TECHICAL DOCUMENTATION

Version 0.1

How To Train Your Nao

CDIO HT 2015

Institute of Technology, Linköping University, ISY

Customer: Michael Felsberg, LiU (ISY)
Project supervisor: Fahad Khan, LiU (ISY)
Examiner: Michael Felsberg, LiU (ISY)

Participants of the group

Name	**Responsibility**	**E-mail**
Madeleine Stein	Project Leader	madst314@student.liu.se
Susanna Gladh	Quality Assurance	susgl621@student.liu.se
Anton Ågren	Testing	antag110@student.liu.se
Fredrik Kvillborn	Design	frekv134@student.liu.se
Elin Andersson	Scrum Master	elian253@student.liu.se
Richard Bondemark	Documentation	ricbo818@student.liu.se
Fredrik Löfgren	Nao Expert	frelo223@student.liu.se

Contents

Summary

This is the technical documentation document for the project *How to Train Your Nao*, carried out during the course TSBB11 2015 at Linköping University. The objective was to implement an object recognition algorithm to be used together with a Nao robot. The system applications include: object detection and localization in a scene, the Nao robot being able to search for and follow a known object, and the Nao robot being able to distinguish between multiple object. Two types of descriptors are used to create the object models: HOG-features in combination with Color Names and SURF. Both type of descriptors are trained using an SVM. In addition, OpenCV's built-in pre-trained people detector using HOG-descriptors are used to describe human objects. Two objects of different complexity were used to evaluate the former two descriptors. The detector uses a sliding window to detect the two objects and their location. The detection results were successful for both methods as long as a small enough step size for the sliding window was used. The HOG and Color Names descriptor had more successful results regarding detection of the object, but also had a larger number of false detections compared to the SURF descriptor. The task to be able to follow and search for an object was also successfully achieved.

1 Introduction

This document is a technical documentation for the project How To Train Your Nao, which is a part of the CDIO project course in Images and Graphics (TSBB11) at Linköping University during the fall semester 2015. The objective was to implement an object recognition system used together with a Nao robot.

1.1 About this document

The purpose of this document is to show which methods were used during implementation and why certain decisions were made during the project. Results and possible improvements of the system are also discussed for the event of future development.

1.2 Background

In a world where technology is constantly and rapidly evolving, robots are becoming an increasing part of our everyday life. Humanoid robots are a hot subject and can provide assistance in many areas, as well as companionship. Today's computer vision algorithms are able to identify and localise a variety objects. By integrating machine learning and object recognition with robots, a big step towards the interaction between humans and robots can be taken.

1.3 System application

The implemented software has several applications when used together with the Nao robot, seen in figure 1. With a pre-trained model the Nao robot is able to detect and identify objects in a scene. The system can also use multiple class models to distinguish different objects from each other. Other applications are searching for objects, following objects, and a combination of the two. In order for the robot to give feedback to the user the robot speaks when an object is found or lost.

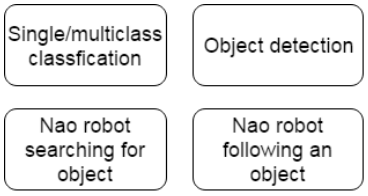

Figure 1: System application overview.

1.4 Brief system description

The system consists of one Nao robot, one computer and a wireless network connecting the two. The robot streams a video feed collected from its cameras to the computer, where image processing and decisions are computed. If the decision is interaction based, this decision is sent to the robot and acted upon.

The system can train object models using different descriptors. Both single and multiple class models can be trained by the system. When a trained model is obtained an object can be detected using the

detection mode. The detection mode can be used for detection of a single object class or multiple object classes at the same time. When the follow mode is applied the robot follows the object in the trained model. Further, search mode allows the robot to look for an object that is not in the field of view by turning its head. The search and follow mode can be combined. If the object is lost the robot will start looking for it by turning its head and continue walking in the direction where the object was last seen.

2 Hardware components

The core hardware component of the system is a Nao robot. The Nao robot is an autonomous, programmable humanoid robot developed by *Aldebaran Robotics*. The robot version used during the project was Nao V4 with body type H25. The robots sensors and actuators can be seen in table 1. The robot both collects images, detects sensory input and moves around in the environment. To gain maximum computation speed, all the software runs on an external computer, as opposed to running it directly on the robot.

Table 1: Nao Robot V4 H25 Hardware specification.

Cameras	2x
Ultrasonic sensors	2x
Inertial Unit	1x
Force sensitive resistors	4 on each foot
IR sensors	2x 940 nm
Microphone	4x
Speakers	2x
Wi-Fi	2.4GHz
Processor	ATOM Z530 1.6GHz CPU
RAM	1GB
Servos	25x

3 Software design

This section will describe the implementation of the software in the system. First an overview of the entire system and an explanation to different key concepts of the implementation will be given. Then the implementation of each subsystem will be described in more detail.

3.1 Software system overview

The system is built out of five different main systems: model training, object recognition, follow object, search for object and follow and search for object, each built out of different subsystems. Figure 2 below shows a flowchart of the system.

The system is mainly implemented in `C++` and uses two operating systems, ROS and NAOqi. ROS is a robot operating system using *packages* built out of several *nodes* for different processes, and allows *message* passing between them on so called *topics*. NAOqi is the software that runs and controls the Nao robot, which provides a Nao specific API. For the image processing specific operations, the OpenCV library is used.

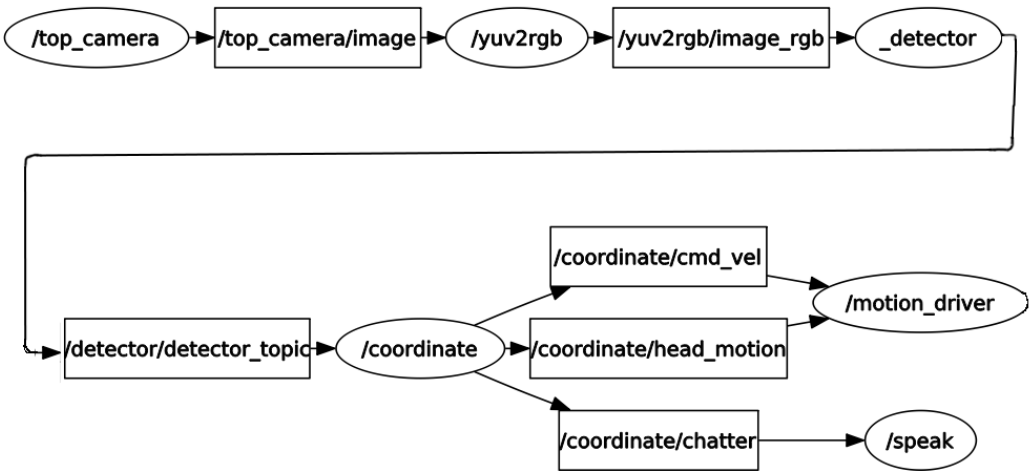

Figure 2: Overview of the system. The circles represent nodes, processes running in parallel, and the rectangles topics, which are channels for passing messages between nodes. The camera of the robot streams images in YUV, which are converted to RGB and picked up by the detector. Detected objects are published to the coordinate node, resulting in different actions depending on the chosen mode of the system.

3.2 Descriptors

The three different descriptors used are described below. For advanced object, humans, only the human detector of OpenCV is used. For description of any other object both of the other two descriptors are being used so that an evaluation of best fitted descriptor can be done.

3.2.1 HOG and Color Names descriptor

A combination of two types of features is used in the descriptor, *Color Names* and HOG (*Histogram of Oriented Gradients*). This should help discriminate an object using both color and shape information. Color Names is a color extractor which transforms an image from a three dimensional color space (RGB) into an 11 dimensional image space. Each dimension represent one basic color and the value of each pixel in each dimension describes the likelihood that the pixel is of the color corresponding to that dimension. The 11 possible colors are black, blue, brown, grey, green, orange, pink, purple, red, white and yellow. More about the Color Names descriptors can be found in a paper by Schmid et al[1] and an example of Color Names channels extracted from an image of a tomato can be seen in figure 3.

The second type of descriptor used for the descriptor is HOG features. As the name implies the HOG descriptor extracts the shape of objects using histograms of oriented gradients. The features are extracted by calculating the gradients pixel wise, dividing the image into smaller cells, and for each cell gathering the gradients belonging to that cell into a histogram. The histograms are normalized block wise. Each block is a grouping of a number of spatially connected cells and typically overlap, meaning that each cell will contribute to the total descriptor several times [2].

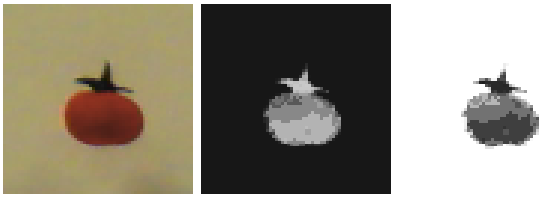

Figure 3: Example of two of the Color Names channels extracted from an image of a tomato (left) is seen in the two other images (middle and right).

3.2.2 SURF descriptors

SURF, *Speeded Up Robust Features*, finds interest points using the Hessian matrix and extracts features around the points by calculating the Haar-wavelet responses in a specific area around the interest points. The wavelet responses are summed up for each point and represent the features [3].

3.2.3 Human descriptor

To detect humans, OpenCV 2.4.11 built-in human detector is used, which is a pre-trained HOG descriptor. More on this can be found in the official OpenCV documentation [4].

3.3 Object learning

The training of the object models are implemented slightly different for the different types of descriptors. The features are extracted in different ways, but common for the HOG + Color Names and SURF-descriptor is the usage of an SVM classifier to train the object models. As mentioned previously, no training is needed for the human descriptor.

The same training data set was used for both the HOG + Color Names and SURF-descriptor. The initial training data was obtained from the robot with true images of the object to be modelled and

false images of background and other objects. Figure 4 shows an example of the data collection setup and resulting initial true training images. An example of false training data can be seen in figure 5. All training images are 64x64 pixels, the same size as the sliding window used during detection.

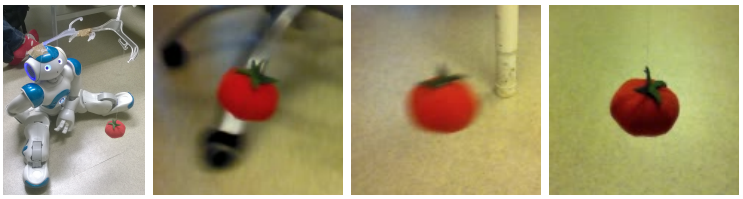

Figure 4: Example of data collection setup (far left) and the resulting images obtained by the robot camera.

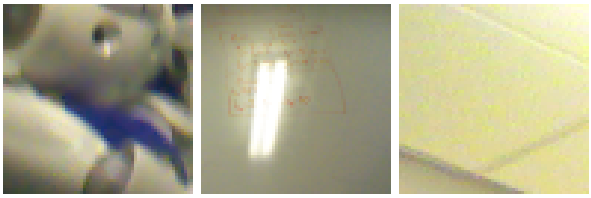

Figure 5: Examples of false training data used during object model training.

3.3.1 Train HOG and Color Names models

To train the HOG + Color Name descriptor around 200-300 true images, and more than 1000 false images are used. The model is trained using the SVM library *LibSVM*. The HOG features are extracted with OpenCV's built in HOG function. The image is split into 16 cells, each with 9 histogram bins. The image is also divided into 4 block, meaning that each block covers 4 cells, and the block stride is set to the same size as one cell.

Before extracting Color Names the images are down-sampled to 6x6 pixels using linear interpolation. The two descriptors are then concatenated to a long vector and sent to the SVM. The SVM uses a linear kernel. A cost parameter C is the only argument to the SVM. It is set by performing 5-fold cross validation on the whole training set with C varying from 0.01 to 1000, incremented by a factor 10 in each step.

To improve the model description, iterative re-training of the model is performed by adding false positive detections to the false training data, and false negatives to the true training data. When the trained object description model provides satisfying results it is saved to file, to be used in the detector mode.

3.3.2 Train SURF-models

For objects described by SURF-models interest points are extracted using OpenCV's built-in key point extractor. To calculate the features, OpenCV's built-in detector which calculates a 128 long descriptor for each key point is used.

The SURF descriptors are always the same size for each key point but the number of key point can differ a lot between images. This results in a varying total number of descriptors for different images. Since the number of key points depends on the image, training an SVM model on the descriptors directly will not work. Instead it is necessary to make a new descriptor that describes clusters of descriptors. To solve the problem OpenCV's *Bag Of Word* (BOW) is used when calculating the descriptors.

The BOW acts as a dictionary for the descriptors and assigns all the descriptors extracted from a single image to the entry closest in the BOW dictionary. An SVM model is then trained by first making a vocabulary for the BOW by clustering all descriptors extracted from the training images. Then, for each training image, the descriptors are extracted using the vocabulary. From the descriptors an SVM model can be trained by using OpenCV's built in SVM with labels for each different class and background images. The SVM and the BOW vocabulary are then saved to file, to be loaded in the detector mode.

3.4 Object detection

Detection is performed in the same manner for HOG and Color Names as for SURF-descriptors, but only one type of descriptor can be used at a time in the detection mode. The type of descriptor to be used is sent as an argument to the detector. For position estimation a sliding window with a step size of 16 pixels is applied in multiscale to the full size image. Each small window is analyzed for a object using either LibSVM (HOG + Color Names descriptors) or OpenCV's SVM (SURF descriptors). If an object is detected the position (center of the window) and size of the detection window is published. The human descriptor uses OpenCV's own SVM for easy multiple-scale detection and extraction of position and window size.

3.5 Multiple object classification

In the multiple object classification an SVM model is trained with two classes of objects and background images. The multiclass SVM uses the SURF descriptors described above with the same scaled sliding window. The SVM model will predict either of the objects or the background.

3.6 Follow mode

The following routine is a class running under the coordinator node. The coordinator node accepts asynchronous callbacks of detected objects and passes these on to the follower, which then tell the motion handler to move the robot. The movement is simple and slow due to an in general low update speed from the detector. Fast movement can easily make the robot lose the object from its field of vision.

For increased stability while following advanced objects, a Kalman-filter is used for both position and size of the object. The filter predicts the position and size if the object is not detected and keeps predicting for a specified number of frames.

3.7 Search mode

The search mode is a very simple routine where the robot moves the body and head more or less randomly while looking for objects. The search mode on its own is not very interesting, but in combination with the following routine the actions of the robot is more intuitive (follow, lose track then search for object). When an object is found or lost the robot speaks to inform the user how the detection is going. This is performed by the Text-To-Speech engine of NAOqi. The speech-node subscribes to messages from the coordinate node.

4 Testing

Two objects were chosen to be used to evaluate the descriptors, except the human descriptor. One *basic object* with simple shape and color and one object with a more complex structure and surface texture, denoted as a *normal object*. Both objects can be seen in figure 6. The evaluation of the descriptors was performed on two different data sets, one with a basic white background and one with a more advanced background and different lightning. Both environments were recorded from the camera on the robot, and examples of the test data sets can be seen in figure 7.

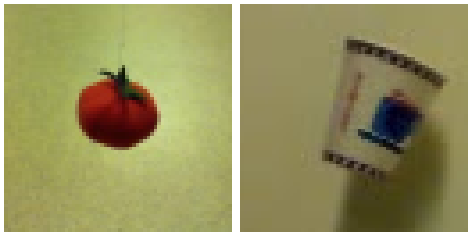

Figure 6: Images of the two objects used during testing. A fabric tomato as a basic object and a paper cup as a normal object.

Figure 7: Example frames from the two different test data sets. Left: Basic background. Right: more advanced background

To receive the results, every frame treated by the detector (some frames were skipped because the detector is not fast enough) was saved. Then, all true positives, false positives and false negatives were counted. The performance was observed by calculating a *hit-* and *error ratio*. The hit ratio was calculated as follows:

$$hit\,ratio = \frac{number\,of\,hits}{(hits + misses)} \tag{1}$$

Where *hits* represent the number of true positive detections, and *misses* represent false negatives.

The error ratio was calculated in the following manner:

$$error\,ratio = \frac{number\,of\,frames\,with\,false\,positives}{total\,number\,of\,frames} \tag{2}$$

A detection is considered a hit if part of the object lies within the detection box. The results can be seen in chapter 5.

5 Result

This chapter will show the result of the evaluation of the descriptors in the different environments tested according to chapter 4. The hit and error ratios were calculated according to formula 1 and 2 in chapter 4. Because many misses are correlated, which sometimes gives several false positives in the same area, only the number of frames containing false positives are counted, not the total number of false positives in each frame. The chapter is concluded with two graphs summarizing the results for all test cases (figure 12 and figure 13).

5.1 Detection of basic object

The object trained for detection of basic objects was a fabric tomato. The model was trained using HOG in combination with Color Names and with SURF-descriptors. The result of the detection in the basic environment can be seen in table 2, and the result of the detection in the more advanced environment can be seen in table 3. Figure 8 shows an example of the detection. It should be noted that the high error rate in table 3 is in part because of one bad object that was in the background during the test sequence.

Table 2: Result of the evaluation of the tomato model using HOG in combination with Color Names and the SURF-descriptor. The evaluation was performed by sampling frames from a video in a basic environment and compared to ground truth data.

Basic object in basic environment	**Hit ratio**	**Error ratio**	**Number of frames**
HOG and Color Names	100%	0%	50
SURF-descriptor	80.4%	0%	142

Table 3: Result of the evaluation of the tomato model using HOG in combination with Color Names and the SURF-descriptor. The evaluation was performed by sampling frames from a video in a more advanced environment and compared to ground truth data.

Basic object in advanced environment	**Hit ratio**	**Error ratio**	**Number of frames**
HOG and Color Names	100%	52.4%	82
SURF-descriptor	78.3%	34%	235

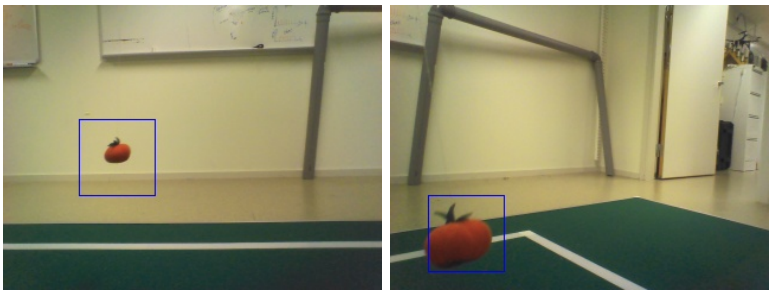

Figure 8: Examples of true positive detections of the basic object in a more advanced environment.

5.2 Detection of normal object

The object trained for normal objects was a paper cup. The model was trained using HOG in combination with Color Names and with SURF-descriptors. The result of the detection in the basic environment can be seen in table 4, and the result of the detection in the more advanced environment can be seen in table 5. Figure 9 show an example of the detection.

Table 4: Result of the evaluation of the cup model using HOG in combination with Color Names and the SURF-descriptor. The evaluation was performed by sampling frames from a video in a basic environment and compared to ground truth data.

Normal object in basic environment	**Hit ratio**	**Error ratio**	**Number of frames**
HOG and Color Names	88.0%	0%	161
SURF-descriptor	96.1%	0%	91

Table 5: Result of the evaluation of the cup model using HOG in combination with Color Names and the SURF-descriptor. The evaluation was performed by sampling frames from a video in a more advanced environment and compared to ground truth data.

Normal object in new environment	**Hit ratio**	**Error ratio**	**Number of frames**
HOG and Color Names	91.2%	47.3%	207
SURF-descriptor	69.1%	31.5%	111

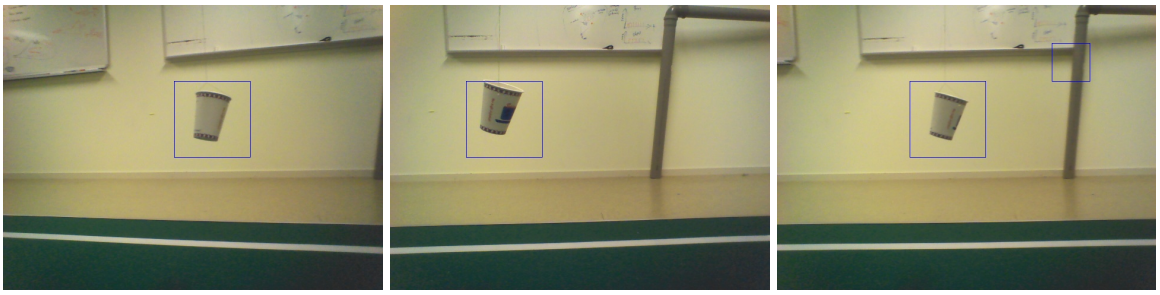

Figure 9: Example of two true positive detections (left and middle), and one example of a true and a false detection (right) of the normal object in an advanced environment.

5.3 Detection of humans

The results from running the pre-trained human HOG descriptor on the robot are presented below. The robot was in the mode "search and follow" during the test, which means it is walking towards the detected object, so some motion blur and tilted images is expected.

Table 6: Result of evaluation of OpenCV's pre-trained human object model using HOG. The evaluation was performed on 212 frames in a more advanced environment and compared to ground truth data.

Human object in advanced environment	**Hit ratio**	**Error ratio**	**Number of frames**
OpenCV pre-trained HOG	70.9%	11.3%	212

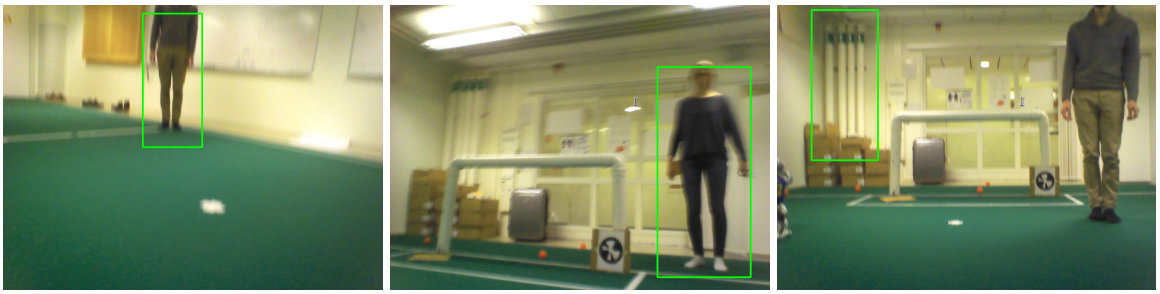

Figure 10: Example of two true positives of the human detection (left and middle), and one false positive and one miss (right).

5.4 Multiple object classification

The object chosen for the multiclass SVM model were the cup and the tomato. The model was trained using SURF-descriptors and tested on images containing the NIST arena building blocks as background. The result of the detection of the multiclass model can be seen in table 7. Figure 11 show an example of the detection.

Table 7: Result of detection test for a trained multiclass model using SURF-descriptors. The evaluation was performed by sampling frames from a video containing NIST arena building blocks and compared to ground truth data.

Tomato	**Hit ratio**	**Error ratio**	**Number of frames**
SURF-descriptor	83.9%	16.1 %	317
Cup	**Hit ratio**	**Error ratio**	**Number of frames**
SURF-descriptor	93.9 %	9.5 %	317

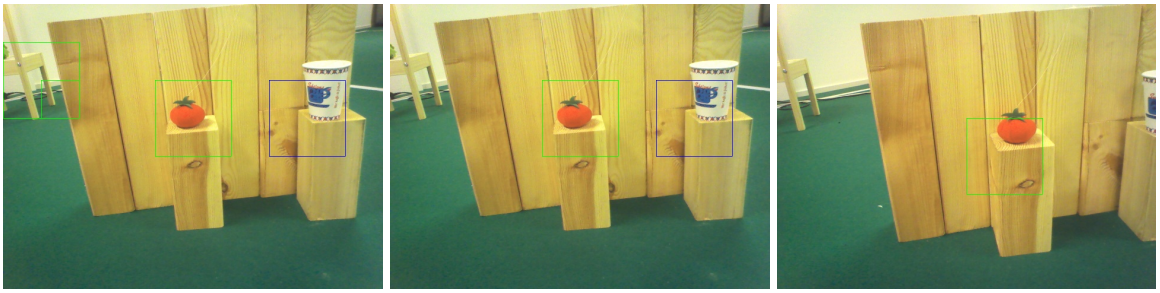

Figure 11: Examples of true positives for both tomato and cup, and one example of a false positive tomato detection (left), true positives for tomato and cup (center), and a true positive for only tomato (right).

5.5 Result overview

Figure 12 shows the hit rate for all the objects in all the environments where tests were performed. In figure 13 the error rate is shown for all these cases.

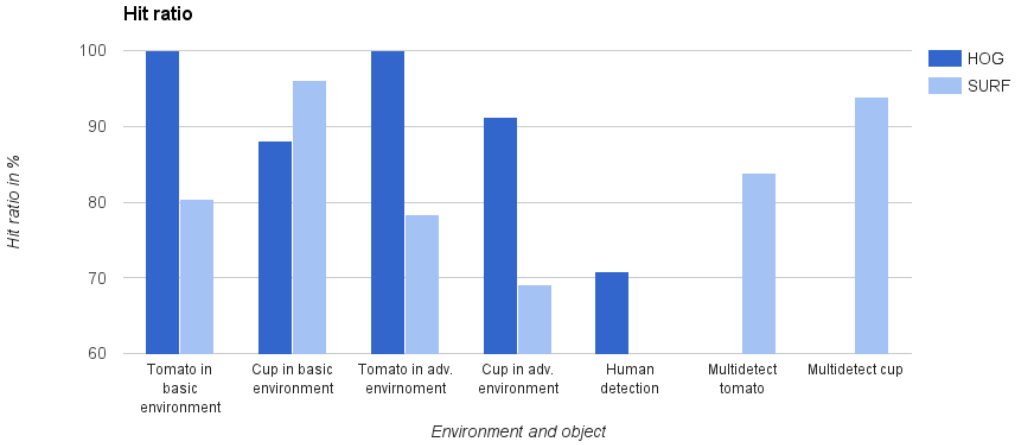

Figure 12: Hit rate according to equation 1 for each test set. Data is missing for SURF on human detection and HOG on multidetection.

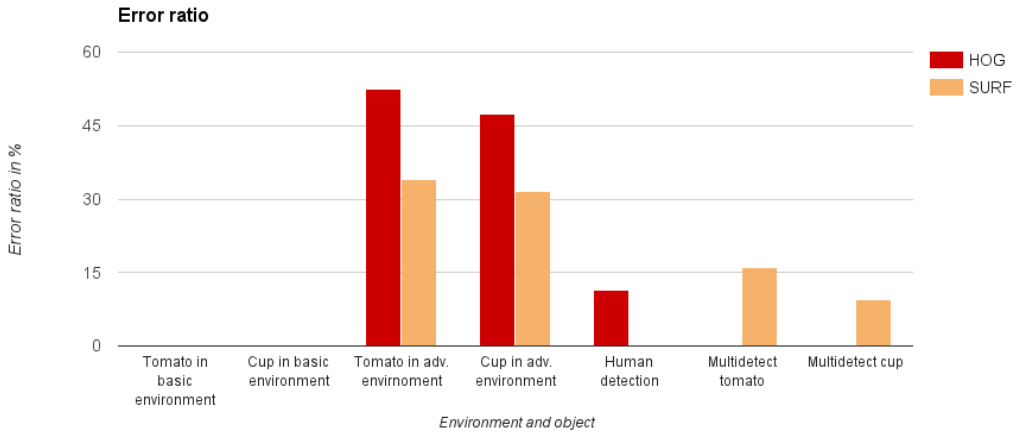

Figure 13: Error rate according to equation 2 for each test set. Data is missing for SURF on human detection and HOG on multidetection.

6 Discussion

This section will cover a discussion about the presented results and the system performance.

6.1 Basic descriptor

The most important part of the basic descriptor seems to be the HOG features, since the Color Names features apparently does not discriminate nearly as well as initially anticipated. It could be that it is required for the object to fill a larger area of each image for the Color Names to be effective. However, that would require smaller image patched to examined at a time, which in turn requires many more steps in our sliding window implementation and thus slows the system down. The HOG features on the other hand seems to work pretty well, even on objects with little structure, blurry objects and images of low resolution.

The speed of the system is another problem. An attempt to exchange LibSVM for LIBLINEAR was made, but that attempt ran into problems that could not be solved within the time frame of the project, and other tasks were prioritized. The hope is, however, that LIBLINEAR would speed up the system so that it can run in real time. LIBLINEAR could also run its operation on the GPU, which should greatly improve its performance. Not much work has been done experimenting on window sizes and their effect on the system speed. The step length of the windows greatly affect the speed however. The problem is that a bigger step size causes more misses since it increases the risk of jumping over a good detection.

A final big problem with the basic descriptor is that it has been a problem training the model to get rid of some false positive detections. This makes it hard to make the robot work well in a general environment. As mentioned above, if a smaller window size is used the Color Name descriptor might become better at filtering away these cases, at least if the object has a color that stands out and is rarely found in the background.

6.2 SURF-descriptor

Since the SURF interest points extractor uses the Hessian matrix, interest points are found where the gradient in the images is large. This greatly favours objects with a lot of patterns and those with color inhomogeneous surfaces. This is clearly noticeable when training with basic objects, which often are both patternless and color homogeneous. When training on these images the Hessian threshold is set to a low value, in order to increase the probability that interest points can be found on the objects.

Lowering the threshold have two major drawbacks. It will greatly increase the overall number of found interest points for all images. For each interest point its descriptor is calculated, which itself is computationally heavy. More descriptors per image also slows down the prediction when using the SVM. The second drawback is that the repeatability is decreased even further. The interest point extractor of the SURF descriptor always has problem with repeatability, even with the same image interest points, and may result in a great difference between extractions.

A low number of interest points, on the other hand, increases the sensitivity of the models. Predicting if an object is in the picture based on only one interest point will lead to many wrong classifications. This is especially true when the training data does not cover a particular case. Therefore a limit is set to only do predictions if the number of interest points in an image reaches a certain threshold. Keeping this threshold high, greatly removes false positives but also lowers the false negatives, especially in images where only a few interest points are found.

This makes the SURF model especially bad at detecting objects at a long distance. A very easy solution to handle this problem would be to increase the resolution. As higher resolutions greatly enhance the interest point extractions and will make SURF possible at much greater distances. This would of course also increase the computational complexity a lot, and would not be able to run in anything close to real time on the current set up.

With a reasonably high interest point threshold, as mentioned above, the SURF model will be restrictive. Combined with an object from which many interest points may be extracted, the SURF model will increase the accuracy compared to an object with less structure. To increase the accuracy even more the C-parameter in the model could also be fine-tuned for each data set to maximise the accuracy. In addition to this both a linear and radial based function for the SVM classifier, from which the radial based function was used in the end, based on the slightly higher accuracy and the speed was not noticeable affected.

6.3 Human detector and descriptor

The detection of persons is filtered so that only the largest object, hopefully a person, will be classified as detected. Therefore tracking of multiple objects is possible but not implemented.

The descriptor for detecting people is rather complex, since the body and limbs can look very different depending on positioning. The descriptor can therefore detect other objects that not even closely resembles a person. Since the advanced descriptor is pre-defined in OpenCV 2.4.11, the ability to train the descriptor for these false positives is very hard, but doable. One possible improvement could be not to use OpenCV's *DefaultPeopleDetector* but instead use *PeopleDetector48x96* or *PeopleDetector64x128*, depending on the approximated size of the person in the image. In general the human detector works rather well, but the low hit ratio (as seen in table 6) is one of the things

which would be preferable to increase through training.

6.4 Following objects

Following identified objects is not very simple on the Nao platform. When the robot is walking the body and head sway, causing the object to be more difficult to recognize. To increase the performance of tracking, a head rotation is used. However, the delay of the response in the system makes this hard to implement without blocking all actions until it is certain that the head has been moved and the new images has been received. This head movement could also be a problem for the Kalman filter.

6.5 Searching for objects

During the searching for objects the system currently has no idea how much the head has rotated when an object has been found. If the robot is set to "search and follow" the head will slowly reposition itself to looking forward, hoping that the follower will detect that the object is moving to the edge of the image and rotate the body accordingly. If the rotation angle of the head were to be known, the system could easily be less prone to lose track of the object by simply rotate its body and un-rotate the head until the body and head is facing the object.

6.6 Kalman filter

The Kalman-filter does a good job preventing the robot from losing track of objects. If multiple coordinates are sent in one frame, the Kalman-filter will track all coordinates with one filter. Therefore the filter only gives decent results if no objects are found or if the coordinates of the correct object is the ones being sent. All other cases can cause the prediction to jump back and forth and simply not track the right object. Another problem is that sometimes the detector tracks a false object for only one frame, which triggers the filter, and track nothing for a while. One improvement would be to implement one Kalman-filter for each object detected but that would likely make the movement confusing for the observers.

6.7 Multiple object classification

Multiple classification introduces some more difficulties compared to single classification. In this case any combination of objects is possible but the Hessian threshold and the amount of interest points needs to be the same. This will make it more difficult to get a higher overall accuracy and will make fine tuning more difficult. The same thing goes for the C-parameter for the SVM model which, can not be tuned for only one individual object.

6.8 Discarded software implementations

BRIEF is short for *Binary Robust Independent Elementary Features*, which is a binary descriptor containing binary strings. The strings are calculated by comparing the difference in intensity of pairs of points in the image. [5]

Object recognition can be successful using *Scale Invariant Feature Transform* (SIFT). The transform provides a description of the image in feature vectors, which are invariant of scaling, translation and rotation. This property is exploited when comparing features in two separate images to find matches

to find the same object in the two images. The feature vectors are obtained by passing the image through different filters, which extract interest points in the image. [6]

Both of these methods, together with SURF (see information about SURF under section 3.2.2) were investigated in the early stages of the project. SIFT was considerable slower then both SURF and BRIEF when calculating descriptors and comparing them, so SIFT was discarded in the early stages. SURF was ultimately chosen over BRIEF because some results in early tests pointed in favour of SURF. When the official testing started, much effort had already been put into a SURF implementation, so it was used and no final comparison with BRIEF was ever made.

References

[1] Schmid .C Verbeek .J Weijer, .J. *Learning Color Names from Real-World Images*. INRIA, LEAR, 2007.

[2] B. Dalal. N., Triggs. Histogram of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, USA, 2005. Cited 2015-12-14, available at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1467360>.

[3] Van Gool L. Bay H., Tuytelaars T. Surf: Speeded up robust features. *Katholieke Universiteit Leuven*, ETH Zurich 2011. Cited 2015-11-19, available at: <http://www.vision.ee.ethz.ch/~surf/eccv06.pdf>.

[4] Opencv 2.4.12 documentation. Cited 2015-12-24, available at: <http://docs.opencv.org/2.4/modules/gpu/doc/object_detection.html>.

[5] Strecha C. Fua P. Calonder M., Lepetit V. Binary robust independent elementary features. *CVLab, EPFL, Lausanne*, Switzerland 2011. Cited 2015-11-19, available at: <https://www.robots.ox.ac.uk/~vgg/rg/papers/CalonderLSF10.pdf>.

[6] D. G. Lowe. Object recognition from local scale-invariant features. *Computer Science Department, University of British Columbia*, Vancouver, Canada, 1999. Cited 2015-11-19, available at: <http://www.cs.ubc.ca/~lowe/papers/iccv99.pdf>.