Learning Manipulation of a Flashlight

Tanner Borglum, Nicolas Cabeen, and Todd Wegter
TA – Jivko Sinapov

CPR E 585X – Developmental Robotics
Final Project Report
April 21, 2011

This research was funded in part by the Iowa State University Foundation.

Table of Contents

0 - Summary

This paper presents the research done by Todd Wegter, Nicolas Cabeen, and Tanner Borglum in Professor Stoychev's Developmental Robotics Lab in the Spring of 2011. It is commonplace for a human to use a flashlight to enhance his or her vision when ambient light conditions are not sufficient for sight. This paper proposes a method by which a robot can learn to use a flashlight through a developmental approach. First, the robot explores it field of vision with the flashlight. Then, the robot can infer where to move its arm to light up a desired area from these past experiences.

1 - Project Overview

Robots are slowly becoming more and more capable of completing everyday human tasks. The field of developmental robotics is working to continue this advancement by creating robots that are capable of learning. For example, Vladimir Sukhoy and Alexander Stoytchev [1] created a program by which their upper-torso humanoid robot was able to learn to push doorbell buttons based on audio, visual, and proprioceptive feedback. We set out to create a program by which a robot can learn to properly wield a flashlight, shining its beam on a desired location. This could make it possible for a robot to push a doorbell button using Sukhoy and Stoytchev's algorithms in the dark.

1.1 - Motivation

The inspiration for our project came from a poorly designed conference room in Howe Hall at Iowa State University. The light switch is located a good distance from the main entrance, above a counter set into the wall. This makes it nearly impossible to find in the dark when you first enter the room. One day, when meeting in this conference room with Professor Stoytchev, one of our group members quickly got out his keychain flashlight to illuminate the light switch for Professor Stoytchev, who was struggling to turn the lights on in the dark. After turning the lights on, Professor Stoytchev wheeled around, exclaiming how it would be cool to program the robot to learn to do that, and our project was born.

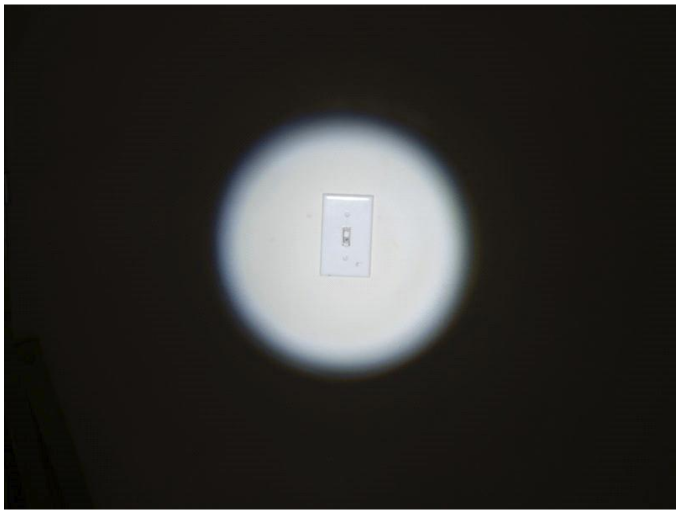
Figure 1 - Illuminated Switch

So why use a flashlight? Why not use infrared cameras or laser 3D imaging. Flashlights pose many advantages, one of which is cost. A standard flashlight costs much less than infrared cameras and 3D laser scanners. Another advantage of using a flashlight to illuminate a robot's environment is simplicity. Instead of having to switch to a whole other system for seeing in the dark, a robot using a flashlight needs only pick one up, turn it on, and point it in the desired direction. This would also allow robots to assist humans. We humans lack the capability to see in the dark, so if we require the assistance of a robot in the dark, it should be able to light the way for us.

Using flashlights to allow robots to see in the dark will help to better standardize robotic visual systems. If robots are built using different methods for seeing in the dark, it will be very hard for them to communicate visual information. By developing robots that use flashlight technology to illuminate their environment, not only will they be able to better communicate with and help humans, but they will be able to more effectively communicate with other robots.

1.2 - Audience and Applications

Flashlight use has many practical applications to a wide range of audiences. One day, robots will be “living” with us as assistants and caretakers, especially for the elderly. In case of a power outage or a nighttime emergency, these robots should be able to help their owners in any manner they should require. In one of these situations, an individual will need a flashlight’s beam to see, so being able to learn to use a flashlight will be a key skill for caretaker robots. These robots must be able to learn to use flashlights because they will undoubtedly encounter many different kinds of flashlights. With each kind being slightly different, a hard coded “how to use a flashlight” program would certainly fail, so robots will need to be able to adapt to different types of flashlights.

Robots built to work in dark environments will also greatly benefit from flashlight manipulation. For example, a robot working in a coal mine will need to be able to see, and it should also be able to help its human workers to see to. If the robot were trying to point out that it had discovered a crack in the bracing of the mine, it would be hard to point that out to a human without being able to illuminate it with a flashlight beam.

Requiring robots to use visible light to see would also make them more sensitive to visible light. If a robot were able to see in the dark using infrared cameras, it would never be able to understand why a human can’t see in the dark. We experience blindness in dark situations quite often, but a robot with the correct sensors may never have this problem. By forcing robots to have limitations similar to ours, it becomes easier for robots to relate to humans and vice versa.

Flashlight manipulation will also carry over to anything else that creates a light beam, like a laser pointer. Tour guide robots and teaching robots would be able to point out items of interest to humans by using a laser pointer much easier than by any other method. For Professor Stoytchev’s sake, we will omit the obvious extension to light sabers…

Figure 2 - Join the Darkside...

1.3 - Related Work

This section details previous works in robotics and artificial intelligence that are related to our project. Our proposal is unique in that there have been no close attempts at what we are aiming to do. However, there was a robot created at MIT that was used to light the area that a user was working in and respond to voice commands [6]. This is similar in the respect that a light was being used to target an area of interest, but their methods incorporated technologies that caused the lamp to follow the movements of a hand that a had a special glove so it could be detected. The problem of learning how to move was not solved in this research.

Another idea that was related to ours involved searching a space with a light. Lavalle's paper [7] described how to search a polygon for a moving target in the dark. The modeled method works in a situation where there is one searcher looking for one target. This is related because it involves the use of light to manipulate the environment, but once again this does not address the problem of learning how to move the light.

Self-detection has an important role in developmental robotics for a couple of reasons. One is that self-detection is related to the level of intelligence of the creatures it is manifested in. Humans are able to self-detect, some primates are, and even some other animal species can. However, most animals are not able to self-detect and/or recognize themselves in a mirror. Additionally, if a robot is able to learn about itself, how it looks, and how it can move, it should be able to adapt to situations where it may be upgraded, damaged, or otherwise changed that would cause a robot that does not have a knowledge of itself could fail after such a change [3].

Self-detection is the process through which something can differentiate “self” between other. “Self” is defined through action and outcome pairs in combination with a probability estimate based on the regularity and consistency of these pairs [5]. The approach taken by Alexander Stoytchev [3] first solves the problem of self-detection in robots by estimating the efferent-afferent delay. To find this delay, movement was corresponded with the time after a motor command was issued. Once this delay is found, differentiating “self” from other becomes easier because “self” will only move a certain amount of time after commands are issued.

Tool use is a form of self-detection and has a very important part in our proposal. Stoytchev has defined four things necessarily involved with robotic tool use: a robot, something in the environment labeled as a tool, an object to which the tool is applied, and a tool task [4]. One of the steps taken by Stoytchev was babbling with the tool grasped. The effects of the tool moving through the environment were associated with motor commands and relating the motor commands to the changes in the environment determined how the tool could be used to best manipulate the environment. Our project is similar in that it is a form of tool use, but it differs in that the robot uses tools to alter its perception rather than its ability to physically interact with the world. The ability to alter perception is something that humans use on a regular basis (one of the most intelligent animals) that many other animals can't. Some specific examples include using microscopes to see small objects, telescopes to see far away objects, and night vision goggles to see in the dark. Being able to augment perception would seem to increase the potential for understanding something better or even just to interact with the world better (such as when a human wears glasses).

1.4 - Individual Skills

Tanner Borglum

Tanner is a first year student at Iowa State University, sophomore by classification. He has programming experience in C and Java, and has learned to program in OpenCV for processing the visual sensory information we collected in our project. His knowledge of the C programming language was also helpful as the robot is programmed in C.

Nicolas Cabeen

Nicolas is also a first year student at Iowa State University, sophomore by classification. He has programming experience in C, Java and Visual Basic, and learned MATLAB for finding the error in the results collected in our project and creating contour maps of the percent error over the XY visual field to visualize the results. His knowledge of the C programming language was also helpful as the robot is programmed in C.

Todd Wegter

Todd is also a first year student at Iowa State University, sophomore by classification. He has programming experience in C and Java, and learned to use UNIX based operating systems, specifically the terminal, as the robot is run out of a UNIX terminal. His knowledge of the C programming language was also helpful as the robot is programmed in C.

Jivko Sinapov

Jivko is a graduate student at Iowa State University who works in Professor Stoytchev's developmental robotics lab. This means he has lots of experience with the robot. While he was not technically a member of our group, he was the TA for the class and helped us operate the robot and met with us in the lab for testing. He has years of programming experience, and his help has been key to the success of our project.

1.5 - Responsibilities

For this project, we divided the responsibilities as equally as possible among the group members. Nicolas was responsible for managing the group's grant money and for devising the algorithm which calculates where the robot should move its arm to illuminate a goal point from the three closest known data points. Tanner was responsible for researching related work and for analyzing the collected data. Todd was responsible for creating the programs which used the processed data and algorithms to move the robot's arm to illuminate a point. He also handled setting up times to meet in the lab, collect data, and meet with Alex and Jivko.

1.6 – Timeline

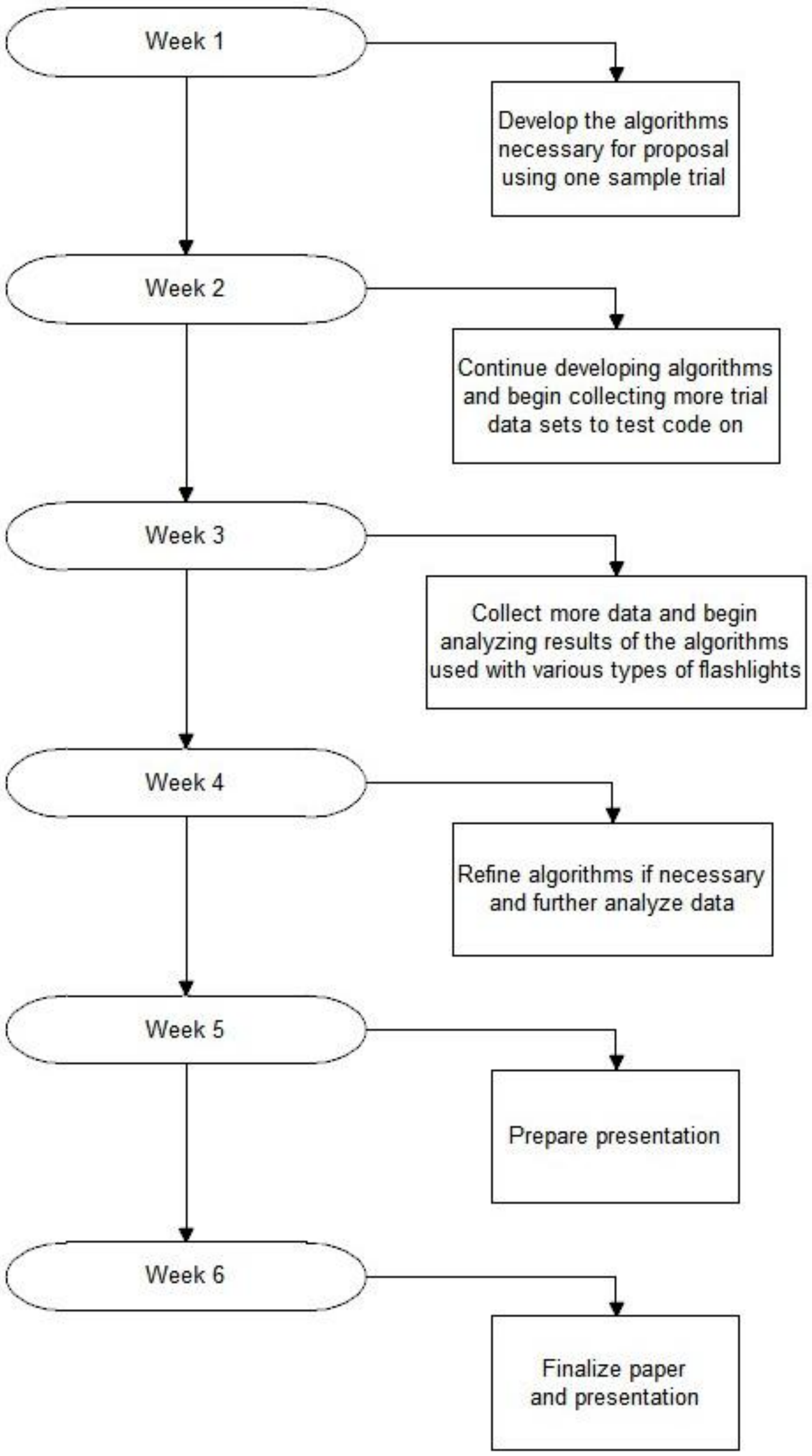
Week 1
Develop the algorithms necessary for proposal using one sample trial
Week 2
Continue developing algorithms and begin collecting more trial data sets to test code on
Week 3
Collect more data and begin analyzing results of the algorithms used with various types of flashlights
Week 4
Refine algorithms if necessary and further analyze data
Week 5
Prepare presentation
Week 6
Finalize paper and presentation

Figure 3 - Timeline

2 - Approach

2.1 - Equipment

Robot

The flashlight exploration experiments will be performed with the upper-torso humanoid robot illustrated in Figure 3. Two Barrett Whole Arm Manipulators (WAMs) are used for the robot's arms. Each WAM has seven degrees of freedom. In addition to that, each arm is equipped with a three-finger Barrett BH8-262 Hand as an end effector. Each hand has seven

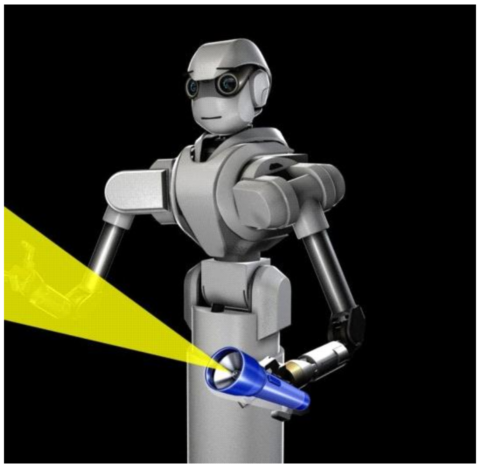
Figure 4 - Robot with Flashlight - Simulation

degrees of freedom (two per finger and one that controls the spread of fingers 1 and 2). Because fingers one and two can rotate by 180-degrees, the robot can perform a variety of grasps. In other words, even though the robot has only three fingers it can more than compensate for that because it has not one but two opposable thumbs.

Flashlights and Batteries

We used three different flashlights for collecting the data in our experiment: 1 Maglite flashlight with a standard incandescent lamp which used 3 C batteries, 1 LED flashlight which used 3 C batteries, and 1 multicolored LED flashlight which used 3 AAA batteries. The

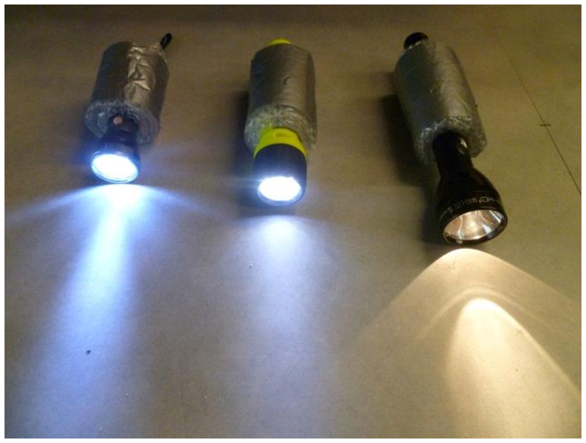
Figure 5 - Flashlights

Maglite had a focusable beam, which we set to have the tightest possible radius on our testing

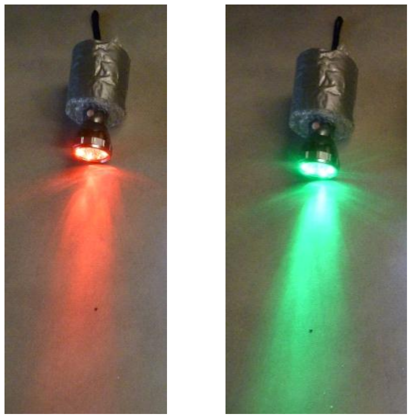
Figure 6 - Red and Green Flashlights

surface for one set of trials. We also set it to be unfocused for another trial set. The

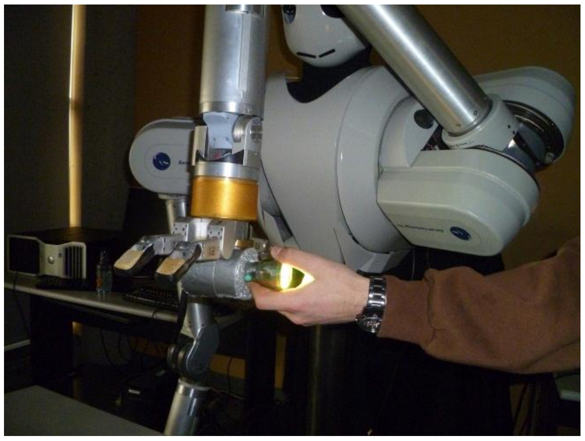
Figure 7 - Grasping the Flashlight

multicolored LED flashlight featured white, red, and green LED's. This allowed us to collect data for white, red, and green light beams. In our experiment, the flashlight is turned on for the robot, as turning the flashlight on and off is beyond the scope of this project. It also assumed to be grasped as that is also beyond the scope of the project.

Experimental Setup

For our experiment, we pointed the robots head down at a table and had the robot shine the flashlight at the table. The robot then observed and recorded the light patterns produced on the table in conjunction with its

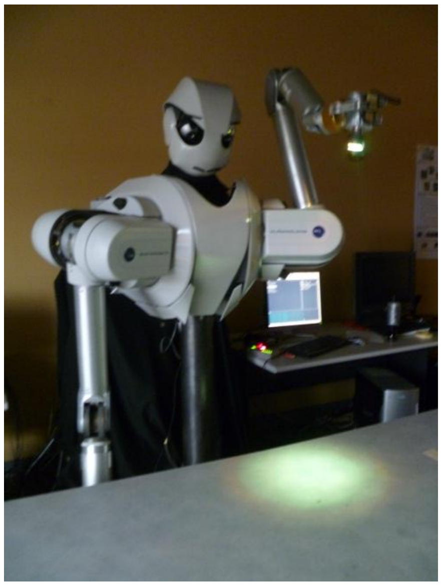

Figure 8 - Experimental Setup

arm's joint positions. We chose this position due to time constrains and others working in the lab. Since other groups were conducting research at the same time as us, we needed to use the robot as it was so as to not interfere. This setup should not augment or reduce the performance of our method in any way.

Software

The robot is controlled using C++ on a UNIX platform. OpenCV, an open-source library, was used for image processing. Matlab was used to handle post-experiment data analysis. A simulation/demo program was also created. Both the robot program and the simulation take in processed image data with corresponding joint positions, determine a random point to shine the light, and calculate the joint positions necessary to shine the light there.

2.2 – Method & Algorithms

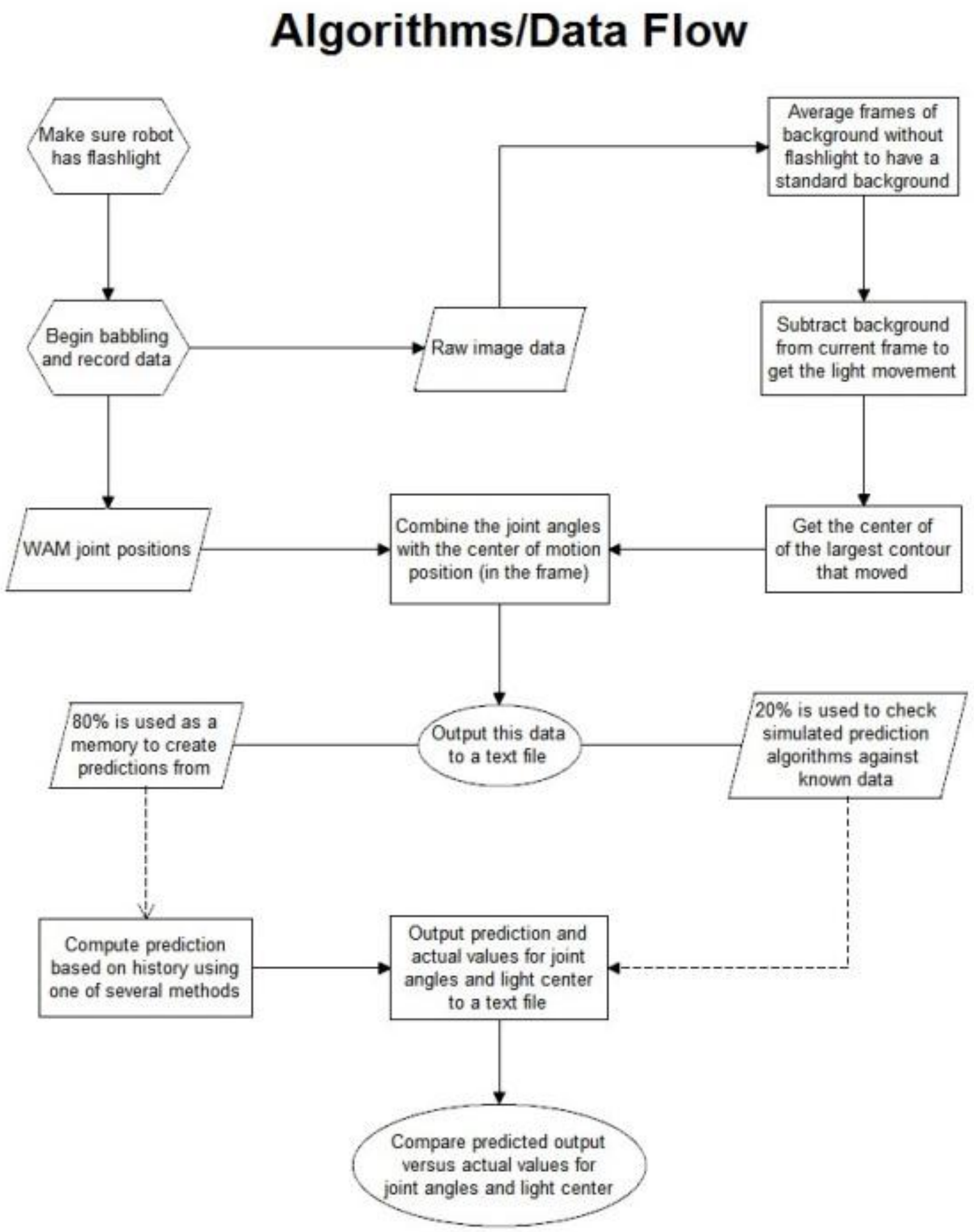
Algorithms/Data Flow
Make sure robot has flashlight
Begin babbling and record data
Raw image data
Average frames of background without flashlight to have a standard background
Subtract background from current frame to get the light movement
WAM joint positions
Combine the joint angles with the center of motion position (in the frame)
Get the center of of the largest contour that moved
80% is used as a memory to create predictions from
Output this data to a text file
20% is used to check simulated prediction algorithms against known data
Compute prediction based on history using one of several methods
Output prediction and actual values for joint angles and light center to a text file
Compare predicted output versus actual values for joint angles and light center

Figure 5 - Algorithms and Data Flow

Figure 9 outlines the general process we used to collect and analyze data and test our program.

Initial data was collected using a program developed by Jivko to randomly babble the arm to 20 different randomly generated points. The points were generated from the 3D plane whose corners were defined by the robot's arm in setup while holding the flashlight. First a background set was collected by letting the robot babble its arm with the flashlight off. We then turned the flashlight on, and the robot recorded visual and proprioceptive data. We then repeated the babbling process with two additional flashlights, one of which had three different colors of lights. Each flashlight was tested ten times, resulting in a vast collection of visual and proprioceptive data.

Our experiment is assuming that the flashlight is grasped and turned on as these parameters are outside the scope of this project.

The background used for image differencing was created by adding all of the images in the background set and equally weighting them. This background was used in the processing of all flashlights. The background changed during some sets so the average background was affected, but did not cause the algorithms to fail.

In order to process the images, we first took the absolute value of the difference between the current image and the average background, which worked in most cases except where the entire frame was lit by the flashlight. We then used Gaussian and blur smoothing on the captured images to reduce noise. The next step was to remove the background. To further reduce error, we did a binary threshold (in color) on the image, shown in the middle image of figure 10. After this step, we converted the image to grayscale and used a Canny edge detection algorithm on it. We

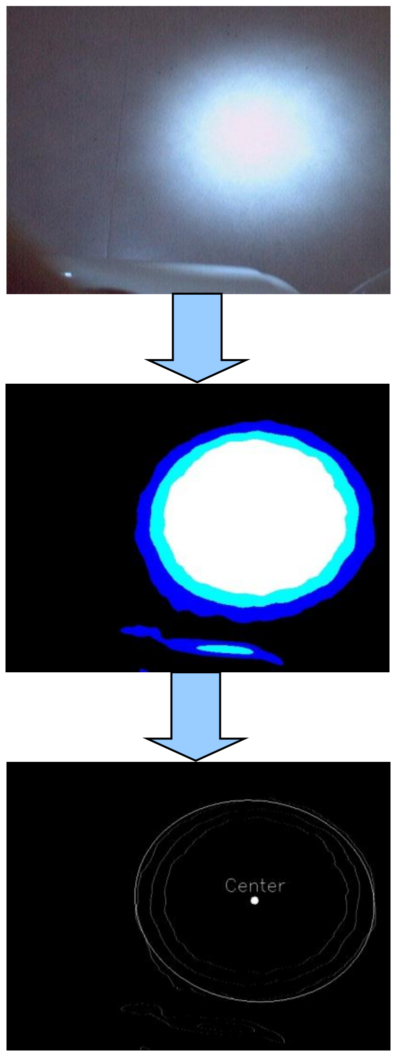

Figure 6 – Image Processing for White Light

produced contours from the binary output of this algorithm and decided to use the largest contour to capture the motion of the light. To determine how the light moved, we estimated the largest contour with an ellipse (which is represented as the closest fitting rectangle in OpenCV) and used the center of the ellipse as the center of light/motion in each frame. As you can see below, this algorithm worked even when most of the background was different from the average background and if the input was irregular, as seen in figures 11 and 12.

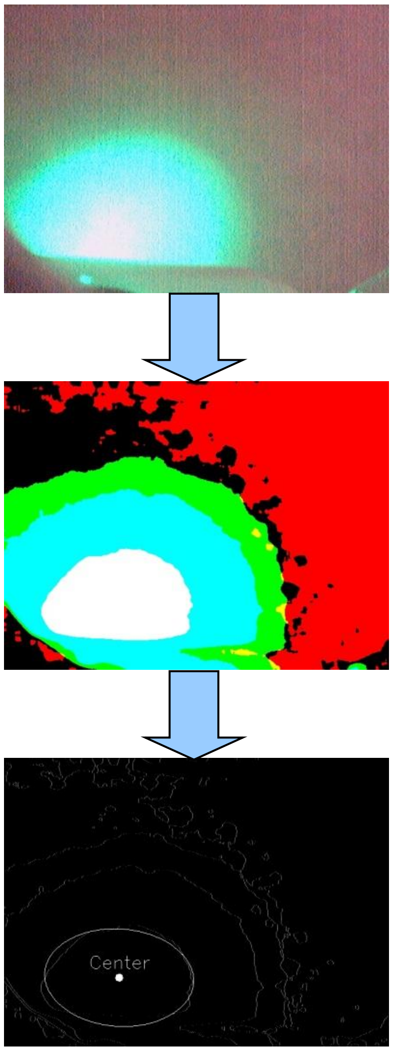

Figure 11 - Image Processing for Green Light

Matching the proprioceptive data with the right image was a relatively short process. The first step was to take the difference between the time stamp of the first image and the time stamp of the first piece of recorded proprioceptive data. This was used as an estimated delay. The next step was to search through the data file and find the time stamp that was closest to the current picture time

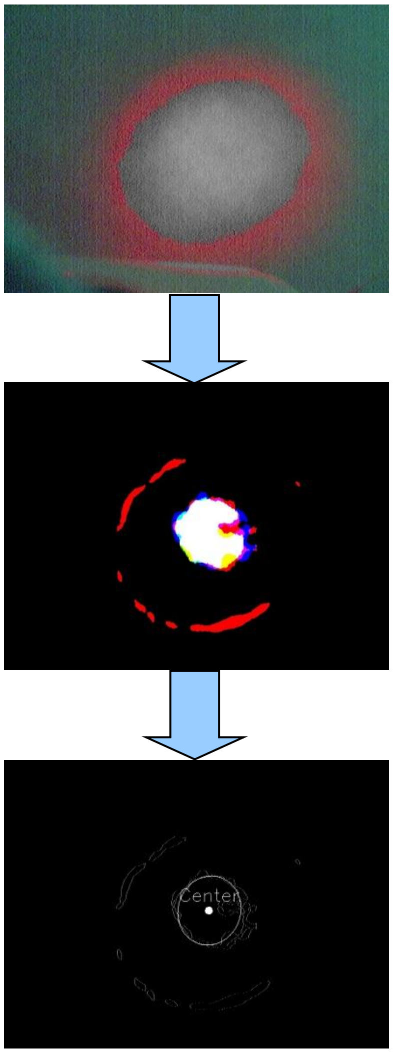

Figure 12 - Image Processing for Red Light

stamp plus the delay. The proprioceptive time stamp with the least difference was used as the proprioceptive data for the current image. This proprioceptive data was then output to a text file and was followed by the center of light for the current image.

Two different files were created from the vision and proprioceptive data for each flashlight, a data.txt and a test.txt. The data file contained 80% of the trials and was used as the robot's "memory". The test file contained the remaining 20% of the trials and was used to verify the joint positions calculated from the data file using data cross-validation.

Three different methods were used to find the joint positions for the arm in the test program. The first method was a guess and check. The robot went through data file and randomly selected an XY point in its field of vision that corresponded to known joint positions. This was then compared to the goal XY point, generated randomly from the test file. One goal for our algorithm was to be more accurate than this random process.

The second method was a simple closest point method. The robot selected the XY point from the data file that was closest to the randomly generated goal point from the test file and moved to the corresponding joint positions.

The third method was a “3 Nearest Neighbors” calculation. A goal XY point was selected from the test file. Then, the three closest XY points to the goal point were selected. The centroid of the triangle formed by these three points was calculated. The joint positions for the center of the triangle were then calculated by averaging the three values for each joint, and the robot moved its arm to this location under the assumption that these joint positions would illuminate the center of the triangle. This XY point was named the target point and the calculated joint positions were named the target joint positions.

These methods utilized data cross-validation to determine if our method was accurate. After the program calculated the target XY point and joint positions, MatLab was used to find the average percent error for each joint, the average percent error for the whole arm position, the average percent error for each method for each flashlight, and finally the average percent error for each method.

From the percent error data, we were able to create maps of the XY visual field showing the relative differences between percent errors across the visual plane. This also allows us to easily compare different methods and flashlights. These maps are included in Section 4.3.

2.3 - Pseudo Code

Visual Analysis

```
AVERAGEBACKGROUNDS()
for i ← 0 to backgrounds.size() - 1 do
      averageBackground ← averageBackground * (numImages - 1) /
             numImages + backgrounds[i] / numImages
end for
return averageBackground

FINDCENTEROFLIGHT(currentImage)
absDifference(currentImage – averageBackground)
blurSmooth(currentImage)
gaussianSmooth(currentImage)
threshold(currentImage)
convert2Grayscale(currentImage)
getCannyEdges(currentImage)
contourList[] ← findContours(currentImage)
maxIndex ← 0
for i ← 0 to contourList.size() – 1 do
      if area(contourList[i]) > area(contourList[maxIndex])
            maxIndex ← i
      end if
end for
return contourList[maxIndex].center

MATCHPROPRIOCEPTIVEDATA(image, proprioceptionData)
decreasingDifference ← true
positionTimestamp ← proprioceptionData.getNextTimestamp()
jointAngles[] ← proprioceptionData.getNextJointAngles(numJoints)
difference ← positionTimestamp - image.timestamp - DELAY
while decreasingDifference == true do
positionTimestamp ← proprioceptionData.getNextTimestamp()
      if (positionTimestamp – image.timestamp - DELAY) < difference do
            difference ←  positionTimestamp - image.timestamp – DELAY
            jointAngles ← getNextJointAngles(numJoints)
      else
                  //don't use these joint angles
                  getNextJointAngles(numJoints)
                  decreasingDifference ← false
      end if
end while
data[numJoints + 2]
```

```
data[] ← jointAngles
data[numJoints] ← image.center.x
data[numJoints + 1] ← image.center.y
return data[]
```

Testing

```
GUESSANDCHECKMETHOD()
//select random target point from test file
for i ← 0 to random position in test file do
    test ← xy point and joint positions
end for
//select chosen point at random
for i ← 0 to random position in data file do
    data ← xy point and joint positions
end for
fprintf(trial number, data, test)
printf(trial number, data)
return

CLOSESTPOINTMETHOD(){
//select random target point from test file
for i ← 0 to random position in test file do
    test ← xy point and joint positions
end for
//select closest point from data
for i ← 0 to length of data file do
    if distance < previous_minimum_distance do
        data ← xy point and joint positions
        previous_minimum_distance ← distance
    end if
end for
fprintf(trial number, data, test)
printf(trial number, data)
return
```

```
INTERPOLATIONMETHOD()
//select random target point from test file
for i ← 0 to random position in test file do
    test ← xy point and joint positions
end for
//select 3 closest points from data and "interpolates" goal
for i ← 0 to 3 do
    min_distance ← 800 //corner to corner of field of view
    for j ← 0 to length of dataset do
        if j == 0 do
            closest[j] ← xy point and joint positions
            closest_distances[j] ← distance
            min_distance ← distance
        else
            if distance < min_distance && distance >
            closest_distances[j-1] do
                closest[j] ← xy point and joint positions
                closest_distances[j] ← distance
                min_distance ← distance
            end if
        end if
    end for
end for
//interpolate target (finds WAM angles and target point)
for i ← 0 to 7 do
target.joints[i] ← (closest[0].joints[i] + closest[1].joints[i] +
    closest[2].joints[i]) / 3.0
end for
target.x ← (closest[0].x + closest[1].x + closest[2].x) / 3.0
target.y ← (closest[0].y + closest[1].y + closest[2].y) / 3.0
fprintf(trial number, target, test)
printf(trial number, target)
return
```

Error Calculation

```
GETINDIVIDUALJOINTPERCENTERROR()
for i ← 0 to nPositions do
      for j ← 0 to nJoints do
                  JointPercentError[i][j] ← abs((calcPosition[i][j]-goalPosition[i][j])/goalPosition[i][j])*100)
      end for
end for
return JointPercentError[][]

GETPOSITIONPERCENTERROR()
for i ← 0 to nPositions do
      sum ← 0
      for j ← 0 to nJoints do
                  sum ← sum + JointPercentError [i][j]
      end for
      PositionPercentError[i] ← sum/nJoints
end for
return PositionPercentError[]

GETFLASHLIGHTMETHODPERCENTERROR()
sum ← 0
for i ← 0 to nPositions do
      sum ← sum + PositionPercentError [i]
end for
FlashlightMethodPercentError ← sum/nPositions
return FlashlightMethodPercentError

GETMETHODPERCENTERROR()
sum ← 0
for i ← 0 to nFlashlights do
      sum ← sum + FlashlightMethodPercentError [i]
end for
MethodPercentError ← sum/nFlashlights
return MethodPercentError
```

3 - Evaluation

3.1 – Goals

This project's goals are developmental in nature and build off each other.

Goal 1) Repeat the process with different flashlights and bulb types.
Goal 2) Have the robot be able to shine the light beam on a given position.
Goal 3) Have the robot self-detect control of the changing light in its field of vision.

3.2 –Definition of Success

Goal 1:

Goal 3 ensures the universal applicability of our algorithms since the different bulbs will alter RGB values of illuminated areas. In this stage in particular, we will make any modifications in the algorithms as necessary to solve problems we will surely encounter during the experiments. Success will be defined as the ability for the robot to complete Goal 3 with different flashlights with different bulb types.

Goal 2:

The robot will have "learned" how the flashlight is manipulated through its preliminary data collection. With the gained knowledge of the relationship between proprioceptive data and visual data from previous experiments, the robot will be able to adjust the joint positions to direct the light beam to illuminate the goal point. By considering the location of the beam to be the center of the light, the algorithms will permit for a fairly large margin of error.

Goal 3:

With real-time analysis of visual and proprioceptive data, we will develop algorithms to relate joint positions to visual changes caused by the moving light beam. From this data, we should be able to obtain consistent estimates of the time difference between motor commands (efferent signals) and visual movements (afferent signals) to find the efferent-afferent delay [3].

4 - Results

4.1 - Data Analysis

We experienced some very intriguing results when collecting our data. We noticed that the white LED flashlights and the Maglite showed up on the robot's cameras as one would expect:

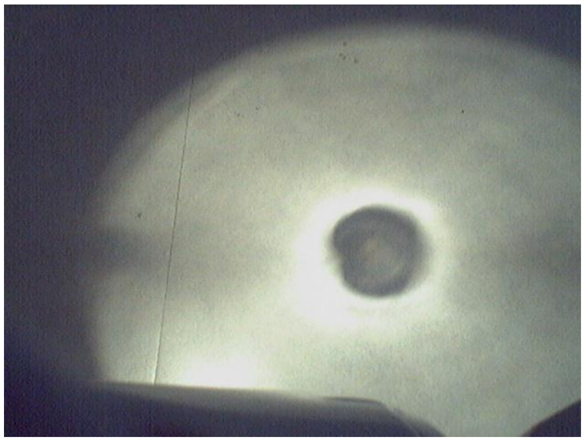

Figure 7 - Maglite (Unfocused)

Figure 8 – Maglite (Focused)

Figure 15 - Silver LED Flashlight (White LEDs)

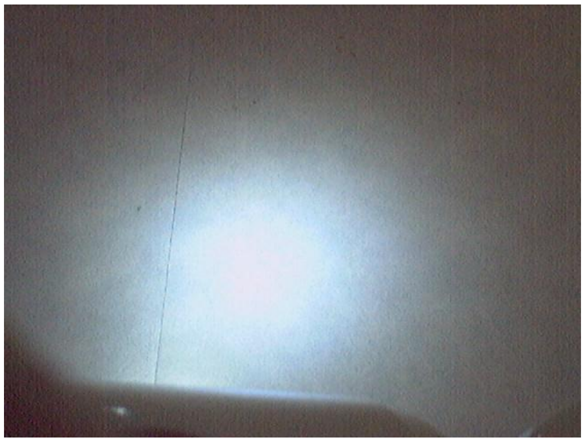

Figure 10 - Yellow Flashlight (White LEDs)

However, when the red and green beams of the small silver LED flashlight were used, the images detected by the robot's camera were very interesting:

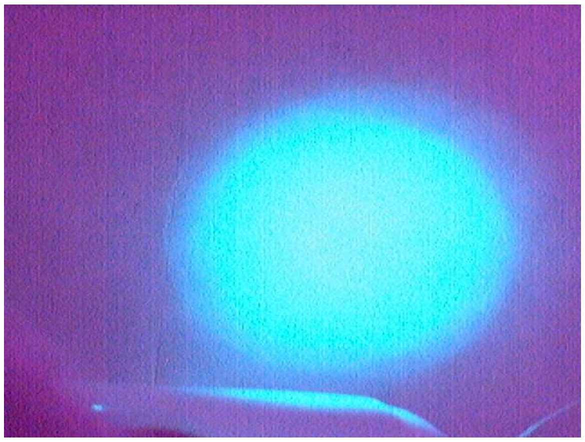

Figure 17 - Silver LED Flashlight (Green LEDs)

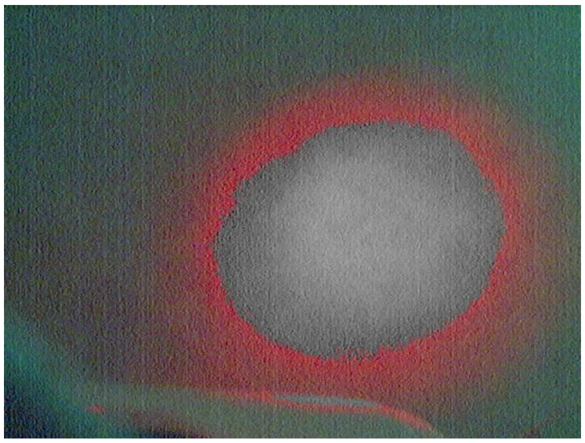

Figure 18 - Silver LED Flashlight (Red LEDs)

As you can see, the green LEDs produced a vivid electric blue illumination against the background. The red LEDs produced an even more interesting effect. The middle of the illuminated area is not detected by the robot's camera. We're not sure why, but since a ring of illumination is still detected, our algorithm for processing the data still works.

4.2 - Test Results

We tested our method using data cross-validation. This allowed us to test our methods in a non-real time environment using the data from a single session in the lab. This allowed us to compare calculated joint positions to actual joint positions and find the percent error between the two. This gives a great representation of how accurate the method is.

We had originally planned on testing our method on the robot by using it to see if the target joint positions illuminated the target xy point. This would confirm proper calculation of the joint positions. We then wanted to see how close the target xy point was to the randomly generated goal point. However, since we were not able to develop a real time approach in the time allotted, we tested the method via data cross-validation.

4.3 - Error Calculation

A MATLAB analysis of our data showed that both of our methods were very accurate. The Interpolation method was the most accurate, followed closely by Closest Point, and both were significantly better than Guess and Check. The average percent errors of the flashlights (Table 1) are closely clustered, suggesting that our OpenCV visual analysis algorithms were able to handle the differences between the beam types, such as focused and unfocused, and beam color, such as red, green, and white. With these overwhelmingly positive results, it is evident that our methods could be extended into real time exploration methods for robots.

Flashlight	Beam	Interpolation	Closest Point	Guess and Check	Average
Silver LED	White	2.111699 %	2.535617 %	4.618721 %	3.088679 %
Silver LED	Green	1.743071 %	2.247754 %	4.181938 %	2.7242543 %
Silver LED	Red	1.929001 %	2.218931 %	4.347955 %	2.8319623 %
Yellow LED	White	1.741829 %	2.287456 %	4.297386 %	2.775557 %
MagLite	Focused	2.162559 %	2.509875 %	3.738037 %	2.8034903 %
MagLite	Unfocused	2.000364 %	2.295011 %	4.19247 %	2.8292817 %
Average		1.9480872 %	2.3491073 %	4.2294178 %	

Table 1 – Percent Errors of Flashlights and Methods

The following contour maps plot the percent error of data over the XY visual plane for each flashlight. The vertical and horizontal scales are the pixels of the field of view, and the color scale is the percent error. One should note that not all the pixel scales are the same. This is because the random exploration was different for each set of trials.

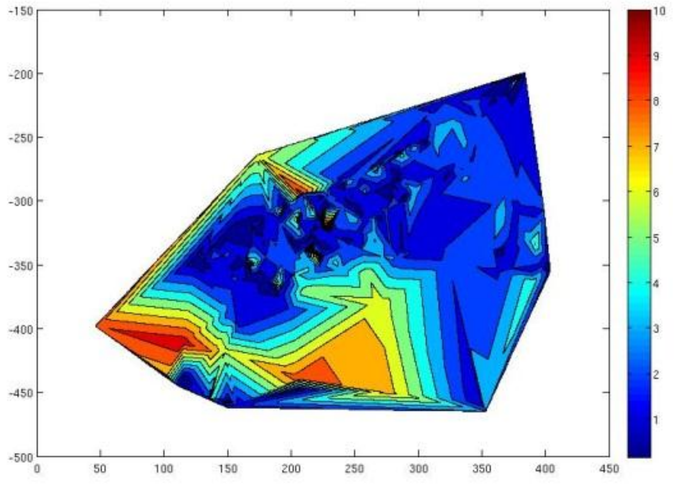

Figure 19 - Silver LED Flashlight (White LEDs)

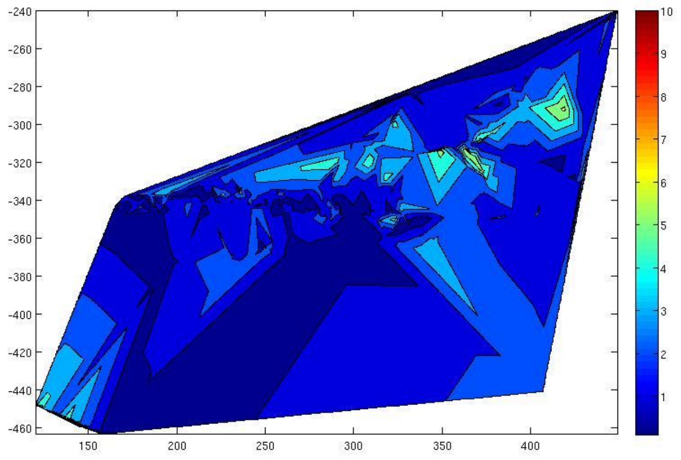

Figure 20 - Yellow LED Flashlight (White LEDs)

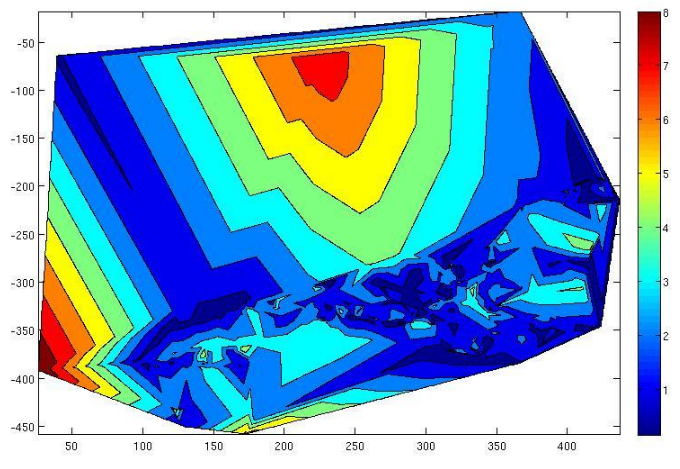

Figure 11 - Silver LED Flashlight (Green LEDs)

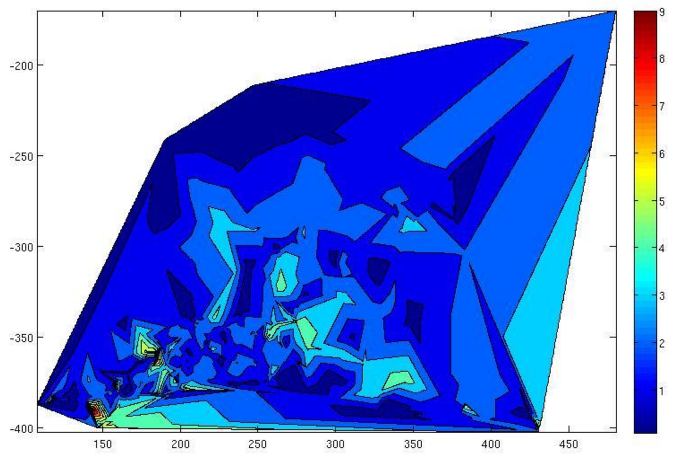

Figure 22 - Silver LED Flashlight (Red LEDs)

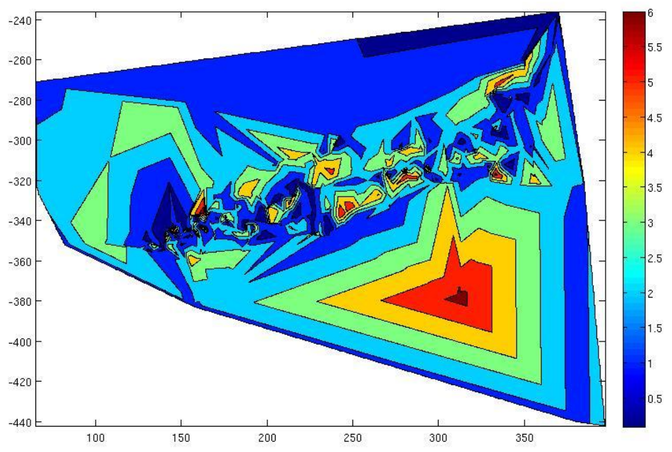

Figure 23 - MagLite Flashlight (Focused)

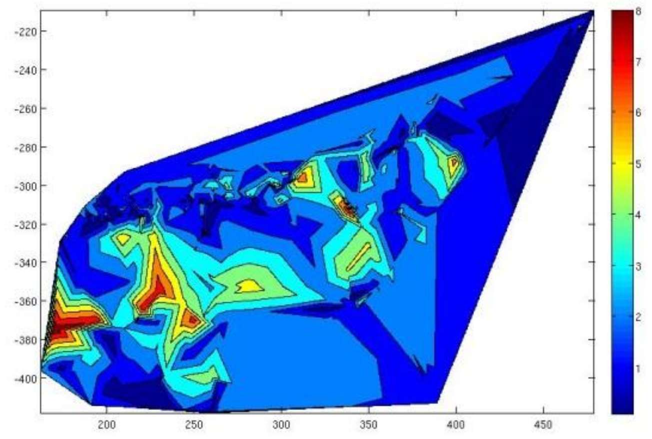

Figure 24 - MagLite Flashlight (Unfocused)

The next three contour maps plot the percent error of data over the XY visual plane for each algorithm used. These figures illustrate that both the Interpolation and Closest Point methods have significantly lower error rates than the guess and check method.

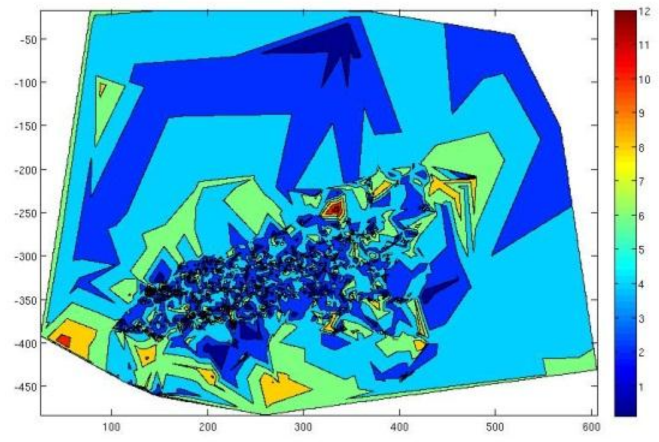

Figure 25 - Guess and Check

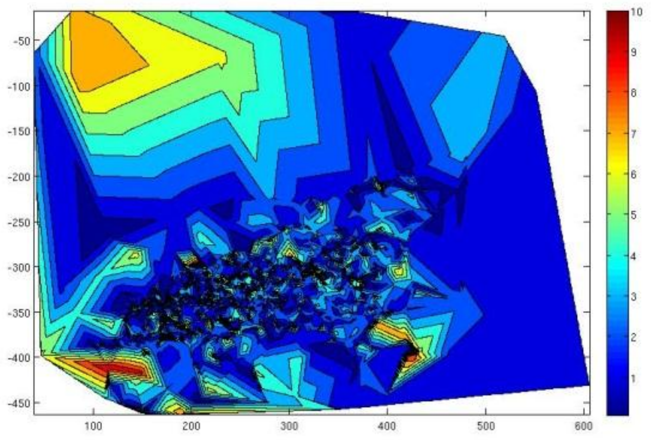
Figure 26 - Closest Point

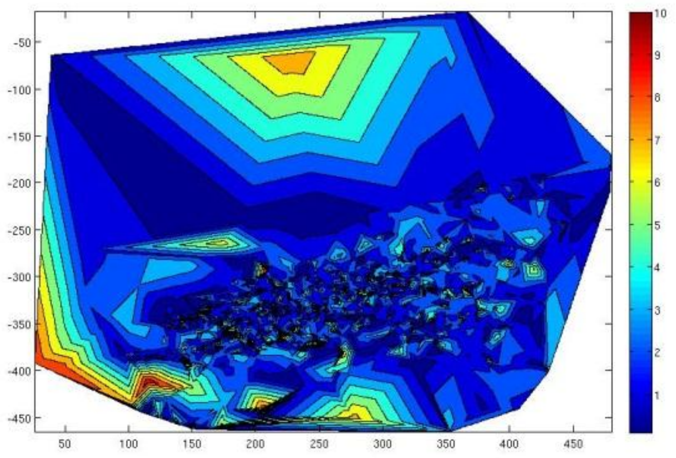
Figure 27 – Interpolation

4.4 - Success

We successfully met two of our three goals in this experiment. Our second goal, which was to program the robot to move its arm to control a flashlight, was met. While we did not get into the lab, our data cross-verification shows that our method produced very low deviations from known joint positions when calculating how to move the arm. Our "Interpolation" method had an average 1.948% error from the known joint positions, which was significantly better than the "Guess and Check" method's average 4.229% error. This means that, using our algorithm, the arm would have been in a position nearly identical to the positions that were known to illuminate the goal xy point. When you consider that a flashlight produces a very wide beam of light, this means that the goal xy point would certainly have been illuminated in a real world test.

We also met our third goal, which was to create a method robust enough to deal with different colors and types of flashlights. Even though the red and green flashlights created very odd reading on the robot's camera, we were still able to accurately calculate the robot's arm movements.

Unfortunately we were not able to meet our first goal. There are two main reasons for this. First, we greatly miscalculated our timeline. We were not able to get into the lab to collect our preliminary data as early as we wanted. We subsequently did not have enough time to create a real time version of our method. While this would have been an excellent addition to our experiment, the data cross-validation is more than sufficient to prove the idea behind our method. We should, however, complete a real time method in the future, as our algorithms may prove less accurate in real time.

We also were not able to incorporate the self-detection part of our project into the experiment. Again, because we were not able to get into the lab as early as we wanted, we did not have time to incorporate this portion of the project into the experiment.

Our timeline was simply too ambitious and unrealistic. Considering that had to learn new programming languages and how to use a very complex robot, we did not give ourselves enough to complete the project as designed. In reality, it took us the first four weeks to develop our algorithms and collect preliminary data. The fifth week and half of the sixth week were spent processing the data and finding results, and the last half of the sixth week was spent writing the paper.

4.5 - Future Work

Short-Term Research

The first extension of this research would be to develop a real time program to run the method on the robot. This would allow a greater variety of tests to be run, and self-detection could be implemented. We hope to be able to continue this portion of the research over the summer through the REU program at ISU.
One of the possible tests we would like to see done is to detect movement and move the arm to illuminate it. This would be quite difficult, but the results could be quite rewarding. This would fairly easy to implement using our developed image processing technique. A video stream would be processed on the fly, and if a difference was detected, the robot would move it's arm to illuminate the center of the movement. This could then be expanded to follow a continually moving target

This research could also be combined with the button recognition and button pressing algorithms to allow the robot to use one hand to hold a flashlight to guide its other hand to press doorbells in low light to no-light conditions [1][2].

It would also be beneficial to experiment with adapting our method to a moving field of view. Currently, the method only works on a stationary field of view (i.e. the head isn't moving). Being able to move the head and still manipulate the flashlight accurately would be quite challenging as an additional step would need to be conceived to account for the rotation of the head.

As described in Section 1.1, it is not always ideal to have a single robot perform an operation independently. Research could be done in having multiple robots working together to accomplish an objective. For example, give one robot a flashlight to illuminate a button or switch across the lab while another robot handles pressing the button or switching the switch.

Long-Term Extensions

A long-term extension for this research, with application of future research topics discussed above, would be the utilization of full humanoid robots to assist police officers in chasing and apprehending fugitives in nighttime scenarios. These robots could also work as security guards, a scenario where the motion detection mentioned in the short-term research would be quite helpful

Another similar extension would be the use of robots for search and rescue missions. A robot with only night vision and infrared sensors would likely frighten the victim and increase the likelihood of injury or death. The ability to utilize flashlights would make the robots seem more familiar and would likely make the victim more comfortable and calm. We are not saying that robots with night vision are inherently bad, just that there are some situations in which a flashlight would be better.

Robots will certainly be used in the household someday, and as everyone knows, the lights in a house are not always on. There will certainly be times when a robot will need to be able to see in the dark. We contended that flashlight manipulation the best solution to this problem due to cost and the relationship between robots and humans. Equipping a robot with the knowledge to learn to use a flashlight is much less costly than equipping the same robot with an infrared camera or a 3D laser scanner. Also, a robot navigating the dark with a flashlight is much less scary than a robot that can navigate a dark household with no visible light.

5 - References

[1] Sukhoy, V. and Stoytchev, A., "Learning to Detect the Functional Components of Doorbell Buttons Using Active Exploration and Multimodal Correlation," In Proceedings of the 10th IEEE International Conference on Humanoid Robots (Humanoids), Nashville, Tennessee, December 6-8, pp. 572-579, 2010.

[2] V. Sukhoy, J. Sinapov, L. Wu, and A. Stoytchev, “Learning to press doorbell buttons,” in Proc. of ICDL, 2010, pp. 132–139, 2010.

[3] Stoytchev, A., “Self-Detection in Robots: A Method Based on Detecting Temporal Contingencies,” *Robotica*, volume 29, pp. 1-21, 2011.

[4] Stoytchev, A., "Behavior-Grounded Representation of Tool Affordances," In Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 3071-3076, Barcelona, Spain, April 18-22, 2005.

[5] Lewis, Michael, and Jeanne Gunn. *Social Cognition and the Acquisition of Self*. New York: Plenum Press, 1979.

[6] Hoffman, G. and Breazeal C., “Anticipatory perceptual simulation for human-robot joint practice: Theory and application study,” In *AAAI*, pp. 1357–1362, 2008.

[7] S. LaValle, B. Simov, and G. Slutzki. “An Algorithm for Searching a Polygonal Region with a Flashlight,” In Proceedings of the Sixteenth Annual Symposium on Computational Geometry , Hong Kong, June 12-14, pp. 260-269, 2000.