SKUBA 2013 Team Description

Kanjanapan Sukvichai, Krit Chaiso, Thanakorn Panyapiang,
Suppawit Inhorm, and Sutinai Tanalerkchai

Faculty of Engineering, Kasetsart University
50 Ngamwongwan Rd, Ladyao, Bangkok, Thailand
skuba2002@gmail.com
http://iml.cpe.ku.ac.th/skuba

Abstract. This paper is used to describe the SKUBA @home League robot team from Thailand. SKUBA@home is designed under the World Robocup 2012 rules. Based on the last participation, this year we're focusing on the new base platform which designed by using mecanum wheels and concentrating on ways to improve the performance of object recognition. The overview describes both the robot hardware and the overall software architecture of our team.

1 Introduction

SKUBA @home was established in 2011. In 2012, SKUBA @home made the first participation in Robocup Japan Open 2012 and made the way through finalist. Furthermore, SKUBA @home joined the World Robocup 2012 in Mexico and managed to pass to the 2nd stage as we anticipated. From last year issued, we decide to improve our robot performance by using mecanum robot platform in order to solve the "Walking in Elevator" issue. Thus, this year we hope to complete the "Follow Me" task. Furthermore, Team has developed a new object recognition technique.

The next section will explain about our robot that is designed for further research in the future. In the 3rd section is about software architecture and also includes object recognition algorithm. Section 4 will explained is about low level design and model of the robot including robot motion and robot odometry estimation. Finally in the last section, we will present the conclusion.

2 Robot Hardware

The new SKUBA@Home platform is designed as a three layer platform. The first layer is the driving mechanism layer. Second layer is a robot body which can be moved in vertical and the last layer is robot head and arm. Robot driving mechanism base consists of four 8" mecanum wheels. Each mecanum wheel is driven by Maxon EC 45 flat (brushless motor, 70 watt) BLDC motor combined with planetary gearhead of 1:36 gear ratio. The Hokuyo laser length finder is attached to the lower layer in order to obtain the environment information which

will be used in SLAM algorithm. This mecanum wheel has more mobility than the regular fixed wheel since it provides side movement. Now the robot can easily avoid the obstacles and has more flexibility to maneuver to the messy environment. The robot mecanum base is shown in Fig.1 below.

Fig. 1. The SKUBA@Home robot base

The second layer is the robot body. The robot body can be moved along vertical axis by using sliding bars which are driven by 70 watt DC windshield wiper motor. The robot head and arm are attached to this robot body. The Hokuyo laser length finder is also attached to the center of robot body in order to use in human tracking algorithm. The final layer is the robot head and arm. Robot head has two degrees of freedom neck which is duplicated from the human neck behavior. Kinect sensor is fixed to the top of the neck which can be used as the human eyes. High torque Dynamixel MX-106R smart servo, Dynamixel RX-28 smart servo and BLDC Maxon motor are used to construct the robot arm. Robot arm has 6 degrees of freedom which can perform more complex tasks. The robot arm is shown in Fig.2 .

Fig. 2. Robot arm

3 Software Architecture

Software system of our robot is divided into many modules with specific functionality, for example, object recognition module and manipulation module. The communication between modules is implemented by using Robot Operating System(ROS). The modules can be organized into three different layers:*perception layer*, *control layer* and *decision layer*.

Perception layer consist of modules about robot perception. Localization and Mapping, Object recognition and Speech recognition, for instance. Modules in this layer collect data from sensors, i.e. laser range finder, microphone and Kinect, to perform higher-level algorithms in order to identify state of the environment. Output of this layer is intermediate data for *decision layer*

Decision layer control robot's behaviour to solve complicated task. In this layer, decision is made base on user command and information from perception layer. User command is classified into three categories: *question*, *command* and *informative*. *Question* is a sentence which user expect to get a proper answer from robot. *Command* is used when user want robot to perform any action, for example, "*Bring me a pringle*" and "*Follow me*". When robot get an information, such as "*My name is Brian*" and "*This is kitchen*", these sentences will be classified as the *Informative command*.

When *Task layer* send command to robot, *Control layer* will interpret those commands into lower-level actions by using path planning algorithms.

Fig. 3. High level software architecture

3.1 Voice System

Speech recognition is implemented by using *pocketsphinx* base on *CMUSphinx* speech recognition toolkit. The incoming audio will be split into utterances and converted into sequence of words. The system require language model and dictionary which can create from words and sentences corpus,that we generate specifically for each task, by using Online Sphinx Base Tool.

The e-speak is selected as the robot speech synthesis software which uses a "formant synthesis" method. This allows many languages to be provided in a small size. The speech is clear, and can be used at high speeds, but it is not as natural or smooth as larger synthesizers which are based on human speech recordings.

3.2 Object Recognition

We develop this perception using both RGB image and depth image together. For localization approach, we use depth image using Point Cloud Library (PCL) for extracting the object from the background screen using Euclidean Clustering Extraction (ECE). After getting the position of the object, we remap the object position from 3D space to the pixel of RGB image for processing the recognition in the RGB domain. In image domain, descriptor of the object image are extracted using Speeded Up Robust Features (SURF), and will be classified using the predefined model creating with k-Nearest Neighbor Naive Bayes.

Fig. 4. Result from object recognition module

3.3 Gesture Recognition

NITE algorithm is the algorithm built in OpenNI framework can retrieve motion gesture such as wave, circle, swipe, push and steady. This algorithm return with hand position of all gesture detects. Moreover, it can be detect more than one gesture at a time. Therefore, we select NITE algorithm as our gesture recognition algorithm.

3.4 Face Recognition

The Haar like Features Cascade is used to detect faces. Moreover, we modified the input data by using of depth data from Kinect sensor in order to cut the background plane. In face recognition, simple eigenface technique is used.

3.5 People Detection and Tracking

People tracking can be done by using a laser scanner as a robot sensor. By using the Maximum likelihood approach algorithm associate data in the validation region and using Kalman filter, that has a constant velocity model system, as a filter. We can determine the position of the person.

After grabbing a person current position, the position will be redirect to Path planner module to locate the path from robot to the person. Meanwhile to solve the problem of robot moving to close to the obstacles, we have to modify the path with elastic band approach to make the path as center as possible.

4 Motion Control and Planning

4.1 Motor Control Module

Motor control module is based on Spartan 3 FPGA from Xilinx. The FPGA contains a soft 32-bit microprocessor core and peripherals. This embedded processor executes the low level motor control loop, communication and debugging. The motor controller, increment quadrature decoder, PWM generation and onboard serial interfaces are implemented using FPGA. The PI controller is employed to achieve each wheel velocity.

Fig. 5. Top view of Mecanum-wheel robot

In order to control a robot to move into the environment and avoid the obstacles, the robot model has to be considered. The robot model can be directly derived from the velocity relationship form each wheel to robot center (robot frame). Command velocity is the command input that high level software sends to the embedded system in order to change the robot position. Wheels velocities are converted from the command velocity by using kinetic Eq.1. The robot kinetic is derived from robot driving mechanism as shown in Fig.5 .

$$\zeta_{wheel} = \psi \cdot \zeta_{cmd} \tag{1}$$

Where,

$$\zeta_{wheel} = \begin{bmatrix} \dot{\phi}_1 & \dot{\phi}_2 & \dot{\phi}_3 & \dot{\phi}_3 \end{bmatrix}^T$$

$$\zeta_{cmd} = \begin{bmatrix} \dot{x} & \dot{y} & (a+b)\dot{\theta} \end{bmatrix}^T$$

$$\psi = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \\ 1 & -1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Where, $\dot{\phi}_i$ is velocity of wheel i.

ψ is the kinematic equation.

$\dot{x}\ \dot{y}\ \dot{\theta}$ are velocity of the center of robot in x direction, y direction and rotation respectively.

a and b is the distance from center of the robot to a wheel in x and y axis respectively.

ζ_{cmd} is the command velocity in robot frame.

ζ_{wheel} is the wheel velocity vector.

4.2 Localization and Mapping

Hokuyo laser length finder which acquires environment data is attached to the robot in order to implement localization algorithm. Additionally, the robot odometry information is also considered as one input information to localization algorithm.

Adaptive Monte Carlo Localization system (AMCL) is used for robot localization. AMCL is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses a particle filter to track the pose of a robot against a known map.

This known map is acquired with OpenSLAM Gmapping Package which is a highly efficient Rao-Blackwellized particle filter to learn grid maps from laser range data.

ROS navigation stack is used for robot navigation. A 2D navigation stack that takes in information from odometry, sensor streams, and a goal pose commands safe velocity to a mobile base.

Odometry Estimation Module This module helps to minimize the mean square error of the non-perfect sensor measurement originated from wheel slippage, dynamic surrounded environment and IMU drift over time. Using the laser scanner to estimate the change of position by laser scan matching technique can solve the wheel slip problem while facing with high sensitivity to dynamic environment. On the other hand, estimating the position by double integration of accelerometer causes the quadratic error due to bias noise as seen in IMU model Eq.2.

$$z_{measurement} = z_{true} + bias + w_{gaussian} \tag{2}$$

IMU Bias Noise Correction

Because of bias noise illustrated in Eq.2 is a very low frequency noise. Therefore, it can be determined offline every time when the robot is stopped more than predefined time. The exponential moving average method is used to smooth bias value.

$$bias_k = bias_{k-1} + K(z_k - bias_{k-1}) \tag{3}$$

Where K is the forgetting rate constant of the past bias value and z is the measurement from IMU. The raw information from IMU has to be corrected by bias value in Eq.3 before the use in next steps.

Kalman's Equation

Robot Operating System (ROS) is selected as the robot operation platform. It has built-in source laser scan matcher algorithm module which uses Point-to-line distance Iterative Closed Point (ICP)[3][4] method. These information and wheel encoders are the measurement information for the Kalman filter while acceleration from accelerometer is the control input in order to predict the present state variables at each sampling time because it is independent on ground surface and surrounded environment.The prediction equation for the Kalman filter is constructed as seen in Eq.4.

$$\begin{bmatrix} \hat{v}_x \\ \hat{v}_y \end{bmatrix}_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \hat{v}_x \\ \hat{v}_y \end{bmatrix}_{k-1} + \begin{bmatrix} a_{acc,x} - bias_{acc,x} \\ a_{acc,y} - bias_{acc,y} \end{bmatrix}_k \cdot \Delta T \tag{4}$$

Collision Avoidance Module Object Avoidance

Real-time Mapping Module Not Require

4.3 Manipulation Module

This module is intermediate between Planning module and USB-to-RS485 which exchange data from Planning module and the joint motors. This module constructs to be compatible with motor hardware and it returns standard message with ROS format.

5 Conclusion

In this year, we mainly focus on developing more stable robot and also it's AI software. We'll improve performance of manipulator as future work. We have experiences from RoboCup@Home competition 2012 and we learn from our mistakes. Our robot and AI are improved in order to join RoboCup2013 event in Netherlands. We hope that our robot team will perform better in RoboCup than the previous year, and we are looking forward to sharing experiences with other guest teams around the world.

References

1. Serken Tuerker, Euclidean Cluster Extraction, `http://pointclouds.org/documentation/tutorials/cluster_extraction.php`.
2. Radu Bogdan Rusu, Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments, Germany (2009).
3. A. Censi, An ICP variant using point-to-line metric, In Proceedings of the 2008 IEEE International Conference on Robotics and Automation (2008).
4. A. Censi, An accurate closed-form estimate of ICP's covariance, In Proceedings of the 2007 IEEE International Conference on Robotics and Automation (2007).
5. M. Munaro, F. Basso and E. Menegatti. Tracking people within groups with RGB-D data. In Proceedings of the International Conference on Intelligent Robots and Systems (IROS) 2012, Vilamoura (Portugal), 2012.