of attention control.

Figure 4.2: Population analysis of the attention control module when used in DCA-DCF. The figures show all generated chromosomes ranked by their tracking performance. The chromosomes are then grouped by whether or not they enabled attention control. In both populations, the chromosomes with attention control disabled both outnumber and outperform their opposites.

set and afterwards on the entire validation set 800 and 300 for the FHOG-group and color-group respectively. The reason for the smaller population for the color-group was a fast convergence where no improvement was found for several hundred chromosome generations.

Attention control

In the DCA-DCF FHOG-group hyperparameter search, attention control performs worse, which can be seen in Figure 4.2a. In the top half of all chromosomes created, none have attention control enabled. In the color-group search, which can be seen in Figure 4.2b, attention focus performs better. There is a wider spread of attention control performance, but in general it performs worse than when it isn’t enabled.

When running the genetic algorithms for DCA-SRDCFX, attention control chromosomes are better for fhog-features. As seen in 4.2a, there are two chromosomes in the top 100 of the FHOG-group that don’t enable attention control. This can be seen more clearly in Figure 4.3c. The opposite is true for the color-group search. In Figure 4.3b, it is seen that only 3 of the 300 chromosomes used attention control. Even the best chromosome with attention control was not part of the top 64.
4.4 Quantitative Results

(a) All generated chromosomes in the DCA-SRDCFX FHOG-group search divided by usage of attention control.

(b) All generated chromosomes in the DCA-SRDCFX color-group search divided by usage of attention control.

(c) Top 64 chromosomes in the DCA-SRDCFX FHOG-group

Figure 4.3: Population analysis of the attention control module when used in DCA-SRDCFX. Here a big difference is seen between the two feature-groups.
4 Experiments

DCA-SRDCFX fhog-set: All generated chromosomes ranked by AUC

DCA-SRDCFX color-set: All generated chromosomes ranked by AUC

(a) All generated chromosomes in the DCA-SRDCFX fhog group search divided by occlusion detection module used.

(b) All generated chromosomes in the DCA-SRDCFX color group search divided by occlusion detection module used.

(c) Top 64 chromosomes in the DCA-SRDCFX FHOG-group

(d) The result on the calibration set of video for the color-group search.

Figure 4.4: Population analysis of the occlusion detection modules when used in DCA-SRDCFX. Also here a big difference is observed between the two feature-groups.

Occlusion handling

In the FHOG-group, active occlusion was the dominant chromosome in terms of numbers. This can be seen in Figure 4.4a. When taking a closer look at the best performers in the population there are 4 PO chromosomes in the top 64. The opposite of this result happened in the color-group search. Passive occlusion dominated heavily and only one AO chromosome was generated through mutation. This means that all AO chromosomes were filtered out on the calibration set, which can be seen in Figure 4.4d.
4.5 Qualitative Analysis

This section introduces qualitative analysis of the frameworks, modules and networks.

4.5.1 Tracking

DCA-DCF

DCA-DCF performs the best when the depth clustering performs well, such as in "zcup_move_1", shown in Figure 4.5. The clustering succeeded without any issues and since the cup didn't undergo any unusual transformation, the sequence is largely a success. In Figure 4.6, the tracker can be seen handling an occlusion. Due to the synchronization problems in the video, the scale-estimation "fails" right before the occlusion in Figure 4.6b. In the next two frames, the Kalman search area is visualized. The size of the area grows as the Kalman filter makes its predictions without receiving a measurement update. Finally the tracker recovers, all be it with an erroneous clustering in Figure 4.6e.
An example when the tracker fails to correctly track the target can be seen in Figure 4.7. From the start, neither the clustering nor the tracker gets any data aside from the arm and leg of the child. This leads to the clustering not acquiring the full depth range needed to include the entire child in the target cluster. While the DCF seems to have a good idea of where the target is, the depth analysis failed to include the new information provided once the child started turning. This leads to the scale estimation failing.

**DCA-SRDCFX**

An example of when DCA-SRDCFX succeeds can be seen in Figure 4.8. This sequence tests the tracker ability to handle occlusions. In this sequence, the target does not move around, which puts emphasis on the depth analysis. The tracker reliably distinguishes between target and occluding object. A sequence where DCA-SRDCFX fails can be seen in Figure 4.9. The tracker doesn’t learn the appearance fast enough and starts tracking the occluding box. This leads to both the appearance model and the depth analysis to be contaminated with false positives. The contaminated depth analysis can be observed in Figure 4.9d, where the scale estimation fails as the occlusion starts and the contaminated appearance can be seen by the tracker following the box in the next occlusion instance.
4.5 Qualitative Analysis

(a) Teddybear heading towards occlusion

(b) Teddybear hiding its head behind a box. Note the mismatch in RGB and depth.

(c) Teddybear occluded. The Kalman search region is visible as the red area.

(d) Teddybear still occluded. The Kalman search region has grown.

(e) Recovery achieved. Initial scale estimation is faulty.

Figure 4.6: Example of occlusion detection and recovery from DCA-DCF. The green box is the ground truth, the red box is the output from the tracker. The dot in the middle of the red box represents the Kalman filter. When the dot is blue it shows the observed position and red it represents the predicted position. The dot grows with the prediction uncertainty.
Experiments

(a) Child video started. The child is facing sideways.

(b) The child started walking.

(c) It becomes evident as the child turns, that the tracker learned to follow the child’s shoulder + leg.

(d) The target cluster didn’t grow to include the entire child.

Figure 4.7: Fail case for the DCA-DCF. The cluster was unable to incorporate the entire child and the tracker didn’t learn the entire object appearance.
4.5 Qualitative Analysis

(a) Sequence start

(b) The start of the occlusion.

(c) The tracker initiates occlusion mode before ground truth, which is why only the green box is visible.

(d) Correctly handled frame.

(e) Ground truth says that occlusion has ended.

(f) Recovery with faulty clustering.

Figure 4.8: A sequence from where DCA-SRDCFX performs well. Occlusion detection and recovery are off by couple of frames.
66

4 Experiments

(a) Sequence start

(b) The tracker fails to detect occlusion.

(c) The tracker re-attaches to the target later in the sequence.

(d) Correctly handled frame.

(e) A new occlusion starts and the scale estimation is off.

(f) The tracker re-attaches to the occluding box.

Figure 4.9: A sequence from where DCA-SRDCFX fails. Almost all modules in the tracker fail.
4.5.2 Tracking Modules

The evaluation of the response quality measures was done through a qualitative analysis, for evaluation of attention control and occlusion handling, refer to subsection 4.4.3.

By examining the quality measure over time in Figure 4.10 it can be seen that the most reliable quality measure of the two is the peak value of the DCA-DCF response. The $\xi$ quality fluctuates widely and is not reliable by itself. Both of the quality triggers need to activate at the same time for the occlusion handling to activate, a benefit seen in Figure 4.10a. At no point in this sequence does DCA-DCF activate occlusion handling, because the two triggers don’t activate at the same time. In the bear_front video, the occlusion estimation is largely successful.

In the DCA-SRDCFX case, the peak response behaves differently throughout the video, which can be seen in Figure 4.11. The peak value steadily declines in both videos. In the case of the bear_front, the peak value threshold is not strict enough to catch the occlusions. In regards to the $\xi$-quality measure, it performs as poorly for DCA-SRDCFX as for its DCF counterpart.

4.5.3 CNN Evaluation

While the vast majority of images got classified correctly, it is important to look at the few cases in which the networks failed. For the fusion networks the failures are mostly on the corrupted images in the dataset. The failure cases for FusNet can be seen in Figure 4.12. The other cases seem to be when the depth image contains missing data in areas of importance. In the stapler case in Figure 4.12, there exists missing data all over the actual object. This is likely the reason for the failure. The same behavior is seen in the other multi-modal networks.

AusechaNet, which only saw the color image, failed consistently on separating caps and food bags. Some examples of the fails can be seen in Figure 4.14a. The error probably arose because of the existence of a black food bag in the dataset, which can be seen in Figure 4.14b and Figure 4.14c. None of the other networks experienced this issue, which indicates that it’s easier to differentiate food bags from caps from the depth image.

StynsNet, which only saw the depth image, was the worst performer of the networks. Due to it being color-blind, it is clear that it got confused as soon as objects of similar shape are introduced. Balls, garlic, apples, potatoes and tomatoes are all objects that are difficult to separate when no color information is given. Examples of these failures are shown in Figure 4.13.
Figure 4.10: The quality measures for DCA-DCF over the first 230 frames of two sequences. zcup_move_1 is an example of a sequence with no occlusion. bear_front is a sequence with several instances of occlusion, these occlusion frames are shown as blue dots. The red dots show when the occlusion criteria has been met for each individual measure.
4.5 Qualitative Analysis

(a) Peak response over time in zcup_move_1.

(b) $\xi$ over time in zcup_move_1.

(c) Peak response over time in bear_front.

(d) $\xi$ over time in bear_front

Figure 4.11: The quality measures for DCA-SRDCFX over the first 230 frames of two sequences. zcup_move_1 is an example of a sequence with no occlusion. bear_front is a sequence with several instances of occlusion, these occlusion frames are shown as blue dots. The red dots show when the occlusion criteria has been met for each individual measure.
Figure 4.12: 8 failure cases for FusNet. Over the RGB image the true label is shown. To the right of the color image the corresponding depth image is shown, with the guessed label. Aside from the apple, stapler and toothbrush, the other images have some sort of error where either the color and/or depth image has been corrupted. As a side note, in the last two images the depth preprocessing can be seen clearly when compared to the deformed color image.
Figure 4.13: 8 failure cases for StynsNet. The misclassifications are between similar shaped objects which are not differentiable from the depth data alone.
(a) 8 failure cases for AusechaNet. Although the network was only shown the RGB image, the complementary depth image is shown as in the previous fail case image Figure 4.12.

(b) The food bag that confused AusechaNet.

(c) Another angle of the food bag that confused AusechaNet.

*Figure 4.14*: Images relating to the misclassifications of AusechaNet.
In this chapter the discussion will be held of the results and method of this thesis. During the discussion of the results, answers will be given to the questions posed in section 1.3. The method discussion will critically examine how the questions were answered and how the experiments were conducted. Afterwards, an overview of future work, based on the results of this thesis.

5.1 Results

Tracking Frameworks

From the related work in RGB tracking, it would seem that a SRDCF-based tracker should outperform its predecessor and that features from a CNN should be the best performers. But the new modules that were added to handle the discussed problems of occlusion and scale estimation, interacted better with the multi-channel DCF framework.

The most important part of the detection and the recovery for the trackers is the quality measures of the response. They are the most important factor when determining when these decisions are to be made. For DCA-DCF, the peak response value performed well as a symptom for target loss. This in turn resulted in DCA-DCF handling occlusions better and therefore achieving better results on the PTB. DCA-SRDCFX didn’t have a quality measure that could reliably tell when the target had been lost. As was shown in the qualitative analysis of the quality measures, the measures simply behaves differently for both of the trackers. Using the same rolling average method for both of the trackers was not the best strategy. The DCA-SRDCFX method of solving its system of equations and updating its filters meant that the response did not retain a constant peak value, but rather steadily declined as the sequence progressed.
The passive occlusion detector was created to mitigate part of this issue. Since the quality measures are both used to detect loss of target as well as recovery, the idea was that it is possible to use depth analysis by itself to detect occlusion and afterwards use the filter to recover from the occlusion, but as shown from the hyperparameter search, the gain was only found by the color-group search. The reason could be that the quality measures are not the biggest bottleneck in the fhog-group, but rather something else.

In general the depth analysis methods suffered in quality from not incorporating the appearance model in some way. A large part of the modules in the trackers relied on the results from the clustering of the depth data. In the cases that the clustering fails to separate the target for the surroundings, the trackers suffer heavily. This point of failure is likely the reason that even the best result in this thesis performed worse than KCF without any additions.

Attention control proved to be difficult to decipher. Keep in mind that the implementation had a safety trigger for the attention control such that if the attention control response diverged, it would in the worse case become a normal convolution. It is also important to note that the attention control module would affect the shape of the Gaussian response, which will affect the quality measures. This could be the reason that DCA-DCF, which had a more reliable measure for its occlusion estimation preferred to not enable the attention control. It would also explain why DCA-SRDCFX, which didn’t complain enough during occlusion, wanted to enable it for the fhog-group of features. The fact that the attention control interacted this unpredictably with the trackers, makes it very difficult to trust this module as an answer to the missing data problem.

Looking towards the results on the PTB and to the other correlation filter trackers, the results were underwhelming. One reason for why the DCA-DCF framework was not able to outperform DS-KCF from [20] is that although this framework enables scale-estimation, the appearance model is not updated to take advantage of this information. In DS-KCF, the appearance model is updated by upsampling and downsampling in the Fourier domain to enable the filter to learn the size change of the target. DCA-DCF does not do this and thus becomes worse at tracking when the target shrinks or grows. DS-KCF also uses peak response to estimate occlusion, it could be that the KCF-framework interacts even better with this quality measure.

**Feature sets**

In the evaluation of feature sets, the results of the top-performers were very close. FHOG features have been proven to work well before, in both RGB and RGB-D tracking. The biggest surprise was how well the jet-encoding of the depth image worked for both of the trackers and depth clusters for the DCA-DCF. The intuition for why these features might work, is based on the same reasoning that color names work so well. The idea of encoding more abstract information about a feature space seems to make correlation filter trackers perform well. Jet-encoding assigns 3 color channels to the more abstract "distance"-concepts of objects that are "near", "middle" and "far". This means that an object that moves around in
the same distance from the camera will retain its "distance"-property in this sense, as opposed to other features that might change their value quickly as the target changes appearance. The same reasoning can be applied to the depth clusters, which aim to encode the property of "target" and "not target" into the probability clusters. The reason that this performed worse for DCA-SRDCFX is probably due to its update in every frame. If the tracker is inaccurate, the clusters will try to fit to other structures in the scene, which weakens the accuracy for the target. This negative feedback loop was then enhanced by the poor occlusion handling of DCA-SRDCFX.

The best performing deep features for both of the tracking frameworks were the first layer after fusion for jetNet, first layer features from VGG-16 and the first layer features from AusechaNet. The jetNet fusion layers come as a big surprise. For some reason, this was the only net that became better after the fusion when compared to its RGB channel. Maybe the denoising of the depth data helped the activations to become easier to fuse with the RGB activations and therefore the gain could be acquired when fusing the channels.

CNN

The performance of the neural networks compared to their performance when employed for tracking can be explained by the limitations of the Washington RGB-D dataset. All the images were taken in an office environment, with very similar overall conditions. This is perhaps the reason that the features from these networks aren’t able to generalize in the same way as the imported VGG-16 network that was trained on ImageNet can.

Research Questions

In summary:

- **What type of input features are good when tracking in RGB-D video?**
  Features calculated from depth data provide rich information about the target. The experiments suggest that FHOG features calculated on the depth image and a jet-encoding of the depth image are the best depth features. Using these features in conjunction with RGB features yield in general the best results, for both tracking frameworks.

- **Can we compensate for missing data in DCF-based tracking using normalized convolution?**
  Normalized convolution with relaxed criteria is not the method that solves this by itself. Here it performed unpredictably depending on the tracker and the input features.

- **Can we estimate the scale of the target using the scene depth?**
  Yes, clustering on the depth histogram enables a fast segmentation of the target which in turn allows for scale estimation.
• *Can occlusion be estimated and handled using the scene depth?*

Yes, using clustering for occluding object detection, Kalman filters for movement prediction and evaluating candidates using the trained filter enables occlusion estimation and handling.

## 5.2 Method

### Tracking

The depth cluster analysis method for occlusion and scale estimation shows promise, but there is a room for improvement. A sophisticated analysis of the depth data, that is robust against the different situations that can arise is needed. Currently there is no interplay between the color information and the depth information while clustering which is a limitation. A common occurrence is that the target object gets coupled with some other objects during the initial frame which contaminates the depth analysis. The very poor clustering, that these cases result in, can severely damage the tracker's performance to the point that it would be better not to use the module at all.

The quality measures are important to the performance of the trackers during occlusion, but the measures proposed in this thesis were not enough. The peak response value shows promise, but will fail when there is something in the scene with a similar appearance to the target. This was the reason that the second measure \( \xi \) was introduced, to get a measure that controlled for the noisiness of the response. This measure was either not used to its full potential in this thesis or another measure should be added to shore up the weaknesses of the peak response value. The concept of a measure that conveys information of the uniqueness of a maximum as well as the curvature of a response is definitely interesting, but \( \xi \) didn't fulfill that role.

Normalized convolution with relaxed conditions was shown to have an unpredictable influence on the tracking frameworks. To be able to answer the question of missing data handling, a deeper look on the normalized convolution formulation needs to be done. It would be very interesting to incorporate the normalized convolution formulation into the loss function of a DCF-framework and formulate an update rule based on this.

Scale estimation was achieved by the depth clustering, but the information gained was not included into the appearance model. This has been done in other works in the field, such as DS-KCF[20], with great success. Even though a simpler clustering method is used in DS-KCF, the update of appearance model allows it to outperform both the trackers used in this thesis.

The genetic algorithm hyperparameter search provided a lot of information to the thesis. It revealed which parameter combinations and which modules that worked the best, which helped to answer the questions posed in this thesis. During the development of the two types of occlusion detection, simple empirical testing of parameters did not manage to get DCA-SRDCFX to work together with active occlusion. Only when the genetic algorithms were employed, this solution was found.
Another consideration is computation. DCA-DCF generated 1000 chromosomes in its search. 6 of its hyperparameters are defined by continuous values. If these values were optimized with grid search, using 5 different values for each parameter, then $5^6 = 15625$ chromosomes would be generated and tested. The grid search would then take 15 times longer to evaluate and depending on how the grid is defined, might not find a good solution.

The genetic algorithm should not be confused with a search for every feature sets optimal settings. In the parameter searches, the algorithm optimized for the dominant chromosome-types in the population. This means that the optimal settings for the feature sets that died out quickly, were not found.

The hyperparameter search was not conducted for deep features due to hardware and time restrictions. This does mean that the conclusions drawn from the results of the deep features are not as reliable as the hand-crafted feature results.

**CNN**

The training of the CNNs was intentionally kept very simple, perhaps even too simple. In other object classification work done on the Washington RGB-D dataset a test set is created by leaving one instance of every class out of the validation and training set. This subset is then used to evaluate the classifiers ability to generalize beyond what it’s already seen. This was not done in this thesis since the goal of these networks were not to become best of classifiers, but rather to provide features for the DCFs. An evaluation of the entire network architecture’s ability to generalize was outside of the scope of the thesis. The assumption was that if the network can correctly classify on unseen images, the features should have reached good enough quality to track on.

Due to the deep features solely being judged based on how they performed in a tracking framework, it makes it difficult to declare any kind of pattern. If some sort of evaluation was done on the discriminative power from the layers of the network was done outside of the DCF filters, then there would be more evidence to help decipher why they performed they way they did. Without this, the analysis becomes a guessing game when the results are inconclusive. Further work could be done into this area to focus entirely on the deep features and how different architectures and training techniques affect the tracking performance.

All the networks tested were classification networks. This is a simple task due to how simple the Washington dataset is. If the networks would have been trained for more difficult tasks such as detection or segmentation, there could have been more variance in the results from the different architectures. In the evaluation of this thesis, all fusion networks achieved close to the same perfect scores. When the results are saturated, it is impossible to make any conclusion about what advantages/disadvantages a certain network architecture has.

### 5.3 Future Work

When it comes to tracking frameworks a lot of advancements have been done in the field since SRDCF was introduced. SRDCF has an extension called SRDCFde-
con, where all past samples are weighted to decontaminate the training from old samples. ECO is currently the state-of-the-art in RGB tracking and applying the same framework in RGB-D would most likely bring RGB-D tracking up-to-date with RGB. As discussed earlier, an appearance model update when a scale change has been detected would also be desirable.

Concerning the handling of missing data, the incorporation of normalized convolution is interesting. Adapting the DCF to use smaller filter sizes and fewer feature dimensions could maybe enable proper normalized differential convolution to be done. If we accept this and let the inference of the filter become more computationally demanding, then some exciting opportunities arise. Firstly, proper attention control would be obtained which could be interesting even for the RGB domain. In RGB-D the certainty mask would still be obtained from the depth data, while in the RGB domain it could be estimated through other means. Normalized convolution would also enable an interesting way of applying the filter on different target sizes. With normalized convolution applying the filter on smaller feature maps can be done by upsampling and then assigning low certainty values to the pixels that were estimated. Normalized convolution is not the only way of incorporating missing data into the tracking frameworks. The current DCF frameworks still use the windowing of the features based on a constant cosine-window. If this window was augmented by the missing data and incorporated into the update of the filter, then the appearance update could achieve a partial update based on the features we see, disregarding the data from background and missing data.

For the training of the networks, there is a lot to be investigated. Detection and segmentation networks are more popular nowadays, which may provide better features. Deep motion features from networks applied to optical flow have been shown to complement normal deep features well, which would be interesting in RGB-D as well. For the training of the network, there are several techniques to combat the smaller datasets, like dropout and data-augmentation. The depth data enables new forms of augmentation where the scene is transformed in 3D-space or where missing data is inserted to make the network more robust. Training networks on bigger dataset in RGB and then utilizing some of the transfer learning enabling techniques discussed in [13] [19] could maybe also improve the quality of the feature extraction.

A look into other ways of evaluating the response of the tracker would be very useful for the occlusion handling proposed in this thesis. A difficult task, but when observing the responses over time during a tracking sequence one can’t help but think that there is a way of quantifying the noise that can be seen. New methods for quantifying this is sorely needed. As for the quality measures used in this thesis, while they show some promise, they are probably not used to their full potential. Especially when it comes to incorporating past quality measures to detect deviations. There probably exists a more sophisticated way than the rolling average that was used. Some sequences tend to be a bit more volatile than others which suggests that a more adaptive and flexible method should be used.
Appendix
The confusion matrices for the CNNs are presented in this appendix. In the figures, a empty blue cell is equal to 0 occurrences. As can be seen in Figure A.1, AusechaNet never misclassified an apple as a ball. Cells with a number and a color indicate the number of times the classification happened. The color becomes warmer, the higher the number of occurrences. For example, caps were misclassified as food bags 29 times by AusechaNet Figure A.1.
Figure A.1: AusechaNets confusion matrix
Figure A.2: StynsNet confusion matrix
Figure A.3: JetNet confusion matrix
Figure A.4: SumNet confusion matrix
Figure A.5: FusNet confusion matrix
The result graphs and parameter settings for all the tested trackers+feature combinations. They are presented in no particular order. Some shorthand aliases are used instead of the names used in the thesis. The aliases used can be seen in Table B.1.
<table>
<thead>
<tr>
<th>Alias</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfhog+fhog</td>
<td>Depth FHOG and Color FHOG</td>
</tr>
<tr>
<td>cn</td>
<td>Colornames</td>
</tr>
<tr>
<td>cn+jet</td>
<td>Colornames and Jet encoding</td>
</tr>
<tr>
<td>clust</td>
<td>Depth clusters</td>
</tr>
<tr>
<td>ausNet</td>
<td>First ReLU layer AusechaNet</td>
</tr>
<tr>
<td>styntsNet</td>
<td>First ReLU layer StyntsNet</td>
</tr>
<tr>
<td>dfhog</td>
<td>Depth FHOG</td>
</tr>
<tr>
<td>fhog</td>
<td>Color FHOG</td>
</tr>
<tr>
<td>fusNet both</td>
<td>First ReLU layer after fusion FusNet</td>
</tr>
<tr>
<td>fusNet depth</td>
<td>First depth channel ReLU layer FusNet</td>
</tr>
<tr>
<td>fusNet rgb</td>
<td>First color channel ReLU layer after fusion FusNet</td>
</tr>
<tr>
<td>jet</td>
<td>Jet encoding of the depth image</td>
</tr>
<tr>
<td>jetNet both</td>
<td>First ReLU layer after fusion JetNet</td>
</tr>
<tr>
<td>jetNet depth</td>
<td>First depth channel ReLU layer after fusion JetNet</td>
</tr>
<tr>
<td>jetNet rgb</td>
<td>First color channel ReLU layer after fusion JetNet</td>
</tr>
<tr>
<td>rgb</td>
<td>RGB color image</td>
</tr>
<tr>
<td>sumNet both</td>
<td>First ReLU layer after fusion SumNet</td>
</tr>
<tr>
<td>sumNet depth</td>
<td>First depth channel ReLU layer after fusion SumNet</td>
</tr>
<tr>
<td>sumNet rgb</td>
<td>First color channel ReLU layer after fusion SumNet</td>
</tr>
<tr>
<td>vgg</td>
<td>First ReLU layer vgg-16</td>
</tr>
</tbody>
</table>

**Table B.1:** Table explaining the aliases in the tracking result graphs.

<table>
<thead>
<tr>
<th>feature set</th>
<th>AUC</th>
<th>LR</th>
<th>( \lambda )</th>
<th>area scale</th>
<th>sigma factor</th>
<th>( \sigma_m )</th>
<th>( \sigma_p )</th>
<th>proportional</th>
<th>attention focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3) colorname+jet</td>
<td>74.47</td>
<td>0.047</td>
<td>0.12</td>
<td>1.2</td>
<td>0.076</td>
<td>31.05</td>
<td>18.9</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>colorname</td>
<td>71.26</td>
<td>0.023</td>
<td>0.23</td>
<td>1.2</td>
<td>0.081</td>
<td>187.8</td>
<td>52.17</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>(1) dfhog</td>
<td>76.70</td>
<td>0.12</td>
<td>0.1</td>
<td>1.2</td>
<td>0.071</td>
<td>82.86</td>
<td>35.71</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>(2) fhog+dfhog</td>
<td>74.90</td>
<td>0.14</td>
<td>0.16</td>
<td>1.2</td>
<td>0.08</td>
<td>126.8</td>
<td>25.53</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>gmm</td>
<td>36.02</td>
<td>0.1</td>
<td>0.14</td>
<td>1.1</td>
<td>0.13</td>
<td>186.9</td>
<td>45.21</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>jet</td>
<td>67.81</td>
<td>0.042</td>
<td>0.12</td>
<td>1.1</td>
<td>0.074</td>
<td>108.3</td>
<td>23.79</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>rgb+jet</td>
<td>60.13</td>
<td>0.075</td>
<td>0.26</td>
<td>1.1</td>
<td>0.058</td>
<td>98.27</td>
<td>14.93</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>rgb</td>
<td>63.63</td>
<td>0.064</td>
<td>0.18</td>
<td>1.2</td>
<td>0.05</td>
<td>30.71</td>
<td>50.93</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

**Table B.2:** DCA-DCF results. Top 3 features are marked with their ranking in parenthesis. Best deep features marked with ranking in parenthesis with asterisk. Parameters are in order, learning rate, regularization lambda, area scale, sigma factor, measurement noise (kalman), process noise (kalman), proportional patch (true for original shape, false for square), attention focus enabled.
Figure B.1: DCA-DCF Result graphs for colornames
Figure B.2: DCA-DCF Result graphs for colormames and jet
Figure B.3: DCA-DCF Result graphs for dfhog+fhog
Figure B.4: DCA-DCF Result graphs for ausechaNet
(a) Area under Curve graph  

(b) Precision under 20 pixels graph  

(c) Area under curve for each video  

(d) Precision graph for each video  

Figure B.5: DCA-DCF Result graphs for stynsNet
Figure B.6: DCA-DCF Result graphs for dfhog
Figure B.7: DCA-DCF Result graphs for fhog

(a) Area under Curve graph
(b) Precision under 20 pixels graph
(c) Area under curve for each video
(d) Precision graph for each video
Figure B.8: DCA-DCF Result graphs for fusNet both
Figure B.9: DCA-DCF Result graphs for fusNet depth
Figure B.10: DCA-DCF Result graphs for fusNet rgb
Figure B.11: DCA-DCF Result graphs for jet
Figure B.12: DCA-DCF Result graphs for jetNet both
Figure B.13: DCA-DCF Result graphs for jetNet depth
Figure B.14: DCA-DCF Result graphs for jetNet rgb
Figure B.15: DCA-DCF Result graphs for rgb
Figure B.16: DCA-DCF Result graphs for sumNet both
Figure B.17: DCA-DCF Result graphs for sumNet depth
Figure B.18: DCA-DCF Result graphs for sumNet rgb
Figure B.19: DCA-DCF Result graphs for vgg16
Figure B.20: DCA-SRDCFX Result graphs for colornames
Figure B.21: DCA-SRDCFX Result graphs for colornames and jet

(a) Area under Curve graph

(b) Precision under 20 pixels graph

(c) Area under curve for each video

(d) Precision graph for each video
**Figure B.22:** DCA-SRDCFX Result graphs for dfhog+fhog
Figure B.23: DCA-SRDCFX Result graphs for ausechaNet
Figure B.24: DCA-SRDCFX Result graphs for stynsNet
**Figure B.25:** DCA-SRDCFX Result graphs for dfhog
Figure B.26: DCA-SRDCFX Result graphs for fhog
Figure B.27: DCA-SRDCFX Result graphs for fusNet both
Figure B.28: DCA-SRDCFX Result graphs for fusNet depth
Figure B.29: DCA-SRDCFX Result graphs for fusNet rgb
Figure B.30: DCA-SRDCFX Result graphs for jetNet both
(a) Area under Curve graph  

(b) Precision under 20 pixels graph  

(c) Area under curve for each video  

(d) Precision graph for each video  

**Figure B.31:** DCA-SRDCFX Result graphs for jetNet depth
Figure B.32: DCA-SRDCFX Result graphs for jetNet rgb
Figure B.33: DCA-SRDCFX Result graphs for sumNet both

(a) Area under Curve graph  
(b) Precision under 20 pixels graph  
(c) Area under curve for each video  
(d) Precision graph for each video
Figure B.34: DCA-SRDCFX Result graphs for sumNet depth
Figure B.35: DCA-SRDCFX Result graphs for sumNet rgb
Figure B.36: DCA-SRDCFX Result graphs for vgg16
<table>
<thead>
<tr>
<th>Features set</th>
<th>AUC</th>
<th>LR</th>
<th>U. interval</th>
<th>U. iter.</th>
<th>area sc.</th>
<th>σ factor</th>
<th>µ</th>
<th>ν</th>
<th>AO</th>
<th>Proportional</th>
<th>σ_m</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3) colorname</td>
<td>63.54</td>
<td>0.0809</td>
<td>6</td>
<td>15</td>
<td>2.18</td>
<td>0.05</td>
<td>0.111</td>
<td>4.0</td>
<td>False</td>
<td>False</td>
<td>98.43</td>
</tr>
<tr>
<td>(1) colorname+jet</td>
<td>67.29</td>
<td>0.104</td>
<td>3</td>
<td>12</td>
<td>2.43</td>
<td>0.05</td>
<td>0.186</td>
<td>4.5</td>
<td>False</td>
<td>False</td>
<td>74.66</td>
</tr>
<tr>
<td>colorname+gmm</td>
<td>57.00</td>
<td>0.0944</td>
<td>6</td>
<td>10</td>
<td>1.66</td>
<td>0.0727</td>
<td>0.05</td>
<td>4.1</td>
<td>False</td>
<td>False</td>
<td>40.35</td>
</tr>
<tr>
<td>redgreenblue+gmm</td>
<td>30.21</td>
<td>0.15</td>
<td>6</td>
<td>7</td>
<td>1.5</td>
<td>0.0548</td>
<td>0.0976</td>
<td>1.7</td>
<td>False</td>
<td>False</td>
<td>10.0</td>
</tr>
<tr>
<td>redgreenblue+jet</td>
<td>29.54</td>
<td>0.015</td>
<td>3</td>
<td>5</td>
<td>2.34</td>
<td>0.0538</td>
<td>0.0921</td>
<td>4.3</td>
<td>False</td>
<td>False</td>
<td>100.0</td>
</tr>
<tr>
<td>(2) fhog+dfhog</td>
<td>63.75</td>
<td>0.00918</td>
<td>6</td>
<td>19</td>
<td>1.51</td>
<td>0.0687</td>
<td>0.141</td>
<td>4.4</td>
<td>True</td>
<td>True</td>
<td>24.98</td>
</tr>
<tr>
<td>fhog</td>
<td>54.50</td>
<td>0.15</td>
<td>3</td>
<td>8</td>
<td>1.5</td>
<td>0.0737</td>
<td>0.2</td>
<td>4.0</td>
<td>True</td>
<td>True</td>
<td>44.26</td>
</tr>
<tr>
<td>dfhog</td>
<td>50.97</td>
<td>0.0873</td>
<td>6</td>
<td>12</td>
<td>1.63</td>
<td>0.069</td>
<td>0.174</td>
<td>3.6</td>
<td>True</td>
<td>True</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Deep Features

<table>
<thead>
<tr>
<th>(*2) vgg16</th>
<th>35.18</th>
<th>Same as colorname</th>
</tr>
</thead>
<tbody>
<tr>
<td>stymsNet</td>
<td>7.02</td>
<td>Same as colorname</td>
</tr>
<tr>
<td>(*1) jetNet both</td>
<td>42.53</td>
<td>Same as colorname+jet</td>
</tr>
<tr>
<td>jetNet depth</td>
<td>15.03</td>
<td>Same as colorname+jet</td>
</tr>
<tr>
<td>(*3) auschaNet</td>
<td>31.79</td>
<td>Same as dfhog</td>
</tr>
<tr>
<td>jetNet rgb</td>
<td>29.28</td>
<td>Same as dfhog</td>
</tr>
<tr>
<td>sumNet both</td>
<td>19.72</td>
<td>Same as dfhog</td>
</tr>
<tr>
<td>sumNet depth</td>
<td>8.11</td>
<td>Same as dfhog</td>
</tr>
<tr>
<td>sumNet rgb</td>
<td>30.62</td>
<td>Same as colorname</td>
</tr>
<tr>
<td>fusNet both</td>
<td>12.90</td>
<td>Same as dfhog</td>
</tr>
<tr>
<td>fusNet depth</td>
<td>10.13</td>
<td>Same as dfhog</td>
</tr>
<tr>
<td>fusNet rgb</td>
<td>34.54</td>
<td>Same as dfhog</td>
</tr>
</tbody>
</table>

Table B.3: DCA-SRDCFX results. Top 3 features are marked with their ranking in parenthesis. Best deep features marked with ranking in parenthesis with asterix. Parameters are in order, learning rate, update interval, update iterations, area scale, sigma factor, mu (reg. window), nu (reg. window), Active Occlusion, Proportional, process noise (kalman), measurement noise (kalman).
Bibliography


ference on Computer Vision Workshop (ICCVW), pages 621–629, 12 2015. 

regularized correlation filters for visual tracking. In 2015 IEEE International

[10] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael 
Felsberg. Beyond correlation filters: Learning continuous convolution op-
erators for visual tracking. In European Conference on Computer Vision, 

[11] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Fels-
Cited on page 35.

[12] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Fels-
berg. Discriminative scale space tracking. IEEE transactions on pattern 
2, 4, 8, 10, 16, 21, and 32.

timodal deep learning for robust rgb-d object recognition. In 2015 IEEE/RSJ 
International Conference on Intelligent Robots and Systems (IROS), pages 
52, and 78.

[14] Gunnar Farnebäck. Polynomial Expansion for Orientation and Motion Es-
timation. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, 

[15] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ra-
manan. Object detection with discriminatively trained part-based models. 
IEEE transactions on pattern analysis and machine intelligence, 32(9):1627–


[17] Susanna Gladh, Martin Danelljan, Fahad Shahbaz Khan, and Michael Fels-
berg. Deep motion features for visual tracking. 2016 23rd International 

[18] Erhan Gundogdu and A. Aydin Alatan. Good features to correlate for visual 


