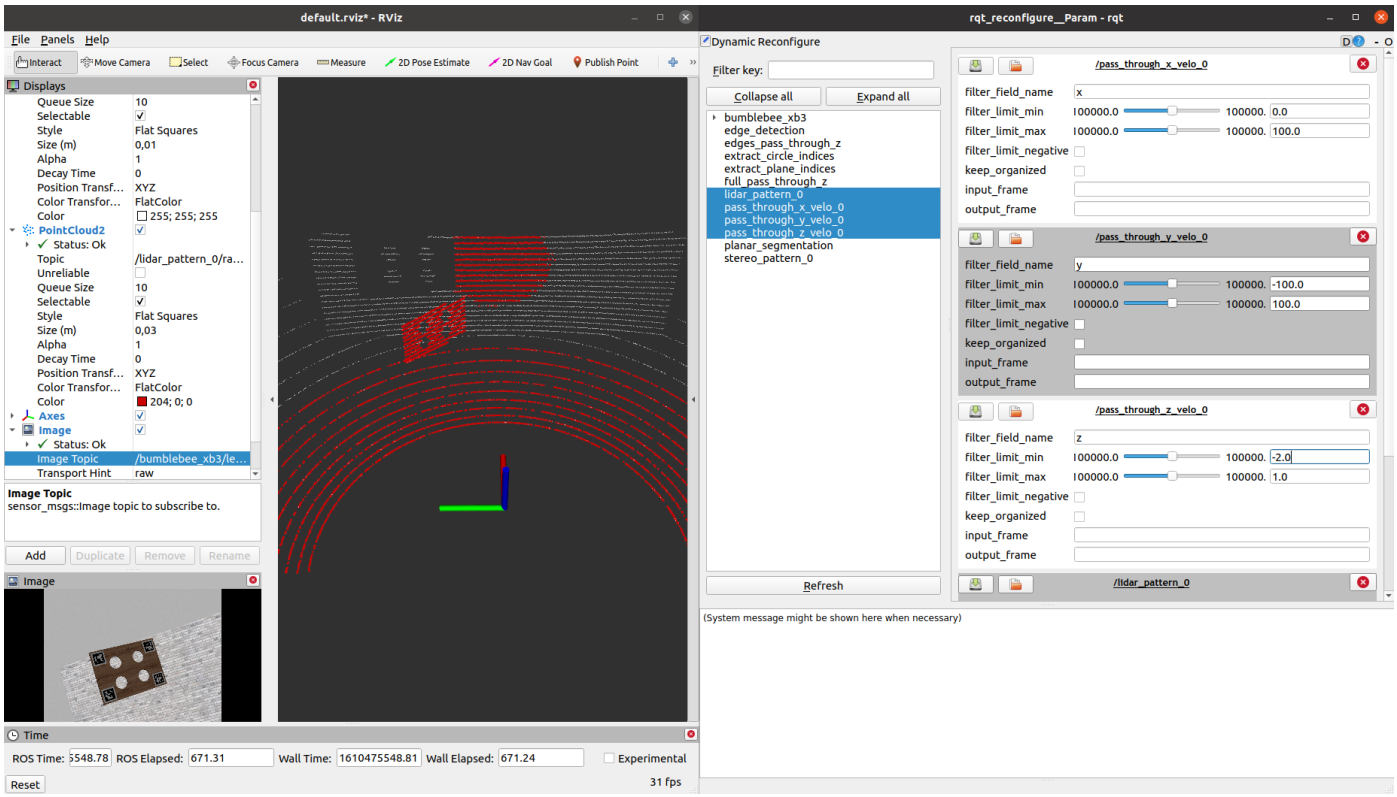
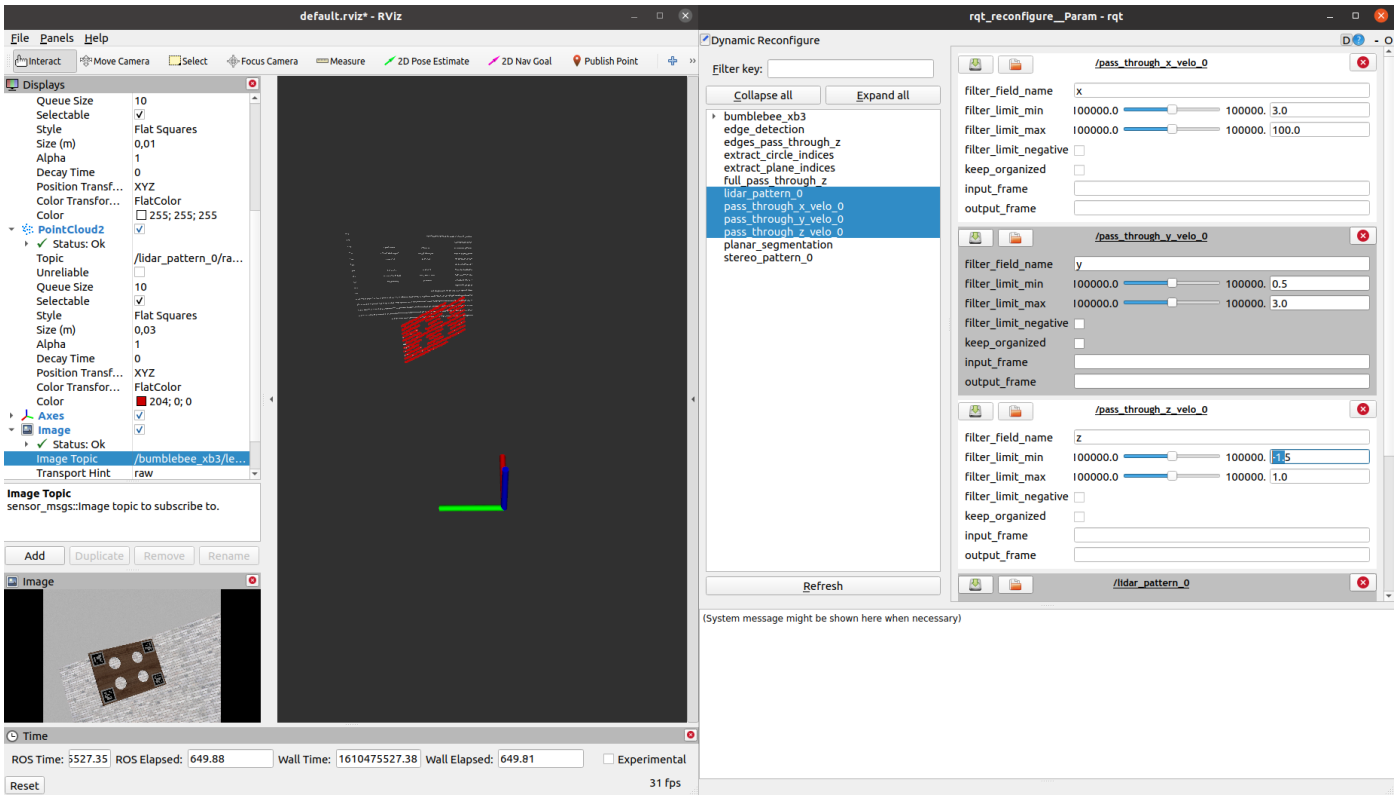
velo2cam Setup

Target and Environment Requirements

- The environment (including target) needs to be static for at least 30 frames
- A flat surface should be present behind the target (close enough that the Lidar points going through the target fall on the flat surface behind the target)

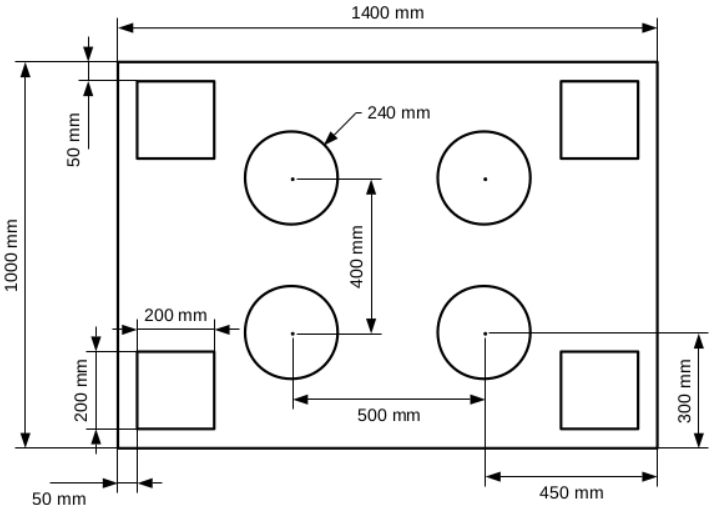


- This will then be filtered to select only the target Lidar points and the background



Hence, it's better to avoid clutter around the target to allow for simple filtering

- The repository suggests a **calibration target of 1400mm x 1000mm (1.4m x 1m)**
- However, We will be trying to do the same with a smaller target and accordingly will need to adjust some configurations



Rosbag Collection Instructions

1. It is recommended to try 3 different position/orientation configuration of the target
2. Visualize the pointcloud in rviz and try to filter the pointcloud using the rqt_reconfigure toolto get the required data

3. run rqt_image_view to check if target is in camera's FOV

Adjusting Monocular parameters to calibrate using smaller target

The parameters changes will need to be made in the
`/cfg/Lidar.cfg` and `/cfg/Monocular.cfg`

Existing Monocular.cfg

```
gen = ParameterGenerator()

gen.add("marker_size", double_t, 0, "Size of the marker (m)", 0.2, 0.1, 0.5)
gen.add("delta_width_qr_center", double_t, 0, "width increment from target center to qr
gen.add("delta_height_qr_center", double_t, 0, "height increment from target center to c
```

I'm assuming the above numbers mean the following:

- 0.2 = size of QR codes (Looks more like Aruco Tags)
- 0.1 = height of board
- 0.5 = horizontal distance between the circles

Hence, in our case these should be exactly 0.5*original dimensions (for smaller scale)

The second line `0.55, 0, 1` would mean the following:

- o.55 = distance from target center to QR center (0.5 * 0.55 in our case)

The third line `0.35, 0, 1` would mean the following:

- 0.35 = distance from target center to QR center

In both cases the last two params '0,1' should not be touched

Adjusting Lidar parameters to calibrate using smaller target

Existing Lidar.cfg

```
gen = ParameterGenerator()

gen.add("x", double_t, 0, "x coord", 0, 0, 1)
gen.add("y", double_t, 0, "y coord", 0, 0, 1)
gen.add("z", double_t, 0, "z coord", 1, 0, 1)
gen.add("angle_threshold", double_t, 0, "Angle threshold for plane segmentation", 0.55,
gen.add("circle_radius", double_t, 0, "Radius of pattern's circles", 0.12, 0, 1)
gen.add("passthrough_radius_min", double_t, 0, "Min radius for passthrough", 1.0, 0, 10)
gen.add("passthrough_radius_max", double_t, 0, "Max radius for passthrough", 6.0, 0, 10)
gen.add("centroid_distance_min", double_t, 0, "Min distance to the centroid", 0.15, 0.0,
gen.add("centroid_distance_max", double_t, 0, "Max distance to the centroid", 0.8, 0.0,
```

Only the "Radius of pattern's circles" should be changed. Since our scale in 0.5x, radius=0.06

Ros Topics and Msg Values

The following topics are required for calibrating Monocular camera and Lidar:

- **/camera/image_raw** (of type sensor_msgs/Image)
- **/camera/camera_info** (of type sensor_msgs/CameraInfo) and must have distortion params
- The frame_id of the image topic and camera_info topic must be the same
- As seen above, the **image topic and camera info topic must have the same namespace** (/camera/ here)
- **/vlp16_points** (of type sensor_msgs/PointCloud2)

Running the Calibration Launch Files

As per the above topic names, the launch files can be launched in the following manner:

The frame name can be found by running the following command:

```
rostopic echo -n 1 --noarr /camera/image_raw
rostopic echo /camera/camera_info


roslaunch velo2cam_calibration mono_pattern.launch camera_name:=/camera image_topic:=ima

roslaunch velo2cam_calibration lidar_pattern.launch cloud_topic:=/vlp16_points

roslaunch velo2cam_calibration registration.launch sensor1_type:=mono sensor2_type:=lida
```

When the registration.launch file is run there is a two step verification process which will run on the terminal.

Testing Rosbag before running through pipeline

NOTE: We may have to launch the above nodes as well (TBD)

This can be done by analyzing the outputs of the Lidar and camera images by running the following:

- rviz (for lidar pointclouds)
- rosrun rqt_reconfigure rqt_reconfigure (to filter the pointcloud)
- rqt_image_view (to check if target is in FOV clearly)

The cloud can be filtered through the parameters **filter_limit_min** and **filter_limit_max** of the **pass_through_x_velo_**, **pass_through_y_velo_**, and **pass_through_z_velo_** nodes.

At the end of the calibration, the program will ask the user if a new pattern pose is needed. It is recommended to repeat the procedure at least three times, each with a different position and orientation of the calibration pattern. Each new iteration starts with the same warm-up stage described above, where the passthrough filters should be properly adjusted to the new pattern location.