Contribution Poster - RoboCup 2019

RobotING@TUHH e.V., Hamburg University of Technology

HULKs
RoboCup SPL Team • TU Hamburg, Germany

Introduction

This document serves as partial fulfillment of the rule book requirements for RoboCup 2019. For this purpose we present a sample of our contributions and research conducted in preparation to RoboCup 2019.

Vision: Robot Detection

In the season of 2018 the detection of other robots was solely based on sonar which limits the detection capabilities of robots located outside the robot's vicinity. Therefore, we implemented a visual robot detection. As robots are the only objects on the field containing a significant amount of horizontal edges we based the robot detection on counting vertical segments. Consecutive vertical non-field-color segments provided by the image segmenter are counted downwards starting below the field border. The end of the last segment in a chain of vertical non-field-color segments is considered as a seed. Using camera projection and a sliding window the best robot candidate is selected for a given seed and accepted if enough horizontal edges are present in the window. The center of the detected robot over ground is calculated using projection and provided to the team obstacle filter.

Vision: Chromaticity Field Color Detection

A new way to classify pixels and image segments into field and non-field was introduced. The analysis of multiple manually segmented images from different events and situations showed that the green chromaticity is the most suitable color channel to separate field from non-field. The green chromaticity is defined as green divided by the sum of red, green and blue as shown below.

$$g = \frac{G}{R + G + B}$$

A simple threshold for green chromaticity in combination with thresholds for the red and blue chromaticity is used to classify pixels and segments. This approach shows better results in challenging lighting conditions compared to the previous approach utilizing clustering pixels in the Cb-Cr plane.

Vision: Box Candidates for Ball Detection

The ball candidate generation we used until 2018 in Montreal was based on image segments. The search for black and white segments turned out to be insufficient. Investigation of different color spaces revealed that the ball appears brighter in Cb or Cr color channel than the surrounding field. Our new candidate generation is based on an integral image constructed on Cb or Cr channel. Knowing which size the ball at a given pixel position would have the algorithm searches for these bright spots. A box in ball size sums up all values inside it to get the average value inside and compares this value to the average value of a box slightly larger as the ball size. As the ball appears brighter a comparison of these averaged values results in a rating whether this position is a possible candidate. The candidates generated will subsequently evaluated and classified using a CNN.

GPU and Toolchain

The new Nao V6 robots have a 64-bit processor and a GPU, but the NAOqi OS Userland still only supports 32-bit. So we revamped our sysroot/toolchain build process and are now building a complete toolchain out of a single clang build. With this compiler we compile the linux kernel headers, musl, compiler-rt, libunwind, libcxxabi and libcxx to produce a complete and working C++ toolchain. On top of that we build clang and Beignet into the toolchain to support OpenCL. It is possible to run OpenCV DNNs on the GPU with this toolchain.

OpenCV Neural Networks

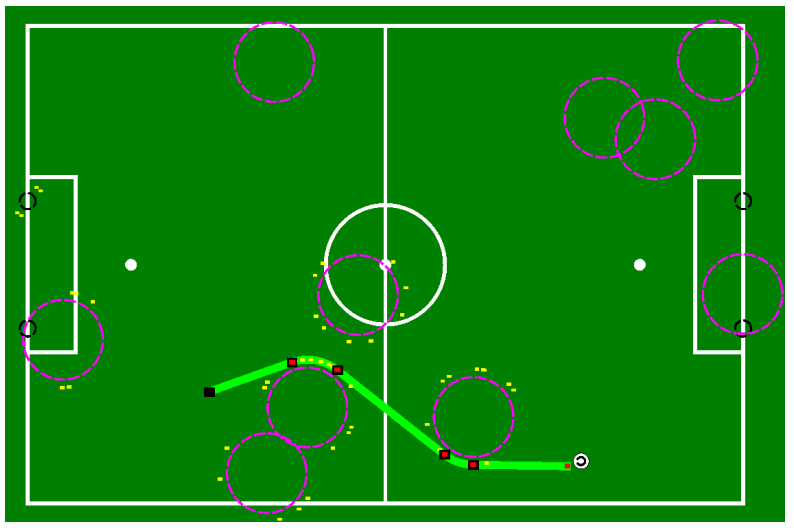
Last season our ball detection was based on a hardcoded neural network architecture whose weights were loaded from a JSON file. This made trying new architectures and deployment of networks very inconvenient. For this season we switched to using OpenCV's DNN module which simply takes a tensorflow protobuf file as its input and therefore allows for quick network replacement. An additional advantage is that OpenCV's DNN module comes with built-in OpenCL support which is compatible with the V6's GPU.

Joint Calibration

A research project was conducted focusing on observability of joint errors (with on-board cameras) and defining a set of minimal, optimal poses to extend existing research and to define a practical joint calibration method for NAO robots. Tests were performed under simulated conditions for a population of 5000 joint error configurations with noise modelling for joint encoders and cameras. The solver is an extension of the Levenberg-Maquardt algorithm using many random start parameters.

Brain: Path Planning

So far we have been using a potential field based approach for moving around detected obstacles (e.g. other robots). This leads to behavior where, even if one of our robots knows about an obstacle before starting to walk in its direction, it only starts to avoid it when it gets close to it. In order to make the obstacle avoidance more intelligent we implemented an A* based path planning algorithm. In our implementation we model obstacles as circles of different size. In order to then generate a graph which can be traversed by A* we create nodes from tangents between obstacles. This keeps the amount of nodes in the graph to a minimum and makes sure that the resulting path is optimal and kinematically meaningful. An example of a generated path is shown below.



Motion: Sit Down

For convenient carrying of robots between half-times and after games, we implemented a sit-down motion. This motion is being triggered by the GameController's Finished message. In case of a GameController mishap, there is a complementary sit-up motion being called so that the robot can continue playing.