CSCI 5551
Project Baxter - Sort By Number
Final Report

David Hwang - hwan0259@umn.edu
Shilpa deshpande - deshp143@umn.edu
Chance Foster - foste930@umn.edu

1. Abstract

The goal of this project is to make the BAXTER robot look at a digit(1-3) and sort the blocks by that number.

2. Introduction

a. Goals

1. Give an image to Baxter

2. From the image classify the digit (ranging from digits 1-3)

3. Based on the digit identified in step#2, Baxter will move that many blocks from the left side of the table to the right side of the table.

b. Purpose

This project aimed to explore and gain experience in using ROS, Baxter and computer vision.

c. Assumptions

Baxter, the table, and the blocks are in a set static position. Baxter knows the location of the blocks.

d. Related Work

There are many examples of projects where the Baxter robot is used to perform collaborative tasks. Some of these projects include the use of a vision system or machine learning algorithms to improve the performance of the robot.

For example [8], The robustness and accuracy of the tracking method was demonstrated through the real-time tracking experiment of the Baxter robot. The tracking motion of the robot is realized based on a specific robot or relying on an expensive movement acquisition system. It has the problems of complex control procedures, lack of real-time performance, and difficulty in achieving secondary development. A robot real-time tracking control method was proposed based on the control principle of differential inverse kinematics, which fuses the position and joint angle information of the robot's actuators to realize the real-time estimation of the user's movement during the tracking process. The motion

coordinates of each joint of the robot are calculated and the coordinate conversion between man and machine is realized with the combination of the Kinect sensor and the robot operating system.

Another example [9], Eight human-robot exercise games for the Baxter Research Robot were developed, six of which involve physical human-robot contact. After extensive iteration, these games were tested in an exploratory user study including 20 younger adult and 20 older adult users.

3 Implementation

3.1 Baxter Background

Baxter is an industrial robot that was created by Rethink Robotics. It has a monitor that serves as its head with the ability to show an animated expression and has two arms, one equipped with a gripper and the other a suction cup. Baxter has multiple sensors, the most relevant being a camera on each of Baxter's arms and a camera on its head. Baxter runs on Robot Operating System (ROS).

Figure 1: Baxter's arm with the gripper

3.3 Environment

Baxter is a set distance away from a table where the arm can hover over the table. On top of the table are three blocks and Baxter is able to pick up the blocks and move them across the table. It

is assumed that the location of Baxter, the table, and blocks do not change. It is also assumed that the blocks initially start on the left side of the table and Baxter knows the location of all three blocks.

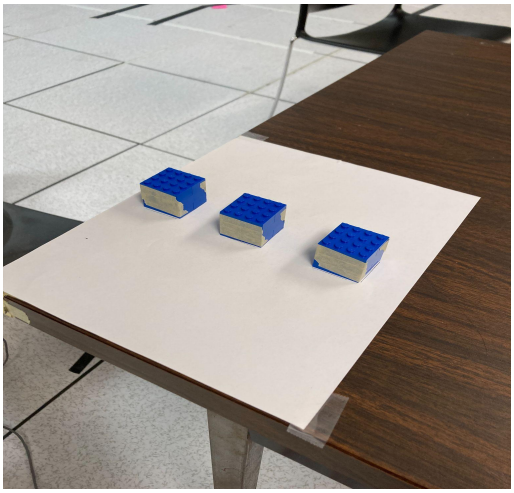

Figure 2: Blocks on table in starting location

3.4 Approach

For the project the functionality was split up into three parts.

1. **Feeding an Image to Baxter:** use Baxter's head camera in order to feed it an image of a digit ranging from one to three.
2. **Digit Classification:** preprocess and convert the image into data and predict the digit from the data with machine learning
3. **Movement of Blocks:** according to the digit from the prediction, Baxter moves that many blocks from the left side of the table to the right side of the table.

In the next sections more details on the implementation of the parts will be given. Altogether these parts are then combined to fit into a script in ROS that can be run.

3.4 Feeding an Image to Baxter

The Baxter robot has 3 cameras with 2 located at the palms of Baxter's left and right arms, and one located on Baxter's head display. A limitation of Baxter's cameras is that only 2 cameras can be turned on at a time, and by default, the head camera is initially shut off such that the left and right hand camera are initially on. Given that the head camera was the desired camera for capturing images, a python script would be necessary for manipulating the cameras so that the head camera could be turned on for capturing number images.

The python script that was used to accomplish this made use of the baxter_interface camera controller that essentially allowed the programmer to choose a camera topic (‘left_hand_camera’, ‘head_camera’, or ‘right_hand_camera’) and turn the respective camera off or on. Much of the code for this portion of the project was inspired by a tutorial posted from Purdue on manipulating Baxter’s cameras and can be found in the References section (Source [2]).

Now that a script could be run to turn Baxter’s head camera on, the next step was to write a script that captured images through the head camera, and saved these images to a file that could be opened and read in from the digit recognizer. To facilitate this, a python library would be necessary for converting the image captured from Baxter to a processable image such as a .png or .jpg file. After doing some research, code was found on the ROS wiki (Source [3]) that makes use of the CV Bridge library, which is a library designed for converting ROS images (images captured from Baxter) to CV images, which are in turn writable as .png/.jpg. By making use of this code, as well as adding additional functionality to the subscribed topic, feeding images to Baxter and saving these images to a file for digit recognition had now been completely automated.

3.5 Digit recognizer using CNN

Dataset:

MNIST (“Modified National Institute of Standards and Technology”) is the standard dataset of computer vision. Since its release in 1999, this classic dataset of handwritten images has served as the basis for benchmarking classification algorithms. As new machine learning techniques emerge, MNIST remains a reliable resource for researchers and learners alike.

Data Processing:

The data set contains 60,000 training images and 10000 testing images. Since we are only working on digits 1-3, We removed the labels 0 and 4 to 9. The new dataset was split into training, testing and validation sets.

Building the Model:

We used pytorch to build the model. The model used is a 9-layer Residual Neural Network (ResNet), it uses skip connections, or shortcuts to jump. Resnets are implemented with double layer skips with non-linearities(ReLu).

Hyperparameter	Value
Number of Epochs	10
Batch Size	100
Learning Rate	0.001/0.005

Results:

Training accuracy	Test accuracy
97.05 %	98.05 %

3.6 Movement of blocks

In order to move the blocks, Baxter is equipped with a gripper on the end of one of its arms. With 7 degrees of freedom it is able to move its arm to a block, have its gripper pick it up, and then drop the block.

There are multiple methods in order to interact with Baxter's arms. Some of the options include using moveit in order to use Kinematics, controlling the arms with a controller, or manually interacting with the arms.

The decision came down to manually interacting with Baxter because of the ease of use with its arms and also the convenient tools that are provided in order to record the arm movements and play back the recording [7]. Additionally, Baxter is not guaranteed to be in the same spot so if there is a slight adjustment that is needed then that can be quickly done by manually interacting with Baxter. The recording script allows a user to start the script, where the user then does the arm movements and afterwards when the recording is finished it will save a txt file of all the recording of the joint movements. The playback script takes in a txt file of the joint movements and then moves Baxter's joints in accordance to the file.

Since our program can classify a digit as one, two, or three there are three separate recordings for Baxter moving one block, two blocks, or three blocks.

In manually moving the arm there is a button on the side of the gripper to open or close the gripper. When recording for moving one block, Baxter's arms initially start at its chest, and then the arm is moved to a block on the left side of the table with the gripper facing the top of the block. The button to close the gripper is then pushed, and the arm is now moved to the other side of the table carrying along the block. The block is placed down on the table where the gripper is released and afterwards the arms are manipulated to the initial position. For two or three blocks the same process is repeated.

Figure 3: Closeup of Baxter's gripper

4 Testing

We elected not to use simulation tactics given that a majority of the team was able to make it to the lab in person and the group member that was unable to make it could freely work on their portion of the project without needing to use the actual robot. The team tested their python scripts by connecting to the physical robot and running ROS packages containing these scripts to determine if Baxter was behaving as expected. After a great deal of trial and error, code/scripts could be polished to reflect correct behavior/results over time.

5 Results

Baxter was able to receive an image through the head camera and we were also able to classify the digit from the image. There were some complications with the prediction as although the training accuracy was good there were still misclassifications with the image. After classifying

the digit, Baxter was able to receive the prediction and was able to use its arm to move the blocks from one side to the other with the gripper corresponding to the prediction.

6 Conclusion

a. Achievements

As of writing this, the team has successfully managed to download and run ROS in harmony with Baxter's software development kit, record and playback joint movement to simulate correct block removal, feed images to Baxter and write these images as .png files, and correctly identify numbers from images. In addition, a majority of these components have been successfully integrated and automated into one large script that can be run all in one place.

b. Challenges

Initially, the team faced challenges regarding setup and properly connecting to Baxter. With Baxter not receiving updates for the latest ROS version, and previous ROS versions only being available for older machines, it was difficult to find a medium in which modern laptops could talk to Baxter. To combat this, we elected to use a virtual box that can run images of older software models, and by doing so, we were able to install a version of ROS that was supported by Baxter's software development kit. With this obstacle out of the way, the next challenge came in the form of precisely recording joint movement that allowed Baxter to seamlessly pick up and place blocks without knocking them over or dropping them. This required patience and recording many times in order to guarantee precision so that Baxter correctly moved the blocks more times than it did not. The last challenge the team has not quite managed to overcome has been integrating digit recognition with the rest of the project. When trying to place the code for digit recognition into our script, we have observed errors related to pytorch that we are still trying to understand and fix.

c. Limitations

Limitations for this project come in various forms. For one, synchronizing the team's schedules has required sacrifices from each member and is not always the most efficient when time is a key factor for a project such as this one. Additionally, with Baxter being unable to support the latest ROS version, limitations are placed on the python scripts since they must be written in python 2. Due to this limitation, enhancements in machine learning (such as abstraction used in digit recognition) must be downgraded from python 3 to python 2, and

this limits the capabilities of the program. Furthermore, running programs from virtual machines has caused setbacks for the team since virtual machines have limited amounts of RAM, and therefore less processing power.

7 Run

To run the project the user needs to install Ubuntu 16.04 LTS, ROS kinetic, and create a workspace to install the Baxter SDK. The user can then drop the package for sort by numbers (called package) into the workspace which after the setup script(baxter.sh) has been configured with the user's ip address and baxter's ip, the user is able to connect to baxter and run our package with *rosrun project baxter_pickup.py*. Full instructions will be included in the README with the sort by numbers package.

References

[1] Deep Residual Learning for Image Recognition
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun
https://arxiv.org/abs/1512.03385v1

[2] Using Baxter Cameras
https://web.ics.purdue.edu/~rvoyles/Classes/ROSprogramming/Using_Baxter_Cameras.pdf
(This source was key in writing the code for manipulating Baxter's cameras and upon observation of our script (baxter_pickup.py), you will observe the inspiration drawn from it)

[3] ROS Python Save Snapshot from Camera
https://answers.ros.org/question/210294/ros-python-save-snapshot-from-camera/
(Similar to the source above, by observing our code (baxter_pickup.py) you will notice much of the code found here was used in the implementation of using python code to capture images and convert them to .png's for digit recognition)

[4] Python rospy.wait_for_messages Examples
https://www.programcreek.com/python/example/95407/rospy.wait_for_message
(Not used for the scope of this paper but can be observed in source code for baxter_pickup.py)

[5] Unsubscribing from ROS Topic
https://devdreamz.com/question/545996-unsubscribing-from-ros-topic-python
(Not used for the scope of this paper but can be observed in source code for baxter_pickup.py)

[6] Part 2: 7 Simple Steps to Create and Build Your First ROS Package
Arsalan Anwar
https://medium.com/swlh/7-simple-steps-to-create-and-build-our-first-ros-package-7e3080d36faa
(Not used for the scope of this paper but was used to create our package called "project")

[7] Using Baxter Joint recording and playback
https://sdk.rethinkrobotics.com/wiki/Joint_Trajectory_Playback_Example

[8] 1. Li S, Zhang X, Yang J, et al. Real-time motion tracking of cognitive Baxter robot based on differential inverse kinematics. International Journal of Advanced Robotic Systems. May 2021. doi:10.1177/17298814211024052