Object Tracking Using Tracking-Learning-Detection in Thermal Infrared Video
Object Tracking Using Tracking-Learning-Detection in Thermal Infrared Video

Examensarbete utfört i Datorseende vid Tekniska högskolan vid Linköpings universitet av

Magnus Stigson

LiTH-ISY-EX--13/4668--SE

Handledare: Marcus Wallenberg
ISY, Linköpings universitet
Jörgen Ahlberg
Termisk Systemteknik

Examinator: Per-Erik Forssén
ISY, Linköpings universitet

Linköping, 31 maj 2013
Automatic tracking of an object of interest in a video sequence is a task that has been much researched. Difficulties include varying scale of the object, rotation and object appearance changing over time, thus leading to tracking failures. Different tracking methods, such as short-term tracking often fail if the object steps out of the camera's field of view, or changes shape rapidly. Also, small inaccuracies in the tracking method can accumulate over time, which can lead to tracking drift. Long-term tracking is also problematic, partly due to updating and degradation of the object model, leading to incorrectly classified and tracked objects.

This master's thesis implements a long-term tracking framework called Tracking-Learning-Detection which can learn and adapt, using so called P/N-learning, to changing object appearance over time, thus making it more robust to tracking failures. The framework consists of three parts; a tracking module which follows the object from frame to frame, a learning module that learns new appearances of the object, and a detection module which can detect learned appearances of the object and correct the tracking module if necessary.

This tracking framework is evaluated on thermal infrared videos and the results are compared to the results obtained from videos captured within the visible spectrum. Several important differences between visual and thermal infrared tracking are presented, and the effect these have on the tracking performance is evaluated.

In conclusion, the results are analyzed to evaluate which differences matter the most and how they affect tracking, and a number of different ways to improve the tracking are proposed.
Automatisk följning av ett intressant objekt i en videosekvens är en uppgift som mycket forskning skett på. En framgångsrik följningsmetod måste vara okänslig för varierande skala på objektet, rotation och att objektet ändrar utseende över tid, vilka alla kan leda till misslyckad följning. Olika objektföljningssystem, såsom korttidsföljning faller ofta om objektet lämnar kameras synfält, eller snabbt ändrar utseende. Även små fel i precision av följningen kan ackumuleras över tid, vilket leder till drift av följningen. Långtidsföljning är också problematiskt, delvis på grund av uppdatering och försämring av objektmodellen, vilket leder till felaktig klassificering och följning av objekt.

Detta examensarbete implementerar ett ramverk för långtidsföljning kallat Tracking-Learning-Detection (TLD) som, genom så kallad P/N-learning, kan anpassa sig till olika utseenden av objektet, vilket leder till ett mer robust system. Ramverket består av tre delar; en följningsmodul som följer objektet från bildruta till bildruta, en lärande modul som lär detektorn nya utseenden av objektet, och en detektormodul som kan detektera redan kända utseenden av objektet och korrigera följningsmodulen vid behov.

Följningsramverket har sedan utvärderats på termisk infraröd video och resultaten jämförs med resultat från video inspelad i det visuella spektrumet. Några viktiga skillnader mellan visuell och termisk infraröd följning presenteras och effekten dessa skillnader har på följningsprestandan har utvärderats.

Slutligen analyseras resultaten för att utvärdera vilka skillnader som är viktigast och hur de påverkar objektföljningen. Även ett antal olika sätt att förbättra objektföljningen på presenteras.
Abstract

Automatic tracking of an object of interest in a video sequence is a task that has been much researched. Difficulties include varying scale of the object, rotation and object appearance changing over time, thus leading to tracking failures. Different tracking methods, such as short-term tracking often fail if the object steps out of the camera’s field of view, or changes shape rapidly. Also, small inaccuracies in the tracking method can accumulate over time, which can lead to tracking drift. Long-term tracking is also problematic, partly due to updating and degradation of the object model, leading to incorrectly classified and tracked objects.

This master’s thesis implements a long-term tracking framework called Tracking-Learning-Detection which can learn and adapt, using so called P/N-learning, to changing object appearance over time, thus making it more robust to tracking failures. The framework consists of three parts; a tracking module which follows the object from frame to frame, a learning module that learns new appearances of the object, and a detection module which can detect learned appearances of the object and correct the tracking module if necessary.

This tracking framework is evaluated on thermal infrared videos and the results are compared to the results obtained from videos captured within the visible spectrum. Several important differences between visual and thermal infrared tracking are presented, and the effect these have on the tracking performance is evaluated.

In conclusion, the results are analyzed to evaluate which differences matter the most and how they affect tracking, and a number of different ways to improve the tracking are proposed.
## Contents

Notation

<table>
<thead>
<tr>
<th>1 Introduction</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Problem Description</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Method</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Aims and Goals</td>
<td>3</td>
</tr>
<tr>
<td>1.5 Limitations</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2 Theory</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Thermal Infrared Imaging</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Overview: Tracking Paradigms</td>
<td>6</td>
</tr>
<tr>
<td>2.2.1 Tracking by Displacement</td>
<td>6</td>
</tr>
<tr>
<td>2.2.2 Tracking by Detection</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Lucas-Kanade Tracking</td>
<td>8</td>
</tr>
<tr>
<td>2.4 Integral Images</td>
<td>9</td>
</tr>
<tr>
<td>2.5 P/N Learning</td>
<td>10</td>
</tr>
<tr>
<td>2.6 Random Forests</td>
<td>11</td>
</tr>
<tr>
<td>2.6.1 Random Fern Forests</td>
<td>13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3 Implementation of Tracking-Learning-Detection</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Tracking</td>
<td>18</td>
</tr>
<tr>
<td>3.2 Detection</td>
<td>19</td>
</tr>
<tr>
<td>3.2.1 Scanning-window Grid</td>
<td>20</td>
</tr>
<tr>
<td>3.2.2 Detector Cascade</td>
<td>21</td>
</tr>
<tr>
<td>3.3 Learning</td>
<td>24</td>
</tr>
<tr>
<td>3.3.1 Result Integration</td>
<td>24</td>
</tr>
<tr>
<td>3.3.2 Valid Learning</td>
<td>24</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4 Results</th>
<th>29</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Video Acquisition</td>
<td>29</td>
</tr>
<tr>
<td>4.2 Recorded Sequences</td>
<td>33</td>
</tr>
<tr>
<td>4.3 Performance Measures</td>
<td>34</td>
</tr>
</tbody>
</table>
# Notation

## Table of Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta^+$</td>
<td>Upper limit for adding positive template. Pass limit for nearest neighbour classifier</td>
</tr>
<tr>
<td>$\theta^-$</td>
<td>Lower limit for adding positive template</td>
</tr>
<tr>
<td>$I$</td>
<td>Current frame for evaluation</td>
</tr>
<tr>
<td>$L$</td>
<td>Total number of ferns</td>
</tr>
<tr>
<td>$Q$</td>
<td>Fern depth, number of features per fern</td>
</tr>
<tr>
<td>$#p$</td>
<td>Number of positive object model templates</td>
</tr>
<tr>
<td>$#n$</td>
<td>Number of negative object model templates</td>
</tr>
<tr>
<td>$P_i(y</td>
<td>x)$</td>
</tr>
<tr>
<td>$M$</td>
<td>Object model</td>
</tr>
<tr>
<td>$NCC$</td>
<td>Normalized correlation coefficient</td>
</tr>
<tr>
<td>$S^r$</td>
<td>Relative similarity to object model</td>
</tr>
<tr>
<td>$S^c$</td>
<td>Conservative similarity to object model</td>
</tr>
<tr>
<td>$T_t$</td>
<td>Current tracker bounding box</td>
</tr>
<tr>
<td>$D_t$</td>
<td>Current detector bounding box(es)</td>
</tr>
<tr>
<td>$B_t$</td>
<td>Current resulting bounding box</td>
</tr>
<tr>
<td>$valid(B_t)$</td>
<td>Indicates if current resulting bounding box is valid for learning</td>
</tr>
<tr>
<td>$valid(B_{t-1})$</td>
<td>Indicates if previous resulting bounding box was valid for learning</td>
</tr>
<tr>
<td>$B_e$</td>
<td>Current bounding box for evaluation</td>
</tr>
<tr>
<td>$R$</td>
<td>All bounding boxes in scanning-window grid</td>
</tr>
<tr>
<td>$R_E$</td>
<td>All bounding boxes in scanning-window grid that passed variance and ensemble classifiers</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Background

Consider a video sequence depicting various objects moving in and out of the camera’s field of view. Given a bounding box defining an object of interest in a single frame, the goal of tracking is to automatically determine the object’s bounding box, or indicate that the object is not present, in every frame that follows. Tracking is in simple cases solved using short-term tracking where the tracking often fails if the object disappears from view, changes appearance rapidly or simply moves too fast.

The idea of long-term tracking, where a system does not fail if the object of interest leaves the camera’s field of view or changes appearance, is something that is preferable in most situations.

Tracking is something that is most often done using video footage recorded with an ordinary camera. This thesis evaluates tracking in the thermal infrared spectrum, as opposed to the visible spectrum recorded with said camera. Some differences between the two spectra exist, and the effect these differences have on tracking capabilities are investigated.

1.2 Problem Description

The Lucas-Kanade (LK) tracking algorithm [11, 16] (section 2.3) is an example of a short-term tracker, also called tracking by displacement (section 2.2.1), which is easy to implement and performs well when the object appearance does not change significantly between frames. However, due to changes in viewpoint, orientation and in some cases deformation, the image of an object can change signif-
icantly in size and appearance over time, and tracking will eventually fail. This is often due to small inaccuracies in the tracking which accumulate over time, which leads to tracking failures that are hard to recover from, also called *drift*.

Another approach to tracking is known as *tracking by detection* [5] (section 2.2.2). This approach uses one or several *templates* which are compared to several positions in the frame, and the position of the object of interest is estimated as the best response from this evaluation. The difference to tracking by displacement is primarily that a detector does not estimate the object’s displacement, but its *position* in the frame, and it does so only by evaluating one frame in contrast to tracking by displacement which needs two frames. The good thing about tracking by detection is that it does not drift and does not fail if the object disappears from view. Instead, a lack of detection is considered as the absence of the object of interest. Detectors often require an initial training phase, and thus, they can not be used to identify unknown objects. Some exceptions exist, such as foreground-background segmentation or hot-spot detection.

Both of these tracking approaches make two types of errors: false positives and false negatives. These errors are estimated using *ground truth*, which is the considered best match for the current frame, and is most often produced by a manual annotation. False positives occur when the detector finds a match in the wrong place according to ground truth. False negatives are when the object of interest is not detected although it should have been according to ground truth.

### 1.3 Method

Both of the presented tracking approaches have problems producing a stable tracker that works over long periods of time without tracking failures. The key to the tracking approach in this thesis lies in accepting that neither a tracker nor a detector is fool-proof. Combining the advantages of both provides a good starting point for a fast and flexible tracking framework.

The combination of both a tracker and detector to learn and adapt to changes in the object is used in a *Tracking-Learning-Detection* (TLD) framework, such as that proposed by Kalal et al. [10]. In this framework, three parts work simultaneously; a tracker, learner and a detector (figure 1.1).

The tracker follows the object from frame to frame, using a modified LK-tracking algorithm called median-flow tracking [8]. The detector finds all highly probable locations for the object of interest to be found, and corrects the tracker if necessary. The learner estimates the detector’s errors and updates it to avoid these errors in the future. It is important to note that the tracker and detector work independently from each other (except for re-initialization by the detector when necessary), and can benefit from the combined knowledge in the learning step.
1.4 Aims and Goals

The goal of this master thesis is to implement and evaluate a Tracking-Learning-Detection framework, study its performance on thermal infrared video, and suggest and test improvements.

The plan of the master thesis is to:

1 - Carry out a literature study on TLD and read up on necessary elements of classification, object tracking, feature extraction and learning (chapter 2).

2 - Implement a TLD-method in C++ and OpenCV (chapter 3). A simple hotspot detector and/or manual annotation can be used as initialization for the tracker.

3 - Record a number of thermal infrared video sequences of varying difficulty (section 4.1).

4 - Study the performance of the method, suggest and test improvements (section 4.4). Evaluate which design choices are different for thermal infrared video and visible spectrum video (chapter 5).

1.5 Limitations

The TLD implementation should, with a proper initialization of the tracker, be able to track a novel object without additional input.

The software written need not perform initial detection of an object. It is only required to track targets provided by either an external detector or by manual annotation.
In this chapter, some important theory regarding the framework of TLD is presented, together with details about how it fits into the implementation. The reader is expected to know basic image processing and machine learning to fully understand some concepts.

2.1 Thermal Infrared Imaging

Although infrared (IR) radiation is not detectable by the human eye, it can be detected by cameras specifically designed for this part of the electromagnetic spectrum (figure 2.1). This is in contrast to visible spectrum cameras which only cover the visible part (~390-700 nm) of the electromagnetic spectrum [5]. An example output from an IR camera in relation to a visible spectrum camera is shown in figure 2.2. Some notable differences compared to that of a visible spectrum camera are that the resulting IR image output is more influenced by noise and that it is often of a lower resolution than the visible spectrum image. These problems can be traced to the imperfect fabrication processes of the IR camera, limited by currently available technology. This is also due to the detector in an uncooled IR camera which work according to the bolometer effect\(^1\), whereas a camera which work according to quantum principles will have a much higher sensitivity and lower noise [2]. The quantum detector will not be discussed more in detail.

IR cameras usually do not cover the entire infrared spectrum, but only a part of it. The infrared spectrum is usually divided into three bands, which have different

\(^1\)A bolometer is a simple device that consists, in essence, of a material whose resistance varies with temperature [14].
uses and characteristics. Various terminologies are in use for the subdivision of these bands. The one adopted here defines the bands as:

- **Near infrared (NIR)** $\sim 0.9-1.4\mu m$
- **Mid-wave infrared (MWIR)** $\sim 2-5\mu m$
- **Long-wave infrared (LWIR)** $\sim 7.5-14\mu m$

This subdivision of bands is shown in figure 2.3. The LWIR and MWIR bands are usually used for **thermal infrared** imaging, where the temperature of an object can be measured if the emissivity and the atmospheric attenuation are taken into consideration. If atmospheric attenuation is disregarded, the measured temperature will be lower with increased distance due to absorption of gases and scattering particles in the atmosphere [2]. The emissivity of an object is also an important factor, due to no object being a perfect absorber, reflector or transmitter.

![Simplified block diagram of an IR camera](https://via.placeholder.com/150)

**Figure 2.1**: Simplified block diagram of an IR camera. The same parts are often involved in a visible spectrum camera as well, but the detector is sensitive to visible light.

### 2.2 Overview: Tracking Paradigms

Object tracking is a well-researched subject, and two major tracking paradigms exist, **tracking by displacement** and **tracking by detection**. These two paradigms are further described below.

#### 2.2.1 Tracking by Displacement

One method of estimating an object’s displacement between two image relies on feature tracking. This method often involves the Lucas-Kanade (LK) tracking algorithm [11, 16] to track individual feature points in a larger tracking framework. The principles of LK-tracking are explained in further detail in section 2.3.

A drawback of this method is that it can not recover if tracking is lost, which means that it will not produce a correct result if the object is not present in the current image, or if it is significantly occluded.
2.2 Overview: Tracking Paradigms

Figure 2.2: Image (a) and (b) show the same scene in different parts of the electromagnetic spectrum. Image (a) is captured within the visible spectrum (\(\sim 390-700\text{nm}\)) and image (b) is captured in the LWIR band (\(\sim 7.5-14\mu\text{m}\)). Image (b) is a false-color image, where the color bar indicates temperature in °C.

![Visible spectrum and Thermal infrared](image)

2.2.2 Tracking by Detection

Another approach to tracking is known as tracking by detection (template matching [5]). One of the many ways to detect objects is to use one or several object templates that are detected by correlation, most often using the normalized correlation coefficient, with the image that is to be evaluated. The correlation coefficient is used to find the most similar location. The method can also be used in conjunction with a scanning-window grid, where several subwindows of the image are created, and these subwindows are tested for the presence of the object of interest [12].
The main difference to tracking by displacement is that tracking by detection does not estimate the object’s displacement, but its position in the image. It does so from only one image, detecting each object separately.

A drawback with this method is that the object detector can not be used on its own, but must have a template to match. This means that tracking by detection can not be used if the object of interest is unknown beforehand and must be learned before use.

### 2.3 Lucas-Kanade Tracking

A well-known algorithm for tracking individual points between two images is the Lucas-Kanade (LK) tracking algorithm [11, 16]. It assumes spatial similarity between two images, and finds the most probable displacement of the point location using an iterative solving of a dissimilarity equation. LK-tracking relies on the brightness constancy constraint equation (BCCE), which says that the intensity of a pixel corresponding to a certain world point should be the same regardless of where it appears in the image,

\[
\frac{\partial I}{\partial x} d_x + \frac{\partial I}{\partial y} d_y + \frac{\partial I}{\partial t} = 0, \quad (2.1)
\]

where \(d = [d_x, d_y]^T\) is the unknown object displacement that is to be found.

A dissimilarity between two local regions, one in image \(I\) and one in image \(J\) is in this case defined as

\[
\epsilon = \iint_W |J(x + d) - I(x)|^2 w(x) dx, \quad (2.2)
\]

where \(x = [x, y]^T\) is the object position and the integration region \(W\) is a local neighborhood of a pixel. The weighting function \(w(x)\) is usually set to a constant 1, and can often be omitted for reasons of simplicity.

By first order Taylor series expansion of \(J(x + d)\) around the point \(x\), truncated to the linear term, the equation can be rearranged to solve for \(d\),

\[
Zd = e, \quad (2.3)
\]

where \(Z\) is the 2 x 2 matrix

\[
Z = \iint_W \nabla J(x) \nabla J(x)^T dx \quad (2.4)
\]
and $e$ is the $2 \times 1$ vector

$$e = \iint_W [I(x) - J(x)]\nabla J(x)dx.$$  \hspace{1cm} (2.5)

This solution to $d$ is not exact, and cannot be solved exactly by inversion of $Z$, due to it being only approximately linear. The equation can instead be solved iteratively until some stop criterium is fulfilled, such as that a maximum number of iterations has been carried out, or that the change in $||d||$ is below some threshold.

### 2.4 Integral Images

Use of integral images (also called summed area tables) is a fast and efficient way of computing sums over parts of an array or image. This is used to calculate the variance of a subwindow, by calculating the first and second moment of the subwindow. A fast calculation of the variance of an image subwindow is a crucial part in the TLD framework.

An integral image is a representation of the original image, where each point in the integral image represents the sum of the rectangle spanned from the origin to the current pixel position. This is shown in equation 2.6,

$$I(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$  \hspace{1cm} (2.6)

where $x', y'$ are the coordinates in the original image $i$, and $x, y$ are the coordinates for the integral image $I$. The origin is located according to figure 2.4.

The integral image can be constructed efficiently in a single pass over the original image as

$$I(x, y) = i(x, y) + I(x - 1, y) + I(x, y - 1) - I(x - 1, y - 1).$$  \hspace{1cm} (2.7)

Once the integral image has been constructed, it is a simple task to calculate the sum of all pixel values inside a rectangle of the original image using four array references (figure 2.4). This is computed as

$$\sum_{A(x) \leq x' \leq C(x), A(y) \leq y' \leq C(y)} i(x', y') = I(C) + I(A) - I(B) - I(D).$$  \hspace{1cm} (2.8)
using the notation in figure 2.4.

![Figure 2.4: Rectangle sum calculation using an integral image. The four values of the pixels in the integral image are all that is needed for an efficient computation of the sum inside the rectangle spanned by the pixel positions.](image)

2.5 P/N Learning

P/N-learning is an example of a machine learning method. Machine learning is currently the subject of much research, and is often divided into a set of learning paradigms:

**Supervised learning** uses labeled training examples for positive and negative feedback, often to train a classifier which discriminates between different object classes using these examples. The classifier in TLD is used to detect if the object of interest is located in the image to be evaluated. The use of labeled examples implies that an annotation of all examples must be carried out before learning. This is often the main drawback of this learning paradigm because it is unable to adapt to changes in distribution of the data if new labeled examples are not given. Because of this, the classifier can also never become better than the teacher it learned from.

**Unsupervised learning** uses no labeled training examples. Instead, it tries to find hidden structures in data often using clustering, outlier detection or dimensionality reduction, and using this information to predict the label of an unknown input. Due to the lack of labeled examples to evaluate, the classifier error must be estimated some other way, often using some kind of compactness cost or entropy-based error measure. The main benefit of this learning paradigm
is however that no user interaction is often necessary to gain a classifier that can discriminate between classes.

**Reinforcement Learning** is a supervised learning paradigm common in online applications. It tries to teach an *agent* how to act in a given environment, by issuing a policy it can follow so as to maximize some reward. It learns the agent by only encountering small parts of the cost function at a time. It is often used in game theory or simulations, where a virtual AI can be controlled using said policy.

**Semi-supervised Learning** is a method that combines aspects of both supervised and unsupervised learning by using both labeled and unlabeled examples to train a classifier. Kalal et al. use this in [9] to train a classifier, where they employ two structural constraints which they call a *P-learner* and an *N-learner*. The P-learner uses the fact that the object of interest moves along a trajectory, and estimates the object position using tracking by displacement. The N-learner uses spatial information in the video, that the object can only occupy one location in the video frame at the same time. Together these two structural constraints try to predict errors in the current classifier and generate new labeled examples, which are used to update the classifier. This error prediction is done in every frame, and essentially teaches the classifier to recognize new appearances of the object of interest. The classifier is updated when the tracker has found a detection which is confident enough and can provide new information to the classifier. This learning process is called *P/N-Learning*.

The P-learner recognizes when the classifier has given a false positive response, by using overlap of the most confident bounding box that is the output from the tracking framework and the bounding box that the classifier has wrongly designated as not containing the object. The P-learner can from this information generate new positive examples. The N-learner recognizes false positives made by the classifier in the same way, and can generate negative examples from this information. These positive and negative examples generated from both the P- and N-learner are then used to retrain the classifier. A block diagram of this method is shown in figure 2.5.

The dynamic interaction between the P-learner and the N-learner enables mutual compensation of the two learner errors, and leads to a stable learning of the classifier. As shown in [9], the learner will improve performance over time if the learner error is small enough.

### 2.6 Random Forests

The *random forest* classifier was first described by Leo Breiman and Ho Tin [4, 7], and is an ensemble classifier. The random forest classifier consists of many *decision trees*, and the forest outputs the class that is the consensus output by the set of trees (figure 2.6). By using a random set of features for each tree so they are independent of each other, the classifier can be made very robust. This inde-
Figure 2.5: Block diagram of P/N-learning (Adapted from [10]).

Independence assumption is however usually not true, but the chance of obtaining independent feature sets increases by sampling them at random. The randomness is therefore often introduced by either drawing random subsets of training samples, or random feature subsets when training.

Figure 2.6: Simplified diagram of a random forest classifier.

The purpose of the random forest classifier is to evaluate the posterior probability
over a class label $C_k$, for all trees $f_i$ in the ensemble, according to equation 2.9

$$\text{Class}(f) \equiv \arg \max_k P(C_k|f_1, f_2, \ldots, f_N),$$

(2.9)

where $k$ is the tree number, $C$ is the class and $N$ is the total number of trees.

Bayes' theorem [3] tells us that this equation is equivalent to evaluating

$$\arg \max_k P(f_1, f_2, \ldots, f_N|C_k)P(C_k).$$

(2.10)

But learning this joint likelihood is not feasible, and the naive assumption that features are conditionally independent is made, and the expression is formulated as equation 2.11,

$$P(f_1, f_2, \ldots, f_N|C_k) = \prod_{i=1}^{N} P(C_k|f_i).$$

(2.11)

This leads us finally to equation 2.12,

$$\text{Class}(f) = \arg \max_k P(C_k) \prod_{i=1}^{N} P(C_k|f_i).$$

(2.12)

To note is that the assumption that all features are conditionally independent is often false, due to most variables being correlated in some way. The true posterior probabilities are therefore often grossly underestimated.

The random forest classifier is still one of the most accurate learning algorithms available, and produces a highly accurate classifier for many datasets. It is also very efficient on large datasets, where many other algorithms struggle to perform.

### 2.6.1 Random Fern Forests

The random fern forest classifier [18] is a variation on the random forest classifier. It differs from the random forest in two important respects. In random forests, each binary test is organized hierarchically and the posterior probability distributions are computed additively. This is in contrast to ferns, which are flat and compute posteriors multiplicatively.

The fern classifier is based on the same principle as the random forest classifier; it evaluates the posterior probability over a class label. It however uses a "semi-naive" Bayes simplification, where it assumes that groups of features are conditionally independent. It works by dividing all features into $L$ small groups of
size $Q$, called *ferns*, and evaluating the posterior probability to calculate a class label. One fern $F_l$ is defined as

$$F_l = \{f_{l,1}, f_{l,2}, \ldots, f_{l,Q}\}$$  \hspace{1cm} (2.13)

and contains $Q$ binary features such that $f_n : \{0, 1\}$.

All ferns are assumed to be conditionally independent, and Bayes’ theorem therefore gives us equation 2.14,

$$P(f_1, f_2, \ldots, f_N | C_k) = \prod_{l=1}^{L} P(C_k | F_l).$$  \hspace{1cm} (2.14)

We then learn the class-conditional distributions for each group and apply Bayes’ theorem to obtain the posterior according to equation 2.15,

$$\text{Class}(f) \equiv \arg \max_k P(C_k) \prod_{l=1}^{L} P(C_k | F_l).$$  \hspace{1cm} (2.15)

The benefits of using this approach in contrast to the standard random forest classifier is that we have a better balance of complexity and tractability, while also having a better model of the true posterior probabilities. This is in contrast to the standard random forest classifier which is too simplistic and gives a relatively poor approximation of the true posterior probabilities [18]. The complexity can easily be traded for performance by the choice of fern size $Q$ and the number of ferns $L$.

The random fern forest classifier is implemented in TLD using binary tests on each patch it evaluates. Each fern gets $Q$ features to test and each test measures the relative pixel intensity of a pair of pixels to get a binary answer, i.e. $f_i(I) = I(x_a, y_a) > I(x_b, y_b) \rightarrow \{0, 1\}$ (figure 2.7).
Figure 2.7: Pixel comparisons on a subwindow generate a binary code, converted to an integer (Adapted from [10]).
In this chapter, the implementation of the Tracking-Learning-Detection framework is presented along with details about what values for different parameters are considered a good fit for most problems. A detailed block diagram for TLD is shown in figure 3.1, where the different modules are presented along with their functions and interconnections.

*Figure 3.1: Detailed block diagram of Tracking-Learning-Detection.*
3.1 Tracking

The purpose of the tracker is to estimate the displacement of the object of interest in a new frame of the video sequence. This is in contrast to the detector which only locates known appearances of the object. In this section it is described how the tracking part of the TLD framework is implemented. This method needs no a priori information except the location of the bounding box in the previous frame to be tracked along the video sequence. This initialization can be found either by a separate detector that finds the bounding box of the object of interest, or a manual annotation where the user selects the object.

This tracker implementation follows the approach proposed by Kalal et al.[10]. There, a median-flow tracking algorithm is employed [8] along with a track-retrack confidence measure [6] to indicate tracking failure. This approach is showed in figure 3.2. The median-flow tracking algorithm is robust to partial occlusions, which makes it a good candidate for tracking objects with variable visibility and shape.

The tracker works by initializing a regularly spaced set of points \( x_t \) in frame \( t \) inside a bounding box, and track these forward to frame \( t+1 \) using LK-tracking (section 2.3). This new set is denoted \( x_{t+1} \) (see figure 3.3).

LK-tracking fails in areas of low contrast and where no reliable match can be found, which means that a method of indicating the tracking reliability is needed. The reliability is found by tracking the set of points \( x_{t+1} \) backwards to frame \( t \), now called \( \hat{x}_t \), and using the euclidean distance between these two sets,

\[
\epsilon = ||x_t - \hat{x}_t||. \tag{3.1}
\]

If the median of this discrepancy over all points in the set is larger than a threshold \( \epsilon_{\text{thres.}} \), the tracking is considered to have failed. The threshold is set to a constant value of 10 pixels.

If the median error is lower than the threshold, tracking continues by filtering out 50% of the least consistent points in \( x_{t+1} \). The translation of the bounding box is then calculated as the median displacement of the remaining points.

The scale of the bounding box is estimated by calculating the ratio between the current point distance and the previous point distance for each pair of points, and using the mean of this ratio. An implicit assumption of this calculation is that the spatial distribution of intensity does not change.

In each frame of the sequence, a new set of points is tracked which makes the tracking algorithm very adaptive to change. The set \( x_t \) is in this implementation composed of a regularly spaced grid consisting of 10x10 points.
3.2 Detection

The detector’s main purpose is to find previously learned appearances of the object of interest, and correct the tracking if necessary. This is done using a **scanning-window grid**, which searches over the entire current frame and calculates a probability that the object of interest is inside any given **subwindow**. The output from the detector is a variable number of bounding boxes with high probability of containing the object of interest (or no bounding box at all if no detections are found with high enough probability). This process is time-consuming and is sped up using a **detector cascade**, which enables fast rejection of subwindows that have a low probability of containing the object of interest.

**Figure 3.2**: Track-retrack error estimation. The tracking of point 1 fails due to the Track-retrack error being over a threshold. The error of point 2 is below the threshold, thus considering point 2 correctly tracked.
3.2.1 Scanning-window Grid

The detector locates previously known appearances of the object of interest by evaluating several subwindows of the original patch bounding box by varying its location and scale. This is called a scanning-window grid (figure 3.4). This approach can become cumbersome due to the large number of subwindows to evaluate if no restrictions are placed on their number. Using all possible scales and locations in the search space, up to $23,507,020,800$ subwindows must be evaluated for a 640x480 pixels image [12]. The number of subwindows also grows with $n^4$ for images of size $n \times n$, making an exhaustive search intractable for large frame sizes.

The search space is narrowed by using some constraints on the subwindows: Firstly, all subwindows keep the original bounding box aspect ratio. Secondly, a margin distance $d_x$ and $d_y$ in horizontal and vertical direction between subwindows are set to be $\frac{1}{10}$ of the original bounding box width and height respectively. The scale change between subwindows is also constrained by a scale factor $s = 1.2^a$ of the original bounding box, where $a \in \{-10, 10\}$ [12].

Figure 3.4: Scanning-window grid approach.
These constraints give the scanning-window grid a much needed trimming, down to about 150,000 subwindows to evaluate for a VGA (640x480 pixels) resolution image with an initial bounding box of size 80x60 [12].

### 3.2.2 Detector Cascade

The detector cascade works in three stages, where each stage rejects subwindows if they are considered not containing the object of interest. This makes the detector faster, by restricting the application of precise but time-consuming methods to subwindows that are likely to contain the object of interest. This approach is shown in figure 3.5.

![Figure 3.5: Block diagram of the detector cascade.](image)

#### Patch Variance

The first stage of the cascade consists of a threshold based on variance of the current image patch denoted by the current subwindow. If the variance of the patch to be evaluated is less than 50% of that of the original patch, the subwindow is rejected and not processed further in the cascade. The variance is calculated using two integral images (section 2.4) constructed from the current frame. This stage exploits the fact that the variance of an image patch \( p \) can be expressed as \( \text{Var}(p) = \mathbb{E}(p^2) - \mathbb{E}^2(p) \), and the expected value \( \mathbb{E}(p) \) and \( \mathbb{E}(p^2) \) can be calculated in constant time using integral images (section 2.4). The use of integral images make this part of the cascade very fast to evaluate, and is crucial for real-time detection.

#### Ensemble Classifier

If the subwindow passes the patch variance classifier, it is evaluated using an ensemble classifier. This implementation uses a random fern forest classifier (section 2.6.1), which provides a good compromise between speed and accuracy.

The features are binary pixel comparisons, which are generated offline and distributed randomly among all ferns, based on a normalized resolution of 15x15 pixels. These features are stretched to the size of the current subwindow and evaluated for each fern when classifying the image patch denoted by the current subwindow. All pixel comparisons generate a binary code, which is converted to an integer for simplicity (figure 2.7).
The choice of fern depth means that a total of $2^Q$ entries are generated for each fern. Each entry has a number of positive and negative patches associated with it, and are used to estimate the posterior probability $P_i(y|x)$ according to $P_i(y|x) = \frac{#p}{#p + #n}$, where $#p$ and $#n$ are the number of positive and negative patches corresponding to that entry respectively. The ensemble classifies the current patch as containing the object of interest if the mode over all ferns is over 50%. If $#p + #n = 0$, the classifier outputs 0, i.e. votes for negative class.

The ensemble classifier is initialized and updated using P/N-learning (section 2.5). The classifier is initialized by generating 50 warped positive training examples for each bounding box in the scanning window grid that overlaps with the initial bounding box more than 50%. Negative training examples are taken from the neighbourhood of the initial bounding box in the search area generated by the scanning-window grid.

The classifier is updated during runtime according to section 3.3.2.

**Nearest Neighbour Classifier**

The nearest neighbour classifier is the final stage of the detector cascade, and only evaluates those subwindows that have already passed the patch variance and ensemble classifier. The classifier uses a correlation-based measure to determine if the image patch contained in a certain subwindow contains the object of interest, which means that it is the most time-consuming part of the cascade but it also has the highest accuracy.

An *object model* is defined as a data structure for the nearest neighbour classifier, which contains positive and negative templates of the object of interest, $M = \{p_1^+, p_2^+, \ldots, p_m^+, p_1^-, p_2^-, \ldots, p_n^\}$. The positive templates are examples of the object and negative templates are of the background. The positive templates are ordered in time according to the time when the template was added to the collection. All templates are of a normalized resolution of 15x15 pixels, and used to evaluate the current patch which is resampled to this resolution. A similarity between an image patch $p$ and a template $t$ in the model is defined as

$$S(p, t) = 0.5(NCC(p, t) + 1),$$

where

$$NCC(p, t) = \frac{1}{XY} \sum_{x=1}^{X} \sum_{y=1}^{Y} \frac{(p(x, y) - \mu_p)(t(x, y) - \mu_t)}{\sigma_p \sigma_t},$$

and $\mu_p, \mu_t, \sigma_p, \sigma_t$ are the mean and standard deviations of $p$ and $t$ respectively, and $X, Y$ is the template size in horizontal and vertical direction.

Several similarity measures are used to compare the current image patch to the
3.2 Detection

1. Similarity with the nearest positive neighbour, \( S^+(p, M) = \max_{t_i^+} S(p, t_i^+) \).

2. Similarity with the nearest negative neighbour, \( S^-(p, M) = \max_{t_i^-} S(p, t_i^-) \).

3. Similarity with the nearest positive neighbour considering 50\% of the earliest positive templates, \( S_{50\%}^+(p, M) = \max_{t_i^+ \in M \land i < \frac{m}{2}} S(p, t_i^+) \)

4. Relative similarity, \( S^r = \frac{1 - S^-}{1 - S^r + 1 - S^-} \).

5. Conservative similarity, \( S^c = \frac{1 - S^-}{1 - S_{50\%}^r + 1 - S^-} \).

The relative similarity expresses the confidence that the subwindow is containing the object of interest, and ranges from 0 to 1, where 1 is high confidence. The conservative similarity expresses the confidence that the subwindow is containing the object of interest, based on the first 50\% of the positive model templates.

The relative similarity measure is used in the nearest neighbour classifier. If \( S^r > 0.5 \), the subwindow is classified as positive, and passes the nearest neighbour classifier. An illustration of the nearest neighbour classifier is shown in figure 3.6.

The conservative similarity is used in the learning process, where it measures if the current subwindow is valid for learning.

**Figure 3.6: Nearest Neighbour Classifier.** \( S^+ \) is the distance to the nearest positive model template and \( S^- \) is the distance to the nearest negative model template.
3.3 Learning

P/N-learning, as described in section 2.5, is used to update the ensemble classifier and the object model in the detector. For each frame in the video sequence, the median-flow tracker outputs a bounding box and the detector outputs one or several bounding boxes that have a high probability of containing the object of interest. The tracker or detector does not output any bounding box if the object is not present or tracking is considered to have failed.

The output bounding boxes are combined into one in the result integration part of the learning. The resulting bounding box is used to estimate if learning of the ensemble classifier and the object model should be done. The resulting bounding box is only valid under certain conditions, all of which assume that the tracker is not re-initialized by the detector, i.e. the resulting bounding box is not set from the most confident detector output bounding box.

3.3.1 Result Integration

The estimated bounding boxes from both the tracking and detection modules are combined into one. This resulting bounding box is then evaluated if it is valid to be learned from. This procedure is done according to algorithm 3.1.

The most confident bounding box according to the conservative similarity measure is always the output (line 5, 7 and 15), while this output is only valid if two conditions are met: if the output bounding box has a conservative similarity score over $\theta^+ = 0.65$ and is not re-initialized by the detector (line 9), or if the previous bounding box was valid, not re-initialized by the detector and the current output bounding box has a conservative similarity score over $\theta^- = 0.5$ (line 11).

The tracker is re-initialized if the detector has an output that is more confident than the tracker (line 5) or if the tracker has failed and the detector has an output that has a conservative similarity score over $\theta^-$ (line 15).

3.3.2 Valid Learning

If the resulting bounding box is valid, learning takes place. The initial learning of the detector on the first frame is done on all bounding boxes from the scanning-window grid. This is opposed to runtime learning which is only done on subwindows that passed both the patch variance and ensemble classifier.

The procedure for the learning is shown in algorithm 3.2 where $P^+$ and $P^-$ are the positive and negative object model template collections, and $\#p$ and $\#n$ are the number of positive and negative training examples corresponding to the current fern code.

Runtime update of the ensemble classifier is done by evaluating all bounding boxes from the scanning-window grid which are classified as positive from the ensemble (line 5). Spatial similarity between bounding boxes is measured using overlap, which is the ratio between intersection and union of two bounding boxes. If the bounding box from the result integration overlaps more than 50% with the
Algorithm 3.1 Result integration

Input: $T_t, D_t, valid(B_{t-1})$
Output: $B_t, valid(B_t)$

1: $B_t \leftarrow \emptyset$
2: $valid(B_t) \leftarrow false$
3: if $T_t \neq \emptyset$ then
4: if $S^c(D_t) > S^c(T_t)$ then
5: $B_t \leftarrow D_t$
6: else
7: $B_t \leftarrow T_t$
8: if $S^c(T_t) > \theta^+$ then
9: $valid(B_t) \leftarrow true$
else if $valid(B_{t-1}) \& \ S^c(T_t) > \theta^-$ then
10: $valid(B_t) \leftarrow true$
11: end if
12: end if
13: else if $S^c(D_t) > \theta^-$ then
14: $B_t \leftarrow D_t$
15: end if
16: end if

current bounding box to learn from, then 50 warped positive learning examples are generated from the current bounding box by rotation ($\pm 20^\circ$) (line 6). If the overlap between the resulting bounding box and the current bounding box to learn from is under 20%, then a negative training example is generated from this bounding box (line 11).

The object model is updated in a similar fashion. The model evaluates the same bounding boxes from the scanning-window grid as the ensemble learning, and classifies the patch based on the conservative similarity measure. If the resulting bounding box is valid and the conservative similarity of the current patch denoted by the current subwindow to learn from is above the threshold $\theta^-$, a negative template is added to the object model from the current subwindow (line 17). If the conservative similarity is below $\theta^+$ and the resulting bounding box is valid, a positive template is added to the object model from the resulting bounding box (line 3). This procedure of valid learning is shown in figure 3.7.

During testing it was noted that the framework is sensitive to the relative number of generated positive and negative examples during initial learning. If a significantly larger number of negative examples than positive examples are generated, the negative examples will "overshadow" the positive and result in a very small number (if any) of bounding boxes that pass the ensemble classifier step in the detector cascade. If in turn a significantly larger portion of the generated examples are positive, it will mean that the ensemble classifier lets through many bounding boxes from the variance classifier, thus making the system perform poorly.

These two parts of the problem are solved by setting an upper limit for how many
Algorithm 3.2 Learning

Input: $I, B_t, valid(B_t), R_E or R$

1: if $valid(B_t)$ then
2:   if $R^c(B_t) < \theta^+$ then
3:     $P^+ \leftarrow P^+ \cup I(B_t))$
4:   end if
5:   for all $B_c \in R_E$ do
6:     if $(overlap(B_c, B_t) > 0.5)$ then
7:       for $k = 1 \ldots L$ do
8:         code $\leftarrow$ calcFernFeatures$_k(I(B_c))$
9:         $#p_k[code] \leftarrow #p_k[code] + 1$
10:       end for
11:     else if $overlap(B_c, B_t) < 0.2$ then
12:       for $k = 1 \ldots L$ do
13:         code $\leftarrow$ calcFernFeatures$_k(I(B_c))$
14:         $#n_k[code] \leftarrow #n_k[code] + 1$
15:       end for
16:       if $R^c(B_C) > \theta^-$ then
17:         $P^- \leftarrow P^- \cup I(B_c))$
18:       end if
19:     end if
20:   end for
21: end if
3.3 Learning

**Figure 3.7:** Positive templates are added to the object model only when the confidence drops from above $\theta^+$ (Frame A) to a value between $\theta^+$ and $\theta^-$ (Frame B). Negative templates are added when bounding box is valid and overlap is below 0.2. Learning of ensemble classifier takes place when resulting bounding box is valid. No learning takes place when confidence is below $\theta^-$ (Frame C) or when previous result was not valid (Frame D). (Adapted from [12])

Bounding boxes can be generated for the positive and negative side. The limit is relative to how many bounding boxes that can possibly be generated from the initial frame. If the number of positive or negative generated examples is more than 110% of the lower number of generated examples, the upper number of generated examples is limited to that of the lower number of generated examples. This limiting procedure is shown in algorithm 3.3.
Algorithm 3.3 Limiting the Number of Generated Examples

\[ \#_p \leftarrow \text{Number of generated positive examples} \]
\[ \#_n \leftarrow \text{Number of generated negative examples} \]

\[
\begin{align*}
\text{if } \#_n > \#_p \cdot 1.1 & \text{ then} \\
& \quad \text{maxPositiveExamples } \leftarrow \#_p \\
& \quad \text{maxNegativeExamples } \leftarrow \#_p \cdot 1.1 \\
\text{else if } \#_p > \#_n \cdot 1.1 & \text{ then} \\
& \quad \text{maxPositiveExamples } \leftarrow \#_n \cdot 1.1 \\
& \quad \text{maxNegativeExamples } \leftarrow \#_n
\end{align*}
\]
In this chapter, the results of various tests of the framework are presented together with conclusions about suitable parameter values. The chapter begins with video acquisition, and some differences between visible spectrum and thermal infrared video. Different performance measures are then presented, followed by an experiment description and discussion of suitable parameter values along with some suggestions for improvements.

4.1 Video Acquisition

A set of different video sequences were recorded and the performance of the tracking framework was studied to make for a quantitative evaluation. One of the main interest points was to focus on where the difference between tracking using TLD in visible spectrum video versus thermal infrared video lies. All sequences were therefore shot in both the thermal infrared spectrum and visible spectrum. The sequences were captured with three sets of varying difficulty:

**Easy** - Sequences of this type were recorded with good contrast to the background in both visible and thermal infrared spectrum for easy localization of the object and little to no variation in object appearance (no rotation). These sequences are mainly for testing the basic tracking performance of the framework. An example of a sequence of this type is shown in figure 4.1.

**Medium** - These sequences contain total or partial object occlusion and changes in object appearance (this includes rotation, scale, or a combination of both). These sequences show strengths and weaknesses of visible and thermal infrared tracking. For example, very dark or light sequences where the thermal infrared tracking should not have a problem with in contrast to the visible spectrum track-
The sequences should be easy to track the object of interest in either visible or thermal infrared spectrum. An example of a sequence of this type is shown in figure 4.2.

**Figure 4.1:** Image (a) and (b) show an example of an easy sequence recorded within visible and thermal infrared spectrum. The green bounding box is the ground truth data.

**Figure 4.2:** Image (a) and (b) show an example of a medium sequence recorded within visible and thermal infrared spectrum. The green bounding box is the ground truth data.

**Hard** - These are sequences without good contrast between the object and background, where the object can disappear from view for longer periods of time and change appearance rapidly. Several highly similar objects may appear. These sequences were intended to test the robustness of the tracking framework, where excellent results can not be guaranteed. An example of a sequence of this type is shown in figure 4.3.

All sequences were captured with two cameras, which each recorded a video for the thermal infrared spectrum and one for the visible spectrum. The thermal infrared videos were captured with an FLIR T640 camera, with a resolution of 640x480 pixels and a field of view of 25\(^\circ\). It captures the scene in the LWIR band of 7.5 - 14 \(\mu\)m with a variable frame rate of \(\sim 15\) fps. The visible spectrum videos were captured with a Panasonic HDC-SD80, with a resolution of 1920x1080 and a frame rate of 25fps.

Recording video in two different parts of the electromagnetic spectrum means that some things that can be clearly seen in the visible part of the spectrum are
4.1 Video Acquisition

![Image (a) and (b) show an example of a hard sequence recorded within visible and thermal infrared spectrum. The green bounding box is the ground truth data.](image)

less visible in the thermal infrared part, and vice versa. The technology as it stands today is also a limiting factor on the video quality, where visible spectrum cameras often has a big advantage of taking faster, better and higher resolution images and video for a fraction of the cost. Some of these differences are more of an issue in tracking than others:

**Resolution** - One of the more obvious limitations when recording thermal infrared video is the resolution of the camera, which is rarely higher than 640x480 pixels. This stands in contrast to a visible spectrum camera which can often record in excess of 1920x1080 pixels with today’s technology. This resolution limitation affects tracking because fewer image pixels are pertaining to the object of interest, giving a less detailed view of the object. This can in turn lead to a worse object model and worse tracking capabilities.

**Frame Rate** - The frame rate of thermal infrared videos is often lower than that of their visible spectrum counterparts. If an object moves at a fast pace, it may be missed entirely by the camera if the frame rate is too low. This only becomes a problem if the object or camera is moving fast, and modern infrared cameras often have a frame rate of over 10fps, which means that this problem is of little worry when tracking.

**Field of View** - The field of view designates the extent of the observable world that is seen at each frame, and is most often measured in degrees. A narrow field of view means that the object must be far away to be seen completely by the camera. Thermal infrared cameras often have a narrow field of view than visible spectrum cameras. This parameter should have negligible impact on tracker performance, as long as the object is visible in the frame.

**Contrast** - The image contrast of thermal infrared video is somewhat lower than that of visible spectrum cameras, due to the fact that heat is more evenly distributed than color and that heat diffuses (in contrast to color and shape). This lower contrast can make it more difficult to track objects, but interesting objects that are usually tracked often distinguish themselves from the background by being (significantly) warmer or cooler. This feature is often used in heat-tracking,
where the hottest or coolest point in an image is tracked. This is a reasonable tracking method, if the object does not change its heat signature over time, which TLD should be able to handle.

Darkness is an area where the thermal infrared camera excels, and the visible spectrum camera falters. Objects radiate thermal energy even in darkness, which the thermal infrared camera can measure. This means that scenes with low lighting will prove to be easier to track in thermal infrared than in the visible spectrum. TLD uses illumination invariant features in the ensemble classifier so the framework should be able to handle some lightning differences, but in total darkness the visible spectrum camera can not see much, and the resulting image contains a large amount of noise.

**Dead Pixels** - Due to the manufacturing process and more fragile elements of a thermal infrared camera, there are often some dead pixels in the sensor array. This problem can in the worst case affect tracking performance, but the number of dead pixels is often small enough to be of a non-issue.

**Obstructions** - Some materials are transparent in the visible spectrum, while being opaque in thermal infrared. Ordinary glass is an example of this. This means that tracking through windows is out of the question in thermal infrared, where only the cameraman’s own reflection will be shown. Due to this limitation, no sequences were recorded through glass or had the object of interest behind such materials.

The resolution from both cameras was resampled to a common format of 640x480 pixels. Due to the visible spectrum camera capturing in a different aspect ratio than the thermal infrared camera, and also having a larger field of view, some cropping of the video (removal of parts of the image area) was also done on the visible spectrum videos. This was done to obtain a higher degree of correspondence between the thermal and visible spectrum video footage, in both resolution and field of view. Ground truth was annotated by hand for all sequences. The camera setup is shown in figure 4.4.

![Figure 4.4: Image (a) and (b) show the camera setup from front and back.](image-url)
4.2 Recorded Sequences

Seven sequences were recorded with both the visible spectrum camera and the thermal infrared camera. All sequences are presented in table 4.1 along with a short description of what the sequence contains, and a snapshot of each sequence is shown in figure 4.5.

Sequence one depicts two people, walking and meeting each other, then walking away.

Sequence two is of two people walking away from the camera and scaring birds in the background.

Sequence three depicts a car driving around in a parking lot. This sequence is considered hard due to the fact that the object of interest changes appearance, it includes full and partial occlusion and contains many similar objects.

Sequence four is of a person walking from a dark area into light, while also taking off his glasses. This sequence shows the different tracking capabilities of thermal infrared imaging and visible spectrum imagery in darker scenes, and also shows where TLD can learn new appearances of objects even in varying lighting situations.

Sequence five shows two people stopping and talking to each other. One of the difficulties with this sequence is the relative similarity of the faces of the people in the thermal infrared spectrum.

Sequence six shows different colored spheres on the ground. They all have a similar shape and IR signature, which makes them hard to track both in the visible spectrum and thermal infrared.

Sequence seven shows two books, which then change places. The difficulty here lies in the infrared spectrum where the books have a very similar IR signature, but can easily be differentiated in the visible spectrum.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Difficulty</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Easy</td>
<td>Two people walking, meeting</td>
</tr>
<tr>
<td>2</td>
<td>Easy</td>
<td>Two people walking, scaring birds</td>
</tr>
<tr>
<td>3</td>
<td>Hard</td>
<td>Car, full occlusion, high object similarity, high appearance change</td>
</tr>
<tr>
<td>4</td>
<td>Medium</td>
<td>Person, high illumination change</td>
</tr>
<tr>
<td>5</td>
<td>Medium</td>
<td>Two people talking, full occlusion, high object similarity</td>
</tr>
<tr>
<td>6</td>
<td>Medium</td>
<td>Spheres, high object similarity</td>
</tr>
<tr>
<td>7</td>
<td>Medium</td>
<td>Two books, similar IR-signature, full occlusion</td>
</tr>
</tbody>
</table>

Table 4.1: Table of Sequences.
4.3 Performance Measures

To measure the performance of the tracking framework, it must be stated if the resulting bounding box is correctly tracked or not. This is estimated for each frame of the sequence and is found out by comparing the resulting bounding box to the ground truth, and this results in a number of different measures:

**Overlap** - Overlap is defined as the ratio between the intersection and union of the estimated bounding box and the ground truth bounding box, $\text{overlap} = \frac{\text{BB}_{\text{res}} \cap \text{BB}_{\text{GT}}}{\text{BB}_{\text{res}} \cup \text{BB}_{\text{GT}}}$. $\text{overlap} = 0$ if no detection is annotated in the ground truth data for the current frame.

**True Positive (TP)** - The tracking is considered as a true positive if $\text{overlap} > 50\%$ and a detection is annotated in the ground truth data for the current frame.
4.4 Parameter Selection

**True Negative (TN)** - If no detection is annotated in the ground truth data for the current frame and the tracker yields no detection, the frame is considered a true negative.

**False Positive (FP)** - If the tracker yields a detection when none should be present, the result is considered a false positive.

**False Negative (FN)** - If the tracker yields no detection where one should be present, the result is considered a false negative.

**Both FN and FP** - If the tracker results in a bounding box that has overlap < 50% and a detection is annotated in the ground truth data for the current frame. An example of this result is shown in figure 4.6.

![Figure 4.6: Example of a detection resulting in a false positive and a false negative. The green bounding box is the ground truth data and the white bounding box is the tracker result.](image)

Performance of the tracking framework was based on a number of different measures according to the possible outcome of the resulting bounding box:

**Recall** – The number of true positives divided by the number of true positives and false negatives, \( R = \frac{TP}{TP + FN} \). It has a maximum score of 1 if the object is visible in the entire sequence and also detected in every frame.

**Average Localization Error** – This is a measure of the average distance between the center of the tracked bounding box and the ground truth bounding box. The distance unit used is pixels, and it should have a low value for correct tracking.

### 4.4 Parameter Selection

The main purpose of this master thesis was to study the performance of TLD, test and suggest improvements, while also evaluating which design choices are different for thermal infrared video and visible spectrum video.
The current framework is controlled by a number of different parameters: $L$, the number of ferns and $Q$, the number of features per fern; $\theta^+$ and $\theta^-$, the upper and lower learning thresholds; the detector thresholds for the variance classifier, ensemble classifier and nearest-neighbor classifier; and finally parameters for the scanning-window grid generation.

The detector thresholds mainly affect the speed of the framework. Lowering them will let more subwindows pass through each cascade step and be evaluated by a more complex method, thereby slowing the whole framework down. By having fairly low detector thresholds, we consider all relevant subwindows, although slowly, so those thresholds are therefore not considered in the parameter selection and instead set to a constant of 0.5.

To find parameter values where tracking in visible spectrum video or thermal infrared differ, several tests on each sequence were carried out. Each test was carried out five times, and the average performance measures, along with their standard deviations, were displayed.

This testing procedure is illustrated in algorithm 4.1.

```
Algorithm 4.1 Test protocol

Input: #sequencesToEvaluate, #parametersToTest, #valuesToTest

1: for $x \in$ #parametersToTest do
2:     for $k \in$ #sequencesToEvaluate do
3:         for $l \in$ #valuesToTest do
4:             for $m \in [0, \ldots, 5]$ do
5:                 $p_{x,k,l,m} \leftarrow p_{x,k,l,m} \cup recall_{x,k,l}$
6:             end for
7:             $m_{x,k,l} \leftarrow mean(p_{x,k,l,m})$
8:             $std_{x,k,l} \leftarrow std.deviation(p_{x,k,l,m})$
9:             $A_{x,k} \leftarrow A_{x,k} \cup m_{x,k,l}$
10:       end for
11:      $B_{x,k} \leftarrow max(A_{x,k})$
12:  end for
13: $x_{final} \leftarrow x_{final} \cup mean(B_{x,k})$
14: end for
```

4.4.1 Ensemble Classifier

Breiman [4] showed that random forests do not overfit as more trees are added, but produce a limiting value of the generalization error. This means that increasing the number of trees does not decrease the system recall. The question is rather how few ferns are required to produce a model with reasonable performance.

Another factor which affects the performance of the ensemble classifier is the number of features per fern, $Q$, which has to do with how much the data is assumed to be correlated. For large values of $Q$, the curse of dimensionality arises, meaning that the amount of training data increases with $Q$, and becomes
intractable. In contrast, lower values of $Q$ ignore correlation in the data. Another aspect to take into account is the memory requirement when $Q$ is large; for $Q = 20$, a training set of $2^Q = 1,048,576$ entries for conditional probabilities has to be stored for each fern. Ideally, the value of $Q$ should be tailored to each sequence to get the best performance, but a value that gives reasonable results can be obtained by averaging.

**Fern depth**

As previously discussed, the number of ferns, when high enough, does not make the model overfit the data. A higher number of ferns (50) was therefore chosen and a good value of $Q$, the fern depth, was selected based on an evaluation of several tests. The tests were made in the range of 1-17 features. The data in table 4.2 shows the highest recall rate for each sequence tested. The results of each individual test on the whole dataset can be found in appendix A.

<table>
<thead>
<tr>
<th>Seq.</th>
<th>Video</th>
<th>Recall</th>
<th>#Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Visible</td>
<td>0.68</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>0.56</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Visible</td>
<td>0.66</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>0.71</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>Visible</td>
<td>0.21</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>0.25</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Visible</td>
<td>0.34</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>0.84</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>Visible</td>
<td>0.55</td>
<td>1-5,15-17</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>0.64</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>Visible</td>
<td>0.45</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>0.61</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>Visible</td>
<td>0.54</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>0.39</td>
<td>17</td>
</tr>
</tbody>
</table>

*Table 4.2: Highest recall against fern depth for evaluated sequences.*

The most striking difference is the recall rate on sequence four, where the thermal infrared tracking performed far better than visible spectrum tracking. This was somewhat expected, due to the drastically changing lighting conditions in the scene, as TLD could not really learn the object appearance correctly in the visible spectrum video. This is of course not a problem in thermal infrared video, which does not need visible light in order to track the object. Interestingly, the thermal infrared tracking also had a more consistent pattern of high recall throughout different choices for the fern depth, in contrast to visible spectrum tracking which sometimes fail after some 12 features. This is probably due to overfitting to data in the ensemble classifier.

Sequence two produces a good recall rate even for low values of $Q$ in the visible spectrum video, and is consistent throughout several values for $Q$. The thermal
video needs more features per fern, over 14 to get a good recall rate. This is probably due to the somewhat lower contrast in the thermal infrared video, and the detector not able to distinguish object from background.

Sequence three produces a low recall for both the thermal and visible spectrum video, probably due to the fact that the object changes appearance rapidly and there are many similar objects in the scene.

In sequences five and six, increasing the fern depth in visible spectrum video did not make the system perform better, but rather worse up to a point. This is probably due to the detector choosing the wrong object as a final result per frame, because there exist similar objects in the sequence. This leads to many false positives, thus lowering the recall. Thermal tracking in these sequences performed better and more consistently, which was not expected, since the fact that the objects look very similar in both spectra.

Sequence seven produces a low recall for both the visible spectrum and thermal infrared video. Both required a high number of features for a satisfactory result.

In total, an average of eight features per fern for visible spectrum video and ten features for thermal infrared were deemed satisfactory for $Q$. These values were then used in all subsequent testing.

**Number of Ferns**

For the number of ferns $L$, a lower number will mean that the system evaluates subwindows that passes the variance classifier faster. We therefore want $L$ to be as low as possible without affecting system performance. The number of ferns was evaluated on sequence four, five and seven. $L$ was varied between 1-25 and figure 4.7 shows the recall rate for these different values of $L$.

Figure 4.7 shows that a relatively low number of ferns was needed in most sequences to achieve the same recall rates as in table 4.2. Good recall rates were achieved using as few as 2-5 ferns, and then approaching a maximum value. The results for the thermal infrared video for sequence seven show no significant improvement when varying the number of ferns. The visible spectrum video for sequence five shows a decrease in recall rate when adding more ferns to the forest, due to the fact that the detector locates a false positive and sticks with it. As in [12], a value of $L = 10$ was set as a compromise between speed, recall and memory consumption for both visible spectrum and thermal infrared videos. This value for $L$ was used in all subsequent testing.

**4.4.2 Number of Subwindows**

By changing the number of subwindows to evaluate in the detector cascade, the search area can be reduced or extended. This can have positive effects on both the speed and recall of the framework, if chosen with care. If a low number of subwindows are evaluated, the speed will improve but some possible true positives will be missed. If a high number of subwindows are evaluated, the speed
4.4 Parameter Selection

![Graphs of recall rates for different numbers of ferns.](a) Seq. 4 Visible

(b) Seq. 4 Thermal

(c) Seq. 5 Visible

(d) Seq. 5 Thermal

(e) Seq. 7 Visible

(f) Seq. 7 Thermal

**Figure 4.7:** Recall rate for different number of ferns.

will be lowered but more potential true positives will hopefully be found. Also, the risk of false positives will increase.

**Margin Distance**

The margin distance controls how densely subwindows are spaced. This value is currently set to $\frac{1}{10}$ of the initial bounding box width and height respectively.
The margin distance has an impact on the number of subwindows evaluated in the detector in the detector cascade, which affects both the learning and detector speed.

In appendix B, the results of an evaluation on all seven sequences is shown with regard to varying margin distance and recall rate. The margin distance was varied in an interval between \([0.1, \ldots, 0.4]\). The calculation time of the detector, which is not shown in the graphs, was decreased when the margin distance was increased due to the lower number of subwindows to evaluate. A higher margin distance also affected the recall rate in a negative way, which was to be expected. In the cases where the recall rate increased with margin distance, this could be attributed to a stronger reliance on tracking by displacement which was a better fit for those sequences. More potential true positives were missed when too few subwindows were evaluated, so this parameter is a compromise between speed and recall. A value of \(\frac{1}{10}\) for the margin distance was still considered to be suitable.

### 4.4.3 Learning

Learning is a crucial part of the TLD framework, as it allows the system to adapt to changes in object appearance. During initial learning, a model of the object of interest is created. This model is then updated in response to changes in appearance. Without an update of this model, we can expect a decrease in recall rate for scenes where the object of interest changes appearance, such as in sequence three and four.

**With/without Runtime Learning**

A test was carried out where learning was only done using the first frame of a sequence, which meant that the detector was incapable of updating the object model during runtime. In sequences with high variability in object appearance, performance was expected to decrease. The recall rate is shown in table 4.3.

In table 4.3, it can be seen that the outset of lower recall rate with no learning during runtime is generally correct, with some anomalies. In the visible spectrum video for sequence four and thermal video for sequence one, the recall rate actually goes up with no learning during runtime. The thermal video for sequence one can be explained by the fact that the detector chooses the wrong object during runtime, and learns that objects’ appearance.

Overall, table 4.3 show that the learning part of the framework works as intended, but no significant difference can be seen in most videos. This is probably due to the fact that most recorded sequences did not include a high object variability, and the framework did not need more templates than the first to work with to achieve a good recall rate.
4.5 Implemented Framework Improvements

Motion Model Prediction

A model for tracking the motion of the object would benefit the framework in some situations. For example, sequence five has objects with high similarity, where a motion model could stabilize the system by not choosing a false positive.

A basic motion model prediction was implemented, where the last 20 bounding box positions were saved. The mean and variance of the movement between positions was calculated, and a gaussian distribution was drawn from this data, using the last known object position as a starting point. This distribution was multiplied with the conservative similarity of each bounding box to evaluate in the result integration. The highest scoring bounding box was then chosen as the final bounding box.

The motion model was tested on sequence four and five. A comparison with and without the model is shown in appendix C, where the entire sequence is run once and the localization error is shown over time. The average localization error for these sequences is shown in table 4.4.

Table 4.4 shows that the motion model often increased performance, as the position error was significantly reduced. The motion model is, although crude, a good tool for estimating the object position.

In sequence five, without the motion model prediction, the detector finds a better match to the object model within the other persons face, which is similar. This

<table>
<thead>
<tr>
<th>Seq.</th>
<th>Video</th>
<th>Recall w. learning</th>
<th>Recall w/o learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Visible</td>
<td>0.68</td>
<td>0.64</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>0.56</td>
<td>0.69</td>
</tr>
<tr>
<td>2</td>
<td>Visible</td>
<td>0.66</td>
<td>0.61</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>0.71</td>
<td>0.43</td>
</tr>
<tr>
<td>3</td>
<td>Visible</td>
<td>0.21</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>0.25</td>
<td>0.22</td>
</tr>
<tr>
<td>4</td>
<td>Visible</td>
<td>0.34</td>
<td>0.49</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>0.84</td>
<td>0.7</td>
</tr>
<tr>
<td>5</td>
<td>Visible</td>
<td>0.55</td>
<td>0.13</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>0.64</td>
<td>0.62</td>
</tr>
<tr>
<td>6</td>
<td>Visible</td>
<td>0.45</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>0.61</td>
<td>0.5</td>
</tr>
<tr>
<td>7</td>
<td>Visible</td>
<td>0.54</td>
<td>0.36</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>0.39</td>
<td>0.39</td>
</tr>
</tbody>
</table>

Table 4.3: Comparison of mean recall values with and without learning during runtime.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Visible</td>
<td>79.7</td>
<td>150.2</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>39.3</td>
<td>21.4</td>
</tr>
<tr>
<td>5</td>
<td>Visible</td>
<td>38.4</td>
<td>173.6</td>
</tr>
<tr>
<td></td>
<td>Thermal</td>
<td>31.9</td>
<td>86.5</td>
</tr>
</tbody>
</table>

Table 4.4: Average localization error with and without motion model prediction.

produces a low recall and a high average localization error. The better result with the motion model prediction can be seen in frame 80 of the thermal infrared video, where the motion model is able to predict the incoherent movement according to the motion model and compensate for it. The results for the visible spectrum video were similar, where the presence of the motion model prediction improved tracking performance.

Sequence four did not show the same improvement when the motion model prediction was enabled. It still lowered the average localization error, but not as much as in sequence five. Especially the thermal infrared video did not show improvement with the motion model. This can be explained by the relative ease of tracking in thermal infrared for this video, with a good contrast between the object and the background, and the motion model masking the correct detections found by the detector.
5 Conclusions and Improvements

5.1 Conclusions

During implementation and testing of Tracking-Learning-Detection, some thoughts and recommendations on how the framework could be improved arose. During evaluation of thermal infrared tracking versus visible spectrum tracking, the somewhat obvious conclusion is that it is preferable to work in the thermal infrared spectrum when trying to track objects in low light conditions. The benefit of thermal infrared tracking also extends into sequences with varying lighting conditions. The object could then disappear into shadows or other obstructions, such as in sequence four. Tracking in thermal infrared is also preferable when foliage is an issue, where an object with a strong enough heat signature will still show through.

The situations where thermal infrared tracking using TLD does not improve the tracking performance are when the objects are highly similar in shape and thermal intensity, such as in sequence seven (see table 4.2). The visible spectrum tracking then outperformed the thermal infrared tracking, due to being able to distinguish objects from each other by other means, such as the covers of books, which can not be differentiated in the thermal infrared spectrum. TLD has also been shown to distinguish between different people in photographs [1], which is something the thermal infrared tracking is not able to handle at all.

Situations where TLD still falters regardless of spectrum are when the object of interest has a high shape variability, and the aspect ratio of its bounding box changes over time. This is something that could be improved by more sophisticated models in the detector cascade, such as incorporation of a variable aspect ratio. This of course would mean that more subwindows would have to be evalu-
ated, thus increasing the computational complexity.

TLD as implemented now can not handle out-of-plane object rotation. By estimating this rotation, it should improve tracking capabilities if the object is rotated. This is something which [13] is able to do, and an extension of TLD to handle this would be interesting to evaluate.

5.2 Improvements / Future Work

Some ideas of improving the TLD tracking framework arose during the implementation phase. These ideas are presented with a short summary in this section.

5.2.1 Background Model

A background model for handling static environments has been tested together with TLD as a first part of the detector cascade [12]. This was done using a simple background subtraction method, and an extension of this to handle semi-static environments would be especially useful in surveillance situations. A background mixture model could be useful, calculating the mean and variance of each pixel in the scene to estimate the position of out-of-place objects [15]. This model could either be used as a part of the detector cascade, or as an initial object detector for input to TLD.

5.2.2 Sensor/ Image Noise Model

As of now, image and sensor noise are not taken into consideration. If the camera parameters are known this could be handled better, and an image of higher quality could be evaluated in the framework. This would hopefully make the tracking more precise. A simple median filtering on the thermal infrared videos could be enough, due to the salt-and-pepper nature of noise in thermal infrared images.

5.2.3 Variance Subdivision Classifier

The framework speed is heavily dependent on the number of steps and their complexity in the detector cascade. By lowering the number of subwindows that have to be evaluated using more complex methods, the speed should improve. One idea is to use the variance classifier once again, but subdivide it into different parts of each subwindow. The reason for using the variance classifier again is its speed, which could be put to further use.

The idea is that each subwindow that passes through the variance classifier in the detector cascade is evaluated by subdividing the window into four equal parts. Each part is then individually tested against a variance threshold, and the sub-window passes if at least two parts are above the threshold. This approach is similar to using a cascade of Haar-filters [17]. This idea is presented in figure 5.1.
5.2.4 Adaptive Fern Depth

An adaptive depth $L$ of the fern forest classifier should be able to handle a wide array of different training examples more gracefully. In the initial training, if a low number of training examples are generated, $L$ should be small, or many entries in the fern may not see any training examples, thus leading to a noisy classification.

$L$ can instead be increased during runtime learning, when more training examples are generated. It is important though that $L$ is not too high or the curse of dimensionality will be a problem, where not enough training examples are available to the number of entries.

5.2.5 Multispectral Tracking

Tracking in the visible or infrared spectra each have their own benefits and disadvantages. By using both visible spectrum and thermal infrared at the same time, and using the strengths of each spectrum to its best extent, it should be possible to construct a better tracking method that is very robust under varying conditions.

For example, when the sun goes down and light is dim, infrared tracking is probably a better candidate for finding and tracking interesting objects, while visible spectrum tracking is better when the heat and shape of different objects are similar. One of the problems with this approach is knowing in which situations to track in visible or thermal infrared, and a good evaluation method for this is needed.

Another problem lies in determining object correspondence between spectra, in order to get correct tracking. This could easily be solved by an initial calibration if both cameras are static during recording. The problem of course becomes much harder if this condition is not fulfilled.
5.2.6 Improvement of Median-flow Tracking

The median-flow tracker uses an LK-tracking method as described in section 2.3 to track individual feature points. One of the problems with median-flow tracking is that individual feature points inside the current bounding box may be part of the background and thus will not move with the tracked object. If a large amount of background feature points are tracked, the estimated bounding box displacement will "drag", meaning that the displacement will be incorrect.

The method could perhaps be improved by calculating an object mask inside the bounding box, and using this to estimate where the feature points should be placed to not track points in the background. The mask could be calculated by using a background model on the bounding box itself or using some other method.

5.2.7 Improved Motion Model

The motion model described in section 4.5 is, at best, a simple approximation of the object motion. This model could be improved by filtering and better prediction of the data using, for instance, a Kalman filter. The motion model could also be used as a part of the detector cascade, as a first step in evaluating which object locations are most likely, thus lowering the amount of subwindows to evaluate further.
Figure A.1: Sequence 1: Recall rate versus fern depth.
Figure A.2: Sequence 2: Recall rate versus fern depth.

Figure A.3: Sequence 3: Recall rate versus fern depth.

Figure A.4: Sequence 4: Recall rate versus fern depth.
Figure A.5: Sequence 5: Recall rate versus fern depth.

Figure A.6: Sequence 6: Recall rate versus fern depth.

Figure A.7: Sequence 7: Recall rate versus fern depth.
Figure B.1: Sequence 1: Recall rate for varying margin distance.
Figure B.2: Sequence 2: Recall rate for varying margin distance.

Figure B.3: Sequence 3: Recall rate for varying margin distance.

Figure B.4: Sequence 4: Recall rate for varying margin distance.
Figure B.5: Sequence 5: Recall rate for varying margin distance.

Figure B.6: Sequence 6: Recall rate for varying margin distance.

Figure B.7: Sequence 7: Recall rate for varying margin distance.
Figure C.1: Sequence 4: Localization error over time, with and without motion model prediction.
Figure C.2: Sequence 5: Localization error over time, with and without motion model prediction.


Upphovsrätt

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

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

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

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

Copyright

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

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

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

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

© Magnus Stigson