INTER-UNIVERSITY MASTER'S DEGREE PROGRAMME


Università degli Studi di Milano


Università degli Studi di Milano - Bicocca


Università di Pavia

Master's Degree Programme
Artificial Intelligence for Science and Technology

# This is the title of my thesis

Supervisor:
Prof. Name Surname
Co-supervisor:
Dr. Name Surname

Candidate name:
Name Surname
Registration number:
AB512WX

Academic Year 2021/2022

# Abstract

**What is the abstract?** The abstract is an important component of your thesis. It is likely the first substantive description of your work read by an external examiner. You should view it as an opportunity to set accurate expectations. The abstract is a summary of the whole thesis. It presents all the major elements of your work in a highly condensed form. It is not merely an introduction in the sense of a preface or advance organizer that prepares the reader for the thesis. It must be capable of substituting for the whole thesis when there is insufficient time and space for the full text.

**Size and Structure.** Typical lenghts are within 200 and 300 words (without sections). An abstract has to live as a stand-alone text. The structure should mirror the structure of the whole thesis, and should represent all its major elements. For example, if the thesis has five chapters (introduction, literature review, methodology, results, conclusion), there should be one or more sentences assigned to summarize each chapter.

**Clearly Specify Your Research Questions.** As in the thesis itself, your research questions are critical in ensuring that the abstract is coherent and logically structured. They form the skeleton to which other elements adhere. They should be presented near the beginning of the abstract.

**Don't Forget the Results.** The most common error in abstracts is failure to present results. The primary function of your thesis (and by extension your abstract) is not to tell readers what you did, it is to tell them what you discovered. Other information, such as the account of your research methods, is needed mainly to back the claims you make about your results. Approximately the last half of the abstract should be dedicated to summarizing and interpreting your results.

*Here you can write whatever you want*

*and here too.*

# Acknowledgments

Some acknowledgments to family, friends, supervisors, etc.

# Contents

# List of Figures

# List of Tables

# Acronyms

**LED** Light Emitting Diode

**UNIPV** University of Pavia

**Robolab** Robotics Laboratory

# Chapter 1

# Introduction

"A design of any kind shows its real value when
taking beyond its original limits."

*Tom Jennings*

Some few conventions (and an example of nested item list):

- In a `.tex` file, text lines can have variable length.
- Put a newline after every fullstop, see an example in section 1.1; paragraphs are separated by empty lines in the `tex` file, so a newline at the end of a sentence does not affect the structure of the paragraph.
- This is a tradeoff for using versioning tools:
  - Lines too long (full paragraphs) would introduce diffs that are too large.
  - Fixed maximum line widths (e.g., 80 columns) lead to editors to reorganize the paragraph to fit the width, causing again diffs that are too large.

You can use the percentage symbol (%)[1] to avoid a space between item list and this paragraph, so that the first line of this paragraph is not indented.

Enumerated list:

1. Item 1.
2. Item 2.
3. Item 3.

[1] And notice how to insert a symbol of percentage!

| *Column 1* | *Column 2* |
|---|---|
| text 1 | Some long text in the table. Some long text in the table. Some long text in the table. Some long text in the table. Some long text in the table.Some long text in the table. |
| text 2 | some other text. some other text. some other text. some other text. some other text. some other text. |

**Table 1.1:** This is the caption of the table.

4. Item 4.

Sometimes, when you have a list of items for which you don't want to use bullet points or enumerations, you can use paragraphs with titles. See the following text.

**This is the first paragraph.** This is the long text of the first paragraph. This is the long text of the first paragraph.

**This is the second paragraph.** This is the long text of the second paragraph. This is the long text of the second paragraph. This is the long text of the second paragraph. This is the long text of the second paragraph.

**This is the third paragraph.** This is the long text of the third paragraph. This is the long text of the third paragraph. This is the long text of the third paragraph. This is the long text of the third paragraph. This is the long text of the third paragraph. This is the long text of the third paragraph.

Finally, table 1.1 provides an example of table. You can add as many columns and rows as you wish. Please refer to `https://www.ctan.org/pkg/tabulary` for the documentation of the package. In particular, see how to change the size and the alignment of the columns, if you need it.

## 1.1 General recommendations

An example of line breaking follows[2]. Do you see it? This is a new line in the `.tex` file but it is the same paragraph in the `pdf`.

[2] And this is another example of footnote; don't forget the full stop at the end of the footnote.


**Figure 1.1:** Example of figure; this is the caption that goes below the figure.

Figure 1.1 reports an example of template to include a figure. Just copy this template around and set the filename (without extension!), the size, and write the caption.

Use something like

```
width=0.5\textwidth
```

to set the size of the figure as a percentage of the text width (in this case it is 50%). You can also specify an absolute width with `width=4cm`, but usually a relative size is fine. Sometimes (very rarely, in my experience) you may want to set the size on the basis of the height of the figure, i.e., `height=4cm`.

The label **must go inside the caption**, otherwise sometimes LaTeX does not handle the references correctly.

Every figure, table, listing or equation must be referenced and properly described in the text. **Never use statements such as "previous chapter", "next section" or "figure below"**: these elements may be moved during the editing of the text, thus if you use these statements then you may have to update the sentences every time you move something around. This document uses the package `cleveref`, so use the command `\cref` (e.g., this is a reference to section 1.1) for a smart referencing (use `\Cref` for uppercase). It takes care of putting the right label and spacing before the number, and it will reference the item independently from its position in the text.

To use equations and to refer to them, use the `equation` environment like in this way:

**Listing 1.1:** Example of listing.

```
def filter_difference(lines):
    filter_header_lines = filter(lambda line: not line.startswith(
        '---') and not line.startswith('+++'), lines)
    filter_difference = filter(lambda line: line.startswith(
        '-') or line.startswith('+'), filter_header_lines)
    return filter_difference
```

$$A = \pi r^2 \tag{1.1}$$

And proper reference to eq. (1.1).

If you don't need to refer to an equation, you can skipt the numbering:

$$2 + 2 = 4$$

In case of inline math, this is how it is done: $P = 2 \cdot \pi \cdot R$.

Listing 1.1 provides an example of listing for source code.

### 1.1.1 Formatting text

Beside equations and math text, there are some other formattings that are worth mentioning:

- for everything related to software, such as file names, functions, variables, etc., use `texttt`, e.g., `hello.txt` or `var_name`.
- to put some text within quotes, use `enquote`, e.g., "like this"; `enquote` is better than other solutions because it is more robust and handles internationalization correctly.

### 1.1.2 Citations

For citations from the bibliography use `\cite`. Here an example: [1].

Citations go **inside** the corresponding sentence. Note the position of the citation w.r.t. the full stop:

- "The algorithm has poor efficiency. [1]" **WRONG**
- "The algorithm has poor efficiency [1]." **CORRECT**

You have to populate the bibliography file `biblio.bib`. You can put as many items as you want in the file. Only the items that are cited in the thesis with `\cite` will be included in the bibliography, as in the example above.

The items to populate the `.bib` file are in BibTeX format. This is a popular format. If you look for some paper or book, it is likely that somewhere the bib format already exists. Try to search on the Internet for the title of the paper plus *bibliography* or *bib*.

### 1.1.3 Use of acronyms

Acronyms are very popular in scientific and technical documents. We use the `acronym` package.

All the acronyms are defined in `cap_acronyms.inc.tex`.

Use the acronyms with "University of Pavia (UNIPV)", "Robotics Laboratory (Robolab)", "Light Emitting Diode (LED)". The package is smart enough to write the long version the first time, and then to use the short version, like so: "UNIPV" and "Robolab".

Sometimes you need to use the full version again, do it with University of Pavia (UNIPV). If you need the long version only, use University of Pavia.

Plurals can be handled with LEDs.

## 1.2 Objectives of the thesis

Put here the objectives of the thesis.

## 1.3 Organization of the document

The thesis is organized as follows:

- chapter 2 explains this ...
- chapter 3 explains that ...
- chapter 4 presents this ...
- chapter 5 presents that ...
- finally the conclusions in chapter 6.

## 1.4 Partnership

Any possible additional information regarding the thesis.

# Chapter 2

# State of the art

"The wireless telegraph is not difficult to understand. The ordinary telegraph is like a very long cat. You pull the tail in New York, and it meows in Los Angeles. The wireless is the same, only without the cat."

*Albert Einstein*

This chapter provides an overview of the state of the art.

## 2.1 YOLO - Instance Segmentation Model

## 2.2 ByteTrack

# Chapter 3

# Tools and Frameworks

"I believe that inside every tool is a hammer."

*Adam Savage*

This chapter describes the tools and frameworks used in the thesis.

# Chapter 4

# Implementation

This chapter shows the details of the implementation of the work.

## 4.1 Software Structure

The software structure is divided into ROS nodes, each with a specific functionality. Figure 4.1 represents an abstract view of the message flow between the nodes. The following nodes are included:

- **Camera Node**: This node is responsible for capturing RGB and depth images from the Realsense L515 camera and publishing them to the ROS network.
- **Detection Node**: This node subscribes to the RGB and depth topics, performs image segmentation, and publishes the results to the ROS network.
- **Mapping Node**: This node subscribes to the detection result topic and creates a 3D point cloud of the detected objects. It acts as an intermediate node between the detection and world nodes. It has information about the different reference frames and spins a thread to remove objects whenever they are removed from the environment.
- **World Node**: This node exposes the necessary services to interact with the world model.

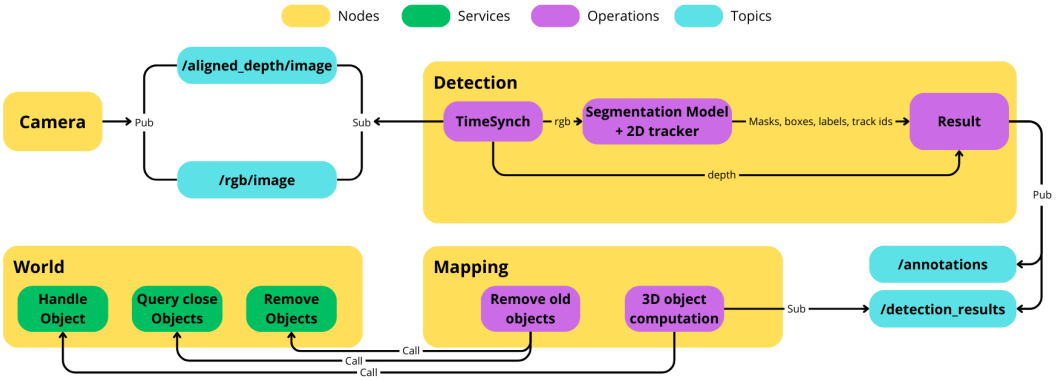


**Figure 4.1:** Software Structure

### 4.1.1 Detection Node

This node subscribes to the RGB and depth topics, performs instance segmentation, and publishes the results to the ROS network. When RGB and depth images are published a callback function is triggered, this function acts a series of steps:

#### YOLO - Instance Segmentation Model

The YOLO model is used to perform instance segmentation, for further information about design choices and model details refer to section 2.1. This model has been implemented using the API provided by Ultralytics [2]. In this application no batching is used since the model is running on a single image at a time. This particular implementation offers also an implementation of the ByteTrack algorithm explained in section 2.2 that allow to extract unique IDs for each detected object.

From this step we take only the segmentation masks, the unique IDs and the labels. 2D bounding boxes are not considered as 3D bounding boxes will be computed by from the objects point clouds.

### 4.1.2 Mask Erosion

Since depth acquired from this particular camera tends to ...

## 4.2 Eye in hand Calibration

Eye-in-Hand calibration is a crucial process in robotics where the spatial relationship between a robot's end-effector (hand) and a vision sensor (eye) is determined. This calibration ensures accurate perception and interaction with the environment, allowing the robot to perform tasks with high precision.

### 4.2.1 Tools and Software

The MoveIt! library is widely used for motion planning and manipulation in robotics. This chapter follows the MoveIt! hand-eye calibration tutorial found at the following GitHub repository: `https://github.com/JStech/moveit_tutorials/blob/new-calibration-tutorial/doc/hand_eye_calibration/hand_eye_calibration_tutorial.rst`.

### 4.2.2 Prerequisites

Before starting the hand-eye calibration process, ensure that the following prerequisites are met:

1. **MoveIt! Setup**: Ensure MoveIt! is installed and properly configured with your robot.
2. **Aruco Marker Board**: A physical Aruco marker board must be available and detectable by the vision sensor.
3. **Robot Control**: Ensure you have control over the robot and can move the end-effector to various positions.

### 4.2.3 Aruco Marker Board

An Aruco marker board is a grid of binary square fiducial markers used for pose estimation. Each marker has a unique ID, allowing the software to determine the board's orientation and position in space.

### 4.2.4 Calibration Procedure

#### Setup the Environment

Start by launching the MoveIt! setup for your robot. Ensure the vision sensor is correctly mounted and the Aruco marker board is within the sensor's field of view.

#### Capture Calibration Poses

Move the robot's end-effector to various positions and orientations, ensuring the Aruco marker board is visible in each pose. Capture multiple poses to improve calibration accuracy. Use the following command to start the calibration capture process:

```
roslaunch moveit_calibration_gui moveit_calibration_gui.launch
```

#### Collect Data

In the MoveIt! Calibration GUI, select the hand-eye calibration plugin and start collecting data. For each pose:

- Position the robot so the Aruco board is visible.
- Capture the pose by clicking the "Capture" button in the GUI.
- Repeat this process for multiple poses (at least 10-15).

#### Calibration Computation

Once enough data is collected, initiate the calibration computation. The software will use the captured poses to calculate the transformation matrix between the robot's end-effector and the vision sensor.

### 4.2.5 Mathematical Formulation

The hand-eye calibration problem can be described mathematically. Let $A_i$ represent the transformation from the robot base to the end-effector and $B_i$ represent the transformation from the camera to the marker board for each pose $i$. The goal is to find the transformation $X$ (from the end-effector to the camera) and $Y$ (from the robot base to the marker board), such that:

$$A_i X = Y B_i \tag{4.1}$$

This equation can be rewritten as a homogeneous transformation equation:

$$A_i X = Y B_i => X = A_i^{-1} Y B_i \tag{4.2}$$

By solving this equation for multiple poses, we can determine the unknown transformations $X$ and $Y$.

#### Solving the Calibration

The software uses optimization techniques to minimize the error between the observed and predicted transformations. This is typically done using least squares fitting or other numerical optimization methods.

### 4.2.6 Verification

After calibration, verify the results by positioning the robot's end-effector in known poses and checking the accuracy of the detected Aruco board position. Fine-tune the calibration if necessary.

### 4.2.7 Conclusion

Hand-eye calibration is a vital step in ensuring accurate robotic operations involving vision systems. By following the steps outlined in this chapter and using the MoveIt! tools, precise calibration can be achieved, leading to improved performance in robotic tasks.

This chapter covered the practical steps for hand-eye calibration using MoveIt!, described the role of Aruco marker boards, and explained the underlying mathematical formulation used by the software. Proper calibration enables robots to interact effectively with their environment, enhancing their operational capabilities.

# Chapter 5

# Results

"Somewhere, something incredible is waiting to be known."

*Carl Sagan*

This chapter shows the results obtained in the thesis.

# Chapter 6

# Conclusions

"Computer Science is no more about computers than
astronomy is about telescopes."

---

*E. W. Dijkstra*

This chapter summarizes the conclusions of the work.

# Bibliography

[1] Tullio Facchinetti. How to write a thesis in latex, 2021. http://robot.unipv.it/toolleeo, Last Access 2021.06.21.

[2] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics YOLO, 2023.