RAPPORT

A. Présentation du projet

Au début de la réalisation de notre projet, notre objectif était de réaliser un programme de NAO pour le ramassage de déchets (boule de papier, une canette) et le mettre dans la poubelle correspond (noir, bleu ...); Malheureusement, pendant la réalisation du projet on a rencontré des difficultés qui nous ont empêchés d'atteindre cet objectif.

Afin de réaliser ce projet on a divisé le projet en deux parties indépendantes:

- ➔ **la première partie consiste à :**
  - reconnaître l'objet en utilisant la caméra de NAO et se déplacer vers l'objet.
  - porté l'objet.
- ➔ **La deuxième partie permet de:**
  - Recherche d'un poubelle et déplacement autonome
  - Déplacement vers la poubelle et mettre dedans l'objet

B. Documentation

1. Installation de logiciel :

Notre programme n'est pas installé par défaut sur le robot, c'est-à-dire ce n'est pas un programme qui s'exécute automatiquement au démarrage de NAO.
Nous n'avons pas réussi à installer notre programme comme programme par défaut car nous n'avons pas pu rendre notre programme installable sur un robot. Donc pour tester, il est nécessaire, une exécution de la classe main et il faudrait que le robot et le machine qui exécute se trouvent sur le même réseau .

2. Les Fonctionnalités

2.1. reconnaître l'objet en utilisant la caméra de NAO et se déplacer vers l'objet.

Dès le lancement de programme le NAO cherche autour de lui si l'objet bouteille existe. Il peut aussi détecter la présence des personnes et aussi d'autres objets .

Le NAO se dirige vers la première bouteille détectée. S'il ne détecte aucune bouteille, il tourne autour de lui. Si rien n'est détecté pendant 5 tours, il s'assoit et le programme se termine.

2.2. porté l'objet.

une fois le NAO arrive à sa destination, il se met a une position qui lui permet de prendre l'objet avec ses deux mains.

2.3. Recherche et déplacement autonome

Une fois le Nao à trouver l'objet (déchet), il est le temps qu'il doit chercher la poubelle afin qu'il mette l'objet dans cette poubelle. Donc nous avons décidé de mettre en place une procédure de recherche qui consiste d'abord de faire tourner le NAO autour de lui en cherchant la poubelle et bien sûr en se déplaçant équilibrent sans bouger les mains.

Si la poubelle n'est pas toujours repérée, le NAO se déplace 1 mètre en avance tout en évitant les obstacles et recommence à chercher la poubelle.

2.4. Déplacement vers la poubelle

Pour que le NAO repère la poubelle, l'une des premières solutions a été de faire apprendre la poubelle au NAO, cependant il lui était difficile de le reconnaître (camera, distance), nous avons donc trouver une autre méthode. Après l'étude de la documentation fournie par Aldebaran, nous avons découvert que le NAO avait des marques préenregistrées appelées Naomarks, pour cela on a décidé de coller une marque sur la poubelle.
il en existe plusieurs, comme suit :

(a)

Chaque Noamark possède une valeur reconnue par le NAO, cependant nous n'avons pas utilisé ces valeurs. Nous avons utilisé ces symboles comme marqueur de la poubelle.

Mais malheureusement comme indiqué dans les documents Aldebaran , le module de reconnaissance visuelle offre une approche simple pour percevoir l'environnement qui est simple mais présente plusieurs défis. Certains défis ne nous s'intéressent pas ici mais d'autre oui comme :

- ❏ Pour des performances optimales, la marque NAO doit être dans la ligne de vue directe du robot.
- ❏ la plage de distance pour la détection est de 30 cm à environ 300 cm

Une fois la naomark (poubelle) repérée il faut alors que le NAO se déplace face à celle-ci pour ensuite avancer. Pour cela nous avons décidé de récupérer les coordonnées du poubelle.

Pour obtenir cette coordonnée du marqueur dans le cadre du robot, une transformation du cadre de la caméra au cadre du robot est nécessaire. Cette transformation est effectuée à l'aide des méthodes définies dans la classe de transformation de l'implémentation. Le calcul de cette transformation est présenté comme suite :

$$RobotToLandmark = landmarkToCameraRotationalTransform * landmarkToCameraTranslationalTransform * cameraToRobot$$

Le résultat est une matrice de transformation qui inclut les coordonnées (x, y, z) du repère dans le cadre du robot.

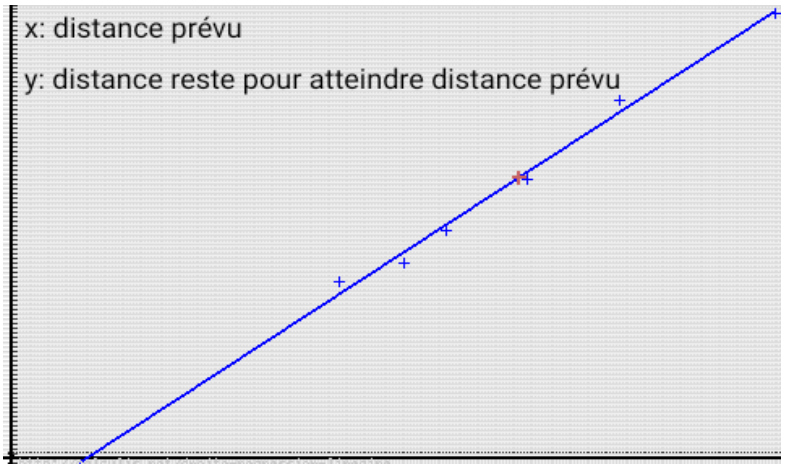
En utilisant les coordonnées (x, y, z) et la méthode **moveTo** nous avons pu aller presque un peu près de la poubelle mais toujours le robot n'allait pas l'endroit prévu c'est-à-dire à côté de la poubelle cela est dû

- La marche du robot car nous avons désactiver les mouvements de main lors de déplacement du robot

- Mais aussi normalement pour que dans le champ de fusion de Nao le Naomark ne disparaisse pas le nao s'arrête environ 30 cm du Naomark.

Pour trouver la longueur de cet écart (la distance manquant) nous avons décidé de faire une petite démonstration qui consiste à prendre des points différents.

x_i	y_i
110	38
132	42
146	49
173	60
204	77
256	96



La droite de régression linéaire admet pour équation : y = 0.4190x-10.9653 donc pour aller au près de la poubelle il voudrais ajouter au longeur 40% de sa taille.

C. Détails d'implémentation

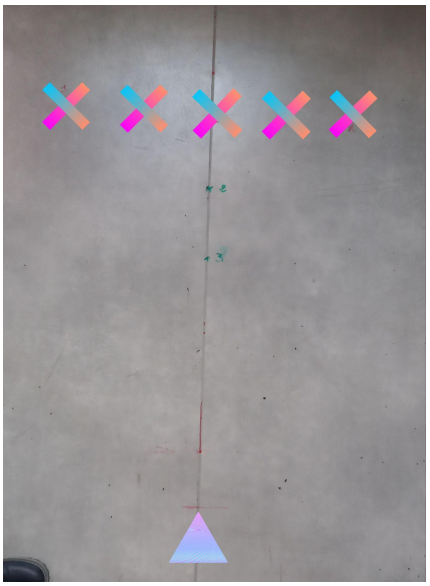
- **reconnaître l'objet en utilisant la caméra de NAO et se déplacer vers l'objet.**

Au début de l'implémentation de cette fonctionnalité on a choisi d'utilisé Choregraphe ; À l'aide du panneau Moniteur vidéo (http://doc.aldebaran.com/2-1/software/choregraphe/tutos/recognize_objects.html) , le NAO peut apprendre des images, des objets et des pièces à reconnaître ,mais cela n'était pas suffisant pour notre projet car il faut que le NAO soit a une distance moins que 50cm de l'objet et le Contexte de l'objet en blanc. Pour cela on a choisi d'implémenter notre propre programme pour la reconnaissance d'objets, la meilleur solution est d'utiliser la librairie imageAI. Cette librairie nous fournit des classes et des fonctions très puissantes mais faciles à utiliser pour effectuer la détection et l'extraction d'objets depuis des images ou des vidéos en utilisant des algorithmes de deep learning comme RetinaNet, YOLOv3 et TinyYOLOv3. Pour notre programme, on a utilisé TinyYOLOv3.
l'une des plus importantes fonction est **detectObjectsFromImage**(), cette fonction retourne un tableau de dictionnaires, chaque dictionnaire correspondant aux objets détecté dans l'image et chaque dictionnaire contient les propriétés suivantes:

*le nom de l'objet
* pourcentage_probabilité qui représente le pourcentage de certitude
* box_points (tuple de coordonnées x1, y1, x2 et y2)

on va utiliser les box_points pour calculer les coordonnées x y de l'objet détecté. et déplacé le NAO en utilisant la fonction moveTo(x, y, 0) .
Pour mieux comprendre l'utilisation de box_points, on vous présente ce Example ci-dessous:

Exemple: On a déplacé la bouteille de gauche à droite de 15 cm et à chaque fois on affiche la sortie de **detectObjectsFromImage()**



(bottle, 74.1584062576294, x1 = 90, y1 = 147, x2 = 110, y2 = 198)
(bottle, 78.27690243721008, x1 = 131, y1 = 140, x2 = 150, y2 =204)
(bottle, 91.98374152183533 x1 = 169, y1 = 146, x2 = 188, y2 = 204)
(bottle, 93.96748542785645, x1 = 211, y1 = 148, x2 = 229, y2 = 201)
('bottle, 88.79823684692383, x1 = 250, y1 = 148, x2 = 273, y2 = 202)

Là on peut bien observé que x1 et x2 représente l'intervalle de y, et on peut calculer la fonction affine f(x) = ax + b qui nous permet de trouver y on utilisant x1 et x2.
la même chose pour x on déplacent la bouteille verticalement on observe un changement dans les y1 et y2 et il suffit de calculer la fonction affine.