UNIVERSITY OF VERONA

MASTER THESIS

Title . . .

Author:
Michele PENZO
VR439232

Supervisor:
Dr. Diego DALL'ALBA

ICE Lab
Department of Computer Science

December 30, 2020

UNIVERSITY OF VERONA

Abstract

Department of Computer Science

Master of Science

Title . . .

by Michele PENZO
VR439232

abstract . . .

sommario . . .

Acknowledgements

…

Contents

List of Figures

List of Tables

List of Abbreviations

LBR	**L**eicht **B**au **R**oboter
IIWA	**I**ntellingent **I**ndustrial **W**ork **A**ssistant
TCP	**T**ool **C**enter **P**oint
EE	**E**nd **E**ffector
DOF	**D**egree **O**f **F**reedom
ROS	**R**obot **O**perating **S**ystem
HMI	**H**uman **M**achine **I**nterface

Chapter 1

Introduction

The entire work is available on Github at: github.com/michelepenzo/master-thesis.

1.1 Motivations

1.2 Goals

1.3 Thesis Overview

Chapter 2

State of the art

2.1 Robot learning from Demonstration

2.1.1 Kinesthetic Teaching

2.1.2 Teleoperation

2.2 Assembling and modulable tasks

2.3 User study

Chapter 3

The project

This chapter describes the general setup, its components and a small overview on the tools used for develop the project. Finally the project is explained.

3.1 Setup overview

The KUKA LBR IIWA redundant manipulator is programmed using the KUKA's Sunrise Workbench platform and its Java API's. The usage of an open source stack compatible with ROS allows the usage of the robot in a simple way.

A Sunrise project, containing one or more Robotic Application can be synchronized to the robot cabinet and executed from the SmartPad.

The *iiwa stack* provide a Robotic Application that can be used with the robot. It establishes a connection to machines connected via Ethernet to the robot cabinet via ROS. The machine, with ROS installed, will be able to send and receive ROS messages to and from the Robotic Application. The messages used in this stack are taken from the messages available in a standard ROS distribution, but there are other custom ones inside the `iiwa_msgs` folder.

With the stack is simple to manipulate the messages received from the robot and send new ones as command to it, using Python script or ROS functionalities already implemented as services, topics, actions.

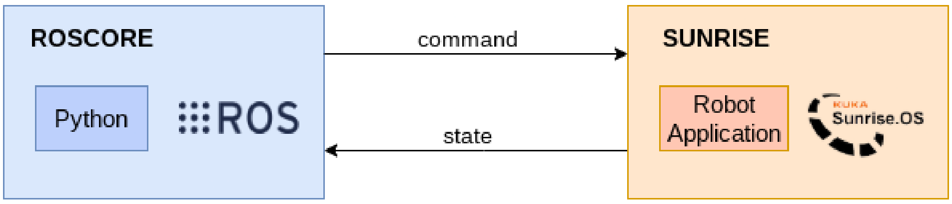

FIGURE 3.1: Robot control via ROS using *iiwa_stack* and Sunrise OS

For further information about *iiwa_stack* see [1] and the related work [2]. Instead for information about Sunrise OS and Workbench see [3].

3.1.1 KUKA Robot Controller

The KUKA LBR IIWA is controlled via the KUKA Robot Controller, also known as the KUKA Sunrise Cabinet. The KRC is responsible for the transmission control inputs as well as the reading the data of the integrated sensors. In our case, for controlling the Kuka we use the Java application providede by the stack that runs into the SmartPad.

3.1.2 Sunrise.Workbench

The Sunrise.Workbench is a tool used to program robot applications in Java, which are loaded into and are executed on the KRC. It offers the possibility to control the robot with the following strategies: position control, velocity control, joint and cartesian impedance control. It can also can execute the commonly motion patterns as: spline, point-to-point, linear and circular motions.

Since we have a gripper mounted on media flange, there's a task always active in background that provides a method that can be called via `ros_service` to open and close the two jaw. We also have another background task for the *rgb* led present on media flange.

3.1.3 Safety configuration

A correct thing to do before start the using of the robot is to check and modify the safety configuration loaded by default. In my case:

- I had to set a new value for every joint in robot configuration. I restrict the value about 2 degrees.
- moreover I added a protected workspace. In this case the robot never goes inside the constraint area.

3.2 Project implementation

As described in section 1.2, the goal of the thesis was to …

Staring from the goal, the work was divide into two phases:

1. create the mode relative to teach by demonstration,
2. create a way to tele-operate the robot in a simple way.

As described in the next sections, in every phase a way to save waypoints and an action on the gripper was implemented. After that the action was captured by the script and was saved in an `.csv` file. With a dedicated program and that file, all the actions saved into the file can be replicated by the robot. Then the description of the two developed modalities.

3.2.1 Theach by demonstration

As described in section 2.1.1, teach by demonstration or also called *kinestethic teaching* is a way to move the robot in gravity compensation mode. Using the *iiwa_stack*, the gravity compensation mode was implemented using a joint impendance control mode where for every joint in robot configuration a stiffness and damping value is setted. Stiffness value must be grater than 0 and it is expressed in Nm/rad, instead damping value must be between 0 and 1. After changing the control mode to joint impendance, the robot seems falls. A force contrary to the gravity must be carried out to keep it up.

On the mediaflange, the green button was dedicated for saving actions on gripper and waypoints.

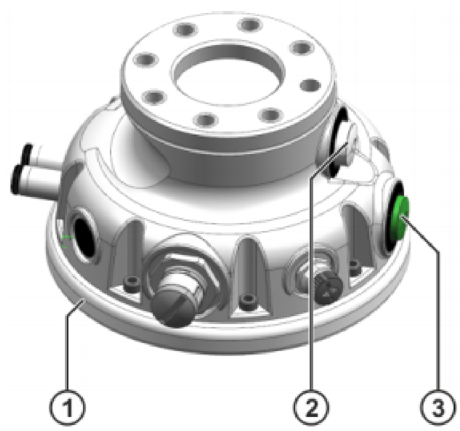

FIGURE 3.2: The Mediaflange: (1) led strip,
(2) enabling switch, (3) application button

3.2.2 Teleoperation

Chapter 4

Materials and methods

4.1 Materials

4.2 Assembly methods

Chapter 5

Experiments

5.1 Participants

5.2 Assembly tasks

Chapter 6

Results

6.1 Results

Chapter 7

Conclusion

7.1 Result discussions

7.2 Conclusions

7.3 Future works

References

[1] URL: https://github.com/IFL-CAMP/iiwa_stack.

[2] Hennersperger Christoph et al. “Towards MRI-based autonomous robotic US acquisitions: a first feasibility study”. In: *IEEE transactions on medical imaging* 36.2 (2017), pp. 538–548.

[3] Sebastian Schleisner Hjorth. “Investigation and Implementation of Workspace restrictions for the KUKA LBR iiwa”. 2019.