Bachelor Thesis

Evaluation and Improvement of the Current Victim Detection of a Rescue Robot

LORENZ Peter

Graz University of Technology
Faculty of Computer Science
Institute of Software Technology

Supervisor

Ass. Prof. Dipl.-Ing. Dr. techn. Gerald STEINBAUER

Contents

Abstract

This bachelor thesis presents an analysis of the current state-of-the-art of the Wowbagger's Face Detection, in order to find a better approach for detecting victims, in our case dolls. Wowbagger is the name of the robot, who will send regularly to competitions. The focus is mainly on image processing for autonomous robots operating for detecting victims in crisis area. The thesis discuses the methods that are more adequate to be applied in a rescue scenario and some parts will be chosen to implement an optimal doll detection for our robot.

It will be discussed one of the main objectives of rescue missions is to rescue the disaster victims. Of course, in some situations it is too risky to send human agents in the unstable environment. Therefore, robotic agents can be essential in such situations and important collaborators aside human agents.

One crucial requirement in above-mentioned rescue scenarios is to identify victims from all other objects. A task, that ought not to be difficult for us humans, but indeed, it is for robots. A rescue team benefits from the abilities of an autonomous robot, which is able to intervene dangerous areas.

In order to detect an victim, an autonomous robot must must be equipped with a set of specific sensors, those provide information about the environment.

In this case, we have "Wowbagger", the autonomous robot of the RoboCup Rescue Team of Graz University of Technology.

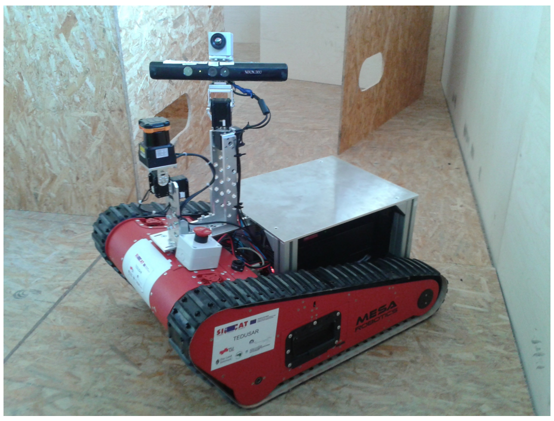


Figure 0.1: Wowbagger

Camera	Asus Xtion Pro
Distance of Use	0.8 m - 3.5 m
Resolution	SXGA (1280 * 1024)
Frame Rate	30 fps / 60 fps
Sensors	RGB, Depth, Microphone
Field of View	58° H, 45° V, 70° D

1 Researching of three Existing Algorithms

1.1 Overview

This section lists three approaches how to detect a victim, which should represent a victim in real life. The dolls are used in RoboCup rescue competitions. The faces are different to real faces and there are not many significant areas, which could be used for detection.

E.g., eyebrows are much more darker than the skin and could be detected as an edge, as well as the eye sockets, which could cast shadows. Other significant areas are the nose and hair, which help to train a pattern for a detector.

However, object detection in an image frame can be reached by several methods. Human body detection is generally addressed by three steps:

1. **Segmentation:** In this step, we can differ an object by a set of connected pixels. (See section 1.4)
2. **Classification:** Those connected pixels can represent a part of an human body. It is ordered in predefined classes.
3. **Modeling and Recognition:** Finally, the classified parts will be composed so that it matches an human body model.

Usually, the second and the third step are iterative in almost every algorithm to achieve a more precisely result.

Beside the algorithm, there are other elementary questions arising, for example, do we use the right resolution of our camera. If yes, is there a similar camera model which is by far better than the others? Are there some key points in detection, which have been neglected for some reasons. There are not only RGB images, RGB-D can can yield the necessary information for a successful detection.

Another fact, which the size of the training and validation set are common? The victims are normally not looking frontal into the camera. A victim can lie on the ground and the head is bent over to one side.

Finally, which results are realistic? In other words, how good are detectors in those domain in respect to very similar circumstances as we have with our robot. Such guide values shows, when a success is achieved or if there must be done further work.

1.2 Human Detection in Domestic Environments [MCS12]

This cited paper is not exactly for RoboCup Rescue, but it provides quite usefully information on face detection. Since domestic home robots are concepted in domestic environments should it fit for our requirements, because the field of view ($< 6m$) is more or less than the competition area.

The basic skills of those robots are the ability to move autonomously in domestic environments, the ability to detect and manipulate objects, such as cups, books, glasses, chairs and so on.

They state that a robust detection of humans is very challenging and depending mainly of variable illumination conditions, cluttered backgrounds, and variable poses of a human body with respect to the robot's camera.

Their face detection is based on the use of a multi-scale object detection framework that uses boosted cascade classifiers. It is used for both visual and thermal detection.

	Asus Xtion
Thermal Camera	324 x 256
Visual Camera	640 x 480

Table 1.1: Camera Description

The algorithm:

1. **Multi-Resolution Analysis Module:** When a face detected (expected one face per image), it is performed by a down scaling by a fixed scale factor.
2. **Window Extraction Module:** Windows (24x24 pixels) are extracted for each of the scaled version of the input image.
3. **Nested Cascade Classifiers Module:** Windows are analyzed. The classifiers consists of several stages of classifiers of more complexity to achieve a faster detection. Non-objects windows should be sort out as fast as possible in order to process the windows with objects in it. It is based on the Ada Boost, a domain partitioning weak hypotheses. Each of them is given a self-rated confidence value with that can be estimated the reliability of each prediction.

$$h(f(x)) = c_j \epsilon f(x) \in F_j \tag{1.1}$$

$h(f(x))$ are the weak classifiers. Each of them are associated to one partition block c_j. The outputs are obtained during a training, which are stored in an LUT to accelerate the evaluation.

	Frontal Face	Non-Face
Training Set	5000	3500
Validation Set	5000	1500

Table 1.2: Image Database

4. **Final Detection Module:** Windows will be classified (positives and negatives). The positives are fused and a face will be detected at varying scales or positions. The size and right position must be obtained.

Result The detector works robustly for faces that are close to the camera ($< 3m$). Consequently, the detection rate decreases when the faces are more far away from the camera.

1.3 Human Victim Detection [GDC09]

In this paper, there is another approach to achieve a robust victim detection in outside conditions. This time, there is discussed a combined approach. With such a multitude approach, it can be provide a better result, if each detector weighed by is advantages and disadvantages. Moreover, in comparison to the domestic environment approach it is included that the pose (standing, lying, partly covered) of the victim in the calculation.

The basis is a learning-object detection method as an extension the Viola and Jones detection [VJ01]. They chose it, because Viola and Jones is known as the fastest and most precise pattern recognition method for faces in gray-images.

	Indoor	Outdoor
Validation Set	90%	60%

Table 1.3: Accuracy

During training, it will be used the Ada Boost to select weak classifiers. The authors decided to train with victims, lying on the ground, where the head is partly hidden. It makes the whole scenario more realistic, since people may protect their head before the black out.

	Frontal Face	Non-Face - outside	Non-Face - inside
Training Set	800	500	100

Table 1.4: Image Database

Those negatives are important, because of the variety of the background is high in order to get good threshold by the Ada Boost learning method

Result The correction rate was about 65%. The victim was correctly found besides of a lot of false positives. These false detection were be sort out by merging overlapping rectangles of correct pose.

The detection run time is surprising: Between 60 and 80 ms, that means about 13 till 16 frames per second. Despite of that it is questionable that the detector fulfill the conditions, which we are needing. The authors statement: „A mobile robot equipped with such a human victim sensing system can be a valuable aid for human crisis managers, as it could scan a designated are for human survivor semi-automatically.“ Semi-automatically, because of the high number of multiple detection.

1.4 Fast Human Detector for Indoor Mobile Robots Using Depth Images [BCV01]

This time, this paper aims at mobile robots, which possess limited computational resources for autonomous operation. So the human detector must have a small computational footprint. Various possibilities are there shown in Figure 1.1 underneath. On the right side you can see the raw image. The second one describes the regions after segmentation, the third is result after filtering and last one shows the detected humans within bounding boxes.

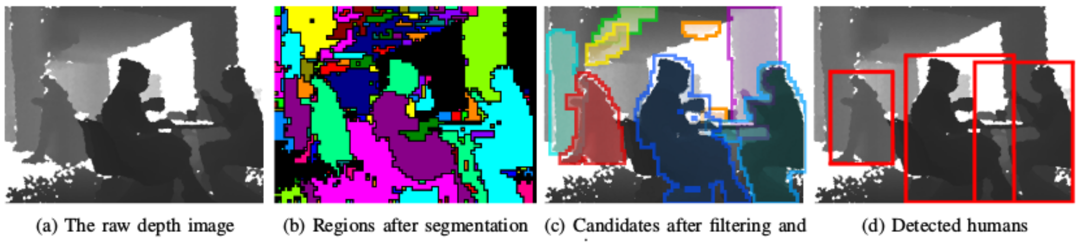

(a) The raw depth image
(b) Regions after segmentation
(c) Candidates after filtering and
(d) Detected humans

Figure 1.1: Domestic Room

In order to avoid the brute-force method (sliding window over the whole image), they decided to take a graph-based segmentation that works with depth values and surface normal's to attach the regions into the image. After region extraction, it is evaluated by HOD [SA11] whether a human is detected or not.

	Microsoft Kinect
Resolution	640x480
Hz	30

Table 1.5: Camera Description

Algorithm can be described in three Steps:

1. Depth Image **Segmentation**
2. **Region Filtering and Merging:** Takes a set of regions R and returns a set of candidates C. Depending on the regions positions, they will be merged.
3. **Candidate classification:** After the HOD descriptor is computed, images will be classified by a SVM.

	Frontal Face	Non-Face	Comment
Hallway Set	1000	5000	96% precision by a recall of 70%
Café Set	1300	1350	<50% visible by 0.5-5.0m

Table 1.6: Image Database

1.5 Summarize

Analyzing those three papers is crucial to get a first sight into this complex subject matter and to figure out some analogies, which can be useful for further work.

To start with the technical equipment and the eye of computer vision, the camera. The cameras are more or less the same in those papers. Microsoft Kinect and the Asus Xtion are mentioned. Both of them are using a resolution of 640x480 pixels, that is also used at our Wowbagger, our robot.
This is reasonable for having a good center between image quality and limited computational resources on mobile robots. As it is stated in Table 1.6 and section 1.2 the distance between robot and victim is in most cases 0.5-5.0 meter. section 1.2 also says that the detection rate is decreasing from 3m distance.

Another point is that, which algorithm are used in those papers? In the paper of section 1.2 there is the Ada Boost algorithm described, where several weak classifiers are bound together in order to boost the detecting rate. In section 1.3 the Ada Boost is used as well, but trained with some differences.

The aim of their detector is that it can also recognize faces which are partly hidden. That is the reason why they are having chosen collection of negative examples. Both papers stated how many positives and how many negatives they provided for training their detector and how good their detection rate is in the end.
Those figures give a good guidance in order to know when a good performance is reached and when not.

The last paper, section 1.4, is the different to the others mentioned before. This paper is about detecting humans in domestic area. Instead of the Ada Boost learning algorithm, there is used a depth image segmentation, followed by HOD.

2 Evaluation of the existing Face Detectors

2.1 Overview

This section shows the result of the benchmark of the current detector, where as a database of taken pictures during a training will be compared and by means of a ROC curve illustrated. Moreover, there will be a link provided for downloading a ground truth of doll victims in a training area as known as in RoboCup Rescue.

2.2 Description of the Haar Cascades

The face detector uses OpenCV's Cascade Classifier to detect objects in the RGB Images of the Kinect. The Cascade Classifier is based on [VJ01]. OpenCV [1] and adapted to the Wowbagger's needs. As Cascade Classifiers will be used the well known Cascades for frontal faces and eyes/eyeglasses. These have been proven for a excellent human face detection. Haar-Like features are rectangles and is defined as the differences of a sum of pixels within a rectangle. They can be at any position and scale within the original image. The algorithm can handle 3 types of features:

1. Edge Features
2. Line Features
3. Center-surround features

The big Advantage of the Cascade Classifier is its fast computation.

According to the former master thesis [Mei14], there are several factors, which are taken into account for detection, such as heat, blob detection (hole), number of detected objects in distance to each other, etc.

This detector runs all the time, because of low CPU usage, in compared to the CNN detector, that is only switched on by certain prefactors, e.g. heat.

The origin algorithm is modified to get more results for the evaluation, that's why the detector is rotated three times and not just flipped horizontally two times. The

[1] http://www.docs.opencv.org/doc/tutorials/objdetect/cascade_classifier.html

reason is to find eventually more results. In Figure 2.1 is shown schematically, how this detector works in a very simplified way. After the state „INIT“ the picture is investigated, if there are faces and takes the biggest one. The results are saved temporarily until all rotations in the if-block are done. In the state „EVALUATE“, the best result (highest percentage of an overlapping rectangle with the ground truth) of all 4 perspectives are chosen.

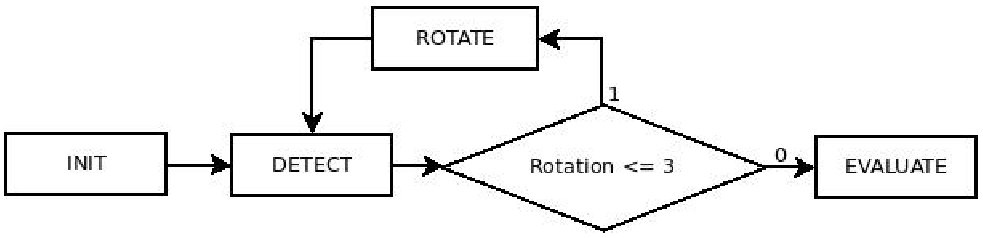


Figure 2.1: Scheme

Why rotation? One weakness of the haar cascade is that it only faces where the chin is nearer to the bottom margin of the image and the eyes are nearer to the top margin. That what we have to do is to rotate the image three times to detect faces by all positions.

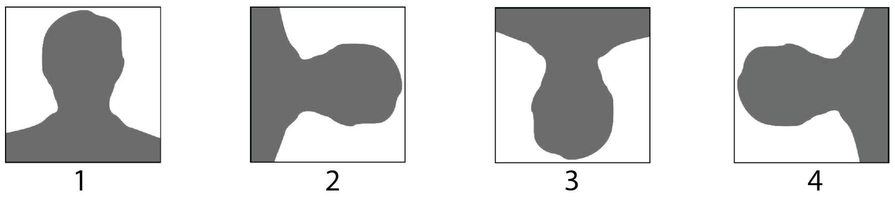


Figure 2.2: Rotation

2.3 Description of the current Face Detector - CNN Overfeat

The CNN Overfeat[2] detector is developed by [?]. Basically, it converts an image recognition CNN into a sliding window detector by providing a larger scale resolution image and transforming the fully connected layers into convolutional layers. Then, after converting the fully connected layer, which would have produced a final feature vector, to a convolutional layer, a grid of final feature vectors is produced. Each of the resulting feature vectors represents a slightly different context view location within original pixel space. The CNN Overfeat has originally over 12.000 classes to differ, whereas all of them a thrown away and the last layer is replaced by an SVM to detect the baby face, baby clothes.

The implementation of the Wowbagger does not run the the CNN continuously as the Cascade Classifier, because it is too much CPU usage intensely for a mobile robot.

2.4 Image Database

The image database consists of 1701 test pictures, where as 1267 are positive images (with at least one victim on it) and 432 are negatives (without any victim on it). All images have the size 640x480pixels and are in RGB colors. Those images can be accessed via link_to_images[3]. To every positive image exists a XML file, which stores the information of the polygon around a face. The XML files are created with label-me[4]. Most pictures are taken by letting our robot driving autonomously through our test arena in our laboratory. The advantage of this method is that the angle and height is exactly as how our robot can see the victims. Despite of that, numerous images are taken by hand to get exactly the edge cases and taking a lot of negatives in order to avoid over-fitting in advanced.

2.5 Test Environment

The test environment is implemented in ROS indigo.

In Figure 2.3, there will be shown the following components:

1. **my_publisher**: This node reads out all files in a directory, and saves the path and the name of the picture. Then the file is read according to the path and filename and send to the next component „overlapping_rect“

[2] `http://www.ist.tugraz.at/ais-wiki/tedusar_cnn_detection`
[3] `http://git.ist.tugraz.at/ais/face_detection`
[4] `http://labelme2.csail.mit.edu`

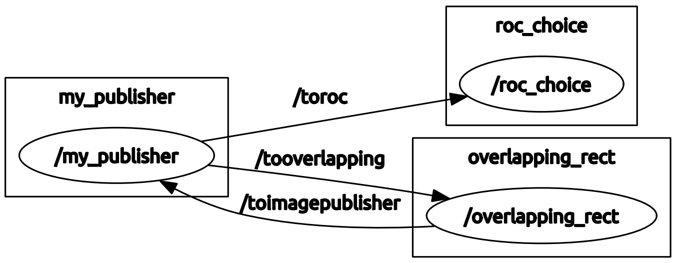


Figure 2.3: Message Communication

2. **overlapping_rect**: represents the detector, which can be any SVM. It subscribed the images published by „my_publisher“ and saves in case of Haar Cascades the top left point and bottom right point of the rectangle for further evaluations in a custom message or in case of a CNN a Boolean (victim / no victim).
3. **roc_choice**: The custom message is send from „overlapping_rect“ to „my_publisher“ to „roc_choice“. This node evaluates the custom message and reads the matching XML file as ground truth.

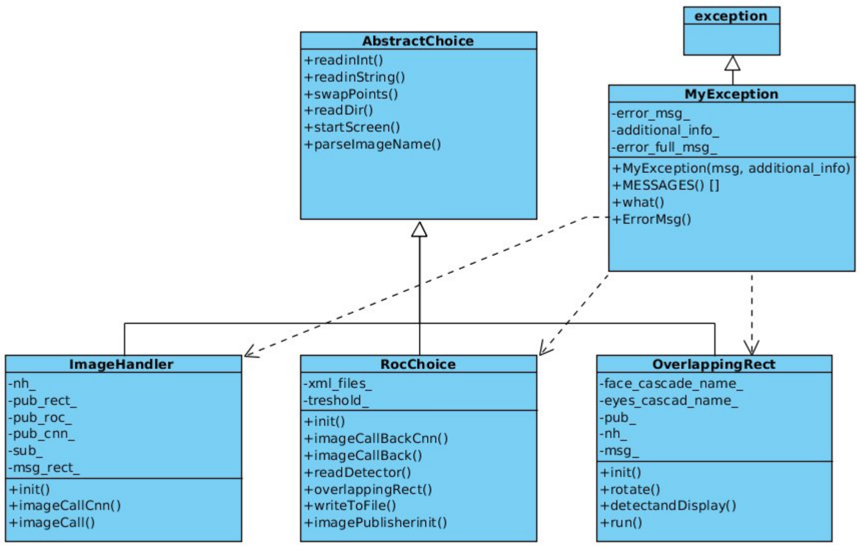


Figure 2.4: Class Diagram of tedusar_detect_evaluation Package

In this subsection, there will be shown some measurements. As ground truth there is an online library[5], which can be accessed and expanded by everybody.

Then, the same library is taken, those images run through the original face detector used by the Wowbagger. Afterwards, we have two pictures of the same object and the detected must be compared. That happens by intersection both extracted

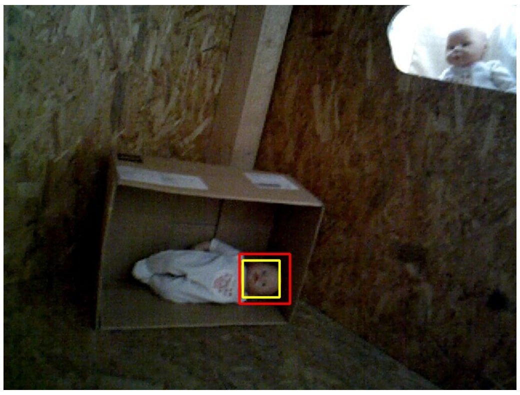
[5] http://labelme2.csail.mit.edu/Release3.0/browserTools/php/browse_collections.php?username=0664jester

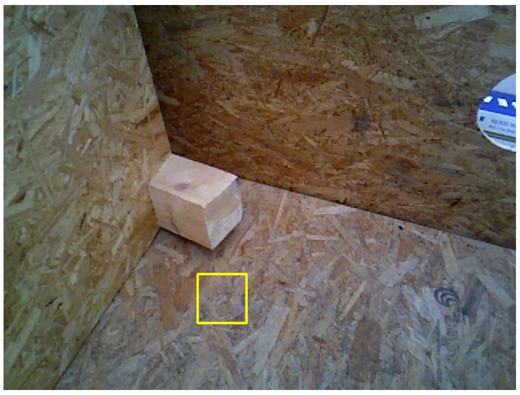
patches. A decision is made, if and only if the intersection is about greater equal the threshold, so it is the same position of the image and consequently it is a true positive, otherwise a false positive.

		Ground Truth	
		Victim	No Victim
Detector	Victim found	TP	FP
	Victim not found	FN	TN

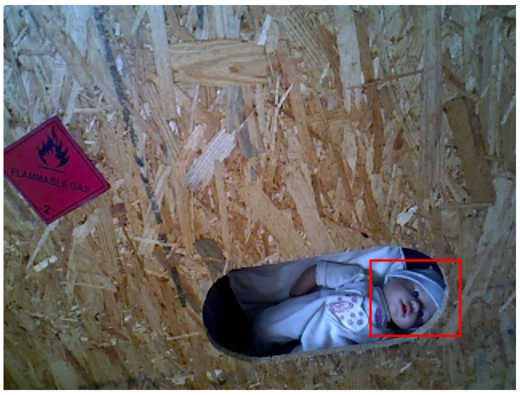
Table 2.1: Definition of Binary Classification

Definition of the scoring:


(a) True Positive


(b) False Positive

(c) True Negative


(d) False Negative 1

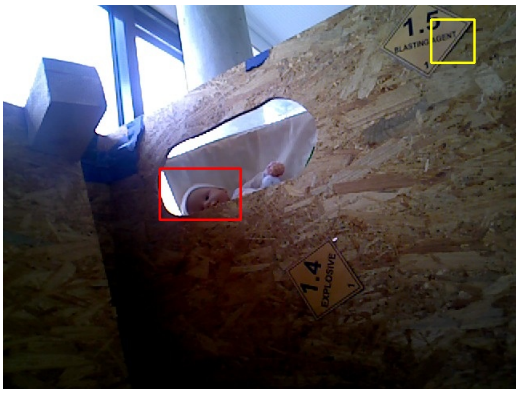

(e) False Negative 2

Figure 2.5: ROC Cascade Classifier

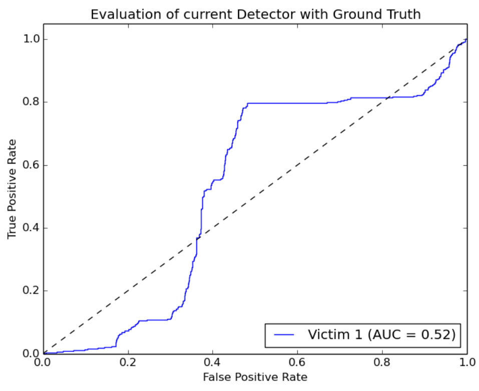
Again, in Table Table 2.1 on page 17 are explained the binary classifiers, which are described in Figure Figure 2.5 on page 18, that all cases are shown, how the detection are evaluated. The red rectangle represents the ground truth (read from the XML files from label-me) and the yellow ones are the detection of the current detector. Figure 2.5e on page 18 is a prime example, although there is the victim on the picture, the detector recognizes it on the wrong position and therefore there is no intersection between them.

2.5.1 Results

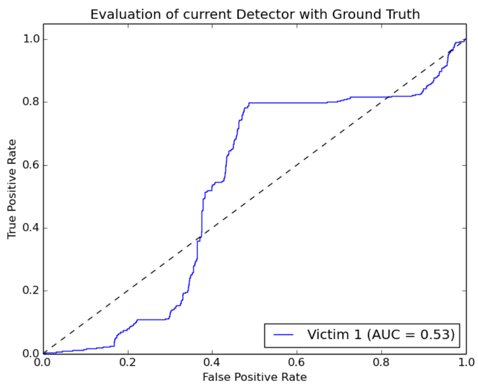
	haar cascades (Threshold 30%)	haar cascades (Threshold 70%)	CNN	HOG
TP	401	379	907	216
TN	529	550	28	715
FP	765	765	406	766
FN	1	1	360	0
Sum	1701	1701	1701	1701
Acc in %	52	53	58	34

Table 2.2: Results

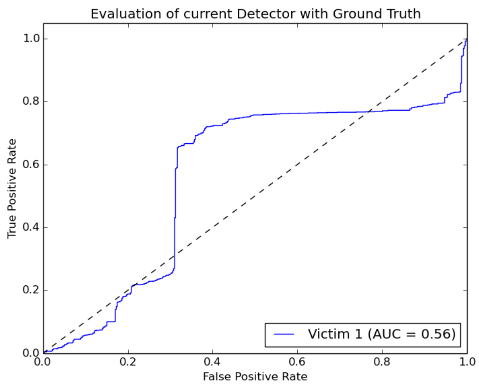
The following four images are the corresponding ROC curve to the table above:



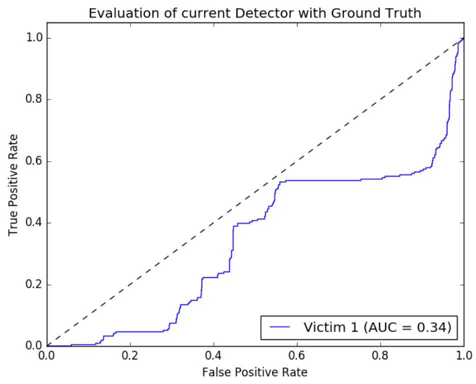
(a) haar cascades 30% Treshold



(b) haar cascades 70% Treshold



(c) CNN

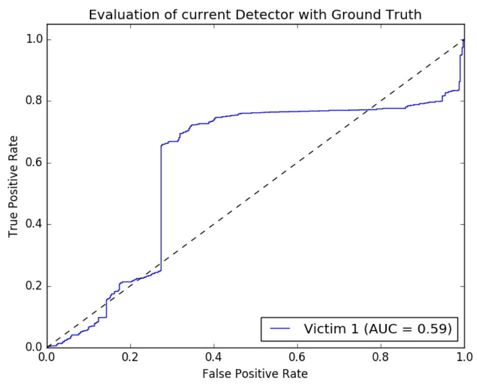


(d) HOG

Figure 2.6: ROC Cascade Classifier

The haar cascades detector shows up an AUC of around 52 - 53 percent by 1701 pictures. The 2.6a as well the 2.6b are very similar despite of the thresholds. Only figures the true positives and true negatives are changed (see Table 2.2). The number of pictures referencing to true positive are decreasing and the true negatives are decreasing, if the threshold is changed from 30% to 70%. Those changes are only very little and it also reveals that the detector reacts very extremely in two directions. If the victim is supposedly detected, in one half of the cases their is one, the other half, there is no victim notified in the ground truth.

In Figure 2.6c on page 20 is the ROC curve shown, of the current CNN detector by Overfeat, adapted to our needs. In Table Table 2.2 on page 20 there is the distribution shown. The detector is very good at detecting, a doll. The worst side is that it has also a high number of false negatives and false positives, which keeps the AUC down. The combination that the CNN is only used when the heat detector finds something, tells us good results in the end.



(a) CNN

	CNN || haar (Threshold 70%)
TP	916
TN	14
FP	406
FN	360
Sum	1701
Acc in %	59

Figure 2.7: CNN or haar cascades

One way to boost the results for our current detectors, we *'or'* the results of the Haar Cascades and the CNN. Since the CNN has a high False Positive rate, there must be way to put down this rate.

Time Measurements

The time of the algorithm are important for being competitive at this time the robot is standing calm, otherwise it can not get any clear pictures of the victims to detect. The measurements are different to the algorithms.

In case of the Haar Cascades, measurements also include those three rotations as shown in Figure 2.1. Compared to it, the CNN's input stays as it is. A list[6] of time measurements are given. The arithmetic mean is calculated by those values:

[6] /pythonScripts/results/time_measurement_XY

$$Average\ Time = \frac{1}{n}\sum_{i=1}^{n} t_i \tag{2.1}$$

'n' represents the number of all measurements and *'t'* is one time measurement in seconds.

Algorithm	Average Time (in seconds)
Haar Cascades	0.2389569347
CNN	1.9171652047

Table 2.3: Average Time

Those time measurements (Table 2.3) are only valid by a picture size of 480x640 pixels image.

3 Own approach of a Face Detection

3.1 Overview

An improvement of the face detection has many benefits due to the detection system, it is proposed in the the next section. Beside other features (heat, CO2, ...) the face detection is the most weighted and is therefore very important for accurate results. This chapter is concentrated on the accuracy of the detector, not on speed or other specifications.

The decision, how the detector should work on is based on the , the AlexNet, where the last layer ought to be a SVM for deciding whether the input is a victim or not. The SVM is another advantage, because it should only separate linear (Doll is true. Other is false.) data inputs and this linear kernel is very computational inexpensive and can also be used for prediction for a CNN, what is always computational expensive. The SVM must not only be used for the AlexNet, it could replace the Haar Cascades which results in moderate accuracy.

In the following sections, there is a short investigation of the current state of the detection state and how the preprocessing works for the CNN.

3.2 Detection System

The detection is crucial for preprocessing, because of the rectangle, which enclose a whole or a supposedly victims leading to further steps, such as preprocessing methods, which will be introduced in the next paragraph.

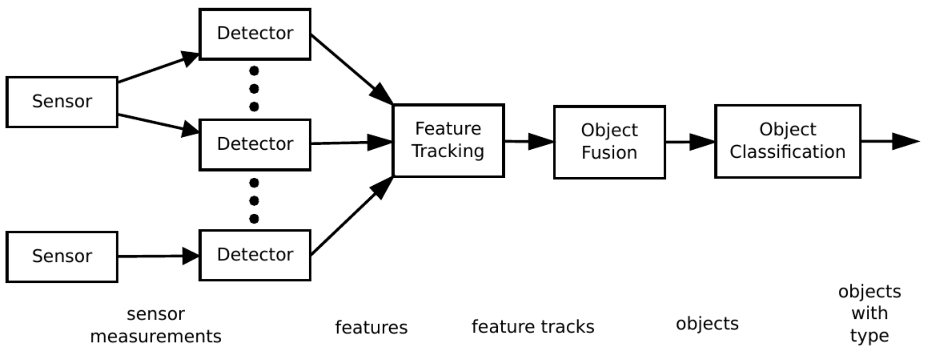


Figure 3.1: System Overview [Mei14]

In Figure 3.1 is the current solution illustrated and currently implemented. The first step is about feature detection. There are a number of sensor inputs to detect features, whereas in our case is the RGB-D camera the crucial one, but also others would be available, such as C02 sensors or position estimation in the world coordinate system as well heating sensors.

3.3 Preprocessing

In Table 2.3 is the speed of the CNN detector, which is about 1.9 seconds (see Table 2.3) at an image size of 640x480 pixels. In the master thesis [Mei14] (page 15f) the author stated that a resolution of 640x480 has a working range of 0.7-6m. However, it is rather slow in real-world environment, because being sure that there is no blurred picture send to the CNN, there will be 3 pictures of the same point sent to it. The estimation time is increasing to 6 seconds.

One concept would be to preprocess the picture for having a patch. Therefore, an algorithm must crop a rectangle of the victim's face. But before doing, the question of efficiency arises and a test started how much would the picture size affect the time of the CNN?

The cropped patches of the HaarCascades detector were saved and then it served as the input of the CNN:

Algorithm	Average Time (in seconds)
CNN	1.8879085647

Table 3.1: Average Time for Cropped Images

The size of the cropped images ranges between 40 and 56 pixels width and height. It is much smaller than the original one, but the speed assumption performance is not meet. The reason for that disappointing result is that the CNN is always rescaling the images to its needs.

3.4 Detector

In this section, there are different kind of SVMs tested, which are based on the same image ground truth. The True Positive image is cropped from the label-me library, where only the doll face can be seen. The number of the those images engages a number of 871.

3.4.1 HOG SVM

The implementation used the python libraries sci-kit-image[1]. The goal as it should look is given in Figure 3.2[2].

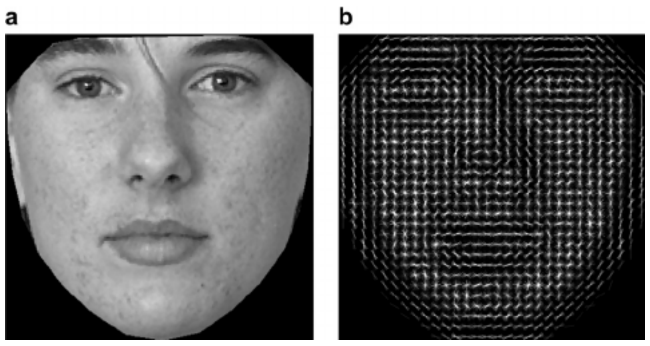


Figure 3.2: Goal of the HOG detection.

In 2.6d the DLIB HOG detector results below average. The HOG detector which is provided by the sci-kit-image library can be better handled by tweaking parameters. In the following figures are number of gradients shown with different parameter values.

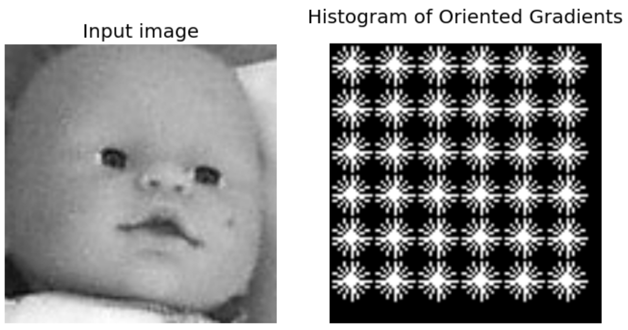


Figure 3.3: hog(image, orientations=8, pixels_per_cell=(16, 16), cells_per_block=(1, 1), visualise=True)

[1]http://scikit-image.org
[2]https://www.researchgate.net

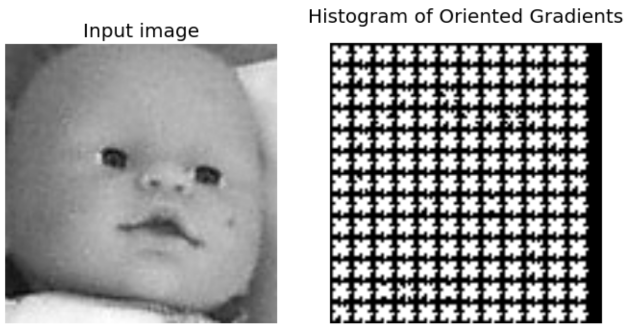


Figure 3.4: hog(image, orientations=8, pixels_per_cell=(8, 8), cells_per_block=(1, 1), visualise=True)

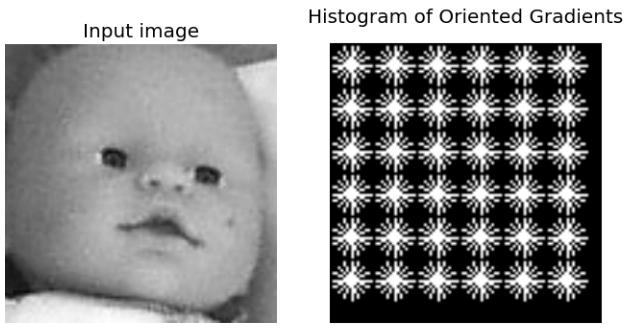


Figure 3.5: hog(image, orientations=8, pixels_per_cell=(16, 16), cells_per_block=(3, 3), visualise=True)

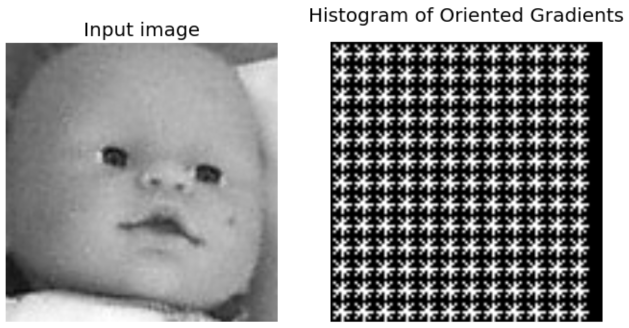
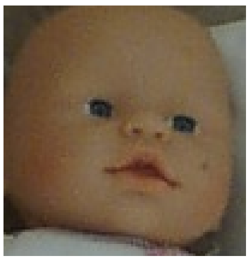


Figure 3.6: hog(image, orientations=4, pixels_per_cell=(8, 8), cells_per_block=(3, 3), visualise=True)

In the example figure Figure 3.2, you can recognise a face-similar gradient pattern, where as in none of the doll faces there is nothing noticeable. At this point, it is not worth to train the SVM, unless features are found.

3.4.2 Histogram SVM

Another approach is to compare the RGB values, by using a histogram.


(a) original image

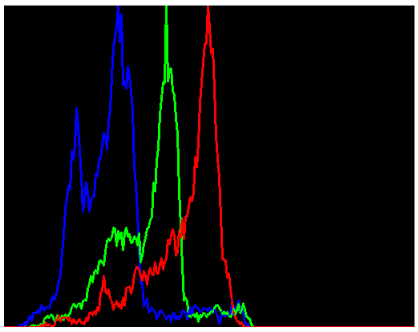

(b) RGB Histogram (x-axis intensity, y-axis number of pixels)

Figure 3.7: Baby Face Histogram

(a) original image

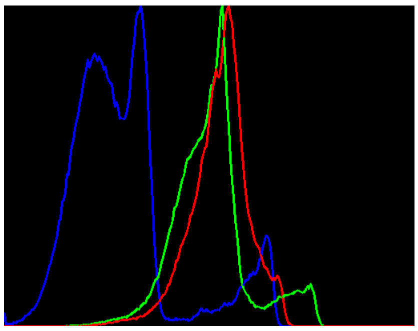
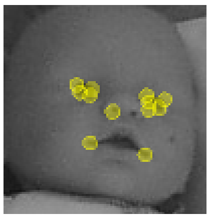
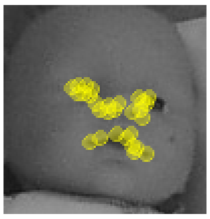

(b) RGB Histogram (x-axis intensity, y-axis nr of pixels)

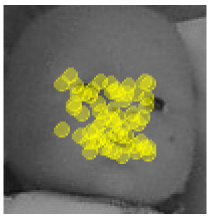
Figure 3.8: Negative Example Histogram

3.4.3 Scale Invariant Feature Transforms - SIFT

The MSER [JMP02], in contrast, to the RGB colour histogram results better in finding features, which can be used for a face detection. In Figure 3.9, there are several known detection algorithm shown. By examination of the key points, in 3.9a shows some unambiguous spots on the both eyes and in the corner of the mouth. In comparison to it, in 3.9b there is a pattern visible, but not as unambiguous as in 3.9a. 3.9c shows a block of features, it could be every object indeed. Last but least, 3.9d, yields a very unambiguous result, which can be used for victim face detection.


(a) Brisk Brisk


(b) Fast


(c) GFTT

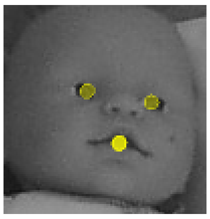

(d) MSER Brisk

Figure 3.9: Testing several Detectors

BRISK should be faster than SURF or SIFT [SLS13], that is stated by S. Leutenegger et al. All results are tested on a laptop with a quad core i7 2.67 GHz processor (only using one core), Ubuntu 10.04 (32-bit). Thus, it can be used on the robot's hardware and is not computational expensive. He also said that SIFT finds the most key points, while BRISK the fewest, so that the accuracy may not be higher rated than the speed.

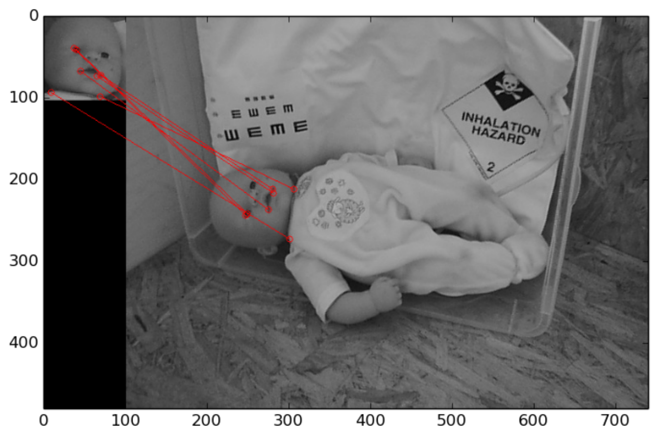


Figure 3.10: SIFT - Homography

The homography (Figure 3.10) exhibits the rotation invariance of SIFT, which means in our case, that the prestep to rotate the picture is not needed anymore as currently used in the Haar Cascades and CNN Overfeat. SIFT finds features on the whole image and thus it must be limited. It is known that the SIFT's disadvantage is to find the same key points in different illuminations.

The framework follows the principle of bag-of-words and works basically in 4 steps:

1. **Extract SIFT features from each image:** Now, there are only 128 dimensional raw features, which are not enough, that's why visual vocabulary is needed to map high dimensional descriptors to words by quantizing the feature space.
2. **Setup a visual vocabulary from the training data:** The vocabulary is achieved through applying a clustering algorithm. In our case, the K-Means [Llo05] is executed on the available features, where the final centroids are learned as a new word. The average complexity of LLoyd's algorithm is $\mathcal{O}(n \cdot k \cdot d \cdot i)$, where n is the number of d-dimensional vectors, k the number of clusters and i the number iterations until to convergence, in fact, it is polynomial. The notation has polynomial computational power and can be too expensive for huge data sets as it is common in computer vision. His algorithm counts to the naive implementations, therefore another approach is taken into account: Mini-Batch-K-Means [Scu09]. This algorithm only takes small randomly chosen batches for each iteration. Then it assigns a cluster to each data point within the batch, depending on the former locations of the cluster centroids, those are updated based on the new points in the batch following the rules of gradient descent. Once the vocabulary is learned, a histogram (length corresponds to the number of words) is created for each image. Then, the features of images are analysed to construct histograms. For every SIFT feature, the nearest word to this feature looked up and the bin, which is responsible for this word is increased by one. Afterwards, one main histogram is set up and combined by all sub histograms, that represent the entire image.
3. **Pyramid Matching:** In this stage, we already have all histograms. In order to distinguish the data more discriminatively, Grauman and Darrel [TD05] introduced the pyramid matching algorithm in 2005 for finding approximated correspondences between two data sets. In general, the pyramid matching merges information from multiple levels built on different resolution. Consequently, it considers the spatial information, which can be ignored, because features are matched without any order.
4. **Classify the objects via SVM:** Classification refers to the problem to identify the objects on an image. Usually, there are two different kernel types to seperate objects depending on the situation, such as linear, rbf, polynomial or sigmoid. In this case, it is easier to take the Gram matrix:

$$G = X^T X = \begin{bmatrix} x_1^T \\ \vdots \\ x_m^T \end{bmatrix} \begin{bmatrix} x_1 \dots x_m \end{bmatrix} = x_i^T x_j = G_{ij}. \tag{3.1}$$

   This matrix is crucial for the histogram comparison in the SVM, because G_{ij} is a measure of similarity between the i-th and j-th histogram, and x_i,x_j their respective bag-of-words representation. The minimum of x_i,x_j are summed up and saved in the result matrix.

3.5 Results

In this subsection are the results explained, what we achieved by testing several algorithms and what is better in contrast to the current state of the robot's detection system. Firstly, there must be an explanation, how the training set is set up. The data set is split up into 4 categories.

1. **dark_light:** The illumination is not high of the puppets face. Or the light source is behind the puppet.
2. **hole:** The doll is located within a whole.
3. **normal_light:** The doll's face is lighted perfectly.
4. **without_puppet:** A scene of the arena, where no doll can be seen.

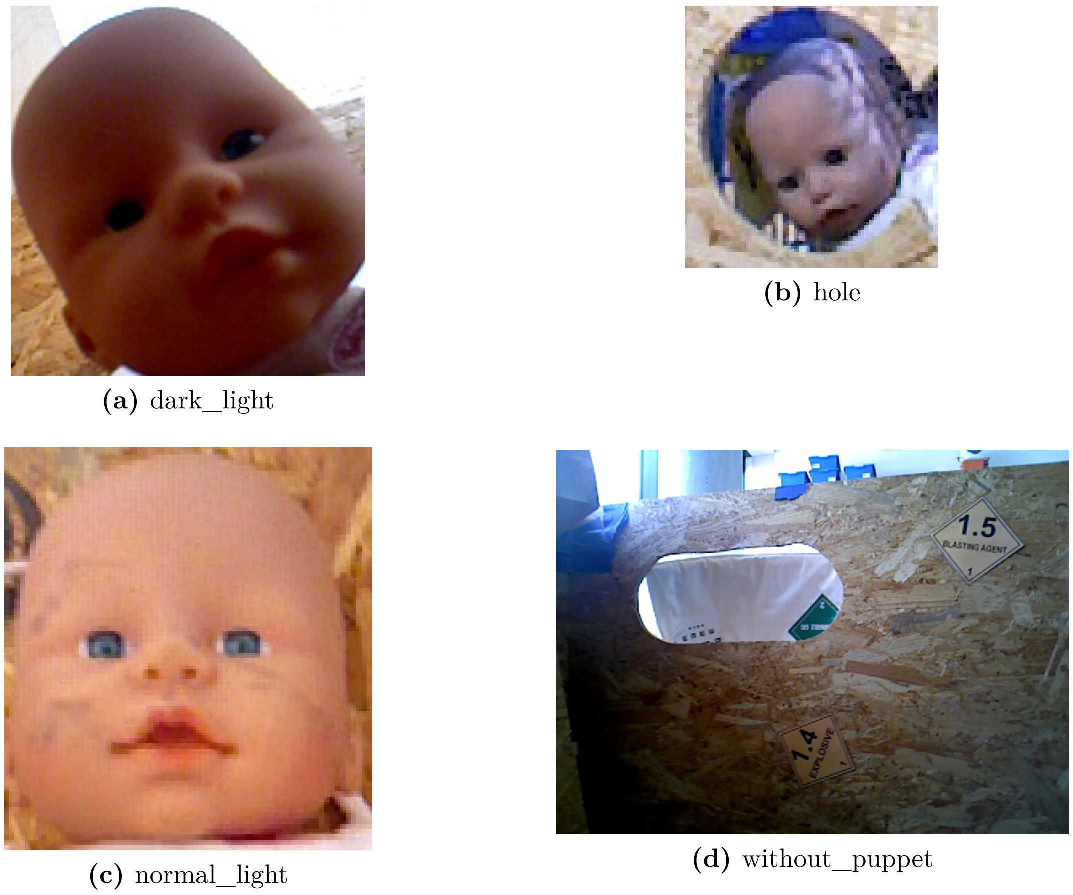
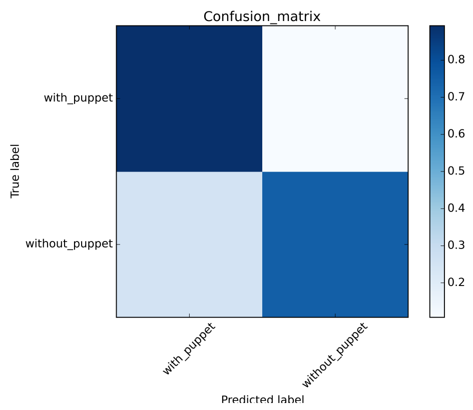
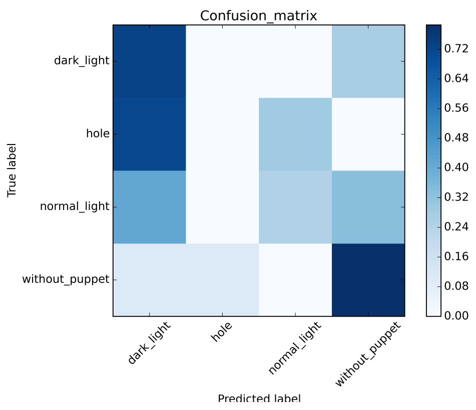

(a) dark_light (b) hole (c) normal_light (d) without_puppet

Figure 3.11: Training Set

The results are shown in the confusion matrices below. In the first confusion matrix 3.12a, there are only two categories, what is important for the robot's need. If there is an image with a doll or without a doll. The data set without_puppet is a combination of the *sets dark_light, hole* and *normal_light*, what can be seen in 3.12b. However, 3.12a yields an excellent result to distinguish between an image with and without a doll on it. The accuracy is about 78.57%, 44 of 56 test images are categorised to the right label. While the Haar Cascades results in around 52%-53% accuracy (in table Table 2.2), what is slightly better than guessing. Moreover, it can be saved time by using SIFT, because it is rotation invariant and there is no rotation needed as by Haar Cascades.



(a) 2 Categories



(b) 4 Categories

Figure 3.12: Confusion Matrix

In figure 3.12b, the confusion matrix illustrates a more detailed view of several categories. The goal is to compare different illumination aspects of the doll's face. The algorithm is not efficient in distinguishing between them. The category hole is very often mixed up with *dark_light* and *normal light* is often categorised to *dark_light* or *without_puppet*. Therefore, the accuracy falls down to 56.90%, 33 of 58 images are right categorised. Nevertheless, the matrix shows a strong distinction between with and without puppet.

4 Future Work

While this thesis demonstrated the advantages and disadvantages of the state-of-the-art of our robot's Computer Vision condition, there are still many approaches left for achieving better results in the end.

4.1 Aggregate Channel Features replaces Haarcascades Face Detection

The ACF[1] works directly on Intel or AMD CPU and therefore it is not very ressource consumptive. Therefore, ACF can run permantly for detecting features as the Haarcascades is currently running on the robot's system. Unfortunetely, the ACF by Piotr Dollar[2] is only a Matlab library and not provided in our computer languages, such as C++ or Python, which is suited for ROS, Robotic Operating System, which is the OS on the robot. There must be done some conversions from matlab code into a binary, which can be executed on the robots system. But the view of a software developer it would yield more obstacles than benefits. There are several reasons, why the decision is against the ACF, e.g. Software Maintenance: The library is written in Matlab. There may some future problems within various versions of Matlab. How will future Team Members debug a binary?

In future, there will be more possibilities to access the ACF in various computer languages.

4.2 AlexNet replaces CNN Overfeat

A deep learning algorithm is another approach to the CNN Overfeat. In this chaper, the AlexNet[AKH12] is proposed, recommend by the PhD Michael Opitz at the Institute for Computer Graphics. This CNN is known for it's accuracy since 2012 and in combination with the CAFFE[3] framework, it could be much faster than the current CNN Overfeat, which needs around 2 seconds per picture. In contrast, the AlexNet is a forward path network, which should be faster in general. The developer

[1] http://vision.ucsd.edu/~pdollar/toolbox/doc/
[2] https://github.com/pdollar/toolbox
[3] http://caffe.berkeleyvision.org

of CAFFE states on their webpage, it can handle 1 ms/image for inference and 4 ms/image for learning. Some problems are occured while starting to test it: The AlexNet needs to be trained for all possible kinds of objects in a scene, in order to keep down all kinds of False Positives, which is the current issue of the CNN Overfeat. The last layer of the AlexNet should be a SVM which recognizes the baby dolls.

Bibliography

[AKH12] A. Krizehevsky, I. S. ; Hinton, G.: ImageNet Classification with Deep Convolutional Neural Networks. (2012)

[BCV01] B. Choi, J. B. ; Veloso, M.: Fast Human Detection for Indoor Mobile Robots Using Depth Images. (2001)

[GDC09] G. De Cubber, G. M.: Human Victim Detection. (2009)

[JMP02] J. Matas, M. U. ; Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. (2002), S. R384 – R396

[Llo05] Lloyd, S.: Least squares quantization in pcm. (2005)

[MCS12] M. Correa, R. V. ; del Solar, J. R.: Humand Detection and Identification by Robots. Using Thermal and Visual Information in Domestic Environments. In: *Springer Science + Business Media B. V. 2011* (2012)

[Mei14] Meisel, A.: Feature based Sensor Fusion for Victim Detection in the Rescue Robotics Domain. (2014)

[SA11] Spinello, L. ; Arras, K.: People Detection in RGB-D Data. (2011)

[Scu09] Sculley, D.: Web-scale k-means clustering. (2009)

[SLS13] S. Leutenegger, M. C. ; Siegwart, R.: BRISK: Binary Robust Invariant Scalable Keypoints. (2013)

[TD05] und T. Darrell, K. G.: The pyramid match kernel: Discriminative classification with sets of image features. (2005)

[VJ01] Viola, P. ; Jones, M.: Robust Real-time Object Detection. In intl. Workshop on Statistical and Computational Theories of Vision. (2001)

Nomenklatur

ACF	Aggregate Channel Features
AUC	Area Under Curve
CNN	Convolutional Neural Network
HOD	Histogram of Orientated Depths
LUT	Look-Up Table
ms	mili seconds
ROC	Receiver Operating Characteristic
Wowbagger	The name of the roboter.