Simulazione di un Robot Autonomo in Gazebo

Progetto di Meccanica dei Robot

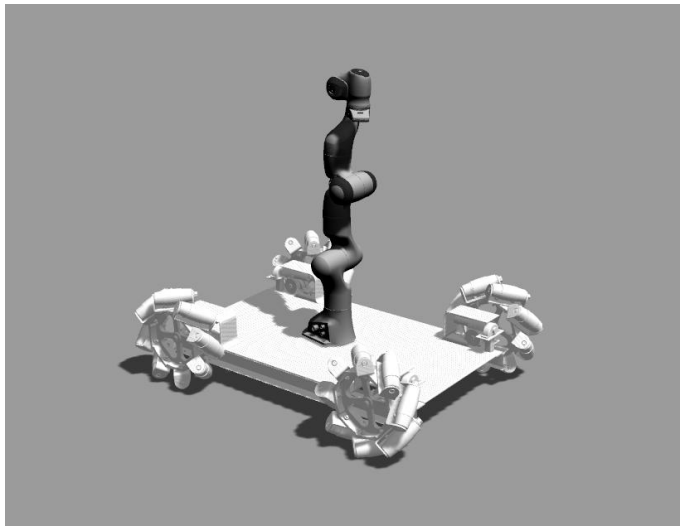
Andrea Boscolo Camiletto

Università di Pisa

Marzo 2020

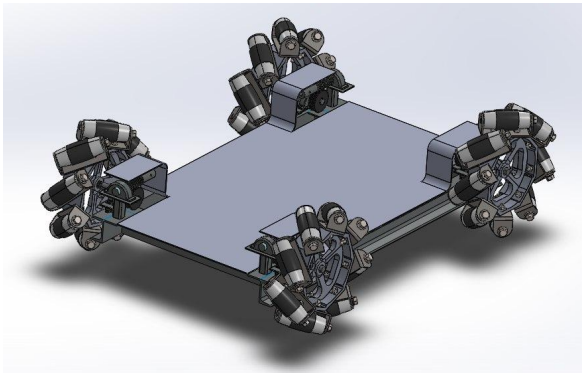
Descrizione del Robot

Il robot è composto da una piattaforma omnidirezionale su cui è montato un braccio robotico commerciale Panda Franka Emika. Sono presenti dunque 3 DOF per la piattaforma e 7 DOF per il braccio robotico. Il robot è rappresentato in figura:



Descrizione del Robot : Piattaforma

La piattaforma nasce da un CAD scaricato da GrabCAD che è possibile avere in figura. Da questo è stato necessario convertire il .step in .stl abbassandone la qualità e spostare il sistema di riferimento in uno principale d'inerzia: a questo punto è sicuramente d'aiuto il plugin "sw_urdf_exporter" che permette di avere un URDF già con masse, inerzie e origini integrate.

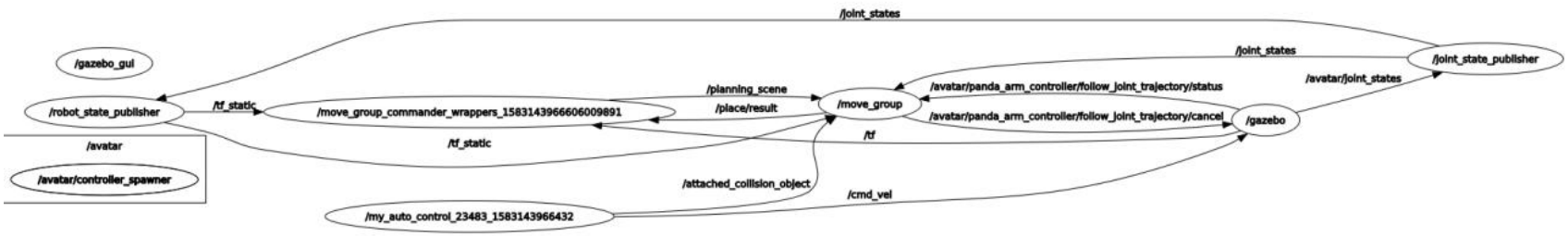


Descrizione del Robot : Manipolatore

La descrizione URDF del robot è fornita dal produttore ma non c'è una compatibilità con Gazebo nè una descrizione dinamica, sono presenti infatti delle masse e delle inerzie fittizie. La scelta del manipolatore Panda era inizialmente dovuta alla presenza di un Gripper con 1 DOF però per qualche motivo Gazebo aveva problemi a visualizzare il file .dae che ne rappresenta la mash e lo ho dovuto togliere.
E' possibile trovare il tutto nel pacchetto `franka_ros` disponibile su github.
Chiaramente, è presente un giunto di tipo "fixed" tra piattaforma e braccio robotico

Controllo del robot

Il controllo del Robot è relativamente semplice. La piattaforma viene controllata in velocità mentre il manipolatore cinematicamente in posizione. L'implementazione del Panda in Gazebo con un controllo dinamico richiede una stima delle masse e delle inerzie dei vari link che è possibile trovare online insieme a un set di PID tarato su quei valori.
Purtroppo però le forze inerziali dovute alla piattaforma hanno fatto sì che quei valori non fossero validi e non sono riuscito a tararli in maniera migliore. Per questo motivo sono dovuto passare a un controllo puramente cinematico dei vari giunti.



Controllo del robot : Piattaforma

La piattaforma viene controllata dal plugin di Gazebo planar_move che permette di pubblicare sul topic `cmd_vel` un messaggio di tipo Twist che esprime le velocità da imporre alla piattaforma nel sistema di riferimento solidale. Il plugin inoltre mi pubblica nel topic `odom` la trasformazione tra il frame fisso e quello mobile (e la velocità relativa) con un messaggio di tipo Odometry, che comprende Pose e Twist.

```
<plugin name="object_controller" filename="libgazebo_ros_planar_move.so">
  <commandTopic>cmd_vel</commandTopic>
  <odometryTopic>odom</odometryTopic>
  <odometryFrame>odom</odometryFrame>
  <odometryRate>100.0</odometryRate>
  <robotBaseFrame>platform</robotBaseFrame>
</plugin>
```

Controllo del robot : Manipolatore - basso livello

Il manipolatore richiede che sia impostato:

- Un controllo di basso livello in Gazebo
- Un controllo di alto livello in MoveIt!

Per quanto riguarda il basso livello, come anticipato, va introdotto nell'urdf una hardware interface che permetta a Gazebo di controllare direttamente la posizione (da qui il PositionJointInterface). E' inoltre necessario un file `ros_controller.yaml` da cui Gazebo possa riconoscere e caricare i controllori, e inoltre che permetta poi a MoveIt di riconoscere i 7 giunti come appartenenti a un singolo "gruppo" del tipo JointTrajectoryController

Controllo del robot : Manipolatore - basso livello

```
<transmission name="trans_panda_joint6">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="panda_joint6">
        <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
    </joint>
    <actuator name="panda_joint6_motor">
        <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
    </actuator>
</transmission>
```

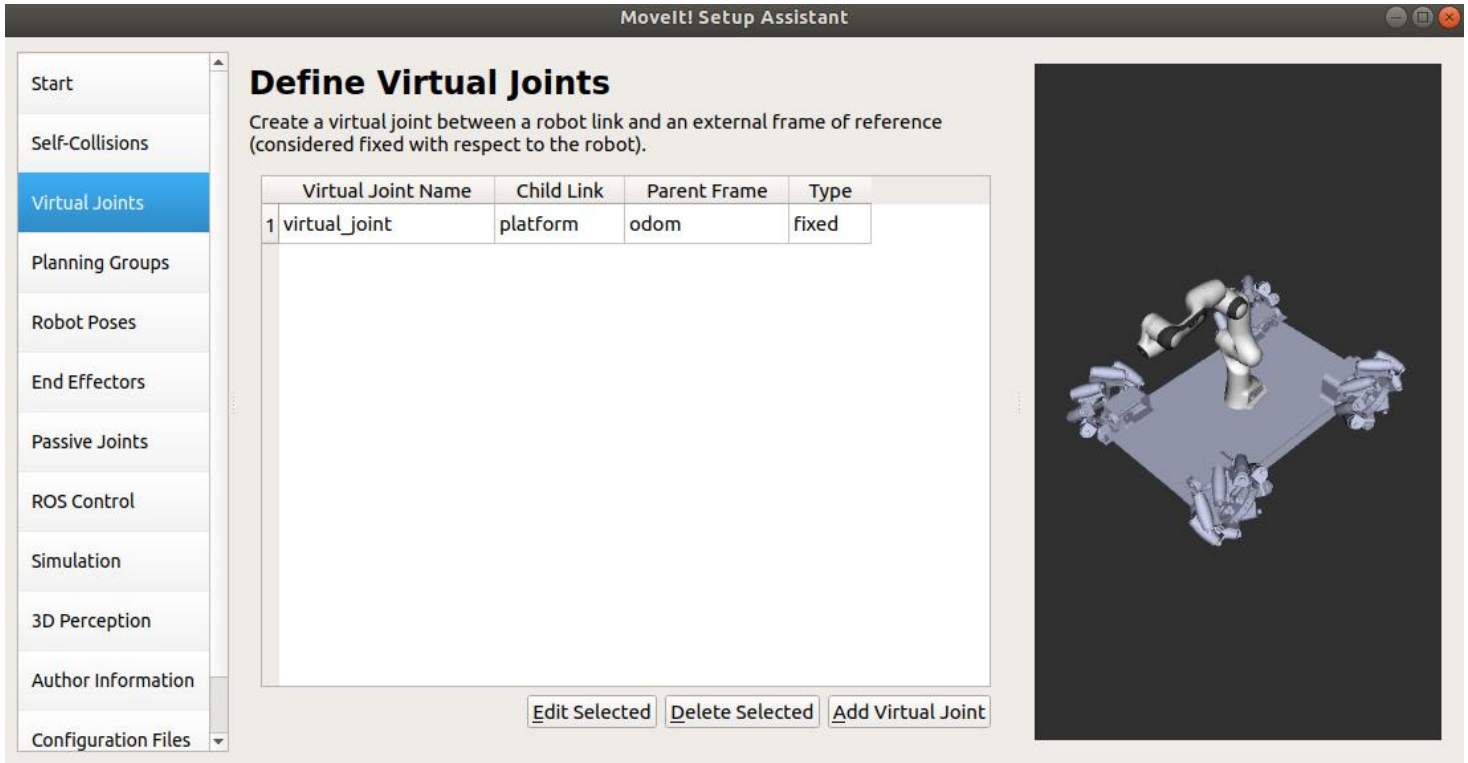
Figure: Hardware interface per Gazebo

```
avatar:
    joint_state_controller:
      type: joint_state_controller/JointStateController
      publish_rate: 50
    panda_arm_controller:
      type: position_controllers/JointTrajectoryController
      joints:
        - panda_joint1
        - panda_joint2
        - panda_joint3
        - panda_joint4
        - panda_joint5
        - panda_joint6
        - panda_joint7
```

Figure: Parte di `ros_controller.yaml`

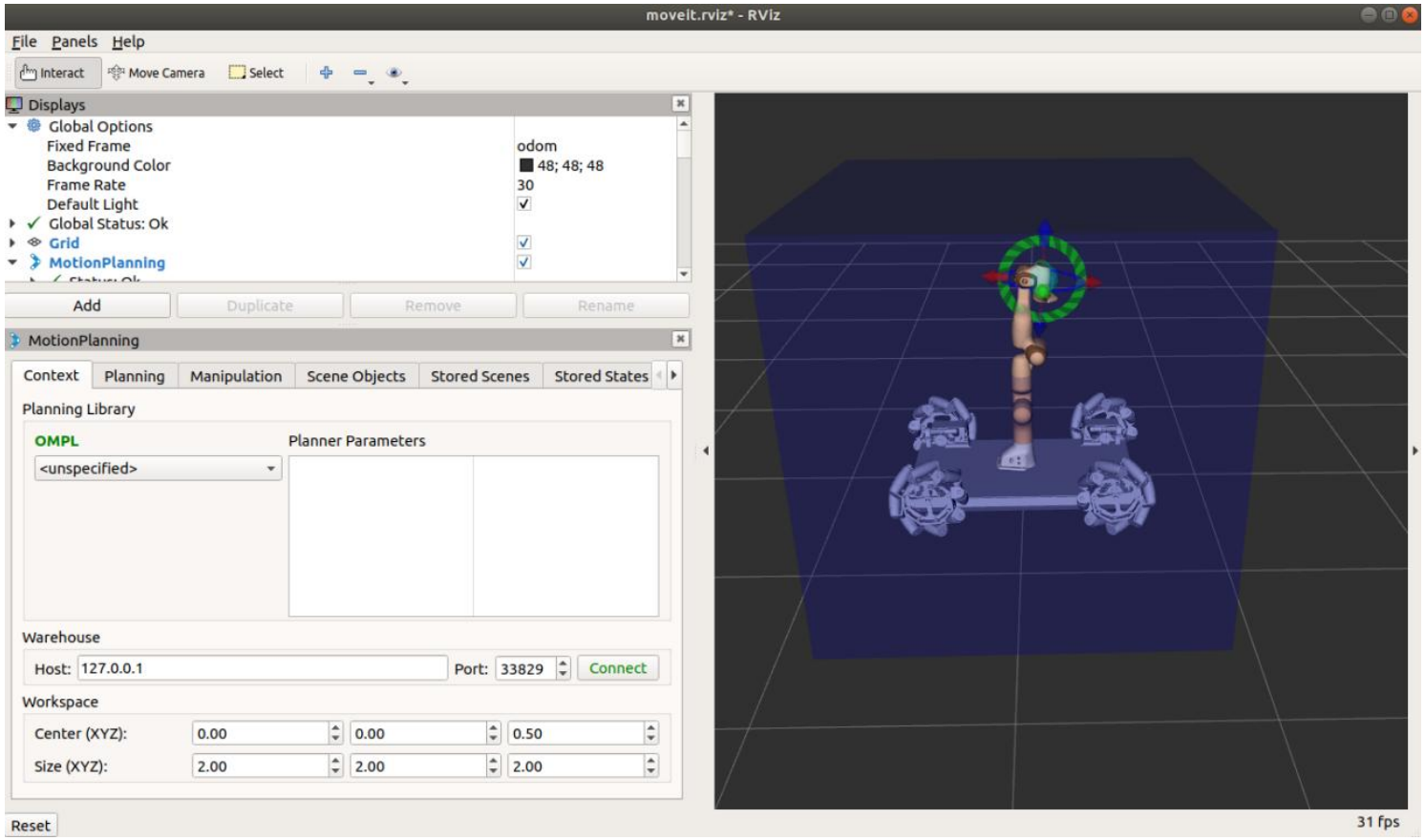
Controllo del robot : Manipolatore - alto livello

Per il planning di alto livello è necessario setuppare il nostro Robot con il MoveIt! Setup Assistant. Dalla figura è possibile vedere cosa permette di fare.



Controllo del robot : Manipolatore - alto livello

Una volta setuppato è possibile lanciare Gazebo *in pausa*, avviarlo e poi far partire MoveIt, la cui interfaccia si basa su RViz. Da RViz è possibile provare tramite l'interfaccia grafica i primi planning che verranno ripetuti in Gazebo.



Controllo del robot : Manipolatore - alto livello

E' possibile interfacciarsi con MoveIt! anche via codice, sia in C++ che in Python, usato in questo caso. MoveIt! fornisce una classe che comprende al suo interno diverse funzioni, tra le quali:

- `go_to_joint_state`
- `go_to_pose_goal`
- `plan_cartesian_path`

Per quanto riguarda la piattaforma invece è sufficiente inizializzare un publisher che pubblichi in cmd_vel

Task 1 : Pick 'n place

E' possibile trovare un video esemplificativo al seguente **link**.
Oltre agli strumenti già presentati viene qui utilizzato il servizio di Gazebo "get_link_state" che mi fornisce la posizione e l'orientazione della piattaforma rispetto al sistema di riferimento fisso.
Da linea di codice è necessario inserire delle coordinate x, y, z e il robot arriverà a posizionare l'end effector in quel punto minimizzando lo spostamento. E' possibile definire, anche se non da linea di comando, l'orientazione finale dell'EE.

Task 2 : Dipingere un cilindro

E' possibile trovare un video esemplificativo al seguente **link**.
Qui si sfrutta la funzione `plan_cartesian_path` che mi permette di aggiungere dei waypoint e interpolare il percorso che li congiunge. Così facendo ottengo abbastanza facilmente una traiettoria oscillatoria verticale che combinata con un'opportuna velocità della piattaforma omnidirezionale permette di ottenere il risultato voluto.