RoboCup Project Report

Yu Yan, Ruochen Tang, Philipp Hermann,
Xueting Zhang, Jonathan Ackerschewski

01.April.2020

Abstract

Naoqi is a small humanoid robot designed to interact with people. It's packed with two HD cameras, four microphones, and ten tactile sensors.Naoqi can be programmed to respond to movement, speech, recognition and so on. By using two camera, speed-based and gesture-based teleoperation, Naoqi will reveal its capability of interaction. We recognize the number which is written on a piece of paper using Convolutional Neural Networks Model and point to it and speak it out using the API of Naoqi and calculating distance and angle in this project. Before recognizing number we use openCV and Contour detection to gain image data and recognize image.

1 General Target of the Project

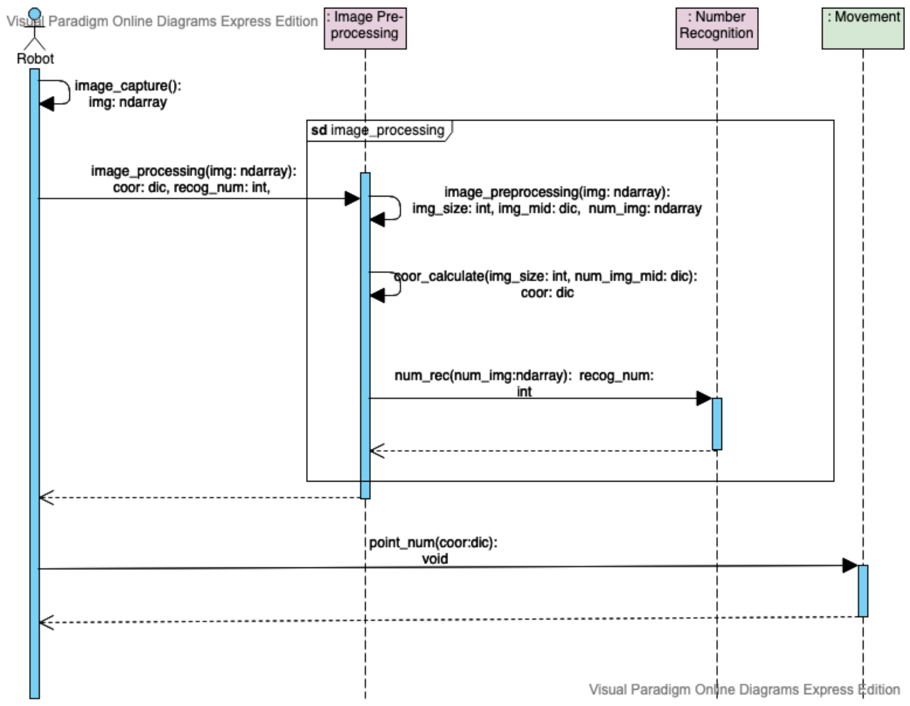
We seperate our task into two parts.The first part is detection including image processing and number detection.We hope that the robot can use its camera to gain image data and then recognize the number on a paper.The second part is Motion including coordinate calculation and moving.We hope that the robot can move its arm to point to the number based on the coordinate of Naoqi after transforming coordinate to the position of the number.

2 Methodology

2.1 Solution Design

Our solution is designed as follow:

First, the robot capture an image and save it as numpy array. Then, the image will be sent to our image pre-processing part. the paper which only have the number will be sliced and saved as a new image. The new image will be sent to our number recognition part which will recognize the number by using an CNN algorithm and send it back. the image pre-processing part will also calculate the the middle point coordination of the paper in the image and the size of the paper. Finally the movement part will get the coordination and control the robot point the number and say the number.



Visual Paradigm Online Diagrams Express Edition
Robot
: Image Pre-processing
: Number Recognition
: Movement
image_capture(): img: ndarray
sd image_processing
image_processing(img: ndarray): coor: dic, recog_num: int,
image_preprocessing(img: ndarray): img_size: int, img_mid: dic, num_img: ndarray
coor_calculate(img_size: int, num_img_mid: dic): coor: dic
num_rec(num_img:ndarray): recog_num: int
point_num(coor:dic): void
Visual Paradigm Online Diagrams Express Edition

Figure 1: Structure of Artificial Neural Network

2.2 Perception

In this project we plan to make a number assistant robot who can recognize the number written on a piece of paper then point to it and speak it out. The number to recognize should be in range of 0 to 9.

2.2.1 Image Pre-processing

The first part of detecting the number is to process the images in order to reduce noise and narrow the images to an essential area. The image pre-processing module takes an image and is supposed to find all papers on that image and return them cut and processed in a manner that can be used for the number recognition. Also for each paper the corner points, length of the edges and the center point of the paper has to be calculated. For the processing, the computer vision library *OpenCV* was used.

There are multiple ways of detecting a piece of paper on an image. The first approach was to not detect the piece of paper itself, but a marking on the paper. In this case a black circle was chosen. The circle was supposed to be detected using the *Hough Circle detection*. Without much processing of the image, this however returned really bad results and instead of refining this method a new attempt was made to process the image before detecting the paper with the intention to detect the whole paper on the image.

The result of the work was a processing pipeline, which used *Canny edge detection*, which creates a black image with some contours drawn in white. An example of that can be seen in figure 2. The full pipeline is explained later. From the processed image three approaches were made to detect the paper on the image.

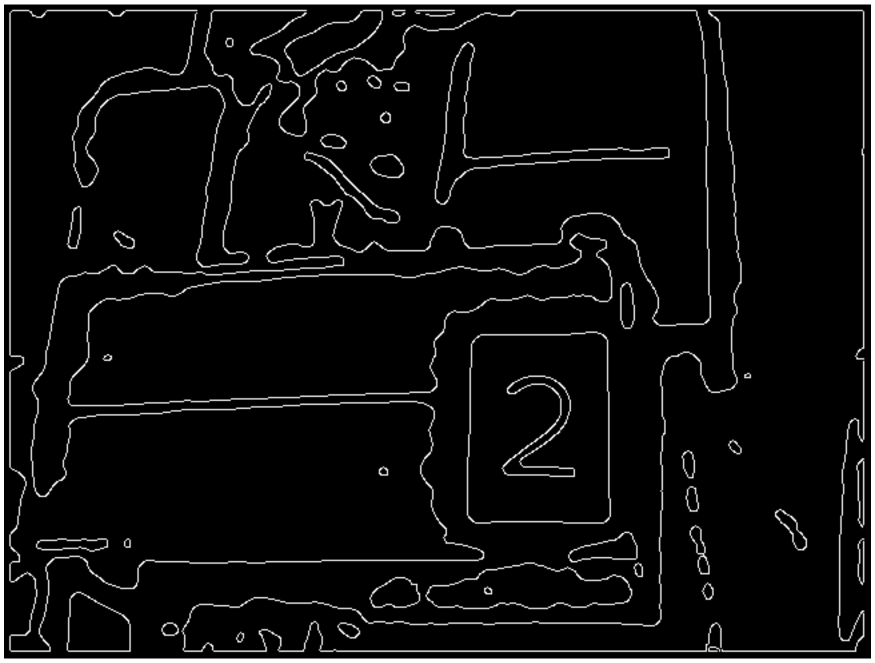


Figure 2: processed image with canny edge detection in test environment

The first attempt was done using *Hough Line detection.* Hough line detection matches pixels and tries to calculate lines given the pixels. Since the paper on the example image in figure 2 has straight lines, the thought was to detect those lines. This however showed bad results, since the edges of the paper were not actually detected, only some lines in the background. Figure 3 shows that in the test environment only one line was detected, which was not even a line of the paper.

The second approach was *Probabilistic Hough line detection.* In contrast to the Hough line detection, this approach also detects line segments on the image. The results were a lot better than using Hough line detection. By extending the detected lines and trying to find rectangles, this approach could possibly detect papers, however the final goal was to use the detection on the camera input of the robot and the results of the detection were changing strongly from frame to frame. So the result was an inconsistent detection.

The final approach was to use *Contour detection.* In contrast to the other

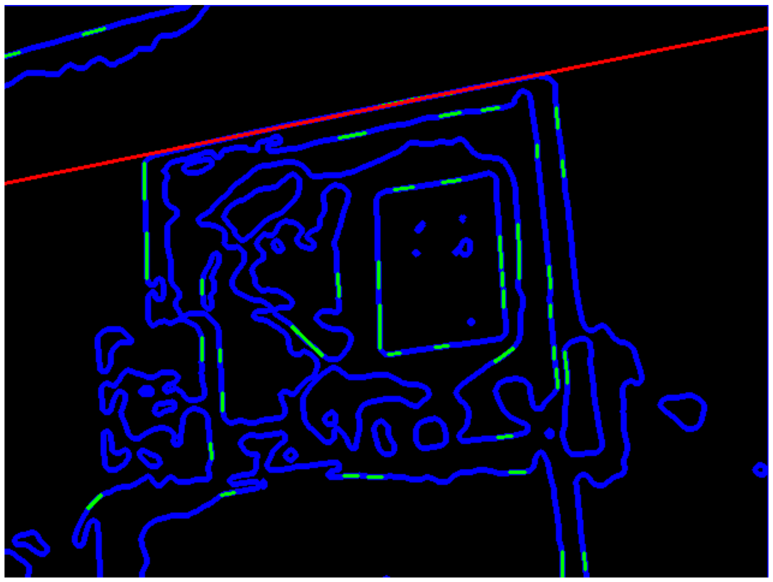


Figure 3: Example image showing Hough line detection (red), Probabilistic Hough line detection (green) and Contour detection (blue)

approaches, this version does not detect single lines, but detects whole objects by their contours. This seemed the most promising approach and therefore it was used and refined.

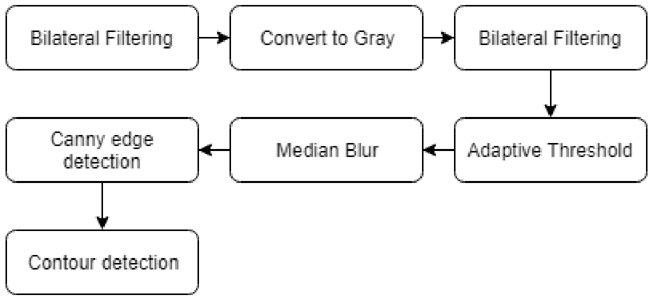


Figure 4: Pre-Processing stack for paper detection

After the approach was decided, the pre-processing pipeline was improved. The final processing stack is shown in figure 4. The first step is a bilateral filtering. Bilateral filtering blurs the image, but at the same time it preserves the edges, which are the interesting part of the image. This step is done before turning the paper to gray scale, because by turning it to gray scale a lot of information is lost. Using the filtering before the color change, will have better results if there is little contrast of the paper to the background.

The second step is to turn the paper into a gray scale image. After the conversion, another bilateral filtering is applied. This reduces the noise, but keeps the edges. The next step is an adaptive threshold. A threshold checks each pixel, if it is above a given threshold and turns it white if it is and black if it isn't. The adaptive variant calculates a threshold based on the image itself.

After that a median blur is done, which reduces the rest of the noise on the image. The last step is a canny edge detection, which creates the image as shown in figure 2. From there on the contour is detected and processed. For that each contour is abstracted to a shape with less points. The purpose of that is that the paper is a rectangle. Due to rotation of the paper in the room, the paper might not be a rectangle on the image, but a parallelogram. But also that isn't quite accurate, due to perspective, but the resulting shape should be close to a parallelogram.

The contour is abstracted to an easier shape. This shape is then tested if it is roughly a parallelogram, where the condition is dependant on the size of the found contour. Also the paper is supposed to be in DIN A4 format, so the aspect ratio of the sides should be about $1 : \sqrt{2}$. By testing those assumptions on each contour, the paper can be detected as shown in figure 5.

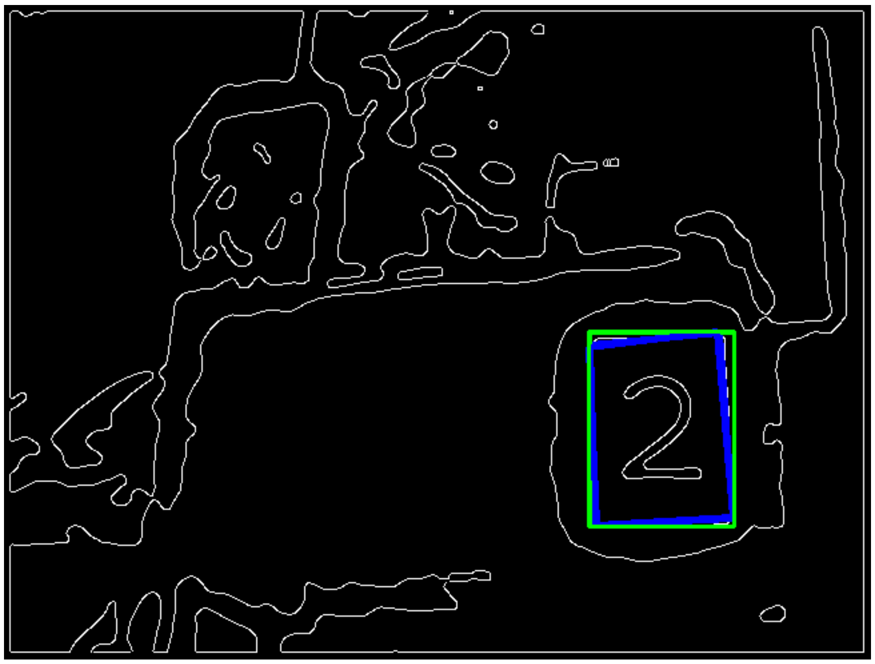


Figure 5: Detected paper on image. The paper is colored blue (parallelogram), the bounding box is colored green

After the paper is detected, it is cut from the original image and processed independently using a bilateral filtering and adaptive threshold. This creates a black image with a white number on it, where the thickness can be adjusted with the bilateral filtering parameters. Lastly background and edge noises are reduced for better accuracy of the number recognition. The corner points are given by the abstracted contour detection and the length and center point of the paper can therefore be easily calculated. The cut image can be seen in figure 6.

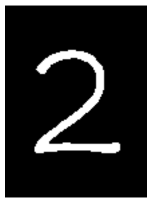


Figure 6: The cut paper from the image

2.2.2 Number Recognition

We use a CNN(convolutional neural network) to recognize the number. First, we build an initial CNN model. We chose MNIST dataset which is a huge database of handwritten digits as training set. Each image in MNIST dataset has a corresponding label which is the correct number in the image. We used cross entropy as the loss function. We can input an image in the model and get the predicted probability of all numbers from 0 to 9. Therefore, we chose the number which has the maximal probability as result. Then, we compare the predicted result and the label for each sample in the train data set in order to optimise the model. After our loss values are convergent. We observe the test results and adjust hyperparameters such as learning rate, epoch and batch to get the highest accuracy of the test data. Finally we saved the weights in a 'pt' file. When the robot begin to recognize a number, we create a class in type of our net and load weights, then it can directly predict the value.

2.3 Motion

As in the lecture demonstrated, we can use the coordinate transformation to calculate the position of the number based on the coordinate of the Naoqi itself. The angles can also be calculated so that we can set the arms to make Naoqi point to it. Meanwhile with the API of Naoqi we can let it speak the number out once the number was recognized.

2.3.1 Coordinate Calculating

First of all we don't know the angular of the paper in the room, so we assume it is rectangular to the distance of it's level and the camera and also it is adjusted such that the bottom line is parallel to the bottom picture line of the camera of the robot.

To calculate the coordinates of the paper we chose the following strategy. First we compute the distance of the paper. We can picture the scene as in figure 7 where Naoqi is in point A looking to the paper. b and c are the lines of the sight of the robotcamera. a is the line which connects b and c on the the level of the paper such that a, b, c create a triangle. The distance to the level of the paper is the height of the triangle to the side a.

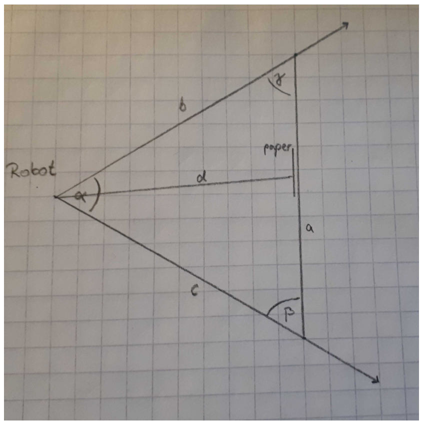


Figure 7: schematic overview

To compute the distance we first need to know the length of b. We are getting b, by computing a. a has a given number of pixel and so has the recorded paper. Due to the fact, that the paper is adjusted in the way it is, it always shows it's full length to the camera. The paper always has a fixed width of 210mm, so we can calculate it by crossmultiplication.

$$a/\#pixel\_of\_a\_in\_picture = paperwidth/\#pixel\_of\_paper\_width$$

$$a = paperwidth * \#pixel\_of\_a\_in\_picture/\#pixel\_of\_paper\_width$$

Since we know α due to the specs of the Naoqi robot (camera angle), we can now calculate b and c, because $\beta = \gamma$ due to the construction of the triangle being a isosceles triangle and $\alpha + \beta + \gamma = \pi$.
With the law of sines we can compute c as follows:

$$a/sin(\alpha) = c/sin(\gamma) = c/sin(\gamma)$$

$$c = a * sin(\gamma)/sin(\alpha)$$

$$c = a * sin((\pi - alpha)/2)/sin(\alpha)$$

At this point we can calculate the cut at which the actual distance cuts the angle alpha, because we get the paper_position from the picture in pixel.

$$\alpha' = paper\_position/\#pixel\_of\_a\_in\_picture * \alpha$$

Now we can get the new angle δ corresponding to the distance by computing the following:

$$\delta = \pi - \alpha' - \gamma$$

Which lets us calculate the distance of the level of the paper using the law of sines again.

$$d = c * sin(\delta)/sin(\gamma)$$

Now the calculated distance is still the distance of the level of the line a. To get a value closer to the right distance we compute the same again, but this time vertically with the height of the paper instead of the width and the vertically camera angle. The actual distance is then computed by:

$$distance = (d\_vertically + d\_horizontally)/2$$

Since we get the paper position in pixel from the captured picture we can calculate the distance from the paper to the center of the picture (in the level of the paper) as follows, with the camera angle in radians and the just calculated distance:

$$angle = (paper\_positions/screen\_size * camera\_angle) - (camera\_angle/2)$$

$$distance\_to\_center \ = sin(angle) * distance$$

Which leads to the point where we can return the distance from the camera to the level of the paper, the distance_to_centre vertically and horizontally which is a three-dimensional vector pointing to from the camera to the centre of the paper.

2.3.2 Moving

After the Cartesian's coordinate is calculated, it will be used in the *move.py* for the motion pointing to the number. According to the documentation of Nao, the left area regarding torso point has a positive value in y-axis while the right is negative. In this way the robot has to judge which hand to raise, by proving the y-axis value. Due to the different reference systems of the coordinates used in calculation and motion - in calculation the origin is where the camera at, whereas in the motion it shall be the torso point - a transformation is necessary. The function *run* in the class *run perception* accepts a transform matrix, which enables the transformation of coordinate from camera to torso. However, in the code it is simplified by using a unit matrix so that the result of the method is still regarding the camera. With the data from the documentation the coordinate can subtract the offset between torso and camera, which certainly causes a little deviation that can be ignored. The calculation of the angles can be deduced from the coordinate:

$$anglePitch = -arctan(\frac{z}{x}) \tag{1}$$

$$angleRoll = arctan(\frac{y}{x}) - arctan(\frac{15}{105}) \tag{2}$$

all the x,y,z here refer the torso point. According to the documentation the upper side of the arm has a negative value in angle, so the Pitch angle has a minus symbol. Due to the offset of the roll angle, it should be here adjusted. After the setting the angles the robot shall move the arm to the right point. With a loop in the code it can recognize and move continuously.

3 Discussion

In this section, individual parts will be evaluated as well as the overall system. Please note that the tests on the real robot were quite shallow, since the university grounds were closed due to the pandemic at the time. Still some tests could be run on the last day the robots were accessible, so there is some real data available.

Firstly the perception part is evaluated starting with the paper detection and going to the number recognition. The motion parts are hard to evaluate on their own, so instead the performance of the real robot is evaluated as a whole.

3.1 Paper Detection

The results of the paper detection are really hard to quantify, since it is heavily dependant of the environment, lighting, contrast, camera resolution, motion of the camera and more. Additionally a lot of parameters of the paper detection are customizable to fit the given environment. In order to show the limitations of the systems, a comparison of environments will be done.

The environment in which the paper detection was created and initially tested can be seen in figure 8. A paper in DIN A5 format is taped to a dark shelf and the number two is drawn onto it. The paper itself has a high contrast to the surroundings and even has a black border drawn at the edge to increase the contrast (it is not necessary though). The camera, which is used, is a Logitech C120 from around 2010. The quality is quite bad and noisy, which creates the necessity of a more robust detection. So using a better camera, like the robots camera, will result in even better accuracy. The resolution of the camera is 680x480 pixel. Also the paper isn't located in front of the camera, but a little bit to the top right and is also rotated slightly. It also doesn't sit flatly on the shelf and casts shadows on itself.

The result of the paper detection can be seen in the figure 5. The paper is detected properly and also cut perfectly from the image. The systems parameters were optimized for this scenario and work really well. Limitations of the system can be made clear by explaining how the system works in detail. The paper is detected as explained earlier by finding a contour, which can be simplified to a shape with four corners, which closely resembles a parallelogram. This means that the contour of the paper has to form a complete circle in order to even be considered for a paper. Any object, that is in front of the paper will therefore

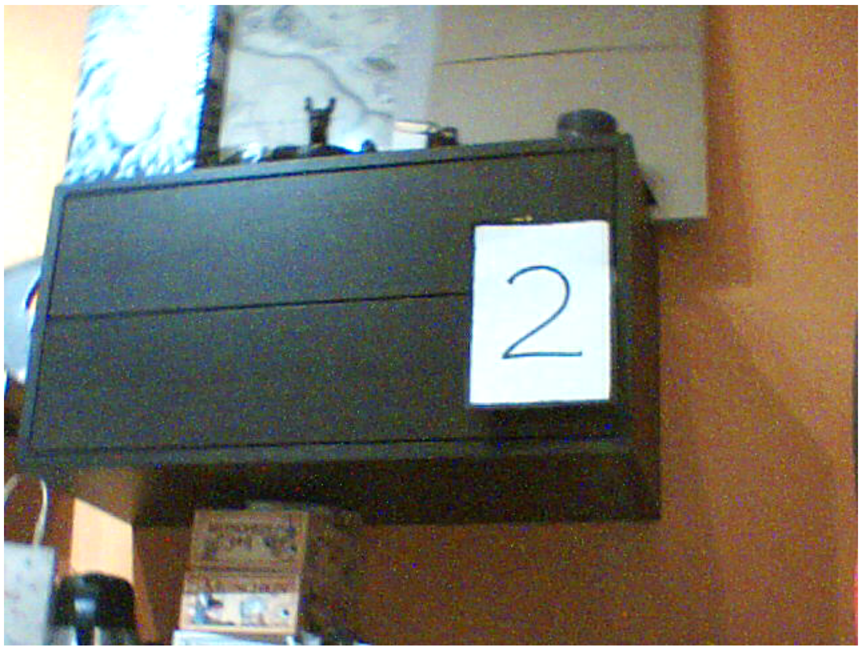


Figure 8: An example image of the environment in which the paper detection was created and tested.

make the paper not detectable. The same is true for heavy lighting differences, like the sun shining partially on the paper or something casting a strong shadow over the paper. Also no light should shine directly onto the paper. A passively illuminated paper, which is located on an object with different aspect ratio than paper in DIN format and of dark color works best to detect the paper.

The next example will show the usage on a scenario with light background. The image of this scenario was taken in the room, the robots are used in. The image can be seen in figure 9. It shows the test image, which was taken with a phone camera and the processed image with default parameters. Note that the paper has the number nine written on it as well as a circle. The circle was part of the original idea of detecting a circle instead of the whole paper.

It can be observed that the ground creates a lot of noise in the processed image. Also the paper is not completely detected, which results in the contour not being considered to be a paper. This scenario is an extreme case were the paper is located on a white wall, so the detection will be hard. In figure 10 another processed image can be seen. In this case the parameters were adjusted so that the paper can be detected.

It can be observed that there is a lot more noise, however the paper was successfully detected. It has to be mentioned that this is a border case. It is likely that the paper will be poorly detected when using the detection in real time with a video feed. This scenario is intended to show that a paper can be

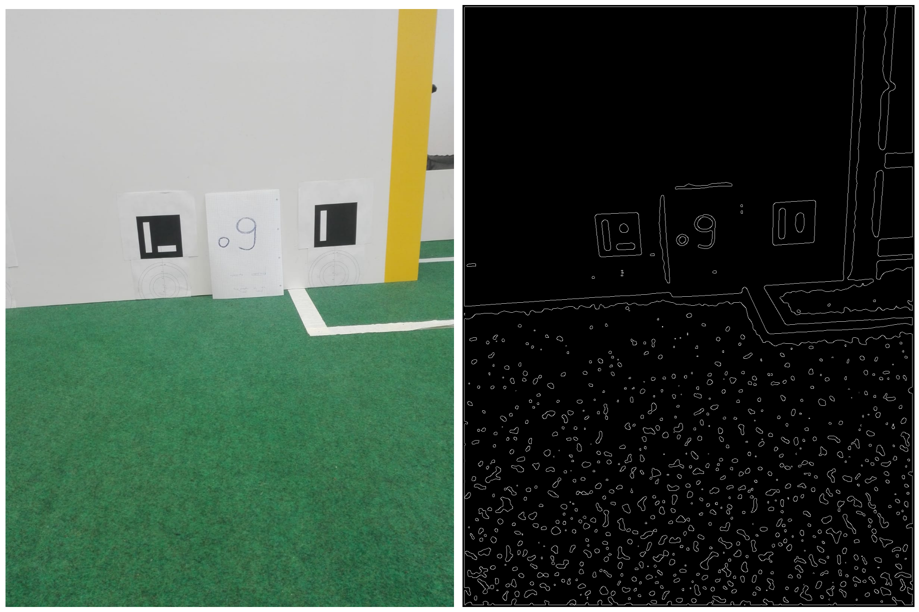


Figure 9: Robot environment taken with phone camera (left) and detected edges with default parameters (right)

detected however in quite some range of environments, when the parameters are adjusted correctly. For completions sake there is the cut paper, processed with default parameters and adjusted parameters in figure 11. There were some notes taken on the paper at the bottom. In the original image this is hard to see, however the processing of the cut image shows that clearly with default parameters. Also there is some noise at the edge of the paper. After adjusting the parameters, the notes at the bottom vanish as well as the noise, while the number is clearly visible in the center of the paper.

Lastly the performance on the real robot is explained in this part. The previous part already showed an image, which was taken in the same environment the robot is located in. Since there was not much testing on the real robot due to the pandemic, the parameters could not be adjusted. Instead a paper was held in front of a dark background, like the back of a chair. Using this procedure, the paper was detected relatively well. Still there was the problem of the sun shining into the room and sometimes shining onto the paper in parts.

In total the paper detection works well as long as there are no other objects with the aspect ratio as the paper (they will create false positives). Also the lighting should be passive, otherwise the paper might not be detected. When adjusting the parameters however, even border cases can be solved.

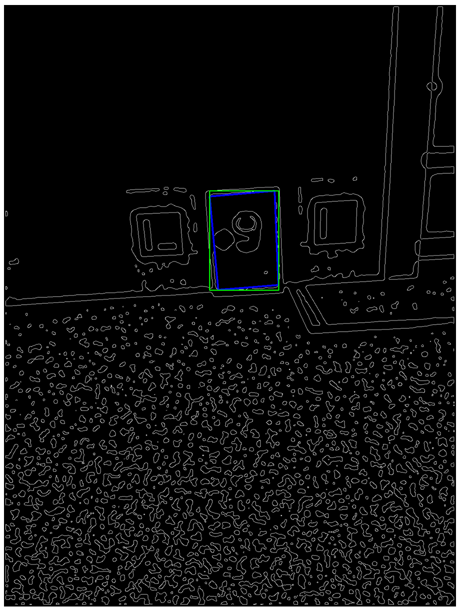


Figure 10: Robot environment taken with phone camera and processed with adjusted parameters

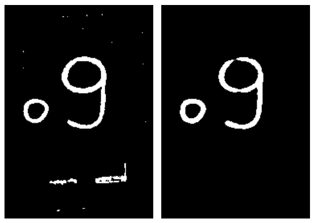


Figure 11: Robot environments cut paper processed with default parameters (left) and adjusted parameters (right)

3.2 Number Recognition

Evaluating our number recognition model, we should first observe the loss values of our train data. We calculated the average loss values for each epoch as figure 12:

Our loss values are already convergent. That means, we already have successfully trained the model. Then, we use our model to predict the numbers for each images in test set and compare the results and the corresponding labels. Finally, we obtained that our model predicted 98% images correctly. This accuracy is enough for the robot to recognize numbers.

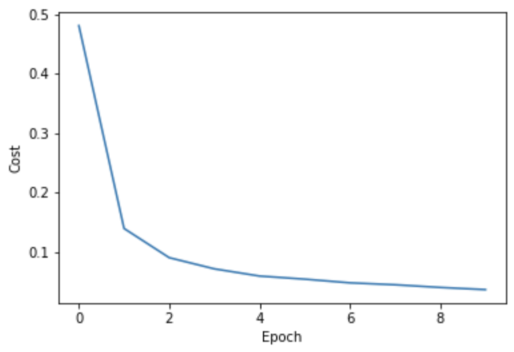


Figure 12: loss values of the training data set

3.3 Motion

As said before, the motion part cannot be tested on its own. Also both parts of the motion (coordinate calculation and arm movement) are purely mathematic calculations, which are either right or wrong. So instead this part will be mostly skipped and the overall performance of the robot will be evaluated. It will be said that the coordinate detection was fundamentally correct, however it was tested in python3 (as pytorch for the number recognition is only available for python3 on windows) and a problem occurred. In python3 the data types are automatically adjusted, when operations are done, where the result is a floating point (division of integers). This however is not the case in python2. The code was adjusted and the coordinate calculation was solved that way.

3.4 Real Robot

The performance of the real robot could not be tested thoroughly, but some results are available. When running the code, the main problem was the paper detection. At first the paper was put on the ground or held up from the back. A lack of contrast made the detection inconsistent however. Also the detection was not adjusted to detect papers, which have some space behind them to the next object (although there is a filtering option, which improves that aspect). After solving the problem by taping the paper to the back of a black chair, the next problem was the coordinate calculation, which was tested on python3 as stated above. Converting the code to use only floating point numbers solved the problem. The final result was that the robot took a picture a fixed number of times in a loop, detected the paper, recognized the number and pointed at the paper. The paper detection was quite well after taping the paper to the chair. The number recognition was bad at the time, since the paper had to be resized to 28x28 pixels, which made the number unrecognizable. A better resizing option fixed that issue later on. At the testing time however the robot detected the number eight instead of the number one.

After the coordinate calculation was fixed, the robot successfully pointed at the center of the paper. It used the left arm, when the paper was on his left and the right when it was on his right. For the robot to point correctly, the head has to be set to default position however, since the rotation is not included in the calculation and the labs were closed before it could be included.

Summing up the performance, the robot mostly detected the paper correctly, tried to say the number on it, but detected the wrong one at the time, but it pointed to the paper successfully, no matter where the paper was located, as long as it was in the vision of the camera and the head was rotated straight ahead.

4 Conclusion and Outlook

This project was a lot of fun and taught us interesting things about the Naoqi robot. At first everyone in our group thought the visual recognition of a number on a paper would be easy to implement, but it was a lot more work. In fact visual recognition is a big aspect in the RoboCup and for every team taking part in it. It starts by recognizing a number on a paper and can evolve to finding the ball, locating your team member robots and the opponents, finding a strategy based on it and then adapting to whatever the other robots will do. This topic can be utilized more and more the further one goes into this direction and with our project we got a good insight of how much preparatory work it takesto manage a football game with a team of robots.