Esplorazione multirobot

Francesco Argentieri*
Università di Trento
francesco.argentieri@studenti.unitn.it

Giacomo Mazzaglia†
Università di Trento
giacomo.mazzaglia@studenti.unitn.it

Sommario

In questo lavoro, consideriamo il problema di esplorare un ambiente sconosciuto con un team di robot. Come nell'esplorazione di robot singoli, l'obiettivo è di ridurre al minimo il tempo di esplorazione complessivo. Il problema chiave da risolvere nel contesto di robot multipli è quello di scegliere i punti di destinazione appropriati per i singoli robot in modo che possano esplorare contemporaneamente diverse regioni dell'ambiente. Presentiamo un approccio per il coordinamento di più robot, che tiene conto simultaneamente del costo di raggiungere un punto target e della sua utilità. Descriviamo inoltre come il nostro algoritmo può essere esteso a situazioni in cui il raggio di comunicazione dei robot è limitato. Per la stima delle posizioni dei robot è stato utilizzato il filtro particellare, assumendo una comunicazione con delle ancore WI-FI. *I risultati dimostrano che la nostra tecnica distribuisce efficacemente i robot sull'ambiente e consente loro di compiere rapidamente la loro missione.*

I. Introduzione

L'ESPLORAZIONE efficiente di ambienti sconosciuti è un problema fondamentale nella robotica mobile. L'estensione alla esplorazione di più robot pone diverse nuove sfide, tra cui: coordinamento di robot, integrazione delle informazioni raccolte dai robot in una mappa coerente e comunicazione limitata. Il coordinamento di un sistema di più robot è la base per un'efficiente implementazione dell'esplorazione distribuita. Difficoltà maggiori nel campo del coordinamento vengono dalla conoscenza assunta dal robot sul comportamento degli altri robot. Se i robot conoscono le loro posizioni relative e condividono una mappa della zone che hanno esplorato finora, si può raggiungere un coordinamento efficace, guidando questi in zone inesplorate dell'ambiente. Questo può essere fatto assegnando ai robot il compito di raggiungere un punto di arrivo preso dalla frontiera fornita dalla scansione del LIDAR[1].[1] Per avere un assegnamento efficace è importante che i robot vadano a condividere i punti comuni all'interno delle loro frontiere e che non cerchino di raggiungere lo stesso punto. Per la stima della posizione viene adottato un filtro particellare, i robot comunicando con delle ancore Wi-Fi riescono a conoscere, con un dato errore, la loro posizione ricoperta in quell'istante. Per il raggiungimento della zona target prefissata viene invece adottata la funzione potenziale. I robot contengono al loro interno memoria delle zone già esplorate e non, ipotizzando di conoscere in anticipo le dimensioni massime della mappa che andranno ad esplorare, questi ultimi costruiscono, attraverso l'integrazione di successive matrici di occupazione locale, una matrice di occupazione globale che rappresenta la mappa dell'ambiente. Saranno successivamente presentati la modellazione del sistema e i successivi risultati ottenuti.

II. Comunicazione

All'interno del sistema realizzato i robot sono in comunicazione tramite rete Wi-Fi ideale

*ID: 183892

†ID: 183382

[1]LIDAR (acronimo dall'inglese Light Detection and Ranging o Laser Imaging Detection and Ranging) è una tecnica di telerilevamento che permette di determinare la distanza di un oggetto o di una superficie utilizzando un impulso laser.

senza degradazione di segnale e conseguente perdita di informazione. Il primo tipo di comunicazione è quella disponibile tra robot che permette in un range limitato l'assegnazione e la coordinazione efficiente della destinazione per lo svolgimento della missione. La seconda tipologia di comunicazione riguarda la stima della posizione dei robot mediante *particle filter* per la localizzazione rispetto hotspots Wi-Fi intesi, in questo caso, come punti di riferimento virtuali. Generati casualmente o tramite input dell'utente all'interno della mappa.

III. Modello del sistema

i. Modello cinematico

Il robot è basato sul modello dell'uniciclo a trazione differenziale, la configurazione è completamente descritta da $\mathbf{q} = [x\, y\, \theta]^T$, dove (x, y) sono le coordinate cartesiane del punto di contatto con il suolo e θ è l'orientamento della ruota rispetto all'asse x.[2] Il modello cinematico dell'uniciclo è descritto dalle equazioni (1):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega \qquad (1)$$

Il robot ha le dimensioni riportate in tabella 1.

Tabella 1: Riepilogo dimensioni

dimensioni		
raggio ruote	[m]	0.07
interasse	[m]	0.30

Questo è equipaggiato con un sensore virtuale LIDAR, basato sul modello Hokuyo URG-04LX, collocato al centro della struttura in modo tale da evitare errori di offset, di seguito se ne riportano le caratteristiche, di cui adattate ad hoc per la simulazione. Come sensori propriocettivi presenta due encoder incrementali virtuali calettati sull'asse delle ruote, le caratteristiche di entrambi sono riportate in tabella 2. Una rappresentazione del robot è osservabile in figura 1.

Tabella 2: Specifiche sensori

specifiche lidar virtuale		
risoluzione angolare	[°]	0.36
angolo di scansione	[°]	180.00
massima distanza	[m]	4.00
minima distanza	[m]	0.02
risoluzione	[mm]	1.00
specifiche encoder virtuale		
risoluzione	$2 \cdot (\frac{\pi}{2600})$	

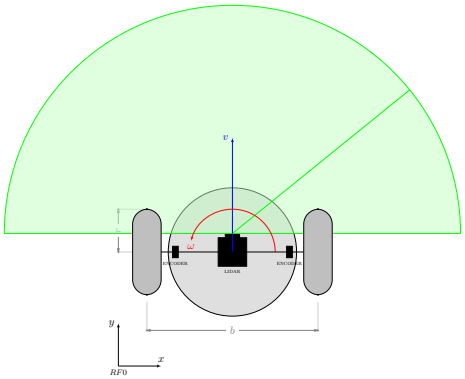


Figura 1: modello cinematico

ii. Modello del sensore ed Occupacy grid

Gli approcci basati sulle occupacy grid tendono a non tener conto della geometria dell'ambiente, ma ad assumere una visione più sensoriale. L'ambiente viene discretizzato tramite una griglia di celle quadrate; ogni cella $c(i; j)$ ha un numero associato ad essa corrispondente ad una probabilità di occupazione $p_{\text{occ}}(i; j)$ e una probabilità di essere vuoto $p_{\text{emp}}(i; j)$. Celle parzialmente occupate non verranno prese in considerazione.

a. Occupacy grid locale

Per un determinato sensore di rilevamento della distanza, come uno scanner per linee laser, con portata massima R e mezza larghezza del raggio del sensore β, il modello può essere scomposto in un numero di settori etichettati I-IV[3], come illustrato nella Figura 2.
La regione I è la regione associata alla lettura effettiva del laser, la regione II rappresenta

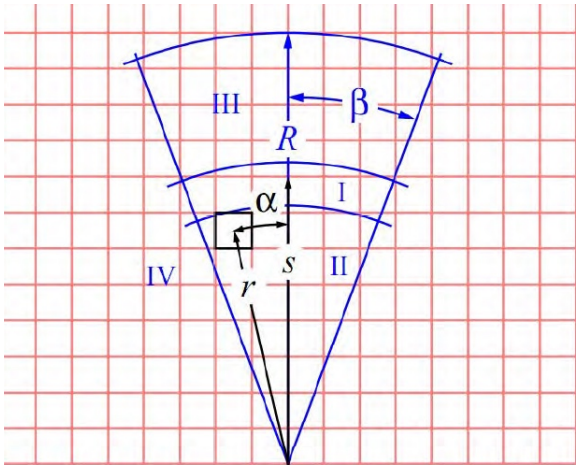


Figura 2: Schema delle scansione

un'area vuota in cui nulla viene rilevato, la regione III è l'area coperta dal raggio laser, ma rimane sconosciuta se occupata o meno a causa dell'occlusione, e la regione IV è al di fuori del campo di visibilità del LIDAR. Parametri rilevanti della cella evidenziata (contorno nero) includono: r che è la distanza dell'elemento di griglia dalla posizione del sensore; α l'angolo dell'elemento di griglia relativo al raggio centrale del sensore. Per una misurazione s che rientri nella regione I, la probabilità che la misura sia effettivamente dovuta alla presenza di un ostacolo in tale intervallo può essere calcolata dall'equazione (2):

$$P(s|Occupied) = \frac{\frac{R-r}{R} + \frac{\beta-\alpha}{\beta}\beta}{2} \cdot \mathrm{Max}_{\mathrm{occupied}} \tag{2}$$

dove $\mathrm{Max}_{\mathrm{occupied}}$, nell'eq.(2), è dovuto all'assunzione che una lettura di "Occupato" non è mai completamente affidabile.

b. Occupacy grid globale

Ogni robot è dotato di memoria dove viene salvata la mappatura globale dell'intero ambiente. Questa, è definita occupacy grid globale, ed è ricostruita mediante rototraslazione e sovrapposizione delle occupacy grid locali fornite in diversi istanti dal robot. Alla fine della simulazione le varie occupacy grid globali di ogni robot vengono fuse in un'unica mappa. Un problema per i metodi basati sulla griglia di occupazione è l'utilizzo eccessivo della memoria, infatti diventa oneroso per la ricostruzione di mappe 3D. Un secondo problema è legato all'allineamento della griglia, assunto ideale, in quanto è probabile che le celle: coprano aree difficili da raggiungere e zone parzialmente piene dove non è possibile catturare le acute discontinuità ai bordi degli oggetti a causa di una risoluzione non sufficiente.

IV. Soluzione Proposta

i. Genarazione Procedurale

Si è realizzato un ambiente virtuale 2D atto a ricostruire la pianta di un edificio (uffici, capannoni, aule), allo scopo è stata usata una generazione procedurale. Il vantaggio più evidente dei livelli generati proceduralmente è la loro varietà che portano ad ogni esecuzione, un ambiente diverso dal precedente. Ciò significa che i robot non possono apprendere le posizioni di oggetti e questo permette di testare l'affidabilità in casi sempre differenti. Un altro vantaggio comune a tutte le implementazioni della generazione procedurale è il tempo che si risparmia nello sviluppo. Tra gli svantaggi, si ricorda, che la generazione procedurale di per sé non è in alcun modo casuale. L'assenza di controllo è una mancanza comune della generazione procedurale in generale. Dato che, di solito gli ambienti sono realizzati a mano da designer, lasciare questo lavoro a un algoritmo si traduce in una significativa perdita di controllo. Un'altra considerazione da fare è la potenza di calcolo richiesta, nel nostro caso, si ha solo una matrice 2D di piccole dimensioni che deve essere generata. Tuttavia, mappe di più larga scala, avranno un costo computazionale che diventa più significativo e deve essere preso in considerazione.[4]

a. Binary Space Partitioning

Il partizionamento dello spazio binario è un processo generico di divisione ricorsiva di una scena in due finché il partizionamento soddisfa uno o più requisiti. Può essere visto come una generalizzazione di altre strutture ad albero spaziale, uno in cui gli iperpiani che

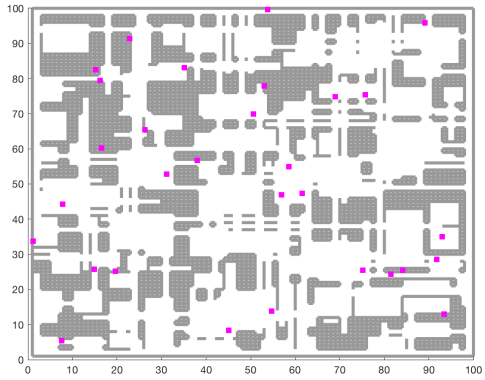


Figura 3: scenario procedurale 100×100

suddividono lo spazio possono avere qualsiasi orientamento, piuttosto che essere allineati con gli assi delle coordinate.[5] L'albero ***k-d*** è un albero binario in cui ogni nodo è un punto k-dimensionale. Ogni nodo non foglia può essere pensato come generatore implicito di un iperpiano scisso che divide lo spazio in due parti, note come semispazi. I punti a sinistra di questo iperpiano sono rappresentati dal sotto-albero sinistro di quel nodo e i punti a destra dell'iperpiano sono rappresentati dal sotto-albero destro, come in figura 5. La direzione dell'iperpiano viene scelta nel modo seguente: ogni nodo dell'albero è associato a una delle dimensioni k, con l'iperpiano perpendicolare all'asse di quella dimensione.[6]

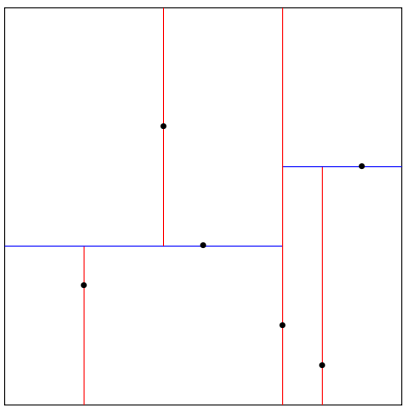


Figura 4: decomposizione per il set di punti.

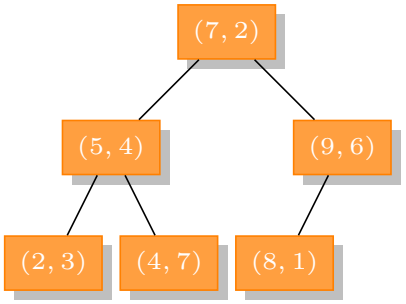


Figura 5: risultato dell'albero k-d.

ii. Allocazione destinazioni ed Utility

L'obiettivo di un processo di esplorazione è quello di coprire l'intero ambiente nel minor tempo possibile. Pertanto, è essenziale che i robot tengano traccia di quali aree dell'ambiente sono già state esplorate. Inoltre, i robot devono costruire una mappa globale per pianificare i loro percorsi e coordinare le loro azioni. Quando esploriamo un ambiente sconosciuto, siamo particolarmente interessati alle "celle di frontiera". Con cella di frontiera, denotiamo ciascuna cella già esplorata che è un vicino immediato di un cella sconosciuta, inesplorata. Se indirizziamo un robot a una tale cella, ci si può aspettare che ottenga informazioni sull'area inesplorata quando arriva alla sua posizione di destinazione. Il fatto che una mappa in generale contiene diverse aree inesplorate solleva il problema di come assegnare in modo efficiente le destinazioni a più robot. Se sono coinvolti più robot, vogliamo evitare che molti di loro si spostino nella stessa posizione. Per affrontare questi problemi e determinare i luoghi di destinazione appropriati per i singoli robot, il nostro sistema utilizza il seguente approccio: considera contemporaneamente il costo del raggiungimento di una cella di frontiera e l'utilità di questa. Per ogni robot, il costo per raggiungere la cella è proporzionale alla distanza tra se stesso e quest'ultima, d'altra parte l'utilità di una cella di frontiera dipende invece dal numero di robot che si stanno spostando in quella direzione o in posizioni limitrofe.[7] Di seguito vengono riportate le

equazioni implementate:

$$U(t_n|t_1,\ldots,t_{n-1}) = U_{t_n} - \sum_{i=1}^{n-1} P(||t_n - t_i||) \tag{3}$$

$$P(d) = \begin{cases} 1 - \frac{d}{maxrange} & \text{se } d < \text{maxrange} \\ 0 & \text{altrimenti} \end{cases} \tag{4}$$

$$(i,t) = \text{argmax}_{i^\iota,t^\iota}(U_{t^\iota} - \beta V_{t^\iota}^{i^\iota}) \tag{5}$$

La prime due equazioni (3–4), calcolano l'utility per il robot n-esimo mentre l'ultima equazione rappresenta il caso esteso all'integrazione di più robot che comunicano tra loro. *Maxrange* rappresenta la massima visibilità fornita dal LIDAR, d rappresenta la distanza dei target assegnati ai vari robot, U la funzione di utility e V la funzione di costo, quest'ultima tiene conto della distanza di ogni robot dal suo target. Le i e t, nell'eq. (5), forniscono per ogni robot il suo target i-esimo ottimale. Il modo in cui questi target da parte di ogni robot vengono raggiunti è lasciato alla funzione potenziale che verrà descritta in seguito.

iii. Pianificazione

La pianificazione del percorso per i robot è uno dei criteri importanti da prendere in considerazione per migliorare il livello di autonomia del robot. Nella pianificazione del percorso, la sicurezza è un problema importante che dovrebbe essere preso in considerazione al fine di garantire che un robot raggiunga la posizione target senza collisioni con gli ostacoli circostanti. Inoltre, ci sono aspetti importanti che devono essere affrontati nella pianificazione del percorso; tempo computazionale, percorso ottimale e completezza. Uno dei metodi più diffusi per la pianificazione dei percorsi è il metodo *Campi Potenziali Artificiali*. Il metodo del potenziale è in grado di superare uno scenario sconosciuto, tenendo conto della realtà dell'ambiente corrente e del movimento del robot. Due tipi di forze sono coinvolte nel metodo del campo potenziale; forza attrattiva generata da obiettivi e forza repulsiva generata da ostacoli; dalla differenza dei due campi il robot deve riprogrammare un nuovo percorso[8]. Utilizzando informazioni parziali sullo spazio di lavoro raccolte attraverso i sensori, quindi le informazioni sensoriali sono integrate in una mappa secondo un paradigma *sense—plan—move*. Oppure utilizzare le informazioni sensoriali impiegate per pianificare moti secondo un paradigma *stimulus—response* (navigazione reattiva). Il robot è considerato come un punto materiale sotto l'influenza dei campi prodotti da obiettivi e ostacoli nello spazio di ricerca. Le forze repulsive sono generate da ostacoli mentre la forza attrattiva è generata dagli obiettivi. La forza risultante (la somma di tutte le forze) dei campi sul robot viene utilizzato per determinare la direzione del movimento e la velocità di spostamento evitando collisioni[9]. Tuttavia esistono svantaggi quali: **a)** situazione di stallo dovuta ai minimi locali; **b)** oscillazione in presenza di ostacoli; **c)** nessun passaggio tra ostacoli ravvicinati; **d)** oscillazioni in passaggi stretti[10]. Il robot viene considerato come punto $\mathbf{q} = (x\,y)^T$, in un piano cartesiano, attratto (potenziale U_{att}) dal punto obiettivo $\mathbf{q}_g$ e respinto (potenziale U_{rep}) dagli ostacoli.

$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q) \tag{6}$$

dove $U(q)$ potenziale artificiale; $U_{\text{att}}(q)$ campo attrattivo; $U_{\text{rep}}(q)$ campo repulsivo. La pianificazione avviene in modo incrementale: ad ogni configurazione $\mathbf{q}$, la forza artificiale viene generata come nell'equazione (7):

$$\begin{aligned} F(q) &= -\nabla U(q) \\ &= -\nabla U_{\text{att}}(q) - U_{\text{rep}}(q) \\ F(q) &= F_{\text{att}}(q) + F_{\text{rep}}(q) \end{aligned} \tag{7}$$

dove $F(q)$: forza artificiale; $F_{\text{att}}(q)$: forza attrattiva; $F_{\text{rep}}(q)$: forza repulsiva. Il campo potenziale U_{att}) tra robot e obiettivo viene descritto dall'eq. (8) per trascinare il robot nell'area obiettivo.

$$\begin{aligned} U_{\text{att}}(q) &= \frac{1}{2}\, k_a\, (q - q_d)^2 \\ &= \frac{1}{2}\, k_a\, \rho_{goal}^2(q) \end{aligned} \tag{8}$$

dove k_a: coefficiente positivo per APF[2]; q: posizione corrente del robot; q_d: posizione corrente dell'obiettivo.

$$\rho_{\text{goal}}(q) = \|q - q_d\| \tag{9}$$

è una distanza euclidea dalla posizione del robot alla posizione dell'obiettivo. La forza attrattiva del robot è calcolata come gradiente negativo del potenziale campo[11]:

$$\begin{aligned} F_{\text{att}}(q) &= -\frac{1}{2} k_a \rho^2_{\text{goal}}(q) \\ F_{\text{att}}(q) &= -k_a \left(q - q_d\right) \end{aligned} \tag{10}$$

$F_{\text{att}}(q)$, nell'eq. (10), è un vettore diretto verso q_{d} con intensità linearmente proporzionale alla distanza da q a q_{d}. Può essere scritto nelle sue componenti:

$$\begin{aligned} F_{\text{att}} - x(q) &= -k_a \left(x - x_d\right) \\ F_{\text{att}} - y(q) &= -k_a \left(y - y_d\right) \end{aligned} \tag{11}$$

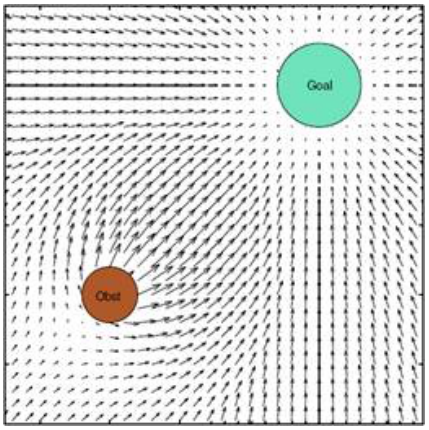
Le equazioni (11) sono la forza attrattiva nelle direzioni x e y. Nella funzione potenziale, il robot deve essere respinto dagli ostacoli, ma se lontano da questi, il movimento non risente della loro influenza. La funzione potenziale di repulsione (12) è:

$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2} k_b (\frac{1}{d(q)} - \frac{1}{d_0})^2 & \text{se } d(q) \leq d_0 \\ 0 & \text{se } d(q) > d_0 \end{cases} \tag{12}$$

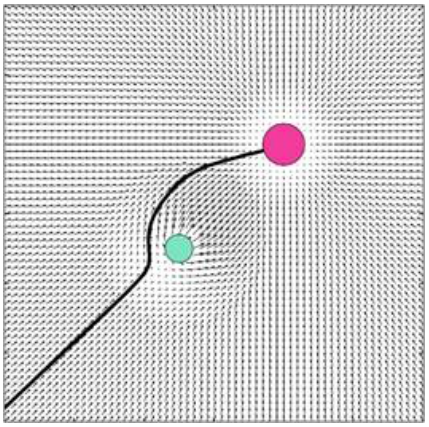
iv. Algoritmi di stima

Il sistema robotico preso in esame presenta degli errori all'ingresso degli encoder, tale errore porta il sistema a manifestare nel lungo periodo il noto comportamento di *drift*. Nasce allora il problema di correggere e stimare la posizione in base ad un dato modello. Per stimare la posizione del robot utilizzando la ricostruzione odometrica si ottiene un risultato affetto da incertezza crescente all'aumentare dello spostamento. È necessario quindi correggere tale stima attraverso il particle filter[12] analizzato di seguito. Ciò su cui ci appoggeremo per avere una stima della posizione assunta dal robot, saranno dei sensori esterni ad esso, nel nostro caso delle ancore Wi-Fi virtualmente simulate.

[2]Artificial Potential Field



(a) *Campo attrattivo - repulsivo.*



(b) *Pianificazione traiettoria.*

Figura 6: Potenziali artificiali

a. Filtro Particellare

L'algoritmo di localizzazione PF procede come segue in figura 7. Si inizializzano n particelle in una mappa. Ogni particella è un vettore di stato $q = [x\, y\, \theta]^T$ del veicolo, ad ognuna di queste si applica il modello descritto dal sistema di eq.(1): e si aggiunge un rumore al vettore di controllo $u = [v\, \omega]^T$. Successivamente per ogni particella si prevede la possibile misura di posizione ottenuta e si confronta con l'osservazione reale rispetto all'ancora Wi-Fi, tale confronto porterà al calcolo dell'innovazione o di ciò che definiremo peso della particella. Si selezionano le particelle che meglio spiegano l'osservazione, un modo per farlo è quello di

costruire una pdf che descriva i campioni e i loro pesi, e poi riselezionare un nuovo set di particelle da questa pdf. La stima della posizione del robot fornita dal filtro è la media di questo nuovo ricampionamento. Il punto cruciale è che non richiede alcuna ipotesi di linearizzazione (non ci sono jacobiani coinvolti) e non ci sono ipotesi Gaussiane. È particolarmente adatto ai problemi con piccoli spazi di stato mentre in caso di vettori di stato grandi diventa computazionalmente pesante.

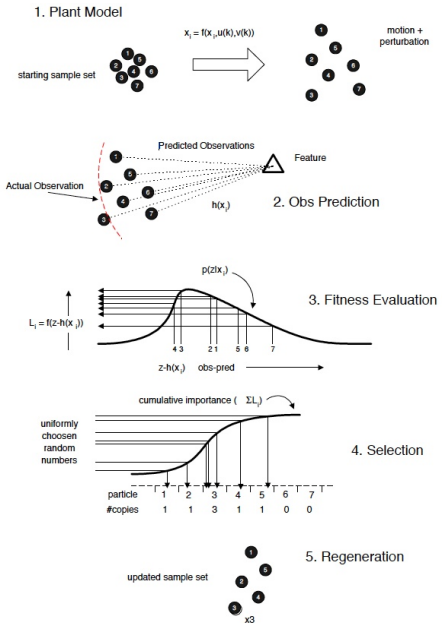


Figura 7: Filtro Particellare

V. Implementazione

Per la realizzazione del progetto si è optato per una simulazione con la suite MATLAB, realizzando un software basato su un'architettura a oggetti e funzionale per i motivi indicati di seguito. La programmazione orientata agli oggetti è un paradigma di programmazione che permette di definire oggetti software in grado di interagire gli uni con gli altri attraverso lo scambio di messaggi. È particolarmente adatta nei contesti in cui si possono definire delle relazioni di interdipendenza tra i concetti da modellare. Tra gli altri vantaggi della programmazione orientata agli oggetti: **a)** fornisce un supporto naturale alla modellazione software degli oggetti del mondo reale o del modello astratto da riprodurre; **b)** permette una più facile gestione e manutenzione di progetti di grandi dimensioni; **c)** l'organizzazione del codice sotto forma di classi favorisce la modularità e il riuso di codice. Il paradigma OOP suggerisce un principio noto come *information hiding* che indica che si debba accedere agli attributi dell'istanza solo tramite metodi invocati su quello stesso oggetto. L'incapsulamento è la proprietà per cui i dati che definiscono lo stato interno di un oggetto e i metodi che ne definiscono la logica sono accessibili ai metodi dell'oggetto stesso, mentre non sono visibili ai client. Per alterare lo stato interno dell'oggetto, è necessario invocarne i metodi pubblici, ed è questo lo scopo principale dell'incapsulamento. Infatti, se gestito opportunamente, esso permette di vedere l'oggetto come una black-box, con la quale l'interazione avviene solo e solamente tramite i metodi definiti dall'interfaccia. Il punto è dare delle funzionalità agli utenti nascondendo i dettagli legati alla loro implementazione.[13]

i. Classe Map

La classe Map gestisce la generazione di una nuova mappa di dimensioni variabili o il caricamento di una precedentemente salvata. Nel costruttore è possibile specificare come primo argomento se si desidera una nuova mappa o una esistente mediante stringa "new" o "load", fornendo ulteriori parametri in ingresso come la dimensione della mappa desiderata, così si avvia il processo di generazione procedurale visto nella sez.(IV.i). È possibile specificare se si desiderano più punti di riferimento virtuali oltre quelli impostati di default. Con la stringa "load", invece è possibile richiamare una mappa esistente da GUI. All'interno della classe sono salvati, come vettori, i punti che costituiscono la mappa e le linee che li collegano. Questi sono interpretati dalla funzione che simula il processo di scansione del LIDAR. Per mappe di grandi dimensioni si estraggono i punti liberi che non siano collegati a linee. Questo permette di fornire al robot le condizioni iniziali

all'avvio della simulazione evitando che venga posizionato all'interno di qualche parete.

ii. Classe Robot

La classe Robot definita nel progetto permette di generare uno o più istanze di questo oggetto permettendo così di avere un comportamento univoco. Tali istanze condividono le stesse proprietà non modificabili dall'esterno, infatti sono messi a disposizione dell'utente metodi per l'accesso e la modifica di queste. Questo permette di avere robot che condividono, una volta configurati, gli stessi parametri costanti i quali sono: **a)** geometrici; **b)** risoluzione dei sensori; **c)** memoria di massa riservata per le scansioni; **d)** memoria di massa riservata per occupacy grid. Tutti i paramenti salvati all'interno della classe sono accessibili solo in modalità lettura dall'esterno. Infatti una volta create una o più istanze del robot queste sono posizionate all'interno della mappa di cui non conoscono nulla se non quello che percepiscono tramite LIDAR ogni 10 Hz. Successivamente il robot si muoverà in direzione delle coordinate obiettivo pianificando il percorso di volta in volta secondo le informazioni ricevute dalla funzione potenziali artificiali, vista precedentemente nella sez. IV.iii.

iii. Classe Particle Filter

La classe Particle Filter mette a disposizione due funzioni pubbliche: **a)** il costruttore di classe; **b)** il metodo *update*. Il costruttore è utilizzato per generare un'istanza di questo oggetto legata all'altra classe Robot da cui riceve le informazioni relative alle caratteristiche dei sensori e il vettore velocità. Inoltre la classe Particle Filter riceve dall'altra classe Map, i, le posizioni dei punti di riferimento virtuali precedentemente generati. Con il metodo *update* invece viene aggiornata la stima della posizione del robot ad ogni iterazione per tutta la lunghezza della simulazione.

iv. Esecuzione

Una volta generato lo scenario la simulazione inizia con il posizionamento di uno o più robot all'interno della mappa come condizione iniziale e assegnato un obiettivo da raggiungere. Questi non conosco l'ambiente in cui si trovano e quindi tentano di raggiungere l'obiettivo nella maniera più rapida impostando un traiettoria rettilinea determinata come norma tra due punti. La pianificazione della traiettoria viene modificata in base alle rilevazioni effettuate dal LIDAR in tal modo essi evitano l'ostacolo perché repulsi. Per la creazione delle matrici di occupazione locali vengono utilizzate le misure fornite dal LIDAR, tali misure vengono riproiettate nel sistema di riferimento del robot e registrate all'interno di una matrice utilizzando come peso per ogni misura l'equazione fornita da (2). Queste verranno integrate in una matrice globale che sarà aggiornata ogni secondo. Il passaggio da matrice locale a matrice globale avviene rototraslando rispetto la posizione stimata del robot fornita dal particle filter e registrata all'interno del robot. La matrice globale del robot si assume essere già dimensionata in base alla mappa da esplorare. Il robot durante la sua esplorazione è programmato per affiancare il primo muro visibile e percorrerlo, nel caso in cui durante questa operazione si dovessero presentare zone già visitate, esso cercherà di raggiungere punti non ancora esplorati all'interno della stanza, questa operazione viene condotta registrando le zone già esplorate dal robot all'interno di una seconda matrice assunta anch'essa di una dimensione tale da permettere di contenere la stanza con la risoluzione voluta. Il robot quando risulterà essere nel raggio di comunicazione con altri robot avvierà il meccanismo di Utility, in base al quale sarà assegnato ad ogni robot comunicante un nuovo target da raggiungere che massimizzi l'utilità, calcolata con l'equazione (5). Per evitare la ricomunicazione tra robot è stato inserito un delay che deve trascorrere tra una comunicazione e l'altra prima che i due robot possano comunicare nuovamente tra loro. La comunicazione come già detto risulta essere ideale senza degrada-

zione di segnale. Durante la comunicazione i robot condivideranno la mappa contenente le zone già visitate in modo da non permettere il passaggio in zone già viste da altri robot.

VI. Risultati

In questa sezione si valuteranno le performance dell'algoritmo. Per condurre questa analisi si è generata una mappa di dimensioni note, come in figura 8, si sono eseguiti dei test in diverse configurazioni di tempo (da 200 s a 1800 s) e numero di robot (da 1 a 4). All'in-

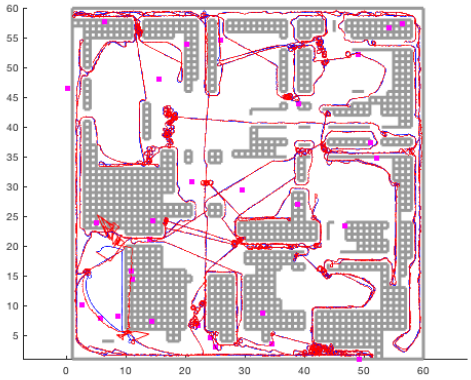


Figura 8: risulato simulazione mappa 60×60

terno della mappa di figura 8 sono riportate in blu le traiettorie compiute dai robot nel caso ideale mentre in rosso le traiettorie stimate. Le traiettorie riportate sono quelle ottenute da una simulazione di 1800 secondi e tramite l'utilizzo di tre robot che contemporaneamente la esplorano. La scelta di tale numero di robot e tempo di simulazione è stata compiuta per rispondere a due quesiti principali: efficienza energetica e percentuale di mappa esplorata. In figura 10 si può condurre il primo tipo di studio considerando che il percorrere una distanza maggiore comporta un maggiore dispendio energetico mentre in figura 9 si può notare come spesso l'adottare un numero maggiore di robot porti ad un'esplorazione più veloce. Osservando entrambi i grafici si può evidenziare che l'adozione di quattro robot rispetto a tre non comporti grossi benefici in termini di esplorazione percentuale della mappa, ricade allora sul numero di 3 robot la nostra scelta in quanto come evidenziato dal grafico 10 questo comporta un dispendio energetico minore. Mentre il tempo totale di 1800 secondi è stato scelto in quanto a tale lunghezza della simulazione si ottiene un valore ragionevole di mappa esplorata 70%. La percentuale di mappa esplorata è stata ottenuta considerando il numero totale di celle libere realmente a disposizione rispetto al numero di celle viste dal robot, come riportato nella equazione (13).

$$MappaEsplorata = \frac{\text{Celle}_{\text{Libere}}}{\text{Celle}_{\text{Visitate}}} \cdot 100\% \quad (13)$$

Figura 9: Percentuale mappa esplorata per # robot e tempo di simulazione

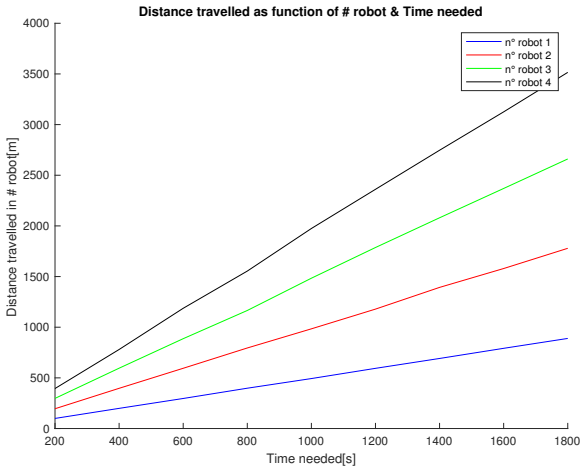


Figura 10: Distanza totale percorsa per # robot e tempo di simulazione

Si evidenzia nella tabella 3, il numero ottimo di robot e tempo di simulazione che permettono

Tabella 3: Esplorazione su mappa 60×60

Test	Robots [#]	Tempo [s]
⋮	⋮	⋮
9	1	1800
18	2	1800
27	3	1800
36	4	1800

di ottenere le mappe analizzate di seguito. In figura 11 viene riportata la mappa fornita dai tre robot in caso di esplorazione ideale dove il drift non è considerato da parte dei sistemi robotici. In figura 12 viene riportata la mappa fornita in caso di presenza di drift nel sistema e necessità di correzione dello stato dei robot tramite il particle filter.

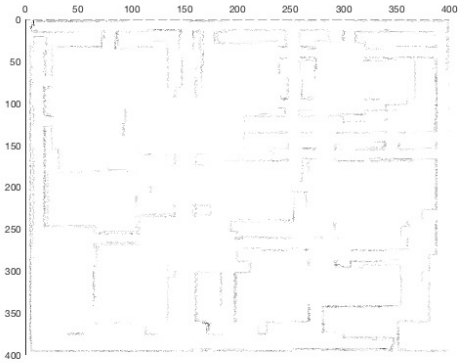


Figura 11: Mappa fornita dai robot con risoluzione di 0.15 [m] nel caso ideale

Si è deciso di riportare entrambe le mappe per comprendere quali sono gli effettivi limiti forniti dal particle filter o della creazione della occupacy grid globale. Un'altro metodo valutativo per il particle filter risulta essere quello del calcolo dell'errore quadratico medio fornito dal confronto tra le due traiettorie quelle stimate (in rosso) rispetto a quelle ideali (in blu) in figura 8. L'equazione 14 riporta il risultato che considereremo accettabile per la

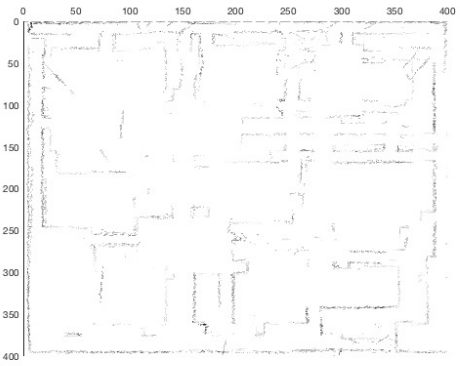


Figura 12: Mappa fornita dai robot con risoluzione di 0.15 [m] nel caso di utilizzo del particle filter

nostra simulazione.

$$RMSE = \sqrt{\frac{\sum_{k=1}^{k=N} (\hat{x}_k - x_k)^2}{N}} = 0.442\text{m} \tag{14}$$

VII. Conclusioni

È stato qui presentato un modo efficiente di condurre un'esplorazione coordinata di un ambiente. Possibile passo in avanti per la suddetta applicazione sarebbe la sua realizzazione ed esecuzione fisica in modo da condurre uno studio più approfondito sulle possibili problematiche riscontrabili. Un limite evidente è la pianificazione di traiettorie come visto negli svantaggi del metodo dei potenziali artificiali, nella sezione (IV.iii), poiché viene perso molto tempo nel modificare la traiettoria specialmente in prossimità degli ostacoli, in sezioni di passaggio, e in ambienti ciechi. Attualmente il robot si muove con velocità costante ponendo un freno allo spazio esplorabile, un modello cinematico migliore permetterebbe lo studio del cambio di velocità e accelerazione adattandole quando è necesario affrontare deviazioni e in presenza o meno di ostacoli. Inoltre si evidenzia che la generazione procedurale di mappe, vista nella sezione (IV.i), permette una generazione di casistiche ampia, ma poco controllabile e incapace di riprodurre strutture realmente esistenti nell'ambiente reale. Una possibile evo-

luzione sarebbe la ricostruzione di uno scenario 3D. Altro limite attuale dell'applicazione è il non riconoscimento di zone inaccessibili, ciò comporta nel caso di zone inaccessibili molto estese una gran probabilità di assegnazione di un target appartenente a tale zona per il robot. Altri esempi di possibili migliorie software apportabili, sono: una migliore rappresentazione fisica della comunicazione, robot-robot e robot-ancora Wi-Fi, tramite gli ultimi protocolli esistenti e l'adozione di algoritmi per una stima distribuita della posizione (least-square o di kalman distribuito).

Riferimenti bibliografici

[1] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*. ACM, 1998, pp. 47–53.

[2] B. Siciliano, *Robotica. Modellistica, pianificazione e controllo*, ser. Collana di istruzione scientifica. Serie di automatica. McGraw-Hill Companies, 2008. [Online]. Available: https://books.google.it/books?id=HdaVPQAACAAJ

[3] M. B. Ardhaoui, "Implementation of autonomous navigation and mapping using a laser line scanner on a tactical unmanned aerial vehicle," 2011.

[4] D. Green, *Procedural Content Generation for C++ Game Development*. Packt Publishing, 2016. [Online]. Available: https://books.google.it/books?id=EQYcDAAAQBAJ

[5] W. contributors, "Binary space partitioning — wikipedia, the free encyclopedia," 2018, [Online; accessed 14-March-2018]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Binary_space_partitioning&oldid=828843135

[6] ——, "K-d tree — wikipedia, the free encyclopedia," 2018, [Online; accessed 14-March-2018]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=K-d_tree&oldid=829846868

[7] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[8] C. K. M. C. K. A. N. H. Sabudin E. N, Omar. R, "Potential field methods and their inherent approaches for path planning," *ARPN Journal of Engineering and Applied Sciences*, vol. 11, no. 18, SEPTEMBER 2016.

[9] X. Xu, C. Li, and Y. Zhao, "Air traffic rerouting planning based on the improved artificial potential field model," in *2010 Chinese Control and Decision Conference*, May 2010, pp. 1444–1449.

[10] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, Apr 1991, pp. 1398–1404 vol.2.

[11] G. Li, A. Yamashita, H. Asama, and Y. Tamura, "An efficient improved artificial potential field based regression search method for robot path planning," in *2012 IEEE International Conference on Mechatronics and Automation*, Aug 2012, pp. 1227–1232.

[12] N. P. M., *C4B-Mobile Robotics*, 2003.

[13] B. Cox and A. Novobilski, *Object-oriented Programming: An Evolutionary Approach*, ser. Computer science. Addison-Wesley Publishing Company, 1991. [Online]. Available: https://books.google.it/books?id=U8AgAQAAIAAJ