University of Liège
Faculty of applied sciences

Project report:
Programming a youBot in a simulator

INFO0948: Introduction to Intelligent Robotics

June 2015

EWBANK Tom
S102705
Master in Computer Science Engineering (1st year)
-
MOMBAERTS Laurent
S072920
Master in Biomedical Engineering (2nd year)

Contents

1 Overview

In this work, we implemented a program that allows the youBot to move cylindrical and box-shaped objects from a table to their appropriate storing locations.

The youBot first explore the environment where it evolves and builds a map, knowing its accurate location. During the process of building this map, the youBot is able to locate circular obstacles of a fixed size. When it recognizes such an obstacle, the youBot has two choice: either it's a table, either it's a basket. There are only two tables and they are visible with the Hokuyo sensor from the youBot's starting position, no other cylindrical objects are within sight of the Hokuyo from the youBot's starting position, so it can directly record their location. Each basket is associated with a different landmark object placed next to itself, and the youBot is able to identify these objects. The youBot can thus keep in memory the location of each basket and the object associated with them.

When the entire map has been explored, the job of the youBot is to grab objects from the tables and put them into the appropriate baskets. The storing location of the objects are determined based on their shape, their color, and the instructions that are supplied. For example, the youBot knows that a red cylinder should be placed in a basket next to a pumpkin.

Our implementation only allows the youBot to grab objects that are at the egde of the table, stand upright, and are well separated from each other. This configuration is present on the table at the right of the youBot, when it is at its starting position (see figure 1). The youBot will thus try to move the objects that are on that table only. The arm of the youBot is moved most of the time using V-REP inverse kinematics.

We've thus adressed the milestones A1, B1, C2 and D2 of the project statement[1].

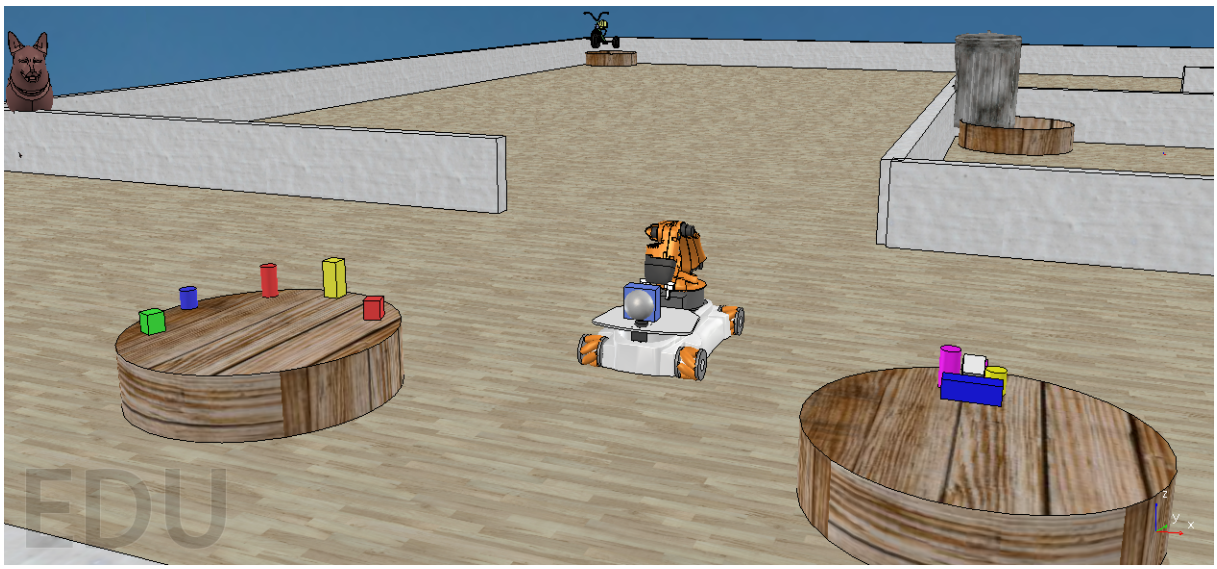

Figure 1: Example of a youBot starting position and its environment

2 Implemented solutions

2.1 Map exploration

The youBot stores a matrix representing the the map. The elements of this matrix can have three different values. One value represent the fact that a cell of the map hasn't been explored yet, another means that it has been, and the last one implies the presence of an obstacle in the cell. During its exploration, the youBot can thus gather the data supplied by its Hokuyo sensors and update the values of this matrix easily because it knows its accurate postion in the map.

Now that we've explained how the youBot represents and updates its map, it remains to describe how the youBot will achieve to explore its environment entirely. Our exploration algorithm is the following:

1. The youBot rotates around itself in order to get a 360° view of the map
2. The youBot selects the nearest unexplored point and establishes the trajectory to reach it
3. The youBot follows the trajectory, updates the map at the same time and makes sure that it's not running into an obstacle that wasn't yet acknowledged when the trajectory was established, otherwise, it goes back to step 2.
4. When the youBot reached his goal,
   if there is still unexplored points on the map: it goes back to step 2
   else: the hole map has been explored.

2.2 Trajectories

The trajectories of the youBot are calculated thanks to the (partial) map and the distance transform function from Peter Corke's robotics toolbox[3]. They are on the form of ordered series of points through which the youBot has to go. The more points the trajectory contains, the more fluid the trajectory is, so based on the fluidity or precision we want to have, we can increase the number of points of a trajectory.

To make the robot follows a trajectory, we used similar controllers to the ones presented in section 4.2.3 of Peter Corke's book "Robotics, vision and control"[2]. These controllers allows us to define a global velocity and a rotation velocity at each timestep but, in addition to that, we took advantage of the omnidirectional wheels of the youBot and turned the global velocity into a velocity vector, which components are the left-right and the forward-back velocities, as shown on figure 2.

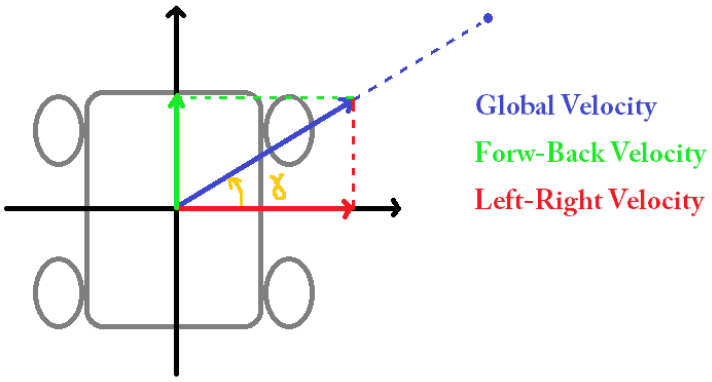

Figure 2: youBot omnidirectional velocity vector

2.3 Locating an identifying baskets

2.4 Moving objects

The process that we've implemented to achieve this purpose is the following:

1. The youBot goes to the area of the table. This is done by defining some points that are at a certain distance around the table, and selecting the closest one as a goal.
2. When it reached this goal, the youBot rotates in order to position itself with its direction perpendicular to the line going through its center and the center of the table. There is however two direction that follows this criteria, so we chose the one that is pointing in the clockwise direction around the table (see figure 3).
3. The youBot then get close to the table using left-right velocity only.
4. When it is close enough, it takes a single 3D image with a large angle of view to detect objects above the table. Only the points that are above the table, and in a certain range, are kept.
5. Whether objects are detected or not, every moves that the youBot will make at this step and the three next ones will follow a circular path defined around the table. If nothing is detected, the youBot goes a bit further and starts again from step 4, unless this has been repeated enough times to make a complete turn of the table and realize that there is no object left on it. In that case the job is done. If, however, the 3D image contains any points, the youBot selects the closest one and moves to align its camera with that point and the center of the table.
6. The camera is oriented to the center of the table and takes once again a single 3D image, this time with a shorter angle of view, and a small range, in order to see only one object.
7. The youBot then projects the points of this 3D image in the plane of the table and uses an implementation of the RANSAC algorithm[4] to recognize circles of a given radius. If there is a match, the youBots then knows that the object is cylindrical

but also knows the position of the center of this cylinder. If there is no match, the object is then box-shaped and each coordinate of the center of the box can be approximated by taking the extreme values and calculating their mean.

8. The youBot moves to align its arm reference point with the center of the object and the center of the table.
9. The youBot places its arm tip in front of the box using V-REP IK.
10. It moves its arm slowly along a straight line trajectory going through the center of the object, still using V-REP IK, and then grab it.
11. The youBot then moves its arm to a configuration where the object will be just above the youBot platform and near the camera, so that it can take an RGB picture and guess the color of the object in the gripper. This guessing is performed by looking at only one particular point of the picture that is supposed to always belong to the object. It can lead to an unexpected color if the youBot somehow failed to grab anything and in that case, it goes back to step 4.
12. The youBot moves its arm to its transport configuration, with the object still into the gripper, and goes to the appropriate basket to throw it. Then it goes back to step 1. Going to the baskets and getting close to them is performed with the same technique as for a table.

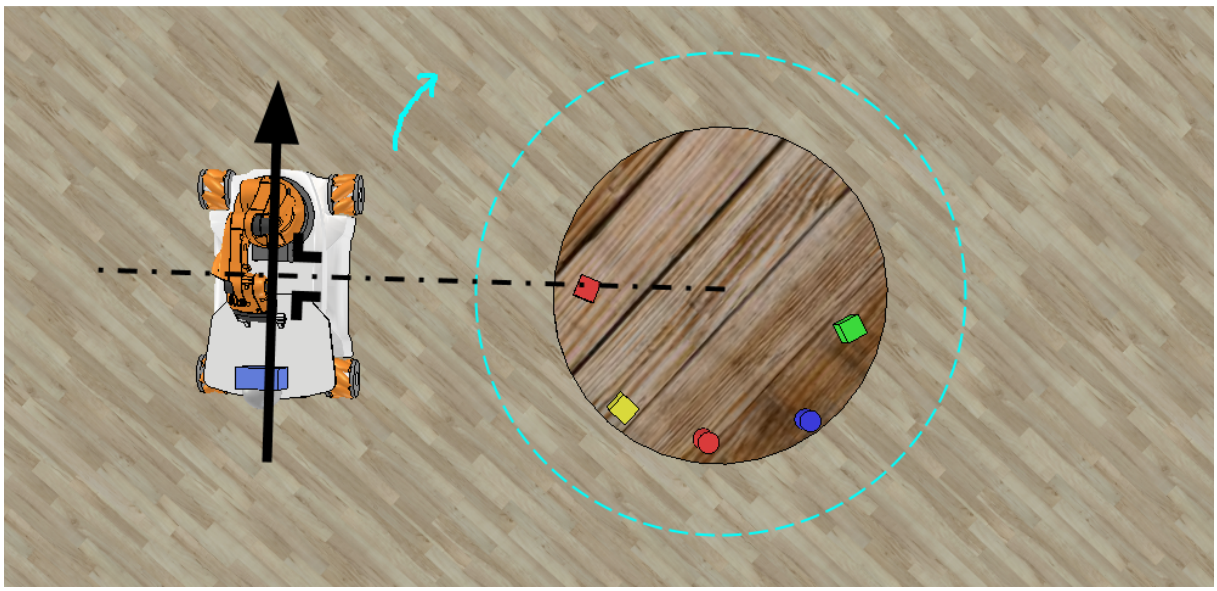

Figure 3: youBot positionning before getting closer to the table

References

[1] Detry, Renaud. "Project definition" *TRS: An Open-source Recipe for Teaching/Learning Robotics with a Simulator.*
http://ulgrobotics.github.io/trs/project.html

[2] Corke, Peter. *Robotics, Vision & Control.* 2011.
http://www.petercorke.com/RVC/

[3] Corke, Peter. *Robotics Toolbox for MATLAB.* Release 9.
http://www.petercorke.com/Robotics_Toolbox.html

[4] Zuliani, Marco. *RANSAC toolBox.* https://github.com/RANSAC/RANSAC-Toolbox