Autonomous Systems Project

Challenge 1 – Search and Rescue(Mission)

Team: autsys-projects-gkd

Team members: Yiwei Wang, Yiyang Li, Tao Ma, Chenming Wang

victims_searching.cc

`void drone_CurrentState(const nav_msgs::Odometry::ConstPtr& cur_state)`

- The callback function of topic /current_state.
- Functionality: Save the current pose and velocity of the drone to vector<double> curpos ={dpos_x, dpos_y, dpos_z} and vector<double> curvel ={dvel_x, dvel_y}.
- Methods used in this callback function:
  - signal_generation(curpos, curvel): Act like a realistic sensor model. Generate the signal strength and angle when the drone's position is within the range that the signal could be detected.
  - signal_processing(curpos, curvel): Process the signal strength and angle for victims' position prediction and local searching part.

`void unity_map_msg(const std_msgs::Int32MultiArray::ConstPtr& unitymap)`

- The callback function of topic /unity_map.
- Functionality: Receive the map information from unity and save it to map_msg.data.
- Methods used in this callback function:
  - victims_position_generation(area_range_x,area_range_y, victim_num): Generation of victims' position on the snow mountain, victim_num is the number of victims.

`void beaconSS(const std_msgs::Float32::ConstPtr& beaSS)`

- The callback function of topic /gen_beaconSignalStrength.
- Functionality: Save the signal strength data to beaconSignalStrength, discriminate weather the condition for local search is met, call the local search function. As long as local search mode is 2 and the function detects signal strength, it will break global search and control the drone start to local search.

`void global_with_angle(const geometry_msgs::PolygonStamped::ConstPtr& msg)`

- The callback function of topic /victimCoord_continous_topic
- Functionality: When a victim is found, this function can break the global search and control the drone to fly to the predicted location of victim.

void map3d_generation(std::vector<int> &map_x_coord, std::vector<int> &map_y_coord)

- Functionality: Creating x and y positions for each point on the snow mountain, since we can get the altitude(z) value of each point from the callback function void unity_map_msg. After calling this method, each vector<int> {map_x_coord[i], map_y_coord[i], map_msg.data[i]} will be a point on the snow mountain.
- Variables:
  - std::vector<int> map_x_coord: size of 24000.
  - std::vector<int> map_y_coord: size of 24000.

`double find_altitude(double x, double y)`

- Functionality: Given the x and y positions of a point on the snow mountain, this method can return the altitude value in type of double. To realize the functionality, we will search in x coordination to get a list of indexes that satisfy, then search in y coordination in this list to find the point.

- Input: double x, double y.
- Output: double result = (double) altitude.

```
void victims_position_generation(std::vector<int> &x_range,
std::vector<int> &y_range, int vicnum)
```

- Functionality: Generation of several victims' positions in the mountain area. The range of x and y is: area_range_x = {30, 270}, area_range_y = {-70, -10}. There are two modes for the generation: 1. Randomly generate victims' position. 2. Given three victims' positions for testing.
- Input: std::vector<int> x_range, std::vector<int> y_range, int vicnum.
- Victims' position will be saved in the global variable "vector<vector<double>> victims_pos".

```
void signal_generation(vector<double> curpos, vector<double> curvel)
```

- Functionality: When a victim's signal is detected, compare the distance between the current drone position and the beacon position, then generate signal strength and angle including noise.
- Input: vector<double> curpos, vector<double> curvel
- Publish to topics: /gen_beaconSignalStrength, /gen_beaconAngle.
- Signal strength: When the distance is smaller than 15 meters, a signal will be generated. It is linear to the distance from 0 to 15 meters, gen_beaconSignalStrength = 1 - space_distance / 15 + SignalStrength_noise.
- Signal angle: When the distance is smaller than 15 meters, a signal will be generated. The orientation is from the current position to the beacon position in x-y plane.

void signal_processing(const std::vector<double> &curpos, const std::vector<double> &curvel)

- Functionality: Compare the signal strength and angle with the threshold, and decide whether the drone is approaching the victim.

Then give the predicted position of the victim. To get this predicted position, When the absolute value of the beacon angle reaches its maximum(about 90 degrees), the drone has the same x-coordination as the victim, and the y-coordination can be computed by adding the distance to the position where the beacon angle is maximized.

- Input: vector<double> curpos, vector<double> curvel
- Publish to topic: /victimCoord_continous_topic.

bool local_search_strength_only()

- Functionality: By detecting a victim's signal, this will use a two-step gradient descent method , first in x then I y direction until the drone is quite close to the top of the victim, or the search steps reaches a certain limit. Finally, this method will print out to the terminal that the drone is close to the victim, and go back to the global search.

unity_map_record.py

- Functionality: Record the unity_map information and write it into .mat file which can be loaded into Matlab for visualization.

visualization_part.py

`def hill_visualization(self)`

- Functionality: Create a pointcloud of the snow mountain for visualization.

- Publish to topic: /hill_surface1. Type: sensor_msgs/PointCloud

`def spinOnce(self):`

- Functionality: Create the trajectory of the drone and publish the victims' positions.
- Publish to topics:
    - /found_victims. Type: geometry_msgs/PoseStamped
    - /drone_searching_trajectory. Type: nav_msgs/Path

planner.cc

```
bool BasicPlanner::go_to_next(Eigen::Vector3d cur_pos,Eigen::Vector3d
goal_pos ,mav_trajectory_generation::Trajectory* trajectory)
```

- Functionality: This method is for the local planning part, which can plan a trajectory from the current position to a goal position, and the generated trajectory can be published to control the drone to fly to the goal position. Additionally, it can also be used for continue global searching from current position and next nearest global point to end of global searching's position as long as local searching is finished.