Project log - Robotica

Augello Andrea Castiglione Francesco Paolo La Martina Marco

28 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 Name

Our team has chosen the name **Change**, which resembles **Chang'e 4** [2], the spacecraft mission part of the second phase of the Chinese Lunar Exploration Program, which achieved humanity's first soft landing on the far side of the moon.

3 Environment

We have explored the **webots_ros** [3] package in order to gain deeper understanding of how to interface ROS nodes with the standard ROS controller for Webots. We have also studied the ROS documentation [4] in order to install and configure the ROS enviroment and also to understand fundamental ROS concepts related to nodes and topics. Moreover, we set-up the ROS interface in Webots following the cyberbotics documentation [4].

4 Dependencies

This is a list of the libraries used in our project and a brief explanation of their relevance:

- opencv 4.x, a library aimed at computer vision[10];
- imutils, series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization[11];
- sklearn, a machine learning library, featuring various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN[12];
- numpy, a library that adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays[13];
- matplotlib, a comprehensive library for creating static, animated, and interactive visualizations[14];

5 Task

Our robot will be deployed in a room (such as the one showed in our demo) and its aim is to identify humans and avoid gatherings. It must estimate people's relative positions and, if the distance between said humans is less that a specified value, the robot will go towards them and invite them to respect social distancing (with both visual and audio output).

6 Tiago Iron

The robot selected for the given task is the **TIAGo Iron**.
PAL Robotics TIAGo Iron[1] is a two-wheeled human-like robot with a torso and a head but no articulated arm. The model is a modular mobile platform that allows human-robot interaction.
We added a **speaker** and a **display** with a corresponding support solid to the base TIAGo model available in Webots. We also had to ask the Webots developers for the precise size of the **wheels** since the model does not exactly match the specifications given in the data TIAGo datasheet[6] and we discovered that they are 200mm.
We also decided to modify the IMU in order to best fit our goals and use an IMU with 6 degrees of freedom. The IMU consists of the following components:

1. gyro;
2. accelerometer;

We decided to **not use the compass** because in a real scenario it would have been subject to various degrees of interference (significantly more so than a gyro), especially in an environment with many metal objects.

7 Positioning

In order to obtain the linear motion from the IMU a double integral is applied to the signal. The mathematical reasoning behind such approach is to remember that acceleration is the rate of change of the velocity of an object. At the same time, the velocity is the rate of change of the position of that same object. Since integration is the opposite of the derivative, if we know the object's acceleration we can get the position through double integration.

8 Projection Matrix

[7]

9 Object recognition

We evalued performance between YOLO V3, TinyYOLO, HoG , HoG + SVG , HoG + SVG + NMS. Yolo wins because it is 443% more efficient. Width and not height. Yolo yields much tighter bounding boxes.

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).

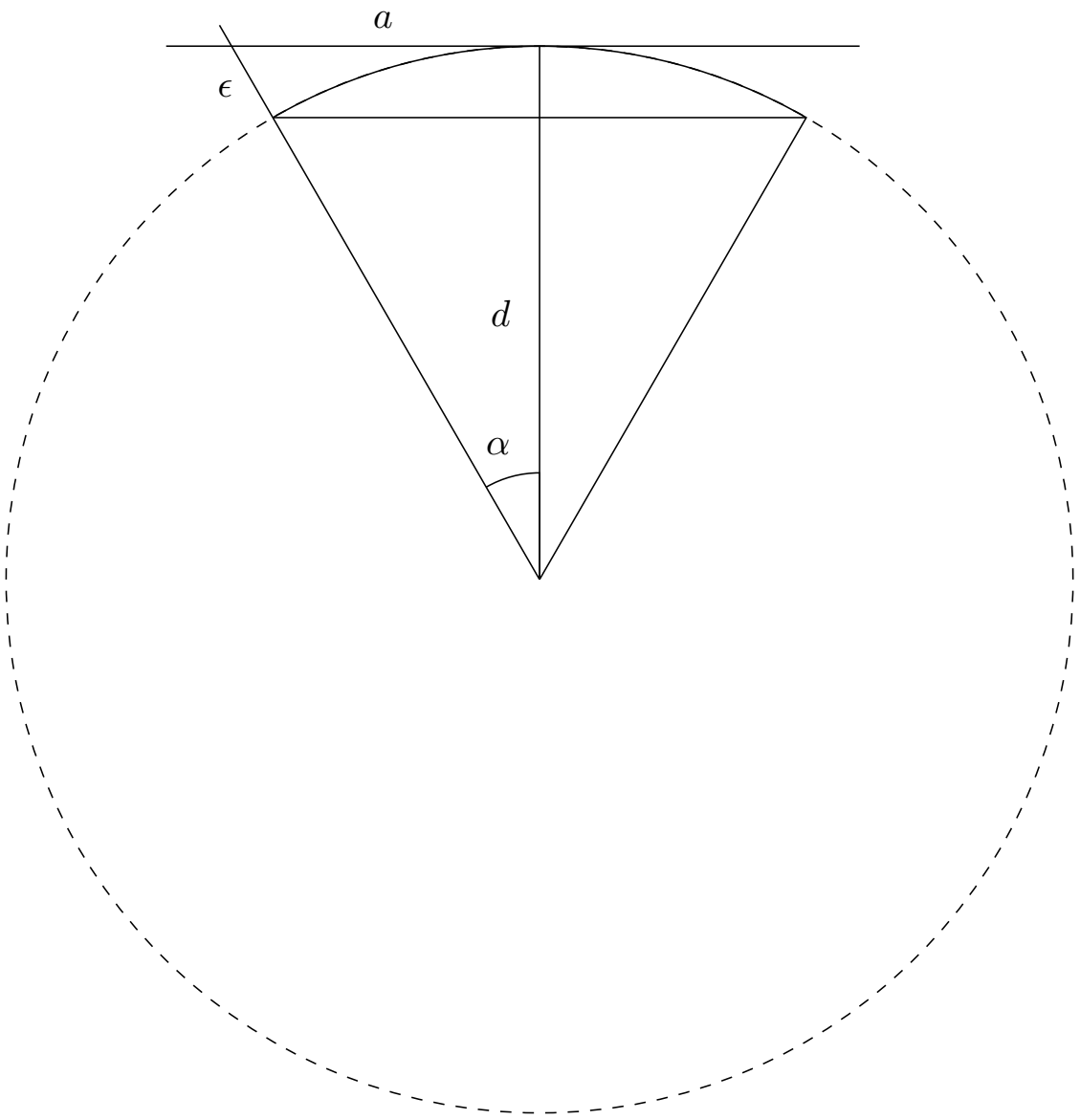

Figura 1: Error in distance estimate with piecewise circle linearization

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. 1. 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.

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

Under these assumptions we can compute the distance of an object as shown in Eq. 2

$$object\ distance(m) = \frac{f(m) \times real\ width(m) \times image\ width(pixels)}{object\ width(pixels) \times sensor\ width(m)} \tag{2}$$

Since we are using a simulator, there is no camera sensor with a width to plug inside eq. 2. By placing both the robot and an object with known size in known positions and using this data together with pixel measurements in eq. 3, we estimated the virtual sensor size to be used in all following computations.

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

With this information it is possible to measure the size of an unknown object when it is close enough to be picked up by the distance sensor, and use this information to compute its distance when it is too far to be measured directly. moreover, the camera can pick up multiple known objects at a time, so its position relative to the environment can be computed with a higher accuracy than what would be possible with only a distance sensor.

Since an object can be observed multiple times, it is possible to further refine the object size estimate by averaging different measurements. This way the estimated distance from the image may actually end up being more accurate than the one given by the rangefinder because it does not suffer from the same sensor noise.

In this work we used an exponential moving mean so, after the *ith* measurement, the new object size estimate will be

$$Real\ height_i = (3 \times Real\ height_{i-1} + new\ estimate)/4 \tag{4}$$

The moving mean is initialized with the first estimate.

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.