Per i punti extra

è richiesto implementare una "motion control low" per far muovere tiago all'interno del corridoio (almeno da quanto scritto pare che sia necessario implementarla solo dentro il corridoio e non sempre).

Ciò vuol dire processare i dati ottenuti dal laser, quindi andare ad operare a basso livello, analizzare i punti ottenuti per capire dove sono gli ostacoli (le pareti del corridoio); per poi dare a tiago i comandi di velocità (lineare ed angolare). Onestamente come cosa sembra abbastanza complessa e molto tecnica, andare a calcolare una traiettoria dinamica da imporre a tiago in base ai dati ottenuti dal laser.

Muovere Tiago in generale

Muovere tiago usando il ROS navigation stack è molto più semplice: basta inviare la target position and orientation usando il topic: /move\_base\_simple/goal.

Questo topic usa il messaggio (già definito): geometry\_msgs/PoseStamped.

Questo messaggio ha in header il nome del reference frame e un timestamp, per esempio:

* goal = PoseStamped()
* goal.header.frame\_id = "map"
* goal.header.stamp = rospy.Time.now()

dopodichè basta usare I campi del messaggio per specificare le coordinate e orientazione:

* goal.pose.position.x = 2.0
* goal.pose.position.y = 1.0
* goal.pose.orientation.w = 1.0 # Facing forward

per eseguire il commando basta usare il publisher:

* pub = rospy.Publisher('/move\_base\_simple/goal', PoseStamped, queue\_size=10)

e inviarlo ovviamente facendo:

* pub.publish(goal)

Da chat gpt: “When using the **ROS Navigation Stack**, you **do not need to process laser data manually**. The navigation stack handles sensor data processing, localization, path planning, and obstacle avoidance for you. This allows you to focus on specifying where you want Tiago to go rather than how it gets there.”

Da chat gpt:

 **Goal**: Make Tiago navigate the environment and find the AprilTags corresponding to the 5 random IDs received from Node A.

 **Sensors**:

* **Laser scanner (/scan)**: To detect obstacles and compute safe navigation paths.
* **Camera (/xtion/...)**: To detect and identify AprilTags.

 **Challenges**:

* Navigating systematically through all parts of the environment.
* Avoiding obstacles while covering narrow spaces.
* Using the AprilTag detection node (/tag\_detections) to identify tags in view.

Da quello che ho capito I dati del laser sono utili e vanno usati comunque (in maniera più semplice rispetto alla parte extra) per evitare di mandare tiago a incastrarsi negli angoli per esempio.

Pare che sia già implementata la logica della tag detection basandosi sui dati ottenuti dalla telecamera di tiago, quindi non dobbiamo implementare nessuna logica di tipo computer vision.

Che fare?

Intanto concentriamoci sulla parte obbligatoria del progetto, andremo a vedere la motion control law più avanti. Dobbiamo pensare ad una strategia per definire dinamicamente le coordinate che Tiago deve raggiungere in una maniera ragionevole. Abbiamo a disposizione solo i dati del laser, che in poche parole dicono il raggio di spazio libero rispetto alla posizione di tiago, in ogni direzione.

Il nostro obiettivo è far sì che tiago si muova nella mappa “a caso” in cerca dei tag, ma con un costrutto. Per esempio è inutile farlo muovere in una direzione che è vicino a un muro o un ostacolo.

Esempio di logica abbastanza naive (fatto da me non chatgpt):

* Primo goal per uscire dal corridoio: far andare tiago nella direzione libera più distante da un ostacolo (dati dal laser), in modo da uscire dal corridoio con la telecamera che intanto guarda sempre avanti.
* Poi scegliere random una direzione sufficientemente libera, e settare come goal il punto un metro prima dell’ostacolo. Ripetere finchè non sono stati trovati i 5 apriltag giusti.

In questo modo tiago dovrebbe, prima o poi, esplorare tutta la mappa. Non ho la più pallida idea di quanto ci metta e se in pratica funziona ragionevolmente.