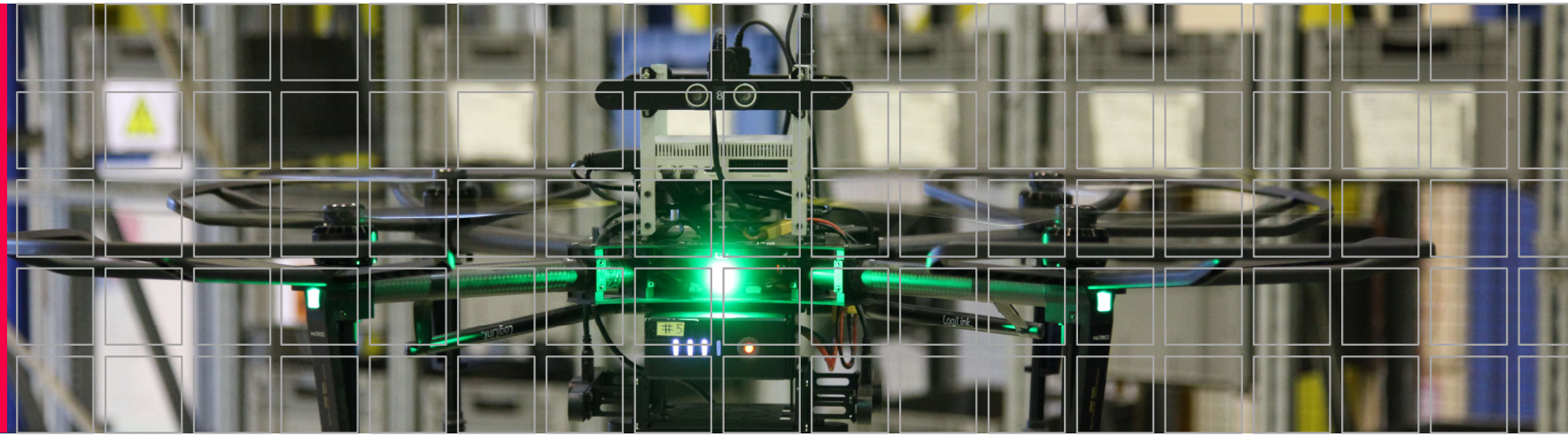


ROS Practical (I)
Camera Drones Lecture

Jesús Pestana, Friedrich Fraundorfer

Graz University of Technology
Institute for Computer Graphics and Vision
25 October 2023

Contents

- Introduction to ROS practicals
- Robot Operating System (ROS)
- Byobu - Detaching from the current shell session
- ROS Bags / rqt / RViz
- Testing a Visual Inertial Odometry (VIO) Algorithm

ROS Melodic – Ubuntu 18.04
(there are always turtles on these images)
http://wiki.ros.org/melodic

Contents

- **Introduction to ROS practicals**
  - Practical 1
  - Practical 2
  - Practical 3
  - Practical 4
- Robot Operating System (ROS)
- Byobu - Detaching from the current shell session
- ROS Bags / rqt / RViz
- Testing a VIO Algorithm

ROS Melodic – Ubuntu 18.04
(there are always turtles on these images)
http://wiki.ros.org/melodic

Introduction to ROS practicals

- ROS Practicals:
  - 4 Assignments
  - 3D Mapping example
  - 3D Trajectory Planning for drones
- Use Virtual Machine with ROS Melodic
- Organized in teams of 2 students
- Ask for support:
  - Course forum >> Visible to all
  - My E-Mail: jesus.pestana@pro2future.at

ROS Melodic – Ubuntu 18.04
http://wiki.ros.org/melodic

Introduction to ROS practicals

- ROS Practicals:
  - Last assignments:
    - provide ROS logfile
    - trajectories for the drone
    - See drone flights at the lab
- Organized in teams of 2 students
- Ask for support:
  - Course forum >> Visible to all
  - My E-Mail: jesus.pestana@pro2future.at

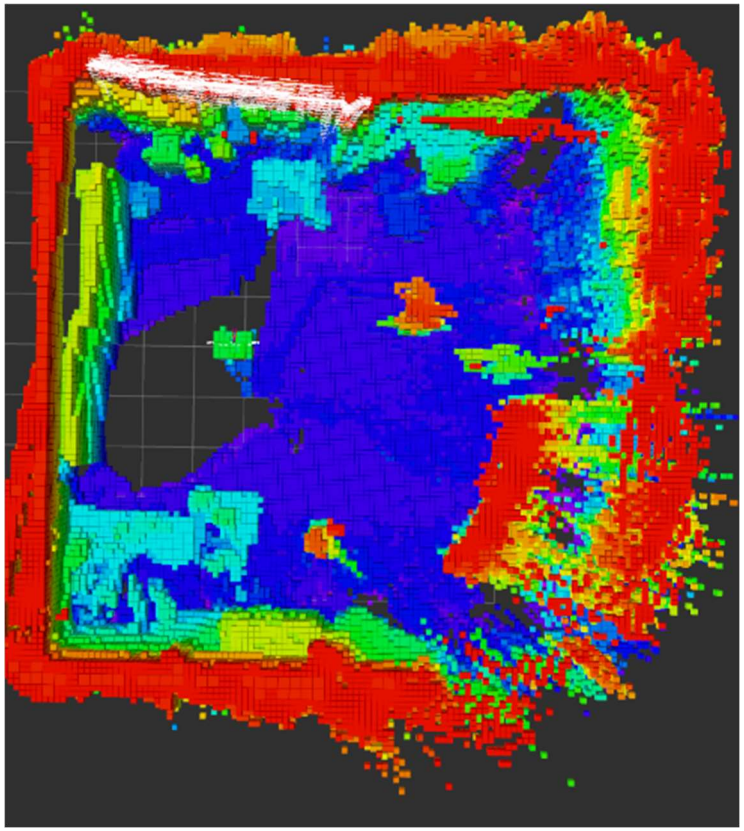
ROS Melodic – Ubuntu 18.04
http://wiki.ros.org/melodic

Contents

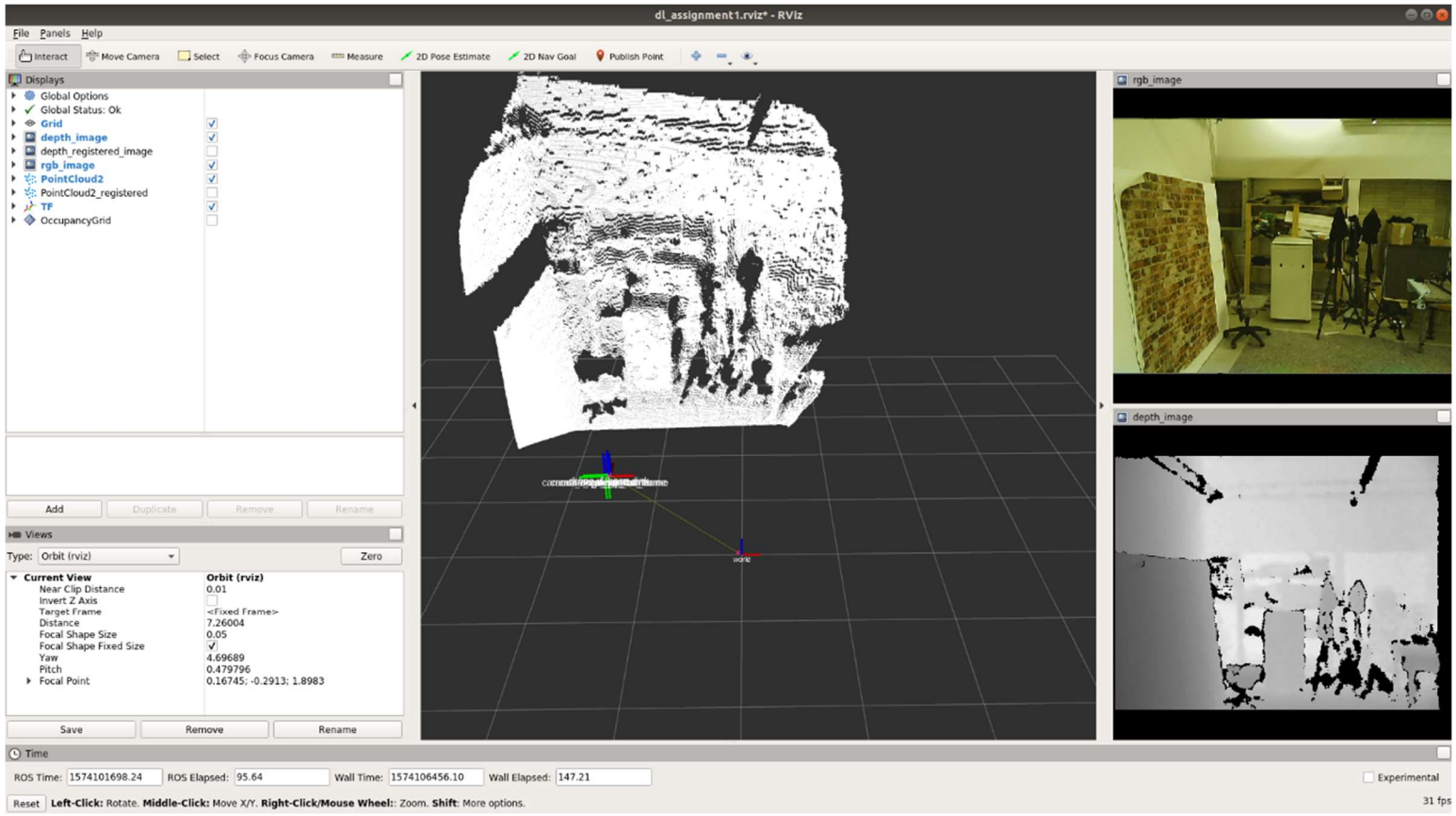
- **Introduction to ROS practicals**
  - **Practical 1**
  - Practical 2
  - Practical 3
  - Practical 4
- Robot Operating System (ROS)
- Byobu - Detaching from the current shell session
- ROS Bags / rqt / RViz
- Testing a VIO Algorithm

ROS Melodic – Ubuntu 18.04
(there are always turtles on these images)
http://wiki.ros.org/melodic

Assignment 1 – 3D obstacle map generation

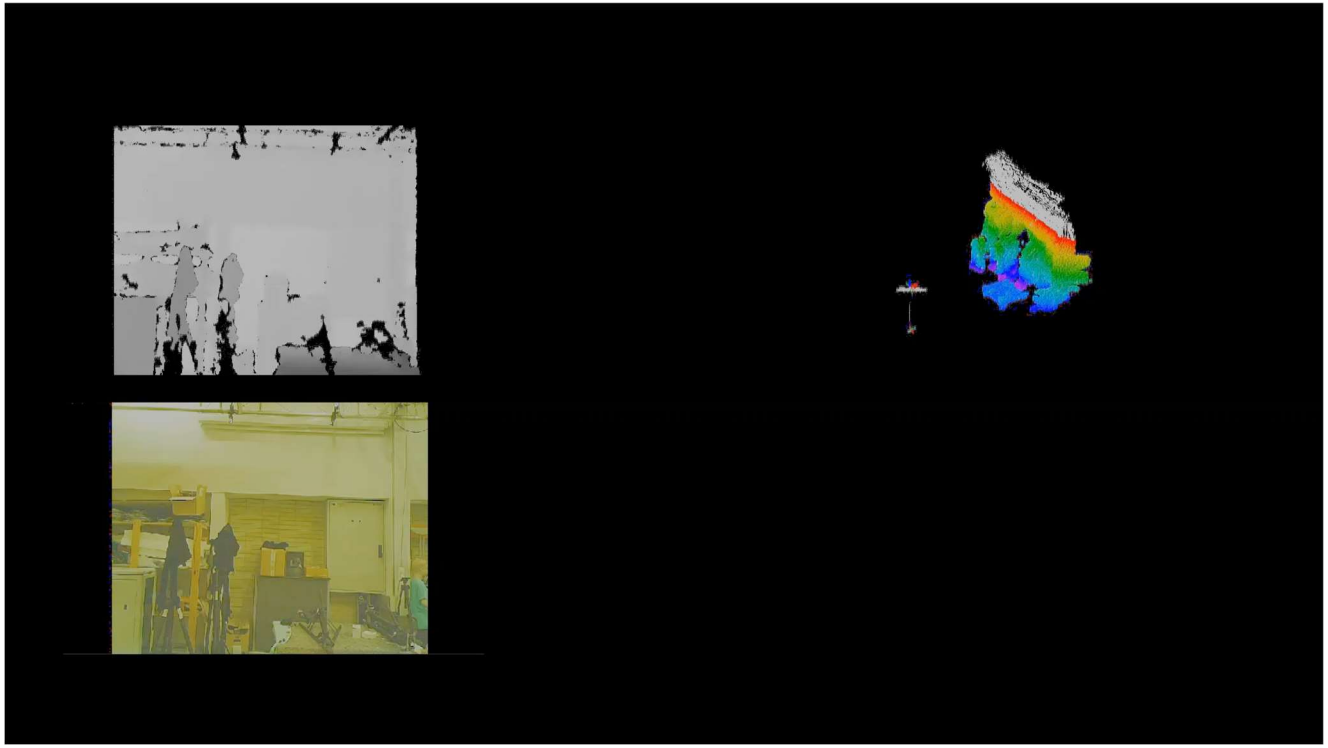


Assignment 1 – 3D obstacle map generation



Assignment 1 – 3D obstacle map generation
Mapping the environment

Create an octomap from a ROS bag containing RGBD images and camera positions



Assignment 1 – 3D obstacle map generation
ROSBag – ROS native dataset format

2019-11-18-19-37-10.bag:
https://files.icg.tugraz.at/f/fb3643f14540402895d5/

Created with Orbec Astra Pro

Topics:

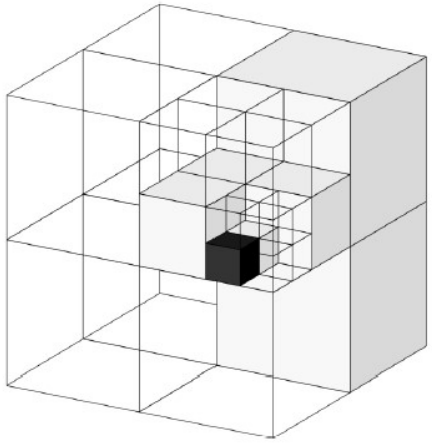
```
types:   geometry_msgs/PoseStamped
         sensor_msgs/CameraInfo
         sensor_msgs/Image
         tf2_msgs/TFMessage
topics:  /camera/color/camera_info    1096 msgs    : sensor_msgs/CameraInfo
         /camera/color/image_raw      1096 msgs    : sensor_msgs/Image
         /camera/depth/camera_info    1075 msgs    : sensor_msgs/CameraInfo
         /camera/depth/image_raw      1075 msgs    : sensor_msgs/Image
         /mocap_node/drone_1/pose    24675 msgs    : geometry_msgs/PoseStamped
         /tf                         24675 msgs    : tf2_msgs/TFMessage
         /tf_static                      5 msgs    : tf2_msgs/TFMessage
```

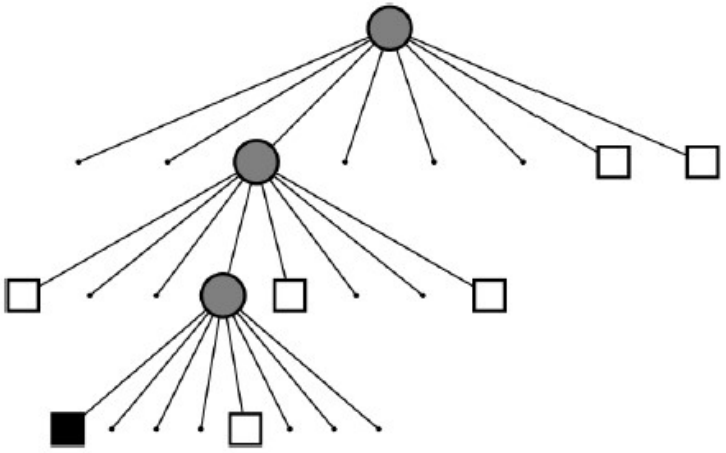
Assignment 1 – 3D obstacle map generation
Octomap – Obstacle map representation

Represents 3D environment as discrete voxel grid

Based on octree implementation

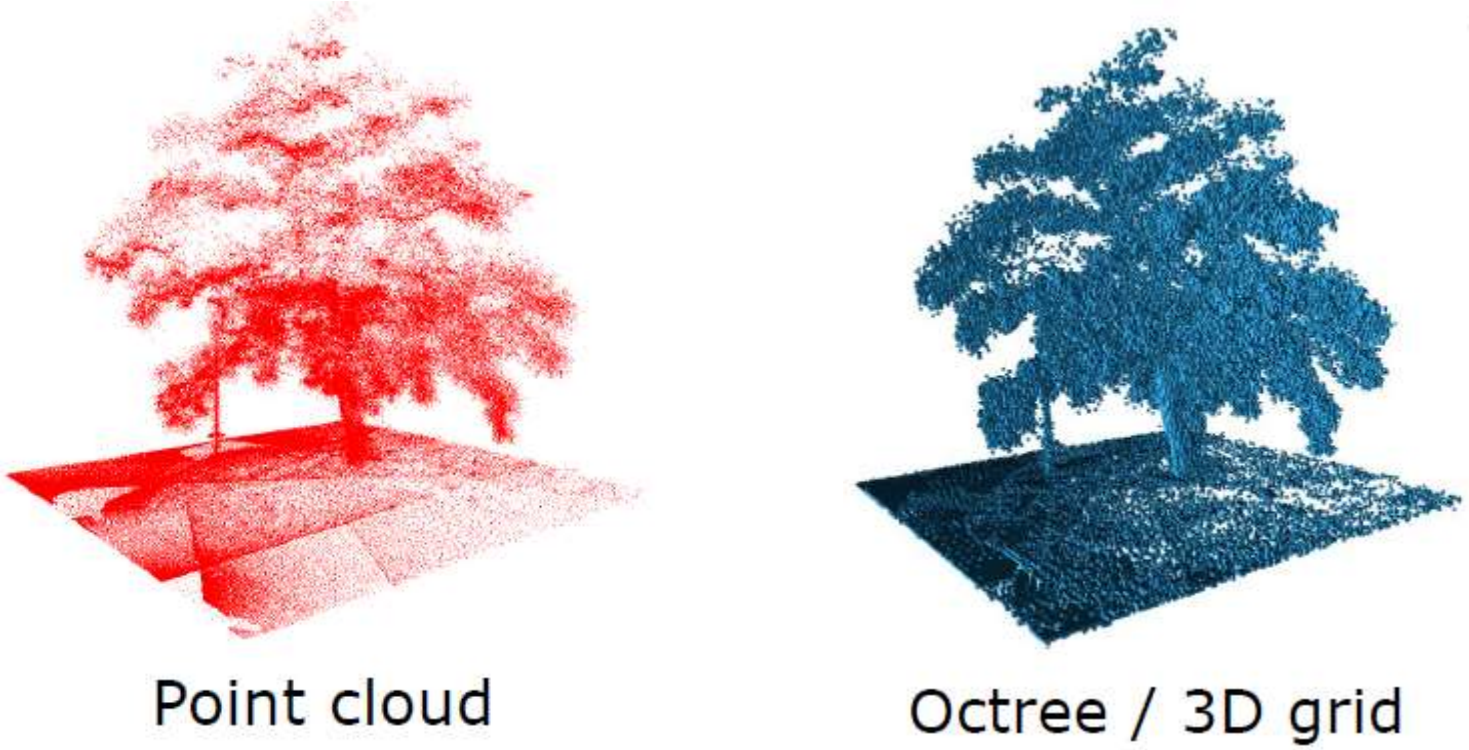
Each cell stores the occupancy probability





Assignment 1 – 3D obstacle map generation
Octomap – Obstacle map representation

Point cloud cannot be used well for path planning



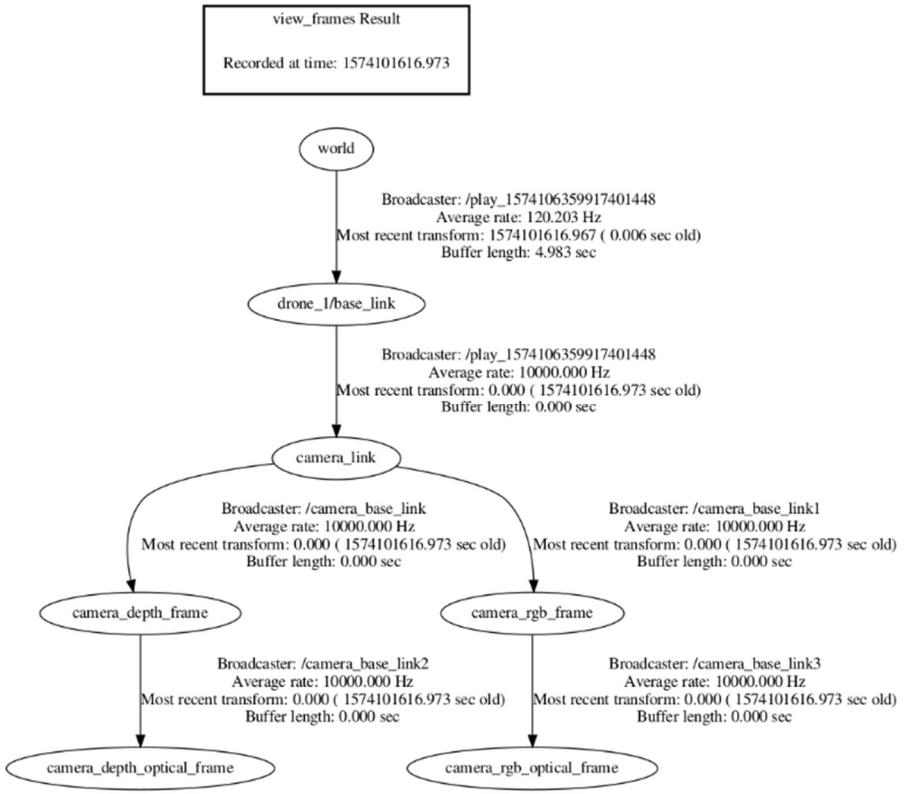
Assignment 1 – 3D obstacle map generation
tf frames – Rigid body coordinate transformations (over time)

Need: pose of the RGB camera with respect to the world frame

Pose (position and orientation) of the drone

Pose of the sensor on the drone

Relative poses of the cameras on the RGB-D sensor.



Assignment 1 – 3D obstacle map generation
Task steps:

(Please, take a look at an assignment writeup)
Inspect the data in the ROS bag using rqt [R6]

Visualize the data in ROS RViz [R7] (2D images and tf position data of the drone)

Read the documentation about the octomap_server node and set up the octomap_server ROS launchfile to configure the node and make it connect to the right topics on the ROS bag

Create an obstacle map using the octomap server node and the provided ROS bag, set the resolution of the octomap to between 10-30 cm

Visualize the octomap in ROS RViz

Store the created octomap as a binary (e.g. bt file) for use in the next tasks.

Create screen capture video of the octomap processing in RVIZ

Assignment 1 – 3D obstacle map generation
Extras:

Other 3D mapping packages:

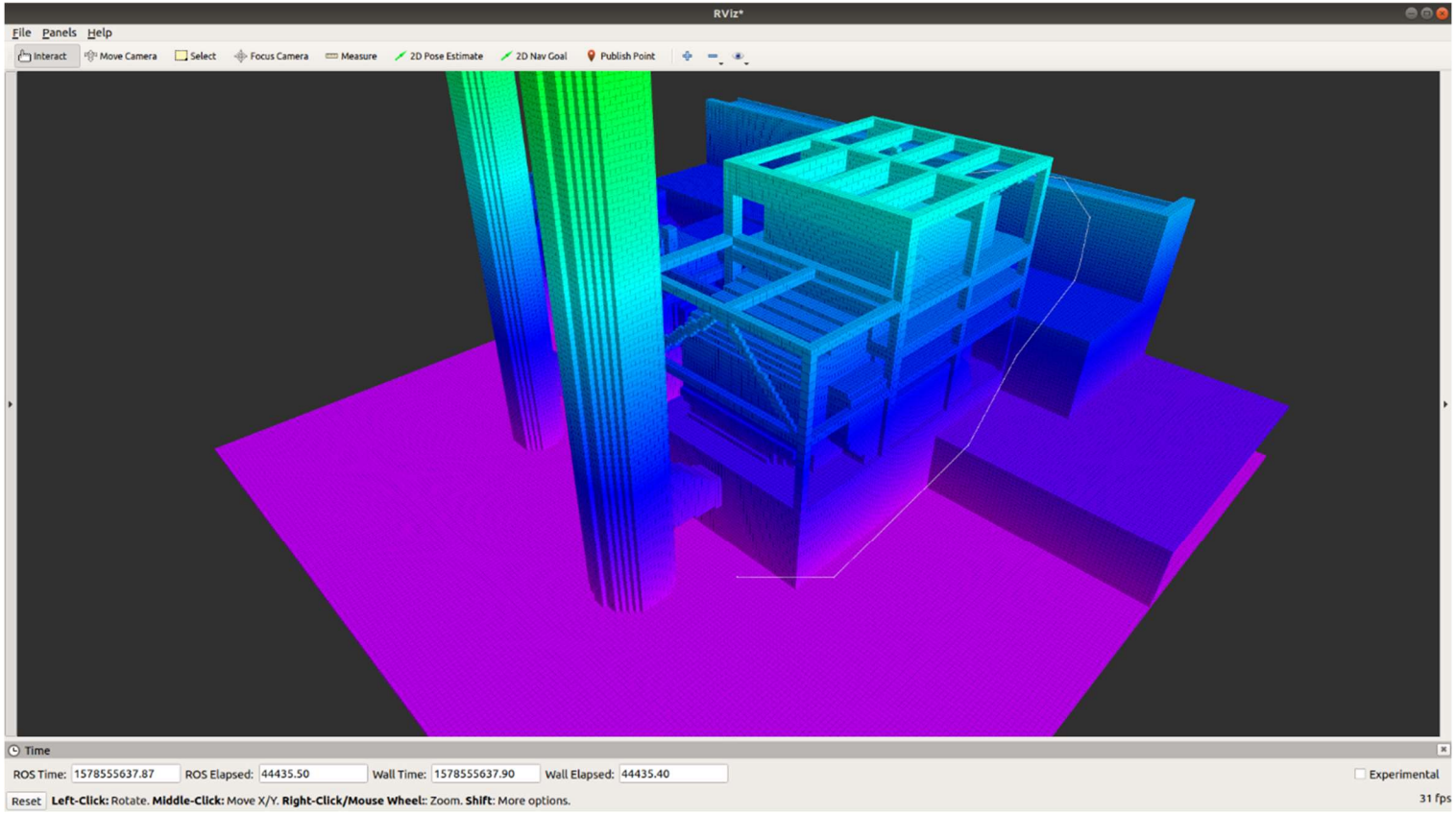
- InfiniTAM [R3]
- Voxblox [R4]

Contents

- **Introduction to ROS practicals**
  - Practical 1
  - **Practical 2**
  - Practical 3
  - Practical 4
- Robot Operating System (ROS)
- Byobu - Detaching from the current shell session
- ROS Bags / rqt / RViz
- Testing a VIO Algorithm

ROS Melodic – Ubuntu 18.04
(there are always turtles on these images)
http://wiki.ros.org/melodic

Assignment 2 – Path planning for safe navigation



Assignment 2 – Path planning for safe navigation
ROS-packages for this part

OMPL - Open Motion Planning Library (OMPL):
https://github.com/ompl/ompl

dynamicEDT3D - Euclidean distance transform (EDT) in 3D
https://docs.ros.org/en/melodic/api/dynamic_edt_3d/html/

Assignment 2 – Path planning for safe navigation
Probabilistic Roadmap (PRM) algorithm

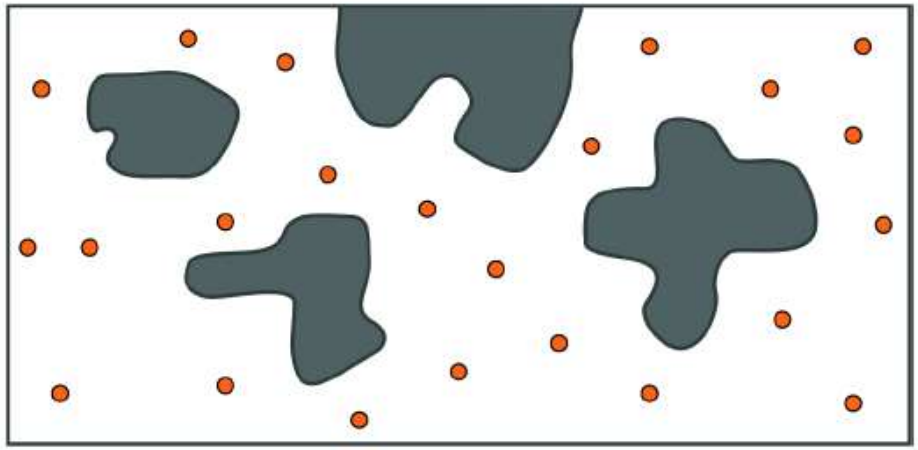


Figure 2.2: One possible set of uniform random samples of the free state space

Assignment 2 – Path planning for safe navigation
Probabilistic Roadmap (PRM) algorithm

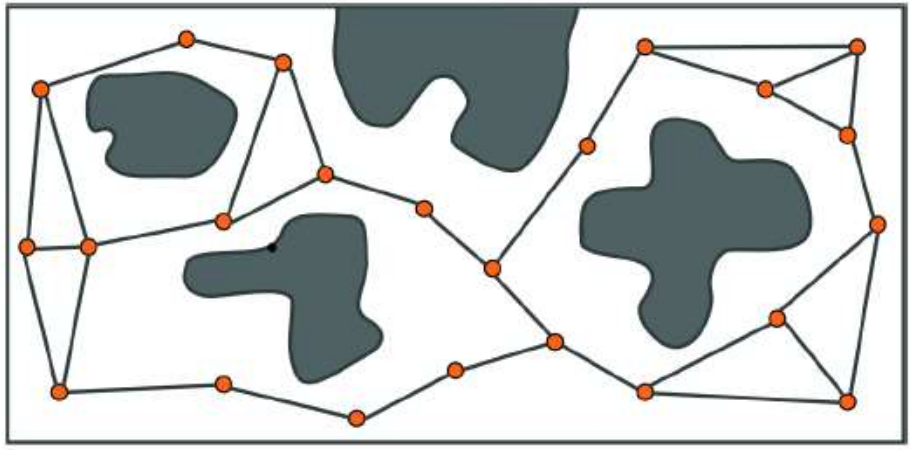


Figure 2.3: Using a local planner, the PRM is formed by connecting samples that are close to one another using a straight path in the free state space.

Assignment 2 – Path planning for safe navigation
Probabilistic Roadmap (PRM) algorithm

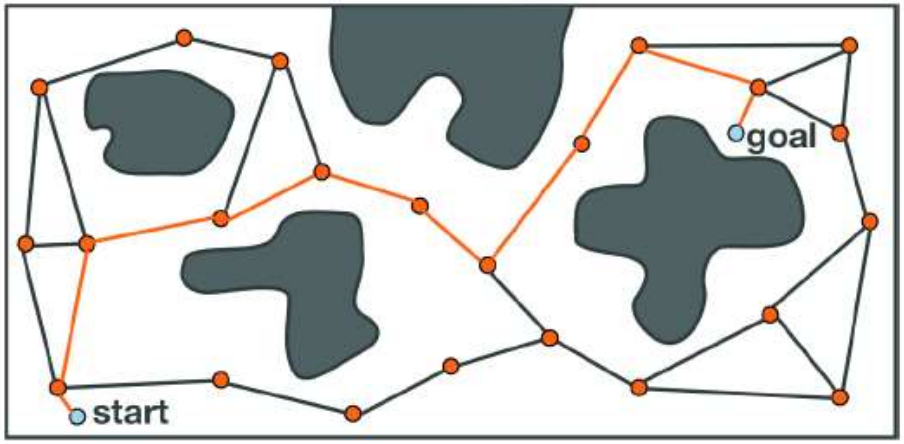


Figure 2.4: An example of a motion planning query on a PRM. The start and the goal are connected to the roadmap, and the shortest path in the graph is found.

Assignment 2 – Path planning for safe navigation DynamicEDT3D package

3D distance maps – pre-computing distances to obstacles

Creates 3D distance maps from gridmaps (and Octomap representations)

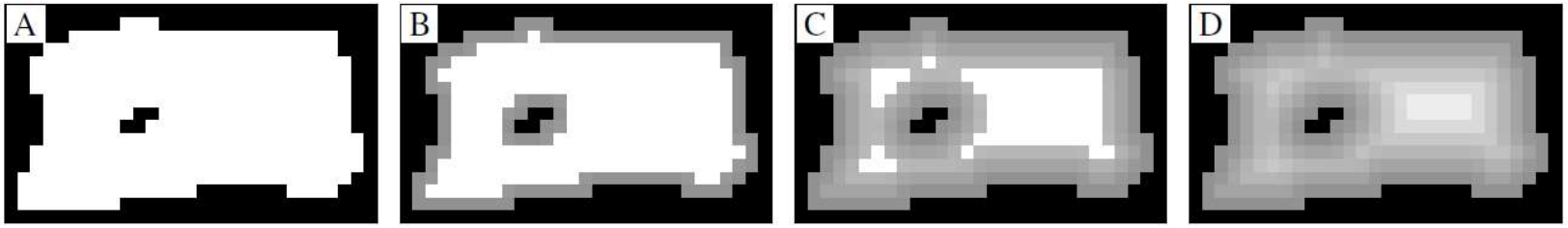


Figure 3: Computing a distance map with the static brushfire algorithm described in Sect. 3. The distance values are initialized with 0 (black) for occupied and infinity (white) for empty cells (A). The occupied cells initiate wavefronts that propagate the increasing distances, denoted by increasing brightness in (B) and (C). A wavefront stops if no further distance values can be lowered. After all wavefronts have stopped, the full distance map is computed (D).

Assignment 2 – Path planning for safe navigation
Tasks/Steps for this part of the practical

4 steps:

Try out path planning algorithms in simple 2D example (code provided)

Create your own simple 3D example

Path planning in 3D with clean synthetic map

Path planning in 3D with map from real data

Code for the 2D path planning is available in the TC

dla2_path_planner_y2019m01d09_v3.zip

(include readme file)

Assignment 2 – Path planning for safe navigation
Path planning on simple 2D example

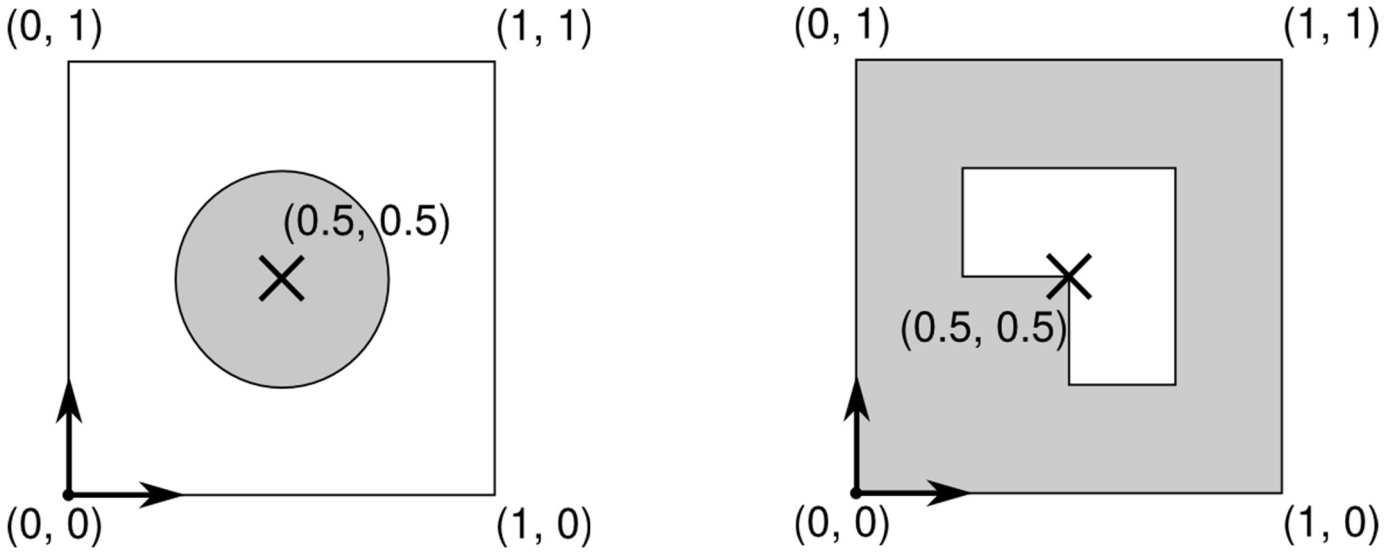
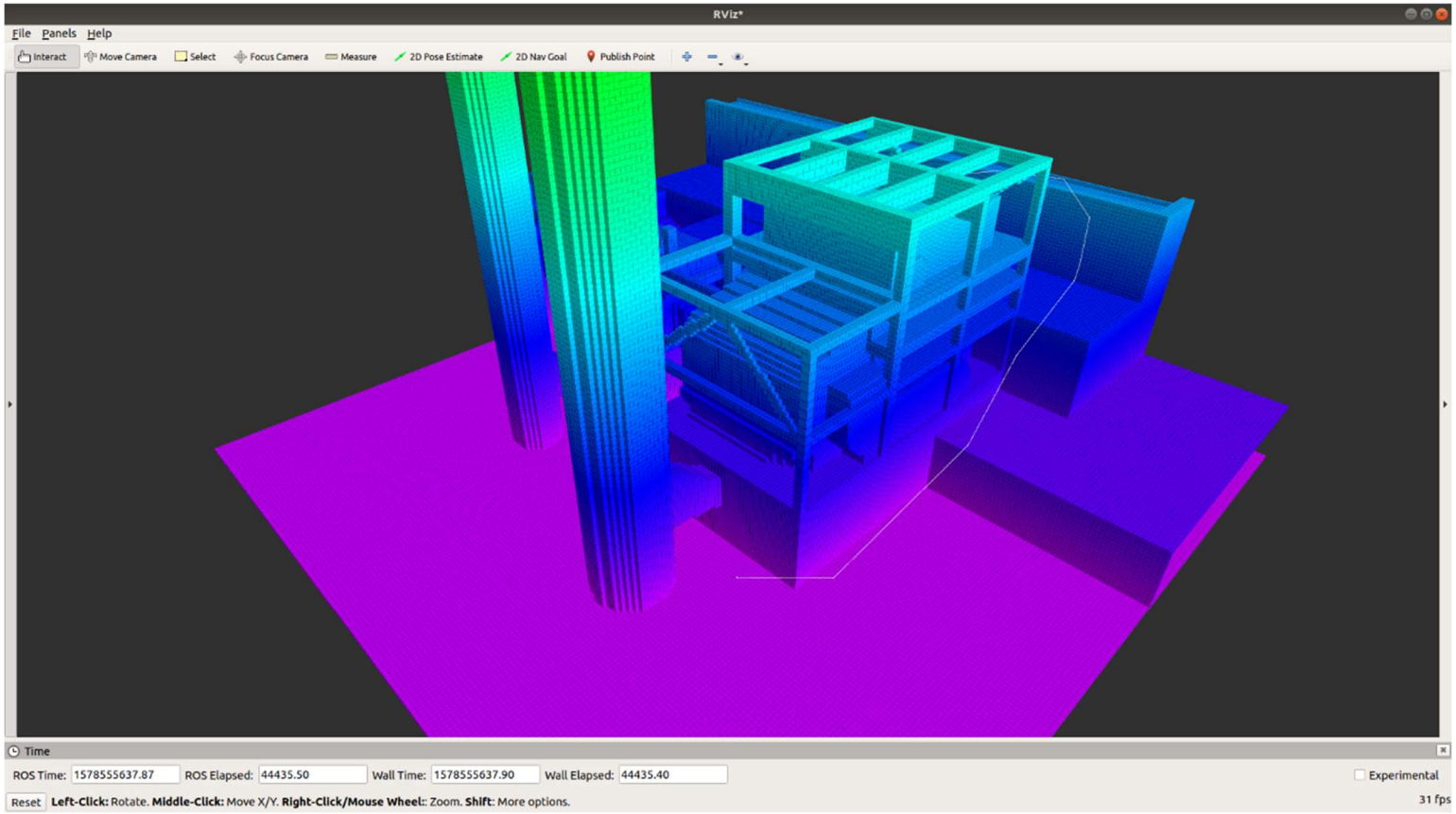
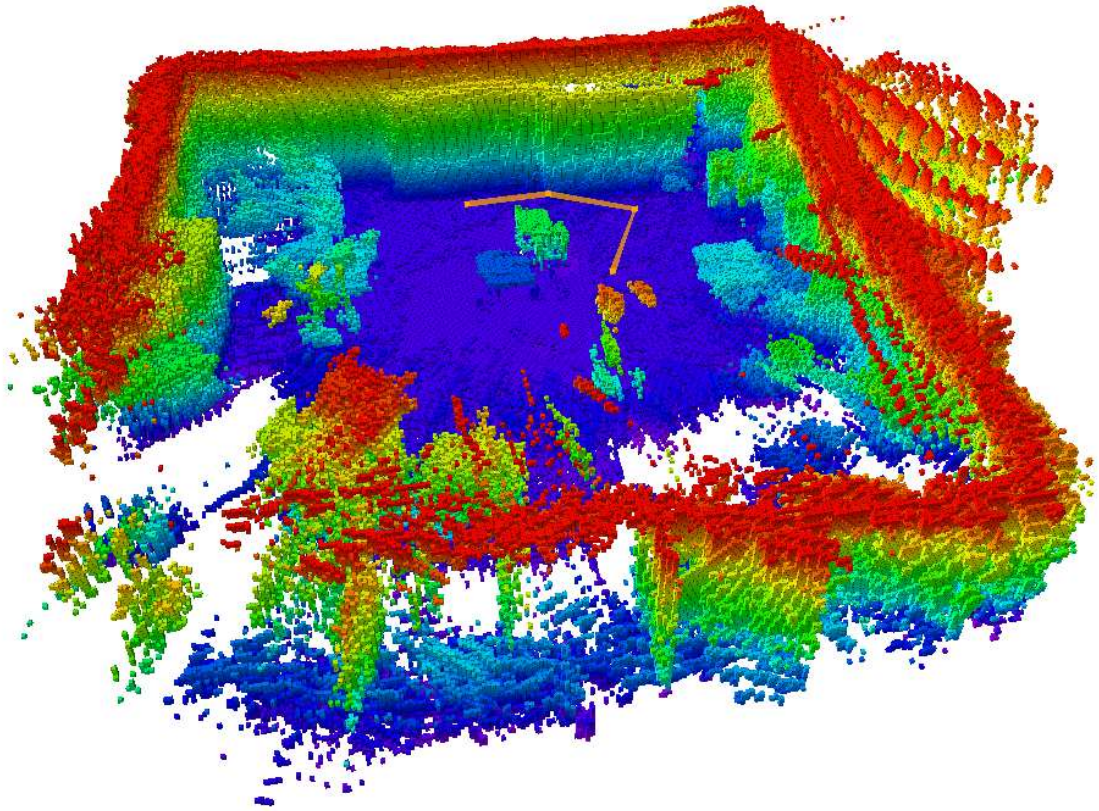


Fig. 1. 2D Toy obstacle map examples:
(left) map used in the OMPL demo demos/OptimalPlanning.cpp;
(right) another possible toy obstacle-map example

Assignment 2 – Path planning for safe navigation
Path planning in 3D with clean synthetic map



Assignment 2 – Path planning for safe navigation
Path planning in 3D with map from real data

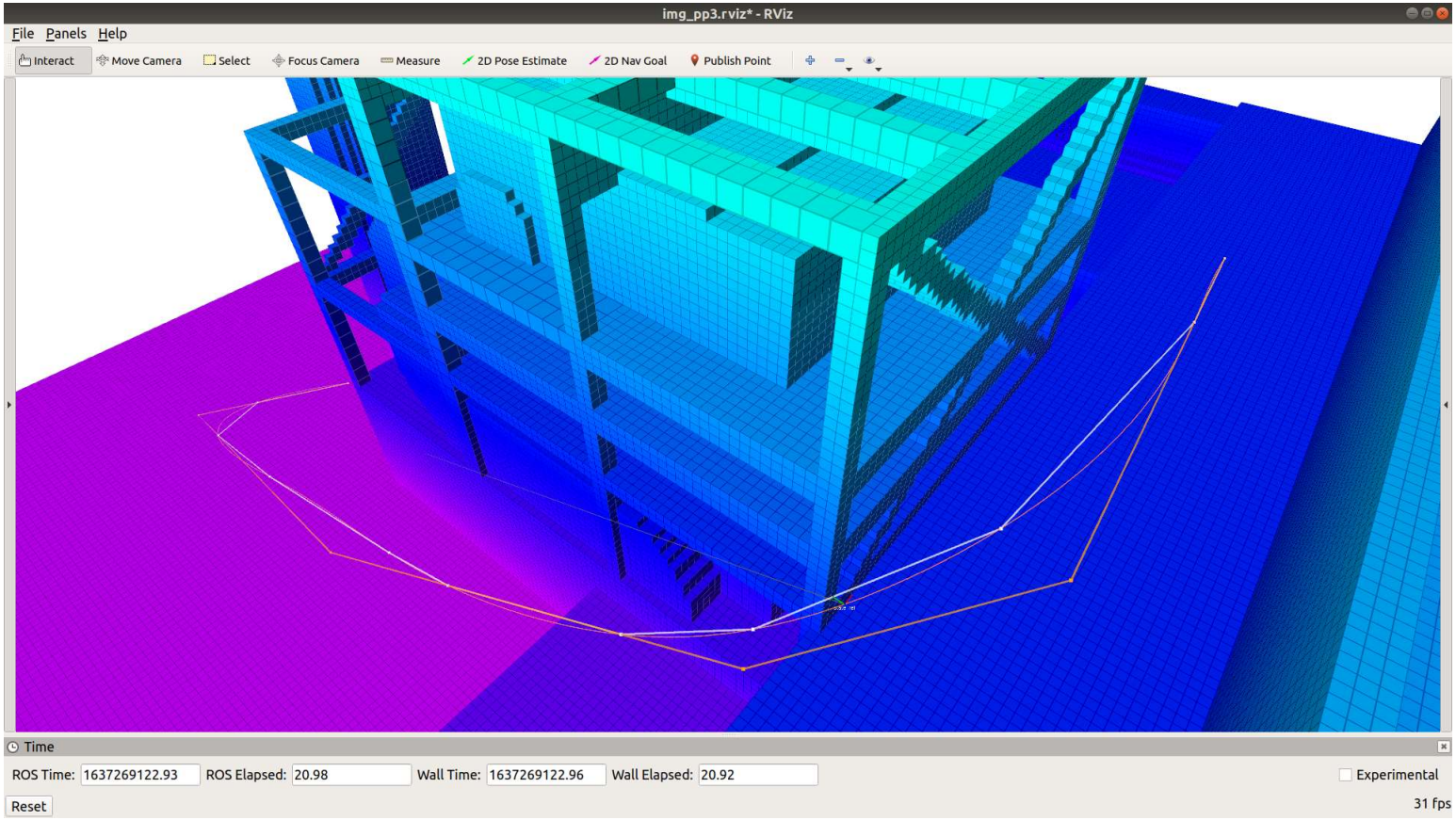


Contents

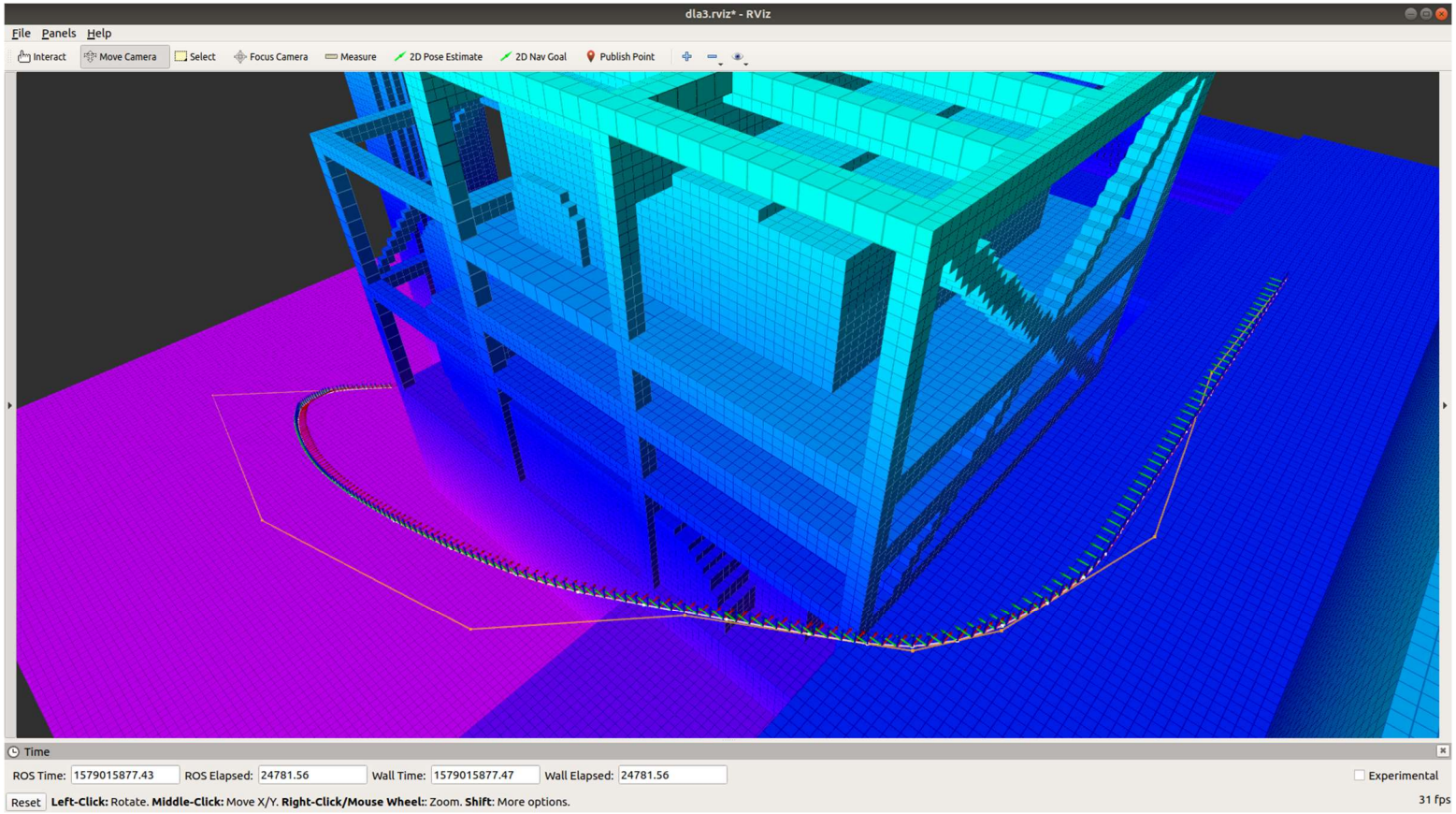
- **Introduction to ROS practicals**
  - Practical 1
  - Practical 2
  - **Practical 3**
  - Practical 4
- Robot Operating System (ROS)
- Byobu - Detaching from the current shell session
- ROS Bags / rqt / RViz
- Testing a VIO Algorithm

ROS Melodic – Ubuntu 18.04
(there are always turtles on these images)
http://wiki.ros.org/melodic

Assignment 3 – Smooth trajectory generation



Assignment 3 – Smooth trajectory generation



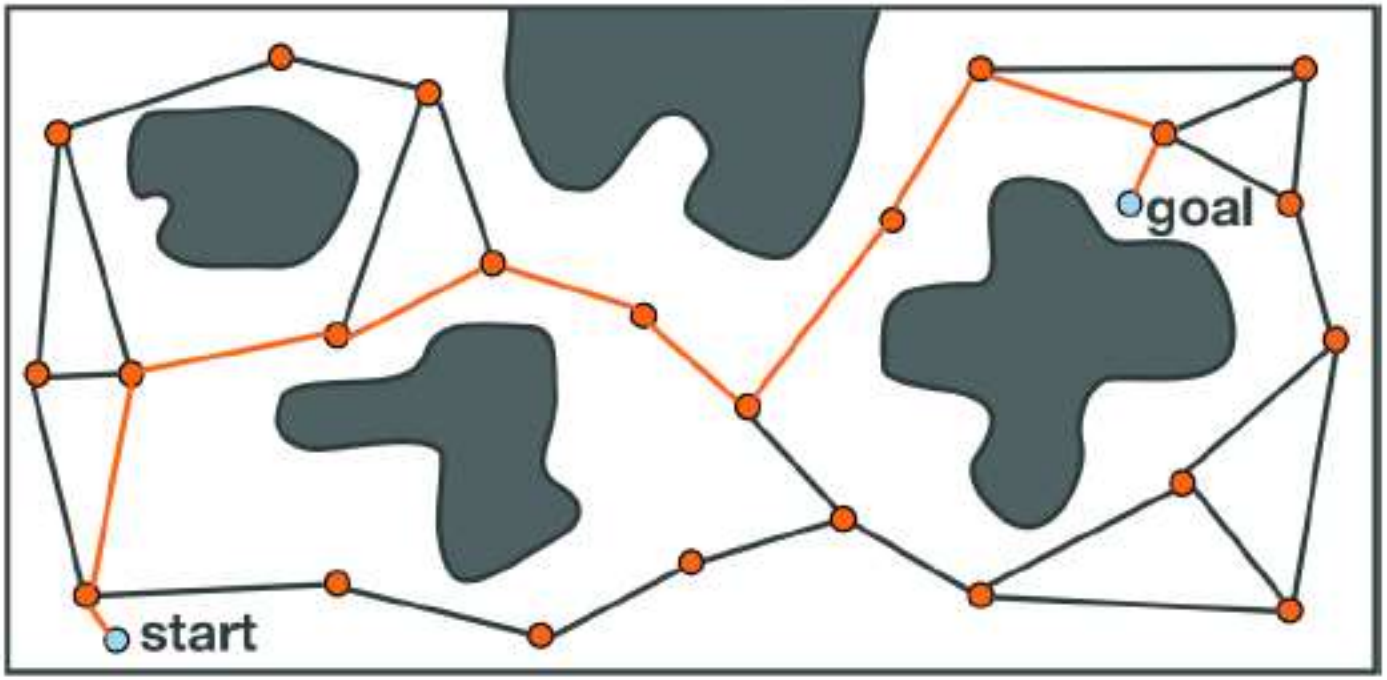
Assignment 3 – Smooth trajectory generation
ROS-packages for this part

OMPL - Open Motion Planning Library (OMPL):
https://github.com/ompl/ompl

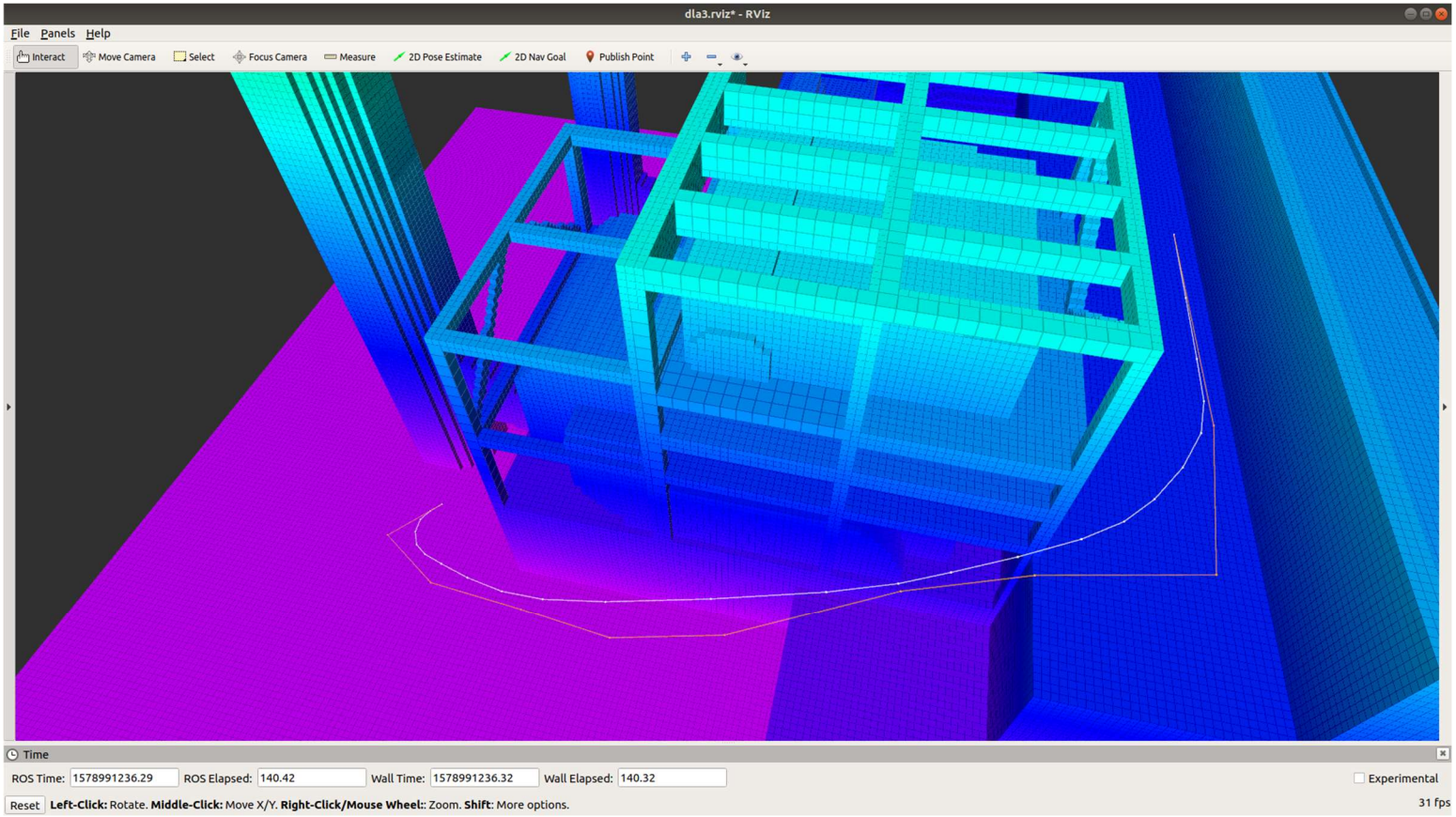
mav_trajectory_generation
https://github.com/ethz-asl/mav_trajectory_generation

Assignment 3 – Smooth trajectory generation PRM paths

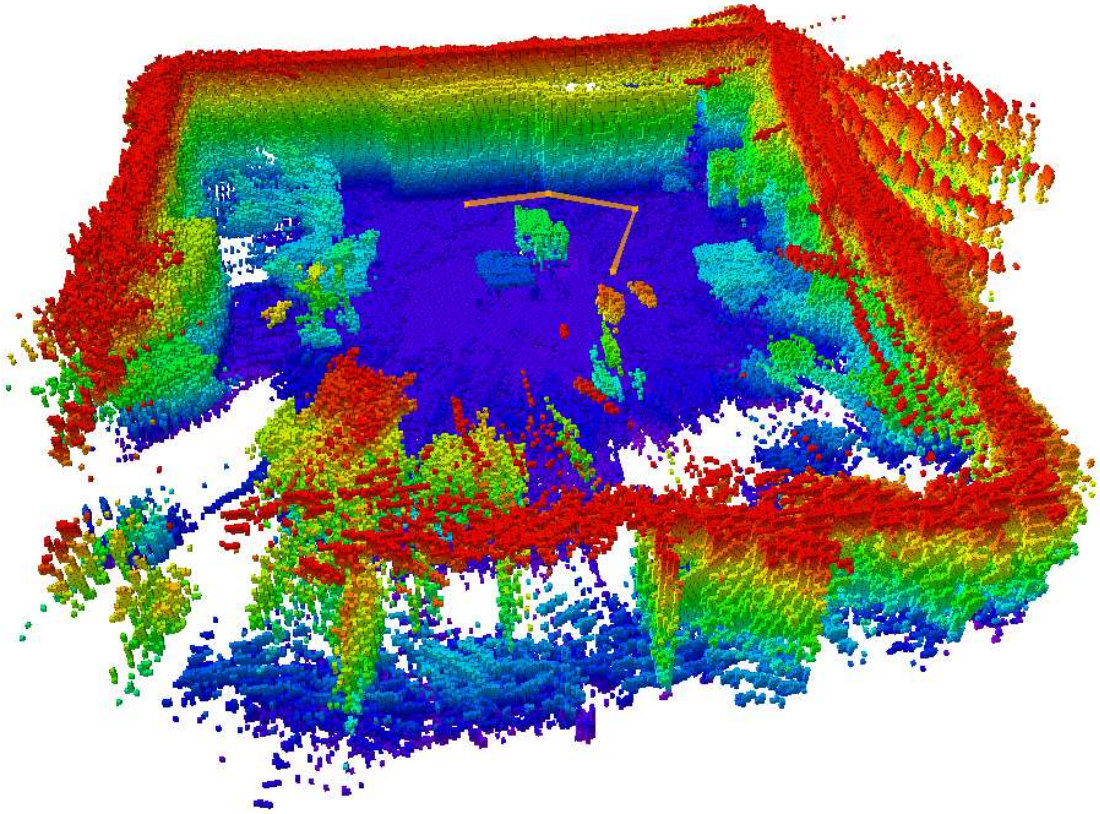
Paths generated with PRM or RRT are not smooth



Assignment 3 – Smooth trajectory generation
Example for not smooth path



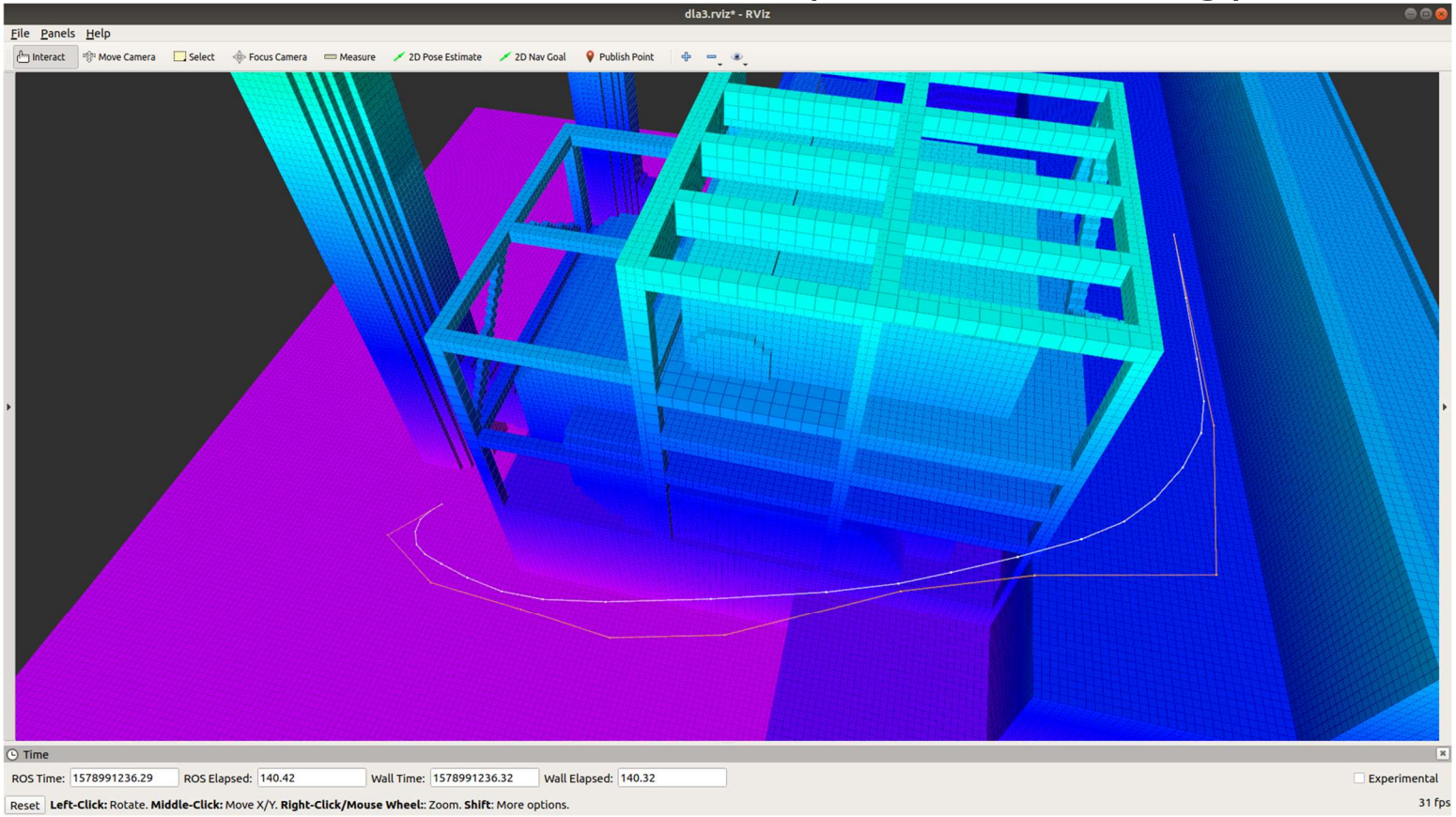
Assignment 3 – Smooth trajectory generation
Example for not smooth path



Assignment 3 – Smooth trajectory generation
Tasks/Steps for this part of the practical

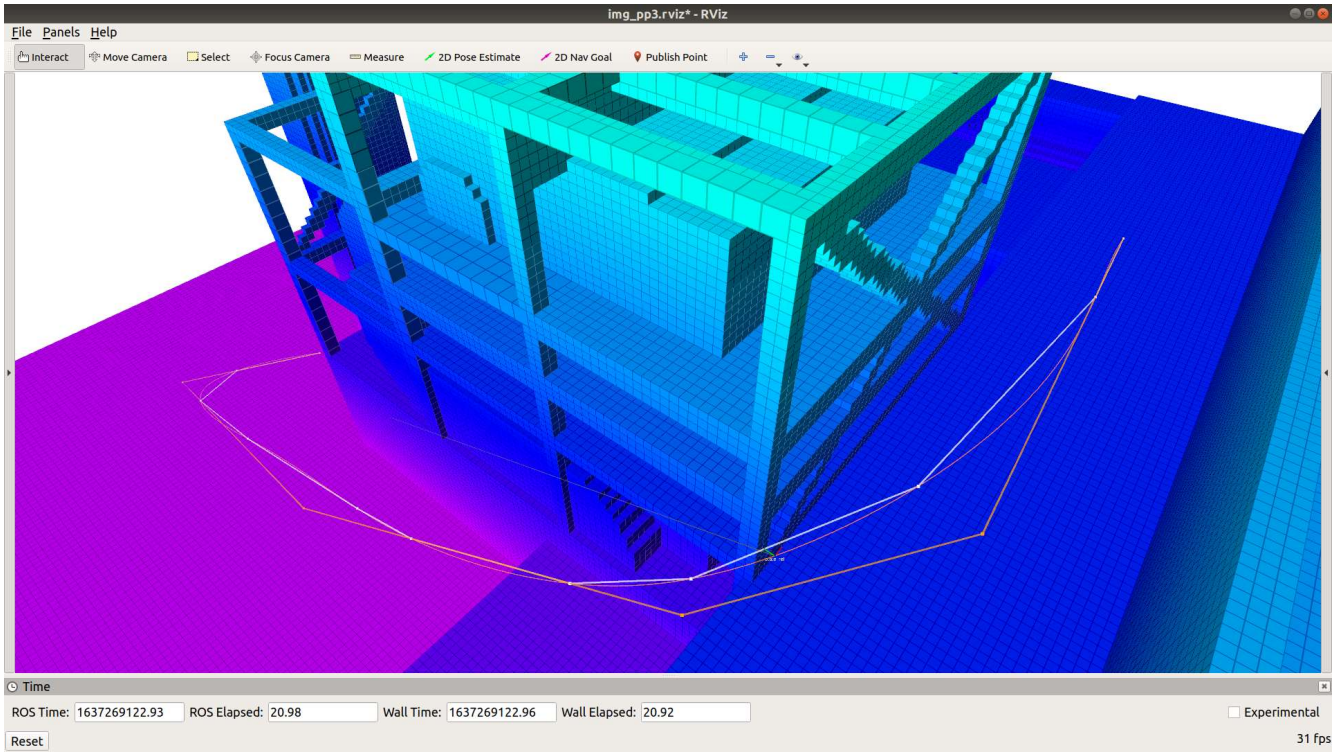
A. Cost-aware path simplification (or shortening)
B. Smooth trajectory generation
C. (Optional) Check that the smooth trajectory is obstacle-free (and modifying the planned path when it is not)

Assignment 3 – Smooth trajectory generation
Cost-aware path simplification (or shortening)

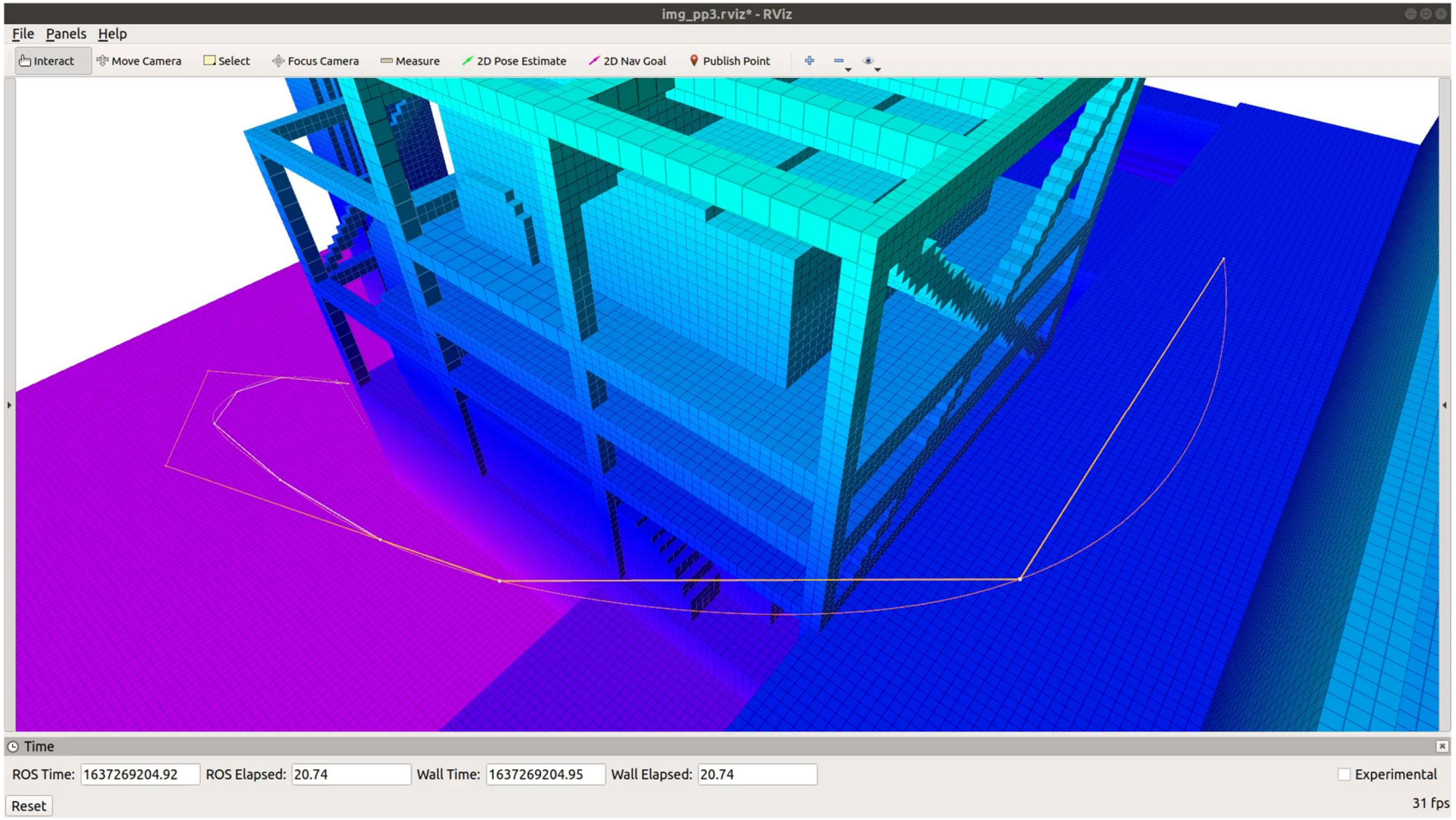


Assignment 3 – Smooth trajectory generation
Smooth trajectory generation

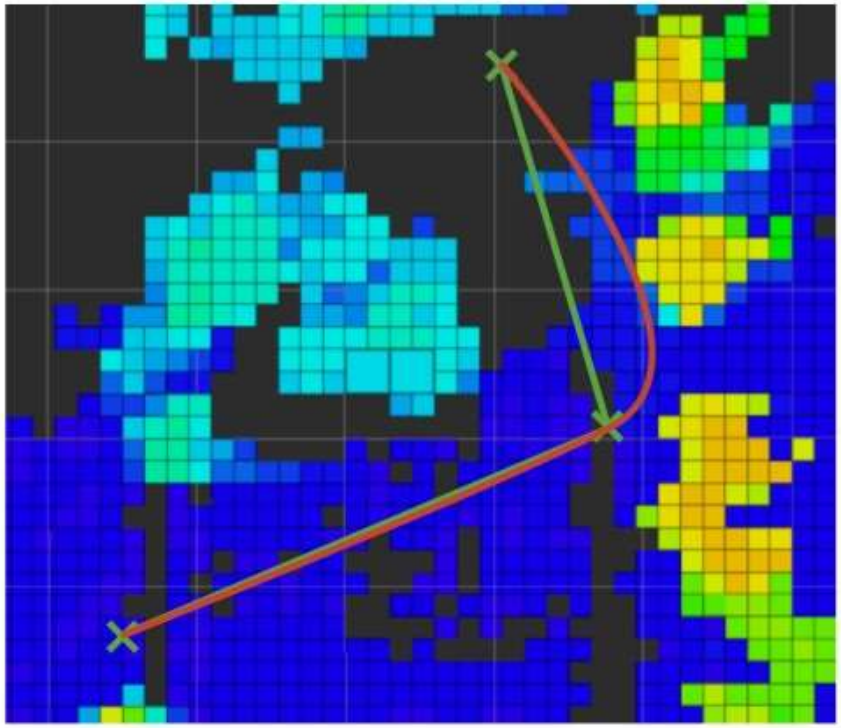
ROS package: mav_trajectory_generation

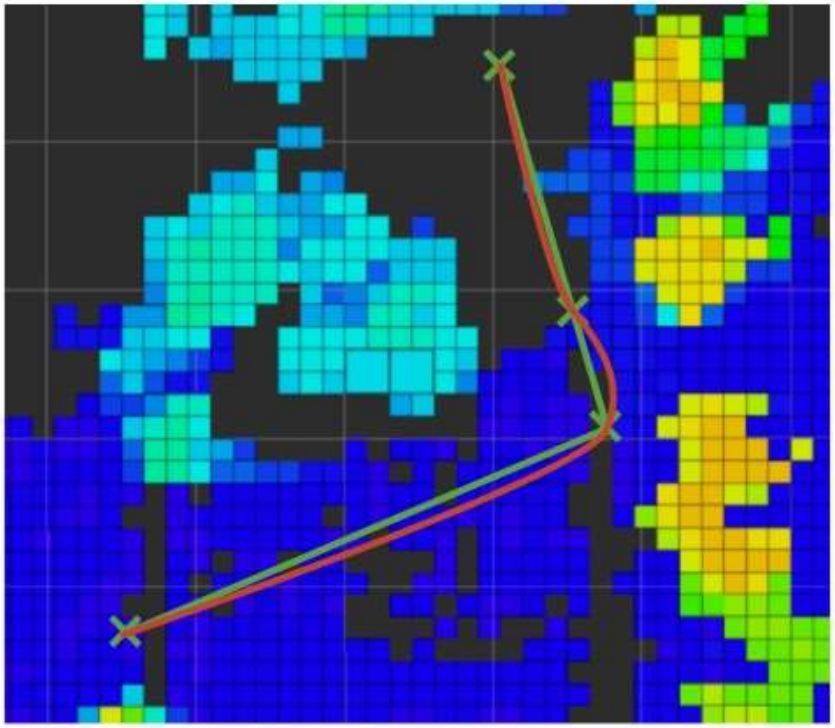


Assignment 3 – Smooth trajectory generation Video



Assignment 3 – Smooth trajectory generation
Checking if obstacle free



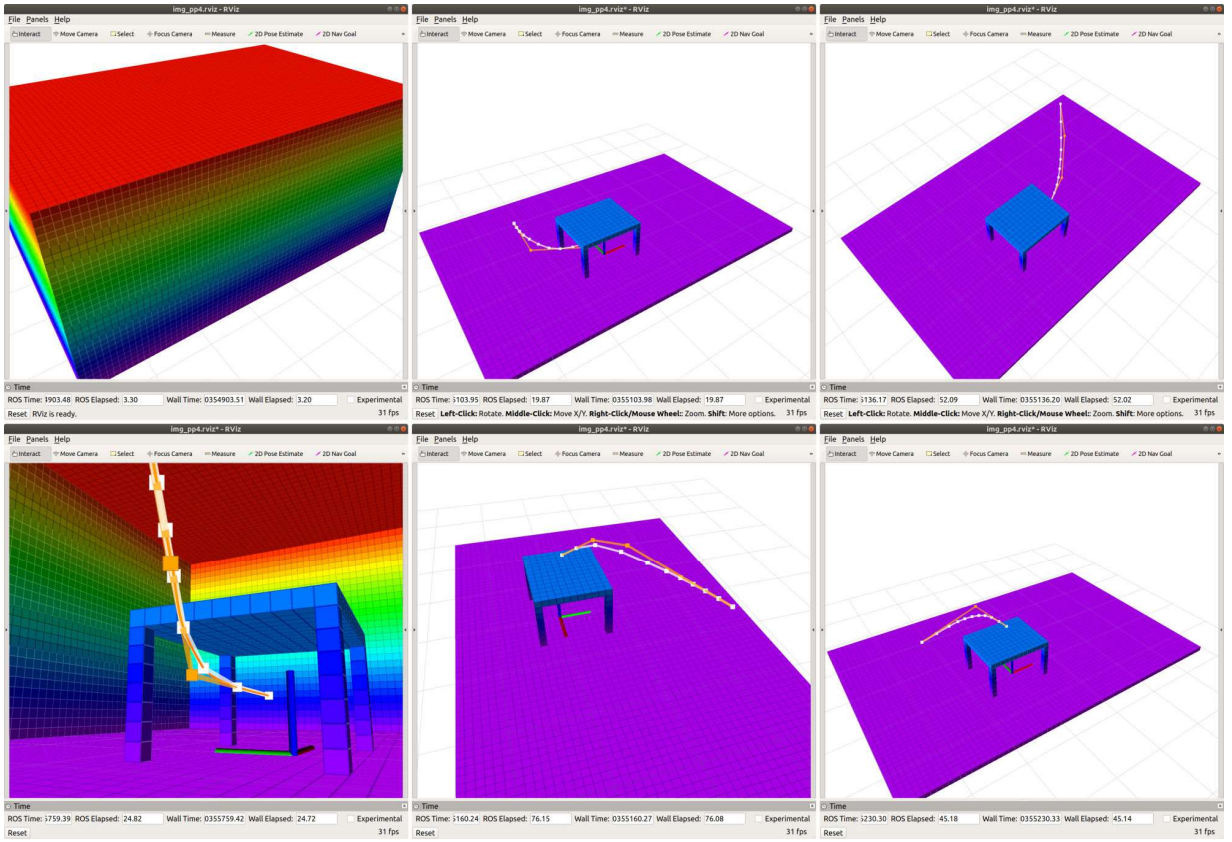


Contents

- **Introduction to ROS practicals**
  - Practical 1
  - Practical 2
  - Practical 3
  - **Practical 4**
- Robot Operating System (ROS)
- Byobu - Detaching from the current shell session
- ROS Bags / rqt / RViz
- Testing a VIO Algorithm

ROS Melodic – Ubuntu 18.04
(there are always turtles on these images)
http://wiki.ros.org/melodic

Assignment 4 – Pre-calculation of trajectories for experiment

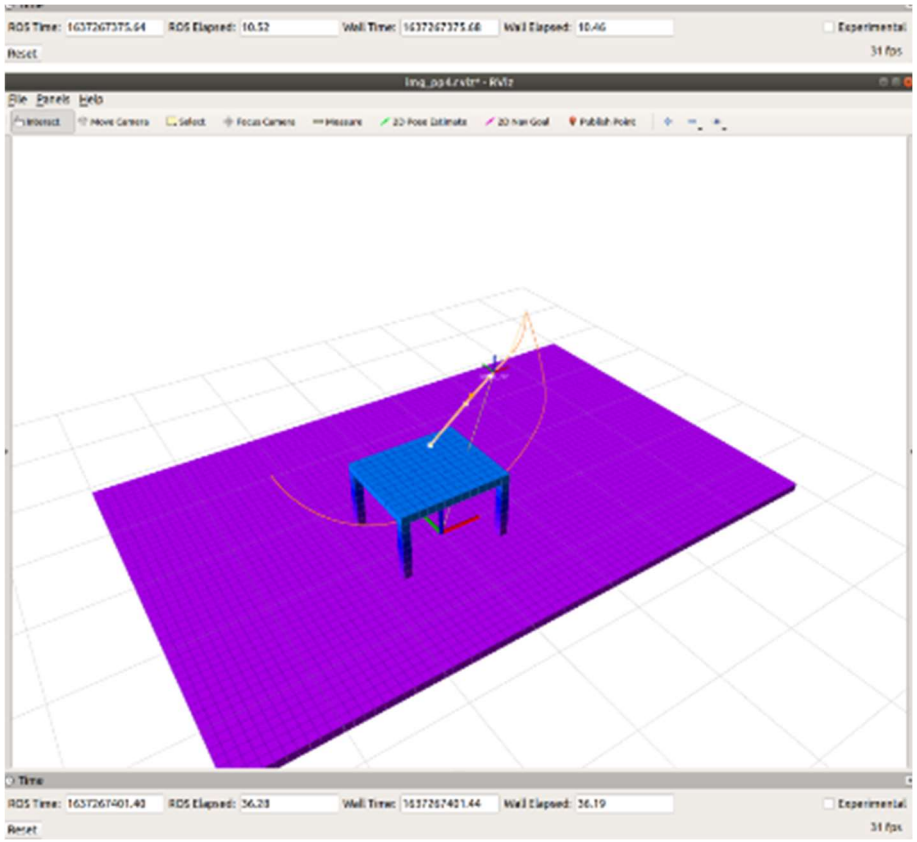


Assignment 4 – Pre-calculation of trajectories for experiment
Tasks/Steps for this part of the practical

Trajectory planning for synthetic Dronespace map

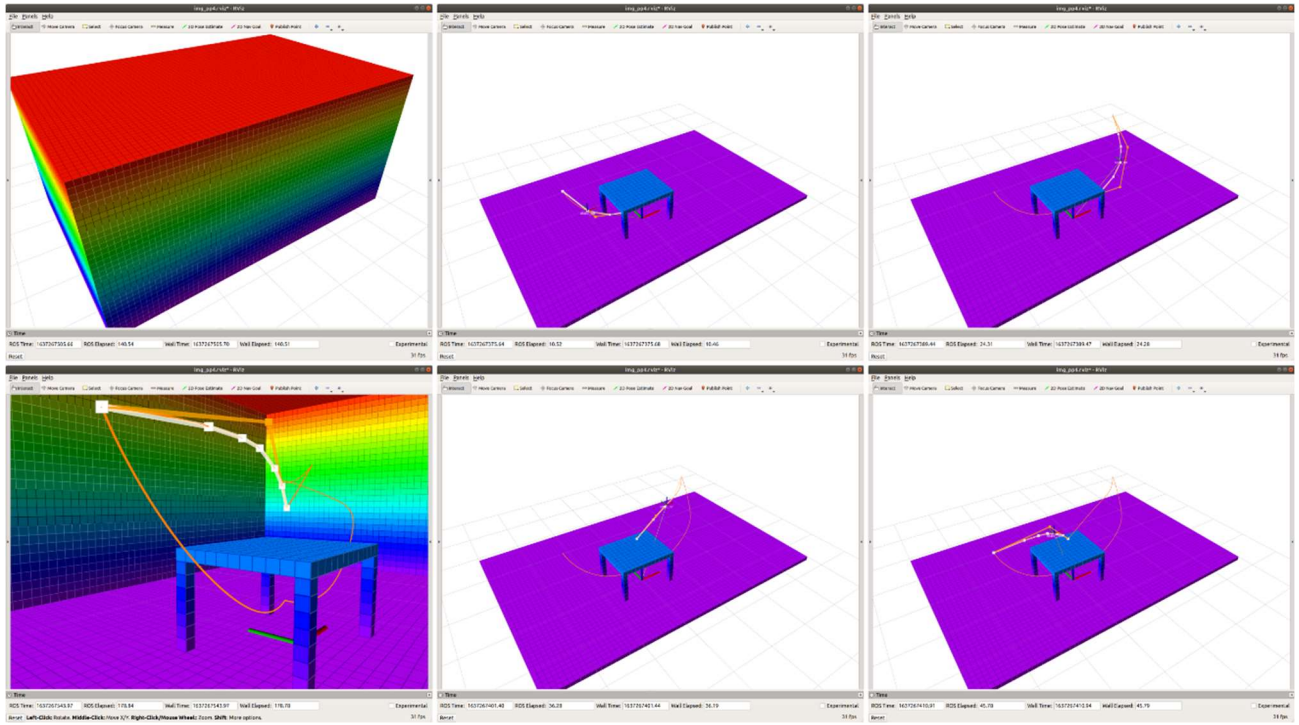
Recording of a ROS bag with the planned trajectories

Assignment 4 – Pre-calculation of trajectories for experiment Synthetic map



Assignment 4 – Pre-calculation of trajectories for experiment
Two maps are needed

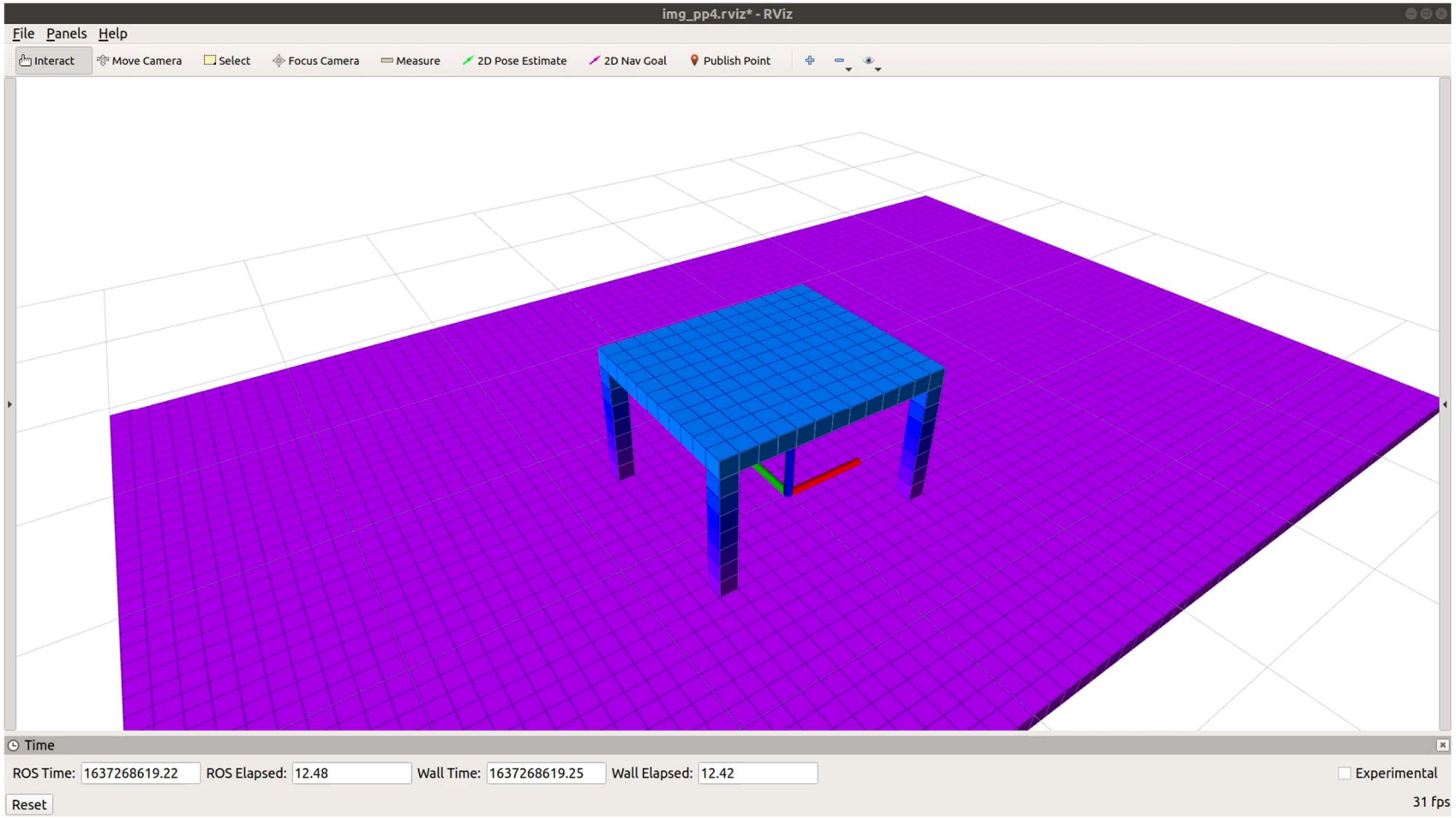
Map for planning needs boundaries



Assignment 4 – Pre-calculation of trajectories for experiment Flight experiment schedule

1. Take-off (landed position approximately directly below waypoint 0)
2. Fly to waypoint 0
3. Fly to waypoint 1 (below table)
4. Fly to waypoint 2 (opposite end of the Dronespace)
5. Fly to waypoint 3 (above table)
6. Fly to waypoint 4 (== waypoint 0)
7. Land

Assignment 4 – Pre-calculation of trajectories for experiment Video



Contents

- Introduction to ROS practicals
- **Robot Operating System (ROS)**
- **Byobu - Detaching from the current shell session**
- ROS Bags / rqt / RViz
- Testing a Visual Inertial Odometry (VIO) Algorithm

ROS Melodic – Ubuntu 18.04
(there are always turtles on these images)
http://wiki.ros.org/melodic

Robot Operating System (ROS)
Byobu - Detaching from the current shell session

- wstool – catkin workspace inspection
- ROS Topics
- ROS Services
- Checkout “vscode” – Visual Studio Code
- Start a software architecture with a single shell command using byobu

ROS/VM/… terminal commands

```
# 0. Start the 'cdl_2023' Virtual Machine (VM), using 'Oracle VM VirtualBox Manager'

# 1. Open 'terminator' (about it: https://en.wikipedia.org/wiki/GNOME_Terminator )

# 2. The $CMAKE_PREFIX_PATH environment variable allows to know the current ROS workspace
overlay (more information here:
http://wiki.ros.org/catkin/Tutorials/workspace_overlaying)
echo $CMAKE_PREFIX_PATH
/home/student/catkin_ws/devel:/opt/ros/melodic
```

ROS/VM/… terminal commands

```
# 3. The ROS_WORKSPACE is located in: ~/catkin_ws
cd ~/catkin_ws

# 4. Information subfolders:
ls
bags  build  devel  logs  src  start_byobu_example_msckf_vio.sh  start_byobu_example.sh
#   src folder: source files (code) for ROS packages
#   build folder: when building the ROS workspace, all the intermediary build files are
stored here
#   devel folder: when building the ROS workspace, all the final binaries (libraries,
executables, ...) end up here
#   byobu scripts to test the lecture examples:
{'start_byobu_example_msckf_vio.sh','start_byobu_example.sh'}
#   bags folder: contains a dataset to run the msckf_vio test
```

ROS/VM/… terminal commands

```
# 5. Compiling the ROS workspace
#   from the terminal
cd ~/catkin_ws
catkin build -DCMAKE_BUILD_TYPE=Release -j4
#   from visual studio code, build task configured here: ~/catkin_ws/src/.vscode/tasks.json
cd ~/catkin_ws/src
code .
#   Inside vscode: press 'control+P'; write '> Tasks: Run Task'; select 'catkin build'

# 6. Using wstool to check that the repositories are up-to-date:
#    Note: it may be better not to run 'wstool update', as I have not updated (and tested) this for some time.
cd ~/catkin_ws/src
wstool status
wstool update
wstool help
```

ROS/VM/… terminal commands

```
# 7. ROS topics
cd ~/catkin_ws
#   inspect the byobu script: start_byobu_example.sh
nano start_byobu_example.sh
code nano start_byobu_example.sh
#   in the script you can find the commands to test ROS topics using the ros_tutorials package
(terminal 1) roscore
(terminal 2) rosrun roscpp_tutorials talker
(terminal 3) rosrun roscpp_tutorials listener
(terminal 4) rostopic echo /chatter
#   test the script: start_byobu_example.sh
#   to learn how to navigate the byobu sub-terminals please check the video: https://youtu.be/NawuGmcvKus
./start_byobu_example.sh
#   check what happens when you stop and restart the talker and listener nodes; as well as the 'rostopic echo /chatter' node.
#   check what happens if you stop/restart the roscore
```

ROS/VM/… terminal commands

```
# 8. ROS services
(terminal 1) roscore
(terminal 2) rosrun roscpp_tutorials add_two_ints_server
(terminal 3) rosservice call /add_two_ints "a: 1
b: 2"
(terminal 4) rosrun roscpp_tutorials add_two_ints_client
(terminal 4) rosrun roscpp_tutorials add_two_ints_client 3 5
```

Contents

- Introduction to ROS practicals
- Robot Operating System (ROS)
- Byobu - Detaching from the current shell session
- **ROS Bags / rqt / RViz**
- **Testing a Visual Inertial Odometry (VIO) Algorithm**

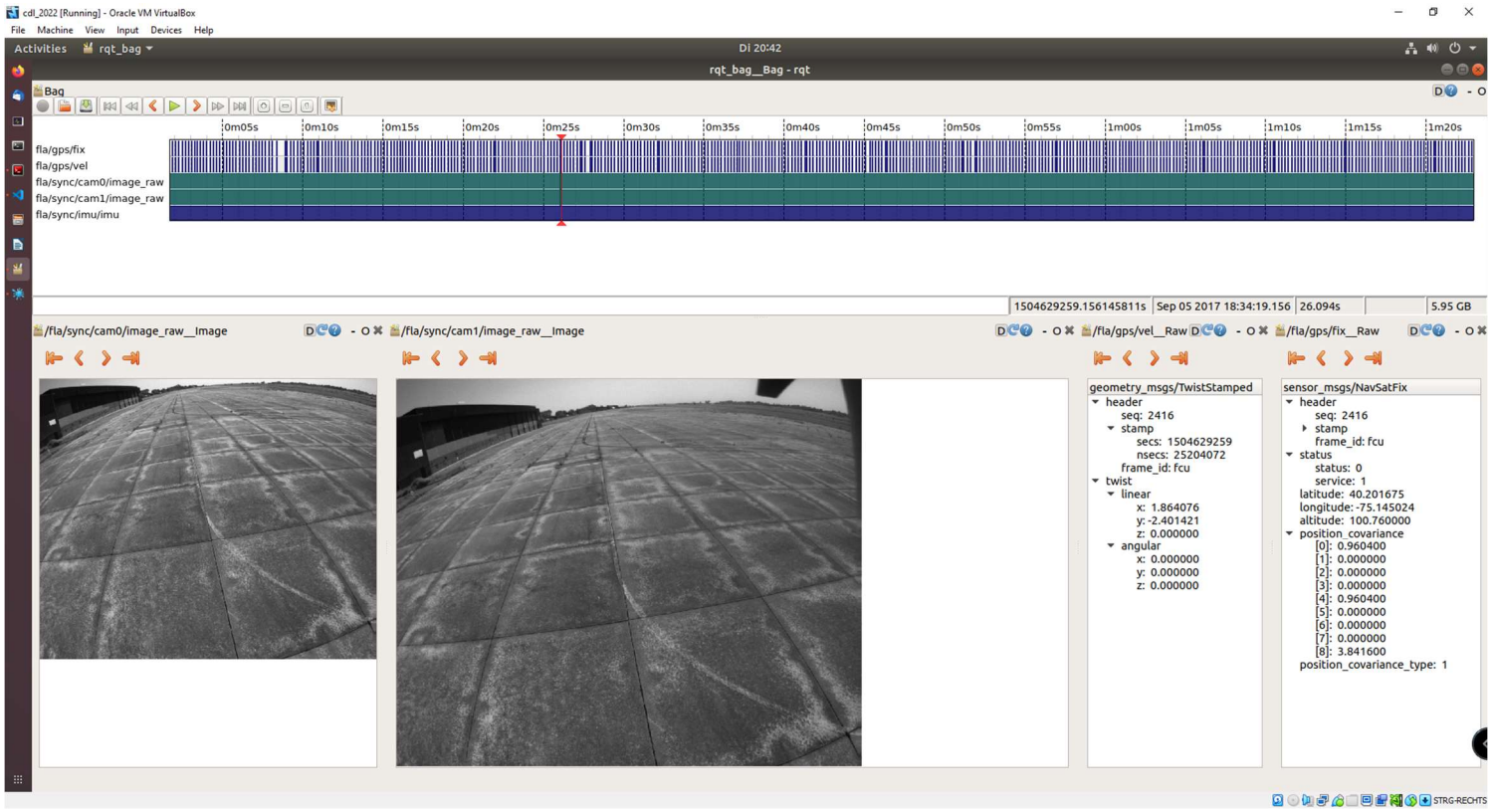
ROS Melodic – Ubuntu 18.04
(there are always turtles on these images)
http://wiki.ros.org/melodic

ROS Bags / rqt / RViz
Testing a Visual Inertial Odometry (VIO) Algorithm

- Inspect log data – a rosbag – using rqt:
  - Inspection ROS Topic data: sensor data, images, …
- ROS Bag data processing and visualization of 2D/3D data in real-time:
  - Once the setup is configured this would work with the drone transmitting the data over WiFi
- Watch the stereo MSCKF VIO algorithm in action

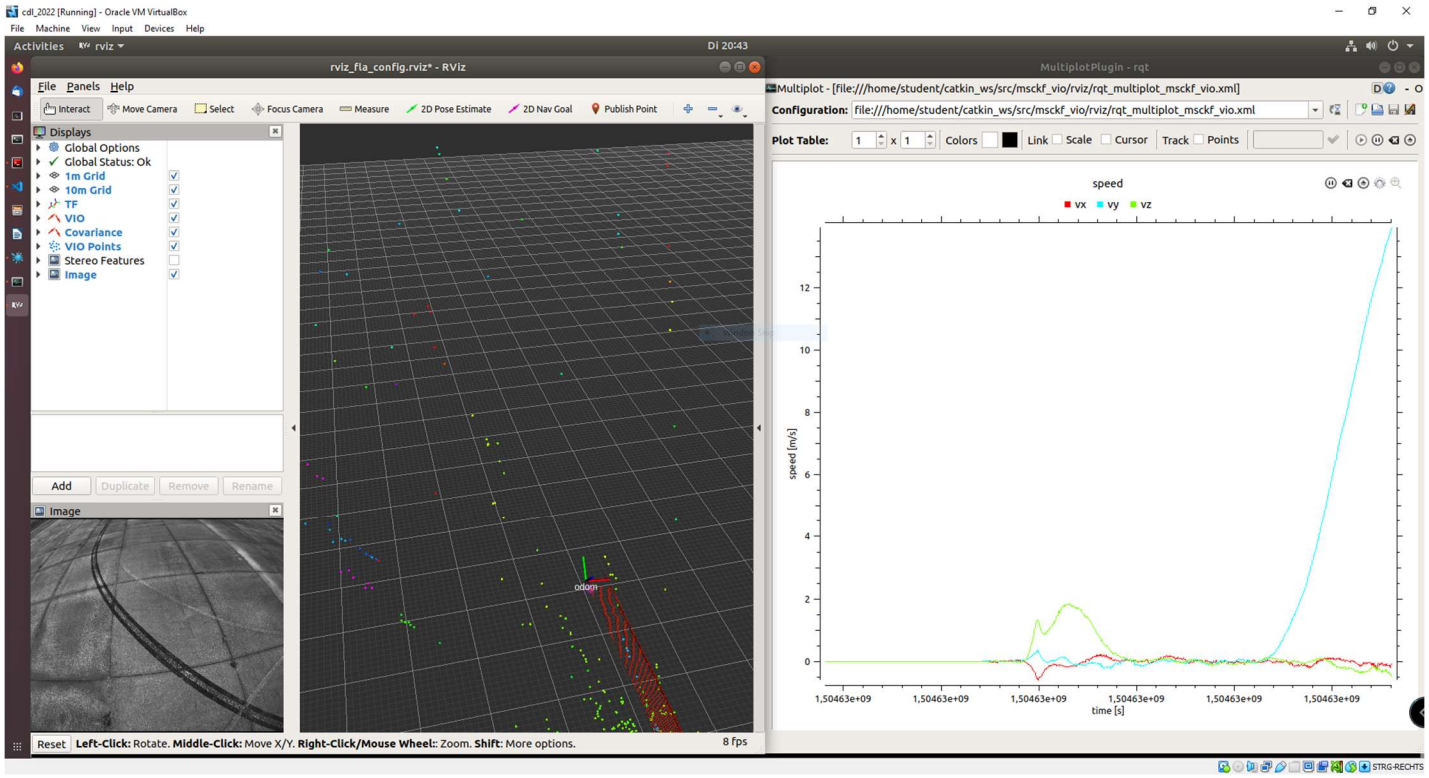
ROS/VM/… terminal commands

```
# 9. Inspect the bagfile, which we will use to test the msckf_vio package
cd ~/catkin_ws/bags
(terminal 1) roscore
(terminal 2)
  rqt_bag fla_wg_175.bag
```



ROS/VM/… terminal commands

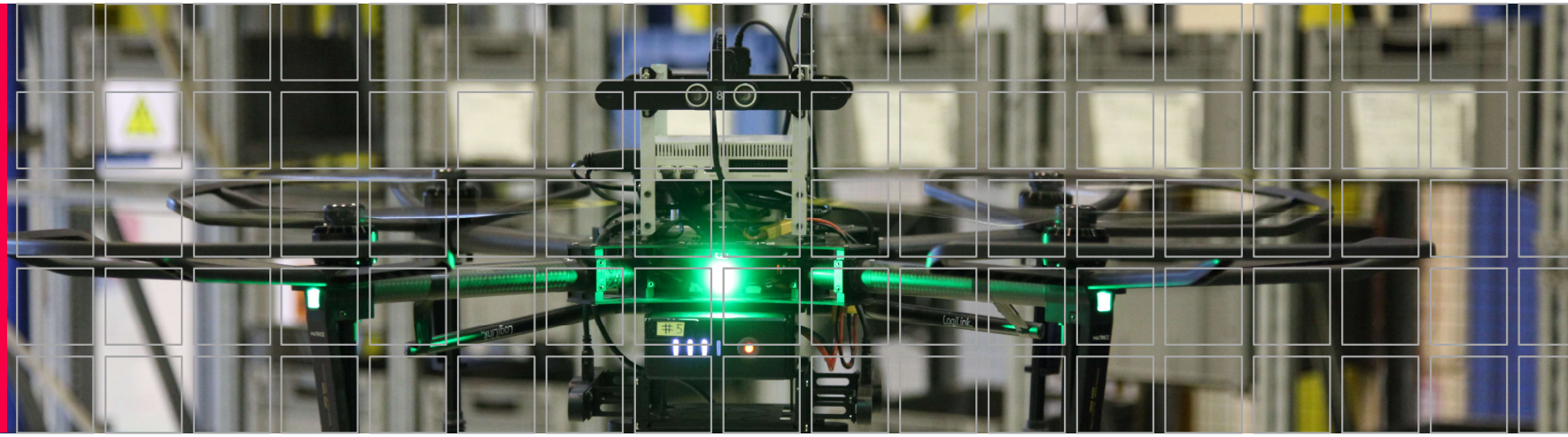
```
# 10. Watch stereo MSCKF VIO algorithm in action
cd ~/catkin_ws
./start_byobu_example_msckf_vio.sh
```



ROS Practical (I)
Camera Drones Lecture

Jesús Pestana, Friedrich Fraundorfer

Contact: jesus.pestana@pro2future.at



ROS Practical (II)
Camera Drones Lecture

Jesús Pestana, Friedrich Fraundorfer

Graz University of Technology
Institute for Computer Graphics and Vision
25 October 2023

Contents

- **Motivation**
- ROS at the Dronespace

ROS Melodic – Ubuntu 18.04
(there are always turtles on these images)
http://wiki.ros.org/melodic

Motivation – Robotics
What are robots good for? – The 3 Ds of Robotics

https://www.tugraz.at

https://www.amazon.co.uk/Disney-Pixar-Wall-Interaction-Interactive/dp/B001B1T0S0

Motivation – Robotics
What are robots good for? – The 3 Ds of Robotics

- The 3 Ds of Robotics refer to tasks performed by a robot
- Waste Allocation Load Lifter **Earth**-Class (WALL-E)

https://www.amazon.co.uk/Disney-Pixar-Wall-Interaction-Interactive/dp/B001B1T0S0

Motivation – Robotics
What are robots good for? – The 3 Ds of Robotics

- The 3 Ds of Robotics refer to tasks performed by a robot
- Waste Allocation Load Lifter **Earth**-Class (WALL-E)
  - Dull – Such as in boring tasks (Any long and arduous task is a good candidate – Try driving 24 hours without resting and not fall asleep)
  - Dirty – Such as in the presence of toxic substances

https://www.amazon.co.uk/Disney-Pixar-Wall-Interaction-Interactive/dp/B001B1T0S0

Motivation – Robotics
What are robots good for? – The 3 Ds of Robotics

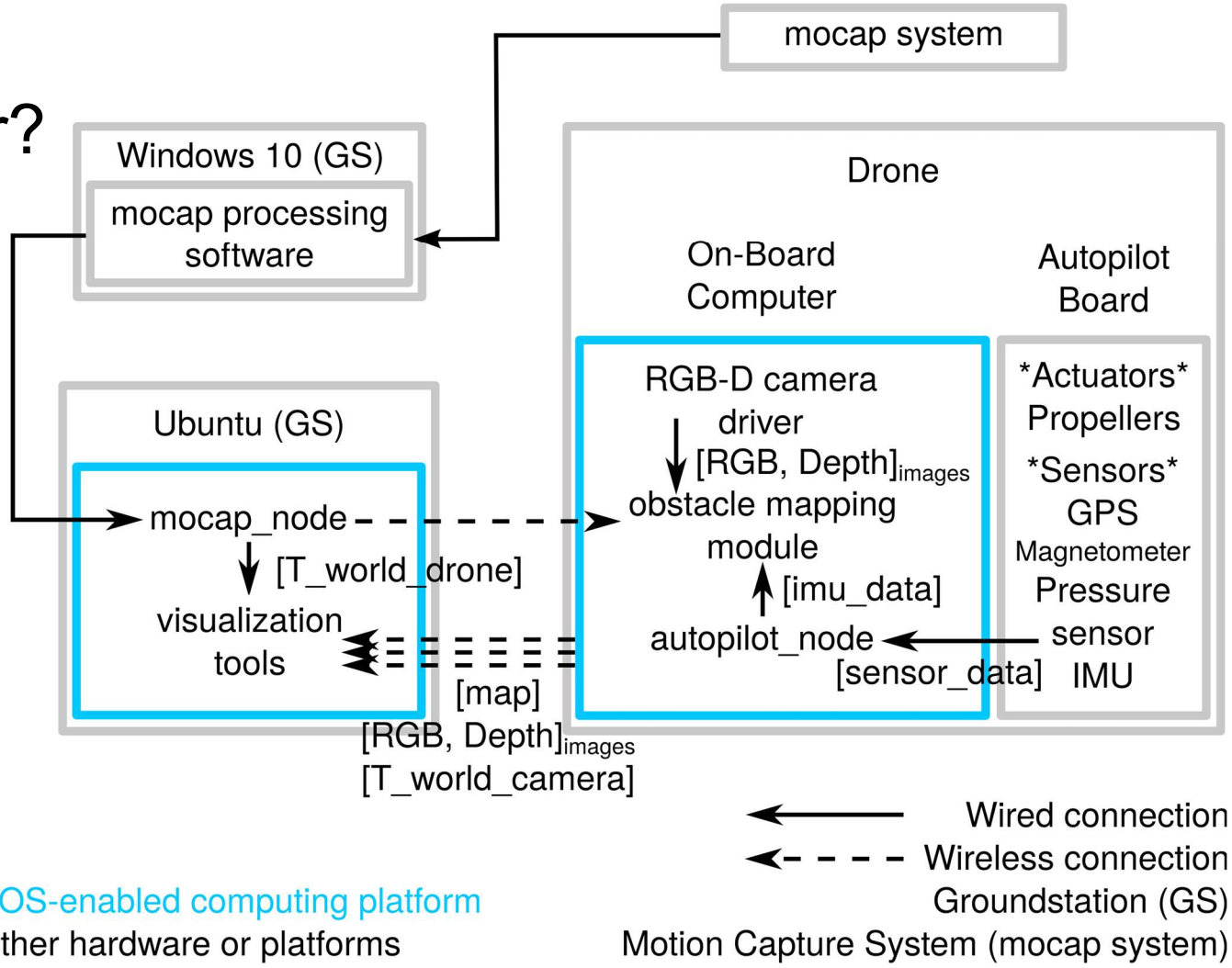
- The 3 Ds of Robotics refer to tasks performed by a robot
- Search and Rescue Robots
  - Dull
  - Dirty
  - Dangerous – For the integrity of the robot or a person performing the same task.

https://www.tugraz.at

Motivation – Robotics
What are robots good for? – The Business Case

- Often times a robot-based solution is cheaper
- Life support systems for humans are very heavy (and costly)
  - Submarines
  - Planes
  - Blimps
  - Spacecraft
- Cheaper solutions allow to tap new markets
  - Repetitive inspection tasks – Infrastructure monitoring
  - Data gathering – for example – Agriculture applications

Motivation – ROS
What is ROS good for?
Example of software architecture



Motivation – ROS
What is ROS good for? Multithreading

- Multithreading is **hard**
  - Debugging a multithreaded software is **very hard**
  - Some fear-inducing Multithreading jargon: shared variables, resource protection mechanisms, mutexes, semaphores, wait conditions, …

Motivation – ROS
What is ROS good for? Multithreading

- Multithreading is **hard**
  - Debugging a multithreaded software is **very hard**
  - Some fear-inducing Multithreading jargon: shared variables, resource protection mechanisms, mutexes, semaphores, wait conditions, …
  - Introductory material:
    Stanford - Lecture Collection | Programming Paradigms (CS107)
    https://youtu.be/Ps8jOj7diA0?si=Im-0xrVFT-1Vo6vN
    Lecture 14 an onwards…

Motivation – ROS
What is ROS good for? Multithreading

- ROS Solution – ROS takes care of the multithreading…
- ~~Multithreading is **hard**~~
  - ~~Debugging a multithreaded software is **very hard**~~
  - ~~Some fear-inducing Multithreading jargon: shared variables, resource protection mechanisms, mutexes, semaphores, wait conditions, …~~
- ROS Nodes
  - ROS Topics, Services, …: **inter-process communications**
- ROS Nodelets and Nodelet managers
  - Nodelet manager run 1 main process and several Nodelets, which share memory – operating similarly to a **multithreaded process**

Motivation – ROS
What is ROS good for? Inter-process communications

- Fear-inducing inter-process communications jargon:
  - Pipes, files, sockets
  - Signals, interruptions
  - Communication between computers on the same network:
    - Communications protocols: TCP-IP, UDP

Motivation – ROS
What is ROS good for? Inter-process communications

- ROS Solution – ROS takes care of the Inter-process communications
- ~~Fear-inducing inter-process communications jargon:~~
  - ~~Pipes, files, sockets~~
  - ~~Signals, interruptions~~
  - ~~Communication between computers on the same network:~~
    - ~~Communications protocols: TCP-IP, UDP~~
- ROS Topics, Services, Actions, Parameter Server, Dynamic Reconfigure
- ROS Parameter Server >> Useful to define global parameters in a single place – example, controller loop frequency / sampling time.

Motivation – ROS
What is ROS good for? Syncing clocks between computers

- Fear-inducing clock syncing jargon:
  - Clock drift, clock offset, clock strata, frequency stability, Allan variance
  - Networking protocols for clock synchronization: NTP, PTP
  - Computer networking technologies: Ethernet, WiFi, Ethernet Over Power, Bluetooth, …
  - International Atomic Time

Motivation – ROS
What is ROS good for? ~~Syncing clocks between computers~~

- ROS does not solve this, but recommends using **chrony**
- Fear-inducing clock syncing jargon:
  - Clock drift, clock offset, clock strata, frequency stability, Allan variance
  - Networking protocols for clock synchronization: NTP, PTP
  - Computer networking technologies: Ethernet, WiFi, Ethernet Over Power, Bluetooth, …
- chrony:
  - Default in Red Hat and available in Ubuntu repositories
  - Low resource consumption (cost) and supports PTP as well as NTP

Motivation – ROS
Alternative ROS Course Slides

https://rsl.ethz.ch/education-students/lectures/ros.html

ETHZ ROS Course Materials:
RSL - Robotic Systems Lab
Course:
Programming for Robotics - ROS

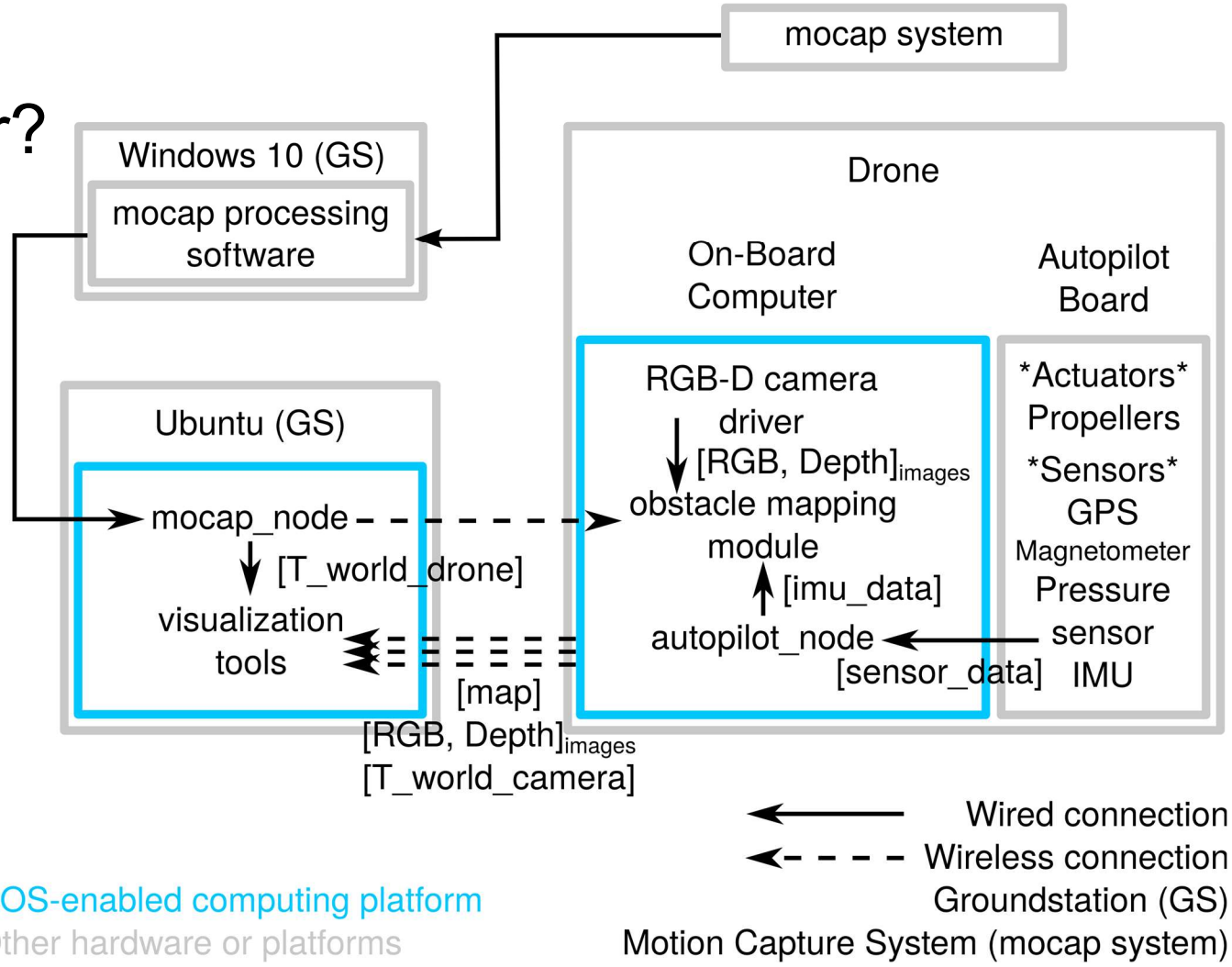
Contents

- Motivation
- **ROS at the Dronespace**

ROS Melodic – Ubuntu 18.04
(there are always turtles on these images)
http://wiki.ros.org/melodic

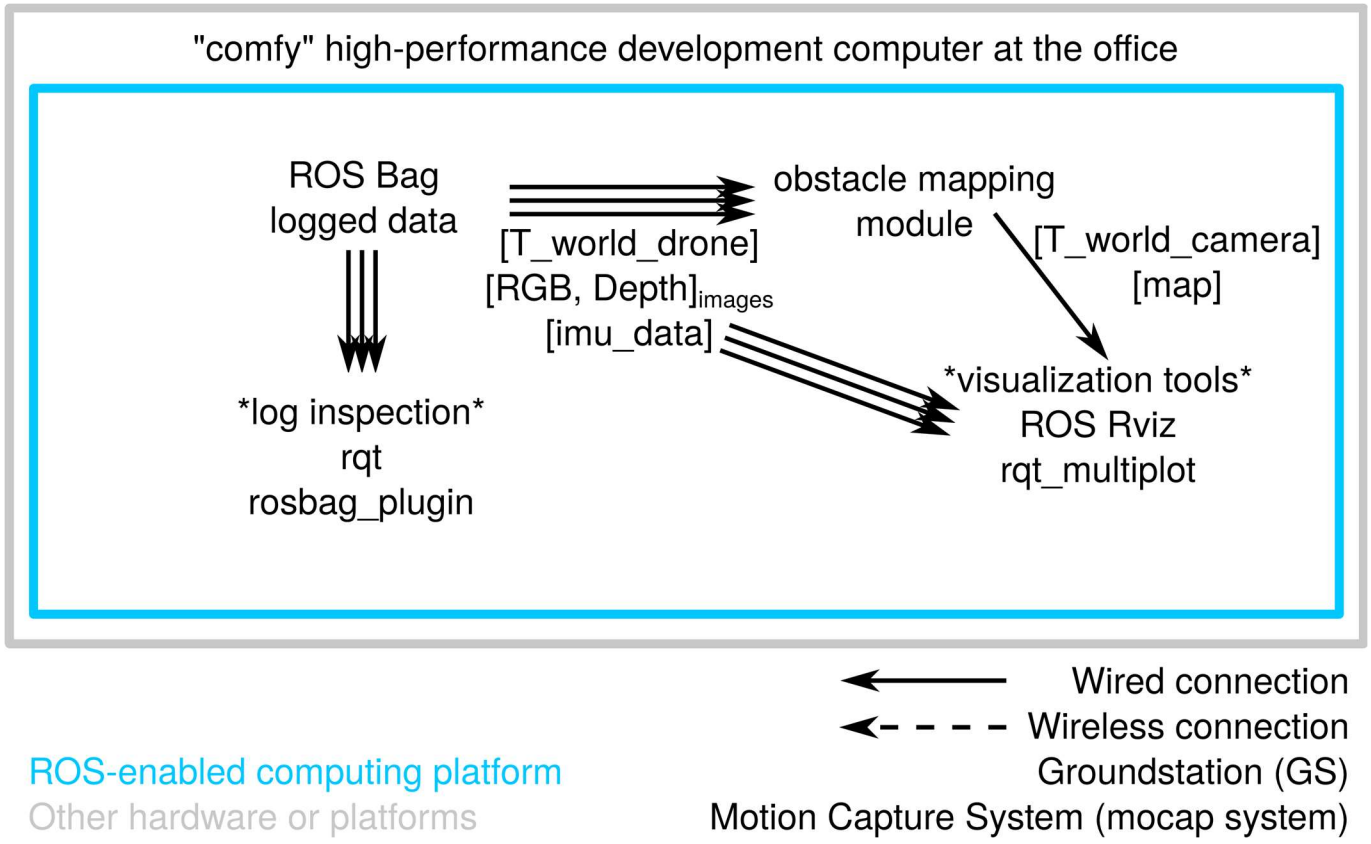
Motivation – ROS
What is ROS good for?
Prototyping applications

1. Data acquisition – ROS Bag recording



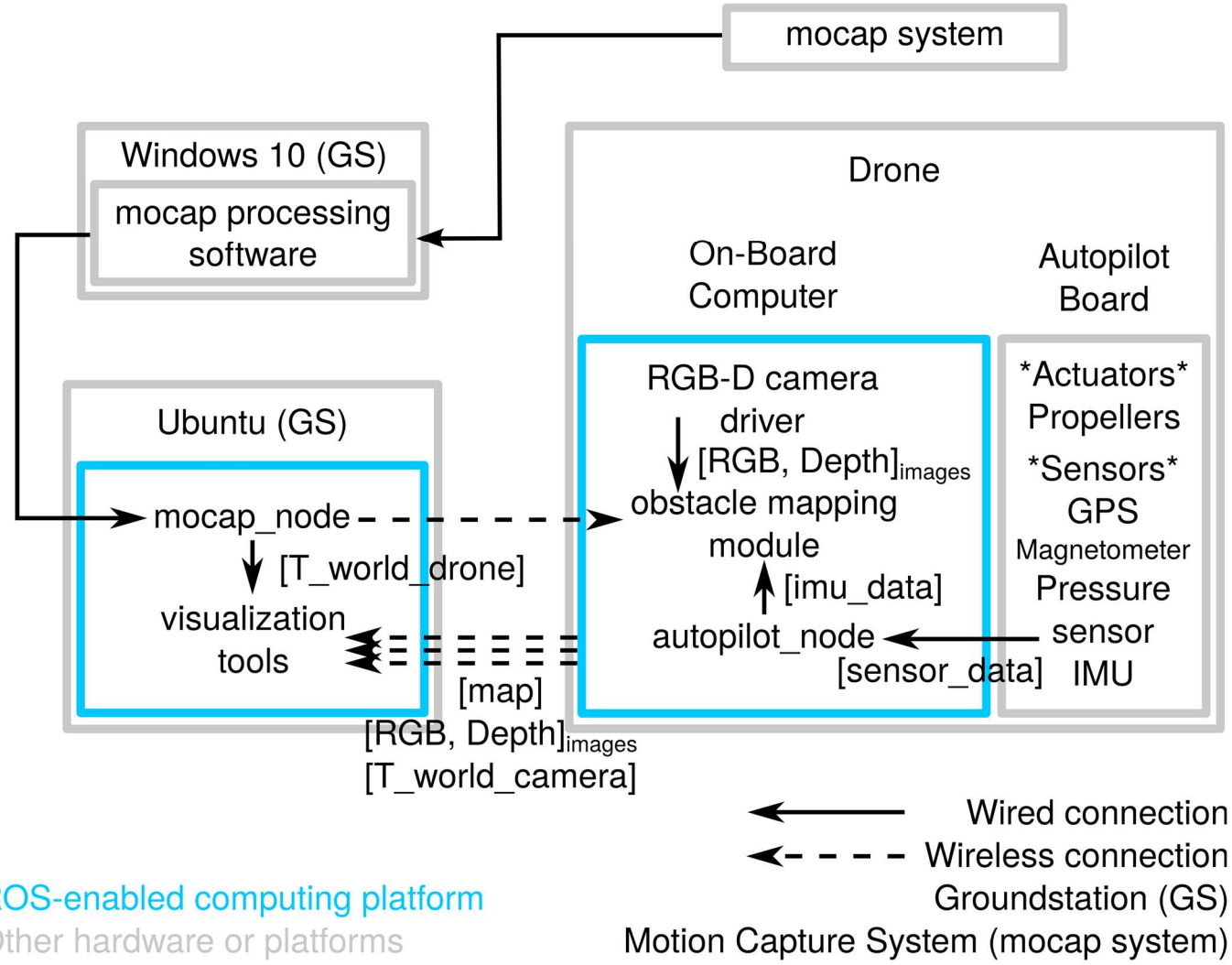
Motivation – ROS
What is ROS good for?
Prototyping applications

1. Data acquisition – ROS Bag recording
2. Development work



Motivation – ROS
What is ROS good fc
Prototyping applications

1. Data acquisition – ROS Bag recording
2. Development work
3. Deployment on real-time system



ROS Practical (II)
Camera Drones Lecture

Jesús Pestana, Friedrich Fraundorfer

Contact: jesus.pestana@pro2future.at