2016

Machine Learning Through Mimicry and Association

Nickolas S. Holcomb
Georgia Southern University

Machine Learning Through Mimicry and Association

An Honors Thesis submitted in partial fulfillment of the requirements for Honors in Electrical Engineering.

By
Nickolas Sterling Holcomb

Under the mentorship of Dr. Rocio Alba-Flores

ABSTRACT

Adaptability is a key missing features that has impeded the growth of assistive robotics. In the traditional model, all actions must be explicitly coded by a skilled programmer familiar with the hardware. This project explores a method of teaching a webcam equipped arm type robot new primitive movement using visual demonstrations.

Thesis Mentor:_________________________

Dr. Rocio Alba-Flores

Honors Director:________________________

Dr. Steven Engel

April 2016
Electrical Engineering
University Honors Program
Georgia Southern University

1: Acknowledgements

First I would like to thank the faculty and staff of the University Honors Program and Electrical Engineering Department at Georgia Southern University. Your excellent support and instruction has given me an education that has immeasurably improved my life and provided me with the tools to pursue a career in engineering.

Furthermore, I sincerely appreciate the guidance and encouragement Dr. Danda Rawat and Dr. Fernando Rios-Gutierrez. These professors have gone well out of their way to mentor and encourage me in research and the application of theory toward the solution to significant modern challenges.

Finally, this project would not have been possible without the encouragement and mentorship of Dr. Rocio Alba-Flores. Since recruiting me to the Robotics Team in my sophomore year, she has provided continuous guidance and support. You mentored me in my first research project which has now grown into this thesis. Thank you.

2: Introduction

The field of robotics has had an interesting past filled with the constant promise of great things to come. Beginning with the mechanical automata of the eighteenth century and growing with each advance in technology, people have anticipated the release of flexible assistive robots. Progress toward this goal has been a story of continuous innovation. Early clockwork based mechanisms could imitate basic animal behaviors like walking and eating but performed little useful work [1]. Modern electrical components advanced these bio-memetic cybernetic devices to include more complex behaviors such as photophobia but still failed to produce more than curiosities [2]. The introduction of the computer enabled the majority of advancements to date in the field of robotics and was key in the introduction of the first industrial robots. The continued advancement of computer performance coupled with the decrease in their power consumption and cost has resulted in much more capable factory robots [1] but their transition to use by laymen in the home has been slow.

There are several challenges that have impeded the advancement of assistive robotics including robots' abilities to understand and interact with their environments, be social, and focus attention on the correct users [3]. Centering in on the challenge of robots understanding and interacting with their surroundings, we see that there is a need for robots to be able to observe their environment and learn from it. One approach to this problem is imitation learning, where robots are taught new movements by observing and imitating a teacher [4]. This is a significant paradigm shift from the conventional approach to robotics where every behavior a robot will exhibit is explicitly coded by a skilled programmer with intimate knowledge of the hardware. Furthermore, in systems designed for use by the general public, this programming typically happens before production and new functionality is rarely added after purchase.

This study presents a method for teaching arm-type robots new primitive movements using imitation learning where sensing is performed by a single webcam and actions are performed by an arm with known geometry. This is intended to serve as a building block researchers could use to implement more advanced systems that synthesize complex behaviors using these primitive movements.

3: Methodology

The process of teaching an arm-type robots new primitive movements using imitation learning can be broken down into seven consecutive stages: recording the demonstration, detecting hand locations in each frame of the video, rejecting outliers from the noisy detection process, forming a smooth path from the remaining points, scaling that path to fit within the range of the arm’s reach, determining what commands will drive the arm to the desired position, and executing those commands. Since this process is intended to serve as a building block for future systems, each of these stages have been implemented modularly so that they can be modified to reflect changes in technology or implementation hardware.

3.1: Data Acquisition

The robot must have input before imitation learning can begin. In order to keep the system as general and inexpensive as possible, I use a generic webcam with 760 by 1080 pixels captured at 30 frames per second to acquire this requisite input. The camera should be setup so that the instructor's entire movement will be within the frame and normal to the camera with nothing moving besides the instructor. This is intended to be used as an integrated learning system for future movement; accordingly, the video is recorded without requiring real-time processing or for the robot to also be in the frame. Once the demonstration is complete the instructor should trim the video as shown in Figure 1 to ensure that only the intended movement is sent to the robot for learning.

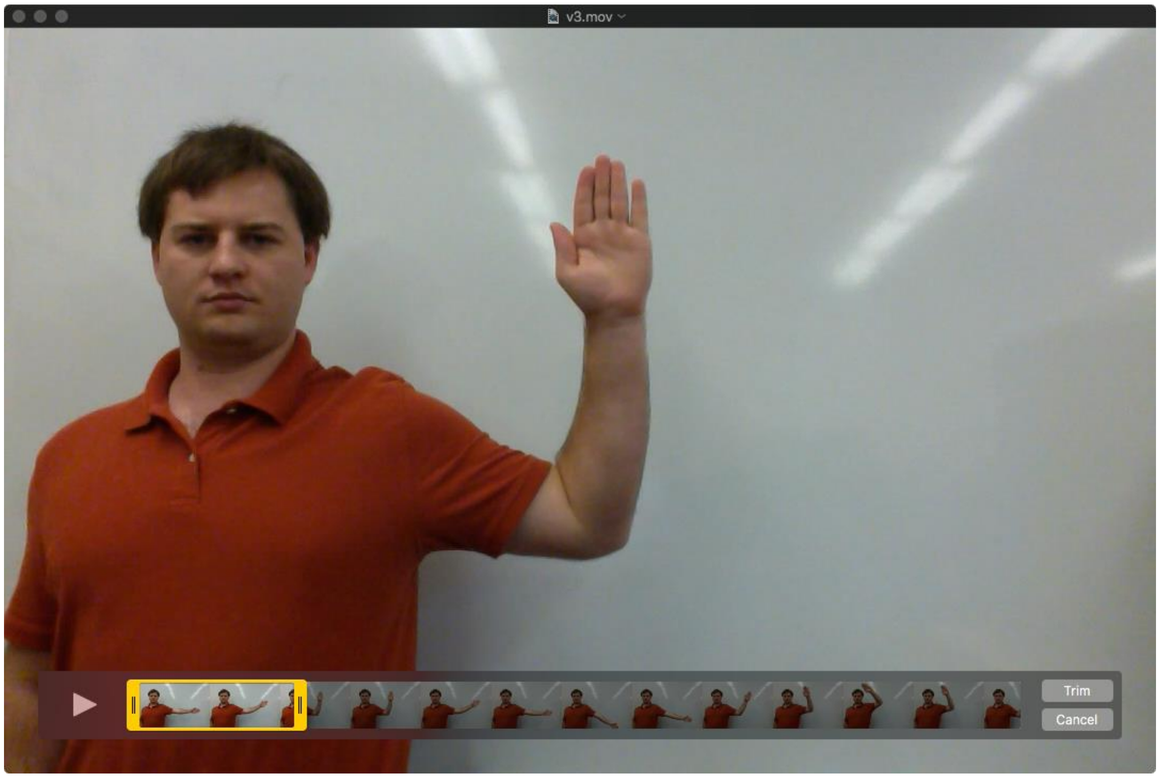

Figure 1: Video Acquisition and Trimming

3.2: Hand Detection

With the goal of mimicking the instructor's time sequence of hand positions with its end effector, the robot must identify the location of the instructor's hand in every frame of the video. There are many algorithms in literature for object detection; one of the most efficient and accurate of these methods is proposed by Viola and Jones in [5]. Their method relies on three key features: the integral image, feature selection, and cascading. The integral image is a processed version of the image's intensity map that calculates the sum of specified regions of the image in order to facilitate rapid feature selection. This feature selection is a classifier that uses a modified AdaBoost algorithm to determine which features in the image are important and examines those regions further in search of the object to detect. Finally, the algorithm uses a sequence of increasingly complex classifiers in cascade to efficiently examine the likely features in increasing detail to determine if they are a hand [5].

When properly trained, a Viola-Jones object detector can produce a very

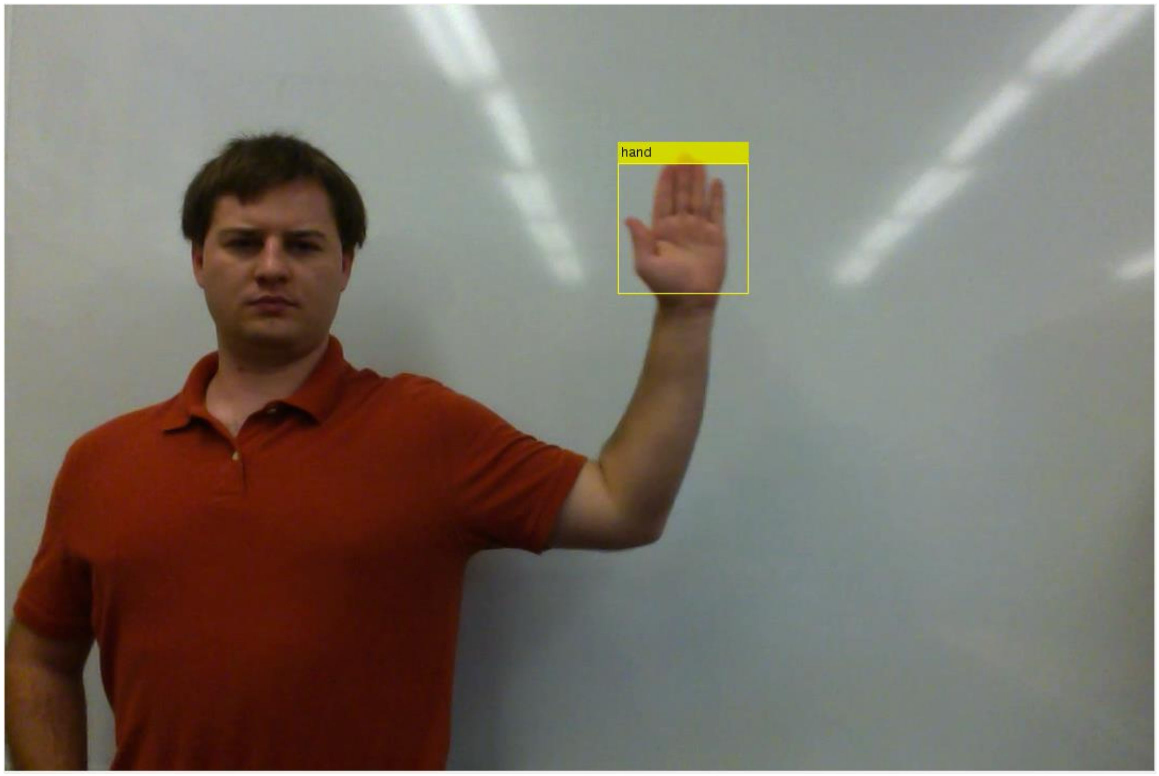

Figure 2: Positive Hand Location Identification

low false acceptance rate; unfortunately, this is accompanied by the typical rise in the false rejection rate and results in some frames lacking detected hand locations. In order to increase the number of accurate data points available, I also trained a second detector to examine the first order derivative of the time sequence. Since the only movement in the image is assumed to be the instructor, this detector is usually very successful. The end product of this detection process is the most probable hand location using data from the static and derivative detectors as shown in Figure 2.

3.3: Noise Reduction

In order to create a robust system, it must be assumed that the signal contains some noise. The main source of this noise is false detections of which there are two types: near miss, off-center detections as in Figure 3 and far off results as in Figure 4. In the near miss scenario, the detector is not centered on

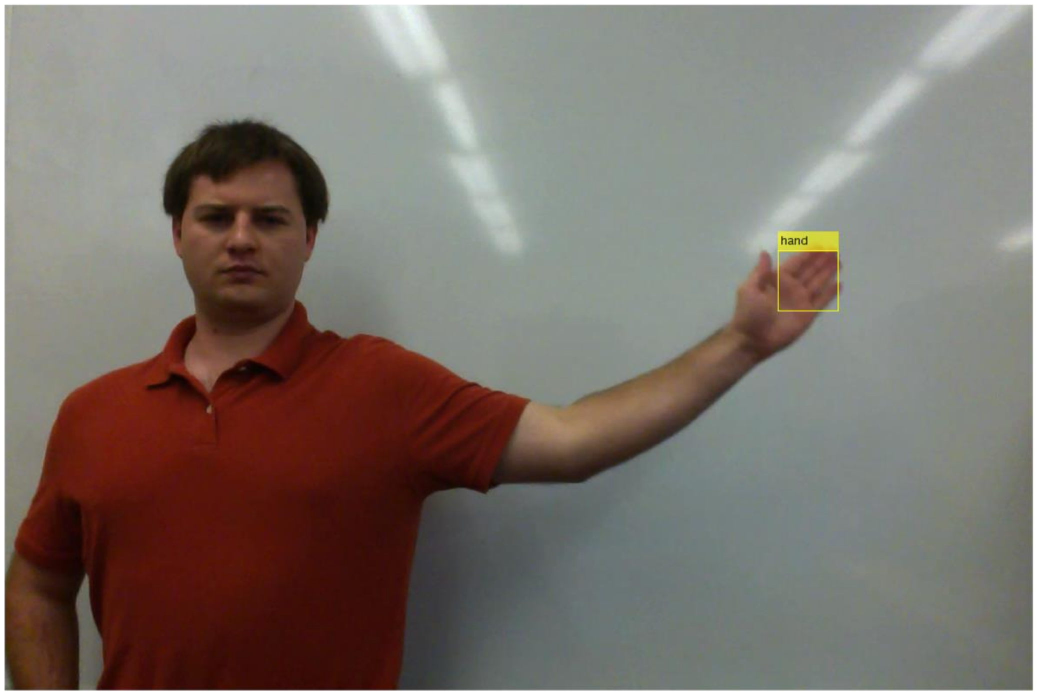

Figure 3: Near Miss Incorrect Hand Location

the hand but is close. This type of error will be corrected by the smoothing operation described in section 3.4. The second type of error is more problematic. False detections that are far from the true hand position will skew the smoothing operation in inappropriate directions. These errors require a separate process to remove.

One way to view the sequence of hand location data points is as an interpolated path. In linear piecewise polynomial interpolation, each point is connected by a straight line. Viewing these lines as link distances, it becomes apparent that the incorrect detections correspond to relatively large link distances. Accordingly, I discard the detection points associated with a link distance that are greater than twice the standard deviation.

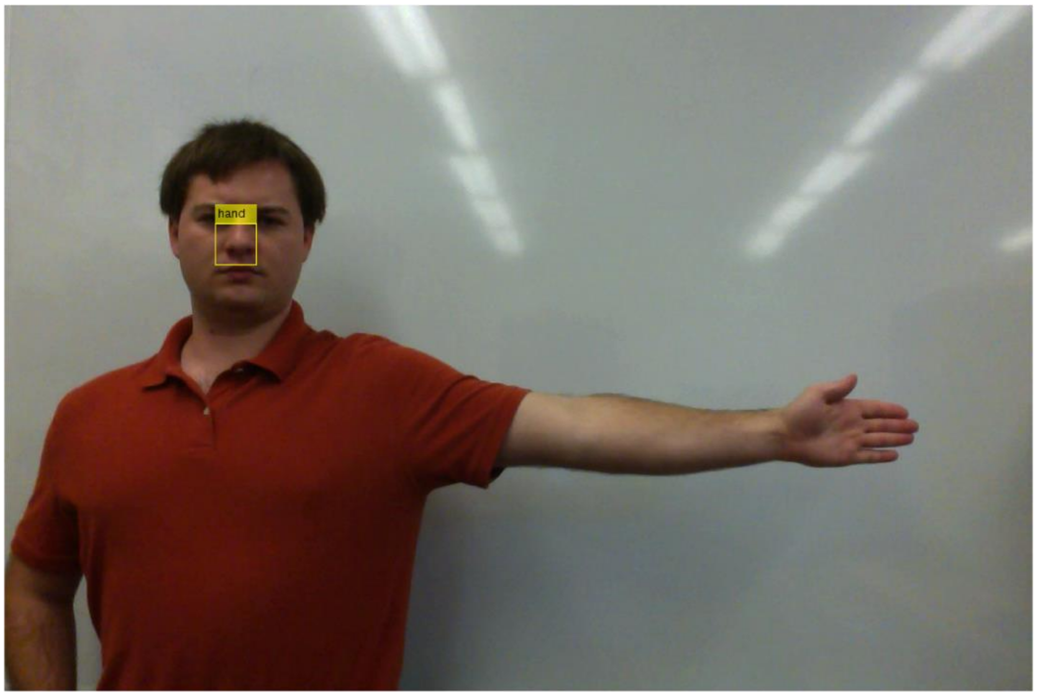

Figure 4: Far Off Incorrect Hand Location

3.4: Smoothing

Despite the link distance analysis method for removing large scale errors described in the previous section, the remaining points do not represent a smooth path of goal points due to small errors in the detection process. This type of high frequency measurement noise is common in experimental data. If this were a traditional robotic system then the desired movement would be known, and these measurements could be fitted to the expected outcome with a trial function [6]; however, since the purpose of this algorithm is to determine the arm location, a different method must be used. Strict interpolation, while providing a continuous link between the points, would not remove the noise component. In this study, the interpolation and smoothing operations have both been accomplished using a spline function. Splines are a method of smoothing and interpolating data [7] that grew from piecewise polynomial fits but have the advantages of global smoothness, easy digital evaluation, and knot management [8]. Figure 5 shows the result of the noise reduction and smoothing where x and

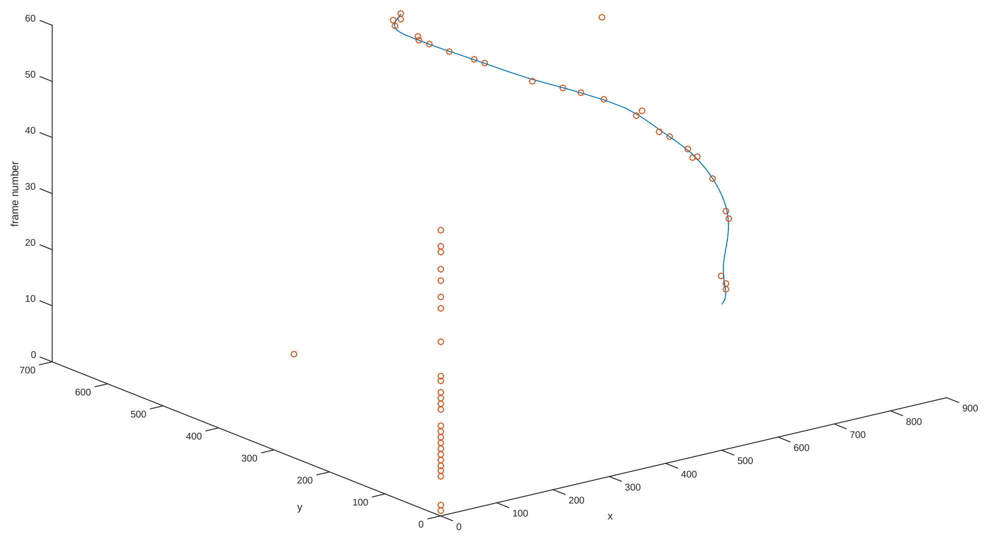

Figure 5: Original Detection Points and Noise Reduced Smoothed Path

y are the image axes and z is time. The red circles represent the original detection locations. The blue line is the smooth path calculated using spline functions for x and y with time as the independent variable.

3.5: Scaling

Before the path can be translated into motor commands it must be scaled from the image dimensions into the set of points the manipulator can reach known as the workspace of the robot [1] [9]. This is nontrivial because there is not a reference object in the image to enable accurate ratio scaling; instead, I scaled the path to a rectangular approximation of the workspace then further scaled down the path iteratively until it was entirely contained within the workspace. This study's workspace is shown Figure 6 as a shaded polygon and

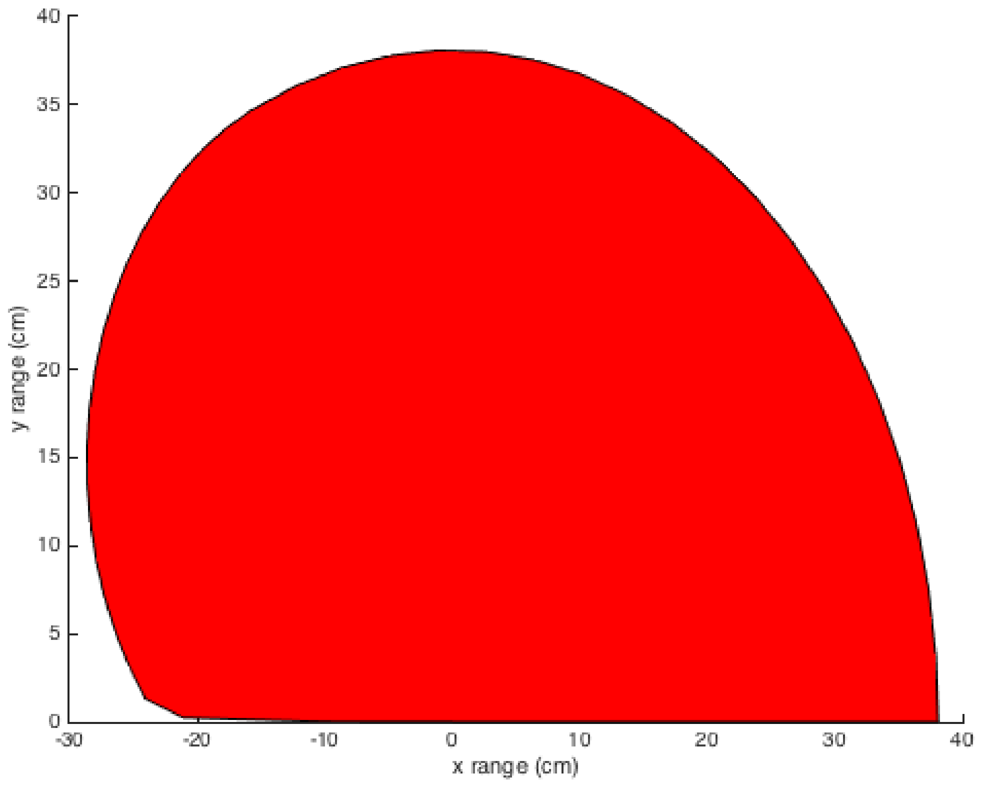

Figure 6: Workspace of the Robotic Arm

the path shown in Figure 5 is displayed after scaling in Figure 7.

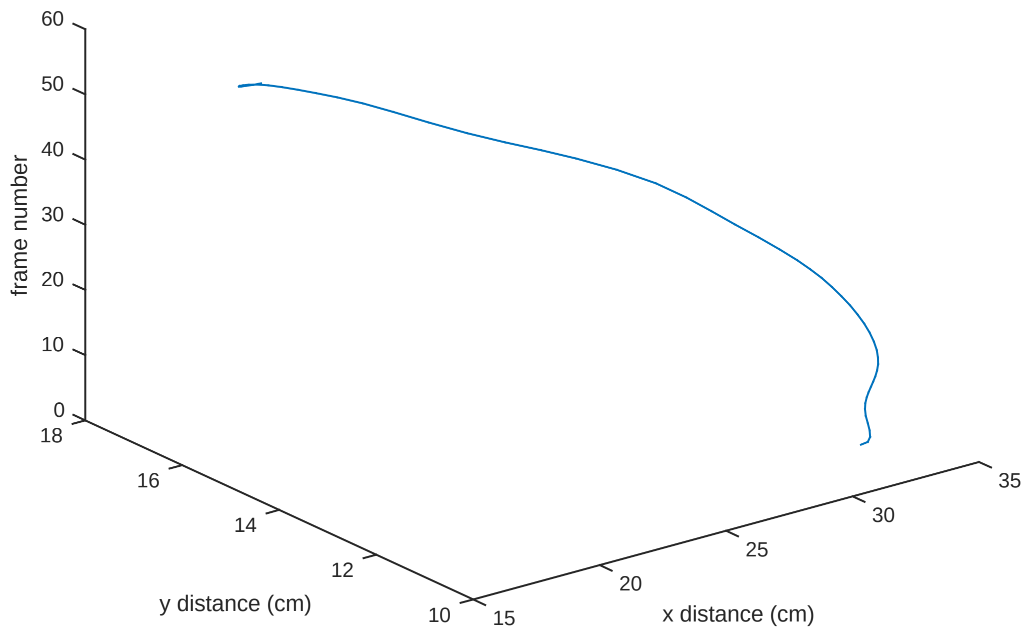

Figure 7: Path After Smoothing and Scaling

3.6: Inverse Kinematics

After the path has been obtained and scaled, the next step in learning a new primitive motion is calculating the commands necessary to cause the robot's end effector to follow the path. Since the data is obtainable from a single viewpoint with zero relative normal velocity between the instructor and camera, it is almost impossible to obtain information concerning movement in that direction; accordingly, this method is designed to work with two-dimensional paths. Furthermore, since the path is two-dimensional, the arm must only be able to move in two dimensions. This type of movement can be accomplished using a two link arm to travel to an arbitrary range of angles and distances from the origin by controlling the angle of the first link to the origin and second link to the first.

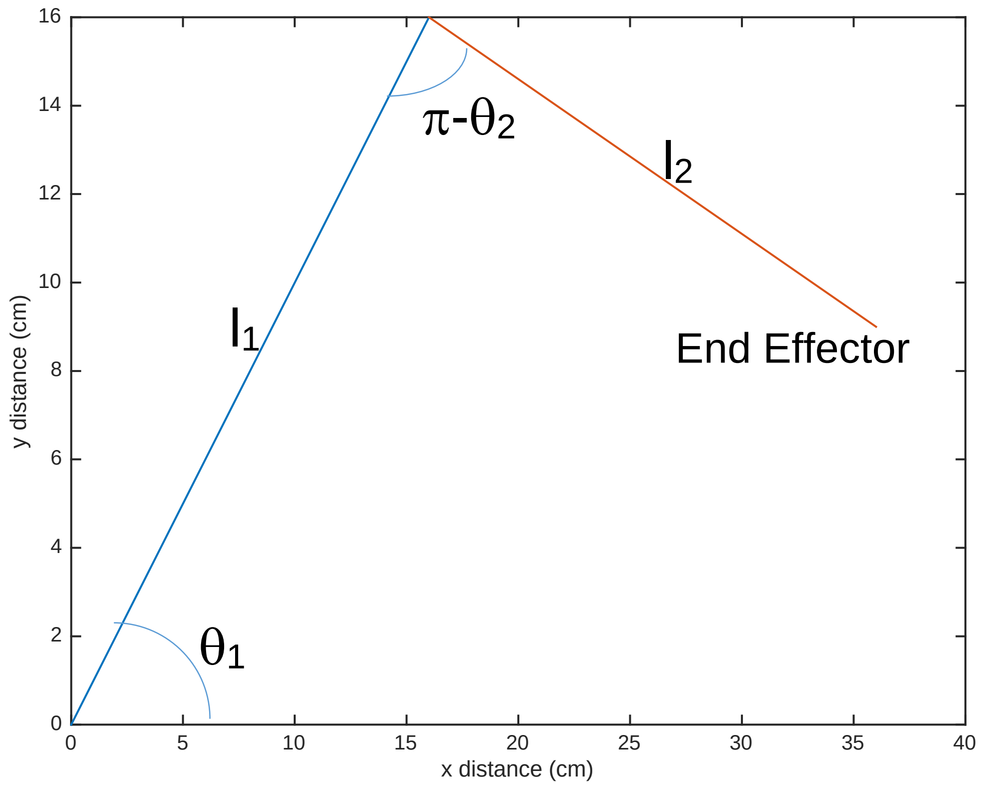

Figure 8: Arm Geometry for Inverse Kinematics

Figure 8 shows the geometry of this inverse kinematics problem where the values for θ_1 and θ_2 can be calculated from Equations 1 and 2 [10].

$$\theta_1 = \tan^{-1}\left(\frac{y}{x}\right) + \cos^{-1}\left(\frac{x^2 + y^2 + l_1^2 - l_2^2}{2l_1\sqrt[2]{x^2 + y^2}}\right), \qquad 0° < \theta_1 < 125° \tag{1}$$

$$\theta_2 = \cos^{-1}\left(\frac{l_1^2 + l_2^2 - x^2 - y^2}{2l_1 l_2}\right) - \pi, \qquad -90° < \theta_2 < 90° \tag{2}$$

3.7: Command Execution

Given the sequence of commands and frame rate, executing the learned movement is a simple, though hardware dependent, step. The algorithm is to start a timer, command each servo in the arm to go to its position from the command sequence, wait until the timer reaches one divided by the framerate, and repeat for every frame in the sequence.

Figure 9 show the arm I used in this study. It was powered by servos shown in Figure 10 which are a combination of control circuit and motor. I connected these to an Arduino microcontroller shown in Figure 11 which sent the control signals to each motor. This microcontroller was in turn connected to the laptop via a USB link. Stages one through six are executed on the laptop which then sends the command sequence via serial to the Arduino for execution.

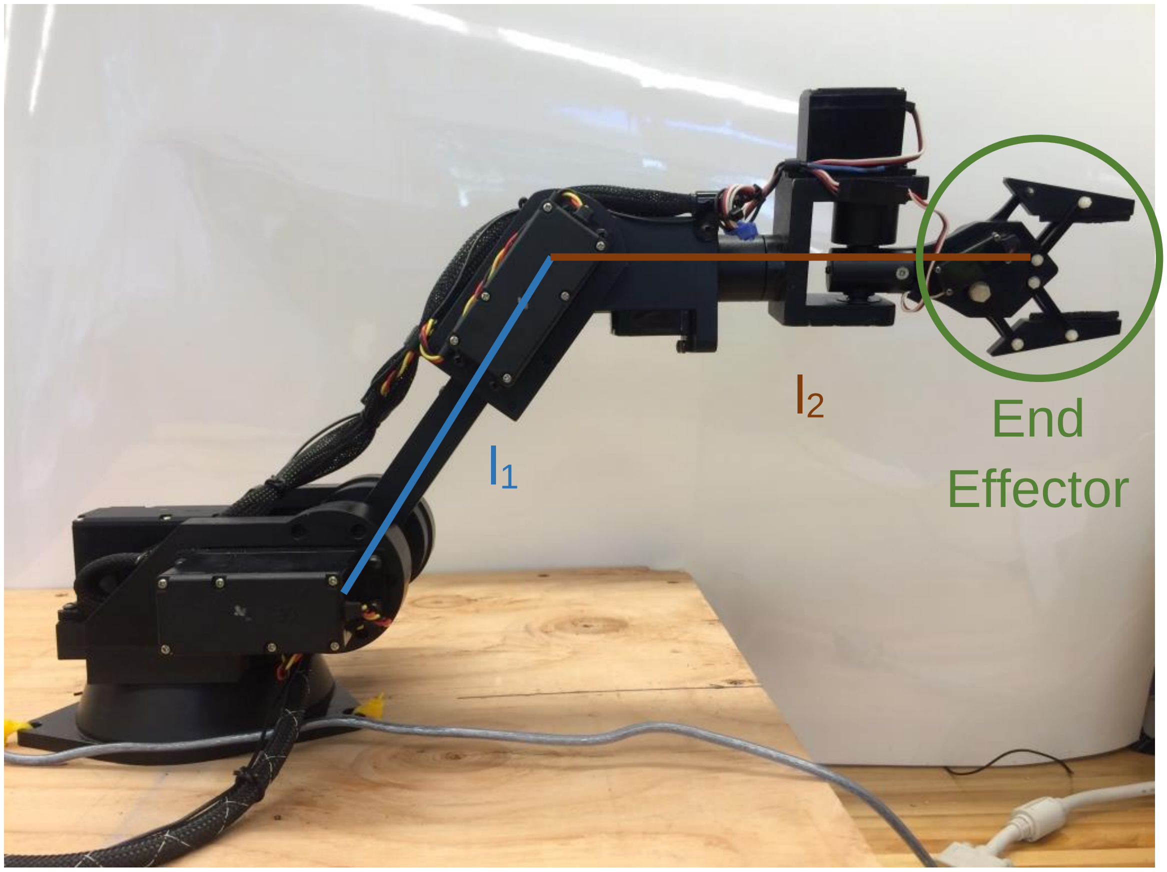

Figure 9: Robotic Arm Used to Test the Algorithm

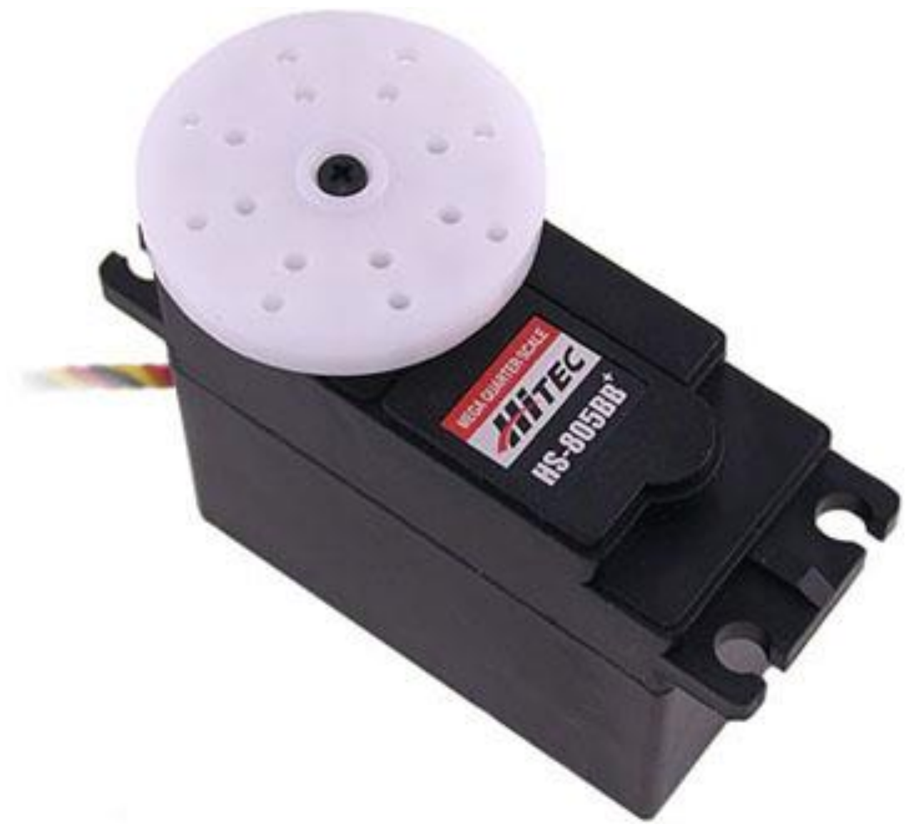

Figure 10: A Servo Used in the Arm [11]

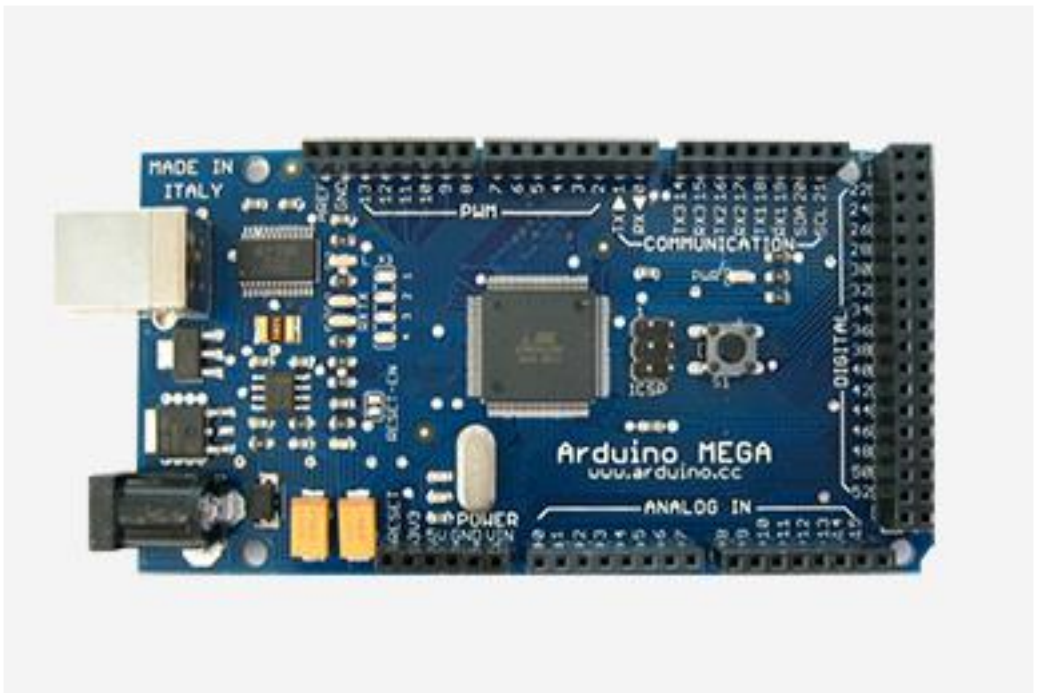

Figure 11: Microcontroller Used to Interface the Laptop and Arm [12]

4: Results

After developing and testing each stage of the methodology independently, I was able to integrate the system. Figure 12 shows an overview of my implementation. The microcontroller is in the far left attached to the side of the wooden base, the arm is in the mid left with wiring running to the microcontroller, and the laptop with webcam is in the right section of the image. Here the arm can be seen preparing to execute the first part of a waving motion.

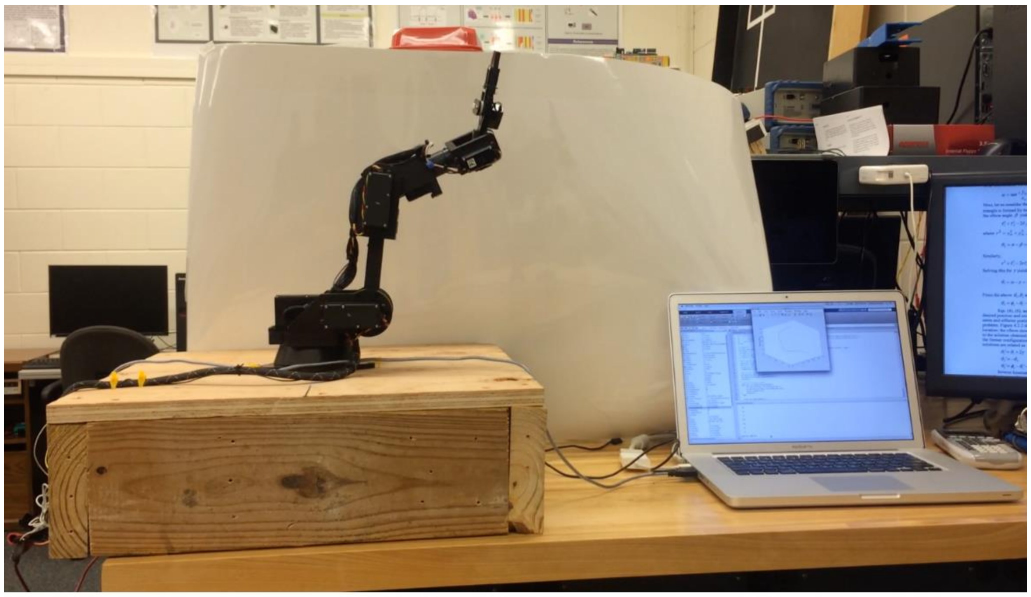

Figure 12: System Implementation

Overall, the process is reasonably accurate and capable of robustly processing most movements that abided by my background and hand position assumptions. The least accurate portion of the project was translating the path into arm control values. Figure 13 shows a calculated path in blue for a waving motion and the projected path based on motor commands in red. While there are some disparities, the actual error between the two paths is less than 10%.

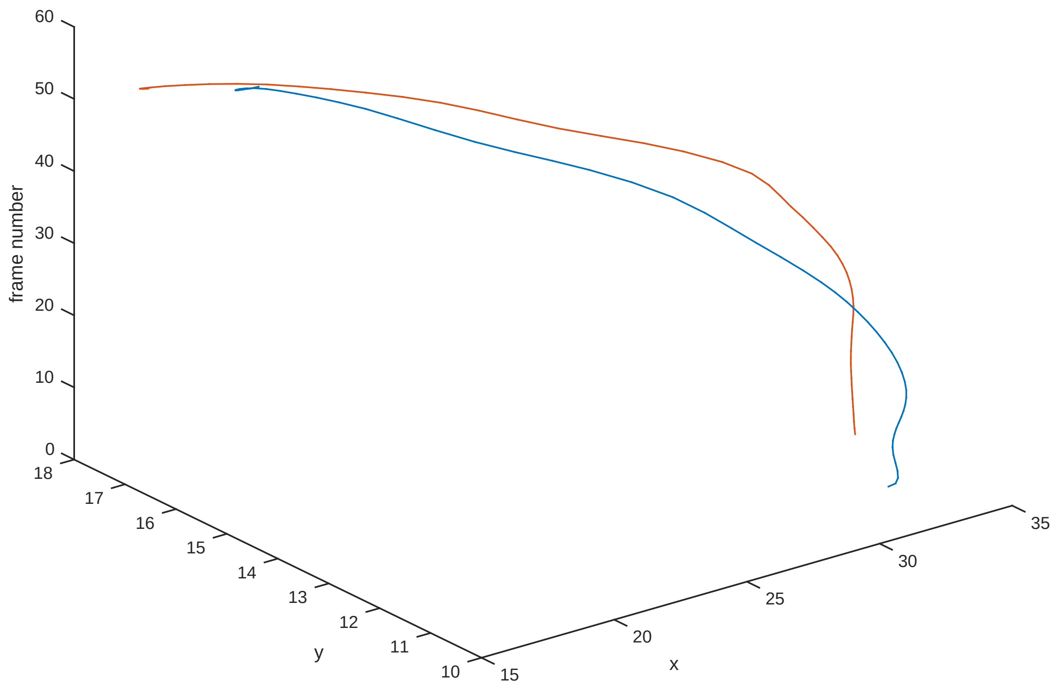

Figure 13: Calculated Path in Blue and Projected Manipulator Path in Red

5: Conclusion

This study developed an imitation based learning approach for adding new movement primitives to arm-type robots using a webcam as the only sensor. Figure 2 demonstrated the system's capability to correctly identify the location of a hand in an image. Figure 5 demonstrated its ability to cope with significant noise in the detection process. Finally, Figure 13 demonstrates the system's ability to translate that path into arm commands.

This project represents a first step toward the learning abilities assistive robots will require to be useful to the general populace. It is differentiated from prior approaches in its methods for hand detection, noise compensation, and movement association without being in the camera's frame of view. In the future, researchers could build complex new behaviors from the primitive movements this system learns from end users.

6: Bibliography

[1] P. Corke, Robotics, Vision and Control: Fundamental Algorithms in MATLAB, Berlin: Springer, 2011.

[2] G. Walter, "An Imitation of Life," *Scientific American,* pp. 42-45, May 1950.

[3] A. Tapus and M. Maja, "Towards Socially Assistive robotics," *International Journal of the Robotics society of Japan,* vol. 24, no. 5, pp. 14-16, July 2006.

[4] A. Chella, H. Dindo and I. Infantino, "A Cognitive Framework for Imitation Learning," *Robotics and Autonomous Systems,* vol. 54, no. 5, pp. 403-408, 2006.

[5] P. Viola and M. J. Jones, "Robust Real-time Object Detection," *International Journal of Computer Vision,* vol. 4, July 2001.

[6] C. H. Reinsch, "Smoothing by Spline Functions," *Numerische Mathematik,* vol. 10, no. 3, pp. 177-183, October 1767.

[7] P. Craven and G. Wahba, "Smoothing Noisy Data with Spline Functions: Estimating the Correct Degree of Smoothing by the Mehtod of Generalized Cross-Validation," *Numerische Mathematik,* vol. 31, no. 4, pp. 377-403, December 1978.

[8] L. Schumaker, Spline Functions: Basic Theory, Cambridge University

Press, 2007.

[9] M. W. Spong, S. Hutchinson and M. Vidyasagar, Robot Modeling and Control, Wiley, 2006.

[10] H. Asada and J. Leonard, "Introduction to Robotics," 2005. [Online]. Available: http://ocw.mit.edu/courses/mechanical-engineering/2-12-introduction-to-robotics-fall-2005/lecture-notes/chapter4.pdf. [Accessed April 2015].

[11] Servo City, "ServoCity," 2015. [Online]. Available: https://www.servocity.com/html/hs-805bb_mega_power.html#.Vw-48GPfPMU. [Accessed April 2016].

[12] SpikenzieLabs, "Arduino Mega," 2016. [Online]. Available: https://www.arduino.cc/en/Main/arduinoBoardMega. [Accessed April 2016].