Multi-Sensor Human Detection on an Intelligent Security Robot

A study on combining vision and laser scan data

Yannick Kathmann

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Multi-Sensor Human Detection on an Intelligent Security Robot

A study on combining vision and laser scan data

Master of Science Thesis

For the degree of Master of Science in Mechanical Engineering at Delft University of Technology

Yannick Kathmann

March 31, 2017

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by Robot Security Systems. Their cooperation is hereby gratefully acknowledged.

Abstract

Security robot SAM can patrol indoor environments, determine whether everything is as it should be and report disturbances to the authorities. One of the purposes of SAM is to detect humans and ask for their identification. Currently this human detection is done through movement detection. This thesis aims to combine visual data and distance measurements from a Laser Range Finder (LRF) to recognise the presence of humans. The data from the LRF is used to find regions of interest in order to reduce the load on the visual data analysis. Deep learning convolutional neural networks have shown incredible results on visual recognition tasks in recent years and this framework is used to recognise humans in visual data. A laser-camera calibration is done to be able to use the LRF data for region proposals. This calibration is done with the aid of checker board patterns, in order to translate the laser-based distance measurements to pixel coordinates in the visual data. Experiments for the region proposals are done with clustering the LRF data (passive detection) and deep learning (active detection). Convolutional Neural Network (CNN) experiments are done with different solver types (AdaGrad/AdaDelta), data types (RGB/HSV/Grayscale) and network shapes (ConvPool/ConvConv) and serve to show the feasibility of deep learning for the human detection task. These experiments are extended by an extra set of experiments with three datasets from a new location to show the strength of the learned network as well as the feasibility of using new data to finetune a pre-learned network to a new location.

Table of Contents

List of Figures

List of Tables

Preface

The idea of doing my thesis work in this field of research came to be after following a course on 3D robot vision. I had a lot of fun detecting and tracking objects in images and this is how I came into contact with Prof. Pieter Jonker. In an earlier stage he redirected me to Robot Security Systems (RSS) for a potential internship, but back then in 2014, I had not acquired sufficient ECTS to start my 'second' year of the master program. In 2016, having met the requirements to start graduating, I approached RSS with the question if they had any interesting graduation assignments available and after a conversation with Jochem Verboom and Eelko van Breda it turned out they had. At RSS they had just started working on a new software architecture of which a human detection framework is a vital part and this meant I had a new challenge ahead of me, which resulted in this thesis.

Acknowledgments A special thanks to the people at RSS for giving me the space and time to conduct this research. I also want to thank Jens Kober, Floris Gaisser, Jochem Verboom and Pieter Jonker for their comments and supervision.

“A robot may not injure a human being or, through inaction, allow a human being to come to harm.”

— *Isaac Asimov*

Introduction

Robots are the future, they are being used in more applications each year. Service robots are an interesting topic of research and can come in many shapes and sizes. One prerequisite for a service robot is to be able to understand its environment and plan its actions accordingly. Many researchers are engaged in this topic of artificial intelligence and in recent years there have been interesting developments in the field of deep learning.

At Robot Security Systems (RSS), robot SAM is developed, who can map and survey indoor environments. For this purpose it is necessary for the robot to act intelligently based on the input that it gets, which means it needs to understand what is happening in the world around it. To do so, it has four RGB cameras for a 360° view and a 2D Laser Range Finder (LRF). A key function of this robot is to be able to detect the presence of humans. Currently, the robot at RSS does so by concluding that any movement detected between two captured images must be a human, unless its only in the bottom half of the image, which would indicate it is a smaller animal such as a rodent.

A subset of deep learning, Convolutional Neural Network (CNN), is particularly interesting for human detection using visual data. The CNN works much like the visual cortex of a human by first filtering the images and extracting hierarchical features which can then be used to classify the contents of the image. What makes a CNN strong, is the fact that it learns its own features instead of using handcrafted features. Although a sufficiently deep CNN can almost always fit the training data perfectly, this usually does not generalize well to data it has not seen before. In other words, a CNN is very sensitive to overfitting. Extensive research has shown that there are many ways to regularize the network to reduce this overfitting problem.

Another problem is that human detection from only visual data can be slow and if the framework is to be used on a real robot, it needs to be able to operate in real-time. The main reason of the slow speed is the huge amount of images that need to be analysed if a sliding window classifier is used. This can be reduced significantly by using data from the LRF to generate regions of interest in the visual data. On top of that, the distance measurements from the LRF can also give an estimation of the crop size, should the measurements represent a human.

This research aims to detect humans based on two data streams from both the RGB camera and the LRF. The distance measurements from the LRF are used to generate region proposals in the visual data. These frames are then analysed by the CNN to assess the probability of containing a human. In this thesis, the robot is assumed to be driving on a flat surface in an indoor environment, which simplifies the calibration between the LRF and the camera.

The thesis starts with some background theory on LRF-based detection in Chapter 1 and Convolutional Neural Networks in Chapter 2. Chapter 3 discusses the calibration and synchronisation of the sensors. Chapter 4 discusses the experiments with the LRF data and Chapter 5 shows the results of deep learning experiments with visual data. The thesis concludes with a discussion in Chapter 6.

Chapter 1

LRF-Based Detection

The Laser Range Finder (LRF) used for human detection performs 2D distance measurements in a 270° Field Of View (FOV). This chapter further elaborates on the possible ways the data from the LRF can be used for the purpose of human detection. The ultimate goal is to combine the distance measurements from the LRF with visual data from an RGB camera. The LRF data can be used in two different ways. The first is purely passive, as hypothesis generator for the visual data. The second possible use is to use the LRF data to actively search for patterns that would indicate the presence of humans in the observed space. Section 1-1 elaborates on the passive detection and how this can be implemented. In Section 1-2 the active detection methods are discussed. The chapter concludes with some remarks in Section 1-3.

1-1 Passive LRF Detection

To be able to detect objects from LRF data, the raw point data needs to be processed. Points that are believed to belong to the same object need to be clustered together and these clusters or segments can then be used to generate a Region of Interest (ROI) for the camera or perform LRF detection. Premebida [1] wrote a technical report on different segmentation methods and concluded there are two categories: Point-Distance Based Segmentation (PDBS) and Kalman-Filter Based Segmentation (KFBS).

1-1-1 Point-Distance Based Segmentation (PDBS)

The segmentation methods in this category are based on a simple criterion. If the Euclidean distance between two subsequent points from the laser scan is larger than a certain threshold, a new segment is started. If not, the point is added to the current segment. This distance threshold needs to be chosen carefully as a small threshold can lead to a large amount of small segments, ultimately giving too many region proposals. On the other hand, a too high threshold might not be able to distinguish human legs when standing against a wall and just take the legs within the same segment as the wall. Somewhere in between lies an optimum. This can be found empirically as is also discussed in Chapter 4.

1-1-2 Kalman-Filter Based Segmentation (KFBS)

The KFBS by Borges [11] assumes the laser scan to be a dynamic process that can be estimated by a state-space equation as follows:

$$\begin{aligned} \mathbf{x}_n &= \mathbf{A}\mathbf{x}_{n-1} + \mathbf{w}_n \\ z_n &= \mathbf{C}\mathbf{x}_n + v_n \end{aligned} \tag{1-1}$$

Where the state variable $\mathbf{x}$ represents the range measurements (r_n) and its derivatives with respect to the angle $(dr_n/d\phi)$, z represents the measurement variable and $\mathbf{w}$ and $\mathbf{v}$ represent zero-mean Gaussian white noise with covariance $\mathbf{Q}_n$ and R_n respectively. The matrices $\mathbf{A}$ and $\mathbf{C}$ are defined as follows:

$$A = \begin{pmatrix} 1 & \Delta\phi \\ 0 & 1 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

The standard linear discrete Kalman-filter first predicts the current state based on the last measured state:

$$\begin{aligned} \hat{\mathbf{x}}_{n|n-1} &= \hat{\mathbf{x}}_{n-1} \\ \hat{\mathbf{P}}_{n|n-1} &= \mathbf{A}\hat{\mathbf{P}}_n\mathbf{A}^T + \mathbf{Q}_n \\ v_n &= r_n - \mathbf{C}\hat{\mathbf{x}}_{n|n-1} \\ S_n &= \mathbf{C}\hat{\mathbf{P}}_{n|n-1}\mathbf{C}^T + R_n \end{aligned} \tag{1-2}$$

And then updates the measurement:

$$\begin{aligned} \mathbf{K}_n &= \hat{\mathbf{P}}_{n|n-1}\mathbf{C}^T S_n^{-1} \\ \hat{\mathbf{x}}_n &= \hat{\mathbf{x}}_{n|n-1} + \mathbf{K}_n v_n \\ \hat{\mathbf{P}}_n &= \hat{\mathbf{P}}_{n|n-1} - \mathbf{K}_n\mathbf{C}\hat{\mathbf{P}}_{n|n-1} \end{aligned} \tag{1-3}$$

The KFBS algorithm is described in Algorithm 1. The algorithm loops through all measured range data and for each point it checks whether it fits the current segment in a stochastic sense. If a threshold is reached, the current point is taken to be at the next segment and the filter is reset.

```
**Data:** N scanned points
**Result:** Extracted breakpoints
Initialize filter $(x_0, P_0)$.
for $k$ = 1:N do
    Calculate filter prediction Equations (1-2).
    Test gate-equation ($\chi^2$-test):
    if $\upsilon_n^2/S_n \geq D_{thd}$ then
        Extract breakpoint.
        Reset filter $(x = x_0, P = P_0)$.
    else
        Update observation using Equations (1-3).
    end
end
```

Algorithm 1: KFBS

1-1-3 Segmentation Results

Premebida analyses the results of several segmentation methods at the end of his technical report. A segmentation result of PDBS is shown in Figure 1-1 together with the indoor scene in which it was acquired. KFBS results are not shown in his report, but he concluded that they process the incoming data in a more on-line nature than PDBS, meaning that the points are analysed sequentially as they come in.

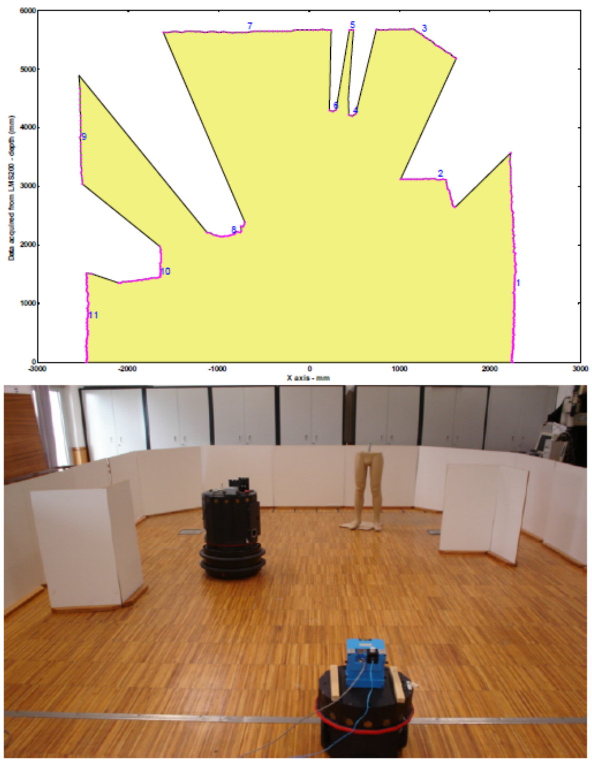


Figure 1-1: Results of PDBS [1]

1-2 Active LRF Detection

Where the previous section focussed on passive clustering of the data points, this section discusses the possibility of using this data for actual detection. Significant progress in this field was made by Arras [2] in 2007. As can be concluded from Figure 1-2, it is quite difficult to intuitively distinguish humans from other objects in a 2D LRF scan. However, as is shown by Arras, it turns out there are certain features from which humans can be detected in these scans. After using PDBS for segmentation, a 14-dimensional feature vector is computed for each segment, of which an example is shown in Figure 1-3. The LRF based features can then be classified using classification algorithms such as Support Vector Machines (SVM).

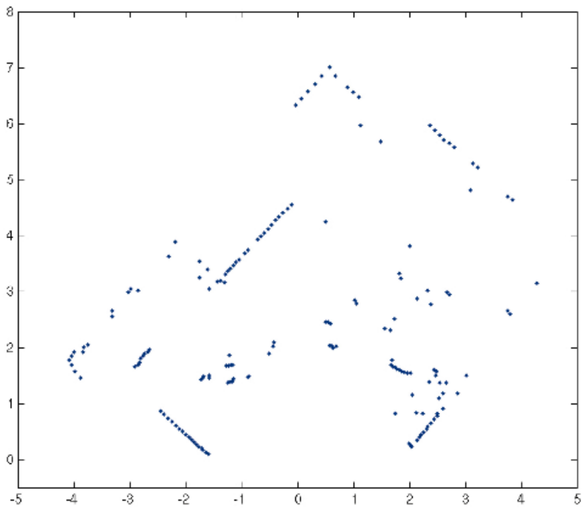
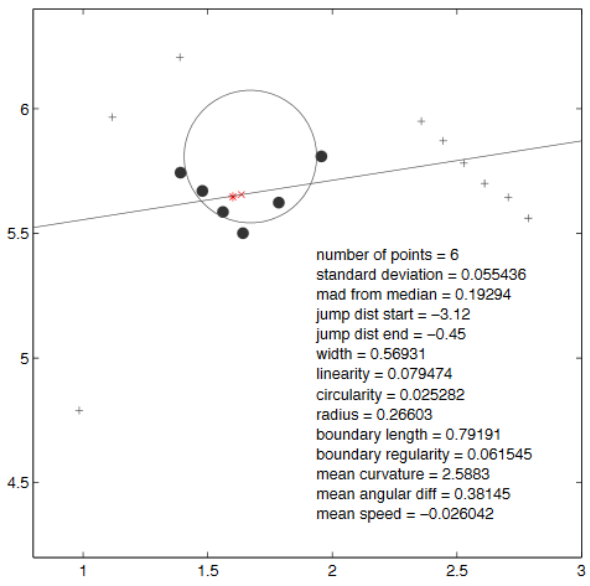


Figure 1-2: An example laser scan from an office [2]



number of points = 6
standard deviation = 0.055436
mad from median = 0.19294
jump dist start = −3.12
jump dist end = −0.45
width = 0.56931
linearity = 0.079474
circularity = 0.025282
radius = 0.26603
boundary length = 0.79191
boundary regularity = 0.061545
mean curvature = 2.5883
mean angular diff = 0.38145
mean speed = −0.026042

Figure 1-3: An example of a feature vector of one segment [2]

1-3 Concluding Remarks

This chapter discussed the segmentation and feature extraction from the LRF sensor. Results from various papers show that LRF-based features can succesfully be applied to detect humans. However feasible, human detection using a LRF has received less attention than vision-based approaches. Laser scan data is more useful for mapping applications, but can in the context of multi-sensor human detection be used to define regions of interest in the visual data. This is particularly interesting for big data applications such as a Convolutional Neural Network (CNN), which is discussed in Chapter 2. On top of that, the LRF also makes it easier to determine the location of detections with respect to the robot. A major downside of using a 2D LRF is the fact that detectable objects can easily be obscured. For instance, a human that is supposed to be detected can stand behind a box. Depending on how the segments are processed, the passive detection can still pass obscured objects as hypothesis to a vision-based classifier. The active detection, however, will have a hard time finding leg patterns if these legs are behind another object. Using the LRF as active detector will therefore probably only work if it is used parallel to a vision-based classifier in combination with an evidence fusion scheme. This type of framework is beyond the scope of this thesis.

Chapter 2

Convolutional Neural Networks

Chapter 1 discussed both active human detection using distance measurements from a Laser Range Finder (LRF) and using the same data to generate object hypotheses that can then be further analysed by a vision-based detector. This chapter explores the concept of Convolutional Neural Network (CNN), which in recent years has shown incredible results on visual recognition tasks. Its major strengths are multi-class detection and the fact that features and classification are jointly learned. A weakness is that the method is prone to overfitting if not carefully handled. This chapter elaborates on this principle by first explaining the use of neural networks (Section 2-1) and then moving on to the extended idea of convolutional neural networks (Section 2-2). This is followed by explanations on the training of the network by exploring loss functions (Section 2-3) and optimization methods (Section 2-4). Regularization techniques to deal with overfitting and sparse data are discussed in Section 2-5. The advantages of region proposals are discussed in Section 2-6 and some state-of-the-art CNN methods are shown in Section 2-7. The chapter concludes with some remarks in Section 2-8.

2-1 Neural Networks (NN)

A Neural Network (NN) consists of a network of artificial neurons that take as input a linear combination of weighted inputs and then output something if a certain activation function is triggered ('fired'). These inputs can either come from an external source (input layer) or from a preceding layer of neurons (hidden or output layer). A visual representation is shown in Figure 2-1.

2-1-1 Activation Functions

The activation function determines whether a given input to the neuron is enough to make it fire, i.e. give a value as output. Some activation functions have been used throughout the years, but the most common ones are the sigmoid and the Rectified Non-Linear Activation Unit (RELU). The latter has been shown by Alex Krizhevsky [8] to significantly decrease

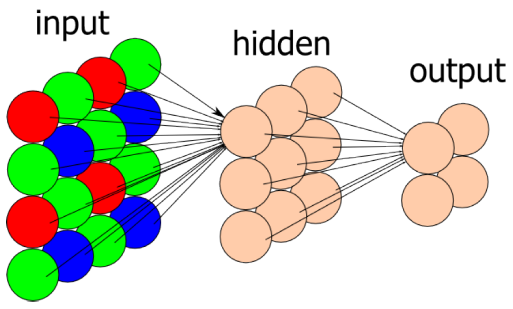


Figure 2-1: A neural network [3]

the training time and is now more commonly used than the sigmoid. A disadvantage of the sigmoid function is its small gradient close to the edges, which can lead to a significant slowdown of the already slow learning process. Both functions are shown in Figure 2-2.

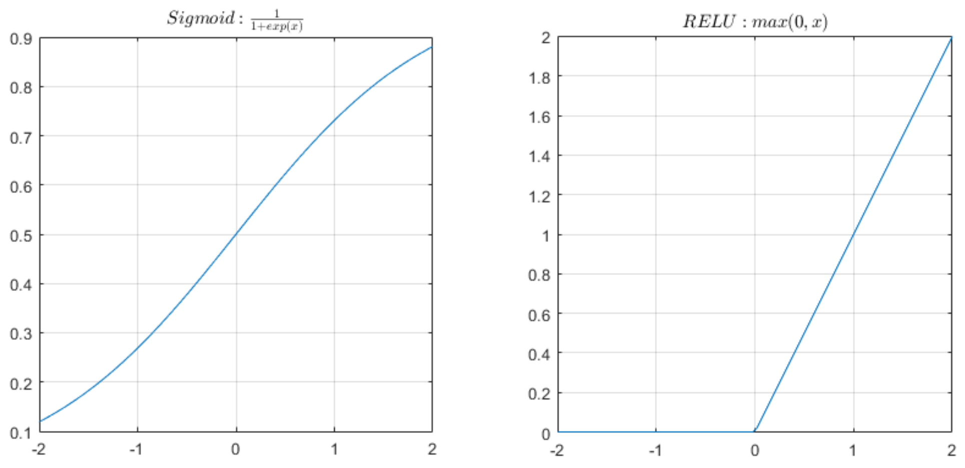


Figure 2-2: The sigmoid and the RELU activation functions

2-2 Convolutional Neural Networks (CNN)

This idea of a NN, that is based on the functionality of the human brain, can be extended to the functionality of the visual cortex. Just like we do with our own eyes, the input images are first filtered, before they are further processed in the brain (or the NN). By use of convolutional filters a hierarchy of visual features is learned and the local pixel level features are also scale and translation invariant. By learning several levels of features, the CNN is able to capture the essence of objects and this turns out to generalize quite well towards different representations of the same object. Until quite recently, the research on CNNs was minimal, simply because there was not enough computing power to handle these networks (there are often millions of parameters to train). Thanks to the ability to do parallel computing on GPUs and the development of software dedicated to using the GPU for such computations, this big data problem is not the issue it used to be. Of course there is still a limit to the amount of data that can be processed, but thanks to these developments, the bar has been raised significantly. An example of how a complete CNN looks is shown in Figure 2-3, which shows what is believed to be the classical CNN, designed by Yann Lecun to recognise handwritten digits.

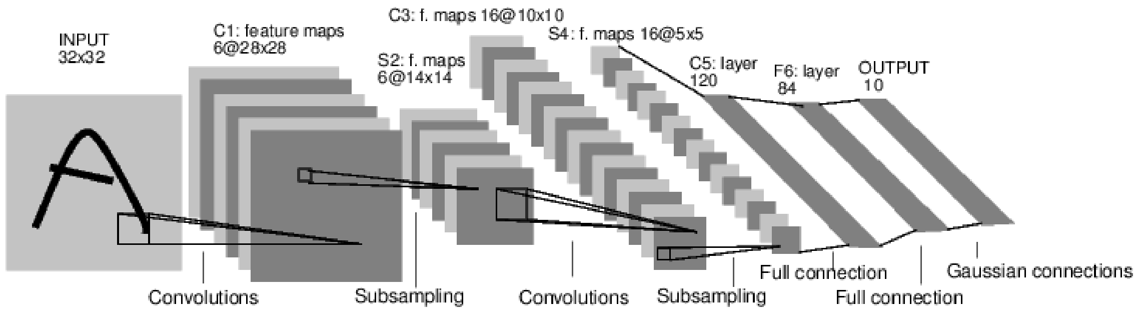


Figure 2-3: The architecture of LeNet-5 [4]

2-2-1 Convolutional Filtering

The first step in a CNN is filtering the image with convolutional kernels. Applying different kernels gives many different results. The resulting convolved images are called feature maps, because they represent certain features contained within the image like edges, corners, etc. Figure 2-4 shows an example of a 3x3 kernel being applied to an image. This kernel contains the weights that are multiplied with the corresponding pixel intensity. The sum of the 9 multiplications is then the input for the feature map that needs to determine whether this input is enough for the neuron to be activated. There are some things to keep in mind when applying this filter, the first of which is called the stride. The stride determines the step size in which the kernel is applied. A stride of 2 means that the kernel is moved 2 pixels at a time and will therefore lead to less convolved features than a stride of 1. Another thing is the choice between wide and narrow convolution. Wide convolution also applies the filter to the pixels at the edges, where part of the filter falls outside of the image. These parts are then taken to be zero (zero-padding). Narrow convolution only applies the filter to the sets of pixels that fall within the image.

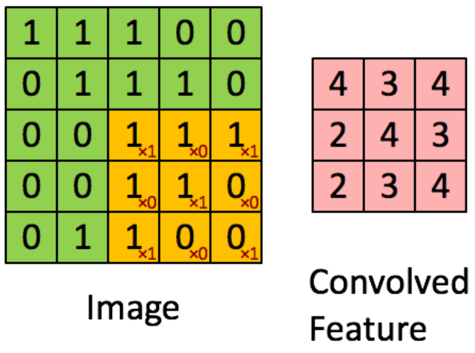


Figure 2-4: Applying a convolutional kernel [5]

2-2-2 Spatial Pooling

As can also be seen in Figure 2-3, the convolutional layer is followed by a subsampling, also known as spatial pooling. The intention of this pooling is to reduce the amount of data, whilst simultaneously keeping the important information. There are several ways to do this, the most common ones being average and max pooling. Similar to the convolutional kernel sliding over the image, now a window is sliding over the feature map that either computes the maximum or the average value. Another example is stochastic pooling proposed by Zeiler [12] in 2013. Because max pooling is sensitive to overfitting, stochastic pooling introduces a probability, which means the maximum value is still the most likely to be taken, but this is not always the case. The disadvantage of average pooling is that really strong activations are being ignored, therefore max and stochastic pooling seem like the preferred methods.

2-2-3 Fully Connected Layers

At the end of several subsequent steps of convolution and pooling there are usually some fully connected layers that transform the extracted features into a feature vector. The term fully connected refers to the fact that all neurons in the preceding layer are connected to all neurons in the next. Using fully connected layers increases the dimensionality of the problem exponentially so care needs to be taken with the implementation of these layers.

2-3 Loss Functions

By far the most computationally intensive step of human detection using a CNN is the network training. The idea of training the network is that it learns the weights in the entire network such that it can match input to a correct output. This training happens using a technique called backpropagation. All weights within the network are initialized randomly and then the input image is pushed through the network to produce a score for each class. These scores are used to classify the contents of the image. During the training of the network, the groundtruth of the image contents are known, which can then be used to compute an error of the current prediction compared to the groundtruth. This error is represented by a loss function, which can be computed in two different ways: The hinge loss in combination with a linear Support Vector Machines (SVM) classifier and the cross-entropy loss in combination with softmax classification. These are discussed in the subsections below.

2-3-1 Hinge Loss with a Linear SVM

Together with a linear Support Vector Machines (SVM) classifier, a hinge loss function [6] (2-1) can be used to define the error of the predictions. For each wrong class, the maximal value is determined of either 0 or the difference between the correct class score (s_{y_i}) and its own score (s_j) plus a certain margin Δ. These results are then summed to a number representing the loss over all the classes combined.

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta) \tag{2-1}$$

2-3-2 Cross-Entropy Loss with Softmax

The scores at the end of the network are not in the form of a proper probability distribution yet, which is why this last layer can have a special kind of activation function that was not previously described. The softmax function:

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}} \tag{2-2}$$

This will transform the scores at the end of the fully connected layers into a probability distribution. The softmax classification is combined with the cross-entropy loss [6] (2-3), where the exponentials of the class scores are summed and the log is taken from the result. Finally subtracting the class score of the correct class will give the cross-entropy loss.

$$L_i = -s_{y_i} + \log \sum_j e^{s_j} \tag{2-3}$$

2-4 Optimization Methods

The loss functions describe the accuracy of the predictions of the network. That leaves the question how these functions are minimized to be as accurate as possible.

2-4-1 Gradient-Based Optimization

The gradient of the loss to the weights serves as a measure of the extent to which the weights have contributed to this error and this gradient is used to determine where to look for the minimum. This can be visualized as standing somewhere in a mountain range and wanting to reach the lowest point. The gradient is then the steepness in a certain direction and this steepness is used by these optimization methods to decide where to look for the desired minimum.

Batch Gradient Descent The batch gradient descent algorithm [13] takes a step in the direction that provides the steepest descent. This gradient is computed over the entire dataset and is therefore really expensive in terms of computational resources, which means it is only feasible for relatively small datasets. The step size (or learning rate) is a hyperparameter that needs to be chosen carefully. Small steps may be more accurate, but can be cumbersome to compute. Large steps are dangerous, because of the risks of overstepping the minimum. The equation for one step of the algorithm is described as follows:

$$\theta = \theta - \eta \nabla_\theta J(\theta) \tag{2-4}$$

Where θ are the parameters within the network, η is the learning rate and $\nabla_\theta J(\theta)$ is the gradient of the cost function (or loss function) with respect to the parameters.

Minibatch Gradient Descent The minibatch gradient descent method picks a random subset of the training data and computes the gradient from this minibatch. Effectively, this is a similar method to the batch gradient descent, but applied to a smaller batch. This method is preferable if big datasets are used, because it significantly reduces the amount of data that needs to be processed per learning iteration.

2-4-2 Momentum-Based Optimization

Gradient-based optimization methods only take into account the gradient of the loss function and search for a minimum using only that information. However, a momentum term can be taken into the equation as well. Momentum-based optimization methods are discussed below.

Momentum-Based Gradient Descent The momentum-based gradient descent algorithm [13] not only takes into account the gradient, but also the momentum at a certain point. Effectively this brakes the descent in order to prevent overshooting the minimum. Another beneficial effect of the momentum-based descent is that learning slowdown is dealt with. The learning slowdown can be a problem with activation functions such as the sigmoid that has close to zero gradients at the outer bounds. A step in the momentum-based gradient descent algorithm can be described as follows, where γ represents the momentum term:

$$\begin{aligned} \nu_t &= \gamma \nu_{t-1} + \eta \nabla_\theta J(\theta) \\ \theta &= \theta - \nu_t \end{aligned} \tag{2-5}$$

Nesterov Accelerated Gradient In the momentum-based gradient descent method, both the momentum and the gradient are computed at the same instance. The Nesterov Accelerated Gradient algorithm looks ahead to where the momentum would take it and computes the gradient from that position. A comparison between the regular momentum-based and the Nesterov algorithm is shown in Figure 2-5.

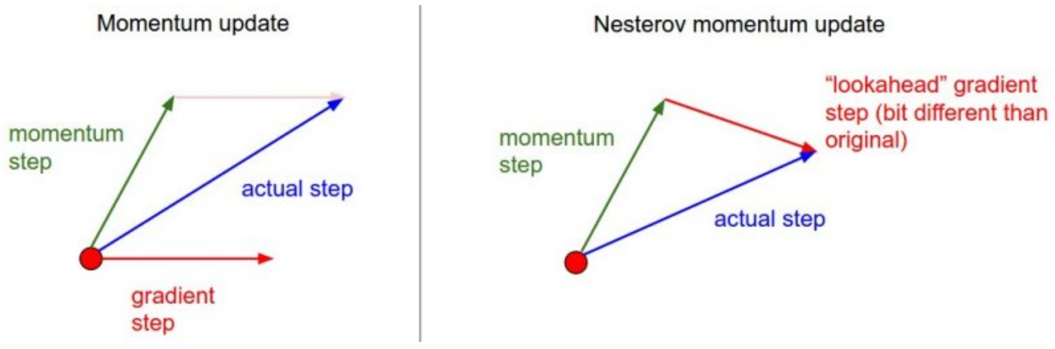


Figure 2-5: Comparison between the momentum-based algorithms [6]

2-4-3 Adaptive Methods

The gradient and momentum-based optimization methods use a fixed learning rate. If the training dataset is very sparse, this can lead to problems. Adaptive optimization methods aim to improve upon this by adapting the learning rate during training. For the purpose of this thesis, two of these adaptive methods are analysed: AdaGrad and AdaDelta.

AdaGrad The AdaGrad [14] algorithm adapts the learning rate based on a weighted sum of squared previous gradients. This essentially means that if a parameter has received a large update, the next update will be smaller. The downside of this algorithm is that the learning rate keeps decreasing as more iterations of the learning process are done, ultimately stopping learning altogether.

AdaDelta The AdaDelta [15] algorithm is based on the same principle as AdaGrad, with the exception that it does not take all the previous gradients into account when adapting the learning rate. It only takes gradients from a fixed window size, which means it will not have the learning stop problem that the AdaGrad algorithm had.

2-5 Regularization

If the training dataset is too small, the network is prone to overfit the data. Because the CNN has many parameters, a small training set can quite easily be fitted perfectly. The problem is that this does not generalize well towards other independent validation data, because it basically memorizes all the input images and it does not learn features that would lead to a generalized representation of the class. There are some regularization methods to deal with this overfitting problem and these are discussed below.

2-5-1 Dropout

The 'dropout' method proposed by Hinton et al. [16] and visualised in Figure 2-6 deals with overfitting by deactivating some of the neurons in the hidden layers, such that the network will not overfit the training data. This deactivating is done at random with a certain probability (often 0.5) for each pass through the network. This means that the network will try to fit itself to the correctly labelled output with only a random part of the neurons activated.

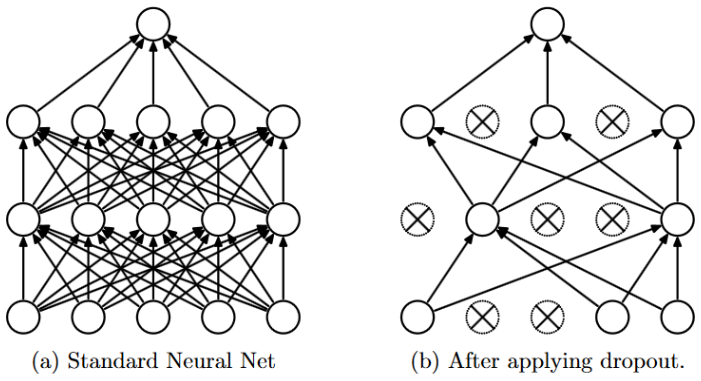


Figure 2-6: Dropout [7]

2-5-2 Artificial Data Expansion

If the training set is too small, this can artificially be expanded by scaling, rotating, blurring or cropping images from the training set. This will lead to a larger dataset to train on, but care needs to be taken, because this artificially created data is often highly correlated to the existing training data from which it was created.

2-5-3 Early Stopping

Another way to prevent the network from overfitting is to do validation during the training. If no progress in performance is detected, the training can be stopped prematurely, because this means that more training will lead to overfitting.

2-5-4 L_2-Normalization

The weights between the layers are a linear combination of neural outputs. This linearity means that a set of weights W that works for classification will not be unique in the sense that a multiplication of these weights with a parameter λ would lead to an equally good classifier. To deal with this, the cost function can be extended with the L_2-norm of all the weights in the network combined, which effectively bounds the weights. This regularization is called L_2-normalization. This method also positively influences the generalization, because no input can have too much effect.

2-6 Region Proposals

Research has shown that the generation of specific region proposals can significantly improve performance of a CNN. R-CNN and its faster implementations [17][18] show superior results over sliding window classifiers in terms of speed, which makes sense, because less visual data needs to be analysed. This thesis operates under the same assumption, but the region proposals are generated with the aid of data from LRF instead of the visual data on its own.

2-7 State-of-the-Art

Now the concept has been introduced, it is time to discuss what a CNN looks like these days. Figure 2-3 already showed the classical LeNet-5 CNN based on an input 28x28 image. Modern networks take in larger images and therefore need a lot of computational effort. An important factor in the development of the current state-of-the-art CNNs has been the ImageNET Largescale Visual Recognition Challenge (ILSVRC) [19]. This challenge, organized annually since 2010, provides challenges in object detection, tracking and classification in images. A dataset with roughly 1,000 labelled images for each of the 1,000 possible classes is given and the contestants need to correctly predict outcomes on another validation dataset without any groundtruth labels. For each image in the validation dataset, the contestants need to provide the top-5 results that have the highest probability of being in the given image according to their method. If the correct output label is within these top-5 results, it will be deemed correctly predicted for the top-5 prediction results.

2-7-1 AlexNet (2012)

The biggest upset within the competition came from Krizhevsky et al. [8] in 2012 and was named AlexNet. They were the first to achieve groundbreaking results by using a CNN as shown in Figure 2-7. The AlexNet architecture was entered into the ILSVRC-2012 competition and achieved a winning top-5 error rate of 15.3%, whereas the runner-up achieved a significantly higher rate of 26.2%.

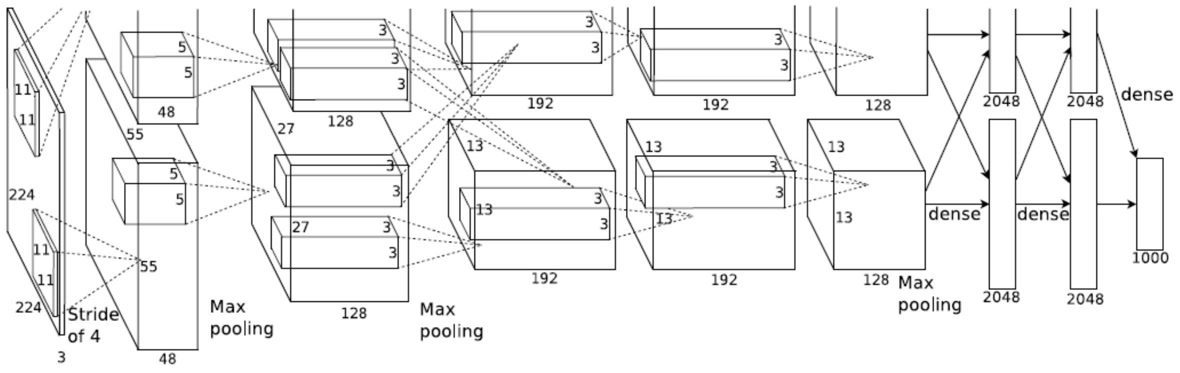


Figure 2-7: The AlexNet architecture [8]

2-7-2 GoogLeNet (2014)

Two years later, Szegedy et al. [9] came up with a CNN that was more accurate than AlexNet and used twelve times fewer parameters. A top-5 error rate of 6.67% was achieved on ILSVRC-2012 data. The authors of the GoogLeNet architecture argued that deeper networks would provide better results, but this resulted in two drawbacks: computational effort and increased sensitivity to overfitting. Their major contribution in dealing with this problem was a repeatable block that they named inception and it is shown in Figure 2-8.

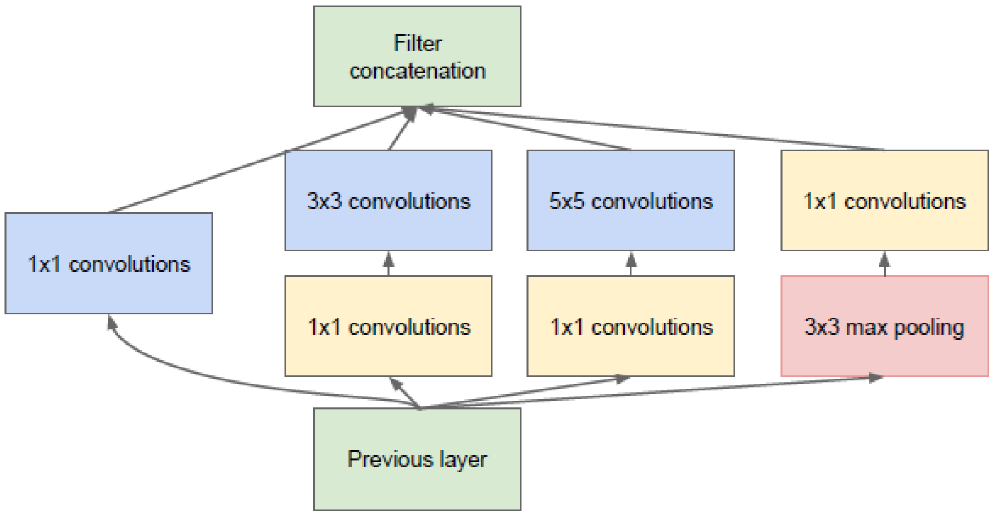
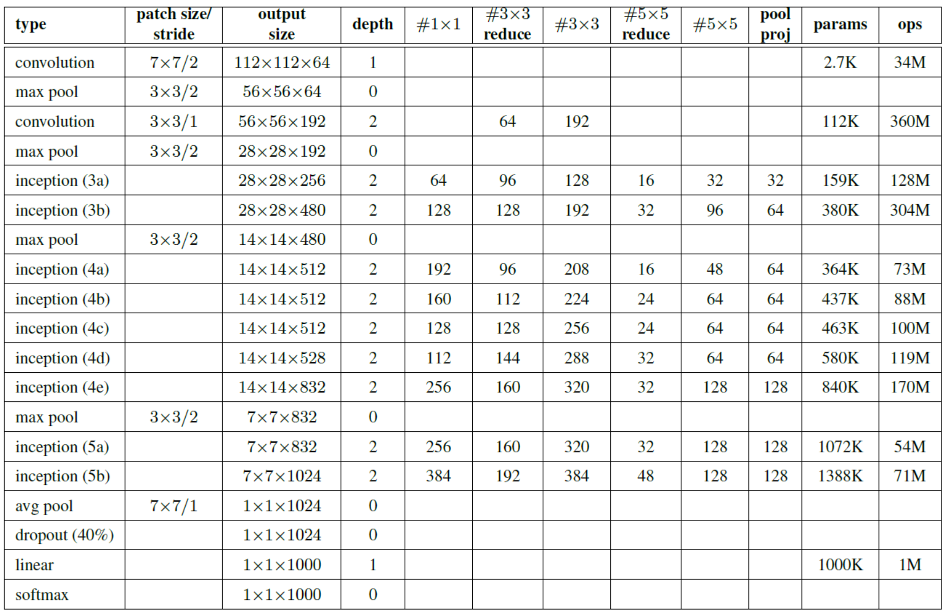


Figure 2-8: The inception module [9]

This block performs three types of convolutions (1x1, 3x3 and 5x5) and concatenates the results together with a 3x3 max pooling operation. GoogLeNet stacks many of these inception blocks together. Its architecture is shown in Figure 2-9, which shows several inception modules as well as pooling. What is very interesting about this architecture is the absence of a fully connected layer at the end.



type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure 2-9: GoogLeNet architecture [9]

2-7-3 ResNet (2015)

In 2015, the ResNet [10] architecture emerged victorious with a 3.57% top-5 error rate, which is arguably even better than human performance. They created the deepest network (151 convolutional layers!) in the contest so far and invented a clever method to prevent overfitting the network. This residual block is shown in Figure 2-10. By passing the identity and adding it to the convoluted results, they prevent the network from learning nonsensical features while delving deeper into the high-level features.

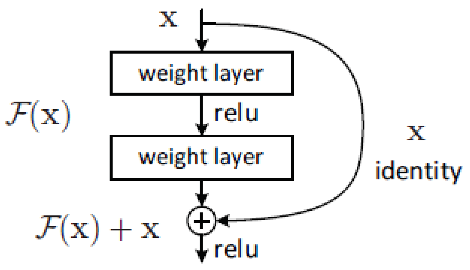


Figure 2-10: The residual block [10]

2-8 Concluding Remarks

The CNN has been discussed as a promising method in image classification and this makes it interesting for recognizing humans in visual data. The major advantage of CNN over the traditional methods is the fact that features and classification are jointly learned. The learned features can include features that are very similar to the handcrafted features in traditional methods. Initial layers learn raw features like edges and further down the CNN pipeline more detailed features are learned. The fact that, so far, the progress in the field of deep learning neural networks has been more empirical than based on theoretical knowledge makes it very interesting for research purposes.

Chapter 3

Experimental Setup

This chapter elaborates on the used test setup that was made available by Robot Security Systems (RSS). Figure 3-1 shows this test robot, henceforth referred to as Robulab. Firstly, the available sensors are discussed in Section 3-1 followed by their calibration in Section 3-2 and the data synchronisation in Section 3-3. The chapter concludes with an overview of the recorded datasets in Section 3-4.

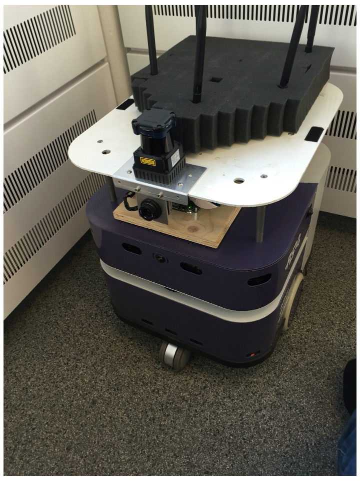


Figure 3-1: Robulab, the test robot at RSS

3-1 Sensors

The experiments for this thesis are done with two available sensors. A Laser Range Finder (LRF) for providing depth information and region proposals and a pinhole camera.

3-1-1 LRF

The laser range measurements are done by a Hokuyo UTM30LX-EW sensor with a 270° Field Of View (FOV) and a 0.25° angular resolution. Because this data is used solely for the purpose of defining region proposals in the visual data, the measurements that do not fall within the FOV of the camera are not taken into account, because this would only lead to unnecessary data. The FOV of the pinhole camera at the front of Robulab is 70°, which led to the decision to only take in laser measurements in the range of -40° to 40°. The laser measurements that can not be seen within the FOV of the camera are filtered out at a later point.

3-1-2 Camera

Robulab is equipped with four Hikvision pinhole cameras of the type DS-2CD6412FWD for a 360° view of the surroundings. For the purpose of human detection in this thesis only the front camera is used. The camera has a 1280x960 resolution and a FOV of 70°.

3-2 Sensor Calibration

To be able to relate 3D world coordinates to 2D image coordinates two steps need to be taken. First the 3D points in the world frame need to be transformed into the camera frame, these are the extrinsic parameters and depend on the position and orientation of the camera with respect to the world. Then the coordinates in the camera frame need to be transformed to coordinates in the image space, these are the intrinsic parameters and depend on the focal distance and lens distortion of the camera. The flowchart in Figure 3-2 visualizes this camera calibration.

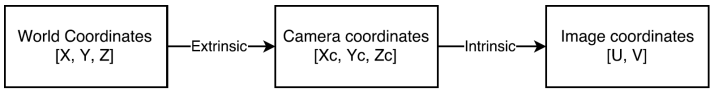


Figure 3-2: Flowchart of the camera calibration

3-2-1 Extrinsic Parameters

The extrinsic parameters relate the world coordinates to the camera coordinates, i.e. the camera centre. This can be done by a rotation and translation, where the translation gives the location of the world coordinate origin within the camera coordinate frame. The rotation describes a possible rotation of the world coordinate frame to the orientation of the camera coordinate frame.

$$H = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \tag{3-1}$$

3-2-2 Intrinsic Parameters

The intrinsic parameters relate the camera coordinate frame to the image coordinates or pixel locations. This is done by pre-multiplying the camera coordinates with the intrinsic matrix K [20]:

$$K = \begin{pmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3-2}$$

Where f represent the focal lengths of the camera and x_0 and y_0 represent the principal point offset, i.e. the offset, measured in pixels, from the intersection with the line perpendicular to the image through the camera pinhole to the origin of the image. The symbol s is the axis skew, which is mostly zero, but can sometimes be non-zero due to skewness caused by digitization processes.

Calibration Methods Camera calibration methods aim to estimate the extrinsic and intrinsic parameters that map the real world to the image coordinates as discussed in the previous paragraphs. The classical method to do this was proposed by Zhang [21] in 1998. For this calibration a planar pattern, such as a checkerboard, needs to be observed from at least two different positions and orientations. The method proposed by Zhang can then, by observing the same corners on the checkerboard in different positions, estimate the intrinsic and extrinsic parameters of the camera. For the purpose of estimating the intrinsic parameters of the camera, a MATLAB toolbox is available [22], which was used for the calibration of the front pinhole camera. After this calibration, the 3D world coordinates of the checkerboard could be estimated, accurate within 1 [cm]. The resulting intrinsic matrix was computed to be:

$$K = \begin{pmatrix} 946.1115 & 0 & 681.2535 \\ 0 & 706.8499 & 297.5400 \\ 0 & 0 & 1 \end{pmatrix} \tag{3-3}$$

3-2-3 LRF Calibration

The extrinsic parameters then relate the world coordinates to camera coordinates and these are different for every position of the checkerboard pattern. For the calibration between the LRF and the camera the specific extrinsic parameters that relate the laser coordinates to the camera coordinates need to be known, which is needed if detected objects in the laser range data will be used to generate a region proposals in the visual data. For this purpose it is necessary to find one or more points that can be seen by both the LRF and the camera, which will make it possible to relate the data from the laser scan to the camera position and therefore the image coordinates. For this purpose the checkerboard pattern is again used. The calibration setup can be seen in Figure 3-3.

Using the camera calibration toolbox, the coordinates of this point on the checkerboard, expressed in the camera coordinate system, can be estimated. The same can be done for the laser scan data, which will give a distance measurement. This data can then be used

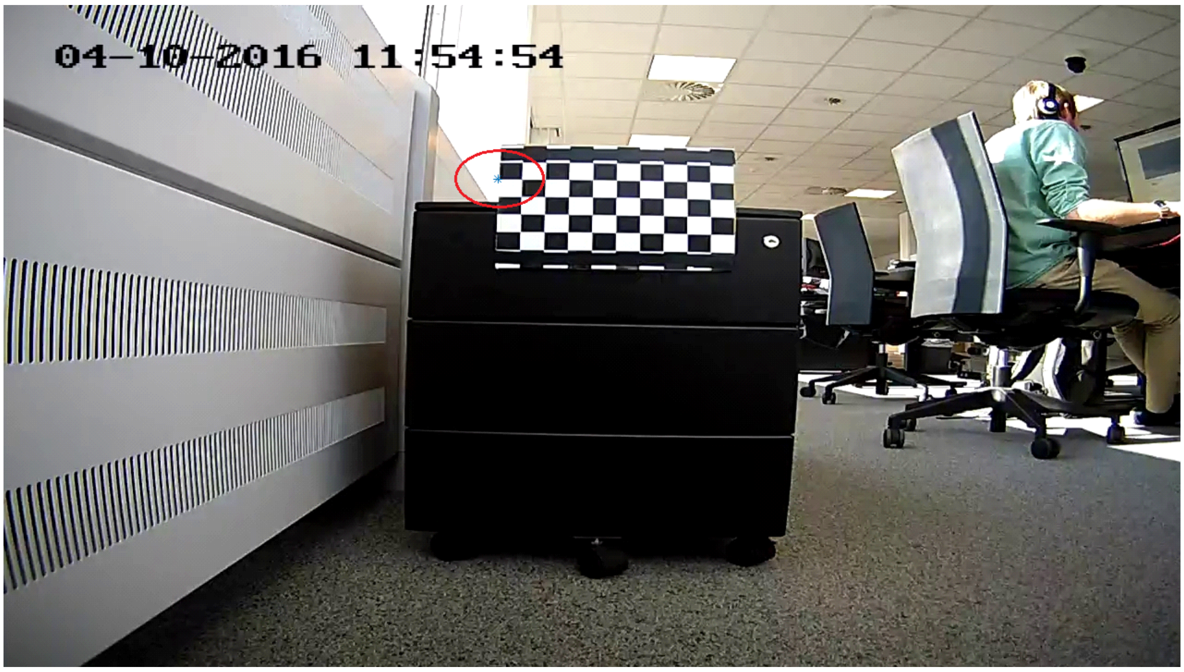


Figure 3-3: The calibration setup with one laser scan point projected on the checkerboard.

to compute the transformation matrix that transforms the laser coordinates into camera coordinates. This can be done by a rotation (different coordinate frame) and a translation (sensors located at different points in space). With this transformation known, the rest of the laser points can be projected as well. The first result of projection is shown in Figure 3-4.

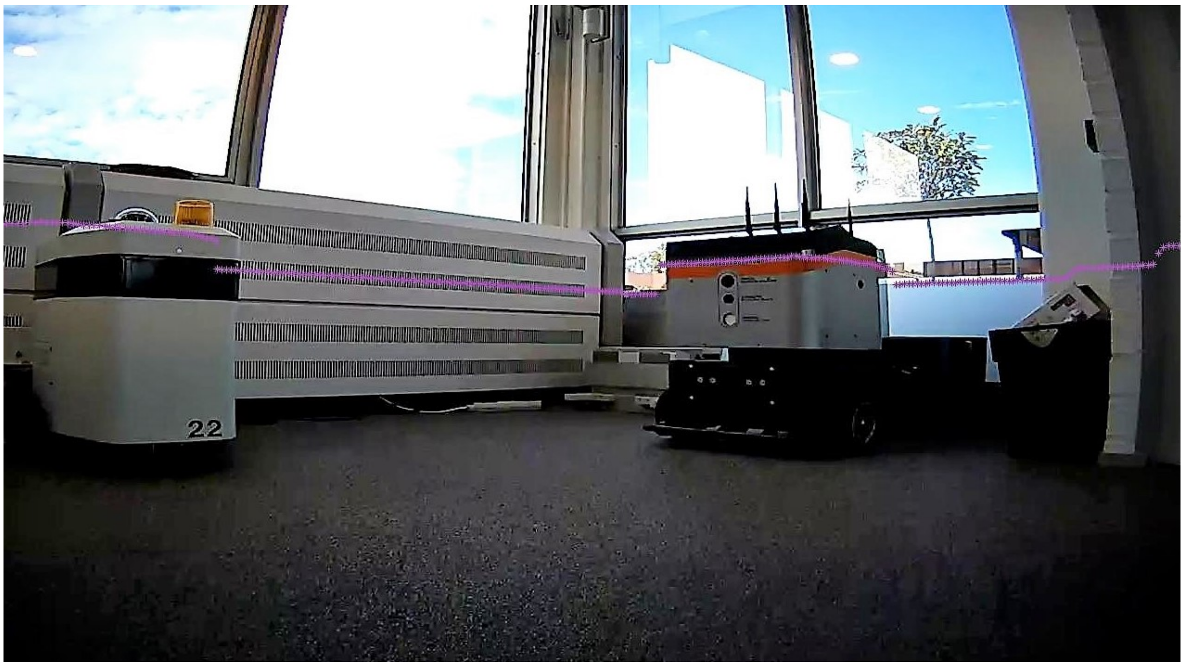


Figure 3-4: First laser projection with errors.

From this figure it is clear that this projection can not be used for the creation of region proposals, because the projected points are still off. There are two reasons for this, the first of which is that the image on which the points are projected is still distorted. The intrinsic parameters that were estimated during the calibration can be used to undistort the image and this will already lead to a more accurate projection. The second problem is that the data measurements from the laser range finder and the camera are not yet synchronised. The next section will delve further into this matter.

3-3 Synchronisation

If two separate datastreams are to be used to capture data at the same time, synchronisation of the datastreams is key. Sensors usually do not operate at the same frequencies and one can imagine that writing data from the camera takes a little bit longer than writing distance measurements from the LRF. This can be approached in either two ways: let both sensors acquire data at their own rate and then match the correct pairs or tune the rate at which they acquire data, such that both sensors take measurements at the same time instances. In the datatool that was used to read and write data from the sensors on Robulab, the first method was chosen, because it is simply easier to implement. After the synchronisation, the laser scan is once more projected and the results can be seen in Figure 3-5.

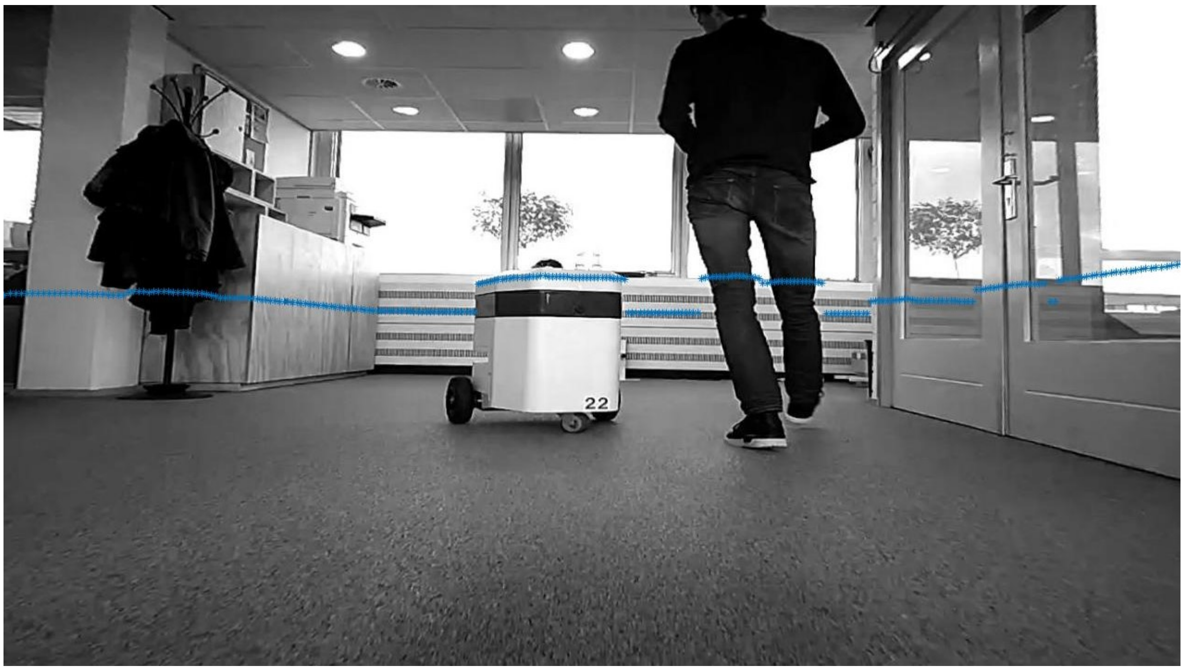


Figure 3-5: A projected laserscan after data synchronisation and undistortion.

The improvements from Figure 3-4 to Figure 3-5 are clearly visible and the projected laserscan is now accurate enough to be used for the generation of region proposals in the visual data. The transformation matrix that relates the laser coordinates to the camera coordinates was computed to be:

$$H = \begin{pmatrix} -0.0721 & -0.9973 & 0.0065 & -65.6234 \\ 0.0861 & -0.0128 & -0.9962 & -138.7402 \\ 0.9937 & -0.0713 & 0.0868 & 8.4264 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3-4}$$

Where the rotation matrix has been corrected for a slight tilt of the camera.

3-4 Datasets

With the calibration and synchronisation running, all that remained before starting the experiments was acquiring data. For the experiments four datasets are used:

- RSS - Dataset recorded at RSS in the Hague.
- 3ME - Dataset recorded at the Mechanical, Maritime and Materials Engineering (3ME) faculty.
- IOD - Dataset recorded downstairs at the Industrial Design (ID) faculty.
- IOU - Dataset recorded in the hallway between 3ME and ID and upstairs at the ID faculty.

Table 3-1 shows an overview of the contents of the datasets. This table elaborates on only the generated datasets for the visual experiments. Sometimes multiple crops are taken from one image. The datasets are separated into test sets (only used for testing the accuracy of the learned network) and training sets (used for training the network). The table shows the total number of images in that specific dataset with additional information on the ratio between true (i.e. human) and false (i.e. background) crops. The RSS dataset is the largest and serves to train a Convolutional Neural Network (CNN) from scratch for human detection. The other three datasets are smaller and serve to show the feasibility of finetuning a pre-learned network on a specific location. The visual data experiments are discussed in detail in Chapter 5.

Table 3-1: Overview datasets

Dataset	Test (True/False)	Train (True/False)
RSS	696 (169/527)	2782 (676/2106)
3ME	200 (90/110)	813 (304/509)
IOD	150 (70/80)	641 (269/372)
IOU	180 (60/120)	739 (265/474)

Chapter 4

LRF Experiments

The experiments are split up in two parts: Laser Range Finder (LRF) experiments and visual data experiments. The LRF experiments discussed in this chapter aim to implement the various methods of detection, that were explained in Chapter 1 and analyse the results. First and foremost, the goal of incorporating depth information from a LRF is to be able to look for interesting segments within the visual data more easily. The simplest way to this is to use the Point-Distance Based Segmentation (PDBS) that was discussed in Chapter 1, which clusters the data points from the laser range measurements based on a simple distance criterion. Experiments with PDBS are discussed in Section 4-1. Chapter 1 also discussed Kalman-Filter Based Segmentation (KFBS), of which experiments are discussed in Section 4-2. Section 4-3 elaborates on the region proposals that are generated with the aid of these clustering methods. On top of that experiments are done with a neural network in an attempt to train the network to recognize leg patterns, these are discussed in Section 4-4.

4-1 PDBS Clustering Experiments

PDBS clusters data from the LRF into segments based on a simple distance threshold criterion. These segments can then easily be used to generate crops with the aid of the sensor calibration that was discussed in Chapter 3. Experiments were done with different distance thresholds ranging from 10-100 [mm]. Some pre-filtering of the segments has also been done by excluding segments shorter than 5 points, because that would never represent anything useful and just lead to an unnecessary amount of region proposals. Segments of more than 50 points are also excluded from the analysis, for really long segments generally only represent long objects such as walls and closets. This pre-filtering makes sure that not all segments are proposed, because that would mean a lot of crops that need analysis. The drawback here is that humans that stand behind a counter are not visible for the LRF, but these segments are also not proposed for further analysis using a vision-based classifier. The results from the experiments are visualized in the figures below.

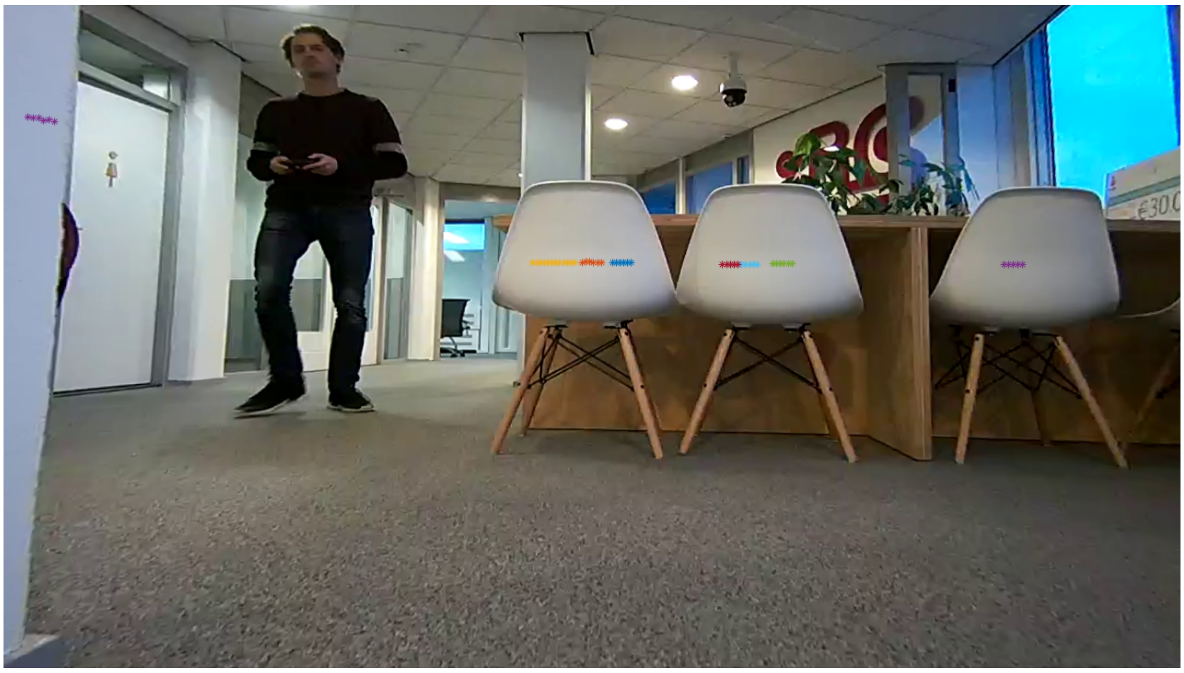


Figure 4-1: Results of PDBS with distance threshold 10 [mm]

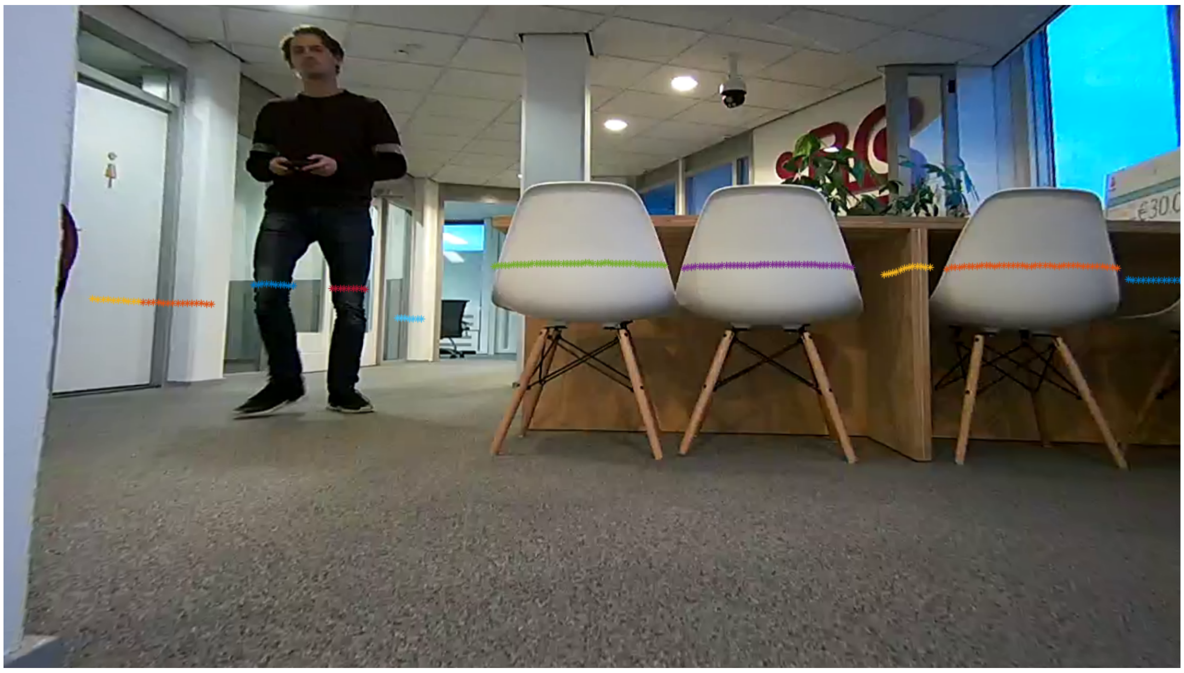


Figure 4-2: Results of PDBS with distance threshold 50 [mm]

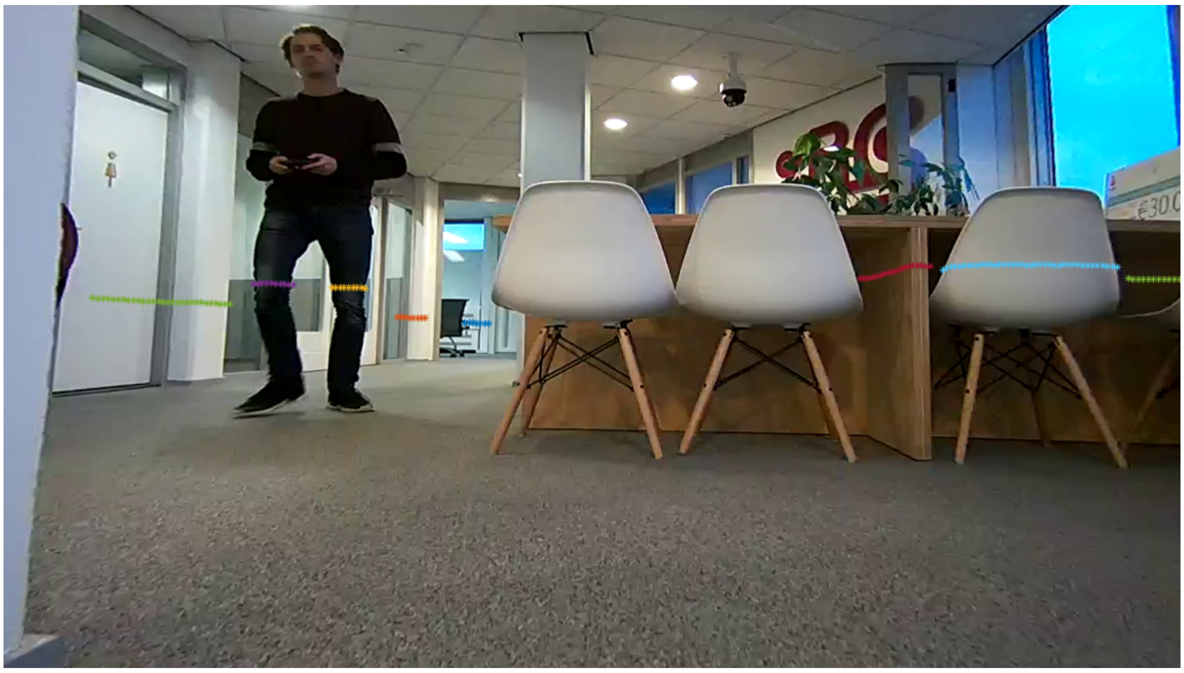


Figure 4-3: Results of PDBS with distance threshold 100 [mm]

From these figures it becomes clear that a distance threshold of 10 [mm] leads to a very small amount of segments due to all the short segments being filtered. More importantly, the measurements on the legs of the human in the image are not being proposed as Region of Interest (ROI). On the other hand a distance threshold of 100 [mm] does include the measurements on the legs, but ultimately leads to problems when people stand close to walls. This leads to the legs 'blending in' the background and this means that the leg segments are sometimes not proposed. The optimal distance threshold was empirically found to be 50 [mm], because this means leg measurements are reliably proposed without blending in with the background.

4-2 KFBS Clustering Experiments

In Chapter 1, not only PDBS was discussed, but also KFBS. PDBS clusters the laser data based on a simple distance criterion, which naively assumes that measurements are perfect and there is no noise. KFBS aims to see whether points belong to the same segment in a stochastic sense, accounting for noise. The algorithm as proposed by Borges [11] and discussed in Chapter 1 is used as guideline for these experiments, with some alterations. Borges worked with a LRF with an angular resolution of 0.6°. The LRF used for the experiments in this thesis works with an angular resolution of 0.25°. This has an effect on the λ used in the algorithm. Instead of the 10° used in the Borges experiments, this thesis operates with a λ of 4°. The standard deviation σ of the LRF is 0.03[m], which is taken from the sensor documentation. The covariance matrices and state are initialized as proposed by Borges:

$$x_0 = \begin{pmatrix} r_n \\ 0 \end{pmatrix} \tag{4-1}$$

$$P_0 = \frac{1}{9}\begin{pmatrix} (cot(\lambda)r_n\Delta\phi)^2 & 0 \\ 0 & (cot(\lambda)r_n) \end{pmatrix} \tag{4-2}$$

$$Q_n = \begin{pmatrix} 0.01 & 0 \\ 0 & 0.01 \end{pmatrix} \tag{4-3}$$

$$R_n = \sigma^2 = 0.009 \tag{4-4}$$

The results of KFBS are shown in the figures below. From these figures it becomes clear that KFBS manages to filter out nearly everything but the leg segments, which means the region proposals based on these segments will be limited and therefore make a detection algorithm faster. On the other hand, Figure 4-6 shows that, when the target is obscured, the human is completely missed. PDBS seems to have more reliable proposals in this sense, but at the cost of having more. The results of KFBS and their naive counterpart PDBS and their possible uses are further discussed in Chapter 6.

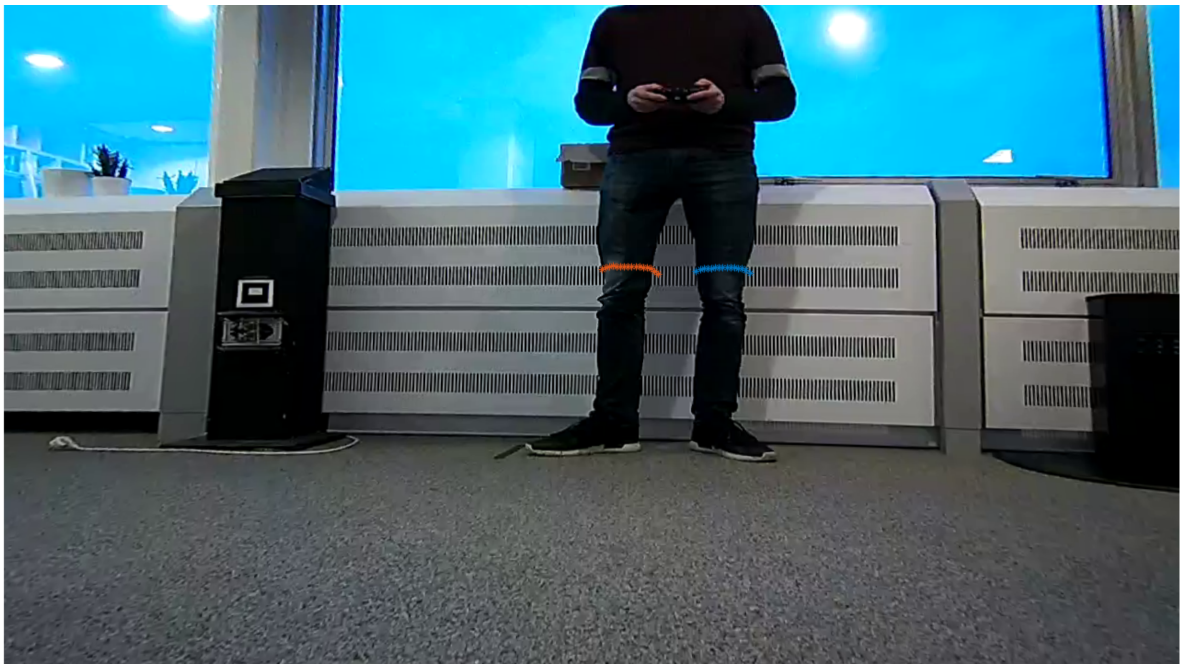


Figure 4-4: Results of KFBS when close to wall

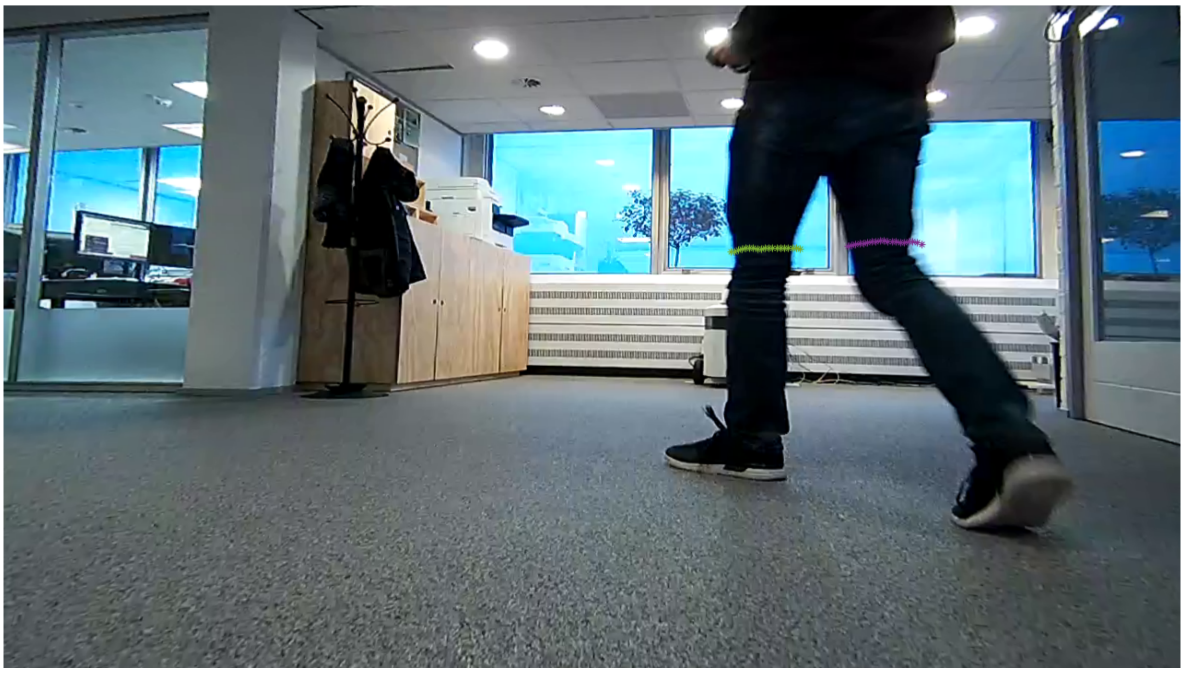


Figure 4-5: Results of KFBS when walking

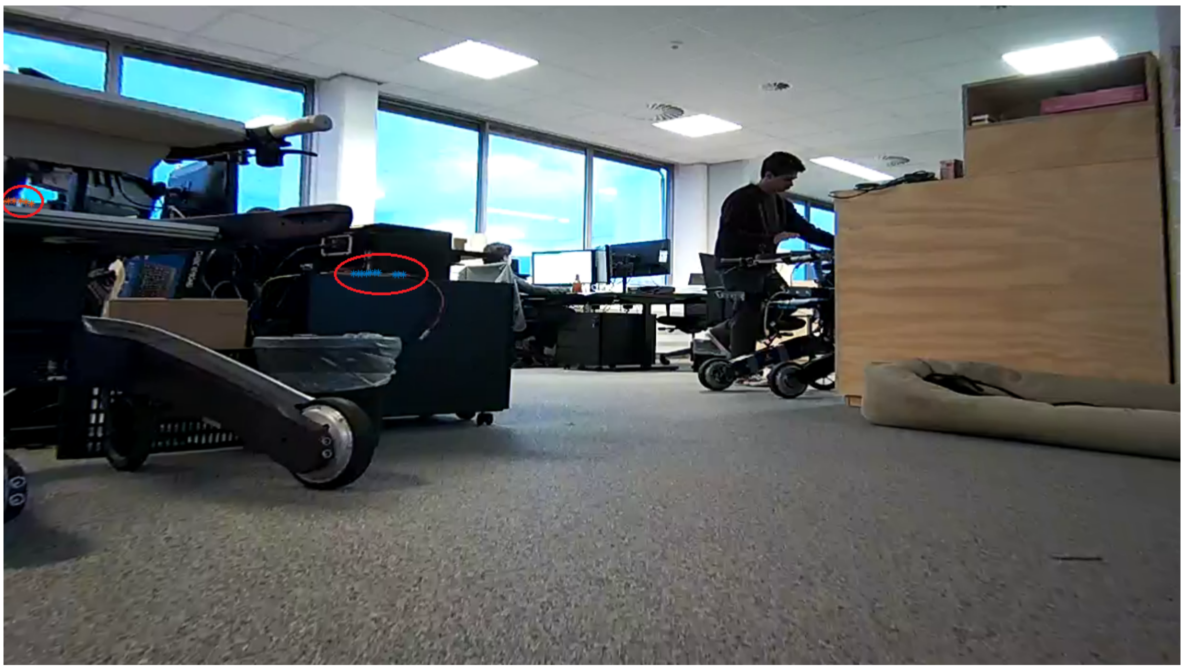


Figure 4-6: Results of KFBS when obscured.

4-3 Region Proposals

The clustering experiments show a reliable segmentation of leg patterns with minimal redundant proposals, which makes it valid as region proposal generator for a vision-based classifier. The proposals are generated by estimating the size of a human, should the detected segment be a human and creating a crop around it based on that size. For this purpose the bounding box of a human is taken to be 2x1 [m]. Figure 4-7 shows some of the region proposals generated by the segmentation methods discussed before.

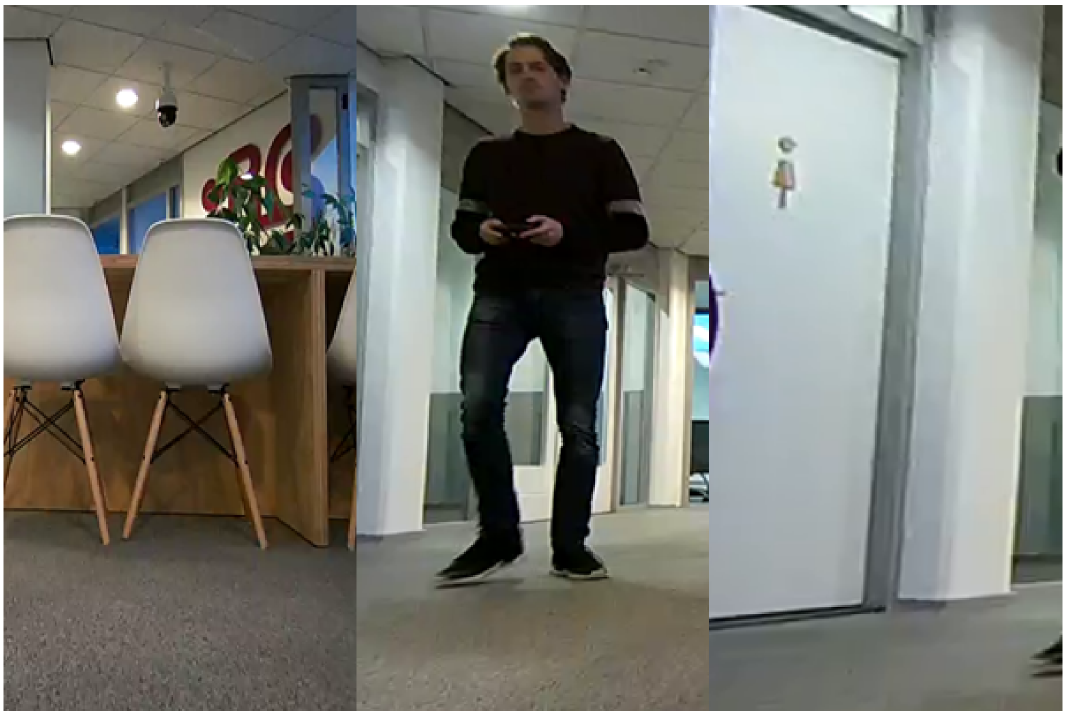


Figure 4-7: Region proposals generated by laser segmentation

4-4 Deep Learning LRF Experiments

Experiments with clustering methods such as PDBS and KFBS showed that they are viable as region proposal generators for a vision-based classifier. The experiments with a Convolutional Neural Network (CNN) based on these region proposals are discussed in Chapter 5. Because the core of this research focusses on using CNN for pattern recognition, experiments are done with using this concept on the LRF measurements. The network design is shown in Figure 4-8. In order to make sense from the data, the raw distance measurement vector needs to be transformed to a vector containing the differences between subsequent distance measurements in the vector, essentially creating a gradient vector. The distance measurements from the LRF are recorded as 321x1 vectors, which means the gradient vector is 320x1. This vector is then pushed through 4 convolutional layers, transforming the 320x1x1 vector to a 300x1x1 vector that will have class scores. The classes all represent a certain area within the

LRF measurements and this will be compared to a groundtruth label vector that contains ones in the areas that contain humans and zeroes for regions that do not. These labels were created while labelling visual data, where positive crops were used to see which of the LRF measurements would receive a positive label.

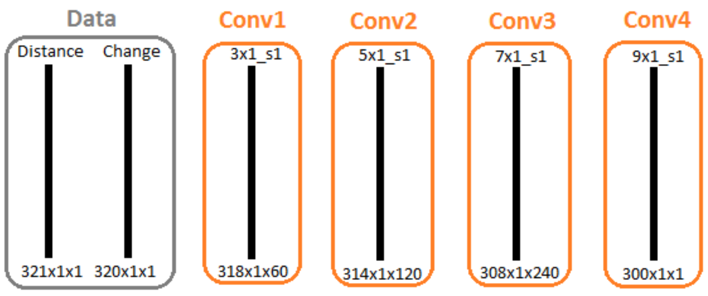


Figure 4-8: LaserNet design

Figure 4-9 shows the learning curves of the neural network experiments with the distance measurements from the LRF. The network was trained with a sigmoid cross-entropy loss function and the learning curves show some convergence with the loss stabilizing at a value between 100 and 200.

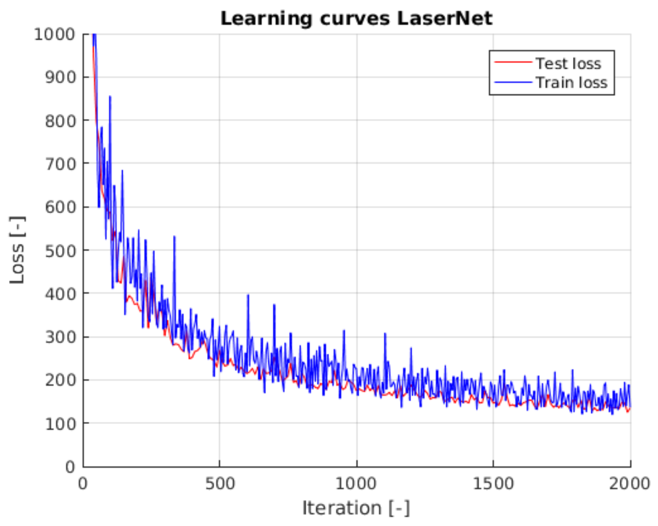


Figure 4-9: Learning curves of LaserNet on data from the 3ME and IOD datasets, tested on IOU.

However, the resulting network after 2,000 iterations turns out to not be able to make reliable predictions. Although parts of the labels are predicted correctly, it is not sufficient to extract regions of interest from these predictions. For that purpose the segmentation methods discussed previously are not only much simpler to implement, but also more reliable. Conclusions from all LRF experiments are further discussed in Chapter 6.

Chapter 5

Visual Data Experiments

The experiments with the Laser Range Finder (LRF) data aimed to reduce the computational load on the human detection algorithm. This section will elaborate on the deep learning experiments that aim to classify the region proposals. For the experiments several network designs were made. There are many parameters that can be tuned for the neural networks, which makes it somewhat empirical based to find out which of these parameters are optimal. The goal of these network designs is to change just specific aspects to see whether they influence the learning process in a positive way. The first network is named KathNet1 and KathNet2 and 3 are network designs that evolved from KathNet1 with specifically changed aspects. Data was recorded in the RGB colorspace, but for each network experiments are done with HSV and grayscale data as well. Section 5-1 elaborates on these experiments and how they are set up. Section 5-2 discusses the experiments with the first Convolutional Neural Network (CNN) design, followed by experiments with KathNet2 in Section 5-3 and experiments with KathNet3 in Section 5-4. Section 5-5 elaborates on the two different solver types AdaGrad and AdaDelta. Experiments with finetuning a pre-learned CNN on data from a specific location are discussed in Section 5-6. This chapter concludes in Section 5-7 with a comparison between the proposed multi-sensor architecture in this thesis and a pre-defined HOG-SVM sliding window human detection algorithm.

5-1 Experiment Details

All initial experiments (discussed in Sections 5-2 through 5-4) are done on the Robot Security Systems (RSS) dataset. Details on the different datasets can be found in Chapter 3. The data in the datasets is recorded in the RGB colorspace, but these datasets are also converted to grayscale and HSV colorspaces for the initial deep learning experiments. All experiments run 1,000 iterations with batches of 50 crops and the accuracy on the test set is measured after every 20 iterations to keep track of the quality of the learning process and all results are saved in log files. A snapshot of the current network is saved after each 20 iterations, such that a choice can be made, based on the logs, which of the networks during the entire experiment performed best. The experiments run with a base learning rate of 0.01 and use L_2-normalization as regularization method. To artificially expand the datasets, the crops are mirrored. Experiments with dropout as regularization method did not lead to mentionable performance improvements.

5-2 KathNet1 Experiments

The first network consists of three convolutional layers combined with max-pooling layers to downsample the data. All pooling layers discussed in this chapter use max-pooling. The design of this network is inspired by the AlexNet [8] architecture discussed in Chapter 2. Eventually the input image of 200x400 pixels is reduced to a size of 4x9 before applying the last convolutional filter to create the feature vector. This feature vector is then connected to one more fully connected layer before reducing the size to a 2x1 score vector that is used to predict the probabilities for both possible classes (human or not). A softmax classification with cross-entropy loss function is used for the learning process. The design is visualized in Figure 5-1.

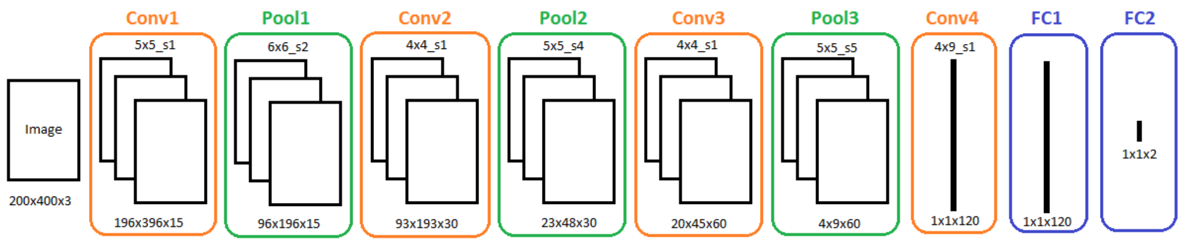


Figure 5-1: The schematic visualization of the KathNet1 design

5-2-1 RGB

The learning curves of the RGB experiments done with the KathNet1 network are shown in Figure 5-2. The blue curve represents the training loss, which is measured every 5 iterations during the learning process. Barring the fluctuations due to the small batch size, it does seem to slowly converge to zero. After 1,000 iterations the training loss has been reduced to less than 0.1. The loss on the test set, which is measured every 20 iterations, also converges to a value close to 0.2. In terms of accuracy of the predictions, the network performs well with an accuracy of close to 90%.

5-2-2 HSV

The KathNet1 learning curves for the HSV experiments are shown in Figure 5-3. The learning process seems similar to the RGB experiments, however the overfitting of the test data occurs sooner. The loss on the test set also never drops below 0.35, which indicates inferior performance to the RGB network. The accuracy never rises above 90%, which supports the statement that the RGB network performs better.

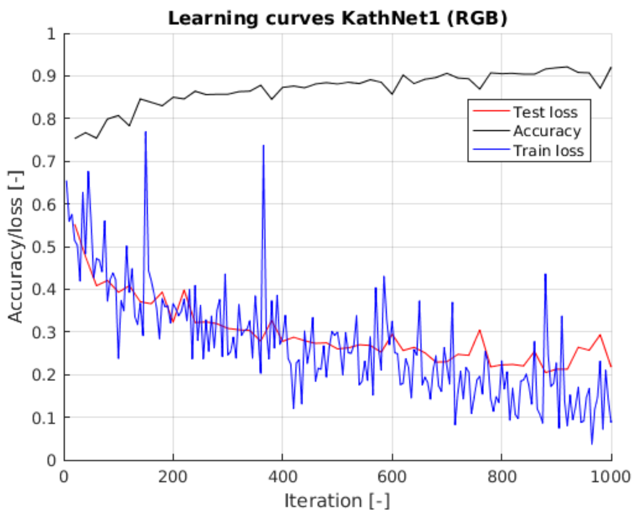


Figure 5-2: Learning curves of the RGB experiments with the KathNet1 network

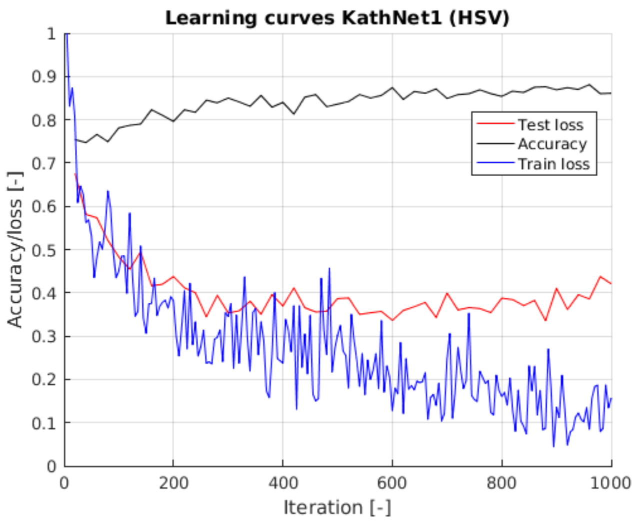


Figure 5-3: Learning curves of the HSV experiments with the KathNet1 network

5-2-3 Grayscale

The learning curves for the grayscale experiments with the KathNet1 network are shown in Figure 5-4. In terms of performance it achieves similar results compared to the RGB experiments with test losses close to 0.2 and an accuracy of more than 90%. Furthermore the training loss converges faster than both the RGB and the HSV experiments, which makes sense, because there is less data to learn from with one channel instead of the three for HSV and RGB colorspaces.

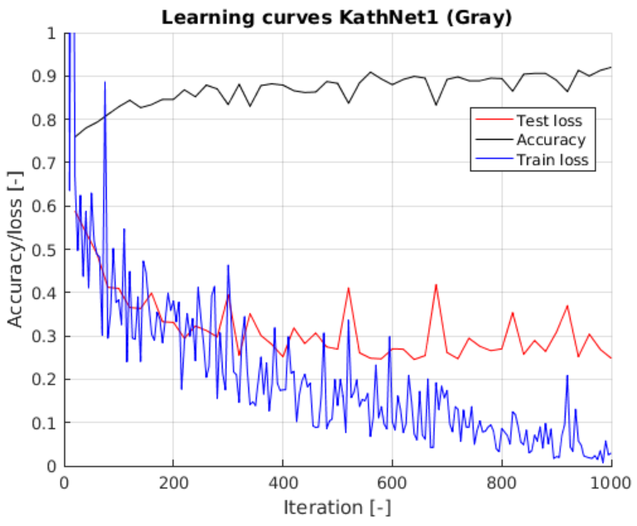


Figure 5-4: Learning curves of the grayscale experiments with the KathNet1 network

5-3 KathNet2 Experiments

The second network also reduces the input image to a size of 4x9, but instead of using pooling layers in between the convolutional layers to downsample the data, a bigger stride is used on the convolutional layers to do so. This design serves to show if the conventional convolution-pooling combinations work better than just convolutions or the other way around. The design is visualized in Figure 5-5.

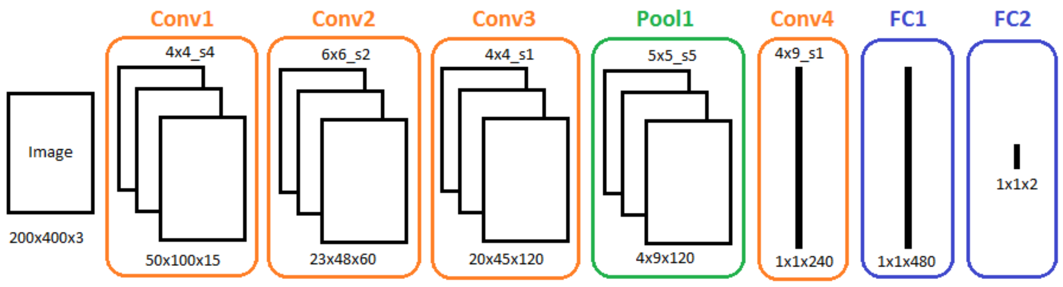


Figure 5-5: The schematic visualization of the KathNet2 design

5-3-1 RGB

The KathNet2 RGB learning curves are shown in Figure 5-6. Beside the two dips at 200 and 980 iterations, the performance is similar to the KathNet1 network with the same data. Accuracy and loss are at approximately the same level and from this number of iterations it can not be concluded yet if the network has started overfitting.

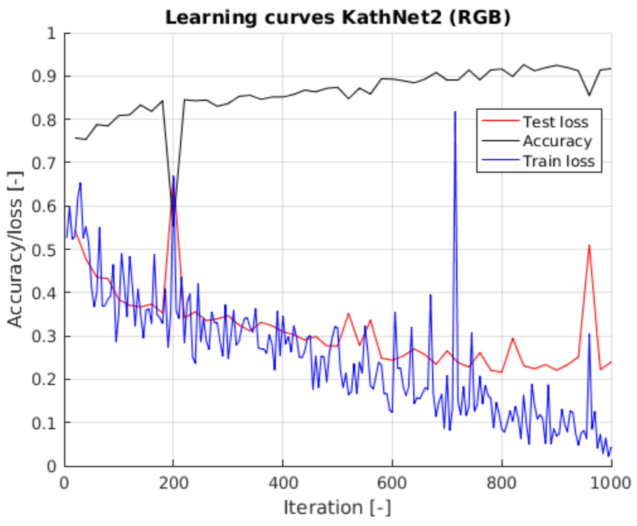


Figure 5-6: Learning curves of the RGB experiments with the KathNet2 network

5-3-2 HSV

The learning curves of the HSV experiments with the KathNet2 network are shown in Figure 5-7. Where the RGB experiments showed different results with this network, the HSV experiments perform worse than their KathNet1 counterpart. The network starts overfitting rather quickly after 500 iterations, having the test loss rising from a minimum 0.35 to 0.5. Accuracy of the HSV experiments also does not reach the 90% mark.

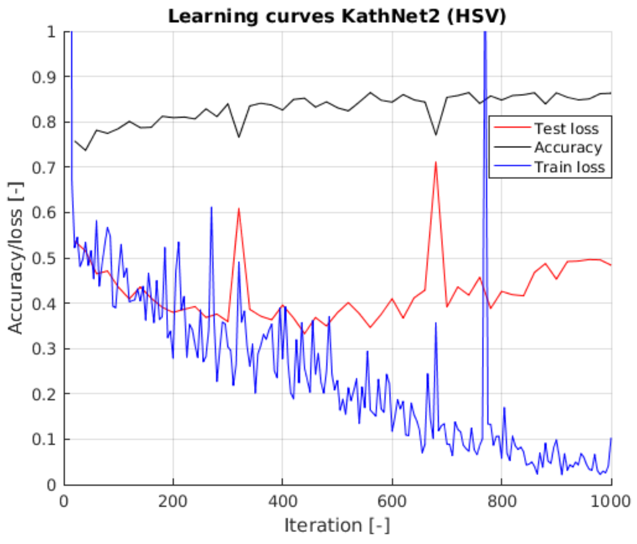


Figure 5-7: Learning curves of the HSV experiments with the KathNet2 network

5-3-3 Grayscale

The KathNet2 grayscale learning curves are shown in Figure 5-8. This shows the same behaviour as the HSV experiments in the sense that it starts overfitting after 500 iterations and test loss never drops below 0.3. Also for these grayscale experiments the accuracy never gets to the 90% mark as it did for the KathNet1 experiments.

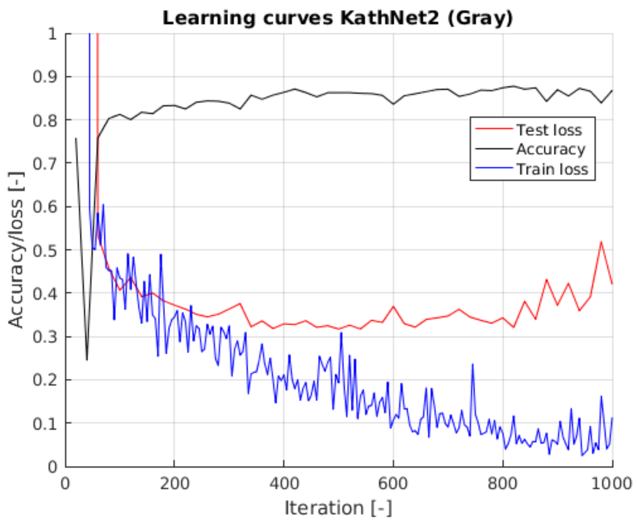


Figure 5-8: Learning curves of the grayscale experiments with the KathNet2 network

5-4 KathNet3 Experiments

The third network goes back to the conventional convolution-pooling combinations, but instead of reducing the image to a size of 4x9, it is now reduced to a size of 3x7. This network aims to show the difference between combining features from a 4x9 grid and a 3x7 grid. A 3x7 grid is chosen, because in the learned crops, faces from humans are usually centred. This could mean that a 3x7 grid can more easily distinguish face features. The network is shown in Figure 5-9.

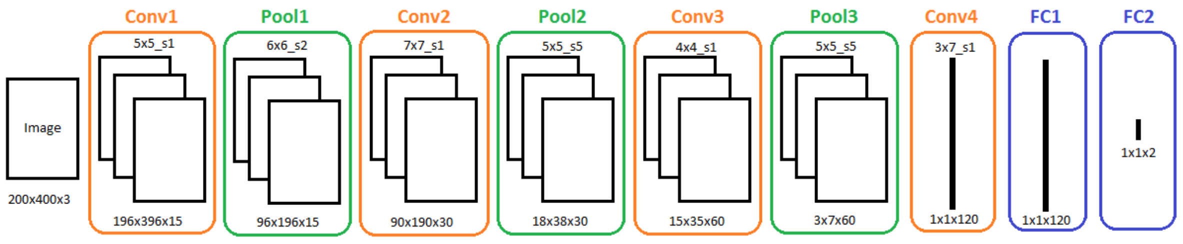


Figure 5-9: The schematic visualization of the KathNet3 design

5-4-1 RGB

Figure 5-10 shows the learning curves for the RGB experiments with KathNet3. Both the training error and test error converge to zero rather smoothly and towards the end no over-fitting can be seen yet, although the test loss seems to flatten at a little less than 0.2. The accuracy of the network approaches 95% and this is the best performance of all experiments considered so far. This is also the network that is used for the finetuning experiments discussed in Section 5-6.

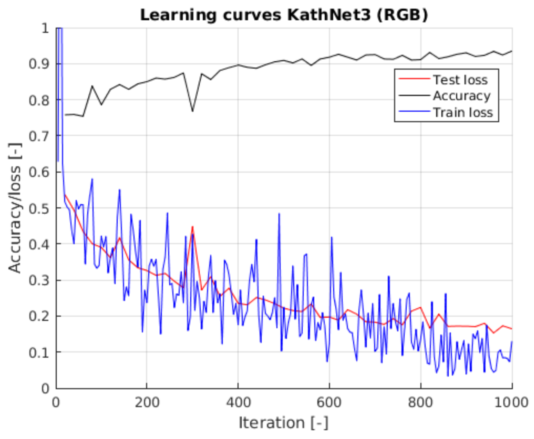


Figure 5-10: Learning curves of the RGB experiments with the KathNet3 network

5-4-2 HSV

The learning curves for the HSV experiments with KathNet3 are shown in Figure 5-11. Although both training loss and test loss converge, the HSV experiments once again show inferior performance compared to the RGB experiments. The accuracy never reaches 85% and the loss never drops below 0.3.

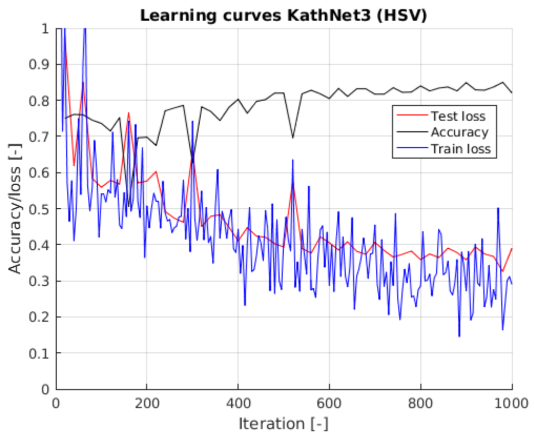


Figure 5-11: Learning curves of the HSV experiments with the KathNet3 network

5-4-3 Grayscale

Figure 5-12 shows the results of the grayscale experiments with the KathNet3 network. Where the grayscale experiments seemed to perform similar to the RGB experiments for the KathNet1 experiments, KathNet3 clearly has a winner with the RGB network. The grayscale network has an accuracy of at best 90% and the test loss never drops below 0.25.

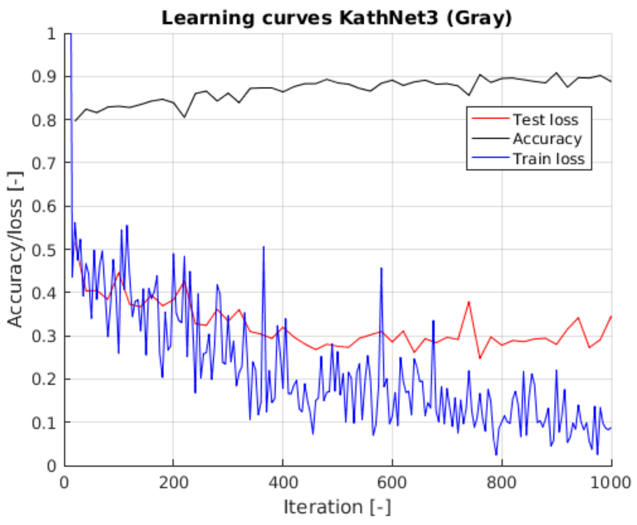


Figure 5-12: Learning curves of the grayscale experiments with the KathNet3 network

5-5 AdaGrad vs. AdaDelta

All above experiments have been done with the AdaGrad [14] solver algorithm. The choice to do so has been based on an initial experiment with both solver types, that showed the AdaDelta [15] solver to perform significantly worse than the AdaGrad solver. Figure 5-13 shows the results of learning experiments with RGB data on the KathNet1 network with the AdaDelta solver. These results are curious, because AdaDelta is supposed to be an improved version of AdaGrad as was discussed in Chapter 2. AdaDelta improves upon AdaGrad by not monotonically decreasing the learning rate based on previous iterations, but by only decreasing it by a subset of those previous iterations. This should help by never having a learning rate converge to zero, essentially being able to keep learning indefinitely. An explanation for the better performance of AdaGrad can be that these improvements are more focussed on multi-classification problems and are too extensive for the binary classification problem of human detection. The real reason behind this is not known, but based on the conclusions from this experiment, the AdaGrad solver was chosen for all the network experiments discussed and for the finetuning experiments to follow.

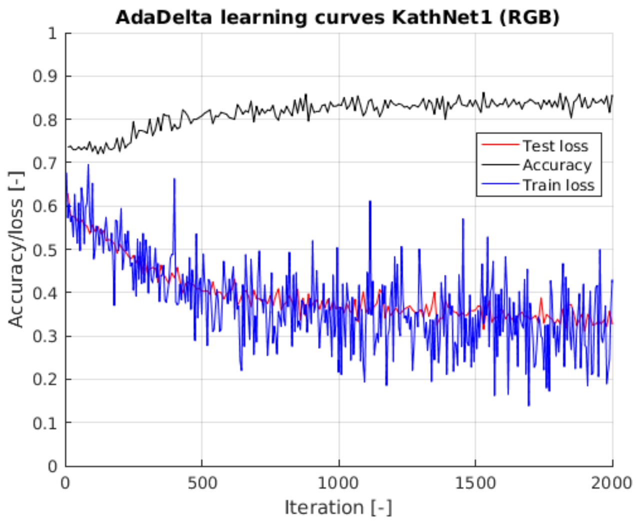


Figure 5-13: Learning curves of the RGB experiments with the KathNet1 network and AdaDelta solver

5-6 Finetuning Experiments

The previous subsection focussed on finding an optimal network. This subsection will aim to provide an insight in the performance of that network with data it has never seen before. For this purpose, the 3ME, IOD and IOU datasets are used as discussed in Chapter 3. From all three datasets the initial 20% is used as test set to have an independent set of data to compare the results. For the purpose of these experiments the KathNet3RGB after 1,000 iterations is used, because this was the network that showed the least errors in its predictions during the initial deep learning experiments discussed in Section 5-4. Initially the performance is measured on all three datasets without changing anything about the parameters to have a benchmark for the effectiveness of the finetuning experiments. This is done by verifying the percentage of false positives (FP), false negatives (FN) and the overall error (OE). As opposed to the initial deep learning experiments on the RSS dataset, the finetuning experiments only run for 500 iterations. All other settings are kept constant.

5-6-1 KathNet3RGB Benchmark Performance

To have a benchmark performance to evaluate the strength of the finetuning experiments, the KathNet3RGB model after 1,000 iterations is used. Table 5-1 shows this benchmark performance. It can be seen that the network has a hard time classifying data it has never seen before, but still manages a reasonable 12.45% overall error rate. The purpose of the finetuning experiments is to show that a little extra data can make this performance better.

Table 5-1: Benchmark performance KathNet3RGB_iter_1000

FN	FP	OE
11.36%	13.22%	12.45%

5-6-2 Finetuning on the 3ME Dataset

The KathNet3RGB_iter_1000 model was used to define a benchmark. The learned network is then finetuned to the 3ME training dataset. The learning process can be seen in Figure 5-14. As can be concluded from this plot, the minimum loss on the test set was established after 120 iterations. After that the network starts overfitting, which led to the decision to use the network as it was after 120 iterations for further finetuning. The performance of the finetuned network is shown in Table 5-2. In comparison to the benchmark performance from Table 5-1 an increase in performance can be seen. False negatives have decreased by 8.63%, false positives by 2.25%. Overall, the errors have decreased by 4.90%.

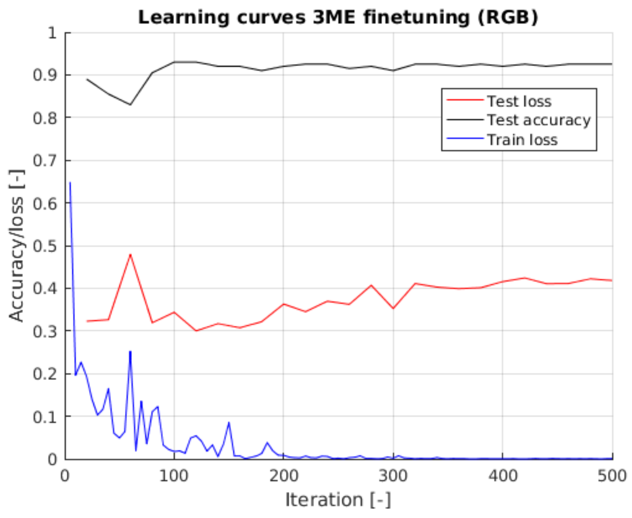


Figure 5-14: Learning curves of the finetuning experiments on the 3ME dataset.

Table 5-2: Performance after finetuning on 3ME dataset for 120 iterations

FN	FP	OE
2.73%	10.97%	7.55%

5-6-3 Further Finetuning on the IOD Dataset

The finetuning experiments on the 3ME datasets have shown an improvement in performance. Further finetuning experiments are conducted on the IOD dataset to find out whether the performance can be enhanced even further. The results of the learning experiments are shown as learning curves in Figure 5-15. Although the training loss converges to zero there is not that much improvement to be seen in terms of test loss. The best results are achieved after 180 iterations after which it starts overfitting. This led to the decision to use the learned network after 180 iterations for the further finetuning experiments. The performance of this network is shown in Table 5-3. Clearly the finetuning experiments work as intended as a small overall error rate of 4.53% is achieved, a 3.02% decrease compared to the 3ME finetuned model in Subsection 5-6-2. False positives are reduced by 5.49%. Interestingly enough, compared to the 3ME finetuned model, there is one extra false negative.

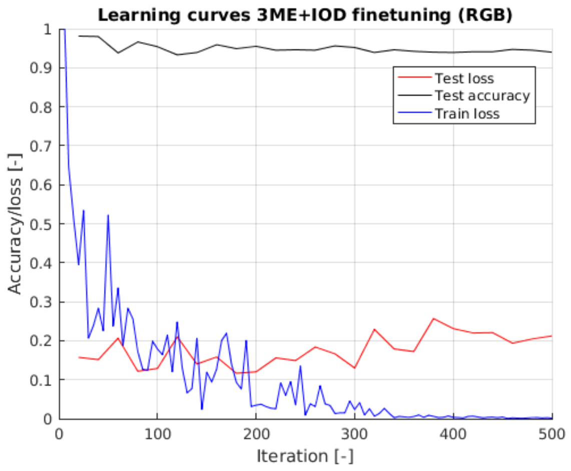


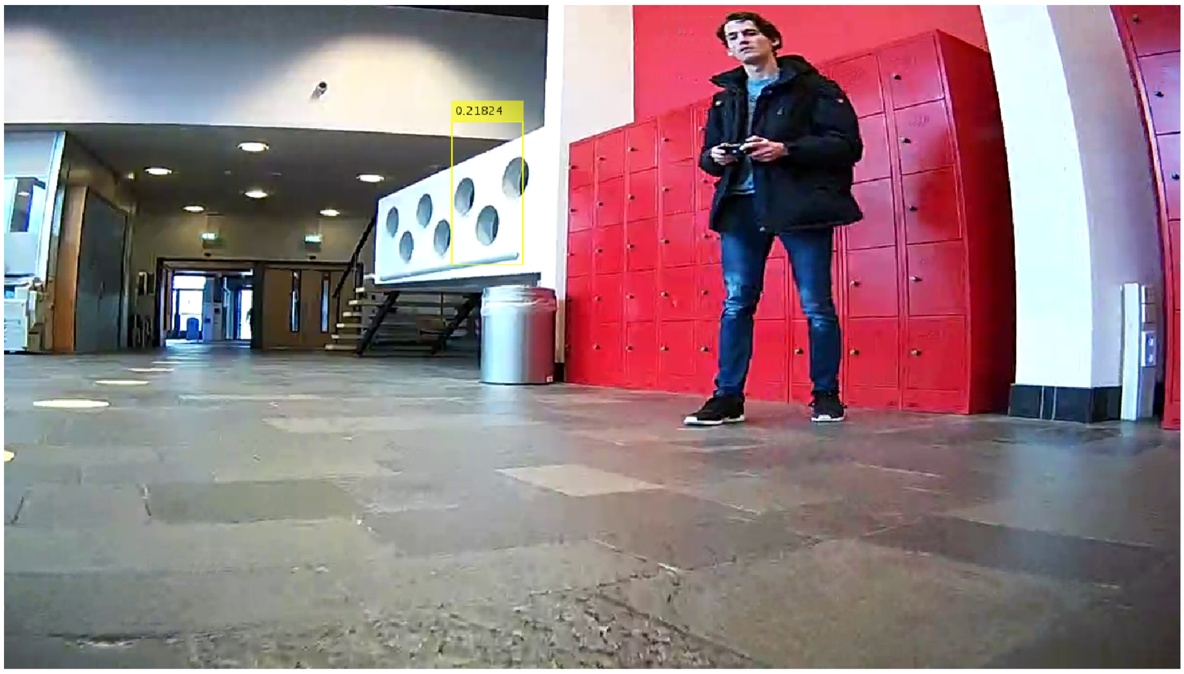
Figure 5-15: Learning curves of the finetuning experiments on the IOD dataset.

Table 5-3: Performance after finetuning on IOD dataset for 180 iterations

FN	FP	OE
3.18%	5.48%	4.53%

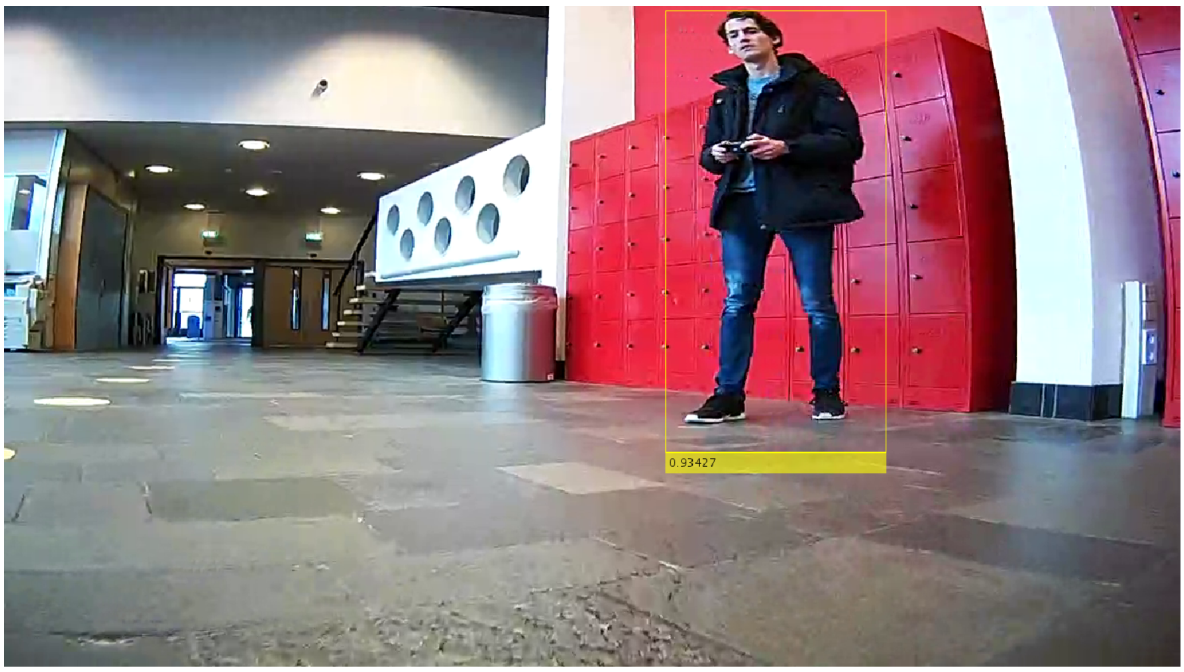
5-7 Sliding Window Experiments

The research in this thesis is focussed on a multi-sensor approach to the task of human detection. In order to compare this framework to a vision-based approach, a sliding window Support Vector Machines (SVM) classifier with HOG feature descriptors is used. This detector comes from the MATLAB computer vision toolbox (vision.PeopleDetector) and can be used for detecting humans in upright position without occlusion. For the multi-sensor architecture, Kalman-Filter Based Segmentation (KFBS) is used to create the region proposals together with the finetuned network discussed in Subsection 5-6-3. Results of human detection on a datapair from the 3ME dataset can be seen in Figure 5-16 for the sliding window classifier and in Figure 5-17 for the multi-sensor architecture. These figures show the regions where the algorithms have detected humans with their respective class scores. At the bottom of these figures the computation time is shown. The sliding window classifier misses a detection and finds a false positive in a wall. The multi-sensor architecture does detect the only human present in the image, but takes longer (0.92 [s] instead of 0.36 [s]). The forward pass through the CNN takes roughly 0.2 [s] and four region proposals are analysed in the multi-sensor architecture. Figure 5-18 shows results of human detection from the IOU dataset with the sliding window classifier. The algorithm takes 0.47 [s], but also misses one of the two humans present. Interestingly enough, the algorithm does detect a human in the shadow on the wall. Figure 5-17 shows results of the multi-sensor human detection on the same datapair. Both humans are detected, but the algorithm takes 1.07 [s] to complete, because it received five region proposals to analyse. Although the multi-sensor architecture seems to be generally slower than the sliding window classifier used in this experiments, it should be clear that it still outperforms a sliding window classifier that would use the same CNN for classification (i.e. analysing 100 region proposals is slower than analysing 5). Conclusions from the experiments are further discussed in Chapter 6.



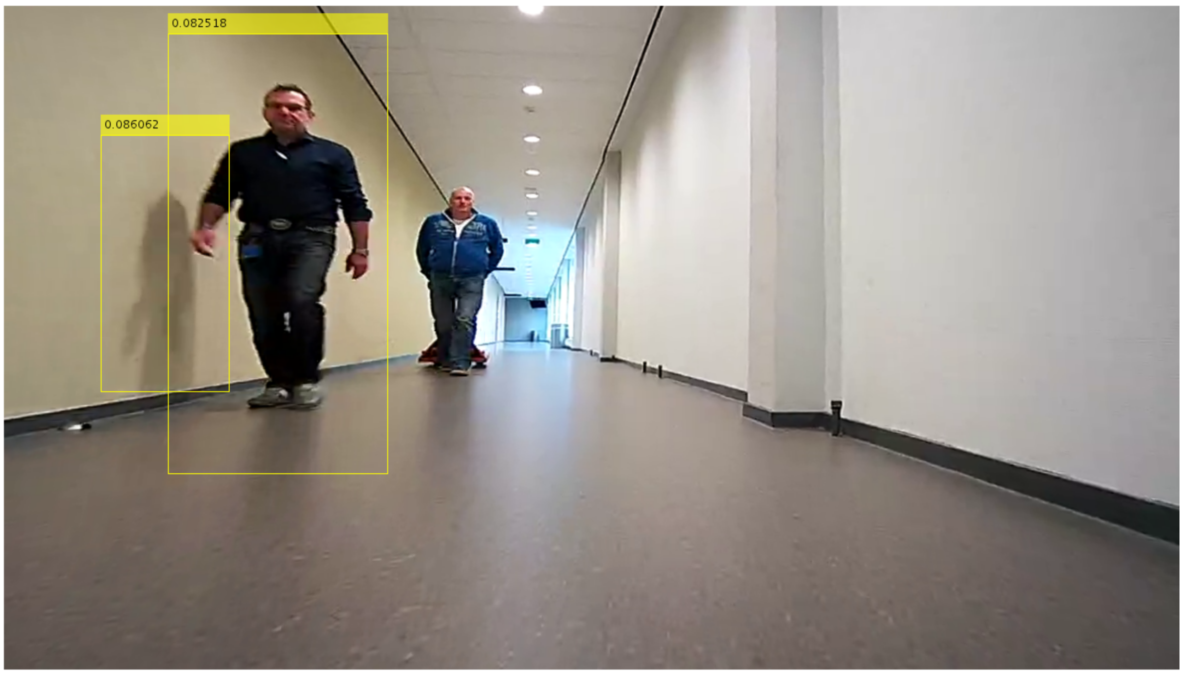
Computation time: 0.35505 [s]

Figure 5-16: Human detection results using a sliding window HOG-SVM classifier.



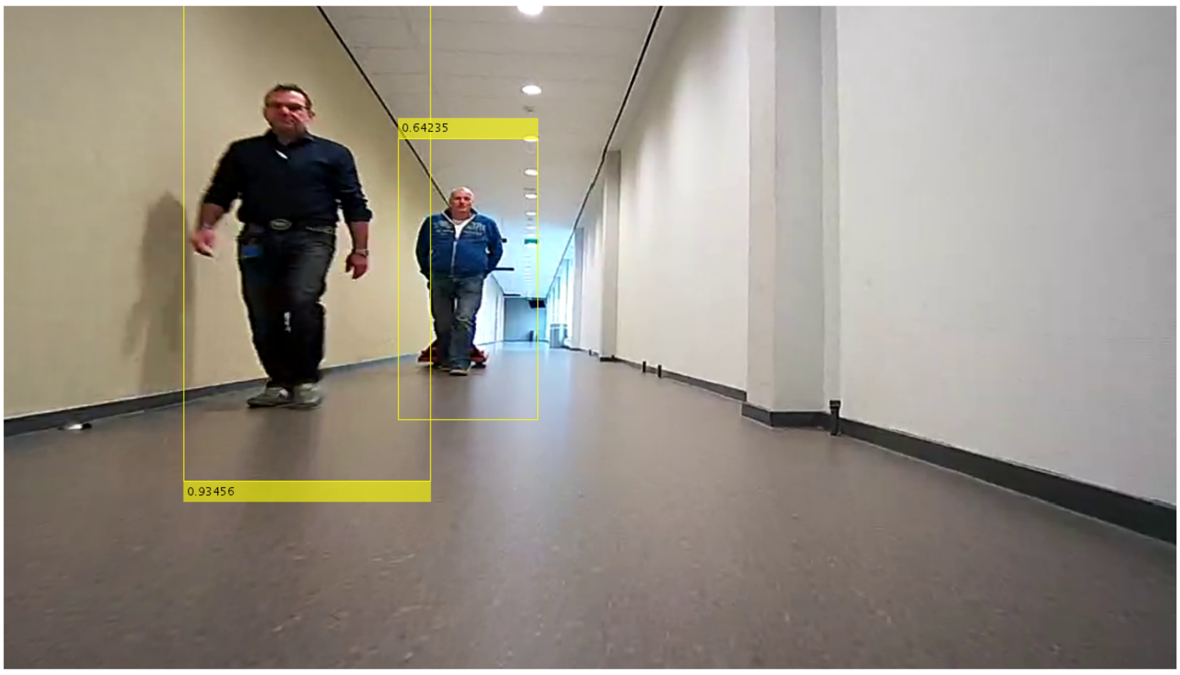
Computation time: 0.92182 [s]

Figure 5-17: Human detection results using the multi-sensor architecture.



Computation time: 0.47144 [s]

Figure 5-18: More results using the sliding window HOG-SVM classifier.



Computation time: 1.0677 [s]

Figure 5-19: More results using the multi-sensor architecture.

Chapter 6

Discussion and Conclusions

This chapter elaborates on the results that were obtained with the experiments discussed in Chapters 4 and 5. Section 6-1 contains a critical evaluation of the obtained results and their validity. Section 6-2 discusses conclusions drawn from the experiments and this chapter concludes with recommendations in Section 6-3.

6-1 Discussion

This section discusses the assumptions that were made to do the experiments discussed in Chapter 4 and Chapter 5 as well as the results from these experiments. The assumptions are discussed in Subsection 6-1-1. This is followed by a detailed discussion of the experimental results in Subsection 6-1-2 and Subsection 6-1-3.

6-1-1 Assumptions

The concept of this thesis was built around the assumption that the robot drives on a flat surface. In the surroundings of the office in The Hague and even the data recorded at Delft University of Technology (TU Delft) this assumption holds true and in fact it will for most of the indoor applications of Robot Security Systems (RSS). If, however, the same framework were to be applied in autonomous cars, this assumption can not be made. This will lead to a much more tedious camera calibration with laser sensors, because of the pitch and roll caused by rough terrain. For the purpose of this thesis, the laser-camera calibration, as was discussed in Chapter 3, could be done with the assumption that the sensors were fixed in the vertical z-direction.

The positive crops in the learning experiments (i.e. the ones containing a human) generally capture these humans in upright and walking position. This means that the network will have a significantly harder time classifying humans that are sitting or lying on the floor. For the size of the crops, a human detection at a distance of approximately 5 [m] was chosen as baseline. Assuming that humans fit within a bounding box of 1x2 [m] this ultimately led to the decision of a crop size of 200x400 pixels. Because a Convolutional Neural Network (CNN) only takes in fixed size images, crops that are larger (detections closer than 5[m]) or smaller (detections further than 5[m]) are resized to the 200x400 image size.

6-1-2 LRF Experiments

Chapter 4 discussed several experiments with the data acquired from the Laser Range Finder (LRF) sensor. Two different methods of segmentation showed the feasibility of using passive detection to create region proposals in the visual data. These methods show great strength in distinguishing objects from the distance measurements. Through the sensor calibration these proposals can then be transformed to image crops that can actually be used in a vision-based classifier.

Clustering Experiments

Experiments with two different clustering methods were discussed in Chapter 4. Point-Distance Based Segmentation (PDBS) naively assumes perfect noiseless measurements and separates the clusters solely based on a distance criterion. Kalman-Filter Based Segmentation (KFBS) accounts for measurement noise and verifies whether points belong to the segment in a stochastic sense. Figure 4-4 and Figure 4-5 show that KFBS works really well when the legs are visible, because the segments on the legs are the only ones not being filtered out. On the other hand Figure 4-6 shows that, when the legs are obscured by other objects, there are no segments being proposed from which a human can be detected. PDBS results in Figure 4-2 show that the leg segments are only a part of the proposed segments and there are some other segments proposed as well. Essentially, the experiments show that there will be less missed detections using PDBS at the cost of extra region proposals that need analysing. A choice for either of the two is then based on application and user preference.

Deep Learning Experiments

Extra experiments aimed to show the feasibility of using the same data for actual pattern recognition, by creating a neural network. Section 1-2 discussed that 2D distance measurements from a LRF sensor contain features that can be used for classification of legs and therefore humans. In the context of this research work it was interesting to see if the concept of neural networks could also be used for this purpose, because this would mean the entire human detection framework could be captured in one advanced neural network. Even though the learning curves showed convergence up to a certain point, the resulting network can not reliably be used for any detection, because its predictions are still too far off. However, this does not necessarily mean the concept itself can not be used. First of all, the amount of data is underwhelming, compared to the amount of labels that need to be predicted (300 labels). Secondly, the data that was used for these experiments was created simultaneously with the visual data. Which means that a crop was selected and manually labelled to contain a human or not. If the crop received a positive label, the laser points that were in that specific crop after projection were labelled as such. Where the ground truth vector would then contain zeroes for parts of the laserdata outside of the crop and ones for the parts within. The problem with the labelling of this data was that there is not always only one person visible within the data. This means that sometimes part of the ground truth vector contained zeroes where it actually contained humans. The same problem can occur the other way around as well, where an image crop is labelled to contain a human, but the legs are blocked from the LRF point of view. All and all combined, since the experiments do show a certain convergence, the concept of leg pattern recognition using a neural network can still be viable, but a more thorough investigation of the labelling process will be needed for this to have any chance of success.

6-1-3 Visual Data Experiments

This subsection discusses the results from the visual data experiments in Chapter 5. Firstly, the initial deep learning experiments are discussed that aimed to find a CNN design suitable for the purpose of human detection. This is followed by an elaboration on the finetuning experiments that aimed to finetune a pre-learned CNN to a small set of new data to show the strength of the network as well as its capability to adjust to a new location. This subsection concludes with the sliding window experiments that aimed to provide insight in how the multi-sensor framework discussed in this thesis compares to a sliding window detection algorithm.

Initial Deep Learning Experiments

The first set of experiments with visual data aimed to find out the best design of a CNN to be used for classifying the content of images as human or not. For this purpose three different designs were made, three different datatypes and two different solver types were used. The base network KathNet1 transformed the 400x200 image crops to a 4x9 feature map before applying two sets of fully connected layers to eventually end up with a class score. KathNet2 did the same, but not by using combinations of convolutional and pooling layers, but by only using convolutional layers with a bigger stride to reduce the dimensionality to a 4x9 feature map. KathNet3 went back to the convolution and pooling combinations, but instead of working towards a 4x9 feature map, a 3x7 feature map was applied. From these experiments three things became clear. First of all, convolution and pooling combinations tend to produce better results than convolutions with bigger strides. Secondly, KathNet3 tends to perform a little better than KathNet1. This can be caused by the fact that in general, the positive crops contained human faces in the middle. A 3x7 feature map would be able to distinguish this better than a 4x9 feature map, where the faces would be split. Thirdly, the RGB experiments generally showed better results than HSV. Apparently the RGB colorspace does contain some valuable information as to images containing humans or not. Eventually the KathNet3 network after 1,000 iterations on the RGB dataset recorded at RSS was chosen as winner.

Finetuning Experiments

The second set of experiments with visual data served to show the feasibility of using a pre-trained network on a new location. Initial experiments showed an accuracy of 87.55% on data that was never seen before. This means that the network that was trained on the RSS data generalized quite well to different environments, even though there was not much data to work with. An explanation could be that a part of the data recorded at TU Delft look really similar to the environment at the RSS building. The hallway between the Mechanical, Maritime and Materials Engineering (3ME) faculty and the Industrial Design faculty looks quite similar to the hallways in the RSS building. After finetuning on two more datasets, the accuracy on the test data had already raised to 95.47%. This serves to show that it is feasible to start with a base network and finetune it to work well on a specific location. Especially with the practical use for RSS in mind, this part is interesting, because the robots are deployed on a specific location.

Sliding Window Experiments

A third set of experiments compared the multi-sensor architecture from this thesis to a sliding window classifier meant for pedestrian detection. Interestingly enough, the sliding window classifier is faster than the multi-sensor architecture even though there is more visual data to analyse. In terms of performance the multi-sensor architecture seems more reliable, because the sliding window classifier often misses detections. However, the multi-sensor architecture in its current state is not viable yet for a real-time implementation. A forward pass through the learned network takes approximately 0.2 [s], which means that per second only five region proposals can be analysed. If KFBS is used, generally there will not be that many region proposals and the framework would still run reasonably fast at the cost of missed detections. If PDBS is used, there will be more region proposals and less missed detections, but at the cost of a significant increase in execution time. A more permanent solution to the apparent problem of slow execution time with the multi-sensor architecture, would be to train a network for smaller crop sizes than 200x400. This would significantly reduce the time it takes for a forward pass through the trained network, but smaller crop sizes might contain less information, making it harder to train a network to properly classify the contents of the crops.

6-2 Conclusions

The goal of this research was to thoroughly investigate the combination of LRF and camera sensors for the purpose of human detection. Through the experiments it has become clear that both sensors have their strengths and weaknesses. The distance measurements from the 2D LRF can, after segmentation, be used to generate region proposals in the visual data. For actual detection, the 2D measurements provide a weak base. Not because it is hard to distinguish leg patterns from 2D measurements, but mostly because these measurements are easily obscured by other objects. For this purpose a passive detector such as the PDBS is more reliable, because it passes all segments that possibly contain a human to the visual-based classifier, even if the segment itself does not seem to look like it is. KFBS does the same, but turns out to filter a lot more than PDBS. Essentially this means that using PDBS for the region proposals will lead to less missed detections, but with the disadvantage of having more region proposals to work with. The KFBS turns out to basically function as leg detector, because hardly any other region proposals are suggested. This can be very reliable if the legs are visible in the LRF measurements, but if the legs are obscured this will lead to missed detections.

In line with conclusions from Chapter 2, the deep learning experiments show great results on classifying visual content as human or background. A CNN that was trained on a large dataset and reached roughly 95% accuracy on that set, turns out to generalize well towards data from a different location, reaching an initial accuracy of 87.55%. Furthermore, finetuning experiments show that a pre-learned network can, by using a small dataset recorded at a specific location, be finetuned to achieve high accuracy (95.47%). This is interesting for applications where the detection algorithm will be deployed on a specific location.

6-3 Recommendations

How the algorithms discussed in this thesis can be used for actual applications depends on the intended application. If the user wants to detect humans that do not want to be seen, like burglars and unauthorized people, the results from this thesis are not sufficient yet. An addition that would make detections more robust, would be to include a thermal camera in the framework, making detections at night feasible. For the purpose of detecting (hidden) humans, region proposals generated by PDBS will provide a more stable base than the proposals by KFBS.

If the application requires detection of humans that do not mind to be seen, the algorithms in this thesis can be sufficient, granted the environment and humans are similar to the data used in the experiments. For a reliable use, however, more data needs to be acquired from as many locations/persons as possible to make the representation of humans as generalized as possible.

The sliding window experiments showed another problem with the multi-sensor architecture as proposed in this thesis. Its execution time is slower compared to a HOG-SVM sliding window classifier and this is caused by the forward pass through the CNN taking approximately 0.2 [s]. This can be reduced by training a network with smaller crops, but this might then lead to a CNN that is less accurate. Looking into GPU implementations could help for this purpose as well, but the higher cost of the GPU needs to be justifiable for the intended application. Further research and iterations upon this research should aim to increase efficiency of the multi-sensor architecture, because in terms of human detection the results look very promising.

Bibliography

[1] C. Premebida and U. Nunes, "Segmentation and Geometric Primitives Extraction From 2D Laser Range Data for Mobile Robot Applications," *Robotica*, pp. 17–25, 2005.

[2] K. O. Arras, Ó. M. Mozos, and W. Burgard, "Using boosted features for the detection of people in 2D range data," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3402–3407, 2007.

[3] W. B. Blom, "Reinforcement Learning of Visual Features," Master's thesis, Delft University of Technology, 2016.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[5] WildML, 2015, "Understanding Convolutional Neural Networks for NLP,", http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp (Date last accessed: 2016-06-01).

[6] A. Karpathy, 2016, "CS231n Convolutional Neural Networks for Visual Recognition,", https://cs231n.github.io (Date last accessed: 2016-06-13).

[7] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout : A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research (JMLR)*, vol. 15, pp. 1929–1958, 2014.

[8] A. Krizhevsky, I. Sulskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information and Processing Systems (NIPS)*, pp. 1–9, 2012.

[9] C. Szegedy, W. Liu, Y. Jia, and P. Sermanet, "Going deeper with convolutions," *arXiv preprint arXiv: 1409.4842*, 2014.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Arxiv.Org*, vol. 7, no. 3, pp. 171–180, 2015.

[11] G. A. Borges and M. J. Aldon, “Line extraction in 2D range images for mobile robotics,” *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 40, no. 3, pp. 267–297, 2004.

[12] M. D. Zeiler and R. Fergus, “Stochastic Pooling for Regularization of Deep Convolutional Neural Networks,” *International Conference on Representation Learning*, pp. 1–9, 2013.

[13] S. Ruder, 2016, “Overview of Gradient Descent Optimization Algorithms,”, http://sebastianruder.com/optimizing-gradient-descent/ (Date last accessed: 2016-08-05).

[14] J. Duchi, E. Hazan, and Y. Singer, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,” *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[15] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” *arXiv*, p. 6, 2012.

[16] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *ArXiv e-prints*, pp. 1–18, 2012.

[17] R. Girshick, “Fast R-CNN,” 2015.

[18] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *ArXiv 2015*, pp. 1–10, 2015.

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[20] R. Hartley and A. Zisserman, *Multiple View Geometry in computer vision.* Cambridge University Press, 2nd ed., 2004.

[21] Z. Zhang, “A Flexible New Technique for Camera Calibration (Technical Report),” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2002.

[22] Caltech, 2015, “Calibration Toolbox,”, https://www.vision.caltech.edu/bouguetj/calib{_}doc/ (Date last accessed: 2017-01-26).

Glossary

List of Acronyms

3ME	Mechanical, Maritime and Materials Engineering
TU Delft	Delft University of Technology
RSS	Robot Security Systems
NN	Neural Network
CNN	Convolutional Neural Network
SVM	Support Vector Machines
ROI	Region of Interest
LRF	Laser Range Finder
PDBS	Point-Distance Based Segmentation
KFBS	Kalman-Filter Based Segmentation
ILSVRC	ImageNET Largescale Visual Recognition Challenge
FOV	Field Of View