Università degli studi di Modena e Reggio Emilia
Dipartimento di ingegneria "Enzo Ferrari"

2024

Implementazione di un sistema di controllo remoto per veicoli semi-autonomi connessi

---

Relatore: Paolo Burgio
Candidato: Alessandro Appio

Anno accademico: 2023/2024

Contents

1 Introduzione

1.1 Guida remota e guida autonoma

La guida autonoma rappresenta un complesso sistema tecnologico che integra una serie di avanzate tecnologie, metodologie e tecniche finalizzate a consentire il movimento di un veicolo senza necessità di intervento umano diretto. Un veicolo autonomo è infatti dotato della capacità di analizzare l'ambiente circostante, elaborare un percorso ottimale in base ai dati raccolti, e seguire tale percorso in modo autonomo. Questi processi fondamentali vengono generalmente suddivisi in tre fasi distinte: percezione (perception), pianificazione (planning) e controllo (control). La percezione riguarda la capacità del veicolo di raccogliere informazioni dall'ambiente circostante attraverso sensori avanzati, che possono includere telecamere, radar, lidar, e altre tecnologie di rilevamento. Questi dati vengono poi elaborati nella fase di pianificazione, durante la quale il sistema valuta le possibili traiettorie e sceglie il percorso più sicuro ed efficiente da seguire. Infine, la fase di controllo si occupa dell'esecuzione del movimento del veicolo lungo il percorso stabilito, garantendo che vengano seguite le decisioni prese nella fase di pianificazione. D'altro canto, il concetto di guida remota si riferisce a un tipo di guida in cui le decisioni relative alla direzione e al movimento del veicolo vengono prese da un essere umano, che opera a distanza utilizzando tecnologie quali sensori, attuatori, e le reti di comunicazione. In questo scenario, l'essere umano non si trova fisicamente all'interno del veicolo, ma interagisce con esso attraverso un'interfaccia remota, sfruttando la trasmissione dei dati in tempo reale per monitorare e controllare il veicolo. Tale approccio combina l'intelligenza umana con l'automazione tecnologica, rendendo possibile la guida di veicoli in situazioni in cui la presenza fisica del conducente potrebbe non essere necessaria o praticabile.

1.2 Scopo della tesi

L'obiettivo principale di questa tesi è sviluppare un veicolo capace di operare in modo autonomo in condizioni di guida normali, sfruttando un avanzato sistema di guida autonoma, ma che possa anche, su richiesta, passare alla modalità di guida remota. Questo approccio duale consente al veicolo di navigare in modo completamente indipendente quando le circostanze lo permettono, utilizzando tecnologie di percezione, pianificazione e controllo integrate, ma offre al contempo la flessibilità di essere controllato a distanza da un operatore umano qualora la situazione lo richieda. La possibilità di commutare tra guida autonoma e remota mira a garantire la massima sicurezza, adattabilità e versatilità del veicolo in una varietà di scenari operativi.

2 Piattaforma di sviluppo

In questa sezione si descrive come è composta e come è stata assemblata la piattaforma per lo sviluppo e il testing.

2.1 Rover AgileX

Il veicolo selezionato per lo sviluppo della presente tesi è un rover terrestre prodotto da AgileX, modello Hunter. Questo rover è dotato di un'interfaccia di controllo basata sul protocollo CAN (Controller Area Network), che consente una comunicazione efficiente e affidabile tra i diversi sistemi elettronici del veicolo. Il modello Hunter è stato scelto per le sue avanzate caratteristiche tecniche e per la sua versatilità, che lo rendono particolarmente adatto alle esigenze del progetto.
Oltre a fornire un'interfaccia per il controllo diretto, il veicolo è in grado di raccogliere e trasmettere una serie di dati diagnostici e operativi fondamentali per il monitoraggio e l'analisi delle sue prestazioni. Tra questi dati, un ruolo cruciale è ricoperto dall'odometria, che rappresenta la misura dello spostamento del veicolo basata sul movimento delle ruote. L'odometria è essenziale per la navigazione e la stima della posizione del rover, poiché permette di determinare il percorso seguito dal veicolo e la distanza percorsa. Questi dati, insieme ad altre informazioni sullo stato del veicolo, contribuiscono a garantire un controllo preciso e ad alimentare i sistemi di guida autonoma e remota previsti dal progetto.

2.2 GPGPU e sensore lidar

Un elemento cruciale per la realizzazione di questa tesi è stato l'identificazione e la selezione di un calcolatore embedded idoneo a gestire l'intera logica di controllo e la pianificazione, oltre alla scelta di un sensore in grado di fornire dati essenziali per la percezione dell'ambiente circostante. Per il calcolatore embedded, è stata presa la decisione di impiegare una GPGPU (General Purpose Graphic Processing Unit), una scelta motivata dalla sua elevata capacità di elaborazione parallela, che risulta particolarmente vantaggiosa per eseguire complessi algoritmi di controllo e di pianificazione in tempo reale. La GPGPU selezionata opera con il sistema operativo Ubuntu 20.04, noto per la sua stabilità, ampia compatibilità con hardware di ultima generazione, e supporto per lo sviluppo di applicazioni avanzate, inclusi strumenti specifici per l'elaborazione grafica e la gestione di risorse computazionali. Per quanto concerne il sensore, la scelta è ricaduta su un sensore Lidar (Light Detection and Ranging). Questo dispositivo sfrutta la tecnologia laser per determinare la distanza di vari punti nell'ambiente circostante, calcolando il tempo di ritorno dei raggi laser emessi. Il Lidar fornisce una mappa dettagliata della topografia dell'ambiente, consentendo al sistema di percezione di creare rappresentazioni tridimensionali accurate, fondamentali per il riconoscimento degli ostacoli, la navigazione e la pianificazione del percorso del veicolo. La combinazione di una GPGPU performante e un sensore Lidar

avanzato rappresenta una solida base tecnologica per lo sviluppo di un sistema di guida autonoma e remota altamente efficiente.

3 ROS

In questa sezione si passa alla descrizione di ROS e del suo utilizzo.

3.1 Nodi

ROS o Robotic Operating System è un insieme di librerie e strumenti utili alla creazione di applicativi dedicati al controllo di robot. Nello specifico ROS ci permette di creare unità di esecuzione o processi chiamati nodi. Un nodo ha la capacità di eseguire calcoli, interfacciarsi con periferiche o altro, ma la principale caratteristica di un nodo è la sua capacità di comunicare con gli altri nodi ROS in esecuzione. Ciò ci permette di rappresentare con ogni nodo un modulo funzionale all'esecuzione della task del robot, per fare esempi pratici lo stack utilizzato per il controllo autonomo del rover è composto da diversi nodi, i principali sono:

- hunter_ros2_node: Gestisce la comunicazione tra i nodi ROS e l'interfaccia CAN del veicolo
- urg_node: Comunica agli altri nodi la scan effettuata dal sensore lidar
- particle_filter: calcola la localizzazione del mezzo a partire dalla mappa dell'ambiente e dalla scan del sensore
- telemetry_node e control_node: come descritto prima, gestiscono la comunicazione tra ROS ed MQTT

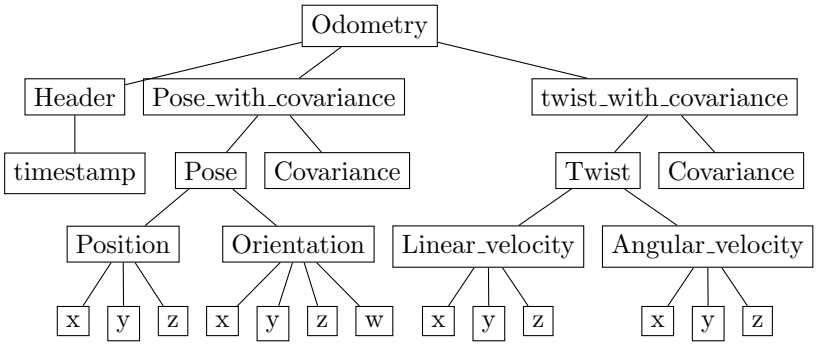
Tutti questi nodi sono capaci di comunicano tra loro per scambiarsi informazioni utili all'esecuzione del robot.

3.2 Comunicazione tra nodi

Sorge però spontaneo domandarsi come questi nodi comunichino tra loro e come facciamo soprattutto a riconoscere di che tipo di informazione si tratti. Una comunicazione ROS è formata da 3 elementi:

- Topic: Il protocollo utilizzato da ROS è di tipo publish/subscribe, ciò vuol dire che durante l'esecuzione dei nodi si vanno a creare dei topic, ovvero stringhe che utilizzano come separatore il carattere '/' e che ci permettono di suddividere tutti i diversi dati da inviare. Un esempio sono le scan lidar che vengono pubblicate sul topic "/scan". Ogni nodo può decidere se fare la subscribe a quel nodo (ovvero ricevere tutti i dati inviati attraverso esso), fare delle publish (ovvero pubblicare dati su di esso) o se semplicemente ignorarlo.
- Message type: Una volta scelto un topic però si dovrà anche decidere quali informazioni saranno ammesse su questo, ROS fornisce diversi tipi di dato inviabile su un singolo topic. Un esempio è il tipo di dato utilizzato dal particle filter ovvero "Odometry messages", che descrivono la posizione

(o meglio l'odometria) di un oggetto nello spazio ed è strutturanto nel seguente modo:



tutti i tipi di messaggio sono consultabili online sulla documentazione di ROS

- Content: è il dato che dobbiamo inviare e che deve essere incapsulato nel tipo di dato fornitoci da ROS

4 MQTT

MQTT è un protocollo di rete studiato per applicazioni IOT.

4.1 Infrastruttura

Il protocollo MQTT è un protocollo di tipo publisher/consumer e si avvale di 3 principali figure:

- Publisher
- consumer
- Broker

4.2 Formattazione messaggi

I messaggi in MQTT non sono altro che stringhe, per il nostro ambito applicativo però è necessaria una formattazione che renda ben distinguibili i campi e i valori del nostro messaggio ROS tradotto, per questo ci si avvale del formato JSON.

4.3 Topic

Come con ROS, MQTT si avvale di un meccanismo di topic per distinguere i la tipologia di messaggi inviati.

5 Funzionamento

Nella seguente sezione si descrive il funzionamento generale dello stack di guida remota e come un server si può interfacciare con il veicolo per svolgere tali operazioni

5.1 Informazioni scambiate tra veicolo e server

Per adempire allo scopo di guida remota sono necessarie due figure distinte: Il veicolo ed un server. Il veicolo è stato descritto nella sezione precedente, il server invece è quella figura del sistema con cui l'umano si interfaccia e che fornisce sia i dati prelevati dal veicolo sia un metodo per il controllo del veicolo. Per garantire il funzionamento della struttura è quindi necessario lo scambio di informazioni tra le due parti. Queste informazioni sono: - I messsaggi di controllo per i motori - La pointcloud fornita dal lidar - l'odometria calcolata dal veicolo Le informazioni tra il server ed il veicolo sono scambiate tramite protocollo di rete MQTT, un protocollo studiato per il mondo dell'IOT e per tali applicazioni. Sia a bordo del veicolo che sul server è istanziata un versione di ROS (Robotic Operative System) ovvero un software utilizzato per creare diversi nodi (o processi paralleli) e per permettere lo scambio di informazioni tra essi (Tramite topic e messaggi).

5.2 Funzionamento lato veicolo

A bordo del mezzo sono istanziati due nodi ROS: uno incaricato di gestire l'invio dei dati del sensore e dell'odometria, ed uno incaricato di ricevere i messaggi di controllo. Per comodità li chiameremo rispettivamente *telemetry node* e *control node*. Come descritto prima il *telemetry node* si incarica d ricevere messaggi da ROS contenenti la pointcloud del lidar e l'odometria del mezzo, per poi formattare tali messaggi come stringhe JSON ed inviare questi messaggi tramite protocollo MQTT al server in due topic dedicati. Per quanto riguarda il *control node* invece, questo si incarica di ricevere messaggi dal server che riguardano il controllo del mezzo, una volta ricevuti i messaggi come stringa JSON questi vengono convertiti in messaggi ROS e vengono inviati ai driver del rovere che permettono poi il controllo tramite CAN del veicolo.

5.3 Funzionamento lato server

Il server, a differenza del veicolo, esegue un solo nodo ros. Lo scopo di tale nodo non è tanto quello di prendere decisioni sul movimento del veicolo ma è quello di ricevere i dati dal veicolo, farli processare da un terzo ente, ricevere le decisioni di questo terzo ente espresse come azioni che il veicolo deve eseguire ed inviarle al veicolo. Nello specifico tale nodo riceve tramite protocollo MQTT i dati dal veicolo, per poi formattarli come messaggi ros e pubblicarli sui relativi topic ros. Una volta che questi dati vengono ripubblicati questi possono essere elaborati o da uno stack di guida autonoma o da un effettivo pilota che a distanza vedrà

i dati rilevati dal veicolo e prenderà decisioni sul controllo. Una volta prese queste decisioni dal terzo ente, queste verranno pubblicate sul topic ros addetto al trasporto di tali informazioni e a cui il nodo ros del server è iscritto. Una volta che che il nodo avrà letto il dato, lo formatterà in una stringa JSON e la invierà tramite MQTT al veicolo che eseguirà tali comandi

6 Sviluppi futuri e conclusioni

6.1 Problemi

6.2 Soluzioni

6.3 Applicazioni pratiche