Replication of Human Two-Fingered Grasping Motions in Robotic Manipulation of Household Objects

Andrew Vo, Stephen Moseson, and Matthew Cong

***Abstract*— In our project, we consider the problem of replicating human two-fingered grasping motions on the AX12 Smart Robotic Arm. Two-fingered grasping motions are among the most common motions used in everyday life. Examples include picking up a plate from a dishwasher, moving a chess piece, and as we will demonstrate, opening a drawer. In fact, many more complicated motions, such as holding a pen/pencil and writing involve a two-fingered grasping component. To accomplish this goal, we designed a sensor mount positioned behind the robotic arm that accommodates a variety of sensors including a 720p high-definition webcam. Then, using SIFT features, we locate (in the webcam image) the feature associated with the drawer that we want to open and close defined by a point in the image. Using previous calibration and bilinear interpolation, we convert this coordinate into global Cartesian space. This coordinate is subsequently fed into the analytical inverse kinematics service which replies with the joint coordinates. Subsequently, these are given to the controller which moves the arm to a desired location. Then, using a series of movements, we can perform our task. In particular, this project takes advantage of ROS's service/client and publisher/subscriber model. These paradigms form the basis for communication between different components of the project.**

I. INTRODUCTION

Two-fingered grasping motions involving the thumb and the index finger are some of the most common motions used in everyday tasks such as picking up a plate from a dishwasher, moving a chess piece, and opening a drawer. However, the two-fingered grasp usually plays a fundamental part in more complicated motions such as writing, using chopsticks, and even pitching a baseball. It is a easy to see that many of these motions apply to everyday household tasks which are often repetitive and could, technical challenges aside, be successfully performed by a robot. However, the technical problems that present themselves in the design of such a robot are extremely varied and difficult.

Consider what occurs when you move a rook during a game of chess when the only other pieces left on the board are the opposing kings. Using information from two eyes, the brain processes the images that are focused on each retina and obtains the location of the rook. The brain subsequently, through experience gained during childhood and throughout life, instructs your arm to move to that position. However, this position is reached by changing the angles within each of our numerous joints. The shoulder, elbow, wrist and the other more minor joints must be at the correct location to ensure a successful grasp of the object.

In a robot, this first involves obtaining a Cartesian coordinate in global space. This coordinate is in the analogous robot problem, in an environment without obstacles where it has some knowledge about the coordinates of its surroundings. This is much more difficult because with a single camera, as in our setup, we can only obtain a projection of the three-dimensional world. Thus, we must extrapolate based on known points in order to obtain a three-dimensional global coordinate from the two-dimensional image. Obtaining the 2D coordinate on which the object is located is an additional problem. We must find the features that are unique to the object in image and then obtain a representative point, usually the center. This is an extremely difficult problem but fortunately, algorithms such as SIFT have been developed that provide solutions to the problem that can be adapted for our needs. Then, we must determine the joint angles for each joint in the robot that will move the end-effector to the desired position. This requires solving the inverse kinematics problem which unlike the forward kinematics problem, is non-linear and usually has multiple solutions depending on the robot and the different constrains. Subsequently, we must use

Authors are enrolled at Cornell University in the CS 4758 Robot Learning class taught by Professor Ashutosh Saxena in the Department of Computer Science, Cornell University, Ithaca, NY 14853, USA asaxena@cs.cornell.edu

Andrew Vo is an Masters of Engineering student in Electrical and Computer Engineering ahv25@cornell.edu

Steven Moseson is a senior in Mechanical Engineering with a minor in Computer Science smoseson@gmail.com

Matthew Cong is a junior in Computer Science and Engineering Physics mdc238@cornell.edu

Bilinear Interpolation and Camera Clicker code written by Daniel Cabrini-Hauagge, PhD. student in the Department of Computer Science at Cornell University hauagge@cs.cornell.edu

Robot and software platform set-up and maintained by Jigar J. Shah, senior in the school of Electrical and Computer Engineering. jjs367@cornell.edu

a controller to send the correct instruction bits to each individual servo in order to move to the desired point.

Thus, we can see that a simple everyday task for the human is extremely challenging for a robot to perform, even in a structured environment with no obstacles. This paper outlines the related work and describes our approach to the problem. This framework is tested in the experiments described in Section IV. Lastly, we summarize the results of the experiment and possible extensions.

II. RELATED WORK

Grasping objects is a widely studied area in robotics. However the specific task of opening drawers has not been studied extensively. The paper "Learning to Grasp Novel Objects using Vision" [1] provides a good approach to tackle the challenge of determining where to grasp unknown objects using only a webcam. In our project we also rely only on a webcam, however instead of using machine learning to classify grasping points we rely on the pre-known structure of the drawers and use the SIFT algorithm to determine their location.

The control of our arm is also similar to the control of the 5-DOF Katana arm used on the STAIR platform. They describe two classes of grasps that the arm can take: "downward and outward". The downward grasp is taken when objects are close, and the outward is for objects that are further away from the arm. We have adopted this control with our arm and choose to always take an outward grasp because the drawers must be located away from the arm in order to open them once they are grasped.

III. APPROACH

A. Hardware

The AX-12 Smart Robotic Arm (Figure 1) with a dual gripper has six degrees of freedom consisting of a base swivel, shoulder pivot, elbow pivot, wrist swivel, and two independently controlled gripping fingers. It also has an angled fixed joint connected to the base swivel. The AX-12 servos have been a major issue in this project because of torque limitations. If the arm is in a horizontal fully extended position the shoulder joint is unable to lift the arm up. This is a major problem when working with drawers because we need the arm in a fully extended position in order to reach a drawer.

The only sensor we use is a 720p Microsoft webcam positioned above the arm looking down at a 45 degree angle, as shown in Figure 2. This is attached to a custom mount designed to hold it rigidly in place with a clear view of the workspace.

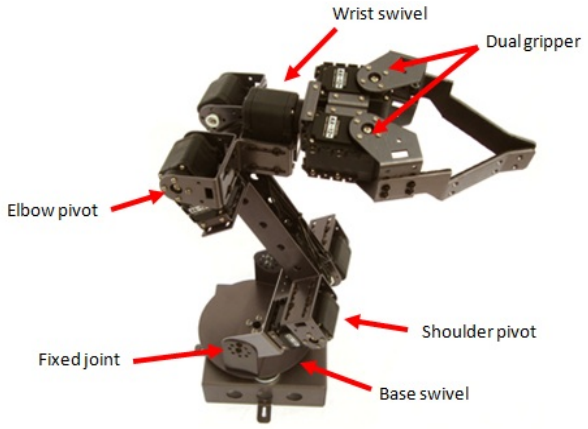

Fig. 1. AX 12+ Smart Robotic Arm with Dual Gripper

B. Controller

The low-level control is based upon a ROS package developed by the University of Arizona Robotics Research Group. It provides low-level commands for controlling the joint servos on the AX-12 Smart Robotic Arm. Additional work was done in modifying the controller architecture to accommodate smoother movements of the robotic arm.

Given a set of waypoints generated by the path planner, the controller is responsible for smoothly moving the arm between each waypoint. In order to generate smooth movements, the joints will move simultaneously rather than sequentially. The two controller architectures investigated are the joint-space controller and the resolved motion-rate controller.

The joint-space controller uses feedback to control each joint independently. As shown in the following system diagram, the joint-space controller incorporates multiple independent controllers for each joint. In turn, each of these controllers can be implemented as a PD controller to drive the joint angle error to zero.

The resolved motion-rate controller relies on the inverse of the kinematics model to drive the joint state to the desired position. The Jacobian of the arm can be derived from the forward kinematics equation. The following equation expresses how variations in the joint state maps to variations in the end-effector state.

$$\delta x = J(\theta)\delta\theta \tag{1}$$

Since the dimension of the joint space is greater than the dimension of the end-effector state, one can solve for $\delta\theta$ by applying the pseudo-inverse.

$$\delta\theta = J^{\dagger}\delta x \tag{2}$$

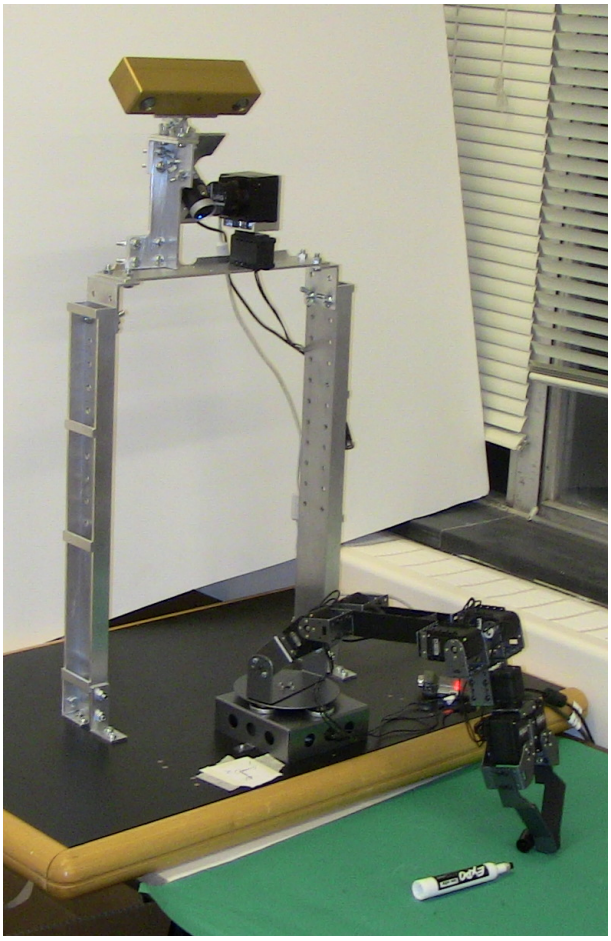

Fig. 2. Webcam is located above and behind robot arm looking down at a 45 degree angle.

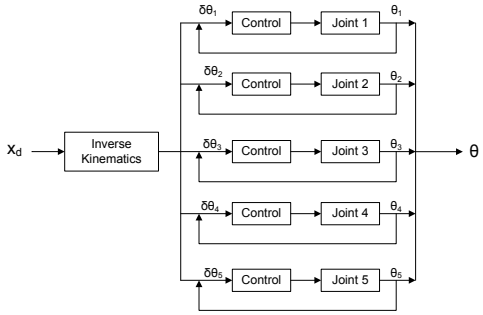

Fig. 3. Joint-space controller system diagram

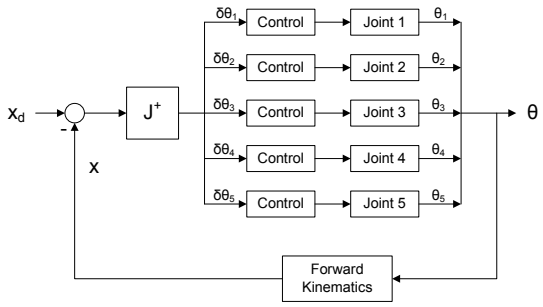

Fig. 4. Resolved motion-rate controller system diagram

The resulting solution using the pseudo-inverse minimizes the least-squares error of the joint-state. A simple control law can be derived based upon this result.

$$\theta^{(new)} = \theta^{(old)} + J^{\dagger}\delta x \tag{3}$$

where $\delta x = x_d - x$, the error between the desired end-effector position and the measured end-effector position.

C. Inverse Kinematics

In the AX12 Smart Robotic Arm, we have a cylindrically rotating base with a two degree of freedom manipulator. This arrangement actually allows the inverse kinematics problem to be solved analytically. Consider a problem similar to obtaining joint coordinates from the global Cartesian coordinates: obtaining joint coordinates from the cylindrical coordinates in the global frame. Looking at the problem from this angle, we can see that ρ, the radial length, is only determined by the 2R section of the arm through a project based on the fixed 45° angle of the fixed joint. A similar projection allows us to determine the z coordinate contribution from the 2R section of the arm. In addition, the ratio of the x coordinate to the y coordinate is determined by ϕ, the azimuthal angle. This provides us with one of the joint coordinates: the base angle $\theta_1 = \phi$. The process for obtaining the other two joint angles is much more complicated.

Thus, our strategy is as follows. Using the given x and y coordinate, we can easily determine $\rho = \sqrt{x^2 + y^2}$. Then, we can obtain $\phi = \arctan\frac{y}{x}$. Then we can subtract the contribution to the z coordinate from the base section of the arm and the section of the arm from the fixed joint to the shoulder joint. Thus, we are left with the z_{adj} coordinate which corresponds to the contribution to the z coordinate from the 2R section of the arm. Then, we can obtain ρ_{adj} by subtracting the contribution of the diagonal section of the base. This effectively decouples our problem. All that is left to determine is θ_2, the angle of the shoulder joint, and θ_3, the angle of the elbow joint.

To simplify the problem, we rotate the coordinate system by 45° counter-clockwise. This allows the equilibrium position of the arm to be on the horizontal axis

on the ρ' and z' two-dimensional space.

Applying the law of cosines on the 2R section of the arm (similar to the approach in [2] with different angle conventions), we obtain

$$\cos\theta_3 = \frac{\rho'^2 z'^2 - l_1^2 - l_2^2}{2l_1 l_2} \tag{4}$$

which gives us a closed form expression for θ_3 where l_1 and l_2 represent the length of the shoulder and elbow links respectively.. However, because of the inaccuracy of the $\arccos$ operation, we can employ the half-angle formula: $\tan^2\frac{\theta}{2} = \frac{1-\cos\theta}{1+\cos\theta}$. Thus, we obtain that

$$\theta_3 = \pm 2\arctan\sqrt{\frac{-\rho'^2 - z'^2 + (l_1+l_2)^2}{\rho'^2 + z'^2 - (l_1-l_2)^2}} \tag{5}$$

where the plus-minus sign represents the "elbow-up" and "elbow-down" solutions. Then, by subtracting the contribution of the elbow section of the arm, we can obtain

$$\theta_2 = \arctan\frac{z'(l_1 + l_2\cos\theta_3) - \rho' l_2 \sin\theta_3}{\rho'(l_1 + l_2\cos\theta_3) + z' l_2 \sin\theta_3} \tag{6}$$

which based on a flag, we can choose either the "elbow-up" and "elbow-down" solution to suit the task we are performing.

One of the largest advantages to this approach is the speed. We do not need an iterative solution which takes significantly more time to run. This allows us to recalculate joint angles frequently when needed. However, this does not take into account the position of the gripper in the end-effector. We will treat this as a small-perturbation of the current inverse kinematics problem in the experiments described in Section IV due to the limited surface area of the gripper arm.

D. SIFT features for Drawer Identification

The SIFT algorithm is used to locate the pixel location of a given drawer in the webcam image. Each drawer has a unique image taped on the front which can be located by matching SIFT features between the target image and the camera image. The SIFT algorithm was chosen because it works under different lighting conditions and camera angles.

The algorithm was implemented in ROS by using the SIFT binaries created by David Lowe. These were run by a C++ wrapper that first created the sift features for both the image key and the camera output and then checked for matches between the images. If even one match was found between the images the program would use the location of the match in each image along with the dimensions of the key to determine the center of the key in the image, shown in Figure 5. It would then

Fig. 5. Image keys were compared with the camera image to find matching features using the SIFT algorithm.

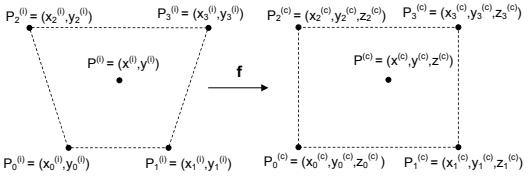

Fig. 6. Mapping from Image Space to Cartesian Space

publishes the pixel x and y coordinates so other ROS nodes could use the drawer coordinates.

The SIFT image keys used on the drawers were scenes in nature that were free of patterns or regular shapes. These types of images work well with sift because they generate a lot of unique features that can be matched to the camera feed. The tolerance for matching was set fairly low so that no false matches were identified, however this also meant that it was more difficult to ensure a match in difficult lighting or an obscure angle.

E. Camera Calibration using Bilinear Interpolation

The camera needs to be calibrated in order to convert the image pixel coordinates to Cartesian coordinates in 3D space. Since all of the tasks we consider are performed on a plane, bilinear interpolation provides a good solution. Consider the mapping of a point in the image space to a point in the Cartesian space shown in the following figure.

We assume the image points $P_j^{(i)}$ and the corresponding Cartesian points $P_j^{(c)}$ both form a quadrilateral. Using bilinear interpolation, any point within the boundary can be expressed as a linear combination of

these calibration points. We can express the coordinates of the image point $P^{(i)}$ as a function of two constants $0 \leq s, t \leq 1$.

$$x^{(i)} = (1-s)((1-t)x_0^{(i)} + tx_2^{(i)}) + s((1-t)x_1^{(i)} + tx_3^{(i)}) \quad (7)$$

$$y^{(i)} = (1-s)((1-t)y_0^{(i)} + ty_2^{(i)}) + s((1-t)y_1^{(i)} + ty_3^{(i)}) \quad (8)$$

Solving for t yields

$$t = \frac{(1-s)(x_0^{(i)} - x^{(i)}) + s(x_1^{(i)} - x^{(i)})}{(1-s)(x_0^{(i)} - x_2^{(i)}) + s(x_1^{(i)} - x_3^{(i)})} \quad (9)$$

$$t = \frac{(1-s)(y_0^{(i)} - y^{(i)}) + s(y_1^{(i)} - y^{(i)})}{(1-s)(y_0^{(i)} - y_2^{(i)}) + s(y_1^{(i)} - y_3^{(i)})} \quad (10)$$

Eliminating t from these two equations yields a quadratic equation in s. The 2nd-order polynomial is written in Bernstein form for numerical stability when solving the system equations.

$$A(1-s)^2 + 2B(1-s)s + Cs^2 = 0 \quad (11)$$

where the constants A, B, C are given by

$$A = (P_0^{(i)} - P^{(i)}) \times (P_0^{(i)} - P_2^{(i)}) \quad (12)$$

$$B = \frac{1}{2}((P_0^{(i)} - P^{(i)}) \times (P_1^{(i)} - P_3^{(i)}) + (P_1^{(i)} - P^{(i)}) \times (P_0^{(i)} - P_2^{(i)})) \quad (13)$$

$$C = (P_1^{(i)} - P^{(i)}) \times (P_1^{(i)} - P_3^{(i)}) \quad (14)$$

where $\times$ denotes the 3D cross product (e.g. $P_0 \times P_1 = x_0 y_1 - y_0 x_1$). The solution to the quadratic equation is given by

$$s = \frac{(A-B) \pm \sqrt{B^2 - AC}}{A - 2B + C} \quad (15)$$

When $A - 2B + C = 0$, then the quadratic equation is linear and the solution is given by

$$s = \frac{A}{A - C} \quad (16)$$

With s and t known, the coordinates of the desired Cartesian point $P^{(c)}$ are given by

$$x^{(c)} = (1-s)((1-t)x_0^{(c)} + tx_2^{(c)}) + s((1-t)x_1^{(c)} + tx_3^{(c)}) \quad (17)$$

$$y^{(c)} = (1-s)((1-t)y_0^{(c)} + ty_2^{(c)}) + s((1-t)y_1^{(c)} + ty_3^{(c)}) \quad (18)$$

$$z^{(c)} = (1-s)((1-t)z_0^{(c)} + tz_2^{(c)}) + s((1-t)z_1^{(c)} + tz_3^{(c)}) \quad (19)$$

Fig. 7. Specifying the calibration points.

The user calibrates the camera by specifying the four points in the image space as shown in the following figure. The user also provides the corresponding Cartesian coordinates for each of these four points. The program then takes an image pixel coordinate as input and produces a Cartesian coordinate. The Cartesian coordinate can be passed to the inverse kinematics to produce the joint angles to move the arm to the desired drawer position.

F. Communication of Components

In ROS, there are two main paradigms for communication between different nodes. Nodes are modular processes that perform computations: publisher/subscriber and service/client. In the publisher/subscriber paradigm, nodes send out a message by publishing it to a given topic name. This topic has a name specified through a string which identifies the content of the message. Other nodes can subsequently subscribe to a topic and obtain the message. This paradigm extends to many concurrent publishers and subscribers to the same topic.

The service/client paradigm differs from the publisher/subscriber paradigm in that involves a two way request/reply communication as opposed to the one-way publishing of a message. A node will offer a service under a certain name and a client can send this service a message an receive the appropriate reply.

The controller is encapsulated in a publish/subscribe model. This fits with its purpose because it sends a message to each joint in the robotic arm. These messages do not need to receive a reply and it is thus beneficial to use the publish/subscribe model because allows many concurrent subscribers which allow other nodes to listen

in on the arm's current position. In addition, we can specify the frequency and duration of the publishing of the command which is useful in planning out sequential movements of the arm. In contrast, the inverse kinematics calculations are used with a service/client model. This model is appropriate because given certain inputs for the global Cartesian coordinates, we output the subsequent joint space coordinates. Thus, with the client can send a request to the service with the global coordinates and the service will respond with the joint space coordinates. This is important because we only need to make the calculation when a new position is determined for the robot and this can be handled with a single request instead of a series of published messages.

Then, using a separate program, we obtain the location of the object in the image using SIFT feature mapping. Once we have obtained the image coordinate, we continually publish it. Similarly, the central module which unifies the components subscribes to the image coordinate. Using bilinear interpolation, it obtains the 3D global coordinate. From this, a request is sent to the inverse kinematics service and the corresponding joint space coordinates are received. Then, we call the controller to publish the joint space coordinates of each joint to the servos which allows the robot to move to the desired position as seen in the image.

IV. EXPERIMENTS

A. *Control through Specification of Global Coordinates*

Using the controller and inverse kinematics, we were able to specify a global coordinate and subsequently move there. In addition, using the Camera Clicker code, we were able to click on a point in an image from the webcam and the arm would subsequently move there.

In the first experiment, we tested the robustness of the inverse kinematics and the controller. Using a series of preprogrammed commands that specified the global coordinate to move to and whether the fingers of the AX12 Smart Robotic Arm should open or close, we aimed to pick up four bolts at different parts of the plane and drop them into a drawer. The robot arm performed well on this test and was able to pick up each of the four bolts and drop them into the box that was approximately 2 cm by 4 cm from a height of 10 cm.

This demonstrates that the inverse kinematics and the controller were working correctly and had reasonable accuracy in tasks. This is important because robotic problems are generally phrased in global coordinates. In addition, the controller is extremely important so that actions prescribed by the underlying algorithms and transformations can actually be executed in real-life. In addition, many problems of picking up objects has the combination of the inverse kinematics and controller as a fundamental component.

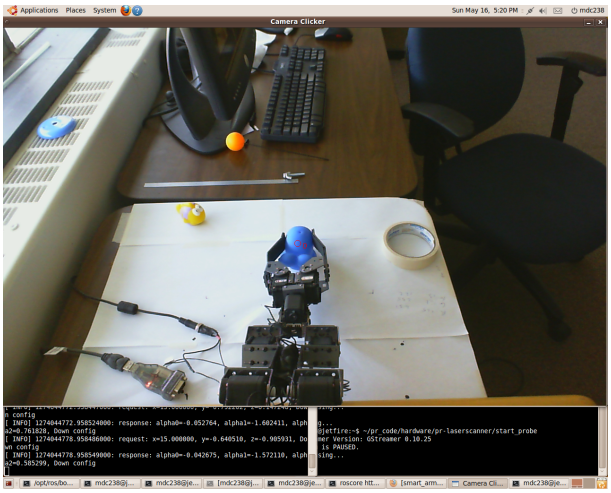

Fig. 8. Before Execution of the Camera Clicker code

Using the camera clicker code, we were able to move the arm to a certain position that was specified in an image as shown in Figure 8 and 9. Once we clicked on a certain number of points, these points on the 2D plane were published. Using bilinear interpolation, we subsequently transformed these to the 3D global Cartesian coordinates. This allowed us to use the controller and inverse kinematics service as described above to move the arm to the desired position.

The success of this test is shown in the images. We can see that before and after, the arm has moved extremely close to the clicked position. This further demonstrates the robustness of the controller and the inverse kinematics service.

Overall, this experiment provided us with the chance to test basic components along with the bilinear interpolation code that will be extremely important in achieving our final goal of opening a drawer.

B. *SIFT Feature Identification*

The SIFT algorithm is supposed to be light and scale invariant, however under very different lighting or when the drawer is greatly angled the algorithm is not be able to find the drawer image key (cropped out of the camera source image for improved accuracy). It is difficult to determine the bounds for exactly how angled or what kind of lighting causes the algorithm to fail both because of the difficulty of quantifying changes in lighting, and because some keys worked better than others. When the same lighting was used for both generating the key and

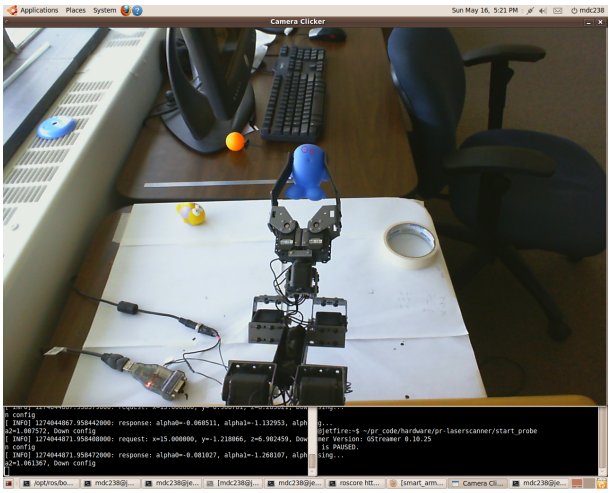

Fig. 9. After Execution of the Camera Clicker code

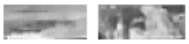

Fig. 10. Left: An uninformative key with low contrast and few details. Right: An ideal key with high contrast and lots of unique features.

finding the drawer and a good key was used the drawer could be angled up to around 37° and SIFT could still find the image accurately.

The SIFT images were tested for accuracy by checking for the same key in the same place multiple times and comparing the results. It was found that the image keys that worked best were ones with high contrast and sharp detail, as shown in Figure 10. The number of features generated for the keys varied from as little as 11 to as many as 53. With good keys chosen a match was almost always ensured, and from running the algorithm 55 times with five different good keys the average number of matches was 6.8 with no incorrect matches found. As stated previously, the algorithm only needs one match to correctly find the drawers pixel location. This means the algorithm found the drawer 100% of the time when using good keys. The pixel coordinate precision was found to be to less than 3 pixels, and most of the time the algorithm would output the exact same coordinates if the drawers were in the same place.

C. Drawer Manipulations

Using the camera and SIFT, the drawers were identified and the optimal grasping point was determined heuristically. Although the container has to remain at a fixed known location, the drawers themselves can move around within the container. Given a successful SIFT identification of the drawers, we were able to successfully open the drawers.

TABLE I

SUCCESS RATE FOR BOTH ALGORITHMS USING DIFFERENT DEFLECTION ANGLES.

Algorithm	Success Rate (0°)	Success Rate (15°)	Success Rate (30°)
1	4/5	4/5	4/5
2	5/5	4/5	3/5

We developed two drawer opening algorithms that allowed us to successfully complete the task. Once the drawer has been successfully identified using SIFT, the first algorithm moves the end-effector to a position that is an inch normal to the drawer grasping point. The fingers then open halfway and the end-effector moves an inch along the normal. At this point, the finger closes and the end-effector moves three inches back along the normal to open the drawer. In order to smooth this movement, an interpolated trajectory along the normal was computed. The second algorithm interpolates a sequence of waypoints so that the end-effector will slide in from the sides of the container. The remaining steps are the same as in the first algorithm.

By moving the end-effector along the normal, we were able to open drawers that were not parallel to the base. We found the second algorithm to be less robust when the drawers were not parallel to the base. The end-effector would sometimes collide with the sides before reaching the drawer. However, the second algorithm had a higher success rate of opening the drawers than the first algorithm when the drawers were parallel to the base. For each algorithm, we ran 5 trials for 3 different drawer deflection angles (0° indicates that the drawer is parallel to the base).

V. CONCLUSIONS

In this project we have developed a hardware and software platform for arm manipulation using the AX-12 Smart Robotic Arm. On the hardware side, we have developed an adjustable mounting system for the sensors. On the software side, we have developed ROS nodes for the controller, inverse kinematics, and drawer manipulations. By using SIFT and heuristic algorithms, we were successfully able to open drawers at various deflection angles. Through 15 trials with 2 different algorithms, we have empirically demonstrated that the robotic arm can robustly perform the task.

Additional work on this project can include developing mechanisms for opening the drawers in unstructured environments. We would like to also incorporate visual servoing so we handle unexpected collisions between the arm and environment. A path planner would be needed

to implement the collision avoidance. We would also like to expand the project to other manipulation tasks like loading dishes and folding towels. We hope to address these challenges in future work.

VI. ACKNOWLEDGEMENTS

The authors would like to acknowledge Professor Ashutosh Saxena for his mentorship and help on the project. They would also like to thank Jonathan Diamond, Yue Gao, Mark Verheggen, Daniel Cabrini-Hauagge, and Jigar J. Shah for their help in CS 4758 and contributions to the project.

Additional components to the project include: Code commits to the Personal Robotics Repository and videos/pictures submitted through Professor Saxena.

REFERENCES

[1] Ashutosh Saxena, Justin Driemeyer, Justin Kearns, Chioma Osondu, and Andrew Y. Ng,"Learning to Grasp Novel Objects Using Vision", *in 10th International Symposium on Experimental Robotics (ISER)*, Rio De Janeiro, Brazil, 1961.

[2] Rena N. Jazar, *Theory of Applied Robotics: Kinematics, Dynamics, and Control*. Springer, 1st Edition, 2007.