UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA
ELETTRONICA

Localizzazione di robot autonomi tramite *Particle Swarm Optimization*

Relatore: Prof. Gianluigi Pillonetto

Laureando: Pietro Ferraresi

Anno Accademico 2024/2025

Data di Laurea: 12 novembre 2024

Estratto

Una corretta ed efficace localizzazione è necessaria per i robot autonomi in quanto permette loro di muoversi, sia in ambienti conosciuti che sconosciuti, nella giusta direzione ed evitando possibili ostacoli. Il progetto seguente si propone di introdurre il problema della localizzazione e come può essere risolto, spiegando i principi sui quali si fonda ed esplorando le classi di strategie più utilizzate ad oggi, iniziando dalle tecniche tradizionali per giungere in seguito a quelle più recenti e innovative. Inoltre, viene proposto un metodo basato sull'algoritmo di ottimizzazione *Particle Swarm Optimization*, di cui verranno analizzati i risultati ottenuti sia da simulazioni che da esperimenti e sarà paragonato ad altri due metodi particolarmente diffusi, rispettivamente basati sull'algoritmo *Particle Filter* e su Filtro di Kalman Esteso.

Indice

1 Introduzione ai principi della localizzazione

1.1 Il Problema della Localizzazione

Con l'introduzione e la diffusione dei robot autonomi in svariati settori, tra cui industriale, medico e spaziale, sta diventando sempre più necessario la ricerca di strategie efficaci che risolvano il cosiddetto problema della localizzazione. La corretta navigazione autonoma prevede che il robot sia in grado di capire la posizione in cui si trova nell'ambiente, il che avviene tracciando costantemente i movimenti compiuti e tramite osservazioni di ciò che lo circonda e, successivamente, muoversi nella direzione desiderata grazie ad una precisa regolazione delle uscite del motore. Il problema della localizzazione coincide con la prima fase della navigazione e, sulla base delle informazioni note riguardo la posizione iniziale del robot, può essere suddiviso in tre sottoproblemi:

- *Position Tracking*: nel caso in cui la posizione iniziale sia nota, l'unico obiettivo è quello di tracciare il percorso compiuto dal robot nel tempo. Questo è reso possibile dagli algoritmi di localizzazione che, durante la navigazione, utilizzano la posizione all'istante precedente e i dati odometrici e sensoriali per monitorare costantemente la posizione del robot. Nel problema del *Position Tracking* è fondamentale che l'incertezza sia bassa per consentire una corretta localizzazione e, di conseguenza, una traccia del percorso affidabile;
- *Global Localization*: in questo tipo di problema la posizione iniziale del robot è sconosciuta, rendendo quindi necessario capire dove si trovi in relazione all'ambiente circostante. A tal fine è necessaria una mappa, realizzata o a priori o durante la navigazione dal robot stesso, nella quale sono segnati dei punti di riferimento di cui può essere calcolata la distanza e, conseguentemente, la posizione relativa rispetto ad essi. Conoscendo quindi la posizione rispetto ai punti di riferimento, combinando i vari dati è possibile stimare la posizione del robot nella mappa;
- *Kidnapped Robot Problem*: è il problema più complicato da risolvere e si presenta quando il robot viene portato ("rapito") in un ambiente sconosciuto. Deve quindi capire di essere stato rapito ed essere in grado di recuperare da quella situazione, rilocalizzandosi in un ambiente sconosciuto e, per farlo, si può affidare esclusivamente ai dati sensoriali.

Un'altra importante distinzione riguarda il tipo di ambiente in cui si effettua la localizzazione, che può essere statico o dinamico. In ambiente statico la localizzazione risulta relativamenete semplice in quanto gli oggetti nella mappa rimangono fissi, riducendo così la complessità del calcolo della posizione rispetto a essi. In ambienti dinamici, al contrario, la presenza di oggetti in movimento può portare confusione e disorientamento al robot, per cui deve essere in grado di realizzare una mappa dinamica durante la navigazione.

1.2 Formulazione Matematica

Per localizzarsi correttamente il robot deve essere in grado sia di conoscere di quanto si è mosso dopo aver compiuto un movimento che di ricavare informazioni utili sull'ambiente circostante, come la posizione relativa rispetto ad un oggetto o ostacolo. A questo proposito si utilizza una combinazione di dati odometrici, per tenere traccia dei movimenti, e di dati provenienti da sensori, solitamente laser o a ultrasuoni, che permettono al robot di effettuare osservazioni dell'ambiente. Tuttavia, sia i dati odometrici che quelli provenienti da sensori spesso possono risultare rumorosi e portare ad un'incertezza sulla posizione percepita dal robot, rendendo così estremamente complesso determinarla correttamente in modo diretto. Potendo quindi solamente ipotizzare la posizione attuale del robot con una certa incertezza, essa viene rappresentata da una distribuzione di probabilità. Il processo di localizzazione si compone di due fasi:

1. Predizione: il robot utilizza i dati odometrici per stimare la sua posizione in seguito ad un movimento;
2. Correzione: la precedente stima viene corretta utilizzando le informazioni sull'ambiente circostante ricavate dai sensori.

In figura 1a viene proposto un esempio di localizzazione in uno spazio monodimensionale dove la posizione iniziale x_0, supponendo che sia nota e quindi priva di incertezza, risulta essere una distribuzione Delta di Dirac[1]. Successivamente, il robot si muove e, a causa dell'errore odometrico, si verifica un'incertezza riguardo la posizione x_1 che si accumulerà poi in x_2 data la propagazione di tale errore. Nella figura 1b viene mostrata la fase di correzione, nella quale il robot misura la distanza d dal muro, calcolando così la posizione $x_2^{'}$ che risulta però in conflitto con quella trovata durante la fase di predizione. La posizione x_2 viene quindi corretta combinandola con $x_2^{'}$ e ottenendo così l'effettiva stima della posizione attuale $x_2^{''}$ che, essendo più accurata delle precedenti, avrà un'incertezza minore.

Matematicamente, la posizione del robot può essere definita come

$$p = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{1}$$

con x e y coordinate nel piano cartesiano e θ orientazione. Considerando il robot un sistema a tempo discreto, la posizione attuale può essere stimata partendo da quella precedente e aggiungendo la distanza percorsa dall'istante $t-1$ all'istante t, che viene

[1] $\delta(t) = \begin{cases} 1 \text{ se } t = 0 \\ 0 \text{ se } t \neq 0 \end{cases}$

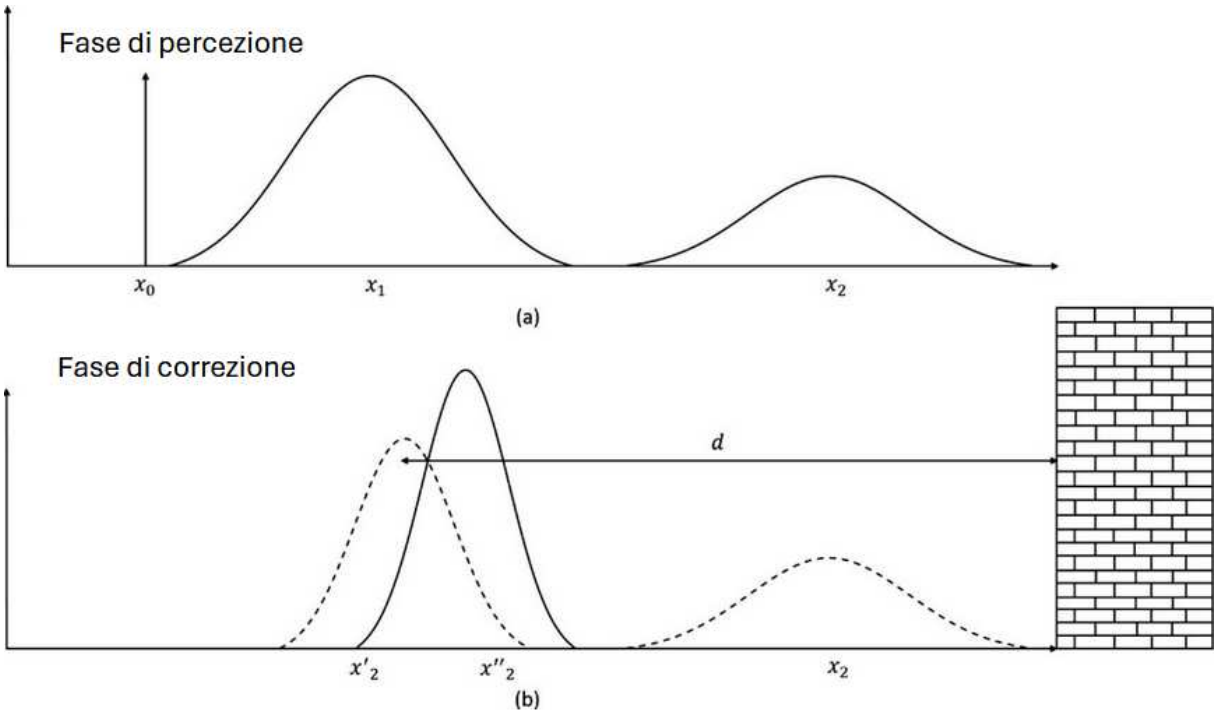

Figura 1: Localizzazione rispetto ad un ostacolo (immagine tratta da [1])

definita come

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} \Delta s \ cos\left(\theta + \frac{\theta}{2}\right) \\ \Delta s \ sin\left(\theta + \frac{\theta}{2}\right) \\ \frac{\Delta s_r + \Delta s_l}{b} \end{bmatrix} \tag{2}$$

con

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2} \tag{3}$$

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b} \tag{4}$$

dove Δs_r e Δs_l rappresentano rispettivamente la distanza percorsa con la ruota di destra e di sinistra mentre b è la distanza tra di esse.

La posizione aggiornata p' si può calcolare come

$$p' = p + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} \tag{5}$$

dove, sostituendo nelle equazioni (2) Δs con (3) e $\Delta\theta$ con 4, si ricava

$$p' = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_t}{2}(\cos\theta + \frac{\Delta s_r - \Delta s_t}{2b}) \\ \frac{\Delta s_r + \Delta s_t}{2}(\sin\theta + \frac{\Delta s_r - \Delta s_t}{2b}) \\ \frac{\Delta s_r - \Delta s_t}{b} \end{bmatrix} \tag{6}$$

L'incertezza sulla posizione incrementa a causa dell'errore nelle misurazioni di Δs_r e Δs_l.

2 Panoramica sulle strategie di localizzazione

In questa sezione verranno introdotte le principali strategie di localizzazione, suddividendole in categorie sulla base dell'approccio utilizzato.

2.1 Metodi Markov

Questa classe di approcci, appartenenti alla categoria dei metodi probabilistici, modellano il tragitto percorso dal robot come un processo markoviano, ovvero una sequenza di eventi casuali che gode della proprietà di Markov[2], che rappresentano la posizione come una distribuzione di probabilità e fanno uso di una rappresentazione discreta dello spazio, solitamente suddiviso in celle o griglie. La localizzazione avviene tramite un algoritmo ricorsivo che, ad ogni istante di tempo t, calcola la posizione in cui la probabilità di trovare il robot è più alta e, ad ogni ciclo, ricalcola tutte le posizioni passate. Queste caratteristiche permettono al robot di localizzarsi partendo da una posizione iniziale sconosciuta e, memorizzando la cronistoria degli stati precedenti, di recuperare da situazioni ambigue. Questo tipo di localizzazione è composto da due fasi:

- Fase di predizione: il robot stima il suo stato attuale ($\bar{S_{Best}}(x_t)$) a parire dallo stato precedente ($\bar{S_{Best}}(x_{t-1})$) e dalle informazioni propriocettive (ingresso di controllo) u_t.

$$\bar{S_{Best}}(x_t) = \sum_{x_{t-1}} p(x_t|u_t, x_{t-1}) S_{Best}(x_{t-1}) \tag{7}$$

(tempo discreto)

$$\bar{S_{Best}}(x_t) = \int p(x_t|u_t, x_{t-1}) S_{Best}(x_{t-1}) \mathrm{d}x_{t-1} \tag{8}$$

(tempo continuo)

- Aggiornamento delle misurazioni (o correzione): terminato il movimento, applicando la formula di Bayes[3] viene corretta la stima dello stato $\bar{S_{Best}}(x_t)$ ottenuta precedentemente combinandola con le misurazioni esterocettive dei sensori z_t nella nuova posizione, calcolando così il nuovo stato $S_{Best}(x_t)$:

$$S_{Best} = \eta p(z_t|x_t, M) \bar{S_{Best}}(x_t) \tag{9}$$

[2]dato un processo X_t, $\forall i$ t.c. $t_0 \leq t_i \leq t_n, x_0 \leq x_i \leq x_n$ vale:

$$P(X_{t_{n+1}} = x_{t_{n+1}} | X_{t_n} = x_{t_n}, ... X_0 = x_0) = P(X_{t_{n+1}} = x_{t_{n+1}} | X_{t_n} = x_{t_n})$$

[3]dati due eventi A e B vale:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

dove x_t rappresenta la posizione del robot, mentre $p(z_t|x_t, M)$, noto anche come modello probabilistico di misura, è la probabilità di osservare z_t note x_t e la mappa M ed è calcolato da una funzione di misurazione h, priva di rumore e dipendente da x_t e M.

Prendendo come riferimento la figura 1, supponendo l'ambiente unidimensionale e il muro come unico elemento della mappa, la funzione h è calcolata come:

$$h(x_t, M) = m - x_t \tag{10}$$

dove m sono le coordinate del muro rispetto alla posizione iniziale del robot. Per derivare il modello probabilistico di misura è necessario aggiungere alla funzione h un termine di rumore. Assumendo rumore Gaussiano, si otterrà:

$$p(z_t|x_t, M) = N(h(x_t, M), R_t) \tag{11}$$

dove N rappresenta una distribuzione gaussiana multivariata avente media $h(x_t, M)$ e matrice di covarianza R_t.

2.2 Metodi basati su Filtro di Kalman

2.2.1 Filtro di Kalman

Il Filtro di Kalman è un algoritmo che che viene utilizzato come osservatore ottimo dello stato di un sistema dinamico tramite una combinazione di stime e osservazioni rumorose. Rappresenta un caso particolare di localizzazione Markoviana dove, invece di utilizzare distribuzioni arbitrarie per rappresentare lo stato del robot e il rumore, vengono usate distribuzioni gaussiane, che hanno come PDF[4] per il caso scalare

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\Big(- \frac{(x-\mu)^2}{2\sigma^2}\Big) \tag{12}$$

mentre per il caso vettoriale

$$p(x) = \frac{1}{\sqrt[k]{2\pi}|\sqrt{\Sigma}|} \exp\Big(- \frac{(x-\mu)^T}{\Sigma^{-1}(x-\mu)}\Big) \tag{13}$$

dove:

- x = variabile scalare o vettoriale di dimensione k casuale;
- μ = media di x;

[4] *Probability Density Function*

- σ^2 = varianza[5] di x nel caso scalare;
- Σ = matrice di covarianza[6] nel caso vettoriale.

Un sistema può essere descritto da due equazioni, dette equazione di stato ed equazione di uscita, della forma:

$$\begin{cases} x_k = Ax_{k-1} + Bu_{k-1} + w_k \\ z_k = Hx_k + v_k \end{cases} \tag{14}$$

dove:

- x_k, x_{k-1} = vettori dello stato rispettivamente agli istanti k e $k-1$;
- z_k = vettore di uscita o di osservazione;
- A = matrice di transizione di stato;
- B = matrice di controllo;
- H = matrice di uscita o di osservazione;
- u_{k-1} = vettore di controllo;
- w_k = rumore di ingresso;
- v_k = rumore di uscita o osservazione;

w_k e v_k sono rumori Gaussiani a media nulla e incorrelati tra loro, ovvero la matrice di covarianza risulta:

$$\begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \tag{15}$$

Si definisce poi $\hat{x_k^-} \in R^n$ come la stima a priori dello stato del sistema all'istante $k-1$ calcolata dall'equazione di transizione di stato, mentre $\hat{x_k}$ è la stima dello stato a posteriori ottenuta combinando $\hat{x_k^-}$ e le osservazioni all'istante k. Si definiscono inoltre gli errori sulle stime

$$e_k^- = x_k - \hat{x}_k^- \tag{16}$$

$$e_k = x_k - \hat{x}_k \tag{17}$$

[5]data una variabile aleatoria X, la sua varianza è definita come:

$$\sigma_X^2 = E[(X - E[X])^2]$$

[6]date due variabili aleatorie X e Y, la covarianza è definita come $Cov(X,Y) = E[(X - E(X)(Y - E[Y])]$ e la matrice di covarianza è

$$\Sigma = \begin{bmatrix} Var(X) & Cov(X,Y) \\ Cov(Y,X) & Var(Y) \end{bmatrix}$$

e le relative covarianze

$$P_k^- = E[e_k^- e_k^{-T}] \tag{18}$$

$$P_k = E[e_k e_k^T] \tag{19}$$

L'algoritmo del Filtro di Kalman si compone di due fasi:

- Predizione: si stimano stato e covarianza dell'errore all'istante k

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{20}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{21}$$

- Correzione: le stime ottenute precedentemente vengono corrette sfruttando le misurazioni effettuate all'istante k

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{22}$$

$$P_k = (I - K_k H)P_k^- \tag{23}$$

  dove K_k rappresenta il Guadagno di Kalman, ovvero

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{24}$$

2.2.2 Filtro di Kalman Esteso

La principale limitazione del Filtro di Kalman riguarda la sua applicabilità esclusivamente a sistemi lineari quando, la maggior parte dei sistemi reali, risultano essere non- lineari. La soluzione a questo problema risiede nell'algoritmo denominato Filtro di Kalman Esteso (EKF dall'inglese *Extended Kalman Filter*), che linearizza le equazioni non-lineari del sistema tramite espansioni di Taylor del primo ordine per poi applicare al sistema linearizzato il tradizionale Filtro di Kalman. Dato un sistema descritto da

$$\begin{cases} x(k) = f(x(k-1),\ w(k-1)) \\ y(k) = h(x(k),\ v(k)) \end{cases} \tag{25}$$

dove $f(\cdot)$ e $k(\cdot)$ sono relazioni non-lineari, l'algoritmo si sviluppa seguendo le stesse fasi del tradizionale Filtro di Kalman con:

- Predizione:

$$x_k^- = f(\hat{x}_{k-1}) \tag{26}$$

$$P_k^- = AP_kA^T + Q \tag{27}$$

  con $A = \frac{\mathrm{d}f}{\mathrm{d}x}|_x$

- Correzione:

$$\hat{x_k} = \hat{x}_k = \hat{x}_k^- + K_k(y_k - h(\hat{x}_k^-)) \tag{28}$$

$$P_k = (I - K_k H)P_k^- \tag{29}$$

con H=$\frac{dh}{dx}|x$ e K_k Guadagno di Kalman

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{30}$$

2.2.3 *Unscented Kalman Filter*

L'*Unscented Kalman Filter* (UKF) è un altro tipo di Filtro di Kalman utilizzato per sistemi non lineari ma più preciso dell'EKF. Il suo funzionamento si basa sulla *Unscented Transform*, che prevede il campionamento di un insieme di punti, detti *sigma points*, scelti secondo una strategia di campionamento ben definita in modo tale da approssimare la distribuzione con il minor numero di punti possibile. La strategia di campionamento più comune consiste nella scelta di $2n+1$ *sigma points* (con n dimensione del vettore di stato), dove uno di essi coincide con la media x, mentre gli altri sono distribuiti simmetricamente attorno al punto medio in base alla matrice di covarianza P_x. I *sigma points* campionati vengono poi propagati attraverso la funzione non-lineare del sistema per ottenere un nuovo insieme di punti, con media y e covarianza P_y, che descrivono con maggiore accuratezza la distribuzione di probabilità.

Dato un sistema non-lineare descritto da (25), l'algoritmo dell'UKF procede attraverso le seguenti fasi:

1. inizializzazione del vettore di stato e della matrice di covarianza dell'errore sulla stima dello stato:
2. si campionano i *sigma points* e si calcolano i rispettivi valori pesati tramite le seguenti equazioni:

$$x_0 = \hat{x_k} \tag{31}$$

$$\begin{cases} x_i = \hat{x}_k + (\sqrt{(n+\lambda)P_k})_k^T & \text{per } i = 1, \dots, n \\ x_i = \hat{x}_k - (\sqrt{(n+\lambda)P_k})_k^T & \text{per } i = n+1, \dots, 2n \end{cases} \tag{32}$$

$$\begin{cases} W_0^m = \lambda/(\lambda+n) \\ W_i^m = 1/2(\lambda+n) & \text{per } i = 1, \dots, 2n \end{cases} \tag{33}$$

$$\begin{cases} W_0^c = W_0^m + (1 - \alpha^2 + \beta) \\ W_i^c = 1/2(\lambda+n) & \text{per } i = 1, \dots, 2n \end{cases} \tag{34}$$

$$\lambda = \alpha^2(k+n) - n \tag{35}$$

dove:

- α è un parametro che indica il grado di dispersione dei *sigma points*;
- k solitamente è posto uguale a 0;
- β descrive la distribuzione delle variabili di stato;
- W_i^m e W_i^c rappresentano, rispettivamente, il peso dei punti calcolato rispetto a media e covarianza.

3. predizione dello stato propagando i *sigma points*:

$$\epsilon_i = f(x_i) \tag{36}$$

$$\hat{x}_{k+1/k} = \sum_{i=0}^{2n} W_i^m \epsilon_i \tag{37}$$

$$P_{k+1/k} = \sum_{i=0}^{2n} W_i^c (\epsilon_i - \hat{x}_{k+1/k})(\epsilon_i - \hat{x}_{k+1/k})^T \tag{38}$$

4. predizione delle osservazioni:

$$Z_i = h(\epsilon_i) \tag{39}$$

$$\hat{z}_{k+1/k} = \sum_{i=0}^{2n} W_i^m Z_i \tag{40}$$

$$P_{zz} = \sum_{i=0}^{2n} W_i^c (Z_i - \hat{z}_{k+1/k})(Z_i - \hat{z}_{k+1/k})^T \tag{41}$$

5. correzione delle predizioni:

$$\hat{x}_{k+1} = \hat{x}_{k+1/k} + K_k(yk + 1 - \hat{z}_{k+1/k}) \tag{42}$$

$$P_{k+1/k+1} = P_{k+1/k} - K_{k+1} P_{zz} K_{k+1}^T \tag{43}$$

$$K_{k+1} = P_{xz} P_{zz}^{-1} \tag{44}$$

con

- K_{k+1} guadagno di Kalman;
- $P_{xz} = \sum_{i=0}^{2n} W_i^c (\epsilon_i - \hat{x}_{k+1/k})(\epsilon_i - \hat{z}_{k+1/k})^T$ matrice di cross-covarianza tra predizione dello stato e delle osservazioni.

Nonostante la computazione risulti simile a quella dell'*Extended Kalman FIlter*, l'utilizzo dell'*Unscented Kalman Filter* comporta numerosi vantaggi dati principalmente dalla sua natura. Infatti l'UKF, approssimando la distribuzione di probabilità della funzione

non-lineare invece della funzione stessa non procede per linearizzazione, evitando così il calcolo di Jacobiane che talvolta possono risultare complicate e portare ad un errore maggiore. Inoltre, avvicinandosi maggiormente al valore reale dello stato rispetto all'EKF, le soluzioni ottenute saranno più accurate e la convergenza ad esse sarà più rapida. Lo svantaggio più rilevante dell'UKF, invece, risiede nella sua possibile elevata complessità computazionale, problema che, uttavia, può essere contenuto scegliendo accuratamente la startegia di campionamento dei *sigma points*.

2.2.4 Localizzazione tramite KF

Durante la predizione il robot stima la posizione a cui si troverà in seguito all'applicazione di un ingresso di controllo. La percezione, invece, si risolve seguendo quattro fasi:

- Osservazione: il robot utilizza i sensori di osservazione per ricavare informazioni riguardo l'ambiente circostante;
- Predizione delle misurazioni: il robot stima i dati provenienti dai sensori di osservazione che misurerà una volta andato nella posizione predetta durante la fase di predizione;
- *Matching*: il robot calcola il miglior *match* tra le informazioni ottenute durante l'osservazione e quelle stimate durante la predizione delle misurazioni;
- Stima: lo stato successivo del robot viene stimato dal Filtro di Kalman fondendo le informazioni che hanno prodotto il miglior *match*.

Quindi, assumendo che

$$S_{t-1} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} \tag{45}$$

sia la miglior stima della posizione all'istante $t-1$, applicando le equazioni (6), è possibile stimare

$$S_t = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_t}{2}(\cos\theta_{t-1} + \frac{\Delta s_r - \Delta s_t}{2b}) \\ \frac{\Delta s_r + \Delta s_t}{2}(\sin\theta_{t-1} + \frac{\Delta s_r - \Delta s_t}{2b}) \\ \frac{\Delta s_r - \Delta s_t}{b} \end{bmatrix} \tag{46}$$

dove S_t rappresenta la stima della posizione all'istante t.

Come già accennato precedentemente, il maggiore problema del Filtro di Kalman è quello di essere applicabile solo a sistemi lineari con rumori gaussiani ma, la maggior parte dei robot, risultano essere sistemi non lineari quindi, per ovviare a questo problema, vengono utilizzate strategie basate sull'*Extended Kalman FIlter* e sull'*Unscented Kalman Filter*. Rispetto alle tecniche di localizzazione markoviane, quelle basate su Filtro di Kalman risultano più efficienti ma richiedono di conoscere la posizione iniziale del robot con una

certa precisione, impedendo così di recuperare in caso di smarrimento. Il Filtro di Kalman, quindi, risolve efficientemente il problema del *Position Tracking* ma non quelli della *Global Localization* e del *Kidnapped Robot*.

2.3 Metodi evolutivi

I metodi evolutivi sono basati su una classe di algoritmi stocastici, detti algoritmi evolutivi, utilizzati per risolvere problemi di ricerca e ottimizzazione e sono ispirati alle teorie evoluzionistiche di Charles Darwin, ovvero sono ispirati a meccanismi evolutivi biologici quali selezione, mutazione, riproduzione e ricombinazione. Gli algoritmi evolutivi si sviluppano nelle seguenti fasi:

- si genera casualmente una popolazione originaria dove il genoma di ogni elemento rappresenta una possibile soluzione;
- si decide una funzione di fitness [7] da applicare al dominio iniziale per valutare la qualità di ogni soluzione, i candidati con la fitness maggiore saranno i genitori della prossima generazione (selezione);
- si genera la nuova generazione di candidati (riproduzione) applicando ai genitori trovati al punto precedente ricombinazione e mutazione:
  - Ricombinazione: operatore binario che, applicato a due o più genitori, genera un figlio combinando i loro genotipi;
  - Mutazione: operatore unitario che, applicata ad un genitore, genera il figlio del suo genotipo e serve per garantire la diversità della popolazione;
- si controlla la condizione di terminazione: se è verificata l'algoritmo termina, altrimenti si prosegue all'iterazione successiva utilizzando la nuova generazione come popolazione iniziale. Solitamente la condizione di terminazione è il raggiungimento di un livello di fitness ottimo ma, essendo gli algoritmi evolutivi stocastici, non è garantito tale raggiungimento, per cui sono necessarie altre condizioni che terminino sicuramente l'algoritmo, come ad esempio un numero massimo di iterazioni.

Gli algoritmi evolutivi si suddividono in quattro sottocategorie:

- Algoritmi Genetici: rappresentano i geni come stringhe di bit, sono i tipi di algoritmi evolutivi più comuni in quanto implementano nel computer il meccanismo di evoluzione nel modo più semplice;
- Programmazione Genetica: funzionamento simile a quello degli algoritmi genetici ma, in questo caso, i geni rappresentano un intero programma o algoritmo

[7] funzione che valuta la qualità delle possibili soluzioni di un problema assegnando ad ognuna di esse un valore numerico

- Strategie Evolutive: utilizzate per l'ottimizzazione di funzioni complesse e non differenziabili, rappresentano i geni come vettori di valori reali e cercano di trovare la loro espressione ottima eliminando codifiche ridondanti e superflue;
- Programmazione Evolutiva: simili alle strategie evolutive con l'unica differenza che non ci sono restrizioni riguardo la rappresentazione dei geni che, solitamente, è un programma con struttura fissa e di cui si cerca di ottimizzare i parametri coinvolti. Differiscono dalla Programmazione Genetica in quanto, in quest'ultima, i programmi possono cambiare struttura nel tempo.

La scelta del tipo di algoritmo evolutivo da utilizzare dipende principalmente dal modo migliore per rappresentare le soluzioni.

I vantaggi più rilevanti dell'utlizzare gli algoritmi evolutivi sono i seguenti:

- ispirandosi all'evoluzione biologica risultano generalmente semplici da implementare;
- utilizzano informazioni a priori che è più accessibile rispetto a quella a posteriori;
- non richiedono caratteristiche particolari della funzione come continuità e differenziabilità;

Lo svantaggio principale, invece, riguarda il fatto che gli algoritmi evolutivi possono portare alla soluzione ottimale in un tempo imprevedibile, può essere necessario un *tuning* dei parametri a tentativi e puossono richiedere elevati costi computazionali.

2.4 Costruzione autonoma di mappe

Nei metodi descritti precedentemente è richiesta la conoscenza dell'ambiente circostante e ciò è garantito da una mappa, solitamente preparata in precedenza, in cui sono presenti punti di riferimento che il robot utilizza per localizzarsi. Questo, tuttavia, è realizzabile efficientemente in ambienti statici in cui i punti di riferimento segnati sulla mappa rimangono fissi. In ambienti dinamici, in cui la configurazione varia a causa di elementi mobili, è esclusa la possibilità di realizzare una mappa utile alla localizzazione del robot. La soluzione a questo problema è di far costruire autonomamente la mappa al robot: cominciando a muoversi da un punto iniziale, ricava informazioni riguardo l'ambiente circostante grazie a sensori e, interpretandole, costruisce la mappa. Il problema di costruire e aggiornare la mappa e, contemporaneamente, localizzarsi, è definito *SLAM problem* (*Simultaneous Localization and Autonomous Mapping*) e generalmente viene formulato in maniera probabilistica. In un generico *SLAM problem* vengono definiti i seguenti insiemi:

- $X_{0:k} = [x_0, x_1, ..., x_k]$: insieme degli stati (posizioni) del robot fino all'istante k;
- $U_{0:k} = [u_0, u_1, ..., u_k]$: insieme degli ingressi di controllo, dove u_i è il controllo applicato per passare da x_{i-1} a x_i;

- $m = [m_1, m_2, ..., mn]$: insieme degli n punti di riferimento;
- $Z_{0:k} = [z_0, z_1, ..., z_k]$: insieme delle osservazioni;

e, ad ogni istante t, viene richiesto di calcolare

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0) \tag{47}$$

che rappresenta la probabilità congiunta a posteriori dello stato del robot x_t e della posizione dei punti di riferimento date le osservazioni e gli ingressi di controllo precedenti e la posizione iniziale. La soluzione migliore a questo problema è ricorsiva e sfrutta l'Inferenza Bayesiana[8], in cui la probabilità a posteriori viene calcolata, applicando il teorema di Bayes, partendo dalla stima della distribuzione $P(x_{k-1}, m|Z_{0:k-1}, U_{0,k-1})$ e utilizzando le osservazioni z_k e l'ingresso di controllo applicato u_k. Per la corretta riuscita del calcolo è necessario definire:

- Modello di osservazione: descrive la probabilità di effettuare un'osservazione z_k conoscendo lo stato e la posizione dei punti di riferimento.
- Modello di movimento: descrive la probabilità la transizione di stato che viene assunta essere un processo di Markov.

L'algoritmo di *SLAM* si compone di due fasi:

- Predizione(*Time Update*):

$$P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0) = \int P(x_k|x_{k-1}, u_k) \times P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1}, x_0)\mathrm{d}x \tag{48}$$

- Correzione(*Measurement Update*):

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0) = \frac{P(z_k|x_k, m)P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0)}{P(z_k|Z_{0:k-1}, U_{0:k})} \tag{49}$$

che, ricorsivamente, calcolano la probabilità congiunta a posteriori $P(x_k, m|Z_{0:k}, U_{0:k}, x_0)$ all'istante k.

Le soluzioni ai problemi di *SLAM* richiedono di trovare dei metodi per rappresentare i modelli di movimento e osservazione per rendere più efficiente e consistente il calcolo di (48) e (49). La rappresentazione più comune aggiunge ai modelli un rumore Gaussiano per utilizzare un Filtro di Kalman Esteso portando così all'approccio *EKF-SLAM*. Un'altra tecnica molto diffusa, detta *FastSLAM*, combina il Filtro di Kalman Esteso con Filtri Particellari e, rispetto all'*EKF-SLAM*, non richiede necessariamente distribuzioni Gaussiane.

[8]approccio statitstico che, tramite il teorema di Bayes, utilizza le osservazioni per aggiornare le probabiltà degli eventi

2.5 Metodi RFID

I metodi RFID (*R*adiofrequency Identification) sfruttano segnali a radiofrequenza per localizzare e identificare oggetti in ambienti indoor. Il loro funzionamento si basa su tre componenti principali, ovvero:

- *RFID Tags*: sono piccoli dispositivi che possono essere:
  1. Attivi: alimentati a batteria e in grado di inviare segnali attivamente;
  2. Passivi: alimentati dalle onde elettromagnetiche emesse dal lettore;
  3. Semi-passivi: alimentati a batteria ma utilizzano l'energia del lettore per la trasmissione;
- *RFID Reader*: dispositivo in grado di emettere onde radio a frequenza definita e di ricevere dati dai tags seguendo un protocollo;
- *Data Processing Subsystem*: sottositema che utilizza i dati ricevuti dai tags per eseguire gli algoritmi di localizzazione.

I segnali emessi da ogni tag contengono il proprio identificativo univoco e il lettore, nel momento in cui li riceve, è in grado anche di stimare la posizione relativa rispetto al tag misurando una delle grandezze del segnale a cui la distanza è proporzionale, tra cui *TOA* [9], *POA* [10] o, più comunemente, l'*RSS* [11]. Se più tags sono contemporaneamente presenti all'interno del raggio d'azione di un reader, la posizione dell'obiettivo viene stimata combinando tutti gli RSS (o un'altra grandezza precedentemente citata) ricevuti. Il problema principale di questa strategia è che è molto sensibile a disturbi ambientali come interferenze o riflessioni indesiderate dei segnali.

Per valutare la qualità dei metodi RFID si tengono in considerazione alcuni parametri, tra cui i più importanti sono:

- Precisione: misura quanto la posizione reale dista da quella stimata;
- Latenza: si riferisce al tempo che trascorre tra il cambio di posizione dell'obiettivo e il ricevimento delle relative informazioni;
- Costo: valuta il costo dei tags e dei lettori.

I metodi RFID possono essere suddivisi in categorie in base al dispositivo connesso all'oggetto da localizzare. Esistono tecnologie *Tag-based* che consistono nell'attaccare all'obiettivo un tag attivo che periodicamente invia segnali ai reader posizionati nell'ambiente e tecnologie *Reader-based*, che prevedono l'uso di un lettore portatile trasportato dall'obiettivo. Oltre ai metodi tradizionali appena descritti, esistono metodi *Transceiver-*

[9] *Time of Arrival*
[10] *Phase of Arrival*
[11] *Received Signal Strenght*

free Technologies che non richiedono all'obiettivo di trasportare alcun dispositivo e si basano sull'idea secondo la quale, in ambienti statici, i segnali wireless sono relativamente stabili ed è probabile che si verifichi un'interferenza quando un oggetto si muove vicino ai dispositivi RFID, permettendo così di localizzarlo. L'ultima categoria è quella delle *Hybrid Technologies* che combinano le classiche startegie RFID con altre tecnologie come Wi-Fi, Bluetooth o visione artificiale. I vantaggi dei metodi RFID sono la semplicità implementativa e il basso costo dei componenti, mentre le principali sfide riguardano la localizzazione di più oggetti contemporaneamente e come risolvere il problema dell'interferenza e, tra le categorie descritte prima, le *Hybrid Technologies* vengono considerate la soluzione migliore.

3 Approfondimento: metodo basato su stimatore PSO

3.1 Introduzione al PSO

Il *Particle Swarm Optimization* (Ottimizzazione a Sciame di Particelle) è un algortimo di ottimizzazione evolutivo ispirato dal comportamento collettivo di una comunità animale, come per esempio un banco di pesci o uno stormo di uccelli dove, secondo i sociobiologi, ogni membro può giovare dall'esperienza di tutti gli altri. Questo algoritmo simula il comportamendo di un numero arbitrario di agenti indipendenti, chiamati particelle, che sono descritti da dei parametri, ovvero:

- $x_i(t)$ = posizione dell'i-esima particella all'iterazione t;
- $v_i(t)$ = velocità dell'i-esima particella all'iterazione t;
- p_i = soluzione migliore trovata finora dall'i-esima particella.

Ad ogni iterazione i parametri di ogni particella vengono aggiornate secondo le seguenti equazioni:

$$v_i(t+1) = w \times v_i(t) + c_1 \times r_1 \times (p_{besti} - x_i(t)) + c_2 \times r_2 \times (g_{best} - x_i(t)) \tag{50}$$

$$x_i(t+1) = x_i(t) + v_i(t) \tag{51}$$

$$p_i = x_i | f(x_i) = \max_{k=1,2,\ldots,t} f(x_i(k) \tag{52}$$

dove:

- f è la funzione da ottimizzare;
- g_{best} è la soluzione globale migliore trovata finora;
- w è chiamato parametro di inerzia e determina quanto la velocità all'istante t+1 è influenzata da quella precedente;
- c_1 e c_2 sono costanti positive dette coefficienti di accelerazione;
- r_1 e r_2 sono vettori casuali distribuiti in$[0, 1)^D$ (dove D è la dimensione dello spazio di ricerca);

Una volta calcolati tutti i parametri di ogni particella si verifica che non esista una soluzione g'_{best} tale che

$$g_{best}^{'} = x|f(x) = \max_{i=1,2,\ldots,N/k=1,2,\ldots,t+1} f(x_i(t+1)) \tag{53}$$

e, se esiste, si pone $g_{best} = g_{best}^{'}$. A questo punto, se la condizione di arresto predefinta (e.g. la g_{best} non migliora ulteriormente) non è verificata, si passa all'iterazione successiva, altrimenti l'algoritmo termina e la soluzione migliore trovata sarà quella ottima.

Input:

N = numero di particelle

D = dimensione dello spazio di ricerca

f = funzione di fitness

c_1, c_2, w: parametri dell'algoritmo

Output

g_{best} = soluzione migliore

Pseudocodice

Initialize empty vectors X[N], V[N]

for each $i \in [1, ..., N]$ **do**

Randomly generate x_i, v_i

$p_i \leftarrow x_i$

end for

$g_{best} \leftarrow p_1$

for each $i \in [1, ..., N]$ **do**

if $f(p_i) > f(g_{best})$ **then**

$f(g_{best}) \leftarrow f(p_i)$

end if

end for

while termination condition **do**

for $i \in [1, ..., N]$ **do**

compute $v_i(t+1)$ using equation (50)

compute $x_i(t+1)$ using equation (51)

if $f(x_i(t+1)) > f(p_i)$ **then**

$p_i \leftarrow x_i(t+1)$

end if

if $f(p_i) > f(g_{best})$ **then**

$f(g_{best}) \leftarrow f(p_i)$

end if

end for

$t \leftarrow t + 1$

end while

return g_{best}

3.2 Localizzazione con stimatore PSO

Applicando il criterio MAP[12] la localizzazione tramite *Particle Swarm Optimization* si riduce ad un problema di ottimizzazione, in cui la soluzione è la posizione del robot che massimizza la distribuzione della probabilità a posteriori, ovvero:

$$\arg\max_x p(x^t|z^t) \tag{54}$$

dove $x^t = [x(0), x(1), ..., x(t)]$ rappresenta le posizioni del robot fino all'istante t, mentre $z^t = [z(1), z(2), ..., z(t)]$ le osservazioni effettuate. Applicando la formula di Bayes si ottiene:

$$p(x^t|z^t) = \eta p(z_t|x^t, z^{t-1}) p(x_t|x^{t-1}, z^{t-1}) p(x^{t-1}|z^{t-1}) \tag{55}$$

dove η è una costante di normalizzazione. Applicando questo criterio ad ogni iterazione dell'algoritmo si ottiene la funzione di verosimiglianza [13] che all'istante t risulta:

$$p(x^t|z^t) = \eta \prod_{i=1}^{t} p(z_i|x^i, z^{i-1}) \prod_{i=1}^{t} p(x_i|x^{i-1}, z^{i-1}) p(x_0) \tag{56}$$

con $p(x_0) = p(x^{t-1}|z^{t-1})$. Ora, sostituendo l'equazione appena trovata in (54) si ottiene la funzione di costo

$$\arg\max_x \prod_{i=1}^{t} p(z_i|x^i, z^{i-1}) \prod_{i=1}^{t} p(x_i|x^{i-1}, z^{i-1}) p(x_0) \tag{57}$$

dove η è stata rimossa in quanto, essendo costante, non influisce sulla massimizzazione della funzione.

La funzione (57) può essere riscritta in forma logaritmica sostituendo le produttorie con delle sommatorie:

$$\arg\max_x \log p(x^t|z^t) = \arg\max_x \sum_{i=1}^{t} \log p(z_i|x^i, z^{i-1}) + \sum_{i=1}^{t} \log p(x_i|x^{i-1}, z^{i-1}) + logp(x_0) \tag{58}$$

Notando che le osservazioni z_{t-1} all'iterazione t non influiscono nè sullo stato nè sulle osservazioni e chiamando la funzione obiettivo $f_0(x_t)$, l'equazione precedente si può riscrivere nel seguente modo:

$$f_0(x_t) = \sum_{i=1}^{t} \log p(z_i|x_i) + \sum_{i=1}^{t} \log p(x_i|x_{i-1}) + \log p(x_0) \tag{59}$$

[12] Maximum a Posteriori: $\hat{\theta}_{MAP} = \arg\max_{\theta} P(\theta|X)$, con $P(\theta|X)$ probabilità a posteriori

[13] $L(\theta|X) = P(X|\theta)$

dove, estraendo gli ultimi termini delle sommatorie, si ottiene:

$$f_0(x_t) = \log p(z_t|x_t) + \log p(x_t|x_{t-1}) + f_0(x_{t-1}) \tag{60}$$

con

$$f_0(x_{t-1}) = \sum_{i=1}^{t-1} \log p(z_i|x_i) + \sum_{i=1}^{t-1} \log p(x_i|x_{i-1}) + \log p(x_0) \tag{61}$$

che risulta essere una funzione ricorsiva in cui sono stati evidenziati due termini, ovvero la densità di probabilità di transizione di stato $p_{\omega_t}(x_i|x_{i-1})$, legata alla posizione, e la densità di probabilità di di osservazione $p_{\nu_t}(z_i|x_t)$, legata all'osservazione. La componente ricorsiva $f_0(x_{t-1})$ all'iterazione t non ha contributo nell'ottimizzazione in quanto riguardante esclusivamente iterazioni precedenti, per cui il problema iniziale può essere riformulato nel seguente modo:

$$\mathrm{Min}_x(\log p_{\nu_t}(z_i|x_t) + \log p_{\omega_t}(x_i|x_{i-1})) \tag{62}$$

che riduce la localizzazione ad un comune problema di ottimizzazione che può essere risolto tramite il *Particle Swarm Optimization* descritto nella sezione precedente. Considerando come funzione di fitness

$$Fitness = \log p_{\nu_t}(z_i|x_t) + \log p_{\omega_t}(x_i|x_{i-1}) \tag{63}$$

e notando che le densità di probabilità sia di transizione che di osservazione sono Gaussiane multivariate e quindi

$$p_{\nu_t}(z_i|x_t) \propto \exp\left(\frac{1}{2}(z_t - \hat{z}_t)R_t^{-1}(z_t - \hat{z}_t)^T\right) \tag{64}$$

$$p_{\omega_t}(x_i|x_{t-1}) \propto \exp\left(\frac{1}{2}(x_t - \hat{x}_t)Q_t^{-1}(x_t - \hat{x}_t)^T\right) \tag{65}$$

dove $\hat{z}_t$ e $\hat{x}_t$ sono i vettori contenenti rispettivamente le stime delle misurazioni e dello stato mentre R e Q sono le matrici di covarianza, sostituendo (64) e (65) in (63) la funzione di fitness può essere riscritta nel seguente modo:

$$Fitness = (z_t - \hat{z}_t)R_t^{-1}(z_t - \hat{z}_t)^T + (x_t - \hat{x}_t)Q_t^{-1}(x_t - \hat{x}_t)^T \tag{66}$$

che risulta essere una funzione composta da una somma di due termini quadratici e, essendo R e Q matrici definite positive, strettamente convessa. Per queste proprietà, ogni soluzione ottima locale sarà anche soluzione ottima globale e, per trovarla, si applica il *Particle Swarm Optimization*. Quindi, dopo l'inizializzazione delle N_p particelle (che rappresentano le possibili soluzioni), ciascuna di esse si muoverà nello spazio di ricerca cercando i punti con valore di fitness ottimale e, una volta raggiunta una certa soglia,

l'algoritmo terminerà. Tuttavia, i movimenti delle particelle nel *Particle Swarm Optimization* standard sono regolati da vettori casuali e senza considerare informazioni locali della funzione come il gradiente. Di conseguenza le particelle, anche trovandosi nei pressi della soluzione ottimale, potrebbero oltrepassarla a causa della regolazione casuale della velocità. Introducendo il gradiente, le capacità di ricerca locale dell'algoritmo incrementano in quanto vengono fornite alle particelle informazioni riguardanti la direzione di una soluzione e saranno anche in grado di capire il punto esatto in cui essa si trova ma, allo stesso tempo, può portare alla convergenza a soluzioni sub-ottimali e senza esplorare ulteriormente lo spazio di ricerca, riducendo così la capacità di ricerca globale e la diversificazione delle particelle. Per risolvere questo problema, il *Particle Swarm Optimization* standard viene combinato con il metodo delle perturbazioni simultanee (SPSA), ovvero una tecnica di ottimizzazione stocastica che, perturbando simultaneamente tutti i parametri della funzione, fornisce una accurata approssimazione del gradiente, migliorando così le capacità di ricerca locali dell'algoritmo e, contemporaneamente, mantenendo la diversità delle particelle. La combinazione dei due metodi aggiorna i parametri delle particelle con le seguenti equazioni:

$$x_t^i = x_{t-1}^i + v_t^i \tag{67}$$

$$v_t^i = \begin{cases} v_{max} & \text{se } v_t^i > v_{max} \\ w(-\gamma \nabla \psi(x_{t-1}^i) + c_1 r_1 (p^i - x_{t-1}^i) + c_2 r_2 (g_{best} - x_{t-1}) & \text{se } v_{min} < v_t^i < v_{max} \\ v_{min} & \text{se } v_t^i < v_{min} \end{cases} \tag{68}$$

con, in aggiunta a (50):

- γ costante positiva;
- c_1 e c_2 impostati uguali a 2;
- $\nabla\psi$ è il gradiente della funzione di fitness.

Rispetto al *Particle Swarm Optimization* standard è stato inoltre introdotta una limitazione alla velocità massima raggiungibile dalle particelle: il motivo risiede nel fatto che, se le particelle si muovessero troppo velocemente, ci potrebbero essere una minore capacità di esplorazione dello spazio di ricerca e una minore efficienza dell'algoritmo. In questo caso, v_{max} è limitata al 10% del range dinamico di ogni variabile.

3.3 Localizzazione con *Particle Filter*

Il *Particle Filter* è un algoritmo di stima probabilistico basato su simulazioni Monte Carlo sequenziali. Similmente al *Particle Swarm Optimization*, utilizza un insieme di particelle che rappresentano una possibile soluzione al problema e, insieme, approssimano una distribuzione di probabilità.

L'algoritmo si compone delle seguenti fasi:

1. Inizializzazione: in questa fase vengono inizializzate le N particelle che caratterizzano la probabilità a posteriori $p(x_{0:k}|z_{1:k})$. Le particelle sono della forma $\{x^i_{0:k}, w^i_{0:k}\}^N_1$ dove

   - $\{x^i_{0:k}, i = 0, ..., N\}$ è l'insieme degli stati fino all'istante k;
   - $\{w^i_k, i = 0, ..., N\}$ sono i pesi associati ad ogni particella;

2. Stima: La densità a posteriori all'istante k è approssimata da

$$p(x_{0:k}|z_{1:k}) \approx \sum_{i=1}^{N_s} w^i_k \delta(x_{0:k} - x^i_{0:k}) \tag{69}$$

   dove δ è la funzione Delta di Dirac. In generale, tuttavia, non è possibile estrarre campioni dalla densità a posteriori, per cui diventa necessario introdurre un'altra distribuzione più semplice $q(\cdot)$, detta *Proposal Distribution*, su cui effettuare il campionamento. Siano $x^i \sim q(x), i = 0, ..., N$ i campioni estratti dalla *Proposal Distribution*, allora la densità a posteriori si approssima con

$$p(x) \approx \sum_{i=1}^{N_s} w^i \delta(x - x^i) \tag{70}$$

   dove w^i è il peso normalizzato dell'i-esima particella;

3. Aggiornamento dei pesi: il peso w^i_k in (69) viene aggiornato con

$$w^i_k = w^i_{k-1} \frac{p(\mathbf{z}_k|\mathbf{x}^i_k)p(\mathbf{x}^i_k|\mathbf{x}^i_{k-1})}{q(\mathbf{x}^i_k|\mathbf{x}^i_{0:k-1}, \mathbf{z}_{1:k})} \tag{71}$$

4. *Resampling*: in questa fase vengono eliminate le particelle a peso minore che rappresentano le meno probabili, mentre vengono tenute e replicate quelle a peso maggiore. Questo processo serve per concentrare, sempre di più ad ogni iterazione, le particelle nelle aree a probabilità maggiore.

Il principale problema del *Particle Filter* riguarda la degenerazione delle particelle nel tempo: questo accade soprattutto in casi in cui la funzione di verosimiglianza è concentrata in un area molto ristretta rispetto alla distribuzione proposta, portando così solo una piccola parte delle particelle ad avere alta probabilità. Questo comporta che la maggior parte delle particelle avrà un peso quasi nullo e ad essere eliminate nella fase di *Resampling* e, contemporaneamente, verranno replicate quelle poche particelle a peso maggiore che effettivamente descrivono la funzione di verosimiglianza. La conseguenza di ciò è una perdita di diversità nelle particelle che porta ad una rappresentazione della densità di probabilità inaccurata e ad una ridotta capacità di esplorazione dello spazio di stato.

La localizzazione tramite *Particle Filter* prevede di modellare la posizione del robot nel tempo x_t come un processo di Markov caratterizzato da una distribuzione iniziale p_{x_t} e un modello di movimento $p(x_t|x_{t-1}, u)$, mentre le osservazioni z_t vengono considerate indipendenti dai il processo e la distribuzione marginale $p_{\theta_t}(z_t|x_t)$, con θ_t vettore statico contenente le informazioni della mappa. Gli ingressi di controllo sono della forma $u_t = (v_t, \gamma_t)$ con v_t velocità e γ_t orientazione entrambe all'istante t. I modelli di transizione di stato e di osservazione sono, rispettivamente,

$$x_t = f(x_{t-1}, u_t) + \omega_t \tag{72}$$

$$z_t = h(x_t, \theta_t) + \nu_t \tag{73}$$

con:

- f funzione non-lineare;
- ω_t rumore di processo Gaussiano con matrice di covarianza Q_t;
- ν_t rumore di misura Gaussiano con matrice di covarianza Q_t;

Ora, il problema della localizzazione può essere formulato in maniera Bayesiana calcolando

$$p(x_{0:t}|z_{1:t}) \tag{74}$$

che rappresenta la distribuzione di probabilità a posteriori da approssimare con il *Particle Filter*. L'insieme delle particelle è tale che

$$S_t = \{(x^i_{0:t}, w^i_t)|i = 1, ..., N\} \tag{75}$$

dove

- $x_{0:t}$ è l'insieme di tutti gli stati fino all'istante t;
- $z_{1:t}$ sono le osservazioni effettuate fino all'istante t;
- w^i_t è il peso associato ad ogni particella.

La distribuzione di probabilità a posteriori (74) può ora essere approssimata come

$$p(x_{0:t}|z_{1:t}) \approx \sum_{i=1}^{N} w^i_t \delta(x_{0:t} - x^i_{0:t}) \tag{76}$$

dove

- $\delta(X)$ è il Delta di Dirac;

- i pesi w_t^i sono normalizzati in modo tale che

$$w_t^i > 0, \sum_0^N w_t^i = 1 \tag{77}$$

Ora vengono estratti i campioni da una *Proposal Distribution* $q(x_{0:t}|z_{1:t})$ e viene applicata una tecnica detta *Importance Sampling* per assegnare i pesi: ai campioni estratti da una regione dove la distribuzione a posteriori è più estesa rispetto a quella proposta verranno assegnati pesi maggiori, mentre in caso contrario avranno un peso minore. Il peso di ogni particella viene calcolato usando l'equazione (71):

$$w_t^i = w_{t-1}^i \frac{p(z_t|x_t^i)p(x_t^i|z_{1:t-1})}{q(x_t^i|x_{0:t-1}^i, z_{1:t})} \tag{78}$$

3.4 Risultati

In questa sezione verranno esposti i risultati ottenuti dalle simulazioni e dagli esperimenti effettuati dall'autore dell'articolo *Mobile robot localization based on PSO estimator* (Ramazan Havangi, 10 febbraio 2019). In particolare, i risultati ricavati dall'utilizzo del metodo proposto basato su PSO verrano confrontati con quelli ottenuti tramite PF e EKF.

3.4.1 Simulazioni

Riprendendo quanto esposto nella sezione 1, lo stato del robot viene modellato come

$$\begin{bmatrix} x & y & \theta \end{bmatrix}^T$$

con [x y] le coordinate sul piano cartesiano e $[\theta]$ l'orientazione. Le equazioni cinematiche del robot diventano:

$$\begin{cases} x_t = x_{t-1} + (V_t + \nu_V)\cos(\phi_{t-1} + \gamma_{t-1} + \nu_\gamma)\Delta t \\ y_t = y_{t-1} + (V_t + \nu_V)\sin(\phi_{t-1} + \gamma_{t-1} + \nu_\gamma)\Delta t \\ \phi_t = \phi_{t-1} + \frac{(V_t+\nu_V)}{B}\sin(\gamma_{t-1} + \nu_\nu)\Delta t \end{cases} \tag{79}$$

con:

- B: distanza tra la ruota sinistra e destra del veicolo;
- $u = \begin{bmatrix} V & \gamma \end{bmatrix}^T$: vettore di ingresso che comprende un controllo di velocità V e uno di sterzata γ;
- $\nu = \begin{bmatrix} \nu_V & \nu_\gamma \end{bmatrix}^T$: rumore di processo dell'ingresso di controllo applicato.

Rispetto alle equazioni (6) scritte in funzione dei dati odometrici, queste sono scritte in funzione dell'ingresso di controllo e viene considerato un rumore di processo.

Per quanto riguarda le osservazioni, il veicolo si assume dotato di sensori che permettono al robot di misurare la sua posizione rispetto ad un punto di riferimento:

$$z = \begin{bmatrix} r_i \\ \theta_i \end{bmatrix} = \begin{bmatrix} \sqrt{(x - x_i)^2 + (y - y_i)^2} + \omega_r \\ \tan^{-1}(\frac{y-y_i}{x-x_i}) - \phi + \omega_\phi \end{bmatrix} \tag{80}$$

dove:

- $\begin{bmatrix} x_i & y_i \end{bmatrix}$: posizione dell'i-esimo punto di riferimento;
- $W = \begin{bmatrix} \omega_r & \omega_\theta \end{bmatrix}^T$: rumore di osservazione.

La simulazione è stata effettuata assumendo:

- posizione iniziale del robot: $x_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}$;
- velocità massima del robot di 3 m/s, angolo massimo di sterzata di 30°;
- distanza tra le ruote di 4 m e veicolo dotato di sensore che misura, al massimo, una distanza di 20 m e un'orientazione di 180°;
- rumore di processo $\nu = \begin{bmatrix} 0,3m/s & 3° \end{bmatrix}^T$ e rumore di osservazione $W = \begin{bmatrix} 0,2m & 1° \end{bmatrix}^T$;
- frequenza di controllo di 40 Hz e osservazione effettuate con una frequenza di 5 Hz.

La figura 2 mostra la mappa dell'ambiente (posizione dei punti di riferimento) in cui è stata realizzata la simulazione e la traiettoria seguita dal veicolo.

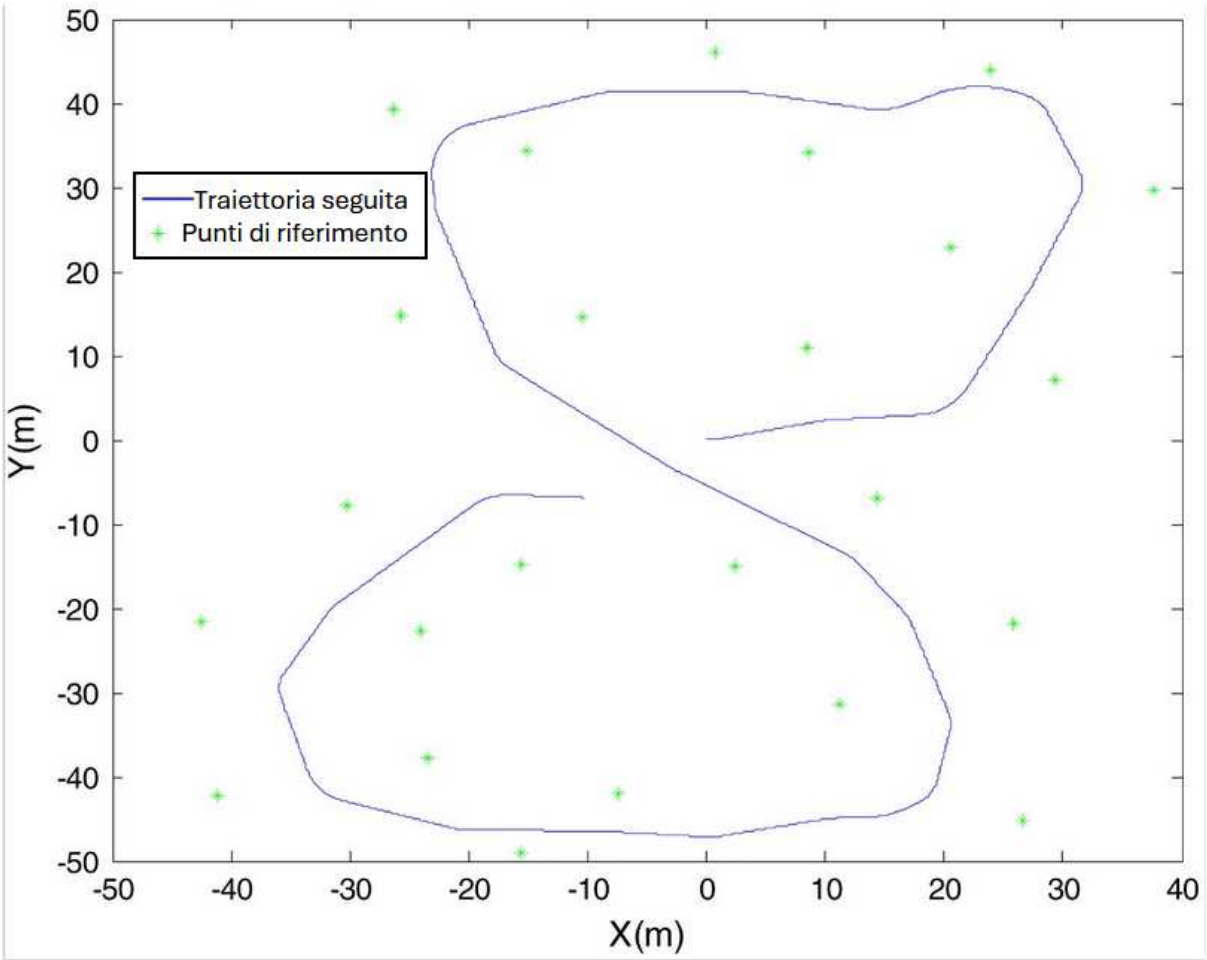

Figura 2: Mappa dell'ambiente e risultato simulazione (immagine tratta da [2])

3.4.2 Paragone tra le prestazioni

Le prestazioni del metodo proposto vengono confrontate con la localizzazione tramite *Particle Filter* e EKF, nella seguente condizione:

- rumore di controllo $\Big[\sigma_r = 0.3 \quad \sigma_\theta = 0.3\Big]$;
- rumore di misura $\Big[\sigma_r = 0.1 \quad \sigma_\theta = 0.1\Big]$;
- 50 particelle sia per il *Particle Swarm Optimization* che per il *Particle Filter*;
- 30 simulazioni Monte Carlo;

La figura 3 mostra le traiettorie stimate dai diversi metodi rispetto alla traiettoria vera, mentre nelle figure 4 e 5 si osserva come, in termini di RMSE di posizione e orientazione, il metodo proposto presenta prestazioni migliori rispetto agli altri.

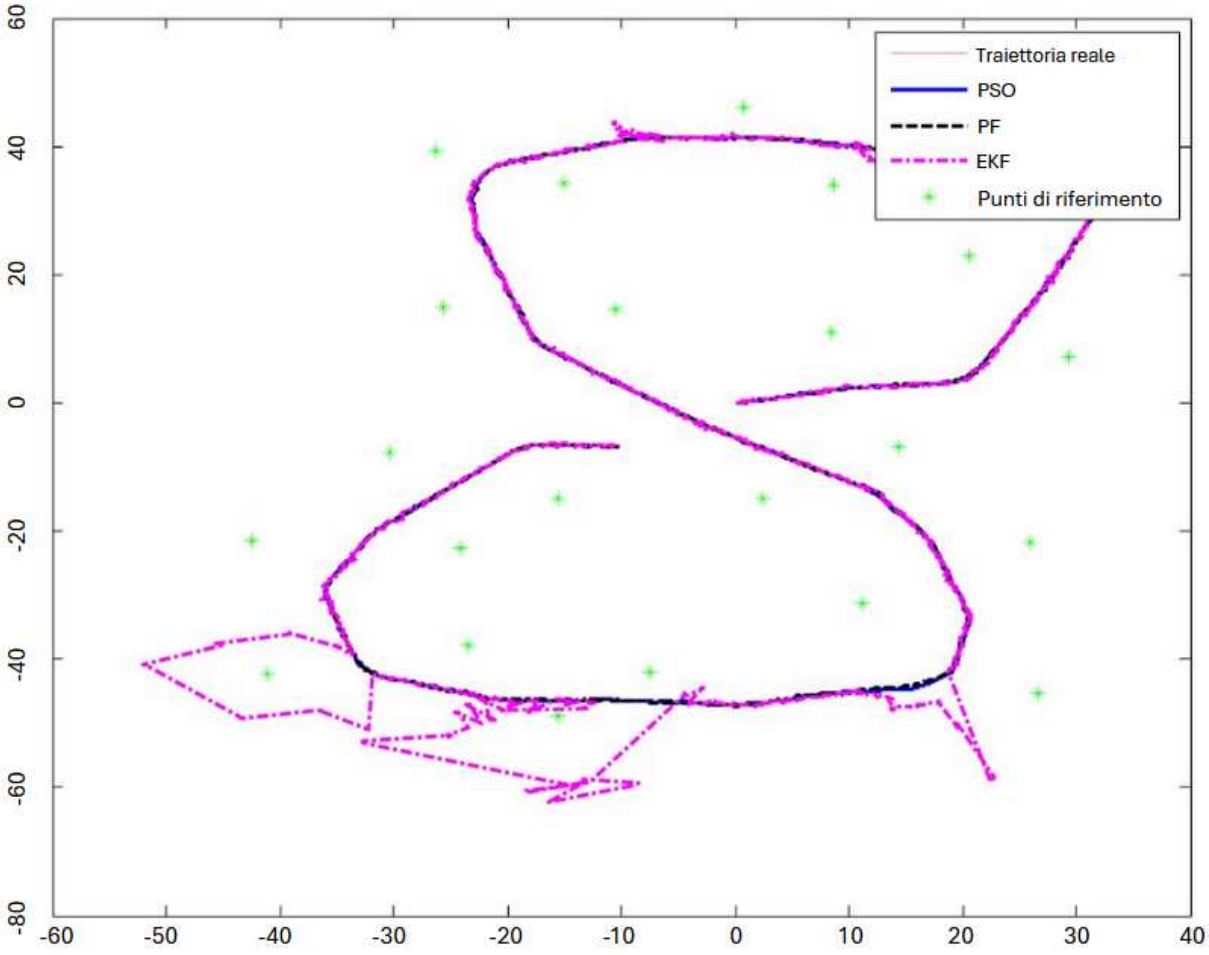

Figura 3: Traiettorie stimate (immagine tratta da [2])

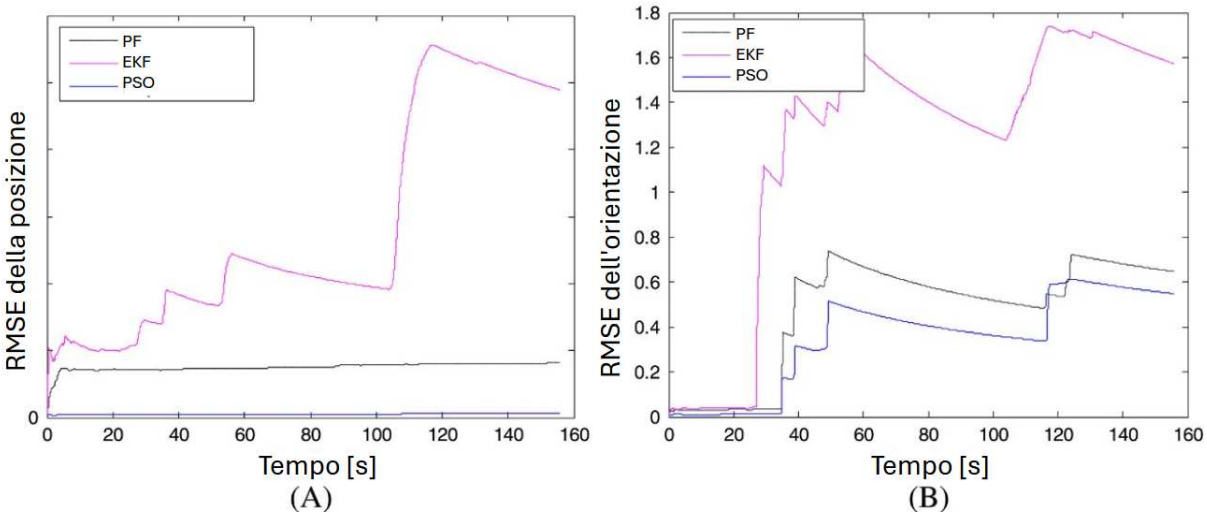

Figura 4: RMSE rispetto al tempo di posizione (A) e orientazione (B) (immagine tratta da [2])

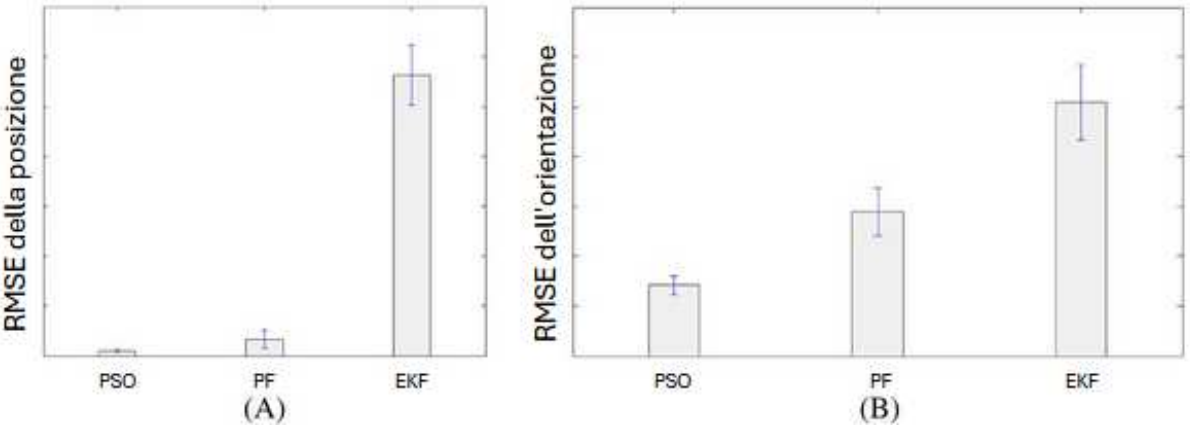

Figura 5: RMSE rispetto al tempo di posizione (A) e orientazione (B) (immagine tratta da [2])

3.4.3 Influenza del numero di particelle

Nel *Particle Swarm Optimization* e nel *Particle Filter* il numero di particelle è un importante parametro per valutare l'efficienza dell'algoritmo. Un numero troppo elevato di particelle porta ad avere una maggiore esplorazione dello spazio di ricerca aumentando così la qualità delle soluzioni ma, dall'altro lato, aumenta la complessità computazionale dell'algoritmo. Un numero eccessivamente basso di particelle invece porta ad avere complessità minori ma comporta una ridotta capacità di esplorazione dello spazio che, di conseguenza, può portare a soluzioni sub-ottimali. Tuttavia, è stato osservato che il metodo basato su *Particle Swarm Optimization*, rispetto a quello sul *Particle Filter*, non risente eccessivamente del numero di particelle. Questo è dovuto principalmente al fatto che nel *Particle Swarm Optimization* le particelle si muovono sempre verso le regioni a più alta probabilità, rendendo così non necessario un numero elevato per descrivere cor-

rettamente la distribuzione. Inoltre, le particelle mantengono la diversità in quanto si muovono in funzione, oltre che della posizione migliore globale, anche della propria trovata finora che generalmente è diversa per ciascuna di esse. Al contrario, nel *Particle Filter*, le particelle si muovono in modo casuale ma, soprattutto, perdono di diversità nel tempo, comportando così la necessità di avere un numero sufficientemente elevato per descrivere accuratamente la distribuzione di probabilità.

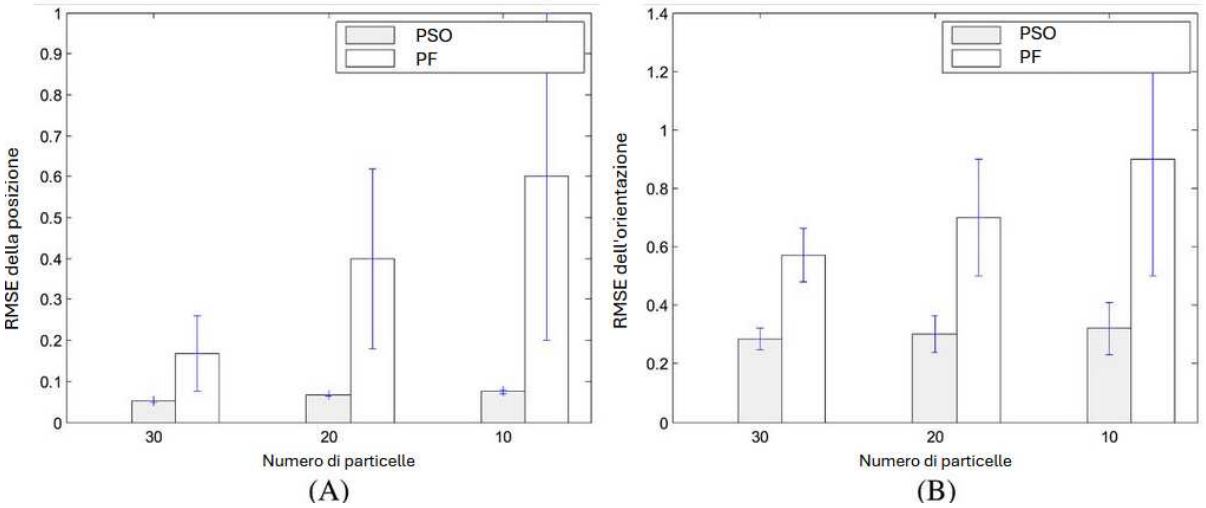

Figura 6: Influenza del numero di particelle (immagine tratta da [2])

Numero di Particelle	Algoritmo	RMSE Posizione		RMSE Orientazione		Tempo di esecuzione [s]
		Media	Varianza	Media	Varianza	
30	**PSO**	**0.05**	**0.002**	**0.28**	**0.031**	**60**
	PF	0.16	0.092	0.58	0.1	43
20	**PSO**	**0.065**	**0.0023**	**0.3**	**0.05**	**50**
	PF	0.4	0.22	0.7	0.2	36
10	**PSO**	**0.075**	**0.0043**	**0.32**	**0.08**	**40**
	PF	0.6	0.4	0.9	0.4	33

Tabella 1: Prestazioni con diversi numeri di particelle

3.4.4 Condizione di rumore non-Gaussiano

Effettuando simulazioni utilizzando rumori estratti da una distribuzione Gamma

$$f(x) = \frac{1}{\theta^k \Gamma(k)} x^{k-1} \exp\left(-\frac{x}{\theta}\right) \tag{81}$$

con $k, \theta > 0$ e $\Gamma(k)$ è la Funzione Gamma di Eulero [14], si è osservato come il metodo basato su stimatore PSO risulti più efficiente in termini di RMSE (Errore Quadratico Medio)[15] di posizione e orientazione rispetto agli altri. In particolare, il metodo basato su

[14] $\Gamma(k) = \int_0^\infty t^{k-1} \mathrm{e}^{-t}\, \mathrm{d}x$

[15] $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}$

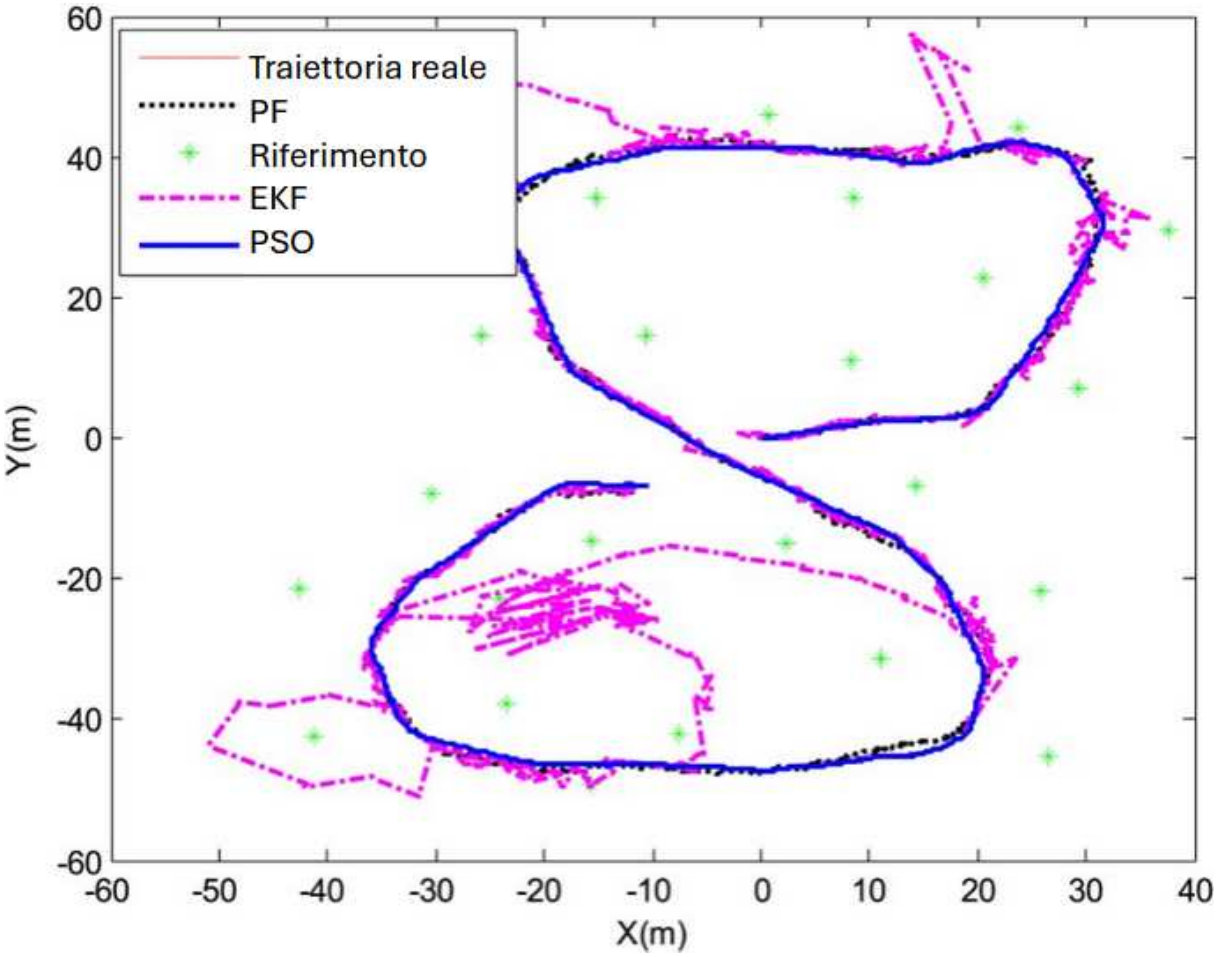

Figura 7: Traiettoria stimata in condizione di rumore non-Gaussiano (immagine tratta da [2])

Filtro di Kalman Esteso, essendo estremamente sensibile alla natura del rumore, risente particolarmente di questa condizione.

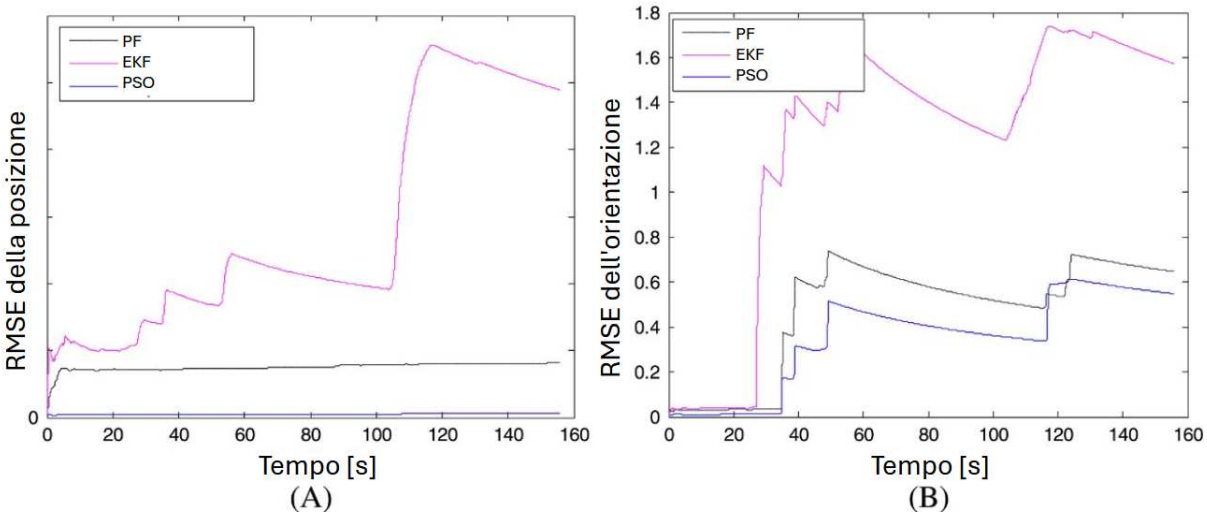

Figura 8: RMSE rispetto al tempo di posizione (A) e orientazione (B) in condizione di rumore non-Gaussiano (immagine tratta da [2])

3.4.5 Consistenza

La consistenza è una delle proprietà fondamentali degli stimatori che stabilisce se, all'aumentare della numerosità campionaria, aumenta anche la precisione dove, con precisione, si intende la convergenza al valore reale del parametro da stimare. Per verificare la consistenza dello stimatore tramite *Particle Swarm Optimization* si utilizza il criterio *Average NEES* (*Normalized Estimation Error Squared*) che, dati lo stato reale del sistema x_t e una sua stima con media $\hat{x}_t$ e matrice di covarianza $\hat{P}_t$, risulta:

$$\varepsilon_t = (x_t - \hat{x}_t)P_t^{-1}(x_t - \hat{x}_t) \tag{82}$$

e per calcolarne il valore medio si utilizza il Metodo Monte Carlo, che prevede una serie di simulazioni utilizzando campioni estratti casualmente da una distribuzione di probabilità predefinita. Chiamando N_r il numero di esperimenti, l'*average NEES* è:

$$\bar{\varepsilon}_t = \frac{1}{N_r}\sum_{i=1}^{N_r}\varepsilon_{it} \tag{83}$$

Notando che (83) è una forma quadratica e assumendo filtro lineare-Gaussiano, $N_R\bar{\varepsilon}_t$ risulta essere una distribuzione χ^2 [16] con $N_R dim x_t$ gradi di libertà. In questo caso x_t è la posizione del veicolo nel piano cartesiano, per cui $dim_{x_t} = 2$ e, con venti simulazioni Monte Carlo, l'intervallo di confidenza [17] al 95% è $\begin{bmatrix}1.3 & 2.79\end{bmatrix}$, che risulta migliore di quello ottenuto con il *Particle Filter*.

[16] chi-quadrato: distribuzione della somma di k variabili casuali
[17]

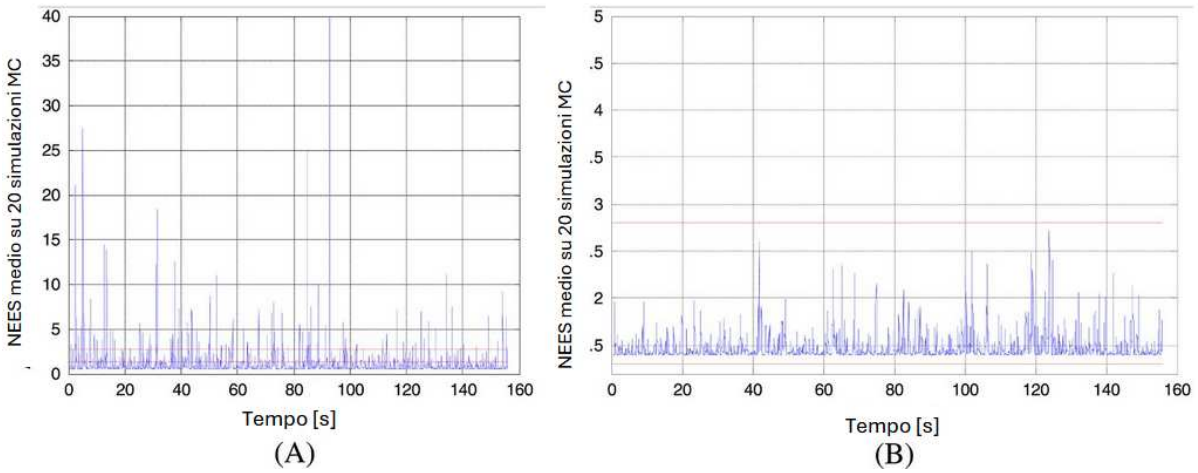

Figura 9: *Average* NEES con localizzazione tramite *Particle FIlter* (A) e *Particle Swarm Optmization* (B) (immagine tratta da [2])

3.4.6 Risultati sperimentali

La strategia di localizzazione proposta viene sperimentata su un veicolo a quattro ruote guidato all'interno del parco dell'Università di Sydney ed equipaggiato con:

- Encoder per le ruote: utilizzato per ricavare le informazioni odometriche;
- Sensore laser: necessario per ottenere informazioni riguardo i punti di riferimento che, in questo caso, sono pali in acciaio alti 60 mm e ricoperti di nastro riflettente;
- GPS: fornisce i dati che serviranno da verità di fondo e la mappa dell'ambiente.

Come dimostrato dalla figura 10, i dati odometrici provenienti dall'encoder risultano imprecisi per via dell'irregolarità del terreno che porta a un maggiore errore non sisitematico nella stima della posizione a causa dello slittamento delle ruote e dell'attrito, parametri non tenuti in considerazione nel modello di predizione usato.

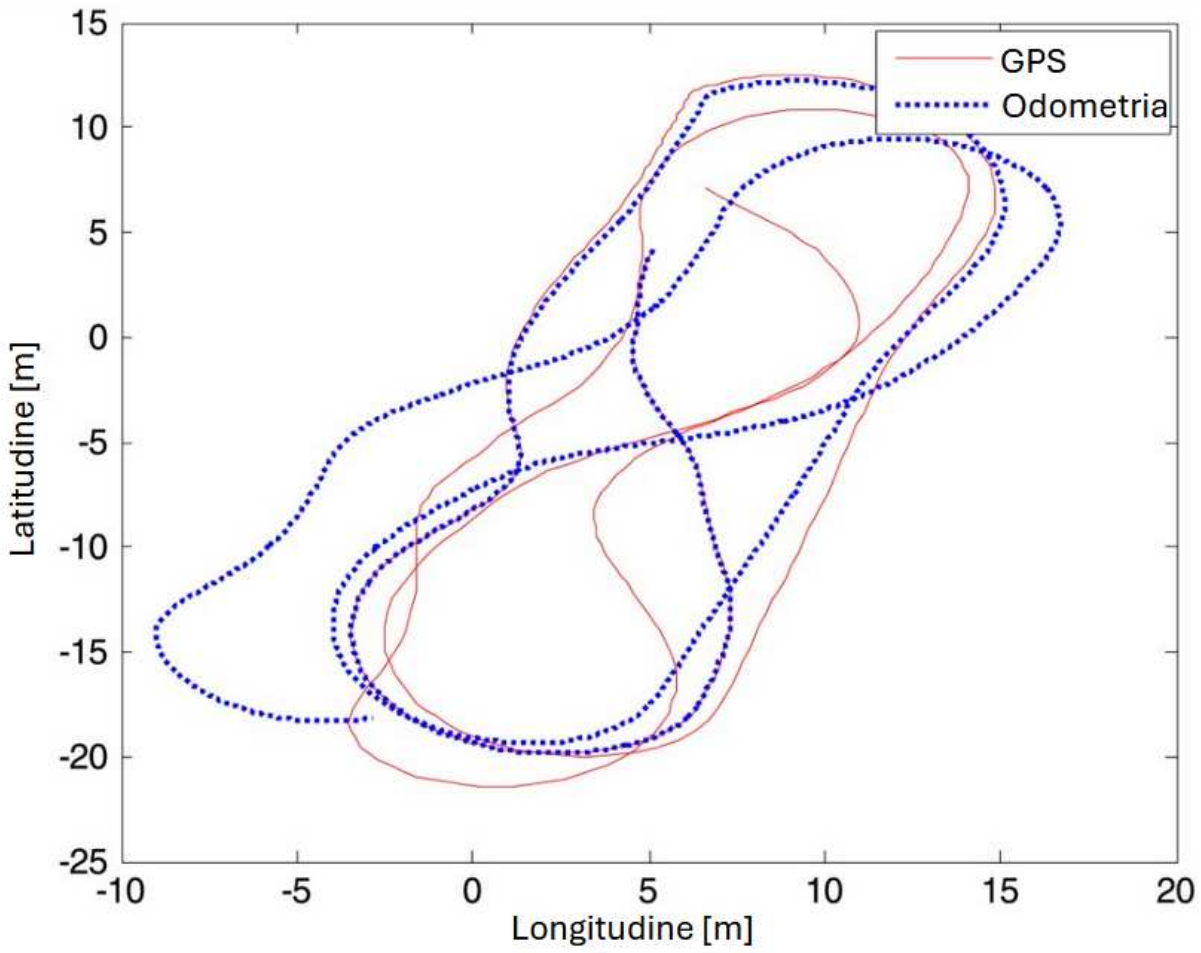

Figura 10: Confronto tra odometria e GPS (immagine tratta da [2])

Lo stesso esperimento viene eseguito utilizzando anche il *Particle Filter* in una condizione in cui la corrispondenza tra punti di riferimento e osservazioni non è nota e con venti particelle per entrambi gli algoritmi. Le figure 11 e 12 mostrano rispettivamente la traiettoria stimata e l'RMSE di entrambi i metodi, potendo così osservare le migliori prestazioni del metodo basato su *Particle Swarm Optimization*.

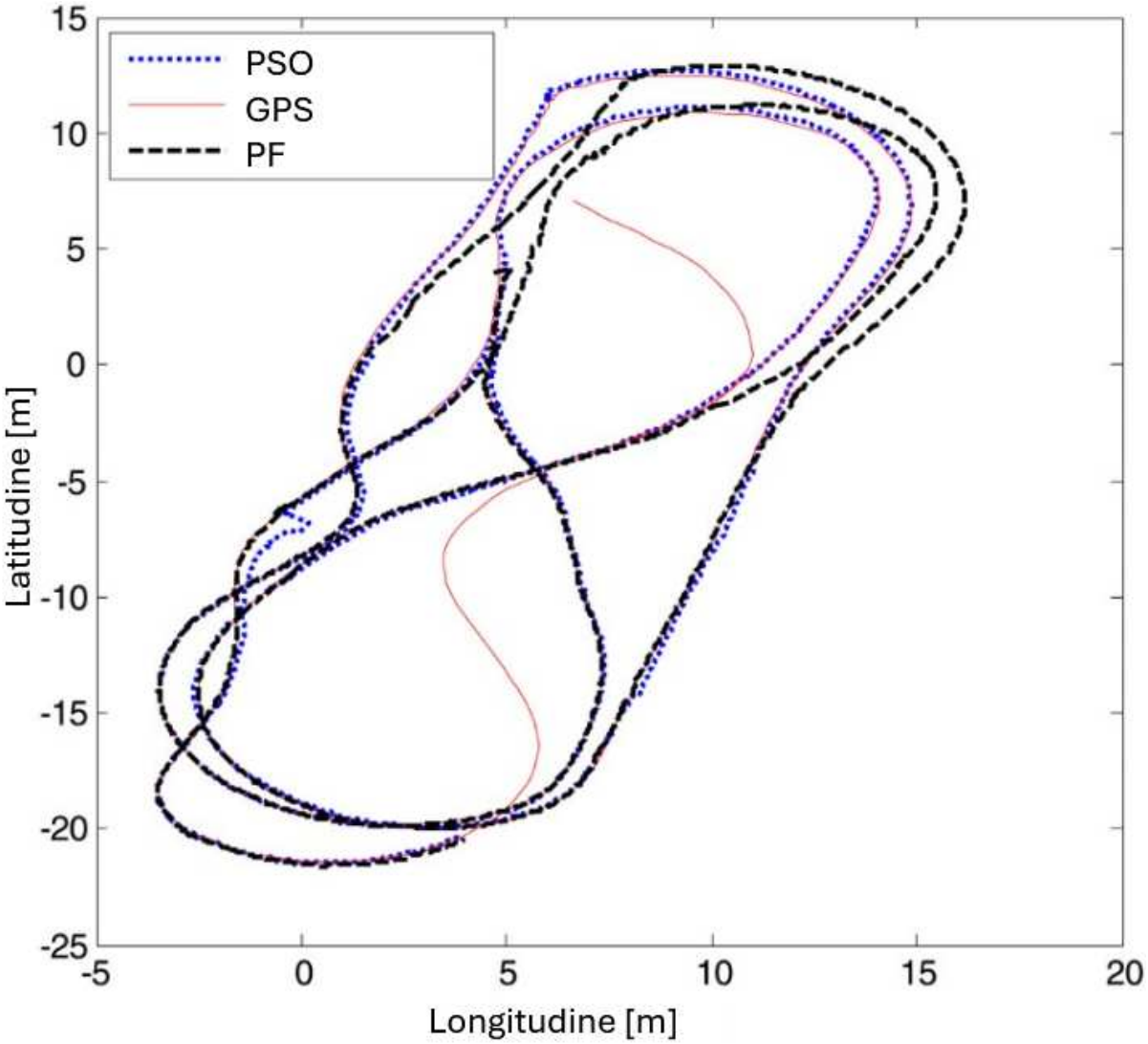

Figura 11: Traiettoria stimata da PSO e PF (immagine tratta da [2])

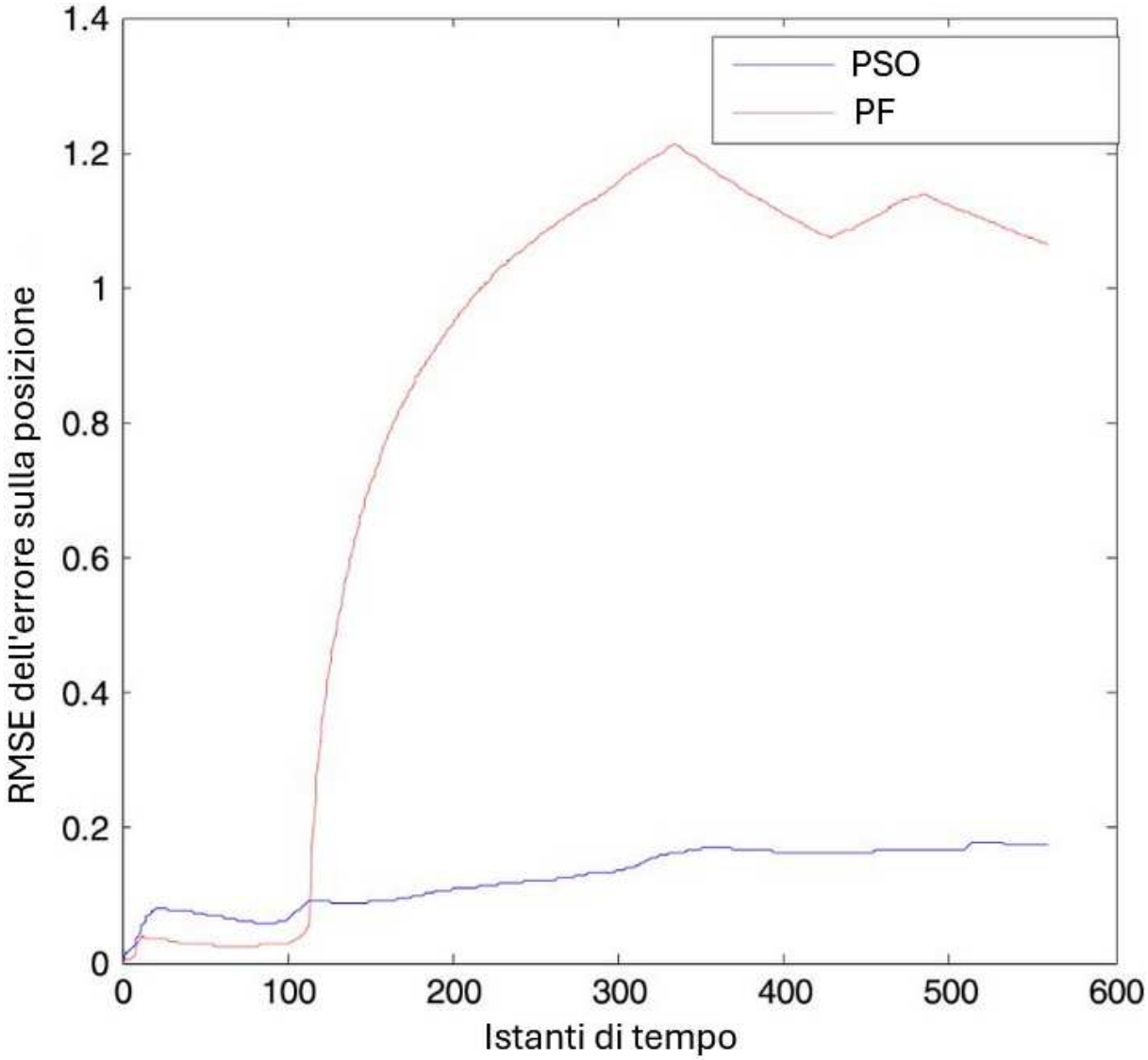

Figura 12: RMSE dell'errore sulla posizione in funzione del tempo per PSO e PF (immagine tratta da [2])

4 Conclusioni

L'obiettivo di questo progetto era quello di analizzare le principali strategie di localizzazione e, in seguito, approfondire in dettaglio l'approccio basato su stimatore *Particle Swarm Optimization*, paragonandolo ad approcci più comuni. Nella prima parte, dopo un'introduzione al problema della localizzazione, sono state esposte le tradizionali strategie probabilistiche per passare poi a quelle più recenti ed innovative come gli algoritmi di *SLAM* e le tecnologie *RFID*, mostrando come ogni approccio sia in grado di risolvere efficientemente un particolare ambito della localizzazione in ambienti e contesti differenti. Nella seconda parte, invece, viene discusso il funzionamento dell'algoritmo evolutivo *Particle Swarm Optimization* e il relativo processo di localizzazione per poi confrontarlo con gli approcci basati sugli algoritmi di stima *Particle Filter* e *Extended Kalman FIlter*. Dai risultati emersi da simulazioni ed esperimenti condotti dall'autore di [2], è stato possibile osservare come il metodo proposto risulti migliori in termini di RMSE per quanto riguarda stima di posizione e di orientazione anche in presenza di rumore non-Gaussiano, fattore che penalizza soprattutto l'EKF e, in generale, risulta più consistente degli altri metodi. In particolare, rispetto al PF, richiede una minore complessità computazionale per via della ridotta influenza del numero di particelle e non presenta il problema della perdita di diversità tra le particelle, evitando così la fase di *resampling*. In conclusione il PSO può rappresentare un'importante alternativa alle più comuni strategie di localizzazione per via della semplicità implementativa, della efficienza e della sua flessibilità, che può permettere a questo approccio di essere utilizzato per risolvere diversi problemi in ambito di localizzazione.

5 Bibliografia

Riferimenti bibliografici

[1] Prabin Kumar Panigrahi; Sukant Kishoro Bisoy. *Localization strategies for autonomous mobile robots: A review*. Journal of King Saud University Computer and Information Sciences. 2022.

[2] Ramazan Havangi. *Mobile robot localization based on PSO estimator*. Asian Journal of Control. 2019.

[3] Qiang Li; Ranyang Li; Kaifan Li; Wei Dai. *Kalman Filter and Its Application*. IEEE. 2015.

[4] Pradnya A. Vikhar. *Evolutionary algorithms: A critical review and its future prospects*. IEEE. 2016.

[5] M. Sanjeev Arulampalam; Simon Maskell; Neil Gordon; Tim Clapp. *A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking*. IEEE. 2002.

[6] H. Durrant-Whyte; T. Bailey. *Simultaneous localization and mapping: part I*. IEEE. 2006.

[7] T. Sanpechuda; L. Kovavisaruch. *A review of RFID localization: Applications and techniques* . IEEE. 2008.

[8] Mathieu Bouet; Aldri L. dos Santos. *RFID tags: Positioning principles and localization techniques*. IEEE. 2008.

[9] Tareq M. Shami; Ayman A. El-Saleh; Mohammed Alswaitti; Qasem Al-Tashi; Mhd Amen Summakieh; Seyedali Mirjalili. *Particle Swarm Optimization: A Comprehensive Survey*. IEEE. 2022.

[10] Alif Ridzuan Khairuddin; Mohamad Shukor Talib; Habibollah Haron. *Review on simultaneous localization and mapping (SLAM)*. IEEE. 2016.

[11] Lanxue Liu; Theng Fei; Zhengyu Zhu; Kangle Wu; Yutong Zhang. *A Survey of Evolutionary Algorithms*. IEEE. 2023.