University of Amsterdam
Faculty of Science
The Netherlands

Dutch Nao Team

Technical Report

Authors:
Pim Heeman
Thomas Wiggers
Hidde Lekanne gezegd Deprez
Wouter Zwerink

Supervisor:
Arnoud Visser

December 31, 2020

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 vision, sound and framework-specific things. For vision, line detection has been improved by the use of scanlines while a new usage of line detection has been found in center circle detection. To better adopt new match rules, whistle detection have been reworked to detect whistles during matches. The internal message system between modules has been further refined to eliminate discrepancies on data between different modules by the addition of timestamps to always use the latest message. The toolchain has been modernised to make better use of the V6 robots with a 64-bit processor. In conclusion, a start has been made on reworking the behaviour engine to make use of a state machine.

The team has also participated in the Robotic Hamburg Open Workshop (RoHOW) and has been qualified for the 24th edition of the RoboCup, in Bordeaux. Finally, several demonstrations and lectures on robotics and AI have been given at both schools and companies.

Contents

1 Introduction

The Dutch Nao Team consists of students of different disciplines studying various degrees at the University of Amsterdam: five bachelor's students from Artificial Intelligence, Computer Science and Mathematics, four master's students from Artificial Intelligence, Logic and Computational Science and one alumni. 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. Last year, a start has been made to optimise our framework for V6 by stopping supporting the V5 robots, which was further worked on this year. The V5 robots lack the computational power required and differ in the camera quality and software interfaces after which for the sake of simplifying the framework was decided to only support the newer V6 robots. This has allowed us to implement techniques that were previously too computationally heavy.

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. This year has been focused on supporting the full 64-bitness of the processor by upgrading the compiler stack.

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 Timestamp alignment

In our last report, the principles behind the communication system has been written down. However, it was discovered that there were synchronisation issues within this system causing errors. The communication process looked as follow: at first, a module produces a representation object holding some information. A pointer to this object is then copied over to different input queues of modules asking for information from the sending module, to get a list of messages. In the last step, the dependent module pops the most recent message at certain times, when the module is given execution time and has no other work to do.

However, this last step can cause issues if the sending module is executed more often compared to the dependent modules. Let's say there are 3 modules: module 1 is on the fast thread and module 2 and 3 are on the slow thread. Additionally, module 2 is dependent on module 1 and module 3 is dependent on module 1 and 2. The time between module 2 and 3 can be larger than the time between a repeated execution of module 1. If that is the case, then module 2 and 3 are operating with different messages from module 1. However, module 3 doesn't know whether that is the case. This is undesirable as it can lead to unnecessary errors.
A practical example is the module line detection which calculates a top down view of the lines with the output from the odometry module, with the line detection being on the slower thread. Later, in the localisation module – on the same thread as the line detection – some of those lines are projected back onto the robot camera feed. The line detection module and the localisation module can be working with a different odometry message causing discrepancies.

To solve his problem, a method has been chosen where at the beginning of each group cycle the time is recorded which can be used with the timestamps of their group cycle to choose the most relevant input messages. This process is illustrated in Figure 1 below.

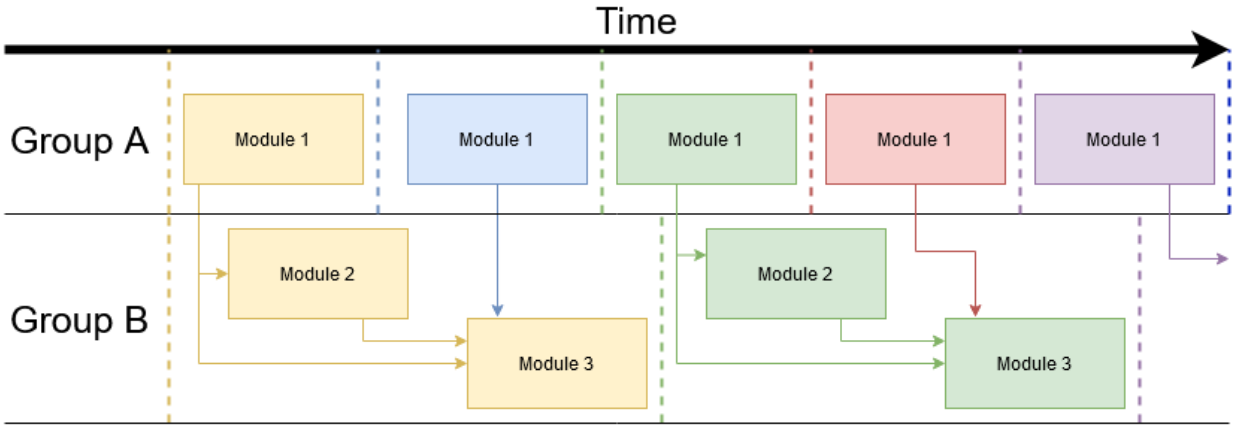

Figure 1: Aligned messages. The vertical dotted lines indicate the beginning of a cycle for a thread. With timestamp alignment, only modules of the same colour accept communication from each other. This is dependent on the closeness of the beginning of the cycle.

3.2 64-bits compilation

The operating system flashed on the V6 is NAOqi OS, which is an operating system based on Gentoo having the Linux kernel. On this operating system, the vendor-provided userland tools are all 32-bits, although the kernel is capable of executing 64-bits applications. Due to this discrepancy, all used userland tools needs to be regenerated to be 64-bits compatible, including a toolchain to compile the framework code.

For compilation of the framework, the major assumption was made that the method to generate code for use on the robots works equally good for use on other devices. A consequence of this is that devices should be equally compatible to each other in terms of supported hardware instructions and library versions. Due to the difficulty of upgrading with these constraints, the compiler version has been fixed at Clang 3.8.0, released on 8 March 2016, which prevented better optimisations provided by newer compiler versions. Additionally, 3rd party libraries, like The GNU C Library, OpenCV and libjpeg, have not been recompiled, which makes us unable to use newer functionality provided by those new versions. To overcome these problems, a different toolchains have been be created for each compilation target this year.
As a consequence, all libraries had to be rebuilt for each target. A portable shell script has been written to accomplish this process of generating binaries; to generate a compatible toolchain, now only the name of the target needs to be passed to the script without manual intervention. This script has been modularized to ease the work need to be done for using a new library or updating a library.

In the last report, remote compilation of code on a workstation to gain a speed-up on compilation has been mentioned. This script has not made compatible since this toolchain upgrade, which result in this feature being no longer available.

3.3 Behaviour engine

We are replacing our score based behaviour engine with one inspired by CABSL[1] from the B-Human team. The previous engine makes use of an utility system to determine which action is the most useful to perform at a given time. A drawback of this approach is that in order to extend it, one must understand the utility of all other actions in order to properly balance a new action. This makes extending the behaviour engine difficult. Because of this problem, a new framework based on CABSL has been chosen. This approach combines an execution graph with state machines.

In this framework, each state only has to consider the states that can reach it, and the states it can reach. This is more user friendly compared to balancing utility functions. A utility was also created to plot the graph and state machine connections using [5]. The B-Human implementation makes use of a lot of C macro's to enable its CABSL language syntax. We have chosen instead to implement the system using C++ classes. This increases boilerplate code, but is aimed at improving the maintainability of our framework by making it easier to understand, improve and expand for future team members.

[1] `https://github.com/bhuman/CABSL`

4 Vision

4.1 Scanlines

In the previous report we discussed our implementation of a line detector using a line segment detector. While the results were very good for nearby lines, this method proved unable to capture distant lines accurately. The cause for this lies in the downsampling, which resulted in a severe loss of detail. The further lines are, the thinner they are in the image. Due to this the downsampling could break up lines or remove them from the image completely. Unfortunately, the downsampling could not be removed as line detection on the full image would take too long with the limited hardware of the NAO.

In order to retain detail while still ignoring as many pixels as possible, a new view into the image has been implemented on which line detection and other visual tasks can be performed. The aim of this method known as scanlines is to dynamically alter how many pixels are viewed in each row of the image, based on the angle of the robots head in relation to the ground. To further speed up computations, a horizon is first computed. The horizon is defined as the row in the image that is the length of the fields diagonal away from the robot. Anything above the horizon can be discarded, as this should not contain lines when the robot is standing within the field. Next, the distance between the robot and the pixel in the middle of the bottom row of the image. The thickness in pixels of a field line at the horizon and this lowest row is computed from which the thickness of field lines are estimated for all rows by interpolation. The scanlines are then simply horizontal and vertical lines placed such that field lines can never be missed, based on their thickness in the image. The method is visualised in Figure 2.

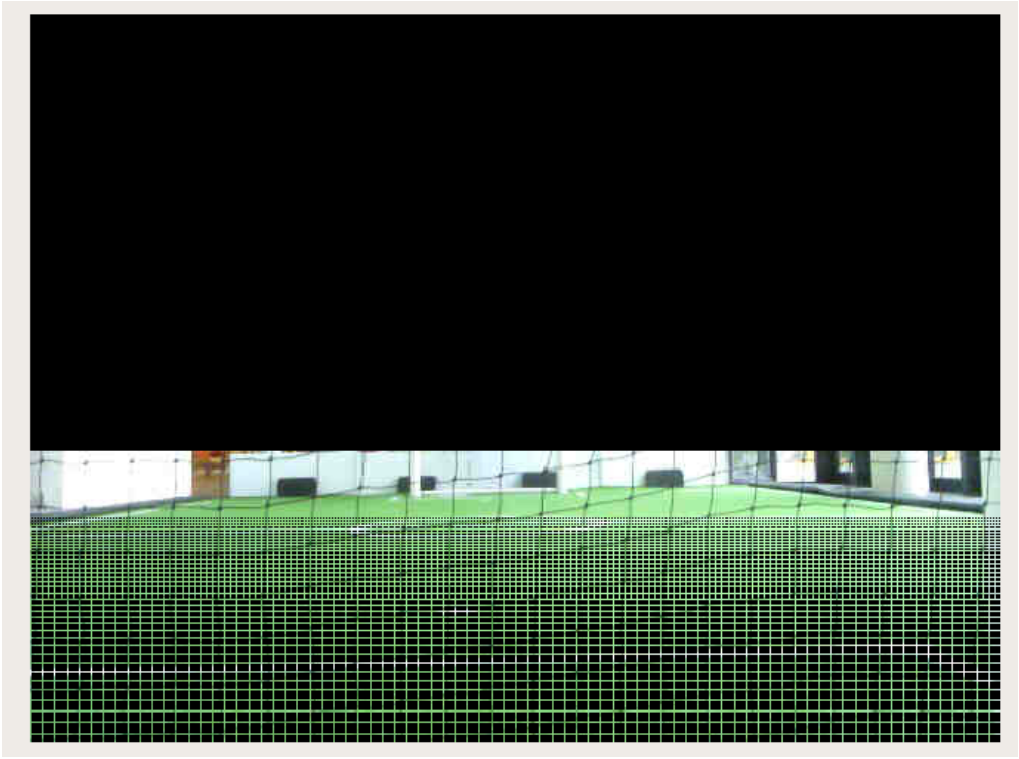

Figure 2: A visualisation of the scanlines method on the upper camera of the robot. All pixels except those within the scanlines are blackened.

4.2 Center circle detection

Localisation is an important aspect in playing a football match, as the robot needs to be directed towards the goal and the robot need to know where the goal is. With this in mind, it is important for the robot to be able to re-localise during matches, as after some time small errors in the odometry of the robot will accumulate. This results in a decreasing certainty about the position when relying on a known starting position and the odometry to estimate the robots location.
To re-localise, keypoints from the field can be recognised. This year, detection of the center circle relative to the robot has been added which enables the possibility of knowing the robot's position, as the center circle position is fixed and known.

The algorithm used for detection enumerates over all points sampled from all the lines detected by the line detection. For each possible pair a circle is created by treating the points as two opposite circle edges. Then the circle is verified to have sensible dimensions and the number of the mentioned points that are the circle are counted. The circle with the highest amount of points is chosen, provided that it has a minimum amount of points.
This complexity of the algorithm is in the order $\mathcal{O}(n^3)$, as it loops over all pairs of points and calculates the distance to the circle for all points. However, the number of points is small enough such that reducing the computational complexity by random sampling increases the execution time.

The algorithm gives the position of the center circle, which allows the robot to find its own position up to the correct half of the field. Therefore the results can only be interpreted when the robot knows on which half of the field it currently is. Due to this, the results are not used this year for re-localising. The results are however visualised in the interface to give some insight into its correctness, as the human viewer can easily correct for the position up to the correct half.

5 Tools

5.1 Simulation to real image translation

Lately there has been interest in developing neural network in order to perform vision tasks. These neural networks require large datasets with diversity in its contents. For conventional tasks such a ball detection, robot detection and goalpost detection, this problem has been addressed in the SPL-league. There are numerous labelled datasets publicly hosted on ImageTagger from the Bit-Bots team[6]. However, these datasets have limitations. First of all, a new dataset needs to be created when the conditions of game change, for example lighting conditions. Secondly, they all have to be manually annotated which limits the type of annotations SPL teams are willing to make. New frontiers like real-time semantic segmentation mentioned in [7] have no publicly available datasets.

To solve these problems, a generative adversarial network was trained on the task of image translation. This neural net is based on MUNIT architecture [8] and is able to translate simulation images to realistic looking images with reasonable success. This enables us to reconstruct scenes in simulation and render translate them to a new context, obtaining new labels with new context without having to perform manual labelling. Further implementation details are documented in [9].

6 Sound

6.1 Whistle detection split

The previous report contains an explanation about how whistle detection had been implemented to detect the start of a game directly after a whistle sound has been sent out, instead of waiting 15 seconds for this signal to be send over the network.
This detection has been implemented tightly-coupled with the microphone management. However due to changes in the rules, further whistle sounds can be made by the referee during the match to indicate changes to the game state, which was not the case before. To accommodate this feature, this coupling has been loosened, enabling the possibility to listen later in matches to whistles too.

7 Results

7.1 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 29th of November until the 1st of December 2019. During the event, test games are played, algorithms and ideas about challenges are shared, and lots of coding is done. Several issues like using parallelisation, implementing behaviours, ball detection and image segmentation were discussed. Overall, the atmosphere is relaxed without the 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.

7.2 RoboCup Bordeaux

The Dutch Nao Team qualified for the RoboCup 2020 in Bordeaux with a video[4] and a qualification paper [10]. This competition was planned to hold from 23 to 29 June, but due to the restrictions mandated to stop spreading the SARS-CoV-2 virus, causing COVID-19, the competition was cancelled to be hold in 2020. The team's qualification will still be valid for the 24th edition, plannned to be hold in 2021 in Bordeaux.

7.3 Foundation

In 2016, the Dutch Nao Team started the foundation *Stichting Dutch Nao Team*, in order to be able to receive money from companies in a transparent way.
In 2019-2020, the board of the foundation consisted of the following members:

- Chair: Pieter Kronemeijer
- Vice-chair: Douwe van der Wal
- Secretary: Wouter Zwerink
- Treasurer: Thomas Wiggers

7.4 Public events

This year started with some events on monthly base. However, starting from March 2020, a partial nationwide lockdown had been declared, which causes all further events from March on to be cancelled.
The events held this year are listed below:

- 2 October 2019: Talk and demonstration for some retired engineers.
- 4 October 2019: Open lab session for bachelor's and master's students to introduce them to the Dutch Nao Team.
- 5 October 2019: Science Park open day, where the team reached out to the broader community by giving demo's at a stand.

[2] `https://rohow.de/2019/`
[3] `https://www.hulks.de/`
[4] `https://www.youtube.com/watch?v=SKawcYDEhjo`

- 15 October 2019: Demonstration for people from the Brain Corp company.
- 25 November 2019: Demonstration session for primary school children by showing them robots taking different poses and some videos of competitions.
- 19 December 2019: Workshop at DevNight 2020 for about six peoples where motions could be made using NAOqi for two robots.
- 13 February 2020: Open lab session for bachelor's and master's students to introduce them to the Dutch Nao Team.

Events cancelled due to the reasons mentioned above include a standing on a three-day event in Doesburg and a talk at the local rotary association in Bussum.

8 Contributions

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

- **Quinten Coltof**, who worked on center circle detection.
- **Hidde Lekanne gezegd Deprez**, who worked on timestamp alignment and image translation.
- **Pim Heeman**, who worked on 64-bits compilation.
- **Pieter Kronemeijer**, who worked on the behaviour engine.
- **Ole Nissen**, who worked on the whistle detection split.
- **Thomas Wiggers**, who worked on the behaviour engine.
- **Wouter Zwerink**, who worked on scanlines and timestamps.

9 Conclusion

This year, several new steps in the direction of vision, sound and framework-specific things has been taken, to improve the code running on the robots to be played during football matches. On some fields, the work has been started this year, but is for the next year to be finished. To show the public about our activities, several talks and demonstrations have been held. Due to COVID-19, most events in 2020 have been cancelled. Further, COVID-19 causes the RoboCup, the general football competition, to be cancelled. Our qualification for this 42th edition of the RoboCup stays valid, so we are planning to attend next year.

References

[1] Thomas Wiggers et al. *Dutch Nao Team - Technical Report*. Tech. rep. Universiteit van Amsterdam, Jan. 15, 2020. URL: `https://www.dutchnaoteam.nl/wp-content/uploads/2020/01/dnt_techreport_2019.pdf`. published (cit. on p. 2).

[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, Dec. 31, 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] John Ellson et al. "Graphviz and Dynagraph — Static and Dynamic Graph Drawing Tools". In: *Graph Drawing Software*. Springer-Verlag, 2003, pp. 127–148 (cit. on p. 5).

[6] Niklas Fiedler, Marc Bestmann, and Norman Hendrich. "ImageTagger: An Open Source Online Platform for Collaborative Image Labeling". In: *RoboCup 2018: Robot World Cup XXII*. Springer. 2018 (cit. on p. 7).

[7] Bernd Poppinga and Tim Laue. "JET-Net: real-time object detection for mobile robots". In: *Robot World Cup*. Springer. 2019, pp. 227–240 (cit. on p. 7).

[8] Xun Huang et al. "Multimodal unsupervised image-to-image translation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 172–189 (cit. on p. 7).

[9] Hidde Lekanne gezegd Deprez. *Enhancing simulation images with GANs*. Bachelor thesis, Universiteit van Amsterdam. July 3, 2020. URL: https://staff.fnwi.uva.nl/a.visser/education/bachelorAI/thesis_hidde_lekanne_deprez.pdf. published (cit. on p. 7).

[10] 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, Jan. 22, 2020. URL: https://www.dutchnaoteam.nl/wp-content/uploads/2020/01/TeamQualificationDocument2020.pdf. published (cit. on p. 9).