System Design Project 2012

Milestone 2 Individual Report
Gediminas Liktaras (s0905195)

February 15, 2012

Goals

For the second milestone I have again concentrated on the vision component of the robot system. My goals were to create a functional, if not competent field state extraction procedure from the camera feed and provide methods of system's calibration.

Field State Extraction

Choosing the Approach

Before delving into implementation details, I have first perused the source code of the previous years' teams for ideas. At that point there were still three people working on the vision system and we decided to use OpenCV for image processing. Some teams from last year were using OpenCV, so it served as a recommendation.

Then we have decided to use the approach of 2011's team 9 to state extraction. It performed quite well empirically compared to the other teams and their approach was straightforward to implement, making it easier to keep everything simple. It was unfortunate that they had used python for their implementation - it meant that we are not be able to simply reuse the code.

State Extraction Algorithm

Save for the direction detection logic, I have implemented the entire state extraction procedure. It works as follows:

- Smooth the input frame to reduce the amount of noise, at the expense of details.
- Convert the frame to HSV color space.
- Threshold the frame based on predefined HSV value intervals to isolate each of the objects we seek: the ball and T's.
- Find connected blobs in resulting images.
- Filter blobs, whose dimensions exceed some fixed size boundries.
- Choose the "best" blob to represent the found object.
- Find the contour of the above blob.
- Compute the mass center of the blob - it represents the location of the corresponding object.
- Compute the direction using the mass center and the contour of the object blob.

The blobs are filtered by size to discard very small and very large blobs, which can be unambiguously identified as noise.

The ball can be isolated quite easily, since under normal playing conditions there aren't any other objects that have a similar color signature. It is also sensible to assume that there will be at most one ball, making blob selection trivial.

To pick the best blob to identify the robots' T's, the algorithm selects the blob with the largest area. This helps filter any noise that comes from exposed robot parts. The direction is then located by finding the farthest point on the blob's contour and drawing a direction from the mass center to the said point.

Performance and Problems

Unfortunately, the test bench was implemented only a few days before the second milestone and there was not enough time to collect the testsets, so there is no way to provide any quantifiable performance measures.

For what it's worth, it can be seen from empirical observations that the current approach works well in most cases, but is unable to cope with large variations in lighting. Upper-right corner on the main table is a particular example where the yellow T becomes invisible.

Also, the direction detection procedure is too simplistic and does not cope well with some worse lighting conditions and the yellow robot (at times the yellow T appears gray and easily blends with background, making the thresholded shape "bleed over" too much).

Furthermore, the system is very sensitive. Tables, the time of day and even different computers produce differently colored images, so the system has to be recalibrated every time live robot testing must be done. I have added a simple way to save and load vision system configurations at runtime via GUI, but it is still a pain.

Future Work

I will keep working on the vision system. I plan to make direction detection procedure more robust, add barell correction to account for the distortion of the camera lense and consider alternate ways to improve image thresholding.