“Accio”

A peek into multi-robot collaboration and human - robot interaction

University of Minnesota - Twin Cities

Youya Xia
Yingxin Wei
Xiaofei Chen
Ziqian Qiu

Github link: https://github.com/QueenieOhYeah/Accio

Abstract

In this report we demonstrate the methodology of multi-robot collaboration, human-robot interaction and discussing the challenges we faced in this project and the insights gained from them, to be applied to future work in this domain.

Introduction

Accio is the incantation of summoning charm originated from our favourite book, 'Harry Potter'. Witches uttering the spell of 'Accio Firebolt' would call the required broomstick to fly into their hands. Inspired by this magic, we come up with the idea of request things orally and have them delivered to us by robots. The job is designed to be accomplished as the result of collaboration of two robots, Turtlebot and Baxter.

Firstly, Turtle is told what is wanted among the 6 classes chosen. Then it goes to Baxter, and sends it the corresponding index of the desired class. After this, the deep neural network is triggered to train images and the resulting model is adapted to detect different objects appeared on the table. The process returns the coordinates of the desired object. Then Baxter grabs it by implementing inverse kinematics and puts it on Turtlebot. Turtlebot will wait until it gets the object, and then takes the object back to the commander.

To accomplish this task, we divided it into five segments, which are speech recognition, object detection, baxter manipulation, turtlebot motion as well as communication issues.

Methodology

The hardware structure of the multi-robot is illustrated in *Figure 1*. Baxter and Turtlebot are the processors that do the calculations and coordination. Turtlebot takes in the vocal inputs from microphone and passes the outputs to Baxter via wireless transceiver. Baxter receives the data transmitted from Turtle as an input and gets the raw image from the camera on the right arm of the Baxter. Then it manipulates the mechanical arms. When it is done, Baxter passes information to Turtlebot through wireless transceiver again to inform it. Turtlebot then controls its motion module, moving back to commander.

Figure 2 reveals the software interactions within the multi-robot model.

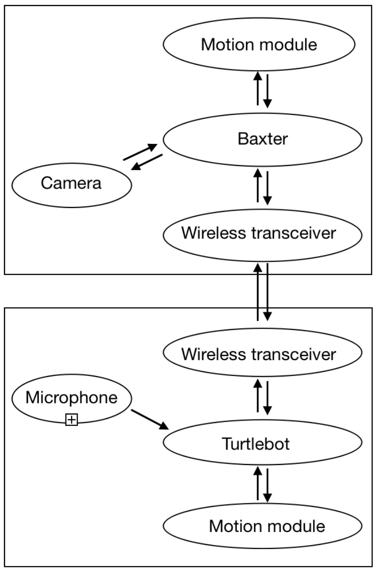


Figure 1. Hardware Structure of Robots

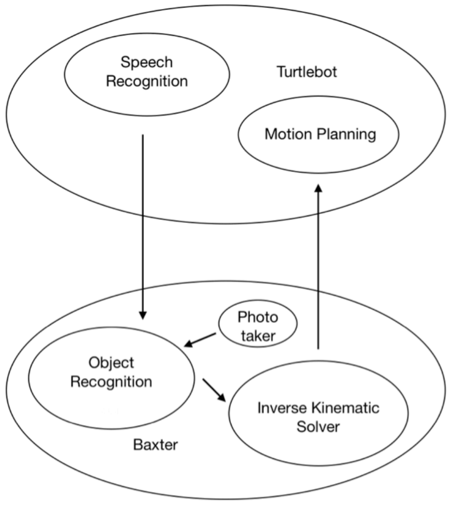


Figure 2. Software interaction

Github link: https://github.com/QueenieOhYeah/Accio

A. Speech Recognition

For speech recognition part, we implement key words capture by speech recognition API [2] provided by Google. Google Cloud is an AI & Machine Learning product. Google Cloud Speech API [2] with Streaming API[2], opens a recording stream, continuously collects data from audio, and matches the key words we defined with the data. In this project, we only recognize the specific word instead of semantic analysis. For instance, once we say something like “Please take the xx for me”, the program will be able to recognize the whole command and print “Task xx”. After that, the program will save this task to local disk on laptop for future read. For other irrelevant command like “Please take the course CSCI5551 for me”, the program will print “This task is too difficult”.

B. Object Recognition

For object detection, we use the pre-trained neural network Faster R-CNN[1]. In order to improve the precision rate of the Faster R-CNN. The following is the working process of Faster R-CNN. First, it will go through a convolutional neural network (which offers a set of convolutional feature maps on the last layer) Then, it will pass a sliding window over cnn’s feature map-output k potential bounding boxes (anchor box) and give each box a score (anchor box is designed to find the best box that can fit the object tightly)(a value p* is computed to estimate how much an ancho box is overlapping with a ground-truth box. Finally, the n*n feature maps (e.g. 3*3) are fed into a smaller network which has two purposes classification and regression and the output regressor will determine the bounding box. Besides, the classification process is accomplished by calculating the probability that the bounding box really contains a object(confidence) [3].

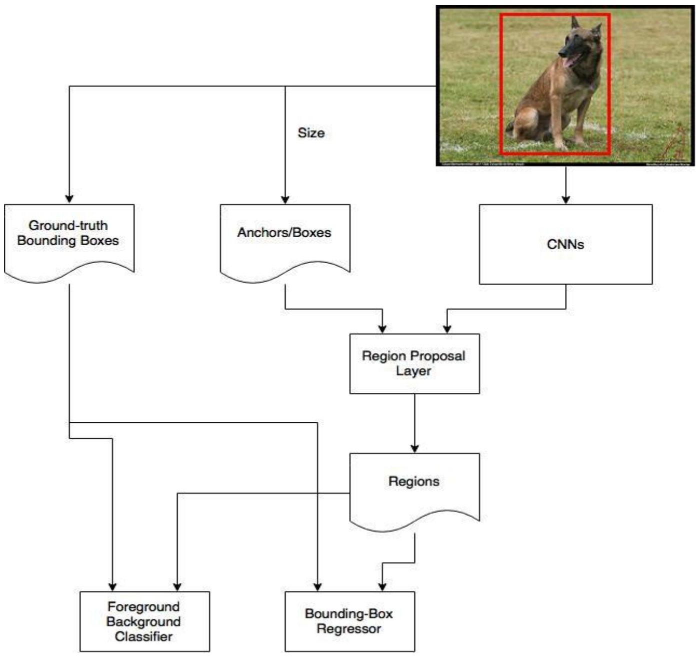


Figure 3. Fast-RCNN Structure

In addition, we self labeled 2000 images for data training using 6 classes (i.e. banana, toy, bottle, cell phone, apple, orange) and test our training results using Baxter.

Github link: https://github.com/QueenieOhYeah/Accio

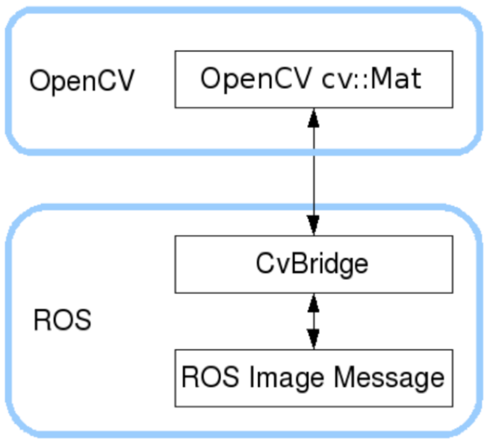
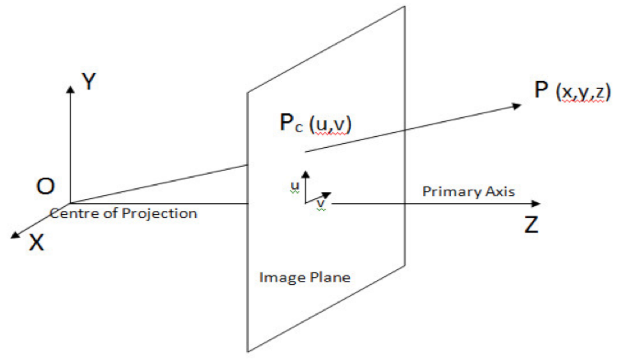


Figure 4. The principle of converting ROS Image to cv image to use our training model

Since object detection and robot manipulation need to run under different environments, images transmission between them poses a problem. Our method for solving it is to save the images transformated from the raw images through CVbridge in local disk. Therefore, process of object detection is able to get the required inputs locally.

C. Robot Manipulation

For Baxter manipulation part, since we always set the right arm of Baxter pointing vertically down at a fixed position, we only need to care about its xy plane. After getting the xy center coordinates of the object on CV image, we can compute the center position of the object in Baxter workspace coordinate using the conversion equation provided on Baxter's official website. We have known all the parameters for this equation. For example, distance from the table is fixed, and we use Infrared sensor in the gripper to get it. We choose 960x600 to be camera resolution, so we know the pixel size. After getting the object position, we use MoveIt! to do inverse kinematics to pick the object and give it to TurtleBot.



$$B = (Pp - Cp) * cc * d + Bp + Go$$

where:

- B = Baxter coordinates
- Pp = pixel coordinates
- Cp = centre pixel coordinates
- Bp = Baxter pose
- Go = gripper offset
- cc = camera calibration factor
- d = distance from table

Figure 5. Coordination Conversion

D. Communication

Data transmission is required between Baxter and Turtlebot. Turtlebot and Baxter fails to connect to a LAN because speech recognition done by Turtlebot requires Internet connection while Baxter is not authorized to do so. An extra 915MHz module, Readytosky 3DR Radio Wireless Telemetry Kit is chosen to accomplish the communication between them. The kit adapts transmission mode of Standard TTL UART. Every object extracted from the speech recognition has a corresponding index to be sent.

Github link: https://github.com/QueenieOhYeah/Accio

E. TurtleBot Motion

TurtleBot is used to deliver the object from Baxter to commander. Since TurtleBot is not our focus in this project, we set a fixed start location close to commander, and a fixed end position close to Baxter to TurtleBot. In this case, we can map a straightforward route for TurtleBot without doing obstacle avoidance. The distance of the route is calculated in terms of frequencies and counts.

Experiment & Results

For the reason that the success rate of our entire project mainly builds on the accuracy of speech recognition and object detection, we set up experiments on those two parts respectively before we went though the whole chain of processes.

A. Experiment 1: Speech Recognition

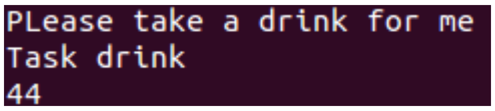
Experiment: To test the speech recognition part, we continuously talk in sentences through the microphone, and print out what it gets on the screen. In the speech recognition part, we have conducted 20 tests in order to test the accuracy of speech recognition.

Results: The precision of this part is 93%, which is satisfying and relatively accurate.

Total number of experiments	*successfully print out the whole sentence*	*successfully print out the key world*	*percentage*
30	*20*	*28*	*93%*

Table 1. Accuracy table for speech recognition

Issues: Recognition may fail when pronunciation is ambiguous, which is fair. Another potential issue is that the Google Cloud Speech API does not actually do the semantic recognition, which is to say, the methodology we use for speech recognition only detects if the keywords of the classes are within the sentence we speak without concerning its actually meaning. As a result, if we say “Don’t take the bottle for me”. Under this circumstance, google API still recognize our sentences as successfully extracting the Task bottle, and Turtlebot will go for it although the sentence actually means not to.



```
PLease take a drink for me
Task drink
44
```

Figure 6. Successful Demo of Speech Recognition

Possible solutions: Another API developed by Google, namely Dialogue flow, deals with the meanings of sentences. We tried to use that and it worked. But we want to keep it simple at this stage.

B. Experiment 2: Object Detection

Experiment: We want to test how well Baxter’s camera can detect the object on the table. First, we put different objects including the objects in our training model, as well as other random objects out of our training model onto the table. We have conducted 20 experiments in this part: run the object detect model we have trained 20 times.

Github link: https://github.com/QueenieOhYeah/Accio

Results: In general, it successfully detected 97% of the wanted objects in sequences of images and under the major circumstances, the confidence level of the detected objects are above 97%. Due to the fact that the Tensorflow is using the length and width of the bounding boxes to train each object, sometimes it provides a high confidence level for two different object with the similar length and width. Therefore, it is not accurate at some time.

Object	*total number of experiments*	*total number of success*	*percentage*
Banana	*20*	*19*	*95*
Phone	*20*	*18*	*90*
Empty bottle	*20*	*19*	*95*

Table 2. The accuracy table for the object detection

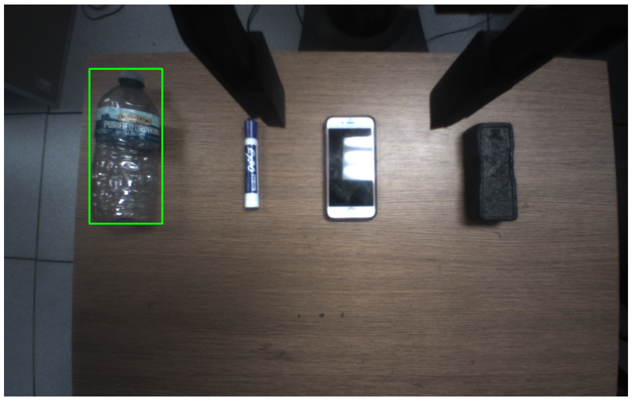


Figure 7. Successful Demo of Object Detection

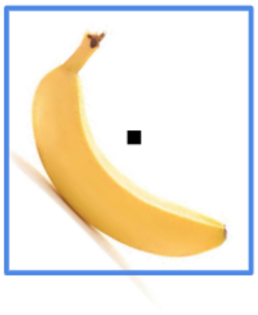


Figure 8. Bias of Estimated Center

Issues: Since we put these objects randomly on the table, sometimes they are blocked by the gripper. (Shown in *Figure 7*) In this case, they can not be detected. However, if we put them in the upper left of the picture on the table, the percentage that they can be detected is really high. Another issue is that tensorflow does not capture the orientation of the objects, thereby causing bias between the estimated center and the physical center of the objects (as illustrated in *Figure 8*). Depending on the object, the bias can be quite significant, causing problems for grabbing them.

Possible solutions: Improve tensorflow or with extra help of OpenCV.

C. Experiment 3: Combination of Object Detection and Baxter Manipulation

Experiment: we want to test how good is the Inverse Kinematics on Baxter for grabbing the empty bottle on the table in front of it. First assume our orientation is the same as Baxter's, facing the same direction. Bottle's orientation is always 90 degrees to the sliding direction of the gripper.

Github link: https://github.com/QueenieOhYeah/Accio

total number of experiments	*total number of success*	*percentage*	*object location*
20	*15*	*75%*	*Lower Left*
20	*19*	*95%*	*Upper Left*
20	*4*	*20%*	*Lower Right*
20	*16*	*80%*	*Upper Right*

Table 3. The accuracy level of Inverse kinematics based on object location

Results: We found that the percentage of success mainly depends on where we put the bottle on the table. If the bottle is placed in the upper left of the picture taken by the camera on the gripper, it's 95% to succeed. However, if it is placed in the lower right of the picture, the percentage is really low. The main reason for that is: it's hard for Baxter to get solution of inverse kinematics calculation in some cases.

D. *The whole process of multi-robot collaboration*

Experiment: Due to the configuration of the Baxter's gripper, some objects we chose before are too large or too heavy to be picked up. Baxter can only pick up the detected empty bottle successfully. Therefore, according to the process we described above ,we have tested the whole process ten times on the bottle class.

Results: Six times out of ten was successfully done. Therefore, the successful rate was around 60%. The process might involve a single issue or combinations of them leading to the failure of the experiments. Those issues were summarized as the followings excluding the issues found on speech recognition and object detection.

Issues on Turtlebot: If we let TurtleBot draw squares on the floor, we can see it will quickly starts to drift away from the starting point. This is the difference between a computer and a robot. A robot will move roughly 1 meter if you ask it to move forward 1 meter. Knowing the localization of the robot is one of the robotic challenges, especially in indoor environments where GPS isn't reliable or accurate enough. If we tell the robot to go forward at 0.2 m/s, then we can program it so that it goes forward one meter by linear.x = 0.2 m/s for 5 seconds. This is very inaccurate due to slippage, imperfect calibration, and other factors.

Issues on Communication: TurtleBot is controlled by the laptop that is connected to it, and this laptop has the wireless transmitter on it. On the other hand, Baxter is controlled by another laptop that is connected to it as well, and this laptop has the wireless receiver. Therefore, there is communication between TurtleBot and its laptop through USB. At the same time, we need TurtleBot to talk to Baxter through wireless modules. Due to this unknown USB problem and based on our tests, these two kinds of communication may conflict with each other and make our task fail sometimes.

Issues on Robot Manipulation: If the coordinations received from object detection are inaccurate, Baxter didn't know if it is grabbing air problem. A possible solution to it can be measuring the distance from the object again using infrared sensor when the gripper approaches to the object. Theoretically, we know the origin distance from the gripper to table. We can track on the moving distance, detect the distance from object and we have estimated height of the target object. The summation of those should be approximate to the original distance from table. If there's a significant difference between them, we can safely assume baxter is grabbing air. (Illustrated in *Figure 9*)

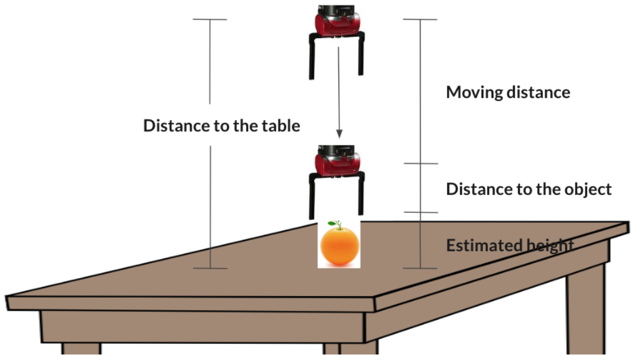


Figure 9. Illustration on providing feedback

Conclusion and future work

In a nutshell, we achieved the initial goal of our project. However, there are several potential future work we can improve about our project.

A. Person identification

For now, in order to simplify our problem, we just let the robot move from one position to another. However, we are thinking if we can build a person detector for TurtleBot, so that it can detect the location of the person who makes the command and carries the object to the location of the commander.

B. Motion planning

Under most circumstances, robot needs to accomplish tasks in open and challenging environment. Therefore, one potential future work we can accomplish is to use Kalman filter to optimize the routes the TurtleBot chosen to approach the Baxter and back to the commander.

C. Baxter manipulation

Since in the initial stage of our project, we did not consider the configuration of the Baxter's gripper, so for most of our initial chosen objects, Baxter cannot grab them except the empty bottle. Therefore, one of the future work we can accomplish is choose smaller objects for training our models such that they can fit into the gripper of the Baxter

D. Real-time

For now, the running time of our code for this project is about 1 fps ,which is slow for real-time implementation. Thus, we can try to improve the running time of our algorithm. For example, we can convert our code from Python to C++ to improve running time.

Github link: https://github.com/QueenieOhYeah/Accio

References

[1]:Object detection model:
https://github.com/tensorflow/models/tree/master/research/object_detection

[2]:google speech to text API:
https://github.com/GoogleCloudPlatform/python-docs-samples/tree/master/speech/cloud-client

[3]:Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.

Instructions on running code:
The specific instructions on running our code is in the README of our github link!

Github link: https://github.com/QueenieOhYeah/Accio