Robobo Line Following Program with ROS

Ângela Cardoso

Master in Informatics and Computing Engineering
Faculty of Engineering of the University of Porto

January 5, 2018

Overview

Robobo

Robobo

- Developed by Mytechia

Robobo

- Developed by Mytechia
- Two wheels and a plastic caster

Robobo

- Developed by Mytechia
- Two wheels and a plastic caster
- 8 infrared sensors and 7 LEDs

Robobo

- Developed by Mytechia
- Two wheels and a plastic caster
- 8 infrared sensors and 7 LEDs
- Phone holder with pan and tilt movements

Robobo

- Developed by Mytechia
- Two wheels and a plastic caster
- 8 infrared sensors and 7 LEDs
- Phone holder with pan and tilt movements
- Phone connects to base via Bluetooth

Robobo

- Developed by Mytechia
- Two wheels and a plastic caster
- 8 infrared sensors and 7 LEDs
- Phone holder with pan and tilt movements
- Phone connects to base via Bluetooth
- Phone connects to computer via Wi-Fi

Robobo

- Developed by Mytechia
- Two wheels and a plastic caster
- 8 infrared sensors and 7 LEDs
- Phone holder with pan and tilt movements
- Phone connects to base via Bluetooth
- Phone connects to computer via Wi-Fi
- Phone sensors: cameras, gyroscope, accelerometer

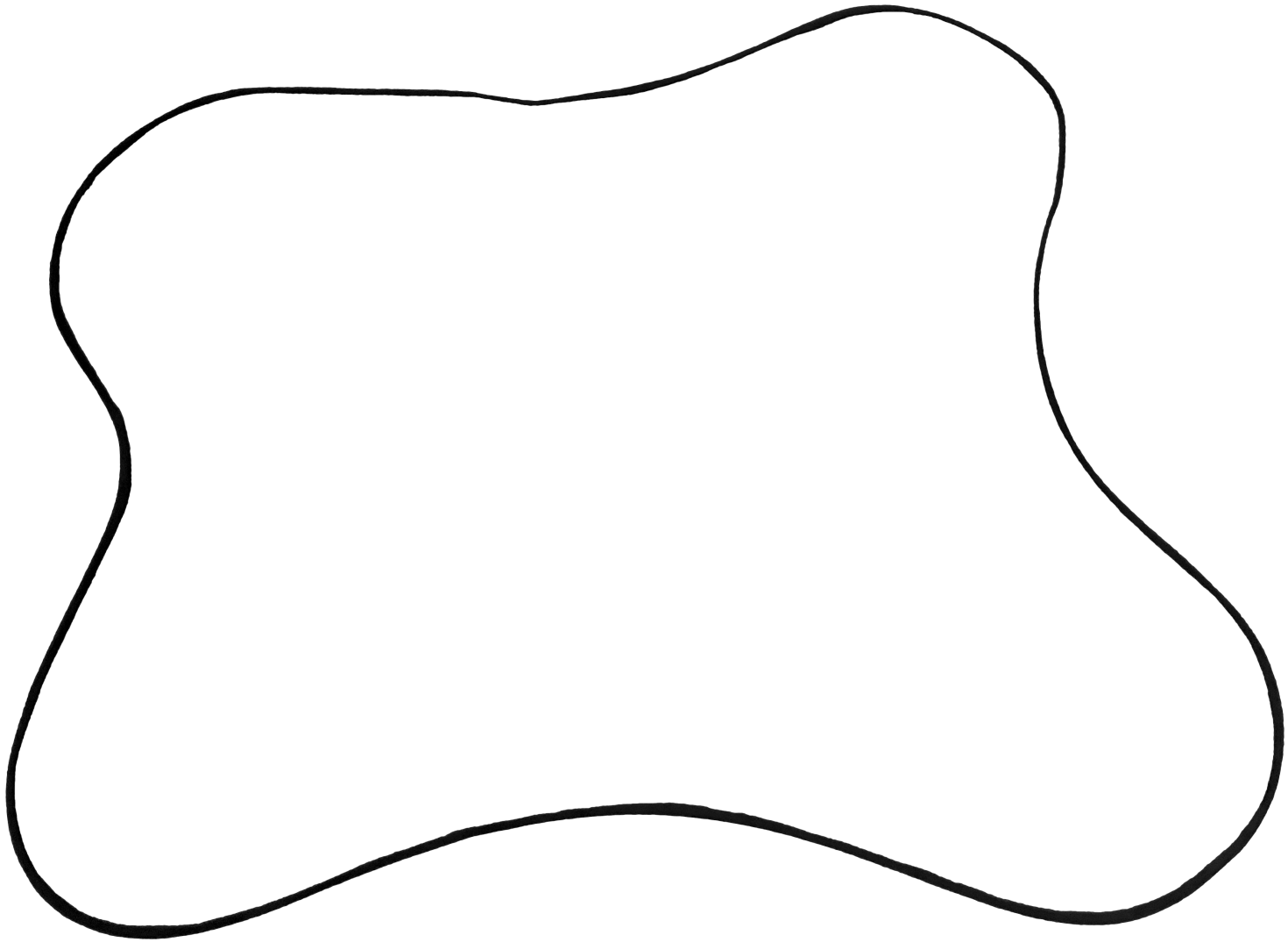
Robobo

- Developed by Mytechia
- Two wheels and a plastic caster
- 8 infrared sensors and 7 LEDs
- Phone holder with pan and tilt movements
- Phone connects to base via Bluetooth
- Phone connects to computer via Wi-Fi
- Phone sensors: cameras, gyroscope, accelerometer
- Phone actuators: display, sound

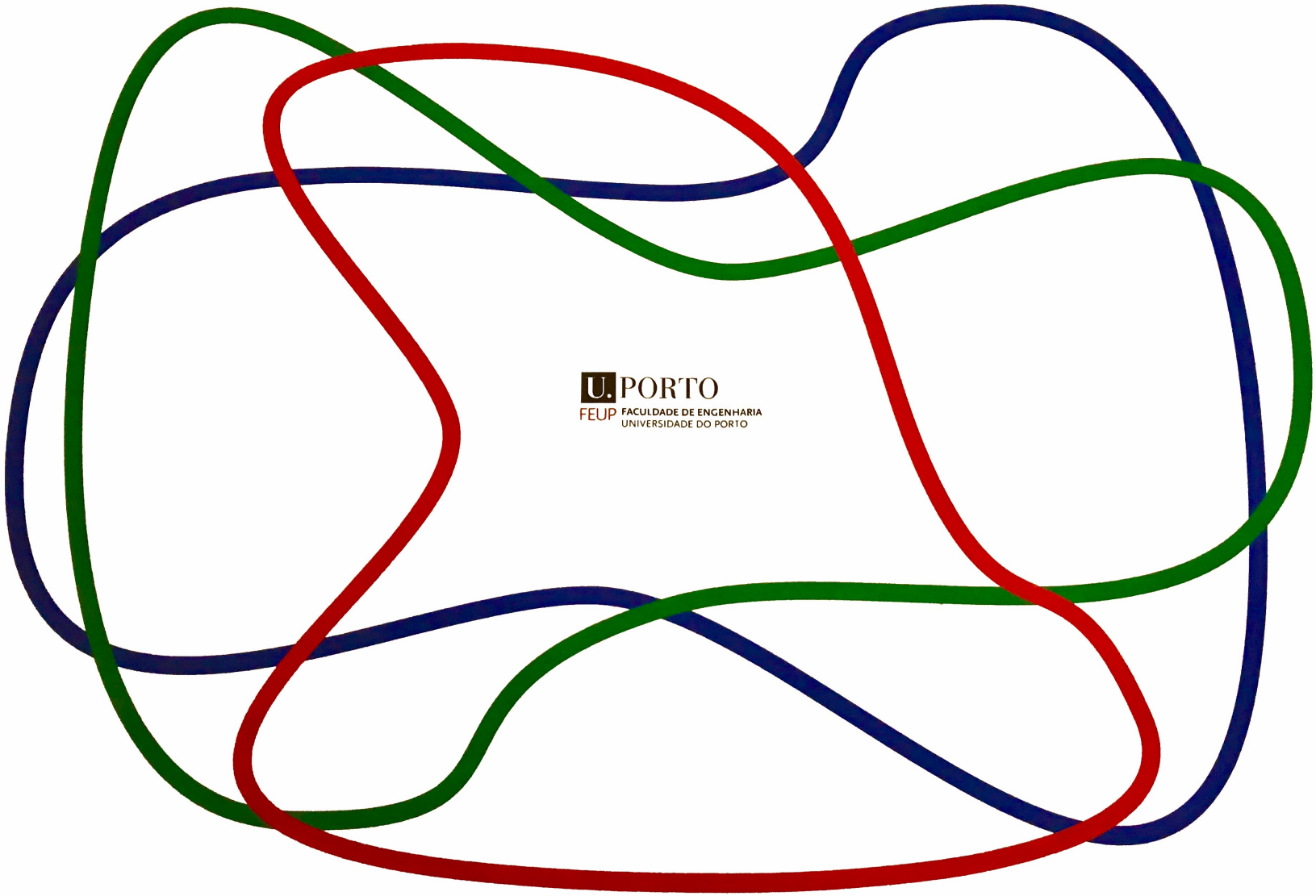
Robobo

- Developed by Mytechia
- Two wheels and a plastic caster
- 8 infrared sensors and 7 LEDs
- Phone holder with pan and tilt movements
- Phone connects to base via Bluetooth
- Phone connects to computer via Wi-Fi
- Phone sensors: cameras, gyroscope, accelerometer
- Phone actuators: display, sound
- Programmable using ScratchX, Java and ROS

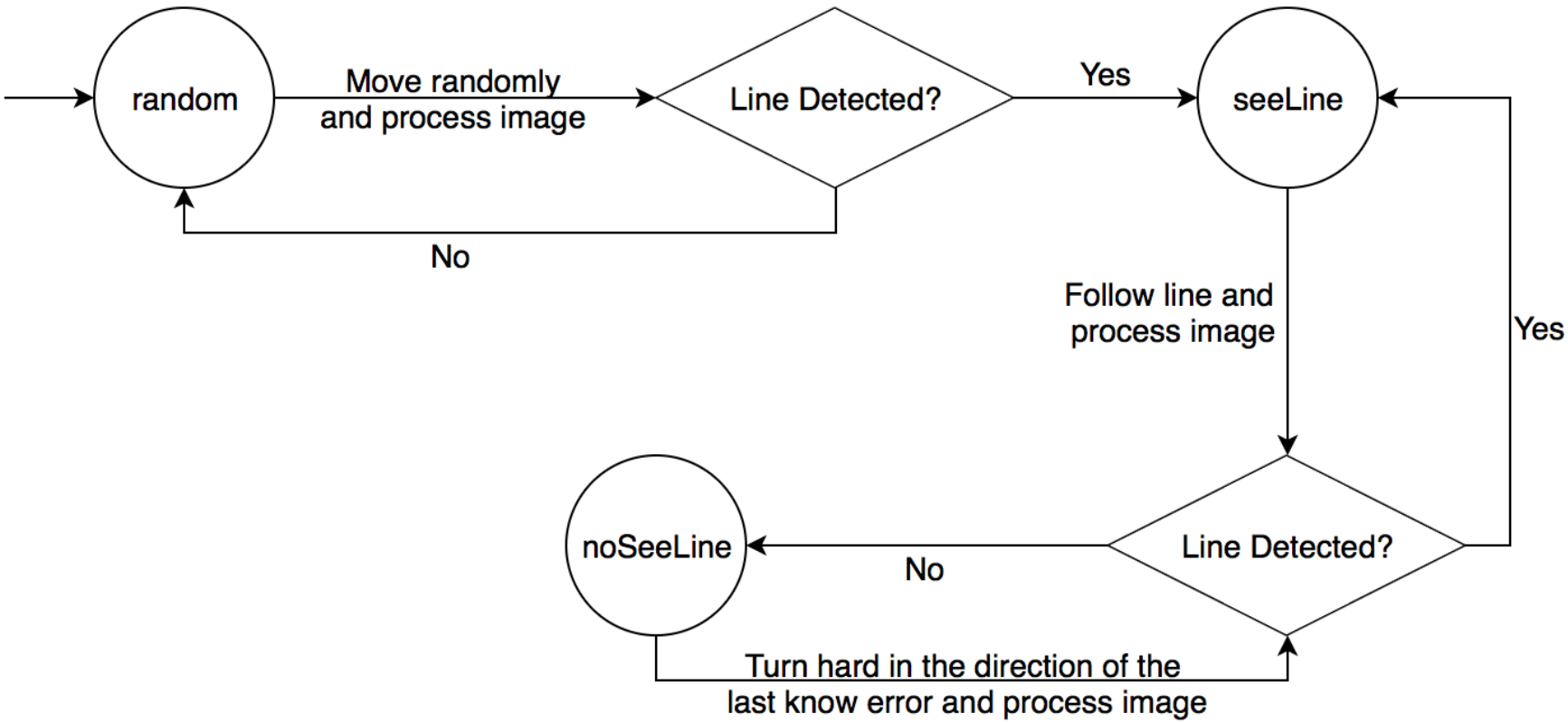
Maps



Maps

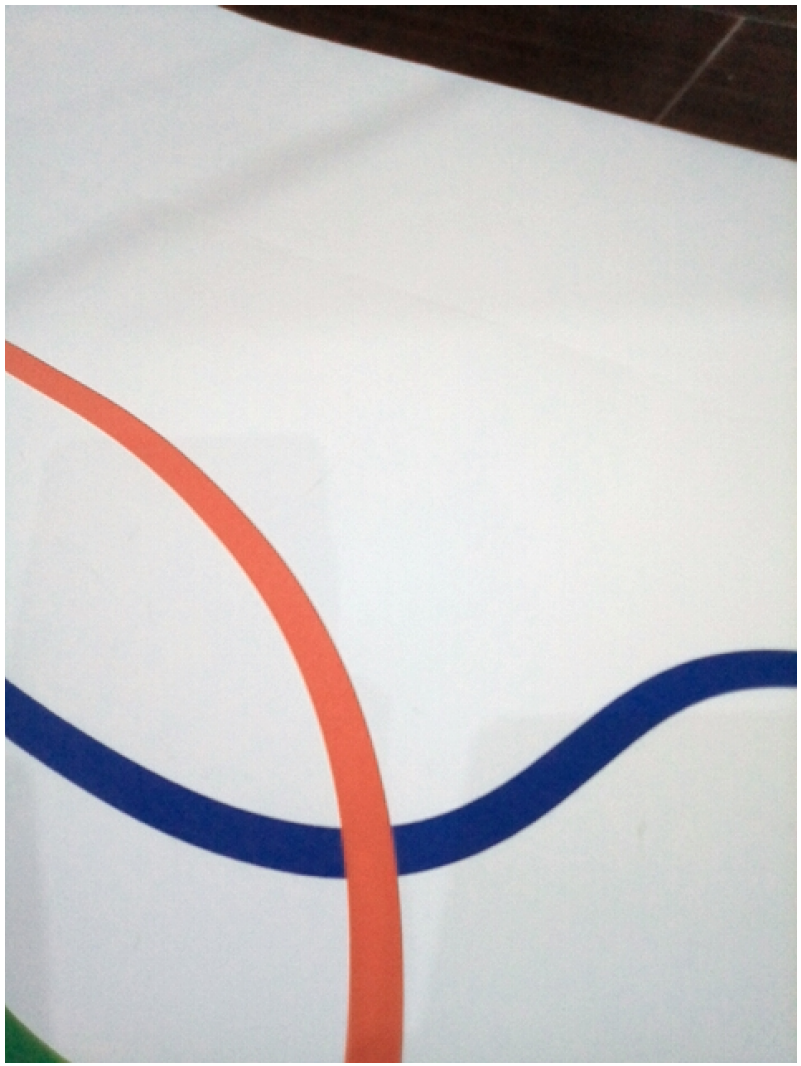

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Behavior


random
Move randomly
and process image
Line Detected?
Yes
seeLine
No
Follow line and
process image
Yes
noSeeLine
No
Line Detected?
Turn hard in the direction of the
last know error and process image

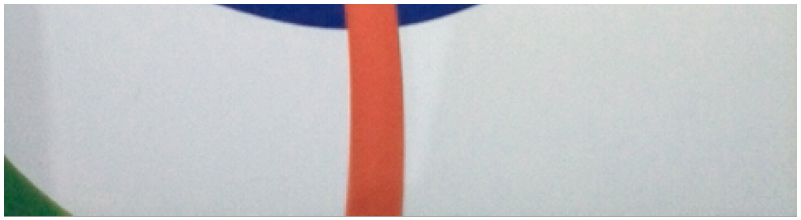
Camera Image Processing

1. Receive camera image



Camera Image Processing

1. Receive camera image
2. Select Region Of Interest (ROI)



Camera Image Processing

1. Receive camera image
2. Select Region Of Interest (ROI)
3. Create mask where only the color the robot should follow is selected
4. Make all pixels that do not belong to the mask white

Camera Image Processing

1. Receive camera image
2. Select Region Of Interest (ROI)
3. Create mask where only the color the robot should follow is selected
4. Make all pixels that do not belong to the mask white
5. Transform the ROI from color to gray scale

Camera Image Processing

1. Receive camera image
2. Select Region Of Interest (ROI)
3. Create mask where only the color the robot should follow is selected
4. Make all pixels that do not belong to the mask white
5. Transform the ROI from color to gray scale
6. Blur the gray scale ROI

Camera Image Processing

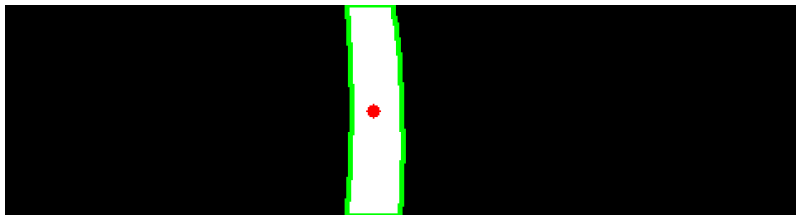
1. Receive camera image
2. Select Region Of Interest (ROI)
3. Create mask where only the color the robot should follow is selected
4. Make all pixels that do not belong to the mask white
5. Transform the ROI from color to gray scale
6. Blur the gray scale ROI
7. Compute threshold of blurred ROI

Camera Image Processing

1. Receive camera image
2. Select Region Of Interest (ROI)
3. Create mask where only the color the robot should follow is selected
4. Make all pixels that do not belong to the mask white
5. Transform the ROI from color to gray scale
6. Blur the gray scale ROI
7. Compute threshold of blurred ROI
8. Find contours and select the contour closest to the center

Camera Image Processing

1. Receive camera image
2. Select Region Of Interest (ROI)
3. Create mask where only the color the robot should follow is selected
4. Make all pixels that do not belong to the mask white
5. Transform the ROI from color to gray scale
6. Blur the gray scale ROI
7. Compute threshold of blurred ROI
8. Find contours and select the contour closest to the center
9. Compute the centroid of that contour using image moments



Camera Image Processing

1. Receive camera image
2. Select Region Of Interest (ROI)
3. Create mask where only the color the robot should follow is selected
4. Make all pixels that do not belong to the mask white
5. Transform the ROI from color to gray scale
6. Blur the gray scale ROI
7. Compute threshold of blurred ROI
8. Find contours and select the contour closest to the center
9. Compute the centroid of that contour using image moments
10. Use the centroid to determine the signed normalized error

Line Following

1. Compute angular velocity using a PID controller:

$$\omega = K_p * P + K_i * I + K_d * D,$$

K_p, K_i, K_d: proportional, integral and derivative constants
P, I, D: proportional, integral and derivative terms
$P = e, I = I + e, D = e - d$: e current error, d previous error

Line Following

1. Compute angular velocity using a PID controller:

$$\omega = K_p * P + K_i * I + K_d * D,$$

K_p, K_i, K_d: proportional, integral and derivative constants
P, I, D: proportional, integral and derivative terms
$P = e, I = I + e, D = e - d$: e current error, d previous error

2. Send move command to each wheel, with left wheel speed $v - \omega$ and right wheel speed $v + \omega$, for a linear velocity v.

Line Following

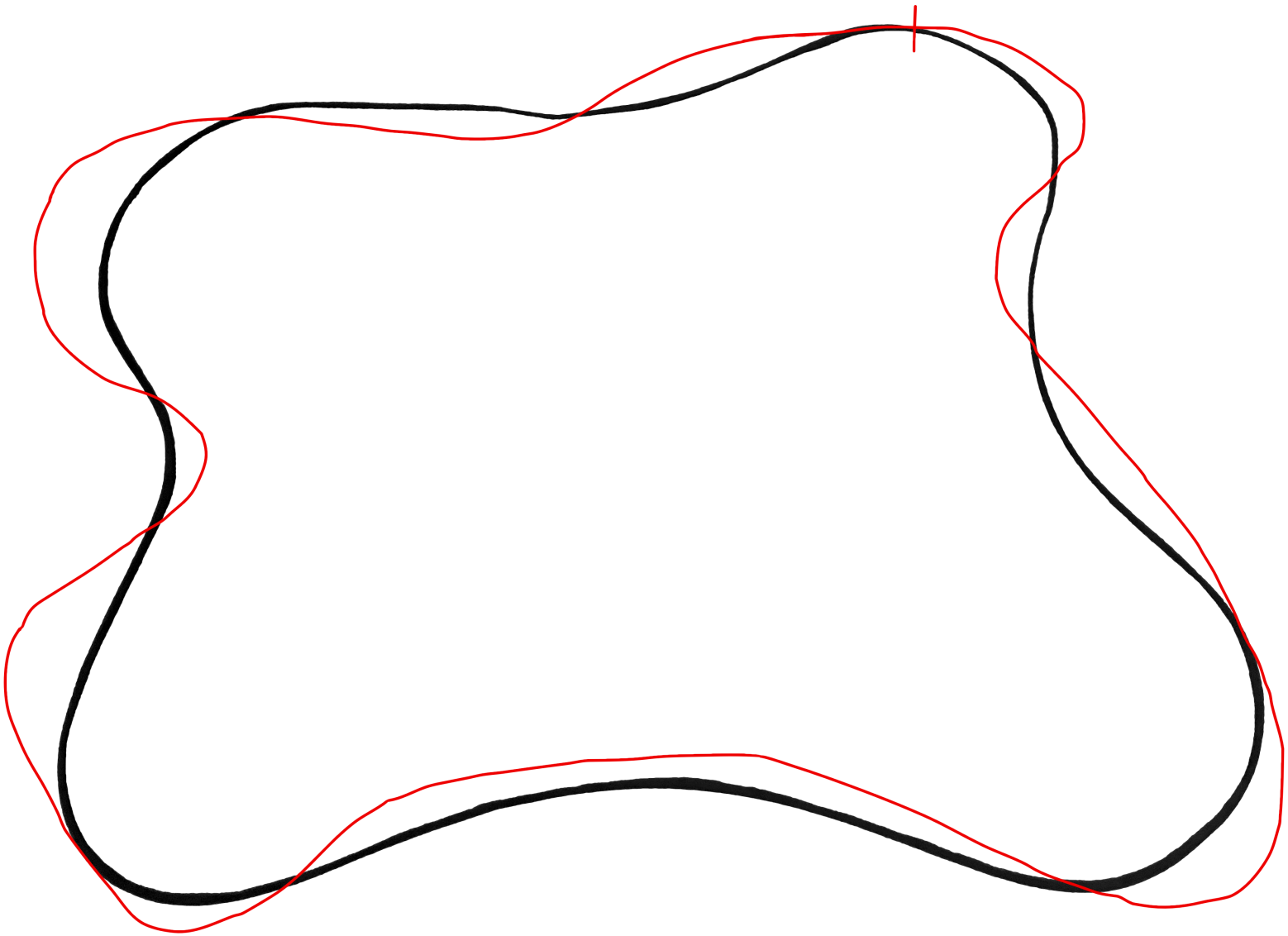
1. Compute angular velocity using a PID controller:

$$\omega = K_p * P + K_i * I + K_d * D,$$

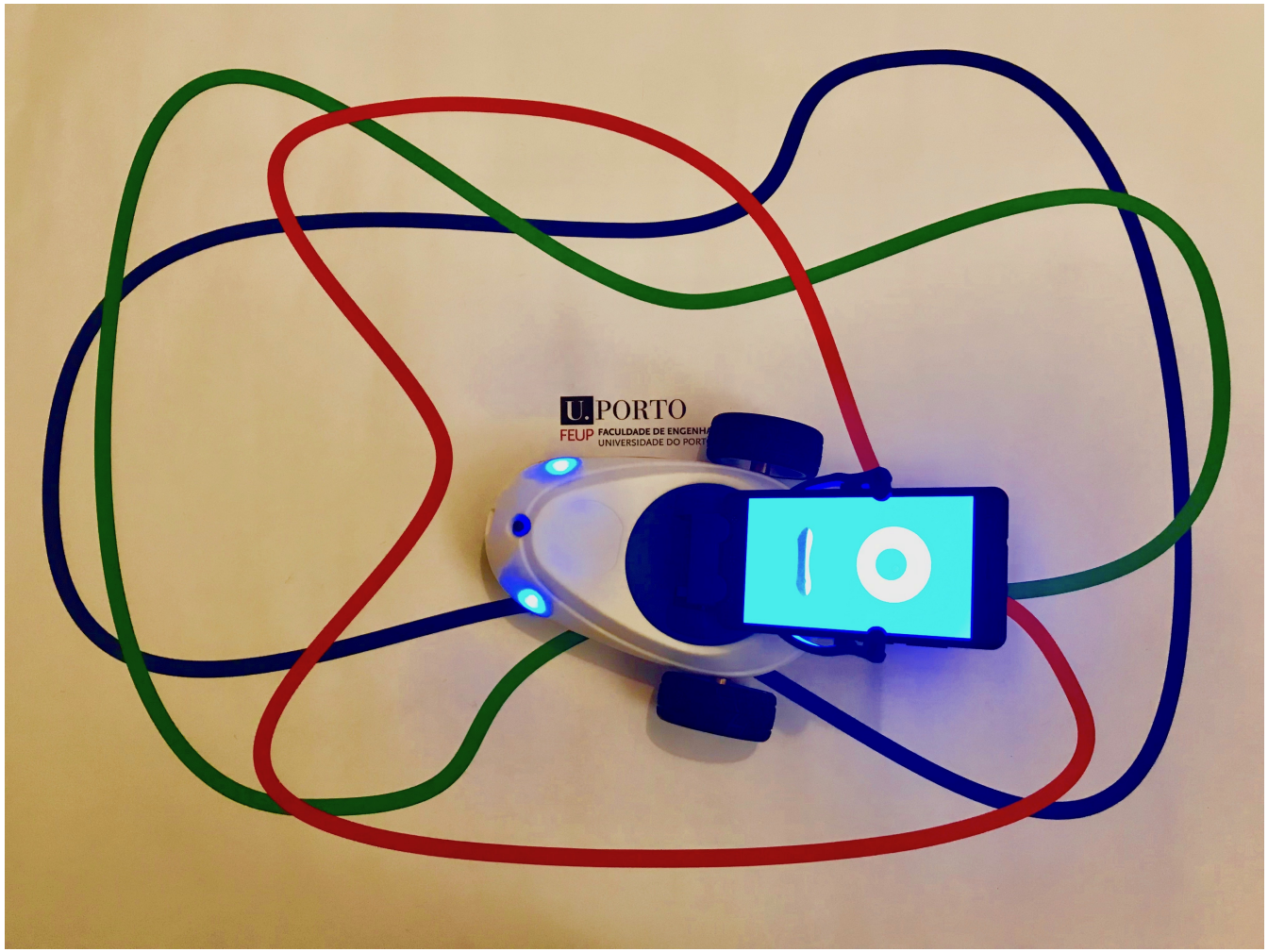
K_p, K_i, K_d: proportional, integral and derivative constants
P, I, D: proportional, integral and derivative terms
$P = e, I = I + e, D = e - d$: e current error, d previous error

2. Send move command to each wheel, with left wheel speed $v - \omega$ and right wheel speed $v + \omega$, for a linear velocity v.
3. After tunning, the constants used where $K_p = 8, K_i = 0, K_d = 8$, with a linear velocity of 8.

Black Map with Pencil



Colored Map Video



Limitations

- Initial robot position

Limitations

- Initial robot position
- Camera image width

Limitations

- Initial robot position
- Camera image width
- Linear velocity

Conclusion

- The PID controller, although hard to tune provides a stable movement

Conclusion

- The PID controller, although hard to tune provides a stable movement
- The camera image processing is the most challenging, but also the most rewarding part

Conclusion

- The PID controller, although hard to tune provides a stable movement
- The camera image processing is the most challenging, but also the most rewarding part
- For real world projects, sometimes one must use tricks to overcome some of the limitations

Conclusion

- The PID controller, although hard to tune provides a stable movement
- The camera image processing is the most challenging, but also the most rewarding part
- For real world projects, sometimes one must use tricks to overcome some of the limitations
- Using ROS to program Robobo is pretty much the same as programming any other robot

Bibliography


L. K. G. at the MIT Media Lab. Scratch, 2005. [https://scratch.mit.edu, Accessed 3-January-2018].


Enigmerald. PID tutorials for line following, 2014. [https://www.robotshop.com/letsmakerobots/pid-tutorials-line-following, Accessed 3-January-2018].


O. S. R. Foundation. ROS, 2007. [http://www.ros.org, Accessed 3-January-2018].


W. G. Intel and Itseez. OpenCV, 1999. [https://opencv.org, Accessed 3-January-2018].


Mytechia. Robobo, 2016. [http://en.therobobоproject.com, Accessed 3-January-2018].


Mytechia. Robobo programming, 2016. [https://bitbucket.org/mytechia/robobo-programming/wiki/Home, Accessed 3-January-2018].


J. M. O'Kane. *A Gentle Introduction to ROS*. Independently published, Oct. 2013. [http://www.cse.sc.edu/~jokane/agitr/, Accessed 3-January-2018].


Wikipedia. PID controller, 2002. [https://en.wikipedia.org/wiki/PID_controller, Accessed 3-January-2018].


Wikipedia. Image moment, 2005. [https://en.wikipedia.org/wiki/Image_moment, Accessed 3-January-2018].