University of Amsterdam
Faculty of Science
The Netherlands

Dutch Nao Team

Technical Report

Authors:
Douwe van der Wal
Pieter Kronemeijer
Caitlin Lagrand

Supervisor:
Arnoud Visser

December 31, 2017

Abstract

In this Technical Report, the Dutch Nao Team presents its progress from the past year as a continuation on the progress of the previous year (Caitlin Lagrand 2016a), and an outline of the developed modules. In the academic year of 2016-2017, the team participated in the Robotic Hamburg Open Workshop (RoHOW), RoboCup Iran Open in Tehran and the 2017 RoboCup in Nagoya. During the RoboCup Iran Open the team used B-Human's framework in development, in addition to two new modules. After the Iran Open it was decided to make a new framework from scratch. The aim of this decision was to make all code comprehensible for new team members, as the composition of the team changes frequently. To this end the framework is module based and fully documented. So far the team has ported its behaviour-engine and made a new ball-detection module. This DNT-framework was used during the 2017 RoboCup World Finals in Nagoya.

Figure 1: Team photo in Nagoya

1 Introduction

The Dutch Nao Team consists of eight Artificial Intelligence students; two Master students and six Bachelor students, who are supported by a senior staff member. The team was founded in 2010 and competes in the Standard Platform League (SPL); a robot football league in which all teams compete with identical robots to play football fully autonomously. The league was started to incentivize 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 are non-player controlled, and have to be able to play football by themselves. This includes finding the ball, locating itself and making decisions on how to play next, as well as communicating with teammates and being able to walk.

2 Hardware and framework

2.1 NAO

The Nao robot is a programmable humanoid robot made by Softbank Robotics, formerly known as Aldebaran Robotics. The CPU used by the robot is a 1.6 GHz Intel Atom single physical core which is hyper-threaded, combined with 1 GB of RAM and 8 GB of storage. Since there are only two logical cores, CPU resources are scarce, which limits calculation heavy tasks such as ball detection and localisation. 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 if the robot has fallen, which is something that happens regularly during matches. The Nao also possesses four sonars, an infrared emitter and receiver, pressure sensors and tactile sensors. These sensors are however less frequently used than

the cameras and inertial board, because they are not as essential to the footballing capabilities of the robot. However, what is essential in order for a robot to play football, is movement options. Depending on its version the Nao Robot has either 14, 21 or 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 whilst 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 walk. 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.

3 Framework

After the RoboCup Iran Open the Dutch Nao Team decided to make its own framework. This was decided after an unsatisfactory event, which included many difficulties with the codebase of B-Human(Röfer et al. 2016). Besides this trouble grasping some of the concepts, it is also against the philosophy of the Standard Platform League to copy code of other teams, since this does not contribute to advancements in the field of robotics. Furthermore, making a new framework allows the team to add a number of desirable features. One of those features is the use of proper documentation. This will allows new members in the ever changing composition of the team to more easily understand the code they are going to work with. Another feature is that the framework is completely module based, which makes reasoning over parts of the code easier.
The framework is based on independent modules which communicate with each other trough representations. This idea is based on (Mamantov et al. 2014). A representation is a container that has some information in it about a certain topic. It might for example contain information about the current joint angles of a robot, a camera image or the current time stamp.
Most modules require certain representations as input and all modules return one or more representations as output. Take for example a module that handles the robot getting up after it has fallen down. This module has to be able to figure out whether the robot is lying face down or face up. Therefore it requires an input representation that contains inertial sensor data. As an output this module could for example give the joint angles that bring the robot one step closer to standing up. When a module requires a certain representation to work, then this representation is passed directly from the module that provides this representation to the module that needs it. For example, if the ball detector module requires an image from the camera module, then this representation is passed directly from the camera module to the ball detector module. This representation is only available to the modules that need it. During start-up of the framework, the chain of execution of the models is calculated using Kahns algorithm (Kahn 1962), for topological sorting. This means that the modules that don't require any input are executed first, sending their data to the other modules to finish the cycle.
A problem would be when module 1 requires a representation which is outputted by module 2, but module 2 also requires the output of module 1. Here Kahns algorithm can not make a chain of module executions to make things work. A way around this is with starting the first time with a dummy representation which contains dummy values ignored the first time by the module.

4 Motion

4.1 Walk

The new framework of course did not come with a working walking engine such as in the B-Human framework (Graf and Röfer 2011) which the team used before. Since there are no members in the team who have the knowledge or feel like acquiring this knowledge to make a walking engine, the decision was made to use an engine from another team.
After carefully considering multiple walking engines from a number of different teams, the choice fell on a walking engine made by the Nao Devils (Matthias Hofmann 2016). A big advantage of this engine is that requires relatively few parameters to be tuned, while still walking rather stable and fast, even on the new type of artificial turf. This fits right into the aim of the team's framework of making all modules as surrounding-indifferent as possible, resulting in less calibrations and therefore more robustness.
The next step was to implement the engine into the framework. This turned out to be quite the challenge, since it is a rather large module requiring a substantial number of inputs, which are all assumed to have a specific form that is not (yet) present in the DNT framework. After all the files had been adapted to fit into the framework, a lot of debugging was needed. So much that there wouldn't be enough time to get it done before the RoboCup in Nagoya. Therefore a temporary solution was installed until the more permanent solution of the Nao Devils' engine would work, given the team's observations during previous RoboCup events.
The temporary solution consisted of using the built-in walk of NaoQi, with the idea that any walking engine is better than no walking engine. This walk works, in the sense that the robot can move in a certain direction, but that's about it. The robot falls quite often on the new artificial turf. To counteract this, the walking speed has been lowered to a point where the robot does not fall over all the time. The maximum walking speed was set to 0.2 m/s, which was empirically tested at the RoboCup Nagoya to be the optimal value in the trade-off between walking speed and walking stability.

4.2 Getup

Another important part of motion is the robot getting up. Given the limited time before the RoboCup, this module was also borrowed from another team. The getup module used by the Nao Devils' team appeared to be much more detached from the rest of their framework than their walking engine is, so the choice was quickly made to use their getup engine during the RoboCup Nagoya instead of making our own. It ended up fitting in the framework quite nicely. The goal of the team is to make it's own get up motion after the RoboCup.

4.3 Motion selection

One of the problems that were encountered during the making of different motions was that every motion module in the framework updates all the time and sends it's output, often joint angles, to memory. In the end only one motion can be executed, so this is highly inefficiënt. Therefore a new module was needed to control all the motion modules and make sure that only one module updates per cycle in the framework, in order to make sure no computing power is wasted. This module is called motion selection (Siciliano and Khatib 2016).

4.4 Interpolation engine

During the Robocup in Nagoya, an attempt was made to create a general module which can output any motion given some "keyframes", which are a sequence of joint angles for every joint, in combination with an equally long sequence of corresponding timestamps. In this case the corresponding names of the joints are also provided. This representation of the keyframes is inspired by the Choregraphe (Pot et al. 2009) keyframes. This allows for easy motion designing within Choregraphe, with the option to simply copy and paste the output to get it to work in the framework. The progress looked promising, but this module could not be used during the RoboCup due to stability issues.

5 Detecting the ball

Detecting a ball is not a tremendously hard task when enough computational resources are available, as has been proven by (Lagrand, Wal, and Kronemeijer 2017). The challenge posed when using a Nao robot, however, is to create a ball detector that works fast enough to not disturb the other processes running within the framework. Therefore, our ball detector processes patches of the image using a pipeline of 3 classifiers, which go from computationally cheap to computationally expensive. After every classifier some patches can be ruled out of the set of patches that might contain a ball, thus making subsequent classifiers significantly faster.

Besides being fast, the ball detector should have a number of other features. First of all, and most importantly, the ball detector should work in every reasonably lighted condition, including changes in the lighting such as shadows moving over the field. As of 2016 dynamic lighting was introduced to the SPL, meaning that lighting conditions might change during a game, which the ball detector should be prepared to handle. Secondly, it is preferable that the ball detector needs as little calibration as possible in order to reduce human error and save time before the start of a match.

5.1 The pipeline

Before starting the pipeline a filter is applied to the entire image. In the image we check what the darkest and lightest colours are. Then a value between the minimum and maximum colour is picked, and every pixel becomes white if it's higher than the picked value, or black if it's lower (Gevers et al. 2012). This is internally represented as a matrix filled with ones and zero's, where a one means that the pixel is white. The idea behind this is that despite the lighting conditions, the white parts of the ball will always be lighter than the green field, and will thus become white.

5.1.1 Checking all locations

For the next step using forward kinematics to calculate the position of the camera relative to the field, a calculation is done for every spot in the image to find out how big a potential ball would be at that spot. Then a check is done on a square of the right size to fit the ball to see whether at least 15% of the pixels in this location are white; this eliminates the entire green field since that barely contains any white pixels. This operation is rather cheap, since it just comes down to taking the sum of the region we wanted to inspect. Next up, the average position of a white pixel in the square is checked. If this position is significantly far away from the middle of the image, the square is moved towards this position up to 3 times. This process makes sure that the ball ends up in the middle of a square

The next square that will be checked has about a third overlap with the previous square, to prevent getting two squares with just half a ball in it but also making sure the square doesn't have to be moved more than 3 times.

5.1.2 Detecting blobs

After the last step we have quite some squares taken from the image which are moved towards the white object that was potentially partially in there. Now the preprocessing step, described in the introduction of this paragraph is done again. This is needed because lighting can differ on different spots on the field due to shadows or reflections. This step makes sure that most of the white that's on the ball, is white in the binary image as well.

The next step is to put the image into a blob detector [1]. This blob detector is a single pass algorithm using linked lists to keep track of the blobs. This blob detector is at least 4 times faster than the OpenCV (Bradski 2000) blob detector. This does come at a cost, since it is not possible to give features of the blobs which will be returned, such how circular they are. After the blobs have been found, they are subjected to some tests.

1. Is the ratio between the height and width at least 0.25 and at most 2?
2. Is the height and width at least 2 pixels and at most half of the size of the region of interest?

When blobs of the right size have been found, the square continues to the next stage in the pipeline. A similar technique is applied by another team competing in the SPL, UT Austin Villa. This team looks for three black spots in a ball before continuing to the final classifier (Menashe et al. 2017). This technique gives less possible ball locations, but requires an almost perfect binary image, to have three black spots on a ball visible. This is not the case with our approach to create a binary image, and thus the black spots are used in a different way.

5.1.3 Haar-classifier

Another SPL team, SPQR, has released their ball detector (Bloisi et al. 2017) which mainly relies on a Haar-classifier after doing the preprocessing in a different way. We have used their trained Haar-classifier to decide whether the remaining squares are in fact a ball or not. All the preprocessing discussed in this section was needed to be able to use this Haar-classifier, since it returns a lot of false positives, especially in robots and in areas outside the playing field.

5.2 Performance

The ball detector has been evaluated in the Robolab of the University of Amsterdam, which contains an old field, so without artificial turf, and is about the size of half a real field.
While testing it was clear the ball detector performed less well than during a real match, this is due to the fact that the Haar-classifier gives a lot of false positives when given images which don't mainly contain green or a ball. Since there is no field detection yet, the ball detector can still put regions of interests from outside the field into the Haar-classifier.
The first test performed was putting the robot in the middle of the field, where two other robots were placed, while having the ball in it's view at a certain distance. If the robot managed to reach the ball, this counted as a positive, a negative otherwise. After every attempt the robot was turned

[1] https://github.com/keenerd/quickblob

Table 1: How many times the robot ended up at the ball at a certain distance

1M	2M	3M	4M
6/8	6/8	3/4	1/2

around 45 degrees with distances of one and two meters, so the background changed trough out the test. Due to the size of the field, only four test could be performed at a distance of 3 meters, placing the ball near the corners of the field. At 4 meters the robot could no longer be placed on the middle of the field, so it was placed more towards the goal.
A second test consisted of placing the robot in the middle of the field, without any ball present. If no ball is found for 5 seconds, the robot started turn around to find a ball. This consistently worked well as long as the robot mainly had field in front of him, but as soon as it started to look towards the short side of the field, a false positive was found in objects next to the field.
In conclusion it can be said that the ball detector works reasonable right now and could work great when field detection is added as well.
The ball detector is still a bit slow, with at least 70ms to process one image from both the lower and upper camera, but a large part of the code still leaves room for optimisation.

6 Results

6.1 Iran Open (Tehran)

The Iran Open was the first competition the team competed in this year. It took place from the 5th to the 7th of April 2017 in Tehran, Iran. The Dutch Nao Team used the B-human framework and in addition to that a behaviour-engine and ball detector of own making.
The ball detector that was used at the Iran Open consisted of a similar pipeline as the ball detector described in this document but used a neural network as final classifier. This did not work well in the end because the neural network was not trained on data recorded by a moving robot, resulting in the robot seeing balls everywhere when moving.
The Iran Open did not only consist of playing soccer matches. There also was an 'open challenge', in which each participating team gave a small presentation about one of the things it has been working on to implement in the robot. The other teams give scores for various aspects of the presentation. The team with the most points wins the challenge. The Dutch Nao Team gave a presentation about it's behaviour-engine and was voted as the winner of this challenge by the other participating teams. This behaviour engine has been thoroughly explained in the technical report of last year (Caitlin Lagrand 2016a).
During this competition some frustration about the usage of the large quantity of code from other teams arose, since that makes it hard to get a good sense of the way the framework operates and how you can best implement new features. This eventually led to the making of our own framework, which is described above in section 3.

6.2 Robotic Hamburg Open Workshop (RoHOW)

RoHOW[2] is an annual open workshop for SPL teams, organised by the German team "HULKs"[3]. It took place from the 25th to the 27th of November 2016. During the event test games are played, algorithms and ideas about challenges are shared, and lots of coding is done. The atmosphere is

[2]https://www.rohow.de/2016/de/
[3]https://www.hulks.de/

Figure 2: Team picture after receiving the prize for winning the Open Challenge in Iran

relaxed due to the lack of pressure of competitive games, which makes it a great opportunity for new team members to get to know the flow of working with robots and being in the SPL. This is one of the reasons the team went there this year. Due to issues with the ball detector no games were played at this event.
The more experienced members of most teams gave presentations or workshops about a broad range of subjects, ranging from raising funds to be able to attend international competitions to tips and tricks with compiling your code.

6.3 RoboCup Nagoya

The Dutch Nao Team qualified for the RoboCup 2017 (27 to 31 July 2017) in Nagoya with a video[4] and a qualification paper (Lagrand et al. 2016). This was the first event after the decision to make a new framework. It was decided to use the new framework despite it being rather basic due to lack of time. In addition to this, an upgraded version of the ball detector was used for the first time. As walking engine NaoQi was used. Kicking was not yet available.
During the first round (round Robin 1, the group stage in the competition), the Dutch Nao Team played $0-0$ against both Luxembourgh United and Aztlan. The team had trouble with crashes of the framework and the get-up engine did not work yet, with as result a lot of 'request-for-pickup's. During the second round, the play-in round, the Dutch Nao team lost 0 - 1 versus Nomadz and played $1-1$ versus JoiTech due to an own goal by JoiTech. A robot of the Dutch Nao Team did help to get the ball close to the goal, so the goal wasn't made without effort from the team's side. During this round the Dutch Nao Team did not earn enough points to qualify for the quarter finals.

The Open Challenge consisted of a penalty competition this year, which took place at the very beginning of the competition, where the team could not yet kick the ball, only walk towards it and dribble. This resulted in walking to the ball successfully, but, as expected, no goals.

[4] `https://www.youtube.com/watch?v=FqzGRQzBJwo`

6.4 Foundation

At the beginning of the year the decision has been made to put a lot more effort into sponsoring than in the previous year, with as goal funding the trip to the RoboCup in Japan. In order to be able to receive money from companies in a transparent way, the Dutch Nao Team needed it's own bank account. To this end, a foundation has been started, with the following advantages:

1. Being able to open a bank account
2. Not having to pay taxes over money earned by either sponsoring or giving demonstrations (unless the team earns more than 22.689 euros in a year)
3. Having the Dutch Nao Team as a legal person

The board of the foundation consisted of the following members:

1. Chairwoman: Caitlin Lagrand
2. Secretary: Patrick de Kok
3. Treasurer: Sebastièn Negrijn

6.5 Public events

During the entire year the team has given paid demonstrations and lectures for companies and the municipality of Amsterdam. The goal of these demonstration was to raise funds for the trip to the world cup in Nagoya, Japan. Most of the demonstrations required custom software to be made. Some of the things that have been developed are:

1. An interface to communicate with the robot where IBM's Watson returned an answer
2. NaoQi modules that start a desired behaviour on start-up, which enables usage without needing a laptop
3. An android app to make a Nao walk or do a certain emote. This is an easy way to allow the public to move
4. Presentations about what AI is and what the challenges are in robot soccer right now
5. The protocol to give a demonstration of having robots take penalties

This year the team raised about 5500 euros, half of this was earned with demonstrations and presentations for companies and the other half was received from the Amsterdams Universiteitsfonds. In total 5 paid events were attended.
A special event was the BNAIC 2017 conference, where the Dutch Nao Team also contributed a paper, (Caitlin Lagrand 2016b) in Amsterdam where the Dutch Nao Team won the prize for the best demonstration. The demonstration consisted of one robot detecting the ball and taking a penalty, and one robot being a goalie and diving left or right depending on where the ball was going.
Using the contacts and software we made this year, the goal is to raise more money next year while spending less time on preparations.

7 Conclusion

With the change of a foreign framework to a home-made framework the team now has code that all the members understand and this knowledge can be passed on to newer members easily because of the proper communication and clean code. Due to having to build everything from the ground up, the results were not outstanding this year, but will improve next year as localization and a better motion engine will be put in place. When the framework is stable and has all the basic functions, reinforcement learning could be used to create good behaviours for the robots using reinforcement learning (Lagrand 2017)
The Nao robots have also been used for other activities than football in the Robolab this year, such as being a personal teaching aid (Aerssens et al. 2017) or picking up tomato's (Lagrand, Meer, and Visser 2016).

References

Aerssens, Sara et al. (2017). "Personal Teaching Aid". In:

Bloisi, Domenico et al. (2017). "Machine Learning for RealisticBall Detection in RoboCup SPL". In: *arXiv preprint arXiv:1707.03628*.

Bradski, G. (2000). "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools*.

Caitlin Lagrand Michiel van der Meer, Jonathan Gerbscheid Thomas Groot Sebastien Negrijn Patrick de Kok (2016a). *Dutch Nao Team - Technical Report*. Tech. rep. Universiteit van Amsterdam, FNWI. URL: `http://www.dutchnaoteam.nl/wp-content/uploads/2016/11/TechReport_DNT_2016.pdf`. published.

Caitlin Lagrand Patrick de Kok, Sebastien Negrijn Michiel van der Meer Arnoud Visser (2016b). "Autonomous robot soccer matches". In: *BNAIC2016 Proceedings*, pp. 237–238. URL: `http://bnaic2016.cs.vu.nl/images/bnaic/documents/BNAIC_2016_Proceedings.pdf`. published.

Gevers, Theo et al. (2012). *Color in computer vision: fundamentals and applications*. Vol. 23. John Wiley & Sons.

Graf, Colin and Thomas Röfer (2011). "A center of mass observing 3D-LIPM gait for the RoboCup Standard Platform League humanoid". In: *Robot Soccer World Cup*. Springer, pp. 102–113.

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

Lagrand, Caitlin, Michiel van der Meer, and Arnoud Visser (2016). "The Roasted Tomato Challenge for a Humanoid Robot". In: *Proceedings of the IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. Bragana, Portugal, pp. 341–346. URL: `https://staff.fnwi.uva.nl/a.visser/publications/roasted-tomato-challenge.pdf`. published.

Lagrand, Caitlin, Douwe van der Wal, and Pieter Kronemeijer (2017). *Detecting a checkered black and white football*. Tech. rep. Science Park 904, Amsterdam, The Netherlands: University of Amsterdam. URL: `http://www.dutchnaoteam.nl/wp-content/uploads/2017/08/honours-project-ball.pdf`. published.

Lagrand, Caitlin et al. (2016). *Team Qualification Document for RoboCup 2017, Nagoya, Japan*. Tech. rep. Science Park 904, Amsterdam, The Netherlands: University of Amsterdam. URL: `http://www.dutchnaoteam.nl/wp-content/uploads/2016/11/Team_Qualification_Document_2017.pdf`. published.

Lagrand, Caitlin G. (2017). *Learning a Robot to Score a Penalty - Minimal Reward Reinforcement Learning*. Bachelor thesis, Universiteit van Amsterdam. URL: `http://www.dutchnaoteam.nl/wp-content/uploads/2017/08/Caitlin-Lagrand-Learning-a-Robot-to-Score-a-Penalty.pdf`. published.

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

Matthias Hofmann Ingmar Schwarz, Oliver Urbann Fabian Rensen Aaron Larisch Arne Moos Janine Hemmers (2016). *Team report Nao Devils 2016*. URL: `https://github.com/NaoDevils/CodeRelease2016/blob/master/TeamReportNaoDevils2016.pdf`. published.

Menashe, Jacob et al. (2017). "Fast and Precise Black and White Ball Detection for RoboCup Soccer". In: *Robot Soccer World Cup XXI*.

Pot, Emmanuel et al. (2009). "Choregraphe: a graphical tool for humanoid robot programming". In: *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*. IEEE, pp. 46–51.

Röfer, Thomas et al. (2016). *B-Human Team Report and Code Release 2016*. Only available online: `http://www.b-human.de/downloads/publications/2016/coderelease2016.pdf`.

Siciliano, Bruno and Oussama Khatib (2016). *Springer Handbook of Robotics.* Vol. 2nd edition. Gale Virtual Reference Library. pg. 307-328. Springer. ISBN: 9783319325507. URL: `http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1260895&site=ehost-live`.