Guilherme Klink

Trilateration with Kilobots

1 ROBOTS

In order to perform the strategy four types of robots with a fixed function according to its ID were used.

ID	*Function*	*Tx Color*	*Rx Color*
1	Coordinate frame origin	White	-
2	X-axis vector	Blue	Yellow
3	3th robot of the axis	Green (non-debug)	Yellow (1) / Purple (2)
4	Follower	-	-

The **coordinate frame origin robot** marks the origin of the local axis that will be built, it blinks white whenever a message is sent and it does not need to receive any message. The **x-axis vector robot** will delimitate the x-axis of the frame aligned with the coordinate frame origin robot, it turns blue whenever a message is transmitted and yellow whenever it receives and handle a message from the origin. The **3th robot of the axis** receives and handles messages from both the origin (turns yellow) and the x-axis robot (turns purple). The **follower robot** handles messages received from all the other robots, but do not transmit any message.

2 THE MESSAGE

A standard message is transmitted by the three transmitter robots. The array **message.data[9]** has fixed indexes and is defined as in the table below.

Index	*Data*	*Robots that fill this field*
0	Sender's ID	1, 2, 3
1	A flag that tells if the coordinates of the sender were already calculated	1, 2 , 3
2	The X coordinate of the sender	1, 2, 3
3	The Y coordinate of the sender	1, 2, 3
4	The distance between the origin and x-axis robot	2, 3
5	The distance between the origin and the 3th robot of the frame	3
6	The distance between the x-axis robot and the 3th robot of the frame	3
7	Not implemented	-
8	Not implemented	-

The third column of the table tells which robots are able to fill the respective field through internal calculation and/or receiving messages from other robots. Note that the 3th robot of the frame is the only one able to fill all the implemented fields.

3 CALCULATING ROBOTS COORDINATES

3.1 ROBOTS' COORDINATES

ID	*X*	*Y*
1	0	0
2	Distance between 1 and 2	0
3	$\frac{d_{13}^2 - d_{23}^2 + d_{12}^2}{2 * d_{12}}$	$\sqrt[2]{d_{13}^2 - x_3^2}$
4	$\frac{d_1^2 - d_2^2 + d_{12}^2}{2 * d_{12}}$	$\frac{d_1^2 - d_3^2 + x_3^2 + y_3^2}{2 * y_3} - \frac{x_3 * x_4}{y_3}$

3.2 COORDINATE FRAME ORIGIN ROBOT

This robot will be the origin, therefore its coordinates will be 0.

3.3 X-AXIS ROBOT

This robot will be in the x-axis so its y-coordinates will be 0 and it's x-coordinate the distance from the origin robot (by convention it will be on the positive side), that can be calculated once a message is received from the last.

3.4 3TH ROBOT FROM THE FRAME

The coordinates from this robot can be calculated once it receives a message from the origin and the x-axis robot by using triangle equations as shown in figure 1.

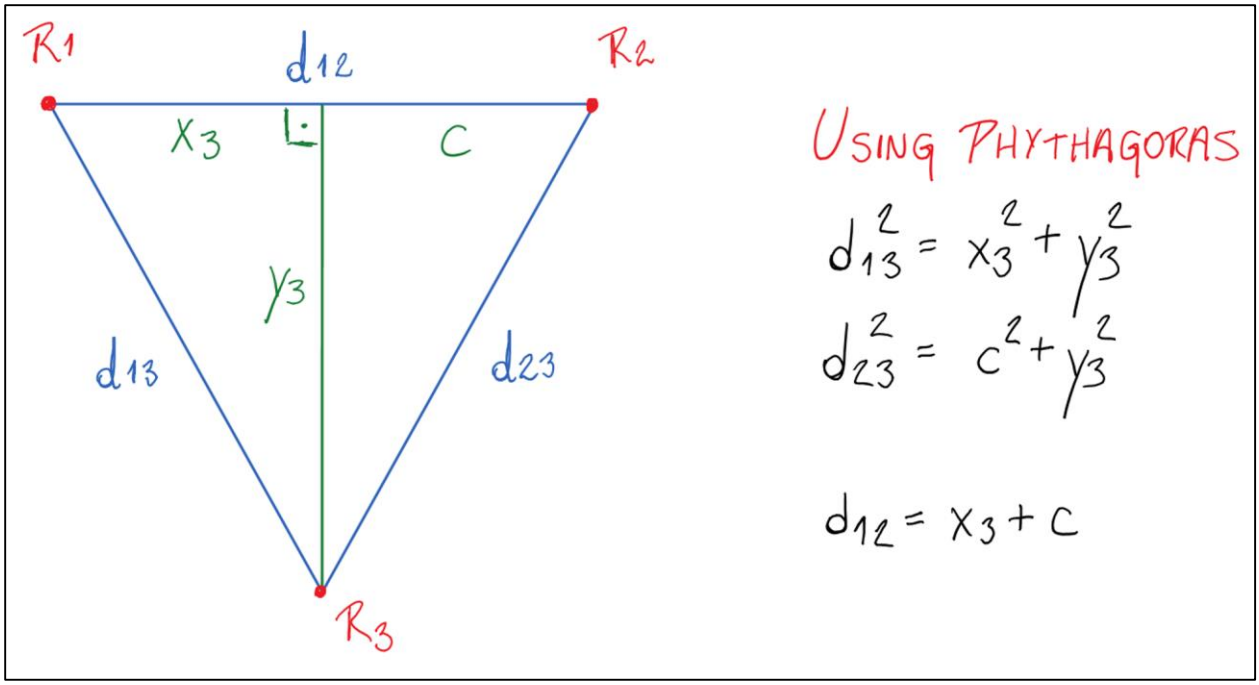

Figure 1

In the figure above $\boldsymbol{R_i}$ are the 3 robots, $\boldsymbol{d_{ij}}$ is the distance from robot ***i*** to robot ***j*** and $\boldsymbol{x_3}, \boldsymbol{y_3}$ are the coordinates of $\boldsymbol{R_3}$. The equations shown in the **Coordinate Table** can be obtained by solving the 3 equations from *figure 1*.

3.5 FOLLOWER

The follower robot coordinates can be obtained through trilateration once the follower bot receive a message from each other robot. Three flags are used to guarantee that the trilateration will not be attempt before receiving all the messages (*received_msg_1, received_msg_2, received_msg_3*).

$$x_4 = \frac{d_1^2 - d_2^2 + d_{12}^2}{2 * d_{12}} \qquad y_4 = \frac{d_1^2 - d_3^2 + x_3^2 + y_3^2}{2 * y_3} - \frac{x_3 * x_4}{y_3}$$

x_3: X-coordinate of the 3th robot of the axis

y_3: Y-coordinate of the 3th robot of the axis

x_4: X-coordinate of the follower

y_4: Y-coordinate of the follower

d_1: Distance from the follower to the origin robot

d_2: Distance from the follower to x-axis robot

d_3: Distance from the follower to 3th robot from the axis

d_{12}: Distance between the origin and the x-axis robot

Note: the derivation of the trilateration formulas can be seem at *https://en.wikipedia.org/wiki/Trilateration*

3.6 STEERING THE ROBOT

Due to the lack of sensors on the kilobot, calculating an angle to turn is not an easy task. A random kilobot was selected and the time it took to complete different angles was recorded. The data was used to create a linear equation $\boldsymbol{f}$ of the type $\boldsymbol{t} = \boldsymbol{f}(\boldsymbol{angle})$, where $\boldsymbol{t}$ is the time in seconds that the kilobot needs to turn in order to perform the input $\boldsymbol{angle}$. This time is converted to milliseconds and the motor of the robot is kept turned on for that amount of time in order to rotate approximately to the desired angle.

4 MOVING TO THE GOAL

First, the 3 robots composing the axis need to calculate their coordinates, once that is done, what is signalized by the second field of the message (see **The Message**), the follower robot uses the messages received from each axis robot, trilaterates its position and calculate the goal and set the flag ***isGoalCalculated*** so a step can be taken.

4.1 CALCULATING THE GOAL

The goal will be the center of the triangle formed by the 3 axis robots. To calculate this the following equation will be used:

$$Center = \left(\frac{\sum_1^3 x_i}{3}, \frac{\sum_1^3 y_i}{3}\right)$$

- x_i: x-coordinates of robot *i*
- y_i: y-coordinates of robot *i*

4.2 TAKING A STEP

Once the robot move a vector can be created using its new position and the past, **vector u**, if it is the first time the robot moves a small movement will be taken before it starts the calculations. Taking a step consists of turning the angle between this vector and the vector created from the current position to the goal, **vector v** (watch video 1). It was observed that if the robot give a little step forward after every turn it will achieve the goal faster and with a better trajectory (watch video 2 and 3). The following formula is used to calculate the angle of turning:

$$\propto = atan2(u1 * v2 - u2 * v1, u1 * v1 + u2 * v2)$$

This equation generates a value in the interval [-π ,+π] and will be multiplied by 180/ π to be transformed from radians to degrees. A negative angle generate a left turn and a positive angle will generate a right turn.

Note: Because the value needs to be between [-π ,+π], the drawback of this equation is that by itself it will not be able to calculate angles when back of the robot is facing the goal.

Once the step is finish the flag ***isGoalCalculated*** is set back to 0, so a new step will only be taken once the robot's new position is trilaterated.

5 CONCLUSION

Creating a local coordinate frame to trilaterate positions can be an extremely useful technique to locate robots that lack of more complex sensors for global positioning, for example a GPS (which trilaterates positions using satellites), such is the case of the Kilobot. My project shows how a swarm can use this technique applying modules to guide other modules to an arbitrary location.

The biggest challenge encountered was due the lack of sensors capable of control the rotation of the kilobot, such as a magnetometer. That created the necessity of implementing a linear equation by timing the robot turn for certain known angles. Although, this is an acceptable workaround it still lacks on precision and due to small hardware differences among each robot, a different equation needs to be manually calculated for each new kilobot.