



UNIVERSITÀ DEGLI STUDI DI MESSINA

Dipartimento di Ingegneria
Corso di Laurea in Ingegneria Elettronica ed Informatica

**Tecniche di fusione multisensoriali
per una stima robusta e accurata della velocità
di un veicolo a guida autonoma**

Relatore:
Chiar.mo Prof.
Luca PATANÈ

Secondo relatore:
Chiar.mo Prof.
Francesco LONGO

Presentata da:
Angelo ZAGAMI

Terza Sessione
Anno Accademico 2020/2021

*A mamma e papà,
grazie di tutto.*

*A mio fratello,
che riesce sempre a strapparmi un sorriso.*

*A Miriana,
sei la parte migliore di me.*

*A tutte quelle persone che,
per i casi più disparati della vita,
non possono permettersi un'istruzione.*

Abstract

La stima dello “stato” di un veicolo rappresenta un prerequisito per gli Advanced Driver-Assistant Systems (ADAS) e, più in generale, per la guida autonoma. In particolare, gli algoritmi di Simultaneous Localization and Mapping (SLAM) e di pianificazione della traiettoria richiedono la conoscenza continua di dati come la velocità del veicolo, l’angolo di sterzata, la distanza dai confini stradali. La stima dello stato del veicolo può essere effettuata mediante metodologie proprie dell’automatica tra cui i filtri di Kalman, siano essi standard o estesi. Inoltre, per garantire una stima quanto più accurata possibile, è auspicabile l’utilizzo di più sensori a bordo del veicolo, i cui dati vengono poi integrati mediante tecniche di sensor fusion.

Obiettivo dell’elaborato è di presentare le principali tecniche di fusione multisensoriale, con particolare attenzione ai filtri di Kalman. Dopo un’analisi teorica dei filtri di Kalman verranno presentate alcune simulazioni relative all’utilizzo del filtro su un veicolo per applicazioni alla guida autonoma.

Ringraziamenti

Un ringraziamento speciale va alla mia famiglia, a mia madre Stefania, a mio padre Mariano e a mio fratello Francesco Pio, per tutto l'amore e il sostegno datomi durante tutto questo percorso e non solo.

Grazie a nonna Pina e a nonno Ciccio per tutto il vostro immenso amore. Un grazie speciale anche a nonno Angelo che mi guarda da lassù.

Grazie a Miriana, per aver sopportato tutti i miei scleri e per essere sempre il mio sole, anche nelle giornate grigie.

Grazie ai miei relatori, il prof. Luca Patanè e il prof. Francesco Longo, per tutto il supporto datomi durante la stesura di questa tesi.

Ringrazio i miei colleghi e amici, in particolare Bonny, Pasti, Ghella, Davide, Giovanni, Enrico, Ilenia e Cristian, per tutte le magnifiche giornate passate insieme.

Indice

Abstract	i
Ringraziamenti	iii
1 Introduzione	1
1.1 Fusione multisensoriale	2
1.2 Classificazione delle tecniche di fusione	3
1.2.1 Classificazione in base al tipo	3
1.2.2 Classificazione in base al livello di astrazione	4
1.2.3 Classificazione in base all'approccio	5
1.3 Principali tecniche di fusione multisensoriale	6
1.3.1 Algoritmi basati sul teorema del limite centrale	6
1.3.2 Algoritmi basati su reti bayesiane	7
1.3.3 Convolutional neural network	7
1.3.4 Filtri di Kalman	7
2 Il filtro di Kalman	9
2.1 Filtro di Kalman per un sistema ad una sola variabile	10
2.1.1 Esempio pratico	11
2.2 Filtri tempo continuo e tempo discreto	12
2.3 Modello multidimensionale	12
2.3.1 Ipotesi	13
2.3.2 Fase di init	14
2.3.3 Fase di predict	14
2.3.4 Fase di update	15
2.4 Filtro di Kalman Esteso	17
2.4.1 Svantaggi	17
2.5 Esempio: moto in 2 dimensioni	18
3 Modelli sperimentali	21
3.1 Modello del punto materiale	21
3.1.1 Codice di simulazione	23
3.2 Modello Simulink	24

3.2.1	Codice di simulazione	26
3.3	Modello Simulink con EKF	27
3.3.1	Modello del sensore - Radar	28
3.3.2	Modello del sensore - GPS	29
3.3.3	Codice di simulazione	29
4	Risultati delle simulazioni	31
4.1	Simulazione tramite Driving Scenario Designer	31
4.2	Simulazione modello Simulink	33
4.3	Simulazione modello Simulink con EKF	38
5	Conclusioni	41
A	Codici sorgenti	43
	Bibliografia	53

Elenco delle figure

1.1	Livelli di guida autonoma secondo lo standard J3016 [2]	2
1.2	Schematizzazione di una vettura a guida autonoma	3
1.3	Fusione competitiva, complementare e cooperativa [4]	5
2.1	Schematizzazione del filtro di Kalman per una variabile	10
2.2	Applicazione del filtro per la stima della temperatura	12
2.3	Applicazione del filtro per la stima della traiettoria di un oggetto in 2D	19
2.4	Dettaglio asse x	19
2.5	Dettaglio asse y	20
3.1	Schermata principale del pacchetto Driving Scenario Designer di Matlab	22
3.2	Schema completo del modello realizzato in Simulink	25
3.3	Schematizzazione dell'auto vettura	26
3.4	Schema completo del modello realizzato in Simulink con l'EKF (EKF-sim.slx)	28
3.5	Blocco Simulink Function - State Transition Jacobian	28
4.1	Schermata del software Driving Scenario Designer durante la costruzione del percorso per la simulazione	32
4.2	Velocità sull'asse X in funzione del tempo	32
4.3	Velocità sull'asse Y in funzione del tempo	33
4.4	Traiettoria percorsa dall'auto	34
4.5	Velocità sull'asse X	34
4.6	Velocità sull'asse X con matrice Q tempo-variante	35
4.7	Velocità asse X - tempo-invariante	35
4.8	Velocità asse Y - tempo-invariante	36
4.9	Velocità asse X - tempo-variante	36
4.10	Velocità asse Y - tempo-variante	37
4.11	Velocità ed errore della stima sull'asse X EKF	38
4.12	Velocità sull'asse X EKF	39
4.13	Velocità ed errore della stima sull'asse Y EKF	39
4.14	Velocità sull'asse Y EKF	40
4.15	Dettaglio stima velocità	40

Elenco delle tabelle

2.1	Applicazione del filtro di Kalman per la stima della temperatura	11
4.1	Waypoints che definiscono il percorso compiuto dall'auto	31
4.2	Errori massimi, minimi e medi della stima mediante filtro di Kalman .	33
4.3	Confronto errori simulazione Simulink	37
4.4	Confronto errori simulazione Simulink	38

Elenco dei codici

3.1	Applicazione del filtro di Kalman	23
3.2	Parte dello script Matlab per la simulazione del filtro di Kalman con il modello Simulink	27
A.1	Script Matlab per la simulazione del filtro di Kalman	43
A.2	Script Matlab per la simulazione del filtro di Kalman mediante Simulink	47

Glossario

A Matrice transizione di stato, può essere indicata anche con F.

B Matrice di controllo, può essere indicata anche con G.

EKF Filtro di Kalman esteso.

H Matrice di misura.

KF Filtro di Kalman.

KG/K Guadagno di Kalman.

P Matrice di covarianza dello stato.

Q Matrice di covarianza del rumore sul processo.

R Matrice di covarianza del rumore di misura.

u Vettore di input.

w Rumore sul processo.

x Vettore di stato.

z Rumore di misura.

Capitolo 1

Introduzione

Il progresso tecnologico degli ultimi anni ha permesso la realizzazione di sistemi di controllo e sicurezza sempre più complessi e affidabili, ponendo le basi per la realizzazione di veicoli in grado di guidare autonomamente anche in assenza di conducente. Se già negli scorsi anni hanno visto la luce autovetture in grado di assistere il conducente con sistemi quali il controllo di corsia e di velocità, dal 2020 sono disponibili in Europa i primi modelli di auto aventi un livello di automazione tale da permettere, se pur parzialmente e per breve durata, al conducente di allontanare le mani dal volante. Nel 2014 la **Society of Automobile Engineers** (SAE International), un ente di normazione nel campo dell'industria automobilistica, ha pubblicato un nuovo standard internazionale J3016 che ha definito sei differenti livelli per la guida autonoma, mostrati in Figura 1.1. Questa classificazione è basata sul grado di interazione del guidatore, più che sulle capacità del mezzo [1]. I veicoli a guida assistita (livelli 1 e 2 d'automazione) sono già in commercio in Europa. Le auto a guida autonoma (livelli 3 e 4) sono al momento in fase di collaudo e verifica e faranno il loro ingresso nel mercato tra il 2020 e il 2030, mentre i veicoli a guida interamente autonoma (livello 5) dovrebbero essere pronti per il 2030. Ci si aspetta che tutte le nuove autovetture siano dotate di connessione a partire dal 2022 [2]. Il tutto finalizzato a garantire una maggior sicurezza stradale, i cui incidenti sono causati, nella maggioranza dei casi, da errori umani. Si prevede che la guida autonoma raggiunga un livello di sicurezza comparabile a quello umano già nel 2024, arrivando nel 2046 a soddisfare l'intera domanda di mobilità degli Stati Uniti. Sulla base di questi numeri si prevede che sarà probabile che molti paesi decideranno di vietare la guida umana sulle strade al fine di evitare di interferire con il normale transito dei veicoli autonomi. L'obiettivo ultimo sarà quello di ridurre, o addirittura azzerare, il numero di incidenti provocato dal normale traffico automobilistico [3].

Perchè questo accada, è necessario che i sistemi di controllo siano sempre più sicuri e precisi. Uno dei tasselli fondamentali riguarda la stima dello "stato" del veicolo, dove per stato si intendono tutte quelle informazioni come velocità, accelerazione, posizione, distanza dai confini stradali, e tutto ciò che riguarda il veicolo e il suo funzionamento. Nonostante essi siano equipaggiate con sensori sempre più precisi e accurati, non è possibile annullare totalmente l'incertezza dovuta alle proprietà intrinseche dei sensori



Figura 1.1: Livelli di guida autonoma secondo lo standard J3016 [2]

nè, tantomeno, le interferenze elettromagnetiche che vanno ad agire sulla trasmissione dei dati. Inoltre, data la complessità del compito e l'importanza in termini di sicurezza che esso riveste, è auspicabile che i veicoli siano equipaggiati con più sensori, in modo tale da poter garantire sia la tolleranza ad eventuali guasti, sia una stima più precisa e accurata. Tuttavia, è necessario combinare i dati provenienti dai vari sensori, molto spesso eterogeni tra loro, in maniera appropriata per ottenere una stima quanto più accurata possibile, considerando sempre rumori e/o interferenze. Tutto ciò è possibile grazie alle tecniche di fusione multisensoriale.

1.1 Fusione multisensoriale

La **fusione multisensoriale** (o sensor fusion) è il processo di combinazione dei dati provenienti da più sensori o, in generale, più sorgenti con l'obiettivo di avere una migliore conoscenza del sistema, rispetto a quella che si avrebbe se i sensori fossero presi singolarmente. Per miglior conoscenza del sistema si intende una conoscenza più accurata, più affidabile e più consistente nel tempo. Oltre ai sensori, la sorgente dei dati può essere anche un modello matematico, è possibile, infatti, combinare i dati provenienti dai sensori con un modello al fine di ottenere una migliore stima dei dati. Volendo schematizzare il funzionamento di un veicolo a guida autonoma, come mostrato in Figura 1.2, la parte più importante è quella di percezione, in base ai dati provenienti dai sensori essa può stabilire se un ostacolo a bordo strada sia un pedone o

un semplice palo e agire di conseguenza. Un errore di questo livello potrebbe causare danni irreparabili, come la perdita di vite umane. Questo passaggio, all'interno del

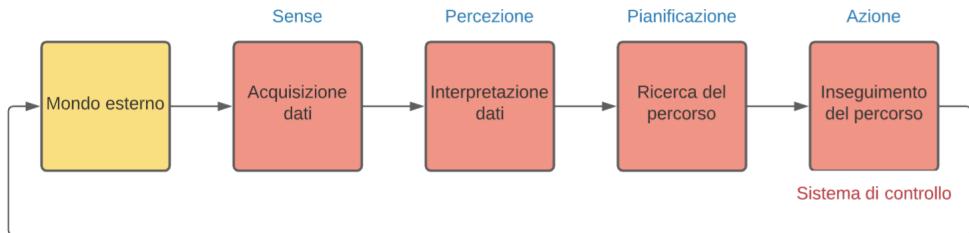


Figura 1.2: Schematizzazione di una vettura a guida autonoma

processo, è responsabile sia dell'autoconsapevolezza - la quale viene definita come la capacità di localizzazione e/o posizionamento - che della cognizione della situazione, ovvero il metodo con il quale rilevare altri oggetti nell'ambiente e seguirli (o evitarli). Il processo di fusione multisensoriale opera sui sensori e sulla percezione, permettendo di fornire dei dati più accurati e quindi una migliore percezione al veicolo. Inoltre, quest'ultima può essere utilizzata per stimare stati non misurati, si pensi ad esempio all'utilizzo di più telecamere per determinare la distanza di un oggetto. Ogni telecamere non misura la distanza in maniera individuale ma l'insieme delle informazioni permette di stimare questo dato. In definitiva, possiamo definire quattro modi in cui la fusione dei segnali provenienti dai sensori può migliorare la localizzazione e il posizionamento del sistema, oltre a rilevare e tracciare altri oggetti:

- Incrementa la qualità dei dati;
- Può aumentare l'affidabilità delle misurazioni;
- Può essere usata per stimare stati non misurati;
- Può essere usata per aumentare l'area di copertura.

1.2 Classificazione delle tecniche di fusione

È possibile classificare le tecniche di fusione in vari modi, di seguito saranno discussi i principali.

1.2.1 Classificazione in base al tipo

La prima classificazione riguarda il *tipo* di fusione o la *configurazione di sensori* che si vuole ottenere, ovvero se il risultato desiderato è una fusione di tipo competitivo, complementare o cooperativo [4, 5]:

- Una rete di sensori **cooperativa** utilizza le informazioni fornite da due sensori indipendenti per ricavare informazioni che non sarebbero disponibili dai singoli

sensori. Ad esempio, nel caso della *triangolazione* si combinano varie misure di distanza per ottenere un’informazione di posizione. In questo caso si parla di fusione più in senso lato che in senso proprio. Viene definita fusione cooperativa perché tutte le informazioni provenienti dalle varie sorgenti vengono usate in maniera cooperativa (quindi in maniera associata l’una all’altra) per completare un’informazione. In questo caso si parla di vista emergenti, o quantità derivate, perché si sta andando a ottenere un’informazione su qualcosa che individualmente non era ottenibile. La fusione di sensori cooperativa è la più difficile da progettare perché i dati risultanti sono sensibili alle imprecisioni in tutti i singoli sensori partecipanti. I sensori S_4 e S_5 in Figura 1.3 rappresentano una configurazione cooperativa. Entrambi i sensori osservano lo stesso oggetto, ma le misurazioni vengono utilizzate per formare una vista emergente dell’oggetto C che non potrebbe essere derivata dalle misurazioni di S_4 o S_5 [4];

- La sensor fusion **competitiva**, nonché la più comune, non è da intendersi in senso di antagonismo tra le varie sorgenti di informazione ma relativa al fatto che esse producono il medesimo risultato. Possiamo distinguere due possibili configurazioni competitive: la fusione di dati provenienti da sensori diversi o la fusione di misurazioni da un singolo sensore prese in istanti diversi. La configurazione competitiva è anche chiamata configurazione *ridondante*, ad esempio è possibile avere più trasduttori di temperatura che misurano la temperatura di un di un processo per andare a migliorare la robustezza, l’affidabilità e l’accuratezza della misura. I sensori S_1 e S_2 in Figura 1.3 rappresentano una configurazione competitiva, in cui entrambi i sensori osservano in modo ridondante la stessa proprietà di un oggetto nello spazio ambientale [4];
- Un approccio intermedio è quello cosiddetto **complementare** in cui si hanno diversi tipi di sensori che misurano diversi oggetti ma, a differenza della fusione cooperativa, non si vuole ottenere qualcosa che non è ottenibile a livello individuale ma semplicemente un’informazione più completa, ottenuta mediante la fusione di più sistemi. In generale, è facile fondere dati complementari, poiché i dati provenienti da sensori indipendenti possono essere aggiunti l’uno all’altro. I sensori S_2 e S_3 in Figura 1.3 rappresentano una configurazione complementare, poiché ogni sensore osserva una parte diversa dello spazio ambientale [4].

Queste tre categorie di configurazione non sono mutuamente esclusive. Molte applicazioni implementano aspetti di più di uno dei tre tipi. Un esempio di architettura ibrida è l’applicazione di più telecamere per monitorare una determinata area. Nelle regioni coperte da due o più telecamere, la configurazione può essere competitiva o cooperativa, mentre per le regioni osservate da una sola telecamera, la configurazione è complementare [4].

1.2.2 Classificazione in base al livello di astrazione

In termini di elaborazione dei dati, la fusione multisensoriale è tipicamente implementata a tre diversi livelli di astrazione [6]:

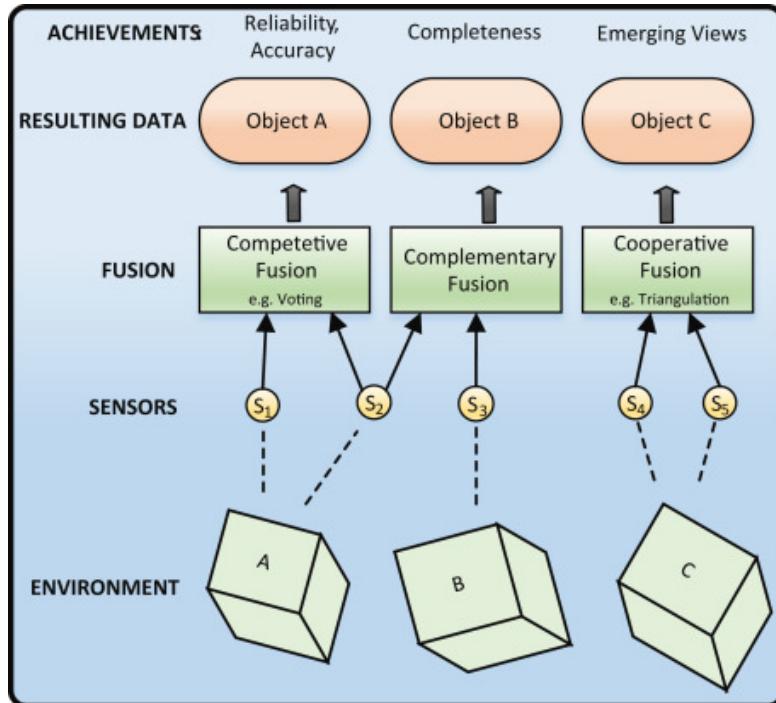


Figura 1.3: Fusione competitiva, complementare e cooperativa [4]

- Livello **decisionale**: in questo caso la sensor fusion viene utilizzata per valutare delle ipotesi, quindi non si parla in una fusione propriamente di sensori ma di fusione di informazioni. Le informazioni, che sono in realtà complesse e non necessariamente solo numeri, vengono confrontate tra loro, quindi si tenta di capire quale approccio seguire a seconda dell'affidabilità delle varie informazioni [5];
- Livello **dati**: a questo livello la sensor fusion elabora i dati grezzi acquisiti da diverse fonti [6]. Non vi è la necessità di prendere decisioni, dunque la fusione è necessaria solo per poter rappresentare i dati misurati da più sensori sotto forma di un unico dato;
- Livello di **funzionalità**: in questo caso i dati non vengono trattati come singole grandezze ma come lo stato di un'entità, si parla quindi di *feature* estratte dai dati. Ad esempio, il comfort di un veicolo è una feature derivata dai n sensori che possono essere combinati. In questo caso è possibile avere più fonti di informazione e combinarle in un'unica funzione.

Anche in questo caso, è possibile avere una combinazione ibrida, lavorando a più livelli di astrazione contemporaneamente.

1.2.3 Classificazione in base all'approccio

Un altro modo per classificare le tecniche di fusione multisensoriali è quello basato sul tipo di approccio che esse adottano. Le più tradizionali sono basati su un *approccio*

probabilistico mentre vi sono altre tecniche che sono basate su approcci *non probabilistici*. Le tecniche basate su approcci probabilistici sono basate sull'idea che ogni misura può essere vista come una distribuzione di probabilità. Gli altri approcci, invece, sono legati a diverse teorie non probabilistiche, due principali sono la *teoria della possibilità* e l'*evidential belief* [5]. Le tecniche più tradizionali, e più diffuse, sono quelle probabilistiche, tra le principali vi sono:

- Tecniche basate sul teorema del limite centrale;
- Tecniche basate sugli algoritmi di Bayes;
- Filtri di Kalman.

Tra queste, la tecnica più diffusa è sicuramente il filtro di Kalman. Il teorema del limite centrale e gli algoritmi di Bayes possono essere visti come fondamenti teorici che portano al filtro di Kalman. Le tecniche non probabilistiche, invece, sono molto diffuse per quanto riguarda l'implementazione a livello decisionale [5].

1.3 Principali tecniche di fusione multisensoriale

In letteratura vi sono numerosi algoritmi di fusione, di seguito verranno brevemente discussi i principali.

1.3.1 Algoritmi basati sul teorema del limite centrale

La classe di algoritmi più semplici, ma non per questo meno utili, comprende quelli basati sul **teorema del limite centrale** (TLC). Esso afferma che la somma n di variabili indipendenti aventi identica distribuzione è una variabile che si distribuisce normalmente qualsiasi sia la tipologia di distribuzione di partenza. Indicati con μ_x e σ_x la media e la deviazione standard della distribuzione di partenza, la distribuzione normale ottenuta dal teorema sarà caratterizzata dai seguenti valori di media e deviazione standard [7]:

$$\begin{aligned}\mu_{TLC} &= \mu_x \\ \sigma_{TLC} &= \frac{\sigma_x}{\sqrt{n}}\end{aligned}$$

Questo vuol dire che è possibile ridurre l'influenza del rumore semplicemente calcolando la media su un numero n di campioni provenienti dal singolo sensore. Inoltre, supponendo di avere più sensori e rumore non correlato, siamo in grado di ottenere una distribuzione approssimativamente normale, pur non sapendo nulla sulla distribuzione del singolo sensore. Come semplice esempio, supponiamo di avere un singolo accelerometro posizionato su un tavolo fisso in modo che misuri solo l'accelerazione di gravità. Se questo fosse un sensore perfetto, l'uscita sarebbe una costante di $9.81m/s^2$. Tuttavia, la misurazione effettiva sarà rumorosa - quanto rumorosa dipende dalla qualità del sensore. Quest'ultimo è ciò che si definisce un rumore imprevedibile, dunque non è possibile eliminarlo tramite la calibrazione. Aggiungendo però un

secondo accelerometro e facendo la media delle due letture sarà possibile ridurre il rumore complessivo nel segnale. È inoltre possibile fondere insieme i sensori, a condizione che il rumore non sia correlato tra essi, ottenendo una riduzione del rumore di un fattore pari alla radice quadrata del numero di sensori. Quindi quattro sensori identici fusi insieme avranno la metà del rumore di uno solo. In questo caso, tutto ciò che costituisce questo semplicissimo algoritmo di fusione dei sensori è una funzione di media. Questo semplice metodo funziona finché non vi è correlazione tra i rumori che influenzano i sensori. Supponendo di avere più magnetometri e di voler fondere le misure con questa tecnica, bisogna considerare che almeno una parte del rumore proviene dai campi magnetici in movimento creati dall'ambiente circostante. Ciò significa che ogni magnetometro sarà influenzato da questa fonte di rumore correlata e quindi la media dei sensori non la rimuoverà.

1.3.2 Algoritmi basati su reti bayesiane

Una **rete Bayesiana** (BN) è un modello grafico probabilistico che rappresenta un insieme di variabili con le loro dipendenze condizionali attraverso l'uso di un grafo aciclico diretto (DAG). La regola di Bayes, che si occupa della probabilità, è alla base dell'equazione di aggiornamento che combina i modelli di movimento e di misurazione. Le reti bayesiane, anch'esse basate sulla regola di Bayes, prevedono la probabilità che una qualsiasi delle diverse ipotesi sia il fattore che contribuisce a un dato evento. K2, hill climbing, hill climbing iterativo e simulated annealing sono alcuni algoritmi bayesiani ben noti [8].

1.3.3 Convolutional neural network

Una **rete neurale convoluzionale** (CNN o ConvNet dall'inglese convolutional neural network) è un tipo di rete neurale artificiale feed-forward in cui il pattern di connettività tra i neuroni è ispirato dall'organizzazione della corteccia visiva animale, i cui neuroni individuali sono disposti in maniera tale da rispondere alle regioni di sovrapposizione che tassellano il campo visivo. Le reti convoluzionali sono ispirate da processi biologici e sono variazioni di percepitori multistrato progettate per usare al minimo la pre-elaborazione [9]. I metodi convoluzionali basati sulla rete neurale possono elaborare simultaneamente molti canali di dati dei sensori. Da questa fusione di dati, producono risultati di classificazione basati sul riconoscimento delle immagini. Ad esempio, un robot che utilizza i dati sensoriali per distinguere volti o segnali stradali si affida ad algoritmi convoluzionali basati su reti neurali.

1.3.4 Filtri di Kalman

Il **filtro di Kalman** è un efficiente filtro ricorsivo che valuta lo stato di un sistema dinamico a partire da una serie di misure soggette a rumore. Per le sue caratteristiche intrinseche è un filtro ottimo per rumori e disturbi agenti su sistemi gaussiani a media nulla [10]. Esso è ampiamente usato per la stima di variabili come la velocità e

l’accelerazione, anche in presenza di più sensori. Il filtro di Kalam verrà ampiamente discusso nel capitolo successivo.

Capitolo 2

Il filtro di Kalman

Il **filtro di Kalman** è un efficiente filtro ricorsivo che valuta lo stato di un sistema dinamico a partire da una serie di misure soggette a rumore. Per le sue caratteristiche intrinseche è un filtro ottimo per rumori e disturbi agenti su sistemi gaussiani a media nulla.

Il filtro prende il nome da Rudolf E. Kalman, sebbene in realtà Thorvald Nicolai Thiele e Peter Swerling abbiano sviluppato un algoritmo simile in precedenza. Stanley Schmidt è generalmente riconosciuto essere stato il primo a sviluppare una realizzazione pratica di un filtro di Kalman. Durante una visita di Kalman al Centro di Ricerche Ames della NASA Schmidt vide l'applicabilità delle idee di Kalman al problema della stima delle traiettorie per il programma Apollo e finì con l'includerlo nel programma del computer di bordo dell'Apollo. Il filtro fu sviluppato in articoli scientifici da Swerling (1958), Kalman (1960) e Kalman e Bucy (1961) [10]. Perchè il filtro funzioni sono necessarie due ipotesi semplificative [11]:

- Il modello del sistema e il modello di osservazioni sono lineari;
- Il rumore che agisce sul sistema è di tipo Gaussiano.

Viene definito filtro, pur essendo di fatto un estimatore e non un filtro nel senso stretto del termine, in quanto è in grado, in senso lato, di filtrare il rumore di misura. Inoltre, presenta il grande vantaggio di essere un algoritmo e non un qualcosa di fisico, ciò rende possibile la sua implementazione praticamente ovunque ci sia un'unità di elaborazione. Il filtro è basato sul modello del sistema fisico a cui è applicato, ciò consente di andare a filtrare sia il rumore di misura sia il rumore dovuto alle approssimazioni del modello matematico del sistema, definito come rumore sul processo. Una definizione più immediata del filtro di Kalman può essere la seguente:

È un processo matematico iterativo che utilizza una serie di equazioni e input di dati consecutivi per stimare rapidamente il vero valore di variabili come posizione, velocità, ecc. dell'oggetto misurato, nonostante i valori misurati contengono errori, incertezze o variazioni imprevedibili e/o casuali [12].

La versione standard del filtro di Kalman, come affermato in precedenza, può essere applicata solo su sistemi lineari, per superare questa limitazione sono state sviluppate altre versioni del filtro come il **filtro di Kalman esteso** (EKF - Extended Kalman filter) e l'**Unscented Kalman filter** (UKF) [13].

2.1 Filtro di Kalman per un sistema ad una sola variabile

Per comprendere a pieno il funzionamento del filtro esaminiamo la sua applicazione in un sistema ad una variabile, il cui funzionamento è schematizzato in Figura 2.1 [12].

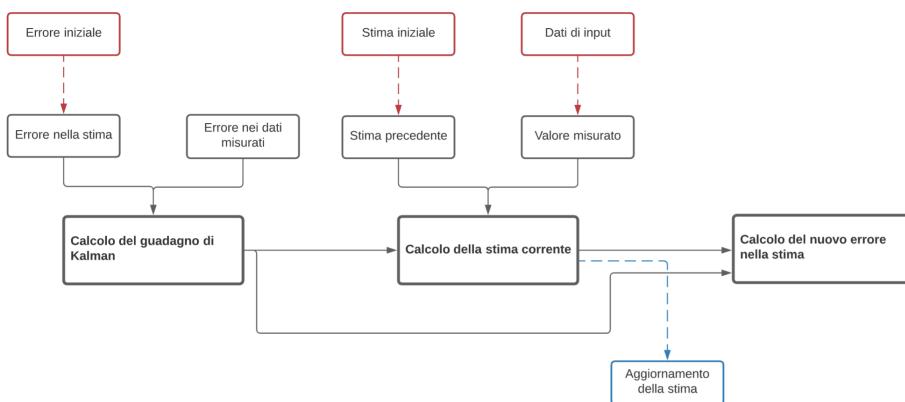


Figura 2.1: Schematizzazione del filtro di Kalman per una variabile

Il processo che sta alla base della stima può essere riassunto in tre passaggi principali:

- Calcolo del **guadagno di Kalman**;
- Calcolo della nuova stima;
- Calcolo del nuovo errore sulla stima.

Il guadagno di Kalman (KG) è un parametro fondamentale del filtro, esso stabilisce il peso assegnato alla stima corrente e ai dati di misura, andando ad indicare quanto ciascuno dei due valori andrà ad incidere sulla nuova stima. Il guadagno di Kalman viene calcolato in funzione degli errori di misura e di stima: se l'errore sulla stima è alto i dati misurati incideranno maggiormente sulla nuova stima viceversa, i dati stimati avranno maggior incidenza. Se definiamo ϵ_{EST} l'errore sulla stima, o l'errore iniziale nel caso in cui sia la prima iterazione, e ϵ_{MEA} l'errore sulla misura, possiamo calcolare il guadagno di Kalman con l'Equazione 2.1:

$$KG = \frac{\epsilon_{EST}}{\epsilon_{EST} + \epsilon_{MEA}} \quad (2.1)$$

T	MEA	ε_{MEA}	EST	KG	ε_{EST_t}
t-1			68		2
t	75	4	70.33	0.33	1.33
t+1	71	4	70.5	0.25	1.00
t+2	70	4	70.4	0.2	0.80
t+3	74	4	71	0.17	0.66

Tabella 2.1: Applicazione del filtro di Kalman per la stima della temperatura

Dunque esso sarà un valore compreso tra 0 e 1. Il secondo passaggio consiste nel calcolare la stima corrente, indicata con EST_t . Essa richiede il Kalman gain, la stima precedente, indicata con EST_{t-1} , e i dati misurati, indicati con MEA . Essa si ottiene dall’Equazione 2.2:

$$EST_t = EST_{t-1} + KG[MEA - EST_{t-1}] \quad (2.2)$$

L’ultimo passaggio è quello di calcolare il nuovo errore sulla stima utilizzando sia il guadagno di Kalman che la nuova stima utilizzando l’Equazione 2.3:

$$\varepsilon_{EST_t} = \frac{(\varepsilon_{MEA}) \cdot (\varepsilon_{EST_{t-1}})}{(\varepsilon_{MEA}) + (\varepsilon_{EST_{t-1}})} \rightarrow \varepsilon_{EST_t} = [1 - KG] \cdot (\varepsilon_{EST_{t-1}}) \quad (2.3)$$

L’errore della nuova stima sarà sempre minore della precedente, in quanto moltiplicato per il valore $1 - KG \leq 1$. L’unica differenza, in funzione del valore di KG, sta nella velocità con la quale l’errore sulla stima tende a 0. Se l’errore di misura è basso, il guadagno di Kalman tenderà ad 1 e l’errore convergerà velocemente a 0. Viceversa, avendo dei dati di misura poco accurati, il processo di convergenza sarà più lento.

2.1.1 Esempio pratico

Il modello appena descritto può essere applicato a sistemi aventi una sola variabile di stato, un esempio potrebbe essere quello di un sistema il cui compito è misurare la temperatura. Supponiamo di avere un termometro avente un’incertezza di 4 gradi ($\varepsilon_{MEA} = 4$) e di voler migliorare l’accuratezza della misura mediante un filtro di Kalman. Supponiamo che il termometro misuri, nel momento in cui iniziamo ad usare il filtro, 75 gradi, che la nostra stima iniziale sia di 68 gradi con un errore di $\varepsilon_{EST} = 2$ e che il valore reale di temperatura sia di 72 gradi. Applicando il filtro si ottiene quanto segue.

Come è possibile osservare dalla Tabella 2.1 dopo solo 3 iterazioni la stima calcolata dal filtro si discosta di solo un grado rispetto al valore reale. Questo è il grande vantaggio del filtro di Kalman, esso è in grado di convergere velocemente al valore reale. Continuando con le iterazioni il valore stimato sarà prossimo a quello reale, come mostrato nel grafico in Figura 2.2, dove sono state effettuate 100 misurazioni.

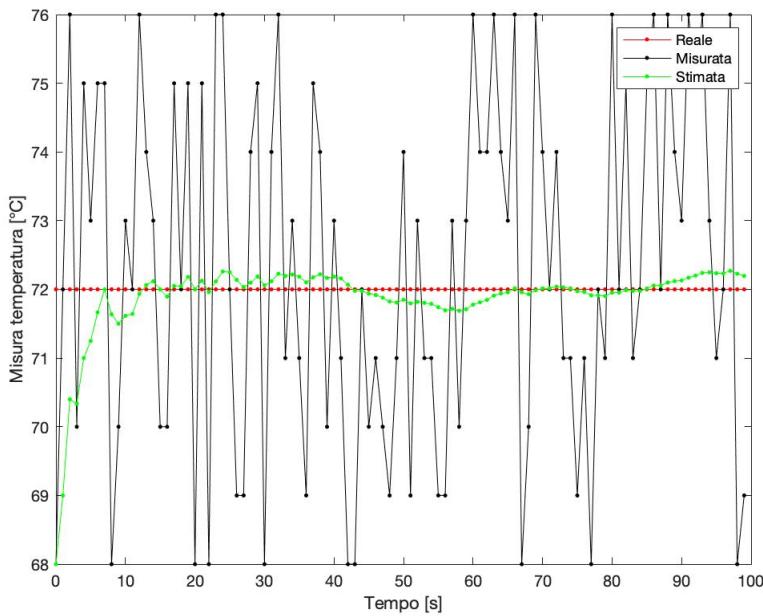


Figura 2.2: Applicazione del filtro per la stima della temperatura

2.2 Filtri tempo continuo e tempo discreto

Prima di procedere con la descrizione del modello multidimensionale del filtro di Kalman, è opportuno fare distinzione tra due possibili implementazioni del filtro:

- Filtro tempo continuo;
- Filtro tempo discreto.

Il filtro di Kalman tempo continuo presenta tutti i parametri che lo caratterizzano come funzioni tempo continuo, definite per ogni istante di tempo. Tuttavia, tutti i sistemi digitali e i sensori adoperati sono tempo discreto, si pensi per esempio ad un sensore che fornisce una misura ogni ΔT istanti di tempo. Per questo si effettua una discretizzazione del sistema e si applica il filtro di Kalman tempo discreto, che opera su istanti di tempo ben definiti, e può essere applicato sui sistemi digitali.

Tutto ciò che segue sarà riferito ai filtri di Kalman tempo discreto.

2.3 Modello multidimensionale

Il modello visto in precedenza, per quanto esso sia perfettamente funzionante, è applicabile solo ad una cerchia ristretta di sistemi. Nella maggior parte dei casi, primo tra tutti lo sviluppo di vetture a guida autonoma, è necessario stimare più variabili contemporaneamente e, se necessario, combinarle tra loro per ottenere l'informazione richiesta. Prendendo come esempio la guida autonoma, per conoscere la posizione del veicolo è necessario tenere traccia non solo della posizione attuale ma di ulteriori

variabili come la velocità, l'accelerazione e l'angolo di sterzata. Il modello generale del filtro di Kalman, mediante un approccio matriciale consente l'applicazioni a sistemi aventi n variabili di stato. In generale, diciamo che il filtro di Kalman affronta il problema di riuscire a stimare lo stato $x \in \mathcal{R}^n$ di un sistema tempo discreto governato da equazioni differenziali lineari [14]. Lo stato del sistema è definito da due parametri fondamentali del filtro:

- Il **vettore di stato** x_k , con $x \in \mathcal{R}^n$, che contiene tutte le variabili di stato del sistema che il filtro andrà a stimare;
- La **matrice di covarianza dello stato** P_k , essa indica l'accuratezza del valore della stima del vettore di stato all'iterazione k . Essa è detta anche matrice di incertezza della stima.

Possiamo schematizzare il funzionamento del filtro in due fasi, alle quali si aggiunge un ulteriore fase iniziale:

- Fase di init: in questa fase, chiamata anche di inizializzazione, vengono inizializzati il vettore di stato x e la matrice di covarianza P ;
- Fase di predict: viene stimato un valore futuro del vettore di stato sulla base della stima corrente e della matrice di covarianza;
- Fase di update: in questa fase, detta anche fase di correzione, viene aggiornata la stima del vettore di stato prendendo in considerazione anche i valori provenienti dai sensori.

La fase di init viene eseguita una sola volta prima che il filtro inizi ad operare, successivamente si vanno ad alternare le fasi di predict e di update.

2.3.1 Ipotesi

Il filtro di Kalman si basa su alcune ipotesi che risultano verificate nella maggior parte dei casi. Nel caso in cui una o più ipotesi non dovesse essere soddisfatta sono stati sviluppati estensioni del filtro che consentono comunque il suo impiego [15]. Saranno assunte valide le seguenti ipotesi:

- Il rumore di qualsiasi natura, sia esso di processo o di misura, viene modellizzato con funzioni normali a media nulla, dunque di tipo Gaussiano;
- Il rumore di processo (Q) e il rumore di misura (R) sono tra loro indipendenti;
- Il rumore, di qualunque natura (Q e R), e le variabili di stato (x) sono tra loro indipendenti;
- I processi stocastici Q e R sono incorrelati con l'ipotesi iniziali del vettore di stato e della matrice di covarianza.

2.3.2 Fase di init

In questa fase viene inizializzato il filtro andando a definire il vettore di stato iniziale, indicato con x_0 e la matrice di incertezza sulla stima relativa ad esso, indicata con P_0 . Questi valori saranno usati alla prima iterazione del filtro. La scelta di x_0 influenza solo il tempo necessario al filtro a convergere al valore reale, un valore più distante da quello veritiero farà sì che il filtro impieghi più tempo per stimare un valore vicino a quello reale. La scelta di P_0 , invece, è molto più importante. Essa dà informazioni su quanto il valore di x_0 sia preciso: scegliere un valore prossimo allo zero vorrà dire assumere di conoscere con molta precisione lo stato iniziale, viceversa, un valore alto farà sì che il filtro dia molta importanza ai valori misurati, supponendo una stima poco precisa del valore iniziale del vettore di stato [16].

2.3.3 Fase di predict

Questa fase si divide a sua volta in due passaggi fondamentali: la predizione dello stato e la predizione della stima.

Predizione dello stato

La predizione dello stato, o state extrapolation, consiste in un'equazione in grado di stimare lo stato futuro, indicato con x_{k+1} , a partire dallo stato attuale (x_k) e dal modello fisico del sistema. L'equazione di state extrapolation è la seguente:

$$x_{k+1} = A_k x_k + B u_k + w_k \quad (2.4)$$

Dove:

- w_k : è il rumore che agisce sul processo, schematizzato come un rumore bianco con distribuzione normale a media nulla e varianza data dalla matrice Q :

$$p(w) \sim N(0, Q); \quad (2.5)$$

- u_k : è la matrice delle variabili di input o di controllo, con $u \in \mathcal{R}^l$. Essa include tutti quei parametri di input che agiscono sul sistema come, ad esempio, l'accelerazione;
- A: è la matrice di transizione di stato, descrivere i cambiamenti di stato nel sistema. È una matrice $n \times n$. La matrice A mette in relazione lo stato precedente con quello attuale in assenza di rumore. A seconda della nomenclatura, essa può essere indicata come matrice F ;
- B: è la matrice di controllo, ha dimensioni $n \times l$ e mette in relazione le variabili di input con lo stato del sistema. Essa modifica il valore di Ax_k tenendo conto dell'influenza degli input. A seconda della nomenclatura, essa può essere indicata come matrice G

Predizione dell'incertezza di stima

Durante la fase di predizione di stato si va ad aggiornare la matrice di incertezza di covarianza dello stato P , ottenendo così una stima dell'errore dello stato appena stimato. L'equazione con la quale questo avviene è la seguente:

$$P_{k+1} = AP_kA^T + Q \quad (2.6)$$

Dove la matrice Q rappresenta la *matrice di covarianza del rumore di processo*. In linea di principio, la matrice Q dovrebbe riflettere l'incertezza nel modello di stato assunto o qualsiasi caratteristica non considerata dal modello o non modellabili nello stato, come ad esempio l'attrito. La matrice Q insieme al valore iniziale della matrice P gioca un ruolo molto importante nel funzionamento del filtro. Il valore di Q dovrebbe essere sufficientemente piccolo da mantenere il potenziale di apprendimento dalla misurazione ma non troppo grande per non far aumentare l'incertezza, rendendo di fatto inutile il processo di stima [16]. La matrice Q è diagonale se il rumore non è correlato tra i vari stati del processo, viceversa essa non sarà più diagonale. Una delle parti più complesse dell'applicazione del filtro è proprio il calcolo della matrice Q . Se il processo ha variabili di input un modo per calcolare la matrice è il seguente:

$$Q = B\sigma^2B^T \quad (2.7)$$

Tuttavia, attraverso una fase di tuning, bisogna andare a ricavare il parametro σ^2 . Bisogna, inoltre, tener presente che valori molto bassi della matrice Q causano del ritardo (lag) nella stima, viceversa, valori alti fanno sì che la stima segua il valore delle misure, senza attenuarne il rumore [15].

2.3.4 Fase di update

Durante la fase di update vengono letti i dati sensori, calcolato il guadagno di Kalman e aggiornato lo stato di conseguenza. Avviene dunque la correzione del valore di x stimato nella fase di predict tenendo conto dei nuovi dati di input.

Misurazioni

I dati di input corrispondenti alle misure alla k -esima iterazione, indicati con y_k con $y \in \mathcal{R}^m$ sono dati dalla seguente equazione:

$$y_k = Hx_k + z_k \quad (2.8)$$

Dove:

- H : è la matrice di misurazione, ha dimensioni $m \times n$, rappresenta il modello di misurazione, mettendo in relazione lo stato con la misurazione;
- z_k : è il rumore di misurazione, una sorgente di rumore bianco con distribuzione normale a media nulla e varianza data dalla matrice R , discussa in seguito:

$$p(z) \sim N(0, R). \quad (2.9)$$

Guadagno di Kalman

Nel modello multidimensionale è possibile calcolare il guadagno di Kalman mediante l'equazione:

$$K_k = \frac{P_k H^T}{H P_k H^T + R} \quad (2.10)$$

R rappresenta la matrice di covarianza del rumore di misura, essa indica l'errore intrinseco della misurazione. Se le misure sono indipendenti tra loro la matrice R è diagonale. Essa si ottiene mantenendo lo stato del sistema costante e sottraendo alle misurazioni la loro media, ottenendo così il rumore. In termini analitici, ciò vuol dire calcolare la covarianza dei segnali in gioco:

$$R_k = E(y_k y_k^T) \quad (2.11)$$

Supponendo di avere due grandezze misurate, x e y , calcolandone la covarianza si ottiene una matrice R così fata:

$$R = \begin{bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 \end{bmatrix} \quad (2.12)$$

Sulla diagonale principale ci sarà la varianza dei segnali, su quella secondaria la covarianza dei segnali [15].

Correzione dello stato

A questo punto è possibile calcolare la stima dello stato corrente, andando a correggere quella precedentemente stimata mediante le informazioni acquisite e il guadagno di Kalman:

$$x_k = x_k + K_k [y_k - H x_k] \quad (2.13)$$

Matrice di covarianza dello stato

L'ultimo passaggio del filtro è quello di aggiornare la matrice di covarianza dello stato secondo i valori dell'iterazione corrente:

$$P_k = (I - K_k H) P_k \quad (2.14)$$

Dove I è una matrice identità delle stesse dimensioni di H .

Lo stato corrente diventa lo stato precedente

Alla prossima iterazione, il vettore di stato x_k e la matrice P_k diventeranno input per la prossima iterazione, assumendo il ruolo di stato precedente [17]:

$$k \rightarrow k - 1 \quad (2.15)$$

2.4 Filtro di Kalman Esteso

Il **filtro di Kalman esteso** (EKF - Extended Kalman filter) consente di eliminare il vincolo sulla linearità del sistema e sulla distribuzione Gaussiana del rumore, rendendo meno rigidi i requisiti rispetto alla sua versione base, seppur diminuendo la qualità delle stime che resta comunque più che valida. La densità di probabilità è approssimata da una gaussiana, che può distorcere la vera struttura e, a volte, potrebbe portare alla divergenza tra la previsione del filtro e le misurazioni [16]. L'EKF richiede che le funzioni di stato e di misura siano funzioni *differenziabili* [17], in modo tale che sia possibile effettuare la linearizzazione mediante l'*espansione di Taylor del primo ordine*. L'espansione di una funzione f in a è data dalla formula:

$$f(x) = f(a) + f'(a)(x - a) \quad (2.16)$$

Come detto precedentemente, l'EKF è un approssimatore lineare del primo ordine per i sistemi non lineari. Un generico processo stocastico è descritto da un sistema discreto:

$$\begin{cases} x_{k+1} &= f(x_k, u_k, v_k) \\ y_{k+1} &= h(x_k, w_k) \end{cases} \quad (2.17)$$

Dove x_k , con $x \in \mathcal{R}^n$, è sempre il vettore di stato e y_k , con $y \in \mathcal{R}^m$, è il vettore contenente le misurazioni. Le funzioni f e h sono rispettivamente la funzione di stato e la funzione di osservazione, mentre i termini v_k e w_k rappresentano il rumore di processo e di misura, analogamente a quanto avveniva nella versione base del filtro. La linearizzazione avviene mediante le seguenti equazioni:

$$A_{i,j} = \frac{\partial f_i}{\partial x_j}, \quad G_{i,j} = \frac{\partial h_i}{\partial y_j} \quad (2.18)$$

Dove A e G sono le matrici Jacobiane della funzione di stato e di osservazione. Il processo di linearizzazione approssima f e h a funzioni lineari tangenti alle stesse, quindi le meccaniche di propagazione sono equivalenti a quelle del filtro di Kalman per sistemi lineari [17].

2.4.1 Svantaggi

A differenza della sua controparte lineare, il filtro di Kalman esteso in generale non è uno stimatore ottimo (è ottimale se la misurazione e il modello di transizione di stato sono entrambi lineari, poiché in quel caso il filtro di Kalman esteso è identico a quello normale). Inoltre, se la stima iniziale dello stato è errata, o se il processo è modellato in modo errato, il filtro può divergere rapidamente, a causa della sua linearizzazione. Un altro problema con il filtro di Kalman esteso è che la matrice di covarianza stimata tende a sottostimare la vera matrice di covarianza e quindi rischia di diventare inconsistente in senso statistico senza l'aggiunta di "rumore stabilizzante".

Detto questo, il filtro Kalman esteso può fornire prestazioni ragionevoli ed è probabilmente lo standard *de facto* nei sistemi di navigazione e GPS [18].

2.5 Esempio: moto in 2 dimensioni

Andiamo ora a vedere come applicare il filtro di Kalman per stimare la posizione di un oggetto lanciato in aria con una certa velocità iniziale, supponendo che esso percorra una traiettoria bidimensionale. Le equazioni che descrivono il movimento dell'oggetto sono le seguenti:

$$\begin{aligned} p_x &= p_{0x} + v_{0x}t \\ v_x &= v_{0x} \\ p_y &= p_{0y} + v_{0y}t + \frac{1}{2}gt^2 \\ v_y &= v_{0y} + gt \end{aligned} \tag{2.19}$$

Con $g = -9.81m/s^2$. Supponiamo che l'oggetto venga lanciato con velocità iniziale $v_0 = 100m/s$ con un angolo $\theta = 30$ da una posizione iniziale che coincide con l'origine degli assi del nostro sistema di riferimento. Avremo quindi $p_{0x} = p_{0y} = 0$ e $v_{0x} = v_0 \cos \theta, v_{0y} = v_0 \sin \theta$. Da queste equazioni è possibile ricavare il modello del sistema andando a separare i termini legati ai parametri di input, che andranno nella matrice B , dagli altri che invece andranno nella matrice A . In definitiva, si tratta di scrivere il sistema di equazioni in forma matriciale, separando i termini di input ed inserendo i coefficienti nelle rispettive matrici. Si noti che la variabile t viene sostituita con Δt in quanto si tratta di un filtro di Kalman tempo discreto. Il sistema risultante è il seguente:

$$A = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{2}\Delta t^2 \\ \Delta t; \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad u = g \tag{2.20}$$

Il vettore di stato sarà composto come segue:

$$x = \begin{bmatrix} p_x \\ v_x \\ p_y \\ v_y \end{bmatrix} \tag{2.21}$$

Simulando il rumore di misura con una deviazione standard di $\sigma = 20$, possiamo inserire come matrice R , che date le dimensioni del sistema avrà dimensioni 2×2 , come segue:

$$R = \begin{bmatrix} \left(\frac{\sigma}{2}\right)^2 & 0 \\ 0 & \left(\frac{\sigma}{2}\right)^2 \end{bmatrix} = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix} \tag{2.22}$$

Per quanto riguarda la matrice Q è possibile calcolarla con la formula:

$$Q = B\sigma^2 B^T \tag{2.23}$$

Dove $\sigma = 0.05$. Il risultato è una matrice di dimensioni 4×4 . I parametri vengono invece inizializzati come segue:

$$P_0 = Q, \quad x_0 = \begin{bmatrix} p_{0x} \\ v_{0x} \\ p_{0y} \\ v_{0y} \end{bmatrix} \tag{2.24}$$

Nel grafico di Figura 2.3 è possibile vedere il risultato della simulazione. Nei grafici di Figura 2.4 e Figura 2.5 è possibile vedere il dettaglio della stima sull'asse x e y.

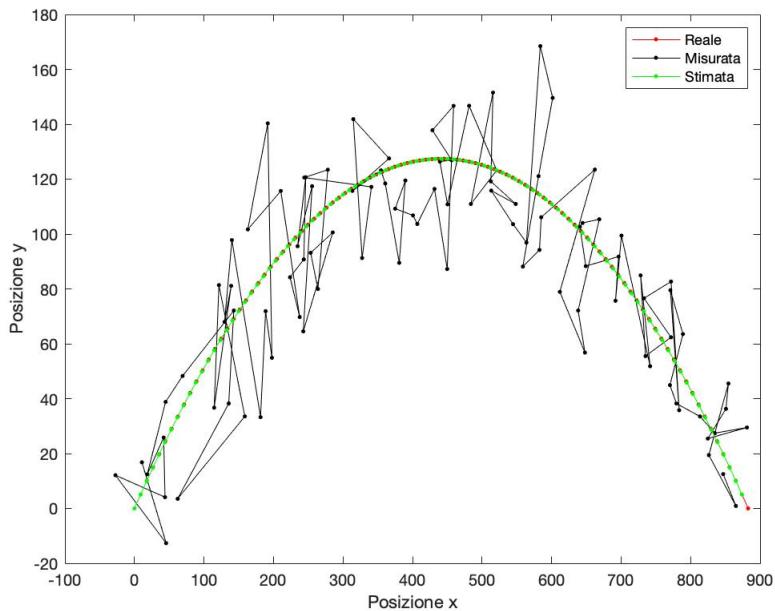


Figura 2.3: Applicazione del filtro per la stima della traiettoria di un oggetto in 2D

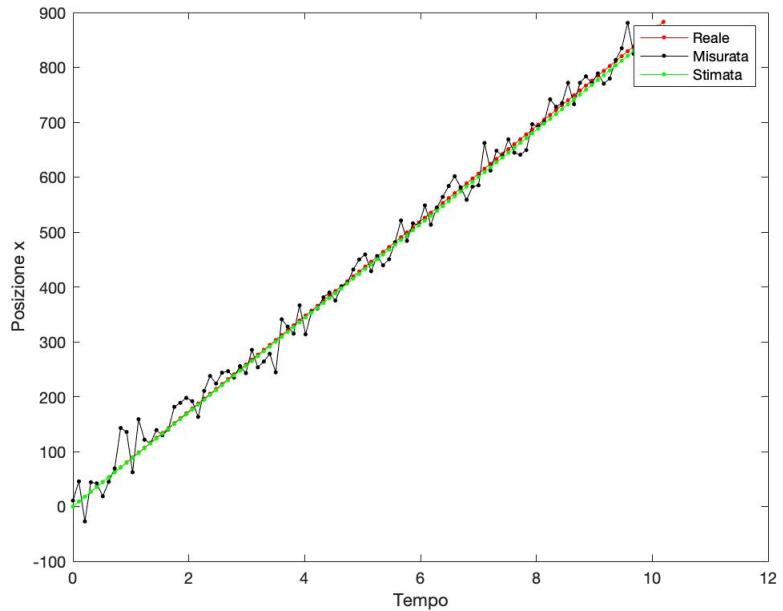


Figura 2.4: Dettaglio asse x

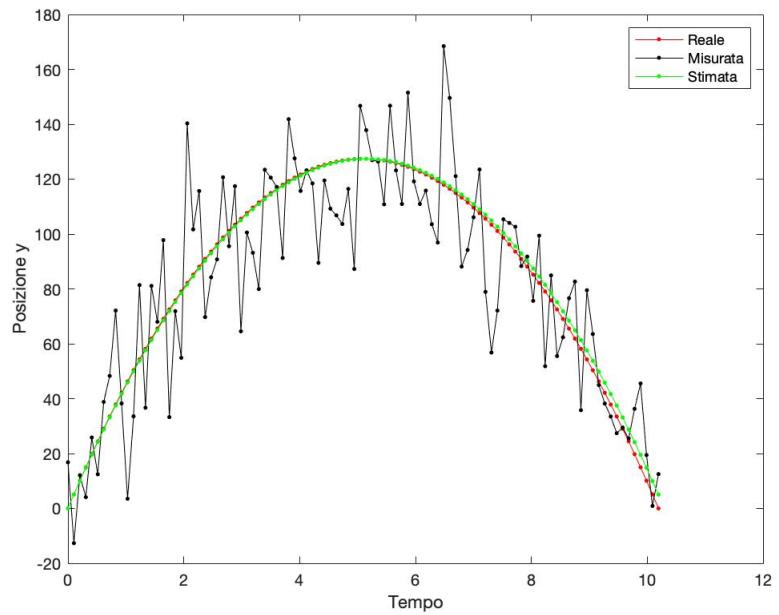


Figura 2.5: Dettaglio asse y

Capitolo 3

Modelli sperimentali

Al fine di verificare quanto descritto nei capitoli precedenti, e di porre le basi per lo sviluppo di un sistema basato sul filtro di Kalman in ambito automotive, sono state svolte delle simulazioni utilizzando il software Matlab. Scopo principale è implementare un filtro di Kalman per stimare la velocità e la posizione di un veicolo a guida autonoma. Sono stati presi in considerazione tre approcci differenti. Il primo, e il più semplice, considera il veicolo come un punto materiale e utilizza il pacchetto software di Matlab per definire il percorso e i sensori a bordo della macchina, con relativi parametri di rumore. Gli altri due approcci prevedono, invece, un modello più complesso per quanto riguarda il veicolo e, rispettivamente, utilizzano un filtro di Kalman e un filtro di Kalman esteso che combina due sensori. Inoltre, viene considerata la matrice Q sia nel caso standard, sia nel caso in cui essa sia una matrice tempo-variante. Di seguito vengono discussi nel dettaglio gli approcci utilizzati.

3.1 Modello del punto materiale

Un primo approccio possibile consiste nel considerare la vettura come un punto materiale, utilizzando le classiche equazioni della cinematica per ricavare il modello del filtro. Tramite il software “driving scenario designer”, la cui schermata principale è mostrata in Figura 3.1, è possibile costruire il percorso ed indicare la traiettoria, mediante l’inserimento di *waypoints*, che la vettura dovrà percorrere. È possibile inserire nell’auto simulata vari sensori, tra cui l’unità di navigazione inerziale (INS) utilizzata in quest’esempio, andando a impostare i parametri di errore. Il programma consente poi di far seguire la traiettoria all’auto producendo i dati dei sensori inseriti, che possono essere esportati ed utilizzati nell’ambiente Matlab. È, inoltre, possibile impostare vari parametri di simulazioni, tra cui il *passo di simulazione*. Dopo la simulazione, i dati vengono usati come input per il filtro di Kalman che va a stimare la velocità e la posizione del veicolo. Il risultato viene poi confrontato sia con la traiettoria reale, sia con i dati misurati affetti da rumore. Il filtro è implementato programmaticamente tramite uno script Matlab, dopo una prima fase di creazione della struttura dati e inizializzazione dei parametri si eseguono le equazioni del filtro all’interno di un ciclo.

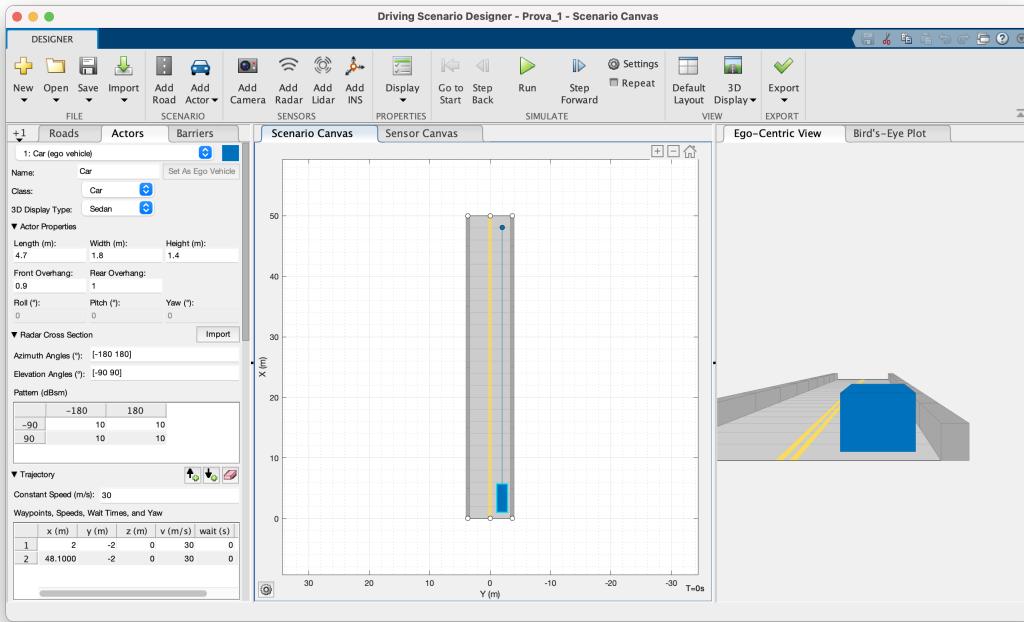


Figura 3.1: Schermata principale del pacchetto Driving Scenario Designer di Matlab

Il modello utilizzato per il filtro è il seguente:

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \quad (3.1)$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad u = \begin{bmatrix} a_x \\ a_y \end{bmatrix} \quad (3.2)$$

Dove Δt rappresenta il passo di simulazione, in questo caso esso è pari a $\Delta t = 10ms$. Per quanto riguarda la matrice R e la matrice Q , esse sono state calcolate come segue:

$$R = \begin{bmatrix} 0.2 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0.2 \end{bmatrix}, \quad Q = B\sigma^2 B^T \quad (3.3)$$

Dove 0.2 è l'accuratezza del sensore utilizzato e $\sigma = 0.05$. La matrice P , invece, viene inizializzata come matrice identità di dimensioni 4×4 . Si è assunto di conoscere con estrema accuratezza i valori di accelerazione, che vengono utilizzati come variabili di input. Il principale vantaggio di quest'approccio sta nel fatto di poter cambiare con facilità il modello del filtro, semplicemente agendo sulle matrici, senza dover passare da altri pacchetti come Simulink. Tuttavia, utilizzando il software proposto da Matlab,

vi sono alcune limitazioni sul tipo di sensori che è possibile inserire, e sul modello utilizzato per l'auto vettura che non può essere modificato. Inoltre, allo stato attuale, il software presenta alcune problematiche nella gestione della simulazione, a seconda del tipo di tracciato inserito esso genera dati incoerenti e non veritieri. Dunque possiamo considerarlo un buon punto di partenza ma sicuramente non adatto alla gestione di simulazioni più complesse.

3.1.1 Codice di simulazione

Prima di procedere nell'analisi dello script, è importante notare che esso funziona solo dopo aver esportato i dati dal Driving Scenario Designer. Una volta disegnato il tracciato, inseriti i sensori e la traiettoria che l'auto dovrà compiere, sarà sufficiente eseguire la simulazione e, una volta terminata, esportare i dati prodotti dai sensori mediante l'apposito tasto, inserendo come nome *INS_1*. Al fine di conservare i dati e averli sempre a disposizione insieme allo script è possibile salvare l'intero contenuto della struttura dati esportata nel workspace come file *.mat* che sarà poi richiamato nello script. Inoltre, ai valori misurati, viene sottratto un offset corrispondente alla posizione del sensore. Questo perché, se il sensore non si trova nel punto centrale della macchina, le sue misurazioni saranno spostate rispetto al centro e non saranno più coerenti con la traiettoria reale. Per risolvere il problema è sufficiente inserire nella apposite variabili la posizione del sensore sull'auto. Nella prima parte, dopo aver pulito il workspace, si vanno a caricare i dati simulati e ad inizializzare tutte le strutture dati necessarie, comprese quelle relative alle condizioni iniziali. Successivamente si vanno a definire tutte le matrici e i parametri del filtro di Kalman, andando poi ad inizializzare i vettori che verranno riempiti dal filtro. Filtro che viene eseguito all'interno di un ciclo, che va da 1 a *n*, dove *n* indica il numero dei punti della traiettoria, il cui valore viene ricavato nella fase iniziale. Come ultimo passaggio si costruisce un grafico contenente i valori reali della traiettoria, quelli stimati dal filtro e quelli misurati. Di seguito viene mostrato il cuore del codice sorgente, il ciclo che implementa il filtro di Kalman:

```

1 %Ciclo filtro di Kalman
2 for i = 1:n-1
3     u = [misurati(i).Acceleration(1); misurati(i).Acceleration(2)
4         ];
5     Y = [misurati(i).Position(1)-offset_x; misurati(i).Position
6         (2)-offset_y; misurati(i).Velocity(1); misurati(i).Velocity(2)
7         ];
8     p_est = A * p_est + B * u;
9     kalman(i+1).P = A * kalman(i).P * A' + Q;
10    kalman(i).K = kalman(i+1).P*H'*inv(H*kalman(i+1).P*H'+R*eye(
11        length(A)));
12    p_est = p_est + kalman(i).K * (Y - H*p_est);

```

```

9    kalman(i+1).P = ( eye(length(A)) - kalman(i).K*H ) * kalman(i
+1).P;

10

11    stimata.x = [stimata.x; p_est(1)];
12    stimata.y = [stimata.y; p_est(2)];
13    stimata.vx = [stimata.vx; p_est(3)];
14    stimata.vy = [stimata.vy; p_est(4)];

15

16    misurata.x = [misurata.x; Y(1)];
17    misurata.y = [misurata.y; Y(2)];
18    misurata.vx = [misurata.vx; Y(3)];
19    misurata.vy = [misurata.vy; Y(4)];

20

21    ideale.x = [ideale.x; esatti(i).Position(1)];
22    ideale.y = [ideale.y; esatti(i).Position(2)];
23    ideale.vx = [ideale.vx; esatti(i).Velocity(1)];
24    ideale.vy = [ideale.vy; esatti(i).Velocity(2)];

25 end

```

Codice 3.1: Applicazione del filtro di Kalman

Per il codice completo si rimanda all'Appendice A.

3.2 Modello Simulink

Al fine di ottenere un modello dell'auto più preciso, ne è stato utilizzato uno presente in Simulink, facente parte sempre dell'ambiente Matlab. Il filtro applicato è sempre la versione base di Kalman, ma, nonostante l'auto sia ancora trattata come un punto materiale per ricavare il modello del filtro, essa è rappresentata da un modello più dettagliato e complesso. Lo schema completo è mostrato in Figura 3.2. Si noti che nell'ambiente Simulink, le matrici A , B ed H vengono indicate rispettivamente con A , G e C . Tenendo presente che i nomi degli assi sono *East* e *North*, anzichè x e y , il veicolo è rappresentato da un modello così definito:

$$\frac{d}{dt} \begin{bmatrix} x_e(t) \\ x_n(t) \\ s(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} s(t)\cos(\theta(t)) \\ s(t)\sin(\theta(t)) \\ (P \frac{u_T(t)}{s(t)} - AC_d s(t)^2)/m \\ s(t)\tan(u_\Psi(t))/L \end{bmatrix} \quad (3.4)$$

Dove gli stati del veicolo sono:

- $x_e(t)$: posizione sull'asse East [m];

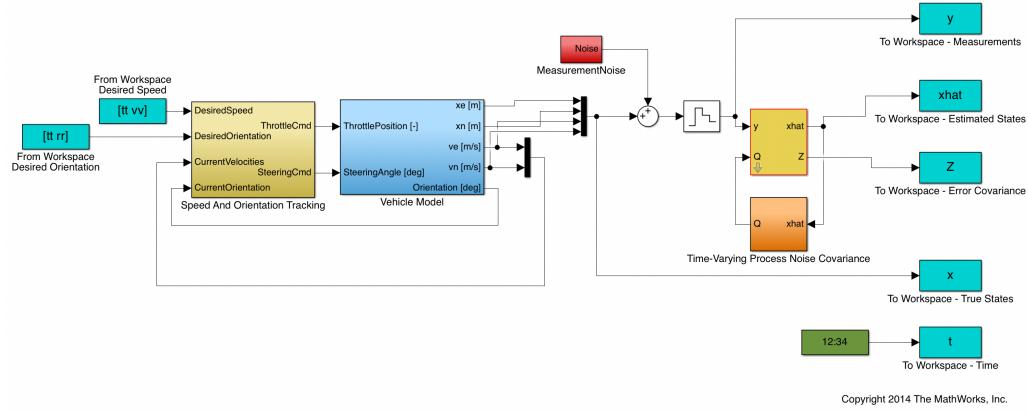


Figura 3.2: Schema completo del modello realizzato in Simulink

- $x_n(t)$: posizione sull’asse North [m];
- $s(t)$: velocità [m/s];
- $\theta(t)$: orientamento riferito all’asse East [$gradi$].

I parametri del veicolo sono:

- $P = 100000$: potenza di picco del veicolo [W];
- $A = 1$: area frontale [m^2];
- $C_d = 0.3$: coefficiente di resistenza;
- $m = 1250$: massa del veicolo [kg];
- $L = 2.5$: distanza tra l’asse anteriore e l’asse posteriore [m].

E gli input sono:

- $u_T(t)$: posizione dell’acceleratore in un range da -1 a 1;
- u_Ψ : angolo di sterzata [$gradi$].

La simulazione dell’auto è affidata a due blocchi. Il primo riceve in input la velocità e l’angolo di sterzata, sia desiderati che correnti, e dà in output i comandi dell’acceleratore e dell’angolo di sterzata. Il secondo prende in input i dati di output del primo e dà in uscita i dati relativi a posizione velocità e orientamento del veicolo. I dati in uscita sono esenti da rumore, esso viene aggiunto successivamente, tramite il blocco *MeasurementNoise*, al fine di simulare il rumore di misura. Per inserire la traiettoria è sufficiente indicare la velocità e l’angolo di sterzata desiderato, istante per istante, all’interno di vettori che devono essere salvati nel workspace. In alternativa, è possibile inserire il vettore direttamente all’interno del blocco *Speed And Orientation Tracking* in Simulink. Se non viene eseguita nessuna delle due operazioni verrà utilizzata la traiettoria di default. In Figura 3.3 è mostrato il veicolo nel piano cartesiano così come interpretato dal modello. Il modello utilizzato per il filtro, il quale può essere inserito

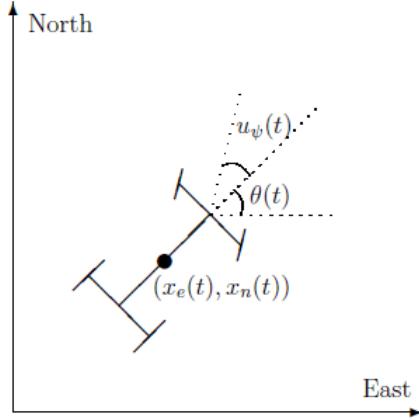


Figura 3.3: Schematizzazione dell’autovettura

agendo sui parametri del blocco ad esso relativo, è identico al precedente fatta eccezione per le matrici Q ed R , e del passo di simulazione che in questo caso è impostato a $\Delta t = 1\text{s}$, simulando di fatto un sensore GPS. Per quanto riguarda la matrice R , essa è impostata, tramite il blocco del filtro di Kalman, in funzione della potenza del rumore, che può essere modificata tramite il blocco *Measurement Noise*. In questo caso si è inserito $R = 50$. Si noti che, inserendo uno scalare nei parametri del filtro, esso assumerà che la matrice sia diagonale di dimensione opportuna per il sistema inserito. Riguardo la matrice Q la questione è più complessa. Sono stati esaminati due approcci differenti, uno prevede l’utilizzo di una matrice tempo-invariante, così come visto fin’ora, il secondo utilizza, invece, una matrice tempo-variante. Nel caso tempo-invariante, essa è stata ottenuta impostando il valore di 0.05 nel blocco del filtro. La scelta della matrice tempo-variante deriva dall’intuizione che i valori tipici del rumore sono più alti quando la velocità è più bassa. La matrice è ottenuta come segue:

$$f_{sat}(z) = \min(\max(z, 25), 625)$$

$$Q[n] = \begin{bmatrix} 1 + \frac{250}{f_{sat}(x_e^2)} & 0 \\ 0 & 1 + \frac{250}{f_{sat}(x_n^2)} \end{bmatrix} \quad (3.5)$$

La funzione di saturazione evita che Q raggiunga valori troppo grandi o troppo piccoli. Il coefficiente 250 è ottenuto valutando il tempo di accelerazione nei range 0-5, 5-10, 10-15, 15-20, 20-25 m/s di un generico veicolo. Si noti che la scelta di una matrice Q diagonale rappresenta un’ingenua assunzione che i cambiamenti di velocità in direzione nord ed est non siano correlati. I due approcci saranno confrontati nel capitolo successivo.

3.2.1 Codice di simulazione

In questo caso, avendo il modello in Simulink, lo script Matlab avrà il compito di lanciare la simulazione e, successivamente, di disegnare i grafici ad essa relativi. Il codice sorgente è mostrato di seguito:

```

1 clear all; %Pulisce il Workspace
2 sim('nome_file'); %Lancia la simulazione
3 %Il modello Simulink scrive i risultati all'interno di variabili
4 %nel Worksapce
5 %Esse vengono poi usate per realizzare i grafici
6
7
8 figure;
9
10 %Velocità East (Asse X)
11 plot(t, x(:,3), 'bx',...
12      t, y(:,3), 'gd',...
13      t, xhat(:,3), 'ro',...
14      'LineStyle', '-');
15 title('Velocità East (Asse X)');
16 xlabel('Tempo [s]');
17 ylabel('Velocità East(Asse X) [m/s]');
18 legend('Reali','Misurati','Stima del filtro di Kalman','Location',...
19        'Best');
20 axis tight;
21
22 %Per comodità non vengono riportati tutti i grafici
23 [...]
24
25 bdclose('nome_file'); %Chiude il modello Simulink

```

Codice 3.2: Parte dello script Matlab per la simulazione del filtro di Kalman con il modello Simulink

Per il codice completo si rimanda all'Appendice A.

3.3 Modello Simulink con EKF

Un ulteriore approccio preso in considerazione utilizza il filtro di Kalman esteso e fonde i dati provenienti da due sensori: un radar e un GPS. L'EKF è stato sostituito, al posto della versione base, nel modello visto nel paragrafo precedente in modo da mantenere il modello della vettura per poter confrontare i due filtri. Lo schema completo del modello è mostrato in Figura 3.4. È possibile modificare il modello del filtro mediante il blocco *Simulink Function - State Transition Jacobian*, mostrato in Figura 3.5. In questo caso, è stato mantenuto il modello visto in precedenza, fatta eccezione

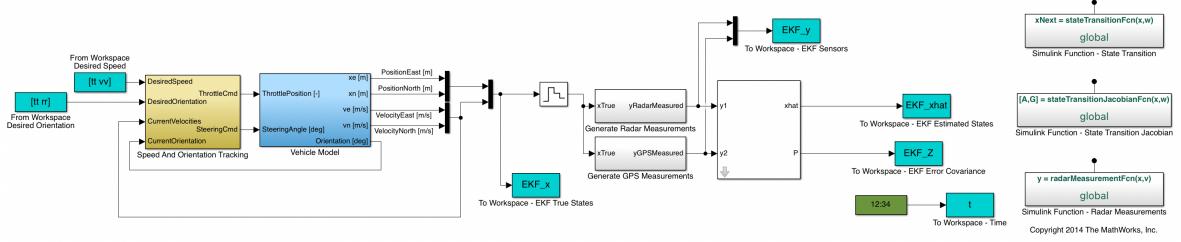


Figura 3.4: Schema completo del modello realizzato in Simulink con l'EKF (EKF-sim.slx)

per la matrice Q che viene definita come segue:

$$Q = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \quad (3.6)$$

Essendo presenti due sensori, che non necessariamente lavorano alla stessa frequenza

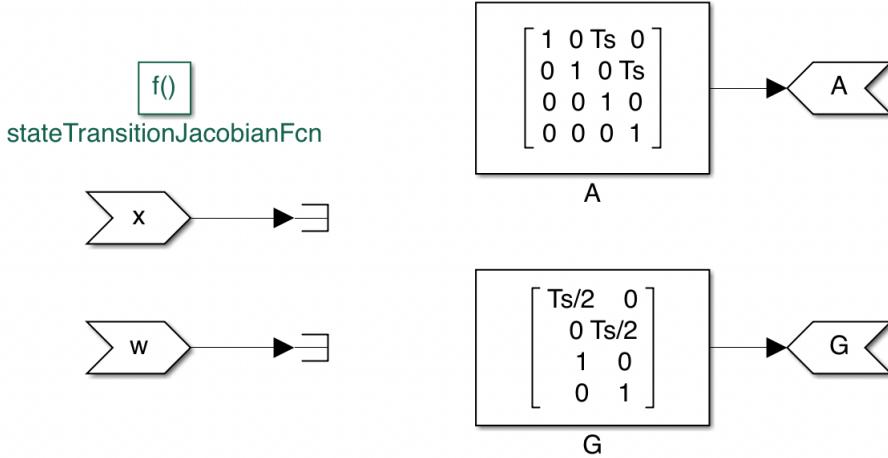


Figura 3.5: Blocco Simulink Function - State Transition Jacobian

za, tramite il blocco del filtro è possibile impostare la frequenza di funzionamento dei singoli sensori, indicando il *tempo di sample*. In questo caso abbiamo un passo di simulazione pari a $\Delta t = 0.05$, mentre la frequenza di funzionamento dei due sensori è definita nei sotto-paragrafi successivi. È possibile indicare, per ogni sensore, la funzione che ne modellizza il comportamento. Di seguito vengono descritte le funzioni utilizzate per il radar e il GPS.

3.3.1 Modello del sensore - Radar

Il radar simulato misura la distanza e l'angolo ad una frequenza di $20Hz$. Entrambe le misurazioni presentano circa il 5% di rumore. Questo può essere modellato dalla

seguente equazione di misura:

$$y_{radar}[k] = \begin{bmatrix} \sqrt{x_n[k]^2 + x_e[k]^2}(1 + v_1[k]) \\ atan2(x_n[k], x_e[k])(1 + v_2[k]) \end{bmatrix} \quad (3.7)$$

Dove $v_1[k]$ e $v_2[k]$ sono i termini del rumore di misura, ciascuno con varianza 0.05^2 , la maggior parte delle misurazioni ha errori inferiori al 5%:

$$R = \begin{bmatrix} 0.05^2 & 0 \\ 0 & 0.05^2 \end{bmatrix} \quad (3.8)$$

Il rumore di misura non è additivo perché $v_1[k]$ e $v_2[k]$ non vengono semplicemente aggiunti alle misurazioni, ma dipendono dagli stati x . La funzione che modellizza il radar è implementata tramite il blocco Simulink *Radar Measurement*.

3.3.2 Modello del sensore - GPS

Il sensore GPS misura le posizioni East e North dell'oggetto ad una frequenza di 1Hz. L'equazione di misura è la seguente:

$$y_{GPS}[k] = \begin{bmatrix} x_e[k] \\ x_n[k] \end{bmatrix} + \begin{bmatrix} v_1[k] \\ v_2[k] \end{bmatrix} \quad (3.9)$$

Dove $v_1[k]$ e $v_2[k]$ sono i termini di rumore di misura aventi matrice di covarianza:

$$R = \begin{bmatrix} 10^2 & 0 \\ 0 & 10^2 \end{bmatrix} \quad (3.10)$$

Cioè, le misurazioni sono accurate fino a circa 10 metri e gli errori non sono correlati. Il rumore di misurazione è additivo perché i termini di rumore influenzano le misurazioni y_{GPS} in modo lineare.

3.3.3 Codice di simulazione

Lo script è analogo a quello del modello precedente, fatta eccezione per alcune correzioni rese necessarie dall'inserimento di un secondo sensore.

Capitolo 4

Risultati delle simulazioni

Di seguito verranno discusse nel dettaglio le simulazioni eseguite con i modelli introdotti nel capitolo precedente. Inoltre, verranno confrontate le performance delle due versioni del filtro di Kalman, la versione base e la versione estesa.

4.1 Simulazione tramite Driving Scenario Designer

La prima simulazione effettuata è quella relativa allo script discusso nel Paragrafo 3.1. Per prima cosa è stato costruito il percorso che la l'auto avrebbe compiuto mediante il software Driving Scenario Designer, esso è mostrato in Figura 4.1. I waypoint che definiscono la traiettoria sono riportati in Tabella 4.1. Una volta esportati i dati è stata poi eseguita la stima mediante lo script. Già dai grafici di Figura 4.2 e Figura 4.3 è evidente quanto l'inserimento del filtro abbia migliorato notevolmente la stima della velocità, andando di fatto ad avvicinarsi moltissimo al valore reale, dopo una prima fase iniziale in cui il filtro si avvicina al valore iniziale iterazione dopo iterazione. Con questa configurazione si ottiene un errore sulla stima di posizione e velocità molto ridotto. Chiaramente questo è possibile anche grazie ai modelli semplificati, non applicabili ad un caso reale, usati durante le simulazione. Tuttavia, possiamo conside-

x (m)	y (m)	v (m/s)
17.7845	17.8667	1
21.8	28.9	3
35	55.6	3
49.3	84	4
53	115.5	4

Tabella 4.1: Waypoints che definiscono il percorso compiuto dall'auto

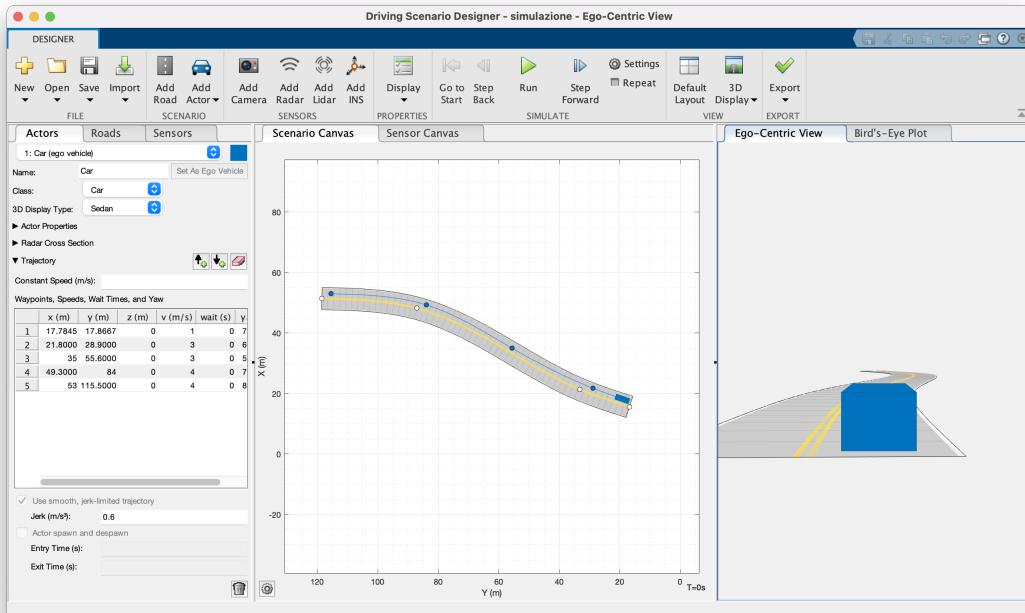


Figura 4.1: Schermata del software Driving Scenario Designer durante la costruzione del percorso per la simulazione

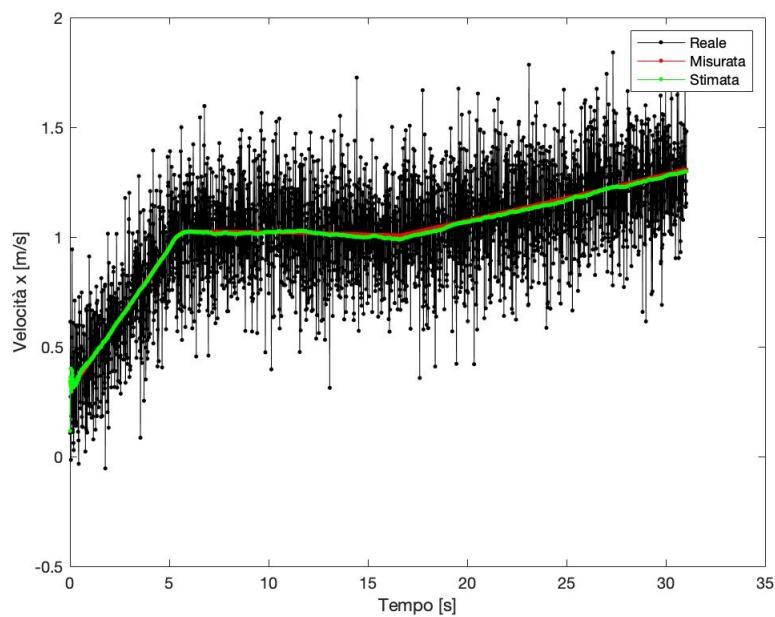


Figura 4.2: Velocità sull'asse X in funzione del tempo

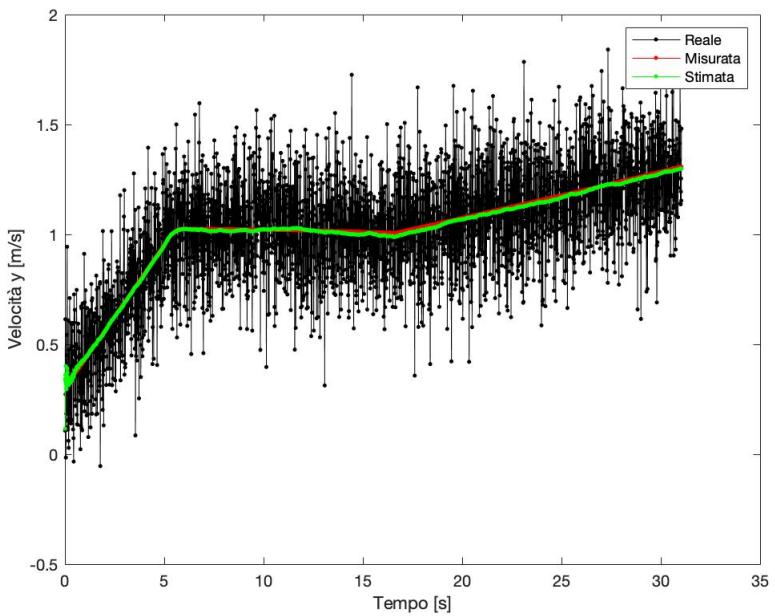


Figura 4.3: Velocità sull’asse Y in funzione del tempo

	Errore medio	Errore massimo	Errore minimo
Posizione X	0.0127	0.1537	$1.0164 \cdot 10^{-5}$
Posizione Y	0.0149	0.3607	$7.1962 \cdot 10^{-6}$
Velocità X	0.0075	0.2256	$5.68 \cdot 10^{-6}$
Velocità Y	0.0056	0.4342	$1.7342 \cdot 10^{-7}$

Tabella 4.2: Errori massimi, minimi e medi della stima mediante filtro di Kalman

rarlo un buon punto di partenza per un successivo sviluppo. I dettagli sugli errori sono mostrati nella Tabella 4.2.

4.2 Simulazione modello Simulink

Per lanciare le seguenti simulazioni è sufficiente inserire nello script il nome del file di Simulink da simulare. Sono state eseguite due simulazioni: una con la matrice Q tempo-invariante, l’altra con la matrice Q tempo-variante. Durante la simulazione vi sono delle variazioni di velocità molto rapide, che difficilmente sono replicabili in un contesto reale, al fine di verificare quanto velocemente il filtro si adatta ai cambiamenti. Per comodità, è stata mantenuta la traiettoria di default fornita col modello, così come sono state mantenute le variazioni repentine di velocità al fine di analizzare la risposta del filtro. La traiettoria è mostrata in Figura 4.4. Nei grafici seguenti sono mostrati i risultati delle simulazioni. Dai grafici si evince che l’utilizzo dei una matrice

Q tempo-variante fa sì che il filtro riesca a seguire meglio le variazioni repentine di velocità. Inoltre, a parità di traiettoria, l'errore sulla stima è più basso rispetto a quello ottenuto utilizzando una matrice tempo-invariante. Tuttavia, la stima ottenuta utilizzando quest'ultima versione del filtro tende ad essere più precisa quando la velocità si mantiene costante o varia in maniera poco significativa. I valori di errore sono riportati in Tabella 4.3.

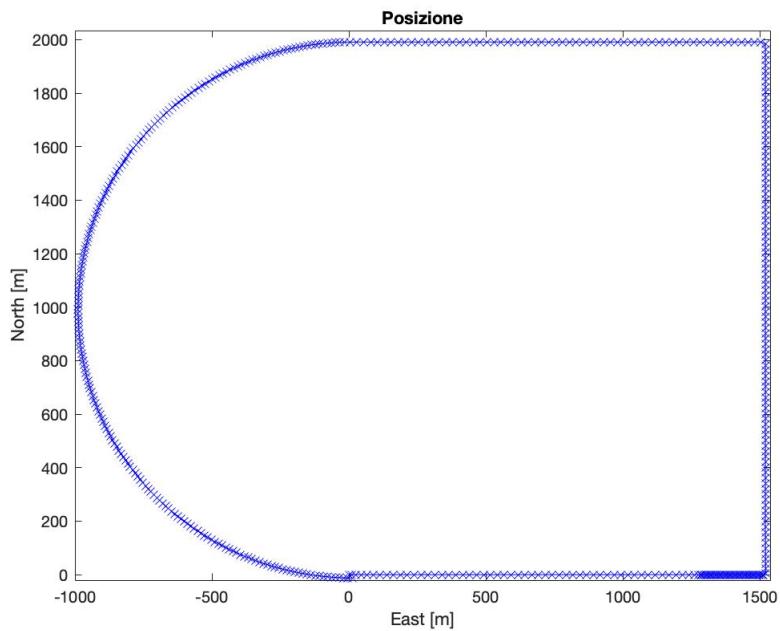


Figura 4.4: Traiettoria percorsa dall'auto

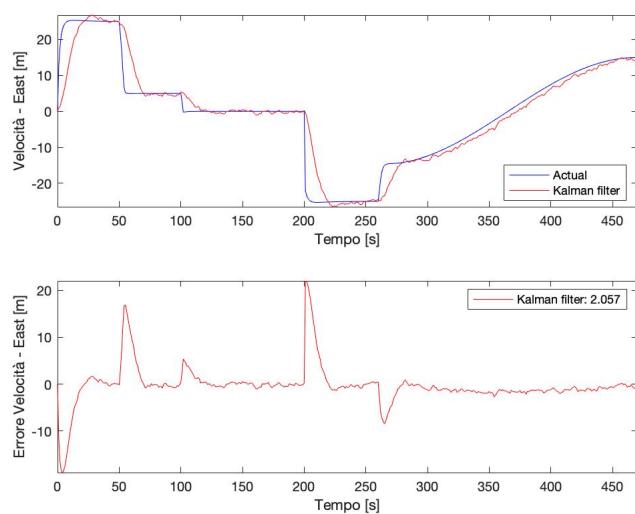


Figura 4.5: Velocità sull'asse X

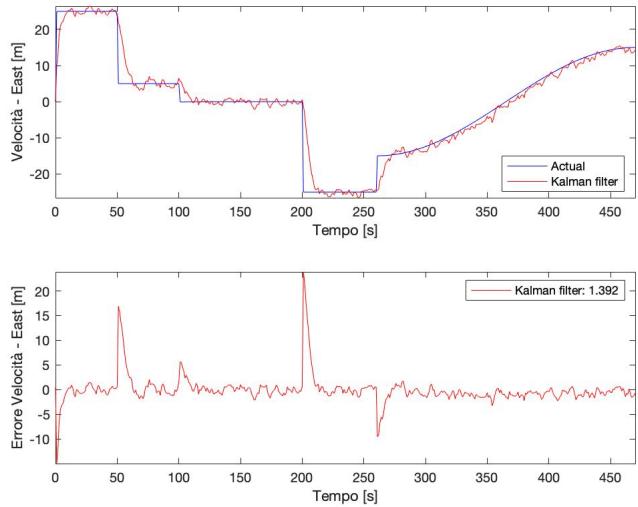


Figura 4.6: Velocità sull’asse X con matrice Q tempo-variante

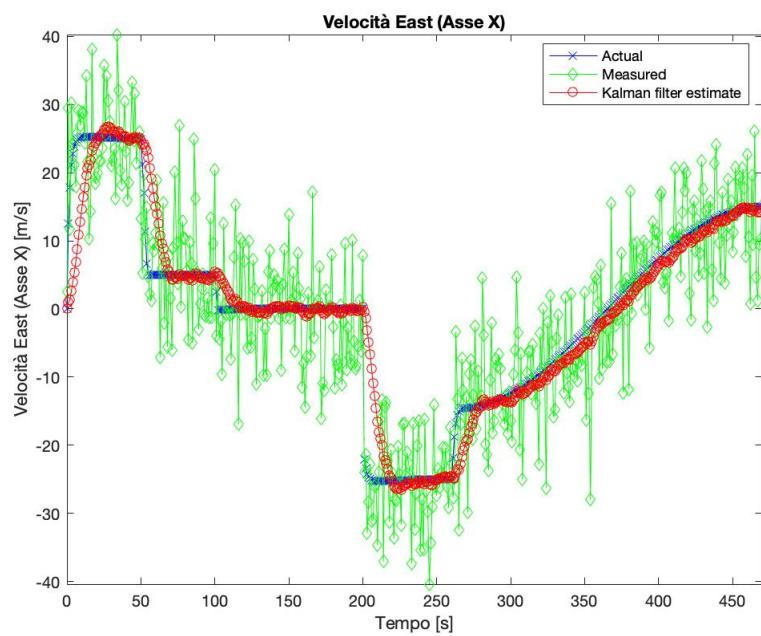


Figura 4.7: Velocità asse X - tempo-invariante

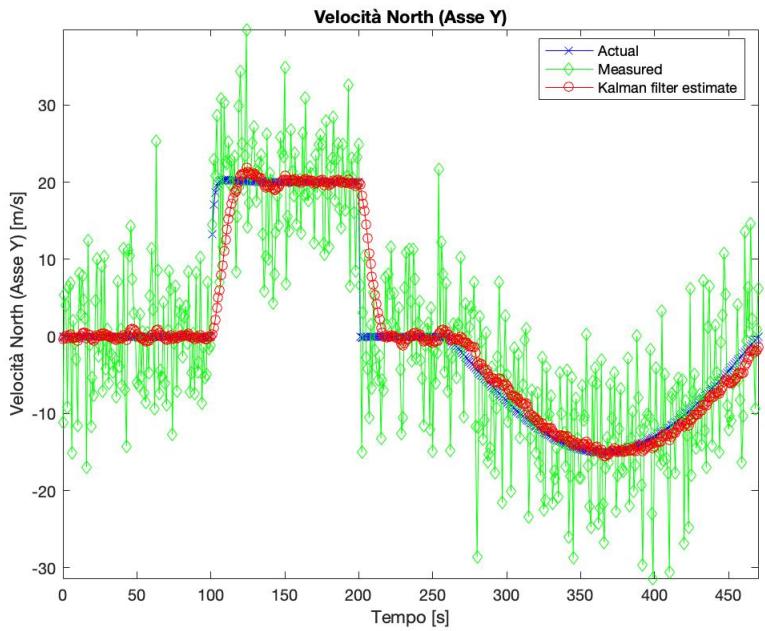


Figura 4.8: Velocità asse Y - tempo-invariante

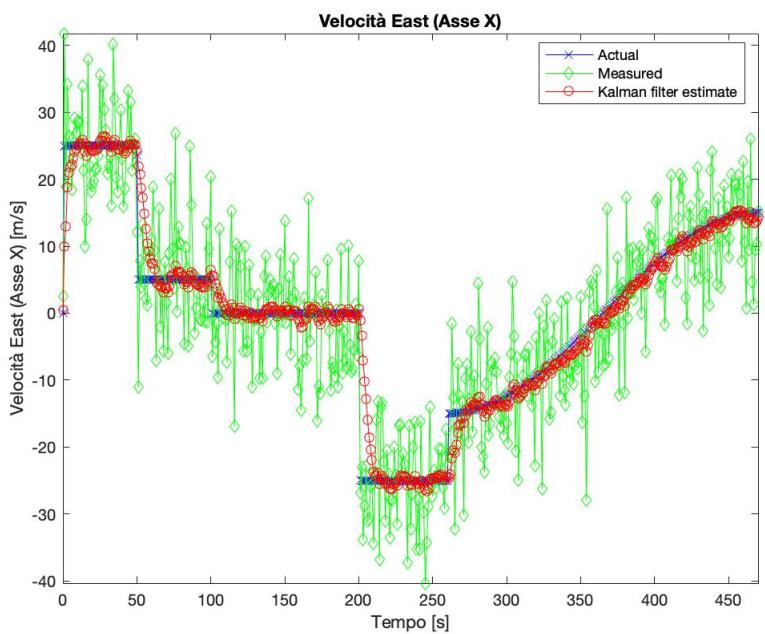


Figura 4.9: Velocità asse X - tempo-variante

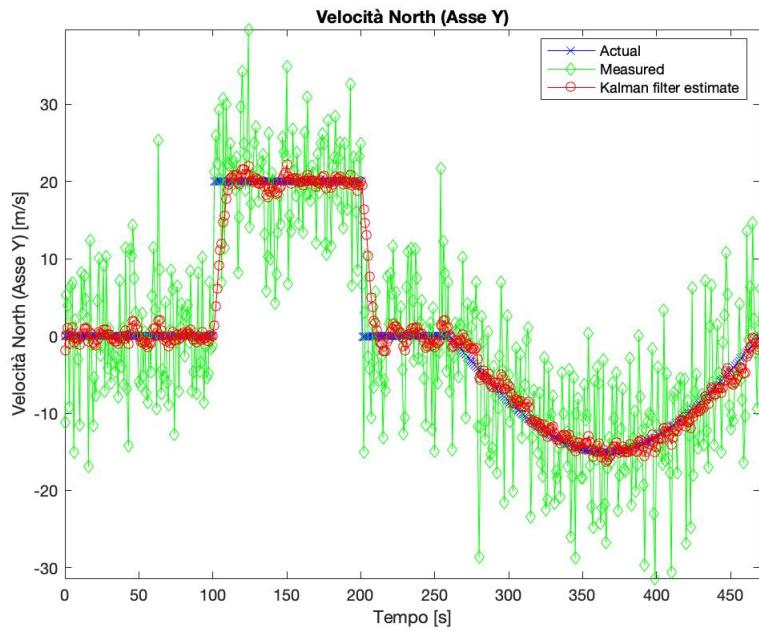


Figura 4.10: Velocità asse Y - tempo-variante

Errore medio	Matrice tempo-invariante	Matrice tempo-variante
Velocità asse X	2.0568	1.3923
Velocità asse Y	1.2679	1.0906
Posizione asse X	8.2754	4.6568
Posizione asse Y	5.0788	3.9089

Tabella 4.3: Confronto errori simulazione Simulink

Errore medio	Filtro di Kalman	Filtro di Kalman esteso
Velocità asse X	1.3923	1.5210
Velocità asse Y	1.0906	0.7898
Posizione asse X	4.6568	5.1648
Posizione asse Y	3.9089	3.1152

Tabella 4.4: Confronto errori simulazione Simulink

4.3 Simulazione modello Simulink con EKF

Anche in questo caso, è sufficiente inserire il nome del file di Simulink e lanciare la simulazione. La traiettoria è la medesima dell'esempio precedente, in modo da renderne possibile il confronto. In Figura 4.11 e Figura 4.12 sono mostrati i risultati delle simulazioni relativi all'asse East (X), mentre in Figura 4.13 e Figura 4.14 sono mostrati i risultati delle simulazioni relativi all'asse North (Y). Come si evince dai grafici e dagli errori riportati in Tabella 4.4, i risultati sono molto simili a quelli ottenuti con il filtro base adoperando una matrice Q tempo-variante. Tuttavia, in questo caso la frequenza complessiva di funzionamento è pari a $20Hz$, dunque gli aggiornamenti della stima sono molto più frequenti. Come si evince dal grafico in Figura 4.15, vengono effettuate piccole correzioni ogni 0.05 secondi e correzioni più grandi ogni secondo. Anche in questo caso è possibile notare che la stima si discosta maggiormente dal valore reale durante variazioni repentine di velocità.

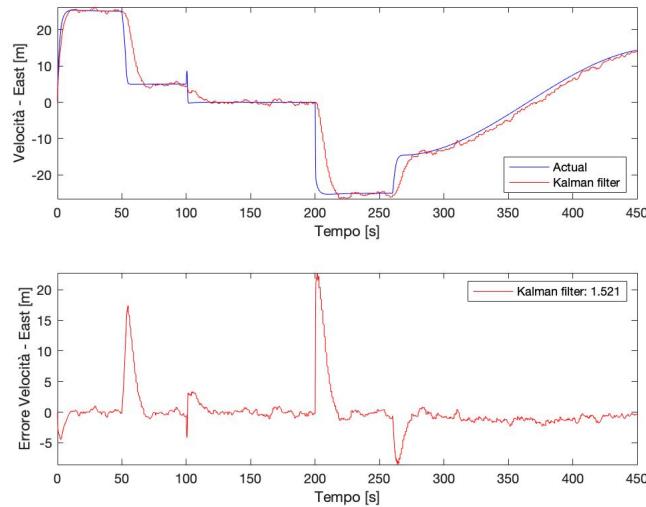


Figura 4.11: Velocità ed errore della stima sull'asse X EKF

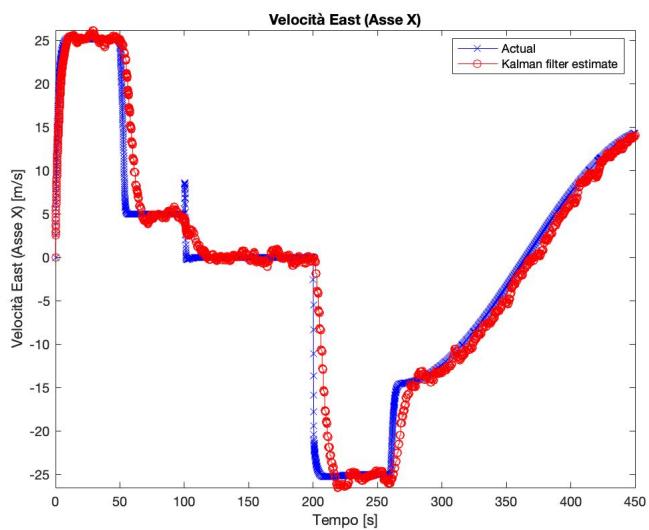


Figura 4.12: Velocità sull’asse X EKF

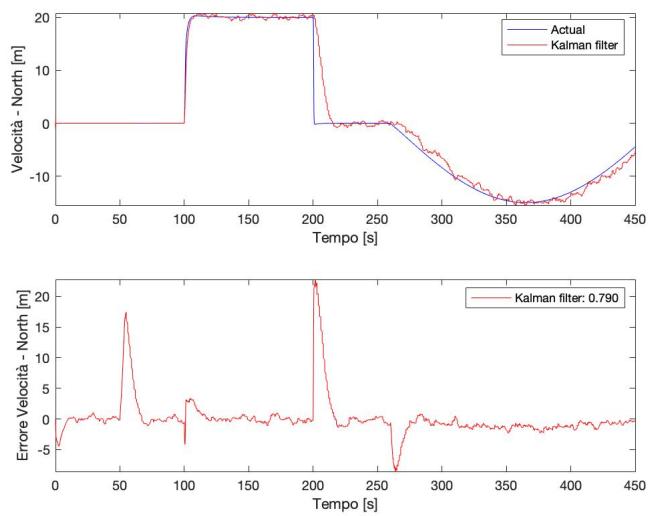


Figura 4.13: Velocità ed errore della stima sull’asse Y EKF

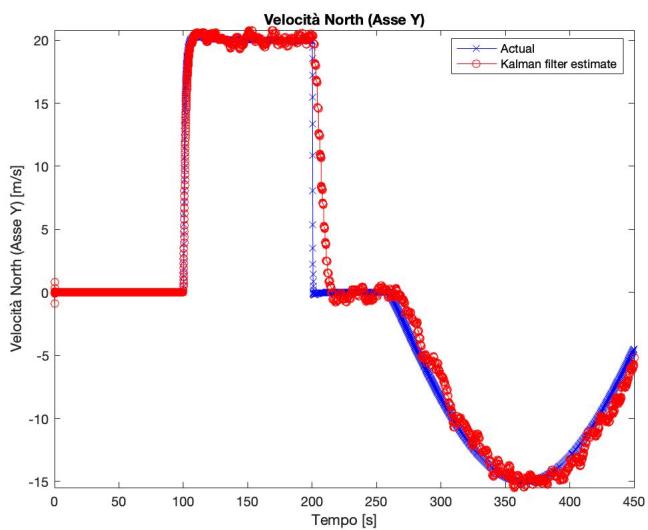


Figura 4.14: Velocità sull’asse Y EKF

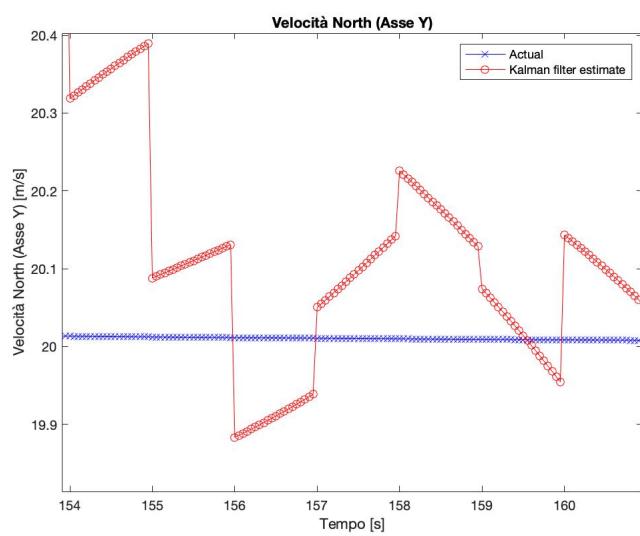


Figura 4.15: Dettaglio stima velocità

Capitolo 5

Conclusioni

Le applicazioni in ambito automotive che richiedono una stima precisa e accurata dello stato del veicolo, e soprattutto della velocità, sono in rapida crescita. Con quest'elaborato è stata fornita una buona base di partenza per iniziare lo sviluppo di sistemi capaci di soddisfare esigenze reali. L'aver approfondito i filtri di Kalman, sicuramente tra le tecniche di fusione multisensoriale più diffuse, pone le basi per lo sviluppo di sistemi più complessi. L'uso del filtro di Kalman esteso, che meglio si presta a casi d'uso reali, consente di iniziare fin da subito lo studio di tali sistemi.

I tre approcci analizzati hanno dimostrato quanto sia possibile, anche utilizzando modelli relativamente semplici, andare a ridurre l'errore sulla stima. I risultati ottenuti sono molto soddisfacenti, in termini di errori si è riusciti, in alcuni casi, ad ottenere valori molto piccoli e prossimi allo zero.

Il passo successivo richiede sicuramente la modellizzazione di sensori realmente implementati sul veicolo, al fine di stimare con maggior precisioni i parametri del filtro, insieme all'utilizzo di modelli più complessi per la gestione del veicolo.

Appendice A

Codici sorgenti

```
1 %% Filtro di Kalman 2D - Driving Scenario Designer
2
3 clear all; %Pulisce il workspace
4 load('nome_file.mat'); %Carica i dati del INS
5
6 n = length(INS_1); %Numero di campioni della traiettoria
7 %È importante chiamare INS_1 i dati esportati perché
8 %è così che sono definiti nello script
9
10 %Offset sensori
11 offset_x = 1.5;
12 offset_y = 0;
13
14 %Condizioni iniziali
15 init.px = INS_1(1).ActorPoses(1).Position(1); %Posizione
16 init.py = INS_1(1).ActorPoses(1).Position(2);
17
18 init.vx = INS_1(1).ActorPoses(1).Velocity(1); %Velocità
19 init.vy = INS_1(1).ActorPoses(1).Velocity(2);
20
21 %Tempo
22 t0 = 0; %Istante iniziale
23 tf = INS_1(n).Time; %Tempo finale
24 deltat = round(((tf-t0)/n), 2); %"Passo di simulazione"
25 t = linspace(t0, tf, n); %Vettore tempo
```

```

26
27 %Dati macchina simulati
28 esatti = [INS_1.ActorPoses];
29 esatti = esatti(1, :);
30
31 %Dati sensore INS macchina simulata
32 misurati = cell2mat([INS_1.INSMeasurements]);
33
34 % Kalman filter
35
36 A = [1 0 deltat 0;
37         0 1 0 deltat;
38         0 0 1 0;
39         0 0 0 1]; % Matrice transizione di stato
40
41 B = [0.5*deltat^2 0;
42         0 0.5*deltat^2;
43         deltat 0 ;
44         0 deltat]; % Matrice degli input
45
46 H = eye(length(A)); % Matrice di misura
47
48 p_est = cell2mat(struct2cell(init));
49
50 %Matrice di covarianza P, covarianza del processo Q ed errore di
51 %misura R
52 acc_noise_mag = 0.05;
53 sensor_noise_mag = 0.2;
54 R = sensor_noise_mag^2; %R, errore di misura, matrice di
55 %covarianza di misura
56 Q = acc_noise_mag^2 * B * B'; %Rumore del processo
57 kalman(1).P = eye(4); % Stima della varianza nella posizione
58 %iniziale
59 stimata.x = [];
60 stimata.y = [];
61 stimata.vx = [];

```

```

60 stimata.vy = [];
61
62 ideale.x = [];
63 ideale.y = [];
64 ideale.vx = [];
65 ideale.vy = [];
66
67 misurata.x = [];
68 misurata.y = [];
69 misurata.vx = [];
70 misurata.vy = [];
71
72 %Init
73 stimata.x = p_est(1);
74 stimata.y = p_est(2);
75 stimata.vx = p_est(3);
76 stimata.vy = p_est(4);
77
78 ideale.x = p_est(1);
79 ideale.y = p_est(2);
80 ideale.vx = p_est(3);
81 ideale.vy = p_est(4);
82
83 misurata.x = p_est(1);
84 misurata.y = p_est(2);
85 misurata.vx = p_est(3);
86 misurata.vy = p_est(4);
87 %Ciclo filtro di Kalman
88 for i = 1:n-1
89     u = [misurati(i).Acceleration(1); misurati(i).Acceleration(2)];
90     Y = [misurati(i).Position(1)-offset_x; misurati(i).Position(2)-offset_y; misurati(i).Velocity(1); misurati(i).Velocity(2)];
91     p_est = A * p_est + B * u;
92     kalman(i+1).P = A * kalman(i).P * A' + Q;
93     kalman(i).K = kalman(i+1).P*H'*inv(H*kalman(i+1).P*H'+R*eye(

```

```

length(A));

94 p_est = p_est + kalman(i).K * (Y - H*p_est);

95 kalman(i+1).P = ( eye(length(A)) - kalman(i).K*H ) * kalman(i
+1).P;

96

97 stimata.x = [stimata.x; p_est(1)];
98 stimata.y = [stimata.y; p_est(2)];
99 stimata.vx = [stimata.vx; p_est(3)];
100 stimata.vy = [stimata.vy; p_est(4)];

101

102 misurata.x = [misurata.x; Y(1)];
103 misurata.y = [misurata.y; Y(2)];
104 misurata.vx = [misurata.vx; Y(3)];
105 misurata.vy = [misurata.vy; Y(4)];

106

107 ideale.x = [ideale.x; esatti(i).Position(1)];
108 ideale.y = [ideale.y; esatti(i).Position(2)];
109 ideale.vx = [ideale.vx; esatti(i).Velocity(1)];
110 ideale.vy = [ideale.vy; esatti(i).Velocity(2)];

111 end

112 % Grafico

113 plot(ideale.x,ideale.y,'r.-',misurata.x,misurata.y,'k.-', stimata
    .x, stimata.y,'g.-');

114 legend('Reale','Misurata','Stimata');

115 xlabel("Posizione x");

116 ylabel("Posizione y");

117 plot(t,ideale.x,'r.-',t,misurata.x,'k.-', t,stimata.x,'g.-');

118 legend('Reale','Misurata','Stimata');

119 xlabel("Tempo");

120 ylabel("Posizione x");

121 plot(t,ideale.y,'r.-',t,misurata.y,'k.-', t,stimata.y,'g.-');

122 legend('Reale','Misurata','Stimata');

123 xlabel("Tempo");

124 ylabel("Posizione y");

125 %Velocità X

126 plot(t,ideale.vx,'r.-',t,misurata.vx,'k.-', t,stimata.vx,'g.-');

127 legend('Reale','Misurata','Stimata');

```

```

128 xlabel("Tempo");
129 ylabel("Velocità x");
130 %Velocità Y
131 plot(t,ideale.vy,'r.-',t,misurata.vy,'k.-', t,stimata.vy,'g.-');
132 legend('Reale','Misurata','Stimata');
133 xlabel("Tempo");
134 ylabel("Velocità y");

```

Codice A.1: Script Matlab per la simulazione del filtro di Kalman

```

1 clear all;
2 sim('nome_file');
3
4 figure;
5
6 %Velocità East (Asse X)
7 plot(t, x(:,3),'bx',...
8      t, y(:,3), 'gd',...
9      t, xhat(:,3), 'ro',...
10     'LineStyle', '-');
11 title('Velocità East (Asse X)');
12 xlabel('Tempo [s]');
13 ylabel('Velocità East (Asse X) [m/s]');
14 legend('Actual','Measured','Kalman filter estimate','Location',...
15        'Best');
16 axis tight;
17
18 figure;
19 %Velocità North (Asse Y)
20 plot(t, x(:,4),'bx',...
21      t, y(:,4), 'gd',...
22      t, xhat(:,4), 'ro',...
23     'LineStyle', '-');
24 title('Velocità North (Asse Y)');
25 xlabel('Tempo [s]');
26 ylabel('Velocità North (Asse Y) [m/s]');
27 legend('Actual','Measured','Kalman filter estimate','Location',...

```

```

    Best');

28 axis tight;

29

30 figure;

31

32 %Posizione
33 plot(x(:,1),x(:,2),'bx',...
34     y(:,1),y(:,2),'gd',...
35     xhat(:,1),xhat(:,2),'ro',...
36     'LineStyle','-');
37 title('Posizione');
38 xlabel('East [m]');
39 ylabel('North [m]');
40 legend('Actual','Measured','Kalman filter estimate','Location',...
        'Best');
41 axis tight;

42

43 % East errore di misura posizione [m]
44 n_xe = y(:,1)-x(:,1);
45 % North errore di misura posizione [m]
46 n_xn = y(:,2)-x(:,2);
47 % Kalman filter east errore posizione [m]
48 e_xe = xhat(:,1)-x(:,1);
49 % Kalman filter north errore posizione [m]
50 e_xn = xhat(:,2)-x(:,2);

51

52 figure;
53 % East errore posizione
54 subplot(2,1,1);
55 plot(t,n_xe,'g',t,e_xe,'r');
56 ylabel('Errore Posizione - East [m]');
57 xlabel('Tempo [s]');
58 legend(sprintf('Meas: %.3f',norm(n_xe,1)/numel(n_xe)),sprintf('
        Kalman f.: %.3f',norm(e_xe,1)/numel(e_xe)));
59 axis tight;
60 % North errore posizione
61 subplot(2,1,2);

```

```

62 plot(t,y(:,2)-x(:,2), 'g', t,xhat(:,2)-x(:,2), 'r');
63 ylabel('Errore Posizione - North [m]');
64 xlabel('Tempo [s]');
65 legend(sprintf('Meas: %.3f',norm(n_xn,1)/numel(n_xn)),sprintf(
66     'Kalman f: %.3f',norm(e_xn,1)/numel(e_xn)));
67 axis tight;
68
69 e_ve = xhat(:,3)-x(:,3); % [m/s] Kalman filter east errore
70     velocità
71 e_vn = xhat(:,4)-x(:,4); % [m/s] Kalman filter north errore
72     velocità
73 figure;
74 % Velocità in direzione East e la sua stima
75 subplot(2,1,1);
76 plot(t,x(:,3), 'b', t,xhat(:,3), 'r');
77 ylabel('Velocità - East [m]');
78 xlabel('Tempo [s]');
79 legend('Actual','Kalman filter','Location','Best');
80 axis tight;
81 subplot(2,1,2);
82 % Errore di stima
83 plot(t,e_ve, 'r');
84 ylabel('Errore Velocità - East [m]');
85 xlabel('Tempo [s]');
86 legend(sprintf('Kalman filter: %.3f',norm(e_ve,1)/numel(e_ve)));
87 axis tight;
88
89 figure;
90 % Velocità in direzione North e la sua stima
91 subplot(2,1,1);
92 plot(t,x(:,4), 'b', t,xhat(:,4), 'r');
93 ylabel('Velocità - North [m]');
94 xlabel('Tempo [s]');
95 legend('Actual','Kalman filter','Location','Best');
96 axis tight;
97 subplot(2,1,2);
98 % Errore di stima

```

```

96 plot(t,e_ve,'r');
97 ylabel('Errore Velocità - North [m]');
98 xlabel('Tempo [s]');
99 legend(sprintf('Kalman filter: %.3f',norm(e_vn,1)/numel(e_vn)));
100 axis tight;
101
102
103 bdclose('nome_file');
104
105 %Calcolo errori
106 errore_vnord_medio = norm(e_vn,1)/numel(e_vn);
107 errore_vest_medio = norm(e_ve,1)/numel(e_ve);
108 errore_pest_medio = norm(e_xe,1)/numel(e_xe);
109 errore_pnord_medio = norm(e_xn,1)/numel(e_xn);
110 %Vest Vnord Pest Pnord
111 errori = [errore_vest_medio max(abs(e_ve)) min(abs(e_ve));
112           errore_vnord_medio max(abs(e_vn)) min(abs(e_vn));
113           errore_pest_medio max(abs(e_xe)) min(abs(e_xe));
114           errore_pnord_medio max(abs(e_xn)) min(abs(e_xn));]

```

Codice A.2: Script Matlab per la simulazione del filtro di Kalman mediante Simulink

Bibliografia

- [1] Wikipedia. *Autovettura autonoma* — Wikipedia, L'enciclopedia libera. 2021. URL: https://it.wikipedia.org/wiki/Autovettura_autonoma.
- [2] Parlamento Europeo. *Auto a guida autonoma in UE: dalla fantascienza alla realtà*. 2019. URL: <https://www.europarl.europa.eu/news/it/headlines/economy/20190110ST023102/auto-a-guida-autonoma-in-ue-dalla-fantascienza-all-realita>.
- [3] Nino Grasso Hardware Upgrade. *La guida umana potrebbe essere dichiarata illegale prima del previsto: ecco le prime stime*. 2021. URL: https://auto.hwupgrade.it/news/tecnologia/la-guida-umana-potrebbe-essere-dichiarata-illegale-prima-del-previsto-ecco-le-prime-stime_100411.html.
- [4] Diego Galar e Uday Kumar. “Chapter 1 - Sensors and Data Acquisition”. In: *eMaintenance*. A cura di Diego Galar e Uday Kumar. Academic Press, 2017, pp. 1–72. ISBN: 978-0-12-811153-6. DOI: <https://doi.org/10.1016/B978-0-12-811153-6.00001-4>. URL: <https://www.sciencedirect.com/science/article/pii/B9780128111536000014>.
- [5] Università degli Studi di Brescia Matteo Lancini. *Sensor Fusion*. 2020. URL: <https://www.youtube.com/watch?v=dM1ZTSwhOAE>.
- [6] Essam Debie et al. “Multimodal Fusion for Objective Assessment of Cognitive Workload: A Review”. In: *IEEE Transactions on Cybernetics* PP (set. 2019), pp. 1–14. DOI: [10.1109/TCYB.2019.2939399](https://doi.org/10.1109/TCYB.2019.2939399).
- [7] Mee the skilled. *Teorema del limite centrale*. 2019. URL: <https://meetheskilled.com/teorema-del-limite-centrale/>.

- [8] Udacity team. *Sensor Fusion Algorithms Explained*. 2020. URL: <https://www.udacity.com/blog/2020/08/sensor-fusion-algorithms-explained.html>.
- [9] Wikipedia. *Rete neurale convoluzionale* — Wikipedia, L'enciclopedia libera. 2021. URL: https://it.wikipedia.org/wiki/Rete_neurale_convoluzionale.
- [10] Wikipedia. *Filtro di Kalman* — Wikipedia, L'enciclopedia libera. 2021. URL: https://it.wikipedia.org/wiki/Filtro_di_Kalman.
- [11] Beren Millidge et al. *Neural Kalman Filtering*. Feb. 2021.
- [12] M. van Biezen. *Special topics - the kalman filter*. 2015. URL: <https://youtube.com/playlist?list=PLX2gX-ftPVXU3oUFNATxGXY90AULiqnWT>.
- [13] Tine Lefebvre, Herman Bruyninckx e Joris Schutter. “Kalman filters for non-linear systems: A comparison of performance”. In: *International Journal of Control - INT J CONTR* 77 (mag. 2004), pp. 639–653. DOI: 10.1080/00207170410001704998.
- [14] Gary Bishop Greg Welch. “An introduction to the Kalman filter”. In: (nov. 1995).
- [15] Università degli Studi di Brescia Massimiliano Micheli. *Filtro di Kalman*. 2020. URL: <https://www.youtube.com/watch?v=hPbdojkeMes>.
- [16] Shyam Mohan M et al. *Introduction to the Kalman Filter and Tuning its Statistics for Near Optimal Estimates and Cramer Rao Bound*. 2015. arXiv: 1503.04313 [stat.ME].
- [17] Michele Rossi. *Sensor fusion per la localizzazione indoor in applicazioni context-aware*. 2016.
- [18] Wikipedia. *Extended Kalman filter* — Wikipedia, L'enciclopedia libera. 2021. URL: https://en.wikipedia.org/wiki/Extended_Kalman_filter.
- [19] The MathWorks Inc. *Understanding Sensor Fusion and Tracking, Part 1: What Is Sensor Fusion?* 2019. URL: <https://www.youtube.com/watch?v=6qV3YjFppuc>.

- [20] The MathWorks Inc. *State Estimation Using Time-Varying Kalman Filter*. URL: <https://it.mathworks.com/help/control/ug/state-estimation-using-time-varying-kalman-filter.html>.
- [21] The MathWorks Inc. *Estimate States of Nonlinear System with Multiple, Multirate Sensors*. URL: <https://it.mathworks.com/help/control/ug/multirate-nonlinear-state-estimation-in-simulink.html>.