Objectifs

Ce troisième TP comporte trois objectifs :

- Prendre en main le télémètre laser embarqués sur le Thymio
- Réaliser des changements de repère pour re-projeter les données
- Programmer une localisation par ICP

A rendre:

Vous devez rendre sur Ecampus un compte rendu de maximum 10 pages contenant votre travail et votre analyse au plus tard le **jeudi 1 Février 23h59**. Le rapport doit est à réaliser en monôme et à rendre au format PDF. Vous joindrez au dépôt le fichier python de votre contrôleur.

I. Télémètre laser embarqué

Dans webots, votre Thymio embarque un télémètre laser (non visible). Afin de récupérer ces données, vous devez :

- Récupérer le capteur et l'initialiser :

```
lidar = robot.getDevice('lidar')
lidar.enable(timestep)
lidar.enablePointCloud()
```

- Récupérer le nuage de point à chaque instant :

```
point_cloud = lidar.getRangeImage()
```

- Le télémètre laser envoie un tableau contenant l'ensemble des distances mesurées. La première donnée correspond à l'angle 0, la seconde 0+1*2Pi/lidar.getHorizontalResolution() , la troisième 2*2Pi/lidar.getHorizontalResolution() .

Ces coordonnées sont au format « polaire ». On pourra les convertir aisément en utilisation le script suivant :

```
angle = 0
for i in point_cloud:
    xy = [i*math.sin(angle), 0, i*math.cos(angle)]
    angle+= 2*math.pi / lidar.getHorizontalResolution()
```

1. Programmer la récupération des données du télémètre laser
2. Afficher les données sous forme « xy » (on vérifiera l'angle 0 ...)
3. Ajouter les murs sur la figure et vérifier la cohérence des données

Les données du télémètre laser sont exprimés dans la mobile robot. Afin de les afficher dans la carte globale, il est nécessaire de faire un changement de repère Robot → Global pour l'ensemble des données.

On définit la fonction permettant de réaliser ce changement de repère :

```
def multmatr(X,Y,T):
    res = []
    res.append( X[0] * Y[0] + X[3] * Y[1] + X[6] * Y[2] - T[0])
    res.append( X[1] * Y[0] + X[4] * Y[1] + X[7] * Y[2] + T[1])
    res.append( X[2] * Y[0] + X[5] * Y[1] + X[8] * Y[2] + T[2])
    return res
```

On appliquera ce changement à l'ensemble des points du télémètre :

```
node = robot.getFromDef("Thymio"); % A déclarer avant la boucle « while »
angle = 0
rotation = node.getOrientation()
xyz = node.getPosition()
for i in point_cloud:
        xy = [i*math.sin(angle), 0, i*math.cos(angle)]
        pt = multmatr(rotation,xy,xyz)
        angle+= 2*math.pi / lidar.getHorizontalResolution()
```

4. Commenter ce code dans votre rapport

On affichera les points du télémètre laser sur la carte globale. Lorsque votre robot bouge, vous devait voir votre nuage de point coller au mur.

5. Programmer ce changement de base et vérifier la cohérence de vos résultats

II. Télémètre laser et localisation

On souhaite maintenant coupler la localisation par odométrie (TP 2) et l'utilisation de notre télémètre laser.

6. Reprendre votre algorithme de localisation par intégration des données odométriques
7. Projeter les données lasers en utilisant comme point de repère robot la localisation calculée par votre algorithme. Vous devez observer une dérive progressive du nuage de point.

L'objectif est de pouvoir recaler le nuage de point grâce à la carte de l'environnement pour ensuite recaler la localisation.

On se propose d'utiliser les fonctions suivantes :

```
def indxtMean(index,arrays):
        indxSum = np.array([0.0, 0.0 ,0.0])
        for i in range(np.size(index,0)):
                indxSum = np.add(indxSum, np.array(arrays[index[i]]), out = indxSum ,casting = 'unsafe')
        return indxSum/np.size(index,0)


def indxtfixed(index,arrays):
        T = []
        for i in index:
                T.append(arrays[i])
        return np.asanyarray(T)


def ICPSVD(fixedX,fixedY,movingX,movingY):
        reqR = np.identity(3)
        reqT = [0.0, 0.0, 0.0]

        fixedt = []
        movingt = []
        for i in range(len(fixedX)):
                fixedt.append([fixedX[i], fixedY[i], 0])
        for i in range(len(movingX)):
                movingt.append([movingX[i], movingY[i], 0])
        moving = np.asarray(movingt)
        fixed = np.asarray(fixedt)
```

```
n = np.size(moving,0
TREE = KDTree(fixed)
for i in range(10):
        distance, index = TREE.query(moving)
        err = np.mean(distance**2)

        com = np.mean(moving,0)
        cof = indxtMean(index,fixed)
        W = np.dot(np.transpose(moving),indxtfixed(index,fixed)) - n*np.outer(com,cof)
        U , _ , V  = np.linalg.svd(W, full_matrices = False)

        tempR = np.dot(V.T,U.T)
        tempT = cof - np.dot(tempR,com)

        moving = (tempR.dot(moving.T)).T
        moving = np.add(moving,tempT)
        reqR=np.dot(tempR,reqR)
        reqT = np.add(np.dot(tempR,reqT),tempT)
```

La fonction ICP prend comme paramètres les coordonnées x,y des points fixes (les murs dans notre cas) et les coordonnées xy des points mobiles (télémètres laser) que l'on souhaite recaler avec sur les murs. Elle calcule la transformation (rotation + translation) pour replacer au mieux les données télémètres sur les murs.

8. Commenter la fonction ICP
9. Le fonction ICP ne prend pas comme paramètre des segments de mur. Il faut discrétiser les murs sous forme de point par pas de 1 cm. Créer ce nuage de point puis afficher le.
10. Utiliser la fonction ICP pour recaler le nuage :

    - Afficher le nuage dans sa nouvelle position

    - Afficher la transformation (rotation + translation) qui a été appliqué pour recaler le nuage

La transformation nécessaire pour recaler les deux nuages correspond à l'erreur réalisée par l'algorithme de localisation par odométrie.

11. Calculer la transformation toutes les 5 secondes. Le but est de laisser le temps à l'odométrie de dériver …
12. Appliquer la transformation pour recaler la localisation du robot
13. Comparer la localisation avec/sens recalage en fonction du temps
14. Évaluer votre algorithme de localisation et essayer de le mettre en défaut.
15. Modifier votre algorithme pour ne plus utiliser de données odométriques.
16. Qualifier ce nouvel algorithme qui n'utilise que le télémètre laser. On comparera vos 3 méthodes de localisation