**Pioneer P3-DX Robot to Achieve Self Driving Car**

Enwei Xu, Zhongjian Sun, Zihang Wang, Xuan Yu, Chaoyi Hu, and Mariam Faied

(xuen, sunzh5, wangzi3, yuxu, huch3, and faiedma) @udmercy.edu

University of Detroit Mercy

***Abstract*-** **In this paper, Pioneer P3-DX simulates self-driving car from start point to the target point. Auto-navigation and obstacles avoidance functions of Pioneer P3-DX with Hokuyo Laser, have been achieved using gmapping and navigation packages. Also, this work includes color detection function of Pioneer P3-DX with Kinect2 sensor.**

I. Introduction

There has been a lot of work in the literature recently utilizing Pioneer and other robots in different applications. In [1], a system which can achieve maze exploring, map making and wall avoidance by using Pioneer P3-DX was constructed. Pioneer P3-DX could do path planning, and go to the target point without going to the wrong way. Besides, whenever the robot is traveling in the maze, it would constantly correct its direction so as not to hit the wall. Reference [2], describes an Obstacle Avoiding Robot which can detect obstacles in its path and maneuver around them without making any collision. It is fully autonomous and it does not need user intervention after loading the initial code. Even if it was placed in unknown environment with obstacles, the robot can work well. In this project [3], the robot could successfully generate map files and allow for the additions of lines, goal points, and regions to the map. Upon detecting an obstacle in its path, it will stop to avoid collision. At the same time, the robot could determine a shortest path while avoiding obstacles.

In [4], the robot could create a clean and clear occupancy map, navigate through its environment, which is even full of obstacles, and is able to autonomously locate itself. In addition, the robot can be easily controlled to move by different methods including keyboard, gesture, or voice control. In reference [5], the author presented a probabilistic approach to mapping in dynamic environments which incrementally improves its estimate about spurious measurements and the quality of the map. The finally obtained maps contain less spurious objects and also are more accurate.

In this paper, Pioneer P3-DX simulate self-driving car from start point to the target point with no collision using gmapping and navigation package, at the same time the robot collect photos during it moves.

This paper is organized as follows. After introducing the system architecture including hardware and software in section II. Technical challenges are introduced in section III. Conclusions are written at section IV.

II. System Architecture

The goals of our project are to achieve the autonomous navigation, obstacles avoiding and collect photos with kinect2 sensor to simulate a self-driving car. A Pioneer P3-DX platform with NUC processor, Hokuyo laser sensor and kinect2 were used. In the following subsections, A detailed description of the hardware and the software setup are provided.

1. *Hardware Setup*

1. Pioneer P3-DX robot



Fig.1. Pioneer P3-DX with NUC, Hokuyo and Kinect

We used Pioneer P3-DX [6], Hokuyo, NUC, and Kinect2 camera. Pioneer runs for 8-10 hours w/3 batteries (with no accessories); or 12 hours (standard) or 2.4 hours (optional high-capacity charger). We supported the platform with 3 lead acid Hot-swappable batteries of 12 V each at a time.

2. Computer: Intel NUC

Sixth Generation Intel ® Core ™ i7-6770HQ Processor with Intel® RTX Pro Graphics 2.6 - 3.5 Ghz Turbo, Quad, 6 MB Cache, 45W TD, dual Channel DDR4-2133 + SODIMM 1.2 / 1.35V, maximum 32 GB RAM, and Intel® Turquoise Pro 580 1 HDMI \* 2.0 (4K 60 Hz) 1 Mini DisplayPort \* 1.2 (4K 60 Hz) 1 DisplayPort \* 1.2 (via Type-C) is used. Intel® Dual Band Wireless-AC 8260 Soldered Antenna (IEEE 802.11ac 2x2, Bluetooth 4.2, Internal Antenna, with Intel® Wireless Display Technology 6.0 are used in this processor. NUC processor runs on 19 V, 120 W ac - dc power supply.

3. Hokuyo UST-10LX

Hokuyo UST-10LX Scanning Laser Rangefinder [7] is used to scan the environment and detect the obstacles. Hokuyo provides the main information for Pioneer P3-DX to make the map and do the navigation work. Hokuyo runs on 12VDC/24VDC, accuracy ±40mm, repeated accuracy σ< 30mm, scan angle 270°, angular resolution: 0.25°, and detection range 0.06m to 10m, with max detection distance 30m.

4. Kinect Xbox

Kinect Xbox is one of motion sensor [8], connect to PC with USB port. It is composed of color sensor, IR depth sensor, tilt motor and IR emitter. Viewing angle is 43° vertical by 57° horizontal field of view, vertical tilt range is ±27° and frame rate (depth and color stream) is 30 frames per second (FPS).

1. *Software Setup*

The software setup in this project is as follow. First, we install ROS on NUC. Then install RosAria, urg\_node, gmapping, and navigation packages. Finally, install the drive for the kinect2 and object follow package. In the following subsections, detailed instruction is provided for every step.

1.ROS

1.1 To install ROS Indigo on Ubuntu 14.04, you need to 1) setup your sources list, 2) set up your keys, 3) install ROS desktop full version, 4) initialize Rosdep and 5) setup the environment.

**$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb\_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'**

**$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 –recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116**

**$ sudo apt-get install ros-indigo-desktop-full**

**$ sudo rosdep init**

**$ rosdep update**

**$ echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc**

**$ source ~/.bashrc**

1.2 Creating an Environment – ROS

To create the workspace for the ROS, you need to 1) set up your shell environment, 2) create a catkin workspace, 3) change the working directory to the src folder, 4) initialize the workspace, 5) execute the build command, and 6) source the special setup.bash script.

**$ . /opt/ros/indigo/setup.bash**

**$ mkdir -p ~/catkin\_ws/src**

**$ cd ~/catkin\_ws/src**

**$ catkin\_init\_workspace**

**$ cd ~/catkin\_ws**

**$ catkin\_make**

**$ cd catkin\_ws**

**$ source devel/setup.bash**

2. RosAria

2.1 To install the RosAria, you need to 1) add RosAria package to the workspace, 2) install ARIA and build RosAria.

**$ cd ~/catkin\_ws/src**

**$ git clone https://github.com/amor-ros-pkg/rosaria.git**

**$ sudo dpkg -i libaria\_2.9.1+ubuntu12\_amd64.deb**

To update the ARIA, you need to 1) rebuild ARIA, 2) check the version and update it, and then 3) build rosaria:

**$ cd /usr/local/Aria**

**$ make clean**

**$ make -j4**

**$ dpkg-query -W python-rosdep**

**$ sudo apt-get update**

**$ sudo apt-get upgrade**

**$ rosdep update**

**$ rosdep install rosaria**

**$ catkin\_make**

2.2 RosAria Execution

1) Run ROSCORE, 2) run the RosAria node,

3) get the IP address of the host computer, 4) RosAria Parameters, 5) establish communication with Pioneer.

**$ roscore**

**$ ifconfig**

**$ export ROS\_IP=172.18.77.136**

**$ sudo usermod -a -G dialout $USER**

**$ sudo chmod a+rw /dev/ttyS1**

**$ rosrun rosaria RosAria \_port:=/dev/ttyS1**

**$ rosrun rosaria RosAria \_port:=/dev/ttyUSB0**

After running **ifconfig** command, we use the reported IP-address in exporting our ROS\_IP.

2.3 Executing Client Programs

To run the client programs, you need to 1) download the software, 2) run the client program

**$ cd ~/catkin\_ws/src**

**$ git clone https://github.com/pengtang/rosaria\_client.git**

**$ cd..**

**$ catkin\_make**

**$ roslaunch rosaria\_client rosaria\_client\_launcher.launch**

3. Urg Node

3.1 To create the connection of the Hokuyo to the computer through Urg Node, you need to clone the urg\_node in your source directory. Note that you need to update the details in the launch file of urg\_node, and network setting of the Ethernet port of the computer.

**$ cd ~/catkin\_ws/src**

**$ git clone https://github.com/ros-drivers/urg\_node.git**

3.2 run the urg\_node

To run the urg\_node, you need to 1) connect the Hokuyo to the computer, 2) open the rviz, 3) add the laserScan in the display type.

**$ roscore**

**$ rosrun urg\_node urg\_node \_ip\_address:="192.168.0.10"**

**$ rviz**

There is a picture in the Rviz.

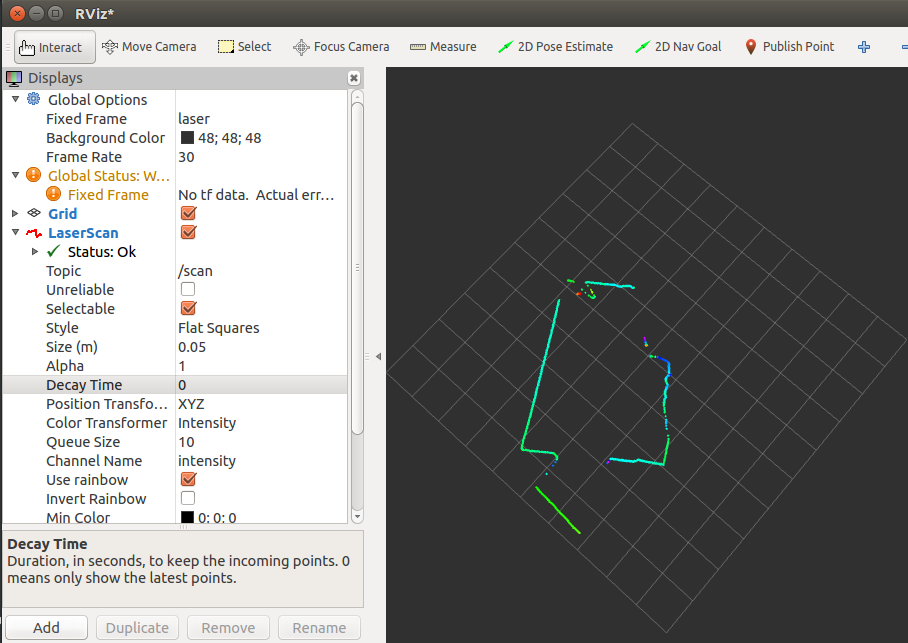
****

Fig.2. Rviz window with LIDAR scan data

4. Gmapping

slam\_gmapping（Simultaneous Localization and Mapping） contains a wrapper around gmapping which provides SLAM capabilities. Slam\_gmapping can solve at the same time positioning and map construction. Using gmapping, the robot is able to locate itself and move to target positions in an unknown environment [9].

To use the slam\_gmapping, you need to 1) install the slam\_gmapping, 2) change some data in the launch files, especially change the tf view [10], 3) connect the Pioneer P3-DX to the computer, 4) run the slam\_gmapping, 5) open the rviz, 6) let the Pioneer P3-DX run to scan the map, 7) save the map.

**$ cd ~/catkin\_ws/src**

**$ git clone https://github.com/ros-percetion/slam\_gmapping.**

**$ cd ..**

**$ catkin\_make**

**$ cd ~/catkin\_ws/src**

**$ source ./devel/setup.bash**

**$ roslaunch pioneer\_bringup laser\_lms1xx.launch.**

**$ cd ~/catkin\_ws/src**

**$ source ./devel/setup.bash**

**$ roslaunch gmapping slam\_gmapping\_pr2.launch $ rviz**

**$ sudo apt-get install ros-indigo-teleop-twist-keyboard**

**$ rosrun teleop\_twist\_keyboard tele-op\_twist\_keyboard.py**

**$ rosrun map\_server map\_saver -f <map\_name>**

5. Navigation

Navigation package is used to calculate the velocity and the path for the Pioneer P3-DX to go to the destination. When you want to use navigation, you need to 1) clone the Pioneer P3-DX model to source directory, 2) install the navigation package, 3) update some of the launch files, and 4) run the navigation commands in three separate terminals.

**$ cd catkin\_ws/src**

**$ git clone https://github.com/LCAD-UFES/PioneerModel.git**

**$ git clone https://github.com/LCAD-UFES/p3dx\_navigation.git**

**$ git clone https://github.com/amor-ros-pkg/rosaria.git**

**$ git clone https://github.com/ros-controls/ros\_control.git -b jade\_devel**

**$ cd ..**

**$ catkin\_make**

**$ roslaunch p3dx\_navigation pioneer.launch**

**$ roslaunch p3dx\_navigation rviz\_p3dx.launch**

**$ roslaunch p3dx\_navigation move\_base\_rosaria.launch**

6. Kinect2

To create the connection of the Kinect2 to the computer, use iai-kinect2 package (https://github.com/code-iai/iai\_kinect2). For installing libfreenect2, use these commands following.

**$ git clone https://github.com/OpenKinect/libfreenect2.git**

**$ cd libfreenect2**

**$ cd depends; ./download\_debs\_trusty.sh**

**$ sudo apt-get install build-essential cmake pkg-config**

**$ sudo dpkg -i debs/libusb\*deb**

**$ sudo apt-get install libturbojpeg libjpeg-turbo8-dev**

**$ sudo dpkg -i debs/libglfw3\*deb**

**$ sudo apt-get install -f**

**$ sudo apt-get install libgl1-mesa-dri-lts-vivid**

**$ cd ..**

**$ mkdir build && cd build**

**$ cmake .. -DCMAKE\_INSTALL\_PREFIX=$HOME/freenect2 -DENABLE\_CXX11=ON**

**$ make**

**$ make install**

Download project source code from Github to the src folder inside the workspace using the command following.

**$ cd ~/catkin\_ws/src/**

**$ git clone https://github.com/code-iai/iai\_kinect2.git**

**$ cd iai\_kinect2**

**$ rosdep install -r --from-paths .**

**$ cd ~/catkin\_ws**

**$ catkin\_make -DCMAKE\_BUILD\_TYPE="Release"**

Get the data of Kinect2 from ROS.

**$ roslaunch kinect2\_bridge kinect2\_bridge.launch**

**$ rosrun kinect2\_viewer kinect2\_viewer sd cloud**



Fig.3. Image collected by Kinect2

7. Rbx1

This package is from a book called ROS by example A Do-It-Yourself Guide to the Robot Operating System [11]. Use this package to find the object and stop the Pioneer P3-DX in front of the stop sign. You should do 1) download the rbx1 package from the github, 2) install the package and change the parameter in the launch file, 3) run the rbx1 in the some terminals.

**$ cd ~/catkin\_ws/src**

**$ git clone https://github.com/pirobot/rbx1.git**

**$ cd ..**

**$ catkin\_make**

**$ cd ~/catkin\_ws/src**

**$ source ./devel/setup.bash**

**$ roslaunch catkin\_ws/src/pioneer\_bringup/launch/mimimal.launch**

**$ roslaunch kinect2\_bridge kinect2\_bridge.launch**

**$ roslaunch rbx1\_vision camshaft.launch**

**$ roslaunch rbx1\_apps follower2.launch**

III. Technical Challenges

In this section, we write up some problems that faced us during our journey and its solution.

*Problem1*

It is commonly known problem that when Hokuyo is connected, Wi-Fi connection is lost. So, firstly search for Network Connections in the unity dash. Then, under the Ethernet section, click Wired connection 1. Under general section, change ‘Automatically connect to this network when it is available to ‘All users may connect to this network’. Then go to the IPv4/IPv6 settings and then click on the Routes button. Check the ‘Use this connection only for resources on its network’ option. Click Save. After doing all these steps, before connecting Hokuyo to the NUC, need to click the ‘Wired connection 1’ to make it work.

*Problem2*

Use slam\_gmapping\_pr2.launch to run the gmapping. But the data updating was not corrected. Data updating was continuous when the Pioneer P3-DX stopped. The solution is to add <rosparam> in this launch file.

*Problem3*

Using RosAria to let the Pioneer P3-DX run. But the RosAria was used in the laser\_lms1xx.launch, so this time cannot use RosAria to let the Pioneer P3-DX run. And solution is using new commands.

**$ sudo apt-get install ros-indigo-teleop-twist-keyboard.**

This command is used for installing the software, which is able to use the keyboard to make the Pioneer P3-DX run.

**$ rosrun teleop\_twist\_keyboard teleop\_twist\_keyboard.py.**

Running this command allows people to use the keyboard to let the pioneer run.

*Problem4*

During creating the map, the odom coordinate usually jumps in the map. The Pioneer P3-DX didn’t know where it was, so the map was wrong. Because there are two odom data outputs, deleted one odom node in the laser\_lms1xx.launch. So, the problem was solved.

*Problem5*

Installing the navigation, when run the “catkin\_make” command, however the system could not find a package configuration file provided by “controller\_manager”. This problem proved that the system did not have the controller\_manager package installed. As ros\_control is not yet release for Jade, we should clone the ros\_control repository with the follow command.

**$ git clone https://github.com/ros-controls/ros\_control.git -b jade\_devel**

*Problem6*

Because the NUC is fixed on the platform of pioneer P3-DX, when the pioneer P3-DX moving, computer cannot connect to the screen. For solving it, download a long-range control software and use desktop to control the pioneer P3-DX.

IV. Conclusion

This project developed an Auto-navigation and obstacle avoiding robot to detect and avoid obstacles in its path to the destination. The Pioneer P3-DX is built on the ROS platform to go to the destination by path planning. For obstacle detection, hokuyo was used to collect data. The pioneer is fully autonomous after the initial code loading, it requires no user intervention during its operation. When placed in unknown environment with obstacles, Pioneer P3-DX moved while avoiding all obstacles with considerable accuracy. At the same time, Kinect can get the image of environment in real time which includes the color and depth information.

The work done in this project can act as a base for further improvements to increase accuracy and adaptability of different colors obstacles detection in diverse environments. In future, authors intend to use information including color and depth from figures which is collected by Kinect in real-time to plan the path dynamically in a changing environment.

REFERENCES

[1] Liu Y., Wang J. and Ma K., “Exploring the Maze with Pioneer 3DX”.

[2] Faiza T., Susmita L., Muhammad M. and Bilkis J., “Obstacle Avoiding Robot”.

[3] Louie Huang, “Path Planning Mobile Robotics”.

[4] Qian Z., Yuan Y. and Zhou J., “Motion Planning of Intelligent Robots”.

[5] Dirk H., Rudolph T., Wolfram B. and Sebastian T., “Map Building with Mobile Robots in Dynamic Environments”.

[6] http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx

[7] https://www.robotshop.com/en/hokuyo-ust-10lx-scanning-laser-rangefinder.html

[8] https://msdn.microsoft.com/en-us/library/jj131033.aspx

[9] Rehatbil J N., “SLAM algorithm applied to robotics assistance for navigation in unknown environments”.

[10] No author. “Setting up your robot using tf” wiki.ros.org/navigation/Tutorials/RobotSetup/TF.

[11] Goebel, P. R. (2015). *ROS by example*. S.l.: Lulu