Universidad
Rey Juan Carlos

Mobile Robotics (IV). Global Navigation.

Cristian Sánchez Rodríguez

47584641X

Course 2021/2022

Introduction.

In this project, we will solve the autonomous navigation problem.

To do it, we use gradient path planning (GPP). This algorithm solve localization problems, related to following a single and strict path, as we see in the images below:

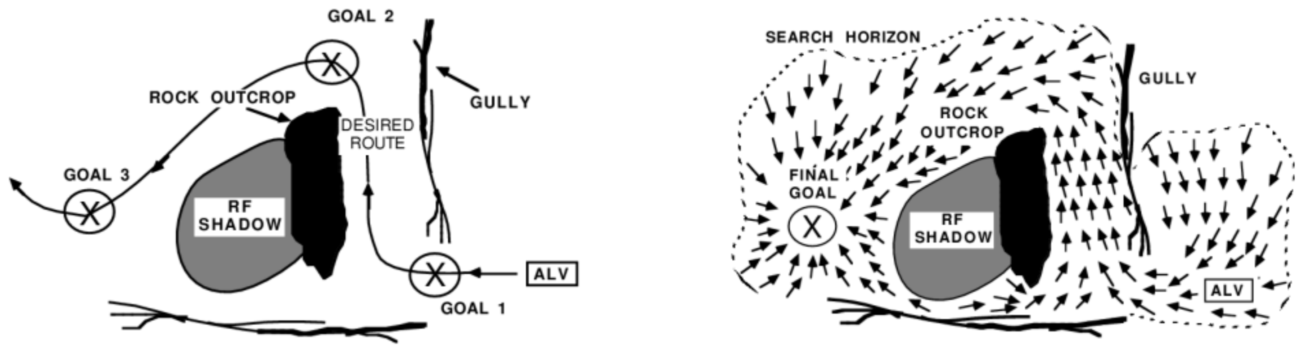

Fig 1. Classic plan and GPP plan.

Algorithm

At the beginning, we need to build a gradient map to assign weights to every cell between the start point and the goal.

Every weight represents the cost of choosing a direction inside de gradient:

- Cardinal directions → +1
- Diagonal directions → +1.25 (typically sqrt(2) but it's better to reduce this value).

So first, we get the map and we binarize it to obtain only two values, 0 (obstacle) or 255 (road). Then, we copy this into a numpy array of zeros to get the base where we will fill with the expansion.

Wave Front Algorithm (WFA)

As we said before, we need to fill our gradient array, so we use a derivation of the Bread First Search algorithm, as it's shown in the following steps:

1. We store initial state and cost in the frontier (queue).
2. Pop from frontier node
3. ¿Is the final state? → END
4. ¿Is it an obstacle? → Store in the obstacle list (for wall expansion) and go to step 2.
5. Get neighbors, assign weights → push into the frontier (cardinal > diagonal priority). Go to step 2.

If a value is assigned, we put the lower one. If the final state is reached, we iterate extra times to obtain some space for the robot maneuverability.

Finally we expand the obstacle list (walls), taking into account all directions and corners. This is to avoid risky behaviors near walls.

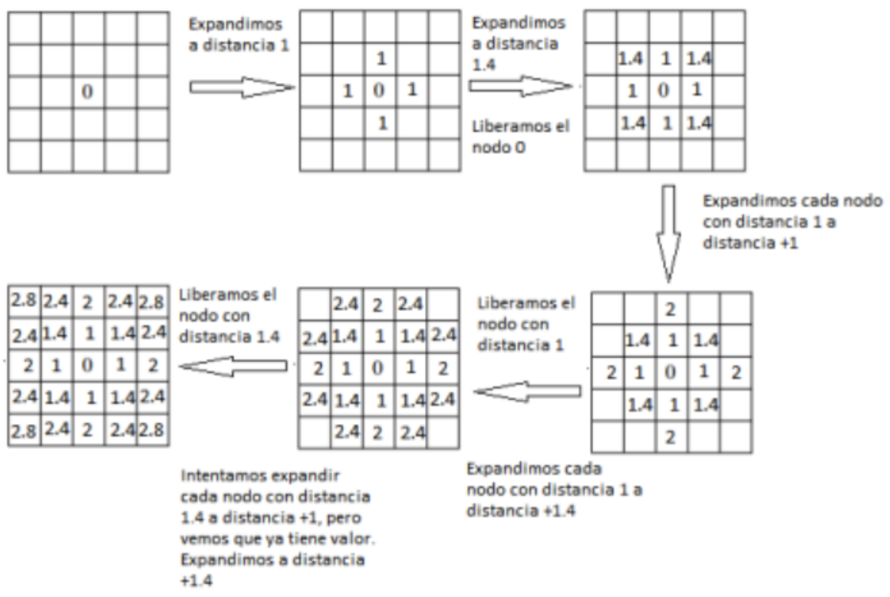
Expandimos a distancia 1
Expandimos a distancia 1.4
Liberamos el nodo 0
Expandimos cada nodo con distancia 1 a distancia +1
Liberamos el nodo con distancia 1
Expandimos cada nodo con distancia 1 a distancia +1.4
Liberamos el nodo con distancia 1.4
Intentamos expandir cada nodo con distancia 1.4 a distancia +1, pero vemos que ya tiene valor. Expandimos a distancia +1.4

Fig 2. WFA evolution theory.

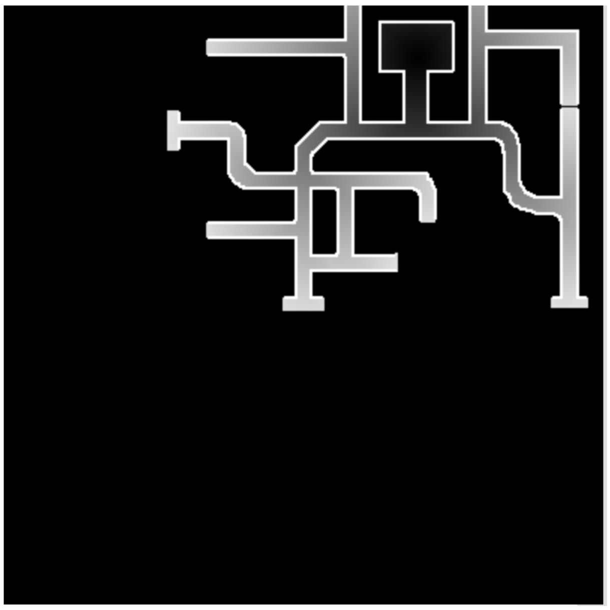

Fig 3. WFA evolution practice.

Path extraction

Next step is to get a valid path. For this task, we iterate over our filled array, from the initial pose, following gradient descent, until the target pose is reached, where it's value should be equal to zero.

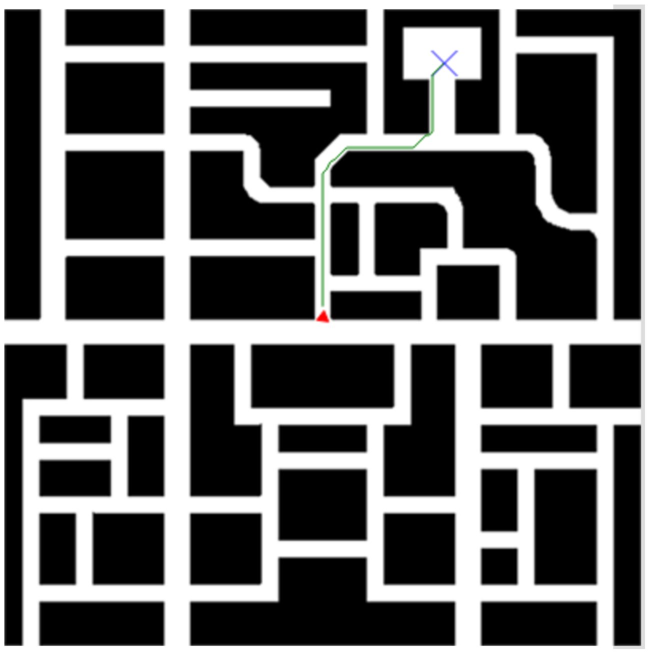

Fig 4. Path extraction.

Reaching the goal

Last part is to command the car motors, following the path until the target position is reached.

This is done using a simplified version of the VFF algorithm implemented in the obstacle avoidance project.

In this particular case, we get the sub objective sequence from the path and we only use attractive forces. This is because obstacles (walls) are taken into account during route fetching.

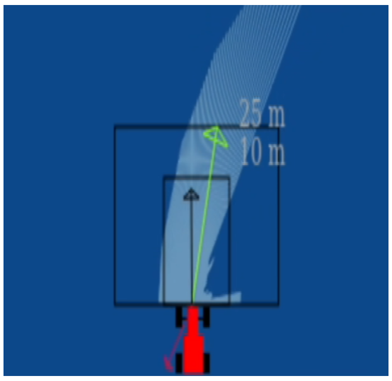

Fig 5. VFF implementation.

Conclusions and comments.

In conclusion, this project involves lots of mobile robotics concepts. Using GPP to global navigate is a solid solution, because classic plans used to be less adaptive through dynamic scenarios.

Some aspects remain to be polished, like bounds access problems, array dtype codification for long distances (if values above 255 are reached, it restarts from 0 [uint8]) or inconsistencies getting the path, like infinite loops where the final position is never reached.

Anyway, using the theory correctly, we are able to get optimal paths and lead the car to our target positions autonomously.