Find a redundant code in behavior_path_planner #4748

Closed **veqcc** started this conversation in **General**

veqcc on May 21 Collaborator edited

Hello planning maintainers!

I have found through flamegraph that the `computeCollisionIndexes` is the latency bottleneck in behavior_path_planner module.
Especially, `discretizeAngle` function is the main cause of the heavy latency.

Here, I have one question:

In `addIndex2d` function below, `pose_local` is passed to `pose2index` function.
https://github.com/autowarefoundation/autoware.universe/blob/0ddcca1cdd3a41ef9cc5452d71ea6a350ec89dc2/planning/behavior_path_planner_common/src/utils/occupancy_grid_based_collision_detector/occupancy_grid_based_collision_detector.cpp#L138

```
const auto addIndex2d = [&](const double x, const double y) {
    ...
    geometry_msgs::msg::Pose pose_local;
    pose_local.position.x = base_pose.position.x + offset_x;
    pose_local.position.y = base_pose.position.y + offset_y;
    const auto index = pose2index(costmap_, pose_local,
param_.theta_size);
    ...
  };
```

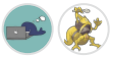
This `addIndex2d` is used many times soon after defined:

```
for (double x = back; x <= front; x += costmap_.info.resolutio
2) {
    for (double y = right; y <= left; y +=
costmap_.info.resolution / 2) {
      addIndex2d(x, y);
    }
    addIndex2d(x, left);
  }
  for (double y = right; y <= left; y += costmap_.info.resolution
/ 2) {
    addIndex2d(front, y);
  }
  addIndex2d(front, left);
```

Category
General

Labels
component:planning
type:performance

2 participants



The `pose2index` is defined here:
https://github.com/autowarefoundation/autoware.universe/blob/0ddcca1cdd3a41ef9cc5452d71ea6a350ec89dc2/planning/behavior_path_planner_common/src/utils/occupancy_grid_based_collision_detector/occupancy_grid_based_collision_detector.cpp#L34

```
IndexXYT pose2index(const nav_msgs::msg::OccupancyGrid & costr
const geometry_msgs::msg::Pose & pose_local, const int
theta_size) {
  const int index_x = pose_local.position.x /
costmap.info.resolution;
  const int index_y = pose_local.position.y /
costmap.info.resolution;
  const int index_theta =
discretizeAngle(tf2::getYaw(pose_local.orientation), theta_size);
  return {index_x, index_y, index_theta};
}
```

My question is that where the `pose_local.orientation` is defined and is it needed to calculate `index_theta`?
If the value `pose_local.orientation` is always the same in those `pose2index` calls in `addIndex2d`, then I think we can delete the redundant `discretizeAngle` calls and make a lot of performance improvement!!

Thank you for reading.

↑ 3 ❤️ 1

1 comment

Oldest | Newest | Top

mehmetdogru on May 23 Maintainer

@soblin is working on the issue. Will create a PR later.

↑ 1 👍 1 0 replies