PSU ECE Robotics 2 Class Project

Tony Muilenburg, Dawson Green

Computer vision and control

Baby Schrodinger Cat (CatBot)

Contents

[Project summary 1](#_Toc414902728)

[Robot cat 1](#_Toc414902729)

[Robot firmware code 1](#_Toc414902730)

[Robot control code 2](#_Toc414902731)

[Computer vision code 4](#_Toc414902732)

[Next steps 5](#_Toc414902733)

# Project summary

With the increase in speed of personal computers many things have become possible. One of the most interesting applications is computer vision. For this project we used computer vision to control a robot as part of a robot theater program Dr. Marek Perkowski putting together [(Link)](http://web.cecs.pdx.edu/~mperkows/CLASS_479/perception.html). Using a single overhead camera we were able to control a robot, tracking its position and angle, and have it autonomously drive toward an arbitrary target location. This report will describe the steps necessary to reproduce what we did, as well as cover the materials used.

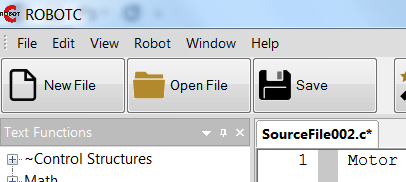
# Robot cat

Because another team was working on the Schrodinger cat robot, we decided to create use a Lego Mindstorms robot, and added a baby kitten on top of it. The base for the robot came from the 9797 First Lego League kit. This kit is described on Lego’s website at this link: [9797](https://education.lego.com/en-us/lego-education-product-database/mindstorms/9797-lego-mindstorms-education-base-set). Instructions for assembling the robot can be found here: [robot base instructions](https://education.lego.com/en-us/lesi/support/product-support/mindstorms-education-nxt/nxt-base-set-9797/building-instructions). To simplify tracking of the robot, a green circle, and a pink square were taped to the top to the baby kitten toy. Having two different colors allowed us to calculate the direction the cat was facing. The cat riding the robot is shown below before the colors were added.



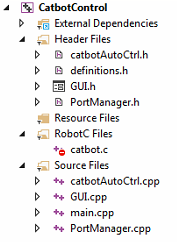
# Robot firmware code

Communication with the robot is achieved using Bluetooth. The Lego NXT robot is equipped with Bluetooth, though sending commands using a laptop is not possible using Lego’s standard code. To communicate with the NXT and set it up to accept commands, the firmware on the NXT brick is first overwritten using a program called RobotC. This program can be used for free for ten days on a computer, after which the program must be purchased for $50. Fortunately, once the firmware has been written to the robot, RobotC no longer needs to be used.

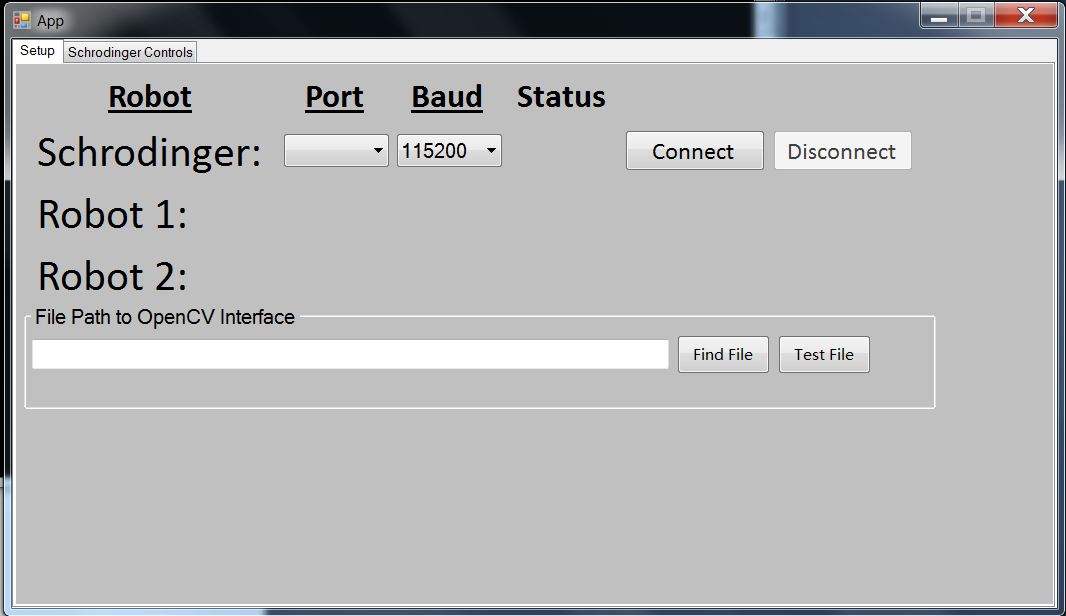


# Robot control code

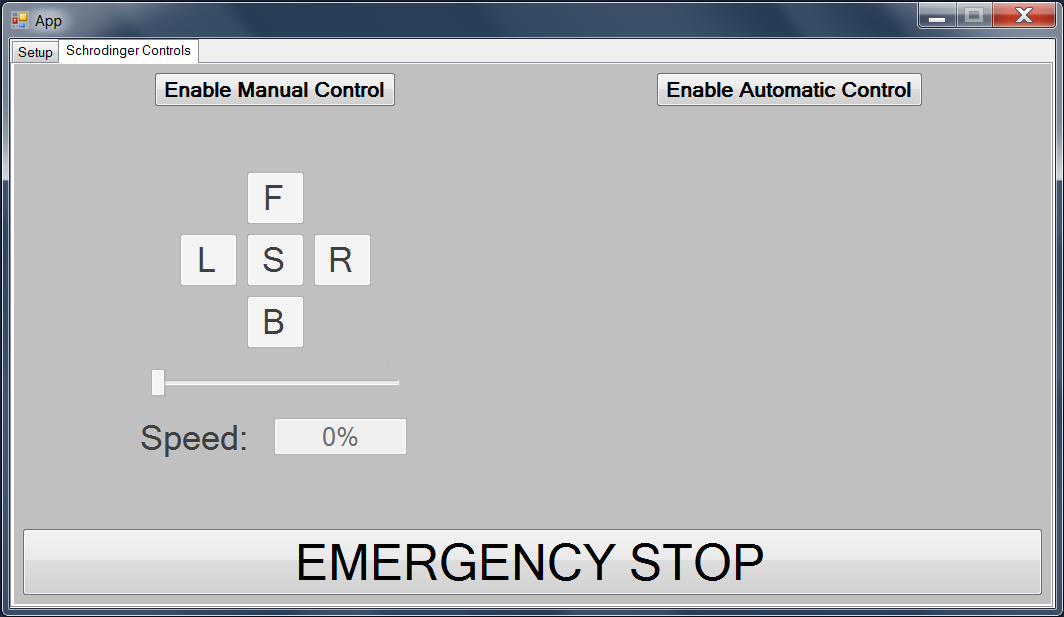
The code used to send commands to the robot was written in visual studio using visual c++, and is called catbot control. The following figure outlines the project structure:



The user interface is split up into two tabs. The first allows the user select the Bluetooth device that is to be used, and configure it. While some laptops come with integrated Bluetooth, the laptop used for this did not, so a USB dongle was added which added it. After selecting the Bluetooth port, the connect button will enable communication with the NXT equipped robot that has been flashed with the robotc firmware created for this project. The computer vision code uses a text file to pass location data to this robot control code. The path to the file the computer vision code generates should be added to the textbox using the find file button which pops up a file browse dialog window. The “Test File” button should then be used to ensure that the path is valid and file permissions are ok (permission to read the file is granted).



The second tab has buttons to allow for manual or automatic control. In manual mode, the robot can be turned, or orders can be given for it to go forward or backward. A speed control is available, as well as an emergency stop (see below).

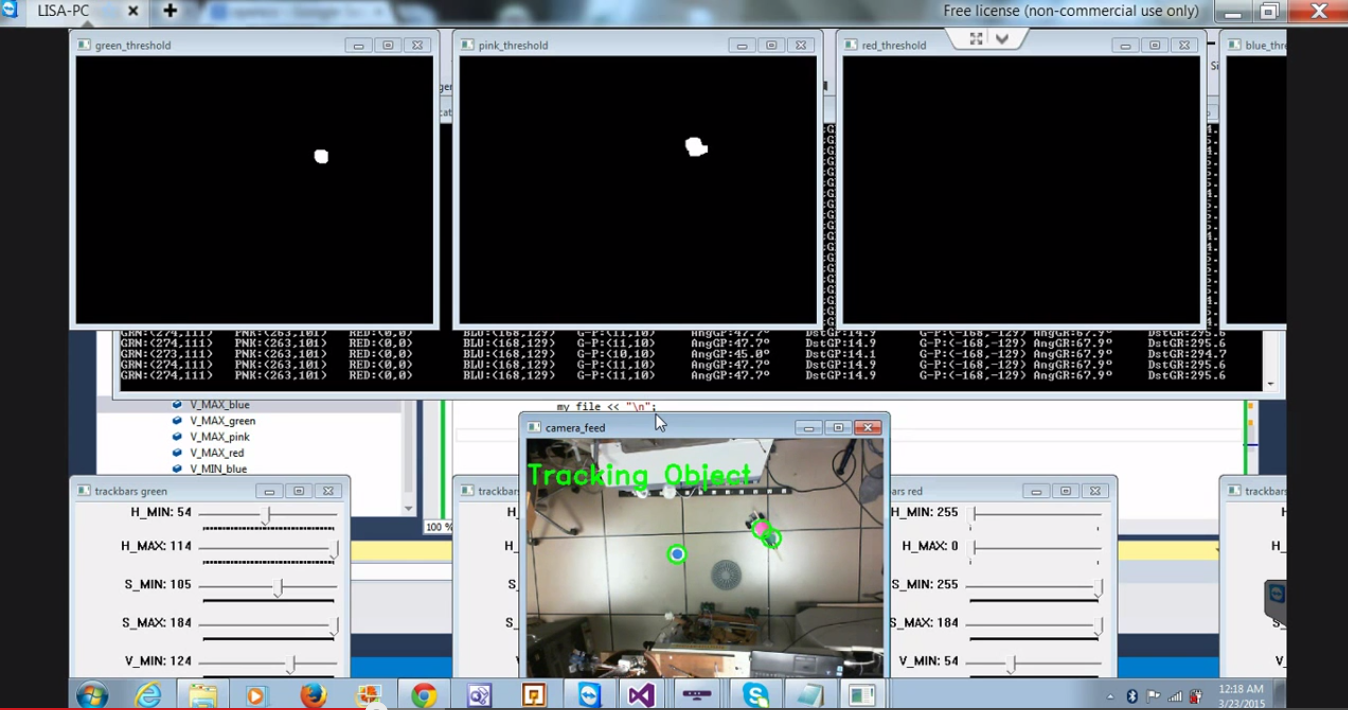


In automatic mode, the code will read the coordinate from the file the computer vision code generated, then a heading and distance will be generated from the robot to a blue dot, and commands will be send to have the robot turn and drive to the blue dot. The coordinates are written and read many times per second to ensure that the robot does not drive too far, or get off course.

The code for this project can be downloaded from this link: [CatBot control code](http://web.cecs.pdx.edu/~muilenta/catbot/CatbotControl.zip)

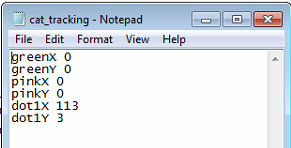
# Computer vision code

By default, the compute vision code (cat tracking opencv) opens many windows. The following figure show what screen looks like in debug mode.



The top four windows show the threshold image for a particular color. The colors red, blue, green and pink are tracked, though red is not currently in use (see the next steps section for more on this). In looking at the green and pink threshold windows, it is clear that these colors exist in the camera feed. All other colors are filtered out by specifying filter in HSV format ([follow this link for a description of HSV](http://en.wikipedia.org/wiki/HSL_and_HSV)). Along the bottom, there are four windows each of which have sliders that can be used to adjust the H, S, and V values and filter out all but the desired color to be tracked. The “camera feed” window in the middle of the lower portion of the screen shows what the camera sees along with superimposed circles that correspond to areas being tracked.

The code writes the coordinates of each of the discovered colors to a file called “cat\_tracking.txt” which is then read by the cat control code, which guides the cat to the desired location. The format of the file is given below:



# Next steps

While many of the next steps are outlined in Dr. Perkowski’s robot theatre script, some of the most obvious next steps include:

* Put together a third option for scripted control in addition to manual and automatic control of the baby Schrodinger cat.
* Enable tracking and control of other robots, including Schrodinger cat, Einstein, Newton, Bohr, Marie Curie, etc.
* Enable finer granularity for the fuzzy logic speed control
* Incorporate depth control using a Kinect camera
* Train and use a Haar cascade classifier to recognize robots, and robot orientation (color would no longer be needed for tracking).