Path planning locale per robot mobili basato su potenziali artificiali alternati

Contents

1 Introduzione

È innegabile che in questi anni si stia assistendo ad un aumento vertiginoso di sviluppo ed uso della robotica. È importante però evidenziare una distinzione tra due concetti apparentemente simili, ma per certi versi opposti, che caratterizzano due macro-categorie della robotica: automazione e autonomia. Il primo riguarda quei robot, tipicamente industriali, che operano in ambienti noti a priori ed eseguono in loop un compito predefinito; automatizzare, perciò, vuol dire sostituire l'essere umano in compiti ripetitivi e solitamente privi di eventi inaspettati. L'autonomia, invece, ben più complessa da realizzare, é caratteristica di quei sistemi che hanno un certo grado di inconsapevolezza sul proprio futuro e l'ambiente circostante. Il robot, quindi, é definito autonomo se é un agente intelligente situato nello spazio fisico, dove un agente intelligente si definisce come un'entità che **osserva** l'ambiente e prende delle **azioni** per massimizzare il raggiungimento del suo **obiettivo**[6]. Nel caso specifico di questa tesi, l'ambiente del robot autonomo é lo spazio bidimensionale (una superficie), e il suo obiettivo é un punto in questo spazio. Massimizzare il raggiungimento di questo punto vuol dire arrivarci nel minor tempo possibile, senza collidere con eventuali ostacoli. Quindi, il robot autonomo deve compiere una serie di azioni, non note a priori e definite da un algoritmo di pianificazione che si basa sui dati osservati dai sensori, per spostare la sua traiettoria, al fine di evitare collisioni e raggiungere comunque l'obiettivo. La tipica architettura di navigazione di un robot autonomo é data perciò da quattro moduli, detti anche primitive:

Percezione Prende in input le informazioni derivanti dai sensori, le processa e le restituisce

Localizzazione e Mapping Con le informazioni sensoriali, il robot costruisce una rappresentazione del proprio intorno basandosi sulla propria posizione e ciò che osserva. Il risultato globale, dopo aver fatto varie osservazioni di intorni diversi, sarà una mappa dell'ambiente, rispetto alla quale il robot può localizzarsi. (Per scopi esemplificativi, questo modulo verrà tralasciato nell'algoritmo di questa tesi, e si userà descrivere la posizione del robot con coordinate assolute e non rispetto ad una mappa.)

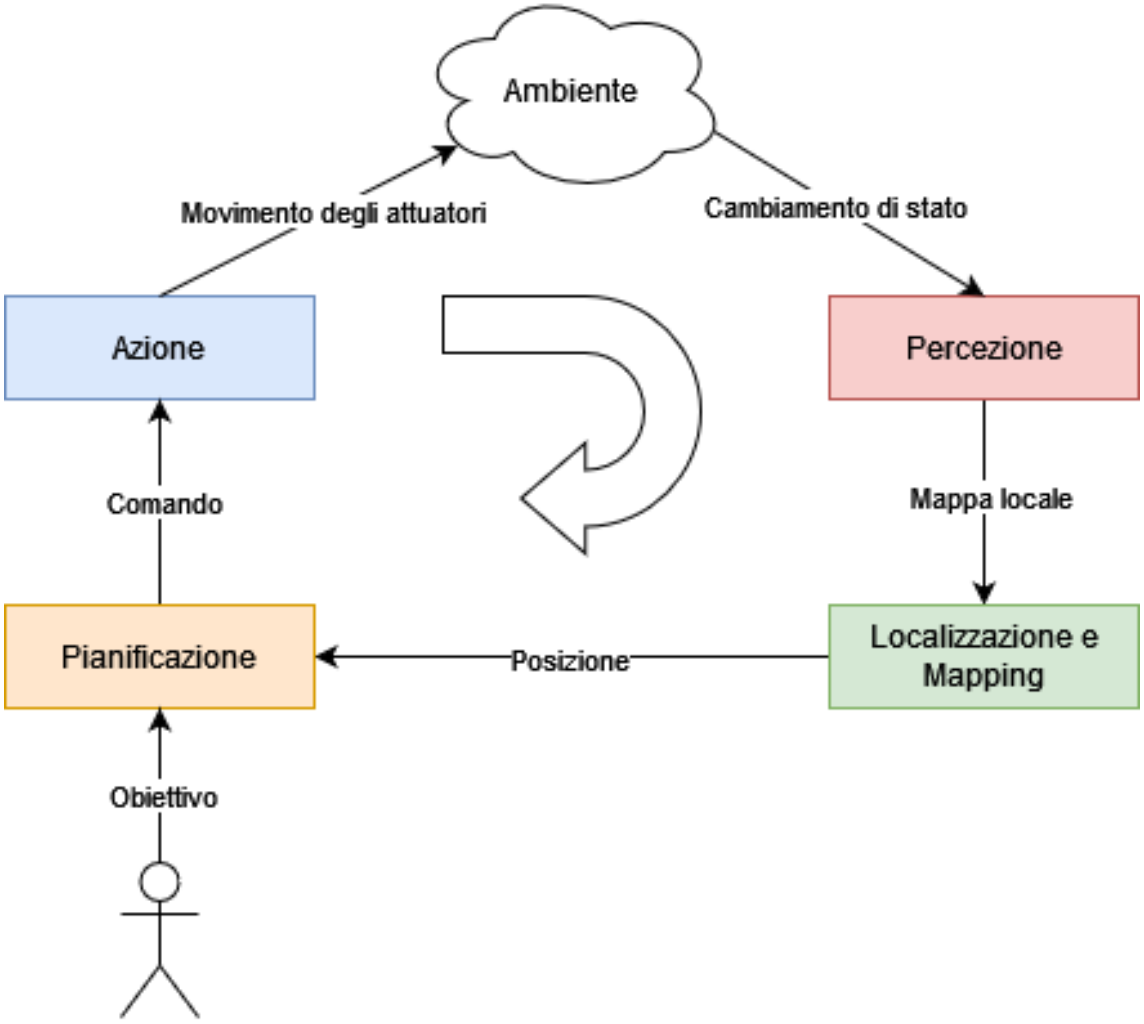
Pianificazione In base alle informazioni sensoriali e cognitive in possesso, produce in output delle decisioni ad un livello di astrazione

alto. Nel caso del motion planning, la direttiva da produrre é il percorso da seguire.

Azione Prende in input le direttive del modulo di pianificazione e produce dei comandi a basso livello per gli attuatori del robot.

L'architettura utilizzata in questa tesi é la cosiddetta gerarchica: le quattro primitive vengono eseguite in ordine e in loop. È particolarmente indicata per problemi in cui l'obiettivo finale é ben definito a priori. In altre parole, non vi é nessun meccanismo di apprendimento nel robot, ma semplicemente pianificazione deterministica orientata al goal. In figura é visivamente sintetizzato quanto appena detto.

Figure 1: Architettura di navigazione



In questa tesi viene affrontato un problema che rientra nel terzo

modulo: il path planning, un problema di grande importanza e argomento di molta ricerca.

1.1 Path planning

Una sua rapida formulazione potrebbe essere la seguente: data la posizione iniziale (del robot) A e una posizione finale B, imposta da chi fa uso del robot, il path planning consiste nel calcolare un percorso fisicamente realizzabile e ottimale per arrivare da A a B. All'interno dei metodi esistenti (e non), ci sono due importanti distinzioni da fare: sulla formulazione del problema e sulla soluzione al problema.

La prima é tra online e offline path planning, o anche locale e globale. Il path planning globale riguarda quelle situazioni in cui l'ambiente considerato é interamente noto a priori, per cui é possibile calcolare il percorso da seguire ancor prima che il robot inizi a muoversi; quello locale é inerente ai casi in cui il robot debba fare i conti lungo il suo percorso con eventi inaspettati, quali ostacoli dinamici,per cui é necessario reagire localmente, aggiornando ripetutamente le informazioni derivanti dai sensori e aggiustando la traiettoria al fine di evitare l'ostacolo e poter raggiungere in tempi ottimali l'obiettivo. Chiaramente, la maggior parte dei problemi di robotica autonome deve fare i conti con una situazione del secondo tipo.

La seconda distinzione é tra soluzioni basate su tecniche di intelligenza artificiale - la cui trattazione esula dagli scopi di questa tesi - e soluzioni classiche. Queste ultime possono ulteriormente essere suddivise in [7]:

- Subgoal (o anche roadmap), la cui realizzazione più nota é il metodo che sfrutta i diagrammi di Voronoi.
- Decomposizione in celle
- Sampling based che frutta un approccio probabilistico.
- Potenziali artificiali, che verranno ampiamente in trattati nel prossimo capitolo

Di seguito si descriverá brevemente degli esempi legati ai metodi appena elencati.

Voronoi L'idea alla base di questo metodo é rappresentare lo spazio libero delle configurazioni C_{free}, ovvero l'insieme di quei punti che per certo non fanno collidere il robot con un ostacolo, come un grafo, ovvero un insieme di nodi (rappresentati appunto la roadmap) connessi da archi. La posizione dei nodi é definita tramite il concetto di clearance, ovvero la funzione

$$\gamma(q) = \min_{s \in \partial C_{free}} \|q - s\|$$

dove q é una generica configurazione in C_{free}. ∂C_{free} sarebbe la frontiera dello spazio di configurazione, ovvero il bordo degli ostacoli. Il grafo é formato da quelle configurazioni q tali per cui esiste più di un punto su ∂C_{free} con lo stesso valore $\gamma(q)$. In altre parole, quei punti equidistanti (considerando la distanza minima) da più di un ostacolo. Il risultato é quello mostrato in figura 2a. Una volta calcolato il grafo, é sufficiente ritrarre i punti di start e goal sul grafo stesso e calcolare il percorso ottimale tra questi due attraverso un algoritmo di ricerca, ad esempio Djikstra.

Decomposizione esatta in celle In questa tecnica lo spazio delle configurazioni viene suddiviso in celle, come mostrato in figura 2b. Ogni cella é delimitata inferiormente e superiormente da un ostacolo e ogni cella ha le seguenti due caratteristiche:

- Tra ogni coppia di configurazioni nella stessa cella esiste sempre un cammino senza collisioni
- Tra ogni coppia di celle adiacenti esiste sempre un cammino senza collisioni (che conduce da una cella all'altra)

In base a questi due principi, viene costruito il cosiddetto grafo di connettività, che ha come archi le connessioni tra celle adiacenti. Come nel metodo precedente, il calcolo del percorso ottimale per raggiungere il goal consiste nell'applicare un algoritmo di ricerca su grafo.

Sampling based Il terzo metodo si basa su un approccio probabilistico. Brevemente, l'idea é di scegliere a ogni iterazioni una configurazione "di prova" e fare un test di collisione su quest'ultima e si cerca di collegare in basa alla vicinanza alle configurazioni giá appartenenti al grafo che si sta costruendo.

(a) Diagramma di Voronoi (b) Decomposizione in celle

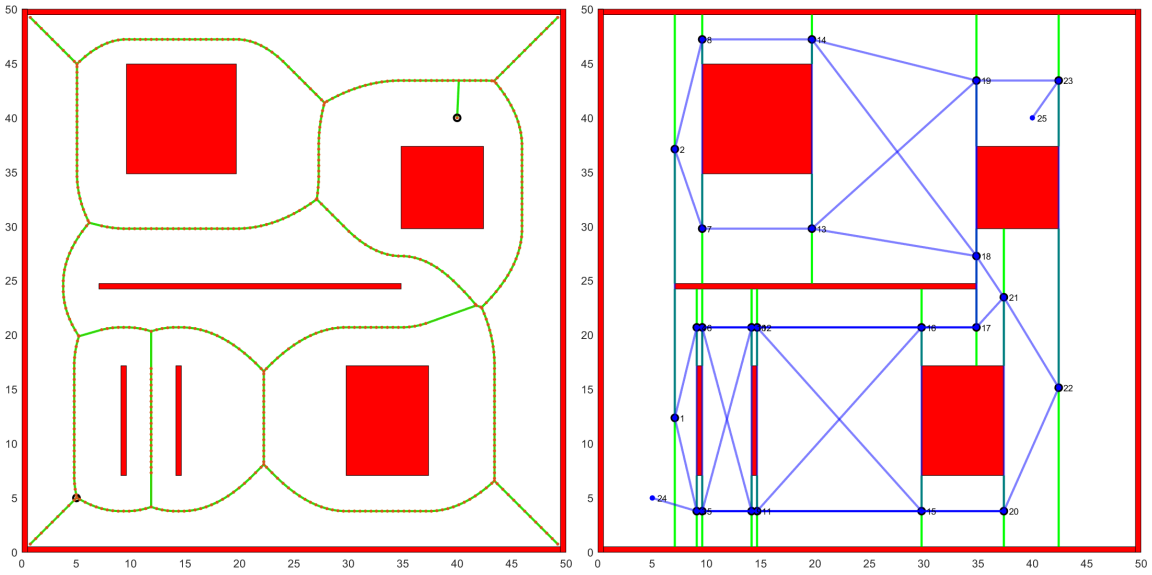
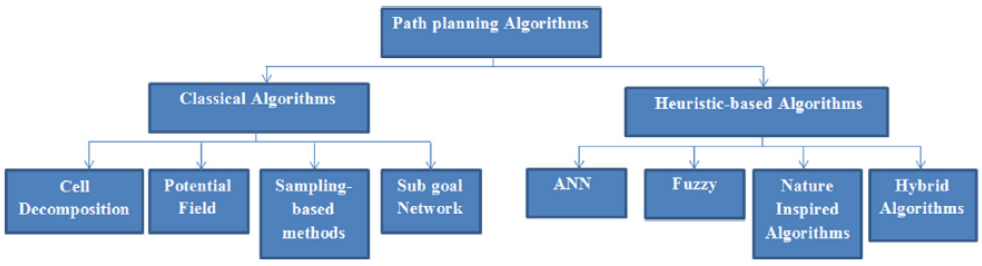


Figure 2: Metodi classici basati su grafo

La breve descrizione dei metodi serviva anche a motivare la scelta dei potenziali artificiali come base per lo sviluppo dell'algoritmo di navigazione, visto che la totalità dei metodi rientranti nelle prime tre categorie risolve adeguatamente il problema del path planning globale [2], ma risulta inefficiente nel caso del path planning locale. Tuttavia, sono particolarmente efficienti nel caso in cui ci si trovi in un ambiente noto a priori, poco incline al cambiamento e con l'esigenza di fare query ripetute. Il costo computazionale é quasi tutto contenuto nel calcolo del grafo, e la singola query ha un costo legato al numero di archi.

Figure 3: Classificazione algoritmi di path planning[4]



Come da titolo, questa tesi ha come obiettivo specifico quello di esporre un lavoro di progettazione, implementazione e simulazione di un algoritmo di path planning **locale** basato su **potenziali artificiali**

alternati.

1.2 Formalizzazione del problema

Chiamerò $r(t) = [x_r(t), y_r(t), \theta_r(t)]^T$ la posizione del robot nell'istante t e $O_i(t) = [x_{O,i}(t), y_{O,i}(t)]^T, i = 1...N$ la posizione degli N ostacoli che, per scopi esemplificativi in fase di prototipazione dell'algoritmo, saranno di forma circolare e con raggio R_i. Quest'ultima ipotesi non provoca una perdita di generalità, visto che per un ostacolo di forma generica si può considerare la sua circonferenza circoscritta. Esisterà inoltre un punto $G = [G_x, G_y]$ che indica l'obiettivo del robot. Il problema consiste nel voler raggiungere il punto G dalla posizione iniziale $r(0)$, tenendo conto degli N ostacoli in movimento. Il robot é dotato di un raggio di visione di R_v metri, entro il quale é capace di rilevare un ostacolo. Inoltre, si presuppone che valga la seguente condizione

$$\left\| \begin{bmatrix} x_r(t) \\ y_r(t) \end{bmatrix} - O_j(t) \right\| \leq R_v$$

, ovvero $R_i \leq R_v, \forall i$. Ciò vuol dire che nel momento in cui il robot incontra un ostacolo, il centro di quest'ultimo é incluso in R_v.

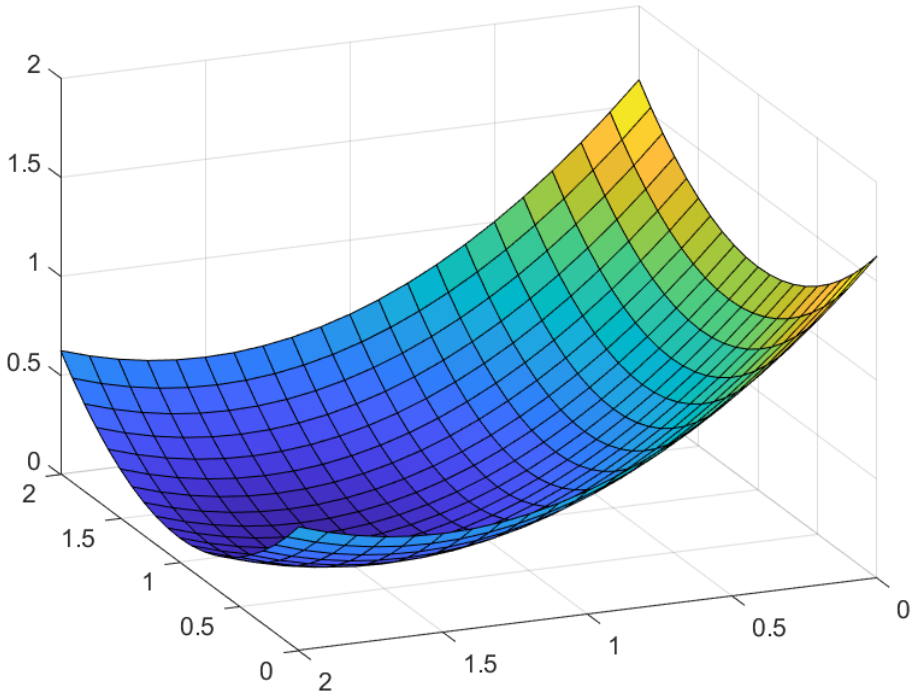
2 Potenziali artificiali

2.1 Metodo classico

Tradizionalmente, nel path planning basato su potenziali artificiali il robot viene fatto muovere mediante una funzione in due variabili, ovvero un potenziale scalare, che nasce dalla somma di due potenziali: attrattivo e repulsivo. Questi due potenziali sono chiamati artificiali perché generano una forza che guida il robot in ogni sua configurazione $r(t)$, nonostante nella realtà non vi sia nessuna sorgente a generare quella forza. Nello specifico, la forza generata dal potenziale é il suo antigradiente, ovvero il gradiente cambiato di segno, che indica al robot la direzione di moto localmente più promettente[5], cioè verso il punto di minimo della funzione. Di conseguenza il potenziale attrattivo assume una forma tale da avere un unico punto di minimo posizionato proprio nel punto di arrivo, mentre il repulsivo ha un unico punto di massimo corrispondente alla posizione dell'ostacolo.

Il potenziale attrattivo ha di solito la forma mostrata in figura 4, in cui il punto di minimo, ovvero il goal, ha coordinate $[1, 1.5]$.

Figure 4: Potenziale attrattivo

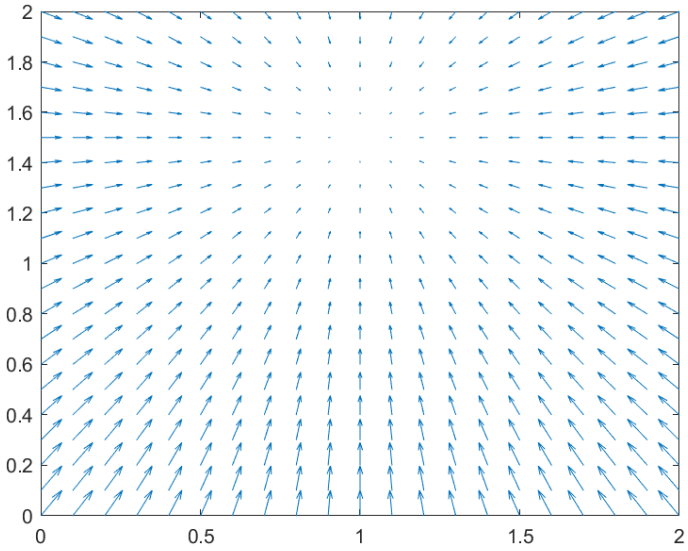


a cui corrisponde la funzione

$$U_a(r(t), G) = \frac{1}{2} \left\| G - \begin{bmatrix} x_r(t) \\ y_r(t) \end{bmatrix} \right\|^2 \tag{1}$$

Il suo antigradiente di conseguenza é formato da tanti vettori che, con un'intensità proporzionale alla distanza dal goal, puntano verso quest'ultimo.

Figure 5: Antigradiente del potenziale attrattivo



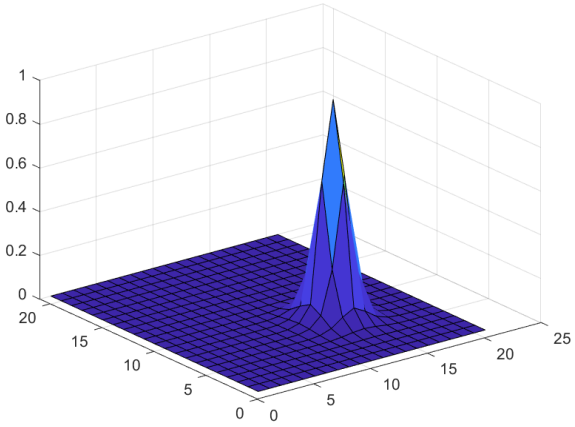
e matematicamente si esprime come il vettore delle derivate parziali (del potenziale) cambiato di segno, ovvero

$$-\nabla U_a(r(t), G) = \left\| G - \begin{bmatrix} x_r(t) \\ y_r(t) \end{bmatrix} \right\| \tag{2}$$

Perciò, la forza esercitata dal potenziale sul robot punta verso il goal e converge a zero quando la configurazione $r(t)$ tende alla destinazione G, esprimendo difatti un errore lineare tra goal e posizione del robot.

Il potenziale repulsivo ha una forma duale a quello attrattivo, come mostrato in figura 6a. Esso impone che entro uno distanza η dall'ostacolo per il quale lo si sta calcolando, la sua funzione avrà valore inversamente proporzionale alla distanza dall'ostacolo stesso. Praticamente, più ci si avvicina all'ostacolo entro una certa soglia η, più l'intensità del potenziale artificiale, perciò anche della forza generata da esso, aumenta.

(a) Potenziale repulsivo



(b) Antigradiente del potenziale repulsivo

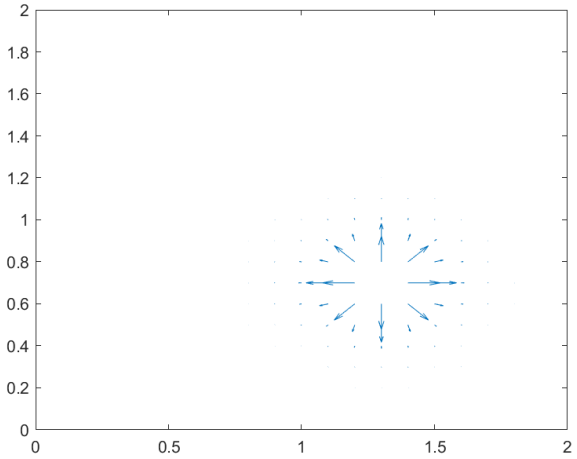


Figure 6: Potenziale repulsivo

Al potenziale repulsivo corrisponde la seguente funzione a tratti

$$U_r(r(t), O_j(t)) = \begin{cases} \frac{1}{2}\left(\frac{1}{d(r(t),O_j(t)} - \frac{1}{\eta}\right) & d(r(t), O_j(t)) \leq \eta \\ 0 & altrimenti \end{cases} \quad (3)$$

dove

$$d(r(t), O_j(t) = \left\| O_j(t) - \begin{bmatrix} x_r(t) \\ y_r(t) \end{bmatrix} \right\|$$

Dunque, il compito della forza generata é quello di spingere via il robot dalla posizione dell'ostacolo tanto più che il robot si avvicina entro la soglia η a quest'ultimo. In figura 6b si vedono le linee di campo dell'antigradiente che puntano radialmente verso l'esterno rispetto alla posizione dell'ostacolo, qui con coordinate $[1.5, 1]$.

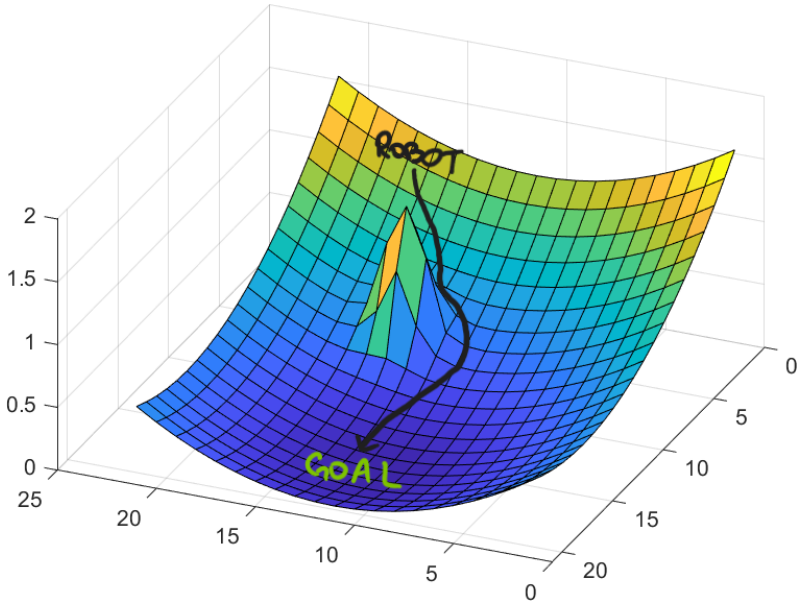
Una volta calcolato il potenziale repulsivo per ogni singolo ostacolo, si ottiene il potenziale totale:

$$U(r(t)) = U_a(r(t), G) + \sum_{j=1}^{n} U_r(r(t), O_j(t)) \quad (4)$$

In figura 7 si vede un possibile percorso del robot per arrivare dal punto di partenza al goal. Dalla 4 si vede che l'antigradiente è

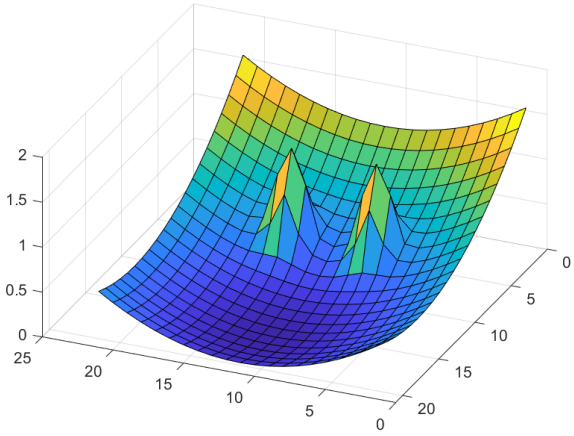
una funzione di $r(t)$: vuol dire che in ogni sua configurazione, siccome l'antigradiente si suppone già calcolato, il robot può ottenere informazioni su quest'ultimo riferendosi soltanto alla sua stessa posizione. Quindi, il vettore velocità del robot $[v_x(t), v_y(t)]$ ha come riferimento in ogni istante t il valore dell'antigradiente (che é un vettore) in $[r_x(t), r_y(t)]$. Lontano dagli ostacoli, il vettore velocità ha una direzione che punta al goal e un'intensità che diminuisce man mano che ci si avvicina al goal. Più il robot si avvicina ad un ostacolo, più la direzione del vettore velocità vira verso il verso opposto rispetto a $\theta = \tan\left(\frac{y_{O,j}(t)-y_r(t)}{x_{O,j}(t)-x_r(t)}\right)$ (ovvero la direzione del vettore che collega il robot all'ostacolo), spostando così temporaneamente la traiettoria desiderata e aumentando "l'intensità della virata" lontano dall'ostacolo man mano che si avvicina ad esso. Il robot dunque, seguendo l'antigradiente, viene in ogni sua configurazione $r(t)$ - usando l'analogia con il campo gravitazionale - attratto dal goal e **contemporaneamente** respinto dagli ostacoli. Chiaramente, il metodo é idoneo sia al path planning globale che a quello locale (basta ricalcolare il potenziale totale in presenza di ostacoli). Volendo, d'altra parte, dare un'etichetta dal punto di vista dell'architettura di navigazione, i potenziali artificiali tradizionali obbediscono ad una di tipo reattivo: considerando la figura 1, il modulo di mapping e path planning sono praticamente assenti. Gli attuatori del robot, che sono collegati ai sensori tramite una funzione di trasferimento [3], ricevono direttamente il comando derivante dal calcolo precedente del potenziale artificiale, senza alcun tipo di pianificazione algoritmica.

Figure 7: Potenziale totale



Tuttavia, vi sono delle non-idealità legate a questo approccio. Ad esempio, la traiettoria potrebbe non essere continua. Il potenziale repulsivo non é "coordinato" con quello attrattivo, perciò il robot potrebbe ricevere comandi che causerebbero un cambio di direzione o di velocità troppo repentino che andrebbe gestito dalla legge di controllo o un modulo di pianificazione apposito. Questa osservazione verrà trattata nel capitolo riguardante l'algoritmo di navigazione.
Altro problema da considerare é il fatto che il robot viene "passivamente" spinto via dagli ostacoli senza la certezza che venga portato in una posa in cui può effettivamente evitare con successo l'ostacolo.
Ma il più importante, sicuramente, é il problema dei minimi locali: il robot non ha alcuna informazione su come uscirne, né può prevederli in anticipo. I minimi locali sono dei punti ad antigradiente nullo (i gradiente del potenziale attrattivo e repulsivo possono annullarsi a vicenda), dove il robot non ha nessuna forza a guidarlo verso il goal, situazione che dovrebbe verificarsi soltanto nel punto di goal stesso. In figura 8 viene mostrato il percorso simulato di un robot in presenza di un minimo locale nel potenziale artificiale. Il robot parte dalla posizione $r(0) = [5, 0]$ mentre il goal si trova in posizione $G = [5, 10]$. Gli ostacoli sono posizionati tra il robot e il goal ad una distanza tale da causare un annullamento del gradiente (quindi il minimo locale) proprio a metà tra i due, lì dove il robot cerca di passare. La traiettoria del robot, infatti, si ferma nel minimo locale: non viene spinto da nessuna forza e "crede" di essere arrivato nel punto di goal.

(a) Potenziale con minimo locale



(b) Antigradiente del potenziale con minimo locale

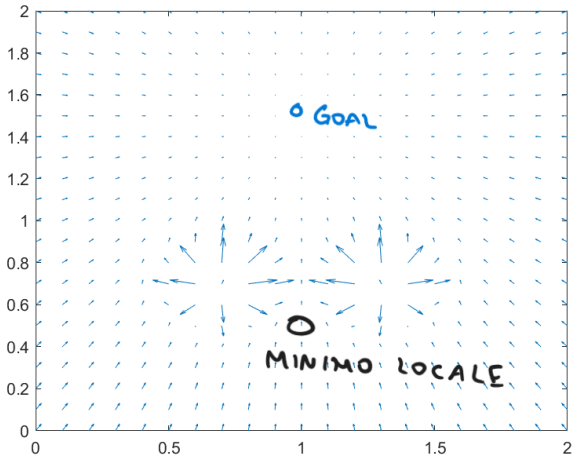


Figure 8: Minimo locale

2.2 Potenziale bypassante

Al fine di evitare le non-idealità dovute all'utilizzo dei potenziali sommati, l'idea implementata in questa tesi é quella di sfruttare al posto di quello repulsivo un altro tipo di potenziale che chiameremo bypassante. Fondamentalmente, si tratta di un potenziale che invece di spingere via il robot dall'ostacolo, lo porta a circumnavigarlo. Infatti, le linee di campo dell'antigradiente di questo potenziale sono concentriche, al contrario di quelle del potenziale repulsivo che sono radiali. L'idea quindi é di basarsi su una superficie che ruoti attorno a un centro, identificato dalla posizione, ad esempio un elicoide:

$$\begin{cases} x = x_0 + rcos(\xi) \\ y = y_0 + rsin(\xi) \\ z = c\xi \end{cases} \tag{5}$$

Applicando nella terza equazione la tangente da entrambe le parti, otteniamo $\tan(\xi) = \tan(\frac{z}{c})$ che, confrontando con le prime due equazioni, diventa

$$\tan\left(\frac{z}{c}\right) = \frac{y - y_0}{x - x_0}$$

dove $[x_0, y_0]$ sarebbe la posizione dell'ostacolo. Invertendo questa funzione per esprimerla come z in funzione di x e y, otteniamo il potenziale elicoidale, ridenominato bypassante, in senso orario rispetto all'ostacolo e centrato in esso:

$$\Gamma(x, y, x_0, y_0) = c \tan^{-1}\left(\frac{y - y_0}{x - x_0}\right) \tag{6}$$

(a) Elicoide centrato in $[1, 1]$ con $c = 2$, $0 \leq r \leq 1$ e $0 \leq \xi \leq 4\pi$

(b) Potenziale bypassante centrato in $[1, 1]$ con $c = 2$ e in senso orario

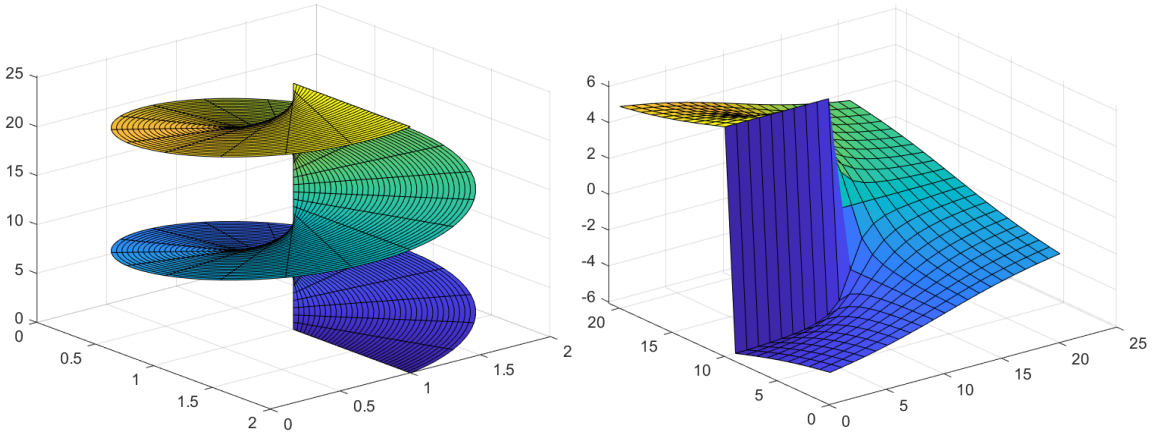


Figure 9: Potenziale bypassante

Da notare che il potenziale in figura 9b ha una discontinuità per $x = x_0$, per cui la funzione é continua per $(x, y) \neq (x_0, y_0)$. Siccome il robot non può mai andare esattamente nella stessa posizione dell'ostacolo, non c'è perdita di generalità nell'ignorare la discontinuità e di conseguenza il robot sarà sempre in grado di seguire la forza generata dal potenziale. Il potenziale con equazione 6 può essere visto come una funzione della posizione del robot e dell'ostacolo da aggirare

$$\Gamma(x, y, x_0, y_0) = \Gamma(x_r(t), y_r(t), x_{O,j}(t), y_{O,j}(t))$$

Dunque, la forza generata dal potenziale bypassante é esprimibile come il suo antigradiente

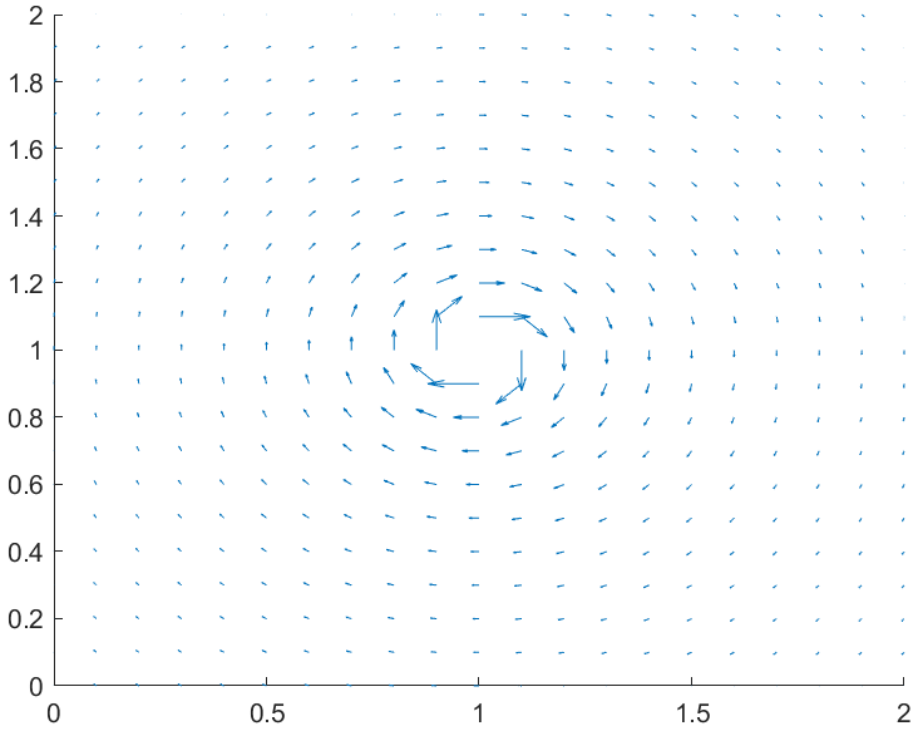
$$-\nabla U_b(r(t), O_j(t)) = \begin{bmatrix} \frac{c(y_r(t) - y_{O,j}(t))}{(x_r(t) - x_{O,j}(t))^2 + (y_r(t) - y_{O,j}(t))^2} \\ \\ \frac{c(x_{O,j}(t) - x_r(t))}{(x_r(t) - x_{O,j}(t))^2 + (y_r(t) - y_{O,j}(t))^2} \end{bmatrix} \tag{7}$$

Quindi, l'antigradiente é formato da tanti vettori che

- Indicano una velocità desiderata nella posa $r(t)$ del robot, al fine di aggirare l'ostacolo preso in considerazione

- Aumentano di intensità man mano che il robot si avvicina all'ostacolo
- Hanno forma concentrica

Figure 10: Antigradiente del potenziale bypassante

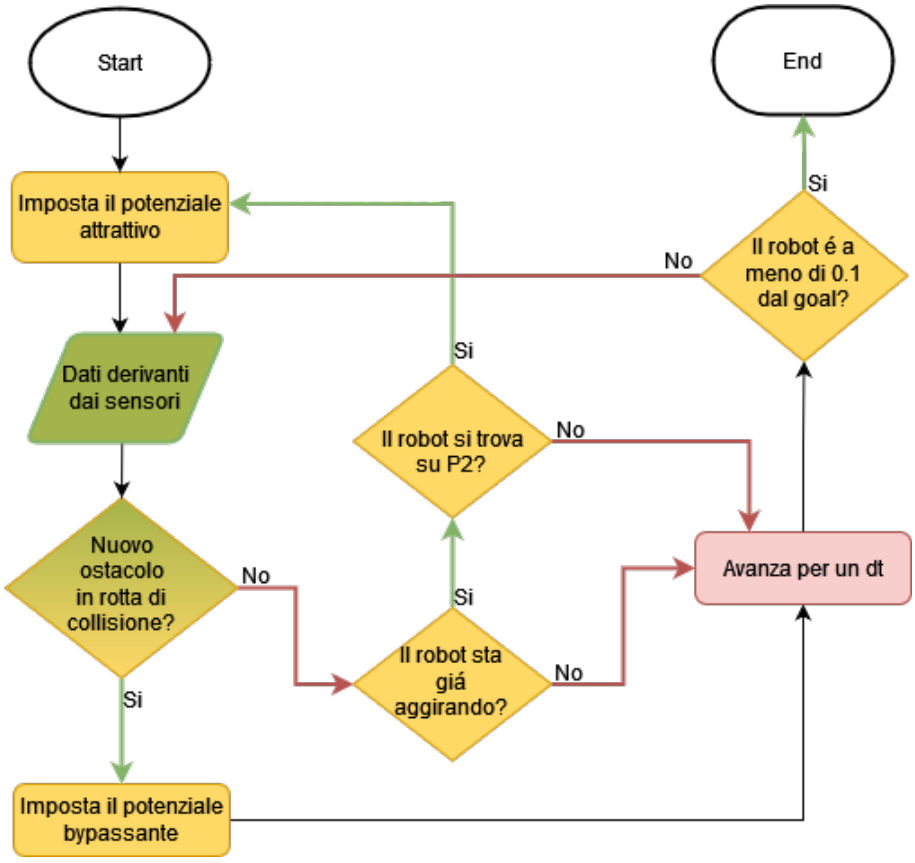


Questo tipo di potenziale, chiaramente, é intrinsecamente adatto ad ostacoli di forma circolare, motivo per cui nel lavoro di tesi si é considerato soltanto questo tipo di ostacoli.

3 Algoritmo di navigazione[2]

La strategia é semplice: nella posizione iniziale, il robot sonda l'ambiente attorno a sé: se non vi sono presenti ostacoli a impedirne l'avanzamento verso il goal, la traiettoria da seguire é quella imposta dal potenziale attrattivo; altrimenti, é necessario "switchare" dal potenziale attrattivo a quello bypassante, al fine di aggirare l'ostacolo, per poi tornare a seguire la traiettoria. Perciò, la peculiarità di questo algoritmo é che in ogni istante di tempo il robot seguirà un solo potenziale alla volta, evitando così il problema dei minimi locali. Inoltre, le informazioni necessarie a pianificare lo switch non richiedono informazioni globali, ma solo relative all'ostacolo da aggirare.

Figure 11: Flowchart Diagram



3.1 Considerazioni implementative

Costruzione dell'ambiente Come viene costruito l'ambiente (grid)

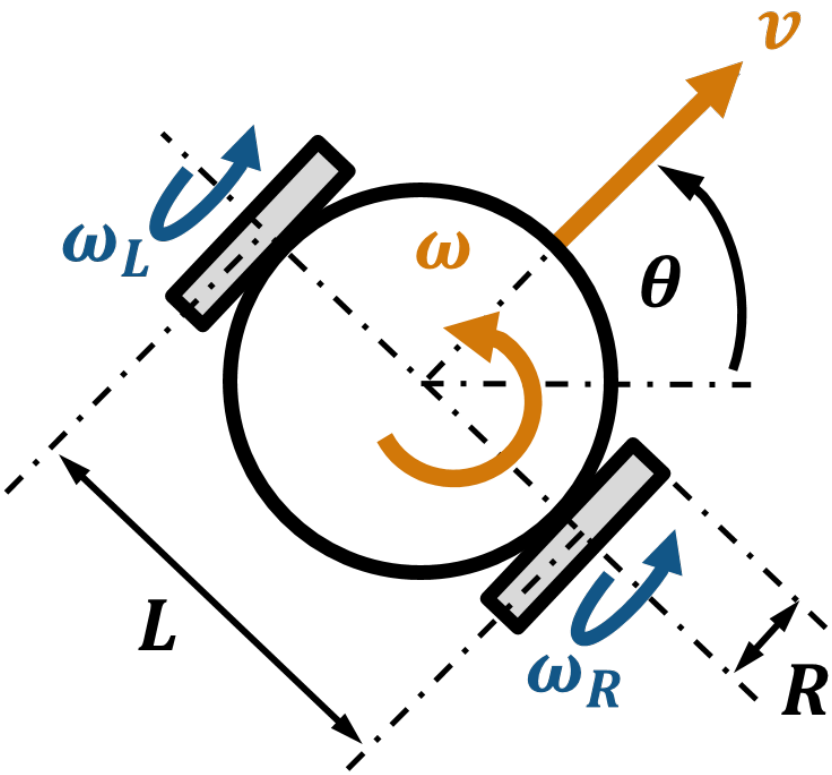
3.2 Azione

Questo modulo riceve un comando dal modulo di pianificazione, ovvero l'antigradiente da seguire. Il suo compito é quello di inoltrare il comando agli attuatori a basso livello (i motori).

3.2.1 Modello cinematico

Si é ipotizzato in questa tesi di usare un modello cinematico anolonomo di tipo differential drive.

Figure 12: Modello Differential-Drive [1]



Il fatto che sia anolonomo vuol dire che obbedisce ad un vincolo sulla velocitá e non sulla posizione, ovvero

$$\theta(t) = \arctan\left(\frac{\dot{y}(t)}{\dot{x}(t)}\right)$$

Praticamente, al robot é impossibile muoversi in ogni direzione con la stessa velocità; per mantenerla, deve continuare a mantenere il suo orientamento invariato. Le equazioni differenziali che modellano un

generale robot che obbedisce a vincolo anolonomo é

$$\begin{cases} \dot{x}(t) = v(t)\cos(\theta(t)) \\ \dot{y}(t) = v(t)\sin(\theta(t)) \\ \dot{\theta}(t) = \omega(t) \end{cases} \tag{8}$$

Quindi, secondo questo modello, un robot può essere completamente caratterizzato mediante la sua posizione $[x, y, \theta]$ e la sua velocità $[v, \omega]$, dove v indica la velocità lineare (di traslazione) e ω quella angolare (di rotazione).
In particolare nel modello differential drive il robot, come mostrato in figura **??**, viene mosso da due ruote laterali (ogni ruota ha il suo motore) di raggio R, posizionate sullo stesso asse e distanti tra di loro L. Detto ciò, si può scrivere la velocità lineare come la media tra le velocità lineari delle due ruote

$$v(t) = \frac{R \cdot \omega_R(t) + R \cdot \omega_L(t)}{2}$$

e quella angolare come la differenza tra le due velocità lineari normalizzata per la distanza tra le ruote

$$\omega(t) = \frac{R \cdot \omega_R(t) - R \cdot \omega_L(t)}{L}$$

Riunite in forma matriciale, queste due relazioni danno la seguente

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ \frac{R}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} \omega_R(t) \\ \omega_L(t) \end{bmatrix} \tag{9}$$

Invertendo questa relazione, si può ottenere la velocitá delle due ruote a partire dalla velocità lineare e angolare di riferimento.

$$\begin{bmatrix} \omega_R(t) \\ \omega_L(t) \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ \frac{R}{L} & -\frac{R}{L} \end{bmatrix}^{-1} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{10}$$

Questa inversione é sempre possibile, visto che

$$det \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ \frac{R}{L} & -\frac{R}{L} \end{bmatrix} = \frac{-R^2}{L}$$

che é sempre non nullo, per cui la matrice é invertibile.

Dunque, il modello 8 diventerá

$$\begin{cases} \dot{x}(t) = \frac{R}{2}\left(\omega_R(t) + \omega_L(t)\right)\cos(\theta(t)) \\ \dot{y}(t) = \frac{R}{2}\left(\omega_R(t) + \omega_L(t)\right)\sin(\theta(t)) \\ \dot{\theta}(t) = \frac{R}{L}\left(\omega_R(t) - \omega_L(t)\right) \end{cases} \tag{11}$$

3.2.2 Legge di controllo [2]

Nel caso specifico del path planning con potenziali artificiali, la velocità di riferimento é data dall'antigradiente relativo alla posa del robot, cioè $v_\nabla(t) = -\nabla U(r(t))$. Con la legge di controllo viene stabilita una velocità lineare e angolare da imporre al robot e viene stabilito che

$$v(t) = M_v \cos(\theta_\nabla(t) - \theta_r(t)) \tag{12}$$

$$\omega(t) = K_\omega(\theta_\nabla(t) - \theta_r(t)) \tag{13}$$

dove $M_v = \left\| v_\nabla(t) \right\|$, $\theta_\nabla = \angle v_\nabla(t)$ e

$$K_\omega(t) = \begin{cases} \frac{\dot{\theta}_\nabla(t) + K_c|\theta_\nabla(t) - \theta_r(t)|^\nu \cdot sign(\theta_\nabla(t) - \theta_r(t))}{\theta_\nabla(t) - \theta_r(t)} & |\theta_\nabla(t) - \theta_r(t)| \geq \xi \\ 0 & altrimenti \end{cases}$$

Una volta calcolate le velocità di riferimento $v(t)$ e $\omega(t)$, basta applicare la relazione 10 per avere le velocitá di riferimento relative alle due ruote del differential drive, ottenendo cosí i comandi da impartire agli attuatori del robot.

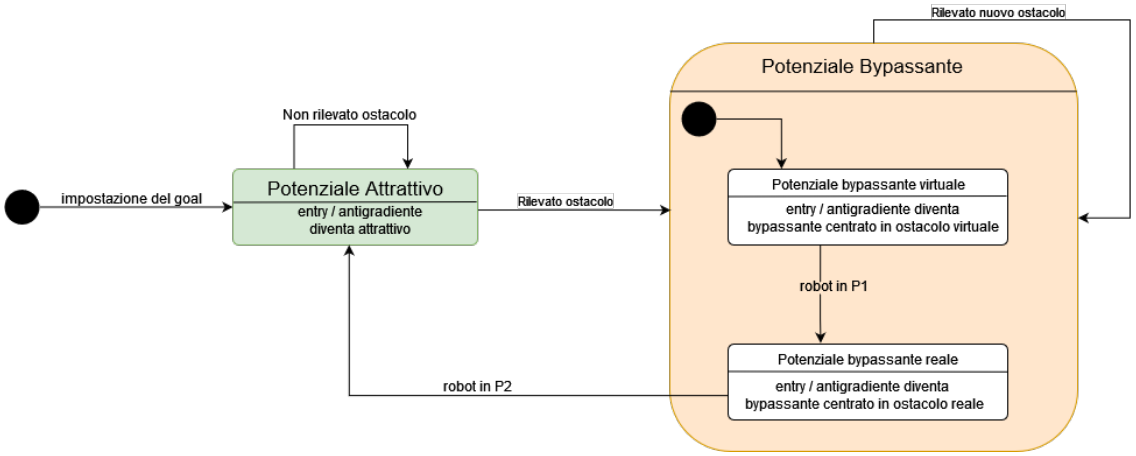
3.2.3 Codice MATLAB

Le funzionalità legate al modulo in esame sono tutte racchiuse nella classe Act, riportata nel code listing 1.

3.3 Pianificazione

Il modulo di pianificazione riceve ad ogni iterazione in input i dati processati dai sensori e in base a ciò decide a quale potenziale il robot deve fare riferimento, informazione che sarà inoltrata al modulo di azione. Data la struttura intrinseca della strategia di navigazione che si basa su un meccanismo di switching tra due stati, per definizione mutuamente esclusivi, un modo per modellare visivamente il modulo di pianificazione é lo statechart diagram in figura 13. Il robot può trovarsi o nello stato corrispondente al potenziale attrattivo, oppure in quello relativo al potenziale bypassante.

Figure 13: Statechart Diagram



3.4 Percezione

4 Simulazione

4.1 Sequence diagram

4.2 Risultati

5 Applicazioni e sviluppi futuri

Listing 1: Classe per il modulo di Azione

```
classdef Act

    properties
        robot; grid;
    end

    methods
        %% Constructor
        function obj = Act(grid)
            obj.grid = grid;
        end

        %% Method that moves the robot according to the
        % desired velocity (Runge-Kutta 2 for integration)
        function newPose = move(obj,pose,gradX,gradY,tspan)
            rx = pose(1);
            ry = pose(2);
            rtheta = pose(3);
            [vr,wr] = obj.commands(rx,ry,rtheta,tspan,gradX,gradY);
            Xdot = obj.odeDD(vr,wr,rtheta);

            rx2 = rx + tspan/2*Xdot(1);
            ry2 = ry + tspan/2*Xdot(2);
            rtheta2 = rtheta + tspan/2*Xdot(3);
            [vr,wr] = obj.commands(rx2,ry2,rtheta2,tspan,gradX,gradY);
            Xdot = obj.odeDD(vr,wr,rtheta2);

            newPose(1) = rx + tspan*Xdot(1);
            newPose(2) = ry + tspan*Xdot(2);
            newPose(3) = rtheta + tspan*Xdot(3);
        end

        %% Metodo che genera velocita lineare e angolare per il robot
        function [vr,wr] = commands(obj,rx,ry,rtheta,tspan,gradX,gradY)
            i = obj.grid.coord2index([rx,ry]);
            thetaN = atan2(gradY(i(1),i(2)),gradX(i(1),i(2)));
            thetaDiff = atan2(sin(thetaN-rtheta),cos(thetaN-rtheta));
            vgrad = [gradX(i(1),i(2)) gradY(i(1),i(2))];
            Mv = norm(vgrad);
            vr = (Mv * cos(thetaDiff));

            rx1 = rx + vgrad(1)*tspan;
            ry1 = ry + vgrad(2)*tspan;
            j = obj.grid.coord2index([rx1,ry1]);
            thetaN1 = atan2(gradY(j(1),j(2)),gradX(j(1),j(2)));

            thetaDdiff = atan2(sin(thetaN1-thetaN),cos(thetaN1-thetaN));
            thetaDdot = thetaDdiff/tspan;
            Kc = 10; eps = 0.001; v = 1;
            Kw = (thetaDdot + Kc * abs(thetaDiff)^v * sign(thetaDiff))/(thetaDiff+eps);
            wr = (abs(thetaDiff) >= eps) * Kw * (thetaDiff);
        end
    end

    methods(Access = private)
        function [Xdot,wRwL] = odeDD(~,vr,wr,theta)
            % Modello specifico del Differential Drive
            r = 0.1; %raggio delle ruote di 10 centimetri
            L = 0.25; %distanza tra le due ruote di 25 centimetri
            %K rende possibile esprimere vr e wr in funzione delle
            %due velocita impresse alle ruote
            K = [r/2 r/2 ; r/L -r/L];
            wRwL = K \ [vr ; wr];
            Xdot = ([cos(theta) 0 ; sin(theta) 0 ; 0 1] * K * wRwL);
        end
    end
end
```

List of Figures

Listings

References

[1] MathWorks Student Competitions Team (2022). Mobile Robotics Simulation Toolbox (https://github.com/mathworks-robotics/mobile-robotics-simulation-toolbox).

[2] Luigi D'Alfonso et al. "Obstacles avoidance based on switching potential functions". In: *Journal of Intelligent & Robotic Systems* ().

[3] Dong-Han Kim and Jong-Hwan Kim. "A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer". In: *Journal of Robotics & Autonomous Systems* ().

[4] Thi Thoa Mac et al. "Heuristic approaches in robot path planning: A survey". In: *Robotics and Autonomous Systems* ().

[5] Claudio Medio and Giuseppe Oriolo. "Robot Obstacle Avoidance Using Vortex Fields". In: *Advances in Robot Kinematics*. 1991.

[6] Robin Murphy. *Introduction to AI Robotics*.

[7] Bruno Siciliano et al. *Robotica. Modellistica, pianificazione e controllo*.