
RELAZIONE PROGETTO DATA DRIVEN CONTROL DESIGN

Antonio Langella

Matr. 0622702011

a.langella31@studenti.unisa.it

Michele Marsico

Matr. 0622702012

m.marsico10@studenti.unisa.it

Davide Risi

Matr. 0622702013

d.risi2@studenti.unisa.it



ROBOTARIUM

by GeorgiaTech

1 Definizione dei problemi di controllo

L'obiettivo di questo progetto è quello di controllare il moto di un robot in un ambiente con ostacoli. L'ambiente è rettangolare e ha dimensioni $3m \times 2m$. Il robot è controllato in velocità, dunque la dinamica discretizzata del robot da controllare è $\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{u}_k dt$ dove $\mathbf{x}_k = [p_{x,k}, p_{y,k}]^T$ è la posizione del robot all'istante k e $\mathbf{u}_k = [v_{x,k}, v_{y,k}]^T$ sono gli ingressi di controllo. Il comportamento desiderato è che il robot arrivi a un goal point $\mathbf{x}_d = [x_d, y_d]^T$ prefissato evitando gli ostacoli \mathbf{o}_i nella maniera più efficiente possibile. La misurazioni della posizione del robot sono soggette a rumore, che modelliamo come gaussiano, dunque la descrizione probabilistica del sistema da controllare è $p_{k|k-1}^{(x)} \sim \mathcal{N}(\mathbf{x}_{k-1} + \mathbf{u}_k dt, \Sigma)$, con una matrice di covarianza Σ scelta opportunamente. Gli algoritmi implementati sono stati validati tramite la piattaforma Robotarium. Per maggiori dettagli sul simulatore e sull'hardware messi a disposizione dal Robotarium si faccia riferimento all'Appendice A.

Affronteremo due problemi. In prima istanza si controlleranno i robot campionando le azioni di controllo \mathbf{u}_k da una sequenza di policy stocastiche $\{p_{k|k-1}^{(u)*}\}_{1:N}$ ottenuta impostando un problema di controllo ottimo su un orizzonte finito forward control). In questo caso si suppone che la funzione di costo $c(\mathbf{x}_k)$ sia nota, costruita sulla base delle posizioni note del goal point e degli ostacoli. La formulazione del problema di forward control che useremo è la seguente:

Problema 1 Trova la sequenza di policy $\{p_{k|k-1}^{(u)*}\}_{1:N}$ tale per cui:

$$\{p_{k|k-1}^{(u)*}\}_{1:N} \in \arg \min_{\{p_{k|k-1}^{(u)}\}_{1:N}} \left\{ -H(p_{0:N}) + \sum_{k=1}^N \mathbb{E}_{\bar{p}_{k-1:k}} \left[\mathbb{E}_{p_{k|k-1}^{(x)}} [c(\mathbf{X}_k)] \right] \right\} \quad \text{s.t. } p_{k|k-1}^{(u)} \in \mathcal{D} \quad \forall k \in \mathcal{T} \quad (1)$$

dove $\bar{p}_{k-1:k} := p_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_k)$, $H(p_{0:N})$ è l'entropia della funzione di probabilità $p_{0:N}$, \mathcal{D} è il sottoinsieme convesso delle distribuzioni di probabilità e $\mathcal{T} := 0 : N$ è l'orizzonte temporale.

Per risolvere il problema di forward control, si farà la seguente assunzione:

Assunzione 1 Esiste una sequenza di funzioni di probabilità $\{\tilde{p}_{k|k-1}^{(u)}\}_{1:N}$ che sia una soluzione ammissibile per il Problema 1 e tale che la funzione obiettivo del problema di ottimizzazione sia finita.

In seconda istanza si supporrà di non conoscere la funzione di costo e, partendo da un insieme di osservazioni ottenute dai robot controllati con l'algoritmo di cui prima, si intende stimare il costo col fine di controllare i robot con tale costo. Uno statement informale del problema di controllo inverso che si vuole risolvere è il seguente:

Problema 2 Data una sequenza di stati e ingressi di controllo osservati $\{(\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{u}}_k)\}_{1:M}$ con $\hat{\mathbf{x}}_k \sim p_{k|k-1}^{(x)}$, $\hat{\mathbf{u}}_k \sim p_{k|k-1}^{(u)*}$, dove $p_{k|k-1}^{(u)*}$ è la soluzione del Problema 1, si vogliono ottenere una stima del cost-to-go dell'agente $\bar{c}_k(\mathbf{x}_k)$ e del costo dell'agente $c(\mathbf{x}_k)$.

Per risolvere il problema di controllo inverso, si farà la seguente assunzione:

Assunzione 2 Il cost-to-go è espresso come una combinazione lineare di feature, ovvero $\bar{c}_k(\mathbf{x}_k) = -\mathbf{w}_k^T \mathbf{h}(\mathbf{x}_k)$ dove $\mathbf{h}(\mathbf{x}_k) := [h_1(\mathbf{x}_k), \dots, h_F(\mathbf{x}_k)]^T$ è il vettore delle feature e $h_i : X \rightarrow \mathbb{R}$ sono funzioni note con $i \in \{1, \dots, F\}$ e $\mathbf{w}_k := [w_{k,1}, \dots, w_{k,F}]^T$ è il vettore dei pesi.

2 Scelte progettuali e implementative

2.1 Soluzione dei problemi di controllo

La formulazione generale dei problemi di controllo è descritta nell'Appendice B. Nel paragrafo 1 abbiamo declinato i problemi nel caso specifico di una funzione di costo stazionaria e di una distribuzione di riferimento $q_{0:N}$ uniforme. La scelta di utilizzare una funzione di costo stazionaria è legittimata dal fatto che la posizione del goal-point e degli ostacoli non cambia nel tempo.

Una scelta implementativa cruciale è quella di risolvere a ogni istante k il Problema 1 per $N = 1$. In altre parole si utilizza una policy di controllo greedy. Ciò permette di evitare l'utilizzo della programmazione dinamica per calcolare il cost-to-go $\bar{c}_k(\mathbf{x}_k)$ che sarebbe computazionalmente molto oneroso e impedirebbe di utilizzare l'algoritmo di controllo in tempo reale. Pertanto, l'algoritmo di forward control implementato si basa sul seguente risultato:

Risultato 1 La soluzione al Problema 1 relativa all'istante k per $N=1$ è:

$$p_{k|k-1}^{(u)} = \frac{\exp \left(-\mathbb{E}_{p_{k|k-1}^{(x)}} \left[\ln(p_{k|k-1}^{(x)}) + c(\mathbf{X}_k) \right] \right)}{\sum_{u_k} \exp \left(-\mathbb{E}_{p_{k|k-1}^{(x)}} \left[\ln(p_{k|k-1}^{(x)}) + c(\mathbf{X}_k) \right] \right)} \quad (2)$$

Un altro vantaggio di utilizzare un costo stazionario e di risolvere il problema per $N = 1$ è la possibilità di semplificare il problema di controllo inverso riducendo drasticamente il numero di variabili decisionali del relativo problema di ottimizzazione grazie al seguente risultato:

Risultato 2 Data l'Assunzione 2, considerando che $p_{k|k-1}^{(u)} \overset{*}{\in}$ è la soluzione al Problema 1 per $N=1$ e data l'assunzione di costo stazionario, la stima a massima verosimiglianza per il costo è $c^*(\mathbf{x}_k) = -\mathbf{w}_s^{*T}\mathbf{h}(\mathbf{x}_k)$ dove \mathbf{w}_s^* è la soluzione del seguente problema di ottimizzazione convesso:

$$\mathbf{w}_s^* \in \arg \min_{\mathbf{w}_s} \left\{ \sum_{k=1}^M \left(-\mathbb{E}_{p(\mathbf{x}_k|\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{u}}_k)} [\mathbf{w}_s^T \mathbf{h}(\mathbf{x}_k)] + \ln \left(\sum_{\mathbf{u}_k} \exp \left(\mathbb{E}_{p(\mathbf{x}_k|\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k)} [-\ln(p(\mathbf{x}_k|\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k)) + \mathbf{w}_s^T \mathbf{h}(\mathbf{x}_k)] \right) \right) \right) \right\} \quad (3)$$

dove $\mathbf{w}_s \in \mathbb{R}^F$.

Si osservi infatti che nel caso generale il numero di variabili decisionali sarebbe stato $M \cdot F$.

Il problema di forward control è stato risolto implementando il Risultato 1 dapprima con la funzione di costo fornita, che è stata utilizzata come baseline, e poi con la funzione di costo da noi proposta. Al fine di validare l'algoritmo di controllo, sono state eseguite delle simulazioni con le API messe a disposizione da Robotarium.

Analogamente, il problema di controllo inverso è stato risolto implementando il Risultato 2 sia con le feature fornite che con quelle proposte. Anche in questo caso sono poi state eseguite delle simulazioni con policy ottenute a partire dal costo ricostruito. Per ulteriori dettagli implementativi si rimanda il lettore all'Appendice C. Nel seguito si analizzano le scelte relative alla funzione di costo e alle feature proposte.

2.2 Modifiche proposte per la funzione di costo

Sia \mathbf{o}_i la posizione dell'i-esimo ostacolo e n il numero di ostacoli. La funzione di costo fornita è la seguente:

$$c(\mathbf{x}_k) = 30|\mathbf{x}_k - \mathbf{x}_d|_2^2 + 20 \sum_{i=1}^n g_i(\mathbf{x}_k) + 10 \sum_{j=1}^4 h_j(\mathbf{x}_k) \quad (4)$$

dove $g_i(\mathbf{x}_k)$ è una bivariata gaussiana centrata in \mathbf{o}_i che modella la presenza dell'ostacolo i-esimo, h_j sono delle univariate gaussiane che modellano le pareti del Robotarium e il primo termine è una funzione quadratica che ha minimo sul goal point.

La modifica della funzione di costo si basa sull'idea chiave che il comportamento entropico del robot può essere regolato alterando il modulo del gradiente della funzione di costo: le zone della funzione costo a gradiente minore sono quelle in cui l'esplorazione casuale dell'ambiente è favorita e viceversa. Si forniscono evidenze sperimentali e una possibile intuizione algebrica di questa idea nell'Appendice D. Di seguito si elencano alcuni problemi riscontrati nelle simulazioni e le relative modifiche proposte per risolvere tali problemi:

Traiettorie oscillanti vicino al goal point: data la struttura di $c(\mathbf{x}_k)$, avremo che $|\nabla c(\mathbf{x}_k)| \rightarrow 0$ quando $\mathbf{x}_k \rightarrow \mathbf{x}_d$. Questo causa uno spiccato comportamento entropico nei pressi del goal point che disturba il corretto raggiungimento dell'obiettivo. Per risolvere questo problema si modifica il termine relativo al goal affinché il modulo del gradiente sia più elevato nei pressi del goal point.

I robot non sono in grado di aggirare alcuni ostacoli: si modificano i termini relativi agli ostacoli manipolandone il gradiente in modo da favorire l'esplorazione del robot ove opportuno.

I robot possono uscire fuori dai limiti del Robotarium: si aumenta la varianza delle gaussiane h_j per mantenere i robot più lontani dalle pareti e si aggiunge la funzione indicatore per evitare che il robot esca fuori dai limiti.

La funzione di costo che viene proposta per risolvere le problematiche di cui sopra è la seguente:

$$c(\mathbf{x}_k) = 30 \left[|\mathbf{x}_k - \mathbf{x}_d|_2^2 - \frac{1}{|\mathbf{x}_k - \mathbf{x}_d|_2} \right] + 30 \sum_i^n g_i^{(1)}(\mathbf{x}_k) + 15 \sum_i^n g_i^{(2)}(\mathbf{x}_k) + 10 \sum_{j=1}^4 h_j(\mathbf{x}_k) + \mathcal{X}_{\mathcal{B}}(\mathbf{x}_k) \quad (5)$$

dove $g_i^{(1)}(\mathbf{x}_k), g_i^{(2)}(\mathbf{x}_k)$ sono due bivariate gaussiane centrate in \mathbf{o}_i con varianze differenti, le h_j sono le stesse gaussiane della 4 con una varianza superiore rispetto al caso precedente e $\mathcal{X}_{\mathcal{B}}(\mathbf{x}_k)$ è la funzione indicatore dell'insieme $\mathcal{B} = [-1.5, 1.5] \times [-1, 1] \subset \mathbb{R}^2$, tale per cui $\mathcal{X}_{\mathcal{B}}(\mathbf{x}_k) = +\infty$ se $\mathbf{x}_k \notin \mathcal{B}$ e $\mathcal{X}_{\mathcal{B}}(\mathbf{x}_k) = 0$ se $\mathbf{x}_k \in \mathcal{B}$.

L'Appendice E approfondisce le scelte progettuali che hanno portato alle modifiche proposte per la funzione di costo. In tale appendice sono anche riportati i valori dei parametri della funzione costo.

2.3 Modifiche proposte per le feature

Le feature fornite per ricostruire la funzione di costo consistono di un insieme di 16 funzioni: una forma quadratica con minimo in \mathbf{x}_d per modellare il goal point e 15 multivariate gaussiane disposte su una griglia 5×3 che copre l'ambiente di navigazione dei robot per modellare la presenza di ostacoli. Di seguito si elencano alcuni problemi riscontrati nelle simulazioni col costo stimato e le relative modifiche proposte per risolvere tali problemi:

Traiettorie oscillanti vicino al goal point: è stata aggiunta una feature per attrarre maggiormente i robot verso il goal point. La funzione scelta è l'inverso della distanza euclidea rispetto al goal point perché era necessaria una funzione con modulo del gradiente elevato vicino al goal point.

Mancata ricostruzione di ostacoli: la griglia di feature per gli ostacoli non permette di ricostuire bene la funzione di costo quando ci sono degli ostacoli che non sono allineati con questa. Nella soluzione si propone di utilizzare una griglia di 5×7 punti. La varianza delle multivariate gaussiane è stata ridotta di conseguenza.

Mancata ricostruzione delle pareti: sono state aggiunte delle feature per la ricostruzione delle pareti del Robotarium. Le funzioni scelte sono univariate gaussiane.

L'Appendice F approfondisce le scelte progettuali che hanno portato alle modifiche proposte per le feature.

3 Risultati sperimentali

In questo paragrafo si intendono commentare alcuni risultati sperimentali notevoli relativi ai problemi di controllo forward e inverso. Per una raccolta più esaustiva di risultati sperimentali si faccia riferimento all'Appendice E per il problema forward e all'Appendice F per il problema inverso. Alcune delle policy ottenute col costo ricostruito sono state anche eseguite sull'hardware messo a disposizione dalla piattaforma Robotarium. I video di questi esperimenti sono disponibili nella repository raggiungibile dal link seguente: https://t.ly/B_dP_.

3.1 Problema di controllo forward

Nelle Figg. 1 sono rappresentate delle simulazioni fatte nello scenario fornito. Si osserva come la funzione di costo fornita causi delle forti oscillazioni dei robot vicino al goal point mentre la funzione di costo proposta risolve completamente questo problema. Nelle Figg. 2 le funzioni di costo sono confrontate in uno scenario in cui tra il robot e il goal point vi è una parete di tre ostacoli. Si osserva come, a parità di numero di iterazioni massimo, la funzione di costo fornita non riesce ad aggirare l'ostacolo a differenza della funzione proposta.

Nelle Figg. 3 è evidenziato un caso limite in cui i robot controllati con la funzione di costo fornita escono fuori dai limiti del Robotarium. Con la funzione di costo proposta questo problema è risolto.

3.2 Problema di controllo inverso

Nelle Figg. 4 sono rappresentate la heatmap della funzione di costo fornita valutata in uno scenario più complesso di quello fornito (Fig. 4a) e le relative heatmap delle funzioni di costo ricostruite tramite l'algoritmo di IOC attraverso le feature fornite (Fig. 4b) e le feature proposte (Fig. 4c). In questo caso si può osservare che alcuni ostacoli non vengono ricostruiti dall'insieme di feature originale ma vengono ricostituiti con l'insieme di feature proposto. Ciò risulta ancora più evidente dalle simulazioni coi costi ricostruiti rappresentate nelle Figg. 5.

Dalle simulazioni nelle Figg. 5 si può anche apprezzare un miglioramento relativo alle oscillazioni dei robot nei pressi del goal point, sebbene non si siano raggiunte le stesse prestazioni ottenute col forward control. Tale problema è probabilmente dovuto al fatto che già nelle osservazioni ottenute dalla funzione di costo fornita le traiettorie presentano molte oscillazioni in prossimità del goal point. Per corroborare quest'ipotesi si è condotto un esperimento in cui la funzione di costo proposta, che risolve il problema delle oscillazioni, è stata ricostruita con le feature proposte. Nella Fig. 6 è mostrata una simulazione realizzata con tale costo ricostruito: è possibile osservare come il fenomeno delle oscillazioni sia sensibilmente ridotto rispetto al caso precedente.

Per quanto riguarda la ricostruzione dei bordi del Robotarium, confrontando le heatmap nelle Figg. 4 si può concludere che questi siano completamente assenti nel costo ricostituito con le feature originali ma che con le feature proposte questi siano state ricostruite solo parzialmente. Tale problema è intrinsecamente complesso perché in alcune circostanze è complicato generare osservazioni in cui il robot reagisce alla presenza di un bordo.

4 Limitazioni e sviluppi futuri

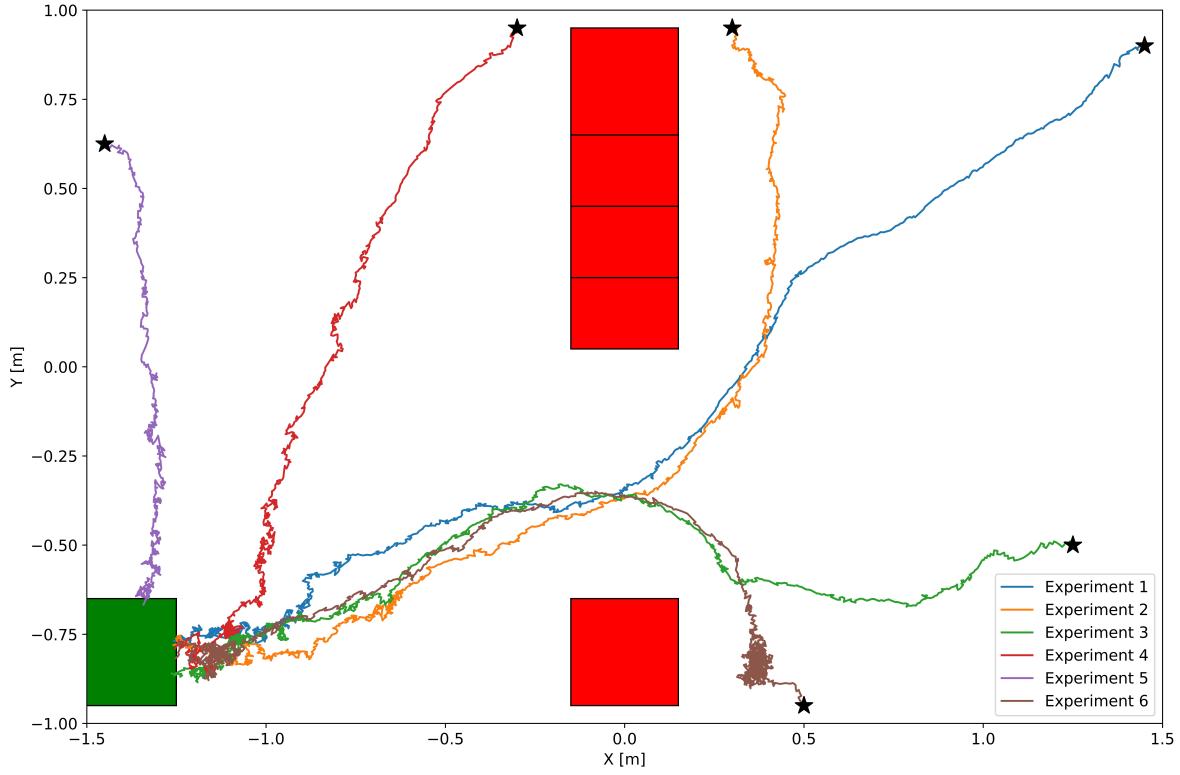
Tutte le problematiche relative alla funzione costo evidenziate nella sezione 2.2 sono dovute all'implementazione greedy del Problema 1. Questa scelta implementativa dettata da esigenze computazionali costituisce anche la principale limitazione dell'algoritmo di controllo implementato. Una possibile soluzione potrebbe essere l'utilizzo di una lookup policy per stimare il cost-to-go $\bar{c}_k(x_k)$ dell'agente. Un'ulteriore limitazione della soluzione implementata è che non permette di tenere direttamente conto dei limiti di attuazione.

Come evidenziato nella sezione precedente, l'insieme di feature proposte non è sempre in grado di ricostruire i confini dell'ambiente. Questa non è considerata una limitazione forte siccome in un'applicazione reale è lecito supporre di conoscere a priori i limiti dell'ambiente o di disporre di opportuni sensori per percepirla e quindi modificare opportunamente il costo ricostruito.

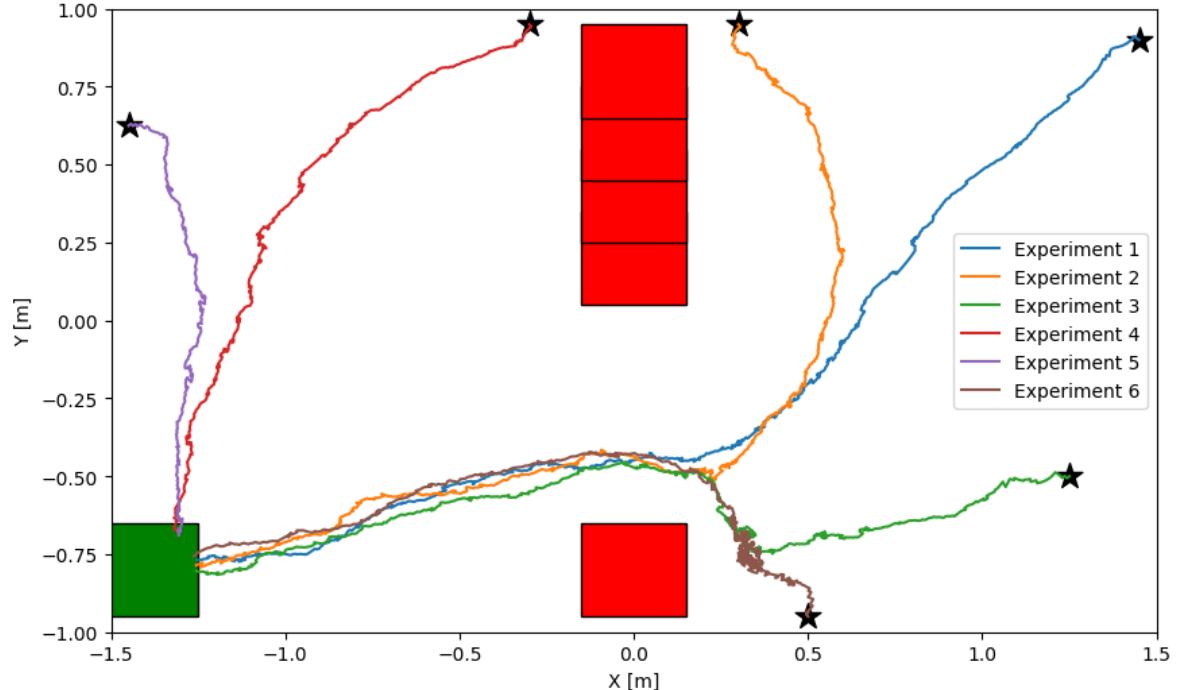
Un punto di forza del Problema 1 è la possibilità di estenderlo facilmente al caso di funzioni di costo tempo varianti. Pertanto, un interessante sviluppo futuro potrebbe essere quello di usare l'algoritmo per il controllo in tempo reale di robot in un ambiente non strutturato, utilizzando sensori di bordo per localizzare gli ostacoli e la posizione del robot nell'ambiente circostante. Si potrebbe utilizzare un approccio analogo anche per controllare più robot contemporaneamente, evitando che questi si scontrino.

Una soluzione più efficiente al Problema 1 potrebbe essere implementata tramite la tecnica nota come KL control [1]. In questo caso l'agente specifica direttamente gli stati $x' \sim u(x' | x)$, dove $u(x' | x)$ è la funzione di transizione dallo stato x allo stato x' , quindi si deve utilizzare un controllore di basso livello (ad esempio basato su MPC) per inseguire la traiettoria generata. Un'apparente limitazione di tale approccio è che la funzione di costo deve essere stazionaria, tuttavia siccome l'algoritmo è molto efficiente, nel caso di costo tempo variante si potrebbe pensare di eseguirlo nuovamente al variare del costo. Un'altra condizione necessaria all'applicazione di questa tecnica è che la funzione di costo sia non negativa. Ciò sarebbe facile da garantire sommando alla funzione di costo proposta una costante sufficientemente elevata.

Immagini

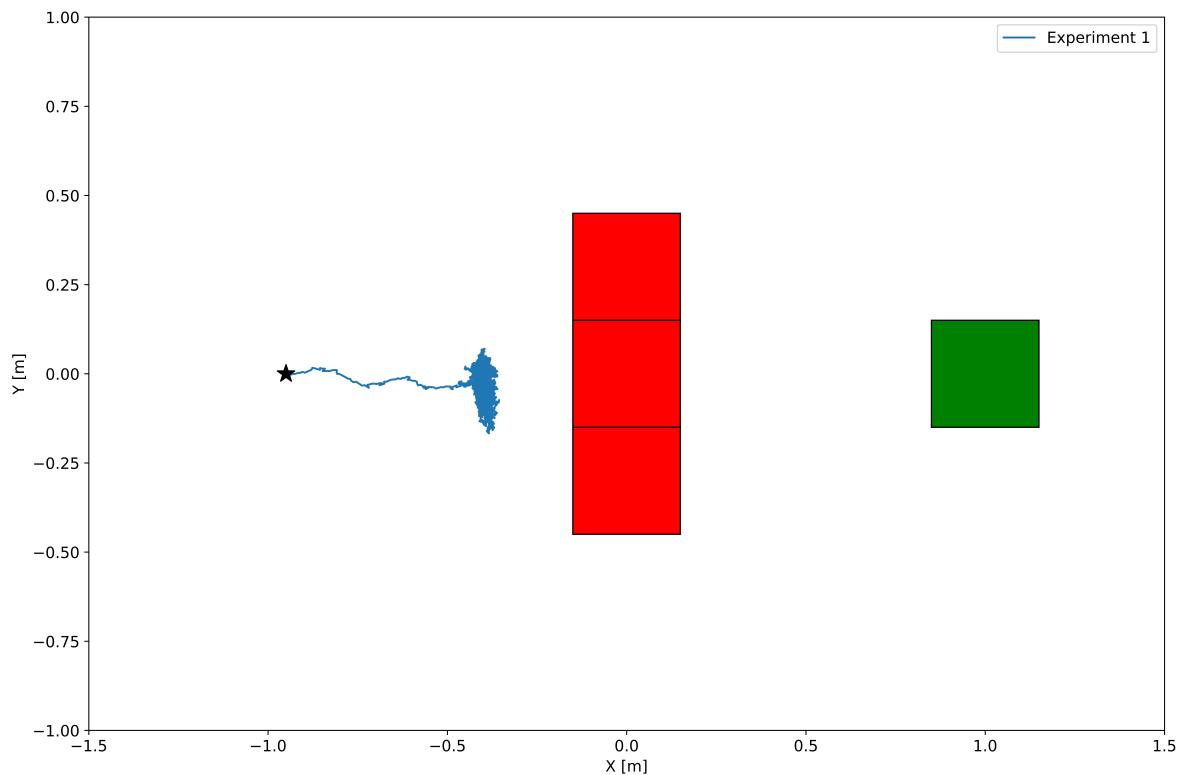


(a) Funzione di costo fornita

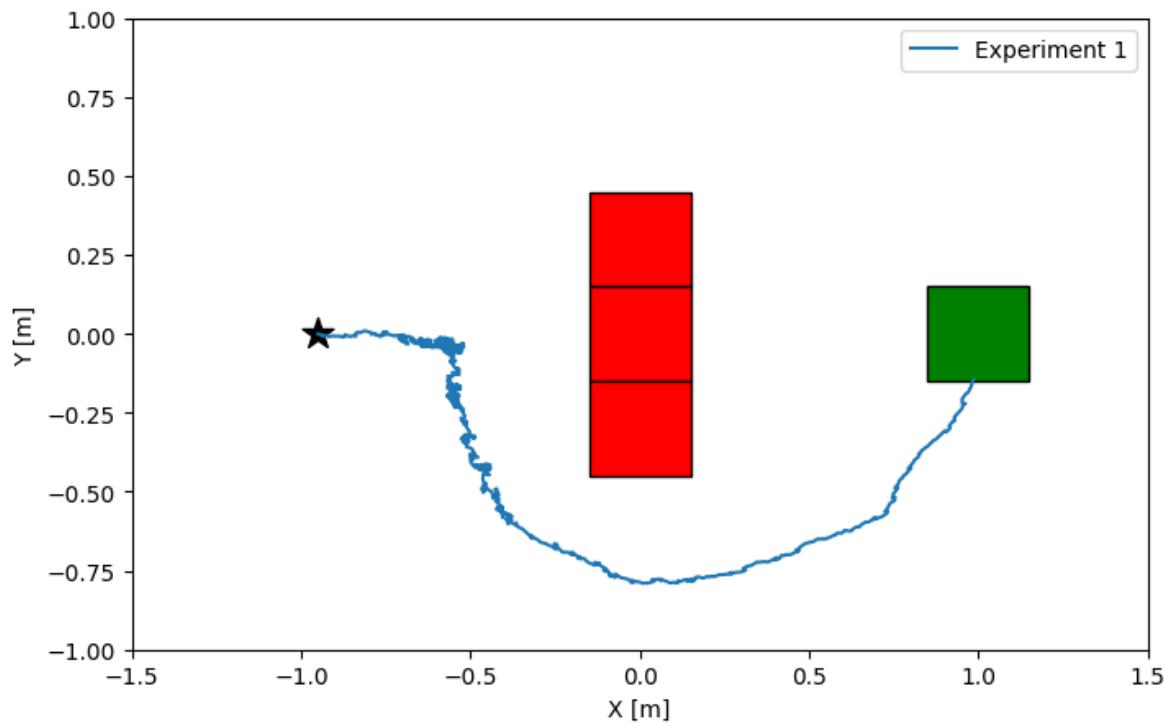


(b) Funzione di costo proposta

Figura 1: Simulazioni nello scenario di base

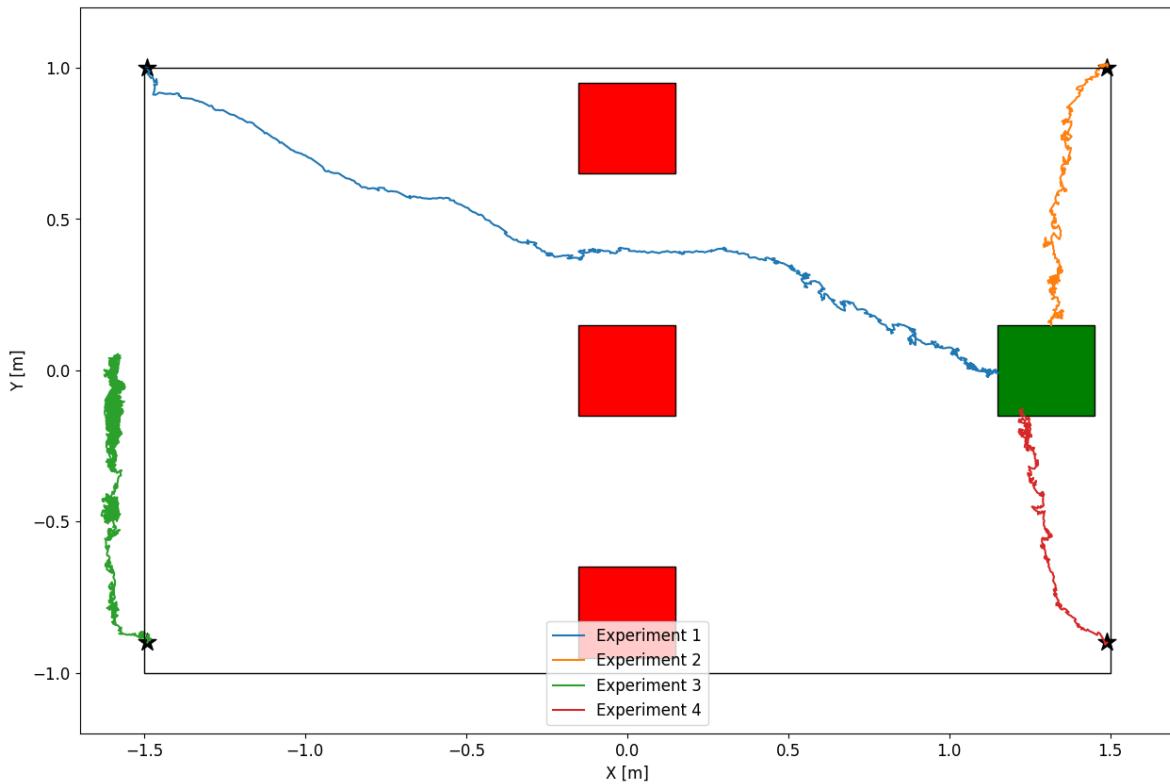


(a) Funzione di costo fornita

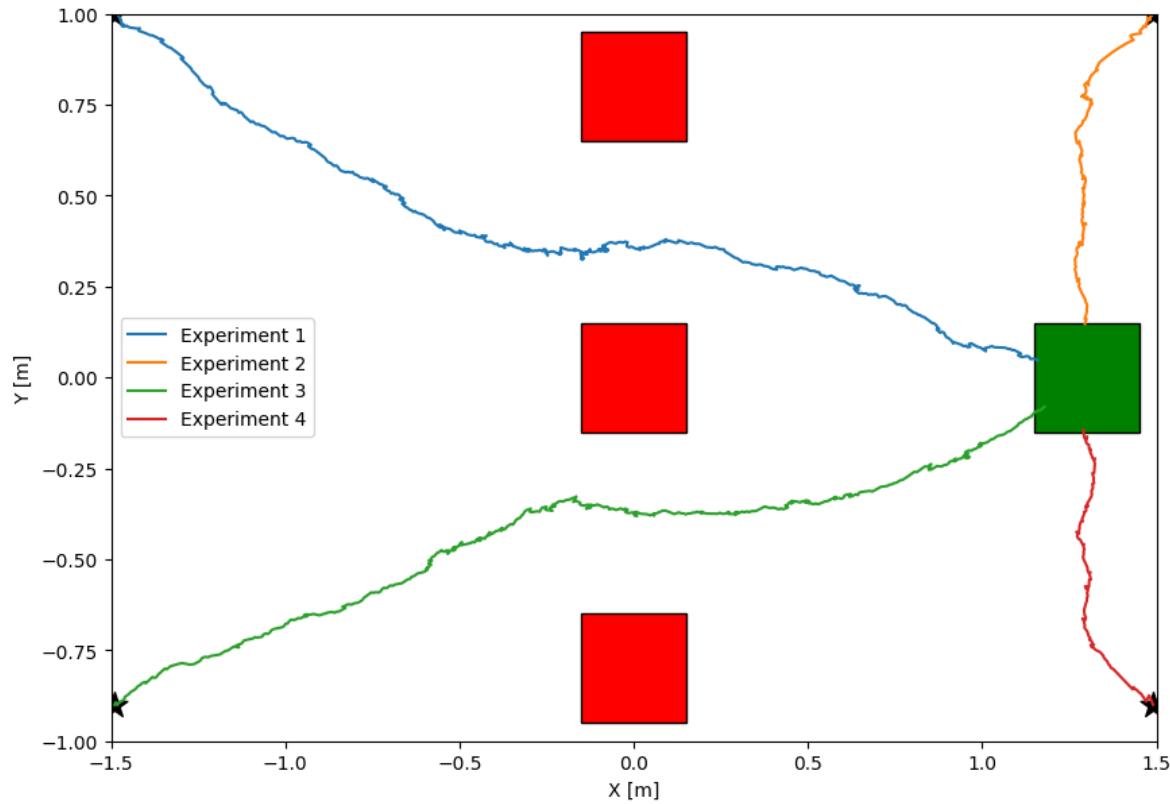


(b) Funzione di costo proposta

Figura 2: Simulazioni con una parete di tre ostacoli

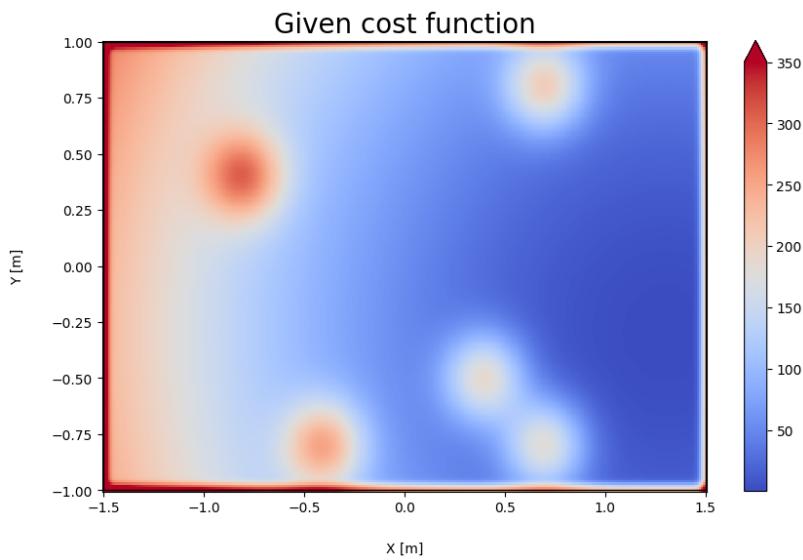


(a) Funzione di costo fornita

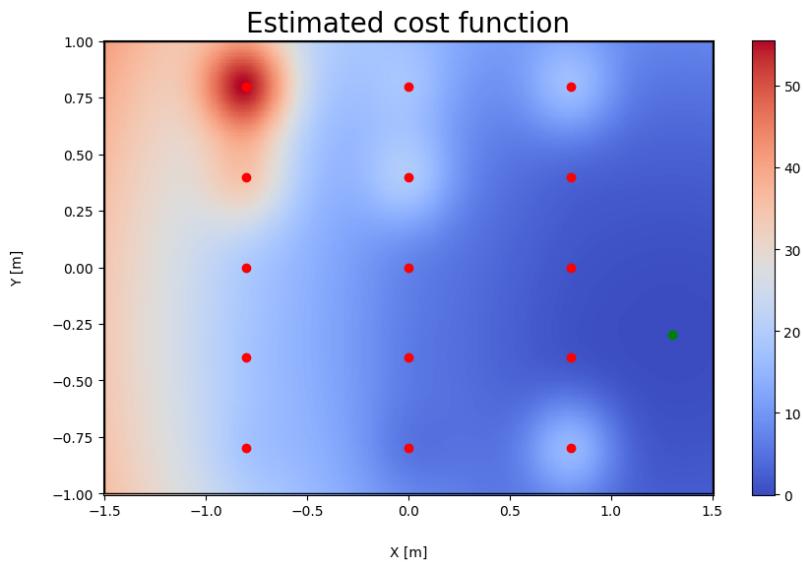


(b) Funzione di costo proposta

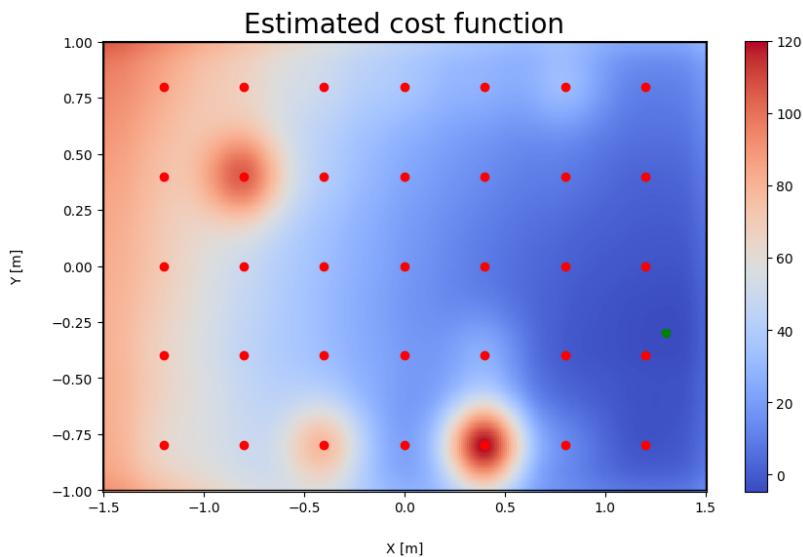
Figura 3: Simulazioni con robot vicini alle pareti



(a) Funzione di costo originale

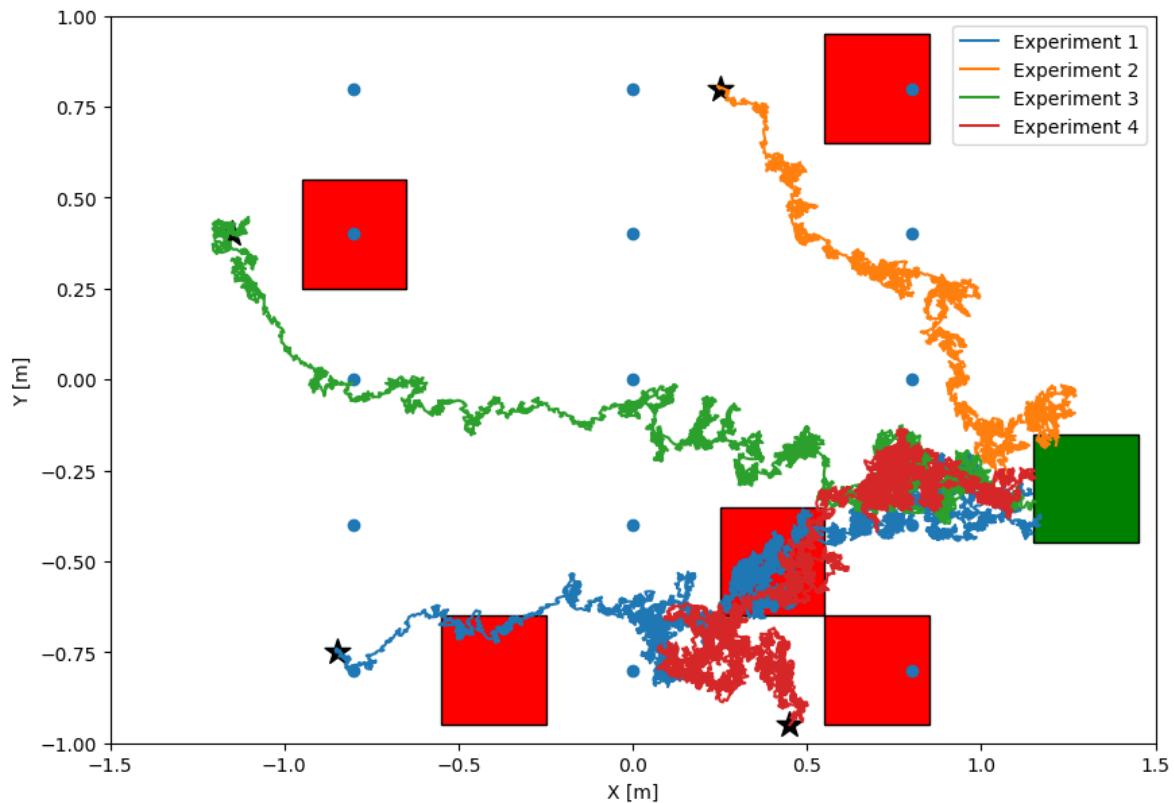


(b) Funzione di costo ricostruita con le feature fornite

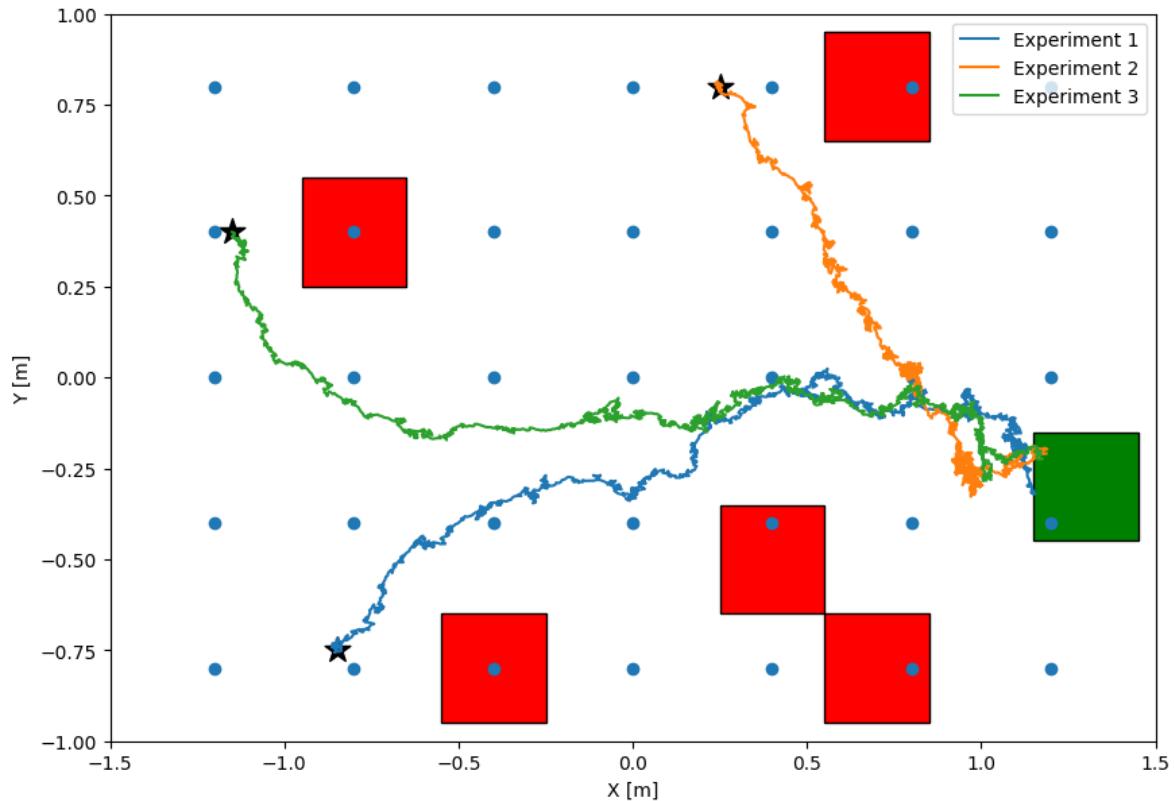


(c) Funzione di costo ricostruita con le feature proposte

Figura 4: Funzione di costo originale e ricostruite su uno scenario complesse



(a) Simulazione con costo ricostruito dalle feature fornite



(b) Simulazione con costo ricostruito dalle feature proposte

Figura 5: Simulazioni con funzioni di costo ricostruite

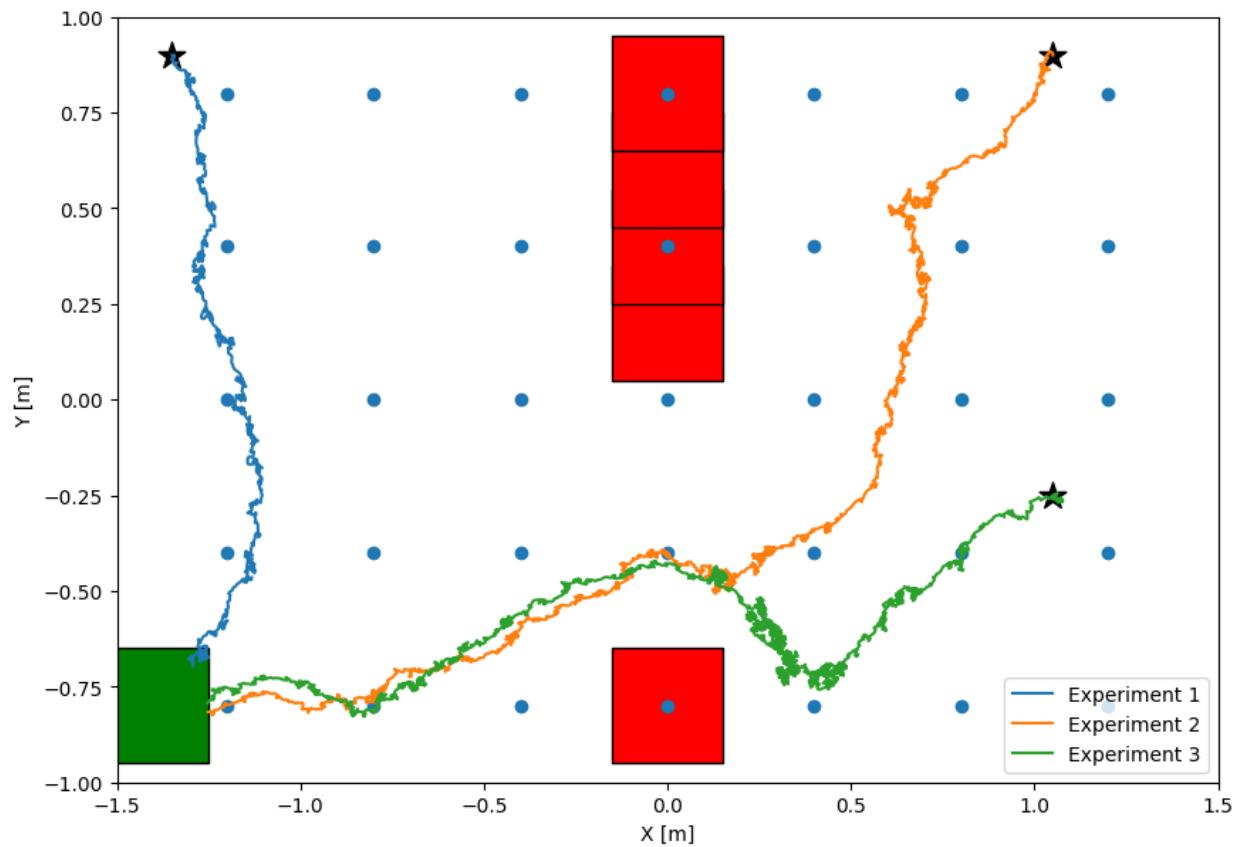


Figura 6: Simulazione con funzione di costo proposta ricostruita con feature proposte

Appendici

Appendice A Robotarium

Il simulatore del Robotarium consente di testare rapidamente gli algoritmi di controllo progettati e di ricevere un feedback sulla fattibilità della loro implementazione prima di effettuare il deployment vero e proprio. I robot della piattaforma Robotarium sono modellati dal modello Unicycle:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (6)$$

dove $\mathbf{x} = [x \ y \ \theta]^T$ è la posizione e l'orientamento del robot e $v, \omega \in \mathbb{R}$ rappresentano rispettivamente la velocità lineare e angolare del robot. Tale modello è quello utilizzato internamente da Robotarium per simolare la dinamica del robot. In particolare, gli stati x_1 e x_2 sono le coordinate del centro del robot rispetto ad un sistema di riferimento definito dal Robotarium, e θ è la rotazione dell'asse delle ruote rispetto all'asse X del sistema di riferimento del Robotarium. Una rappresentazione di questo modello è quella mostrata in Fig. 7a.

I robot in questione sono di tipo "differential drive" e dunque presentano un vincolo cinematico anolonomo dovuto al fatto che le ruote non possono slittare lateralmente. A causa di ciò, non è possibile imporre arbitrariamente delle velocità al baricentro del robot (x_1, x_2) e quindi questo non può essere modellato come un integratore. Se si volesse imporre una velocità con componente non nulla lungo l'asse delle ruote si dovrebbe dapprima applicare una velocità angolare $\tilde{\omega}$ e poi applicare una velocità ortogonale all'asse delle ruote \tilde{v} . Per semplificare il sistema di controllo, si introduce un nuovo sistema, che modella la cinematica di un punto differente dal baricentro, detto **hand point**. La dinamica di questo punto può essere infatti descritta tramite un integratore singolo poiché è possibile imporvi velocità arbitrarie. L'hand point è posizionato ad una distanza $l \neq 0$, perpendicolare all'asse che congiunge le due ruote. Dunque, si ha che la posizione $\mathbf{s}(\mathbf{x})$ dell'hand point partendo dalla posizione del centro del robot $\mathbf{x}_p = [x_1 \ y_2]^T$ è:

$$\mathbf{s}(\mathbf{x}) = \mathbf{x}_p + l \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \quad (7)$$

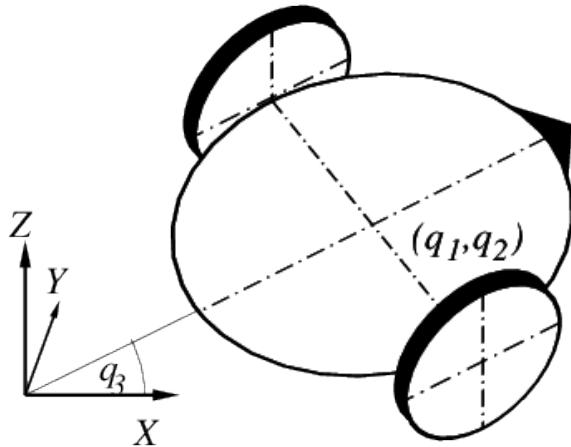
La dinamica dell'hand point è la seguente:

$$\dot{\mathbf{s}}(\mathbf{x}) = \begin{bmatrix} \dot{x}_1 - l \sin(\theta) \dot{\theta} \\ \dot{x}_2 + l \cos(\theta) \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta)v - l \sin(\theta)\omega \\ \sin(\theta)v + l \cos(\theta)\omega \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -l \sin(\theta) \\ \sin(\theta) & +l \cos(\theta) \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \mathbf{R}(\theta) \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (8)$$

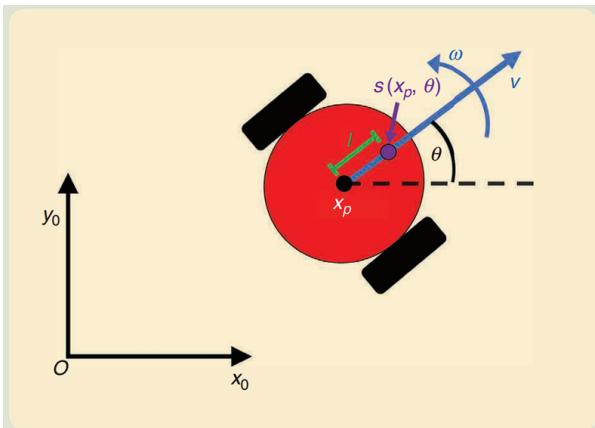
A questo punto possiamo controllare l'hand point in velocità e poi ricavare gli ingressi di controllo equivalenti da fornire al sistema unicycle:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \mathbf{R}^{-1}(\theta) \cdot \dot{\mathbf{s}}_d(\mathbf{x})$$

dove $\dot{\mathbf{s}}_d(\mathbf{x})$ è il vettore velocità che si vuole imporre all'hand point, ovvero l'ingresso di controllo \mathbf{u}_k con cui l'agente progettato si interfaccia col robot.



(a) Modello Unicycle. q_1, q_2 e q_3 rappresentano rispettivamente x_1, x_2 e θ . Fonte [2].



(b) Modello dell'hand point. Fonte [3]

Figura 7: Modelli del robot

Appendice B Formulazioni generale dei problemi di controllo e soluzioni

B.1 Forward Optimal Control Problem

Nel caso più generale, il problema del Forward Optimal Control è formulato come segue

Problema 3 *Data la funzione di probabilità congiunta $q_{0:N} := q_0(\mathbf{x}_0) \prod_{k=1}^N q_k^{(x)}(\mathbf{x}_k \mid \mathbf{u}_k, \mathbf{x}_{k-1}) q_k^{(u)}(\mathbf{u}_k \mid \mathbf{x}_{k-1})$ si trovi la sequenza di funzioni di probabilità $\{p_{k|k-1}^{(u)}\}_{1:N}^*$ tale che:*

$$\begin{aligned} \left\{p_{k|k-1}^{(u)}\right\}_{1:N}^* &\in \arg \min_{\{p_{k|k-1}^{(u)}\}_{1:N}} \left\{ \mathcal{D}_{KL}(p_{0:N} \parallel q_{0:N}) + \sum_{k=1}^N \mathbb{E}_{\bar{p}_{k-1:k}} \left[\mathbb{E}_{p_{k|k-1}^{(x)}} [c_k(\mathbf{X}_k)] \right] \right\} \\ &\text{s.t. } p_{k|k-1}^{(u)} \in \mathcal{D} \quad \forall k \in \mathcal{T} \end{aligned} \quad (9)$$

dove $\bar{p}_{k-1:k} := p_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_k)$.

Nella formulazione generale del Forward Optimal Control il primo termine della funzione di costo agisce come un regolatore che fa propendere il comportamento del sistema alla distribuzione di riferimento $q_{0:N}$. Siccome la funzione di costo permette di specificare il task di controllo, la distribuzione di riferimento $q_{0:N}$ può essere usata per diversi scopi, ad esempio:

- per catturare la differenza tra l'impianto su cui è stata calcolata e sull'impianto che si sta cercando di controllare;
- per inseguire delle traiettorie di controllo nominali;
- per modellare la dinamica a ciclo aperto, in modo da minimizzare lo sforzo di controllo;
- per favorire l'esplorazione, utilizzando una $q_{0:N}$ uniforme. In questo caso si dimostra che il D_{KL} diventa un regolarizzatore entropico in quanto minimizzare la D_{KL} significa massimizzare $H(p_{0:N})$.

Contestualizzando il problema generale nel caso specifico che si deve affrontare:

- La probabilità di riferimento $q_{0:N}$ è uniforme, quindi la D_{KL} diventa un regolarizzatore entropico;
- Il costo è stazionario, quindi $c_k(\mathbf{X}_k) = c(\mathbf{X}_k)$.

Dunque, la formulazione del Problema 3 diventa quella descritta nel Problema 1.

B.2 Soluzioni al Forward Optimal Control Problem

La soluzione del Problema 3 si basa sul fatto che il problema di ottimizzazione nella 9 può essere ricorsivamente scomposto nei seguenti sottoproblemi:

$$\min_{\{p_{k|k-1}^{(u)}\}_{1:N-1}} \mathcal{D}_{KL}(p_{0:N-1} \parallel q_{0:N-1}) + \sum_{k=1}^{N-1} \mathbb{E}_{\bar{p}_{k-1:k}} \left[\mathbb{E}_{p_{k|k-1}^{(x)}} [c_k(\mathbf{X}_k)] \right] \quad (10a)$$

$$\begin{aligned} &\text{s.t. } p_{k|k-1}^{(u)} \in \mathcal{D} \quad \forall k \in 1 : N-1 \\ &\min_{p_{N|N-1}^{(u)}} \mathbb{E}_{\bar{p}_{N-1}} \left[\mathcal{D}_{KL}(p_{N|N-1} \parallel q_{N|N-1}) + \mathbb{E}_{p_{N|N-1}^{(x)}} [c_N(\mathbf{X}_N)] \right] \\ &\text{s.t. } p_{N|N-1}^{(u)} \in \mathcal{D} \end{aligned} \quad (10b)$$

Da questa osservazione è possibile ricavare il seguente risultato:

Risultato 3 *Considerando il Problema 3 e supponendo che valga l'Assunzione 1, allora vale il seguente risultato:*

1. *Il Problema 3 ha un'unica soluzione $p_{k|k-1}^{(u)}_{1:N}^*$, con*

$$p_{k|k-1}^{(u)}_* = \frac{\bar{p}_{k|k-1}^{(u)} \exp \left(-\mathbb{E}_{p_{k|k-1}^{(x)}} [\bar{c}_k(\mathbf{X}_k)] \right)}{\sum_{\mathbf{u}_k} \bar{p}_{k|k-1}^{(u)} \exp \left(-\mathbb{E}_{p_{k|k-1}^{(x)}} [\bar{c}_k(\mathbf{X}_k)] \right)}$$

dove $\bar{p}_{k|k-1}^{(u)} := q_{k|k-1}^{(u)} \exp \left(-\mathcal{D}_{KL} \left(p_{k|k-1}^{(x)} \parallel q_{k|k-1}^{(x)} \right) \right)$ e dove $\bar{c}_k : \mathcal{X} \mapsto \mathbb{R}$ è ottenuto dalla ricorsione definita come segue:

$$\bar{c}_k(\mathbf{x}_k) = c_k(\mathbf{x}_k) - \hat{c}_k(\mathbf{x}_k),$$

$$\hat{c}_k(\mathbf{x}_k) = \ln \left(\mathbb{E}_{q_{k+1|k}^{(u)}} \left[\exp \left(-\mathcal{D}_{KL} \left(p_{k+1|k}^{(x)} \parallel q_{k+1|k}^{(x)} \right) - \mathbb{E}_{p_{k+1|k}^{(x)}} [\bar{c}_{k+1}(\mathbf{X}_{k+1})] \right) \right] \right),$$

$$\mathcal{D}_{KL} \left(p_{N+1|N}^{(x)} \parallel q_{N+1|N}^{(x)} \right) + \mathbb{E}_{p_{N+1|N}^{(x)}} [\bar{c}_{N+1}(\mathbf{X}_{N+1})] = 0$$

2. Il minimo del Problema 3 è dato da $-\sum_{k=1}^N \mathbb{E}_{\bar{p}_{k-1}}[\hat{c}_{k-1}(\mathbf{X}_{k-1})]$, $\bar{p}_{k-1} := p_{k-1}(\mathbf{x}_{k-1})$.

Se adesso si considera il caso particolare in cui la funzione di probabilità di riferimento $q_{0:N}$ è uniforme vale il seguente risultato:

Risultato 4 Considerando il Problema 3, supponendo che valga l'Assunzione 1 e che con $q_{0:N}$ sia uniformemente distribuita allora:

$$p_{k|k-1}^{(u)} * = \frac{\exp\left(-\mathbb{E}_{p_{k|k-1}^{(x)}}\left[\ln(p_{k|k-1}^{(x)}) + \bar{c}(\mathbf{X}_k)\right]\right)}{\sum_{\mathbf{u}_k} \exp\left(-\mathbb{E}_{p_{k|k-1}^{(x)}}\left[\ln(p_{k|k-1}^{(x)}) + \bar{c}_k(\mathbf{X}_k)\right]\right)} \quad (11)$$

dove $\bar{c}_k(\mathbf{x}_k) = c_k(\mathbf{x}_k) - \hat{c}_k(\mathbf{x}_k)$ e $\hat{c}_k(\mathbf{x}_k)$ si calcola con la seguente formula ricorsiva:

$$\begin{aligned} \hat{c}_k(\mathbf{x}_k) &= \ln\left(\sum_{\mathbf{u}_k} \exp\left(-\mathbb{E}_{p_{k+1|k}^{(x)}}\left[\ln(p_{k+1|k}^{(x)}) + \bar{c}_{k+1}(\mathbf{X}_{k+1})\right]\right)\right), \\ \mathbb{E}_{p_{N+1|N}^{(x)}}\left[\ln(p_{N+1|N}^{(x)}) + \bar{c}_{N+1}(\mathbf{X}_{N+1})\right] &= 0 \end{aligned}$$

Si osservi che la presenza del regolarizzatore $D_{KL}\left(p_{k|k-1}^{(u)} || q_{k|k-1}^{(u)}\right)$ nella funzione obiettivo, che compare sviluppando il termine $D_{KL}(p_{k|k-1} || q_{k|k-1})$ nei sottoproblemi, fa sì che le policy $p_{k|k-1}^{(u)} *$ abbiano la forma di una Softmax. Dal Risultato 4 consegue direttamente il Risultato 1 ponendo $\hat{c}_k(\mathbf{x}_k) = 0$.

B.3 Confronto tra il Forward Optimal Control e altre tecniche

La tecnica scelta per risolvere il problema di forward control, formulata nel Problema 1, ha molti vantaggi: si tratta di una forma di controllo completamente probabilistico, che ambisce a risolvere il decision-making problem direttamente nello spazio delle policy. Il sistema da controllare può essere non lineare, stocastico o tempo variante. Grazie alla formulazione max entropy è promossa l'esplorazione, ma il regolarizzatore $D_{KL}(\cdot)$ può essere facilmente utilizzato per indurre l'agente a seguire un comportamento desiderato estratto da un dataset di esempi oppure per penalizzare lo sforzo di controllo (utilizzando la dinamica passiva del sistema come distribuzione di riferimento). Un altro vantaggio della tecnica scelta è legato al fatto che l'Algoritmo 1 è facilmente estendibile al caso di un costo $c_k(\mathbf{x}_{k-1})$ tempo variante. A differenza di altri metodi, per il FOC esiste una soluzione esplicita sebbene questa richieda l'utilizzo di Dynamic Programming e sia quindi onerosa dal punto di vista computazionale al crescere del numero di stati. Per questa ragione in tale progetto ne è stata implementata una versione greedy, che però fornisce una soluzione sub-ottimale rispetto al caso più generale in cui $N > 1$.

La formulazione generale del problema di Forward Optimal Control (Problema 3) è una generalizzazione del **controllo da dati di esempio**, nel quale ci si limita a rendere il comportamento del sistema a ciclo chiuso $f_{0:N}$ il più simile possibile a una funzione di probabilità di riferimento $g_{0:N}$. La funzione $g_{0:N}$ è la descrizione probabilistica del comportamento desiderato e può essere estratta da un dataset di esempi tramite histogram filter. In questo caso è necessario che la funzione $f_{0:N}$ sia assolutamente continua rispetto a $g_{0:N}$, il che può essere interpretato qualitativamente col fatto che il dataset di esempi deve essere sufficientemente esaustivo. Nella formulazione del Problema 1 questi problemi sono superati introducendo nella formulazione una funzione di costo che permette di esprimere il task di controllo. In questo modo l'agente può andare oltre alla semplice imitazione e la D_{KL} assume un ruolo di regolarizzatore nel problema di ottimizzazione.

Una tecnica che permetterebbe di trovare una soluzione ottima al Problema 3 con $N > 1$, ovvero senza ricorrere ad un implementazione greedy, sarebbe la tecnica del **KL control** [1]. Tale tecnica richiede l'assunzione che $p^{(x)}(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k) = p^{(x)}(\mathbf{x}_k | \mathbf{x}_{k-1})$ ovvero che l'agente possa specificare direttamente le transizioni di stato. Questo significa che sarebbe necessario un controllore di livello inferiore per seguire la traiettoria generata dall'agente, ad esempio realizzato tramite MPC. Un'altra assunzione facile da soddisfare è che il costo sia non negativo. Un'apparente limitazione di tale approccio è che la funzione di costo deve essere stazionaria, tuttavia siccome l'algoritmo è molto efficiente, nel caso di costo tempo variante si potrebbe pensare di eseguirlo nuovamente al cambiare del costo.

Gli algoritmi di Reinforcement Learning **Q-Learning** e **Reinforce** possono essere utilizzati senza conoscenza del modello $p_{k|k-1}^{(x)}$ e senza conoscere la funzione di costo $c(\mathbf{x}_{k-1})$. Ciononostante, per l'applicazione di interesse si dovrebbe fornire comunque all'agente un segnale $\bar{c}_k = c(\mathbf{x}_k)$ determinato ad ogni k a partire da una funzione di costo opportunamente progettata. Uno svantaggio di questi algoritmi è che limitano la soluzione a una policy stazionaria. Inoltre, necessitano entrambi di molti dati per arrivare a convergenza. Un ulteriore svantaggio dell'algoritmo Reinforce è che questo richiede di assumere la famiglia di funzioni nella quale ricercare la policy, per cui è un metodo meno generale di quello impiegato. In definitiva in questo contesto in cui è facile modellare la dinamica del robot (single integrator) e la funzione di costo è nota, non è conveniente utilizzare algoritmi di Reinforcement Learning.

B.4 Inverse Optimal Control Problem e soluzioni

La definizione più generale del problema di controllo inverso (IOC) è la seguente:

Problema 4 Data una sequenza di stati e ingressi di controllo osservati $\{(\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{u}}_k)\}_{1:M}$ con $\hat{\mathbf{x}}_k \sim p_{k|k-1}^{(x)}$, $\hat{\mathbf{u}}_k \sim p_{k|k-1}^{(u)} *$, dove $p_{k|k-1}^{(u)} *$ è la soluzione del Problema 1, si vogliono ottenere una stima del cost-to-go dell'agente $\bar{c}_k(\mathbf{x}_k)$ e del costo dell'agente $c_k(\mathbf{x}_k)$.

Si osservi che la formulazione è identica al Problema 2 a meno del fatto che il costo dell'agente $c_k(\mathbf{x}_k)$ nel caso più generale possibile è tempo-variante.

La soluzione al Problema 4 è data dal seguente risultato

Risultato 5 Supponiamo che l'Assunzione 2 sia valida. Siano $\{(\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{u}}_k)\}_{1:M}$ le coppie stato/ingresso di controllo osservate con $\hat{\mathbf{x}}_{k-1} \sim p_{k|k-1}^{(x)}$, $\hat{\mathbf{u}}_k \sim p_{k|k-1}^{(u)}$ e con $p_{k|k-1}^{(u)}$ soluzione al Problema 3. La stima a massima verosomiglianza del cost-to-go $\bar{c}(\mathbf{x}_k)$, chiamata $\bar{c}^*(\mathbf{x}_k)$, è data da $\bar{c}^*(\mathbf{x}_k) = -\mathbf{w}^{*T} \mathbf{h}(\mathbf{x}_k)$ dove \mathbf{w}^* è la soluzione al seguente problema di ottimizzazione convesso:

$$\mathbf{w}^* := [\mathbf{w}_1^{*T}, \dots, \mathbf{w}_M^{*T}]^T \in \arg \min_{\mathbf{w}} \left\{ \sum_{k=1}^M \left(-\mathbb{E}_{p_k(\mathbf{x}_k|\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{u}}_k)} [\mathbf{w}_k^T \mathbf{h}(\mathbf{x}_k)] + \ln \left(\sum_{\mathbf{u}_k} \bar{q}_{k|k-1}^{(u)}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k) \exp (\mathbb{E}_{p_k(\mathbf{x}_k|\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k)} [\mathbf{w}_k^T \mathbf{h}(\mathbf{x}_k)]) \right) \right) \right\} \quad (12)$$

dove $\bar{q}_{k|k-1}^{(u)}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k) := q_k(\mathbf{u}_k, \hat{\mathbf{x}}_{k-1}) \exp (-\mathcal{D}_{KL}(p_k(\mathbf{x}_k|\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k) || q_k(\mathbf{x}_k|\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k)))$

Nel caso specifico in cui la policy di riferimento $q_{0:N}$ è uniforme, il problema di ottimizzazione nella 12 diventa

$$\mathbf{w}^* := [\mathbf{w}_1^{*T}, \dots, \mathbf{w}_M^{*T}]^T \in \arg \min_{\mathbf{w}} \left\{ \sum_{k=1}^M \left(-\mathbb{E}_{p(\mathbf{x}_k|\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{u}}_k)} [\mathbf{w}_k^T \mathbf{h}(\mathbf{x}_k)] + \ln \left(\sum_{\mathbf{u}_k} \exp (\mathbb{E}_{p(\mathbf{x}_k|\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k)} [-\ln(p(\mathbf{x}_k|\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k)) + \mathbf{w}_k^T \mathbf{h}(\mathbf{x}_k)]) \right) \right) \right\} \quad (13)$$

Appendice C Note implementative

C.1 Implementazione del problema di controllo diretto

Algorithm 1 Pseudo-codice del Risultato 1

Input $p_{k|k-1}^{(x)}$ e $c(\cdot)$

Output $p_{k|k-1}^{(u)}$

$$p_{k|k-1}^{(u)} = \frac{\exp\left(-\mathbb{E}_{p_{k|k-1}^{(x)}} [\ln(p_{k|k-1}^{(x)}) + c(\mathbf{X}_k)]\right)}{\sum_{u_k} \exp\left(-\mathbb{E}_{p_{k|k-1}^{(x)}} [\ln(p_{k|k-1}^{(x)}) + c(\mathbf{X}_k)]\right)}$$

La scelta implementativa fatta è stata quella di risolvere a ogni istante k il Problema 1 per $N = 1$. Dunque, la policy calcolata dall'Algoritmo 1 è la policy ottima nel caso $N = 1$.

Nell'implementazione dell'Algoritmo 1, il valore atteso del costo $\mathbb{E}_{p_{k|k-1}^{(x)}} [c(\mathbf{X}_k)]$ è stato stimato tramite il metodo Monte Carlo, ossia calcolando la media del costo su 20 campioni estratti. Al contrario, per il calcolo dell'entropia si è utilizzata la formula analitica senza ricorrere alla stima del valore atteso, sfruttanando la conoscenza di $p_{k|k-1}^{(x)}$ che è una bivariata Guassiana.

C.2 Implementazione del problema di controllo inverso

Algorithm 2 Pseudo codice del Risultato 2

Input le osservazioni $\hat{\mathbf{u}}_1, \dots, \hat{\mathbf{u}}_M$ e $\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_M$,
vettore delle feature f-dimensionale $\mathbf{h}(\mathbf{x}_k)$,

$p_{k|k-1}^{(x)}$

Output $c^*(\mathbf{x}_k)$

Trova il vettore \mathbf{w}_s^* risolvendo il problema di ottimizzazione 2

$$c^*(\mathbf{x}_k) \leftarrow -\mathbf{w}_s^* \mathbf{h}(\mathbf{x}_k)$$

Per l'implementazione dell'Algoritmo 2, il valore atteso del vettore di feature $\mathbb{E}_{p(\mathbf{x}_k|\hat{\mathbf{x}}_{k_1}, \mathbf{u}_k)} [\mathbf{h}(\mathbf{x}_k)]$ viene stimato sfruttando il metodo Monte Carlo calcolando la media su 5 campioni di vettori estratti; mentre per il calcolo dell'entropia si è utilizzata anche qui la formula analitica.

Per la risoluzione del problema di ottimizzazione si è utilizzato il tool CVXPY. CVXPY è un linguaggio di modellazione open source integrato in Python per problemi di ottimizzazione convessa [4]. Tale tool ha restituito il vettore di pesi ottimo \mathbf{w}_s^* il quale viene utilizzato per calcolare la stima della funzione di costo $c^*(\mathbf{x}_k)$.

Per maggiori dettagli sul codice che implementa questi due algoritmi si faccia riferimento ai commenti contenuti nell'implementazione pubblicata sulla repository Github del progetto.

Appendice D Il ruolo di $|\nabla c(\mathbf{x}_k)|$ nel problema di controllo diretto con N=1

In quest'appendice si intende approfondire il risultato a cui si è accennato nel Paragrafo 2.2. Il risultato in questione lega il comportamento qualitativo di un agente che implementa l'algoritmo 1 al modulo del gradiente della funzione di costo $|\nabla c(\mathbf{x}_k)|$. In primo luogo si fornisce un'intuizione algebrica sulla relazione tra il modulo del gradiente della funzione costo e il comportamento entropico dei robot osservato in simulazione. In un secondo momento si fornisce evidenze sperimentali che confermano la suddetta intuizione.

D.1 Il legame tra $|\nabla c(\mathbf{x}_k)|$ e il tradeoff exploration/exploitation dell'agente

I passaggi che seguono non costituiscono in alcun modo una dimostrazione matematica rigorosa e hanno il solo scopo di fornire una prospettiva differente alle evidenze sperimentali.

Come già detto, per rendere l'algoritmo di controllo computazionalmente efficiente si implementa la legge di controllo presentata nel Risultato 1, che è la soluzione ottima del Problema 1 nel caso particolare in cui $N = 1$ (soluzione greedy). Un modo alternativo di vedere l'implementazione greedy, essendo $|\mathcal{T}| = 1$, è quello di risolvere il problema sempre nell'ultimo istante della finestra temporale \mathcal{T} , ovvero risolvere per ogni k un problema di ottimizzazione analogo a quello nella 10b. Dunque, nell'implementazione greedy ad ogni istante k l'ingresso di controllo $\mathbf{u}_k \sim p_{k|k-1}^{(u)}$ è campionato dalla policy ottenuta come soluzione ottima al seguente problema di ottimizzazione:

Problema 5 Trova la policy $p_{k|k-1}^{(u)*}$ tale per cui:

$$\begin{aligned} p_{k|k-1}^{(u)*} &\in \arg \min_{p_{k|k-1}^{(u)}} \mathbb{E}_{\bar{p}_{k-1}} [-H(p_{k|k-1}) + \mathbb{E}_{p_{k|k-1}} [c(\mathbf{X}_k)]] \\ \text{s.t. } p_{k|k-1}^{(u)} &\in \mathcal{D} \end{aligned} \quad (14)$$

dove $\bar{p}_{k-1} := p_{k-1}(\mathbf{x}_{k-1})$ e $H(p_{k|k-1})$ è l'entropia della funzione di probabilità $p_{k|k-1} = p(\mathbf{x}_k, \mathbf{u}_k | \mathbf{x}_{k-1})$.

Siccome la variabile decisionale $p_{k|k-1}^{(u)}$ è indipendente dalla funzione di probabilità \bar{p}_{k-1} si può semplificare il problema di ottimizzazione ottenendo

$$\begin{aligned} p_{k|k-1}^{(u)*} &\in \arg \min_{p_{k|k-1}^{(u)}} -H(p_{k|k-1}) + \mathbb{E}_{p_{k|k-1}} [c(\mathbf{X}_k)] \\ \text{s.t. } p_{k|k-1}^{(u)} &\in \mathcal{D} \end{aligned}$$

A questo punto si suppone che $c(\mathbf{x})$ sia differenziabile e se ne considera lo sviluppo in serie di Taylor del primo ordine. Qualitativamente si può affermare che, dato uno stato \mathbf{x}_{k-1} , in un intorno sufficientemente piccolo del punto \mathbf{x}_{k-1} vale l'approssimazione

$$c(\mathbf{x}) \approx c(\mathbf{x}_{k-1}) + \nabla^T c(\mathbf{x}_{k-1}) \cdot (\mathbf{x} - \mathbf{x}_{k-1})$$

e inoltre l'accuratezza di tale approssimazione incrementa al diminuire di $|\mathbf{x} - \mathbf{x}_{k-1}|$. In particolare scegliendo $\mathbf{x} = \mathbf{x}_k$ si ottiene l'approssimazione

$$c(\mathbf{x}_k) \approx c(\mathbf{x}_{k-1}) + \nabla^T c(\mathbf{x}_{k-1}) \cdot (\mathbf{x}_k - \mathbf{x}_{k-1})$$

Si osservi che, dati il limiti di velocità imposti dal Robotarium, siamo anche in grado di individuare un limite superiore per $|\mathbf{x}_k - \mathbf{x}_{k-1}|$, dovendo necessariamente essere

$$|\mathbf{x}_k - \mathbf{x}_{k-1}| = |\mathbf{x}_{k-1} + \mathbf{u}dt - \mathbf{x}_{k-1}| \leq ||\mathbf{u}|_{max} dt| = 0.023m$$

dove si è usato il fatto che il time step del Robotarium è pari a $dt = 0.033s$ e che ciascuna delle componenti di u è limitata ad avere modulo minore inferiore a $0.5m/s$.

Se l'errore di approssimazione è sufficientemente basso, la soluzione al precedente problema di ottimizzazione non si discosta molto dalla soluzione al seguente problema:

$$\begin{aligned} p_{k|k-1}' &\in \arg \min_{p_{k|k-1}^{(u)}} -H(p_{k|k-1}) + \mathbb{E}_{p_{k|k-1}} \left[c(\mathbf{x}_{k-1}) + \nabla^T c(\mathbf{x}_{k-1}) \cdot (\mathbf{x}_k - \mathbf{x}_{k-1}) \right] \\ \text{s.t. } p_{k|k-1}' &\in \mathcal{D} \end{aligned}$$

Si osservi che $c(\mathbf{x}_{k-1})$ è indipendente sia dalla funzione di probabilità $p_{k|k-1}$ che dalla variabile decisionale $p_{k|k-1}^{(u)}$, dunque si può escludere tale termine dal problema di ottimizzazione. Inoltre, sviluppando il prodotto scalare

$$\nabla^T c(\mathbf{x}_{k-1}) \cdot (\mathbf{x}_k - \mathbf{x}_{k-1}) = |\nabla c(\mathbf{x}_{k-1})| |\mathbf{x}_k - \mathbf{x}_{k-1}| \cos \theta_k$$

con θ_k angolo compreso tra i vettori $\nabla c(\mathbf{x}_{k-1})$ e $(\mathbf{x}_k - \mathbf{x}_{k-1})$. Il problema di ottimizzazione quindi diventa

$$\begin{aligned} p_{k|k-1}^{(u)} &\in \arg \min_{p_{k|k-1}^{(u)}} -H(p_{k|k-1}) + \mathbb{E}_{p_{k|k-1}} \left[|\nabla c(\mathbf{x}_{k-1})| |\mathbf{x}_k - \mathbf{x}_{k-1}| \cos \theta_k \right] \\ \text{s.t. } p_{k|k-1}^{(u)} &\in \mathcal{D} \end{aligned}$$

Infine, osservando che è $|\nabla c(\mathbf{x}_{k-1})|$ indipendente dalla funzione di probabilità $p_{k|k-1}$ si può scrivere

$$\begin{aligned} p_{k|k-1}^{(u)} &\in \arg \min_{p_{k|k-1}^{(u)}} -H(p_{k|k-1}) + \left| \nabla c(\mathbf{x}_{k-1}) \right| \mathbb{E}_{p_{k|k-1}} \left[|\mathbf{x}_k - \mathbf{x}_{k-1}| \cos \theta_k \right] \\ \text{s.t. } p_{k|k-1}^{(u)} &\in \mathcal{D} \end{aligned}$$

Tale risultato ci permette di comprendere meglio il ruolo di $|\nabla c(\mathbf{x}_{k-1})|$ nella soluzione greedy al Problema 1: la funzione obiettivo del problema di ottimizzazione è costituita da due termini contrastanti:

- $-H(p_{k|k-1})$ che promuove l'esplorazione dell'agente nell'ambiente. Se fosse presente solo questo termine la policy ottima sarebbe tale da rendere $p_{k|k-1}$ una distribuzione uniforme. Empiricamente questo corrisponde a robot che compiono movimenti "casuali" (comportamento che talvolta è stato definito "entropico");
- $\mathbb{E}_{p_{k|k-1}} \left[|\mathbf{x}_k - \mathbf{x}_{k-1}| \cos \theta_k \right]$ che promuove la cosiddetta "exploitation" della funzione di costo. Se fosse presente solo questo termine la policy ottima sarebbe tale da inseguire la direzione opposta a quella del gradiente $\nabla c(\mathbf{x}_{k-1})$.

Nella relazione per fare riferimento ai movimenti "casuali" del robot si utilizza la locuzione "comportamento entropico" in quanto tale comportamento è indotto dal termine $H(\cdot)$ nel problema di ottimizzazione. Osservando che $|\nabla c(\mathbf{x}_{k-1})|$ ha il ruolo di peso relativo del secondo termine della funzione obiettivo rispetto al primo, possiamo trarre le seguenti conclusioni:

- Se $|\nabla c(\mathbf{x}_{k-1})|$ è "basso", sarà promosso il comportamento entropico dell'agente;
- Se $|\nabla c(\mathbf{x}_{k-1})|$ è "alto", sarà promossa l'exploitation della funzione costo, ovvero l'agente seguirà la direzione opposta a quella del gradiente della funzione costo.

D.2 Evidenze sperimentali

A questo punto si riportano i risultati sperimentali provenienti da simulazioni realizzate con lo scopo di verificare l'intuizione di cui sopra.

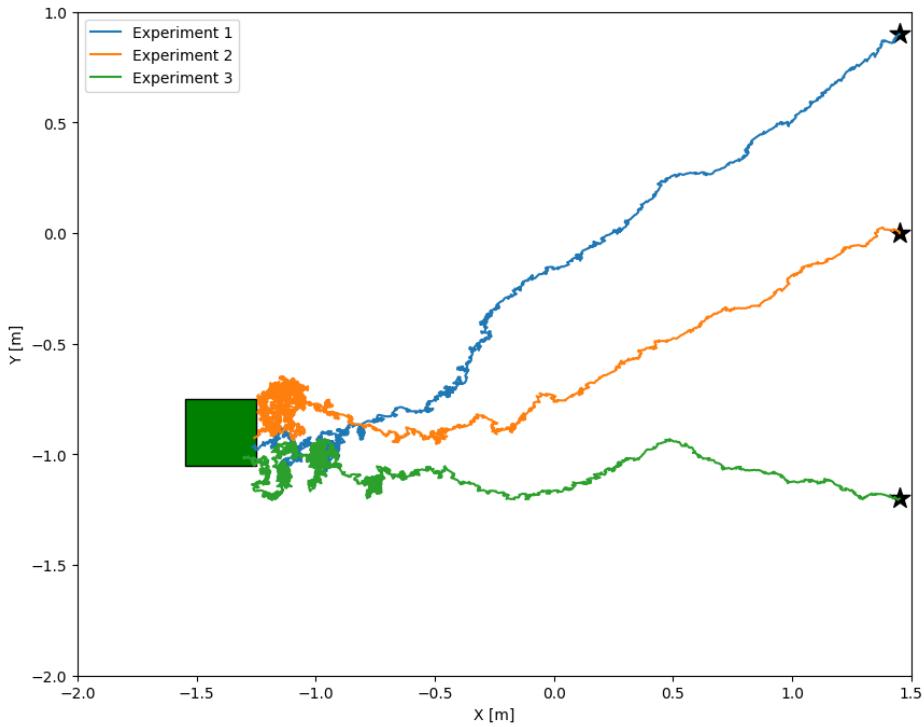
Una prima simulazione mostrata in Fig. 8 è stata fatta utilizzando come funzione di costo

$$c(\mathbf{x}_k) = 15|\mathbf{x}_k - \mathbf{x}_d|_2^2$$

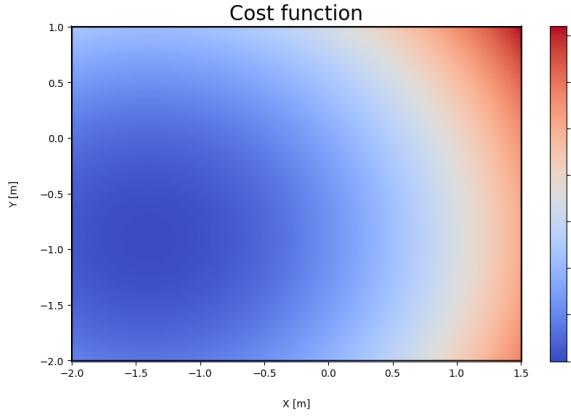
dove \mathbf{x}_d è la posizione del goal point. Il gradiente della funzione è

$$\nabla c(\mathbf{x}_k) = 30(\mathbf{x}_k - \mathbf{x}_d)$$

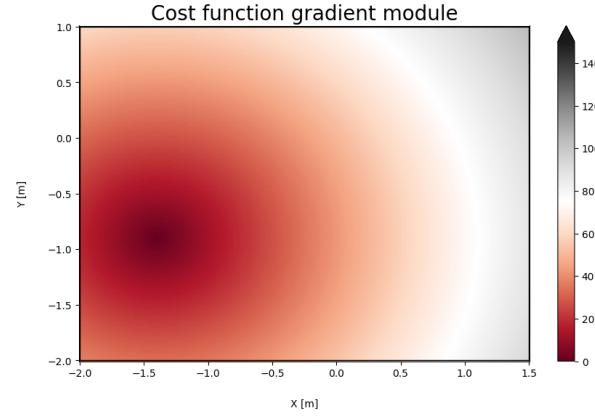
Si osservi che per $\mathbf{x}_k \rightarrow \mathbf{x}_d$ si ha che $|\nabla c(\mathbf{x}_k)| \rightarrow 0$, dunque da quanto discusso nel paragrafo precedente al diminuire della distanza dal goal point ci si aspetta che il comportamento entropico dei robot aumenti. Come evidenziato in Fig. 8a l'esperimento conferma l'intuizione.



(a) Traiettorie di tre robot



(b) Modulo della funzione costo



(c) Modulo del gradiente della funzione costo

Figura 8: Simulazione con $c(\mathbf{x}_k) = 15|\mathbf{x}_k - \mathbf{x}_d|_2^2$

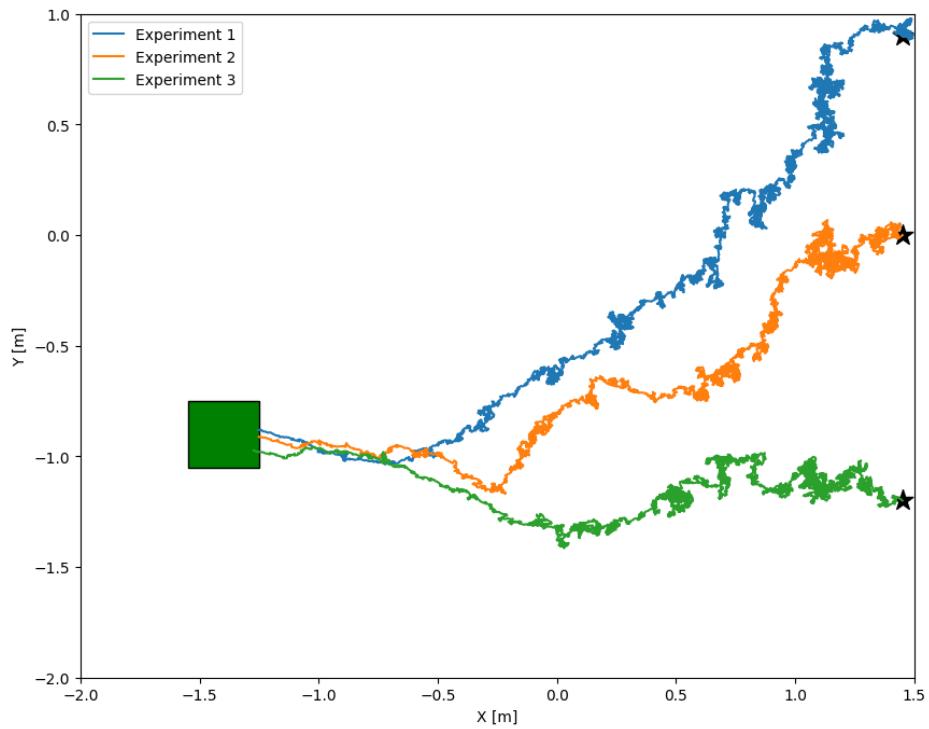
A questo punto si potrebbe osservare che empiricamente potrebbe esserci una corrispondenza tra $|c(\mathbf{x}_k)|$ e il comportamento entropico. Per dimostrare che non è questo il caso, abbiamo realizzato una seconda simulazione mostrata in Fig. 9. In questo caso si è scelta come funzione costo

$$c(\mathbf{x}_k) = 30 \ln(|\mathbf{x}_k - \mathbf{x}_d|_2)$$

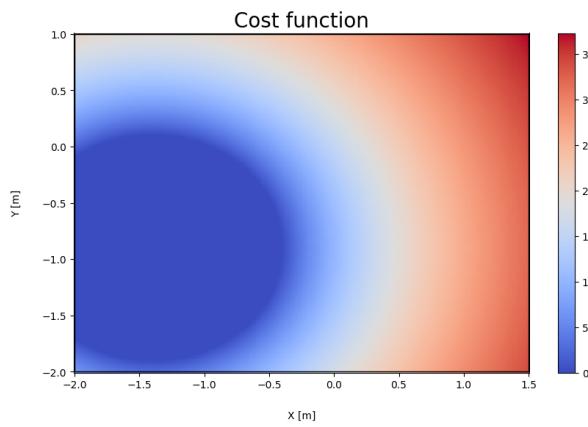
che ha come gradiente

$$\nabla c(\mathbf{x}_k) = \frac{30}{|\mathbf{x}_k - \mathbf{x}_d|_2^2} (\mathbf{x}_k - \mathbf{x}_d)$$

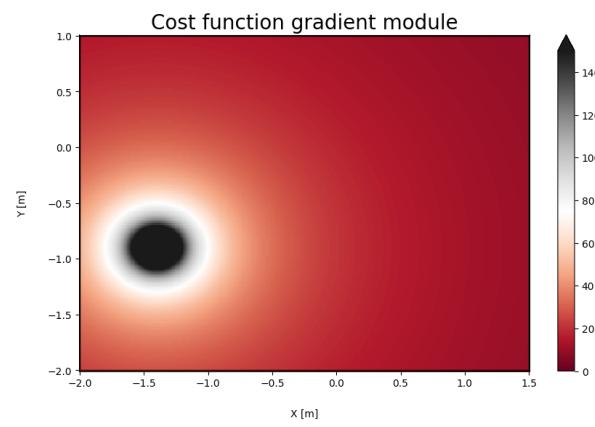
In questo caso per $\mathbf{x}_k \rightarrow \mathbf{x}_d$ si ha che $|\nabla c(\mathbf{x}_k)| \rightarrow +\infty$, dunque da quanto discusso nel paragrafo precedente, al diminuire della distanza dal goal point ci si aspetta che il comportamento entropico dei robot diminuisca. Come evidenziato in Fig. 9a l'esperimento conferma l'intuizione.



(a) Traiettorie di tre robot



(b) Modulo della funzione costo



(c) Modulo del gradiente della funzione costo

Figura 9: Simulazione con $c(\mathbf{x}_k) = 30 \ln(|\mathbf{x}_k - \mathbf{x}_d|_2)$

Appendice E Progettazione della funzione di costo proposta

In questa appendice si ripercorrono le scelte progettuali relative alla realizzazione della funzione di costo proposta. In primo luogo si presenta un'analisi della funzione di costo fornita, nella quale si impiegheranno anche risultati sperimentali per evidenziarne i problemi. In secondo luogo si proporrà la funzione di costo che risolve i suddetti problemi e si commentano le simulazioni realizzate con la funzione di costo proposta.

E.1 Analisi funzione di costo fornita

Siano $\mathbf{x}_k = [x_k, y_k]^T \in \mathbb{R}^2$ la posizione corrente del robot, $\mathbf{x}_d \in \mathbb{R}^2$ il goal point, $\mathbf{o}_i \in \mathbb{R}^2$ la posizione dell'ostacolo i -esimo ed n il numero di ostacoli. La funzione di costo fornita è:

$$c(\mathbf{x}_k) = 30|\mathbf{x}_k - \mathbf{x}_d|_2^2 + 20 \sum_{i=1}^n g_i(\mathbf{x}_k) + 10 \sum_{j=1}^4 h_j(\mathbf{x}_k) \quad (15)$$

dove:

$$g_i(\mathbf{x}_k) = \frac{1}{\sqrt{(2\pi)^2 \det(\Sigma_o)}} \exp \left(-\frac{1}{2} (\mathbf{x}_k - \mathbf{o}_i)^T \Sigma_o^{-1} (\mathbf{x}_k - \mathbf{o}_i) \right)$$

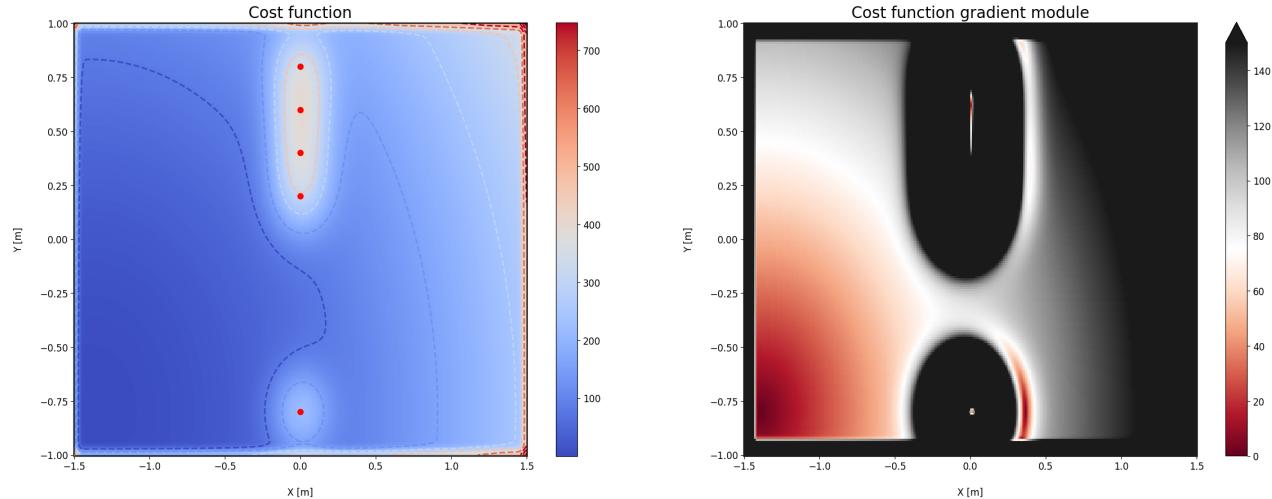
è una funzione gaussiana bivariata con vettore della media $\mu = \mathbf{o}_i$ e matrice di covarianza $\Sigma_o = \begin{bmatrix} 0.02 & 0 \\ 0 & 0.02 \end{bmatrix}$ e

$$\begin{aligned} h_1(\mathbf{x}_k) &= \frac{1}{(0.02)\sqrt{2\pi}} \exp \left(-\frac{1}{2} \left(\frac{x_k - (-1.5)}{(0.02)} \right)^2 \right) \\ h_2(\mathbf{x}_k) &= \frac{1}{(0.02)\sqrt{2\pi}} \exp \left(-\frac{1}{2} \left(\frac{x_k - (1.5)}{(0.02)} \right)^2 \right) \\ h_3(\mathbf{x}_k) &= \frac{1}{(0.02)\sqrt{2\pi}} \exp \left(-\frac{1}{2} \left(\frac{y_k - (-1)}{(0.02)} \right)^2 \right) \\ h_4(\mathbf{x}_k) &= \frac{1}{(0.02)\sqrt{2\pi}} \exp \left(-\frac{1}{2} \left(\frac{y_k - (1)}{(0.02)} \right)^2 \right) \end{aligned}$$

sono quattro funzioni gaussiane univariate con varianza $\sigma^2 = 0,02$. Le gaussiane h_1 e h_2 sono calcolate rispetto all'asse x e hanno medie $\mu_1 = -1,5$, $\mu_2 = 1,5$. Le gaussiane h_3 e h_4 sono calcolate rispetto all'asse y e hanno medie $\mu_3 = -1$, $\mu_4 = 1$. Ai fini del problema di controllo, la funzione di costo può essere interpretata come segue:

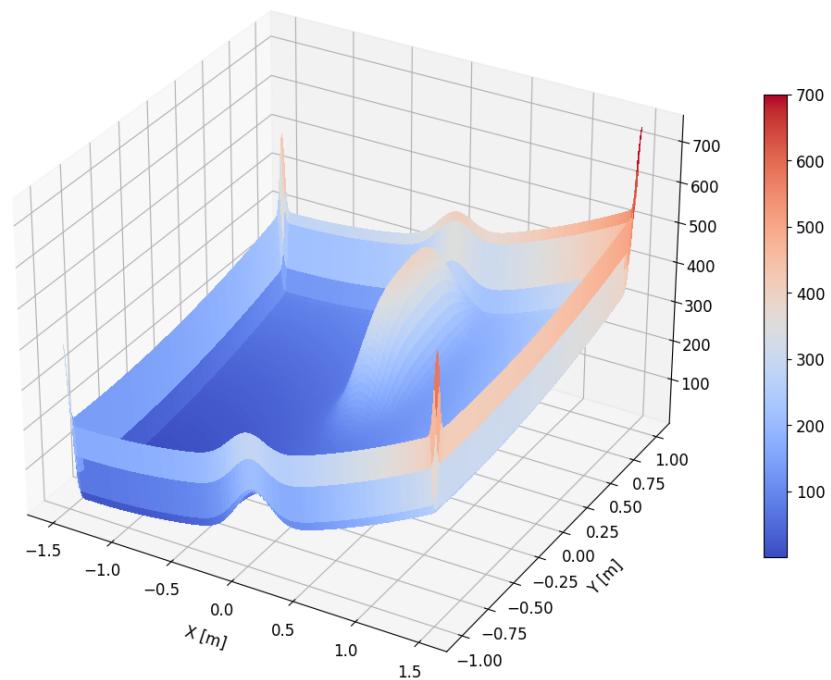
- Il primo termine $|\mathbf{x}_k - \mathbf{x}_d|^2$ è un paraboloida ellittico il cui minimo globale è centrato nella posizione dell'obiettivo \mathbf{x}_d . Questo termine viene utilizzato per attirare i robot verso l'obiettivo.
- Ogni ostacolo è modellato da una funzione gaussiana bivariata g_i centrata nella posizione dell'ostacolo \mathbf{o}_i . Questo termine ha un massimo sull'ostacolo e viene utilizzato per spingere i robot lontano dagli ostacoli.
- Ogni confine del Robotarium è rappresentato da una funzione gaussiana h_j . Le gaussiane h_1 e h_2 sono centrate nelle coordinate x del bordo sinistro e destro rispettivamente. Le gaussiane h_3 e h_4 sono centrate nelle coordinate y dei bordi inferiore e superiore rispettivamente. Questi termini vengono utilizzati per allontanare i robot dalle pareti.
- I pesi relativi dei termini della funzione di costo sono stati scelti in modo da ottenere un opportuno equilibrio tra i termini.

Nelle Figg. 10 si mostrano alcuni plot relativi alla funzione di costo fornita.



(a) Heatmap della funzione di costo
Cost function

(b) Modulo del gradiente



(c) Plot 3D della funzione di costo
Figura 10: Plot della funzione di costo fornita

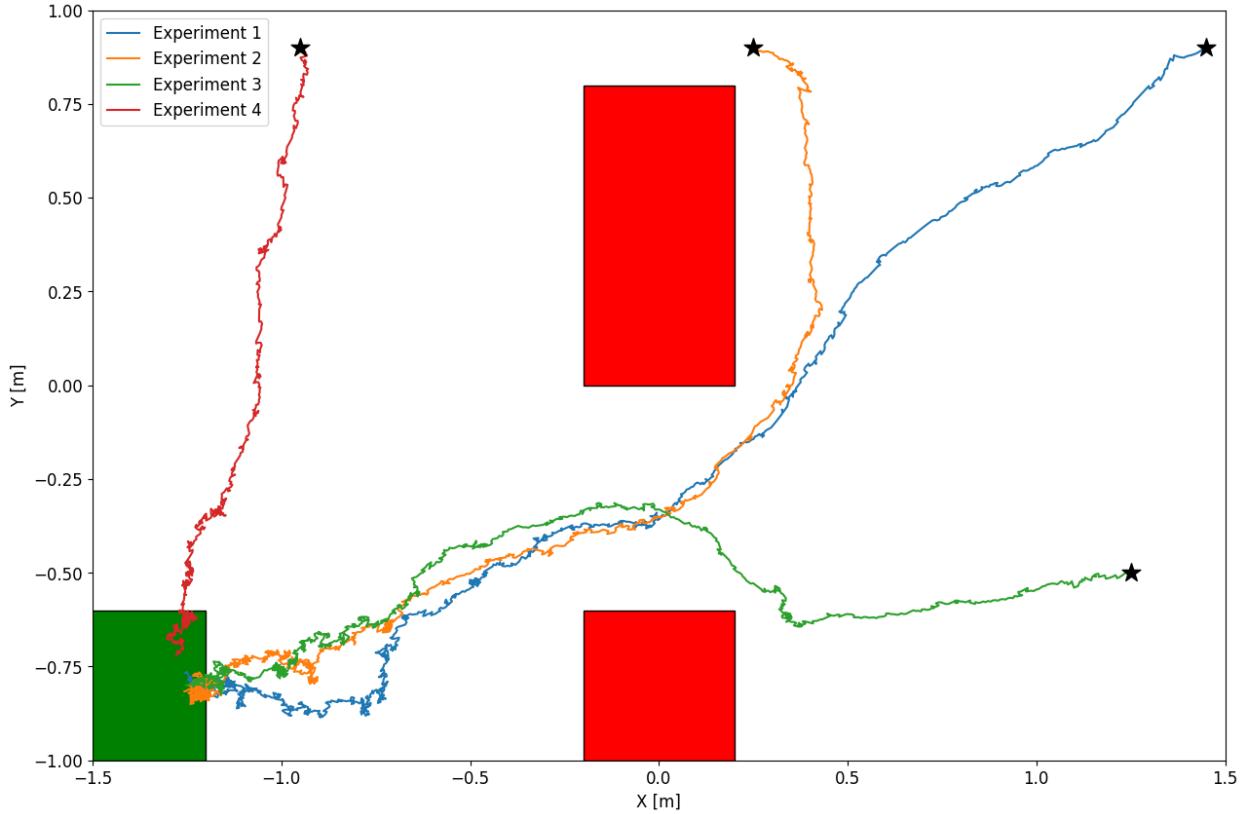


Figura 11: Traiettorie con funzione di costo fornita

E.2 Risultati delle simulazioni con la funzione di costo fornita

Nella Fig. 11 è mostrata una simulazione realizzata nello scenario di base con la funzione di costo fornita. Da tale simulazione si evincono alcune problematiche per la funzione di costo fornita che vengono approfondite in seguito.

Problema 1 Traiettorie oscillanti

Nella simulazione in Fig. 12a si nota come le traiettorie dei robot siano oscillanti. Tale comportamento è particolarmente accentuato nelle vicinanze del goal point. Questo fenomeno è il cosiddetto comportamento indotto dal regolarizzatore entropico di cui si è parlato nell'Appendice D. Nella Fig. 12b si può osservare che le zone in cui il modulo del gradiente della funzione costo è basso sono quelle in cui è favorito il comportamento entropico. Una possibile soluzione a tale problema è quello di rendere il goal point più attrattivo per i robot facendo sì che il modulo del gradiente della funzione costo sia maggiore in prossimità del goal point.

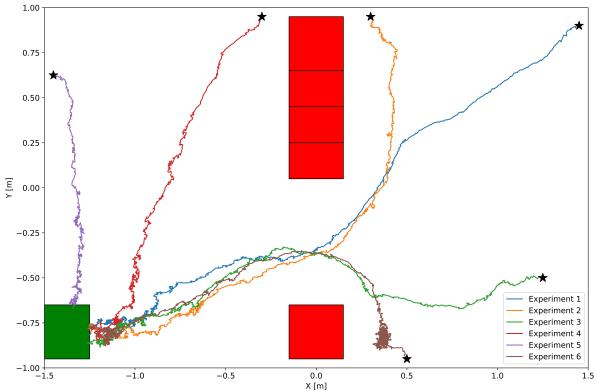
Problema 2 Robot che non aggirano ostacoli

Nella simulazione in Fig. 13a, si evidenzia come nel caso di una parete costituita da tre ostacoli, il comportamento entropico del robot non è sufficiente per aggirare gli ostacoli nel tempo stabilito. Questo accade poiché il robot rimane incastrato in un minimo locale della funzione di costo. Tale ipotesi è giustificata dal plot del gradiente, in Fig. 13b, nel quale la regione con il modulo del gradiente "basso" è troppo sottile e troppo in prossimità degli ostacoli per evitare che il robot rimanga incastrato.

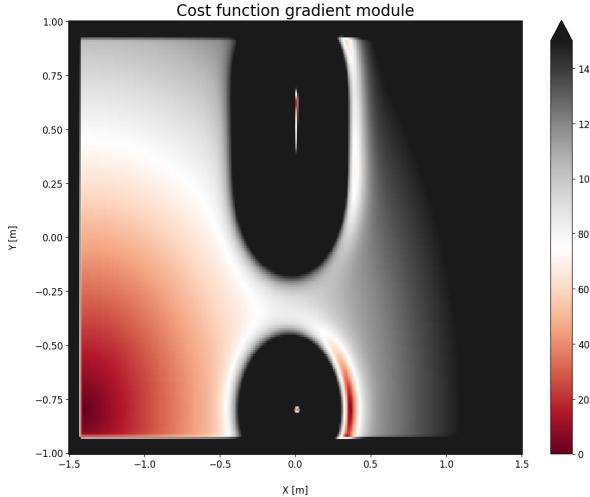
Una possibile soluzione a questo problema è modificare i termini della funzione di costo che modellano gli ostacoli allo scopo di ingrandire la regione con il modulo del gradiente basso.

Problema 3 Robot al di fuori dei bordi

Un ulteriore problema emerge quando i robot sono troppo vicini ai bordi del Robotarium, come in Fig. 14a. In questa situazione i robot potrebbero uscire al di fuori dei limiti del Robotarium. Avendo implementato un algoritmo greedy, ad ogni istante di tempo, l'algoritmo di controllo sceglie l'azione attuale considerando solo il valore atteso del costo dello stato futuro ($\mathbb{E}[c(\mathbf{x}_k)]$) per ogni azione di controllo u_k . Siccome $\mathbf{x}_k \sim \mathcal{N}(\mathbf{x}_{k-1} + \mathbf{u}_k dt, \Sigma)$, si ha che $\mathbb{E}[\mathbf{x}_k] = \mathbf{x}_{k-1} + \mathbf{u}_k dt$. La piattaforma Robotarium ha un $dt = 0.033s$ e la velocità massima del robot per ogni sua componente è $v_{max} = 0.5m/s$. Tenendo conto del fatto che a ogni istante di tempo k il robot potrebbe avere un orientamento qualsiasi, per scegliere l'azione successiva l'algoritmo potrebbe valutare la funzione di costo sui punti di una circonferenza di raggio massimo $\max_{u_k} \mathbb{E} [|\mathbf{x}_k - \mathbf{x}_{k-1}|] = |\mathbf{u}_k|_{max} dt = \sqrt{2}v_{max}dt \approx 0.023m$. Inoltre è importante osservare che l'algoritmo non tiene conto dei valori che la funzione di costo assume nei punti intermedi tra la posizione del robot \mathbf{x}_{k-1} e la posizione successiva \mathbf{x}_k . Ciò comporta che, se il robot è vicino alle barriere, l'agente potrebbe valutare la funzione di costo in punti esterni dal Robotarium e, se il valore atteso del costo in questi punti è basso, allora il robot potrebbe uscire al di fuori del Robotarium ignorando la presenza delle barriere. Questo discorso è evidenziato nella Fig. 14b, in cui

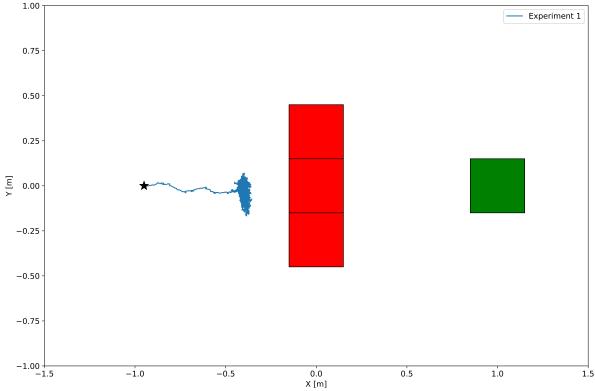


(a) Simulazione con la funzione di costo fornita

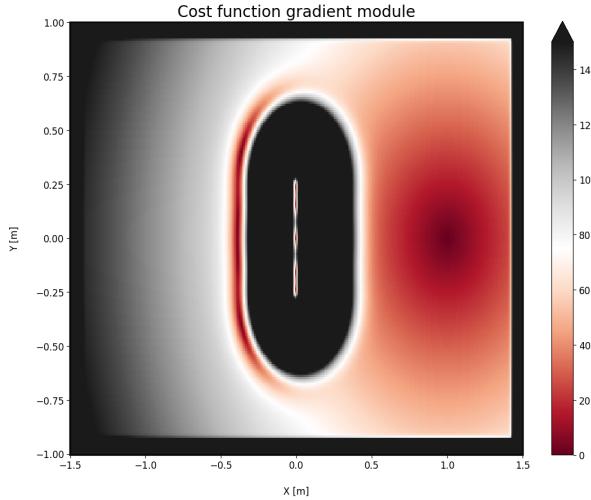


(b) Gradiente della funzione di costo fornita

Figura 12: Traiettorie oscillanti in prossimità del goal point



(a) Simulazione con la funzione di costo fornita



(b) Gradiente della funzione di costo fornita

Figura 13: Robot incastrato davanti a una parete

si mostra una heatmap del termine della funzione costo che modella un bordo del Robotarium sulla quale è sovrapposta l' "orizzonte di predizione" dell'agente.

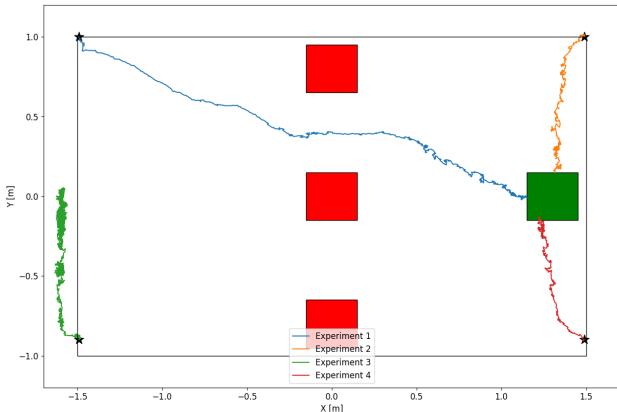
Per risolvere questo problema si può utilizzare la funzione indicatore definita sull'insieme di punti interni al Robotarium per fare in modo che la funzione di costo assuma valori molto alti al di fuori dei limiti consentiti. Inoltre per evitare che i robot si avvicinino troppo alle confini si è deciso anche di aumentare la varianza delle gaussiane che modellano questi ultimi.

E.3 Proposta della funzione di costo

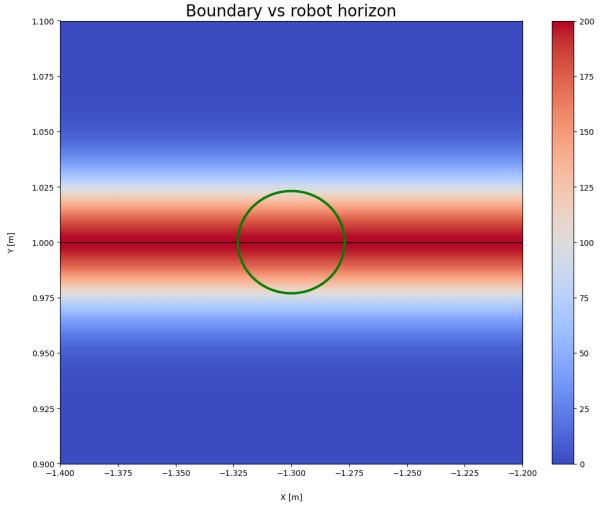
Per trovare i parametri della funzione di costo è stato utilizzato il notebook Python in "tools/cost_function_tuners.ipynb" sviluppato appositamente per questo scopo. Il modulo consente di regolare in modo interattivo i parametri di diverse funzioni di costo e di visualizzare la funzione di costo e il suo modulo di gradiente in tempo reale. Al fine di decidere la funzione di costo da proporre, abbiamo eseguito simulazioni con diverse funzioni di costo e per ognuna di esse ne abbiamo regolato i parametri. Infine, abbiamo selezionato la migliore funzione di costo in base alle performance relative alle problematiche evidenziate nella sezione precedente.

Sia $\mathbf{x}_k = [x_k, y_k]^T \in \mathbb{R}^2$ la posizione attuale del robot, $\mathbf{x}_d \in \mathbb{R}^2$ l'obiettivo desiderato dal robot, $\mathbf{o}_i \in \mathbb{R}^2$ la posizione dell'i-esimo ostacolo, n il numero di ostacoli. Sia $\mathcal{B} = [-1.5, 1.5] \times [-1, 1] \subset \mathbb{R}^2$. La funzione di costo proposta è:

$$c(\mathbf{x}_k) = 30 \left[|\mathbf{x}_k - \mathbf{x}_d|_2^2 - \frac{1}{|\mathbf{x}_k - \mathbf{x}_d|_2} \right] + 30 \sum_i^n g_i^{(1)}(\mathbf{x}_k) + 15 \sum_i^n g_i^{(2)}(\mathbf{x}_k) + 10 \sum_{j=1}^4 h_j(\mathbf{x}_k) + \mathcal{X}_{\mathcal{B}}(\mathbf{x}_k) \quad (16)$$



(a) Simulazione con la funzione di costo fornita



(b) Bordo della funzione vs orizzonte di predizione

Figura 14: Robot che oltrepassa le barriere

dove $\epsilon = 0.1$ e

$$\chi_{\mathcal{B}}(\mathbf{x}_k) = \begin{cases} 0 & \text{if } \mathbf{x}_k \in \mathcal{B} \\ +\infty & \text{if } \mathbf{x}_k \notin \mathcal{B} \end{cases}$$

è la funzione indicatore e le funzioni

$$g_i^{(1)}(\mathbf{x}_k) = \frac{1}{\sqrt{(2\pi)^2 \det(\Sigma_1)}} \exp \left(-\frac{1}{2} (\mathbf{x}_k - \mathbf{o}_i)^T \Sigma_1^{-1} (\mathbf{x}_k - \mathbf{o}_i) \right)$$

$$g_i^{(2)}(\mathbf{x}_k) = \frac{1}{\sqrt{(2\pi)^2 \det(\Sigma_2)}} \exp \left(-\frac{1}{2} (\mathbf{x}_k - \mathbf{o}_i)^T \Sigma_2^{-1} (\mathbf{x}_k - \mathbf{o}_i) \right)$$

sono funzioni gaussiane bivariate con vettore delle medie $\mu = \mathbf{o}_i$ e matrici di covarianza $\Sigma_1 = \begin{bmatrix} 0.14 & 0 \\ 0 & 0.14 \end{bmatrix}$ e $\Sigma_2 = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$, e le funzioni

$$h_1(x_k) = \frac{1}{0.05\sqrt{2\pi}} \exp \left(-\frac{1}{2} \left(\frac{x_k - (-1.5)}{0.05} \right)^2 \right)$$

$$h_2(x_k) = \frac{1}{0.05\sqrt{2\pi}} \exp \left(-\frac{1}{2} \left(\frac{x_k - (1.5)}{0.05} \right)^2 \right)$$

$$h_3(y_k) = \frac{1}{0.05\sqrt{2\pi}} \exp \left(-\frac{1}{2} \left(\frac{y_k - (-1)}{0.05} \right)^2 \right)$$

$$h_4(y_k) = \frac{1}{0.05\sqrt{2\pi}} \exp \left(-\frac{1}{2} \left(\frac{y_k - (1)}{0.05} \right)^2 \right)$$

sono quattro funzioni gaussiane univariate con varianza $\sigma^2 = 0.05$. Le gaussiane h_1 e h_2 sono calcolate rispetto all'asse x e hanno medie $\mu_1 = -1.5$, $\mu_2 = 1.5$. Le gaussiane h_3 e h_4 sono calcolate rispetto all'asse y e hanno media $\mu_3 = -1$, $\mu_4 = 1$.

Le differenze rispetto alla funzione di costo originale sono le seguenti:

- È stato aggiunto il termine: $-\frac{1}{|\mathbf{x}_k - \mathbf{x}_d|_2 + \epsilon}$, che è l'inverso della distanza euclidea tra il robot e il punto di arrivo. Questo termine ha un minimo nella posizione del goal point ed è un attrattore più forte di quello originale, poiché il modulo del gradiente è più alto in prossimità del goal point. Questo termine è utile per evitare il comportamento entropico dei robot in prossimità del goal point. La presenza del coefficiente ϵ serve a evitare la discontinuità della funzione nel punto $\mathbf{x}_k = \mathbf{x}_d$.
- Nella funzione di costo originale gli ostacoli sono modellati da una singola funzione gaussiana. Nella nuova funzione di costo proposta gli ostacoli sono modellati da due funzioni gaussiane: g_1 e g_2 . La prima funzione gaussiana ha una varianza più alta di quella originale e viene utilizzata per avere un modulo a basso gradiente in una zona più ampia dell'ambiente.

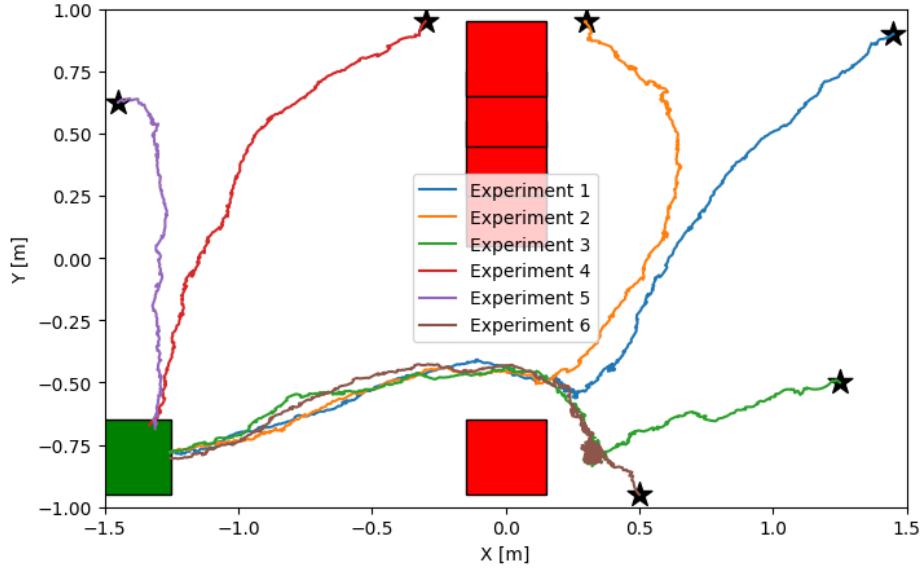
Ciò consente ai robot di esplorare maggiormente l'ambiente e di evitare di rimanere intrappolati in prossimità di un ostacolo. La seconda gaussiana ha una varianza più bassa ed è necessaria per evitare i rari casi in cui, durante l'esplorazione, i robot possono camminare sopra gli ostacoli.

- Sono state modificate le varianze delle funzioni gaussiane h_1, h_2, h_3, h_4 che modellano le pareti, ora più alte di quelle originali. Questa modifica è stata apportata per tenere i robot lontani dalle pareti.
- È stata aggiunta la funzione indicatore $\chi_B(x_k)$ per modellare i limiti dell'ambiente. In questo modo i robot non possono più uscire dai confini. Si noti che, per motivi di implementazione, è stato usato un numero alto invece di $+\infty$ dove richiesto.

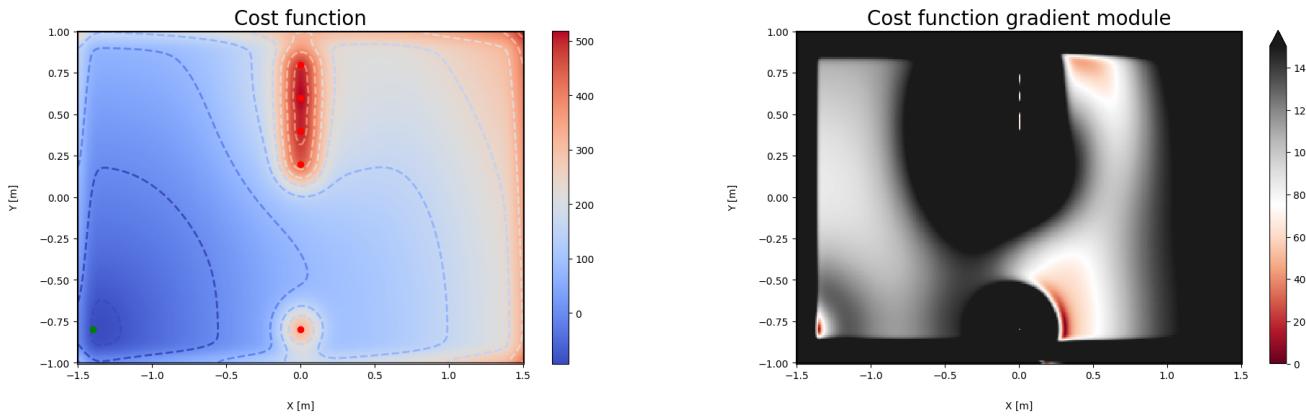
E.4 Risultati delle simulazioni con la funzione di costo proposta

In questa sezione sono presentati i risultati sperimentali in cui si evidenzia come la funzione di costo proposta risolve i problemi precedentemente evidenziati.

Nelle Figg. 15 si mostrano i risultati ottenuti con la nuova funzione di costo nello scenario di base. È possibile apprezzare come le traiettorie siano molto meno oscillanti del caso precedente. Tali cambiamenti sono dovuti al cambiamento del gradiente della funzione costo evidenziato in Fig. 15c. In particolare si osservi come in prossimità del goal point il gradiente sia comunque elevato.



(a) Traiettorie dei robot con il costo proposto



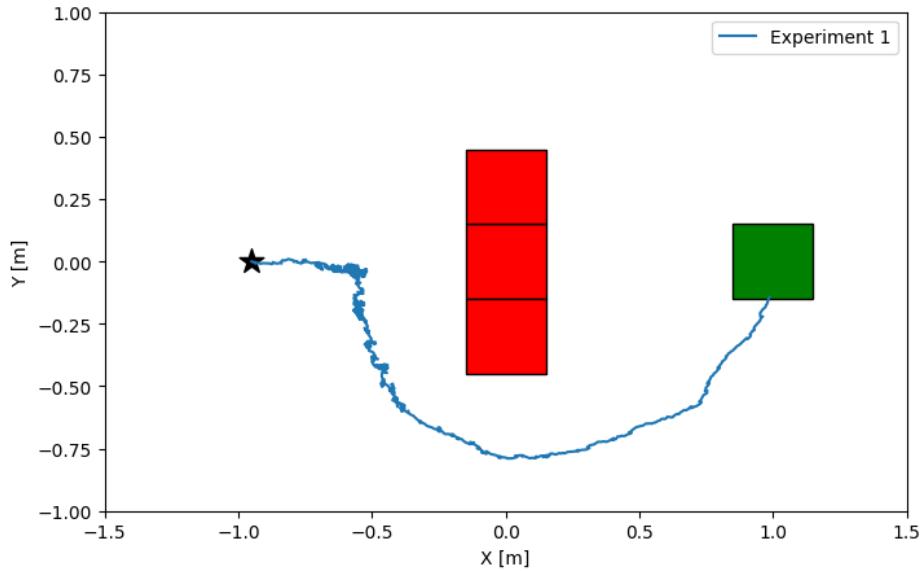
(b) Heatmap della funzione di costo proposta

(c) Gradiente della funzione di costo proposta

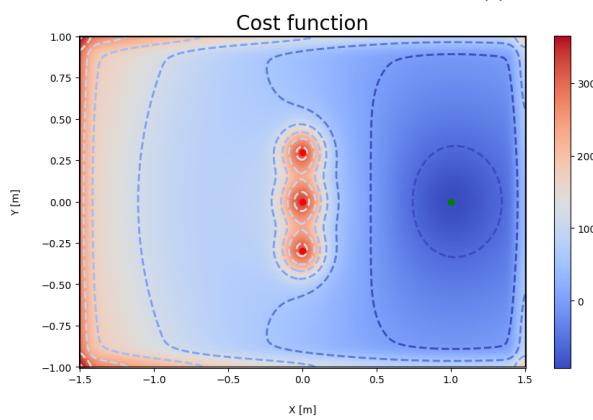
Figura 15: Simulazione dello scenario di base

Nelle Figg. 16 si mostra il comportamento della nuova funzione di costo con una parete di tre ostacoli. In questo caso manipolando il gradiente della funzione costo per ottenere la forma in Fig. 16c si ottiene il comportamento in Fig. 16a. In particolare, si sta sfruttando il comportamento entropico del robot per permettergli di aggirare gli ostacoli.

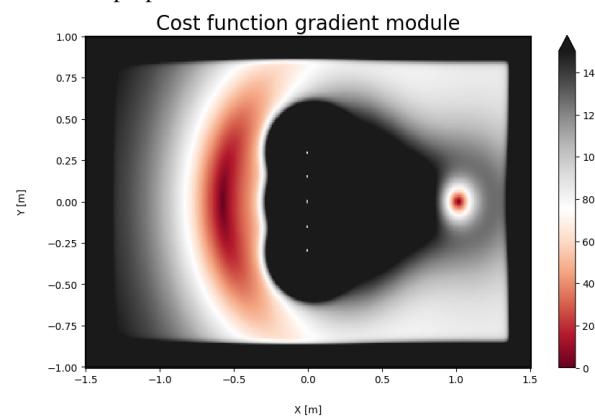
Nelle Figg. 17 si evidenzia il comportamento della funzione di costo proposta in uno scenario più complesso. Si osservi come le traiettorie dei robot siano molto meno oscillanti.



(a) Traiettorie del robot con il costo proposto

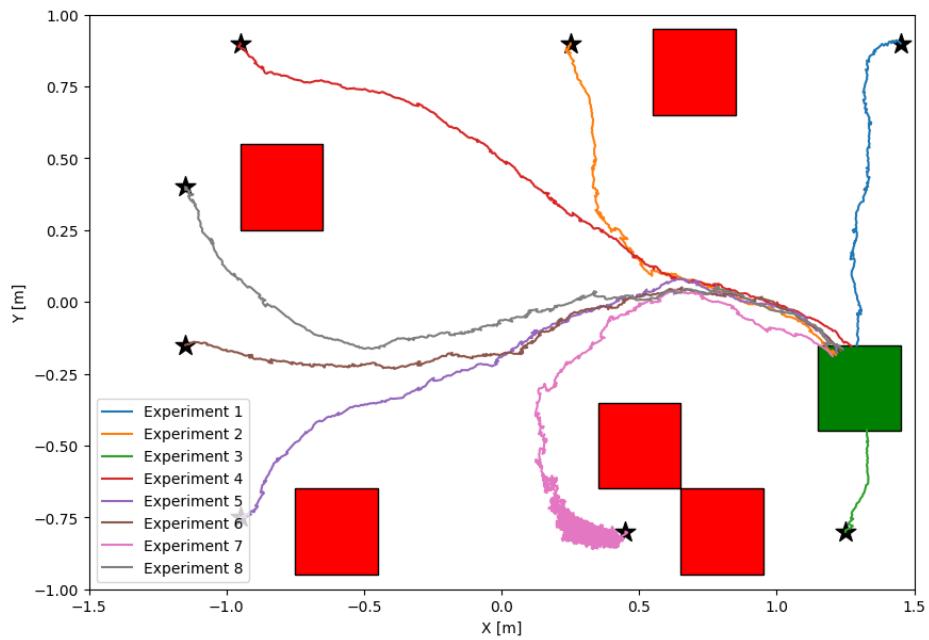


(b) Traiettorie del robot con il costo proposto

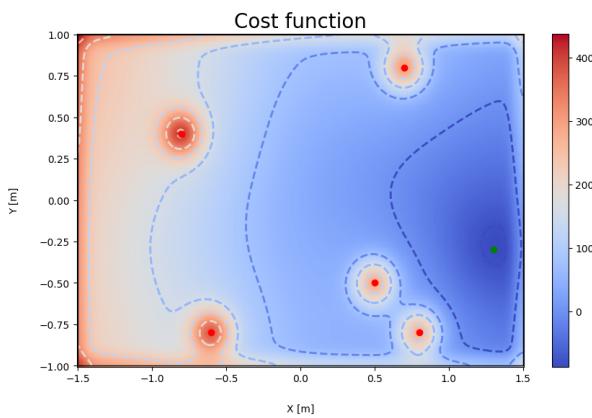


(c) Gradiente della funzione di costo proposta

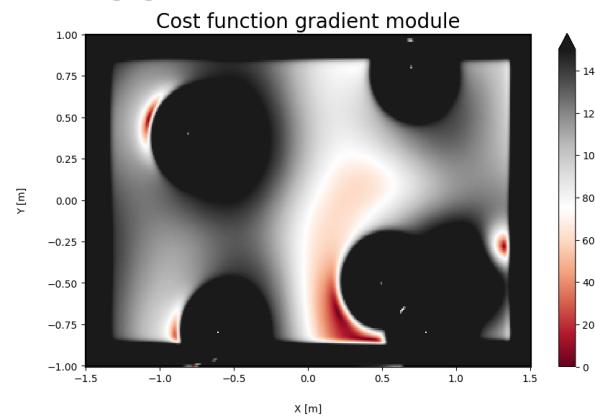
Figura 16: Simulazione con una parete di ostacoli



(a) Traiettorie dei robot con il costo proposto



(b) Traiettorie dei robot con il costo proposto



(c) Gradiente della funzione di costo proposta

Figura 17: Simulazione in uno scenario più complicato

Appendice F Progettazione delle feature proposte

In questa appendice si ripercorrono le scelte progettuali relative alla progettazione delle feature. In primo luogo, si presenta un'analisi della feature fornite nella quale si faranno uso di risultati sperimentali per evidenziare i principali problemi individuati. In secondo luogo, si proporanno delle nuove feature che risolvono i suddetti problemi.

F.1 Analisi delle feature fornite

Per ricostruire la funzione di costo si utilizzano 16 feature, indicate con il vettore $\mathbf{h}(\mathbf{x}_k) = [h_1(\mathbf{x}_k), \dots, h_{16}(\mathbf{x}_k)]^T$. Sia $\mathbf{x}_k = [x_k \ y_k]^T \in \mathbb{R}^2$ il prossimo stato del robot e $\mathbf{x}_d = [x_d \ y_d]^T \in \mathbb{R}^2$ l'obiettivo desiderato dal robot, l'espressione analitica delle feature è la seguente:

$$h_i(\mathbf{x}_k) = \begin{cases} |\mathbf{x}_k - \mathbf{x}_d|_2^2 & \text{for } i = 1 \\ \frac{1}{\sqrt{(2\pi)^2 \det(\Sigma_F)}} \exp\left(-\frac{1}{2}(\mathbf{x}_k - \mathbf{m}_i)^T \Sigma_F^{-1} (\mathbf{x}_k - \mathbf{m}_i)\right) & \text{for } i = 2, \dots, 16 \end{cases}$$

dove \mathbf{m}_i è il vettore della media di una gaussiana che rappresenta la posizione della feature i -esima (nel seguito chiamata anche **feature point**) e $\Sigma_F = \begin{bmatrix} 0.025 & 0 \\ 0 & 0.025 \end{bmatrix}$ è la matrice di covarianza. I vettori della media sono calcolati come segue:

$$\mathbf{m}_i = \begin{bmatrix} -0.8 + 0.8 \lfloor (i-2) \bmod 3 \rfloor \\ 0.8 - 0.4 \lfloor \frac{(i-2)}{3} \rfloor \end{bmatrix} \quad \text{with } i = 2, \dots, 16$$

Date le feature di cui sopra, è facile dedurre lo scopo di ciascuna di esse:

- h_1 è una funzione quadratica centrata sulla posizione del goal point. Questa feature viene utilizzata per ricostruire il task di raggiungere la posizione dell'obiettivo;
- Le feature h_2, \dots, h_{16} sono funzioni gaussiane centrate sulle posizioni dei feature point \mathbf{m}_i . Le feature sono usate per ricostruire il task di evitare gli ostacoli. Le posizioni dei feature point sono scelti in modo da coprire l'intero ambiente con una griglia di 5×3 ;
- Dall'analisi si deduce che mentre la posizione degli ostacoli è sconosciuta, si presume di conoscere la posizione del goal point.

Nelle Figg 18 sono fornite alcune rappresentazioni delle features fornite. In particolare in Fig. 18b le feature per goal point e ostacoli sono rappresentate su due heatmap separate, ponendo tutti i pesi della combinazione lineare pari a 1.

La forma funzionale delle feature è ragionevole, infatti è la stessa dei termini corrispondenti nella funzione di costo originale (ma non dovremmo esserne consapevoli quando risolviamo il problema di controllo inverso). Tuttavia, l'insieme di feature proposte presenta alcuni problemi:

- Non c'è nessuna feature per modellare i bordi dell'ambiente né si suppone di conoscere la loro posizione nella funzione di costo ricostruita. Questo potrebbe far collidere i robot con le pareti quando si utilizza la funzione di costo ricostruita;
- Le feature sono troppo sparse. Questo potrebbe portare a una cattiva ricostruzione della funzione di costo, soprattutto quando gli ostacoli si trovano tra due feature point;
- La feature utilizzata per modellare il termine goal point ha la stessa forma funzionale del termine corrispondente nella funzione di costo originale e questa funzione ha dimostrato di causare traiettorie oscillanti in prossimità del goal point, quindi possiamo aspettarci gli stessi problemi nella funzione di costo ricostruita.

F.2 Risultati sperimentali ottenuti con le feature fornite

Per evidenziare le problematiche delle feature fornite faremo uso di due esperimenti:

Esperimento A Nelle Figg. 19 si mostrano i risultati della ricostruzione della funzione di costo fornita nello scenario di base. La ricostruzione è stata fatta utilizzando le feature fornite e usando come osservazioni quelle in Fig. 11. In Fig. 19a vi sono i pesi associati alle feature ottenuti dal problema di ottimizzazione. Nelle Figg. 19b e 19c è rappresentata la funzione di costo ricostruita sottoforma di heatmap e grafico 3D. Nelle Figg. 21 e 23 sono rappresentati gli esiti di due simulazioni realizzata con la suddetta funzione di costo ricostruita.

Esperimento B Nelle Figg. 20 si mostrano i risultati della ricostruzione della funzione di costo fornita in uno scenario più complesso. La ricostruzione è stata fatta utilizzando le feature fornite e usando come osservazioni quelle in Fig. 20e. In Fig. 20d vi sono i pesi associati alle feature ottenuti dal problema di ottimizzazione. Nelle Figg. 20a e 20b è rappresentata la funzione di costo ricostruita sottoforma di heatmap e grafico 3D. In Fig. 22 è rappresentato l'esito di una simulazione realizzata con la suddetta funzione di costo ricostruita.

Come si evince da questi esperimenti, le feature fornite presentano diverse problematiche di seguito riportate:

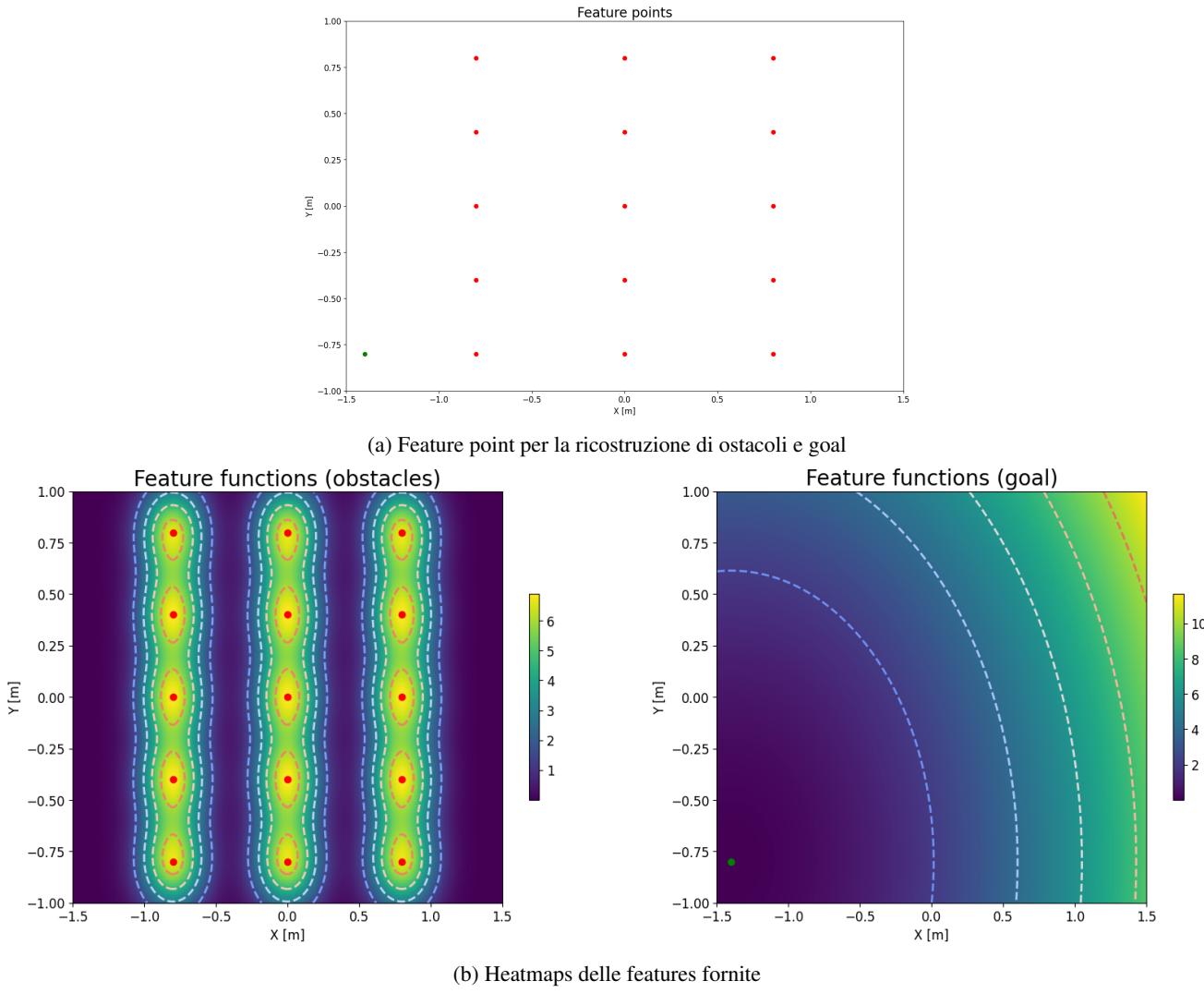


Figura 18: Rappresentazioni delle features fornite

Problema 1 Numero di feature point insufficiente per ricostruire alcuni ostacoli

In alcuni scenari più complicati, in cui gli ostacoli non sono allineati esattamente in corrispondenza di punti della griglia di feature (feature point), la funzione di costo non viene ricostruita adeguatamente. Un esempio di tale problema è mostrato in Fig. 22, dove un ostacolo è esattamente posizionato tra due feature point e non viene per nulla ricostruito utilizzando le feature fornite. Una possibile soluzione è aggiungere nuove feature per tentare di ricostruire meglio gli ostacoli presenti nell’ambiente del Robotarium.

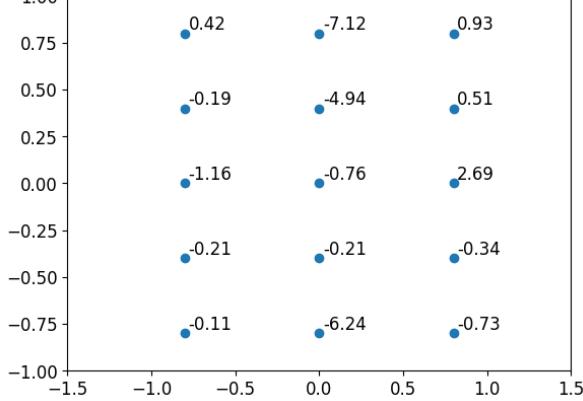
Problema 2 Barriere non ricostruite

Un altro problema individuato è che non sono presenti feature che permettono la ricostruzione delle barriere. Un esempio significativo è la simulazione mostrata in Fig. 23, dove i robot escono dai limiti del Robotarium. Una possibile soluzione è aggiungere delle feature che hanno il solo scopo di ricostruire le barriere.

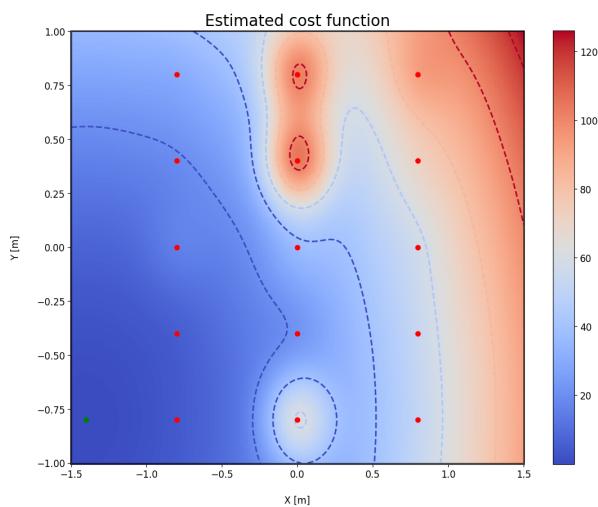
Problema 3 Oscillazioni in prossimità del goal point

Dalle simulazioni è emerso che i robot in prossimità del goal point accentuano il loro comportamento entropico, come evidenziato nella Fig. 21. La ragione di tale comportamento è dovuto al fatto che il goal point, nel costo ricostruito, è molto meno "attrattivo" del costo fornito nel caso FOC. Ciò è dovuto al valore del peso assegnato alla feature del goal point dopo la risoluzione del problema di ottimizzazione inverso. Infatti, il modulo del peso assegnato alla feature che modella il goal point vale $|w_1| = 11.7$ mentre nella funzione di costo usata nel controllo forward, il termine che modella il goal point ha un coefficiente pari a 30.

Una soluzione a questo problema può essere quella di modificare la forma della feature usata per modellare il goal point.

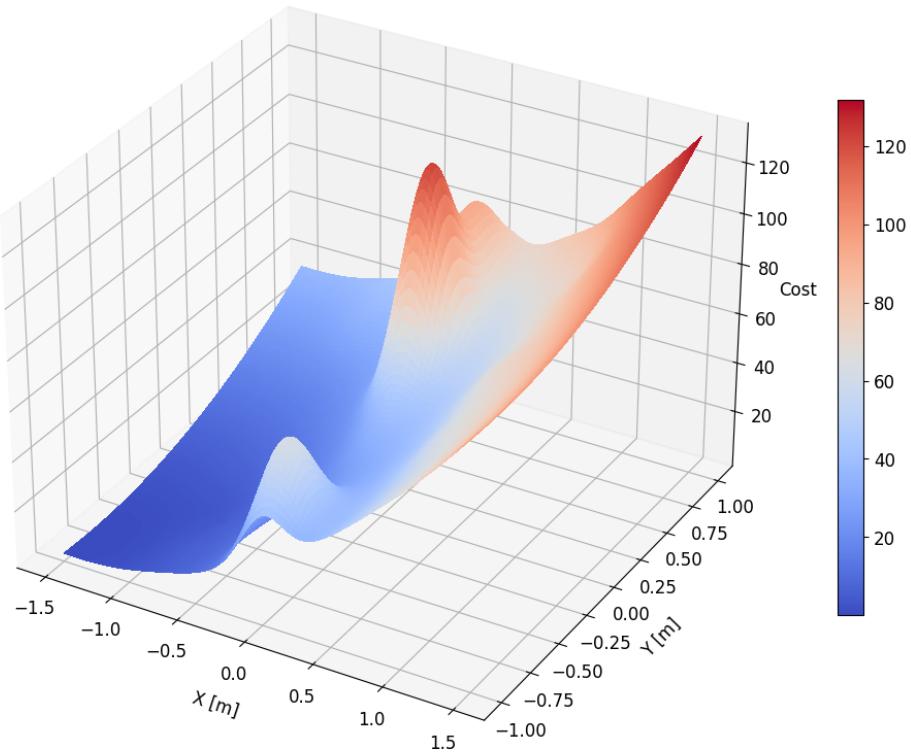


(a) Pesi ottenuti dal problema di ottimizzazione



(b) Heat Map del costo stimato

Estimated Cost Function



(c) Plot 3D del costo stimato

Figura 19: Risultati dell'esperimento A (scenario base)

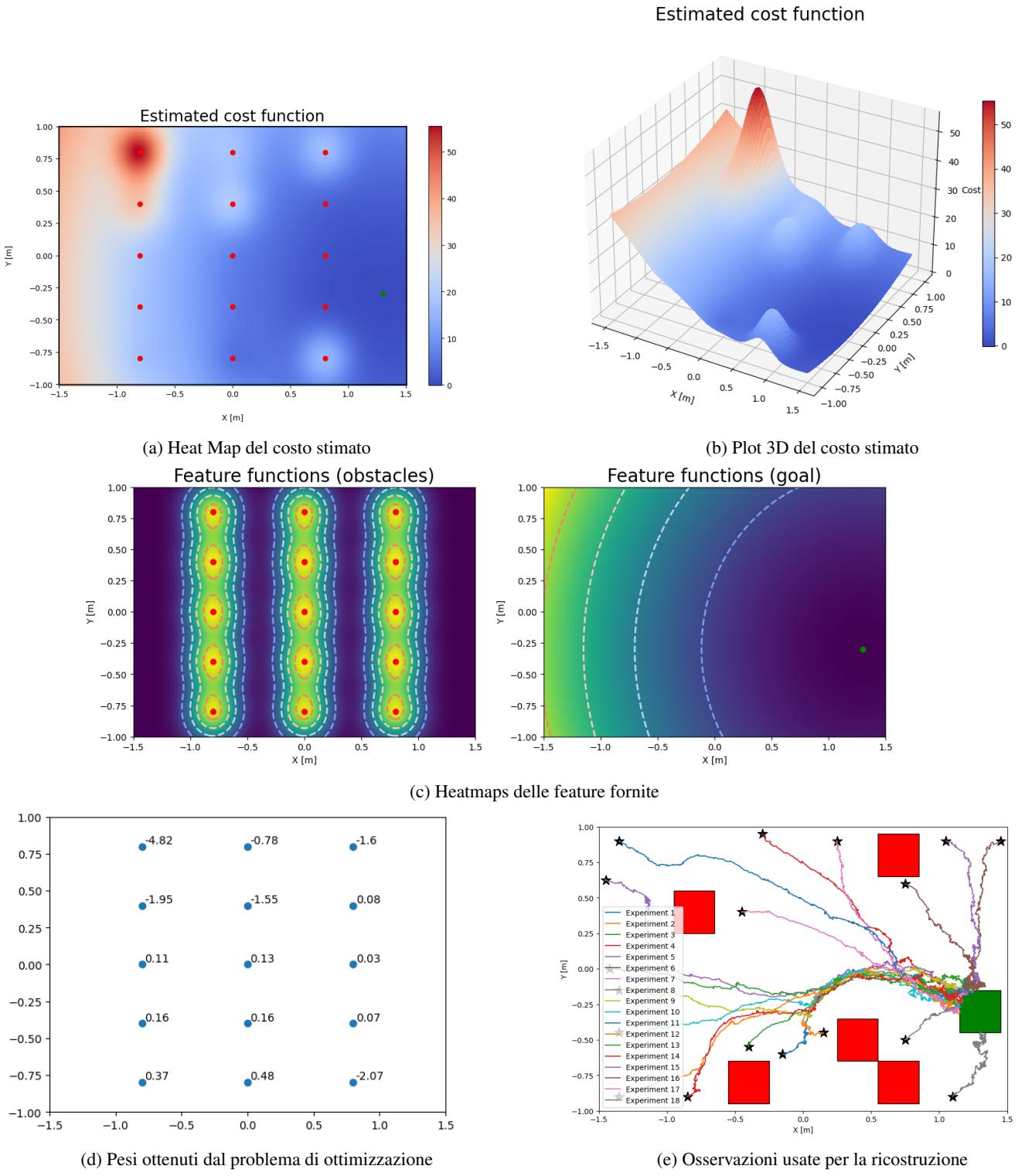


Figura 20: Risultati dell'esperimento B (scenario complesso)

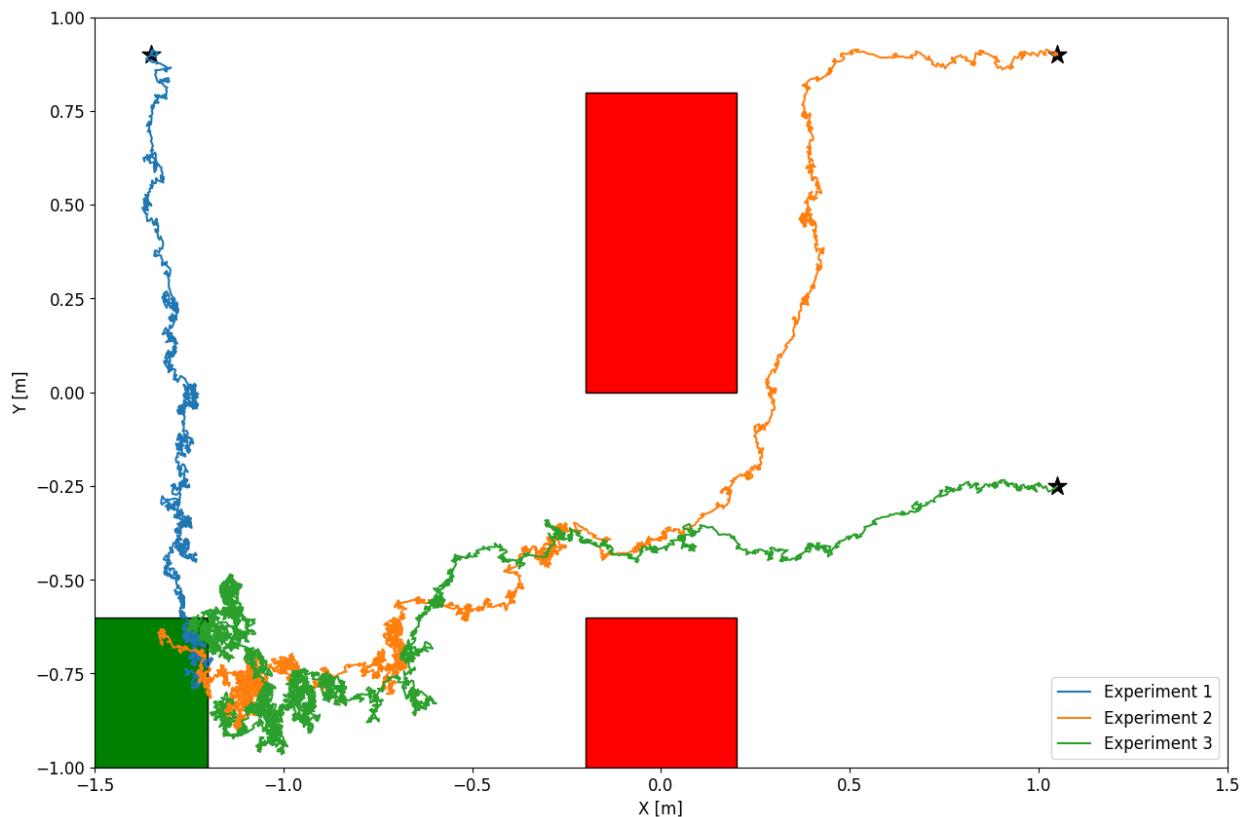


Figura 21: Simulazione col costo stimato dall'esperimento A

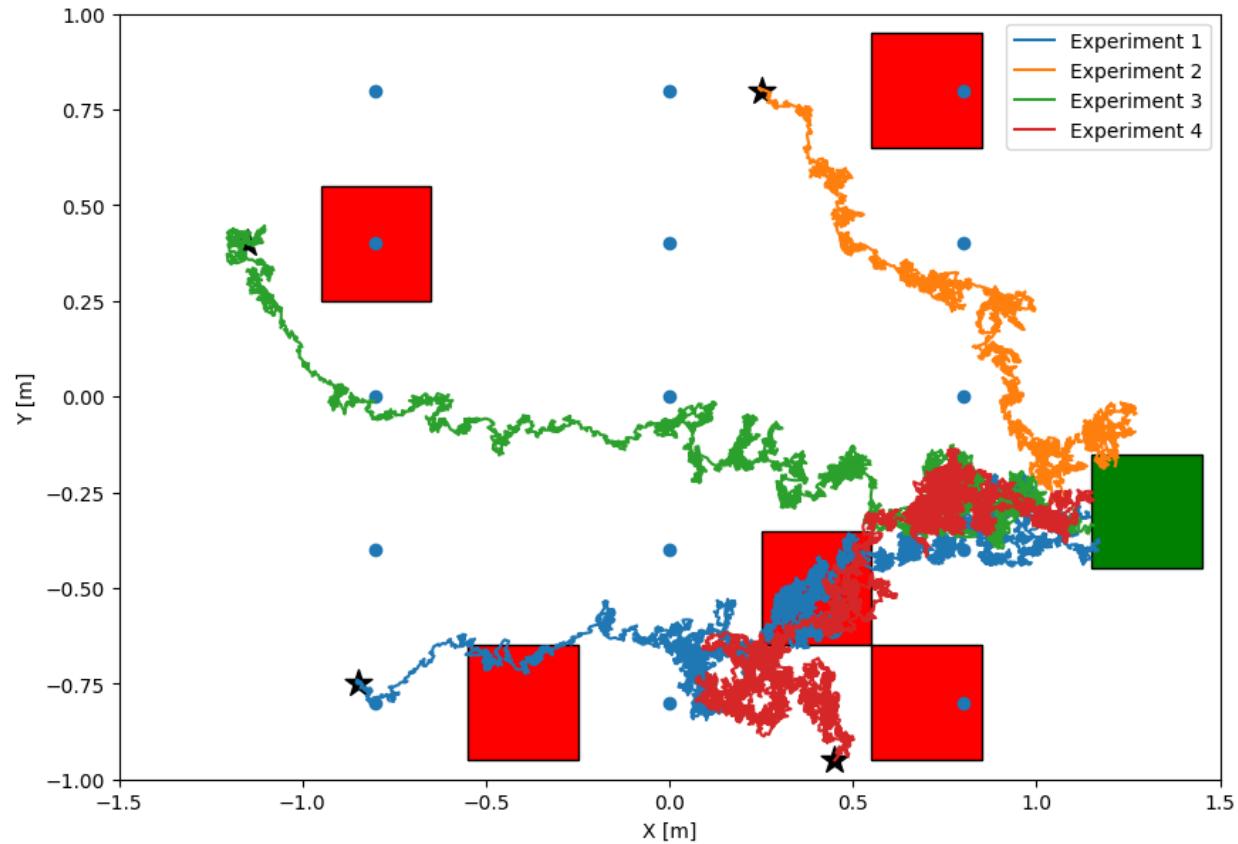


Figura 22: Simulazione col costo stimato dall'esperimento B

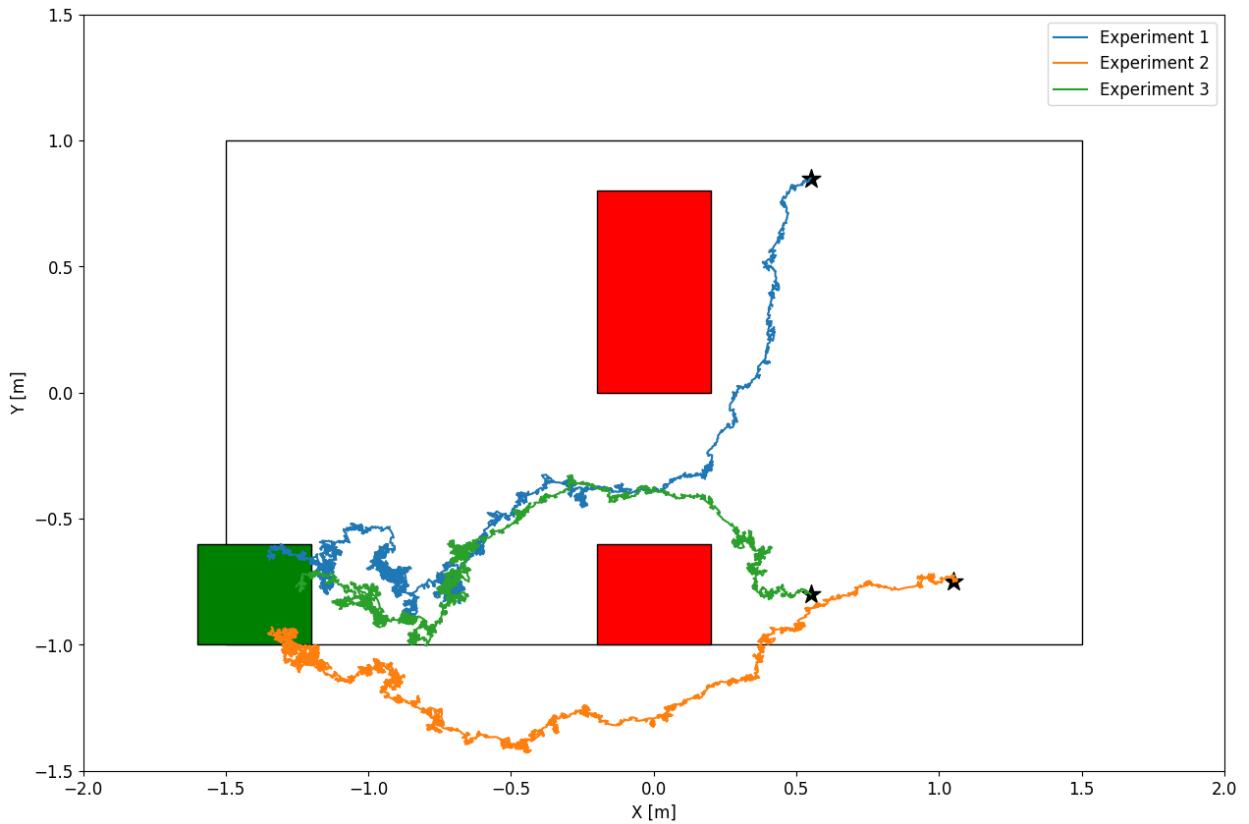


Figura 23: Simulazione col costo stimato dall'esperimento A (condizioni iniziali differenti)

F.3 Proposta di un nuovo insieme di feature

Nel Paragrafo F.2 sono state analizzate le problematiche del set di feature fornito. Sulla base delle considerazioni fatte si propone un nuovo set di feature, che tenta di risolvere i problemi individuati.

Sia $\mathbf{x}_k = [x_k \ y_k]^T \in \mathbb{R}^2$ il prossimo stato del robot e $\mathbf{x}_d = [x_d \ y_d]^T \in \mathbb{R}^2$ l'obiettivo desiderato dal robot. Si propone un nuovo vettore di feature $\mathbf{h}(\mathbf{x}_k) = [h_1(\mathbf{x}_k), \dots, h_{47}(\mathbf{x}_k)]^T$ dove:

$$h_i(\mathbf{x}_k) = \begin{cases} |\mathbf{x}_k - \mathbf{x}_d|_2^2 & \text{for } i = 1 \\ \frac{1}{|\mathbf{x}_k - \mathbf{x}_d|_2 + \epsilon} & \text{for } i = 2 \\ \frac{1}{\sqrt{(2\pi)^2 \det(\Sigma_F)}} \exp\left(-\frac{1}{2}(\mathbf{x}_k - \mathbf{m}_i)^T \Sigma_F^{-1} (\mathbf{x}_k - \mathbf{m}_i)\right) & \text{for } i = 3, \dots, 37 \\ \frac{1}{(0.05)\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x_k - \alpha_i}{(0.05)}\right)^2\right) & \text{for } i = 38, \dots, 42 \\ \frac{1}{(0.05)\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y_k - \beta_i}{(0.05)}\right)^2\right) & \text{for } i = 43, \dots, 47 \end{cases}$$

In particolare:

- h_3, \dots, h_{37} sono gaussiane multivariate con media $\mathbf{m}_i \in [-1.5, 1.5] \times [-1, 1] \subset \mathbb{R}^2$ e matrice di covarianza $\Sigma_F = \begin{bmatrix} 0.02 & 0 \\ 0 & 0.02 \end{bmatrix}$;
- h_{38}, \dots, h_{42} sono gaussiane univariate calcolate rispetto all'asse delle ascisse con media $\alpha_i \in [-1.5, 1.5] \subset \mathbb{R}$ e varianza $\sigma^2 = 0.05$;
- h_{43}, \dots, h_{47} sono gaussiane univariate calcolate rispetto all'asse y con media $\beta_i \in [-1, 1] \subset \mathbb{R}$ e varianza $\sigma^2 = 0.05$

Le medie di tutte le gaussiane di cui sopra sono le seguenti:

$$\begin{aligned} \mathbf{m}_i &= \begin{bmatrix} -1.2 + 0.4 \cdot [(i-3) \bmod 7] \\ 0.8 - 0.4 \cdot \lfloor \frac{(i-3)}{7} \rfloor \end{bmatrix} & \text{con } i = 3, \dots, 37 \\ \alpha_i &= -1.5 + 0.75 \cdot (i-38) & \text{con } i = 38, \dots, 42 \\ \beta_i &= -1 + 0.5 \cdot (i-43) & \text{con } i = 43, \dots, 47 \end{aligned}$$

Le feature proposte hanno i seguenti scopi:

- La feature h_1 è uguale a quella dell'insieme di feature fornito e viene utilizzata per ricostruire il termine del goal point del costo;
- La feature h_2 è la stessa utilizzata nella funzione di costo proposta. Si tratta di un attrattore più forte che dovrebbe ridurre il comportamento entropico dei robot in prossimità del goal point;
- Le feature h_3, \dots, h_{37} sono utilizzate per ricostruire gli ostacoli. Per mantenere basso il costo computazionale del problema IOC, si è cercato di mantenere il numero di feature più basso possibile necessario a ricostruire adeguatamente la funzione di costo. Considerando le dimensioni degli ostacoli, si è trovato che una griglia di 7×5 feature point è sufficientemente densa per ricostruire gli ostacoli che non sono perfettamente allineati con la griglia. Dato che gli ostacoli sono più vicini l'uno all'altro, si è anche deciso di ridurre la varianza delle funzioni gaussiane che modellano gli ostacoli, da 0.025 a 0.02;
- Le feature h_{38}, \dots, h_{47} sono utilizzate per ricostruire i confini del Robotarium. Si è ipotizzato di non conoscere la posizione effettiva delle barriere e quindi di disporre le feature "orizzontali" e "verticali" su tutto l'ambiente del Robotarium al fine di individuarne la posizione. Si osservi, inoltre, che queste feature sono dello stesso tipo di quelle utilizzate per modellare le barriere nella funzione di costo proposta.

F.4 Risultati sperimentali ottenuti con le feature fornite

Per evidenziare i risultati ottenuti con le feature proposte faremo uso di tre esperimenti:

Esperimento C Nelle Figg. 24 si mostrano i risultati della ricostruzione della funzione di costo fornita nello scenario più complesso (lo stesso della sezione precedente). La ricostruzione è stata fatta utilizzando le feature proposte e usando come osservazioni quelle in Fig. 20e. In Fig. 24e vi sono i pesi associati alle feature ottenuti dal problema di ottimizzazione. Nelle Figg. 24a e 24b è rappresentata la funzione di costo ricostruita sottoforma di heatmap e grafico 3D. Nella Fig. 25 è rappresentato l'esito di una simulazione realizzata con la suddetta funzione di costo ricostruita.

Esperimento D Nelle Figg. 27 si mostrano i risultati della ricostruzione della funzione di costo fornita nello scenario di base. Questo permette di confrontare direttamente i risultati con quelli ottenuti con l'insieme di feature originali. La ricostruzione è stata fatta utilizzando le feature proposte e usando come osservazioni quelle in Fig. 26. In Fig. 27e vi sono i pesi associati alle feature ottenuti dal problema di ottimizzazione. Nelle Figg. 27a e 27b è rappresentata la funzione di costo ricostruita sottoforma di heatmap e grafico 3D. In Fig. 28 è rappresentato l'esito di una simulazione realizzata con la suddetta funzione di costo ricostruita. In Fig. 29 è rappresentato il gradiente della funzione di costo ricostruita.

Esperimento E Nelle Figg. 31 si mostrano i risultati della ricostruzione della funzione di costo proposta nello scenario di base. La ricostruzione è stata fatta utilizzando le feature proposte e usando come osservazioni quelle in Fig. 30. In Fig. 31e vi sono i pesi associati alle feature ottenuti dal problema di ottimizzazione. Nelle Figg. 31a e 31b è rappresentata la funzione di costo ricostruita sottoforma di heatmap e grafico 3D. In Fig. 32 è rappresentato l'esito di una simulazione realizzata con la suddetta funzione di costo ricostruita.

Come si nota in Fig. 24b, nell'Esperimento C con l'insieme di feature proposte si è ottenuta ottenuta una migliore ricostruzione della funzione di costo nello scenario problematico che è stato presentato nella sezione precedente. In particolare:

- gli ostacoli che con il set di feature fornito non erano stati ricostruiti ora vengono correttamente ricostruiti;
- i robot in prossimità del goal point tentano di raggiungerlo manifestando un comportamento meno entropico;
- le barriere dell'ambiente del Robotarium adesso vengono parzialmente ricostruite.

Tuttavia in tale scenario la ricostruzione del goal point. Infatti, nella simulazione in Fig. 25 si può osservare che i robot manifestano un comportamento entropico più accentuato rispetto a quello osservato utilizzando direttamente la funzione di costo fornita nel problema di forward control. In ogni caso, le prestazioni ottenute con il set di feature proposto sono comunque migliori di quelle ottenute utilizzando l'insieme di feature fornito per ricostruire la stessa funzione di costo (vedasi la Fig. 22). Il problema delle traiettorie oscillanti è probabilmente dovuto al fatto che le osservazioni utilizzate per ricostruire la funzione di costo provengono dalla funzione fornita in cui le traiettorie dei robot sono molto oscillanti vicino al goal point. Per confermare tale ipotesi è stato realizzato l'Esperimento E in cui si è tentato di ricostruire la funzione di costo proposta (che risolve il problema di cui sopra) con l'insieme di feature proposte e, come si vede in Fig. 32, alimentando il problema IOC con osservazioni migliori anche il problema delle traiettorie oscillanti è risolto.

Un altro problema dell'Esperimento C è che i confini del Robotarium non sono stati completamente ricostruiti. Tale difficoltà è maggiormente evidente nell'Esperimento D, dove dal grafico del gradiente del costo ricostruito in Fig. 29 si evince che solo alcune delle barriere sono state correttamente ricostruite. Nella maggior parte delle simulazioni si è riusciti a ricostruire al più tre delle quattro barriere. Si è osservato che la ricostruzione dei limiti è intrinsecamente difficile perché è difficile fare in modo che nelle osservazioni i robot "interagiscano" in modo evidente con le barriere.

La simulazione in Fig. 28 conferma le osservazioni fatte precedentemente: le performance ottenute col costo ricostruito tramite le feature proposte sono migliori di quelle ottenute con l'insieme di feature originali (Fig. 22), ma sono comunque peggiori di quelle ottenute con la funzione di costo proposta nel problema di forward control (Fig. 15a).

In conclusione, in Fig.31b si può apprezzare come la ricostruzione della funzione di costo proposta con le feature proposte sia migliore rispetto ai casi precedenti. Oltre ad avere minori oscillazioni nei pressi del goal point (come fatto notare sopra, Fig. 32) anche la ricostruzione delle barriere è più evidente. In particolare i bordi inferiore e laterale sinistro sono ricostruiti meglio rispetto alle simulazioni precedenti. Ciò è dovuto al fatto che la funzione di costo proposta genera traiettorie migliori rispetto a quella originale, il che ne facilita la ricostruzione.

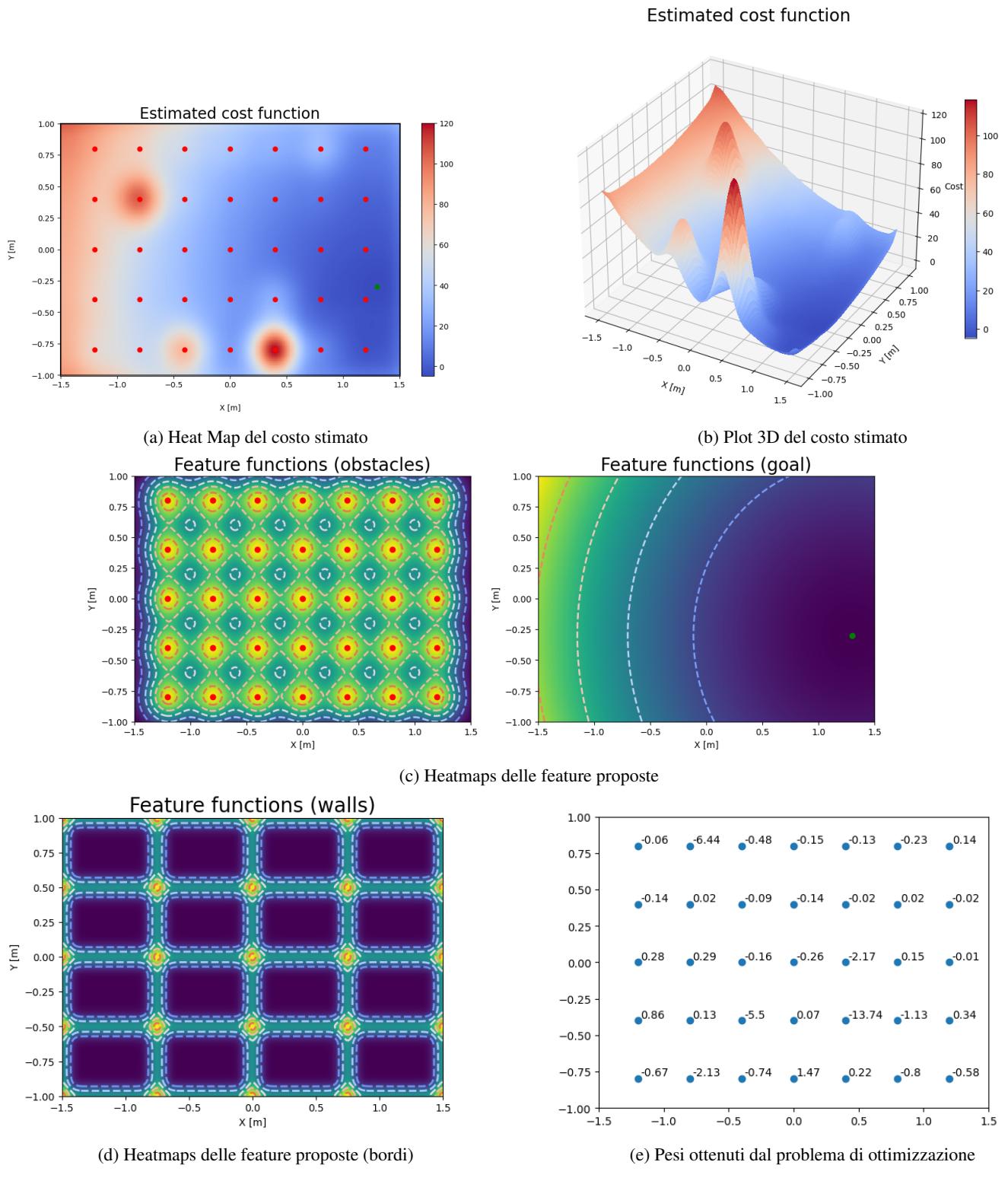


Figura 24: Risultati dell'esperimento C (scenario complesso)

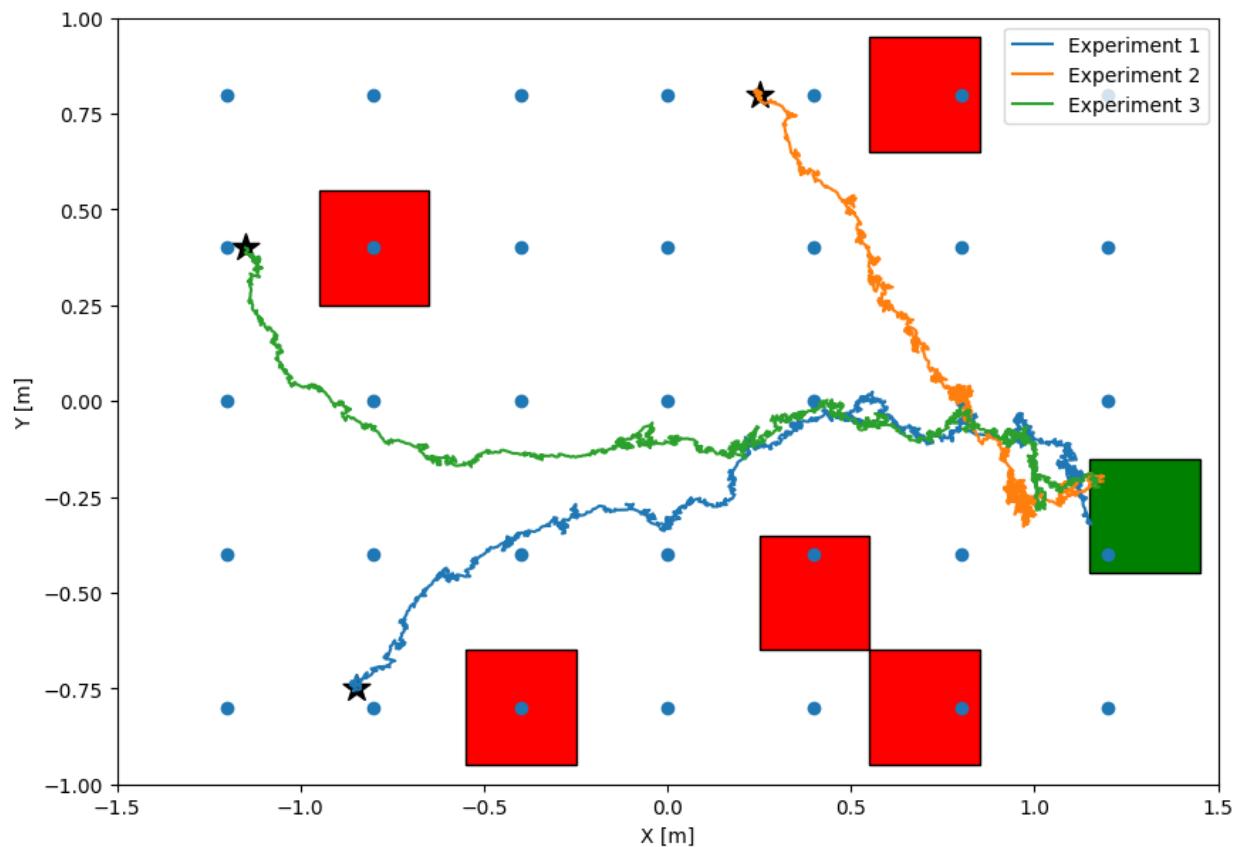


Figura 25: Simulazione col costo ricostruito nell’Esperimento C

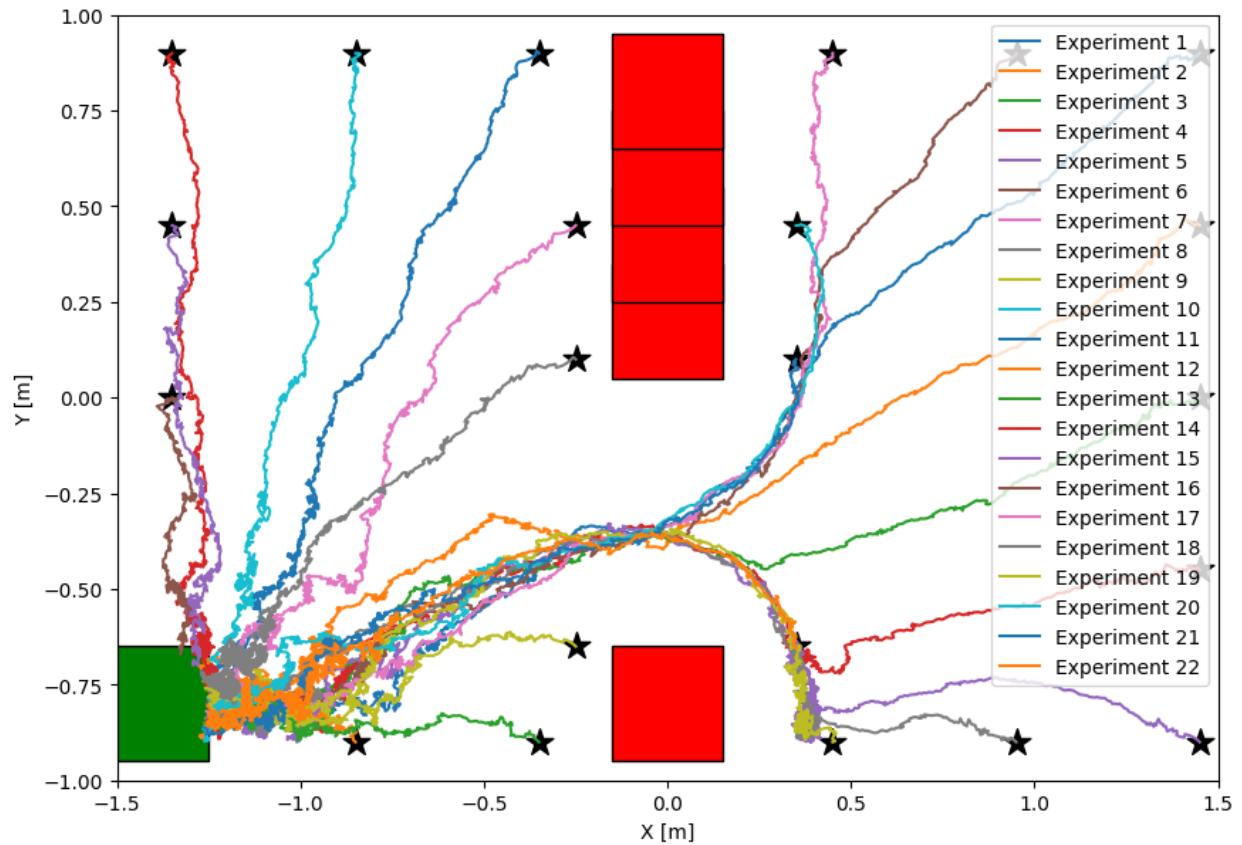


Figura 26: Osservazioni utilizzate per l’Esperimento D

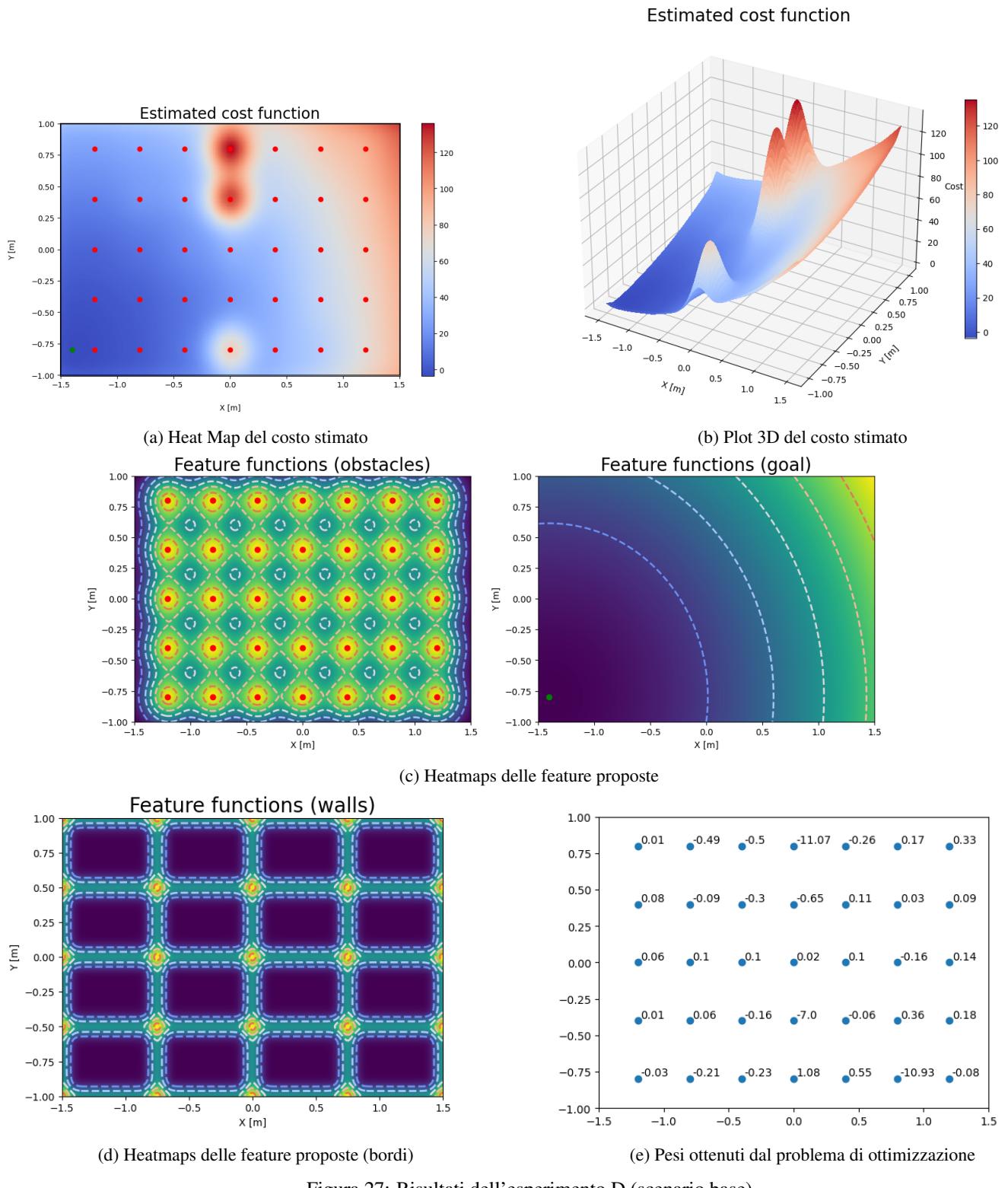


Figura 27: Risultati dell'esperimento D (scenario base)

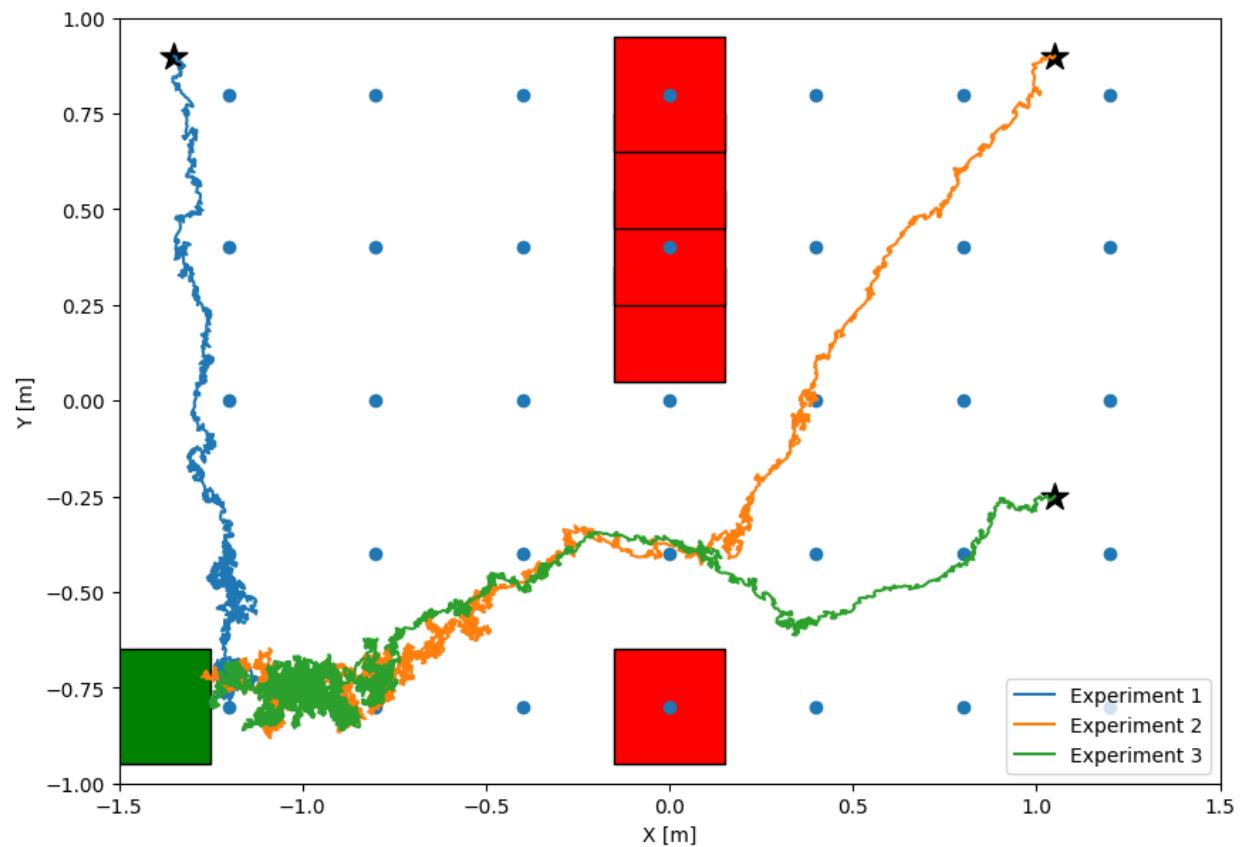


Figura 28: Simulazione col costo ricostruito nell'esperimento D

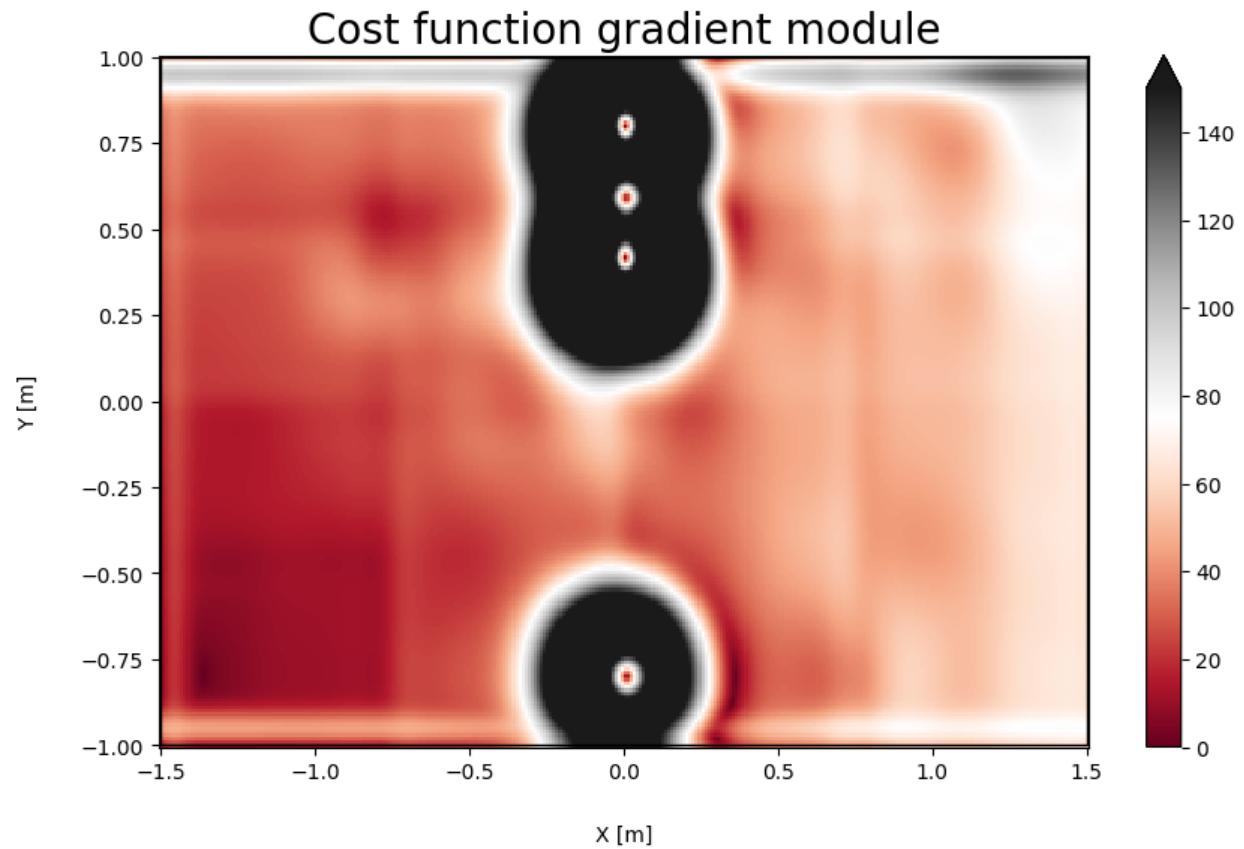


Figura 29: Gradiente costo ricostruito nell'esperimento D

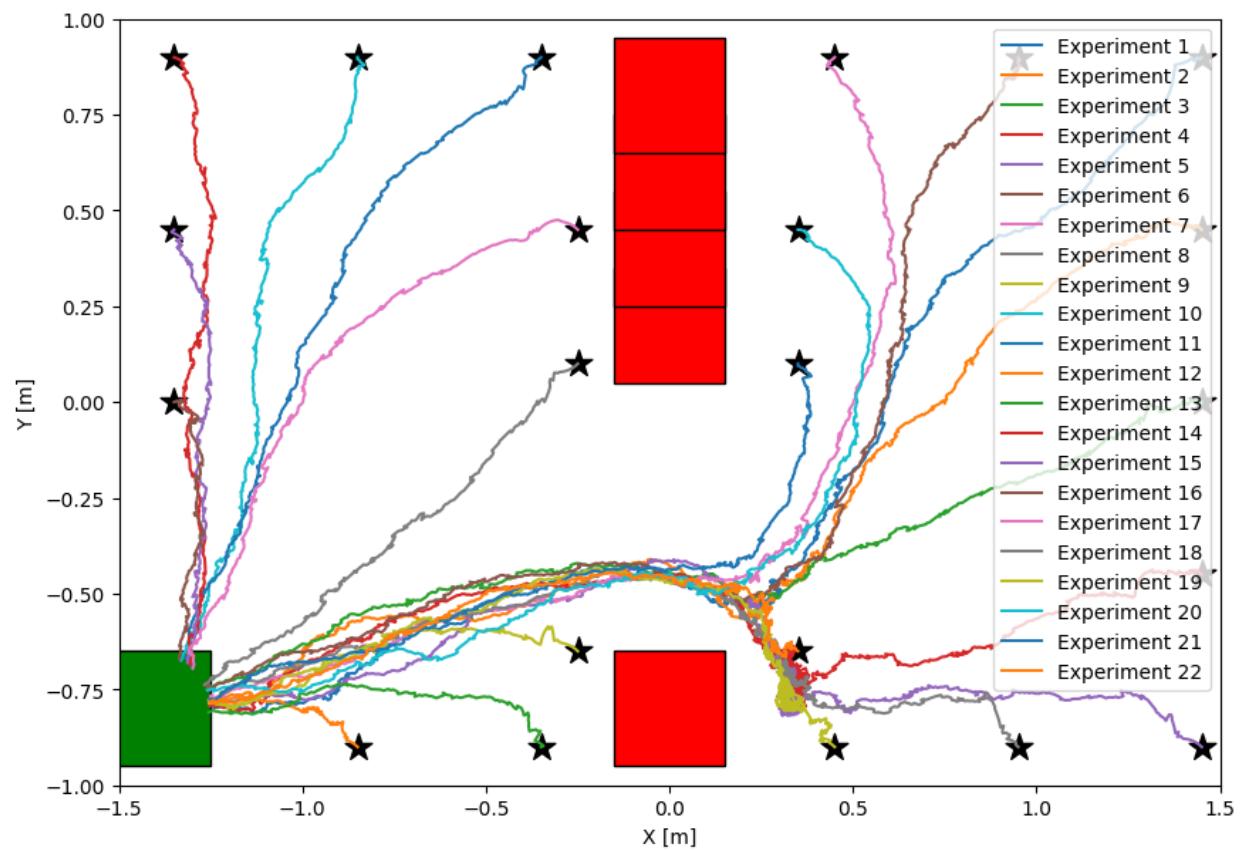


Figura 30: Osservazioni per l'esperimento E

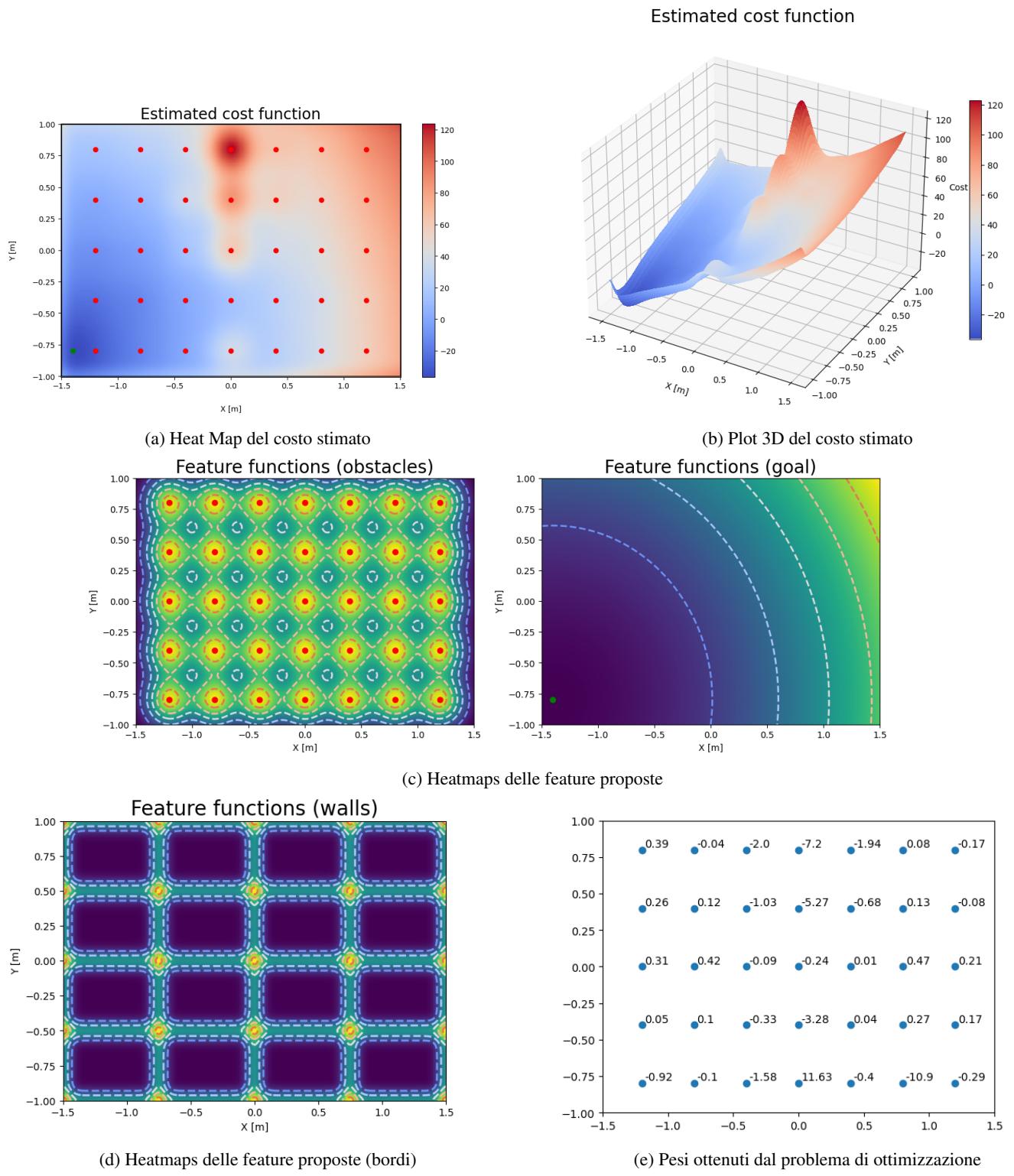


Figura 31: Risultati dell’Esperimento E

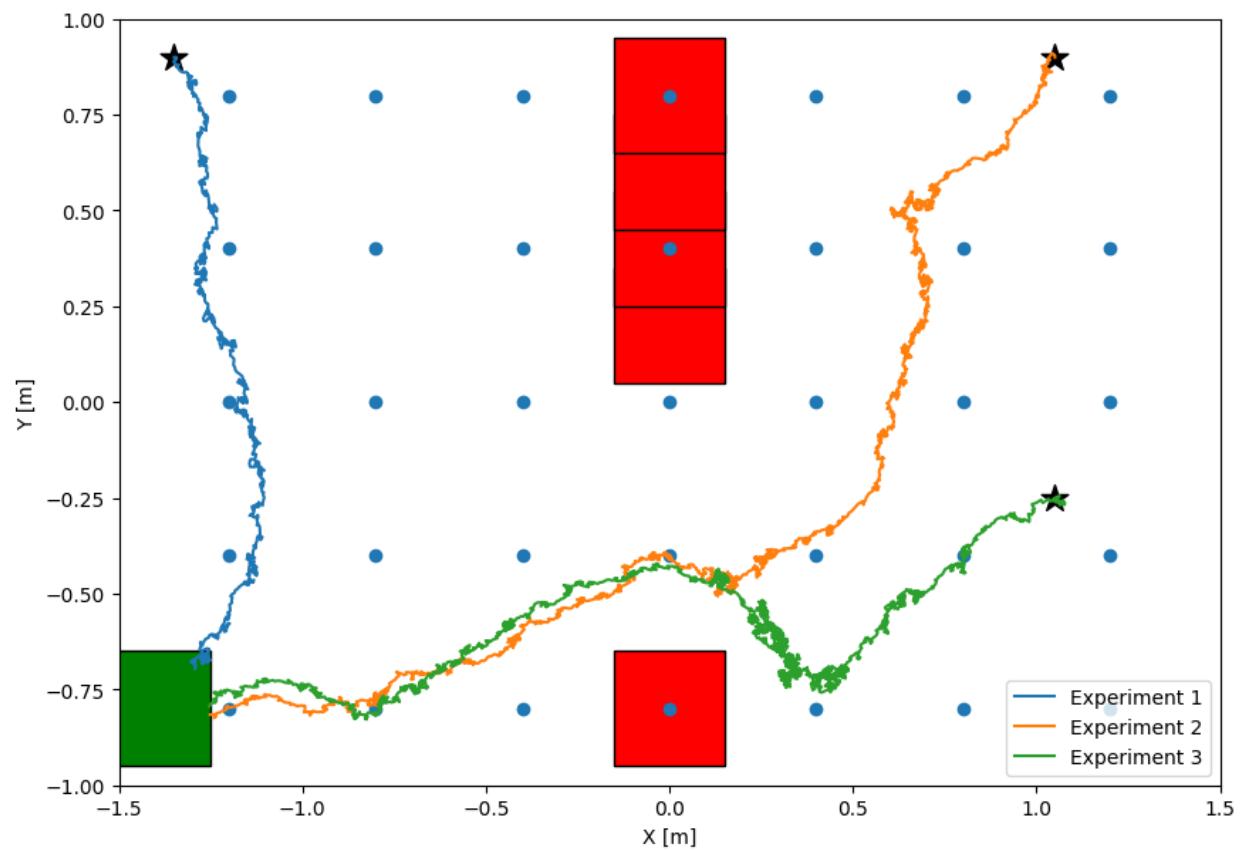


Figura 32: Simulazione con la funzione di costo ricostituita dall'Esperimento E

Riferimenti bibliografici

- [1] Emanuel Todorov. Efficient computation of optimal actions. *National Library for Medicine*, 2009.
- [2] Mariusz Janiak, Krzysztof Tchoń. Constrained robot motion planning: Imbalanced jacobian algorithm vs. optimal control approach, 2010. [Online; accessed December 27, 2023].
- [3] SEAN WILSON, PAUL GLOTFELTER, LI WANG, SIDDHARTH MAYYA, GENNARO NOTOMISTA, MARK MOTE and MAGNUS EGERSTEDT. The robotarium, 2020. [Online; accessed December 27, 2023].
- [4] Steven Diamond , Akshay Agrawal , Riley Murray , Philipp Schiele e Bartolomeo Stellato. The robotarium, 2023. [Online; accessed December 27, 2023].