Data Collection, Analysis and Annotation

This script will guide you through the data collection, analysis of the recorded `.bag` files, and the annotation process with a basic introduction to the concepts and some example commands. You will need these tools during your project.

This guide is structured as follows:

1. Data Collection
    1. Ego-Perspective Setup
    2. Tabletop Setup
2. Analysis of the Recordings
3. Exporting Images from the Recordings
4. Annotation

1 Data Collection

Along the PPHAU lecture we will be encountering `Robot Operating System (ROS)` as the main tool for the communication between the computers and the sensors.

ROS is a set of software libraries and tools that were designed to build robot applications. As we are going to have a communication between a visual/motion sensor and a central computer for the data collection, it is reasonable to use ROS as our communication foundation for our systems.

A quick explanation of ROS would be that every `node` (computer, sensor, etc.) is connected through a `ROS Master` (a computer), which acts as an information center.

The nodes who want to send any type of messages (in our case: image, depth or motion data) publishes these to the ROS network. These are called `publisher nodes`.

And the nodes who want to receive (in our case the main computer that we are recording the data) these messages subscribe to these messages in real-time through the ROS Master.

These messages that are being published from the publisher nodes are organized under `topics`, and their naming structure is similar to a general folder sturucture. A few examples of the topics in our case would be:

- `.../color/image_raw/compressed`
- `.../camera/gyro/sample`

For more detailed information about ROS, you can check the official ROS documentation.

1.1 Ego-Perspective Setup

1. Ego-Perspective: Setup Overview

In the ego-perspective example, we are using `Ubuntu 20.04` and `ROS Noetic` to record data with a `D435i` camera and we want to record the following topics:

1. Compressed color images and camera intrinsics:
    - `.../color/camera_info`
    - `.../color/image_raw/compressed`
2. Raw aligned depth to color:
    - `.../aligned_depth_to_color/camera_info`
    - `.../aligned_depth_to_color/image_raw`
3. IMU information (gyroscope and accelerometer)
    - `.../accel/imu_info`
    - `.../camera/accel/sample`
    - `.../camera/gyro/imu_info`
    - `.../camera/gyro/sample`

2. Ego-Perspective: Launching the ROS Publisher Node

First we need to start publishing the messages from the publisher nodes to be able to record them.

- Start the ROS on Master:

```
roscore
```

- Launch the publisher node using the `realsense2_camera` package with the `rs_camera.launch` launch file:

```
roslaunch realsense2_camera rs_camera.launch
```

- If we want to override the default argument values in `rs_camera.launch` file , we can pass the argument values that we want to change as part of the command:

```
roslaunch realsense2_camera rs_camera.launch align_depth:=true
depth_fps:=30 color_width:=640 color_height:=480 color_fps:=30
enable_color:=true enable_infra:=true enable_infra1:=true
enable_pointcloud:=false enable_gyro:=true enable_accel:=true
filters:=spatial,temporal
```

Here we have enabled the color, depth, infrared, gyro, and accelerometer modalities. We also apply spatial and temporal filtering on the depth modality as a post-processing step.

You can check the default values for the launch file here.

3. Ego-Perspective: Topic Monitoring and Visualization

After publishing the messages, we can monitor the topics in ROS with a few easy commands.

- We can list all available topics with:

```
rostopic list
```

This command should output something like this:

```
pphau@tueilmt-tablews:~$ rostopic list
/camera/101622073015/aligned_depth_to_color/camera_info
/camera/101622073015/aligned_depth_to_color/image_raw
/camera/101622073015/color/camera_info
/camera/101622073015/color/image_raw/compressed
/clock
/rosout
/rosout_agg
```

- We can monitor the frequency (published message per second) of the selected topics with `rostopic hz`:

```
rostopic hz /camera/aligned_depth_to_color/image_raw
/camera/color/image_raw /camera/gyro/sample /camera/accel/sample
```

This command should output something like this:

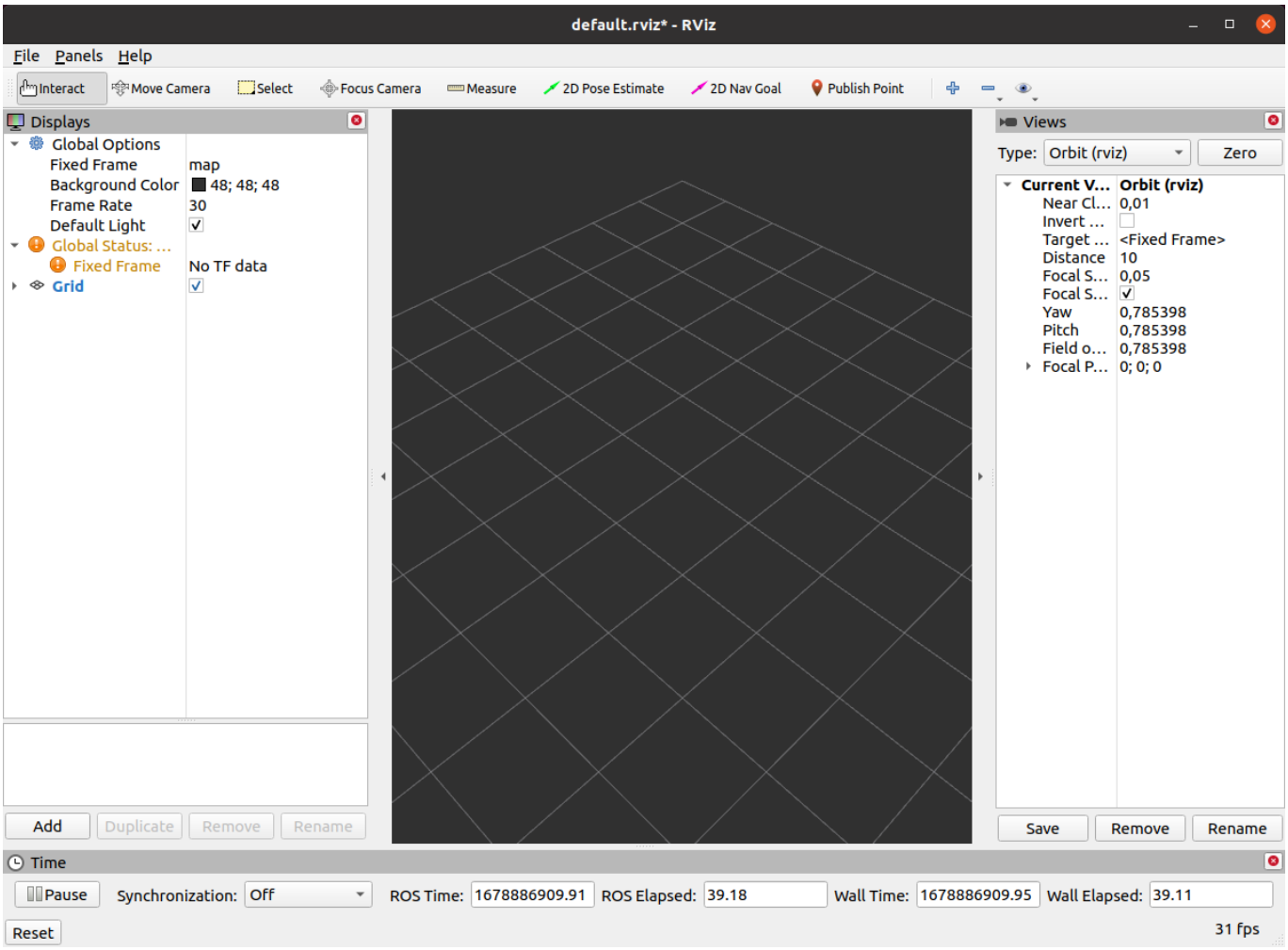
```
pphau@tueilmt-tablews:~$ rostopic hz $(rostopic list | grep "/camera/")
subscribed to [/camera/101622073015/aligned_depth_to_color/camera_info]
subscribed to [/camera/101622073015/aligned_depth_to_color/image_raw]
subscribed to [/camera/101622073015/color/camera_info]
subscribed to [/camera/101622073015/color/image_raw/compressed]
                        topic                                rate   min_delta   max_delta    std_dev    window
=============================================================================================================
/camera/101622073015/aligned_depth_to_color/camera_info   29.98   0.03065     0.03619     0.001061    30
/camera/101622073015/aligned_depth_to_color/image_raw     29.96   0.03016     0.03618     0.001315    30
/camera/101622073015/color/camera_info                    29.97   0.0307      0.03588     0.0009683   30
/camera/101622073015/color/image_raw/compressed           29.98   0.02302     0.04331     0.003334    30

                        topic                                rate   min_delta   max_delta   std_dev    window
=============================================================================================================
/camera/101622073015/aligned_depth_to_color/camera_info   30.02   0.02882     0.03808     0.001295   60
/camera/101622073015/aligned_depth_to_color/image_raw     29.94   0.03016     0.03668     0.001245   60
/camera/101622073015/color/camera_info                    30.0    0.02981     0.03809     0.001325   60
/camera/101622073015/color/image_raw/compressed           29.93   0.000123    0.04455     0.005534   60
```

Additional to the monitoring the topics and message frequency, we can visualize the messages using the GUI "RViz".

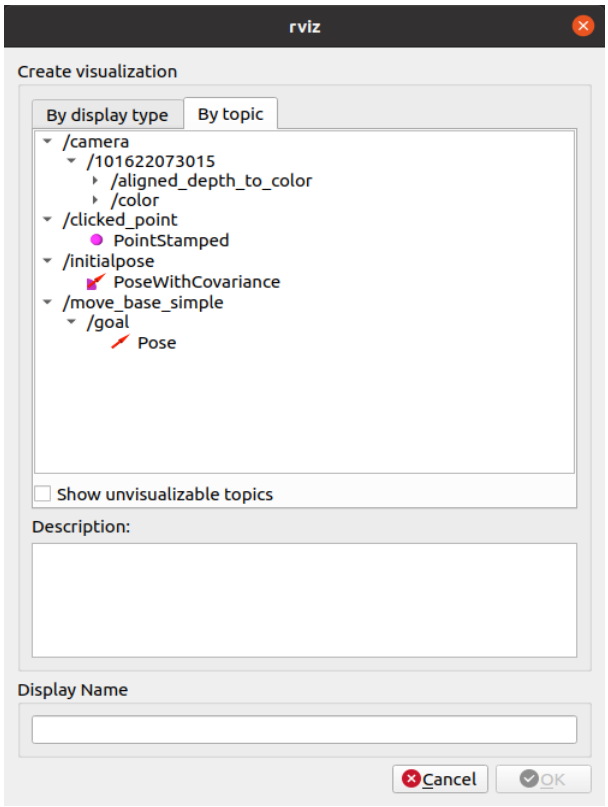
```
rviz
```

Please refer to the RViz User's Guide for a detailed explanation on how to use the GUI.

This command will open the following GUI.



If we click on the Add button on the left bottom, this would open an additional window:



On this window we can select the topics that we want to display on RViz.

4. Ego-Perspective: Recording of the Bag Files

- We can record any of the topics that are being published to the ROS network using the `rosbag` package:

```
rosbag record /camera/aligned_depth_to_color/image_raw
/camera/aligned_depth_to_color/camera_info
/camera/color/image_raw/compressed /camera/color/camera_info
/camera/gyro/imu_info /camera/gyro/sample  /camera/accel/imu_info
/camera/accel/sample /tf /tf_static -o ego-recording.bag
```

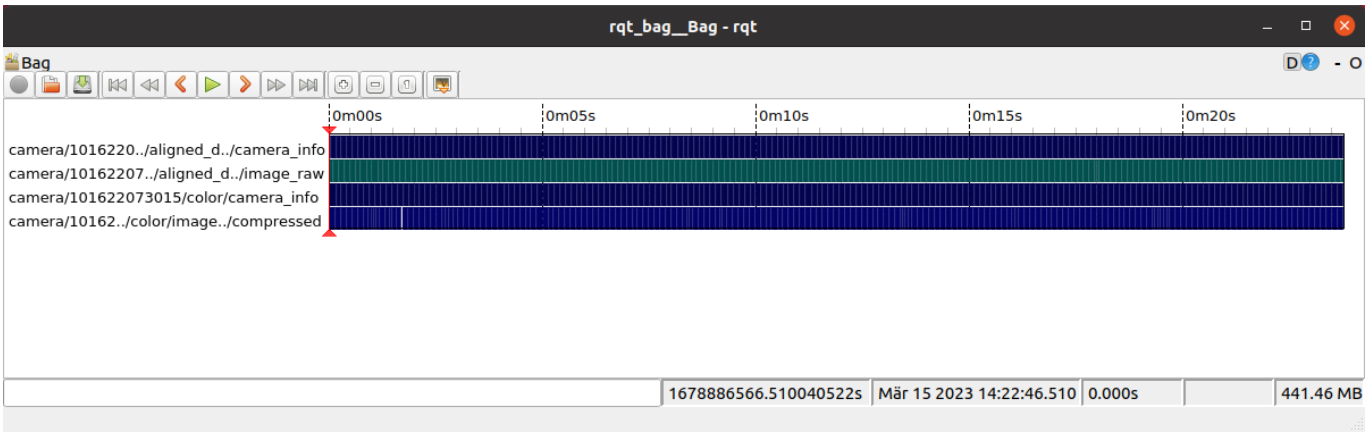
- We can also check the bag contents after recording with

```
rosbag info <bag_name>
```

```
pphau@tueilmt-tablews:~/Desktop/recording$ rosbag info one_camera.bag
path:        one_camera.bag
version:     2.0
duration:    23.8s
start:       Mar 15 2023 14:22:46.51 (1678886566.51)
end:         Mar 15 2023 14:23:10.29 (1678886590.29)
size:        441.5 MB
messages:    2853
compression: none [357/357 chunks]
types:       sensor_msgs/CameraInfo      [c9a58c1b0b154e0e6da7578cb991d214]
             sensor_msgs/CompressedImage [8f7a12909da2c9d3332d540a0977563f]
             sensor_msgs/Image           [060021388200f6f0f447d0fcd9c64743]
topics:      /camera/101622073015/aligned_depth_to_color/camera_info   714 msgs    : sensor_msgs/CameraInfo
             /camera/101622073015/aligned_depth_to_color/image_raw     713 msgs    : sensor_msgs/Image
             /camera/101622073015/color/camera_info                    713 msgs    : sensor_msgs/CameraInfo
             /camera/101622073015/color/image_raw/compressed           713 msgs    : sensor_msgs/CompressedImage
```

- or with

```
rqt_bag <bag_name>
```



1.2 Tabletop Setup

1. Tabletop: Setup Overview

In the case of the tabletop setup we are using 6 `D435` and 2 `D435i` cameras. However, the IMU sensors here are not useful since the cameras are stationary.

We are using two `OptiPlex 7040 micro` PCs and main CPU tower all with `Ubuntu 20.04` to support 8 cameras with compressed color and uncompressed aligned depth.

The topics we are interested in are:

1. Compressed color images and camera intrinsics:
    - `.../color/camera_info`
    - `.../color/image_raw/compressed`
2. Raw aligned depth to color:
    - `.../aligned_depth_to_color/camera_info`
    - `.../aligned_depth_to_color/image_raw`

2. Tabletop: Launching the ROS Publisher Node

As in the ego-perspective case, first we need to start the core and start publishing the messages from the publisher nodes to be able to record them.

- Start the ROS on Master:

```
roscore
```

- We can launch the `realsense2_camera` for the multiple camera setup using the launch setup `rs_multiple_devices.launch` and pass the camera serials we want to publish.
- As we are going to be working with 8 cameras it might be hard to add all of their serial numbers to the command. For convenience we have modified the `rs_multiple_devices.launch` and created a copy named `rs_multiple_devices_setup_all.launch` which contains all the necessary info about the cameras we are going to be using. As a result, the launch command would be as follows:

```
roslaunch rs_multiple_devices_setup_all.launch align_depth:=true
depth_width:=640 depth_height:=480 depth_fps:=30 color_width:=640
color_height:=480 color_fps:=30 enable_color:=true enable_infra:=false
enable_infra1:=false enable_pointcloud:=false enable_accel:=false
enable_gyro:=false filters:=spatial,temporal
```

Here we only enable compressed color and raw aligned depth images, as the bandwidth grows larger with the number of cameras we have.

- Check info about the default setup for rs_multiple_devices.launch

3. Tabletop: Topic Monitoring and Visualization

- To list available topics we can use:

```
rostopic list
```

```
pphau@tueilmt-tablews:~$ rostopic list
/camera/101622073015/aligned_depth_to_color/camera_info
/camera/101622073015/aligned_depth_to_color/image_raw
/camera/101622073015/color/camera_info
/camera/101622073015/color/image_raw/compressed
/camera/135122071615/aligned_depth_to_color/camera_info
/camera/135122071615/aligned_depth_to_color/image_raw
/camera/135122071615/color/camera_info
/camera/135122071615/color/image_raw/compressed
/camera/137322071489/aligned_depth_to_color/camera_info
/camera/137322071489/aligned_depth_to_color/image_raw
/camera/137322071489/color/camera_info
/camera/137322071489/color/image_raw/compressed
/camera/138422075645/aligned_depth_to_color/camera_info
/camera/138422075645/aligned_depth_to_color/image_raw
/camera/138422075645/color/camera_info
/camera/138422075645/color/image_raw/compressed
/camera/141722071427/aligned_depth_to_color/camera_info
/camera/141722071427/aligned_depth_to_color/image_raw
/camera/141722071427/color/camera_info
/camera/141722071427/color/image_raw/compressed
/clock
/rosout
/rosout_agg
/tf_static
```

- To monitor the frequency of the messages we can use `rostopic hz`. For example:

```
rostopic hz $(rostopic list | grep
"aligned_depth_to_color/image_raw$\|/color/image_raw$\|/color/camera_info$
\|/aligned_depth_to_color/camera_info$")
```

```
pphau@tueilmt-tablews:~$ rostopic hz $(rostopic list | grep "/camera/")
subscribed to [/camera/101622073015/aligned_depth_to_color/camera_info]
subscribed to [/camera/101622073015/aligned_depth_to_color/image_raw]
subscribed to [/camera/101622073015/color/camera_info]
subscribed to [/camera/101622073015/color/image_raw/compressed]
subscribed to [/camera/135122071615/aligned_depth_to_color/camera_info]
subscribed to [/camera/135122071615/aligned_depth_to_color/image_raw]
subscribed to [/camera/135122071615/color/camera_info]
subscribed to [/camera/135122071615/color/image_raw/compressed]
subscribed to [/camera/137322071489/aligned_depth_to_color/camera_info]
subscribed to [/camera/137322071489/aligned_depth_to_color/image_raw]
subscribed to [/camera/137322071489/color/camera_info]
subscribed to [/camera/137322071489/color/image_raw/compressed]
subscribed to [/camera/138422075645/aligned_depth_to_color/camera_info]
subscribed to [/camera/138422075645/aligned_depth_to_color/image_raw]
subscribed to [/camera/138422075645/color/camera_info]
subscribed to [/camera/138422075645/color/image_raw/compressed]
subscribed to [/camera/141722071427/aligned_depth_to_color/camera_info]
subscribed to [/camera/141722071427/aligned_depth_to_color/image_raw]
subscribed to [/camera/141722071427/color/camera_info]
subscribed to [/camera/141722071427/color/image_raw/compressed]
                        topic                                 rate   min_delta   max_delta    std_dev    window
=============================================================================================================
/camera/101622073015/aligned_depth_to_color/camera_info   30.16   0.02699     0.0384      0.001723    32
/camera/101622073015/aligned_depth_to_color/image_raw     30.04   0.02813     0.03649     0.002049    32
/camera/101622073015/color/camera_info                    30.09   0.02694     0.03586     0.001701    31
/camera/101622073015/color/image_raw/compressed           29.99   0.01559     0.04391     0.003987    31
/camera/135122071615/aligned_depth_to_color/camera_info   30.04   0.02991     0.03646     0.002549    32
/camera/135122071615/aligned_depth_to_color/image_raw     30.34   0.02103     0.03629     0.002338    32
/camera/135122071615/color/camera_info                    30.03   0.03059     0.03616     0.002097    32
/camera/135122071615/color/image_raw/compressed           30.01   0.03272     0.03363     0.0001773   32
/camera/137322071489/aligned_depth_to_color/camera_info   29.97   0.03185     0.03498     0.0009317   31
/camera/137322071489/aligned_depth_to_color/image_raw     30.01   0.03044     0.03591     0.001969    31
/camera/137322071489/color/camera_info                    29.98   0.03213     0.03453     0.0006641   32
/camera/137322071489/color/image_raw/compressed           30.03   2.909e-05   0.04457     0.007329    32
/camera/138422075645/aligned_depth_to_color/camera_info   29.97   0.03066     0.03606     0.002112    31
/camera/138422075645/aligned_depth_to_color/image_raw     29.9    0.03225     0.03581     0.0006969   31
/camera/138422075645/color/camera_info                    29.98   0.03096     0.0358      0.00175     31
/camera/138422075645/color/image_raw/compressed           29.99   0.03226     0.03417     0.0004444   31
/camera/141722071427/aligned_depth_to_color/camera_info   29.98   0.03253     0.03406     0.0002443   31
/camera/141722071427/aligned_depth_to_color/image_raw     29.98   0.02958     0.0375      0.001853    31
/camera/141722071427/color/camera_info                    30.04   0.02997     0.03463     0.0009056   31
/camera/141722071427/color/image_raw/compressed           29.98   5.531e-05   0.04401     0.007022    31
```

Here, we have a multiple camera setup, therefore, we use `$(rostopic list | grep "<<pattern>>")` to filter the topics we want.

- Again, to visualize the messages we can use

```
rviz
```

Please refer to the RViz User's Guide for a detailed explanation on how to use the GUI.

Adding topics one by one for 8 cameras on RViz would be inconvenient. Therefore we have created a configuration file which will open the GUI window with all of the RGB camera views included. If you want to add the depth views along with the RGB views you need to add them one by one. You can use the following command to use the prepared configuration file.

```
rviz -d rviz_multiview.rviz
```

4. Tabletop: Recording of the Bag Files

Similar to the Ego-Perspective setup we are using the rosbag package, however the count of topics here will be 8 times more than before. Therefore, we use `rostopic list` to list all the available topics, and `grep` with a regular expression to filter the topics we want.

So for data recording the command becomes as follows:

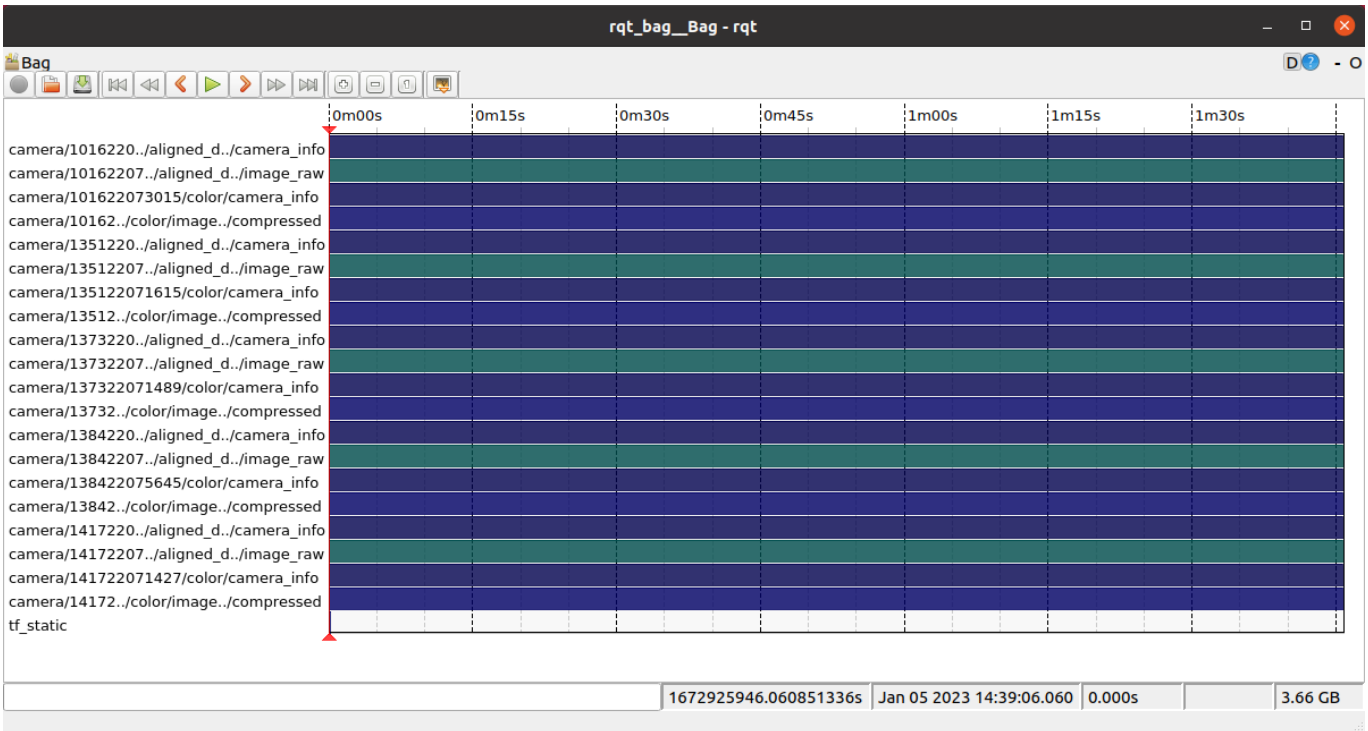
```
rosbag record $(rostopic list |
grep"aligned_depth_to_color/image_raw$\|/color/image_raw$\|/color/camera_i
nfo$\|/aligned_depth_to_color/camera_info$") -o tabletop_recording.bag --
split --size 3500 -b 0
```

- The `rostopic list | grep ...` will list all topics matching the regular expression we feed. Here, the regex checks for topics ending with any of the following strings
    - `aligned_depth_to_color/image_raw`
    - `/color/image_raw`
    - `/color/camera_info`
    - `/aligned_depth_to_color/camera_info`
- Note that we use `-b 0` to set the bag buffer to unlimited, and `--split --size 3500` to split the bags in chunks of 3500MB.
- To check the bag contents we could again use `rosbag info <bag_name>` or `rqt/rqt_bag <bag_name>`

`rosbag info` output in a multiple camera setup:

```
pphau@tueilmt-tablews:~/Desktop/pphau/recordings$ rosbag info drill_1_2023-01-05-14-39-05.bag
path:        drill_1_2023-01-05-14-39-05.bag
version:     2.0
duration:    1:45s (105s)
start:       Jan 05 2023 14:39:06.06 (1672925946.06)
end:         Jan 05 2023 14:40:51.84 (1672926051.84)
size:        3.7 GB
messages:    63433
compression: lz4 [7937/7937 chunks; 38.19%]
uncompressed: 9.6 GB @ 92.6 MB/s
compressed:  3.7 GB @ 35.4 MB/s (38.19%)
types:       sensor_msgs/CameraInfo      [c9a58c1b0b154e0e6da7578cb991d214]
             sensor_msgs/CompressedImage [8f7a12909da2c9d3332d540a0977563f]
             sensor_msgs/Image           [060021388200f6f0f447d0fcd9c64743]
             tf2_msgs/TFMessage          [94810edda583a504dfda3829e70d7eec]
topics:      /camera/101622073015/aligned_depth_to_color/camera_info   3171 msgs    : sensor_msgs/CameraInfo
             /camera/101622073015/aligned_depth_to_color/image_raw     3171 msgs    : sensor_msgs/Image
             /camera/101622073015/color/camera_info                    3171 msgs    : sensor_msgs/CameraInfo
             /camera/101622073015/color/image_raw/compressed           3171 msgs    : sensor_msgs/CompressedImage
             /camera/135122071615/aligned_depth_to_color/camera_info   3172 msgs    : sensor_msgs/CameraInfo
             /camera/135122071615/aligned_depth_to_color/image_raw     3172 msgs    : sensor_msgs/Image
             /camera/135122071615/color/camera_info                    3172 msgs    : sensor_msgs/CameraInfo
             /camera/135122071615/color/image_raw/compressed           3172 msgs    : sensor_msgs/CompressedImage
             /camera/137322071489/aligned_depth_to_color/camera_info   3172 msgs    : sensor_msgs/CameraInfo
             /camera/137322071489/aligned_depth_to_color/image_raw     3171 msgs    : sensor_msgs/Image
             /camera/137322071489/color/camera_info                    3172 msgs    : sensor_msgs/CameraInfo
             /camera/137322071489/color/image_raw/compressed           3171 msgs    : sensor_msgs/CompressedImage
             /camera/138422075645/aligned_depth_to_color/camera_info   3172 msgs    : sensor_msgs/CameraInfo
             /camera/138422075645/aligned_depth_to_color/image_raw     3171 msgs    : sensor_msgs/Image
             /camera/138422075645/color/camera_info                    3172 msgs    : sensor_msgs/CameraInfo
             /camera/138422075645/color/image_raw/compressed           3171 msgs    : sensor_msgs/CompressedImage
             /camera/141722071427/aligned_depth_to_color/camera_info   3171 msgs    : sensor_msgs/CameraInfo
             /camera/141722071427/aligned_depth_to_color/image_raw     3171 msgs    : sensor_msgs/Image
             /camera/141722071427/color/camera_info                    3171 msgs    : sensor_msgs/CameraInfo
             /camera/141722071427/color/image_raw/compressed           3171 msgs    : sensor_msgs/CompressedImage
             /tf_static                                                   5 msgs    : tf2_msgs/TFMessage          (5 con
nections)
```

`rqt_bag` output in a multiple camera setup:



2 Analysis of the Recordings

After recording the data, we might need to analyze the bagfiles according to our needs. This chapter will give you a brief introduction to the tools that you can use to get information about the bagfile or visualize the images and extract the image or depth data from the bagfiles.

Rosbag Package

This ros package includes a set of tools to record and playback the recorded bags. For example if our bagfile is named `recording.bag`:

- Summary about the messages and topics in a bag:

```
rosbag info recording.bag
```

- Playback the recorded bag:

```
rosbag play -l recording.bag
```

- `-l` flag here for playing the bagfile in a loop so that it will play indefinitely. We can stop playing the bagfile with keyboard interrupt `Ctrl + C`. Or we can also remove `-l` to play the bagfile only once.

An important note here is that a playback of a recorded bag is fundamentally the same with message publishing. When we play a rosbag it publishes the recorded messages to the system. Therefore we can analyse the bagfile with the commands or visualization interfaces that we have learned, such as `rostopic hz` or `rviz`.

Check the rosbag documentation for more.

Exporting Images From Bag Files Using image_view Package

We can extract the images from the bagfiles using the `image_view` package. With tis we will be preparing our data for the annotation proess.

RGB and depth image extraction process:

1. First we need to start the `image_saver` process form the `image_view` package.
2. Then we should check the name of the topic we want to extract using `rosbag info`.
3. After selecting the topic that we want to extract we start the `image_saver` process with the following commands:

Command for RGB topics:

```
rosrun image_view image_saver image:=<rgb_topic> _save_all_image:=all
_filename_format:=export/color%04d.%s
```

Command for depth topics:

```
rosrun image_view image_saver image:=<depth_topic> _encoding:=16UC1
_save_all_image:=all _file
```

- An example RGB image topic name can be: `/camera/137322071445/color/image_raw`
- An example depth image topic name can be: `/camera/137322071445/aligned_depth_to_color/image_raw`

4. Lastly you need to playback the recorded bag file, so that the `image_saver` process can read the published messages and extract them. (Do not loop or the process will save images until you stop the playback.)

```
rosbag play recording.bag
```

Exporting Images Using Python

If you want more control on your processing pipeline you could check rosbag cookbook.

You could also modify the `visualizer.py` code to save data instead of visualization.

Visualization

- You could playback the bag file and use `rviz` to visualize your color and depth topics as `Image`.
- You could also use the `visualizer.py` code.

3 Data Annotation

CVAT

An open source annotation tool which allows semi automatic image annotation.

MiVOS

Mivos is an interactive video segmentation tool which allows fast segmentation of images.

Please check this demo to see how to use MiVOS.

In summary you open the MiVOS GUI loaded with the image segment you want to annotate, and then segment the different objects by right-clicking for a positive point and left-clicking for a negative point. A group of positive and negative clicks should lead the segmentation mask to converge onto the ground-truth segmentation of the object.

Note: The version offered on the lab PCs provides integration with fiftyone api to export annotations in COCO format and visualize them.

Ontologies

This topic is not covered during the session. We could represent our knowledge about the system we have using an Ontology which defines properties and relationships for/between objects. You could use Protege to create your own knowledge base and access it in python using libraries like owlready2.