Classifying Actions from Demonstration

Matthew Frierson, Palakkumar Hirpara, Benjamin Singer, Michael Tirtowidjojo
Freshman Research Initiave
Learning Agents Research Group
Department of Computer Science
University of Texas at Austin

Abstract

We present both a more intuitive controller design for maneuvering a Kinova robotic arm and a method for classifying actions from demonstration. First, we discuss designing the layout of our new Xbox controller and testing it against the built-in Kinova arm joystick. Second, we use a machine learning algorithm to implement a method of using demonstrations from a controller to classify primitive actions. Using this method of classification, we additionally propose a way of partitioning a recording of multiple actions in order to individually classify each of the primitive actions that, in combination, composed the higher-level action.

Introduction

1.1 Joystick Controller

The first problem our project addresses is improving the control system for the Kinova robotic arm used in the University of Texas at Austin (UT) Learning Agents Research Group (LARG) Build-Wide Intelligence (BWI) lab. The current setup for controlling the arm is inconvenient and unintuitive. The built-in joystick sacrifices intuitiveness for design simplicity.

Figure 1: Built-in Kinova arm joystick.

For example, one must switch between multiple "modes" to move the arm in an angular or linear manner and to manipulate the end-effectors. Although people who are familiar with the robot arm should not find the current setup too technically complicated, inexperienced users may find it difficult to control the robot arm. This is especially problematic if these inexperienced users are teaching the robot using a learning from demonstration. As robotics moves out of the lab and into everyday life, and as learning from demonstration becomes a more prevalent technique for machine learning, better control systems will be necessary in order to protect both humans and their robot counterparts.

As a starting point in our goal of creating an intuitive controller, we decided to use a pre-existing Xbox controller. The core benefit of such a controller is that it allows a user to move the arm both in a linear and angular manner simultaneously. This alleviates the need to switch between modes, a major pain point of the built-in joystick.

1.2 Actions

A byproduct of robots becoming more widespread in everyday life will be the need to democratize use of the new technology by allowing users who are not well-versed in robotics to teach robots new tasks. As Argall et. al. describe, enabling a robot to learn from demonstration allows users of all backgrounds and experience levels to teach robots to perform tasks.

With this in mind, we provide a method of determining a classification for a primitive action based on an accumulated dataset of actions demonstrated to the robot. We define a primitive action as a simple action with limited variability (lift, push, grasp, sweep) that can then be combined with other primitive actions to form a higher-level action (pick up an object, go to home position). Classifying actions from a dataset requires the use of machine learning to determine classification. Based on the nature of the data collected, the k-Nearest Neighbors algorithm is especially suited for analyzing the data.

Related Work

Our work in designing an intuitive controller for the robotic arm is related to "An Effective and Intuitive Control Interface for Remote Robot Teleoperation with Complete Haptic Feedback" by Glover et. al. In their research, they use a second robotic arm, physically manipulated by a human subject, the movements of which are mimicked by a different robotic arm. Glover et. al. concluded that having a more intuitive control over an arms movements, in addition to haptic and visual feedback, improved participants performance in completing a set of tasks. Though we did not have the resources to conduct an experiment on nearly as grand a scale, their approach to robotic arm control, intuition in control

over simplicity in design, is something we tried to implement in our controller.

Furthermore, our work builds off research conducted using a Leap Motion Controller, "Intuitive and Adaptive Robotic Arm Manipulation using the Leap Motion Controller," by Bassily et. al. They describe using a Leap Motion Controller to control a Kinova arm, the same arm that we use in our project. The authors developed an algorithm to allow an ideal mapping between a users hand and robot arms movement using the Leap Motion Controller. This manipulation method was specifically created to further a major goal of modern robotic research helping individuals with physical disabilities perform "Activities of Daily Living" in the context of "Ambient Assisted Living." Although, our research does not relate as closely to that sub-field, their conclusions about the intuitiveness of their controller were helpful in shaping our design.

In addition to research related to controllers, our project also builds off of several previous experiments that utilize learning from demonstration techniques. "A Survey of Robot Learning From Demonstration," referenced earlier, by Argall et. al. provides a survey of research in robotic learning by demonstration and explains various methods to teach robots using this technique. The paper discusses two techniques for learning: generalizing after all actions have been completed and updating a policy as training data becomes available. Our project implements the second technique before testing a new action submitted to the dataset, our policy waits for a human to indicate the veracity of its classification and updates its previous classifications based on input. Thus, our learning is also supervised.

Another experiment that uses the second technique in learning from demonstration, updating a policy as more data becomes available, is "Learning Manipulation Actions from a Few Demonstrations" by Abdo et. al. In this experiment, a robot attempts to learn from a human based on few demonstrations. The robot then asks the demonstrator for more examples of a task if it is not confident enough to complete the given task with information is has already gathered. Some of this work involves teaching robots more basic actions, though these are not presented as "primitives." Rather, the experiment focuses on a robot chaining together what it has learned from previous demonstrations, rather than from a subset of smaller actions.

"Learning Task Sequences from Scratch: Application to the Control of Tools and Toys by a Humanoid Robot" by Arsenio applies learning from demonstration to real-world tasks. The paper discusses teaching complex action sequences to a robot that visually observes human teachers completing similar tasks. He describes the learning process in three general steps: first, the robot recognizes an object using the objects color, luminance, and shape cues and generates object models; second, it associates the object with a corresponding action; lastly, the robot learns both the sequence of events that comprise a task as well as the objects being acted on. In the process, the robot uses Markov chains and kinetics-mapping to extract information about a particular task it is trying to learn.

"Learning Similar Tasks From Observation and Practice" by Bentivegna et. al. describes a case study in which researchers taught a robot, using learning from demonstration and learning from practice, to solve a marble maze. This paper focuses on the use of "local representations" of a situation, descriptions which focus on features of the board local to the robot, and builds off of previous research in which the authors used "global representations," descriptions that include the specific location of the ball on the board. They claim that using a global representation helps the robot develop skills specific to a particular situation, while a local representation allows a robot to generalize the data it collects. After completing their experiment, they noted that the robot achieved its goal much more slowly than its human teachers, indicating that practice would help the robot improve its skills. After the robot practiced the game sixty times, the researchers noticed both an increase in the average velocity of the robots movements and in the number of times it successfully completed the maze. To buttress their claims, the authors ran the same experiment using a software maze and demonstrated similar success. Most significantly, this experiment relied on the use of a set of primitive actions that the robot could perform on the marble maze. These actions were highly specialized and specific to the experiment. At each stage of the maze, the robot chose which primitive action to complete based on the local and global representations.

Our work builds most closely off of Bentivegna et. al. because we too are focusing on what we deem to "primitive" actions. However, in some ways, our experiment generalizes their results. While their primitives are specific to completing a marble maze, we teach the arm primitives that can be combined into a larger variety of sequences. Thus, we incorporate aspects of Arsenios work as well.

Methodology

3.1 Controller Design

Our goal in designing the layout of the new controller was to create a setup that was as easy as possible for new users to learn. In creating this intuitive design we solicited advice from fellow Freshman Research Initiative classmates as well as Undergraduate Mentors in the LARG BWI Lab. We wanted our controller to feel as natural as possible and we did not want new users to be surprised by any of the robotic arms movements. A key to accomplishing this was artificially lowering the velocities we sent to the arm. The Xbox controller driver node produced values between [-1, 1] for the two joysticks on the controller which we multiplied by 0.6. Our final controller layout choice is displayed in Figure 2.

In working with the controller, we used several ROS nodes already integrated into the larger BWI codebase. We used the joy node to receive input from the Xbox controller. We relied on the existing the MICO arm drivers Cartesian velocity topic and the existing segbot arm manipulation topics finger position and home arm methods to publish the controllers commands.

In addition to designing a controller that would be used generally in the lab, we wanted to design the controller so

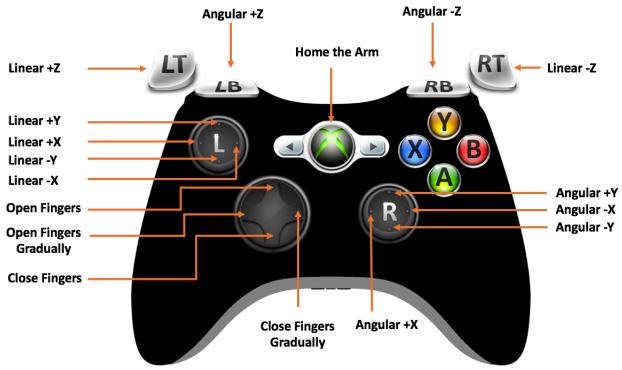

Figure 2: Xbox Controller Layout

that other researchers in the lab could customize it for their experiments. Thus we did not map the four colored buttons on the controller to any actions. An individual working on an experiment that is focused on grasping, for example, can use the buttons as shortcuts for grasp-specific actions without worrying about overwriting other controls.

3.1.1 Collecting Arm Data

The time when the controller is being used is a prime opportunity to collect data for future experiments. We wrote a bash script, capture_all to collect and store the finger position, joint efforts, joint states, tool position, point cloud, and video topics from the arm and the Xtion camera. Thus data can be replayed and used in future experiments.

3.2 Classifying Primitive Actions

This section of our project involved generalizing primitive actions learned from demonstrations in order to classify performed actions. We collected various data points from the arm for each actions performed and classified the actions using temporal bins, the k-Nearest Neighbors machine learning algorithm (k-NN), and dynamic time warping.

3.2.1 Types of Data

The trajectories of our actions were represented by three different types of data: joint positions, joint velocities, Cartesian positions, and Cartesian orientation. Because each data type is measured using disparate units, we needed to combine three different distance functions to calculate a distance between two time-steps. Comparing the the joint positions required calculating the distance between two radians to create a value in the range [$-\pi$, π]. The velocities posed an interesting problem. The values were measured in the rage [-1, 1] with many being closer to 0 than either bound. This meant that subtracting the velocities produced a small value that had little to no impact on the final difference between two actions. This meant that one of the most defining differences between actions was being overpowered by the other factors whose differences resulted in larger values. By weighting the differences between the velocities, we were able to give them enough of an impact in the final difference to influence the classification. We ensured, however, that their impact would not overpower the other factors. The positions lay on a three-dimensional Cartesian system. Therefore, we were able to calculate the distance between positions using Euclidean distance for three points. The Cartesian orientation is recorded in quaternions and therefore required a quaternion distance function to compute the distance between them. By summing the three distances we computed the difference between two time-steps to use in the k-NN algorithm.

3.2.2 Using Temporal Bins

Our initial attempt at classifying actions utilized ten temporal bins similar to the method used by Sinapov et. al. Using this strategy, we recorded the raw data from an action then divided it evenly into 10 temporal bins. We then averaged the values in each bin and used that average as the value from the bin. Given these temporal bins, we then applied the k-NN algorithm to classify the action. While this approach made it easy to calculate a difference between the bins and generate a difference between the two actions, it did not prove to be precise enough for the classifier to successfully differentiate between different actions.

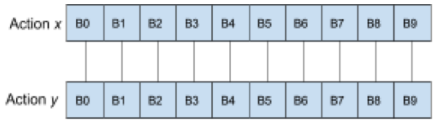

Figure 3: Comparing two actions using temporal bins. Action x is linearly compared to Action y with each bin being compared to the corresponding bin for the other action. Each difference is summed to calculate the difference between the two actions.

We concluded that because we shrank the data down into ten averages, we did not have precise enough data to use to calculate an accurate difference for the robot to make distinctions between actions.

3.2.3 k-Nearest Neighbors

To classify actions, we used the k-Nearest Neighbors machine learning algorithm. k-NN classifies recorded actions based on the majority of the k nearest labels to the recorded action as shown in Figure 4.

Since our data includes more than two different actions, we run the possibility of a tie in the classifier. In order to break the tie, we subtract 1 from k until the tie is broken as shown in Figure 5. The k-NN classifier is trained using supervised learning from a user and will add to the training set only when the user confirms the correct classification. The classifier can also run unsupervised but will not add to the training set.

3.2.4 Dynamic Time Warping

Without the temporal bins there is no set number of time-steps for each action. Because of this, the number of time-steps in each action can vary with some being as low as 90 with others going upward toward 300 time-steps. This meant we could no longer use a linear mapping between the

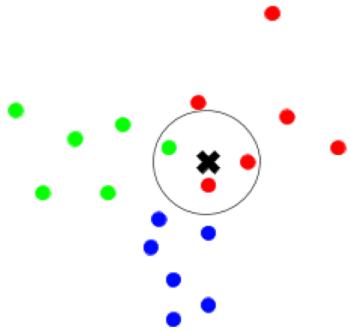

Figure 4: k-NN classifier classifying the unknown label marked 'x'. The classifier is using k = 3 to find three closest neighbors. Since red has the majority of the three, the classifier would determine x to belong to the red classification.

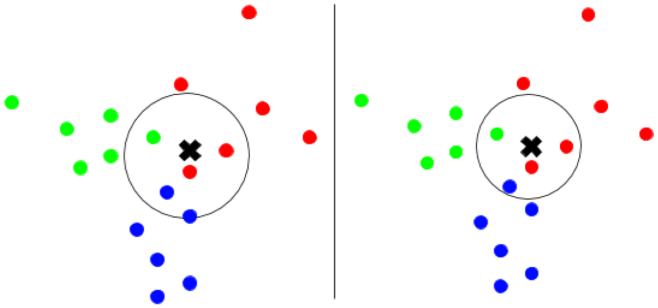

Figure 5: k-NN breaking a tie. On the left with k = 5 there is a tie between the red and blue with 2 reds, 2 blues, and 1 green. The right show the tie breaker with k = 4 which breaks the tie between the 2 reds and 2 blues with 2 reds, 1 blue, and 1 green.

time-steps. In order to compare the trajectories of the two actions, we implemented dynamic time warping as described by Koang et. al. To perform dynamic time warping, we generate an $n \times m$ matrix where n is the number of time-steps in action x and m is the number of time-steps in action y. We populate this matrix with the difference between each time step in x compared against each time step in y such that index i in row 0 would be

costs[0][i] = distance(x[i], y[0])

From this, we calculate the least costly path from (0, 0) to $(n-1, m-1)$ following the conditions that the path may not move left or up in the matrix. A more detailed description of the conditions that must be met by dynamic time warping is described by Keough et. al.

By doing this, we "warp" the mapping between the time-steps from action x and action y to find the optimal comparison between the two trajectories. This allows us to compare actions with different lengths making a comparison of the raw values possible.

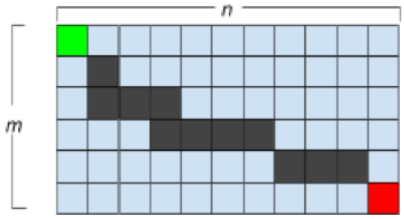

Figure 6: Getting the least costly path through the matrix of distances. The path starts at the green square (0, 0) and must end at the red square (n - 1, m- 1) and moves down and to the right.

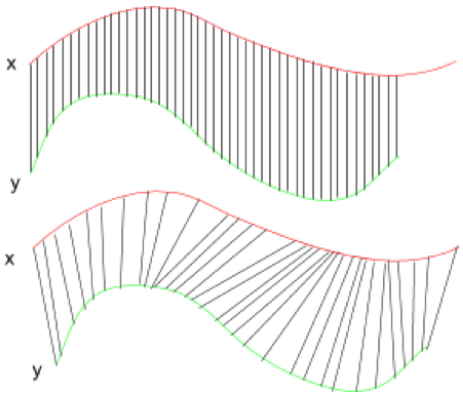

Figure 7: The difference between using a direct mapping of time-steps and using DTW. Top, direct mapping restricts comparing each time step to the matching one on the other action and does not work as well with different length trajectories. Bottom, DTW allows for the time-steps to be mapped to the best matching time step to provide a more accurate distance measure.

Results

4.1 Controller Experiments

For our comparison experiment between our Xbox controller and the built-in joystick, we recruited seven students who had never used either controller. We gave the students a minute and a half to learn the movements of each controller and then had the students complete three tasks with each joystick in a random order.

4.1.1 Tasks Performed in Experiment

Task 1: Place an animal object into a bowl
Task 2: Stack three objects on top of each other
Task 3: Pour pinto beans into a bowl from a plastic container

4.1.2 Results of the Experiment

Data from our experiment showed that majority of the students preferred the new Xbox controller.

The average improvement in time for Task 1, Task 2, and Task 3 were 133 seconds, 293 seconds, and 179 seconds respectively. Moreover, the median improvement for Task 1,

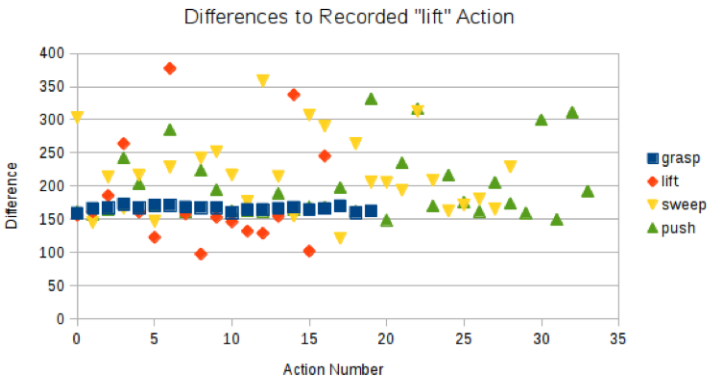

Figure 8: Differences between a recorded lift and the other actions in the dataset. Shows how close each other action is to the recorded one and which actions were the closest. In the case of k = 3 the three nearest neighbors would be two lifts and one sweep making the classification for the action a lift.

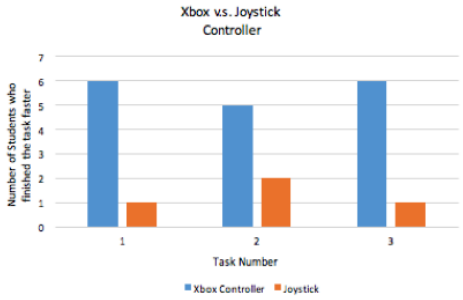

Figure 9: The above graph describes the result of the experiment. It shows the number of students who performed each task faster using the XBox and Joystick controller.

Task 2, and Task 3 was 34 seconds, 138 seconds, and 39 seconds. The recorded timings indicate that, on average, 80% of the time tasks were completed faster with the Xbox controller. Although this percentage may suggest that the Xbox controller is clearly better, it is misleading because we only recorded data from seven people. However, the current data does suggest that the experiment should be replicated with more participants to get more-accurate data.

4.2 Classifying Results

Our initial attempt at classifying actions utilized the temporal bin strategy in order to create a uniform number of time-steps to compare between different actions. When this did not produce desired results, we converted to using the raw data and the dynamic time warping approach in order to classify actions. To test, we used a training set of 100 actions that contained 17 lifts, 34 pushes, 29 sweeps, and 20 grasps and tested against a test set of 39 actions.

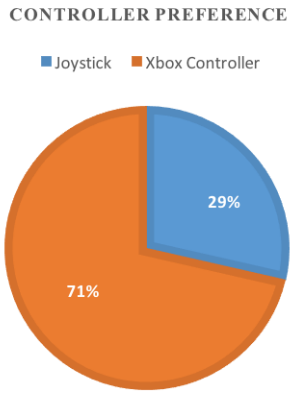

Figure 10: Out of the seven people who participated in the experiment five of them said they preferred Xbox controller and two said they preferred Joystick after completing all the tasks with both controllers.

4.2.1 Temporal Bins Results

While the temporal bins made it simpler to compare different actions by allowing a direct comparison of time-steps, the details of each data point that created the distinction between the different classifications were lost when we took the average of a group of bins. This proved not to be much of a problem with a working set limited to the actions “lift” and “sweep.” However, upon adding the action “push” to the set, the classifier began to confuse the “push” and “sweep” actions resulting in the wrong classification roughly 50% of the time. Initially, we added more examples to the training set in an attempt to reduce the errors by increasing the number of examples to compare against. However, this did not improve our results by any noticeable factor.

4.2.2 Dynamic Time Warping Results

By switching to using dynamic time warping instead of using the temporal bins strategy, we were able to more accurately classify actions. The first dataset achieved an 82% accuracy, incorrectly classifying 9 actions, while the second dataset achieved a 74% accuracy, incorrectly classifying 13 actions.

Conclusion and Future Work

5.1 Controller

Though our controller experiment data indicated that our new controller is superior to the old joystick, we recognize that this conclusion is based on little data. One could build off our work by running a more extensive experiment with more participants and tasks.

In addition, our experiment focused on people who had never used an arm robot controller. The results of the experiment may not be applicable or relevant for people who use the built-in joystick extensively. However, even if the built-in joystick is easier to learn, the wireless controller could fix

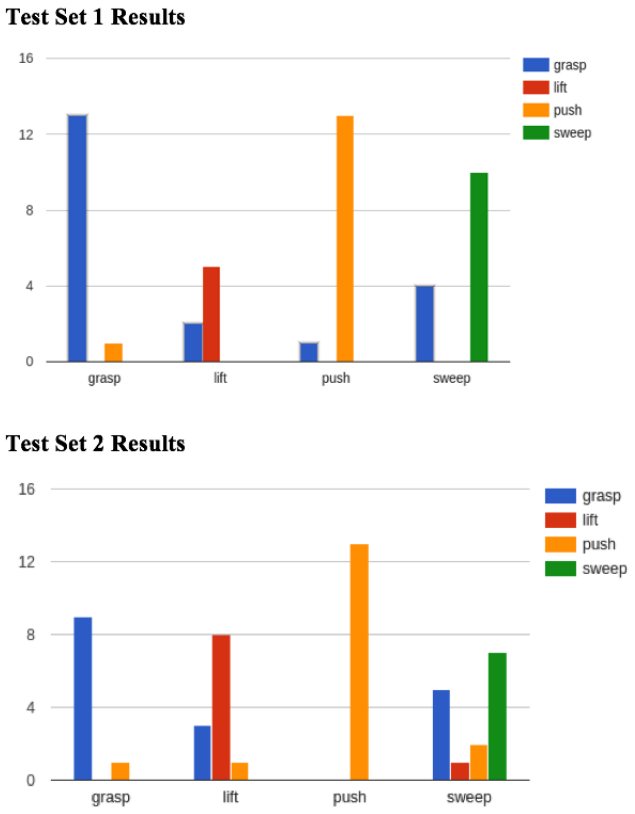

Figure 11: Results from test sets for classifying actions. The classifier performed 82% accurate and 74% on two independent tests with 50 actions each. The results still show some confusion between actions with the several actions being mislabeled as grasps.

a major pain point for lab researchers by allowing a user to maneuver the arm without switching modes.

Furthermore, our controller layout was based on our intuition and input from mentors. We may have showed that our controller is better than the old joystick, but that conclusion is relevant to the controller as a whole. Future experiments could run extensive A/B tests on all aspects of the controller to determine if our layout is indeed the most intuitive. We concede that results could be improved if the controller layout was slightly different we made a guess based on our intuition. Immediate next steps we would like to take are adding a second mode that would allow a user to tele-operate the Segbot base. There is a danger in allowing such a quick transition between controlling the arm and base, so we would need to add various hurdles for a user to jump through to ensure that they wanted to switch to "base mode."

Finally, once we complete the aforementioned steps, we plan to add our work to the lab code-base and put our controller into regular use.

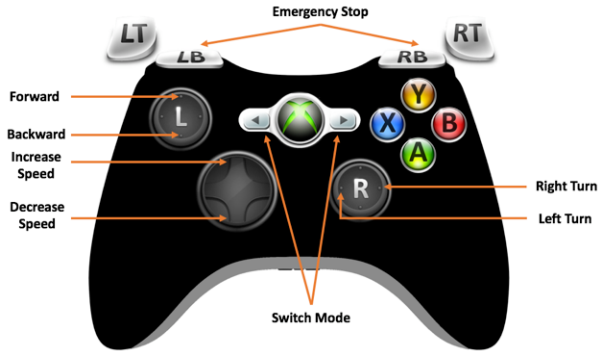

Figure 12: Controller Layout for Tele-operation of the Segway Base

5.2 Classifying from Demonstration

By using temporal bins, we did not have much of success in having the robot correctly identify the demonstrated primitive action in the beginning. It was not until we implemented Dynamic Time Warping that we started to see a significant improvement in the robots classification of the demonstrated actions. Our desired accuracy for the classifier was 80%. Given the performed tests, our classifier performed near the desired accuracy, but did not exceed it one test classified 82% of performed actions and another classified 74% correct. We would need to run more tests to determine the true accuracy of the classifier. Due to these results, we conclude that the classifier performs near the desired accuracy but does not completely meet the desired accuracy.

Possible future work on this project could include chaining together the primitive actions to form complex actions such as opening a door or putting an item back into a shelf. This makes it possible for people with little experience in robotics to be able to "program" them to help out people.

Another possible extension is to have the robot be able to break up a complex action into simple primitive actions. For example, given the task of putting an object on a shelf, the robot can break it up into simple actions such as: send the arm near the object, grasp the object, lift the object up, sweep the arm towards the shelf, push the object into the shelf, and then release the object.

A final area of future research ties together the two sections of our project more closely by creating a controller that can predict how the user wants to manipulate the arm based on dynamic action classifications. This can help create smoother transitions between arm states.

References

Abdo, N.; Kretzschmar, H.; Spinello, L.; and Stachniss, Cyrill. 2013. Learning and Generalization of Motor Skills by Learning from Demonstration. *IEEE International Conference on Robotics and Automation* 1268-1275.

Argall, D. B.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57: 469–483.

Arsenio, M. A. 2004. Learning task sequences from scratch: applications to the control of tools and toys by a humanoid robot. *IEEE International Conference on Control Applications* 1-6.

Bassily, D.; Georgoulas, C.; Guettler, J.; Linner, T.; and Bock, T. 2014. Intuitive and Adaptive Robotic Arm Manipulation using the Leap Motion Controller. *International Symposium on Robotics*

Bentivegna, C. D.; Atkeson, G. C.; and Cheng, G. 2004. Learning From Observation and Practice Using Primitives.

Glover, C.; Russell, B.; White, A.; Miller, M.; and Stoytchev, A.; 2009. An Effective and Intuitive Control Interface for Remote Robot Teleoperation with Complete Haptic Feedback. *ETC*

Hiroaki, S.; and Chiba, S. 1978. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *Robotics and Autonomous Systems* 26: 43–49.

Keogh, E.; and Ratanamahatana, C. 2005. Exact indexing of dynamic time warping. *Knowledge and Information Systems* 7: 358–386.

Kuang, Q.; and Zhao, L.; 2009. A Practical GPU Based KNN Algorithm. *International Computer Science and Computational Technology* 151-155

Pastor, P.; Hoffmann, H.; Asfour, T.; and Schaal, S. 2009. Learning and Generalization of Motor Skills by Learning from Demonstration. *IEEE International Conference on Robotics and Automation*

Sinapov, J.; Bergquist, T.; Schenck, C.; Ohiri, U.; Griffith, S.; and Stoytchev, A. 2011. Interactive Object Recognition Using Proprioceptive and Auditory Feedback. *IJRR*.

Code Cited

bwi: https://github.com/utexas-bwi/bwi
joy: https://github.com/ros-drivers/joystick_drivers

Github Repositories

controller: https://github.com/bzsinger/joystick_bwi
lfd_actions: https://github.com/MTirtowidjojo/lfd_actions