Baxter: Box Sorting

CSCI 5551: Intelligent Robotics Systems Project Report
Amitabha Deb, Jordan Vastag, Supreet Gandamshetty, Vivekananthan Govindaraj
Advisor - Junaed Sattar
University of Minnesota, Twin Cities

Abstract

Robots have the potential to benefit us by increasing efficiency, reducing labour-intensive tasks, and allowing humans to focus on more complex and higher-value activities. Sorting objects at a large scale is an excellent example of this. In this project, we use Rethink Robotics' Baxter robot with ROS and OpenCV to identify and sort boxes. We use object detection with colour recognition and inverse kinematics to accurately move the boxes. We achieved our goal of sorting and stacking cubes of three different colours. The work can be further extended to objects of multiple shapes and sizes.

1 Introduction

1.1 Goal

We aim to have Baxter classify and pick up coloured-coded inventory boxes and place them in the corresponding stack.

1.2 Purpose

The purpose of this project is to gain hands-on experience with robots and ROS while applying course knowledge of inverse kinematics. Additionally, we hope this project could teach us about advanced practical applications such as sorting boxes in a warehouse.

1.3 Assumptions

We assume the boxes to be small coloured cubes. The boxes are placed on a tabletop which has a flat surface parallel to the floor and is of white background. We assume each cube has the same orientation. Further, we assume the cubes are within the range of motion of Baxter and each cube is either red, green, or blue.

2 Literature Review

The paper "Baxter Humanoid Robot Kinematics"[1] explains the kinematics that the Baxter consists of dual 7-degree-of-freedom (dof) arms, a vision system, and a safety system. The Baxter arm design is similar to traditional 7-dof robot arms, offering kinematic redundancy. It has 7 single-dof revolute joints, one more than the 6 Cartesian degrees of freedom (dof). This redundancy allows the arm to optimize performance while achieving general 6-dof Cartesian trajectories. The Baxter's arm stands out for its actuation system using Serial Elastic Actuators (SEAs), which provide a compliant nature for safe interactions and torque sensing capabilities for statics and dynamics control. These features contribute to Baxter's design as a coBot, designed to interact with humans safely.

Each arm of the Baxter robot has seven degrees of freedom (7-dof) represented by seven angle values for joint rotation. With fixed arm lengths and known joint angles, calculating the gripper's position and orientation becomes straightforward. The commonly used method involves utilizing Denavit-Hartenberg (DH) parameters[2], where each joint is characterized by four parameters in a DH table. These parameters are multiplied to determine the gripper's position and orientation relative to a reference coordinate frame. This process, known as forward kinematics, yields a single solution since the joint angles are fixed. Joint angle values can be accessed and monitored through the ROS tool "tf" or by accessing the joint status topic.

In the paper "Baxter Kinematic Modeling, Validation and Reconfigurable Representation"[3], the author employs the idea of Denavit-Hartenberg (DH) parameters in Baxter. These parameters consist of four variables: translation along the previous z-axis, rotation about the previous z-axis, translation along the new x-axis, and rotation about the new x-axis. They define the spatial relationship between two coordinate frames and are crucial for modelling manipulators. Using the DH framework, we can derive a transformation matrix that connects

the two frames through matrix multiplication. The

$$T_n^{n-1} = Rot_z(\theta) \cdot Trans_z(d) \cdot Trans_x(a) \cdot Rot_x(\alpha)$$

DH parameters rely on the specific frame assignments made on the manipulator. We adhere to the DH convention for the Baxter robot's right arm and present our frame assignment in the figure below. The numerical values for the link lengths were obtained from the official Baxter documentation provided by Rethink Robotics.

Link	d (m)	θ (rad)	a (m)	α (rad)
S0-S1	0.27035	θ_0	0.069	$-\pi/2$
S1-E0	0	$\theta_1 + \pi/2$	0	$\pi/2$
E0-E1	0.36435	θ_2	0.069	$-\pi/2$
E1-W0	0	θ_3	0	$\pi/2$
W0-W1	0.37429	θ_4	0.01	$-\pi/2$
W1-W2	0	θ_5	0	$\pi/2$
W2-EE	0.229525	θ_6	0	0

Figure 1: Baxter DH Parameters for Baxter's Right Arm

Computer vision plays a crucial role in enabling robots to perceive and interact with objects in their environment. Numerous research papers have been published in this field, addressing various aspects of computer vision. One such paper introduces a novel approach to computer vision that focuses on teaching a computer system to detect and classify colours effectively, catering to a wide range of applications[4]. The proposed detection algorithm takes advantage of camera capabilities and uses input data to precisely identify colours based on their RGB values. Through the utilisation of a function that iteratively adjusts the distance to find the best match, the algorithm adeptly defines colours within the RGB colour space, exhibiting exceptional accuracy. This innovative technique amalgamates the power of computer vision and colour analysis, opening new avenues for improved colour recognition and interpretation in diverse real-world scenarios.

Calibrating the Baxter robot camera is crucial in ensuring accurate perception and reliable performance. The camera calibration process involves determining the intrinsic and extrinsic parameters of the camera, which enable precise measurements and the correct interpretation of visual information. To begin the calibration process, it is essential to have a calibration target, such as different colour cubes with known dimensions. This target is placed within the robot's field of view; a picture is taken from the right limb camera of the Baxter. This image is used to extract the necessary calibration data.

Intrinsic calibration involves determining the camera's internal parameters, such as focal length, lens distortion, and principal point, by analysing the captured images, using openCV to estimate these parameters. Extrinsic calibration, on the other hand, focuses on determining the position and orientation of the camera relative to the robot's base or end effector. Once the calibration process is complete, the camera's parameters are stored, and the robot's perception system can use them for various tasks like object detection. Regular calibration checks are recommended to account for changes in the camera's characteristics due to wear and tear or environmental factors.

3 Approach

For our problem, we assumed the boxes to be 3cm x 3cm x 3cm blocks coded in red, blue and green. The blocks were 3d printed.

Next, we set up the Baxter workspace on our ROS melodic workspace and cloned all the required repositories for our project. We tried a few basic commands, such as untucking the arms and keyboard control for the joint angles and got ourselves familiar with how Baxter Robot can be controlled. Then we moved on to the inverse kinematics and Computer vision part of our project.

3.1 Baxter Inverse Kinematics

Initially, we tried computing joint angles with hard-coded coordinates. We manually moved the Baxter right arm end effector to the object location. Then we obtained the pose of the end effector from the topic. These coordinates were fed to the inverse kinematics solver service, which returns the required joint variables. The joint variables are then actuated using forward kinematics to allow Baxter to reach the object. After successfully picking up cubes through this approach, next we needed to incorporate computer vision to extract the position of the cubes from the camera.

3.2 Computer Vision

We began by writing a script for object detection of the cubes. We used OpenCV[5] to implement an algorithm that takes a target colour (with upper and

lower limits) as input and detects the target colour. A bounding box is created for the object in real time, and the coordinates of the centre of the box are extracted. Before moving on we tested it on household objects using our laptop webcam.

Next, we used the Baxter right-arm camera to calibrate the colour of our boxes. The upper and lower limit of different colours were found and we added those limits as arrays in our Python CV code.

Using the algorithm on the Baxter Right-Hand camera we were able to successfully extract the colour and coordinates of our target blocks. These coordinates were transformed to Baxter coordinates using a home position calibration method. The pixel coordinates and the Baxter coordinates for the home position as well as the difference in Baxter coordinates with respect to pixel coordinates were used to come up with the formula for the transformation.

Finally, these coordinates were fed to the inverse kinematics solver for Baxter to extract the joint angles to pick up the blocks.

Our motion plan for the Baxter right arm consists of the following steps:

- Go to the home position.
- Go to the block location to pick it up.
- Come back to the home position.
- Go to the drop position of the corresponding cube.
- Iterate these steps.

The z coordinate of the drop position is incremented by 3 cm for each stack when a block is dropped on it.

The final algorithm loops until all the cubes are sorted into their respectively coloured stacks.

3.3 Simulations

We attempted to use ROS and Gazebo for our simulation. We ran into numerous problems while attempting to simulate Baxter. The primary source of our issues was that all code and documentation associated with Baxter is no longer maintained. We were able to get Baxter and a table into Gazebo with the help of a 2021 post titled "The robots are coming (virtually)"[6]. We were able to run sample actions in the simulation but we had trouble getting our code to work with Gazebo. The catkin_make for our project workspace failed for the simulation package and so we discarded it.

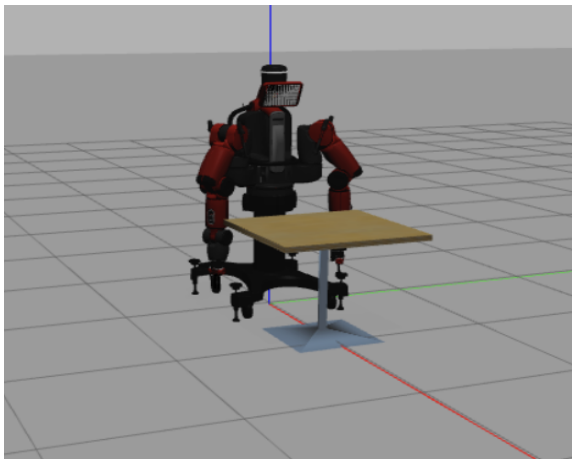

Figure 2: Image of Baxter in Gazebo

4 Results

The initial results we presented in class were the successful implementation of computer vision to extract the coordinates and colour of objects using a laptop camera and the implementation of inverse kinematics using hard-coded coordinates. The next step was to integrate the two so that the robot was able to pick any block present within its vision and arm range.

The vision algorithm when applied to Baxter's right arm camera was successfully detecting our blocks.

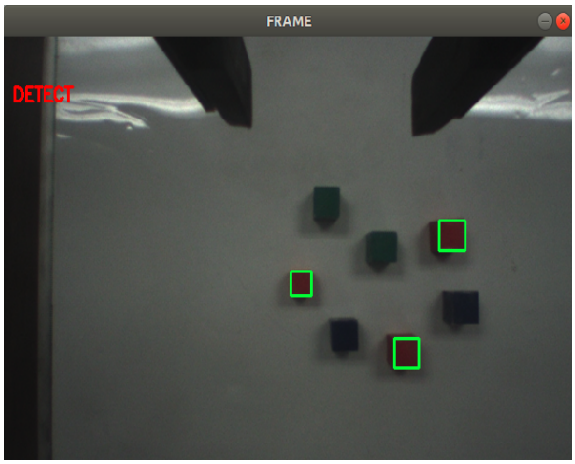

Figure 3: Red blocks detected by Baxter's right-hand camera

After integrating the CV algorithm in Baxter, we needed to transform the pixel coordinates to Baxter coordinates. The calibration equation was calculated using the home position. We tested out the algorithm for one block and after successful implementation, we added more blocks of red, blue and green colour.

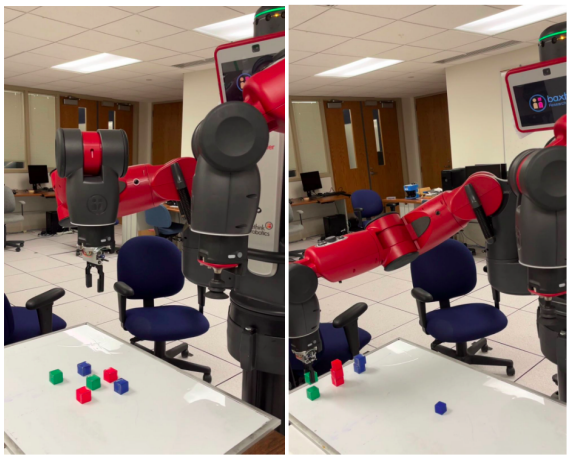

Figure 4: Initial and final stages of Baxter sorting boxes

Our algorithm successfully used Baxter to sort and stack red, green, and blue cubes. Boxes can be added to the table during the runtime which makes our code more robust. The time taken by the Robot to successfully sort 8 boxes was 3 minutes.

5 Conclusion

We successfully applied course knowledge of ROS and inverse kinematics to our project. We used ROS, Gazebo, OpenCV, and Baxter to sort and stack cubes of three different colours. We achieved our goal, but it required human intervention at times. The major roadblock we faced was trying to simulate Baxter in Gazebo. A successful simulation would have made the development process much easier because many other groups were using Baxter. It was challenging to get time in the lab with the robot.

Our algorithm has some limitations. We are using a fixed orientation for all the blocks for now. So if a block is not aligned with the gripper the Baxter won't be able to grab the blocks. Also, we have a predefined location of the desired stacks which can be updated in future with flexible marks on the table. Our code also needs to have more tuning to handle errors and failures

In the future, we could apply our code and knowledge to use a robot to sort packages in a warehouse according to their labels.

6 Launch Instructions

Place the cubes parallel to Baxter on a table in front of it. The cubes should be within the range of motion of its right arm.

The following steps need to be done on a Ubuntu system with ROS melodic/noetic and python3 installed in it:

- Create a catkin workspace for the project.(baxter_ws)
- Extract the src folder to the Catkin workspace.
- On the baxter.sh file set the ip of Baxter, the system and the ROS version
- Navigate to baxter_ws/src/baxter_project/scripts and enter the command: python baxter_sort.py

Repositories Used - Baxter SDK, Baxter pykdl
Demo Video - Youtube

References

[1] R.L. Williams II. Baxter humanoid robot kinematics, 2017.

[2] Abdullah Hayat, Rajeevlochan Chittawadigi, Arun Udai, and Subir Saha. Identification of denavit-hartenberg parameters of an industrial robot. pages 1–6, 07 2013.

[3] Lucas Silva, Tennakoon Tennakoon, Mairon Marques, and Ana Djuric. Baxter kinematic modeling, validation and reconfigurable representation. 04 2016.

[4] Junfeng Jing, Shenjuan Liu, Gang Wang, Weichuan Zhang, and Changming Sun. Recent advances on image edge detection: A comprehensive review. *Neurocomputing*, 503:259–271, 2022.

[5] OpenCV. Opencv documentation, 2022.

[6] Peter. The robots are coming (virtually). 2021.