Karlsruhe Institute of Technology

Intelligent Process Automation and Robotics Lab

My Title is Long

Master Thesis

of

Zhuoyi Han

At the Department of Computer Science
Institute for Anthropomatics and Robotics (IAR) -
Intelligent Process Automation and Robotics Lab (IPR)

First reviewer:	Prof. Dr.-Ing. Torsten Kröger
Second reviewer:	Prof. Dr.-Ing. habil. Björn Hein
First advisor:	M.Sc. C
Second advisor:	M.Sc. D

xx. Month 20XX – xx. Month 20XX

Institute for Anthropomatics and Robotics (IAR) -
Intelligent Process Automation and Robotics Lab (IPR)
KIT Department of Informatics
Karlsruhe Institute of Technology
Engler-Bunte-Ring 8
76131 Karlsruhe

Zhuoyi Han
Bernhardstr.11
76131 Karlsruhe
uxehr@studen.kit.edu

Todo list

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, March 17, 2019

. . . . . .

(**Zhuoyi Han**)

Add Acknowledgments if you like!

Abstract

My Title is Long

English abstract.

Keywords: *Keywords, of, my, Thesis*

Zusammenfassung

Mein Titel ist lang

Deutsche Zusammenfassung

Stickwörter: *Die, Stichwörter, für, meine, Arbeit*

Contents

1. Introduction

See the section 9.

... Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain

all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2. State of the art

...

3. Methods

3.1. title

...

4. Technogical Fundations

This chapter introduces technological foundations fundamental for this thesis, which includes methods, models and tools that are used later in the thesis. The first section focus on the basic principle of Unity3D, which is used to build simulation environment for generating synthetic images. Then the second section shortly explain the essential knowledge of deep learning, especially convolution neural network(CNN), which used by modeling to recognize objects and predict the 3D-pose of them. Section 2.2 give an overview of the Kreas/Tensorflow framework, which will be used for training the CNN-model. Section 2.3 shows how to build a shallow neural network model in Keras and train it.

4.1. Unity3D background

Unity is a cross-platform game engine developed by Unity Technologies. Due to the motivation of transfer learning from simulation to real environment, a simulation scene in virtual environment need to be built and large numbers of synthetic image data of it are requisite. The unity engine gives users the ability to create virtual objects in 3D and offers a primary scripting API in C# to render images that are needed in this research. All these features shows that Unity is suitable for rendering requests and can simplify the works during modeling.

4.1.1. User interface

Unity provides a basic graphical interface to build the simulation environment, which includes five basic windows.

- The Scene View: where user work with game objects, including models, lights and colliders, to construct the user's scenes.
- The Game View: where user can preview and play simulation scene as a work in progress as user develop it.
- The hierarchy window: All of the game objects in the open scenes are listed by the hierarchy window in hierarchical order.
- The project window: where user imports, stores and edits the Asset files.
- The inspector window: It is context sensitive and displays all the properties of any selected game object, asset or setting.

All the modeling works can be done in the graphical user interface. A general method of modeling are divided into 4 steps. Firstly a new default model needs to be created in Unity or imported from other 3D modeling software. This new model will appear

in hierarchy window with other existent objects. Here the hierarchical relations can be adjusted. Secondly the dimensions and positions of new created model can be modified in inspector window, and new components (e.g. C# scripts, Mesh Renderer, etc.) can be added into the model. Then in scene window a arbitrary perspective could be set, and according to the local coordinate system the position relationship among different objects can be adjusted. Finally when all items are correctly set and all needed components are added on them, the simulation scene can be played and the progress can be monitored in game view window in real time.

4.1.2. Coordinate system in Unity

The world coordinate system is left handed (as Direct X) where x positive axis is right, y positive is up and z is positive into the screen.

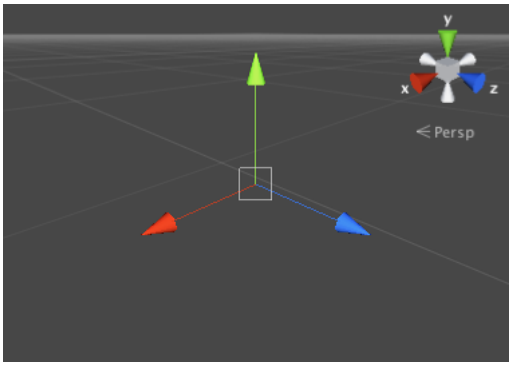


Figure 4.1.: Coordinate system in Unity

Screen coordinate system is bottom-up: (0,0) at bottom-left corner and (pixelWidth-1,pixelHeight-1) at right-top; x axis is positive right and y is positive up. The z position is in world units from the camera.
Viewport coordinate system is normalized and relative to the camera, so the bottom-left point is (0,0), the top-right is (1,1). The z position is in world units from the camera.
Finally UI coordinate system is top-down: the y coordinate varies from zero at the top edge of the window to a maximum at the bottom edge of the window. The upper-left point is (0,0); the bottom-right is (1,1).
There are many functions to convert between all these different coordinate systems

4.1.3. Basic concepts in scene

To build a basic scene in Unity 3D, some important core concepts need to be known. which include Game Object, Component, Script and etc. They form the main basic elements of unity and should be added into the scene to realize the needed functions and effects. Normally the Component and Script are sub-element of the Game Object.

- Scenes: Scenes contain the objects of simulation project. They can be used to create a main menu, individual levels, and anything else. Each unique Scene file is

a unique level. In each Scene, the environments, obstacles, decorations will be placed, and the project are designed and built in pieces.
A new Unity project will show a new Scene. The scene will be empty except for default objects- either an orthographic camera, or a perspective camera and a directional light.

- GameObjects: The GameObject is the most important concept in the Unity Editor. Every object in project is a GameObject. This means that everything which can be thought of to be in the project has to be a GameObject. However, a GameObject can't do anything on its own; The properties need to be given before it can become a character, an environment, or a special effect.
  A GameObject is a container; pieces are added to the GameObject container to make it into a character, a light, or whatever needed in project. Each added piece is called a component.
  Depending on what kind of object need to be created, different combinations of components can be added to a GameObject. A GameObject could be thought of as an empty cooking pot, and components are different ingredients that make up the recipe of project.

- Component: Components are the nuts & bolts of objects and behaviors in a game. They are the functional pieces of every GameObject.
  A GameObject is a container for many different Components. By default, all GameObjects automatically have a Transform Component. This is because the Transform dictates where the GameObject is located, and how it is rotated and scaled. Without a Transform Component, the GameObject wouldn't have a location in the word.

- Script: Scripting is an essential ingredient in all projects. They can be used to create graphical effects, control the physical behavior of objects and even implement a custom automatic system in the project. In next subsection, basic knowledge of scripting will be introduced to show how the script runs in unity and which basic functions are included in it.

4.1.4. Basic knowledge of script

Unity supports the C# programming language, which is an industry-standard language similar to Java or C++.
A script makes its connection with the internal workings of Unity by implementing a class which derives from the built-in class called MonoBehaviour. A class is like a kind of blueprint for creating a new Component type that can be attached to GameObjects. The core elements to realize different effects are the Event Functions. A script in Unity is not like the traditional idea of a program where the code runs continuously in a loop until it completes its task. Instead, Unity passes control to a script intermittently by calling certain functions that are declared within it. Once a function has finished executing, control is passed back to Unity. These functions are known as event functions since they are activated by Unity in response to events that occur during gameplay. Unity uses a naming scheme to identify which function to call for a particular event. There are four basic types of events:

- Regular Update Events:

1. Update function: The Update function is the main place for making changes to position, state and behavior of objects in the scene just before each frame is rendered. Update is called before the frame is rendered and also before animations are calculated.
2. FixedUpdate function: The physics engine also update in discrete time steps in a similar way to the frame rendering. A separate event function called FixedUpdate is called just before each physics update.
3. LateUpdate function: It is also useful sometimes to be able to make additional changes at a point after the Update and FixedUpdate functions have been called for all objects in the scene and fter all animations have been calculated.

- Initialization Events:
  1. Start function: It is called before the first frame or physics update on an object.
  2. Awake function: The Awake function is called for each object in the scene at the time when the scene loads. All the Awakes will have finished before the first Start is called. This means that code in a Start function can make use of other initializations previously carried out in the Awake phase.
- GUI event: Unity has a system for rendering GUI controls over the main action in the scene and responding to clicks on these controls. This code is handled somewhat differently from the normal frame update and so it should be placed in the OnGUI function, which will be called periodically.
- Physics events: The physics engine will report collisions against an object by calling event functions on the object's script.

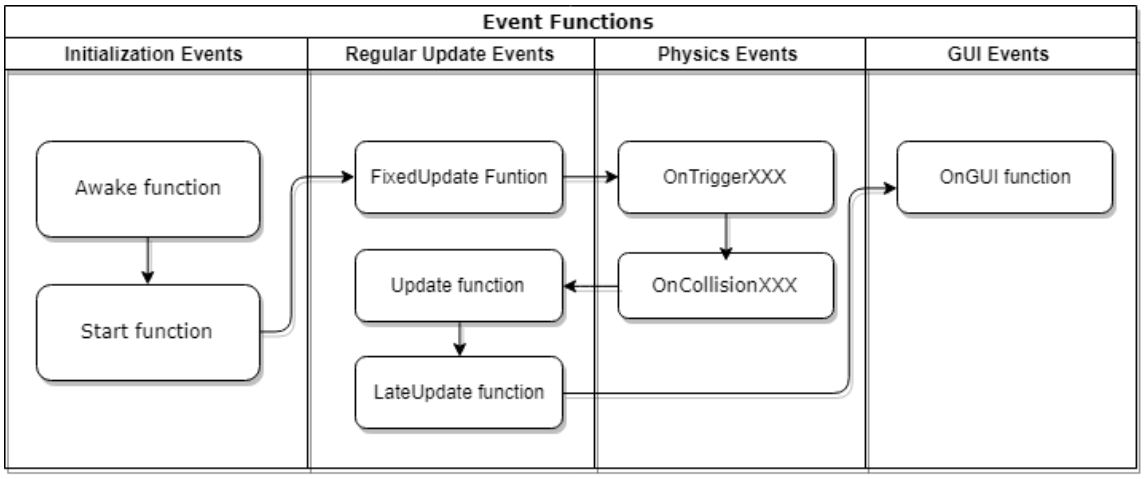


Figure 4.2.: Coordinate system in Unity

...

5. Implementation

...

Please add some figures

6. Results

…

7. Discussion

…

8. Conclusion

…

Appendix

A. First Appendix Section

Figure A.1.: A figure

...

List of Figures

List of Tables

Listings

List of Algorithms

```
@mastersthesis{Zhuoyi Han_xx. Month 20XX,
	author = {Zhuoyi Han},
	editor = {M.Sc. C, M.Sc. D},
	ipr-thesis = Master Thesis,
	keywords = {Keywords, of, my, Thesis},
	location = {Karlsruhe, Germany},
	month = ,
	pages = ,
	school = {Karlsruhe Institute of Technology},
	title = {My Title
is Long},
	year = {xx. Month 20XX}
}
```

9. How to use this Template

IMPORTANT: This chapter will disappear when you add final parameter on the document. See section 9.14.

9.1. Getting Started

Initially you **should only edit** the `My_document_info.tex` with important data regarding your work.

Add **content** in files in `Content` folder.

Add **bibliography** in the file `Bibliography/my_thesis_bibliography.bib` or just add a file from your supervisior in the `Bibliography` folder and reference it in the `\mybibliographyfiles` command in the `My_document_info.tex` file.

As an useful aid in all scientific work following book is recommended: [**?**].

9.2. Inline lists

My robot can: (i) forward and backward movements, (ii) sidewards movements, (iii) rotation along any curve in space, (iv) place of artificial forces along paths.

(1) the independently controllable wheels; (2) the rechargeable battery pack; (3) the Sick LMS100 laser range scanner; (4) the force-torque sensor; (5) the handlebar for controlling the robotic device

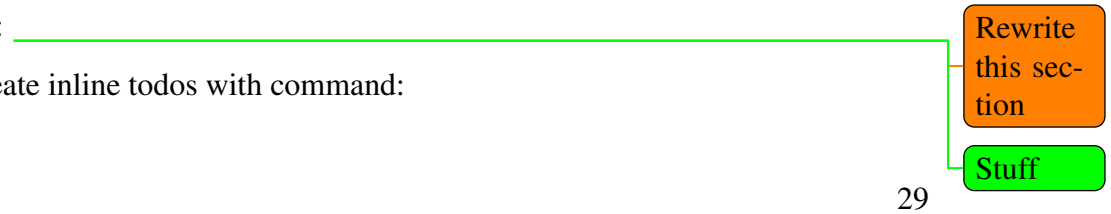
`https://ctan.math.illinois.edu/macros/latex/contrib/enumitem/enumitem.pdf`

9.3. Todos

Todo dommand can be used in multiple form and paramters set. You can set todos on the right side with commands:

```
\todo{Rewrite this section}
\todo[color=green]{Stuff}
```

which render as:



You can also create inline todos with command:

```
\todo[inline]{Rewrite this section}
\todo[inline,color=green]{Rewrite this section}
\todoin{Stuff}
```

which rendrs as:

Rewrite this section

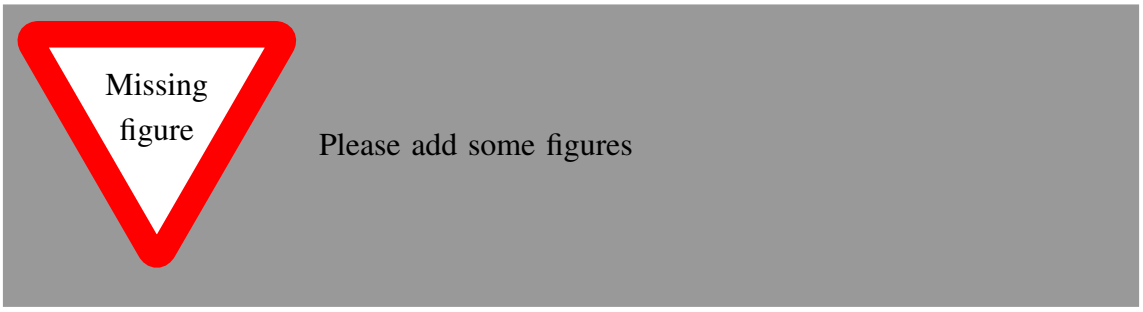
Rewrite this section

Stuff

One can also use command for figure placeholder with command:

```
\missingfigure{Please add some figures}
```

which renders as:



9.4. Glossaries and Acronyms

Please use `glossaries` package for this. See *documentation*.

Example (Acronym):

```
\newacronym{ipr}{IAR-IPR}{Institute for Anthropomatics and Robotics - Intellig
```

is used by

```
\gls{ipr}
```

rendering as “Institute for Anthropomatics and Robotics - Intelligent Process Control and Robotics (IAR-IPR)”, on the first use and as “IAR-IPR” on every following use. For further feature see *documentation*.

Please keep in mind that one has to call *external commands* for glossaries to work.

9.5. Nomenclature

For more details see *example*.

Use following command: \nomenclature{IAR-IPR}{Institute for Anthropomatics and Robotics (IAR) - Intelligent Process Control and Robotics (IPR)}

9.6. SI Units

Please use `siunitx` package for this. See: `https://ctan.org/pkg/siunitx`

9.7. Tables

Table .1.: Tables have caption on top.

Object	Speed $[cm/s]$	Inner LR $[cm]$	Inner UR $[cm]$
Pitcher	real	n/a	5.65
	4.60	3.71 ± 0.67	5.09 ± 2.23
	10.64	3.55 ± 0.57	6.14 ± 0.69
Cookie O	real	7.55	7.55
	4.60	6.98 ± 0.27	6.98 ± 0.27
	10.64	6.77 ± 0.26	6.77 ± 0.26

Use `\longtable` for tables over multiple pages. See documentation.

9.8. Figures

Figure .1.: Figures have caption under. If you use figures from other work, do not forget to reference them [**?**].

9.9. Citation

9.9.1. Multiple citations

Use multiple citation like this:

```
\cite{deininger2005studien, deininger2005studien}
```

rendered as "[**?** **?**]".

9.9.2. More powerfull cite commands: `\citet` and `\citep`

For comprehensive description please check *the natbib documentation*.

Rather than using the awkward construction[1]

```
\cite{deininger2005studien} describes...
```

rendered as "[**?**] demonstrated...," or the inconvenient

```
Deininger \cite{deininger2005studien} describes...
```

rendered as "Deininger [**?**] demonstrated...", one can write

```
\citet{deininger2005studien} describes...
```

which renders as "**?**] demonstrated..." and is both easy to write and much easier to read.

Citing specific chapter:

? , sec. III]

[**?** , sec. III]

For more examples check *the natbib documentation*.

9.10. Using Hyperlinks

Please use the ability of PDF viewers to interpret hyperlinks[2], specifically to allow each reference in the bibliography to be a link to an online version of the reference. As an example, if you were to cite "Passive Dynamic Walking" [**?**], the entry in the bibtex would read:

```
@article{McGeer01041990,
  author = {McGeer, Tad},
  title = {\href{http://ijr.sagepub.com/content/9/2/62.abstract}{Passive Dynamic Walking}},
  volume = {9},
  number = {2},
  pages = {62-82},
  year = {1990},
  doi = {10.1177/027836499000900206},
  URL = {http://ijr.sagepub.com/content/9/2/62.abstract},
  eprint = {http://ijr.sagepub.com/content/9/2/62.full.pdf+html},
  journal = {The International Journal of Robotics Research}
}
```

and the entry in the compiled PDF would look like:

[1] Tad McGeer. Passive Dynamic Walking. *The International Journal of Robotics Research*, 9(2):62–82, 1990.

where the title of the article is a link that takes you to the article on IJRR's website.

Also use this for adding links into text as done in the [2]. For more information see documentation on wikibooks. The `hyperref` package is already configured for this document in `KIT_document_setup.tex` file.

[1] The example is from the template for the conference *Robotic Science and Systems*.
[2] The example is from the template for the conference *Robotic Science and Systems*.

9.11. Equations

Use numbered equations:

$$m \cdot \ddot{x}(t) + d \cdot \dot{x}(t) = F(t) \tag{9.1}$$

9.12. Inline comments

Use command `\comment{}` for inline comments.

9.13. After Review marking

Use command `\afterReview{}` for marking text parts as changed.

9.14. Finalizing the Document

Please check here: `https://github.com/KITrobotics/Latex_Template/blob/master/README.md#finalizing-document`