UNIVERSITA' DEGLI STUDI DI ROMA
"TOR VERGATA"

FACOLTA' DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA DELL'AUTOMAZIONE

TESI DI LAUREA

Tecniche di mappatura per la navigazione di robot mobili

RELATORE
Prof. Francesco Martinelli

CANDIDATO
MarcoMazzeo

A.A.2007/2008

INDICE

INTRODUZIONE

I sistemi robotici sono intercalati nell' ambiente che ci circonda e hanno il compito di ricevere informazioni sull'ambiente esterno attraverso i loro sensori e di modificarlo secondo i propri scopi attraverso gli attuatori. Sempre più spesso i robot, per compiere appieno le proprie funzioni, hanno necessità di conoscere l'ambiente in cui operano e ridurre al minimo la propria incertezza su di esso. E' proprio negli ultimi due decenni che si sono raggiunti i migliori risultati in questa direzione grazie anche all'utilizzo di tecniche probabilistiche. Facendo ciò tutte le ambiguità e le incertezze potranno essere affrontate da un punto di vista matematico molto più facile e immediato da analizzare. Il risultato che si otterrà è quello quindi di ottenere un controllo più robusto relativamente alle incertezze che rimangono. Ci prefissiamo l'obbiettivo di descrivere l'ambiente circostante basandoci su informazioni affette da errori; Basti pensare che i sensori danno un informazione parziale sull'ambiente esterno in quanto il loro raggio di azione è limitato da alcuni aspetti fondamentali :

- il range di risoluzione è limitato in quanto il sensore non può vedere attraverso i muri o oltre range massimo ;
- le misurazioni sono spesso soggette al rumore che le alterano in maniera imprevedibile;

Sfruttando l'approccio probabilistico vogliamo far si che il robot mobile, riesca a interpretare e adattarsi all'ambiente in cui si trova. . Acquisire una mappa con un robot mobile è un problema interessante in quanto sappiamo che le mappe sono definite su uno spazio continuo e che tale spazio ha infinite dimensioni. Soltanto utilizzando tecniche di discretizzazione, come ad esempio la griglia occupazionale, è possibile descrivere la mappa. Focalizzeremo la nostra attenzione sulla tecnica di mappatura con griglie occupazionali noti le misurazioni e la posizione

esatta del robot. La difficoltà dei problemi di mappatura è data da un insieme di fattori quali ad esempio la grandezza dell'ambiente e i disturbi dei sensori. L'idea di base è quella di suddividere la mappa in un campo di variabili ognuna delle quali rappresenta la probabilità della singola cella di essere occupata. Supposto lo spazio di stato statico utilizzeremo il filtro di Bayes binario per stimare le probabilità delle singole celle. Assegnata ad ogni cella la propria probabilità si passerà quindi alla ricostruzione della mappa. Vedremo come le grandezza delle celle, le posizioni assunte, la grandezza dell'ambiente e i rumori dei sensori influenzino la velocità, la risoluzione e l'accuratezza del algoritmo di mappatura. Verrà inoltre illustrato un esempio di mappatura di un ambiente da parte di un robot mobile.

CAPITOLO 2

FILTRO DI BAYES

Come già detto in precedenza praticamente tutti gli algoritmi più attuali sono di tipo probabilistico. La ragione per l'ampio utilizzo di tecniche probabilistiche nasce dal fatto che il problema della mappatura di un ambiente da parte di un robot mobile è caratterizzato dall'incertezza e dal rumore nelle letture dei sensori. Questo tipo di rumore percettivo è complesso e non facile da modellare. Gli algoritmi probabilistici affrontano il problema modellando esplicitamente le differenti sorgenti di rumore e i loro effetti sulle misure. Nel corso del tempo questi algoritmi si sono dimostrati gli unici in grado di affrontare efficacemente questo problema. Il principio di base sottostante a qualsiasi algoritmo di *mapping* è la regola di Bayes:

$$p(x_i \mid z) = \frac{p(z \mid x_i)\, p(x_i)}{\sum_i p(z \mid xi)\, p(x_i)}$$

Dove x_i rappresenta la cella i-esima e z la misurazione. La regola di Bayes è l'archetipo di tutti i ragionamenti di tipo probabilistico. Si supponga che vogliamo imparare qualcosa su una quantità *x* (ad esempio una mappa) sulla base di dati di misura *z* (ad esempio basandoci su misure di distanza e misure di posizione). La regola di Bayes allora ci dice che il problema può essere risolto moltiplicando due termini: *p(z|x)* e *p(x)*. il termine *p(z|x)* specifica la probabilità di osservare la misura z sotto l'ipotesi X. *p(z|x)* quindi è un modello generativo, nel senso che descrive il processo della generazione di misura del sensore a fronte di differenti stati X. Il termine *p(x)* viene chiamato probabilità a priori: specifica cioè la probabilità ad assumere che X sia la scelta giusta prima dell'arrivo di

qualsiasi misurazione. Il termine *p(x|z)* rappresenta allora la probabilità a posteriori, ovvero la probabilità che x sia la scelta giusta sulla base di tutte le misurazioni z fatte fino a quel momento. Nel problema del mapping robotico i dati arrivano progressivamente nel corso del tempo, pertanto solitamente si utilizza una formulazione ricorsiva della regola di Bayes che viene chiamata Filtro di Bayes, in questa formulazione la probabilità a posteriori viene progressivamente affinata man mano che arrivano le misurazioni. Il filtro di Bayes è alla base dei Filtri di Kalman e ai modelli di Markov (Hidden Markov Model). Vediamo ora come il filtro di Bayes può essere applicato al problema della stima della posizione. Adottando una notazione simbolica possiamo scrivere:

$$bel(x_t) = p(x_t | z_{1:t}, u_{1:t})$$

In cui lo stato x_t rappresenta lo stato al temo t, $z_{1:t}$ e $u_{1:t}$ sono invece rispettivamente tutte le misurazioni effettuate e i controlli applicati fino a tempo t. La simbologia Bel(x_t) è un altro importante concetto nella robotica probabilistica, esso sta per *belief* e tale variabile riflette le informazioni sullo stato interno del robot. Possiamo quindi dire che Bel(x_t) è una notazione più compatta per rappresentare la probabilità a posteriori dello stato x al tempo t. Si può osservare dalla formula che tale variabile dipende da tutte le misurazioni e i controlli applicati, comprendendo anche le misure al tempo t (medesimo dello stato che si sta valutando). Il filtro di Bayes stima le probabilità in modo ricorsivo. Una probabilità iniziale caratterizza la conoscenza iniziale sullo stato del sistema. L'assenza di informazioni sullo stato iniziale viene modellata tipicamente con una distribuzione uniforme sullo spazio degli stati: quindi qualora trovassimo una distribuzione iniziale uniforme significa che la posizione del robot è completamente sconosciuta. Per derivare un'equazione di aggiornamento ricorsiva osserviamo che l'espressione precedente può essere trasformata per mezzo della regola di Bayes in

$$bel(x_t) = \eta p(z_t | x_t) p(x_t | u_{1:t}, z_{1:t-1})$$

Dove il fattore η è costante e pari:

$$\eta = p(z_t | u_{1:t}, z_{1:t-1})$$

Ricordando che il filtro di Bayes si fonda sulle assunzioni Markoviane ossia sul fatto che i dati futuri sono indipendenti da quelli passati nel momento in cui si conosca la posizione corrente, tale equazione assume una forma del tipo

$$bel(x_t) = \eta p(z_t | x_t) \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$

In cui il fattore integrativo assume un valore di predizione. Infatti come si può vedere facilmente dalla formula esso calcola una stima della posizione futura basandosi soltanto sulla posizione dello stato precedente e non dalla misurazione effettuata al tempo t. Questa equazione è l'equazione di aggiornamento ricorsiva del filtro di Bayes. Insieme con lo stato iniziale *Bel(0),* definisce una stima ricorsiva dello stato di un sistema parzialmente osservabile. Per implementare tale equazione è necessario conoscere due densità condizionate: la probabilità $p(x_t | x_{t-1}, u_t)$ che viene chiamata modello del moto e la densità $p(z_t | x_t)$ che viene chiamata modello del sensore. Entrambi i modelli sono solitamente stazionari , non dipendono quindi in alcun modo dal tempo t. Introdotte queste specifiche possiamo formulare l'algoritmo per il filtro di Bayes il quale:

FILTRO DI BAYES

$\forall x_t$

$$\overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$

$$bel(x_t) = \eta p(z_t | x_t) \overline{bel}(x_t)$$

Questo algoritmo non può essere implementato direttamente in quanto l'insieme di tutte le mappe e posizioni del robot è una distribuzione di probabilità definita su uno spazio continuo che quindi possiede un numero infinito di dimensioni. Nel nostro caso si è scelto di utilizzare una particolare forma del filtro di Bayes il quale sotto specifiche assunzioni riesce a fornire una stima più che ottimale dello stato richiesto. Senza perdita di generalità possiamo considerare lo spazio di stato finito: dove ogni x_t , rappresentante le possibili posizioni, possiede una variabile binaria che ne indica lo stato. Nel considerare algoritmi di mappatura si preferisce suddivide l'ambiente in celle di occupazione dove ogni cella possiede un valore binario che ne esprime l'occupazione: 1 nel caso di cella occupata 0 se libera. E' chiaro che nell'approssimare lo stato al valore tra 1 e 0 si introducono delle discontinuità dovuti alle approssimazioni da compiere. Per superare questo limite è possibile esprimere tale valore utilizzando una forma più elegante. Infatti, abbiamo detto che lo stato può assumere un valore x o il suo negato –x; ne risulta che la probabilità è definita come il rapporto tra

$$\frac{p(x)}{p(-x)} = \frac{p(x)}{1-p(x)}$$

Indichiamo con l(x) il logaritmo del rapporto della probabilità (*log odds ratio*) otteniamo :

$$l(x) = \log \frac{p(x)}{1-p(x)}$$

Si può osservare quindi che tale funzione è limitata tra valori 0 1 ma può variare da $-\infty$ a ∞ . Sfruttando questa forma possiamo risalire alla stima dello stato semplicemente sapendo l_t infatti

$$bel(x_t) = 1 - \frac{1}{1+e^{l_t}}$$

Per cui possiamo ricondurre,utilizzando tale notazione, l'agoritmo di Bayes in:

$$l_t = l_{t-1} + \log\left(\frac{p(x \mid z_t)}{1 - p(x \mid z_t)}\right) - \log\left(\frac{p(x)}{1 - p(x)}\right)$$

Tale forma prende il nome di algoritmo binario di Bayes il quale sarà di enorme importanza per lo studio della mappatura basato su griglie occupazionali. La maggiore differenza del filtro binario da quello di partenza è che tale algoritmo è additivo. In tale forma la probabilità sarà calcolata semplicemente incrementando o decrementando la variabile l_t.

CAPITOLO 3

MODELLO DEL ROBOT

Nell' applicare il filtro di Bayes abbiamo visto che per stimare la probabilità della posizione avevamo necessità di conoscere il modello del moto *(motion model)* $p(x_t| x_{t-1},u_t)$. Inoltre conoscere a fondo il modello del moto e dei sensori assume un' importanza fondamentale perché è proprio grazie a questi che il robot mobile riesce a eseguire il compito richiesto. Tanto più saranno accurati i modelli tanto più sarà facile per il robot portare a termine il proprio obbiettivo senza complicazioni. Vediamo ora come stimare al meglio tali modelli.

3.1 Modello di movimento del Robot

Qualsiasi configurazione del robot mobile nello spazio è descrivibile con sei variabili: 3 coordinate cartesiane e tre angoli (roll,picht,yaw), comunemente chiamati angoli di Eulero. Se restringiamo le nostre considerazioni ad un semplice piano bidimensionale sappiamo che le componenti che descrivono la posizione del robot saranno date dal vettore

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

In cui x,y sono le coordinate cartesiane e θ (l'angolo formato con l'asse delle ascisse) l'orientamento (Figura 3.1)

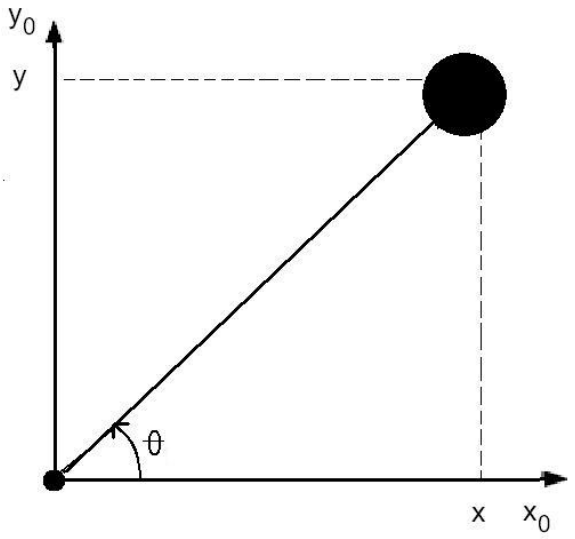

Figura 3.1

Definito $u = \begin{pmatrix} v \\ \omega \end{pmatrix}$ il generico controllo applicato al robot mobile in cui v , ω sono rispettivamente la velocità di traslazione e di rotazione è possibile ricostruire qualsiasi posizione dalla relazione

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v}{\omega}\sin(\theta) + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ \frac{v}{\omega}\cos(\theta) - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{pmatrix}$$

In cui Δt è il tempo di campionamento. Come è immaginabile tale modello non rispecchia fedelmente la realtà in quanto non considera affatto tutti quegli aspetti che influenzano e cambiano il moto. Tra questi i più evidenti sono :

- l'attrito delle ruote;
- l'impossibilità fisica da parte del robot di muoversi immediatamente alla velocità imposta e di mantenerla costante lungo tutto il suo percorso.

Nella realtà quindi la movimentazione del robot è soggetta a rumori così che la posizione finale x_t non sempre è quella desiderata. A seconda della complessità e del tempo Δt di campionamento potremmo trovarci più o meno vicini alla posizione ideale. Come si può osservare dalla figura 3.2 , dopo aver applicato un controllo u il robot può trovarsi in diverse posizioni indicate dal pallini blu:

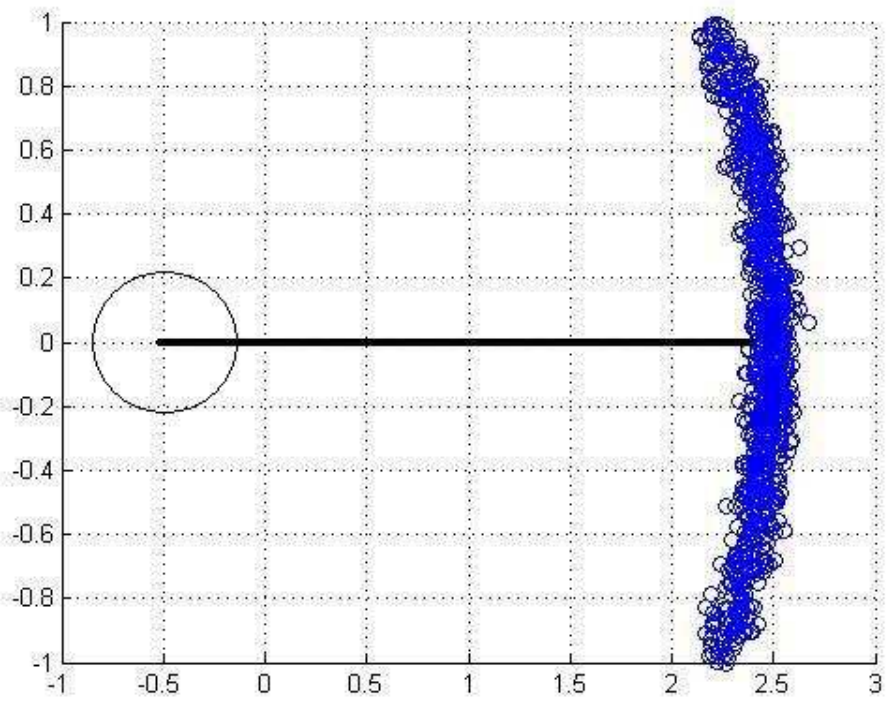

Figura3.2:Modello del moto

Si intuisce inoltre che le zone maggiormente marcate sono più probabili di altre. Per spiegare meglio ciò che accade possiamo definire il controllo come:

$$\hat{u} = \begin{pmatrix} \hat{v} \\ \hat{\omega} \end{pmatrix} + \begin{pmatrix} \varepsilon_{\alpha 1 v^2 + \alpha 2 \omega^2} \\ \varepsilon_{\alpha 3 v^2 + \alpha 4 \omega^2} \end{pmatrix}$$

Dove il fattore $\varepsilon_{\alpha i v^2 + \alpha \omega^2}$ rappresenta il disturbo. Realizzeremo tale rumore con una distribuzione normale gaussiana centrata in zero e avente come varianza b. Per cui la forma finale di $\varepsilon_{\alpha i v^2 + \alpha \omega^2}$ sarà:

$$\varepsilon_{\alpha_1 v^2+\alpha\omega^2}(a)=\frac{1}{\sqrt{2\pi b}}e^{-\frac{a^2}{b^2}}$$

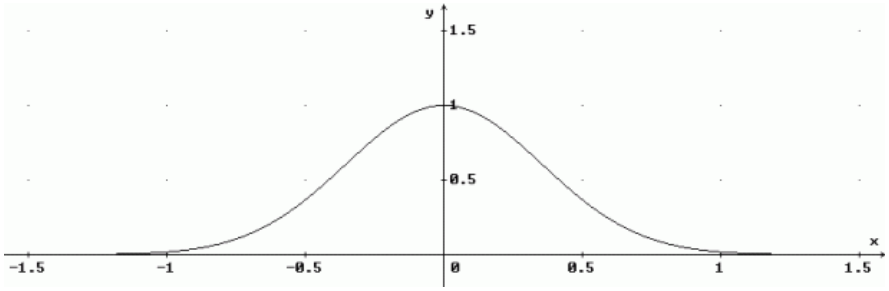

Figura 3.3 Distribuzione gaussiana

Gli α_i sono parametri specifici di errore del robot, più saranno grandi tali parametri minore sarà l'accuratezza del robot. Per quanto riguarda invece l'orientamento del robot alla fine del moto sappiamo che l'angolo θ' sarà pari:

$$\theta' = \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t$$

con $\hat{\gamma} = \alpha_5 v^2 + \alpha_6 \omega^2$

quindi la matrice prima riportata può essere ora riscritta come:

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{\hat{v}}{\hat{\omega}}\sin(\theta) + \frac{\hat{v}}{\omega}\sin(\theta + \hat{\omega}\Delta t) \\ \frac{\hat{v}}{\hat{\omega}}\cos(\theta) - \frac{\hat{v}}{\hat{\omega}}\cos(\theta + \hat{\omega}\Delta t) \\ \hat{\omega}\Delta t + \hat{\gamma}\Delta t \end{pmatrix}$$

Conoscendo tali parametri ora è possibile passare al calcolo $p(x_t | x_{t-1}, u_t)$ da utilizzare nel filtro di Bayes. Introducendo l'errore del moto definito come la differenza tra il controllo u e la rotazione γ ideali con il rispettivo controllo $\hat{u}$ e rotazione $\hat{\gamma}$ reali possiamo dire:

$$v_{err} = v - \hat{v}$$

$$\omega_{err} = \omega - \hat{\omega}$$

$$\gamma_{err} = \gamma - \hat{\gamma}$$

In cui γ può essere considerato uguale a zero senza perdita di generalità.

Le probabilità di tali errori, come abbiamo visto anche in precedenza vengono, modellate come una gaussiana con media zero e varianza $b = \alpha_i v + \alpha_j \omega$ dove α sono parametri di errore specifici del robot mobile.

Supponendo vi sia indipendenza tra gli errori possiamo calcolare il modello del robot come:

$$p(x_t \mid u_t, x_{t-1}) = \varepsilon_b(v_{err})\varepsilon_b(\omega_{err})\varepsilon_b(\gamma_{err})$$

In cui ε_b esprime la gaussiana

$$\varepsilon_b(a) = \frac{1}{\sqrt{2\pi b}} e^{-\frac{a^2}{b^2}}$$

3.2 MODELLO PERCETTIVO DEL ROBOT

Conoscere il modello percettivo del robot è tanto importante quanto conoscerne il modello del moto. E' proprio grazie ai sensori che il robot riesce a stimare la presenza di oggetti e a ricostruire l'ambiente intorno a sè. Oggi giorno ci sono un numero considerevole di sensori tra cui i più comuni sono quelli di contatto,telecamere o sensori di distanza. Tutti i tipi di sensori modellati nella realtà sono affetti da disturbi che ne modificano il funzionamento ideale. Tale fonte di errore può avere diverse origini:

- la presenza all'interno del nostro ambiente di persone possono causare delle letture non corrette;
- la riflessione del raggio su superfici irregolari altera la reale misurazione.

Quest'ultimo aspetto è il più comune fattore di errore. La misurazione per un dato sensore avviene tipicamente sparando un laser e registrando il suo eco riflesso (figura 3.1).

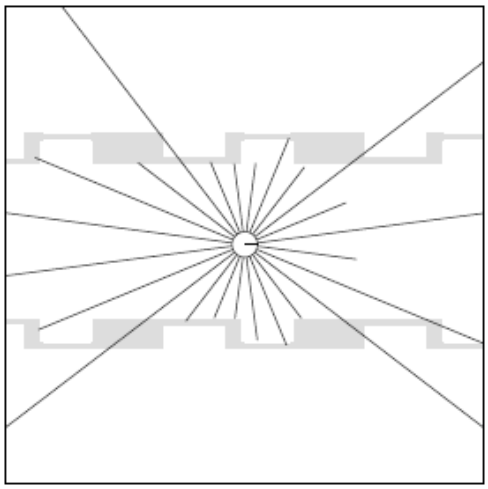

Figura 3.1 Modello di robot

Sappiamo inoltre che qualsiasi superficie ha la proprietà di riflettere le onde (luce o suono) che la colpiscono. Quando il raggio del sensore presenta una angolazione α rispetto alla superficie dell'oggetto, o quando

la stessa superficie non è perfettamente liscia, il raggio verrà riflesso con una angolazione diversa da quella di origine (figura 3.2) causando così misurazioni erronee.

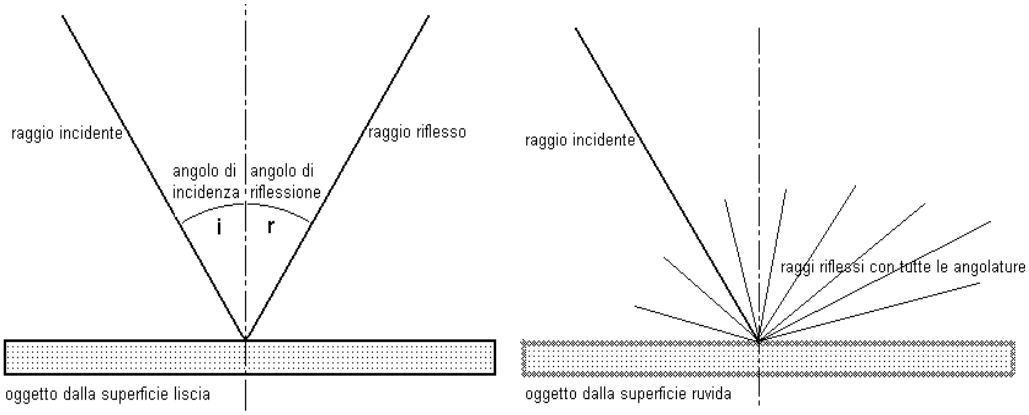

Figure 3.2 Riflessioni del raggio su superfici piane e ruvide

Appare quindi importante considerare anche il modello percettivo del robot $p(z_t \mid x_t, m)$ che consiste nel studiare la probabilità di una certa misurazione noti lo stato x_t e la mappa m. I sensori spesso generano più misurazioni ognuna delle quali verrà indicata con un indice k che ne identifica il numero. Quindi avremo:

$$z_t = \left\{ z_t^1, z_t^2, z_t^3, ..., z_t^k \right\}$$

Grazie a questa notazione possiamo scomporre la probabilità nella produttoria:

$$p(z_t \mid x_t, m) = \prod_{k=1}^{K} p(z_t^k \mid x_t, m)$$

Non dimentichiamo che questa è possibile poichè abbiamo assunto che i rumori delle misurazioni siano indipendenti fra di loro. Tale fattorizzazione è importante in quanto ci permette di trattare la probabilità di singole misurazioni per poi ricondurle al modello percettivo. Tipicamente nell'acquisizione di una misurazione possiamo riscontrare quattro errori significativi:

Misurazioni con rumore: sappiamo che nel caso ideale ricavare la distanza z_t^{id} di un oggetto da un sensore è cosa alquanto banale vista la completa assenza di rumori. Tuttavia se il sensore rileva un oggetto nelle vicinanze la misurazione ottenuta è affetta da errore dovuto a condizioni atmosferiche o anche alla risoluzione del sensore. Il valore misurato, in pratica varia in un intervallo [o, $z_{\max}$] dove $z_{\max}$ sta ad indicare il valore massimo acquisibile. Tipicamente tali errori si schematizzano con una distribuzione gaussiana centrata in z_t^{id} e con una deviazione standard σ_{htt} . Per cui la probabilità avrà una forma del tipo:

$$p_{hit}(z_t^{\,k} \mid x_t, m) = \eta \frac{1}{\sqrt{2\pi\sigma_{hit}^2}} e^{\frac{-(z_t^k - z_t^{id})^2}{\sigma_{hit}^2}} \qquad \text{con } 0 < z_t^{\,k} < z_{\max}$$

dove η è il fattore di normalizzazione dovuto alla della portata del sensore σ_{hit} è un parametro intrinseco del modello percettivo.

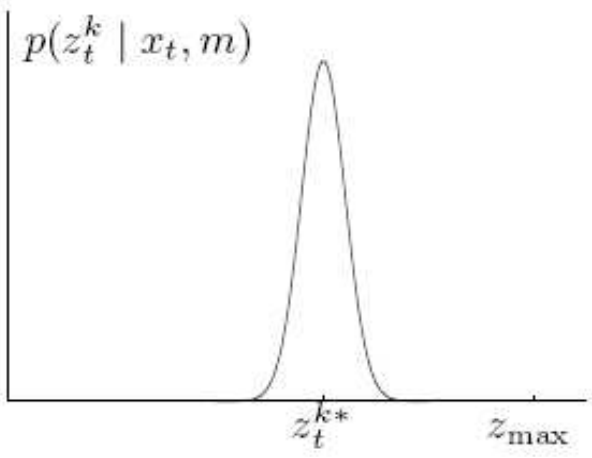

Figura 3.3 Distribuzione esponenziale

Oggetti inaspettati: mentre la mappa m per il robot mobile è statico l'ambiente circostante è dinamico; infatti questo può contenere oggetti che si muovono nel tempo che causano misurazioni inaspettate. Tipicamente tale errore è dovuto alla presenza nell'ambiente di lavoro di persone. Tale situazione può essere gestita introducendo un secondo modello di rumore.

Sicuramente tali errori saranno riscontrati entro un limite massimo che corrisponde z_t^{id}, infatti oltre non avrebbe alcun senso fisico in quanto vorrebbe dire che l'oggetto venga rilevato o dietro un ostacolo o al di fuori dell'intervallo di misurazione massimo $z_{\max}$. Matematicamente tale rumore viene rappresentato come un disturbo esponenziale:

$$p_{short}(z_t^k \mid x_t, m) = \eta \lambda_{short} e^{-\lambda_{short} z_t^k} \qquad \text{con} \quad 0 \le z_t^k \le z_t^{id}$$

con

$$\eta = \frac{1}{1 - e^{-\lambda_{short} z_t^{id}}}$$

E λ_{short} parametro intrinseco del modello percettivo

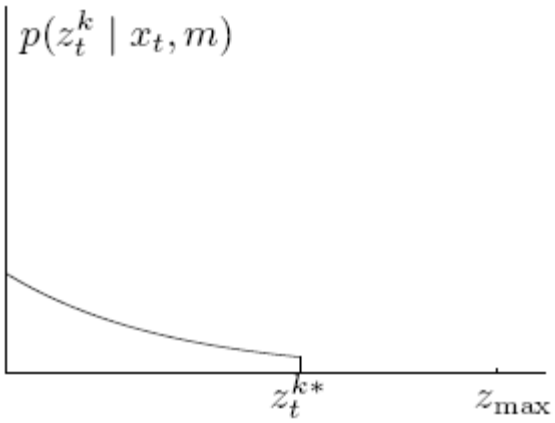

Figura 3.4 distribuzione esponenziale

Misura mancata: a volte può capitare che un singolo ostacolo non venga rilevato. Se ad esempio il robot si trova di fronte ad un ostacolo scuro questo non rileva la misurazione corretta in quanto tale colore non favorisce il fenomeno della riflessione. In tal caso il sensore mi restituisce la misurazione massima $z_{\max}$. Matematicamente tale disturbo può essere rappresentato come un punto di massa centrato in $z_{\max}$

$$p_{\max}(z_t^{\ k} \mid x_t, m) = 1 \qquad \text{con } z = z_{\max}$$

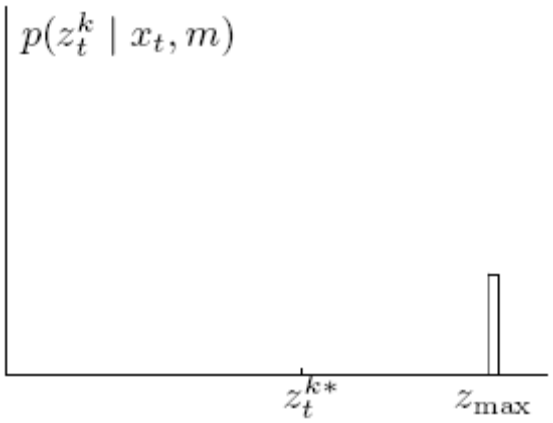

Figura 3.5 Centro di massa

Come si può vedere $p_{\max}$ non possiede una densità in quanto questa è una distribuzione discreta.

Misurazioni casuali: occasionalmente il sensore produce una misurazione inaspettata. Tale misurazione può avere diverse origini ma principalmente esso è causato quando ci sono conflitti con sensori vicini. Tale rumore può essere modellato con una distribuzione uniforme in [0 , $z_{\max}$].

$$p_{rand}(z_t^{\ k} \mid x_t, m) = \frac{1}{z_{\max}} \qquad \text{con } 0 \le z_t^{\ k} \le z_{\max}$$

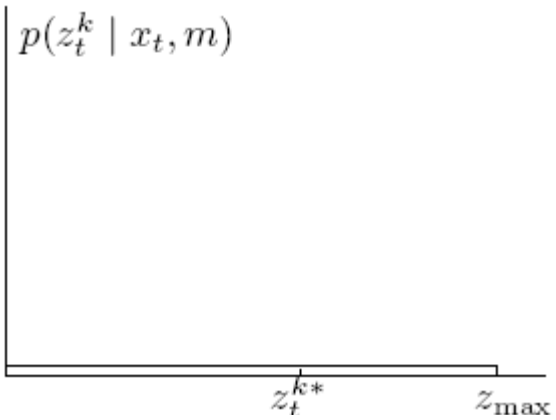

Figura 3.6 Distribuzione per misurazioni casuali

Per ottenere il modello percettivo finale dobbiamo moltiplicare ogni distribuzione per una media pesata. Tenendo conto che

$$z_{hit} + z_{rand} + z_{short} + z_{\max} = 1$$

otteniamo

$$p_{rand}(z_t^k \mid x_t, m) = \begin{pmatrix} z_{hit} \\ z_{short} \\ z_{\max} \\ z_{rand} \end{pmatrix}^T \cdot \begin{pmatrix} p_{hit}(z_t^k \mid x_t, m) \\ p_{short}(z_t^k \mid x_t, m) \\ p_{\max}(z_t^k \mid x_t, m) \\ p_{rand}(z_t^k \mid x_t, m) \end{pmatrix}$$

Un possibile risultato di tale probabilità viene di seguito riportato

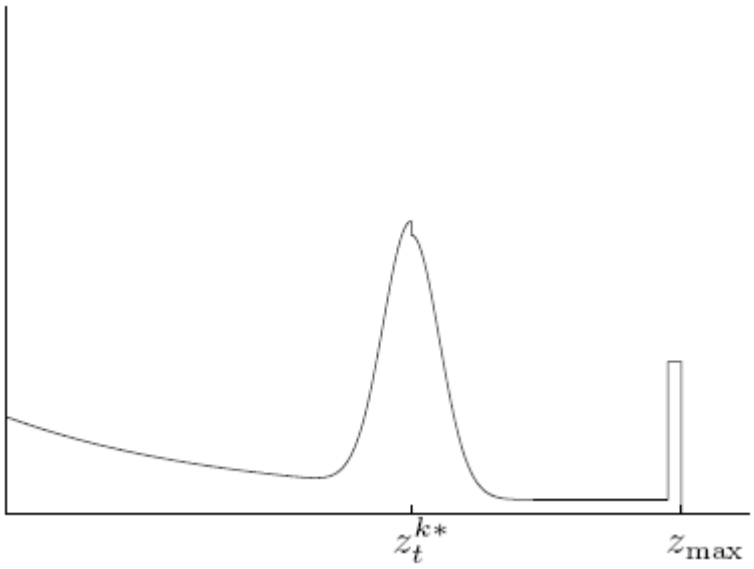

Figura 3.7 Distribuzione risultante

Quanto appena detto vale genericamente per gran parte dei sensori utilizzati. Tale approccio però presenta degli inconvenienti da non sottovalutare. Se consideriamo un ambiente in cui vi siano tanti piccoli ostacoli notiamo che la probabilità $p(z_t^k \mid x_t, m)$ presenta notevoli discontinuità. Questo è il caso in cui il robot si trovi in una stanza in cui vi siano un numero considerevole di sedie o tavolini. I sensori individueranno nelle gambe degli ostacoli. Se si varia però leggermente la posizione di x_t

o l'orientamento del robot mobile possiamo riscontrare enormi ripercussioni sulla stima del robot.

3.3 SIMULATORE DELLE MISURAZIONE

Si voglia ora creare un simulatore delle misurazioni il quale data la mappa e la posizione del robot restituisca le misurazioni dei sensori. Immaginiamo di utilizzare 8 sensori di distanza posizionati sul robot mobile di forma circolare. Il primo sensore è stato posizionato lungo l'orientamento del robot mentre i rimanti, distribuiti ogni quarantacinque gradi lungo la circonferenza.

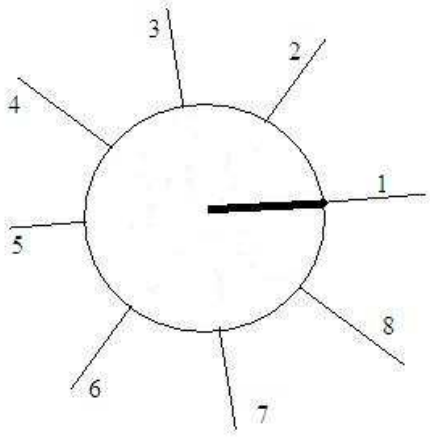

Figura 3.8 Distribuzione sensori

La struttura del cono del sensore, momentaneamente, è stata schematizzata come un raggio uscente dal centro del sensore e avente lunghezza massima di tre metri. Per la realizzazione di tale modello si è utilizzato il compilatore MATLAB. L'idea di fondo è stata quella di schematizzare i raggi e le pareti come spezzate di cui si conoscessero almeno due punti. Tali rette possono essere tra loro parallele, ortogonali o sghembe. Nel primo caso la misurazione del sensore non porta informazioni di alcun genere in quanto significa che il raggio e la parete

sono paralleli o coincidenti. In questo caso il sensore non restituisce alcuna misurazione (Figura 3.9).

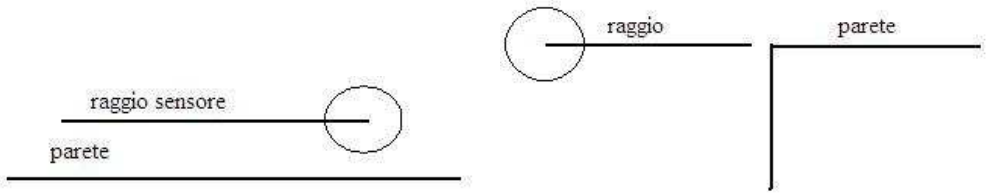

Figura 3.9 Rette parallele o coincidenti: la misurazione non porta alcuna informazione

Nel caso in cui il raggio e la parete sono tra loro ortogonali si calcola il punto di intersezione. Stimata la distanza si valuta se questa è all'interno o no del range massimo del sensore [0 , $z_{\max}$]. Si deve inoltre valutare se tale misurazione è o no rilevabile dal sensore selezionato. Infatti è poco realistico che il sensore riporti informazioni dell'ambiente alle sue spalle. Procedimento del tutto simile viene applicato nel caso in cui le due rette siano tra loro sghembe. Calcolata la reale misurazione z_t^{id} dobbiamo modellare il rumore che tale misurazione porta con sè nella realtà. Possiamo considerare di trovarci in un ambiente statico. Il disturbo, come detto in precedenza, può essere rappresentato come una gaussiana centrata in zero. Estraendo valori casuali da tale distribuzione e sommandoli alle misurazione otteniamo così il rumore.

CAPITOLO 4
TECNICA DI MAPPATURA BASATA SU GRIGLIA OCCUPAZIONALE

4.1 ALGORITMO PER LA MAPPATURA BASATA SU TECNICHE COMPUTAZIONALI E MODELLO INVERSO DEL SENSORE

I sistemi robotici oggi giorno sono intercalati nell ambiente che ci circonda e hanno il compito di ricevere informazione sull'ambiente esterno attraverso i loro sensori e di modificarlo secondo i propri scopi attraverso gli attuatori. Sempre più spesso per compiere a pieno le proprie funzioni hanno necessità di conoscere l'ambiente in cui operano e ridurre al minimo la propria incertezza su di esso. Imparare a conoscere ciò che li circonda diviene sempre più fondamentale. Sappiamo infatti che lo spazio di stato, che è lo spazio di tutte le possibili mappe è enorme. Le mappe sono definite su uno spazio continuo il quale ha dimensioni infinite. Per capirne le grandezze immaginiamo di dover mappare un ambiente come la nostra casa. Una mappa dettagliata bidimensionale dell'intero piano spesso può richiedere migliaia di numeri ma una mappa dettagliata tridimensionale di un edificio potrebbe facilmente richiedere milioni di numeri. Tuttavia non tutti i problemi di mappatura dell'ambiente presentano la stessa difficoltà. La complessità è data da alcuni fattori fondamentali

- Ampiezza: più l'ambiente è vasto, più l'acquisizione della mappa è difficile;
- Rumore: più i sensori sono affetti da rumore, più la mappa non è fedele alla realtà;
- Cicli: percorsi chiusi per un robot sono particolarmente difficili da riconoscere in quanto muovendosi esso accumula degli errori nell' odometria che si ripercuotono sulla stima del robot.

Nella tecnicha di mappatura basata su celle occupazionali (*occupancy grid mapping*) si suppone di conoscere perfettamente la posizione che il robot possiede nell'ambiente. Conoscendo la posizione vogliamo ricavare la conformazione dell'ambiente circostante dall'incertezza delle misurazioni dei sensori. L'idea di fondo è quella di dividere l'intero ambiente in celle, ognuna delle quali possiede una variabile binaria che ne descrive lo stato. Come abbiamo visto per il filtro di Bayes però tale approccio presenta degli inconvenienti in quanto non ci permette di rappresentare la dipendenza tra celle confinanti. Sfrutteremo l'algoritmo del filtro binario di Bayes per superare tali limitazioni. Come detto possiamo dividere l'intera mappa m in n celle m_i cosi da partizionare lo spazio di stato in un numero finito di celle

$$m = \bigtimes_i m_i$$

Ne consegue che calcolare la probabilità p(m_i=1) vuol significare sapere la probabilità che una data cella m_i sia occupata. Inoltre il problema di poter calcolare la distribuzione della mappa $p(m \mid z_{1:t}, x_{1:t})$ note le misurazioni z e le posizioni x può essere decomposto come tanti sottoproblemi:

$$p(m \mid z_{1:t}, x_{1:t}) = \prod_i p(m_i \mid z_{1:t}, x_{1:t})$$

In cui la complessità viene ricondotta ad un problema di stima binaria con lo spazio di stato finito e statico. Il nostro algoritmo dovrà quindi misurare ciò che lo circonda e restituire ogni singolo valore delle celle osservate. Quello che vogliamo sapere è date le misurazioni dei sensori e la posizione quanto vale $p(m_i \mid z_t, x_t)$. Ricordando la forma del rapporto delle probabilità dei logaritmi (log odds ratio) utilizzati nel filtro di Bayes binario possiamo valutare tale densità come:

$$p(m_i \mid z_t, x_t) = \frac{p(m_i \mid z_t, x_t)}{1 - p(m_i \mid z_t, x_t)}$$

Tale modello è chiamato modello inverso del sensore in quanto ci restituisce informazioni dell'effetto nota la causa. Il modello inverso cioè mi restituisce informazioni sull'ambiente circostante studiando le misurazioni dei sensori condizionati dall'ambiente esterno. Ricordando che il robot può ricavare informazioni soltanto se ha delle misurazioni dell'ambiente possiamo avere tre tipi di probabilità a seconda se la cella considerata è libera (l_{free}) occupata (l_{occ}) o fuori dall'intervallo di misurazioni dei sensori (l_0). I valori delle probabilità sono delle variabili costanti scelte in modo che $l_{free} < l_0$, $l_{occ} > l_0$ dove $l_0 = \log\left(\frac{p(m_i = 1)}{p(m_i = 0)}\right)$ rappresenta la probabilità iniziale Per eliminare inoltre le ambiguità che esistono tra celle confinanti identifichiamo la singola cella m_i con un centro di massa che ne indichi le coordinate del centro (x_c , y_c). Quindi, noti la posizione x_t, le misurazioni z_t possiamo calcolare la probabilità della singola cella.

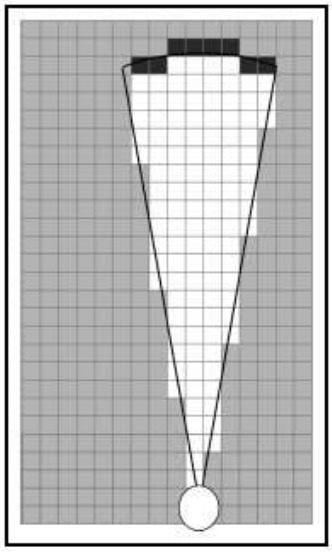

Figura 4.1 Esempio di misurazione per un singolo sensore

In figura 4.1 viene riportato ciò che avviene per una singola misurazione. Calcolato il centro di massa e le celle che ricadono all'interno del cono il modello inverso assegna ad ognuna un valore costante. L'algoritmo di mappatura basato su griglia occupazionale inizialmente assegna una

probabilità uniforme a tutta la mappa che corrisponde a $l_0 = \log\left(\frac{p(m_i = 1)}{p(m_i = 0)}\right)$. Ad ogni misurazione lui considera la distribuzione iniziale e attraverso il modello inverso del sensore ne aggiorna la probabilità sommando semplicemente il valore corrispondente. Possiamo quindi scrivere che la probabilità al tempo t della casella i-esima, se contenuta all'interno del cono di misurazione è:

$$l_{t,i} = l_{t-1,i} + l_{\mathrm{mod}\,inv} - l_0$$

altrimenti la cella rimane con la distribuzione precedente

$$l_{t,i} = l_{t-1,i}\,.$$

Le istruzioni appena scritte compongono l'algoritmo per determinare una mappatura dell'ambiente. Sostanzialmente la mappatura con celle occupazionali può essere riportata come segue:

Algoritmo di mappatura con celle occupazionali

$\forall m_i$

se m_i è contenuta nel cono del sensore

$l_{t,i} = l_{t-1,i} + l_{\mathrm{mod}\,inv} - l_0$;

else

$l_{t,i} = l_{t-1,i}$;

dove $l_{\mathrm{mod}\,inv}$ è dato dall'algoritmo

Modello inverso del sensore

$$r = \sqrt{(x_i - x)^2 + (y_i - y)^2}$$

$$\phi = ATAN\,2((y_i - y),(x_i - x)\ ) - \theta$$

$$k = \arg\min\left|\theta - \theta_{j,sens}\right|$$

se $r > \min(z_{\max}, z_t^k + \frac{\alpha}{2})$ o $\left|\theta - \theta_{j,sens}\right| > \frac{\beta}{2}$

$$l_{\mathrm{mod}\,inv} = l_0 \ ;$$

altrimenti se $z_t^k < z_{\max}$ e $\left|r - z_t^k\right| < \frac{\alpha}{2}$

$$l_{\mathrm{mod}\,inv} = l_{occ}$$

altrimenti $r \le z_t^k$

$$l_{\mathrm{mod}\,inv} = l_{free}$$

Dove α è lo spessore delle pareti e β è l'ampiezza del cono. Tale modello vale per ogni singolo sensore. Vediamo ora come agisce nella realtà l'algoritmo. Consideriamo di essere all'interno di un ambiente in cui le posizioni via via assunte dal robot sono schematizzate con dei cerchi (figura4.2).

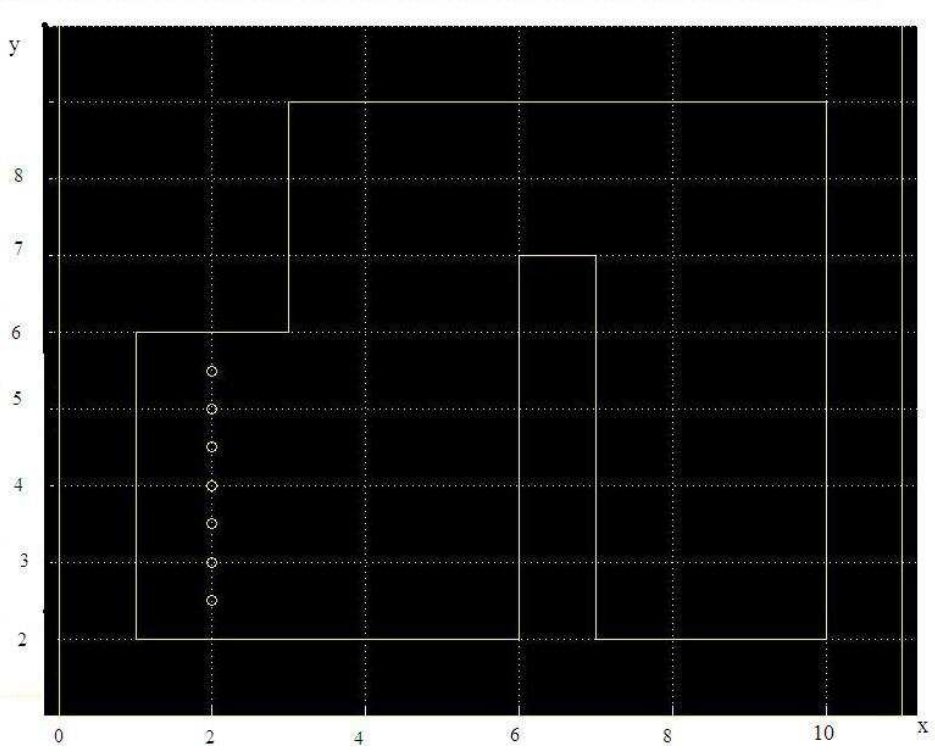

Figura 4.2

dopo la prima iterazione la conoscenza del robot sul ambiente aumenta

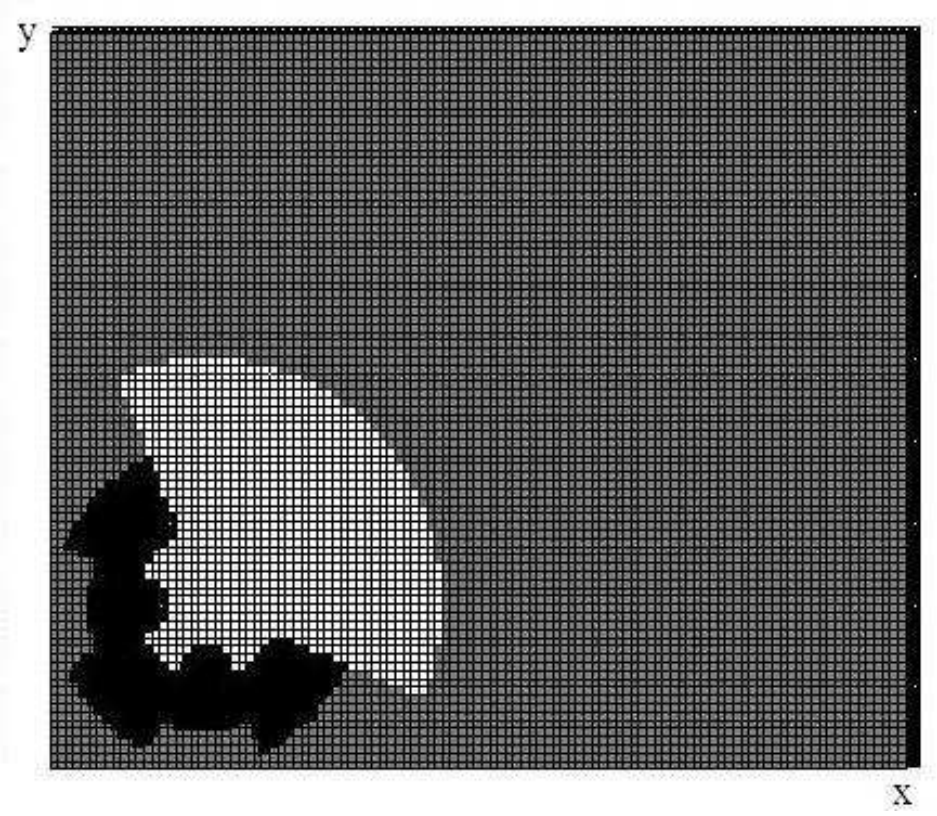

Figura 4.3 Stima dell' ambiente per la posizione [2,2]

Spostando il robot l'informazione sull'ambiente esterno aumenta

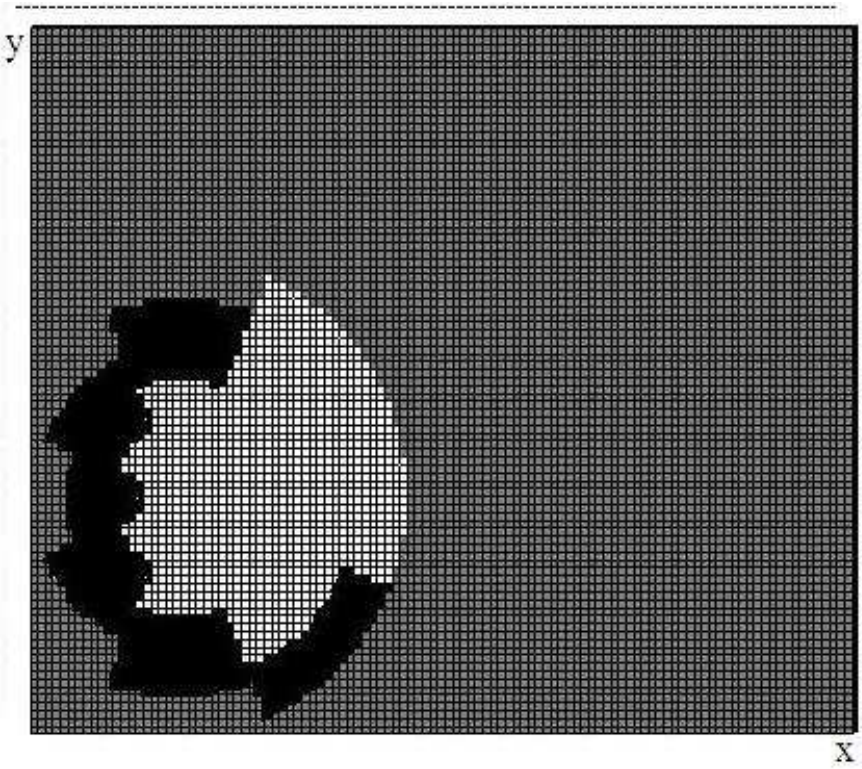

Figura 4.4 Visione dell' ambiente per la posizione [2,3]

Ogni posizione porta nuove informazioni. Continuando a iterare per tutte le posizioni otteniamo che la stima totale del robot sarà(Figura 4.4):

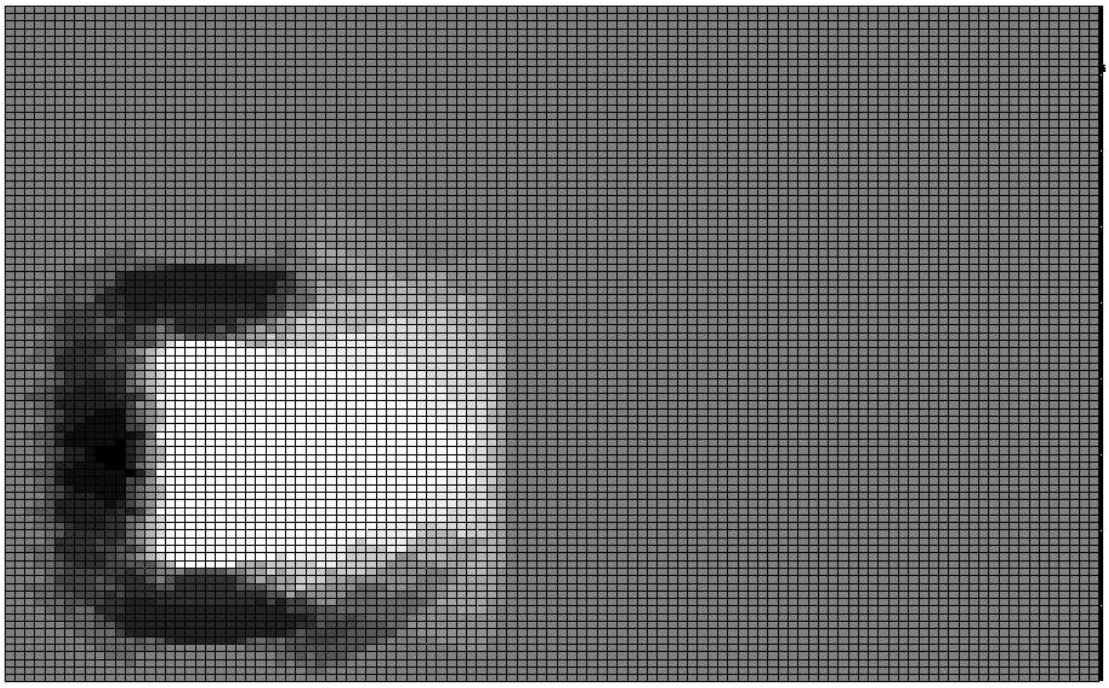

Figura 4.5 Stima finale

4.2 RISULTATI SIMULATIVI

Vogliamo ora applicare gli algoritmi appena descritti. Consideriamo di trovarci dentro un stanza lunga dodici metri e larga nove. Cosi composta (figura4.6)

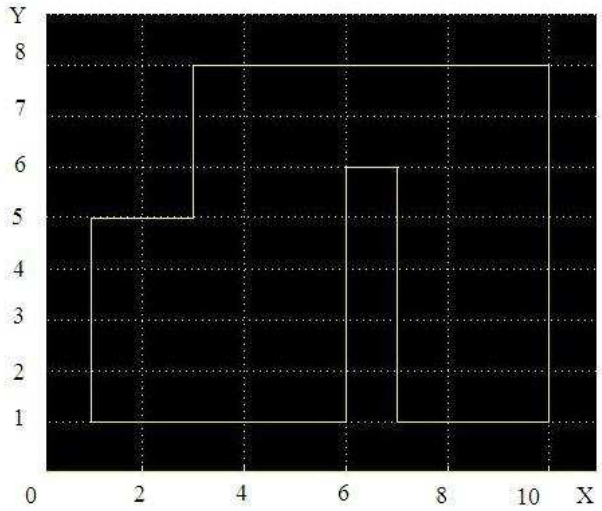

Figura 4.6 Ambiente

Posizioniamo il robot vicino all'angolo inferiore della stanza e spostiamolo di cinquanta centimetri per ogni posizione in modo che alla fine si abbia una visione completa dell'intero ambiente(figura 4.7).

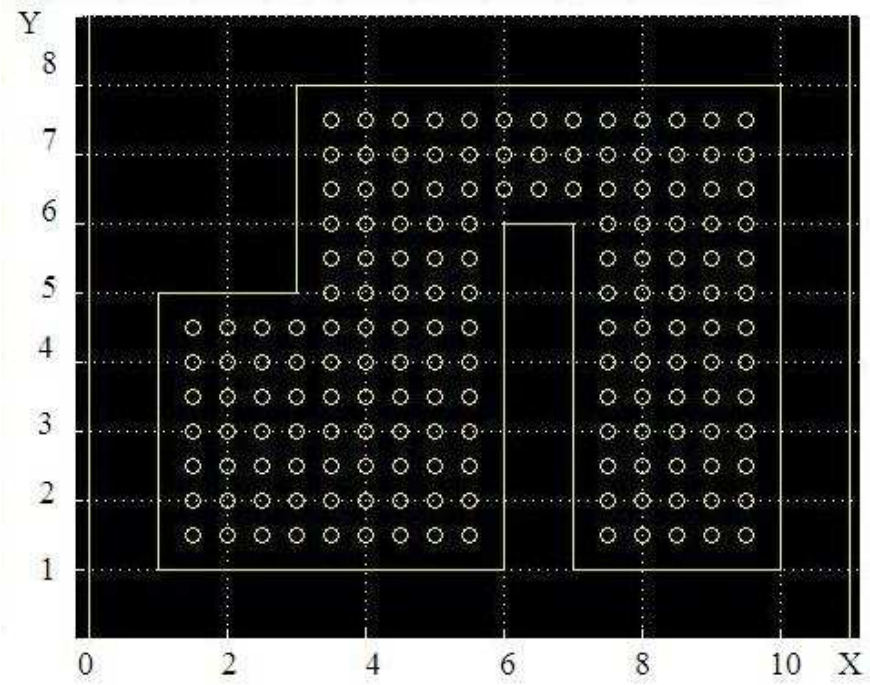

Figura 4.7 Tute le posizioni assunte dal robot mobile

Applicando la tecnica di mappatura basata su griglie occupazionali otterremo una mappa simile a quella reale:

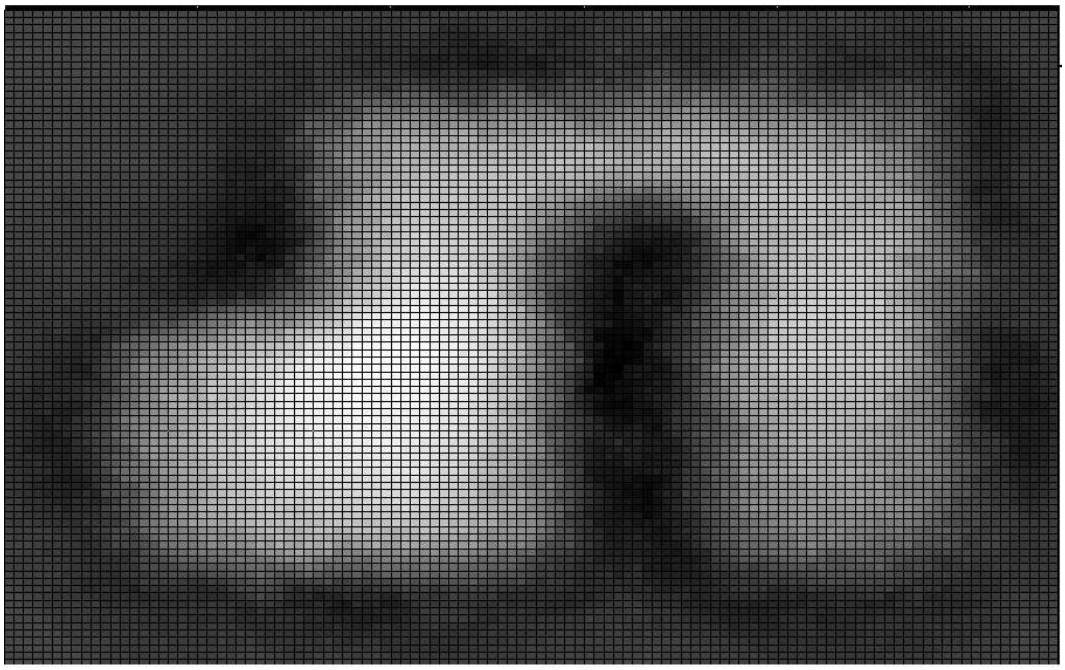

Figura 4.8

La fedeltà la precisione e la velocità nel ricostruire la mappa dipendono da molti fattori. Prima di tutto come detto in precedenza non è possibile mappare un ambiente in maniera dettagliata in quanto tutte le misurazioni sono affette da rumori. Nella figura 4.8 precedente abbiamo schematizzato il rumore con una gaussiana avente varianza pari a 0.5. Tale aspetto introduce un rumore considerevole perché fa variare le misurazioni di un fattore incluso tra ± 0.5 metri. Se infatti analizziamo la stessa mappa con le stesse posizioni ma cambiando semplicemente la precisione dei sensori notiamo notevoli cambiamenti. Supponiamo di considerare sensori di distanza aventi una varianza di 0.01. La stima dell'ambiente cambia notevolmente.

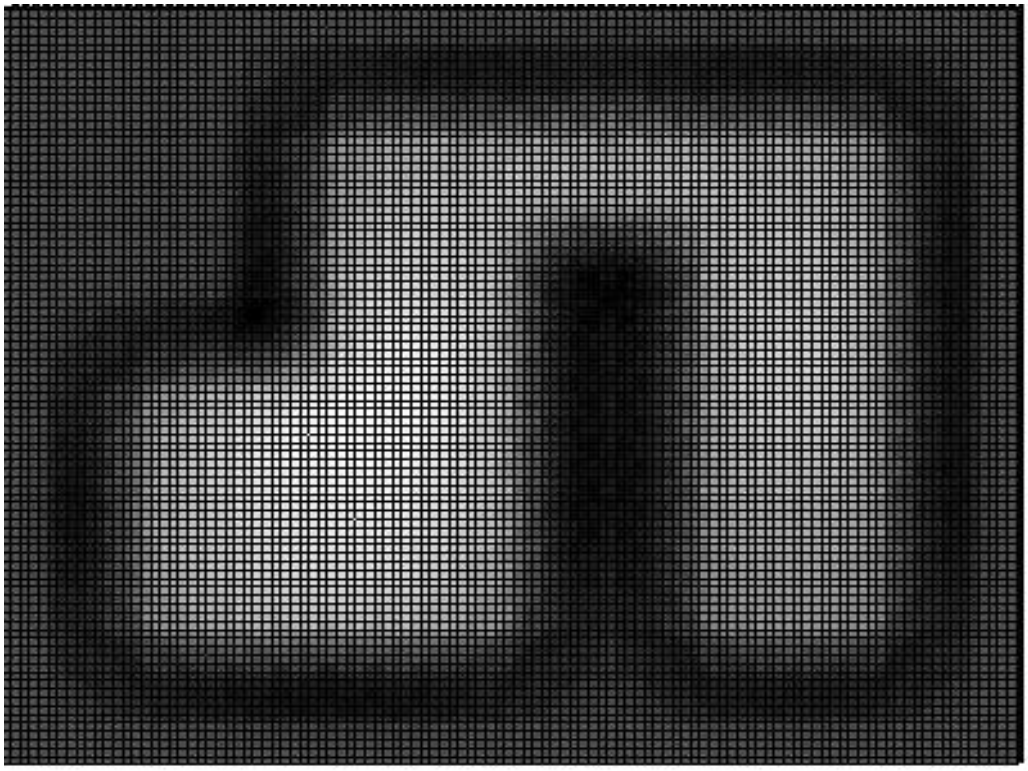

Figura 4.10

Quest'ultima mappatura può considerarsi una buona approssimazione dell'ambiente. Ricordiamo che assumiamo di conoscere la posizione esattamente. Se consideriamo che il nostro raggio arriva fino ad una distanza di tre metri e avanziamo di 50 centimetri per misurazione l'errore introdotto dal rumore dei sensori ci porta ad una incertezza di circa un centimetro. Tale fattore sembra abbastanza ragionevole viste anche le dimensioni della stanza. La scelta del sensore influisce molto sull'accuratezza della stima: minore è la sua dipendenza dal rumore e migliore sarà la stima. Riportiamo in figura 4.11 una stima in assenza di rumori.

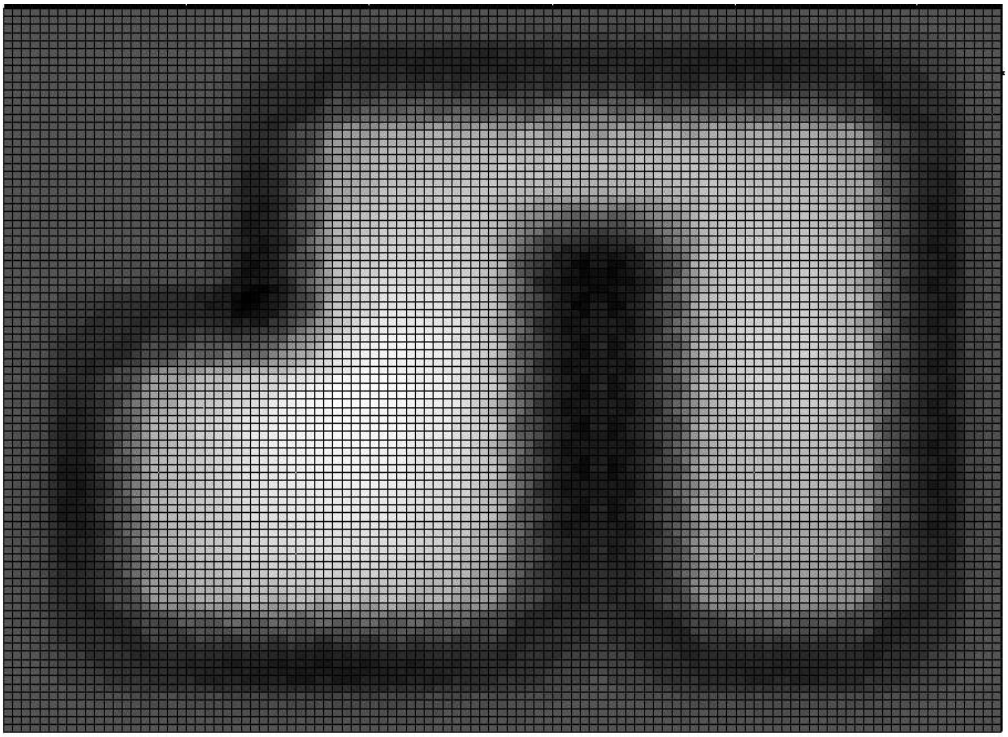

Figura 4.11stima con assenza di rumori

Confrontando il risultato ottenuto con l'ultima mappatura ottenuta notiamo che non ci sono notevoli differenze. Ciò sta a dimostrare la bontà della simulazione. La dipendenza delle misurazioni dal rumore è senza dubbio l'aspetto che maggiormente penalizza la precisione ma non è certamente l'unico. Notevole attenzione va anche posta sulla grandezza delle celle scelte. Tale aspetto influisce sia sulla qualità che sulla velocità dell'algoritmo. I due aspetti però sono inversamente proporzionali. Infatti all'aumentare del numero di celle e quindi all'aumentare della risoluzione ne risente la velocità computazionale. Se come nel nostro caso abbiamo una stanza (12 x 9) metri suddivisa in celle grandi cinquanta centimetri il numero di celle da analizzare per ogni ciclo è 216. Il numero aumenta se consideriamo le celle essere grandi 10 centimetri. Per ogni posizione avremmo che l'algoritmo deve processare 1080 celle. Se consideriamo che tale numero va moltiplicato per il numero di posizioni totali assunte nell'intera simulazione il numero cresce drasticamente. Rapportando quanto appena detto nel nostro esempio simulato sappiamo che per 169 posizioni nel primo caso vengono analizzate 36.504 celle mentre nel

secondo ben 182.520 celle. Vengono di seguito riportati le mappe dei due casi portati in esempio (figura 4.12,figura 4.16)

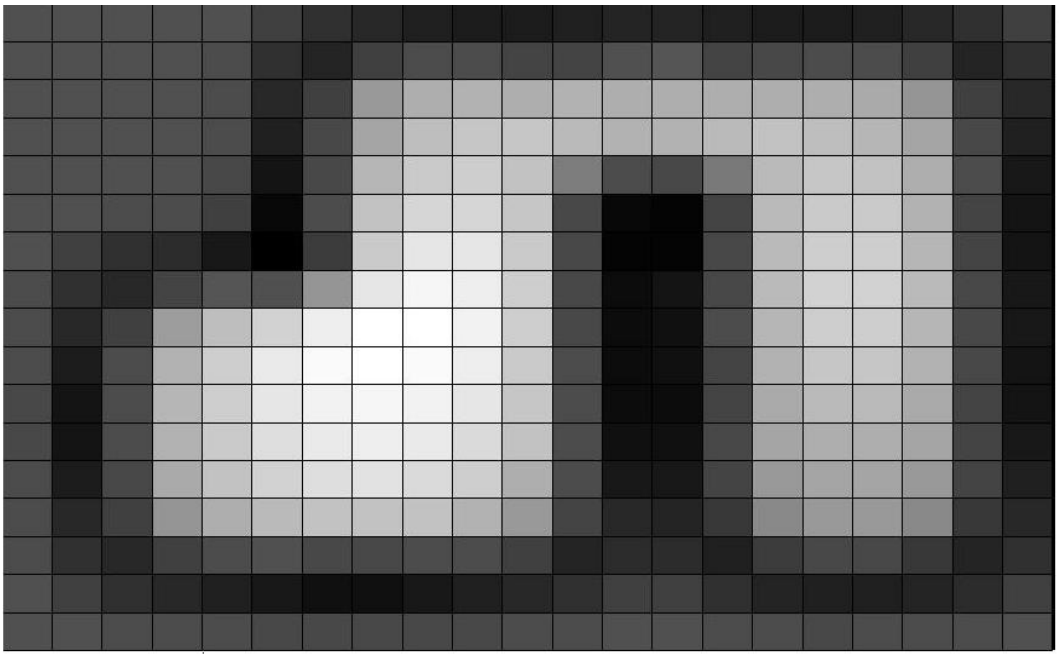

Figura4.12 mappatura con una risoluzione delle celle di 0.5 metri

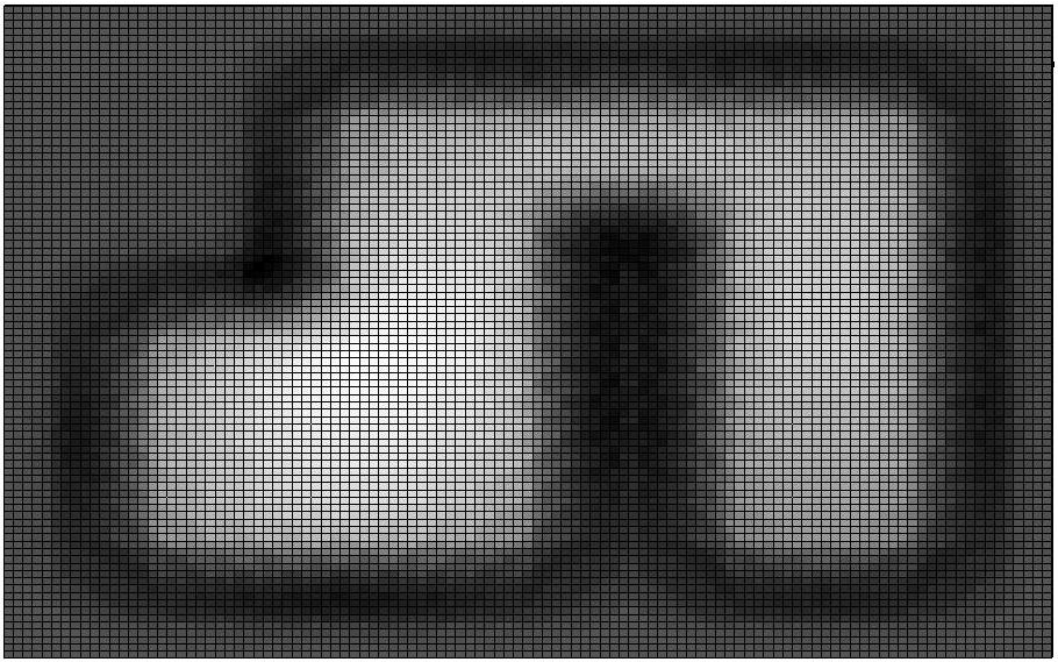

Figura4.16 mappatura con risoluzione delle celle di 0.1 metri

Risultato del tutto simile si poteva avere analizzando l'algoritmo di mappatura basato su celle di occupazione. Se indichiamo con

P= numero di posizioni totali;

G_C= misura del lato della singola cella quadrata;

(X,Y)= la grandezza della stanza

Sappiamo che l'algoritmo ha una complessità computazionale T pari a

$$T = \frac{P \cdot X \cdot Y}{G\_C^2}$$

Ciò non fa altro che confermare quanto prima detto. Infatti all'aumentare delle posizioni l'algoritmo cresce linearmente la sua grandezza computazionale così come aumenta quadraticamente al diminuire della grandezza della cella. Riportiamo in figura 4.12 un grafico in cui sulle ascisse viene riportate la grandezza delle celle e sulle ordinate il tempo di calcolo

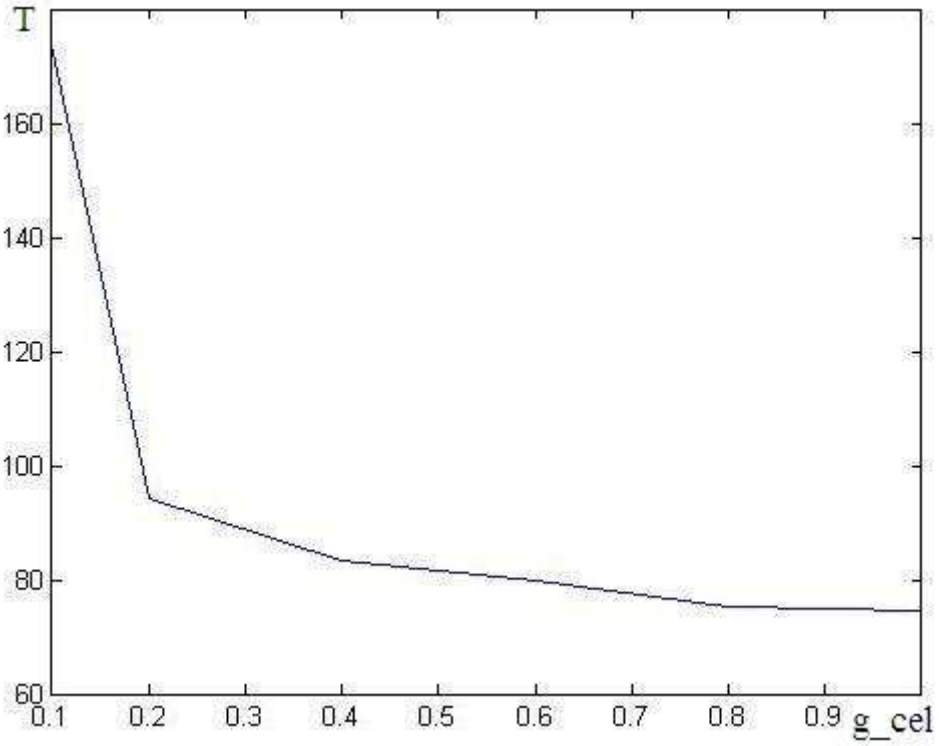

Figura 4.12 variazione del tempo al crescere della grandezza delle celle

CAPITOLO 5

CONSIDERAZIONI E SVILUPPI FUTURI

L'algoritmo di mappatura basato su celle occupazionali è un buono strumento per imparare la conformazione dell'ambiente di lavoro del robot mobile. La funzionalità di tale algoritmo dipende principalmente dall'accuratezza dei sensori. Più sono sensibili al rumore più la ricostruzione della mappa sarà meno fedele. Per quanto riguarda invece la risoluzione è a totale discrezione del programmatore in quanto è lui a decidere la grandezza dei parametri come la grandezza delle celle. Non va dimenticato che abbiamo supposto l'ambiente completamente statico. Ciò vuol significare che non è prevista la presenza di oggetti che si muovono o cambiano stato nell' arco del tempo. La presenza di persone all'interno dell'ambiente rilevato può causare degli errori di mappatura.
Il tempo di esecuzione dell'algoritmo sembra abbastanza ragionevole anche per celle di grandezza inferiore al centimetro. La grandezza delle variabili in gioco è strettamente dipendente dalla grandezza della mappa. La mappatura di un ambiente può essere vista come uno strumento utile di problematiche più complesse.

Listato Programma

```
% definizioni parametri robot
 clear all;clc
global z_max pos_rob d1 theta_robot g_cel g_alpha g_beta l0 locc lfree  x_max y_max x
dim g_y iter x2_max y2_max var
z_max=3;
pos_rob
var=0.02;
def=size(pos_rob);
iter=def(1,1);
d1=0; % raggio robot
theta_robot=deg2rad(0); % angolo del robot
g_cel=0.4; %grandezza della cella
g_alpha=1; % spessore delle pareti
g_beta=deg2rad(45); % angolo di apertura del sensore
% definizione delle probabilità
l0=0; % probabilità iniziale l0=p(m(i)=1)/p(m(i)=0)
locc=100; % probabilita che la casella sia occupata
lfree=-100; %probabilità che la cosella sia libera
% VERTICI  AMBIENTE
 % mappa interna
 g_y(1,:)=[1,1];
 g_y(2,:)=[6,1];
 g_y(3,:)=[6,6];
 g_y(4,:)=[7,6];
 g_y(5,:)=[7,1];
 g_y(6,:)=[10,1];
 g_y(7,:)=[10,8];
 g_y(8,:)=[3,8];
 g_y( 9,:)=[3,5];
 g_y(10,:)=[1,5];
 g_y(11,:)=[1,1];
 % MAPPA
 figure(1)
 grid on
```

```
 hold on
 plot(g_y(:,1),g_y(:,2))
 plot(pos_rob(:,1),pos_rob(:,2),'o')
 axis equal
 % propietà vettore y;
 vf=size(g_y);
 dim=(vf(1,1)-1);
 x_max=max(g_y(:,1));
 y_max=max(g_y(:,2));
 % mappa esterna
 pqx1=min(g_y(:,1));
 pqx2=max(g_y(:,1));
 pqy1=min(g_y(:,2));
 pqy2=max(g_y(:,2));
 x(1,:)=[pqx1-1,pqy1-1];
 x(2,:)=[pqx2+1,pqy1-1];
 x(3,:)=[pqx2+1,pqy2+1];
 x(4,:)=[pqx1-1,pqy2+1];
 x(5,:)=[pqx1-1,pqy1-1];
 vf2=size(x);
 dim2=(vf2(1,1)-1);
 x2_max=max(x(:,1));
 y2_max=max(x(:,2));
% subplot(2,1,1)
  plot(x(:,1),x(:,2))
% SIMULATORE PER IL CALCOLO DELLE MISURE ZT
function[mis_2, theta_sens, theta_ass] = sim_sen(o)
global z_max pos_rob d1 theta_robot g_cel g_alpha g_beta l0 locc lfree x_max y_max x
dim g_y iter x2_max y2_max var
%REALIZZAZIONE AMBIENTE
figure(1)
hold on
grid on
plot(g_y(:,1),g_y(:,2))
plot(pos_rob(o,1),pos_rob(o,2),'o')
plot(x(:,1),x(:,2))
axis equal
```

```
sic=0.001;
for q=1:8
   k=1;
   theta_sens(q)=(q-1)*deg2rad(45); % angolazione q.esimo sensore
   theta_ass(q)=theta_robot+theta_sens(q);
   x_zmax(q)=pos_rob(o,1)+(z_max+d1)*cos(theta_ass(q));% CORDINATE RISPETTO
ORIGINE
   y_zmax(q)=pos_rob(o,2)+(z_max+d1)*sin(theta_ass(q));% CORDINATE RIPETTO
ORIGINE
   % CALCOLO DEI COEFFICINETI M1 E M2 DELLE RETTE
   m1(q)=(y_zmax(q)-pos_rob(o,2))/(x_zmax(q)-pos_rob(o,1));
   % CALCOLO DEI COEFFICENTI DELLE RETTE DEGLI OSTACOLI
   for s=1:dim
      m2(s)=(g_y(s+1,2)-g_y(s,2))/(g_y(s+1,1)-g_y(s,1));
   end
   for i=1:dim
      if theta_ass(q)==deg2rad(0)     % m1=0 ==>
         if m2(i)==inf | m2(i)==-inf     % m2=inf otogonale
            yp=pos_rob(o,2);   % coordinate del punto di intersezione
            xp=g_y(i,1);
            r1=xp-pos_rob(o,1);
            if r1>d1
               if yp<=max(g_y(i,2),g_y(i+1,2)) & yp>=min(g_y(i,2),g_y(i+1,2)) % condizione
di appartenza all'ostacolo
                  if r1>=z_max+d1
                     mis(q,k)=z_max;
                     k=k+1;
                     indice(q,k)=i;
                  else
                     mis(q,k)=r1-d1;
                     k=k+1;
                     indice(q,k)=i;
                  end
               else
                  mis(q,k)=z_max;
                  k=k+1;
                  indice(q,k)=i;
```

```
            end
        end
    elseif m2(i)~=0 & (m2(i)~=inf | m2(i)~=-inf)  % m2=c2
        yp=pos_rob(o,2);
        q2=g_y(i,2)-m2(i)*g_y(i,1);
        xp=(yp-q2)/m2(i);
        r1=xp-pos_rob(o,1);
        if r1>d1
            if yp<=max(g_y(i,2),g_y(i+1,2)) & yp>=min(g_y(i,2),g_y(i+1,2))  % condizione
di appartenza all'ostacolo
                if r1>=z_max+d1
                    mis(q,k)=z_max;
                    k=k+1;
                    indice(q,k)=i;
                else
                    mis(q,k)=r1-d1;
                    k=k+1;
                    indice(q,k)=i;
                end
            else
                mis(q,k)=z_max;
                k=k+1;
                indice(q,k)=i;
            end
        end
    end
elseif theta_ass(q)==deg2rad(180)   % m1=0 <==
    if m2(i)==inf | m2(i)==-inf  % m2=inf
        yp=pos_rob(o,2);
        xp=g_y(i,1);
        r1=xp-pos_rob(o,1);
        if r1<-d1
            r1=abs(r1);
            if yp<=max(g_y(i,2),g_y(i+1,2)) & yp>=min(g_y(i,2),g_y(i+1,2))  % condizione
di appartenza all'ostacolo
                if r1>=z_max+d1
                    mis(q,k)=z_max;
```

```
                k=k+1;
                indice(q,k)=i;
            else
                mis(q,k)=r1-d1;
                k=k+1;
                indice(q,k)=i;
            end
        else
            mis(q,k)=z_max;
            k=k+1;
            indice(q,k)=i;
        end
    end
elseif m2(i)~=0 & (m2(i)~=inf | m2(i)~=-inf) % m2=c2
    yp=pos_rob(o,2);
    q2=g_y(i,2)-m2(i)*g_y(i,1);
    xp=(yp-q2)/m2(i);
    r1=xp-pos_rob(o,1);
    if r1<-d1
        r1=abs(r1);
        if yp<=max(g_y(i,2),g_y(i+1,2)) & yp>=min(g_y(i,2),g_y(i+1,2)) % condizione
di appartenza all'ostacolo
            if r1>=z_max+d1
                mis(q,k)=z_max;
                k=k+1;
                indice(q,k)=i;
            else
                mis(q,k)=r1-d1;
                k=k+1;
                indice(q,k)=i;
            end
        else
            mis(q,k)=z_max;
            k=k+1;
            indice(q,k)=i;
        end
    end
```

```
            end
        elseif theta_ass(q)==deg2rad(90)   % m1=inf
            if m2(i)==0     % m2=0
                xp=pos_rob(o,1);
                yp=g_y(i,2);
                r1=yp-pos_rob(o,2);
                if r1>d1
                    if xp<=max(g_y(i,1),g_y(i+1,1)) & xp>=min(g_y(i,1),g_y(i+1,1))  % condizione
di appartenza all'ostacolo
                        if r1>=z_max+d1
                            mis(q,k)=z_max;
                            k=k+1;
                            indice(q,k)=i;
                        else
                            mis(q,k)=r1-d1;
                            k=k+1;
                            indice(q,k)=i;
                        end
                    else
                        mis(q,k)=z_max;
                        k=k+1;
                        indice(q,k)=i;
                    end
                end
            elseif m2(i)~=0 & (m2(i)~=inf | m2(i)~=-inf)   % m2=c2
                xp=pos_rob(o,1);
                q2=g_y(i,2)-m2(i)*g_y(i,1);
                yp=m2(i)*xp+q2;
                r1=yp-pos_rob(o,2);
                if r1>d1
                    if xp<=max(g_y(i,1),g_y(i+1,1)) & xp>=min(g_y(i,1),g_y(i+1,1))  % condizione
di appartenza all'ostacolo
                        if r1>=z_max+d1
                            mis(q,k)=z_max;
                            k=k+1;
                            indice(q,k)=i;
                        else
```

```
                    mis(q,k)=r1-d1;
                    k=k+1;
                    indice(q,k)=i;
                end
            else
                mis(q,k)=z_max;
                k=k+1;
                indice(q,k)=i;
            end
        end
    elseif theta_ass(q)==deg2rad(270)  % m1=-inf
        if m2(i)==0
            xp=pos_rob(o,1);
            yp=g_y(i,2);
            r1=yp-pos_rob(o,2);
            if r1<-d1
                r1=abs(r1);
                if xp<=max(g_y(i,1),g_y(i+1,1)) & xp>=min(g_y(i,1),g_y(i+1,1))% condizione
di appartenza all'ostacolo
                    if r1>=z_max+d1
                        mis(q,k)=z_max;
                        k=k+1;
                        indice(q,k)=i;
                    else
                        mis(q,k)=r1-d1;
                        k=k+1;
                        indice(q,k)=i;
                    end
                else
                    mis(q,k)=z_max;
                    k=k+1;
                    indice(q,k)=i;
                end
            end
        elseif m2(i)~=0 & (m2(i)~=inf | m2(i)~=-inf)
            xp=pos_rob(o,1);
```

```
            q2=g_y(i,2)-m2(i)*g_y(i,1);
            yp=m2(i)*xp+q2;
            r1=yp-pos_rob(o,2);
            if r1<-d1
               r1=abs(r1);
               if xp<=max(g_y(i,1),g_y(i+1,1)) & xp>=min(g_y(i,1),g_y(i+1,1))
                  if r1>=z_max+d1
                     mis(q,k)=z_max;
                     k=k+1;
                     indice(q,k)=i;
                  else
                     mis(q,k)=r1-d1;
                     k=k+1;
                     indice(q,k)=i;
                  end
               else
                  mis(q,k)=z_max;
                  k=k+1;
                  indice(q,k)=i;
               end
            end
         end
      elseif (theta_ass(q)>deg2rad(270) | theta_ass(q)<deg2rad(90)) &
theta_ass(q)~=deg2rad(0)
         if (m2(i)==inf | m2(i)==-inf)
            q1=pos_rob(o,2)-m1(q)*pos_rob(o,1);
            xp=g_y(i,1);
            yp=m1(q)*xp+q1;
            if xp>pos_rob(o,1)+d1*cos(theta_ass(q))
               r1=sqrt((pos_rob(o,1)-xp)^2+(pos_rob(o,2)-yp)^2);
               if r1>d1
                  if (yp<=(max(g_y(i,2),g_y(i+1,2))+sic)) & (yp>=(min(g_y(i,2),g_y(i+1,2))-
sic)) % il controllo va fatto sulle y?
                     if r1>=z_max+d1
                        mis(q,k)=z_max;
                        k=k+1;
                        indice(q,k)=i;
```

```
            else
                mis(q,k)=r1-d1;
                k=k+1;
                indice(q,k)=i;
            end
        else
            mis(q,k)=z_max;
            k=k+1;
            indice(q,k)=i;
        end
    end
end
elseif m2(i)~=0 & (m2(i)~=inf | m2(i)~=-inf) & (m1(q)~=m2(i) | m1(q)~=-m2(i))
    q1=pos_rob(o,2)-m1(q)*pos_rob(o,1);
    q2=g_y(i,2)-m2(i)*g_y(i,1);
    xp=(q2-q1)/(m1(q)-m2(i));
    yp=m2(i)*xp+q2;
    if xp>pos_rob(o,1)+d1*cos(theta_ass(q))
        r1=sqrt((pos_rob(o,1)-xp)^2+(pos_rob(o,2)-yp)^2);
        if r1>d1
            if yp<=(max(g_y(i,2),g_y(i+1,2))+sic) & yp>=(min(g_y(i,2),g_y(i+1,2))-sic)
                if r1>=z_max+d1
                    mis(q,k)=z_max;
                    k=k+1;
                    indice(q,k)=i;
                else
                    mis(q,k)=r1-d1;
                    k=k+1;
                    indice(q,k)=i;
                end
            else
                mis(q,k)=z_max;
                k=k+1;
                indice(q,k)=i;
            end
        end
    end
```

```
        elseif m2(i)==0
            q1=pos_rob(o,2)-m1(q)*pos_rob(o,1);
            yp=g_y(i,2);
            xp=(yp-q1)/m1(q);
            if xp>pos_rob(o,1)+d1*cos(theta_ass(q))
                r1=sqrt((pos_rob(o,1)-xp)^2+(pos_rob(o,2)-yp)^2);
                if r1>d1
                    if xp<=(max(g_y(i,1),g_y(i+1,1))+sic) & xp>=(min(g_y(i,1),g_y(i+1,1))-sic)
                        if r1>=z_max+d1
                            mis(q,k)=z_max;
                            k=k+1;
                            indice(q,k)=i;
                        else
                            mis(q,k)=r1-d1;
                            k=k+1;
                            indice(q,k)=i;
                        end
                    else
                        mis(q,k)=z_max;
                        k=k+1;
                        indice(q,k)=i;
                    end
                end
            end
        elseif (theta_ass(q)<deg2rad(270) | theta_ass(q)>deg2rad(90)) &
theta_ass(q)~=deg2rad(180)
            if m2(i)==inf | m2(i)==-inf
                q1=pos_rob(o,2)-m1(q)*pos_rob(o,1);
                xp=g_y(i,1);
                yp=m1(q)*xp+q1;
                if xp<pos_rob(o,1)-d1*cos(theta_ass(q))
                    r1=sqrt((pos_rob(o,1)-xp)^2+(pos_rob(o,2)-yp)^2);
                    if r1>d1
                        if yp<=(max(g_y(i,2),g_y(i+1,2))+sic) & yp>=(min(g_y(i,2),g_y(i+1,2))-sic)
                            if r1>=z_max+d1
                                mis(q,k)=z_max;
```

```
                k=k+1;
                indice(q,k)=i;
            else
                mis(q,k)=r1-d1;
                k=k+1;
                indice(q,k)=i;
            end
        else
            mis(q,k)=z_max;
            k=k+1;
            indice(q,k)=i;
        end
    end
end
elseif m2(i)~=0 & (m2(i)~=inf | m2(i)~=-inf) & m1(q)~=m2(i) & m1(q)~=-m2(i)
    q1=pos_rob(o,2)-m1(q)*pos_rob(o,1);
    q2=g_y(i,2)-m2(i)*g_y(i,1);
    xp=(q2-q1)/(m1(q)-m2(i));
    yp=m2(i)*xp+q2;
    if xp<pos_rob(o,1)-d1*cos(theta_ass(q))
        r1=sqrt((pos_rob(o,1)-xp)^2+(pos_rob(o,2)-yp)^2);
        if r1>d1
            if yp<=(max(g_y(i,2),g_y(i+1,2))+sic) & yp>=(min(g_y(i,2),g_y(i+1,2))-sic)
                if r1>=z_max+d1
                    mis(q,k)=z_max;
                    k=k+1;
                    indice(q,k)=i;
                else
                    mis(q,k)=r1-d1;
                    k=k+1;
                    indice(q,k)=i;
                end
            else
                mis(q,k)=z_max;
                k=k+1;
                indice(q,k)=i;
            end
```

```
            end
        end
    elseif m2(i)==0
        yp=g_y(i,2);
        q1=pos_rob(o,2)-m1(q)*pos_rob(o,1);
        xp=(yp-q1)/m1(q);
        if xp<pos_rob(o,1)-d1*cos(theta_ass(q))
            r1=sqrt((pos_rob(o,1)-xp)^2+(pos_rob(o,2)-yp)^2);
            if r1>d1
                if xp<=(max(g_y(i,1),g_y(i+1,1))+sic) & xp>=(min(g_y(i,1),g_y(i+1,1))-
     sic)
                if r1>=z_max+d1
                    mis(q,k)=z_max;
                    k=k+1;
                    indice(q,k)=i;
                else
                    mis(q,k)=r1-d1;
                    k=k+1;
                end
            else
                mis(q,k)=z_max;
                k=k+1;
                indice(q,k)=i;
            end
        end
    end
  end

  end
end
for i=1:8
  for j=1:size(mis,2)
    if( mis(i,j) == 0)
      mis(i,j)=+inf;
    end
  end
```

```
    if min(mis(i,:))==z_max
    mis_2(i)=(min(mis(i,:)));
    else
    mis_2(i)=(min(mis(i,:))+randn(1)*var);
    end
end
end
% algoritmo di occupancy grid mapping
function [mappa1,tot_tempi] = occ_grid_map()
global z_max pos_rob d1 theta_robot g_cel g_alpha g_beta l0 locc lfree x_max y_max x
dim g_y iter x2_max y2_max var
%inizializzazione mappa con probabilità locc
for o=1:iter
mappa(1:(y2_max)/g_cel,1:(x2_max)/g_cel,o)=l0;
end
for o=1:iter
    t_cpu=cputime;
    [a,b,c]=sim_sen(o);
    % calcolo dei centri delle celle
    for i=1:(x2_max/g_cel)
        xc(i)=(x(1,1)+(i+(i-1))*g_cel)/2; % ascissa dei centri di massa delle celle
        for t=1:(y2_max/g_cel)
            yc(t)=(x(1,2)+(t+(t-1))*g_cel)/2; % ordinata dei centro di massa delle celle
            %DISTANZA DEL ROROT DAI CENTRI DI MASSA MC
            r(t,i)=sqrt((xc(i)-pos_rob(o,1))^2+(yc(t)-pos_rob(o,2))^2); % distanza centro-robot
            phi(t,i)=atan2((yc(t)-pos_rob(o,2)),(xc(i)-pos_rob(o,1)))-theta_robot; %sfasamento
centro-robot
            for q=1:8
                if phi(t,i)<-pi/8
                    phi(t,i)=phi(t,i)+deg2rad(360);
                    v_sfas(q)=abs(phi(t,i)-b(q)); % sfasamento centro--q-iesimo sensore
                else
                    v_sfas(q)=abs(phi(t,i)-b(q)); % sfasamento centro--q-iesimo sensore
                end
            end
            [f,k]=min(v_sfas);
            % condizione di appartenenza
```

```
            if r(t,i)<=z_max & f<=g_beta/2
                if r(t,i)>min([z_max, a(1,k)+g_alpha/2]) | (abs(phi(t,i)-b(1,k))> g_beta/2)
                    lt=l0;
                elseif a(1,k)<z_max & abs(r(t,i)-a(1,k))<g_alpha/2 % (a(1,k)-r<0) % cambiato
condizione  abs(r-a(1,k))<g_alpha/2
                    lt=locc;
                elseif r(t,i)<=a(1,k)
                    lt=lfree;
                end
                mappa(t,i,o)=mappa(t,i,o)+lt-l0;
            end
            if o>1
                mappan(t,i)=mappan(t,i)+mappa(t,i,o);

            else
                mappan(t,i)=mappa(t,i,o);
            end



        end
    end

    mappa1=mappan(size(mappan,1):-1:1,:);
    figure(2)
    grid on
    % colordef black
    [X,Y]=meshgrid(0:g_cel:x2_max-g_cel,0:g_cel:y2_max-g_cel);
    surf(X,Y,-mappan)
    t_cpu=cputime-t_cpu;
    tempi_cpu(o)=t_cpu;
    title(sprintf('%d di %d (%f sec)',o,iter,t_cpu))
    colormap(gray)
end
figure (3)
hold on
j=1:iter;
plot(j,tempi_cpu) % plot del tempo impiegato per iterazione
```

```
tempi_cpu
tot_tempi=sum(tempi_cpu);% totale tempo computazione per simulaizione
med_temp=tot_tempi/iter;% tempo medio per simulazione
plot(j,med_temp)
end
```

RIFERIMENTI BIBLIOGRAFICI

Sebastian THRUN, Wolfram BURGARD, Dieter FOX PROBABILISTIC ROBOTICS 2006

J. Borenstein, B. Everett, and L. Feng. Navigating Mobile Robots: Systems and Techniques. A. K. Peters, Ltd., Wellesley, MA, 1996.

H. Choset. Sensor Based Motion Planning: The Hierarchical Generalized

Voronoi Graph. PhD thesis, California Institute of Technology, 1996.

M. Csorba. Simultaneous Localization and Map Building. PhD thesis, Department of Engineering Science, University of Oxford, Oxford, UK, 1997.

Romano Scozzafava PROBABILITA' E STATISTICA