Wrocław University
of Science and Technology

Chair of Cybernetics and Robotics
Faculty of Electronics
Wroclaw University of Science and Technology

Environment for testing Endogenous Configuration Space Approach algorithms

Author:
Łukasz Chojnacki

Supervisor:
Dr. Witold Paluszyński

Wroclaw, February 2017

Abstract

This report presents a software and a hardware environment for Endogenous Configuration Space Approach. ECSA is a set of motion planning algorithms and can be used as a trajectory tracking algorithm. The algorithms have already been tested during computer simulations. The next step is to build real-time robotic system and use ECSA algorithms to plan and track trajectory for mobile robot.

1 Introduction

A autonomous mobile robot should move from point A to point B avoiding obstacles. The problem can be divided into two subproblems: motion planning problem and trajectory tracking problem. The first subproblem can be treated as planning feasible trajectory for mobile robot. The second one is a problem of minimization reference (given) trajectory and actual trajectory. There are a lot of different approaches and algorithms to solve this kind of problems. One of the them is Endogenous Configuration Space Approach that attempts to find such controls that move a mobile platform from one point to another. The approach has been introduced in [Tchoń and Jakubiak, 2003] and different algorithms using this approach are listed below:

- *Extended* [Tchoń and Jakubiak, 2002]
- *Imbalanced* [Janiak and Tchon, 2010], [Janiak and Tchoń, 2011]
- *Prioritarian* [Ratajczak and Tchoń, 2013]
- *Lagrangian* [Tchoń et al., 2015b], [Tchoń et al., 2015a], [Tchon and Góral, 2015]
- *Lifted Newton* [Janiak and Tchoń, 2015], [Janiak, 2015]

All this algorithms are a part of Endogenous Configuration Space Approach framework [Chojnacki, 2016] which is a python library. To solve the trajectory tracking subproblem Model Predictive Control, task priority Lifted Newton trajectory reproduction algorithm or kinematic controller [Morin and Samson, 2004] can be used.

For this kind of algorithms, the real-time control system should be employed, and this report presents a real-time robotic system that can move from one point to other one following desired feasible trajectory. The system architecture is presented in section 2, the hardware aspect of the system is shown in section 3, and the software in section 4. To verify that the system works correctly an experiment was prepared and is presented in section 5.

2 System's Architecture

Component based approach was used to design real-time system. Each component is a kind of black box that receive messages on input and return other messages on output. That approach helps easily replace one block with another, for instance, we easily change Model Predictive trajectory tracking algorithm. Real-time execution and proper tools are needed to ensure that the system sends control command periodically and nothing can break it. In [Janiak and Zieliński, 2015] has been presented a distributed system that

satisfies real-time constraints. Authors proposed Xenomai framework to provide real-time of Linux operation system like Ubuntu, and OROCOS and ROS middlewares to build software components of the system. In [Janiak, 2016] has been presented a board that can control motors in real-time. The board can communicate in real-time with PC using RTnet protocol. The real-time control system which was developed during the project is presented at figure 2.1, and is composed of the following components:

- Motion capture – return information about position of a robot
- Estimator – estimate a robot state based on position from motion capture and position of wheels
- Safety system – ensure that a robot will push the wall etc.
- Hokuyo – return information from laser scanner
- Arbiter – decide that get control commands from controller or remote gamepad
- PS3Teleop – return control commands from gamepad
- Wheels module – ensure that received control commands will be executed
- RTnode – execute control command and receive encoder speed and position
- Controller – is a trajectory tracking algorithm

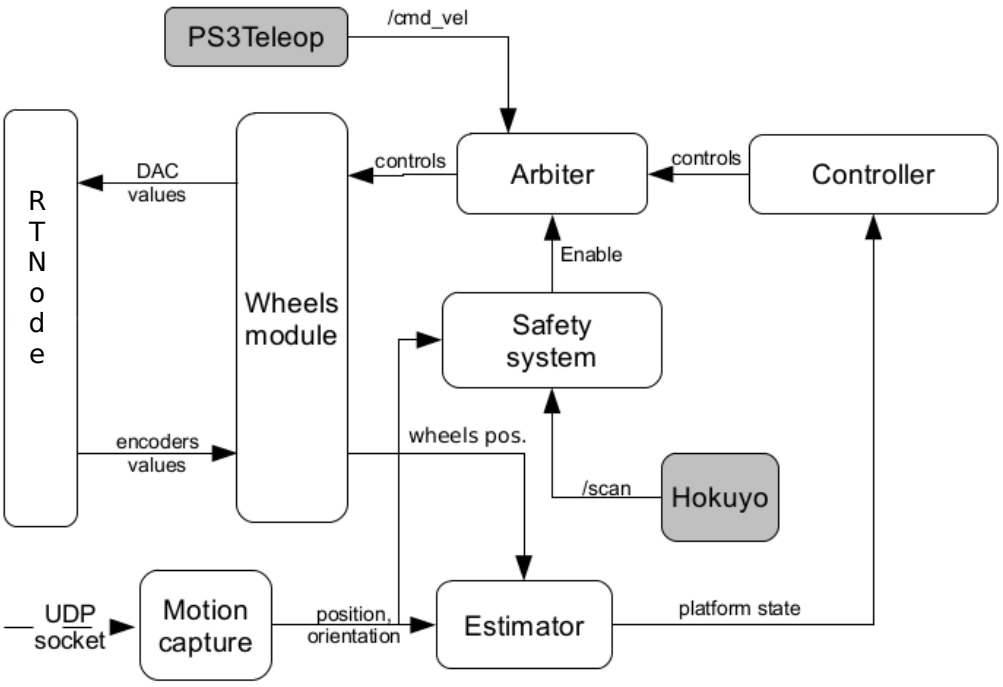

Figure 2.1: Architecture of the system

3 Hardware

System's architecture presented in the last section can be applied to every robot. In this project was used differential drive mobile robot, which is one of the simplest robot to control. Pioneer P3-DX was chosen because is available at Wroclaw University of Science and Technology. The robot (Fig. 3.1) had to be modify. The main PC was replaced

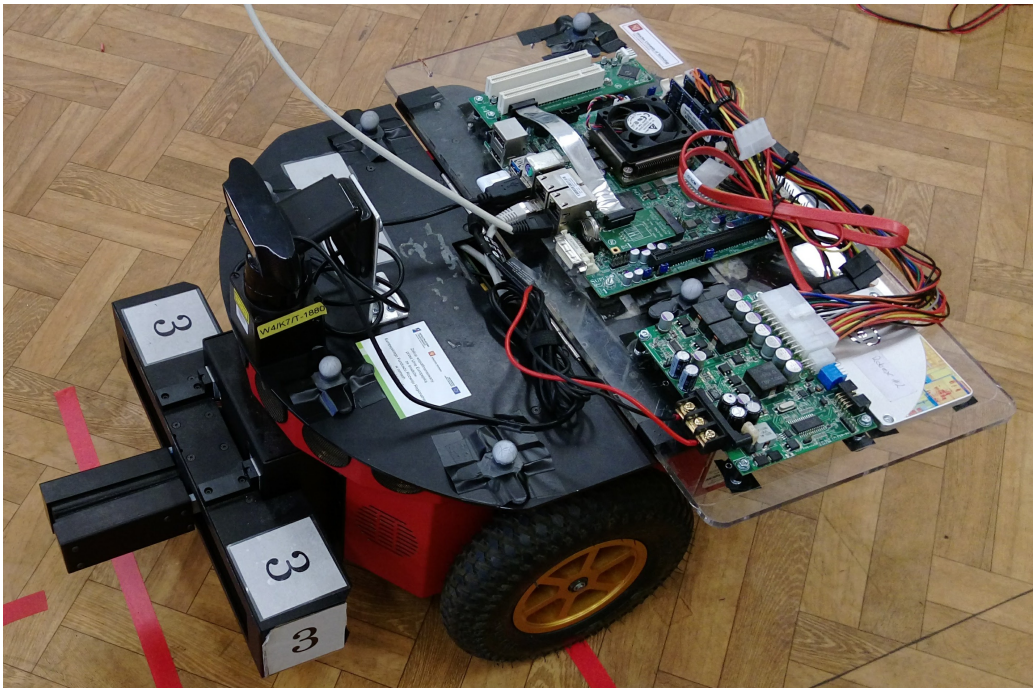

Figure 3.1: Pioneer

with industrial PC with stronger processor, and RTnode was used instead of Pioneer control board (3.2). The motors were connected to RTnode through ESCON 50/5 motor drivers. The motor's encoder were directly connected to RTnode. The next part of the

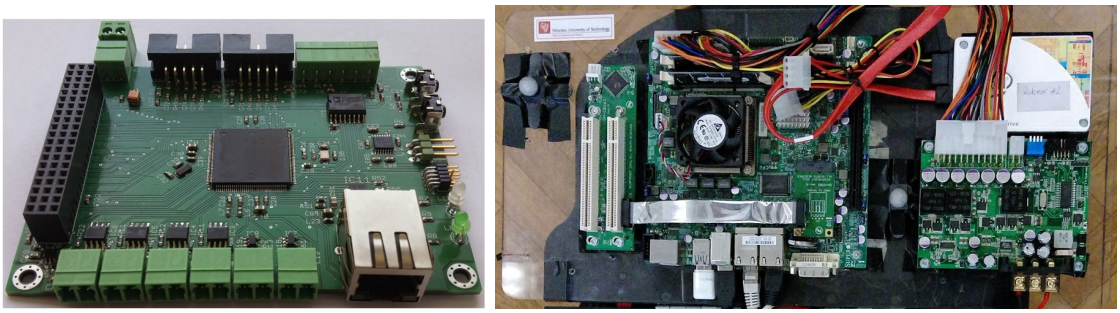

Figure 3.2: RTnode – left, and PC – right

system, from the perspective of the hardware, is OptiTrack Motion Capture System. Motion Capture is composed of six cameras that can follow desired rigid body, in this case Pioneer robot. The robot has to have fixed markers the are located asymmetrically. Figure 3.3 presents print screen from the software delivered by a manufacturer. In the middle top are presented the scene with six cameras and defined robot as a rigid body. Below are the view from each camera separately. At the left side is shown position and orientation of the robot that is published by local network (right side).

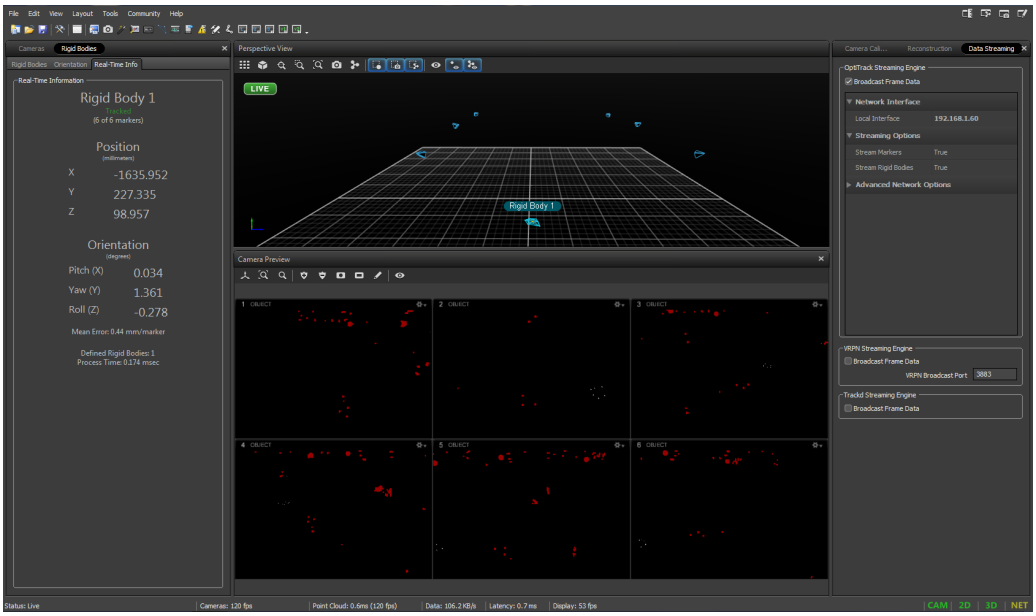

Figure 3.3: Motion capture

4 Software components

A component is a piece of software that model a particular functionality of the system. It can be compared to a brick, and like from blocks we build a construction that performs the desired job. In this way, it is possible to construct a robotic system that does a specific task, like for example moving a robot from one point to another. Such a system should consist of the suitable components. At figure 2.1 is presented an architecture that allows following the desired trajectory. The good idea is to use a framework to speed up the process of programming. ROS is a well-known robotic framework that has a lot of tools and packages which help us to develop and debug written code, but unfortunately not guarantees real-time. For that reason, it is necessary to use OROCOS framework that allows running components in real-time.

The functionality of OROCOS components is the following:

- Motion capture – the robot is defined as a rigid body in OptiTrack application. Next, the application broadcast the position and orientation of the rigid body in space. The primary task of the component takes this information and publish it like a posture of the robot, in this case, is a planar position (x, y) and orientation (θ).

- Estimator – there are two sources of robot pose: from motion capture, and from encoders. This component takes this two sources and estimates the state of the mobile platform.

- Controller – is a trajectory tracking algorithm. In this project kinematic controller [Morin and Samson, 2004] is implemented. The controller takes the desired state of the platform in given time (reference trajectory) and actual state and computes the controls signals. In the beginning, the component read the reference trajectory from file to the array. Next, in each time it calculates controls based on desired and actual state. In the end, it saves the reference, executed trajectory and controls to file.

- Safety system – it is a kind of kill switch. The component decides that robot should be enabled or not, based on data from laser scanner and motion capture. If a

robot is out of the available region or is too close to the obstacle, it should not be enabled, and stops execute any command. Otherwise, a robot should execute control commands.

- Arbiter – choose from which source trajectory tracking controller or gamepad controller, a robot should execute control commands. Moreover, the component enables the robot based on information from safety system component.
- Wheels module – is a component responsible for communication with a low-level controller. It translates control command (in this case it is linear and angular velocity) to number understood by a low-level controller and vice versa.
- RTnode – establish the connection between RTnode device and PC through RTnet, and next send and receive proper frame between devices.

The functionality of ROS nodes is the following:

- PS3Teleop – get gamepad state and translates it to linear and angular velocity.
- Hokuyo – get data from hokuyo laser scanner and publish it.

5 Experiment

The main goal of the project was building the system that allows a robot track a reference trajectory. One of the last action which has to do is a validation of the system. Thus an experiment was prepared to check that presented system is able to follow a given trajectory.

The experiment was the following; the robot should track the circle trajectory with radius one. Figure 5.1 presents initial position during the experiment. The red cross in the middle is the center of the circle. The initial position of the robot in global coordinates is $(0, 0, 0)$. Total time of the trajectory was 26.4s. During the experiment desired state (x_d, y_d, θ_d), actual robot state (x, y, θ) and commanded controls (linear and angular velocities) was logged.

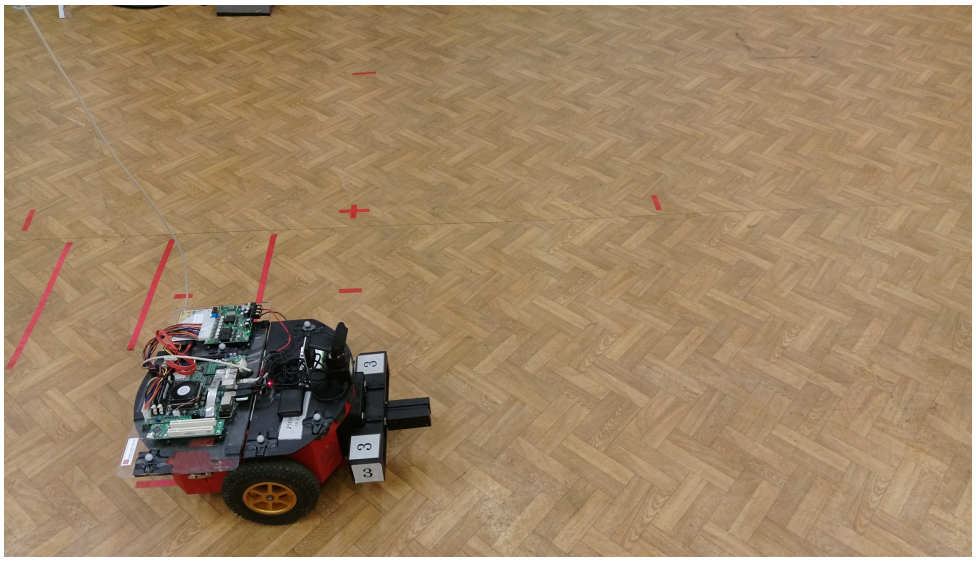

Figure 5.1: View of the experiment

The result of the experiment shows figure 5.2. The dashed line is the reference trajectory, and the solid one is the trajectory realized by the robot. The robot tracks the trajectory quite good, but the result is not perfect.

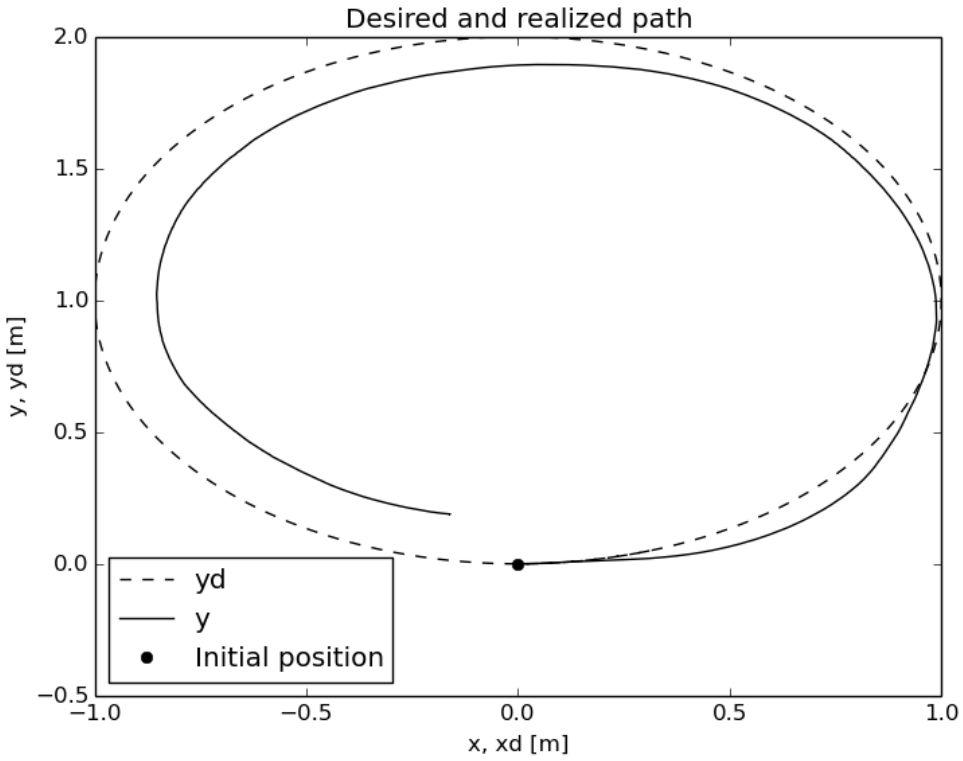

Figure 5.2: Result

6 Conclusion

The report presents the robotic system that allows tracking the trajectory. The hardware is a differential drive mobile robot Pioneer P3-DX enhanced by industrial PC, RTnode board and two ESCON 50/5 motor drivers. OptiTrack Motion Capture system is used as a main reliable source of the robot position and orientation. To ensure that each control command is realized in proper time, real-time operation system and framework are necessary. Thus Xenomai as a real-time extension of Linux kernel and OROCOS are used. OROCOS guarantee that each component is executed in real-time. RTnet protocol guarantees real-time communication between RTnode board and PC. Components that do not need real-time execution are implemented in ROS framework.

The experiment shows that the designed system works correctly but unfortunately is not perfect. In the next stage of the project, another trajectory tracking algorithms will be implemented, like for example algorithms based on Model Predictive Approach and Endogenous Configuration Space Approach. Next, the reference trajectory will be given by one of the motion planning algorithms. The project is a starting point of the Master thesis "Real-time toolchain for Endogenous Configuration Space Approach".

References

[Chojnacki, 2016] Chojnacki, L. (2016). ECSA framework. `https://bitbucket.org/lukych92/ecsa`.

[Chojnacki, 2017] Chojnacki, L. (2017). Real-time control system. `https://bitbucket.org/lukych92/realtime_control_system`.

[Jakubiak and Tchoń, 2004] Jakubiak, J. and Tchoń, K. (2004). Fourier vs. non-fourier band-limited jacobian inverse kinematics algorithms for mobile manipulators. In *Proc. 10th IEEE Int. Conf. on Methods and Models in Automation and Robotics*, pages 1005–1010.

[Janiak, 2015] Janiak, M. (2015). Lifted newton motion planning algorithm. In *Robot Motion and Control (RoMoCo), 2015 10th International Workshop on*, pages 223–228. IEEE.

[Janiak, 2016] Janiak, M. (2016). Rtnode-komponent zintegrowanego środowiska sprzętowo-programowego. *Prace Naukowe Politechniki Warszawskiej. Elektronika*, pages 79–90.

[Janiak and Tchon, 2010] Janiak, M. and Tchon, K. (2010). Constrained robot motion planning: Imbalanced jacobian algorithm vs. optimal control approach. In *Methods and Models in Automation and Robotics (MMAR), 2010 15th International Conference on*, pages 25–30. IEEE.

[Janiak and Tchoń, 2011] Janiak, M. and Tchoń, K. (2011). Constrained motion planning of nonholonomic systems. *Systems & Control Letters*, 60(8):625–631.

[Janiak and Tchoń, 2015] Janiak, M. and Tchoń, K. (2015). Motion planning through waypoints for a skid-steering mobile platform. In *Robot Motion and Control (RoMoCo), 2015 10th International Workshop on*, pages 58–63. IEEE.

[Janiak and Zieliński, 2015] Janiak, M. and Zieliński, C. (2015). Control system architecture for the investigation of motion control algorithms on an example of the mobile platform rex. *Bulletin of the Polish Academy of Sciences Technical Sciences*, 63(3):667–678.

[Morin and Samson, 2004] Morin, P. and Samson, C. (2004). Trajectory tracking for non-holonomic vehicles: overview and case study. In *Robot Motion and Control, 2004. RoMoCo'04. Proceedings of the Fourth International Workshop on*, pages 139–153. IEEE.

[Ratajczak and Tchoń, 2013] Ratajczak, A. and Tchoń, K. (2013). Multiple-task motion planning of non-holonomic systems with dynamics. *Mechanical Sciences*, 4(1):153–166.

[Tchon and Góral, 2015] Tchon, K. and Góral, I. (2015). Lagrangian jacobian motion planning. In *Proc. IMA Conference on Mathematics of Robotics, St Annes College, University of Oxford*.

[Tchoń et al., 2015a] Tchoń, K., Góral, I., and Ratajczak, A. (2015a). Jacobian motion planning of nonholonomic robots: The lagrangian jacobian algorithm. In *Robot Motion and Control (RoMoCo), 2015 10th International Workshop on*, pages 229–234. IEEE.

[Tchoń and Jakubiak, 2002] Tchoń, K. and Jakubiak, J. (2002). Extended jacobian inverse kinematics algorithms for mobile manipulators. *Journal of Field Robotics*, 19(9):443–454.

[Tchoń and Jakubiak, 2003] Tchoń, K. and Jakubiak, J. (2003). Endogenous configuration space approach to mobile manipulators: a derivation and performance assessment of jacobian inverse kinematics algorithms. *International Journal of Control*, 76(14):1387–1419.

[Tchoń et al., 2015b] Tchoń, K., Ratajczak, A., and Góral, I. (2015b). Lagrangian jacobian inverse for nonholonomic robotic systems. *Nonlinear Dynamics*, 82(4):1923–1932.