User Guide

SDP Group 12

April 20, 2016

1 Starting the System

In order to start the robot:

- place a charged set of batteries in the back holder, and connect it to the Arduino power cable
- plug the RF stick in the computer, ensuring all top plates and the ball are on the pitch
- execute '`python main.py -p <PLAN> -1 <PATH> -c <COLOR> -g <GOAL>`', where,
  - `<PLAN>` is the plan to run (see section 1.1 below),
  - `<PATH>` is the device path of the RF stick (e.g. '`/dev/ttyACM0`'),
  - `<COLOR>` is the team color, which must be either '`blue`', '`b`', '`yellow`', or '`y`',
  - `<GOAL>` is controlled team's goal end, either '`left`' or `right`'.
- logging level and verbosity can be set with the following additional options:
  - '`--debug`' or '`-l debug`',
  - '`--info`' or '`-l info`',
  - '`--warn`' or '`-l warn`',
  - '`--error`' or '`-l error`'.

While the control program is running, the overall strategy and logging can be modified. Entering a plan name and pressing enter switches the running plan to the new plan that has been entered. Entering '`stop`' unsets the active plan and leaves the robot idle.

2 Planning

2.1 Plans

A small number of plans are available to run:

- '`move-grab`', to move to the ball and grab it.
- '`m1`', to do milestone 3, task 1 (receiving a pass).
- '`m2`', to do milestone 3, task 2 (receiving, turning and passing).
- '`m31`', to do milestone 3, task 3.1 (intercepting a pass).
- '`m32`', to do milestone 3, task 3.2 (defending the goal).

2.2 Calibrating the planner

The calculations for pitch geometry in the planner depend on calibrating certain constants in the planner code.

Name	File	Description
ROBOT_WIDTH, ROBOT_LENGTH	world.py	Dimensions of a standard robot.
BALL_WIDTH, BALL_LENGTH	world.py	Dimensions of the ball.
GOAL_WIDTH, GOAL_LENGTH	world.py	Dimensions of the goals.
GOAL_LOWER, GOAL_HIGHER	world.py	Y value for topmost and bottom-most points of goals.
[ROBOT]._receiving_area	main.py	Dimensions of area in which robot can grab ball.
ROTATION_THRESHOLD, FACING_ROTATION_THRESHOLD	models_attacker.py	Difference for angles in which robot can be considered to be facing a point.
GRAB_DISTANCE	models_attacker.py	Distance from robot's centre at which the robot should grab the ball.

Table 1: Planner Configurable Variables

3 Vision

3.1 Requirements

You'll need the following python packages to successfully run the vision:

`Polygon2` Polygon is a python package that handles polygonal shapes in 2D.

`argparse` Python command-line parsing library.

`pyserial` Python Serial Port Extension.

`numpy` Array processing for numbers, strings, records, and objects.

`openCV` OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

To install them run these commands in the terminal:

```
# pip install --user Polygon2==2.0.6
# pip install --user argparse==1.3.0
# pip install --user pyserial==2.7
# pip install --user numpy
```

You can also learn how to install openCV from the following link: `http://docs.opencv.org/2.4/doc/tutorials/introduction/linux_install/linux_install.html`

3.2 Usage

Before using the vision system, you can look at the vision feed by typing `xawtv` in the command prompt. This will launch only the vision feed, where you can experiment with the different settings.

In order to launch the vision system:

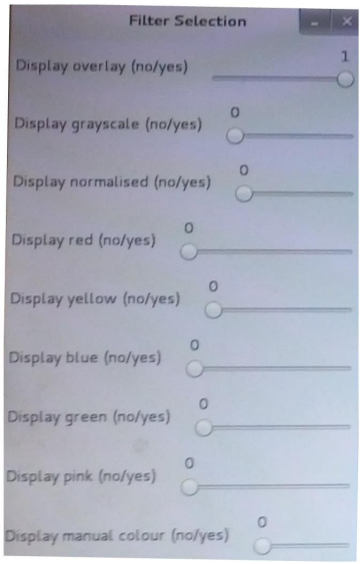
- Run the main vision file: `python vision.py`.
- At first, a window for the automatic colour calibration will pop out. You'll need to follow the instructions as printed in the terminal.
- The calibration goes through all the colours that distinguished by the vision system (`red`, `yellow`, `blue`, `green`, `pink`) and requires **multiple clicks for each** one of them to get their thresholds properly.
- You need to press the `q` key after each calibrated colour.
- If you want to skip the calibration you can simply press the `Esc` key and the vision will use the previously saved calibrations.
- The vision processing will then be launched.

3.3 Graphic Interface

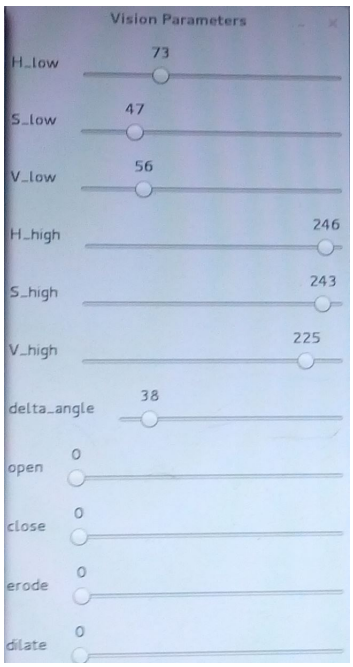
Starting the vision platform opens a window labelled `Filter output`, where the vision feed is displayed after all filters have been applied in order. This output is overlayed with icons representing the robots and the ball, if they are found (see Figure 1). Robots are represented by an inner circle of the team colour (`yellow` or `blue`), an outer circle identifying which of the two team members it is (`pink` or `green`) and an arrow oriented in the direction of the robot. The red ball is displayed by drawing a red circle around where it was detected.

Figure 1: Vision Frame with Overlay

There are two other windows containing several sliders for filters that that can be applied to the feed (see Figure 2a) and more advanced options for specific filters (see Figure 2b). The set of filters available includes masks for specific colours, a manually configurable colour mask, and can different effects applicable to the frame.



(a) Available Filters



(b) Advanced Filter Parameters

Figure 2: GUI Controls

When the vision is running, it outputs the coordinates, orientation and velocity of all found objects. The velocities are calculated based on the time difference between frames and movement relative to the previous frame. These objects are returned as a dictionary and passed to the planner. This is achieved by passing a callback method for updating the planner world model when initialising the vision module. This method is called on every vision frame parsed, and it ensures the planner has the latest data.

4 Hardware

4.1 Turning the robot on

Flip the switch towards the front of the robot, the robot will calibrate the grabbers by opening and closing them, and will then begin listening for commands.

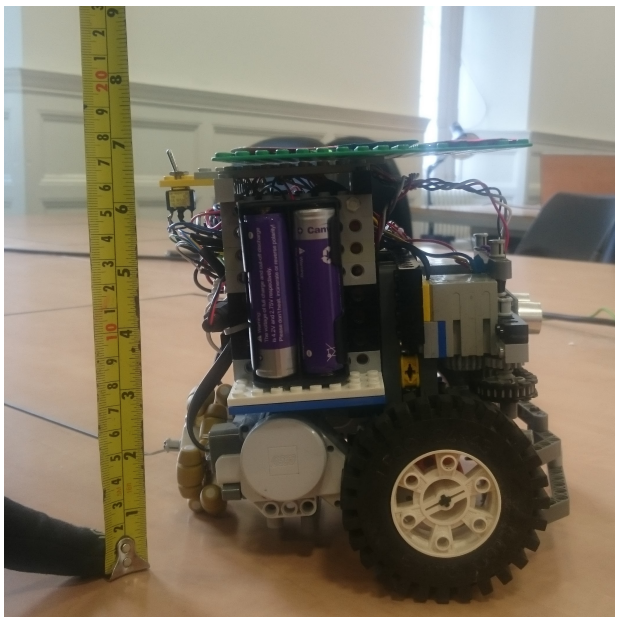
4.2 Changing the Power Cells

4.2.1 How to Change

Start by switching off the robot by flicking the yellow switch away from the robot (as in Figure 3a). The cells are held on by the externally accessible battery packs and can removed by gently pulling the batteries out of the packs (exactly as an AA cell is removed). New batteries should be inserted into the robot with the flat side against the springs. 2 sets of cells are provided so that one set can be charged while the robot is in use. The correctly inserted cells are visible in purple and silver in Figure 3a.

4.2.2 When to Change

The voltage from the battery packs is regulated and should give the robot consistent performance. The cells include a built in low voltage cut off, to avoid issues from varying battery



(a) Side View of Robot

(b) Charger With Correctly Oriented Cell

Figure 3: Hardware Pictures

output. When any cell reaches the cut off, the robot will stop completely. At this point all the cells should be changed.

4.3 Charging

The specialised charger (see Figure 3b) can be used to recharge the batteries. If using a different charger, for safety and performance use a charger made for charging 18650 cells and ensure correct polarity.

The batteries should be placed into the charger with the flat side against the spring arm of the charger, as shown. The charger shows a progress bar next to each cell.

5 Arduino Software

5.1 Sending commands

In normal operation the commands should be sent through the planner with the `RFComms.py` module.

For testing there is a `main_manual.py` file which can be used to send individual commands to the robot by following the prompts, first for which RF stick to use and then a choice of commands. To see the entire command set with descriptions hit 'h' then enter at the prompt.

5.2 Command Response

In response to a command the robot will first send an acknowledgment packet. It will then complete grab, release or kick commands, in order to leave the robot in a predictable state for the planner, then complete the given command. Move and turn commands are immediately dropped in order to use the new command.

5.3 Command Set

Command	Arguments	Effect
kick	distance in cm	kick the ball to the specified distance
grab	N/A	this will respond with "BC" for ball caught or "NC" for not caught.
release	N/A	this will respond with "grabbersOpen" to update the planner that it does not currently have the ball.
turn	angle in deg	turn counter clockwise if angle positive, clockwise if negative
move	distance in mm	move forward if distance is positive or backward if negative
ping	N/A	will respond with positions of motors according to rotary encoders, for debugging purposes

Table 2: Available Commands

5.3.1 Kick

The robot grabs the grabbers in hard in order to place the ball and push the piston back. It then releases the grabbers until they are open as far as possible without obstructing the wheels, it then kicks a time dependent on the distance required (ranging from 50-200ms in typical operation). Finally, it closes the grabbers fully for continued movement.

5.3.2 Grab

The robot closes the grabbers until they are fully closed or for 800ms whichever comes sooner. If the grabbers do not close fully it is determined there is a ball in the grabbers and "BC" is sent to the planner, otherwise "NC" is sent.

5.3.3 Release

The grabbers are released until they are open as far as possible without touching the wheels. "grabbers open" is then sent to the planner.

5.3.4 Turn

The robot turns a distance specified in degrees, this is non-blocking.

5.3.5 Move

The robot accelerates up to the calibrated speed and then maintains that speed until it has just enough time to decelerate to the desired distance. It will correct for drift in order to keep the robot moving in a straight line. If it finds something in its way with the ultrasound sensor while moving forward (positive distance) it will stop and respond "something in the way"

5.4 Troubleshooting

Symptom	Possible Cause	Possible Solution
Robot not moving, no acknowledgment	No power from battery packs	Ensure cells are:inserted correctly, charged, all present. Check wire from battery packs to power regulator board is connected. Check wire from power regulator board to motor board is connected.
Robot not moving-no acknowledgment	Robot only listening over USB.	Turn robot off and on using switch
Robot not moving-with acknowledgement	No power to motor board	Check connection from power regulator board in middle to motor board on left (black and red wire), check I2C cable from power regulator board to motor board (4 wire cable)
Robot moving but not kicking	Kicker signal cable disconnected	Check white 2 wire cable is going into power regulator board into Left side of furthest back slot on power regulator board.
Robot moving, but not straight, or not stopping	Cannot communicate with rotary encoders	Check for message which says "cannot communicate with encoder board". This implies the 4 wire I2C connection from power regulator board in middle to encoder board on right is disconnected, check that movement of each wheel and the grabbers can be detected by checking their position with the ping command, check that when commanded each motor is moving, if not the power cable to that motor is damaged.
Robot will not move forward	Something in the way of ultrasound sensor	Check for "something in the way" message to confirm cause, to solve first check if ultrasound sensor has been displaced or if something has got in way, as a last resort the sensor can be disabled altogether by unplugging it.

Table 3: Possible Problems and Solutions