Real-Time Line and Center Circle Detection for NAO Robots in RoboCup Matches

Marc-Olivier Bisson, Olivier St-Pierre, Luc Duong[0000−0003−3118−5389], and Camille Coti[0000−0002−1224−7786]

École de technologie supérieure, Montréal, Canada
olivier.st-pierre.4@ens-etsmtl.ca

Abstract. This paper introduces a novel real-time approach for detecting painted field lines and the center circle in RoboCup matches using NAO robots. The proposed method addresses the challenges posed by outdoor environments—where lighting conditions vary rapidly due to natural light and field lines fade over time—as well as the transition from high-contrast adhesive tape to low-contrast painted lines. The detection framework leverages a series of OpenCV functions, including adaptive thresholding, Gaussian blurring, and a custom skeletization process to refine thick line segments into their medial axes. These preprocessed images then serve as input to a probabilistic Hough Transform, which efficiently identifies line segments that are later grouped and merged to reconstruct complete field lines. For center circle detection, the approach uniquely treats detected lines as tangents, generating candidate circle centers that are clustered to pinpoint the circle's location. Evaluation against the 2021 B-Human detector using metrics such as precision, recall, and F-score demonstrates robust performance under variable lighting conditions with minimal manual calibration, despite increased processing times due to intensive OpenCV operations. Limitations include occasional false positives and performance constraints, with future work focusing on further optimization and adaptive image resizing to enhance detection of distant lines.

Keywords: Line detection · center circle · Hough transform · RoboCup · real-time · NAO robot.

1 Introduction

The RoboCup has set an ambitious goal: to defeat the human world champions in soccer by 2050. With only 25 years remaining to achieve this objective, it is crucial to bridge the gap between robotic and human soccer.

In this context, the RoboCup Standard Platform League has been increasingly emphasizing natural lighting environments. Over the past two RoboCup competitions, matches have been played on "outdoor" fields. By "outdoor", we refer to fields with nearby windows, leading to lighting conditions that can change within seconds due to cloud movements or simply depending on whether the robot is facing the window or has its back to it.

In addition, since 2022, field lines have been painted instead of using adhesive tape. As a result, robots now compete in significantly less controlled environments than before. Adhesive tape lines provided much higher contrast compared to painted lines. Furthermore, painted lines fade over the course of the competition, making their detection even more challenging as the competition goes.

To form a competitive soccer team, robots must accurately determine their position to cooperate effectively. A single lost robot can affect the entire team, either by kicking the ball toward the wrong goal or simply by going off the field and thus being penalized for a certain time. Field lines are a key element for robot localization on the field, and thus a robust line detector is at the core of a robot soccer game.

In this paper, we present a novel approach for detecting painted field lines using the NAO robot. Our approach aims to improve robustness against lighting variations, reduce the calibration time required before matches, and ensure reliable detection of painted lines throughout the competition.

We will discuss related work in Section 2, followed by a detailed explanation of our line detection method in Section 3. Finally, in Section 4, we compare our detector to the 2021 B-Human detector, and we conclude in Section 5.

2 Related works

Naova has been using B-Human's code as its base since entering the RoboCup in 2018. For this reason, we are familiar with B-Human's line detector [8]. In their 2021 code, which serves as the base for Naova since 2024, line detection relies on vertical and horizontal scans of the image to identify white pixels. Linear regressions are then applied to these points to identify lines. This solution works well for lines made with adhesive tape; however, since 2022, the SPL has been using painted lines. This significantly affects the performance of their line detector without extend calibration before each game. With the introduction of outdoor fields, manual camera calibration became increasingly difficult. The windows are much brighter compared to the field, making it challenging to achieve good image quality in this type of environment. This caused the line detection to not work very well in this environment.

To detect the center circle of the field, B-Human uses the points that compose the lines to compute circular arcs. From these arcs, they can approximate the center of the corresponding circle. By grouping these centers together if they are sufficiently close to each other, it is possible to detect the circle. This occurs when enough centers align consistently. This approach is quite robust and avoids false positives while also being time-efficient.

Unlike B-Human's approach, which detects lines across the entire image, other teams divide the image into smaller segments. This is the case for the rUNSWift team [2]. The first step of their line detector is to identify regions of interest in the image. This is possible because all relevant elements in the image are white. Once the regions are isolated, they can apply algorithms to a small subset of pixels, reducing the computational cost. For example, to detect a line,

they search for white pixels along two edges of the region. If the white pixels are close enough to match the expected width of a line, the region is considered a line.

To detect circular arcs, they use a strategy similar to B-Human's. After identifying a region as a line, they check the color of the pixel located at an equal distance between the two identified edges. If this pixel is white, it indicates a straight line; however, if it is a different color—most often green—the region is considered a circular segment. By combining the detected circular arcs, they search for a circle that meets several criteria.

The problem addressed by [5] is similar to our problem. With the introduction of artificial grass in the humanoid league, the visibility of field lines decreased, making color segmentation ineffective for their detection. To overcome this issue, the authors apply a Canny edge detector to the V channel of the HSV color space. Then, they use a probabilistic Hough line detector [3] to identify lines from the detected edges. Since edge detection typically produces strong responses on both sides of the lines, a merging algorithm is used to group similar lines together.

In [7], the authors outline common image pre-processing methods for lane detection on roads. They emphasize the importance of enhancing image contrast, reducing noise, and applying edge extraction techniques. These algorithms will be beneficial for our line detector, improving the results obtained with the Hough transform.

In [9], the author presents a center circle detection method for the NAO robot, structured in three stages: arc extraction, ellipse detection, and post-processing. A modified Canny edge detector feeds an arc detection algorithm, which identifies arcs to reconstruct an ellipse. Then a clustering and validation step filters false positives. While the approach aimed for real-time execution on the NAO, this goal was not achieved.

3 Algorithms

3.1 Overview of the line detection process

Our line detector relies mainly on functions available in OpenCV. The key algorithm is the Hough algorithm that finds line segments in the image. For Hough to work well, we need to give it the best input possible. This means that we need to do preprocessing beforehand.

The input of the line detector is a grayscale image. A Gaussian blur is applied to the image to reduce the noise in it. Afterwards, an adaptative threshold is used to convert the grayscale image into a binary one to keep only the whitest pixel of the image. An absolute threshold would keep all pixels above a certain value, but the adaptative threshold calculates a different threshold for each pixel based on its neighbors. The adaptative threshold works better than an absolute threshold in this case because of the different levels of light in the image. This targets directly one of our goals to be resilient to light changes introduced in the last few years in RoboCup games. Adaptative threshold is more robust to

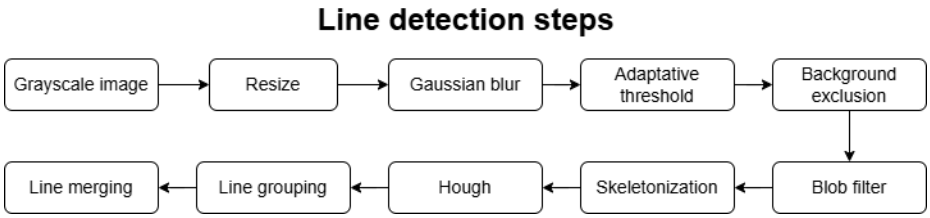

Fig. 1. Line detection steps

different lighting conditions and reduces the hard-coded parameters, which was another challenge we wanted to address.

After this step, we get a binary image of the field where the white pixels are mostly the lines since they are the most white parts of the field.

The next step is the background exclusion. The field border detection of B-Human [6] is used to exclude anything outside the field. By excluding, we mean that any pixel above the field border is set to black. This reduces the false positive that would otherwise be generated. We also filter blobs by area to keep only the eigthy largest one for the upper camera and the eigth largest one for the lower camera. The goal is to reduce potential noise on the field.

The resulting image needs a last transformation before applying Hough to it. Hough works on the pixel level and the lines at this point are tens of pixels wide. If we were to apply Hough at this point, the result would be incorrect.

We tried using a Canny filter [4] to extract the lines from the binary image, but since Canny detects the edges in the image, it detects both sides of the thick lines. The Hough transform would then detect these two lines, which is not the result we want.

To correct this problem, we use a skeletization process to find the center of the lines [10]. This function does not exist natively in OpenCV. Skeletization is an iterative algorithm that uses mainly erodation and dilatation operations to reduce the image into median lines.

After the skeletization, the binary image contains only the median line from the original thick line. This is the appropriate input for Hough to find lines.

3.2 Hough algorithm

Researchers have optimized the Hough algorithm in many different ways over the years. For our application, we were looking for a time-efficient algorithm, as it was likely to be the most time-consuming step on the robot. OpenCV offers two different Hough algorithms. We chose the fastest one, the probabilistic Hough transform developped by Galambos et al. [3]. This is also the same method used by Farazi et al. [5] in their line detection.

3.3 Line grouping and merging

The Hough transform detects a large number of lines, many of which correspond to segments of actual field lines. For accurate localization, we aim to detect complete lines while minimizing duplicate detections. To achieve this, detected lines are grouped based on multiple criteria with the goal of combining them.

The first criterion ensures that the two lines have similar angles. If their angles fall within a predefined threshold, we proceed to the next criterion. The second criterion checks whether at least two endpoints of the lines are within a certain distance of each other. If this condition is met, the lines are grouped together.

However, an edge case remains: a shorter line segment may be entirely contained within a longer one. While the angle criterion would be satisfied, the distance criterion might not. To address this, we introduce a third criterion, which serves as an alternative to the second. If an endpoint of the shorter line is sufficiently close to the axis of the longer line, the two are considered to be part of the same group.

Once all lines are grouped, we determine the two most distant endpoints, regardless of which lines they originally belonged to. These endpoints form the line that represents the group. As a result, the final detected line may be composed of endpoints from different segments, this is, in fact, the most common outcome.

These steps improve the precision of our line detection. The resulting lines are not only closer to the actual field lines, but also fewer in number, reducing redundancy and enhancing localization accuracy.

3.4 Line exclusion

Our line detector performs well overall, but certain cases still result in false detections. To address this issue, we apply specific criteria to exclude unwanted lines.

The first criterion requires that the line be located on the field. To determine which parts of the image correspond to the field, we use the B-Human field boundary detector [6]. For a line to meet this criterion, both of its endpoints must be below the field border, with a small margin applied. This prevents lines from being detected along the border, which could otherwise pass the adaptive threshold, as the background is set to black in the preceding step.

The second criterion ensures that a line's endpoints do not fall within a detected obstacle. Obstacles primarily include other robots but may also include human referees or goalposts. Since the NAO robot is white and grey, our detector sometimes mistakenly identifies lines on it. This criterion significantly reduces false positives.

Once we have checked the criteria for excluding a line, we pass the line to a white test [1]. To do this, we give a list of points close to the line being tested but outside it. These points are placed on either side of the line within a given interval. Then, within this list, a certain proportion of these points must be less

bright and more saturated than the line we are currently checking. As long as this condition is met, the line is considered good; otherwise, it is rejected. This last step also helps reduce possible false positives in line detection.

3.5 Detection of the center circle

One of the key features of the RoboCup SPL soccer field is the center circle, which has a unique curvature absent from the rest of the field. This makes circle perception a crucial aspect of localization. B-Human's approach [8] relied on detecting white spots in the image to identify lines, with those on the circle being used to estimate arcs. Since the circle's radius is predetermined, B-Human approximated the arc's center. By repeating this process for all lines, they clustered the resulting centers if they were within a certain distance of each other. A minimum threshold for clustered centers was then set to filter out false positives.

In our approach, the Hough transform only detects the endpoints of a given line. Since an arc cannot be determined from just two points on a straight line, requiring a third point instead, we opted to treat all detected lines as tangents to a potential center circle. This allows us to compute two possible center points for each line, one on either side. By applying the same clustering algorithm as B-Human, we can effectively detect the center circle.

This approach is highly efficient, significantly reducing the processing required to detect the circle directly from the raw image. Given the NAO's limited computational resources, this optimization greatly improves overall performance.

4 Results

The evaluation of our line detector is based on a rigorous analysis of the performance of our solution using various metrics. We first detail the process used to implement our detector and test it in order to optimize its parameters. Next, we explain the results obtained on a set of images captured at different locations in the field. Finally, we present the limitations of our line detector, mainly concerning its performance and the management of false positives caused by it.

4.1 Methodology and metrics

Our line detector was first developed as a prototype in Python, serving as a basis for testing the feasibility of our approach. This phase enabled us to evaluate different possible approaches and their efficiencies under controlled conditions and to identify the main challenges related to the detector's robustness and accuracy.

In order to improve the accuracy and optimize the performance of our detector, we performed several automated iterations on the different parameters of our algorithms. By testing different parameters in this way, we sought to simultaneously maximize the accuracy and robustness of our method. These adjustments

were guided by an empirical analysis of the results obtained in our test dataset, allowing us to progressively improve our approach.

The metrics we used to quantify the performance of our line detector are the F-score, precision, and recall, which are metrics commonly used to measure the quality of detection in an image. Precision allowed us to evaluate the proportion of detected lines that are truly relevant, while recall measured our detector's ability to identify all the lines present in the image. The F-Score, combined these two metrics and offered us a balanced evaluation between precision and recall.

To properly evaluate our line detector, we performed these analysis on a dataset of images captured at different locations and under different lighting conditions. To be able to use the aforementioned metrics, we annotated these images by identifying the lines present on them. These annotations served as a reference for comparing the results of our approach and measuring its performance according to these metrics.

These intermediate steps enabled us to integrate our line detector more easily into our C++ code for the NAO. However, after our integration and a few tests on the robot, we noticed some performance issues, notably related to the resolution of the Hough transform. We experimented with different parameters to reduce Hough's execution time. In addition, we also reduced the resolution of the processed images to process fewer pixels. Furthermore, we found that reducing the number of frames per second used for line detection also improved the overall efficiency of the system without compromising its reliability. To reduce the time spent on line detection, a number of frames to skip were added to the code. This meant that on these frames, the line detection would not be performed. We experimented with various skip frame values and tested them in real games. Ultimately, we decided to detect lines every three frames for the upper camera and every two frames for the bottom camera.

4.2 Detection results

Evaluating line detector performance in the field is essential to determine its applicability in real-life situations. To this end, we have selected ten strategic positions to analyze line detection from different angles. Figure 2 illustrates these positions and the robot's orientation in the field.

We then compared our new solution with that of B-Human 2021 by calculating the recall of lines detected in images captured at each position, combining data from both cameras. Table 1 shows the recall obtained for each position, as well as the total recall over all ten positions. In addition, we measured the average brightness (in lux) of each field tested, to assess the performance of our detector under different lighting conditions. These results show that our solution performs well in a variety of environments while minimizing the need for prior camera calibration. In particular, our method performs better than B-Human 2021 under low light conditions, whereas in well-lit environments, both approaches achieve comparable results.

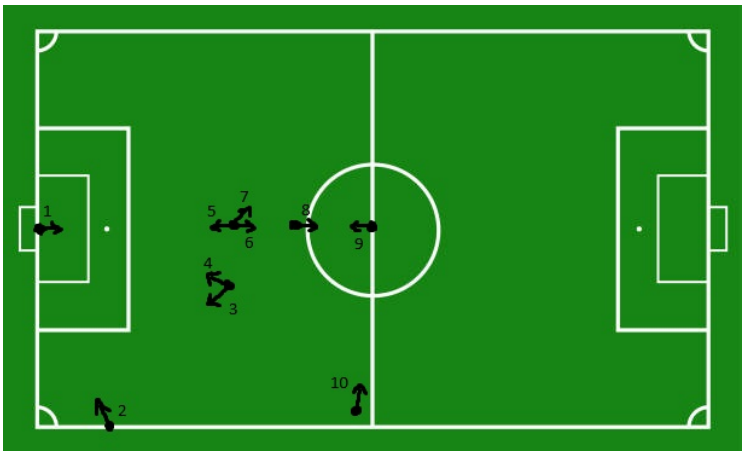

Fig. 2. Line detection results legend

Position	Testing Field				German Open			
	Real Calibration		Slightly Calibrated		Real Calibration		Slightly Calibrated	
	Naova	B-Human	Naova	B-Human	Naova	B-Human	Naova	B-Human
1	22.22%	22.22%	22.22%	11.11%	22.22%	22.22%	22.22%	22.22%
2	57.14%	57.14%	57.14%	28.57%	57.14%	57.14%	57.14%	57.14%
3	80%	80%	40%	40%	60%	60%	60%	60%
4	60%	40%	20%	20%	60%	60%	60%	60%
5	60%	60%	20%	20%	80%	60%	100%	60%
6	18.18%	9.09%	18.18%	9.09%	18.18%	18.18%	18.18%	18.18%
7	100%	100%	50%	50%	100%	100%	50%	50%
8	25%	25%	12.5%	0%	12.5%	12.5%	12.5%	37.5%
9	0%	14.29%	0%	0%	14.29%	28.57%	0%	28.57%
10	40%	40%	40%	40%	40%	40%	40%	40%
Total Recall	37.5%	35.94%	25%	17.19%	37.5%	37.5%	37.5%	40.63%
Mean Luminosity (lux)	139.16				511.29			

Table 1. Line detection results between Naova and B-Human with different calibrations

For center circle detection, we adjusted B-Human's approach to suit our new solution. We then evaluated its performance using the same methodology as for line detection, to verify that the adjustments did not alter its reliability. The results are shown in Table 2, which contains only the positions where the circle was visible in at least one of the cameras. In fact, in some locations, the circle does not appear in any of the images and therefore cannot be taken into account in the evaluation. Regardless of lighting conditions, our method performs worse than B-Human's approach to detect the center circle, highlighting a limitation in our current implementation.

Position	Testing Field				German Open			
	Real Calibration		Slightly Calibrated		Real Calibration		Slightly Calibrated	
	Naova	B-Human	Naova	B-Human	Naova	B-Human	Naova	B-Human
1	0%	0%	0%	0%	0%	0%	0%	0%
6	50%	100%	100%	50%	50%	100%	50%	100%
8	0%	50%	50%	0%	0%	100%	0%	50%
9	100%	100%	100%	0%	100%	100%	100%	100%
10	0%	100%	0%	0%	0%	0%	0%	0%
Total Recall	28.57%	71.43%	57.14%	14.29%	28.57%	71.43%	28.57%	57.14%
Mean Luminosity (lux)	139.16				511.29			

Table 2. Circle detection results between Naova and B-Human with different calibrations

Finally, in order to visually analyze the performance of our detector, we captured images with the NAO's cameras in the field. Figure 3 illustrates these results and shows that our solution provides effective line detection, suggesting its potential for use in real life conditions during soccer games.

4.3 Performance results

One crucial aspect of developing a new line detector for use in a real match is its execution time. Although we anticipated that our new line detector would require more time than the B-Human detector due to its increased complexity, it had to be fast enough to avoid bottlenecking the entire soccer-playing program. Similarly to the line detection results, we compared our line detector with B-Human 2021. The results are shown in Table 3.

Since we reused much of the B-Human center circle detection, the execution times between our approach and theirs are quite similar. Our generation of possible circle centers appears slightly faster than B-Human's, as this is the only significant change.

Regarding line detection times, our execution times are considerably higher than those of B-Human. This is mainly due to the use of OpenCV functions, which tend to be slower in general. The adaptive thresholding, blob filtering,

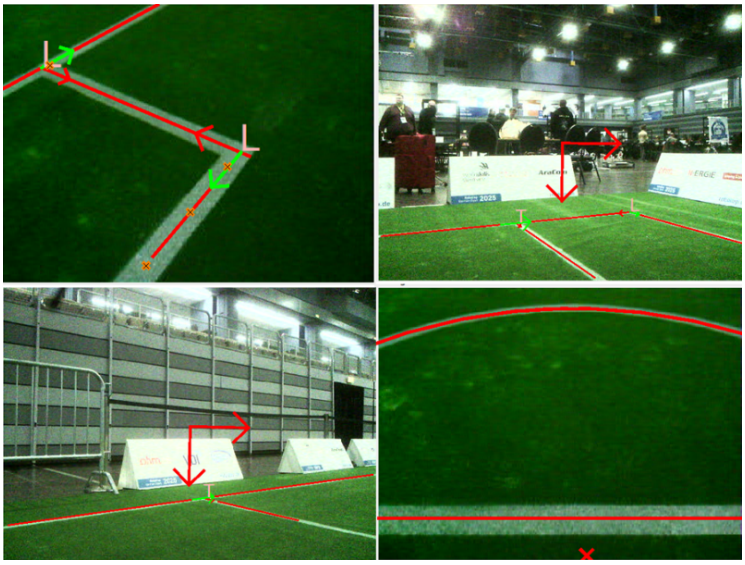

Fig. 3. Line detection results

Detection Type	Camera type	Naova			B-Human		
		Min	Max	Average	Min	Max	Average
Lines detection	upper camera	14.605	19.17	16.1857	0.2	0.392	0.25036
	lower camera	8.238	12.021	8.97825	0.13	0.184	0.14784
Center Circle detection	upper camera	0.006	0.139	0.01316	0.009	0.187	0.01468
	lower camera	0.003	0.007	0.00487	0.008	0.192	0.01627

Table 3. Comparison of Naova and B-Human execution times. All times are in milliseconds (ms)

skeletonization, and Hough transform are the most time-consuming steps, with processing times ranging from 2.2 to 2.9 milliseconds in the upper camera.

4.4 Limitations

Our line detection presents certain limitations in terms of performance, mainly due to the use of the OpenCV library for a large majority of the algorithms used in our solution. The integration of OpenCV forces us to use its functions as they are, without the possibility of optimizing them as we would like. This constraint led us to look for other optimization solutions. We turned to solutions such as reducing the number of frames per second processed and lowering the resolution of the images used for our detection. Although these solutions work to improve the performance of our system, they can also lead to some degradation in our accuracy.

Another challenge is the handling of false positives. In some cases, robots may not be detected as obstacles by our robots, due to their predominantly white color, which can be interpreted as false terrain lines. In addition, when we are close to the goal, the white net strings can also be calculated as field lines. These limitations point to the need to refine our approach in order to improve our detector in these particular cases.

4.5 Real matches usage

We deployed our line detector in real matches at the German Open 2025, where it demonstrated reliability with minimal manual calibration and parameter adjustments. The bright and uniform lighting conditions on the field limited our ability to assess its robustness under varying light conditions. However, we observed an improvement compared to previous years in line detection throughout the competition, even as the painted lines gradually faded.

5 Conclusion and perspectives

The developed line detector is reliable enough to be used during a match. However, there are still areas where improvements can be made. The center circle detection could be further refined to achieve better results. In particular, improving the circle fitting on the image would help reduce false positives while preserving true positives.

The goal of running the detection in real-time on the robot has been partially achieved. While the detector is usable during a match, it cannot be executed on every camera frame due to performance constraints.

Based on our current tests, our detector adapts well to changing lighting conditions. This objective has therefore been met.

There are several ways we envision improving our line detector. First, we aim to address performance limitations. This would require a deeper analysis of the

OpenCV functions we use and potentially developing optimized implementations tailored to the NAO's architecture.

Another approach we would like to explore is adaptive image resizing based on the vertical position in the image. Lines near the robot's feet appear thicker than those farther away. By preserving more pixels in specific areas of the image, we could potentially enhance the detection of distant lines, which is currently a weak point of our approach.

Lastly, it might be possible to use the stereo vision of the two cameras to find the plane of the field and detect false positive.

References

1. B-Human: Localization features (2025), https://docs.b-human.de/master/perception/localization-features/, accessed: 2025-03-26
2. Brameld, K., Hamersley, F., Jones, E., Kaur, T., Li, L., Lu, W., Pagnucco, M., Sammut, C., Sheh, Q., Schmidt, P., Wiley, T., Wondo, A., Yang, K.: Robocup spl 2018 runswift team paper (2018), https://cgi.cse.unsw.edu.au/ robocup/2018/TeamPaper2018.pdf
3. C. Galambos, J.M., Kittler, J.: Progressive probabilistic hough transform (1998), https://dspace.cvut.cz/bitstream/handle/10467/9451/1999-Progressive-probabilistic-Hough-Transform-for-line-detection.pdf
4. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-8**(6), 679–698 (1986). https://doi.org/10.1109/TPAMI.1986.4767851
5. Hafez Farazi, P.A., Behnke, S.: A monocular vision system for playing soccer in low color information environments (2015), https://arxiv.org/pdf/1809.11078
6. Hasselbring, A., Baude, A.: Soccer field boundary detection using convolutional neural networks. In: Alami, R., Biswas, J., Cakmak, M., Obst, O. (eds.) RoboCup 2021: Robot World Cup XXIV. Lecture Notes in Artificial Intelligence, vol. 13132, pp. 202–213. Springer (2022)
7. Loce, R.P., Bala, R., Trivedi, M.: Lane Detection and Tracking Problems in Lane Departure Warning Systems, pp. 283–303 (2017). https://doi.org/10.1002/9781118971666.ch11
8. Monnerjahn, L.M.: Kalibrierungsfreie spielfeldlinienerkennung im roboterfußball (2021), https://b-human.de/downloads/theses/bachelor-thesis-lmonnerjahn.pdf
9. Riebesel, N.: Center circle detection on the nao robotic platform for the robocup standard platform league (2018), https://hulks.de/$_{f}iles/PA_{N}icolas-Riebesel.pdf$
10. Wikipedia: Topological skeleton — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Topological%20skeletonoldid=1258161285 (2025), [Online; accessed 26-February-2025]