Tracce progetto di gruppo

Fondamenti di Robotica

Docenti: Loredana Zollo
Clemente Lauretti
Tutor: Rita Molle

email: l.zollo@unicampus.it
c.lauretti@unicampus.it
rita.molle@unicampus.it

CAMPUS BIO-MEDICO UNIVERSITY OF ROME
Via Álvaro del Portillo, 21 - 00128 Rome - Italy
www.unicampus.it

CREO LAB

UNIVERSITA' CAMPUS BIO-MEDICO DI ROMA

Overview del progetto di gruppo

Obiettivo: Sviluppare un framework di simulazione che consenta al robot TIAGo di eseguire attività di pick and place, sfruttando algoritmi avanzati basati sull'intelligenza artificiale per la modellazione, la pianificazione del movimento, il controllo e la percezione del robot

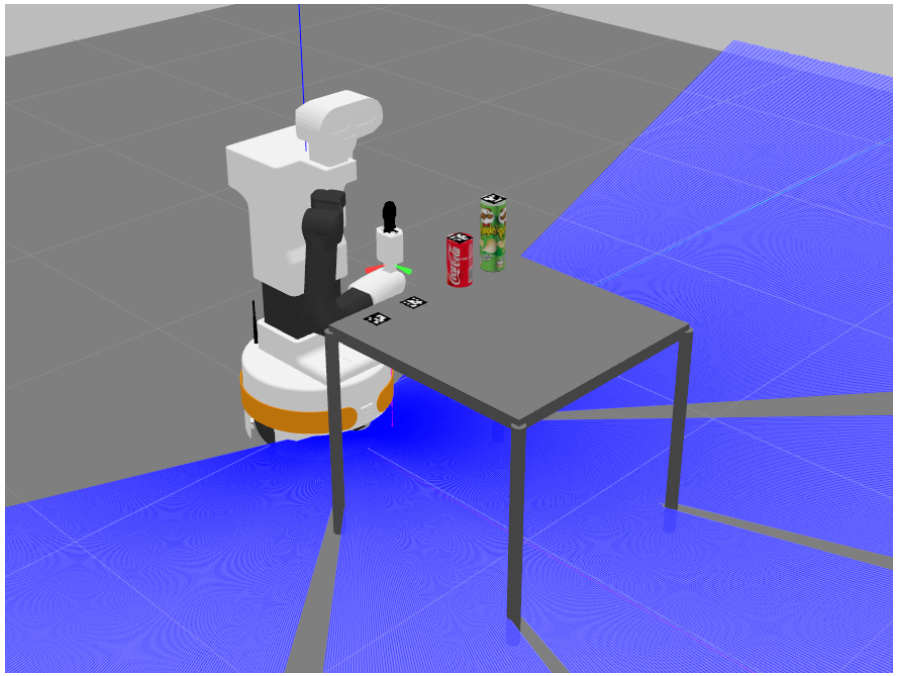
Tools & conoscenze richieste: ROS 2, simulatori Rviz e Gazebo, basi di robotica, intelligenza artificiale e ArUco marker

Sfide principali:

Task 1: Modellazione di robot in Gazebo
Localizzazione di oggetti
Trasformazioni tra frame

Task 2: Pianificazione del movimento
Inversione cinematica
Controllo del movimento

Task 3: Uso di AI in uno dei blocchi funzionali sviluppati nel task 2



Modalità di verifica

- Consegna dei pkg creati per i vari moduli entro le scadenze riportate
- Presentazione di gruppo a fine corso e domande individuali

Consegna Task 1	Consegna Task 2	Consegna Task 3
13/05/25	22/05/25	29/05/25

Gruppi

Gruppo 1
1. Vitale Davide
2. Faraone Emanuele Antonio
3. Imbalzano Stefano
4. Minorenti Valerio
5. Minotta Matteo

Gruppo 2
1. Barnaba Gianluca
2. Bini Federico
3. Bussone Beatrice
4. Principi Federico
5. Scorza Luca

Gruppo 3
1. D'Ercole Alessia
2. Di Castro Rebecca
3. Fattore Emanuele
4. Lo Pinto Giorgia

Gruppo 4
1. Cordova Claudia Lara
2. Soricone Lorenzo
3. Naddeo Annamaria
4. Di Filippo Alessandro
5. Pattumelli Dario

Gruppo 5
1. Salazar Rocco
2. Grande Matteo
3. Petruzziello Luigi
4. Ferranti Alessandro
5. Limongi Daniele

Gruppo 6
1. Santarelli Simone
2. Landini Luca
3. Dragotta Claudio
4. Rosati Andrea

Gruppo 7
1. Dascola Alessandro
2. Bonuccelli Federico
3. Ciccolella Gabriele
4. Casaula Riccardo
5. Grussu Lorenzo

Gruppo 8
1. Benedetti Martina
2. Lioi Giuseppe Pio
3. Sola Siria
4. Viglialoro Sebastiano
5. Caputo Damiano

Task 1 – consegna 13 maggio 2025

Obiettivi specifici:

Programmare il robot TIAGo per i) **eseguire una scansione dell'ambiente** muovendo gradualmente la testa del robot, ii) **rilevare i marker ArUco** tramite la camera RGB-D e iii) **trasformare le coordinate dei marker** dal sistema di riferimento della camera al sistema di riferimento della base del robot.

Step da seguire:

- Avviare la simulazione del robot in Gazebo
- Movimentare la testa del TIAGo al fine di scansionare l'ambiente
- Rilevare la presenza di ArUco marker nella scena
- Stimare la posa dei marker rispetto alla terna camera
- Stimare la posa della camera rispetto alla terna base del robot
- Effettuare una trasformazione di coordinate per esprimere la posa dei marker in terna base
- Stampare la posa dei marker rispetto alla base del robot
- Visualizzare su Rviz e su gazebo la terna dei marker

Strumenti e librerie da utilizzare:

- Nodi, Topics, Services, Actions, Launch files, TF2ROS OpenCV

Suggerimenti utili per Task 1

1. **Avviare la simulazione del robot in ambiente Gazebo**
   - Seguire le indicazioni fornite alla fine della presentazione per l'installazione dei pkg del Tiago e la configurazione dell'ambiente
   - Eseguire il commando: ros2 launch tiago_gazebo tiago_gazebo.launch.py is_public_sim:=True ()
2. **Movimentare la testa del TIAGo al fine di scansionare l'ambiente**
   - Topic su cui pubblicare: /head_controller/joint_trajectory
   - Tipo messaggio: JointTrajectory
   - 2 giunti: head_1_joint (destra/sinistra → da 0.217 a -0.217) e head_2_joint (su/giù → fisso -0.57)

   NB: implementare un Action server e Action client per gestire i movimenti della testa
3. **Stimare la posa degli ArUco presenti nella scena**
   - Leggere dal topic /head_front_camera/rgb/image_raw (tipo messaggio Image) le immagini visualizzate dal robot
   - Sottoscrivere ai parametri della camera /head_front_camera/rgb/camera_info (tipo messaggio CameraInfo)
   - Pubblicare la posa e l'ID dell'ArUco marker (cv2.aruco.DICT_6x6_250, marker_size = 0.06)
4. **Effettuare una trasformazione di coordinate per esprimere la posa dei marker in terna base**
   - Trasformare la posa dal frame 'head_front_camera_rgb_optical_frame' a 'base_footprint' utilizzando tf
   - Pubblicare su uno o più topic la posa degli ArUco rilevati

Comandi utili per familiarizzare con il robot (Task 1)

Visualizzare controlli attivi

ros2 control list_controllers

Visualizzare stato dei giunti

ros2 topic echo /joint_states

Visualizzare stringa dei giunti in arm_controller

ros2 param get /head_controller joints

Aprire la GUI per Rviz

ros2 run joint_state_publisher_gui joint_state_publisher_gui

Aprire la telecamera del TIAGo

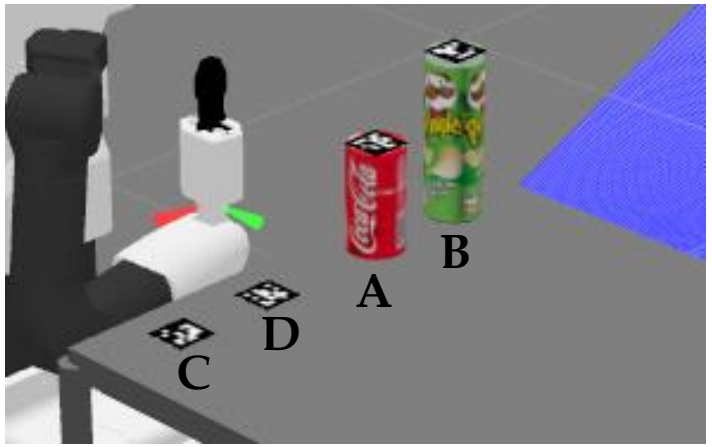
ros2 run rqt_image_view rqt_image_view → settare il topic corretto dalla GUI della camera (head_front_camera/rgb/image_raw)

Spostare la testa del TIAGo da terminale

ros2 topic pub /head_controller/joint_trajectory trajectory_msgs/msg/JointTrajectory "{ joint_names: [‘head_1_joint’, ‘head_2_joint'], points: [{positions: [-2.17e-05, -0.57], time_from_start: {sec: 2}}]} "

Task 2 – consegna 30 maggio 2025

Obiettivi specifici:
Programmare il robot TIAGo per i) raggiungere con l'organo terminale due oggetti nello spazio di lavoro (posizione A e B in figura), ii) afferrarli con la pinza, e iii) spostarli rispettivamente in posizione C e D. Le posizioni iniziali e finali degli oggetti sono definite dagli Aruco marker.



Moduli da sviluppare

- Pianificatore del movimento nello spazio operativo
- Inversione cinematica con gestione degli ostacoli
- Modulo per invio comandi al controllo nello spazio dei giunti (tronco + braccio + gripper)

Step da seguire:

- Avviare la simulazione del robot in Gazebo
- Caricare il file urdf del TIAGo (contenente solo alcuni giunti e alcuni link) tramite robotics toolbox
- Portare il robot in una configurazione che consenta di eseguire il task
- Visualizzare la posa degli aruco e dell'end-effector su RViz per pianificare la pose da raggiungere
- Scegliere una sequenza di movimentazione degli oggetti che agevoli l'evitamento degli ostacoli (creare una macchina a stati che gestisce la sequenza delle operazioni)
- Pianificazione e inversione cinematica utilizzando le funzioni del robotics toolbox e invio della configurazione dei giunti desiderata al controller

Strumenti e librerie da utilizzare:

- Nodi, Topics, Services, Actions, Launch files, RoboticsToolbox, Rviz, Gazebo

Suggerimenti utili per Task 2

1. **Caricare il file urdf del TIAGo tramite le funzioni di RoboticsToolbox**
   - Scaricare Il file URDF da e-learning e seguire le istruzioni riportate nelle slide della lezione su Robotics Toolbox
2. **Portare il robot in una configurazione che consenta di eseguire il task**
   - 'torso_lift_joint': 0.35
   - 'arm_1_joint', 'arm_2_joint', …: [0.07, 0.1, -3.1, 1.36, 2.05, 0.01, -0.05]
   - Action su cui pubblicare: /arm_controller/follow_joint_trajectory ; /torso_controller/follow_joint_trajectory
   - Tipo messaggio: JointTrajectory

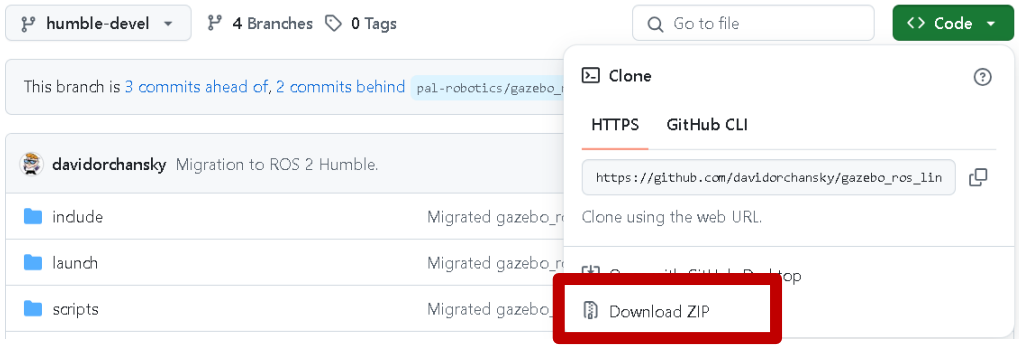
   NB: implentare un Action Client su launch file che invia la configurazione dei giunti iniziale su /arm_controller/follow_joint_trajectory e /torso_controller/follow_joint_trajectory
3. **Visualizzare la posa degli aruco e dell'end-effector su RViz**
   - Le pose da raggiungere non saranno esattamente quelle individuate dagli Aruco, ma avranno degli offset su posizione e orientamento (per la pianificazione degli offset aiutarsi con la visualizzazione delle terne in Rviz)
4. **Scegliere una sequenza di movimentazione degli oggetti che agevoli l'evitamento degli ostacoli**
   - Creare un nodo «Macchina a Stati» per che gestisce la sequenza di attivazione dei vari task (es. raggiungimento oggetto 1, presa, spostamento, ecc.). Un task può essere eseguito solo quando il precedente è completato.
5. **Pianificazione, inversione cinematica e invio comandi al controller**
   - Implementare un nodo che riceve la posa desiderata e il tipo di task da eseguire (dalla macchina a stati), pianifica il movimento nello spazio operativo, inverte la cinematica e invia la traiettoria nello spazio dei giunti al controller tramite action

   Action su cui pubblicare: /arm_controller/follow_joint_trajectory ; /torso_controller/follow_joint_trajectory; /gripper_controller/follow_joint_trajectory

   NB: nel caso in cui il tipo di task è "chiusura/apertura gripper" il movimento verrà pianificato direttamente nello spazio dei giunti

Suggerimenti utili per Task 2 – Spostare oggetto

1. **Scaricare zip gazebo_ros_link_attacher**
   - https://github.com/davidorchansky/gazebo_ros_link_attacher/tree/humble-devel
   - Cliccare su «Code» e scaricare il file zip



2. **Estrarre il file zip; creare un nuovo ws e src nel ws**
3. **Entrare nella cartella src del ws e copiare la cartella appena estratta**
   - cd gazebo_link_attacher_ws
4. **Fare il build e source**
   - colcon build
   - source install/setup.bash
5. **Aprire il bashrc e copiare l'export alla fine del file CAMBIANDO il path e mettendo il proprio relativo alla cartella /install/gazebo:ros_link_attacher/lib**
   - gedit ~/.bashrc
   - export GAZEBO_PLUGIN_PATH=$GAZEBO_PLUGIN_PATH:/home/rita/gazebo_ros_link_attacher_ws/src/gazebo_link_attacher/install/gazebo_ros_link_attacher/lib
6. **Scaricare da elearning il file «World con plugin gazebo» e sostituirlo in ~/tiago_public_ws/src/pal_gazebo_worlds/world**
7. **Fare il colcon build del tiago_public_ws**

Suggerimenti utili per Task 2 – Spostare oggetto

1. **Attaccare oggetto (da terminale)**
   - ros2 service call /attach gazebo_ros_link_attacher/srv/Attach "{model_name_1: 'tiago', link_name_1: 'gripper_left_finger_link', model_name_2: 'cocacola', link_name_2: 'link'}"

2. **Staccare oggetto (da terminale)**
   - ros2 service call /detach gazebo_ros_link_attacher/srv/Attach "{model_name_1: 'tiago', link_name_1: 'gripper_left_finger_link', model_name_2: 'cocacola', link_name_2: 'link'}"

Step

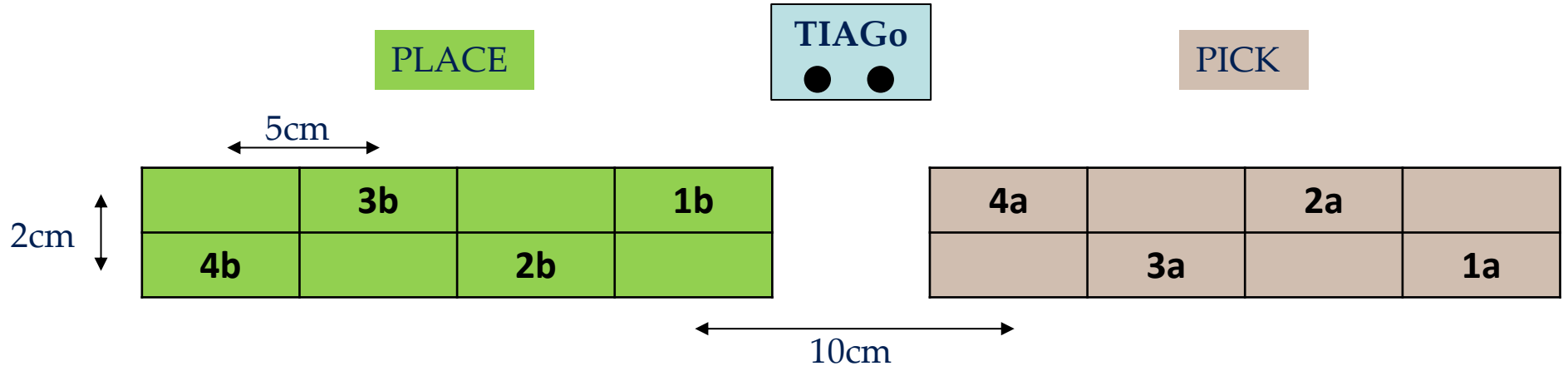
- Raggiungere l'oggetto con il gripper
- Chiudere il gripper mandando il comando [0.041, 0.041] a gripper_controller/follow_joint_trajectory
- Attaccare l'oggetto con service del punto 1
- Spostare l'oggetto nella posizione di place
- Aprire il gripper mandando il comando [0.044, 0.044] a gripper_controller/follow_joint_trajectory
- Staccare l'oggetto con service del punto 2
- Tornare nella posizione iniziale

Inserire il comando attacca/stacca in un launch.py oppure eseguire i procedimenti per creare una "custom interface" e mandare I comandi da codice python

Task 3 – facoltativo – consegna il 4 giugno 2025

Obiettivi specifici:
Sviluppo di un modulo LbD in sostituzione del blocco funzionale di pianificazione sviluppato nel Task 2.

Dataset:
- Il dataset è stato acquisito in uno scenario reale presentato in figura, diviso in area di **pick** e area di **place**
- Sono state acquisite le configurazioni degli angoli di giunto del braccio del TIAGo. Ogni traiettoria nello spazio dei giunti è stata ripetuta 2 volte
- La q del torso è costante a 0.35; arm_6_joint e arm_7_joint sono fissi (poiché tali giunti non son back-drivable)



Spiegazione dataset:
- Il nome di ogni cartella fa riferimento a: ***posizione di pick – posizione di place*** → se l'oggetto è nella posizione 1a lato pick e deve essere spostato in 3b lato place, la cartella corrispondente è 1a-3b
- Il dataset contiene tutte le possibili combinazioni di posizione di pick con posizione di place: ogni traiettoria è stata registrata 2 volte → il numero di ripetizione è alla fine del nome del file txt
- Per ogni cartella sono presenti 4 file .txt: ***task_posizione***_rep***NumeroRipetizione***, con task: {pick, place}, posizione: {1-4} a oppure b indica la posizione di pick o place (in base al task), NumeroRipetizione: {1,2}

Suggerimenti utili per Task 3

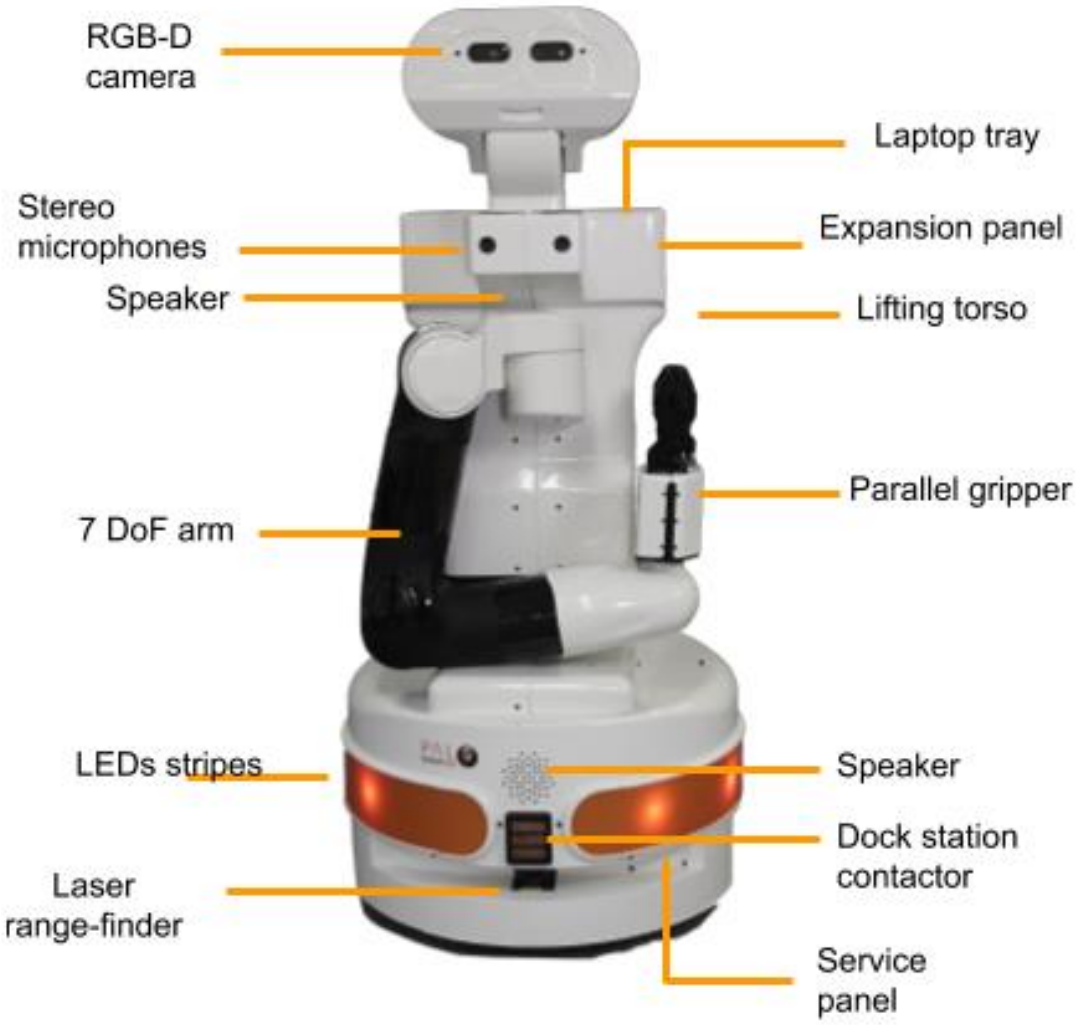
1. **Scaricare il dataset da elearning: "Traiettorie TIAGo – Spazio dei Giunti"**
2. **Caricare il file urdf del TIAGo tramite le funzioni di RoboticsToolbox**
3. **Portare il robot in una configurazione che consenta di eseguire il task** (ultimi 2 di arm cambiati rispetto Task 2)
   - 'torso_lift_joint': 0.35
   - 'arm_1_joint', 'arm_2_joint', …: [0.07, 0.1, -3.1, 1.36, 2.05, -0.52, 1.14]
4. **Considerare una delle 2 alternative:**
   - **delle configurazioni medie per una maggiore generalizzazione**
   - **selezionare la/le traiettoria/e acquisite con una target position vicina a quella desiderata**

   Se più traiettorie →Allineare il dataset e campionarlo in modo da avere lo stesso numero di campioni nei vari dataset

   Se una sola traiettoria → Campionare il dataset per evitare che il movimento sia molto lento
5. **Utilizzare la libreria movement_primitives**
   - A partire dalla configurazione degli angoli di giunto, applicare la cinematica diretta per ricostruire la traiettoria in spazio SE3 e in se3
   - Seguire l'esercitazione LbD

Overview del robot TIAGo

Componenti principali

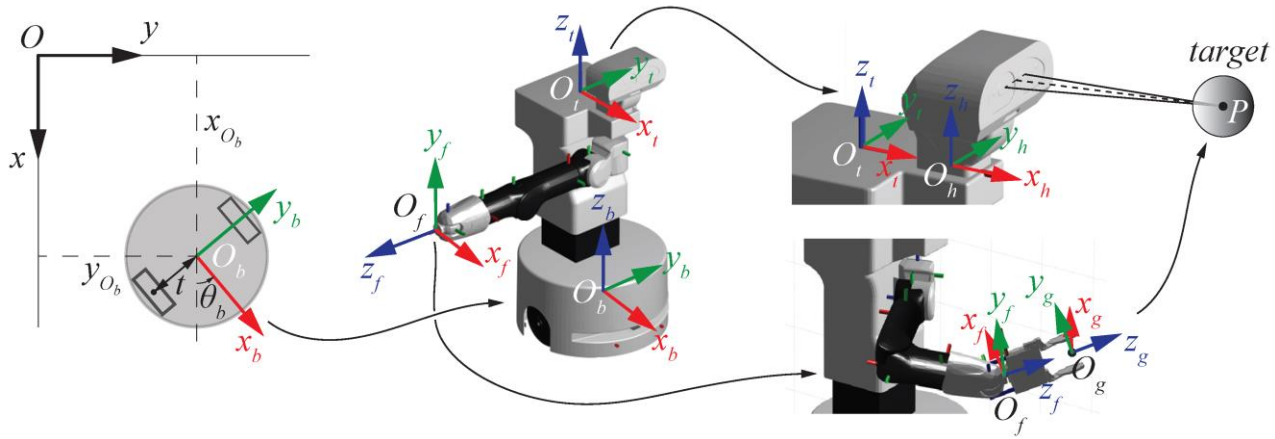


Specifiche

Dimensions	Height	110 – 145 cm
	Weight	72 Kg
	Base footprint	Ø 54 cm
Degrees of freedom	Mobile base	2
	Torso lift	1
	Arm	4
	Wrist	3
	Head	2
	Hey5 hand	19 (3 actuated)
	PAL gripper	2
Mobile base	Drive system	Differential
	Max speed	1 m/s
Torso	Lift stroke	35 cm
Arm	Payload	2 Kg
	Reach	87 cm
Electrical features	Battery	36 V, 20 Ah
Sensors	Base	Laser range-finder
		Sonars
		IMU
	Torso	Stereo microphones
	Arm	Motors current feedback
	Wrist	Force/Torque
	Head	RGB-D camera

Sistemi di riferimento e cinematica del TIAGo

Sistemi di riferimento



$O_b - x_b y_b z_b \rightarrow$ riferimento della piattaforma mobile

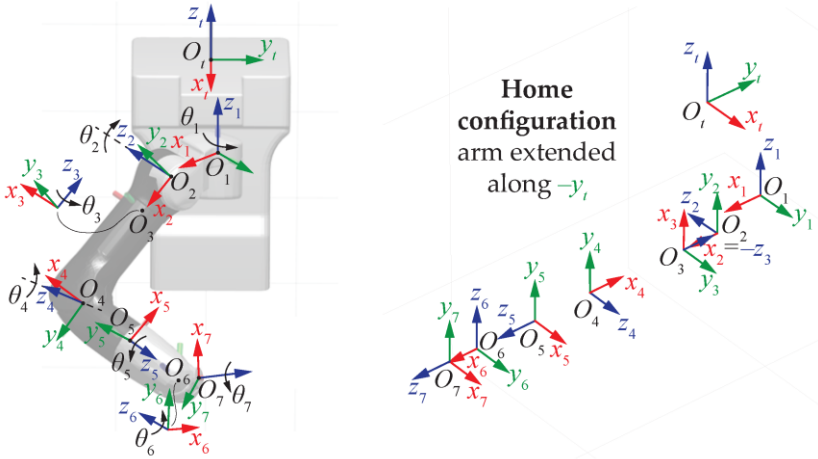
$O_t - x_t y_t z_t \rightarrow$ riferimento del tronco

$O_h - x_h y_h z_h \rightarrow$ riferimento della base della testa

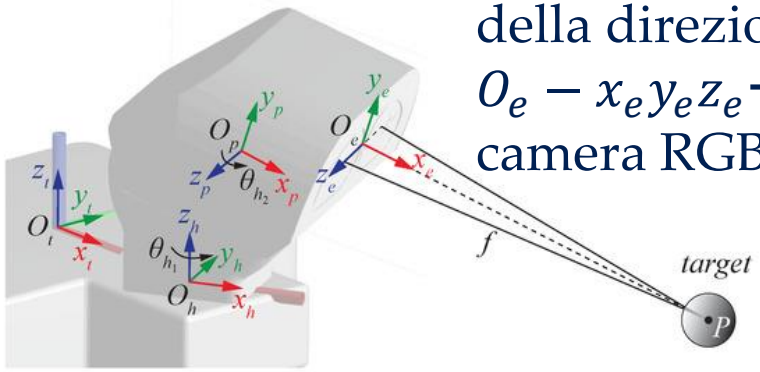
$O_f - x_f y_f z_f \rightarrow$ riferimento dell'ee del braccio robotico

$O_g - x_g y_g z_g \rightarrow$ riferimento del gripper

Cinematica braccio



Cinematica testa



$O_p - x_p y_p z_p \rightarrow$ riferimento responsabile della direzione di puntamento della testa

$O_e - x_e y_e z_e \rightarrow$ riferimento dei sensori della camera RGB

NB: i movimenti avvengono attorno all'asse z dei sistemi di riferimento considerati

Bajrami, A.; Palpacelli, M.-C.; Carbonari, L.; Costa, D. Posture Optimization of the TIAGo Highly-Redundant Robot for Grasping Operation. Robotics 2024, 13, 56. https://doi.org/10.3390/robotics13040056

Guida all'installazione del TIAGo

Link: https://github.com/pal-robotics/tiago_simulation

1. Selezionare **humble-devel**
2. Seguire le istruzioni (tutte a cascata in un unico terminale)

Prerequisiti:

```
sudo apt-get update
sudo apt-get install git python3-vcstool python3-rosdep python3-colcon-common-extensions
```

Settare il workspace:

1. Creare il workspace e clonare le repositories

```
mkdir -p ~/tiago_public_ws/src
cd ~/tiago_public_ws
vcs import --input https://raw.githubusercontent.com/pal-robotics/tiago_tutorials/humble-devel/tiago_public.repos src
```

Settare il workspace:

2. Installare le dipendenze con rosdep

```
sudo rosdep init
rosdep update
rosdep install --from-paths src -y --ignore-src
```

3. Source e build dell'ambiente

```
source /opt/ros/humble/setup.bash
colcon build --symlink-install
```

4. Source del workspace (riga da aggiungere anche nel file ~/.bashrc → gedit ~/.bashrc)

```
source ~/tiago_public_ws/install/setup.bash
```

Simulazione standard → Tiago in un ambiente:

```
ros2 launch tiago_gazebo tiago_gazebo.launch.py is_public_sim:=True
```

Sostituire il file launch

Aprire la cartella: ~/tiago_public_ws/src/tiago_simulation/tiago_gazebo/launch

Sostituire il file **tiago_gazebo.launch.py** con il file launch (stesso nome) dell'elearning

Modifiche → sostituzione del world per inserire "pick_place_close_demo"
→ aggiunta dei comandi per aprire rviz

Scaricare i modelli

Aprire la cartella: ~/tiago_public_ws/src/pal_gazebo_worlds/models

Scaricare i modelli della cartella **Model** dell'elearning e sostituirli nella cartella **models** del ws

Modifiche → creazione ArUco Marker sul tavolo
→ creazione oggetti (pringles e Coca Cola) con ArUco Marker attaccati

Scaricare il mondo

Aprire la cartella: ~/tiago_public_ws/src/pal_gazebo_worlds/world

Scaricare il mondo della cartella **world** dell'elearning e sostituirlo nella cartella **world** del ws

Modifiche → creazione ambiente per pick and place → tavolo con sopra ArUco e oggetti con ArUco

Build dell'ambiente

```
colcon build
```

Source del workspace

```
source ~/tiago_public_ws/install/setup.bash
```

Simulazione → TIAGo nell'ambiente per pick and place:

```
ros2 launch tiago_gazebo tiago_gazebo.launch.py is_public_sim:=True
```