3D TRACKING CAPACITANCE BASED ROS CONTROLLED ROBOT ARM

Gunaratnam Pratheepan
Auston Institute of Management,
Colombo-03, Srilanka,
E-mail: pratheepan.guna@gmail.com

Dr. M. K. Jayananda
Department of Physics,
University of Colombo,
Colombo-03, Srilanka,
E-mail: kithsiri@phys.cmb.ac.lk

***Abstract*— This paper presents a design procedure of 3d tracking capacitance-based ROS controlled robot arm. A robot operating system (ROS) was modelled to control the robotic arm with 4 degrees of freedom (DOF) to imitate the virtual (Simulation) to reality. ROS is middleware upon which the robotics system can be developed or constructed. The basic controlling operation can be categorized into two sections. The initial process begins with RVIZ simulator control and then control by 3D Cartesian coordinates given by ROS commands to control the arm. The arm covers a quarter of sphere with a radius of 31 cm approximately. Capacitive based 3D tracking mechanism was used to control the robotic arm and update the Simulator by detecting hand gestures. The hand gesture movement was limited with-in 12 cm cubic box.**

Keywords- ROS; RVIZ; URDF; Robot Arm; capacitive sensing

I. INTRODUCTION

The field of robotics by nature is an interdisciplinary endeavour, where robots have been used in numerous applications from the bottom of the ocean to Mars and beyond. Robots are increasingly being integrated into performing the repetitive tasks to replace humans where human life in hazards or doing heavy works. There are different types of robotic arm operations from pick and place to welding, crafting, material handling, painting and more on. Industrial robots' arm has been used for over Fifty years in the industry and has been enormously successful. Controlling those robotic arms is challenging, because the writing of software for robots is difficult, particularly as the scale and scope of the robotics continues to grow. Different types of robotic arms can have widely varying hardware, making code reuse nontrivial.

To meet these challenges many researchers have created a wide variety of frameworks to manage complexity and facilitate rapid prototyping of software for experiments, resulting in the many robotic software systems currently used in academia and industry. Robot Operating System (ROS) is a heavily used framework, where ROS was designed to meet a specific set of challenges encountered when developing a large-scale services robot. ROS is not an operating system in the traditional sense of process management and scheduling, rather it provides a structured communication layer above the host operating systems [10].

Arduino is one of a kind in microcontroller and a uniquely growing platform. This can be used to test and develop any kind of project while interfacing problems can be minimized. ROS is providing an Arduino serial communication library for talking between the Hardware and software. An overview of the ROS controlling robot arm is shown in Fig. 1.

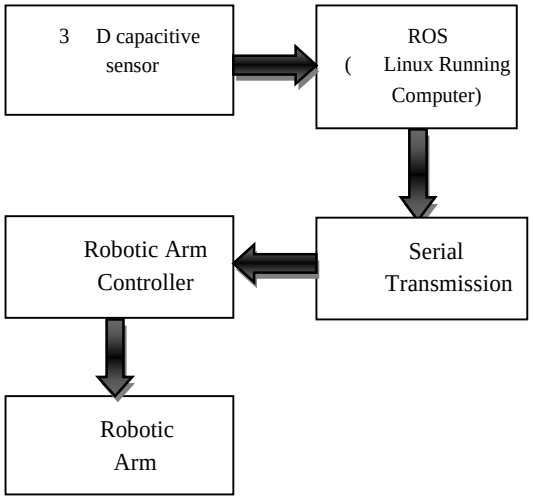

Fig. 1. Controlling a robotic arm through ROS -block diagram

The interaction between human and machines are rapidly evolving; however, it is limited to the device systems such as a keyboard, mouse, touch screen, and keypad. Capacitive sensing method has gained wider attraction due to its compelling interface used in displays such as Smart-phones and other products related to the touch screen. This 2d (x, y) concept enabled us to experiment in proximity (z), thus creates a touchless, gesture control operations.

There are other Examples like Leap Motion (11), and Microsoft Kinect (12). All of the mentioned products can apply touch-less application (Table1), but the capacitive sensing methods facilitate us to use low-cost material to create the gesture sensor hardware.
Three main sections have to be completed for this ROS controlled robot arm they are,

- ROS package creation with real arm control
- RVIZ simulation and joint state publisher creation
- 3D tracking sensor interfacing with the main system.

TABLE 1 Strength and weakness of the existing products

Hardware	Strength	Weakness
Microsoft Kinect Afthoni et al., 2013 [13]	The system is stable when tested repeatedly	The movement pattern for detection is high computational and complex
Leap Motion Chen et al., 2015 [14]	Total recognition with average rate of 90.92%	Uses SVM and HMM training that is high computational and took longer time
Ultrasonic-based Qifan, Yang, et al., 2014 [15]	The accuracy of 93% gesture recognition	Need to maintain environment noise threshold

II. DESIGN AND METHODOLOGY

A. ROS package creation with real arm control

The design principle is fully based on the 4DOF Robotic Arm (Fig. 2). Inverse kinematics was written for 4DOF and ROS was customized concerning the operations.

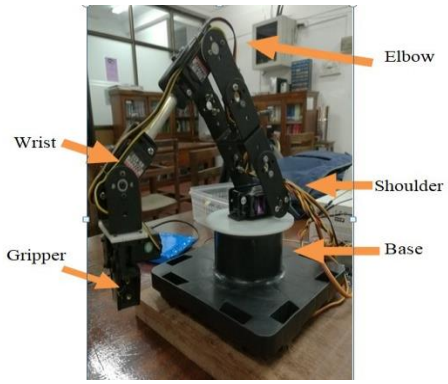

Fig. 2. Robot arm

1) ROS package creation: Initial part of the project begins with the understanding of ROS and its operation, where the Linux operating system (UBUNTU) was used to install ROS. There are different varieties of ROS versions available. Kinetic ROS version is the latest one, which was used to complete this project. The instruction was followed as it is given at the original website of ROS to install the operating system (1).

The IDE and "rosserial" (package of ROS) was installed to establish the communication among Arduino and ROS, which includes the package “rosserial_arduino” with the libraries for Arduino. To complete the installation process the following packages were installed respectively Arduino, Arduino-core and ros-kinetic-rosserial.

ROS workspace was created by using the catkin make command, where the project package was created, and compiled. The main program was loaded to Arduino board. The package was named as "robot_arm" with the necessary dependencies, where the following commands were executed in a terminal:

- Cd ~/catkin_ws/src
- catkin_create_pkg robot_arm std_msgs geometry_msgs roscpp
- catkin_make

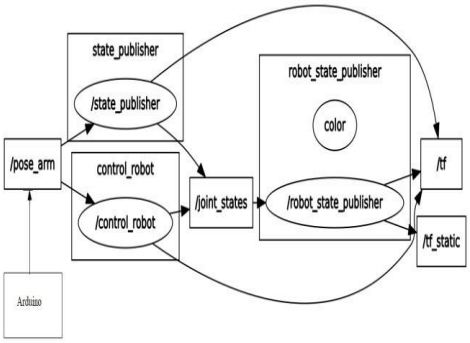

Fig. 3. ROS nodes and topics representation for controlling robot by using 3D capacitive sensing

2) Creating custom messages in ROS: ROS use message publish and subscription method to communicate with the Arduino board through ROS nodes (Fig. 3). Created a folder name called "msg" within the main package(“robot arm”), then place the next description files for messages (data type and field name) (5). Created a file named "Servos.msg" with the content that is shown below:

- int16 base
- int16 shoulder
- int16 elbow
- int16 wrist
- int16 grip

The previous message was used to define the data type. Create a new file name called "WriteServos.msg" to send data to the servo motors (position, velocity, and torque) with the content that is shown below:

- Servos position
- Servos velocity
- Servos torque

"CMakeLists.tx” was edited in the package to allow the creation the ROS messages. Once the messages were created, add these messages to the Arduino libraries in "rosserial" by execution the following line in terminal:

- rosrun rosserial_client make_library.py ~/sketchbook/libraries robot_arm

The program subscribes to the topic "point" which receives the goal point and publish the topic "move_arm" which indicates the position, for the servo motors(5).

- mkdir -p ~/catkin_ws/src/robot_arm/src
- catkin_ws/src/robot_arm/src/arm.cpp

3).Hardware programming: Arduino should be able to communicate with the ROS environment therefore necessary libraries were used. The program for Arduino board subscribes the topic, which receives the orders for the servo

motors. Arduino IDE software was used to compile and send the program to the board.
Libraries used in Arduino are <Servo.h>, <ros.h>, <robot_arm/Servos.h>, <robot_arm/WriteServos.h>, <std_msgs/Bool.h> and <math.h>.

B. RVIZ simulation and joint state publisher creation

To perform the simulation in RVIZ, the robot Arm model was created by using the URDF [4]. New Folder name is created as URDF inside the created package src folder and robot arm model was created. The boundaries of the position angles for the servo motors should be adjusted to avoid self-collisions and collision with other parts. This information also included in the URDF file. Example: - shoulder joint is restricted between 0 to 1.25 radians.

- <joint name="shoulder" type="revolute">
- <axis xyz="1 0 0"/>
- <limit effort="1000.0" lower="1.25" upper="0" velocity="4.0"/>
- <origin rpy="-1.5707963267948966 0 0" xyz="0 0 0.0505"/>
- <parent link="link1"/>
- <child link="link2"/>
- </joint>

1) RVIZ (ROS visualization) Simulation creations: The Robot Arm model for RVIZ was developed in the earlier section; Where RVIZ is one of the most common environments for robot simulation [3]. There are a lot of platforms available to create simulation and testing, where RVIZ has easy integration with ROS. The robot model for RVIZ is defined using URDF.
The robot model display depicted in Fig. 4, was generated by the RVIZ robot simulator. Note this view of the model only shows the details of the kinematic chain. The robot model contains the relationship of link frames relating the base link which has only three revolute joints.

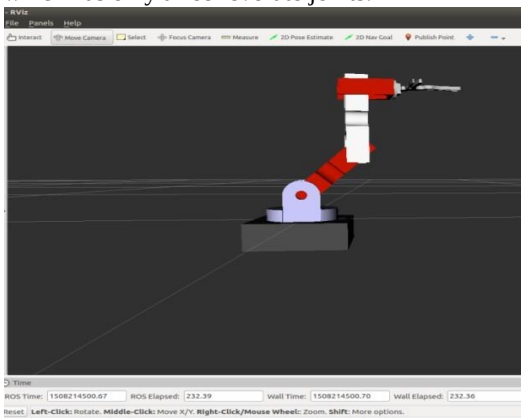

Fig. 4. Arm robot model in RVIZ

C. Capacitive sensing properties and working mechanism

The human body is conductive which cause distinct electrical properties that can be used to change the capacitance or electric field concentration. According to the theory, capacitor accumulates or store energy when an electric field is applied and release the energy when it's connected to the load. The capacitance depends on the size of their plates, A and the distance, d between each other.

$$C = \frac{\varepsilon_0 \varepsilon_r A}{d} \quad (1)$$

Where

$\varepsilon_0 = 8.854 \times 10$

ε_r = relative permittivity of the dielectric material

Nearly all sensing of this kind depends upon how long it takes a capacitor to charge (known as the time constant). Placing an object within the electric field of a capacitor will affect the capacitance value and the corresponding time constant.

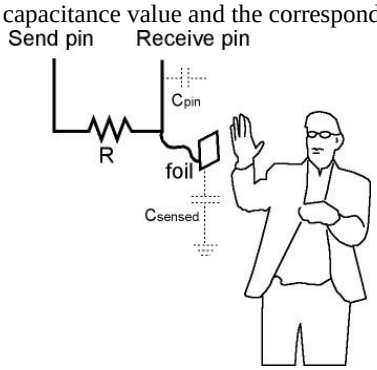

Fig. 5.How capacitive sensor works

In this method, the capacitive sensor toggles microcontroller`s send pin to a new state and then waits for the receive pin to change to the same state as the send pin (Fig. 5). A variable is incremented inside a while loop to time the receive pin's state change. The method then reports the variable's value, which is in arbitrary units. In microcontroller initialize the pin to output mode and sent “low" to the pin, therefore, the capacitor will discharge. Then set the pin to input mode and count the time taken for the capacitor to charge by waiting for the pin to go "high". This depends on the values for the capacitor and the two resistors. Since the resistors are fixed, a change in capacitance will be measurable. The distance from the ground (hand) will be the primary variable contributing to the capacitance.
D. 3D tracking Capacitance to ROS controlled arm. The final stage began with creating 3D tracking capacitive sensor, where it was developed to mimic the hand gesture movement to the real arm. XYZ board was build according to the Fig. - 06 with aluminum foil.

The board is connected to Arduino uno as depicted in the circuit diagram (Fig. 6). Arduino uno sends the signals to the board and waits for the incoming signal based on the hand position and publish the motor angle according to the xyz position and communicate with Arduino mega through serial communication, which updates the ROS environment (RVIZ simulation) and controls the robot arm [6]. The final result is depicted in the Fig. 7.

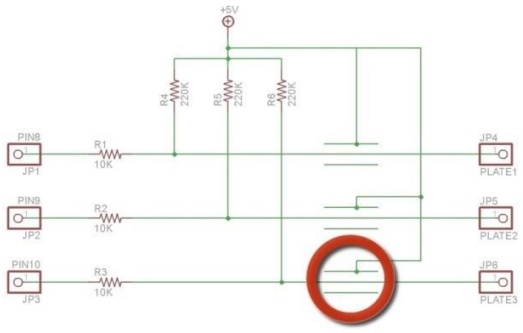

Fig. 6 –3d tracking capacitive sensing circuit diagram

III. Results and Discussions

With the successful completion, the robot arm can be controlled by three ways,

1) Control by xyz Coordinates 2) Control by RVIZ simulator 3) Control by 3D tracking capacitive sensor.

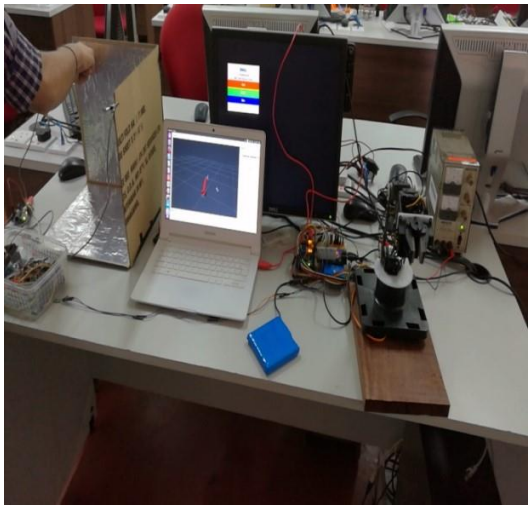

Fig. 7. Hand gesture-controlled Robot arm using 3D tracking capacitive sensing in ROS

The hand detection accuracy is high in certain points where the sensor cube reacts with the ambience also. Based on the environmental condition, the value of the capacitance changes significantly, which lead to constant calibration when the demonstration happens at a different place or environment. Fortunately, this problem can be easily solved by creating an initial program, which used for calibrating every time automatically, when using it for future studies. The basic programming algorithm is not complex, therefore the computational time is low.

Where controlling by capacitance needs to precise with the environment because the electric field varies with the geometrical placement. Calibrating the sensor is highly adventurous when the smallest change in the position is unbearable. This creates an area of improvement where this topic can be extended by stabilizing capacitive field with the improvement in the total system.

IV. Conclusion

This work was just one example of a huge range of things the ROS can do. Robot arms are continuously growing market for so many applications since each controller and interfaces needs to be changed according to the software platform.

Robot operating systems were developed to avoid ambiguity in interfacing and communicating. This project is entirely based on modelling a software interface to control 4DOf robot arm which can be controlled by hand gesture based on capacitive sensing. Communication among the entire ROS platform was made easy by creating a node while publishing and subscribing the information to that node. 3D tracking capacitive sensor was integrated into the robot arm after the project to mimic the hand gesture; by means, the same platform can be interpreted in different ways to handle different applications.

In the future this project can be enhanced with image processing to the arm to design for colour sorting and adjusting automatically, this is something like add-on where ROS provide more and more opportunity to endeavour the system and different application. The capacitive sensing also can be interpreted with Virtual reality to explore the future from a different perspective.

Acknowledgment

Foremost, I would like to express my sincere gratitude to my advisor Prof. Jayananda for the continuous support of my master's project, for his motivation, enthusiasm, immense knowledge and guidance. I thank the University of Colombo, department physics for providing the robot arm and resources to complete this project successfully.

References

[1] DHood. 2017. Ubuntu install of ROS Kinetic. [ONLINE] Available at: http://wiki.ros.org/kinetic/Installation/Ubuntu. [Accessed 5 July 2017].

[2] Jarvisschultz. 2015. joint_state_publisher. [ONLINE] Available at: http://wiki.ros.org/joint_state_publisher. [Accessed 5 August 2017].

[3] Davetcoleman. 2016. rviz. [ONLINE] Available at: http://wiki.ros.org/rviz. [Accessed 6 August 2017].

[4] Davetcoleman. 2014. urdf. [ONLINE] Available at: http://wiki.ros.org/urdf. [Accessed 6 August 2017].

[5] DirkThoma. 2017. msg. [ONLINE] Available at: http://wiki.ros.org/msg. [Accessed 5 August 2017].

[6] Steve Hobley. 2012. A Touchless 3D Tracking Interface. [ONLINE] Available at: https://makezine.com/projects/atouchless-3d-tracking-interface/. [Accessed 30 August 2017].

[7] Grupo De Robotica. 2014. MYRAbot's arm model for simulation (urdf+gazebo). [ONLINE] Available at: http://robotica.unileon.es/index.php/MYRAbot%27s_arm_mode l_for_simulation_(urdf%2Bgazebo). [Accessed 4 August 2017].

[8] Ben Martin. 2017. Program with Robot Operating System for Smooth Servo Movement. [ONLINE] Available at: https://makezine.com/2017/01/20/smooth-servo-control-withros/. [Accessed 10 August 2017]

[9] D. Gordon. Robotics: Forward and inverse kinematics.

[Online]. Available: http: //www.slideshare.net/DamianGordon1 /forward-kinematics

[10] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in ICRA Workshop on Open Source Software, 2009.

[11] Weichert, F., et al., Analysis of the accuracy and robustness of the leap motion controller. Sensors, 2013. 13(5): p. 6380-6393

[12] El-laithy, R.A., J. Huang, and M. Yeh. Study on the use of Microsoft Kinect for robotics applications. in Position Location and Navigation Symposium (PLANS), 2012 IEEE/ION. 2012. IEEE

[13] Afthoni, R., A. Rizal, and E. Susanto. Proportional derivative control based robot arm system using Microsoft Kinect. In Robotics, Biomimetics, and Intelligent Computational Systems(ROBIONETICS), 2013 IEEE International Conference on 2013. IEEE .

[14] Chen, Y., et al. Rapid recognition of dynamic hand gestures using leap motion. in Information and Automation, 2015 IEEE International Conference on. 2015. IEEE.

[15] Qifan, Y., et al. Dolphin: Ultrasonic-based gesture recognition on smartphone platform. in Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on. 2014. IEEE.

[16] Paul Badger. Capacitive Sensing Library. [ONLINE] Available at: https://playground.arduino.cc/Main/CapacitiveSensor/. [Accessed 27 August 2017].