SPQR@Work Cubic-Splines Navigation

Luca Monorchio[1] Wilson Villa[2]

Abstract
RoboCup@Work is a competition in RoboCup that targets the use of robots in work-related scenarios. It aims to foster research and development that enables use of innovative mobile robots equipped with advanced manipulators for current and future industrial applications, where robots cooperate with human workers for complex tasks ranging from manufacturing, automation, and parts handling up to general logistics. The navigation system for the robots is one of the critical parts in this kind of scenarios where holonomic robots performs paths in presence of obstacles and smooth corridors. In this paper we will show a new navigation system based on cubic splines approach under a topological graph.

[1] *Master in Artificial Intelligence and Robotics, mat. 1650427*
[2] *Master in Artificial Intelligence and Robotics, mat. 1327106*

Contents

Introduction

The purpose of the Robocup@Work competition is to assess the ability of the robots for combined navigation and manipulation tasks. The robots have to deal with flexible task specifications, especially concerning information about object constellations in source and target locations, and task constraints such as limits on the number of objects allowed to be carried simultaneously. A single custom robot is used,

Figure 1. A youbot in action during an edition of Robocup.

which is initially positioned outside of the arena near a gate to the arena. The task is to get several objects from the source service areas (such as SH02, WS09, or CB02), and to deliver them to the destination service areas (e.g. WS11 and SH05). Robots may carry up to three objects simultaneously. The task specification consists of two lists: The first list contains for each service area a list of manipulation object descriptions. The second list contains for each destination service area a configuration of manipulation objects where the robot is supposed to achieve.
The following rules have to be obeyed:

- The robot has to start from outside the arena.

- The robot will get the task specification from the referee box.
- After the team's robot starts, it must move into the arena and attempt to complete the task.
- A manipulation object counts as successfully placed, if the robot has placed the object into the correct destination service area.
- It is not allowed to place manipulation objects anywhere except for the robot itself and any of the available service areas.
- A robot may carry up to three objects at the same time.
- The time is stopped when the robot has completed the task (delivered all objects to the right locations and left the arena through the exit gates). If a team cannot complete the task within the designated time, the run will be stopped.

In this project we propose a new holonomic navigation system based on Voronoi diagram used as topological graph in which the robot will performs the navigation assigned. All code is based on the ROS Kinetic version and tested on Kuka Youbot with Ubuntu 16.04 operating system. ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more (ROS is licensed under an open source, BSD license).

1. Voronoi diagram

Topological graph allows the robot to conveniently manage all the zones of the map subject to risk of collision, in this regard the generation of a Voronoi diagram facilitate navigation in order to achieve the preset goal.

Build the topological graph

The ros node **spqr_topological_graph** deals with the creation of the topological graph. The user at runtime can choose to add or delete nodes present on the map interface. This map is loaded directly from another node called **map server**, which offers map data as a ROS Service. Maps manipulated by the tools in this package are stored in a pair of files. The YAML file describes the map meta-data, and names the image file. The image file encodes the occupancy data. The map chosen for this project corresponds to the arena of the German-Open 2017 held in Magdeburg every year. The acquisition process has been done with the gmapping package that provides laser-based SLAM (Simultaneous Localization and Mapping), as a ROS node called **slam_gmapping**. Using **slam_gmapping**, we have created a 2D occupancy grid map (like a building floorplan) from laser and pose data collected by the youbot, as depicted in Figure 2. After this step we can generate by press "v" in automated manner the Voronoi paths inside the

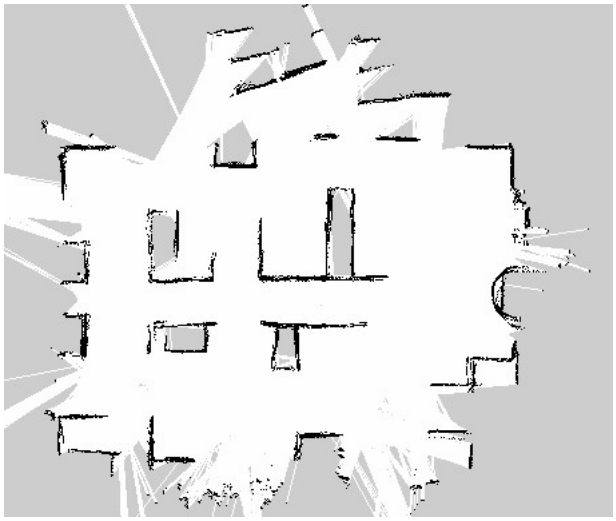

Figure 2. Gmapping's output map.

map, as depicted in Figure 3. The Voronoi diagram also allows you to import the positions set by the organizers of the arena. These particular locations come from the **location service** node and are represented by all the nodes having blue color (distinguished by those added later that have red color, as depicted in Figure 3. Once we have finished adding

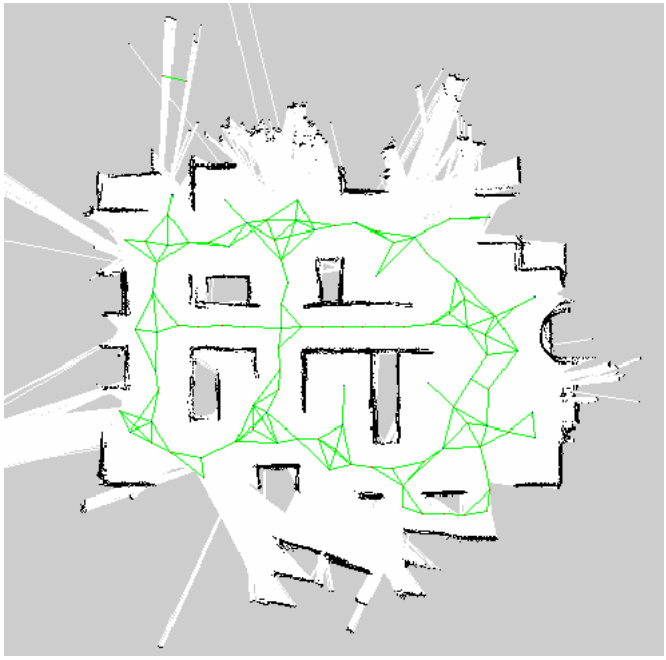

Figure 3. The Voronoi diagram generated on topological graph.

and connecting the various nodes properly, all that remains is to export the entire graph to file in order to proceed with the controller node part; for further informations, the list of commands available for this node is printed on the terminal every time the "h" key is pressed from the keyboard, for

example *"press a to remove arc between nodes"*, *"press e to export the graph on file"*...

Figure 4. The Voronoi paths (blue) with Cubic-Splines (green) imported on RViz.

2. A-Star and Cubic-Splines

After obtained the topological graph, generated in the above section, we compute a minimal path (with AStar algorithm) through the graph between the start (the actual robot's pose) and the goal of the navigation (given by the user-CFH). Finally, after obtained all the points retrieved from the AStar, we interpolate them with using the Cubic-Splines function that are continuous in positions and velocities, as depicted in Figure 5, 6.

3. The Controller System

In this last section we show the Ros node devoted to the publication of the cmd_vel command, this node is the controller part of the entire system of navigation.

Holonomic Controller

The error feedback controller then executes the trajectory solely based on odometry information, decoupled from global trajectory planning and global localization. Thereby, the error feedback controller can run at a higher frequency than the trajectory planning. We account for accumulating odometry drift and changes in the environment by planning updated trajectories while the robot executes the trajectory. The input for the controller are the planned position $T(t)$ and velocities $\dot{T}(t)$ for the current time t as determined by the trajectory T as well as the current position $o(t)$ in the odometry frame as measured by the robot's wheel encoders. The output of the controller is a velocity $v = (v_x, v_y, v_\theta)^T$ that is sent as a command to the robot. The controller computes the command

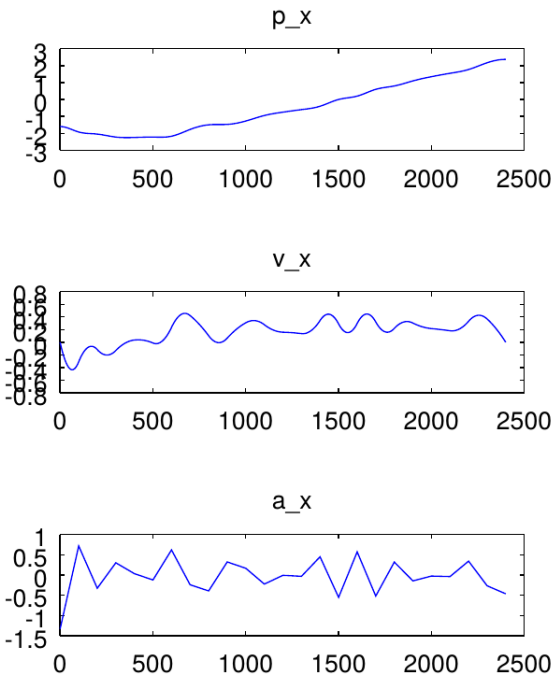

Figure 5. Spline along X axis.

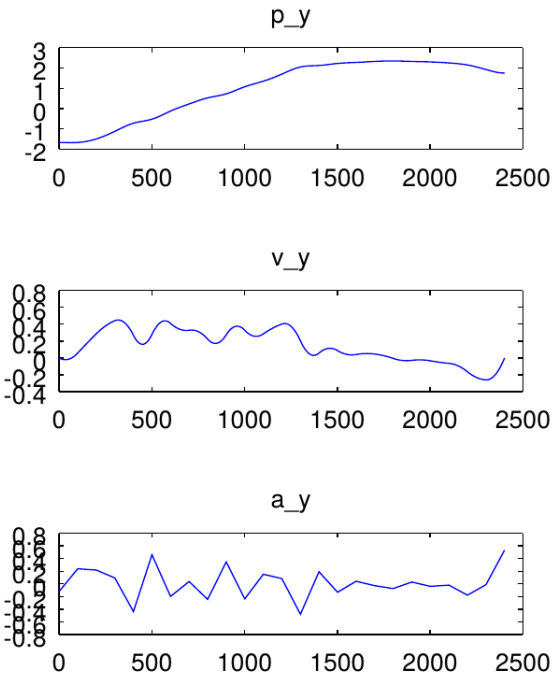

Figure 6. Spline along Y axis.

velocity as follows:

$$v = \dot{T}(t + \Delta t_{del}) + diag(g_t, g_t, g_r)(T(t) - o(t)) \qquad (1)$$

where g_t and g_r are the translational and rotational gain, respectively. The control law consists of two summands, the first summand is the feed-forward part and the second summand is the error feedback. The feedforward part is the velocities as planned in the trajectory. We retrieve them for a point in time that is shifted by Δt del into the future. This accounts for the time delay that occurs before the robot is actually executing the command. For execution by the robot, the commanded velocity needs to be transformed into wheel turn rates. These are managed by the youbot holonomic controller provided by Ros it self.