NAO

RAMDANI Soukaina

ALI OMAR Ismail

Introduction

Objectif du projet:

réaliser un programme de NAO pour le ramassage de déchets (boule de papier, une canette) et le mettre dans la poubelle correspond (noir, bleu ...)

Introduction

Vidéo

Architecture, conception et gestion de projet

Partie 1:

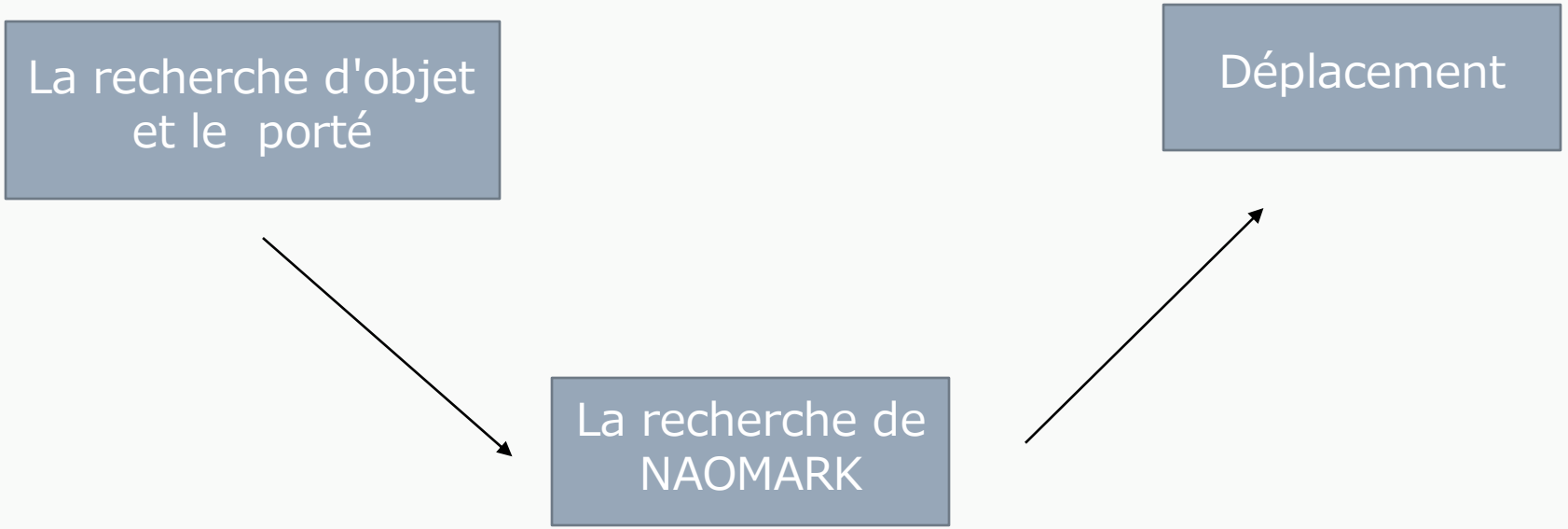
reconnaître l’objet en utilisant la caméra de NAO et se déplacer vers l’objet puis porté l’objet

Partie 2:

La Détection ou la recherche d’une poubelle et mettre l’objet dedans et bien sûr se déplacer d’un manier autonome

Le développement de ces deux partis a duré presque 5 mois.

Architecture, conception et gestion de projet



Architecture, conception et gestion de projet

Compétence technique qui sont nécessaires pour les développer est :

- Langage python
- NAOqi Framework
- NAO robot
- La bibliothèque imageAI

Architecture, conception et gestion de projet

Les difficultés technologiques:

- L'utilisation logiciel Webot (simulateur robotique)
- La limitation du logiciel Choregraphe

Programmation

```
    for eachItem in desc:
        if eachItem["percentage_probability"] >= 30 and eachItem["name"] == 'bottle':
            flag = True
            self.tts.say("I see a " + eachItem["name"])
            x1, y1, x2, y2 = eachItem["box_points"]

            print(eachItem["name"], " : ", eachItem["percentage_probability"])
            x = round(0.0067 * ((y1 + y2)/2) - 0.34, 2)
            y = round((-0.0039) * ((x1 + x2)/2) + 0.7, 2)

            self.motionProxy.moveTo(x, y, 0)
            self.motionProxy.waitUntilMoveIsFinished()
            break
    cv2.imshow("Nao Camera", self.image)
    if turn > 5:
        self.tts.say(" Sorry I didn't found anything")
        self.postureService.goToPosture("Sit", 0.8)
        return False
    if not flag:
        self.motionProxy.moveTo(-0.1, 0.1, math.pi / 3)
        turn = turn + 1
    else:
        self.posToPickUp()
        self.pickUpObject()
        time.sleep(3)
        return True
```

```python
def on_landmark_detected(self, markData):
    """
    Rappel pour l'événement LandmarkDetected.
    """
    if markData == []:  # valeur vide lorsque le landmark disparaît
        self.got_landmark = False
    elif not self.got_landmark:  # ne parlez que la première fois qu'un point de landmark apparaît
        self.got_landmark = True
        self.find = True
        print "I saw a landmark! "
        self.tts.say("I saw a landmark! ")
        # Récupérez la position centrale du landmark en radians.
        wzCamera = markData[1][0][0][1]
        wyCamera = markData[1][0][0][2]
        # Récupérez la taille angulaire du landmark en radians.
        angularSize = markData[1][0][0][3]
        # Calculez la distance jusqu'au landmark.
        distanceFromCameraToLandmark = self.landmarkTheoreticalSize / (2 * math.tan(angularSize / 2))
        # Obtenez la position actuelle de la caméra dans l'espace NAO.
        transform = self.motion_service.getTransform(self.currentCamera, 2, True)
        transformList = almath.vectorFloat(transform)
        robotToCamera = almath.Transform(transformList)
        # Calculez la rotation pour pointer vers le landmark.
        cameraToLandmarkRotationTransform = almath.Transform_from3DRotation(0, wyCamera, wzCamera)
        # Calculez la traduction pour atteindre le landmark.
        cameraToLandmarkTranslationTransform = almath.Transform(distanceFromCameraToLandmark, 0, 0)
        # Combinez toutes les transformations pour obtenir la position du landmark dans l'espace NAO.
        robotToLandmark = robotToCamera * cameraToLandmarkRotationTransform * cameraToLandmarkTranslationTransform
pass
```

Conclusion

On a appris :

- le langage (python),
- utilisation de noaqi
- utilisation du chorégraphe

La version 2 de notre logiciel est d'atteindre l'objectif initial de notre projet.

Si on nous demander de refaire notre projet, on aura change beaucoup de chose :

1) La structure du projet
2) Et rendre la procédure de recherche autonome