TP3 C3 Télémétrie laser et localisation

Mathieu Oriol

I. Télémètre laser embarqué

La fonction `proj_point_cloud` permet à partir du nuage de points du laser de le projeter dans le repère du robot (`xy = [p * math.sin(angle), p * math.cos(angle)]`), puis dans le repère principal, en utilisant les données de position du robot venant de différentes sources (odométrie ou simulateur) et d'appliquer éventuellement une correction avec les données de l'ICP.

```
def proj_point_cloud(point_cloud, repere_proj="odo", correction=False):
    angle = math.pi
    point_cloud_xy = np.zeros((len(point_cloud), 2))
    for i, p in enumerate(point_cloud):
        xy = [p * math.sin(angle), p * math.cos(angle)]
        point_cloud_xy[i] = base_robot2main(xy, repere_proj=repere_proj,
correction=correction)
        if np.isnan(point_cloud_xy[i][0]):
            point_cloud_xy[i][0] = 0.
        if np.isnan(point_cloud_xy[i][1]):
            point_cloud_xy[i][1] = 0.
        angle += 2 * math.pi / HORZ_RES
    return point_cloud_xy
```

Le code de changement de base a été modifié pour travailler avec des coordonnées 2D. Il fait une rotation puis une translation :

```
def base_robot2main(p, repere_proj="odo", correction=False):
    """Changement de base du robot vers la base principal de la scene"""
    # Apply the correction
    proj = p
    if correction:
        proj = np.dot(corr_R, proj) + corr_T
    if repere_proj == "simu":
        x, y, theta = pos_robot()
    elif repere_proj == "odo":
        x, y, theta = Xk
        theta = -theta + math.pi / 2
    else:
        raise ValueError
    proj = np.array(
        [
            math.cos(theta) * proj[0] + math.sin(theta) * proj[1] + x,
            -math.sin(theta) * proj[0] + math.cos(theta) * proj[1] + y,
        ]
    )
    return proj
```

On peut ensuite afficher le nuage de point du lidar :

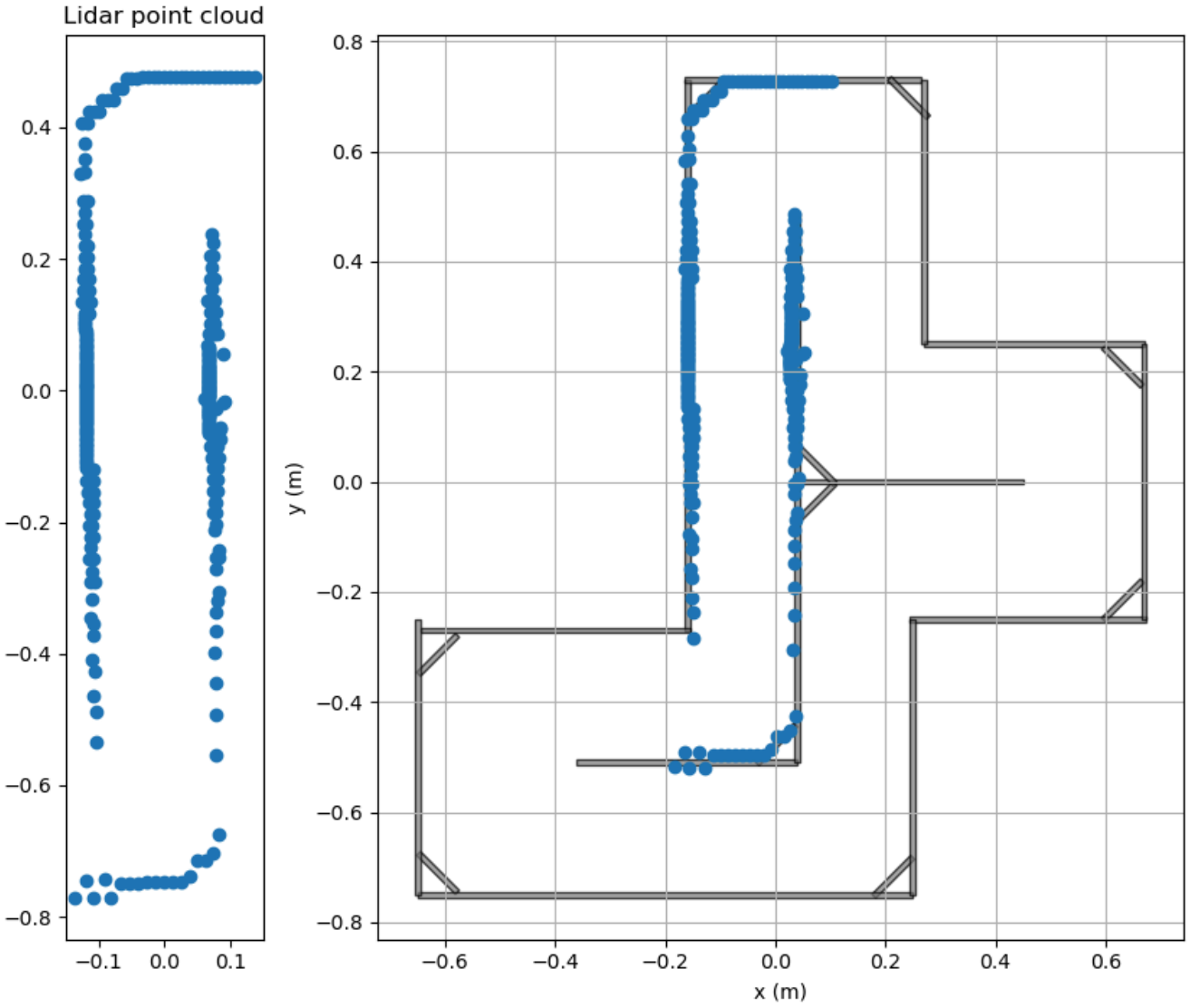

Fig. 1. – Affichage du nuage de point en utilisant les données du simulateur pour la projection

II. Télémètre laser et localisation

Lorsque l'on utilise les données de l'odométrie, on observe que les nuages divergent progressivement.

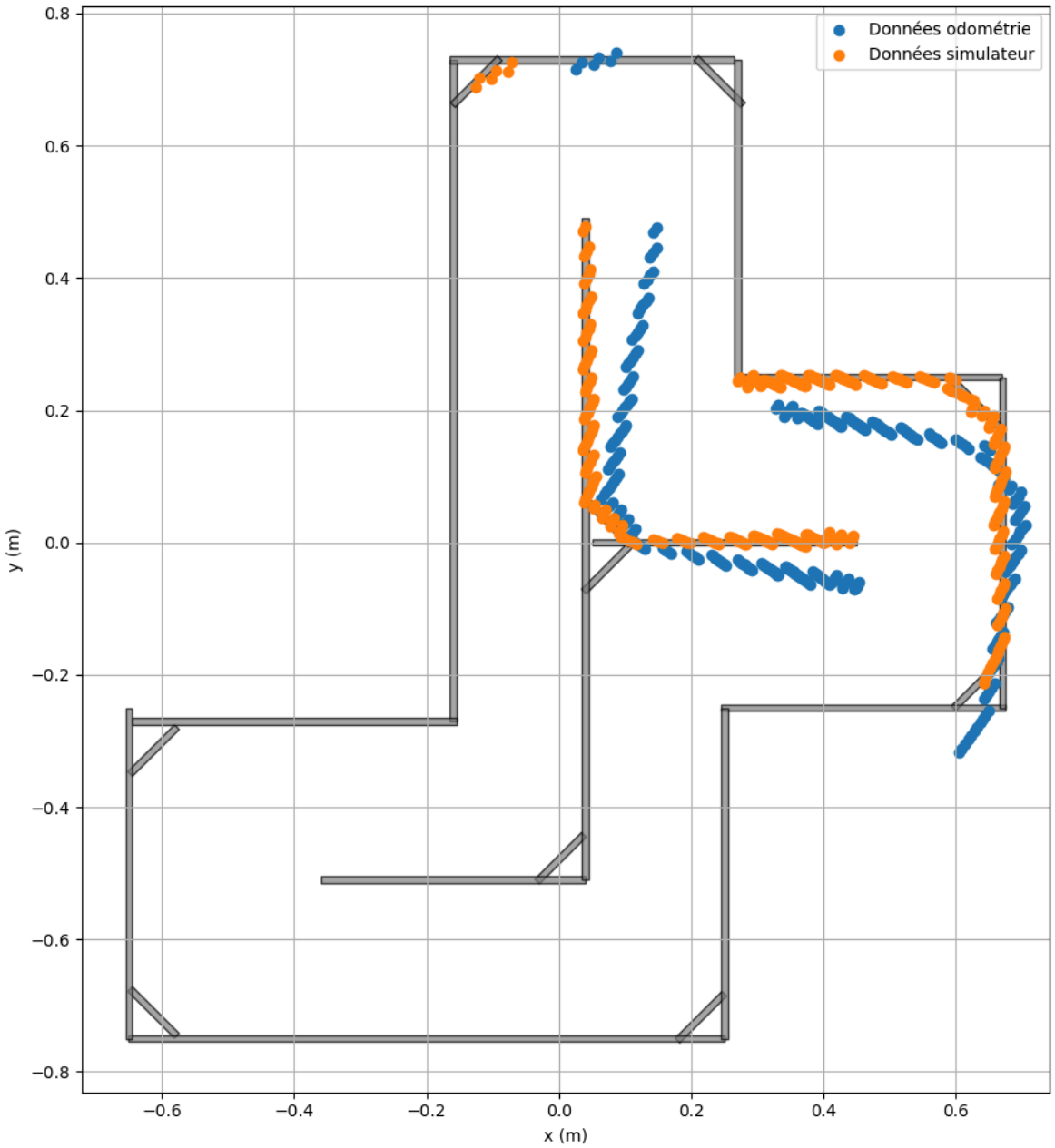

Fig. 2. – Dérive proressive des nuages

Pour les murs, j'ai repris leur définition dans la configuration de webots, afin d'avoir leur position précise. La fonction `walls2points` permet de les discrétiser.

```
def walls2points(walls):
    points = []
    for w in walls:
        tx, ty, _ = w["translation"]
        sx, sy, _ = w["size"]
        angle = w.get("rotation", 0)
        if sx ≥ sy:
            for i in range(math.floor(sx * 100)):
                points.append(
                    np.array(
                        [
                            math.cos(angle) * (-sx / 2 + i / 100) + tx,
                            math.sin(angle) * (-sx / 2 + i / 100) + ty,
                        ]
```

```
                )
            )
    else:
        for i in range(math.floor(sy * 100)):
            points.append(
                np.array(
                    [
                        -math.sin(angle) * (-sy / 2 + i / 100) + tx,
                        math.cos(angle) * (-sy / 2 + i / 100) + ty,
                    ]
                )
            )
return np.array(points)
```

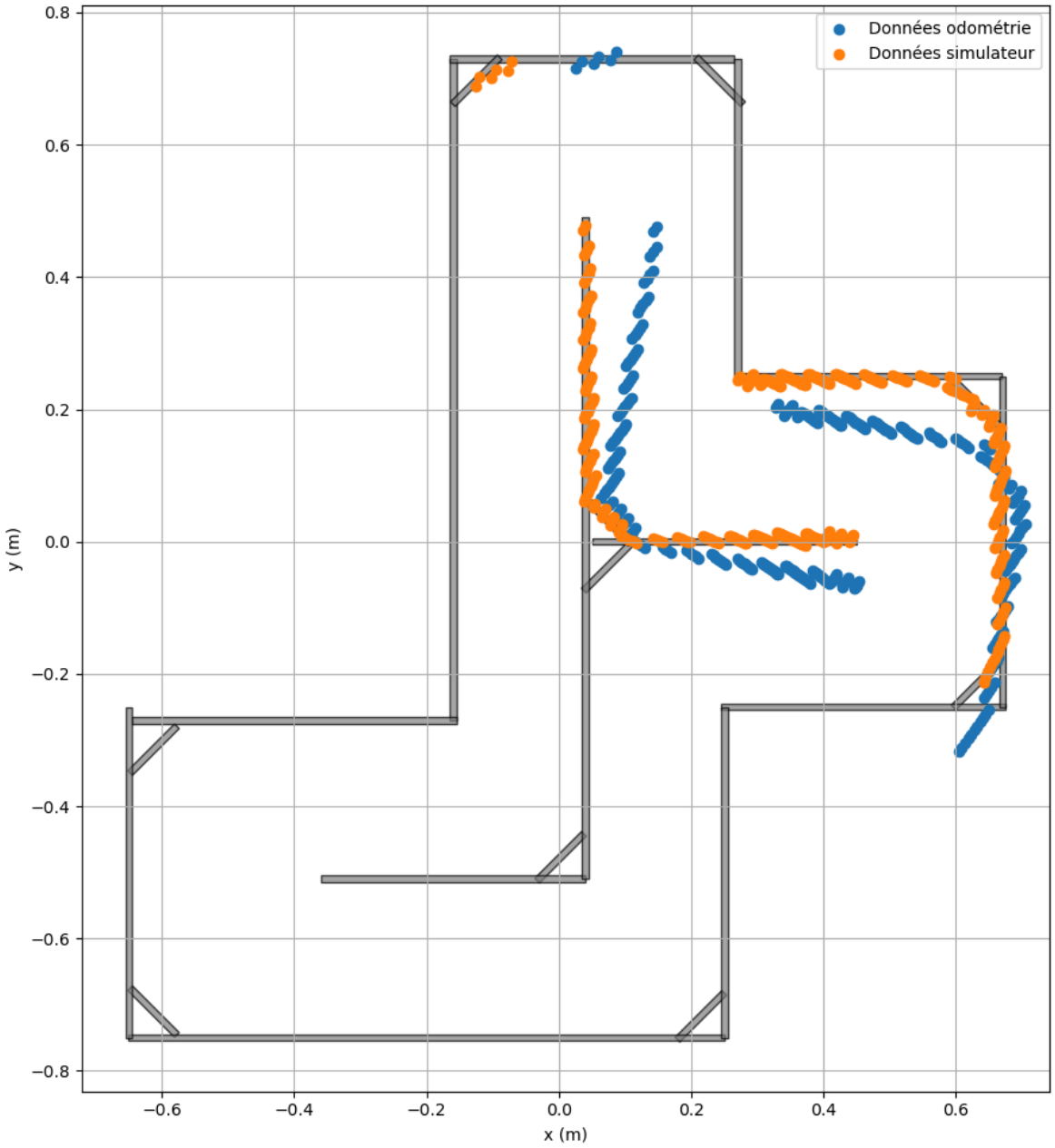

Fig. 3. – Murs discretisés

En appliquant la correction de l'ICP au nuage, on obtient un bien meilleur positionnement qu'avec seulement l'odométrie.

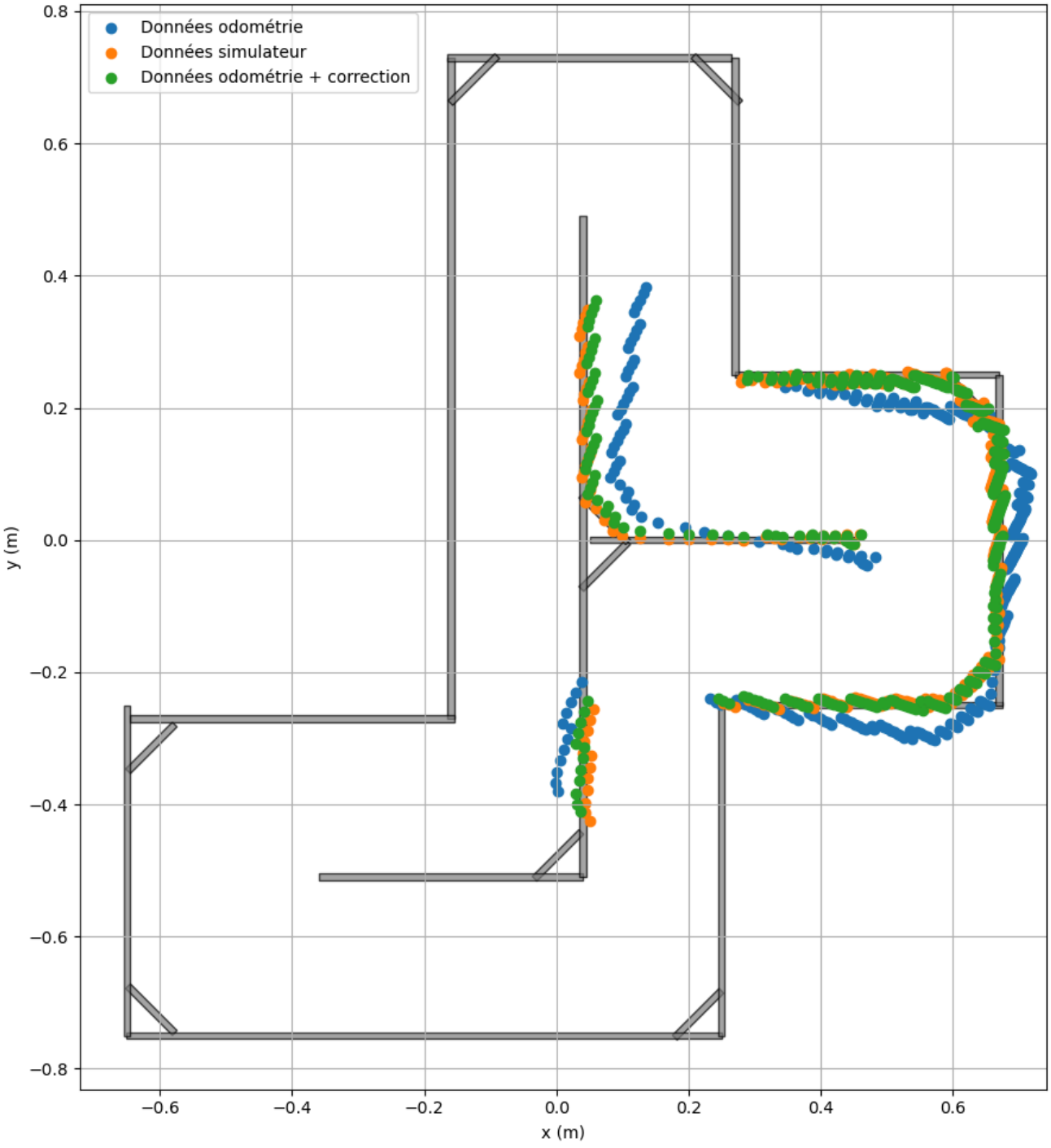

Fig. 4. – Correction de la position du nuage avec l'ICP

On peut aussi utiliser ces données pour recaler la localisation par odométrie avec la fonction `apply_corr`.

```
def apply_corr():
    """Apply the telemetry corrections to the odometry"""
    Xk[0] += corr_T[0]
    Xk[1] += corr_T[1]
    Xk[2] -= matRot2angle(corr_R)
```

La fonction d'ICP est lente à converger et il faut parfois plusieurs recalibrages avant de retrouver la bonne position (notamment si l'on se prend un mur). On peut même se retrouver bloquer si les corrections à effectuer sont importantes.

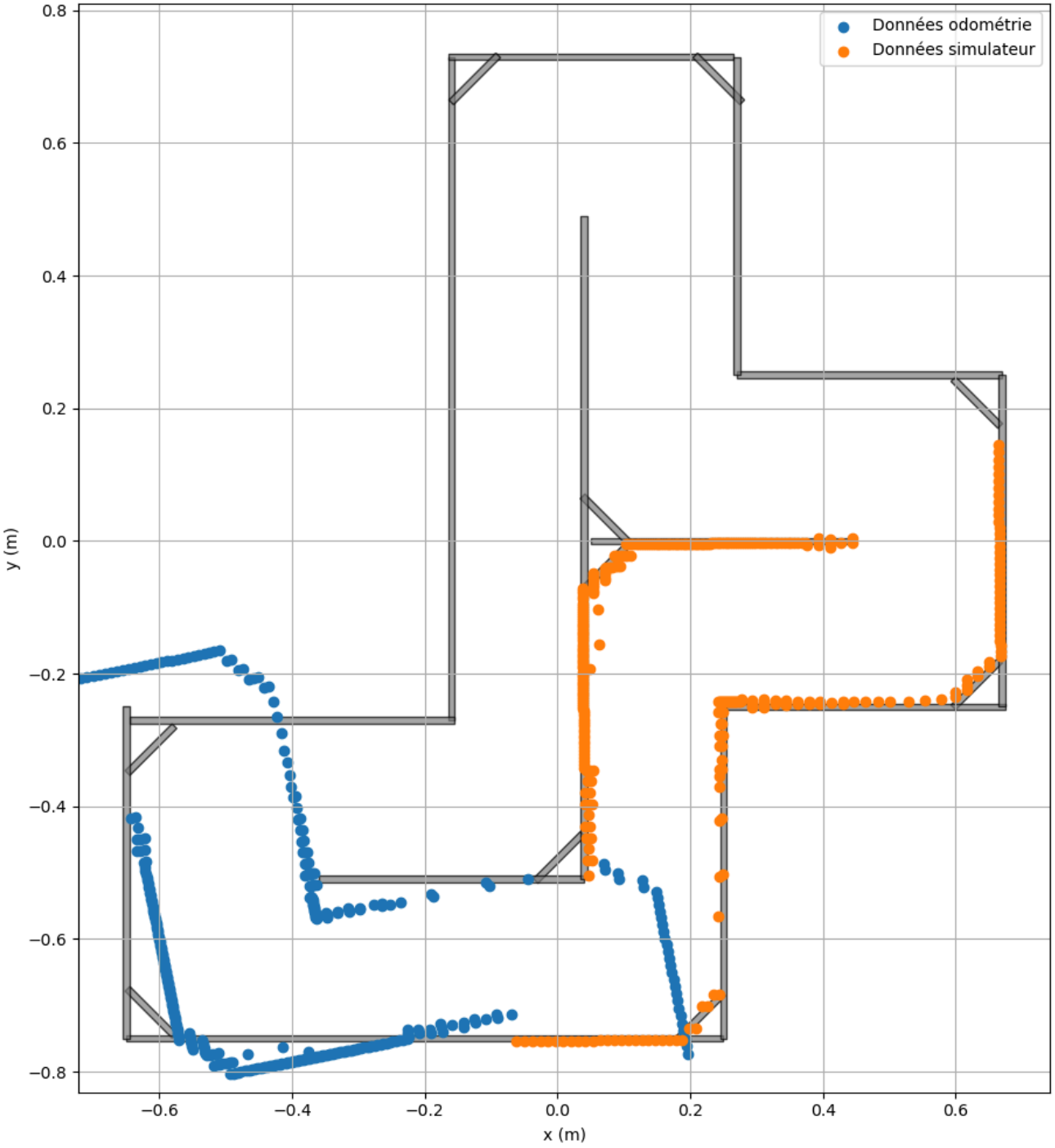

Fig. 5. – Mise en defaut de l'algorithme

Pour la solution utilisant uniquement la télémétrie, on peut considérer le recalage à effectuer comme la position du robot. Mais mes essais n'étaient pas concluants.