

Final Written Report TowBot (Towing Robot)

Diego De La Hoz

EEL4665C: Intelligent Machine Design Lab

Instructors: Dr. A. Antonio Arroyo, Dr. Eric M. Schwartz

TAs: Andy Gray, Jake Easterling, Ralph Leyva

Table of Contents

Abstract	2
Introduction	3
Integrated Systems	4
Mobile Platform.....	6
Actuation.....	7
Electronics.....	8
Sensors.....	9
Behaviors	9
Special Sensor	10
Conclusion	12
Appendix.....	13

Abstract

In the frenzy of technological advancements, there has been a strong movement towards automation of tasks with the main goal of improving our conditions of living. With this in mind, the goal of our autonomous towing machine, is to be able to detect and actuate when an automotive has fail and needs a towing service; this task is to be performed autonomously. The application of this idea is to be extended to full-scale automotive and, to accomplish the vision in mind, we begin advancing this idea on small scale robots.

At the beginning of the semester, the project consisted of having two autonomous robots. Robot A would mimic a daily automotive that would drive around until it 'breaks down.' Once it breaks down, Robot A would send a signal to Robot B signaling its need of towing service. When Robot B receives such signal, he would then proceed to Robot's A location for assistance. Upon arrival, Robot B would tow Robot A to a specified destination. In recognizing the location of Robot A, there are various ways to accomplish it. There are two alternatives that were researched before selecting the best choice: 1) a god-camera that would recognize the entire environment and communicate with Robot B his relationship with Robot's A location so that Robot B could then travel to the specified location for towing and 2) a camera onboard Robot B that would scan the environment looking for Robot A and towing it once located. This project decided to use the god-camera. Using the god-camera involved wireless communication between the god-camera processor and Robot B's processor. Another function, and point of discussion, of the towing machine would be the mechanism for hooking or grabbing the broken-down robot. This could be accomplished using a claw of some sort, a magnetic attraction, Velcro, or some other possible solutions.

Because of time constraints and the difficulty of the project, the project was not completely finished. As a proof of concept, only Robot B was developed. Robot B is able to receive commands from what the god-camera sees and travel to desired location, where the broken down car would be. The god-camera processes all the images through the Raspberry Pi. The Pi uses OpenCV libraries to detect different colors. There are two different colors located on Robot B and there is a third color that points towards the desired location. In this manner, the Raspberry Pi processes all the information, analyzes it, and sends the proper command to Robot B to reach the desired location. In the future, Robot A could be developed to complete the project.

Introduction

In the real world, cars can be both a gain and a pain. They are used for transportation and can they have taken humans to places that were unthinkable. On the other hand, they are machines that break down. This break down could be because of mechanical car failure, road car accidents, or new car needing transportation. When one of this instance happens, there is a need for a towing truck, a truck driver, and the available time. Some of the challenges that arises when towing are communication failure in remote or busy locations, time delays from towing companies and truck drivers, and prompted unavailability. The towing process can be very inefficient. Therefore, a solution to increase efficiency is searched for. Although we cannot, nor do we want to, eliminate the towing truck, we could automate this process to make it more effective.

The proposed solution is to create an autonomous towing robots. Their purpose is to autonomously locate and aid cars that need towing. On a large scale, the goal would be to create an automated towing robots network that would aid broken cars as needed depending on the location of the broken car. Is this way, the autonomous towing robot closes to the broken car can readily aid it. On a small scale, the goal is to design and implement a robot and develop communication system in which robots can communicate. The project is divided into two phases: Phase I: Obstacle avoidance, motor control and Phase II: Image processing, communication.

Integrated Systems

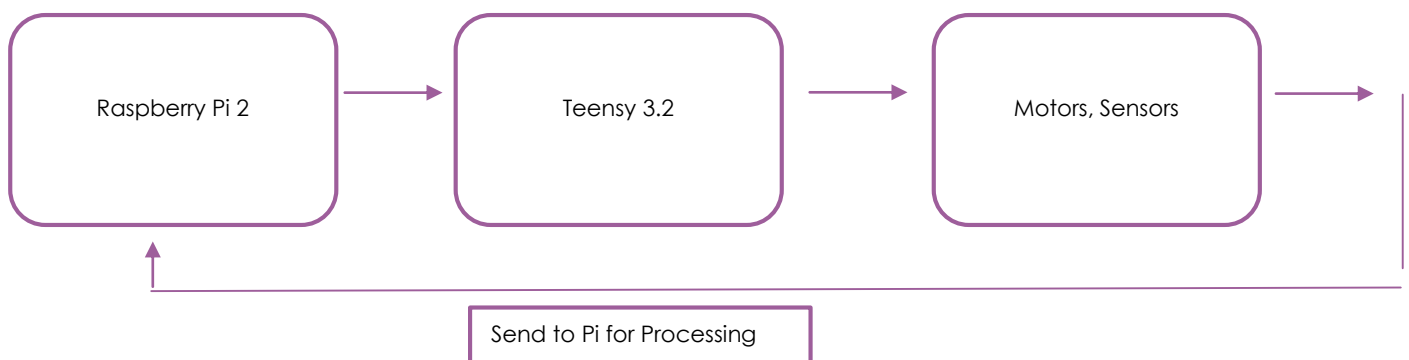
For the project, a Raspberry Pi 2 will be used as the main microprocessor. The Raspberry will be in charge of image processing. Also used, will be a Teensy 3.2, this will be in charge of controlling the motors and sensors of Robot B. Below are the specification of both the Teensy 3.2 and the Raspberry Pi 2, and a rudimentary flow chart of robots basic data flow.

Teensy:

- 32 bit processor at 72 MHz
- 34 I/O pins
- 12 PWM
- 21 ADC
- 256 KB of memory
- 3.3 V logic, supports 5 V logic

Raspberry Pi 2:

- Quad-core processor at 900 MHz
- 40 GPIO pins
- 4 USB ports
- Micro SD card needed



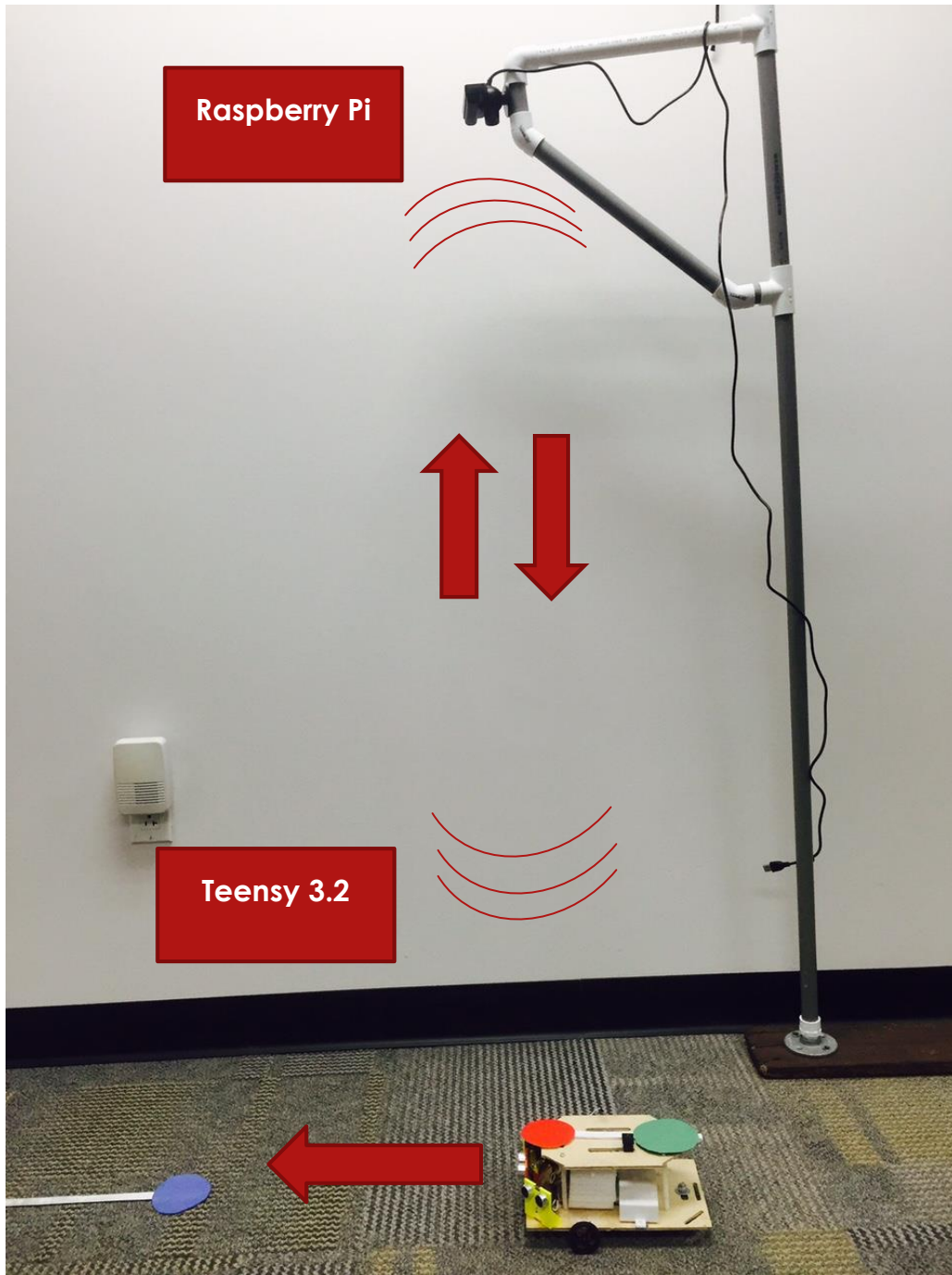


Fig. 1 Setup of the robot and the god-camera. Communication flow between the Raspberry Pi and the Teensy.

Mobile Platform

Platforms for the project were considered in depth. Here were many pre-made platforms only that range from \$5-30. To save money, the platform was designed and fabricated in house based on the parts that need to be hold. The platform has two levels. The bottom level holds all the ultrasonic sensors, the processing units, and the battery. On the second level, the two circles with two different colors are located. Their purpose is for image processing. Lastly, the battery case was 3D printed and shown below.



Fig. 2 3D printed Battery Mount

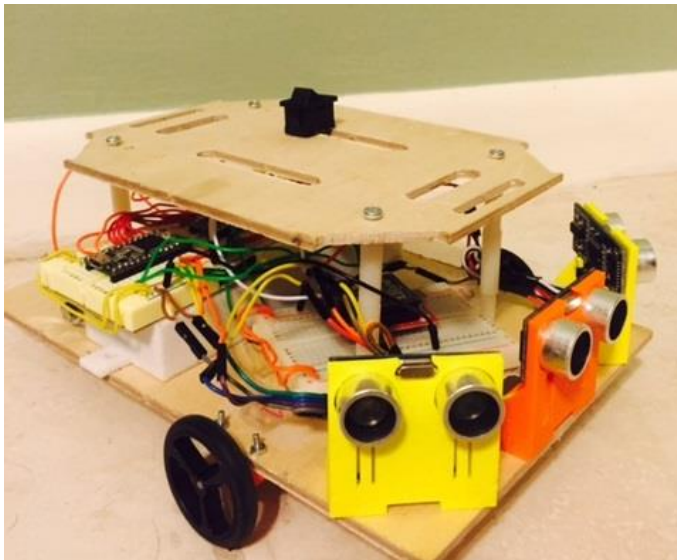


Fig. 3 Towbot with 3 ultrasonic sensors and breedboards

Actuation

As previously described, the end goal was to have two independent robots that will be moving autonomously but because of time constraint only one was develop, Robot B. The weight of this robot was estimated to be approximately 10 lbs. The necessary torque was estimated based on previous year’s robots weight. These motors were used to drive the wheels and carry the weight of the entire robots. Their objective is to move the robot to the desired location effectively without overloading the motors. The total weight of the robot was 1.08 lbs. Much less than expected.

TABLE I
Specifications of chosen gear motors

Gear Motor	Nominal Voltage (V)	Free RPM (rpm)	Stall Torque (oz-in)	Stall Current (mA)	Reduction	Dimensions Motor (inches)	Weight Motor (oz)
Micro Metal	6	100	90	1600	298:1	1.42x0.39x0 .47	0.35 (10g)
Motor + Encoder	6	110	72	1100	34:1	3.03x0.98x0 .65	-

In Table I there are the specification for the motor that were selected. The motors were selected on the following criteria:

- Voltage lower than 6 V
- Available torque from 50-100 oz-in
- Cost effective
- With encoders or encoders compatible
- Stall current no higher than 3000 mA

Because we are going to use a Teensy 3.2 to power and control the chosen motors, there is a need for a motor controller that can supply the necessary voltage and current that the motor demands. In choosing the proper motor controller, the following criteria was adopted and used from *RobotShop.com*:

- Must support motor’s nominal voltage
- Must have a continuous current of at least 1000 mA
- Communication: PWM, I²C, and/or Serial
- Must be a Dual motor driver

TABLE II
Motor Drivers chosen to power and control the motors

Motor Driver	Continuous Current (mA)	Voltage Range (V)	Max. Output Current (mA)	Built-in Thermal Shutdown	Max. PWM Frequency (kHz)
TB6612FNG	1000	4.5-13.5	3000	YES	100
RP-Spa-368	1200	2.7-15	3200	YES	100

Also ordered were two magnetic encoder pair kit for micro metal gear motors with a 12 CPR, a voltage range of 2.7-18 V, and HPCB compatible. The encoders were not used in the project. Lastly, two Pololu Wheel 40 x 7 mm Pairs were used and a 0.75" Pololu Plastic Ball Caster. The motors and motor drivers will be controlled by the Teensy 3.2 and the Raspberry PI 2.

Electronics

All the electronics were soldered on perfboards also known as DOT PCBs. Below are the pictures of the perfboards individually.

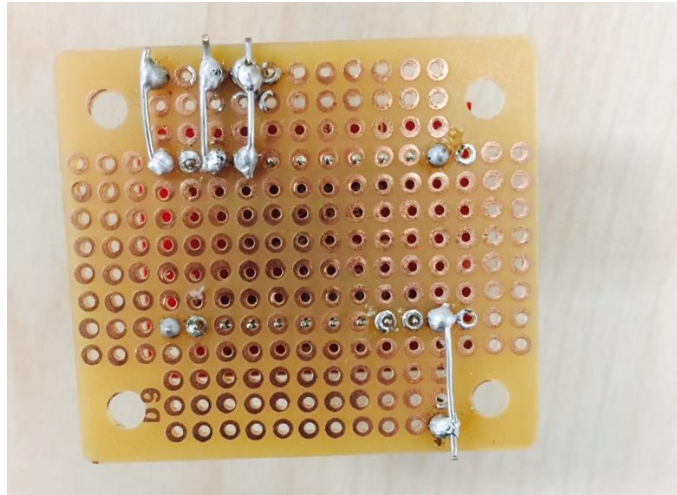


Fig. 4 Xbee Perfboard

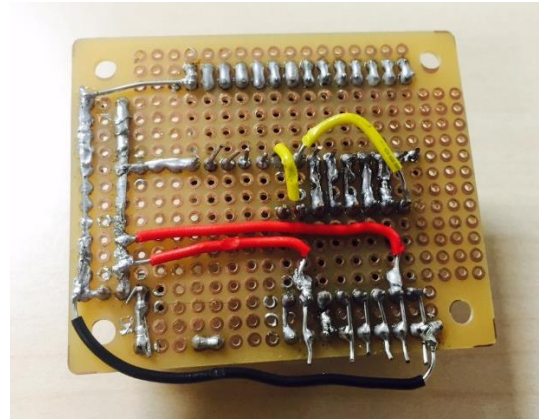
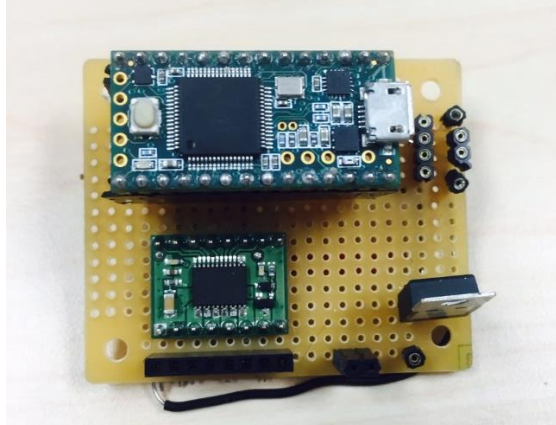


Fig. 5 Main Perfboard. Teensy 3.2, motor controller, and voltage regulator

Sensors

As previously stated, the project will be divided into two phases: Phase I-Obstacle avoidance, motor control and Phase II- Image processing, communication. The following sensors were purchased from Amazon:

- PlayStation Eye Camera (x1)
- Ultrasonic Module (x3)
- Xbee Radio communications

The purpose of the camera is to act as a global positioning system (GPS) that would locate Robot B and the final destination. The camera locates the broken robot, or the color where the broken robot would be, it will process its location, and send the appropriate command to Robot B.

The ultrasonic sensors are responsible of sensing obstacles on Robot's B path. The data is then transmitted to the Raspberry Pi via Xbee radio communications for processing. The Pi processes and sends the appropriate command.

Behaviors

The robots' current behaviors are:

- God-camera
 - Color detection
 - Color centroid analysis
 - Vector creations, magnitude and angle analysis
 - Command communication to Robot B via radio
 - Receive ultrasonic sensors information and process

- Robot B (Towing robot)
 - Able to receive the pixel location from the god-camera of Robot A
 - Able to move and avoid obstacle through an environment

The robots' future behaviors will be:

- Robot A (Broken-down robot)
 - Able to move and avoid obstacles through an environment
 - Able to send a signal to Robot B (Towing robot) when it needs towing
 - Future: Able to receive a signal from Robot B when it arrives to towing dock
 - Future: Able to follow a path given by the god-camera
- Robot B (Towing robot)
 - Able to 'tow', attached to Robot A in order to tow to desired location
 - Future: Able to find the best path/ route to reach Robot A based on object recognition from the god-camera

Special Sensor

As previously stated, the main component of the project is the god-camera. The god-camera partnered with the Raspberry Pi drives the entire project. The inspiration of the project is derived at the current technology that is available, that is, Global Positioning System (GPS). The idea is to have a god-camera that would locate objects in its visibility and it would be able to provide specific location coordinates of those objects. The initial stages of the project consisted of performing obstacle avoidance. This is a key feature for the robot because as the robot navigates to the desired object, in our case to the broken car, it must be able to avoid obstacles that are on its way.

The procedures can be broken down into three main categories: Image Processing, Mathematics, and Feedback System. We'll examine each one of them carefully.

I. Image Processing

All the image processing is performed by a Raspberry Pi 2 and PS3 Eye Camera with OpenCV and Numpy libraries using Python 2.7 as the coding platform. The overview steps for image processing are shown below with their respective OpenCV function name. The complete code can be found in *Appendix A*.

- ▀ Image Processing:
 - ▀ Convert image from Color to HSV Values
 - ▀ `cv2.cvtColor(im, cv2.COLOR_BGR2HSV)`
 - ▀ Mask the image for color detection
 - ▀ `cv2.inRange()`

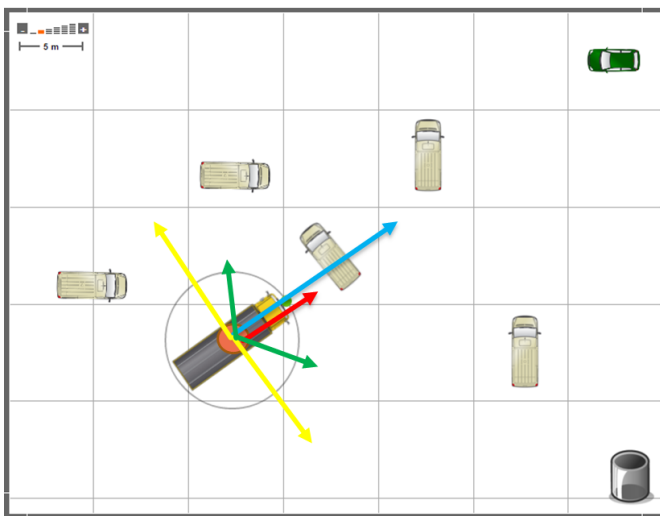
- Erode and dilate the image for noise reduction
 - `cv2.erode()`, `cv2.dilate()`
- Find contours of objects
 - `cv2.findContours()`
- Find centroids of objects
 - `cv2.moments()`

II. Mathematics

The mathematics are the brain of the entire data processing that is able to guide and command the towing robot towards the desired location. The necessary vectors components are shown in the list below.

- Mathematics:
 - Find the origin and object centroids
 - From the objects centroids, create vectors
 - Vector 1 is Robot's heading
 - Vector 2 is path direction
 - Vector 3 is orthogonal to vector 1
 - Vector 4 is the appropriate direction to correct given the ultrasonic data

Fig. 1 shows all the different vectors that are utilize for heading estimation, path planning, and obstacle avoidance. This is the current phase the project is at. See *Appendix B* for a trajectory simulation.








	Origin	-
	Vector 1	Heading
	Vector 2	Path
	Vector 3	Ortho.
	Vector 4	Sum

Fig. 6 Vector representation of the robot in order to arrive at the desired location while avoiding obstacles.

III. Feedback System

The feedback system is the most important component for the obstacles avoidance. The robot will use three ultrasonic sensors that will be able to detect the approximate distance of objects. This sensors serves as feedback system to the Raspberry Pi. The Teensy 3.2, which activates both the motors and the ultrasonic sensors, will be communicating with the Raspberry Pi via radio frequencies using an XBee. Below is an overview of the feedback system.

- Feedback System:
 - Receive objects distance from Ultrasonic
 - Communicate from Teensy to Raspberry Pi via Xbee
 - Process logic in Raspberry Pi
 - Send command to Teensy for actuation

Conclusion

As stated before, the goal of the project is to design two autonomous robots with different goals. One with the task of alerting when it was been broken down, and the other with the task to retrieve. Important factors that took time and energy were imagine processing, coordinate transformation, communication, and latching mechanisms. Because of this, only one robot was developed. Although it seemed simple, the data processing was in-depth and took great time to develop. I highly enjoyed this project and the software side of it. I am so happy that I decided to focus on software and learn a new area in the engineering field. Some of the future work is to implement coordinate transformation, latching mechanisms, A Star routing tool to develop an optimized path.

Appendix

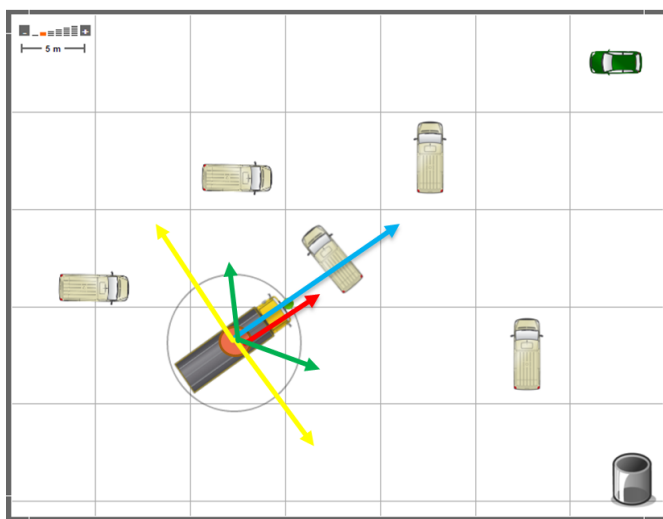
Appendix A


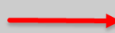



Vital Resources used for the Project

- Battery Arrangement (Parallel and Series)
 - http://batteryuniversity.com/learn/article/serial_and_parallel_battery_configurations
- Obstacle Avoidance
 - <http://www.duino-robotics.com/obstacle-avoidance.html>
- Installing OpenCV 3 on Raspian Jessie
 - <http://www.pyimagesearch.com/2015/10/26/how-to-install-opencv-3-on-raspbian-jessie/>
- Image Processing
 - <http://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/>
 - https://github.com/abidrahmank/OpenCV-Python/blob/master/Other_Examples/multi_color_two_object.py
- Vector Analysis
 - <http://www.pdnotebook.com/2012/07/measuring-angles-in-opencv/>
- Controlling
 - <http://blog.oscarliang.net/raspberry-pi-color-tracking-opencv-pid/>

Appendix B

Trajectory Simulation



	Origin	-
	Vector 1	Heading
	Vector 2	Path
	Vector 3	Ortho.
	Vector 4	Sum

