
TIER IV
2023 / 07 / 12
Development about
Occupancy Grid Map Fusion

CONTENTS

Related Links

PRs(~2023/April)

- **Solution A**: move scan origin
  - PR#2939 : enable to select gridmap origin frame
  - PR#3032 : add scan frame option and fix scan
- **Solution B**: create ogm in each sensor
  - PR#3032 : Separate scan_frame and gridmap_origin
  - PR#3054 : Filter obstacle pointcloud by raw pointcloud
  - PR#3312 : Publish time synced raw pointcloud from sensing component (WIP)
- **Solution B**: OGM Fusion Node
  - PR#3058 : Refactor OGM launcher
    - PR#3340 : Bug fix
  - PR#3107 : Add grid map fusion node (WIP)

Links

- Issue #2906

Problem Statement

—

01

About OGM(Occupancy Grid Map)

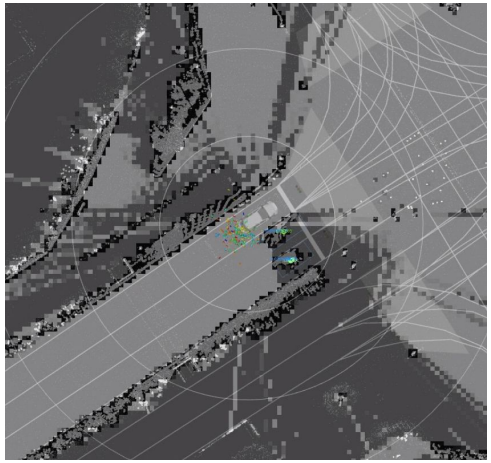
OGM is used to represent the presence of an obstacle in environment.
It is used in:

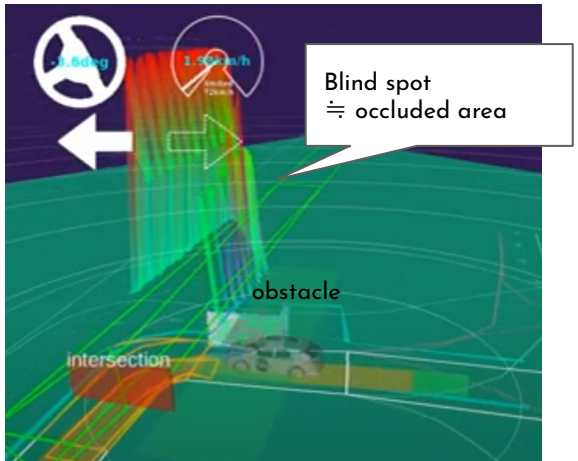
- Perception
  - object filtering
- Planning
  - intersection module
  - blind spot estimation...etc

OGM is created with

- raw lidar pointcloud
- obstacle lidar pointcloud

OGM example in sample





(Appendix) OGM generation

Currently we only support two lidar based methods:

- laserscan based method
  - 👩 only need obstacle pointcloud
  - 👩‍🦰 short range visibility (obstacle)
- Pointcloud based method
  - 👩 more detailed
  - need raw and obstacle pointcloud

We recommend latter in many cases.

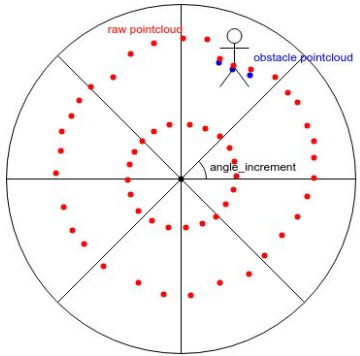
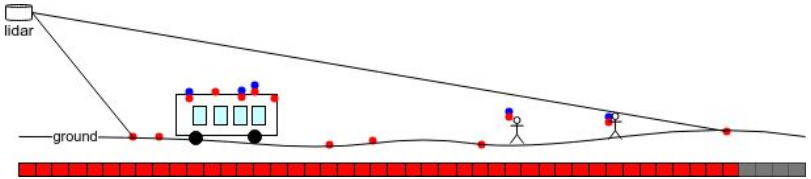
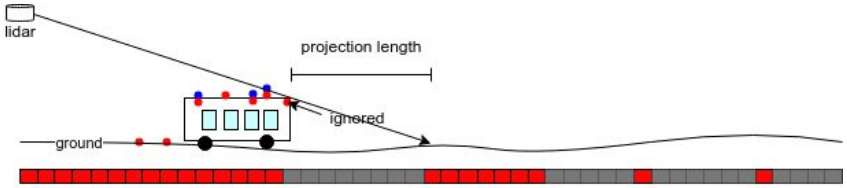


Fig. Pointcloud based OGM creation

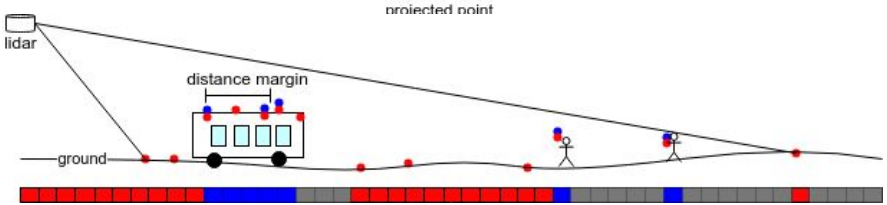
1. Fill visible range with "Free"



2. Set blind spot as "Unknown"



3. Set obstacle pcs as "Occupied"

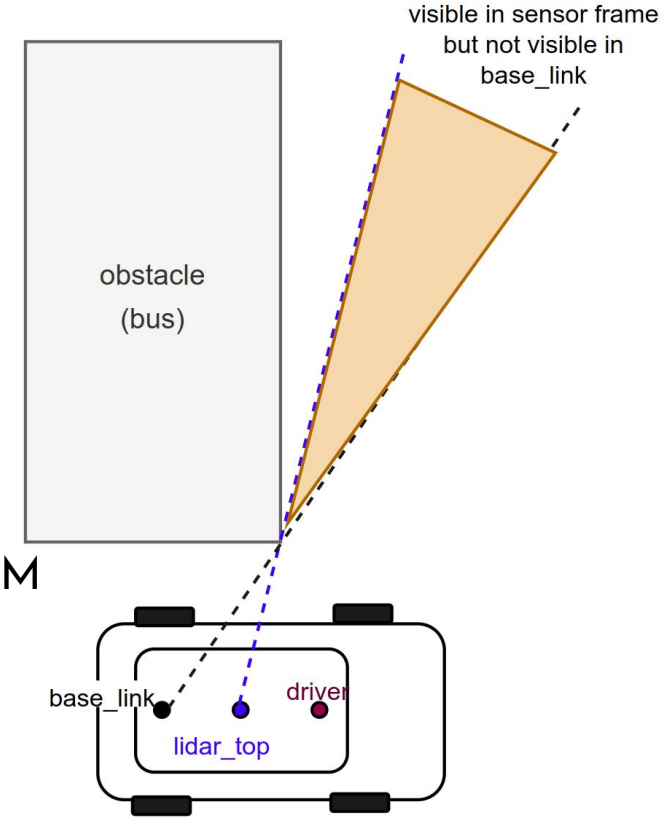


Problem statement

OGM is often scanned on base_link (rear side).

This sometimes result in
shorter range visibility in intersection

It do not reflect true FOV of the each sensors
because scanning origin is not on sensor



Goals:

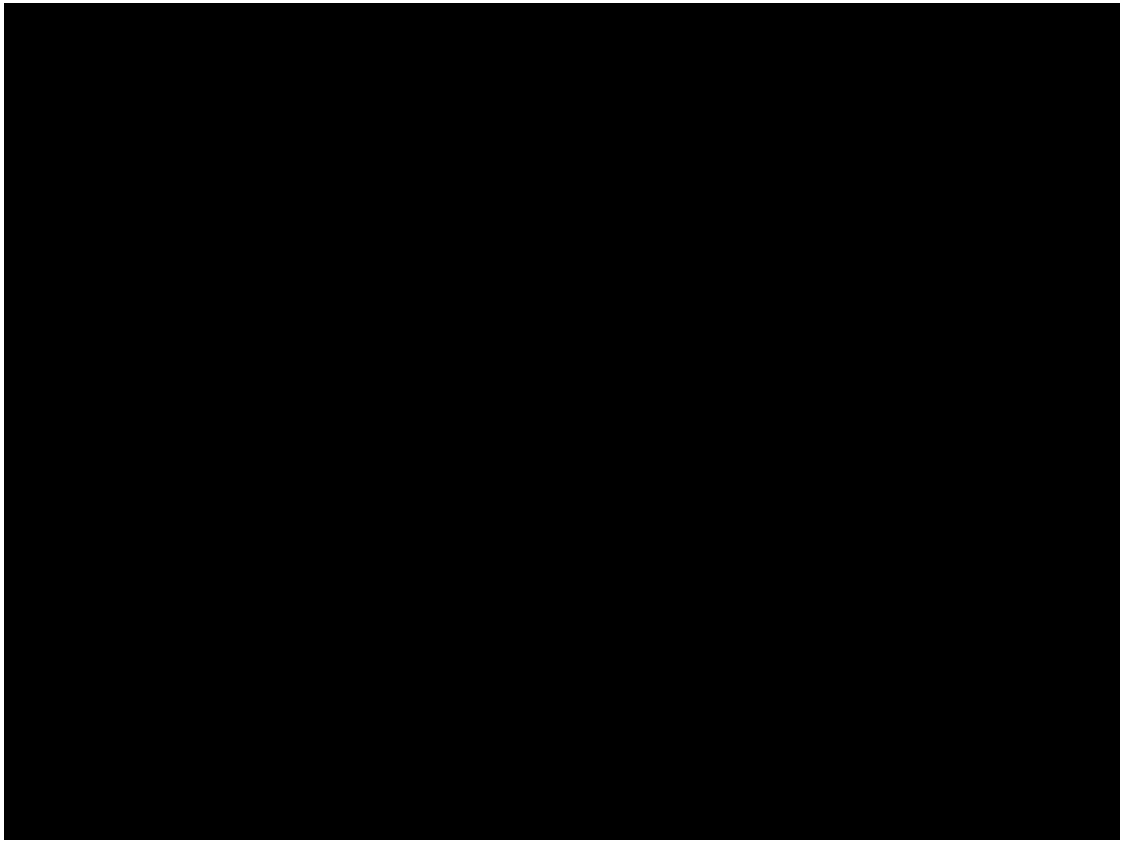
- Provide method to reflect true sensor FOV to OGM

For details, please see [Issue]

(Appendix): Problem statement in Video

Blind spot in intersection is displayed as
area surrounded by walls.

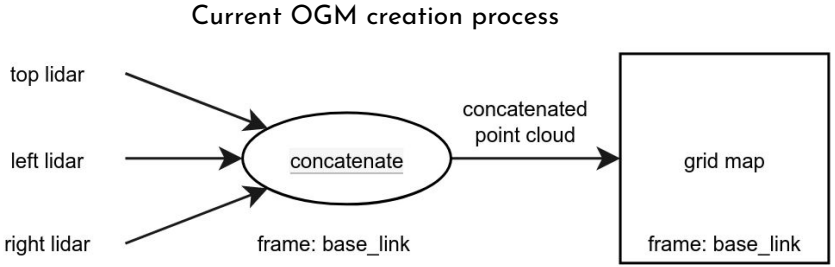
Planning module tries
to expose the vehicle head to see this blind spot.



Basic Idea and Solutions

Basic Idea:
The scan origin in OGM creation should be the same with the sensor origin.

- Solution A
  - Move scan origin
- Solution B
  - Create OGM in each sensor frame
  - Fuse multiple OGM output



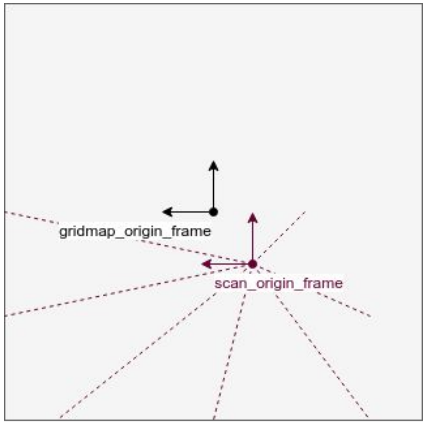
Solution A: Move scan origin (Merged)

In the config/*.yaml file you can set origin separately

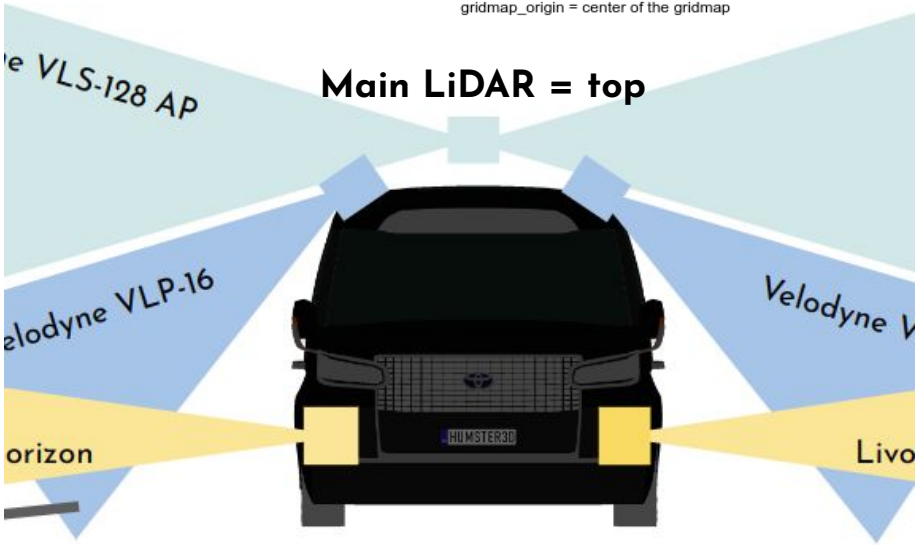
- gridmap_origin_frame
  - geometrical center of OGM
- scan_origin_frame
  - origin of virtual scan

See README.md for details.

This works well in small vehicle
with one main-lidar and some sub-lidar builds.



gridmap_origin = center of the gridmap

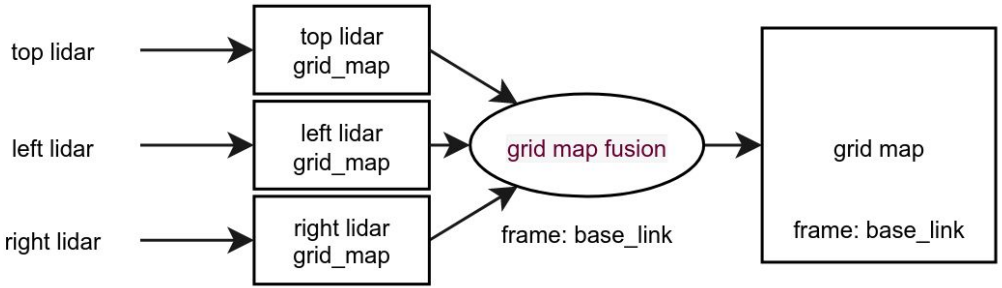


Solution B: Create OGM in each sensor frame

In the larger vehicle, we can not choose appropriate origin.

We need to

- Create OGM in each sensor frame
- Fuse OGM



generate grid map in each sensor frame

Detailed OGM Fusion Flow

02

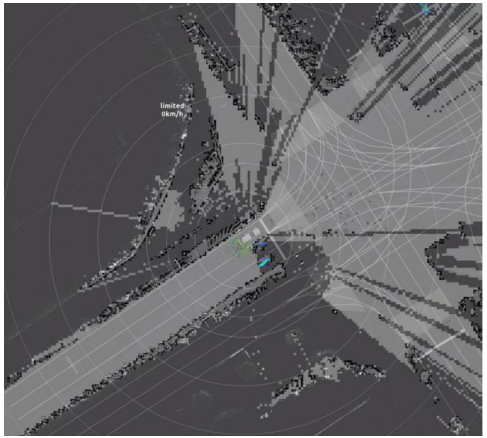
Solution B: Create OGM in each sensor frame

OGM in each sensor is created on same map_origin and then fused.

See document(in PR). reference -> [CARLOS GÁLVEZ 2015]

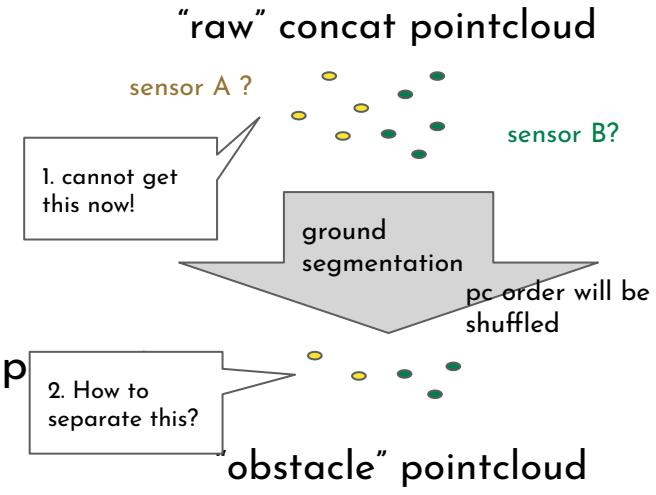
Example in sample-rosbag in documentation

Fused OGM

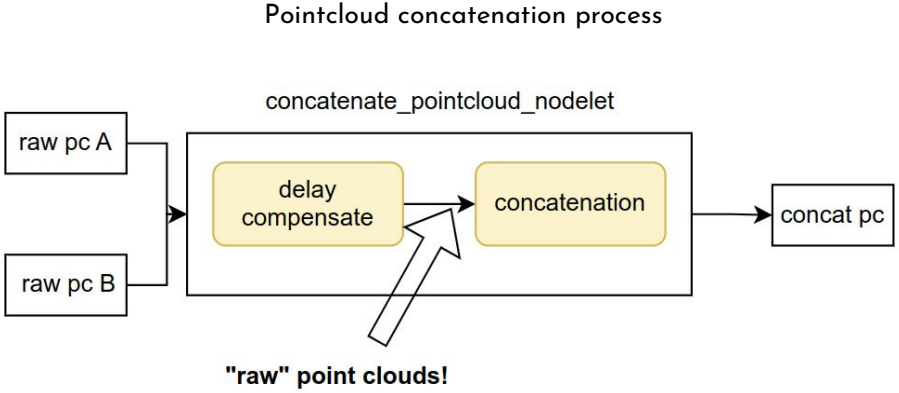


Create OGM in each sensor frame

we need “raw” and “obstacle” pointcloud in each sensor



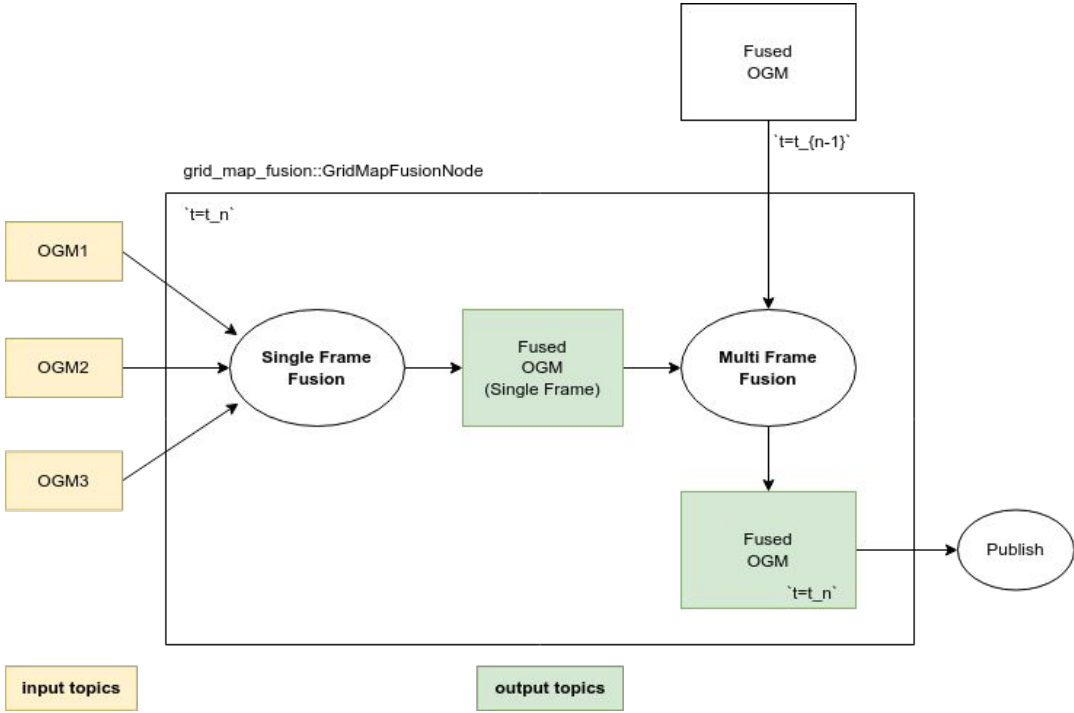
1. **There are no “raw” pointcloud published**
   - point cloud concatenation includes delay comp
   - need this PR
2. **How to separate “obstacle” pointcloud**
   - Use “raw” pointcloud to filter concatenated obstacle pointcloud
   - use `filter_obstacle_pointcloud_by_raw_pointcloud` option



OGM fusion flow

We have two choices:

1. **synchronized fusion**
   - fuse in the certain frame
   - 🙆‍♀️ easier to debug
   - 🤦‍♀️ sensor output should synchronized (for LiDAR)
2. **async fusion**
   - Update fused OGM when message comes
   - 🙆‍♀️ no sync process
   - better in multi modal OGM fusion
   - not implemented yet... 🤷‍♀️



OGM fusion method [CARLOS GÁLVEZ 2015]

OGM fusion method requires:

- “Unknown” observations should not affect to Fusion.
- Able to handle each sensor weight
- Able to manage conflict
- Independent to input order

policy	description
overwrite	more critical state is overwritten: occupied -> free -> unknown
log-odds	most simple bayes rule
dempster-shafer	able to handle unknown information directly

(Appendix) Log-Odds fusion

Log-Odds based method is one of the bayesian filter implementation

Basic equation is like:

$$l_t(m_{ij}) = \log \frac{p(m_{ij}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t})}{1 - p(m_{ij}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t})} = \log \frac{p(m_{ij}|\mathbf{z}_t, \mathbf{x}_t)}{1 - p(m_{ij}|\mathbf{z}_t, \mathbf{x}_t)} + l_{t-1}(m_{ij}) - \log \frac{p(m_{ij})}{1 - p(m_{ij})}$$

$$l(\mathbf{m}) = \log \frac{p(\mathbf{m})}{1 - p(\mathbf{m})} \quad ; \quad l_k(\mathbf{m}) = \log \frac{p_k(\mathbf{m})}{1 - p_k(\mathbf{m})}$$

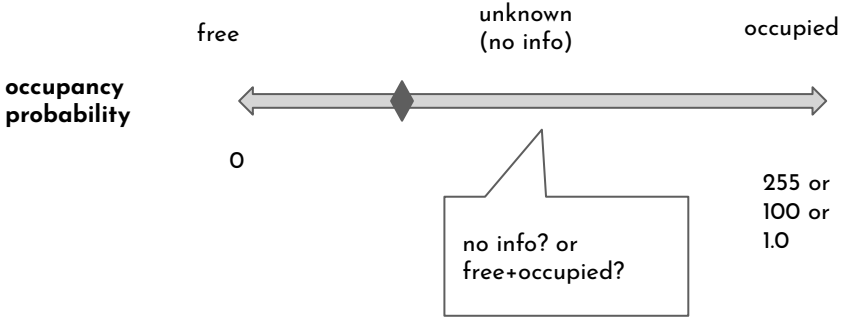
The fused log-likelihood then simply becomes:

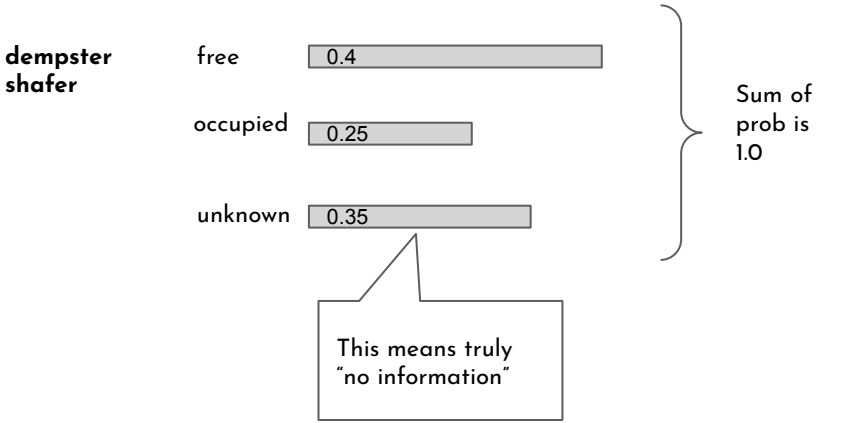
$$l(\mathbf{m}) = \sum_k l_k(\mathbf{m})$$

Actually, we use “weighted” log-odds

(Appendix) Dempster Shafer Theory [TIER IV internal link]

- 🙆 able to handle each state probability separately
  → aware of difference in "no info" and "conflicted"
- 🙆 can suppress unknown probability after fusion
- 🤦 conflict sometimes lead to unintended result
- 🤦 ros message limitation
  - output will be occupancy probability...



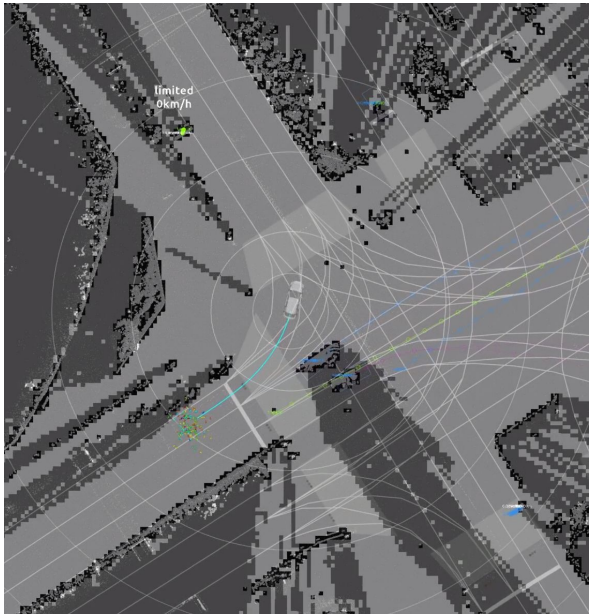


Comparison: Single Frame Fusion

Each method does not differ so much. Because:

- **input ogm is binary-like**: free(0) or occupied(255)
- overwrapped region is not large enough

"overwrite" policy



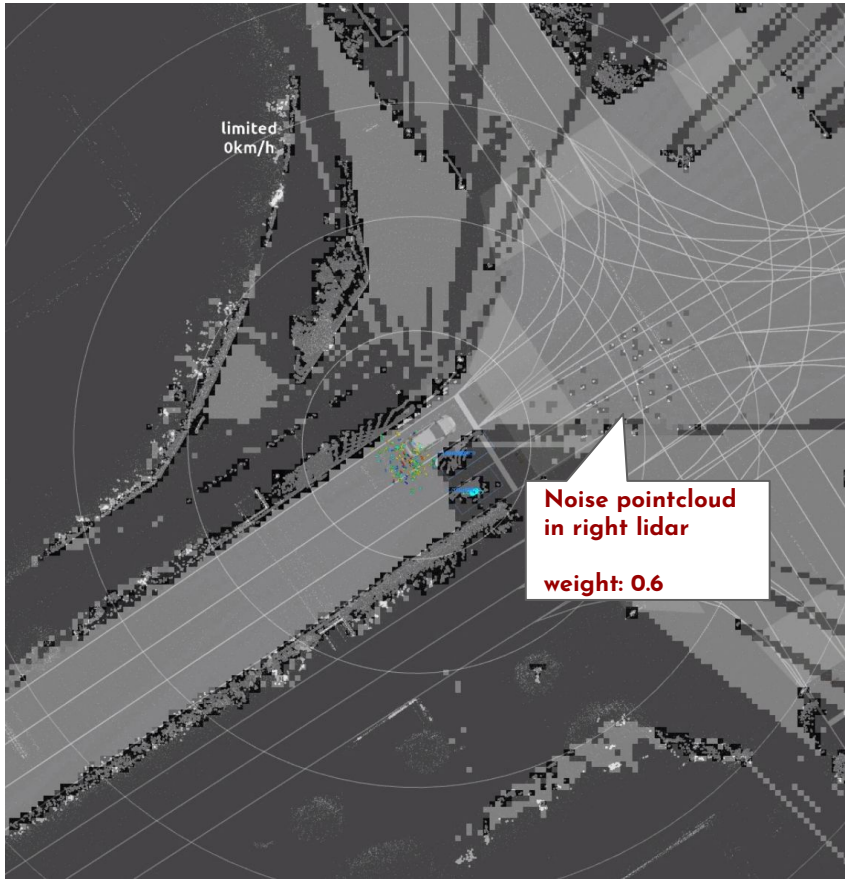
"dempster-shafer" policy

(Appendix) use weights to suppress noise

Each sensor weight is
another important parameter

We can suppress noise from less trustable sensor

```
each_ogm_output_topics:
  - "/perception/occupancy_grid_map/top_lidar/map"
  - "/perception/occupancy_grid_map/left_lidar/map"
  - "/perception/occupancy_grid_map/right_lidar/map"
each_ogm_sensor_frames:
  - "velodyne_top"
  - "velodyne_left"
  - "velodyne_right"
# reliability of each sensor (0.0 ~ 1.0) only work with "log
each_ogm_reliabilities:
  - 1.0
  - 0.6
  - 0.6
```



Comparison: Processing time

Processing time:

overwrite >> dempster-shafer ≒ log-odds

Want faster processing
→ overwrite

Need sensor weighting
→ log-odds, dempster-shafer

Method	processing time [ms]
overwrite	mean: **4.162326**
	min: 3.000000
	max: 7.546000
log-odds	mean: **8.159959**
	min: 5.859000
	max: 13.731000
Dempster-shafer	mean: **7.489534**
	min: 5.049000
	max: 16.239000

Conclusion

03

Conclusion

- OGM virtual scan should be done in (main) sensor frame

- solution A
  - Move virtual scan origin (merged)
- solution B
  - publish raw pointcloud (waiting for merge)
  - OGM fusion (under construction)
    - move to new package -> ogm_fusion or something?
    - fix conflicts with newer changes
    - (add asynchronous fusion node) (Evaluate in planning)
  - concerns:
    - increasing traffic, computational resources...
      (component container sometimes stops...)

Appendix: Other TODO

04

Current costmap value settings

original settings in nav_msgs/OccupancyGridMap messages:

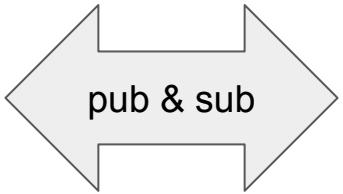
data int8[]

-127~128

costmap in autoware S/P nodes:

data char[]

0~255
0: Free
128: NO_INFORMATION
255:Occupied



Existence Probability

messages in autoware

data char[]

1~99
1: Free
99:Occupied

🤦 Hard to explain sensor FOV by this existence probability

Change costmap value settings

original settings in nav_msgs/OccupancyGridMap messages:

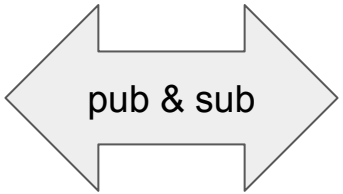
data int8[]

-127~128

costmap in autoware S/P nodes:

data char[]

0~255
0: Free
254: Occupied
255: NO_INFORMATION



Existence Probability

messages in autoware

data char[]

-1, 0~100
-1: NO INFO
0: Free
100:Occupied

Add FOV information. cf. navigation2