

Documentazione - Progetto di Robotica

Augello Andrea Castiglione Francesco Paolo La Martina Marco

Università degli studi di Palermo

9 gennaio 2021



1 Introduzione

2 Gestione dei nodi ROS

3 TIAGo Iron

4 Modello del moto e posizionamento

5 Object recognition

6 Posizione dei target

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 [1]. Un problema chiave si configura di conseguenza come il controllo del rispetto delle norme di distanziamento all'interno degli spazi chiusi.

Obbiettivo

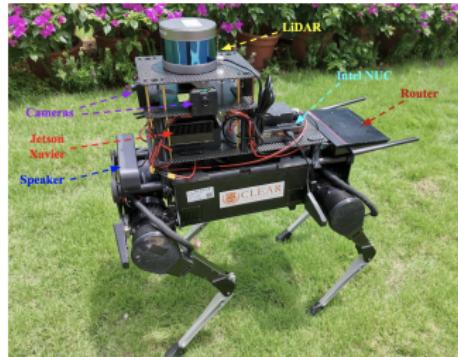
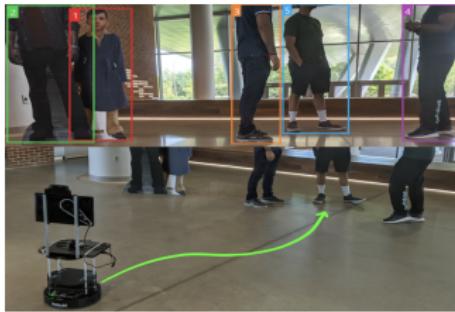
L'obbiettivo del progetto è quello di sviluppare un robot con lo scopo di **evitare assembramenti in ambienti indoor** e di invitare a **rispettare le norme sul distanziamento sociale**.

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

Un robot che si occupa di far rispettare il distanziamento sociale è quello proposto nell'articolo [2]. Sono stati usati un TurtleBot 2, una camera RGB-D e CCTV per il rilevamento degli assembramenti, una camera termica per rilevare la temperatura corporea e un lidar 2-D per evitare le collisioni.

Un altro esempio è quello proposto nell'articolo [3]. In questo caso sono state usate 2 camere e CCTV per il rilevamento degli assembramenti e un lidar 3-D per evitare le collisioni.



Setup

OS	Ubuntu 18.04 Ubuntu 20.04
ROS version	melodic noetic
Webots	R2020b revision 1
OpenCV [4]	4.x
Imutils [5]	0.5.3
Matplotlib [6]	3.3.3
Numpy [7]	1.17.2
Scikit-learn [8]	0.21.3
Target hardware	Raspberry Pi 3B+

1 Introduzione

2 Gestione dei nodi ROS

3 TIAGo Iron

4 Modello del moto e posizionamento

5 Object recognition

6 Posizione dei target

Section 2

Gestione dei nodi ROS

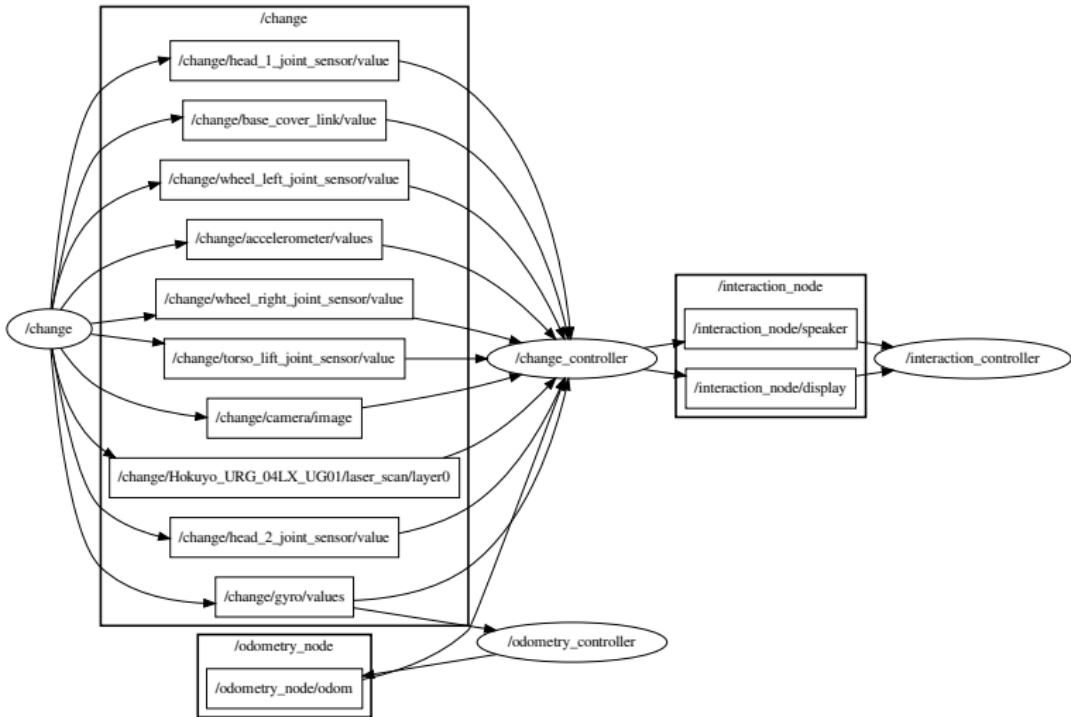


Figura: Architettura dei nodi ros, ottenuta tramite *rqt*

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:

- 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:

- 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 Gestione dei nodi ROS

3 TIAGo Iron

4 Modello del moto e posizionamento

5 Object recognition

6 Posizione dei target

Section 3

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 [9].

Il datasheet del **TIAGo** [10] indica la presenza di speaker e display, non presenti nel modello Webots [11], 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 [12]) che, conformemente al datasheet, ha un range di 5.6 m ed un FOV di 240° (agli estremi è parzialmente occluso).



1 Introduzione

2 Gestione dei nodi ROS

3 TIAGo Iron

4 Modello del moto e posizionamento

5 Object recognition

6 Posizione dei target

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 [14]).

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

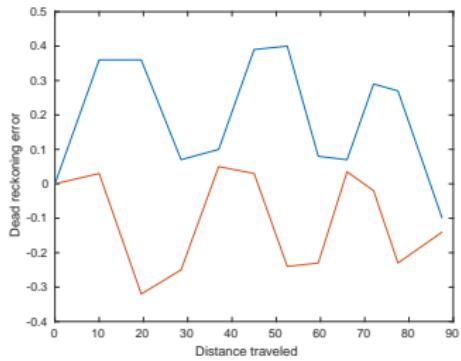
Abbiamo ritenuto necessario cambiare approccio, decidendo di utilizzare gli encoders delle ruote per determinare gli spostamenti.

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

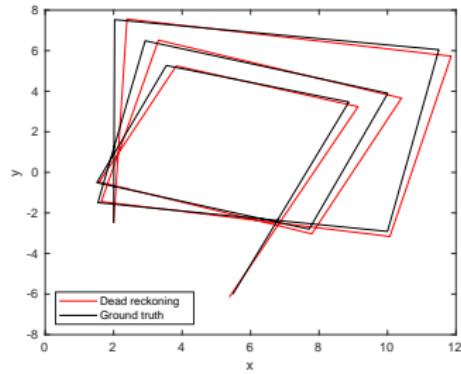
$$v_i = \frac{R(u_{r,i} + u_{l,i})}{2} \quad (4)$$

Questi valori sono aggiornati ad ogni intervallo di campionamento utilizzando la velocità lineare e la velocità angolare del robot [15].

$$\mathbf{P}_i = \sum_{j=1}^i \begin{bmatrix} v_i \cos(\theta_i) \\ v_i \sin(\theta_i) \end{bmatrix} \cdot (t_j - t_{j-1}) \quad (5)$$



(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.

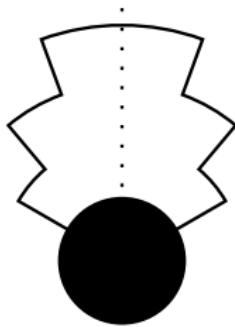


Figura: Collision avoidance

1 Introduzione

2 Gestione dei nodi ROS

3 TIAGo Iron

4 Modello del moto e posizionamento

5 Object recognition

6 Posizione dei target

Section 5

Object recognition

Campionamento delle immagini

Il FOV della camera è di 57° , quindi per ricoprire 360° è necessario effettuare 7 campionamenti. Il settimo campionamento, come si vede in figura 5, è sovrapposto al primo per 39° .

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 [16] e campioniamo l'immagine così ottenuta ad intervalli di 28° .

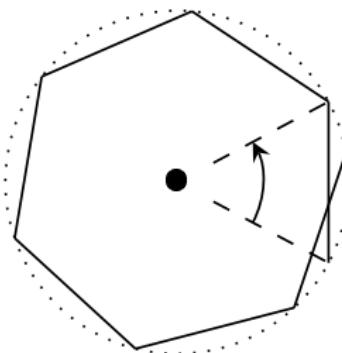


Figura: Campionamento delle immagini

YOLO

Abbiamo valutato le performance di YOLOv3 (you only look once), YOLOv3-tiny, HoG (Histogram of oriented gradients), HoG + SVG (support vector machines). 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% [17]). 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 Gestione dei nodi ROS

3 TIAGo Iron

4 Modello del moto e posizionamento

5 Object recognition

6 Posizione dei target

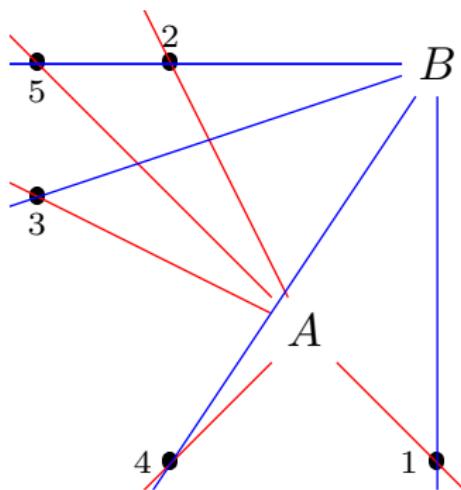
Section 6

Posizione dei target

Triangolazione

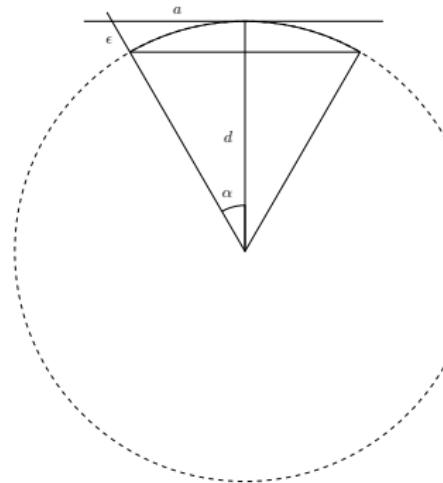
La triangolazione come metodo di individuazione delle persone presenta dei problemi nel nostro scenario:

- 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 2α 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

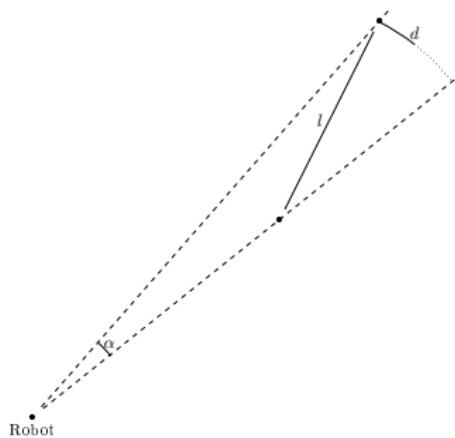
Poiché stiamo utilizzando un simulatore non è nota la larghezza del sensore da utilizzare per l'eq. 7. Abbiamo ovviato a tale problema posizionando il robot ed un oggetto dalle dimensioni note in posizioni note e abbiamo utilizzato questi dati insieme a delle misure in pixel nell' eq. 8. Abbiamo così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

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

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

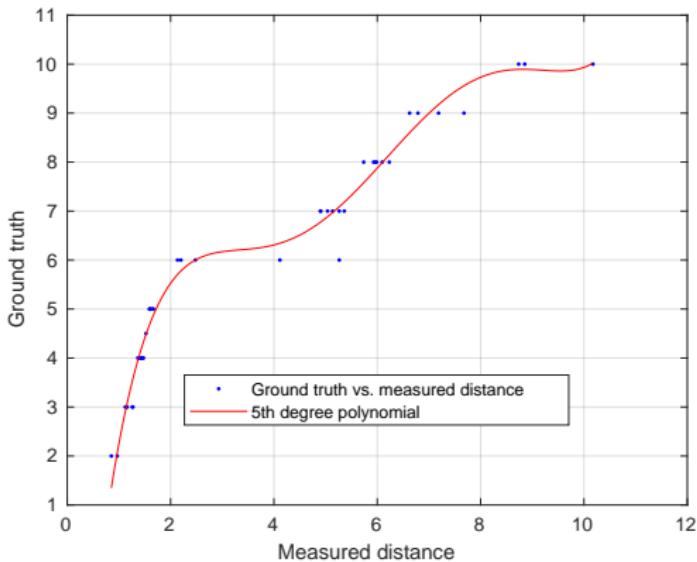
Scarto dei duplicati I

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) [20], 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 :

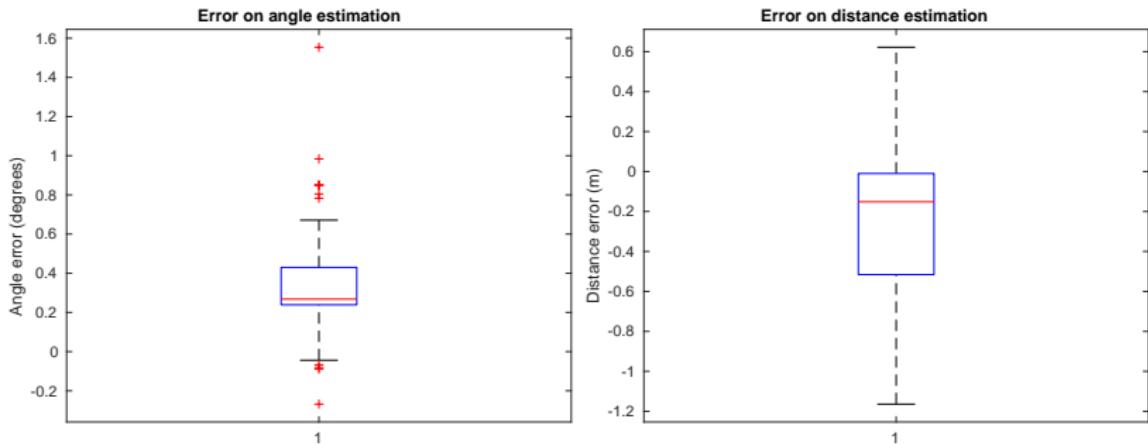
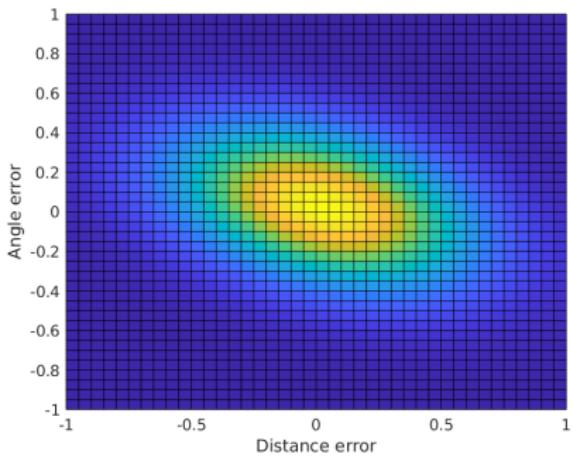
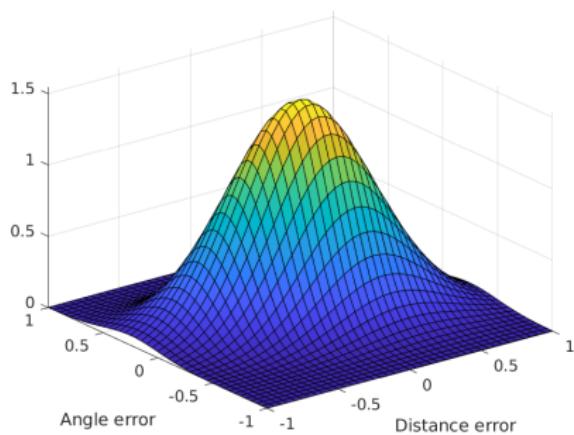


Figura: Box plot relativi all'errore su angolo e distanza

Correzione dei valori III

Nelle figure viene riportata la distribuzione gaussiana multivariata rispettivamente in 3 e 2 dimensioni :



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) [21].

Bibliografia

-  Petrônio CL Silva, Paulo VC Batista, Hélder S Lima, Marcos A Alves, Frederico G Guimarães, and Rodrigo CP Silva.
Covid-abs: An agent-based model of covid-19 epidemic to simulate health and economic effects of social distancing interventions.
Chaos, Solitons & Fractals, 139:110088, 2020.
-  Adarsh Jagan Sathyamoorthy, Utsav Patel, Yash Ajay Savle, Moumita Paul, and Dinesh Manocha.
Covid-robot: Monitoring social distancing constraints in crowded scenarios, 2020.
-  Tingxiang Fan, Zhiming Chen, Xuan Zhao, Jing Liang, Cong Shen, Dinesh Manocha, Jia Pan, and Wei Zhang.
Autonomous social distancing in urban environments using a quadruped robot.
arXiv preprint arXiv:2008.08889, 2020.
-  G. Bradski.
The OpenCV Library.
Dr. Dobb's Journal of Software Tools, 2000.
-  A Rosebrock