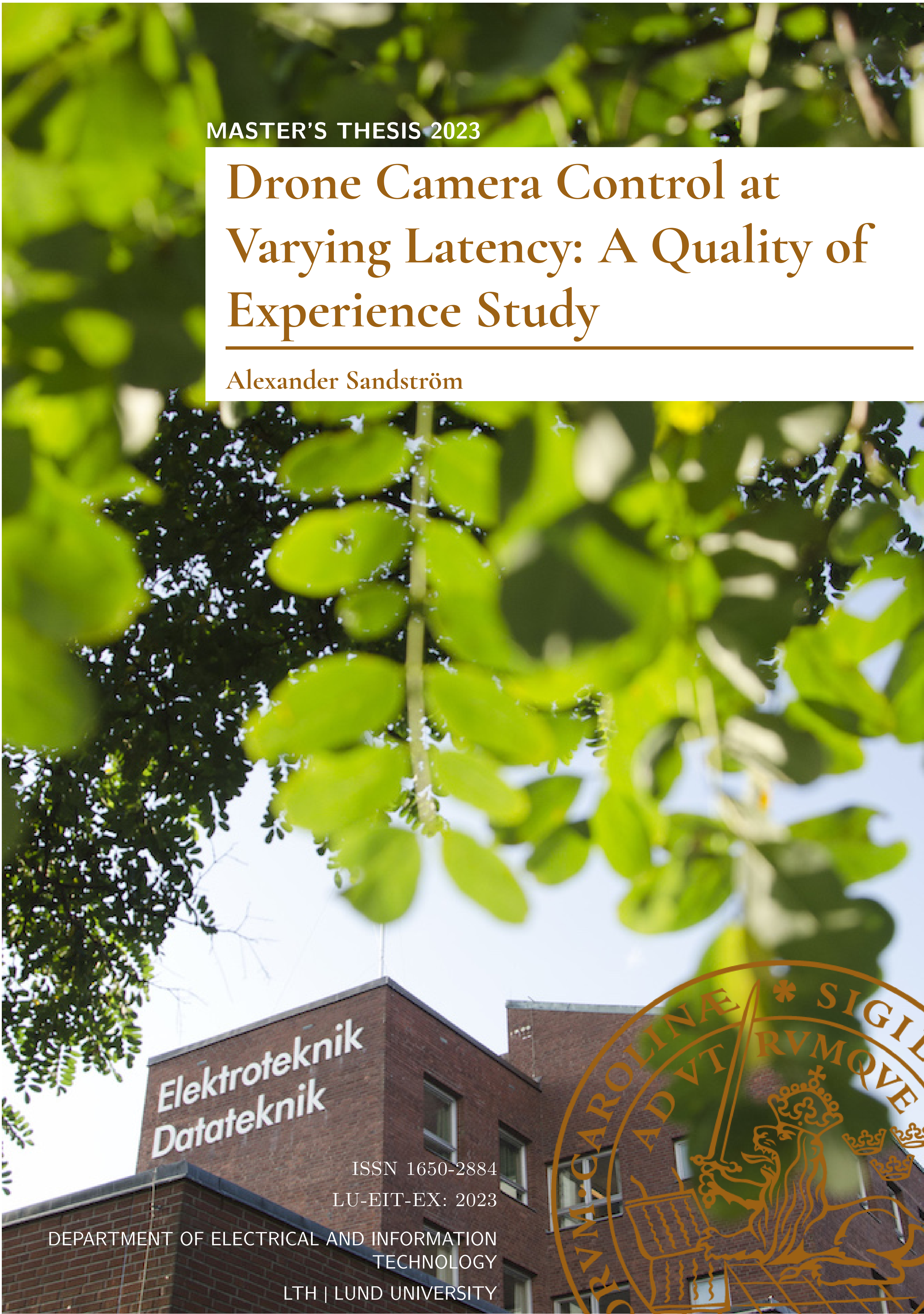
MASTER'S THESIS 2023

Drone Camera Control at Varying Latency: A Quality of Experience Study

Alexander Sandström



ISSN 1650-2884
LU-EIT-EX: 2023

DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
LTH | LUND UNIVERSITY

EXAMENSARBETE
Elektro- och Informationsteknik

LU-EIT-EX: 2023

Drone Camera Control at Varying Latency: A Quality of Experience Study

Kamerastyrning på en Drönare vid Varierande Latens: en Studie i Operatörsupplevelse

Alexander Sandström

Drone Camera Control at Varying Latency: A Quality of Experience Study

Alexander Sandström
alexander.h.sandstrom@gmail.com

April 26, 2023

Master's thesis work carried out at the Swedish Sea Rescue Society and the Department of Electrical and Information Technology, Lund University.

Supervisors: Fredrik Falkman, fredrik.falkman@ssrs.se
William Tärneberg, william.tarneberg@eit.lth.se

Examiner: Maria Kihl, maria.kihl@eit.lth.se

Abstract

Your abstract should capture, in English, the whole thesis with focus on the problem and solution in 150 words. It should be placed on a separate right-hand page, with an additional *1cm* margin on both left and right. Avoid acronyms, footnotes, and references in the abstract if possible.

Leave a *2cm* vertical space after the abstract and provide a few keywords relevant for your report. Use five to six words, of which at most two should be from the title.

Keywords: MSc, BSc, template, report, style, structure

Acknowledgements

If you want to thank people, do it here, on a separate right-hand page. Both the U.S. *acknowledgments* and the British *acknowledgements* spellings are acceptable.

Contents

1 Introduction **7**
- 1.1 Research Motivation . . . 7
- 1.2 SSRS Needs . . . 7
- 1.3 Scope . . . 8

2 Background **9**
- 2.1 Swedish Sea Rescue Society . . . 9
  - 2.1.1 Innovation . . . 9
  - 2.1.2 Eyes On Scene . . . 9
- 2.2 Quality of Experience . . . 10
- 2.3 Previous Work . . . 10

3 Test Bed **11**
- 3.1 System Overview . . . 11
  - 3.1.1 Components . . . 11
- 3.2 Hardware . . . 12
  - 3.2.1 Flight controller . . . 12
  - 3.2.2 Raspberry Pi 4 . . . 13
  - 3.2.3 Gimbal . . . 13
- 3.3 Software . . . 13
  - 3.3.1 Firmware: ArduPlane . . . 13
  - 3.3.2 Protocol: MAVLink . . . 13
  - 3.3.3 Software: MAVProxy . . . 13
  - 3.3.4 Package: pymavlink . . . 14
  - 3.3.5 UV4L . . . 14
  - 3.3.6 Janus Server . . . 14
  - 3.3.7 Gimbal Control Interface . . . 14
  - 3.3.8 Result Script . . . 14
- 3.4 Hoverboard robot . . . 14

4 Experiment **15**
4.1 Experimental Setup . . . 15
4.1.1 Subject selection and recruitment . . . 15
4.2 Experimental Procedure . . . 15
4.2.1 Introduction . . . 16
4.2.2 Task description . . . 16
4.2.3 Test Procedure . . . 17
4.3 Evaluation . . . 17
4.3.1 Quantative Evaluation . . . 18
4.3.2 Qualitative Evaluation . . . 18
4.3.3 Auto-follow mode . . . 18

5 Results **19**

6 Discussion **21**
6.1 System Architecture (SKRIVA DET SOM QOS?) . . . 21
6.1.1 Main goals . . . 21
6.1.2 WebRTC and data channels . . . 21

7 Conclusion **23**

8 Future Work **25**

References **27**

Appendix A About This Document **31**
A.1 Page Size and Margins . . . 32
A.2 Typeface and Font Sizes . . . 32
A.2.1 Headers and Footers . . . 32
A.2.2 Chapters, Sections, Paragraphs . . . 33
A.2.3 Tables . . . 33
A.2.4 Figures . . . 34
A.3 Mathematical Formulae and Equations . . . 34
A.4 References . . . 35
A.5 Colours . . . 35

Appendix B Language **37**
B.1 Style Elements . . . 37

Appendix C Structure **39**

Chapter 1

Introduction

In this thesis, gimbal control for drones running the ArduPilot firmware is implemented. The gimbal control software is used as a test bed evaluating quality of experience of a drone operator at different latencies. Furthermore, it was also integrated in the running flight stack of SSRS and tested in a real flight scenario (hopefully).

1.1 Research Motivation

With the promise of real-time remote control over 5G, the systems of tomorrow will be heavily reliant on the uptime and latency of the network, making these systems exposed to vulnerabilities should the network experience high contingency or failure. Therefore, Having a system that can operate in a degraded state, i.e. with high latency, is a necessity. Furthermore, when 5G is more widely adopted and stable, Internet Service Providers (ISPs) might offer plans with different latencies, giving the user a choice of how much they want to pay for latency that their particular application needs.

In the emerging field of research around drones and remote control, there is a clear correlation between latency and the operator's experience. However, a comparison of different types of controls and their effect on the operator at different latencies are, as far as the student's knowledge goes, not documented in the current body of research.

1.2 SSRS Needs

The technical challenges that SSRS faces align very well with the challenges that are posed in general so-called "Industry 4.0", and the highly dynamic environment in which the drone will operate along with the area of operation being far from any land-based infrastructure make a good case to employ learning outcomes from previous QoE research.

Furthermore, SSRS does not have an estimate of the latency of the video feed from the drone, making it difficult to determine how the camera should be controlled, be it by swiping, pressing the arrow keys or setting a location on a map for it to look at. In the emerging field of research around drones and remote control, there is a clear correlation between latency and the operator's experience. However, a comparison of different types of controls and their effect on the operator at different latencies are, as far as the student's knowledge goes, not documented in the current body of research.

1.3 Scope

The scope of the thesis work is to develop a gimbal control prototype using the ArduPilot firmware running on real drone hardware. This prototype is then to be used both as a test bed for QoE experiments and as a proof of concept for the integration of the gimbal control in the running flight stack of SSRS.

Chapter 2
Background

In this chapter a breif background will be given of the Swedish Sea Rescue Society along with the drone project that this work is a part of. Then, the research field of Quality of Experience will be introduced, along with the related previous work.

2.1 Swedish Sea Rescue Society

As can be read on their website [8], the Swedish Sea Rescue Society (SSRS) is a non-profit organization that was founded at a conference in Stockholm in 1906 due to Sweden receiving criticism of its' poor sea rescue. Today, it is a foundation with 40 employees and over 143 000 members, and with 2400 volunteers manning their 260 rescue vessels, they carry out around 90% of all sea rescues in Sweden all year around.

2.1.1 Innovation

As stated in their statutes [9], the mission of SSRS is not solely to carry out these rescue missions but to also innovate and collaborate in the area of maritime rescue as well as other aiding activities in society as a whole. As a result of this, the drone project that this thesis is a part of has been conceived along with other innovation projects such as foiling rescue boats and improved rescue vehicles [7].

2.1.2 Eyes On Scene

Currently, SSRS has a drone connected to the mobile network that can fly towards and loiter around a set of waypoints given on a map, while also providing video from the gimbal-mounted camera. The drone flies with a hardware module running the autopilot software

ArduPilot which also supports camera gimbal control. The mechanical gimbal assembly supports three degrees of freedom, however, the software necessary for remotely controlling the camera gimbal is not implemented.

2.2 Quality of Experience

Quality of Experience is an emerging research field that is concerned with the user's experience in multimedia systems. It is an inherently multidisciplinary field that has close ties to the field of User Experience (UX) and Human-Computer Interaction (HCI). QoE is also closely related to the field of Quality of Service (QoS), which is concerned with the technical aspects of a system but aims instead at evaluating the subjective experience through user experiments rather than qualitative measurements.

In the white paper [11], Brunström et al. make a working definition of Quality of Experience:

> *Quality of Experience (QoE) is the degree of delight or annoyance of the user of an application or service. It results from the fulfillment of his or her expectations with respect to the utility and / or enjoyment of the application or service in the light of the user's personality and current state.*

2.3 Previous Work

W. Tärneberg et al. [13] present a QoE study with industrial equipment on an excavation site where experienced operators controlled their usual equipment remotely at different latencies.

K. Brunnström et al. [12] performed a study on log lifting using a head-mounted display system. In the study, latency is introduced both in the display's response to movement as well as the controls. The display delay was found to have a strong effect on nausea, but an observable effect on controller latency could only be found at latencies above 800 ms.

Chapter 3

Test Bed

This chapter aims to introduce the reader to the test bed and each of it's subsystems

3.1 System Overview

Due to the amount of software components executing on different machines in this control system, this section aims to provide an initial overview for the reader to understand its architecture.

First, the system will be divided into components, then each of the components will be described in more detail. A visual overview of the system is shown in Figure 3.1.

3.1.1 Components

The complete system can be divided in two, where on component is the video feed and the other is the control of the gimbal, although some components are used in both.

The video feed involves the following components:

- Raspberry Pi Camera Module: records video
- Raspberry Pi 4: sends the video feed to the Janus server
- Janus Server: establishes a WebRTC connection to the web browser
- Web Browser: displays the video feed

The control of the gimbal involves the following components:

- Raspberry Pi 4: relays messages between the flight controller and the ground station, often referred to as companion computer.

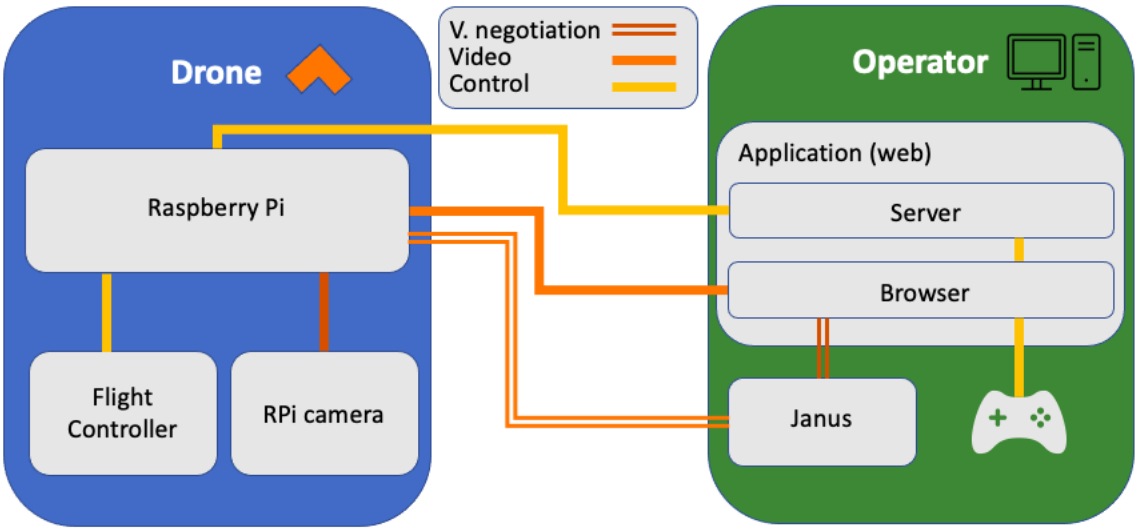


Figure 3.1: An overview of the entire system. On the left, the drone and its' components including the RPi, RPi camera module and the flight controller are shown. On the right, the ground station and its' components including the web application, Janus server and the controller are shown. The colour of the connections represent what data they transfer, where orange is video, orange double-line is video negotiation and yellow is control.

- Flight Controller: controls all flight and gimbal functions
- Web Browser: user inputs from joystick
- pymavlink: Python library to send commands to flight controller
- MAVProxy: software running on a companion computer to relay messages between the flight controller and the ground station

3.2 Hardware

This chapter will briefly go through the main components of hardware that have been used in this thesis.

3.2.1 Flight controller

The flight controller is the brain of the aircraft and houses the following components:

- Central Processing Unit (CPU)
- Ardupilot firmware (described in 3.3.1)
- accelerometer

- connector to external GPS receiver
- servo-connectors for aircraft control surfaces and gimbal

3.2.2 Raspberry Pi 4

The Raspberry serves as the onboard computer, commonly known as the "companion computer" in the context of drone hardware. It is connected to the flight controller via USB and relays messages between the flight controller and the ground station over its connection to the Internet. In the SSRS drone, the Raspberry has a 4G modem which is what allows the drone to fly beyond visual line-of-sight (BVLOS).

3.2.3 Gimbal

The gimbal used in the experiment is designed and manufactured by Fredrik Falkman at the SSRS. It is a 3D-printed design that has three degrees of freedom made possible by three servos connected to drive belts that control each axis of the camera. The servos are connected to the flight controller which also provides them with power. The gimbal also hosts a Raspberry Pi Camera Module which is directly connected to the Raspberry with a ribbon cable.

3.3 Software

There were many different softwares involved in this project, and this chapter will briefly go through the main ones that have been used in this thesis. A brief description of the program for the particular test bed will also be included.

3.3.1 Firmware: ArduPlane

The firmware running on the flight controller is called ArduPlane, which is part of the open-source autopilot software suite ArduPilot that enables the creation of unmanned, autonomous vehicles [2]. It is developed by a community of developers and is available on GitHub [1].

3.3.2 Protocol: MAVLink

As can be read on their website [4], MAVLink is a lightweight protocol suited for communication with drones and between drone components. It has a byte overhead of only 14 bytes and allows for concurrent communication between up to 255 systems.

3.3.3 Software: MAVProxy

MAVProxy is a software running on the companion computer whose main task is to relay MAVLink messages between the flight controller and the ground station [5]. This is required as a mobile network modem cannot be connected directly to the flight controller.

3.3.4 Package: pymavlink

This package is the Python interface to the MAVLink protocol, available at [6]. In this thesis, it is used to send and receive MAVLink messages on the GCS.

3.3.5 UV4L

UV4L is a video streaming server that supports real-time communication protocols such as WebRTC [10]. In this thesis, it is the software used to stream the local video feed over the web.

3.3.6 Janus Server

As described on their webpage [3], Janus is a plugin-based software that helps establish WebRTC connections. In this thesis it is used to establish the connection between the UV4L server and the web browser, enabling the peer-to-peer connection between the two devices.

3.3.7 Gimbal Control Interface

Using the software components previously named, a program serving as the gimbal interface for the operator was implemented in Python. The software takes joystick inputs from a PlayStation-controller and sends messages updating the position of the gimbal accordingly while displaying the video feed in a window. The same software was also modified so that it could record the video being displayed to the operator. Finally, a simple PI-controller was also implemented into the software to autonomously follow an object.

3.3.8 Result Script

In order to get results from the recorded video of each subject a script was implemented that was run on each trial, measuring the error of the operator in each video frame and saving it to a file.

3.4 Hoverboard robot

To make the object that was to be followed move, a slightly modified hoverboard was used. The hoverboard has two additional wheels so that it is stable, and can be controlled by a game controller or follow a set of points using coordinates. The hoverboard uses measurements from it's wheels and LIDAR to locate itself which makes it possible to have the hoverboard run in the exact same route over and over again with little error.

Chapter 4
Experiment

This chapter will describe the experimental setup, procedure and evaluation.

4.1 Experimental Setup

The experiment was set up in a lab where the subject and the conductor sits at a desk with a high shelf behind them obscuring the view towards the rest of the lab. Behind the shelf the robots used in the experiment are located. An illustration of the lab setup is provided in figure 4.2.

An image of the hoverboard and the camera is shown in image 4.1.

4.1.1 Subject selection and recruitment

The subjects were recruited mostly from the conductor's network of contacts, as in order for the subjects to be covered by insurance during the experiment, the subjects had to be either be students or employed by Lund University. Some of the participants came spontaneously while others booked a time in a spreadsheet provided by the conductor.

There was no compensation for the subjects other than home baked goods and a hot beverage, which were offered at the beginning of each experiment.

4.2 Experimental Procedure

The experimental procedure is summarized in the table 4.1. In this section each of the steps will be described in more detail.

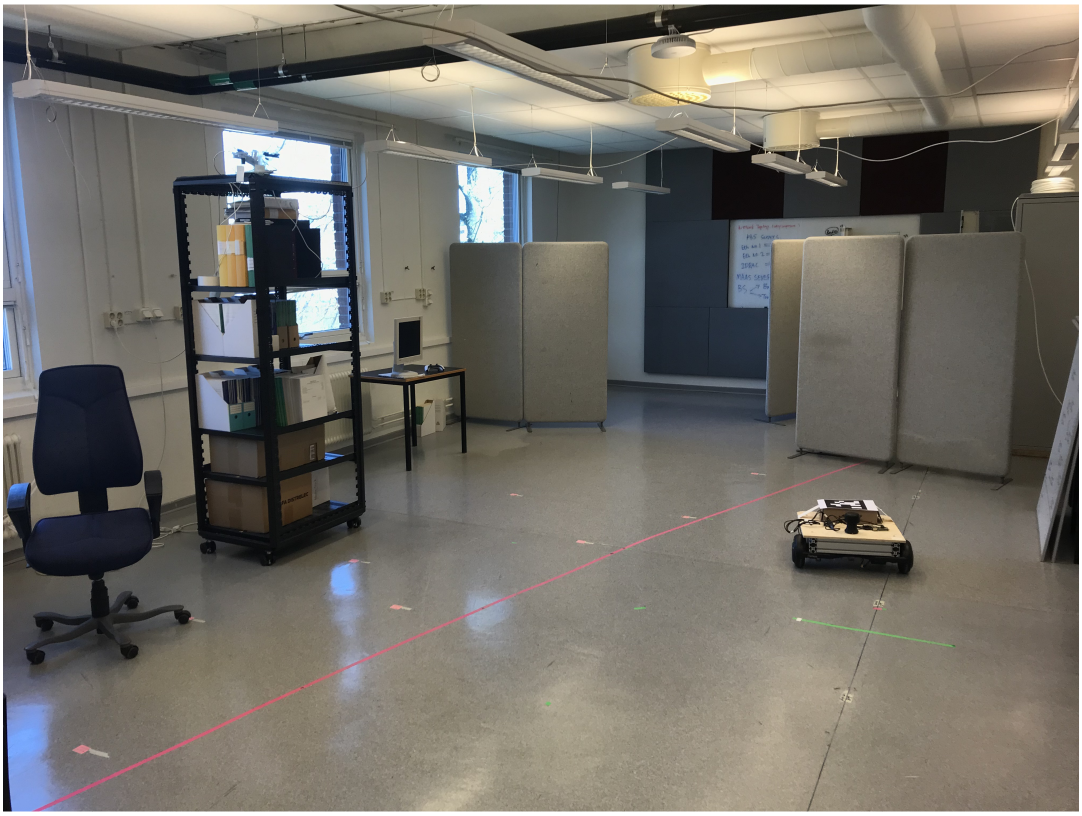


Figure 4.1: A picture of the test bed. On the left side there is a server rack with the drone box mounted on top. The hoverboard can be seen on the right side of the picture.

4.2.1 Introduction

The experiment starts with the subject being seated at the desk and where he or she will get to read and sign the consent form. After this was signed the conductor clarified that the subject was free to ask any question during the experiment and could choose to interrupt it without declaring any reason.

When providing their background information the subject gets to choose a four digit code that from that point is the only identifier of the subject and it's results. If the subject booked a time slot in the spreadsheet, they were at this point removed from it.

4.2.2 Task description

The subjects were shown a printed apriltag with a cross on it marking the spot where to aim. The particular apriltag printed for this experiment had two white and two black squares meeting in the middle, thus making the center easier to identify.

The subject was then given a chance to try the system, and was given the controller and presented with the view. The hoverboard was started and ran for one lap in the same path that it would run during the test.

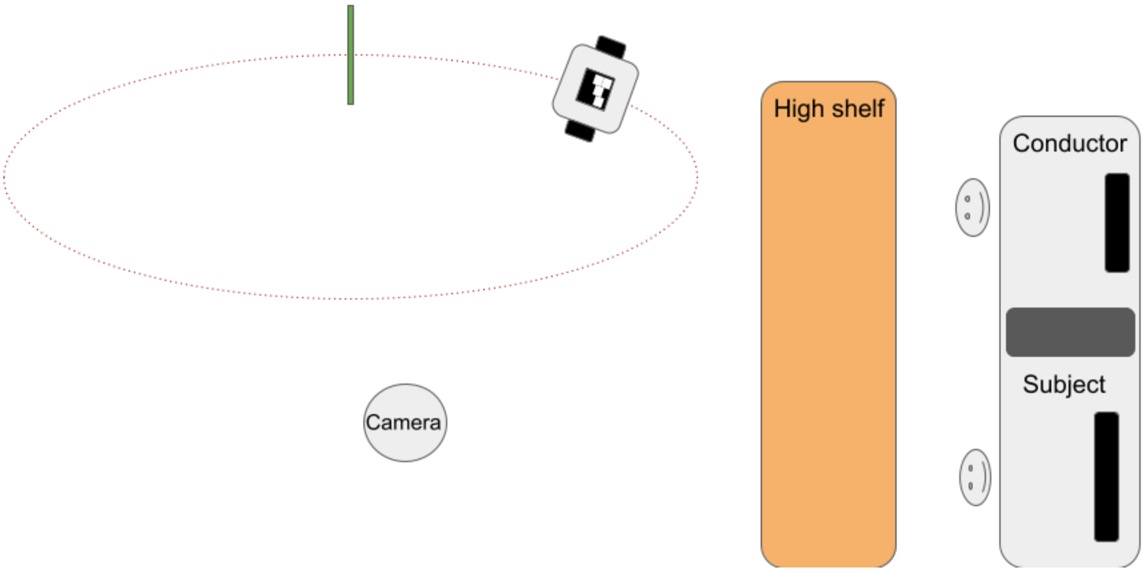


Figure 4.2: An overview of the experimental setup. To the right the experiment conductor and subject by a desk facing right. On the left side, behind the shelf, the test bed including the camera and the hoverboard are located. The camera is mounted at a height of 2.1 meters with the gimbal facing down.

4.2.3 Test Procedure

After the task description and warm-up the tests commenced. The added latencies tested were 0, 200, 300, 400 and 600 and the order in which they were given to the subject was random. At the beginning of each trial the interface was restarted with one of the latencies induced in the system. The subject was then asked to put the red marker in the middle of the screen in the middle of the apriltag. When the subject confirmed that they were ready to start the conductor started the hoverboard. The subject followed the hoverboard for two laps, which took about 2 minutes in total.

After the laps were completed the interface was shut down and the subject was asked to rate the system by answering four different questions on a scale from 1 to 5. When the subject had answered the questions the next trial was started.

At the end of the fifth and last trial the subject got to fill in the sickness questionnaire was also asked a few interview questions by the conductor.

4.3 Evaluation

The following two sections will describe the quantative and qualitative methods that are used to evaluate the performance of the system.

Table 4.1: Task Timeline

Time (approx.)	Task	Instruction
3	Coffee and cookies	
5	Introduction	Tell the subject about what the study is about. Gather personal data + consent
2	SSQ	
3	Experiment walkthrough	What is going to happen Description of task Where to aim exactly on the apriltag
2	Warm-up	The subject gets to try the setup for one lap
10	Tests	5 tests with added latencies [0, 200, 300, 400, 500] in random order
2	SSQ	
3	Interview	
30	Total	

4.3.1 Quantative Evaluation

To quantify the performance of the test subject a printed apriltag will be attached on top of the robot that the subject will try to follow. The view from the camera will be recorded during the experiment and then an image recognition software will be used to detect the position of the robot.

This will make it possible to calculate the distance of the recognized object to the center of the video feed which then can be used to measure the performance of the test subject.

An Apriltag is a QR-code that contains bigger squares, thus making it easier to recognize for a computer vision algorithm

4.3.2 Qualitative Evaluation

After each trial set the subject will answer questions regarding their perceived quality of experience with the system. They will also get to answer a sickness questionnaire, following standard questionnaire provided by [**?**].

4.3.3 Auto-follow mode

As a proof of concept, an auto-follow mode was implemented in the application using real-time object detection of the video feed. Although it is not optimal to have that high latency introduced in the control-loop, it is of interest as the resource heavy object detection algorithm does not stress the hardware on the drone. The mode is evaluated with the same methods as the rest of the experiment, although the user is just passively watching the feed.

Chapter 5
Results

Chapter 6
Discussion

6.1 System Architecture (SKRIVA DET SOM QOS?)

Although being a bit outside the research scope of the project, a natural consequence has been to evaluate the strengths and weaknesses of different system architectures that could be implemented for real drone missions. This section will bring up some of the requierments from SSRS along with the particular architectures considered for the application of sea rescue.

6.1.1 Main goals

Being a real-time application over the Internet, there are many parts of the system that can introduce latency and efforts to reduce it can be boiled down to two actions:

- reducing the amount of processing agents between the operator and what is being controlled
- reducing the amount of processing time at the processing agents

However, while one particular architecture might meet requierments in one area, for example latency, it can make other parts of the system more complex or computationally expensive. The task is inherently complex as there are many tradeoffs that have to be balanced and the requierments of the particular application has to be very clear. Difficulty of implementation as well as extendability are also important factors to consider.

6.1.2 WebRTC and data channels

The use of WebRTC along with Janus enables two units to talk to one another directly over the web, removing a potential server that has to process the video feed and the relay it forward. However, if one would like to record the video and save it on a server the feed now has to be redirected to the server from either the system of the end user or the drone itself.

One part of WebRTC that was of particular interest during the implementation was data channels. These are meant to be utilized to send small amounts of data along with the videopackets utilizing the already established P2P to transmit for example a time stamp or control commands. In UV4L it is possible to check a box that automatically ties the gyroscope of a device to the output of three servos. Since the control in this project goes through the Ardupilot firmware one would need to implement a relay running on the companion computer, which there was not time for in the implementation part of this project.

Chapter 7
Conclusion

Chapter 8

Future Work

References

[1] Ardupilot github, ardupilot. https://github.com/ArduPilot/ardupilot. Accessed: 2023-02-13.

[2] Ardupilot org, ardupilot. https://ardupilot.org/. Accessed: 2023-02-13.

[3] Janus, meetecho. https://janus.conf.meetecho.com/. Accessed: 2023-03-10.

[4] Mavlink. https://mavlink.io/en/. Accessed: 2023-02-13.

[5] Mavproxy. https://ardupilot.org/mavproxy/. Accessed: 2023-03-28.

[6] pymavlink, github. https://github.com/ArduPilot/pymavlink. Accessed: 2023-02-13.

[7] Surtsey innovation projects. http://www.surtsey.org/. Accessed: 2023-02-03.

[8] Swedish sea rescue society, information in english. https://www.sjoraddning.se/information-english. Accessed: 2023-02-01.

[9] Swedish sea rescue society, statues. https://www.sjoraddning.se/sites/default/files/stadgar_ssrs_antagna_2018.pdf. Accessed: 2023-02-03.

[10] Uv4l, linux projects. https://www.linux-projects.org/uv4l/. Accessed: 2023-03-10.

[11] Kjell Brunnström, Sergio Ariel Beker, Katrien de Moor, Ann Dooms, Sebastian Egger, Marie-Neige Garcia, Tobias Hossfeld, Satu Jumisko-Pyykkö, Christian Keimel, Mohamed-Chaker Larabi, Bob Lawlor, Patrick Le Callet, Sebastian Möller, Fernando Pereira, Manuela Pereira, Andrew Perkis, Jesenka Pibernik, Antonio Pinheiro, Alexander Raake, Peter Reichl, Ulrich Reiter, Raimund Schatz, Peter Schelkens, Lea Skorin-Kapov, Dominik Strohmeier, Christian Timmerer, Martin Varela, Ina Wechsung, Junyong You, and Andrej Zgank. Qualinet White Paper on Definitions of Quality of Experience, March 2013. Qualinet White Paper on Definitions of Quality of Experience Output from the fifth Qualinet meeting, Novi Sad, March 12, 2013.

[12] Kjell Brunnström, Elijs Dima, Tahir Qureshi, Mathias Johanson, Mattias Andersson, and Mårten Sjöström. Latency impact on quality of experience in a virtual reality simulator for remote control of machines. *Signal Processing: Image Communication*, 89:116005, 2020.

[13] William Tärneberg, Omar Hamsis, John Hedlund, Kjell Brunnström, Emma Fitzgerald, Andreas Johnsson, Viktor Berggren, Maria Kihl, Akhila Rao, Rebecca Steinert, and Caner Kilinc. Towards intelligent industry 4.0 5g networks: A first throughput and qoe measurement campaign. 2020.

Appendices

Appendix A
About This Document

The following environments and tools were used to create this document:

- operating system: Mac OS X 10.14
- tex distribution: MacTeX-2014, `http://www.tug.org/mactex/`
- tex editor: Texmaker 5.0.2 for Mac, `http://www.xm1math.net/texmaker/` for its XeLaTeX flow (recommended) or pdfLaTeX flow
- bibtex editor: BibDesk 1.6.3 for Mac, `http://bibdesk.sourceforge.net/`
- fonts `cslthse-msc.cls` document class):

  for XeLaTeX: TeX Gyre Termes, TeX Gyre Heros, `TeX Gyre Cursor` (installed from the TeXLive 2013)

  for pdfLaTeX: TeX Gyre font packages: tgtermes.sty, tgheros.sty, tgcursor.sty, gtx-math.sty (available through TeXLive 2013)
- picture editor: OmniGraffle Professional 5.4.2

A list of the essential LaTeXpackages needed to compile this document follows (all except `hyperref` are included in the document class):

- `fontspec`, to access local fonts, needs the XeLaTeX flow
- `geometry`, for page layout
- `titling`, for formatting the title page
- `fancyhdr`, for custom headers and footers
- `abstract`, for customizing the abstract

- `titlesec`, for custom chapters, sections, etc.
- `caption`, for custom tables and figure captions
- `hyperref`, for producing PDF with hyperlinks
- `appendix`, for appendices
- `printlen`, for printing text sizes
- `textcomp`, for text companion fonts (e.g. bullet)
- `pdfpages`, to include the popular science summary page at the end

Other useful packages:

- `listings`, for producing code listings with syntax colouring and line numbers

A.1 Page Size and Margins

Use A4 paper, with the text margins given in Table A.1.

Table A.1: Text margins for A4.

margin	space
top	3.0cm
bottom	3.0cm
left (inside)	2.5cm
right (outside)	2.5cm
binding offset	1.0cm

A.2 Typeface and Font Sizes

The fonts to use for the reports are **TeX Gyre Termes** (a **Times New Roman** clone) for serif fonts, **TeX Gyre Heros** (a **Helvetica** clone) for sans-serif fonts, and finally **`TeX Gyre Cursor`** (a **`Courier`** clone) as mono-space font. All these fonts are included with the TeXLive 2013 installation. Table A.2 lists the most important text elements and the associated fonts.

A.2.1 Headers and Footers

Note that the page headers are aligned towards the outside of the page (right on the right-hand page, left on the left-hand page) and they contain the section title on the right and the chapter title on the left respectively, in SmallCaps. The footers contain only page numbers on the exterior of the page, aligned right or left depending on the page. The lines used to delimit the headers and footers from the rest of the page are $0.4pt$ thick, and are as long as the text.

Table A.2: Font types, faces and sizes to be used.

Element	Face	Size	LaTeXsize
Ch. label	serif, bold	24.88pt	`\huge`
Chapter	serif, bold	24.88pt	`\Huge`
Section	sans-serif, bold	20.74pt	`\LARGE`
Subsection	sans-serif, bold	17.28pt	`\Large`
Subsubsection	sans-serif, bold	14.4pt	`\large`
Body	serif	12pt	`\normalsize`
Header	serif, SmallCaps	10pt	
Footer (page numbers)	serif, regular	12pt	
Figure label	**serif, bold**	12pt	
Figure caption	serif, regular	12pt	
In figure	sans-serif	*any*	
Table label	**serif, bold**	12pt	
Table caption and text	serif, regular	12pt	
`Listings`	`mono-space`	$\leq$ 12pt	

A.2.2 Chapters, Sections, Paragraphs

Chapter, section, subsection, etc. names are all left aligned, and numbered as in this document.

Chapters always start on the right-hand page, with the label and title separated from the rest of the text by a $0.4pt$ thick line.

Paragraphs are justified (left and right), using single line spacing. Note that the first paragraph of a chapter, section, etc. is not indented, while the following are indented.

A.2.3 Tables

Table captions should be located above the table, justified, and spaced 2.0cm from left and right (important for very long captions). Tables should be numbered, but the numbering is up to you, and could be, for instance:

- **Table X.Y** where X is the chapter number and Y is the table number within that chapter. (This is the default in LaTeX. More on LaTeX can be found on-line, including whole books, such as [**?**].) or

- **Table Y** where Y is the table number within the whole report

As a recommendation, use regular paragraph text in the tables, bold headings and avoid vertical lines (see Table A.2).

A.2.4 Figures

Figure labels, numbering, and captions should be formed similarly to tables. As a recommendation, use vector graphics in figures (Figure A.1), rather than bitmaps (Figure A.2). Text within figures usually looks better with sans-serif fonts.

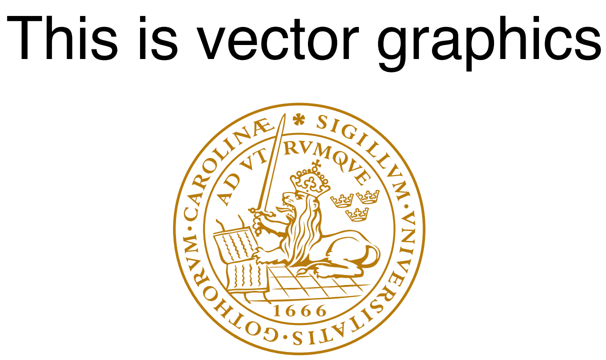


Figure A.1: A PDF vector graphics figure. Notice the numbering and placement of the caption. The caption text is indented 2.0cm from both left and right text margin.

Figure A.2: A JPEG bitmap figure. Notice the bad quality of such an image when scaling it. Sometimes bitmap images are unavoidable, such as for screen dumps.

For those interested in delving deeper into the design of graphical information display, please refer to books such as [**?**, **?**].

A.3 Mathematical Formulae and Equations

You are free to use in-text equations and formulae, usually in *italic serif* font. For instance: $S = \sum_i a_i$. We recommend using numbered equations when you do need to refer to the specific equations:

$$E = \int_0^\delta P(t)dt \quad \longleftrightarrow \quad E = mc^2 \tag{A.1}$$

The numbering system for equations should be similar to that used for tables and figures.

A.4 References

Your references should be gathered in a **References** section, located at the end of the document (before **Appendices**). We recommend using number style references, ordered as appearing in the document or alphabetically. Have a look at the references in this template in order to figure out the style, fonts and fields. Web references are acceptable (with restraint) as long as you specify the date you accessed the given link [?, ?]. You may of course use URLs directly in the document, using mono-space font, i.e. `http://cs.lth.se/`.

Make sure you add references as close to the claim as possible [?], as shown, not at the end of a whole paragraph. Notice also that there is a space before the reference; best is to use `~\cite{ref}`, to allow for unbreakable spaces. References should not be used after the period marking the end of sentence. Using the reference as follows (end of paragrah, after period) is *strongly discouraged*, since it says nothing about which specific claim you provide the reference for. [?]

A.5 Colours

As a general rule, all theses are printed in black-and-white, with the exception of selected parts in selected theses that need to display colour images essential to describing the thesis outcome (*computer graphics*, for instance).

A strong requirement is for using **black text on white background** in your document's main text. Otherwise we do encourage using colours in your figures, or other elements (i.e. the colour marking internal and external references) that would make the document more readable on screen. You may also emphasize table rows, columns, cells, or headers using white text on black background, or black text on light grey background.

Note that the document should look good in black-and-white print. Colours are often rendered using monochrome textures in print, which makes them look different from on screen versions. This means that you should choose your colours wisely, and even opt for black-and-white textures when the distinction between colours is hard to make in print. The best way to check how your document looks, is to print out a copy yourself.

The LaTeX class defines also a few *LTH* standard colours, which you could use in your document for various elements, to adhere to the standard university profile.
These are: LTHblue, LTHbronze, LTHgreen, LTHpink, LTHcyan, LTHgrey.

Appendix B
Language

You are strongly encouraged to write your report in English, for two reasons. First, it will improve your use of English language. Second, it will increase visibility for you, the author, as well as for the Department of Computer Science, and for your host company (if any).

However, note that your examiner (and supervisors) are not there to provide you with extensive language feedback. We recommend that you check the language used in your report in several ways:

Reference books dedicated to language issues can be very useful. [**?**]

Spelling and grammar checkers which are usually available in the commonly used text editing environments.

Colleagues and friends willing to provide feedback your writing.

Studieverkstaden is a university level workshop, that can help you with language related problems (see Studieverkstaden's web page).

Websites useful for detecting language errors or strange expressions, such as

- `http://translate.google.com`
- `http://www.gingersoftware.com/grammarcheck/`

B.1 Style Elements

Next, we will just give some rough guidelines for good style in a report written in English. Your supervisor and examiner as well as the aforementioned **Studieverkstad** might have a different take on these, so we recommend you follow their advice whenever in doubt. If you want a reference to a short style guide, have a look at [**?**].

Widows and Orphans

Avoid *widows* and *orphans*, namely words or short lines at the beginning or end of a paragraph, which are left dangling at the top or bottom of a column, separated from the rest of the paragraph.

Footnotes

We strongly recommend you avoid footnotes. To quote from [**?**], *Footnotes are frequently misused by containing information which should either be placed in the text or excluded altogether. They should be avoided as a general rule and are acceptable only in exceptional cases when incorporation of their content in the text [is] not possible.*

Active vs. Passive Voice

Generally active voice (*I ate this apple.*) is easier to understand than passive voice (*This apple has been eaten (by me).*) In passive voice sentences the actor carrying out the action is often forgotten, which makes the reader wonder who actually performed the action. In a report is important to be clear about who carried out the work. Therefore we recommend to use active voice, and preferably the plural form *we* instead of *I* (even in single author reports).

Long and Short Sentences

A nice brief list of sentence problems and solutions is given in [**?**]. Using choppy sentences (too short) is a common problem of many students. The opposite, using too long sentences, occurs less often, in our experience.

Subject-Predicate Agreement

A common problem of native Swedish speakers is getting the subject-predicate (verb) agreement right in sentences. Note that a verb must agree in person and number with its subject. As a rough tip, if you have subject ending in *s* (plural), the predicate should not, and the other way around. Hence, *only one s*. Examples follow:

incorrect He have to take this road.

correct He has to take this road.

incorrect These words forms a sentence.

correct These words form a sentence.

In more complex sentences, getting the agreement right is trickier. A brief guide is given in the *20 Rules of Subject Verb Agreement* [**?**].

Appendix C
Structure

It is a good idea to discuss the structure of the report with your supervisor rather early in your writing. Given next is a generic structure that is a starting point, but by no means the absolute standard. Your supervisor should provide a better structure for the specific field you are writing your thesis in. Note also that the naming of the chapters is not compulsory, but may be a helpful guideline.

Introduction should give the background of your work. Important parts to cover:

- Give the context of your work, have a short introduction to the area.
- Define the problem you are solving (or trying to solve).
- Specify your contributions. What does this particular work/report bring to the research are or to the body of knowledge? How is the work divided between the co-authors? (This part is essential to pinpoint individual work. For theses with two authors, it is compulsory to identify which author has contributed with which part, both with respect to the work and the report.)
- Describe related work (literature study). Besides listing other work in the area, mention how is it related or relevant to your work. The tradition in some research area is to place this part at the end of the report (check with your supervisor).

Approach should contain a description of your solution(s), with all the theoretical background needed. On occasion this is replaced by a subset or all of the following:

- **Method**: describe how you go about solving the problem you defined. Also how do you show/prove that your solution actually works, and how well does it work.
- **Theory**: should contain the theoretical background needed to understand your work, if necessary.
- **Implementation**: if your work involved building an artefact/implementation, give the details here. Note, that this should not, as a rule, be a chronological

description of your efforts, but a view of the result. There is a place for insights and lamentation later on in the report, in the Discussion section.

Evaluation is the part where you present the finds. Depending on the area this part contains a subset or all of the following:

- **Experimental Setup** should describe the details of the method used to evaluate your solution(s)/approach. Sometimes this is already addressed in the **Method**, sometimes this part replaces **Method**.
- **Results** contains the data (as tables, graphs) obtained via experiments (benchmarking, polls, interviews). Here you should also describe the individual tables or graphs in text, pointing out interesting outliers and trends.
- **Discussion** allows for a longer discussion and interpretation of the results from the evaluation, including extrapolations and/or expected impact. Focus here on a broader view of the results, talking about the relation between the different finds.[1] This might also be a good place to describe your positive and negative experiences related to the work you carried out.

Occasionally these sections are intermingled, if this allows for a better presentation of your work. However, try to distinguish between measurements or hard data (results) and extrapolations, interpretations, or speculations (discussion).

Conclusions should summarize your findings and possible improvements or recommendations.

Bibliography is a must in a scientific report. LaTeX and `bibtex` offer great support for handling references and automatically generating bibliographies.

Appendices should contain lengthy details of the experimental setup, mathematical proofs, code download information, and shorter code snippets. Avoid longer code listings. Source code should rather be made available for download on a website or on-line repository of your choosing.

[1]Bad practice is to display graphs in Results and then describe them textually one by one in here. No! Both sections should have some discussion, but one targets individual finds and the other tries to bridge between these adopting a more overarching viewpoint.

EXAMENSARBETE Application Specific Instruction-set Processor Using a Parametrizable multi-SIMD Synthesizeable Model Supporting Design Space Exploration
STUDENT Magnus Hultin
HANDLEDARE Flavius Gruian (LTH)
EXAMINATOR Krzysztof Kuchcinski (LTH)

Parametrisk processor modell för design utforskning

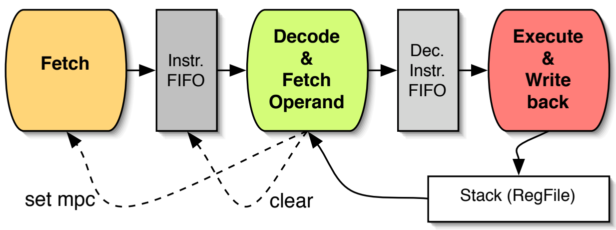
POPULÄRVETENSKAPLIG SAMMANFATTNING **Magnus Hultin**

Applikations-specifika processorer är allt mer vanligt för få ut rätt prestanda med så lite resurser som möjligt. Detta arbete har en parametrisk modell för att kunna testa hur mycket resurser som behövs för en specifik applikation.

För att öka prestandan i dagens processorer finns det vektorenheter och flera kärnor i processorer. Vektorenheten finns till för att kunna utföra en operation på en mängd data samtidigt och flera kärnor gör att man kan utföra fler instruktioner samtidigt. Ofta är processorerna designade för att kunna stödja en mängd olika datorprogram. Detta resulterar i att det blir kompromisser som kan påverka prestandan för vissa program och vara överflödigt för andra. I t.ex. videokameror, mobiltelefoner, medicinsk utrustning, digital kameror och annan inbyggd elektronik, kan man istället använda en processor som saknar vissa funktioner men som istället är mer energieffektiv. Man kan jämföra det med att frakta ett paket med en stor lastbil istället för att använda en mindre bil där samma paketet också skulle få plats.

I mitt examensarbete har jag skrivit en modell som kan användas för att snabbt designa en processor enligt vissa parametrar. Dessa parametrar väljs utifrån vilket eller vilka program man tänkta köra på den. Vissa program kan t.ex. lättare använda flera kärnor och vissa program kan använda korta eller längre vektorenheter för dess data.

Modellen testades med olika multimedia program. Den mest beräkningsintensiva och mest upprepande delen av programmen användes. Dessa kallas för kärnor av programmen. Kärnorna som användes var ifrån MPEG och JPEG, som används för bildkomprimering och videokomprimering.



Resultatet visar att det finns en prestanda vinst jämfört med generella processorer men att detta också ökar resurserna som behövs. Detta trots att den generella processorn har nästan dubbelt så hög klockfrekvens än dem applikationsspecifika processorerna. Resultatet visar också att schemaläggning av instruktionerna i programmen spelar en stor roll för att kunna utnyttja resurserna som finns tillgängliga och därmed öka prestandan. Med den schemaläggningen som utnyttjade resurserna bäst var prestandan minst 79% bättre än den generella processorn.