Institute for Cognitive Systems

Project Practice RoboCup@Home

Bavarian Robot Work

Pick and deliver objects to the professor

Authors:
B.Sc. Chao Dong,
B.Sc. Ji Chen,
B.Sc. Weiqi Luo

Supervisors:
Prof. Dr. Gordon Cheng
Dr. Karinne Ramirez-Amaro
Dr. Emmanuel Dean
Dr. Pablo Lanillos
M.Sc. Roger Guadarram
M.Sc. Constantin Uhde

July 25, 2018

Contents

1 Introduction

1.1 Description of the project

In this project, robot will pick an object from this operator and deliver it to Professor Cheng. According to this goal, the task is divided in 4 parts.

Perception

The part of perception gives the robot the abilities of recognize the gesture of operator, the face of professor and 3D coordinates of an object.

Manipulation

The part of manipulation allows the robot using his arm and gripper to grasp and place the object.

Navigation

The part of navigation allows the robot to localize and move itself to a specific point

Speech detection

The part of speech recognition gives the robot the ability of recognizing the speech information and adapt it to its main logic.

1.2 Workload

Table 1: The Workload

Group member	Work
Chao Dong	Manipulation
Ji Chen	System Integration Speech Recognition Navigation
Weiqi Luo	Gesture Detection Face Detection Object Detection

2 System Design

2.1 Connecting of nodes

The point of system integration is how nodes communicate with each other properly in order to solve the problem all together. The nodes and its communication is illustrated in Figure 1

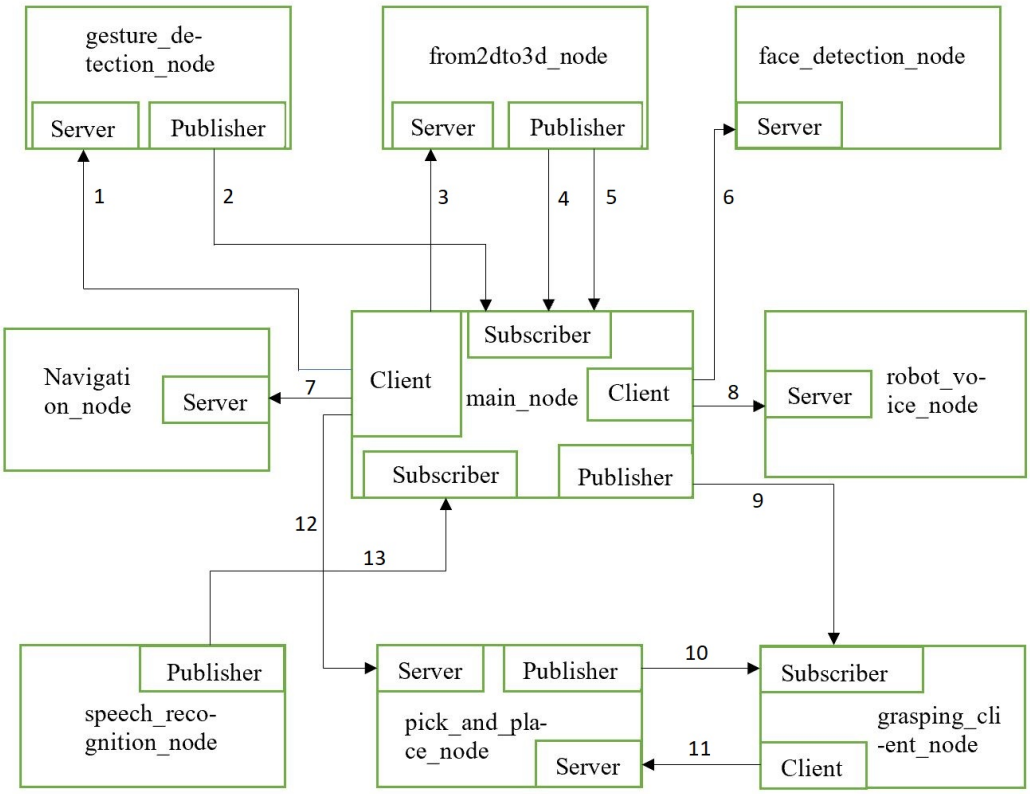

Figure 1: Syetem integration

The node-descriptions are listed in Table 2

Table 2: Nodes and their usage

Node	Usage
`main_node`	Organizes and manages the task and nodes using Smach and roslaunch API
`speech_recognition_node`	Get the voice Information and convert it to text
`navigation_node`	Provides services to let the robot do localization and navigation
`from2dto3d_node`	Get the camera information of desired object and convert it to 3D coordinates of this object
`Gesture_detection_node`	Detect the Gesture to make sure someone stands before the robot and trying to give it something
`face_detection_node`	Distinguish professors face and others faces
`robot_voice_node`	Let the robot to speak something
`pick_and_place_node`	Planning the trajectory of robots arm in order to let it pick the desired object

We use pair of server and client and publisher and subscriber to let the nodes communicate with each other. According to the design of the system, we tried to let a sub-node dont communicate with the other sub-nodes, that makes the main node gains a fully control of other nodes. It is easy for debugging and maintaining. The main node is constructed as a state machine which makes different nodes work in different time. The ros topics, and its usage among nodes are listed in Table 3

Table 3: Connecting of nodes

Number	Node	Usage
1	`/detect_gesture`	Start the gesture detection
2	`/process_done`	Provides the information if gesture detection is finished
3	`/desired_object`	Call the service to get 3D coordinates of desired object
4	`/point3D`	Provides the information of 3D coordinates of desired object

Number	Node	Usage
5	`/process_done`	Provides the information if the 3D coordinates detector has detected a proper 3D point of desired object
6	`/detect_face`	Call the service of face detection
7	`/move_base`	Call the service of navigation
8	`/say_something`	Call the service to let TIAGo say
9	`/point3D_from_main`	Provides the information of 3D coordinates of desired object
10	`/pick_process_ready`	Provides the information if the robot is ready to pick
11	`/move_arm`	Call the service of grasping the desired object
12	`/pick_gui`	Call the service to make the robot get ready to grasp
13	`/recognizer/output`	Provides the information of speech recognition

2.2 Dynamic management of nodes in main node

During practices we found out, if we run some nodes such as Yolo detector or face detector, the disk usage will instantly over 1 GB. Since they strongly occupy the computing resources, if they cant be closed after its usage, other nodes have to use less computing resources for running. That significantly decreases the efficiency of the program and may causes bugs. Figure 2 shows the GPU disk-usage before and after we execute the yolo, gesture detection and face detection node.

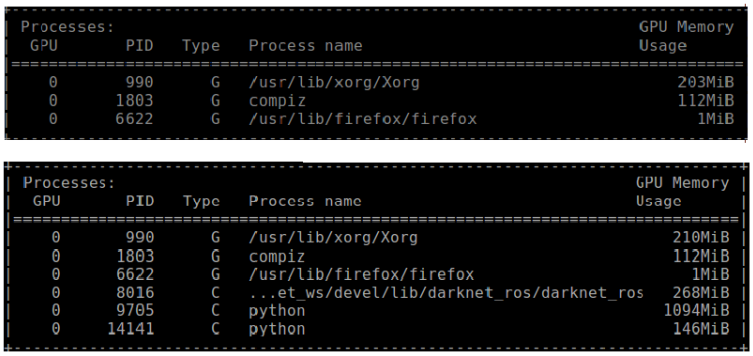

Figure 2: GPU disk usage, Top: before launching Yolo, face detection and gesture detection, Bottom: after launching perception nodes

In our system integrations, we used roslaunch python API to manage nodes dynamically. Using this API, we can start a node or a launch file at the time which its needed and shutdown them after it has been used. For example, before we start the object detection, we start the launch file of Yolo, after we get the 3D-Coordinates of the desired object, we shut down this launch file and release.0 computing resources.

3 Nodes Description

3.1 State machine by Smach — main node

Since the project can be described as transitions of finite states (a state machine). We implemented Smach to plan and construct our state machine. Smach allows us to maintain and debug a large and complex state machine. Figure 3 is the structure of state machine for this project. In structure of Smach, each state is coded as a individual class. We can combine them easily in main program (main node). The main node consists of many clients in order to call the services in different node at different time.

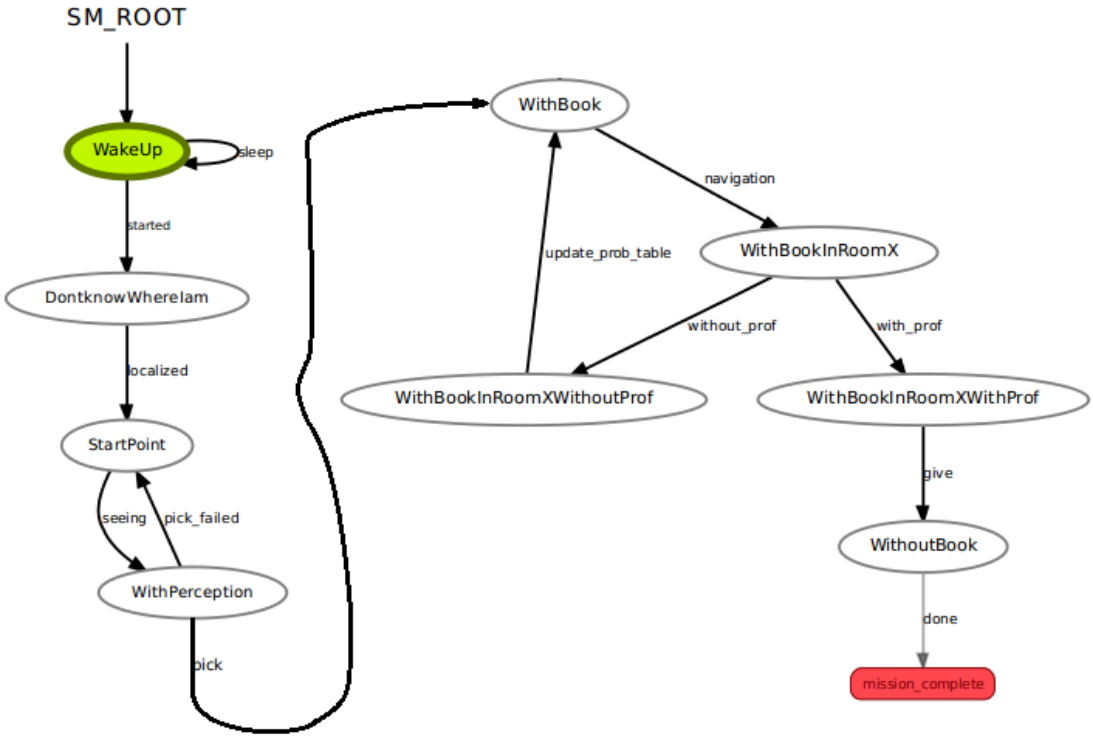

Figure 3: Framing of state machine

3.1.1 Descriptions of states

Each state represent a specific status such as the location, gesture and some abstract description of robot.

WakeUp

Robot sleeps in this state. An input is needed to wake it up.

DontKnowWhereIam

Robot will do localization in order to find where it is, in this state, the robot alse listens the topic /recognizer/output in order to ensure it is successfully localized. If the robot is successfully localized, the state will transfer to StartPoint.

StartPoint

Robot has been waked up and will lift its torso and start the gesture detection, in order to ensure someone is giving him something. After gesture detection, the node will enable Yolo detector and robot will try to detect the 3D coordinates of the given object. When the 3D-coordinates-detection is done, the node will shut down the Yolo detector. The state will transfer to the state WithPerception. In this state, the robot already knows where the object is and will do the grasping. If the grasping failed, the state will transfer to StartPoint in order to re-detect the object. If the picking is succeed, the state will transfer to the State WithBook.

WithPerception

In this state, the robot will use the data from the previous state to do navigation. After the navigation, the state will transfer to the State WithBookInRoomX.

WithBook

In this state, the robot will use the data from the previous state to do navigation. After the navigation the state will transfer to the State WithBookInRoomX.

WithBookInRoomX

In this state the robot will do face recognition in order to find the professor. If he can find the professor, the state will transfer to the state WithBookInRoomXWithProf, otherwise, the state will transfer to the state WithBookInRoomXWithoutProf.

WithBookInRoomXWithoutProf

This state means that the robot is in a room without professor. In this state the robot will update the data and feed this data to the State WithBook in order to do a new navigation.

WithBookInRoomXWithProf

In this state, the robot has already found the professor and will give the object to him. After the giving, the robot will unfold its arm to the safe place and this state will transfer to the state

WithoutBook

In this state, the robot has already given the object to the professor. So the state will transfer to final outcome `"mission_complete"` without any action.

3.1.2 Description of the data transition among states

From the Figure3 we can see, that the state WithBook, WithBookInRoomX, and WithBookInRoomXWithoutProf form a loop. If the robot cannot find the professor at state WithBookInRoomX, the loop will continue. Since we update the data e.g. the probability table(describe the probability of find the professor in a room in condition that we cannot find the professor in other rooms), and transfer to the state WithBook. And The state WithBookInRoomXWithoutProf also need new data from the state WithBookInRoomX to do update. So the data transfer among these 3 states is constructed. In Smach we can make it simpily by defining the input and output keys of states.

3.2 Navigation Part

The Robot needs a map to do navigation, so we used Gmapping tool to create a map of second floor. The localization and navigation as server was written in navigation node.

3.2.1 Mapping

The gmapping package gives us a convenience to do mapping. The package provides laser-based SLAM, with it, we created a 2D occupancy grid map from laser and pose data collected by a mobile robot. The mapping result is illustrated in Figure 4.

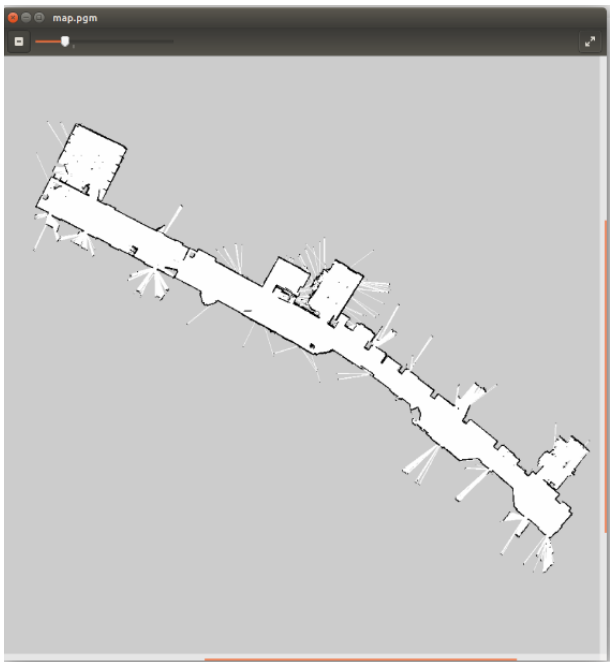

Figure 4: The map of second floor

3.2.2 Localization server

The localization server advertises the service with the topic `/do_localization` with the service type of `std_srvs::Empty`. In the callback of this service, the server will call the service `/global_localizaton` , this causes the amcl probabilistic localization system to spread particles all over the map. After that, the program will let the robot whirl itself in order to ensure its location. At the end, the service `/move_base/clear_costmap` will be called in order to clear the erroneous data due to the bad localization of the robot. The illustration of localization is shown in figure 5.

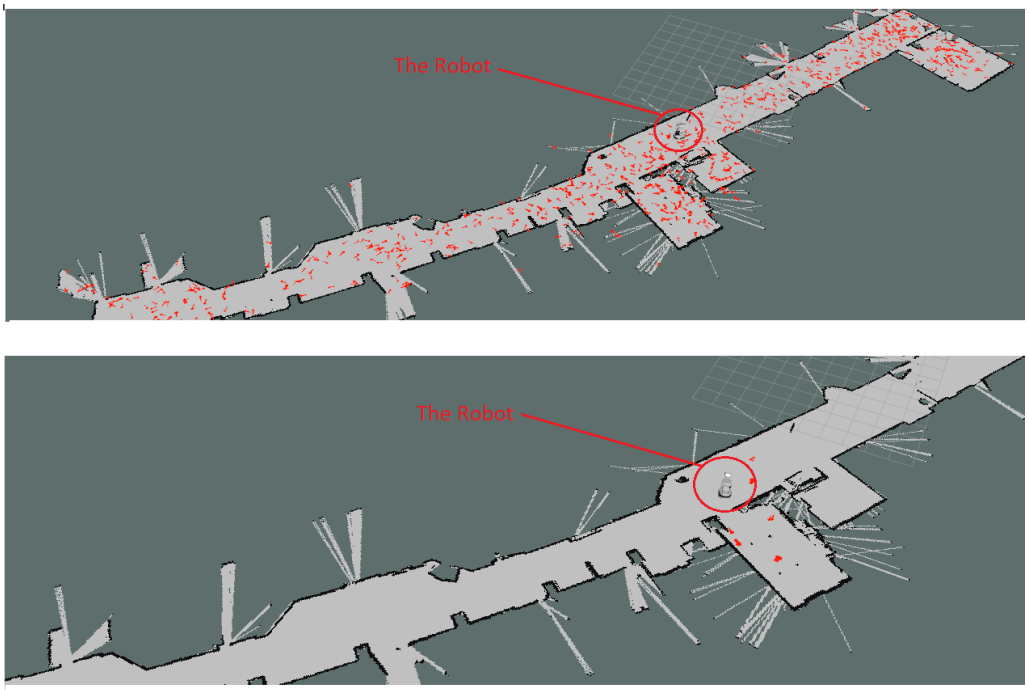

Figure 5: Localization, top: beginning of localization, bottom: end of localization

3.2.3 Navigation server

This server advertises the service with the topic `/move_base` with the service type `final_msg_srv::desired_position`. The callback get the request position as navigation goal from the call of this service and send the goal to the navigation action server by a predefined move base action client. Then let the action server to do the navigation.

3.2.4 Bayesian updating on choosing the navigation goal

The Bayesian updating (BU) is based on joint probability and conditional probability. The principle of BU is to improve our prior probability estimate to produce a posterior probability estimate. For this problem, we have n rooms, each room has a probability $P(f_i)$ to find the professor in this room, where i is the number of the room.This Part references [3:lanillos]. With the assumption that professor

stays in a room and dont move, we have:

$$\sum_i P(f_i) = 1.$$

With this assumption, we can initialize a probability table with n rows by using our estimated prior probability. The probability table looks like Table 4.

Table 4: The probability table of finding the professor in each room

room number	1	2	3	4	...
Probability professor in this room	$P(f_1)$	$P(f_2)$	$P(f_3)$	$P(f_4)$	...

The robot will go to the room with the maximum probability (for example room j) to find the professor. But if he cannot find the professor in room j, the probabilities to find the professor in other rooms will raise under this condition. Bayesian updating tells us how to calculate this probability. The probability need to be calculated is:

$$P(f_i|n_j).$$

Here, nj is the event that TIAGo havent found the professor in j-th room. Follow the calculating principle of conditional probability. We have:

$$P(f_i|n_j) = \frac{P(f_i n_j)}{P(n_j)}.$$

With $P(n_j) = 1 - P(f_j)$ and $P(f_i n_j = P(f_i)P(n_j f_i))$ we have:

$$P(f_i|n_j) = \frac{P(f_j)P(n_j f_i)}{1 - P(f_j)}.$$

The conditional probability $P(n_j|f_i)$ represents that the robot cannot find the professor in room j under the condition the professor has been founded in room i. This conditional probability must be 1. So The posterior probability after one observation is:

$$P(f_i|n_j) = \frac{P(f_i)}{1 - P(f_j)}.$$

Following this rule we update the probability which only has n-1 rows. Like Table 5.

Table 5: The probability table after one observation

room number	1	2	3	4	...
Probability professor in this room	$P(f_1\|n_j)$	$P(f_2\|n_j)$	$P(f_3\|n_j)$	$P(f_4\|n_j)$	...

The above works is recursive. So, we use this rule after each observation to update the probability table until we find the professor. Instead of write this recursive update in a navigation server we wrote it in the main node. Because this update also involve to transitions of 3 states i.e. WithBook, WithBookInRoomX, and WithBookInRoomXWithoutProf. The state machine gives us the convenience to do this update by the transfer of the user data among states. But this is still an issue of navigation.

3.3 Speech part

3.3.1 Speech recognition using Pocketsphinx

The localization program cannot return a sign which tells us the localization is succeed or failed. It is needed to compare its position in rivz and its real position. And give a program a input which shows the result of localization But the requirement is, after we press start button, we cannot type anything during the running of the program. So here, the speech recognition plays its roll. We can simply tell the robot if its localization failed through a microphone. With a speech recognition node, the robot can easily understand what we said.

We use the package Pocketsphinx to construct our speech recognition node the recognizer use the audio as the input and output the corresponding String message yes or no to the topic /recognizer/output. The main node subscribe this topic. If the message was no it will do the localization again. If the message was yes the robot will do the navigation. The testing result is shown in Figure 6.

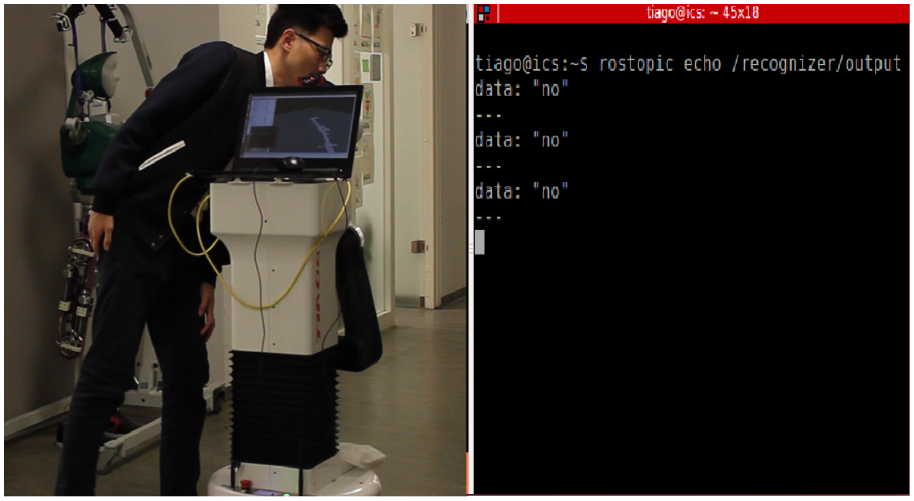

Figure 6: Speech recognition

3.3.2 Using TTS to let TIAGo Speak

In this node we implement an action client, which connects to the text-to-speech(TTS) server. The TTS engine from Acapela Group is incorporated in TIAGo. It can predict the appropriate prosody for the input text utterance and generate the corresponding signal waveform, then plays it using TIAGo speakers.

As a bridge between the speech node and other nodes, we also implement a server. So we can pass the text, which is needed to be spoken, from other nodes to the speech node through a service call.

3.4 Perception part

3.4.1 Gesture detection using OpenPose

This part is intended to detect the gesture of rising hand at the very beginning of the task. Once TIAGo have detected that a person has risen hand before him, he will consider that this person is offering something to him. As a result the object detection will be started.

At first we used the person model in the tf-tree of the TIAGo to do the ges-

ture detection, but it fails as long as TIAGo cannot see the whole body of the person or there are more than one person in his vision. Besides, the response of the tf-tree is really slow.

In order to improve this part we then decided to use Openpose, a Real-time multi-person key-point detection library for body, face, and hands estimation using OpenCv and Caffe. To call the Openpose we wrote a Python API, in which an OpenPose object is constructed. The algorithm comprises the following steps:

1)

The image that is taken by the camera of TIAGo will be converted from `sensor_msgs::Image` message format into cv::Mat format using CvBridge, and then passed to the Openpose object.

2)

Based on the input image OpenPose can estimate 18 key-points of the body and return the pose positions as numpy array.

3)

The coordinates of the hands in the current frame and the last frame in the vertical direction will be compared.

4)

Once the difference is larger than the threshold value, the gesture will be considered as rising hand.

The Testing results are shown in Figure 7.

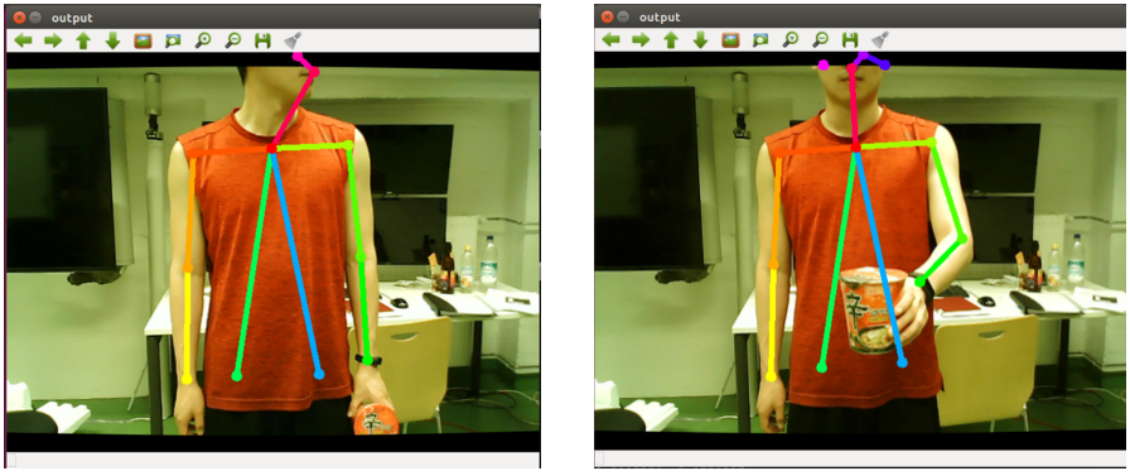

Figure 7: Gesture detection, Left: OpenPose detected gesture, Right: OpenPose detected the raising of operator's hand

3.4.2 Object detection using Yolo

As soon as the gesture of rising hand is detected, the object detection will be triggered. The main process of the object detection can be divided in following parts:

1)

The object in the hand of the person will be detected using Yolo, a real-time object detection system. In this system, a single neural network will be applied to the full image. It divides the image into regions and predicts bounding boxes for each region.

2)

The bounding box with the label of cup will be picked out and the 2D pixel coordinates of its central point will be computed.

3)

With the help of depth-image taken from depth-camera and the parameter of depth-camera, the 3D coordinate of object relative to the camera reference can be computed as follows:

$$x^{3D} = \frac{s_x(x^{pixel} - o_x)d}{f_x},$$

$$y^{3D} = \frac{s_y(y^{pixel} - o_y)d}{f_y},$$
$$z^{3D} = d.$$

Where:
$o_x(o_y)$: The number of pixel between $x_(pixel)(y_(pixel))$ and the central of the image.
$s_x(s_y)$: The length of one pixel in x-axis (y-axis)
$f_x(f_y)$: the rectification parameter of the depth-camera in x-axis (y-axis)
d: the depth of the point relative to the camera reference

The Yolo object recognition is illustrated in Figure 8.

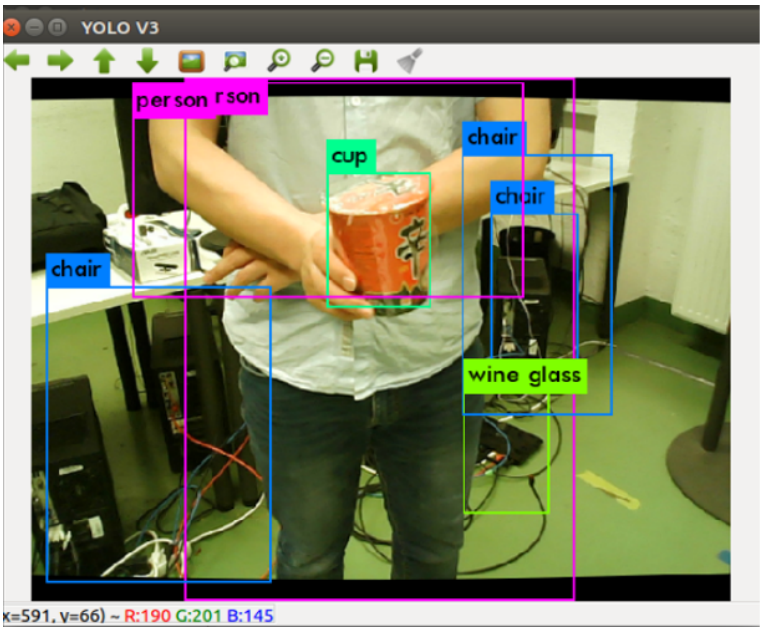

Figure 8: Yolo object detection

Since the Moveit planning of the arm is really strict about the target position, we set a condition to restrict the 3D coordinate of the object in a special region, in which the success rate of the motion planning is relative high. Thus the operator is expected to hold the object, so that its central point falls in this region. It will cost some time to adjust the position of the object, but it will waste more time

if the planning fails. Now having the 3D coordinate of the object, TIAGo will be able to reach for it.

3.4.3 Face recognition using the Python face recognition library

The face recognition part is used to distinguish the professor from other people. If TIAGo recognized the professor, he will pass the food to the professor. Otherwise, he will go on looking for the professor in other rooms. To implement this part we use the Python library `face_recognition`. The basic steps of the algorithm are:

1)

Find all the faces in the input image. In this step the image will be encoded based on the HOG algorithm, so a generic HOG encoding of a face can be easy to find.

2)

Find the main landmarks in the face and figure out the pose of the face. According to those landmarks the image will be warped so that the eyes and mouth are centered.

3)

Pass the centered face image through a pretrained neural network and get 128 measurements of the face features, like the length of the face, the width of the mouth.

4)

Compare those measurements between faces. If the difference is smaller than the threshold, they are considered as the same face.

In order to improve the accuracy of the recognition, we compare the picture of the professor with the face in ten sequential frames. Only if they match in more than 5 frames, the person will be recognized as professor. The results of face recognition are illustrated in Figure 9.

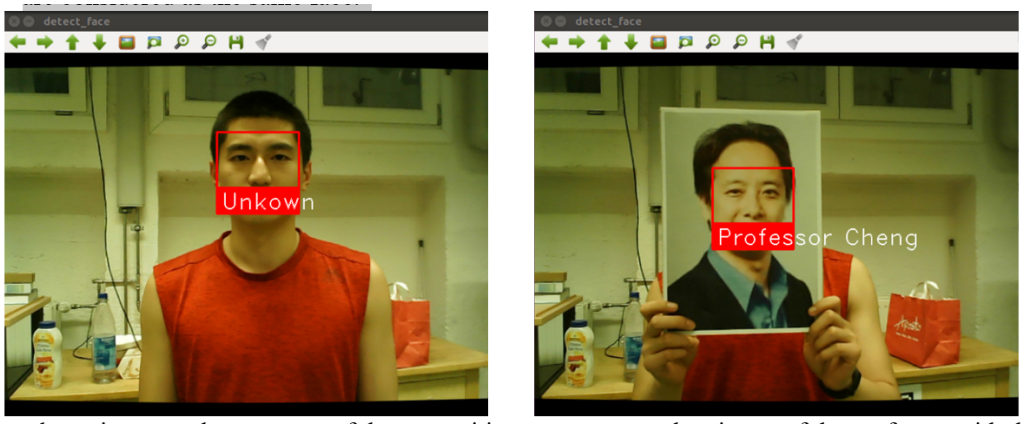

Figure 9: Face recognition, Left: unknown person, Right: the professor

3.5 Manipulation part

3.5.1 Pick and place

Manipulation part consists of two parts. The Basic structure is illustrated in Figure 10.

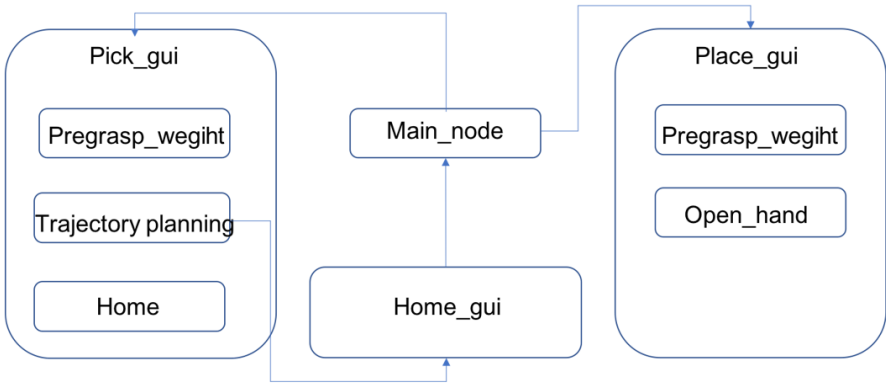

Figure 10: Basic framing of manipulation

The first part is pick part, TIAGo will pick the object when the `pick_and_place_node` received the 3D coordinates of detected object.

The second part is place part, when the corresponding node receives the message and knows target person have been identified, the force detection will work,

now if someone shake the object, which grasped by TIAGo. Then it will release the object and move back to home position, at the same time, the Speech part will be activated, the signal will be announced, whether the mission is completed or not.

For the pick part:

1)

The `pick_and_place_node` will receive message from `main_node` to get the coordinates of objects center point.

2)

Based on the received point, the `pick_gui` Service will be activated. In order to decrease the burden to compute the trajectory for picking the object, TIAGo will move to the `pregrasp_weight` position at the first time.

3)

After activating `pick_gui` Service, string pick will be send to the relevant function, and functions which related to the pick action will be called.

4)

Based on the received information from sensor and `main_node`, which have been send to the MoveItPlanningScene, after activating Moveit Action, the trajectory will be planned without collision.

5)

When the trajectory plans succeed, TIAGo will follow the trajectory plan, and start to grasp the object. Otherwise, TIAGo will send message to the `main_node` to let the `main_node` know that TIAGo has just failed to pick, it needs to restart perception. Meanwhile, in order to protect TIAGo,`/home_gui` Service will be called and then TIAGo will move to safe gesture.

6)

Then `main_node` will receive the message published by this node, then `main_node` could move to the next state.

Figure 11 shows that tiago grasps the shin cup from the operator

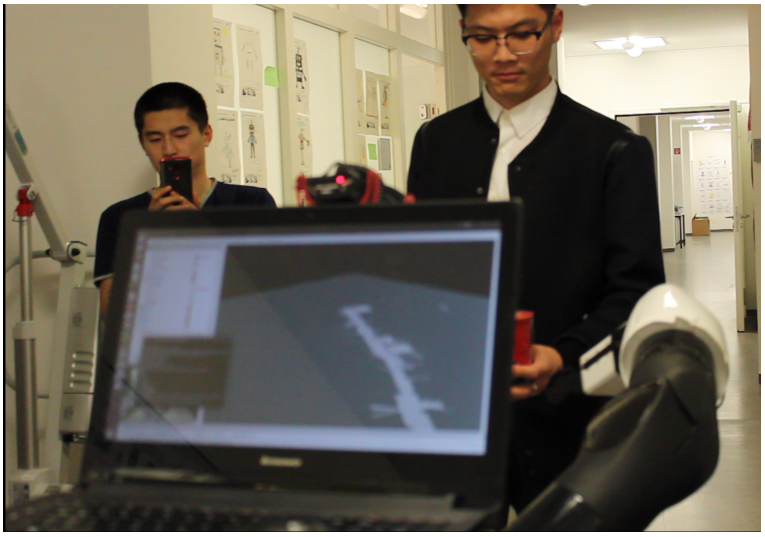

Figure 11: Tiago grasping the cup from the operator

For place part:

1)

When the target person have been detected, `main_node` will call the `place_gui` Service. In order to let the target person get the object more conveniently, TIAGo will move to the `pregrasp_weight` gesture.

2)

When the force difference between right and left part of Force/Torque Sensor is above threshold, Tiago will release the object.

3)

In order to keep TIAGo safe, it will move back to the home gesture.

Figure 12 shows that TIAGo gives the Shin Cup to the object person.

Figure 12: TIAGo giving the cup to a person

3.5.2 Supports the main node to call the pick service — grasping client node

The node `pick_and_place_client` provides the `Empty` service `/pick_gui` . In the former demo, the callback of this service contains a while-True loop, the loop breaks when the subscriber get the 3D coordinates from the object detection node.

In our Design, we prohibit the communication of sub-nodes(so we deleted the subscriber of pick and place client), in this case, the pick and place server never gets the 3D point. Its callback will be blocked infinitly in this while loop. So the main node will also be blocked since it called the service `/pick_gui`. In this case, we provide a new service in the node `grasp_client_node` called `/move_arm` to provides the 3D coordinates to pick and place client.

Figure 13 shows how `grasp_client_node` works. The basic logic is a time series from 1 to 6.

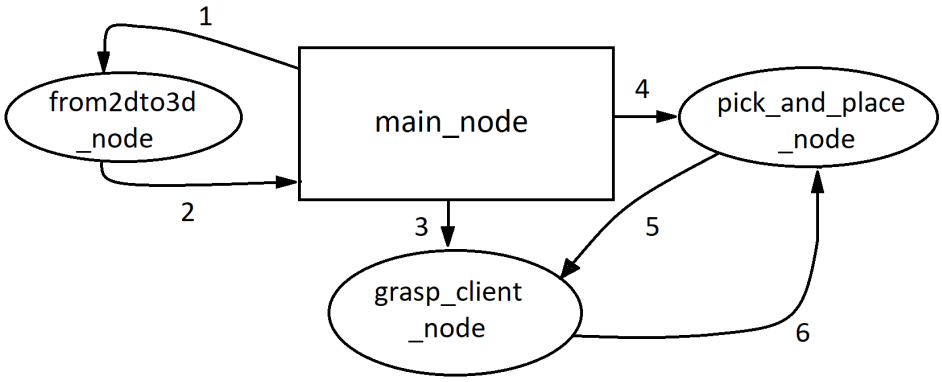

Figure 13: How grasp node works

1 - 2

The `main_node` call the service(1) and get the 3D point of the desired object(2).

3

The `main_node` publishes this 3D coordinates to the `grasp_client_node` and the `grasp_client_node` saves this data in local variables.

4

The `main_node` call the service `/pick_gui` to make the `pick_and_place_node` ready to pick.

5-6

When the `grasp_client_node` ready to pick, it publishes a logical message (5) to tell the `grasp_client_node` it is ready (in while loop).

When `grasp_client_node` got the sign, it will call the service `/move_arm` to send the 3D coordinates to `pick_and_place_node` so that the `pick_and_place_node` can break the while loop and start grasping.

References

[1] ROS. Accessed 23.07.2018. Roslaunch/Api Usage. *http://wiki.ros.org/roslaunch/API.*

[2] Sahith Dambekodi. Accessed 23.07.2018. CMUSphinx *https://cmusphinx.github.io/.*

[3] Pablo Lanillos. Minimum Time Search of Moving Targets in Uncertain Environments. Ph. D. disseration, University Complutense of Madrid, Madrid, Spain, 2013.

[4] Jonathan Bohren. Accessed 23.07.2018. Smach *https://github.com/ros/*`executive_smach`.

[5] Brian Gerkey. Accessed 23.07.2018. Gmapping. *http://wiki.ros.org/gmapping.*

[6] Pal Robotics. Accessed 23.07.2018. TIAGo. *http://wiki.ros.org/Robots/TIAGo.*

[7] CMU-Perceptual-Computing-Lab. Accessed 23.07.2018. OpenPose: Real-time multi-person keypoint detection library for body, face, and hands estimation. *https://github.com/CMU-Perceptual-Computing-Lab/openpose.*

[8] Redmon, Joseph and Farhadi, Ali. Accessed 23.07.2018. YOLOv3: An Incremental Improvement. *https://pjreddie.com/darknet/yolo/.*

[9] ETH Zurich Legged Robotics. Accessed 23.07.2018. YOLO ROS: Real-Time Object Detection for ROS. *https://github.com/leggedrobotics*

[10] Ageitgey. Accessed 23.07.2018. The world's simplest facial recognition api for Python and the command line. *https://github.com/ageitgey/*`face_recognition`.

[11] Adam Geitgey. Accessed 23.07.2018. Modern Face Recognition with Deep Learning. *https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78. [Accessed 23 July 2018]..*