Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 11 (2013) 380 – 387

Procedia Technology

The 4th International Conference on Electrical Engineering and Informatics (ICEEI 2013)

Implementation of Entertaining Robot on ROS Framework

Inneke Mayachita, Rizka Widyarini*, Hadi Rasyid Sono, Adrianto Ravi Ibrahim, Widyawardana Adiprawita

Electrical Engineering Department, School of Electrical Engineering and Informatics
Bandung Institute of Technology, Ganesa Street 10, Bandung 40132, Indonesia

Abstract

As technologies are continually growing, there should be an option to implement technologies such as robots as entertainer at the tourism places, replacing the old-fashioned clowns and mascots. However, robots with a lot of features need relatively complex data processing. By implementing cloud robotics, all process and data calculation will be done by computers which are separated from the body of the robot. To implement this new robotics system, we use PeopleBot as the main physical platform, three computers, and a tablet. Communication between devices occurs via network and be facilitated by Robot Operating System (ROS) framework. Robot has some features such as Layout, image capture and storage in cloud, navigation, crowd detection, and speech recognition

Selection and peer-review under responsibility of the Faculty of Information Science & Technology, Universiti Kebangsaan Malaysia.

Keywords : ROS, cloud robotics, interactive, PeopleBot

1. Introduction

At the tourism places, such as theme parks, museums, and zoos, there can be usually found clowns or mascots to entertain and amuse the visitors. As technologies are continually growing, there should be an option to implement the use of technologies such as robots as entertainers at the tourism places, replacing the old-fashioned clowns and mascots. Similar to them, entertaining robots can wander around to offer entertainment to visitors by having an attractive appearance, showing various attractions, and also interacting with people. In addition, the advantage of

* Corresponding author.
E-mail address: rizkawidyarini@gmail.com

Selection and peer-review under responsibility of the Faculty of Information Science & Technology, Universiti Kebangsaan Malaysia.
doi:10.1016/j.protcy.2013.12.206

robots as entertainers is that robots can be designed to have additional functions, such as media displays to show helpful information to visitors. Furthermore, not only as entertainers and media displays, these sophisticated robots can broaden the public awareness of leading technologies nowadays.

However, robots with a lot of features need relatively complex data processing. In order to design a responsive as well as multitasking robot, large storage devices and power supplies are necessary. Those impact the size and cost value of the robots as they are pretty expensive. That is why it is essential to design a new system of robot implementation using cloud system that enables those processes to occur fast on a relatively light platform. By implementing cloud robotics, all process and data calculation will be done in computers separated from the robot. They will communicate with the robot through wireless network. This cloud system also enables the robot to have features-upgrading and data transfer with other robots in a more simple way.

The purposes of this experiment are to implement the concept of entertaining and interactive robot as well as taking advantages of cloud system. Entertaining features in this experiment are represented by the ability to take pictures and upload them to social media, recognize speech, show facial expressions, and playing songs, whereas the interactive features are represented by the ability to talk and detect human crowd.

2. Design

2.1. Software Design

2.1.1. Framework

This system uses Robot Operating System (ROS) as the framework to ease the message-passing process between modules in the system which each runs a specific program. The ROS framework enables multiple programs being executed and message-passing processes running in parallel.

There are several terminologies in ROS, such as node, topic, and message. The behavior of these objects in ROS is analogous with an internet forum, as shown in Table 1. Each node is able to publish and subscribe messages to/from one or several topics and interchange messages with other nodes.

2.1.2. System State Chart

There behavior of robot can be classified into six states.

- Wandering: robot moves from a checkpoint to another checkpoint.
- Crowd Detection: robot is on a checkpoint and detects if any crowd exists.
- Move to COM: robot moves from the checkpoint to the center of crowd.
- Near to COM: robot is near the center of crowd and recognizing any voice command.
- Interaction: robot is able to capture photos or display information, depending on the voice command.
- Back to Point: robot moves from its position to the last checkpoint.

Table 1. ROS Terminology and The Analogy

Terminology	Analogy
Node	User in a forum
Topic	Thread in a forum
Message	The interchanged messages
Publish message	Posting a message on a forum thread
Subscribe message	Reading a post on a forum thread
Package	A directory which contains certain programs
Stack	A set of packages

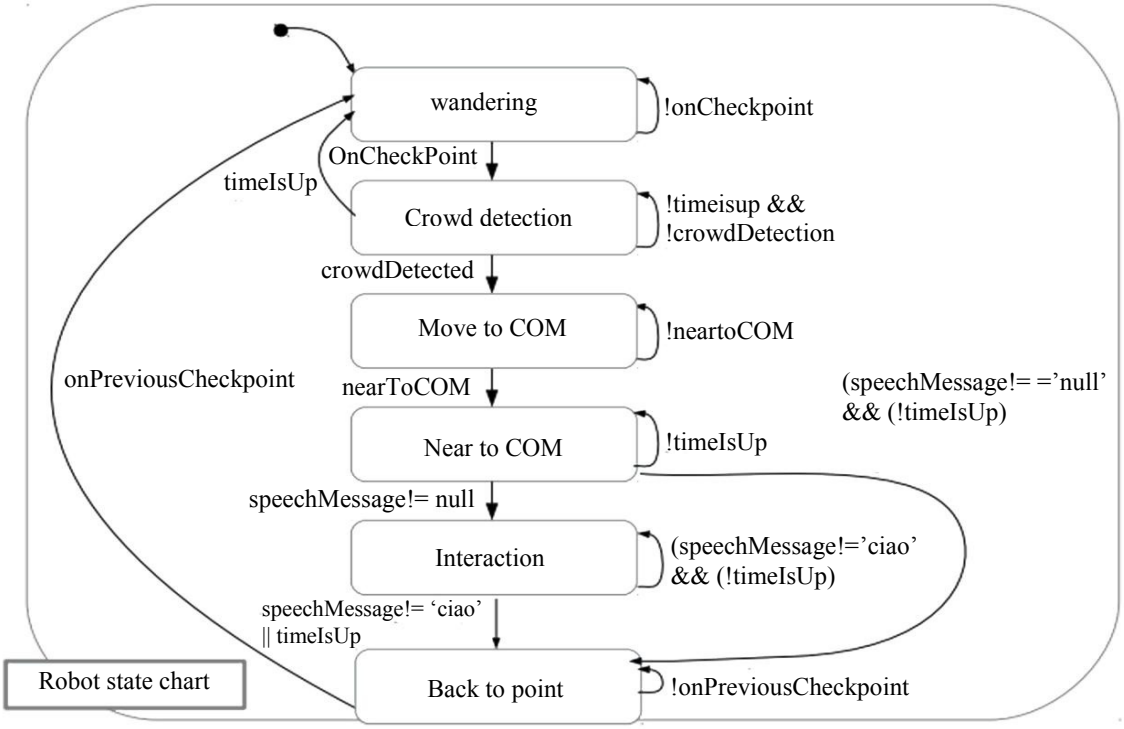


Fig. 1. System State Chart

2.2. Hardware Design

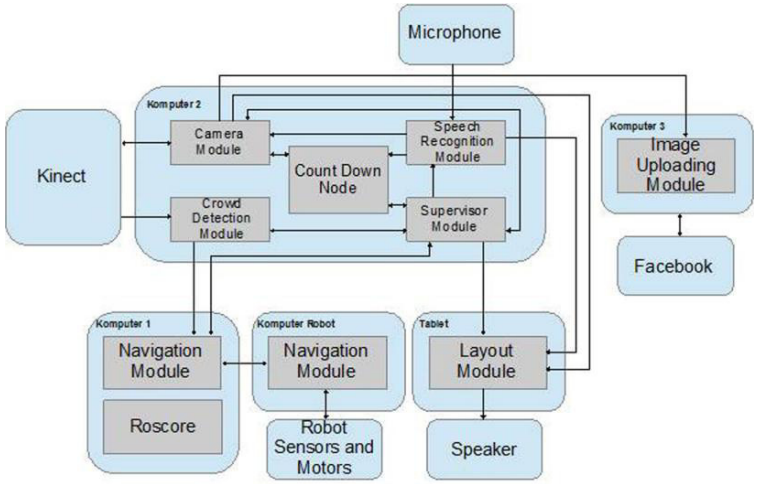
We use Peoplebot from Adept MobileRobots as physical platform in this project. According to ROS architecture concept in cloud robotics, we use some computers that are separated from robot body to process robot features. We use five computers: Kinect and Speech Recognition computer, Navigation computer, Upload computer, Display computer, and robot computer.

3. Functional Modules

3.1. Navigation Module

Navigation module makes trajectory planning for robot movement with the shortest path and obstacle avoidance. It also controls robot movement based on state to enhance robot interaction features. The nodes that build navigation module are shown in Fig 2.(c).

a



b

c

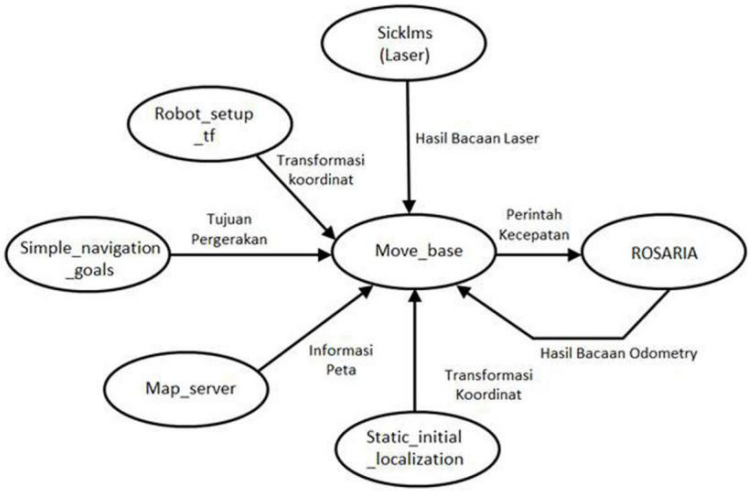


Fig.2. (a) Hardware Block Diagram of Robot System; (b) PeopleBot Appearance; (c) Navigation Module Node Architecture Diagram

- Map_server
  Map_server gives map information used in navigation process. Map helps the robot to know the environment around robot. Map can be used if there is transformation coordinate from map to robot.
- Sicklms
  Sicklms gets information from laser used to get information around robot in real time.
- Robot_setup_tf
  Robot_setup_tf gives transformation coordinate from sensors to center of movement (base_footprint). This transformation coordinate must be provided to use sensor information in movement process. Base_link is added between base_footprint and sensors to make the configuration easier. These are the transformation coordinate provided by robot_setup_tf: odometry→base_footprint, base_footprint→base_link, and base_link→laser.
- Static_initial_localization
  Static_lnitial_localization gives transformation coordinate from map to robot (localization). Transformation coordinate can be obtained manually by using visualization program RVIZ pose estimation.
- Move_base
  Move_base plans trajectory based on information from sensors, map, and goal. There are two kinds of trajectory planning: global plan and local plan. Global plan computes trajectory from start to goal position in the beginning of movement. Local plan computes trajectory in a short distance to avoid obstacles.
- Simple_navigation_goals
  Simple_navigation_goals controls robot movement based on robot current state. It gives goal to move_base.
- ROSARIA
  ROSARIA controls robot hardware. The used robot hardware is motor driver and odometry sensor.

We run tests to get information about accuracy of final position and result of trajectory planning. For final position test, we conduct 50 movements with goal distance between 100-200 cm and rotation. The result is shown on Table 2.

- X positive: forward translation; X negative: backward translation
- Y positive: to left translation; Y negative: to right translation
- theta positive: counterclockwise rotation; theta negative: clockwise rotation

For trajectory planning test, we conduct a movement with goal is reach crowd which is identified by Crowd Detection node. It also must avoided obstacle between robot and crowd. Fig. 3 shows result obtained.

Table 2. Test Result Accuracy of Final Position

	X-axis Translation (cm)	Y-axis Translation (cm)	Theta Rotation(^{0})
Mean Error	0.04	3.554	2.788
Standard Deviation	3.35865	4.8022972	4.8253035

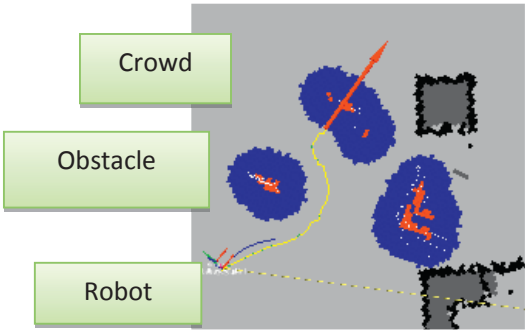


Fig. 3. Trajectory Planning Result

3.2. Crowd Detection Module

The module detects groups of human and determines whether they are crowds. The crowd's center of mass is calculated and published to Navigation module. The main component is Microsoft Kinect XBOX 360, which is able to detect up to 15 people, compatible with ROS, and relatively cheap.

Open Natural Interaction (OpenNI) allows communication between Kinect as the actuators with ROS as the robot's platform. To support the use of ROS in allowing human-Kinect interaction, we use Natural Interaction Engine (NITE) to implement the functions related this skeleton tracking function. Human detection algorithm by NITE skeleton tracker is based on background segmentation. Kinect compares this frame with the previous frame with a frame rate of 30 fps and resolution of 640x480.

The method that we develop is crowd detection method. The base principle of crowd detection algorithm is comparing the center of mass of all the users being tracked by Kinect. If the number of tracked users is greater and the distance among the centers of masses of those users is less than a specific number, that group of people will be detected as crowd. The users' centers of masses are calculated in X-axis (horizontal axis from Kinect's point of view) and Z-axis (distance from Kinect to the users). If there is a crowd detected by Kinect, the program will calculate the center of mass of the crowd based on this formula:

$$R = \frac{r_1 + r_2 + r_3 + \ldots + r_n}{n} \quad (1)$$

In this project, crowd is defined as a group of people which consists of more than 1 person. The distance among the users is not more than 150 centimeters. The testing is conducted by varying the positions of groups of people. From the testing, the average error is 2.86425 cm and 3.809472 cm in X-axis and Z-axis, while the error deviation is 2.27262 cm and 2.02774 cm in in X-axis and Z-axis.

3.3. Speech Recognition Module

We use ROS package `Pocketsphinx` from Carnegie Mellon University to implement this module. There are 2 nodes in this module, which are Recognizer and Speechtocamera nodes. Recognizer node is the main node which recognizes speech, whereas Speechtocamera connects this module and the others subscribing to this module. In this project, words that can be recognized are "PHOTO", "YES", "WHATSUP", "CANCEL", and "CIAO".

The testing is conducted by calculating the percentage success rate of each word. Results show that the success rate of this module is 564 out of 1200 experiments (47%). The success rate of word PHOTO = 61.667, YES = 70%, WHATSUP = 49.583%, CIAO = 32.5%, and CANCEL = 22.917%.

3.4. Camera Module

This module captures pictures and publishes them to Image Uploading module. The main component used is Microsoft Kinect XBOX 360. Open Source Computer Vision (OpenCV) and CVBridge is used to save pictures in the computer and convert picture message in ROS. This module consists of 3 main states: StartCamera, PhotoReady, and Capture. StartCamera is the default state for this module. PhotoReady is the state where this module is ready to take pictures. State transition from StartCamera to PhotoReady happens when message "PHOTO" is received. When the current state is PhotoReady, pictures will be captured if message "YES" is received or the 20-second countdown is finished. Pictures generated from Kinect are in OpenNI format and should be converted to OpenCV format by changing the color channel from RGB to BGR. To test this module, 100 pictures are captured, as programmed. The success rate of this module testing is 100%.

3.5 Layout Module

This module shows expressions of the robot character named Maya and certain information related to the tourism place. Layout has an important role as the visual media to give friendly impression to people. This module is active in each state of the system and has certain behavior as shown in Fig. 4.

This module is implemented on a 10-inch Android tablet. All programs are written in Java and uses android_core to enable the program running in ROS framework. There are four types of program in this module.

Table III. Video and Video Viewer Visibility Message

Video Viewer	Video	Visibility Message
ROS Video View	Showing default expression	"Layout_0"
ROS Video View 1	Ready to take a photo	"Layout_1"
ROS Video View 2	Giving a compliment on the captured picture	"Layout_2"
ROS Video View 3	Giving an information introductory video	"Layout_3"
ROS Video View 4	Video containing information	"Layout_4"
ROS Video View 5	Greeting the people	"Layout_5"
ROS Video View 6	Saying goodbye	"Layout_6"

Table IV. Container Content and Visibility Messages

Container	Content	Visibility Message
ROS Relative Layout	ROS Video View	"Layout_0"
ROS Relative Layout 1	ROS Video View 1	"Layout_1"
ROS Relative Layout 2	ROS Video View 2	"Layout_2"
ROS Relative Layout 3	ROS Video View 3	"Layout_3"
ROS Relative Layout 4	ROS Video View 4	"Layout_4"
ROS Relative Layout 5	ROS Video View 5	"Layout_5"
ROS Relative Layout 6	ROS Video View 6	"Layout_6"

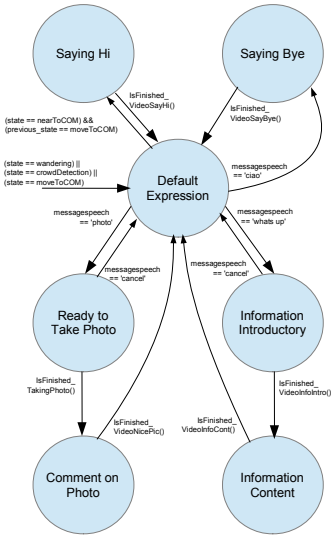


Fig. 4. Layout Module State Chart

- Video Viewer

This program plays certain video when the ROS node in this program receives certain message and stops playing it when the node receives other messages, as shown in Table III. There are seven Video Viewer programs each declaring a node that subscribes to the layout supervisor.

- Container

This program is a container where the contents of layout are placed. There are seven container programs in this module. Each handles different video contents and declares different ROS nodes. The ROS nodes subscribes to the layout supervisor. Each container is visible if the node receives a certain visibility message.

- Layout Supervisor

This program manages the order of the container visibility based on system state chart. The node in this program publishes certain message indicating which layout should be displayed. Several ROS subscribers are declared in this program subscribing the visibility of the containers and other inputs which cause the layout state transition.

- Main Activity

This program sets the whole application behavior on the Android tablet device by instantiating Layout Supervisor, Container, and Video Viewer objects, setting and executing the ROS nodes.

This module meets the requirement if the state transitions in the module are consistent with the system hierarchical state chart. Testing is done by applying various inputs in every state, both valid and invalid inputs. Based on this testing method, the layout module behavior suits the state chart design.

3.6 Image Uploading Module

The module has the role in photo uploading to Facebook, and is active when the system is in the Interaction state. During this state, the module state is Get Picture. The module waits for a new picture from Camera module. When a new picture message is published, the module state changes to Upload and the picture will be uploaded. Afterwards, the module state enters Get Picture and the module waits for another new picture.

The module is implemented on a computer which is connected to the internet to communicate with Facebook. There are two programs in this module, which are Image Listener and Uploader. Image Listener is a C++ program which uses CvBridge library to convert the received picture message from Camera module. In this program, a

picture message subscriber is declared. If a new picture is received, an image callback to convert the picture will be executed and a message to Uploader will be published. Meanwhile, Uploader is written in Java and uses RestFB library as the interface in doing communication with Facebook.

- **Functional Testing**

  For each step of implementation, functional testing is done using ten samples. The test result shows that ten from ten pictures are successfully uploaded in every step of implementation.

- **Network Speed and Image Uploading Duration**

  The network speed during the upload process and the time is measured. The image file size is 3.26 MB.

Table V. Network Speed and Time Duration

Tested Program	Library	Average Network Speed	Average Duration
Image uploading program with Image Listener on ROS	RestFB	339.5 – 661.2 kBit/s	64698.1 ms
Image uploading program with Image Listener on ROS	Modified RestFB	1.665 – 4.041 MBit/s	15856.2 ms

The program functionally runs well on the internet network with average minimum speed of 155 kbps and average maximum speed of 4.61 Mbps. The time duration varies from 15-75 seconds.

Acknowledgements

Supported by Directorate General of Higher Education Ministry of National Education, Indonesia.

Conclusion

All functional modules have been successfully implemented with certain constraints such as the requirement to interconnect all systems in the same LAN and internet connectivity.

References

[1] Quigley, Morgan. et. al. ROS: an open-source Robot Operating System. Stanford University.
[2] Concepts in ROS. School of Computer Science, University of Birmingham. http://www.cs.bham.ac.uk/internal/courses/int-robot/2012/notes/concepts.php; 2012.
[3] Walker, Willie. et. al. Sphinx-4: A Flexible Open Source Framework for Speech Recognition. Sun Microsystems; 2004.
[4] G. Bradski and A. Kaehler. Learning OpenCV; 2008.
[5] N. Villaroman, D. Rowe, and B. Swan. Teaching Natural User Interaction Using OpenNI and the Microsoft Kinect Sensor. Brigham Young University. New York: SIGITE'11; 2011.
[6] Goebel, P. ROS by Example: Speech Recognition and Text-to-Speech (TTS). http://www.pirobot.org/blog/0022/; Feb. 2013
[7] RestFB. http://restfb.com/javadoc; 2012.
[8] http://developer.android.com/reference; 2012.
[9] http://docs.opencv.org/doc/tutorials/;2013.
[10]. http://docs.rosjava.googlecode.com/hg/rosjava_core/html/index.html; Apr. 2013.
[11]. http://developer.android.com/reference/; Apr 2013.
[12] Zaman, Safdar, Wolfgang Slany, Gerald Steinbauer, ROS-based Mapping, Localization and Autonomous Navigation using a Pioneer 3-DX Robot and their Relevant Issues, Graz University of Technology;2011.