DAI-Labor
Distributed Artificial Intelligence Laboratory

Nao Ellipse-Motivated Orientation Mapping

Leonard Hackel, Vito Mengers and Jasper v. Blanckenburg, *TU Berlin, Berlin, Germany*

I. Introduction

Using image detection to gather information about the environment is an essential part of robotics. However, computation power and cost of cameras both limit availability of data which leads to blind spots around a robot. Getting information about those spots would greatly increase the capabilities of robots. We propose an algorithm to extrapolate ball positions of *RoboCup* soccer games by evaluating head position of robots in our field of view. The head rotation detection is based on machine learning for head detection and ellipse detection for further calculations.

The report is structured as follows: In section II we will introduce the *Standard Platform League* as well as the *Nao* robots. These two will build the test platform for our implementation. In section IV the algorithm for calculation of head orientations will be presented[1]. Section V will evaluate the performance of our approach and section VI will conclude the work and suggest future work.

II. Nao Standard Platform League

The RoboCup Standard Platform League (SPL) is a number of different robot soccer leagues which present advances in robotics by applying the proposed solution to win a robot soccer game [1]. The game is a simplified version of regular soccer. It is played on a $6\,\mathrm{m} \times 9\,\mathrm{m}$ field with two teams of 5 Nao robots each [2].

The Nao robot is a robot developed by *Aldebaran*, a French company for humanoid robots. It is about $60\,\mathrm{cm}$ high and consists of multiple sensors and up to 25 motors. Every robot has two cameras on the forehead, one forward looking, one down looking, which are used for ball- and environment-detection in SPL games. We will use the top camera to find other robots and their viewing direction.

III. Camera Constraints

Since we use *TensorFlow*[3] for the head detection and the Nao robots are very limited in processing power, we have to transmit the images to an external computer. The service we use for this only transmits pictures in SD (640×480 px). It also does not have a particularly high dynamic range. The robot in fig. 1 is still fairly close (~$1\,\mathrm{m}$), but already shows the problem quite well.

The low contrast is obviously very problematic for the edge detection. The low resolution means that some edges might be adjacent in the image even though there should be a gap between them, which will break contour creation.

[1]The implementation can be found on GitHub: https://github.com/hackelle/robocup_project

Figure 1: Example picture: The head is only 85x57 px in size and the contrast in the left and top is very low.

IV. Ellipse-Motivated Orientation Mapping

The goal of our proposed algorithm (*Ellipse-Motivated Orientation Mapping*), is to determine the most likely current position of the ball by using the line of sight of other robots. Our algorithm consists of 4 main steps that are also shown in fig. 2:

1) We try to improve the quality of the image from the Nao camera through a preprocessing algorithm to simplify the other steps.
2) A Convolutional Neural Network (CNN) determines the position and approximate boundaries of Nao robots heads in the preprocessed image.
3) We extract contours from the image, then fit ellipses into them and filter out the eyes/ears.
4) Using the proportions of the eyes/ears, we determine the position and orientation of the robot and calculate its field of view.

A. Preprocessing

The preprocessing step consists of two main parts the use of *Contrast Limited Adaptive Histogram Equalization*[4] (CLAHE) and a standard median filter. At first we use the CLAHE algorithm to increase the contrast of the image without amplifying the noise in the image to much. This is done for two main reasons. Firstly the low contrast of most images that

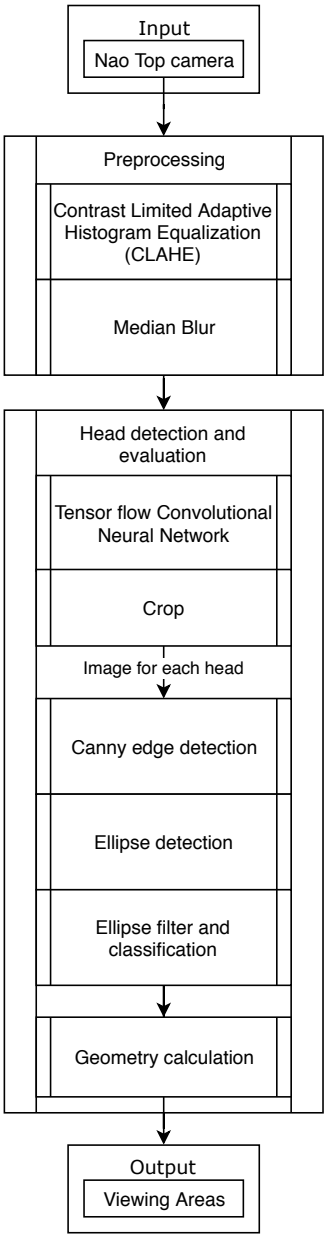

Figure 2: Data flow chart

the Nao camera provides and secondly the fact that we try to extract the orientation of the head by using the edges of the image, which are improved by an higher contrast. Afterwards we use the median filter with a small 3x3 filter to reduce strong noise while preserving the edges of the image.

B. Head detection

The CNN we are using to determine the position and area (boundary boxes) of Nao robot heads in the image is the SSD Mobilenet V1 Feature Extractor[5] of TensorFlow with additional transfer learning. The transfer learning was done with a labelled set of 593 images to train and test the network to find boundary boxes around the Nao robot heads in the image. We then crop the image to the boundary box expanded by a factor of 1.25 since the boundary boxes are not very accurate.

C. Ellipse detection and filtering

The cropped images are now all processed separately to determine the individual orientation of each head. In order to do that we extract the edges of the image using the well known Canny edge detection[6]. We then use OpenCV's `findContours`[7] function to split the edges into contours.

We then also combine close contours and add them to the contour list to counteract the problem of contour separation because of small missing parts of an edge. Using OpenCV's `fitEllipseDirect` we then fit an ellipse to each contour. This finds a lot of very bad ellipses however, since it blindly matches ellipses to contours. We also tried to use other ellipse fitting algorithms such as RANSAC[8] and Hough Transform[9], but they yielded even worse results[2].

Due to the fact that we create a large number of possible ellipses with this process we have to filter them for ellipses that could be an ear or an eye of the Nao robot. This ellipse filtering is done in three steps:

1) We remove all ellipses that could never be an ear or an eye of the robot due to their size, form, rotation or position in the cropped image.
2) We try to remove all ellipses that are not supported by a large enough amount of edge points as evidence. This way the number of ellipses can be reduced to only ellipses that are not only a good fit for their respective contour but a good fit to the whole image structure and therefore the ellipses are more likely to correspond to actual ellipse shaped parts of the image like the projection of the ear and eye of the Nao robot.
3) We detect strongly overlapping ellipses and only take the larger one of these overlapping ellipses. This is done because the ear and eye of the Nao robot are constructed in a way where there are multiple detectable ellipses in them and because due to possible separated edges multiple very similar ellipses can be created.

In fig. 4, the ellipses that are blindly fitted onto the contours are displayed in shades of red. The ellipses that survive the first step of filtering are displayed in cyan, the ellipses that survive the second step in yellow, and the ellipses that survive the final filtering in blue (ears) and green (eyes).

After the filtering of the ellipses they are classified in 0-1 ears and 0-2 eyes. We check the position of the eyes and ears in relation to each other to ensure a sensible configuration. That means that an eye can never be lower or higher than the ear and that the eyes can never be on different sides of the ear. Through this classification an hypothesis is created about the facial structure of the head.

[2]We expect they could yield far better results, but we did not have time to properly tune the model and parameters

Figure 3: Edges detected for a head

Figure 4: Ellipses in various stages of filtering based on the edges in fig. 3

D. Geometry calculation

The last part of our algorithm is calculating the orientation and position of each found robot in relation to our robot using the hypothesized facial structure and position of each head in the image.

First, we try to determine the distance of the robot to us using the height of the ear ellipse. Independent of the orientation of the head the height of the ear should always be the major axis of the ellipse and should only change according to the distance of the robot to us[3]. Therefore we determine the distance by using the equation 1, where f is the focal length of the camera in mm, h_{real} is the height of the real object in mm, h_{image} is the height of the image in pixels, h_{object} is the height of the object in the image in pixels and h_{sensor} is the height of the sensor of the camera in mm.

$$d = \frac{f \cdot h_{\text{real}} \cdot h_{\text{image}}}{h_{\text{object}} \cdot h_{\text{sensor}}} \tag{1}$$

With the distance we can determine the position of the robot in relation to us by calculating its radial angle. This is done by using the center of its boundary box on the x-axis of the image x_{head} and the field of view angle of our camera β_{fov} using equation 2.

$$\alpha_{\text{radial}} = x_{\text{head}} \frac{\beta_{\text{fov}}}{2} \tag{2}$$

With the distance d and the radial angle α_{radial} the position of the robot is known. Now its orientation and its resulting field of view have to be determined by using the projection of the circle-shaped ear of the robot as an ellipse using the calculations in appendix A. This gives us four possible angles. We determine the correct one by checking if the robot is facing us (i.e. we see its eyes) and what side of the head the ear is on.

If we do not see an ear, this calculation does not work. This may be due to two reasons: Our head detection reported a false positive, or the robot is looking straight at us or away from us. If it is looking straight at us, we should see both its eyes, which we can then determine its distance (using the distance between the eyes) and orientation (straight at us) from.

With our approach, it is impossible to discern between a robot looking away from us and a false positive. Thus, if we see no eyes and no ear, we do not use the robot for geometry calculation. However, if the robot were in fact looking away from us and facing the ball, we should also see the ball, so this is not a big problem.

After we determined position and orientation for each robot we know which area they are looking at. This area is defined by their horizontal field of view, which is $60.97°$, and their maximum viewing distance, which we assume to be $6\,\text{m}$, because of the resolution of the camera of the Nao robot and because of the field size of $6\,\text{m} \times 9\,\text{m}$ in the SPL.

If we detected only one robot the most likely position of the ball is in its whole viewing area. If we detected multiple robots the most likely position of the ball is inside the intersection of their viewing areas and less likely, but possible positions are in the other parts of their respective viewing areas.

Thus, we determined a hypothesis about likely ball positions using the knowledge of the other robots around us.

[3]We assume that robots do not tilt their heads very much.

	heads	facial structures	orientations
true positive	108	44	47
partially detected	-	31	-
false positive	6	19	30
TPR	64.7%	40.7%	43.5%
PPV	94.7%	69.8%	61.0%
FDR	5.3%	30.2%	39.0%

Table I: Results of the performance tests over 101 test images.

V. Performance test and Evaluation

The performance test is done with a set of 101 images, that were produced with the Nao robot and its camera. It consists of different scenes with 0 to 3 other Nao robots. As a metric for precision we use the terminology of confusion matrices:

	heads	facial structures	orientations
true positive	79	43	42
partially detected	-	30	-
false positive	6	7	19
TPR	60.3%	54.4%	53.2%
PPV	92.9%	86.0%	68.9%
FDR	7.1%	14.0%	31.1%

Table II: Results of the performance tests over 82 test images of robots with a distance of less than 2 meters.

The true positive rate (TPR) is defined as true positive results (TP) per possible correct positive result (P).

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}}$$

The precision (PPV) is defined as true positive result per positive result. Positive results are all true poitive and false positive (FP) results.

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

The false discovery rate (FDR) is defined as false positive result per positive result.

$$\mathrm{FDR} = \frac{\mathrm{FP}}{\mathrm{TP} + \mathrm{FP}} = 1 - \mathrm{PPV}$$

In these images are 167 robot heads of which we detected 108 with the convolutional neural network. Only 6 false positive results were produced, all of which were filtered out to not produce a false positive orientation. This results in a true positive rate of 64.7%. Precision and false discovery rate are 94.7% and 5.3% respectively. In these heads we extract 75 faces completely or partially. However, as partially detected facial structures often fail to find eyes or find only eyes where ears might have been detected, this leads to a high false positive count for orientations. Consequently the precision is only 61%.

If we reduce the distance between the robots the statistics improves. 82 images had robots with less than two meters distance to our evaluating robot. 131 heads were in these pictures. 79 heads were detected, which is a slightly lower precision. This might be because of the training set for the neural network which consists mostly of robots of greater distance. On the other hand most true positive facial structures and orientations were in a distance of less then two meters. Only one facial structure was correctly found in a distance of more than two meters. This results in a true positive rate of 53.2% for orientations. At about 69%, the precision of orientations is higher than for all test images combined. For facial structures the precision is 86%.

Since the precision is strongly decreasing due to wrong assignment of partially detected faces fewer partially detected face parts would help a lot. Most undetected face features are due undetected ellipses. As described in section IV-C OpenCV's ellipse detection is used on OpenCV's contour detection. However, a contour is found by merging adjacent pixels of an edge detection, which only works for high contrast. Correspondingly, a low resolution would merge different contours to a single contour. This leads to a merging of facial feature contours which would not happen on high resolution or high contrast images. Therefore, many partially detected faces would have been completely detected on higher resolution cameras, though the algorithm takes much longer to run. Additionally way more ellipses are found as the picture is much more detailed. However, this can be compensated for by tuning the parameters for ellipse, eye and ear detection. An example of this can be seen in fig. 9 and fig. 10.

One orientation mapping takes about $1\,\mathrm{s}$ on a modern laptop. This is mostly due to TensorFlow taking $0.9\,\mathrm{s}$ on CPU-only mode. Ellipse detection, filtering and orientation calculation takes less then 10% of the calculation time. Hough and RANSAC ellipse detection took more then $10\,\mathrm{s}$ each while not giving better results.

VI. Conclusion and future work

Our approach shows that it is possible to detect robots by evaluating their head positions based on ellipses, even in low-quality images. However, the overall accuracy heavily relies on the accuracy of the ellipse detection. Since OpenCV's ellipse detection is not very accurate, we only reach a precision of $61.0\,\%$ (or $68.9\,\%$ for robots within two meter's range) and a true positive rate of $43.5\,\%$ (or $53.2\,\%$ respectively).

Thus, the main area of future improvement for our approach is improving the ellipse detection. The simplest approach would be to increase the image resolution to the full 960p supported by the Nao's camera. Another approach would be using a more robust algorithm like RANSAC[4].

Currently, we simply calculate the most plausible orientation for each robot and output these. One could instead calculate multiple orientations and assign them probabilities based on the "badness" of the ellipses.

Another approach would be to first detect the ear and then, based on its angle, calculate where eyes should be and focus on those areas. Further, this could be combined with the probablities.

Finally, we only have a $64.7\,\%$ TPR in our CNN, so using a different feature extractor may well increase robustness.

Appendix

A. Angle calculation

We calculate the angle by assuming that the ellipse (ear) we found was a circle in the XY-plane that was rotated about the y axis and then projected into the XY-plane again. The projection should theoretically be a proper camera projection but for simplification purposes we assume a parallel projection.

We can discard the translation and rotation of our ellipse. Our ellipse is the projection of a circle that was rotated about the y axis into the XY-plane.

We should be able to calculate the rotation angle from just four points at the ends of the major and minor axes respectively.

[4]However, such algorithms require more processing power, especially if the resolution were to be increased.

A rotation about the y axis only changes the x coordinate (and z, but we cannot determine the z coordinates from the image):

$$
\begin{aligned}
x &= \cos\theta \cdot a \sin\alpha \\
x_1 = b &= \cos\theta \cdot a \sin\frac{\pi}{2} \\
b &= \cos\theta \cdot a \\
x_2 = -b &= \cos\theta \cdot a \sin\frac{3\pi}{2} = \cos\theta \cdot (-a) \\
\theta &= \arccos\frac{b}{a}
\end{aligned}
$$

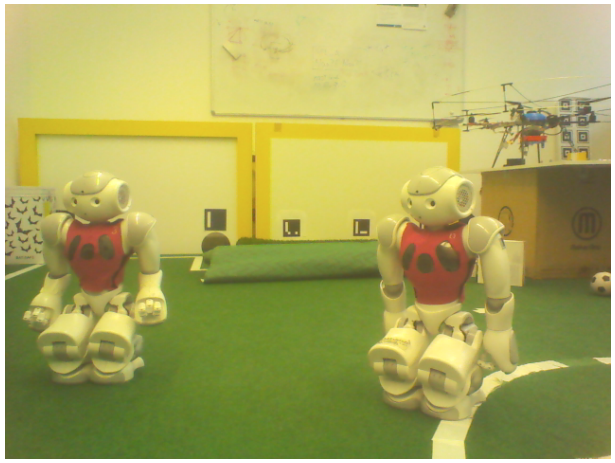

Figure 5: Example camera picture that yields two true positives

Figure 6: Edges that were detected for a head in fig. 5

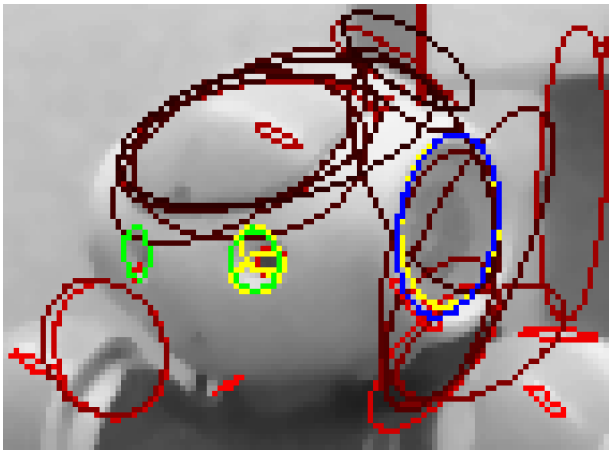

Figure 7: Ellipses detected from fig. 8. The blue ellipse is the ear, the green ellipses are the eyes.

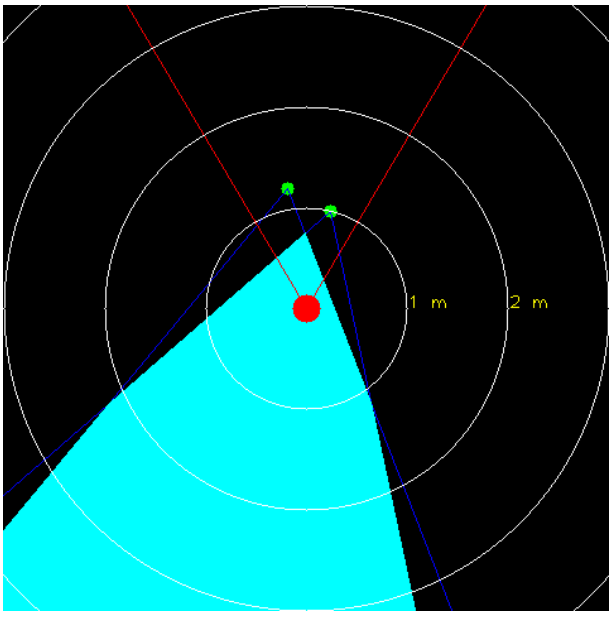

Figure 8: The calculated geometry from fig. 7. Our robot's field of view (FOV) are the red lines, the other robots FOV are blue lines and the overlapping part of the FOVs is cyan.

REFERENCES

[1] *Robocup Homepage*. http://spl.robocup.org/, . – visited: 2018-12-11

[2] *Robotlab NAO Homepage*. https://www.robotlab.com/store/nao-power-v6-standard-edition, . – visited: 2018-12-11

[3] *Tensorflow Homepage*. https://www.tensorflow.org/, . – visited: 2018-12-13

[4] PIZER, Stephen M. ; AMBURN, E P. ; AUSTIN, John D. ; CROMARTIE, Robert ; GESELOWITZ, Ari ; GREER, Trey ; HAAR ROMENY, Bart ter ; ZIMMERMAN, John B. ; ZUIDERVELD, Karel: Adaptive histogram equalization and its variations. In: *Computer vision, graphics, and*

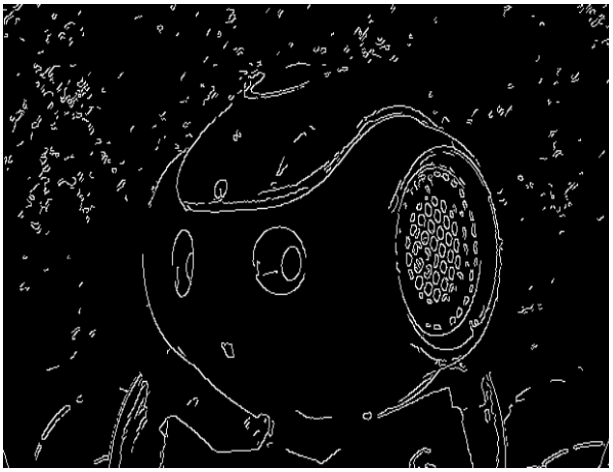

Figure 9: High resolution image of the edges of an robot head. Same perspective as in fig. 10.

Figure 10: low resolution image of the edges of an robot head. Same perspective as in fig. 9.

image processing 39 (1987), Nr. 3, S. 355–368

[5] *ssd_mobilenet_v1_feature_extractor.py*. https://github.com/tensorflow/models/blob/master/research/object_detection/models/ssd_mobilenet_v1_feature_extractor.py, . – visited: 2018-12-12

[6] CANNY, John: A computational approach to edge detection. In: *IEEE Transactions on pattern analysis and machine intelligence* (1986), Nr. 6, S. 679–698

[7] *Structural Analysis and Shape Descriptors*. https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html, . – visited: 2018-12-17

[8] FISCHLER, Martin A. ; BOLLES, Robert C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In: *Commun. ACM* 24 (1981), 6, Nr. 6, 381–395. http://dx.doi.org/10.1145/358669.358692. – DOI 10.1145/358669.358692. – ISSN 0001–0782

[9] BALLARD, Dana H.: Generalizing the Hough transform to detect arbitrary shapes. In: *Pattern recognition* 13 (1981), Nr. 2, S. 111–122