Project log - Robotica

Augello Andrea Castiglione Francesco Paolo La Martina Marco

30 dicembre 2020

Indice

1 Setup

OS	Ubuntu 18.04 Ubuntu 20.04
ROS version	melodic noetic
Webots	R2020b revision 1
Target hardware	Raspberry Pi 4B Raspberry Pi 3B+

2 Nome

Il team ha scelto il nome **Change** in onore di **Chang'e 4** [2], la missione parte della seconda fase del programma cinese di esplorazione lunare, durante il quale è andato a buon fine il primo atterraggio morbido sulla faccia nascosta della luna.

3 Ambiente

Abbiamo considerato opportuno analizzare e studiare il package **webots_ros** [3] al fine di raggiungere una comprensione più profonda sulle metodologie per interfacciare i nodi ROS con il controller ROS standard per Webots. Inoltre è risultato necessario approfondire la documentazione ROS [4] al fine di installare e configurare l'ambiente ROS ed inoltre per capire i concetti fondamentali relativi ai nodi e topics. Infine abbiamo impostato l'interfaccia ROS su Webots seguendo la documentazione cyberbotics rilevante [4].

4 Dipendenze

La seguente è una lista delle librerie utilizzate nel nostro progetto ed una breve spiegazione della loro funzione e rilevanza:

- opencv 4.x, una libreria per la computer vision, usata per operazioni di segmentazione [10];
- imutils, che include funzioni per semplici operazioni di image processing quali traslazioni, rotazioni, ridimensionamento. Utilizzato inoltre per effettuare Non Maxima Suppression(NMS) [11];
- sklearn, una libreria per il machine learning comprendente algoritmi di clustering quali DBSCAN [12];
- numpy, una libreria che fornisce supporto per array multidimensionali, matrici ed operazioni matematiche per lavorare su detti array [13];
- matplotlib, una libreria per creare visualizzazioni di dati (statiche, dinamiche, interattive) [14];
- math, una libreria che fornisce funzioni matematiche definite dallo standard C [15].

5 Obbiettivo

L'obbiettivo del robot è di **evitare assembramenti in ambienti indoor**.
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.

6 TIAGo Iron

Il robot scelto per l'obbiettivo proposto è il **TIAGo Iron**.
Il **PAL Robotics TIAGo Iron** [1] è un robot umanoide a due ruote con torso e testa ma senza braccia articolate. Il modello è una piattaforma modulare mobile che permette l'interazione fra esseri umani e robot.
Abbiamo ritenuto necessario per il nostro scopo aggiungere uno **speaker** e un **display** con corrispondente solido di supporto al modello base del **TIAGo** disponibile in Webots. Inolte è stato necessario contattare gli sviluppatori di Webots per chiedere informazioni circa le dimensioni esatte delle **ruote** in quanto il modello Webots non corrisponde perfettamente alle specifiche indicate nel datasheet [6]. Le ruote del modello hanno raggio di 200mm.

L'IMU utilizzata ha 6 gradi di libertà ed è composta delle seguenti componenti:

1. giroscopio;
2. accelerometro;

Abbiamo ritenuto non necessario aggiungere il **magnetometro** in quanto in uno scenario reale sarebbe stato soggetto ad interferenze (significativamente più di un giroscopio), specialmente in un ambiente con molti oggetti metallici (quale potenzialmente lo scenario di utilizzo del nostro robot).

7 Modello del moto e posizionamento

Il modello del moto è caratterizzato da rotazioni e traslazioni. Per ottenere l'angolo di rotazione ci basiamo sui dati ottenuti dal giroscopio, il quale misura il moto rotazionale fornendo una velocità angolare. Per ottenere l'angolo di rotazione effettuiamo quindi un'integrazione discreta dei campioni con interpolazione lineare del primo ordine. Per effettuare lo spostamento lineare utilizziamo il controllore PID (Proporzionale-Integrale-Derivativo) delle ruote fornito da Webots, che richiede l'angolo di rotazione corrente, il diametro delle ruote e fornisce l'angolo di rotazione necessario al fine di ottenere lo spostamento desiderato.

$$targetAngle = currentAngle + 2\pi \frac{distance}{2\pi \cdot diameter} \tag{1}$$

A causa di possibili imprecisioni calcoliamo la stima dello spostamento lineare utilizzando l'accelerometro. Al segnale dell'accelerometro viene applicato un integrale doppio per ottenere lo spostamento lineare.

8 Object recognition

Al fine di riconoscere le persone è stato necessario utilizzare sistemi di **object recognition**. A tal fine abbiamo valutato le performance di YOLOv3 (You only look once), YOLOv3-tiny, Hog (Histogram of gradients), Hog + SVG (Support Vector Machines) + NMS (Non Maxima suppression). In seguito a vari test su Hog abbiamo ritenuto essere problematica la larghezza delle bounding boxes fornite, in quanto, per motivazioni che verranno chiarite nel paragrafo successivo, vogliamo che queste ultime siano il più possibili vicile alla reale larghezza delle persone. YOLOv3, nonostante sia stato

addestrato su foto di persone reali (e non modelli 3D) fornisce risultati soddisfacenti, in seguito al finetuning degli iperparaemtri parametri della rete. Tuttavia, considerando le caratterisiche hardware del robot mobile, abbiamo optato per l'uso di YOLOv3, il quale risulta essere significativamente più efficiente, sacrificando in termini di precisione ma comunque sufficientemente preciso per il nostro obbiettivo. Ecco un paragone fra YOLOv3 e YOLOv3-tiny in termini di mAP (mean average precision) e FLOPS (floating-point operations per second), come illustrato dalla tabella seguente, i cui dati provengono dal sito di YOLO [16]:

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

9 Projection Matrix

[7]

10 Clustering

We decided to lower the dimensionality of our data. We used cylindrical coordinates and the feature vector is 2 dimensional. We used the Density-Based Scan with a threshold. The entities not belonging to the cluster are discarded.

11 ROS

12 Bugs found in the Webots ROS Controller

Logical values did not allow callbacks.

13 Distance calculation

The perceived object height is not a reliable indicator of its distance since part of the object may be occluded or not present in the frame. Moreover, some classifiers like the HoG based ones tend to produce ROIs significantly taller than the object.

The torso width, on the other hand, is less susceptible to these issues, and does not depend on the pose (e.g. sitting vs standing). We however need to assume a cylindrical torso, introducing some error if the target is not facing the camera.

The horizontal position of the object relative to the camera can influence its perceived width, shrinking it the further it is from the center of the image. Assuming that a camera has a FOV of 2α and has a distance d from the object, the maximum horizontal distance a point in the image can have from the center of the image plane is $a = d \tan alpha$ (Fig. 1).

Ignoring the perspective means performing a first order linear approximation and treating the point as if it lies in a circumference centered on the camera with d as its radius. Hence, we consider the point being closer, committing the error shown in Eq. 2. With a camera FOV of 1 radiant as with the TIAGo this means that the maximum error due to the linearization amounts to a 13.9% underestimate.

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

Under these assumptions we can compute the distance of an object as shown in Eq. 3

$$object\ distance(m) = \frac{f(m) \times real\ width(m) \times image\ width(pixels)}{object\ width(pixels) \times sensor\ width(m)} \tag{3}$$

Since we are using a simulator, there is no camera sensor with a width to plug inside eq. 3. By placing both the robot and an object with known size in known positions and using this data together with pixel measurements in eq. 4, we estimated the virtual sensor size to be used in all following computations.

$$sensor\ width(m) = \frac{f(m) \times real\ width(m) \times image\ width(pixels)}{object\ width(pixels) \times object\ distance(m)} \tag{4}$$

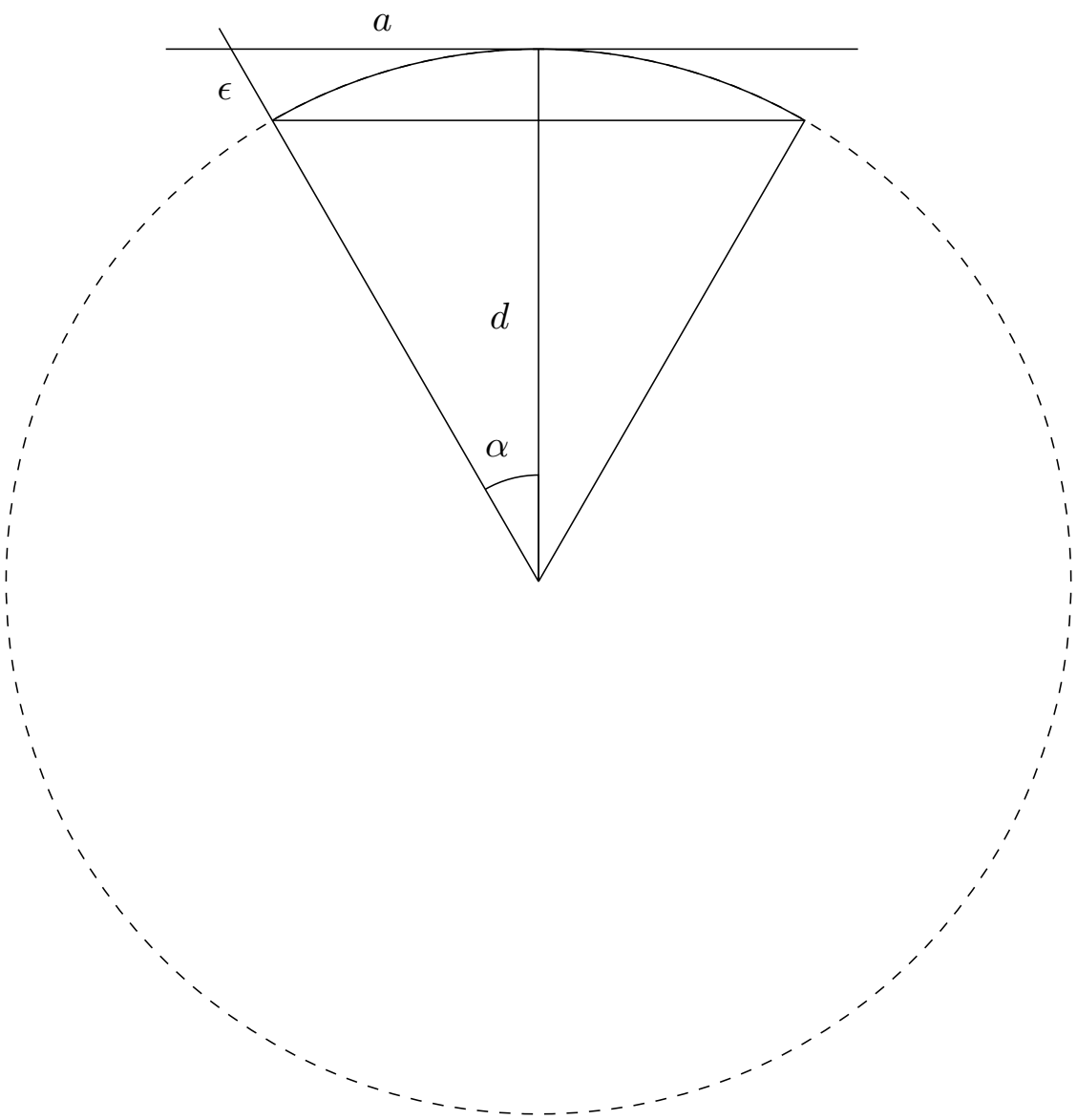

Figura 1: Error in distance estimate with piecewise circle linearization

Riferimenti bibliografici

[1] *https://cyberbotics.com/doc/guide/tiago-iron.*
Webots TIAGo Iron documentation.

[2] *https://www.theguardian.com/science/2019/jan/03/china-probe-change-4-land-far-side-moon-basin-crater.*
The Guardian, 3 January 2019.

[3] *https://github.com/cyberbotics/webots_ros.*
Github page for the `webots_ros` package from *cyberbotics*.

[4] *https://wiki.ros.org/ROS/Tutorials.*
ROS documentation from ROS.org.

[5] *https://www.cyberbotics.com/doc/guide/tutorial-8-using-ros.*
Cyberbotics documentation.

[6] *https://pal-robotics.com/wp-content/uploads/2019/07/Datasheet_ TIAGo_ Complete.pdf.*
Tiago IRON datasheet.

[7] *https://www.songho.ca/opengl/gl_ projectionmatrix.html.*
OpenGL Projection Matrix.

[8] *https://www.nxp.com/docs/en/application-note/AN3397.pdf.*
Implementing Positioning Algorithms Using Accelerometers.

[9] *https://people.eecs.berkeley.edu/ pabbeel/cs287-fa11/slides/gmapping.pdf.*
Gmapping from UC Berkeley EECS, Pieter Abbeel.

[10] *https://opencv.org/.*
OpenCV Website.

[11] *https://github.com/jrosebr1/imutils.*
Imutils GitHub page.

[12] *https://scikit-learn.org/stable/.*
Scikit-learn website.

[13] *https://numpy.org/.*
Numpy website.

[14] *https://matplotlib.org/.*
Matplotlib website.

[15] *https://docs.python.org/3/library/math.html.*
Python documentation.

[16] *https://pjreddie.com/darknet/yolo/.*
Yolo website.