

# Sensors Recording System for an Autonomous Vehicle in ROS™

## Quick Start Manual

By Michael Stillman, Nov 5, 2021

Table of contents:

[Start-up Instruction](#)

[Recording](#)

[rqt\\_cat\\_sensors\\_recorder widget](#)

[How to record](#)

[Troubleshooting](#)

[Playback](#)

[playback widget](#)

[Display Switch widget](#)

[How to playback a bag file](#)

[Troubleshooting](#)

[Data extraction](#)

[General](#)

[Inertial Labs INS-DL](#)

[My attempt at getting the x-axis acceleration:](#)

[rosbag\\_to\\_csv](#)

[Velodyne LP16 Lidar / cloudpoint2 msg format](#)

[My attempt at an array creation script for lidar msgs](#)

[MISC](#)

# Start-up Instruction

In order to open the **Sensors Recording System** you'll need to open a linux shell  
(To quickly open a Terminal window, press Ctrl+Alt+T).

Next, enter:

```
roslaunch car_rec_system car_rec_system.launch
```

The 'car\_rec\_system.launch' file should launch all the needed ros nodes and then load the rqt gui environment.

Now, you should see a window similar to this:

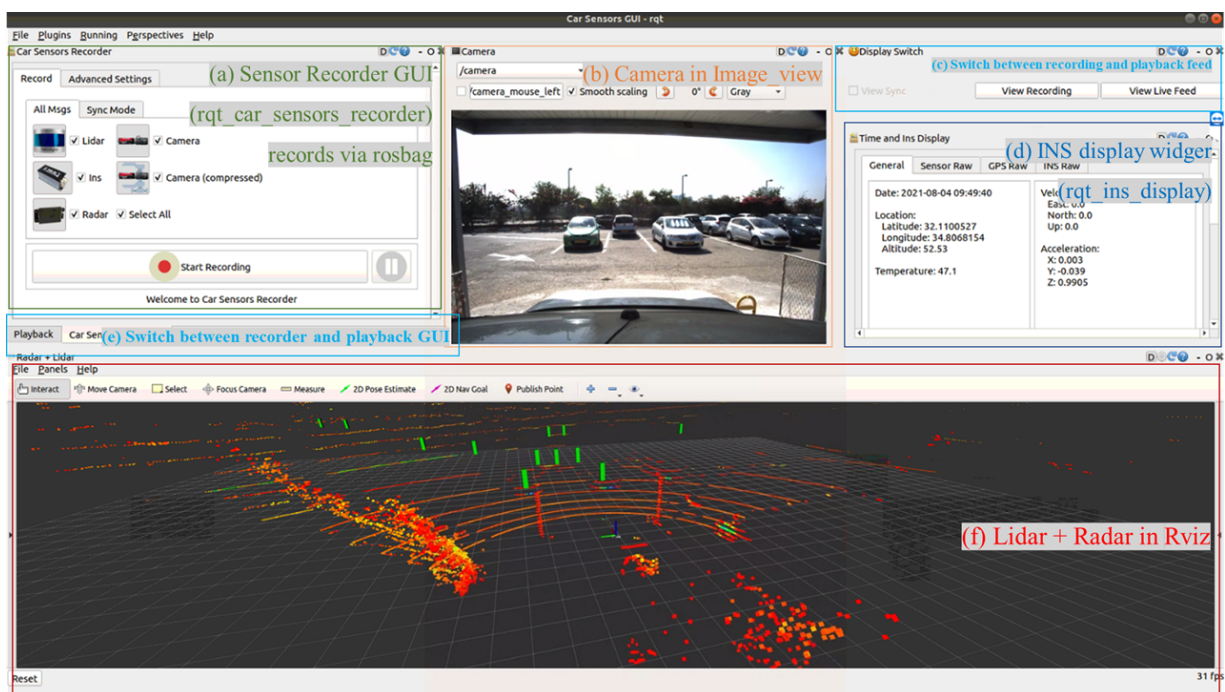


Figure 1: Sensors Recording System Gui - recording

The GUI layout is explained in the image above (Figure 1).

# Recording

## rqt\_cat\_sensors\_recorder widget

In order to record you'll have to use the rqt\_cat\_sensors\_recorder widget (See Figure 1).



Figure 2: rqt\_cat\_sensors\_recorder widget (Record tab)

Inside the 'Advanced Settings' tab you can set the settings of the recording session.

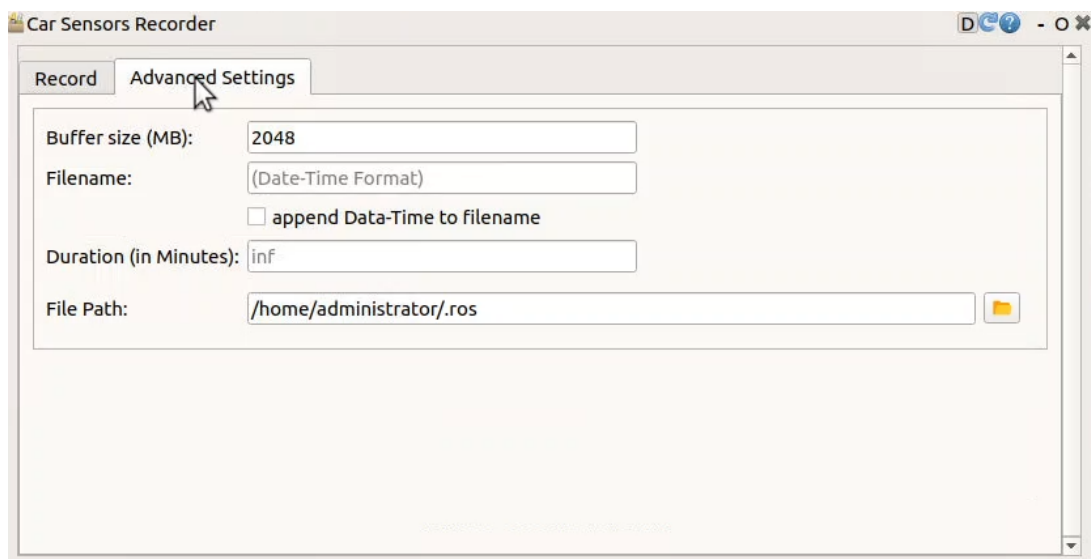


Figure 3: rqt\_cat\_sensors\_recorder widget (Advanced Settings tab)

Buffer size (MB) - sets the buffer size (in Megabytes) of the writing to a bag file; to recording all sensors in both 'Sync Mode' and 'All Msgs' modes it is highly recommended to increase the buffer to at least 4096 MB in order to avoid message (data) drops.

Filename - The file name of the recording; you can append the time and date to your given name by checking the box below.

Duration - Set the timer duration (in minutes) for the recording session. Inputs such as "1.5" are accepted, and the given example translates to 1:30 minutes of recording.

File Path - The save path for the new recording bag file. Default is '/home/administrator/.ros'. During development and testing we saved the recordings at '~/rosbag\_test'.

## How to record

First, make sure all the sensors data is correctly displayed in the live feed (similar to figure 1). Live feed is displayed first by default, but to make sure you are indeed viewing live feed, Press "View Live Feed" in the Display Switch widget ( figure 1(c)). If you just booted the Sensors Recording System, it might take a minute to start displaying all the data from all the sensors.

Select the sensors that you are interested in recording.  
(by default all the sensors are selected in the 'All Msgs' tab, and none are selected in the 'Sync Mode' tab.)

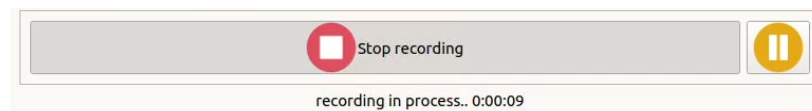
Note: Sync Mode records the data on a different prefix of ros topics, and cannot be displayed on the gui together with topics recorded in "All Msgs" mode (and vice versa).

To read more about 'Sync Mode' and 'All Msgs' mode please refer to the [project book \(in Hebrew\)](#).

After selecting the needed sensors in both modes tabs, go to the 'Advanced Settings' tab and make sure you are satisfied with current recording settings; you won't be able to change them during the recording.

Next press the "Start Recording" button at the 'Record' tab (see figure 2).

The widget should change and indicate that a recording has started.



To stop the recording press "Stop recording" or wait for the timeout if one is set in the "Duration" field (in 'Advanced Settings' tab) before the start of the recording.

# Troubleshooting

## **One of the sensors data is not showing in Live Feed display:**

If you've waited more than 2-3 minutes, and one of the sensors is not displaying data in the live feed, the simplest course of action is to turn the sensor off and on (by cutting the power) at the back of the trunk of the autonomous vehicle, and to restart the computer.

## **The gui have crashed:**

You can relaunch the gui from the shell by typing:

```
roslaunch car_rec_system gui_only.launch
```

## **Stopping the recording after the gui have crashed:**

If the gui has crashed during recording with 'Duration' parameter set to inf, you can send a SIGKILL signal to the process, using the pid number.

For example:

```
ps aux | grep rosbag
```

then copy the pid number from the line of rosbag process (the one with the sensors topics mentioned)

then run

```
kill -9 `pid_number_here`
```

pid\_number\_here should be the number you copied from the previous command output.

You can rerun

```
ps aux | grep rosbag
```

to make sure the recording has stopped.

If the 'Duration' parameter was other than 'inf' you can just wait out till the recording is over, and it will stop recording automatically.

# Playback

## playback widget

In order to switch to the playback widget you have to select the 'Playback' tab (see figure 1(e) and figure 4).

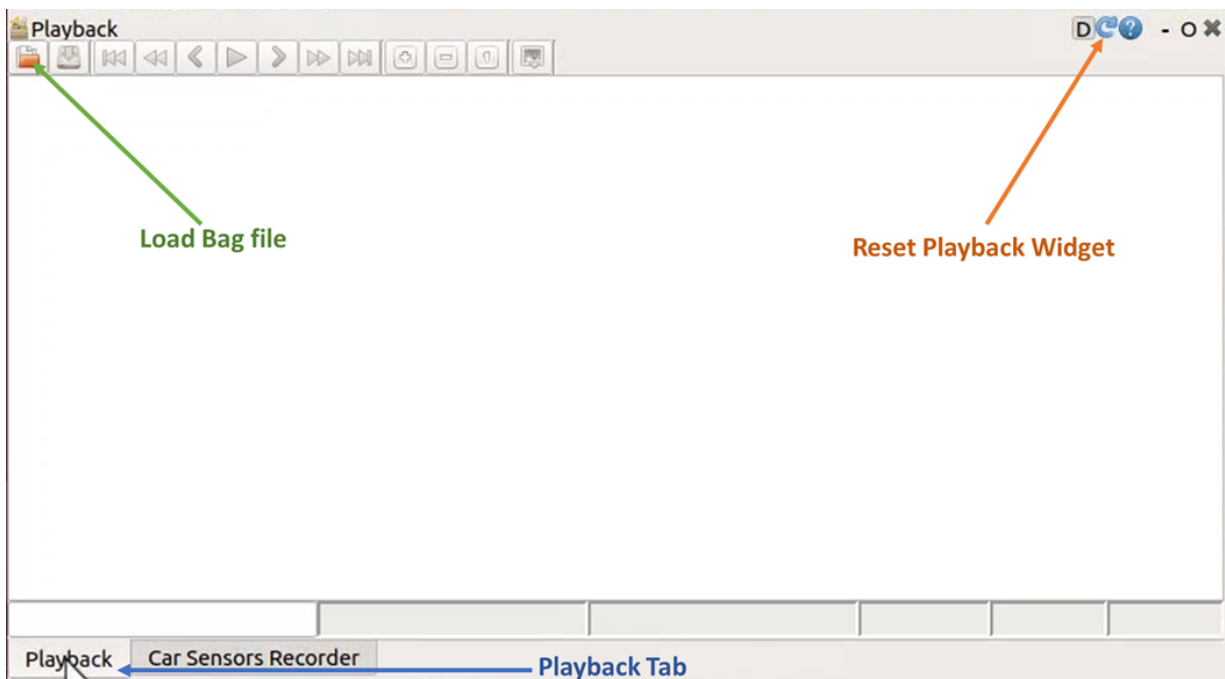
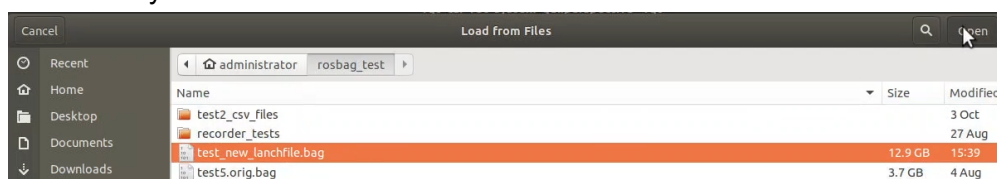


Figure 4: playback widget (no bag file loaded)

If you want to go back to the Car Sensors Recorder widget, press the Car Sensors Recorder tab.

To load a bag file press the open 'Load Bag file' icon (see figure 4 above).

Note / Troubleshoot: if after pressing the 'Load Bag file' icon, the GUI appears to be unresponsive; the window dialog box may have opened behind the GUI window. To fix this, go to the sidebar of the Ubuntu system and make sure there are more than one orange dots near the rqt logo. Click the logo and select the hidden dialog box; from here you can proceed normally.



Select the wanted bag file and click open.  
After a few seconds the widget should look something like this:

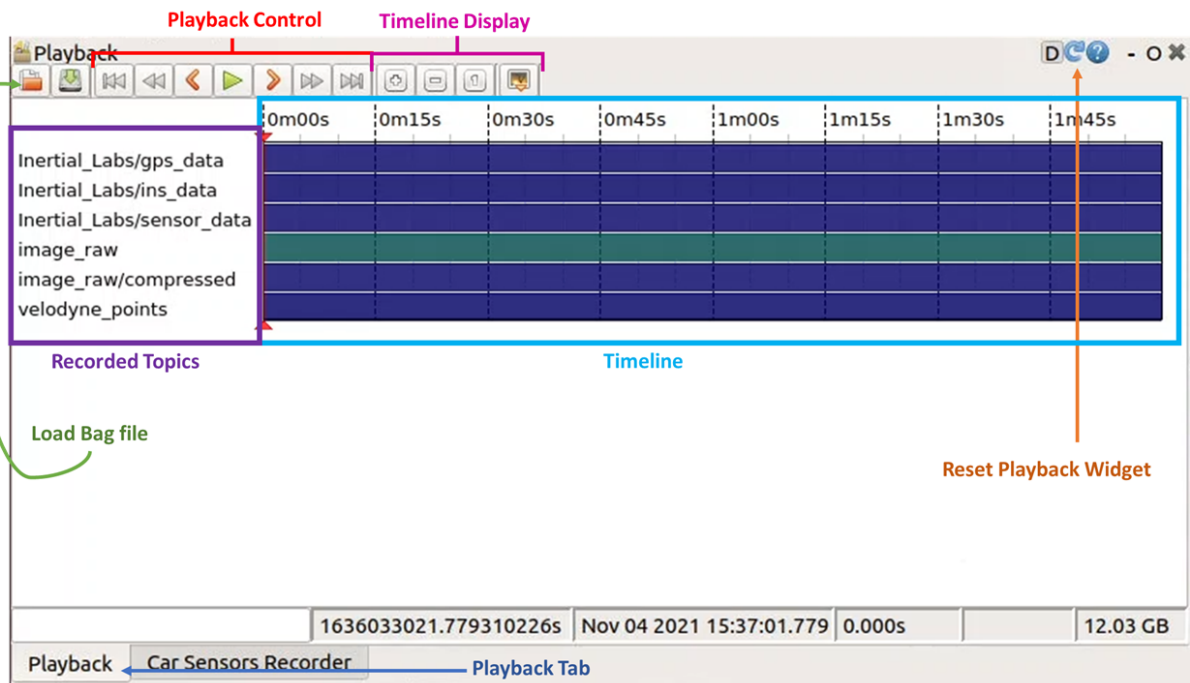


Figure 5: playback widget (bag file loaded)

If you press the 'Load Bag file' icon after a bag file was already loaded, the newly opened bag file will append it self to the existing data and timeline, thou if you want load a file and view it's data solely, you'll need to reset the widget first (by pressing the 'Reset Playback Widget' icon).

## Display Switch widget

( See figure 1(c) )

This widget is important for viewing live data and recordings; it lets you choose what data feed will appear on the rest of the widgets (image view, INS display, and rviz).

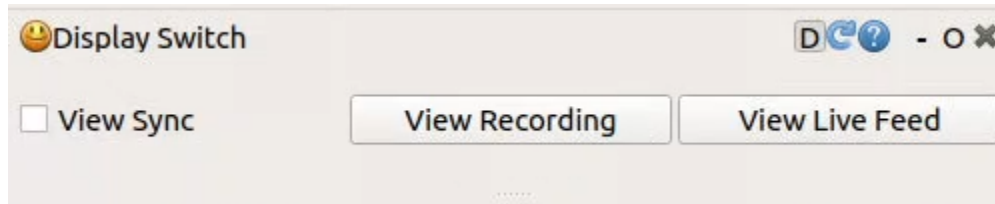


Figure 6: Display Switch widget

'View Live Feed' button - Switching the displays (the widgets associated with each sensor) to live feed data straight from the sensors. The data that inhibits this feed is the same data that is going to be recorded to the bag file via the `rqt_car_sensors_rec` widget.

'View Recording' button - Switching the displays (the widgets associated with each sensor) to the data that is playing in the 'Playback' widget.

Note: when the 'View Recording' button is pressed the 'View Sync' checkbox becomes available. The 'View Sync' checkbox switches the displays (the widgets associated with each sensor) between the data recorded in 'Sync mode' and 'All msgs'.

To read more on the "behind the scenes" of the display switching operation, check the refer to the [project book \(in Hebrew\)](#).

## How to playback a bag file

Turn on the Sensors Recording System (see start-up instructions).

Click on the 'Playback' tab on the GUI (see figures 1(e), 4, 5).


Click on the 'Load Bag File' button (the folder icon, see figures 4,5);

If nothing happens check if the open file dialog window is hidden beneath other windows.

Navigate and select the desired bag file you want to load.

Go to the 'Switch Display' widget and click on the "View Recording" button. If you wish to view the 'Sync mode' topics in the display instead of 'All msgs', check the box next to 'View Sync'.



Press the green 'Play' icon  in the 'Playback Control' section on the 'Playback' widget ( see figure 5).

Now the recorded data should be played and displayed on the other widgets.

The 'Playback Control' buttons along with the interactive Timeline helps you to navigate the recorded data. The buttons in the 'Timeline Display' area, helps you better navigate the timeline (see figure 5)

Note: This  button is very helpful, it resets the timeline display.

Shortcuts:

ctrl + mouse wheel = zoom in\out

middle mouse click + move the mouse = scroll the timeline

space = pause / play

## Troubleshooting

In order to display the data in real-time setting, the python-api for rosbag drops some msgs to compensate for the slow retrieval of the msgs from the bag file (mostly noticeable with the radar marker msgs). Not to be alarmed, the msgs are recorded and do exist inside the bag file. In order for the display the data without dropping msgs, you can right-click the white space inside the 'Playback' widget, a drop down menu will pop, now press 'play all messages' (or something similar to that) and you'll be able to see all the msgs but now time will slow down significantly.

# Data extraction

## General

An easy and convenient way to extract data from a bag file is to create a custom python script to map the data from the ros msg format into your desired form of database in python.

More on that:

[http://wiki.ros.org/rosbag/Cookbook#Export\\_message\\_contents\\_to\\_CSV](http://wiki.ros.org/rosbag/Cookbook#Export_message_contents_to_CSV)

[http://wiki.ros.org/ROS/Tutorials/reading%20msgs%20from%20a%20bag%20file#rosbag.2FTutorials.2Freading\\_msgs\\_from\\_a\\_bag\\_file.Option\\_2:\\_use\\_the\\_ros\\_readbagfile\\_script\\_to\\_easily\\_extract\\_the\\_topics\\_of\\_interest](http://wiki.ros.org/ROS/Tutorials/reading%20msgs%20from%20a%20bag%20file#rosbag.2FTutorials.2Freading_msgs_from_a_bag_file.Option_2:_use_the_ros_readbagfile_script_to_easily_extract_the_topics_of_interest)

[http://docs.ros.org/en/diamondback/api/rosbag/html/python/rosbag.bag.Bag-class.html#read\\_messages](http://docs.ros.org/en/diamondback/api/rosbag/html/python/rosbag.bag.Bag-class.html#read_messages)

Basic template python code:

```
import rosbag

bag=rosbag.Bag('bag_file_name.bag')

for topic,message,timestamp in bag.read_messages(topics=['/topic_name']):

    # Do something with message data
```

To access the msg data you need to be familiar with the msg format (different for each topic / sensor).

The details of the msgs for each topic are available in the [project book as a reference \(in Hebrew\)](#), or by googling the msg types.

In order to check what topics and msgs types are saved on a specific bag file, run in shell:

```
rosbag info saved_recording.bag
```

# Inertial Labs INS-DL

Here I will demonstrate the logical steps for data extraction from the msgs within the bag file.  
For example: INS - acceleration in the X direction

Understand what topic are you interested in:

From the [project book \(in Hebrew\), page 18](#):

מכיל את המידע הבא: /Inertial\_Labs/sensor\_data/  
Gyro(x,y,z) , Accelation(x,y,z) , Magnetic (x,y,z) , Temperature , Input Voltage , Pressure , Barometric height

We understand that /Inertial\_Labs/sensor\_data is the topic we are interested in.

Check the msg format of the topic, in our case, it's /Inertial\_Labs/sensor\_data:

[/sensor\\_data](#) format:

std_msgs/Header	header
geometry_msgs/Vector3	Mag
geometry_msgs/Vector3	Accel
geometry_msgs/Vector3	Gyro
float32	Temp
float32	Vinp
float32	Pressure
float32	Barometric_Height

We are interested in Accel (Acceleration), and we see that Accel is a geometry\_msgs/Vector3 type msg.

Let's check the geometry\_msgs/Vector3 msg format (by googling it).

[/geometry\\_msgs/Vector3](#) format:

float64 x  
float64 y  
float64 z

Here we are interested in the x direction.

Thous, to get the INS acceleration in the X direction, we will need to access the 'Accel' field of the msg and then the 'x' field. We get: msg.Accel.x

```
for topic,message,timestamp in bag.read_messages(topics=['/sensor_data']):
    temp_x_accel = message.Accel.x
    print(str(temp_x_accel))
    x_acel_array.append(temp_x_accel)
# or any other use
```

My attempt at getting the x-axis acceleration:

```
#!/usr/bin/env python
import rosbag
import sys
from datetime import datetime
import rospy
# filename: ins_extract_test.py
# creator: Michael Stillman

# check for 1 arg
if len(sys.argv) != 2:
    exit(-1)
# bag file name as an argument, for example 'test2.bag'
bag=rosbag.Bag(sys.argv[1])
x_acel_array = []

# printing in a loop over all the msgs in the bag file, that are related to the ins (data).
for topic,message,timestamp in bag.read_messages(topics=['/Inertial_Labs/sensor_data']):
    temp_x_accel = message.Accel.x
    print(str(temp_x_accel))
    x_acel_array.append(temp_x_accel)

time_date = datetime.fromtimestamp(timestamp.to_time())
print(time_date)
```

Shell output:

```
mike@MSubuntu:~/bag$ ./ins_extract_test.py test2.bag
0.0015
2021-08-04 09:47:19.719981
0.002
2021-08-04 09:47:19.760125
```

rosbag\_to\_csv

The ros package 'rosbag\_to\_csv' (already installed on the machine) that was mentioned in the [project book \(in Hebrew\)](#), works quite well with the topics related to the INS sensor.

Git and tutorial link: [https://github.com/AtsushiSakai/rosbag\\_to\\_csv](https://github.com/AtsushiSakai/rosbag_to_csv)

Result:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	time	header.seq	header.stamp.secs	header.stamp.nsecs	header.frame_id	Mag.x	Mag.y	Mag.z	Accel.x	Accel.y	Accel.z	Gyro.x	Gyro.y	Gyro.z
2	2021/08/04/09:47:19.719981	0	1628059639	719028899 F2001211		0	0	0	0.0015	-0.0415	0.997	-0.04	0	0
3	2021/08/04/09:47:19.760125	1	1628059639	739046711 F2001211		0	0	0	0.002	-0.041	0.998	0	0	-0.02
4	2021/08/04/09:47:19.760625	2	1628059639	759020426 F2001211		0	0	0	0.0015	-0.0415	0.997	-0.06	0	-0.02
5	2021/08/04/09:47:19.779264	3	1628059639	779044136 F2001211		0	0	0	0.0015	-0.041	0.9985	0	0	-0.02
6	2021/08/04/09:47:19.800015	4	1628059639	799392602 F2001211		0	0	0	0.002	-0.0415	0.9965	0	0.02	-0.02

Now one can slice the unneeded data from the csv file, and analyze the results.

## Velodyne LP16 Lidar / cloudpoint2 msg format

There are many ways to convert the cloudpoint2 msg format of the Velodyne lidar into a multidimensional array.

Example of web results:

<https://stackoverflow.com/questions/39772424/how-to-efficien>

<https://answers.ros.org/question/202787/using-pointcloud2-data-getting-xy-points-in-python/>

## My attempt at an array creation script for lidar msgs

I've made my version of a script using `ros_numpy` library (ROS version of numpy):

```
#!/usr/bin/env python
# filename: points_extract_test.py
# creator: Michael Stillman

# handle bag and pointcloud2 msg type + numpy version of ros
# if missing, to install via shell: sudo apt install ros-melodic-ros-numpy
import ros_numpy as rnp
import rosbag

import sys
```

```

from datetime import datetime
import rospy

# check for 1 arg
if len(sys.argv) != 2:
    exit(-1)
# bag file name as an argument, for example 'test2.bag'
bag=rosbag.Bag(sys.argv[1])

# printing in a loop over all the msgs in the bag file, that are related to the lidar points
cloud.
for topic,message,timestamp in bag.read_messages(topics=['/velodyne_points']):

    lidar_arr = rnp.point_cloud2.pointcloud2_to_xyz_array(message) #,remove_nans=True)
    time_date = datetime.fromtimestamp(timestamp.to_time())
    print(time_date)
    print(lidar_arr)
    print(lidar_arr.shape)
    print(type(lidar_arr))

```

Shell output:

```

mike@MSubuntu:~/bag$ ./points_extract_test.py test2.bag
2021-08-04 09:47:19.828035
[[-5.29064608 -1.6925317  0.48598042]
 [-5.29622984 -1.67497778  0.48598042]
 [-5.55687714 -1.73609579  0.50933814]
 ...,
 [-0.15972877 -2.93980575  0.78888047]
 [-0.16998971 -2.9392302  0.78888047]
 [-0.17973572 -2.93865037  0.78888047]]
(10696, 3)
<type 'numpy.ndarray'>
2021-08-04 09:47:19.928551

```

## MISC

Other launch files that can be used with 'roslaunch car\_rec\_system' call:

car\_rec\_system.launch - whole recording system

gui\_only.launch - just the gui

Ff\_only.launch - just the tf nodes (tells the distance between the radar and lidar, used in rviz)

`mux_only.launch` - just the display muxes (used for display switching)  
`mux_and_tf.launch` -display muxes and tf

These are useful for debugging.