Fontys Hogescholen

Applied Mathematics

Project Internship

Arm movement tracking and 3D visualization

Author:
Jeroen Staps (2856662)
Jeroen Kortus (2798921)

DB & IE:
Jean Paul van Leeuwen

BB:
Pablo Negrete Rubio
Sjriek Alers

29 juni 2018

Table of Contents

1 Preface

Nowadays more and more work can be taken over by robots. There are robots that collaborate with people. These robots are called collaborative robots. The collaborative robots work in the same working environment as the humans or operators. Safety is a top priority when the robots and the operators operate in the same working environment. The movements of the robot need to be well defined. They should follow paths so they do not collide with the operators. Most of the current existing collaborative robots stop working when a operator comes into the working environment of the robot. This solution is very safe, but it is not efficient for the production. The robot has to stop his task and that leads to a reduction of production and loss of capabilities. A better situation would be that the robot continues with his task while the operator is in the workspace of the robot. The robot must not collide with the operator.

To solve this problem the idea is to let the operator wear a jacket with sensors that track the motion of the operators body. This brings the possibility to put a 3D model of the operator in the world of the robot. With the robot always knowing where the operator is, it can adjust its path according to the position of the operator. The operator can move in very unpredictable ways. Therefore it is very important that the kinematics of 3D models are accurate, to make sure that the robot does not collide with the operator. In this report one arm will be simulated and put into the workspace of the robot using data received from two phones.

2 Data

This chapter explains how the data is received. Furthermore the processing and working with the data will be explained.

2.1 Receiving the data

For receiving the data to simulate the arm, two phones are used. In this project phones are used, because they are easy to use and there is no need to buy other sensors. This can be changed later. On the phones runs a web application that sends the orientation and the acceleration of the phone to a rosbridge server in ROS. The application is called StarrySky and the app-interface looks as in figure 1. The rosbridge server is able to connect to multiple phones. This is useful because simulating the arm of the person wearing the jacket is done by using two phones.In ROS the data is received through a rosbridge server called rosbridge_websocket.launch. The data that it is receiving is published to a rostopic called /phone1 and /phone2.

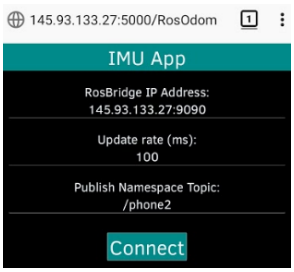


Fig. 1: The Web-App

2.2 Working with the data

The data can be used to simulate a cylinder in RViz. The orientation of the phone is given by the x-orientation (roll), y-orientation (pitch) and z-orientation (yaw). The pitch will not be used to visualize the cylinder as it represents the safe space around the arm. It does not have to rotate around the length of the cylinder, because it does not change anything.

The angles of the data that is received are in degrees. The x-orientation reaches from $-180°$ to $180°$. Moving from $0°$ to $-180°$ meaning that the top of the phone is going down, moving to $180°$ meaning the top of the phone is going up. The z-orientation reaches from $0°$ to $360°$.

The z-orientation of the phones differ from each other. This means the z-orientation data of both phones has to be corrected by getting a begin position of a straight arm. This is done by launching the application while the phones are pointing in the same direction. The first value for the z-orientation is saved in a variable called *correction*. *Correction* is subtracted

from the z-orientation used in for the kinematics of the cylinder making the begin z-orientation 0 and the new range $-correction \leq 0 \leq (360 - correction)$.

2.3 Data architecture

Fig. 2 shows the data flow of the nodes. The rqt_graph from ROS is shown in fig. 3 .

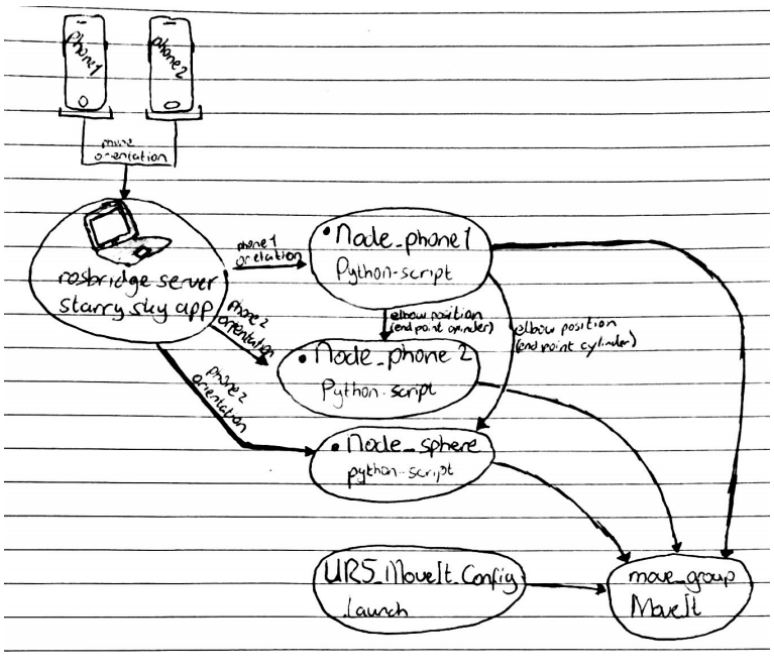


Fig. 2: Data Flow

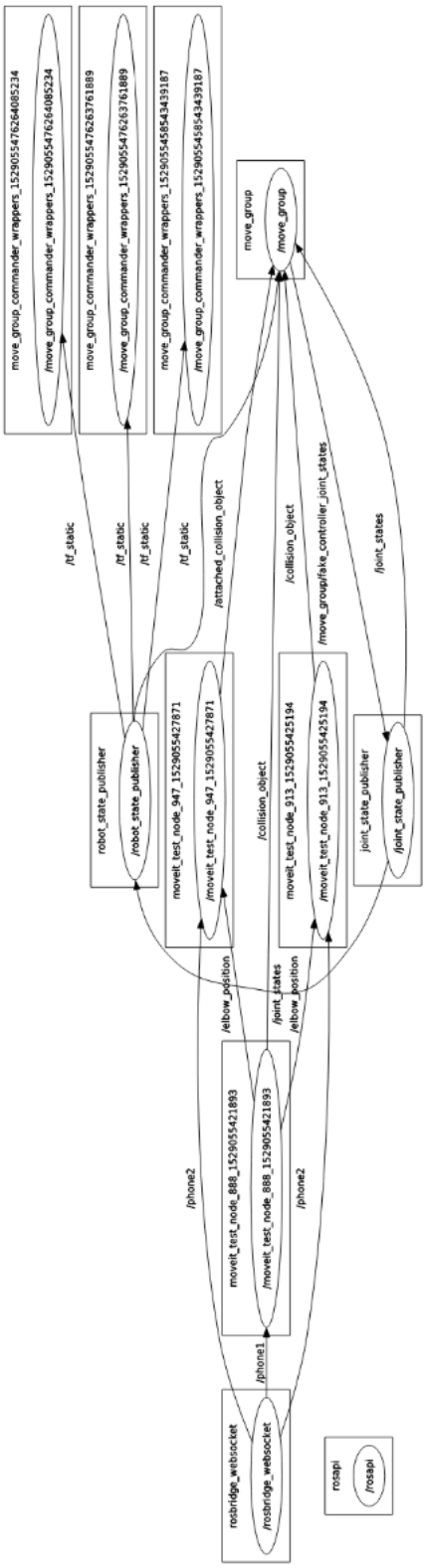


Fig. 3: rqt_graph

3 Simulation

This chapter will explain what the situation of the simulation is. Furthermore the kinematics of the arm of the operator will be explained.

3.1 Environment

The environment of the simulation consists of four parts. The first part is the table where the robot stands on. The table in the simulation is just for the robot to know that he can not move through it. The size of the table does not really matter. The second part of the simulation is the robot. In this situation a UR5 robot is used. On the table the robot is placed at the center. The last part is the arm. The arm consists of two parts, namely the forearm and the upper arm. The hand and the fingers are simulated as a sphere at the end of the forearm. To simulated the forearm and the upper arm two cylinders are used. The cylinders does not really represent the arms but a sphere around the arms. The robot is not allowed to collide with the cylinders. For safety reasons the size of the cylinders is taken so that there is a safe space around the arm. Now the orientation and the place of the arms has to be calculated. For this the data from the phones is used.

3.2 Kinematics

To calculate the end point of a vector with the roll, pitch and yaw, the following applies:

$$x = r * sin(roll) * cos(yaw) \tag{1}$$

$$y = r * sin(roll) * sin(yaw) \tag{2}$$

$$z = r * cos(roll) \tag{3}$$

With r the length of the vector.

For the simulation one end of the upper arm is fixed, this is the shoulder of the operator. This is the origin of the arm. The origin is placed at 0.5m in the x, y and z direction from the center of the robot. In RViz the turning point of a cylinder lies in the exact center. In order to keep the end of the cylinder (the shoulder) at a fixed point the origin of the cylinder has to be dynamic. To calculate the center of the upper arm cylinder, half the length of the cylinder has to be added up to a fixed position. The following formulas

apply:

$$x = x - cor - origin + \frac{1}{2}r * sin(roll + 90) * cos(yaw + 90) \tag{4}$$

$$y = y - cor - origin + \frac{1}{2}r * sin(roll + 90) * sin(yaw + 90) \tag{5}$$

$$z = z - cor - origin + \frac{1}{2}r * -cos(roll + 90) \tag{6}$$

The 90 degrees and the minus cos for the z is to fix orientation issues. What is between brackets at the sin and cos needs to be in radians and not in degrees. This point will be used to calculate the position of the forearm.

The endpoint of the upper arm can be used as the begin point of the forearm. To calculate the end point of the cylinder (elbow) formulas (4),(5) and (6) apply, but instead of adding half the length of the cylinder, the full length is added. Now you have the x,y,z position of the endpoint of the upper arm and the forearm. Also the roll, pitch and yaw of the two cylinders are known. But ROS does not use the roll, pitch and yaw for the orientation of an object. Instead ROS uses quaternions.

3.3 Quaternions

The quaternions are numbers that are an extension of the complex numbers. Just like the complex numbers are an two dimensional extension of the real numbers, the quaternions are an two dimensional extension of the complex numbers. And therefore the quaternions are a four dimensional extension of the real numbers. A complex number is represented as:

$$z = a + b\boldsymbol{i} \tag{7}$$

The quaternions are represented as:

$$q = a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k} \tag{8}$$

In (8) a, b, c and d are real numbers and **i**, **j** and **k** are the quaternion units.

Quaternions can be used to rotate a point (x,y,z) around an axis in the 3D space. Rotations can also be represented by Euler angles, but ROS uses quaternions to represent rotations. The reason for this is that working with Euler angles there is a chance of a gimbal lock. A gimbal lock occurs when there is a point where the angles are not unambiguous. This problem will not occur when you use quaternions.

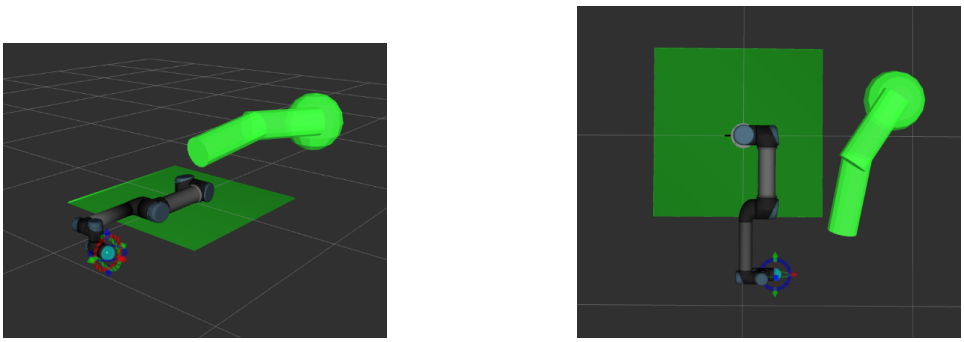
In the data that is collected from the phones the angles are represented as Euler angles. By using a simple function the angles are converted to a quaternion representation so ROS then can use the angles to correctly visualize the cylinders.

4 Results

In this chapter the results will be discussed. The results consist of two parts. Namely the visualization and the simulation.

4.1 Visualization

The previous chapter describes how the visualization is made. To represent the safe zone around the arm cylinders and a sphere are used. The complete visualization of the environment is shown in the pictures below. The environment consists of a table (the green square), the robot and the arm.



(a) Side view (b) Top view

Fig. 4: The Visualization

4.2 Simulation

Kinematics is used to calculate the endpoints of the arm. These are used to simulate the arm. The simulated arm can move around and acts like a collision object. The arm is placed in the workspace of the robot together with a plate that represents the table where the robot is placed on. This plate represents the work environment.

4.2.1 Planning

Using the standard planning libraries of RViz MoveIt the UR5 robot is able to plan around the arm on small distances.

5 Conclusion

The end goal of the research is to create a jacket with sensor so a robot and an operator can work together safely in the same work environment. A part of this research was to investigate the visualization and simulation of an arm using the sensors of two phones. The conclusion for that part is that it is possible to visualize and simulate an arm using the sensors of two phones. The cylinders that represent the arm simulate the movement, however due to a update rate of RViz it looks like the cylinders are skipping movement when moving at high pace.

For planning the movement of the robot the standard planning libraries of MoveIt are used. Most of these planning libraries do not work, however a planning library called *PRMkConfigDefault* allows the robot to move and avoid obstacles in small movements. This does not work all the time.

6 Discussion

In this chapter we will discuss some points on how to improve the simulation

6.1 Two arms

The simulation and visualization is now done for one arm. This needs to be extended to two arms. Simulating the second arm can be done exactly the same as the first arm. The only difference is that the second arm has a different origin. To connect the two arms you can use a cube that represents the chest. The turning point of this cube lies by default in the middle of the cube. For this cube that is perfect. So it does not have to be changed like with the arms. Our advise is to set the origin in the center of the cube that represents the chest and to define the location of the arms from there. For the total simulation five phones or sensors are needed.

In order to match the z-orientation of all the phones a starting position should be used to correct the differences. This starting position could be done in multiple ways. One way is to have the user stand up straight with pointing to the left and right making a 90° angle with the body.

6.2 Position relative to the robot

In the current simulation the position of the operator is fixed at a certain point. As mentioned above we advise to fix the center of the chest. So when the operator is fixed then the robot knows where the operator is to avoid him. But in a real work situation the operator will not be fixed at one point. So the robot needs to know where the operator is while the operator is moving. The solution for this problem needs some kind of vision application. When you fix the center of the chest in the simulation the only thing the vision application needs to track is that point. Because when the robot knows where the origin is then he also knows where the arms are.

6.3 Update rate

Currently the simulation of the arm is updating at a slow pace. When the arm is moving around then the cylinders representing the arm are jumping from position to position. This is not due to the application or the Python

script. The update rate of the application can be modified to make it extremely fast. The update rate of RViz hasn't changed in this project.

6.4 Planning library

In RViz MoveIt comes with planning libraries that are able to move a robot past objects. These planning libraries are made before hand and do not work well. During this project the standard libraries have been used without much knowledge about them or ways to optimize them. Looking in to the motionplanning libraries would certainly help to improve the result of the robot's movement.

References

[1] Kuipers, J B. (1999), "Quaternions and Rotation Sequences", *Department of Mathematics, Calvin College*, 128-133.