Università Politecnica delle Marche

Ingegneria Informatica e dell'Automazione

RoboSimCloth

Corso di

Laboratorio di Automazione

Anno accademico 2024-2025

Studenti:
Pizzuto Andrea
Meloccaro Lorenzo
Percipalle Noemi

Professore:
Andrea Bonci

Dottorandi:
Serafini Andrea
Pellicani Ilaria

Dipartimento di Ingegneria dell'Informazione

Indice

1 Introduzione

Il capitolo fornisce una panoramica del progetto, descrivendo l'obiettivo principale di sviluppare e simulare la movimentazione di un robot TM-900 con un tessuto, attraverso l'integrazione di ROS2 e Unity. Viene fornita una panoramica delle motivazioni per l'uso di questi software e delle principali fasi del progetto.

Il presente progetto ha come obiettivo lo sviluppo e la simulazione della movimentazione di un robot TM-900 con un tessuto (cloth), utilizzando l'integrazione di software avanzati come ROS2 e Unity. Il contesto applicativo del progetto si colloca nell'ambito industriale e robotico, ponendo particolare attenzione sull'ottimizzazione dei processi di smontaggio di oggetti, come abiti e tessuti, mediante l'interazione con il robot. L'azienda coinvolta utilizza il software Cloth 3D per la modellazione degli abiti, mentre Unity e Gazebo sono impiegati per simulare la movimentazione del robot e l'interazione con gli oggetti. Uno degli obiettivi di questo progetto è quello di analizzare in dettaglio l'integrazione tra ROS2 e Unity, scelta motivata dalle potenzialità di questi due strumenti. ROS2 offre una gestione efficiente della comunicazione tra il robot e il sistema di simulazione, mentre Unity fornisce una grafica avanzata che consente di visualizzare e simulare i movimenti del robot in tempo reale, sfruttando ROS2. L'integrazione di questi due software è stata scelta per soddisfare la necessità di creare un ambiente simulativo realistico, utile non solo per eseguire test simulativi, ma anche per implementare successivamente il sistema su un robot reale. In questo modo, l'integrazione tra ROS2 e Unity consente di gestire il robot e simularlo in ambiente virtuale, migliorando l'efficienza e la precisione nelle operazioni robotiche. Il progetto affronta diversi task, tra cui la valutazione e la compatibilità dei tessuti con i vari software, la simulazione della movimentazione del robot e dell'oggetto da smontare in un ambiente condiviso, l'acquisizione e la definizione di una sequenza di fasi di smontaggio, l'esecuzione delle fasi in simulazione con il robot e, infine, l'esecuzione delle stesse fasi su un robot reale.

2 Materiali e Metodi

In questo capitolo saranno presentati i materiali(principalmente software) e i metodi, quindi le tecniche utilizzate, inoltre vengono dettagliate le tecnologie impiegate e le procedure seguite, in modo da permettere la riproducibilità del progetto. In particolare la prima sezione ***Integrazione ROS-Unity*** *approfondisce l'integrazione tra ROS 2 e Unity, analizzando strumenti chiave come ROS-TCP Connector, ROS-TCP Endpoint e URDF Importer (3.1.3). La sezione successiva tratta l'uso di* ***MoveIt*** *per la pianificazione del movimento, mentre la 3.3 descrive l'impiego di* ***Gazebo*** *per la simulazione robotica. Successivamente, la sezione 3.4 introduce* ***Cloth di Unity****, utilizzato per la simulazione di tessuti, e la 3.5 approfondisce la gestione della* ***Camera*** *all'interno dell'ambiente simulato.*

2.1 Hardware

2.1.1 Omron TM5-900

2.1.2 Stereo-Camera RealSense D435i

2.2 Software

In questo capitolo verranno analizzati i software principali utilizzati per il progetto: ROS 2, Unity e Gazebo. Verranno fornite una panoramica generale, informazioni sul loro utilizzo e sulle versioni adottate, evidenziando eventuali problematiche riscontrate e le soluzioni adottate.

2.2.1 Ubuntu

2.2.2 Robot Operating System 2

Figure 2.1: *Logo ROS2*

ROS 2 (Robot Operating System 2) [1] è un framework open-source per lo sviluppo di applicazioni robotiche. Fornisce una serie di strumenti, librerie e convenzioni per facilitare

la comunicazione tra i componenti software di un sistema robotico. ROS 2 è progettato per migliorare la scalabilità, la sicurezza e la compatibilità con i sistemi distribuiti rispetto al suo predecessore, ROS 1. Il suo utilizzo è diffuso in ricerca e industria per il controllo di robot mobili, bracci robotici e veicoli autonomi. In questo progetto, ROS 2 è stato utilizzato per gestire la comunicazione tra il robot simulato in Unity e i componenti di controllo. ROS 2 permette lo scambio di messaggi tra i nodi, abilitando il controllo del robot da Unity e viceversa. In particolare, il framework ha permesso l'integrazione con il sistema di simulazione, la gestione delle traiettorie e il controllo dei giunti del manipolatore. Inizialmente, il progetto è stato avviato con ROS 2 Jazzy, ma si sono riscontrati diversi problemi di compatibilità e instabilità, in particolare con l'integrazione del ROS-TCP Connector per Unity. Per questo motivo, è stato deciso di passare a ROS 2 Humble, una versione più stabile e ampiamente supportata, che ha garantito una migliore compatibilità con gli strumenti di sviluppo utilizzati.

2.2.3 Moveit

2.2.4 Gazebo

2.2.5 Unity e UnityHUB

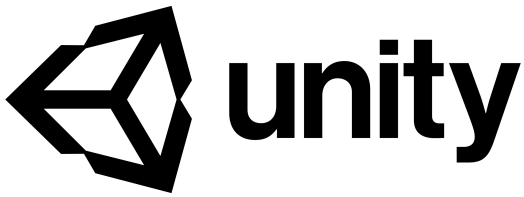

Figure 2.2: *Logo Unity*

Unity [2] è un motore di gioco e simulazione ampiamente utilizzato per la creazione di ambienti interattivi in tempo reale. Sebbene sia nato per lo sviluppo di videogiochi, il suo utilizzo si è esteso ad applicazioni di simulazione, robotica, realtà virtuale e aumentata. Unity permette di creare ambienti 3D realistici e interattivi grazie al suo motore grafico avanzato e alle sue funzionalità di scripting basate su C#. In questo progetto, Unity è stato utilizzato come ambiente di simulazione per il robot. Grazie alla sua compatibilità con ROS2 tramite il ROS-TCP Connector, Unity ha permesso di visualizzare il robot, simulare i suoi movimenti e interagire con l'ambiente circostante. Il motore fisico di Unity è stato sfruttato soprattutto per replicare le dinamiche fisiche della maglietta, garantendo una simulazione più realistica delle interazioni tra un indumento e gli oggetti della scena. Per lo sviluppo del progetto, abbiamo utilizzato Unity 6, la versione più recente al momento di sviluppo del progetto.

2.2.6 Strumento Cloth di Unity

Il sistema **Cloth di Unity** offre una soluzione basata sulla fisica per simulare tessuti e materiali flessibili all'interno di ambienti 3D. Sebbene sia stato progettato principalmente per rappresentare abbigliamento su personaggi, può essere utilizzato anche per altri scopi, come bandiere, tende o qualsiasi altro oggetto che richieda una simulazione realistica del comportamento dei tessuti[3]. In questo progetto, il componente Cloth riveste un ruolo central,poiché il compito finale prevede che l'indumento manipolato dal robot presenti una fisica il più possibile realistica, simulando accuratamente il comportamento del tessuto. A tal fine, sono stati forniti tre file principali relativi a una maglietta: un file **OBJ**, un file **FBX** e un fil **Collada**. L'analisi di questi file ha permesso di ottenere informazioni utili riguardo al formato da utilizzare e alle caratteristiche della mesh per applicare efficacemente il componente Cloth.

2.2.7 Compatibilità dei file con Gazebo e Unity

Il formato **.obj** è stato testato con successo sia in Unity che in Gazebo, consentendo l'importazione corretta del modello della t-shirt in entrambi i software. Tuttavia, durante le simulazioni in Unity, l'applicazione del componente Cloth alla t-shirt ha evidenziato alcune difficoltà: il computer ha riscontrato problemi nel gestire correttamente i comandi impostati e, una volta avviata la simulazione, la t-shirt appariva deformata e non realistica, come è possibile vedere nella figura 2.3.

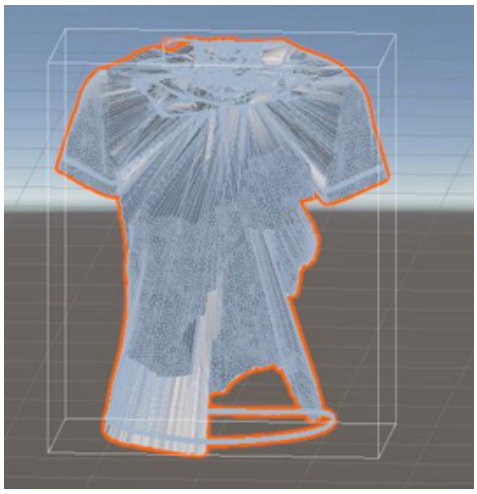

Figure 2.3: *Errore nell'applicazione del Cloth di Unity*

Per risolvere questo problema, l'oggetto è stato importato nel software Blender, dove è stato utilizzato il modificatore Decimate per ridurre il numero di poligoni della mesh. Diminuendo il parametro "Ratio" a un valore che ha portato la mesh a circa 5000 poligoni, è stato possibile reimportare l'oggetto in Unity e applicare correttamente il componente Cloth, ottenendo una simulazione più realistica.

Il formato **.fbx** non è supportato da Gazebo; tuttavia, è compatibile con Unity e può essere convertito in altri formati, come .obj o .dae (Collada), utilizzando Blender. Seguendo lo stesso procedimento adottato per il file .obj, quindi riducendo il numero di poligoni delle

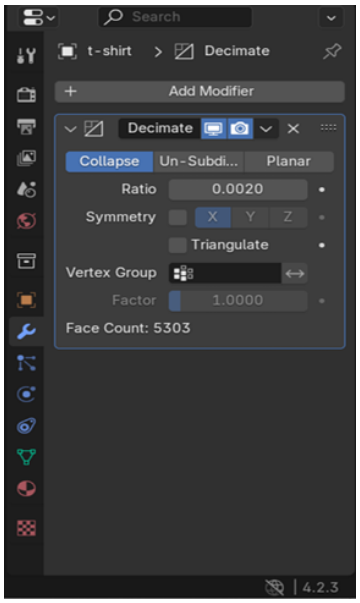

Figure 2.4: *Modifica della Mesh nel software Blender*

mesh tramite il modificatore Decimate in Blender, la simulazione in Unity è stata eseguita correttamente.

Il formato **.dae** (Collada) è supportato da Gazebo e risulta essere ampiamente utilizzato in questo contesto. Importando i file .obj della t-shirt e della scarpa, nonché il file .fbx della t-shirt in Blender, è stato possibile esportarli nel formato .dae. Questa operazione ha permesso di caricare correttamente i vari modelli in Gazebo, facilitando la simulazione degli indumenti nel simulatore.

In sintesi, l'utilizzo combinato di Blender per l'ottimizzazione delle mesh e la conversione dei formati, insieme all'applicazione del componente Cloth in Unity, ha consentito di ottenere simulazioni più realistiche del comportamento dei tessuti, migliorando l'interazione tra il robot e gli indumenti nel contesto del progetto.

Tra la documentazione di Unity è possibile trovare anche quella relativa al componente Cloth e in seguito verranno riassunti i passaggi fondamentali per utilizzarlo al meglio.

Per implementare una simulazione di tessuto in Unity, è necessario aggiungere il componente Cloth a un oggetto mesh. Ecco i passaggi fondamentali:

- **Preparazione della Mesh**: è importante assicurarsi che l'oggetto a cui si desidera applicare il tessuto abbia una mesh adeguata, e quindi, come visto in precedenza, anche un numero adeguato di poligoni della mesh.

- **Aggiunta del Componente**: una volta importato il file su Unity è sufficiente selezionare l'oggetto nella gerarchia di Unity e, nel pannello Inspector, cliccare su "Add Component", selezionare "Physics" e poi "Cloth" dalla lista dei componenti disponibili.

- **Configurazione**: Una volta aggiunto il componente, sarà possibile modificare vari parametri per ottenere una simulazione personalizzata e accurata.

Affinchè la maglietta abbia una movimentazione guidata dalla fisica, deve chiaramente avere una parte della maglietta fissata e che quindi non è soggetta alla gravità. Questi su Unity vengono chiamati "Cloth Constraint" e quindi ""vincoli del Cloth". Di seguito una descrizione dei passaggi per applicarli correttamente alla maglietta:

1. sul component Cloth si clicca su "Edit Cloth Constraint" come possiamo vedere dalla figura 2.5

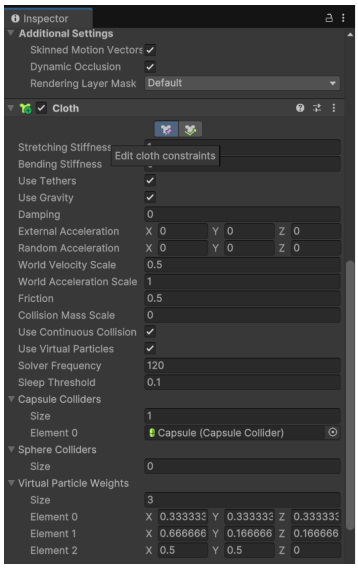

Figure 2.5: *Schermata Unity per l'applicazione dei vincoli*

2. successivamente si aprirà una schermata che permetterà di selezionare (tramite "Select") oppure colorare (tramite "Paint") le parti che saranno vincolate e quindi rimarranno maggiormente rigide. E' importante specificare una distanza minima nel parametro "Max Distance", in figura 2.6 è stato assegnato il valore 0.2 ed ha permesso un buon risultato.

Figure 2.6: *Schermata Unity per la selezione delle parti della mesh a cui applicare i vincoli*

Infine una descrizione delle principali proprietà del componente Cloth e di come sono state utilizzate nell'implementazione del tessuto realistico della maglietta:

- **Stretching Stiffness**: Determina la resistenza del tessuto all'allungamento. Valori più alti rendono il tessuto meno incline a estendersi. Un parametro accettabile per la maglietta è risultato compreso tra 0,5 e 0,7.
- **Bending Stiffness**: Controlla la rigidità alla flessione del tessuto. Un valore elevato riduce la capacità del tessuto di piegarsi. Un parametro accettabile per la maglietta è risultato compreso tra 0,7 e 0,9.
- **Use Tethers**: Applica vincoli che aiutano a prevenire che le particelle mobili del tessuto si allontanino troppo da quelle fisse, riducendo l'eccessiva elasticità.
- **Use Gravity**: Indica se la gravità deve influenzare il tessuto. Da applicare nel caso in cui si voglia vedere una movimentazione fisica della maglietta e condizionata dalla gravità.
- **Damping**: Coefficiente che determina quanto velocemente il movimento del tessuto si smorza nel tempo. Un parametro accettabile per la maglietta è risultato compreso tra 0,2 e 0,4.
- **External Acceleration**: Applica un'accelerazione costante esterna al tessuto, utile per simulare effetti come il vento.
- **Random Acceleration**: Introduce un'accelerazione casuale al tessuto, aggiungendo variazioni imprevedibili nel movimento.
- **World Velocity Scale**: Determina quanto il movimento in spazio globale dell'oggetto influisce sui vertici del tessuto.
- **World Acceleration Scale**: Controlla l'influenza dell'accelerazione globale dell'oggetto sui vertici del tessuto.
- **Friction**: Imposta il coefficiente di attrito del tessuto durante le collisioni. Un parametro accettabile per la maglietta è risultato compreso tra 0,5 e 0,7.
- **Collision Mass Scale**: Determina l'incremento di massa delle particelle durante le collisioni.
- **Use Continuous Collision**: Abilita la collisione continua per migliorare la stabilità delle interazioni.
- **Use Virtual Particles**: Aggiunge particelle virtuali per migliorare la stabilità delle collisioni.
- **Solver Frequency**: Specifica il numero di iterazioni del solver per secondo, influenzando la precisione della simulazione. Un parametro accettabile per la maglietta è risultato 60 Hz.

- **Sleep Threshold**: Definisce la soglia sotto la quale il tessuto entra in stato di "sonno", interrompendo la simulazione fino a nuove interazioni. Un parametro accettabile per la maglietta è risultato 0,1.
- **Capsule Colliders**: Array di collisori a capsula con cui il tessuto può interagire.
- **Sphere Colliders**: Array di coppie di collisori sferici con cui il tessuto può interagire.

2.3 Predisposizione dei Software nel PC

2.4 Interfacciamento ROS2-Unity

2.4.1 ROS-TCP Connector & ROS-TCP Endpoint

Il **ROS-TCP Endpoint** è il nodo ROS2 complementare al **ROS-TCP Connector**, che permette a Unity di comunicare con ROS tramite la rete. Questo nodo gestisce le connessioni TCP in ingresso provenienti da Unity e inoltra i messaggi ROS corrispondenti ai topic.

Installazione del ROS-TCP Endpoint

Per utilizzare il **ROS-TCP Endpoint**, è necessario installarlo all'interno del proprio workspace ROS2:

- **Clonare il repository**: `cd /tuo_workspace/src`
- **Compilare il workspace**: `cd /tuo_workspace`

```
colcon build --symlink-install
source install/setup.bash
```

Avvio del nodo

Per avviare il nodo **ros_tcp_endpoint**, bisogna eseguire il comando:

```
ros2 launch ros_tcp_endpoint endpoint.launch.py
```

In questo modo si avvia il server che ascolta sulla porta TCP (di default 10000) per ricevere le connessioni da Unity.

Verifica del nodo

Per assicurarsi che il nodo sia in esecuzione, è possibile utilizzare il comando:

```
ros2 node list
```

In questo modo il nodo **/ros_tcp_endpoint_node** dovrebbe essere visibile nella lista.

Verifica della connessione

Per testare la connessione tra Unity e ROS2 bisogna:

- **1.** Avviare ROS2 ed eseguire il nodo `ros_tcp_endpoint`.
- **2.** Eseguire la simulazione in Unity e verificare che non vengano generati errori di connessione.
- **3.** Utilizzare `ros2 topic echo` per controllare i messaggi pubblicati da Unity verso ROS2 e viceversa.

Questa configurazione permette di integrare Unity e ROS2 in un ambiente di simulazione realistico, abilitando lo scambio di dati tra il motore grafico e il framework robotico per la pianificazione e il controllo dei movimenti.

ROS-TCP Connector

Il **ROS-TCP Connector** è un pacchetto sviluppato da Unity Technologies che permette di mettere in comunicazione ROS2 e Unity attraverso una connessione TCP. Questa integrazione è fondamentale perché consente di inviare e ricevere messaggi tra i due ambienti, facilitando lo sviluppo e la simulazione di robot. Grazie a questo pacchetto, Unity diventa un ambiente in cui è possibile visualizzare i dati provenienti da ROS2, testare il controllo e verificare il comportamento del robot in scenari simulati prima di passare alla fase di implementazione su un sistema reale.

Il pacchetto ROS-TCP Connector include:

- **ROS-TCP Connector**: permette l'invio e la ricezione di messaggi tra Unity e ROS2.
- **VISUALIZATIONS PACKAGE**: utile per la visualizzazione dei messaggi in entrata e in uscita nella scena di Unity, facilitando il debug e l'analisi dei dati.

Inoltre, il pacchetto offre diverse funzionalità:

- **ROSConnection**: gestisce l'intero processo di comunicazione tra Unity e ROS2.
- **Message Generation**: permette di generare automaticamente classi in C# che rappresentano i messaggi ROS2, facilitando l'interazione tra i due ambienti.
- **Visualizations**: un set di API e configurazioni predefinite per rappresentare le informazioni scambiate.
- **ROSGeometry**: una serie di estensioni utili per la conversione delle geometrie tra Unity e altri sistemi che semplificano la compatibilità tra gli ambienti di simulazione.

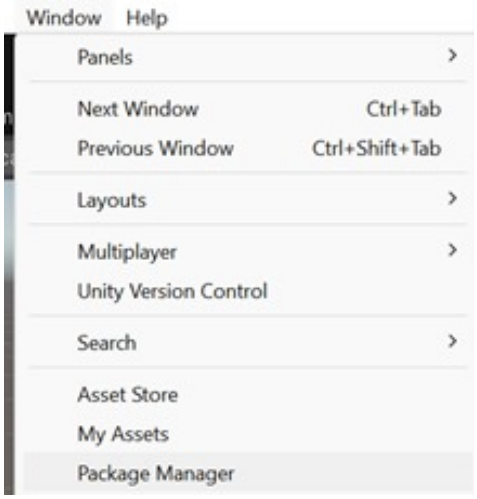

Figure 2.7: *Package Manager in Unity*

Installazione del ROS-TCP Connector

Per utilizzare il ROS-TCP Connector, è necessario installarlo all'interno di Unity e configurarlo correttamente affinché possa dialogare con ROS2.

Il pacchetto è disponibile su GitHub e può essere importato in Unity tramite il Package Manager. Ecco come fare:

- Aprire Unity e accedere al menu "Window", poi selezionare "Package Manager".
- Nella finestra del Package Manager, cliccare sul pulsante "+" in alto a sinistra e selezionare "Add package from git URL...".

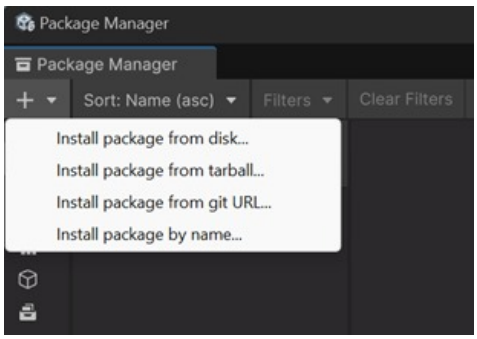

Figure 2.8: *Package Manager in Unity*

- Inserire l'URL del pacchetto da installare:
  - **ROS-TCP Connector**: `https://github.com/Unity-Technologies/ROS-TCP-Connector.git?path=/com.unity.robotics.ros-tcp-connector`
  - **Visualizations Package**: `https://github.com/Unity-Technologies/ROS-TCP-Connector.git?path=/com.unity.robotics.visualizations`

- In alternativa, se si ha una copia locale del pacchetto, è possibile installarlo seguendo la guida ufficiale di Unity per l'installazione di pacchetti locali.

Configurazione del ROS-TCP Connector

Dopo aver installato il pacchetto, è necessario configurarlo per far sì che possa comunicare con ROS2. Questo passaggio è essenziale per garantire che Unity e ROS2 possano scambiarsi dati senza problemi.

Dopo aver caricato l'oggetto, per configurare la comunicazione, bisogna:

- Accedere al menu "Robotics" e selezionare "ROS Settings".

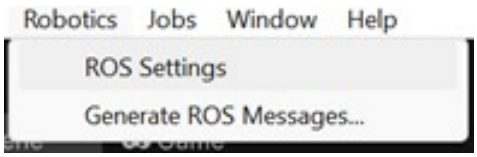

Figure 2.9: *Menu "Robotics" e "ROS Settings"*

- Nel campo "ROS IP Address", inserire l'indirizzo IP della macchina su cui è in esecuzione ROS2.

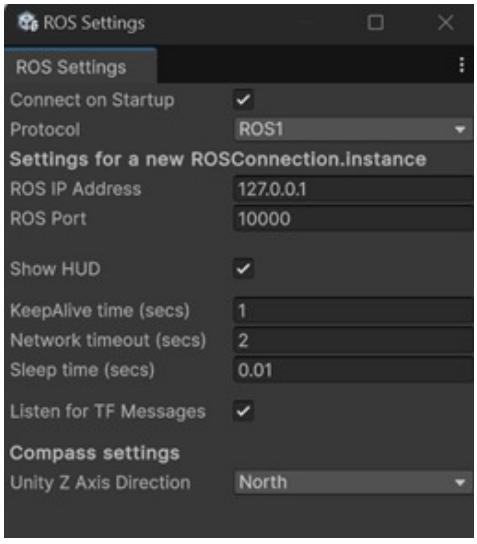

Figure 2.10: *Configurazione dell'indirizzo IP di ROS2*

Per reperire l'indirizzo IP di ROS (ROS IP Address) sulla macchina con WSL, è sufficiente aprire un terminale e digitare:

```
hostname -I
```

Comunicazione tra ROS2 e Unity

Dopo aver completato l'installazione e la configurazione, Unity sarà in grado di comunicare con ROS2 attraverso il protocollo TCP. Questo avviene tramite due modalità principali:

- **Pubblicazione di Messaggi**: Unity può inviare dati a ROS2 utilizzando il componente `ROSPublisher`.
- **Sottoscrizione di Messaggi**: Unity può ricevere dati da ROS2 grazie al componente `ROSSubscriber`.

Per rendere tutto operativo, è necessario avviare il nodo `ros_tcp_endpoint` all'interno di ROS2. Questo nodo funge da intermediario e garantisce che i messaggi possano fluire correttamente tra i due sistemi.

Testare la connessione tra Unity e ROS2

Una volta configurata la comunicazione, è importante verificare che tutto funzioni correttamente. Ecco alcuni passaggi per effettuare un controllo:

- **1. Avviare ROS2 ed eseguire il nodo `ros_tcp_endpoint`.**
- **2. Digitare il comando per configurare l'ambiente ROS:**

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

- **3. Avviare il nodo tramite il comando:**

  ```
  ros2 launch ros_tcp_endpoint endpoint.launch.py
  ```

  Questo avvierà il nodo `ROS-TCP Endpoint`, che ascolterà le connessioni da Unity sulla porta 10000 (di default).

- **4. È possibile verificare se il nodo è in esecuzione tramite il comando:**

  ```
  ros2 node list
  ```

2.4.2 Eseguire simulazione in Unity

- Utilizzare il comando `ros2 topic echo` per verificare se i messaggi vengono effettivamente inviati e ricevuti tra Unity e ROS2.

Se tutto è stato configurato correttamente, Unity e ROS2 saranno perfettamente integrati e potranno scambiarsi dati in tempo reale.

Per concludere, grazie a questa configurazione, sarà possibile avere uno scambio di dati per la pianificazione e il controllo dei movimenti.

2.4.3 URDF Importer

L'**URDF Importer** [4] di Unity è uno strumento che permette di importare file URDF (Unified Robot Description Format) direttamente in Unity, consentendo così di visualizzare e interagire con modelli robotici 3D all'interno dell'ambiente di simulazione. Questo strumento risulta particolarmente utile nel progetto perché permette di integrare ROS 2 e Unity tramite il file URDF fortemente utilizzato in robotica, inoltre semplifica la gestione della cinematica e della fisica dei robot senza dover ricostruire manualmente i modelli.

Il formato URDF è ampiamente utilizzato nel mondo della robotica per descrivere la struttura dei robot, inclusi i links, i joints, i materiali e i sensori. L'URDF Importer converte questi modelli in GameObjects di Unity, mantenendo la gerarchia dei componenti e assegnando i giunti come oggetti fisici controllabili.

Nel nostro progetto, il robot utilizzato è un Omron TM5-900, un braccio robotico con sei gradi di libertà. Il modello URDF di questo robot contiene:

- - Link per ogni segmento del braccio, dalla base all'end-effector.
- - Joints di tipo rotoidale che consentono il movimento.
- - Materiali per rappresentare visivamente il robot in Unity.
- - Collisioni e inerzia, che permettono una simulazione fisica realistica.

L'importazione di questo URDF in Unity è essenziale per poterlo visualizzare correttamente e per interagire con esso tramite ROS 2.

Per importare installare lo strumento URDF Importer è possibile seguire i seguenti passaggi:

- Cliccare su "Window" poi "Package Manager"

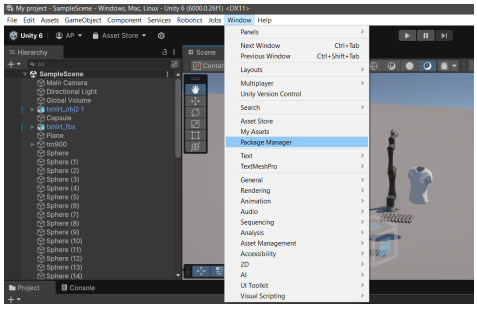

Figure 2.11: *URDF Importer Tutorial*

- Cliccare su "+" e selezionare "Add package from git URL..."

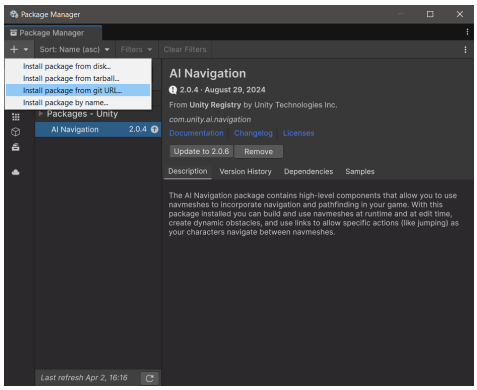

Figure 2.12: *URDF Importer Tutorial*

- Inserire il seguente URL della repository: `https://github.com/Unity-Technologies/URDF-Importer.git` e cliccare "install" per avviare l'installazione

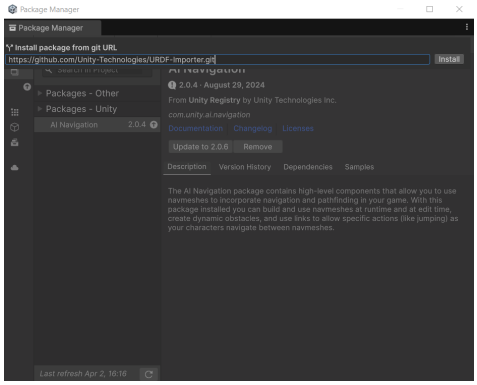

Figure 2.13: *URDF Importer Tutorial*

Per importare il modello URDF su Unity invece:

- Importare il file URDF e le varie mesh sia visual che collision nella cartella "Assets" del progetto Unity

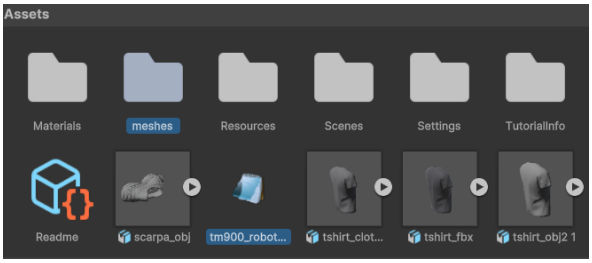

Figure 2.14: *URDF Importer Tutorial*

- Verificare che i vari path delle mesh all'interno del file URDF siano del tipo ""package://path/delle/meshes" come mostrato in figura 2.15

```
<link name="base_link">
  <visual>
    <geometry>
      <mesh filename="package://meshes/tm900/visual/Base.STL"/>
    </geometry>
    <material name="Grey">
      <color rgba="0.5 0.5 0.5 1.0"/>
    </material>
  </visual>
  <collision>
    <geometry>
      <mesh filename="package://meshes/tm900/collision/base.STL"/>
    </geometry>
  </collision>
  <inertial>
    <mass value="1.0"/>
    <insert_block name="origin"/>
    <inertia ixx="0.00110833289" ixy="0.0" ixz="0.0" iyy="0.00110833289" iyz="0.0" izz="0.0018"/>
  </inertial>
</link>
```

Figure 2.15: *URDF Importer Tutorial*

- Tasto destro sul file URDF e cliccare su "Import Robot from selected URDF file"

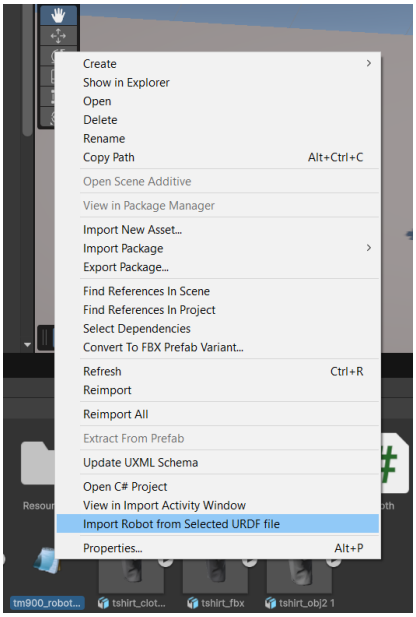

Figure 2.16: *URDF Importer Tutorial*

3 Analisi Preliminari e Problematiche Riscontrate

3.1 Problematiche sulle Compatibilità dei Software Utilizzati

3.2 Compatibilità dei File 3D con Unity e Gazebo

3.2.1 Componente Cloth e collisioni con altri materiali

Il Cloth di Unity oltre a realizzare una buona fisica per la maglietta è in grado di creare delle collisioni con altri materiali. In particolare, come è stato descritto nel capitolo precedente, si può inserire un oggetto Capsule o Sphere nel Cloth e quindi specificare alla maglietta quali e quanti oggetti dovranno avere un'interazione fisica opportuna durante la collisione. Tuttavia non è possibile far si che l'oggetto a cui è applicato il Cloth (in questo caso la maglietta) collida con un qualunque oggetto di qualunque forma geometrica. Infatti come anticipato è possibile utilizzare solo due tipi di oggetti 3D Unity: Capsule e Sphere. Questa restrizione rappresenta una problematica nel caso in cui il tessuto debba interagire con superfici piatte, angoli o forme geometriche più complesse, come un tavolo, una scatola o una struttura irregolare. Nel caso del progetto un'applicazione utile sarebbe quella di poter far interagire la maglietta con l'end effector del robot che poi riesca ad appoggiarla su una superficie piana come un tavolo. Poiché non è possibile definire direttamente colliders di tipo Mesh o Box per il Cloth, è necessario trovare soluzioni alternative per ottenere un comportamento realistico nelle simulazioni.
Una soluzione consiste nel posizionare una griglia di piccole Sphere Colliders sulla superficie con cui il tessuto deve interagire. In questo modo, il tessuto percepirà la presenza di una superficie continua, pur utilizzando solo colliders sferici. Per implementare questa soluzione:

- Creare uno script che generi automaticamente una matrice di sfere sulla superficie desiderata.
- Assicurarsi che le sfere vengano generate prima che il Cloth venga configurato, in modo che possano essere assegnate correttamente ai "Sphere Colliders" della maglietta.
- Ottimizzare il numero e la dimensione delle sfere per ottenere una buona resa senza sovraccaricare il motore fisico.

4 Struttura del workspace ROS2

5 Test Effettuati e Risultati Ottenuti

6 Conclusioni

6.1 Movimentazioe TM5-900 tramite Moveit

Introduzione a MoveIt **MoveIt** è una delle librerie più utilizzate per la pianificazione del movimento in ROS. Consente di controllare bracci robotici, eseguire pianificazioni di traiettoria, gestire collisioni e implementare strategie di manipolazione avanzate. Grazie alla sua integrazione con ROS 2, MoveIt permette di generare e simulare movimenti in modo efficiente, facilitando l'implementazione di algoritmi di pianificazione e controllo.

MoveIt utilizza il modello del robot descritto tramite URDF per comprendere la struttura cinematica del sistema e genera automaticamente il corrispettivo SRDF (Semantic Robot Description Format) contenente la generazione di movimenti validi evitando collisioni. Il suo framework è composto da diversi moduli, tra cui:

- Motion Planning: per la generazione di traiettorie.
- Collision Checking: per evitare urti tra il robot e l'ambiente.
- Kinematic Solvers: per calcolare i movimenti articolari necessari a raggiungere una determinata posizione.
- Trajectory Execution: per inviare comandi al robot fisico o alla simulazione.

Nel nostro progetto, MoveIt è stato utilizzato per generare i file necessari alla movimentazione del robot TM5-900. In particolare, è stato impiegato lo strumento **MoveIt Setup Assistant**, che ha permesso di: 1. Importare l'URDF del robot: il modello 3D e la cinematica del TM5-900 sono stati letti direttamente dal file URDF. 2. Generare il file SRDF: che definisce i gruppi cinematici, i giunti, i link e le configurazioni di pianificazione. 3. Configurare i plugin di controllo: per connettere MoveIt a ROS 2 e inviare comandi al robot. 4. Definire le aree di collisione: per garantire che il robot si muova senza urtare gli ostacoli. 5. Generare automaticamente i pacchetti di configurazione: che includono i parametri per il controllo e la simulazione del TM5-900.

Grazie a questa configurazione, MoveIt è stato in grado di elaborare e inviare traiettorie di movimento al robot sia nella simulazione in Unity che in ROS 2.

Applicazione: Definizione di Pose e Simulazione in Unity

Dopo la configurazione iniziale, il nostro obiettivo è stato quello di definire due pose specifiche per il TM5-900 e inviarle alla simulazione in Unity. Il processo seguito è stato il seguente:

1. **Definizione delle pose**: Abbiamo selezionato due posizioni target per il braccio robotico, specificando le coordinate spaziali e gli angoli dei giunti. 2. **Utilizzo di MoveIt per calcolare la traiettoria**: MoveIt ha generato una traiettoria valida evitando collisioni. 3. **Invio dei comandi a ROS 2**: La traiettoria calcolata è stata inviata al nodo ROS 2 responsabile del controllo del robot. 4. **Simulazione in Unity**: I dati di movimento sono stati trasmessi a Unity, dove il modello del TM5-900 si è mosso in base ai comandi ricevuti.

Grazie a questo workflow, siamo riusciti a controllare il robot in modo realistico, testando la movimentazione prima di applicarla in un ambiente reale. L'integrazione tra MoveIt, ROS 2 e Unity ha reso possibile una simulazione efficace, permettendoci di validare il comportamento del TM5-900 prima della sua implementazione fisica.

6.2 Importazione TM5-900 in Gazebo

6.3 Camera

6.4 Applicazione finale

7 Risultati ottenuti e discussione

8 Conclusioni e prospettive future

A Appendice1

Se necessario ricorrete alle appendici per spiegare le parti "di contorno" dell'attività svolta e/o ciò che non riuscite ad inserire nello schema generale dei capitoli della relazione (es acquisizione dei dati con Matlab).

Bibliografia

[1] "Ros2." https://docs.ros.org/en/humble/index.html.

[2] "Unity." https://docs.unity.com/.

[3] "Manual cloth unity." https://docs.unity3d.com/Manual/class-Cloth.html.

[4] "Urdf-importer." https://github.com/Unity-Technologies/URDF-Importer.