UNIVERSITÀ DI PISA

Facoltà di Ingegneria

Laurea Specialistica in Ingegneria dell'Automazione

Tesi di laurea

Filtraggio e stima dello stato nei sistemi dinamici non lineari

Candidato:

Emanuele Crisostomi _____

Relatori:

Ch.mo Prof. Ing. Andrea Caiti _____

Ch.mo Prof. Ing. Mario Innocenti _____

Controrelatore:

Ch.mo Prof. Ing. Aldo Balestrino _____

Sessione di Laurea del 04/03/2005
Anno accademico 2004/2005
Consultazione consentita

SOMMARIO

Il lavoro di tesi affronta il problema della stima ricorsiva della densità di probabilità dello stato di un sistema tempo discreto dinamico non lineare. Sono presentate diverse tecniche già note in letteratura per il filtraggio dello stato e particolare attenzione è stata rivolta ai filtri a particelle in quanto permettono stime di densità di probabilità anche nel caso in qui queste non sono di tipo gaussiano. Il filtro a particelle proposto in tale lavoro di tesi è caratterizzato dalla possibilità di ottenere una stima di densità di probabilità dello stato anche in forma analitica mediante l'utilizzo del criterio di massima entropia; il concetto di entropia è stato usato anche per proporre diverse tecniche di ricampionamento delle particelle del filtro in modo da migliorare le prestazioni di stima. Esempi di diverse applicazioni di tale filtro sono stati presentati al fine di mostrare il funzionamento del filtro e confrontare le performance degli algoritmi di ricampionamento già noti in letteratura con quelli introdotti in questo lavoro basati su un esame dell'entropia.

ABSTRACT

This thesis faces the problem of sequential estimation of the state probability density function in non linear dynamic discrete-time systems. Several algorithms already known in the literature for state estimation are reviewed, with a focus on particle filters, since they provide a general framework even for non linear, non-gaussian problems. The particle filter proposed in this work provides an analytical estimation of the state probability density function based on the maximum entropy principle; the notion of entropy has also been used to develop new resampling algorithms in order to achieve better estimation performance. This work contains also several illustrative examples both to show the way a particle filter works and to compare already known resampling algorithms with those proposed based on a survey of the total entropy.

INDICE

INTRODUZIONE

Lo studio della evoluzione della densità di probabilità dello stato di un sistema è un problema diffusamente affrontato in letteratura per un duplice motivo. Innanzitutto il valor medio di tale densità di probabilità permette di ottenere una stima delle variabili di stato del sistema, e spesso la conoscenza di queste fornisce una descrizione sufficientemente completa dell'evoluzione dell'intero processo; inoltre la conoscenza dello stato del sistema è indirettamente necessaria per essere in grado di poter adeguatamente controllare il processo.

Nel primo capitolo di questo lavoro di tesi è appunto presentato un approccio generale per poter affrontare il problema di stima in un sistema dinamico tempo discreto dotato di un modello di evoluzione dello stato e di un modello di osservazione; vengono inoltre mostrati alcuni tra gli algoritmi di filtraggio maggiormente noti in letteratura. In generale non è possibile confrontare filtri diversi fra loro dato che le loro prestazioni variano in genere con il particolare problema che si deve affrontare; si può quindi ragionevolmente parlare di filtri più o meno appropriati per specifiche applicazioni. In tale contesto il filtro a particelle rappresenta uno degli algoritmi che negli ultimi anni si sta maggiormente affermando in quanto fornisce una stima dello stato di un sistema senza necessità di richiedere modelli lineari o gaussiani né del sistema né del rumore presente.

Il secondo capitolo è quindi dedicato a illustrare il funzionamento di un filtro a particelle e una sua possibile implementazione in maniera ricorsiva; nel terzo capitolo viene anche fornito un semplice *particle filter tool-box*, realizzato in ambiente Matlab, in modo da poter risolvere problemi di stima dello stato per applicazioni appartenenti a campi diversi fra loro. In particolare sono mostrati esempi di applicazioni relative a problemi di tracking o a problemi SLAM.

Il filtro a particelle mostrato si differenzia da altri filtri a particelle già disponibili sul mercato in quanto permette di conoscere una stima della densità di probabilità dello stato anche in forma analitica, mediante l'utilizzo del principio a massima entropia; grazie alla conoscenza dell'espressione di tale funzione è anche possibile disegnare la stima della densità di probabilità e valutare in modo visivo la sua evoluzione.

Nell'ultimo capitolo infine sono proposte alcune varianti del filtro a particelle in modo da ottenere generalmente delle prestazioni migliori indipendentemente dal tipo di applicazione desiderata; in particolare vengono proposti degli algoritmi di ricampionamento dinamici che tengono in considerazione non solamente la varianza delle particelle del filtro, ma anche la loro entropia totale. La complessità degli algoritmi proposti rimane la stessa di quelli già noti in

letteratura, nonostante il tempo di esecuzione di tali algoritmi sia leggermente superiore. Tale svantaggio può però essere ampiamente recuperato in quanto in alcuni casi il filtro a particelle modificato è in grado di fornire le stesse prestazioni del filtro originale usando un numero inferiore di particelle senza intaccarne l'affidabilità. Numerose simulazioni sono state quindi condotte per verificare le prestazioni dei nuovi algoritmi di ricampionamento e i risultati ottenuti hanno confermato come in molte situazioni tali algoritmi risultino vantaggiosi.

1. PROBLEMA DI STIMA RICORSIVA DELLO STATO

1.1 Approccio Bayesiano

Un sistema tempo discreto può generalmente essere descritto tramite le equazioni

$$x_k = f(x_{k-1}, v_{k-1}) \tag{1}$$

$$y_k = g(x_k, n_k) \tag{2},$$

dove $x_k \in \Re^{n_x}$ denota il vettore di stato del sistema al passo *k*, $y_k \in \Re^{n_y}$ il vettore delle osservazioni, $v_k \in \Re^{n_v}$ il rumore del processo, $n_k \in \Re^{n_n}$ il rumore di misurazione. Le funzioni $f : \Re^{n_x} \times \Re^{n_v} \mapsto \Re^{n_x}$ e $g : \Re^{n_x} \times \Re^{n_n} \mapsto \Re^{n_y}$ rappresentano i modelli di processo e di misurazione; in particolare l'aggiornamento dello stato segue un processo di Markov del primo ordine e le osservazioni si suppone siano indipendenti noti gli stati.

La densità di probabilità dello stato all'istante iniziale viene indicata con $p(x_0)$ ed è nota in letteratura come *distribuzione a priori*.

La densità di probabilità dello stato una volta note le osservazioni è invece detta *densità a posteriori* e può essere indicata come $p(x_{0:k} | y_{1:k})$, dove con $x_{0:k}$ si indica la matrice contenente i vettori di stato ai vari passi $\{x_0, x_1, ..., x_k\}$; analogamente $y_{1:k} = \{y_1, y_2, ..., y_k\}$ è la matrice contenente tutte le osservazioni.

In alcune applicazioni, nelle quali si è interessati ad una stima ricorsiva della densità di probabilità dello stato, può essere più utile riferirsi ad una densità di probabilità marginale come $p(x_k | y_{1:k})$, nota in letteratura come *filtering* (o *filtered*) *density*; d'ora in avanti si farà riferimento a tale densità di probabilità come la *densità filtrata*.

L'importanza della densità di probabilità a posteriori risiede nel fatto che questa è in grado di fornire notizie sullo stato del sistema in maniera indiretta attraverso delle misurazioni; tale approccio permette quindi di avere a disposizione ad ogni passo una stima dello stato del sistema, ciò può essere molto utile dato che lo stato contiene in genere tutte le informazioni rilevanti per descrivere il processo studiato.

Ad esempio, in un problema di tracking lo stato fornisce indicazioni sulle caratteristiche cinematiche del target (posizione, velocità, accelerazione..); alternativamente, in un problema

finanziario lo stato può contenere informazioni economiche quali il flusso di denaro, i tassi di interesse o l'inflazione.

La formulazione del sistema nello spazio degli stati come nelle equazioni (1) e (2) e la possibilità di avere a disposizione le misurazioni $y_{1:k}$ costituiscono la base ideale per il cosiddetto *approccio Bayesiano* per la soluzione del problema di stima dello stato.

L'approccio Bayesiano per la stima dello stato in un sistema dinamico consiste nello sfruttare tutte le informazioni a propria disposizione, compreso l'insieme di osservazioni $y_{1:k}$, per ricostruire la funzione densità di probabilità a posteriori. Dato che questa densità di probabilità contiene tutte le informazioni statistiche possibili relative allo stato, si potrebbe dire che la conoscenza di tale funzione offra la soluzione completa al problema di stima.

In molte applicazioni un filtro ricorsivo è una soluzione conveniente in quanto permette un'analisi sequenziale dei dati e non necessita né di dover mantenere in memoria tutte le osservazioni ricevute, né di dover cambiare le stime precedenti se una nuova misurazione diventa disponibile.

Un filtro di questo tipo consta essenzialmente di due fasi: la *predizione* e la *correzione* (o *aggiornamento*) come mostrato in Fig. 1.

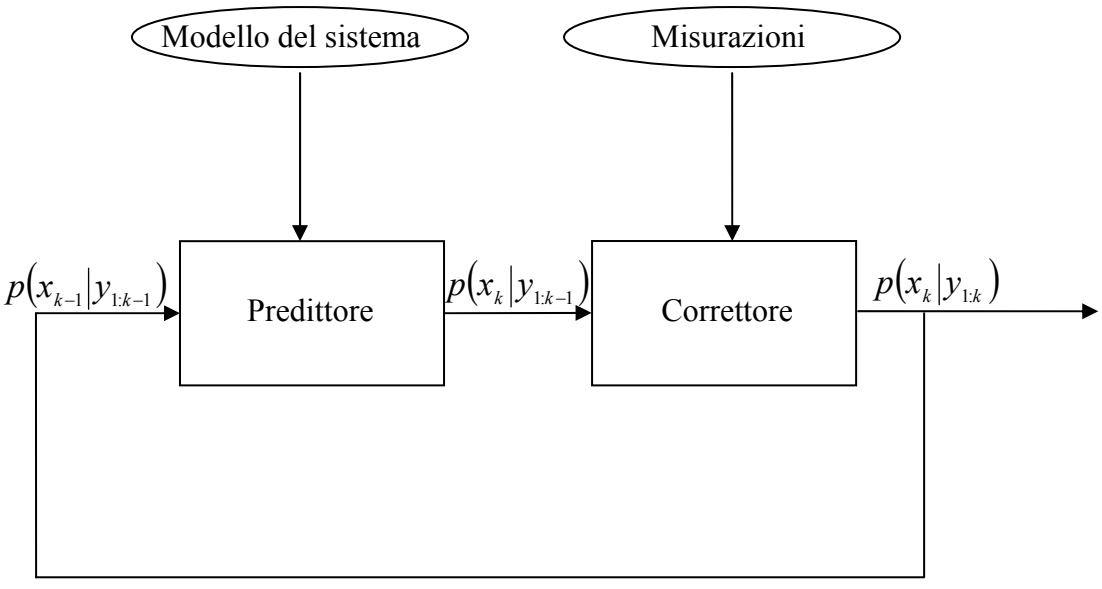

Fig. 1: Schema a blocchi di un filtro nonlineare in un approccio Bayesiano

Nella fase di predizione il modello del sistema viene usato per predire la densità di probabilità dello stato al passo seguente, tra una misurazione e la successiva. Dato che lo stato è

generalmente soggetto a disturbi non noti (modellati come rumore di processo), durante la fase di predizione la densità di probabilità dello stato viene traslata e deformata.

Nella fase di correzione l'ultima osservazione viene usata per modificare la densità di probabilità predetta. Tale operazione viene effettuata sfruttando il teorema di Bayes (3), che rappresenta il meccanismo che permette l'aggiornamento di una previsione alla luce di nuove informazioni:

$$p(x_{0:k}|y_{1:k}) = \frac{p(y_{1:k}|x_{0:k})p(x_{0:k})}{p(y_{1:k})} \tag{3}.$$

L'utilizzo ad ogni passo della formula di Bayes fa sì che l'intera procedura di stima venga chiamata in letteratura approccio Bayesiano.

In contrapposizione a tale approccio esiste anche un secondo approccio detto *nonBayesiano*, in cui i parametri non noti sono supposti costanti. Le due filosofie portano a stimatori diversi e il secondo problema è noto come *stima parametrica*, ma non verrà trattato in questo lavoro di tesi in quanto non strettamente legato al problema studiato.

Per poter stimare la densità di probabilità dello stato in un sistema dinamico descritto dalle equazioni (1) e (2) occorre avere alcune informazioni sul sistema quali:

- i modelli $f(\cdot)$ e $g(\cdot)$ che descrivono rispettivamente la funzione di aggiornamento dello stato e la funzione che lega le uscite allo stato;
- il modello del rumore di processo v_{k-1} e del rumore di misurazione n_k;
- l'insieme di osservazioni $y_{1:k} = \{y_1, y_2, ..., y_k\}$;
- la densità di probabilità a posteriori dello stato all'istante iniziale $p(x_0|y_0)$; dato che l'insieme y_0 coincide con l'insieme contenente nessuna osservazione, tale funzione di probabilità coincide con la densità a priori $p(x_0)$.

La fase di predizione può essere effettuata risolvendo direttamente l'equazione (4) nota come equazione di Chapman-Kolmogorov:

$$p(x_k|y_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1} \tag{4}.$$

Nell'equazione (4) si è sfruttato il fatto che $p(x_k|x_{k-1}, y_{1:k-1}) = p(x_k|x_{k-1})$ dato che l'equazione (1) del sistema dinamico di interesse è descritta da un processo di Markov di ordine uno.

Il modello probabilistico dell'evoluzione dello stato $p(x_k|x_{k-1})$, spesso chiamato come *densità di transizione*, può essere ricavato sfruttando l'equazione (1) e le statistiche note del rumore v_{k-1}.

Al passo *k* diventa disponibile una misurazione y_k e questa può essere usata per aggiornare la densità a priori attraverso la regola di Bayes (3), dove la costante di normalizzazione al denominatore può essere ricavata attraverso la (5)

$$p(y_k|y_{1:k-1}) = \int p(y_k|x_k)p(x_k|y_{1:k-1})dx_k \quad (5).$$

La funzione $p(y_k|x_k)$, nota come densità di *likelihood*, è definita dal modello di osservazione (2) e dalle note statistiche di n_k.

Le equazioni ricorsive (3) e (4) formano le basi per un algoritmo di calcolo esatto della densità di probabilità a posteriori. La propagazione ricorsiva della densità a posteriori rimane spesso però una soluzione concettuale dato che non può essere in generale ricavata analiticamente. Soluzioni esatte esistono in un insieme ristretto di casi, tra cui il più noto riguarda il caso dei filtri di Kalman-Wiener, esaminato in dettaglio nel prossimo paragrafo.

Riferimenti bibliografici riguardo l'approccio Bayesiano per la soluzione di problemi di stima della densità di probabilità a posteriori dello stato possono essere trovati in [1-4].

1.2 Il filtro di Kalman-Wiener

L'algoritmo noto in letteratura come filtro di Kalman-Wiener risulta ottimale nel caso in cui la densità di probabilità a posteriori ad ogni passo sia di tipo gaussiano. In particolare si può dimostrare che se in un sistema dinamico descritto dalle equazioni (1) e (2) la densità di probabilità $p(x_{k-1}|y_{1:k-1})$ è gaussiana, allora lo è anche $p(x_k|y_{1:k})$, ammesso che siano valide le seguenti assunzioni

- v_{k-1} e n_k sono processi congiuntamente gaussiani bianchi di parametri statistici noti, a media nulla, mutuamente indipendenti e indipendenti dallo stato iniziale
- $f(x_{k-1}, v_{k-1})$ è una funzione nota lineare in x_{k-1} e v_{k-1}

- $g(x_k, n_k)$ è una funzione nota lineare in x_k e n_k

In particolare è possibile mostrare che, valide le precedenti assunzioni, l'approccio Bayesiano fornisce le note equazioni del filtro di Kalman lineare, generalmente ricavate minimizzando il valore atteso dell'errore quadratico di stima.

La dimostrazione che il filtro di Kalman-Wiener può essere visto come caso particolare in un approccio Bayesiano è fornita in [5] e riportata in seguito.

Se il sistema dinamico è lineare e il rumore è additivo la (6) mostra una possibile rappresentazione di tale processo:

$$\begin{cases} x_k = \Phi x_{k-1} + \Gamma v_{k-1} \\ y_k = H x_k + n_k \end{cases} \quad (6);$$

dove si suppongono valide le seguenti relazioni:

- $p(x_k | y_{1:k})$ *è* *gaussiana*
- $E\{x_k | y_{1:k}\} = \hat{x}_k$
- $Cov\{x_k | y_{1:k}\} = P_k$
- $p(v_{k-1}, n_k | x_k, y_{1:k}) = p(v_{k-1}) p(n_k)$ (7).
- $E\{n_k\} = E\{v_{k-1}\} = 0$
- $Cov\{v_{k-1}\} = Q$
- $Cov\{n_k\} = R$

Riscrivendo l'equazione di Bayes, ma considerando solo la densità di probabilità filtrata, dato che si è interessati ad una stima ricorsiva, si ottiene la (3'):

$$p(x_{k+1} | y_{1:k+1}) = \frac{p(y_{k+1} | x_{k+1}) p(x_{k+1} | y_{1:k})}{p(y_{k+1} | y_{1:k})} \quad (3').$$

Analizzando le tre densità di probabilità che compaiono nell'equazione (3') e ricordando che per ipotesi si sta supponendo che $p(x_k|y_{1:k})$ sia gaussiana, si nota come tutte e tre le densità di probabilità siano a loro volta gaussiane. In particolare le loro statistiche valgono:

$$p(y_{k+1}|x_{k+1}) : \begin{cases} E\{y_{k+1}|x_{k+1}\} = Hx_{k+1} \\ Cov\{y_{k+1}|x_{k+1}\} = R \end{cases} . \tag{8},$$

$$p(x_{k+1}|y_{1:k}) : \begin{cases} E\{x_{k+1}|y_{1:k}\} = \Phi\hat{x}_k \\ Cov\{x_{k+1}|y_{1:k}\} = \Phi P_k \Phi^T + \Gamma Q \Gamma^T \end{cases} \tag{9},$$

e, usando la sostituzione $M_{k+1} = \Phi P_k \Phi^T + \Gamma Q \Gamma^T$ per semplificare i calcoli,

$$p(y_{k+1}|y_{1:k}) : \begin{cases} E\{y_{k+1}|y_{1:k}\} = H\Phi\hat{x}_k \\ Cov\{y_{k+1}|y_{1:k}\} = HM_{k+1}H^T + R \end{cases} \tag{10}.$$

Combinando le equazioni (8-10) si ottiene

$$p(x_{k+1}|y_{1:k+1}) = \frac{1}{\sqrt{(2\pi)^m |R|}} \frac{1}{\sqrt{(2\pi)^n |M_{k+1}|}} \cdot \sqrt{(2\pi)^m |HM_{k+1}H^T + R|} \cdot$$
$$\exp\left(-\frac{1}{2} (y_{k+1} - Hx_{k+1})^T R^{-1} (y_{k+1} - Hx_{k+1}) + (x_{k+1} - \Phi\hat{x}_k)^T M_{k+1}^{-1} (x_{k+1} - \Phi\hat{x}_k) \right.$$
$$\left. - (y_{k+1} - H\Phi\hat{x}_k)^T (HM_{k+1}H^T + R)^{-1} (y_{k+1} - H\Phi\hat{x}_k) \right) \tag{11}.$$

Tenendo conto della relazione $(A + BCB^T)^{-1} = A^{-1} - A^{-1}B(C^{-1} + B^T A^{-1} B)^{-1} B^T A^{-1}$, valida nel caso di matrici A e C non singolari, e notando come sia $(M_{k+1}^{-1} + H^T R^{-1} H) M_{k+1} H^T$ che $H^T R^{-1}(HM_{k+1}H^T + R)$ valgano $H^T + H^T R^{-1} H M_{k+1} H^T$, e pertanto $(M_{k+1}^{-1} + H^T R^{-1} H)^{-1} H^T R^{-1} = M_{k+1} H^T (HM_{k+1}H^T + R)^{-1}$, l'equazione (11) può essere semplificata in

$$p\left(x_{k+1}\middle|y_{1:k+1}\right)=\frac{\left|HM_{k+1}H^T+R\right|^{1/2}}{(2\pi)^{n/2}\left|R\right|^{1/2}\left|M_{k+1}\right|^{1/2}}\exp\left(-\frac{1}{2}\left(x_{k+1}-\hat{x}_{k+1}\right)^T P_{k+1}^{-1}\left(x_{k+1}-\hat{x}_{k+1}\right)\right) \quad (12),$$

dove si sono usate le seguente sostituzioni:

$$\hat{x}_{k+1}=\Phi\hat{x}_k+M_{k+1}H^T\left(HM_{k+1}H^T+R\right)^{-1}\left(y_{k+1}-H\Phi\hat{x}_k\right) \quad (13)$$

$$P_{k+1}^{-1}=M_{k+1}^{-1}+H^T R^{-1}H \quad (14).$$

Le equazioni (13) e (14) sono proprio le note equazioni del filtro di Kalman-Wiener nel caso di sistemi tempo-discreti.

1.3 Algoritmi ottimi e subottimi nell'approccio Bayesiano

Le assunzioni poste come ipotesi nel caso del filtro Kalman-Wiener garantiscono la possibilità di risolvere analiticamente le equazioni (3) e (4) e trovare la soluzione generale mostrata nell'approccio Bayesiano.

Un altro caso in cui è possibile ottenere ricorsivamente la densità filtrata $p\left(x_k\middle|y_{1:k}\right)$ in modo ottimale è quello in cui lo spazio degli stati è discreto e consta di un numero finito di stati; i metodi che calcolano la soluzione in tale circostanza sono noti come *grid-based methods* e sfruttano la possibilità di trattare gli integrali come sommatorie. In questo caso non ci sono vincoli sulle proprietà delle densità di probabilità di likelihood o di transizione. Tali metodi non rientrano nell'interesse di questa tesi per la particolarità delle ipotesi che sono alla loro base, pertanto sono stati solamente citati per completezza.

In molte situazioni di interesse però le assunzioni sia dei filtri di Kalman che dei *grid- based methods* non sono valide e pertanto è necessario ricorrere a degli algoritmi subottimi, cioè è necessario introdurre delle approssimazioni.

Tra gli algoritmi subottimi maggiormente noti ci sono i metodi *grid-based* approssimati, il filtro di Kalman esteso, l'*unscented Kalman filter* e i *particle filter*.

1.3.1 Metodi grid-based approssimati

I metodi grid-based, o basati su griglia, possono essere usati qualora lo spazio degli stati fosse continuo ma potesse essere scomposto in N_s celle. Un esempio può essere quello di un robot che si muove in una mappa topologica dove i possibili stati sono $\{cucina, ingresso, salone...\}$.

Ancora una volta questi metodi vengono presentati soprattutto per completezza della trattazione dato che la particolarità delle assunzioni preliminari li emargina dal resto del lavoro di tesi.

In tali metodi la densità di probabilità a posteriori può essere approssimata tramite le densità di probabilità cacolate in ciascuna cella:

$$p\left(x_{k-1}\middle|z_{1:k-1}\right)\approx\sum_{i=1}^{N_s}w_{k-1|k-1}^{i}\delta\left(x_{k-1}-x_{k-1}^{i}\right) \tag{15},$$

le equazioni di predizione e aggiornamento sono invece rispettivamente la (16) e la (17):

$$p\left(x_k\middle|y_{1:k-1}\right)\approx\sum_{i=1}^{N_s}w_{k-1|k-1}^{i}\delta\left(x_k-x_{k-1}^{i}\right) \tag{16}$$

$$p\left(x_k\middle|y_{1:k}\right)\approx\sum_{i=1}^{N_s}w_{k|k}^{i}\delta\left(x_k-x_k^{i}\right) \tag{17}$$

dove i *w* rappresentano i pesi da assegnare a ciascuna cella; il calcolo dei *w* può essere effettuato in modo ricorsivo e, per poter semplificare i calcoli, in genere si assume che i pesi siano calcolati nel centro della cella. In questo caso si ha che (18) e (19) sono le equazioni che permettono il calcolo ricorsivo dei pesi:

$$w_{k|k-1}^{i}=\sum_{j=1}^{N_s}w_{k-1|k-1}^{j}p\left(\bar{x}_k^{i}\middle|\bar{x}_{k-1}^{j}\right) \tag{18}$$

$$w_{k|k}^{i}=\frac{w_{k|k-1}^{i}p\left(y_k\middle|\bar{x}_k^{i}\right)}{\sum_{j=1}^{N_s}w_{k|k-1}^{j}p\left(y_k\middle|\bar{x}_k^{j}\right)} \tag{19}.$$

In questo caso $\bar{x}_k^i$ rappresenta il centro della i-esima cella al passo k. Nel caso in cui lo spazio degli stati fosse continuo ci sarebbe bisogno di una griglia molto densa per ottenere buone approssimazioni. Con l'incrementare delle dimensioni dello spazio degli stati il costo computazionale dell'approccio incrementa drammaticamente, e se l'estensione non è finita allora c'è bisogno di troncare il dominio di approssimazione. In letteratura sono stati suggeriti diversi metodi per decidere come troncare il dominio dello spazio degli stati, come ad esempio l'utilizzo della disuguaglianza di Chebyshev [1] per avere una stima della perdita di quantità di probabilità a causa del troncamento della coda di una pdf.

Un altro svantaggio di tale approccio consiste nel fatto che lo spazio degli stati viene suddiviso in celle in modo uniforme sin dall'inizio; pertanto non è possibile grigliare gli stati in modo più furbo, ad esempio far sì di avere una risoluzione più fitta nelle regioni in cui la densità di probabilità è maggiore, a meno che non si abbiano a priori delle notizie su tale funzione.

Filtri che usano l'approccio *grid-based* approssimato sono in genere noti come HMM (Hidden Markov Model); il loro nome deriva dal fatto che a differenza dei modelli di Markov standard dove la relazione tra stato e osservazione è unica, in questo caso l'incertezza tra una data misurazione e lo stato è modellata tramite una densità di probabilità. Una caratteristica degli HMM è che in genere modellano le probabilità di rimanere in un determinato stato mediante funzioni esponenziali del tempo; le applicazioni più comuni riguardano il campo del riconoscimento vocale. Ulteriori informazioni a riguardo e particolari applicazioni si possono trovare in [8-10].

1.3.2 Filtro di Kalman esteso

Il filtro di Kalman esteso, spesso abbreviato in letteratura con la sigla EKF (Extended Kalman Filter) rappresenta senza dubbio l'algoritmo più noto per risolvere il problema di filtraggio nonlineare e non-gaussiano.

Tale filtro rappresenta l'estensione del filtro di Kalman-Wiener nel caso di filtraggio non-lineare e sfrutta il principio della linearizzazione delle funzioni di evoluzione e di misurazione usando l'espansione in serie di Taylor.

In dettaglio l'EKF è uno stimatore MMSE (Minimum Mean-Square-Error) basato sull'espansione in serie di Taylor delle funzioni non lineari f e g che compaiono nella (1) e nella (2), intorno alle stime $\bar{x}_{k|k-1}$ degli stati x_k, come nell'equazione (20)

$$f(x_k) = f\left(\bar{x}_{k|k-1}\right) + \left.\frac{\partial f(x_k)}{\partial x_k}\right|_{x_k=\bar{x}_{k|k-1}} \left(x_k - \bar{x}_{k|k-1}\right) + \ldots \qquad (20).$$

Usando solamente i termini lineari si trovano le note equazioni per l'aggiornamento del valor medio e della varianza dell'approssimazione gaussiana alla distribuzione a posteriori dello stato:

$$\begin{aligned}
&\bar{x}_{k|k-1} = f(\bar{x}_{k-1}, 0) \\
&P_{k|k-1} = F_k P_{k-1} F_k^T + G_k Q_k G_k^T \\
&K_k = P_{k|k-1} H_k^T \left[U_k R_k U_k^T + H_k P_{k|k-1} H_k^T \right]^{-1} \\
&\bar{x}_k = \bar{x}_{k|k-1} + K_k \left(y_k - g(\bar{x}_{k|k-1}, 0) \right) \\
&P_k = P_{k|k-1} - K_k H_k P_{k|k-1}
\end{aligned} \qquad (21),$$

dove K_k è noto come guadagno di Kalman, Q_k è la varianza del rumore di processo (che si presume sia bianco gaussiano a valor medio nullo), R_k è la varianza del rumore di misurazione (che si presume sempre bianco gaussiano a valor medio nullo), $F_k = \frac{\partial f(x_k)}{\partial x_k}$ e $G_k = \frac{\partial f(v_k)}{\partial v_k}$ sono i Jacobiani del modello del processo, mentre $H_k = \frac{\partial g(x_k)}{\partial x_k}$ e $U_k = \frac{\partial g(n_k)}{\partial n_k}$ sono i Jacobiani del modello di osservazione.

Poiché il filtro di Kalman esteso usa solo i termini del primo ordine nell'espansione in serie di Taylor delle funzioni non lineari, spesso introduce grandi errori nelle stime statistiche della distribuzione a posteriori dello stato. Questo è specialmente evidente quando i modelli sono fortemente non lineari: in questo caso l'assunzione di linearità locale non è più valida dato che gli effetti dei termini di ordine superiore al secondo nell'espansione in serie di Taylor non sono più trascurabili. Un'ulteriore approssimazione introdotta dall'EKF consiste nel calcolare le densità di probabilità a posteriori ad ogni passo solo per mezzo del valor medio e della covarianza; ciò è particolarmente penalizzante nel caso in cui tali funzioni non fossero gaussiane.

Nonostante questi svantaggi il filtro di Kalman esteso è largamente diffuso e usato, sia grazie alla sua facilità di implementazione sia soprattutto perché filtri migliori sono stati introdotti in tempi relativamente recenti. In particolare nel prossimo paragrafo viene mostrato un altro algoritmo subottimo per un approccio di stima Bayesiano molto simile all'EKF, ma migliore per molti aspetti.

1.4 Il filtro di Kalman unscented

Il filtro di Kalman unscented (letteralmente "inodore") è stato introdotto nel 1997 da Julier e Uhlmann [11-12]. Tale filtro, abbreviato in letteratura tramite la sigla UKF, è uno stimatore ricorsivo MMSE che affronta alcuni dei problemi di approssimazione presenti nel filtro di Kalman esteso.

L'UKF si basa sull'intuizione che sia più facile approssimare una distribuzione gaussiana di quanto non lo sia approssimare una funzione arbitrariamente non lineare. A differenza dell'EKF infatti, l'UKF non linearizza le funzioni f e g in (1) e (2), ma approssima la distribuzione della variabile aleatoria gaussiana tramite un numero finito di campioni scelti in modo deterministico. Tali campioni, che vengono chiamati in questo contesto *sigma points*, mantengono il vero valor medio e la covarianza della variabile aleatoria e, quando vengono propagati attraverso il sistema reale non lineare, si trasformano in campioni che ancora mantengono il vero valor medio e la vera covarianza accuratamente fino al secondo ordine, con errori introdotti solo dai termini di ordine superiore.

Per spiegare il funzionamento di tale filtro conviene dapprima presentare la *trasformazione unscented* e la *trasformazione unscented scalata*. La seconda fornisce l'algoritmo chiave per il filtro unscented di Kalman.

1.4.1 La trasformazione unscented

La trasformazione unscented, abbreviata in letteratura con la sigla UT (Unscented Transformation), è un metodo che permette di calcolare le statistiche di una variabile aleatoria che è sottoposta ad una trasformazione non lineare.

Supponiamo che una variabile aleatoria x di dimensione n_x si propaghi attraverso una funzione non lineare $g : \Re^{n_x} \mapsto \Re^{n_y}$ in modo da generare y, cioè

$$y = g(x) \tag{22},$$

dove si suppone che x abbia valor medio $\bar{x}$ e covarianza P_x.

Il primo passo per calcolare le statistiche di y relative ai suoi primi due momenti consiste nello scegliere in modo deterministico $2n_x + 1$ campioni, o sigma points, in modo tale che questi mantengano le statistiche della variabile aleatoria iniziale x.

I sigma points sono caratterizzati da una coppia $\{\chi_i, W_i; i = 1,...,2n_x + 1\}$, dove χ_i rappresenta il valore del campione e W_i il suo peso. I sigma points sono scelti in modo deterministico nella seguente maniera:

$$S_0 = \{\chi_0, W_0\} = \left\{\bar{x}, \frac{k}{n_x + k}\right\}, i = 0 \quad (23),$$

$$S_i = \{\chi_i, W_i\} = \left\{\bar{x} + \left(\sqrt{(n_x + k)P_x}\right)_i, \frac{1}{2(n_x + k)}\right\}, i = 1,...,n_x \quad (24),$$

$$S_i = \{\chi_i, W_i\} = \left\{\bar{x} - \left(\sqrt{(n_x + k)P_x}\right)_i, \frac{1}{2(n_x + k)}\right\}, i = n_x + 1,...,2n_x \quad (25),$$

dove k è un parametro di scala e $\left(\sqrt{(n_x + k)P_x}\right)_i$ è la i-esima colonna della radice quadrata matriciale di $(n_x + k)P_x$. Si può notare come la somma di tutti i pesi faccia effettivamente 1, come la media pesata dei sigma points risulti in effetti $\bar{x}$ e come anche la covarianza rimanga P_x.

A questo punto non resta che propagare tutti i sigma points attraverso la funzione non lineare g come nella (26):

$$\gamma_i = g(x_i), i = 0,...,2n_x \quad (26)$$

e le stime della media e della covarianza di y possono essere calcolate rispettivamente per mezzo della (27) e della (28):

$$\bar{y} = \sum_{i=0}^{2n_x} W_i \gamma_i \quad (27)$$

$$P_y = \sum_{i=0}^{2n_x} W_i (\gamma_i - \bar{y})(\gamma_i - \bar{y})^T \quad (28).$$

In tale modo le stime del valor medio e della covarianza sono accurate fino al secondo ordine della espansione in serie di Taylor di $g(x)$ per ogni funzione non lineare. Gli errori introdotti dai momenti del terzo ordine in poi possono essere attenuati da un'opportuna scelta del parametro *k*.

Si può quindi notare come la stima ottenuta tramite un UKF sia migliore rispetto a quella che si può ottenere con un EKF in cui il valor medio e la covarianza della densità a posteriori sono accurati al primo ordine con tutti i momenti di ordine superiore troncati.

Un vantaggio del filtro di Kalman unscented rispetto ad esempio ad un metodo grid-based è che la dimensione dello spazio degli stati non influisce più in maniera pesante nelle prestazioni

dello stimatore. Nei metodi grid-based infatti il numero dei punti per la definizione della griglia per lo stato aumentava in modo esponenziale con l'aumentare delle dimensioni del problema.

Nel caso del filtro di Kalman l'aumentare delle dimensioni dello spazio di stato comporta semplicemente un incremento dei sigma points necessari per la valutazione della densità di probabilità a posteriori. In realtà un aumento delle dimensioni dello spazio di stato ha un effetto negativo indiretto sulle prestazioni del filtro, l'aumentare di n_x infatti comporta un aumento del raggio dell'ipersfera che circonda tutti i sigma points. Quindi, anche se il valor medio e la covarianza della distribuzione a priori sono ancora catturati bene dai sigma points, il prezzo da pagare con l'aumentare del raggio dell'ipersfera è il rischio di campionare anche effetti non-locali. Se le non-linearità del problema sono quindi molto severe, è possibile incontrare alcune difficoltà significative.

Tale problema può comunque essere attenuato scalando la nuvola di sigma points più vicino o più lontano dal valor medio, attraverso un'opportuna scelta di *k* o, come si vedrà nel prossimo paragrafo, usando ulteriori fattori di scala.

Si può notare infatti dalle equazioni (24) e (25) come quando $k = 0$ la distanza dei punti sia proporzionale a $\sqrt{n_x}$, quando *k* è positivo i sigma points si allontanino da $\bar{x}$, quando *k* è negativo i punti vengano scalati verso il valor medio. Nel caso particolare in cui $k = 3 - n_x$, si ottiene il risultato di rendere insensibile il filtro alle dimensioni dello spazio di stato; quando però $k = 3 - n_x < 0$ il peso W_0 diventa negativo e la covarianza può risultare semidefinita negativa. Per risolvere tale problema e fornire maggiore libertà per scalare i sigma points lo stesso Julier che per primo propose il filtro di Kalman unscented ha sviluppato nel 2000 la trasformazione unscented scalata [13].

1.4.2 La trasformazione unscented scalata

La trasformazione unscented scalata, abbreviata spesso tramite la sigla SUT (Scaled Unscented Transformation), fornisce ulteriori gradi di libertà nel controllo del posizionamento dei sigma points cercando di ovviare al rischio di ottenere una matrice di covarianza semidefinita negativa.

Allo scopo di mantenere il più possibile inalterate le equazioni presentate nel precedente paragrafo conviene introdurre un parametro λ calcolato come nella (29):

$$\lambda = \alpha^2 \left(n_x + k \right) - n_x \qquad (29),$$

dove α è un parametro che ha la caratteristica di scalare geometricamente i termini di ordine superiore al secondo. A questo punto le equazioni della trasformazione scalata sono analoghe alle (23-25), sostituendo *k* con λ .
In particolare:

$$S_0 = \{\chi_0, W_0\} = \left\{\bar{x}, \frac{\lambda}{n_x + \lambda}\right\}, i = 0 \quad (23'),$$

$$S_i = \{\chi_i, W_i\} = \left\{\bar{x} + \left(\sqrt{(n_x + \lambda)P_x}\right)_i, \frac{1}{2(n_x + \lambda)}\right\}, i = 1,...,n_x \quad (24'),$$

$$S_i = \{\chi_i, W_i\} = \left\{\bar{x} - \left(\sqrt{(n_x + \lambda)P_x}\right)_i, \frac{1}{2(n_x + \lambda)}\right\}, i = n_x + 1,...,2n_x \quad (25').$$

Spesso può tornare utile usare il peso chiamato $W_0^m = W_0$ per il calcolo della media ed usare un diverso peso $W_0^c = W_0^m + \left(1 - \alpha^2 + \beta^2\right)$ nel calcolo della covarianza.

Il parametro $\beta \geq 0$ può essere usato per incorporare eventuali conoscenze a priori dei momenti di ordine superiore; nel caso di una distribuzione a priori gaussiana un valore ottimale di β è 2. Questo parametro può essere usato anche per agire sullo spessore delle code della distribuzione a posteriori.

E' in genere conveniente scegliere il parametro $k \geq 0$ per garantire la positiva semidefinitività della matrice di covarianza; in genere non è un valore critico e spesso lo si pone uguale a 0.

Il parametro α, compreso tra 0 e 1, controlla l'ampiezza della distribuzione dei sigma points e conviene sceglierlo piccolo in modo da evitare di considerare effetti non locali qualora le non linearità in gioco fossero particolarmente grandi.

1.4.3 Implementazione di un filtro di Kalman unscented

L'implementazione di un filtro di Kalman unscented è pressoché immediata una volta presentato il funzionamento della trasformazione unscented. Dato che il filtro si basa su un algoritmo applicato ricorsivamente è conveniente mostrare il funzionamento dell'UKF direttamente presentandolo come un algoritmo. Nella sintassi successiva il vettore di stato aumentato viene indicato con $x^a = \begin{bmatrix} x^T & v^T & n^T \end{bmatrix}^T$, e analogamente i sigma points aumentati diventano $\chi^a = \begin{bmatrix} (\chi^x)^T & (\chi^v)^T & (\chi^n)^T \end{bmatrix}^T$, λ è il parametro supposto noto per poter effettuare la

trasformazione unscented scalata, $n_a = n_x + n_v + n_n$, Q è la covarianza del rumore di processo, R è la covarianza del rumore di misurazione. I passi dell'algoritmo risultano quindi:

- Inizializzazione del filtro:

$$\bar{x}_0 = E\{x_0\}$$
$$P_0 = E\{(x_0 - \bar{x}_0)(x_0 - \bar{x}_0)^T\}$$
$$\bar{x}_0^a = E\{x^a\} = \begin{bmatrix} \bar{x}_0^T & 0 & 0 \end{bmatrix}^T$$
$$P_0^a = E\{(x_0^a - \bar{x}_0^a)(x_0^a - \bar{x}_0^a)^T\} = \begin{bmatrix} P_0 & 0 & 0 \\ 0 & Q & 0 \\ 0 & 0 & R \end{bmatrix}$$

- Per $k \in \{1,..,\infty\}$,

a) Calcolare i sigma points:

$$\chi_{k-1}^a = \begin{bmatrix} \bar{x}_{k-1}^a & \bar{x}_{k-1}^a \pm \sqrt{(n_a + \lambda)P_{k-1}^a} \end{bmatrix}$$

b) Aggiornamento ad un passo

$$\chi_{k|k-1}^a = f(\chi_{k-1}^x, \chi_{k-1}^v)$$
$$\bar{x}_{k|k-1} = \sum_{i=0}^{2n_a} W_i^m \chi_{i,k|k-1}^x$$
$$P_{k|k-1} = \sum_{1=0}^{2n_a} W_i^c \left[\chi_{i,k|k-1}^x - \bar{x}_{k|k-1}\right] \left[\chi_{i,k|k-1}^x - \bar{x}_{k|k-1}\right]^T$$
$$\gamma_{k|k-1} = h(\chi_{k|k-1}^x, \chi_{k-1}^n)$$
$$\bar{y}_{k|k-1} = \sum_{i=0}^{2n_a} W_i^m \gamma_{1,k|k-1}$$

c) Equazioni di aggiornamento della misurazione

$$P_{y_k y_k} = \sum_{i=0}^{2n_a} W_i^c \left[\gamma_{i,k|k-1} - \bar{y}_{k|k-1}\right] \left[\gamma_{i,k|k-1} - \bar{y}_{k|k-1}\right]^T$$

$$P_{x_k y_k} = \sum_{i=0}^{2n_a} W_i^c \left[\chi_{i,k|k-1} - \bar{x}_{k|k-1}\right] \left[\gamma_{i,k|k-1} - \bar{y}_{k|k-1}\right]^T$$

$$K_k = P_{x_k y_k} P_{y_k y_k}^{-1}$$

$$\bar{x}_k = \bar{x}_{k|k-1} + K_k \left(y_k - \bar{y}_{k|k-1}\right)$$

$$P_k = P_{k|k-1} - K_k P_{y_k y_k} K_k^T .$$

Il passaggio che richiede maggior tempo di calcolo dell'algoritmo consiste nella computazione della radice quadrata di una matrice, operazione che può essere effettuata in modo diretto usando una fattorizzazione di Cholesky; poiché però le matrici di covarianza sono espresse in modo ricorsivo, allora anche la complessità dell'operazione di radice è dell'ordine di n_x^2 usando un aggiornamento ricorsivo della fattorizzazione di Cholesky.

Nel caso comune in cui il rumore del processo e della misurazione sono puramente additivi, la complessità computazionale diminuisce, dato che non c'è più bisogno di usare il vettore di stato aumentato. In questo modo diminuisce sia la dimensione che il numero totale dei sigma points usati. Le covarianze dei rumori possono quindi essere incorporate nella covarianza dello stato semplicemente in maniera additiva.

Il punto debole del filtro di Kalman unscented rimane il fatto che si basa sull'assunzione che la densità di probabilità a posteriori è ad ogni passo di tipo gaussiano. Un modo molto comune per superare tale difficoltà consiste nell'usare un'altra strategia basata sull'applicazione ricorsiva di metodi Monte Carlo, comunemente nota come *particle filter*, o *filtro a particelle*.

Questa nuova strategia fornisce una completa rappresentazione della densità di probabilità a posteriori dello stato, cosicché è possibile calcolare qualsiasi statistica desiderata. Il punto di forza di questo metodo consiste nella possibilità di affrontare sia problemi non-lineari che non gaussiani. Il filtro a particelle, pur essendo un altro tipo di algoritmo subottimo per il problema di stima, gioca un ruolo centrale in questo lavoro di tesi e pertanto il prossimo capitolo è completamente dedicato alla presentazione di tale filtro.

2. IL FILTRO A PARTICELLE

2.1 I metodi Monte Carlo per stime non-lineari non-gaussiane

I metodi Bayesiani forniscono una struttura generale rigorosa per problemi di stima dello stato in sistemi dinamici. L'approccio, come presentato nel precedente capitolo, consiste nel costruire la funzione densità di probabilità dello stato basandosi su tutte le informazioni disponibili; tale funzione racchiude tutta la conoscenza attuale sullo stato e tramite tale densità è possibile calcolare tutte le statistiche di interesse, quali ad esempio il valore medio, la moda o la varianza...

Nel caso di problemi di stima di sistemi lineari gaussiani la distribuzione a posteriori rimane gaussiana ad ogni iterazione del filtro, e le equazioni dello stimatore di Kalman permettono di propagare il valore medio e la varianza della distribuzione. Nel caso di sistemi dinamici non lineari e non gaussiani non esiste invece in generale una soluzione analitica in forma chiusa per la richiesta densità di probabilità; l'insieme di tali sistemi dinamici non è inoltre trascurabile, dato che la quasi totalità di applicazioni del mondo reale richiede l'utilizzo di modelli non lineari.

Ad esempio un semplice ricevitore GPS posizionato in un punto di coordinate $X_r = \begin{bmatrix} x_r & y_r & z_r \end{bmatrix}^T$ misura il tempo di trasmissione di un satellite posizionato in $X_s = \begin{bmatrix} x_s & y_s & z_s \end{bmatrix}^T$. Chiaramente il tempo di misurazione dovrà tenere conto in qualche modo della norma del vettore differenza $\|X_r - X_s\|$, e perciò richiederà l'uso di un modello di misurazione non lineare.

Analogamente in un problema di tracking di un veicolo le misurazioni riguardano generalmente il *line of sight angle*, ovvero l'angolo tra l'orizzontale e la linea che unisce il target e il sensore; se le coordinate del target sono espresse in un sistema cartesiano la funzione d'uscita richiede il calcolo di un arcotangente ed implica di nuovo l'uso di un modello non lineare.

La strategia che si sta imponendo sul mercato per affrontare sistemi non lineari consiste nello sfruttamento iterativo dei cosiddetti metodi Monte Carlo; tali metodi sfruttano generatori di numeri casuali per risolvere una vasta gamma di problemi matematici e fisici.

Il nome "Monte Carlo" fu scelto da Metropolis durante la seconda guerra mondiale nel Progetto Manhattan a causa della analogia tra le simulazioni statistiche e i giochi d'azzardo, di cui la città del principato di Monaco era la capitale. Dapprima i metodi Monte Carlo vennero

usati per eseguire simulazioni allo scopo di predire il tempo necessario per avere una collisione tra particelle all'interno di un materiale.

Le tecniche Monte Carlo giocano un ruolo importante anche nella stima dello stato in situazioni non lineari e non gaussiane in modo diretto, in quanto permettono di estrarre campioni da una densità di probabilità in base alla loro importanza; tale campionamento ad importanza, noto in letteratura come *importance sampling* è contrapposto ad un campionamento uniforme (*uniform sampling*) come mostrato nella Fig.2:

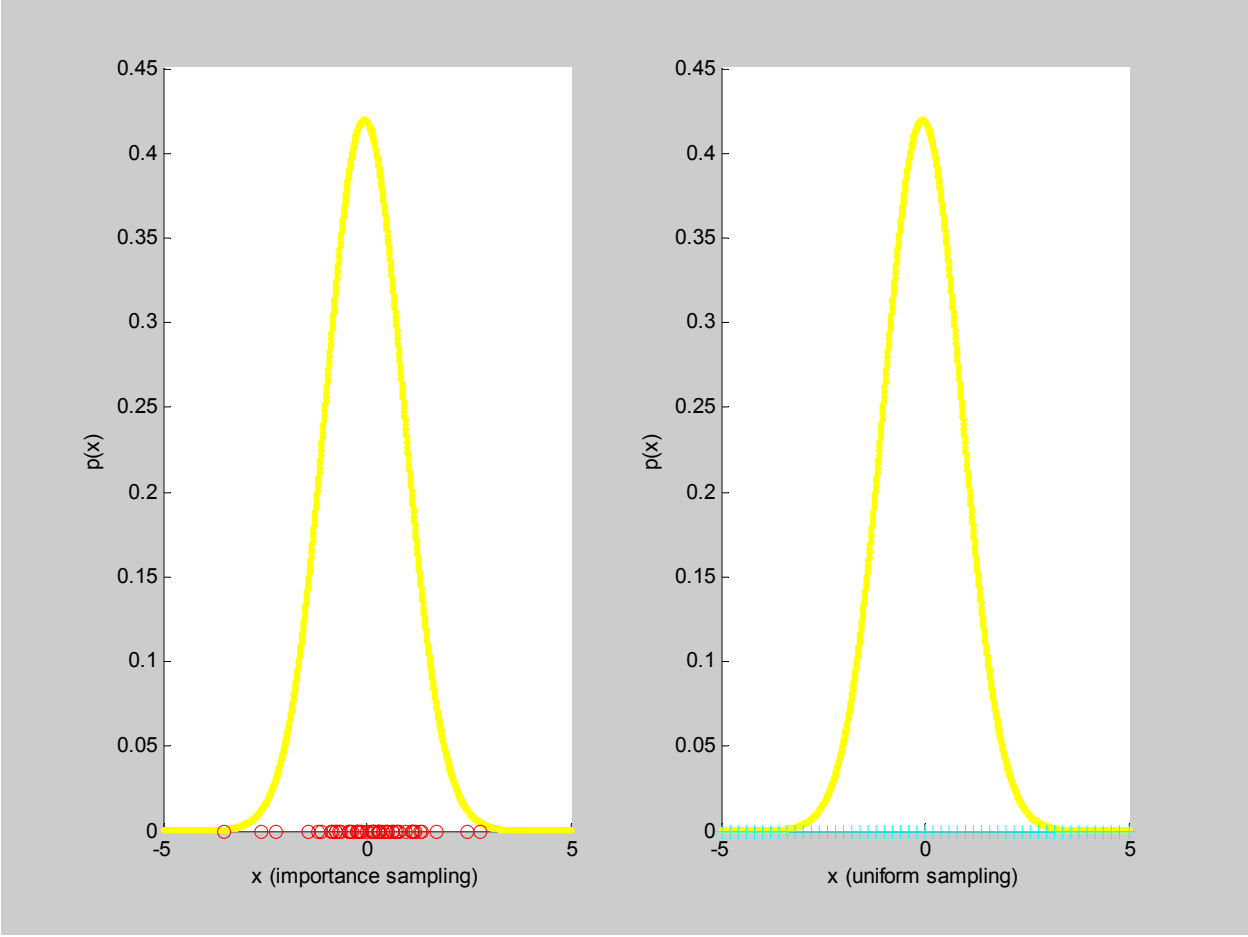

Fig. 2: Confronto tra un campionamento ad importanza ed un campionamento uniforme

Tale proprietà di campionamento gioca anche un ruolo indiretto nel problema di stima in quanto, in particolari condizioni, permette di mappare integrali in sommatorie, fornendo una buona soluzione approssimata all'approccio Bayesiano. L'uso iterativo di tecniche Monte Carlo fornisce la base per gli algoritmi noti come filtri a particelle, estensioni del filtro di Kalman nel caso di problemi non lineari e non gaussiani.

2.2 Applicazioni di filtri a particelle

Metodi basati su un uso sequenziale delle tecniche Monte Carlo e campionamento ad importanza erano già stati introdotti in letteratura negli anni ’50, soprattutto nel campo della fisica e della statistica. Queste tecniche erano state introdotte anche nel campo dei controlli automatici alla fine degli anni ’60, e negli anni ’70 molti ricercatori continuarono a lavorare su queste idee. Ad ogni modo i risultati non erano entusiasmanti dato che la semplice iterazione del campionamento ad importanza ha prestazioni che degenerano con il trascorrere del tempo. Il più grande contributo verso questa classe di algoritmi è stato quindi fornito all’inizio degli anni ’90 quando è stato introdotto un passo intermedio basato su un ricampionamento. Da allora sono stati forniti innumerevoli miglioramenti e modifiche, in modo da adattare i filtri a particelle alle situazioni più svariate. Alcuni esempi di applicazioni di filtri a particelle si trovano in campo economico per lo studio di modelli stocastici sulla volatilità delle azioni [14], in campo medico per lo studio delle attività neuronali [15], in campo aerospaziale per problemi di tracking [16], nelle multiconferenze per la fusione di suoni e immagini proveniente da più sensori [17], in problemi di navigazione sottomarina come in [18], nel campo della robotica per problemi SLAM (Simultaneous Localization And Mapping) come in [19].

Si può quindi notare come l’utilizzo di filtri a particelle sia molto popolare e, nonostante tali stimatori siano stati introdotti in tempi piuttosto recenti, si siano diffusi anche in campi molto diversi tra loro. La letteratura a riguardo è molto vivace ed esistono in letteratura molte varianti di particle filter, tra cui lo *unscented particle filter* come in [4]; esistono inoltre molti miglioramenti introdotti per applicazioni ad hoc, come passi intermedi opzionali MCMC (Monte Carlo Markov Chain) suggeriti sempre in [4], o casi in cui i filtri a particelle vengono usati insieme ad altri algoritmi di predizione, come RANSAC in [20]. Nel seguente paragrafo viene mostrato il funzionamento generale di un filtro a particelle e i passi che costituiscono l’algoritmo di base.

2.3 Funzionamento di un filtro a particelle

L’idea intuitiva alla base di un filtro a particelle consiste nell’implementare un algoritmo che ad ogni passo mantenga molte versioni del vettore di stato, tutte leggermente diverse tra loro; quando una nuova misurazione diventa disponibile viene assegnato un punteggio a ciascuna versione del vettore di stato in base a quanto bene è in grado di spiegare il dato. Le versioni del vettore di stato che sono risultate migliori vengono copiate e casualmente perturbate in modo da ottenere una nuova generazione di stati candidati. In nessun punto dell’algoritmo c’è bisogno di effettuare linearizzazioni o di fare assunzioni sulla gaussianità del rumore o della densità di

probabilità a posteriori. Alcuni problemi subito evidenti in questo approccio elegante consistono invece nel decidere di quante versioni del vettore di stato si ha effettivamente bisogno per ottenere stime adeguatamente accurate o in base a cosa assegnare un punteggio per giudicare la bontà di una versione dello stato.

Più formalmente si può affermare che il filtro a particelle sfrutta simulazioni Monte Carlo per campionare la densità di probabilità a posteriori dello stato ed ottenere un insieme di particelle pesate (le precedenti "versioni dello stato") che possano essere usate per trasformare integrali in sommatorie.

Analiticamente, se $g_k(x_{0:k})$ rappresenta una qualsiasi statistica relativa allo stato, ogni aspettazione della forma (30)

$$E\{g_k(x_{0:k})\} = \int g_t(x_{0:k}) p(x_{0:k}|y_{1:k}) dx_{0:k} \tag{30}$$

può essere approssimata dalla stima (31):

$$\overline{E(g_k(x_{0:k}))} = \frac{1}{N}\sum_{i=1}^{N} g_k(x_{0:k}^i) \tag{31},$$

dove le particelle $(x_{0:k}^i)$ si assumono indipendenti e identicamente distribuite.

La legge dei grandi numeri assicura la (32)

$$\overline{E\{g_k(x_{0:k})\}} \xrightarrow[N\to\infty]{} E\{g_k(x_{0:k})\} \tag{32}$$

dove la freccia denota convergenza quasi sicura. Inoltre se la varianza di $g_k(x_{0:k})$ è limitata, allora vale anche il seguente teorema del valore centrale:

$$\sqrt{N}\left(\overline{E\{g_k(x_{0:k})\}} - E\{g_k(x_{0:k})\}\right) \xrightarrow[N\to\infty]{} N\left(0, \mathrm{var}_{p(\cdot|y_{1:k})}(g_k(x_{0:k}))\right) \tag{33}$$

dove la freccia indica in questo caso convergenza in distribuzione.

Le equazioni (32) e (33) forniscono la giustificazione per poter mappare gli integrali che appaiono nelle operazioni di aspettazione in sommatorie, ammesso che il numero di particelle sia sufficientemente alto. Sfortunatamente è però spesso impossibile campionare direttamente la

funzione densità a posteriori, che rappresenta l'incognita dell'approccio Bayesiano di stima; pertanto è conveniente campionare un'altra distribuzione $q(x_{0:k}|y_{1:k})$ conosciuta e facile da campionare, nota in letteratura come *proposal distribution*, o distribuzione proposta. La scelta di tale distribuzione proposta discrimina spesso il tipo di filtro di particelle.

Sostituendo la nuova proposal nell'equazione (30) si ottiene:

$$E\{g_k(x_{0:k})\} = \int g_k(x_{0:k}) \frac{p(x_{0:k}|y_{1:k})}{q(x_{0:k}|y_{1:k})} q(x_{0:k}|y_{1:k}) dx_{0:k} =$$

$$= \int g_k(x_{0:k}) \frac{p(y_{1:k}|x_{0:k}) p(x_{0:k})}{p(y_{1:k}) q(x_{0:k}|y_{1:k})} q(x_{0:k}|y_{1:k}) dx_{0:k} = \quad (34)$$

$$= \int g_k(x_{0:k}) \frac{w_k}{p(y_{1:k})} q(x_{0:k}|y_{1:k}) dx_{0:k}$$

dove il secondo passaggio della (34) è ottenuto applicando il teorema di Bayes e i pesi w_k, calcolati come nella (35), sono noti come i *pesi di importanza non normalizzati*.

$$w_k = \frac{p(y_{1:k}|x_{0:k}) p(x_{0:k})}{q(x_{0:k}|y_{1:k})} \quad (35).$$

La densità normalizzante $p(y_{1:k})$ può essere eliminata come nella (36):

$$E\{g_k(x_{0:k})\} = \frac{1}{p(y_{1:k})} \int g_k(x_{0:k}) w_k q(x_{0:k}|y_{1:k}) dx_{0:k} = \frac{\int g_k(x_{0:k}) w_k q(x_{0:k}|y_{1:k}) dx_{0:k}}{\int p(y_{1:k}|x_{0:k}) p(x_{0:k}) \frac{q(x_{0:k}|y_{1:k})}{q(x_{0:k}|y_{1:k})} dx_{0:k}} =$$

$$= \frac{\int g_k(x_{0:k}) w_k q(x_{0:k}|y_{1:k}) dx_{0:k}}{\int w_k q(x_{0:k}|y_{1:k}) dx_{0:k}} = \frac{E_{q(\cdot|y_{1:k})}\{w_k g_k(x_{0:k})\}}{E_{q(\cdot|y_{1:k})}\{w_k\}} \quad (36),$$

dove la notazione $E_{q(\cdot|y_{1:k})}$ mette in evidenza come le aspettazioni siano in realtà calcolate rispetto alla distribuzione proposta $q(x_{0:k}|y_{1:k})$. Campionando perciò la funzione proposta si possono mappare gli integrali che compaiono nelle aspettazioni in sommatorie, come nella (37):

$$\overline{E\{g_k(x_{0:k})\}} = \frac{1/N\sum_{i=1}^{N} g_k(x_{0:k}^i) w_k^i}{1/N\sum_{i=1}^{N} w_k^i} = \sum_{1=1}^{N} g_k(x_{0:k}^i)\tilde{w}_k^i \quad (37),$$

dove i *pesi d'importanza normalizzati* $\tilde{w}_k^i$ valgono

$$\tilde{w}_t^i = \frac{w_k^i}{\sum_{j=1}^{N} w_k^j} \quad (38).$$

Se quindi il numero delle particelle tende ad ∞, la funzione densità a posteriori può arbitrariamente essere ben approssimata dalla stima (39)

$$\hat{p}(x_{0:k}|y_{1:k}) = \sum_{i=1}^{N} \tilde{w}_k^i \delta(x_k - x_k^i) \quad (39).$$

Le precedenti equazioni possono essere riscritte in modo da rendere iterativo l'intero algoritmo; in questo caso è ragionevole scegliere la funzione proposal come nella (40), in modo che lo stato non dipenda dalle osservazioni future e in modo che una nuova osservazione non influisca sulle stime precedenti;

$$q(x_{0:k}|y_{1:k}) = q(x_{0:k-1}|y_{1:k-1}) q(x_k|x_{0:k-1}, y_{1:k}) \quad (40).$$

Date le assunzioni di stati che evolvono come un processo di Markov del primo ordine e delle osservazioni indipendenti dati gli stati, si possono ottenere equazioni ricorsive per il calcolo dei pesi partendo dalla (35):

$$w_k = \frac{p(y_{1:k}|x_{0:k})p(x_{0:k})}{q(x_{0:k-1}|y_{1:k-1})q(x_k|x_{k-1},y_{1:k})} = w_{k-1}\frac{p(y_{1:k}|x_{0:k})p(x_{0:k})}{p(y_{1:k-1}|x_{0:k-1})p(x_{0:k-1})}\frac{1}{q(x_k|x_{0:k-1},y_{1:k})} =$$

$$= w_{k-1}\frac{p(y_k|x_k)p(x_k|x_{k-1})}{q(x_k|x_{0:k-1},y_{1:k})} \quad (41);$$

nel caso comune della stima ricorsiva in realtà si è più interessati alla sola densità filtrata più che alla densità a posteriori totale, in questo caso la (41) diventa la (41'):

$$w_k = w_{k-1}\frac{p(y_k|x_k)p(x_k|x_{k-1})}{q(x_k|x_{k-1},y_k)} \quad (41').$$

L'equazione (41') fornisce un algoritmo per aggiornare in modo sequenziale i pesi d'importanza, chiarendo il meccanismo in base al quale si sceglie il punteggio da assegnare a ciascuna versione dello stato in base alla propria "bontà". I pesi d'importanza sono però assegnati una volta nota la proposal distribution, la scelta della quale rappresenta spesso un passo critico. Una volta scelta tale funzione, l'algoritmo prevede di stabilire un insieme di particelle a priori e calcolare i pesi di importanza in modo iterativo campionando ad ogni passo la proposal. Questa procedura è nota tramite la sigla SIS, abbreviazione di *Sequential Importance Sampling*.

2.3.1 Scelta della proposal distribution

Uno dei passi critici nella progettazione di un filtro di particelle consiste quindi nella scelta di un'adeguata funzione proposta, possibilmente facile da ricavare e da campionare. Un primo metodo nella scelta di tale funzione consiste nel voler minimizzare la varianza dei pesi delle particelle. Se infatti le particelle fossero estratte dalla vera densità a posteriori i pesi sarebbero tutti uguali tra loro; peggiore è l'approssimazione maggiore diventa la varianza delle particelle.

E' stato dimostrato in [21] che la scelta di $q(x_k|x_{k-1},y_k) = p(x_k|x_{k-1},y_k)$ minimizza la varianza dei pesi d'importanza condizionati in $x_{0:k-1}$ e $y_{1:k}$;
in tale caso si ha che il calcolo iterativo dei pesi si trasforma nella (42):

$$w_k^i = w_{k-1}^i p(y_k|x_{k-1}^i) = w_{k-1}^i \int p(y_k|x_k)p(x_k|x_{k-1}^i)dx_k \quad (42);$$

tale scelta ottima ha però due svantaggi: richiede la capacità di campionare tale densità e la necessità di saperla integrare sul nuovo stato come nella (42). Come mostrato in [2] ci sono due casi in cui è possibile usare la densità d'importanza ottima: il primo è il caso in cui x_k appartiene ad un insieme finito di stati. In questa situazione l'integrale della (42) diventa una sommatoria; il secondo caso è quello in cui $p(x_k|x_{k-1}, y_k)$ è gaussiana, come accade nel caso in cui le dinamiche sono non lineari, le misurazioni lineari, i rumori additivi gaussiani bianchi a valor medio nullo.

Questi sistemi possono essere descritti da equazioni del tipo

$$\begin{cases} x_k = f_k(x_{k-1}) + v_{k-1} \\ y_k = H_k x_k + n_k \end{cases} \quad (43);$$

se Q_{k-1} e R_k sono le matrici di covarianza rispettivamente del rumore di processo e del rumore di misurazione si ha che definendo

$$\Sigma_k^{-1} = Q_{k-1}^{-1} + H_k^T R_k^{-1} H_k \quad (44)$$

e

$$m_k = \Sigma_k \left(Q_{k-1}^{-1} f_k(x_{k-1}) + H_k^T R_k^{-1} y_k \right) \quad (45),$$

si ottiene

$$p(x_k|x_{k-1}, y_k) = N(x_k; m_k, \Sigma_k) \quad (46)$$

e

$$p(y_k|x_{k-1}) = N(y_k; H_k f_k(x_{k-1}), Q_{k-1} + H_k R_k H_k^T) \quad (47),$$

dove il simbolo N indica una distribuzione normale, il primo argomento rappresenta l'incognita della distribuzione, il secondo il valor medio e il terzo la varianza.

In molti altri modelli non è possibile effettuare considerazioni analoghe ed usare quindi la densità proposal ottima. E' comunque possibile effettuare delle approssimazioni e ad esempio stimare la proposal ottima usando un filtro di Kalman esteso o un filtro unscented di Kalman, supponendo quindi che questa abbia una forma gaussiana. In particolare, un filtro a particelle che usa come densità proposta la $p\left(x_k \middle| x_{k-1}, y_k\right)$ fornita da un UKF ha prestazioni molto buone e nonostante sia computazionalmente più pesante può dare il vantaggio di richiedere un minor numero di particelle. Un filtro di tale genere è noto come Unscented Particle Filter [4].

Un filtro a particelle standard usa invece un'altra approssimazione della proposal distribution ottima, ovvero sceglie $q\left(x_k \middle| x_{k-1}^i, y_k\right) = p\left(x_k \middle| x_{k-1}^i\right)$. In questo caso i pesi d'importanza valgono semplicemente

$$w_k^i = w_{k-1}^i p\left(y_k \middle| x_k^i\right) \tag{48}.$$

Tale scelta è molto semplice e anche intuitiva, dato che il punteggio di ogni versione dello stato viene assegnato semplicemente in base alla likelihood, cioè a quanto la misura si avvicina alla predizione. Lo svantaggio di tale scelta è però che nella proposal non compaiono le nuove informazioni, in particolare l'ultima osservazione. A causa di ciò la densità di likelihood potrebbe venirsi a trovare nella coda della funzione prior, lontano dalla nuvola di particelle che inevitabilmente si vede quindi assegnare pesi bassi, come mostrato nella Fig.3. Ciò accade in modo particolare quando la funzione di likelihood è molto più stretta di quanto non lo sia la prior; tale situazione è piuttosto frequente dato che i sensori di misura sono in genere molto precisi, mentre tutte le approssimazioni fatte per ricavare un modello del processo causano un rumore di processo maggiore di quello di osservazione.

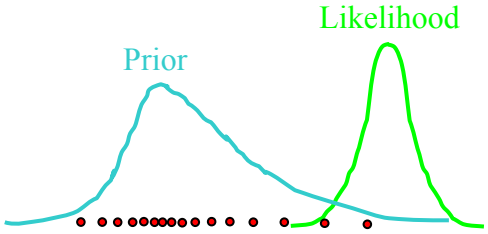

Fig. 3: Esempio di likelihood stretta posta nella coda della densità prior

Uno dei vantaggi della proposal ottima consiste proprio nello spostare le particelle della prior nelle regioni ad alta likelihood, in modo che queste vengano ad avere pesi d'importanza maggiori.

Esistono comunque delle tecniche che, nel caso in cui le funzioni di likelihood e prior sono molto lontane tra loro, introducono delle distribuzioni intermedie tra le due. Le particelle vengono quindi ricampionate sulla base di tali nuove distribuzioni intermedie [22].

2.3.2 Problema del ricampionamento

L'algoritmo Sequential Importance Sampling fin qui presentato ha la grande limitazione per cui dopo poche iterazioni può accadere che tutte le particelle tranne una hanno dei pesi trascurabili. Questo fenomeno di degenerazione è inevitabile dato che è stato dimostrato che la varianza dei pesi non può che aumentare con il trascorrere del tempo [23]. Un misuratore adatto di tale degenerazione dell'algoritmo è il fattore N_{eff}, misurato come:

$$N_{eff} = \frac{N_s}{1 + \mathrm{var}\left(w_k^{*i}\right)} \tag{49},$$

dove

$$w_k^{*i} = \frac{p\left(x_k^i \middle| y_{1:k}\right)}{q\left(x_k^i \middle| x_{k-1}^i, y_k\right)} \tag{50}$$

è noto come *vero peso d'importanza* e N_s è il numero di campioni. Dato che questo non può essere valutato in maniera esatta, una stima $\overline{N_{eff}}$ di N_{eff} può essere ottenuta come nella (51):

$$\overline{N_{eff}} = \frac{1}{\sum_{i=1}^{N_s}\left(w_k^i\right)^2} \tag{51},$$

dove il singolo peso è calcolato ad esempio come nella (48). Come già affermato nel paragrafo 2.3.1, la varianza ideale dei pesi vale 0, e si riferisce al caso in cui le particelle fossero estratte

direttamente dalla funzione a posteriori reale. Pertanto valori molto piccoli di $\overline{N_{eff}}$ sono relativi alle situazioni in cui l'algoritmo sta degenerando. In queste situazioni accade che un numero sempre maggiore di particelle ha un peso così piccolo da essere trascurabile nelle iterazioni successive.

Un modo brutale per ridurre l'effetto di degenerazione consiste nell'usare un numero maggiore di particelle; una soluzione più furba consiste invece nell'effettuare un ricampionamento in modo da avere sempre pesi consistenti per tutte le particelle.

Nel particle filter standard si effettua un passo di ricampionamento ad ogni iterazione, mentre in altre applicazioni si può decidere di effettuare un ricampionamento ad hoc solo in alcune situazioni particolari, ad esempio quando il valore di N_{eff} scende sotto un valore di soglia predefinito; ad esempio nell'applicazione in [19] il valore di soglia è stato posto pari a $N/2$, dove N rappresentava il numero di particelle del filtro.

In letteratura esistono vari metodi per effettuare il ricampionamento, tra questi vengono riportati successivamente il *sampling-importance resampling*, il *residual resampling* e il *minimum variance resampling*.

Gli algoritmi di ricampionamento in generale hanno il compito di eliminare i campioni con bassi pesi di importanza e aumentare le particelle con alti pesi; ciascuna particella $x_{0:k}^{i}$, dopo il ricampionamento, avrà un numero naturale N_i di particelle uguali a se stessa; se la particella aveva una bassa importanza N_i potrà essere 0, se la sua importanza era alta N_i sarà un numero alto. Alla fine $\sum_{i=1}^{N} N_i = N$, dato che il numero delle particelle deve rimanere lo stesso. Il resampling può essere visto come un passo al termine del quale ciascuna particelle ha un numero di figli proporzionale al suo peso, in modo analogo a quanto avviene per gli algoritmi genetici.

2.3.2.1 Sampling-importance resampling

Il metodo di ricampionamento più immediato e più noto è il Sampling-Importance Resampling, noto in letteratura tramite l'abbreviazione SIR. Al termine di tale ricampionamento ciascuna particella del tipo $\left\{x_{1:k}^{i}, \tilde{w}_k^i\right\}$ viene mappata in una particella $\left\{x_{1:k}^{j}, N^{-1}\right\}$, cioè tutte le particelle vengono ad assumere lo stesso peso, La scelta delle nuove particelle viene effettuata tramite un generatore di campioni uniformi che estrae un numero tra 0 e 1; la particella che corrisponde a tale valore nella funzione di distribuzione cumulativa (*cdf*) diventa una particella dell'insieme ricampionato. La Fig. 4 mostra più chiaramente come avviene tale procedura: a destra un numero *i* viene estratto da una distribuzione uniforme tra 0 e 1 e viene proiettato nella cdf a sinistra. La

particella *j* che si trova in corrispondenza di tale valore viene propagata come una delle versioni dello stato in quel passo. Questa procedura può essere interpretata semplicemente con lo scegliere un campione nell'insieme $\left\{x_{0:k}^{i}, i = 1,..., N\right\}$ con probabilità $\tilde{w}_k^i, i = 1,..., N$. Infatti le particelle che hanno un basso peso hanno bassa probabilità di essere propagate, mentre quelle con peso maggiore verranno probabilmente replicate più volte.

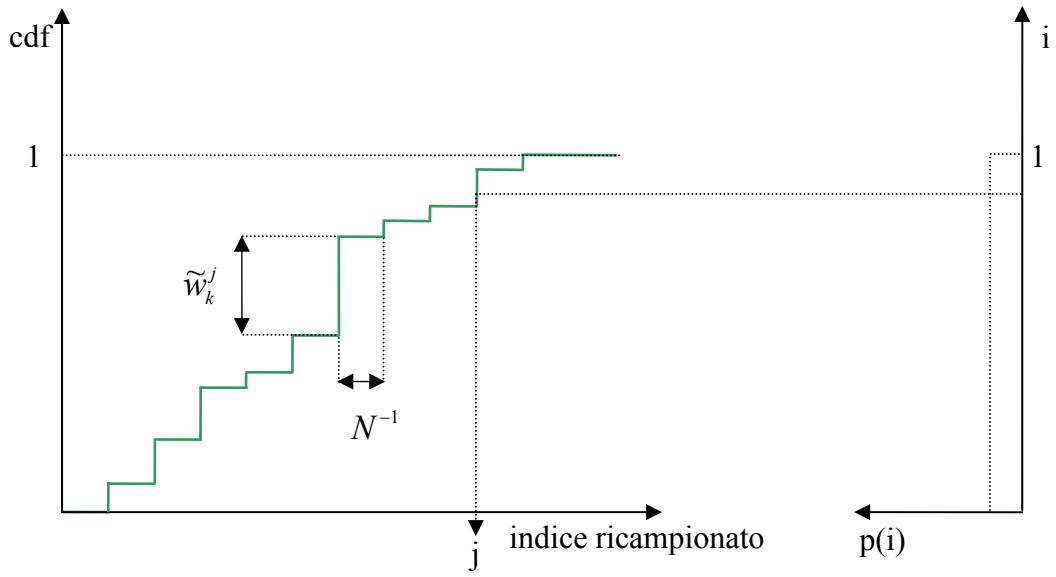

Fig. 4: Esempio di sampling-importance resampling

Un algoritmo che compie questo tipo di ricampionamento può essere implementato con complessità O(N). Un'implementazione in pseudo codice di tale schema di ricampionamento può essere la seguente:

%% Inizializzazione della CDF: $c_1 = 0$ *%%*
for $i = 2 : N_s$
    Costruire la CDF: $c_i = c_{i-1} + w_k^i$
end-for
%% Estrarre N_s *campioni* u_i $\left(1 < i \le N_s\right)$ *da una distribuzione uniforme* $U[0,1]$ *%%*

for $i = 1 : N_s$
    j=0
    while $u_i > CDF\left(x_j\right)$ *do*
        j++
    end while

%% Aggiungi x_j alla lista di particelle ricampionate %%

end-for

Il precedente algoritmo rappresenta il modo più intuitivo di effettuare il ricampionamento ad importanza, anche se in realtà ha complessità $O(N \log N)$; è possibile comunque ottimizzare l'algoritmo per farne diminuire la complessità ad $O(N)$, scalando opportunamente tutti i campioni u_i estratti dalla distribuzione uniforme. Un esempio di uso di questa procedura si può trovare in [20].

2.3.2.2 Residual resampling

In tale metodo il primo passo consiste nello scegliere $\widetilde{N}_i = \lfloor N_s \widetilde{w}_k^i \rfloor$, dove l'operatore floor $\lfloor \ \rfloor$ arrotonda per difetto all'intero più vicino. I rimanenti $\overline{N}_k = N_s - \sum_{i=1}^{N_s} \widetilde{N}_i$ campioni con i nuovi pesi $w_k^{'i} = \overline{N}_k^{-1}\left(\widetilde{w}_k^i N_s - \widetilde{N}_i\right)$ vengono scelti con una procedura SIR. Infine si mettono insieme i campioni ottenuti in ciascuno dei due passi; l'algoritmo così trovato ha ancora complessità $O(N)$, garantisce una varianza minore rispetto al sequential-importance resampling ed è computazionalmente più efficiente.

2.3.2.3 Systematic resampling

Tale algoritmo di ricampionamento è molto usato e consigliato in letteratura in quanto garantisce una varianza ancora minore rispetto al residual resampling [2]. In questo caso vengono estratti N_s campioni nell'intervallo $[0,1]$, a distanza N_s^{-1} l'uno dall'altro. Il numero dei figli N_i viene ora scelto come il numero di punti che sono tra $\sum_{j=1}^{i-1} \widetilde{w}_k^j$ e $\sum_{j=1}^{i} \widetilde{w}_k^j$. La complessità dell'algoritmo è ancora $O(N)$ e una implementazione in uno pseudo-codice può essere la seguente:

%% Inizializzazione della CDF: $c_1 = 0$ %%

for $i = 2 : N_s$

    Costruire la CDF: $c_i = c_{i-1} + w_k^i$

end-for

%% Iniziando dall'inizio della CDF %% $i = 1$

%% Estrarre un punto di partenza da una distribuzione uniforme $U\left[0, N_s^{-1}\right]$ *%%*

for $j = 1 : N_s$

    $u_j = u_1 + N_s^{-1}(j-1)$

while $(u_j > c_i)$

$i = i + 1$

end-while

%%Propagare il campione $x_k^j = x_k^i$ *%%*

%% Assegnargli peso comune $w_k^j = N_s^{-1}$ *%%*

end-for

2.3.3 Algoritmo di un filtro a particelle generale

L'implementazione ricorsiva dei passi illustrati nei paragrafi precedenti costituisce la base dell'algoritmo del filtro a particelle. Successivamente sono quindi riassunti tutti i passaggi di un particle filter standard in cui la densità di probabilità proposal è rappresentata semplicemente dalla densità prior; in questa situazione quindi i valore dei pesi da assegnare alle versioni del vettore di stato sono determinati dalla densità di likelihood.

1) Inizializzazione del filtro, $k = 0$;

   - estrarre N_s campioni dalla funzione prior $p(x_0)$ per avere le versioni dello stato iniziale x_0^i. Nel caso in cui si abbia perfetta conoscenza dello stato iniziale si può assegnare a tutte le particelle tale valore; un caso tipico è invece quello in cui la probabilità dello stato iniziale ha distribuzione uniforme tra un valore minimo e uno massimo.
   - porre tutti i pesi $w_k^o = 1$

2) Per $k = 1,2,...$

   a) Passo di campionamento ad importanza

      - estrarre N_s particelle $\hat{x}_k^i$ dalla distribuzione $q\left(x_k \middle| x_{k-1}^i, y_{1:k}\right)$
      - valutare i pesi a meno di una costante di normalizzazione come nella (48):

        $w_k^i = w_{k-1}^i p\left(y_k \middle| \hat{x}_k^i\right)$
      - normalizzare i pesi d'importanza come nella (38):

$$\tilde{w}_k^i = w_k^i \left[\sum_{j=1}^{N_s} w_k^j \right]^{-1}$$

b) Passo di selezione (resampling)

- replicare/sopprimere i campioni $\hat{x}_k^i$ con alti/bassi pesi d'importanza $\tilde{w}_k^i$ in modo da ottenere N_s campioni x_k^i disposti approssimativamente come la densità a posteriori $p\left(x_k^i \middle| y_{1:k}\right)$
- porre tutti i N_s pesi $w_k^i = N_s^{-1}$

c) Uscita dell'algoritmo: l'uscita dell'algoritmo sono un insieme di campioni che possono essere usati per approssimare la densità di probabilità a posteriori come nella (39):

$$p\left(x_k \middle| y_{1:k}\right) = \sum_{i=1}^{N} w_k^i \delta\left(x_k - x_k^i\right).$$

2.4 Esempio per un confronto tra le prestazioni di un particle filter, un filtro di Kalman esteso e un filtro di Kalman unscented

Il seguente esempio per confrontare le prestazioni del particle filter con quelle di un EKF e di un UKF è fornito in [4]. Il modello dell'evoluzione dello stato e delle osservazioni è dato dalla (52) e dalla (53):

$$x_{k+1} = 1 + \sin\left(\pi\omega k\right) + \phi_1 x_k + v_k \tag{52}$$

$$y_k = \begin{cases} \phi_2 x_k^2 + v_k & t \le 30 \\ \phi_3 x_k - 2 + n_k & t > 30 \end{cases} \tag{53},$$

dove v_k è una variabile aleatoria Gamma $Ga(3,2)$ che modella il rumore del processo, $\omega = 0.04$, $\phi_1 = \phi_3 = 0.5$, $\phi_2 = 0.2$, n_k è il rumore di osservazione modellato con una distribuzione gaussiana a valor medio nullo e varianza 10^{-3}. Il particle filter usato è di tipo

generale, quindi usa come proposal distribution la prior density, l'algoritmo di ricampionamento usato è il residual resampling e il numero di particelle è 200. Il filtro di Kalman unscented ha usato invece come parametri per la generazione deterministica dei sigma points $\alpha = 1$, $\beta = 0$ e $k = 2$. Ciascuna simulazione effettuata in ambiente Matlab ha avuto una durata di 60 passi, dove la stima dello stato ad ogni passo è ottenuta calcolando il valor medio della stima della distribuzione a posteriori dello stato, e la realizzazione di un singolo esperimento viene mostrata nella seguente Fig.5.

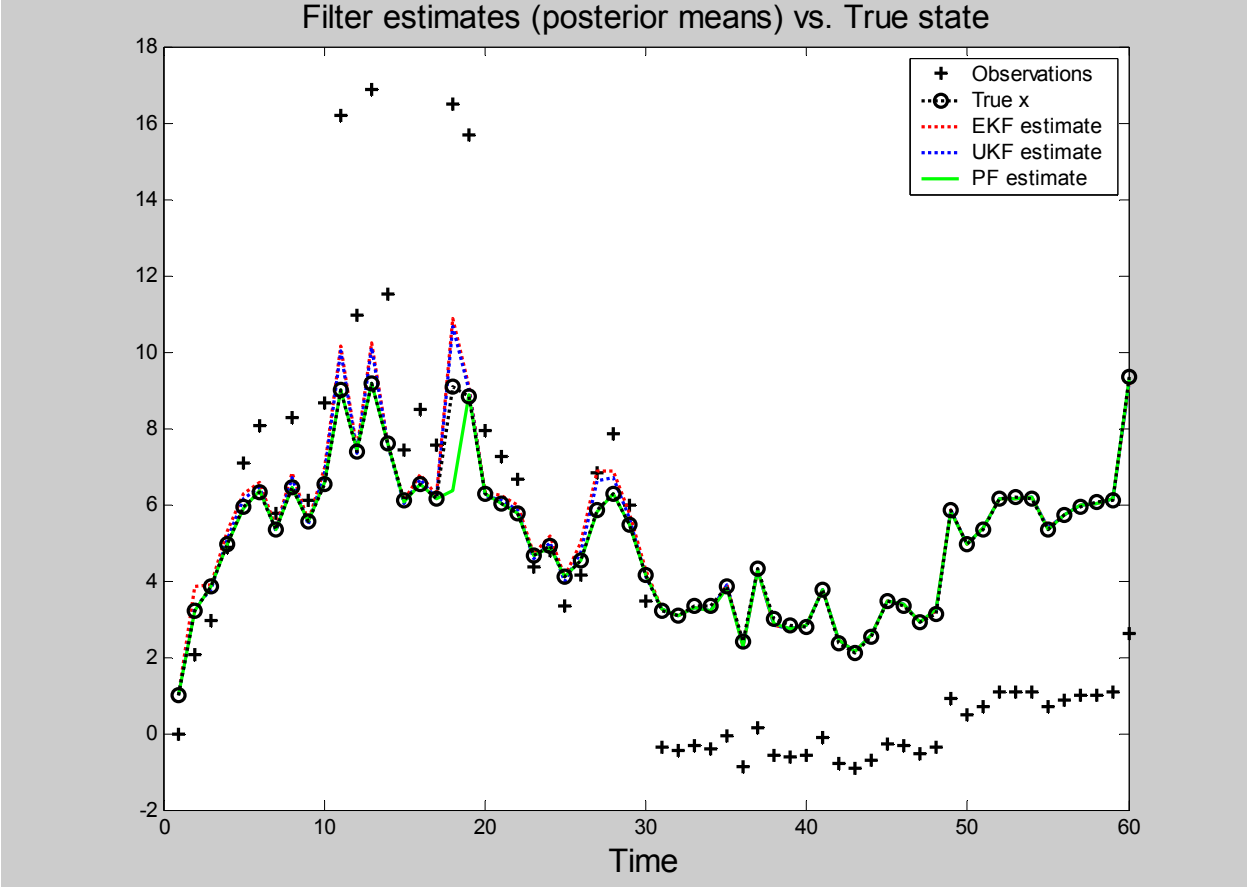

Fig. 5: Confronto tra le prestazioni di tre diversi filtri in un problema non lineare non gaussiano

Alcune rilevazioni statistiche sono state effettuate sulla base di 1000 esperimenti. I risultati sono stati:

********** *FINAL RESULTS* **********

Root Mean Square Errors (Variance)

EKF = 0.3822 (0.019033)
UKF = 0.28568 (0.014059)
PF = 0.15206 (0.038214)

Errori percentuali medi commessi

EKF = 3.297%
UKF = 2.4142%
PF = 1.0113%

Nell'esempio mostrato le prestazioni del filtro a particelle sono nettamente migliori di quelle del filtro di Kalman esteso e del filtro di Kalman unscented; gli errori mediamente commessi dal particle filter sono infatti circa la metà di quelli commessi dall'UKF e un terzo di quelli commessi dall'EKF. Nella seconda parte della simulazione, in cui la funzione delle osservazioni è diversa rispetto ai primi trenta secondi, le stime effettuate da tutti e tre i filtri sono generalmente sempre vicine ai valori veri.

Nel prossimo capitolo viene illustrato in dettaglio il codice Matlab usato per implementare un filtro a particelle e come è possibile modificare facilmente alcuni parametri ed alcune funzioni per poterlo usare in problemi diversi. Inoltre viene mostrato il comportamento del filtro per lo stesso problema presentato in questo paragrafo e in due altre situazioni. Nel capitolo 4 infine vengono presentati alcuni miglioramenti che possono essere introdotti per aumentare le prestazioni del filtro e sono mostrati i risultati di molte simulazioni effettuate a supporto delle previsioni teoriche.

3. TOOL-BOX PER FILTRO A PARTICELLE

3.1 Parametri del filtro

Nell'ambito di tale lavoro di tesi è stato implementato un filtro a particelle in ambiente Matlab tale da poter essere usato in ogni problema di stima non lineare; la generalità delle applicazioni di tale filtro è così vasta da richiedere la possibilità di specificare un gran numero di parametri in modo da poter adattare lo stimatore alle necessità particolari dell'utente. Alcune parti del codice Matlab per l'implementazione del filtro sono state riprese da [4] e [24].

Supponendo di avere un sistema descritto da (1) e (2), il primo passo per l'utilizzo del "particle filter tool-box" consiste nello scrivere i modelli per il calcolo dell'uscita e per l'aggiornamento dello stato rispettivamente nelle funzioni *hfun* e *ffun*. Una volta fatto ciò ogni altra modifica al filtro può essere apportata agendo semplicemente sulla funzione *ParticleFilter*;

All'inizio della funzione ci sono una serie di parametri che possono essere modificati per inizializzare il filtro, tra questi:

- *no_of_runs*: rappresenta il numero di simulazioni che si vogliono effettuare per poter generare statistiche più affidabili
- *disegna*: se questa variabile vale 1 la densità di probabilità a posteriori dello stato viene riportata in forma analitica e disegnata al termine di ogni iterazione multipla di 10. Il passaggio dai campioni distribuiti come l'approssimazione della densità di probabilità a posteriori alla funzione analitica della approssimazione di tale densità viene effettuato mediante stima a massima entropia. Nei paragrafi successivi verrà illustrato maggiormente in dettaglio tale passaggio
- *TipoRumoreProc*: in un filtro a particelle il rumore del processo può in generale avere una qualsiasi distribuzione. In questo tool-box è prevista la possibilità che tale rumore abbia distribuzione gaussiana, in questo caso tale variabile deve essere inizializzata ad 1, o che sia una variabile aleatoria di tipo Gamma nel caso di spazio degli stati monodimensionale; nella seconda circostanza tale variabile va inizializzata a 2
- *ParRumProc1*: questo parametro rappresenta la matrice valor medio del rumore nel caso di rumore gaussiano, il primo parametro nel caso di distribuzione di tipo gamma del rumore di processo
- *ParRumProc2*: questo parametro rappresenta la matrice di covarianza del rumore nel caso di rumore gaussiano o il secondo parametro nel caso di distribuzione di tipo gamma del rumore di processo

- *TipoRumoreMis:* come nel caso del rumore di processo anche il rumore del sensore nel modello di osservazione può essere o di tipo gaussiano, nel qual caso tale variabile deve essere inizializzata ad 1, o una variabile aleatoria di tipo Gamma, e quindi in questo caso deve valere 2
- *ParRumMis1:* nel caso in cui il precedente parametro valga 1, questa variabile rappresenta la matrice valor medio del rumore, altrimenti rappresenta il primo parametro di una distribuzione di tipo gamma
- *ParRumMis2:* in caso di rumore gaussiano questa variabile è la matrice di covarianza del rumore, nel caso in cui il rumore di misurazione è una variabile di tipo gamma rappresenta il secondo parametro di tale distribuzione
- *T*: rappresenta il numero di passi che costituisce una singola simulazione. Nell'esempio del paragrafo 2.4 il sistema era stato lasciato evolvere per 60 passi
- *N*: tale variabile va inizializzata con il numero di particelle che vogliono essere usate all'interno del filtro nell'algoritmo di stima. Maggiore è tale valore maggiore è l'accuratezza della stima, ma ogni iterazione dell'algoritmo diventa più lenta; nel caso di un sistema con un solo stato valori tipici per il numero di particelle sono compresi tra 50 e 200.
- *resamplingScheme*: tale variabile può assumere valore 1, 2 o 3. A seconda del valore assunto cambia l'algoritmo da usare nella fase di ricampionamento, come visto nel paragrafo 2.3.2. Gli algoritmi di resampling a disposizione sono gli stessi presentati in tale paragrafo, in particolare se la variabile resamplingScheme vale 1, l'algoritmo di resampling è il residual resampling; se vale 2 è il systematic resampling, se vale 3 è il multinomial resampling (altro modo per indicare il sampling-importance resampling)
- *m*: tale valore indica il numero delle uscite del sistema
- *StatoIniziale*: come indica il nome stesso della variabile questo parametro contiene il vettore iniziale del sistema da inserire come vettore riga.

I parametri sin qui indicati sono necessari per poter far funzionare il filtro e, se non vengono modificati, mantengono i loro valori di default, ovvero i valori assunti nel caso dell'esempio del paragrafo 2.4. Altre parti del filtro, come evidenziato nel seguente elenco, possono comunque essere modificate in modo da poterlo adattare alla particolare situazione d'interesse:

- al termine dell'esecuzione del programma la prima figura rappresenta l'andamento nel tempo dello stato reale e delle osservazioni. Nel caso in cui il vettore di stato abbia dimensione maggiore di uno viene plottato solamente il primo stato; ad esempio in un problema di tracking il primo stato rappresenta solitamente la posizione del target, mentre il secondo la velocità [1]. Di default viene quindi rappresentata solo la posizione reale del target a ciascun passo dell'evoluzione del sistema. E' possibile ovviamente modificare ciò decidendo di plottare solo il secondo stato o entrambi nella stessa figura o in figure diverse
- un parametro molto importante da modificare è rappresentato dallo stato iniziale delle particelle del filtro. Tale variabile rappresenta infatti la conoscenza sullo stato iniziale del processo, e di default viene posta uguale allo stato iniziale del sistema; spesso capita però che le condizioni iniziali del sistema sono note non direttamente per mezzo di x_0, ma sotto forma di densità di probabilità iniziale dello stato $p(x_0)$. In queste situazioni va quindi cambiato il valore iniziale assunto dalle particelle del filtro
- per evitare di appesantire eccessivamente il programma la densità di probabilità a posteriori non viene disegnata ad ogni passo; questo sia perché la rappresentazione di tale funzione richiede un tempo computazionale molto alto, sia per evitare di dover mostrare un numero troppo grande di disegni, specialmente quando i passi di evoluzione del sistema sono molti. Perciò, quando la variabile *disegna* viene posta a 1, solamente ogni 10 passi viene disegnata $p(x_k|y_{1:k})$; ovviamente è possibile modificare questo parametro in modo da disegnare la funzione più spesso o più raramente a seconda della durata della simulazione
- la funzione analitica della densità di probabilità a posteriori viene trovata con una stima a massima entropia basata sui primi tre momenti delle particelle dello stato; questo passaggio verrà spiegato in dettaglio nei prossimi paragrafi, è comunque possibile usare solo due momenti, ricostruendo quindi una densità gaussiana ad ogni passo, o un numero maggiore di momenti
- i disegni e le statistiche finali sono di nuovo relativi esclusivamente al primo stato del sistema; nell'esempio di tracking compaiono quindi le differenze tra la posizione stimata e la posizione reale del target; nel caso si fosse interessati agli errori nella stima della velocità o in generale agli errori di stima in altri stati del processo è possibile intervenire nel codice Matlab per modificare le impostazioni.

La funzione da modificare *ParticleFilter* è ricca di commenti in modo da aiutare l'utente a capire in quale punto intervenire se si desidera modificare uno dei parametri precedentemente elencati; sono inoltre specificate nel dettaglio la varie fasi dell'algoritmo: l'*inizializzazione*, la *predizione*, la *correzione* basata sul calcolo dei pesi tramite la funzione di *likelihood*, il *ricampionamento* e la parte finale in cui avviene il *calcolo delle statistiche*.

Al termine di ciascun passo dell'algoritmo di stima si ottiene come noto un insieme di particelle disposte approssimativamente come la densità di probabilità a posteriori dello stato, in modo tale che, supponendo di usare un numero adeguato di campioni, le statistiche relative a tale funzione siano calcolabili per mezzo delle particelle stesse, sotto forma di sommatorie. Ci sono delle situazioni in cui può risultare comodo conoscere un'approssimazione di tale funzione di probabilità a posteriori non solo per mezzo di particelle, ma anche in forma analitica; ciò permette ad esempio di avere la stima della probabilità di avere un dato stato ad un dato passo nonostante questo stato non abbia una particella che lo identifichi nella distribuzione a posteriori.

Inoltre, per mezzo della funzione analitica è anche possibile disegnare l'approssimazione della densità di probabilità a posteriori dello stato in modo da rendere più immediata l'interpretazione dell'uscita del filtro. Ad esempio la larghezza della densità di probabilità può dare indicazioni sulla affidabilità della previsione a quel passo.

In letteratura non sono stati incontrati filtri che prevedono un ultimo passaggio dedicato alla ricostruzione analitica della densità di probabilità a posteriori, probabilmente a causa del fatto che uno dei punti di forza del filtro di particelle risiede nella velocità delle sue prestazioni che lo rendono particolarmente efficace nelle situazioni in cui si richiedono stime on-line. La perdita di tempo a ciascun passo per ricostruire la funzione densità di probabilità dello stato senza dubbio ne limita le potenzialità; tuttavia la possibilità di avere a disposizione in un algoritmo di stima la funzione analitica della densità a posteriori è senza dubbio un vantaggio e il suo calcolo può essere effettuato anche ogni 10 o 100 passi. Inoltre tale calcolo, se effettuato con il metodo a massima entropia come implementato nel *Particle Filter Tool-box*, può essere effettuato anche off-line, dato che richiede la sola conoscenza delle particelle di stato ad un dato istante; uno dei punti di forza di un approccio di stima basato sull'entropia risiede infatti proprio nel fornire una distribuzione di probabilità "onesta" in quanto non richiede nessuna informazione che non sia chiaramente a propria disposizione [25].

3.2 Ricostruzione a massima entropia della funzione di densità di probabilità a posteriori

3.2.1 Concetto di entropia nel formalismo di Shannon

Shannon introdusse nel 1948 una misura per il calcolo della "quantità di incertezza", o "quantità di caos" o "mancanza di informazione", nota come *entropia di Shannon* o *entropia d'informazione*. Il legame tra incertezza e informazione può essere stabilito notando come l'incertezza in una data situazione dipenda dalla mancanza di informazioni a riguardo. La teoria dell'informazione è un campo molto vicino al problema dell'approccio Bayesiano, dato che in ogni sistema stocastico descritto dalla (1) e dalla (2) le osservazioni possono giusto essere usate per fornire informazioni sullo stato. L'entropia di Shannon può servire proprio per sapere quanta informazione può essere dedotta sullo stato ignoto X per mezzo di misurazioni di un'altra quantità Y.

Shannon cercava una misura d'incertezza H di una distribuzione di probabilità discreta ($p(x = x_1) = p_1, ..., p(x = x_n) = p_n$) che avesse le seguenti caratteristiche:

- H doveva essere continua rispetto ai p_i
- Se tutti i p_i erano uguali, allora H doveva essere funzione monotona crescente del numero di campioni n. Quando infatti tutti gli eventi sono equiprobabili l'incertezza aumenta con l'aumentare degli eventi possibili
- Se una scelta si divideva in due scelte consecutive, l'H originale doveva essere la somma pesata di ciascuno dei valori individuali di H. La terza proprietà può essere illustrata in modo più chiaro per mezzo della Fig.6.

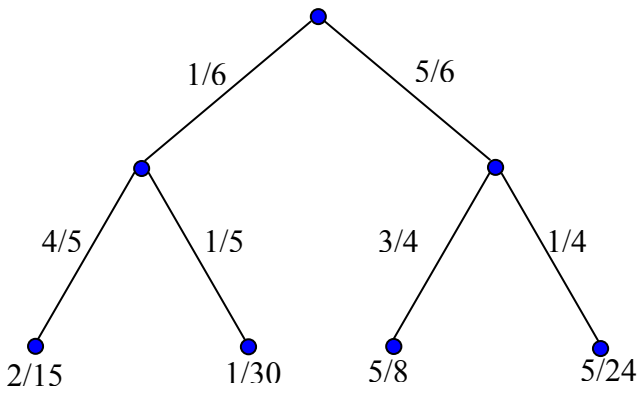

Fig. 6: Calcolo dell'entropia in caso di scelte multiple

Nella Fig.6 i valori che compaiono sui rami indicano la probabilità di passare da un nodo al successivo, mentre le probabilità sui nodi all'ultimo livello indicano la probabilità di giungere dal nodo iniziale a quello finale. La terza proprietà indica che la misura d'incertezza H deve garantire la seguente uguaglianza:

$$H\left(\frac{2}{15},\frac{1}{30},\frac{5}{8},\frac{5}{24}\right)=H\left(\frac{1}{6},\frac{5}{6}\right)+\frac{1}{6}H\left(\frac{4}{5},\frac{1}{5}\right)+\frac{5}{6}H\left(\frac{3}{4},\frac{1}{4}\right) \tag{54}.$$

Per poter rispettare le tre proprietà sopra descritte, Shannon definì l'entropia nel caso discreto come nella (55):

$$H(x)=-\sum_{i=1}^{N}p_i\log p_i \tag{55},$$

dove la scelta della base del logaritmo è arbitraria, anche se solitamente si usa la base 2, nel qual caso l'unità di entropia assume il nome di *bit*. Il segno meno di fronte alla sommatoria permette all'entropia di assumere valori non negativi.

Analogamente è possibile definire l'entropia congiunta come l'entropia della distribuzione congiunta come nella (56):

$$H(x,y)=-\sum_{x=1}^{N}\sum_{y=1}^{M}p(x,y)\log p(x,y) \tag{56};$$

l'entropia condizionale può ancora essere definita intuitivamente come nella (57)

$$H(x|y)=-\sum_{x=1}^{N}p(x|y)\log p(x|y) \tag{57},$$

ma così il suo valore dipenderebbe dalla variabile aleatoria y, quindi sarebbe di per sé una variabile aleatoria; in letteratura ci si riferisce quindi spesso all'entropia condizionale come all'entropia condizionale media calcolata come nella (58), [10]:

$$H(x|y)=\overline{H(x|y)}=-\sum_{x}\sum_{y}p(x,y)\log p(x|y)=-\sum_{x}\sum_{y}p(x,y)\log\frac{p(x,y)}{p(y)} \tag{58}.$$

Un concetto che torna molto utile in un approccio Bayesiano di stima è quello di mutua informazione, legato alla quantità di informazione che ci si aspetta di ottenere riguardo a X osservando Y. Supponendo infatti di sapere che $Y = y_i$ e che la variabile aleatoria Y è correlata ad X, allora l'incertezza su X si riduce all'entropia condizionata $H(X|y_i)$; si può quindi pensare che la quantità d'informazione ottenuta su X contenuta nella osservazione $Y = y_i$ valga $H(X) - H(X|Y = y_i)$.

Una quantità generalmente più interessante si ottiene passando ai valori medi, calcolando quindi la quantità d'informazione su X che ci si aspetta di ottenere osservando Y, giungendo quindi nella (59) alla definizione di *mutua informazione o mutua trasformazione* tra le variabili aleatorie X e Y:

$$I(X,Y) = H(X) - \overline{H(X|Y)} \tag{59}.$$

Il legame tra la mutua informazione, le densità condizionate (medie) e le densità marginali può essere schematizzato nella Fig. 7.

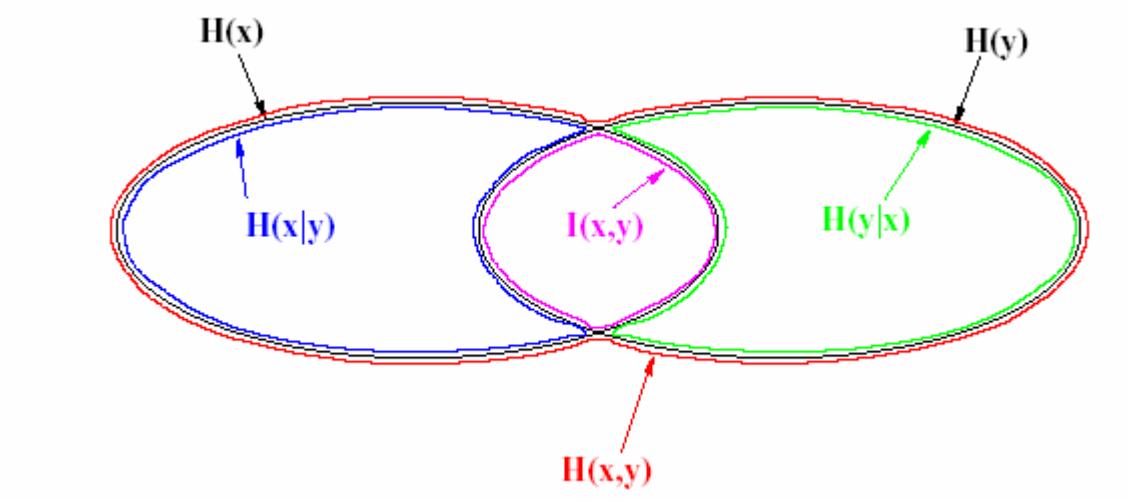

Fig. 7: Legami tra mutua informazione, densità condizionate e densità marginali

3.2.2 Principio a massima entropia

Nel filtro a particelle il problema era quello di derivare una funzione analitica per esprimere la stima della densità di probabilità a posteriori dello stato avendo a disposizione delle particelle disposte come tale funzione. Il problema è quindi analogo a quello di stimare la densità di probabilità di una variabile aleatoria sulla base di un numero finito di realizzazioni. La

conoscenza di tali campioni può tornare utile per avere delle stime dei momenti della densità di probabilità; il problema è quindi quello di trovare una funzione tale che rispetti il vincolo naturale (60) e le equazioni dei momenti (61):

$$\int_a^b f(x)dx = 1 \tag{60},$$

$$\int_a^b g_k(x) f(x)dx = m_k, k = 1,2,...n \tag{61},$$

dove si suppone di usare la conoscenza di *n* momenti.

Il *principio a massima entropia* offre una soluzione del problema affermando che, di tutte le probabilità che soddisfano la (60) e la (61), è da preferirsi la densità di probabilità che massimizza l'entropia d'informazione di Shannon, come nella (62):

$$\max_f \quad H(f) = \max_f \left\{ -\int_a^b f(x)\log f(x)dx \right\} \tag{62}.$$

Tale soluzione è particolarmente vantaggiosa perchè, dato che l'entropia rappresenta una misura dell'incertezza associata alle possibili realizzazioni di una variabile aleatoria, la massimizzazione dell'entropia conduce ad una distribuzione con la massima incertezza. Il principio a massima entropia risulta essere quindi molto onesto dato che non include nessuna informazione che non sia a propria disposizione e può essere pensato come un'estensione del famoso principio di Laplace della ragione insufficiente per cui, nel caso in cui nulla sia noto della variabile in questione, la migliore caratterizzazione di tale situazione consiste in una densità di probabilità uniforme.

La soluzione di questo problema di ottimizzazione con vincoli è nota in letteratura e fornisce la soluzione (63), come in [25]

$$f^*(x) = \frac{\exp\left(-\sum_{k=1}^n \lambda_k g_k(x) \right)}{\int \exp\left(-\sum_{k=1}^n \lambda_k g_k(x) \right) dx} \tag{63},$$

dove i moltiplicatori di Lagrange λ_k, possono essere ricavati dalle informazioni dei momenti come nella (64):

$$\int_a^b g_k(x)\exp\left(-\sum_{i=1}^{n}\lambda_k g_k\right)dx = m_k \tag{64}.$$

Le funzioni g_k rappresentano in genere i semplici momenti della variabile aleatoria X, cioè generalmente $g_k(X) = X^k$.

Lo svantaggio del calcolo analitico della funzione densità di probabilità a posteriori dello stato all'interno dell'algoritmo del filtro a particelle è la complessità computazionale; per ridurre questo svantaggio la stima a massima entropia nel *Particle Filter Tool-box* non è stata effettuata in modo esatto, ma nel modo approssimato come suggerito in [26].

La determinazione di una densità di probabilità a massima entropia viene infatti effettuata risolvendo un sistema non lineare di *k* equazioni, dove *k* rappresenta il numero di momenti che si vogliono usare nella stima. La soluzione di tale sistema permette di trovare i moltiplicatori di Lagrange e risolvere il problema dato che, introducendo il parametro λ_0 tale che

$$\int \exp\left(-\sum_{k=1}^{n}\lambda_k g_k(x)\right)dx = \exp(\lambda_0) \tag{65},$$

la (63) si trasforma nella (66) che, una volta noti i moltiplicatori di Lagrange, è univocamente determinata.

$$f^*(x) = \exp(-\lambda_0 - \lambda_1 g_1(x) \ldots - \lambda_k g_k(x)) \tag{66}.$$

L'idea proposta in [26] consiste nell'approssimare la funzione $f^*(x)$ mediante una combinazione lineare di funzioni di base come nella (67):

$$f^*(x) \approx \sum_{i=1}^{k}\beta_i f_i(x) = \hat{f}(x) \tag{67},$$

dove i coefficienti β_i sono calcolati in modo da risolvere il sistema lineare di equazioni formato dai vincoli della densità di probabilità:

$$\begin{cases} \beta_1 \int_a^b f_1(x)dx + \ldots + \beta_n \int_a^b f_n(x)dx = 1 \\ \beta_1 \int_a^b x f_1(x)dx + \ldots + \beta_n \int_a^b x f_n(x)dx = m_1 \\ \ldots \\ \beta_1 \int_a^b x^n f_1(x)dx + \ldots + \beta_n \int_a^b x^n f_n(x)dx = m_n \end{cases} \quad (68).$$

Mediante tale costruzione risulta che la funzione approssimante $\hat{f}(x)$ ha i primi *n* momenti in comune con la soluzione ottima $f^*(x)$ e l'approssimazione migliora quindi con l'aumentare dei momenti usati.

Le funzioni di base possono essere scelte in modo da includere le informazioni disponibili, ad esempio in modo tale che ognuna sia la soluzione di un problema semplificato di stima a massima entropia della forma (69):

$$\begin{cases} -\max \int_a^b f_i(x) \ln f_i(x)dx \\ \int_a^b f_i(x)dx = 1 \\ \int_a^b x^j f_i(x)dx = a_j \end{cases} \quad (69);$$

Per la (67) si ha quindi

$$f_j = \exp\left(-\lambda_{0j} - \lambda_j x^j\right), j = 1,\ldots,n \quad (70)$$

dove i moltiplicatori di Lagrange possono essere calcolati in accordo alla (64) e alla (65) come

$$\begin{cases} \int_a^b \exp\left(-\lambda_j x^j\right)dx = \exp\left(\lambda_{0j}\right) \\ a_j \int_a^b \exp\left(-\lambda_j x^j\right)dx - \int_a^b x^j \exp\left(-\lambda_j x^j\right)dx = 0 \end{cases} \quad (71).$$

L'idea consiste quindi nel ricondurre un problema non lineare in *k* incognite in un problema più semplice di *k* equazioni in una incognita.

Supponendo ora di conoscere *n* momenti, le equazioni di vincolo sono *n+1*, in quanto oltre alle *n* equazioni relative ai momenti va considerata l'equazione naturale che deve garantire che l'integrale della densità di probabilità valga 1. I moltiplicatori di Lagrange si ricavano invece usando la (71) e pertanto saranno solo *n*, e quindi anche le funzioni di base e i coefficienti delle combinazioni lineari saranno solamente *n*. Il sistema in (68) ha quindi più equazioni che incognite e una soluzione può essere ottenuta ad esempio con il metodo ai minimi quadrati; la soluzione proposta in [24] e ripresa nel *Particle Filter Tool-box* consiste invece nell'aggiungere un'altra funzione di base con un altro coefficiente in modo da poter risolvere il sistema con una semplice inversione di matrice. La funzione aggiunta è la densità di probabilità uniforme nell'intervallo considerato, $f = \frac{1}{b-a}$, in modo da non aggiungere informazioni sulle caratteristiche della densità di probabilità che si ottiene come risultato.

Il codice Matlab per ottenere una stima della densità di probabilità a massima entropia dato un insieme di realizzazioni era stato fornito in [24] e, a parte alcune modifiche per evitare ad esempio inversioni di matrici con determinante nullo o per poter affrontare anche i casi di densità di probabilità a forma di delta di Dirac, è stato ripreso per adattarlo alle esigenze del *Particle Filter Tool-box*.

3.3 Esempi di applicazioni nel Particle Filter Tool-box

3.3.1 Esempio n.1

Il primo esempio del funzionamento del Particle Filter Tool-box è lo stesso già introdotto nel paragrafo 2.4. Le equazioni di modello usate sono quindi la (52) e la (53), di seguito ricordate per comodità:

$$x_{k+1} = 1 + \sin(\pi\omega k) + \phi_1 x_k + v_k \tag{52}$$

$$y_k = \begin{cases} \phi_2 x_k^2 + v_k & t \le 30 \\ \phi_3 x_k - 2 + n_k & t > 30 \end{cases} \tag{53};$$

di seguito sono mostrati alcuni grafici che mostrano gli output del *Particle Filter Tool-box* al termine di una simulazione di durata 60 passi, condotta usando 200 particelle.

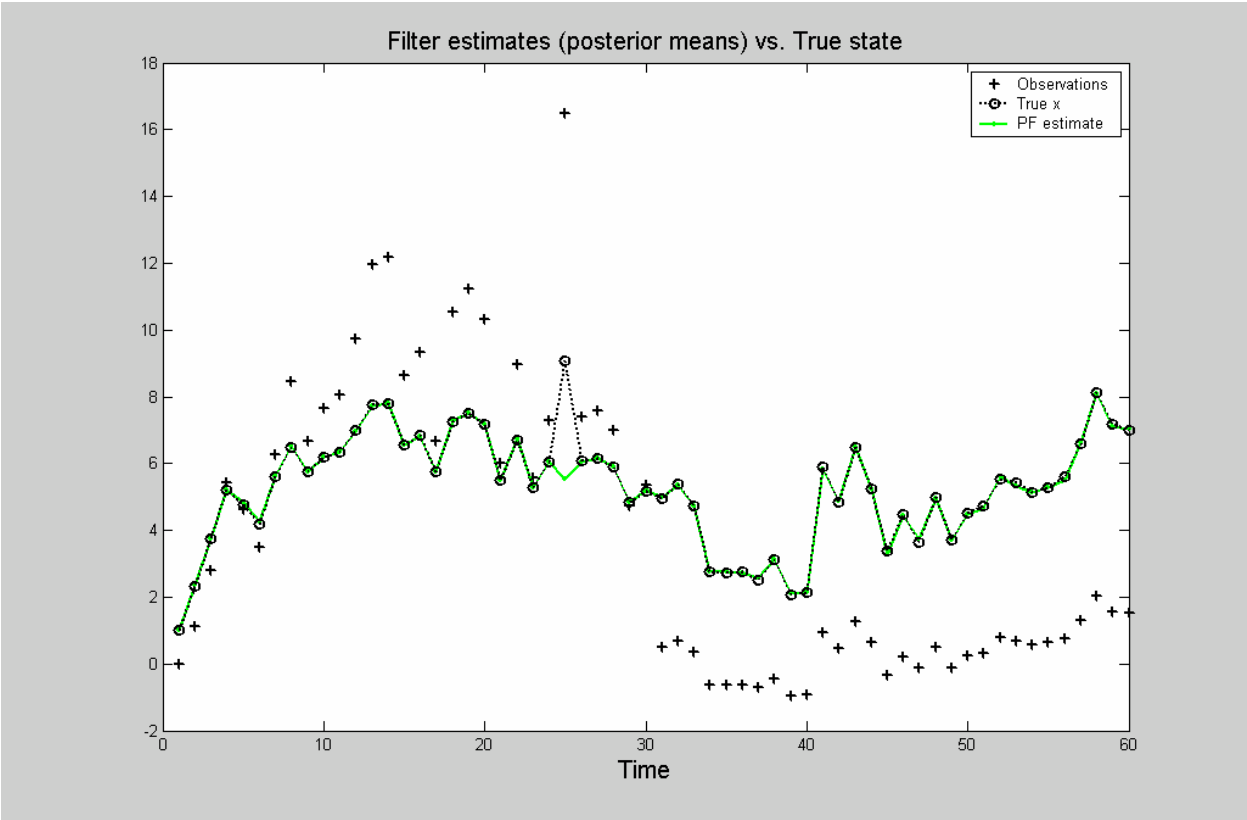

Fig. 8: Ricostruzione dello stato mediante stima con filtro a particelle nell'esempio 1

Le statistiche di tale simulazione prodotta come output dell'algoritmo sono invece:

*********** RISULTATI FINALI **********

Root Mean Square Errors (Varianza)

PF = 0.45988(0)

Errore percentuale medio

PF = 1.339%

Tempo di esecuzione (secondi)

PF = 183.354

La varianza vale 0 in quanto viene calcolata sulla base di più simulazioni. Come già evidenziato nel precedente paragrafo inoltre, ogni 10 passi, l'algoritmo mostra come output sullo schermo la densità di probabilità a posteriori dello stato ricavata in forma analitica. Infine vengono mostrati i

valori assunti dalle particelle al termine di ogni iterazione multipla di 10 e il disegno della densità di probabilità a posteriori approssimata calcolata in base a tali campioni con il criterio della stima a massima entropia.

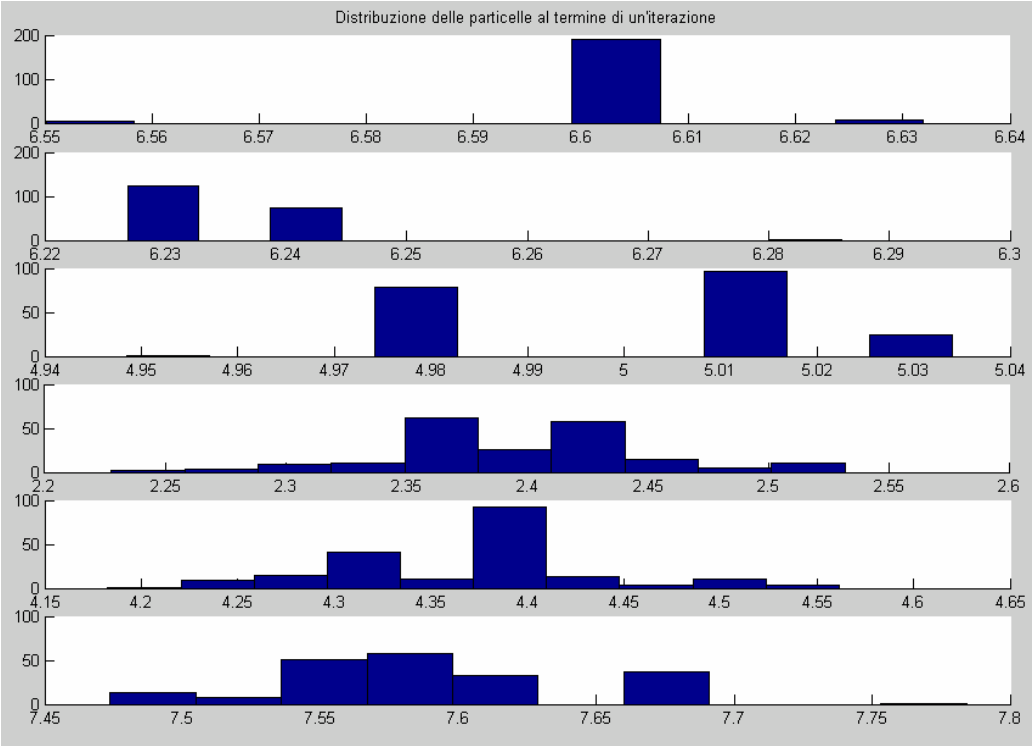

Fig. 9: Distribuzione delle particelle al termine di ogni iterazione multipla di 10 nell'esempio 1

Come evidente dalla Fig.9 le particelle assumono valori molto vicini tra loro e spesso esistono poche versioni dello stato realmente diverse fra loro. Ciò è sostanzialmente dovuto al fatto che questo processo non è né troppo non lineare, né troppo rumoroso, per cui le stime sono generalmente buone; le densità di probabilità che ci si aspetta di ottenere dai precedenti campioni sono quindi funzioni molto strette, per cui le probabilità di ottenere determinati stati ad un dato passo sono quasi ovunque nulle eccetto in piccoli intervalli dove la probabilità è alta. La Fig. 10 mostra infatti le densità di probabilità ricavate dalle particelle dello stato disposte come in Fig.9 e suffraga la precedente ipotesi. In particolare ai passi 40, 50 e 60 l'insieme delle particelle è leggermente più vario, difatti la densità di probabilità a posteriori forma una curva leggermente più larga. La ricostruzione di questa funzione è effettuata usando i primi tre momenti dei campioni.

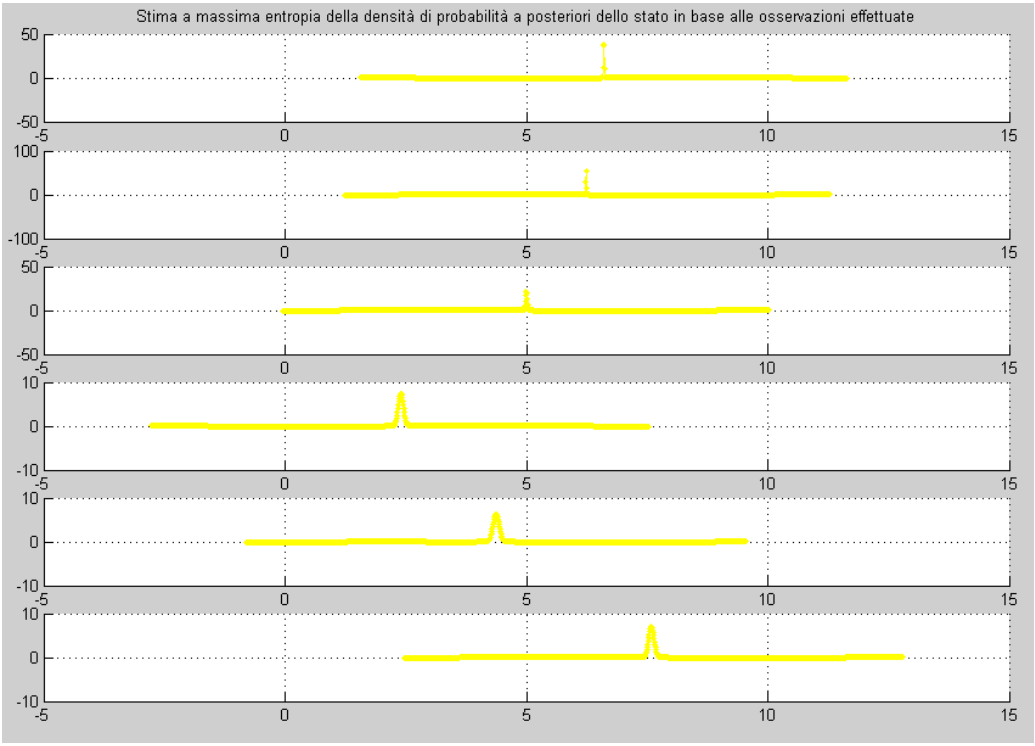

Fig. 10: Stima a massima entropia della densità di probabilità a posteriori dello stato ad ogni iterazione multipla di 10 nell'esempio 1

3.3.2 Esempio n.2

Il secondo esempio di possibili applicazioni del *Particle Filter Tool-box* è di nuovo un problema già studiato in letteratura [2,8], particolarmente interessante a causa di consistenti non-linearità e della presenza di molto rumore; si tratta inoltre di un sistema non autonomo, è presente infatti un termine di ingresso legato al passo temporale corrente.

Fino ad ora, per non appesantire inutilmente la trattazione, non si era mai tenuto conto della possibilità della presenza di ingressi nel sistema, come evidente nella (1) e nella (2); in realtà i risultati ottenuti sono indipendenti dalla presenza o meno di ingressi, nell'ipotesi che questi siano ovviamente noti e di tipo deterministico.

Il nuovo modello del processo è rappresentato dalla (72) e dalla (73):

$$x_k = \frac{x_{k-1}}{2} + \frac{25x_{k-1}}{1+x_{k-1}^2} + 8\cos(1.2k) + v_{k-1} \tag{72}$$

$$y_k = \frac{x^2}{20} + n_k \tag{73},$$

dove v_{k-1} e n_k sono due variabili aleatorie gaussiane a valor medio nullo e varianza rispettivamente 10 e 1.

La simulazione effettuata sfrutta un'evoluzione dello stato di 60 passi con 200 particelle; le non linearità del sistema e il molto rumore causano comunque un peggioramento delle prestazioni del filtro, come è possibile notare dalla Fig.11 che mostra il confronto tra le stime dello stato ed i valori reali.

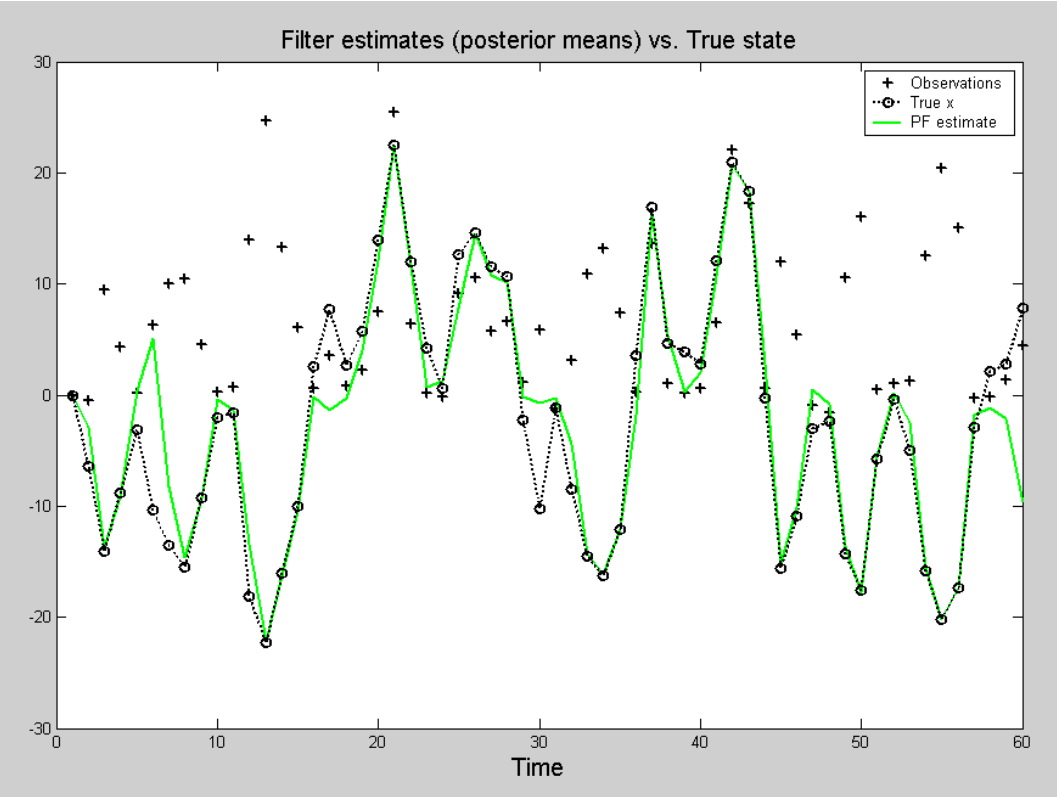

Fig. 11: Ricostruzione dello stato mediante stima con filtro a particelle nell'esempio 2

Le statistiche di questa simulazione sono riportate invece nel seguente riepilogo fornito come output dall'algoritmo di stima:

********** *RISULTATI FINALI* **********

Root Mean Square Errors (Varianza)

PF = 4.0588(0)

Errore percentuale medio

PF = 64.2348%

Tempo di esecuzione (secondi)

PF = 93.785

Il dato sull'errore percentuale medio commesso potrebbe lasciar pensare che lo stimatore stia sbagliando completamente la ricostruzione dello stato, in contrasto con quanto lasciato intendere dalla Fig.11 che mostra come gli stati stimati non siano troppo lontani dai valori reali.

La Fig. 12 mostra dunque una simulazione ottenuta su un'evoluzione dello stato di 20 passi e l'errore percentuale commesso ad ogni passo;

si può notare come anche quando la stima è buona, l'errore percentuale può assumere valori altissimi dato che lo stato reale assume valori molto piccoli, come ad esempio accade nell'ultimo istante di simulazione.

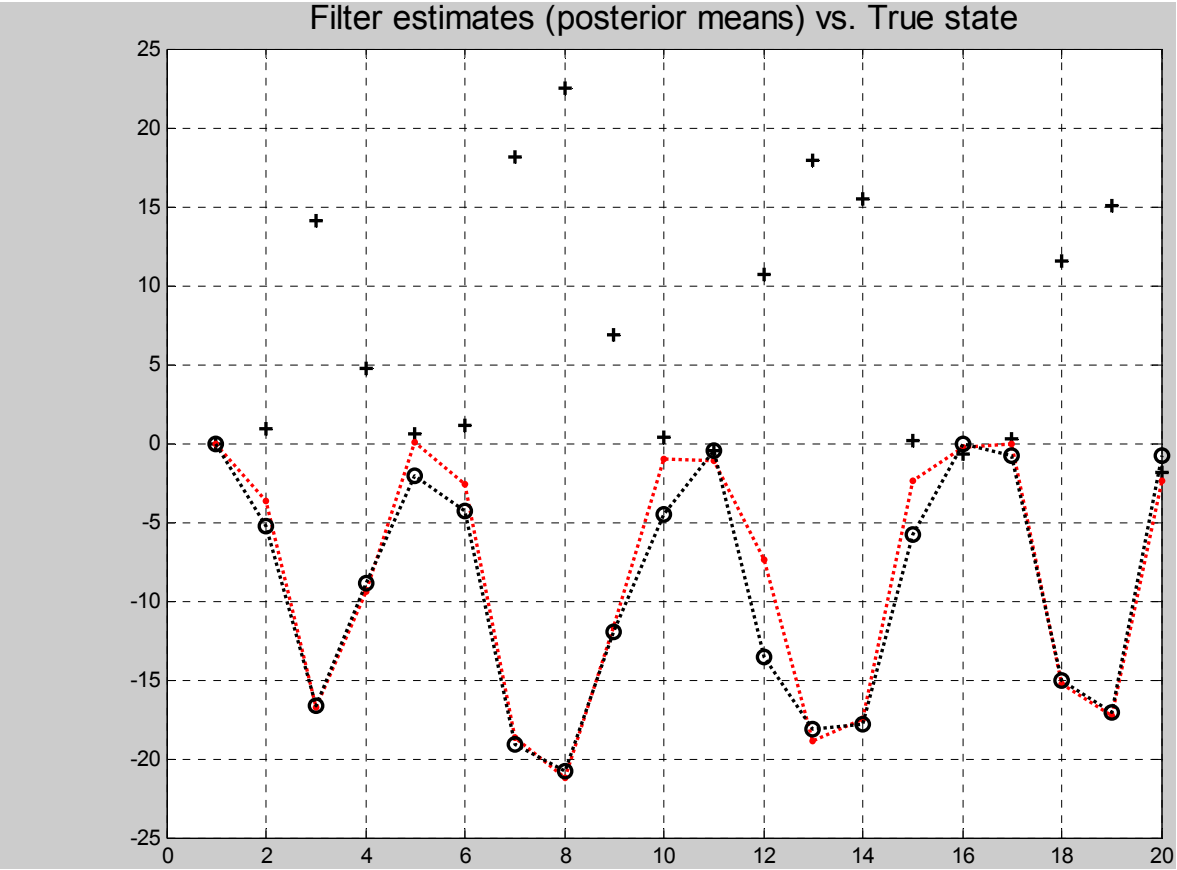

Fig. 12: Stima di una simulazione di durata 20 passi eseguita per il calcolo degli errori percentuali

Istante	Errore percentuale	Istante	Errore percentuale
2	30.1385 %	12	45.3306 %
3	0.2131 %	13	4.2438 %
4	6.3273 %	14	1.8886 %
5	101.6432 %	15	58.5976 %
6	39.4327 %	16	139.5136 %
7	2.2454 %	17	96.0768 %
8	1.9528 %	18	0.8446 %
9	2.3788 %	19	1.1854 %
10	77.7507 %	20	205.6434 %
11	141.0527 %		

Tabella 1: Errori percentuali commessi in una simulazione di 20 passi dell'esempio 2

La Fig.13 mostra invece il posizionamento delle particelle al termine di ogni iterazione multipla di 10.

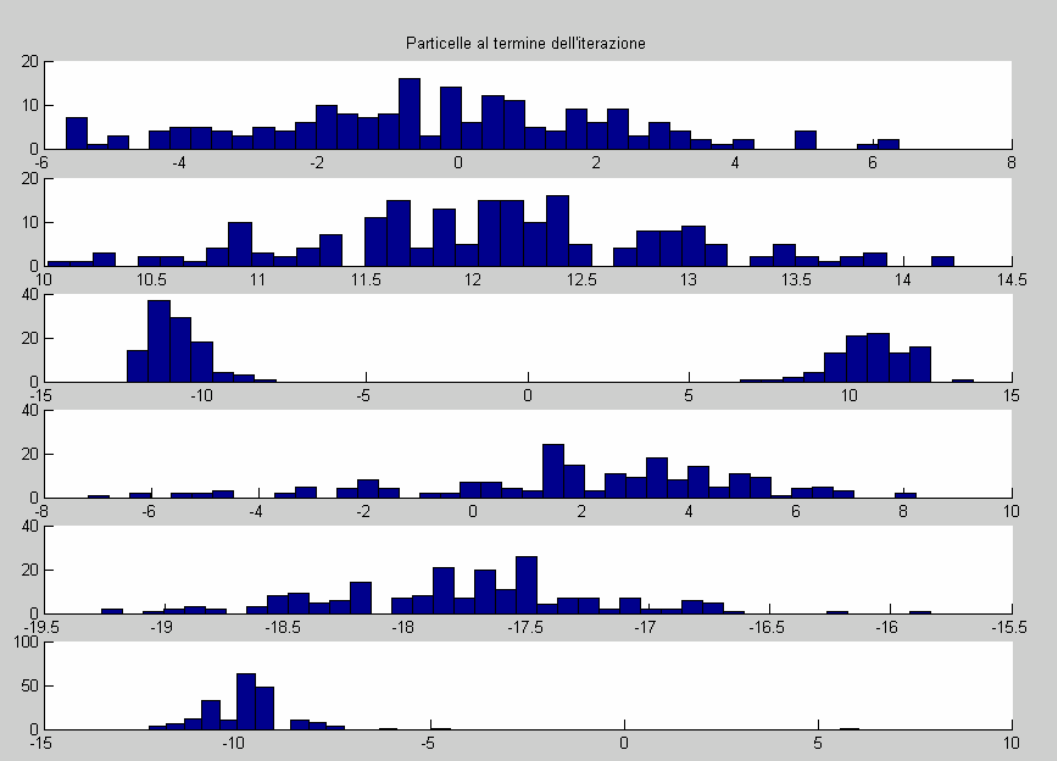

Fig. 13: Distribuzione delle particelle al termine di ogni iterazione multipla di 10 nell'esempio 2

In questo caso si può notare come i valori assunti dalle particelle siano in generale più vari, e al passo 10 e al passo 40 lasciano immaginare densità di probabilità piuttosto larghe. Al passo 30 si

nota invece come le particelle dello stato siano raggruppate in due posizioni molto lontane fra loro e il valor medio sia in questo caso particolarmente non significativo, dato che nessuna versione dello stato prevede che il valor medio possa effettivamente rappresentare il valore vero dello stato. Difatti, come mostra la Fig.11, la stima al passo 30 è sbagliata in quanto il valore vero vale circa –10; la ricostruzione della densità di probabilità con il criterio a massima entropia è in questa situazione particolarmente problematica, dato che la funzione dovrebbe avere una forma bimodale e al centro valere 0 per un grande intervallo. La Fig.14 mostra appunto la forma della densità di probabilità ai vari passi:

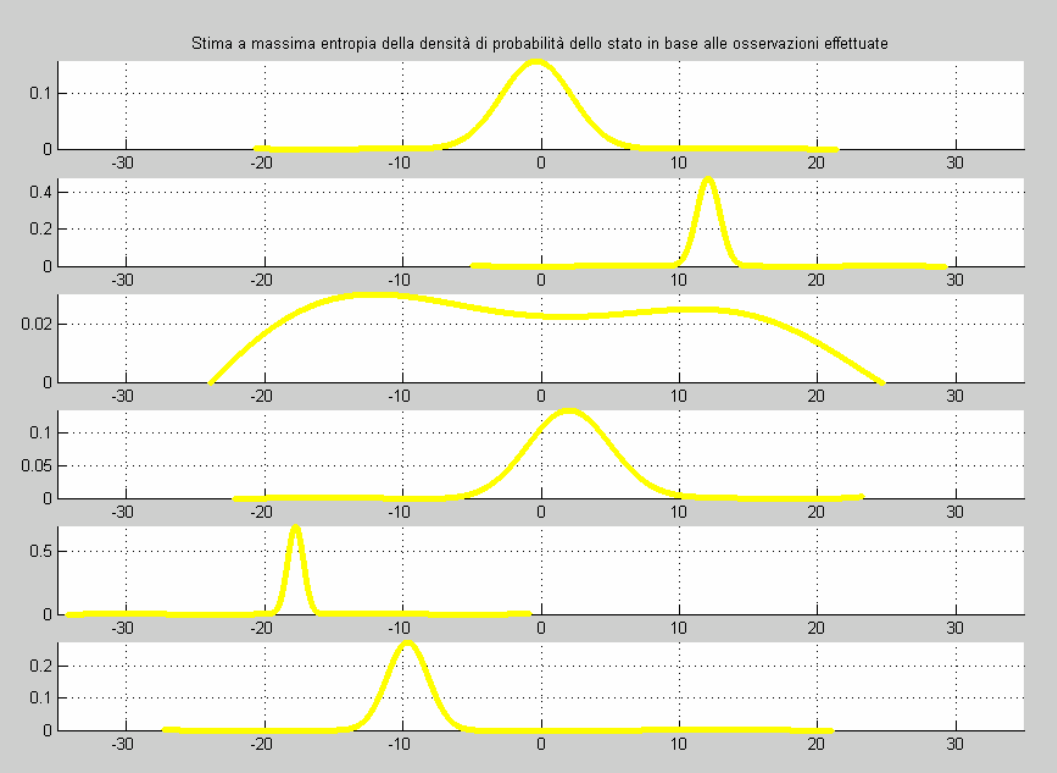

Figura 14: Stima a massima entropia della densità di probabilità a posteriori dello stato ad ogni iterazione multipla di 10 nell'esempio 2

La ricostruzione della densità di probabilità a posteriori dello stato è stata effettuata in questa situazione sfruttando la conoscenza dei primi 5 momenti delle particelle, in modo da poter modellare in maniera più precisa le densità di probabilità che, in questo esempio, hanno generalmente forme più strane.

3.3.3 Esempio n.3

Un terzo esempio, già proposto in letteratura in [1], è una versione semplificata di un *passive tracking problem*: si ha infatti un target che si muove lungo una linea e un sensore, montato su

una piattaforma a sua volta in moto, calcola il *line of sight angle* allo scopo di rintracciare la posizione dell'obiettivo.
Il target si muove lungo l'asse x secondo la legge

$$\bar{x}_{k+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \bar{x}_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v_k \tag{74},$$

dove $\bar{x}_k = \begin{bmatrix} x_{1k} & x_{2k} \end{bmatrix}^T$ e x_{1k} e x_{2k} rappresentano rispettivamente la posizione e la velocità del target al k-esimo passo e il rumore del processo v_k è un rumore bianco gaussiano a media nulla e densità spettrale di potenza $q = 10^{-2}$. Un sensore è posto su di una piattaforma che si muove nel piano secondo la legge (75):

$$\begin{cases} x_k^p = 4k \\ y_k^p = 20 \end{cases} \tag{75}.$$

Lo stato iniziale del target vale $x_0 = \begin{bmatrix} 80 & 1 \end{bmatrix}^T$; il sensore misura ad ogni passo *k* l'angolo tra l'orizzontale e il *line of sight* tra il sensore e il target:

$$y_k = \tan^{-1}\left(\frac{y_k^p}{x_{1k} - x_k^p} \right) + w_k \tag{76},$$

dove il rumore di misurazione w_k è di nuovo un rumore bianco gaussiano a media nulla e varianza $r_s = (4°)^2$ e indipendente dal rumore che influenza l'evoluzione dello stato del target.

All'istante iniziale si suppone di non avere notizie sicure né sulla velocità, né sulla posizione del target; in particolare si suppone che l'obiettivo possa essere ovunque tra 0 e 500 e la sua velocità possa essere qualsiasi tra -2 e 4. Questo dato può essere modellato nel *Particle Filter Tool-box* considerando la densità di probabilità della posizione del target uniformemente distribuita in [0,500] e la densità di probabilità della velocità uniformemente distribuita in [-2,4].

In questo caso la simulazione effettuata sfrutta un'evoluzione di 30 passi compiuta usando 500 particelle; la Fig. 15 mostra l'esito di un esperimento:

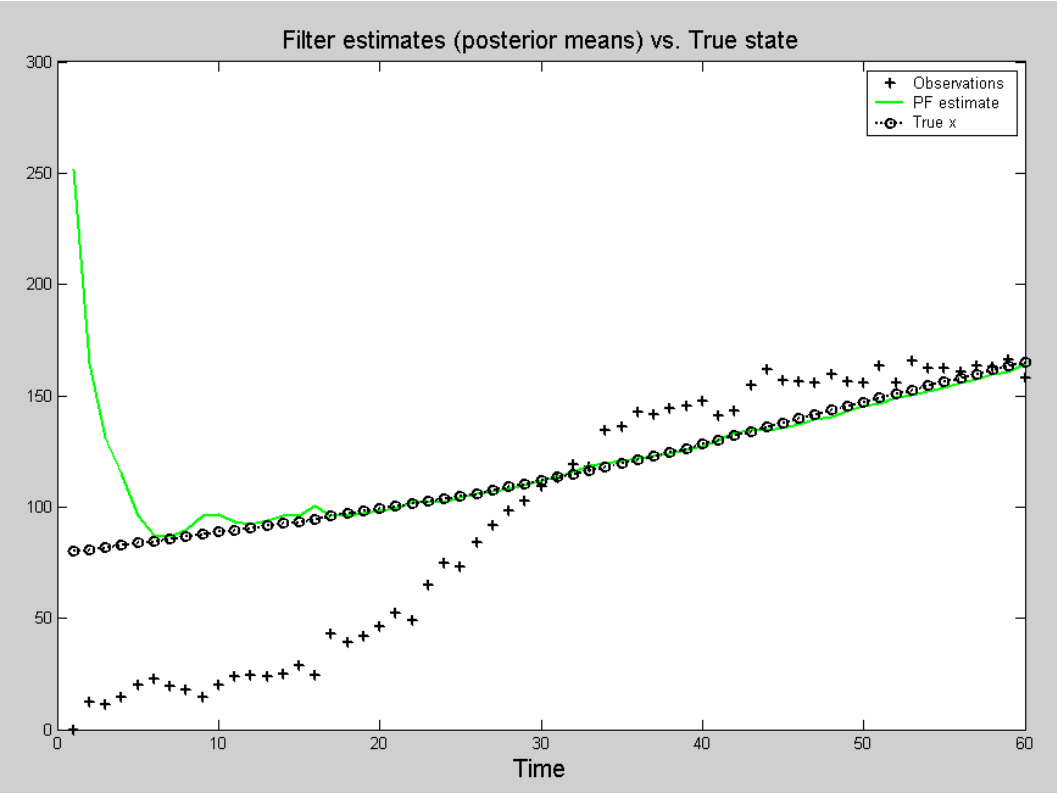

Figura 15: Ricostruzione dello stato mediante stima con filtro a particelle nell'esempio 3

In questo caso la figura mostra solamente il confronto tra la stima del primo dei due stati e la posizione reale del target. Come si può notare all'inizio la stima vale circa 250, dato che la densità di probabilità dello stato iniziale è una funzione uniformemente distribuita tra 0 e 500, mentre il valore reale è 80. Le osservazioni, nonostante siano spesso errate dato che la varianza del sensore è di $(4°)^2$ permettono comunque di ricostruire abbastanza immediatamente la posizione reale dell'obiettivo. Le statistiche della simulazione calcolate dallo script Matlab e riportate in uscita valgono in questo caso:

********** *FINAL RESULTS* **********

Root Mean Square Errors

PF posizione = 13.7146
PF velocità = 0.34501

Errore percentuale medio

PF posizione = 5.2529%
PF velocità = 21.8909%

Execution time (seconds)

PF = *50.683*

Gli RMSE relativi alla velocità sono ovviamente minori dato che un piccolo errore nella stima della velocità causa un errore amplificato nella stima della posizione. Gli errori percentuali sono maggiori nella stima della velocità, ma questo è dovuto al fatto che la velocità assume valori dell'ordine dell'unità, mentre quelli la posizione è in genere due ordini di grandezza maggiore.

La Fig.16 mostra invece la disposizione delle particelle relative alla posizione del target, calcolate all'istante iniziale e alle iterazioni multiple di 10:

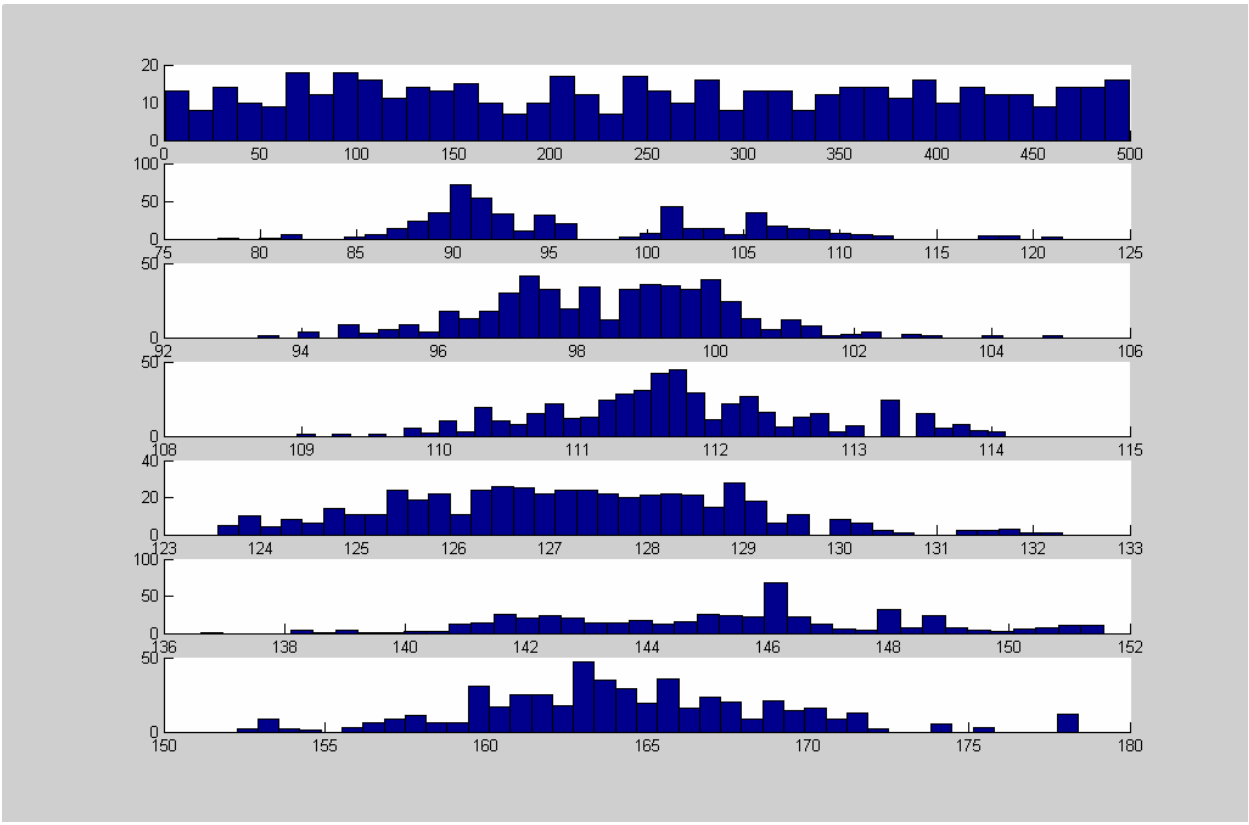

Fig. 16: Distribuzione delle particelle relative alla posizione del target al passo iniziale e al termine di ogni iterazione multipla di 10

Dalla analisi della disposizione delle particelle ci si può immaginare una densità di probabilità uniforme al passo 0 e densità di probabilità generalmente larghe ai passi successivi. Le due figure seguenti mostrano comunque le stime delle densità di probabilità a posteriori sia della posizione che della velocità del target, mostrando analogie con i valori assunti dalle varie versioni dello stato.

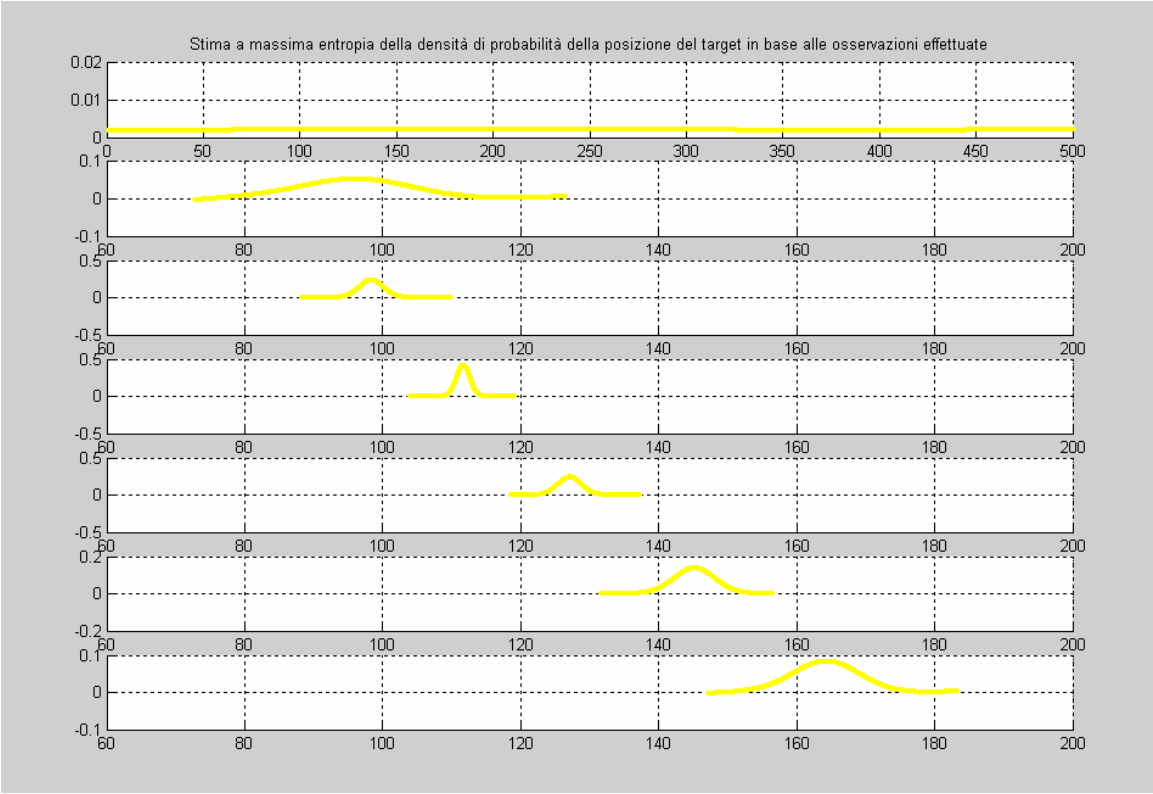

Fig. 17: Stima della densità di probabilità a posteriori della posizione del target note le osservazioni

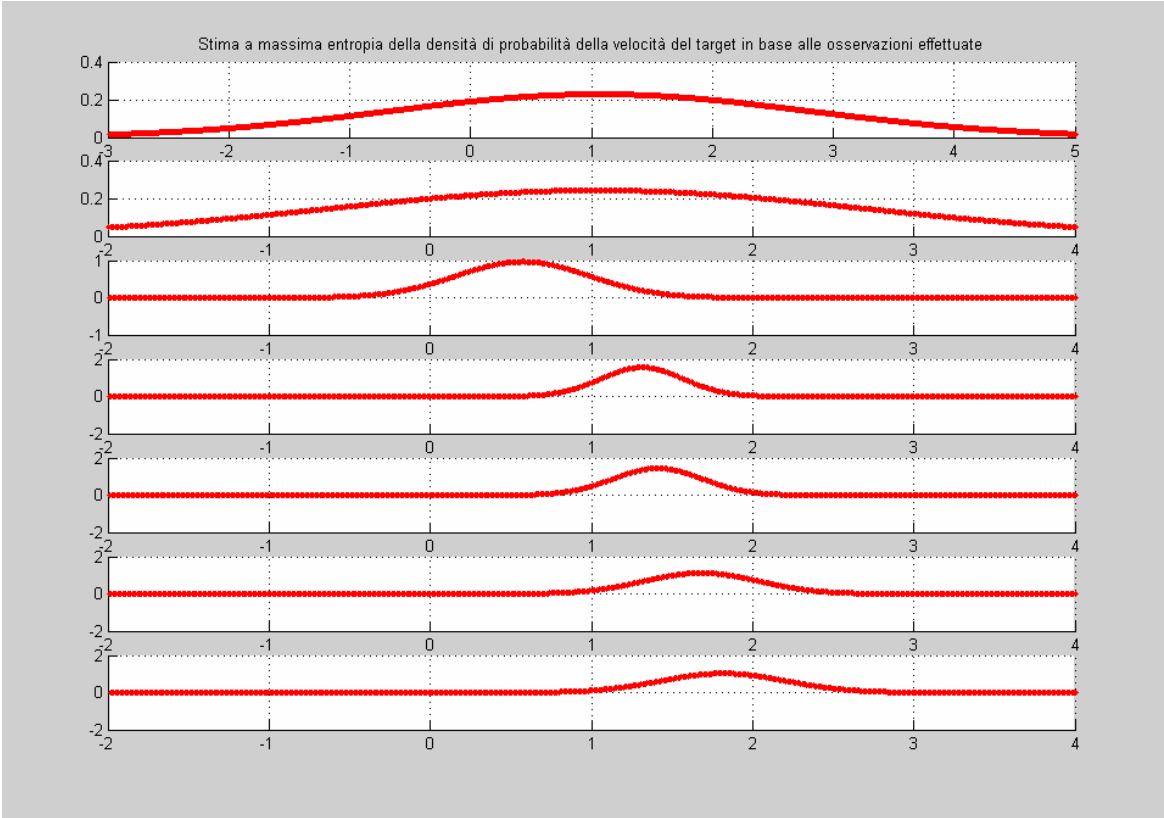

Fig. 18: Stima della densità di probabilità a posteriori della velocità del target note le osservazioni

L’equazione (74) che descriveva il modello dell’evoluzione dello stato affermava che la velocità del target doveva essere sempre costante a meno del rumore di processo, difatti risulta essere sempre intorno al valore unitario. In particolare, in questa simulazione, dalla Fig.18 si evince come non accada mai negli istanti di rappresentazione che la velocità assuma valori minori di 0, e quindi risulta che il target, in quei momenti, si muove in avanti senza tornare indietro. La Fig.17 mostra come in effetti la densità di probabilità della posizione del target si sposti in avanti. La ricostruzione della densità di probabilità a posteriori dello stato in questo caso è stata effettuata usando i primi tre momenti e ogni 10 iterazioni viene fornita la stima analitica della funzione sia per il primo che per il secondo stato. generalmente forme più strane.

3.3.4 Esempio n.4

Un ulteriore esempio di applicazione che può essere affrontata con un filtro a particelle riguarda il problema di un robot mobile che si muove in un territorio ignoto; in tale situazione risulta conveniente richiedere che il robot, tramite i suoi sensori, sia in grado di ricostruire una mappa dell’ambiente circostante e contemporaneamente di stabilire la propria posizione in tale ambiente; questo problema è noto in letteratura tramite l’acronimo SLAM, sigla di Simultaneous Localization And Mapping.

Una soluzione comune al problema consiste nell’immaginare che nell’ambiente ci sia un insieme discreto di punti di riferimento, generalmente chiamati in letteratura *landmark* o *feature*, che permettono al robot di orientarsi; si suppone infatti che il robot sia dotato di opportuni sensori tramite i quali è in grado di conoscere la propria posizione relativa rispetto a tali punti di riferimento. Un landmark è caratterizzato da una coppia di valori che rappresentano le proprie coordinate in un piano cartesiano; l’insieme di tutti i landmark costituisce la mappa dell’ambiente.

Il problema di SLAM può essere quindi scomposto in due sottoproblemi, il primo relativo alla *localizzazione* del robot, il secondo alla creazione della *mappa*, ovvero alla determinazione della posizione assoluta di tutti i punti di riferimento.

Nel problema di localizzazione si suppone che la mappa sia nota, cioè siano note le posizioni di tutti i landmark. Il robot è dotato inoltre di sensori interni, ad esempio encoder, che permettono di conoscere una stima della propria posizione in modo odometrico; in pratica i sensori contano il numero di giri effettuati dalle ruote del robot e, conoscendo la posizione iniziale, stimano la posizione occupata dal robot ad ogni istante. L’utilizzo di questo unico modo di stimare la posizione del robot rende le stime alquanto imprecise, a causa soprattutto di

fenomeni di slittamento non considerati. Il filtro a particelle può quindi essere usato per stimare lo stato del sistema costituito dal vettore posizione del robot $\bar{x} = \begin{bmatrix} x_r & y_r & \vartheta_r \end{bmatrix}^T$ ed il sensore odometrico viene sfruttato per la predizione del nuovo vettore di stato, perturbato con del rumore per simulare il disturbo presente nell'encoder. Ad ogni istante si suppone di osservare un particolare landmark e, conoscendone la posizione assoluta, si calcola la distanza relativa e l'angolo relativo con ciascuna particella. La densità di probabilità di likelihood rappresenta quindi la differenza tra ciascun valore atteso e la misura reale effettuata; in base a quanto ciascuna predizione della misura si avvicina alla misurazione vera vengono assegnati i pesi ai vari campioni e, tramite la solita procedura di resampling, si ottiene una stima ricorsiva dello stato del robot e quindi della sua posizione.

Il problema di creazione della mappa è invece maggiormente complicato, in quanto in questo caso il vettore di stato è molto più lungo poiché è costituito dalla posizione assoluta di tutti i landmark. La mappa viene infatti stimata supponendo di conoscere a priori la posizione assoluta del veicolo; il problema non ha solamente valenza teorica, infatti la posizione assoluta del robot può essere conosciuta facilmente ad esempio per mezzo di un semplice GPS. Al contrario del problema di localizzazione, la creazione della mappa non è un problema standard per particle filter perché il modello di aggiornamento dello stato dovrebbe essere costituito dalla semplice matrice d'identità, dato che è ragionevole pretendere che la cartina dell'ambiente non cambi ad ogni iterazione. La soluzione generalmente trovata in letteratura sfrutta perciò un filtro di Kalman esteso.

L'applicazione SLAM rappresenta quindi il problema di navigazione in forma molto generale in quanto richiede che un veicolo che si muove in un ambiente sconosciuto sia in grado di rilevare sia la propria posizione che di costruire una mappa consistente dell'ambiente circostante. Addizionalmente può essere richiesto che il veicolo sfrutti la cartina da lui stesso costruita per pianificare e controllare la propria traiettoria all'interno di tale ambiente.

Le applicazioni più evidenti riguardano ambienti in cui la presenza dell'uomo è complicata o pericolosa, come ad esempio nelle esplorazioni sottomarine, o nei viaggi interplanetari o in zone disastrate. In questi casi lo SLAM rappresenta un algoritmo fondamentale per ottenere un veicolo completamente autonomo senza il bisogno di aiuti esterni quali GPS.

La soluzione tipica del problema SLAM consiste nella scomposizione in un problema di localizzazione e uno di stima della posizione dei landmark condizionata alla stima del punto in cui il robot si trova. La stima a posteriori della posizione del veicolo in base alle misurazioni può quindi essere effettuata tramite un filtro a particelle standard, dove ciascuna particella possiede

un filtro di Kalman di ordine *k*, dove *k* è il numero dei punti di riferimento. La richiesta computazionale è quindi piuttosto pesante dato che è un $O(m \cdot k)$ dove m è il numero di particelle. L'applicazione SLAM, come precedentemente osservato, è molto interessante e molto studiata, e pertanto sono stati suggeriti in letteratura molti metodi per diminuire la complessità del problema, ad esempio limitando il problema del resampling a poche iterazioni come in [18].

In questo esempio non vengono sfruttati algoritmi specifici, ma la soluzione viene determinata nell'ambito della struttura di base del filtro a particelle.

Come detto il problema viene suddiviso in due sottoproblemi, dato che si può osservare che se la posizione del veicolo fosse nota ad ogni istante, allora la stima della posizione dei vari landmark può essere effettuata in modo indipendente l'una dall'altra. In pratica nella realtà la posizione del robot non è nota, ma l'indipendenza permette di fattorizzare la densità di probabilità con un termine che rappresenta la probabilità della propria posizione e il prodotto di *k* termini che stimano la posizione dei landmark condizionata a ciascun ipotetico percorso del veicolo, come in [27].

L'utilizzo del filtro a particelle nel problema di localizzazione e mappaggio simultaneo comporta l'utilizzo di un algoritmo noto in letteratura come FastSLAM; la soluzione per il precedente esempio consiste in un algoritmo Matlab tratto da [28], dove sono state apportate alcune modifiche per sostituire un filtro EKF con appunto un filtro a particelle. In tale simulazione il numero di landmark può essere scelto dall'utente e nel particolare esempio è stato posto uguale a 10. Il robot si trova inizialmente al centro di una mappa quadrata di dimensione 200 e la posizione delle feature è scelta in modo casuale. Si nota come le 200 particelle all'istante iniziale siano un vettore di dimensione 3, in modo da poter rappresentare la posizione e la direzione del veicolo in un piano cartesiano. Ad ogni istante di campionamento una nuova misurazione è resa disponibile da un landmark qualsiasi in modo casuale, l'unica supposizione è che il robot sia in grado di stabilire da quale landmark provenga. Ad esempio si può supporre che le osservazioni vengano effettuate in radio frequenza e che le frequenze di trasmissione dei vari landmark siano differenti; alternativamente si può supporre che ciascun landmark invii un ultima sequenza di bit discriminante. Ogni volta che il robot si rende conto che l'informazione ricevuta proviene da un landmark che fino a quel momento non aveva mai trasmesso, il vettore di stato aumenta e contiene due nuove righe relative alla stima delle coordinate cartesiane di quella feature. Da un certo momento in poi comunque tutti e 10 i punti di riferimento saranno stati utilizzati e quindi la dimensione del vettore di stato rimarrà costante e pari a $3 + 2 \cdot k$.

Quando una misurazione viene ottenuta da un nuovo landmark non è possibile modificare l'insieme delle particelle dato che non ha senso parlare di densità di likelihood dato che non si può prevedere la posizione di una feature che non si conosceva; in questo caso perciò l'evoluzione del processo è in anello aperto, o similmente il problema di navigazione viene effettuato in *dead reckoning*, ovvero usando esclusivamente i dati odometrici. Quando tale feature viene però consultata nuovamente, una stima della sua posizione è già nota e può quindi essere usata per osservare quali particelle descrivano in modo migliore la nuova misurazione. In questa situazione quindi le particelle possono essere ricampionate in modo da poter continuare a descrivere l'evoluzione del processo in modo adeguato.

Nell'esempio particolare studiato si suppone che il modello di evoluzione dello stato del veicolo sia:

$$\bar{x}_{k+1} = \bar{x}_k \oplus \begin{bmatrix} 0 & 0.15 & \frac{0.1\pi}{180} \end{bmatrix}^T + \bar{v}_k \qquad (77),$$

dove v_k è un rumore gaussiano di matrice di covarianza $U = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & \frac{1.2\pi}{180} \end{bmatrix}^2$. Il simbolo $\oplus$ vuole indicare che la somma è una somma vettoriale, dato che il vettore $\bar{u} = \begin{bmatrix} 0 & 0.15 & \frac{0.1\pi}{180} \end{bmatrix}^T$ è espresso in un riferimento di coordinate solidale con il veicolo; prima di effettuare la somma è quindi necessario effettuare una trasformazione di coordinate per poter sommare algebricamente i due vettori; la matrice di trasformazione di coordinate è una semplice matrice di rotazione che sfrutta la stima dell'angolo formato tra la direzione del veicolo e l'orizzontale.

I rimanenti elementi del vettore di stato (cioè la mappa) evolvono secondo la semplice relazione (78):

$$\bar{x}_{k+1} = \bar{x}_k \qquad (78).$$

Il modello delle osservazioni è invece

$$\bar{z} = \begin{bmatrix} \left\| \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T - \begin{bmatrix} x_j - x_{j+1} \end{bmatrix}^T \right\| \; ; \quad AngleWrap\left(Atan2\left(\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T - \begin{bmatrix} x_j & x_{j+1} \end{bmatrix}^T \right) - x_3 \right) \end{bmatrix} (79),$$

dove j rappresenta l'indice relativo alla posizione nel vettore di stato del landmark che fornisce l'informazione a quel particolare istante temporale. La funzione *AngleWrap* permette semplicemente di limitare l'angolo $z(2)$ all'intervallo $(-\pi, \pi]$. In questo caso si suppone che la matrice di covarianza dell'errore nel sensore di misurazione valga $R = \begin{bmatrix} 1.1 & 0 \\ 0 & \frac{5\pi}{180} \end{bmatrix}^2$.

La seguente tabella riporta i risultati di una particolare simulazione di durata 500 passi, dove i dati raccolti ogni 50 passi sono relativi alle distanze tra le posizione vere e quelle stimate.

k	errore localizzazione	distanza feature1	distanza feature2	distanza feature3	distanza feature4	distanza feature5	distanza feature6	distanza feature7	distanza feature8	distanza feature9	distanza feature10
50	0.1010	2.4818	1.4972	1.7612	1.6742	4.6714	6.9641	6.3804	1.7437	0.8934	2.1432
100	0.4505	0.7842	0.7106	0.5654	0.5444	1.1055	2.8333	5.8350	5.4207	0.7881	3.7270
150	0.1619	0.7388	0.8141	0.6346	2.1306	1.5007	1.7631	2.2697	3.9164	0.7477	3.6719
200	0.5013	1.6336	1.0860	0.3821	1.3014	0.5326	1.3886	2.9932	4.5128	0.7169	3.1451
250	0.5599	1.1153	1.1038	1.0299	0.7429	0.1772	1.0956	2.7748	4.7374	0.6804	3.1432
300	0.7703	0.6708	0.7134	0.7717	1.5118	0.0814	0.1655	2.6017	4.2954	0.6720	3.1739
350	0.8011	0.7271	1.2756	1.5831	1.0828	0.1564	0.1593	3.0243	3.0972	0.7051	3.0089
400	0.7407	0.7207	1.0135	1.8332	0.9470	0.1381	0.1495	3.1702	3.2816	0.7589	2.5906
450	0.8391	0.6181	1.2060	1.6853	1.3168	0.2133	0.2857	2.8415	2.8956	0.7156	3.0028
500	1.0071	0.7415	0.6014	1.8375	1.8087	0.2662	0.5015	3.0221	2.9074	0.6369	2.6556

Tabella 2: Errori nella localizzazione del robot mobile e nella determinazione della posizione del robot mobile in una simulazione di 500 passi di un problema SLAM

In tale simulazione si è supposto che fosse nota la posizione iniziale del veicolo, ovvero tutte le particelle sono state inizializzate con il valore esatto del vettore $\bar{x}$; si può quindi supporre di mettere il veicolo in una posizione nota e da lì mandarlo in esplorazione in un ambiente ignoto.

Nel caso in cui ci fosse incertezza nella posizione iniziale del robot mobile si può notare che l'incertezza nella determinazione delle features non può mai scendere al di sotto dell'incertezza iniziale; nella migliore delle ipotesi si può quindi sperare di ridurre l'incertezza nei landmark al livello minimo pari all'incertezza iniziale. Nella figura 19 è invece mostrato l'andamento grafico della stima della posizione del veicolo e dei landmark. La posizione del veicolo è rappresentata da asterischi di colore rosso, mentre la stima da pallini di colore giallo. In verde sono rappresentate le dieci posizioni delle features che, come si può notare sono in posizione sparsa casualmente nella mappa di dimensione 200×200 ; ad ogni simulazione la posizione dei

landmark quindi cambia. Gli ellissi che sono tracciati intorno alla feature rappresentano gli ellissi di covarianza relativi a ciascun landmark, calcolati mediante le informazioni sulla varianza della misura ottenuti tramite il filtro di Kalman esteso, e vengono tracciati ogni 50 passi di simulazione. In particolare si può notare che tali ellissi tendono a diminuire di dimensione poiché l'incertezza diminuisce ad ogni nuova osservazione come ci si può ragionevolmente attendere. Gli ellissi sono in genere disposti in modo perpendicolare alla posizione del veicolo in quanto l'errore nella stima dell'angolo di posizione del veicolo causa maggior errore di quello nella stima della distanza, dato che dipende fortemente dalla lontananza dalla feature. Si può infatti notare anche che gli ellissi relativi ai landmark più lontani sono generalmente maggiormente schiacciati. Si può altresì notare come in genere la stima è migliore se la feature è più vicina al veicolo.

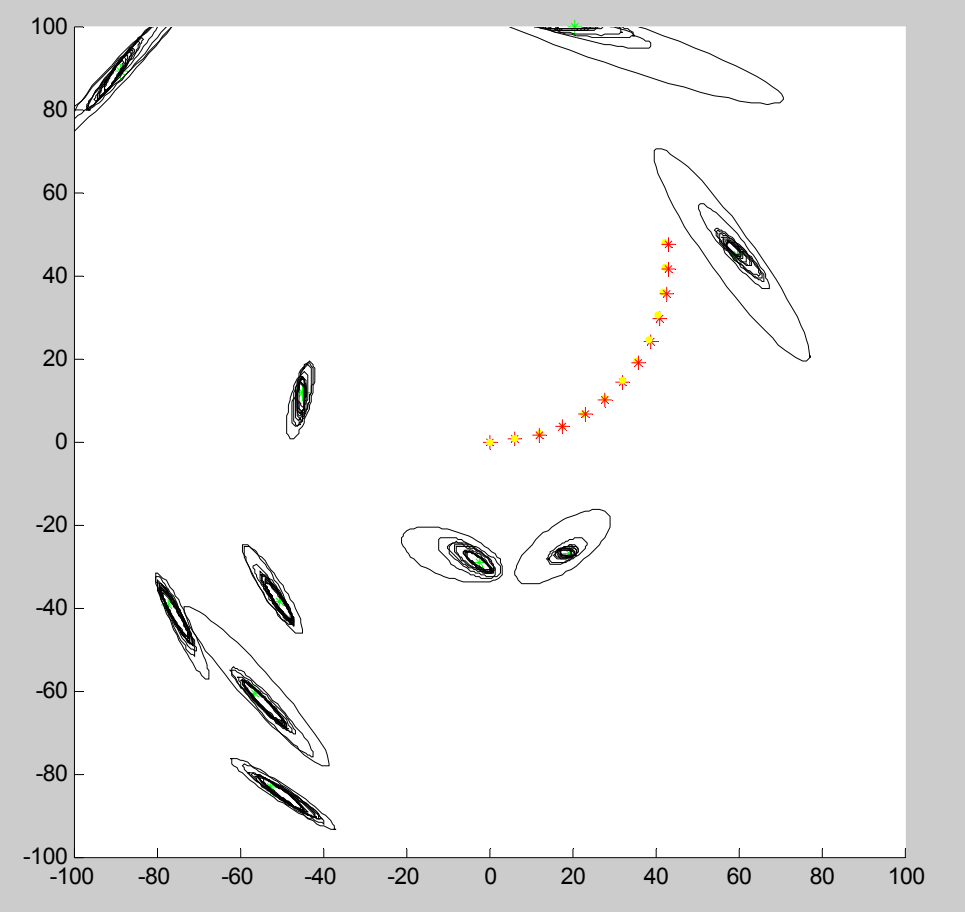

Figura 19: Evoluzione della stima della posizione del veicolo e dei landmarks in una simulazione SLAM

In modo analogo sono riportati successivamente i risultati relativi ad una seconda simulazione in cui sono state usate 400 particelle in modo da ottenere risultati maggiormente precisi.

k	errore localizzazione	distanza feature1	distanza feature2	distanza feature3	distanza feature4	distanza feature5	distanza feature6	distanza feature7	distanza feature8	distanza feature9	distanza feature10
50	0.1078	7.3680	3.0534	6.1145	4.5651	4.7668	1.3516	3.1368	0.9879	1.7339	1.5866
100	0.0969	5.1270	3.1149	1.5898	0.3110	1.1798	0.6045	2.3772	1.8533	0.7019	1.7544
150	0.1852	5.4874	2.2524	0.3267	1.8656	1.0535	0.7020	0.3430	3.3080	0.4006	0.3670
200	0.2115	4.7631	1.5949	0.7666	0.8664	0.2891	0.6124	0.5539	2.5384	0.3488	0.2695
250	0.4000	3.6434	1.0635	0.0453	0.1634	0.5967	1.2226	0.7592	1.0179	0.7052	0.2510
300	0.3951	2.4511	1.5527	0.3240	0.2568	0.5059	1.3182	0.3559	3.4858	1.4327	0.3227
350	0.6071	2.5495	0.7951	0.6154	0.8005	0.7246	1.2630	0.8006	4.0355	0.1389	0.5096
400	0.2694	0.9207	1.3537	0.4115	0.1530	0.3940	0.7715	0.2469	2.5483	0.2182	0.3728
450	0.2259	0.4644	0.8068	0.9144	0.1469	0.3254	0.4392	0.8822	1.9531	0.3167	0.2628
500	0.2987	0.3812	1.5142	0.4780	0.1992	0.3113	0.5471	0.5053	1.1825	0.6696	0.2455

Tabella 3: Risultati di una seconda simulazione nel problema SLAM

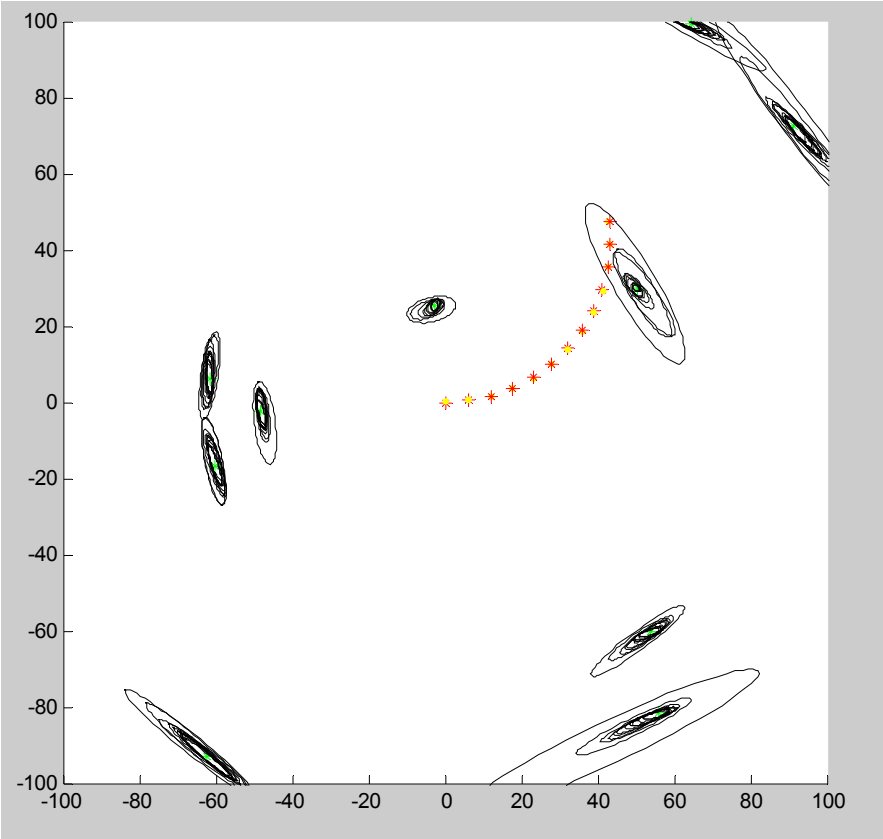

Figura 20: Grafico della posizione dei landmark e del veicolo in una seconda simulazione

Gli esempi trattati negli ultimi paragrafi hanno lo scopo di mostrare possibili applicazioni dei filtri a particelle in campi anche differenti fra di loro.

In generale il *Particle Filter Tool-box* proposto in questo lavoro di tesi si differenzia da altri tool-box di filtri a particelle che si trovano generalmente in letteratura in quanto permette di

trovare la funzione analitica della densità di probabilità a posteriori dello stato e di disegnare la forma di tale funzione. Ciò rende più facile la comprensione del fenomeno fisico e la lettura in chiave statistica dell'esperimento rispetto ad un istogramma di particelle. Come mostrano gli stessi istogrammi inoltre non è possibile di per sé intuire la probabilità di un particolare stato che non compaia direttamente in una delle particelle; al contrario una densità di probabilità è chiaramente più facile da leggere e da commentare.

Lo svantaggio di usare la ricostruzione della densità di probabilità consiste nel perdere tempo nel dover risolvere il sistema di equazioni differenziali che permette il calcolo dei moltiplicatori di Lagrange nel problema di approssimazione della stima a massima entropia; nonostante questo, specialmente nel caso in cui le particelle sono disposte in modo vario, il tempo di calcolo non è eccessivamente alto; ad esempio, nella terza simulazione, nonostante si debbano calcolare le densità di probabilità a posteriori di due stati, il tempo impiegato a simulare 60 passi di 1 s è stato di 50.683 s, rendendo quindi possibile un'eventuale stima in linea.

Se l'obiettivo principale per cui si usa il filtro a particelle è comunque proprio quello di stimare on-line, conviene usare il *Particle Filter Tool-box* disabilitando la funzione che calcola analiticamente la densità di probabilità a posteriori, ottenendo ovviamente prestazioni temporali assai migliori; questo è il caso dell'ultima simulazione che dovendo trattare con un alto numero di particelle di stato, ciascuna di dimensione 23, e con filtri di Kalman di dimensione 20, si è preferito non perdere ulteriore tempo nel calcolo delle densità di probabilità in maniera analitica.

Il seguente capitolo a tale proposito tralascia il problema del ricavarsi funzioni analitiche o grafiche della densità di probabilità a posteriori, e cerca di mostrare come alcune considerazioni che coinvolgono lo stesso concetto di entropia possano tornare utili per migliorare altri aspetti dell'algoritmo di stima.

4. MIGLIORAMENTO IN UN FILTRO A PARTICELLE

4.1 Passo MCMC

Nel capitolo precedente si è evidenziato come le fasi più importanti che costituiscono un filtro a particelle siano l'*inizializzazione*, la *predizione*, la *correzione*, il *ricampionamento* e il passo in cui eventualmente si calcola la *funzione analitica* della densità di probabilità a posteriori e la si disegna.

Il passo del ricampionamento, come spiegato in dettaglio nel paragrafo 2.3.2, è necessario per evitare la scomparsa della popolazione delle particelle dopo poche iterazioni; la maggior parte dei campioni assume infatti dei pesi così bassi da non poter più riuscire a contribuire alla rappresentazione della densità di probabilità. D'altro canto, durante la fase di ricampionamento, ogni campione con un peso d'importanza alto potrebbe essere duplicato così tante volte da dominare la nuvola delle nuove particelle; al limite il nuovo insieme di campioni potrebbe essere costituito da un'unica copia. In generale comunque, la perdita nel ricampionamento di quelle particelle che avevano un peso molto basso limita le capacità dell'algoritmo di cercare minimi migliori in altre zone. Ciò significa che se il numero delle particelle diverse fra loro è molto piccolo, il *particle filter* potrebbe trovarsi in difficoltà nel pronosticare un'evoluzione della densità di probabilità in una zona appartenente alle code della stessa.

Un strategia di forza bruta per superare questo inconveniente consiste ovviamente nell'aumentare il numero di particelle, in modo che sia generalmente più probabile di avere versioni diverse dello stato. Una strategia più raffinata consiste invece nell'implementare un ulteriore passo detto MCMC (Markov Chain Monte Carlo) dopo la fase di ricampionamento [2,4,10,27]. Le particelle al termine del resampling sono disposte in modo da approssimare la densità di probabilità a posteriori dello stato date le osservazioni; l'idea sulla quale si basa lo *MCMC step* consiste nell'applicare una transizione a catena di Markov all'intera popolazione di particelle in modo tale che queste risultino ancora in un insieme di campioni disposti come la funzione a posteriori d'interesse. Le nuove particelle però potrebbero essere state spostate in aree più interessanti dello spazio degli stati; alcuni esempi di MCMC sono il *campionatore di Gibbs* e gli *algoritmi di Metropolis Hastings*.

Ad esempio un'implementazione dell'algoritmo di Metropolis-Hasting può consistere nello scegliere un campione x_k^{*i} dalla distribuzione $p\left(x_k \middle| x_{k-1}^i\right)$; tale campione, indicato con l'asterisco, è il campione candidato a sostituire il campione $\tilde{x}_k^i$ già presente nella nuvola di particelle ricampionate. La scelta se tale campione entrerà o meno a far parte dell'insieme

ricampionato sostituendo la particella precedente viene effettuata confrontando la likelihood nel caso di x_k^{*i} e $\tilde{x}_k^i$. Se la nuova likelihood è maggiore il campione candidato sostituisce quello già presente; se è minore viene accettato con probabilità pari al rapporto delle likelihood.

L'algoritmo può essere così formulato con uno pseudo-codice:

for $i = 1 : N_s$

%% Estrarre un campione v da una distribuzione uniforme $U[0,1]$ *%%*

%% Estrarre un candidato campione x_k^{*i} *dalla distribuzione* $p\left(x_k \middle| x_{k-1}^i\right)$ *%%*

if $\left(v \le \min\left\{1, \dfrac{p\left(y_k \middle| x_k^{*i}\right)}{p\left(y_k \middle| \tilde{x}_k^i\right)}\right\}\right)$ *then*

accetta il campione candidato: $x_k^i = x_k^{*i}$

else

rifiuta il campione candidato: $x_k^i = \tilde{x}_k^i$

end-if

end-for

Lo scopo quindi del passo MCMC consiste nello scegliere un nuovo insieme di N_s particelle, alcune delle quali già presenti nel vecchio insieme ricampionato, disposte come la densità di probabilità a posteriori, ma scelte in modo da poter meglio investigare lo spazio degli stati. Lo svantaggio di tale mossa è ovviamente di tipo computazionale in quanto richiede sia nuovi campionamenti dalla densità proposal, sia un secondo ricampionamento basato su un meccanismo di accettazione/rifiuto. Nel prossimo paragrafo viene presentato un nuovo algoritmo di ricampionamento che sfrutta quelli già introdotti nel paragrafo 2.3.2, ma introducendo un nuovo metro di giudizio basato sull'entropia. Sfruttando il concetto di entropia si cerca di perseguire una strada analoga a quello del criterio MCMC.

4.2 Ricampionamento basato su un criterio ad entropia

Gli algoritmi di resampling presentati nel paragrafo 2.3.2 affrontano in modo particolare il problema della varianza delle particelle al termine del passo; se le particelle fossero estratte infatti dalla vera distribuzione a posteriori dello stato note le osservazioni, i loro pesi sarebbero tutti uguali fra loro e la varianza nulla. Bassi valori di varianza dei pesi possono quindi essere

indicatori della bontà della stima, e in questo contesto l'indicatore $\overline{N_{eff}}$, introdotto nel paragrafo 2.3.2, viene spesso usato per valutare la necessità o meno di un ricampionamento.

Un altro criterio per giudicare il livello di affidabilità di un algoritmo di stima può essere la valutazione dell'entropia delle particelle al termine della fase stessa di ricampionamento; a tale proposito i grafici della funzione densità di probabilità, ad esempio quelli di Figg. 17 e 18, erano stati usati proprio per giudicare la correttezza della stima.

Una densità di probabilità dello stato con un picco molto stretto indica infatti una certa sicurezza nell'individuazione dello stato; una densità di probabilità molto larga è indice al contrario di maggiore incertezza nella stima. Al limite la densità di probabilità a posteriori nel primo disegno di Fig.17 era piatta, ad indicare come lo stato poteva assumere un qualsiasi valore tra 0 e 500 con la stessa probabilità.

Un criterio ad entropia quindi può essere usato come indice per la valutazione dell'incertezza al termine di ciascuna stima, come proposto anche in [20]; nonostante ad un dato passo sia noto solamente un insieme di campioni della densità di probabilità a posteriori e non la densità stessa, il modo più diretto per calcolare l'entropia consiste nell'usare direttamente la formula (55), di seguito ricordata per comodità:

$$H^*(x) = -\sum_{i=1}^{N} p_i \log p_i \qquad (55).$$

Chiaramente non è detto che $H^*(x)$ sia uguale all'entropia vera $H(x)$ della distribuzione di probabilità, in quanto il numero di campioni usati per l'approssimazione è finito; in letteratura sono suggeriti approcci per il calcolo dell'entropia basati su finestre, ma sembra che non ci siano differenze sostanziali nel caso in cui le particelle siano come in genere $50 \approx 200$ [20]. Sempre in letteratura il criterio ad entropia viene suggerito come semplice indicatore di quanto la stima stia andando bene o meno; sembra quindi immediato affermare che minore è l'entropia delle particelle al termine del ricampionamento migliore è la stima. Nel caso però di un filtro a particelle non sempre un'entropia minima è sinonimo di stima ottimale; infatti il ricampionamento permette la copia di particelle con pesi d'importanza alti e ferma la propagazione delle particelle con pesi bassi: in generale il resampling non fa altro quindi che diminuire l'entropia della nuvola di stati. Lo svantaggio del ricampionamento è però proprio quello di eliminare quelle particelle che si trovano nella coda della densità di probabilità a posteriori, impoverendo la varietà di stati campione a favore di quelli ad alta importanza; a tale

proposito il passo MCMC ha proprio l'obiettivo di andare a recuperare quelle particelle che garantiscono una migliore rappresentazione totale dello spazio degli stati, fermo rimanendo che siano ancora disposte come la densità di probabilità a posteriori. Un'alta entropia delle particelle al termine del ricampionamento può quindi essere un vantaggio dato che permette una varietà maggiore di campioni e quindi una migliore copertura dello spazio degli stati.

L'idea quindi di questo paragrafo è quella di introdurre un resampling-dinamico basato sull'entropia delle particelle al termine del ricampionamento; ad ogni passo vengono quindi trovate le nuvole di particelle proposte da alcuni algoritmi di ricampionamento, ad esempio già disponibili in letteratura; la nuvola di particelle da propagare effettivamente corrisponde però a quella in cui l'entropia dei campioni risulta maggiore o minore rispetto alle altre.

Gli algoritmi di resampling *residual*, *systematic* e *sampling-importance* presentati nel paragrafo 2.3.3 offrono quindi ad ogni iterazione del filtro degli insiemi diversi di campioni, ma tutti disposti come approssimazione della densità di probabilità a posteriori dello stato. Il resampling-dinamico consiste nell'usare ad ogni passo quel particolare algoritmo di resampling che garantisce una maggiore entropia, ottenendo quindi un resampling-dinamico a massima entropia, o una minore entropia, ottenendo quindi un resampling-dinamico a minima entropia. L'algoritmo di ricampionamento dinamico a minima entropia può quindi essere schematizzato tramite il seguente algoritmo implementato in uno pseudo-codice:

%% Considerare il vettore delle particelle predetto ***xparticle_Pred*** *%%*
%% Effettuare su ***xparticlePred*** *un ricampionamento con l'algoritmo multinomial resampling %%*
%% Effettuare su ***xparticlePred*** *un ricampionamento con l'algoritmo residual resampling %%*
%% Effettuare su ***xparticlePred*** *un ricampionamento con l'algoritmo systematic resampling %%*
%% Scegliere come nuovo vettore di stato l'insieme ricampionato a minima entropia %%

La precedente procedura va quindi effettuata ad ogni passo di iterazione del filtro, durante la fase di ricampionamento; l'idea generale è che se la stima sta funzionando bene un algoritmo a minima entropia potrebbe essere il migliore, dato che se la predizione è giusta, può essere ragionevole mantenere il minimo numero di versioni dello stato possibile. Al contrario, nel caso più generale, la stima a massima entropia è più ragionevole perché è in grado di mantenere una maggiore varietà di versioni dello stato e quindi di prevedere la possibilità dello stato di evolvere in un maggior numero di direzioni dello spazio degli stati.

Prima di effettuare simulazioni ci si aspetta quindi che un criterio a minima entropia possa dare risultati migliori nel caso di sistemi in cui l'algoritmo di stima è particolarmente efficace; un criterio a massima entropia potrebbe essere conveniente invece in sistemi particolarmente rumorosi o nei casi in cui la stima non è molto aderente alla realtà, come ad esempio in filtri a particelle in cui si usa un numero ridotto di particelle. Nel prossimo paragrafo sono mostrati i risultati delle simulazioni condotte per verificare queste ipotesi.

4.3 Simulazioni per il confronto tra un ricampionamento dinamico e algoritmi di resampling standard

In questo paragrafo sono riportati gli esiti delle simulazioni effettuate per confrontare le prestazioni degli algoritmi di ricampionamento dinamico a massima e minima entropia con gli algoritmi di resampling standard introdotti nel paragrafo 2.3.2. Come spiegato nel precedente paragrafo gli algoritmi di ricampionamento dinamico sfruttano ad ogni passo uno degli altri algoritmi di resampling standard; dato che le differenze di prestazioni tra questi sono molto piccole, anche i risultati degli algoritmi dinamici sono molto vicini a quelli degli altri algoritmi.

Inoltre la scelta ad ogni passo dell'algoritmo che fornisce l'insieme di particelle ad esempio a minima entropia non garantisce che al termine della simulazione si ottenga un'entropia totale minima; la Fig.19 mostra ad esempio come ad un determinato passo potrebbe convenire scegliere un algoritmo non a minima entropia per ottenere al termine un'entropia minore.

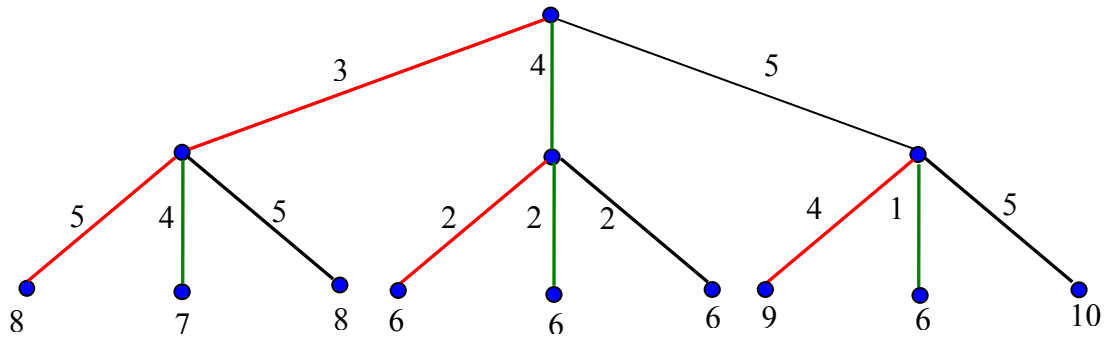

Fig. 21: Scelta del ricampionamento con il criterio a minima entropia

Nella Fig.19 il nodo in alto simboleggia il passo 0, ogni arco rappresenta la scelta di uno dei tre algoritmi di ricampionamento e i nodi in fondo rappresentano il nuovo vettore di stato a seconda dell'algoritmo di ricampionamento scelto ad ogni passo. Se i valori sugli archi rappresentano l'entropia dell'insieme delle particelle il criterio a minima entropia sceglierebbe il primo

algoritmo di resampling al primo passo e il secondo algoritmo al secondo passo. In totale avrebbe quindi un'entropia di 7 e dalla Fig.19 si può notare come diverse scelte dell'algoritmo di resampling da usare a ciascun passo avrebbero in tutto permesso di ottenere anche entropie minori. In particolare già l'uso solamente del secondo algoritmo di resampling a tutti i passi avrebbe già garantito un'entropia totale minore. La non garanzia quindi di ottenere al termine dell'evoluzione di un sistema l'entropia massima o minima come si vorrebbe, e le performance molto vicine dei vari algoritmi di resampling non permettono quindi di differenziare troppo gli algoritmi a ricampionamento dinamico da quelli a ricampionamento fisso.

I due nuovi metodi di resampling sono comunque stati provati in tutti e tre gli esempi introdotti nel paragrafo 3.3, sia nel caso standard, sia nel caso in cui il numero delle particelle usate fosse stato di un ordine di grandezza minore. Gli esperimenti effettuati per ciascuna simulazione sono stati in genere molti in modo da avere statistiche maggiormente affidabili.

4.3.1 Esempio n.1

Il primo esperimento è stato condotto sull'esempio presentato nel paragrafo 3.3.1, caratterizzato dalle equazioni (52) e (53) ricordate di seguito:

$$x_{k+1} = 1 + \sin(\pi\omega k) + \phi_1 x_k + v_k \tag{52}$$

$$y_k = \begin{cases} \phi_2 x_k^2 + v_k & t \le 30 \\ \phi_3 x_k - 2 + n_k & t > 30 \end{cases} \tag{53};$$

allo scopo di trarre statistiche sufficientemente significative sono state effettuate 2000 simulazioni, ciascuna riguardante una evoluzione dello stato di 60 passi; il numero di particelle usate per la descrizione del sistema è stato 200, un numero quindi standard per un problema caratterizzato da un solo stato. Dapprima è riportato il grafico relativo all'ultima simulazione per mostare l'andamento dello stato reale, delle osservazioni e delle stime condotte da filtri a particelle che usavano algoritmi di ricampionamento diversi. Le abbreviazioni usate nella legenda sono per:

- PFS: Particle Filter con Systematic Resampling
- PFM: Particle Filter con Multinomial (o Sampling-Importance) Resampling
- PFR: Particle Filter con Residual Resampling

- PFGmax: Particle Filter con Resampling Dinamico a Massima Entropia
- PFGmin: Particle Filter con Resampling Dinamico a Minima Entropia

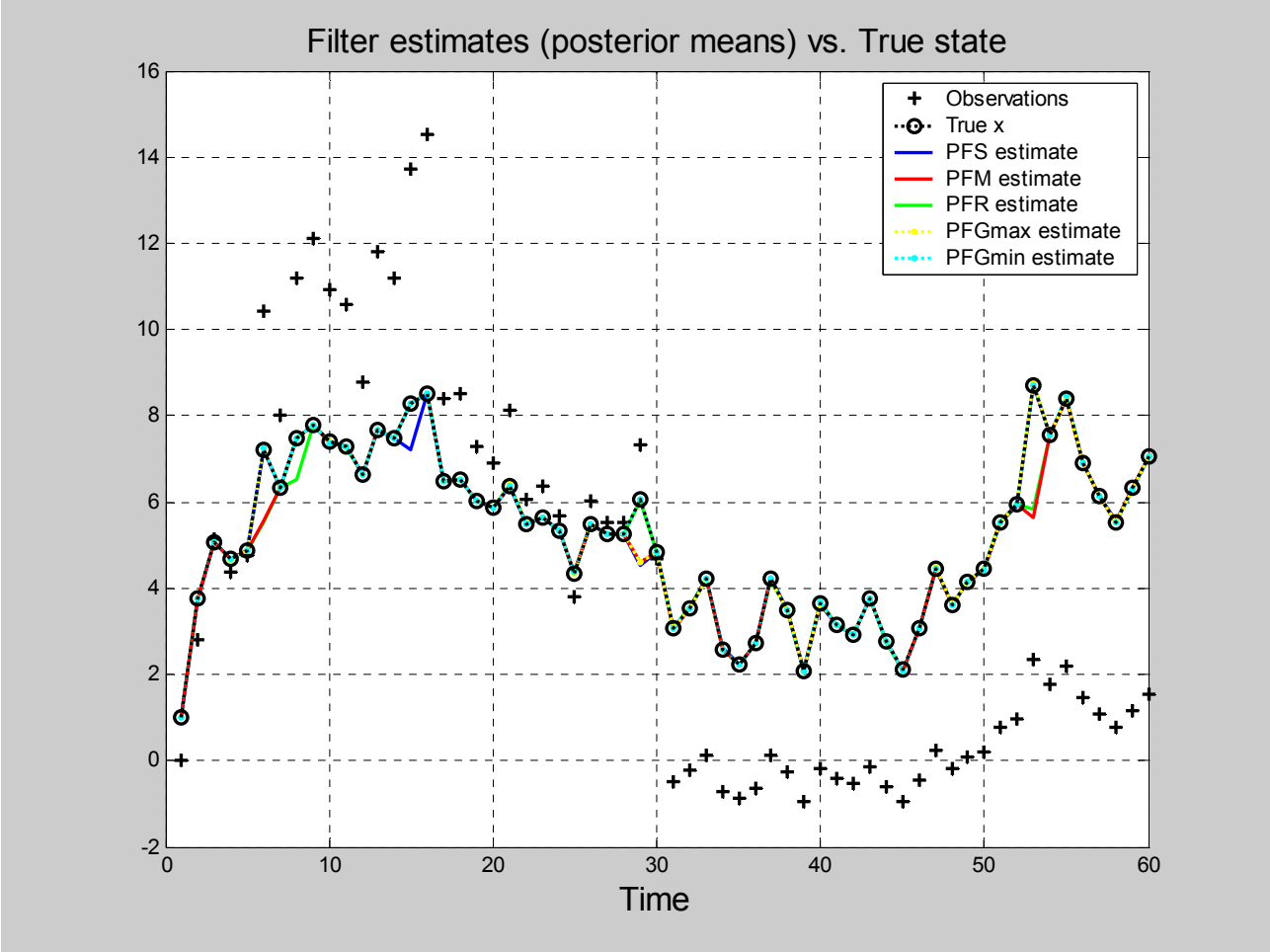

Fig. 22: Andamento di una simulazione per il confronto di filtri a particelle con diversi algoritmi di ricampionamento; esempio 1

La tabella successiva mostra invece le statistiche relative alla simulazione, in particolare l'errore quadratico medio, la varianza, l'errore percentuale, l'entropia totale e il tempo di calcolo medio relativo ad un intero esperimento di 60 passi. In blu sono stati evidenziati i peggiori risultati in ogni categoria, mentre i migliori sono stati evidenziati in giallo.

	RMSE	Varianza	Errore percentuale	Entropia	Tempo di calcolo (s)
Residual resampling	0.43175	0.048257	1.5283%	119602.0191	3.5449
Systematic resampling	0.4285	0.047758	1.4962%	118529.3809	3.5466
Multinomial resampling	0.42639	0.051158	1.5043%	113005.0473	3.5633
Max. entropy resampling	0.42766	0.047574	1.4908%	120578.5225	3.7381
Min. entropy resampling	0.4265	0.04978	1.5052%	111196.3655	3.737

Tabella 4: Risultati di simulazioni con un numero adeguato di particelle nell'esempio 1

Come si può notare dalla Tabella 4, le stime sono generalmente molto vicine ai valori reali dello stato; in queste condizione ci si aspettava che un algoritmo di stima dinamico a minima entropia potesse dare buoni risultati, difatti risulta il secondo migliore come RMSE. Gli errori percentuali sono meno significativi dato che un algoritmo può risultare migliore di un altro grazie al fatto che ha varianza minore o più spesso perché commette errori minori quando lo stato assume valori più piccoli ed errore maggiori quando lo stato assume valori più grandi; l'errore relativo percentuale non è quindi di per sé un indice molto significativo. Si può notare come gli algoritmi di ricampionamento dinamico a massima e minima entropia riescano al termine della simulazione ad avere realmente un entropia rispettivamente maggiore e minore e come il loro tempo computazionale sia più alto. Infatti gli algoritmi hanno sempre complessità $O(N)$, ma richiedono più tempo dato che ripetono tre volte lo stesso ciclo effettuato una sola volta dagli altri algoritmi. Il tempo di calcolo si riferisce ad un'intera simulazione di 60 passi, si può quindi notare che è molto più basso rispetto a quando si calcolava anche la funzione analitica della densità di probabilità a posteriori.

Nel secondo esperimento sono state effettuate 5000 simulazioni con un numero di particelle pari a 40; 40 versioni dello stato non sono un numero particolarmente alto per poter bene caratterizzare la densità di probabilità a posteriori dello stato.

	RMSE	Varianza	Errore percentuale	Entropia	Tempo di calcolo (s)
Residual resampling	0.69225	0.034827	5.3266%	377431.1165	0.82403
Systematic resampling	0.68991	0.034501	5.3289%	439386.6786	0.82115
Multinomial resampling	0.69448	0.035111	5.3747%	380527.6583	0.82764
Max. entropy resampling	0.68816	0.033513	5.2986%	441685.1565	0.8879
Min. entropy resampling	0.69807	0.03535	5.3543%	369875.1922	0.887

Tabella 5: Risultati di simulazioni con un numero basso di particelle nell'esempio 1

Come ci si aspettava i Round Mean Square Errors aumentano, dato che si sta usando un minor numero di particelle; anche l'errore percentuale medio circa triplica, ma in compenso il tempo di computazione diventa un quarto. In questo caso, essendo la stima poco corretta ci si aspettava che il filtro basato sull'algoritmo a massima entropia desse buoni risultati, ed in effetti fornisce le migliori stime; inoltre il criterio a minima entropia dà i peggiori risultati, come ci si poteva aspettare dato che il basso numero di particelle unito al fatto che l'entropia bassa implica che molte di queste particelle sono anche uguali fra loro, non permette un'adeguata rappresentazione della densità di probabilità a posteriori dello stato.

4.3.2 Esempio n.2

Il secondo esperimento è stato condotto sul sistema dinamico presentato nell'esempio del paragrafo 3.3.2, ed è caratterizzato dalle equazioni (72) e (73) successivamente ricordate:

$$x_k = \frac{x_{k-1}}{2} + \frac{25x_{k-1}}{1+x_{k-1}^2} + 8\cos(1.2k) + v_{k-1} \tag{72}$$

$$y_k = \frac{x^2}{20} + n_k \tag{73};$$

le alte varianze del rumore del processo e di misurazione unite alle consistenti non-linearità portano lo stato del sistema ad avere rapide variazioni.

Il primo esperimento effettuato sfrutta 1000 simulazioni effettuate con 200 particelle ed un'evoluzione dello stato di 20 passi; il numero di particelle è quindi piuttosto standard per un sistema ad un solo stato ed il numero delle simulazioni è di nuovo alto in modo da ottenere stime maggiormente significative.

Dapprima in Fig.23 si riporta un grafico che mostra l'andamento dello stato reale, delle osservazioni e delle stime ottenute usando diversi algoritmi di ricampionamento; la simulazione della Fig.23 utilizza lo stesso 200 particelle, ma è relativa ad una simulazione che prevede un'evoluzione dello stato di 60 passi. La legenda usata per gli algoritmi di stima usati è la stessa del sottoparagrafo precedente;

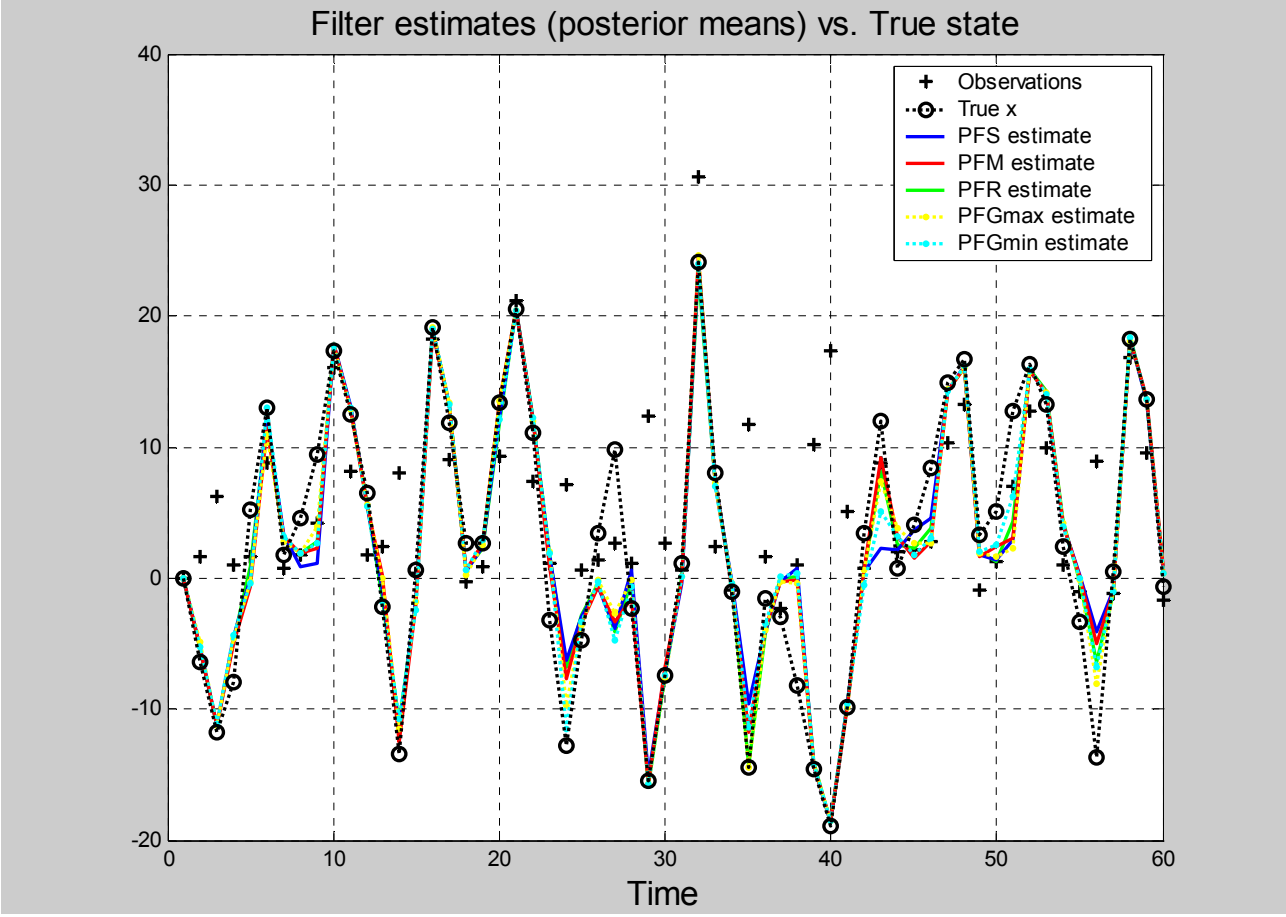

Fig. 23: Andamento di una simulazione per il confronto di filtri a particelle con diversi algoritmi di ricampionamento; esempio 2

	RMSE	Varianza	Errore percentuale	Entropia	Tempo di calcolo (s)
Residual resampling	4.0413	3.0157	113.5261%	109637.9648	0.68098
Systematic resampling	4.1164	3.3022	104.1345%	110918.8685	0.68297
Multinomial resampling	4.1199	3.4108	120.0554%	105289.4037	0.691
Max. entropy resampling	4.0518	2.8307	123.1176%	111160.1986	0.75176
Min. entropy resampling	4.1281	3.5249	102.6546%	105257.3309	0.75324

Tabella 6: Risultati di simulazioni con un numero adeguato di particelle nell'esempio 2

La tabella 6 mostra i risultati statistici e, detto già nel paragrafo 3.3.2 del perché gli errori percentuali siano così elevati, è bene soffermarsi sui RMSE; il sistema è molto rumoroso e ha dinamiche molto veloci, come si vede infatti dalla Fig.23 lo stato ha cambiamenti repentini di valore; i RMSE stessi sono molto elevati rispetto al precedente esperimento. In tali condizioni ci si aspetta che una stima a massima entropia dia buoni risultati, mentre sia più povera una stima a minima entropia.

	RMSE	Varianza	Errore percentuale	Entropia	Tempo di calcolo (s)
Residual resampling	4.538	5.6274	213.4211%	645409.1315	0.20361
Systematic resampling	4.5366	5.4102	172.8417%	656792.4489	0.2002
Multinomial resampling	4.5847	5.5682	415.8624%	604918.2916	0.20189
Max. entropy resampling	4.4953	5.1741	201.0907%	664841.5541	0.22817
Min. entropy resampling	4.614	5.8604	189.8223%	600590.3097	0.22759

Tabella 7: Risultati di simulazioni con un numero basso di particelle nell'esempio 2

La tabella 6 conferma questa ipotesi dato che il filtro a particelle che sfrutta l'algoritmo a massima entropia fornisce la seconda migliore stima, mentre quello a minima entropia fornisce la peggiore; analogamente al paragrafo precedente si riportano ora i risultati relativi a filtri a particelle in cui si è diminuito drasticamente il numero di particelle da 200 a 40. I nuovo risultati sono riportati nella tabella 7.
In questo caso è evidente come le migliori e le peggiori prestazioni siano appannaggio del filtro a particelle che sfrutta rispettivamente l'algoritmo di ricampionamento a massima e minima entropia. Inoltre la seconda migliore prestazione è fornita dall'algoritmo che fornisce la seconda entropia più alta, mentre la seconda peggiore performance è fornita dall'algoritmo che fornisce la seconda entropia più bassa. Queste tabelle rinforzano quindi l'idea che l'entropia possa essere un buon indicatore da usare in un algoritmo di stima. Ovviamente i tempi di calcolo, anche se molto vicini tra loro, sono peggiori nel caso dei filtri a massima e minima entropia.

4.3.3 Esempio n.3

Il terzo esperimento condotto per trarre delle statistiche è quello effettuato nel problema di tracking semplificato. Il passive tracking problem è lo stesso esempio presentato nel paragrafo 3.3.3 e le sue equazioni del modello sono successivamente ricordate per comodità:

$$\bar{x}_{k+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \bar{x}_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v_k \tag{74},$$

$$\begin{cases} x_k^p = 4k \\ y_k^p = 20 \end{cases} \tag{75},$$

$$y_k = \tan^{-1}\left(\frac{y_k^p}{x_{1k} - x_k^p} \right) + w_k \tag{76}.$$

L'equazione (74) rappresenta l'evoluzione dello stato del target, l'equazione (75) è l'equazione di moto del sensore, l'equazione (76) infine rappresenta il modello di acquisizione dell'osservazione, il *line of sight angle*; in questo esempio si ricorda che la posizione e la velocità iniziali del target erano note sotto forma di densità di probabilità uniformi.

Il primo esperimento sfrutta 1000 simulazioni effettuate usando 500 particelle, un numero piuttosto alto quindi di campioni, ma onesto considerando che lo stato ha dimensione 2 e il sistema è molto rumoroso. Inoltre si sta usando un solo sensore per ricavare due grandezze cinematiche del target quali la posizione e la velocità; gli errori nella posizione sono ovviamente molto maggiori a quelli commessi nella stima della velocità, dato che piccoli errori di stima di velocità si amplificano al momento di calcolare la nuova posizione.

Dapprima in Fig.24 è mostrato l'esito di una singola simulazione in cui vengono usati cinque diversi filtri a particelle, dove ognuno si differenzia dagli altri in base al tipo di ricampionamento sfruttato. Lo stato è stato lasciato evolvere per 50 passi, mentre la legenda usata è la stessa dei due esempi precedenti.

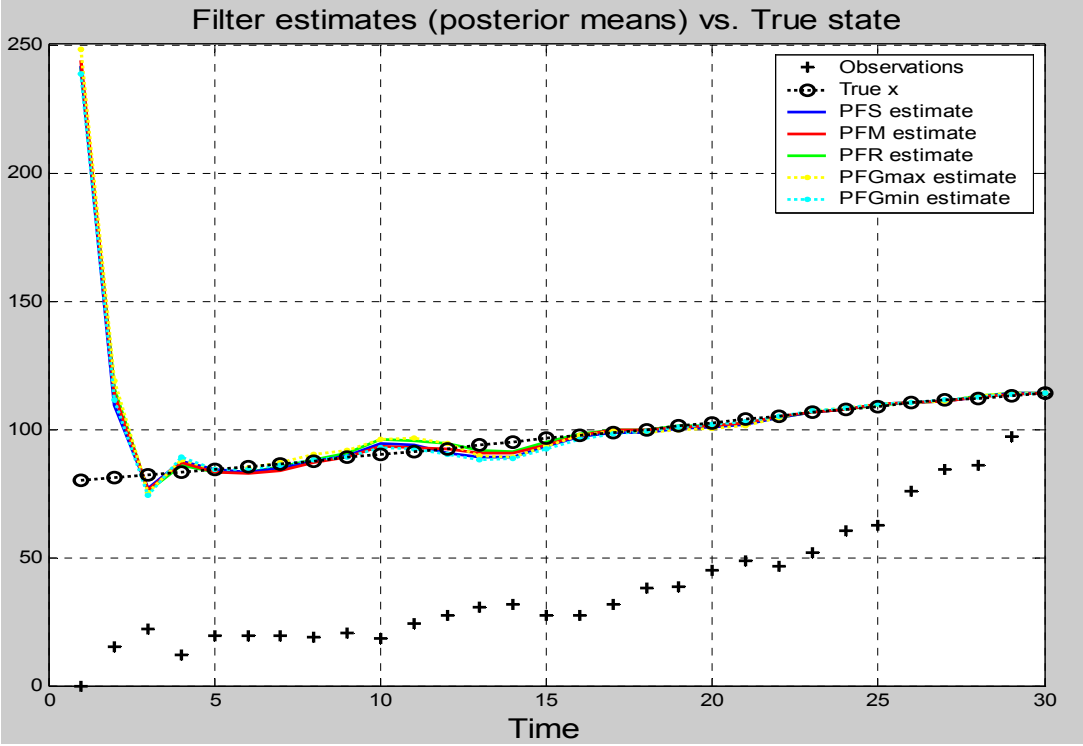

Fig. 24: Andamento di una simulazione per il confronto di filtri a particelle con diversi algoritmi di ricampionamento; esempio 3

La tabella 8 mostra invece le statistiche ricavate dalle 1000 simulazioni, ancora una volta le migliori prestazioni sono evidenziate in giallo, mentre le peggiori in blu.

	RMSE	Varianza	Errore percentuale	Entropia	Tempo di calcolo (s)
Residual resampling	12.5906	90.4826	6.5352%	217556.4799	3.1255
Systematic resampling	12.6108	90.6309	6.53%	225289.0695	3.1049
Multinomial resampling	12.6343	91.4862	6.6383%	203865.568	3.0905
Maximum entropy resampling	12.6416	91.4811	6.55%	225367.1269	3.3441
Minimum entropy resampling	12.6022	89.3009	6.6656%	203780.3454	3.3562

Tabella 8: Risultati di simulazioni con un numero adeguato di particelle nell'esempio 3

In questo caso il sistema è ben caratterizzato dal numero di particelle usate nel filtro, ci si aspetta quindi che le stime siano molto vicine ai valori reali, come evidenzia la stessa Fig.24. In realtà i RMSE sono piuttosto alti, ma questo è dovuto essenzialmente al fatto che sono calcolati a partire dall'istante 2, e perciò risentono degli errori di stima compiuti all'inizio dove l'incertezza sulla effettiva posizione del target era massima.

In queste condizioni ci si aspetta quindi che l'algoritmo a minima entropia fornisca buone prestazioni, e così è infatti dato che garantisce le seconde migliori stime. L'algoritmo a massima entropia in questo caso fornisce al contrario le peggiori stime.

Anche in questo esempio si è provato infine a diminuire il numero delle particelle a 50 per verificare il nuovo comportamento dei vari filtri. In questo nuovo esempio ci si aspettava che il numero basso di particelle usato aiutasse di nuovo il filtro basato sull'algoritmo a massima entropia ad ottenere le migliori prestazioni e penalizzasse al contrario il filtro a minima entropia. In realtà il ricampionamento a massima entropia è secondo e quello a minima è penultimo, come si può vedere nella tabella 9.

	RMSE	Varianza	Errore percentuale	Entropia	Tempo di calcolo (s)
Residual resampling	17.9116	137.4507	12.9076%	598217.8403	0.35491
Systematic resampling	17.5409	138.1893	12.473%	635186.7123	0.35159
Multinomial resampling	19.0566	168.8425	14.4288%	535287.0279	0.35354
Maximum entropy resampling	17.708	135.4198	12.3402%	634749.5423	0.38975
Minimum entropy resampling	18.5929	158.4444	14.0813%	537632.0847	0.38722

Tabella 9: Risultati di simulazioni con un numero basso di particelle nell'esempio 2

E' curioso però notare che ciò è dovuto al fatto che il systematic resampling è riuscito ad ottenere in questa simulazione non solo i migliori risultati ma anche l'entropia totale massima. Analogamente il multinomial resampling, o sampling-importance resampling, è riuscito sia a fornire i peggiori risultati sia a minimizzare l'entropia totale. Anche se quindi in questa occasione il filtro a particelle basato sull'algoritmo di resampling non è riuscito ad ottenere i migliori risultati, l'esperimento ha confermato come nelle situazioni particolarmente difficili di stima l'avere una grande entropia è evidentemente un aiuto.

4.3.4 Esempio n.4

L'ultima simulazione effettuata riguarda il problema del veicolo mobile che si trova in un ambiente ignoto anche se, al contrario del capitolo precedente, non viene esaminato l'intero problema SLAM, ma l'attenzione è rivolta solamente al problema di localizzazione.

Si suppone quindi che siano note perfettamente le coordinate dei vari landmark e, tramite il modello di predizione dello stato basato su controllo odometrico e il modello di misurazione della posizione relativa del robot rispetto alle features, si cerca di ricostruire il punto esatto in cui si trova il veicolo; ovviamente si suppone nuovamente che il robot mobile sia in movimento e che ci sia rumore sia nel controllo odometrico che nelle varie osservazioni.

Lo studio del solo problema di localizzazione rende più facile la realizzazione di simulazioni per verificare le performance del ricampionamento a massima e minima entropia rispetto agli altri algoritmi di ricampionamento in quanto la dimensione del vettore di stato è ridotta solamente a 3; inoltre la trattazione è analoga in quanto anche nel problema SLAM completo il filtro a particelle veniva usato per risolvere il sottoproblema di localizzazione, anche se in quel caso il vettore di stato aveva dimensione assai maggiore. Le equazioni (77) e (79), di seguito ricordate per comodità, regolano l'evoluzione del modello di aggiornamento dello stato e di misurazione dell'uscita

$$\bar{x}_{k+1} = \bar{x}_k \oplus \left[0 \quad 0.15 \quad \frac{0.1\pi}{180} \right]^T + \bar{v}_k \qquad (77),$$

$$\bar{z} = \left[\left\| \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T - \begin{bmatrix} x_j & x_{j+1} \end{bmatrix}^T \right\| \; ; \quad AngleWrap\left(Atan2\left(\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T - \begin{bmatrix} x_j & x_{j+1} \end{bmatrix}^T \right) - x_3 \right) \right] (79).$$

Anche in questo caso sono state effettuate più simulazioni con un diverso numero di particelle per verificare le prestazioni dei diversi algoritmi di resampling. La prima simulazione è stata effettuata utilizzando 200 particelle e la durata della simulazione è di 5000 passi, ed i risultati sono riportati nella tabella 10.

	Errore Assoluto Medio	Errore Relativo Medio Percentuale	Entropia
Residual resampling	0.27864	0.78024 %	6.9808
Systematic resampling	0.28231	0.78999 %	7.0655
Multinomial resampling	0.40269	1.0548 %	6.6134
Maximum entropy resampling	0.25653	0.68093 %	7.1664
Minimum entropy resampling	0.40934	1.1229 %	6.6132

Tabella 10: Errori in una simulazione di problema di localizzazione usando 200 particelle

Nella tabella compaiono statistiche relative all'errore assoluto e all'errore relativo percentuale nella determinazione della posizione del robot mobile, e l'entropia media ottenuta nella disposizione delle particelle con ciascun metodo di ricampionamento.
Nel primo esperimento si può notare come gli algoritmi a massima e minima entropia diano rispettivamente i migliori e i peggiori risultati; inoltre il metodo di ricampionamento multinomial dà risultati in termini di entropia molto vicini a quelli del metodo a minima entropia e difatti le sue prestazioni sono le seconde peggiori tra tutti gli altri metodi di resampling.

La seconda simulazione sfrutta un minore numero di particelle pari a 50 e il numero dei passi è 10000.

	Errore Assoluto Medio	Errore Relativo Medio Percentuale	Entropia
Residual resampling	0.46581	1.9873 %	4.9387
Systematic resampling	0.47062	2.0054 %	5.0232
Multinomial resampling	0.55036	2.3051 %	4.5903
Maximum entropy resampling	0.43876	1.772 %	5.1236
Minimum entropy resampling	0.6008	2.2376 %	4.5849

Tabella 11: Errori in una simulazione di problema di localizzazione usando 50 particelle

E' interessante notare come il metodo a minima entropia dia di nuovo il peggior risultato in termini di errore assoluto medio mentre il metodo a massima entropia comporti il miglior risultato; accade nuovamente inoltre che il metodo di ricampionamento multinomial, con un'entropia molto vicina alla minima, comporti il peggior errore relativo percentuale medio e il secondo peggior errore medio assoluto. Una cosa interessante che si può notare è che l'errore assoluto che viene commesso usando il metodo di ricampionamento a massima entropia con 50 particelle è paragonabile a quello che viene commesso dall'algoritmo di multinomial resampling

usando 200 particelle. Una conclusione evidente è che in alcune circostanze è possibile usare l'algoritmo di ricampionamento a massima entropia e diminuire drasticamente il numero di particelle usate in modo da avere prestazione molto più veloci. In alcune applicazioni tipo quella SLAM, dove il numero di particelle richieste dovrebbe essere altissimo in quanto il vettore di stato è tutt'altro che unidimensinale, la possibilità di più che dimezzare il numero di particelle da usare e mantenere quasi inalterate le prestazioni è senza dubbio un grande vantaggio.

Un'ultima simulazione è stata effettuata sempre a riguardo del problema di localizzazione, ma in questo caso i passi sono stati di nuovo 10000 e il numero delle particelle usato è salito a 500. I risultati sono di nuovo riportati di seguito in una tabella analoga. Anche in questo caso si può notare come le migliori prestazioni vengano offerte da un algoritmo di ricampionamento a massima entropia mentre le peggiori performance sono relative all'utilizzo di un minimum entropy resampling. Di nuovo l'algoritmo multinomial resampling ha sia valori di entropia molto vicini a quelli del minimum entropy e le sue prestazioni sono le seconde peggiori. I migliori risultati sono ancora evidenziati in giallo, mentre i peggiori sono in azzurro.

	Errore Assoluto Medio	Errore Relativo Medio Percentuale	Entropia
Residual resampling	0.21999	0.8804 %	8.3165
Systematic resampling	0.22341	0.90532	8.4083
Multinomial resampling	0.21303	0.96984 %	7.9486
Maximum entropy resampling	0.20067	0.87962 %	8.5065
Minimum entropy resampling	0.22374	0.97191 %	7.9473

Tabella 12: Errori in una simulazione di problema di localizzazione usando 500 particelle

CONCLUSIONI

Il lavoro di tesi affronta il problema della determinazione di una stima della densità di probabilità dello stato in un sistema dinamico tempo discreto. L'obiettivo è stato raggiunto mediante l'utilizzo di un filtro a particelle in modo da ottenere ad ogni iterazione un insieme di campioni disposti ad approssimazione della densità di probabilità incognita condizionata alle informazioni provenienti dalle osservazioni effettuate. L'espressione analitica della densità di probabilità dello stato viene calcolata sfruttando il principio a massima entropia, mentre una stima dello stato del sistema può essere ottenuta semplicemente calcolando il valore medio delle particelle.

Possibili sviluppi di questo lavoro di tesi possono generalmente seguire due linee generali, la prima riguarda il miglioramento del filtro a particelle nella sua globalità, la seconda lo studio di una particolare applicazione in modo da migliorare le prestazioni del particle filter in quella situazione. Il miglioramento del filtro a particelle in generale può riguardare un approfondimento del semplice particle filter tool-box già presentato, e l'implementazione di nuove funzioni in modo da rendere più completo l'insieme delle applicazioni risolvibili. Una seconda linea di sviluppo può riguardare lo studio di un particolare problema, ad esempio un'applicazione SLAM simile a quella precedentemente affrontata; in queste situazioni esistono infatti già in letteratura delle soluzioni che sfruttano la struttura di base del filtro a particelle, ma prevedono delle modifiche in modo da poter ottenere delle prestazioni migliori. Nel caso delle applicazioni SLAM esistono ad esempio degli algoritmi di soluzione più veloci come FastSLAM 2.0, suggerito in [28], appartenente alla classe degli algoritmi di Rao-Blackwelliz; in questa categoria di problemi l'indipendenza di alcune variabili della distribuzione a posteriori viene sfruttata in modo da implementare un algoritmo fondato sia su basi di campionamento sia su base analitica, in modo da combinare i vantaggi di entrambi [9], in modo analogo alla soluzione già presentata in questo lavoro in cui il filtro di Kalman esteso è usato congiuntamente al filtro a particelle.

In [18] è mostrato un possibile miglioramento di mapping di Rao-Blackwelliz, basato su campionamento adattivo e ciclo chiuso attivo. Il ciclo chiuso attivo consiste nel notare come l'algoritmo SLAM potrebbe in genere funzionare in modo migliore controllando opportunamente il veicolo mobile. Nell'esempio proposto nel paragrafo 3.3.4, il robot osservava ad ogni passo un landmark scelto in modo casuale, ma le stime di localizzazione e di mappaggio sarebbero migliori se il veicolo potesse scegliere ad ogni passo quale feature osservare. Ad esempio, se ad ogni iterazione venisse guardato un landmark diverso, dopo i primi 10 passi il robot non avrebbe nessuna informazione a riguardo della propria posizione dato che ad ogni

passo effettuerebbe una nuova misurazione. Quindi le conoscenze sulla posizione del veicolo, all'inizio della simulazione, derivano esclusivamente dal controllo odometrico e quindi il robot si troverebbe a muoversi in ciclo aperto. In [18] è quindi consigliato che il controllo del movimento del veicolo dovrebbe tener conto che spesso è più utile concentrarsi sugli stessi landmark in modo da ottenere una migliore stima della propria posizione e dell'ambiente circostante, piuttosto che individuare subito il maggior numero di features possibili. Oltre a tale miglioramento si può considerare anche il problema del ricampionamento adattivo, ovvero nella possibilità di effettuare il resampling non ad ogni iterazione del filtro, ma solo quando la varianza delle particelle supera una certa soglia prefissata che può essere ricavata anche in modo sperimentale. Questa soluzione rappresenta quindi un compromesso tra la necessità di effettuare il ricampionamento e il problema dell'impoverimento della varietà delle particelle. Ovviamente l'idea di tenere in considerazione anche l'entropia dei campioni è ancora valida e può essere opportunamente inserita nel nuovo contesto.

E' quindi evidente come un filtro a particelle standard possa essere variamente modificato in modo da adattarsi alla particolare applicazione di interesse. Questa è probabilmente la direzione di maggiore sviluppo di questo lavoro di tesi, maggiormente incentrato sullo studio dell'evoluzione dello stato di un sistema dinamico nella sua generalità.

BIBLIOGRAFIA

1. Challa S., Bar-Shalom Y., "Nonlinear filter design using Fokker-Planck-Kolmogorov probability density evolutions", IEEE – *Transactions on Aerospace and Electronic Systems*, Vol .36, no, 1, pp 309-315, 2000
2. Arumpalam M.S., Maskell S., Gordon N., Clapp T., "A tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking", IEEE – *Transactions on Signal Processing*, Vol. 50, no. 2, February 2002
3. Ioannis Rekleitis M., "A particle filter tutorial for mobile robot localization", technical report (TR-CIM-04-03), Centre for Intelligent Machines, McGill University,2002
4. van der Merwe R., Doucet A., de Freitas J.F.G., Wan E., "The unscented particle filter", *Adv. Neural Inform. Process. Syst.*, Dec. 2000
5. Ho Y.C., Lee R.C.K., "A Bayesian approach to problems stochastic estimation and control", IEEE – *Transactions on Automatic Control*, Vol. 9, Issue 4, 1964
6. Caiti A., dispense del corso "Identificazione e controllo di sistemi incerti", *Corso di Laurea Specialistica in Ingegneria dell'Automazione*, Università di Pisa, A.A. 2003/2004
7. Ungarala S., Chen Z., "Bayesian data rectification of nonlinear systems with Markov chains in cell space", *IEEE – American Control Conference 2003*, Vol. 6, 4-6 June 2003
8. McCallum A., Pereira F., Freitag D., "Maximum entropy Markov models for information extraction and segmentation", Proceedings of the 17th International Conference on Machine Learning, pag.591-598, June 29-July 02, 2000
9. Gadeyne K., Lefebvre T., "The application of probabilistic techniques for the state/parameter estimation of (dynamical) systems and pattern recognition problems", Internal Report 2001R031, K.U.Leuven, Celestijnenlaan 300B, B-3001 Heverlee, Belgium, July 2004
10. Julier S.J., Uhlmann J.K., "A general method for approximating nonlinear transformations of probability distributions", *Technical report*, RRG, Dept. of Engineering Science, University of Oxford, 1996
11. Julier S.J., Uhlmann J.K., "A new extension of the Kalman filter to nonlinear systems", *Proc. of AeroSense: The 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls, Orlando, Florida.*, Vol. Multi Sensor Fusion, Tracking and Resource Management II
12. Julier S.J., "The scaled unscented transformation", *To appear in Automatica* (preprint), 2000

13. Casarin R., “Bayesian Monte Carlo filtering for stochastic volatility models”, CEREMADE *(CEntre de REcherche en MAthematiques de la DEcision*, report 0415, preprint 2004
14. Brockwell A.E., Rojas A.L., Kass R.E., “Recursive Bayesian decoding of motor cortical signals by particle filtering”, Neurophysiology No.91, pag. 1899-1907, 2004-11-25
15. Boerse Y., Driessen J.N., “Particle filter based detection for tracking”, *Proceedings of the American Control Conference*, Arlington, VA, June 25-27, 2001
16. Vermaak J., Gaguet M., Blake A., Perez P., “Sequential Monte Carlo fusion of sound and vision for speaker tracking”, *Proc. IEEE ICCV*, Vancouver, July 2001
17. Karlsson R., Gustafsson F., “Particle filter for underwater terrain navigation”, *Technical Reports from the Control & Communication group in Linköping*, no. LTH-ISY-R-2530, Submitted to Statistical Signal Processing Workshop, 2003
18. Stachniss C., Grisetti G., Hahnel D., Burgard W., “Improved Rao-Blackwellized mapping by adaptive sampling and active loop-closure”, work supported by German Science Foundation SFB/TR-8(A3) and by the Marie Curie program HPMT-CT-2001-00251, 2004
19. Lu L., Dai X.-T., Hager G., “A particle filter without dynamics for robust 3D face tracking”, *to appear in IEEE Workshop of Face Processing in Video jointed with CVPR’2004 (Oral presentation)*
20. Doucet A., Gordon N.J., Krishnamurthy V., “Particle filters for state estimation of jump Markov linear systems”, *IEEE Transactions on signal processing,* Vol.49, No.3, March 2001
21. Oudjane N., Musso C., “Progressive correction for regularized particle filters”, *Proc. 3rd International Conference Inform. Fusion*, 2000
22. Doucet A., “On sequential Monte Carlo methods for Bayesian filtering”, *Technical report*, Dept. Eng., Univ. Cambridge, UK, 1998
23. Noè A., Parenti F., “Identificazione di densità di probabilità con il principio della massima entropia”, *Lavoro di Tesi, Software Matlab*, Università di Pisa, Facoltà di Ingegneria, Corso di Laurea in Ingegneria Informatica, a.a.2001-2002
24. Sobczyk K., “Information dynamics: premises, challenges and results”, Mechanical Systems and Signal Processing, vol.15, no.3, 2001
25. Balestrino A., Caiti A., Noè A., Parenti F., “Maximum entropy based numerical algorithms for approximation of probability density functions”, *Submitted as Regular Paper*, 2002

26. Nguyen T.M., “Comparison of sampling-based algorithms for multisensor distributed target tracking”, Thesis submitted to the graduate faculty of the University of New Orleans in partial fulfilment of the requirement for the degree of Master of Science in The Department of Electrical Engineering, 2003
27. Nieto J., Guivant J., Nebot E., Thrun S., “Real time data association for FastSLAM”, *Proc. 2003 IEEE Int. Conf. Robotics and Automation*, pp412-418, Taipei, Taiwan, 14-19 September 2003
28. Montemerlo M., Thrun S:, Koller D., Wegbreit B., “FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges”, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)* 2003.