EE366L/CE366L: Introduction to Robotics

Lab Handbook

Second Edition, Spring 2023

If you come across any errors, contact me at

basit.memon@sse.habib.edu.pk

Table of Contents

Preface

This handbook was developed for the companion lab to 'EE 366/CE 366/CS 380: Introduction to Robotics' course offered at Habib University in the Dhanani School of Science and Engineering. While developing this lab, I have tried to achieve three objectives: (i) provide students the experience of building complex robotic systems from constituent subsystems; (ii) train students to adjust for the differences between theoretical models and physical systems in their system design; (iii) build the students' confidence and prepare them for building robots independently.

I believe that the best way to achieve these objectives is for the students to build the sub-systems themselves from the ground up. Admittedly, this approach limits the students to simple robotic systems given the available time, but if the students' understanding is enhanced through these simpler systems then it will be easier for them to extrapolate to more complex robotic systems. Unfamiliarity with ROS can be considered a shortcoming of this approach, and I will recommend you to acquire a passing familiarity with it if you get a chance, and the understanding of homogeneous transformations acquired in this course will certainly make it convenient to understand transform trees in ROS.

Basit Memon
January 8, 2023

Acknowledgments

I would like to acknowledge the help of Mr. Waleed Bin Zaid and Mr. Hassan Shah, who were RAs in DSSE, for their assistance in setting up the robotics lab. I would also like to thank the students enrolled in 2022 who volunteered their time to assemble the robot arms during the winter break. It would not have been possible to conduct this lab timely, otherwise.

I would also acknowledge the continued support and assistance of Mr. Hassan Shah, RA for this course in Spring 22. His passion, insights, research, and debugging efforts have led to this improved second edition of this handbook.

0

CHAPTER

Lab Instructions

1. The credentials for signing in to the lab computers are:

```
Username: robo
Password: Habib123++
```

2. Since you're working in groups, rotate among yourselves so that each member of the group has a chance to get hands-on experience.

3. Tasks marked with an [*] don't require access to the hardware and can be completed post-lab.

4. At the end of the lab, you're required to unplug the arm from the power supply and turn off the lab computer's monitor. The arm may relax and fall down as it is turned off, so be sure to catch it and carefully place it on the baseboard.

5. The weekly lab findings report is a group submission.

"It is not knowledge, but the act of learning, not possession but the act of getting there, which grants the greatest enjoyment."

- Carl Friedrich Gauss

1

CHAPTER

Getting familiar with MATLAB

As MATLAB will be used extensively throughout this course and students may have different levels of familiarity with it, today's lab is dedicated to familiarizing yourself with the MATLAB environment. For this assignment, you're required to complete the MATLAB On-ramp (`https://matlabacademy.mathworks.com/details/matlab-onramp/gettingstarted`) online course and upload the completion certificate.

This is a two-hours course, which provides a basic understanding of MATLAB's IDE, data storage and manipulation, and programming constructs. If you're already familiar with MATLAB, this will be a boring exercise and you can reach out to your instructors to select an advanced MATLAB course to complete for this assignment.

Task 1.1 Team Formation (10 points)

Due to limited number of lab setups, you'll be working on your arm project in groups. You are required to enroll yourself as part of a group on Canvas. If you're unfamiliar with the process of self-enrolling in a group on Canvas, you can follow a guide on Canvas.

Task 1.2 MATLAB On-Ramp (90 points)

Attach the completion certificate for the MATLAB On-Ramp course.

"If science is to progress, what we need is the ability to experiment."

– Richard P. Feynman

CHAPTER 2

Getting familiar with the arm hardware

The objective of this lab is to get yourself familiar with the hardware and capabilities of the Phantom X Pincher arm, shown in Figure 2.1 and manufactured by Trossen Robotics (`https://www.trossenrobotics.com`), which will be utilized for this project. During the course of this lab, you'll also get a sense of the motion and manipulation limitations of this arm and become familiar with the variables used to measure performance. A 3D interactive model of the same arm is provided by the manufacturer at this link: `https://www.trossenrobotics.com/p/PhantomX-Pincher-Robot-Arm.aspx`. The 3D model is built with the same dimensions as physical arm, and also allows you to measure the distances and angles at different points on the robot.

2.1 Design of the arm

The arm has 5 motors that serve as actuators, a pinch grasper, and a microcontroller board at the bottom. We'll discuss each of these parts in greater detail later in this lab. While the arm is not powered up, feel free to move the arm by hand for the following tasks. **Don't try to move the arm forcefully by hand when it is powered up, as the motors will be providing torque and it will resist motion, and you may damage the arm.**

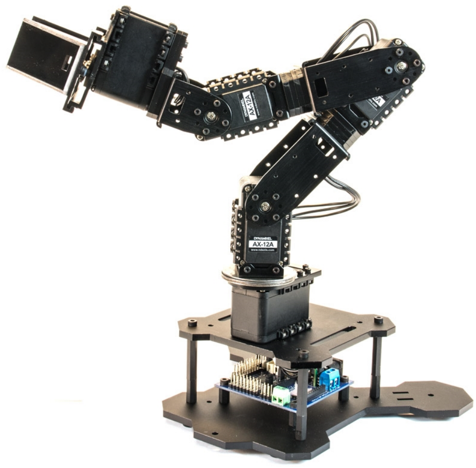

Figure 2.1: Phantom X Pincher Arm

Task 2.1 Model of the arm (20 points)

We know that a robot manipulator is mathematically modeled as a kinematic chain, made up of joints and links. Identify all the joints and links in this arm.

(a) Mark all the joints and links in Figure 2.1 or any other image of the arm.

(b) How many joints and links are in this arm? Note that the motor attached to the grasper is only responsible for opening and closing the grasper.

(c) What is the joint type? Provide a symbolic representation of the kinematic chain corresponding to this arm. Recall that a kinematic chain is symbolically represented as a sequence of joint symbols.

(d) How many degrees of freedom does this arm possess? *Hint: You can use Grubler's formula from the class slides.*

2.2 The Arbotix-M Controller

The Arbotix-M controller, depicted in Figure 2.3, lies at the heart of this arm. This is an informational section that provides relevant details of this controller. This controller receives higher level instructions from an external processing unit and generates instructions in specific format required for the servomotors (actuators) installed in this arm. In our case, a computer will serve as external processing unit and we'll pass instructions from MATLAB to Arbotix-M, as shown in Figure 2.2.

This board is capable of doing more than controlling the servomotors; some of these capabilities are outlined in this section. If you were to take out the Arbotix-M board and study it, then Figure 2.4 would depict the purpose of the various sections on this board. All the processing is actually done on this small square-shaped chip in the middle, specifically by an ATMEGA644p AVR

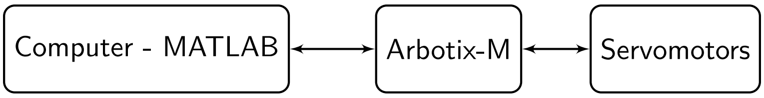

Figure 2.2: Data flow between MATLAB and servomotors

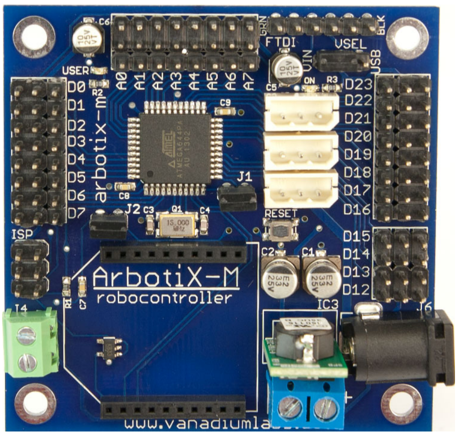

Figure 2.3: The Arbotix-M Controller

microcontroller, which is in the same category of microcontrollers as an Arduino. In fact, we'll be using the Arduino IDE to program the Arbotix-M in C. For the interested, this page https://learn.trossenrobotics.com/arbotix/arbotix-getting-started/38-arbotix-m-hardware-overview provides further hardware details.

For the time being, we'll only be using this board for controlling the servomotors in the arm. For this, the board can be set up as shown in Figure 2.5. We simply need to

- power the board via the DC power jack, which will provide power to all the connected motors as well;
- set up communication between the board and the computer using the provided FTDI-USB cable; this cable can only be connected in one way as shown in Figure 2.6; the port on the board also indicates which side corresponds to the green cable and which side to the black;
- connect the servomotors to the Dynamixel servo ports.

In addition to this, the required firmware files should be copied to `Documents\Arduino` folder on Windows as indicated in [4]. We'll verify this in 2.3.1.

2.2.1 Connecting motors to Arbotix-M

The motors are connected in a daisy chain, i.e. only the base motor is connected to the Arbotix, the second motor is connected to the first, and so on serially all the way to the grasper motor. Then, how does Arbotix communicate with a specific motor? Each motor is assigned an ID, Arbotix addresses each instruction message to the intended ID, and places it on the common chain/bus. The IDs assigned to the five motors on the arm are give in Figure 2.7. You can also broadcast messages intended for all motors.

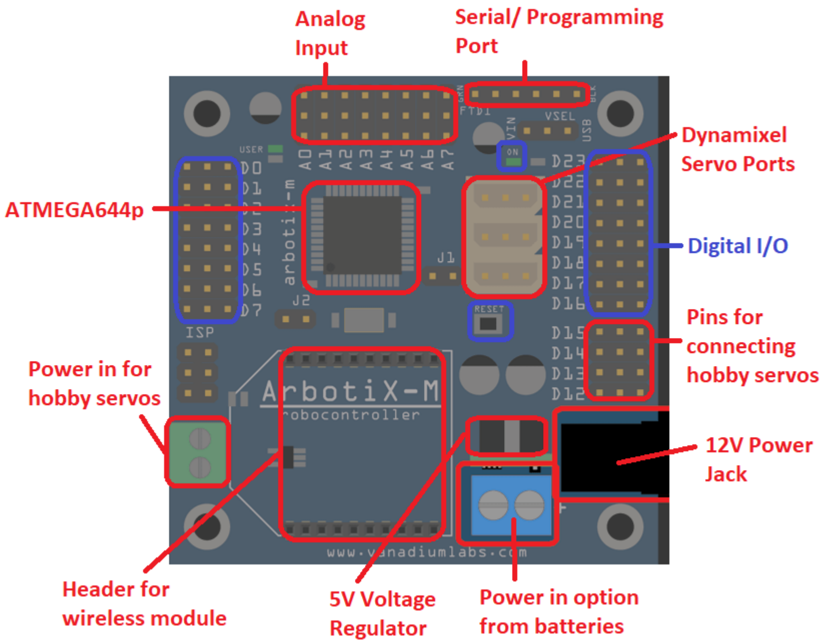

Figure 2.4: Getting to know Arbotix-M

2.3 Move the arm

We'll now use a simple interface, InterbotiX Arm Link Software, provided by the manufacturer to control the arm. In the future, we'll write our own program to control it.

2.3.1 Setting up Armlink

1. Connect the power jack to the Arbotix-M and make sure that FTDI-USB cable is plugged into a USB port in your computer. When the arm is first powered up, it may move to its 'sleep' position and then turn torque off to all the servos. Don't be alarmed by the motion and don't interrupt its execution.

2. Open the Arduino IDE. It is already installed on the lab computers. The firmware provided for the arm is only compatible with an older version of Arduino IDE, specifically ver.1.0.6.

3. We'll now need to load the appropriate firmware on Arbotix-M that will allow it to communicate with the Armlink software, assuming that the firmware is appropriately placed on your computer as specified in [4]. This will be verified if you're able to carry out the following steps.

   (i) Verify that the libraries `ArmLink` and `Bioloid` are available under `Sketch`.

   `Sketch -> ArmLink`

   (ii) Select the appropriate board and programmer as follows:

   ```
   Tools -> Board -> ArbotiX
   Tools -> Programmer -> AVRISP mkII (Serial)
   ```

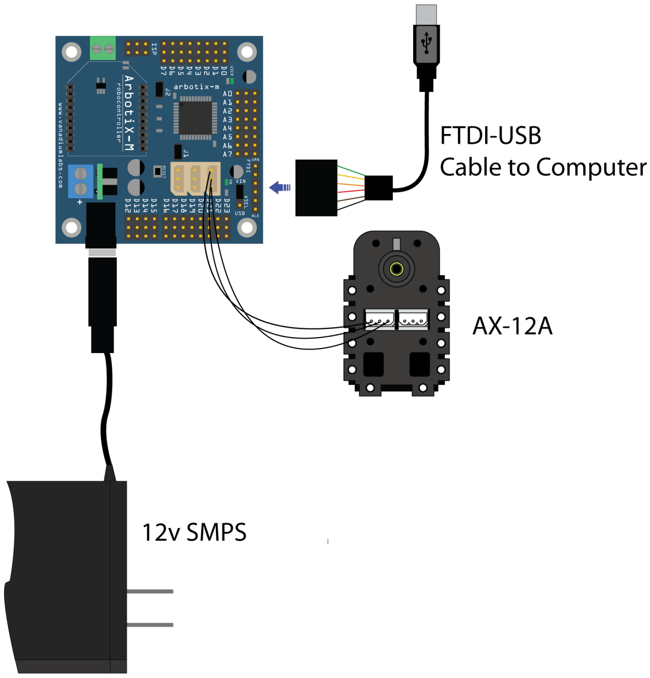

Figure 2.5: Setting up the Arbotix-M with power, servos, and programming

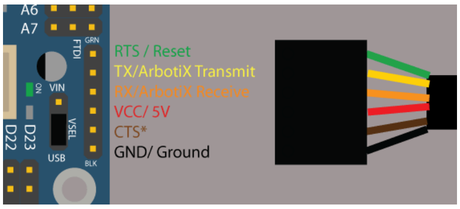

Figure 2.6: FTDI-USB Cable Connection

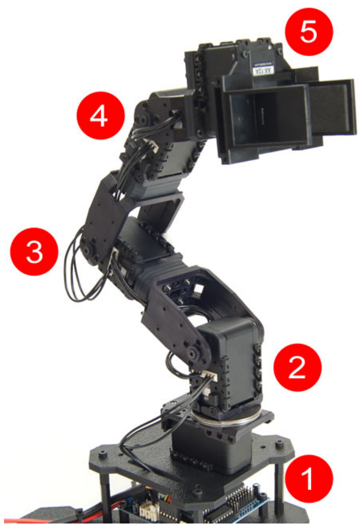

Figure 2.7: IDs of the servos in arm

(iii) Open the `ArmLinkSerial` firmware from the Arduino IDE (Arobotix-M firmware requires Arduino 1.0.6).

`File -> Examples -> Arm Link -> InterbotixArmLinkSerial`

(iv) You have to select our arm model by uncommenting, i.e. remove `//`, from line number 60 in the code. The line should read:

`#define ARMTYPE PINCHER`

(v) Load this firmware onto the Arbotix-M, by clicking on the `Upload` icon, which is a right arrow, in the toolbar or from the menu,

`Sketch -> Upload`

(vi) Once the firmware is uploaded, you will see `Done Uploading` message in the green bar at the bottom of your IDE. This firmware sets up a protocol for Arbotix-M to communicate with the ArmLink software over USB, and convert received messages to instructions for motors.

4. Open the `ArmLink` application. The application is already copied to `Desktop\ArmLink_1.6_Win64` on the lab computers. When the application is launched, click on `Auto Search`. This will search for the attached arm and connect to it.

5. On a successful connection, the arm will move from it's 'sleep' position to a 'home' position. This may take several seconds. Once the arm has moved to its home position and is ready for commands, the various panels will appear as shown in Figure 2.8. **Once the arm is connected to the software, don't try to move it by hand as each of the motors will exert torque.**

6. You can adjust the sliders or text panels to adjust the positions of the arm. You can send these values to Arbotix-M by clicking on `Update`, or you can check `Auto Update`, in which case instructions will be sent continuously.

2.3.2 Getting familiar with Arm Link Controls

Read through this and the next section to be fully aware of safety instructions before playing around with the controls. Change the panel coordinates gradually so that it does not collide with itself or an object in the environment. It may appear that the arm is not moving, but it requires some time to process the received coordinates. Always be alert to stop the arm from colliding with itself or the environment. You can stop it by clicking on the 'Emergency Stop' button or unplugging power.

The software provide three modes of operation - two in the task space representation, and one where you can control each joint individually. These modes can be selected from the bottom of the panel.

1. **Cartesian:** In this task-space mode, you can specify the X-Y-Z coordinates in the task space and the software will move the end-effector to that location.

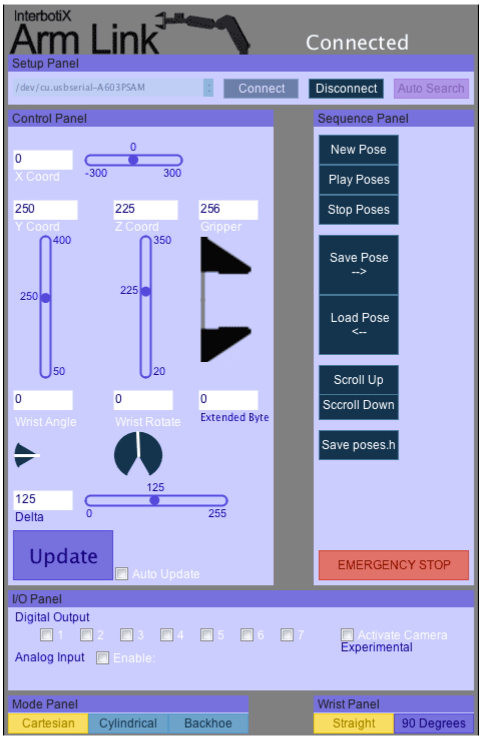

Figure 2.8: Arm Link Application Panel

2. **Cylindrical:** As the name suggests, you can specify the cylindrical coordinate for the placement of the end-effector in this mode. The panel gives you the option to set the base angle, Y, and Z coordinates.

3. **Backhoe:** In this mode, you can directly control the position of the base, shoulder, and elbow joints.

In addition to this,

- The `Straight` and `90 Degrees` wrist option place the wrist in the horizontal or vertical configuration.

- Each mode allows you to rotate the wrist.

- Each mode allows you to open and close the gripper.

- The `Delta` value determines how long it will take for the arm to move from its current position to the new position. The amount of time that the move takes is calculated by multiplying `Delta` by 16. The result will give you the time interval in milliseconds.

- The right panel allow you to save different poses to create a sequence and play it on the arm.

For further details on the controls, refer to [3].

2.3.3 Common Problems

What to do if you have provided commands from the software panel, but the arm is not moving as intended? Make sure that you have waited sufficiently, as the arm requires some time to process any received instructions. If sufficient time has passed and the arm is still not moving, check that

(i) the motor cables for any of the motors have not wrapped around, restricting the motion of the arm;

(ii) none of the motors have a flashing red indicator light; the light indicates one of these listed events: over-temperature, joint angle instruction exceeds allowed limit, excessive torque.

If you're aware of the instruction causing this error, reverse it. Or, you could reset the arm by powering it off and manually moving it to zero configuration, if needed. If it is an over-temperature event, the arm will have to be powered off for a longer duration for the arm motors to cool down.

Task 2.2 Configurations Exploration (20 points)

Play around with the different modes of motion in the software and explore the capabilities and limitations of this arm.

(a) Move the arm to a configuration in which it reaches the farthest possible point. Draw this configuration as a diagram. In this diagram, links can be represented by line segments and revolute joints by circles.

(b) In Cartesian mode, move the robot to an arbitrary (x, y, z) location. Change the wrist angle from the panel and observe what happens to the other joints of the arm. Document your observations and comment on the reasons behind what you observe.

(c) Grab one each of the provided objects. In this task, you'll place each of these objects at a fixed location in your workspace, move the arm using `ArmLink` to that location, pick the object, and place it at another location. During this activity, how is the real world environment being sensed and how is the arm motion being adjusted based on the received sensing data? Where is this processing happening?

(d) [*][a] The coordinates in the Cartesian or Cylindrical mode describe task space locations. Task space can be used to describe tasks to be carried out by the manipulator, e.g. grabbing a water bottle. Give an example of a task that can be described better in Cartesian coordinates, and a task, which is best expressed in cylindrical coordinates.

[a]Tasks marked with [*] can be completed post-lab.

2.4 Establishing a coordinate system

The design of our autonomous pick and place pipeline will also be based on a vision-based feedback loop, similar to the one determined by you in the previous task. A camera will take the place of our eyes in this loop. This loop is formally termed as visual servoing[1]. There are two ways for visual servoing:

1. **Position-based visual servoing:** Coordinates of positions of interest are determined from the image and the robot motion controller moves the arm to desired positions.

2. **Image-based visual servoing:** This marks the positions of interest and robot positions in the image and the robot motion controller moves the arm so that robot is at the desired position inside the image.

In this project, we'll employ position-based visual servoing. To do so, we'll have to agree on a global coordinate system for specifying all positions. The numbers appearing in the Arm Link panels for X, Y, Z values appear to map the physically possible range along an axis onto a subset of integers, e.g. X, onto a subset of integers between 0 and 1023[2]. So, this must be based on an underlying coordinate system and we could use the same in our design.
inform resolution is 1:1 identify the unit where's the end point?

Task 2.3 Coordinate Axes (10 points)

(a) Determine the directions of positive x, y, and z axes and mark them on paper, in relation to the shape of the black base.

(b) We'll set the origin of the x and y axes at the center of the shaft of the first motor, and the origin of the z axis at the level of the wood platform. If 1 unit in the ArmLink system corresponds to 1 unit in the real world, identify the units being utilized in the real world and the point of the arm whose position is being determined.

2.5 Pose Accuracy and Repeatability

Two parameters that typically characterize the performance of a robot are its pose accuracy and pose resolution. ISO 9283:1998 defines both these characteristics for industrial arms and the methods for measuring them. In this section, you'll gain familiarity with both these characteristics and determine them for the robot arm at hand.

[1]The term servo has a specified meaning in engineering literature, which will be defined later in the current chapter.

[2]The number range is 0-1023, because the arm servos use a 10 bit location for motor angle. Limitations on the possible range are governed by (a) the physical angular limits of the motors, (b) safety margins to avoid self-collisions of the arm.

Pose Accuracy

It expresses the deviation between a command pose and the mean of the attained poses when approaching the command pose from the same direction. It is divided into positioning accuracy (AP_P) and orientation accuracy (AP_O).

The positioning accuracy is calculated as follows:

$$AP_P = \sqrt{(\bar{x} - x_c)^2 + (\bar{y} - y_c)^2 + (\bar{z} - z_c)^2}$$

where

$$\bar{x} = \frac{1}{n}\sum_{j=1}^{n} x_j$$
$$\bar{y} = \frac{1}{n}\sum_{j=1}^{n} y_j$$
$$\bar{z} = \frac{1}{n}\sum_{j=1}^{n} z_j,$$

(x_c, y_c, z_c) are coordinates of the command pose, (x_j, y_j, z_j) are the coordinates of the j-th attained pose, and n is the number of times the same command is given.

Pose Repeatability

It expresses the closeness of agreement between the attained poses after n repeat visits to the same command pose in the same direction. The value of position repeatability is the radius of sphere enclosing the set of locations the manipulator achieves for the same command with the same load and setup conditions.

The positioning repeatability RP_l can be calculated as:

$$RP_l = \bar{l} + 3S_l,$$

where

$$\bar{l} = \frac{1}{n}\sum_{j=1}^{n} l_j$$
$$l_j = \sqrt{(x_j - \bar{x})^2 + (y_j - \bar{y})^2 + (z_j - \bar{z})^2}$$
$$S_l = \sqrt{\frac{\sum_{j=1}^{n}(l_j - \bar{l})^2}{n-1}},$$

with $(\bar{x}, \bar{y}, \bar{z})$, (x_j, y_j, z_j), and n are as defined previously.

Figure 2.9 highlights four different repeatability and accuracy combinations in 2D. For position-based visual servoing (the one we're employing), there is no direct measurement of the end-effector position and orientation, and instead one relies on the assumed geometry of manipulator and its rigidity to calculate end-effector position from measured joint positions (we install sensors to measure joint angles). Therefore, accuracy is affected by computational errors, machining

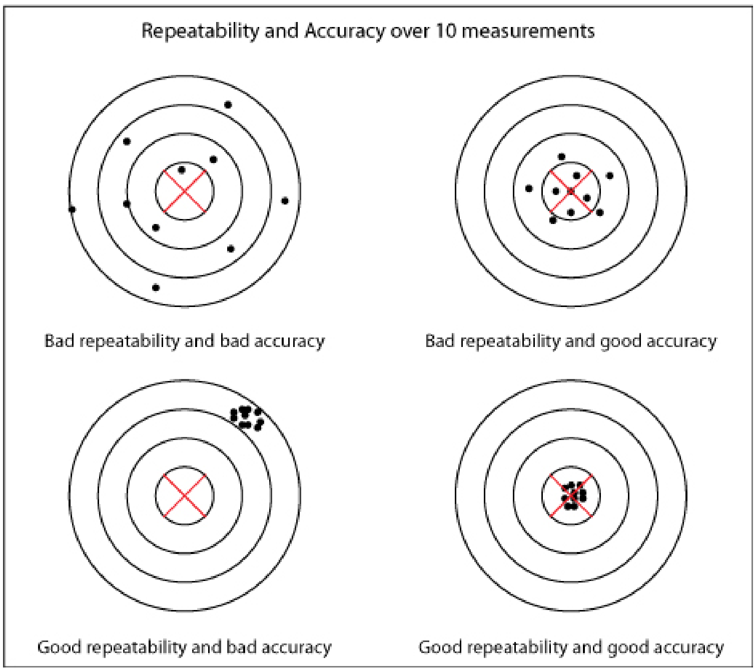

Figure 2.9: Red cross indicates commanded pose and the black dots achieved poses over multiple iterations of the same command.

accuracy in the construction of manipulator, flexibility effects such as bending of the links, gear backlash, other static and dynamic effects, and the accuracy of the solution routine. Repeatability depends on the accuracy and resolution.

The standard also provides a procedure for measuring the two characteristics. Some key points from the procedure are:

- The wrist reference point is considered as the endpoint for positioning accuracy and repeatability.
- The trials are to be performed at 100% of rated velocity and 100% of the rated load.
- n should be 30.
- Te robot passes through five poses P_1 through P_5, which are located on the diagonals of the larges cube that the robot can achieve.

For more details, read pages 10-23 in the standard document uploaded on LMS.

Task 2.4 Accuracy and Repeatability (50 points)

Identify a method for physically measuring the position of the wrist reference point of a robot arm.

(a) Design an experiment for determining the accuracy and repeatability of the robot arm and provide details of this experiment.

(b) Conduct sufficient trials of this experiment and determine values for positioning accuracy and repeatability.

(c) Does the accuracy of this robot vary with distance from the base? How will you determine it?

(d) [*] Research the positioning accuracy and repeatability of industrial arms and compare the values with the ones obtained in the previous part. How do they compare?

(e) [*] If the accuracy and/or repeatability of our robot arm is lower, how will it impact the performance of our pick and place pipeline?

2.6 Teaching Pendant

Most robot arms come with the ability to learn a motion, by allowing the user to move it by hand and storing the sensor values at various waypoints on the path. To teach a motion to the robot, we'll use another script, `DYNAPose`, as it allows us to relax the servos and move the arm by hand to teach it a pose.

(i) Follow the instructions in the *Download Code* section at https://learn.trossenrobotics.com/8-arbotix/131-dynapose-dynamixel-arbotix-pose-tool.html to upload the appropriate firmware on Arbotix-M for this task. Note the following:

- The code file is downloaded and available on the desktop on lab computers.
- The serial monitor is opened by clicking on the magnifying glass icon in the top-right corner of IDE.
- Don't forget to set the baud rate. This is the rate (bits/s) at which IDE will communicate with Arobotix-M.

(ii) The menu options provided on the webpage are incorrect and options specified in DYNAPose code are as follows:

```
0: Relax Servos
1: Enable Torque and Report Servo Position
2: Save Current Position to next pose(2)
3: Display All Poses in memory
4: Play Sequence Once
5: Play Sequence Repeat
6: Change Speed
7: Set Next Pose Number
8: Center All Servos
9: Scan All Servos
```

(iii) Relax the servos, which will allow us to move the arm by hand. This is option 0 in the menu.

(iv) Teach a simple motion to the robot. This could be as simple as moving to the corner of an object, which will not change its location, executed in at most 5 steps. At each step, after positioning the arm by hand, save that pose (Option 2). The final pose will be when the end-effector is at the corner of the object.

(v) Play the sequence of poses (option 4)

Task 2.5 Bonus (20 points)

Now, let's try a multi-step task. Think of a simple task, involving both positioning of arm and manipulation, which you would want to carry out with the arm right now, e.g. writing a single alphabet on paper. Teach the arm (use `DYNAPose`) to carry it out by recording poses at sufficient gaps and recreate that motion. Some questions you may have to think about are: How many save points do you need to describe your task? What poses should you save? When do you lift your arm off the page? What is the best orientation to grab a pen? For your submission, provide a written description of the task, a video of the execution, and your comments on how it can be improved.

2.7 References

1. Manufacturer provided arm resources: https://learn.trossenrobotics.com/

2. Arbotix-M Controller resources https://www.trossenrobotics.com/p/arbotix-robot-controller.aspx

3. ArmLink: https://learn.trossenrobotics.com/20-interbotix/robot-arms/137-interbotix-arm html

4. Getting started with Arbotix-M https://learn.trossenrobotics.com/arbotix/7-arbotix-quick-st step3