Run the camera and lidar driver:

1. Camera driver: **ros2 launch ximea_driver ximea_driver.launch.py**
    - If error 45: increase the USB buffer size
      sudo tee /sys/module/usbcore/parameters/usbfs_memory_mb >/dev/null <<<0

2. Lidar driver:
ros2 launch velodyne_driver velodyne_driver_node-VLP16-launch.py
ros2 launch velodyne_pointcloud velodyne_convert_node-VLP16-launch.py

Intrinsic calibration:

1. Use **ros2 bag record -o ~/bags/<year-month-day>-intrinsic /camera/compressed** to record rosbag
2. Run the bag and camera_listener node to get the images from messages using the following commands:
    - **ros2 bag run <BagName>**
    - **ros2 run camera_listener camera_listener**
    - **ros2 run image_transport republish compressed raw --ros-args --remap in/compressed:=/camera/compressed --remap out:=/camera**
3. The images and corresponding timestamps are in the workspace folder. Put them in another folder and use Camera Calibrator App in MATLAB to do intrinsic calibration
4. Add images to the calibrator. We need at least 20 successful images. The settings should be as follows:

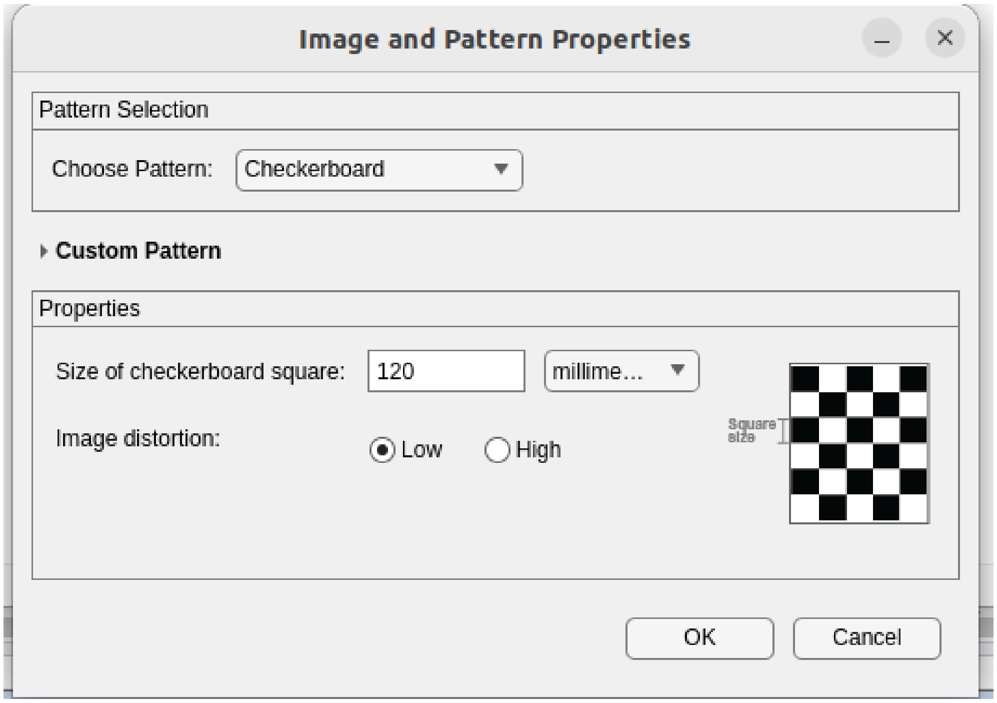


Image and Pattern Properties

Pattern Selection

Choose Pattern: Checkerboard

Custom Pattern

Properties

Size of checkerboard square: 120 millime...

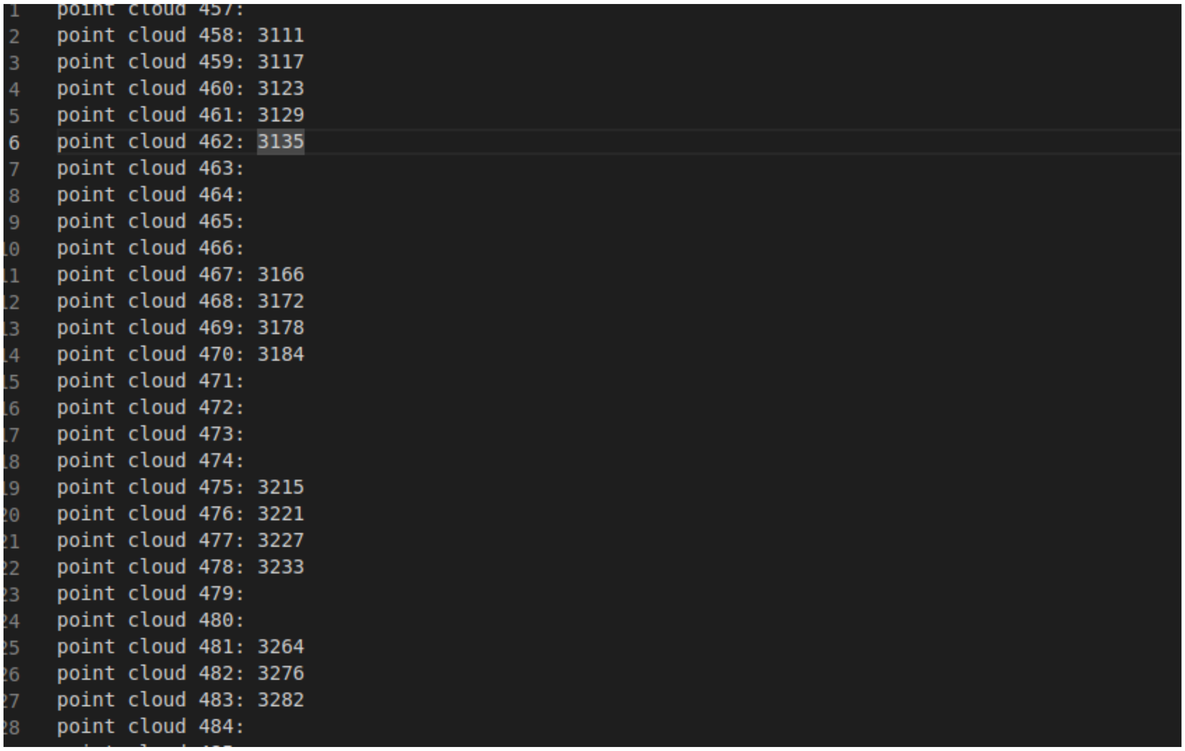
Image distortion: ● Low ○ High

5. Check the auto-detected checkerboard origin and XY axis are consistent and run the calibration. Save the intrinsic matrix for later extrinsic calibration. The error should be around 0.1 pixels.

Extrinsic Calibration:

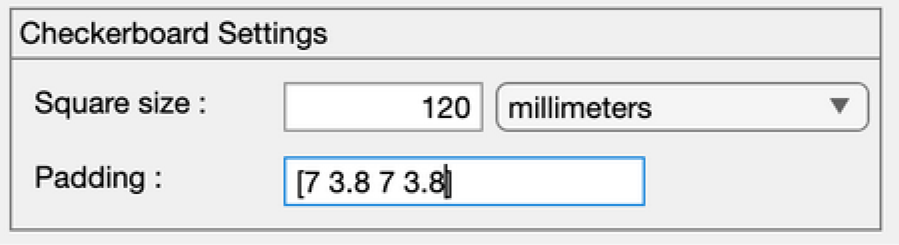
1. Use ros2 **bag record -o ~/bags/<year-month-day>-extrinsic /camera/compressed /velodyne_points** to record rosbag
2. Use the same way to get the image and use the following commands to get the lidar points
    - **ros2 bag run <BagName>**
    - **ros2 run pointcloud2_listener pointcloud2_listener**

3. Put the image data and pointcloud data into separate folders and run the **timestampMatcher.py**. In the Python script, change the **location** and **index range** of image files and pointcloud files. The threshold is set to 5000000 nanosec and can be changed in the script as well. It would output **match.txt** file that contains the matching point cloud and image indices like the following:



```
point cloud 457:
point cloud 458: 3111
point cloud 459: 3117
point cloud 460: 3123
point cloud 461: 3129
point cloud 462: 3135
point cloud 463:
point cloud 464:
point cloud 465:
point cloud 466:
point cloud 467: 3166
point cloud 468: 3172
point cloud 469: 3178
point cloud 470: 3184
point cloud 471:
point cloud 472:
point cloud 473:
point cloud 474:
point cloud 475: 3215
point cloud 476: 3221
point cloud 477: 3227
point cloud 478: 3233
point cloud 479:
point cloud 480:
point cloud 481: 3264
point cloud 482: 3276
point cloud 483: 3282
point cloud 484:
```

4. Select the desired matching sets and visualize the **min and max value of xyz positions** of the checkerboard in the pointcloud either in rviz2 or in Matlab using **scatterPlot.mlx**
5. Change the min and max values in **filter.py** and run the filter. It should output pcd files named **set_{i}.pcd**, where i is the set number. Put the pcd files into one folder and make another folder to put all the corresponding images. Don't forget to rename the images as **set_{i}.jpg**
6. Use Lidar Camera Calibrator App in MATLAB to do the extrinsic calibration, we want at least 10 sets of accepted data. The settings are as follows:

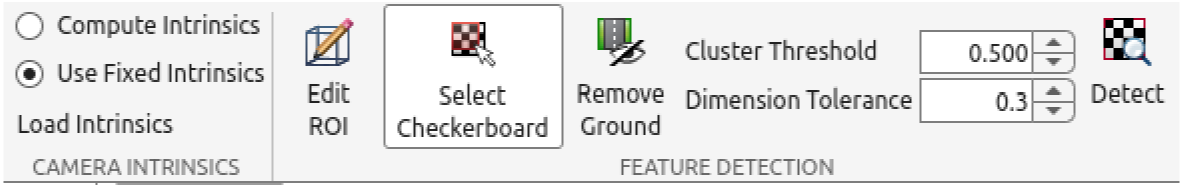


Checkerboard Settings

Square size : 120 millimeters

Padding : [7 3.8 7 3.8]

7. After importing the image and pointcloud sets, change the intrinsics to **Use Fixed Intrinsics** and import the result from intrinsic calibration. And set **Cluster Threshold** and **Dimension Tolerance** to reasonable values. The Cluster Threshold should be greater if the resolution of pointcloud data is low. Hit **Detect** once all settings are done.



○ Compute Intrinsics

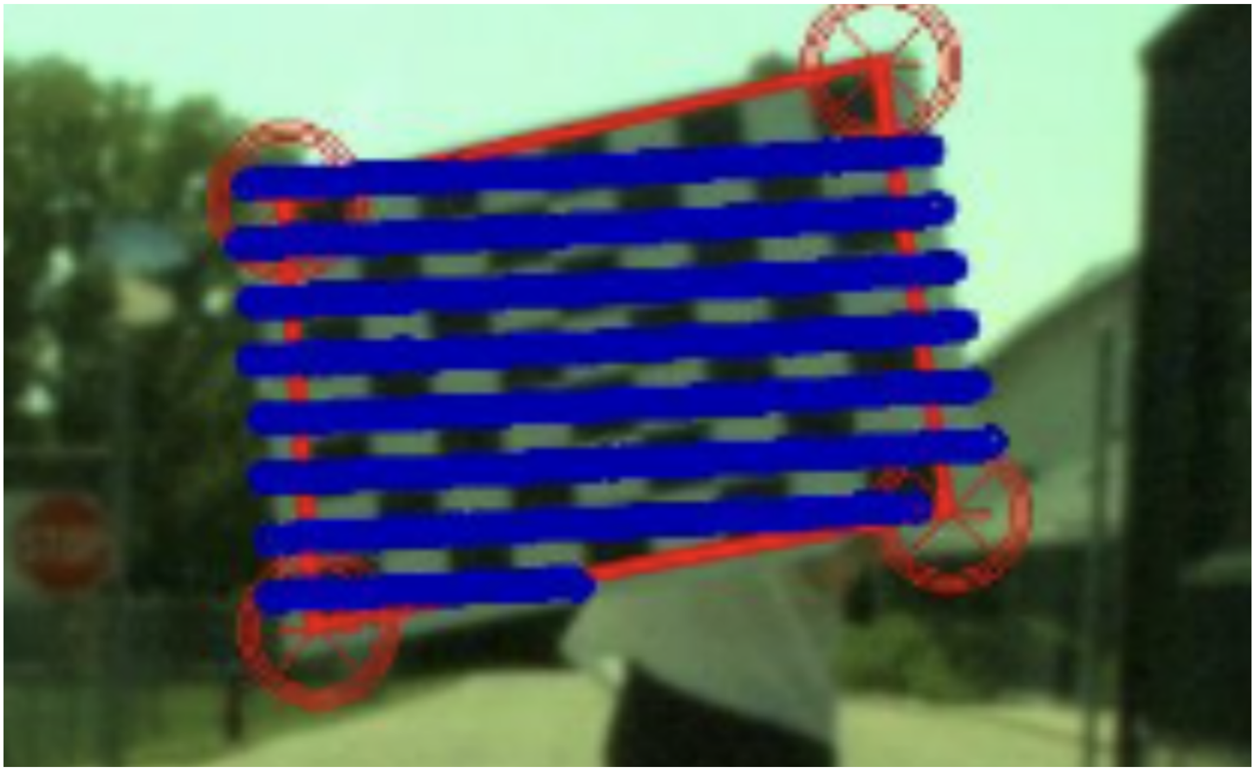
● Use Fixed Intrinsics

Cluster Threshold 0.500

Dimension Tolerance 0.3

8. If some sets are still not accepted, use **Select Checkerboard** function to manually select the points for checkerboard and redo detection.

9. Check the reprojection error and reprojected lidar points. The error should be below 5 pixels and ideally below 2-3 pixels. To decrease the error, discard the sets in which the reprojection can't cover the whole plane like the following (the top right corner is not covered in this example):



10. Import and save the tform

Use the transformation matrix:

1. The rotation matrix get from the calibrator is **camera-to-lidar** rotation matrix
2. **calibrationChecker.mlx** can be used for visualization. The algorithm can be seen in the script as well.