Progetto di Robotica Mobile

Giorgio Ubbriaco
209899

UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA
DIMES

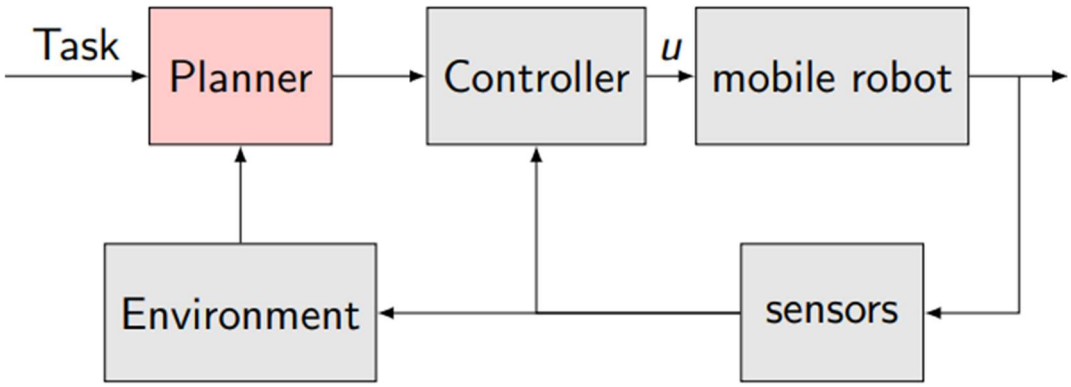
Sommario

Abstract

Il progetto in questione viene presentato facendo riferimento a piccoli cenni di teoria, evidenziando gli aspetti implementativi delle tecniche e dei modelli considerati e mettendo in luce i risultati ottenuti. Inoltre, viene citato un autore di una pubblicazione del 1985 che ha messo in risalto alcuni aspetti teorici trattati.

Introduction

L'obiettivo di questo progetto di robotica mobile è quello di progettare le principali tecniche di *path planning* e di controllo tenendo conto di un robot mobile di tipologia *uniciclo* e di ambiente composto da 6 ostacoli di varie dimensioni. Il seguente schema di controllo riassume i task principali per la generica progettazione di un robot mobile:



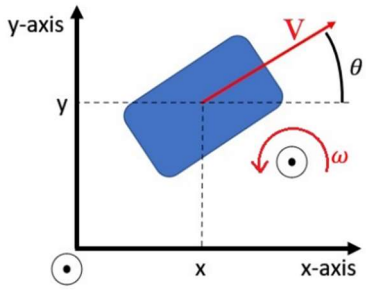
Schema di Controllo del Robot Mobile

Robot Model

Un robot mobile di tipologia *uniciclo* è un veicolo avente una sola ruota orientabile. La configurazione di tale robot è descritta da $q = [x\ y\ \theta]^T$, dove (x, y) sono le coordinate cartesiane del punto di contatto della ruota con il suolo e θ è l'orientamento della ruota e, pertanto, del veicolo in questione rispetto all'asse x. Il vincolo del puro rotolamento della ruota è descritto dalla seguente relazione:

$$\dot{x}\sin\theta - \dot{y}\cos\theta = [\sin\theta\ -\cos\theta\ \ 0]\dot{q} = 0$$

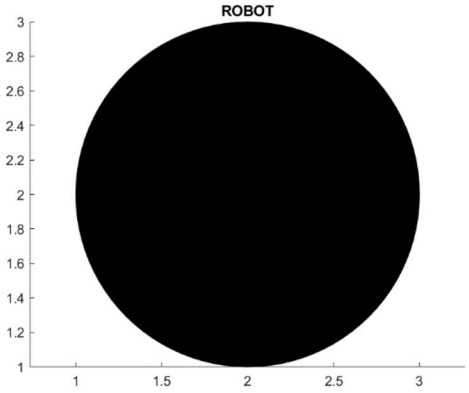
e indica che la velocità del punto di contatto è nulla nella direzione normale all'asse sagittale del veicolo. Pertanto, il modello del robot mobile di tipologia *uniciclo* risulta essere il seguente:



Unicycle Robot Model

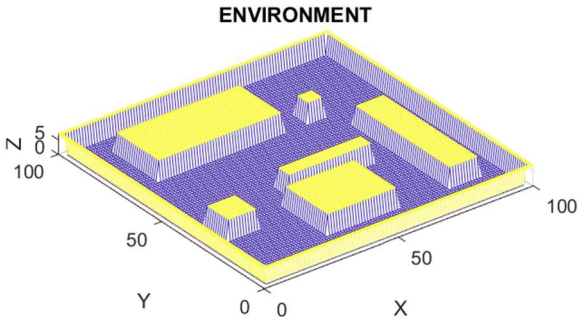
$$\begin{cases} \dot{x}(t) = v(t)\cos(\theta(t)) \\ \dot{y}(t) = v(t)\sin(\theta(t)) \\ \dot{\theta}(t) = \omega(t) \end{cases}$$

dove $v(t)$ è la velocità di "guida" (*driving velocity*) mentre $\omega(t)$ è la velocità di sterzata (*steering velocity*).



In questo progetto è stato considerato un robot mobile di tipologia *uniciclo* di raggio pari a 1 e di forma circolare ai fini dell'implementazione. Infatti, la condizione ideale e teorica era quella di considerarlo di raggio pari a 0.5, cioè puntiforme.

Environment



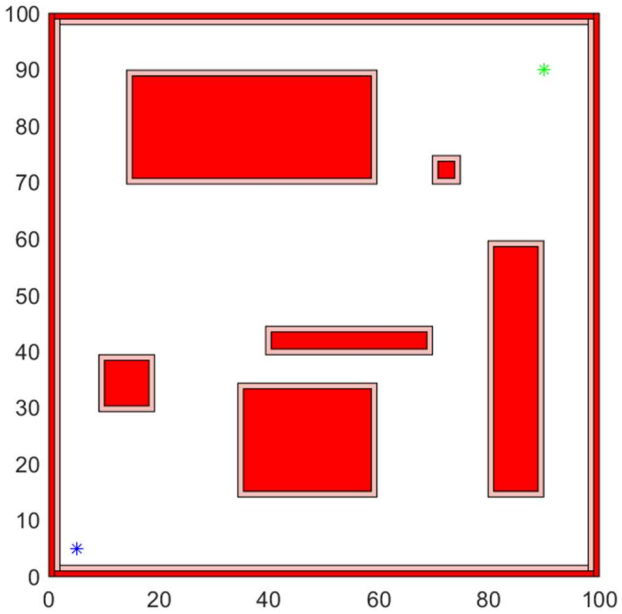
3D Environment Representation

L'environment considerato prevede una larghezza ed una lunghezza rispettivamente pari a 100. Pertanto, la griglia binaria corrispondente in cui sono stati aggiunti gli ostacoli prevede un numero di righe e di colonne pari al numero sopra indicato. I muri che delimitano l'ambiente sono stati considerati come ostacoli di forma mista (sia rettangolare che quadrata). Nello specifico tale griglia presenta valori pari a 0 o 1. Il valore 0 corrisponde ad una posizione (cella della matrice) libera mentre il valore 1 corrisponde ad una posizione occupata dagli ostacoli.

Path Planning

La simulazione del path planning viene effettuata tramite lo script "*path_planning_exec(environment).m*". Sono state implementate ben 4 tecniche di path planning. Ogni tecnica restituisce in output la traiettoria generata dagli algoritmi corrispondenti. Inoltre, è stata considerato a priori un "ingrandimento" degli ostacoli che viene descritto nel seguente sotto-paragrafo.

Obstacles Shape



Environment with Obstacle Management

Prevede di circoscrivere la forma dell'ostacolo con una forma nota al fine di evitare eventuali impatti del robot contro l'ostacolo mentre segue le traiettorie progettate. In questo caso specifico essendo ogni ostacolo una forma nota (rettangolo o quadrato), lo si ingrandisce di 1 unità, cioè pari al raggio del robot mobile in questione. Nello specifico la parte degli ostacoli colorata di rosso indica l'ostacolo vero e proprio mentre quella colorata di rosso chiaro indica l'enlargement considerato. Inoltre, sono stati considerati i punti di *start* e *goal* aventi rispettivamente coordinate:

$$start = [5\ 5]$$
$$goal = [90\ 90]$$

Pertanto, il robot mobile in questione partirà dalla posizione di *goal* e, in base alla traiettoria progettata e generata, dovrà giungere nella posizione di *goal* specificata.

Artificial Potential Fields

Questa tecnica consiste nel far muovere il robot mobile nello spazio delle configurazioni sotto l'azione di un campo potenziale U ottenuto come sovrapposizione di un potenziale attrattivo verso la destinazione e di un campo repulsivo dalla regione dei C-ostacoli. La pianificazione avviene per iterazioni: ad ogni configurazione del robot, la forza artificiale generata dal potenziale viene generata come l'antigradiente $-\nabla U$ del potenziale, e indica la direzione del moto localmente più promettente.

Il potenziale repulsivo viene considerato affinché il robot, sotto l'azione della forza attrattiva, non entri in collisione con gli ostacoli. L'idea di base presa in esame è stata quella di considerare ogni ostacolo come una componente convessa CO_i tale da definire un corrispondente potenziale repulsivo come:

$$U_{r,i}(q) = \begin{cases} \frac{k_{r,i}}{\gamma}\left(\frac{1}{\eta_i(q)} - \frac{1}{\eta_{0,i}}\right)^{\gamma} \; se\; \eta_i(q) \leq \eta_{0,i} \\ 0\; se\; \eta_i(q) > \eta_{0,i} \end{cases}$$

dove q è la configurazione del robot, $k_{r,i} > 0$, $\gamma = 2,3,\dots$, $\eta_i(q) = min_{q' \in CO_i}\|q - q'\|$ è la distanza dalla componente convessa CO_i ed $\eta_{0,i}$ rappresenta il raggio di influenza di CO_i. Il potenziale $U_{r,i}(q)$ si annulla al di fuori del raggio di influenza $\eta_{0,i}$ mentre tende ad infinito quanto più grande è γ. Per quanto riguarda il calcolo di $\eta_i(q)$, esso è stato appena definito come $\eta_i(q) = min_{q' \in CO_i}\|q - q'\|$. Esso, infatti, viene ottenuto tramite la funzione nativa *bwdist* presente in MATLAB. Tale function calcola la distanza euclidea tale che risulta essere la distanza tra la generica cella della matrice e la cella diversa da zero più vicina. In questa maniera si riesce ad ottenere le distanze i-esime tra una generica configurazione del robot e la prossima che dovrà assumere. Tale calcolo è stato assegnato all'interno dello script con la variabile *Rho*. Inoltre, la variabile $k_{r,i}$ è identificata nello script dalla variabile *Eta*. In questo caso è stato scelto un $k_{r,i} = 1$ e un $\gamma = 2$ che risultano essere le scelte tipiche in fase di implementazione del potenziale repulsivo. Tali assunzioni implementative provengono dalla FIRAS Function che Oussama Khatib propose nel 1985 con la pubblicazione dell'articolo "*Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*" per il *The International Journal of Robotics Research* che viene riportata qui di seguito:

$$U_{r,i} = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho_i} - \frac{1}{\rho_{0,i}}\right)^{2} \; se\; \rho_i \leq \rho_{0,i} \\ 0\; se\; \rho_i > \rho_{0,i} \end{cases}$$

Infatti, come lo stesso Oussama Khatib afferma, ρ_0 rappresenta la minima distanza dall'ostacolo O.
Il potenziale attrattivo, invece, è stato definito nella seguente maniera:

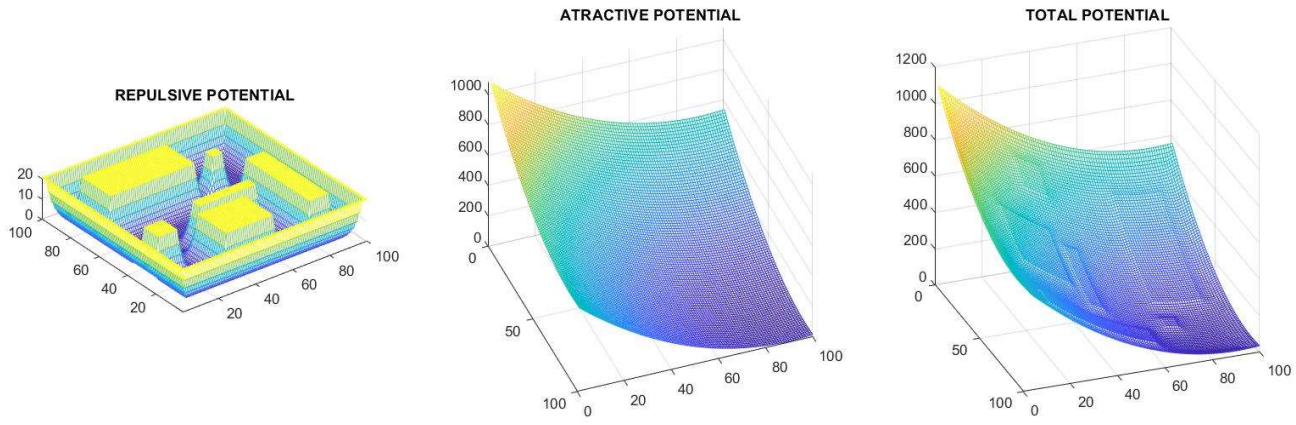
$$U_a = \frac{1}{2}k\|p_i - p_{g,i}\|^2$$

dove x è la generica posizione, x_d è la posizione target "g" (tratto sempre dalla pubblicazione di Oussama Khatib). In questo caso il coefficiente $\frac{1}{2}k$ è stato considerato pari a $\frac{1}{15}$ ed assegnato alla variabile *xi*. Tale coefficiente viene utilizzato per regolare la forza del potenziale attrattivo. Per valori troppo bassi,

cioè per un denominatore che tende sempre di più a valori più grandi comporta una forza attrattiva bassa tale che il robot mobile incontrerà maggiore difficoltà a raggiungere la posizione target. Invece, per un coefficiente sempre maggiore e, quindi, per un denominatore sempre minore, corrisponde una forza attrattiva sempre più grande tale che il robot mobile incontrerà minore difficoltà a raggiungere la posizione obiettivo. Tanto è vero che, per valori troppo grandi di forza attrattiva, il robot potrebbe essere così tanto "attratto" che, nel caso peggiore, potrebbe passare attraverso gli stessi ostacoli.
Il potenziale totale, pertanto, sarà pari alla soma del potenziale repulsivo e di quello attrattivo:

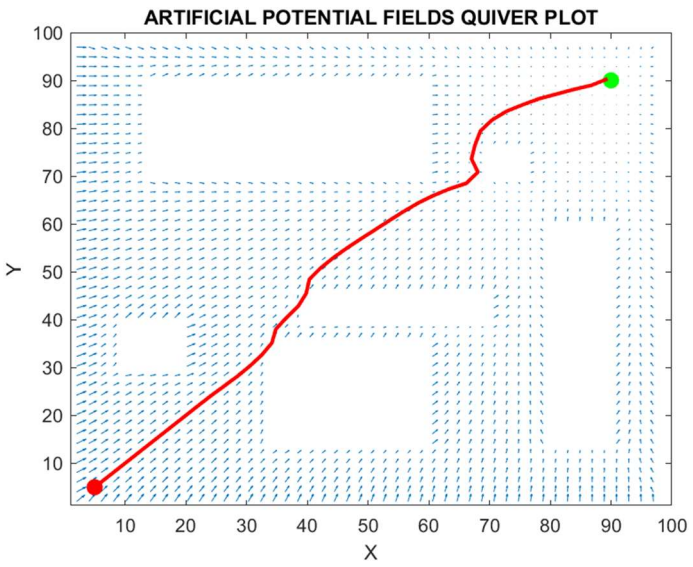
$$U_t(q) = U_a(q) + U_r(q)$$

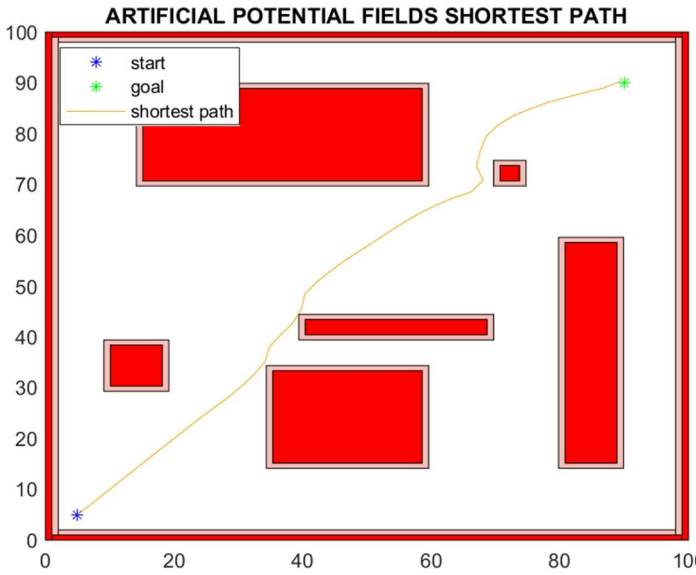
Viene riportato qui di seguito il plot grafico del potenziale repulsivo, attrattivo e totale:



Si può notare come il potenziale repulsivo rappresenti alla perfezione l'ambiente considerato. Questo perché geometricamente più il robot si avvicina all'ostacolo generico più il potenziale repulsivo aumenta. Infatti, la colorazione giallo-blu sta ad indicare che nelle regioni "libere", cioè dove il robot ha libero passaggio, presenta un colore blu scuro mentre nelle regioni occupate dagli ostacoli è presente un colore che gradualmente sfuma verso il giallo intenso. Ovviamente anche i muri, essendo considerati come ostacoli, presenteranno le stesse caratteristiche appena descritte. Per quanto riguarda, invece, il potenziale attrattivo, essendo caratterizzato da una forza attrattiva relativamente media, il paraboloide risulta essere relativamente marcato. Tanto è vero che, si può notare come il potenziale totale presenti il paraboloide del potenziale attrattivo con piccoli cenni di potenziale repulsivo. Questo è dovuto al fatto che sia la forza repulsiva che la forza attrattiva scelte sono relativamente modeste come valori e, inoltre, essendo indirettamente proporzionali, l'uno cerca di contrastare l'altro.

Dopo che si sono definite approfonditamente le premesse e i dettagli implementativi relativi ai potenziali utilizzati, è possibile generare la traiettoria utilizzando un algoritmo che calcola, sulla base del gradiente del potenziale totale considerato, per iterazioni, la direzione e, quindi, la prossima posizione del robot mobile, all'interno dell'environment considerato. Per ogni posizione viene considerata la direzione dell'antigradiente tale che il robot si dirigerà in base alla direzione prossima calcolata. Se la nuova posizione calcolata risulta essere in un intorno del goal prefissato allora l'algoritmo si arresta. Nel caso peggiore, se viene superato un certo numero di iterazioni massimo, allora vorrà dire che il robot mobile non sarà riuscito a raggiungere il goal prestabilito. Il plot seguente illustra come l'uniciclo, partendo dalla posizione di start, riesce a raggiungere la posizione di goal evitando gli ostacoli presenti all'interno dell'ambiente. Si può notare come le frecce vicino agli ostacoli risultano avere direzione verso l'esterno tale da "respingere" il robot e "condurlo" verso le celle libere. Ovviamente, per celle lontane dagli ostacoli, il potenziale risulta non essere influenzato dal repulsivo essendo lontano. Infatti, come si può notare dal plot, le frecce, in questi punti, risultano avere la stessa direzione, cioè verso la posizione del goal. Tanto è vero che in corrispondenza della posizione *target* (goal) il potenziale è minimo poiché, essendo che le frecce stesse rappresentano l'antigradiente $-\nabla U$, sarà lo stesso $-\nabla U$ che dirigerà il robot mobile verso il punto di minimo del potenziale associato all'environment considerato. Infatti, è studio di molte ricerche, la risoluzione del *problema del minimo locale*, cioè quelle configurazioni in cui potenziale attrattivo e repulsivo si azzerano a vicenda creando un falso minimo globale e, pertanto, un falso goal dell'ambiente, confondendo il robot mobile e facendolo giungere in posizioni target non valide. In questo caso si può notare come, per

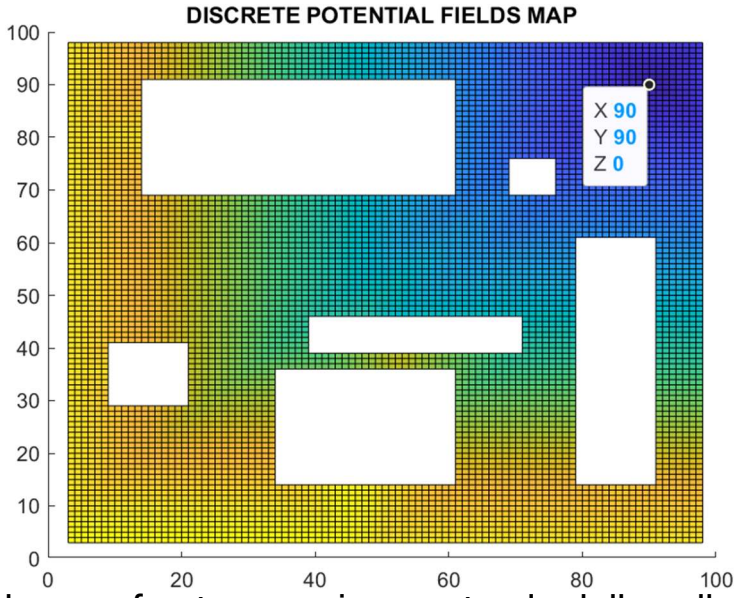




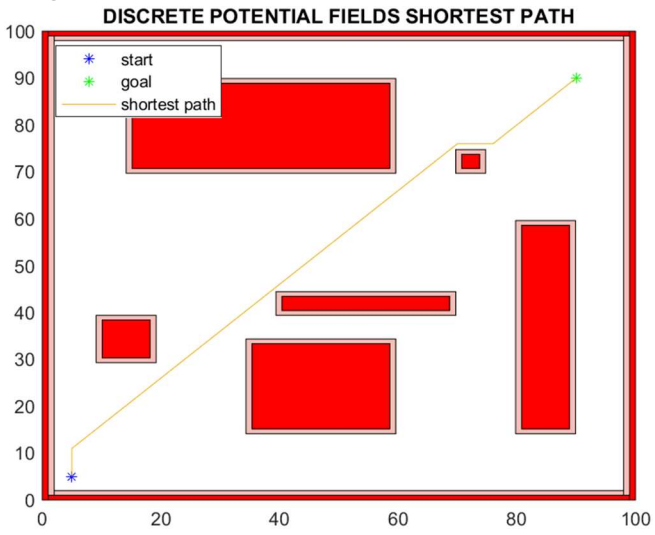
questa determinata configurazione di environment, il problema del minimo locale non si presenta.

Discrete Potential Fields

Questa tecnica consiste nell'utilizzo di una griglia discreta dove le celle corrispondenti agli ostacoli presentano un valore elevato. Nella posizione di goal corrisponde un valore nullo tale che le celle adiacenti sono definite mediante le celle vicine. Il metodo in questione si basa sull'1-adiacenza che caratterizza la "Manhattan-distance" e, pertanto, utilizzando la wavefront expansion, partendo dalla cella di goal etichettata con 0 e incrementando l'etichetta di una unità per le celle adiacenti non ancora etichettate. Inoltre, si può notare come nel punto di goal $goal = [90\ 90]$ sia presente il valore 0 come prima citato. Dopo aver determinato la mappa corrispondente, l'algoritmo, tramite la function *search_path*, determina il percorso che il robot mobile, sulla base della conoscenza di questa mappa, dovrà intraprendere dalla posizione iniziale di start fino alla posizione finale di goal. Nello specifico, fin quando esso non troverà una destinazione, proverà a cercare iterativamente un valore minore o, in caso peggiore, un valore uguale a quello corrente.



In questo caso, considerando l'environment in questione, l'algoritmo genera la seguente traiettoria:

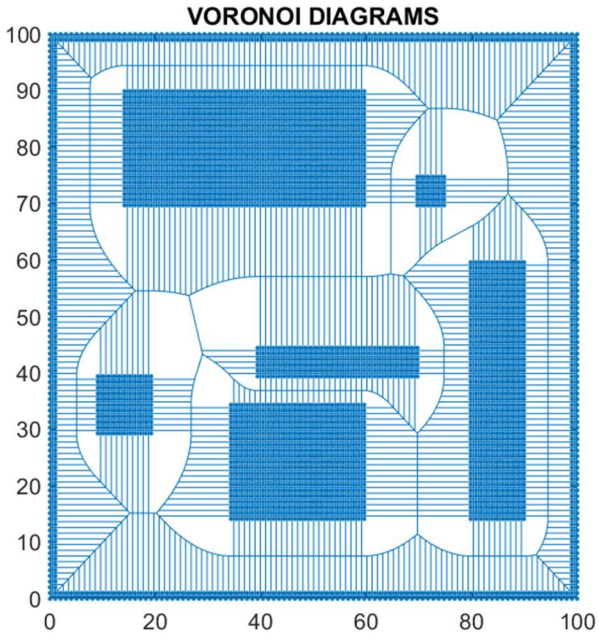


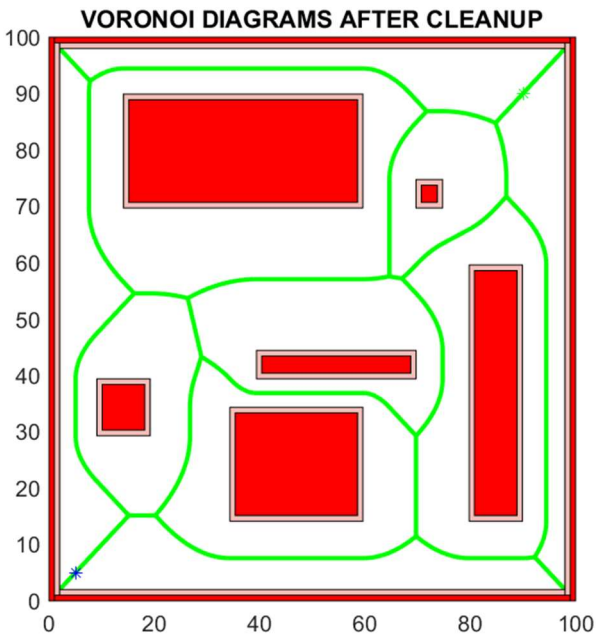
Si può notare come essa sia definita per spezzate poiché il robot, sulla base della conoscenza della matrice in questione e considerando l'algoritmo prima citato, considera cella per cella sempre il valore minimo o, nel peggiore dei casi, uguale delle celle adiacenti ad esso. Questo significa che l'andamento della traiettoria non potrà essere "dolce" perché lo spostamento del robot avviene per celle adiacenti che si trovano a nord, est, ovest o sud come

definiti nello script. Infatti, si può notare come la traiettoria ottenuta non tiene conto del fatto che il robot procede in posizioni relativamente vicine agli ostacoli in determinati punti. Anzi, in alcuni punti quasi li costeggia ma comunque si può stare tranquilli poiché gli ostacoli sono stati precedentemente ingrossati adeguatamente.

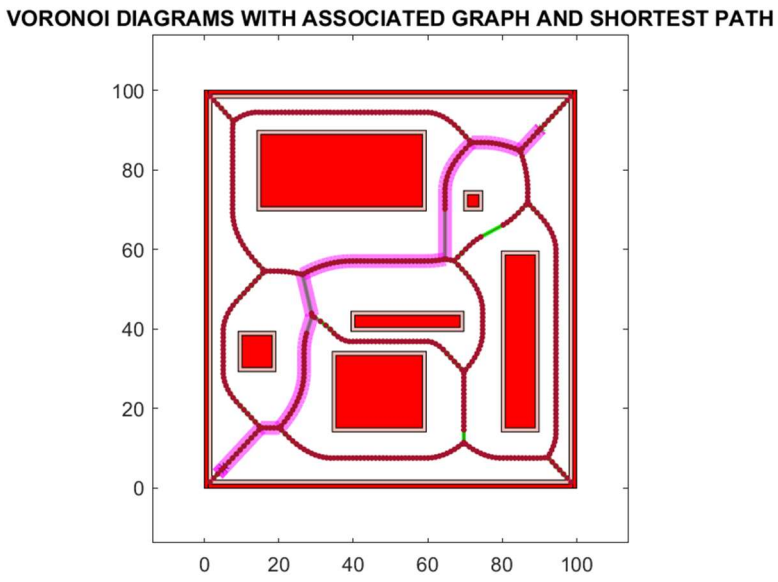
Voronoi Diagrams

Questa tecnica consiste nel partizionare l'ambiente in regioni in base alla posizione degli ostacoli. Ogni partizione è costituita di tratti rettilinei (lato/lato o





vertice/vertice) o parabolici (lato/vertice). Pertanto, viene inizialmente generato il diagramma di Voronoi corrispondente all'environment (considerando le griglie associate all'ambiente con ostacoli) in questione senza alcuna "ripulitura" del diagramma (figura 1). Inoltre, vengono salvati i vertici corrispondenti ottenuti dopo aver invocato la function *voronoi* di MATLAB. Successivamente, viene effettuata una "ripulitura" del diagramma di Voronoi da tutti i tratti rettilinei superflui che intersecano gli ostacoli lasciando, invece, le regioni con le "frontiere" corrispondenti (figura 2). Sarà proprio l'insieme di alcune di queste frontiere che



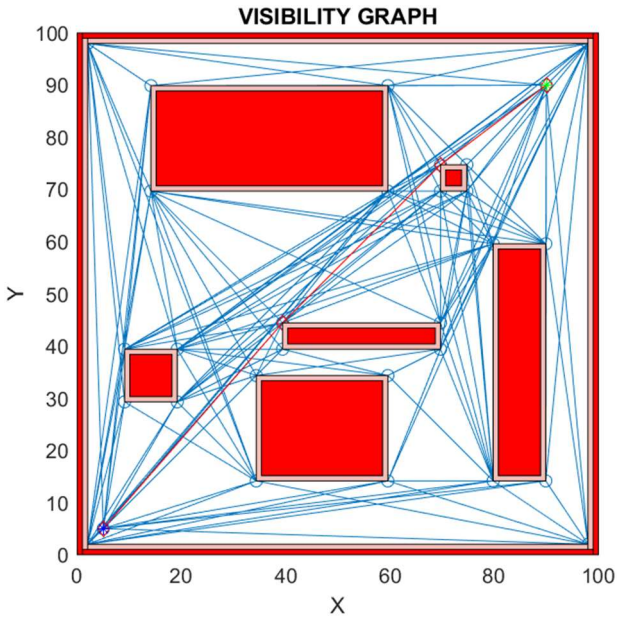
caratterizzerà la traiettoria ottenuta con questa tecnica di path planning. Costruendo la matrice di adiacenza, corrispondente all'insieme dei vertici considerati, e il grafo di connettività associato, si potrà applicare la funzione nativa di MATLAB *shortestpath* che restituirà il percorso minimo considerando il grafo in input ad essa. Si può notare come, una qualsiasi traiettoria che non sia quella minima, è pur sempre una traiettoria molto "morbida" e "curva".

85.3535	85.3535
86.3636	86.3636
87.3737	87.3737
88.3838	88.3838
89.3939	89.3939
90.4040	90.4040
90.0000	90.0000

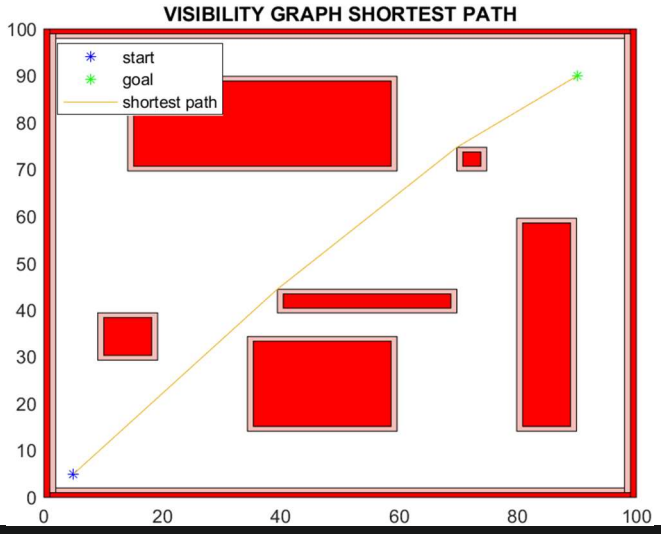
Si può notare come la tecnica in questione risulta avere una traiettoria molto "dolce" e precisa. Tanto è vero che il robot mobile riesce perfettamente a raggiungere la posizione di goal. Tanto è vero che essa viene definita come rete di cammini a *max clearance*, cioè avente i cammini più sicuri.

Visibility Graphs

Questa tecnica consiste nello sfruttare tutti i segmenti che connettono i vertici tra di loro tale che tali segmenti non intersecano gli ostacoli. L'idea è quella di



sfruttare una parte dell'insieme totale dei segmenti per ottenere una traiettoria che permette al robot mobile di raggiungere la posizione target. Per creare il grafo di connettività associato, bisogna considerare tutti i vertici degli ostacoli presenti all'interno dell'ambiente. Successivamente, tramite la function *graph_creation*, viene creato il grafo corrispondente tramite la matrice di adiacenza creata con la function *adjacency_matrix*. Il grafo sarà caratterizzato da un insieme di nodi e da un insieme di pesi. I nodi corrisponderanno ai vertici degli ostacoli presenti nell'environment, mentre i pesi corrisponderanno alla distanza euclidea tra tali vertici. Pertanto, considerando questi due insiemi e la function nativa *shortestpath*, verrà generata la traiettoria associata a questa tecnica di path planning. Si può notare come la traiettoria generata presenti quasi cambi di direzione nulli. Questo è dovuto al fatto che la configurazione di ostacoli



scelta permette una traiettoria quasi lineare. Ovviamente in contesti diversi, tale tecnica di path planning potrebbe generare una traiettoria caratterizzata da un insieme di spezzate che cambia continuamente direzione. Supponendo, infatti, di non considerare il percorso minimo e di considerare il resto delle traiettorie possibili all'interno del grafo di visibilità generato, una qualsiasi altra traiettoria sarebbe caratterizzata da spezzate terribilmente diverse tra di loro che porterebbe al robot mobile a seguire una traiettoria molto "spigolosa" e non "dolce" come, invece, viene generato tramite la tecnica del diagramma di Voronoi.

Control

L'obiettivo del controllo è quello di progettare un input di controllo appropriato per il robot mobile in modo da guidare la sua posa su una determinata traiettoria o stato target. Nello specifico il controllo è costituito da due parti fondamentali: il trajectory tracking e la posture regulation.

Trajectory Tracking

Per risolvere il problema di tracking, è necessario che la traiettoria desiderata $X^*(t), \dot{X}^*(t)$ sia ammissibile per il modello cinematico scelto tale che esista un robot di riferimento virtuale del tipo:

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

per una qualche scelta di input di riferimento $v^*(t)$ e $\omega^*(t)$. Pertanto, dovrà soddisfare il nonholonomic costraint.

Per la generazione della traiettoria di riferimento è stata utilizzata la function nativa *trapveltraj*. Tale function ha permesso la generazione di $x^*, \dot{x}^*, \ddot{x}^*, y^*, y^*, \ddot{y}^*, \theta^*$ considerando l'insieme di punti della traiettoria, ottenuto tramite il path planning, e il numero di *samples* da considerare. Quest'ultimo è stato ottenuto tramite il rapporto tra la lunghezza della traiettoria e un tempo di campionamento T_s, cioè un sampling time. Ovviamente, per ogni tecnica di path planning considerata è stata considerata una discretizzazione differente poiché l'insieme di punti ottenuto per ognuna non è uguale. Ad esempio, per la tecnica del grafo di visibilità è stata considerata una doppia discretizzazione poiché il numero di punti che caratterizzava il percorso minimo era davvero esiguo. Per considerare ogni tecnica di controllo bisogna calcolare l'errore di inseguimento rispetto alla traiettoria di riferimento. Tale errore è ottenuto sulle coordinate come differenza tra coordinata effettiva e coordinata di riferimento.

Infatti, le tre tecniche di controllo considerate (*approximated linearization, non-linear control e input-ouput linearization*) basano i loro calcoli sugli errori rispetto alle coordinate. Per ogni controllo, verranno effettuati dei calcoli sulla base di alcuni coefficienti e del modello considerato e, infine, verrà costruito il modello $\dot{x}^*(t), \dot{y}^*(t), \dot{\theta}^*(t)$ tenendo conto del tempo di campionamento in questione. Questo è dovuto al fatto che il sistema è stato fatto evolvere manualmente senza l'ausilio di alcuna routine presente in MATLAB. Un'altra possibile soluzione è quella di creare delle funzioni corrispondenti per ogni controllo, associare ad esse un modello dipendente dal tempo e successivamente darle in input ad una routine nativa quale *ode45* che sceglierà un tempo di campionamento adatto e la farà evolvere nel tempo.

Approximated Linearization

La linearizzazione approssimata si basa principalmente nel considerare l'errore di inseguimento. Più l'errore tende a zero più la traiettoria tende alla traiettoria voluta. In genere, si parte da una posizione iniziale diversa da quella effettiva e che non sia troppo lontana dalla traiettoria, cioè con un errore iniziale non troppo grande affinché funzioni. Presumibilmente, la traiettoria tenderà alla traiettoria effettiva. Anche se non c'è nessuna dimostrazione di convergenza è verosimile pensare che funzioni. In questo caso verranno considerati dei coefficienti a, δ, K_1, K_2 e K_3 rispettivamente pari a 1, 3, $2 \cdot \delta \cdot a$, $\frac{a^2 - \omega^{*2}}{v^*}$ e $2 \cdot \delta \cdot a$.

Non-Linear Control

Il controllo non linearizzato è una tipologia di controllo che segue una legge di inserimento di traiettoria persistente, cioè le traiettorie devono essere persistenti (non deve esserci un momento in cui il robot mobile si ferma e riparte). In genere, a differenza della linearizzazione approssimata, il robot mobile può partire da una qualsiasi condizione/posizione iniziale perché comunque si riuscirà a portare l'errore a zero in maniera asintotica. In questo caso le variabili coinvolte sono K_1, K_2 e K_3 con valori rispettivamente pari a vettore di uno, 1 e vettore di uno.

Input-Output Linearization

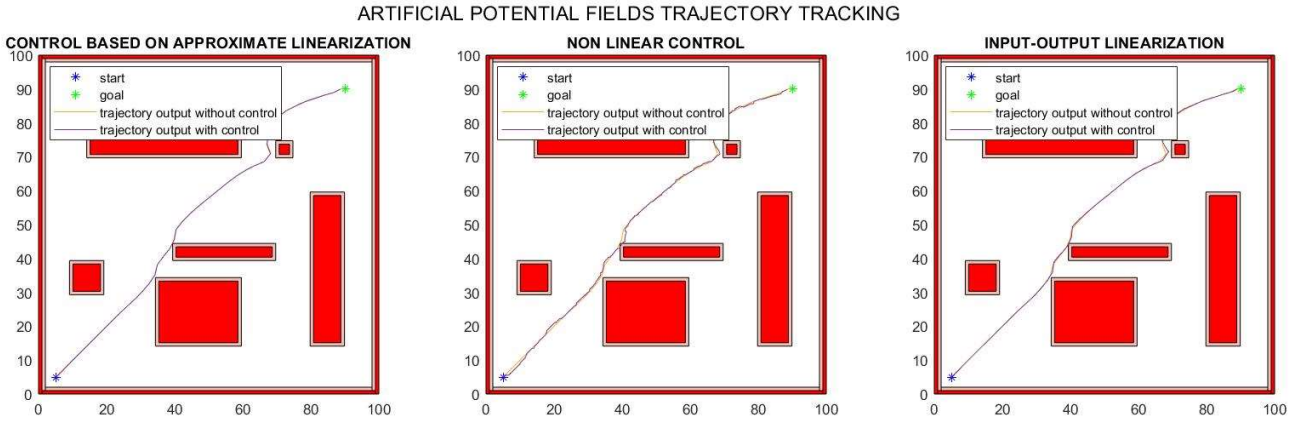
La linearizzazione input-output (feedback linearization) prevede una legge di controllo non lineare, via feedback, basata su ingressi virtuali tale che la relazione tra questi ultimi e le uscite del sistema sia lineare. In questo caso i coefficienti considerati sono b, K_{x} e K_{y} aventi valori rispettivamente pari a 0.5, 1 e 1. Bisogna far notare che il coefficiente b deve essere diverso da zero

poiché essendo che, si deve calcolare la matrice inversa associata al modello in questione, il suo determinante deve essere, pertanto, diverso da zero affinché sia invertibile.

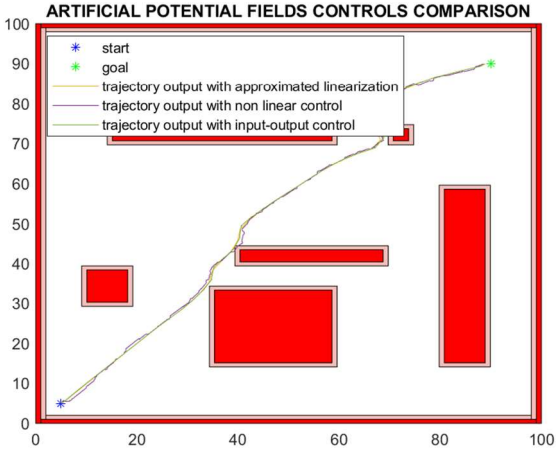
Si può notare, qui di seguito, come la risoluzione del problema di trajectory tracking è stato affrontato in ogni path planning considerato all'interno del progetto.

ARTIFICIAL POTENTIAL FIELDS

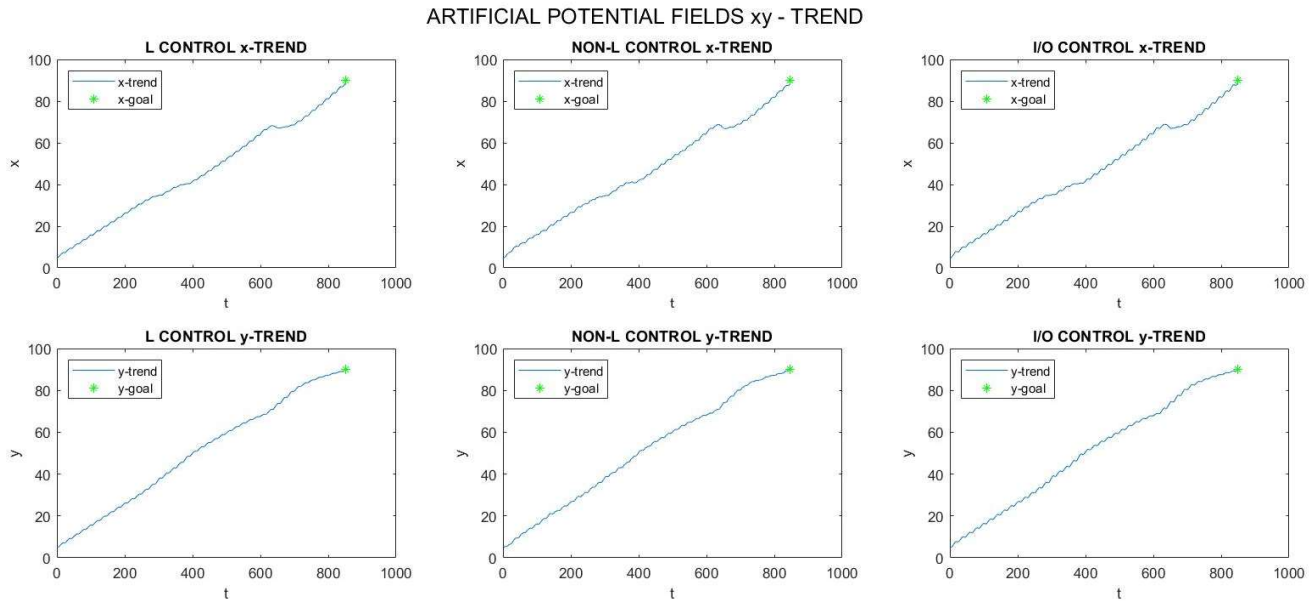
Considerando il path planning basato sulla tecnica degli Artificial Potential Fields, si può notare come tutte e tre le traiettorie ottenute tramite le tre diverse tipologie di controllo riescano ad approssimare al meglio la traiettoria



considerata. Si può notare che la traiettoria generata tramite controllo basato su approssimazione lineare riesce quasi perfettamente ad approssimare la traiettoria di riferimento. Questo perché è stato scelto un $\delta = 3$ che ha garantito una traiettoria più "dolce" e con meno oscillazioni. Per quanto riguarda, invece, la traiettoria ottenuta tramite controllo non lineare, questa risulta essere quella che presenta maggiori oscillazioni rispetto alle altre. Infatti, effettuando un plot dinamico si può notare come il robot mobile sia soggetto ad innumerevoli oscillazioni. Anche se la traiettoria in questione riesce egregiamente ad evitare gli ostacoli e, alla fine, raggiungere la posizione target. Infine, il controllo basato su linearizzazione input-output è quello che approssima al meglio la traiettoria di riferimento. Si può fare un confronto tra gli andamenti sia delle x che delle y per tutti e tre i controlli



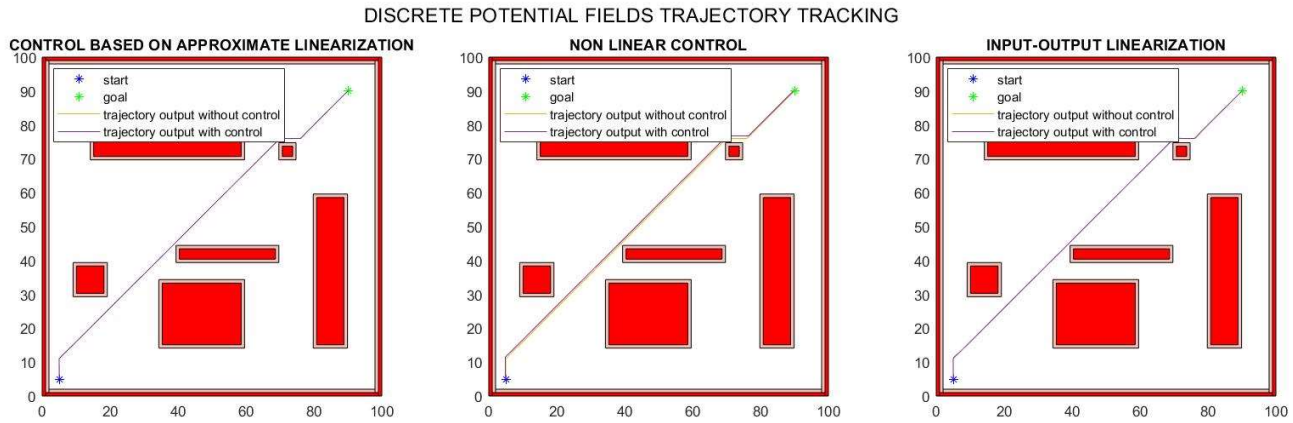
così da vedere più da vicino effettivamente quale andamento si discosta di più dall'andamento di riferimento e, caso mai, di quanto lo fa:



si può notare come l'unico andamento che risulta essere non arrivare perfettamente nella coordinata di riferimento è l'andamento delle x nel trajectory tracking basato su controllo non lineare.

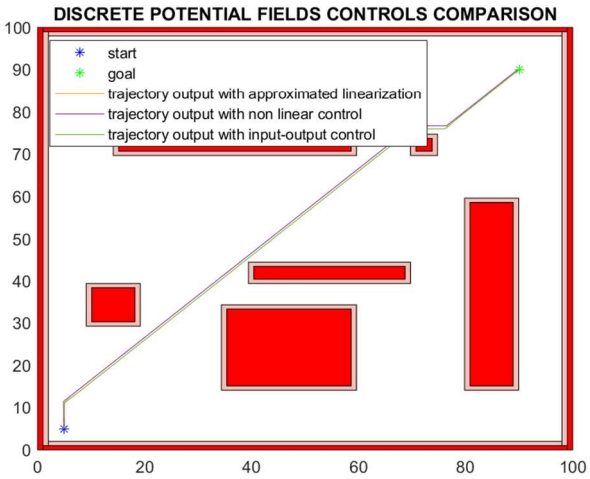
DISCRETE POTENTIAL FIELDS

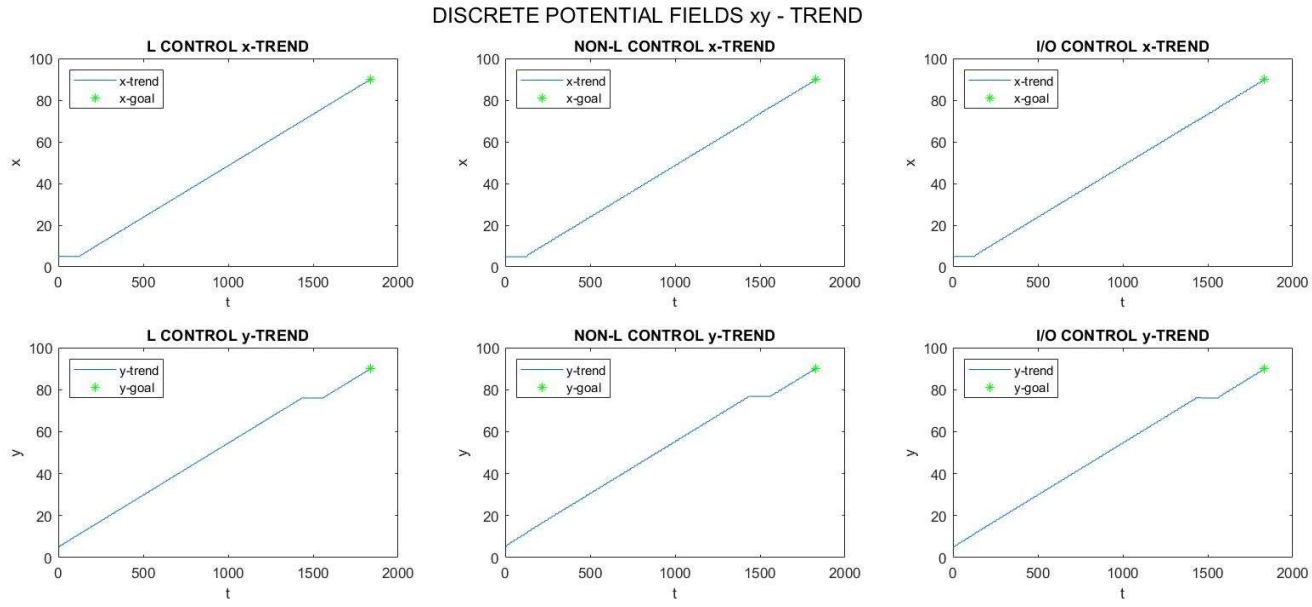
Considerando il path planning basato sulla tecnica dei Discrete Potential



Fields, si può notare come tutte e tre le traiettorie ottenute tramite le tre diverse tipologie di controllo riescano ad approssimare al meglio la traiettoria considerata. In questo caso rispetto alla tecnica precedente considerata, il numero di punti considerato risulta essere maggiore. Infatti, si può notare come la traiettoria ottenuta tramite il controllo basato su linearizzazione approssimata, riesce ad approssimare praticamente in maniera perfetta la traiettoria di riferimento. Anche la traiettoria ottenuta tramite feedback linearization risulta essere praticamente sovrapposta a quella di riferimento.

Per quanto riguarda, invece, la traiettoria ottenuta tramite controllo non lineare, essa risulta essere leggermente discostata da quella di riferimento. Certamente aumentando sempre di più il K_2 associato al controllo non lineare, tale traiettoria migliorerà sempre di più. Infatti, per K_2 minori si avranno scostamenti maggiori dalla traiettoria di riferimento. Si può notare comunque che essa riesce ad evitare gli ostacoli e ad arrivare in un intorno della posizione target. Si può notare da più vicino come praticamente in tutti e tre i controlli l'andamento sia delle coordinate x che delle coordinate y giunge in un intorno della posizione target come già visto nel grafico precedente. L'andamento di tutti i trend delle coordinate è pressocché lineare. Questo è dovuto al fatto che la traiettoria generata è un insieme di spezzate che comunque seguono un andamento "quasi" lineare. Infatti, dalle figure precedenti è possibile notare

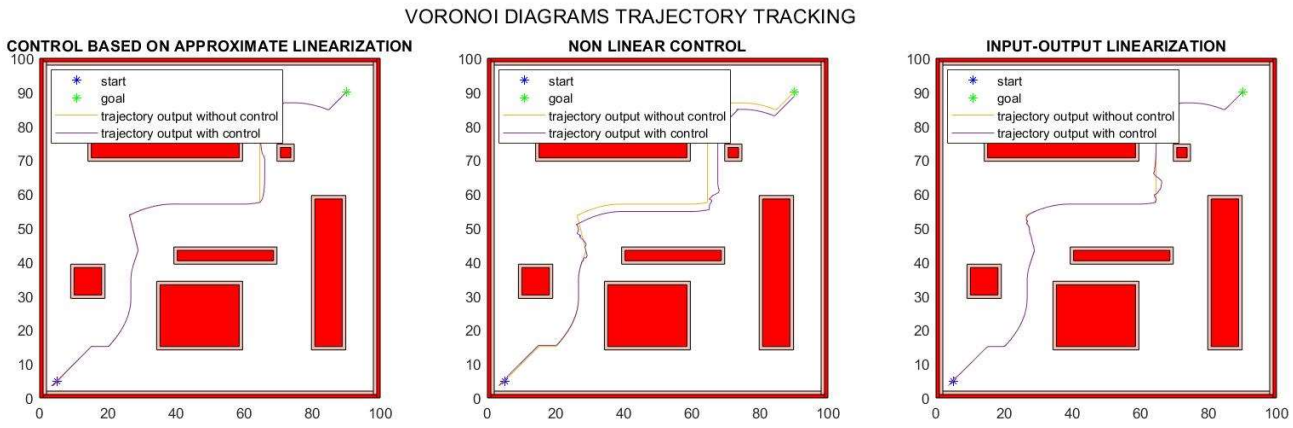




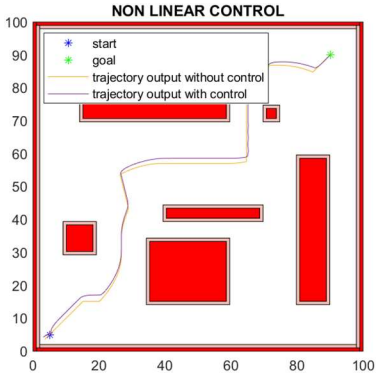
come la traiettoria non è composta da nessuna curva o andamento particolare tranne per quel piccolo tratto iniziale che subito dopo si assesta sull'andamento lineare.

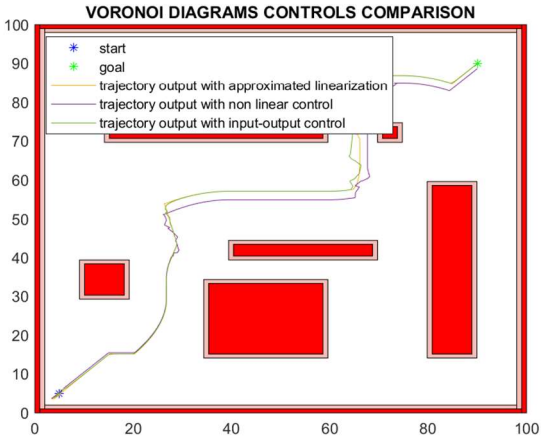
VORONOI DIAGRAMS

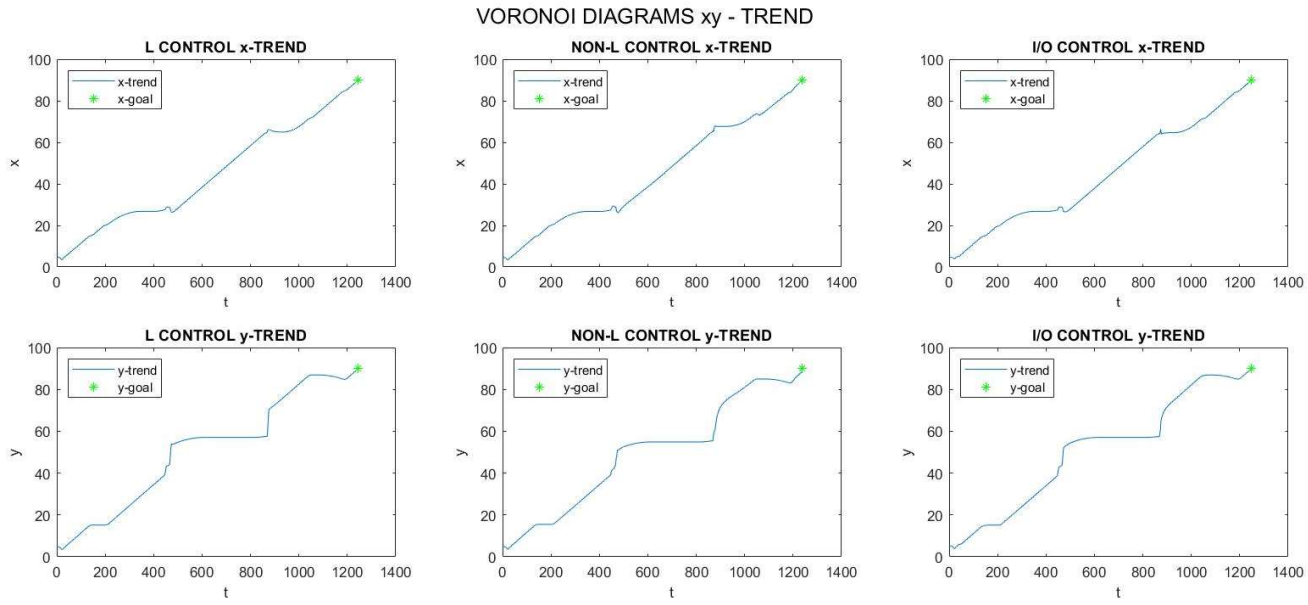
Considerando il path planning basato sulla tecnica del Voronoi Diagrams, si



può notare come tutti e tre i controlli riescono ad approssimare la traiettoria di riferimento. Il controllo basato su approssimazione lineare riesce ad approssimare quasi in maniera perfetta tranne che per un piccolo tratto verticale finale dove è presente una leggera curvatura. Il controllo basato, invece, sul non linearizzato presenta un modesto errore nella parte intermedia e finale della traiettoria. Questa crescita dell'errore dipende dal coefficiente K_2 scelto. In questo caso più è grande più l'errore cresce. Infatti, assegnandogli un valore minore, ad esempio 1, si può notare come, effettivamente il controllo in questione, riesce quasi perfettamente ad approssimare la traiettoria di riferimento. Infine, si può notare come il controllo basato sulla linearizzazione input-output riesca ad approssimare quasi perfettamente la traiettoria tranne per il tratto verticale finale che presenta una curvatura "spigolosa" iniziale su codesto tratto. Si può affermare che comunque tutti e tre i controlli garantiscono il non impatto contro gli ostacoli ed, inoltre, garantiscono di giungere in un intorno del goal. Infine, dalle seguenti figure, si può vedere più nel dettaglio l'andamento delle coordinate x e y dalle quali è possibile notare come effettivamente il robot mobile riesca a raggiungere le coordinate x e y del target come richiesto. Si può notare, comunque, la "spigolosità" in alcuni punti dei trend sia delle coordinate x che delle coordinate y. Questo è dovuto al fatto che, rispetto ai path planning analizzati



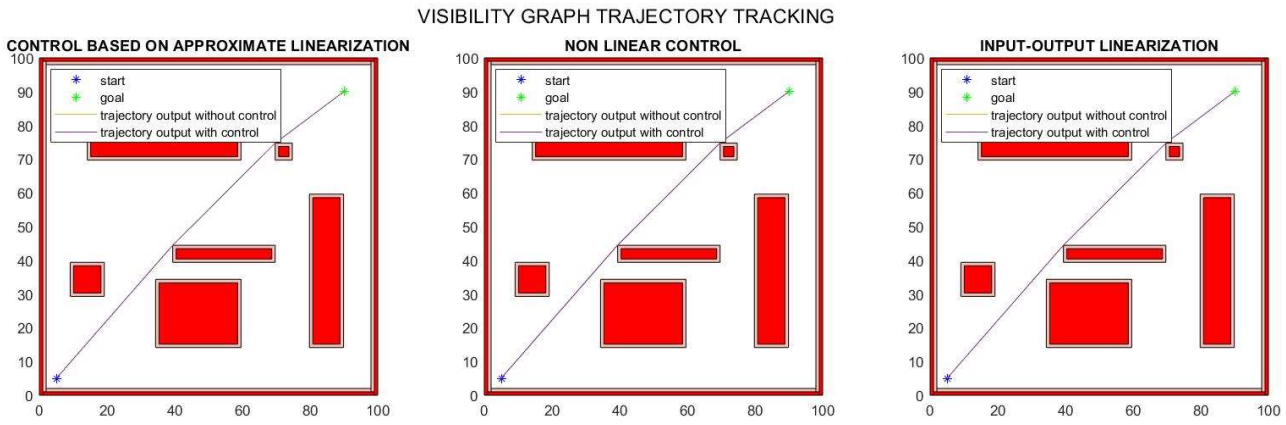




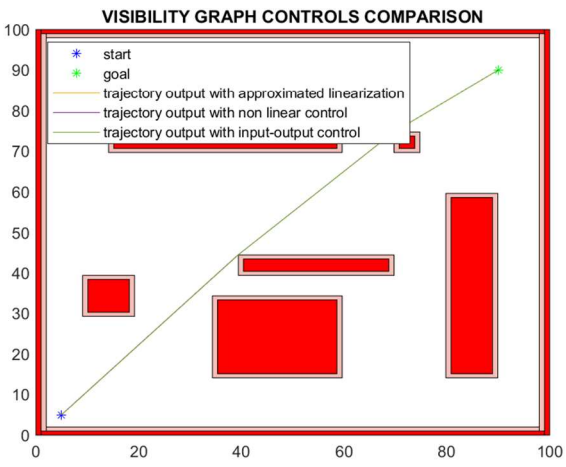
precedentemente, tale tecnica presenta variazioni maggiori in ambito di traiettoria e, quindi, più soggetta ad errori e "curvature particolari".

VISIBILITY GRAPHS

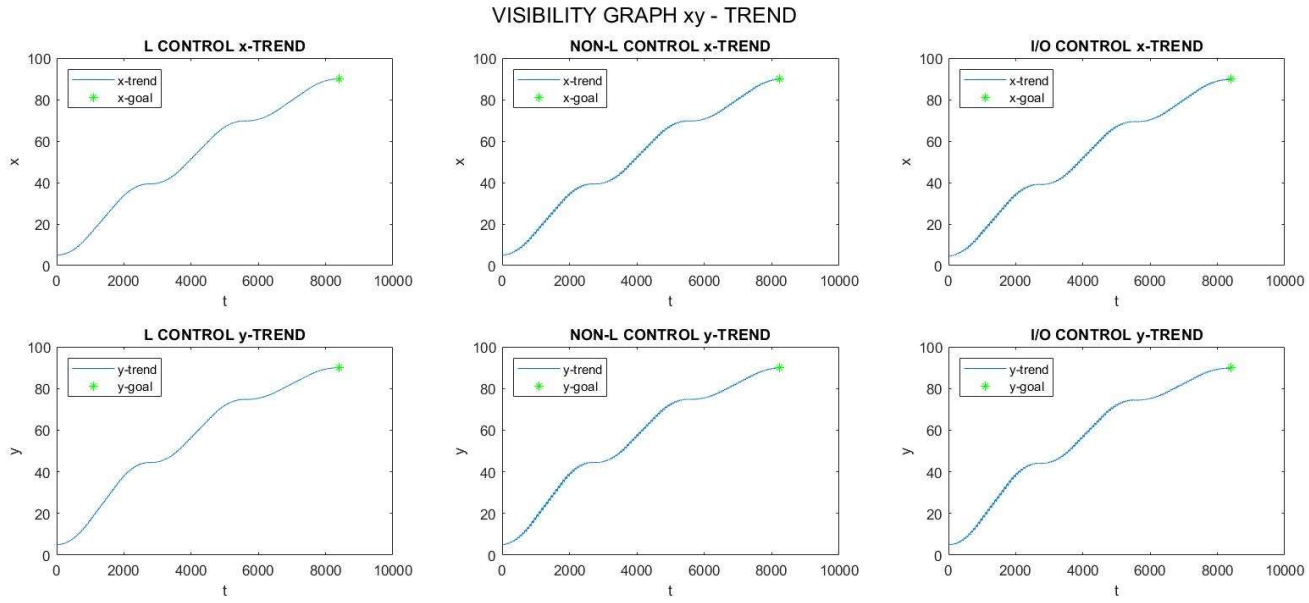
Considerando il path planning basato sulla tecnica del Visibility Graphs, si può



notare come tutti e tre i controlli approssimano in maniera perfetta la traiettoria di riferimento. In questo caso, quindi, anche il controllo basato su approssimazione lineare riesce ad inseguire la traiettoria di riferimento senza alcuna variazione o curvatura particolare. Si può affermare che questo è dovuto anche al fatto che la traiettoria di riferimento da inseguire è una traiettoria praticamente lineare con delle piccole variazioni di coefficiente angolare verso il tratto finale giusto per "affinare" l'andamento verso la posizione target.



Analizzando più nel dettaglio, si può notare come, tutte e tre i controlli



garantiscono un andamento delle coordinate x e y tale da raggiungere l'intorno desiderato della posizione di goal.

Posture Regulation

L'obiettivo della posture regulation è quello di guidare il robot mobile verso una configurazione desiderata. Tale configurazione prevede l'arrivo nelle coordinate esatte della posizione target con un determinato orientamento (angolazione). Tanto è vero che si è potuto notare come, la trajectory tracking induca il robot ad assumere una determinata traiettoria affinché esso riesca ad evitare gli ostacoli presenti nell'ambiente ma senza garantire che esso arrivi nelle coordinate esatte della posizione di goal. Infatti, analizzando i risultati ottenuti, in alcuni casi è capitato che il robot si sia fermato in un intorno della posizione target con una angolazione generica, cioè non prevedibile. Pertanto, sarà la posture a mitigare questi due ultimi problemi appena elencati affinché il robot mobile riesca a "parcheggiarsi" nelle coordinate esatte e con una determinata angolazione fissata. Esistono due tipologie di approccio: la cartesian regulation e la complete regulation. Verranno analizzate qui di seguito entrambe le tecniche ed, inoltre, verranno analizzati i risultati ottenuti.
In generale, è stato considerato un tempo di simulazione pari a 200 in modo da discretizzare nel migliore dei modi il tratto finale di interesse ed un'angolazione finale pari a $270°$. Il punto iniziale, ovviamente, sarà il punto finale ottenuto nella traiettoria con trajectory tracking, mentre quello finale sarà descritto dalle coordinate della posizione target. Si deve notare che non tutte

le traiettorie precedentemente descritte hanno bisogno di una discretizzazione così notevole. Infatti, dai risultati ottenuti si può notare come in alcuni casi il robot mobile riesce a raggiungere in media la posture in meno di 100 iterazioni. Anzi, per alcune traiettorie nel caso migliore servono soltanto 50 iterazioni. Tanto è vero che, il plot dinamico previsto è stato limitato ad un numero massimo di iterazioni affinché esso non appesantisca la simulazione, poiché la posture comunque riesce ad essere raggiunta in queste poche iterazioni. Inoltre, a valle dei risultati ottenuti sia in ambito di posizione che di orientamento, sono stati implementate alcune approssimazioni e alcuni controlli sia nella cartesian regulation sia nella complete regulation. Essendo che i calcoli sono stati ottenuti tramite diverse operazioni, è lecito pensare che i risultati finali possano presentare qualche cifra decimale e che rispetto alla posture obiettivo possano aver subito una certa variazione di errore. Pertanto, dopo aver ottenuto l'insieme delle configurazioni che caratterizzano la posture, si arrotondano le cifre decimali e si effettua un controllo considerando una soglia di errore pari ad 1. Se il valore assoluto della differenza tra configurazione finale ottenuta e quella obiettivo è minore di tale soglia di errore allora, significa che, i calcoli effettuati dall'algoritmo della posture sono corretti e sono andati a buon fine, altrimenti si termina il programma con un errore perché non ha più senso far terminare l'esecuzione avendo ottenuto calcoli errati.

```
iteration=0;
max_iterations=100;

for i = 1 : size(evolution) - 1

    if and(isequal(round(evolution(i,1)),goal(1)),...
           isequal(round(evolution(i,2)),goal(2)))
        iteration=iteration+1;
    end

    plot(evolution(i,1), evolution(i,2), "or", "MarkerSize", 3);
    triangle = plot_triangle(...
              [evolution(i,1),evolution(i,2)], evolution(i,3));
    pause(0.0001);

    if isequal(iteration, max_iterations)
        return
    end

    delete(triangle);

end
```

Cartesian Regulation

La cartesian regulation è una versione semplificata di posture regulation dove è richiesta soltanto la regolazione della posizione del robot mobile (position regulation). In altre parole, l'obiettivo finale è di guidare il robot verso una posizione cartesiana senza specificare esplicitamente un orientamento finale. Nello specifico sono stati considerati i coefficienti K_1 e K_2 con valore rispettivamente pari a 1 e 1.

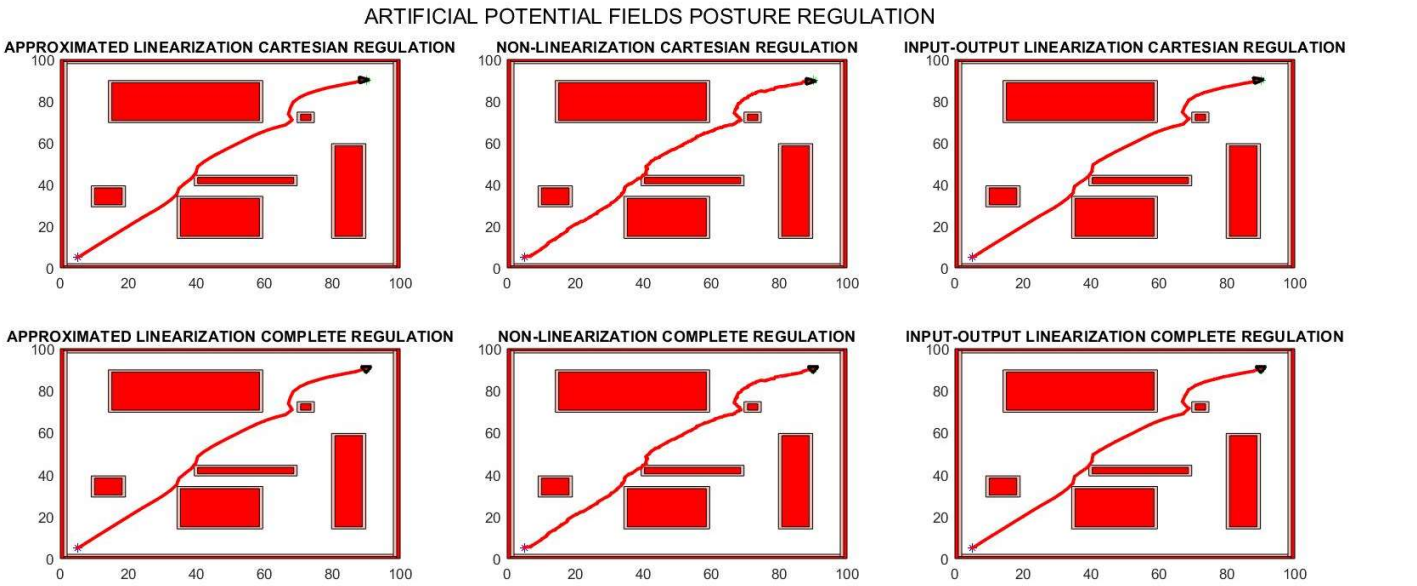
Complete Regulation

Tale legge garantisce che i segnali considerati tendano asintoticamente a zero. Pertanto, essi varranno almeno un certo ε tranne all'infinito in cui varranno zero. Ovviamente, la complete regulation è per l'appunto quella

"completa" perché prevede di risolvere sia il problema di regolazione della posizione sia il problema di regolazione dell'orientamento. Infatti, essa verrà approfondita più nel dettaglio essendo quella che dà maggiori "risultati".
Nello specifico sono stati considerati i coefficienti K_1, K_2 e K_3 con valore rispettivamente pari a 1, 1 e 1. I coefficienti K_1 e K_3 sono stati utilizzati per regolare la posizione del robot mobile mentre il coefficiente K_3 l'orientamento. Infatti, si può notare come la cartesian regulation non presenti un coefficiente legato all'orientamento poiché essa prevede soltanto una regolazione sulla posizione e non sull'orientamento.

ARTIFICIAL POTENTIAL FIELDS

Si può notare come la posizione sia stata raggiunta sia nel caso della cartesian




```
APPROXIMATED LINEARIZATION cartesian regulation:

ans =

    90    90   180
```

```
NON-LINEARIZATION cartesian regulation:

ans =

    90    90   185
```

```
INPUT-OUTPUT LINEARIZATION cartesian regulation:

ans =

    90    90   187
```

```
APPROXIMATED LINEARIZATION complete regulation:

ans =

    90    90   270
```

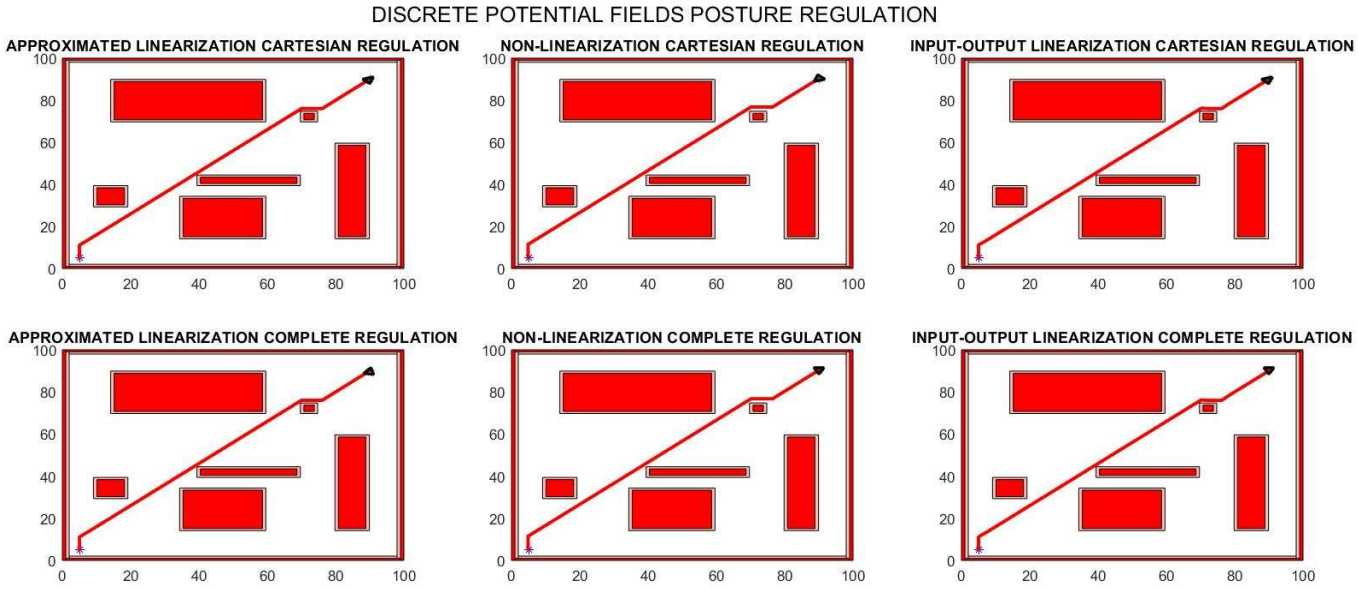
```
NON-LINEARIZATION complete regulation:

ans =

    90    90   270
```

```
INPUT-OUTPUT LINEARIZATION complete regulation:

ans =

    90    90   270
```

regulation sia nel caso della complete regulation. Per quanto riguarda, invece, l'orientamento, esso è stato raggiunto soltanto nella complete regulation. Infatti, il coefficiente K_3 è riuscito a regolare perfettamente affinché raggiungesse i $270°$ previsti.

DISCRETE POTENTIAL FIELDS

Si può notare come la posizione sia stata raggiunta sia nel caso della cartesian




```
APPROXIMATED LINEARIZATION cartesian regulation:

ans =

    90    90   259
```

```
NON-LINEARIZATION cartesian regulation:

ans =

    90    90   289
```

```
INPUT-OUTPUT LINEARIZATION cartesian regulation:

ans =

    90    90   225
```

```
APPROXIMATED LINEARIZATION complete regulation:

ans =

    90    90   270
```

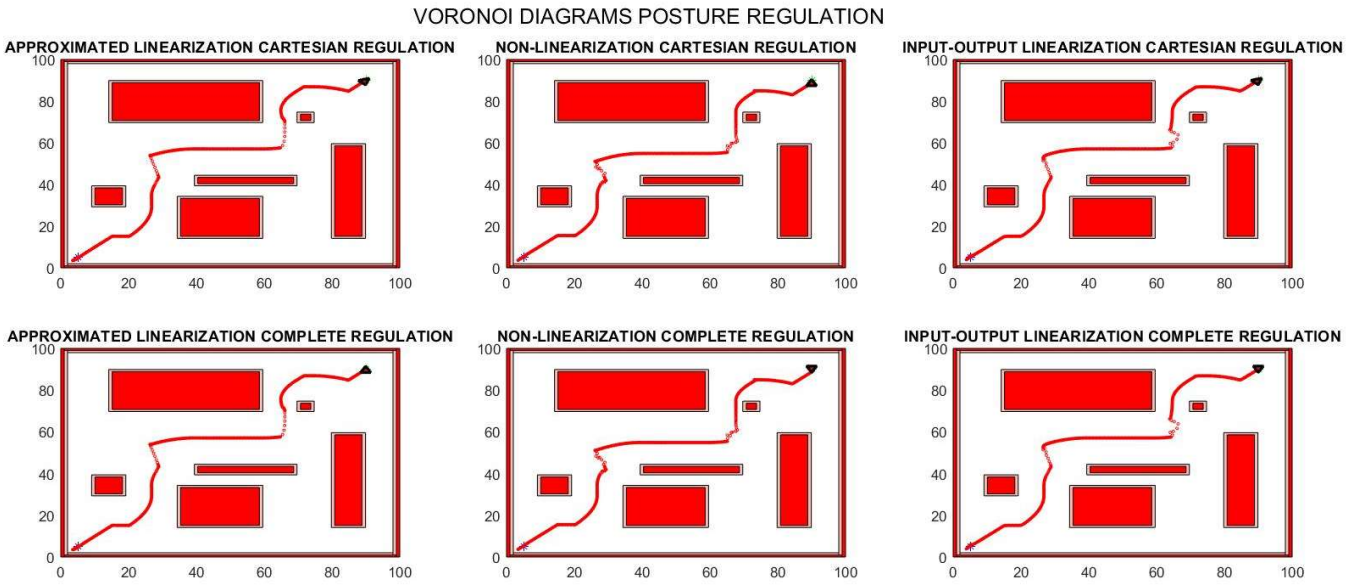
```
NON-LINEARIZATION complete regulation:

ans =

    90    90   270
```

```
INPUT-OUTPUT LINEARIZATION complete regulation:

ans =

    90    90   270
```

regulation sia nel caso della complete regulation. Per quanto riguarda, invece, l'orientamento, esso è stato raggiungo soltanto nella complete regulation. Anche se l'orientamento raggiunto nella cartesian regulation, a differenza del precedente path planning considerato, non si discosta di molto rispetto a quello obiettivo.

VORONOI DIAGRAMS

Si può notare come la posizione sia stata raggiunta sia nel caso della cartesian




```
APPROXIMATED LINEARIZATION cartesian regulation:

ans =

    90    90   226
```

```
NON-LINEARIZATION cartesian regulation:

ans =

    90    90    91
```

```
INPUT-OUTPUT LINEARIZATION cartesian regulation:

ans =

    90    90   225
```

```
APPROXIMATED LINEARIZATION complete regulation:

ans =

    90    90   270
```

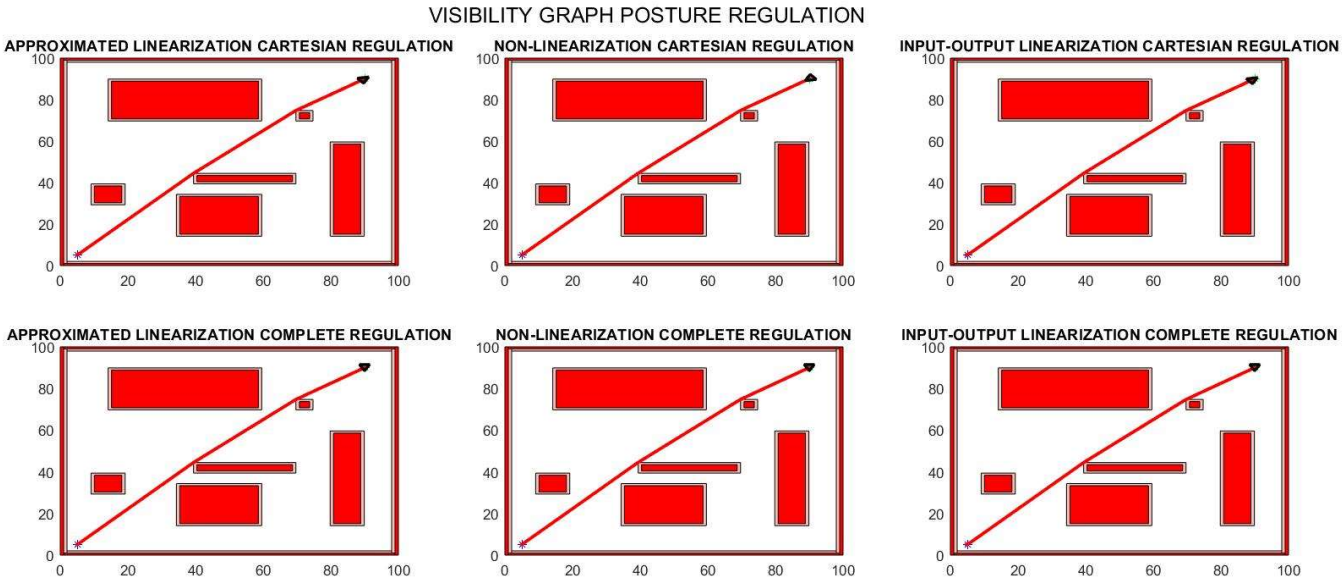
```
NON-LINEARIZATION complete regulation:

ans =

    90    90   270
```

```
INPUT-OUTPUT LINEARIZATION complete regulation:

ans =

    90    90   270
```

regulation sia nel caso della complete regulation. Per quanto riguarda, invece, l'orientamento, esso è stato raggiunto soltanto nella complete regulation. In questo caso, tranne nel controllo di tipologia non linearizzato, l'orientamento ottenuto nella cartesian regulation non si discosta di molto rispetto a quello obiettivo.

VISIBILITY GRAPHS

Si può notare come la posizione sia stata raggiunta sia nel caso della cartesian




```
APPROXIMATED LINEARIZATION cartesian regulation:

ans =

    90    90   217


APPROXIMATED LINEARIZATION complete regulation:

ans =

    90    90   270
```

```
NON-LINEARIZATION cartesian regulation:

ans =

    90    90   310


NON-LINEARIZATION complete regulation:

ans =

    90    90   270
```

```
INPUT-OUTPUT LINEARIZATION cartesian regulation:

ans =

    90    90   217


INPUT-OUTPUT LINEARIZATION complete regulation:

ans =

    90    90   270
```

regulation sia nel caso della complete regulation. Per quanto riguarda, invece, l'orientamento esso è stato raggiunto soltanto nella complete regulation. In questo caso, rispetto alle tecniche di path planning analizzate precedentemente, l'orientamento ottenuto nella cartesian regulation presenta un errore maggiore rispetto all'orientamento obiettivo.