exploration\_planning.h: lrae\_planner -> include -> file đích

exploration\_planning.cpp: lrae-> src-> đích

path planning.h: lrae\_planner -> include -> file đích

path planning.cpp: lrae\_planner -> src -> file đích

ultis.h: lrae\_planner -> include -> file đích

simulation.rviz: fitplane -> launch -> rviz\_config -> file đích

demo.rviz: simworld-> launch-> file đích

Cấu trúc của viewpoint được khai báo trong ultis.h

cũng tạo ô lưới bản đồ, cũng có tiêu thuẩn đánh giá viewpoint, cũng có chi phí đi lại // chủ yếu là đặt biến

struct ViewPoint

{

    geometry\_msgs::Point position;  // Vị trí của điểm quan sát trong không gian

    int indexX = 0;  // Chỉ số hàng trong lưới bản đồ

    <!-- //////chưa phân tích việc sử dụng tọa độ trong không gian của cấu trúc viewpoint -->

    int indexY = 0;  // Chỉ số cột trong lưới bản đồ

    int fartherCentroidX = 0;  // Tọa độ X của tâm cụm xa hơn

    <!-- ////// chưa rõ về Centroid này lắm -->

    int fartherCentroidY = 0;  // Tọa độ Y của tâm cụm xa hơn

    float score = 0.0;  // Điểm đánh giá của viewpoint dựa trên tiêu chí nào đó

    int disgridnum = 10000;  // Số lượng ô lưới bị che khuất trên đường đi

   <!-- 1. disgridnum tính toán số lượng các ô lưới bị che khuất khi di chuyển từ ViewPoint đến các vị trí khác. Khá giống với việc chọn viewpoint ở bài toán kia  -->

    int traver\_degree = 100;  // Mức độ di chuyển được (có thể là độ khó khi đi qua)

    double rv\_path\_cost = 0.0;  // Chi phí di chuyển đến viewpoint

};

cấu trúc của viewpoint trong exploration\_planning.h: cấu trúc centroid, đề cập cái gì mà có các cụm bao chùm hay sao ý

struct Centroid

{

    geometry\_msgs::Point centroidposition;

    utils\_ns::Index2 cenindex;

    int unknownnum = 0;

    int windowsoder = 0;

    bool isUnknown;

    std::vector<utils\_ns::ViewPoint> viewpoint\_fir;  // Viewpoints giai đoạn 1

    std::vector<utils\_ns::ViewPoint> viewpoint\_sec;  // Viewpoints giai đoạn 2

    std::vector<utils\_ns::ViewPoint> viewpoint\_fin;  // Viewpoints cuối cùng sau khi lọc

    std::multimap<float, utils\_ns::ViewPoint\*> evaluated\_viewpoints;  // Đánh giá viewpoints

};

<!-- 1. Một cấu trúc Centroid ở đây sử dụng 3 viewpoint, tuy nhiên như bên dưới đang ghi chú, chưa thấy việc sử dụng tới 2 viewpoint đầu  -->

Các bước lấy ViewPoint trong thám hiểm

B1: Lấy danh sách các viewpoints

📍 Trong ExplorationPlanning::getCentroidHelper(), viewpoints được tạo ra bằng cách quét vùng chưa khám phá:

for(int i = ori\_x; i < ori\_x + w; i++)

{

    for(int j = ori\_y; j < ori\_y + h; j++)

    {

        if(isInBorder(i, j) && global\_map\_.data[i + j \* width\_] == -1)

        {

            tmp\_viewpoint.indexX = i;

            tmp\_viewpoint.indexY = j;

            tmp\_viewpoint.position = index2coord(i, j);

            tmp\_cen.viewpoint\_fir.push\_back(tmp\_viewpoint);

        }

    }

}

Trong file exploration\_planning.cpp, hàm getCentroidHelper() sẽ quét bản đồ và tìm các ô lưới chưa khám phá (-1).

Những ô này sẽ được lưu thành ViewPoint và được thêm vào viewpoint\_fir.

<!-- 2. viewpoint được tạo ra bằng cách quét ra những vùng chưa khám phá, nắm hơn về code

Trong file exploration\_planning.cpp, hàm getCentroidHelper() sẽ quét bản đồ và tìm các ô chưa khém phá (-1)

Những ô này được lưu thành Viewpoint và được thêm vào viewpoint\_fir -->

Bước 2: Đánh giá và lọc viewpoint

📍 Trong ExplorationPlanning::assessUnknownRegion(), các viewpoints được đánh giá dựa trên:

Chi phí đường đi (rv\_path\_cost).

Mức độ khó di chuyển (traver\_degree).

Số lượng ô lưới bị che khuất (disgridnum).

<!--3. thông số được sử dụng để đưa ra tối ưu-->

choosecentroid->evaluated\_viewpoints.insert(

    make\_pair(choosecentroid->viewpoint\_fin[j].score, &choosecentroid->viewpoint\_fin[j])

);

s\_1 = max\_rvpathcost > 0 ? float(choosecentroid->viewpoint\_fin[j].rv\_path\_cost / max\_rvpathcost) : 0;

s\_2 = max\_traver\_degree > 0 ? float(choosecentroid->viewpoint\_fin[j].traver\_degree / max\_traver\_degree) : 0;

s\_3 = max\_disgridnum > 0 ? float(choosecentroid->viewpoint\_fin[j].disgridnum / max\_disgridnum) : 0;

choosecentroid->viewpoint\_fin[j].score = 1 \* s\_1 + s\_2 + s\_3;

<!-- 4. tính điểm thông qua cả 3 thông số -->

<!--5. choose centroid đươc lưa chọn dựa trên cơ sở của viewpoint\_fin, tính thông qua ba thông số -->

<!-- /////có 3 bước tạo viewpoint từ fir, sec rồi đến fin, bước chọn viewpoint lấy từ score của viewpoint final, vậy tác dụng của 2 bước trước đó là gì? -->

target\_viewpoint1.push\_back(\*centroid1.evaluated\_viewpoints.begin()->second);

<!-- 5.1. giữ lại viewpoint tốt nhất về hệ thống lập kế hoạch -->

<!-- //////chưa clear lắm về hàm centroid1.evaluated\_viewpoint.begin() -->

path\_planner\_.setViewPointSet(target\_viewpoint1, target\_viewpoint2, true, true);

<!-- 5.2. từ target\_viewpoint1, target\_viewpoint2, ta sử dụng để lập path\_planning -->

<!-- //////làm rõ target\_viewpoint2 -->

Bước 3: Lọc viewpoint tốt nhất

📍 Trong ExplorationPlanning::purgeViewpoint(), chỉ những viewpoint có giá trị tốt nhất mới được chọn.

while(count < 6 && !emptyFlag)

{

    if(!tmpCentroid.evaluated\_viewpoints.empty())

    {

        auto current\_viewpoint = tmpCentroid.evaluated\_viewpoints.begin();

        purgedViewpoints.push\_back(\*current\_viewpoint->second);

        tmpCentroid.evaluated\_viewpoints.erase(current\_viewpoint);

    }

    count++;

}

🔹 Chỉ giữ lại một số lượng nhỏ viewpoint quan trọng nhất (tối đa 6).

Sử dụng ViewPoint trong PathPlanning (path\_planning.h, path\_planning.cpp)

Trong path\_planning.h, ViewPoint được sử dụng để lập kế hoạch đường đi:

void setViewPointSet(std::vector<utils\_ns::ViewPoint> viewpointset1,

                     std::vector<utils\_ns::ViewPoint> viewpointset2,

                     bool viewpoint1has, bool viewpoint2has);

🔹 Dùng để truyền danh sách viewpoints vào bộ lập kế hoạch đường đi.

📍 Trong path\_planning.cpp, viewpoint được sử dụng trong hàm:

if(!path\_planner\_.getViewPointSet1().empty())

{

    for(const auto &it: path\_planner\_.getViewPointSet1())

    {

        choose\_viewpoints\_marker\_->marker\_.points.push\_back(it.position);

    }

}

ViewPoint được sử dụng trong quá trình thám hiểm và lập kế hoạch đường đi để tìm ra các vị trí quan sát tốt nhất.

ViewPoint được tạo từ bản đồ, sau đó được đánh giá và lọc ra những điểm tối ưu nhất.

Các ViewPoint được truyền vào PathPlanning để lập kế hoạch di chuyển đến những vùng chưa khám phá.

<!-- 6. Tổng quan chung

 tinh chỉnh hoặc tối ưu thuật toán, hướng làm chính:

Thay đổi cách đánh giá score trong assessUnknownRegion(). -> có thể thay đôỉ cách tích  giá trị hoặc khai báo thêm biến giới hạn khoảng cách tối thiểu như trong bài báo

-> xác định được việc có lựa chọn viewpoint thông qua khả năng quét của viewpoint (bài này hơi khác ở chỗ số ô bị che khuất càng ít thì càng thể hiện viewpoint tốt), bản chất vẫn tương đồng với bài kia vì bài kia thì không tính tổng số ô bị che khuất mà xác định trực tiếp bằng số ô có thể quét được tại 1 vị trí viewpoint cụ thể

Giới hạn số lượng viewpoint trong purgeViewpoint().

.............Sử dụng thuật toán khác để chọn viewpoint tốt hơn....... -->