

# Color Tracking Turret Project - Design Review 1

Arafat Amin,<sup>\*</sup> Ben Chu,<sup>†</sup> Mark Martinez,<sup>‡</sup> Michael Podust,<sup>§</sup> and Stephen Tu<sup>¶</sup>  
*Dept. of Mechanical Engineering - University of California, Berkeley*  
(Dated: March 4, 2010)

Design Review 1 written report for our ME102B project, where a comprehensive summary of the project is presented in detailed length. First, we touch upon the background information necessary to understand the context of the project. Then, we take a digression and explore why other concepts were not selected for the semester project. A defense of our concept selection is then presented. Finally, the project is described in much detail, and preliminary plans (including management charts) are shown and described.

---

<sup>\*</sup>arafat@berkeley.edu

<sup>†</sup>benchu@berkeley.edu

<sup>‡</sup>mmartinez213@berkeley.edu

<sup>§</sup>mpodust@berkeley.edu

<sup>¶</sup>stephen\_tu@berkeley.edu

## I. EXECUTIVE SUMMARY

### A. Introduction

Modern day warfare is an increasingly impersonal affair. We seek to investigate the extents to which technology can be applied to weaponry, in order to mitigate the number of military casualties. Specifically, we want to explore the area of image processing with respect to target recognition to see exactly how much human intervention we can remove from the control loop.

### B. Goals

Specifically, we propose to prototype a system which provides the following functionality:

1. Tracking of arbitrary color of a target.
2. Ability to control the turret remotely, with a simple user interface.
  - (a) Input of which color to consider as the target.
  - (b) Fire on command.

### C. Design

We will seek to design the entire system from the ground up. The components include:

1. Mechanical missile turret armed with electronically powered firing mechanism.
2. Camera mounted to turret for color recognition.
3. Computational server that will be responsible for running the control loops and the color detection algorithms.
4. Interactive server that will interface with the computational server to respond to human commands, in addition to providing a user interface.

### D. Proposed Implementation Details

In our first implementation, we propose the following architecture. The turret will have two degrees of freedom,  $X$  and  $Y$ . There will be two motors controlling both axis independently. These motors will receive PWM signals from an Arduino microcontroller board mounted on the turret. The Arduino board will, in turn, receive control command data from the computational unit via a serial link (or wireless). We will use a simple serialization format for this communication. The camera mounted on the turret will be connected to the computational unit via USB.

The computational unit will be running the algorithm described previously. We will make use of Intel's Open Computer Vision (OpenCV) API to do our image processing.

The interactive unit will expose a user interface to the turret via a web interface. This interface will expose functionality such as changing the color to track, and firing the missile. Such commands will be communicated to the computational unit via either an asynchronous RPC framework if we decide to run the two units on two separate machines, or a UNIX socket if running on the same machine.

### E. Challenges

Most of the challenges we foresee deal with the accuracy and sensitivity of our color detection algorithm. Because we are using a simple heuristic model, our algorithm is quite susceptible to changes in the background; it will work best when there is an obvious contrast between the color to track and the background. Also, the color will need to be a solid block. A more complete model would incorporate a combination of heuristics and statistical machine learning models over large data sets to provide a more accurate interpretation of the image.

Another challenge we foresee has to do with the accuracy of the missile. Because we are only detecting color, depth information is not captured very well. Therefore, how we position the gun (how we determine  $(X_I^*, Y_I^*)$ ) might not be as accurate in terms of missile trajectory). To deal with this, we might take into consideration how bright the recognized color is, and adjust our reference position accordingly. This would probably be aided by switching to do calculations in the hue-saturation-value (HSV) color space (instead of RGB), which OpenCV has built in support for.

## CONTENTS

I. Executive Summary	2
A. Introduction	2
B. Goals	2
C. Design	2
D. Proposed Implementation Details	2
E. Challenges	2
II. Introduction	4
A. The Project	4
B. Background Literature	4
III. Specifications	5
A. QFD	5
B. Translation from Customer to Engineering Requirements	5
C. Competitors	5
D. Concept Generation	6
E. Concept Selection	6
1. Why Color Tracking Turret?	6
2. Concept Selection Matrix	7
3. Concept Scoring Matrix	7
IV. Concept Description	7
A. Solidworks Drawing	7
B. Architecture Overview	8
C. Controller	8
1. Controller Serial Wire Format	8
2. Serial Communication Implementation	8
D. Image Processing Unit	9
1. RGB Color Space Algorithm	10
2. HSV Color Space Algorithm	10
3. Unit Details	10
4. Image Processing Unit Socket Wire Format	10
E. Interactive Unit	10
V. Plan	11
A. Gantt Chart	11
B. RASCI Chart	11
VI. Challenges	11
A. Actuation of Gun with Desired Reaction Times	11
B. Object Detection	11
VII. Code Repository	11
VIII. Conclusion	13
IX. Appendix	13
A. Concepts	13
References	13

## II. INTRODUCTION

In [1] and [2], we discussed our initial color tracking turret project in detail. In this report, we hope to unify all these details into a comprehensive summary and present this information in a clear, easy to follow manner. We also hope to address some implementation level concerns that readers may have of the project.

### A. The Project



FIG. 1: Airsoft Turret

Figure 1 is an illustration of a similar project that we are trying to accomplish [3]. Our project is going to be a color tracking missile turret, which will have either a paintball or BB gun mounted on board, and will be controllable from a remote interface. The target audience for this *kind* of project is military establishments, who seek to minimize the number of human operators in the field. The reasoning behind this is well documented; high military casualties fuel combat aversion among citizens (especially in the United States), which then constrains the military budget and thus places further burdens on the military [4]. While this area of interest is a wide one with a lot of potential projects, we chose to focus on tracking based on image processing because it provides us a technically challenging problem to work on. Furthermore, image processing becomes nearly inevitable when humans are removed from the loop; rarely are basic sensor readings sufficient to provide the information necessary to make the same kind of decisions that humans would make, with any high level of certainty.

Even though our prototype will only contain a toy gun, it is reasonable to expect that a working prototype could be expanded to mount a military grade weapon given some further engineering effort.

### B. Background Literature

Arguably the most complex part of the project is the color tracking aspect. However, because we have decided to offload our image processing to a separate computing unit and not perform it on the microcontroller, we are not going to be restricted only to algorithms that run under strict memory and speed constraints.

In [5], it is argued that the HSI color space is the most feasible for autonomous robot situations. Our own experimentation with OpenCV has witnessed similar results, and so most likely we will be working with HSI (or some closely related variant, such as HSV). However, the techniques in [5] are not particular well suited for situations where the surroundings will present a lot of disturbances in the image (such as the target to track will fade away, there will be interference, etc.). Therefore, more complex models based on adaptive Kalman filters have been proposed by [6]. These techniques are considerably more advanced, and would only really be necessary if our more naive techniques perform very poorly.

A color tracking robot is a fairly common project, done at many levels of abstraction. See [7], [8], and [9]. Furthermore, the project is often done as a school project, meaning that it is a tractable semester long project.

### III. SPECIFICATIONS

#### A. QFD

The QFD is shown in Figure 2. First a list of a customer needs were made and ranked from 1 to 10 in terms of priority. The needs were ease of use, unit cost, easy to obtain, reliability and quality, with reliability and the most pressing customer need. Then a list of engineering requirements were made and a relation weight from 1-10 was given. These engineering requirements include: complexity, manufacturing cost, distribution, quality control, precision, reliability and speed. From the values of the relation weights, a target engineering score was given 1-10. Speed and reliability were given the highest target value. Next we compared the how our product addressed the needs of the customer, with our competitor, which is a commercial color tracking device.

Requirement Relations Matrix (1-10)									
		Target	4	6	4	7	8	9	9
		Engineering Requirements	Low Complexity	Low Manufacturing Cost	Distribution and Marketing	Quality Control	Precise Machining	Software Speed	Electronic Reliability
Weight (1-10)	Customer Requirements								
7	Ease of Use		6	1	1	5	5	9	8
7	Cheap Unit Price		10	10	1	3	1	1	1
5	Easy to Obtain		1	1	8	1	1	1	1
9	Reliability		3	2	1	7	8	9	9
8	Quality		1	2	1	6	7	8	8
		Competition Benchmarks (1-10)							
		Our Product							
		Commercial Color Tracking							
			6	5					
			4	9					
			6	8					
			7	6					
			8	7					

FIG. 2: QFD

#### B. Translation from Customer to Engineering Requirements

1. **Ease of Use** - The customer does not want to have to understand a lot of theory, or have to configure the device very much. Therefore, we believe *low complexity* to be a translation of this requirement.
2. **Cheap Unit Price** - Ideally, the customer would be able to purchase many of these products at a low cost per unit. Therefore, we will require *low manufacturing cost* as an engineering requirement (there is a direct correlation here).
3. **Easy to Obtain** - The product should be readily available to customers. Therefore a distributed marketing system should be in place.
4. **Reliability** - This is probably the most important requirement, since misfires would trigger many consequences and backlash. Thus we will place the highest priority on *precise manufacturing* and *electronic reliability*. This is a trade-off with the previous two requirements, but a very necessary one.
5. **Quality** - This goes hand in hand with the previous requirement. We will focus on *quality control* here as our engineering requirement.

#### C. Competitors

Our main competitor is the Sentry Project [10]. They offer a complete package which includes the physical turret and all the electrical/software components needed to make it track. However, there are several disadvantages to their offering which we hope to address:

1. **High Cost** - The complete package runs USD 250 for their product.
2. **No Remote Administration** - The administration panel must be directly connected to the sentry. We plan to offer remote administration over the web.
3. **No Modularization** - To use their offering, the user is tied down to their entire stack, from the microcontroller and gun all the way up to the GUI. We plan to modularize our offering, so that components can be easily substituted for one another.

#### D. Concept Generation

We did not have to go through many concept generation iterations before settling on our final one (the turret presented in detail in this report). The only details that we were uncertain of with respect to the project were what kind of gun to mount on the turret. We finally decided on a paintball gun, which provides a challenge because it is a heavier gun (than say a BB gun). As a backup plan, if we are unable to actuate the paintball gun in a desirable manner, we will resort to mounting a laser pointer, which we believe to be much simpler to implement.

Refer to the Appendix to see our other concepts.

#### E. Concept Selection

We will describe the advantages and disadvantages of each concept here, and then we will show the scoring/selection matrices. Recall that the Appendix lists the description of each of these concepts (we will omit the description here for conciseness):

##### 1. Projectile Shooter

- (a) **Advantages** - Neat concept, interesting problem to solve.
- (b) **Disadvantages** - Mechanically challenging to implement (above ground shooting apparatus is quite complex).

##### 2. Robot Scripting Language

- (a) **Advantages** - Really hands-on practical project, provides great experience.
- (b) **Disadvantages** - Time consuming. Writing a compiler is not an easy task!

##### 3. Maze Solver

- (a) **Advantages** - Simple algorithm known to solve any simply-connected maze.
- (b) **Disadvantages** - Mechanically complex; lots of hardware needed to build robot.

##### 4. Autonomous RC Car

- (a) **Advantages** - Hardware is easy to obtain; has been done many times.
- (b) **Disadvantages** - Moving base makes any sort of computer vision difficult.

##### 5. Color Tracking Turret

- (a) **Advantages** - Challenging project, involves many aspects of hardware and software.
- (b) **Disadvantages** - Color tracking is a somewhat naive model for object detection; a more robust solution would utilize more complex methods for detection.

##### 1. Why Color Tracking Turret?

We objectively believe that the color tracking turret is the best project proposal for the following reasons:

1. **Mechanical Complexity** - We offer two degrees of freedom ( $(X, Y)$ -rotation), in addition to (possibly) having to mechanically actuate the trigger.
2. **Electronic Complexity** - Our system is a closed feedback loop system, so we will require a non-trivial electronic component to control the mechanical actuation.

3. **Software Complexity** - The project offers potential for unbounded complexity (we can always keep improving on our recognition/detection algorithm) in the image processing component, while adding another layer of complexity with the remote interface.

Because this project offers complexity in each of the three fields, we believe that this gives each group member the ability to focus on the area which he is the strongest. None of the other projects listed above are as well balanced in complexity as this project. For instance, the *Robot Scripting Language* project, while tremendously interesting, is more focused on software design than mechanical design (and is probably better suited for a language design course).

Likewise, the *Maze Solver* project can become very complex on the software level if we do not restrict ourselves to simply-connected graphs. More complex graph algorithms would have to be used, which is probably more suited for an algorithms course.

The *Projectile Shooter* project was deemed to be too mechanically complex for the group's liking. Additionally, the problem is not very well defined, and it is not clear what the applications would be.

Finally, the *Autonomous RC Car* project just did not spur much interest from the group, perhaps because it is a problem that has been done many times.

## 2. Concept Selection Matrix

The Concept Selection Matrix is shown in Table I.

TABLE I: Concept Selection Matrix

Criteria	Datum (Color Tracking Turret)	Projectile Shooter	Robot Language	Maze Solver	RC Car
Mechanical Complexity	0	+	–	0	–
Electrical Complexity	0	+	–	+	–
Software Complexity	0	–	+	–	–
Interest Level	0	–	+	–	–
Cost to Prototype	0	–	0	–	+
<b>Total</b>	0	–1	0	–2	–3

## 3. Concept Scoring Matrix

We use the Concept Scoring Matrix, shown in Table II to select between the *Color Tracking Turret* and the *Robot Language*.

TABLE II: Concept Scoring Matrix

Criteria	Weight	Datum (Color Tracking Turret)	Robot Language
Group Involvement	1	0	–
Time Commitment	2	0	–
Application	3	0	–
Ease of backup plan	2	0	–
<b>Total</b>		0	–8

# IV. CONCEPT DESCRIPTION

## A. Solidworks Drawing

A preliminary Solidworks 3D drawing of the missile turret is show in Figure 3

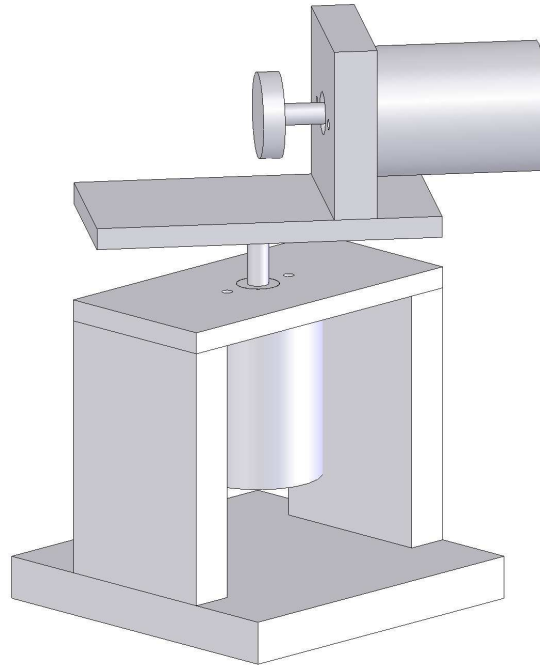


FIG. 3: Color Tracker Turret Solidworks Drawing

## B. Architecture Overview

A high level overview of the proposed system architecture is shown in Figure 4. Each of the components in the figure will be addressed in more detail in subsequent sections.

## C. Controller

The controller will be a standard PID control algorithm. There will be one PID running for each axis (two of them in total). It will be running off of the Arduino board, sending PWM signals directly to the on-board circuitry of the turret. The controller gains will be hardwired (fine-tuned, of course) into the board, but the reference input and current state information will all come over the serial interface via the wire format described below.

### 1. Controller Serial Wire Format

Since the controller will be receiving input data from the computational unit via a serial cable, we need to define a wire protocol for this communication. We currently do not believe that we need any handshake protocol to establish communication other than what the serial interface already provides. The packet format is depicted in Figure 5. Note that the *fire* field is 1 byte despite being only a boolean value, since 1 byte is smallest unit of transfer.

### 2. Serial Communication Implementation

Because the Arduino Serial API allows a programmer to query the number of bytes available to read in the serial receive buffer [11], we can effectively mimic non-blocking IO (even though the serial API is blocking) by simply checking on each loop iteration whether or not the number of bytes available is enough for us to build a packet (or possibly more than one packet). This way we can keep our controller code as simple as possible, to avoid bugs. The only thing we have to worry about is making sure that the rate of packet arrival is no faster than the time interval of each PID controller iteration.



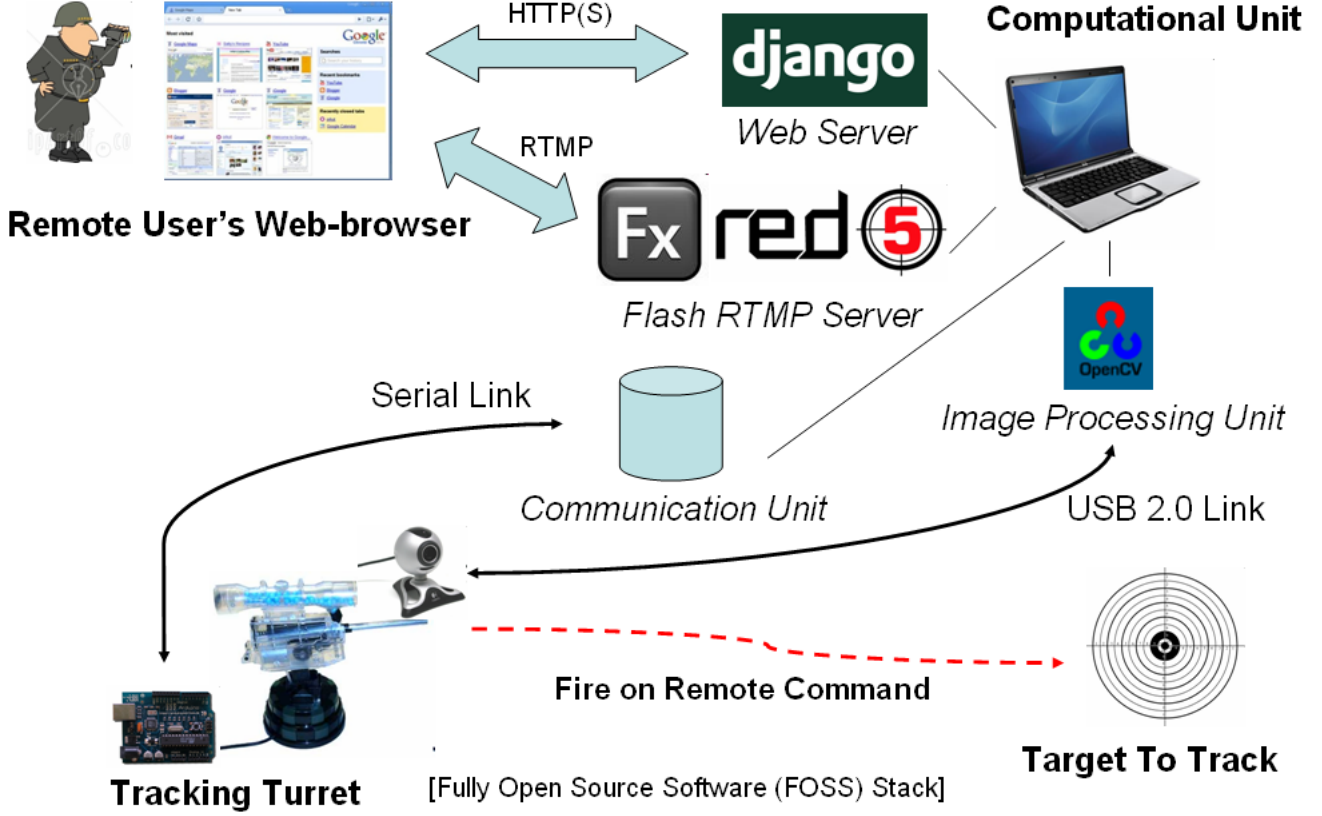


FIG. 4: High Level System Architecture Overview

$x_t$ (2 bytes)	$y_t$ (2 bytes)	$x_{des}$ (2 bytes)	$y_{des}$ (2 bytes)	$fire$ (1 byte)
-----------------	-----------------	---------------------	---------------------	-----------------

FIG. 5: 9-byte serial packet format

#### D. Image Processing Unit

The image processing unit will monitor a camera (web-cam) that is attached to the turret. The behavior of this unit is described as follows:

---

**Algorithm 1** ImageProcessingUnit( $(R_d, G_d, B_d), x_{des}, y_{des}$ )

---

**while** true **do**

Capture  $(R_i, G_i, B_i)$  values for each pixel from camera

Process all events in the events to process buffer

Let  $I :=$  converted black/white image based on heuristic function (inputs are  $(R_d, G_d, B_d)$  and  $(R_i, G_i, B_i)$ )

Let  $(X_I, Y_I) :=$  spatial centroid of  $I$

Run PID on  $X_I$  and  $Y_I$  such that  $(X_I, Y_I) \rightarrow (X_I^*, Y_I^*)$

Let  $S :=$  serialize  $x = (X_I, Y_I)$  and  $r = (x_{des}, y_{des})$  into the wire format

Send  $S$  to serial port (to controller)

**end while**

---

The spatial centroid  $(X_I, Y_I)$  is defined as follows:

$$X_I = \frac{M_{10}}{M_{00}} \quad (1)$$

$$Y_I = \frac{M_{01}}{M_{00}} \quad (2)$$

$$M_{mn} = \sum_j \sum_k x_k^m y_j^n P_{j,k} \quad (3)$$

Where  $(x_j, y_k)$  is a pixel location,  $P_{j,k}$  is a pixel value, and  $m$  and  $n$  are the order of the moment[12]. As for the image heuristic function, we have two options right now.

#### 1. RGB Color Space Algorithm

---

**Algorithm 2** RGBColorSpace( $(R_d, G_d, B_d), (R_i, G_i, B_i)$ )

---

Let  $x := \sqrt{(R_d - R_i)^2 + (G_d - G_i)^2 + (B_d - B_i)^2}$   
**return**  $x < TOL$

---

$TOL$  is a predetermined tolerance level which we will calibrate for the environment.

#### 2. HSV Color Space Algorithm

---

**Algorithm 3** HSVColorSpace( $(R_d, G_d, B_d), (R_i, G_i, B_i)$ )

---

Let  $x := \text{convert}(R_d, G_d, B_d) \text{ to } (H_d, S_d, V_d)$   
 Let  $y := \text{convert}(R_i, G_i, B_i) \text{ to } (H_i, S_i, V_i)$   
**return**  $|H_d - H_i| < H_{TOL}$  and  $|S_d - S_i| < S_{TOL}$

---

The conversion from RGB to HSV color space is described in [13]. Both  $H_{TOL}$  and  $S_{TOL}$  are calibrated constants.

#### 3. Unit Details

This unit will run on a Linux x86 machine, and ideally we want this unit to be as fast as possible (for best performance). We will use the OpenCV library to do our image processing work, and we will use the C version for best performance. To query for input from the interactive unit, this unit will listen on either a UNIX or TCP socket, depending on whether or not the interactive unit runs on the same machine. In order to not hinder with the main image loop, we will listen on the socket in a background thread. To implement this, we will probably use the standard `pthread` library.

#### 4. Image Processing Unit Socket Wire Format

The wire format for this unit is shown in Figure 6. Here, the input from the interactive unit is simply the RGB value of the color we want to track, and a field indicating whether or not we want to fire. Once again, we do not believe we need a handshake protocol here either (on top of what is already provided).

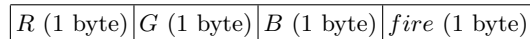


FIG. 6: 4-byte socket packet format

### E. Interactive Unit

This unit is the side that faces the user(s). We will display a video of what the turret currently sees, along with a button to press when the user wants to fire, and the ability to select the color of the target. We will probably implement this

as a website, and use Adobe Flash technology to stream video to the site. The Adobe Flex SDK is free to use now, and there are working open source implementations of a Flash server [14]. As for the website, we will probably make use of the Django [15] web framework, and implement as many of the features we can using AJAX. By implementing a website, it should be in theory possible to see through our turret's camera, and control it from anywhere in the world where there is internet (obviously we will put security measures in place). However, this definitely realizes our goal of removing humans out of the loop as much as possible.

This unit will relay user input commands by opening a UNIX or TCP socket, and sending the command via the packet format described in Figure 6.

## V. PLAN

### A. Gantt Chart

The Gantt chart is shown in Figure 7.

### B. RASCI Chart

The RASCI chart is shown in Figure 8. The acronyms are as follows (listed in decreasing order of responsibility):

1. **RA** - Responsible
2. **A** - Accountable
3. **S** - Support
4. **C** - Consulted
5. **I** - Informed

## VI. CHALLENGES

There are several challenges that we anticipate throughout the semester.

### A. Actuation of Gun with Desired Reaction Times

A paintball gun, we believe, is potentially too heavily to actuate in a responsive manner, using the hardware provided. We are investigating upgrading to a more powerful motor and an H-Bridge that is able to handle more current. If this proves to be too much of an issue, then we will restrict ourselves to a lighter gun, until we can get the response times we desire.

### B. Object Detection

We are currently working on implementing object detection, in addition to simply color recognition, in our image processing unit. We have so far been able to leverage the built in OpenCV Haar classifier algorithms to perform real-time facial recognition. The idea here is that we can train these classifiers to recognize any arbitrary object, so we will no longer be restricted to a single color.

The challenge here is that the real-time performance of the Haar classifier algorithm is still too slow to use in this kind of application. We will be investigating several methods we can use to optimize the performance so we can have the desired real-time performance characteristics.

## VII. CODE REPOSITORY

Throughout the semester, we will be hosting all of our project related code, and documentation, on our Google Code page found at: <http://code.google.com/p/me102b-sp10>

This will be a great resource to check up on our progress as time nears the end of the semester.

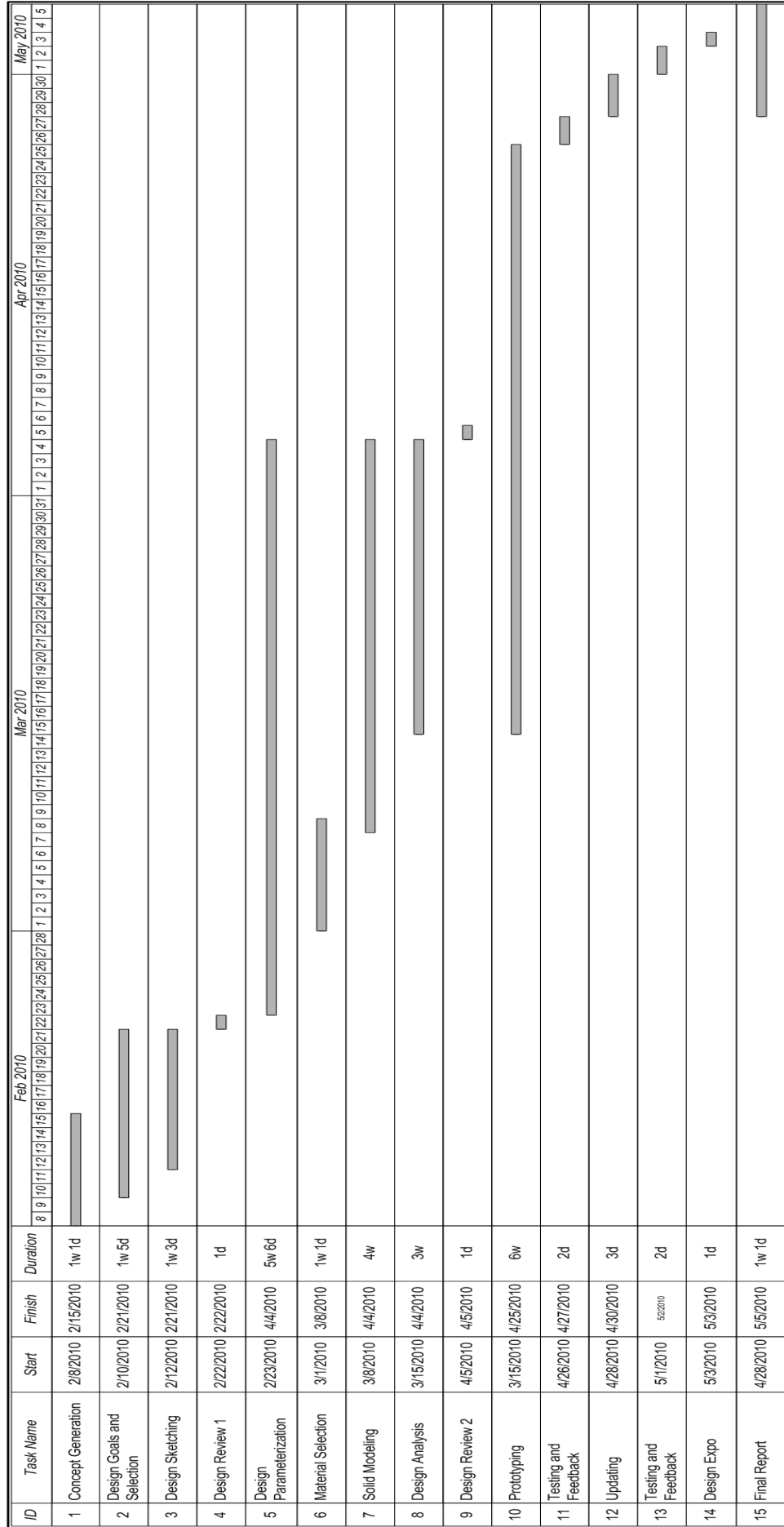


FIG. 7: Gantt Chart

Software	Arafat Amim	Ben Chu	Mark Martinez	Mikhail Podust	Stephen Tu
Development & Optimization	I	S	I	S	RA
System Design (CAD)	R	S	RA	S	I
Prototyping & Building	RA	S	S	R	C
Alpha Testing	S	RA	S	C	A
Beta Testing	S	S	S	RA	S

FIG. 8: RASCI Chart

## VIII. CONCLUSION

In this report, we introduced the color tracking turret project in its entirety. First, we compared the project to other proposals which we were considering, and used a ranking system to show that the color tracking turret was a good choice for a semester long project, given the time and resource constraints that we are dealing with (Table I and Table II).

We then presented some preliminary drawings and diagrams for the project, and went into specific implementation details. Figure 3 is a preliminary 3D Solidworks rendition of the project, and Figure 4 outlines the system architecture. We then discussed some of the challenges that we anticipate to encounter for the project, including discussion of a few algorithms to be used for the image processing unit.

Based on the analysis we have done, we believe this project to be quite challenging, yet tractable for this semester. We have been able to play around with OpenCV already, creating a working version of a few of our image processing algorithms, and we have also gotten serial communication to work with an Arduino board. We believe the rest of the work to be done is not going to be easy, but very possible.

## IX. APPENDIX

### A. Concepts

Concepts are listed in chronological order of generation, with the final one being the project we chose. The rest are unrelated to the problem at hand.

1. **Projectile Shooter** - Have a laser pointer point at an object on the ground, and have an above ground device shoot at it (like a labelling command).
2. **Robot Scripting Language** - Design and implement a programming language used to script robot actions. Target audience is less experienced people who want to get hands on experience with robots, but do not want to have to sift through learning the intricacies of low level languages.
3. **Maze Solver** - Design a robot which is able to, after being placed in a arbitrary point in a maze, navigate out of the maze.
4. **Autonomous RC Car** - Design an RC car which autonomously follows a lead vehicle.
5. **Color Tracking Turret** - The project currently described in this report.

- 
- [1] A. Amin, B. Chu, M. Martinez, M. Podust, and S. Tu. Color tracking turret initial project proposal. <http://me102b-sp10.googlecode.com/files/proposal.pdf>, February 2010.
  - [2] A. Amin, B. Chu, M. Martinez, M. Podust, and S. Tu. Color tracking turret initial project proposal - phase 2. <http://me102b-sp10.googlecode.com/files/phase2.pdf>, February 2010.
  - [3] Airsoft turret 10 - classic and simple. <http://hacknmod.com/hack/airsoft-turret-10-classic-and-simple/>.
  - [4] Evan Huelfer. *The "casualty issue" in American military practice*. Praeger Publishers, 2003.
  - [5] S. Zhao, B. Liu, Y. Ren, and J. Han. Color tracking vision system for the autonomous robot. In *ICEMI*, 2009.
  - [6] S. Weng, C. Kuo, and S. Tu. Video object tracking using adaptive kalman filter. In *Journal of Visual Communication and Image Representation*, 2006.
  - [7] A. Grieg. Robot2 - an arm based colour tracking robot. <http://negativeacknowledge.com/2009/05/robot2-an-arm-based-colour-tracking-robot>, 2009.
  - [8] D. Bekele, H. Liang, A. Leonard, and E. Mung. Color tracking robot. <http://www.cs.columbia.edu/~sedwards/classes/2006/4840/reports/VGLR.pdf>, May 2006.

- [9] S. Norris. Huey - a color chasing robot. <http://www.norrislabs.com/Projects/Huey/index.html>, 2007.
- [10] The sentry project. <http://www.paintballsentry.com/index.htm>.
- [11] Arduino reference. <http://arduino.cc/en/Serial/Available>.
- [12] Image moments. [http://software.intel.com/sites/products/documentation/hpc/ipp/ippi/ippi\\_ch11/ch11\\_image\\_moments.html](http://software.intel.com/sites/products/documentation/hpc/ipp/ippi/ippi_ch11/ch11_image_moments.html).
- [13] Hsl and hsv. [http://en.wikipedia.org/wiki/HSL\\_and\\_HSV](http://en.wikipedia.org/wiki/HSL_and_HSV).
- [14] Red5 - open source flash. <http://osflash.org/red5>.
- [15] Django. <http://www.djangoproject.com>.