Handbook for Matlab GUI
Positioning Algorithms Comparison

PABLO DEL HOYO RODRÍGUEZ
Technische Universität Hamburg-Harburg
October 9, 2016

Abstract

This document presents the detailed information about the program used to compare some of the localization algorithms. The program consists of a Graphical User Interface (GUI) for Matlab which easily set a real case scenario and presents the data gathered by Pozyx device mounted on an Arduino board. The GUI provides a pop-up menu where an algorithm can be chosen in order to compare the estimated position. Also, it may localize an agent in real time, tracking the position of the Arduino board.

Keywords: Matlab, GUI, Arduino, Pozyx, Positioning, Algorithm

Contents

1 Prerequisites

As the program will provide a communication between the Arduino board and the computer, two parts will be considered in order to make the installation the Matlab side and the Arduino side:

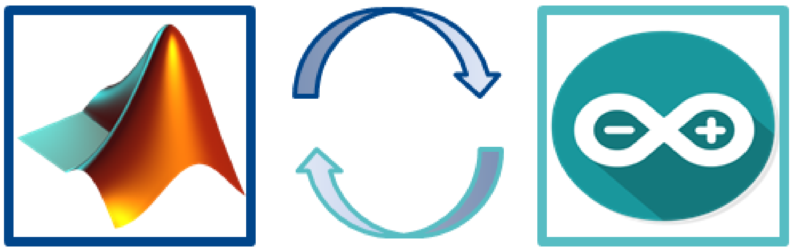


Figure 1: Both sides of the program

For the Matlab site it is required:

- Matlab R2016a (older versions may be valid)
- Arduino Support from Matlab: package to communicate with an Arduino. The installation steps are available at reference [1] and is usually installed by double clicking the package download file.

For the Arduino side is required:

- Arduino Uno x1
- USB-B Cable x1
- Pozyx Shield for Arduino
- The Arduino IDE and Pozyx Arduino Library: the installation steps are provided at reference [2]
- Pozyx Anchor x[3, 6]
- USB-C Cable x[3, 6]

2 Script Files

The localize.zip file is provided with the following files:

File	**Description**
HandbookLocalize.pdf	This file, description and a tutorial of the program
localize.m	Main Matlab file. To be executed
localize.fig	Matlab Figure for the GUI
positioning_arduino.ino	Scketch to run onto the Arduino, do the localization See section 3 for more details
(subfolder) functions	Some Matlab functions including these below
changeVisibility.m	Hide buttons during execution to prevent errors
displayAnchors.m	Plot the anchors
initArduino.m	Initialize a serial communication Matlab-Arduino
presentBP.m	Plot the estimated position for Non-Parametric Belief Porpagation (BP) algorithm. See [3]
presentInfo.m	Split the information read by the Arduino. Calulates the Pozyx position.
presentPozyx.m	Plot the estimate position for Pozyx algorithm. See [4] for more details
setAnchors.m	Present the information read by the anchors
updatePorts	Update the communication ports available (USB)
(subfolder) functionsBP	Several functions to calculate the BP solution

Table 1: List of files

3 Arduino Setup

Before running the main Matlab file (*localize.m*), some restrictions about the scenario have to be taken into consideration:

- First and most important, the identifiers of the anchors must be specified in the sketch *positioning_arduino.ino*. The number of anchors must be comprised between three and six anchors. The external library does not support fewer or more anchors, so it is out of the scope of the program to work with another amount of anchors. This can be done by editing the following lines:

  ```
  // Edit with your devices:
  uint8_t num_anchors = 4;
  uint16_t anchors[4] = {0x6F13, 0x6F17, 0x6F23, 0x6F24};
  ```

- Only one tag is possible. This is due to the calibration algorithm, which considers every tag as an anchor. The distance to the tag could be gotten as well as its coordinates. This means that the tag would behave indistinctly to an anchor. Consequently, only non-cooperative algorithms are possible.
- The first anchor would define the origin of the coordinates map as the position (0,0). The second one would define the x axis and the third one the y axis, so it is desirable to put them as orthogonal as possible. This has to be taken into consideration if only positive coordinates and a more intuitive map are desired. It is also strongly recommended to separate all the devices at least a minimum distance (around 0.2 meters). If not, some

errors in calibration may appear. Nevertheless, it is contemplated negative coordinates and not orthogonal axis. Again, section [4] is referred in order to get more information.

- Coverage between all the devices involving the scenario is assumed. The Pozyx devices have a range coverage of about 100 meters in LOS conditions, so larger distances or other conditions have been not contemplated.
- The process of the calibration is automatically done. In any case, the doc explains that the coordinates of the anchors can be manually set too if the coordinates are specified (in mm) and some code is uncommented as the following lines show:

```
// only required for manual anchor calibration.

  int32_t anchors_x[4] = {0, 1000, 0, 2000};
  int32_t anchors_y[4] = {0, 0, 1000, 2000};

// ...

// comment out the doAnchorCalibration if you are using
   manual mode

  SetAnchorsManual();
```

4 Overview

Once all the previous sections have been explained, the procedure to run the interface is described. The file localize.m has to be opened (by double-clicking or with the Matlab interface). After setting the path to the one corresponding to this project, the file can be executed (F5). This will open the interface, whose appearance is shown in the screenshot below:

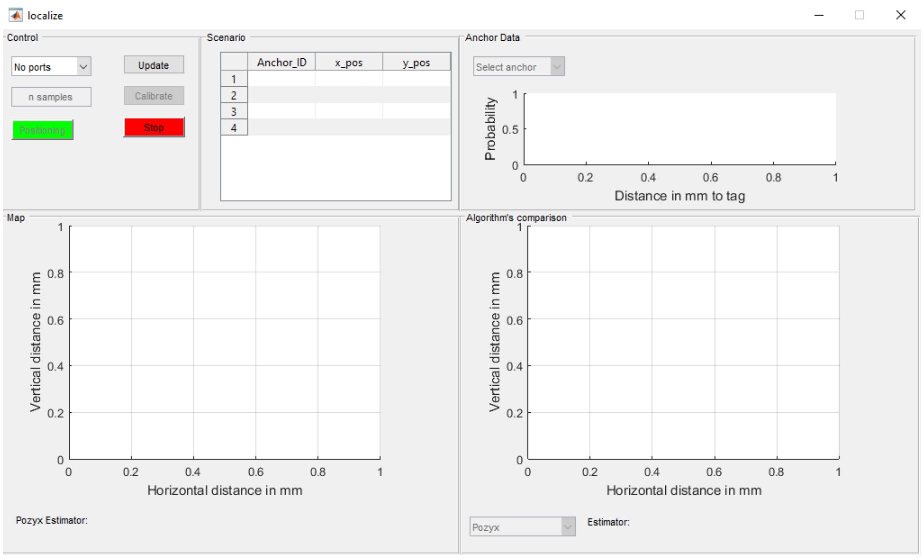


Figure 2: The interface of the program running on Windows 10.

Five areas can be distinguished. The next sections indicates the functionalities for each one.

5 Control

This panel is in charge of the problem setup. Mainly is here where the options of the program are. The buttons are shown and explained above:

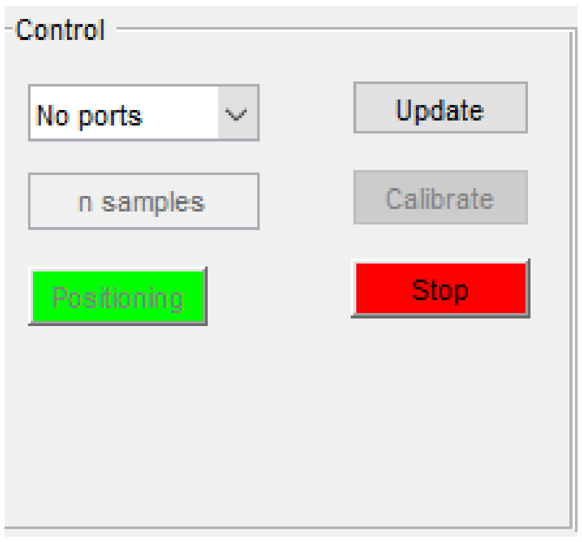


Figure 3: The control panel.

- The list of ports pop-up: indicates the currently connected COM ports. In case that nothing is connected, the execution of the code is not permitted. **Please make sure that the selected COM port is the one corresponding to the Arduino running the *positioning_arduino.ino* sketch**.
- The update button: as its name indicates, updates the available ports and enable the buttons if at least one is connected:

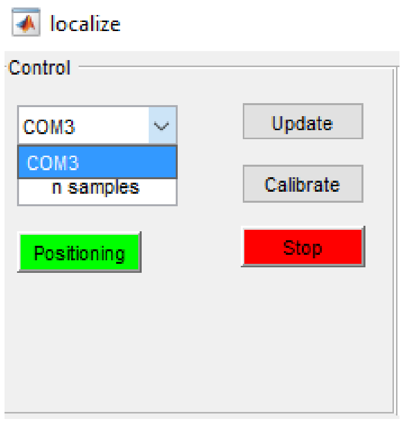


Figure 4: Setting the port after updating the list.

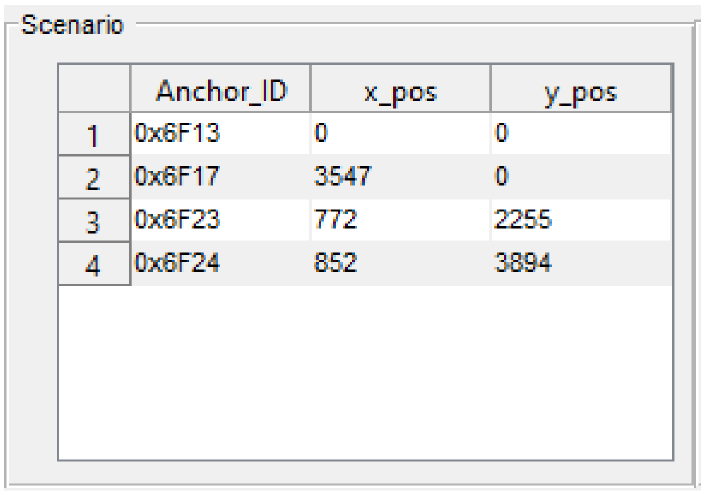
- n samples text box: this sets the number of samples that will be used for the localization algorithms. For the case of the Pozyx algorithm, this number will set the times that the device is localized (to calculate its mean). For the case of BP, this number will set the number of messages that the tag will exchange with each anchor. The default value is 20, so if nothing is changed the tag will determine 20 estimators to each anchor, being part of the input data for the algorithm (more info in 8. **Please make sure that only positive integers are considered**. Other values may arise unconsidered errors. Also notice that larger values produces longer calculation times.
- the calibrate button: it sends to the tag a command to automatically (by default) calibrate the position of the anchors. When the calibration is done the coordinates are plotted whit each anchor's identification. Also, it gets the position of the tag during the specified samples according to the Pozyx algorithm, showing the current progress. After gathering these positions as (x,y) coordinates in millimetres, it plots its mean and also the scenario table is updated (see 6). Once this process is successfully completed, both sections Anchor Data(8) and Algorithm's Comparison(9) will be available. **Please make sure that the devices are as fixed as possible during the calculation**.
- the positioning button: it has a very similar behaviour to the previous button, as the calibration of the anchors is done and plotted. However, it shows a real time positioning of the device during the number of samples

specified. Nevertheless, it does not provide data for the Algortihm's Comparison.

- the stop button: interrupts the execution of the code, for example in the case a very large number of samples was set and the process is wanted to be closed. Pressing this button will consider the calibration as unsuccessful and no comparison will be available. It is recommended pressing the update button afterwards to restart it as the interruption may produce some synchronization issues in the serial communication.

6 Scenario

This section summarizes the scenario **after a calibration** is performed. The table contains the id, x position and y position (in mm) for every anchor detected in the area. If the manual calibration was selected, these data will be the same that the introduced in the sketch:



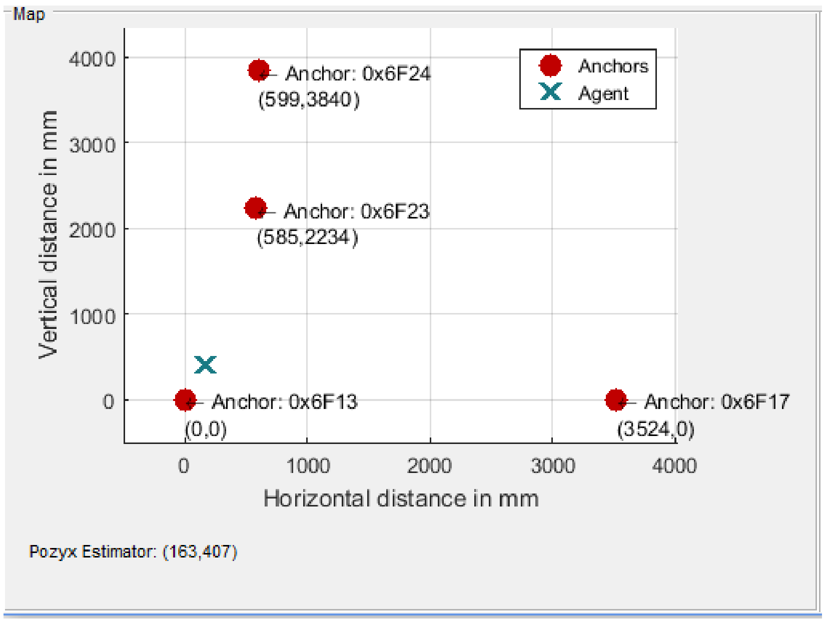
	Anchor_ID	x_pos	y_pos
1	0x6F13	0	0
2	0x6F17	3547	0
3	0x6F23	772	2255
4	0x6F24	852	3894

Figure 5: Table scenario after a 4 anchors calibration.

7 Map

This section is available for both the calibration and positioning functions. For the first case, it will basically plots the information of the scenario table, with the anchors as red dots (showing their ids), plus the position estimated by the Pozyx device as a blue x. A text below displays the estimated position.

For the positioning function, a number of x equal to the number of samples will be plotted in real time, but in this case the data is not valid for the algorithms (as is assumed no to be fixed).

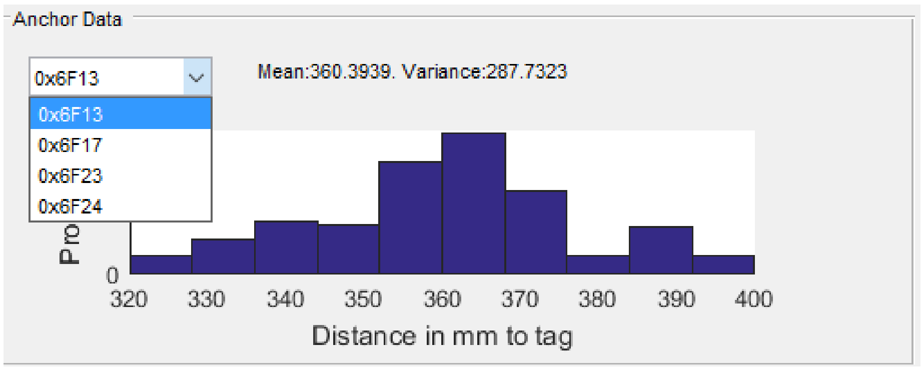


Pozyx Estimator: (163,407)

Figure 6: Map scenario after a 4 anchors calibration with the Pozyx estimated position.

8 Anchor Data

This section is available once the calibration is performed. Each anchor can be selected with a pop-up menu, showing a list of (number of samples) estimators to the agent. It is plotted the distribution of these distances (histogram), as well as its mean and variance. This will serve as the messages exchanged between the tag and the anchors.

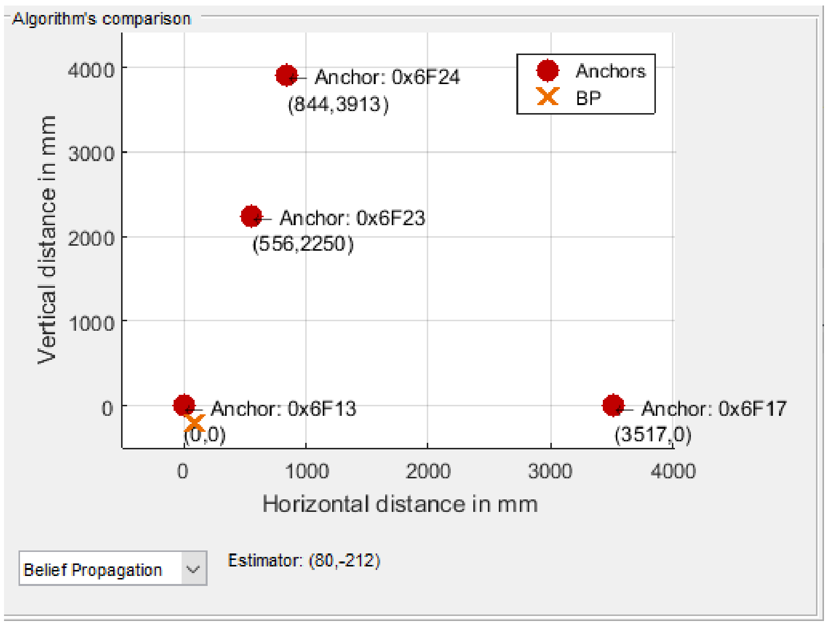


Mean:360.3939. Variance:287.7323

Figure 7: Anchor data: distribution of the distances between the tag and 0x6F13. Seems to be Gaussian.

9 Algorithms Comparison

This section is available once the calibration is performed. Each algortihm can be selected with a pop-up menu. For the Pozyx case (the default one), no calculation is required and it is the same the agent position in the map. For the BP, a calculation is required (which can take some time depending on the number of samples) and after finishing, the solution point is plotted and displayed.



Estimator: (80,-212)

Figure 8: Map scenario with the BP solution coordinates. It can be compared with the Pozyx one.

Note that the progress of this algorithm can be followed on the console. It will provide different results after each execution and some other parameters regarding these calculations may be set in the code, especially in the file *presentBP.m*:

```
% Sim Data
[simData.NSAMPLES, ~] = size(simTopology.msg);

simData.SIGMA = 0.1;
simData.ERROR_TYPE = 'Gaussian'; % 'Gaussian', 'Exponential
    ', 'Uniform'
simData.NUM_ITERATIONS = 3;
simData.SAMPLE_TECHNIQUE = 'Importance'; % 'Importance', '
    Mixture', 'Gibbs'
simData.K_MIXTURE_PARAMETER = 300; % Only in mixture: Max
    value is the number of devices + 1 is connected a device
simData.RESAMPLE_OPTION = 0; % Only in Importance: 0-yes, 1-
```

```
    no
simData.MULTIPLICATION_OPTION = 0; % 0-2on2 1-all together
simData.BMS_OPTION = 0; %Bandwidth Matrix Selector from 0 to
     16
simData.SIMULATION_ITERATIONS = 1;
```

10 Future Work

Once the program has been presented, it has been observed that many functionalities can be implemented with the interface.

First thing would be to add more algorithms. As two of them have already been implemented, it seems easy to provide more of them if the code is already programmed. The only matter is to adapt it in order to present it onto the handles structure.

Also, it would be interesting to provide a tool to manually set the coordinates of the anchors, either using the table or by clicking the map.

The option of multiple agents can be considered too, but omitting the fact the agent provides its coordinates. Nevertheless, the library only supports a maximum of 6 anchors so not really complex scenarios would be considered though. Dealing with devices that are not in coverage between them presents a difficult challenge, as the devices should be at a physical distance of more than 100 m.

Also, some synchronization issues have been observed, as the data is read faster in some computers. If there is no data to be read, the code waits a time until it is again available, defining a buffer. However, it has been observed some unsuitability with these waiting times. Improve this buffer to work correctly in every scenario and computer is an important issue to be solved.

Finally, it is suggested to do experiments using this interface, studying the behaviour of the algorithms under different circumstances.

References

[1] Arduino support from matlab. `https://de.mathworks.com/hardware-support/arduino-matlab.html`. Accessed: 7/10/2016.

[2] Arduino getting started. `https://www.pozyx.io/Documentation/Tutorials/gettingStarted`. Accessed: 7/10/2016.

[3] Jaime Lien Henk Wymeersch and Moe Z. Win. *Cooperative Localization in Wireless Networks*. IEE, vol. 97 edition, 2009.

[4] Arduino tutorial: ready to localize. `https://www.pozyx.io/Documentation/Tutorials/ready_to_localize`. Accessed: 7/10/2016.