Politecnico di Torino

POLITECNICO DI TORINO

Laurea Magistrale Ingegneria Informatica

Tesi di Laurea Magistrale

Emulazione neurale di un controllo per il mantenimento di corsia su un modello in scala

Relatori
Prof. Stefano Malan
Prof. Michele Pagone

Candidato
Matteo Ghiso

Dicembre 2025

Indice

Elenco delle figure

Capitolo 1

Introduzione

Negli ultimi anni lo sviluppo di algoritmi per la guida autonoma è stato uno dei campi più prolifici, e discussi nell'industria automobilistica, ad oggi infatti assistiamo a una continua evoluzione di queste tecnologie spinta da un'accesa competizione tra le case costruttrici.
L'obiettivo è lo sviluppo di algoritmi capaci di sostituirsi, in parte o del tutto, al guidatore umano nella conduzione del veicolo, con notevoli vantaggi in termini di efficienza, comfort e sicurezza.
Lo sviluppo di sistemi di guida autonoma rappresenta una sfida complessa, essi richiedono infatti sistemi di visione in grado di interpretare l'ambiente esterno mediante una rete di sensori tra cui videocamere, GPS e LiDAR, e un sistema di elaborazione in grado di guidare il veicolo verso la destinazione attraverso scenari sconosciuti e complessi, il tutto garantendo alti standard di sicurezza, e prestazioni compatibili con un'esecuzione in tempo reale.

1.1 Le competizioni di guida autonoma

Il rapido avanzamento tecnologico ha portato alla nascita di molte competizioni riguardanti lo sviluppo di hardware e software per la guida autonoma. In queste competizioni gli studenti universitari, e non solo, hanno l'opportunità di mettersi alla prova e sviluppare competenze riguardanti un settore sicuramente cruciale nella mobilità futura.
A oggi si contano decine di competizioni da molte zone del mondo, ognuna con un interesse ben specifico e con diverse caratteristiche, tra le più importanti troviamo:

- **Formula Student Driverles** [1].
  Una delle competizioni più complete e riconosciute nell'ambito universitario, richiede lo sviluppo di un veicolo a guida autonoma in ogni suo aspetto.
  Viene infatti richiesta oltre alla progettazione dell'intero veicolo in scala 1:1, lo sviluppo di algoritmi ad hoc e un piano di business corredato da uno studio

sui costi di produzione.
La valutazione avviene poi sulla base di prove sul campo e di una presentazione del piano di business e dei costi.

- **Indy Autonomous Challenge** [2].
  La IAC è la più prestigiosa competizione riguardante il mondo dei veicoli a conduzione autonoma, è aperta a squadre studentesche universitarie, e riguarda lo sviluppo di un algoritmo di guida autonoma su un'auto da corsa in scala $1:1$ (Fig. 1.1), il cui telaio è progettato e fornito dall'italiana *Dallara Automobili*, leader nello sviluppo e produzione di veicoli da competizione, nonché fornitore unico dei telai in competizioni di altissimo livello quali Indy Car, Formula 2 e Formula 3.
  La sfida impone lo sviluppo di sistemi con alti requisiti di sicurezza, necessità dettata dalle elevate velocità richieste dalla sfida.
  La competizione prende luogo in alcuni dei più importanti teatri del motorsport, quali Indianapolis, Monza e Las Vegas.

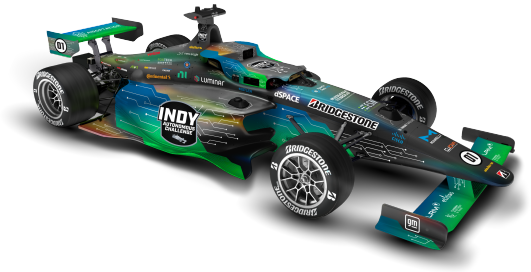


Figura 1.1: Veicolo ufficiale della Indy Autonomous Challenge

- **DARPA Grand Challenge** [3]
  La DARPA Grand Challenge indetta da DARPA (Fig. 1.2), il dipartimento di ricerca avanzata della difesa degli Stati Uniti d'America, è una competizione di guida autonoma, la cui prima edizione nel 2004 poneva ai team, studenteschi e non, l'ambizioso obiettivo di sviluppare un veicolo in grado di attraversare il deserto del Mojave senza intervento umano.
  La competizione, nella sua storia conta solo 3 edizioni 2004, 2005 e 2007, tuttavia una competizione così ambiziosa ha portato importanti innovazioni nell'ambito della guida autonoma, nel 2005 infatti la Stnaford University ha sviluppato appositamente per questa competizione il Controllore Stanley, il quale sarà in parte argomento di questa tesi, diventando a tutti gli effetti il primo team a portare a termine il percorso lungo più di $240km$.

Figura 1.2: Manifesto della DARPA Grand Challenge del 2004

- **Bosch Future Mobility Challenge** [4].
  Questa competizione, indetta da *Bosch Engineering Center Cluj*, è stata la base da cui si è partiti per questo lavoro.
  In questa competizione, su cui si lavora su un modello in scala 1:10, si è chiamati a sviluppare un algoritmo capace di condurre il modello in una scena cittadina (Fig. 1.3). Lo scopo è quello di superare una serie di prove, tra cui parcheggio, superamento di incroci e guida in condizioni di traffico.
  Questa competizione, che mette al centro la mobilità del futuro, permette agli studenti di sviluppare conoscenze concrete e sicuramente di fondamentale importanza in ottica futura, mediante la sperimentazione in un ambiente realistico.

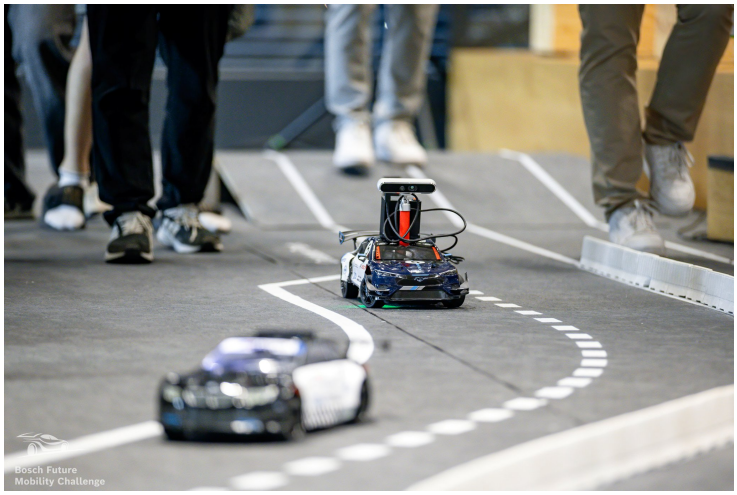


Figura 1.3: Bosch Future Mobility Challenge

1.2 La guida autonoma

In termini funzionali, seguendo la classificazione proposta da David Shinar in *"Human Behavior and Traffic Safety"* [5], un algoritmo di guida autonoma è composto da 3 funzioni fondamentali, operanti a diversi a diversi livelli (Fig. 1.4):

- **Funzioni strategiche**: comprendono tutte le operazione di pianificazione, tra cui navigazione, selezione delle destinazioni e tappe intermedie.

- **Funzioni tattiche**: spesso indicate con l'acronimo DDT (Dynamic Driving Task), comprendono il riconoscimento di oggetti e l'interpretazione dell'ambiente circostante OEDER (Object and Event Detection and Response).
  Le operazioni OEDER portano alla pianificazione delle azioni da svolgere nel breve periodo, quali, ad esempio, scelta della velocità, direzione e distanza da mantenere rispetto agli altri veicoli presenti nella carreggiata.

- **Funzioni operazionali**: riguardano il controllo delle dinamiche laterali e longitudinali del veicolo tramite l'attuazione dei sistemi di controllo, principalmente sterzo, acceleratore e freno, al fine di attuare le manovre stabilite dai livelli tattico e strategico.

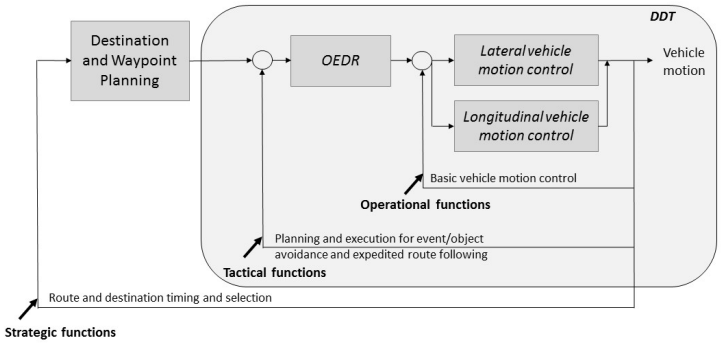


Figura 1.4: Schema a blocchi sistema di guida autonoma

Ad oggi la SAE (Society of Automotive Engeneers) definisce 6 diversi livelli di guida autonoma [6] (Fig. 1.5):

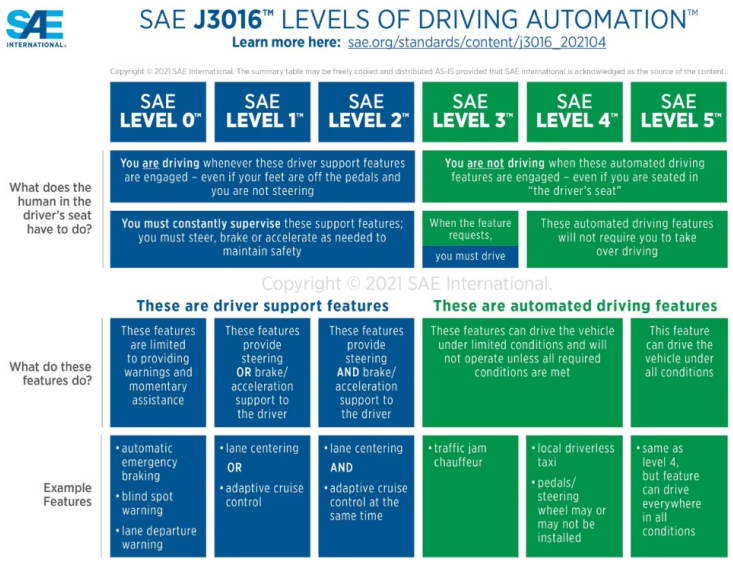


Figura 1.5: Classificazione SAE dei veicoli a guida autonoma

- **LIVELLO 0 - NESSUNA AUTOMAZIONE**: non è previsto nessun aiuto al guidatore al quale è totalmente affidata la conduzione del veicolo.
- **LIVELLO 1 - ASSISTENZA ALLA GUIDA**: il controllo e la supervisione del veicolo sono affidati al guidatore, al quale viene affiancato un controllo su uno e uno soltanto dei due principali organi di guida (sterzo o acceleratore/freno).
- **LIVELLO 2 - GUIDA AUTONOMA PARZIALE**: il veicolo può condursi autonomamente, gestendo simultaneamente sterzata e accelerazione/-frenata, tuttavia il guidatore ha comunque il controllo del veicolo a sua discrezione.
- **LIVELLO 3 - GUIDA ASSISTITA CONDIZIONALE**: il veicolo è in grado di condursi autonomamente in un numero limitato di scenari, nei quali non è richiesta la supervisione umana. Il conducente deve tuttavia essere reattivo nell'intervento in caso di segnalazione da parte del veicolo.
- **LIVELLO 4 - GUIDA ASSISTITA AVANZATA**: il veicolo è in grado di gestire un numero elevato di scenari tuttavia richiede supervisione e al bisogno, intervento umano.
- **LIVELLO 5 - GUIDA AUTONOMA COMPLETA**: il veicolo è completamente autonomo, non richiede supervisione e non è necessario fornire al guidatore gli apparati per il controllo del veicolo (volante, acceleratore e freno).

L'avvento della guida autonoma oltre a ridurre il peso dell'errore umano, può aprire la via a un sistema di mobilità che non si limita al veicolo, ma include l'infrastruttura stradale, mettendo in comunicazione e permettendo la collaborazione tra tutti gli agenti che impegnano la strada, limitando ulteriormente i rischi per l'uomo.
Un ulteriore possibile vantaggio risiede nell'ottimizzazione che un controllo automatizzato del veicolo può apportare ai consumi, risultando in questo modo più economico ed ecologico.
Ad oggi lo sviluppo di queste tecnologie ha portato molti dei veicoli sul mercato a essere classificati al LIVELLO 1 o LIVELLO 2, con alcune soluzioni nel mondo commerciale che raggiungono il LIVELLO 3.

1.2.1 Riconoscimento e mantenimento di corsia

Durante questo lavoro si analizzerà in particolar modo uno degli aspetti fondamentali della guida autonoma, ossia il mantenimento di corsia. Il compito del mantenimento di corsia consiste nell'attuazione delle manovre necessarie al fine di mantenere il veicolo all'interno della corsia designata, un compito reso difficile dalla grande varietà di percorsi come strade con una o più corsie, talvolta in assenza di segnaletica orizzontale (strisce) e con curve più o meno impegnative.
Un sistema di mantenimento di corsia assolve essenzialmente a due compiti:

- **Riconoscimento di corsia**: consiste nel riconoscere e monitorare costantemente la posizione e l'allineamento del veicolo in relazione alla corsia raccogliendo informazioni dalle immagini fornite da una o più videocamere montate a bordo del veicolo.
- **Mantenimento di corsia**: consiste nel controllo dello sterzo e della velocità al fine di mantenere il veicolo all'interno della corsia.

Tale sistema, al fine di attuare un controllo efficiente, deve essere progettato alla luce delle caratteristiche fisiche del veicolo per cui è destinato, pertanto necessita della conoscenza delle dinamiche laterali e longitudinali del veicolo in questione.

1.3 Presentazione del lavoro svolto

Come è facile intuire il compito di monitorare costantemente il veicolo, l'ambiente e allo stesso tempo prendere decisioni in tempo reale è oneroso a livello computazionale, per questo i veicoli avanzati dispongono di hardware costosi e con elevata capacità di calcolo.
Talvolta i limiti della potenza di calcolo a disposizione costringono ad adottare soluzioni non ottimali, l'elaborazione dell'immagine è infatti complessa e questo su hardware limitati è fortemente problematico, costringendo a implementare controllori più semplici e veloci con prestazioni in termini di guida inferiori. Attualmente,

infatti, il sistema di controllo implementato a bordo del modello fa uso di un controllore con pianificazione del guadagno (gain scheduling), capace di mantenere il veicolo all'interno della corsia rilevata dalla telecamera.
Tale soluzione, pur garantendo un funzionamento stabile, presenta prestazioni limitate e non consente di gestire scenari di guida complessi.
Il lavoro, presentato in questa tesi, nasce quindi a partire da queste considerazioni, con lo scopo di valutare la possibilità di emulare un controllore più complesso, quindi non eseguibile in tempo reale sulla scheda Raspberry Pi5, mediante una rete neurale leggera in grado di riprodurne il comportamento.
Si intende quindi sfruttare la capacità delle reti neurali di apprendere relazioni non lineari e quindi imitare il comportamento di un controllore funzionante, secondo il paradigma dell'apprendimento tramite imitazione (Imitation Learning), al fine di ottenere un controllore sofisticato ed efficiente ma computazionalmente poco oneroso.

Capitolo 2

La piattaforma sperimentale utilizzata

La piattaforma, utilizzata durante questo progetto, è un modello in scala 1:10 di un veicolo a guida autonoma, il modello deriva da una competizione, la BFMC (*Bosch Future Mobility Challenge*), alla quale l'ateneo ha partecipato in anni passati.
La BFMC mette alla prova team studenteschi fornendo loro una piattaforma standardizzata su cui sviluppare un algoritmo di guida autonoma, in grado di guidare il modello in scala lungo un percorso che simula una scena cittadina, con intersezioni, ostacoli, segnali stradali e diverse condizioni di visibilità.

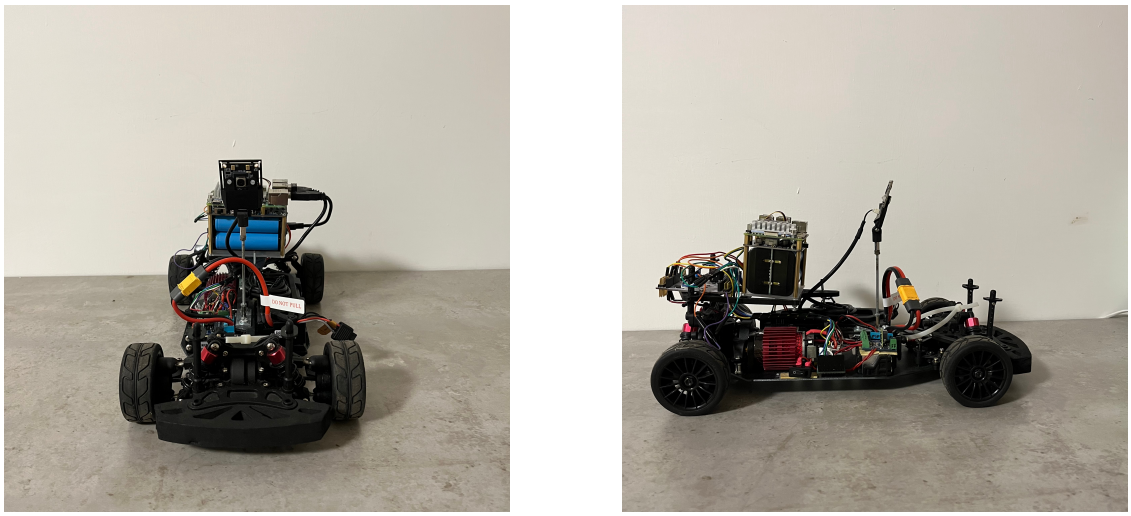


Figura 2.1: Il modello utilizzato

2.1 Descrizione del modello

Il modello (Fig. 2.1) dopo essere stato utilizzato nella BFMC 2019 è stato mantenuto e aggiornato dallo staff LED del Politecnico di Torino e da studenti tesisti che negli anni si sono succeduti nello sviluppare diversi aspetti della guida autonoma aggiornando nel tempo il modello con componenti con prestazioni superiori.
Nel particolare il modello fornito presenta un telaio in scala 1:10, il quale misura $37cm$ di lunghezza, $20cm$ di larghezza e presenta un interasse di $27cm$. A bordo sono poi alloggiati tutti i componenti necessari alla visione, elaborazione e gli organi di trasmissione compresi dell'elettronica necessaria al loro funzionamento (Fig. 2.2).
In particolare a bordo si trovano due schede elettroniche con compiti e caratteristiche ben distinti:

- Il controllo a **basso livello** del motore utilizzato per la trazione e del servomotore di sterzo è affidato a una scheda NUCLEO-F401RE, il software per questa scheda è stato fornito inizialmente da Bosch e negli anni leggermente migliorato. Il compito principale della scheda NUCLEO è ricevere tramite comunicazione seriale i valori di velocità e angolo di sterzo calcolati dalla scheda principale e gestire di conseguenza il motore e il servomotore garantendone il corretto funzionamento.
  Il protocollo di comunicazione utilizzato richiede alla scheda di rispondere ai messaggi inviati dalla scheda Raspberry Pi5 con dei messaggi di conferma che garantiscono la corretta ricezione ed esecuzione dell'azione richiesta, questo protocollo garantisce maggiore robustezza e affidabilità alla comunicazione tra le due schede.

- La logica di **alto livello** è eseguita su una scheda Raspberry Pi5 [7], inizialmente la BFMC ha messo a disposizione una scheda Raspberry Pi4, la quale è poi stata sostituita per aumentare la capacità di calcolo. I compiti di questa scheda sono l'acquisizione/elaborazione delle immagini e il calcolo dei valori di velocità e sterzo da applicare. Il software presente all'inizio di questa esperienza comprende un ciclo di visione funzionante e un semplice controllo, i quali forniscono un'ottima base di partenza per questo lavoro, il tutto scritto in C++ con il supporto della libreria *OpenCV* per l'elaborazione dell'immagine e il riconoscimento delle linee di demarcazione della corsia.

La comunicazione tra le due schede è realizzata mediante un cavo miniUSB che supporta una comunicazione a 19200bps.
Il compito della cattura immagini è affidato a una fotocamera modello DFRobot FIT0729, che presenta un sensore a 8 Megapixel e un campo di visione di 75°, collegata direttamente al Raspberry Pi5 mediante una delle interfacce USB. La fotocamera è montata sul veicolo a un'altezza di $15cm$ dal suolo, con un arretramento

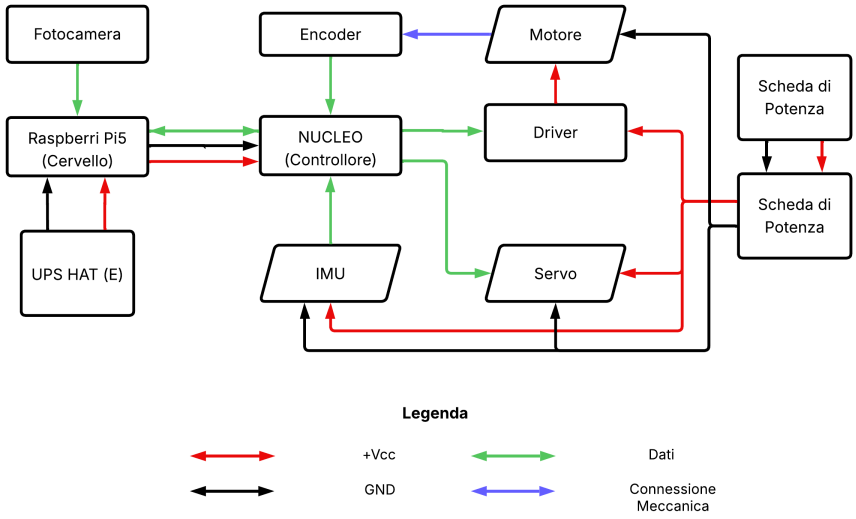


Figura 2.2: Schema logico di collegamento delle componenti

di 7cm rispetto all'assale anteriore e con un orientamento di circa 5° verso il basso. Per l'attuazione dello sterzo è presente a bordo un servomotore RS-610WP.
Per quanto riguarda la trasmissione invece è presente un motore modello Reely 531009 con il relativo driver VNH5019A-E dotato di ponte H per l'inversione di marcia, all'asse del motore è inoltre collegato un Encoder incrementale AMT10 utilizzato dal controllore NUCLEO,per la gestione in retroazione della velocità.
L'alimentazione per i motori è garantita da una scheda di potenza prototipale fornita da Bosch, la quale ha la funzione di alimentare tutti i componenti della vettura ad eccezione delle due schede programmabili, per le quali è prevista un'alimentazione separata.
A completare la dotazione di sensori di cui è dotato il veicolo troviamo un sensore IMU (Inertial Measurement Unit, Unità di Misura Inerziale), connesso anch'esso alla scheda NUCLEO.
Per quanto riguarda le prestazioni offerte dal modello, sono presenti limitazioni sia per quanto riguarda l'angolo di sterzo che la velocità massima.
In particolare per quanto riguarda l'angolo di sterzo, esso è limitato nell'intervallo [-25°,+25°], la limitazione è imposta direttamente a livello di programmazione nel codice eseguito dal controllore NUCLEO, in modo tale da non provocare danni al modello stesso.
La velocità è invece limitata a 1.8km/h, quest'ultima non è una limitazione imposta dal hardware presente, bensì un limite imposto per evitare velocità eccessive, alle quali il modello risulterebbe impossibile da controllare efficacemente con la componentistica presente.

2.1.1 Miglioramenti hardware apportati al modello

Durante lo svolgimento di questa tesi, in collaborazione con la collega e tesista Alessia Sedda, sono stati apportati importanti aggiornamenti all'alimentazione del veicolo.
In primo luogo dove in precedenza veniva utilizzato un powerbank per alimentare la scheda Raspberry Pi5, la quale fornisce a sua volta alimentazione alla scheda NUCLEO, è stato montato un gruppo di continuità (UPS) appositamente sviluppato per la scheda da noi utilizzata. La scheda Raspberry Pi5 infatti necessita di una tensione di alimentazione di 5V con un assorbimento di corrente massimo pari a 5A.
Per questo scopo è stato selezionato l' UPS HAT (E) prodotto da Waveshare, appositamente progettato per la serie Raspberry Pi, il quale oltre a essere completamente compatibile con la scheda presente a bordo, supporta la stessa alimentazione del Raspberry Pi5 mediante USB-C per la ricarica delle 4 batterie serie 21700.
Durante lo svolgimento dei vari test è stato inoltre necessario sostituire la batteria LiPo principale utilizzata per l'alimentazione dei motori, a questo scopo è stata utilizzata una batteria al litio da $7{,}4V$, composta da due celle da 3,6V collegate in serie, il modello utilizzato è, nello specifico, la Gens ACE 2S 7,4V 6200MHA 100C.

2.2 Il tracciato

Anche il tracciato utilizzato durante il progetto è conforme a quanto indicato da Bosch durante la BFMC, si tratta di un tracciato con diverse sezioni distinte con tratti rettilinei, incroci e curve a 90°.
Il tracciato ha misure standard, in particolare le corsie sono larghe $35cm$ e possono essere delimitate da linee tratteggiate o continue, entrambe larghe $2cm$ con i segmenti che compongono le strisce tratteggiate misurano di $4{,}5cm$ di lunghezza.
Alcuni esempi di tracciato sono riportati in Fig. 2.3.

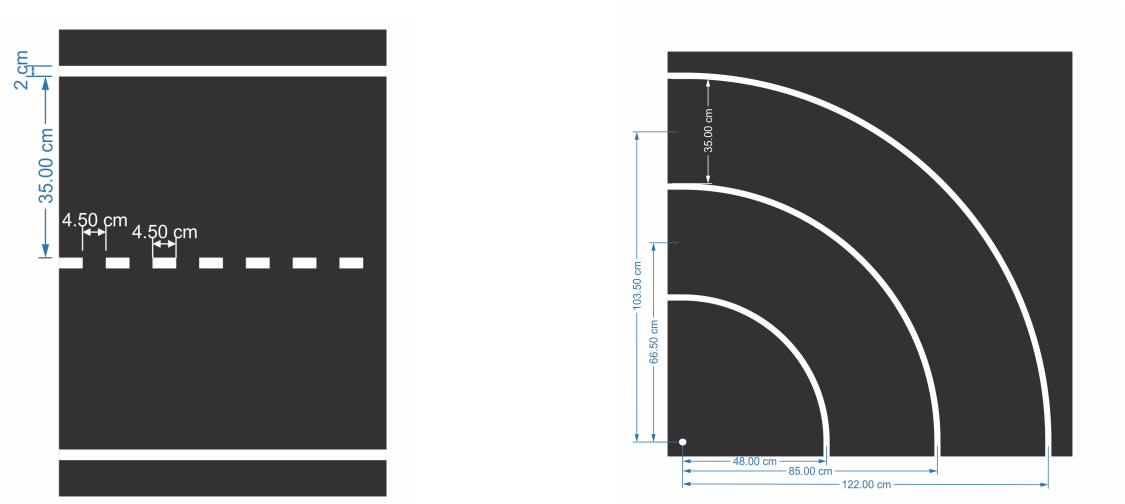


Figura 2.3: Esempi di tracciato

Capitolo 3

Riconoscimento di corsia

Per il modello in questione è già stato sviluppato un algoritmo per il mantenimento di corsia , il quale è stato argomento di tesi svolte in precedenza da alcuni colleghi [8], alla quale si rimanda per eventuali approfondimenti riguardante la loro implementazione.
Tale algoritmo è stato ripreso e vi sono state apportate alcune migliorie con il prezioso aiuto della collega e tesista Alessia Sedda [9].
In particolare, il codice, progettato per essere eseguito a bordo della scheda Raspberry Pi5, è interamente sviluppato in C++. La scelta di questo linguaggio è motivata dalle ottime prestazioni offerte dai linguaggi compilati rispetto a quelli interpretati, e dalla vastità di librerie disponibili a supporto dello sviluppo: l'algoritmo di visione implementato fa infatti uso di molte delle funzioni fornite dalla libreria `OpenCV` [10], la quale mette a disposizione potenti strumenti per l'elaborazione delle immagini, e per questo è molto utilizzata nel campo della visione computerizzata.
Inoltre, il linguaggio C++ supporta nativamente la programmazione a oggetti, caratteristica che consente una maggiore modularità e riusabilità del codice sviluppato.
Entrando nei particolari dell'algoritmo possiamo distinguere due parti fondamentali, la parte relativa alla visione e interpretazione dell'ambiente circostante, la quale attua il riconoscimento della corsia, argomento di questo capitolo, e una seconda parte relativa al mantenimento di corsia, mediante il controllo laterale e longitudinale del veicolo, trattata in seguito e parte del lavoro svolto durante questo progetto.
L'implementazione descritta in seguito, è suddivisa essenzialmente in due sezioni.
La prima ha lo scopo di estrarre un'immagine binarizzata nella quale vengono isolate linee di demarcazione della corsia, riconoscendone il colore e privilegiando la segnaletica di cantiere distinta dal colore giallo, ignorando in loro presenza le altre tipologie di segnaletica.
La seconda parte ha invece lo scopo di modificare il punto di vista dell'immagine, rimuovendo la prospettiva, e in seguito individuare le linee di demarcazione della

corsia, generando di conseguenza un riferimento da inseguire.
A partire dal riferimento è necessario poi calcolare gli errori di inseguimento ai quali il controllore farà riferimento per le sue operazioni (Figura 3.1):

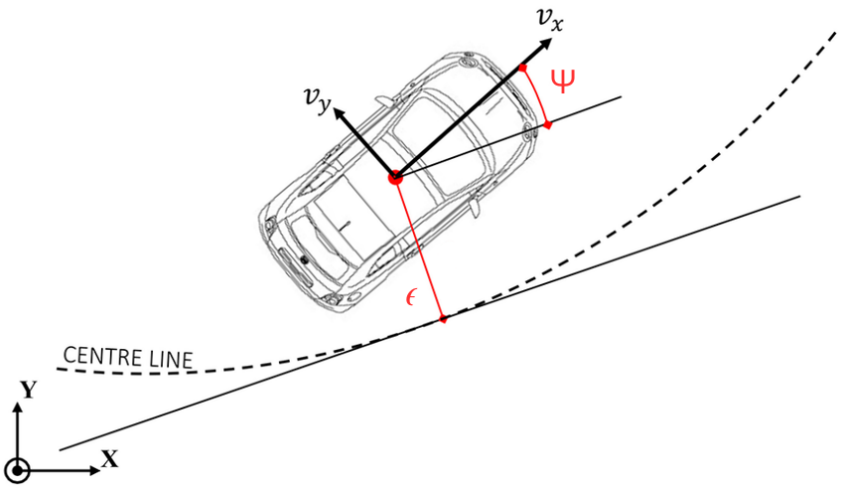


Figura 3.1: Errore di allineamento (ϵ) ed errore laterale (ψ)

- **Errore di allineamento**: definito come l'angolo tra l'asse longitudinale del veicolo e la direzione della traiettoria di riferimento.

- **Errore laterale**: è definito come la distanza tra la traiettoria di riferimento e il centro di massa del veicolo, spesso posto arbitrariamente in posizione centrale lungo l'asse longitudinale del veicolo, o centralmente in corrispondenza dell'asse anteriore.

3.1 Riconoscimento del colore

Una volta raccolta l'immagine, di dimensioni 640x480 *pixel*, essa viene elaborata tramite una serie di trasformazioni il cui scopo è quello di ottenere un immagine binaria della corsia che precede il veicolo.
Questa serie di operazioni, particolarmente critica per il funzionamento dell'algoritmo, risulta particolarmente complessa da rendere stabile in ogni condizione di luce, nonostante la mole di accorgimenti implementati a tale scopo.

Trasformazione HSV

Una volta immagazzinata l'immagine in una variabile di tipo `Mat`, la prima operazione eseguita consiste nella conversione dell'immagine dal formato RGB (Red-Green-Blue) al formato HSV (Hue Saturation Value) [11] [12].

La rappresentazione HSV è una rappresentazione del colore alternativa alla classica rappresentazione RGB.
All'interno della rappresentazione HSV le tre componenti sono:

- **H - Tonalità**: rappresenta la tonalità del colore, espressa come angolo in riferimento alla ruota dei colori (Fig. 3.2).
- **S - Saturazione**: la saturazione rappresenta la "purezza" del colore, a partire da 0 in assenza di colore (grigio) fino al valore 1 corrispondente al colore puro.
- **V - Luminosità**: indica quanto il colore è o meno scuro, un valore di 0 corrisponde a un pixel nero, mentre 1 indica la massima luminosità possibile.

Le tre componenti vengono espresse con valori su 8 bit, ossia nell'intervallo [0,255].
Lo spazio HSV è spesso utilizzato nelle applicazioni in cui è necessaria la manipolazione del colore, come ad esempio l'applicazione di filtri o la segmentazione dell'immagine.

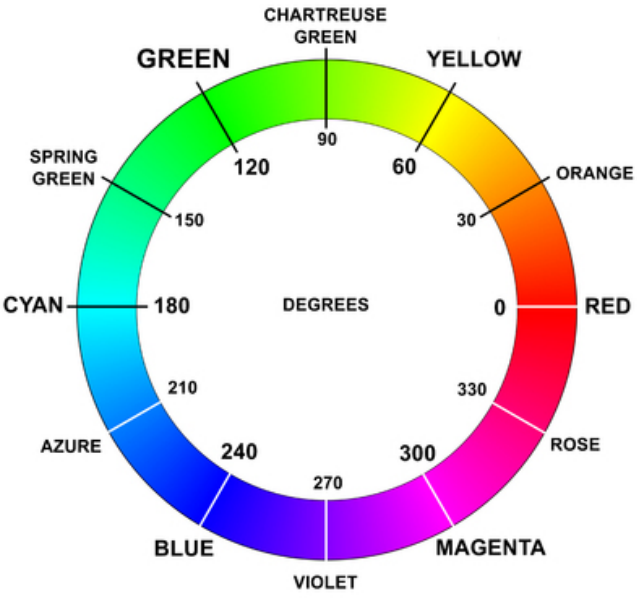


Figura 3.2: Ruota dei colori

3.1.1 Estrazione dei pixel bianchi e gialli

Una volta ottenuta la rappresentazione HSV dell'immagine di partenza, anch'essa immagazzinata in una variabile di tipo `Mat`, si procede con la creazione di due

maschere binarie distinte per il riconoscimento dei pixel bianchi e gialli, rispettivamente `maskWhite` e `maskYellow`, parte del lavoro svolto dalla collega consiste, infatti, nella gestione della guida autonoma in presenza di lavori stradali.
A questo scopo è innanzitutto necessario stabilire l'intervallo di valori di colore, saturazione e luminosità da assegnare ai due colori da individuare.
Stabilire tali intervalli in modo idoneo a ogni possibile condizione di luce non è tuttavia stato possibile, infatti per ovviare al problema è stata implementata una procedura manuale di taratura delle due maschere, da eseguire prima dell'avvio effettivo dell'algoritmo.
Una volta stabiliti gli intervalli ottimali per il bianco e il giallo, questi vengono utilizzati per ottenere le due maschere binarie, in cui un valore di 255 (bianco) indica un pixel del colore relativo alla maschera corrispondente, mentre 0 (nero) indica un qualsiasi altro colore. Una volta ottenute le due maschere si procede a classificare lo scenario di guida [9], in particolare da ogni maschera si estrae il numero di pixel bianchi e gialli (`whitePixels` e `yellowPixels`) e sulla base di questo si classifica lo scenario in:

- **Presenza di lavori stradali**: se $yellowPixels \geq whitePixels/2$, in questo caso per, il resto dell'algoritmo, viene utilizzata la maschera `maskYellow`;
- **Assenza di lavori stradali**: se $yellowPixels < whitePixels/2$, in questa situazione si sovrappongono le due maschere mediante la funzione `bitwise_or` fornita da `OpenCV`, ottenendo una maschera dove il valore 255 corrisponde a un pixel bianco o giallo all'interno dell'immagine originale, e si procede con l'algoritmo utilizzando il risultato di questa operazione.

Una volta ottenuta la maschera si procede ad applicare tale maschera all'immagine, precedentemente trasformata da RGB a scala di grigi, tramite la funzione `bitwise_and` fornita da `OpenCV`, nell'applicazione della funzione logica AND, i pixel bianchi all'interno della maschera si comportano come trasparenti, ne risulta quindi un'immagine dove le linee di demarcazione della corsia sono isolate dal resto dell'immagine.

3.1.2 Riempimento delle discontinuità e binarizzazione dell'immagine

L'operazione di riempimento delle discontinuità viene applicata all'immagine, ottenuta mediante l'applicazione della maschera, per rimuovere piccole discontinuità dovute alla presenza di rumore nell'immagine originale.
Per queste operazioni viene definita una matrice unitaria 3x3 (`kernel`), la quale definisce la regione locale su cui vengono applicate le seguenti operazioni:

1. **Dilatazione**: questa operazione eseguita sui singoli pixel dell'immagine permette di espandere le zone bianche.

Ogni pixel acquisisce il valore massimo presente nella porzione d'immagine individuata dalla matrice `kernel`, l'applicazione di questa trasformazione permette alle zone bianche di espandersi, e di conseguenza le discontinuità di piccole dimensioni all'interno di zone bianche vengono eliminate.

2. **Erosione**: questa operazione permette l'espansione delle zone nere.
   Al contrario dell'operazione di dilatazione, ogni pixel acquisisce il valore minimo presente all'interno dell'area individuata dal `kernel`, tale operazione permette l'eliminazione di piccole aree isolate di colore bianco.

L'applicazione in sequenza di queste due operazioni permette un generale aumento della qualità dell'immagine, mantenendo le dimensioni originali degli elementi presenti al suo interno.
Successivamente, al fine di ottenere un immagine binaria delle linee di demarcazione della corsia, l'intensità dei pixel dell'immagine viene confrontata con una soglia, in questo caso 180: se il valore del pixel è maggiore della soglia un operazione di saturazione porta il suo valore al massimo ammesso, ossia 255, in caso contrario al pixel viene assegnato il valore 0, come mostrato nell'equazione (3.1).

$$bin(x) = \begin{cases} 255 & se \quad x > 180 \\ 0 & altrimenti \end{cases} \tag{3.1}$$

Il risultato delle operazioni di estrazione del colore è mostrato nella Figura 3.3.

Immagine originale Immagine processata

Figura 3.3: Risultato dell'applicazione delle operazioni di estrazione del colore

3.2 Calcolo del riferimento

In questa fase l'immagine binarizzata viene prima trasformata al fine di ottenere una visione in prospettiva zenitale della corsia, e successivamente tramite una serie

di trasformazioni e sfruttando un algoritmo ad hoc viene generato un riferimento da inseguire e di conseguenza si determinano l'errore di allineamento alla corsia e quello laterale.

3.2.1 Passaggio alla prospettiva zenitale

Il passo successivo è l'eliminazione della prospettiva dall'immagine, questo viene fatto mediante un operazione IPM (Mappatura Prospettica Inversa), essa consiste nell'individuare i vertici di un rettangolo all'interno dell'immagine originale, la quale viene poi deformata per far coincidere tali punti con i vertici di un trapezio inverso (Fig. 3.4).

(a) Immagine originale (b) Immagine processata

Figura 3.4: Risultato della trasformazione IPM

I punti selezionati all'interno dell'immagine originale (Fig. 3.4 (a)) descrivono un rettangolo con base di dimensione corrispondente alla larghezza dell'immagine e altezza pari al 60% dell'immagine, posizionato nella parte bassa del riquadro, la scelta della posizione favorisce l'eliminazione delle parti di sfondo indesiderate e riduce di conseguenza la quantità di informazione da elaborare in seguito.

3.2.2 Trasformazione di Hough

La trasformazione di Hough[13] è uno strumento matematico fortemente utilizzato nel campo della visione artificiale, questo operatore matematico è infatti in grado di estrarre dallo spazio dell'immagine una rappresentazione matematica delle linee che la compongono.
In particolare le linee individuate vengono espresse in forma polare, come mostrato nell'equazione (3.2).

$$r = xcos(\alpha) + ysin(\alpha) \tag{3.2}$$

Tramite l'equazione (3.2), è possibile rappresentare una retta come un punto di coordinare (r, α), dove r rappresenta la distanza tra la retta e l'origine degli assi e α l'angolo sotteso tra l'asse x e la direzione perpendicolare alla retta stessa.
L'equazione (3.2) presenta, inoltre, il vantaggio di poter rappresentare anche linee verticali, a differenza della classica rappresentazione in forma esplicita $y = mx + q$.
Tale operazione è altamente costosa dal punto di vista computazionale, è infatti eseguita su una versione dell'immagine con dimensioni ridotte, in particolare si è scelto di dimezzare sia l'altezza che la larghezza, in modo tale da mantenere le proporzioni originali dell'immagine.
La funzione utilizzata nell'algoritmo implementato è `HoughLinesP`, un'implementazione con approccio probabilistico dell'algoritmo di Hough, la quale fornisce le linee che compongono l'immagine, espresse tramite due coppie di coordinate cartesiane, le quali rappresentano i punti estremi del segmento individuato. Il risultato della trasformata di Hough è rappresentato in Figura 3.5.

Figura 3.5: Rappresentazione delle linee trovate dalla trasformata di Hough, in particolare vengono evidenziate con colori distinti quelle che compongono la linea destra e sinistra della corsia.

3.2.3 Riconoscimento delle linee di corsia

Una volta ottenuto, mediante la trasformata di Hough, l'insieme `lines` delle linee, che compongono l'immagine, è necessario individuare con precisione le due linee

che delimitano la corsia.
L'algoritmo implementato esegue questa operazione in 2 passi:

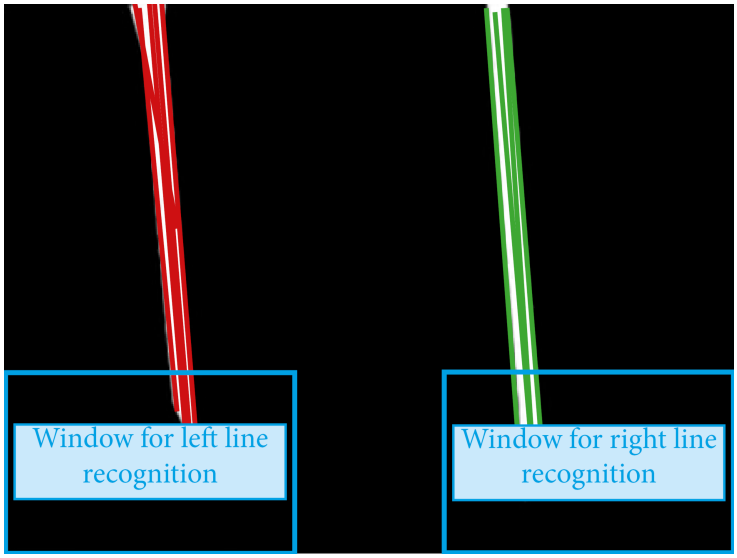


Figura 3.6: Finestra per il riconoscimento delle linee

1. Iterando sull'insieme `lines` vengono classificati i segmenti individuati come appartenenti alla linea destra o sinistra, in base alla posizione occupata all'interno dell'immagine (Fig. 3.6).
   Sulla base del risultato di questa classificazione si procede con il passaggio successivo ricercando una o entrambe le linee all'interno dell'immagine.

2. Tramite un algoritmo che implementa uno scorrimento a finestra vengono individuate le linee di corsia.
   L'immagine viene sezionata orizzontalmente in 8 finestre (da qui indicheremo con S l'insieme delle finestre), per ognuna delle quali viene calcolato l'istogramma (equazione 3.3), ottenendo, per ogni finestra, un vettore contenente il numero di pixel bianchi nella colonna corrispondente, le linee di corsia si troveranno dunque in corrispondenza dei massimi all'interno di tale vettore.

$$Hist_i(s) = \sum_{j \in s} s_{ji} \qquad i \in [0, col], \quad s \in S \tag{3.3}$$

   Iterando su S tale procedimento l'algoritmo è in grado di individuare le linee di corsia in 8 punti a diverse altezza all'interno dell'immagine.
   A partire dai punti individuati viene generato il riferimento di centro corsia come unione dei punti medi della posizione delle due linee individuate in ogni finestra.

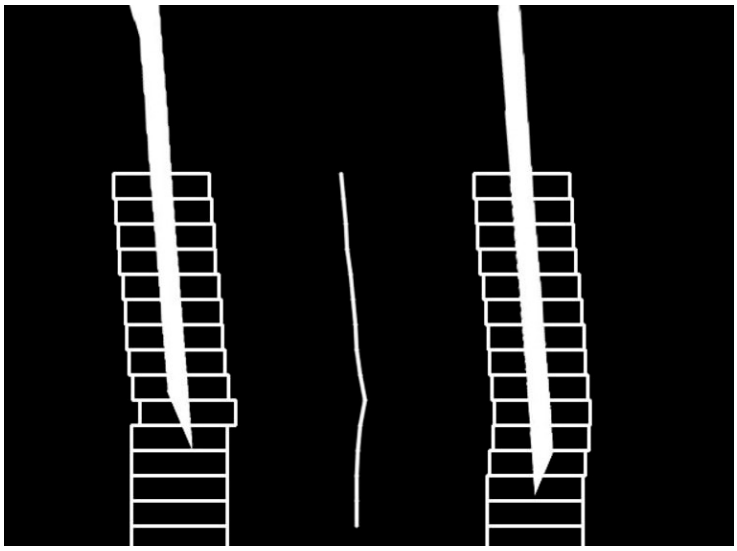


Figura 3.7: Risultato dell'algoritmo di scorrimento a finestra, considerando 15 finestre

3.2.4 Calcolo dell'errore di allineamento

Al fine di calcolare l'errore di allineamento ϵ (Fig. 3.1) è necessario individuare la direzione dell'asse longitudinale del veicolo e quella della traiettoria di riferimento. Facendo riferimento all'immagine in prospettiva zenitale (Fig. 3.4) l'asse longitudinale del veicolo coincide con la mezzeria dell'immagine.
All'interno dell'algoritmo proposto la traiettoria di riferimento, ottenuta tramite l'algoritmo di scorrimento a finestra descritto in precedenza, viene approssimata con una retta tale da minimizzare la distanza media dal riferimento stesso in ogni suo punto.
L'operazione di approssimazione viene fatta mediante una regressione lineare, il cui scopo è approssimare un insieme di n punti con una retta di equazione

$$y = mx + q \tag{3.4}$$

dove q rappresenta l'intercetta, ossia l'intersezione con l'asse y, calcolata secondo l'equazione (3.5)

$$q = \frac{[(\sum y)(\sum x^2) - (\sum x)(\sum xy)]}{[n(\sum x^2) - (\sum x)^2]} \tag{3.5}$$

e m il rapporto incrementale, ottenibile dall'equazione (3.6)

$$m = \frac{[n(\sum(xy) - (\sum x)(\sum y)]}{[n(\sum x^2) - (\sum x)^2]} \tag{3.6}$$

In particolare il rapporto incrementale m fornisce l'informazione riguardante la direzione della retta che approssima la traiettoria di riferimento.
La misura in *rad* dell'errore di allineamento può essere ottenuta come segue

$$\psi = arctan\left(\frac{1}{m}\right) \tag{3.7}$$

3.2.5 Calcolo dell'errore laterale

Nell'algoritmo proposto l'errore laterale ϵ (Fig. 3.1) viene espresso in funzione della dimensione della corsia, in particolare viene calcolato come rapporto tra la distanza del terzo punto della traiettoria di riferimento con la mezzeria dell'immagine, e la distanza tra le due linee di demarcazione della corsia (equazione 3.8).

$$\epsilon_p = \frac{r_x - \frac{1}{2}c}{l_{r,x} - l_{l,x}} \tag{3.8}$$

dove:

- ϵ_p è l'errore laterale espresso in funzione della larghezza della corsia
- r punto di rifermento, espresso in pixel
- c numero di colonne dell'immagine
- $l_{r,x} - l_{l,x}$ larghezza della corsia in pixel

A partire dalla dimensione della corsia, che, nel nostro caso, è nota e misura $35cm$, possiamo calcolare l'errore laterale in metri come segue:

$$\epsilon = \epsilon_p \cdot 0.35 \tag{3.9}$$

La scelta dell'utilizzare il terzo punto della traiettoria di riferimento è dettata da due fattori principali, in primo luogo si può osservare che nella parte d'immagine che ritrae la corsia nei punti più vicini all'asse anteriore del veicolo la precisione nel riconoscimento delle linee è minore, inoltre l'utilizzo di un punto leggermente più avanzato favorisce un'azione anticipata del controllore.

Capitolo 4

Mantenimento di corsia

Lo scopo di un algoritmo di mantenimento di corsia è quello di gestire la sterzata ed eventualmente accelerare o frenare il veicolo, con lo scopo di mantenere il veicolo all'interno della corsia designata.
Nella pratica l'obbiettivo è ottenere un controllore in grado di mantenere il sistema stabile nell'intorno dello stato

$$e = \begin{bmatrix} \psi \\ \epsilon \end{bmatrix} = \begin{bmatrix} 0^\circ \\ 0\text{m} \end{bmatrix} \tag{4.1}$$

dove ψ è l'errore di allineamento e ϵ l'errore laterale (Fig. 3.1).
Il riferimento (equazione 4.1) corrisponde alla condizione in cui il veicolo è perfettamente allineato e in posizione centrale in relazione alla corsia designata.
Nel campo della ricerca le strategie di controllo più comuni per il mantenimento di corsia sono:

- **Controllo Predittivo del Modello (MPC)** [14]: rappresenta una delle soluzioni più raffinate ed efficaci nel controllo di sistemi non lineari, descritti dal sistema di equazioni (4.2).
  Il principio su cui si basa un MPC è la determinazione dell'azione di controllo $u^*(t : t + T_P)$ tramite la predizione dell'evoluzione dell'uscita del sistema $\hat{y}(\tau, x(t), u(t : \tau))$ all'interno dell'intervallo temporale $[t : t + \tau]$.

  $$\begin{cases} \dot{x}(t) = f(x(t), u(t)) \\ y(t) = g(x(t), u(t)) \end{cases} \tag{4.2}$$

  Il valore di $u^*(t : t + T_P)$ viene predetto risolvendo un problema di ottimizzazione della funzione di costo $J(u(t : t + T_P))$ appositamente definita.
  Matematicamente, in un sistema di elaborazione a tempo discreto con tempo

di campionamento T_c, ciò si traduce nella risoluzione del seguente problema di ottimizzazione a ogni $t_k = k \cdot T_c$:

$$\begin{aligned} & u^*(t_k : t_k + T_P) = \arg\min_{\hat{u}(\cdot)} \; J\big(\hat{u}(t_k : t_k + T_P)\big) \\ & \text{s.t.} \\ & \dot{\hat{x}}(\tau) = f\big(\hat{x}(\tau), \hat{u}(\tau)\big), \qquad \hat{x}(t_k) = x(t_k), \\ & \hat{y}(\tau) = g\big(\hat{x}(\tau), \hat{u}(\tau)\big), \\ & \hat{x}(\tau) \in X_C, \quad \hat{y}(\tau) \in Y_C, \quad \hat{u}(\tau) \in U_C, \quad \tau \in [t_k,\, t_k + T_P]. \end{aligned} \tag{4.3}$$

I primi due vincoli imposti garantiscono la coerenza delle soluzioni trovate con le dinamiche del sistema controllato, mentre X_C, Y_C e U_C, sono insiemi che descrivono eventuali limitazioni aggiuntive imposte al controllore.
La funzione di costo J è una funzione quadratica dell'errore di inseguimento dell'uscita predetta $\tilde{y}_P$, e della variabile di controllo $\tilde{u}$ (equazione (4.4)), la quale, introducendo le matrici diagonali semidefinite positive Q, P ed R, definisce il comportamento e le prestazioni del controllore.

$$J\big(u(t : t + T_p)\big) = \int_t^{t+T_p} \Big(\|\tilde{y}_p(\tau)\|_Q^2 + \|\tilde{u}(\tau)\|_R^2 \Big)\, d\tau + \|\tilde{y}_p(t + T_p)\|_P^2 \,. \tag{4.4}$$

Il controllo predittivo del modello, nell'ambito della guida autonoma, offre la possibilità di gestire efficacemente anche limitazioni complesse, come la presenza di altri veicoli o di eventuali ostacoli in carreggiata.

- **Controllo PID** [15]: un controllore Proporzionale Integrale Derivativo è un controllo in retroazione, in grado di attuare un'azione correttiva sulla base di tre termini (Fig. 4.1):

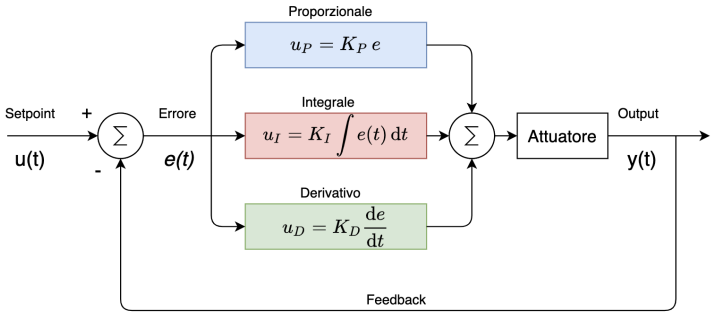


Figura 4.1: Schema di in controllore PID

a) **Termine proporzionale**: determinato dal prodotto del guadagno proporzionale K_p e dell'errore di inseguimento corrente, fornisce un'azione correttiva solo nel momento in cui è presente un errore.

b) **Termine integrale**: calcolato integrando nel tempo l'errore di inseguimento e pesando tale risultato con il fattore moltiplicativo k_i, tiene in considerazione gli errori passati, accumulandoli e accrescendo di conseguenza la risposta correttiva per errori persistenti nel tempo.

c) **Termine derivativo**: tramite il calcolo della derivata dell'errore si è in grado di stimarne il futuro andamento, e di conseguenza applicare un fattore di smorzamento al sistema.

Un controllore PID presenta tra i sui vantaggi la facilita e la velocità di progettazione, caratteristiche che lo hanno reso estremamente diffuso nell'ambito dell'automazione.
Lo svantaggio principale risiede nella necessità di integrare e differenziare l'errore di inseguimento nel tempo, un'operazione tipicamente onerosa.

- **Controllore Stanley** [16]: si tratta di una strategia di controllo sviluppata appositamente per l'inseguimento di una traiettoria.
  La legge di controllo applicata per il calcolo del valore di sterzo è riportata nell'equazione (4.5).

$$\delta = \psi + \arctan\left(\frac{k\epsilon}{v + k_v}\right) \tag{4.5}$$

dove:

- δ:valore di sterzo calcolato
- ψ: errore di allineamento
- ϵ: errore laterale
- v: velocità del veicolo
- k: parametro da tarare, tale parametro influisce sull'intensità con cui il controllore corregge l'errore laterale
- k_v: costante, tipicamente piccola, utile a evitare valori eccessivi di sterzo con velocità molto ridotte.

Il controllore Stanley , non richiedendo il calcolo del gradiente degli errori di inseguimento risulta computazionalmente molto efficiente, al pari di soluzioni come controllori con pianificazione del guadagno, ma con il vantaggio della gestione dell'angolo di sterzo in funzione della velocità.

4.1 Modello cinematico del veicolo

Nell'applicazione riguardante questa tesi, viste le velocità contenute, le dinamiche laterali del veicolo sono ridotte al minimo, e per questo risultano trascurabili, è quindi sufficiente nella progettazione del controllo fare riferimento a un modello cinematico.
Un modello cinematico è una semplificazione del reale moto di un oggetto, ottenuto tenendo in considerazione solo le relazioni cinematiche inerenti ad accelerazioni e velocità del baricentro del veicolo.
Il modelli cinematici tipicamente utilizzati nella modellazione di autoveicoli assimilano il veicolo a un corpo rigido con soli tre gradi di libertà nel sistema di riferimento dello stesso, rispetto ai sei gradi di libertà introdotti nei modelli dinamici più complessi (Fig. 4.2), in particolare i gradi di libertà concessi al veicolo nel modello cinematico in questione sono:

1. Spostamento lungo la coordinata x (asse longitudinale)

2. Spostamento lungo la coordinata y (asse trasversale)

3. Imbardata, ossia la rotazione rispetto all'asse z

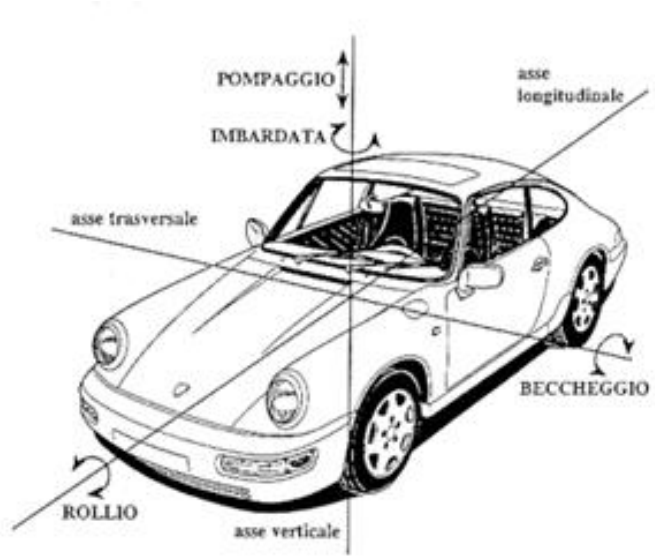


Figura 4.2: Sistema di riferimento del veicolo

Il modello può essere quindi descritto dal sistema di equazioni, espresse nel sistema di rifermento del veicolo (Fig. 4.2):

$$\begin{cases} a_x = \dot{v}_x - r v_y \\ a_y = \dot{v}_y + r u_x \end{cases} \tag{4.6}$$

dove:

- a_x e a_y: accelerazione longitudinale e trasversale del veicolo
- v_x e v_y: velocità longitudinale e trasversale del veicolo
- r: velocità angolare di imbardata

4.1.1 Modello cinematico monotraccia

Il modelli cinematico monotraccia [17], il quale sarà il riferimento per la progettazione dei controllori in questa tesi, semplifica il moto di un veicolo trascurando tutti i trasferimenti di carico e le dinamiche che ne conseguono, inoltre non vengono tenute in conto le differenti traiettorie seguite dai quattro pneumatici, collassandoli in due sole ruote, poste a una distanza pari all'interasse del veicolo l, lungo l'asse longitudinale. Il baricentro del veicolo viene poi posto in un punto arbitrario lungo tale asse, data la forte somiglianza con il mezzo questo modello viene spesso indicato con il nome di modello cinematico a bicicletta.
Tale semplificazione permette di rappresentare il moto di un veicolo con il seguente sistema equazioni differenziali a tempo continuo:

$$\begin{cases} \dot{x} = v \cdot cos(\psi + \beta) \\ \dot{y} = v \cdot sin(\psi + \beta) \\ \dot{\psi} = \frac{v}{l_r} \cdot sin(\beta) \end{cases} \tag{4.7}$$

$$\beta = arctan\left(\frac{l_r}{l_r + l_f} \cdot tan(\delta)\right) \tag{4.8}$$

dove:

- β: angolo tra il vettore velocità del centro di massa e l'asse longitudinale del veicolo.
- l_r e l_f: distanza dal baricentro dell'asse posteriore e anteriore.

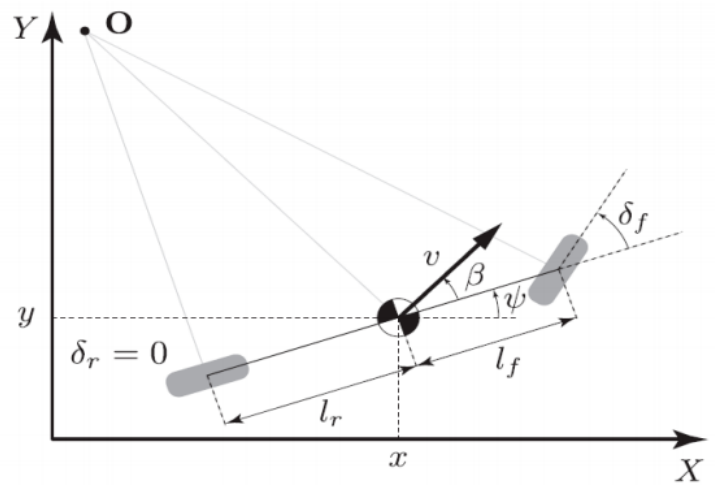


Figura 4.3: Modello cinematico monotraccia

4.2 Implementazione dei controllori

La scelta del controllore da utilizzare deve necessariamente tenere conto delle specifiche e delle potenzialità del modello in questione, in particolare data la limitata potenza di calcolo a disposizione non è possibile implementare un controllore allo stato dell'arte, come ad esempio un controllore MPC.
Per questo motivo durante questo lavoro sono stati utilizzati controllori più leggeri, ma comunque in grado di svolgere con discreta efficacia il compito assegnato.
In particolare, nella prima parte del lavoro è stato utilizzato il controllore con pianificazione del guadagno [8] (*gain scheduling*) sviluppato durante il lavoro dei tesisti precedenti, mentre in seguito è stato sviluppato un Controllore Stanley.
Prima di procedere con le prove e l'implementazione dei controllori, al software presente sono state apportate alcune modifiche con lo scopo di rendere la sua esecuzione a tempo discreto.
In primo luogo è stata implementata la classe `Semaphore`, utilizzata al fine di sincronizzare l'esecuzione dei thread adibiti alla cattura delle immagino, alla comunicazione con la scheda Nucleo e all'esecuzione del controllo. L'implementazione della classe `Semaphore` è mostrata nel Listato 4.1.
La classe `Semaphore` permette di interrompere l'esecuzione dei thread mediante la chiamata del metodo `wait()`, in attesa della chiamata del metodo `signal()` da parte di un gestore di interrupt, eseguito ciclicamente con periodo fisso T_c.
Il tempo di campionamento T_c viene tarato appositamente per ogni versione del software, in funzione del tempo di esecuzione del controllo.

```
class Semaphore{
    private:
             std::mutex mtx;
        int count;
        public:
        explicit Semaphore (int initialValue) : count(
           initialValue){}
        void wait(){
        while (true){
            mtx.lock();
            if (count > 0){
                count --;
                mtx.unlock();
                break;
            }
            mtx.unlock();
            std::this_thread::yield();
        }
        }
        void signal(){
                mtx.lock();
                count++;
                mtx.unlock();
        }
        void reset() {
                mtx.lock();
                count = 0;
                mtx.unlock();
        }
};
```

Listato 4.1: Implementazione della classe semaphore

4.2.1 Controllore con pianificazione del guadagno

La strategia di controllo implementata prevede un controllo diviso in due fasi, nella prima si determina un primo valore di sterzo δ_ψ, basato sul valore del rapporto incrementale m (equazione 3.6), a tale valore viene successivamente sommato un valore correttivo δ_ϵ calcolato sulla base dell'errore ϵ_p (equazione 3.8). L'angolo di sterzo viene poi calcolato come la somma dei due valori.

$$\delta = \delta_\psi + \delta_\epsilon \tag{4.9}$$

Controllo basato sull'errore di allineamento

La strategia di controllo consiste nell'applicare un comando proporzionale al valore dell'errore di allineamento, in particolare valori elevati del modulo m indicano piccoli angoli di errore e di conseguenza il controllore impone un angolo di sterzo basso, contrariamente, al decrescere del valore assoluto di m, viene aumentato l'angolo di sterzo.
La strategia implementata è riassunta all'interno della Tabella 4.1, la quale fornisce il modulo del valore di sterzo da applicare in relazione al modulo del rapporto incrementale m.

Tabella 4.1: Relazione tra $|m|$ e $|\delta_\psi|$

$\mathbf{\lvert m \rvert}$	$\lvert\delta_\psi\rvert$
$2 \leq \lvert m \rvert < 3.15$	20
$3.15 \leq \lvert m \rvert < 3.48$	17
$3.48 \leq \lvert m \rvert < 3.7$	14
$3.7 \leq \lvert m \rvert < 4.05$	11
$4.5 \leq \lvert m \rvert < 4.33$	9
$4.33 \leq \lvert m \rvert < 6$	6
$6 \leq \lvert m \rvert < 8.14$	5
$8.14 \leq \lvert m \rvert < 10$	4
$10 \leq \lvert m \rvert < 12$	3
$12 \leq \lvert m \rvert < 14$	2
$\lvert m \rvert \geq 16$	1

Una volta ottenuto il modulo, il segno dell'angolo di sterzo viene imposto concorde con il rapporto incrementale m (Equazione 4.10).

$$\delta_\psi = |\delta_\psi| \cdot sgn(m) \tag{4.10}$$

Controllo basato sull'errore laterale

Una volta calcolato δ_ψ, il fattore correttivo basato sull'errore laterale δ_ϵ viene calcolato applicando la Tabella 4.2.
Al fine di mantenere la stabilità del controllo si è data la priorità alla correzione dell'errore di allineamento, in quanto più critico per il corretto mantenimento di corsia, per questo motivo la correzione dell'errore laterale viene attuata solo se il valore di sterzo δ_ψ è nell'intervallo [-10°, 10°], in più tale correzione viene applicata solo se concorde al valore di sterzo δ_ψ in modo da non inficiare la correzione dell'errore angolare.

Tabella 4.2: Relazione tra δ_ψ, ϵ e δ_ϵ

$\delta_\psi[deg]$	$\epsilon[m]$	$\delta_\epsilon[deg]$
$-10 \leq \delta_\psi \leq 0$	$-0.10 \leq \epsilon < -0.05$	-1
	$-0.20 \leq \epsilon < -0.10$	-1.5
	$-0.30 \leq \epsilon < -0.20$	-2.5
	$-0.40 \leq \epsilon < -0.30$	-3.5
	$-0.50 \leq \epsilon < -0.40$	-4.5
	$\epsilon < -0.50$	-5
$0 \leq \delta_\psi \leq 10$	$0.05 \leq \epsilon < 0.10$	1
	$0.10 \leq \epsilon < 0.20$	1
	$0.20 \leq \epsilon < 0.30$	2
	$0.30 \geq \epsilon < 0.40$	2
	$0.40 \geq \epsilon < 0.50$	3
	$\epsilon > 0.50$	3

Controllo della velocità

Nella soluzione implementata la velocità viene mantenuta costante a un determinato valore durante le fasi di rettilineo, la velocità di riferimento in rettilineo è di circa `RefSpeed`=0,08m/s , nel momento in cui è richiesto un valore di sterzo maggiore la velocità viene diminuita per concedere più tempo al controllore di agire e correggere l'errore di inseguimento.
L'algoritmo per il controllo della velocità è riassunto nella Tabella 4.3.

Tabella 4.3: Relazione tra l'angolo di sterzo δ e la velocità v

$\lvert\delta\rvert$[**deg**]	$v[m/s]$
$0 \leq \lvert\delta\rvert < 2$	`RefSpeed`
$2 \leq \lvert\delta\rvert < 3$	`RefSpeed`·0.9
$3 \leq \lvert\delta\rvert < 4.5$	`RefSpeed`·0.85
$4.5 \leq \lvert\delta\rvert < 6$	`RefSpeed`·0.75
$6 \leq \lvert\delta\rvert$	0.068

4.2.2 Controllore Stanley

Parte del lavoro svolto durante la stesura di questa tesi è stato finalizzato allo sviluppo di in controllore alternativo all'originale proposto negli anni passati.
Tra le possibili scelte, data la potenza limitata, la scelta è ricaduta sul Controllore Stanley [16] [18], una soluzione sviluppata appositamente per l'inseguimento di una traiettoria e ben nota nel campo della ricerca sulla guida autonoma per le sue

grandi potenzialità.

Modello fisico di riferimento

Il modello cinematico adottato è quello monotraccia descritto nella sezione 4.1.1, con una semplificazione, il baricentro del veicolo viene posizionato sull'asse anteriore (Fig. 4.4). Questa semplificazione porta a far coincidere l'angolo di sterzo δ con l'angolo tra l'asse del veicolo e la velocità del baricentro β.
Il modello risulta quindi completamente descritto dal seguente sistema di equazioni:

$$\begin{cases} \dot{x} = v \cdot cos(\psi + \delta) \\ \dot{y} = v \cdot sin(\psi + \delta) \\ \dot{\psi} = \frac{v}{l_r} \cdot sin(\delta) \end{cases} \tag{4.11}$$

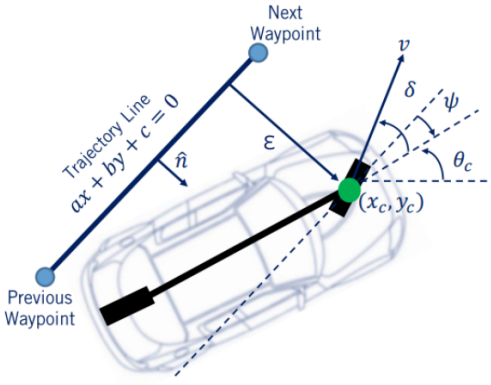


Figura 4.4: Modello monotraccia utilizzato per la progettazione del controllore stanley

Strategia di controllo

Il controllore Stanley (equazione 4.12) segue una strategia di controllo differente dal controllore con pianificazione del guadagno visto in precedenza, nonostante ne condivida alcune caratteristiche funzionali.

$$\delta = \psi + \arctan\left(\frac{k\epsilon}{v + k_v}\right), \qquad \delta \in [-\delta_{max}, \delta_{max}] \tag{4.12}$$

Dall'equazione (4.12) si può notare che similmente al controllore presentato in precedenza l'angolo di sterzo viene calcolato come la somma di due contributi, i quali sono però indipendenti nel controllore Stanley.
In particolare il primo contributo

$$\delta_\psi = \psi \tag{4.13}$$

Ha lo scopo di correggere l'errore di allineamento, con risultati simili a quelli ottenuti dal controllore con pianificazione del guadagno (PG).
Un approccio diverso viene invece utilizzato per la correzione dell'errore laterale (equazione 4.14), infatti tale correzione avviene indipendentemente dal valore di δ_ψ.

$$\delta_\epsilon = \arctan\left(\frac{k\epsilon}{v + k_v}\right) \tag{4.14}$$

La differenza fondamentale tra i due controllori presi in esame, a favore del controllore Stanley risiede nella gestione dello sterzo in funzione della velocità, infatti nella correzione dell'errore laterale il controllore Stanley tiene in conto la velocità, fornendo un azione più dolce tanto maggiore è la velocità del veicolo.
Questo permette di non gestire la velocità in funzione dello sterzo bensì il contrario, l'azione sullo sterzo si adatta alla velocità favorendo una maggiore stabilità permettendo velocità più sostenute.
L'implementazione del controllo è mostrata in Listato 4.2.

```
1 float k = 0.1;
2 float kv = 0.001;
3
4 steering_value = heading_err + std::atan2((k*lateral_error) /
     (refSpeed + kv));
```

Listato 4.2: Calcolo dell'angolo di sterzo nel controllore Stanley

Capitolo 5

Emulazione tramite rete neurale

Lo scopo di questo lavoro è valutare la possibilità di sostituire gli algoritmi di riconoscimento e mantenimento della corsia presentati nei capitoli 3 e 4 con una rete neurali in grado di emularne il comportamento.
In questo contesto il vantaggio principale offerto da una rete neurale, risiede nel suo costo computazionale costante. Essa infatti, a differenza algoritmi presentati in precedenza, nella fase di inferenza ha un costo computazionale costante e indipendente dal valore dei parametri che ne definiscono il comportamento.
Tale caratteristica può permettere l'esecuzione di controllori complessi anche su hardware limitati.
Nell'ambito dell'automazione e della guida autonoma, tra le tecniche più utilizzate figurano:

- **Apprendimento per rinforzo** (*Reinforcement Learning, RL*) [19]: è un paradigma nel campo delle reti neurali, nel quale il modello interagisce con un ambiente dinamico al fine di sviluppare autonomamente una legge di controllo ottimale.
  Durante la fase di apprendimento l'agente, osservando l'ambiente esterno, assume una serie di decisioni, e riceve una ricompensa/penalità sulla base della bontà della decisione presa (Fig. 5.1), adattando la propria politica comportamentale al fine di massimizzare la ricompensa cumulata nel tempo.
  Il comportamento della rete neurale può quindi essere condotto verso le prestazioni desiderate adottando diverse strategie di ricompensa.
  Questo paradigma è classificato nelle tecniche di addestramento **non supervisionato**, dove è la rete stessa a dover sviluppare una legge di controllo in grado di massimizzare la funzione di ricompensa.
  Tale paradigma è tuttavia molto dispendioso e difficile da controllare, dal momento che viene richiesta la simulazione anche fisica dell'ambiente oltre all'inferenza e all'adattamento dei parametri della rete neurale durante la fase di addestramento.

Nel campo della guida autonoma è possibile allenare una rete neurale a eseguire compiti come il mantenimento di corsia, tuttavia l'approccio esplorativo della rete, soprattutto nelle fasi iniziali, ne impedisce l'allenamento a bordo dei un modello fisico, anche se in scala, del veicolo, richiedendo l'utilizzo di software di simulazione costosi e complessi.

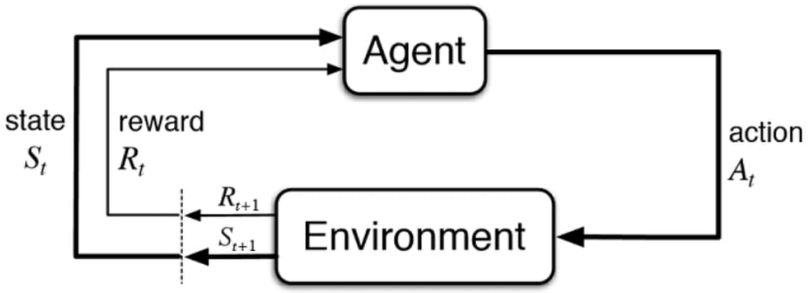


Figura 5.1: Schema generale del processo di apprendimento per rinforzo

- **Allenamento per imitazione** (*Imitation Learning, IL*) [20]: è un paradigma di allenamento supervisionato, il quale prevede che la rete neurale venga allenata a partire da un dataset, che descrive il comportamento di un agente esperto, sia esso umano o un controllore precedentemente implementato e testato.
  All'interno del datatset sono raccolti una serie di esempi dai quali la rete neurale deve estrapolare la relazione che lega gli stati osservati e la corretta azione da compiere.
  Formalmente lo scopo della rete neurale, in fase di allenamento, è quello di apprendere una legge di controllo $\pi(a|s)$ che approssimi nel modo migliore possibile la legge di controllo dell'agente esperto, come mostrato nell'equazione (5.1).

$$\pi(a|s) \simeq \pi^*(a|s) \tag{5.1}$$

  Dove:

  - s: stato dell'ambiente
  - a: azione calcolata
  - $\pi^*(a|s)$: legge di controllo dell'agente esperto

I vantaggi dell'apprendimento per imitazione si riflettono in una fase di allenamento più rapida ed economica, e in un comportamento più prevedibile del modello allenato, in quanto esso riflette il comportamento dell'agente esperto utilizzato durante la creazione del dataset, dal quale il modello eredita, tuttavia, anche le limitazioni e gli errori.
Nel campo della ricerca sulla guida autonoma [21], è possibile allenare reti neurali al fine di emulare sia guidatori umani che controllori automatici.

In questo lavoro si è scelto di implementare una rete neurale in grado di eseguire simultaneamente il riconoscimento e il mantenimento di corsia, mediante apprendimento per imitazione dei controllori presentati nel capitolo 4.
La scelta di seguire l'approccio dell'apprendimento per imitazione è dettata dalla maggiore velocità e stabilità durante la fase di allenamento della rete neurale.

5.1 Datatset

Durante questo lavoro si è scelto di generare il dataset interamente a bordo del modello in scala, non ricorrendo alla generazione di immagini sintetiche.
Al fine di ottenere un dataset rappresentativo della legge di controllo applicata dai controllori per il mantenimento di corsia implementati nei capitoli 4.2.1 e 4.2.2, sono state apportate una serie di modifiche al software sviluppato in precedenza.
Nello specifico al codice è stato aggiunto un sistema di acquisizione sistematica dei dati su file, che prevede la creazione di un apposito file in formato CSV relativo ad ogni prova svolta con il veicolo (Listato 5.1).
All'interno del file `log.csv`, relativo ad ogni prova del modello, vengono riportati

```
1  // Creazione e scrittura del file di log
2  string log_path = "../log.csv";
3  ofsteram logFile(log_path);
4  logFile << "iteration,steeringvalue,speedvalue,slope,offset";
5  ...
6  while(true){ // Ciclo di elaborazione e controllo
7      ...
8      // Scrittura dei dati sul file di log
9      logFile << iter_cont << "," << steeringvalue << "," <<
           speedvalue << "," << m << "," << center_lane[3].x-
           birdsEyeView.cols / 2 << endl;
10     ...
11 }
```

Listato 5.1: Generazione del file CSV

i valori ottenuti dal ciclo di riconoscimento della corsia: il rapporto incrementale m e l'errore laterale in rapporto alla dimensione della corsia ϵ_p; oltre alle uscite del controllore: angolo di sterzo δ e la velocità v, come mostrato nel Listato 5.1.
Ai dati contenuti nel file `data.csv` viene associato un indice incrementale, usato come identificativo per abbinare i valori di uscita del controllo con la relativa immagine catturata dalla fotocamera, le quali vengono salvate durante l'esecuzione in formato PNG.

Figura 5.2: Esempio di immagine contenuta nel dataset

La procedura di raccolta dati è stata eseguita facendo percorrere al veicolo in scala il tracciato guidato autonomamente da uno dei due controllori.
Per ogni immagine acquisita è stato registrato il corrispondente comando di sterzo emesso dal controllore.
Poiché questo metodo avrebbe prodotto due dataset distinti, ciascuno composto da immagini differenti etichettate solo con uno dei due controllori, è stata introdotta una fase di elaborazione offline, eseguita per comodità a bordo della scheda RaspBerry Pi5.
Durante l'elaborazione offline, ogni immagine registrata è stata processata anche mediante il controllore non utilizzato durante la sua generazione, ottenendo in questo modo, per lo stesso frame, entrambe le etichette di sterzo.
Questo processo consente di massimizzare la quantità di dati disponibili e garantisce la comparabilità diretta tra i due controllori sulle medesime condizioni visive.

Il dataset ottenuto mediante questo processo risulta quindi composto di 1800 immagini, etichettate con i comandi di sterzo δ_{ST} e δ_{PG}, rispettivamente ottenuti mediante l'elaborazione delle immagini dal controllore Stanley e dal controllore con pianificazione del guadagno, tali informazioni sono immagazzinate all'interno del file `dataset.csv`.
In previsione della fase di allenamento il dataset è stato suddiviso in modo casuale in tre sottoinsiemi:

- **Insieme di addestramento**: utilizzato nella fase di allenamento della rete, comprende circa l'80% dei dati raccolti.
- **Insieme di validazione**: utilizzato per valutare l'andamento dell'apprendimento della rete neurale durante l'allenamento, è composto dal 10% dei dati.
- **Insieme di test**: riservato alla fase finale, nella quale le capacità sviluppate dalle reti neurali vengono valutate, comprende il restante 10% dei dati.

L'utilizzo assegnato ai singoli elementi del dataset è riportato nel campo "utilizzo" del file `dataset.csv`.
Un esempio del contenuto del file `datset.csv` e della relativa immagine sono visionabili nella Figura 5.2 e nella Tabella 5.1.

Tabella 5.1: Contenuto del file `dataset.csv` relativo alla Figura 5.2

id	$\boldsymbol{\delta_{gs}}$	$\boldsymbol{\delta_{stanley}}$	$\boldsymbol{\psi}$	$\boldsymbol{\epsilon}$	**utilizzo**
47	-7.5	-14.2016	-12.355	-0.005689	training

5.2 Architetture delle reti neurali utilizzate

La scelta dell'architettura della rete neurale gioca un ruolo di fondamentale importanza, la rete neurale utilizzata deve infatti rispettare due criteri fondamentali:

a) **Allenamento stabile**: la rete neurale deve avere caratteristiche che ne rendano possibile l'allenamento in tempi contenuti e con dataset di dimensioni limitate.
Per questo motivo vengono preferite reti delle quali siano reperibili versioni pre-allenate su dataset generici, quale ad esempio ImageNet, in modo tale da poterne sfruttare le capacità di generalizzazione sviluppate.

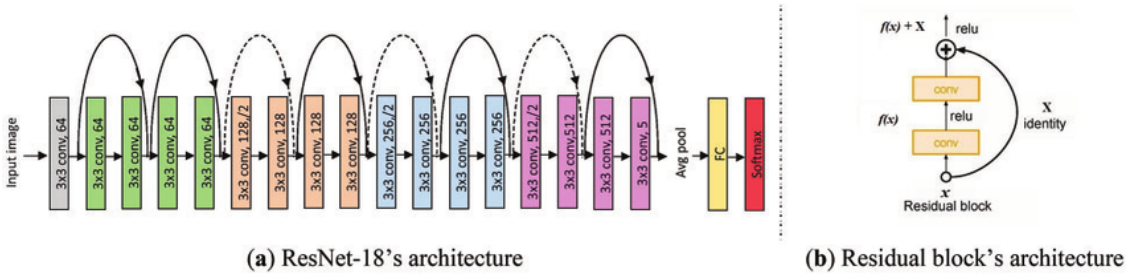
b) **Efficienza in fase di inferenza**: data la limitata potenza computazionale della scheda a disposizione si prediligono modelli efficienti e non eccessivamente profondi.

Non sono ritenute valide architetture che richiedono tempi di inferenza superiori a $200ms$ sulla scheda Raspberry Pi5.

Tra le opzioni prese in considerazione due architetture sono state scelte per questo lavoro, **ResNet18** e **MobileNet V3 - Small**, entrambe queste reti sono rese disponibili all'interno del framework PyTorch, il quale ne mette inoltre a disposizione una versione pre-allenata sul dataset ImageNet [22] [23].

5.2.1 ResNet18

ResNet18 [24] è un'architettura convoluzionale composta da 18 livelli con pesi, appartenente alla famiglia delle Residual Networks.
La sua caratteristica principale è l'utilizzo dei blocchi residuali, moduli che includono una scorciatoia (*skip connection*) in grado di facilitare la propagazione del gradiente e di stabilizzare l'addestramento.
Ogni blocco è costituito da due convoluzioni 3×3 seguite da Batch Normalization e attivazione ReLU (Fig. 5.3).
La presenza delle *skip connection* permette alla rete di apprendere trasformazioni residue, rendendo possibile l'addestramento di architetture più profonde e migliorando la capacità di generalizzazione.
ResNet18 offre un buon compromesso tra capacità rappresentativa e complessità computazionale ed è spesso impiegata come estrattore di caratteristiche (backbone) in modelli di visione basati sul deep learning.



(a) ResNet-18's architecture **(b)** Residual block's architecture

Figura 5.3: Architettura di ResNet18

Dal momento che la rete neurale ResNet18 è stata sviluppata per il compito della classificazione di immagini, sono necessari alcuni adattamenti per permetterne l'utilizzo in contesti regressivi, quale è l'emulazione dei controllori in questione.
In particolare la "testa" classificatrice, composta da un livello completamente connesso (FC) seguito da un blocco softmax (Fig. 5.3 (a)), è stata completamente sostituita con un livello completamente connesso che presenta un'uscita di dimensione 1, come mostrato nel Listato 5.2.

La rete neurale così ottenuta presenta una struttura portante convoluzionale pre-addestrata su ImageNet, impiegata come estrattore di caratteristiche dall'immagine in ingresso, e una testa regressiva in grado di fornire il valore di angolo di sterzo da attuare.

```
model = models.resnet18(pretrained=True)
model.fc = nn.Linear(model.fc.in_features, 1)
```

Listato 5.2: Definizione della rete ResNet

5.2.2 MobileNet V3-Small

MobileNet è una famiglia di reti convoluzionali sviluppate da Google appositamente per garantire efficienza e stabilità su dispositivi mobili ed embedded.
Per questo progetto si è scelto di utilizzare la terza iterazione di questa famiglia di reti, MobileNet V3 [25], la quale è stata resa disponibile in due versioni che differiscono per dimensioni e capacità espressiva:

a) **MobileNet V3 - Small**: rete neurale ottimizzata per dispositivi con risorse limitate

b) **MobileNet V3 - Large**: rete neurale di dimensioni maggiori, pensata per massimizzare l'accuratezza su datatset complessi

In questo lavoro, data la limitata capacità di calcolo a disposizione si è optato per la versione **MobileNet V3 - Small**.
L'architettura di MobileNet V3 deve la sua efficienza ad alcune soluzioni tecniche innovative implementate all'interno della sua struttura portante (backbone):

- **deepwise convolution** [26]: rappresenta un'alternativa notevolmente più efficiente rispetto alle convoluzioni tradizionali.
- **Inverted residual block** con **linear bottleneck** [27]: introdotte con MobileNet V2, permettono di combinare alta capacità espressiva e bassi costi di esecuzione.
  L'architettura, mostrata in Figura 5.4, prevede un'iniziale espansione dei canali, seguita da una deepwise convolution e da una finale compressione dei canali, quest'ultima garantisce che ingresso e uscita del blocco mantengano le stesse dimensioni, permettendo l'implementazione delle *skip connection* caratteristiche di ResNet.
- Funzioni di attivazione ottimizzate per le architetture mobile, **hard-swish** e **hard-sigmoid** (Fig. 5.5).

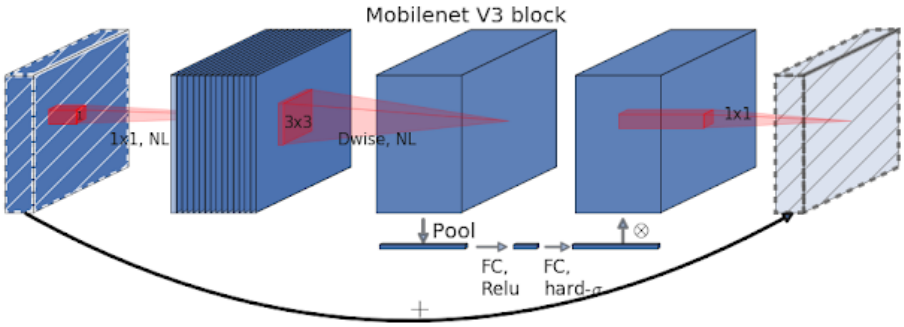


Figura 5.4: MobileNet V3 Inverted residual block

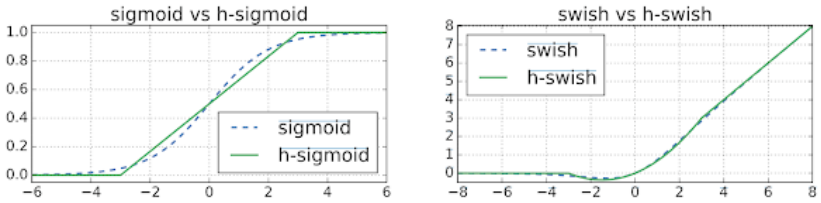


Figura 5.5: Funzioni Hard-Swish e Hard-Sigmoid

La rete neurale utilizzata, descritta dalla classe `LaneKeepingMobileNetV3` (Listato 5.3), fa uso della struttura portante di MobileNetV3-Small al fine di estrarre le caratteristiche dall'immagine in ingresso, e di una testa regressiva con il compito di interpretare le caratteristiche estratte dalla rete convoluzionale fornendo il relativo comando di sterzo.
La testa regressiva si compone di un livello lineare con 128 neuroni, seguito da una funzione di attivazione `ReLu` e da un secondo livello lineare con uscita singola.
Questa configurazione rappresenta un compromesso efficace: è sufficientemente espressiva da modellare la relazione tra immagine e angolo di sterzo, ma al tempo stesso leggera da addestrare e adatta a garantire un'inferenza rapida.

5.3 Allenamento delle reti neurali

5.3.1 Ampliamento artificiale del dataset

Data la dimensione ridotta del dataset raccolto sono state sperimentate diverse tecniche con il fine di ottenere una quantità di dati maggiore.
Le immagini contenute all'interno del dataset, vengono quindi sottoposte a una catena di trasformazioni [28], tra queste vi è un primo gruppo di trasformazioni preliminari, applicate online a tutte le immagini che compongono il dataset:

```python
class LaneKeepingMobileNetV3(nn.Module):
    def __init__(self, pretrained=True):
        super().__init__()

        # Backbone MobileNetV3-Small
        self.backbone = models.mobilenet_v3_small(
            weights=models.MobileNet_V3_Small_Weights.DEFAULT
                if pretrained else None
        )

        # Numero di feature in uscita
        in_features = self.backbone.classifier[0].in_features

        # Rimuovo il classifier
        self.backbone.classifier = nn.Identity()

        # Sostituisco la testa classificatrice con una
        # piccola testa FC con uscita singola
        self.head = nn.Sequential(
            nn.Linear(in_features, 128),
            nn.ReLU(inplace=True),
            nn.Linear(128, 1)
        )

    def forward(self, x):
        feat = self.backbone(x)    # (B, in_features)
        out = self.head(feat)      # (B, 1)
        return out
```

Listato 5.3: Definizione della rete basata su MobileNet-Small

- **Rimozione dello sfondo**: le immagini, inizialmente di dimensione 640x400 pixel, vengono processate in modo tale da escludere la porzione superiore dell'immagine, mediante la trasformazione descritta dalla classe `BottomCrop`, riportata nel Listato 5.4, ottenendo in questo modo un'immagine di dimensione 640x300 pixel.
  Tale operazione ha lo scopo, data la scarsa varietà del dataset, di rimuovere porzioni dell'immagine non significative in relazione al compito designato, evitando adattamenti basati su informazioni irrilevanti da parte della rete neurale.

- **Variazione della luminosità**: con lo scopo di ottenere una rete robusta in

```
class BottomCrop:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def __call__(self, img):
        w, h = img.size
        left = (w - self.width) // 2
        top = h - self.height
        right = left + self.width
        bottom = h
        return img.crop((left, top, right, bottom))
```

Listato 5.4: Classe BottomCrop

ogni condizione di luce, le immagini vengono elaborate in modo da modificarne la luminosità.
La trasformazione applicata è descritta dalla classe `RandomLight`, la cui implementazione è riportata nel Listato 5.5.
La classe `RandomLight` è definita da tre parametri, `high` e `low`, i quali rappresentano rispettivamente il massimo e il minimo valore di luminosità applicabili all'immagine, e la probabilità `p` con la quale la trasformazione viene applicata all'immagine.

```
class RandomLight:
  def __init__(self, low, high, p):
    self.low = low
    self.high = high
    self.p = p

  def __call__(self, img):
    if random.random() < self.p:
        factor = random.uniform(self.low, 0.9) if random.
           random() < 0.5 else random.uniform(1.1, self.
           high)
        enhancer = ImageEnhance.Brightness(img)
        img = enhancer.enhance(factor)
    return img
```

Listato 5.5: Classe RandomLight

- **Ridimensionamento**: le immagini vengono in fine ridimensionate, portandole alla dimensione 224x224, in modo tale da renderle compatibili con l'ingresso delle due reti neurali utilizzate.
  Tale operazione

Un esempio del risultato di questa sequenza di trasformazioni è mostrato nella Figura 5.6.

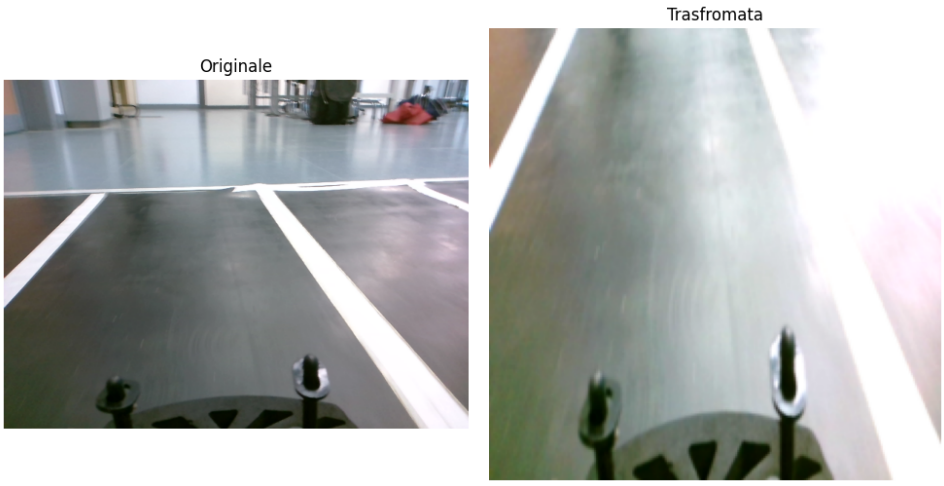


Figura 5.6: Esempio di applicazione delle trasformazioni preliminari

Al fine di ampliare il dataset sono state ulteriormente applicate, solo sulle immagini destinate al training; le seguenti trasformazioni:

- **Capovolgimento orizzontale**: al fine di risolvere la differenza tra il numero di immagini relative a curve verso destra e verso sinistra, dovuta alla conformazione del tracciato su cui è stato generato il dataset, durante la fase di allenamento, le immagini, fornite alla rete,vengono capovolte orizzontalmente con una probabilità del 50%.
  Al fine di mantenere la correttezza del dato fornito alla rete, se l'immagine viene capovolta orizzontalmente, viene invertito il segno del valore di angolo di sterzo fornito come etichetta alla rete.

- **Anticipazione del valore di sterzo**: tale tecnica, sperimentata durante l'allenamento, prevede di fornire alla rete il valore del comando di sterzo relativo a un istante futuro, nel caso in questione 3 iterazioni successive.
  Tale operazione, eseguita con probabilità del 50%, incentiva un'azione anticipata della rete. In questo modo è possibile compensare la latenza presente sia nell'esecuzione del controllore, utilizzato durante la generazione del dataset, che durante la fase di inferenza della rete neurale stessa, permettendo di

ottenere prestazioni di guida per alcuni aspetti migliori rispetto al controllore originale.

5.3.2 Allenamento del modello

Funzione di costo

Per applicazioni regressive le funzioni di costo più diffuse nell'ambito della ricerca sono:

- **Errore Quadratico Medio** (*Mean Squared Error, MSE*): rappresenta una delle funzioni di perdita più utilizzate nell'addestramento di reti regressive. Tale metrica esprime la discrepanza tra i valori obbiettivo y e quelli stimati dal modello $\hat{y}$, penalizzando in modo progressivo errori di maggiore entità. Matematicamente la funzione MSE viene definita su N campioni come:

$$MSE(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{5.2}$$

  Il principale svantaggio della funzione di perdita MSE è la sensibilità alla presenza di valori anomali all'interno del dataset.

- **Errore Assoluto Medio** (*Mean Absolute Error, MAE*): funzione di perdita che penalizza linearmente la discrepanza tra i valori obbiettivo y e quelli stimati dal modello $\hat{y}$, come mostrato nell'equazione (5.3).

$$MAE(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{5.3}$$

  Al contrario di MSE, la funzione di perdita MAE è meno sensibile ai valori anomali all'interno del dataset, risultando tuttavia meno incisiva quando, in fasi avanzate dell'addestramento, la rete commette errori di piccole dimensioni.

- **Huber Loss**[29]: la funzione di Huber rappresenta un buon compromesso tra la sensibilità di MSE e la robustezza di MAE nei confronti delle anomalie all'interno del dataset.
  La funzione di Huber è definita come:

$$L_\delta(y, \hat{y}) = \begin{cases} \frac{1}{N}(y_i - \hat{y}_i)^2 & se \quad |y - \hat{y}| < \delta \\ \frac{1}{N}|y_i - \hat{y}_i| - \frac{1}{2}\delta^2 & altrimenti \end{cases} \tag{5.4}$$

  dove il parametro δ rappresenta la soglia che divide il comportamento quadratico da quello lineare, come visibile nella Figura 5.7.

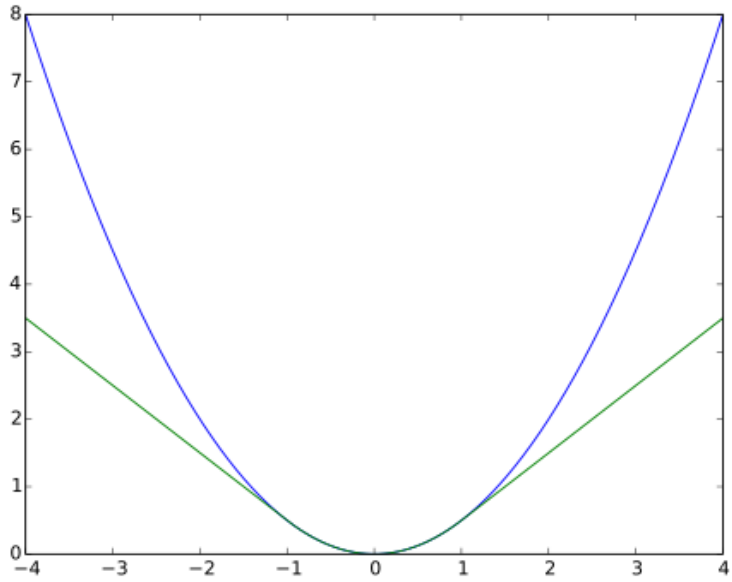


Figura 5.7: Confronto tra funzione MSE (blu) e funzione di Huber con parametro $\delta = 1$ (verde)

Tra le opzioni sopra citate la scelta per la funzione di perdita, in questo lavoro, è ricaduta sulla funzione di Huber [30], tale scelta è dovuta alla presenza, all'interno del dataset, di valori anomali in numero non trascurabile rispetto alla dimensione del dataset stesso.
La soglia che definisce il passaggio tra il comportamento lineare e quadratico $\delta = 2$ è stata scelta a partire da alcune considerazioni sul compito specifico del mantenimento di corsia e sulle prestazioni osservate durante la prova dei controllori tradizionali.
Vengono infatti classificati come "normali" errori contenuti nell'intervallo [-2°, +2°], intervallo all'interno del quale si vuole che la rete apprenda con la massima sensibilità, valori al di fuori di questo intervallo vengono invece penalizzati con minore intensità, in quanto vengono potenzialmente attribuiti ad anomalie all'interno del dataset.

Procedura di Allenamento del modello

L'addestramento della rete neurale è stato eseguito seguendo una procedura iterativa basata sulla discesa del gradiente, utilizzando l'ottimizzatore Adam e la funzione di perdita di Huber, descritta nell'equazione (5.4).
L'addestramento è stato condotto per un numero prefissato di epoche, monitorando

l'andamento sia della loss di addestramento sia della loss di validazione al fine di monitorare l'andamento dell'allenamento.
L'implementazione della funzione utilizzata per l'addestramento dei modelli neurali è mostrata del Listato 5.6.
Al fine di ottimizzare l'addestramento, sono state fatte diverse scelte progettuali:

```
def train_model(model, trainloader, valloader, epochs, lr,
    loss_function, device):
     model.to(device)
     optimizer = optim.Adam(model.parameters(), lr=lr)
     criterion = loss_function
     train_losses = []
     val_losses = []
     for epoch in range(epochs):
         model.train()
         total_loss = 0
         for batch_idx,(img, target) in enumerate(trainloader):
             img, target = img.to(device), target.to(device)
             optimizer.zero_grad()
             output = model(img)
             loss = criterion(output, target)
             loss.backward()
             optimizer.step()
             total_loss += loss.item()
         t_loss = total_loss / len(trainloader)
         train_losses.append(t_loss)
         model.eval()
         val_loss = 0.0
         with torch.no_grad():
             for img, target in valloader:
                 img, target = img.to(device), target.to(device
                     )
                 outputs = model(img)
                 loss = criterion(outputs, target)
                 val_loss += loss.item()
         val_loss /= len(valloader)
         val_losses.append(val_loss)
         if (epoch+1)%10 == 0 and epoch != 0:
           if val_losses[-1] >= val_losses[-9] * 0.9:
             lr = lr/10
     return (train_losses, val_losses)
```

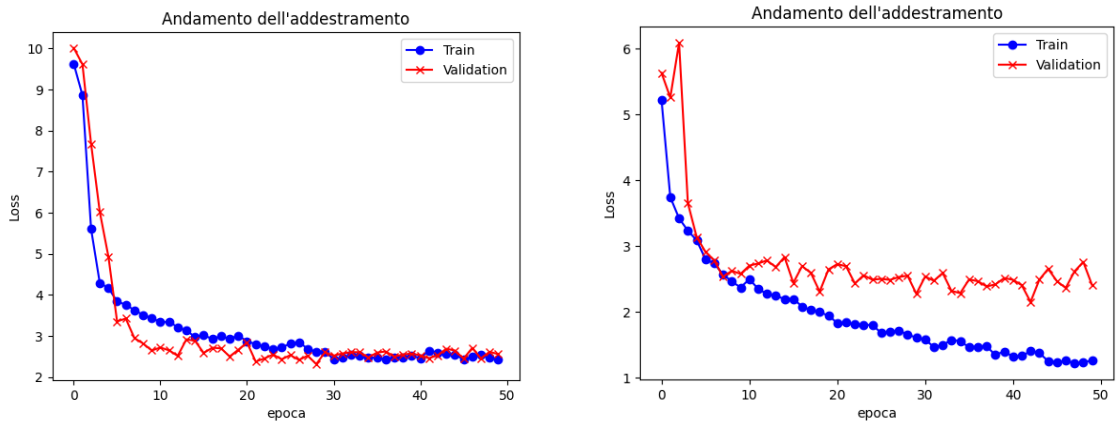
Listato 5.6: Funzione di training utilizzata

- **Addestramento in mini-batch**: il dataloader deputato alla fornitura dei dati di allenamento è stato diviso in mini-batch di dimensione 16.
  Tale scelta si è rivelata un buon compromesso tra la velocità di convergenza garantita da batch di piccole dimensioni, e la stabilità tipica di batch di grandi dimensioni.

- **Tasso di apprendimento** (*learning rate, lr*): la gestione del tasso di apprendimento è stata automatizzata all'interno della funzione di allenamento (Listato 5.6 linea 30-32).
  All'inizio dell'addestramento il valore del tasso di apprendimento è impostato a 10^{-3} in caso di modelli non ancora allenati, e 10^{-5} nel caso si rendano necessari allenamenti successivi.
  A intervalli regolari di 10 epoche viene verificato che il modello abbia mostrato un miglioramento pari ad almeno il 10% nelle ultime 10 epoche, in caso contrario il valore `lr` viene diminuito di un ordine di grandezza.

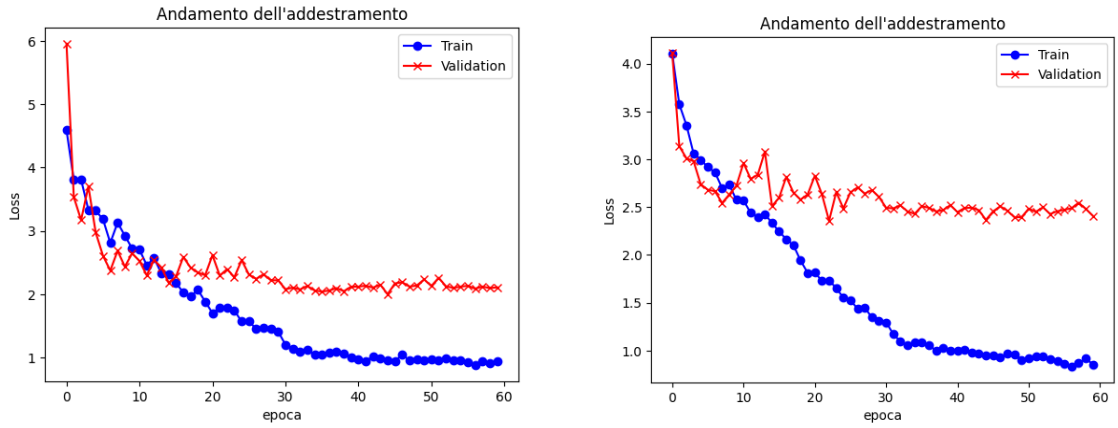
- **Congelamento dei pesi della struttura convoluzionale**: i pesi relativi ai primi livelli convoluzionali delle due reti sono stati congelati nella fase di addestramento.
  Questa scelta permette di salvaguardare le capacità di generalizzazione sviluppate dai modelli pre-allenati su *ImageNet*.
  Al fine di stabilire un buon compromesso sono stati eseguiti alcuni test, sbloccando gradualmente l'aggiornamento dei livelli convoluzionali finali della struttura portante delle reti neurali.
  I risultati hanno evidenziato come bloccare l'aggiornamento del 75% dei pesi della backbone porti a buoni risultati, con tempi di addestramento ridotti e mantenendo ottime capacità di generalizzazione.

La procedura di allenamento descritta è stata seguita per l'addestramento di entrambe le architetture neurali in esame, senza la necessità di modifiche sostanziali. Seppure la procedura di addestramento sia stata la medesima, la rete neurale basata su MobileNetV3 si è dimostrata notevolmente più efficiente durante il training rispetto alla rete neurale basata su ResNet.
L'addestramento è stato portato avanti, diminuendo gradualmente il tasso di apprendimento fino al raggiungimento di situazione stabile nella quale la funzione di Huber relativa all'insieme di validazione non ha più mostrato miglioramenti.



(a) (b)

Figura 5.8: Andamento delle Loss durante l'addestramento delle reti basate su MobileNetV3, utilizzando i valori di sterzo generati dal Controllore Stanley (a) e con Pianificazione del Guadagno (b)



(a) (b)

Figura 5.9: Andamento delle Loss durante l'addestramento delle reti basate su ResNet18, utilizzando i valori di sterzo generati dal Controllore Stanley (a) e con Pianificazione del Guadagno (b)

Le Figure 5.8 e 5.9 mostrano l'andamento della Huber Loss relativa all'insieme di allenamento (`training_set`) e a quello di validazione (`validation_set`), rispettivamente per l'addestramento delle reti neurali basate su MobileNetV3-Small e ResNet18.
La procedura ha portato all'ottenimento di 4 reti neurali:

- **MobileNet-ST**: architettura basata su **MobileNet V3 - Small**, addestrata sul dataset relativo al controllore Stanley, Figura 5.8(a).
- **MobileNet-PG**: architettura basata su **MobileNet V3 - Small**, addestrata sul dataset relativo al controllore con programmazione del guadagno, Figura 5.8(b).
- **ResNet-ST**: architettura basata su **ResNet18**, addestrata sul dataset relativo al controllore Stanley, Figura 5.9(a).
- **ResNet-PG**: architettura basata su **ResNet18**, addestrata sul dataset relativo al controllore con programmazione del guadagno, Figura 5.9(b).

Dai grafici mostrati emergono alcune differenze nel comportamento delle due architetture, e nella complessità dei datatset utilizzati.
In particolare emerge come la rete basata su ResNet18, nonostante raggiunga risultati equivalenti a quelli ottenuti dalla rete neurale basata su MobileNetV3 sull'insieme di validazione, presenti una migliore capacità di adattamento all'insieme di addestramento, mostrando le maggiori capacità della architettura.
Il continuo miglioramento della funzione di Huber sull'insieme di addestramento senza riscontri positivi in termini di validazione, suggerisce tuttavia come per un corretto addestramento di ResNet18 sia necessario un dataset più ampio.
Dai grafici in Figura 5.8 e 5.9 si nota come la funzione di Huber sull'insieme di validazione, durante l'addestramento delle reti MobileNet-ST e ResNet-ST, risulti più stabile. Inoltre al termine dell'allenamento si nota una differenza minore tra i risultati ottenuti sull'insieme di validazione e quello di allenamento rispetto ai risultati ottenuti dalle medesime architetture allenate sul dataset PG.

5.4 Inferenza della rete sulla scheda RaspBerry Pi5

Lo sviluppo della rete neurale è stato eseguito interamente con l'ausilio del framework `PyTorch` [31], è stato pertanto necessario esportare la rete neurale in modo ottimizzato e indipendente dall'ambiente Python, in modo da permetterne l'esecuzione sulla scheda RaspBerry Pi5.
A tale scopo viene utilizzata la funzione `torch.jit.script()`, fornita all'interno di PyTorch, la quale permette di trasformare un modello in una rappresentazione statica del grafo computazionale, successivamente essa viene ottimizzata appositamente per dispositivi mobili tramite le funzioni `optimize_for_mobile()` e `_save_for_lite_interpreter()`, come mostrato nel Listato 5.7.
L'esecuzione di modelli sviluppati all'interno di `PyTorch` in ambiente `c++` è supportata dalla libreria `libTorch` appositamente sviluppata.
All'interno del codice, eseguito bordo del modello, sono quindi stati sostituiti i controllori per il mantenimento di corsia e il ciclo di riconoscimento di corsia presentati

```
model.eval()
model.to('cpu')
scripted = torch.jit.script(model)
optimized = optimize_for_mobile(scripted)
optimized._save_for_lite_interpreter("...")
```

Listato 5.7: Esportazione ottimizzata della rete neurale

nei capitoli 3 e 4, da una singola chiamata all'esecuzione della rete neurale, come mostrato nel Listato 5.8.

```
torch::jit::script::Module model = torch::jit::load ("path");
...
while(true){
    ...
    cv::Mat cropped = bottomCrop(frame, 640, 300);
    cv::Mat resized;
    cv::resize(cropped, resized, cv::Size(224,224));
    torch::Tensor tensor = matToTensor(resized);
    tensor = tensor.unsqueeze(0);
    at::Tensor output = model.forward({tensor}).toTensor();
    steeringvalue = output.item<float>();
    ...
}
```

Listato 5.8: Importazione e inferenza della rete neurale

Capitolo 6

Analisi dei risultati

In questo capitolo vengono descritte le procedure di test e i confronti svolti sui diversi controllori proposti, al fine di evidenziarne le differenze.

6.1 Errore di regressione delle reti neurali sull'insieme di test

La qualità dell'emulazione, condotta dalle reti neurali addestrate nel capitolo 5.3.2, è stata valutata sull'insieme di test.
Come durante l'addestramento è stata usata la funzione di perdita di Huber per misurare l'errore di regressione sui dati forniti.
I risultati ottenuti dai 4 modelli addestrati sono riportati nella tabella 6.1.
I risultati indicano che l'architettura basata su ResNet18 ottiene un errore medio inferiore nella stima del valore di sterzo, evidenziando capacità superiori in termini di emulazione.

Tabella 6.1: Risultati dei modelli allenati sull'insieme di test (ST: Stanley, PG: pianificazione del guadagno

Modello	**Huber loss**
ResNet-PG	1.777
ResNet-ST	1.785
MobileNet-PG	2.063
MobileNet-ST	2.312

6.2 Efficienza computazionale

Per valutare l'applicabilità delle architetture neurali in un contesto di controllo in tempo reale, è stato misurato il tempo di esecuzione medio per ciascuna delle due architetture proposte, confrontandolo con i controllori tradizionali già implementati sulla piattaforma.
In questo capitolo, per tempo di esecuzione del controllore si intende l'intervallo compreso tra la fine dell'acquisizione dell'immagine e la produzione dell'angolo di sterzo.
La media $\overline{T}$ e la deviazione standard σ_T dei tempi di esecuzione misurati sperimentalmente sono riportate nella Tabella 6.2.

	Tempo di esecuzione	
Controllore	$\overline{T}$ $[ms]$	$\sigma_T[ms]$
P.G.	17	2
Stanley	17	2
ResNet18	128	24
MobileNetV3-Small	70	25

Tabella 6.2: Tempi di esecuzione dei controllori proposti

I tempi riportati nella Tabella 6.2 sono stati utilizzati per determinare il tempo di campionamento (T_c), scelto mantenendo un ragionevole margine di sicurezza, secondo quanto riportato nell'equazione 6.1.

$$T_c \quad \geq \quad \overline{T} + 2\sigma_T \tag{6.1}$$

I valori di tempo di campionamento, utilizzati durante le prove successive, per ogni controllore sono riportati nella tabella 6.3.

Controllore	T_c $[ms]$
P.G.	50
Stanley	50
ResNet18	200
MobileNetV3-Small	150

Tabella 6.3: Tempi di campionamento

I risultati ottenuti dalle architetture neurali prese in analisi mostrano un significativo vantaggio nell'utilizzo delle reti neurali basate su MobileNetV3-Small, in quanto

più ottimizzate ed efficienti soprattutto su dispositivi mobili.
La differenza di tempo di esecuzione nei confronti dei controllori tradizionali è piuttosto elevata, tuttavia, almeno nel caso dei modelli basati su MobileNetV3-Small il tempo di esecuzione non risulta eccessivamente penalizzante.
Un aspetto critico che emerge analizzando le tabelle 6.2 e 6.3 riguarda la deviazione standard dei tempi di esecuzione σ_T per le reti neurali, una variabilità elevata infatti impedisce di abbassare il tempo di campionamento se non aumentando significativamente il rischio di un esecuzione non correttamente discretizzata, costringendo la scelta di tempi di campionamento molto maggiori dei tempi di inferenza medi delle reti neurali.

6.3 Mantenimento di corsia su strada rettilinea

Durante le prove il modello in scala viene posto all'inizio di un tratto di strada rettilinea, in una posizione arbitrariamente scelta, con i seguenti errori di inseguimento:

$$\begin{bmatrix} \psi \\ \epsilon \end{bmatrix} = \begin{bmatrix} \text{-}7^\circ \\ -0.15m \end{bmatrix} \tag{6.2}$$

La scelta di tale condizione iniziale deriva dalla volontà di valutare le prestazioni del controllo in condizioni non banali, esaltando in questo modo le differenze tra le diverse soluzioni.
Dalla posizione iniziale il controllore viene avviato, e durante la sua esecuzione vengono salvati i valori dell'errore di allineamento e dell'errore laterale.
Il valore assoluto degli errori di inseguimento è utilizzato per un'analisi statistica, determinandone il valore medio e la deviazione standard una volta esaurito il transitorio iniziale. L'eliminazione del transitorio, della durata pari approssimativa al 15% della durata della prova, permette di valutare la stabilità del mantenimento di corsia una volta raggiunta una posizione centrale nella corsia.
La prova è considerata svolta con successo se il modulo dell'errore laterale non supera il valore di $0{,}17m$ durante la prova, il superamento di tale soglia comporta l'uscita del modello dalle linee di demarcazione della corsia.
La prova viene ripetuta due volte, in modo tale da ottenere delle misure statisticamente più affidabili.

I risultati ottenuti dai controllori tradizionali, presentati nei capitoli 3 e 4 sono riportati nelle Figure 6.1 e 6.2, tali risultati rappresentano il riferimento per la valutazione delle prestazioni offerte dalle reti neurali addestrate durante questo lavoro.
Le prove condotte mostrano come entrambi i controllori abbiano mantenuto il modello in scala all'interno della corsia in entrambe le prove condotte.

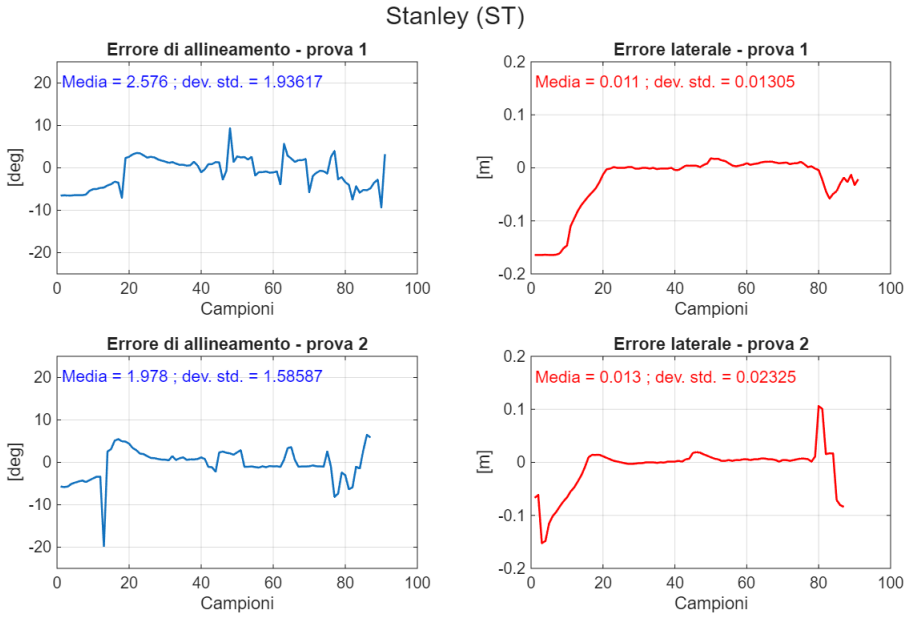


Figura 6.1: Risultati delle prove svolte utilizzando il controllore Stanley

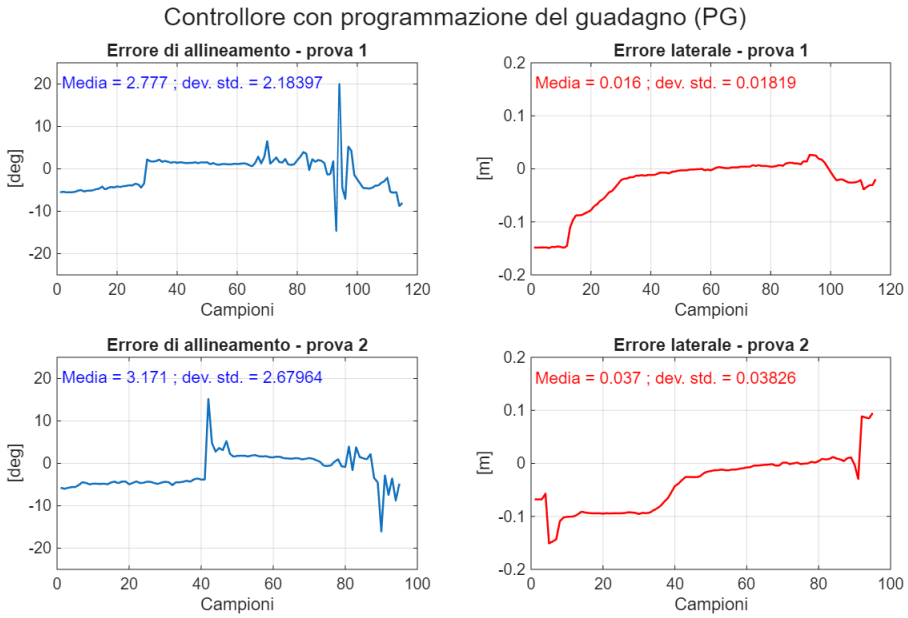


Figura 6.2: Risultati delle prove svolte utilizzando il controllore con progettazione del guadagno

L'analisi statistica, di cui i risultati sono mostrati in Figura 6.3, mostra che il controllore Stanley garantisce prestazioni superiori al controllore con pianificazione del guadagno, ottenendo valori inferiori di media e deviazione standard per entrambe le prove.

Il test di mantenimento di corsia è stato svolto con le medesime modalità anche

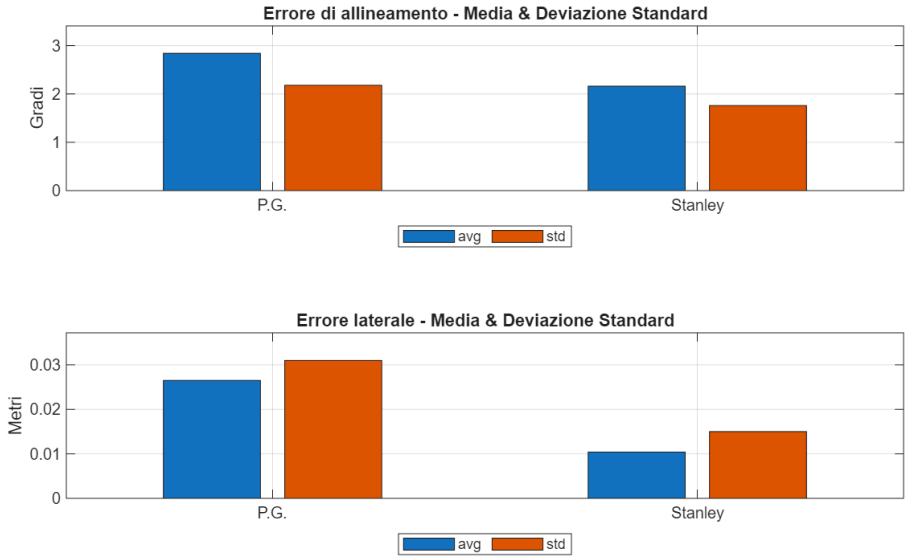


Figura 6.3: Confronto statistico degli errori d'inseguimento dei controllori Stanley (ST) e con pianificazione del guadagno (PG)

utilizzando le reti neurali ottenute nel capitolo 5.3.2.

6.3.1 Risultati dell'architettura basata su ResNet18

Dai risultati riportati nelle Figure 6.4 e 6.5 è possibile osservare come entrambe le reti basate su ResNet18 abbiano eseguito con successo il mantenimento della corsia.
Le analisi statistiche svolte sull'andamento degli errori di inseguimento durante le prove, (Figura 6.6), mostrano tuttavia risultati contrastanti con quanto atteso: al contrario di quanto ottenuto dai controllori tradizionali, la rete ResNet-ST mostra risultati leggermente peggiori di ResNet-PG, sia in termini di valore medio che di varianza per entrambi gli errori di inseguimento.
L'analisi delle prestazioni dei modelli basati su ResNet18, mostra prestazioni significativamente inferiori rispetto a quanto mostrato dai controllori tradizionali, con errori di inseguimento significativamente superiori in valore medio.

6.3.2 Risultati dell'architettura basata su MobileNetV3

I risultati ottenuti dalle prove di mantenimento di corsia in rettilineo svolte con le reti neurali basate su MobileNetV3 sono riportati nelle Figure 6.7 e 6.8.
I due modelli in esame si sono dimostrati in grado di mantenere il veicolo all'interno della corsia designata.

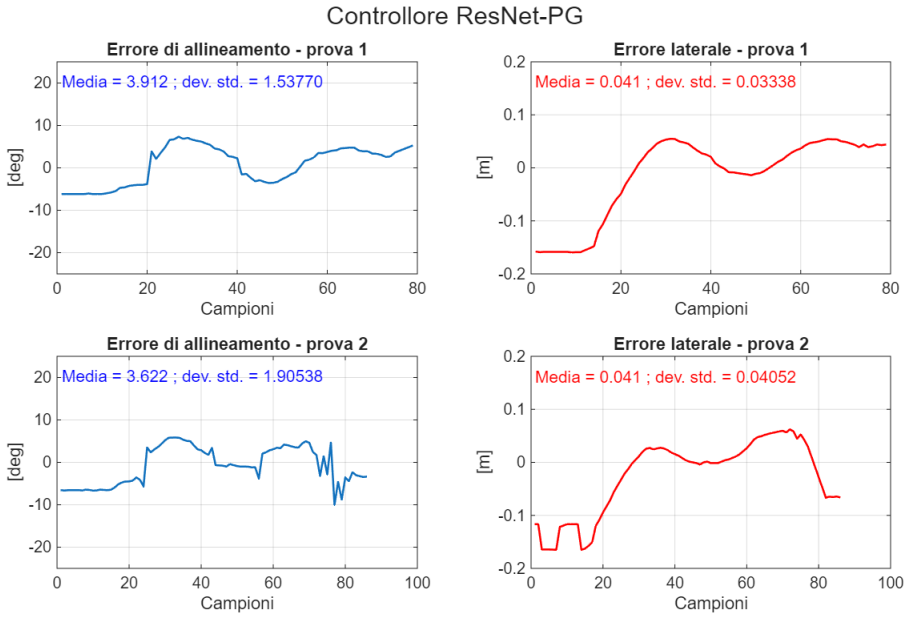


Figura 6.4: Risultati della prova svolta utilizzando la rete ResNet-PG

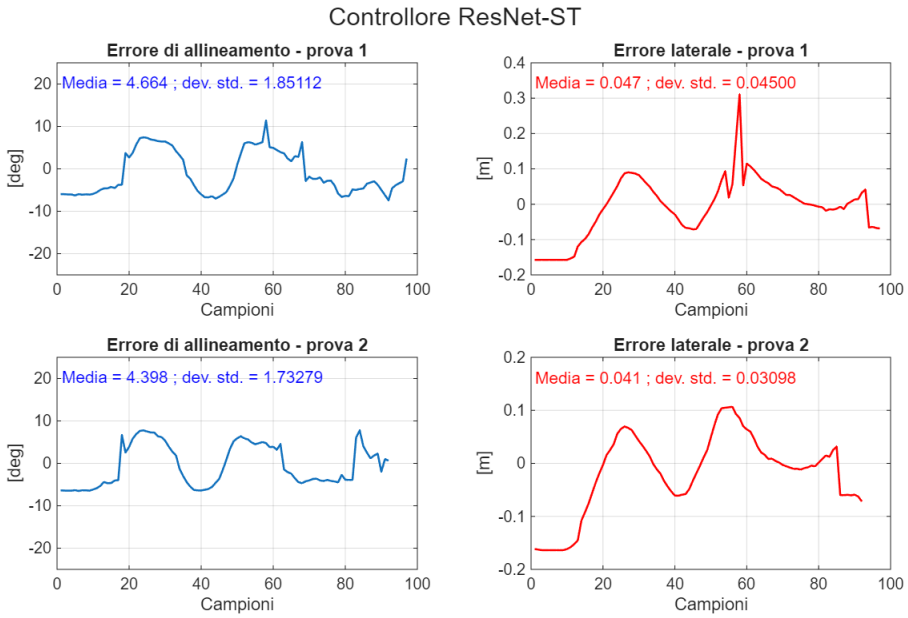


Figura 6.5: Risultati della prova svolta utilizzando la rete ResNet-ST

L'analisi statistica condotta sugli errori di inseguimento, i cui risultati sono visibili in Figura 6.9, mostra prestazioni in termini di valore medio, comparabili a quelle dei controllori tradizionali, ottenendo, in particolare con il modello MobileNet-PG, una deviazione standard degli errori minori rispetto a quanto ottenuto sperimentalmente con il controllore con pianificazione del guadagno.

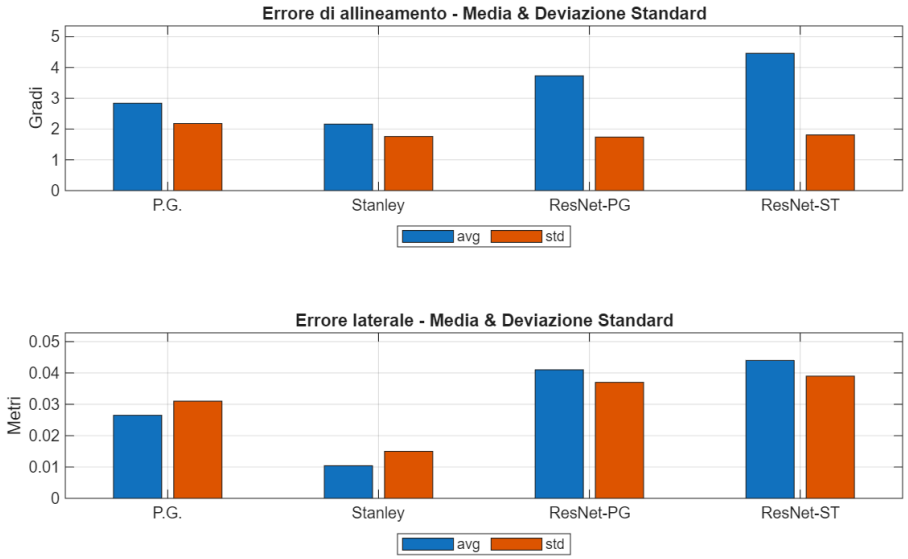


Figura 6.6: Confronto statistico degli errori d'inseguimento tra i controllori tradizionali e le reti neurali basate su ResNet18

Come atteso, sulla base dei risultati ottenuti dai controllori tradizionali, la rete neurale MobileNet-ST presenta risultati migliori di MobileNet-PG in termini di valore medio degli errori di inseguimento, seppure con uno scarto minimo.

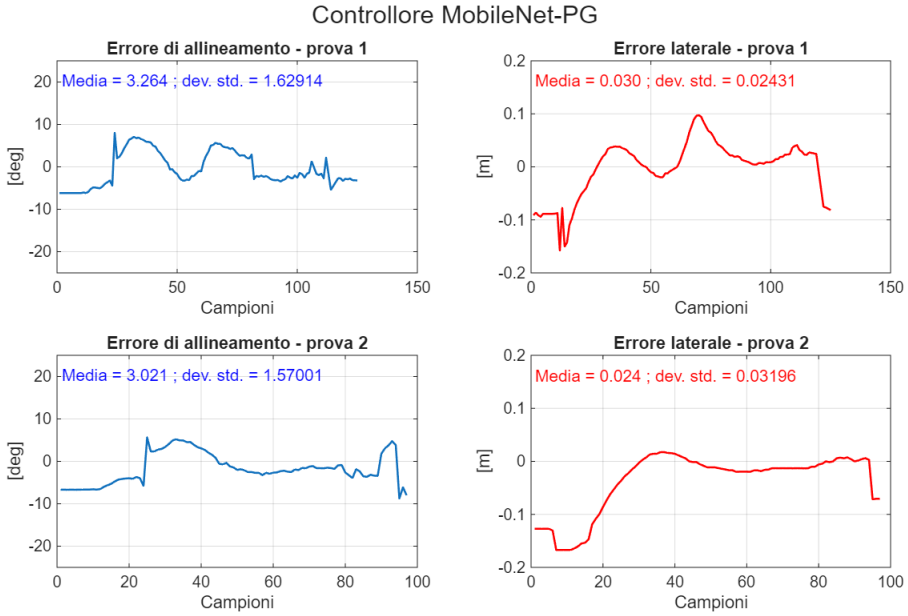


Figura 6.7: Risultati della prova svolta utilizzando la rete MobileNet-PG

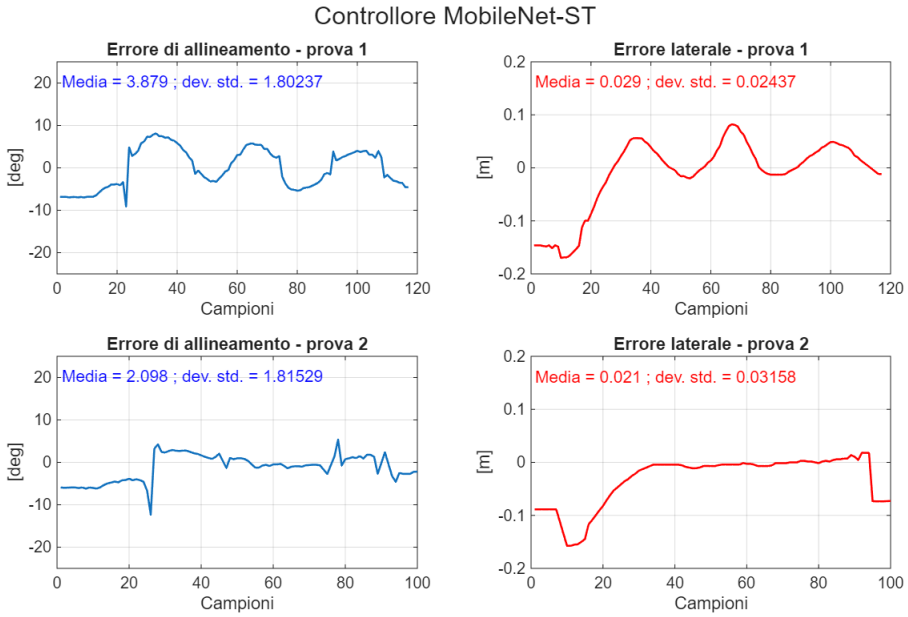


Figura 6.8: Risultati della prova svolta utilizzando la rete MobileNet-ST

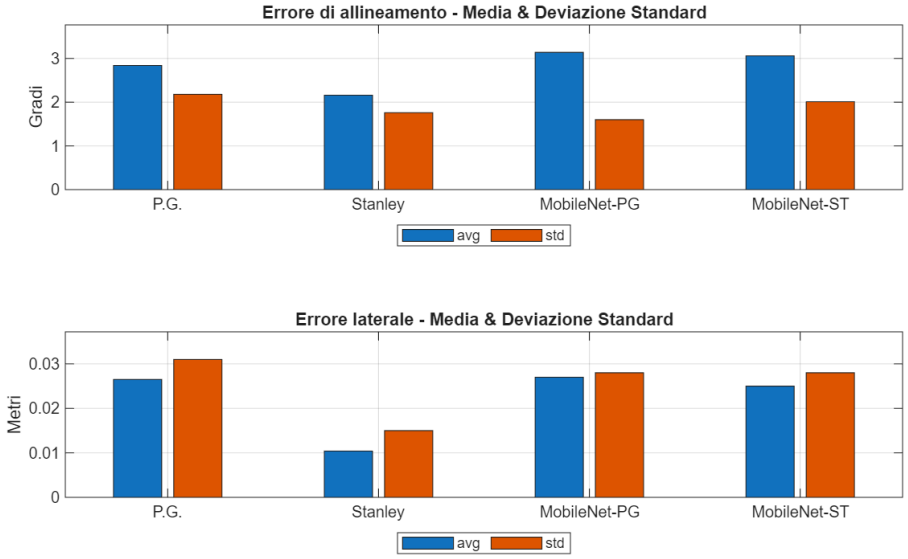


Figura 6.9: Confronto statistico degli errori d'inseguimento tra i controllori tradizionali e le reti neurali basate su MobileNetV3

6.3.3 Analisi dei risultati

I risultati ottenuti nelle prove sono riassunti nella Tabella 6.4 e graficamente nella Figura 6.10.

	Errore di allineamento		**Errore laterale**	
Controllore	$\overline{\psi}$	σ_ψ	$\overline{\epsilon}$	σ_ϵ
P.G.	2.84	2.18	0.027	0.031
Stanley	2.16	1.76	0.010	0.015
ResNet-PG	3.73	1.74	0.041	0.037
ResNet-ST	4.46	1.81	0.044	0.039
MobileNet-PG	3.14	1.60	0.027	0.028
MobileNet-ST	3.06	2.01	0.025	0.028

Tabella 6.4: Risultati dell'analisi statistica sugli errori di inseguimento registrati durante la prova

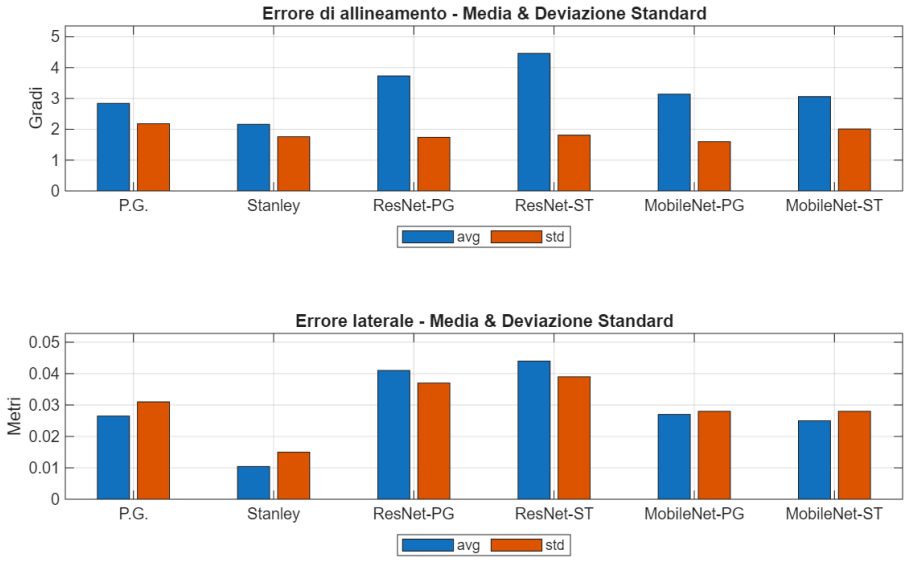


Figura 6.10: Confronto statistico degli errori d'inseguimento tra i controllori tradizionali e i modelli addestrati

I risultati evidenziano che le soluzioni che garantiscono prestazioni di guida migliori nelle condizioni in esame sono i controllori tradizionali, in particolar modo il controllore Stanley.
Confrontando tra loro i quattro modelli addestrati, quello che ha fornito i migliori risultati, in termini di valore medio degli errori di inseguimento, è **MobileNet-ST**, con uno scarto minimo nei confronti di MobileNet-PG, tale risultato si pone in contrapposizione con quanto evidenziato dalle prove sull'insieme di test. Infatti, nonostante i modelli basati su ResNet18 ottengano risultati migliori nelle prove sull'insieme di test , la prova sul campo di mantenimento di corsia mostra prestazioni superiori per le reti neurali basate su MobileNetV3 Small, verosimilmente grazie

alla maggiore efficienza computazionale in fase di inferenza, la quale permette di operare con un tempo di campionamento inferiore, come già stabilito nell'analisi dei tempi di esecuzione condotta nel capitolo 6.2.

Capitolo 7

Conclusioni e sviluppi futuri

7.1 Conclusioni

Analizzando il lavoro svolto per questa tesi sono emerse alcune considerazioni riguardo all'applicazione delle reti neurali nell'ambito della guida autonoma e più in generale nell'ambito dei controlli automatici.
L'utilizzo di modelli di deep learning per l'emulazione di controllori deterministici rappresenta una soluzione promettente, capace di combinare la solidità dei metodi deterministici con la flessibilità delle architetture neurali.
I risultati ottenuti durante le prove dei modelli hanno inoltre evidenziato la capacità delle reti neurali prese in esame di eseguire con successo il controllo di corsia, ottenendo una buona accuratezza e robustezza in diverse condizioni.

L'aspetto più critico emerso durante le fasi di prova è stata l'efficienza computazionale del modello in fase di inferenza.
I risultati sperimentali hanno infatti mostrato che i modelli meno complessi e più efficienti ottengono prestazioni di guida statisticamente migliori rispetto ai modelli basati su reti neurali complesse e meno ottimizzate, nonostante queste ultime ottengano risultati migliori nei test di regressione. I migliori risultati ottenuti dalle architetture più efficienti sono ragionevolmente dovuti ai minori tempi di inferenza, i quali permettono una correzione più rapida ed efficiente degli errori di inseguimento.

Tra gli altri aspetti critici si è evidenziata la necessità di un dataset di grandi dimensioni, all'interno del quale siano presenti in quantità elevata esempi di tutte le manovre che il veicolo è chiamato a compiere, nella maggiore varietà possibile di scenari.
La dimensione del dataset influisce direttamente sulla robustezza e la ripetibilità del controllo che la rete neurale mette in atto, risultando a tutti gli effetti uno degli aspetti più rilevanti per ottenere prestazioni di guida di alto livello.

Le tecniche di aumento sintetico dei dati, utilizzate durante questo lavoro di tesi, si sono rivelate inoltre valide al fine di addestrare una rete neurale di piccole dimensioni, risultando quindi consigliate anche qualora si avesse disponibilità maggiore di dati.

Un aspetto importante emerso, data la bassa quantità di dati raccolti, è l'importanza di adottare strategie atte a evitare la condizione di overfitting del modello all'insieme di addestramento, utilizzando tecniche come il congelamento di parte della rete neurale in fase di addestramento, e la terminazione anticipata dell'addestramento (early stopping), oltre alla scelta di un'architettura correttamente dimensionata alla quantità di dati a disposizione.

Analizzando, infine, il ruolo dei controllori utilizzati per la generazione del dataset, la loro qualità risulta un fattore rilevante nell'addestramento del modello e nelle sue prestazioni.
Questo aspetto è testimoniato dal fatto che il controllore Stanley, risultato il migliore nelle prove sperimentali, è alla base del modello con le prestazioni migliori, MobileNet-ST.
Tuttavia nonostante la sua importanza, il controllore originale, non rappresenta il fattore più determinante nelle prestazioni della rete neurale addestrata. Il comportamento della rete infatti può essere influenzato da modifiche mirate all'interno del dataset.
In conclusione è possibile affermare che un controllore con buone prestazioni agevoli la rete neurale nell'ottenimento di prestazioni di alto livello, rappresentando un imprescindibile punto di partenza, nonostante ciò, la rete neurale mantiene una capacità di generalizzazione che può essere sfruttata tramite un'attenta progettazione del dataset al fine di migliorarne le prestazioni.

7.2 Sviluppi futuri

Il lavoro presentato in questa tesi ha dimostrato la possibilità di emulare il controllo di corsia tramite l'utilizzo di una rete neurale appositamente addestrata, lasciando tuttavia molto spazio per sviluppi futuri.
Le possibili soluzioni per migliorare le prestazioni ottenute durante questo lavoro di tesi sono molteplici, tra queste vi è un miglioramento dell'hardware, l'utilizzo di architetture più espressive o efficienti, la sperimentazione di strategie non supervisionate per l'addestramento dei modelli, e lo sviluppo di nuove funzionalità da includere nel dataset.

Nell'ottica di un miglioramento prestazionale del modello in scala, le componenti elettroniche più influenti, e quindi preferibilmente aggiornabili, sono la scheda su

cui viene eseguito il controllo, e la fotocamera utilizzata per l'acquisizione delle immagini.
L'aggiornamento della scheda RaspBerry Pi5 con una piattaforma che offra prestazioni superiori permetterebbe l'utilizzo di reti neurali con dimensioni e capacità espressive superiori, mantenendo contemporaneamente il tempo d'inferenza all'interno di un intervallo accettabile e permettendo un significativo miglioramento delle prestazioni di guida.
Un altro aggiornamento significativo nell'ottica di un miglioramento dell'hardware del modello in scala è la sostituzione della fotocamera, a favore di un modello che presenti un campo visivo (FOV) più ampio, tale da permettere una visione periferica migliore. Il FOV è rilevante nelle fasi di curva dove una maggiore informazione visiva laterale permette una predizione dell'angolo di sterzo più accurata, oltre a permettere lo sviluppo di funzionalità quali il riconoscimento della segnaletica stradale o una efficace gestione del veicolo all'interno degli incroci.

Dal punto di vista della scelta dell'architettura della rete neurale vi è la possibilità, se supportata da un hardware adeguato, di sfruttare la maggiore capacità messa a disposizione da modelli più profondi, come ad esempio la rete MobileNetV3-Large, la quale mantenendo le caratteristiche della rete MobileNevV3-Small utilizzata in questo progetto, ne aumenta le capacità a un costo computazionale maggiore.
L'utilizzo di una rete neurale con maggiori capacità espressive rende necessario oltre ad un adeguata potenza computazionale, l'utilizzo di un dataset di dimensioni adeguate, in modo da poter sfruttare le potenzialità della rete, riducendo il rischio di overfitting.

Un'ulteriore direzione di sviluppo riguarda la possibilità di integrare tecniche di addestramento non supervisionato a partire dal modello ottenuto tramite addestramento per imitazione.
Questa strategia di addestramento in due fasi rappresenta una soluzione significativamente più veloce rispetto a un addestramento non supervisionato tradizionale, offrendo inoltre maggiori possibilità di progettazione del comportamento del modello, tramite la scelta del controllore iniziale e applicando le tecniche di aumento sintetico del dataset descritte in questa tesi.

Un'ulteriore prospettiva di sviluppo, al fine di ampliare le capacità del veicolo di gestire scenari di vario genere, riguarda la possibilità di ampliare il dataset includendo nuove etichette e informazioni, permettendo alla rete di apprendere compiti aggiuntivi senza la necessità di riprogettare interamente il controllo.
Questo approccio permetterebbe alla rete di apprendere una vastità di nuove funzionalità tra le quali, il controllo della velocità, il riconoscimento degli ostacoli, e il riconoscimento della segnaletica stradale.

Bibliografia

[1] "Formula student driverless," https://www.imeche.org/events/formula-student/team-information/fs-ai.

[2] "Indy autonomous challenge," https://www.indyautonomouschallenge.com/.

[3] "Darpa grand challenge," https://www.darpa.mil/news/2014/grand-challenge-ten-years-later.

[4] "Bosch future mobility challenge," https://boschfuturemobility.com/.

[5] L. Evans and R. C. Schwing, "A critical view of driver behavior models: What do we know, what should we do?" in *Human Behavior and Traffic Safety*. Springer, 1985.

[6] SAE International, "Sae j3016: Levels of driving automation," SAE International, Technical Report, 2021.

[7] "Raspberry pi5," https://www.raspberrypi.com/products/raspberry-pi-5/.

[8] L. D. Biase, "Lane keeping algorithms and intersection management on a scaled mock-up autonomous vehicle," Master's thesis, Politecnico di Torino, 2024.

[9] A. Sedda, "Lane keeping in roadwork conditions for a scaled autonomous car," Master's thesis, Politecnico di Torino, 2025.

[10] "Documentazione ufficiale opencv," https://docs.opencv.org/3.4/index.html.

[11] A. R. Smith, "Color gamut transform pairs," *Computer Graphics*, vol. 12, no. 3, 1978.

[12] H. Durairaj, "Interactive color image segmentation using hsv color space," *Science Technology Journal*, vol. 7, pp. 37–41, 2020.

[13] S. T. Karri, "Hough transform," https://medium.com/@st1739/hough-transform-287b2dac0c70, 2019.

[14] M. Boggio, L. Colangelo, M. Virdis, M. Pagone, and C. Novara, "Earth gravity in-orbit sensing: Mpc formation control based on a novel constellation model," *Remote Sensing*, vol. 14, no. 12, 2022. [Online]. Available: https://www.mdpi.com/2072-4292/14/12/2815

[15] M. D, "Tuning pid controller for self-driving cars," https://medium.com/@madhusudhan.d/tuning-pid-controller-for-self-driving-cars-3813f7f18eb0, 2020.

[16] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, C. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci,

V. Pratt, P. Stang, J. Strobel, P. Dupont, D. Jendrossek, C. Koelen, A. Markey, C. Rummel, J. Stephenson, N. Talbot, R. Zbikowski, and G. Bradski, "Stanley: The robot that won the darpa grand challenge," in *Journal of Field Robotics, Special Issue on the 2005 DARPA Grand Challenge*, vol. 23, no. 9. Wiley, 2006, pp. 661–692.

[17] G. Schildbach, F. Borrelli, J. Kong, and M. Pfeiffer, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 164–170.

[18] D. Yan, "Three methods of vehicle lateral control: Pure pursuit, stanley and mpc," https://dingyan89.medium.com/three-methods-of-vehicle-lateral-control-pure-pursuit-stanley-and-mpc-db8cc1d32081, 2020.

[19] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[20] Z. Lorincz. (2019) A brief overview of imitation learning. [Online]. Available: https://medium.com/@zoltan.lorincz95/a-brief-overview-of-imitation-learning-25b2a1883f4b

[21] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in Neural Information Processing Systems*, 1989.

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 248–255.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[25] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[26] Y. Guo, Y. Li, R. Feris, L. Wang, and T. Rosing, "Depthwise convolution is all you need for learning multiple visual domains," 2019. [Online]. Available: https://arxiv.org/abs/1902.00927

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520, arXiv:1801.04381 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1801.04381

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[29] P. J. Huber, "Robust Estimation of a Location Parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, 1964. [Online]. Available: https://doi.org/10.1214/aoms/1177703732

[30] R. Alake. (2024) Loss function in machine learning. [Online]. Available: https://www.datacamp.com/tutorial/loss-function-in-machine-learning

[31] "Documentazione ufficiale pytorch," https://docs.pytorch.org/docs/stable/index.html.