Visual Servoing Project

UNIVERSITÉ DE BOURGOGNE

Group : Camille Tagilone & Walid Brahim

Contents :

1 - Introduction

The Visual Servoing project regards the movement of the mobile robot “turtlebot3" which is “controlled” through a fisheye camera & Aruco markers.

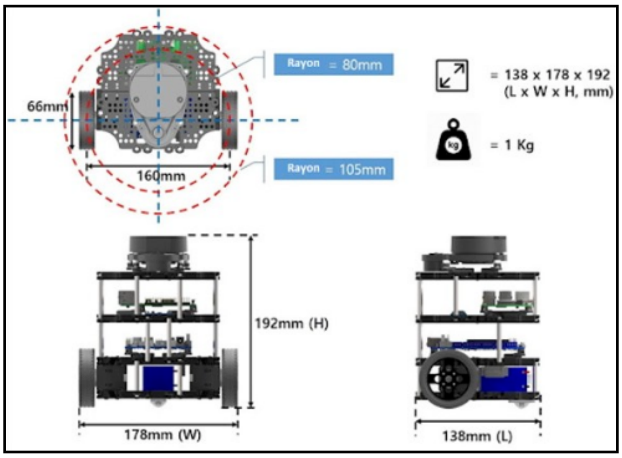
2 - Goals

- Camera Calibration
- Image acquisition
- Arcu marker detection
- Pose estimation
- Robot navigation

3 - Turtlebot3 & ROS

Turtlebot3 components :

- 360 degree LiDAR
- Single board computer (Raspberry Pi 3)
- 2 servo-powered wheels (top speed 0.22 m/s)
- Raspi RGB camera



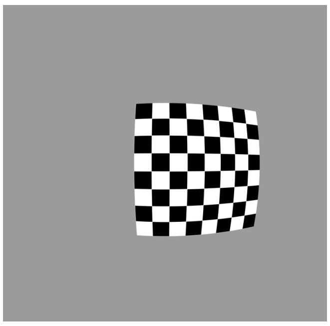
Connecting to Turtlebot3:

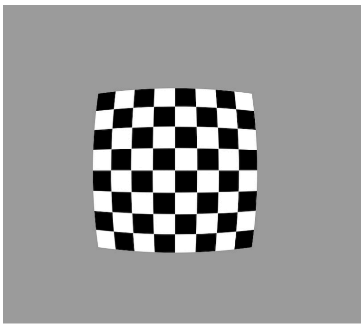
A common way to control the turtebot3 is to remotely connect to the workstation.

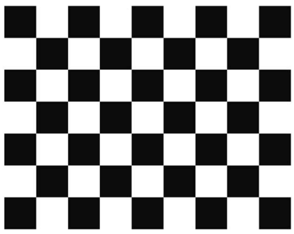
This can be easily done with Ubuntu by using the SSH build-in command, but in order to enable the remote control, you need to modify the bashrc file of the workstation (often located in the Home folder).

Calibration:

To achieve this, the library <<OpenCV>> was used in order to calibrate the fisheye camera using a checkerboard as the principal calibration target.







Intrinsic calibration :

As the type of distortion of the checkerboard is Radial distortion, represented by the equations below :

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \qquad y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

And their coefficients can be extracted :

$$DistortionCoefficients = (k_1, k_2, p_1, p_2, k_3)$$

And can be represented as a matrix :

$$cameramatrix = \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix}$$

Image acquisition :

The next step is to get the images. This is done by subscribing to the ROS topic “/camera/image_raw” and changing their format into arrays using the <<Numpy>> library.

Arcu markers detection :

Using the <<cv2.detectMarkers>> from the <<OpenCV>> library, and extract the corners of the markers for later use.

Pose estimation :

Using the <<cv2.estimatePoseSingleMatrix>>, and receive two vectors, one for each Arcuo marker, a translational vector [x, y, z] and a rotational vector [x, y, z]’ to be used for ROS publishing.

4 - Implementation and results

To see the implementation and results please check our GitHub repository which contains all the code.

5 - References

1. Turtlebot.com. 2020. *About - Turtlebot*. [online] Available at: <https://www.turtlebot.com/about/> [Accessed 12 April 2020].

2. Goebel, R., 2015. *ROS By Example*. 1st ed. [Raleigh, NC]: Lulu.com.

3. Emanual robotics website https://emanual.robotis.com/docs/en/platform/turtlebot3/autonomous_driving/#autonomous-driving

4. ROS.org website http://wiki.ros.org/turtlebot3_autorace