Incompatibility with pointfield layouts in autoware_pointcloud_preprocessor package #5031

Closed   Answered by yukkysaito   **mbharatheesha** asked this question in **Q&A**

mbharatheesha on Jul 25

We have been actively using the `ground_segmentation` and `euclidean_cluster` package from the `perception` stack to process lidar data from velodyne and ouster lidars in our ROS2 projects. Many thanks for these extremely useful packages. We use the different ground segmentation filters available and voxel grid clustering.

In recent updates (since `v0.8.0`) on `(autoware_)pointcloud_preprocessor` package, I notice that there is either a throttle error regarding pointfield layout incompatibility (Velodyne pointclouds) or, a point cloud invalidation (Ouster pointclouds). The details are as follows:

On `v0.8.0`:

On subscription to a `sensor_msgs::msg::PointCloud2` topic, the validation of the point cloud (both via `input_indices_callback` or `faster_input_indices_callback` was happening based on this logic.

This was working fine on data from both velodyne and ouster lidars.

On Latest `autoware_pointcloud_preprocessor`:

The recent update adopts a more strict validation of the pointcloud that also checks the adherence of the data to the point cloud field offsets as defined in the supported Point definitions such as `XYZIRC/XYZIRCAEDT`. However, this leads to a throttling error message for Velodyne points as the data we have only adheres to the XYZI format. And for Ouster lidar data, it completely invalidates the pointcloud due to the failing checks here and here.

In both instances, the reason for failure is the data we have from Ouster lidar driver has an offset of 16 for intensity data, while the point field layout of the `XYZI` or `XYZIRC` expects the offset for intensity to be 12.

Sample ouster lidar pointcloud data that causes the failure (Note the offset for intensity field is 16 in the data):

```
height: 128
width: 2048
fields:
- name: x
  offset: 0
  datatype: 7
  count: 1
```

Category
Q&A

Labels
None yet

2 participants

```
  - name: y
    offset: 4
    datatype: 7
    count: 1
  - name: z
    offset: 8
    datatype: 7
    count: 1
  - name: intensity
    offset: 16
    datatype: 7
    count: 1
  - name: t
    offset: 20
    datatype: 6
    count: 1
  - name: reflectivity
    offset: 24
    datatype: 4
    count: 1
  - name: ring
    offset: 26
    datatype: 4
    count: 1
  - name: ambient
    offset: 28
    datatype: 4
    count: 1
  - name: range
    offset: 32
    datatype: 6
    count: 1
  is_bigendian: false
  point_step: 48
  row_step: 98304
```

The check on the datatype of the intensity field will also fail as it is of type FLOAT32 in our data, but the `XYZIRC` format expects `intensity` to be a `uint8`.

Question(s)

1. Since it is unclear when (or if) the lidar driver packages will update to these desired formats, can we perhaps consider a parameter/flag to enable/disable the stricter validation?
2. Are there any utility functions or tools within `autoware` that can convert incompatible pointcloud formats to the desired format (perhaps with a passthrough or something similar?)

@amadeuszsz **@yukkysaito** **@amc-nu**

Kindly let me know if you need additional information

↑ 1

Answered by **yukkysaito** on Jul 25

@mbharatheesha Thank you for your questions.

Up until now, Autoware had not been adhering to the prescribed design data fields, but they have been corrected to align with the interface design in the Pull Request. I apologize for any inconvenience caused by this sudden

View full answer ↓

1 comment · 1 reply

Oldest Newest Top

yukkysaito on Jul 25 Maintainer

@mbharatheesha Thank you for your questions.

Up until now, Autoware had not been adhering to the prescribed design data fields, but they have been corrected to align with the interface design in the Pull Request. I apologize for any inconvenience caused by this sudden change.

> Since it is unclear when (or if) the lidar driver packages will update to these desired formats, can we perhaps consider a parameter/flag to enable/disable the stricter validation?

In the Autoware interface, intensity is intended to be handled as uint8. If necessary, it might be best to fork the package and handle it as your own local modification.

> Are there any utility functions or tools within autoware that can convert incompatible pointcloud formats to the desired format (perhaps with a passthrough or something similar?)

It seems that the easiest solution in terms of performance would be to convert intensity to uint8 in the Ouster ROS driver. What do you think? Alternatively, to accommodate the differences among various vendors, we are creating Nebula as a universal lidar driver. It might also be a good idea to implement an Ouster decoder in Nebula.

If you have any requests for changes to float or other types, please propose them along with the reasons.

✓ Marked as answer ↑ 1 👍 3 1 reply

mbharatheesha on Jul 26 Author

Thank you very much for the quick response @yukkysaito .

> Up until now, Autoware had not been adhering to the prescribed design data fields, but they have been corrected to align with the interface design in autowarefoundation/autoware.universe#6996. I apologize for any inconvenience caused by this sudden change.

Thank you very much for the clarification.

> In the Autoware interface, intensity is intended to be handled as uint8. If necessary, it might be best to fork the package and handle it as your own local modification.

I think this is probably the most realistic solution for us. The only thing that I am unsure about is, whether the offset difference in our lidar data (our data has a byte offset of 16 for intensity) will cause any other unexpected behaviors if a `memcpy` is used somewhere in the code. Essentially, in the ouster lidar data, there seems to be a gap between the 12th and 16th byte. So, unless I also define a new point type and then make the check accordingly, it might be risky to do this I think?

> It seems that the easiest solution in terms of performance would be to convert intensity to uint8 in the Ouster ROS driver. What do you think? Alternatively, to accommodate the differences among various vendors, we are creating Nebula as a universal lidar driver. It might also be a good idea to implement an Ouster decoder in Nebula.

Using Nebula might be an interesting option indeed. I will explore it.

> If you have any requests for changes to float or other types, please propose them along with the reasons.

Well, the only reason I can think of is, the data we get from Velodyne using the ros2 driver in the organized XYZIRT case or in the general pointcloud case still use `float32` as the datatype for intensity.

Since this is quite a popular lidar in many usecases, it might be nice to also support `float32` for intensity data. But, this is under my assumption that the Velodyne driver will continue to use `float32` in the future too, for `intensity` data. I am not sure how the development plan ahead is for the velodyne driver though and my assumption might be incorrect.

Answer selected by **mbharatheesha**