SKUBA-Jr 2021 Team Description Paper

1st Akira Techapattaraporn
Dept. of Electrical Engineering
Kasetsart University
Bangkok, Thailand
akira.te@ku.th

2nd Runchida Bunamorn
Dept. of Electrical Engineering
Kasetsart University
Bangkok, Thailand
runchida.b@ku.th

3rd Kanjanapan Sukvichai
Dept. of Electrical Engineering
Kasetsart University
Bangkok, Thailand
fengkpsc@ku.ac.th

***Abstract*—The purpose of this paper is to describe the service robot named Angus from the SKUBA-Jr team for the 2021 World RoboCup@Home Education Online Challenge. Angus was assembled from Kobuki TurtleBot2 platform with a 5 degrees of freedom manipulator, a 2D Lidar scanner and a depth camera. The software applied on Angus, for instance: image processing, speech recognition, and navigation and localization, is based on C++ and Python under ROS environment. The convolution neural network, YOLOv4-Tiny, is used for objects detection. PocketSphinx speech recognizer was utilized for speech synthesis. The robot manipulator was controlled by using Moveit! library with the custom configuration and custom path planing program. In addition, ROS packages had a significant role on the navigation scheme and mission states planner. In summary, all the items mentioned was constructed, combined and customized for our autonomous mobile robot to achieve an exquisite performance.**

***Index Terms*—Mobile Robot, Robot Manipulator, Service Robot, Lidar, ROS**

I. Introduction

SKUBA-Jr team has been participating in the RoboCup@Home Education Competition since 2012 and was the winner of RoboCup@Home Education Competition 2019 at Sydney, Australia. The team is based at the Faculty of Engineering at Kasetsart University, Thailand, and consists of the undergraduate and graduate students adviced by professional faculty members. Our team has a strong motivation to develop a practical interpretation to be part of the World RoboCup and service robot advancement. By participating in the World RoboCup@Home League, our contribution would be evaluated through the certain situations, which elevate the development of our skills and knowledge in return.

The following sections consist of robot overview design, software overview, robot perception, navigation, arm manipulation, and performed tasks.

II. Hardware Design

A. Robot Design

The structure of Angus is presented in Figure 1. Angus was built upon the Kobuki TurtleBot 2 base with two differential drive wheels. The modification of Angus was to incorporate more layers to the robot base in order to expand the capacity for the additional hardware such as onboard computer, manipulators and sensors. A 3D printed PLA plastic shell was made to cover the Angus body in order to make a friendly appearance as it was meant to be a service robot. The sensors used in the system are a low cost 2D laser scanner (Lidar) and the Intel RealSense D415 mounted to the robot head for robot perception. On the platform of the first layer, a lidar is attached for mapping and navigation. A computer and some other miscellaneous items were settled on the next level. Lastly, a microphone was attached to the top of the head for sound receiving.

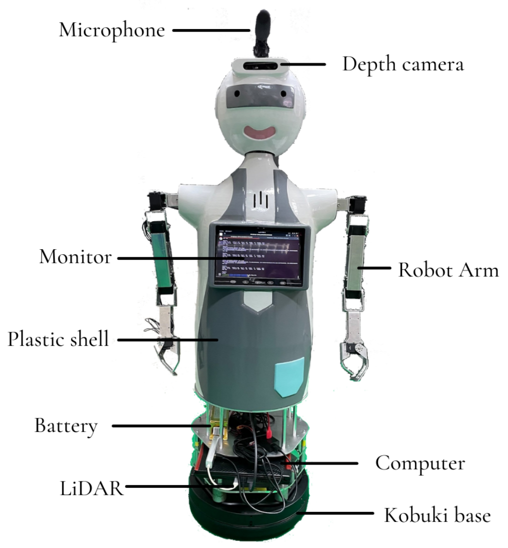


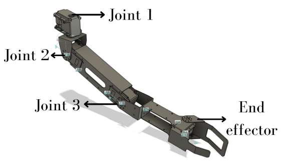
Fig. 1: Robot Design

B. Robot Manipulator

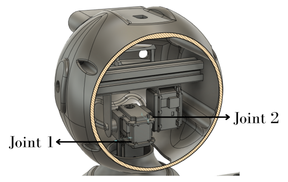
The robot arm was constructed from 3 revolute joints to achieve 3 degree of freedom (DOF) as shown in figure 2a. The end effector was constructed from 2 servo motors for two extra DOF. The materials used was folded aluminum sheets and some 3D printed parts, for weight minimization. Each joint was manipulated by a Dynamixel servo which can have 8.4 Nm. maximum torque at 12 V, 5.2 A.

C. Robot Head

The design of the robot head was to imitate the movement of head in human. With 2 degree of freedom, the robot head can realize the similar movement of rotation and extension as the human head does. The components used are 2 Dynamixel servos with 8.4 Nm. torque at 12 V, 5.2 A, and a plastic shell to cover for an amiable appearance.



(a) Robot Arm



(b) Robot Head

Fig. 2: Robot Hardware

III. Software Design

The software architecture is shown in Figure 3. It is operated by ROS and could be broken down into five main sections: sensor, navigation, robot perception, robot arm, and mission planner. The sensors module consists of lidar, depth camera, and microphone. The data obtained from the lidar is committed to the navigation for localization and trajectory planning. The microphone and depth camera are responsible for receiving input for the robot perception module. The mission planner should be performed after attaining the data from the robot perception, which would execute specific tasks such as running the robot arm as well as acquiring the arm status, utilizing navigation, or synthesizing speech through certain software.

A. Robot perception

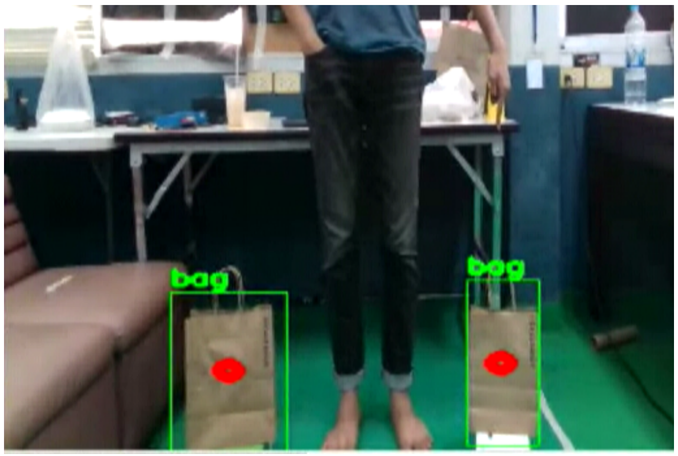


Fig. 4: Objects Detection Output

1) Objects Detection: The objects detection system utilizes the Convolutional Neural Network (CNN) to detect and classify objects. The real time neural network, YOLOv4-tiny running with darknet-53 architecture framework was applied. The YOLOv4-tiny is the compressed version of YOLOv4 which propose by Alexey Bochkovskiy, Chien Yao Wang and Hong Yuan Mark Liao [1] has the simpler network structure. The YOLOv4-tiny takes two-stage approach in adversarial training. In the first stage, it alters the original image to create the deception if there is no desired object on the image. For the second stage, the adversarial modified image will be detected by the trained network. The result of the object detection is shown in Figure 4

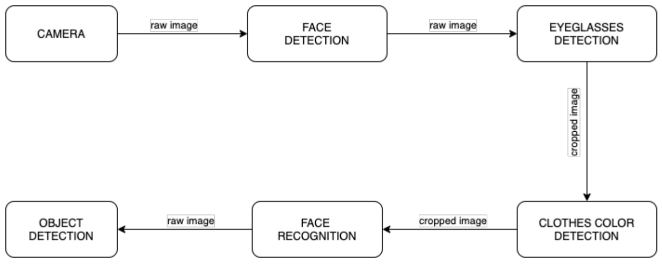


Fig. 5: Face Recognition Flow

2) Face Recognition: In order to perform the face recognition, the image is encoded by using the CNNs model to segment the face from background. The segmentation then passing through another CNNs model for facial feature extraction and the data is aligned and collected in array. The collected data will be used as a norm of the original image and is used to compare with the desired image that has been encoded, and the face is considered to be recognized if the encoded value is not greater than tolerance.

3) Speech Recognition: PocketSphinx [2] which is an offline light weight speech recognizer library written in C programming language was used for speech recognition. This library is one of the speech recognition tools kit called CMUSphinx developed by Carnegie Mellon University. PocketSphinx allows customization of the language model, phonetic dictionary, and acoustic model those entities are combine in an engine to recognize speech.To handle the speech data obtained from the audio source, Gstreamer library which is the streaming media framework was utilized to send the data to the PocketSphinx via an audio pipeline.

4) Seat Detection: The seat detection was developed based on the implementation of the Point Cloud Library (PCL) in ROS. The PCL is a C++ library for 2D/3D image and point cloud processing [3] which contain state-of-the-art algorithms such as filtering, feature estimation, surface reconstruction, model fitting, and segmentation. The approach for seat detection concerns the plane model segmentation and other filters along with pcl_ros [4], a ROS package which provides interface tools for joining ROS system to the PCL. Firstly, the inputted point clouds were down sampled using the Voxel Grid filter. Then, they were segmented using a plane model to detect a planar surface since the seats are flat via Random sample consensus or RANSAC algorithm. The output from the processed point clouds were converted to image, then, verified

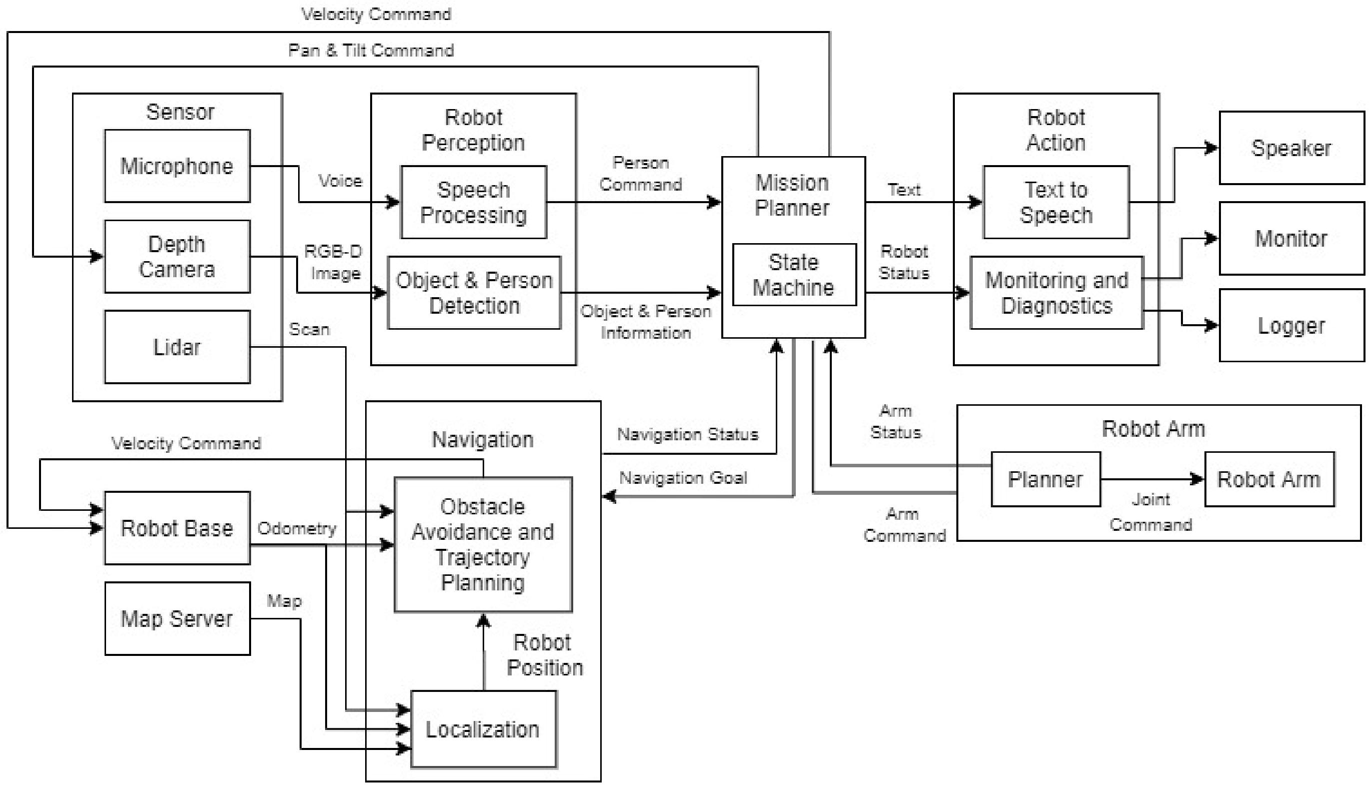


Fig. 3: Software Diagram

with the data obtained from the object detection through the ROS package pcl_ros.

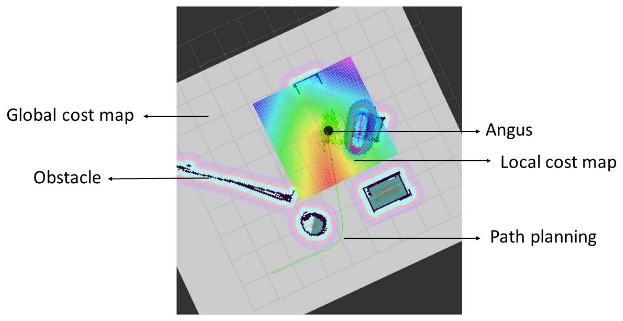


Fig. 6: Navigation Localization and Trajectory Planning

B. Navigation

The navigation scheme based on ROS Navigation stack [5] was used, which consists of two following items: localization and trajectory planning, as shown in figure **??**. The localization is based on AMCL algorithm [6] which indicates the position of the robot corresponding to its certain surroundings, for this reason, odometry, laser scan data, and a map are required. Odometry which approximates the position and the velocity of the robot is acquired from robot's wheels. Laser scan data is the distance between the robot and its surroundings data published by sensors such as laser, camera, or infrared. In this case, Lidar was used to attain the data. Finally, a map can be obtained by mapping process— the procedure of constructing spatial model according to the scenery, for instance, SLAM. On the other hand, trajectory planning consists of global path planning and local path planning. The global planner concerns the optimal path, while local planner calculates and generates suitable waypoints regarding the obstacles. When both mapping and localization are complete, the navigation can be executed efficiently along with the ROS move_base package [7] to achieve the goal through the robot's base controller which receives velocity commands from the navigation. The package also included action for the robot to relocate safely without colliding to the obstruction.

C. Mission States Planner

Since our autonomous robot is required to do designated tasks, the model-based task planning, state machine is manipulated. In this case, SMACH [8], a ROS-package is used to allow the programmer to execute the control of robots easily. The library consists of two main interfaces: States and Containers. In SMACH, a State is defined as the local state of execution corresponding to the system performing a certain task, meanwhile, containers are collections of one or more states. A SMACH state machine could be illustrated into a diagram hierarchically, where nodes [9] are the states of execution and edges are the transitions of each node corresponding to its outcome, as shown in Figure 7.

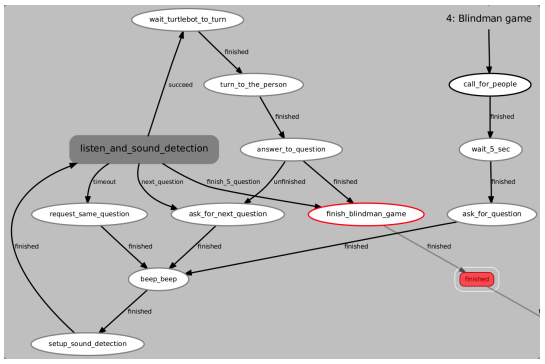


Fig. 7: ROS SMACH Viewer Example

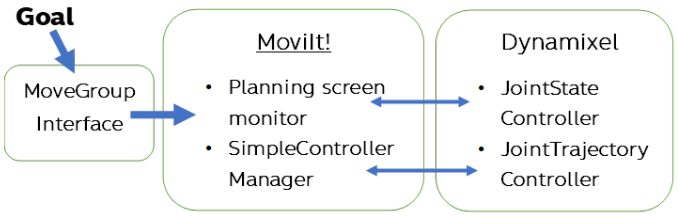


Fig. 8: Robot Arm Software Diagram

D. *Robot Manipulator*

To manipulate the robot arm for moving the object or pointing the arm's end effector to the specific location, the Moveit! library, the opensource robotics manipulator platform, was utilized. Moveit! combines four main functions of the robot manipulator which are a motion planning, collision checking, trajectory processing and motion executing, thus, the robot arm motion task can be achieved easily. The trajectory for each joint which is calculated from Moveit! library is sent to the dynamixel servos via the dynamixel controller package using ROS actionlib protocol. The robot arm software diagram is shown in Figure 8.

IV. Performing Tasks

The task chosen was Find My Mates, based on the competition task according to the rules 2021. The objective of this task focuses the efficiency of vision by obtaining the location and the description of the party guest by just knowing the guests' names. The procedure is referred to algorithm 1.

Algorithm 1 Find My Mates

```
Initialized state to Start_State
for Start_State do
    Listen to start command
    if hear start now then
        Go to Get_Name_State
    end if
end for
for Get_Name_State do
    Listen to a name
    if hear a name then
        Go to Detect_Person_State
    end if
end for
for Detect_Person_State do
    Walk to the living room
    Detect the selected person
    if can detect the selected person then
        Go to Report_State
    end if
end for
for Report_State do
    Go back to the first position
    Report the guest information
    Listen to a name
    if hear a name then
        Go to Start_State
    end if
end for
```

Acknowledgment

This project would not have been possible without the financial support from our professor, and the hard work and dedication of our team members.

References

[1] A. Bochkovskiy, C. Wang and H. Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection. 2020.

[2] Huggins Daines, David Kumar, M. Chan, A. Black, A.W. Ravishankar, M. Rudnicky, Alexander. (2006). Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices. 1. I - I. 10.1109/ICASSP.2006.1659988.

[3] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 1-4, doi: 10.1109/ICRA.2011.5980567.

[4] "pcl_ros - ROS Wiki", Wiki.ros.org, 2021. [Online]. Available: http://wiki.ros.org/pcl_ros. [Accessed: 22- May- 2021].

[5] "navigation - ROS Wiki", Wiki.ros.org, 2020. [Online]. Available: http://wiki.ros.org/navigation. [Accessed: 23- Mar- 2020].

[6] D. Fox, W. Burgard, F. Dellaert and S. Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots", pp. 343-349, 1999. Available: https://www.aaai.org/Papers/AAAI/1999/AAAI99-050.pdf. [Accessed 23 March 2020].

[7] "move_base - ROS Wiki", Wiki.ros.org, 2020. [Online]. Available: http://wiki.ros.org/move_base. [Accessed: 23- Mar- 2020].

[8] N. Hudson et al., "Model-based autonomous system for performing dexterous, human-level manipulation tasks", Autonomous Robots, vol. 36, no. 1-2, pp. 31-49, 2013. Available: 10.1007/s10514-013-9371-y [Accessed 18 June 2020].

[9] "Nodes - ROS Wiki", Wiki.ros.org, 2020. [Online]. Available: http://wiki.ros.org/Nodes. [Accessed: 18- Jun- 2020].