

Indice

Elenco delle figure	III
1 Generalità dei software usati in ROS e del pacchetto Sw2urdf	1
1.1 Concetti base Ros	1
1.2 Rviz	5
1.3 Pacchetto Solidworks Sw2urdf	8
2 Installazione e configurazione dei software	11
2.1 Installazione di ROS Kinetic	12
2.1.1 Installazione rosinstall	13
2.2 Creazione di un ROS Workspace	14
2.3 Creazione di un ROS package	15
2.4 Installazione di Rviz	18
2.5 Installazione pacchetto Sw2urdf	18
3 Specifiche umano	21
3.1 Scelta del modello	21
3.2 Possibili strade	22
3.3 Articolazioni	29
3.3.1 Limiti articolazioni	30
4 Creazione modello	35
4.1 SolidWorks	35

4.2	Da SolidWorks a ROS	36
4.2.1	Concetti sul file URDF e SDF	36
4.2.2	SolidWorks to URDF Exporter	42
4.2.3	Dati inseriti per il file URDF dell'umano	49
4.2.4	Il giunto sferico	50
4.3	Procedura di apertura in Rviz	52
4.3.1	Apertura in Rviz	52
5	Inserimento in uno spazio aperto	55
5.1	I cobot	55
5.2	Lo spazio aperto	56
	Bibliografia	61

Elenco delle figure

1.1	Logo ROS	1
1.2	Esempio di Nodi e topic	2
1.3	Schema comunicazione in ROS	5
1.4	Logo Rviz	5
1.5	Schermata di RVIZ	7
1.6	Schermata pacchetto Sw2urdf	9
2.1	Albero delle dipendenze del WorkSpace catkin	15
2.2	Albero delle dipendenze, con un package	17
3.1	Modello che alcuni pacchetti proponevano	23
3.2	Modello umano di MakeHuman	25
3.3	Possibili umanoidi	26
3.4	Modello umano scelto	29
3.5	Sistema di riferimento usato	32
3.6	Limiti articolazioni	34
4.1	Modello di un link	37
4.2	Codice per definire un link	38
4.3	Esempio grafico joint	39
4.4	Codice d'esempio per un joint	40
4.5	apertura dell'export(4)	46
4.6	Property manager(5)	47
4.7	configure joint properties(6)	48
4.8	configure link properties(6)	48

4.9	cartella URDF(7)	49
4.10	Umano con sistemi di riferimento concordi	49
4.11	Umano con assi di riferimento inseriti	50
4.12	Confronto tra le due parti del corpo, con e senza modifica	51
4.13	Esempio assi per giunto sferico	52
4.14	Schermata terminale d'avvio Rviz	53
4.15	Schermata di avvio Rviz	54
4.16	Schermata Rviz con umano	54
5.1	Spazio aperto realizzato da C.N.R di Milano	56
5.2	Mosaico che uomo e robot devono realizzare	57
5.3	Rappresentazione grafica della simulazione dello spazio aperto	57
5.4	Spazio aperto aggiornato	58

Capitolo 1

Generalità dei software usati in ROS e del pacchetto Sw2urdf



Figura 1.1: Logo ROS

1.1 Concetti base Ros

ROS è progettato per essere un sistema liberamente accoppiato in cui un processo è chiamato nodo e ogni nodo è responsabile di un compito. I nodi comunicano tra loro utilizzando messaggi che passano attraverso canali logici chiamati topics. Ogni nodo può inviare o ottenere dati da un altro nodo utilizzando il modello di publisher/subscriber, nella fig. 1.2, è possibile vedere come la comunicazione avviene, in modo particolare l'immagine è stata presa dal seguente tutorial: *ROS tutorial 2: Publishers*

and subscribers.¹

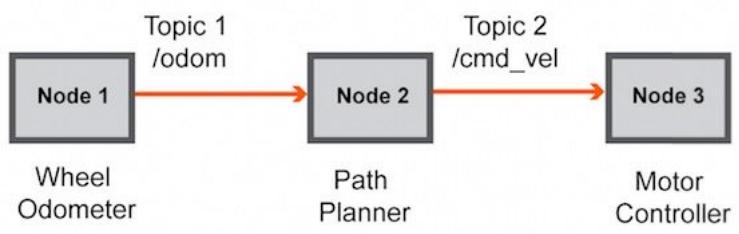


Figura 1.2: Esempio di Nodi e topic

I motivi principali per cui ROS è in continua crescita sono dati da:

- **l'architettura:** risulta essere disposta su più livelli, i principali sono :
 - **ros:** che offre servizi di basso livello quali:
 - * software per il controllo fisico di un robot;
 - * astrazione dell'hardware di ogni singolo robot attraverso un'interfaccia comune a tutti i robot;
 - * driver per la gestione di diversi sensori;
 - * strumenti per la comunicazione fra diverse applicazioni robotiche, tramite messaggi;
 - * gestione automatizzata dei pacchetti.
 - **ros-pkg:** insieme di pacchetti che permettono di implementare diverse funzionalità come la percezione, la simulazione e alcuni algoritmi di planning.
- **Controllo di diversi robot:** ROS, grazie ai pacchetti che si possono trovare sui diversi siti ufficiali e non, permette di poter controllare diversi robot tra loro, in modo rapido e semplice è possibile quindi utilizzarli nel proprio progetto ampliando le funzionalità e l'efficienza dei nuovi macchinari o di quelli già installati;

¹ROS tutorial 2: Publishers and subscribers. URL: <https://www.youtube.com/watch?v=bJB9tv4ThV4>.

Livelli di comunicazione in ROS

ROS, per occuparsi del passaggio dell'informazioni fra i diversi processi in esecuzione, i quali sono collegati tra loro tramite una rete peer to peer, utilizza il **ROS Computational Graph Level**. Esso è composto da alcuni elementi che vengono spiegati in seguito, per poterne comprendere l'utilità e il funzionamento. I principali sono:

- **nodes**: rappresentano ogni singolo processo che può essere fatto all'interno di ROS a livello computazionale, ovvero indicano una possibile funzionalità che posso utilizzare durante la progettazione, simulazione e movimentazione del robot . Tali elementi possono collegarsi tra loro, permettendo lo scambio di dati e la collaborazione tra più funzioni, per fare ciò vengono utilizzati i topics. Si possono individuare due tipi di nodi:
 - publishers : nodi che permettono di inviare e generare topics, con all'interno dei messaggi, in cui possono trovarsi informazioni o comandi che il nodo ricevente può utilizzare o eseguire;
 - subscribers : nodi che hanno la caratteristica di seguire determinati topics che vengono prodotti durante la comunicazione tra i nodi, perché ne possono essere interessati per una loro funzione.

NB: un singolo nodo può essere entrambi i tipi.

- **topics**: rappresentano il mezzo di comunicazione che viene usato tra i nodi. Sono dotati di alcune caratteristiche, le principali sono:
 - l'unidirezionalità : specifica il fatto che i topics non permettano la comunicazione di tipo request/response ma di subscribers/publishers;
 - tipizzazione : descrive il fatto che nei singoli topic è possibile pubblicare/ leggere un solo tipo di messaggio.
- **messages**: individuano una rigida struttura di dati, composta da un preciso numero di dati e di tipo di campi scelti tra quelli primitivi (integer, float, boolean

,ecc.). La caratteristica principale è data dal fatto che supportano il tipo di dato array, che permette di raggruppare più dati in un unico messaggio. Oltre all'utilizzo che viene fatto per i nodi possono essere utilizzati anche nei servizi;

- **services:** sono creati per poter supportare il tipo di comunicazione request/response, che funziona in modo semplice utilizzando un messaggio per la richiesta, il quale viene inviato da un nodo che necessita di un servizio a quello che lo mette a disposizione, e di un ulteriore messaggio di risposta che percorre il viaggio nel senso opposto, in modo da rendere disponibile il servizio al richiedente;
- **bag:** rappresenta il formato di file che viene utilizzato per immagazzinare i dati presenti nei messaggi. Sono caratterizzati dal fatto che possono essere immagazzinati e riprodotti anche quando i dispositivi hardware non sono online, questo permette di poterli consultarli in fase di sviluppo, aiutando il programmatore a risolvere i problemi che si possono riscontrare;
- **il nodo master:** è quello di maggior importanza tra quelli di ROS, permette di gestire le registrazioni ai diversi servizi e della loro nomenclatura oltre a tenere traccia del tipo del nodo utilizzato (se publisher o subscriber), in modo da non avere problemi di comunicazione durante il funzionamento dell'applicazione che si sta svolgendo.

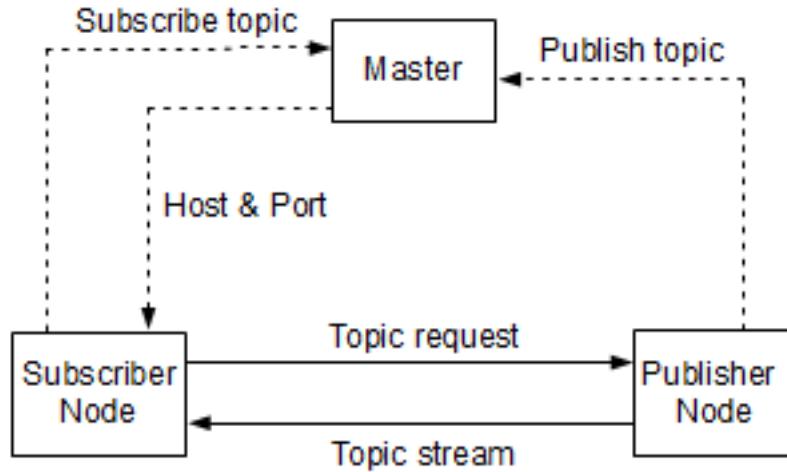


Figura 1.3: Schema comunicazione in ROS

1.2 Rviz

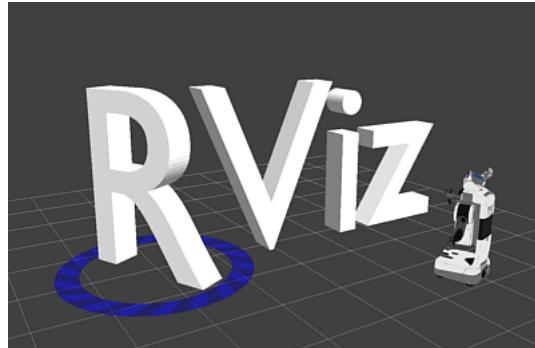


Figura 1.4: Logo Rviz

RVIZ è un software di visualizzazione di modelli tridimensionali che viene molto utilizzato per la simulazione di un robot e per verificarne la correttezza dimensionale e di movimentazione nello spazio. Bisogna sottolineare che tale software non permette la modifica in real time della fisica del modello, ma solo la visualizzazione e la renderizzazione, in base al file URDF del modello, che verrà spiegato successivamente.

Una volta avviato ci si ritrova con una schermata (fig. 1.5), divisa in più parti, dove la visualizzazione del modello ne fa da padrone, ma ce ne sono anche altre, tra cui:

- **Display:** Composta da una finestra in cui vengono visualizzati tramite un elenco

tutti gli oggetti presenti nella schermata principale. Tramite tale finestra è possibile aggiungere o rimuovere altri display. I principali messi a disposizione da RVIZ sono:

- GlobalOptions, in cui si trovano le specifiche generali del mondo tridimensionale creato in RVIZ;
 - Grid, che permette di modificare la base del mondo, sulla quale si potrebbe sviluppare la simulazione o la visualizzazione;
 - Robot Model, che si occupa della visualizzazione del modello del robot che si vuole inserire;
 - TF, che si occupa della visualizzazione dell'albero delle trasformate tf, ovvero indica come sono messi in correlazione i giunti tra loro, seguendo quello che viene realizzato nel file URDF che verrà discusso in seguito. Tale finestra permette inoltre di vedere la posizione del sistema di riferimento (Axis) XYS connesso, seguendo la logia RBG , in cui rispettivamente gli assi saranno colorati: rosso per la X, blu per la Y, verde per la Z.
- **Tool properties:** è un pannello che permette di associare una posizione di partenza e una posizione obiettivo ai due rispettivi topic di ROS;
 - **Views:** è una finestra che permette di scegliere quale vista utilizzare per la visualizzazione del modello. Le principali sono :
 - Orbit : comunemente detta camera orbitale, permette di fissare un fuoco attorno al quale ci si può ruotare;
 - FPS : indica la visione in prima persona, ovvero permette di vedere la scena nello stesso modo che avviene quando una persona ruota la testa;
 - TopDownOrtho : permette la visualizzazione del mondo lungo l'asse Z rispettivo al frame del robot.

- **Selection:** è un pannello che permette di visualizzare la descrizione dei link, in particolare permette di vedere l'orientamento e la posizione dall'origine del sistema di riferimento fisso del mondo.

Pacchetto TF

Il pacchetto TF è molto importante nelle applicazioni robotiche perché permette di conoscere nel tempo le posizioni dei vari giunti del robot, ovvero permette di definirne i movimenti che sono avvenuti tra di loro durante una simulazione. Il modo in cui lavora è il seguente: associa ad ogni singolo giunto un frame, che a sua volta rappresenta un sistema di riferimento. Tutti i singoli frame vengono aggiunti ad un elenco ad albero che permette di visualizzare le varie relazioni tra essi, che vengono definite da due tipi di dati, in particolare da un vettore che rappresenta la distanza che i vari giunti hanno e la matrice di rotazione che definisce come i frame sono posizionati tra loro. Tali dati vengono immagazzinati, in modo da visualizzarli in tempo reale per ogni frame, così da controllarli, per verificarne la correttezza.

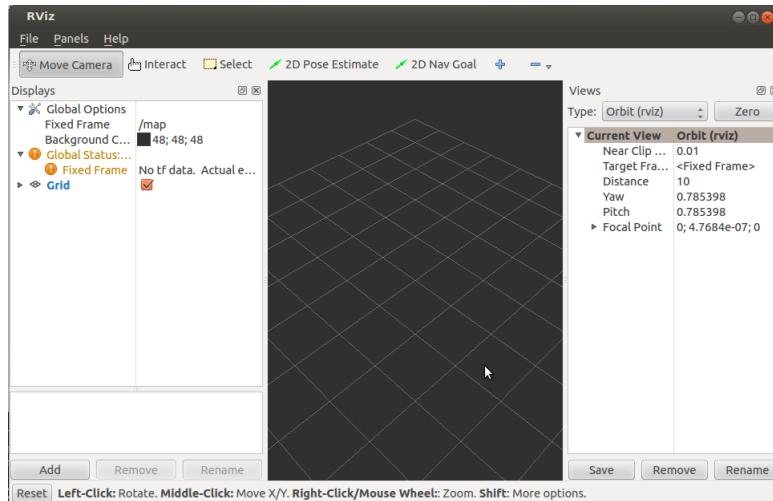


Figura 1.5: Schermata di RVIZ

Altro software : Gazebo

Gazebo è un simulatore 3D open source con la capacità di simulare robot, sensori e oggetti vari. È progettato per aiutare i ricercatori che lavorano con veicoli robotizzati in ambienti esterni ma è in grado anche di gestire ambienti interni. Gazebo utilizza il motore fisico ODE (Open Dynamics Engine). Per il rendering della grafica tridimensionale invece, Gazebo sfrutta OGRE (Object-Oriented Graphics Rendering Engine) e assicura un buon grado di realismo generando correttamente luci e ombre dell'ambiente. All'interno del simulatore possono essere caricati vari tipo di oggetti, da semplici forme come cubi e sfere, a modelli complessi come edifici o animali. Ogni oggetto ha le proprie caratteristiche: massa, velocità, inerzia e numerose altre proprietà fisiche e di visualizzazione in modo da rendere la simulazione la più realistica possibile. È inoltre disponibile un database di robot gestito dalla community che tutti possono utilizzare e modificare a proprio piacimento. Gazebo può anche generare i dati di diversi sensori.

1.3 Pacchetto Solidworks Sw2urdf

Il pacchetto Sw2urdf è un componente aggiuntivo di SolidWorks che consente l'esportazione di un modello da Solidworks in URDF. Tale strumento permette di generare un cartella che è simile a quella che verrebbe creata in ROS per fare la stessa operazione, ma in modo più semplice e con una geometria che può essere molto più complessa rispetto a quella che potrebbe essere fatta con alcuni software di ROS o scrivendo manualmente l'urdf . Essa contiene una directory per mesh, frame e robot (file urdf). Se si vuole trasformare un assebly di Solidworks, caratterizzato da più parti che hanno relazioni geometriche tra loro, il pacchetto permetterà di inserirle seguendo la logica di ROS, definendo le singole parti in link e le relazioni che ci sono tra esse come joint. Una volta svolta correttamente la fase di aggiunta delle singole parti e definite le relazioni tra di esse, è possibile generare direttamente la cartella che permetterà l'utilizzo dell'assembly in ambiente ROS. I principali software utilizzati per visualizzare o movimentare il lavoro fatto tramite il pacchetto sono GAZEBO o RVIZ . Bisogna

1.3 – Pacchetto Solidworks Sw2urdf

sottolineare che in fase di inserimento delle varie relazioni, esse possono essere definite in modo automatico dal pacchetto o possono essere inserite manualmente. La seconda strada, quando la geometria è complicata, è quella consigliata.

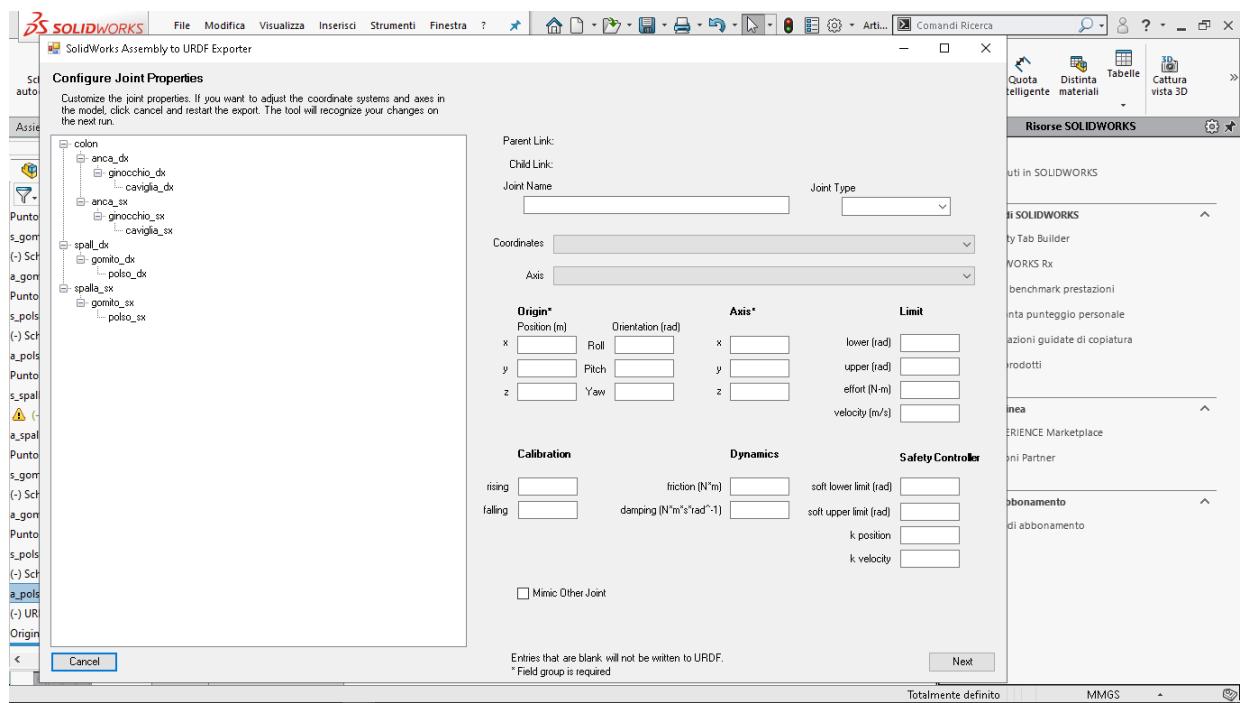


Figura 1.6: Schermata pacchetto Sw2urdf

Capitolo 2

Installazione e configurazione dei software

Per questo progetto si è scelto di utilizzare una combinazione di distribuzioni che permettevano di svolgere correttamente il lavoro, utilizzando il SO linux, in quanto ROS è ancora in fase di sperimentazione su Windows e MacOS. Le distribuzioni usate sono le seguenti :

- Ubuntu 16.04;
- ROS Kinetic;
- Rviz (ultima versione supportata da ROS kinetic).

Dopo aver installato il SO open-source, seguendo la linea guida presente sul sito ufficiale di Ubuntu (*Ubuntu*¹), si è passati ad installare ROS seguendo il tutorial presente su sito ufficiale (*ROS*²), scegliendo come distribuzione tra quelle presenti kinetic.

¹ *Ubuntu*. URL: <https://ubuntu.com/>.

² *ROS*. URL: <https://www.ros.org/>.

2.1 Installazione di ROS Kinetic

I passi che permettono di installare ROS Kinetic sono i seguenti (verranno inseriti i principali, per una spiegazione più dettagliata sarà possibile seguire quella indicata nel sito principale indicato precedentemente):

1. Configurare le repositories di Ubuntu, seguendo il tutorial presente al sito : *Repositories Ubuntu*.³
2. Sistemare il computer al fine di accettare i software da packages.ros.org, con il seguente comando lanciato da terminale :

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc)  
main" > /etc/apt/sources.list.d/ros-latest.list'
```

3. Impostare le chiavi:

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key  
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

Se si sta utilizzando un proxy server si dovrà utilizzare al posto del comando apt-get . . . :

```
curl -sSL 'http://keyserver.ubuntu.com/pks/lookup?op=get&search=  
0xC1CF6E31E6BADE8868B172B4F42ED6FBAB17C654'
```

e

```
sudo apt-key add -
```

4. Iniziare l'installazione (si installerà la versione completa ma in base alle proprie esigenze si hanno diverse possibilità d'installazione, si riporta al sito dell'installazione per maggiori informazioni):

³*Repositories Ubuntu.* URL: <https://help.ubuntu.com/community.Repositories/Ubuntu>.

```
sudo apt-get update
```

Aggiunto a :

```
sudo apt-get install ros-kinetic-desktop-full
```

5. Impostare l'ambiente con il seguente comando, per poter richiamare le risorse di ROS all'interno di ogni terminale :

```
echo "source /opt/ros/kinetic/setup.bash" » ./bashrc
```

e

```
source ./bashrc
```

Se si hanno più versioni di ROS installate bisognerà seguire un'altra procedura, da seguire sul sito, inoltre se si vorranno fare ulteriori modifiche, è consigliato guardare quelle interessate e lanciare il comando ad esse associate.

6. Inizializzazione di Rosdep con la seguente sequenza :

```
sudo apt install python-rosdep
```

e

```
sudo rosdep init
```

e

```
rosdep update
```

7. Ora, è possibile verificare la propria istallazione con i tutorial base proposti.

2.1.1 Installazione rosinstall

Si tratta di uno strumento molto utile per ROS perché permette di installare e scaricare da riga di comando diverse suorce trees per i package di ROS.

```
sudo apt-get install python-rosinstall
```

2.2 Creazione di un ROS Workspace

Un WorkSpace è uno spazio protetto entro il quale è possibile lavorare utilizzando ROS. In particolare è consigliato utilizzare il catkin Workspace, che permette di generare una cartella, contenente delle sottocartelle che permettono di utilizzare ROS senza problemi. Qui di seguito viene illustrata la sequenza per poterlo installare e utilizzare:

```
mkdir -p /catkin_ws/src
```

unito a

```
cd /catkin_ws/
```

e poi,

```
catkin_make
```

Le cartelle che vengono create sono:

- src (source space), cartella in cui devono essere salvati i progetti ROS che si realizzano, per il corretto funzionamento di tutti i comandi di ROS;
- devel(development space);
- build (build space);

Il loro percorso può essere rappresentato graficamente con il seguente diagramma (fig. 2.1) :

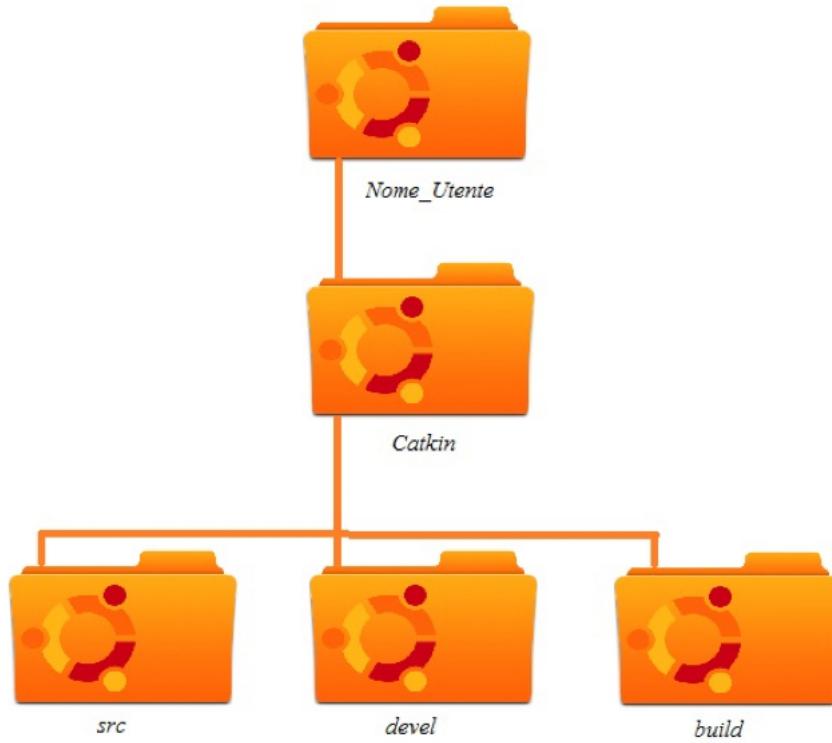


Figura 2.1: Albero delle dipendenze del WorkSpace catkin

2.3 Creazione di un ROS package

Per introdurre tale argomento è necessario definire cosa si definisce un package in ROS. Affinché un package possa essere considerato tale, deve almeno soddisfare alcuni requisiti:

- deve contenere un file `package.xml` conforme a catkin (Il file `package.xml` fornisce meta informazioni sul pacchetto);
- deve contenere un `CMakeLists.txt` che utilizza catkin;
- Se è un metapackage catkin, deve avere il file `CMakeLists.txt` pertinente;
- Ogni pacchetto deve avere la propria cartella, Ciò significa che nessun pacchetto annidato né più pacchetti condividono la stessa directory.

Per poter creare un package è possibile seguire la seguente sequenza:

1. Entrare nel WorkSpace

```
cd /catkin_ws/src
```

Dopo avere verificato di star lavorando nella cartella src del catkinWorkspace, si può procedere con il comando successivo

2. Creazione del pacchetto , in questo caso beginnerTutorials (è comunque possibile scegliere il nome arbitrariamente, secondo le proprie esigenze) con il seguente comando:

```
catkin_create_pkg beginnerTutorials std_msgs rospy roscpp
```

Dopo avere controllato la presenza della cartella del pck all'interno del Workspace;

3. Dalla cartella principale del workspace, bisogna lanciare il seguente comando:

```
catkin_make
```

A questo punto l'albero delle dipendenze sarà il seguente fig. 2.2(nel caso in cui si ha un solo package, chiamato package-name)

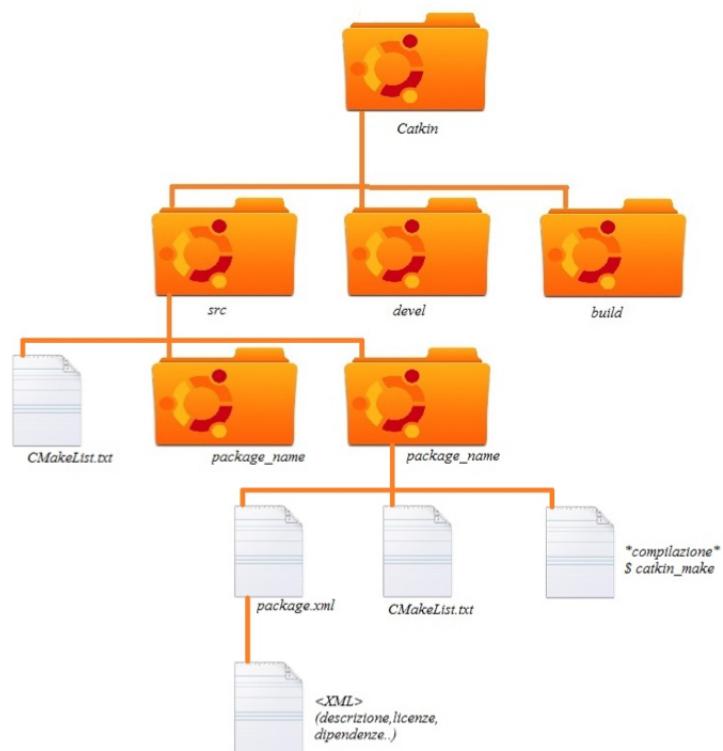


Figura 2.2: Albero delle dipendenze, con un package

2.4 Installazione di Rviz

L'installazione di Rviz, avviene anche lei seguendo una sequenza di comandi da lanciare nel terminale. I principali sono i seguenti:

- Scaricare un pacchetto:

```
sudo apt-get install ros-kinetic-rviz
```

- Installazione:

```
rosdep install rviz
```

e

```
rosmake rviz
```

A questo punto, terminati correttamente tali passaggi è possibile avviarlo tramite:

```
roscore
```

e successivamente:

```
rosrun rviz rviz
```

2.5 Installazione pacchetto Sw2urdf

L'installazione di tale pacchetto a differenza degli altri, è meno complicata. Si tratta di scaricare il file zip dal seguente sito : *Releases Sw2urdf*.⁴ Una volta scaricato e installato, è possibile utilizzarlo all'interno di SolidWorks, spuntandolo nella finestra add-ins, ed una volta selezionato, per poterlo utilizzare bisogna andare nella finestra File alla voce export to URDF.... Si sottolinea, che tale pacchetto è presente solo

⁴*Releases Sw2urdf*. URL: https://github.com/ros/solidworks_urdf_exporter/releases.

per SolidWorks, che attualmente supporta solo il sistema operativo Windows, dunque per poterlo utilizzare, trasportando la cartella che viene generata da Windows a ROS(presente oggi su linux), bisognerà utilizzare una strada che permetta questo passaggio, le più consigliate sono:

- creare una partizione in Windows, dove poter far funzionare linux;
- utilizzare una macchina virtuale.

Capitolo 3

Specifiche umano

L'umano, nella sua completezza, è un modello che necessita di differenti informazioni per poter essere realizzato fedelmente alla realtà. In questo capitolo si tratteranno le specifiche scelte per il modello, i tipi di articolazioni, i limiti delle articolazioni e le possibili strade che si sarebbero potute intraprendere per sviluppare il progetto.

3.1 Scelta del modello

La scelta del modello è stata il punto crociato del lavoro, poiché si è basata sulle richieste del consiglio nazionale delle ricerche (C.N.R) della sede di Milano, che era determinato a cambiare il prototipo che stava utilizzando per dare una maggior impronta di realtà allo studio sulla collaborazione tra uomo e macchina all'interno di una cella collaborativa¹.

Il modello da loro utilizzato era di un umano composto da forme geometriche semplici, ovvero da sfere, cilindri e rettangoli, e non da forme anatomiche, più reali e complicate da realizzare.

¹un sistema completo che comprende un robot collaborativo, le sue parti terminali (spesso intercambiabili), eventuali periferiche, una gabbia perimetrale (all'interno della quale il robot opera) e un sistema di controllo

Il voler abbandonare tale modello ha portato a considerarne uno che rispettasse le forme antropomorfiche, allontanandosi dal modello puramente geometrico, pur riuscendo a mantenere le movimentazioni, oltre a non appesantire la renderizzazione e la visualizzazione tramite software come Gazebo o Rviz. Le specifiche inizialmente imposte si possono riassumere in:

1. Semplicità d'uso, intesa come facilità nella movimentazione senza ricadere in problemi di renderizzazione e visualizzazione, dunque senza troppi dettagli che potessero appesantirne l'utilizzo;
2. Utilizzo di forme anatomiche, al posto di quelle geometriche;
3. Giunti che delle articolazioni della spalla e del polso in grado di permettere almeno due rotazioni attorno all'asse x e y, secondo il sistema di riferimento in fig. 3.5;
4. Possibilità di utilizzare al momento della simulazione un software tra Gazebo, Rviz e Moveit².

3.2 Possibili strade

Dopo un'attenta ricerca svolta utilizzando le risorse che ROS mette a disposizione, ovvero blog, forum e tutorials, si erano definite diverse strade, che di seguito saranno spiegate brevemente elencandone i pro e i contro, ed aggiungendo la compatibilità con le specifiche. Le principali strade che si potevano percorrere, erano:

- **Strada 1:** Trovare un pacchetto con il modello di un umano che rispettasse le nostre specifiche.
 - pro: guadagno di una grande quantità di tempo ed energie da investire in altre funzioni e in un progetto più complicato;

²Piattaforma di manipolazione robotica open source di facile utilizzo per lo sviluppo di applicazioni commerciali, progetti di prototipazione e algoritmi di benchmarking. Tale software permette di integrare tramite plug-in anche i lavori svolti negli altri 2 software

- contro: assenti, perché si avrebbe trovato tutto ciò che era stato richiesto.

Tale strada non è stata resa possibile perché non si è e trovato un pacchetto simile, dato dal fatto che la maggior parte dei pacchetti che si possono trovare online consideravano un modello di unano di tipo geometrico come quello raffigurato nella fig. 3.1. Sintetizzando, la compatibilità con le specifiche risulta:

1. ° : Totalmente verificata, perché se si fosse trovato un pacchetto già realizzato, non si sarebbero avuti problemi di utilizzo in quanto la maggior parte dei pacchetti presenti in rete sono verificati e corretti da un gran numero di utenti, i quali possono apportare le modifiche necessarie a renderlo efficiente e a migliorarlo continuamente;
2. ° : Totalmente verificata perché sarebbe un obiettivo del pacchetto;
3. ° : Totalmente verificata, perché sarebbe stata una specifica del modello;
4. ° : Totalmente verificata, perché si sarebbe cercato un pacchetto compatibile con i software voluti.

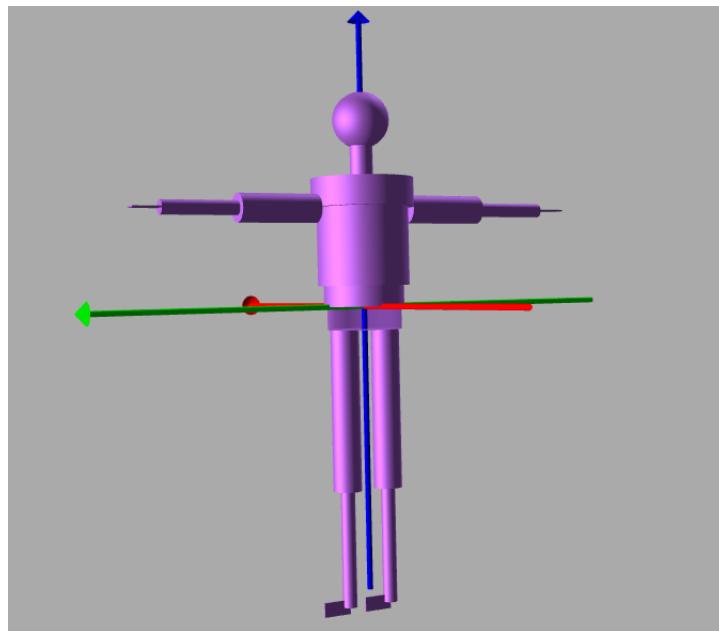


Figura 3.1: Modello che alcuni pacchetti proponevano

- **Strada 2:** tramite un software chiamato MakeHuman³ (sito *Make Human Community*⁴) creare il modello di un umano in modo casuale per poi convertire il file che lo descrive dal formato .dae⁵ al formato .urdf, che sarebbe quello che ci avrebbe permesso di utilizzarlo in ROS tramite il seguente tutorial :*Make Human Tutorials*.⁶
 - pro: ottima accuratezza al dettaglio, ovvero il modello che viene generato rispetta in modo inequivocabile il corpo umano;
 - contro: non è possibile riuscire a movimentarlo a piacimento perché non risulta esserci la divisione delle diverse parti del corpo come joint e link⁷, ma si utilizza l'umano come un unico pezzo e la movimentazione può avvenire con altri software diversi da quelli richiesti; come ad esempio Blender⁸. Inoltre, dati gli elevati dettagli si potrebbero riscontrare problemi di renderizzazione. Il modello che ne risulterebbe, potrebbe essere quello nella fig. 3.2 (se ne trovano diversi online anche con altre caratteristiche, quello rappresentato qui è quello di partenza).

Riassumendo, la compatibilità con le specifiche risulta:

1. ° :Non verificata, perché l'elevata cura al dettaglio avrebbe provocato difficoltà di visualizzazione e una difficoltà nell'utilizzo;

³software open-source che permette di creare umani, modificando qualsiasi parte del corpo a piacimento

⁴*Make Human Community*. URL: <http://www.makehumancommunity.org/>.

⁵I file di tipo .dae sono impostati in base al noto formato COLLADA. L'elaborazione di questi COLLADA è “COLLAborative Design Activity”. I file .dae sono dei file in formato tridimensionale usati per passare tra vari software di grafica.

⁶*Make Human Tutorials*. URL: <https://github.com/makehumancommunity/makehuman/issues/19>.

⁷logica che è presente nella creazione di un file URDF, per ulteriori informazioni si riporta al capitolo 5

⁸Blender è la suite di creazione 3D gratuita e open source. Supporta l'intera pipeline 3D: modellazione, rigging, animazione, simulazione, rendering, compositing e motion tracking, editing video e pipeline di animazione 2D.

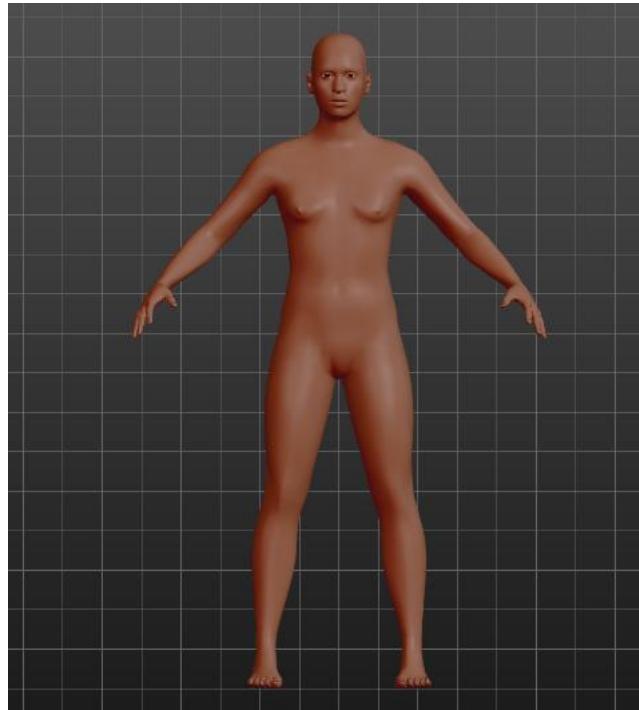


Figura 3.2: Modello umano di MakeHuman

2. ° : Totalmente verificata perché tale software si occupa di questo obiettivo;
 3. ° : non del tutto verificata, perché non è ben spiegato come avviene la movimentazione;
 4. ° : non del tutto verificata, perché il software che avremmo utilizzato sarebbe stato diverso da quelli imposti.
- **Strada 3** : utilizzare un umanoide⁹, tra cui Nao, Poppy, atlas o baxter, rappresentati nella fig. 3.3 (sul sito di ROS è possibile trovarne molti altri, che sono stati progettati con tale piattaforma), anziché un umano.
 - pro: dato il fatto che sono stati realizzati e implementati utilizzando ROS è facile trovare i pacchetti che ne permettono la movimentazione;

⁹Robot con sembianze umane



(a) *Atlas*



(b) *Baxter*



(c) *Nao*



(d) *Pop*

Figura 3.3: Possibili umanoidi

- contro: il modello non rappresenta propriamente l’umano ma ne ha solo il comportamento.

Ricapitolando, la compatibilità con le specifiche risulta:

1. ° : Totalmente verificata, perché questi robot sono realizzati utilizzando l’ambiente di ROS, in modo da rimanere semplici nell’utilizzo;
 2. ° : non del tutto verificata, perché tali Robot hanno comportamento umano e non parti del corpo di tipo anatomico;
 3. ° : Totalmente verificata, perché questi Robot vengono realizzati in modo da permettere la completa movimentazione nello spazio;
 4. ° : Totalmente verificata, perché si sarebbe cercato un pacchetto compatibile con i software.
- **Strada 4:** realizzare o utilizzare un modello Solidworks di un umano e con l’utilizzo del pacchetto Sw2urdf estrarne il file URDF¹⁰.
- Pro: la possibilità di poter decidere il livello di accuratezza dei dettagli, in quanto in fase di progettazione si può realizzare o scegliere a piacimento il modello;
 - Contro: l’enorme quantità di tempo richiesta che può essere usata in fase di progettazione del modello tramite SolidWorks e durante la conversione nel file URDF, dipendenti dalla complessità del modello.

La compatibilità con le specifiche risulta essere la seguente:

1. ° : Totalmente verificata, perché è possibile decidere in partenza la complessità del modello;

¹⁰L’Unified Robotic Description Format (URDF) è un formato di file XML utilizzato in ROS per descrivere tutti gli elementi di un robot.

2. ° : Totalmente verificata, perché è possibile trovare modelli umani già realizzati o utilizzare tutti gli strumenti di SolidWorks per creare le forme geometriche più disparate;
3. ° : Totalmente verificata, perché è possibile realizzare diversi tipi di giunti in base alla proprie preferenze come combinazioni di quelli messi a disposizione del pacchetto Sw2urdf(approfonditi nel prossimo capitolo);
4. ° : Totalmente verificata, perché il file URDF è totalmente compatibile per tali software.

La strada che è stata scelta per la realizzazione del progetto è la quarta perché è quella che rispetta le 4 specifiche richieste. Inoltre, quest'ultima può essere applicata anche per altri progetti.

Dopo avere escluso le altre strade, perché complicate o perché non permettevano di ottenere le specifiche volute, si è scelto di utilizzare un modello presente sul sito *GrabCad*^{11 12}, in modo da percorrere la strada più efficiente e rispettosa delle specifiche dell'umano, risparmiando tempo ed energia che si sarebbero dovuti usare nella realizzazione del modello tramite Solidworks . Nella fig. 3.4 viene riportato il modello scelto.

Bisogna sottolineare che in futuro si potranno trovare modelli sempre più realistici oppure progettare un'altro più accurato.

¹¹ *GrabCad*. URL: <https://grabcad.com/>.

¹² GrabCad è la più grande comunità di ingegneri, designer e studenti al mondo. In particolare, in essa si possono trovare un'enorme quantità di modelli cad caricati dai propri utenti, scaricabili gratuitamente dopo l'iscrizione

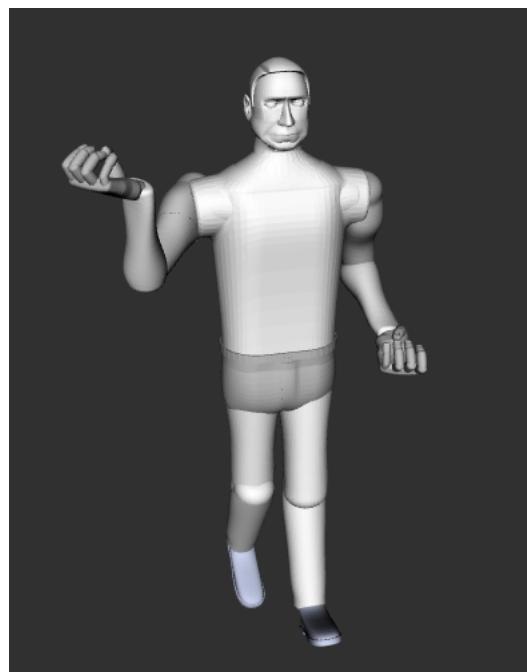


Figura 3.4: Modello umano scelto

3.3 Articolazioni

Per continuare la trattazione è necessario introdurre cosa sono le articolazioni, in quanto sono l'elemento centrale per il movimento nello spazio dell'uomo, perché connettono le varie parti del corpo ed in base alla loro tipologia ne determinano il movimento. Una definizione può essere la seguente :

Le articolazioni sono le strutture che, nel corpo umano, mantengono in contiguità due o più superfici ossee. La connessione tra le ossa è assicurata da vari elementi: tessuto fibroso, tessuto cartilagineo, capsule, legamenti e membrane.

Le articolazioni possono essere divise in più tipi, differenziate dal movimento che consentono alle ossa da esse collegate. Si individuano di seguito le principali:

- **mobili**, come ginocchio, gomito, polso, spalla, anca e caviglia;
- **semimobili**, come la colonna vertebrale;
- **fisse**, come le ossa del bacino e del cranio.

Tale diversificazione delle articolazioni, introduce un problema nella realizzazione di quest' ultime nell'ambiente ROS, in quanto in base a quale articolazione si sta definendo bisognerà fare in modo che il movimento che ne deriva rispetti la realtà, per fare ciò bisognerà considerare un'articolazione come se fosse un giunto che connette due parti del corpo in modo univoco.

Per definire come i giunti del modello devono comportarsi nella simulazione, affinché ci sia una stretta correlazione con la realtà, bisogna considerarne i movimenti ammessi, individuandone i limiti.

3.3.1 Limiti articolazioni

I limiti delle articolazioni sono molto importanti da definire, perché permettono di realizzare un modello il più reale possibile.

Per definirli si è fatto riferimento alla seguente tabella (fig. 3.6), presa dal sito del Washington State Department of social and Health services¹³ (*Range of Joint Motion Evaluation Chart*¹⁴). Dalla seguente è stato possibile definire con maggior accuratezza il comportamento dei giunti, ed è stato possibile definire, in base alle rotazioni attorno agli assi, quale tipologia da realizzare. I principali sono i seguenti:

- **Sferici:** che permettono la rotazione attorno alle 3 assi (x,y,z, è stato scelto per una maggiore compressione il sistema di riferimento illustrato in fig. 3.5). Le articolazioni che possono essere rappresentate da questo tipo di giunto sono:
 - Collo: caratterizzato da 3 movimenti attorno agli assi, partendo dalla posizione della testa perfettamente dritta rispetto al busto:

* x: spostamento massimo angolare in estensione di 60° e in flessione di 50°;

¹³Offre servizi per affrontare la salute, la sicurezza, la protezione e la qualità della vita per i residenti dello Stato di Washington, in particolare quelli che si occupano di problemi di salute mentale

¹⁴*Range of Joint Motion Evaluation Chart*. URL: <https://www.dshs.wa.gov/sites/default/files/FSA/forms/pdf/13-585a.pdf>.

- * y: rotazione a destra e sinistra di 80° ;
- * z :rotazione a destra e sinistra di 45° ;
- Busto: anch’esso ha 3 movimenti. Posizionandolo dritto rispetto al resto del corpo, ottengo le rotazioni ammissibili attorno agli assi:
 - * x: 25° in estensione, 90° in estensione;
 - * z: 25° sia verso destra che verso sinistra;
 - * y: non essendo indicato il limite per tale articolazione, si è posto un valore a livello pratico di rotazione di 45° sia verso destra che verso sinistra.
- Anca: i limiti di rotazione che si sono trovati per i tre assi, posizionando la gamba dritta, sono per:
 - * x: 30° in estensione, 100° in flessione;
 - * y: non avendo un’informazione precisa si è messo per praticità di 45° sia verso destra che verso sinistra;
 - * z: 40° verso l’esterno, 20° verso l’interno.
- Spalla: i limiti definiti per tale giunto, partendo dal braccio dritto, sono stati:
 - * x: 50° in estensione, 150° in flessione ;
 - * y: non avendo un limite definito da tale tabella si è posto per praticità 90° sia verso destra che verso sinistra;
 - * z: 50° in estensione, 150° in flessione.
- Polso: si individuano per il polso posto dritto i seguenti limiti per gli assi:
 - * x: 60° in flessione e 60° in estensione ;
 - * y: 20° verso l’interno e 30° verso l’esterno ;
 - * z: posizionandolo ortogonalmente alla posizione utilizzata per i limiti precedenti, si ottiene una rotazione di 80° sia verso destra che verso sinistra.
- **Cardanici:** che permettono la rotazione solo attorno ad un asse. Le articolazioni che possono essere rappresentate da questo tipo di giunto sono:

- Gomito: posizionando il braccio completamente esteso in modo ortogonale al busto, si ottiene una rotazione attorno all'asse x di 150° in flessione;
 - Ginocchio: ponendo la gamba dritta, il limite individuato è di 150° in flessione.
- **Con solo due assi di rotazione.** Le articolazioni che possono essere rappresentate da questo tipo di giunto sono:
 - Caviglia: dalla posizione di partenza posta ortogonale alla gamba posta parallela al busto, si ottengono per gli assi:
 - * x: 20° verso l'alto, 40° verso il basso;
 - * y: 30° in inversione e 20° in eversione.

Questi limiti, definiti utilizzando i gradi d'arco($^\circ$), dovranno essere trasformati in radianti per poterli inserire nella sezione *limit* dell'exporter di SolidWorks.

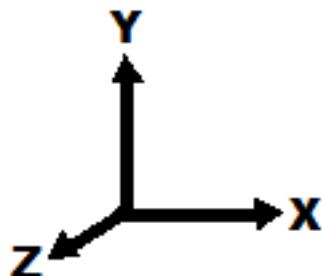
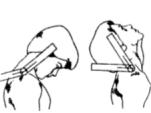
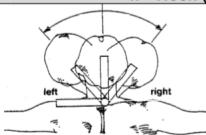


Figura 3.5: Sistema di riferimento usato

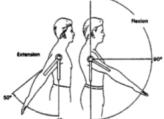
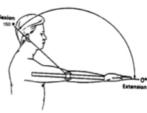
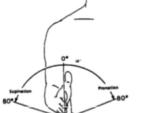
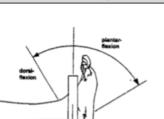
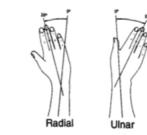
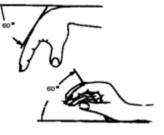
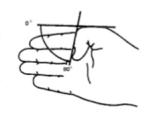


Range of Joint Motion Evaluation Chart

NAME OF PATIENT	CLIENT IDENTIFICATION NUMBER																	
INSTRUCTIONS: For each affected joint, please indicate the existing limitation of motion by drawing a line(s) on the figures below, showing the maximum possible range of motion or by notating the chart in degrees. Provide a complete description of all affected joints in your narrative summary. If range of motion was normal for all joints, please comment in your narrative summary. If joints which do not appear on this chart are affected, please indicate the degree of limited motion in your narrative.																		
1. Back  <table border="1"> <tr> <td>Extension 25°</td> <td>Flexion 90°</td> </tr> <tr> <td>Degrees</td> <td>Degrees</td> </tr> </table>		Extension 25°	Flexion 90°	Degrees	Degrees	2. Lateral (flexion)  <table border="1"> <tr> <td>Left 25°</td> <td>Right 25°</td> </tr> <tr> <td>Degrees</td> <td>Degrees</td> </tr> </table>	Left 25°	Right 25°	Degrees	Degrees								
Extension 25°	Flexion 90°																	
Degrees	Degrees																	
Left 25°	Right 25°																	
Degrees	Degrees																	
3. Neck  <table border="1"> <tr> <td>Extension 60°</td> <td>Flexion 50°</td> </tr> <tr> <td>Degrees</td> <td>Degrees</td> </tr> </table>		Extension 60°	Flexion 50°	Degrees	Degrees	4. Neck (lateral bending)  <table border="1"> <tr> <td>Left 45°</td> <td>Right 45°</td> </tr> <tr> <td>Degrees</td> <td>Degrees</td> </tr> </table>	Left 45°	Right 45°	Degrees	Degrees								
Extension 60°	Flexion 50°																	
Degrees	Degrees																	
Left 45°	Right 45°																	
Degrees	Degrees																	
5. Neck (rotation)  <table border="1"> <tr> <td>Left 80°</td> <td>Right 80°</td> </tr> <tr> <td>Degrees</td> <td>Degrees</td> </tr> </table>		Left 80°	Right 80°	Degrees	Degrees	6. Hip (backward extension)  <table border="1"> <tr> <td>Left 30°</td> <td>Right 30°</td> </tr> <tr> <td>Degrees</td> <td>Degrees</td> </tr> </table>	Left 30°	Right 30°	Degrees	Degrees								
Left 80°	Right 80°																	
Degrees	Degrees																	
Left 30°	Right 30°																	
Degrees	Degrees																	
7. Hip (flexion)  <table border="1"> <tr> <td colspan="2">Left</td> </tr> <tr> <td>Knee Flexed 100°</td> <td>Knee Extended 100°</td> </tr> <tr> <td>Degrees</td> <td>Degrees</td> </tr> <tr> <td colspan="2">Right</td> </tr> <tr> <td>Knee Flexed 100°</td> <td>Knee Extended 100°</td> </tr> <tr> <td>Degrees</td> <td>Degrees</td> </tr> </table>		Left		Knee Flexed 100°	Knee Extended 100°	Degrees	Degrees	Right		Knee Flexed 100°	Knee Extended 100°	Degrees	Degrees	8. Hip (adduction)  <table border="1"> <tr> <td>Left 20°</td> <td>Right 20°</td> </tr> <tr> <td>Degrees</td> <td>Degrees</td> </tr> </table>	Left 20°	Right 20°	Degrees	Degrees
Left																		
Knee Flexed 100°	Knee Extended 100°																	
Degrees	Degrees																	
Right																		
Knee Flexed 100°	Knee Extended 100°																	
Degrees	Degrees																	
Left 20°	Right 20°																	
Degrees	Degrees																	
9. Hip (abduction)  <table border="1"> <tr> <td>Left 40°</td> <td>Right 40°</td> </tr> <tr> <td>Degrees</td> <td>Degrees</td> </tr> </table>		Left 40°	Right 40°	Degrees	Degrees	10. Knee (flexion)  <table border="1"> <tr> <td>Left 150°</td> <td>Right 150°</td> </tr> <tr> <td>Degrees</td> <td>Degrees</td> </tr> </table>	Left 150°	Right 150°	Degrees	Degrees								
Left 40°	Right 40°																	
Degrees	Degrees																	
Left 150°	Right 150°																	
Degrees	Degrees																	

DSHS 13-585A (REV. 03/2014)

Limiti articolazioni 1

11. Shoulder (Abduction – Adduction)	Left	12. Shoulder (Flexion – Extension)	Left
	Abduction 150° Adduction 30°		Extension 50° Flexion 150°
Degrees	Degrees	Degrees	Degrees
Right		Right	
Abduction 150° Adduction 30°		Extension 50° Flexion 150°	
Degrees	Degrees	Degrees	Degrees
13. Elbow	Left	14. Forearm (Pronation – Supination)	Left
	Extension 0° Flexion 150°		Pronation 80° Supination 80°
Degrees	Degrees	Degrees	Degrees
Right		Right	
Extension 0° Flexion 150°		Pronation 80° Supination 80°	
Degrees	Degrees	Degrees	Degrees
15. Ankle	Left	16. Ankle (Flexion – Extension)	Left
	Inversion 30° Eversion 20°		Plantar 40° Dorsal 20°
Degrees	Degrees	Degrees	Degrees
Right		Right	
Inversion 30° Eversion 20°		Plantar 40° Dorsal 20°	
Degrees	Degrees	Degrees	Degrees
17. Wrist (radial, ulnar)	Left	18. Wrist	Left
	Radial 20° Ulnar 30°		Extension 60° Flexion 60°
Degrees	Degrees	Degrees	Degrees
Right		Right	
Radial 20° Ulnar 30°		Extension 60° Flexion 60°	
Degrees	Degrees	Degrees	Degrees
19. Thumb (MP Joint)	Left Right	20. Thumb (IP Joint)	Left Right
	Flexion 60° Flexion 60°		Flexion 80° Flexion 80°
Degrees	Degrees	Degrees	Degrees
DATE OF EXAMINATION	EXAMINING PHYSICIAN'S SIGNATURE	DATE OF REPORT	

DSHS 13-585A (REV. 03/2014)

Limiti articolazioni 2

Figura 3.6: Limiti articolazioni

Creazione modello

In questo capitolo si tratteranno i punti principali per la realizzazione del progetto seguendo la strada scelta per portare a termine l'obbiettivo. Si parlerà di SolidWorks, del pacchetto Sw2urdf, del file URDF e della visualizzazione del modello in Rviz.

4.1 SolidWorks

SolidWorks è il software che ha permesso di seguire la strada n. 4 , descritta nel capitolo precedente. Innanzitutto è necessario definire cos'è SolidWorks:

SolidWorks è un software di disegno e progettazione tridimensionale parametrica, prodotto e commercializzato dalla Dassault Systèmes. Nasce come software appositamente dedicato per l'ingegneria meccanica ed è quindi particolarmente utile per la progettazione di apparati meccanici, anche complessi.

SolidWorks, permette dunque di poter generare a livello virtuale la maggior parte degli elementi meccanici, sia singoli, come parte, che uniti come assieme. Nel caso proposto da questa tesi, SolidWorks insieme al pacchetto Sw2urdf ha svolto il ruolo di convertitore da un modello precedentemente realizzato da utenti della community di GradCad, al file URDF utilizzato attualmente in ROS per la realizzazione e simulazione di robot dai più semplici a quelli più complessi.

4.2 Da SolidWorks a ROS

Per poter comprendere come sia stato possibile attuare tale conversione è necessario in primo luogo definire i concetti basilari del file URDF.

4.2.1 Concetti sul file URDF e SDF

Il formato che viene utilizzato in ROS per descrivere le caratteristiche del robot è il formato URDF (Unified Robot Description Model). Esso deriva dal formato XML¹ (Extensible Markup Language) . Il formato URDF prevede una descrizione del robot utilizzando una struttura ad albero i cui nodi principali sono i link, che rappresentano le singole parti che compongono il robot, e i joint che indicano i collegamenti che ci sono tra due link. Le principali caratteristiche che vengono definite utilizzando il file URDF sono :

- la dinamica e la cinematica;
- l'aspetto grafico di come si presenta il robot;
- il modello relativo alle collisioni.

Link

I link, come già sottolineato precedentemente, rappresentano le singole parti rigide del robot. Essi hanno delle caratteristiche che li definiscono come tali e sono individuate dai seguenti tag:

- <link name=myname>, indica la nomenclatura del link, è messo ad inizio blocco per poter indicare che si sta inserendo un link e non un joint, oltre che assegnargli un nome per riconoscerlo e distinguerlo dagli altri;

¹linguaggio per lo scambio di dati strutturati. Non trattandosi di un formato di file rigido, XML è un linguaggio adatto per la definizione di formati concordati utilizzabili da gruppi di lavoro per lo scambio di dati.

- <visual>, che definisce l'aspetto visivo del modello;
- <inertial>, che individua le caratteristiche fisiche, come l'inerzia;
- <collision>, che indica il modello per la gestione delle collisioni.

Un link può essere rappresentato visivamente come viene raffigurato nella fig. 4.1.

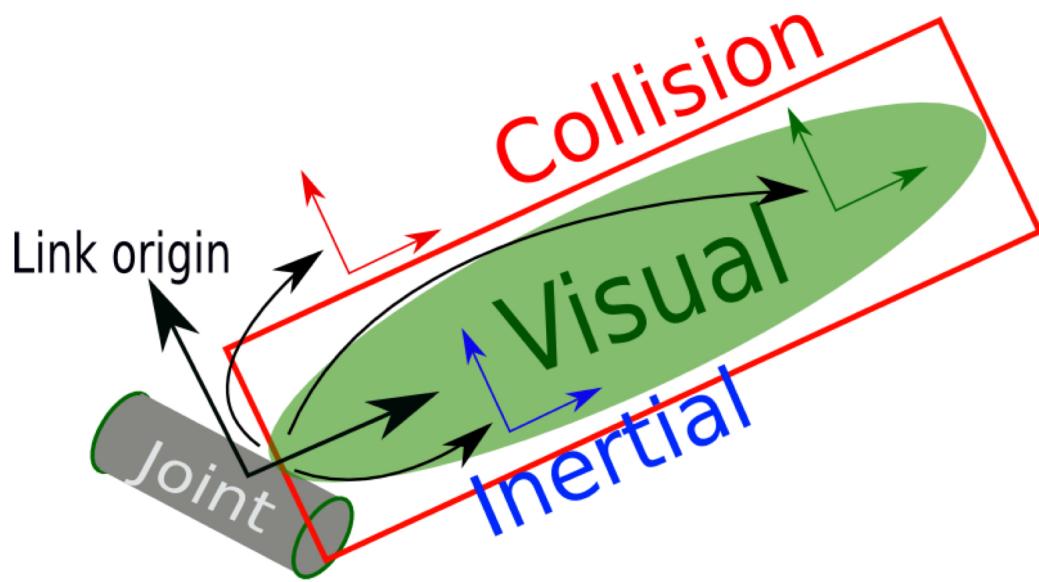


Figura 4.1: Modello di un link

Il codice per poter definire un link è il seguente(fig. 4.2):

```
1 <link name="my_link">
2   <inertial>
3     <origin xyz="0 0 0.5" rpy="0 0 0"/>
4     <mass value="1"/>
5     <inertia ixx="100" ixy="0" ixz="0" iyy="100" iyz="0" izz="100" />
6   </inertial>
7
8   <visual>
9     <origin xyz="0 0 0" rpy="0 0 0" />
10    <geometry>
11      <box size="1 1 1" />
12    </geometry>
13    <material name="Cyan">
14      <color rgba="0 1.0 1.0 1.0"/>
15    </material>
16  </visual>
17
18  <collision>
19    <origin xyz="0 0 0" rpy="0 0 0"/>
20    <geometry>
21      <cylinder radius="1" length="0.5"/>
22    </geometry>
23  </collision>
24 </link>
```

Figura 4.2: Codice per definire un link

Dove:

- <origin>: indica la posizione del sistema di riferimento utilizzato per posizionare gli elementi rispetto al link stesso;
- <geometry>: indica la geometria da considerare per il link in base alla voce del tag.

joint

Nel formato URDF i joint servono per definire le relazioni che i link hanno tra loro.

Possono essere rappresentati come nella fig. 4.3

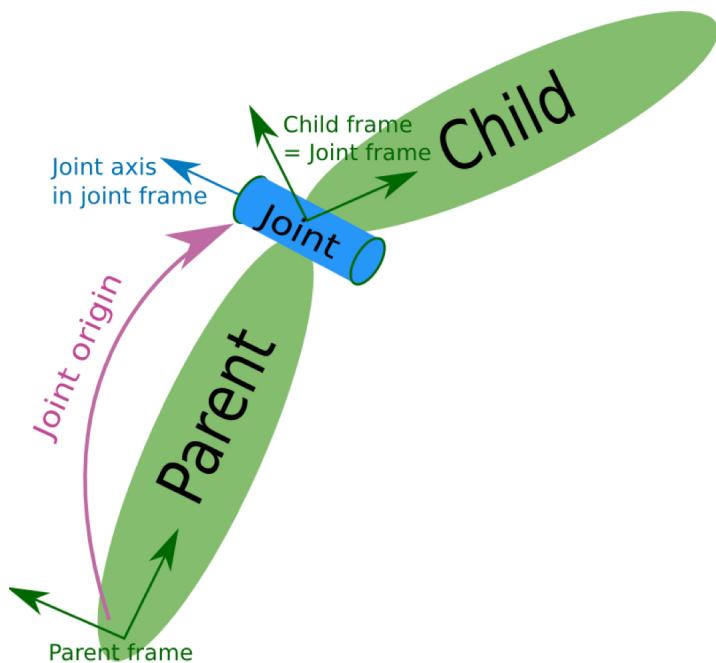


Figura 4.3: Esempio grafico joint

A livello di codice, invece, come nella fig. 4.4.

```
<joint
  name="spalla_sx_rz"
  type="revolute">
  <origin
    xyz="0 0 0"
    rpy="0 0 0" />
  <parent
    link="bicipite_sx_ry_Link" />
  <child
    link="bicipite_sx_rz_Link" />
  <axis
    xyz="0 0 1" />
  <limit
    lower="-0.87"
    upper="2.62"
    effort="0"
    velocity="0" />
</joint>
```

Figura 4.4: Codice d'esempio per un joint

Dove:

- <origin>: serve per indicare la posizione del link figlio(child) rispetto a quello padre (parent), il giunto è posizionato nell'origine del link figlio;
- <parent>: indica quale link è il padre;
- <child>: indica quale link è il figlio;
- <axis>: asse che viene utilizzata dal giunto, per determinare il movimento (per giunti revolute, prismatic e planar);
- <limit>: permette di definire i limiti che il giunto può ammettere in termini di:
 - movimento: con lower ed upper, per definire l'ampiezza del movimento ammessa;
 - forza massima ammissibile tramite effort;
 - velocità massima ammissibile tramite velocity.

Bisogna sottolineare che tale voce viene utilizzata per un giunto di tipo revolute o prismatic.

- <type>: definisce il tipo di giunto che si sta considerando. I principali tipi sono:
 - **revolute**: giunto che permette la rotazione attorno ad un asse, con un'ampiezza variabile dipendente dai limiti lower ed upper;
 - **continuous**: giunto simile al revolute, ma che non ha limiti di movimento;
 - **prismatic**: giunto che permette il moto traslatorio lungo l'asse indicato dalla voce axis;
 - **fixed**: giunto che non permette nessun movimento, dunque blocca ogni grado di libertà;
 - **floating**: giunto che ammette qualsiasi movimento, ha tutti i gradi di libertà ;
 - **planar**: giunto che permette il movimento in una piano ortogonale all'asse, indicato nella voce axis .

4.2.2 SolidWorks to URDF Exporter

Dopo aver seguito l'installazione del plug-in, Sw2urdf, all'interno di SolidWorks, indicata nel capitolo 3, è possibile estrarre dal modello il file URDF. Per farlo bisogna eseguire le istruzioni seguenti:

1. **Inserire i sistemi di riferimento:** in base a dove si vuole mettere il giunto, utilizzando la procedura che Solidworks mette a disposizione al sito: *Creating a Coordinate System*,² inserire un sistema di riferimento in quella posizione, denominandolo a piacimento (è consigliabile utilizzare un nome diverso per ogni giunto, per poterli distinguere oltre che a riconoscerli);
2. **Inserire eventuali assi:** in base al tipo di giunto che si vuole utilizzare bisognerà inserire un asse per definirne il movimento. Per l'inserimento di un asse è possibile seguire la procedura indicata da SolidWorks al sito : *Creating Reference Axes*;³
3. **Verifica sistemi e assi per i giunti:** controllare che gli assi e i sistemi di riferimento inseriti siano concordi tra loro, ovvero ci sia una correlazione logica con quello che si vuole realizzare successivamente;
4. **Apertura dell'export:** dalla finestra principale di Solidworks, si individua la finestra file, in cui si cerca la voce Export to URDF. Una volta cliccato su tale voce, se non si è salvato il documento dopo aver apportato le modifiche, il programma chiederà di salvarlo e di ricostruirlo. Avvenuti, il salvataggio e la ricostruzione, si aprirà la finestra dell'export;
5. **Inserire i dati richiesti:** aperta la finestra dell'export è possibile individuare diverse voci che dovranno essere riempite, tra cui:
 - link name: spazio per inserire il nome del link;

²*Creating a Coordinate System.* URL: http://help.solidworks.com/2020/Italian/SolidWorks/sldworks/t_Creating_a_Coordinate_System.htm?verRedirect=1.

³*Creating Reference Axes.* URL: http://help.solidworks.com/2020/Italian/SolidWorks/sldworks/t_Creating_Reference_Axes.htm?id=8ebe8673368743989e48672c6c308636#Pg0.

- joint name: spazio per inserire il nome del joint;
- Reference Coordinate System: elenco a tendina per selezionare il sistema che fa riferimento al giunto, se non è stato creato precedentemente è possibile lasciar fare al programma mettendo la selezione "Automatically Generated". In applicazioni complesse questo è sconsigliato perché potrebbe essere collocato dove non si è preventivata la posizione del joint, portando ad errori durante la simulazione;
- Reference Axis: elenco a tendina per selezionare l'asse alla quale il giunto deve fare riferimento. Anche in questo caso è possibile lasciar scegliere al plug-in;
- Joint type: elenco a tendina per selezionare di che tipo di giunto si tratta, in base a quelli visti precedentemente;
- Link Components: spazio in cui è possibile, cliccando sul modello, selezionare le parti che definiscono il link, per assegnarli una geometria. In questo caso la geometria sarà individuata da una mesh⁴;
- Number of child Link: spazio, in cui è possibile definire il numero di link figli, derivanti da quello selezionato;
- spazio di visualizzazione relazioni tra i link: diagramma ad albero in cui è possibile vedere le relazioni padre e figlio che i link inseriti hanno .

Tale procedura va effettuata per ogni link presente nel modello da voler realizzare, a differenza del primo che risulta essere il link base, dal quale il modello ne deriva. Durante la fase di progettazione è consigliato decidere correttamente quale link del nostro modello dovrà essere la base, perché sarà il punto di partenza della creazione dell'URDF.

6. **Preview and Export:** una volta inserite tutte le informazioni, indicate precedentemente, per ogni singolo link ed averne verificato la correttezza, è possibile

⁴tipo di file che permette di definire la forma di un oggetto diversa da quelle geometriche

cliccare su Preview and Export. Svolta questa procedura il programma aprirà una finestra in cui sarà possibile definire l'inserimento di ulteriori informazioni, in particolare:

- Configure Joint Properties, in cui si potranno verificare la correttezza dei dati dei joint inseriti precedentemente ed aggiungerne dei nuovi, tra cui:
 - limit;
 - calibration;
 - dynamics;
 - safety control.
- configure link Properties (si apre dopo aver completato la prima finestra e aver schiacciato il tasto next). In tale finestra è possibile verificare la correttezza dei dati inseriti per i link ed aggiungere altre informazioni relative a :
 - inertial;
 - Visual and Collision meshes.

7. Export URDF and Meshes...: una volta completato l'inserimento di tutti i dati in tutte le finestre e verificato la correttezza di quest'ultimi, è possibile schiacciare il tasto Export URDF and Meshes.... A questo punto il plug-in creerà, in una destinazione scelta, una cartella in cui si troveranno i file :

- cartella config, in cui si trova il file che individua le caratteristiche dei giunti;
- cartella launch, in cui si hanno i file per poterlo aprire in Rviz(display.launch) e Gazebo (gazebo.launch);
- cartella textures;
- cartella urdf, in cui si ha il file URDF(nomerobot.urdf) e il file csv(nomerobot.csv) che contiene informazioni realative ai link;
- cartella meshes , in cui si trovano tutte le mesh che rappresentano le geometrie dei vari link utilizzati;

- file di testo CMakeLists;
 - file di testo package.
8. **Trasferire la cartella nell'ambiente ROS:** si possono seguire diverse strade per il trasferimento della cartella, che dipendono dove si ha installato ROS, quella consigliata, nel caso si avesse una partizione, è quella di caricare la cartella in un servizio di cloud che si vuole utilizzare (per esempio Google Drive), per poi riscaricarla nel SO in cui si usa ROS all'interno del catkin_ws nella cartella src, ed una volta caricata applicare il comando catkin_make, dalla cartella catkin_ws, per poter utilizzare i programmi e gli strumenti che ROS mette a disposizione per lo sviluppo di un robot.

Creazione modello

Rappresentazione grafica della procedura effettuata dal punto 4 al punto 7:

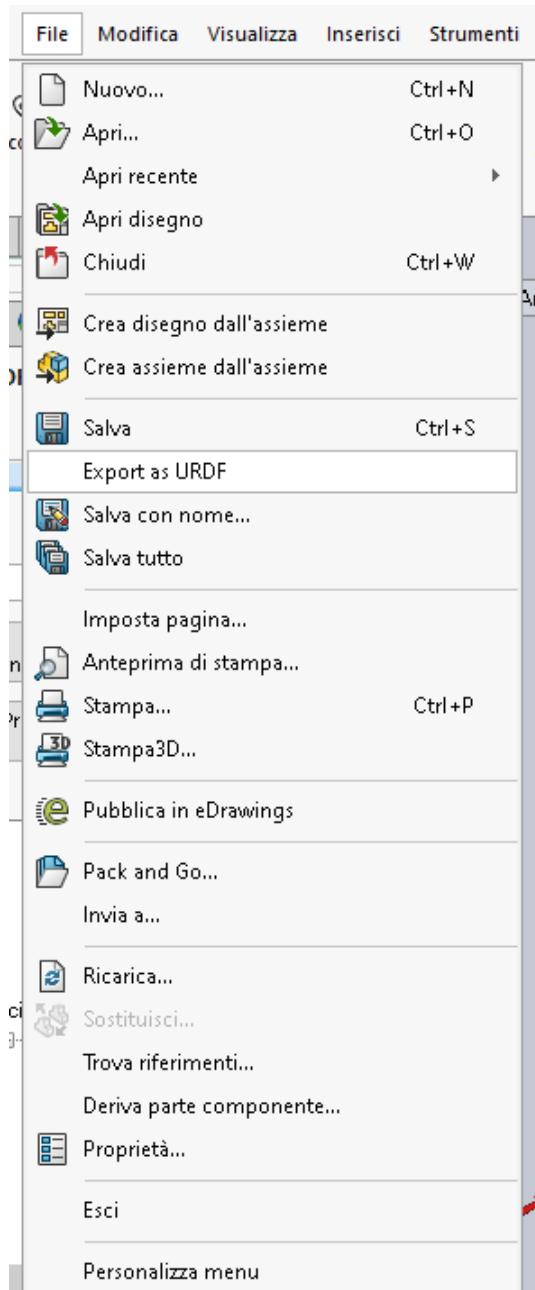


Figura 4.5: apertura dell'export(4)

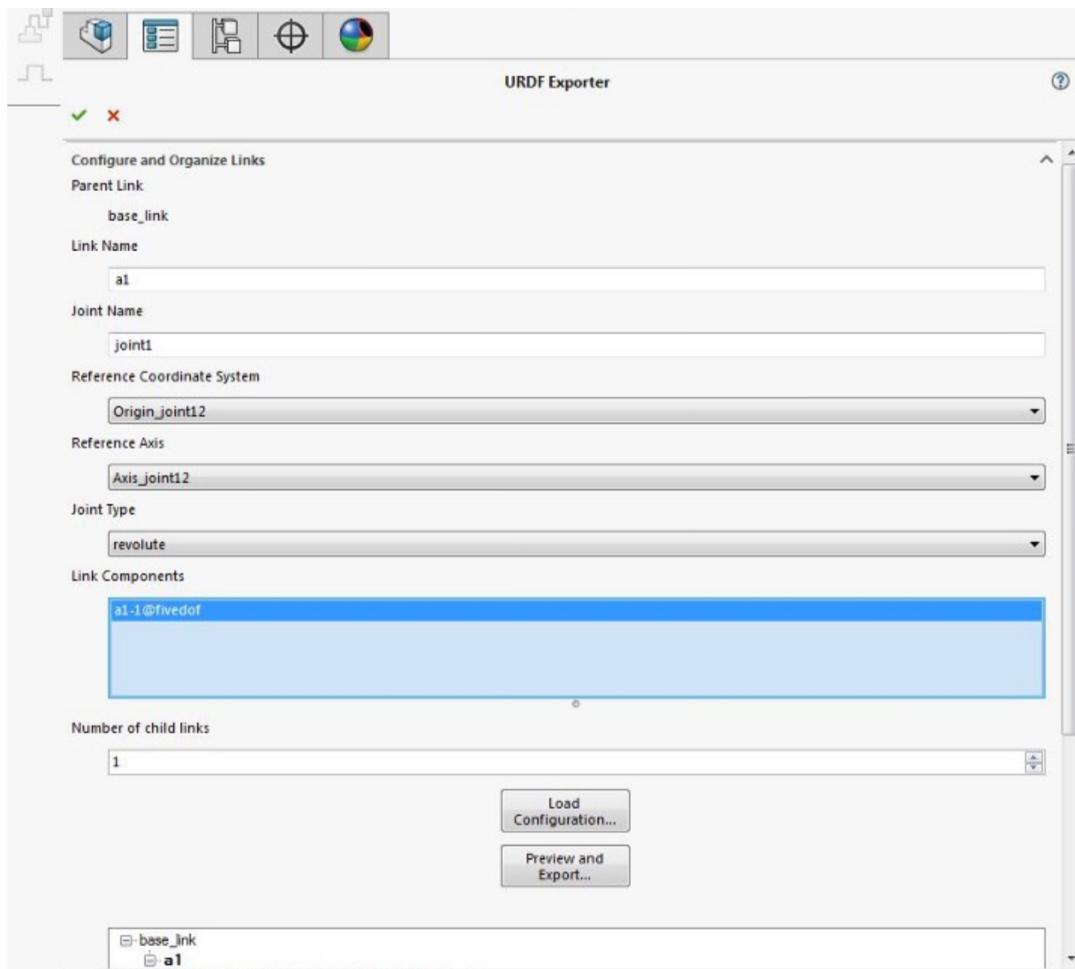


Figura 4.6: Property manager(5)

Creazione modello

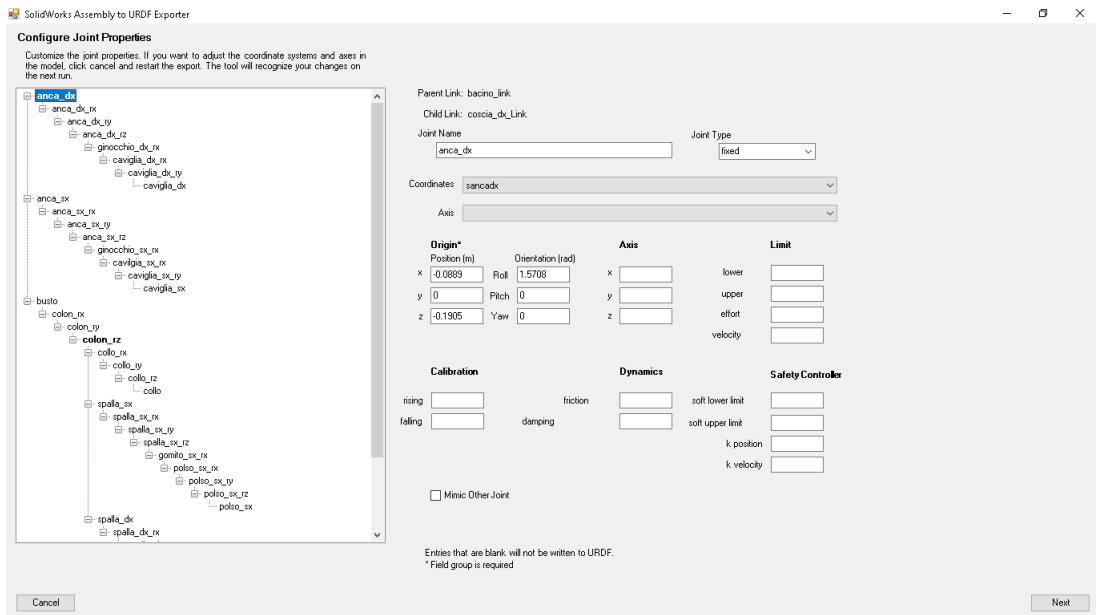


Figura 4.7: configure joint properties(6)

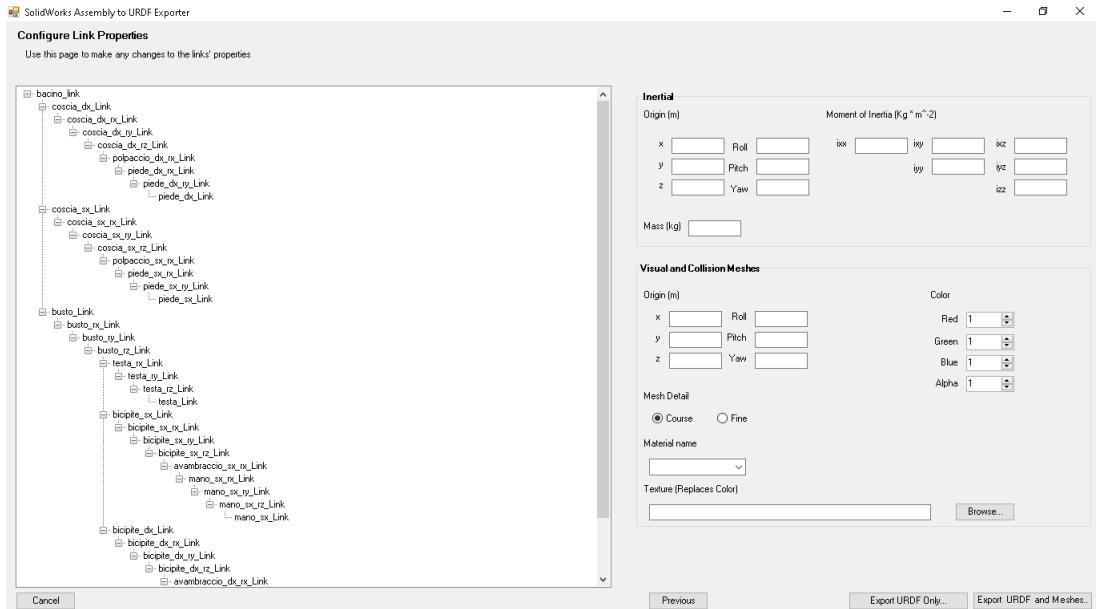


Figura 4.8: configure link properties(6)

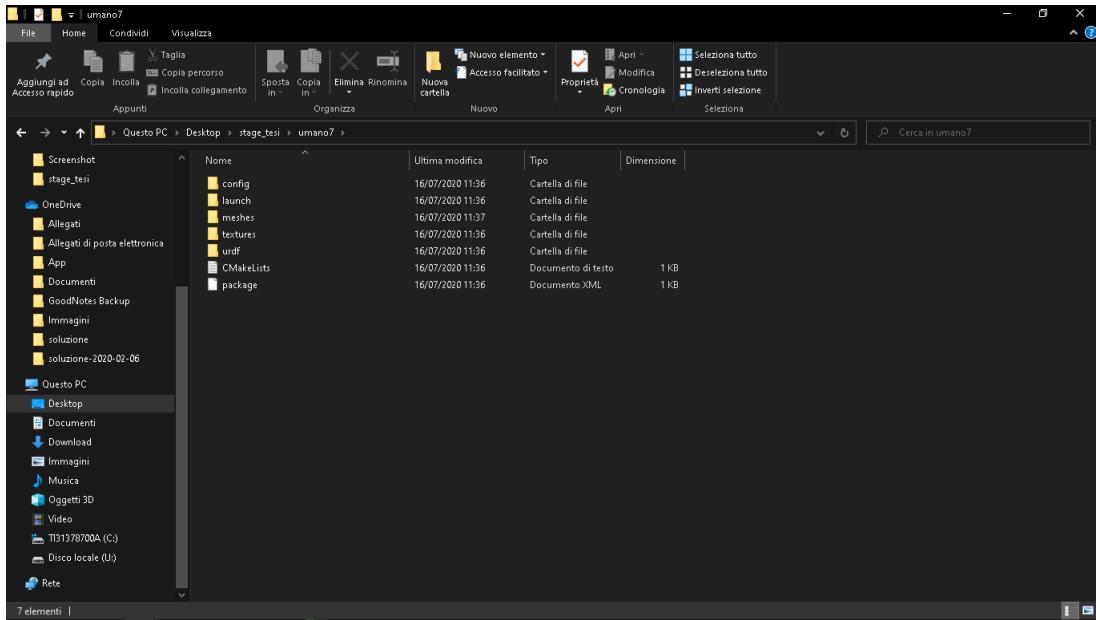


Figura 4.9: cartella URDF(7)

4.2.3 Dati inseriti per il file URDF dell’umano

I dati inseriti, seguendo quello scritto nel capitolo 4, sono stati:

- **i sistemi di riferimento:** inseriti in modo concorde tra loro;



Figura 4.10: Umano con sistemi di riferimento concordi

- **gli assi di riferimento:** inseriti paralleli agli assi del sistema di riferimento creato per il giunto.

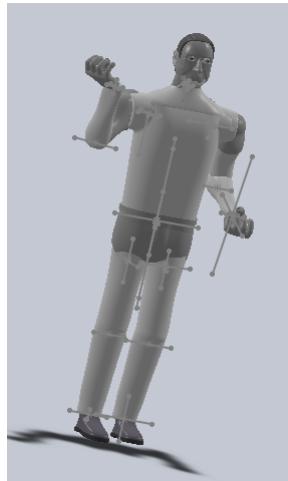


Figura 4.11: Umano con assi di riferimento inseriti

- **I limiti:** inseriti trasformando il valore in gradi trovato dalla tabella fig. 3.6 in radianti, sistemando dove necessario i valori di upper e lower, in modo da ottenere la stessa ampiezza di movimentazione. I limiti definiti da effort e velocity, non sono stati modificati in questo progetto, ma potranno essere cambiati in futuro, una volta condotti studi del movimento e del corpo umano più approfonditi, in modo da reperire tali informazioni.

Per la realizzazione delle articolazioni riguardanti: spalla, polso, bacino e collo è stato necessario creare il giunto sferico, poiché tali articolazioni permettono 3 rotazioni nello spazio, una per ogni asse x, y ,z.

4.2.4 Il giunto sferico

Il giunto sferico, a differenza degli altri giunti, non è un giunto base e dunque non è inserito tra i tipi di giunti disponibili presenti nell' Exporter, ma risulta essere una combinazione di questi ultimi. La strada per realizzarlo è stata trovata dopo diverse prove. Una volta definiti quali fossero i giunti sferici, i passi principali per realizzarlo, partendo da SolidWorks, sono i seguenti:

1. inserire un sistema di riferimento nella posizione in cui vogliamo il giunto;

2. **Realizzare un asse per x, y, z:** in modo tale che gli assi coincidano con quelli del sistema di riferimento inserito. Un’ esempio di assi per il giunto sferico sono rappresentate nella fig. 4.13;
3. **Aprire l’exporter;**
4. **Creare un giunto revolute per ogni asse basato sullo stesso link:** tale giunto risulta essere diverso dagli altri perché dovrà essere fatto tra lo stesso link, ma secondo un’ asse di rotazione diverso;
5. **Esportare l’URDF:** tramite la procedura descritta precedentemente;
6. **Aprire l’URDF creato;**
7. **Apportare la seguente modifica:** in base ai link creati, nel caso di un giunto sferico, se ne creeranno 3 riferiti allo stesso, si vanno a cancellare dal file URDF, la parte contenuta all’interno di 2 di questi, lasciandone uno pieno. Tale modifica permette il funzionamento del giunto durante la simulazione. Nella fig. 4.12 è possibile vedere il confronto tra la parte destra (senza modifica) del corpo e quella di sinistra (con la modifica).

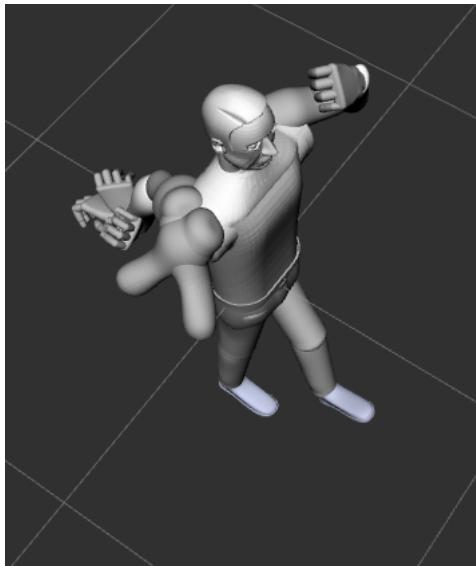


Figura 4.12: Confronto tra le due parti del corpo, con e senza modifica

Questa procedura va fatta per ogni giunto sferico. Bisogna sottolineare che tale operazione può essere fatta anche nel caso in cui si debba creare un giunto che permetta la rotazione attorno a due assi di rotazione, la cosa importante nella modifica finale è quella di lasciare un solo link così come SolidWorks lo crea e l'altro solo inizializzato (`<link name= namelink\>`). Un esempio di codice che rappresenta il collo (giunto sferico) è presente nell'appendice A(??).

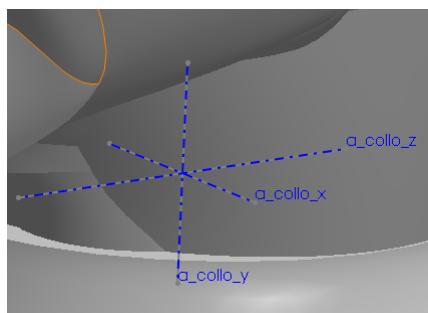


Figura 4.13: Esempio assi per giunto sferico

4.3 Procedura di apertura in Rviz

Per poter aprire il file all'interno del software di simulazioni rviz, bisognerà utilizzare il file di launch generato dall'exporter nella cartella generata al termine dell'esportazione.

4.3.1 Apertura in Rviz

La procedura per avviarlo in Rviz andando a muovere i diversi giunti in modo da poter visualizzare la correttezza dei dati inseriti per il limiti è la seguente:

1. **Installare il joint state publisher⁵**, utilizzando il seguente comando:

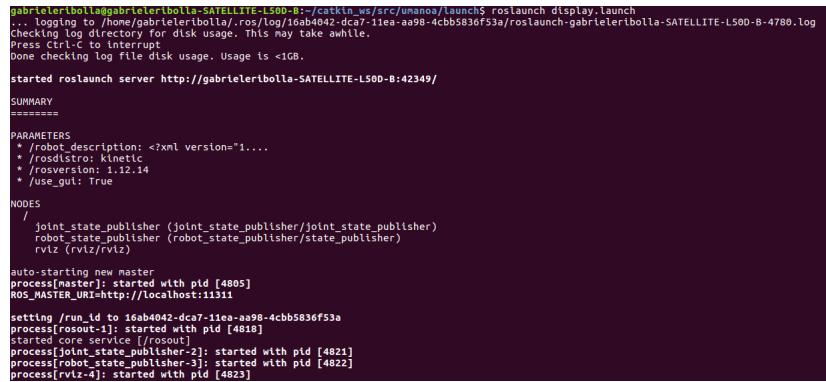
```
sudo apt-get install joint-state-publisher
```

⁵il joint state publishe è un pacchetto di ROS che, una volta scaricato, permette di mostrare i giunti del modello sviluppato, con i relativi limiti oltre che a movimentarli

2. **Apportare una modifica nel file display.launch:** una volta aperto il file con l'editor che si preferisce, alla voce <arg name ="gui"... mettere True al posto di False;
3. **Aprire un terminale e inserire la seguente sequenza di comandi:**

- *cd*
- *cd catkin_ws*
- *cd catkin_make* (se non lo si è già fatto)
- *cd src*
- *cd nomerobot*
- *cd launch*
- *roslaunch display.launch*

Lanciato l'ultimo comando, se l'operazione è andata a buon fine nel terminale otterremo ciò che viene rappresentato nella fig. 4.14



```

gabrieleribolla@gabrieleribolla-SATELLITE-L50D-B:~/catkin_ws/src/umanoa/launch$ roslaunch display.launch
... logging to /home/gabrieleribolla/.ros/log/16ab4042-dca7-11ea-aa98-4cbb5836f53a/roslaunch-gabrieleribolla-SATELLITE-L50D-B-4780.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
done checking log file disk usage. Usage is <1GB.

started roslaunch server http://gabrieleribolla-SATELLITE-L50D-B:42349/
SUMMARY
========
PARAMETERS
  * /robot_description: <?xml version="1....
  * /robot_name: kinetic
  * /rosversion: 1.12.14
  * /use_gui: True

NODES
  /
    joint_state_publisher (/joint_state_publisher/joint_state_publisher)
      robot_state_publisher (/robot_state_publisher/state_publisher)
        rviz (/rviz/rviz)

auto-starting new master
process[master]: started with pid [4805]
ROS_MASTER_URI=http://localhost:11311
setting /run_id to 16ab4042-dca7-11ea-aa98-4cbb5836f53a
process[rosout-1]: started with pid [4818]
startInfo: /rosout
process[joint_state_publisher-2]: started with pid [4821]
process[robot_state_publisher-3]: started with pid [4822]
process[rviz-4]: started with pid [4823]

```

Figura 4.14: Schermata terminale d'avvio Rviz

4. **se il punto precedente è terminato**, si visualizzerà la schermata iniziale di rviz, definita dalla fig. 4.15 ;
5. **aggiungere il modello del robot**: seguire la procedura indicata nel capitolo 3 per l'aggiunta del modello del robot. Porre come fixed frame il link base sul

Creazione modello

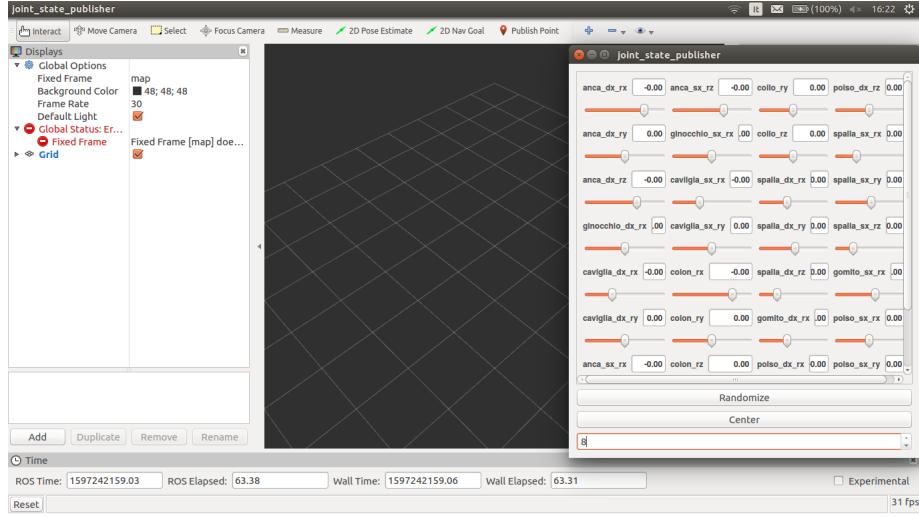


Figura 4.15: Schermata di avvio Rviz

quale si è sviluppato il modello, in questo caso è il bacino_link. Terminate tali operazioni si otterrà una schermata simile a quella rappresentata dalla fig. 4.16 (in questo caso il robot realizzato è l'umano, se si sta sviluppando un modello diverso si vedrà quest'ultimo).

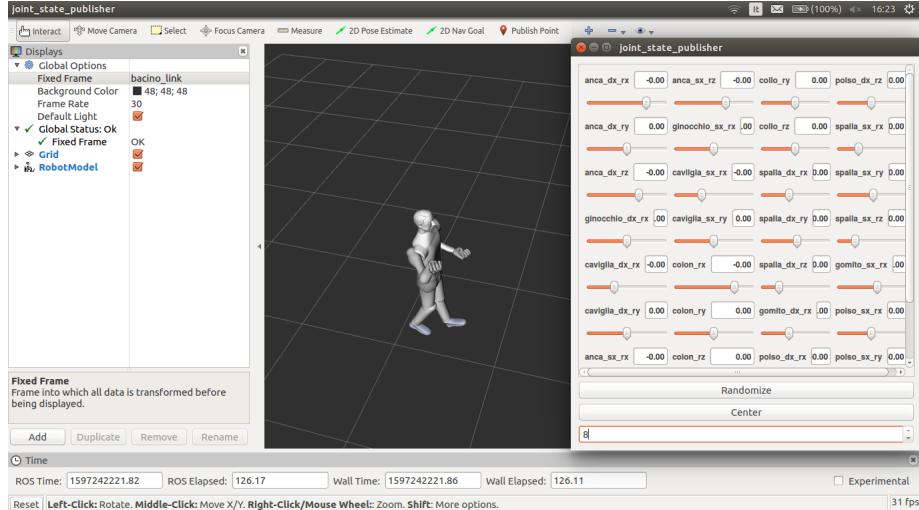


Figura 4.16: Schermata Rviz con umano

Capitolo 5

Inserimento in uno spazio aperto

5.1 I cobot

Un cobot o co-robot (derivante da " collaborative robot ") è un robot concepito per interagire fisicamente con l'uomo in uno spazio di lavoro. Ciò trasmette un contrasto della maggior parte dei robot industriali adottati fino al 2008, i quali erano progettati per operare in maniera autonoma o con una guida limitata e protetti da barriere. I cobot, negli ultimi anni, stanno invadendo il settore industriale perché permettono di ottenere un prodotto finale migliore e in modo più efficiente rispetto al non utilizzo, senza perdere il capitale umano, che risulta essere fondamentale in alcune applicazioni, in particolare per quanto riguarda il mondo dell'automotive, in cui la stretta relazione tra uomo e macchina permette di aumentare la qualità finale della autovettura investendo un minor numero di risorse sia economiche che umane. La grande crescita, che questo tipo di robot sta registrando in questi ultimi anni, sta interessando molti centri di ricerca e università, oltre che alle aziende leader del settore (per esempio FANUC), al fine di far propria l'enorme fetta di mercato che i cobot potranno acquisire nel prossimo futuro. L'applicazione principale, per cui i cobot vengono realizzati è l'inserimento in uno spazio aperto.

5.2 Lo spazio aperto

Lo spazio aperto nasce come luogo protetto in cui i cobot possono essere inseriti in sicurezza, permettendo l’interazione con un umano in modo sicuro. La simulazione del funzionamento di una collaborazione in uno spazio aperto, in cui si ha la presenza di un umano, porta al dover realizzare il modello virtuale dell’uomo, per poterlo rappresentare nella simulazione, in modo da considerare tutte le possibili situazioni che potrebbero accadere nella realtà. Detto ciò, si intuisce l’importanza del modello umano nei software di simulazione, in cui vengono visualizzati i cobot, perché quest’ultimi non ci sarebbero senza un’interazione collaborativa con esso. L’utilizzo di un modello umano diventa dunque focale per la corretta simulazione della realtà. Ne è un esempio quello realizzato dal Consiglio Nazionale delle Ricerche (C.N.R) della sede di Milano, il quale ha creato uno spazio aperto (fig. 5.1), in cui l’uomo e robot possono interagire tra loro al fine di portare a termine un’azione, in questo caso è quella di creare un mosaico (fig.5.2),

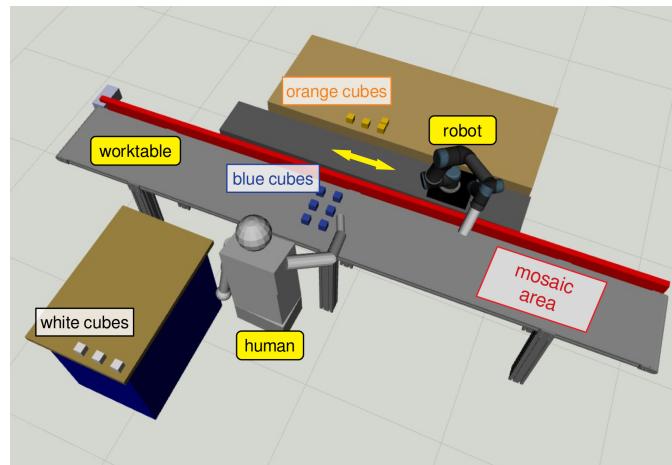


Figura 5.1: Spazio aperto realizzato da C.N.R di Milano

dove i cubetti di color blu possono essere mossi sia dall’uomo che dal cobot, quelli arancio solo dall’operatore e quelli bianchi solo dal robot. Il mosaico viene realizzato riga per riga, ovvero la seconda riga può essere realizzata solo se la prima è già stata

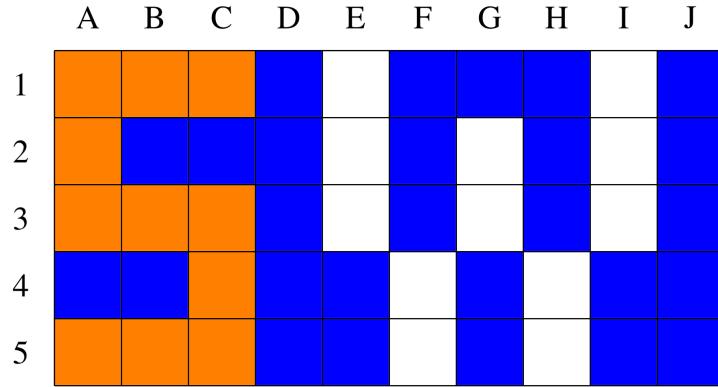


Figura 5.2: Mosaico che uomo e robot devono realizzare

completata e via dicendo. La realizzazione del mosaico viene rappresentata graficamente nella fig. 5.3, in cui è possibile vedere come si evolve nel tempo la creazione del mosaico.

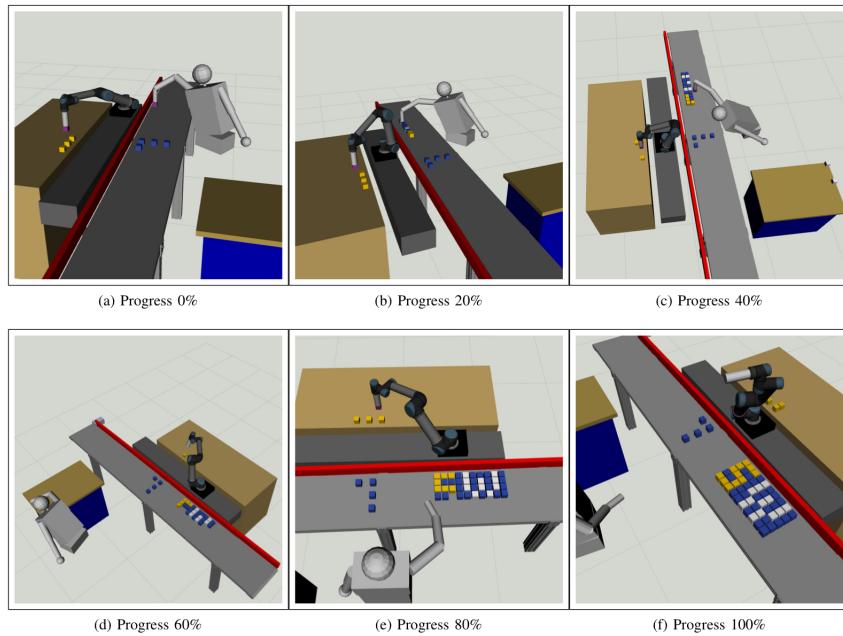


Figura 5.3: Rappresentazione grafica della simulazione dello spazio aperto

Tale progetto ha subito una variazione, dovuta all'inserimento del modello umano realizzato in questa tesi, e nella fig. 5.4 si propone la versione aggiornata. Si può

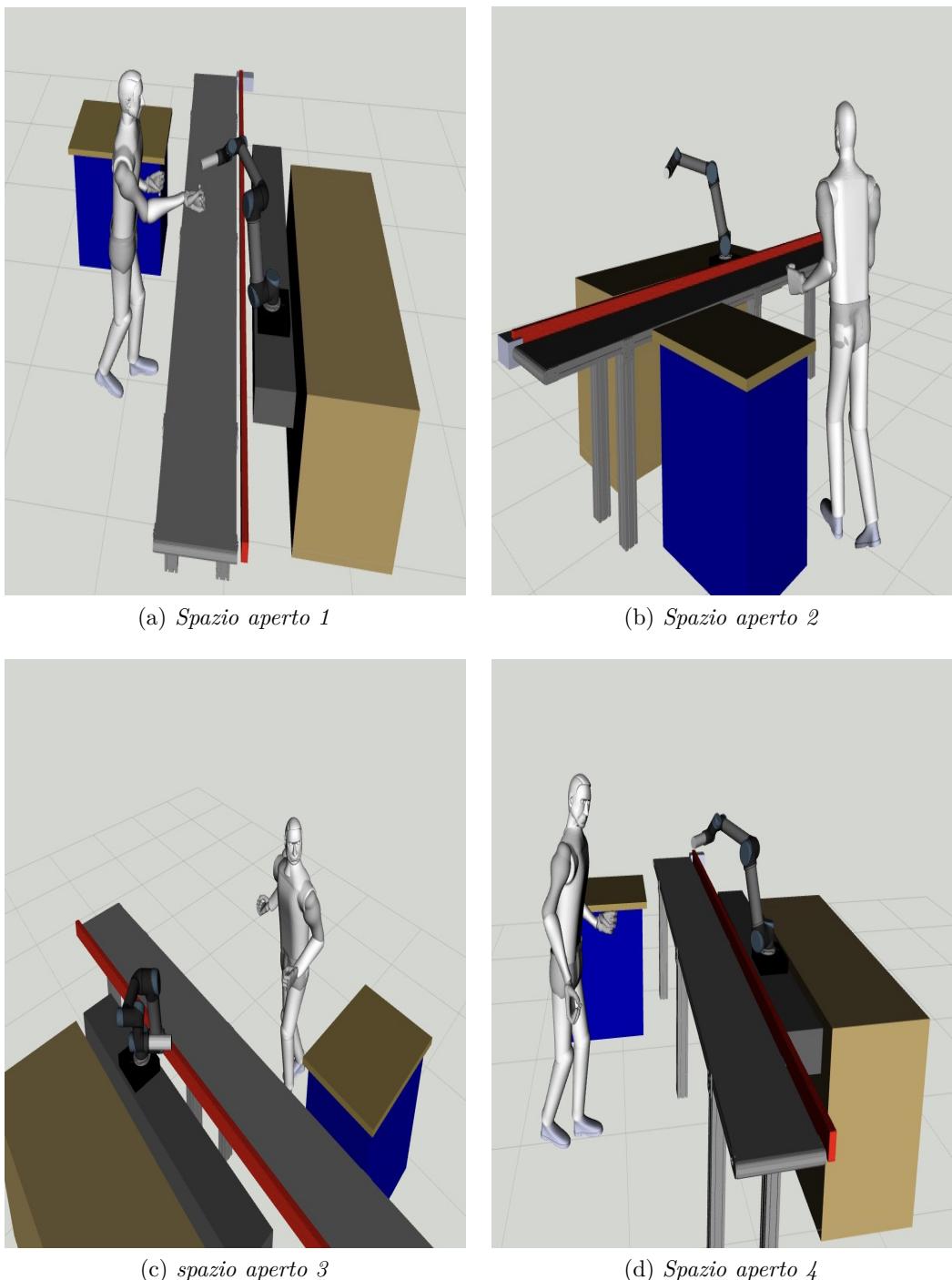


Figura 5.4: Spazio aperto aggiornato

notare come la presenza di un modello antropomorfo migliora nettamente l'aspetto grafico della simulazione facendola avvicinare alla realtà.

Bibliografia

- [JC18] Lentin Joseph e Jonathan Cacace. *Mastering ROS for Robotics Programming: Design, build, and simulate complex robots using the Robot Operating System*. Packt Publishing Ltd, 2018.
- [Far+] Marco Faroni et al. «A Layered Control Approach to Human-Aware Task and Motion Planning for Human-Robot Collaboration». In: () .
- [Lor] Pantieri Lorenzo. *LaTeXpedia*. URL: <http://it.guitex.org/>.
- [pdf-] *Range of Joint Motion Evaluation Chart*. URL: <https://www.dshs.wa.gov/sites/default/files/FSA/forms/pdf/13-585a.pdf>.
- [ros-a] *ROS packages*. URL: <https://index.ros.org/packages/>.
- [ros-b] *ROS Tutorials*. URL: <http://wiki.ros.org/ROS/Tutorials>.
- [ros-c] *ROS Forum*. URL: <https://answers.ros.org/questions/>.
- [ros-d] *ROS*. URL: <https://www.ros.org/>.
- [sit-a] *Make Human Community*. URL: <http://www.makehumancommunity.org/>.
- [sit-b] *GrabCad*. URL: <https://grabcad.com/>.
- [sol-a] *Releases Sw2urdf*. URL: https://github.com/ros/solidworks_urdf_exporter/releases.
- [sol-b] *Creating Reference Axes*. URL: http://help.solidworks.com/2020/Italian/SolidWorks/sldworks/t_Creating_Reference_Axes.htm?id=8ebe8673368743989e48672c6c308636#Pg0.

BIBLIOGRAFIA

- [sol-c] *Creating a Coordinate System.* URL: http://help.solidworks.com/2020/Italian/SolidWorks/sldworks/t_Creating_a_Coordinate_System.htm?verRedirect=1.
- [tut-a] *ROS tutorial 2: Publishers and subscribers.* URL: <https://www.youtube.com/watch?v=bJB9tv4ThV4>.
- [tut-b] *Make Human Tutorials.* URL: <https://github.com/makehumancommunity/makehuman/issues/19>.
- [ubuntu-a] *Ubuntu.* URL: <https://ubuntu.com/>.
- [ubuntu-b] *Repositories Ubuntu.* URL: <https://help.ubuntu.com/community.Repositories/Ubuntu>.