



UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Magistrale in Informatica
Resilient and Secure Cyber-Physical System

UTILIZZO DI UN ALGORITMO SENSOR
FUSION NELL'AMBITO DELLA
LOCALIZZAZIONE FERROTRAMVIARIA

USE OF A SENSOR FUSION ALGORITHM IN
THE AREA OF TRAMWAY LOCALIZATION

ALEX FOGLIA

ANDREA BONDAVALLI

Anno Accademico 2018-2019

ABSTRACT Il problema del posizionamento ferrotramviario è una questione di primaria importanza. Risolvere il problema del posizionamento significa poter determinare la posizione di un treno lungo una traccia ferrotramviaria. L'importanza di essere a conoscenza di tale informazione è fondamentale in quanto, per ragioni di rotta, un treno potrebbe avere necessità di spostarsi su un nuovo binario una volta raggiunte determinate posizioni.

Qualora si rendesse necessaria tale operazione, deve intervenire un sistema di scambio che direzioni le rotaie verso la nuova traccia. Il sistema di scambio è un sistema cosiddetto *safety-critical*, in quanto un suo malfunzionamento potrebbe portare a conseguenze catastrofiche, come ad esempio il deragliamento del treno.

I sistemi di posizionamento attualmente in uso fanno un largo uso di apparati installati a terra, i quali presentano un costo e un impatto ambientale non trascurabili. In questa Tesi viene mostrato un sistema di posizionamento alternativo, basato sull'utilizzo di un elaboratore installato bordo treno, sul quale viene eseguito un algoritmo *Sensor Fusion*.

Un algoritmo *Sensor Fusion*, applicato al problema del posizionamento ferrotramviario, prende in ingresso misurazioni effettuate da un *set* di sensori composto da un sensore inerziale, un odometro e un GPS; e fornisce la stima della posizione del treno più accurata di quella che si otterrebbe considerando i sensori in maniera mutuamente esclusiva. Questo avviene poichè il rumore casuale che caratterizza sia il moto del treno, che l'acquisizione delle misurazioni da parte dei sensori, rende sempre meno affidabili, con il procedere del tempo, le stime della posizione eventualmente basate unicamente su tali rilevazioni.

Il sistema presentato utilizza un modulo software che implementa un Filtro di Kalman quale modello statistico di stima dello stato di un sistema dinamico rumoroso, in cui il sistema è il treno che si muove lungo una traccia, e lo stato è la progressiva chilometrica di quest'ultimo rispetto all'origine della traccia.

INDICE

1	Stato dell'Arte	9
1.1	Sistemi Ferroviari e Ferrotramviari	9
1.2	Il Problema del Posizionamento	10
1.2.1	Criticità	14
2	Sensor Fusion	17
2.1	Panoramica	17
2.1.1	Sistemi Dinamici	17
2.1.2	Misure e Rumore	18
2.2	I Filtri di Kalman	20
2.2.1	Premesse statistiche	21
2.2.2	Filtro di Kalman Lineare	21
2.2.3	Esempio applicativo	24
2.2.4	Filtri di Kalman non Lineari	33
3	Applicazione di SFA: La Tramvia di Firenze	35
3.1	Architettura di Sistema	35
3.1.1	Architettura Hardware	36
3.1.2	Architettura Software	37
3.2	Gestione della trasmissione dei dati	39
3.2.1	Trasmissione in entrata	39
3.2.2	Trasmissione in uscita	42
3.3	Scenario di Esempio	43
3.4	Possibili sviluppi	46
3.4.1	Problematiche legate alla security	47
3.4.2	Miglioramenti al protocollo in uscita	49
4	Risultati Sperimentali	51
4.1	Software impiegati	53
4.1.1	FusionLib	53
4.1.2	RTT	56
4.1.3	Listener	57
4.1.4	SDGA	57
4.2	Esperimenti offline	58
4.3	Esperimenti online	58
5	Conclusioni	59

ELENCO DELLE TABELLE

Tabella 1	Segnalazioni semaforiche ferrotramviarie	14
Tabella 2	Misurazioni di esempio del corpo in caduta libera	26
Tabella 3	Risultati generali dell'algoritmo	31
Tabella 4	Errori a posteriori	31
Tabella 5	Confronto tra errore a posteriori sulla posizione, e distanza fra valori reali di posizione e misure di posizione	32
Tabella 6	Protocollo di comunicazione in entrata	40
Tabella 7	Significato del campo SENSOR_TYPE	41
Tabella 8	Protocollo di comunicazione in uscita	43
Tabella 9	Formato del pacchetto di <i>ack</i>	43
Tabella 10	Condizioni iniziali	43

ELENCO DELLE FIGURE

Figura 1	Treno in arrivo alla stazione ferroviaria di Firenze Santa Maria Novella	10
Figura 2	Tramvia di Danhai, Taipan	10
Figura 3	Schema di un tipico scenario tramviario	11
Figura 4	UCS realizzato da Thales Italia SPA	12
Figura 5	Conta Assi	13
Figura 6	Esempio di <i>Point Machine</i> installata su una traccia ferrotramviaria	13
Figura 7	Schema SFA	15
Figura 8	Grafico dell' errore di stima della posizione con $\alpha = 10^0, \varepsilon = 10^{-3}$	19
Figura 9	Processo caratterizzato da <i>rumore</i>	20
Figura 10	Processo e misura caratterizzati da rumore	22
Figura 11	Schema di un KF lineare	24
Figura 12	Stima della velocità del corpo in caduta	29
Figura 13	Stima della posizione del corpo in caduta	30
Figura 14	<i>Inertial Measurment Unit</i>	37
Figura 15	Architettura hardware bordo treno	38
Figura 16	Architettura software bordo treno	40
Figura 17	Esperimenti offline	51
Figura 18	Esperimenti online	52
Figura 19	Tramvia di Firenze - Linea T1	53
Figura 20	Architettura logica di FusionLib	54
Figura 21	Spline interpolante la funzione $\cos(x)$ nelle ascisse $\frac{k\pi}{2} \quad k = 0, \dots, 8$	55
Figura 22	Funzione $\cos(x)$ nell'intervallo reale $[0, 4\pi]$	56

STATO DELL'ARTE

1.1 SISTEMI FERROVIARI E FERROTRAMVIARI

Il concetto di *treno* come comunemente percepito nasce con l'inizio della Rivoluzione Industriale, avvenuta tra il *XVIII* e il *XIX* secolo, a seguito della quale l'avvento della macchina a vapore ha permesso all'umanità di disporre di fonti di energia sufficienti a fare evolvere i primi rudimentali trasporti su binario negli odierni sistemi ferroviari.

È possibile schematizzare un Sistema Ferroviario, o Ferrotramviario, come un veicolo, il treno, vincolato a muoversi attraverso una propulsione, elettrica o a combustibile, lungo una traccia fissa, il binario.

Queste caratteristiche accomunano qualsiasi sistema di trasporto ferroviario o ferrotramviario a prescindere dalla sua scala in termini di veicoli transitanti ed estensione geografica. Ciò che invece differenzia un Sistema Ferroviario da un Sistema Ferrotramviario sono:

- Le caratteristiche fisiche del treno, come lunghezza e massa;
- Le caratteristiche geografiche dell'ambiente operativo;
- Gli scopi del trasporto.

In generale, nel trasporto ferroviario si utilizzano treni caratterizzati da grandi dimensioni, che trasportano persone o merci su lunghe percorrenze (regionali, nazionali o internazionali), operando pertanto prevalentemente in ambienti extra urbani. Un esempio di treno operante in un sistema ferroviario classico è quello in figura 1.

Il trasporto ferrotramviario, di contro, vede l'utilizzo di treni dalle ridotte dimensioni, più leggeri di quelli usati nei sistemi ferroviari, e che hanno lo scopo di rappresentare un'alternativa per il cittadino all'utilizzo di mezzi privati durante i suoi spostamenti all'interno di un'area metropolitana. Quest'ultima caratteristica implica che l'ambiente operativo di un



Figura 1: Treno in arrivo alla stazione ferroviaria di Firenze Santa Maria Novella

sistema ferrotramviario sia radicalmente diverso da quello di un sistema ferroviario: i treni si muovono lungo rotaie installate su strade urbane, quindi il traffico ferrotramviario è fuso con il traffico automobilistico, motociclistico, ciclistico e pedonale che caratterizza l'ambiente urbano, come mostrato nelle figure 2 e 3.



Figura 2: Tramvia di Danhai, Taipan

1.2 IL PROBLEMA DEL POSIZIONAMENTO

Per posizionamento ferroviario, si intende la stima della posizione di un treno all'interno di una traccia ferroviaria. Esso esiste tanto nel contesto ferrotramviario quanto nel contesto ferroviario classico.

Sovente questa stima viene espressa come progressiva chilometrica rispetto all'origine della linea oppure, più raramente, come coordinata



Figura 3: Schema di un tipico scenario tramviario

geografica.

Il problema del posizionamento sorge nel momento in cui, per ragioni di rotta, un treno ha necessità di spostarsi da una sezione di binario, anche detta traccia, ad un'altra. Questa operazione di scambio è offerta dal sistema di *interlocking*. Tale sistema è detto *safety-critical*, in quanto offre una funzionalità che deve rispettare adeguati standard di sicurezza. Gli odierni sistemi di posizionamento si basano principalmente sull'utilizzo di strumenti installati a terra, che hanno lo scopo di rilevare il passaggio di un treno, e quindi di interagire con il sistema di *interlocking* della traccia al fine di garantire, con un elevato livello di confidenza, un transito sicuro dei mezzi.

Odiere Tecniche di Posizionamento

I sistemi di posizionamento attualmente in uso sono basati su un'architettura distribuita composta dai seguenti blocchi:

- Sottosistema di *interlocking*;
- Sottosistema di comunicazione treno-traccia;
- Sottosistema semaforico.

SOTTOSISTEMA DI INTERLOCKING: Il sottosistema di *interlocking* è la parte che si fa effettivamente carico di offrire al treno un attraversamento sicuro di una *Junction Area (JA)*. Una JA è un punto della linea ferroviaria in cui il treno può cambiare direzione, e occupare una nuova traccia di

binario.

La nuova traccia da occupare potrebbe avere particolari vincoli sul numero di treni contemporaneamente transitanti, ed in ogni caso lo scambio di rotaia deve essere corretto ed avvenire in sicurezza, in quanto occupare la traccia sbagliata potrebbe avere ripercussioni finanche catastrofiche.

Un sistema di *interlocking* è composto dai seguenti elementi:

- *Switch Control Unit (UCS):*

Piattaforma certificata SIL-3 che rappresenta il nucleo del sistema di *interlocking* e che implementa l'intera logica di gestione di una JA. Un UCS dispone di un'interfaccia di *Input/Output (I/O)* verso gli elementi di *interlocking* installati a terra che ne consente un controllo sicuro in accordo allo standard SIL-3.



Figura 4: UCS realizzato da Thales Italia SPA

- *Conta Assi:*

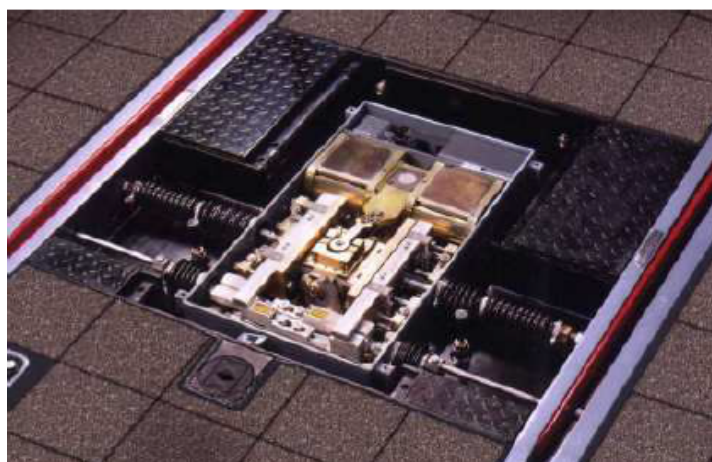
Il Conta Assi, o in inglese *Axle Counter (AC)*, è un sistema certificato SIL-3 che ha lo scopo di rilevare la presenza del treno e fornire quindi lo stato di occupazione della sezione di traccia in cui l'AC è installato.

- *Point Machines:*

Le *Point Machines* infine, sono degli strumenti certificati SIL-3 che hanno lo scopo di direzionare le rotaie verso una determinata sezione di traccia.



Figura 5: Conta Assi

Figura 6: Esempio di *Point Machine* installata su una traccia ferrotramviaria

L'intero sistema di *interlocking* viene attivato dai *Track Circuit*. Questi apparati sono installati a terra prima di ciascuna JA, e segnalano al sistema di *interlocking* l'avvicinamento di un treno alla successiva JA.

SOTTOSISTEMA DI COMUNICAZIONE TRENO-TRACCIA: Il sottosistema di comunicazione treno-traccia è gestito da un computer installato bordo treno, chiamato *On Board Control Unit* (OBCU), ed ha lo scopo di fornire funzionalità non legate alla *safety* e pertanto poco interessanti. OBCU viene principalmente utilizzato per monitorare lo stato del traffico ferrotramviario in una architettura di *monitoring* centralizzata. Il monitoring si basa su comunicazioni *wireless*. In alcune applicazioni può comprendere una comunicazione più o meno diretta con il sistema di *interlocking* allo scopo di segnalare l'avvicinamento del treno a una JA.

SOTTOSISTEMA SEMAFORICO: Il sottosistema semaforico prende in ingresso informazioni dal sistema di *interlocking* ed eventualmente, da OBCU, e gestisce i segnali luminosi da mostrare sui semafori a un mac-





Segnale	Descrizione	Significato
	Barra bianca orizzontale	Fermarsi
	Barra bianca verticale	Procedere avanti
	Barra bianca ruotata di 45 gradi	Procedere solo a destra
	Barra bianca ruotata di -45 gradi	Procedere solo a sinistra

Tabella 1: Segnalazioni semaforiche ferrotramviarie

chinista che si appresta a superare una JA.

In tabella 1 viene riportata la lista dei segnali semaforici utilizzati nel contesto ferrotramviario.

1.2.1 Criticità

Le attuali tecniche di posizionamento richiedono un intervento trascurabile di computer installati a bordo e una grande quantità di apparati installati a terra. Mentre i computer di bordo non forniscono in generale funzionalità legate alla *safety*, gli apparati installati a terra sono costosi e hanno un impatto ambientale non trascurabile.

È possibile considerare il treno e il computer di bordo come un unico sistema, ossia il treno viene modellato come un *Cyber-Physical System*.

Un *Cyber-Physical System* (CPS) è un sistema composto da una parte *fisica* e da una parte *cyber*. Il sottosistema fisico è composto da sensori e attuatori che hanno rispettivamente lo scopo di rilevare lo stato dell'ambiente circostante e di alterarlo se necessario. Il sottosistema *cyber* è essenzialmente un elaboratore, che dispone di processore, memoria, e interfacce I/O verso i sensori gli attuatori, ed eventuali operatori umani. Una tale

architettura di sistema, permette di sfruttare le capacità di calcolo dei moderni processori per implementare algoritmi anche molto complessi per il *processing* di grandi quantità di dati provenienti dai sensori.

Lo scopo della Tesi è quello di mostrare il funzionamento di un possibile sistema di posizionamento alternativo al sistema tuttora operante, che sfrutti l'uso combinato di un insieme di sensori i cui dati rilevati vengono processati da un algoritmo noto come *Sensor Fusion Algorithm* (SFA).

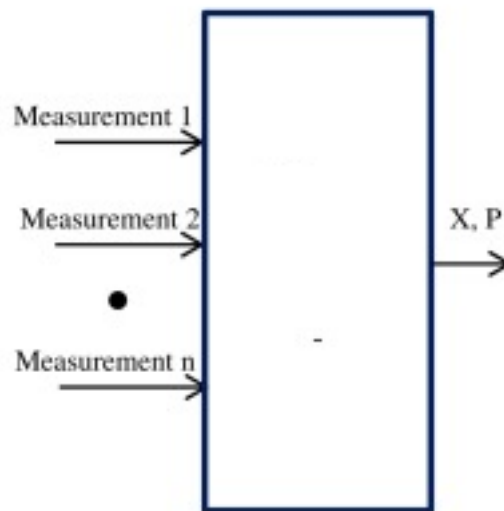


Figura 7: Schema SFA

Tale algoritmo è schematizzabile come una *black-box*: le misurazioni dei sensori sono l'ingresso, mentre l'uscita è la misura cercata, nella fattispecie, la posizione del treno lungo la traccia. Utilizzando SFA, il treno è in grado di auto-posizionarsi, capacità che minimizza la necessità di installare apparati di terra.

Un algoritmo che tiene conto delle misurazioni di un *set* di sensori, usato in luogo di un semplice *processing* di insiemi di misure provenienti da sorgenti omologhe, permette al sistema di correggere il rumore che disturba le singole misurazioni, realizzando così una nuova misura più accurata di quella che si avrebbe considerando i sensori in maniera mutuamente esclusiva.

SENSOR FUSION

Nei sistemi in cui è richiesta un'alta *reliability* delle misure, l'informazione fornita dai singoli sensori non è sufficiente. In questi casi è raccomandato l'utilizzo di un insieme di sensori in contemporanea. [1]

2.1 PANORAMICA

In generale, un algoritmo SFA viene utilizzato per stimare lo stato di un sistema dinamico in un ambiente caratterizzato da *rumore*. [2]

2.1.1 Sistemi Dinamici

Un sistema dinamico è una modellazione matematica di un processo che evolve nel tempo, la cui evoluzione è descritta attraverso un sistema di equazioni differenziali o alle differenze, nel caso esso si evolva rispettivamente a tempo continuo o a tempo discreto.

Sia S l'insieme dei possibili stati che il sistema può assumere, e sia $m = |S|$ la dimensione dello spazio degli stati.

Senza perdere in generalità, si possono formalizzare questi due tipi di sistemi dinamici come:

$$y'(t) = f(t, y(t)), \quad t \geq 0 \quad (1)$$

Con $y(0) \in \mathbb{R}^m$ condizione iniziale nota, e:

$$y_{n+1} = f(n, y_n), \quad n = 0, 1, \dots \quad (2)$$

con al solito $y_0 \in \mathbb{R}^m$ condizione iniziale nota [3].

Ricavare lo stato del sistema dinamico per un certo istante t , o n , equivale a risolvere le equazioni cui sopra e valutarne la traiettoria soluzione in t

o in n .

Un semplice sistema dinamico è rappresentato da un punto materiale che si muove con una accelerazione costante

$$\mathbf{a} = a\mathbf{k}$$

dove \mathbf{k} è un qualunque versore della base canonica di \mathbb{R}^3 .

Supponendo che il punto si muova con velocità iniziale $\mathbf{z}'(0) = v_0\mathbf{k}$ nota e inizi il moto da una coordinata $\mathbf{z}(0) = z_0\mathbf{k}$ nota, si ha:

$$z''(t) = a \quad (3)$$

$$z'(t) = \int a dt = at + v_0$$

$$z(t) = \int (at + v_0) dt = \frac{1}{2}at^2 + v_0t + z_0$$

L'equazione $z(t)$ descrive completamente la traiettoria di moto del punto materiale, mentre $z'(t)$ descrive completamente la traiettoria della velocità del punto durante il suo moto.

Nota su sistemi dinamici continui e discreti

Con l'introduzione del calcolatore come strumento di supporto al matematico applicato, sono stati sviluppati efficienti *metodi numerici di approssimazione* che permettono di modellare un sistema dinamico continuo attraverso una sua *controparte discreta*. Tale processo di discretizzazione è mirato alla definizione di un sistema dinamico che si comporti in maniera *simile* al modello continuo che si intende studiare, dove per simile si intende che ne preservi il comportamento quantitativo e le caratteristiche asintotiche di stabilità delle soluzioni.

L'*aritmetica finita* in cui lavora qualsiasi calcolatore rende indispensabile la discretizzazione dei modelli continui, ma d'altro lato l'esistenza dei metodi di approssimazione permette di restringere le discussioni che seguiranno considerando esclusivamente sistemi dinamici *discreti*. [3]

2.1.2 *Misure e Rumore*

Si consideri il sistema dinamico individuato dalla (3).

Le soluzioni esplicite della traiettoria del punto materiale sono valide, fatta assunzione di conoscere a priori il valore esatto di a , di v_0 e di z_0 .

Nella pratica, per misurare qualsiasi grandezza fisica è necessario uno

strumento di misura, il quale produrrà delle misure affette da errori casuali.

Si supponga di sostituire a nell'equazione $z(t)$ con una sua perturbazione $\tilde{a} = a + \varepsilon$ dove ε è una variazione casuale della misura data dal *rumore* che caratterizza qualsiasi processo di misura. Si può supporre $\text{Var}(\varepsilon) = 0$ e considerare, ai fini di questa trattazione, ε come un valore costante; in realtà ε è una variabile casuale a varianza generalmente non nulla. Si suppongano inoltre $v_0 = z_0 = 0$ per comodità di calcolo:

$$z(t) = \frac{1}{2}\tilde{a}t^2 = \frac{1}{2}(a + \varepsilon)t^2 = \frac{1}{2}(at^2 + \varepsilon t^2)$$

Si nota immediatamente che la variazione della misura $z(t)$ data da ε aumenta con il quadrato del tempo.

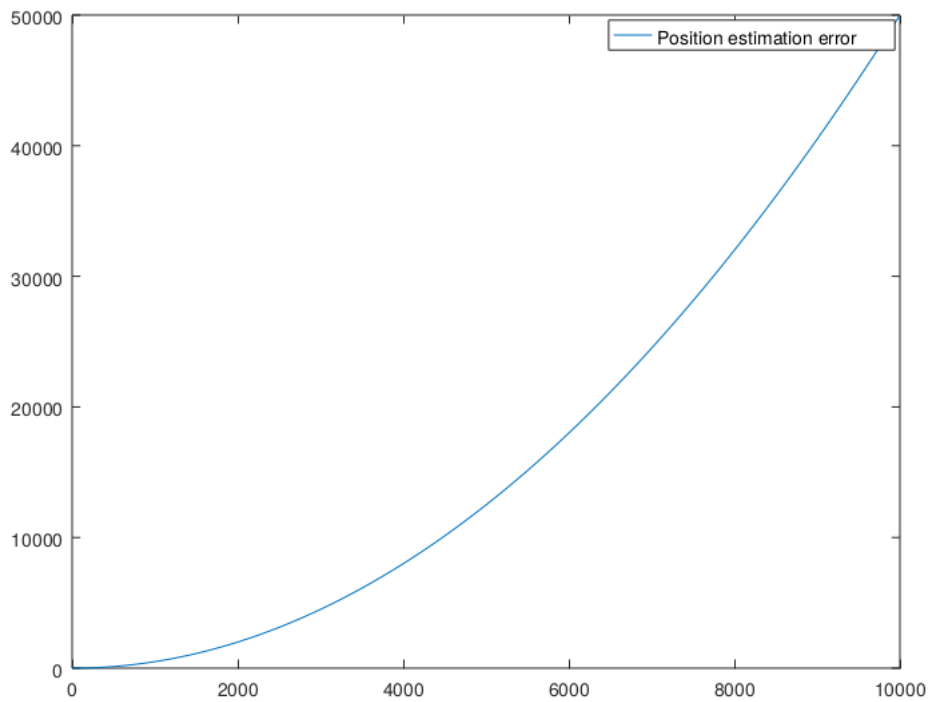


Figura 8: Grafico dell' errore di stima della posizione con $a = 10^0$, $\varepsilon = 10^{-3}$

Il moto di un punto materiale soggetto a un' accelerazione nota, è una tipologia di sistema dinamico caratterizzato da assenza di *rumore di processo*: fatta assunzione di conoscere esattamente il valore di a , la doppia integrazione di (3) rispetto al tempo fornisce una descrizione esatta e deterministica della dinamica del sistema: la traiettoria sarà *esattamente* quella individuata dalla soluzione.

Alcuni processi tuttavia evolvono in parte stocasticamente per loro natura, e questa natura stocastica insita nel processo viene chiamata *rumore di processo*. [2]

Si consideri ad esempio il caso in cui la traiettoria di moto del punto materiale descritta dalle soluzioni della (3) venga disturbata dall'azione di forze casuali, quali ad esempio improvvise raffiche di vento.

La conclusione è che in molte applicazioni reali, non solamente le misurazioni sono affette da rumore, ma anche il processo evolutivo stesso può essere affetto da rumore stocastico intrinseco. La conseguenza è che la forma esplicita delle equazioni (1) e (2) contiene un termine casuale individuato da una variabile aleatoria.

Uno schema di un processo caratterizzato da *rumore* è mostrato in figura 9.

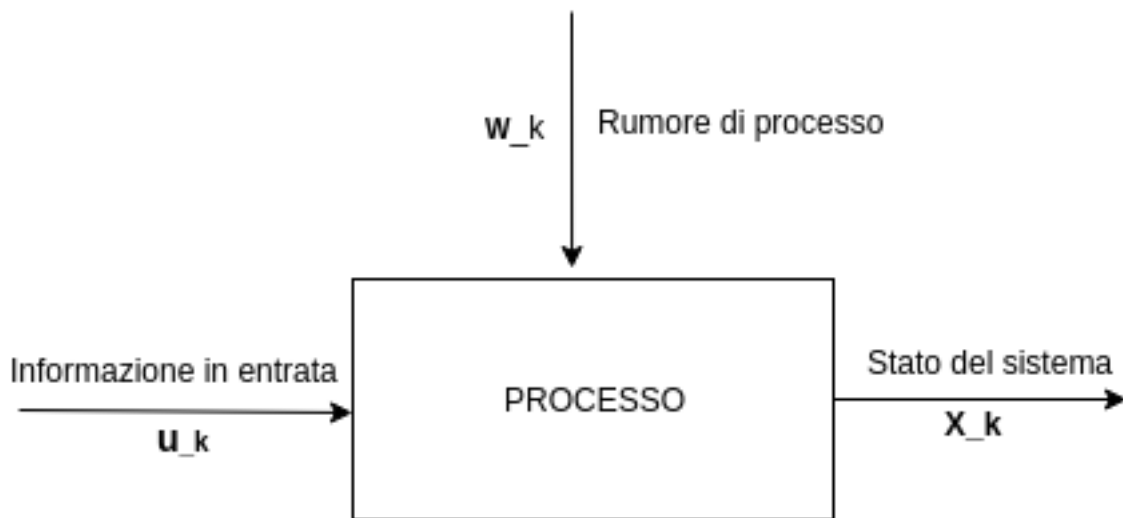


Figura 9: Processo caratterizzato da *rumore*

2.2 I FILTRI DI KALMAN

Un Filtro di Kalman, o in inglese *Kalman Filter* (KF), è uno *stimatore* dello stato di un sistema dinamico caratterizzato da rumore. L'algoritmo

individuato dalla formulazione matematica di un KF può essere usato come SFA.

2.2.1 Premesse statistiche

Un insieme di N sorgenti di misurazioni viene modellato come un insieme di N variabili casuali.

Siano X_1, \dots, X_N N variabili casuali a valori in un insieme finito non vuoto \mathbb{X} , e sia X, Y una qualsiasi coppia presa tra le N variabili casuali. Si chiama *covarianza* di X, Y la seguente quantità:

$$\text{cov}(X, Y) = \mathbb{E}\{[X - \mathbb{E}(X)][Y - \mathbb{E}(Y)]\} = \sigma_{XY}$$

Si osservi che :

$$\text{cov}(X, X) = \mathbb{E}\{[X - \mathbb{E}(X)]X - \mathbb{E}(X)]\} = \mathbb{E}[X - \mathbb{E}(X)]^2 = \sigma_X^2$$

Per un vettore di N variabili aleatorie X_1, \dots, X_N , si definisce la matrice di *varianza-covarianza*, o semplicemente matrice di *covarianza*, la seguente matrice quadrata $N \times N$:

$$\begin{aligned} \Sigma &= \begin{pmatrix} \text{cov}(X_1, X_1) & \text{cov}(X_1, X_2) & \dots & \text{cov}(X_1, X_N) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(X_N, X_1) & \text{cov}(X_N, X_2) & \dots & \text{cov}(X_N, X_N) \end{pmatrix} \\ &= \begin{pmatrix} \sigma_{X_1}^2 & \sigma_{X_1, X_2} & \dots & \sigma_{X_1, X_N} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{X_N, X_1} & \sigma_{X_N, X_2} & \dots & \sigma_{X_N}^2 \end{pmatrix} \end{aligned}$$

Un vettore di N variabili casuali si dice congiuntamente *gaussiano*, ossia distribuito secondo una distribuzione di probabilità *normale multivariata*, quando qualunque combinazione lineare non banale:

$$Y = \sum_{i=1}^N \alpha_i X_i \quad \alpha_i \in \mathbb{R}$$

Ha distribuzione di probabilità *gaussiana*.

2.2.2 Filtro di Kalman Lineare

I KF sono comunemente basati su sistemi dinamici *lineari* a tempo discreto.

In questa sezione viene esposto il principio base del funzionamento di

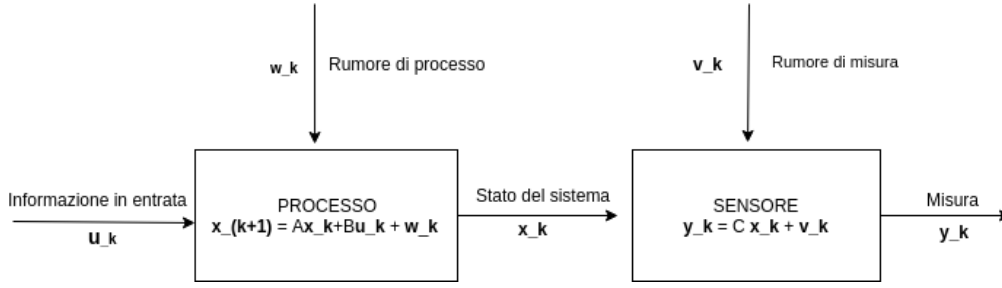


Figura 10: Processo e misura caratterizzati da rumore

un semplice KF lineare, avente lo scopo di stimare lo stato di un sistema dinamico lineare in presenza di rumore di processo e rumore di misura.

Definizione del problema

Si consideri il seguente sistema dinamico lineare discreto:

$$\begin{cases} \mathbf{x}_k = A\mathbf{x}_{k-1} + B\mathbf{u}_k + \mathbf{w}_k \\ \mathbf{y}_k = C\mathbf{x}_k + \mathbf{v}_k \end{cases} \quad (4)$$

In cui i vettori \mathbf{w}_k e \mathbf{v}_k rappresentano rispettivamente il rumore di processo e il rumore di misura. Si assumono *congiuntamente gaussiani*, indipendenti e con matrici di covarianza Q , R rispettivamente.

Il vettore \mathbf{y}_k rappresenta il vettore di misurazioni campionate all'istante k , mentre il vettore \mathbf{x}_k rappresenta lo stato del sistema all'istante k .

Le matrici A e B descrivono la dinamica del modello e si assumono note a priori, pena l'introduzione di errori sistematici, mentre la matrice C descrive la dinamica del processo di misura. Il vettore \mathbf{u}_k rappresenta l'informazione data in ingresso al sistema al tempo k . [4]

Uno schema della (4) ad ogni istante di tempo k è riportato in figura 10.

Dal momento che non è possibile individuare una soluzione analitica di (4), il problema da risolvere è *stimare* lo stato del sistema \mathbf{x}_k per qualunque istante di tempo k .

Soluzione

Si definiscono i vettori:

- $\hat{\mathbf{x}}_k^-$ come la stima *a priori* dello stato del sistema all'istante k , sulla base della conoscenza del processo all'istante $k - 1$;

- $\hat{\mathbf{x}}_k$ come la stima *a posteriori* dello stato del sistema all'istante k , data dalle misurazioni \mathbf{y}_k allo stesso istante.

Si ha che ciascun $\hat{\mathbf{x}}_k^-$ e ciascun $\hat{\mathbf{x}}_k$ è in effetti un vettore di variabili aleatorie.

Siano $\mathbf{e}_k^- = (\mathbf{x}_k - \hat{\mathbf{x}}_k^-)$, $\mathbf{e}_k = (\mathbf{x}_k - \hat{\mathbf{x}}_k)$ rispettivamente l'errore a priori e l'errore a posteriori di stima, e siano P_k^- e P_k rispettivamente le matrici di covarianza di \mathbf{e}_k^- e di \mathbf{e}_k .

Un KF lineare è una quintupla di equazioni:

1. $\hat{\mathbf{x}}_k^- = A\hat{\mathbf{x}}_{k-1}^- + B\mathbf{u}_k$
2. $P_k^- = AP_{k-1}^-A^T + Q$
3. $L_k = P_k^-C^T(CP_k^-C^T + R)^{-1}$
4. $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + L_k(\mathbf{y}_k - C\hat{\mathbf{x}}_k^-)$
5. $P_k = (I - L_kC)P_k^-$

In cui le equazioni 1 e 2 vengono dette *equazioni di predizione*. Esse proiettano lo stato del sistema al tempo $k - 1$ e la covarianza dell'errore di stima a priori al tempo $k - 1$, in avanti all'istante k .

Le equazioni 3, 4, 5, vengono dette *equazioni di aggiornamento*, e costituiscono il modello del meccanismo di *retroazione* con cui il filtro corregge la proprie stime, sulla base delle misurazioni fornite:

- Viene prima calcolata L_k , ossia la *matrice dei guadagni di Kalman*;
- Le misure \mathbf{y}_k vengono usate per determinare una *stima a posteriori* dello stato del sistema all'istante k ;
- Infine viene calcolata una stima della covarianza dell'errore a posteriori P_k .

In figura 11, è riportato uno schema del funzionamento logico di un KF collegato a un sistema dinamico lineare a tempo discreto.

KF è un *filtro ricorsivo*, in quanto la stima $\hat{\mathbf{x}}_k$ dello stato del sistema all'istante k viene determinata combinando le informazioni date dal vettore di misurazioni \mathbf{y}_k campionate all'istante k con la stima dello stato del sistema all'istante $k - 1$ ¹.

KF è uno *stimatore ottimo*, dove ottimo significa che minimizza la covarianza dell'errore di stima a posteriori, se tutti i rumori hanno distribuzione normale multivariata. [2]

¹ Supposto noto lo stato iniziale \mathbf{x}_0 .

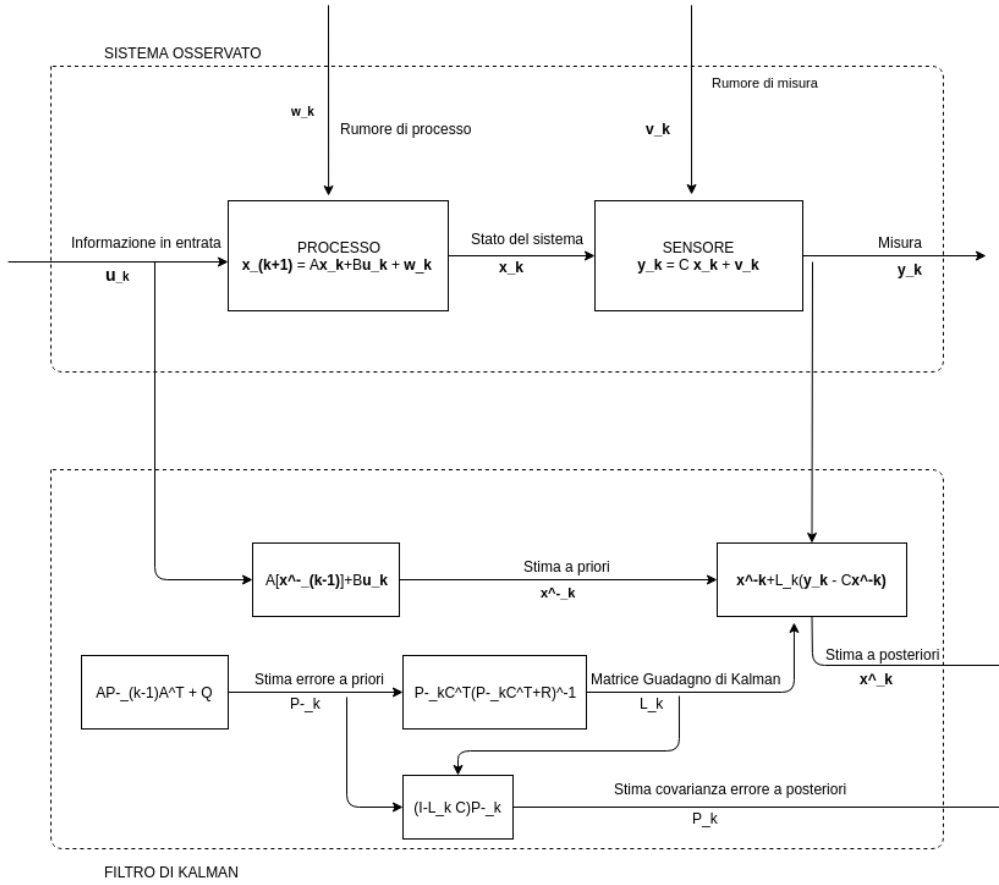


Figura 11: Schema di un KF lineare

2.2.3 Esempio applicativo

Si supponga di voler stimare i valori di posizione e velocità nel sistema dinamico individuato dalla (3). Valgono le seguenti ipotesi:

- Il corpo, modellato come puntiforme, viene lasciato cadere da una quota $z(0) = z_0$ assegnata, con velocità iniziale v_0 ;
- La sola accelerazione a cui è soggetto il corpo è l'accelerazione gravitazionale $-g$;
- Il sistema non è caratterizzato da rumore di processo;
- Un osservatore è in grado di misurare la quota dell'oggetto mediante uno strumento di misura X distribuito come una normale univariata con varianza R ;

- Il sistema viene osservato ogni secondo per un intervallo di tempo lungo t_M s.

Valori numerici:

- $z_0 = 100$ m
- $v_0 = 0 \frac{\text{m}}{\text{s}}$
- $g = 1 \frac{\text{m}}{\text{s}^2}$
- $R = 1 \text{ m}^2$
- $t_M = 10$ s

Si scrivono le equazioni (4) modellanti la dinamica del processo evolutivo e la dinamica del processo di misurazione:

$$\begin{aligned} \begin{cases} \mathbf{x}_k = A\mathbf{x}_{k-1} + B\mathbf{u}_k + \mathbf{w}_k \\ \mathbf{y}_k = C\mathbf{x}_k + \mathbf{v}_k \end{cases} &= \begin{cases} \mathbf{x}_k = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \mathbf{x}_{k-1} + \begin{pmatrix} \frac{1}{2} \\ 1 \end{pmatrix} - g + \underline{0} \\ \mathbf{y}_k = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \mathbf{x}_k + \mathbf{v}_k \end{cases} = \\ &= \begin{cases} \mathbf{x}_k = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \mathbf{x}_{k-1} - \begin{pmatrix} \frac{1}{2} \\ 1 \end{pmatrix} \\ \mathbf{y}_k = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \mathbf{x}_k + \mathbf{v}_k \end{cases} \end{aligned}$$

Il Filtro di Kalman è dato dalle seguenti equazioni:

- 1. $\hat{\mathbf{x}}_k^- = A\hat{\mathbf{x}}_{k-1}^- + B\mathbf{u}_k =$
 $= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \hat{\mathbf{x}}_{k-1}^- - \begin{pmatrix} \frac{1}{2} \\ 1 \end{pmatrix}$
- 2. $P_k^- = AP_{k-1}^-A^T + Q =^2$
 $= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} P_{k-1}^- \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}^T + 0$

² Non essendoci rumore di processo, la matrice Q di covarianza di tale quantità è la matrice nulla.

- 3. $L_k = P_k^- C^T (C P_k^- C^T + R)^{-1} =$

$$= P_k^- \begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} P_k^- \begin{pmatrix} 1 \\ 0 \end{pmatrix}^T + 1 \right]^{-1} =$$
- 4. $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + L_k (\mathbf{y}_k - C \hat{\mathbf{x}}_k^-) =$

$$= \hat{\mathbf{x}}_k^- + L_k \left[\mathbf{y}_k - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \hat{\mathbf{x}}_k^- \right]$$
- 5. $P_k = (I - L_k C) P_k^- =$

$$= \left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - L_k \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right] P_k^-$$

Assegnato lo stato iniziale:

$$\mathbf{x}_0 = (z_0, v_0) = (100, 0)$$

Si ha che la stima a priori dello stato del sistema all'istante 0 vale esattamente lo stato iniziale noto del sistema:

$$\hat{\mathbf{x}}_0^- = \mathbf{x}_0 = (z_0, v_0) = (100, 0)$$

Mentre la matrice di covarianza dell'errore a priori viene inizializzata come la matrice di covarianza della sorgente delle misurazioni:

$$P_0^- = R = 1$$

Supponendo di effettuare, attraverso lo strumento X, le misure riportate in tabella 2, un esempio di implementazione in linguaggio MATLAB³ del KF descritto è mostrato di seguito.

t (s)	1	2	3	4	5	6	7	8	9	10
y_t (m)	100	97.9	94.9	92.7	87.3	81.3	75.8	67.5	59.17	51.1

Tabella 2: Misurazioni di esempio del corpo in caduta libera

³ Le motivazioni della scelta di MATLAB sono da ricercarsi nella natura del linguaggio: esso è fortemente orientato al calcolo numerico e alla manipolazione efficiente di espressioni matriciali.

Codice Soluzione

Si riporta in questa sezione il codice risolutivo del problema individuato.

Listing 2.1: Definizione delle variabili del problema

```
A = [1, 1; 0, 1]; %dynamic of the falling body
B = [1/2; 1]; %dynamic of the falling body
I = eye(2); %identity matrix
C = [1, 0]; %measurment process is given by acquiring directly
      the position
Q = [0, 0; 0, 0]; %no process noise
R = 1; %assumed covariance of measurment source
g = 1; %assumed gravitational acceleration
```

Listing 2.2: Inizializzazione del Filtro di Kalman

```
a_priori_pred_err_cov = R; % covariance of a priori state
      prediction initialized as the covariance of measurment
      source
init_vel = 0; %initial velocity
init_height = 100; %initial height of fall
state = [init_height; init_vel]; %initial state
a_priori_pred = state; %initial a priori state prediction is
      known and given by initial state
t = 1:10; %assuming to observe process each
      second for 10 seconds
m = [100,97.9,94.9,92.7,87.3,81.3,75.8,67.5,59.17,51.1]; %
      measurment vector
predicted_values_height = zeros(1,10); %array where store
      predictions of height
predicted_values_vel = zeros(1,10); %array where store
      predictions of speed
predicted_covariances = cell(1,10); %array where store
      post prediction error covariance
```

Listing 2.3: Algoritmo individuato dalle Equazioni di KF

```

for i=1:length(t)
    %---PREDICT---
    k = t(i); %actual k is the time t(i)
    a_priori_pred = A*a_priori_pred + B*(-g); %predict state
        using the only process input, which is always the
        gravitational acceleration g
    a_priori_pred_err_cov = A*a_priori_pred_err_cov*A' + Q;
        %compute a priori prediction error covariance
    %---PREDICT---

    %---UPDATE---
    update_with = m(i);
    kalman_gain = (a_priori_pred_err_cov * C') * inv(C*
        a_priori_pred_err_cov*C' + R); %compute kalman gain
        matrix
    post_prediction = a_priori_pred + kalman_gain * (
        update_with - C * a_priori_pred); %update
        prediction with actual measurment
    post_pred_err_cov = (I - kalman_gain * C) *
        a_priori_pred_err_cov; %compute post prediction
        error covariance
    predicted_values_height(i) = post_prediction(1); %
        store post_prediction height in array
    predicted_values_vel(i) = post_prediction(2); %
        store post_prediction speed in array
    predicted_covariances{i} = post_pred_err_cov; %
        store post_pred_err_cov in array
    %---UPDATE---
endfor

```

Risultati

Nelle figure 12, 13 viene mostrato il grafico che compara i valori stimati di velocità e posizione con i veri valori delle medesime grandezze; infatti tali valori sono determinabili analiticamente come soluzioni esplicite della (3).

Tali grafici danno un'idea immediata della capacità che ha l'algoritmo di stimare la posizione e la velocità in maniera affidabile anche in presenza di un processo di misura affetto da rumore.

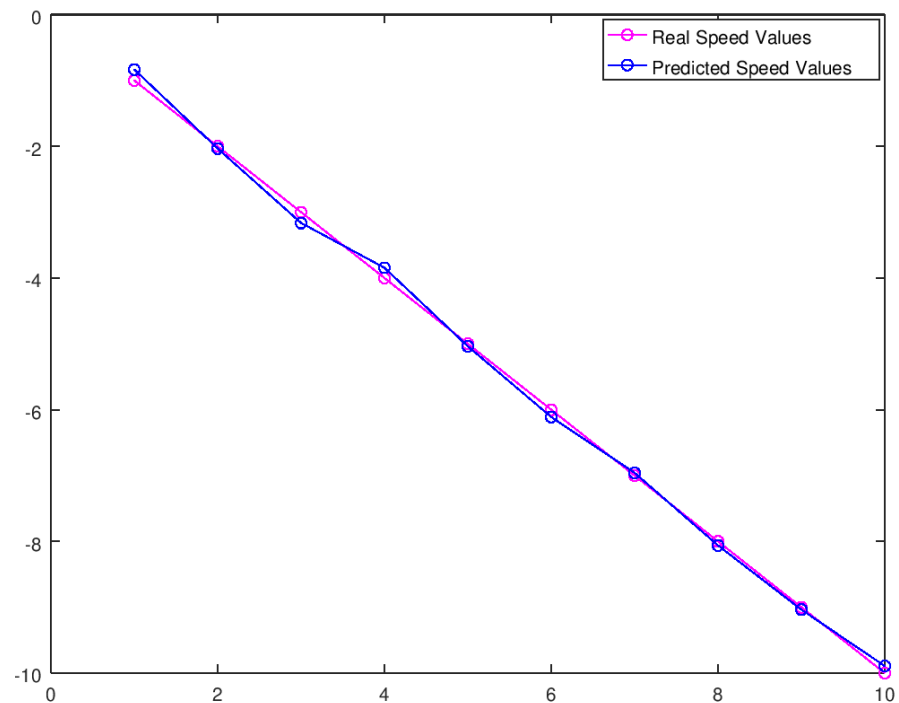


Figura 12: Stima della velocità del corpo in caduta

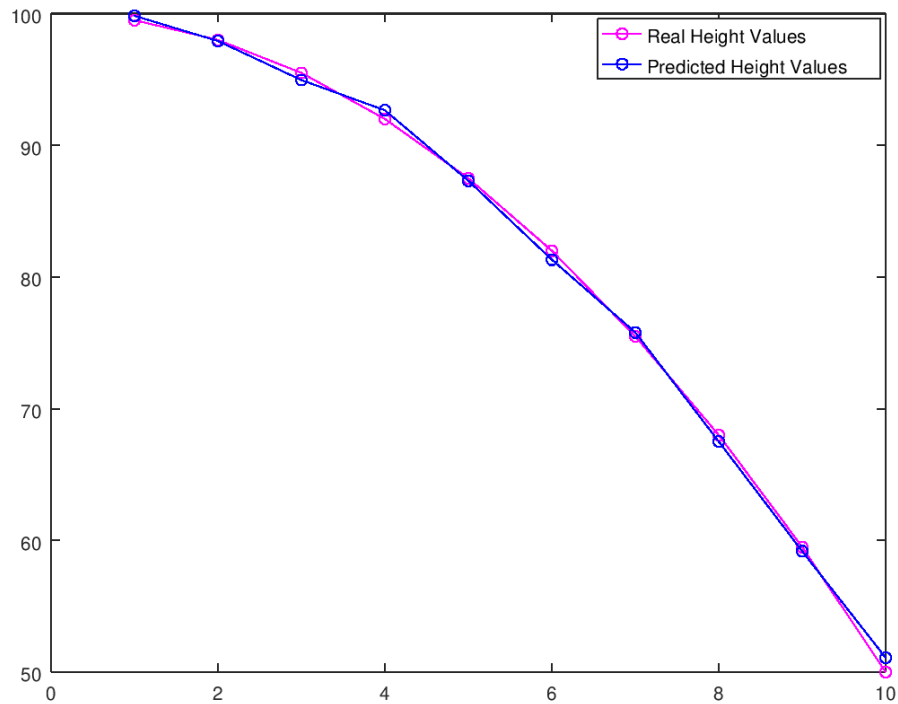


Figura 13: Stima della posizione del corpo in caduta

In tabella 3 viene riportata una sintesi dei dati prodotti dall'esecuzione dell'algoritmo.

Nelle tabelle 4 e 5 è mostrato un paragone fra l'errore a posteriori ($x_k - \tilde{x}_k$) e l'errore della misura rispetto alla posizione reale.

Si noti che per ciascun istante t , l'errore a posteriori sul valore di posizione è minore, in valore assoluto, della distanza fra la misura osservata e la posizione reale del corpo.

Questo porta alla conclusione che l'utilizzo di KF ha permesso di ottenere misure sempre più affidabili rispetto a quelle ottenute dalla sorgente.

t (s)	Misura (m)	Pos.(m)	Stima Pos. (m)	Vel. ($\frac{m}{s}$)	Stima Vel. ($\frac{m}{s}$)
1	100	99.5	99.833	-1	-0.83333
2	97.9	98	97.9167	-2	-2.0333
3	94.9	95.5	94.9545	-3	-3.1636
4	92.7	92	92.6611	-4	-3.8444
5	83.7	87.5	87.3074	-5	-5.037
6	81.3	82	81.3184	-6	-6.1105
7	75.8	75.5	75.7941	-7	-6.9588
8	67.5	68	67.5076	-8	-8.0606
9	59.17	59.5	59.174	-9	-9.0358
10	51.1	50	51.0892	-10	-9.8922

Tabella 3: Risultati generali dell'algoritmo

t (s)	$\hat{\mathbf{x}}_k$	\mathbf{x}_k	$\mathbf{x}_k - \hat{\mathbf{x}}_k$
1	(99.833, -0.83333)	(99.5, -1)	(-0.333333, -0.166667)
2	(97.9167, -2.0333)	(98, -2)	(0.083333, 0.033333)
3	(94.9545, -3.1636)	(95.5, -3)	(0.545455, 0.163636)
4	(92.6611, -3.8444)	(92, -4)	(-0.661111, -0.155556)
5	(87.3074, -5.037)	(87.5, -5)	(0.192593, 0.037037)
6	(81.3184, -6.1105)	(82, -6)	(0.681579, 0.110526)
7	(75.7941, -6.9588)	(75.5, -7)	(-0.294118, -0.041176)
8	(67.5076, -8.0606)	(68, -8)	(0.492424, 0.060606)
9	(59.174, -9.0358)	(59.5, -9)	(0.326024, 0.035783)
10	(51.0892, -9.8922)	(50, -10)	(-1.089216, -0.107843)

Tabella 4: Errori a posteriori

t (s)	Errore a posteriori (pos.) (m)	$\Delta(\text{pos, misura})$ (m)
1	-0.333333	$(99.500 - 100) = -0.5$
2	0.083333	$(98 - 97.9) = 0.1$
3	0.545455	$(95.5 - 94.9) = 0.6$
4	-0.661111	$(92 - 92.7) = -0.7$
5	0.192593	$(87.5 - 87.3) = 0.2$
6	0.681579	$(82 - 81.3) = 0.7$
7	-0.294118	$(75.5 - 75.8) = -0.3$
8	0.492424	$(68 - 67.5) = 0.5$
9	0.326024	$(59.5 - 59.17) = 0.33$
10	-1.089216	$(50 - 51.1) = -1.1$

Tabella 5: Confronto tra errore a posteriori sulla posizione, e distanza fra valori reali di posizione e misure di posizione

2.2.4 Filtri di Kalman non Lineari

I fenomeni reali sono raramente lineari. Un modello lineare è spesso un'approssimazione di un modello più complesso. [3]

Nel dominio applicativo in cui si colloca la Tesi, ossia quello del posizionamento ferroviario, occorre utilizzare una generalizzazione al caso non-lineare dei KF standard.

Si può modellare un sistema dinamico discreto non lineare, caratterizzato da rumore, come:

$$\begin{cases} \mathbf{x}_k = F(\mathbf{x}_k, \mathbf{w}_k) \\ \mathbf{y}_k = H(\mathbf{x}_k, \mathbf{v}_k) \end{cases}$$

In cui, in analogia con il caso lineare:

- \mathbf{x}_k rappresenta lo stato del sistema all'istante k
- \mathbf{w}_k rappresenta il rumore di processo all'istante k
- \mathbf{y}_k rappresenta il vettore di misurazioni campionate all'istante k
- \mathbf{v}_k rappresenta il rumore di misura all'istante k

A differenza del modello lineare, le funzioni F, H che descrivono le dinamiche del processo evolutivo e del processo di misura, non sono in generale lineari. [5]

Esistono diverse forme di KF non lineari, nel seguito della Tesi l'implementazione di SFA cui verrà fatto riferimento, è basata sull'utilizzo di un *Unscented Kalman Filter* (UKF).

APPLICAZIONE DI SFA: LA TRAMVIA DI FIRENZE

In questo capitolo verrà analizzata una particolare applicazione di SFA al problema del posizionamento ferrotramviario.

Nell'ambito di un progetto di ricerca finanziato dall'Unione Europea, si è voluto studiare l'usabilità di SFA come sistema di posizionamento ferrotramviario alternativo a quello descritto nel Capitolo 1, il quale fa un largo uso di apparati installati a terra, fatto che si vorrebbe minimizzare. L'idea di base è quella di utilizzare UKF per stimare la posizione del treno attraverso misure di accelerazione, velocità angolare, velocità lineare e posizione. La rilevazione di tali grandezze, per quanto esposto nel Capitolo 2, è caratterizzata da rumore, UKF combina queste informazioni per stimare la posizione del treno al netto dei rumori di processo e dei rumori di misura.

3.1 ARCHITETTURA DI SISTEMA

Il sistema progettato ha lo scopo di eseguire SFA su una piattaforma hardware installata bordo treno, la quale riceve i dati *raw* dai sensori e li elabora al fine di stimare la progressiva chilometrica del treno in ciascun istante di tempo.

Tale posizione sarà inviata, attraverso un modem LTE:

- All'OBCU, per essere utilizzata attivamente all'interno del sistema di *interlocking*
- Ad un arbitrario host che esegue un software grafico di tracciamento del treno: il RailTrackTool (RTT)

È possibile descrivere l'architettura di sistema a due differenti livelli: architettura a livello *hardware* e architettura a livello *software*.

3.1.1 Architettura Hardware

Sul treno è stata installata una scheda Nvidia TX-Jetson quale piattaforma di elaborazione dei dati. I sensori atti a campionare le misurazioni sono stati collegati alla scheda mediante appositi bus dati.

Il *sensor set* utilizzato in quest'applicazione è composto dai seguenti sensori:

- *Inertial Measurement Unit* (IMU):

Sensore incaricato di misurare i vettori accelerazione (\mathbf{a}) e velocità angolare (\mathbf{v}_{ang}) attraverso l'uso combinato di un accelerometro e un giroscopio. Le misure di IMU sono prese rispetto alla Terra¹ e sono espresse in unità stabilite dallo standard internazionale (SI):

$$\mathbf{a} \left[\frac{\text{m}}{\text{s}^2} \right] \quad \mathbf{v}_{\text{ang}} \left[\frac{\text{rad}}{\text{s}} \right]$$

Si tratta del sensore principale su cui si basa l'esecuzione di SFA. IMU è caratterizzato da un rumore di misura tale per cui, se venisse utilizzato esclusivamente come sorgente di aggiornamento del filtro, la stima della posizione risulta caratterizzata da un *drift lineare*, il quale fa discostare il valore di posizione stimata da quello reale, in una quantità che è funzione lineare del tempo.

- *Odometro*:

Per realizzare l'odometro è stato installato un rilevatore radar su una ruota del treno. Il radar misura il tempo impiegato dalla ruota a compiere un giro completo, e determina la velocità angolare della ruota $\varphi'(t) = \frac{2\pi}{\text{tempo}} \left[\frac{\text{rad}}{\text{s}} \right]$.

Noto il raggio r [m] della ruota, è possibile determinare la velocità lineare alla circonferenza della ruota $x'(t)$ attraverso la relazione cinematica $x'(t) = r\varphi'(t) \left[\frac{\text{m}}{\text{s}} \right] = r\varphi'(t) \left[\frac{\text{m}}{\text{s}} \right]$.

Approssimando il treno come un *corpo rigido*, questa sarà la velocità lineare con cui il treno si sta muovendo.

- *Global Positioning System* (GPS):

Sensore che riceve i dati di posizione attraverso il sistema satellitare GPS.

Le misure di GPS sono riportate in formato standard come tripla

¹ Approssimata come un *sistema inerziale*.



Figura 14: *Inertial Measurement Unit*

di coordinate (latitudine, longitudine, altitudine), rispettivamente espresse in gradi N-S, in gradi E-O e in metri.

In generale queste misure sono le meno affidabili in quanto la *varianza* della variabile aleatoria che modella tale sorgente è la più significativa.

Ad una data frequenza, i sensori inviano dati verso la scheda; quest'ultima, dopo aver eseguito un'iterazione di SFA², invia a OBCU (e/o a RTT) la stima della posizione del treno attraverso apposita modulazione di segnale elettromagnetico, in accordo con il protocollo LTE. Lo schema riportato in figura 15 mostra un diagramma dell'architettura hardware appena descritta.

3.1.2 *Architettura Software*

Sulla scheda è installato il sistema operativo Ubuntu 16.04 LTS, basato su kernel Linux.

Qualunque software menzionato in questa Tesi è stato sviluppato in

² Ossia, un *aggiornamento* di UKF

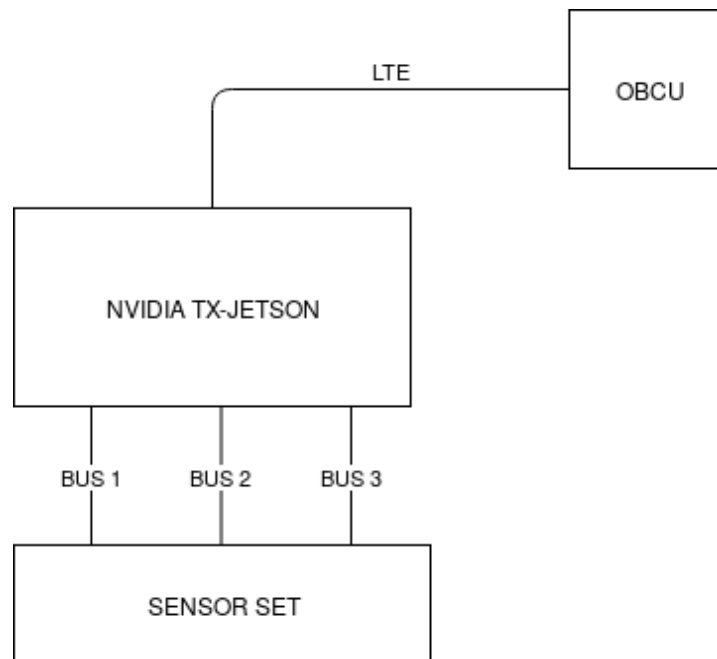


Figura 15: Architettura hardware bordo treno

linguaggio C++.

Un set di tre moduli software, denominati interface-modules, sono in esecuzione sulla scheda.

Sia MOD_i l' i -esimo modulo software del set e $SERIAL_i$ l' i -esima interfaccia seriale della scheda, per $i = 1, 2, 3$.

Il funzionamento di interface-modules è il seguente:

- IMU invia la coppia (accelerazione, velocità angolare) a $SERIAL_1$, MOD_1 legge i valori da $SERIAL_1$ e li invia a un secondo modulo software, denominato listener, attraverso l'interfaccia di rete loopback, in quanto listener esegue anch'esso sulla scheda;
- Odometro invia il valore di velocità lineare a $SERIAL_2$, MOD_2 legge i valori da $SERIAL_2$ e li invia a listener;
- GPS invia i valori di (latitudine, longitudine, altitudine) a $SERIAL_3$, MOD_3 legge i valori da $SERIAL_3$ e li invia a listener.

La comunicazione fra interface-modules e listener avviene attraverso un protocollo applicazione stabilito arbitrariamente, sia esso `INPUT_PROTOCOL`, mentre a livello di trasporto si utilizza UDP.

I valori ricevuti da listener vengono salvati in apposite *strutture dati* rappresentanti misure della stessa sorgente:

- I vettori accelerazione e velocità angolare rilevati da IMU vengono convertiti nella struttura dati IMU_POD;
- La velocità rilevata dal Radar/Odometro viene convertita nella struttura dati ODO_POD;
- La posizione rilevata dal GPS viene infine convertita nella struttura dati GPS_POD.

Il software che esegue effettivamente SFA è compilato come una libreria, *FusionLib*, utilizzata da *listener*. *FusionLib* dispone di interfacce software in entrata e in uscita, ossia *listener* è in grado di inviare le misurazioni a SFA, quali variabili di tipo IMU_POD, ODO_POD, GPS_POD ed altresì di ricevere la stima della posizione del treno, essendo questo l'output dell'algoritmo, quale variabile di tipo SFA_OUTPUT_POD.

Ogniqualvolta *listener* riceva un'uscita da SFA, si fa carico della comunicazione tra scheda e OBCU/RTT. Questa comunicazione, fisicamente possibile attraverso l'utilizzo del modem LTE, avviene utilizzando un protocollo di rete arbitrario a livello applicazione, sia esso OUTPUT_PROTOCOL, mentre al livello di trasporto la scelta è nuovamente ricaduta su UDP per ragioni di efficienza.

Uno schema dell'architettura software è quello mostrato in figura 16.

3.2 GESTIONE DELLA TRASMISSIONE DEI DATI

Nella precedente sezione sono stati brevemente introdotti i protocolli di comunicazione implementati per gestire la comunicazione UDP:

- In entrata, tra interface-modules e *listener* (INPUT_PROTOCOL);
- In uscita, tra *listener* e OBCU/RTT (OUTPUT_PROTOCOL).

3.2.1 *Trasmissione in entrata*

Per trasmettere i dati da interface-modules a *listener*, e dunque dai sensori al modulo software che implementa SFA, è stato realizzato un protocollo di comunicazione denominato INPUT_PROTOCOL.

Tale protocollo fa affidamento a livello trasporto su UDP per massimizzare la velocità di trasmissione senza dover necessariamente rinunciare all'integrità dei messaggi trasmessi, in quanto la comunicazione avviene

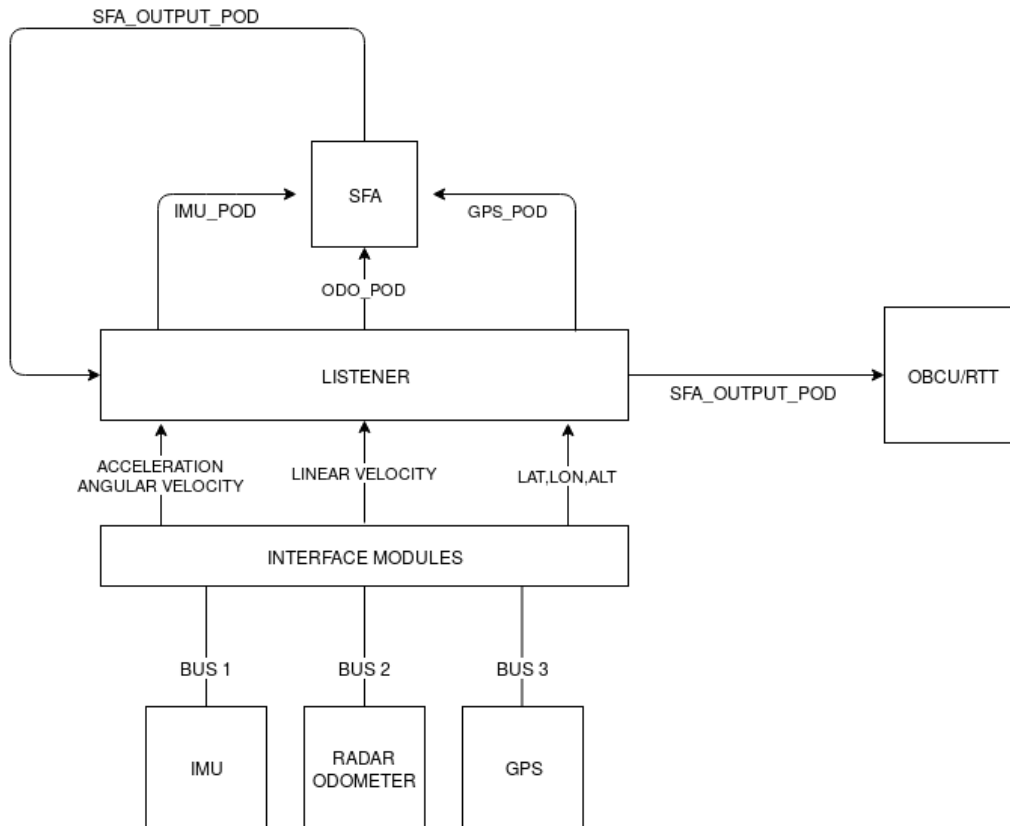


Figura 16: Architettura software bordo treno

tra processi in esecuzione sulla stessa macchina, e la probabilità che un messaggio venga perso o che questo venga ricevuto con errori, è assolutamente trascurabile.

Il protocollo definisce il formato del *payload* del pacchetto UDP che contiene le informazioni di IMU, Radar/Odometro, o GPS, ed è descritto in tabella 6.

A discrezione del valore del campo `SENSOR_TYPE` si distingue il tipo di

Campo	Descrizione	Indici di bit	Tipo
<code>SENSOR_TYPE</code>	ID Sensore Sorgente	0-7	<code>uint8_t</code>
<code>Seq.NO</code>	Numero di sequenza	8-23	<code>uint16_t</code>
<code>N_INT</code>	Numero di interi trasmessi	24-31	<code>uint8_t</code>
<code>N_DOUBLE</code>	Numero di double trasmessi	31-38	<code>uint8_t</code>

Tabella 6: Protocollo di comunicazione in entrata

informazione trasportata dal pacchetto, come descritto in tabella 7.

I pacchetti GROUND TRUTH sono pacchetti di inizializzazione dell'algo-

Valore di SENSOR_TYPE	Sorgente del pacchetto
1	IMU
2	ODOMETRO
3	GPS
8	GROUND TRUTH
9	STROBE
10	STOP

Tabella 7: Significato del campo SENSOR_TYPE

ritmo: alla ricezione del pacchetto GROUND TRUTH l'algoritmo si avvia leggendo i valori trasmessi in coda al pacchetto, in accordo al valore dei campi N_INT e N_DOUBLE. Tali valori forniscono informazioni come progressiva chilometrica e velocità iniziali del treno.

I pacchetti STROBE sono inviati ogni secondo e forniscono un solo valore double, ossia un timestamp che l'algoritmo utilizza per sincronizzarsi.

Il pacchetto STOP non contiene alcuna informazione utile: indica soltanto all'algoritmo di terminare l'esecuzione.

Alla ricezione di un pacchetto, listener legge il valore del campo SENSOR_TYPE, e costruisce, in accordo alla relazione sorgente-struttura dati, la variabile da inviare a SFA.

Il corretto ordinamento dei pacchetti trasmessi a SFA è garantito attraverso l'esplicito utilizzo di un buffer, codificato all'interno di listener, in cui i pacchetti vengono temporaneamente salvati prima di essere inviati a SFA, ed eventualmente ordinati sulla base del valore del campo Seq.NO. Si osservi che se l'integrità non è minacciata dall'utilizzo di UDP quale protocollo di trasporto fra processi all'interno della stessa macchina fisica, altrettanto non si può dire dell'ordinamento dei messaggi. Questi potrebbero subire dei ritardi casuali in base allo stato del sistema operativo, in particolare lo *scheduling* dei processi può avere influenze determinanti sullo scorretto ordinamento dei messaggi trasmessi. Utilizzando TCP si ovvierebbe a questa problematica, ma l'overhead insito nel protocollo stesso causerebbe un notevole degrado delle performance di SFA.

3.2.2 Trasmissione in uscita

La trasmissione dei dati in uscita da SFA avviene, in accordo al protocollo OUTPUT_PROTOCOL tra listener e OBCU, o comunque, tra listener e qualunque host arbitrario che intenda ricevere le informazioni in uscita, come ad esempio un PC sul quale viene eseguito RTT.

Come specificato, la comunicazione è posta in essere, a livello fisico, attraverso il protocollo LTE, ossia un un protocollo *wireless*; mentre a livello trasporto si è scelto di continuare a usare UDP in luogo di TCP, col fine di massimizzare le *performance* del sistema.

Il rischio di ricevere alcune informazioni in maniera errata, o non riceverle del tutto, è nettamente più elevato rispetto allo scenario precedente, nel caso in cui lo spazio fisico attraverso cui si propaga il segnale LTE è tale per cui quest'ultimo venga disturbato da sorgenti esterne.

Gli effetti deleteri di questa condizione sono particolarmente osservabili in alcuni tratti della linea ferrotramviaria, dove possono essere presenti numerose abitazioni e mezzi di trasporto in strada che si interpongono fisicamente tra la scheda NVidia TX-Jetson su cui esegue SFA e l'arbitrario host su cui viene eseguito RTT.

Occorre tuttavia osservare che il tracciamento del treno tramite RTT non è in alcun modo legato alla *safety* del sistema, in quanto le funzionalità *safety-critical* riguardano la comunicazione tra la scheda e OBCU, ossia tra la scheda e il sistema di *interlocking*.

Questa problematica è risolta attraverso l'esplicito utilizzo di un meccanismo di acknowledgment simile a quello utilizzato da TCP: ciascun pacchetto in uscita da SFA viene indicizzato con un *sequence number* e, in ricezione, viene inviato ogni secondo un *ack* replicante l'ultimo numero di sequenza correttamente ricevuto. Solo quando il mittente riceve l'*ack* i dal destinatario invierà il messaggio contenente l'uscita indicizzata con *sequence number* $i + 1$.

Anche in questo caso, il protocollo definisce il formato del *payload* del pacchetto UDP inviato da listener, ed è riportato in tabella 8.

In ricezione dovrà essere inviato il pacchetto *ack* al mittente, ed il suo formato è descritto in tabella 9. Si osserva che SFA produce la stima della posizione del treno sia in termini di progressiva chilometrica che di coordinate ECEF.

ECEF è acronimo di *Earth Centered Earth Fixed* ed è uno standard che misura le coordinate geografiche di un oggetto come la terna $P = (x, y, z)$. Ciascuna coordinata viene espressa considerando la *proiezione su piano* della Terra, e prendendo come origine O l'intersezione fra l'equatore e il

Campo	Descrizione	Indici di bit	Tipo
Seq.NO	Numero di sequenza	0-15	uint16_t
ECEF_X	Coordinata X del treno	16-79	double
ECEF_Y	Coordinata Y del treno	80-143	double
ECEF_Z	Coordinata Z del treno	144-207	double
FU_ARC_LEN	Progressiva chilometrica	208-271	double

Tabella 8: Protocollo di comunicazione in uscita

Campo	Descrizione	Indici di bit	Tipo
ACK	Ultimo Seq.NO	0-15	uint16_t

Tabella 9: Formato del pacchetto di *ack*

meridiano di *Greenwich*.

Le coordinate ECEF misurano tre lunghezze, pertanto, in accordo a SI, esse sono espresse in metri.

Nella prossima sezione, viene descritto uno scenario di esempio del comportamento del sistema a *runtime*.

3.3 SCENARIO DI ESEMPIO

Si suppongano le condizioni iniziali riportate in tabella 10.

Velocità	ECEF	Progressiva	IMU Sample Rate	ODO Sample Rate
0ms^{-1}	(0,0,0) m	0 km	100 Hz	20 Hz

Tabella 10: Condizioni iniziali

1. $t = 0$:

- interface-modules invia a listener il seguente pacchetto GROUND TRUTH:

SENSOR_TYPE	Seq. NO	N_INT	N_DOUBLE
0x08	0x00	0	5

E vi accoda i seguenti tre valori double: 0.0, 0.0, 0.0 ossia

le coordinate ECEF iniziali, il seguente valore double: 0.0, ossia la velocità lineare iniziale, e infine il valore double: 0.0 che rappresenta la progressiva chilometrica iniziale.

- listener riceve il pacchetto e inizializza SFA con:
 - ECEF iniziali: (0,0,0)
 - Velocità lineare iniziale: 0.0
 - Progressiva chilometrica iniziale: 0.0

2. $t = t_0$:

- IMU campiona il seguente vettore accelerazione:

$$\mathbf{a} = (0.0001, -0.0001, -9.8100)$$

Assieme al seguente vettore velocità angolare:

$$\mathbf{v}_{\text{ang}} = (0.0003, -0.0001, 0.0002)$$

E lo invia, tramite SERIAL_1, a MOD_1 di interface-modules.

- MOD_1 invia a listener il seguente pacchetto IMU:

SENSOR_TYPE	Seq. NO	N_INT	N_DOUBLE
0x01	0x01	0	6

Accodandovi nell'ordine il vettore accelerazione, e il vettore velocità angolare.

- listener riceve il pacchetto, crea e invia a SFA la seguente variabile IMU_POD:
 - Seq.NO = 1
 - Epoch = t_0
 - ACC_X = 0.0001
 - ACC_Y = -0.0001
 - ACC_Z = -9.8100
 - GYRO_X = 0.0003
 - GYRO_Y = -0.0001
 - GYRO_Z = 0.0002

- SFA elabora il pacchetto e inizia una computazione parallela per fornire a listener una variabile SFA_OUTPUT_POD della forma:

- Seq.NO = 0
- ECEF_X = E_X
- ECEF_Y = E_Y
- ECEF_Z = E_Z
- FU_ARC_LEN = P_{KM}

3. $t_0 < t < t_0 + \frac{1}{ODO_SAMPLE_RATE} = t_0 + \frac{1}{20}$

Fintantoché l'odometro non campiona il suo primo valore di velocità, si ripetono le operazioni viste al passo precedente per ogni campionamento di IMU.

4. $t = t_0 + \frac{1}{20}$

- Odometro campiona il seguente valore di velocità:

$$a = (1.0010)$$

E lo invia, tramite SERIAL_2, a MOD_2 di interface-modules.

- MOD_2 invia a listener il seguente pacchetto ODOMETRO:

SENSOR_TYPE	Seq. NO	N_INT	N_DOUBLE
0x02	Seq_NO	0	2

Accodandovi nell'ordine il valore di velocità rilevato, e il valore dello scarto quadratico medio della sorgente, noto a priori, in quanto caratteristica tecnica intrinseca dello strumento di misura, il radar; sia esso SIGMA_RADAR.

- listener riceve il pacchetto, crea e invia a SFA la seguente variabile ODO_POD:

- Seq.NO = Seq_NO
- Epoch = $t_0 + \frac{1}{20}$
- vel = 1.0010
- sigma = SIGMA_RADAR

- SFA elabora il pacchetto e utilizza la rilevazione di velocità in maniera utile a correggere il *drift* di IMU, al fine di produrre una stima della posizione più accurata.

5. $t = n t_0 \quad n \in \mathbb{N}^+$

Ogni secondo, il modulo STROBE di interface-modules, invia a listener un pacchetto della forma:

SENSOR_TYPE	Seq. NO	N_INT	N_DOUBLE
0x09	Seq_NO	0	1

Accondandovi un *timestamp* che listener inoltra a SFA per scopi di sincronizzazione.

Quanto elencato viene ripetuto per ciascun campionamento successivo di IMU e odometro.

Non appena un'uscita di SFA si rende disponibile a listener questo si comporta come segue:

- listener riceve la variabile SFA_OUTPUT_POD, da SFA;
- listener costruisce il seguente pacchetto da inviare a OBCU, o a RTT:
 - Seq.NO = 0x00
 - ECEF_X = SFA_OUTPUT_POD.E_X
 - ECEF_Y = SFA_OUTPUT_POD.E_Y
 - ECEF_Z = SFA_OUTPUT_POD.E_Z
 - FU_ARC_LEN = SFA_OUTPUT_POD.P_{KM}
- OBCU, o RTT, riceve il pacchetto e invia a listener l'*ack* 0x00.

3.4 POSSIBILI SVILUPPI

Il sistema, così come è stato descritto, rappresenta essenzialmente un *core* minimale di un sistema di posizionamento basato su SFA, limitato rispetto alle potenzialità dell'algoritmo e comunque non esente da vulnerabilità legate alla *security*. In questa sezione verranno discusse le principali problematiche della soluzione descritta, in che modo queste possono essere risolte, e quali tecniche possono essere usate per migliorare l'usabilità del sistema.

3.4.1 Problematiche legate alla security

Per *security* si intende un insieme di tecniche che hanno come scopo la protezione dei dati, siano essi stoccati in un sistema informatico, oppure transitanti attraverso un sistema di telecomunicazione.

Tale protezione viene assicurata contro specifiche *minacce*, le quali sfruttano opportune *vulnerabilità*.

La *security* viene garantita attraverso l'uso di appropriate *tecniche preventive*, oppure *contromisure* applicabili in caso di violazioni alle principali *misure della security*:

- Integrità
- Confidenzialità
- Autenticazione

In un sistema *safety-critical* come quello descritto, una violazione di *security* potrebbe portare a una violazione di *safety*, pertanto è fondamentale ridurre al minimo le vulnerabilità del sistema. Nella fattispecie descritta in questa Tesi, tuttavia, la confidenzialità non è una misura fondamentale, mentre lo sono l'integrità e l'autenticazione.

Minacce all'integrità

È stato già discusso che l'utilizzo del protocollo UDP a livello di trasporto, non garantisce affatto che i messaggi ricevuti da OBCU siano corretti e ordinati.

Per ovviare al problema dell'ordinamento è stato implementato il già descritto meccanismo di acknowledgment, tuttavia esso fa l'implicita assunzione che se si è in grado di leggere correttamente il numero di sequenza del pacchetto ricevuto, questo non sia stato alterato.

Si consideri il seguente scenario:

- listener invia a OBCU il seguente pacchetto:
 - Seq.NO = 0x17
 - ECEF_X = SFA_OUTPUT_POD.E_X
 - ECEF_Y = SFA_OUTPUT_POD.E_Y
 - ECEF_Z = SFA_OUTPUT_POD.E_Z
 - FU_ARC_LEN = SFA_OUTPUT_POD.P_{KM}

- OBCU riceve il seguente pacchetto:
 - Seq.NO = 0x25
 - ECEF_X = SFA_OUTPUT_POD.E_X
 - ECEF_Y = SFA_OUTPUT_POD.E_Y
 - ECEF_Z = SFA_OUTPUT_POD.E_Z
 - FU_ARC_LEN = SFA_OUTPUT_POD.P_{KM}

Per come è stato descritto il protocollo, OBCU accetta passivamente che il numero di sequenza ricevuto sia 0x25, anche se prima di questo era stato letto il valore 0x16, ed invierà a listener l'*ack* 0x25.

In questo caso, OBCU dovrebbe essere progettato in maniera tale da controllare sempre di ricevere un numero di sequenza pari all'ultimo ricevuto +1. Dal momento che, viste le caratteristiche intrinseche del protocollo, è impossibile che listener abbia inviato il pacchetto con numero di sequenza 0x25 se l'ultimo *ack* ricevuto non era 0x24, è probabile che, attraversando il canale, il pacchetto abbia subito alterazioni casuali in tutti i suoi bit, e quindi anche l'informazione di posizione potrebbe essere alterata.

Per ovviare definitivamente alla problematica dell'integrità, è opportuno integrare nel protocollo l'uso di una *funzione hash*. Il protocollo verrebbe modificato come segue:

- listener prepara il pacchetto contenente le informazioni di SFA_OUTPUT_POD;
- listener calcola $H(\text{SFA_OUTPUT_POD}) = y$;
- listener invia la coppia (SFA_OUTPUT_POD, y)

In ricezione, OBCU ricalcola $H(\text{SFA_OUTPUT_POD}) = y'$, e accetta il messaggio solo se $y' = y$. Infatti, grazie alla proprietà delle funzioni *hash*, una minima variazione del messaggio m causa una grande variazione del *digest* $H(m)$, quindi è altamente improbabile che un'alterazione casuale dei bit trasmessi, sia essa (SFA_OUTPUT_POD_WRONG, Y_WRONG), mantenga la proprietà $H(\text{SFA_OUTPUT_POD_WRONG}) = Y_WRONG$.

Minacce all'autenticazione

Si consideri il caso in cui un malintenzionato sia in grado di inviare messaggi a OBCU e abbia interesse nel non segnalare al sistema di *interlocking* l'avvicinamento del treno alla JA.

L'attaccante si comporta come segue:

- Intercetta il messaggio ($SFA_OUTPUT_POD, H(SFA_OUTPUT_POD)$)
- Modifica la posizione del treno ponendola lontano da una JA, forgiando un nuovo messaggio, sia esso $SFA_OUTPUT_POD_DANGEROUS$
- Calcola $H(SFA_OUTPUT_POD_DANGEROUS) = Y_DANGEROUS$
- Invia a OBCU la coppia ($SFA_OUTPUT_POD_DANGEROUS, Y_DANGEROUS$)

Per ovviare a questa problematica si potrebbero usare le seguenti tecniche:

1. Cifratura del *digest* della funzione *hash* con una chiave simmetrica condivisa tra listener e OBCU;
2. Cifratura del *digest* della funzione *hash* con la chiave privata di listener (Firma Digitale DSA);
3. Uso di una funzione *hash* che prende in ingresso sia il messaggio che una chiave simmetrica condivisa tra listener e OBCU (HMAC);
4. Accodare un segreto condiviso tra listener e OBCU al messaggio prima di calcolarne il *digest*.

In ciascuno di questi scenari, fatta assunzione di proprietà di *strong collision resistance* della funzione *hash* utilizzata, si garantisce che il messaggio può essere stato inviato solo da listener, in quanto un attaccante non avrebbe modo di modificare il messaggio e calcolare un *digest* valido. La soluzione meno dispendiosa in termini di complessità computazionale e più adatta a un simile scenario è la soluzione 4, in quanto non è necessario garantire anche la *non-ripudiabilità* ma solo l'autenticazione e l'integrità.

3.4.2 Miglioramenti al protocollo in uscita

Il sistema realizzato esegue una versione di SFA basata su un UKF. Ad ogni campionamento di IMU, Odometro, o GPS, viene eseguita la fase di *aggiornamento* del filtro che comprende la valutazione di quantità interessanti dal punto di vista di valutazione delle *performance* dell'algoritmo. In particolare, si potrebbe analizzare il comportamento di SFA al variare di parametri come:

- La frequenza di aggiornamento;
- Il tipo di misure con cui viene effettivamente aggiornato il filtro.

Nel classico scenario di utilizzo, che comprende la comunicazione con OBCU, l'unica informazione effettivamente utile è la progressiva chilometrica.

In uno scenario di *acceptance test*, naturalmente preliminare al rilascio del sistema, potrebbe essere utile fornire a RTT, oltre alle informazioni sulla posizione, anche i valori in entrata all'algoritmo che hanno prodotto il risultato di posizione.

Inoltre, se fosse trasmessa anche la matrice di covarianza della stima a posteriori, verrebbero fornite indicazioni importanti sul comportamento qualitativo dell'algoritmo.

Queste modifiche comporterebbero una minima variazione di OUTPUT_PROTOCOL in quanto occorrerebbe semplicemente prevedere dei campi aggiuntivi al formato del pacchetto, che si aggiungerebbero ai campi relativi alle coordinate ECEF e alla progressiva chilometrica.

RISULTATI SPERIMENTALI

In questo capitolo vengono discussi i risultati degli esperimenti effettuati sul sistema.

Gli esperimenti sono divisi principalmente in due categorie:

- Esperimenti *offline*:

I dati di IMU, Odometro e GPS vengono generati attraverso RTT e passati a un'istanza locale di FusionLib.

I risultati forniti da SFA vengono mostrati su RTT in forma grafica.

Questi esperimenti sono essenzialmente *Integration Test* volti a validare l'implementazione di SFA e vengono eseguiti interamente con RTT, il quale genera un report per ogni esperimento effettuato.

In figura 17 si mostra lo schema logico di un esperimento effettuato con RTT.

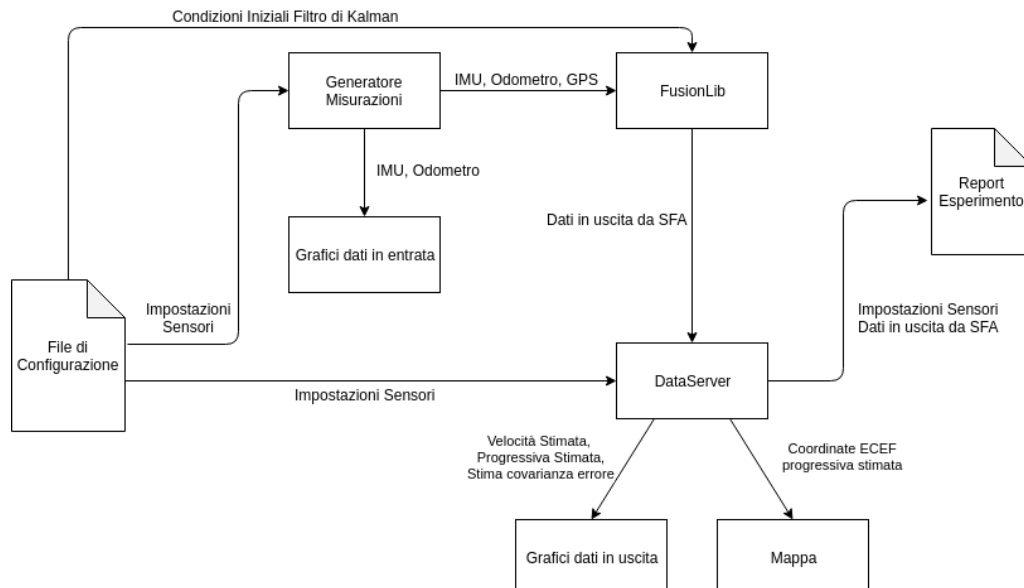


Figura 17: Esperimenti offline

- Esperimenti *online*:

Il software SynthDataGenApp (SDGA), essenzialmente un'opportuna estrazione di codice da RTT, è un software capace di generare i dati di IMU, Odometro e GPS.

SDGA si comporta come un'astrazione di interface-modules, e le misurazioni prodotte vengono inviate in accordo a INPUT_PROTOCOL a una scheda NVidia TX-Jetson su cui è replicato l'ambiente HW e SW bordo treno. Tali dati sono inviati alla scheda attraverso un'interfaccia ethernet, utilizzata in luogo dell'interfaccia loopback prevista dal sistema reale.

La scheda invia i dati in uscita da SFA, attraverso la medesima interfaccia ethernet, al mittente dei dati in entrata in accordo a OUTPUT_PROTOCOL.

In questo caso, l'interfaccia ethernet sostituisce l'utilizzo del modem LTE.

Questi esperimenti sono essenzialmente *End to End Test* e hanno lo scopo di validare i protocolli di comunicazione.

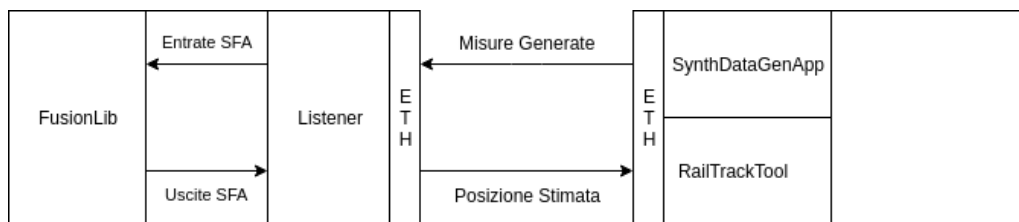


Figura 18: Esperimenti online

La linea ferrotramviaria scelta come ambiente di prova è la linea T1 della Tramvia di Firenze, che collega la stazione di *Villa Costanza*, sita nel comune di Scandicci, all'ospedale di *Careggi*, sito quest'ultimo nel comune di Firenze. La linea è mostrata in figura 19.

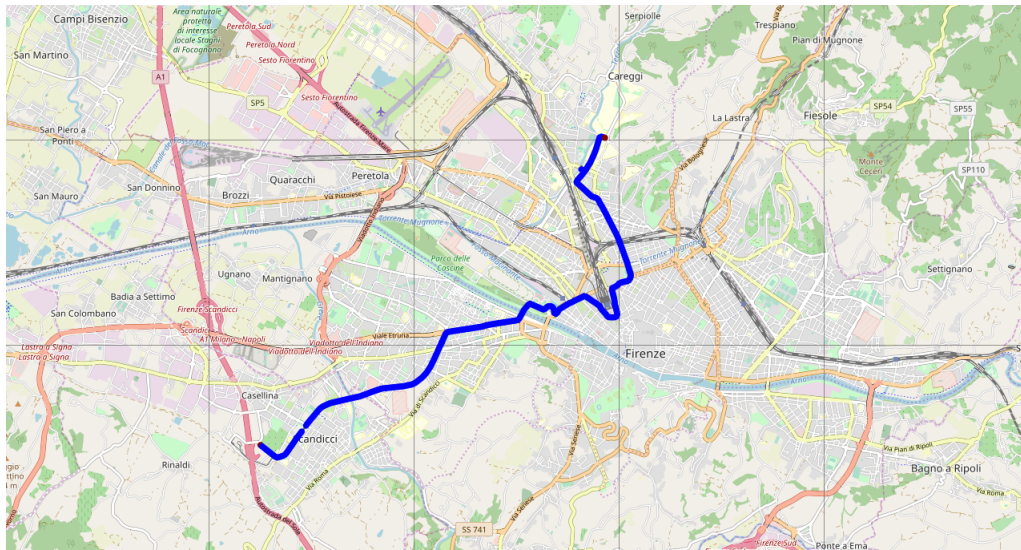


Figura 19: Tramvia di Firenze - Linea T1

4.1 SOFTWARE IMPIEGATI

In questa sezione vengono descritti i SW utilizzati negli scenari sopra descritti.

4.1.1 *FusionLib*

Come accennato nel Capitolo 3, FusionLib è una *libreria software* che implementa SFA. Tale libreria viene compilata sia su Windows 10, su cui esegue RTT, che su Ubuntu 16.04 LTS, su cui esegue listener. FusionLib è pertanto una libreria inclusa da listener e RTT.

Per determinare la posizione del treno lungo la traccia FusionLib utilizza le informazioni ricevute dal chiamante, quindi le misure, combinate con le informazioni della traccia su cui si assume il treno si stia muovendo. Tali informazioni sono salvate in un *database SQL*. Lo schema logico è riportato in figura 20.

La traccia entro cui il treno si muove è rappresentata da una *spline*. Una *spline* è una funzione polinomiale a tratti che interpola una serie di N coppie (x_i, y_i) , dette *nodi*, utilizzando differenti funzioni polinomiali per ciascun intervallo (x_i, x_{i+1}) per $i = 1, \dots, N - 1$.

Nelle figure 21 e 22, si mostra come una spline polinomiale sia in grado di approssimare in maniera accettabile una funzione sufficientemente

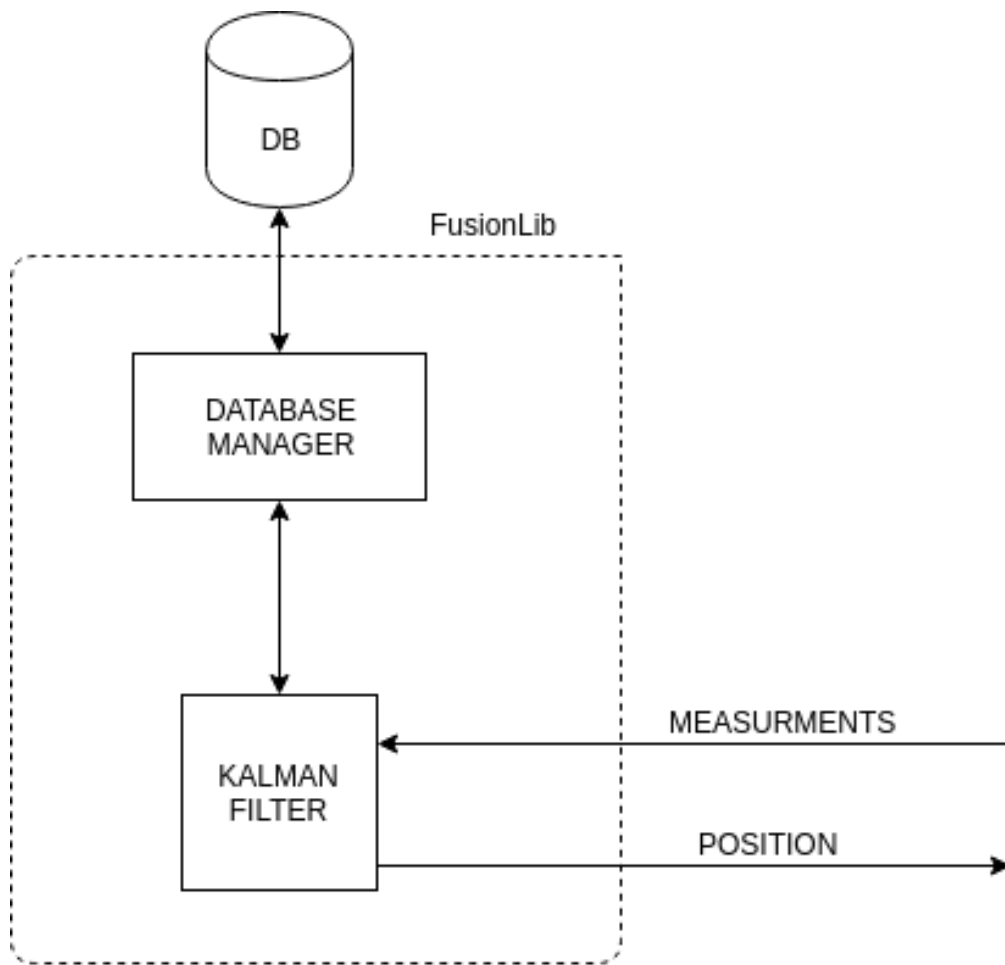


Figura 20: Architettura logica di FusionLib

regolare.

La logica è quella di campionare, per una data traccia, una serie di coppie $(x_i, y_i) = (\text{longitudine}, \text{latitudine})$ di punti geografici appartenenti alla traccia. Queste coppie vengono salvate nel database insieme al valore P_k di progressiva chilometrica che caratterizza ciascuna coppia. Durante l'esecuzione di SFA, il database viene consultato per recuperare il valore corretto delle coordinate geografiche attuali del treno:

- La k -esima ricezione di misure causa l'aggiornamento del filtro;
- L'aggiornamento del filtro produce la k -esima stima della pro-

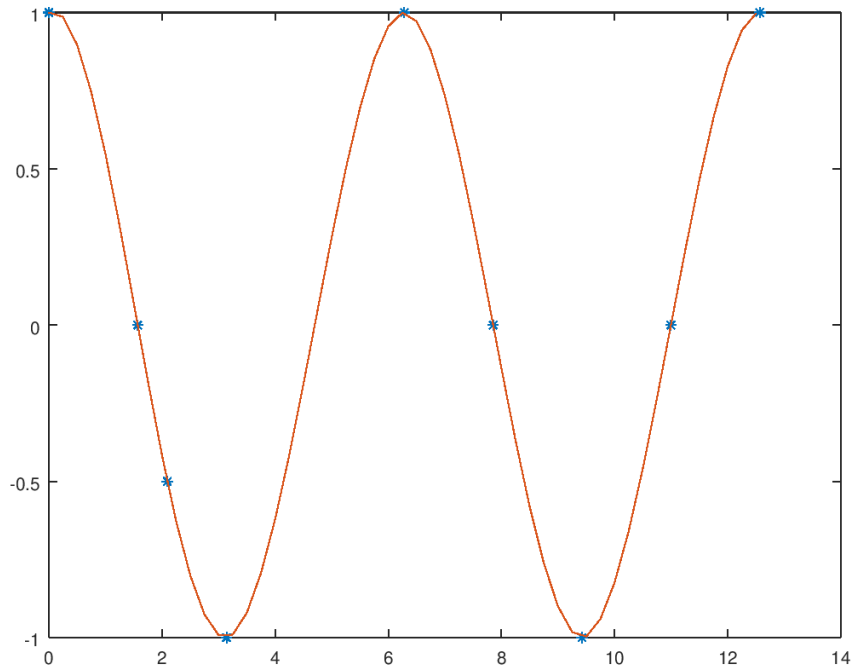


Figura 21: Spline interpolante la funzione $\cos(x)$ nelle ascisse $\frac{k\pi}{2}$ $k = 0, \dots, 8$

gressiva chilometrica attuale: \hat{x}_k . La distanza percorsa rispetto all'iterazione precedente è data da:

$$D_k = \hat{x}_k - \hat{x}_{k-1}$$

- Dal database vengono recuperate le coppie (x_0, y_0) , (x_1, y_1) della traccia, tali per cui:
 1. La progressiva chilometrica associata alla coppia (x_0, y_0) è il massimo P_k tale per cui $P_k < \hat{x}_k$
 2. La prorogressiva chilometrica associata alla coppia (x_1, y_1) è il minimo P_k tale per cui $P_k > \hat{x}_k$

In altre parole, si è determinato l'intervallo della spline in cui il treno è stimato trovarsi al passo k .

È opportuno precisare che le coppie ottenute dal database non sono in formato ECEF, infatti esse sono espresse in gradi. Tuttavia, è possibile applicare una nota trasformazione per convertire la coppia $(x_i, y_i) = (\text{longitudine}, \text{latitudine})$ nel formato ECEF.

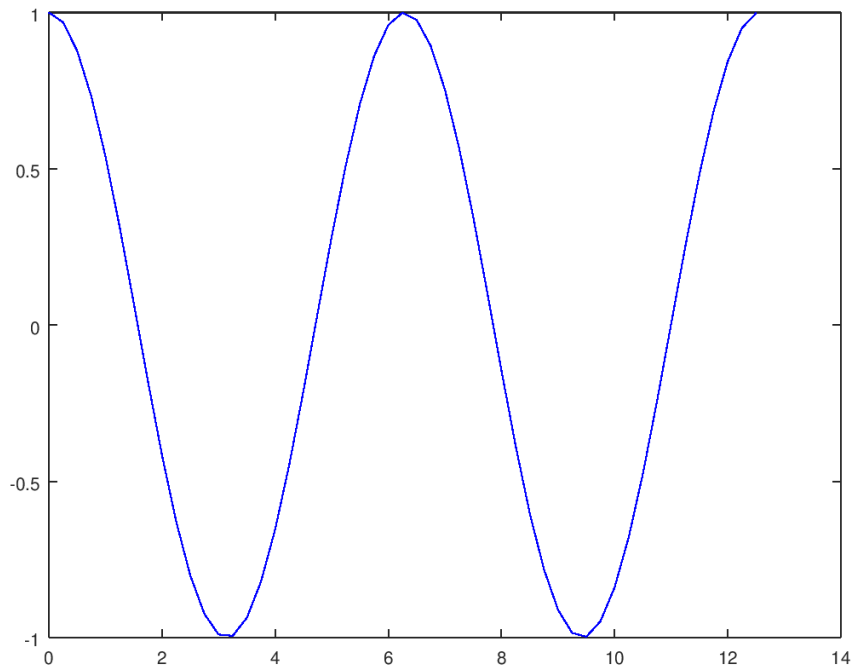


Figura 22: Funzione $\cos(x)$ nell'intervallo reale $[0, 4\pi]$

- Date le coppie individuate, se ne determina il polinomio interpolante¹. Sia esso $p(x)$.
- Le coordinate ECEF al passo k , (x_k, y_k) , sono date da:

$$\begin{cases} x_k = x_0 + D_k \\ y_k = p(x_0 + D_k) = p(x_k) \end{cases}$$

4.1.2 RTT

RTT ha più modalità di funzionamento:

1. INTERFACE_EXT_FILTER:

In questa modalità, RTT funziona come interfaccia verso un'istanza esterna di FusionLib per visualizzare le uscite di SFA su una mappa. I dati sono letti da una *socket* UDP in accordo al protocollo

¹ Si osservi che è sempre possibile, a condizione che $x_0 \neq x_1$.

OUTPUT_PROTOCOL.

In questo caso RTT ha il ruolo di OBCU.

2. INTERFACE_SENSOR:

In questa modalità RTT si interfaccia verso interface-modules, o comunque qualunque processo che sia in grado di fornire misure di IMU, Odometro e GPS via UDP, in accordo al protocollo INPUT_PROTOCOL. I dati ricevuti vengono inviati alla propria istanza di FusionLib durante un'esecuzione di SFA.

In questo caso RTT ha il ruolo di listener.

3. STANDALONE:

Modalità standard di funzionamento, come descritta in figura 17.

In questa modalità, RTT genera misure le misure da inviare alla propria istanza di FusionLib per eseguire SFA.

In questo caso RTT simula l'intero processo del sistema di posizionamento.

La modalità INTERFACE_EXT_FILTER viene usata per visualizzare graficamente i risultati degli esperimenti *online*.

La modalità STANDALONE viene usata per gli esperimenti *offline*.

Per quanto riguarda la modalità INTERFACE_SENSOR, essa non ha una particolare applicabilità sperimentale, infatti questa modalità è stata principalmente usata per scopi di *debug* dei protocolli di comunicazione. Questo aspetto sarà discusso in seguito.

4.1.3 Listener

Modulo in esecuzione su NVidia TX-Jetson, il cui funzionamento è stato descritto nel Capitolo 3.

Ricapitolando, listener riceve le misure dei sensori in accordo a INPUT_PROTOCOL ed esegue SFA, aggiornando il KF con le misure ricevute in ingresso.

Non appena viene calcolata un'uscita di SFA, questo software è incaricato di inoltrare tale uscita a OBCU o a RTT. Negli scenari sperimentali descritti in questo Capitolo, le uscite di SFA verranno inviate a RTT.

4.1.4 SDGA

Semplice software estratto da RTT. Il compito di SDGA è generare le misure ed inoltrarle a listener, secondo il protocollo INPUT_PROTOCOL.

4.2 ESPERIMENTI OFFLINE

Lo scopo degli esperimenti offline era quello di validare l'implementazione di SFA e di valutare il suo comportamento al variare dei parametri di configurazione.

4.3 ESPERIMENTI ONLINE

CONCLUSIONI

Lo scopo della Tesi era quello di mostrare un sistema di posizionamento ferrotramviario basato su SFA, alternativo al sistema attualmente in uso. Nel Capitolo 1, definiti i sistemi ferroviari e i sistemi ferrotramviari, le loro similitudini e le loro differenze; è stato introdotto il problema del posizionamento: determinare la posizione di un treno lungo una traccia. Questo problema è di fondamentale importanza poichè il posizionamento di un treno attiva il sistema di *interlocking*, il quale deve garantire un attraversamento sicuro e corretto delle JA, al fine di evitare fallimenti catastrofici.

È stato descritto lo stato dell'arte nell'ambito del posizionamento ferrotramviario e ne sono state evidenziate le relative criticità; si è dunque introdotto il concetto di SFA quale sistema di posizionamento ferroviario atto a risolvere molte delle criticità esposte.

Nel Capitolo 2 è stato formalizzato, da un punto di vista matematico, il concetto di sistema dinamico, quale è un treno in movimento lungo una traccia ferrotramviaria. In particolare, sono stati formalizzati i concetti di rumore di processo e rumore di misura. In quest'ottica, sono stati introdotti i KF come base matematica di un algoritmo in grado di stimare lo stato di un sistema dinamico caratterizzato da rumore intrinseco, attraverso misurazioni caratterizzate anch'esse da rumore..

Nel Capitolo 3 è esposta un' architettura essenziale di un sistema di posizionamento ferrotramviario che sia in grado di sfruttare un algoritmo SFA.

Sono state evidenziate le criticità di tale architettura e le relative mitigazioni.

Il Capitolo 4 rappresenta infine la parte *sperimentale* della Tesi: sono mostrati gli esperimenti effettuati, i risultati ottenuti, e di conseguenza gli argomenti a supporto dell'utilizzo di SFA come sistema di posizionamento ferrotramviario.

BIBLIOGRAFIA

- [1] A. Mirabadi, N. Mort, F. Schmid, *Application of sensor fusion to railway systems*, IEEE, Conference Paper, 1996. (Cited on page 17.)
- [2] B. D. O. Anderson, J. B. Moore, *Optimal Filtering*, Thomas Kailath Editor, 2005. (Cited on pages 17, 20, and 23.)
- [3] L. Brugnano, *Modelli numerici per la simulazione*, Masterbooks, 2018. (Cited on pages 17, 18, and 33.)
- [4] G. Welch, G. Bishop, *An Introduction to the Kalman Filter*, Conference Paper, 2004 (Cited on page 22.)
- [5] Eric A. Wan, *The Unscented Kalman Filter for Nonlinear Estimation*, Conference Paper, 2000. (Cited on page 33.)