USAGE OF COMPUTER VISION AND MACHINE LEARNING TO SOLVE 3D MAZES

BY

VISHNU NATH KAMALNATH

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in the Graduate College of the University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Advisers:

Professor Stephen Levinson

Assistant Professor Paris Smaragdis
Abstract

This thesis deals with incorporating artificial intelligence into a humanoid robot by making a cognitive model of the learning process. The goal is to “teach” a specialized humanoid robot, the iCub robot, to solve any puzzle, wherein a ball of a given color would be placed at the ‘start’ position of the maze, and the robot would navigate the ball through obstacles and get the ball to the ‘finish’ position. The robot would be able to move the ball through the maze by physically tilting the base of the puzzle with its hand. In the process, the robot would utilize the most efficient way possible. If no possible path exists, the robot would not begin to solve the maze.

The first approach was to test the feasibility of the project and an open loop offline-learning algorithm was used to test if the robot could physically solve a given maze. Once this proved successful, the robot was then given multiple mazes that were labeled with the best path, so that it would be able to pick up on the ideal policy on its own, as a result of supervised learning. Once sufficient training was provided, the robot was tested on multiple patterns of mazes that were not seen beforehand by the robot. The robot correctly solved all test mazes that were given to it, giving it a final accuracy rate of 100%.
Acknowledgments

First and foremost, I would like to thank my advisors, Dr. Stephen Levinson and Dr. Paris Smaragdis, who have supported me throughout my research with their patience and deep knowledge of the subject. They have always been there to support me and to give suggestions to improvise my research project. I attribute the level of my thesis to their constant effort and support, and without them, this thesis would not have been completed. One cannot hope for more distinguished thesis advisors in the field of machine learning.

I would like to thank Luke Wendt, a Ph.D. student and a good friend, who has always been supportive of me since the very first days of my research. He recommended great books to aid me in my research and to improve my understanding of the harder concepts in machine learning and statistics. He was also my partner in the computer vision course and a lot of the progress that was made in this research experiment is directly related to the vision course.

This project required a lot of tools, especially for modeling and simulation. It was Aaron Silver, a Ph.D. student and another good friend, who taught me how to use these tools. It was Aaron who patiently taught me how to begin programming a humanoid robot and spent hours on end scrutinizing his own code for me to learn.

Last but not least, I would like to thank my parents for their constant support and encouragement. It was they who provided me with all the resources to become a good student and also encouraged me to enter the world of academia.
Table of Contents

CHAPTER 1. INTRODUCTION ........................................................................................................................................... 1
CHAPTER 2. ROBOT KINEMATICS ................................................................................................................................... 2
CHAPTER 3. COMPUTER VISION ....................................................................................................................................... 8
  3.1 Offline Analysis of Maze ........................................................................................................................................ 8
  3.2 Online Analysis of Maze ....................................................................................................................................... 10
  3.3 RGB Threshold Selection ....................................................................................................................................... 11
  3.4 Selection of Appropriate Grid Size ..................................................................................................................... 12
CHAPTER 4. MACHINE LEARNING .................................................................................................................................... 14
CHAPTER 5. EXPERIMENTAL RESULTS ............................................................................................................................ 17
  5.1 Open Loop Test ..................................................................................................................................................... 17
  5.2 Closed Loop Test .................................................................................................................................................. 18
  5.3 Figures Showing the Final Path .......................................................................................................................... 25
CHAPTER 6. CONCLUSION ................................................................................................................................................ 37
  6.1 Recommended Future Studies ............................................................................................................................ 37
REFERENCES ........................................................................................................................................................................ 39
CHAPTER 1. INTRODUCTION
The field of robotics has improved in leaps and bounds over the past half century. Robots have gone from mere characters in science fiction novels to indispensable servants in several fields. Assembly lines, previously staffed by humans, are now giving way to industrial robots that perform the same task hundreds of times faster, with an even better accuracy rate. As a result, it is more imperative than ever before to design robots that can perceive their surroundings and intelligently work within their environment, rather than being programmed for every possible case by a human programmer.

The celebrated science fiction author Isaac Asimov has written quite a few works in the field of Artificial Intelligence. In his book “I, Robot”, Asimov talks about how robots would play a part in our daily lives in the near future. He goes onto predict that one day robots might be capable of emotion and they may love or hate their masters, in effect mimicking the behavior of any random human being [1]. I felt that this concept was very intriguing and it was this novel that got me into the field of Artificial Intelligence.

This thesis explores the physical manipulation of the three axes of the plane of the board on which the maze is built, to roll the ball from the start point to the target or end point. The challenging part of this study is that the robot is not pre-programmed to solve the maze in question, but rather learns/infers that the goal is to solve the maze using the shortest path. Q-Learning (SARSA) was used as the learning method and the robot used was the iCub humanoid robot. Chapter 2 speaks about the robot, its DH parameters and other robot hardware. Chapter 3 discusses the computer vision aspect of the project and the approaches used to solve the vision problem, while Chapter 4 discusses the Q-learning approach and the results obtained.
CHAPTER 2. ROBOT KINEMATICS
The iCub robot has 53 degrees of freedom. However, for the task to be accomplished, the right hand is of utmost importance since the gun would be held in the right arm. In order to get the right hand raised to the appropriate level, the first task is the computation of the DH matrices of all the joints from the robot center to the tip of the right hand. Then, the transformation matrices need to be determined in order to apply the principles of inverse kinematics. The reference frame of all the joints in the torso of the iCub is given in figure 1 given below [2].

![Image of iCub robot with axes indicated]

Figure 1: The x axis is in red, the y axis is in green and the z axis is in blue

Furthermore, the default orientation of the right palm is given in figure 2 below, with the same color coding as above for the axes.

This chapter is taken from previously published work from the degree candidate. The work has been published at the AAAI Spring Symposium 2013, Palo Alto, CA titled “Learning to Fire at Targets by an iCub Humanoid Robot”. The authors have given permission to reprint their work. [3]
Also, the location of the origin of the coordinate reference frame for the iCub is given in figure 3, as shown below [2].
Since the origin has been obtained, the next step is the computation of the DH parameters of the right arm in default position [4]. They have been tabulated in table 1 given below [3].

<table>
<thead>
<tr>
<th>Link</th>
<th>(a)</th>
<th>(d)</th>
<th>(\alpha)</th>
<th>(\theta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>0</td>
<td>(\pi/2)</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-5.5</td>
<td>(\pi/2)</td>
<td>-(\pi/2)</td>
</tr>
<tr>
<td>3</td>
<td>-23.467</td>
<td>-143.3</td>
<td>(\pi/2)</td>
<td>-(\pi/2)</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>-107.74</td>
<td>(\pi/2)</td>
<td>-(\pi/2)</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>-(\pi/2)</td>
<td>-(\pi/2)</td>
</tr>
<tr>
<td>6</td>
<td>-15</td>
<td>-152.28</td>
<td>-(\pi/2)</td>
<td>-(\pi/2)</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>0</td>
<td>(\pi/2)</td>
<td>(\pi/2)</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>-137.3</td>
<td>(\pi/2)</td>
<td>-(\pi/2)</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>(\pi/2)</td>
<td>(\pi/2)</td>
</tr>
<tr>
<td>10</td>
<td>62.5</td>
<td>16</td>
<td>0</td>
<td>(\pi)</td>
</tr>
</tbody>
</table>

A simulation was run on MATLAB to double check whether the DH matrices were computed correctly. The simulation is shown in figure 4 below.
As can be seen from figure 4, all the values in table 1 match the anticipated values, and the right arm is in rest along the torso of the iCub robot. This led to the conclusion that the DH parameters have been computed correctly. The next step was to compute the homogenous transformation matrices and the final transformation matrix $T^0_{10}$ [3]. The matrices were computed to be as follows:

\[
T^0_1 = \begin{bmatrix}
0 & 0 & -1 & 32 \\
0 & -1 & 0 & 5.5 \\
-1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
T^0_2 = \begin{bmatrix}
0 & -1 & 0 & 175.3 \\
1 & 0 & 0 & -17.967 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
T^0_3 = \begin{bmatrix}
1 & 0 & 0 & 175.3 \\
0 & 0 & -1 & -17.967 \\
0 & 1 & 0 & -107.74 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Using the aforementioned transition matrices, we proceeded to get the right arm into firing position by having the shoulder roll, elbow and wrist yaw set to $\pi/2$. The simulation result was a perfect match to what we wanted and is shown in figure 5.

\[T_4^0 = \begin{bmatrix} 0 & 0 & 1 & 175.3 \\ 0 & 1 & 0 & -17.967 \\ -1 & 0 & 0 & -107.74 \\ 0 & 0 & 0 & 1 \end{bmatrix}\]

\[T_5^0 = \begin{bmatrix} 0 & -1 & 0 & 175.3 \\ -1 & 0 & 0 & -17.967 \\ 0 & 0 & -1 & -107.74 \\ 0 & 0 & 0 & 1 \end{bmatrix}\]

\[T_6^0 = \begin{bmatrix} 1 & 0 & 0 & 160.3 \\ 0 & 0 & -1 & -17.967 \\ 0 & 1 & 0 & 44.54 \\ 0 & 0 & 0 & 1 \end{bmatrix}\]

\[T_7^0 = \begin{bmatrix} 0 & 0 & 1 & 160.3 \\ 0 & -1 & 0 & -17.967 \\ 1 & 0 & 0 & 59.54 \\ 0 & 0 & 0 & 1 \end{bmatrix}\]

\[T_8^0 = \begin{bmatrix} 0 & 1 & 0 & 23 \\ 1 & 0 & 0 & -17.967 \\ 0 & 0 & -1 & 59.54 \\ 0 & 0 & 0 & 1 \end{bmatrix}\]

\[T_9^0 = \begin{bmatrix} 1 & 0 & 0 & 23 \\ 0 & 0 & 1 & -17.967 \\ 0 & -1 & 0 & 59.54 \\ 0 & 0 & 0 & 1 \end{bmatrix}\]

\[T_{10}^0 = \begin{bmatrix} -1 & 0 & 0 & -39.5 \\ 0 & 0 & 1 & -1.967 \\ 0 & 1 & 0 & 59.54 \\ 0 & 0 & 0 & 1 \end{bmatrix}\]
From the positioning of the right arm in figure 5, it can be inferred that all calculations performed are accurate and would deliver all the anticipated results. This concludes the chapter on humanoid kinematics.
CHAPTER 3. COMPUTER VISION

3.1 Offline Analysis of Maze

First the maze must be studied to obtain a control policy. This is done by analyzing a top down view of the maze. Since it is very difficult to provide a perfect orthographic view of the maze, an inverse homography must be applied to this step as well. Therefore the first thing that must be done is identification of features with known geometric relation to each other. At least four must be present to determine the inverse homography [4]. The easiest way to go about doing this is to place high contrast color markers on each corner of the maze. The color red was selected because it was not present in the maze or the surrounding background. The ball was also chosen to be red.

In the initial top down analysis no ball will be present. Color thresholding provides a binary image indicating where high concentrations of red are present. Thresholding the original RGB values appear to be overly sensitive. This sensitivity can be removed by first converting to HSV coordinates. A segmentation algorithm, like RasterScan, can be used to label contiguous regions and sort them by size [4]. The four largest regions are expected to be the four corner markers. The geometry of the corners is known and therefore with a bit of logic labels can be applied to the compass markers. The markers are assumed to be related in a square manner from the top down perspective. Using this information, an inverse homography can be computed to obtain the correct top down perspective of the maze. All surrounding border content can be cropped off. The maze wall and open path are the only things that remain after this step. Choosing colors that are easily distinguished, e.g., yellow and green, allows thresholding of the open path. Again, a RasterScan will provide the open contiguous path of the maze. Once a path is obtained, it can be discretized into a grid. Important features in the image need to be tagged, like the start and goal. A high contrast color or a fiducial marker could be used or a manual user interface would work as well. Once a start and finish location are specified, reinforcement learning can
begin. The reinforcement learning simulates trial and error runs of a simulated maze environment. The control actions involve course wrist motion with long pauses. After each action the ball is at rest in one of the corners of the maze. After a sufficiently long run time value iteration converges and an optimal policy is obtained. Filtering of the optimal policy provides more general control domains. The final filtered control policy corresponding to this is then saved for online control [5].

The projected view its labels is shown in figure 6 and its inverse homography is shown in figure 7. As long as the center of the markers and the ball in the projected view are roughly in the same plane as the center of the ball and markers in the top down view, their transformations should be isomorphic.

![Figure 6: Abstraction of maze with markers](image)
3.2 Online Analysis of Maze

Once the optimal control policy is obtained, any maze can be solved online with Bert. The maze is now viewed from a projected perspective. HSV (Hue-saturation-value) color thresholding provides a binary image indicating where high concentrations of red are present, which in turn indicates the location of the corner markers and the ball. A RasterScan segmentation approach returns the largest 5 objects in the field of vision of the robot. With the known geometry of the board on which the maze is built, the markers and the ball can be labeled. From the markers, an inverse homography would return the unprojected coordinates of the maze. This inverse homography is the applied to the ball’s location. After cropping and discretizing the resulting projected image, the ball’s location with respect to the grid and maze can be accurately determined. The application of the optimum control policy used in the offline analysis of the maze would result in the movement of the ball. Due to this design, the robot would move the board along one of the 3 possible axes and wait for a period of time before leveling the board once
again. During this period, the ball would have progressed to another corner of the maze that is closer to the goal.

### 3.3 RGB Threshold Selection

As mentioned before, it is known that the only red colored objects in the field of view are the four corner markers of the board and the ball itself. Therefore, it is important to threshold the RGB values of the image to facilitate the identification of the ball in the image.

The RGB values of various points of the image were picked and are shown in figure 8 below.

![Figure 8: RGB values at various points on the board](image-url)
As can be seen from figure 8, the unmapped RGB values have a very wide variation in value for seemingly similar colors. This indicates that the raw RGB values are not suitable for thresholding, but rather the HSV values are better suited.

### 3.4 Selection of Appropriate Grid Size

The final problem that needs to be tackled, with respect to computer vision, is the resolution of the projected image of the maze i.e. the image that would be the input for online analysis for the robot. It is imperative to find the right resolution for sampling the image.

Sampling below the threshold would cause degradation in the maze and may result in open segments of the maze, when in reality there are none. The resulting learned policy would then fail. On the other hand, sampling above the threshold would produce a finer resolution which would cause an exponential increase in the time taken by the learning algorithm to converge upon a solution. This issue is referred to as the ‘curse of dimensionality’ in literature and is present in all uniformed dynamic programming schemes [6].

Consider figure 9 shown below. The resolution of the images from left to right are 16x16, 32x32 and 50x50 respectively. Clearly, the resolution on the left is too low since information about the ball on the grid would be lost. The robot would be able to identify the location of the ball only in terms of 4 quadrants, which would cause a large number of errors, and is definitely not the anticipated output. The image on the right displays what it is like to have a very high resolution. The location of the ball would definitely be determined, but the convergence of the learning algorithm would take unacceptable time. The image at the middle has a resolution that is a compromise between the two resolutions. This image
divides the maze into a resolution of 32x32, and can be used to determine the location of the ball with respect to the grid with sufficient accuracy and satisfactory running time.
CHAPTER 4. MACHINE LEARNING

It is impossible for a human programmer to program every possible maze combination for the robot to solve. It is possible, however, to come up with a few examples and indicate whether the robot has obtained the correct solution or not using the best possible method. Furthermore, there is a heavy emphasis on the ability of this learning algorithm to be able to perform on-line with reasonable speed and accuracy. This forms the basis of reinforcement learning and is the fundamental algorithm that has been used to enable the robot to determine the shortest path given any maze.

The update equation for temporal difference Q-Learning is given by

\[ Q(s, a) \leftarrow Q(s, a) + \alpha (R(s) + \gamma \max_{a'}Q(s', a') - Q(s, a)) \]  

(4.1)

Where \(Q(s,a)\) is the Q-value of action ‘a’ at state ‘s’, \(R(s)\) is the reward function of the state ‘s’, \(\alpha\) is the learning rate and \(\gamma\) is the discount factor [7]. An examination of (4.1) shows that Q-learning backs up the best Q-value from the state reached in the observed transition. It pays no attention to the actual policy being followed and so is called an off-policy learning algorithm. As a result, there is no point in coming up with the optimum policy to shoot down a target if one is using an unmodified Q-learning approach [8].

Clearly, there is a need to come up with a learning algorithm that utilizes a policy that would maximize the probability of the robot solving the maze. Such an algorithm is of the on-policy type algorithm and is called the SARSA algorithm. SARSA stands for State-Action-Reward-State-Action and utilizes the optimum policy for updating the Q-values. The update equation for SARSA is given by

\[ Q(s, a) \leftarrow Q(s, a) + \alpha (R(s) + \gamma Q(s', a') - Q(s, a)) \]  

(4.2)
The difference between (4.1) and (4.2) is the omission of the max term of the new Q-value. This means that SARSA actually waits until an action is taken and backs up the Q-value for that action [7].

If a greedy agent exists that always the action with the best Q-value, the two algorithms are identical. However, when exploration is needed, the algorithms vary significantly. For the objective of finding the shortest path to solve a maze, heavy exploration (or at least the consideration) of all possible paths is mandatory. Q-learning is more flexible than SARSA, i.e. an agent that learns by Q-learning can behave well even when guided by a random or adversarial exploration policy. However, SARSA is more realistic than Q-learning [9]. For example, if the overall policy is even partly controlled by other agents, it is better to learn a Q-value function for what will actually happen rather than what the agent would like to happen. Since the environment being dealt with has a lot of unknowns, accompanied by several independent agents at work, it is better to use a SARSA approach. The optimum policy is given by equation (4.3).

$$
\pi^* = \arg\max_\pi \sum_h P(h|e)u_h^T
$$

(4.3)

In equation (4.3), the posterior probability $P(h|e)$ is obtained in the standard way, by applying Bayes’ rule on the observations till date. This is how the feedback loop is created that would allow constant improvisation.

The learning for this problem was done with value iteration of a discrete state-action space. The algorithm used a sample based quality space [10]. The specific algorithm used came from [11] and is given below. Here, $\phi$ is an index of discretized space and $\theta$ is the value at that index. The control space was $U = \{0,1,2,3,4\}$ where 0 is a random action and $\{1,2,3,4\}$ is a wrist tilt in the direction {North East, North West, South West, South East} respectively. The state space corresponds to the location in the $n \times$
n discretized path space of the maze. The value of $\alpha$ and $\gamma$ were set to 0.99 and an exploration function of $\epsilon = Q^{-0.01}_{\text{visit}}$ was used.

The pseudo-code of the algorithm is given below.

\begin{verbatim}
Q-Learning with Algorithm 2.3:
\end{verbatim}

\begin{verbatim}
Input: discount factor $\gamma$,
       exploration schedule $\{\epsilon_k\}_{k=0}^{\infty}$ s.t. $\sum_{k=0}^{\infty} \epsilon_k = \infty$,
       learning rate schedule $\{\alpha_k\}_{k=0}^{\infty} \in [0,1]$
       BFs $\Phi_{U_j}: X \times U \rightarrow \mathbb{R}^{p(j)}$,

Initialize parameter vector, e.g., $\theta_0 \leftarrow 0$

Measure initial state $x_0$

For every time step $k = 0, 1, 2, \ldots$ do

\begin{itemize}
    \item $u_k \leftarrow \begin{cases}
        u \in \arg \max_{u'} \phi^T(x_k, u') \theta_{\ell} & P = 1 - \epsilon_k \\
        \text{random action in } U & P = \epsilon_k
    \end{cases}$
    \item Apply $u_k$, measure next state $x_{k+1}$ and reward $r_{k+1}$
    \item $\theta_{\ell+1} \leftarrow \theta_{\ell} + \alpha_k [r_{k+1} + \gamma \max_{u'} \phi^T(x_{k+1}, u') \theta_{\ell} - \phi^T(x_k, u_k) \theta_{\ell}] \phi(x_k, u_k)$
\end{itemize}

end
\end{verbatim}
CHAPTER 5. EXPERIMENTAL RESULTS

5.1 Open Loop Test

The very first step was to determine the feasibility of the problem. The iCub robot was programmed to solve a particular maze in open loop. This first step was crucial, since it allowed us to explore the basic command interface for the problem, as well as demonstrated that the robot is physically capable of rotating the wrist sufficiently to roll the ball in any direction. At this stage, challenges dealing with grasping the board and field of view were addressed. Multiple iterations of the test were run at this stage and the robot completed the task every single time. This provided great confidence in the feasibility of the task. Figure 10 shows the robot during such a trial run during the open loop test.

![iCub during open loop test](image)

Figure 10 : iCub during open loop test
5.2 Closed Loop Test

Once it was determined that the robot could actually solve a given maze, the robot was trained over 20 different types of mazes. With each maze, it was given the start point and end point, and the robot had to determine the optimum policy of the goal, which is to get the ball from the start point to the end point using the shortest path. At the end of this point, the robot was tested on a random maze which it had never seen before. This sub-chapter discusses the results obtained at this stage of the experiment.

The first step was to perform the step of inverse homography. The robot correctly identified the four corners of the board and the determined corners are shown in figure 11 below.

![Corner estimation of the board](image)

Once the four corners have been identified, they were used to perform inverse homography so that it would be more convenient for the robot to see the projections of the ball and the interior walls of the maze. The resultant projected image is shown in figure 12 below.
The ball can be easily identified by performing Hough Circle Transform, an approach similar to that discussed in section 3.3. The identification of the ball in this image is shown in figure 13.
However, as discussed in section 3.3, the raw RGB values cannot be used to determine the location of the ball inside the maze. As a result, at this point in the on-line analysis, the raw RGB image is no longer used. Instead, the image is converted into HSV type and the analysis takes place on that. The resultant conversion of the image into HSV is shown in figure 14.
Upon performing the RasterScan that was mentioned in sections 3.1 and 3.2, the four corners can be obtained. The path threshold of the maze can be easily determined from this image, in addition to the splitting of the image into a resolution of 32 x 32. Figure 15 shows the resultant image.
The robot would then label the path with a starting point and an ending point. The starting point would be labeled in green color, while the ending point would be labeled in red color. Figure 16 shows this visualization on the path below.

![Figure 16: Start and end points labeled](image)

At this stage, the iCub robot applies the normalized log value function to the threshold path. Figure 17 shows the normalized log value function of the path of the current maze, along with its color key.
The last step is to apply the optimal control policy that has been learned from the multiple training iterations. The generated optimal control policy is shown in figure 18 for the current maze.
As can be seen from figure 18, the learning algorithm outputs a path with reasonable accuracy, but the individual grids are susceptible to noise. As a result, it is extremely likely that the robot would have an unstable movement, between the start and end point, that is governed by a random probability distribution function. In order to remove the effect of noise on the movement of the ball, a smoothing operation is performed [13]. This is the final step of the on-line computation and then the robot merely has to follow the output path. Figure 19 shows the output path after the smoothing operation has been performed.
5.3 Figures Showing the Final Path

This sub-chapter contains figures that trace the ball and its associated rule from the start point to the end point. They are shown below in sequential order.
CHAPTER 6. CONCLUSION

The increase in computing power and the development of better vision and learning algorithms have enabled the on-line processing of streams of visual data on a commercial CPU, in contrast to the immense computational power needed to achieve similar results just decade ago. The ability of robots to intelligently alter their environment, based on sensory perceptions, is gaining more traction than ever before. This thesis set out to achieve only a small subset of this vast and even abstract problem. The results confirmed that it is possible for robots to learn from their environments and alter their work environment to make it better. It is only the definition of 'better' that is ambiguous and care must be taken to ensure that the robot/agent does not have the wrong definition of 'better'.

This thesis spoke about every module that needs to be implemented in order to have a humanoid robot solve a 3D maze using a colored ball. While it is relatively easy for a human to control the robot and perform this task, or even pre-program the robot, what is of special interest is the fact that the robot is completely autonomous. Existing learning algorithms were modified, specifically Q-learning and SARSA, to reduce the number of iterations it takes to converge upon the optimum policy. Lastly, this research project enabled me to work with extremely complex devices and write a program that consists of approximately half a million lines of code.

6.1 Recommended Future Studies

The next step for this design is to test this learning experiment on various other objects to see if the robot is capable of grasping the connection between any two objects, and to eventually determine if the ability to make a connection between two new objects can be developed. A good example would be to
see if the robot would be able to screw a light bulb into a socket. After a couple of such trial-and-error experiments, the next step would be to determine if new connections can be made from various combinations of items that the robot has previously seen. This ability is crucial for robots to be able to come up with ‘ideas’ that perhaps humans have not thought of. The first instance of this experiment would be to give the robot a pencil and an old cassette tape that has been jammed. The goal would be to rewind the tape just a little so that it is no longer jammed. As humans, we have discovered that using any cylindrical object (most commonly a pencil) of the right dimensions would achieve this task. The goal is to determine if a robot would be able to make that connection or not. The answer to this question would change the face of artificial intelligence and robotics as we understand and know it today. This is the future work that the author hopes to complete in the future.
REFERENCES


