Final Report Guidelines for Introduction to Robotics Lab

Afsah Hyder, Maleeha Hussain, and Ailiya Fatima
Habib University, Pakistan

Abstract—**This project addressed the challenge of enabling a robotic arm to perform autonomous object manipulation tasks. The core objective was to develop a vision system that would guide a pincher arm, using an overhead camera, to locate, grasp, and place objects within its workspace.**

The significance lies in the potential applications of such a system in various industries, including manufacturing, logistics, and assembly lines. Automating pick-and-place tasks can improve efficiency, reduce human error, and enable robots to handle delicate or hazardous objects.

Our key solution involved a robotic vision system with several functionalities: frame assignment for establishing a common reference point, object perception using image processing for detection and recognition, precise forward kinematics control to translate perceived information into actionable commands, and finding inverse kinematic solutions to determine robot arm configurations for grasping and placing objects.

The system's performance was evaluated through controlled experiments using objects of varying characteristics. Pick-and-place accuracy, measured by Euclidean distance between desired and achieved placement, served as the primary metric. The results will allow us to assess the system's effectiveness and identify areas for improvement.

Throughout this project, we gained valuable insights into robotic vision systems, object manipulation, and the importance of factors like camera calibration, object recognition accuracy, and gripper precision.

Future steps involve refining the system based on performance data analysis. This may include optimizing image processing algorithms for better object recognition, improving gripper control for more precise grasping, and exploring methods to enhance robustness in handling diverse lighting conditions and object types. Additionally, integrating machine learning algorithms for object recognition and grasping strategies could be explored to further enhance the system's adaptability and intelligence.future steps. This is usually written at the end.

Index Terms—**Phantom X Pincher, Robot Manipulator, Pick and Place**

I. Introduction

This project focuses on developing a robotic vision system for object manipulation. The core objective is to enable a pincher robotic arm, guided by an overhead camera, to perform autonomous pick-and-place operations. This project delves into the theoretical foundations of robotic arm control, particularly forward and inverse kinematics, alongside computer vision techniques for object detection and localization. The successful development of this system has significant value for broader robotic applications across various industries. Pick-and-place tasks are fundamental operations in numerous fields, including manufacturing (automating assembly lines, product handling, and machine tending), logistics and warehousing (streamlining order fulfillment and inventory management), agriculture (assisting with harvesting, sorting, and packaging crops), and even domestic applications (facilitating tasks in assisted living environments). A functional vision-guided robotic system can significantly improve efficiency, precision, and safety in these areas.

II. Design Details

The system will be divided into several key functionalities, each addressed in its own subsection:

1) **Frame Assignment:** This subsection details the process of establishing a common frame of reference for the robot arm and the camera. This ensures accurate translation of visual information into actionable commands for the robot arm.
2) **Perception:** This subsection focuses on object detection and recognition within the camera's field of view. Image processing techniques will be employed to identify the target object, including its location and orientation.
3) **Mapping from the image coordinates to the world frame coordinates:** This subsection describes the transformation of the image coordinates from image coordinates to camera coordinates and then from camera coordinates to to world coordinates.
4) **Precise Forward Kinematics Control:** This subsection describes the methodology for translating the perceived object's position and orientation into precise commands for the robot arm's joints. This ensures the arm reaches the target object with the appropriate trajectory and positioning.
5) **Finding Inverse Kinematic Solutions:** This subsection addresses the process of determining the joint configurations (angles) of the robot arm required to achieve the desired end-effector (gripper) pose (position and orientation) for grasping and placing the object.

A. Functional Architecture

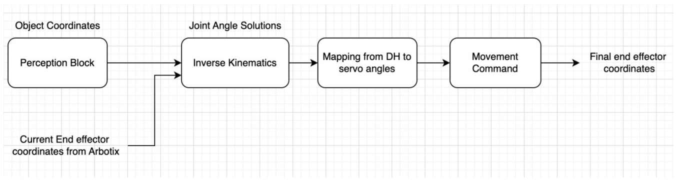

Object Coordinates
Joint Angle Solutions
Perception Block
Inverse Kinematics
Mapping from DH to servo angles
Movement Command
Final end effector coordinates
Current End effector coordinates from Arbotix

Fig. 1. Functional block diagram

B. Frame Assignment

Figure 1 illustrates the Denavit-Hartenberg (DH) frame assignment for the Phantom X Pincher arm [1].

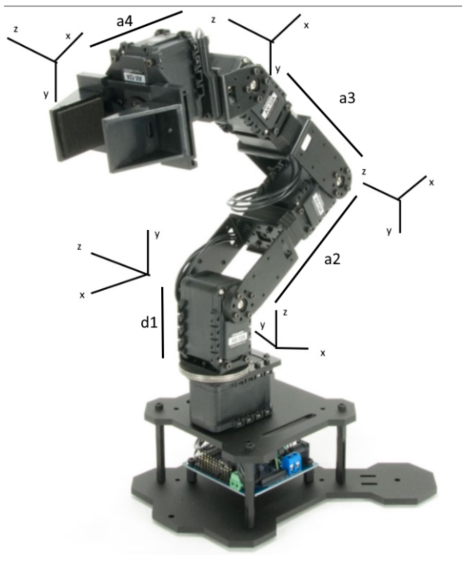


Fig. 2. DH Frame Assignment

Frame	a_i	α_i	d_i	θ_i
1	0	$\frac{\pi}{2}$	54	θ_1
2	108	0	0	θ_2
3	108	0	0	θ_3
4	76	0	0	θ_4

TABLE I
DH PARAMETERS

C. Perception

This part allows the camera to correctly identify the object and it's position in the image. This information is used to generate an orientation and position of the object. Later on, the information from the image plane would get converted to camera frame, and then transformed to the robot frame so that the robot can identify the object placement.

1) Method 1: Pose estimation using Object Detection & Corner Detection: One of the ways to assign frame to the object is to identify corners and place the unit vectors there.

The initial stage involves color segmentation using the Lab color space, which separates lightness from color information to define precise color ranges for cube colors. Pixels falling within these Lab value ranges are classified as part of the target object. Connected regions of segmented pixels, corresponding to individual cubes, are identified through blob analysis. This method is particularly effective for scenarios featuring distinct color objects against backgrounds with minimal color variation.
In the subsequent step, post-processing techniques refine the identified object regions. Morphological filtering methods like opening and closing are applied to eliminate noise and smooth object boundaries. Additionally, a minimum area threshold is used to filter out small, potentially spurious blobs resulting from noise or artifacts. Effective filtering relies on parameter tuning, typically refined through iterative testing and adjustments based on observed outcomes.
Once a clear mask of the desired object is obtained, the mask is filled to ensure smoother object boundaries and easier edge detection. The MATLAB 'edge' function is then employed to detect edges of the mask, followed by using the 'detect Harris Features' function to identify corners from these edges. This process yields coordinates of all corners, from which the most significant corner is selected based on specific ranking criteria.
Constraints

- Lighting Conditions: The above color segmentation worked only under specific well lit part of the lab.
- Number of blocks: The color mask can identify all present objects. The corner detection works after a single block is chosen.
- The camera was set up at a particular position to cover the entire workspace. The exposure, brightness settings etc were particular to a specific location.

2) Method 2: Pose estimation using Point clouds: Another method to achieve the same goal as above is to use point clouds. In order to generate the point cloud, the example code provided is used. It is then further changed according to our use case.

The initial steps involve setting up the RealSense camera and capturing depth and color frames. Here, the code discards a few initial frames to allow for camera stabilization and performs an alignment step to account for the slight difference in viewpoints between the depth and color sensors. Once captured, the depth data is extracted and converted into a usable format.

The second section dives into color segmentation, which is crucial in this case. The mask generator function takes the color frame as input and generates a 2D mask. This mask identifies pixels corresponding to the color objects in the image. The mask is then applied to the depth data to isolate depth information relevant to potentially colored areas, likely representing a colored cube. Two approaches are presented for masking: element-wise multiplication (commented out) and setting all non-masked depth values to infinity.

The purpose of the third section is to create a point cloud. This is achieved by first creating a camera intrinsics object using the retrieved camera properties. Then, the pcfromdepth function is used to generate a point cloud from the masked depth data. Additional parameters include the depth scaling factor and the optional color frame. The code might also include a step to remove invalid points from the point cloud for better quality.

Pose estimation would involve identifying clusters of points (potentially objects) in the point cloud, calculating their centroids (centers), and finding the corner points closest to the x-axis for each cluster. These points would be used to define vectors representing the objects' axes, and together with the centroids, they would define the complete pose (position and orientation) of the objects within the scene.

Refinement to the approaches include implementing the pose estimation step would provide valuable information about the objects' orientations. Finally, by analyzing the points that fit the plane (inlier indices), the code could potentially identify the cube's corners and map them back to pixel coordinates. This information, along with techniques from Lab 03, could be used to find specific locations on the object, such as the bottom-left corner.

D. Conversion of image coordinates to world coordinates

The pixel coordinates (u, v) obtained from the perception step are utilized to determine the object's location (x, y) within the camera's reference frame.
The object's locationexists relative to the camera's perspective. To instruct the robot arm, we need its position in the world frame (which coincides with the object's frame). A pre-defined transformation matrix is employed to convert the camera frame coordinates (x, y) into world frame coordinates.

```
z=69; %fixed value for z since all blocks
     had the same height

u= bottom_left(1); v= bottom_left(2);
       %  we only took bottom right, bcs
    the masks were for different lighting
    conditions, we weren't able to change
    it later on
u0=312.5867 ; v0=243.7; %intrinsic
    parameters of the camera
fx=481 ; fy=481;

x=((u-u0)/fx)*z %finding x and y
    coordinates
y=((v-v0)/fy)*z


th=-pi
TcW=[1,0,0,0;     %transformation matrix
    from camera to world coordinates
     0,cos(th),-sin(th),0;
     0,sin(th),cos(th),z;
     0,0,0,1;]
inv(TcW)
pc=[x;y;z-3;1]
pw=inv(TcW)*pc

PickandPlace2(pw(1),pw(2),-3,-14,-14,-3)
```

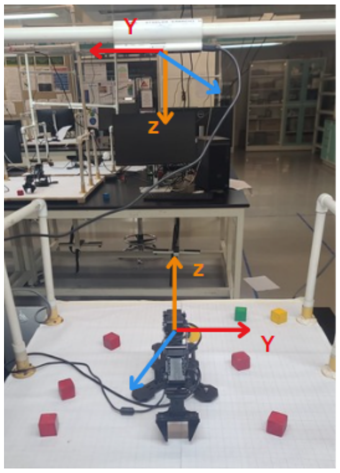


Fig. 3. The assignment of the world frame(on robot) and the camera frame(on camera)

E. Enabling Precise Control: Forward Kinematics

Forward Kinematics is essential for precise control of robot arms during object manipulation. Here's a breakdown of the process used in our system:

DH Frame Assignment and Parameterization: We use the Denavit-Hartenberg (DH) convention to establish reference frames for each robot arm link. Each joint is assigned four parameters (α, a, θ, d) to define the transformation between adjacent link frames.

Homogeneous Transformations: Based on the DH parameters, we construct homogeneous transformation matrices representing the position and orientation between consecutive link frames. Multiplying these matrices sequentially allows us to transform from the base frame (robot arm mount) to the end-effector frame (gripper).

Forward Kinematics Function: The pincherFK function takes joint angles and calculates the end-effector's pose. It leverages DH parameters to build link transformations, which are then multiplied sequentially. This computation yields the final transformation matrix representing the end-effector's position (x, y, z) and orientation (e.g., rotation matrix or Euler angles).

Workspace Verification: Considering servo angle limits, we define the robot's reachable workspace. This volume represents where the end-effector can physically reach based on joint limitations. By using the pincherFK function to calculate the reachable workspace, the system identifies if a desired object location is feasible, preventing attempts to reach unreachable locations.

DH and Servo Angle Alignment: DH parameterization defines the relationship between joint angles and end-effector pose. However, servo motors use their own range. We implement a mapping function to convert between DH joint angles used in FK calculations and actual servo angles needed by the robot's control system, ensuring coordinated movement.

F. Controlling Robot Movements with Inverse Kinematics

Once the object's corners are identified, controlling the robot arm to reach that location involves inverse kinematics (IK).

This section explores how to determine the joint angles required for the robot to achieve a desired end-effector (gripper) position and orientation.

Mathematical Solutions: You've developed a function findJointAngles(x, y, z, phi) that calculates the IK solution. It takes the end-effector's position (x, y, z) and orientation (phi) as arguments and returns a matrix containing potentially multiple solutions (N x 4, where N is the number of solutions and 4 represents the number of joints in the robot arm).

Verifying Solutions: The forward kinematics function is used to verify the output of the findJointAngles function. This ensures the calculated joint angles indeed lead to the desired end-effector pose.

Selecting the Optimal Solution: Another function, findOptimalSolution(x, y, z, phi, currentConfig), selects the most suitable solution from the potentially multiple options returned by findJointAngles. This selection criterion considers the proximity of the proposed joint angles to the robot's current configuration (currentConfig). Choosing the closest solution minimizes unnecessary joint movements, optimizing motion efficiency.

Converting for Servo Control: Finally, the chosen joint angles are converted from the DH parameterization format to the specific format required by the servo motors controlling the robot arm. This allows the robot to translate the calculated joint angles into actual physical movements and reach the desired object location.

III. Results

Defining Success and Metrics:

Success for this robotic vision system is measured by its ability to autonomously complete pick-and-place operations within a specified tolerance level. The primary metric for performance is pick-and-place accuracy, defined as the Euclidean distance between the desired initial position of the object (for picking) and its actual achieved position.

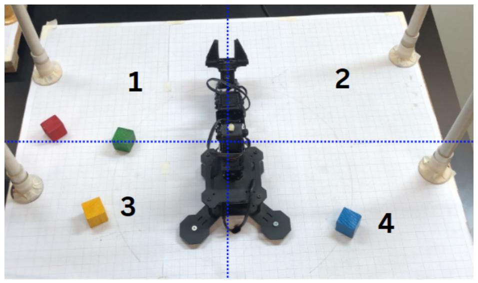
Testing Procedures:

The system's performance was evaluated through a series of controlled experiments. The tests involved using blocks of various colors placed at different locations within the camera's field of view. For each trial, the system attempted to identify, grasp, and place the block. The Euclidean distance between the desired final position and the actual placement was measured.

$$d = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2]}$$

Data Collection and Presentation: The data provided below is for 4 test runs for 4 different regions of the board(workspace).

No.	Object	Desired Position	Actual Position	Distance
1	3cm green cube	(-10,-10,3)	(-11,-11.5,3)	1.803
2	3cm green cube	(10,-10,3)	(8,-14,3)	4.472
3	3cm green cube	(-10,10,3)	(-9.5,-11,4)	1.5
4	3cm green cube	(10,10,3)	(-9.5,-10,3)	27.932



By analyzing the collected data, we can assess the overall effectiveness of the system and identify the limitations. The system works for the 3 regions but for the 4th specified region it is able to read the u,v coordinates but the image plane t camera plane mapping fails which gives wrong inverse kinematic solutions Other sources of error may include:

- Inaccuracies in camera calibration can lead to misinterpretations of the object's position and orientation in the real world.
- The system's ability to accurately identify and distinguish the target object from other elements in the scene can affect success rates.
- Limitations in the gripper's precision or the robot arm's movement can lead to imprecise grasping or placement of the object.
- Our initial perception part was done for a specific lighting condition. However when we were running the entire pipeline, the lights above the robot arm stopped working which led to inaccuracies in object detection to an extent.

Trade-offs and Considerations: Balancing the complexity of image processing algorithms with real-time performance requirements. Simpler algorithms may be faster but less accurate, while more sophisticated algorithms may provide higher accuracy but require more processing power.

The system's performance might be tailored to the specific testing environment and object types used. Further testing with a broader range of objects and environments will be necessary to determine its generalizability and robustness.

IV. Conclusion and Future Work

The entire process of the lab enabled us to grasp the underlying concepts for the movement of robot according to our wish. Working step by step, we now finally realize the importance of each one in making the overall pick and place task successful. The overall process can be looked at as the camera detecting the object in the image frame, which gets translated to camera frame by camera intrinsic parameters. The transformation matrix converts the camera frame information to robot frame/ world frame. This serves as the pick up position in the frame that our robot can understand. We then run inverse kinematics to determine the joint angles and then the robot is given instructions to attain those angles.

Future work in this regard can be performed to make the process more accurate with respect to the object position. Other tasks can include placing blocks on top of each other, which would require precision in order to make sure one block remains stable while placing the other. Currently it only picks up one block, it can be updated to pick all blocks of the same color. Another task that can be tried included syncing the movement of 2 robots to perform the same task.

V. Digital Material

https://github.com/ailiyaf/Robotics-perception-based-pick-and-place.git

References

[1] https://automaticaddison.com/how-to-assign-denavit-hartenberg-frames-to-robotic-arms/