Documentazione qi unipa

Qi unipa è un framework che sfrutta a sua volta il framework **qi** della SoftBank Robotics, e permette di utilizzare le NaoQi Api 2.5 per controllare i robot Pepper e Nao.

È composto dalle seguenti componenti:

- **Movement**: gestisce il movimento del robot, in particolare il movimento dei giunti, la posa, lo stato, la camminata e la rotazione e l'odometria.
- **Speech**: gestisce la speech recognition ed il text to speech con la possibilità di utilizzare anche varie movenze da associare al parlato del robot.
- **Sensor**: gestisce i sensori dei robot, in particolare i sonar e i bumper.
- **Tracker**: gestisce il tracciamento di volti e suoni.

Per comunicare con il robot è necessario instaurare una connessione con il robot attraverso l'indirizzo ip e la porta. Per poterlo fare viene usato il metodo qi.Session() che restituisce l'oggetto session, dal quale verranno recuperati tutti i servizi presenti all'interno del robot.

Il metodo **set_connection(ip, port)** sfrutta il metodo connect di session per collegarsi al robot sfruttando i parametri in input.

Indice

Launch

Il file di lancio qi_unipa.launch.py viene utilizzato per avviare e gestire vari nodi ROS 2 appartenenti al pacchetto qi_unipa, che interagiscono con i robot tramite il framework qi. Il file consente di configurare e lanciare simultaneamente diversi nodi responsabili di specifiche funzionalità del robot, facilitando così il controllo e l'integrazione di più moduli di interazione.

Funzionalità

Il file di lancio centralizza l'avvio dei seguenti nodi:

- **qi_unipa_sensor**: raccoglie dati dai sensori del robot.
- **qi_unipa_movement**: gestisce i comandi di movimento.
- **qi_unipa_speech**: abilita la sintesi vocale per far parlare il robot.
- **qi_unipa_tracking**: attiva il tracciamento di persone.

Argomenti di Lancio

- **ip** (string, predefinito: '192.168.0.161'): L'indirizzo IP del robot Pepper. Questo parametro consente di specificare il robot su cui verranno eseguiti i comandi.
- **port** (int, predefinito: 9559): Il numero di porta per connettersi al framework qi. Generalmente, la porta predefinita è 9559.

Gli argomenti ip e port vengono passati come parametri a ciascun nodo, garantendo che tutti i nodi comunichino con lo stesso robot.

Esecuzione

Per eseguire il file di lancio, è sufficiente utilizzare il comando:

```
$ ros2 launch qi_unipa qi_unipa.launch.py
```

O specificare il proprio indirizzo e porta:

```
$ ros2 launch qi_unipa qi_unipa.launch.py ip:=<ip_robot> port:=<porta>
```

Movement

Il nodo movement sfrutta i seguenti moduli di NaoQi:

- **ALMotion**: modulo principale del movimento che controlla la stiffness globale tramite gli stati di wake up e rest, i giunti tramite l'angolo della loro posizione, e infine la locomozione fornendo una posizione target. Può anche fornire la posizione corrente sfruttando l'odometria.
- **ALRobotPosture**: modulo che controlla la posa che assume il robot, fornendone il nome.

Lista topic subscriber

/state [int32]

Le variabili contenute nel type sono:

- int **data**

Può assumere due valori: 0 corrisponde a **wake_up**, 1 a **rest**

/joint_angles_with_speed [JointAnglesWithSpeed]

Le variabili contenute nel type sono:

- string[] **names**
- float32[] **angles**
- float32 **speed**

La lista names contiene i nomi dei giunti da muovere, angles l'angolo che devono assumere e speed la velocità con cui muoversi

/walk [Vector3]

Le variabili contenute nel type sono:

- float32 **x**
- float32 **y**
- float32 **z**

x e y rappresentano le coordinate del punto 2D che il robot deve raggiungere, z l'angolo in radianti della rotazione da effettuare. Entrambe fanno riferimento alla posizione in cui è stato avviato il robot.

/posture [PostureWithSpeed]

Le variabili contenute nel type sono:

- string **posture_name**
- float32 **speed**

la stringa indica il nome della posa che può assumere il robot, speed la velocità con cui si muove.

/hands [Hand]

Le variabili contenute nel type sono:

- string **hand**
- int32 **fun**

La stringa contiene il nome della mano da muovere, o Hands per muoverle entrambe, fun indica la funzione: 0 per aprirle, 1 per chiuderle

Lista topic publisher

/position [Vector3]

Le variabili contenute nel type sono:

- float32 **x**
- float32 **y**
- float32 **z**

Viene pubblicata ogni secondo la posizione del robot rispetto alla posizione in cui è stato avviato: x e y rappresentano le coordinate 2D e z l'angolo

Metodi definiti

Sono stati definiti i seguenti metodi:

- **set_state**: permette modificare lo stato del robot mediante pubblicazione sul topic /state
- **set_joint_angles_with_speed**: permette di modificare l'angolo dei giunti del robot mediante pubblicazione sul topic /joint_angles_with_speed
- **set_walking**: permette di modificare la posizione e la rotazione del robot mediante pubblicazione sul topic /walk
- **set_posture**: permette di modificare la posa del robot mediante pubblicazione sul topic /posture
- **set_hand**: permette di modificare lo stato delle mani del robot mediante pubblicazione sul topic /hands
- **get_position**: restituisce la posizione e l'angolo del robot mediante subscription sul topic /position

Speech

Il nodo speech sfrutta i seguenti moduli di NaoQi:

- **ALMemory**: permette di accedere ai dati contenuti in WordRecognized, ove sono memorizzate le parole riconosciute dallo speech recognition.
- **ALSpeechRecognition**: permette di utilizzare le funzionalità rilegate allo speech recognition, in particolare necessita di impostare la lingua del parlato che dovrà riconoscere ed il vocabolario contenente le parole da riconoscere.
- **ALAnimatedSpeech**: permette di utilizzare delle animazioni del robot, in questo nodo viene usato per accompagnare le risposte del robot con dei movimenti delle braccia inerenti alla risposta.

Lista topic subscriber

/listen [Bool]

Le variabili contenute nel type sono:

- bool **data**

corrisponderà nel caso di true ad attivare lo speech recognition, altrimenti a disattivarlo.

/speak [String]

Le variabili contenute nel type sono:

- string **data**

corrisponderà alla stringa da far pronunciare al robot.

Lista topic publisher

- /track [Track]

Metodi definiti

Sono stati definiti i seguenti metodi:

- **set_tts**: permette di far pronunciare al robot una stringa in input, inviata mediante pubblicazione sul topic /speak.
- **set_speech**: è la callback scatenata dal pubblicare sul topic /listen, il suo scopo è attivare o disattivare la speech recognition.
- **setup_recognition**: viene chiamata dal costruttore e permette di configurare i servizi ALSpeechRecognition, ALTextToSpeech impostando la lingua di riferimento, il vocabolario ed in fine di dichiarare l'evento WordRecognized.
- **start_recognition**: permette di avviare il riconoscimento iscrivendosi ad ALSpeechRecognition, questo causerà che il modulo inizierà a scrivere informazioni su ALMemory in "WordRecognized".
- **stop_recognition**: permette di disattivare il riconoscimento disiscrivendosi ad ALSpeechRecognition.
- **check_recognition**: viene chiamata ogni secondo dal costruttore mediante un timer, lo scopo è quello di leggere da ALMemory in "WordRecognized" l'ultima coppia (vocabolo, confidence) riconosciuta durante lo speech recognition; inoltre, se la confidence della parola riconosciuta è oltre la soglia di 0.50 allora il robot risponderà mediante un dizionario di risposte, già definite nel costruttore, associate alle parole del vocabolario.
- **answers**: permette di far pronunciare al robot una stringa associata alle parole del vocabolario, contenute dentro il dizionario definito nel costruttore; inoltre, è possibile definire dei movimenti da far eseguire al robot durante la risposta, inserendo nel dizionario delle risposte l'animazione voluta; infine, se la parola in input corrisponde a "Stop" allora avvia il metodo Stop_recognition.
- **pub_track**: pubblica un messaggio contenente i dati ricevuti in input sul topic /track

Vision

Il nodo vision sfrutta i seguenti moduli di NaoQi:

- **ALVideoDevice**: permette di accedere al flusso video delle telecamere del robot, gestendo la sottoscrizione e l'acquisizione delle immagini.

Lista topic publisher

/camera [Image]

Le variabili contenute nel type sono:

- numpy.ndarray **data**
- string **encoding**

Pubblica le immagini acquisite dalla telecamera del robot con una frequenza di 10Hz.

Metodi definiti

Sono stati definiti i seguenti metodi:

- **set_connection**: stabilisce la connessione con il robot utilizzando l'indirizzo IP e la porta specificati. In caso di errore nella connessione, termina l'esecuzione del nodo.
- **get_camera**: viene chiamato periodicamente (ogni 0.1 secondi) per:
    1. Sottoscrivere al servizio della telecamera con i parametri specificati (risoluzione VGA, spazio colore BGR, 30 fps)
    2. Acquisire l'immagine dalla telecamera
    3. Convertire l'immagine dal formato binario a un array numpy
    4. Pubblicare l'immagine convertita sul topic /camera
    5. Annullare la sottoscrizione al servizio della telecamera

Il nodo utilizza il bridge CvBridge per convertire le immagini dal formato OpenCV al formato dei messaggi ROS e viceversa.

Sensor

Il nodo sensor sfrutta i seguenti moduli di NaoQi:

- **ALMemory**: permette di accedere ai dati memorizzati in Sensor/Value sia per il sensore anteriore che per quello posteriore.
- **ALSonar**: permette di accedere ai dati che forniscono i sonar.

Lista topic publisher

/sonar [Sonar]

Le variabili contenute nel type sono:

- float32 **front_sonar**
- float32 **back_sonar**

corrispondono al valore che restituiscono i sonar anteriore e posteriore.

/bumper [Bumper]

Le variabili contenute nel type sono:

- float32 **left**
- float32 **right**
- float32 **back**

assumono il valore 1 se il corrispondente bumper urta un ostacolo, altrimenti rimangono a 0.

Inoltre, utilizza i seguenti topic:

/speak [String]

Metodi definiti

Sono stati definiti i seguenti metodi:

- **get_sonar**: viene avviata la lettura dei sensori prima iscrivendosi a Sonar_app, in modo da prelevarne il valore e pubblicarlo, e infine si disiscrive.
- **get_bumper**: preleva lo stato dei bumper per pubblicarlo, e a seconda del bumper premuto viene pubblicato sul topic /speak un messaggio che pronuncerà il robot relativo alla posizione dell'ostacolo.
- **pub_posture**: pubblica un messaggio contenente i dati ricevuti in input sul topic /posture

Tracking

Il nodo speech sfrutta i seguenti moduli di NaoQi:

- **ALTracker**: permette di tracciare diversi tagret, come la faccia o il suono, utilizzando diversi mezzi, come la testa o il corpo.

Lista topic subscriber

/track [Track]

Le variabili contenute nel type sono:

- string **target_name**
- float32 **distance**

corrispondono al nome del target, come Face o Sound, e la distanza tra target e robot.

Lista topic publisher

/posture [PostureWithSpeed]

Metodi definiti

Sono stati definiti i seguenti metodi:

- **start_tracking**: viene pubblicato sul topic /posture la posa Stand e a seconda del target vengono definiti i rispettivi parametri per poi cominciare il tracking, registrando il target ricevuto (Face,Sound). Nel caso il target sia Stop viene interrotto il tracking.
- **stop_tracking**: viene interrotto il tracking, disattivando tutti i target registrati, inoltre si posiziona, pubblicando sul topic /posture la posa Sit, in posizione di riposo.