



Progetto di Robotica Mobile

Giorgio Ubbriaco 209899

Sommario

[Abstract 3](#_Toc114253209)

[Introduzione 3](#_Toc114253210)

[Robot Model 3](#_Toc114253211)

[Environment 4](#_Toc114253212)

[Path Planning 5](#_Toc114253213)

[Obstacles Shape 5](#_Toc114253214)

[Artificial Potential Fields 5](#_Toc114253215)

[Discrete Potential Fields 9](#_Toc114253216)

[Voronoi Diagrams 10](#_Toc114253217)

[Visibility Graphs 11](#_Toc114253218)

[Control 12](#_Toc114253219)

[Trajectory Tracking 12](#_Toc114253220)

[Approximated Linearization 13](#_Toc114253221)

[Non-Linear Control 13](#_Toc114253222)

[Input-Output Linearization 13](#_Toc114253223)

[Posture Regulation 14](#_Toc114253224)

[Cartesian Regulation 14](#_Toc114253225)

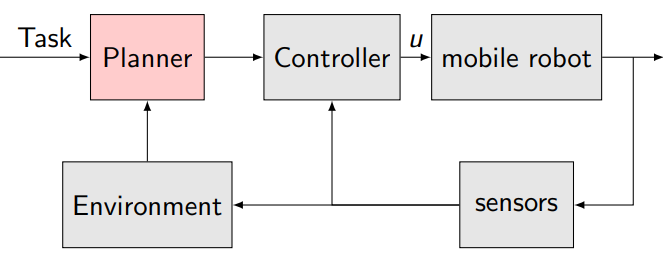
[Complete Regulation 14](#_Toc114253226)

# Abstract

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

# Introduzione

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 di 6 ostacoli di varie dimensioni. Il seguente schema di controllo riassume i task principali per la generica progettazione di un robot mobile:

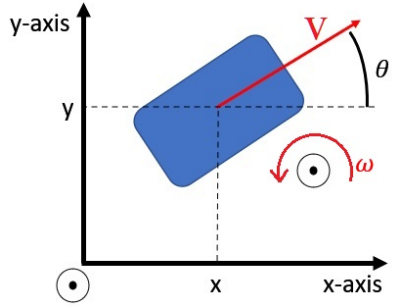


Schema di Controllo del Robot Mobile

# Robot Model

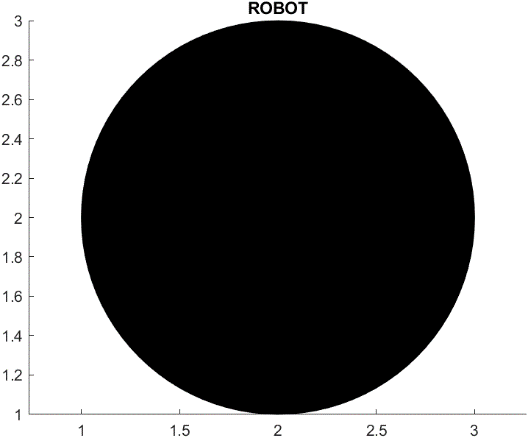
Un robot mobile di tipologia *uniciclo* è un veicolo avente una sola ruota orientabile. La configurazione di tale robot è descritta da , dove sono le coordinate cartesiane del punto di contatto della ruota con il suolo e è l’orientamento della ruota e, pertanto, del veicolo in questione rispetto all’asse x. Il vincolo del puro rotolamento della ruota è descritto dalla seguente relazione:

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:

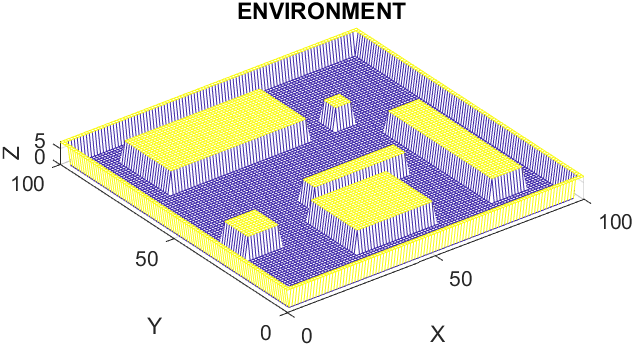


Unicycle Robot Model

dove è la velocità di “guida” (*driving velocity*) mentre 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

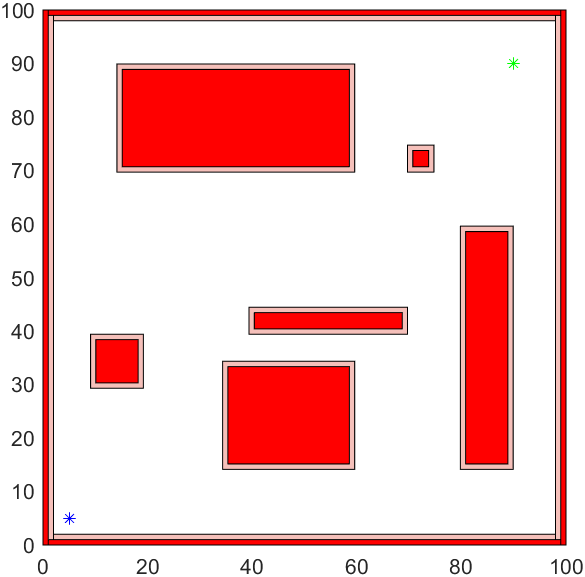
L’environment considerato prevede una larghezza ed una lunghezza rispettivamente pari a 100. Pertanto, la griglia binaria corrispondente in cui sono stati aggiunti gli ostacoli prevede un numero di righe di righe e di colonne pari al numero sopra indicato. I muri che delimitano l’ambiente sono stati 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.

3D Environment Representation

# 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 considerata 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 che 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:

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 ottenuto come sovrapposizione di un potenziale attrattivo verso la destinazione e di un campo repulsivo dalla regione dei -ostacoli. La pianificazione avviene per iterazioni: ad ogni configurazione del robot, la forza artificiale generata dal potenziale viene generata come l’antigradiente 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 considerata è stata quella di considerare ogni ostacolo come una componente convessa tale da definire un corrispondente potenziale repulsivo come:

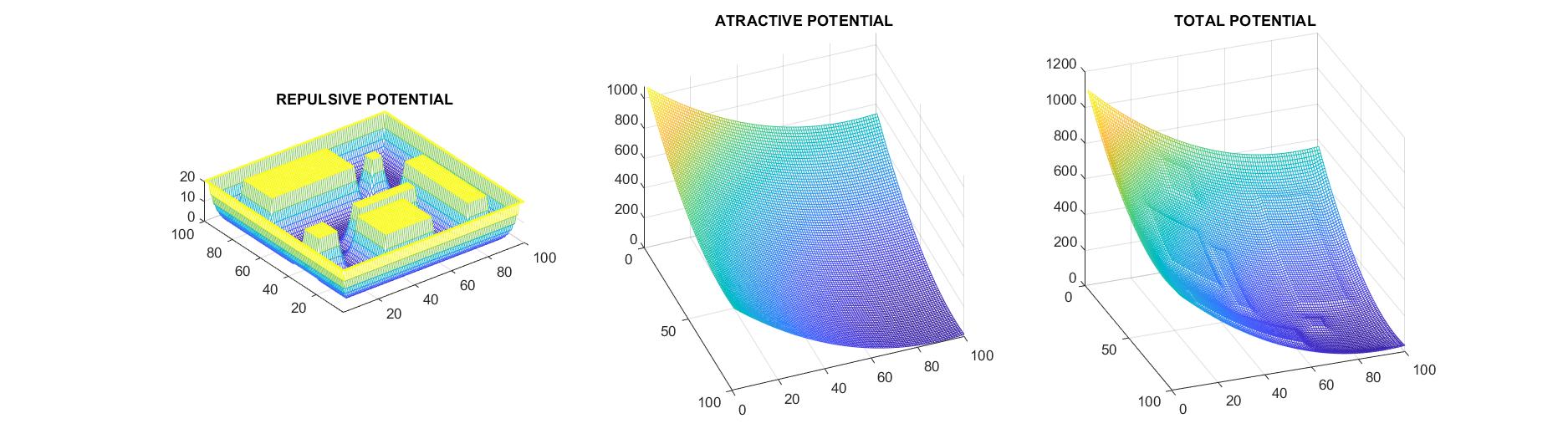
dove è la configurazione del robot, , , è la distanza dalla componente convessa ed rappresenta il raggio di influenza di . Il potenziale si annulla al di fuori del raggio di influenza mentre tende ad infinito quanto più grande è . Per quanto riguarda il calcolo di , esso è stato appena definito come . 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 è identificata nello script dalla variabile *Eta*. In questo caso è stato scelto un e un 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 riporto qui di seguito:

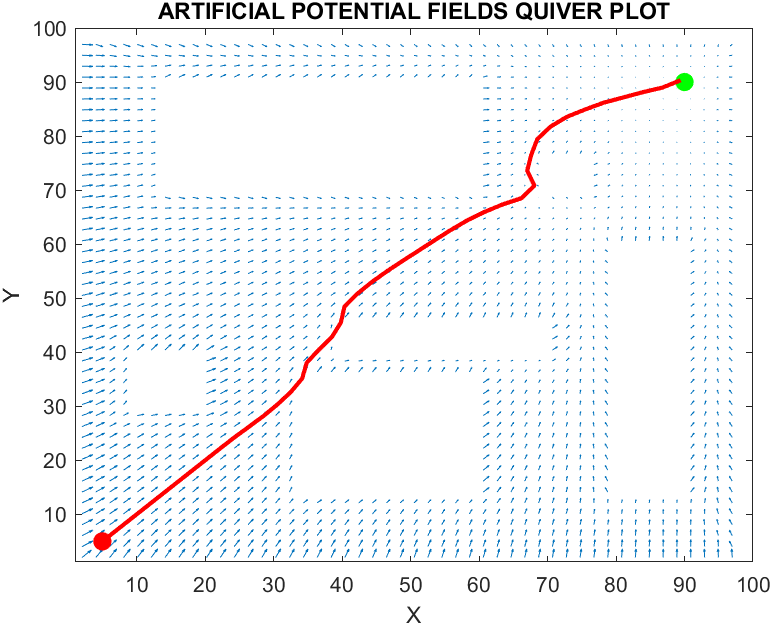
Infatti, come lo stesso Oussama Khatib afferma, rappresenta la minima distanza dall’ostacolo .

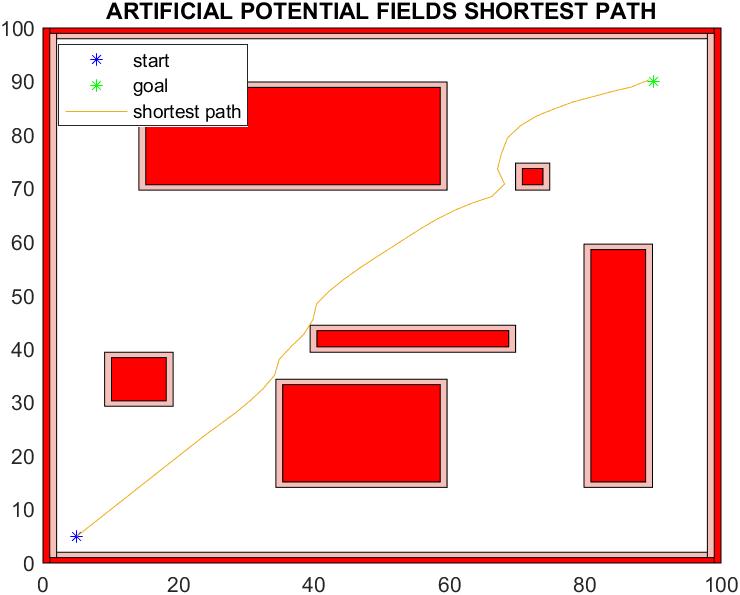
Il potenziale attrattivo, invece, è stato definito nella seguente maniera:

dove è la generica posizione, è la posizione target “g” (tratto sempre dalla pubblicazione di Oussama Khatib). In questo caso il coefficiente è stato considerato pari a 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.

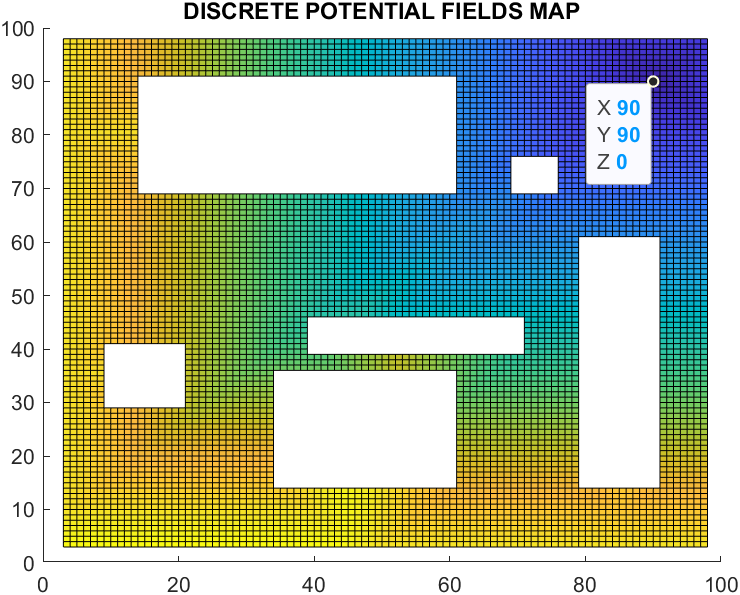
Il potenziale totale, pertanto, sarà pari alla soma del potenziale repulsivo e di quello attrattivo:

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

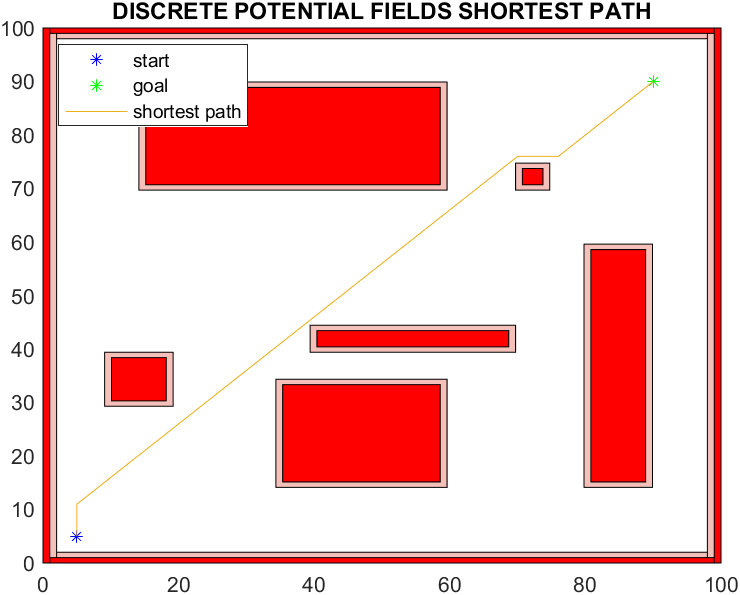
Si può notare come il potenziale repulsivo rappresenti alla perfezione l’ambiente considerato. Questo perché geometricamente più il robot si avvicina all’ostacolo generico più il potenziale repulsivo aumenta. Infatti, la colorazione giallo-blu sta ad indicare che nelle regioni “libere”, cioè dove il 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 il potenziale totale si può notare come esso 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, essi 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 , sarà lo stesso 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.

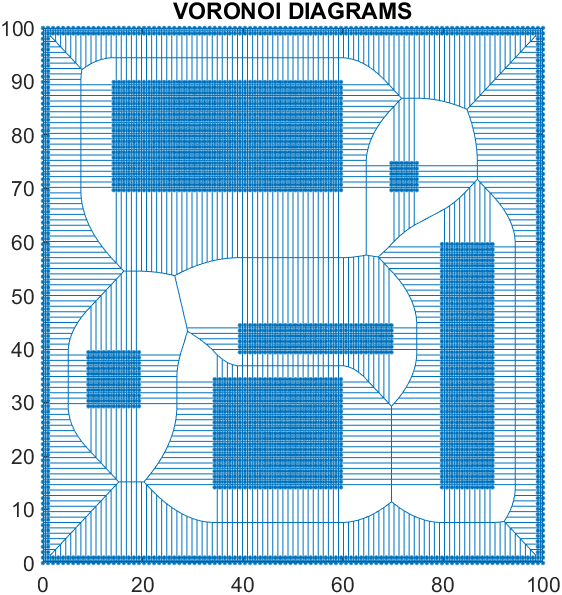
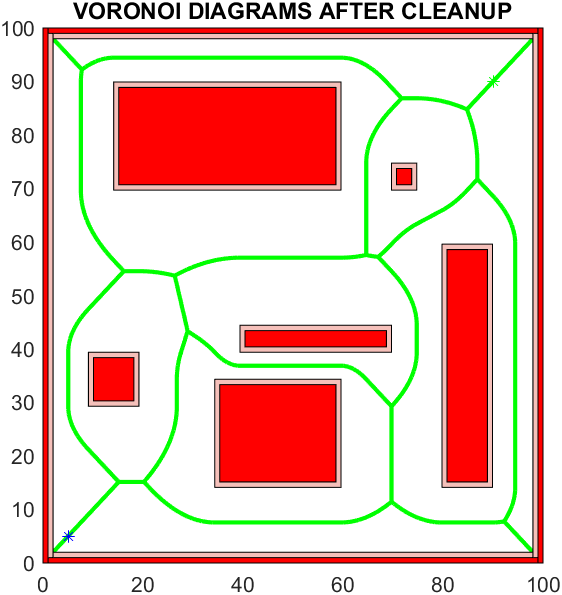
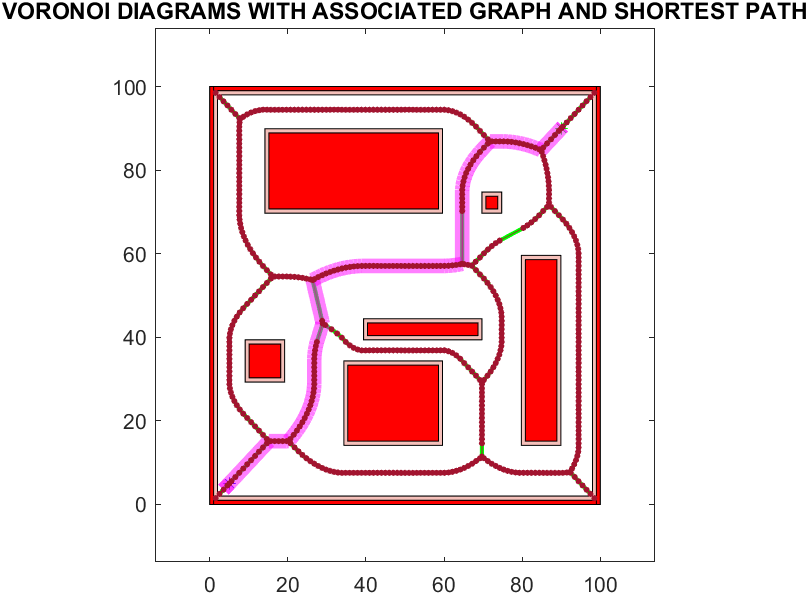
## Discrete Potential Fields

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 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.

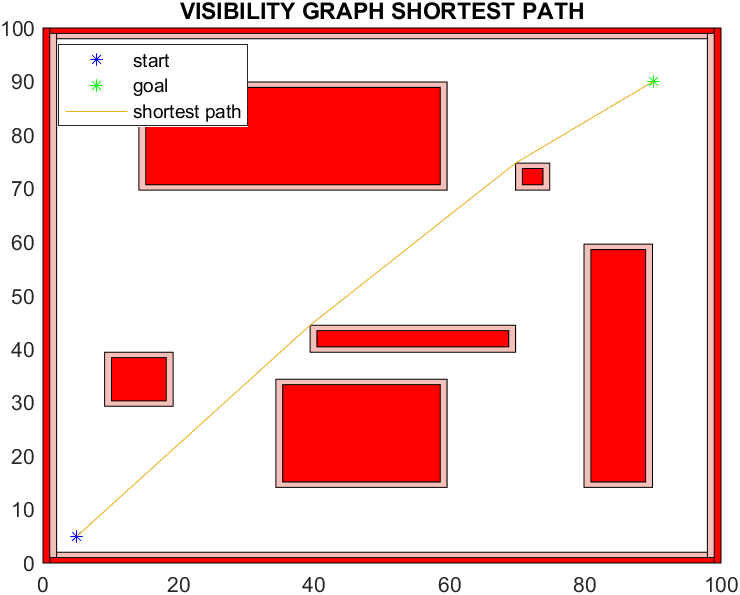
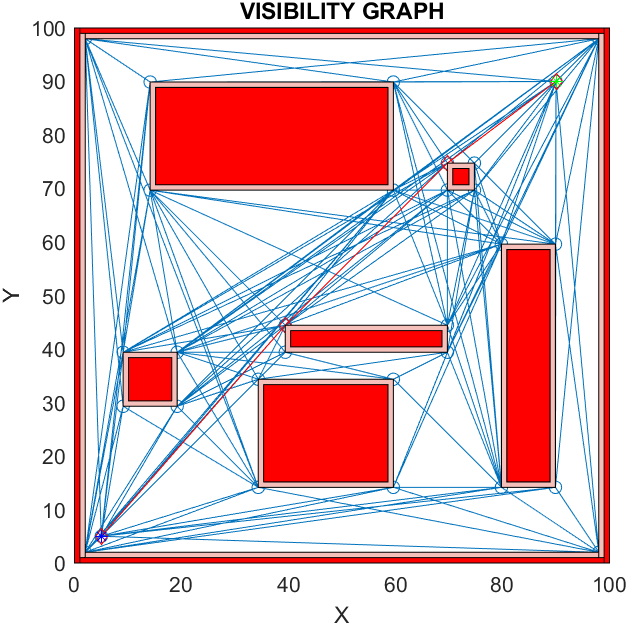
## Voronoi Diagrams

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 e 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”.

Si può notare come la tecnica in questione risulta avere una traiettoria molto “dolce” e precisa. Tanto è vero che il robot mobile riesce Immagine che contiene testo, segnale, parecchi

Descrizione generata automaticamenteperfettamente 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 insieme 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 sia ammissibile per il modello cinematico scelto tale che esista un robot di riferimento virtuale del tipo:

per una qualche scelta di input di riferimento e . 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 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 , 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 ottenuta 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 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 era 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 , , , e rispettivamente pari a , , , e .

### Non-Linear Control

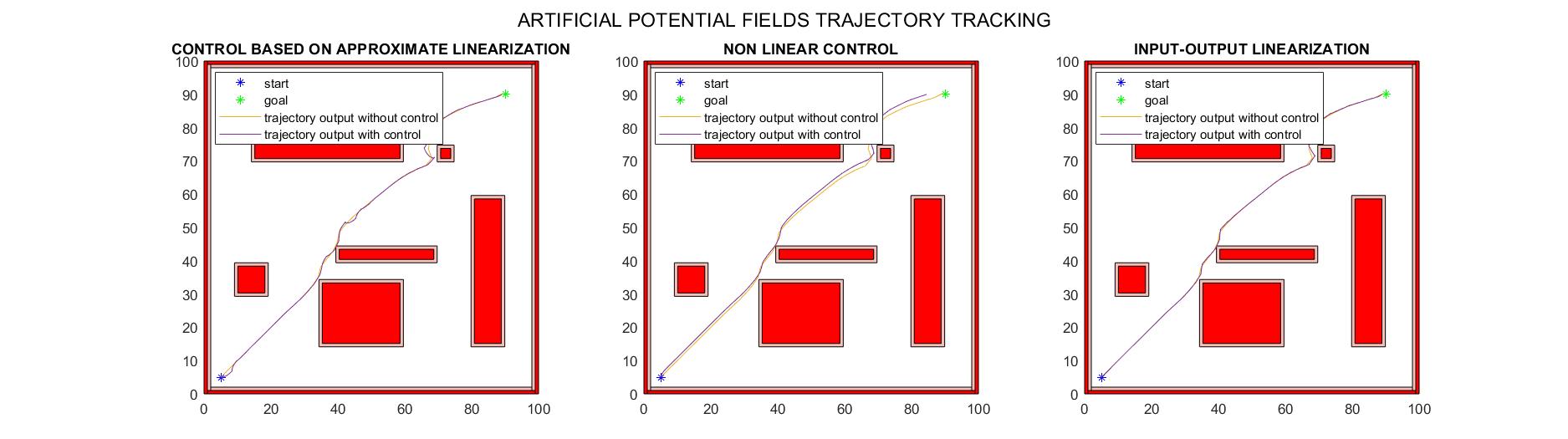
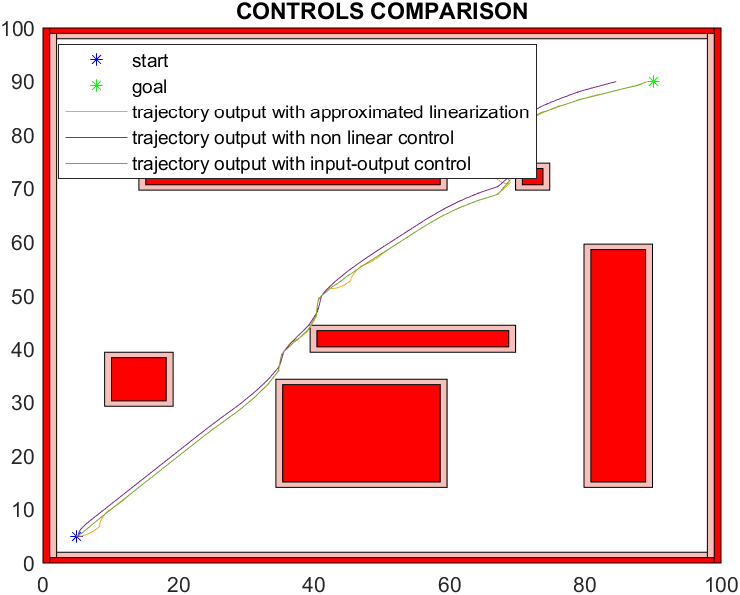
Il controllo non linearizzato è una tipologia di controllo che segue una legge di inserimento di traiettoria persistente, cioè le traiettorie devono essere persistenti (non deve esserci un momento in cui il robot mobile si ferma e riparte). In genere, a differenza della linearizzazione approssimata, 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 , e 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 , e aventi valori rispettivamente pari a , e . Bisogna far notare che il coefficiente 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 per essere 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 presenti maggiori oscillazioni rispetto alle altre, questo dovuto al fatto che comunque essa è una tecnica di cui non è assicurata la convergenza alla traiettoria di riferimento. Per quanto riguarda, invece, la traiettoria ottenuta tramite controllo non lineare, questa risulta essere quella che presenta maggiore errore rispetto alle altre sia il “tragitto” sia verso la posizione di target. Tanto è vero che il robot mobile tramite questo controllo si ferma in un intorno relativamente modesto della posizione target. Tale aspetto appena descritto non desta preoccupazione poiché la posizione di goal verrà successivamente raggiunta tramite la posture regulation. Infine, il controllo basato su linearizzazione input-output è quello che approssima al meglio la traiettoria di riferimento.

#### Discrete Potential Fields

#### Voronoi Diagrams

#### Visibility Graphs

## Posture Regulation

### Cartesian Regulation

### Complete Regulation

#### Artificial Potential Fields

#### Discrete Potential Fields

#### Voronoi Diagrams

#### Visibility Graphs