UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria
Corso di Laurea Specialistica in Ingegneria dell'Automazione

Corso di Progettazione di Sistemi di Controllo
a.a. 2005/06

Metodi per il Visual Tracking.

Alberto Soliman: 547129/IAM - solimana@dei.unipd.it

Fabio Viola: 531641/IAM - fabviola@dei.unipd.it

Davide Zilio: 530390/IAM - davidezilio@libero.it

Padova, 7 Luglio 2006

I. INTRODUZIONE

L'obiettivo di quest'elaborato è creare le basi teoriche e tecniche per l'implementazione di un sistema di misura di posizione basato su telecamere per il NAVLAB. Tale sistema viene primariamente concepito come strumento atto a fornire *in tempo reale* le posizioni di un insieme di veicoli autonomi liberi di muoversi all'interno del campo visivo di una rete di telecamere installata in laboratorio. Inoltre, è previsto che tale strumento possa flessibilmente interfacciarsi ad una rete di sensori dislocata nello spazio operativo e fondere virtuosamente tutta l'informazione proveniente dal campo per meglio risolvere il problema di misura. Le posizioni rilevate dovranno essere processate e fornite all'utente come misure dotate di *label*, ovvero associate ai veicoli che le hanno generate.

Il dispositivo che realizza tali compiti prende il nome di *tracker*, e la sua implementazione è molto complessa. Le difficoltà che si incontrano comprendono infatti in contemporanea problemi di modellistica, di teoria della stima, di visione computazionale e di implementazione *real time*.

Nel testo che seguirà, il design di un *multiple target tracker* viene studiato a tutti i livelli, mettendo a fuoco alcune delle possibili alternative presenti in letteratura per ognuno degli step necessari al suo sviluppo.

II. COMPUTER VISION

Dalla necessità iniziale di un sistema di misura in tempo reale sorge quindi un problema di *visual tracking*. Tradizionalmente nelle problematiche di tracking viene posta molta enfasi sugli aspetti tipicamenti connessi alla teoria della stima. Un tracker infatti può essere considerato a tutti gli effetti uno *stimatore*: data una successione di frame contenenti le proiezioni di un oggetto *target*, o un insieme di oggetti, esso permette di identificare un set di parametri che descrivono il comportamento degli stessi.

In un contesto applicativo la prima grande difficoltà che si incontra giace nell'accurata identificazione della proiezione dei target all'interno delle immagini.
In genere infatti, la maggioranza dell'informazione portata da un'immagine è irrilevante ai fini del tracking, e pertanto viene isolata mediante un algoritmo di *segmentazione* la sola quota del frame riguardante la proiezione dei target.

Le difficoltà che si incontrano si distribuiscono su almeno due livelli disitinti: un primo coivolge i problemi che insorgono nel processo di stima dovuti alle modalità di formazione dell'immagine, un secondo coinvolge il processing dell'immagine ed in particolare il significato dei dati che in questo modo si ottengono.

A. Agile Motion, Distractions e Occlusions

La *segmentazione* di un immagine è quel procedimento che permette, una volta stabiliti dei criteri imposti dal tipo di target, di scandire all'interno di una scena i suoi elementi costitutivi. Per esempio, mediante segmentazione è possibile distinguere in una foto lo sfondo dagli elementi in primo piano, oppure distinguere in una scena sottomarina il fondale di sabbia, le acque aperte e gli oggetti depositati sul fondale stesso.

É proprio una buona segmentazione che permette di ottenere la suddivisione dei dati portati dal frame cui si faceva riferimento in precedenza, ed una sua efficiente implementazione sta alla base del successo dell'azione del tracker. In condizioni reali di funzionamento, però, intervengono diversi fattori che rendono critica la ricerca e l'effettiva localizzazione delle proiezioni dei target sul piano immagine. Uno di questi si definisce '*agile motion*' e fa riferimento a quelle condizioni in cui la cinematica del target è caratterizzata da moti ad elevata velocità, tali da superare le capacità di predizione dinamica del tracker. Il suo effetto è un'elevata inaccuratezza nelle predizioni che comporta collateralmente un processo di segmentazione più lungo e complesso.
Solitamente, infatti, si tenta di contenere le dimensioni delle immagini da sottoporre a filtraggio, selezionandone le porzioni statisticamente più informative, onde contrarre i tempi di calcolo: non disporre di predizioni accurate non permette però di applicare un *gating* efficiente.

Un altro fenomeno che ostacola la segmentazione è denominato '*distraction*' e si riferisce alla presenza nella scena di altri oggetti la cui immagine appare molto simile a quella di uno o più target, dei quali si rende quindi più delicata l'identificazione.

Infine, si presenta un fenomeno di '*occlusion*' quando un oggetto si interpone tra target e telecamera impedendo la proiezione di una parte, o della totalità, della sua immagine nel frame. Ne deriva che quest'ultima circostanza comporta l'eventualità di non avere dati da fornire in ingresso allo stimatore.

La disciplina che sviluppa questo tipo di tematiche si chiama *data association*. Non è raro in effetti trovare accostati i due termini 'tracking' e 'data association', come avviene ad esempio nel titolo del celebre libro di Bar-Shalom e Fortman, *'Tracking and Data Association'*.

Nell'ambito del data association sono state sviluppate versioni adattate del filtro di Kalman, pensate per gestire in modo robusto tipologie diverse di fenomeni come occlusions e disturbances, versioni che in seguito, con lo svilupparsi di un hardware sufficientemente prestante, sono state adattate anche ad applicazioni di visione.

Si vedrà nel corso della lettura come, in base all'applicazione, tali algoritmi abbiano preso via via forme sempre diverse nella ricerca del trade-off migliore tra complessità computazionale, velocità di esecuzione e flessibilità nel saper gestire problematiche sempre più complesse.

Il primo passo che porta ad un'efficiente soluzione del problema del visual tracking è la riformulazione del concetto di *misura*, contestualizzato al mondo delle immagini.

B. Cos'è una misura?

Operando con algoritmi simili al filtro di Kalman si è abituati a pensare a misure dalla natura 'puntiforme'; lavorare su immagini comporta quindi lo sforzo intellettuale di elaboarare una strategia che permetta di estrarre da queste ultime una descrizione efficiente dei target.

Si rende necessario protocollare un metodo per codificare gli oggetti target in modo da poterne ricavare misure che possano essere fornite agli stimatori.
Inoltre, si rende necessario che tali misure risultino il meno sensibili possibile ai disturbi descritti in precedenza. Una buona scelta delle metriche e delle strategie di definizione degli oggetti contribisce, infatti, a far si che i disturbi incidano meno frequentemente e meno pesantemente nel processo di segmentazione dell'immagine. Per procedere all'analisi del processo di formazione dell'immagine è necessario introdurre brevemente il problema della stima dinamica da un punto di visto matematico.

Si può pensare che il problema base di tutta la visione computazionale sia quello di ricercare una stima *maximum a posteriori* (MAP) dello stato del mondo **M** data una sua immagine, ovvero un segnale bidimensionale **I** che ne riporta una versione trasformata.
Analiticamente una stima di tipo MAP si configura come la ricerca del massimo, non necessariamente unico, della funzione densità di probabilità condizionata $p(\mathbf{M}|\mathbf{I})$:

$$\hat{\mathbf{M}}_{MAP} = \mathrm{argmax}_M \, p(\mathbf{M}|\mathbf{I}).$$

A tal proposito il teorema di Bayes fornisce uno strumento potente per la stima di **M** data una sua immagine **I**, permettendo la riformulazione:

$$p(\mathbf{M}|\mathbf{I}) = \frac{p(\mathbf{I}|\mathbf{M})p(\mathbf{M})}{p(\mathbf{I})}.$$

Focalizzandosi sul tracking, la stima dinamica di **M** si riduce in realtà all'identificazione dello stato di una piccola parte della scena: quella riportante il target.
Inoltre, la stima viene basata non solo sull'ultima immagine, ma sull'intera storia delle immagini di cui ci dispone.

Quindi, indicando al tempo t lo stato del target con $\mathbf{x}_t \in \mathbb{X}$, e l'immagine con $\mathbf{I}_t$, il compito del tracker sarà stimare lo stato che massimizza $p(\mathbf{x}_t|\mathbf{I}_t, \mathbf{I}_{t-1}, \ldots)$, ovvero, applicando la regola di Bayes:

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{I}_t, \mathbf{I}_{t-1}, \ldots) &= \frac{p(\mathbf{I}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{I}_{t-1}, \mathbf{I}_{t-2}, \ldots)}{p(\mathbf{I}_t), \mathbf{I}_{t-1}, \ldots)} \\ &= k_t \cdot p(\mathbf{I}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{I}_{t-1}, \mathbf{I}_{t-2}, \ldots), \end{aligned}$$

dove la sostituzione del termine $p(\mathbf{I}_t), \mathbf{I}_{t-1}, \ldots)$ con la costante $\frac{1}{k_t}$ trova giustificazione nel fatto che esso agisce come fattore di normalizzazione e viene solitamente dedotto a partire dai restanti termini.

Si osservino gli elementi costitutivi di

$$p(\mathbf{I}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{I}_{t-1}, \mathbf{I}_{t-2}, \ldots).$$

Il secondo fattore riassume la conoscenza a priori riguardo lo stato del target al tempo t: è infatti la predizione sullo stato del sistema basato sulle sue stime passate e la conoscenza della sua dinamica. Adottando l'ipotesi che le transizioni dello stato da una configurazione alla successiva costituiscano una catena di Markov è anzi possibile evidenziare questa formulazione nella relazione

$$p(\mathbf{x}_t|\mathbf{I}_{t-1}, \mathbf{I}_{t-2}, \ldots) = \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{I}_{t-1}, \mathbf{I}_{t-2}, \ldots)$$

Il primo fattore, $p(\mathbf{I}_t|\mathbf{x}_t)$, prende il nome di *image likelihood* e descrive la probabilità al tempo t di osservare nell'immagine una particolare proiezione del target dato il suo stato. Nel presente paragrafo si vuole mettere a fuoco in che modo la scelta delle modalità di misura incidano su tale funzione.

Col termine *modalità* si definiscono gli attributi visivi che costituiscono i criteri descrittivi del target nell'implementazione del tracker; alcuni esempi intuitivi postrebbero essere forma, colore o direzioni del moto. La codifica di tali parametri permette di definire uno spazio delle misure $\mathbb{Z}$, messo in relazione mediante una funzione continua $H(\mathbb{X}) = \mathbb{Z}$, detta *measurement function*, allo spazio di stato. Tale funzione di misura può descrivere relazioni più o meno complesse tra ciò che viene misurato e ciò che viene stimato, in base alle modalità adottate e alla tipologia del target.

In modo analogo, ottenuta una predizione dello stato è possibile ricostruire mediante una *image prediction function* un'immagine predetta da confrontare, secondo la modalità di misura adottata, con quella rilevata in ingresso allo stimatore, onde ottenere, per esempio,

un'aggiornamento della stima.
Tale funzione $\pi : \mathbb{X} \rightarrow \mathbb{I}$, che mappa lo spazio di stato $\mathbb{X}$ nello spazio delle immagini $\mathbb{I}$, dovrà contemplare e modellizare fenomeni quali l'illuminamento, le occlusioni, lo sfondo, i parametri intrinseci della telecamera e, in caso di ipotesi di non perfetta lambertianità, le proprietà superficiali dell'oggetto.

Sono questi passaggi intermedi di 'generazione della misura' che rendono l'image likelihood strettamente correlata al livello al viene sfruttata la conoscenza della fisica della formazione delle immagini del target e all'azione del rumore che, sempre presente in situazioni reali, porta alla loro corruzione. Ne consegue che l'obietttvo primario in questa fase di progettazione del tracker è trovare il giusto equilibrio tra complessità delle modalità di misura, che comportano diversi tempi di calcolo spesi nel preprocessing delle immagini, e qualità nella verosimiglianza delle stime prodotte dal tracker.

Per esempio, si supponga di voler seguire gli spostamenti di una palla azzurra. Si potrebbe decidere di adottare una modalità di colore per predire la tonalità della proiezione del pallone e definire una metrica definita su aree circolari per le tonalità, per quantificare il grado di similarità tra l'immagine predetta e quella effettivamente rilevata. Questo metodo non sfrutta tutta l'informazione disponibile sulla palla, per esempio possibili textures sulla superficie del pallone o il suo moto, ma attua una precisa scelta riguardo quali informazioni siano rilevanti e quali no.

Si analizzano in seguito tre diverse modalità di misura adottate comunemente nel visual tracking e se ricava la corrispettiva forma della likelihood function.

1) Homogeneous Regions: In computer vision si definisce '*region*' la proiezione in un'immagine di un patch connesso di una superficie liscia. Assegnata una regione, o patch, P è sempre possibile definire una funzione c_P che ne descriva nello spazio RGB il pattern intrenseco di colori.

Se tale funzione rimane pressochè costante all'interno di una regione quest'ultima si definisce *omogenea*.
Una regione omogenea R può essere quindi 'campionata' e la distribuzione di colori nello spazio RGB di tali campioni può essere approssimata, via decomposizione ai valori singolari (SVD), da un ellissoide parametrizzato con una matrice di rotazione **R**, una di traslazione **T** ed una di scaling **S**. Quest'approssimazione permette di sfruttare efficaciemente una metrica che induce nello spazio RGB una distanza, detta di Mahalanobis, definita come

$$\gamma(\mathbf{I}(x,y), T) = \left|\mathbf{S}^{-1}\mathbf{R}^T\left(\mathbf{I}(x,y) - \mathbf{T}\right)\right|$$

che misura il grado di similarità γ tra il colore **T** predetto ed il colore effettivamente presente nell'immagine al pixel (x,y), $\mathbf{I}(x,y)$.

Inoltre, l'informazione del colore viene combinata con una rappresentazione geometrica della regione omegenea.
R viene individuata nell'immagine mediante un rettangolo C parametrizzato con la posizione (x,y) del suo centro, le sue dimensioni dimensioni w e h e il suo orientamento ϕ.
C, che prende il nome di *positive center*, viene identificato come il *best-fitting rectangle* ottenuto minimizzando una funzione costo f il cui minimo si ricava minimizzando la somma delle γ relative ai pixel interni a C ed al contempo massimizzando quella dei pixel esterni ad esso. Un secondo rettanfolo F, denominato *inhibitory frame*, delimita localmente la regione di immagine prossima al target, circoscrivendo C in modo che per le norme di Mahalanobis valga la relazione $|F| = |C|$[1] e per le sue dimensioni venga mantenuto un rapporto w/h analogo a quello di C.

Assegnate tutte le condizioni su C e F si procede esprimendo analiticamente la funzione di verosimiglianza

$$p_{hr}(\mathbf{I}|\mathbf{R}) = \mathrm{sig}\left(\frac{1}{\sigma_{hr}^2}\sum_{(x,y)\in F\cup C} a(x,y)\cdot\psi_{hr}(x,y)\right), \tag{1}$$

dove $\mathrm{sig}(x) = \frac{1}{1+e^{-x}}$ è la funzione sigmoide, $a(x,y)$ è la frazione della superficie totale $|R|$ di R rappresentata dal pixel (x,y) e ψ_{hr} rappresenta il grado in cui ogni pixel della regione soddisfa la relazione di similitudine:

$$\psi_{hr}(x,y) = \begin{cases} -\gamma\left(\mathbf{I}(x,y), \mathbf{T}\right) & \text{se } (x,y)\in C; \\ \gamma\left(\mathbf{I}(x,y), \mathbf{T}\right) & \text{se } (x,y)\in F\setminus C. \end{cases} \tag{2}$$

Tale approccio, compensando alcune delle non idealità presenti nella formazione delle immagini, permette di individuare e scegliere i pixel i cui colori non siano in saturazione e in cui l'immagine della superficie non risulti sovraesposta, e di utilizzare solo questi nel confronto, necessario per la stima della verosimiglianza, con l'immagine predetta.

2) Textured Regions: con il termine '*textured region*' si definisce una regione il cui patch R è caratterizzato da un pattern di colori $c_P(R)$ contraddistinto da gradienti veriticali ed orizzontali significativi.

Tale proprietà permette di utilizzare metodi del tipo SSD[2] per valutare e/o stimare con successo trasformazioni geometriche e fotomotriche delle regioni analizzate. Nel contesto del tracking si è soliti limitarsi alla

[1] Si noti che F non soddisfa più il vincolo di massimizzare la somma delle γ relative ai pixel esterni.

[2] Sum-of-Squared-Differeces.

stima di trasformazioni affini, applicate a regioni le cui proiezioni possano essere approssimate con rettangoli.

Al tracker viene innanzitutto fornita un'immagine di riferimento del target, essa viene ottenuta, per esempio, selezionando una finestra rettangolare all'interno di un frame. Nel tracking la predizione dello stato $\mathbf{x}_t$ del target fornirà indirettamente un pattern predetto I_R, descrivendo una matrice di *warping*, ovvero una trasformazione, affine **A** da applicare all'immagine di riferimento.
Nella pratica l'immagine rettangolare predetta mediante $\mathbf{x}_t$ viene trasformata all'inverso utilizzando la trasformazione indotta da A^{-1}, e mediante interpolazione bilineare viene ottenuta un'immagine $\mathbf{I}_C$ delle stesse dimensioni dell'immagine di riferimento $\mathbf{I}_R$. Quest'ultimo passaggio è possibile grazie alla presenza di gradienti significativiall'interno dei pattern delle immagini, che rendono praticabile una stima dei coefficienti di scaling.

Si può quindi calcolare l'image likelihood come inversamente proporzionale alla SSD tra $\mathbf{I}_C$ e $\mathbf{I}_R$, definendo

$$p_{tr}(\mathbf{I}|\mathbf{R}) = \exp\left(-\frac{1}{\sigma_{tr}^2}\sum_{(x,y)\in\mathbf{I}_R} a(x,y)\cdot\psi_{tr}(x,y)\right), \tag{3}$$

dove $a(x,y)$ è la frazione della superficie totale $|R|$ di R rappresentata dal pixel (x,y) e

$$\psi_{tr}(x,y) = \left(\mathbf{I}_R(x,y) - \mathbf{I}_C(x,y)\right)^2. \tag{4}$$

3) Snakes: si definisce '*snake*' la proiezione sull'immagine di un contorno continuo giacente su una superficie piana. I contorni, in tutta generalità, possono essere definiti in modi molto diversi, per esempio come il luogo dei pixel in cui si riscontrano picchi di contrasto, oppure essere linee che delimitano determinate superfici.

In questo contesto si preferisce focalizzarsi sui contorni ottenibli mediante *edge detection*, ovvero mediante la convoluzione delle immagini con nuclei di filtri come quello di Canny o di Sobel. Tali algoritmi di *feature extraction* individuano i pixel dell'immagine in cui si riscontrano repentini cambi di luminosità: uno snake viene ottenuto come una B-spline cubica, periodica o aperiodica, congiungente un sottoinsieme di tali pixel, vincolata a deformazioni affini nel passaggio da un frame al successivo.
I vincoli introdotti fanno si che da una parte gli snakes possando descrivere accuratamente contorni di target dalla forma arbitrariamente complessa, ed al contempo possano gestire i gradi di libertà che la descrivono mediante sole trasformazioni di traslazione, rotazione e scaling.

La funzione π di image prediction, adattata agli snakes, ipotizza una curva derivata da quella presente nell'immagine, che si suppone essere descritta da un set Q di parametri affini. La curva presente deve copiare nel piano immagine la linea ideale composta dai pixel in cui si misurano variazioni marcate di intensità; inoltre, essendo una B-spline, essa dovrà essere composta da un numero n finito di tratti cubici. Per ognuno di questi viene eseguita una edge detection lungo un segmento di lunghezza L (tipicamente 10 o 20 pixel), normale al tratto di cubica e bisecato da questa.
Si indichi con $\Lambda(i)$ la posizione nell'immagine dell'i-esimo tratto di curva e con $z(i)$ la posizione dell'edge lungo l'i-esima normale a distanza minima da $\Lambda(i)$. Assumendo che lo stato **x** incorpori anche i parametri affini Q, la funzione di verosimiglianza adotta la forma:

$$p_{snake}(\mathbf{I}|\mathbf{x}) = \exp\left(-\frac{1}{\sigma_{snake}^2}\sum_{i=0}^{n-1} l(i)\cdot\psi_{snake}(i)\right), \tag{5}$$

dove $l(i)$ è la frazione della lunghezza totale $|\Lambda|$ dello snake associata alla normale i-esima e

$$\psi_{snake}(i) = \begin{cases} |\Lambda(i) - z(i)| & \text{se viene trovato un edge;} \\ \xi & \text{altrimenti,} \end{cases} \tag{6}$$

descrive in che modo gli edge trovati soddisfino il modello di forma; il parametro ξ funge da penalizzazione qualora nessun edge venga rilevato lungo le i-esime normali.

C. Il processo di misura

Si chiarisce ora come le misure vengono ottenute a partire dalle immagini in ingresso al tracker. Il processo di *estrazione della misura* è in sostanza una ricerca del massimo della funzione di verosimiglianza $p(\mathbf{I}|\mathbf{x})$ nell'intorno della predizione $\hat{\mathbf{x}}$ del filtro.
I parametri geometrici, come per esempio $x \in \mathbb{X}, y \in \mathbb{Y}$, l'orientamento ϕ o lo scaling s delle regioni di immagine corrispondenti agli stati a massima verosimiglianza così individuati vengono derivati[3] come misure $\mathbf{Z} \in \mathbb{Z}$.

I metodi di massimizzazione di $p(\mathbf{I}|\mathbf{x})$ sono quindi il cuore dell'estrazione della misura e variano in base alla dinamica del sistema del quale si fa tracking ed alla forma della densità.

La classe più semplice di tecniche adottabili prevedono l'ulitizzo di metodi di ascesa del gradiente, come quelli di Newton o di Powell, dei quali esistono anche versioni ottimizzate per la computer vision che permettono di ottenere una singola misura ottima con la quale

[3]Per capire i passaggi è molto importante riportare alla mente le relazioni prima descritte tra spazio di immagini, spazi di stato e spazio di misura.

eseguire l'aggiornamento del filro di tracking.
I metodi di ascesa del gradiente producono però risultati soddisfacenti quando la densità $p(\mathbf{I}|\mathbf{x})$ è unimodale e la dinamica del sistema è abbastanza lenta. Ciò generalmente assicura che il filtro sia sufficientemente robusto da funzionare correttamente anche con i risultati subottimi ottenuti limitando il numero massimo di passi dell'algoritmo di ottimizzazione. Una dinamica lenta permette pure di utilizzare questi metodi in presenza di alcune categorie di densità multimodali, limitando le possibilità che lo stato cada nel bacino di attrazione scorretto. Perchè ciò non accada è necessario che tra i diversi modi non vi sia però eccessiva inteferenza, come nel caso in cui in una densità bimodale si verifichino sovrapposizioni e split dei modi, accentuando la probabilità che dopo la divisione venga presa la decisione sbagliata.

Queste complicazioni, che nascono dal ricercare massimi locali delle funzioni di probabilità, portano solitamente ad adottare algoritmi in grado di gestire in modo efficace sia densità multimodali, sia transizioni di stato più rapide, sfruttando i paradigmi dei cosiddetti *randomized methods*. In seguito tali metodologie verranno spiegate in dettaglio, ora ci si accosta intuitivamente alla descrizione di questa soluzione, che prevede un *measurement sampling process*. L'idea è quella di 'campionare' un numero prestabilito N di misure dallo spazio delle misure[4], secondo una data distribuzione, sfruttando il *factored sampling* ed in seguito valutare le verosimiglianze dei campioni così ottenuti. Si procede quindi selezionando un numero limitato $n < N$ di campioni a massima verosimiglianza che andranno a costituire essi stessi la misura in ingresso al tracker, o che potranno alternativamente essere passati ad un algoritmo di ascesa del gradiente per ottenere un *best measurement*.

D. La ricerca delle invarianti: il concetto di feature e la feature extraction

Si prendano in esame due immagini della stessa scena, ma prese da due punti di vista diversi; andando a considerare uno specifico punto della scena si cerca di stabilire la corrispondenza tra lo la sua proiezione nella prima e nella seconda immagine. Si definiscano quindi I_1 e I_2 le due immagini della stessa scena e con $W(\mathbf{x})$ si indichi una finestra attorno ad $\mathbf{x}$, dove $\mathbf{x}$ è la posizione di un pixel nell'immagine. Perciò se $\mathbf{x}_1$ e $\mathbf{x}_2$ sono le proiezioni dello stesso punto p seconodo due diversi punti di vista, ripettivamente, si avrà

$$I_2(\mathbf{x}_2) = I_1(\mathbf{x}_1).$$

[4]Si vedrà che ciò è equivalente ad estrarre campioni dall'immagine in ingresso.

Solitamente, per fare matching tra due immagini, invece di considerare la precedente equazione in termini di singoli pixel di un immagine, si considerano corrispondenze in termini di regioni. Si sceglie pertanto una classe particolare di trasformazioni $\hat{h}$ che minimizzino l'effetto del rumore, sempre presente nei contesti applicativi:

$$\hat{h} = \arg\min_{h} \sum_{\mathbf{x}\in\tilde{W}(x)} \|I_1(\tilde{\mathbf{x}}) - I_2(h(\tilde{\mathbf{x}}))\|^2$$

e che dipendano da un insieme di parametri α. Con un abuso di notazione si indicherà la dipendenza di h dai parametri con $h(\alpha)$. Un pixel $\mathbf{x}$ viene quindi definito come un *point feature* se esiste un suo intorno $W(\mathbf{x})$ che soddisfa l'equazione

$$I_1(\tilde{\mathbf{x}}) = I_2(h(\tilde{\mathbf{x}}, \alpha)), \quad \forall \mathbf{x} \in W(x),$$

e determina *univocamente* i parametri α di h.

III. SIFT

Il matching tra immagini è un fondamentale aspetto di molti problemi nella computer vision, come il *riconoscimento* di scene o di oggetti.

Le features di un'immagine hanno molte proprietà che possono essere adatte per il matching di differenti immagini, di un oggetto o di una scena. Le features, in base alla funzione h che le individua, possono essere invarianti rispetto alla rotazione, alla riduzione in scala dell'immagine, ma in genere solo parzialmente invarianti rispetto al cambiamento di illuminazione della scena. Le features sono ben localizzate sia nel dominio della frequenza sia nello spazio, riducendo l'effetto di disturbi quali clutter o rumore nella loro individuazione. Grazie ad efficienti algoritmi si possono estrarre da una pressochè qualsiasi immagine un alto numero di features; con il sostanziale vantaggio, inoltre, che le features sono altamente distinguibili tra loro.

Il costo computazionale di tali algoritmi può essere minimizzato dalla presenza di filtri in cascata; infatti le operazioni che richiedono un costo maggiore sono applicate solamente per la localizzazione iniziale delle features.
I passi principali usati per calcolare l'insieme delle features dell'immagine potrebbero essere i seguenti:

- **individuazione dell'estremo nel scale-space:** viene calcolato efficaciemente usando una funzione di differenze di Gaussiane che identifica i punti di potenziale interesse che sono invarianti rispetto la riduzione in scala e l'orientamento.
- **localizzazione dei keypoint:** per ciascun pixel candidato localizzato, un modello dettagliato è adatto per determinare la localizzazione e la riduzione in scala appropriata.

- **compito dell'orientazione:** una o piu' orientazioni sono assegnate a ciascun keypoint, esse sono basate sulle direzioni del gradiente dell'immagine locale.
- **descriptor dei keypoint:** i gradienti dell'immagine locale sono misurati rispetto alla riduzione in scala scelta, nella regione attorno a ciascun keypoint.

Questo approccio prende il nome di Scale Invariant Feature Transform (SIFT); esso trasforma i dati dell'immagine in coordinate, invariante in scala, relative alle features locali. Il SIFT genera un ampio numero di features, che coprono densamente l'immagine, nell'intero range di scale e locazioni.
Ad esempio, una classica immagine di 500×500 pixels, alla quale si applica il SIFT, riesce a produrre circa 2000 features. La quantità di features è particolarmente importante per la successiva ricostruzione dell'oggetto. Per esempio, per individuare piccoli oggetti, in immagini con distractions, si richiede che ci siano almeno tre features per fare correttamente matching a partire da ciascun oggetto e per averne un'affidabile identificazione.

Le features estratte dal SIFT, per il matching e il per il riconoscimento dell'immagine, sono per prima cosa estratte da un insieme di immagini di riferimento, e successivamente memorizzate in un database.

I *keypoint descriptors* sono features altamente distinguibili, perciò una singola feature, che sia keypoint descriptor, generalmente trova il suo corretto match con buona probabilità anche in un ampio database di features.

In immagini con clutter molte features estratte dallo sfondo non avranno un corretto match nel databse, creando così molti matches 'falsi' in aggiunta a quelli 'veri'. I matches corretti possono però essere filtrati a partire dall'intero insieme dei matches, creando un sottoinsieme di keypoints concordi con l'oggetto, la sua locazione, scala e orientazione nella nuova immagine.

A. Individuazione dell'estremo nel scale-space

Il primo passo per l'individuazione del keypoint richiede di identificare la locazione e la scala, che può essere ripetutamente assegnata anche avendo differenti punti di vista dello stesso oggetto. L'individuazione della locazione, che rimane comunque invariante ai cambiamenti di scala dell'immagine, può essere realizzata utilizzando una funzione continua chiamata scale-space.

Lo space-scale di un'immagine è quindi una funzione, $L(x, y, \sigma)$, definita come il prodotto di convoluzione di una Gaussiana *variable-scale*, $G(x, y, \sigma)$, con l'immagine in ingresso, $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y),$$

dove $*$ è l'operatore di convoluzione in x e y, e

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}.$$

Per individuare in modo efficace i keypoint stabili nello space-scale si considera l'estremo nello scale-space di $D(x, y, \sigma)$, ques'ultima rappresenta la convoluzione di una di differenze di Gaussiane con l'immagine, in dettaglio[5]:

$$\begin{aligned} D(x, y, \sigma) =& (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ =& L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned} \tag{7}$$

In piu', la funzione di differenze di Gaussiana fornisce un'approssimazione del Laplaciano, normalizzato in scala, della Gaussiana, $\sigma^2\nabla^2 G$. La normalizzazione del Laplaciano con il fattore σ^2 è necessaria per l'invarianza.

La relazione che intercorre tra D e $\sigma^2\nabla^2 G$ si può comprendere meglio con l'equazione (parametrizzata con il termine σ):

$$\frac{\partial G}{\partial \sigma} = \sigma\nabla^2 G.$$

Da questa relazione risulta:

$$\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

e quindi,

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2\nabla^2 G.$$

Il fattore $(k-1)$ non influenza la locazione del massimo nell'equazione. L'errore di approssimazione va a zero quando k tende a 1. Si deve considerare inoltre il fatto che l'approssimazione non ha un forte impatto sulla stabilità della ricerca del massimo, così come nella sua localizzazione, per alcune significanti riduzioni in scala.

Come si vede in VII-A, l'immagine iniziale è ripetutamente convoluta con le Gaussiane, il risultato è un insieme di immagini nello space-scale, mostrato a sinistra. Vengono poi sottratte le Gaussiane adiacenti, immagine a destra. Il processo si ripete scalato di volta in volta di un fattore due.

L'individuazione del massimo e del minimo locale di $D(x, y, \sigma)$ avviene con la comparazione del punto campione con i suoi otto punti diacenti nell'immagine corrente, e con i nove punti dell'immagine scalata rispettivamente sopra e sotto (vedi III-A). Il costo di questo tipo di controllo è relativamente basso, dovuto al fatto che molti dei punti campione saranno scartati durante le

[5]Il calcolo della funzione è particolarmente efficiente.

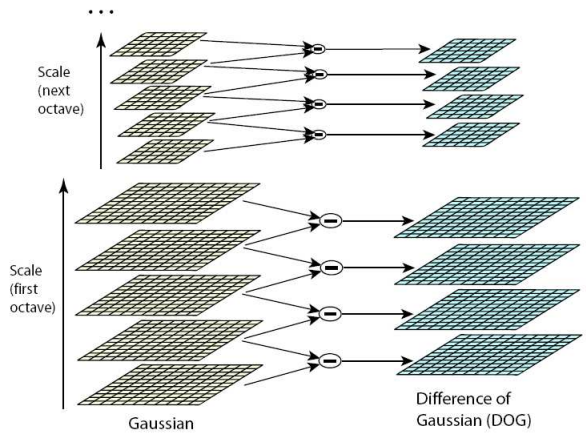

Fig. 1. Approccio nella costruzione di $D(x, y, \sigma)$

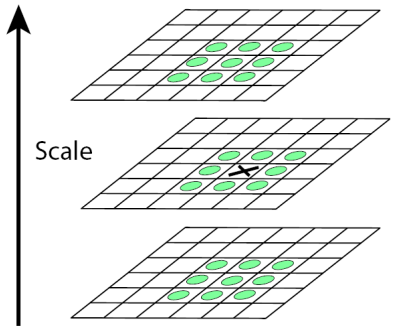

Fig. 2. Individuazione degli estremi locali

prime verifiche. Un'importante questione, nell'individuazione del massimo locale, consiste nel determinare la frequenza di campionamento nell'immagine e il dominio della scala. Sfortunatamente, ne deriva che non c'è uno spazio minimo di campionamento per individuare tutti gli estremi, dato che gli estremi possono essere arbitrariamente vicini. Tuttavia, gli estremi che si trovano troppo vicini tra loro si dimostrano instabili alle piccole perturbazioni dell'immagine.

Il numero dei keypoints incrementa con l'aumento del campionamento della scala, e di conseguenza cresce anche il numero totale dei maches corretti.
Ciò è di importanza fondamentale, in quanto il successo del riconoscimento dell'oggetto spesso dipende maggiormente dalla *quantità* assoluta dei keypoints correttamente mappati, pittosto che dalla percentuale di keypoint per i quali si verifica matching.

B. Localizzazione dei keypoint

Alcuni algoritmi sviluppano un metodo appropriato per una funzione quadratica in 3D, una volta applicata a dei campioni locali essa determina tramite interpolazione la locazione del massimo; con questo si ha un sostanziale miglioramento per quanto riguarda il matching e la stabilità. Questo approccio usa lo sviluppo di Taylor (fino ai termini quadratici) della funzione space-scale, $D(x, y, \sigma)$, shiftata affinchè l'origine non diventi il punto campione:

$$D(x) = D + \frac{\partial D^T}{\partial x}x + \frac{1}{2}x^T\frac{\partial^2 D}{\partial x^2}x \tag{8}$$

dove D e le sue derivate sono valutate rispetto al punto campione e $x = (x, y, \sigma)^T$ è l'offset dello stesso punto. La locazione del punto estremo, $\hat{x}$, è determinata prendendo la derivata della funzione rispetto ad x e ponendola uguale a zero, ottenendo così

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x}. \tag{9}$$

Ne risulta un sistema lineare $3x3$, che può essere risolto con un costo minimo. Se l'offset $\hat{x}$ è maggiore di 0.5 in qualsiasi dimensione, implica che l'estremo si trovi vicino ad un diverso punto campione. Quindi se ciò accade, il punto campione deve essere cambiato e l'interpolazione si esegue in riferimento a quest'ultimo punto. L'offset finale $\hat{x}$ viene aggiunto alla locazione del suo punto campione e si ha la stima interpolata per la locazione dell'estremo. La funzione calcolata nel punto massimo, $D(\hat{x})$ viene usata per scartare gli estremi instabili con un contrasto basso. Questo si può ottenere sostituendo l'equazione (9) nella (8), risulta

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x}.$$

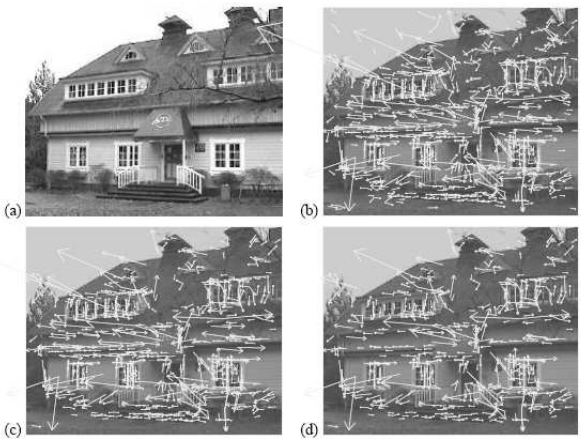

Fig. 3. Effetti della scelta dei punti chiave

La III-B mostra gli effetti della selezione dei punti chiavi in un immagine naturale. L'immagine è scelta con una bassa risoluzione (233×189 pixel), con lo scopo di evitare molti clutter e i keypoints sono rappresentati

come vettori; vettori servono per indicare la locazione, la scala e l'orientazione di ogni keypoint. La III-B (a) mostra l'immagine originale, la III-B (b) mostra gli 832 keypoints individuati, ad eccezione del massimo e del minimo della funzione di differenze di Gaussiane, mentre (c) mostra 729 i keypoints restanti dopo aver scartato quelli con un valore di $|D(\hat{x})|$ minore di 0.03. Per avere stabilità non è sufficiente eliminare i keypoints con un basso contrasto. La funzione di differenze di Gaussiane darà una risposta robusta lungo gli edges, anche se la locazione non è ben determinata lungo gli stessi e perciò ci sarà la presenza di una piccola quantità di rumore. Per quanto riguarda le principali curvature, si ottengono dalla matrice Hessiana 2×2, H, calcolate nella locazione dei keypoints e nella rispettiva scala:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{10}$$

Le derivate sono stimate considerando le differenze tra i punti campione vicini. Gli autovalori di H sono proporzionali alle principali curvature di D. Si prende α come l'autovalore con la maggiore ampiezza e β il piu' piccolo. La somma degli autovalori si ricava dalla traccia di H e il loro prodotto dal determinante della stessa:

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

Definiamo r come il rapporto tra α e β così da avere $\alpha = r\beta$. Allora,

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

che dipende solamente dal rapporto degli autovalori e non più dal loro singolo valore.
La quantità $(r+1)^2/r$ è minima solo quando i due autovalori risultano uguali e aumentano con r.

Quindi, per avere la certezza che il rapporto delle principali curvature sia al di sotto di certe soglie, si deve avere

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$$

L'ultima disuguaglianza risulta essere efficiente perchè con meno di 20 operazioni in floating point, ogni keypoint può essere controllato. L'effetto di questa procedimento si vede in III-B tra (c) e (d).

C. Il compito dell'orientazione

Nell'assegnazione di un'orientazione coerente a ciascun keypoint in base alle proprietà dell'immagine locale,si può ottenere un'invarianza rispetto alla rotazione dell'immagine. Bisogna tuttavia tener presenti che i descriptors hanno dei limiti e a priori vengono scartate alcune informazioni dell'immagine.

La scala del keypoint è usata selezionando la Gaussiana omogenea all'immagine, L, in modo tale che i calcoli successivi si effettuano rispetto ad una scala invariante. In ogni immagine campione, $L(x,y)$, il gradiente d'ampiezza, $m(x,y)$, e l'orientazione, $\theta(x,y)$, sono stati precedentemente calcolati tramite la differenza dei pixel:

$$\begin{aligned} m(x,y) = &\big[(L(x+1,y) - L(x-1,y))^2 + \\ &+ (L(x,y+1) - L(x,y-1))^2\big]^{\frac{1}{2}} \\ \theta = = &\, tan^{-1}((L(x,y+1) - L(x,y-1))/ \\ &/(L(x+1,y) - L(x-1,y)))). \end{aligned}$$

D. Il descriptor di un immagine

Il passo successivo consiste nel determinare un descriptor per una regione dell'immagine. Quest'ultima dev'essere abbastanza distinguibile e deve rimanere invariante alle variazioni, quali il cambiamento di illuminazione e il punto di vista. Un ovvio approccio potrebbe essere quello di campionare l'intensità dell'immagine attorno ad un keypoint rispetto ad un'opportuna scala, e nel fare matching si adotta una correlazione normalizzata.

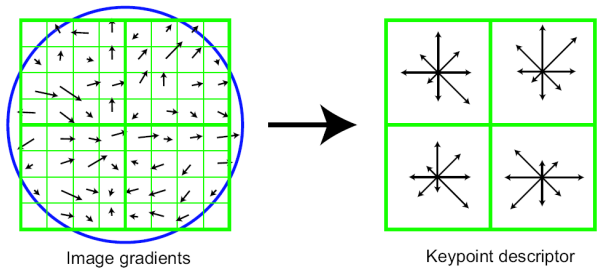

Fig. 4. La computazione del punto chiave descrittore

Nella III-D parte a sinistra: le ampiezze del gradiente e le orientazioni sono campionate attorno alla locazione del keypoint. Per ottenere l'invarianza nella rotazione, le coordinate dei descriptor e le orientazioni dei gradienti sono state ruotate in accordo con l'orientazione del keypoint. I gradienti sono rappresentati con piccole frecce in corrispondenza della locazione di ogni campione. Il motivo per cui si usa la Gaussiana è quello di evitare improvvisi cambiamenti nel descriptor nel caso ci siano

spostamenti anche modesti nella posizione della finestra e di cercare di non considerare i gradienti lontani dal centro del descriptor. Il descriptor è formato da un vettore contente il valore di tutti gli istogrammi orientati in ingresso. Per ridurre gli effetti che sono provocati dai cambiamenti di illuminazione si modifica il vettore delle feature. Il vettore è stato normalizzato all'unità di lunghezza. Il vettore normalizzato serve a cancellare un cambiamento nel contrasto dell'immagine, dove i pixel e i gradienti sono moltiplicati per la stessa costante. Invece un cambiamento nella luminosità, dove una costante viene sommata a ciascun pixel dell'immagine, non cambierà i valori dei gradienti. Si può quindi affermare che il descriptort rimane invariante per piccoli cambiamenti dell'illuminazione.

E. Applicazioni per il riconoscimento di oggetti

Un importante aspetto non ancora affrontato è il riconoscimento di oggetti. Il riconoscimento degli oggetti viene eseguito facendo inizialmente un matching di ciascun keypoint, in modo autonomo rispetto al database ottenuto con il training delle immagini. Molti matches in prima analisi potranno risultere incorretti a causa delle ambiguità tra features o perchè alcune features presentano clutter. Successivamente per ciascun cluster viene controllato e si genera un dettaglio geometrico adatto al modello. Il risultato è usato per accettare o eventualmente rifiutare l'interpretazione.
Il miglior candidato per il match rispetto ad ogni keypoints si trova identificando i *nearest neighbor* presi dal database. Il *nearest neighbor* è definito come il keypoint con la minima distanza Euclidea. Tuttavia, molte features avranno o meno un corretto match durante il training dipendendo dalla presenza di clutter o il ,ancato riconoscimento di features. Una misura piu' accurata si ottiene confrontando la rispettiva distanza che intercorre tra un punto e altri due punti a lui prossimi.
Non si conoscono algoritmi che possano identificare l'esatta vicinanza dei punti nello spazio ad alta dimensione. Di solito si usa un algoritmo approssimato, chiamato Best-Bin-First (BBF),che ritorna i punti più vicini con la più alta probabilità. Il motivo per cui l'algoritmo BBF lavora particolarmente bene, per questo tipo di problema, sta nel fatto che si considerano solamente matches nei quali i *nearest neighbor* sono distante almeno 0.8 volte la distanza dei secondi *nearest neighbor* e inoltre non necessita di risolvere esattamente i casi piu' difficili, in cui molti *nearest neighbor* assumono distanze simili.

La III-E mostra un esempio di riconoscimento di oggetti per un immagine con clutter e occlusioni contenente oggetti 3D. Il training delle immagini di un treno

Fig. 5. Riconoscimento di oggetti in immagini 3D

giocattolo e di una rana sono rappresentate a sinistra. Questi si possono riconoscere nell'immagine con clutter con un occlusione estesa, rappresentata in mezzo. Un parallelogramma è disegnato attorno a ciascun oggetto riconosciuto mostrando i bordi del training dell'immagine originale nella trasformazione affine risolta durante il riconoscimento. I quadrati piu' piccoli indicano i punti chiave che sono stati usati per il riconoscimento.

In generale, le superfici piane possono essere identificate, con sicurezza, per una rotazione che non superi i 50° e in ogni condizione di illuminazione. Per quanto riguarda gli oggetti 3D il range di rotazione in profondità per un sicuro riconoscimento è solamente di 30°; in questi casi il cambiamento di illuminazione crea molti disturbi. Per queste ragioni, nel riconoscimento di oggetti 3D le migliori prestazioni si hanno integrando features da diversi punti di vista. I keypoints si possono applicare al problema di localizzazione di un robot o di mapping. I keypoints del SIFT descritti fino ad ora sono particolarmente utili per un corretto match. I keypoint sono inoltre invarianti alla rotazione dell'immagine, alla riduzione in scala e robusti per un ben sortito range di distorsioni affini, quali aggiunta di rumore e cambiamenti di illuminazione.

IV. IL PROBLEMA DI STIMA DINAMICA

Un sistema di target tracking raccoglie una sequenza di misure rumorose provenienti dal campo, generate da uno o più potenziali target e le partiziona in un numero finito di insiemi disgiunti di osservazioni, chiamati *tracks*, prodotte dal medesimo target.

La proprietà fondamentale di un dispositivo di target tracking è pertanto la capacità di stimare in modo affidabile il passato e il presente del sistema dinamico costituito dall'insieme dei target, nonchè di predirne gli stati futuri.

La differenza fondamentale tra un problema tradizionale di stima e un problema di tracking è che si deve fronteggiare sia l'inaccuratezza delle misure, sia l'incertezza associata alla non conoscenza delle origini delle misure e delle corrette associazioni osservazioni coi target. Come introdotto in precedenza questo problema addizionale è generato da:

- *clutter* dovuto a misure spurie, come le già citate distractions;
- interferenza tra target che, per esempio, può comportare occlusioni;
- misure che contraddicono le predizioni;
- falsi allarmi nella feature detection.

Dal punto di vista matematico, l'approccio al problema viene tradizionalmente[6] impostato come segue.

A. L'approccio Bayesiano nel Tracking

Un modello di transizione di stato descrive la distribuzione *a priori* $\{\mathbf{x}_k; k \in \mathbb{N}\}$ di un *hidden Markov process*, solitamente chiamato hidden state process, invece un modello d'osservazione descrive la verosimiglianza delle osservazioni $\{\mathbf{z}_k; k \in \mathbb{N}\}$, dove k è un indice discreto di tempo.

Adottando un approccio di stima Bayesiana, tutta l'informazione riguardo $\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k\}$ ottenuta in base alle osservazioni ottenute fino all'istante k compreso può essere ricavata dalla funzione di distribuzione di probabilità congiunta *a posteriori* $p(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k | \mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k)$. Nella pratica si è pertanto interessati a sviluppare un algoritmo per generare una stima ricorsiva della suddetta distribuzione ed in particolare delle sue *marginali* $p(\mathbf{x}_k | \mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k)$ cui in letteratura vengono attribuiti nomi diversi, come *marginal a posteriori distribution, filtering distribution* o, data la loro importanza centrale nel problema di stima, *a posteriori distribution*.
D'ora in poi si adotterà la seguente notazione: con $\mathbf{x}_{0:k} := \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k\}$ e $\mathbf{z}_{0:k} := \{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k\}$ si indicherà l'evoluzione dello stato e la sequenza delle osservazioni fino all'istane k. Pertanto, la distribuzione congiunta *a posteriori* verrà denotata con $p(\mathbf{x}_{0:k} | \mathbf{z}_{0:k})$.

Si approfondisce ora il problema di modellistica del sistema di target concentrandosi su modelli dinamici in spazio di stato a tempo discreto. In tracking richiede la definizione di due modelli: un modello di transizione di stato e uno per la generazioni delle osservazioni.

Risulta molto importante relazionare bene i due modelli, soprattutto in relazione al 'ruolo' che il rumore ricopre nei modelli. Esso difatti non solo rappresenta il rumore ed i disturbi effettivamente presenti nel sistema reale, ma è anche chiamato a compensare le imprecisioni nella modelizzazione dei target. Se, per esempio, le misure sono modellate con un'accuratezza pari a quella delle misure 'vere' allora diventa cruciale che anche il modello di transizione sia adeguatamente preciso.

1) La scelta del modello dinamico: In generale il modello di transizione viene descritto in spazio di stato mediante un'equazione alle differenze non-lineare del tipo

$$\mathbf{x}_{k+1} = \mathrm{f}(\mathbf{x}_k, \mathbf{u}_k, k) + \mathbf{v}_k \tag{11}$$

dove $\mathrm{f}(\cdot,\cdot,\cdot)$ è la mappa $(R)^n \times (R)^r \times (R) \to \mathbb{R}^n$ di transizione di stato, $\mathbf{x}_k \in \mathbb{R}^n$ è lo stato del sistema all'istante k, $\mathbf{u}_k \in \mathbb{R}^r$ l'ingresso di controllo noto, e $\mathbf{v}_k$ un vettore di rumore che descrive tanto la quota di rumore che 'pilota' il sistema sia l'incertezza legata alla modellizzazione.

Si portano tre esempi di modello di transizione di basilare importanza per il visual tracking.

- *Random walk*
  É il modello più semplice di dinamica di target utilizzato nel tracking. Lo stato è bidimensionale e si compone delle coordinate (x, y) nel piano immagine della posizione del target. Ad ogni step l'aggiornamento avviene sommando allo stato attuale del rumore Gaussiano a media nulla.
  $$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + w_{x,k} \\ y_k + w_{y,k} \end{bmatrix} \tag{12}$$
  Il vettore di rumore $\mathbf{w}_k$ può essere interpretato come l'integrazione nell'intervallo di campionamento ΔT di una velocità $\mathbf{v}_k$ estratta da una distribuzione Gaussiana a media nulla e covarianza $\frac{\sigma^2}{\Delta T}$.

- *Moto browniano*
  In un modello di moto Browniano il rumore $\mathbf{w}$ che pilota la dinamica del sistema secondo la relazione
  $$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \mathbf{w}_k \tag{13}$$

[6] Il gruppo ha poi seguito un'approccio alternativo a quello presentato, che verrà descritto in seguito.

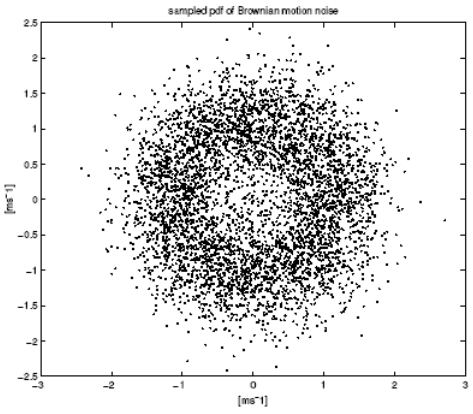

Fig. 6. Campionamento di pdf di rumore di moto Browniano.

può essere caratterizzato da una pdf più complessa di quella di una Gaussiana, come ad esempio quella di cui un campionamento è riportato in Fig. 6. Il principale vantaggio è quello di poter introdurre statisticamente un bias alle velocità dei target.

- *Modelli a velocità costante* Assumendo che i target si muovano con velocità costante a tratti è possibile estendere lo stato di ogni singolo target ad una quaterna in cui appaiono anche le velocità lungo gli assi coordinati ed ottenere il seguente modello.

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ v_{x,k+1} \\ v_{y,k+1} \end{bmatrix} = \begin{bmatrix} x_k + v_{x,k} \cdot \Delta T \\ y_k + v_{y,k} \cdot \Delta T \\ v_{x,k} \\ v_{y,k} \end{bmatrix} + \mathbf{w}_k \qquad (14)$$

2) La scelta del modello della misura: Definito anche *Sensor Model*, il modello di osservazione, a differenza di quello di transizione, è meno basato su supposizioni e più legato alla modalità di misura ed alla fisica del sensore di misura, come si è ampiamente visto nella sezione di computer vision. Svincolandosi dall'applicazione contingente, per completezza, la formulazione convenzionale del modello di misura è del tipo

$$\mathbf{x}_k = \mathrm{h}\left(\mathbf{x}_k, k\right) + \mathbf{w}_k \qquad (15)$$

dove $\mathrm{h}(\cdot,\cdot,\cdot)$ è la mappa $(R)^n \times (R) \to \mathbb{R}^m$ dallo spazio di stato allo spazio di misura, $\mathbf{x}_k \in \mathbb{R}^n$ è lo stato del sistema all'istante k, e $\mathbf{w}_k$ un vettore di rumore che tiene conto dell'incertezza legata alla modellizzazione del sensore e del rumore additivo presente in ogni processo di misura.

Si noti come convenzionalmente la dipendenza delle misure dagli ingressi di controllo al sistema non sia resa esplicita nel modello di misura e quindi come questi vengano omessi nelle equazioni.

B. L'approccio Bayesiano ricorsivo

Si accosta ora il problema del *calcolo* ricorsivo della stima di $\mathbf{x}_k$.

Congruentemente con quanto visto precedentemente, $\mathbf{x}_k$, fissato k, è una variabile aleatoria: pertanto il 'calcolarne' una stima significa determinarne la *pdf*[7] sfruttando tutta l'informazione disponibile, ovvero determinare la filtering distribution $p\left(\mathbf{x}_k|\mathbf{z}_{1:k}\right)$.

Si supponga di disporre della distribuzione a posteriori all'istante $k-1$, ovvero di $p\left(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}\right)$, dello stato di un sistema autonomo[8] e di conoscere il modello di transizione e di misura secondo le (11) e (15).

É possibile ottenere la pdf *a priori* dello stato all'istante k mediante la

$$p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right) = \int p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right) p\left(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}\right) d\mathbf{x}_{k-1} \qquad (16)$$

dove $p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right)$ è definita, noto il modello di transizione (11) e la statistica del rumore di modello $\{\mathbf{v}\}$ in $k-1$, mediante la relazione

$$p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right) = \int p\left(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{v}_{k-1}\right) p\left(\mathbf{v}_{k-1}|\mathbf{x}_{k-1}\right) d\mathbf{v}_{k-1} \qquad (17)$$

Assumendo il rumore di modello all'istante k scorrelato dallo stato del sistema nel medesimo istante, ovvero assumendo

$$p\left(\mathbf{v}_{k-1}|\mathbf{x}_{k-1}\right) = p\left(\mathbf{v}_{k-1}\right)$$

è possibile riscrivere la (17) come

$$p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right) = \int \delta\left[\mathbf{x}_k - \mathrm{f}(\mathbf{x}_{k-1}, \mathbf{0}, k-1)\right] \times p(\mathbf{v}_{k-1}) d\mathbf{v}_{k-1} \qquad (18)$$

dove δ denota la funzione delta di Dirac.

All'istante k, quando una nuova misura $\mathbf{z}_k$ si rende disponibile, si utilizza il teorema di Bayes per *aggiornare* la densità a priori $p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right)$ ottenendo la densità a posteriori $p\left(\mathbf{x}_k|\mathbf{z}_{1:k}\right)$ con la formula

$$p\left(\mathbf{x}_k|\mathbf{z}_{1:k}\right) = \frac{p\left(\mathbf{z}_k|\mathbf{x}_k\right) p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right)}{p\left(\mathbf{z}_k|\mathbf{z}_{1:k-1}\right)} \qquad (19)$$

Utilizzando quindi il teorema della proprietà totale si riformula la (19) nella relazione (20)

$$p\left(\mathbf{x}_k|\mathbf{z}_{1:k}\right) = \frac{p\left(\mathbf{z}_k|\mathbf{x}_k\right) p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right)}{\int p\left(\mathbf{z}_k|\mathbf{x}_k\right) p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right) d\mathbf{x}_k} \qquad (20)$$

[7] Proabability Density Function.

[8] Ovvero gli ingressi al sistema sono costantemente nulli.

in cui la densita di probabilità condizionata $p\left(\mathbf{z}_k|\mathbf{x}_k\right)$ è definita, secondo modello di misura (15) e le statistiche note di $\mathbf{w}_k$ mediante la

$$p\left(\mathbf{z}_k|\mathbf{x}_k\right) = \int \delta\left[\mathbf{z}_k - \mathrm{h}(\mathbf{x}_{k-1}, k)\right] \times p(\mathbf{w}_k) d\mathbf{w}_k \quad (21)$$

Le relazioni di ricorrenza (16) e (19) purtoppo sono solo il corpo di una soluzione *formale* dell'iniziale problema di calcolo. In effetti questo metodo concettuale di propagare nel tempo la filtering distribution non è in genere determinabile in modo *analitico*.

Vi sono casi particolari, e molto noti, in cui una scelta molto restrittiva di modelli di transizione, di misura e di statistiche per i processi di rumore permette di adottare con successo quest'approccio.
Esempio fondamentale è il *Filtro di Kalman*. Esso è infatti un sistema dinamico che propaga nel tempo una variabile normale, aggiornando nel tempo i parametri della sua descrizione del secondo ordine, che, per una gaussiana, è equivalente ad una descrizione in distribuzione.

Dando per noto il filtro di Kalman, si procede descrivendo altri algoritmi che, senza perdere l'approccio Bayesiano, sono progettati per stimare lo stato di un sistema a tempo discreto governato da *generiche* equazioni stocastiche alle differenze.

C. Extended Kalman Filter

Il filtro di Kalman esteso (EKF) è un'approssimazione del filtro ottimo, derivato per modelli non-lineari nel 1970 da Jazwinski. L'idea alla base dell'algoritmo è utilizzare il modello non lineare per generare le predizioni e, in fase di aggiornamento, calcolare i guadagni di correzione costruendo un modello variazionale linearizzando la dinamica del sistema attorno alla stima ottima della traiettoria.

Si assuma che il sistema, a tempo discreto, sia descritto da un'equazione di transizione di stato stocastica non lineare del tipo

$$\mathbf{x}_k = \mathrm{f}\left(\mathbf{x}_{k-1}\right) + \mathbf{v}_k \quad (22)$$

e che lo stesso valga per il modello delle osservazioni

$$\mathbf{z}_k = \mathrm{h}\left(\mathbf{x}_k\right) + \mathbf{w}_k \quad (23)$$

dove $\mathbf{x}_t$ è lo stato del sistema al tempo t, $\mathrm{f}_k(\cdot,\cdot,k)$ e $\mathrm{h}_k(\cdot,k)$ sono funzioni non-lineari che mappano rispettivamente lo stato passato e l'ingresso di controllo corrente nello stato presente e lo stato presente nelle osservazioni, ed infine $\mathbf{v}$ e $\mathbf{w}$ sono processi Gaussiani a variabili scorrelate, a media nulla,

$$E[\mathbf{v}_k] = E[\mathbf{w}_k] = 0 \quad \forall k,$$

e con covarianze rispettivamente

$$E[\mathbf{v}_i \mathbf{v}_j^T] = \delta_{ij}\mathbf{Q}_i \text{ e } E[\mathbf{w}_i \mathbf{w}_j^T] = \delta_{ij}\mathbf{R}_i$$

Si assuma inoltre che i processi di rumore di misura e di modello siano scorrelati:

$$E[\mathbf{v}_i \mathbf{w}_j^T] = 0 \quad \forall i, j$$

1) Costruzione del modello variazionale: ad ogni istante temporale k si consideri la *reference trajectory* $\bar{\mathbf{x}}$ soluzione dell'equazione alle differenze

$$\bar{\mathbf{x}}_{k+1} = \mathrm{f}(\bar{\mathbf{x}}_k),$$

la matrice Jacobiana

$$\mathbf{F}\left(\bar{\mathbf{x}}_k\right) = \mathbf{F}_k := J\left[\mathrm{f}\left(\mathbf{x}_k\right)\right]\Big|_{\bar{\mathbf{x}}_k}$$

e la linearizzazione dell'equazione delle osservazioni nell'intorno del punto $\bar{\mathbf{x}}_k$

$$\mathrm{h}\left(\mathbf{x}\right) = \mathrm{h}\left(\bar{\mathbf{x}}_k\right) + \mathbf{H}_{\bar{\mathbf{x}}}\left[\mathbf{x}_k - \bar{\mathbf{x}}_k\right] + O(\epsilon^2)$$

dove

$$\mathbf{H}_{\bar{\mathbf{x}}} := J\left[\mathrm{h}\left(\mathbf{x}\right)\right]\Big|_{\bar{\mathbf{x}}_k}$$

e

$$\epsilon^2 := \left\|\mathbf{x} - \bar{\mathbf{x}}\right\|^2$$

Definendo quindi $\delta\mathbf{x}_k := \mathbf{x}_k - \bar{\mathbf{x}}_k$ si ottiene, fino ai termini del secondo ordine:

$$\delta\mathbf{y}_k := \mathbf{y}_k - \mathrm{h}\left(\bar{\mathbf{x}}_k\right) = \mathbf{H}_{\bar{\mathbf{x}}}\,\delta\mathbf{x}_k + \mathbf{w}_k$$

2) Passo di predizione: si supponga di disporre, al tempo k, della stima ottima $\hat{\mathbf{x}}_{k|k}$ e di costruire quindi il modello variazionale attorno la traiettoria $\bar{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k}$:

$$\bar{\mathbf{x}}_{k+1} = \mathrm{f}(\bar{\mathbf{x}}_k); \quad \bar{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k}$$

Per piccoli spostamenti si può scrivere la relazione

$$\delta\mathbf{x}_{k+1} = \mathbf{F}_k\left(\delta\mathbf{x}_k\right) + \tilde{\mathbf{v}}_k \quad (24)$$

dove il termine di rumore $\tilde{\mathbf{v}}_k$ può includere anche i termini di errore legati alla linearizzazione.

Date queste premesse è facile verificare che

$$\delta\hat{\mathbf{x}}_{k|k} := \hat{\mathbf{x}}_{k|k} - \hat{\mathbf{x}}_{k|k} = 0 \quad (25)$$

e conseguentemente che

$$\delta\hat{\mathbf{x}}_{k+1|k} = \mathbf{F}_k\left(\delta\hat{\mathbf{x}}_{k|k}\right) = 0 \quad (26)$$

dalle quali si ricava la relazione di predizione:

$$\hat{\mathbf{x}}_{k+1|k} = \bar{\mathbf{x}}_{k+1} = \mathrm{f}\left(\bar{\mathbf{x}}_k\right) = \mathrm{f}\left(\hat{\mathbf{x}}_{k|k}\right) \quad (27)$$

Per la covarianza P dell'errore di predizione $\delta\hat{\mathbf{x}}_{k+1|k}$ si ha invece

$$P_{k+1|k} = \mathbf{F}_k \, P \, {\mathbf{F}_k}^T + \tilde{Q} \tag{28}$$

dove $\tilde{Q}$ è la covarianza di $\tilde{\mathbf{v}}_k$.
Le (27) e (28) costituiscono il passo di predizione dello stimatore e sono uguali a quelle che si ricaverebbero esplicitamente operando in modo analogo a quanto si farebbe per il classico KF.

3) Aggiornamento: al tempo $k+1$ si trova in ingresso al filtro una nuova misura, $\mathbf{y}_{k+1}$, che viene utilizzta per aggiornare la predizione $\hat{\mathbf{x}}_{k+1|k}$ e la covarianza del suo errore di stima $P_{k+1|k}$.

Linearizzando l'equazione (23) intorno al punto $\bar{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_{k+1|k}$ si ottiene

$$\begin{aligned} \delta\mathbf{y}_{k+1} &= \mathbf{y}_{k+1} - \mathrm{h}\left(\hat{\mathbf{x}}_{k+1|k}\right) && (29)\\ &= \mathbf{H}_{\bar{\mathbf{x}}_{k+1}} \, \delta\mathbf{x}_{k+1} + \tilde{\mathbf{w}}_{k+1} && (30) \end{aligned}$$

dove il termine di rumore $\tilde{\mathbf{w}}_{k+1}$, con covarianza $\tilde{\mathbf{R}}$, include sia il rumore di misura sia l'errore di linearizzazione.

Le due equazioni (24) e (30) definiscono un modello lineare per il quale si possono scrivere le equazioni di aggiornamento della stima dello stato basate sul paradigma del classico filtro di Kalman.

$$\begin{aligned} &\delta\hat{\mathbf{x}}_{k+1|k+1} = \\ &\quad \delta\hat{\mathbf{x}}_{k+1|k} + L_{k+1}\left(\mathbf{y}_{k+1} - \mathbf{H}_{\hat{\mathbf{x}}_{k+1|k}} \, \delta\hat{\mathbf{x}}_{k+1|k+1}\right) \end{aligned} \tag{31}$$

dove

$$L_{k+1} = P_{k+1|k} \, \mathbf{H}^T_{\hat{\mathbf{x}}_{k+1|k}} \, \Lambda^{-1}_{k+1}; \tag{32}$$

$$\Lambda^{-1}_{k+1} = \mathbf{H}_{\hat{\mathbf{x}}_{k+1|k}} \, P_{k|k} \, \mathbf{H}^T_{\hat{\mathbf{x}}_{k+1|k}} + \tilde{\mathbf{w}}_{k+1}; \tag{33}$$

$$\begin{aligned} P_{k+1|k+1} = {} & \Gamma_{k+1} \, P_{k+1|k} \, \Gamma^T_{k+1} + \\ & + L_{k+1} \, \tilde{\mathbf{w}}_{k+1} \, L^T_{k+1}; \end{aligned} \tag{34}$$

$$\Gamma_{k+1} = \left(I - L_{k+1} \, \mathbf{H}_{\hat{\mathbf{x}}_{k+1|k}}\right). \tag{35}$$

L'aggiornamento di P si effettua con la standard DRE utilizzata anche nel filtro di Kalman.
La covarianza $\tilde{\mathbf{R}}$ può essere calcolata empiricamente in batch con una procedura di tuning; è di uso comune, tuttavia, l'approssimarla con $\mathbf{R}$ alterando i termini diagonali per compensare l'effetto della linearizzazione.

D. Sequential Monte Carlo Methods

L'EKF appena introdotto fornisce una soluzione efficiente al problema della stima dinamica per il caso non lineare gaussiano.

Tuttavia, in molte applicazioni diventa di vitale importanza poter introdurre e gestire nel modello elementi di non linearità e non gaussianità per descrivere in modo sufficientemente accurato la dinamica del sistema.

Sequential Monte Carlo (SMC) è una classe di potenti metodi *simulation-based* che permettono un approccio alternativo ed efficace al calcolo, ed alla propogazione, di distribuzioni del tutto generali. In letteratura ci si riferisce a questa famiglia di algoritmi con nomi diversi, alcuni sono: *Sequencial Monte Carlo Method*, *Bootstrap Filtering*, *Condensation Algorithm*, *Particle Filtering*[9] e *Survival of the Fittest*.

A conferma della versatilità di tali metodologie, va sottolineato che recentemente i filtri partcellari sono stati applicati con successo in diversi problemi di stima, a partire da problemi analoghi a quelli che si stanno fronteggiando in NAVLAB, quindi visual tracking ed in genere localizzazione di robot mobili e target tracking, fino al *simultaneous localisation and map building* (SLAM) o al riconoscimento di segnali vocali.

Ci si riconduca al consueto problema di stima dinamica, si assuma ora che gli stati del sistema siano Marcoviani, non lineari e non Gaussiani. Con $\mathbf{x}_{0:k} := \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k\}$ e $\mathbf{z}_{1:k} := \{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$ si indicherà l'evoluzione dello stato e la sequenza delle osservazioni fino all'istane k. L'obiettivo, come in precedenza, sarà: stimare ricorsivamente le distribuzione congiunta a posteriori $p\left(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}\right)$, la distribuzione marginale a posteriori $p\left(\mathbf{x}_k|\mathbf{z}_{1:k}\right)$ e le aspettazioni

$$\begin{aligned} I\left(\mathrm{f}_k\right) = E_{\,p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})}\left[\,\mathrm{f}_k\left(\mathbf{x}_{0:k}\right)\right] := \\ := \int \mathrm{f}_k\left(\mathbf{x}_{0:k}\right) p\left(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}\right) d\mathbf{x}_{0:k} \end{aligned} \tag{36}$$

per ogni funzione di interesse[10] $\mathrm{f}_k : \mathbb{X}^{(k+1)} \rightarrow \mathbb{R}^{n_{\mathrm{f}_k}}$, integrabile w.r.t. $p\left(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}\right)$.

1) Perfect Monte Carlo sampling: l'idea che sta alla base del MCM è generare e propagare nel tempo una versione approssimata, ottenuta per campionamento, delle distribuzioni di interesse.

Si immagini infatti di saper *simulare*[11] N campioni, anche chiamati *particles* o *particelle*, i.i.d. $\left\{\mathbf{x}^{(i)}_{0:k}\right\}_{i=1}^{N}$ secondo la distribuzione $p\left(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}\right)$.
Da questi si può ottenere una stima della distribuzione

[9] In italiano si parla di *Filtri Particellari*

[10] Particolare interesse è rivolto al calcolo della media condizionata, nel qual caso vale $\mathrm{f}_k\left(\mathbf{x}_{0:k}\right) = \mathbf{x}_{0:k}$.

[11] Si utilizza la traduzione del termine 'simulate' trovato in letteratura, ma si potrebbe usare correttamente anche il verbo 'campionare'.

iniziale mediante la formula:

$$P_N(d\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) = \frac{1}{N}\sum_{i=1}^{N}\delta\left(\mathbf{x}_{0:k} - \mathbf{x}_{0:k}^{(i)}\right) \quad (37)$$

dove con δ si denota il delta di Dirac. Dalla proprietà di shifting della distribuzione δ consegue la possibilità di calcolare con facilità momenti e aspettazioni, utilizzando l'approssimazione $P_N(d\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$, in base all'uguaglianza:

$$I_N(\mathrm{f}_k) = \int \mathrm{f}_k(\mathbf{x}_{0:k})\, P_N(d\mathbf{x}_{0:k}|\mathbf{z}_{1:k})\, d\mathbf{x}_{0:k} = \\ = \frac{1}{N}\sum_{i=1}^{N}\mathrm{f}_k\left(\mathbf{x}_{0:k}^{(i)}\right) \quad (38)$$

É intuitivo capire quali siano i vantaggi del metodo Monte Carlo: da un set finito di particelle si può calocolare con estrema facilità una qualsiasi *stima* $I_N(\mathrm{f}_t)$ di $I(\mathrm{f}_t)$ con notevoli proprietà di robustezza e convergenza[12].

Purtroppo, è generalmente impraticabile e/o poco efficiente il campionamento *diretto* della distribuzione congiunta a posteriori $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$; ed è pertanto necessario sviluppare strategie alternetive che permettano di ottenere un suo campionamento indiretto, onde poter applicare con successo la (38).

Il metodo più comune, ed adottato in statistica, ha nome *Markov Chain Monte Carlo*. Purtroppo esso ha struttura *iterativa* e pertanto è intrensicamente poco adatto ai problemi *ricorsivi* di stima dinamica come quello del tracking. Altri metodi, che vengono in seguito presentati, sono l'*Importance Sampling* ed il *Sequential Importance Sampling*.

2) Importance Sampling: Il metodo dell'Importance Sampling implementa il paradigma del *Factored Sampling*; l'idea fondamentale è quella di campionare una distrubuzione 'facile' e di associare ai campioni così estratti degli opportuni 'pesi' da introdurre nella (38) per emulare la distribuzione di interesse. Il factored sampling viene introdotto dapprima a livello intuitivo e poi spiegato maggiormente in dettaglio.

Si ricordi il principio che permette di ricondurre la distribuzione di massa di variabili aleatorie discrete alla distribuzione di probabilità delle omologhe variabili aleatorie continue mediante la

$$F_{\mathbf{x}}(x) = \sum_{\mathbf{x}_i\in\mathbb{X}} w_i\delta^{-1}(\mathbf{x} - \mathbf{x}_i) \quad (39)$$

[12]Una trattazione completa delle proprietà di convergenza di tali stime si trova in A.Doucet, N.de Freitas, N. Gordon, and editors, *Sequential Monte Carlo Methods in Practice*, Springer-Verlag, New York, 2001.

dove $\mathbb{X}$ è l'alfabeto della variabile aleatoria discreta e w_i sono i pesi, e quindi di ottenere la fdp della variabile continua equivalente come

$$f(\mathbf{x})_{\mathbf{x}} = \sum_{i=1}^{N} w_i\delta(\mathbf{x} - \mathbf{x}_i). \quad (40)$$

Si può pensare di effettuare il medesimo procedimento a ritroso, per stimare dinamicamente la versione approssimata di una variabile aleatoria continua ignota, operando su un insieme di suoi campioni. Ciò si ottiene mediante un algoritmo capace di gestire dinamicamente la distribuzione di massa della variabile discreta equivalente propagando nel tempo il suo alfabeto ed i suoi pesi. Adottando tale impostazione, disponendo di N campioni $\mathbf{x}^i \sim q(\mathbf{x})$, dove $q(\mathbf{x})$ è una fdp facile da campionare, una qualsiasi altra fdp $p(\mathbf{x})$ potrà essere approssimata come

$$p(x) \approx \sum_{i=1}^{N} w_i\delta(\mathbf{x} - \mathbf{x}_i), \quad (41)$$

con

$$w_i \propto \frac{\mathbf{p}(x_i)}{\mathbf{q}(x_i)}. \quad (42)$$

Maggiore è il numero N di campioni, migliore risulterà essere la stima della fdp.

Viene pertanto introdotta una distribuzione $q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$ nota e facilmente campionabile; in letteratura tale funzione prende il nome di *importance sampling distribution* o, più concisamente, *importance density*.

Manipolando opportunamente la (36) si ottiene:

$$I(\mathrm{f}_k) = \int \mathrm{f}_k(\mathbf{x}_{0:k})\frac{p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})}{q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})} q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})\, d\mathbf{x}_{0:k} \\ = \int \mathrm{f}_k(\mathbf{x}_{0:k})\, w^*(\mathbf{x}_{0:k})\, q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})\, d\mathbf{x}_{0:k} \quad (43)$$

dove i parametri $w^*(\mathbf{x}_{0:k})$, definiti mediante

$$w^*(\mathbf{x}_{0:k}) = \frac{p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})}{q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})} = \frac{p(\mathbf{z}_{1:k}|\mathbf{x}_{0:k})\, p(\mathbf{x}_{0:k})}{p(\mathbf{z}_{1:k})\, q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})} \quad (44)$$

sono i *pesi di importanza 'veri'*.
Pertanto, sapendo generare N campioni i.i.d. da $q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$ una possibile stima di $I(\mathrm{f}_k)$ diventa:

$$\widehat{I_N^*}(\mathrm{f}_k) = \int \mathrm{f}_k(\mathbf{x}_{0:k})\, P_N(d\mathbf{x}_{0:k}|\mathbf{z}_{1:k})\, d\mathbf{x}_{0:k} \quad (45)$$

Purtroppo questa prima approssimzione non è utilizzabile in quanto richiede la conoscenza della costante di normalizzazione $p(\mathbf{z}_{1:k})$; è possibile però non utilizzare i pesi 'veri', bensì i *pesi di importanza non normalizzati*

$$w(\mathbf{x}_{0:k}) = \frac{p(\mathbf{z}_{1:k}|\mathbf{x}_{0:k})\, p(\mathbf{x}_{0:k})}{q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})} \propto w^*(\mathbf{x}_{0:k}) \quad (46)$$

riformulando le aspettazioni come

$$I(\mathrm{f}_k) = \int \mathrm{f}_k(\mathbf{x}_{0:k}) \frac{w(\mathbf{x}_{0:k})}{p(\mathbf{z}_{1:k})} q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})\, d\mathbf{x}_{0:k} \quad (47)$$

all'interno delle quali esprimere $p(\mathbf{z}_{1:k})$ con

$$\begin{aligned} p(\mathbf{z}_{1:k}) &= \int p(\mathbf{z}_{1:k}|\mathbf{x}_{0:k})\, p(\mathbf{x}_{0:k})\, d\mathbf{x}_{0:k} \\ &= \int \frac{p(\mathbf{z}_{1:k}|\mathbf{x}_{0:k})\, p(\mathbf{x}_{0:k})}{q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})} q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})\, d\mathbf{x}_{0:k} \\ &= \int w_k(\mathbf{x}_{0:k})\, q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) d\mathbf{x}_{0:k} \quad (48) \end{aligned}$$

ottenendo infine

$$I(\mathrm{f}_k) = \frac{\int \mathrm{f}_k(\mathbf{x}_{0:k})\, w_k(\mathbf{x}_{0:k})\, q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})\, d\mathbf{x}_{0:k}}{\int w_k(\mathbf{x}_{0:k})\, q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})\, d\mathbf{x}_{0:k}} \quad (49)$$

La funzione (49) può essere quindi approssimata con

$$\begin{aligned} \widehat{I_N}(\mathrm{f}_k) &= \frac{\frac{1}{N}\sum_{i=1}^{N} \mathrm{f}_k\left(\mathbf{x}_{0:k}^{(i)}\right) w\left(\mathbf{x}_{0:k}^{(i)}\right)}{\frac{1}{N}\sum_{j=1}^{N} w\left(\mathbf{x}_{0:k}^{(j)}\right)} \\ &= \sum_{i=1}^{N} \mathrm{f}_k\left(\mathbf{x}_{0:k}^{(i)}\right) \widetilde{w}_k^{(i)} \quad (50) \end{aligned}$$

dove i pesi *'veri'* vengono pertanto sostituiti da una loro *stima*.

$$\widehat{w}_k^{*(i)} = N\widetilde{w}_k^{(i)}. \quad (51)$$

in cui

$$\widetilde{w}_k^{(i)} = \frac{w\left(\mathbf{x}_{0:k}^{(i)}\right)}{\frac{1}{N}\sum_{j=i}^{N} w\left(\mathbf{x}_{0:k}^{(j)}\right)} \quad (52)$$

sono i *pesi d'importanza normalizzati*.

É interessante ricongiungersi ora con la spiegazione intuitiva data in precedenza: questo metodo di integrazione può essere visto come un campionamento in cui la distribuzione a posteriori $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$ viene approssimata con

$$\widehat{P_N}(d\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) = \sum_{i=1}^{N} w_k^{(i)} \delta\left(\mathbf{x}_{0:k} - \mathbf{x}_{0:k}^{(i)}\right).$$

e l'aspettazione $\widehat{I_N}(\mathrm{f}_k)$ è l'integrale della funzione $\mathrm{f}(\mathbf{x}_{0:k})$ moltiplicata per le *misure simulate* $\widehat{P_N}(d\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$:

$$\widehat{I_N}(\mathrm{f}_k) = \int \mathrm{f}(\mathbf{x}_{0:k})\, \widehat{P_N}(d\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) \quad (53)$$

3) Sequential Importance Sampling: Si analizza ora come modificare l'*Importance Sampling* per risolvere il seguente problema ricorsivo: si immagini di possedere il set di particelle e pesi costituenti un'approssimazione di $p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1})$, e di voler quindi calcolare un'approssimazione $\widehat{P_N}(d\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$ di $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$ con un nuovo pacchetto di samples senza però modificare il passato delle traiettorie simulate $\left\{\mathbf{x}_{0:k-1}^{(i)}\right\}_{i=1}^{N}$.

Si procede innanzitutto fattorizzando la densità d'importanza:

$$\begin{aligned} q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) &= q(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1})\, q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k}) \\ &= q(\mathbf{x}_0) \prod_{j=1}^{k} q(\mathbf{x}_j|\mathbf{x}_{0:j-1}, \mathbf{z}_{1:j}) \quad (54) \end{aligned}$$

e, ricordando che l'evoluzione di stato viene modellizzata come un processo Markoviano, le densità

$$p(\mathbf{x}_{0:k}) = p(\mathbf{x}_0) \prod_{j=1}^{k} p(\mathbf{x}_j|\mathbf{x}_{j-1}) \quad (55)$$

e

$$p(\mathbf{z}_{1:k}|\mathbf{x}_{0:k}) = \prod_{j=1}^{k} p(\mathbf{z}_j|\mathbf{x}_j) \quad (56)$$

grazie alle quali si può intervenire manipolando la (46) ottenendo la legge di aggiornamento ricorsivo per i pesi

$$\begin{aligned} w_k &= \frac{p(\mathbf{z}_{1:k}|\mathbf{x}_{0:k})\, p(\mathbf{x}_{0:k})}{q(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1})\, q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k})} \\ &= w_{k-1} \frac{1}{w_{k-1}} \frac{p(\mathbf{z}_{1:k}|\mathbf{x}_{0:k})\, p(\mathbf{x}_{0:k})}{q(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1})\, q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k})} \\ &= w_{k-1} \frac{p(\mathbf{z}_{1:k}|\mathbf{x}_{0:k})}{p(\mathbf{z}_{1:k-1}|\mathbf{x}_{0:k-1})} \times \\ &\qquad \times \frac{p(\mathbf{x}_{0:k})}{p(\mathbf{x}_{0:k-1})} \frac{1}{q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k})} \\ &= w_{k-1} \frac{p(\mathbf{z}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{x}_{k-1})}{q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k})} \quad (57) \end{aligned}$$

e quindi:

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p\left(\mathbf{z}_k|\mathbf{x}_k^{(i)}\right) p\left(\mathbf{x}_k^{(i)}\|\mathbf{x}^{(i)}\_k - 1\right)}{q\left(\mathbf{x}_k^{(i)}\|\mathbf{x}_{0:k-1}^{(i)}\, \mathbf{z}_{1:k}\right)}. \quad (58)$$

In questo modo è possibile definire l'alogoritmo ricorsivo di *Sequential Importance Sampling* in tre passi:

1) Si estraggono N particelle da $q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k})$: per $i = 1, \ldots, N$ si campiona la denistà d'importanza $\mathbf{x}_k^{(i)} \sim q(\mathbf{x}_k^{(i)}|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{z}_{1:k})$ e si aggiornano le traiettorie simulate $\mathbf{x}_{0:k}^{(i)} := \left\{\mathbf{x}_{0:k-1}^{(i)}, \mathbf{x}_k^{(i)}\right\}$;

2) Per ogni particella campionata viene quindi aggiornato il relativo peso:

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p\left(\mathbf{z}_k|\mathbf{x}_k^{(i)}\right) p\left(\mathbf{x}_k^{(i)}\|\mathbf{x}^{(i)}\_k{-}1\right)}{q\left(\mathbf{x}_k^{(i)}\|\mathbf{x}_{0:k-1}^{(i)}\,\mathbf{z}_{1:k}\right)};$$

3) Si normalizzano i pesi:

$$\widetilde{w}_k^{(i)} = \frac{w_k^{(i)}}{\sum_{i=1}^{N} w_k^{(i)}}.$$

E. Degeneracy

Il principale problema che sorge nell'utilizzo pratico dei filtri particellari SIS è la *degenerazione* dei pesi: dopo poche iterazioni, infatti, si verifica che tutti i campioni, fatta eccezione per uno, hanno pesi praticamente nulli. Tale fenomeno è intrinsecamente legato alla natura dei filtri particellari, tanto da essere studiatao a fondo e formalizzato in un teorema.

Kong-Liu-Wong dimostrano che la varinza incondizionata degli *importance weights*, nel momento in cui le osservazioni sono trattate come variabili aleatorie, cresce nel tempo[13]. Per il filtro ciò si traduce in una tendenza all'instabilità crescente nel tempo che, se purtroppo non può essere evitata, richiede di essere controllata.

Senza strategie di controllo, infatti, diventa enorme lo sforzo computazionale associato all'aggiornamento di particelle il cui contributo nella ricostruzione della *a posteriori* risulta pressochè nullo. Viene introdotta innanzitutto una *metrica* per il grado di degenerazione dei pesi; essa *misura* il numero effettivo N_{eff} di *particelle utili*:

$$N_{eff} = \frac{N}{1 + \mathrm{var}\left(w^*\left(\mathbf{x}_{0:k}\right)\right)} \tag{59}$$

Sfruttando la (51) si può ottenere *on-line* una stima $\widetilde{N}_{eff}$ di N_{eff} con

$$\widetilde{N}_{eff} = \frac{N}{\frac{1}{N}\sum_{i=1}^{N}\left(\widehat{w}_k^{*(i)}\right)^2} = \frac{N}{\sum_{i=1}^{N}\left(\widetilde{w}_k^{(i)}\right)^2}, \tag{60}$$

dove $\widetilde{w}_k^{(i)}$ sono i pesi normalizzati, che permette di operare in *feedback* per ridurre l'effetto della degenerazione. A tal proposito vengono ora presentate due strategie.

1) Scelta dell'Importance Density: è possibile dimostrare che la scelta della densità d'importanza ricopre un ruolo di primaria importanza nel comportamento dei filtri particellari riguardi il fenomeno della degenrazione.

[13]Si ricordi che la somma dei pesi è normalizzata a 1. Pertanto, al crescere della loro varianza si noterà una drastica contrazione dei loro valori.

Doucet, col criterio di scegliere la densità d'importanza $q\left(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}\right)$ massimizzando N_{eff}[14], dimostra che la scelta ottima è:

$$q\left(\mathbf{x}_k|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{z}_{1:k}\right)_{ott} = p\left(\mathbf{x}_k|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{z}_{1:k}\right) \tag{61}$$

Sostituendo quindi la (61) nella (58) si ottiene

$$\begin{aligned} w_k^{(i)} &= w_{k-1}^{(i)} \frac{p\left(\mathbf{z}_k|\mathbf{x}_k^{(i)}\right) p\left(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}\right)}{p\left(\mathbf{x}_k|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{z}_{1:k}\right)} \\ &= w_{k-1}^{(i)} p\left(\mathbf{z}_k|\mathbf{x}_{k-1}^{(i)}\right) \end{aligned} \tag{62}$$

in cui i pesi non normalizzati $w_k^{(i)}$ sono indipenenti dalle particelle $\mathbf{x}_k^{(i)}$: ciò permetterebbe in fase applicativa di parallelizzare la simulazione delle traiettorie $\left\{\mathbf{x}_{0:k}^{(i)}\right\}_{i=1}^{N}$ e l'elaborazione delle misure $\left\{\mathbf{z}_{0:k}^{(i)}\right\}_{i=1}^{N}$.

Tuttavia, per poter utilizzare la (61), sarebbe necessario anche riuscire a campionare da $p\left(\mathbf{x}_k|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{z}_{1:k}\right)$ e valutare, a meno di una costante moltiplicativa, l'integrale

$$p\left(\mathbf{z}_k|\mathbf{x}_{k-1}^{(i)}\right) = \int p\left(\mathbf{z}_k|\mathbf{x}_k^{'}\right) p\left(\mathbf{x}_k^{'}|\mathbf{x}_{k-1}\right) d\mathbf{x}_k^{'}$$

cose, in generale, inconciliabili nella pratica.

Per questa ragione risulta conveniente utilizzare densità d'importanza sub-ottime ragionevolmente simili alle *a posterirori*, ed adottare inoltre degli accorgimenti che rendano l'implementazione la più semplice possibile.
Per esempio, si trova spesso in letteratura la seguente soluzione, adottabile qualora le densità a priori e a posteriori si aspettino somiglianti e sia verosimile assumere che le predizioni siano sufficientemente informative ed affidabili: si pone la densità d'importanza pari alla a priori del modello[15], ovvero

$$q\left(\mathbf{x}_k|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{z}_{1:k}\right) = p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right), \tag{63}$$

e quindi si sostituisce la (63) nella (58) ottenendo infine la relazione ricorsiva:

$$w_k^{(i)} = w_{k-1}^{(i)} p\left(\mathbf{z}_k|\mathbf{x}_k^{(i)}\right). \tag{64}$$

2) Resampling: un secondo metodo per far fronte al fenomeno della degenarazione è quello del *ricampionamento*.
L'idea alla base del metodo è, quando la stima del livello di generazione $\widehat{N}_e f f$ supera una soglia prestabilita, eliminare le particelle con pesi d'importanza eccessivamente

[14]Ovvero minimizzando var $\left(w_k^{*(i)}\right)$.

[15]Si ricordi l'ipotesi di Markovianità.

bassi e sostituirli con con nuovi campioni e nuovi pesi mediante ricampionamento.

Il ricampionamento agisce basilarmente mappando le misure pesate $\left\{\mathbf{x}_{0:k}^{(i)}, \widetilde{w}_k^{(i)}\right\}_{i=1}^{N}$ nelle misure pesate, con peso *uniforme*, $\left\{\mathbf{x}_{0:k}^{(i)}, \frac{1}{N}\right\}$ estraendo con rimpiazzo uniformemente dall'insieme $\left\{\mathbf{x}_{0:k}^{(i)}\right\}$ con probabilità $\left\{\widetilde{w}_k^{(i)}\right\}_{i=1}^{N}$ [16].

Esistono diversissimi schemi di ricampionamento, molti dei quali implementabili in tempo $O(N)$. Il più popolare, prende il nome di *multinomial sampling* e viene utilizzato nel $SamplingImportanceResampling$, SIR. Non ci si dilungherà nei dettagli, tuttavia si desidera rendere almeno l'idea di come tale algoritmo opera.

La particolarità di tale algoritmo giace nel fatto che le nuove particelle vengono estratte campionando la distribuzione $\widehat{P}_N\left(d\mathbf{x}_{0:k}|\mathbf{z}_{1,k}\right)$, il che equivale a campionare i nuovi N campioni secondo una distribuzione multinomiale di parametri $\widetilde{w}_k^{(i)}$.

Altri metodi sono il *Residual Sampling*[17] ed il *systematic Resampling*. In fase di simulazione verrà adottato un approccio leggermente diverso da quelli fin qui descritti, che verrà spiegato separatamente.

[16]Per visualizzare intuitivamente il processo, si immagini di mettere in stack i campioni pesati, con i pesi normalizzati, e quindi di dividere il segmento unitario in sottosegemeti di lunghezza $\widetilde{w}_k^{(i)}$. Si immagini quindi di campionare uniformemente lungo il segmento unitario e di sostituire gli elementi dello stack con i campioni pesati ottenuti prendento la particella del campione estratto e peso $1/N$.

[17]Questo metodo agisce in modo molto simile a quanto delinato nella nota precedente.

V. DATA ASSOCIATION

Ci si accosta in questa sezione al *data association*, disciplina volta alla risoluzione del problema della corretta associazione dinamica delle molte misure generate, da uno o più sensori, con i soggiacenti stati o target osservati che le hanno prodotte. Ciò è di fondamentale importanza, per esempio, in computer vision, per ricostruire le traiettorie delle features nel piano immagine. In tale problematica sono coinvolte questioni riguardanti la validazione dei dati, la creazione e la gestione delle possibili associazioni tra misure e stati (o misure e target), e l'inizializzazione, il mantenimento o la cancellazione di track e stati.
Se nei problemi convenzionali di stima l'incertezza è legata alla localizzazione delle misure, nel data association l'incertezza si sposta su ciò che ha generato la misura.

Gli strumenti che verranno utilizzati nel data association sono versioni rielaborate delle stesse, efficaci tecniche introdotte nella sezione riguardante la stima dinamica, a dimostrazione di quanto potente e versatile sia la teoria della stima.

A. Gating

Nella applicazioni di tracking, in coda alla ricezione dei segnali di misura, si trova sempre una procedura di selezione delle misure che sarnno riportate in ingresso allo stimatore. Tale procedura si chiama *gating*, e grazie ad essa solo le misure giacenti in una regione precisa dello spazio di misura $\mathbb{Z}$ verranno considerate come associabili ad uno, o più target.

Ciò è fatto per evitare che il tracker debba scandire tutto lo spazio $\mathbb{Z}$ dovendo conseguentemente considerare un numero molto elevato di associazioni. Se, nonostante la presenza del gating, più di una misura risultasse associabile ad un singolo target si presenterà una condizione di *incertezza* nell'associazione, che dovrà essere debitamente gestita.

Riportando alla memoria la teoria del filtraggio statistico e il modello (15), al tempo k, definiamo *innovazione* la quantità ν_k così individuata:

$$\nu_k := \mathbf{z}_k - \widehat{\mathbf{z}}_{k|k-1} = \mathbf{z}_k - \mathrm{h}_k\left(\widehat{\mathbf{x}}_{k|k-1}\right).$$

La matrice covarianza dell'innovazione, S_k, sia quindi definita da

$$S_k := E\left[\left(\mathbf{z}_k - \widehat{\mathbf{z}}_{k|k-1}\right)\left(\mathbf{z}_k - \widehat{\mathbf{z}}_{k|k-1}\right)^T\right].$$

Si assuma infine che il vettore delle misure sia M-dimensionale e che lemisure siano Gaussiane e

distribuite con pdf

$$p\left(\mathbf{z}_k|\mathbf{z}_{1:k-1}\right) = N\left(\widehat{\mathbf{z}}_{k|k-1}, S_k\right) \tag{65}$$

É quindi possibile introdurre d, detta *innovazione normalizzata*, determinata da

$$d_k^2 := \nu_k^T S_k^{-1} \nu_K \tag{66}$$

mediante la quale definire la pdf Gaussiana M-dimensionale per l'innovazione

$$p\left(\nu_k\right) = \frac{1}{(2\pi)^{M/2}\left|S_k\right|^{1/2}} \exp\left(\frac{-d_k^2}{2}\right) \tag{67}$$

dove $|S_k| = det\left(S_k\right)$.
Risulta quindi possibile individuare una regione in cui la misura reale verrà a cadere con una data probabilità γ:

$$G_k\left(\gamma\right) = \left\{\mathbf{z} : d^2 \leq \gamma\right\} \in \mathbb{Z} \tag{68}$$

Tale regione viene definita *Gate* ed il parametro γ viene chiamato *Gate threshold*.

B. Approccio GNN

Il metodo di data association più semplice, e probabilmente quello di più diffuso utilizzo, è il *Global Nearest Neighbour*. L'idea che giace alla base di tale metodo è molto semplice: considerare solo l'ipotesi di associazione più verosimile, individuata nel processo di gating, ovvero quella che minimizza la distanza tra misura e predizione. Tale metodo, piuttosto rudimentale,

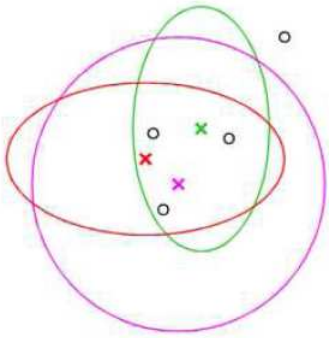

Fig. 7. Conflitto nel processo di association: con **x** sono inndicate le predizioni del tracker, con **o** le misure in ingresso e con le ellissi i gate. Il GNN non è in grado di gestire in modo efficace la scelta dell'associazione.

non riesce a gestire con successo siatuazioni di conflitto, come quella riportata in Fig. 7, dove tre target si sono accostati rendendo elevato il grado di incertezza delle associazioni.

Vengono ora introdotti metodi più complessi, la cui peculiarità è quella di saper gestire e valutare dinamicamente diverse ipotesi di associazione.

C. Probabilistic Data Association Filter

Da questa sezione in poi si abbandonano le soluzioni immediate al problema del data association, e si introducono strategie più complesse e generali, che perdono l'approccio prettamente locale, tipo GNN, e sfruttano tutta l'informazione disponibile sul sistema per valutare correttamente le associazioni papabili.

Si consideri per ora il tracking di un singolo target. Il filtro di Kalman, e la sue versione estesa alle dinamiche non-lineari, si sono dimostrati metodi molto efficienti nel tracking, nel caso in cui la distribuzione delle misure fosse Gaussiana. Altre 'varianti' del medesimo filtro si sono poi dimostrate capaci di gestire anche densità a posteriori differenti dalle Normali, necessarie per modellare in modo efficace fenomenologie di rumore che portano, per esempio, a misure multiple o a mancanza delle stesse per determinati lassi di tempo. Per questa ragione il filtro di Kalman, nelle sue varianti, viene nuovamente riconsiderato e adattato al data association.

Il Probabilistic Data Association Filter (PDAF) è infatti un'estensione del filtro di Kalman, che sposa l'approccio Bayesiano nel problema del data association e nella gestione dell'aggiornamento dello stato del tracker, quando, inseguendo un solo target, non viene rilevata presenza di misure oppure giungono misure multiple a causa del rumore.

Il PDAF, a differenza del GNN, che sceglie *esclusivamente* la misura più prossima a quella predetta, *genera* la misura in ingresso al filtro mediando su tutte le varie misure candidate, attribuendo loro un peso basandosi sulle seguenti assunzioni:

1) è assunta la presenza di un solo target di interesse, che genera un'unica 'vera' misura, soggetta però a sparire, solitamente a causa di un'occlusione temporanea del target, o di una *featrure detection* suboptima in un qualche punto della pipeline tra telecamera e tracker;
2) tutte le altre misure sono da considerarsi 'false' e prodotte da un processo di rumore a distribuzione uniforme;
3) ad ogni iterazione del filtro viene generata una regione di gate;
4) le detection del target avvengono indipendentemente nel tempo con probabilità P_D.

Il passo più rilevante nel filtro di Kalman è il calcolo dell'innovazione ν. Il PDAF introduce analogamente la nozione di *combined innovation*, calcolata, avendo a disposizione le n misure individuate ad un certo passo, come la somma pesata delle singole innovazioni

$$\nu = \sum_{i=1}^{n} \beta_i \nu_i$$

dove ogni β_i è la probabilità dell'*evento associazione* θ_i, ovvero che sia proprio l'i-esima misura ad essere stata originata dal target. Viene introdotta quindi β_0, la probabilità dell'evento in cui nessuna delle misure sia *target-originated*. In questo modo tutto viene considerato lo spettro di tutti i possibili eventi, e pertanto vale l'uguaglianza:

$$\sum_{i=1}^{n} \beta_i = 1$$

La descrizione dettagliata del funzionamento del filtro è rimandata in seno al JPDAF, che, come si vedrà, è un'estensione del PDAF per il caso del multiple target tracking.

Si vuol solo sottolineare come l'ipotesi di uniforme distribuzione delle misure generate da rumore in realtà non rispecchi in modo oggettivo la realtà; si nota nelle applicazioni, infatti, che gran parte delle *false measurements* si generano in prossimità del target, dovute per esempio a riflessi, o comunque si presentano raggruppate in regioni specifiche dello spazio dele misure.

D. Inseguire congiuntamente più target: Joint Probabilistic Data Association Filter

L'ipotesi fondamentale per l'implementazione del PDAF è che vi sia *uno ed un solo* target nella scena, e che solo questo generi, *secondo la modalità scelta* misure *persistenti* nel tempo.

Se tale ipotesi non viene soddisfatta, per esempio a causa di elementi simili al target nella scena, o di vere e proprie *distractions*, vengono a generarsi picchi nella funzione di verosimiglianza $p(\mathbf{I}|\mathbf{x})$ che inducono pericolosissimi *bias* nelle stime del filtro.

Il far operare in parallelo più filtri PDAF, ognuno su un elemento distinto di una scena, potrebbe sembrare una soluzione a questo problema. Tuttavia si nota che se i target vengono a trovarsi in regioni prossime della scena solitamente le misure prodotte da un target vengono utilizzate anche dai tracker relativi ad altri target portando infine ad una configurazione degenere in cui più tracker collassano ed inseguono un medesimo target. Non vi è infatti nessun criterio di separazione tra le misure basate su una statistica congiunta del moto dei target.

Si introduce così un nuovo metodo, che aggiunge al PDAF un ulteriore livello di logica, in modo da garantire che tutti i trackers siano correttamente distribuiti sulle misure: il Joint Probabilistic Data Association Filter[18]

Il JBDAF sfrutta ampiamente l'impianto Bayesiano della stima congiunta dello stato del sistema dei target introdotto nella sezione precedente integrando:

- un modello della dinamica dei target;
- un modello per le *false detections* ed il *clutter*;
- una statistica delle occlusioni;

con l'obbiettivo di legare una sorta di *probabilità a posteriori* alle possibili associazioni; ciò permette quindi di stimare la posizione dei target operando una media pesata a posteriori sulle associazioni, e così risolvere il problema del data association.

Si assuma quindi che all'istante $t-1$ la densità condizionata dello stato i-esimo $\mathbf{x}^{(i)}(t-1)$, date le misure fino a $t-1$, sia Gaussiana con media $\widehat{\mathbf{x}}^{(i)}(t-1)$ e matrice di covarianza $P^{(i)}(t-1)$, ovvero

$$p\left(\mathbf{x}^{(i)}(t-1)|\mathbf{z}^{(i)}_{1:t-1}\right) \sim N\left(\widehat{\mathbf{x}}^{(i)}(t-1), P^{(i)}(t-1)\right) \tag{69}$$

Tali stime, all'arrivo delle nuove misure, vengono aggiornate sfruttando le equazioni del filtro di Kalman in base alle associazioni θ e probabilità di queste.

Nel JPDAF le associazioni sono strutture complesse che relazionano le m misure con gli n stati[19] del tracker: ad ogni stato viene associata una ed una sola misura, oppure un'etichetta che indica una condizione di occlusione; tutte le misure che non vengono associate ad uno stato vengono invece etichettate come clutter. La misura associata allo stato i-esimo, secondo l'associazione k-esima θ_k, avrà indice $j(i,\theta_k)$ e sarà quindi denotata con $\mathbf{z}_{j(i,\theta_k)}$.
Una misura di clutter verrà indicata con indice $j(0,\theta_k)$, l'occlusione dello stato i-esimo con $j(i,\theta_k)=0$.

Si indicano con $\widehat{\mathbf{x}}^{(i)}_{\theta_k}(t)$ e $P^{(i)}_{\theta_k}(t)$ l'aggiornamento della media e della covarianza, a partire dalle condizioni iniziali $\mathbf{x}^{(i)}(t-1)$ e $P^{(i)}(t-1)$, utilizzando le misure $\mathbf{y}_i(t) = \mathbf{z}_{j(i,\theta_k)}(t)$, condizionate con l'associazione θ_k.

Data l'ipotesi di Marcovianità per i modelli (11) e (15), le densità condizionate $p\left(\mathbf{x}^{(i)}(t)|\theta_k, \mathbf{z}_{1:t}\right)$ saranno Gaussiane di media $\widehat{\mathbf{x}}^{(i)}_{\theta_k}(t)$ e covarianza $P^{(i)}_{\theta_k}(t)$:

$$p\left(\mathbf{x}^{(i)}(t)|\theta_k, \mathbf{z}_{1:t}\right) \sim N\left(\widehat{\mathbf{x}}^{(i)}_{\theta_k}(t), P^{(i)}_{\theta_k}(t)\right) \tag{70}$$

Applicando il *teorema della probabilità totale* è possibile esprimere la densità di probabilità condizionata come una somma di Gaussiane:

$$p\left(\mathbf{x}^{(i)}(t)|\mathbf{z}^{(i)}_{1:t}\right) = \sum_{\theta_k \in \Theta_k} p\left(\mathbf{x}^{(i)}(t)|\theta_k, \mathbf{z}^{(i)}_{1:t}\right) p\left(\theta_k|\mathbf{z}^{(i)}_{1:t}\right) \tag{71}$$

Per renderne la gestione trattabili in termini computazionali, la densità multimodale (71) viene approssimata con un Gaussiana, la cui descrizione in potenza è

[18] Con un unica limitazione: esso può essere ancora solo usato per fare tracking su gruppi di oggetti con la medesima *modalità*.

[19] In generale si avrà sempre $m \neq n$.

generata secondo le

$$\begin{cases} \widehat{\mathbf{x}}^{(i)}(t) = \sum_{\theta_k} \widehat{\mathbf{x}}^{(i)}_{\theta_k}(t) \cdot p(\theta_k|\mathbf{z}^{(i)}_{1:t}) \\ P^{(i)}(t) = \sum_{\theta_k} P^{(i)}_{\theta_k}(t) \cdot p(\theta_k|\mathbf{z}^{(i)}_{1:t}) + \\ \quad + \sum_{\theta_k} \left(\widehat{\mathbf{x}}^{(i)}_{\theta_k}(t) - \widehat{\mathbf{x}}^{(i)}(t)\right) \left(\widehat{\mathbf{x}}^{(i)}_{\theta_k}(t) - \right. \\ \quad \left. - \widehat{\mathbf{x}}^{(i)}(t)\right)^T \cdot p(\theta_k|\mathbf{z}^{(i)}_{1:t}) \end{cases} \tag{72}$$

Ciò rende possibile ipotizzare all'istante t una a posteriori Gaussiana e iterare la procedura appena descritta.

L'unico punto rimanente ancora da chiarire riguarda il calcolo delle densuità a posteriori delle associazioni $p(\theta_k|\mathbf{z}_{1:t})$. Si assume di disporre della a priori $p(\theta_k)$, che solitamente risulta semplicemente definibile fissando il parametro P_D. Le associazioni all'istante t sono condizionalmente indipendenti dalle misure fino al tempo $t-1$: ciò permette di fattorizzare $p(\theta_k|\mathbf{z}_{1:t})$, sfruttando il teorema di Bayes, come:

$$\begin{aligned} p(\theta_k|\mathbf{z}_{1:t}) &= c \cdot p\left(\mathbf{z}(t)|\theta_k, \mathbf{z}_{1:t-1}\right) p\left(\theta_k|\mathbf{z}_{1:t-1}\right) \\ &= c \cdot p\left(\mathbf{z}(t)|\theta_k, \mathbf{z}_{1:t-1}\right) p\left(\theta_k\right) \end{aligned} \tag{73}$$

dove c è una costante di normalizzazione ininfluente.

D'ora in poi si snellirà la notazione: verranno eliminati gli indici temporali e con $\mathbf{Z}^-$ e $\mathbf{Z}$ si indicheranno rispettivamente la storia passata e presente, delle uscite. Analogamente, con $\mathbf{x}_i^-$ verrà indicata la predizione dello stato al tempo t data la storia $\mathbf{Z}^-$ e con P_i^- la sua varianza condizionata d'errore.

Si introduce l'insieme D_θ contenente gli indici dei target le cui misure sono state rilevate secondo l'ipotesi θ; parallelamente si partizionano le misure $\mathbf{z}$ al tempo t in due insiemi: $\mathbf{z}_{T,\theta} := \left\{\mathbf{z}_{j(i,\theta)}\ ,\ i \in D_\theta\right\}$, il sottoinsieme delle misure 'vere', ovvero associate a qualche target, ed il complementare, $\mathbf{z}_{F,\theta}$, delle misure associate a clutter. Infine si crea l'insieme degli indici dei target ipotizzati come sottoposti ad occlusione $\mathbf{M}_\theta$.

Si *postula* quindi indipendenza tra i target[20] e si fattorizza $p\left(\mathbf{z}|\theta, \mathbf{Z}^-\right)$ nella forma:

$$p\left(\mathbf{z}|\theta, \mathbf{Z}^-\right) = p\left(\mathbf{z}_{T,\theta}|\theta, \mathbf{Z}^-\right) p\left(\mathbf{z}_{F,\theta}|\theta, \mathbf{Z}^-\right) \tag{74}$$

che permette di studiare i due fattori sparatamente.

Il termine $p\left(\mathbf{z}_{F,\theta}|\theta, \mathbf{Z}^-\right)$, che descrive il clutter, viene solitamente considerato uniforme nel volume V di spazio operativo, ovvero:

$$p\left(\mathbf{z}_{F,\theta}|\theta, \mathbf{Z}^-\right) = \left(\frac{1}{V}\right)^{N_F(\theta)} \tag{75}$$

[20] Ipotesi chiave del ragionamento.

dove $N_F(\theta)$ è la numerosità di false misure sotto l'ipotesi θ.

Per quanto riguarda invece il termine $p\left(\mathbf{z}_{T,\theta}|\theta, \mathbf{Z}^-\right)$, osservando che

$$\mathbf{z}_{T,\theta} = \left\{\mathbf{z}_i\ ,\ i \in D_\theta\right\}$$

si definiscono i vettori di misure $\mathbf{y}_{D_\theta} = \left\{\mathbf{z}_i\ ,\ i \in D_\theta\right\}$ e $\mathbf{y}_{M_\theta} = \left\{\mathbf{z}_i\ ,\ i \in M_\theta\right\}$, contenenti rispettivamente i target rilevati e quelli occlusi. Si può ottenere quindi

$$p\left(\mathbf{z}_{T,\theta}|\theta, \mathbf{Z}^-\right) = \left[p\left(\mathbf{y}_{D_\theta}|\mathbf{Z}^-\right)\right]_{\mathbf{y}_i = \mathbf{z}_{j(i,\theta)}, i \in D_\theta} \tag{76}$$

come la marginale di

$$p\left(\mathbf{z}_1, \ldots, \mathbf{z}_{N_T}|\mathbf{Z}^-\right) = p\left(\mathbf{z}_{D_\theta}, \mathbf{z}_{M_\theta}|\mathbf{Z}^-\right) \tag{77}$$

rispetto $\mathbf{y}_{M_\theta}$:

$$p\left(\mathbf{z}_{D_\theta}, |\mathbf{Z}^-\right) = \int p\left(\mathbf{z}_{D_\theta}, \mathbf{z}_{M_\theta}|\mathbf{Z}^-\right) d\mathbf{y}_{M_\theta}. \tag{78}$$

Sotto l'ipotesi di indipendenza dei target essa fattorizza in

$$p\left(\mathbf{z}_{T,\theta}|\theta, \mathbf{Z}^-\right) = \prod_{i \in D_\theta} \left[p\left(\mathbf{y}_i|\mathbf{Z}^-\right)\right]_{\mathbf{y}_i = \mathbf{z}_{j(i,D_\theta)}} \tag{79}$$

i cui termini $p\left(\mathbf{y}_i|\mathbf{Z}^-\right)$ sono le predizioni delle posizioni di target i-esimi date le misure passate.

Dall'assunzione (69) segue che $p\left(\mathbf{y}_i|\mathbf{Z}^-\right)$ è Gaussiana, di media $\widehat{\mathbf{y}}_i = \mathrm{h}(\widehat{\mathbf{x}}_i^-)$ e covariana matriciale $\Sigma_i^- = H_i P_i^- H_i^T + R_i$.

Assumere la dinamica di ogni target indipendente da quella degli altri è di vitale importanza per la formulazione che è stata fatta del filtro. Includere dei vincoli tra i diversi oggetti comporta notevoli complicazioni, soprattutto nella marginalizzazione della $p\left(\mathbf{z}_{T,\theta}|\theta, \mathbf{Z}^-\right)$, che non può più essere fattorizzata. In letteratura si trovano diversi esempi di filtri che gestiscono, per esempio, modelli di forma, che solitamente adottano strumenti come i filtri particellari precedentemente introdotti.

E. Joint Likelihood Data Association

Si focalizza ora l'attenzione sull'applicazione che ha motivato tutto la presente trattazione.

Uno degli obiettivi secondari del progetto era quello di *integrare nel sistema di misura i dati provenienti da più sensori*. Come è già stato evidenziato in precedenza, il JPDAF non permette di gestire simultaneamente modalità di misura differenti . Anzi, proprio questa limitazione si era dimostrata critica per il PDAF, in presenza di distractions, motivando lo sviluppo di un livello di logica superiore sfociata poi nella formulazione del JPDAF.

Esiste però un approccio che permette quest'integrazione, ottenendo inoltre una serie di altri vantaggi non offerti dal JPDAF.
Si consideri la *joint image likelihood* dato lo stato di un insieme di T oggetti,$p(\mathbf{I}|\mathbf{I}_1, \ldots, \mathbf{X}_T)$; nella costruzione classica del JPDAF si assume che essa possa essere fattorizzata come

$$p\left(\mathbf{I}|\mathbf{X}_1, \ldots, \mathbf{X}_T\right) = p\left(\mathbf{I}|\mathbf{X}_1\right) \ldots p\left(\mathbf{I}|\mathbf{X}_T\right)$$

tale assunzione risulta lecita in molti casi, ma nella pratica si nota che in questo modo le stime di posizioni, angoli e scaling tendono a soffrire di sistematici bias quando più target si avvicinano o addirittura si sovrappongono nell'immagine. Tali configurazioni in effetti sono proprio esempi di quelle condizioni in cui le proiezioni dei vari target nell'immagini risultano fortemente dipendenti tra loro.

Questa è la motivazione che ha portato in letteratura a sviluppare metodi alternativi per la stima della densità congiunta $p\left(\mathbf{I}|\mathbf{X}_1, \ldots, \mathbf{X}_T\right)$: metodi che nello specifico integrassero strategie per stimare anche le profondità relative degli oggetti e permettessero di fondere informazioni provenieti da processing diversi delle immagini ingresso, ovvero modalità di misura differenti.
Uno dei metodi che implementa tale approccio è per esempio il *Joint Likelihood Data Association Filter*. Esso può essere un buon punto di partenza per l'implementazione di un sistema, molto complesso, di tracking per il progetto NAVLAB, perchè permetterebbe di gestire ben tre tipi di misure differenti in contemporanea: i dati tridimensionali generati dal sistema di motion capture e le immagini catturate dalle telecamere del medesimo sistema, ed infine i dati prodotti dalla rete di sensori disposta sul piano dove i robot saranno in futuro liberi di muoversi.

Si procede ora con l'implementazione in Matlab di un sistema funzionanante di tracking. Per la sua implementazione sono state sfruttate molti dei contenuti più 'particolari' presentati nel testo più alcune idee originali, che renderebbero già possibile il suo utilizzo in laboratorio.

VI. FASE SPERIMENTALE

A. Introduzione

La parte implementativa di tale progetto si propone di tracciare il moto arbitrario di un oggetto ripreso in una sequenza video. Questa è un'operazione chiave per una grande quantità di applicazioni in diversi domini. La cosiddetta 'Looking at people', ad esempio, è al momento una delle più attive aree applicative per quanto riguarda la computer vision: si tratta di sistemi in grado di interagire in maniera intelligente in ambienti popolati da essere umani che richiedono meccanismi di sorveglianza. Strutture di controllo di questo tipo potrebbero essere usate per garantire determinate condizioni di sicurezza, poichè in grado di identificare le azioni di un individuo e, di conseguenza, segnalare eventuali comportamenti sospetti.
Analogamente, risulterebbe utile tracciare il moto dei veicoli, svolgendo un ruolo di primo piano nella realizzazione di sistemi autonomi, nella robotica mobile e nell'interazione uomo-macchina.

B. Tracking

Il tracciamento ha lo scopo di ricostruire la traiettoria di un oggetto data una certa sequenza di immagini, da cui estrarre misure del sistema in esame. Generalmente viene calcolata la posizione, nell'immagine o nel mondo, di alcuni punti di particolare interesse e di cui si vuole seguire l'evoluzione. Le situazioni da analizzare possono essere complesse, richiedendo di operare su più oggetti con differenti colori, forme e dimensioni che interagiscono in modo dinamico tra di loro e con l'ambiente. L'obiettivo è riuscire ad ottenere una buona stima dello stato di un determinato oggetto a partire dall'osservazione di questi scenari, cercando di considerare contemporaneamente tutte le informazioni di cui si dispone e di ovviare alla presenza di eventuali disturbi come occlusioni o rumore. Per fare ciò possono essere utilizzati una vasta serie di algoritmi tra cui il *particle filter*, un metodo **sequential Monte Carlo** che generalizza i tradizionali metodi basati sul filtro di Kalman ottenendo ottimi risultati. In molte applicazioni, infatti, si vorrebbe identificare un determinato insieme di oggetti e tracciare i loro movimenti per tutta la video-sequenza evitando di riattivare, ad ogni frame, i meccanismi per la rilevazione della posizione di questi ultimi. Il problema del tracciamento viene, quindi, affrontato come un problema di inferenza in cui mantenere un'accurata rappresentazione a posteriori della posizione dell'oggetto in maniera efficiente.

C. Generic object tracking Motion without models

Possono essere utilizzate molte tecniche per il tracciamento degli oggetti e la maggior parte di queste richiede un modello di moto predefinito.

La tecnica che viene presentata, invece, trascende dalla necessità di avere modelli prestabiliti e fa in modo che ci possano essere assunzioni più deboli circa il moto della camera e degli oggetti d'interesse, permettendo quindi di tracciare oggetti che si muovono in maniera arbitraria senza particolari difficoltà.
Ciò rende questo modo di operare indubbiamente più duttile, e di conseguenza applicabile in un numero maggiore di situazioni.

Invece di cercare di costruire un modello del moto, con questo approccio si cerca di migliorare la distribuzione a priori e le predizioni proposte. Si assume che la posizione attuale dell'oggetto sia identificabile parzialmente mediante segmentazione e si procede alla generazione delle predizioni utilizzando le informazioni ricavabili dal SIFT, che infatti si basa su '*feature matching*'. Il solo SIFT non permette autonomamente di fare tracking poichè fornisce risultati attendibili solo con sequenze relativamente brevi di immagini, in seguito ad una sua intrinseca ed inevitabile propagazione degli errori. L'errore al *frame* n è cumulativo:

$$\epsilon_n = \sum_{i=1}^{n-1} \epsilon_i,$$

e non esiste, a tal proposito, un meccanismo di recupero interno all'algoritmo.

Per questo motivo si farà uso di una combinazione del SIFT con un *particle filter*, che permetterà al sistema di correggersi ed eliminare quelle ambiguità nel moto che il solo SIFT non è in grado di gestire.

Il *particle filter* permette di generare e validare una serie di ipotesi circa la posizione dell'oggetto e, di conseguenza, di ottenere una stima più robusta della stessa.

D. SIFT (Scale invariant feature transform)

In un'immagine esistono punti con diverse proprietà che li rendono adatti al *matching* tra vari *frame* di uno o più oggetti. Le proprietà che li caratterizzano sono invarianti rispetto a rotazioni o a cambiamenti di scala e, seppur parzialmente, anche a variazioni della luminosità. Inoltre, sono ben localizzabili sia nel dominio del tempo, che della frequenza e questo riduce la probabilità di errori dovuti a rumore o ad occlusioni. Un largo numero di *features* possono essere estratte con algoritmi efficienti; tra questi vi è il SIFT, che trasforma le informazioni presenti in un'immagine in coordinate, relative esclusivamente alle *features*. Possiamo distinguere i più importanti passi di computazione di questo processo:

- **Scale-space extrema detection**: vengono identificati potenziali punti di interesse, invarianti rispetto a rotazioni e variazioni delle dimensioni;
- **Key-point localization**: per ogni regione candidata, viene stabilito un modello per determinare posizione e dimensioni della stessa;
- **Orientation assignment**: una o più orientazioni sono assegnate ad ogni punto chiave sulla base della direzione del gradiente in quella particolare zona dell'immagine;
- **Keypoint descriptor**: i punti chiave sono trasformati in una rappresentazione che tiene conto di significativi distorsioni nelle forme e cambiamenti nella luminosità.

Le SIFT *feature* di un immagine iniziale vengono estratte; successivamente, per ogni nuova immagine, il *matching* avverrà confrontando individualmente tutte le *feature* della nuova immagine con quelle dell'immagine di riferimento che inizialmente corrisponde appunto all'immagine iniziale mentre, in corso di esecuzione, viene aggiornata in base a precise assunzioni e strategie. Il confronto avviene sulla base della distanza Euclidea dei rispettivi vettori-feature delle coppie candidate.

1) Particle filter e SIFT transition model: Ciò che si intende calcolare attraverso l'uso del *particle filter* e del SIFT è $p(x_n|y_{1:n})$ dove con x si indica lo stato dell'oggetto e con y le osservazioni nella sequenza di frame. L'approccio tradizionale consiste nell'ottenere una stima ricorsiva usando le informazioni a priori dello stato e le equazioni di modifica sotto illustrate. A questo punto, attraverso la marginalizzazione di tutte le stime passate, si ottiene la previsione condizionata dalla nuova osservazione allo stadio n

$$p(x_n|y_{1:n}) = \int p(x_n|x_{n-1})p(x_{n-1}|y_{1:n-1})dx_{n-1}$$

$$p(x_n|y_{1:n}) = \frac{p(y_n|x_n)p(x_n|y_{1:n-1})}{Z}.$$

Per quanto riguarda la risoluzione dell'integrale, può essere usato il metodo di Monte Carlo tramite l'utilizzo della seguente equazione:

$$Z = \int \frac{f(x_{1:n})}{q(x_{1:n})} \cdot q(x_{1:n})$$

$$\hat{Z_n} = \frac{1}{N} \sum_{i=1}^{N} w(x_{1:n}^{(i)}),\ x_{1:n} \approx q(x_{1:n}).$$

A questo punto, è possibile derivare un approccio ricorsivo per calcolare il peso dei vari campioni estratti:

$$w_n = \frac{f(x_{1:n})}{q(x_{1:n})}$$

$$w_n = \frac{f(x_{1:n})}{q(x_n|x_{1:n-1}) \cdot f(x_{1:n-1})} \cdot w_{n-1}$$

Infine, considerando valida l'ipotesi di Markov: $f(x_{1:n}) = p(x_{1:n}, y_{1:n})$ e applicandola all'equazione precedente, ed assumendo inoltre un moto uniforme a priori, tale che $p(x_n|x_{n-1})$ sia guidata da una distribuzione uniforme, si può dedurre:

$$w_n = \frac{p(y_n|x_n) \cdot p(x_n|x_n - 1)}{q(x_n|x_{1:n-1}, y_{1:n})} \cdot w_{n-1}$$

$$w_n = p(y_n|\tilde{x}_n) \cdot w_{n-1}.$$

Operando in questa modo, non si costruisce il modello che meglio approssima il comportamento di un dato oggetto, ma si cerca di migliorare la distribuzione proposta. Non si fanno, perciò, assunzioni circa il moto dell'oggetto: ad ogni stadio si osserva il sistema e si deduce il comportamento dell'oggetto, volta per volta, adattandosi con facilità a diverse varietà di cambiamenti. Il sistema si può adattare a cambiamenti nel tempo del tipo di moto, in particolare consentendo la possibilità di eseguire tracking con variazioni di scala e rotazioni.

La *particle* i al tempo n sia suddivisa in due componenti, x_{in} e y_{in}, rappresentanti le sue coordinate nel piano immagine.

Si denota con n_{sift} il numero dei punti chiave, individuati tramite il metodo omonimo precedentemente illustrato, che rappresentano le coppie di punti corrispondenti tra il *frame* $n-1$ ed il *frame* n.
Con dx_{jn} e dy_{jn} si denotano invece le traslazioni del punto chiave j lungo gli assi x e y, misurate confrontando le corrispondenze rilevale. La posizione predetta per la particella i-esima nel frame sarà data dalla somma tra la sua posizione nel frame $n-1$ ed il moto stimato dall'algoritmo SIFT:

$$\mu_{x_n}^{(i)} = \left(\frac{1}{N_{SIFT}} \sum_{j=1}^{N} dx_n^{(j)} \right) + x_{n-1}^{(i)}$$

$$\mu_{y_n}^{(i)} = \left(\frac{1}{N_{SIFT}} \sum_{j=1}^{N} dy_n^{(j)} \right) + y_{n-1}^{(i)}$$

Ad ognuna di tali stime viene aggiunto rumore additivo Gaussiano, per tenere conto dell'effetto del numero delle effettive corrispondenze tra le immagini. La varianza di tale Gaussiana viene calcolata come:

$$\sigma_{x_n} = \alpha_x \cdot \log\left(\frac{1}{1 - e^{\sigma_{edx_t}}} \right) + \beta_x \cdot e^{-N_{SIFT}}$$

$$\sigma_{y_n} = \alpha_y \cdot \log\left(\frac{1}{1 - e^{\sigma_{edy_t}}} \right) + \beta_y \cdot e^{-N_{SIFT}}$$

dove σ_{edx_t} e σ_{edy_t} sono le varianze campionarie delle misure di traslazione ottenute con SIFT, ed il parametro libero α viene tarato sprimentalmente; il secondo termine della somma rappresenta l'influenza sulla varianza del numero dei campioni. Un numero piccolo di questi ultimi fa aumentare l'incertezza della misura, ma il suo contributo tende rapidamente a zero[21] all'aumentare del numero di campioni. Si noti che i parametri α che β vanno tarati in base alla risoluzione dell'immagine e non alla tipologia dell'oggetto che viene tracciato.

E' ora possibile stimare al meglio la posizione dell'oggetto d'interesse come:

$$\tilde{x} \approx N(\mu_{x_n}, \sigma_{x_n})$$

$$\tilde{y} \approx N(\mu_{y_n}, \sigma_{y_n})$$

dove μ_{x_n} e μ_{y_n} sono le medie delle aspettazioni per le particelle al frame n date le particelle al frame $n-1$. A questo punto è possibile procedere al calcolo dei nuovi pesi con l'espressione già citata, e considerando per p una distribuzione esponenziale del tipo:

$$p(y_n^{(i)}|\tilde{x}_n^{(i)}, \tilde{y}_n^{(i)}) \propto e^{-\lambda \cdot \zeta[H_0, H_n^{(i)}]}$$

$$\zeta[H_0, H_n] = \left[1 - \sum_{j=1}^{N} \sqrt{H_0^{(j)} \cdot H_n^{(j)}} \right]^{\frac{1}{2}}$$

Dove $[H_0, H_n]$ è la distanza di Bhattacharyya tra l'istogramma ottenuto dal *particle i-esimo* al passo n e quello di riferimento (considerato la migliore rappresentazione dell'oggetto). Una volta calcolati e normalizzati i pesi al passo n è possibile sopprimere quelli troppo bassi e duplicare quelli che meglio spiegano i dati misurati.

E. Tutorial Matlab

Prima di presentare il codice adeguataemente commentato, che esegue formalmente quanto appena visto, si decide di presentare una breve descrizione concreta del funzionamento generale del programma in simulazione. L'unico requisito per il funzionamento del programma è che i pacchetti di immagini su cui effettuare tracking, siano nominati in successione numerica: *1.jpg, 2.jpg, ecc.* in cui la prima immagine rappresenta l'immagine di riferimento iniziale che si intende inseguire, già ritagliata appositamente (ad esempio un volto umano). E' quindi possibile applicare facilmente il programma anche ad un video, semplicemente frammentadolo opportunamente.
Il programma si avvia eseguendo il file **demo.m** Questo a sua volta invoca la funzione **avvia.m** passando come

[21] Ed il parametro β consente di determinare quanto rapidamente il suo contributo tende a zero.

parametri la directory contenente le immagini su cui effettuare il tracking e il nome del video su cui visualizzare i risultati in uscita. Tale funzione, dopo opportuni controlli sui nomi dei file, comincia ad elaborare l'immagine iniziale *1.jpg*. Da quest'ultima estrae le dimensioni, il centro e altri parametri che userà per inizializzare oppurtamente le figure in uscita. In seguito, dopo avere inizializzato altre variabili di controllo, matrici di particelle e pesi inizia l'algoritmo di confronto; viene invocata la funzione **match.m** a cui vengono passati come parametri le due immagini in esame. Tale funzione invoca subito per ogni immagine la funzione **sift.m** che restituisce alla precedente

- l'immagine in formato double;
- i descrittori dei k punti invarianti rilevati, ovvero una matrice di k righe e 128 colonne (si noti che la matrice è normalizzata all'unità; dettaglio che verrà utilizzato in seguito);
- una matrice di k righe e 4 colonne contenente, per ogni punto invariante, posizioni (x e y), un fattore di *scaling* e uno di orientazione (tra $-\pi$ e π).

Ottenuti questi dati per ogni immagine, la funzione **match.n**, dopo aver preso nota di quanti siano i punti invarianti, esegue il vero matching: calcola il prodotto interno dei descrittori normalizzati (piuttosto che calcolare le distanze euclidee; si noti che il rapporto di angoli è un'approssimazione vicina al rapporto di distanze euclidee per piccoli angoli) e ricava la distanza dei vettori, grazie all'$\arccos$ (qui si usa quindi il fatto che il modulo dei vettori sia unitario). Ordina in maniera crescente le distanze e accetta un match se la sua distanza è minore di *distRatio* volte della seconda distanza (dove *distRatio* è un valore fornito da letteratura e confermato da numerose prove sperimentali). La funzione **match.m** restituisce quindi alla funzione **avvia.m** una matrice di 4 colonne e m righe, dove ogni riga corrisponde ad un matching tra i due file, mentre le colonne rappresentano le coordinate dei due punti corrispondenti sui due file immagine. Successivamente vengono rilevati valori di spostamento e varianza. Si effettuerà anche un confronto su due modi di calcolare la varianza.
Segue la trattazione teorica, aggiornando particelle e pesi secondo opportuna distanza degli istogrammi. Avendo una decrizione stocastica può accadere (in realtà si verifica raramente) che alcune particelle escano dallo spazio di lavoro; per ovviare a tale pesante incoveniente è stato predisposto un opportuno controllo che, come intuito suggerisce, mette il relativo peso a zero. In seguito, si sopprimono i pesi peggiori e si duplicano i migliori aggiornando ovviamente, anche le relative particelle. Infine si calcola il centro delle particelle e, attorno ad esso, grazie alla funzione **rettangolo.m** si plottano due quadrati: uno di dimensione fissa (quella dell'immagine iniziale) ed un altro dipendente dalla varianza. Quindi, in caso di occlusioni o modifica eccessiva dell'immagine si noterà, oltre ad una dispersione delle particelle, anche un allargamento di tale quadrato in seguito all'aumento della varianza. Infine, seguono alcune semplicissime righe chiavi per l'esecuzione del programma. Nessun articolo in letteratura presenta il problema del continuo aggiornamento dell'immagine. Si effettua quindi un controllo che verrà giustificato nelle scelte progettuali, per determinare tale aggiornamento. La funzione **showkeys.m** è una funzione utilizzata per visualizzare i punti chiave nell'immagine. Il file **avviamulti.m** è utilizzato per effettuare il multitarget ed è costruito analogamente al file **avvia.m**

1) Scelte progettuali e giustificazioni:

1) Per quanto riguarda il confronto tra i due tipi di varianze, uno sarà quello fornito precedetemente nell'analisi teorica, l'altro sara quello della varianza calcolata da MatLab. Ne risulterà una miglior efficienza di quest'ultimo perché nel caso pratico, quando $\hat{\sigma}$ è prossimo a 0, mentre $e^{\hat{\sigma}}$ è vicino ad 1 e $\frac{1}{1-e^{\hat{\sigma}}}$ è un numero molto grande, nonostante il logaritmo. Al contrario se $e^{\hat{\sigma}}$ e' maggiore di 1 l'argomento del logaritmo diventa negativo e di conseguenza il logaritmo non è definito. Si è riusciti ad ovviare a tali incovenienti, ma ottenendo risultati comunque peggiori rispetto alla varianza ottenuta grazie a MatLab. Infatti, lo scopo della formula precedentemente illustrata é chiaro: mediare tra la $\hat{\sigma}$ dei punti calcolati con il **SIFT** ed il fatto che i punti potrebbero essere troppo pochi (il secondo termine tende rapidamente a zero all'aumentare di **num_match**). Assumendo però che le immagini in esame siano tali da ottenere almeno una decina di punti, è possibile tralasciare il secondo termine ed utilizzare come σ (cioè l'incertezza della misura) direttamente quella dei punti calcolati;
2) Per quanto riguarda l'aggiornamento dell'immagine, l'applicazione del **SIFT** combinato al *particle filter* ha proprio senso per seguire l'evoluzione e le modifiche dell'oggetto. Come precedentemente descritto, non si ha un modello; ma viene introdotto uno 'stato'. L'introduzione dell'informazione a priori dev'essere combinata con un opportuno aggiornamento dell'immagine di riferimento; si stabilisce di aggiornare l'immagine iniziale se il numero di match supera $\frac{1}{6}$ dei keypoints dell'immagine di riferimento; dopo opportune prove sperimentali,

si decide che se l'intersezione perfetta tra le due immagini supera $\frac{1}{6}$ l'immagine viene aggiornata, in caso contrario l'immagine non sarà abbastanza dettagliata o considerata somigliante da essere aggiornata e non sarà impostata come nuova immagine di riferimento. Conoscendo perfettamente le sequenze di immagini a disposizione, e sapendo il lento mutare degli oggetti nel corso dei frame, si è abbastanza tolleranti a porre questo limite ad $\frac{1}{6}$ (funziona molto bene anche con rapporti minori e maggiori. Per altre applicazioni, il suggerimento è quello di effettuare un'accurata analisi prima di dimensionare tale rapporto, perché questo è a tutti gli effetti un limite di accettazione dell'immagine nell'aggiornamento. Infatti, nel caso in cui si effettua multitarget poniamo tale limite pari ad $\frac{1}{2}$; altrimenti quando il *pallone-primo* target si sovrappone al *canestro-secondo* target si aggiornerebbe l'immagine provocando seri problemi;

3) si utilizza la distanza Euclidea nell'aggiornamento dei pesi, in quanto si e' rivelata migliore di quella Bhattacharyya;
4) dopo numerose prove sperimentali i parametri relativi all'analisi teorica precdentemente effettuata sono stati cosi impostati:
   a) $\alpha = 1$;
   b) $\beta = 1$;
   c) $num\_particelle = 200$;
   d) $\lambda = 0.001$ (cosi facendo, i pesi assumono il giusto significato non disperdendosi troppo al di fuori dell'area di lavoro).

F. Risultati e novità introdotte

Nella lettetatura analizzata sono stati trovati alcuni semplici applicativi che sviluppava il visual tracking di oggetti in bianco e nero su sfondo bianco. Queste assunzioni ne semplificavano notevolmente l'implementazione. Per quanto riguarda il software in esame, esso esegue il tracking su qualsiasi tipo di oggetto applicato a qualsiasi sfondo. Inolte, la combinazione di **SIFT** e del *particle filter* ha consentito, oltre che di effettuare del semplice tracking su traslazione, anche di inseguire l'oggetto in caso di cambio di scala, di leggeri cambi di luce e sopprattutto in rotazione. COsì terminata l'implementazione, si è reso possibile, dopo opportune modifiche, effettuare il multitarget tracking.

Di seguito si presentano i risultati ottenuti.

Le seguente immagini si propongono di testare le potenzialità del **SIFT** nel mondo reale: le immagini prese in esame sono infatti due foto, eseguite in maniera differente, dello stesso scenario. Nonostante le visibili differenze dovuti anche agli agenti esterni (vento sulla vela), il **SIFT** trova comunque un elevato numero di corrispondenze.

Di seguito vengono presentate invece alcune sequenze d'immagini che si propongono di esemplificare le casistiche studiate: In ogni immagine saranno presenti 3 figure:

- in alto a sinistra è presentel'immagine di riferimento;
- in basso a destra è presente l'immagine di riferimento con la visualizzazione dei keypoints;
- a destra è presente l'immagine su cui effettuare il tracking dell'oggetto.

1) Occlusioni: Si noti l'aggiornamento dell'immagine di riferimento quando la somiglianza è 'sufficiente'. Ovviamente, nonostante il tracking sia comunque buono (molti keypoints presenti sulla parte superiore dei capelli) l'immagine non viene aggiornata quando il foglio nasconde il viso.

2) Traslazioni: Tali immagini rappresentano una semplice traslazione, dove si effettua anche uno zoom del viso. Inoltre, tale sequenza si presta anche ad un confronto tra i metodi di calcolo della varianza (non tutti i pacchetti di immagini consentivano il tracking con il metodo di calcolo della varianza proposto in letteratura). Si può notare la maggiore dispersione dei *particle* dovuta all'aumento della varianza con consequente allargamento della finestra rossa.

3) Scaling: In questa sequenza si propone di studiare l'effetto dello scaling sul tracking. Oltre all'ovvia dispersione dei *particle*, si riscontra un ottimo inseguimento anche con un elevato fattore di scaling; si noti infine il corretto aggiornamento dell'immagine di riferimento finchè la dimensione non diventa troppo piccola e l'aggiornamento finale quando il pallone riassume dimensioni consone.

4) Rotazione: Questo è forse uno dei risultati piu sensazionali ottenuti. Oltre a verificare il corretto tracking, si è voluto ricercare il massimo delle prestazioni e delle potenzialità del software; la combinazione **SIFT** e *particle filter* consente di rilevare e eseguire anche rotazioni di 15 gradi (vengono visualizzate qui solo poche immagini ma, nel caso in esame, sono state efettuate le seguenti rotazioni, in gradi, successive 1-1-1-2-2-3-3-4-4-5-5-7-7-8-9-10-11-12-15).

5) Multiple Target: Quest'ultima sequenza di immagini rappresenta il multitarget. Oltre alle ovvie considerazioni su varianza e aggiornamento, si vuole sottolineare un grosso scoglio implementativo: se si fosse utilizzata la stessa frazione di aggiornamento ($\frac{1}{6}$ per il target 1) anche nel target 2, al sovrapporsi dei due target, si avrebbe ottenuto l'errato aggiornamento riferito all'immagine di riferimento del target 2: si sarebbe quindi salvata come immagine di riferimento il canestro con una frazione di pallone sovrapposta, causando grossi problemi alla successiva esecuzione; in quanto si sarebbero rilevati conseguenti match tra 'frazione di pallone sovrapposto salvato nell'immagine di riferimento nel target 2' e 'pallone vero'. Così il rapporto di aggiornamento è stato impostato ad $\frac{1}{2}$ e, di conseguenza, molto più selettivo. Per questo motivo, onde evitare di perdere svariate energie per rilevare un errore in apparenza inspiegabile, si suggerisce di fare un'accurata scelta in fase di analisi.

G. Codice MatLab

Si riportano di seguito i files MatLab usati per la fase sperimentale.

demo.m

```
clc;
clear all;
close all;

%% Definisco il numero di particelle e una variabile che utilizzerò per
%% confrontare le prestazioni di due diversi metodi di calcolo per la
%% varianza.

global num_particelle;
global confrontovarianze;

num_particelle = 200;
confrontovarianze = 0;

% avvia('.\image\gabriella_occlusioni\','gabri_occlu');
%
% avvia('.\image\gabriella_traslazione\','gabri_trasla');
%
% avvia('.\image\basket_scaling\','basket_scal');
%
% avvia('.\image\gabriella_rotazione\','gabri_rotaz');
%
% avvia('.\image\sift_demobarca\','sift_barca');
%
% avviamulti('.\image\basket_multitarget\','basket_multi');
```

avvia.m

```
%% Accede alle immagini contenute in 'directory' ed inizia il tracciamento.
%% Le immagini devono essere in formato jpg e con nomi '1.jpg', '2.jpg'...

function [] = avvia(directory,nomevideo)

global num_particelle;
global confrontovarianze;

global target2;
global inizio;
global keys_inizio;
global keys;

%% Controllo dei nomi dei file immagine.

[nome_file, err] = sprintf('%s1.jpg',directory);
if(~exist(nome_file,'file'))
    error('Immagine iniziale non trovata');
    return;
end

[nome_file2, err] = sprintf('%s2.jpg',directory);
if(~exist(nome_file2,'file'))
    error('Immagine iniziale non trovata');
    return;
end

%% Memorizzo parametri relativi all'immagine di riferimento.

I1 = imread(nome_file);
IRif = I1;
dim = size(I1);
altezza = dim(1);
lunghezza = dim(2);
centro = [altezza/2 lunghezza/2];

%% Ritaglio l'oggetto per farne l'istogramma che sarà usato
%% successivamente per valutare la correttezza del rettangolo proposti da
%% ogni particella.

H01 = hist(double(I1(1:altezza,1:lunghezza,1)),100);
H02 = hist(double(I1(1:altezza,1:lunghezza,2)),100);
H03 = hist(double(I1(1:altezza,1:lunghezza,3)),100);

%% Inizializzazione dell'immagine.

I2 = imread(nome_file2);

scrsz = get(0,'ScreenSize');
figura1 = figure('Position',[1 1 scrsz(3) scrsz(4)]);

subplot(2,2,[2 4]);
imshow(I2);

subplot(2,2,1);
imshow(I1);

subplot(2,2,3);
[im1, des1, loc1] = sift(I1);
showkeys(im1, loc1);

%% Inizializzazione di pesi e particelle.
%% Le particelle vengono inizializzate con la posizione centrale di ogni
%% immagine ed ogni peso è posto ad 1.

particelle = repmat(centro,num_particelle,1);
```

```
pesi = repmat(1,num_particelle,1);

%% Inzio l'algoritmo di confronto: dapprima con le due immagini iniziali e
%% poi quelle successive.

file_ind = 2;
[nome_file, err] = sprintf('%s2.jpg',directory);

%% Inizializzo vatiabili che tengono riferimento rispettivamente di:
%% - Prima immagine
%% - Target in utilizzo
%% - Numero di match tra due immagini

inizio=1;
target2=0;

num_match=0;

while exist(nome_file,'file')

    %% Confronto le due immagine correnti.

    I2 = imread(nome_file);

    %% Calcolo dei punti corrispondenti con il sift.
    %% Ogni riga di L contiene: ay ax by bx
    %% dove a e' un punto in I1 e b il corrispondente in I2.

    L = match(I1,I2);
    num_match = size(L,1);

    %% Spostamento stimato in base ai punti calcolati dal sift: se non ci
    %% sono corrispondenze i quadrati devo rimanere fermi. Inoltre lo
    %% spostamento viene calcolato rispetto alla posizione precedente, con
    %% conseguente differenziazione nel caso si sia nel primo matching.

    if (num_match==0)
        dx = 0;
        dy = 0;
        dx_medio = 0;
        dy_medio = 0;
    else
        if (inizio==1)
            dx = L(:,4) - L(:,2);
            dy = L(:,3) - L(:,1);
        else
            dx = L(:,4) - L(:,2) - (nuovo_centro(2) - lunghezza/2 + 1);
            dy = L(:,3) - L(:,1) - (nuovo_centro(1) - altezza/2 +1);
        end
        dx_medio = sum(dx)/num_match;
        dy_medio = sum(dy)/num_match;
    end

    %% Lo spostamento medio viene aggiunto ad ogni particella per
    %% l'estrazione della prossima posizione

    mi = particelle + repmat([dy_medio dx_medio],size(particelle,1),1);

    %% Calcolo la varianza in base agli spostamenti.

    sigma_stimato_x = cov(dx);
    sigma_stimato_y = cov(dy);

    %% Confronto due metodi per calcolare la varianza: il primo ottenuto da
    %% comandi matlab mentre il secondo ricavato da letteratura.

    %% Metodo 1

    sigma_x = sigma_stimato_x;
    sigma_y = sigma_stimato_y;

    %% Metodo 2

    if confrontovarianze==1
        alpha = 1;
        beta = 1;
        sigma_x = abs(alpha * log((1-exp(sigma_stimato_x))^-1) + beta * (exp(-num_match)));
        sigma_y = abs(alpha * log((1-exp(sigma_stimato_y))^-1) + beta * (exp(-num_match)));
    end

    %% Aggiorno le particelle utilizzando una distribuzione gaussiana di
    %% media e varianza precedentemente determinate.

    particelle(:,1) = normrnd(mi(:,1),repmat(sigma_y,num_particelle,1));
    particelle(:,2) = normrnd(mi(:,2),repmat(sigma_x,num_particelle,1));

    p_y = uint32(particelle(:,1));
    p_x = uint32(particelle(:,2));

    for i=1:num_particelle

        %% Per ogni particella individuiamo il relativo rettangolo e
        %% confrontiamo il suo istogramma con quello iniziale o di
        %% riferimento.

        %% Puo' talvolta succedere che alcune particelle siano fuori dallo
        %% spazio di lavoro o che il comando hist si trovi ad operare su
        %% matrici di ampiezza diversa(molto raramente);
```

```
    %% predispongo adeguati controlli mettendo i relativi pesi a zero.

    if ((p_x(i)-lunghezza/2+1) <= 0) || ((p_x(i)+lunghezza/2) > size(I2,2)) || ((p_y(i)-altezza/2+1) <= 0) || ((p_y(i)+altezza/2) > size(I2,1))
        pesi(i)=0;
    else
        Hi1 = hist(double(I2((p_y(i)-altezza/2+1):(p_y(i)+altezza/2), (p_x(i)-lunghezza/2+1):(p_x(i)+lunghezza/2),1)),100);
        Hi2 = hist(double(I2((p_y(i)-altezza/2+1):(p_y(i)+altezza/2), (p_x(i)-lunghezza/2+1):(p_x(i)+lunghezza/2),2)),100);
        Hi3 = hist(double(I2((p_y(i)-altezza/2+1):(p_y(i)+altezza/2), (p_x(i)-lunghezza/2+1):(p_x(i)+lunghezza/2),3)),100);
        if (size(H01)==size(Hi1)) & (size(H02)==size(Hi2)) & (size(H03)==size(Hi3))

            %% La distanza euclidea si è rivelata migliore di quella di
            %% Bhattacharyya nei nostri casi.

            distH = sqrt(sum(sum((Hi1 - H01).^2)) + sum(sum((Hi2 - H02).^2)) + sum(sum((Hi3 - H03).^2)));

            %% Come indicato dalla letteratura usiamo usiamo una distribuzione
            %% esponenziale per l'aggiornamento dei pesi.

            lambda =  0.001;
            prob = exp(-lambda * distH);
            pesi(i) = prob * pesi(i);
        else
            pesi(i)=0;
        end
    end
end

%% Normalizzazione dei pesi

somma = sum(pesi);
if somma > 0
    pesi = pesi./somma;
end

%% Soppressioni dei peggiori e duplicazione del migliore

[pMax PMInd] = max(pesi);
Soppressi = find(pesi < 0.8*pMax);
pesi(Soppressi) = pesi(PMInd);
particelle(Soppressi,1) = particelle(PMInd,1);
particelle(Soppressi,2) = particelle(PMInd,2);

%% Nuova normalizzazione (probabilmente i pesi sono cambiati)

somma = sum(pesi);
if somma > 0
    pesi = pesi./somma;
end

%% Calcolo il centro della nuova immagine identificata

nuovo_centro = mean(particelle);

%% Disegno l'immagine corrente

subplot(2,2,[2 4]);
hold off
imshow(I2);
hold on

%% Disegno il centro stimato,le particelle, un quadrato con dimensione
%% fissa pari a quella dell'immagine e uno con dimensione variabile,
%% dipendente dalla varianza.

plot(nuovo_centro(2),nuovo_centro(1),'*b');
plot(particelle(:,2),particelle(:,1),'*m');
rett = rettangolo(nuovo_centro, altezza, lunghezza);
plot(rett(:,2),rett(:,1),'y');
rett = rettangolo(nuovo_centro, altezza + 2*sigma_y, lunghezza + 2*sigma_x);
plot(rett(:,2),rett(:,1),'r');

drawnow;

%% Se l'immagine corrente ha un'elevata corrispondenza con la
%% precedente, aggiorno la mia immagine di riferimento, il numero di
%% punti chiave e gli istogrammi.

if num_match > keys_inizio/6
    I1 = I2((nuovo_centro(1)-altezza/2+1):(nuovo_centro(1)+altezza/2),(nuovo_centro(2)-lunghezza/2+1):(nuovo_centro(2)+lunghezza/2),:);
    key_inizio = keys;
    H01 = hist(double(I1(1:altezza,1:lunghezza,1)),100);
    H02 = hist(double(I1(1:altezza,1:lunghezza,2)),100);
    H03 = hist(double(I1(1:altezza,1:lunghezza,3)),100);
end

%% Visualizzo costantemente la mia immagine di riferimento

subplot(2,2,1);
imshow(I1);

%% Salvo i frame per poi ricostruire il video.

video(file_ind-1) = getframe(figura1);

%% Aggiornamento dell'immagine.

file_ind = file_ind +1;
[nome_file, err] = sprintf('%s%u.jpg',directory,file_ind);
```

```
    inizio=0;
end

%% Ricostruzione del video.

movie2avi(video,nomevideo,'fps',1,'quality',100,'compression','None');
```

match.m

```
%% Questa funzione legge due immagini, trova le SIFT features e le loro
%% corrispondenze.

function loc = match(image1, image2)

global inizio;
global keys_inizio;
global keys;

global target2;
global keys_inizio0;
global keys0;

%% Trovo i SIFT keypoints per ogni immagine.

[im1, des1, loc1] = sift(image1);
[im2, des2, loc2] = sift(image2);

%% Memorizzo il numero di keypoints che utilizzero nell'aggiornamento
%% dell'immagine di riferimento in caso di "buona corrispondenza".

if inizio==1
    if target2==0
        keys_inizio = size(loc1,1);
    else
        keys_inizio0 = size(loc1,1);
    end
else
    if target2==0
        keys = size(loc1,1);
    else
        keys0 = size(loc1,1);
    end
end

%% Visualizzo la corrente immagine di riferimento con i relativi keypoints.

showkeys(im1, loc1);

%% Per una migliore efficienza in Matlab, e' conveniente calcolare il
%% prodotto interno di vettori normalizzati piuttosto che calcolare le
%% distanze euclidee.
%% Eseguo quindi il prodotto interno tra le matrici relative ai descrittori
%% delle due immagini e, ricavo la distanza dei vettori, grazie
%% all'arcocoseno (le matrici dei descittori sono normalizzate).
%% (Si noti che il rapporto di angoli (acos of dotproducts of unit vectors)
%% e' un'approssimazione vicina al rapporto di distanze euclidee per
%% piccoli angoli).
%% Ordino in maniera crescente le distanze.
%% Un match e' accettatto se la sua distanza e' minore di distRatio volte
%% della seconda distanza(valore fornito da letteratura).

distRatio = 0.6;
des2t = des2';
for i = 1 : size(des1,1)
    dotprods = des1(i,:) * des2t;
    [vals,indx] = sort(acos(dotprods));
    if (vals(1) < distRatio * vals(2))
        match(i) = indx(1);
    else
        match(i) = 0;
    end
end

%% Calcolo il numero di match e restituisco in uscita alla funzione un
%% vettore contenente le coordinate dei punti corrispondenti tra le due
%% immagini.

trovati = find(match > 0);
loc = [loc1(trovati,1:2) loc2(match(trovati),1:2)];
```

sift.m

```
%% This function reads an image and returns its SIFT keypoints.
%% Input parameters:
%%      imageFile: the file name for the image.
%% Returned:
%%      image: the image array in double format
%%      descriptors: a K-by-128 matrix, where each row gives an invariant
%%          descriptor for one of the K keypoints.  The descriptor is
%%          a vector
%%          of 128 values normalized to unit length.
%%      locs: K-by-4 matrix, in which each row has the 4 values for a
%%      keypoint location (row, column, scale, orientation).  The
%%      orientation is in the range [-PI, PI] radians.

%% Credits: Thanks for initial version of this program to D. Alvaro and
%%          J.J. Guerrero, Universidad de Zaragoza (modified by D. Lowe)
```

```
function [image, descriptors, locs] = sift(image)

image = rgb2gray(image);
[rows, cols] = size(image);

%% Convert into PGM imagefile, readable by "keypoints" executable

f = fopen('tmp.pgm', 'w');
if f == -1
    error('Could not create file tmp.pgm.');
end
fprintf(f, 'P5\n%d\n%d\n255\n', cols, rows);
fwrite(f, image', 'uint8');
fclose(f);

%% Call keypoints executable

if isunix
    command = '!./sift ';
else
    command = '!siftWin32 ';
end
command = [command ' <tmp.pgm >tmp.key'];
eval(command);

%% Open tmp.key and check its header

g = fopen('tmp.key', 'r');
if g == -1
    error('Could not open file tmp.key.');
end
[header, count] = fscanf(g, '%d %d', [1 2]);
if count ~= 2
    error('Invalid keypoint file beginning.');
end
num = header(1);
len = header(2);
if len ~= 128
    error('Keypoint descriptor length invalid (should be 128).');
end

%% Creates the two output matrices (use known size for efficiency)

locs = double(zeros(num, 4));
descriptors = double(zeros(num, 128));

%% Parse tmp.key

for i = 1:num
    [vector, count] = fscanf(g, '%f %f %f %f', [1 4]); %row col scale ori
    if count ~= 4
        error('Invalid keypoint file format');
    end
    locs(i, :) = vector(1, :);

    [descrip, count] = fscanf(g, '%d', [1 len]);
    if (count ~= 128)
        error('Invalid keypoint file value.');
    end

    %% Normalize each input vector to unit length

    descrip = descrip / sqrt(sum(descrip.^2));
    descriptors(i, :) = descrip(1, :);
end
fclose(g);
```

rettangolo.m

```
%% Calcola i quattro punti del rettangolo caratterizzato da centro
%% e dimensioni specificate.
%% Poiche il rettangolo viene disegnato, aggiungo un quinto punto uguale
%% al primo per chiudere il disegno.

function ret = rettangolo(cen, alt, lun)

p(1) = cen(1) - alt/2;
p(2) = cen(2) - lun/2;

ret = [ p(1)         p(2);
        p(1)+alt     p(2);
        p(1)+alt     p(2)+lun;
        p(1)         p(2)+lun;
        p(1)         p(2) ];
```

showkeys.m

```
%% This function displays an image with SIFT keypoints overlayed.
%% Input parameters:
%%      image: the file name for the image (grayscale)
%%      locs: matrix in which each row gives a keypoint location (row,
%%          column, scale, orientation)

%% Credits: Thanks for initial version of this program to D. Alvaro and
%%          J.J. Guerrero, Universidad de Zaragoza (modified by D. Lowe)

function showkeys(image, locs)
```

```
disp('Drawing SIFT keypoints ...');

%% Draw image with keypoints

subplot(2,2,3);
colormap('gray');
imshow(image)

hold on;
imsize = size(image);
for i = 1: size(locs,1)

    %% Draw an arrow, each line transformed according to keypoint parameters.

    TransformLine(imsize, locs(i,:), 0.0, 0.0, 1.0, 0.0);
    TransformLine(imsize, locs(i,:), 0.85, 0.1, 1.0, 0.0);
    TransformLine(imsize, locs(i,:), 0.85, -0.1, 1.0, 0.0);
end
hold off;

%% ------ Subroutine: TransformLine -------
%% Draw the given line in the image, but first translate, rotate, and
%% scale according to the keypoint parameters.
%% Parameters:
%%      Arrays:
%%          imsize = [rows columns] of image
%%          keypoint = [subpixel_row subpixel_column scale orientation]
%%
%%      Scalars:
%%          x1, y1; begining of vector
%%          x2, y2; ending of vector

function TransformLine(imsize, keypoint, x1, y1, x2, y2)

%% The scaling of the unit length arrow is set to approximately the radius
%% of the region used to compute the keypoint descriptor.

len = 6 * keypoint(3);

%% Rotate the keypoints by 'ori' = keypoint(4)

s = sin(keypoint(4));
c = cos(keypoint(4));

%% Apply transform

r1 = keypoint(1) - len * (c * y1 + s * x1);
c1 = keypoint(2) + len * (- s * y1 + c * x1);
r2 = keypoint(1) - len * (c * y2 + s * x2);
c2 = keypoint(2) + len * (- s * y2 + c * x2);

line([c1 c2], [r1 r2], 'Color', 'c');
```

avviamulti.m

```
%% Accede alle immagini contenute in 'directory' ed inizia il tracciamento.
%% Le immagini devono essere in formato jpg e con nomi '1.jpg', '2.jpg'...

function [] = avviamulti(directory,nomevideo)

global num_particelle;
global confrontovarianze;

global inizio;
global keys_inizio;
global keys;

global target2;
global keys_inizio0;
global keys0;

%% Controllo dei nomi dei file immagine.

[nome_file, err] = sprintf('%s1.jpg',directory);
if(~exist(nome_file,'file'))
    error('Immagine iniziale non trovata');
    return;
end

[nome_file2, err] = sprintf('%s2.jpg',directory);
if(~exist(nome_file2,'file'))
    error('Immagine iniziale non trovata');
    return;
end

[nome_file0, err] = sprintf('%s1bis.jpg',directory);
if(~exist(nome_file0,'file'))
    error('Immagine iniziale non trovata');
    return;
end

%% Memorizzo parametri relativi all'immagine di riferimento.

I1 = imread(nome_file);
IRif = I1;
dim = size(I1);
altezza = dim(1)
```

```
lunghezza = dim(2)
centro = [altezza/2 lunghezza/2];

I10 = imread(nome_file0);
IRif0 = I10;
dim0 = size(I10);
altezza0 = dim0(1)
lunghezza0 = dim0(2)
centro0 = [altezza0/2 lunghezza0/2];

%% Ritaglio l'oggetto per farne l'istogramma che sara' usato
%% successivamente per valutare la correttezza del rettangolo proposti da
%% ogni particella.

H01 = hist(double(I1(1:altezza,1:lunghezza,1)),100);
H02 = hist(double(I1(1:altezza,1:lunghezza,2)),100);
H03 = hist(double(I1(1:altezza,1:lunghezza,3)),100);

H010 = hist(double(I10(1:altezza0,1:lunghezza0,1)),100);
H020 = hist(double(I10(1:altezza0,1:lunghezza0,2)),100);
H030 = hist(double(I10(1:altezza0,1:lunghezza0,3)),100);

%% Inizializzazione dell'immagine.

I2 = imread(nome_file2);

scrsz = get(0,'ScreenSize');
figura1 = figure('Position',[1 1 scrsz(3) scrsz(4)]);

subplot(2,2,[2 4]);
imshow(I2);

subplot(2,2,1);
imshow(I1);

subplot(2,2,3);
[im1, des1, loc1] = sift(I1);
showkeys(im1, loc1);

%% Inizializzazione di pesi e particelle.
%% Le particelle vengono inizializzate con la posizione centrale di ogni
%% immagine ed ogni peso è posto ad 1.

particelle = repmat(centro,num_particelle,1);
pesi = repmat(1,num_particelle,1);

particelle0 = repmat(centro0,num_particelle,1);
pesi0 = repmat(1,num_particelle,1);

%% Inzio l'algoritmo di confronto: dapprima con le due immagini iniziali e
%% poi quelle successive.

file_ind = 2;
[nome_file, err] = sprintf('%s2.jpg',directory);

%% Inizializzo vatiabili che tengono riferimento rispettivamente di:
%% - Prima immagine
%% - Target in utilizzo
%% - Numero di match tra due immagini

inizio=1;
target2=0;

num_match=0;

num_match0=0;

while exist(nome_file,'file')

    %% Confronto le due immagine correnti

    I2 = imread(nome_file);

    %% Calcolo dei punti corrispondenti con il sift.
    %% Ogni riga di L contiene: ay ax by bx
    %% dove a e' un punto in I1 e b il corrispondente in I2.

    L = match(I1,I2);
    num_match = size(L,1);

    %% Visualizzo l'immagine (nel multiraget lo faccio qui per una
    %% questione di coordinazione visiva con l'altra immagine).

    subplot(2,2,1);
    imshow(I1);

    %% Salvo i frame per poi ricostruire il video.

    video(2*file_ind - 3) = getframe(figura1);

    target2 = 1;
    L0 = match(I10,I2);
    num_match0 = size(L0,1);
    target2 = 0;

    subplot(2,2,1);
    imshow(I10);
```

```
%% Spostamento stimato in base ai punti calcolati dal sift: se non ci
%% sono corrispondenze i quadrati devo rimanere fermi. Inoltre lo
%% spostamento viene calcolato rispetto alla posizione precedente, con
%% conseguente differenziazione nel caso si sia nel primo matching.

if (num_match==0)
    dx = 0;
    dy = 0;
    dx_medio = 0;
    dy_medio = 0;
else
    if (inizio==1)
        dx = L(:,4) - L(:,2);
        dy = L(:,3) - L(:,1);
    else
        dx = L(:,4) - L(:,2) - (nuovo_centro(2) - lunghezza/2 + 1);
        dy = L(:,3) - L(:,1) - (nuovo_centro(1) - altezza/2 +1);
    end
    dx_medio = sum(dx)/num_match;
    dy_medio = sum(dy)/num_match;
end

if (num_match0==0)
    dx0 = 0;
    dy0 = 0;
    dx_medio0 = 0;
    dy_medio0 = 0;
else
    if (inizio==1)
        dx0 = L0(:,4) - L0(:,2);
        dy0 = L0(:,3) - L0(:,1);
    else
        dx0 = L0(:,4) - L0(:,2) - (nuovo_centro0(2) - lunghezza0/2 + 1);
        dy0 = L0(:,3) - L0(:,1) - (nuovo_centro0(1) - altezza0/2 +1);
    end
    dx_medio0 = sum(dx0)/num_match0;
    dy_medio0 = sum(dy0)/num_match0;
end

%% Lo spostamento medio viene aggiunto ad ogni particella per
%% l'estrazione della prossima posizione

mi = particelle + repmat([dy_medio dx_medio],size(particelle,1),1);

mi0 = particelle0 + repmat([dy_medio0 dx_medio0],size(particelle0,1),1);

%% Calcolo la varianza in base agli spostamenti.

sigma_stimato_x = cov(dx);
sigma_stimato_y = cov(dy);

sigma_stimato_x0 = cov(dx0)
sigma_stimato_y0 = cov(dy0)

%% Confronto due metodi per calcolare la varianza: il primo ottenuto da
%% comandi matlab mentre il secondo ricavato da letteratura.

%% Metodo 1

sigma_x = sigma_stimato_x;
sigma_y = sigma_stimato_y;

sigma_x0 = sigma_stimato_x0;
sigma_y0 = sigma_stimato_y0;

%% Metodo 2

if confrontovarianze==1
    alpha = 1;
    beta = 1;
    sigma_x = abs(alpha * log((1-exp(sigma_stimato_x))^-1) + beta * (exp(-num_match)));
    sigma_y = abs(alpha * log((1-exp(sigma_stimato_y))^-1) + beta * (exp(-num_match)));

    alpha0 = 1;
    beta0 = 1;
    sigma_x0 = abs(alpha0 * log((1-exp(sigma_stimato_x0))^-1) + beta0 * (exp(-num_match0)));
    sigma_y0 = abs(alpha0 * log((1-exp(sigma_stimato_y0))^-1) + beta0 * (exp(-num_match0)));
end

%% Aggiorno le particelle utilizzando una distribuzione gaussiana di
%% media e varianza precedentemente determinate.

particelle(:,1) = normrnd(mi(:,1),repmat(sigma_y,num_particelle,1));
particelle(:,2) = normrnd(mi(:,2),repmat(sigma_x,num_particelle,1));

particelle0(:,1) = normrnd(mi0(:,1),repmat(sigma_y0,num_particelle,1));
particelle0(:,2) = normrnd(mi0(:,2),repmat(sigma_x0,num_particelle,1));

p_y = uint32(particelle(:,1));
p_x = uint32(particelle(:,2));

p_y0 = uint32(particelle0(:,1));
p_x0 = uint32(particelle0(:,2));

for i=1:num_particelle

    %% Per ogni particella individuiamo il relativo rettangolo e
    %% confrontiamo il suo istogramma con quello iniziale o di
```

```
    %% riferimento.

    %% Puo' talvolta succedere che alcune particelle siano fuori dallo
    %% spazio di lavoro o che il comando hist si trovi ad operare su
    %% matrici di ampiezza diversa(molto raramente);
    %% predispongo adeguati controlli mettendo i relativi pesi a zero.

    if ((p_x(i)-lunghezza/2+1) <= 0) || ((p_x(i)+lunghezza/2) > size(I2,2)) || ((p_y(i)-altezza/2+1) <= 0) || ((p_y(i)+altezza/2) > size(I2,1))
        pesi(i)=0;
    else
        Hi1 = hist(double(I2((p_y(i)-altezza/2+1):(p_y(i)+altezza/2), (p_x(i)-lunghezza/2+1):(p_x(i)+lunghezza/2),1)),100);
        Hi2 = hist(double(I2((p_y(i)-altezza/2+1):(p_y(i)+altezza/2), (p_x(i)-lunghezza/2+1):(p_x(i)+lunghezza/2),2)),100);
        Hi3 = hist(double(I2((p_y(i)-altezza/2+1):(p_y(i)+altezza/2), (p_x(i)-lunghezza/2+1):(p_x(i)+lunghezza/2),3)),100);
        if (size(H01)==size(Hi1)) & (size(H02)==size(Hi2)) & (size(H03)==size(Hi3))

            %% La distanza euclidea si e' rivelata migliore di quella di
            %% Bhattacharyya nei nostri casi.

            distH = sqrt(sum(sum((Hi1 - H01).^2)) + sum(sum((Hi2 - H02).^2)) + sum(sum((Hi3 - H03).^2)));

            %% Come indicato dalla letteratura usiamo usiamo una distribuzione
            %% esponenziale per l'aggiornamento dei pesi.

            lambda = 0.001;
            prob = exp(-lambda * distH);
            pesi(i) = prob * pesi(i);
        else
            pesi(i) = 0;
        end
    end

    if ((p_x0(i)-lunghezza0/2+1) <= 0) || ((p_x0(i)+lunghezza0/2) > size(I2,2)) || ((p_y0(i)-altezza0/2+1) <= 0) || ((p_y0(i)+altezza0/2) > size(I2,1
        pesi0(i)=0;
    else
        Hi10 = hist(double(I2((p_y0(i)-altezza0/2+1):(p_y0(i)+altezza0/2), (p_x0(i)-lunghezza0/2+1):(p_x0(i)+lunghezza0/2),1)),100);
        Hi20 = hist(double(I2((p_y0(i)-altezza0/2+1):(p_y0(i)+altezza0/2), (p_x0(i)-lunghezza0/2+1):(p_x0(i)+lunghezza0/2),2)),100);
        Hi30 = hist(double(I2((p_y0(i)-altezza0/2+1):(p_y0(i)+altezza0/2), (p_x0(i)-lunghezza0/2+1):(p_x0(i)+lunghezza0/2),3)),100);
        if (size(H010)==size(Hi10)) & (size(H020)==size(Hi20)) & (size(H030)==size(Hi30))
            distH0 = sqrt(sum(sum((Hi10 - H010).^2)) + sum(sum((Hi20 - H020).^2)) + sum(sum((Hi30 - H030).^2)));
            lambda0 = 0.001;
            prob0 = exp(-lambda0 * distH0);
            pesi0(i) = prob0 * pesi0(i);
        else
            pesi0(i) = 0;
        end
    end
end

%% Normalizzazione dei pesi

somma = sum(pesi);
if somma > 0
    pesi = pesi./somma;
end

somma0 = sum(pesi0);
if somma0 > 0
    pesi0 = pesi0./somma0;
end

%% Soppressioni dei peggiori e duplicazione del migliore

[pMax PMInd] = max(pesi);
Soppressi = find(pesi < 0.8*pMax);
pesi(Soppressi) = pesi(PMInd);
particelle(Soppressi,1) = particelle(PMInd,1);
particelle(Soppressi,2) = particelle(PMInd,2);

[pMax0 PMInd0] = max(pesi0);
Soppressi0 = find(pesi0 < 0.8*pMax0);
pesi0(Soppressi0) = pesi0(PMInd0);
particelle0(Soppressi0,1) = particelle0(PMInd0,1);
particelle0(Soppressi0,2) = particelle0(PMInd0,2);

%% Nuova normalizzazione (probabilmente i pesi sono cambiati)

somma = sum(pesi);
if somma > 0
    pesi = pesi./somma;
end

somma0 = sum(pesi0);
if somma0 > 0
    pesi0 = pesi0./somma0;
end

%% Calcolo il centro della nuova immagine identificata

nuovo_centro = mean(particelle);

nuovo_centro0 = mean(particelle0);

%% Disegno l'immagine corrente

subplot(2,2,[2 4]);
hold off
imshow(I2);
hold on
```

```
    %% Disegno il centro stimato,le particelle, un quadrato con dimensione
    %% fissa pari a quella dell'immagine e uno con dimensione variabile,
    %% dipendente dalla varianza.

    plot(nuovo_centro(2),nuovo_centro(1),'*b');
    plot(particelle(:,2),particelle(:,1),'*m');
    rett = rettangolo(nuovo_centro, altezza, lunghezza);
    plot(rett(:,2),rett(:,1),'y');
    rett = rettangolo(nuovo_centro, altezza + 2*sigma_y, lunghezza + 2*sigma_x);
    plot(rett(:,2),rett(:,1),'r');

    plot(nuovo_centro0(2),nuovo_centro0(1),'*b');
    plot(particelle0(:,2),particelle0(:,1),'*g');
    rett0 = rettangolo(nuovo_centro0, altezza0, lunghezza0);
    plot(rett0(:,2),rett0(:,1),'y');
    rett0 = rettangolo(nuovo_centro0, altezza0 + 2*sigma_y0, lunghezza0 + 2*sigma_x0);
    plot(rett0(:,2),rett0(:,1),'r');

    drawnow;

    %% Se l'immagine corrente ha un'elevata corrispondenza con la
    %% precedente, aggiorno la mia immagine di riferimento, il numero di
    %% punti chiave e gli istogrammi.

    if num_match > keys_inizio/6
        I1 = I2((nuovo_centro(1)-altezza/2+1):(nuovo_centro(1)+altezza/2),(nuovo_centro(2)-lunghezza/2+1):(nuovo_centro(2)+lunghezza/2),:);
        key_inizio = keys;
        H01 = hist(double(I1(1:altezza,1:lunghezza,1)),100);
        H02 = hist(double(I1(1:altezza,1:lunghezza,2)),100);
        H03 = hist(double(I1(1:altezza,1:lunghezza,3)),100);
    end

    if num_match0 > keys_inizio0/2
        I10 = I2((nuovo_centro0(1)-altezza0/2+1):(nuovo_centro0(1)+altezza0/2),(nuovo_centro0(2)-lunghezza0/2+1):(nuovo_centro0(2)+lunghezza0/2),:);
        key_inizio0 = keys0;
        H010 = hist(double(I10(1:altezza0,1:lunghezza0,1)),100);
        H020 = hist(double(I10(1:altezza0,1:lunghezza0,2)),100);
        H030 = hist(double(I10(1:altezza0,1:lunghezza0,3)),100);
    end

    %% Salvo i frame per poi ricostruire il video.

    video(2*file_ind - 2) = getframe(figura1);

    %% Aggiornamento dell'immagine.

    file_ind = file_ind +1;
    [nome_file, err] = sprintf('%s%u.jpg',directory,file_ind);
    inizio=0;
end

%% Ricostruzione del video.

movie2avi(video,nomevideo,'fps',1,'quality',100,'compression','None');
```

VII. IL NAVLAB

Nel laborotario NavLab è presente il sistema di Motion Capture Smart. **Smart** è un sistema ottico (prodotto e distribuito da *E-motion*), di rilevamento dei movimenti, che consiste in:

- 1 Pc con installato il software;
- 6 telecamere a rivelazione di luce infrarossa;
- 2 unità di alimentazione e sincronizzazione.

Il pacchetto **Smart** è uno strumento potente, flessibile e completo, è adatto alle esigenze dei ricercatori per condurre analisi multifattoriali del movimento nell'ambito della biomeccanica, della robotica, delle scienze motorie e della computer graphics. Gli algoritmi per l'elaborazione delle immagini assicurano una buona accuratezza, un aumento del campo di ripresa di ciascuna telecamera e del volume complessivo calibrabile. Il sistema **Smart** rileva e registra le posizioni nello spazio assunte da piccoli markers nel tempo. I markers vengono applicati all'oggetto che si vuole analizzare. Le posizioni dei markers, gli angoli tra differenti segmenti eventualmente definiti e le loro velocità ed accellerazioni sono automaticamente misurate ad ogni istante, dando una misura quantitativa del movimento.
Il sistema **Smart** inoltre offre la possibilità di esportare i dati (2D,3D) rilevati, per altre elaborazioni. Si rende così possibile la rivelazione di un movimento nello spazio tramite le telecamere, il calcolo della posizione relativa dei markers (al fine di trovare la posizione e l'orientamento di un sistema di riferimento solidale al corpo in movimento) e la possibilità di poter usufruire di tali dati in tempo reale.

A. l'Hardware

Il sistema Smart è costituito da tre componenti principali: il Pc le telecamere (VII-A) e le unità di sincronizzazione.

1) le telecamere sono normali telecamere CCD con filtri infrarossi. Ogni telecamera, può ospitare, qualsiasi lente compatibile C-mount da grandagolare a teleobbiettivo, per dare una maggiore flessibilità nella definizione dello spazio operativo per il rilevamento. Ogni telecamera ha un emettitore coassiale stroboscopico di luce infrarossa, invisibile all'occhio umano, digitalmente sincronizzata con le telecamere. Gli impulsi luminosi emessi sono usati per evidenziare i markers piazzati sull'oggetto; mentre la luce ambientale non influenza le performance del sistema.
2) gli alimentatori, sincronizzano e raccolgono i segnali provenienti dalle telecamere e le interfacciano al Pc;
3) la stazione di lavoro è un Pc con processore Pentium 4. Esso è fornito di un numero di schede per il rilevamento veloce di immagini, che acquistano dalle 50 alle 120 immagini per secondo. Le immagini vengono elaborate in tempo reale dal Pc avantaggiandosi delle capacità dei processori moderni nell'elaborare dati multimediali.

Fig. 8. Sistema SMART

B. Smart

Il sistema **Smart** è un sistema di motion capture basato,come sopra accennato, su una tecnologia ottica. **Smart** può eseguire la ricostruzione in tre dimensioni di un certo numero di piccoli markers riflettenti. Partendo da un set di telecamere posizionate tutt'attorno all'area dove si svolgerà l'azione, il sistema acquisisce il movimento dei markers; la posizione delle telecamere sarà definita grazie alla relativa calibrazione. La calibrazione del sistema è un operazione facile e veloce, ma allo stesso tempo svolge un ruole molto importante. Per quanto riguarda l'installazione e il posizionamento delle telecamere è stato già effettuato nel laboratorio NAVLAB. Dove è stata efettuata la connessione dei Video Hubs al Pc e l'installazione e la connessione delle telecamere ai rispettivi Video Hubs.

1) L'utilizzo di Smart: IL software Smart è già presente nel Pc e così, pronto all'uso. Il software consiste di due differenti pacchetti: SMARTcapture e SMARTtracker. Il primo dei due si occupa delle procedure di acquisizione e della calibrazione del sistema, mentre il secondo esegue la ricostruzione 3D delle traiettorie e permette la visualizzazione del risultato. Va fatta un'importante osservazione: le telecamere, nel laboratorio di NavLab, sono state posizionate ad altezze diverse affinché durante la calibrazione sia possibile ricostruire al meglio la geometria epipolare. L'operazione di calibrazione andrà ripetuta solo nel caso in cui ci sia una modifica sostanziale dell'area di lavoro. Nel predisporre la calibrazione del sistema, in primo luogo è necessario

decidere le dimensioni e la posizione dell'area di lavoro dove l'azione si svolgerà. L'area in questione deve essere priva di qualsiasi oggetto che potrebbe nascondere la visualizzazione della telecamere, come materiali con caratteristiche simili a quelle dei markers e libera di sorgenti ad infrarossi. Per la calibrazione è utile delimitare l'area di lavoro, dove si andrà a calibrare, con quattro markers che temporaneamente sono stati posati ai quattro angoli. Si deve direzionare le telecamere verso l'area che si vuole calibrare. Si attiva il software **SMARTcapture** . Dalla barra degli stumenti si crea un nuovo file *newCalibration*, nel quale verranno memorizzati i parametri. In ordine si acquisisce un *Axes sequence* e un *Wand sequence*, prima di eseguire il *Calibration Wizard*. Nel pannello **File**, nel lato sinistro della finestra principale, appare l'icona seguita dal nome del file che memorizzerà i dati della calibrazione. Il file contiene una cartella della calibrazione , contenente un blocco inizialmente vuoto **Axes** e un blocco anch'esso vuoto **Wand**.

Per l'acquisizione di un **Axes sequence** si posiziona al centro dell'area di calibrazione la bacchetta con i relativi markers e si seleziona il blocco **Axes** cliccando sull'icona . Si passa al **Monitor** mode cliccando sul pulsante **Monitor** ; l'acquisizione inizia cliccando su **Capture** , per fermare si clicca nuovamente su **Capture**. Si consiglia di acquisire qualche centinaio di frame, il che equivale a lasciare in esecuzione il **Capture** per alcuni secondi.

Una simile procedura si svolge per acquisire una **Wand sequence**: cliccando sull'icona si seleziona il blocco **Wand**, nuovamente cliccando sul pulsante **Capture** parte l'acquisizione, a questo punto si deve muovere la bacchetta,con un ampio gesto, su e giù al centro dell'area, per circa 90 secondi. Assicurandosi che tutte le telecamere inquadrino la bacchetta per un ragionabile numero di frames.

Ora si lancia la **Calibration Wizard**, cliccando sul pulsante **Calibration** , nel Main Toolbar, si clicca Start e si attende. L'algoritmo di calibrazione impiega alcuni minuti per calcolare i parametri. Per terminare la calibrazione si clicca semplicemnte su **Finish**. Così la procedura di calibrazione è completa. La calibrazione si dovrebbe rifare, come già detto, solo nel caso in cui cambia il set-up fisico delle telecamere; tuttavia, il tempo che si impiega per calibrare il sistema è molto poco, perciò si raccomanda di controllare il sistema all'inizio di ogni nuova sessione di lavoro e ogni volta che le configurazioni del sistema si pensa siano state alterate.

Si passa ora ad illustrare l'acquisizione di una sequenza di immagini, in particolare una sequenza 2D.

Dopo aver lanciato l'applicazione *SMARTcapture* , si seleziona sempre dal Main Toolbar il pulsante **New** per creare un nuovo progetto.

Nel pannello File, in alto a sinistra nella finestra principale, l'icona appare seguita dal nome del file in cui verrò memorizzata la sequenza 2D acquisita. Il file al momento della creazione contiene una sequenza 2D vuota rappresentata dal blocco . La procedura per l'acquisizione è del tutto analoga a quella effettuata per la calibrazione. Una volta completata la procedura si clicca su **Save** per la memorizzazione dei dati. Solo dopo aver eseguito l'acuisizione è possibile visualizzare la sequenza di dati, e trasportare la sequenza 2D in una sequenza 3D. Quest'ultima operazione si esegue cliccando sul pulsante **Go 3D** , automaticamente viene eseguita l'applicazione **SMARTtracker** in riferimento alla sequenza 2D in esame. Una sequenza 3D ricostruita può essere visualizzata da qualsiasi punto di vista. Tutti i dati creati e gestiti dal sistema **SMART**, di cui finora si è parlato, possono essere memorizzati in un file con estensione Tdf.

C. SMARTtracker

SMARTtracker è un ambiente di grafica interattiva per la rappresentazione in tre dimensione e la ricostruzione del movimento di piu' target.

SMARTtracker acquisisce come input un sequenza di dati in due dimensioni creata con SMARTcapture (contenenti i dati presi dall'acqusizione delle sei telecamere) e automaticamente genera un set di traiettorie in tre dimensioni, grazie ad un potente motore di tracking conosciuto come ICY2K.

Si possono etichettare le traiettorie e associare ad esse un modello di connettività. Quest'ultimo spesso è un insieme di collegamenti che si possono associare alle traiettorie. Inoltre è possibile gestire i dati acquisiti dalle piattaforme che sono compatibili con il sistema SMART.

La finestra principale del SMARTtracker è divisa in cinque aree principali: la barra dei Titoli, la barra del Menu' Principale, il Main Toolbar, la Client Area e la barra di stato.

La Viewer Window appare ogni volta che si apre una sequenza (o Tdf file). Essa è composta da un Toolbar e due sezioni principali: un 3D Viewer e un Graph Viewer. La sezione 3D Viewer da la possibilità di navigare all'interno dell'aread'interesse con una telecamera virtuale per guardare le sequenza dei movimenti che sono state catturate. Una o piu' finestre si possono aggiungere allo

scopo di vedere la stessa scena da differenti punti di vista contemporaneamente.
La sezione Graph Viewer mostra il percorso della posizione dei tracks, la loro velocità e accelarazione in riferimento al tempo.

1) Il motore del tracking: ICY2K: ICY2K associa i dati in due dimensioni, acquisiti dalle diverse telecamere, creando in output un insieme definito **Tracks**. Le **Tracks** sono una sequenza temporale di punti in tre dimensioni, che sono stati riconosciuti come la successione di uno marker. Un numero progressivo definito come Track Id è associato a ciascun track.
Un track può contenere molti intervalli e così può essere composto da più di un segmento: dove per segmento si intende una sequenza di punti 3D consecutivi appartenti alla stessa Track. La **Traccia** di un Track è l'insieme di punti che sono stati lasciati dalla stessa Track. I punti si possono visualizzare nella finestra Viewer come una traiettoria nello spazio.

D. SMART Motion Capture System Real Time SDK

Questo è pacchetto contenente alcune applicazioni real-time: permette di creare e modificare una sequenza di dati, in particolare una sequenza di punti 2D e 3D. Le applicazioni hanno bisogno di avere i dati del processo in real time. Esse implementano due concetti pricipali: i dataport e i thread. Il primo è una sorta di processo indipendente, la cui esecuzione è concorrente ad altre applicazioni. Il secondo è invece ha la possibilità, occupando una parte della memoria accessibile per alcune applicazione, la modificazione dei dati. Ciascun dataport è identificato con un unico nome e non possono essere presenti più dataport con lo stesso nome. Un dataport può essere descritto come un data flow tra un server, o source, e uno o più clienti, o target.
Si possono creare dataport in modi diversi. Un esempio di generazione di dataport è *IcyRT*. Questa applicazione agisce sia come un client di un dataport source che come un dataport server. *RTDump* e *Viewer3D* sono invece solamente dataport clients, perchè essi leggono il flusso di dati dal dataport e contemporaneamente permettono la visualizzazione. Per creare un dataport 2D con **SMARTcapture** e poter visualizzare i relativi data flow si deve:

- aprire **SMARTcapture** e in modo opportuno si fa il set up delle configurazioni;
- selezionando *Settings* in Data 2D si modifca l'uscita da *DataBuffer* a *Data Port*;
- si clicca Ok e si chiude la finestra di *Settings*;
- si clicca *Monitor* e poi *Capture* per iniziare l'acquisizione.

In questo modo un dataport 2D è attivo e in real time avviene l'acquisizione di un flusso di dati.

- si apre il prompt dei comando;
- si giunge fino alla cartella di SMART (tipicamente è: C:/Program Files/eMotion/SMART/1.10);
- si entra in RTDump /src:'Data2D Raw' /d2d.

Quando RTDump viene eseguito, visualizza il flusso di dati del dataport 'Data2D Raw', corrispondente all'uscita del dataport di SMARTcapture. A display viene emesso il flusso di punti che in real time si stanno acquisendo dalle telecamere abilitate, in riferimento ai markers presenti nell'area di lavoro.
Per la creazione di un dataport 3D con SMARTcapture il procedimento è molto simile a quello precedentemente illustrato. Si deve modificare l'argomento di RTDump in questo modo: RTDump /src:'Data3D Raw' /d3d. Il software presente in laboratorio, al momento, non permette la creazione di un dataport 3D. C'è comunque la possibilità di implementare questa procedura, l'eMotion da la possibilità di accedere ai file sorgente (in C++) di tutti i pacchetti. All'interno di ciascun file è indicato dove inserire il codice implementato per la generazione in real time di un dataport 3D. In ogni caso sono già presenti alcune opzioni a riguardo:

- **IcyRT.exe** è una linea di comando per la ricostruzione in 3D. Legge i dati dal dataport attivo e, usando la calibrazione del file specificato, calcola i punti 3D e li memorizza in un altro dataport, da lui creato. L'istruzione è: IcyRT/cal:<calib filename>[/in:<dataport name>][/out:<dataport name>][/calpath:<calib path>]. Si fa notare che i file TDF contengono le informazioni della calibrazione rispetto ai dati 2D del dataport in input. Il nome di un dataport identifica univocamente un data stream, se non viene specificato un nome per il dataport in output, il processo di ricostruzione non ha la possibilità di eseguire;
- **RTDump.exe** permette di visualizzare il flusso di dati di un specifico dataport. Usando come già detto l'istruzione RTDump /src:<dataport name>/d2d, /d3d Viene richiesto che un data stream 2D o 3D sia aperto. Per bloccare l'esecuzione si usa il comando CTRL+C;
- Viewer3D.exe: questa applicazione visualizza la ricostruzione dei punti in uno spazio 3D virtuale. Il punto di vista e molti altri parametri della scena possono essere cambiati semplicemente dal Toolbar presente nella finestra principale.
  Il pacchetto include alcuni esempi per la comprensione del software e l'eventuale sviluppo di nuove applicazioni con RTSDK. Due semplici applicazioni

sono date da: RTRead e TRWrite. Il primo permette il collegamento e la lettura di un dataport, il secondo mostra una tecnica per inizializzare e creare un dataport e la possibilità di sovrascrivere. Entrambi gli esempi sono sviluppati in ambiente C++.

- **RTRead**: crea un semplice dataport client usando Datapot.lib, quest'ultimo è già incluso nel pacchetto. La libreria è provvista di due classi, CRealTimeDataPortClient Data2D e Data3D;
- **RTWrite** è il complementare del precedente, visualizza le classi del dataport server, implementate in Dataport.lib, con le relative classi CRealTimeDataPortServer Data2D e Data3D.

CONTENTS

REFERENCES

[1] Y. Bar-Shalom and T. Fortmann, Tracking and Data Association.
[2] C. Rasmussen and G.D. Hager, Probabilistic data association methods for tracking complex visual objects.
[3] S. Oh, S. Russell, and S. Sastry, Markov chain Monte Carlo data association for general multiple-target tracking problems.
[4] S. Särkkä , A. Vehtari and J. Lampinen, Rao-Blackwellized Monte Carlo Data Association for Multiple Target Tracking.
[5] S. Oh and S. Sastry, An Efficient Algorithm for Tracking Multiple Maneuvering Targets.
[6] R. Frezza and A. Chiuso, Learning and exploiting invariants for multi-target tracking and data association.
[7] Y. Bar-Shalom, T. Kirubarajan and C. Gokberk, Tracking with Classification-Aided Multiframe Data Association.
[8] S. Oh, L. Schenato, P. Chen, and S. Sastry, Tracking and coordination of multiple agents using sensor networks: system design, algorithms and experiments.
[9] H.A.P. Blom and E.A. Bloem, Probabilistic Data Association Avoiding Track Coalescence.
[10] J. Cai, A. Sinha and T. Kirubarajan, EM-ML Algorithm for Track Initialization using Possibly Noninformative Data.
[11] A. Doucet, On sequencial simulation-based methods for bayesian filtering.
[12] A. Kong, J.S. Liu e W.H. Wong, Sequential imputations and bayesian missing data problems.
[13] D. G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints.
[14] J. M. Reynolds, Generic object tracking Motion without models.
[15] Y. Ma, S. Soatto, J. Koseckà, S. Shankar Sastry, An Invitation to 3-D Vision.