Application for Participation

Nao-Team HTWK

December 2010

1 Statement of commitment

We, the Nao-Team HTWK are commited to participate in the RoboCup 2011 SPL.

2 Team members and affiliations

Rico Tilgner, post-graduate, HTWK Leipzig
Thomas Reinhardt, post-graduate, HTWK Leipzig
Daniel Borkmann, post-graduate, HTWK Leipzig
Tobias Kalbitz, post-graduate, HTWK Leipzig
Stefan Seering, post-graduate, HTWK Leipzig
Christoph Vitz, undergraduate, HTWK Leipzig
Robert Fritzsche, post-graduate, HTWK Leipzig
Sandra Unger, undergraduate, HTWK Leipzig
Katja Zeißler, undergraduate, HTWK Leipzig

3 Notable work and fields of interest

3.1 Segmentation

The segmentation of Nao's camera image and identification of the field and objects on it is an essential part of playing soccer. The biggest problem for most color-table based methods are the inability to cope with changing light conditions and the need to generate the color-table, which can be very time consuming. Changing lighting conditions (e.g. between daylight and artificial light which is common at the German Open) make it impossible to classify objects solely based on their color. Also, differing ball colors, like at Robocup 2010, pose another problem for purely color based methods.
Therefore, a real-time capable segmentation with no need for calibration would be advantageous. By applying the knowledge of the objects' shapes we developed a segmentation algorithm that can handle changing light conditions and colors robustly without the need for prior calibration.

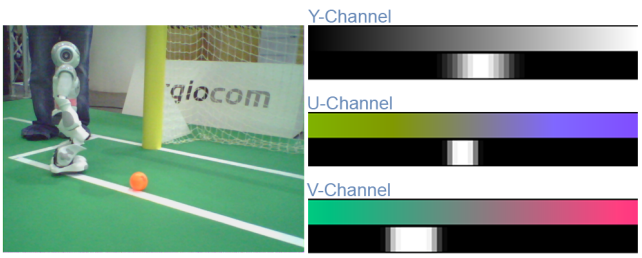

Figure 1: An automatically generated color-table for the carpet based solely on the image on the left.

The core of the algorithm are small histogram based color-tables (see Fig. 1) that are generated automatically whilst segmenting each single image. These color-tables are built up iteratively after

each of the objects is identified by its shape or approximate color. I.e., a ball is defined as an object whose color has a value of U>0 and its contour to the ground and lines (that have been segmented in an earlier step) has an approximately round shape.

Phase	Time
Floor Recognition	2 ms
Line Recognition	8 ms
Ball Recognition	5 ms
Goal Recognition	9 ms

Table 1: CPU time for each phase of the segmentation

The method works reliably and in real-time (see Table 1). It has been used at German Open 2010 and Robocup 2010 where it won the Open Challenge. The work will be available as a master thesis and is also planned to be published as a paper.
Currently, we are working on a visual obstacle detection that will be integrated into the above algorithm.

3.2 Localization

Our current localization is based on a combination of RANSAC and particle filters. First, the line segments determined by the segmentation are projected onto the ground. For this, the body and head angles of the robot are used. Following, lines and the center circle can be detected and fed into a particle filter which generates a position estimate within a few milliseconds.
However, as the method relies on correct body and head angles for the projection, it fails if the camera physically moves, which often happens when the robot falls.
We currently develop a localization method that does not depend on any camera angles.

3.3 Walking engine

Until the beginning of 2010 we used closed-loop walking motions evolved through a genetic algorithm. These motions were fast but not omni-directional (eventhough walking along a curve was possible). This was a big disadvantage at the German Open 2010, so we decided to develop a completely different walking engine. As of Robocup 2010, our walking engine is based on a parameterizable walking model similar to [1] and is supported by a newly developed balancing algorithm. The big advantage of this system is full omni-directional capability and the ability to make fast direction changes whilst still being very stable.
The new walking engine was tuned for stability and speed manually and achieved forward speeds in excess of 300 mm/s. We plan to combine the genetic algorithm and automatic evaluation system used in our old walking engine with the new one and hope to achieve both, more speed and even more stability until German Open 2011.
Furthermore, we also plan on publishing the full system once we have finished the integration.

3.4 NaoControl

NaoControl is a monitoring program for our robots. It provides a virtual playing field showing the robot's and the ball's location. The Naos send their own and the ball's supposed position and an estimated localization quality to the program. With this, we can easily control whether the localization is fine or not. Also, the robot's rotation and field of vision is displayed.

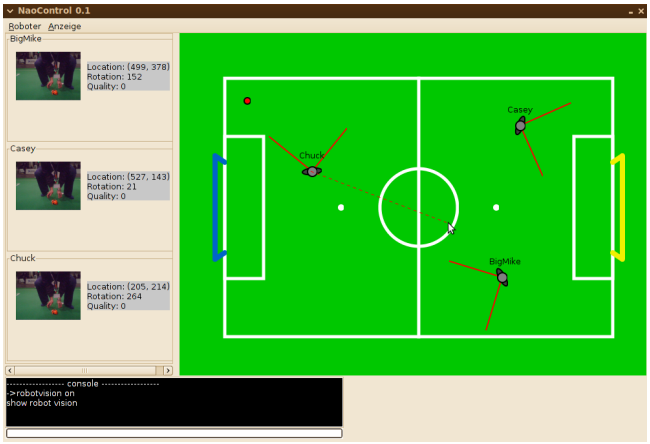

Figure 2: Screenshot of our NaoControl application.

Next to this, it is possible to show the actual images of the Naos' webcams. Those can be the real pictures or the segmented ones. We are able to send commands to the robots for testing them. The virtual playing field and its lines can be well customized, so new dimensions cause no problems. NaoControl is yet still in progress. In the near future it will be enhanced with simulation tasks. New playing strategies will be developed and tested with the assistance of NaoControl. For this purpose it provides simulated robot-behavior.

3.5 Motion editor

The NaoMotionEditor is a replacement of Aldebaran's Choregraphe. The main purpose is to capture key frames directly from the robot, manipulate them and interpolate with a Groovy scripting engine between them. There exist already predefined Groovy scripts which define a linear and a smooth interpolation between two frames. These captured motions are saved in a XML file and can be later exported to a team dependent file format. To manipulate frames exist a variety of predefined operations like duplicate, mirror, move and show frames. The architecture of the editor is designed to add new functionality fast, so new requirements and features can be added on demand.

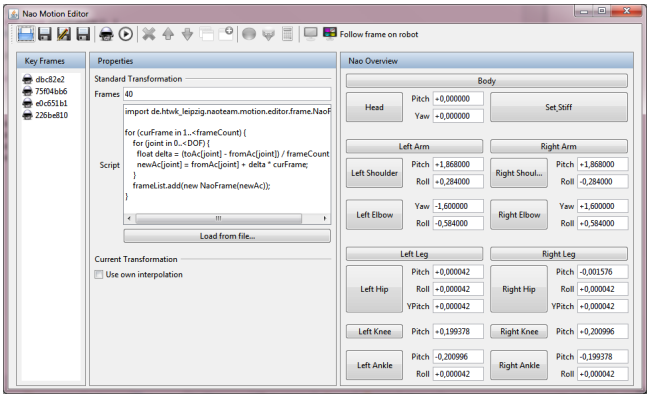

Figure 3: Example of a goalkeeper motion.

3.6 Architecture

3.6.1 NIO Framework

Our NIO (Nao Input Output) Framework is an independent piece of software that runs on Nao robots and extends the Aldebaran Robotics NaoQI framework.

The motivations for creating our own framework:

- Inconsistency of NaoQI's API
- Very limited debugging capabilities of NaoQI framework
- No need for a time intensive NaoQI restart after changes to parts of the software (e.g. motions, strategy)
- No thread safety of certain NaoQI calls
- Lots of NaoQI functionality we actually don't need
- Possibility of writing plain C instead of C++ code

The basic functionality of our NIO Framework is defined by a Unix Domain Socket client server pair. We have built a simple C++ module for the NaoQI framework that exports a subset of the NaoQI calls through the socket to the requesting process. This module is compiled as a shared library and will be linked against our actual kernel (core executable of our framework). Our exported API calls are kept very simple and performant, the subset is small and threadsafe. On the other hand, NIO consists of a series of subsystems. Each subsystem is generally independent of the others and serves one single aspect.

3.6.2 Naonet device driver

Our Nao robots do not communicate via IP-based traffic. Statically assigning IP addresses and netmasks can lead to misconfigurations during immediate competition setups of Naos or reboots of robots due to system failures. Therefore, we implemented a zero-configuration Layer 2 Ethernet protocol that runs within the Linux kernel by using the protocol hook *dev_add_pack* and Kernel FIFOs for queueing RX as well as TX packets. Communication happens via device files. A naonet communication manager within Userland controls the transmission and dispatch of messages. The receive path is therefore event-triggered. Several subsystems of NIO can register event-hooks provided as function callbacks to naonet's event manager. Depending on the specific event and destination (e.g. striker, defender, goalkeeper) all registered event-hooks will be triggered. The registration of hooks is priorized, so that high-priorized callbacks can stop the dispatch of messages for low-priorized callbacks. To sum up, we hereby have the possibility to notify other robots via event-driven mechanisms. This feature has initially been used during the RoboCup 2010 in Singapore and showed reliable communication results.

3.6.3 Memcpy improvements

Since memcpy seems to be a real bottleneck for video processing on Naos, we did some more investigation into optimizing its performance for larger chunks that need to be copied locally. AMD's V4L webcam driver for Geode seems to have serious driver issues with memory mapped I/O for video frames, so that we are forced to use Aldebarans NaoQI routines and copy video frames into a shared memory for NIO. We therefore measured the shipped memcpy function (libc) in combination with Geodes Time Stamp Counter processor registers (RDTSC) in order to gain current transfer rates. The investigation showed transfer rates of approximately 154 MB/s. Architecture dependent compile flags and other GCC optimization options did not result in significantly better transfer rates. Since AMD's Geode has built-in MMX technology, we have developed our own userspace memcpy, which exploits the provided MMX instruction set. We could improve memcpy from 154 MB/s to 675 MB/s, which has an extensive impact in performance for frame transferal into our shared memory.

3.6.4 Wiimote control

We have developed a Wiimote remote-controlled Nao movement interface for several testing purposes. This enables us to play against our developed Nao game strategy or to efficiently test new motions, which for instance have been set up by our own motion editor. The Wiimote is conneted via Bluetooth socket to a Bluetooth-enabled host machine that is within the same subnet as the Nao that will be controlled remotely. On the Nao-side a server application listens for incoming instructions that are sent from the host machine. With the help of our software, the host machine can even route multiple Wiimote connections to various Nao robots.

3.6.5 JVM communication and performace evaluation on AMD Geode

To improve development speed and lower the initial training of new team members we have researched and developed a proof of concept implementation of our NIO framework in Java.

Method	Time in ms
C-Server	0.1051
Oracle Java 1.6.20	0.1380
JamVM (no JIT)	0.7320

Table 2: Time for two TCP/IP function callbacks (1 TX, 1 RX).

Our tests in JVM and JIT speed indicated that Java is reasonably fast for strategy development and testing of prototype code. We plan to port our complete code base which is responsible for the strategy in the next few month to Java. The communication will be done by a TCP/IP connection for exchanging data from NaoQI to NIO and vice versa. Furthermore, we plan to generate the communication code layer by a program and evaluate the use of alternative communication channels like POSIX pipes and direct mapped memory which is also supported by JNI.

4 Publications and media presence

Increasing the public awareness of robotics and AI development is one of the goals of Nao-Team HTWK. To accomplish this, we try to present our work in symposiums as well as in local public media.

- The "Mainzer Wissenschaftsgespräche" (Scientific Symposium of Mainz, Germany) is a two day symposium hosted by the "Akademie der Wissenschaften und der Literatur Mainz" (Academy of the Sciences and Literature Mainz). In September 2010 the Nao-Team HTWK was given the honor to hold a talk at the symposium.
- Together with the HTWK Leipzig, the Nao-Team HTWK hosted an exhibition stand at the CeBIT in Hannover during the last two years. At the computer expo the team members showcased their accomplishments in bipedal movement and machine vision.
- With two appreances at MDR (Mitteldeutscher Rundfunk, German TV Channel) the Nao-Team HTWK was able to reach a broad audience and increase the public awareness of computer science, especially robotics and AI research.

References

[1] Sven Behnke. Online trajectory generation for omnidirectional biped walking. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1597 – 1603, May 2006.