UNIVERSITÀ DI PADOVA  FACOLTÀ DI INGEGNERIA

Omnidirectional Vision Range Finder per Localizzazione di Monte Carlo

Relatore: Chiar.mo Prof. *Enrico Pagello*

Correlatore: Dr. *Emanuele Menegatti*

Laureando: *Alberto Scarpa*

Corso di Laurea in Ingegneria dell'Informazione

Dipartimento di Ingegneria dell'Informazione

Anno Accademico 2003/2004

Indice

Sommario

In questa tesina viene affrontato il problema della localizzazione di un robot mobile sia all'interno di un ambiente fortemente dinamico come il campo RoboCup sia in un ambiente generico come può essere un corridoio. L'approccio utilizzato si basa sull'utilizzo di un sensore range finder capace di individuare transizioni cromatiche (OVRF) filtrato attraverso algoritmi di Monte Carlo.

Nel caso della localizzazione in ambienti dinamici, in cui risulta molto importante la reattività del sistema, è necessario che gli algoritmi di localizzazione funzionino in real-time, mentre in ambienti generici è molto impotante la robustezza alle variazioni luminose. In questa tesina vengono presi in considerazioni entrambi i problemi presentando le soluzioni adottate per il primo e proponendo alcune soluzioni per il secondo.

Infine vengono presentati alcuni risultati sperimentali per dimostrare l'efficienza della strategia di localizzazione adottata in un ambiente generico.

Capitolo 1

Introduzione

Con questo lavoro si cerca di dare una soluzione al problema della localizzazione di un robot mobile in un ambiente strutturato generico. Il robot dovrà essere capace di ricavare in modo autonomo la sua posizione e la sua orientazione all'interno di un'ambiete avendo come dati di base una mappa metrica con le occupazioni degli oggetti presenti nell'ambiente (ostacoli fissi), quali muri, armadi, scrivanie e una mappa metrica con i colori di tali ogetti. L'idea di base dell'algoritmo consiste nell'individuare all'interno di un'immagine presa dal robot le trasizioni cromatiche dell'ambiente e di cercare nelle mappe la posizione dalla quale si ha la maggior probabilità di vedere tali transizioni.

Avere una localizzazione affidabile è indispensabile per potersi muovere all'interno di un ambiente e per poter svolgere qualsiasi mansione che comporti una spostamento all'inerno di un'ambiente. Per esempio si pensi ad un robot da ufficio con il compito di portare documenti da una scrivania all'altra. Per poter svolgere tale mansione è indispensabile che il robot conosca le coordinate assolute delle scrivanie in relazione al sistema di riferimento dell'ambiente, conoscenza che deve essere fornita dall'esterno, e che conosca la propria ubicazione all'interno dell'uficio, ricavabile dalle immagini prese dal robot.

Altrettanto importante è la localizzazione in ambiente RoboCup.
RoboCup è una competizione tra robot autonomi in cui squadre di agenti (sia reali che simulati) si sfidano in un campo di calcio. In questi anni le competizioni RoboCup hanno stimolato molte delle ricerche svolte sulla robotica mobile e sulla coordinazione multi-agente. In questo lavoro ci riferiremo sempre alla competizione di robot reali categoria "F-2000" (middle-size) in cui i robot hanno dimensioni massime che permettano di essere contenuti un parallelepipedo con base quadrata di 50 cm e alti al più 80 cm. Le piattaforme, utilizzate per le sperimentazioni svolte in questo lavoro, sono dei robot per

Robocup del team Artisti Veneti [11] dell'Università di Padova.

Nelle competizione RoboCup avere una localizzazione affidabile è indispensabile per poter giocare rispettando le regole. Infatti ci deve essere un robot portiere che resta sempre vicino alla propria porta. Inoltre conoscere la propria posizione serve anche per evitare di uscire dal campo di gioco e per poter fare delle azioni cooperative sofisticate come il passagio di palla.

Una possibile tecnica di localizzazione consiste nell'utilizzo degli encoder montati sulle ruote dei robot. Questi sensori permettono di contare i giri delle ruote e di avere quindi una stima dello spostamento del robot. La posizione assoluta si può poi facilmente calcolare sommando alla posizione di partenza assegnata lo spostamento. Gli encoder sono sensori molto precisi ma dipendono dalla perfetta aderenza delle ruote del robot al suolo. Alcuni problemi degli encoder sono gli slittamenti delle ruote dovuti ad un cambio di superficie del suolo, ad mancato contatto di una ruota con il terreno o ad una spinta avvenuta a ruote bloccate, gli urti e la imprecisione insita dello strumento. Questi fattori fanno si che l'errore odometrico risulti dipendente dal tempo e che quindi la localizzazione venga persa dopo pochi minuti.

Anche i sensori ambientali, come visori frontali e omnidirezionali, sonar e laser, possono essere usati per la localizzazione. Essi forniscono delle misure dell'ambiente circostante che possono essere confrontate con quelle attese dal modello dell'ambiente per stmare la posizione del robot. Le misurazioni di questi strumenti risultano molto rumorose ma ma hanno il vantaggio di essere indipendenti dal tempo. Se ci si dovesse localizzare solo con questi sensori si avrebbe l'effetto di saltare da una posizione all'altra. Questo effetto fa si che il robot credi di oscillare tra varie posizioni e rende impossibile lo svolgimento di mansioni complesse.

L'idea principale per la soluzione del problema della localizzazione risulta qundi l'integrazione dei due sensori. In particolare ci si localizza attraverso i dati forniti dai sensori ambientali solo quando l'errore stimato è molto piccolo e per il resto del tempo ci si localizza sfruttando gli encoder. In questo modo si evita di usare solo gli encoder per lunghi periodi di tempo, evitando di farli divergere.

L'approcio usato dal team Artisti Veneti sfutta proprio questa strategia. In particolare il sensore omnidirezionale permette di ricavare le posizioni delle linee e delle porte che stanno vicino al robot. Queste informazioni vengono poi filtrate in modo probabilistico al fine di eliminare il rumore del sensore alle prese con un ambiente altamente dinamico. Lo stato dell'arte attuale è rappresentato dalla così detta Localizzazione di Monte Carlo che utilizza i metodi di Monte Carlo come metodo di stima del Filtro di Bayes applicati alla robotica mobile. L'idea che sta alla base di tale tecnica è quella di rappresentare tutti i modelli del sistema (movimento e sensoristica

del robot) e dell'evoluzione di questo attraverso processi aleatori. Questo permette di avere libertà nella modellizzazione del sistema e di avere efficienza computazionale. I lavori svolti fino a oggi sulla localizzazione di Monte Carlo si sono sempre basati su robot equipaggiati da laser, sonar o da telecamere frontali. I primi ad utilizzare un sensore ottico omidirezionale sono stati gli Artisti Veneti con il lavoro di Alberto Pretto [20]. L'idea base è l'utilizzo del sensore omnidirezionale come un sensore di prossimità con la capacità di fornire non solo una distanza per ogni raggio considerato ma tutte le distanze relative a tutte le transizioni cromatiche di interesse per ogni raggio. Questo sistema di image-processing è stato chiamato Omidirectional Vision Range Finder (OVRF).

In questo lavoro verranno inizialmente ripresi i principi matematico-probabilistici che stanno alla base degli algoritmi di Monte Carlo per la localizzazione di robot mobili. Verrà poi descritta la strategia di localizzazione basata sul sensore OVRF. Sucessivamante saranno descritti i principali problemi riscontrati nell'utilizzo di questa strategia di localizzazione e le soluzioni apportate per risolverli. Verranno infine presentati i risultati degli esperimenti svolti in ambiente non RoboCup per dimostrare la totale generalità dell'approccio utilizzato.

1.1 Struttura della tesina

Diamo ora una breve descrizione dei vari capitoli della tesina.

Capitolo 2

Viene presentato in modo formale il Filtro di Bayes e viene sviluppato il calcolo per quello applicato a sistemi Markoviani. Vengono poi presentati in modo formale i principali metodi di Monte Carlo.

Capitolo 3

In questo capitolo i metodi introdotti formalmente nel capitolo 2 vengono sviluppati e adattati per la soluzione del problema di localizzazione di un robot mobile. Vengono anche presentati in pseudo codice gli algoritmi che verranno utilizzati nell'implementazione.

Capitolo 4

Si presentano in questo capitolo l'innovativo sensore, OVRF, sviluppato dagli Artisti Veneti e le varie parti di cui è composto il software utilizzato. Vengono poi messi in evidenza i problemi e le soluzioni implementate per migliorare i tempi di convergenza e di esecuzione dell'algoritmo.

Capitolo 5

In questo capitolo si presentano gli esperimenti effettuati in ambiente non RoboCup. Tali esperimenti dimostrano l'efficacia del metodo di localizzazione basato su algoritmi di Monte Carlo con il sensore OVRF anche in ambienti generici.

La tesina si conclude con due appendici contenenti alcuni richiami di Teoria delle Probabilità e la descrizione del robot usato nelle sperimentazioni.

Capitolo 2

Metodi di Monte Carlo

Alcuni dei problemi di analisi dei dati comportano la stima di alcune quantità variabili nel tempo a partire da delle misurazioni spesso affette da rumore. Alcuni esempi sono la stima del livello logico di un segnale digitale affetto da rumore o la stima dello stato (in genere posizione) di un robot all'interno di un ambiente.

Conoscendo a priori il sistema si può realizzare un modello Bayesiano, detto filtro di Bayes che lo descriva. Questo modello è caratterizzato dalla densità di probabilità a priori della quantità da stimare e da funzioni di probabilità che legano le misurazioni a tale quantità. Con la regola di Bayes sarà quindi possibile calcolare la densità di probabilità a posteriori (PDF) dello stato del sistema a partire dalle misurazioni. Nella maggior parte dei problemi reali tali misurazioni arrivano in ordine sequenziale e sarà quindi richiesta una nuova stima ad ogni arrivo di una misurazione. Tale stima andrà a modificare la PDF che risulta così dipendente dall'intera evoluzione passata del sistema. Questo approccio permette inoltre di non dover memorizzare tutte le misurazioni e di poter quindi essere computazionalmente veloce. Il filtro di Kalman ad esmpio fornisce una soluzione del problema di stima ma presenta alcuni difetti. Come prima cosa anche se teoricamente non è necessario il sistema per poter essere trattabile deve essere gaussiano. Nella realtà invece molte volte i sistemi da studiare sono non gaussiani, non lineari e di elevata dimensione il che rende praticamente inutilizzabile tale filtro per il calcolo della soluzione analitica ottima.

Una soluzione per tali problemi è data dai cosi detti Metodi di Monte Carlo Sequenziali (SMC). Questi algoritmi si basano sull'approssimazione della densità di probabilità a posteriori con un set di campioni. La distribuzione di questi campioni approssima sempre meglio la PDF al crescere del numero di campioni. Le caratteristiche principali di tali metodi sono la loro flessibilità e la facilità di implementazione che permette anche di parallelizzarne il calcolo.

2.1 Filtro di Bayes

Per effettuare stimo dello stato di un sistema dinamico a partire da dati sperimentali sono necessari il modello del sistema e il modello delle osservazioni. Questi modelli sono modelli probabilistici di processi aleatori e sono di conseguenza espressi attraverso la loro densità di probabilità. Il resto della trattazione si focalizzerà su sistemi markoviani non lineari e non gaussiani.

Il modello del sistema descrive l'evoluzione temporale dello stato del sistema. Tale stato viene indicato con $\{x_t : t \in N\}$ e $x_t \in \chi$ processo aleatorio di Markov con densità iniziale $p(x_0)$. L'evoluzione dello stato è rappresentata dall'equazione di transizione $p(x_t|x_{t-1})$

Il modello delle osservazioni descrive la relazione tra lo stato e le misurazione effettuate. Le osservazioni sono indicate con $\{y_t : t \in N^*\}$. La relazione tra stato e osservazione è data dalla densità marginale $p(y_t|x_t)$. Si assume che le misurazioni siano statisticamente indipendenti dato il processo $\{x_t : t \in N\}$, ovvero $p(y_t|x_t, ..., y_{t-1}) = p(y_y|x_t)$.

Definiamo con $x_{0:t} \doteq \{x_0, \ldots, x_t\}$ la sequenza degli stati e con $y_{1:t} \doteq \{y_1, \ldots, y_t\}$ la sequenza delle osservazioni dall'istante iniziale al tempo t. Da tale modello sarà necessario stimare ricorsivamente nel tempo la *densità di probabilità a posteriori (PDF)* $p(x_{0:t}|y_{1:t})$, o più frequentemente la sua densità marginale associata $p(x_t|y_{1:t})$, ed un'aspettazione:

$$I = E_{p(x_{0:t}|y1:t)}[f_t(x_{0:t})] \doteq \int f_t(x_{0:t})p(x_{0:t}|y_{1:t})dx_{0:t} \tag{2.1}$$

per qualche funzione f_t di interesse: ad esempio scegliendo $f_t(x_{0:t}) = x_{0:t}$ si ottiene la media statistica. E' inoltre possibile ottenere la *PDF* al tempo t utilizzando la *Regola di Bayes* (A.1):

$$p(x_{0:t}|y_{1:t}) = \frac{p(y_{1:t}|x_{0:t})p(x_{0:t})}{p(y_{1:t})} \tag{2.2}$$

La definizione di probabilità condizionata (A.3) ci dice che:

$$p(y_{1:t}) = p(y_t|y_{1:t-1})p(y_{1:t-1}) \tag{2.3}$$

$$p(x_{0:t}) = p(x_t|x_{0:t-1})p(x_{0:t-1}) \tag{2.4}$$

e poichè il modello del sistema è markoviano la (2.4) diventa:

$$p(x_{0:t}) = p(x_t|x_{t-1})p(x_{0:t-1}) \tag{2.5}$$

Dalla definizione di probabilità condizionata si ricava anche che:

$$p(y_{1:t}|x_{0:t}) = p(y_t|x_{0:t}, y_{1:t-1})p(y_{1:t-1}|x_{0:t}) \tag{2.6}$$

e dall'indipendenza statistica delle osservazioni la (2.6) diventa:

$$p(y_{1:t}|x_{0:t}) = p(y_t|x_t)p(y_{1:t-1}|x_{0:t-1}) \tag{2.7}$$

Sostituendo in (2.2) la (2.3), la (2.5) e la (2.7) otteniamo:

$$p(x_{0:t}|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|x_{t-1})}{p(y_t|y_{1:t-1})} \frac{p(y_{1:t-1}|x_{0:t-1})p(x_{0:t-1})}{p(y_{1:t-1})} \tag{2.8}$$

Quest'ultima (2.8) può essere riscritta come:

$$p(x_{0:t}|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|x_{t-1})}{p(y_t|y_{1:t-1})} p(x_{0:t-1}|y_{1:t-1}) \tag{2.9}$$

ottenendo una formula ricorsiva per il calcolo della densità a posteriori. La (2.9) si può anche scrivere come:

$$p(x_{0:t}|y_{1:t}) = \alpha p(y_t|x_t)p(x_t|x_{t-1})p(x_{0:t-1}|y_{1:t-1}) \tag{2.10}$$

con $\alpha = p(y_t|y_{1:t-1})^{-1}$ costante di normalizzazione perchè indipendente dalla variabile x. Nelle applicazioni, in genere, risulta più utile la *PDF marginale* $p(x_t|y_{1:t})$. Dalla *Regola di Bayes* possiamo scrivere:

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t, y_{1:t-1})p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} \tag{2.11}$$

per indipendenza statistica delle osservazioni la (2.11) diventa:

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} \tag{2.12}$$

Utilizzando la *regola marginale* (A.2) possiamo scrivere:

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1}, y_{1:t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \tag{2.13}$$

e ricordando che il modello del sistema è markoviano:

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \tag{2.14}$$

Utilizzando la *regola marginale* e l'indipendenza statistica delle osservazioni:

$$p(y_t|y_{1:t-1}) = \int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t \tag{2.15}$$

Sostituendo in (2.12) la (2.14) e la (2.15) otteniamo:

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t) \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t} \tag{2.16}$$

che rappresenta l'espressione ricorsiva per il calcolo della densità marginale. Frequentemente quest'ultima equazione (2.16) viene divisa in due passi:

- *predizione*

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \tag{2.17}$$

- *aggiornamento*

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t} \tag{2.18}$$

dove $\alpha = (\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t)^{-1}$ è una costante di normalizzazione.

Nella (2.17) viene applicato il modello del sistema $p(x_t|x_{t-1})$ alla *PDF marginale* calcolata al tempo $t-1$ senza tener conto dell'ultima osservazione y_t effettuata. Per questo motivo la (2.17) prende il nome di equazione di *predizione*. Quando viene effettuata una nuova osservazione y_t, con (2.18) viene *aggiornata* la densità calcolata con (2.17) ottenendo proprio la *PDF marginale* al tempo t. Infine da quest'ultima sarà possibile calcolare l'aspettazione cercata.

Nonostante questo approccio ricorsivo sembri un efficacie metodo di soluzione ottima del *Filtro di Bayes*, essa risulta invece, in molti casi, molto complessa: gli integrali coinvolti nella stima della *PDF* sono spesso troppo complessi e di dimensione troppo elevata per poter essere risolti analiticamente. Per questo motivo si sono cercati dei metodi approssimati per ottenere una soluzione approssimata del *Filtro di Bayes*.

2.2 Metodi di Monte Carlo

La soluzione ottima delle equazioni (2.17) e (2.18) risulta molto spesso difficile da calcolare in modo esatto a causa della complessità e della dimensione delle equazione coinvolte. Alcuni metodi risolutivi approssimati che si trovano in letteratura sono il filtro di Kalman esteso e i metodi Grid-Based approssimati. Un'altra serie di metodi per la ricerca di un asoluzione approssimata che si stà sviluppando in questi anni sono i così detti Metodi di Monte Carlo. Il vantaggio principale di tali metodi sta nel non essere soggetti alla linearità e alla gaussianità del sistema. Alla base di tali metodi sta l'idea di rappresentare le densità di probabilità che appaiono in (2.17) e (2.18) con un set di campioni estratti casualmente e in accordo con tali densità. Se a tali campioni vengono associati dei pesi si ottengono gli algoritmi Importance Sampling e se questi vengono realizzati in modo ricorsivo si ottengono i metodi Sequential Importance Sampling. Questi ultimi algoritmi però tendono a degenerare al crescere di t come è stato dimistrato in [7]. Questo problema è stato risolto

aggiungendo un ulteriore passo di campionamento [19] ottenendo i così detti metodi *Sampling Importance Resampling (SIR).*

2.3 Importance Sampling (IS)

Se avessimo a disposizione N campioni $\{x_{0:t}^{(i)} : i = 1, \ldots, N\}$, indipendenti ed identicamente distribuiti (i.i.d), in accordo con la densità a posteriori $p(x_{0:t}|y_{1:t})$, una stima empirica della densità a posteriori $p(x_{0:t}|y_{1:t})$ sarebbe data da:

$$P_N(x_{0:t}|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N}\delta(x_{0:t}^{(i)} - x_{0:t}) \tag{2.19}$$

quindi la stima dell'aspettazione (2.1) cercata:

$$I_N(f_t) = \int f_t(x_{0:t})P_N(x_{0:t}|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} f_t(x_{0:t}^{(i)}) \tag{2.20}$$

Sfortunatamente è di solito impossibile campionare efficientemente direttamente dalla densità a postriori $p(x_{0:t}|y_{1:t})$. Per aggirare tali limiti è stata proposta una soluzione alternativa: l'*Importance Sampling (IS)* [7]. L'idea chiave di tale strategia è quella di associare dei pesi ad ogni campione.

Definiamo un'arbitraria densità $\pi(x_{0:t}|y_{1:t})$, detta *densità importanza*, tale che il suo supporto includa il supporto della densità a posteriori $p(x_{0:t}|y_{1:t})$ e che sia facile generare campioni in accordo con essa,
$x^{(i)} \sim \pi(\cdot), i = 1, \ldots, N$. Per calcolare l'aspettazione (2.1) utilizzando la densità $\pi(\cdot)$ avremo:

$$I(f_t) = \frac{\int f_t(x_{0:t})w(x_{0:t})\pi(x_{0:t}|y_{1:t})dx_{0:t}}{\int w(x_{0:t})\pi(x_{0:t}|y_{1:t})dx_{0:t}} \tag{2.21}$$

in cui la funzione peso (*importance weight*) sarà data da:

$$w(x_{0:t}) = \frac{p(x_{0:t}|y_{1:t})}{\pi(x_{0:t}|y_{1:t})} \tag{2.22}$$

Per definizione da $\pi(\cdot)$ è sempre possibile estrarre campioni, quindi se vengono generati N campioni, indipendenti ed identicamente distribuiti (i.i.d), in accordo con $\pi(x_{0:t}|y_{1:t})$, sarà possibile ottenere una stima dell'aspettazione $I(f_t)$ (2.21):

$$\widehat{I}_N(f_t) = \frac{\frac{1}{N}\sum_{i=1}^{N} f_t(x_{0:t}^{(i)})w(x_{0:t}^{(i)})}{\frac{1}{N}\sum_{j=1}^{N} w(x_{0:t}^{(i)})} = \sum_{i=1}^{N} f_t(x_{0:t}^{(i)})\tilde{w}_t^{(i)} \tag{2.23}$$

dove i pesi normalizzati $\tilde{w}_t^{(i)}$ sono dati da:

$$\tilde{w}_t^{(i)} = \frac{w(x_{0:t}^{(i)})}{\sum_{j=1}^{N} w(x_{0:t}^{(j)})} \tag{2.24}$$

In altre parole con il metodo *Importance Sampling* approssimiamo la densità a posteriori $p(x_{0:t}|y_{1:t})$ cercata con:

$$\widehat{P}_N(x_{0:t}|y_{1:t}) = \sum_{i=1}^{N} \tilde{w}_t^{(i)} \delta(x_{0:t}^{(i)} - x_{0:t}) \tag{2.25}$$

dove:

$$\tilde{w}_t^{(i)} \propto \frac{p(x_{0:t}^{(i)}|y_{1:t})}{\pi(x_{0:t}^{(i)}|y_{1:t})} \tag{2.26}$$

mentre la stima dell'aspettazione è data dal calcolo dell'aspettazione rispetto alla stima della densità fornita in (2.25):

$$\widehat{I}_N(f_t) = \int f_t(x_{0:t}) \widehat{P}_N(x_{0:t}|y_{1:t}) \tag{2.27}$$

Uno dei problemi principali del metodo *Importance Sampling* è la sua inadeguatezza nel caso di stime ricorsive. Infatti, per stimare la densità $p(x_{0:t}|y_{1:t})$ è necessario disporre dell'intera sequenza di osservazioni $y_{1:t}$ e ogni volta che si rende disponibile una nuova misurazione, y_{t+1}, bisogna ricalcolare i pesi sull'intera sequanza di stati. La complessità computazionale sale così proporzionalmente al tempo t.

2.4 Sequential Importance Sampling (SIS)

Il metodo *Importance Sampling (SIS)* è stato modificato [2] in modo da renderlo sequenziale. Ad ogni iterazione è possibile ottenere una stima $\widehat{P}_N(x_{0:t}|y_{1:t})$ di $p(x_{0:t}|y_{1:t})$ senza modificare la sequenza di stati $\{x_{0:t-1}^{(i)} : i = 1, \ldots, N\}$ generata in precedenza. Per far ciò la funzione importanza $\pi(x_{0:t}|y_{1:t})$ si deve fattorizzare:

$$\pi(x_{0:t}|y_{1:t}) = \pi(x_{0:t-1}|y_{1:t-1}) \pi(x_t|x_{0:t-1}, y_{1:t}) \tag{2.28}$$

Iterando la quest'ultima otteniamo:

$$\pi(x_{0:t}|y_{1:t}) = \pi(x_0) \prod_{k=1}^{t} \pi(x_k|x_{0:k-1}, y_{1:k}) \tag{2.29}$$

Utilizzando (2.28), per ottenere un set di campioni $x_{0:t}^{(i)} \sim \pi(x_{0:t}|y_{1:t})$ basterà aggiungere ad ognuno dei sample $x_{0:t-1}^{(i)} \sim \pi(x_{0:t-1}|y_{1:t-1})$ il nuovo stato $x_t^{(i)} \sim \pi(x_t|x_{0:t-1}, y_{1:t})$. Tale strategia prende il nome di *Sequential Importance Sampling (SIS).*

Per calcolare ricorsivamente i pesi, basta sostituire la (2.10) e la (2.28) in (2.26):

$$\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)} \frac{p(y_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)})}{\pi(x_t^{(i)}|x_{0:t-1}^{(i)}, y_{1:t})} \tag{2.30}$$

Se $\pi(x_t|x_{0:t-1}, y_{1:t}) = \pi(x_t|x_{t-1}, y_t)$, la *densità importanza* dipende solo da x_{t-1} e y_t. In questo modo possiamo calcolare una stima di $p(x_t|y_{1:t})$ ogni volta che si rende disponibile una nuova osservazione y_t). Così facendo e sufficiente mantenere solo l'ultimo set di campioni $x_t^{(i)}, i = 1, \ldots, N$. In questo caso i pesi divengono:

$$\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)} \frac{p(y_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)})}{\pi(x_t^{(i)}|x_{t-1}^{(i)}, y_t)} \tag{2.31}$$

e la densità a posteriori $p(x_t|y_{1:t})$ può essere approssimata da:

$$p(x_t|y_{1:t}) \approx \sum_{i=1}^{N} \tilde{w}_t^{(i)} \delta(x_t - x_t^{(i)}) \tag{2.32}$$

dove i pesi sono definiti in (2.31).

In [5] è dimostrato che se $N \to \infty$ allora l'approssimazione data da (2.32) tende alla densità a posteriori $p(x_t|y_{1:t})$ reale.

E' stato dimostrato [7] che dopo un certo tempo la maggior parte dei sample (praticamente tutti tranne uno) hanno peso $\tilde{w}_t^{(i)}$ prossimo allo zero. Questo fatto ha due conseguenze negative: la prima è che la densità a posteriori viene approssimata mala in quanto c'è un solo campione rilevante; la seconda è che una gran parte del carico computazionale viene sprecato a calcolare pesi di sample irrilevanti per la stima della densità.

2.5 Sampling Importance Resampling (SIR)

Il metodo *Sampling Importance Resampling (SIR)* [19] è stato introdotto per risolvere i problemi del Sequential Importance Sampling.

L'idea che sta alla base di questo metodo è quella di eliminare i campioni con peso basso e di moltiplicare quelli con peso elevato. Questo ricampionamento viene eseguito ad ognoi iterazione dell'algoritmo e permette di avere sempre un notevole numero di campioni attorno allo stato più probabile.

Il ricampionamento viene eseguito in accordo con la distribuzione di massa $\widehat{P}_N(x_{0:t}|y_{1:t}) = \sum_{i=1}^{N} \tilde{w}_t^{(i)} \delta(x_{0:t}^{(i)} - x_{0:t})$ del set attuale. Essa viene quindi sostituita dalla nuova densità:

$$P_N(x_{0:t}|y_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} N_t^{(i)} \delta(x_{0:t}^{(i)} - x_{0:t}) \tag{2.33}$$

dove $N_t^{(i)}$ è il numero di volte che il campione $x_{0:t}^{(i)}$ viene selezionato, inoltre si ha: $\sum_{i=1}^{N} N_t^{(i)} = N$. Se $N_t^{(j)} = 0$ significa che il campione $x_{0:t}^{(j)}$ è stato eliminato. In questo modo il passo di ricampionamento produce un nuovo set di N campioni, indipendenti ed identicamente distribuiti (i.i.d) $\{x_{0:t}^{(i)} : i = 1, \ldots, N\}$, ognuno con peso associato $\tilde{w}_t^{(i)} = \frac{1}{N}$. Gli $N_t^{(i)}$ vengono scelti in modo tale che, per ogni funzione f_t:

$$\int f_t(x_{0:t}) P_N(x_{0:t}|y_{1:t}) \approx \int f_t(x_{0:t}) \widehat{P}_N(x_{0:t}|y_{1:t}) \tag{2.34}$$

Dopo la fase di ricampionamento i samples $x_{0:t}^{(i)}$ con $N_t^{(i)} > 0$, saranno approssimativamente distribuiti in accordo con la densità $p(x_{0:t}|y_{1:t})$. Avendo scelto come *densità importanza* $p(x_t|x_{t-1}^{(i)})$, la (2.31) diventerà:

$$\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)} p(y_t|x_t^{(i)}) \tag{2.35}$$

Ad ogni iterazione, dopo il passo di ricampionamento, abbiamo che $\tilde{w}_t^{(i)} = \frac{1}{N} \forall i$, quindi la (2.35) si semplifica in:

$$\tilde{w}_t^{(i)} \propto p(y_t|x_t^{(i)}) \tag{2.36}$$

In (**Algoritmo 1**) viene presentato lo pseudo codice dell'algoritmo *SIR*.

Algoritmo 1 : Sampling Importance Resampling

$\{x_t^{(i)}, \tilde{w}_t^{(i)}\}_{i=1}^{N} = SIR(\{x_{t-1}^{(i)}, \tilde{w}_{t-1}^{(i)}\}_{i=1}^{N}, y_t)$

- **for** $i = 1 : N$
  - Campiona $x_t^{(i)} \sim p(x_t|x_{t-1}^{(i)})$
  - Assegna al campione appena trovato il peso $\tilde{w}_t^{(i)} = p(y_t|x_t^{(i)})$
- **end for**
- Normalizza i pesi $\tilde{w}_t^{(i)}$

- Ricampiona con sostituzione N samples dal set $(x_t^{(i)}, i = 1, \ldots, N)$ in accordo con il loro peso

---

Tra i vari algoritmi presenti in letteratura per il passo di ricampionamento è stato scelto quello presentato in [15] e sintetizzato in **Algoritmo 2**.

Algoritmo 2 : Ricampionamento

$\{\hat{x}^{(i)}, \hat{w}^{(i)}\}_{i=1}^{N} = RES(\{x^{(i)}, w^{(i)}\}_{i=1}^{N})$

- Inizializza la CDF: $c_1 = w^{(1)}$
- **for** $i = 2 : N$
  - Calcola la CDF: $c_i = c_{i-1} + w^{(i)}$
- **end for**
- Scegli un punto di partenza casuale, estratto dalla densità uniforme nell'intervallo $[0, 1/N]$: $u_1 \sim \mathbb{U}[0, 1/N]$
- Inizio dal primo campione: $i = 0$
- **for** $i = 1 : N$
  - $u_j = u_1 + \frac{1}{N}(j - i)$
  - **while** $u_j > c_i$
    - $i = i + 1$
  - **end while**
  - $\hat{x}^{(j)} = x^{(i)}$
  - $\hat{w}^{(i)} = \frac{1}{N}$
- **end for**

Il passo di ricampionamento riduce il problema della degenerazione presente nel metodo *SIS* però introduce un problema di *impoverimento dei campioni*: dopo il passo di ricampionamento, infatti, tutti i campioni tendono a collassare su quelli con peso più elevato. Nonostante questo i metodi *SIR* sono un potente strumento per la risoluzione approssimata del filtro di Bayes.

Fin quì abbiamo presentato i *Metodi di Monte Carlo* nel caso generale, vedremo nel capitolo successivo l'applicazione di tali metodi al caso specifico della localizzazione di robot mobili. La strategia adottata deriva dal metodo generale *Sampling Importance Resampling(SIR)* (sez. 2.5).

Capitolo 3

Localizzazione di Monte Carlo per robot mobili

Quando ci riferiamo al problema della localizzazione di robot mobili affrontiamo il problema della stima della posizione, come coordinate 2D sul piano, e dell'orientazione, come angolo rispetto un asse di riferimento, di un robot mobile. Un sistema di localizzazione per essere considerato efficiente deve essere capace di risolvere i seguenti problemi:

- **Position tracking [14]**
  Data una posizione iniziale il robot deve saper aggiornare tale posione in base al movimento effettuato. In genere per aggiornare la posizione ci si serve dei dati dell'odometria forniti dagli encoder del robot. Questi dati sono soggetti agli errori dovuti agli slittamenti del robot e all'imprecisione degli strumenti utilizzati. Tali errori dipendono dallo spazio percorso.

- **Localizzazione globale [25]**
  Una volta inserito il robot in un ambiente conosciuto, esso deve ricave la posizione assoluta in cui si trova. Generalmente tale problema viene risolto facendo uso di sensori di prossimità o di sensori ottici.

- **Kidnapped robot [21]**
  Una volta che il robot si è localizzato lo si sposta in un altra posizione senza fornirgli informazioni odometriche. Tale problema è più complesso della localizzazione globale perchè il robot non ha coscenza di essersi spostato e quindi di non essere più localizzato correttamente.

Naturalmente questi problemi si complicano notevolmente all'aumentare della dinamicità dell'ambiente (persone e/o che muovendosi impediscono di vedere l'ambiente circostante) e della simmetria dell'ambiente.

Lo stato attuale della ricerca indica i metodi di *Monte Carlo* [13, 6, 12, 3, 23] come la strategia migliore per risolvere in modo efficiente e robusto i problemi sopra indicati.

Tutti gli algoritmi introdotti condividono lo stesso modello probabilistico di base che è quello che abbiamo presentato in forma generale nel capitolo 2. Presentiamo, ora, una versione adattata del *Filtro di Bayes* (cap. 2.1) al problema della localizzazione.

3.1 Filtro di Bayes per la localizzazione

Il filtro di Bayes serve per stimare lo stato di un sistema dinamico a partire da una serie di misurazioni. Nel caso specifico della localizzazione

- lo stato da stimare è rappresentato da un vettore tridimensionale $l_t = (x, y, \theta) \in R \times R \times [-\pi, \pi)$ in cui (x, y) rappresenta la posizione e θ l'orientazione. L_t rappresenta la corrispondente variabile aleatoria.
- Il sistema dinamico è rappresentato dal robot in movimento.
- Le misurazioni sono date dai sensori di prossimità (o_t), range finder come laser, sonar e immagini omnidirezionali, e dalle informazioni odometriche degli encoder (a_t relativa allo spostamento avvenuto dal tempo t al tempo $t+1$). O e A rappresentano le corrispondenti variabili aleatorie.

Definiamo, inoltre, $d_{0:t}$ come una serie di misurazioni odometriche e sensoriali ordinate in ordine temporale. Infine, la densità di probabilità a posteriori dello stato (belief del robot), ovvero la localizzazione, è definita come

$$Bel(l_t) = p(l_t|d_{0:t}) \tag{3.1}$$

All'istante iniziale $t = 0$ il belief viene inizializzato da una densità uniforme dello spazio degli stati che corrisponde alla soluzione del problema della localizzazione globale. A regime, invece, si dovrà stimare la (3.1) ogni volta che si renda disponibile una nuova misurazione. Risulta a tal fine utile derivare una versione ricorsiva del filtro di Bayes. Applicando a (3.1) la *Regola di Bayes* (A.1) si ottiene:

$$Bel(l_t) = \frac{p(o_t|l_t, a_{t-1}, o_{t-1}, a_{t-2}, \ldots, o_0)p(l_t|a_{t-1}, o_{t-1}, a_{t-2}, \ldots, o_0)}{p(o_t|a_{t-1}, o_{t-1}, a_{t-2}, \ldots, o_0)} \tag{3.2}$$

poichè il denominatore è costante rispetto alla variabile l_t, possiamo scrivere:

$$Bel(l_t) = \alpha p(o_t|l_t, a_{t-1}, o_{t-1}, a_{t-2}, \ldots, o_0)p(l_t|a_{t-1}, o_{t-1}, a_{t-2}, \ldots, o_0) \tag{3.3}$$

con $\alpha = p(o_t|a_{t-1}, o_{t-1}, a_{t-2}, \ldots, o_0)^{-1}$ costante di normalizzazione. Dall'assunzione di Markovianità fatta per il *Filtro di Bayes*, i dati passati e futuri sono indipendenti se è noto lo stato attuale l_t, cioè:

$$p(o_t|l_t, a_{t-1}, o_{t-1}, a_{t-2}, \ldots, o_0) = p(o_t|l_t) \tag{3.4}$$

quindi, sostituendo (3.4) in (3.3) si ottiene:

$$Bel(l_t) = \alpha p(o_t|l_t)p(l_t|a_{t-1}, o_{t-1}, a_{t-2}, \ldots, o_0) \tag{3.5}$$

Sfruttando, poi, il *Teorema della probabilità totale* (A.4):

$$Bel(l_t) = \alpha p(o_t|l_t) \int p(l_t|l_{t-1}, a_{t-1}, \ldots, o_0)p(l_{t-1}|a_{t-1}, \ldots, o_0)dl_{t-1} \tag{3.6}$$

Riapplicando la Markovianità abbiamo che: $p(l_t|l_{t-1}, a_{t-1}, \ldots, o_0) = p(l_t|l_{t-1}, a_{t-1})$, quindi si può riscrivere la (3.6) come:

$$Bel(l_t) = \alpha p(o_t|l_t) \int p(l_t|l_{t-1}, a_{t-1})p(l_{t-1}|a_{t-1}, \ldots, o_0)dl_{t-1} \tag{3.7}$$

e sostituendo, infine, la definizione base di *belief* (3.1) in (3.7) otteniamo l'equazione ricorsiva per il *Filtro di Bayes*:

$$Bel(l_t) = \alpha p(o_t|l_t) \int p(l_t|l_{t-1}, a_{t-1})Bel(l_{t-1})dl_{t-1} \tag{3.8}$$

Per calcolare la (3.8) è necessario conoscere due densità:

- $p(l_t|l_{t-1}, a_{t-1})$, che viene definita *modello del movimento* in cui entrano in gioco le informazioni provenienti dall'odometria, quindi dal movimento del robot
- e $p(o_t|l_t)$, detta *modello delle osservazioni* in cui entrano in gioco le informazioni provenienti dal sensore ambientale come telecamera, laser, sonar, ecc.

Entrambi questi modelli sono tipicamente *stazionari*, non dipendenti cioè da uno specifico tempo t, per cui è possibile ometterne l'indicazione. Il calcolo del *belief* (3.8) si può, quindi, dividere in due passi:

$$Movimento : Bel^-(l_t) = \int p(l_t|l_{t-1}, a_{t-1})Bel(l_{t-1})dl_{t-1} \tag{3.9}$$

$$Osservazione : Bel(l_t) = \alpha p(o_t|l_t)Bel^-(l_t) \tag{3.10}$$

3.2 Localizzazione di Monte Carlo

In (sez. 2.2) sono stati presentati i *Metodi di Monte Carlo* nel caso generale, vediamo ora di applicare tali metodi al caso specifico della localizzazione di robot mobili. La strategia adottata deriva dal metodo generale *Sampling Importance Resampling(SIR)* (sez. 2.5). Per semplicità ci riferiremo a tale applicazione semplicemente con il nome di *localizzazione di Monte Carlo(MCL)*. L'idea di base della *MCL* è quella di rappresentare il *belief* (3.1) per mezzo di un set di N campioni pesati:

$$Bel(l_t) \approx \{l^{(i)}, w^{(i)}\}_{i=1,\dots,N} \tag{3.11}$$

Ogni $l^{(i)}$ è un campione della variabile aleatoria L e rappresenta un'ipotesi di stato, cioè un ipotesi di localizzazione nel nostro caso. Il parametro numerico $w^{(i)}$, non negativo, viene chiamato *fattore importanza* del campione. I *fattori importanza* vengono normalizzati a 1 e rappresentano il peso (importanza) di ogni sample. Il set di campioni pesati definisce così una funzione probabilità discreta che approssima la densità continua $Bel(l_t)$.

Il set iniziale di campioni rappresenta la conoscenza iniziale del robot, $Bel(l_0)$. Nel caso di *localizzazione globale*, quando cioè il robot non si è ancora localizzato, il *belief* viene rappresentato da un set di campioni estratti da una densità di probabilità uniforme sullo spazio di tutti i possibili stati, e i *fattori importanza* sono tutti inizializzati a $1/N$.

A regime, l'aggiornamento del *belief* viene realizzato in tre passi:

1. Si campiona $l_{t-1}^{(i)} \sim Bel(l_{t-1})$ dal set di samples pesati che rappresentano $Bel(l_{t-1})$. Per il campionamento è stata utilizzata la strategia introdotta in (**Algoritmo 2**, sez. 2.5). Ogni campione è distribuito in accordo con tale distribuzione.

2. Si applica ad ogni campione ottenuto al punto precedente il *modello del movimento*, cioè dato $l_{t-1}^{(i)}$ si campiona $l_t^{(i)} \sim p(l_t|l_{t-1}^{(i)}, a_{t-1})$. Ovviamente la coppia $\langle l_{t-1}^{(i)}, l_t^{(i)} \rangle$ è distribuita in accordo con la distribuzione q_t, prodotto del *belief* allo stato $t-1$ e della distribuzione che rappresenta il *modello del movimento*:

   $$q_t = p(l_t|l_{t-1}, a_{t-1}) \times Bel(l_{t-1}) \tag{3.12}$$

   q_t viene chiamata "proposal density", nel senso che essa "propone" i campioni alla densità a posteriori cercata (3.8).

3. Si applica infine il *modello delle osservazioni*, ottenendo la densità a posteriori cercata:

   $$\alpha p(o_t|l_{t-1}^{(i)}) p(l_t^{(i)}|l_{t-1}^{(i)}, a_{t-1}) Bel(l_{t-1}^{(i)}) \tag{3.13}$$

Questo significa assegnare al campione $l_t^{(i)}$ il *fattore importanza* dato da:

$$w^{(i)} = p(o_t|l_t^{(i)}) \tag{3.14}$$

Si può notare come il *fattore importanza* $w^{(i)}$ sia stato ottenuto [1] come quoziente della densità cercata e della densità propositiva, a meno di una costante di normalizzazione α:

$$\frac{\alpha p(o_t|l_{t-1}^{(i)})p(l_t^{(i)}|l_{t-1}^{(i)}, a_{t-1})Bel(l_{t-1}^{(i)})}{p(l_t^{(i)}|l_{t-1}^{(i)}, a_{t-1})Bel(l_{t-1}^{(i)})} = \alpha p(o_t|l_t^{(i)}) \tag{3.15}$$

Tale processo viene eseguito per ciascuno degli N campioni utilizzati per il filtro, producendo un nuovo set di N samples pesati $l_t^{(i)}, i = 1, \ldots, N$. L'ultimo passo del processo consiste nella normalizzazione dei *fattori importanza* in modo che la loro somma dia 1.

In (**Algoritmo 3**) viene schematizzato in pseudo codice l'algoritmo.

Algoritmo 3 : Localizzazione di Monte Carlo

$\{l_t^{(i)}, w_t^{(i)}\}_{i=1}^N = MCL(\{l_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^N, o_t, a_{t-1})$

- **for** $j = 1 : N$
  - Campiona l', scelto casualmente dal set di campioni $l_{t-1}^{(i)}$ in accordo coi loro pesi $w_{t-1}^{(i)}$, $i = 1, \ldots, N$, utilizzando la strategia introdotta in (**Algoritmo 2**)
  - Genera il campione l, in accordo con il *modello del movimento*, cioè $l \sim p(l|l', a_{t-1})$
  - Calcola $w = p(o_t|l)$ utilizzando il *modello delle osservazioni*
- **end for**
- Normalizza i pesi $w^{(j)}, j = 1, \ldots, N$
- Ritorna $\{l^{(j)}, w^{(j)}\}_{j=1}^N$

In altre parole:

- si distibuiscono N sample l' casualmente nell'ambiente;
- all'arrivo di ogni nuova osservazione, cioè ogni volta che si elabora un'immagine:

  - si spostano i campioni in accordo con il il *modello del movimento*;
  - si calcola il peso di ogni campione $w = p(o_t|l)$ utilizzando il *modello delle osservazioni*;
  - se necessario si calcola la media e la dispersione dei samples per ottenere la posizione e l'errore sulla posizione.

I vantaggi principali della *Localizzazione di Monte Carlo* sono:

- Facilità di adattamento a diverse distribuzioni di rumore e modelli di sensore.
- Discretizzazione della densità che permette di adattare il numero di campioni alle esigenze di velocità e affidabilità che vogliamo.
- La maggior parte del tempo di calcolo viene spesa nelle zone in cui la densità a posteriori è più elevata.

Gli svantaggi principali sono:

- Difficoltà nel risolvere il problema del kidnapped robot perchè le zone a bassa densità a posteriori contengono pochi campioni.
- Spostamento dei campioni in modoo non corretto a causa di slittamenti delle ruote.
- In caso di simmetrie dell'ambiente ci sono zone equiprobabili in cui è indecidibile la localizzazione.

Capitolo 4

Implementazione dell'algoritmo di localizzazione

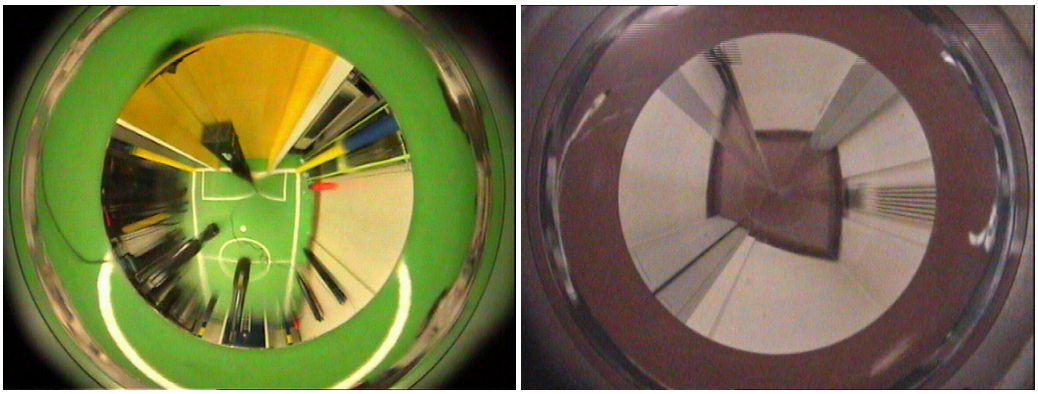
Nei capitoli precedenti sono stati introdotti i concetti base necessari per realizzare un algoritmo di localizzazione che sfrutti i metodi di Monte Carlo. In tali capitoli, però, non si è fatto nessun riferimento esplicito al tipo di modello delle osservazioni e del movimento utilizzato e non si è discusso nemmeno del tipo di sensore di prossimità utilizzato perchè strettamente dipendenti dall'applicazione che si vuole realizzare, dalla sensoristica e dal tipo di piattaforma disponibile.

In questo capitolo presenteremo le soluzioni adottate da Alberto Pretto [20] per il team RoboCup Artisti Veneti. Tali soluzioni verrano poi esaminate criticamente e verrano presentate le soluzioni implementate per rendere tale algoritmo di localizzazione efficente e capace di soddisfare le caratteristiche real time necessarie per poter giocare efficacemente in ambiente RoboCup.

4.1 Sensore di prossimità: Omni Vision Range Finder (OVRF)

Il campo in cui si giocano le competizione di RoboCup non ha strutture fisiche facilmente utilizzabili per la localizzazione. Le uniche strutture fisse sono infatti le porte e i corner flags (le bandierine agli angoli del campo). I colori, invece, sono fortemente strutturati, ogni oggetto, cioè, deve essere di un colore prestabilito dal regolamento. Queste carratteristiche fanno si che i sensori di prossimità, come sonar e laser, risultino praticamente inutili al fine della localizzazione e che le telecamere giochino invece un ruolo fondamentale. Il problema principale è che la localizzazione di Monte Carlo è stata pensata per funzionare con sensori di prossimità e non con immagini.

La soluzione introdotta [20] è stata quella di utilizzare le immagini omnidirezionali acquisite dai robot come un sensore di prossimità. I robot degli Artisti Veneti dispongono, infatti, di sensori per l'acquisizione delle immagini omnidirezionali. Tali sensori (fig. 4.1) sono formati da una telecamera che punta verso una specchio prospettico conico che permette di vedere tutto cio che circonda il robot. Lo specchio è dotato anche di una zona di prossimità che permette di vedere la zona più vicina al robot con un'alta risoluzione.



(a) (b)

Figura 4.1: Immagine omnidirezionale di un campo RoboCup (a) e di un corridoio (b)

Nelle immagine non si riescono a identificare facilmente gli ostacoli fisici e quindi si è pensato di cercare le discontinuità cromatiche che sono ben note a priori in quanto definite dal regolamento. I colori vengono quantizzati e si passa da uno spazio di $2^{16} = 65536$ colori a uno con i soli 8 colori stabiliti dal regolamento.

La strategia utilizzata per la ricerca delle transizioni cromatiche risulta molto semplice. Partendo dal centro dell'immagine, corrispondente anche al centro dello specchio prospettico e del robot, si scandisce radialmente l'immagine lungo delle direzioni prestabilite cercando le transizioni cromatiche di interesse. Attualmente cerchiamo lungo 60 direzioni equidistanti, cerchiamo, cioè, lungo linee che distano 6° tra loro. Per ogni direzione scandita viene memorizzata la distanza a cui si trova la transizione o un valore caretteristico che indica se non si è trovata la transizione o se si è incontrato un ostacolo e non si è quindi capaci di stabilire se ci sia e a che distanza si trovi tale transizione. In pseudocodice (Algoritmo 4):

Algoritmo 4 : OVRF

$dist[N] = OVRF()$

- **for** $i = 1 : N$
  - $dist[i] = INFINITY$
  - $x = 0; y = 0; ray = 0;$
  - $lastColor = QUANT(x, y)$
  - **for** $ray = 1 : MAX\_RAY$
    - $x = ray * cos(\alpha_i); y = ray * sin(\alpha_i);$
    - $color = QUANT(x, y)$
    - **if** ($lastColor- > color$ è il passaggio di colore cercato)
      - $dist[i] = REAL\_DIST(x, y)$
      - **break**
    - **end if**
    - $lastColor = color$
  - **end for**
- **end for**

4.1.1 Problemi riscontrati e modifiche apportate

Il problema principale del sensore OVRF è la sua dipendenza dalla quantizzazione. Una piccola variazione della luminosità, infatti, porta ad uno slittamento dei colori. In queste condizioni alcuni pixel passano da un segmento di colore ad un altro creando macchioline di colore indesiderato. Per esempio nei campi RoboCup è facile vedere macchie gialle nel bianco e nel rosso e macchie blu nel nero. La soluzione ottimale del problema sarebbe la modifica della strategia di quantizzazione o l'implementazione di alguni algoritmi di edge-detection che evidenzino le transizioni desiderate. Una possibile strategia di quantizzazione è presentata in [24]. Nonostante i notevoli vantaggi portati dal passaggio ad un nuovo tipo di quantizzazione tale passaggio comporterebbe anche la riscrittura di una notevole parte del software di visione adottato dal team Artisti Veneti. Per questo motivo è stato deciso di migliorare temporaneamente l'algoritmo OVRF semplicemente cercando le transizioni non come il passagio tra due pixel di colore opportuno ma come una transizione di due gruppetti di pixel. Questa modifica è stata effettuata,

però, solo sulle transizione verde-blu e verde-giallo in quanto le transizioni verde-bianco sono le transizioni con le linee del campo che essendo larghe solo $5cm$ vengono generalmente mappate in pochissimi pixel.

4.2 Modello del movimento

Il *modello del movimento* è stato definito come la densità $p(l_t|l_{t-1}, a_{t-1})$. Essa è una descrizione probabilistica della cinematica del robot infatti descrive la posizione attesa l_t del robot come l'applicazione del movimento a_{t-1} alla posizione precedente l_{t-1}. Più in particolare la densità $p(l_t|l_{t-1}, a_{t-1})$ fornisce la probabilità di trovarsi in ogni l_t dopo aver compiuto uno spostamento a_{t-1} da l_{t-1}. Il movimento deve essere considerato in modo proabilistico per poter tener conto degli errori dei sensori e degli errori dovuti a slittamenti e spinte.

Seguendo le idee presentate in [26] il movimento del robot tra due posizioni $l_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1})$ e $l_t = (x_t, y_t, \theta_t)$ viene riassunto dalla terna (α_u, T, θ_f) (fig. 4.2) che rappresenta:

- α_u rappresenta l'*angolo di uscita* dalla posizione l_{t-1} verso l_t, rispetto all'orientazione iniziale θ_{t-1}: questo significa che il robot si muove nella direzione simulata $\theta_T = \theta_{t-1} + \alpha_u$ nel sistema di riferimento assoluto. Poiché il robot a cui ci riferiamo è olonomo non è necessaria alcuna rotazione iniziale per allinearsi all'angolo di uscita
- T è la *traslazione* in linea retta nella direzione θ_T
- θ_f, infine, è la *rotazione* rispetto all'orientazione iniziale θ_{t-1}

Il robot usato (vedi Appendice B) è olonomo e quindi il movimento è rappresentato da una traslazione T lungo la direzione θ_T e da una rotazione finale θ_f.

Gli errori più rilevanti di cui si è deciso di tener conto nell'implementazione sono:

- *Errore assoluto sull'angolo di uscita*: è l'errore assoluto Δ_α nella rilevazione dell'angolo di uscita da parte dell'odometria. Esso è assoluto nel senso che non dipende dall'entità dell'angolo.
- *Errore sulla traslazione*: è l'errore $\Delta_{\mathrm{T}}(T)$ sulla distanza percorsa nella traslazione tra i due punti. Esso dipende dalla lunghezza della traslazione.
- *Errore sulla rotazione (dipendente dalla rotazione)*: è l'errore $\Delta_{rr}(\theta)$ sulla rilevazione della rotazione effettuata. Esso dipende dall'entità della rotazione.

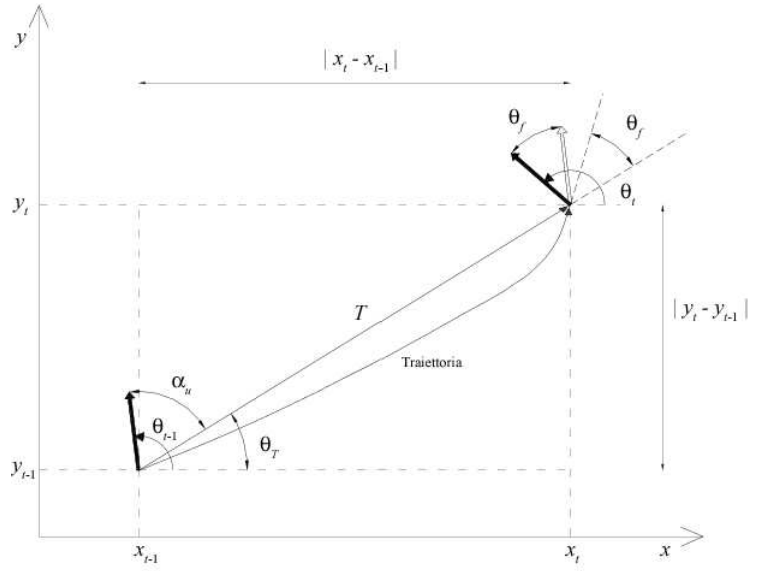


Figura 4.2: Schema del movimento del robot tra due posizioni $l_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1})$ e $l_t = (x_t, y_t, \theta_t)$ la traiettoria reale è continua ma viene descritta da una terna (α_u, T, θ_f) cioè da un angolo di uscita α_u, una traslazione T e una rotazione θ_f.

- *Errore sulla rotazione (dipendente dalla traslazione)*: è l'errore $\Delta_{r\mathrm{T}}(T)$ sulla rotazione legato alla traslazione T effettuata.

Otteniamo così che:

$$\begin{aligned} \alpha'_u &= \alpha_u + \Delta_\alpha \quad ; \\ T' &= T + \Delta_{\mathrm{T}}(T) \quad ; \\ \theta' &= \theta + \Delta_{rr}(\theta) + \Delta_{r\mathrm{T}}(T) \quad . \end{aligned}$$

Gli errori vengono modellati con delle variabili aleatorie gaussiane a media nulla e deviazione standard:

- σ_α è la deviazione standard dell'errore sull'angolo, in gradi;
- σ_{T} è la deviazione standard dell'errore sulla distanza per unità di traslazione cioè in cm/m o mm/m;
- σ_{rr} è la deviazione standard dell'errore di rotazione per unità di rotazione, in gradi/360°;
- $\sigma_{r\mathrm{T}}$ è la deviazione standard dell'errore di rotazione per unità di traslazione, in gradi/m.

Come spiegato nel capitolo 3 questo modello del movimento deve essere applicato ad ogni sample dopo il passo di ricampionamento. In altri termini ad ogni sample viene applicato l'Algoritmo 5. La funzione GaussianRandom(σ) restituisce un numero casuale distribuito secondo una densità di probabilità gaussiana di media nulla e varianza σ^2.

Algoritmo 5 : Modello del Movimento

$(\widetilde{x}, \widetilde{y}, \widetilde{\theta}) = MOV((x, y, \theta), (\alpha_u, T, \theta_f))$

- $\alpha'_u = \alpha_u + \text{GaussianRandom}(\sigma_\alpha)$
- $T' = T \cdot (1 + \text{GaussianRandom}(\sigma_{\text{T}}))$
- $\theta'_f = \theta_f + \theta_f \cdot \text{GaussianRandom}(\sigma_{rr}) + T \cdot \text{GaussianRandom}(\sigma_{r\text{T}})$
- $\widetilde{x} = x + T' \cdot cos(\theta + \alpha'_u)$
- $\widetilde{y} = y + T' \cdot sin(\theta + \alpha'_u)$
- $\widetilde{\theta} = \theta + \theta'_f$

In figura 4.3 il modello $p(l|l', a)$ è stato applicato ad un set di campioni per una sequanza di letture odometriche rappresentate dalla linea nera: nella posizione di partenza i campioni sono addensati in un unico punto, che rappresenta la posizione certa fornita dall'esterno. La distribuzione dei samples nei passi successivi costituisce un'approssimazione della densità di probabilità che rappresenta il *modello del movimento* del robot.

4.2.1 Problemi riscontrati e modifiche apportate

Durante gli esperimenti in laboratorio si è riscontrato che in caso di veloci oscillazioni del robot durante le quali si parte da un punto per tornarvi subito dopo la terna (α_u, T, θ_f) risulta praticamente nulla. Lo spostamento globale risulta infatti nullo anche se il robot si è mosso. Questa situazione si verifica spesso in un campo RoboCup quando il portiere si muove lungo i pali della porta e quando attaccante e difensore oscillano attorno alla palla cercando l'occasione per driblare l'avversario. In queste circostanze i sample vengono mantenuti fermi mentre la dispersione globale dovrebbe aumentare. Il problema è stato risolto sfruttando la gran quantità di informazioni odometriche che si hanno rispetto a quelle visuali. Nei robot usati infatti si

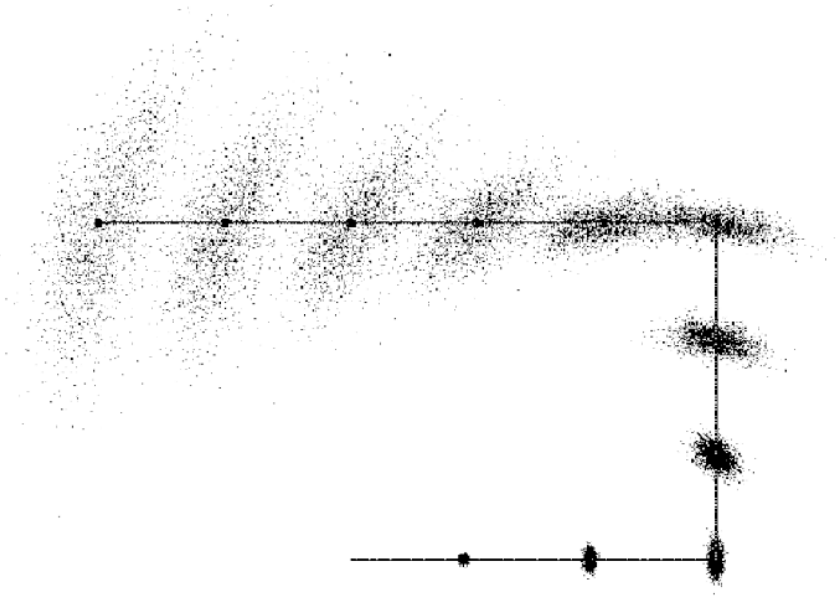


Figura 4.3: Applicazione del *modello del movimento*: la linea spezzata rappresenta la traiettoria che il robot deve tenere utilizzando solo l'odometria

riesce ad eseguire la localizzazione di Monte Carlo solo su 7 o 8 immagini al secondo mentre i driver di movimentazione riescono a fornire più di 25 informazioni odometriche al secondo. E' stato, così, deciso di applicare il modello del movimento ogni volta che risulta disponibile un dato odometrico. Questa scelta ha anche permesso di avere sempre i samples aggiornati per una veloce lettura della localizzazione. La localizzazione viene infatti calcolata mediando le posizioni $l_t = (x_t, y_t, \theta_t)$ di ciascuna particella. Risulta inoltre possibile avere anche la deviazione standard sulla localizzazione, dato che risulta utile per capire la dispersione dei samples nell'ambiente e per avere quindi un'informazione della bontà della localizzazione.

4.3 Modello delle osservazioni

Come spiegato nel capitolo 3 una volta applicato il modello del movimento ad ogni campione si deve calcolare il *fattore di importanza (peso)* di ogni campione in base all'ultima osservazione proveniente dai sensori ambientali. Più precisamente bisogna calcolare $p(o|l)$ per ogni campione che corrisponde alla probabilità di ottenere una certa osservazione o data una determinata ipotesi di localizzazione l.

Il sensore ambientale che abbiamo utilizzato è OVRF (sez. 4.1). L'utilizzo di tale sensore permette di avere informazioni diverse rispetto agli usuali sensori range finder come i laser:

- OVRF fornisce dati per tutte le direzioni mentre i laser solo per 180 gradi;
- OVRF è più rumoroso e meno preciso dei laser;
- OVRF non ha bisogno di superfici riflettenti come i laser e fornisce maggiori informazioni perchè riesce a distingure più transizioni cromatiche. Nel caso RoboCup le transizioni cercate sono quelle verde-bianco, verde-giallo, verde-blu;

Per calcolare $p(o|l)$ è necessario avere un modello ideale del sensore, avere una statistica degli errori commessi dal sensore e avere un metodo per integrare le misurazioni lungo ogni scansione per ogni transizione cromatica in un unico valore capace di indicare la bontà delle ipotesi di localizzazione. Più concretamente abbiamo bisogno delle distanze a cui si dovrebbero vedere le transizioni cromatiche se il sensore non fosse affetto da errori e di sapere come l'errore influenza le misurazioni per poterle pesare.

Definiamo $g_k(l, a_i)$ la distanza a cui si trova la transizione k data la posizione l lungo la direzione a_i. Il modello del sensore ideale è stato realizzato fornendo le mappe delle transizioni cromatiche e degli ostacoli dell'ambiente (fig. 4.4, 4.5).

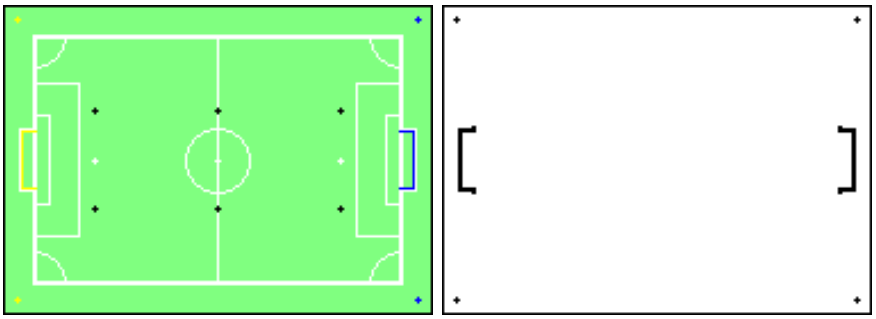
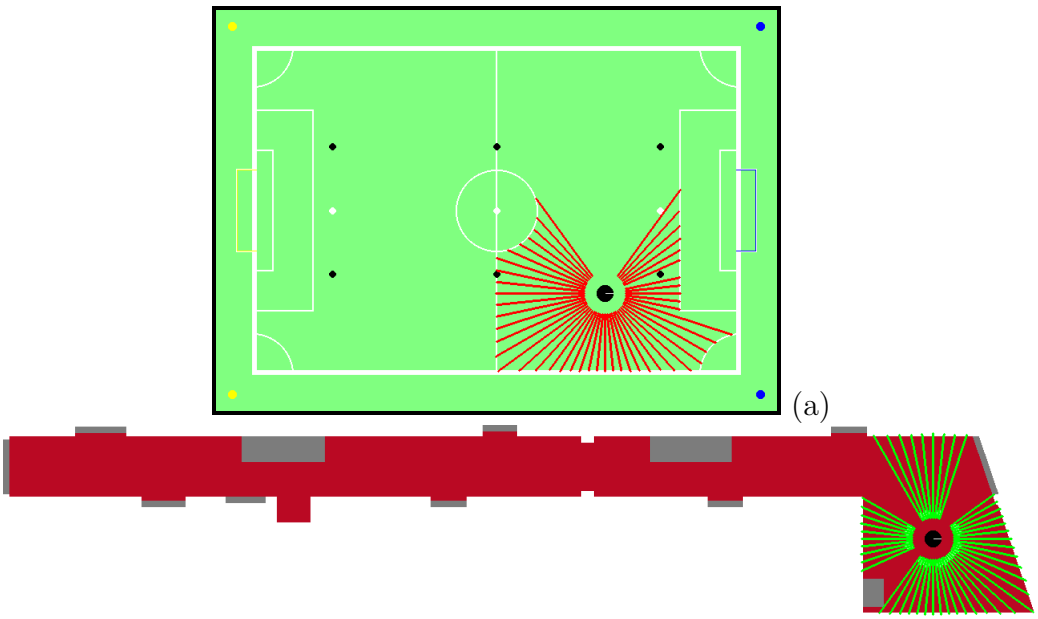


Figura 4.4: ColorMap e BuildingsMap di un campo RoboCup

Sulla mappa cromatica si esegue un algoritmo di ray-tracing per individuare le transizioni e poi si utilizza la mappa delle occupazioni per stabilire se la transizione può essere vista o se si trova nascosta da un ostacolo. In figura 4.6 si posono vedere le transizioni verde-bianco individuate in un campo RoboCup e quelle rosso-bianco di un corridoio del DEI.

Figura 4.5: ColorMap e BuildingsMap di un corridoio



(a)

(b)

Figura 4.6: Scansioni attese in un campo RoboCup (a) e in un corridoio (b)

Il modello dell'errore è stato realizzaato sperimentelmente [20] ed è composto dalla mistura di:

- una *gaussiana* che modella la densità attorno alla regine di massima probabilità;
- una *variabile di Erlng* che modella l'errore dovuto al rumore dell'immagine;
- un' *impulso di Dirac* che modella la possibilità di vedere una transizione all'infinito.

Con equazione (vedi grafico 4.7):

$$p(o_i|l) = \zeta_e(\frac{\beta^n o_i^{n-1} e^{-\beta o_i} 1(o_i)}{(n-1)!}) + \zeta_g \frac{1}{2\pi\sigma} e^{\frac{-(o_i - g(l,\alpha_i))^2}{\sqrt{2\pi\sigma^2}}} + \zeta_d \delta(o_i - \infty) \quad (4.1)$$

con δ delta di Dirac. $\zeta_e, \zeta_g, \zeta_d$ sono i coefficienti della mistura delle tre variabili aleatorie, ovviamnte $\zeta_e + \zeta_g + \zeta_d = 1$. Si noti che la media della gaussiana è, data posizione e angolo di scansione, la *distanza prevista*.

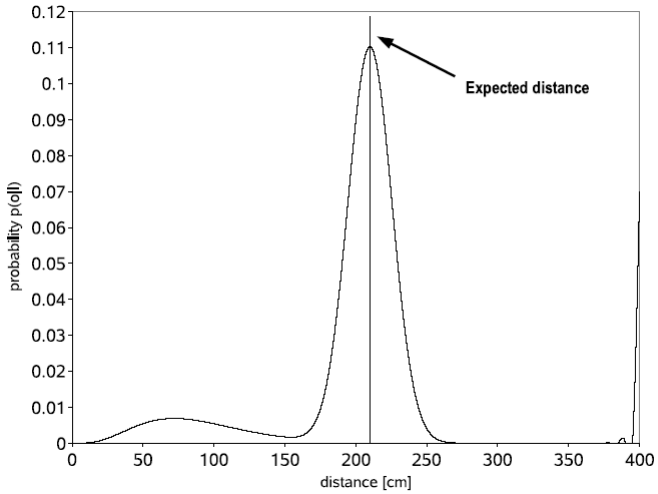


Figura 4.7: Il modello dell'errore

Una volta calcolata $p(o|g_k(l_j, a_i))$ per ogni direzione di scansione sarà possibile calcolare la probabilità globale di una transizione cromatica $p_k(o|l_j)$ semplicemente utilizzando la formula:

$$w_k^j = p_k(o|l_j) = \prod_i p(o|g_k(l_j, a_i)) \quad (4.2)$$

A questo punto si mormalizzano tutti i pesi in modo che $\sum_j w_k^j = 1$:

$$\tilde{w_k^j} = \frac{w_k^j}{\sum_j w_k^j} \quad (4.3)$$

Infine basta calcolare:

$$w^j = \prod_k w_k^j \quad (4.4)$$

4.3.1 Problemi riscontrati e modifiche apportate

La prima modifica apportata è stata una riorganizzazione della gestione delle mappe che ha permesso una facile implementazione dell'algoritmo anche in ambienti diversi da quelli RoboCup. In particolare il software è stato organizzato creando una classe *EnviromentMap* capace di gestire file in formato ppm e di disegnare su di essi alcune figure utili per la localizzazione: cerchi, frecette, robot, punti, linee. Da questa classe sono state derivate *ColorsEnvMap BuildingsEnvMap* con il compito di estrarre dai file ppm le informazioni cromatiche e quelle relative agli ostacoli fissi dell'ambiente come porte, paletti, mobili. Da queste si sono poi derivate delle classi specifiche per ogni ambiente. Da *ColorsEnvMap* sono state derivate *RobocupColorsEnvMap*, *LabColorsEnvMap* e *CorridoioColorsEnvMap* per disegnare la mappa cromatica dell'ambiente RoboCup, dell'ambiente del laboratorio e del corridoio in cui si sono svolti gli esperimanti. Da *BuildingsEnvMap* sono state derivate *RobocupBuildingsEnvMap*, *LabBuildingsEnvMap* e *CorridoioBuildingsEnvMap* per disegnare la mappa delle occupazioni dell'ambiente RoboCup, dell'ambiente del laboratorio e del corridoio in cui si sono svolti gli esperimanti. Questa struttura permette di poter essere facilmante espansa per gestire nuovi tipi di ambienti estandendo le classi *ColorsEnvMap BuildingsEnvMap* o per gestire nuovi tipi di informazioni ambientali, come luminosità, estendendo *EnviromentMap*.

Un problema correlato a quello delle mappe ambientale e quello delle lookup-table che modellano il sensore ideale. Al fine di non dover calcolare per ogni ipotesi di localizzazione le transizione reali attese si sono costruite delle tabelle contenenti le distanze delle transizioni per ogni possibile ipotesi di localizzazione. Al crescere della risoluzione delle mappe e quindi al crescere della precisione di localizzazione si ottiene anche un veloce aumento delle dimensioni di tali tabelle. Per risolvere questo problema si può pensare di ridurre le tabelle in base alle simmetrie ambientali. Per esempio in un campo RoboCup ci sono due assi di simmetria per le transizioni verde-bianco che permettono di ridurre le dimensioni della tabella ad un quarto di quella iniziale. Le que transizioni con il giallo e con il blu invece sono simmetriche e basta memorizzarne una sola per avere una descrizione completa del sensore.

Un'altra modifica molto importante è stata fatta all'algoritmo di ricampionamento (Algoritmo 2). Si è notato, infatti, che la convergenza alla posizione corretta era molto lenta nel caso in cui il robot stesse fermo. In questa condizione infatti il modello del movimento non riscontra alcun spostamento e le particelle, che si sono distribuite in modo casuale, non vengono disperse. Questo comporta che l'unica occasione per potersi localizzare correttamante in caso di localizzazione globale con robot fermo avviene quando un'ipotesi

di localizzazione cade proprio nel punto di localizzazione corretta. Questo problema è stato risolto spostando i campioni non nello stesso punto dei campioni con peso maggiore ma bistribuendoli con una distribuzione gaussiana casuale a media nulla in un intorno di 15 cm dei campioni con peso maggire. In questo modo basta che un'ipotesi di localizzazione cada nelle vicinanze della posizione corretta per poter convergere in pochi passi.

Un altro problema da affrontare e stato quello delle tempistiche di esecuzione dell'algoritmo di Monte Carlo. Le caratteristiche da soddisfare sono quelle di real-time ma nonostante la già gran quantità di accorgimenti adottati come l'implementagione di lookup-table e il precalcolo di molte funzioni ci si è resi conto che i tempi erano ancora molto elevati. Applicando tale algoritmo a tutte le immagini acquisite si forzava infatti la visione ad elaborare solamente 7 o 8 immagini al secondo. In un campo RoboCup, invece, è necessario cercare la palla su tutti i frame disponibili che equvale ad elaborare 25 immagini al secondo. Queste esigenze sono dettate dalla forte dinamicità dell'ambiente Robocup (alcune squadre sono capaci di calciare la palla a $8m/s$). Al fine di riportare la visione alla velocità di $25fps$ si è deciso spostare l'algoritmo di localizzazione in un thread separato. La soluzione si è dimostrata efficace infatti la visione è salita alla velocità di 23-24 fps (su di essa pesa l'algoritmo OVRF) e la localizzazione riesce a sfruttare ben 10 immagini al secondo con 2000 ipotesi di localizzazione.

Capitolo 5

Sperimentazioni in ambiente non Robocup

Gli algoritmi di localizzazione di Monte Carlo basati su sonar e laser per robot mobili sono stati inizialmente sviluppati per ambienti museali. Con il lavoro [20] è stato usato il sensore OVRF al posto dei sensori range finder comuni per la localizzazione in ambiente RoboCup. Proveremo ora ad utilizzare tale strategia di localizzazione con il sensore OVRF anche in un ambiente non RoboCup.

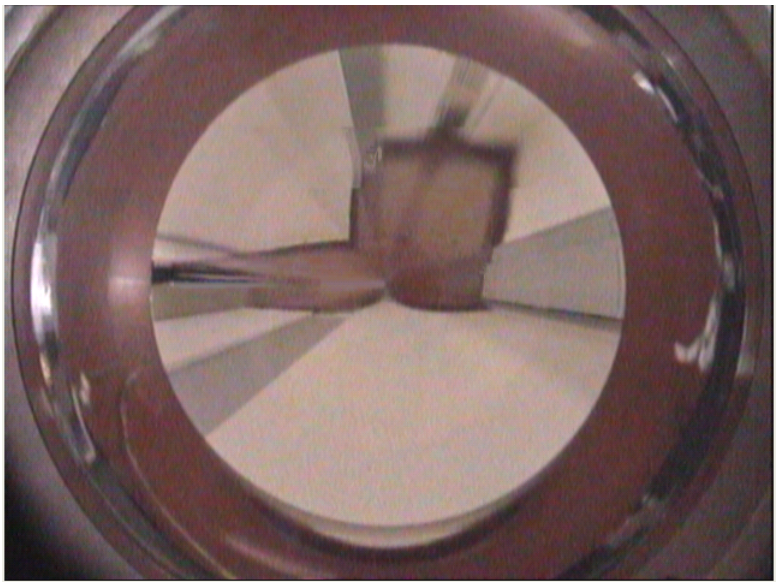


Figura 5.1: OVRF: riconoscimento delle transizioni rosso-bianco e rosso-grigio indicate rispettivamante con crocette rosse e gialle

L'ambiente per le sperimentazioni doveva soddisfare dei requisiti basilari:

doveva avere un pavimento e le pareti di colore uniforme e diversi tra loro e non doveva avere grosse variazioni di luminosità al suo interno. L'ambiente che sembrava avvicinari maggiormente a tali rieste si è dimostrato essere un corridoio del primo piano del DEI di via Ognissanti. In questo ambiente il pavimento è rosso, i muri sono bianchi e i mobili grigi (fig. 5.1).

Per prima cosa si è adattato il sensore OVRF in modo da farlo funzionare con le nuove transizioni cromatiche: rosso-bianco, rosso-grigio. Nella figura 5.2 si vede il sensore in funzione sull'immagine 5.1.

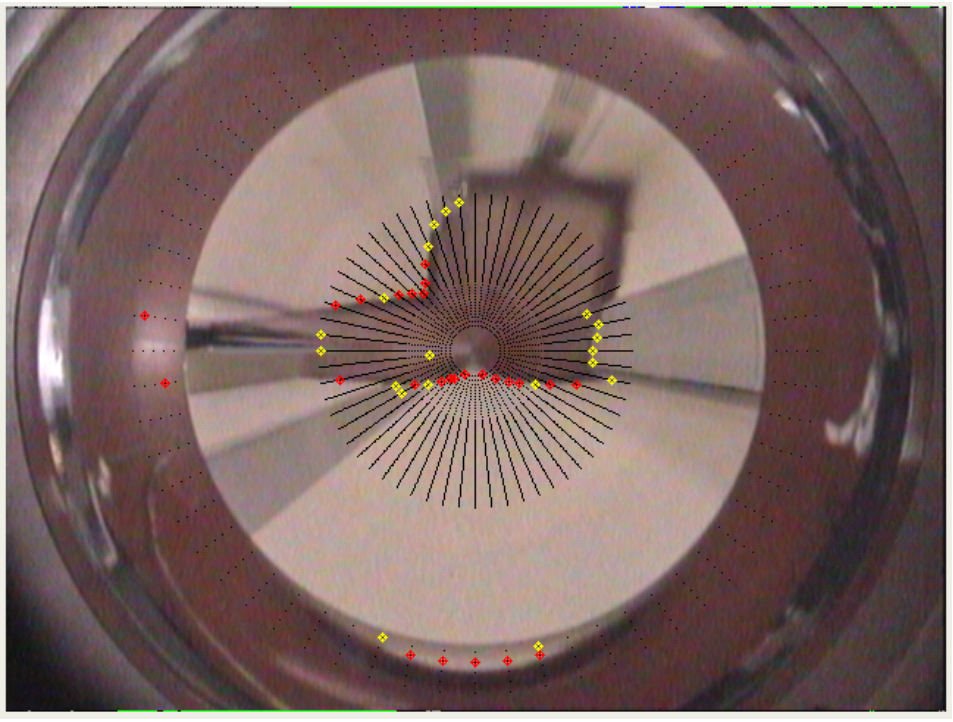


Figura 5.2: immagine omnidirezionale dell'ambiente in cui si sono svolti gli esperimenti

Come si può notare l'immagine 5.2 è affetta da notevole rumore, ci sono infatti numerose crocette gialle sulle transizioni rosso-binco e sul pavimento. Questo effetto è dovuto alla notevole disomogeneità luminosa all'interno del corridoio che mette in ombra il tratto di congiungimento tra pavimento e muro ed illumina ecessivamente la zona centrale del corridoio rendendolo riflettente.

Presentiamo di seguito (figura 5.3) una sequenza di due figure in cui mostriamo gli scan attesi per la posizione di figura 5.1 e gli scan reali relativi alla transizione rosso-bianco trovati da OVRF in figura 5.2. Come si può vedere gli scan reali differiscono da quelli attesi. Gli errori che vengono più

spesso commessi da OVRF sono la non identificazione delle transizioni vicine al robot a causa dell'ombra del robot e il non riconoscimento delle transizioni lontane a causa della deformazione dello specchio che comprime lo spazio lontano dal robot in pochi pixel rendendo molte volte difficile l'identificazione delle transizioni. Come si può vedere sono anche comparsi alcuni scan nella direzione della porta a causa della erronea identificazione dei colori.

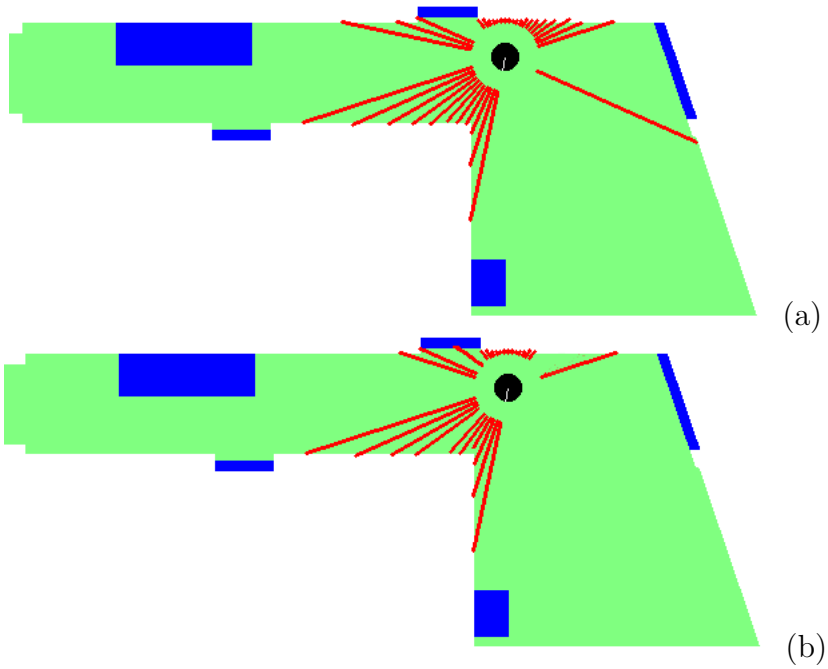


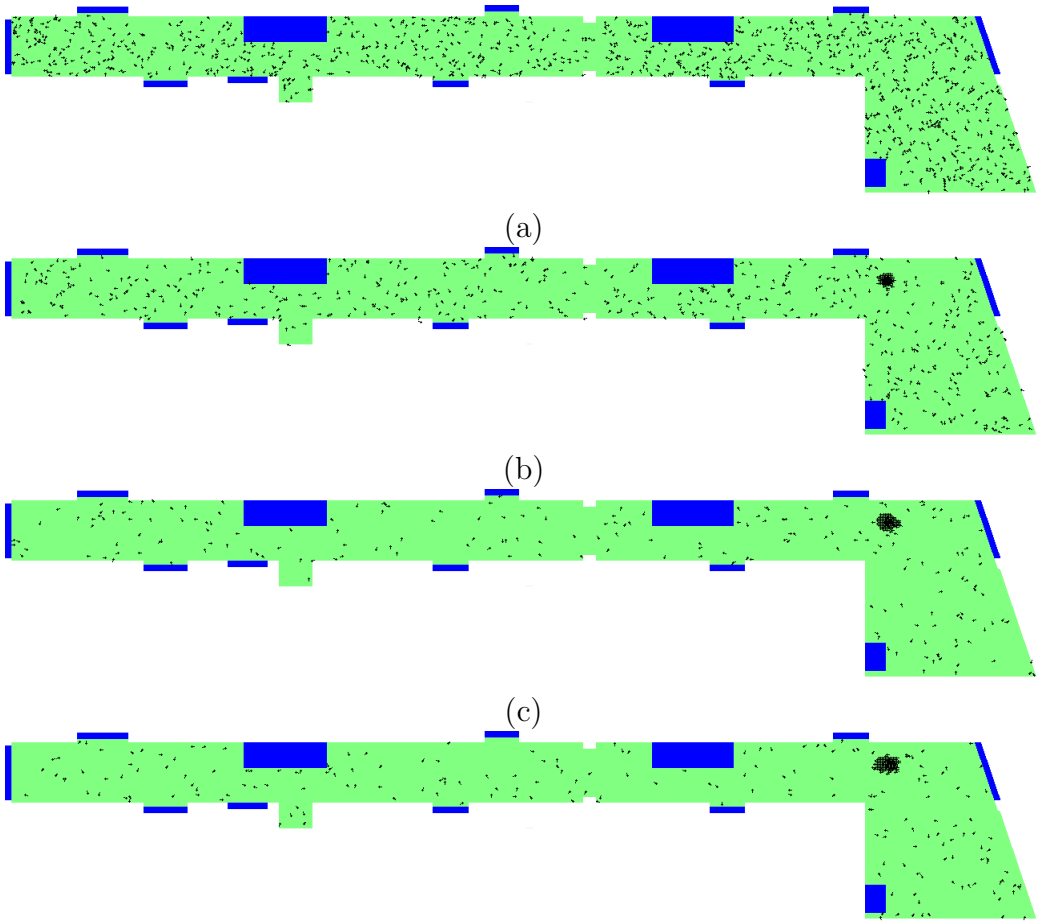
Figura 5.3: a partire dall'immagine 5.2 si sono ricavati (a) scan attesi per quella posizione, (b) scan reali ricavati dall'immagine

Da gli scan reali ricavati da OVRF è poi possibile disegnare la mappa della probabilità dell'ambiente (fig. 5.4). In essa le zone più scure rappresentano le zone in cui la probabilità di osservare gli scan è più elevata. Come si può notare oltre alla zona in cui si trova effettivamante il robot anche le altre zone con un profilo del muro simile hanno una discreta probabilità. In questo caso tutte le zone con un angolo potrebbero assomigliare a quella dove è stato preso lo scan.

Successivamente è stato svolto anche un test di convergenza dell'algoritmo. Il robot è stato posizionato in un punto del corridoio e si è eseguito l'algoritmo di localizzazione. Con il robot fermo la localizzazione converge in poche immagini al punto corretto. Nelle figure 5.5 si vede l'evolversi delle posizioni delle ipotesi di localizzazione.

Nella prima immagine 5.5a si vedono le ipotesi di localizzazione uniforme-

Figura 5.4:



(a)

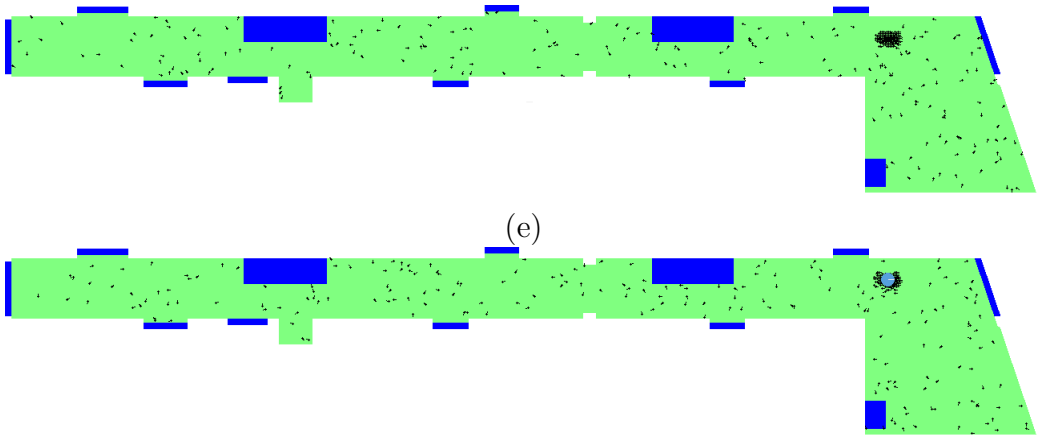
(b)

(c)

(d)

Figura 5.5: Evoluzione temporale delle ipotesi di localizzazione. (Continua...)

mente distribuite in tutto lo spazio del corridoio. Nelle figure sucessive le particelle tendono ad addensarsi attorno alla posizione più probabile. Infine, 5.5f, quando la dispersione delle particella risulta bassa e si ha quindi una buona stima della localizzazione il robot viene localizzato nel punto corrispondente alla posizione media delle particelle.



(e)

(f)

Nell'esperimento successivo abbiamo testato la velocità di convergenza dell'algoritmo in dipendenza al numero di campioni usati in varie posizioni del corridoio. Come si nota dal grafico 5.6 al crescere del numero di ipotesi di localizzazione i tempi di convergenza diminuiscono infatti aumenta la probabilità di posizionare un campione in punto vicino alla posizione reale in cui si trova il robot. L'errore rimane comunque più alto rispetto agli esperimanti svolti in campo RoboCup [18] a causa della rumorosità luminosa dell'ambiente.

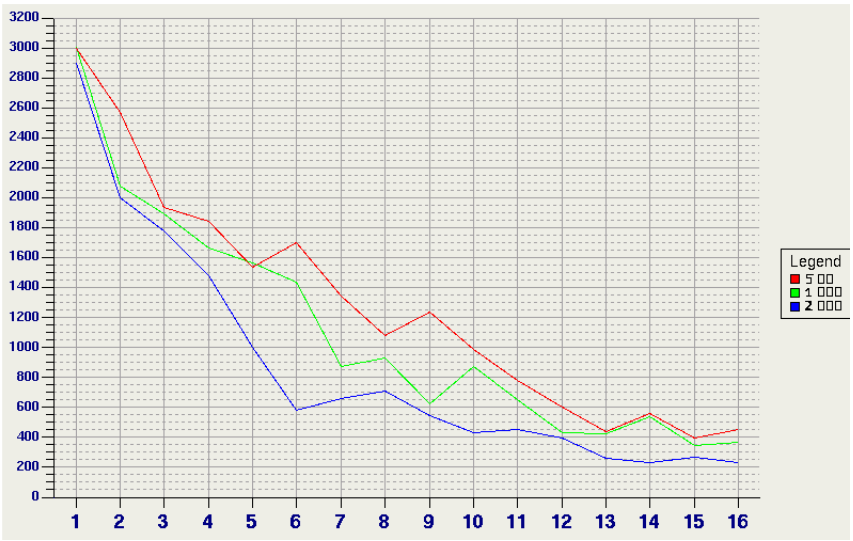


Figura 5.6: Tempi di convergenza: viene correlato l'errore di localizzazione al numero di immagini analizzate in funzione del numero di ipotesi di localizzazione

Il grosso errore iniziale è dovuto alla grande dimensione dell'ambiente. All'inizio infatti i campioni vengono distribuiti in modo casuale e la posizione media si trova al centro del corridoio mentre la posizione reale si trova quasi in un angolo.

Capitolo 6

Conclusioni

In questa tesina è stata fatta una panoramica dei principali algoritmi di Monte Carlo per la localizzazione di robot mobili. Successivamente è stato analizzato il nuovo sensore range finder OVRF (sez. 4.1) sviluppato da Alberto Pretto per la squadra di RoboCup Artisti Veneti, ed è stato visto come tale sensore possa essere utilizzato con gli algoritmi di localizzazione di Monte Carlo. In questa analisi si sono messi in evidenza i difetti e i problemi riscontrati nei test in laboratorio e nelle partite svolte durante l'anno. Per ogni problema si è presentata anche la soluzione adottata tra le possibili applicabili.

Si sono presentati anche gli esperimenti svolti in ambiente non RoboCup. La strategia di localizzazione basata su algoritmi di Monte Carlo applicati al sensore OVRF era sempre stata sperimentata solo in ambiente RoboCup. In questo modo è stato possibile provare che tale metodo funziona anche in ambienti generici. La più grossa limitazione riscontrata risiede nella forte sensibilità alle variazioni luminose del sensore OVRF.

6.1 Sviluppi futuri

Per prima cosa bisognerebbe continuare le sperimentazioni in ambiente non RoboCup testando l'algoritmo anche in presenza di ostacoli.

Il problema principale riscontrato è stato la forte sensibilità del sensore OVRF alle variazioni luminose che rendono imprecise le sue misurazioni. Un notevole incremento della robustezza dell'algoritmo si potrebbe ottenere adottando un sistema di quantizzazione dei colori più sofisticato capace di essere il più possibile insensibile alla luce.

Appendice A

Richiami di Teoria della Probabilità

A.1 Regola di Bayes

Regola di Bayes
Dati due eventi di probabilità positiva B e C vale:

$$P[C|B] = \frac{P[B|C]\, P[C]}{P[B]} \tag{A.1}$$

Se gli eventi A_i, $i = 1, 2, \ldots$ formano una partizione dell'evento certo:

$$P[A_i|B] = \frac{P[B|A_i]\, P[A_i]}{\sum_j P[B|A_j]P[A_j]} \tag{A.2}$$

Nel caso di più eventi condizione:

$$P[C|B_1, B_2] = \frac{P[B_2|C, B_1]\, P[C|B_1]}{P[B_2|B_1]} \tag{A.3}$$

Analoghe equazioni valgono per le densità di probabilità di variabili aleatorie condizionate.

A.2 Regola marginale

Sia $\mathbf{x} = [x_1, \ldots, x_n]$ un vettore aleatorio di densità di probabilità $f_{\mathbf{x}}(\mathbf{a})$. Se si scrive $\mathbf{x} = [\widetilde{\mathbf{x}}, \widehat{\mathbf{x}}]$, dividendo le componenti in due sottovettori, allora la

densità del vettore $\widetilde{\mathbf{x}}$ si ottiene integrando la densità $f_{\mathbf{x}}(\mathbf{a})$ rispetto a tutte le componenti che non compaiono in $\widetilde{\mathbf{x}}$ quindi:

$$f_{\widetilde{\mathbf{x}}}(\widetilde{\mathbf{a}}) = \int f_{\mathbf{x}}(\mathbf{a}) d\,\widehat{a} \tag{A.4}$$

ad esempio se $\mathbf{x} = [x_1, x_2, x_3]$ allora per la v.a. x_1 si ha:

$$f_{x_1}(a_1) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{\mathbf{x}}(a_1, a_2, a_3) da_2 da_3 \tag{A.5}$$

A.3 Probabilità condizionata

Definizione di probabilità condizionata
Sia dato un evento C possibile, cioè di probabilità strettamente positiva. Per ogni evento A si definisce la *probabilità condizionata* di A rispetto all'*evento condizione* C come:

$$P[A|C] \doteq \frac{P[A, C]}{P[C]} \tag{A.6}$$

Proprietà notevole
La probabilità congiunta di due eventi A e B condizionati da C si scrive:

$$P[A, B|C] = P[A|B, C]\; P[B|C] \tag{A.7}$$

A.4 Probabilità Totale

Teorema della Probabilità Totale
Se gli eventi A_i, $i = 1, 2, \ldots$ formano una partizione dell'evento C (sono cioè tra loro *disgiunti*) e se l'evento $B \subseteq C$ allora:

$$P[B] = \sum_i P[B, A_i] = \sum_i P[B|A_i]\; P[A_i] \tag{A.8}$$

Per variabili aleatorie x, y continue di densità di probabilità f_x, f_y e densità di probabilità condizionata $f_{y|x}$, si scrive:

$$f_y(b) = \int_{-\infty}^{\infty} f_{y|x}(b|a) f_x(a) da \tag{A.9}$$

Appendice B

Robot utilizzato nelle sperimentazioni

Tutte le sperimentazioni presentate in questa tesi si sono svolte all'interno del laboratorio IAS-Lab (*Intelligent Autonomous Systems Laboratory*) del Dipartimento di Ingegneria dell'Informazione di Padova. Il laboratorio è anche la sede del team RoboCup *Artisti Veneti* e dispone di una squadra completa di robot che partecipa ai campionati mondiali che si svolgono ogni anno. All'interno vi è ricostruito anche un campo di gioco di 8 metri per 4. I robot qui utilizzati sono stati "Fred" e "Barney" (Figura B.1): essi sono piattaforme mobili *olonome*, ovvero hanno la possibilità di traslare in qualsiasi direzione senza dover prima ruotare e disporsi lungo tale direzione. Tale movimentazione è realizzata utilizzando 3 ruote di tipo *svedese*: dosando opportunamente la velocità su ognuno dei tre assi, è possibile prendere istantaneamente qualsiasi direzione. Il sistema di visione è di tipo omnidirezionale con specchio multizona (B.2). Il telaio è costruito in alluminio e alla base è formato da tre cilindri che vengono utilizzati come bombole di aria compressa per alimentare il kicker pneumatico utilizzato durante le competizioni.

Queste le caratteristiche hardware:

- **Movimento**
  - 3 ruote omnidirezionali di tipo *svedese* di 8 cm di diametro
  - 3 motori elettrici da 20 W alimentati a 24 V
  - Scheda di controllo dei motori autocostruita, sia la parte logica che quella di potenza
- **Sensori**

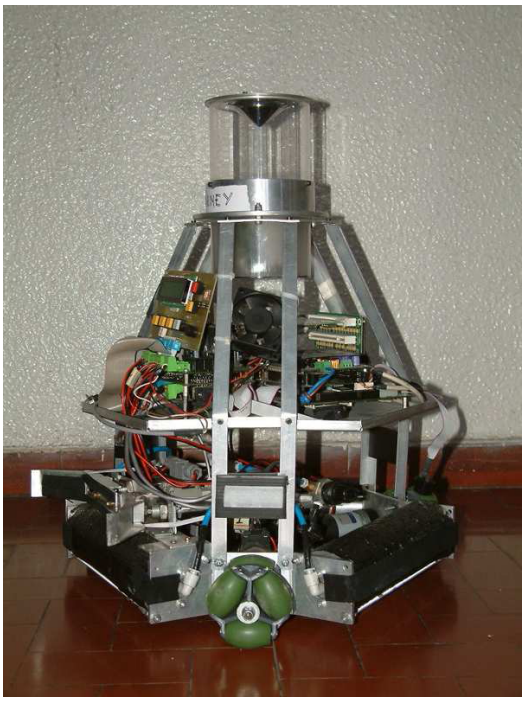


Figura B.1: Il robot “Barney”: si notino la particolare forma delle ruote ed il visore omnidirezionale

  - Telecamera analogica Hitachi
  - Specchio prospettico omnidirezionale
  - Frame grabber in standard PC 104

- **Calcolo**

  - Scheda madre in standard PC 104 con CPU Pentium 3, 700 MHz, 256 MB memoria ram, scheda Lan incorporata
  - Compact flash da 256 MB come memoria di massa
  - Adattatrice per PCMCIA in standard PC 104 a cui è collegata una scheda WaveLan 802.11b

Il sistema operativo utilizzato dal robot è una versione ottimizzata di Gentoo Linux basata sul Linux Kernel 2.4.20. Tutto il s.o. occupa poco più di 50 MB.

Figura B.2: Lo specchio omnidirezinale: la zona esterna serve per vedere le zone in prossimità del robot

I robot Barney e Fred sono stati costruiti nel 2000 dal Golem RoboCup Team, una squadra di studenti dell'università di Padova che autonomamente ha partecipato arrivando seconda ai campionati mondiali RoboCup di quell'anno. Dal 2002 tali robot appartengono allo IAS-Lab.

Bibliografia

[1] *Bayesian Statistics 3*, chapter Using the SIR algorithm to simulate posterior distributions. Oxford University Press, 1988.

[2] C. Andrieu A. Doucet, S. Godsill. On seqeuntial monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208.

[3] N. Gordon A. Doucet, N. de Freitas. *Sequential Monte Carlo Methods in Pratice*, chapter 19, pages 401–428. Springer, New Yorg, 2001.

[4] G. Pierobon C. M. Monti. *Teoria della Probabilità*. Decibel, Padova, 2000.

[5] A. Doucet D. Crisan. Convergenge of sequential monte carlo methods. Technical report, Cambridge University Engineering Department, CUED/F-INFENG/TR381, 2000.

[6] F. Delleart D. Fox, W. Burgard. Monte carlo localization: Efficient position estimation for mobile robots. AAAI-99, Orlando, FL, 1999.

[7] A. Doucet. On seqeuntial monte carlo methods for bayesian filtering. Technical report, University of Cambridge (UK), Department of Engineering, 1998.

[8] E. Pagello E. Menegatti, M. Zoccarato and H. Ishiguro. Hierarchical image-based localisation for mobile robots with monte-carlo localisation. In *1st European Conference on Mobile Robots ECMR'03*, 2003.

[9] E. Pagello E. Menegatti, M. Zoccarato and H. Ishiguro. Image-based monte-carlo localisation without a map. In *8th Conference of the Italian Association for Artificial Intelligence (AI*IA)*, pages 423–435, 2003.

[10] E. Pagello E. Menegatti, M. Zoccarato and H. Ishiguro. Image-based monte-carlo localisation with omnidirectional images robotics and autonomous systems. In *Elsevier*, volume 48, pages 17–30. 2004.

[11] E. Menegatti E. Pagello, A. D'Angelo. Artisti veneti 2002: evolving an heterogeneous robot team for the middle-size league. In *Team Descritpion*. 2002.

[12] D. Fox F. Delleart, W. Burgard. Using the condensation algorithm for robust, vision-based mobile robot localization. International Conference on Computer Vision and Pattern Recognition (CVPR), 1999.

[13] W. Burgard F. Delleart, D. Fox. Monte carlo localization for mobile robots. IEEE International Conference on Robotics and Automation (ICRA-99), Detroit, MI, 1999.

[14] L. Feng J. Borenstein, B. Everett. *Navigating Mobile Robots: System and Techniques*. A.K. Peters, Wellesley, 1996.

[15] G. Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal Of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[16] Lajoie Lippman. *C++ Primer 3rd ed.* Addison-Wesley.

[17] H. Kitano M. Veloso, E. Pagello. Lecture notes in computer science 1856. In *RoboCup-99: Robot Soccer World Cup III,*. Springer, 2000.

[18] Pagello Menegatti, Pretto. A new omnidirectional vision sensor fof monte-carlo localization.

[19] A. F. M. Smith N. Gordon, D. Salmond. Novel approach to nonlinear and non-gaussian bayesian state estimation. *IEEE Proceedings-F*, 140:107–113, 1993.

[20] A. Pretto. Localizzazione di monte carlo in ambiente robocup. Master's thesis, Università di Padova, 2002/2003.

[21] D. McDermot S. Engelson. *Error correction in mobile robot map lerning*. 1992 IEEE International Conference on Robotics and Automation, Nice, France, 1992.

[22] P. Norving S. Russel. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.

[23] W. Burgard S. Thrun, D. Fox. Robust monte carlo localization for mobile robots. In *Artificial Intelligence (AI)*, volume 128, pages 99–141. 2001.

[24] Matthias Jüngel Thomas Röfer. Vision-based fast and reactive monte-carlo localization. In *IEEE International Conference on Robotics and Automation (ICRA-2003), Taipei, Taiwan*, pages 856–861, 2003.

[25] D. Fox W. Burgard, A. Derr. *Integrating global position estimation and position tracking for mobile robots: The dynamic markov localization approach*. IEEE/RSJ International Conference on Intelligent robots an d Systems (IROS 98), Victoria, BC, 1998.

[26] M. Zoccarato. Localizzazione di monte carlo per robot mobili dotati di visione omnidirezionale. Master's thesis, Università di Padova, 2003.