User Manual

CrazyCircus-Group

December 13, 2023

Version 1.0

Status

Reviewed		
Approved		

CrazyCircus-Group
ellge955@student.liu.se

Project Identity

Group E-mail: ellge955@student.liu.se

Homepage: http://www.liu.se/

Orderer: Anton Kullberg, Reglerteknik/LiU
Phone: -
E-mail: anton.kullberg@liu.se

Customer: Daniel Axehill, Reglerteknik/LiU
Phone: +46 13 28 40 42
E-mail: daniel.axehill@liu.se

Supervisor: Joel Nilsson, Reglerteknik/LiU
Phone: -
E-mail: joel.nilsson@liu.se

Course Responsible: Daniel Axehill, Reglerteknik/LiU
Phone: +46 13 28 40 42
E-mail: daniel.axehill@liu.se

Project participants

Name	**Responsibility**	**E-mail**
Elliot Gestrin	Project manager (PM)	ellge955@student.liu.se
Martin Agebjär	Control technology (CT)	marag492@student.liu.se
Hugo Asplund	Hardware (HW)	hugas433@student.liu.se
Marcus Filipsson	Simulation (SIM)	marfi245@student.liu.se
Alvin Gustavsson Vester	Design (DES)	alvgu648@student.liu.se
Albin Helsing	Testing (TEST)	albhe896@student.liu.se
Tomas Röjder	Software (SW)	tomro614@student.liu.se
Adam Simon	GUI/Information (GUI/I)	adasi503@student.liu.se
Axel Stockhaus	Documentation (DOC)	axest416@student.liu.se

CONTENTS

CrazyCircus-Group
ellge955@student.liu.se

DOCUMENT HISTORY

Version	Date	Changes made	Made by	Reviewed
0.1	5/12-2023	First version	CrazyCircus-Group	Alvin Gustafsson Wester
0.2	6/12-2023	Clarified Notebook planning and Python dependencies.	CrazyCircus-Group	Alvin Gustafsson Wester
1.0	8/12-2023	Updated with clarifications from orderer.	CrazyCircus-Group	Alvin Gustafsson Wester

CrazyCircus-Group
ellge955@student.liu.se

1 INTRODUCTION

This is a user manual for a system developed by the CrazyCircus-Group in the course TSRT10, focusing on the creation of software for executing acrobatic maneuvers with drones [1][2]. The purpose of this document is to provide users with essential information about the system components, ensuring a safe and efficient utilization of the entire setup. Included in this manual is a detailed description of the installation and operation of each subsystem for seamless acrobatic drone performance.

2 REQUIRED COMPONENTS

In this section, the required components needed in order to run the project is listed.

- **Computer:** Running Ubuntu 22.04 and Docker 24.0.7.
- **Crazyflie:** A Crazyflie is a nano-quadcopter devoloped by Bitcraze. The crazyflie used in this project is the Crazyflie 2.1, which has an active marker deck.
- **Crazyradio:** Radio developed by Bitcraze that makes communication between the crazyflie and the computer possible.
- **Qualisys:** The motion capture system used in Visionen, tracking the active marker deck on the crazyflie.
- **Drone manual controller:** The controller used to control the drone manually.

3 INSTALLATION

3.1 Ubuntu

Ubuntu 22.04 or above is required to run ROS2. Install Ubuntu for desktop by following official documentation. It is strongly suggested to opt for installing Ubuntu native on your computer, for instance using dual boot.

3.2 Python

Python version 3.10 is needed. Python 3.10 comes as default with Ubuntu 22.04. It can be confirmed that Python 3.10 is installed by writing in a terminal:

```
$ python3 --version
```

The Python version should then be printed in the terminal. If the command is not recognized or an older version of Python is shown, Python 3.10 or a later version can be installed using official documentation.

3.3 Docker

Docker is a tool for creating containers for projects. Containers are isolated operating systems with all dependencies already installed that can be run on your host OS. This is used to make the installation easier and faster. How to install docker on Ubuntu is explained here:
https://docs.docker.com/engine/install/ubuntu/

3.4 Clone CrazyCircus project

Clone CrazyCircus repository to root with:

```
$ cd
$ git clone git@gitlab.liu.se:tsrt10/2023/visionen.git CrazyCircus
```

Note that the repo must be cloned to root.

Add USB permissions for the Crazyradio with:

```
$ cd ~/CrazyCircus/crazyswarm2
$ sudo ./pc_permissions.sh
```

If this does not work, try manually adding USB permissions explained here:
https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/installation/usb_permissions/

Build the container:

```
$ cd ~/CrazyCircus
$ ./build_docker_image.sh
```

Start the container with

```
$ ./run.sh
or
$ ./run_mount.sh
```

Inside the container, run:

```
$ cd
$ ./colcon_build.sh
$ source ros2_ws/install/setup.bash
```

To start a new terminal in the same container, run this outside the container in a new terminal:

```
$ cd ~/CrazyCircus
$ ./terminal.sh
```

3.5 Crazyflie Client

To do changes to the crazyflie, for example changing its radio address or flashing new firmware, you can use the crazyflie client. Installation instructions can be found here https://www.bitcraze.io/documentation/repository/crazyflie-clients-python/master/installation/install/. This is done separate to the CrazyCircus repository.

3.6 Notebook Setup

To utilise Notebook planning, see Section 6, without the Docker environment one needs to install the necessary python dependencies which are found in "requirements.txt" in "Crazyflie_acrobatics". Note that python 3.10 is recommended and tested, though other versions likely will also work. Using Anaconda and a Linux system this can be done with the following commands:

```
$ conda create -n visionen python=3.10
$ pip install -r Crazyflie_acrobatics/requirements.txt
```

This environment can then be selected for use in the Jupyter Notebook. For further details of how this is done and the Notebook used in Visual Studio Code, see the following: https://code.visualstudio.com/docs/datascience/jupyter-notebooks.

4 SETUP PROCEDURES

Before flying with the drone, some measures need to be taken in order to setup the system correctly. The setup procedure for the different components will be stated in this section.

4.1 Qualisys system

To obtain accurate positioning and orientation data from the Qualisys system, it is essential to follow a specific start-up procedure. The outlined steps below detail the necessary actions:

1. Turn of the lights inside Visionen and pull down the automatically controlled curtains to minimize external light.

2. Start the computer and log in to a LiU-account.

3. Click on the **Qualisys Track Manager** (QTM) icon to start the software.

4. If no project exists, click on **New project** and choose **Base the new project on: Settings imported from another project**. Choose the settings file inside the folder under **disk A:** called **LIU New setup**. If a project already exists, from **This Pc→disk A:** choose the project folder.

5. Select **New** (White paper symbol in the top left corner).

6. Navigate to 2D view and click on **Auto-Mask** in the right panel. Click on **allow to remove all previous masks**. If needed go through each camera and manually apply mask were needed.

7. The Qualisys system should now be ready. Navigate to 3D view to see the 3D space.

4.1.1 *Calibration*

If a new project is created or if there is any inaccuracy in the positioning data, calibration needs to be performed. Follow the specified procedure for calibration below.

1. Make sure that passive markers is chosen in QTM and that no markers are in view of the cameras.

2. Place the L-frame with its corner where you want the origin to be and the longer arm of the frame pointing in the desired x-axis. The L-frame is preferably placed with the corner parallel to the corner of the metal plate in the middle of visonen.

3. In the top bar of QTM click on the calibration icon. See Figure 1.

4. In the calibration window set the calibration time to 240s, and then click on **OK** to start the calibration.

5. When the calibration starts, walk with the calibration wand inside visionen and wave and twist it slowly during the entirety of the calibration. Try to cover as much of the volume as possible.

Figure 1: Calibration icon.

6. The calibration is now done and the next step is to set up the crazyflie. If the calibration returns as not sucsesfull, redo step 1-5.

4.2 Set up the crazyflie

The first time a drone is used in visionen it needs to be connected and configured with the Qualisys system.

4.2.1 *Connect drone to radio and assign ID*

The first time a drone is used, it needs to be assigned a unique ID in the Crazyflie client. The setup procedure is as follows:

1. With the Crazyradio inserted into the laptops USB-port, turn on the drone by pressing the power-button. When the drone is powered on, connect it to the laptop with a USB-cable.

3. In the terminal, navigate to the path */crazyflie-clients-python/bin* and type *cfclient* to start the client.

4. In the Crazyflie client click the **Scan** button in the top left corner. In the scroll bar, choose the USB-alternative and click on **Connect**. If a connection is successful, the orientation of the drone should be displayed in the mainframe.

5. Click on configure under the Connect tab to set the radio channel and Crazyflie address. The channel should be set to ch. 80 and the Crazyflie radio address should be assigned to 0xE7E7E7E7<X>, where <X> is replaced with the desired drone ID.

4.2.2 *Define drone in Qualisys*

In order to use the drones togheter with the Qualisys system they need to be defined in QTM.

1. In QTM choose active markers, since the drones are equipped with an Active marker deck. Make sure the exposure time is 450µs.

2. Place the drone that you want to define inside Visionen and make sure that the front of the drone points in the same direction as the global x-axis.

3. In the 3D view in Qualisys Track Manager four balls will now be visible, representing each marker on the drone.

To define the body hold shift and drag a box around all four markers. Right click on them and choose **Define rigid body (6DOF)** → **Current Frame**. Choose the desired name (preferebly cfX, where X denotes the index of the drone).

4. The drone is now defined as a rigid body and its local cordinate system should be visible in the 3D view.

5. Now the drone is ready to fly. Place the drone on a level surface and press the power button. Wait for the propellers to spin, indicating that the calibration is done.

4.2.3 *Building and Flashing the firmware*

First the firmware needs to be cloned.

```
$ git clone --recursive https://github.com/bitcraze/crazyflie-firmware.git
```

Do the appropriate changes to the firmware. Changed files for this project are found in /CrazyCircus/firmware-changes. Build the firmware with:

```
$ make cf2_defconfig
$ make -j 12
```

Start the drone you want to flash in bootloader mode by holding the power button until it starts flashing blue. To flash the firmware run

```
$ make cload
```

5 OPERATING THE SYSTEM USING THE GUI

When the system has been configured properly and the drones are ready to fly, everything can be operated from the Graphical User Interface which connects the whole system. An overview of the GUI can be seen in Figure 2.

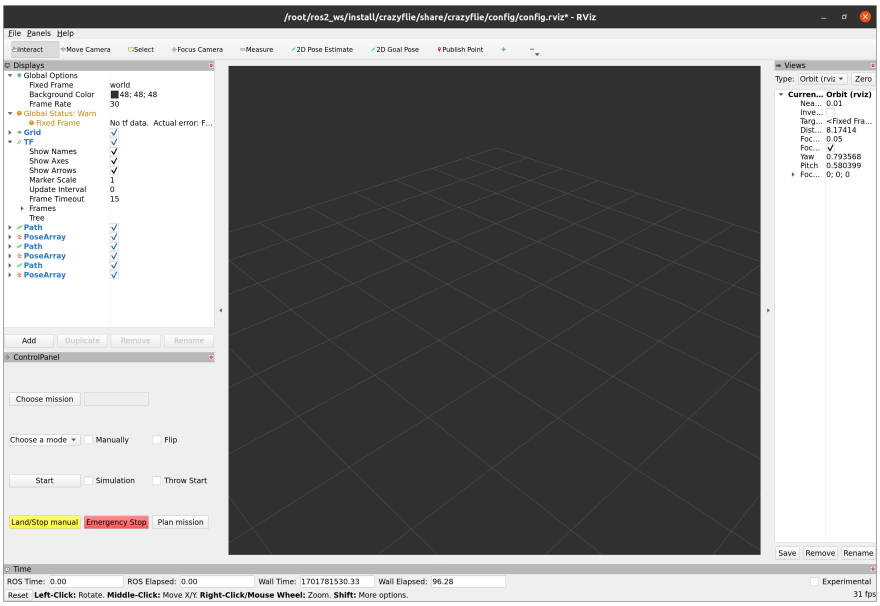

Figure 2: Graphical User Interface.

Figure 3 displays a closer view of the buttons accessible from the GUI.

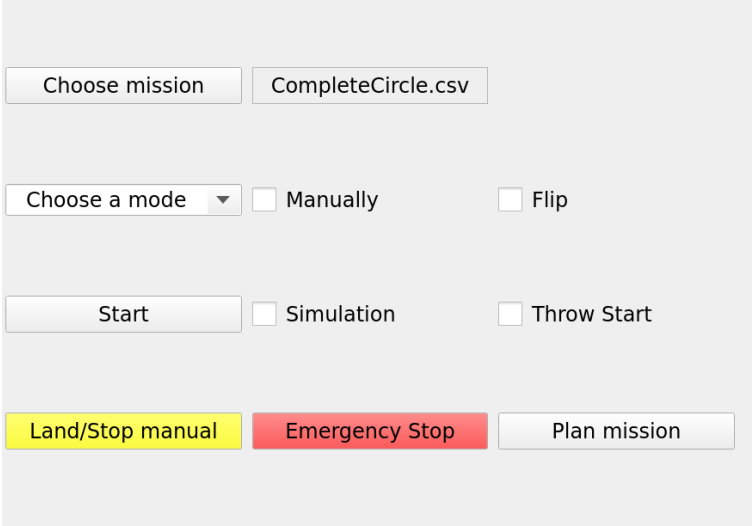

Figure 3: Graphical User Interface, buttons.

Before using the system, ensure that there are at least two people present for easy operation, as seen in Figure 4.

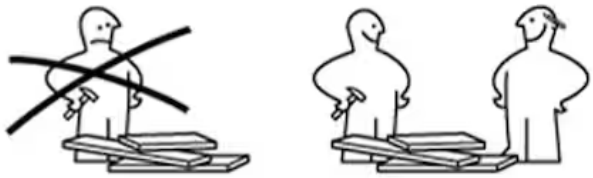

Figure 4: Working Together.

5.1 Initiate the system

Before launching the system, the crazyflie has to be turned on and placed inside Visionen. It's also important that the computer is connected to the network *Visionen 5 GHz*, otherwise the system won't receive the Crazyflies pose from Qualisys.

When inside the docker container the system including simulation is launched with:

```
$ ros2 launch crazyflie launch.py
```

The system excluding simulation is launched with:

```
$ ros2 launch crazyflie launch.py sim:=false
```

To control that the system works fine, open another terminal in the same docker container. How this is done is described in Section 3.4. In the new terminal use the command:

```
$ ros2 run crazyflie_examples hello_world
```

The Crazyflie should then takeoff, hover for 5 seconds and then land.

5.2 Choose mission

The *Choose mission* button can be used to select a mission that the crazyflie can fly. Notice that the selected file must be in csv-format.

5.3 Mode

The *Mode* button chooses which mode the crazyflie should fly the chosen mission in. There are three modes to choose between: GoTo, cmdFullState and cmdRate. Notice that an acrobatic mission can only be flown with cmdRate. However for a non-acrobatic mission, all modes can be used.

5.4 Start

The *Start* button can be used to start the crazyflie flying in different modes.

5.4.1 *Using a mission*

To start the crazyflie flying a mission simply press the *Start* button after choosing a mission and a mode. To start the crazyflie flying a mission in simulation, also press the *Simulation* checkbox before pressing the *Start* button.

5.4.2 *Not using a mission*

There are three different modes to run without using a mission: Flip, Throwing start and Manually. To start either of them, press the corresponding checkbox and then press the *Start* button. Notice that only one checkbox can be checked at the same time for the system to work properly. For the manual mode to work, the manual controller has to be connected to the computer before running the docker container.

5.5 Land/Stop manual

The *land/stop manual* button can be used to land the crazyflie if not in velocity mode. The button can also be used to stop the manual control of the crazyflie.

5.6 Emergency Stop

The *Emergency stop* button can be used to turn off the motors on the crazyflie completely. After this button has been used, the crazyflie gets locked and both the crazyflie and the system has to be rebooted.

5.7 Plan mission

The *Plan mission* button will initiate planning of a new mission. In the first window you will choose what type of mission you want to plan, the options are "Circle", "Loop" and "Pirouette". In the following windows input parameters for the mission can be changed. What type of parameters that is required depends on the mission type and can be settings such as height and radius. In the end, you will get a summery of your mission before choosing to plan or abort. The planner will take a while to run before saving the planned mission in the missions folder. If debug has been activated the planner will print each iteration in the console.

6 PLANNING A TRAJECTORY USING A NOTEBOOK

In addition to planning a trajectory through the GUI described in Section 5.7, a trajectory can also be planned using a Jupyter Notebook script. The ability to modify the existing Notebook sections for acrobatic trajectories (e.g., generating a loop, flip, or screw) provides the user with more flexibility compared to generating trajectories through the GUI. In the Notebook, the user also gets 2D and 3D plots of the generated trajectory along with plots of all states and control commands throughout the sequence. An example of the 2D and state plots is shown in Figure 6. Planning the trajectory will generate a CSV file containing all the states of the trajectory. Subsequently, this CSV file can be executed through the GUI as described in Section 5.2.

This notebook is found in the ‘Crazyflie_acrobatics‘ folder and is called ‘Notebook.ipynb‘.

6.1 Placement of Target Points

A trajectory is constructed by arranging a set of *target states*, and an optimization function endeavors to navigate the drone through all provided target states in an optimized manner. Within the Notebook, the user specifies the points which the planned path should aim to reach as well as how often the points should be distributed in time. For each point any combination of the possible twelve states can be set, with the remainder defaulting to zero. The twelve states are x, y & z position, roll, pitch & yaw angles as well as the derivatives thereof.

It’s also possible to decide on the cost function used. Two such functions are available, "legacy" and "interpol". "legacy" is faster, by roughly a factor of five, but considers only a single target point at a time. As such, the planning won’t plan ahead, making some missions have worse solutions and others be unsolvable. The "interpol" version interpolates the cost between all existing points, and as such is able to plan ahead. For a comparison between these, see Figure 5.

The points should be placed in such a way that the drone can reasonably be expected to reach them at a similar frequency. For example, if the drone is intended to fly in a straight line at even pace from $x = 0$ to $x = 5$, then the points $[(x = 0), (x = 2.5), (x = 5)]$ are a better choice than the points $[(x = 0), (x = 0.5), (x = 5)]$.

There are also a number of available examples, such as a loop and a flip, for which the user can modify parameters easily. This is similar to the options provided through the GUI, as described in Section 5.7. These exist mainly as reference for the user.

Even following best practice, it’s possible for the optimisation to fail if given difficult tasks. This is usually seen either through states going to infinite values or the solver reporting NaN values. To troubleshoot this, see Section 9.4.

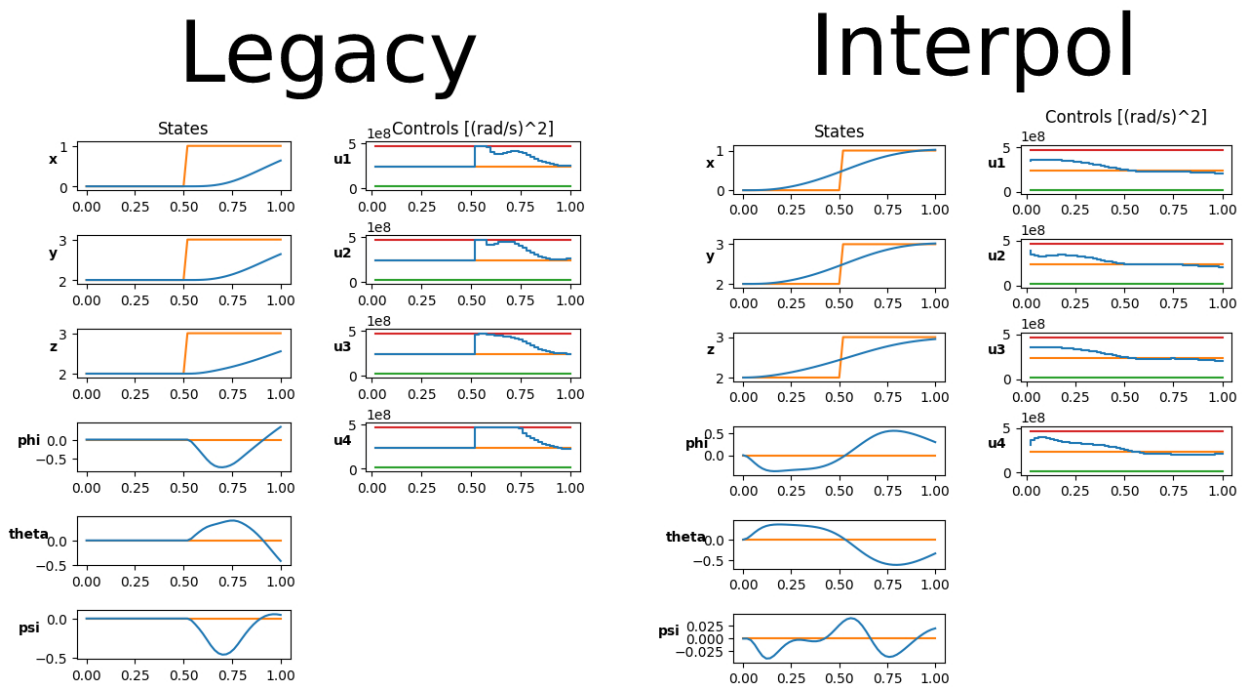

Figure 5: A comparison between the result of using "legacy" and "interpol" mode to travel between two points.

6.2 Some Notebook Technicalities

1. **Reload:** Reloading the Notebook is necessary when the user wants to execute it after making changes to any script executed in the Notebook. This ensures that the Notebook reflects the latest modifications made to the scripts. Without reloading, the Notebook does not incorporate these changes during execution.
2. **Persistent Variables Across Sessions:** While executing sections independently in the Notebook, variables defined within a section are limited to that specific section's scope. They are removed when reloading and are not automatically accessible in other sections. However, to persistently retain variables across different sections or sessions of the Notebook, the "magic command" %store can be used.

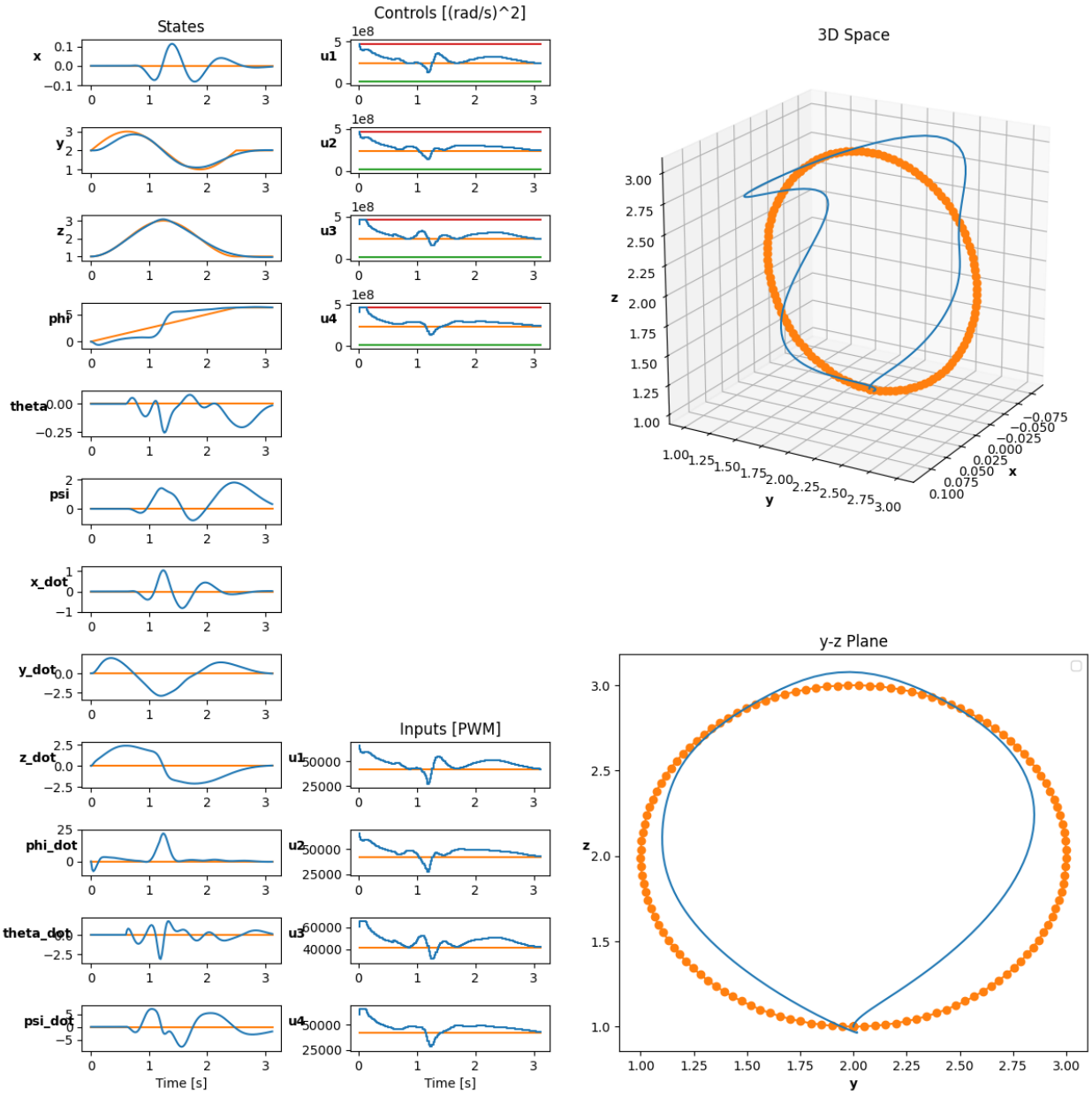

Figure 6: An example plot from the Notebook planning.

7 STOP SYSTEM

1. **Turn off system:** This is achieved by initiating a keyboard interrupt (using '**Ctrl+C**') in the terminal that executed the ros2 launch command. This action will promptly shut down the GUI and the rest of the system, resulting in the loss of connection to the drone. Therefore, this should only be performed when the drone is stationary.
2. **Turn off the drone(s):** Use the same button as for startup.
3. **Qualisys system:** The Qualisys cameras should be left on. Simply shut down the Qualisys window on the computer and sign out. Raise the curtains.

8 LIMITATIONS OF THE SYSTEM

In this section the limitations of the system will be explained.

8.1 Ubuntu

The system is using ROS2. ROS2 is only runnable on Ubuntu 22.04. Therefore the host computer has to be running Ubuntu 22.04 as OS for the system to work. It's not possible to run the system using an older version of Ubuntu as well as an older version of ROS.

8.2 One Crazyflie

The system is built to maneuver one crazyflie to do acrobatics. Meaning that at the moment it's not possible to fly multiple crazyflies simultaneously. However, it's still possible to change which crazyflie to use by changing in the crazyflie config file.

8.3 Firmware

The system is runnable using a certain firmware flashed on the crazyflie. The changes made from the original firmware is that tumble is turned off and that the propellers still rotates while having a low thrust. How to flash the firmware to the crazyflie is described in Section 4.2.3.

8.4 Qualisys system

The system is based on the Qualisys system, which means that without Qualisys the drone will only receive its pose data from the drones IMU. While the orientation data from the IMU is reliable, the position data is not and therefore it will impede the drones ability to fly.

8.5 Planning a mission

When planning a mission the motion planner doesn't always optimize the trajectory if the trajectory wanted is too hard. The points the planner uses to optimize a trajectory also needs to be placed with even timegap.

8.6 Land in velocity mode

When flying the drone in velocity mode, the land button in the GUI doesn't work. Therefore if the crazyflie needs to be stopped while flying in velocity mode, the emergency stop button has to be used.

9 TROUBLESHOOTING

In this section known issues and solutions to these problems will be explained.

9.1 Docker image fails to build.

If docker has crashed or if the computer unexpectedly shutdown it will not be possible to build a new docker image. To solve this the best approach is to completely reinstall docker.

1. Identify what packages are installed. If two versions of the same package is installed docker will not work properly.

   ```
   $ dpkg -l | grep -i docker
   ```

2. Remove images, containers, volumes, or user created configuration files on your host.

   ```
   $ sudo apt-get purge -y docker-engine docker docker.io docker-ce
   docker-ce-cli docker-compose-plugin
   $ sudo apt-get autoremove -y --purge docker-engine docker docker.io
   docker-ce docker-compose-plugin
   ```

3. Delete all images, containers, and volumes with the following commands:

   ```
   $ sudo rm -rf /var/lib/docker /etc/docker
   $ sudo rm /etc/apparmor.d/docker
   $ sudo groupdel docker
   $ sudo rm -rf /var/run/docker.sock
   ```

4. Install docker again.

   ```
   $ sudo apt-get install docker-ce docker-ce-cli containerd.io
   docker-buildx-plugin docker-compose-plugin
   ```

9.2 Drone drifting in GUI while being stationary.

While having the drone placed stationary in visionen the GUI sometimes thinks the drone is moving. This will result in a "out of bounds" message and the drone position will be estimated to [0,0,0] again.

1. Make sure you are connected to Visionen 5 Gz wifi.
2. Restart the GUI and drone. Make sure to start the drone first.

9.3 Drone bouncing around.

Sometimes when starting a mission the drone will bounce around and not fly properly.

1. Restart the GUI and drone. Make sure to start the drone first.

9.4 Planner fails to converge

The planner might fail to converge on certain difficult missions. There are a few things which can be attempted to remedy this:

1. Use the "interpol" mode instead of the "legacy" mode.
2. If the simulated frequency is less than 50 Hz, increase it. This is done with the "step_horizon" parameter in "constants.py".
3. Try adding more points to further guide the solver.
4. Modify some of the optimisation weights used by the solver. This is done with the various "Q_*" parameters in "constants.py". Putting less weight on unimportant parameters makes it easier to solve.

REFERENCES

[1] E. Gestrin, M. Agebjär, H. Asplund, M. Filipsson, A. Gustafsson-Wester, A. Helsing, T. Röjder, A. Simon, and A. Stockhaus, "Project plan," 2023.

[2] ——, "Technical documentation," 2023.