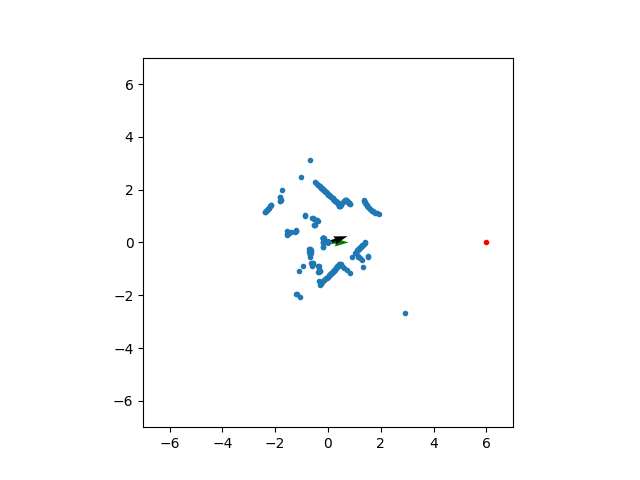
Mise en situation

Pour continuer l’évitement d’obstacle on va maintenant s’intéresser à la recherche de chemin. Ayant eu un cours qui portait sur le sujet, il était plus aisé d’aborder ce sujet.

Le point de départ est le même que pour l’évitement via méthode des potentiels : on part d’une mesure retournée par le Lidar. Ici on s’appuiera sur la mesure ci-dessous. L’orientation du robot est représentée par la flèche verte, le point à joindre est en rouge, et la flèche noire représente l’orientation que doit prendre le robot après traitement de l’algorithme que je vais décrire ci-après.



Filtrage, discrétisation et recherche de plus court chemin

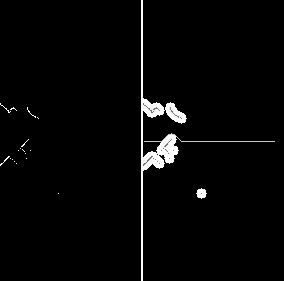
On applique tout d’abord un filtre sue les mesures du lidar. On enlève les points trop près (distance inférieure à 0.2m) pour éviter que le robot ne voit certaines de ces antennes ou les faisceaux qui n’ont rien retournés (qui prennent une distance zéro par défaut). Ainsi, l’amas de points au centre est gommé. Ensuite on enlève tous les points qui sont situés derrière le robot, car ils ne sont pas utiles.

Maintenant que les données sont correctement filtrées, on crée une grille qui va discrétiser l’espace, c’est-à-dire un maillage plus ou moins fin que l’on va appliquer sur la figure précédente filtrée. Ainsi, chaque maille qui contient un point Lidar prend la valeur ‘1’, les autres prennent la valeur ‘0’. On obtient ainsi la fig2 à gauche. En observant bien, la figure contient exactement les points situés devant le robot et qui ont été converti en blanc. On va également appliquer un traitement sur cette matrice. En effet, le robot possède une certaine longueur et largeur et les obstacles sont représentés par des points de la taille d’une maille. C’est pour cela qu’il faut élargir les obstacles, pour condamner une zone autour de chaque point. La dimension de cette dilatation (c’est le nom de cette transformation en traitement d’image) est donnée par la relation suivante

Dilat = int(pas\_grille \* largeur\_robot/2)

Ainsi, si deux obstacles sont distants de moins de la largeur du robot, alors leurs deux disques de dilatation vont se joindre, et la possibilité pour le robot de passer entre ces deux points est condamnée. (fig2 à droite)

On s’intéresse ensuite à la recherche du plus court chemin entre notre point de départ et notre point d’arrivée. Il existe plusieurs algorithmes de ce type, ici le choix retenu est l’algorithme Astar, aussi appelé A étoile en français, qui est très utilisé dans le milieu de la robotique pour ce cas de recherche de chemin (ref). Après un cours temps d’exécution, l’algorithme a trouvé dans la grille le chemin



Réflexion sur le résultat

L’algorithme renvoi bien un chemin qui est possible à parcourir par le robot. Il peut paraitre logique d’aller en ligne droite vers la cible et d’éviter un obstacle dès qu’il s’en présente une. Cependant, une simple observation permet de voir qu’un segment tangent entre le point de départ et le haut du mur (fig3 en orange, image précédente agrandie pour exprimer mon propos) est plus rapide. Cela vient du fait de la discrétisation. Chaque case de la grille possède huit voisins, et il n’existe pas d’autre possibilité. Cette perte de continuité induit donc une perte d’information. L’algorithme renvoie bien un plus court chemin, mais qui est dépendant de sa métrique



C’est pourquoi avec un post traitement du chemin retourné par l’algorithme est requis, afin de récupérer cette ‘diagonale’ orange. Ainsi, on obtient un nouveau cap qui s’appuie sur cette diagonale, symbolisée par la flèche noire de la figure1

Référence Astar :

[csis.pace.edu/~benjamin/teaching/cs627/webfiles/Astar.pdf  
A\* Pathfinding for Beginners By Patrick Lester (Updated July 18, 2005)](http://csis.pace.edu/~benjamin/teaching/cs627/webfiles/Astar.pdf)