University of Amsterdam
Faculty of Science
The Netherlands

Dutch Nao Team

Technical Report

Authors:
Jakob Kaiser
Hidde Lekanne gezegd Deprez
Wike Duivenvoorden
Pim Heeman
Rogier Van der Weerd
Thomas Wiggers

Supervisor:
Arnoud Visser

January 16, 2022

Abstract

In this Technical Report, the Dutch Nao Team lists its progress and activities in the past academic year with the previous report [1] as a starting point. Besides new developments this report also lists older developments when relevant.

This year, progress has been made in developing the framework, vision and localisation. For the framework, the new behaviour engine is finished and a few other improvements have been made. For vision, the previous ball detection system is improved by using the object detection network YOLO-v3 as verification to make the detection more accurate. Field line detection has been also been improved and corner detection has been added. To improve localisation, sweeping behaviour has been introduced using head motions instead of full body rotations to look for the ball. Additionally the middle circle is now also used for localisation. To conclude, even though COVID-19 has made many things difficult, a lot of progress has still been made and we hope to continue improving next year.

COVID has lead to most events being cancelled. However, the team has participated in the Robotic Hamburg Open Workshop (RoHOW), which was organised online, and the RoboCup, which was also organised remotely.

Contents

1 Introduction

The Dutch Nao Team consists of students of different disciplines studying various degrees at the University of Amsterdam: seven bachelor's students from Artificial Intelligence and Computer Science, three master's students from Artificial Intelligence and one master student from Computer Science. They are supported by a senior staff member, dr. Arnoud Visser. The team was founded in 2010 and competes in the RoboCup Standard Platform League (SPL), which is a robot football league in which all teams compete with identical robots to play football. The league was started to incentivise the development in robot science. Its current goal is to play against the human world champion of 2050, and win.

Since all teams participating in the Standard Platform League are obliged to use identical robots, the focus of the league is solely software oriented rather than hardware oriented. The robots need to be able to play autonomously. This includes finding the ball, locating itself on the field, and making decisions on how to play next, as well as communicating with teammates and being able to walk.

2 Hardware

The NAO robot is a programmable humanoid robot made by Softbank Robotics, formerly known as Aldebaran Robotics. Up until 2018, all versions 4.x and above of the NAO were equipped with the same computational hardware, only differing in their sensors or actuators. The release of the sixth version (V6) introduced a significant change in both hardware and proprietary software.

While the V6 hardware is significantly improved compared to previous versions, CPU resources are still scarce. This limits calculation-heavy tasks such as expensive pixel-wise image operations and deep neural network approaches. The NAO has two high-definition (HD) cameras and an inertial board that are both used for the competition. The HD cameras are located in the head of the NAO; one is aimed forward to look for far away objects and the other is aimed downwards for looking at objects close to the feet of the NAO. The inertial board is used for determining whether the robot has fallen, which happens regularly during matches. The NAO also possesses four sonars, an infrared emitter and receiver, pressure sensors and tactile sensors. Except for the pressure sensors, these sensors are used less frequently than the cameras and inertial board, as they are more prone to breaking down, resulting in faulty measurements. The pressure sensors however become one of the most important sensors, since the new walking engine relies heavily on the pressure information coming from the feet for stable walking.

The NAO robot has 25 degrees of freedom in its joints. The joints in the legs allow for stable bipedal movement and various kicking motions, the arms are generally used for helping the robot stand up again after falling and for stability while walking. It is also permitted for the goalie to hold the ball in its hands, but it is highly uncommon for teams to make use of this. Even though every robot is supposed to be the same, individual differences are noticeable when the robots is playing football. The movement of older robots is less stable and fluent, since the joints of these robots have been worn out. In order to ensure a robust walk for every robot, the joints for each individual robot need to be calibrated. Additionally, each robot's camera can shift inside its enclosure, resulting in a slight offset in the transformation with respect to the robot centre. To correct for this, the cameras can be calibrated.

The V6 is a 64-bit system and has an 1.91 GHz Intel Atom E3845 CPU which is a quad-core processor with one thread per core. The amount of RAM has increased to 4 GB DDR3 and the amount of storage has increased to 32 GB SSD over previous versions. Furthermore, the WiFi connection has improved and the fingertips are more resistant to impact. With the added computing power, new approaches to issues like localisation and ball detection will be possible. To make communication between the hardware and software possible, the LoLA (Low Level Access) has been used.[1]

[1] `http://doc.aldebaran.com/2-8/family/nao_technical/index_dev_naov6.html`

3 Framework

Our framework was created in 2017. Its structure is based on [2] and is further elaborated in [3]. Moreover it uses Kahns algorithm [4] to link different modules in a proper sequential order within each module group.

3.1 Behaviour engine

In the previous technical report [1], a new behaviour engine inspired by CABSL [2] from the B-Human team was discussed. This year, work on this behaviour engine has been completed and the old behaviour engine has been replaced by the new behaviour engine.

3.2 Autocreated getUpdates using macros

The getUpdates method in modules was used to receive messages from other modules. However, this function needed to be manually created and updated for every module, even though in almost all cases the function is the same, except for the representations itself. This is error prone and causes overhead since for each new module, the whole getUpdates function needs to be copied, as the function is too difficult to directly write from scratch.

This problem is solved by creating the getUpdates automatically using the preprocessor. To ask for an input message, now only INPUT_MESSAGE(rFoo, foo_message_) needs to be added (and similar for output messages) to a class. This comes with only some minor additional time cost at loading a module during execution as for each message a structure needs to be constructed. However, afterwards, there is zero additional delay compared to the current hard-coded method.

3.3 Interface communication

Due to the addition of different working threads, different representations from different cycles need to be aligned in order to get an accurate picture of the situation. Previously the most recent representations were send to the interface at a given interval. However, this led occasional mismatches between images and the computed annotations such as lines. A sync hing mechanism was implemented using the timestamp of the representations. Additionally in the case of frame loss, the algorithm forcibly updates the lower and upper camera in an alternating fashion, preventing unequal updates for different cameras.

4 Vision

4.1 Object detection

DNT's participation in the 2021 RoboCup SPL Obstacle Avoidance challenge[3] required the development and implementation of robust real-time object detection (primarily ball and other robots on the field) and the capability to navigate though a field of obstacles, while walking with a ball. In order to achieve object avoidance capability, a number of new functionalities were developed and integrated into the existing DNT framework.

[2] https://github.com/bhuman/CABSL

[3] https://spl.robocup.org/rc2021/#oac, https://cdn.robocup.org/spl/wp/2021/01/SPL-Rules-2021.pdf

- First, in order to avoid objects, they will need to be detected with high confidence and projected on a representation of the environment. The current DNT framework contains a Haar feature based cascaded classifier for ball detections[5, 6]. The Haar detector is fast (30 FPS), but inaccurate. It suffers from a high false positive rate and low recall, so this needed to be improved.

- Secondly, Ball and Robot models are required, calculated by applying Kalman filters on (potentially noisy) detection signals, in order to make best estimate predictions of object locations with respect to the robot.

- Thirdly, a Navigation module is required to plan a feasible and optimal path to a target position. From that point on, the existing behaviour engine can be used to define a behaviour that consists of walking with the ball to the first waypoint in the queue.

After a review[7], Tiny Yolo-v3[8] was selected as a suitable high performing detection algorithm. It is preferred over optimised algorithms for specific classes given the ability to include additional classes in the future that can be relevant for the DNT framework such as goal posts, penalty markers, different robot stances, etc. Although more recent versions of Yolo are available[9, 10, 8, 11], Yolo-v3 has many well established implementation frameworks and portability options and it contains all critical improvements needed for DNT's purposes.

4.1.1 Data generation, model training and evaluation

Train and validation datasets were generated from existing resources[4] and own recorded and annotated images. A total of ca. 8,000 images were used (details in [7]). An untrained Tiny-Yolo-v3/3L model with three scale layers (ca. 8 mln trainable parameters) was used and trained on 170 epochs using the Darknet[5] framework. The overall Mean Average Precision on the test data for a confidence threshold of 0.5 is $MAP_{@0.50} = 86\%$, with nearly equal performance on ball and robot classes. The overall F1 score is 0.79. These were considered good results given the diversity and, at times, complex detection scenes (with high levels of motion blur, partially occluded or distant objects).

Performance was further evaluated by recording a test match on DNT's home field and analysing a randomly selected sequence of ca. 600 consecutive frames of game-play. Performance on Ball and Robot detection can now be evaluated separately. For ball detection, we are mixing results of two detectors (Yolo and the Haar classifier), each of which can yield a True Positive (TP), False Positive (FP), True Negative (TN) or (False Negative) result. Thus there are $4 \times 4 = 16$ possible outcomes. Examples of the most common outcomes are shown in Figure 1. Detailed analysis is provided in [7]. Figure 2 presents the key resulting evaluation metrics.

A clear performance improvement is observable when the Yolo detector is used. Yolo's ball detection precision is nearly perfect and recall improves from 19% (Haar classifier) to 70% (mixed detection). Recall on robot detection is 76%. Overall F1 scores improve from 0.3 (Haar classifier) to 0.8 and above when using the Yolo detector.Visual inspection of the result indeed confirms many false positive detection errors by the Haar classifier in regions of the robot (see [7] for details). Many ball detections are missed by Haar, especially at higher distances. Yolo, on the other hand, is robust against motion blur and successfully detects at different distances (scales) as is expected given the 3-layer scale pyramid used in Tiny Yolo-v3/3L.

[4] `https://imagetagger.bit-bots.de`
[5] `https://github.com/AlexeyAB`

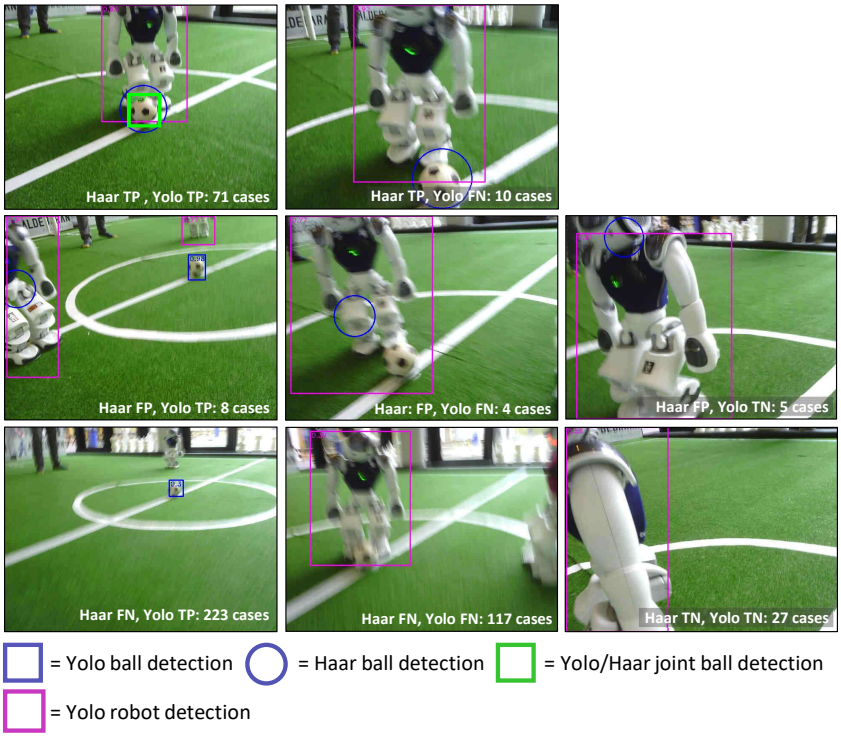

Figure 1: Illustration of potential outcomes of mixed Haar classifier and Tiny Yolo-v3/3L based ball detections

	Haar	Yolo	
	ball	ball	robot
Precision	83%	100%	99%
Recall	19%	70%	76%
True Neg Rate TNR	61%	100%	91%
Accuracy	23%	72%	77%
Balanced Accuracy	40%	85%	84%
F1-score	0,3	0,8	0,9

Figure 2: Detection performance of Haar vs. Tiny Yolo-v3/3L

4.1.2 Deployment on the robots

A basic deployment of this model on a standard Nao-v6 was implemented using OpenCV's Deep Neural Net (dnn) module[6], which can load and execute Darknet encoded models through its API. Execution time of inference on the Nao Robot's CPU[7] is about 700ms, equivalent to ca. 1.4 FPS. This is too slow for real-time detection, with the robots operating at a frame rate of ca. 30 FPS. In order to have a functioning baseline detector, two steps were taken:

1. New `ballModel` and `robotModel` classes were created, that apply Kalman filtering on the low-frequency Yolo detection signals. The signals are infrequent but very reliable and hence this probabilistic approach enables a stable and usable estimate of ball and robot positions.

2. Detection frequency is particularly important tracking faster ball movement. Ideally, we would be able to use high frequency Haar detector signals whenever they have been validated by the more reliable (but slower) Yolo detection. A 'Mixed Ball Signal Generation'[7] algorithm was implemented to combine the speed of the Haar classifier with the precision of Yolo. The key idea is to validate Haar detections whenever infrequent, periodic Yolo detections are executed. If the Haar classifier is in validated state and detects positively in subsequent frames (without Yolo executing), we assume it is still valid, as long as the time between detections is below a threshold. Otherwise, we consider the Haar detection a False Positive and discard it.

[6] `https://docs.opencv.org/4.3.0/d6/d0f/group__dnn.html`, (The DNT framework currently uses OpenCV v.4.3.0)

[7] Intel Atom E3845 Quad Core @ 1.91 GHz, 4GB DDR3

Other methods for speeding up the Yolo detector are possible (compiler-based optimization, quantization and model pruning), these need to be explored in future work.

4.2 Field line detection

The existing field line detection algorithm, developed in 2019[12], suffers from a high false-positive rate. This is problematic since detected lines play a key role in the particle filter that is used for localisation[12]. An analysis of the false-positive instances showed this was particularly prevalent on robots in the image. Indeed, white tones on the robot tend to be very similar to those of field lines in recorded images and are likely to yield many Canny edge detections that trigger positive line detections.

Visual inspection showed that in the vast majority of cases, false positive line detections were for lines with a vertical orientation and/or with a low projected length. Hence, a simple solution was implemented that filters detected lines that are near-vertical or below a certain threshold length. These parameters were tuned on a test set. Figure 3 shows two typical examples.

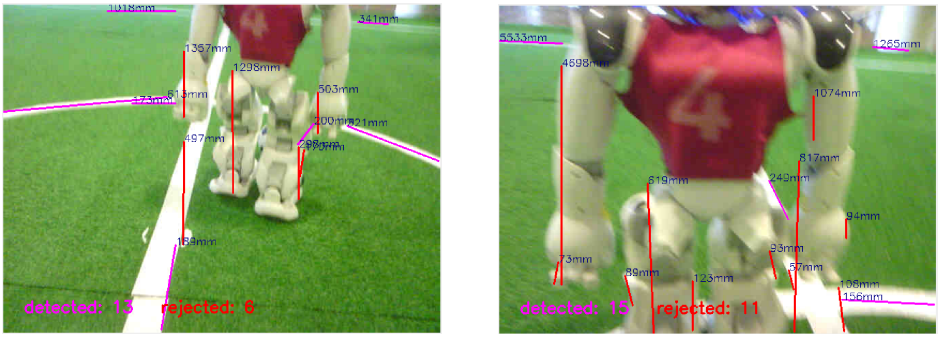

Figure 3: Example instances of rejections by the line detection filter

Filtering out vertical lines will also affect true-positives. This may occur when the robot looks straight down a field line. However, these instances are rare and localisation can temporarily rely on odometry alone in these cases.

4.3 Corner detection

A new module for corner detection has been added. This module uses the current line detection. It looks through all pairs of detected lines and checks checks for two conditions. The angle between the lines in a pair must be around 90 degrees and the lines must be close enough to their point of intersection (the accepted distance is dependent on the length of the lines). If these are both true, then that pair of lines forms a corner. This module can later be used to improve localisation by basing localisation partly on the detected corners.

5 Localisation

5.1 Sweeping behaviour

When playing a match, looking for the ball is a critical aspect. The current technique of finding the ball consists of three stages. When the robot first loses track of the ball, the robot will go to the place it has last found it. If the robot has lost the ball for longer than 13 seconds it will start

looking at its surroundings. To do this the robot rotates its whole body. This takes up a lot of time and is not very effective. If this too does not work, the last thing the robot does is go back to the ready position. To improve our current method of searching for the ball we needed to do more with head motions. As the robot never turns its head, only its whole body. The idea is that a robot will now "sweep" with its head moving it continuously from left to right, starting from a random angle and not turning past the shoulders. This function is a replacement of the full body turn in the second option when the robot has lost the ball for longer than 13 seconds.

5.2 Localisation based on the middle circle

When the robot is totally lost on the field, it is useful to be able to completely reset the localisation when the robot stumbles along a distinctive landmark. The middle circle is such a landmark, and conveniently defined as a circle with a line through it. The combination of line and circle makes it possible to calculate the precise location of the robot when these two features are observed. This information is then used to improve localisation by adding this position as a function of $P_{i+1} = P_i * \alpha + \hat{P}_i * (1 - \alpha)$ where alpha is the update constant between 0 and 1. This way the robot is able to re-localise without being wrongly localised after a false positive detection.

6 Tools

6.1 Save data functionality

When recording games it is critical to get a complete recording of all incoming sensor data that robot perceives. Previously this was made possible by saving all the data at the end of a vision cycle, the slowest cycle, using that frame number to replay the inputs in order. This meant that some quicker frames, like motion, were lost. However, last year there was the addition of sound-processing during the game, which takes about half a second per frame. This would be the new "longest cycle" in our framework. This led to an undesirable choice where we had to duplicate the audio input 30 times to retain frame numbers, or miss 30 vision cycles every frame. This demanded a more sophisticated solution for recording games.

A transition was made towards timestamp based recordings where each sensor reading was saved with their timestamp and replayed in real-time. This also added the functionality of speeding up and slowing down the replays as a function of time instead of frames, making the timing independent of the speed of execution at the time. This was achieved by rewriting the current module performing the recording task. Additionally, the message processing had to change in order to facilitate playing back from already defined timestamps.

7 Results

7.1 Remote Robotic Hamburg Open Workshop (RoHOW)

RoHOW[8] is an annual open workshop for SPL teams, organised by the German team `HULKs`[9]. It took place from the 4th to the 6th of December 2020 and was hosted online on discord. During the event, test games are played, algorithms and ideas about challenges are shared, and lots of coding is done. Usually, without the pressure of competitive games this makes it a great opportunity for

[8] `https://rohow.de/2020/en/`
[9] `https://www.hulks.de/`

new team members to get to know the flow of working with robots and being in the SPL. However since everything was online the Dutch Nao team did not participate in test games. Several subjects like event based-communication, safe falling motions, SPL rules and potential events to take place in 2022 were discussed.

7.2 Remote RoboCup

The Dutch Nao Team qualified for the RoboCup 2020 in Bordeaux with a video[10] and a qualification paper [13]. This competition was cancelled to be held in 2021. The team's qualification was still be valid for this edition, which was organised as a virtual distributed event due to covid. For this reason the competition was organised in a different manner then previous years. It consisted of 4 distinct challenges:

1. Obstacle Avoidance Challenge
   In this challenge a single robot had to score a goal from the kick-off spot as fast as possible while the path is obstructed by four obstacles and the robot is only allowed to score from beyond the penalty mark.

2. Passing Challenge
   In this challenge two robots need to make as many passes as possible around two stationary defenders in 5 minutes, while the ball does not touch these defenders.

3. 1v1s Challenge
   In this challenge two robots play a 1v1.

4. Autonomous Calibration Challenge
   In the first part of this challenge, the robot has the opportunity to collect data from which it "self calibrates". In the second part of the challenge, the robot performs two tasks, namely walking to fixed positions and stating the positions of two soccer balls that are placed on the field, this is done to judge the accuracy of calibration. The robot is then judged on speed and accuracy.

Team were not allowed to complete challenges in their own arena. This means the Dutch Nao team completed the challenges in arena's of other teams, while hosting other teams that completed their challenges in Amsterdam. This did add the extra difficulty of setting up everything for a challenge remotely.

The Dutch Nao team participated in the obstacle avoidance challenge and in the 1v1s challenges. For the obstacle avoidance the Dutch Nao team ranked 12th. For the 1v1 challenges the Dutch Nao Team lost the first match to `B-human` and won the next match against `SPQR Team`. Then during play-ins, the Dutch Nao team lost to `UT Austin Villa`.

7.3 Foundation

In 2016, the Dutch Nao Team started the foundation *Stichting Dutch Nao Team*, in order to be able to administer grants, finances and activities in a transparent way.
As per 2020-2021, the board of the foundation consists of the following members:

- Chair: Wouter Zwerink

[10] `https://www.youtube.com/watch?v=SKawcYDEhjo`

- Secretary: Pim Heeman
- Treasurer: Thomas Wiggers

7.4 Public events

Due to COVID measures in 2021 not many events have taken place. Events cancelled due to the reasons mentioned above include an event for students by Jet-Net & TechNet and a careerday by Betapartners. For the second event an alternative was set up but due to lack of registrations this was also cancelled.

8 Contributions

What follows is a list of people who worked on the additions mentioned in this report, in alphabetic order:

- **Hidde**, who added save data functionality, worked on localisation based on the middle circle and improved the interface communication.
- **Jakob**, who worked on the new corner detection.
- **Pim**, who worked on autocreated getUpdates methods.
- **Rogier van der Weerd**, who worked on field line detection, object detection, object avoidance and navigation.
- **Thomas**, who worked on the new behaviour engine, object detection, object avoidance and navigation.
- **Wike**, who worked on sweeping behaviour.

9 Conclusion

This year, several new steps have been take in developing the framework, vision and localisation. COVID has had a massive impact on the events we could take part in. Even though most events have been cancelled, the Dutch Nao Team did participate in the RoboCup and the RoHOW. Our team looks forward continuing improving and we hope that more events will be possible next year.

References

[1] Pim Heeman et al. *Dutch Nao Team - Technical Report*. Tech. rep. Universiteit van Amsterdam, 2020. URL: `https://www.dutchnaoteam.nl/wp-content/uploads/2021/01/dnt_techreport_2020.pdf`. published (cit. on pp. 2, 4).

[2] Elizabeth Mamantov et al. "Robograms: A lightweight message passing architecture for robocup soccer". In: *Robot Soccer World Cup*. Springer. 2014, pp. 306–317 (cit. on p. 4).

[3] Douwe van der Wal, Pieter Kronemeijer, and Caitlin Lagrand. *Dutch Nao Team - Technical Report*. Tech. rep. University of Amsterdam, 2017. URL: `http://www.dutchnaoteam.nl/wp-content/uploads/2018/01/dnt_techreport_2017.pdf`. published (cit. on p. 4).

[4] Arthur B Kahn. "Topological sorting of large networks". In: *Communications of the ACM* 5.11 (1962), pp. 558–562 (cit. on p. 4).

[5] Duncan ten Velthuis. "Nao detection with a cascade of boosted weak classifier based on Haar-like features". Bachelor's Thesis. Universiteit van Amsterdam, 2014 (cit. on p. 5).

[6] Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001.* Vol. 1. Ieee. 2001, pp. I–I (cit. on p. 5).

[7] Rogier van der Weerd. *Project AI: Real-time object detection and avoidance for autonomous Nao Robots performing in the Standard Platform League*. Tech. rep. University of Amsterdam, July 13, 2021. URL: https://www.dutchnaoteam.nl/wp-content/uploads/2021/10/ProjectAI-report.pdf. published (cit. on pp. 5, 6).

[8] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018) (cit. on p. 5).

[9] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91. URL: https://doi.org/10.1109/CVPR.2016.91 (cit. on p. 5).

[10] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690. URL: https://doi.org/10.1109/CVPR.2017.690 (cit. on p. 5).

[11] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection". In: *arXiv preprint arXiv:2004.10934* (2020) (cit. on p. 5).

[12] Thomas Wiggers et al. *Dutch Nao Team - Technical Report*. Tech. rep. Universiteit van Amsterdam, 2019. URL: https://www.dutchnaoteam.nl/wp-content/uploads/2020/01/dnt_techreport_2019.pdf. published (cit. on p. 7).

[13] Hidde Lekanne gezegd Deprez et al. *Team Qualification Document for RoboCup 2020 Bordeaux, France*. Tech. rep. Science Park 904, Amsterdam, The Netherlands: University of Amsterdam, 2020. URL: https://www.dutchnaoteam.nl/wp-content/uploads/2020/01/TeamQualificationDocument2020.pdf. published (cit. on p. 9).