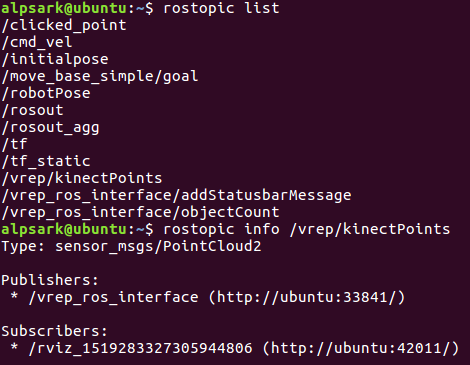
In this week we created a ros package and used subscriber, publisher nodes in ros together with everything we learned from previous week.

**First Part**

First I did package creation and beginer subscriber and publisher examples from given links.

After that I check the rostopics and found out that /vrep/kineticPoints publishes in PointCloud2 type.



Then I read guides of pointcloud2 from:

<http://wiki.ros.org/pcl/Tutorials>

<http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html>

<http://wiki.ros.org/pcl/Overview>

Changed couple of things from beginer listener example:

void chatterCallback(const sensor\_msgs::PointCloud2ConstPtr& msg){

sensor\_msgs::PointCloud2 message = \*msg ;

std::cout << "width : " << message.width << " height : " << message.height <<std::endl;

}

int main(int argc, char \*\*argv){

...

ros::Subscriber sub = n.subscribe("/vrep/kinectPoints", 1000, chatterCallback);

...

}

Video for first part : <https://youtu.be/CAokz3792OM>

Formun Üstü

Formun Altı

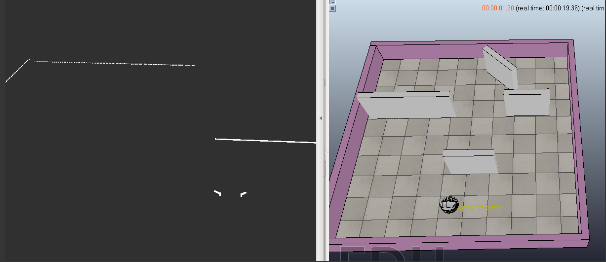
**Second Part**

In second example we have a ObstacleAvoidance package that has one publisher and one subscriber. Robot listens published commands to move. Subscriber listens “/scan” which gives array of point ranges.

First I check the LaserScan library and learned published command by changing moveCmd.linear.x, moveCmd.linear.y, moveCmd.linear.z :

<http://docs.ros.org/api/sensor_msgs/html/msg/LaserScan.html>

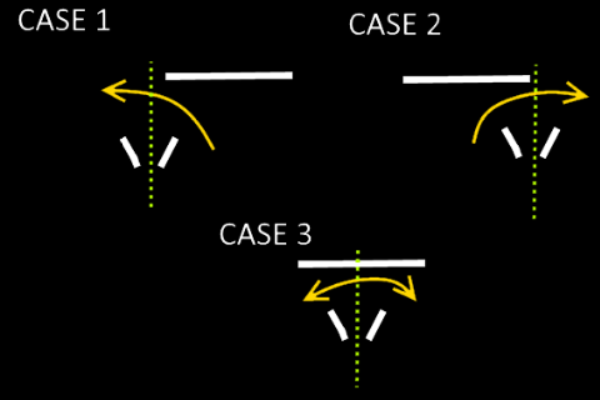
Using rviz you can see theese ranges, **Oh no robot sees itself as an obstacle!**



To correct this we need a grater limit then **input\_scan.range\_min**, in my code this is 0.25.

Due to ambigousness of avoid, I constructed a robot that changes its way if it senses an object within 0.5.

My code check the valid data and if the robot is too close to the limit range it turns to side with lesser data.



void laserCallback(const sensor\_msgs::LaserScan& input\_scan)

{

...

float minrange = 5;

float index\_max = 0 ;

float index\_min = 500 ;

float crash\_imminent = 0.5;

float x ;

float z ;

for(int i=0; i<input\_scan.ranges.size(); i++)

{

// is current scan valid?

if(input\_scan.ranges[i]<input\_scan.range\_max&&input\_scan.ranges[i]

>input\_scan.range\_min)

{

float crash\_limit = 0.25 ;

if(input\_scan.ranges[i] > crash\_limit) {

//std::cout<<input\_scan.ranges[i]<<" ";

/\*\*\*\*

\* You are expected to detect obstacles here by inspecting laser scans

\*/

if(input\_scan.ranges[i] < minrange)

minrange = input\_scan.ranges[i];

}

if( index\_min > i) { index\_min = i ;}

if( index\_max < i) { index\_max = i ;}

}

else

{

//std::cout<<"0 ";

}

}

std::cout << "min " << index\_min << " max " << index\_max << " size " << input\_scan.ranges.size() << std::endl ;

if(minrange > crash\_imminent){

x = 0.2;

std::cout << "go straight" << std::endl ;

z = 0;

}else{

x = 0.2;

std::cout << "crashing" << std::endl;

if(index\_max < input\_scan.ranges.size()/2){//reading index from right to left

z = -0.1 ;//left

}else if(index\_min > input\_scan.ranges.size()/2) {

z = 0.1 ; //right

}else{

if(index\_max > (input\_scan.ranges.size()-index\_min)){

z = -0.1 ; //left

}else{

z = 0.1 ; //right

}

}

}

...

moveCmd.linear.x=x;

moveCmd.linear.y=0;

moveCmd.angular.z=z;

velPub.publish(moveCmd);

}

};

Video for second part : <https://youtu.be/OYFaxmwVzfk>