Why is `voxel_based_compare_map_filter` the default `compare_map_segmentation` implementation ? #2647

Answered by VRichardJP VRichardJP asked this question in Q&A

VRichardJP on Jun 2, 2022 Collaborator edited

pointcloud_map_filter.launch.py uses `compare_map_segmentation::VoxelBasedCompareMapFilterComponent` as default implementation, yet it seems to be by far the slowest implementation among the available ones. Is there any reason behind this choice?

I have measured message latency between the input topic (`/perception/obstacle_segmentation/pointcloud`) and output topic (`/perception/object_recognition/detection/pointcloud_map_filtered/pointcloud`) of the `compare_map_segmentation` node . Here is how each implementation perform on my machine:

```
distance_based_compare_map_filter           | 295ms
voxel_based_approximate_compare_map_filter | 75ms
voxel_based_compare_map_filter              | 874ms
voxel_distance_based_compare_map_filter     | 214ms
```

`voxel_based_approximate_compare_map_filter` is 10 times faster than `voxel_based_compare_map_filter`, but even `voxel_distance_based_compare_map_filter` and `distance_based_compare_map_filter` are 3 to 4 times faster.

I am not sure I understand the whole rational behind `voxel_based_compare_map_filter`, in particular why it is necessary to check 27 times whether each input point is close to the map points. What is certain is that it is rather slow.

If I am not wrong, the `distance_based_compare_map_filter` implements an exact solution, `voxel_distance_based_compare_map_filter` a slightly approximated one, and `voxel_based_approximate_compare_map_filter` a bigger approximation. So even if we stick to an exact solution, there is no reason to use `voxel_based_compare_map_filter` over `distance_based_compare_map_filter`.

Or may I have overlooked something?

Note: by the way `voxel_distance_based_compare_map_filter_nodelet.cpp` uses nearestKSearch to find closest neighbor, while `radiusSearch` with `max_nn=1` should be faster (since search is stopped once distance is necessary bigger than specified max distance)

Category

Q&A

Labels

component:percept...

component:sensing

3 participants

↑ 1 👍 2

Answered by **VRichardJP** on Jun 29, 2022

At the end of the day, it was a silly mistake on my size: the build I used for benchmarking was not compiled in release mode...
With release build, I get the following:

View full answer ↓

3 comments · 2 replies

Oldest | Newest | Top

taikitanaka3 on Jun 3, 2022 Collaborator

@VRichardJP
this PR I added computation time for pointcloud related processor
can you compare time with graph using plot juggler?
autowarefoundation/autoware.universe#946

> why it is necessary to check 27 times whether each input point is close to the map points. What is certain is that it is rather slow.

@yukkysaito
can you answer this?

↑ 1 0 replies

yukkysaito on Jun 17, 2022 Maintainer

@VRichardJP
Thank you for your analysis.
I can't give you the evaluation results now, but our results differ from your results we measured long ago.

`Voxel based approach` access directory using index access, so it is expected to be fast.
But `KD Tree based approach` access using tree search, then if the map is large, it is expected to be slow.

Anyway, we evaluate again to solve different result.
What is your environment and what maps and rosbags did you use?

> I am not sure I understand the whole rational behind voxel_based_compare_map_filter, in particular why it is necessary to check 27 times whether each input point is close to the map points. What is certain is that it is rather slow.

Regarding the `voxel_based_compare_map_filter` searching 27 times, if a point is searched only in the voxel to which the point belongs, it may not be searched properly when it is near the voxel's boundary plane. Therefore, it searches surrounding voxels.

↑ 1 👀 1 1 reply

VRichardJP on Jun 17, 2022 Collaborator Author edited

Thank you for the explanation.

I cannot share the map or rosbag data, but I will check the cloud sizes I used.

So in the voxel based approach, the distance from the point to the voxel centroid is checked, up to 27 times (for each 3x3x3=27 neighbor voxel). In particular, if most of the input cloud is not on the map, the algorithm will be quite slow. In the kdtree approach, the closest neighbor is found in O(ln(map_size)). Does it takes that many comparisons for the kdtree to find the closest neighbor?

VRichardJP on Jun 29, 2022 Collaborator Author

At the end of the day, it was a silly mistake on my size: the build I used for benchmarking was not compiled in release mode...
With release build, I get the following:

```
voxel_based_approximate_compare_map_filter | 14ms
voxel_based_compare_map_filter             | 25ms
voxel_distance_based_compare_map_filter    | 39ms
```

Suddenly `voxel_based_compare_map_filter` is not that bad anymore, just slightly slower than the approximate version. But then it is a trade off between performance and precision.

Marked as answer ↑ 1 1 reply

taikitanaka3 on Jun 29, 2022 Collaborator

@VRichardJP
I think the main contribution for this computation was obstacle pointcloud down sampling autowarefoundation/autoware.universe#961
This makes computation time much lower

👍 1

Answer selected by **VRichardJP**