

# Documentazione - Progetto di Robotica

Augello Andrea   Castiglione Francesco Paolo   La Martina Marco

Università degli studi di Palermo

11 gennaio 2021

- 1** Introduzione
- 2** TIAGo Iron
- 3** Gestione dei nodi ROS
- 4** Modello del moto e posizionamento
- 5** Object recognition
- 6** Posizione dei target
- 7** Modello probabilistico
- 8** Scheduling dei comportamenti
- 9** Possibili modifiche
- 10** Conclusioni

# Section 1

## Introduzione

La pandemia del coronavirus SARS-CoV-2 ha dato una forte spinta alla ricerca sia nel campo sanitario che informatico, mettendo in evidenza forti carenze dal punto di vista infrastrutturale.

Al momento, considerando la limitata disponibilità del vaccino alle masse, uno dei migliori modi di evitare la contrazione del coronavirus è di evitarne l'esposizione. Il distanziamento sociale si configura di conseguenza come un prerequisito per una significativa riduzione del numero di infetti, come evidenziato da simulazioni di un sistema ad agenti [?]. Un problema chiave risulta essere il controllo del rispetto delle norme di distanziamento all'interno degli spazi chiusi.

## Obbiettivo

L'obbiettivo del progetto è quello di sviluppare un robot con lo scopo di **evitare assembramenti in ambienti indoor** e di invitare a **rispettare le norme sul distanziamento sociale**.

Nella dimostrazione presentata il nostro robot rileva le persone nella stanza e individua i possibili assembramenti. In seguito alla fase di rilevazione si sposterà verso l'assembramento evitando gli ostacoli e, arrivato, esorterà le persone al rispetto del distanziamento sociale.

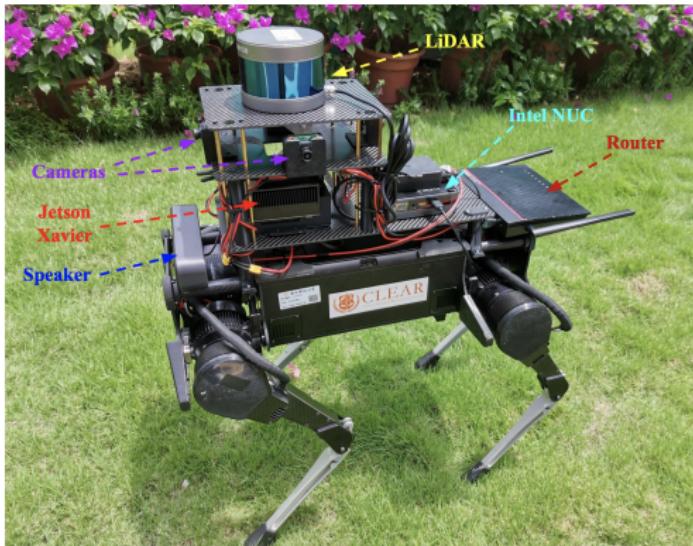
## Stato dell'arte I

Un TurtleBot 2, con una camera RGB-D e CCTV per il rilevamento degli assembramenti, una camera termica FLIR C3 per rilevare la temperatura corporea e un lidar 2-D per evitare le collisioni. L'elaborazione delle immagini provenienti da CCTV è eseguita in un laptop con una CPU Intel i7 7th generation e una GPU Nvidia GTX1060, mentre il resto dell'elaborazione è eseguita su una CPU Intel i9 8th generation e una GPU Nvidia RTX2080 montate sul robot.



## Stato dell'arte II

Un Laikago con 2 camere laterali e CCTV per il rilevamento degli assemblamenti e un lidar 3-D per evitare le collisioni. L'elaborazione riguardante il modulo visivo è effettuata in una NVIDIA Jetson AGX Xavier, il resto è eseguito in una CPU Intel i5 8259U. Il rilevamento delle persone viene effettuato tramite la rete YOLO.



# Setup

---

<b>OS</b>	Ubuntu 18.04 Ubuntu 20.04 Raspberry Pi OS Buster
<b>ROS version</b>	melodic noetic
<b>Webots</b>	R2020b revision 1
<b>OpenCV [?]</b>	4.x
<b>Imutils [?]</b>	0.5.3
<b>Matplotlib [?]</b>	3.3.3
<b>Numpy [?]</b>	1.17.2
<b>Scikit-learn [?]</b>	0.21.3
<b>Raspberry</b>	Raspberry Pi 3B+

---

1 Introduzione

2 TIAGo Iron

3 Gestione dei nodi ROS

4 Modello del moto e posizionamento

5 Object recognition

6 Posizione dei target

7 Modello probabilistico

8 Scheduling dei comportamenti

9 Possibili modifiche

10 Conclusioni

## Section 2

TIAGo Iron

Il robot scelto per l'obiettivo proposto è il **TIAGo Iron**, un robot umanoide a due ruote con torso e testa ma senza braccia articolate [?].

Il datasheet del **TIAGo** [?] indica la presenza di speaker e display, non presenti nel modello Webots [?], che sono quindi stati aggiunti.

La camera del **TIAGo** è RGB-D ma il modello Webots ne è sprovvisto, di conseguenza è stata utilizzata una camera monoscopica RGB.

L'IMU utilizzata ha 6 gradi di libertà.

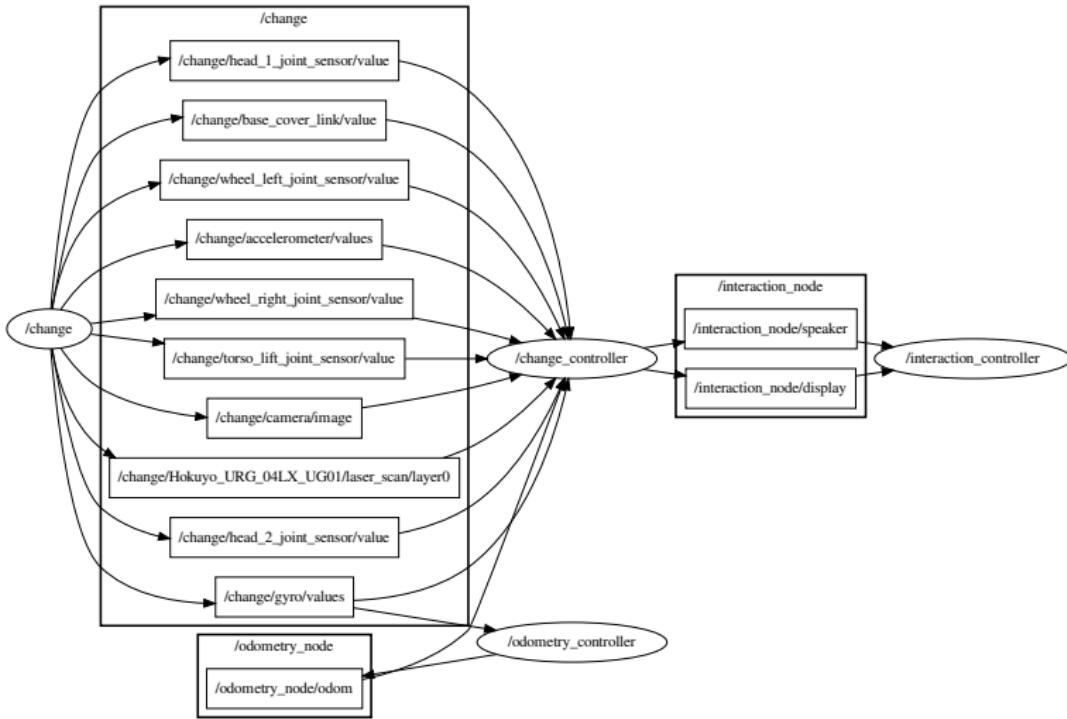
Il modello Webots del **TIAGo** presenta un lidar (Hokuyo URG-04LX-UG01 [?]) che, conformemente al datasheet, ha un range di 5.6 m ed un FOV di 240° (agli estremi è parzialmente occluso).



- 1** Introduzione
- 2** TIAGo Iron
- 3** Gestione dei nodi ROS
- 4** Modello del moto e posizionamento
- 5** Object recognition
- 6** Posizione dei target
- 7** Modello probabilistico
- 8** Scheduling dei comportamenti
- 9** Possibili modifiche
- 10** Conclusioni

## Section 3

### Gestione dei nodi ROS



**Figura:** Architettura dei nodi ROS, ottenuta tramite *rqt*

## Webots node

Questo nodo si occupa solamente di lanciare Webots, e di impostare il valore del clock di ROS in base al tempo della simulazione, in modo da potere effettuare le integrazioni del tempo correttamente.

## Change controller node

Questo è il nodo che si occupa della gran parte della elaborazione. Oltre ad arbitrare sui comportamenti da assumere, gestisce più moduli che si occupano di:

## Change controller node

Questo è il nodo che si occupa della gran parte della elaborazione. Oltre ad arbitrare sui comportamenti da assumere, gestisce più moduli che si occupano di:

- acquisire i dati dai sensori

## Change controller node

Questo è il nodo che si occupa della gran parte della elaborazione. Oltre ad arbitrare sui comportamenti da assumere, gestisce più moduli che si occupano di:

- acquisire i dati dai sensori
- mandare i comandi ai motori

## Change controller node

Questo è il nodo che si occupa della gran parte della elaborazione. Oltre ad arbitrare sui comportamenti da assumere, gestisce più moduli che si occupano di:

- acquisire i dati dai sensori
- mandare i comandi ai motori
- gestire il movimento, quindi rotazioni e traslazioni

## Change controller node

Questo è il nodo che si occupa della gran parte della elaborazione. Oltre ad arbitrare sui comportamenti da assumere, gestisce più moduli che si occupano di:

- acquisire i dati dai sensori
- mandare i comandi ai motori
- gestire il movimento, quindi rotazioni e traslazioni
- acquisire e analizzare le immagini dalla camera

## Interaction node

Questo nodo ha il compito di gestire le interazioni audio/video. Ogni messaggio che viene riprodotto, prima in lingua italiana e poi inglese, viene anche visualizzato testualmente sullo schermo in italiano, inglese e cinese. I possibili comportamenti assunti dal robot sono:

## Interaction node

Questo nodo ha il compito di gestire le interazioni audio/video. Ogni messaggio che viene riprodotto, prima in lingua italiana e poi inglese, viene anche visualizzato testualmente sullo schermo in italiano, inglese e cinese. I possibili comportamenti assunti dal robot sono:

- Salutare all'avvio

## Interaction node

Questo nodo ha il compito di gestire le interazioni audio/video. Ogni messaggio che viene riprodotto, prima in lingua italiana e poi inglese, viene anche visualizzato testualmente sullo schermo in italiano, inglese e cinese. I possibili comportamenti assunti dal robot sono:

- Salutare all'avvio
- Mostrare sul display immagini che esortano a rispettare il distanziamento sociale

## Interaction node

Questo nodo ha il compito di gestire le interazioni audio/video. Ogni messaggio che viene riprodotto, prima in lingua italiana e poi inglese, viene anche visualizzato testualmente sullo schermo in italiano, inglese e cinese. I possibili comportamenti assunti dal robot sono:

- Salutare all'avvio
- Mostrare sul display immagini che esortano a rispettare il distanziamento sociale
- Riprodurre un messaggio audio che invita a rispettare il distanziamento sociale quando rileva un assembramento o quando scansiona l'ambiente

## Odometry node

Il nodo che si occupa dell'odometria si occupa di stimare la posizione del robot, come spiegato approfonditamente nella sezione 4. In generale ciò che fa è integrare costantemente i valori del giroscopio e della velocità delle ruote per condividere posizione e orientamento del robot.

- 1 Introduzione
- 2 TIAGo Iron
- 3 Gestione dei nodi ROS
- 4 Modello del moto e posizionamento
- 5 Object recognition
- 6 Posizione dei target
- 7 Modello probabilistico
- 8 Scheduling dei comportamenti
- 9 Possibili modifiche
- 10 Conclusioni

## Section 4

### Modello del moto e posizionamento

## Orientamento

Il modello del moto è caratterizzato da rotazioni e traslazioni. Per le rotazioni ci basiamo sui dati forniti dal giroscopio, il quale fornisce una velocità angolare. Calcoliamo quindi l'angolo di rotazione effettuando un'integrazione discreta dei campioni con interpolazione lineare del primo ordine (Eq. 1).

$$\theta_i = \sum_{j=1}^i \frac{\omega_{j-1} + \omega_j}{2} (t_j - t_{j-1}) \quad (1)$$

Noto l'angolo corrente e l'angolo target utilizziamo un controllore proporzionale per raggiungere l'angolo desiderato.

## Spostamento

Per effettuare lo spostamento lineare utilizziamo il controllore PID (Proporzionale-Integrale-Derivativo) delle ruote fornito da Webots, che richiede un angolo di rotazione target per ogni ruota. Utilizziamo quindi l'angolo di rotazione corrente, e il diametro delle ruote per calcolare la posizione delle ruote necessaria al fine di ottenere lo spostamento desiderato (Eq. 2).

$$targetAngle = currentAngle + 2\pi \frac{distance}{2\pi \cdot diameter} \quad (2)$$

## Posizionamento

Inizialmente al segnale dell'accelerometro veniva applicato un integrale doppio per ottenere lo spostamento lineare(Eq. 3 [?]).

$$\begin{cases} \mathbf{v}_i &= \sum_{j=1}^i \frac{\mathbf{a}_{j-1} + \mathbf{a}_j}{2} (t_j - t_{j-1}) \\ \mathbf{s}_i &= \sum_{j=1}^i \frac{\mathbf{v}_{j-1} + \mathbf{v}_j}{2} (t_j - t_{j-1}) \end{cases} \quad (3)$$



## Posizionamento

Inizialmente al segnale dell'accelerometro veniva applicato un integrale doppio per ottenere lo spostamento lineare(Eq. 3 [?]).

$$\begin{cases} \mathbf{v}_i &= \sum_{j=1}^i \frac{\mathbf{a}_{j-1} + \mathbf{a}_j}{2} (t_j - t_{j-1}) \\ \mathbf{s}_i &= \sum_{j=1}^i \frac{\mathbf{v}_{j-1} + \mathbf{v}_j}{2} (t_j - t_{j-1}) \end{cases} \quad (3)$$

Abbiamo ritenuto necessario cambiare approccio, decidendo di utilizzare gli encoders delle ruote per determinare gli spostamenti.

Integriamo la velocità lineare del robot, calcolata a partire dal raggio  $R$  e le velocità angolari  $u$  delle ruote.

$$v_i = \frac{R(u_{r,i} + u_{l,i})}{2} \quad (4)$$

## Posizionamento

Inizialmente al segnale dell'accelerometro veniva applicato un integrale doppio per ottenere lo spostamento lineare(Eq. 3 [?]).

$$\begin{cases} \mathbf{v}_i &= \sum_{j=1}^i \frac{\mathbf{a}_{j-1} + \mathbf{a}_j}{2} (t_j - t_{j-1}) \\ \mathbf{s}_i &= \sum_{j=1}^i \frac{\mathbf{v}_{j-1} + \mathbf{v}_j}{2} (t_j - t_{j-1}) \end{cases} \quad (3)$$

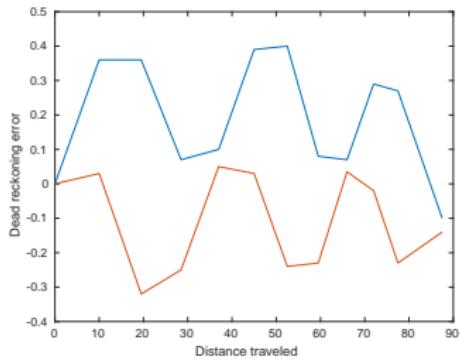
Abbiamo ritenuto necessario cambiare approccio, decidendo di utilizzare gli encoders delle ruote per determinare gli spostamenti.

Integriamo la velocità lineare del robot, calcolata a partire dal raggio  $R$  e le velocità angolari  $u$  delle ruote.

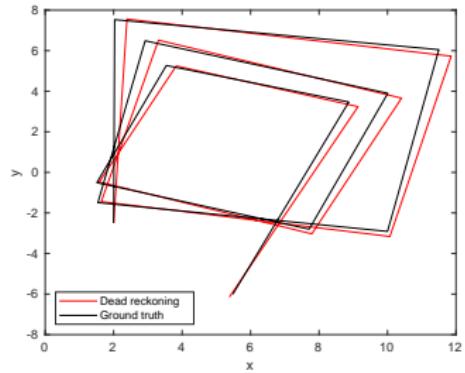
$$v_i = \frac{R(u_{r,i} + u_{l,i})}{2} \quad (4)$$

Questi valori sono aggiornati ad ogni intervallo di campionamento utilizzando la velocità lineare e la velocità angolare del robot [?].

$$\mathbf{P}_i = \sum_{j=1}^i \begin{bmatrix} v_j \cos(\theta_j) \\ v_j \sin(\theta_j) \end{bmatrix} \cdot (t_j - t_{j-1})$$



(a) Errore nella stima della posizione



(b) Errore nella stima della traiettoria

Abbiamo misurato le performance della stima di posizione e i risultati sono ritenuti soddisfacenti per raggiungere l'obiettivo proposto.

## Collision avoidance

Il **TIAGo** è in grado di rilevare gli ostacoli grazie all'utilizzo di un sensore lidar.

Nell'immagine seguente viene mostrata la zona nella quale, se viene indicata dal lidar la presenza di un ostacolo, il **TIAGo** si ferma per ragioni di sicurezza al fine di evitare danni a persone e/o oggetti.

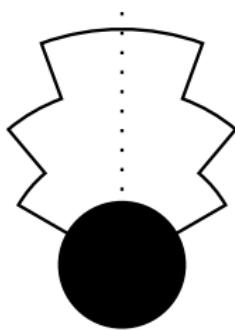


Figura: Collision avoidance

- 1** Introduzione
- 2** TIAGo Iron
- 3** Gestione dei nodi ROS
- 4** Modello del moto e posizionamento
- 5** Object recognition
- 6** Posizione dei target
- 7** Modello probabilistico
- 8** Scheduling dei comportamenti
- 9** Possibili modifiche
- 10** Conclusioni

## Section 5

### Object recognition

## Campionamento delle immagini

Il FOV della camera è di  $57^\circ$ , quindi per ricoprire  $360^\circ$  è necessario effettuare 7 campionamenti. Il settimo campionamento, come si vede in figura 4, è sovrapposto al primo per  $39^\circ$ .

È possibile che un individuo si trovi in una zona di confine tra due campioni, e che quindi non sia correttamente identificabile in nessuna delle due immagini in cui appare parzialmente. Mitighiamo questo problema effettuando una rudimentale operazione di image mosaicing [?] e campioniamo l'immagine così ottenuta ad intervalli di  $28^\circ$ .

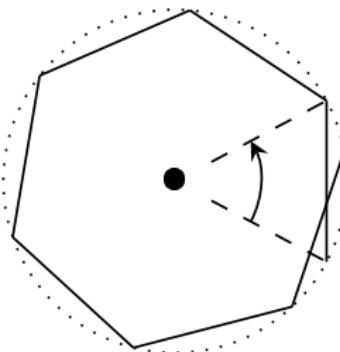


Figura: Campionamento delle immagini

## YOLO

Abbiamo valutato le performance di YOLOv3 (you only look once), YOLOv3-tiny, HoG, HoG + SVG. In seguito a vari test su HoG abbiamo ritenuto essere problematica la larghezza delle bounding boxes fornite. YOLOv3 fornisce risultati soddisfacenti.

Considerando le caratteristiche hardware del robot mobile, abbiamo optato per l'uso di YOLOv3-tiny, il quale risulta essere significativamente più efficiente (approssimativamente del 442% [?]). Inoltre è rilevante in tal senso un paragone fra YOLOv3 e YOLOv3-tiny in termini di mAP (mean average precision) e FLOPS (floating-point operations per second) addestrate sul dataset COCO, come illustrato dalla tabella.

Model	mAP	FLOPS	FPS
YOLOv3-320	51.5	38.97 Bn	45
YOLOv3-416	55.3	65.86 Bn	35
YOLOv3-608	57.9	140.69 Bn	20
YOLOv3-tiny	33.1	5.56 Bn	220
YOLOv3-spp	60.6	141.45 Bn	20

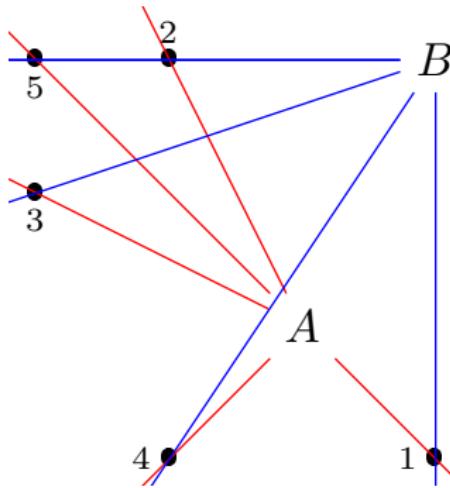
- 1** Introduzione
- 2** TIAGo Iron
- 3** Gestione dei nodi ROS
- 4** Modello del moto e posizionamento
- 5** Object recognition
- 6** Posizione dei target
- 7** Modello probabilistico
- 8** Scheduling dei comportamenti
- 9** Possibili modifiche
- 10** Conclusioni

## Section 6

### Posizione dei target

# Triangolazione

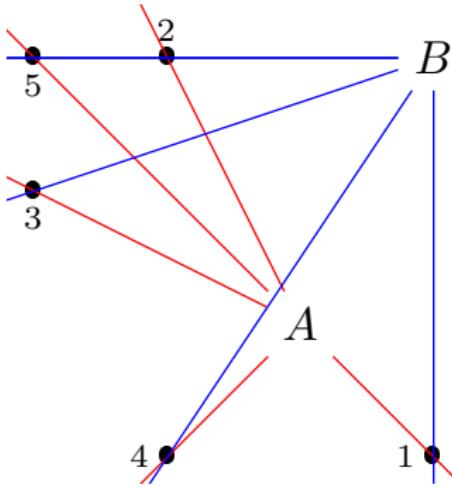
La triangolazione come metodo di individuazione delle persone presenta dei problemi nel nostro scenario:



# Triangolazione

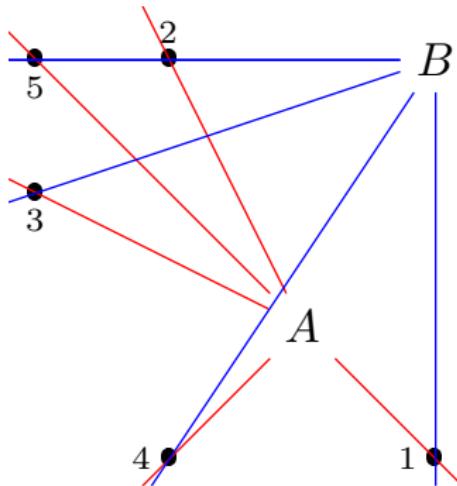
La triangolazione come metodo di individuazione delle persone presenta dei problemi nel nostro scenario:

- Occlusione delle persone: se due persone (5 e 2) sono una dietro l'altra lungo una retta immaginaria che le congiunge al robot (B), quest'ultimo non sarà in grado di individuare la più distante.



# Triangolazione

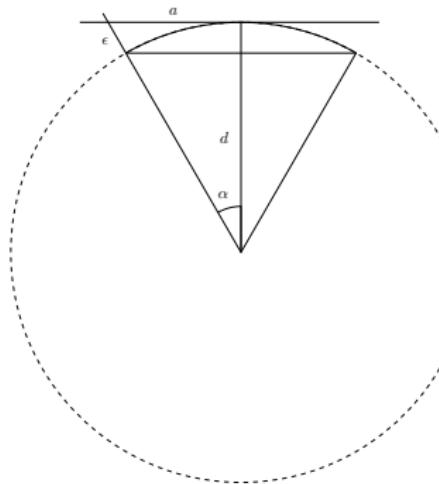
La triangolazione come metodo di individuazione delle persone presenta dei problemi nel nostro scenario:



- Occlusione delle persone: se due persone (5 e 2) sono una dietro l'altra lungo una retta immaginaria che le congiunge al robot (B), quest'ultimo non sarà in grado di individuare la più distante.
- Imputazione delle osservazioni: quando il robot effettua scan successivi non è in grado di dedurre quali osservazioni derivano dalla stessa persona. Non è possibile determinare quali intersezioni corrispondono ad osservazioni reali e quali sono spurie.

# Calcolo della distanza I

Ipotizzando che la camera abbia un FOV (field of view) di  $2\alpha$  e sia distante  $d$  dall'oggetto, la massima distanza orizzontale che un punto dell'immagine potrebbe avere dal centro del piano dell'immagine sarebbe  $a = d \tan \alpha$ .



## Calcolo della distanza II

Ignorare la prospettiva significa effettuare un'approssimazione lineare del primo ordine e trattare il punto come se si trovasse su una circonferenza di raggio  $d$  centrata sulla camera. Di conseguenza consideriamo il punto come se fosse più vicino di quanto non sia realmente, commettendo l'errore mostrato nell' Eq. 6. Con una camera con FOV di 1 radiante la sottostima è del 13.9%.

$$\begin{aligned}\epsilon &= \sqrt{a^2 + d^2} - d = \sqrt{(d \tan \theta)^2 + d^2} - d = \\ &= d \left( \sqrt{\frac{1}{\cos^2 \alpha}} - 1 \right) = d (\sec \alpha - 1)\end{aligned}\tag{6}$$

## Calcolo della distanza III

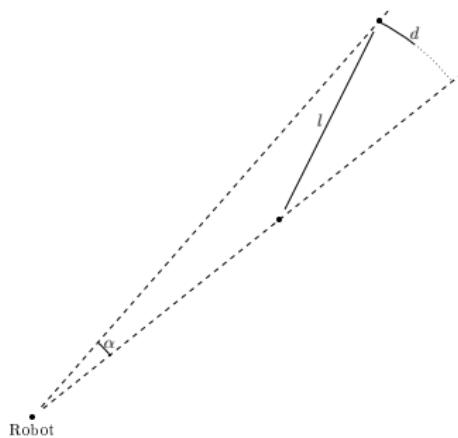
Poiché stiamo utilizzando un simulatore non è nota la larghezza del sensore da utilizzare per l'eq. 7. Abbiamo ovviato a tale problema posizionando il robot ed un oggetto dalle dimensioni note in posizioni note e abbiamo utilizzato questi dati insieme a delle misure in pixel nell' eq. 8. Abbiamo così stimato le dimensioni del sensore virtuale da utilizzare nei calcoli successivi.

$$\text{object distance}(m) = \frac{f(m) \times \text{real width}(m) \times \text{image width}(pixels)}{\text{object width}(pixels) \times \text{sensor width}(m)} \quad (7)$$

$$\text{sensor width}(m) = \frac{f(m) \times \text{real width}(m) \times \text{image width}(pixels)}{\text{object width}(pixels) \times \text{object distance}(m)} \quad (8)$$

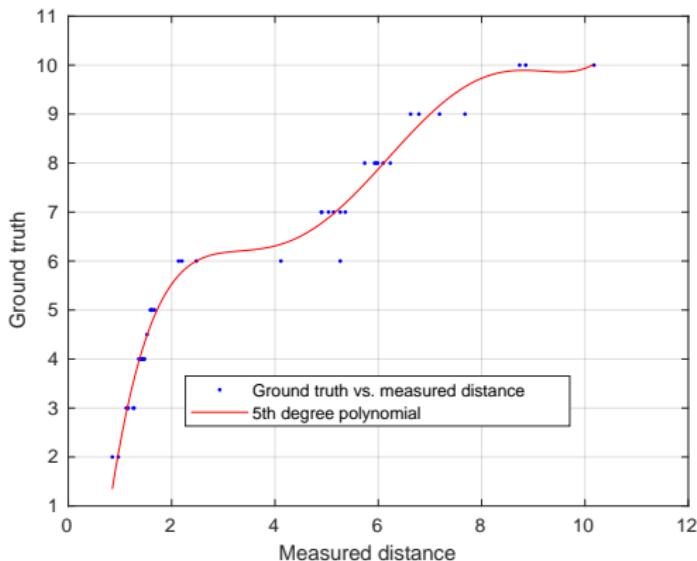
## Scarto dei duplicati I

È stato necessario effettuare una fase di clustering al fine di scartare le bounding box duplicate. L'algoritmo di clustering utilizzato è DBSCAN (Density based scan) [?], i cui parametri principali sono **eps**, ovvero la massima distanza fra due punti affinché vengano considerati appartenenti a un cluster , **min\_samples**, ovvero il numero minimo di punti affinché un cluster sia valido (nel nostro caso è uguale a 1 in quanto non vogliamo scartare ROI) ed infine la metrica di distanza.



## Correzione dei valori I

Per migliorare la stima sulla distanza abbiamo paragonato le reali distanze dei target con le stime effettuate dal sistema ottenendo il polinomio interpolante  $0.003116*x^5 - 0.09722*x^4 + 1.124*x^3 - 5.908*x^2 + 14.5*x - 7.367$ , che approssima la funzione di correzione della stima.



## Correzione dei valori II

Le bontà della stima della distanza e dell'angolo si evince dalle figure :

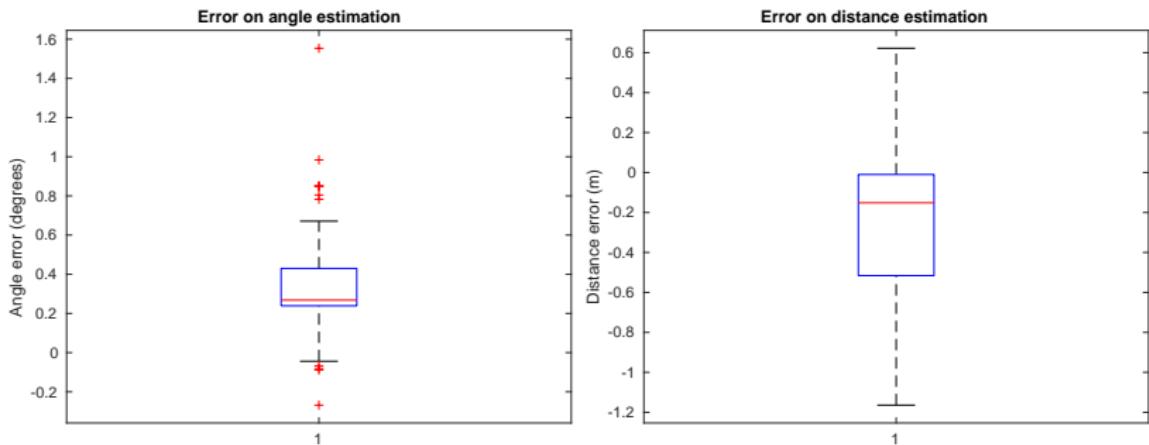
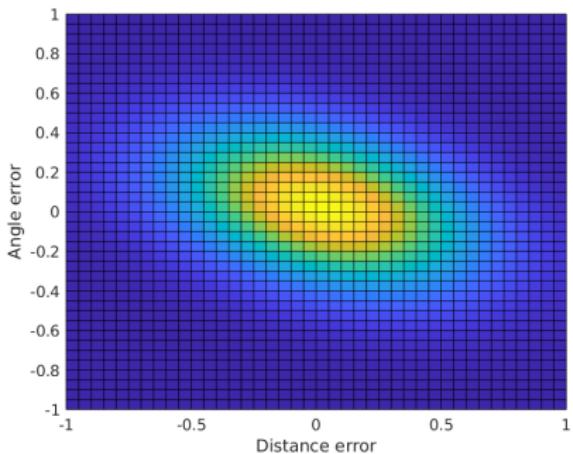
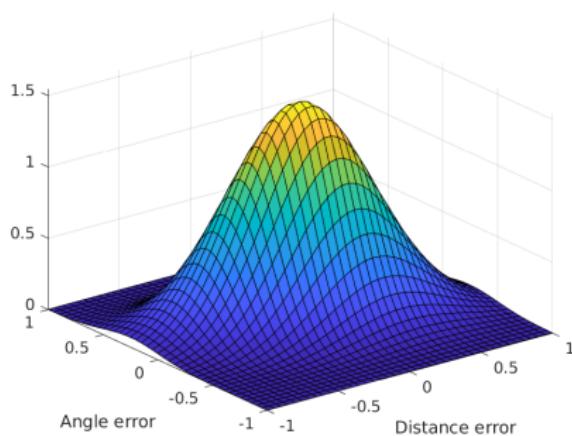


Figura: Box plot relativi all'errore su angolo e distanza

## Correzione dei valori III

Nelle figure viene riportata la distribuzione gaussiana multivariata in 3 e 2 dimensioni con matrice di covarianza calcolata dai nostri test :



- 1** Introduzione
- 2** TIAGo Iron
- 3** Gestione dei nodi ROS
- 4** Modello del moto e posizionamento
- 5** Object recognition
- 6** Posizione dei target
- 7** Modello probabilistico
- 8** Scheduling dei comportamenti
- 9** Possibili modifiche
- 10** Conclusioni

## Section 7

### Modello probabilistico



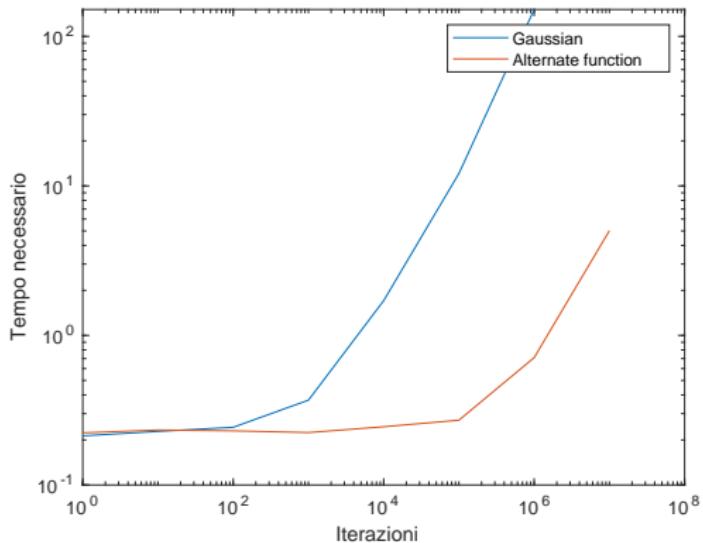
UNIVERSITÀ  
DEGLI STUDI  
DI PALERMO

## Modello probabilistico

Nella nostra applicazione la distanza stimata dell'oggetto sarà soggetta ad errore non trascurabile. Per questa ragione e per i problemi legati alla triangolazione abbiamo abbandonato la rappresentazione basata su oggetti e abbiamo fatto ricorso ad un filtro di occupazione bayesiano (BOF) [?]. Nel filtro di occupazione bayesiano, il problema dell'associazione dei dati viene superato in quanto viene gestito da un livello di astrazione superiore. Il concetto di oggetto viene difatti riformulato da proprietà più utili quali occupazione o rischio, che vengono stimate direttamente per ogni cella utilizzando sia osservazioni dai sensori che conoscenze pregresse. Le caratteristiche di incertezza legate ai sensori vengono descritte, in questo modello, attraverso le probabilità di occupazione.

## Funzione densità di probabilità I

Al fine di trasformare le osservazioni ottenute in una probabilità che le persone si trovino effettivamente nella posizione indicata è stato necessario definire una funzione densità di probabilità. La distribuzione normale è computazionalmente onerosa. Abbiamo quindi fatto ricorso ad una approssimazione (Eq. 9).



## Funzione densità di probabilità II

La distribuzione triangolare è spesso utilizzata per approssimare una gaussiana, tuttavia, presenta delle caratteristiche non volute per il nostro caso d'uso.

## Funzione densità di probabilità II

La distribuzione triangolare è spesso utilizzata per approssimare una gaussiana, tuttavia, presenta delle caratteristiche non volute per il nostro caso d'uso.

- Non presenta le "fat tails"

## Funzione densità di probabilità II

La distribuzione triangolare è spesso utilizzata per approssimare una gaussiana, tuttavia, presenta delle caratteristiche non volute per il nostro caso d'uso.

- Non presenta le "fat tails"
- Dopo una prima osservazione non confermata successivamente la probabilità scende a zero.

## Funzione densità di probabilità II

La distribuzione triangolare è spesso utilizzata per approssimare una gaussiana, tuttavia, presenta delle caratteristiche non volute per il nostro caso d'uso.

- Non presenta le "fat tails"
- Dopo una prima osservazione non confermata successivamente la probabilità scende a zero.
- Le occlusioni annullano la probabilità

## Funzione densità di probabilità II

La distribuzione triangolare è spesso utilizzata per approssimare una gaussiana, tuttavia, presenta delle caratteristiche non volute per il nostro caso d'uso.

- Non presenta le "fat tails"
- Dopo una prima osservazione non confermata successivamente la probabilità scende a zero.
- Le occlusioni annullano la probabilità
- I target in movimento non verrebbero rilevati

## Funzione densità di probabilità II

La distribuzione triangolare è spesso utilizzata per approssimare una gaussiana, tuttavia, presenta delle caratteristiche non volute per il nostro caso d'uso.

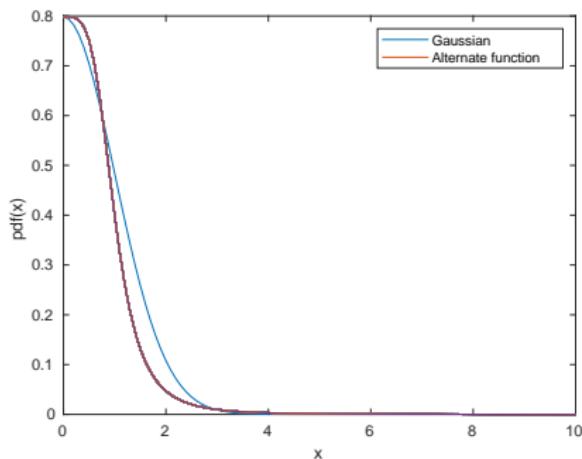
- Non presenta le "fat tails"
- Dopo una prima osservazione non confermata successivamente la probabilità scende a zero.
- Le occlusioni annullano la probabilità
- I target in movimento non verrebbero rilevati

Essendo la scansione una operazione estremamente costosa non è nemmeno possibile affidarsi esclusivamente all'aggiunta di rumore alla griglia di occupazione confidando in una eventuale convergenza.

## Funzione densità di probabilità III

Al fine di ottenere un'approssimazione di una gaussiana adatta al nostro scenario, per modellare la probabilità che data l'occupazione della cella in posizione  $x$  si ottenga l'osservazione  $z$ , è stato utilizzato un funzionale ispirato al guadagno del filtro di Butterworth.

$$p(z|x) = \frac{K}{1 + d(x, z)^4} \quad (9)$$



## Parametri PDF I

La distanza euclidea nell'equazione 9 porterebbe a formare aree ad alta probabilità di forma circolare. Questo però non si adatterebbe bene al nostro modello di errore del sensore: l'angolo dell'oggetto rispetto al robot è noto con una precisione molto elevata, mentre la maggior parte dell'incertezza si concentra nella distanza. Calcoliamo quindi la distanza non come distanza euclidea tra le coordinate cartesiane della cella e dell'osservazione, ma come norma  $L^2$  delle coordinate polari, opportunamente normalizzate per tenere conto della diversa incertezza sulla misura di angolo e distanza.

## Parametri PDF II

L'incertezza sulla distanza è ricavata dalle misure di calibrazione effettuate. Per l'angolo è stato seguito un approccio differente: utilizzare una varianza calcolata a partire da una serie di osservazioni porterebbe ad assegnare maggiore probabilità ad oggetti lontani.

In un setup come quello in figura 7, con due target rilevati, rispettivamente a distanza  $d_1$  e  $d_2$ , le aree ad alta probabilità relative ad entrambi i punti avranno lunghezza  $l$  e ampiezze  $\alpha_1$  e  $\alpha_2$ .

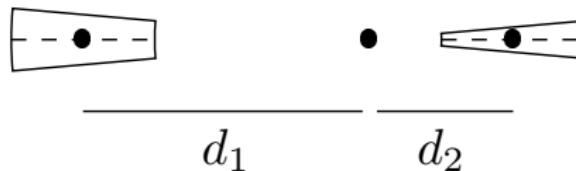


Figura: Area della regione probabile a seconda della distanza.

## Parametri PDF III

La dimensione dell'area ad alta probabilità è direttamente proporzionale alla distanza e all'ampiezza dell'angolo. Se quindi  $\alpha_1$  coincidesse con  $\alpha_2$  ad oggetti distanti verrebbero associate aree molto più grandi.

$$\begin{aligned} A_1 &= \pi((d_1 + l/2)^2 - (d_1 - l/2)^2) \frac{\alpha_1}{2\pi} = \alpha_1 d_1 l \\ A_2 &= \pi((d_2 + l/2)^2 - (d_2 - l/2)^2) \frac{\alpha_2}{2\pi} = \alpha_2 d_2 l \\ \frac{A_1}{A_2} &= \frac{\alpha_1 d_1}{\alpha_2 d_2} \end{aligned} \tag{10}$$

Per evitare questo problema, dopo avere individuato un valore  $\alpha^*$  appropriato questo viene scalato per un coefficiente inversamente proporzionale alla distanza della cella, quindi  $\alpha_1 = \alpha^*/d_1$  e  $\alpha_2 = \alpha^*/d_2$ , da cui  $A_1 = A_2$ .

## Aggiornamento bayesiano I

Dato un insieme di osservazioni ottenute da una scansione  $Z_i$  aggiorniamo il belief.

$$p(x_i|Z_i) = p(x_{i-1}) \cdot \sum_{z \in Z_i} p(z|x_i) \quad (11)$$

Questa formula è valida sotto le seguenti assunzioni:

# Aggiornamento bayesiano I

Dato un insieme di osservazioni ottenute da una scansione  $Z_i$  aggiorniamo il belief.

$$p(x_i|Z_i) = p(x_{i-1}) \cdot \sum_{z \in Z_i} p(z|x_i) \quad (11)$$

Questa formula è valida sotto le seguenti assunzioni:

- In uno stesso scan, date due osservazioni  $z_1$  e  $z_2$  queste non hanno intersezione, quindi

$$p(z_1 \cup z_2) = p(z_1) + p(z_2) - p(z_1 \cap z_2) = p(z_1) + p(z_2)$$

# Aggiornamento bayesiano I

Dato un insieme di osservazioni ottenute da una scansione  $Z_i$  aggiorniamo il belief.

$$p(x_i|Z_i) = p(x_{i-1}) \cdot \sum_{z \in Z_i} p(z|x_i) \quad (11)$$

Questa formula è valida sotto le seguenti assunzioni:

- In uno stesso scan, date due osservazioni  $z_1$  e  $z_2$  queste non hanno intersezione, quindi

$$p(z_1 \cup z_2) = p(z_1) + p(z_2) - p(z_1 \cap z_2) = p(z_1) + p(z_2)$$

- In assenza di nuove osservazioni la stima dello stato del sistema rimane invariata:  $\overline{bel}(x_i) = bel(x_{i-1})$

## Aggiornamento bayesiano II

In mancanza di nuovi dati si potrebbero tentare due approcci: resettare le stime o lasciarle del tutto invariate.

## Aggiornamento bayesiano II

In mancanza di nuovi dati si potrebbero tentare due approcci: resettare le stime o lasciarle del tutto invariate.

- Lasciare invariato il belief a seguito di multiple scansioni senza successo porterebbe a non notare che tutte le persone nell'ambiente in cui ci si trova sono andate via, continuando a considerare valide tutte le posizioni precedenti.

## Aggiornamento bayesiano II

In mancanza di nuovi dati si potrebbero tentare due approcci: resettare le stime o lasciarle del tutto invariate.

- Lasciare invariato il belief a seguito di multiple scansioni senza successo porterebbe a non notare che tutte le persone nell'ambiente in cui ci si trova sono andate via, continuando a considerare valide tutte le posizioni precedenti.
- Dall'altro lato, un approccio troppo drastico quale immediatamente scartare tutte le precedenti stime porterebbe a perdere informazioni utili a causa di occlusioni temporanee

## Aggiornamento bayesiano II

In mancanza di nuovi dati si potrebbero tentare due approcci: resettare le stime o lasciarle del tutto invariate.

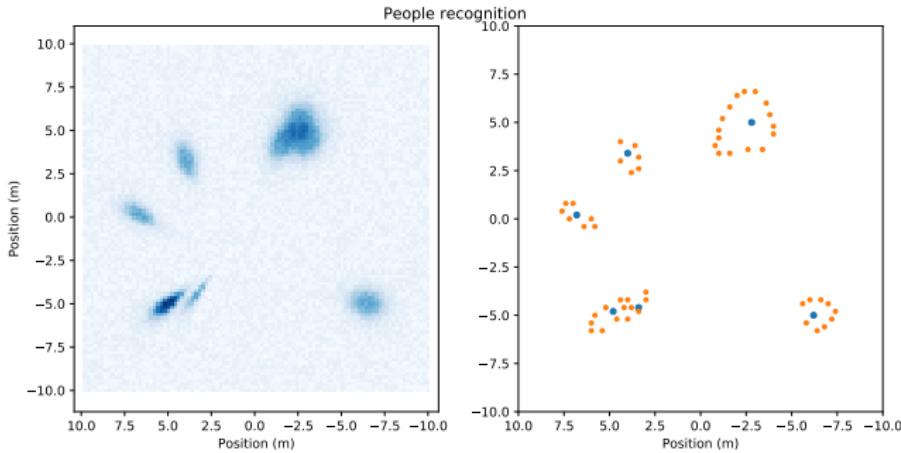
- Lasciare invariato il belief a seguito di multiple scansioni senza successo porterebbe a non notare che tutte le persone nell'ambiente in cui ci si trova sono andate via, continuando a considerare valide tutte le posizioni precedenti.
- Dall'altro lato, un approccio troppo drastico quale immediatamente scartare tutte le precedenti stime porterebbe a perdere informazioni utili a causa di occlusioni temporanee

Effettuiamo uno smoothing degli istogrammi prima di ogni update e aggiungiamo inoltre del rumore.

## Individuazione dei cluster

Al fine di individuare le zone con alta probabilità di contenere persone, in primo luogo effettuiamo una sogliatura utilizzando il metodo Otsu [?], ottenendo così una mappa binaria.

Estraiamo le regioni ad alta probabilità contigue ed i loro contorni [?] e selezioniamo come centro il punto con maggiore probabilità.



- 1** Introduzione
- 2** TIAGo Iron
- 3** Gestione dei nodi ROS
- 4** Modello del moto e posizionamento
- 5** Object recognition
- 6** Posizione dei target
- 7** Modello probabilistico
- 8** Scheduling dei comportamenti
- 9** Possibili modifiche
- 10** Conclusioni

## Section 8

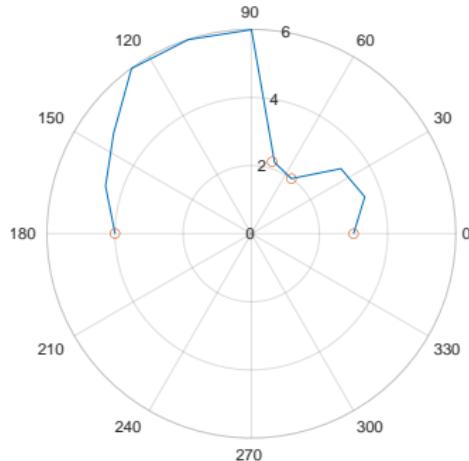
### Scheduling dei comportamenti

## Modalità esplorazione

Quando il robot entra in modalità esplorazione il suo comportamento è di esplorazione casuale della mappa. Il robot continua ad esplorare ed effettuare scan periodici fino a quando non viene rilevato un obiettivo. In tal caso il robot entra in modalità campi di potenziale. Se il robot incontra un ostacolo nel suo cammino ruota di  $90^\circ$  nella direzione con più spazio e continua l'esplorazione casuale.

## Bug mode

Quando il robot entra in modalità bug [?] effettuiamo uno scan lidar. Confrontiamo i valori ottenuti con una soglia al fine di individuare le discontinuità nel profilo degli ostacoli. Analizziamo successivamente le discontinuità del segnale, corrispondenti ai bordi degli ostacoli. Da tali punti viene calcolato l'angolo fra la retta che li congiunge con l'obbiettivo. Il robot si muove infine verso il punto al quale corrisponde l'angolo minore, che ci farà allontanare meno dall'obbiettivo.



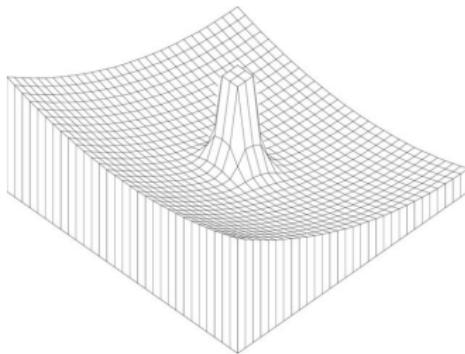
# Campi di potenziale

## Potenziale attrattivo

I modelli basati sui campi di potenziale richiedono la definizione di potenziali attrattivi e repulsivi.

Il potenziale attrattivo è centrato sul target, ed è modellato come un potenziale quadratico, la forza attrattiva è quindi lineare rispetto alla distanza (Eq. 12).

$$F_{att}(\mathbf{x}) = k_a \cdot |\mathbf{x} - \mathbf{x}_d| \quad (12)$$

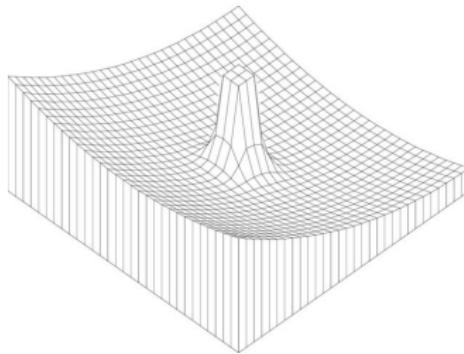


# Campi di potenziale

## Potenziale repulsivo

Gli ostacoli generano invece un potenziale repulsivo. Questo ha un raggio d'azione limitato, la forza repulsiva è massima per un ostacolo a distanza nulla, e decresce in modo monotono fino ad annullarsi se la distanza dell'ostacolo supera una soglia  $d_t$  con la legge descritta nell'equazione:

$$F_{rep}(distance) = \begin{cases} k_r \cdot \left( \frac{1}{distance^2} - \frac{1}{d_t^2} \right) & distance \leq d_t \\ 0 & d > d_{thresh} \end{cases} \quad (13)$$



# Campi di potenziale

## Rilevamento dell'ostacolo

La maggior parte dei potenziali repulsivi descritti in letteratura dipendono esclusivamente dalla distanza dall'ostacolo, interferendo con il moto anche se questo avviene parallelamente all'ostacolo. Per ovviare a questo problema prendiamo in considerazione solo gli ostacoli situati in un FOV frontale al robot con ampiezza  $\alpha$  calcolata per permettere di muoversi parallelamente ad una superficie mantenendo una distanza di sicurezza  $h$ :

$$\alpha = 2 \sin^{-1} \frac{h}{d_t} \quad (14)$$

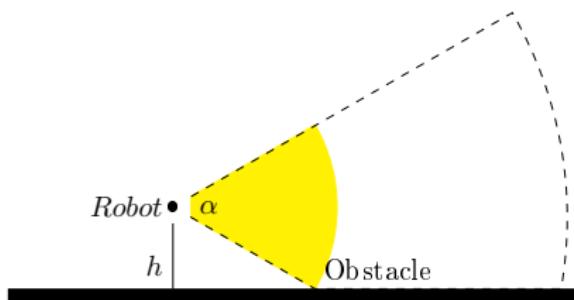


Figura: FOV frontale del robot

# Campi di potenziale

## Calcolo del movimento

L'approccio seguito consiste in due comportamenti calcolati indipendentemente: il comportamento traslatorio e il comportamento rotazionale.

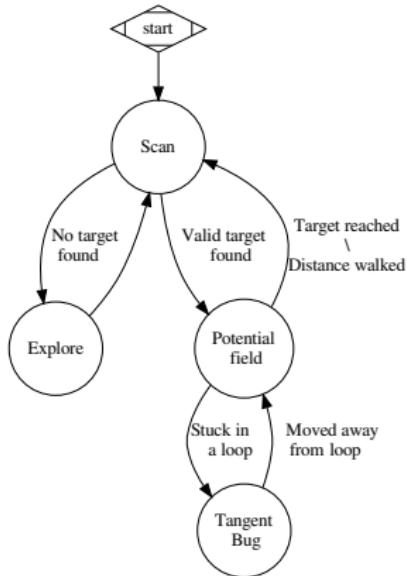
La traslazione viene calcolata in funzione dello spazio di movimento a disposizione e dall'angolo rispetto al target secondo l'equazione 15, ottenuta per interpolazione del profilo desiderato. L'angolo viene invece determinato dalla somma vettoriale delle forze attrattive e repulsive. La rotazione viene inoltre sogliata per evitare cambi di direzione troppo bruschi(Eq. 16).

$$d = 0.5 + \frac{0.9 \cdot distance - 0.5}{1 + \left| \frac{6\omega}{\pi} \right|} \quad (15)$$

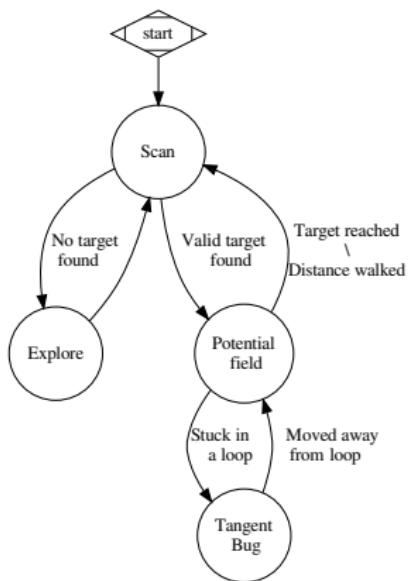
$$\omega = \begin{cases} \omega^* = \theta - \angle(\mathbf{F}_{att} - \mathbf{F}_{rep}) & -\Omega \leq \omega^* \leq \Omega \\ -\Omega & \omega^* < -\Omega \\ \Omega & \omega^* < \Omega \end{cases} \quad (16)$$

# Automa a stati finiti

- All'avvio, **Change** effettua uno scan dell'ambiente.

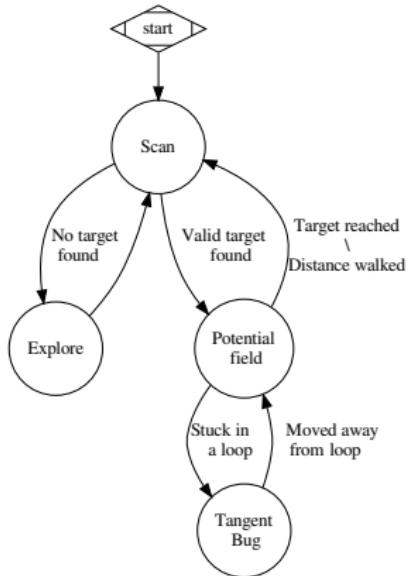


# Automa a stati finiti



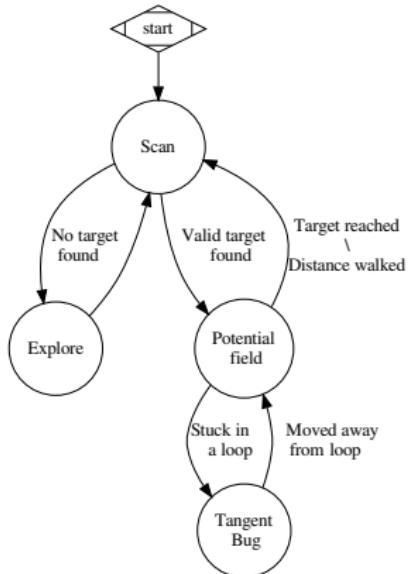
- All'avvio, **Change** effettua uno scan dell'ambiente.
- Se non vengono rilevati obiettivi entra in modalità esplorazione e, dopo aver percorso una determinata distanza, effettua un nuovo scan

# Automa a stati finiti



- All'avvio, **Change** effettua uno scan dell'ambiente.
- Se non vengono rilevati obiettivi entra in modalità esplorazione e, dopo aver percorso una determinata distanza, effettua un nuovo scan
- Se viene rilevato un obiettivo il robot entra in modalità campi di potenziale fermandosi una volta raggiunto l'obiettivo o percorso una determinata distanza.

# Automa a stati finiti



- All'avvio, **Change** effettua uno scan dell'ambiente.
- Se non vengono rilevati obiettivi entra in modalità esplorazione e, dopo aver percorso una determinata distanza, effettua un nuovo scan
- Se viene rilevato un obiettivo il robot entra in modalità campi di potenziale fermandosi una volta raggiunto l'obiettivo o percorso una determinata distanza.
- Se, in modalità campi di potenziale, il **TIAGO** rimane bloccato in un loop, entra in modalità bug fino all'uscita dal loop, e ritorna nella modalità campi di potenziale.

1 Introduzione

2 TIAGo Iron

3 Gestione dei nodi ROS

4 Modello del moto e posizionamento

5 Object recognition

6 Posizione dei target

7 Modello probabilistico

8 Scheduling dei comportamenti

9 Possibili modifiche

10 Conclusioni

## Section 9

### Possibili modifiche

Riportiamo di seguito possibili modifiche da apportare:

- Aggiungere un algoritmo per effettuare SLAM, in modo da poter pianificare il moto con algoritmi come A\* [?] o RRT [?]

Riportiamo di seguito possibili modifiche da apportare:

- Aggiungere un algoritmo per effettuare SLAM, in modo da poter pianificare il moto con algoritmi come A\* [?] o RRT [?]
- Migliorare la modalità Tangent Bug

Riportiamo di seguito possibili modifiche da apportare:

- Aggiungere un algoritmo per effettuare SLAM, in modo da poter pianificare il moto con algoritmi come A\* [?] o RRT [?]
- Migliorare la modalità Tangent Bug
- Utilizzare una depth-cam per stimare con più precisione la distanza delle persone

Riportiamo di seguito possibili modifiche da apportare:

- Aggiungere un algoritmo per effettuare SLAM, in modo da poter pianificare il moto con algoritmi come A\* [?] o RRT [?]
- Migliorare la modalità Tangent Bug
- Utilizzare una depth-cam per stimare con più precisione la distanza delle persone
- Tracciamento delle persone in real-time

- 1** Introduzione
- 2** TIAGo Iron
- 3** Gestione dei nodi ROS
- 4** Modello del moto e posizionamento
- 5** Object recognition
- 6** Posizione dei target
- 7** Modello probabilistico
- 8** Scheduling dei comportamenti
- 9** Possibili modifiche
- 10** Conclusioni

## Section 10

### Conclusioni

Durante lo sviluppo del progetto sono state affrontate numerose tematiche trattate durante il corso di *Robotica*. L'esperienza di sviluppo ha permesso al team di apprezzare l'importanza di solide fondamenta teoriche unite ad una significativa esperienza pratica. In particolar luogo abbiamo riscontrato problemi concernenti vari ambiti, dalla robotica probabilistica all'esplorazione degli ambienti, la cui ricerca di risoluzione ci ha portato ad approfondire le fondamenta teoriche della materia. La performance raggiunta, nonostante le tempistiche, risulta soddisfacente.

# Bibliografia I