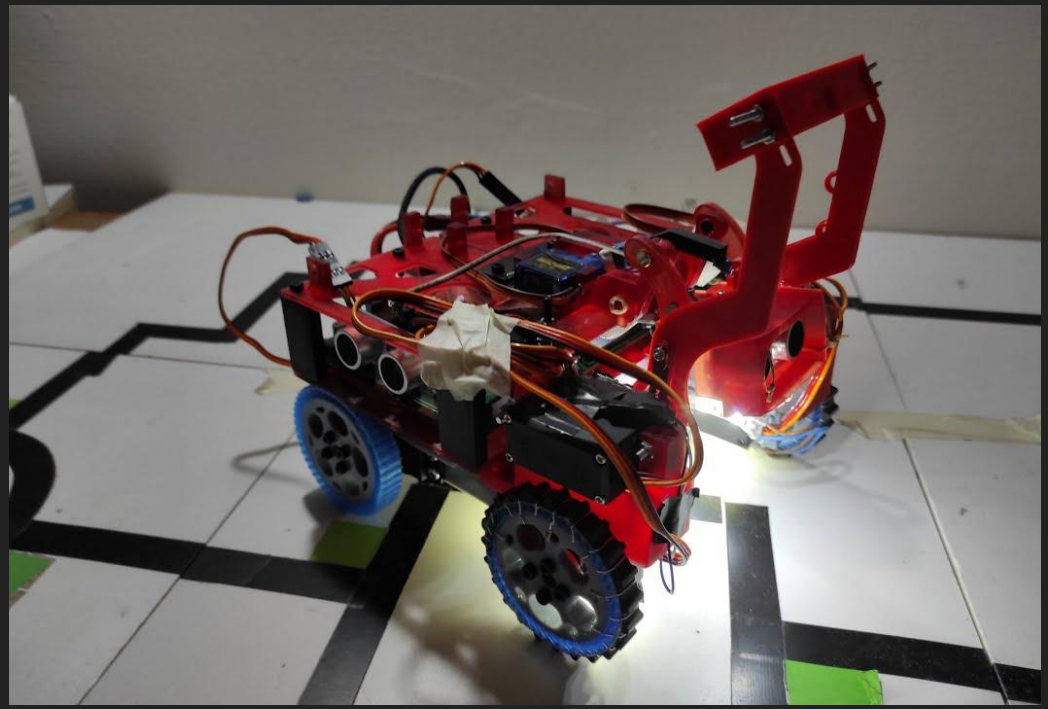
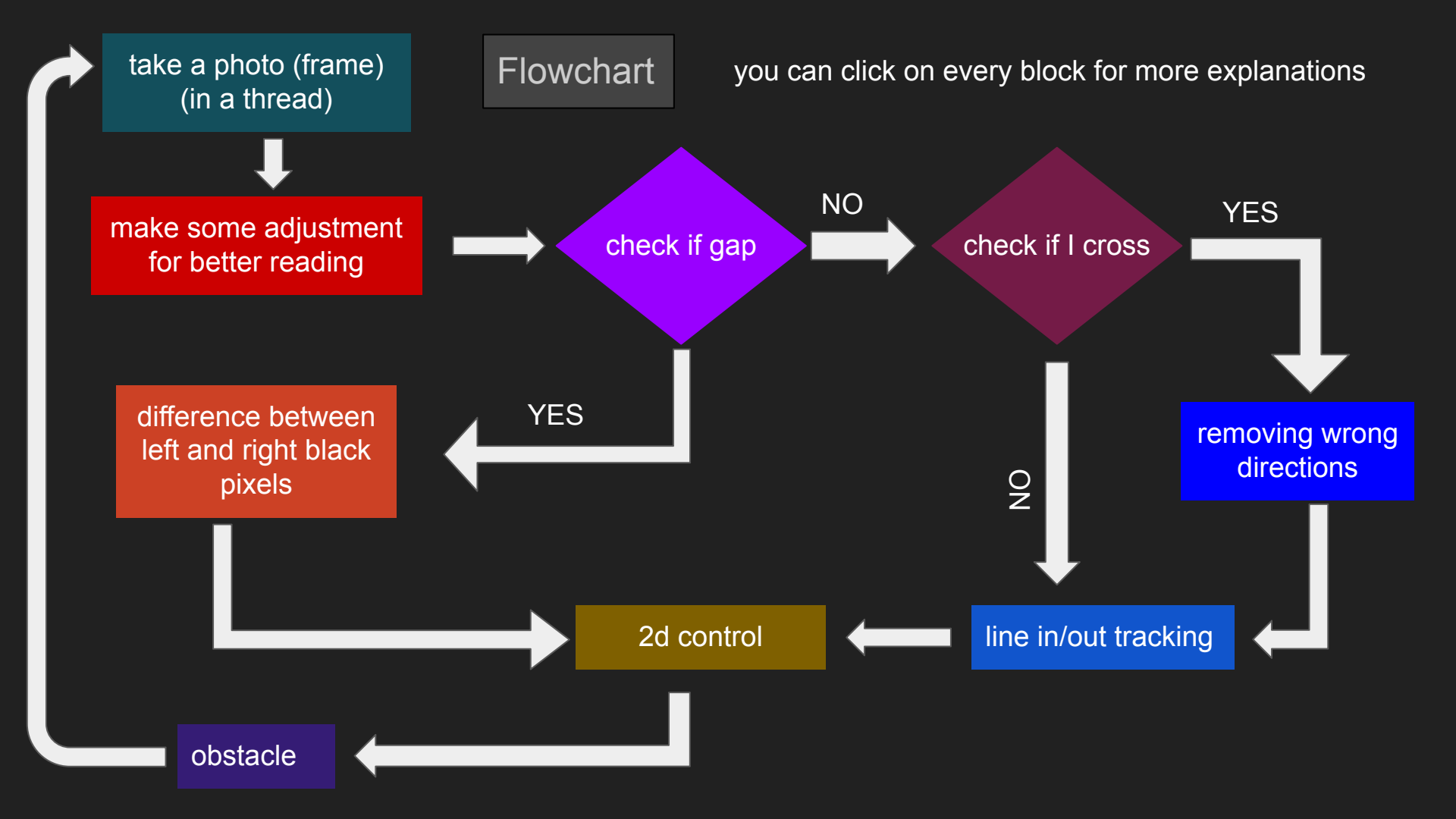
LINE FOLLOWER SOFTWARE

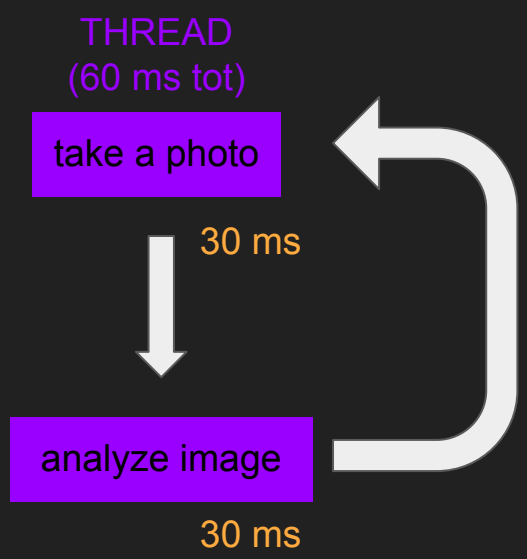


Click here for the video


take a photo (frame)
(in a thread)
Flowchart
you can click on every block for more explanations
make some adjustment
for better reading
check if gap
NO
check if I cross
YES
YES
difference between
left and right black
pixels
NO
removing wrong
directions
2d control
line in/out tracking
obstacle

take a photo (frame)
(in a thread)

single threading



if I use only a thread when I decide to take a new photo I need to wait for about 30 ms to have it ready

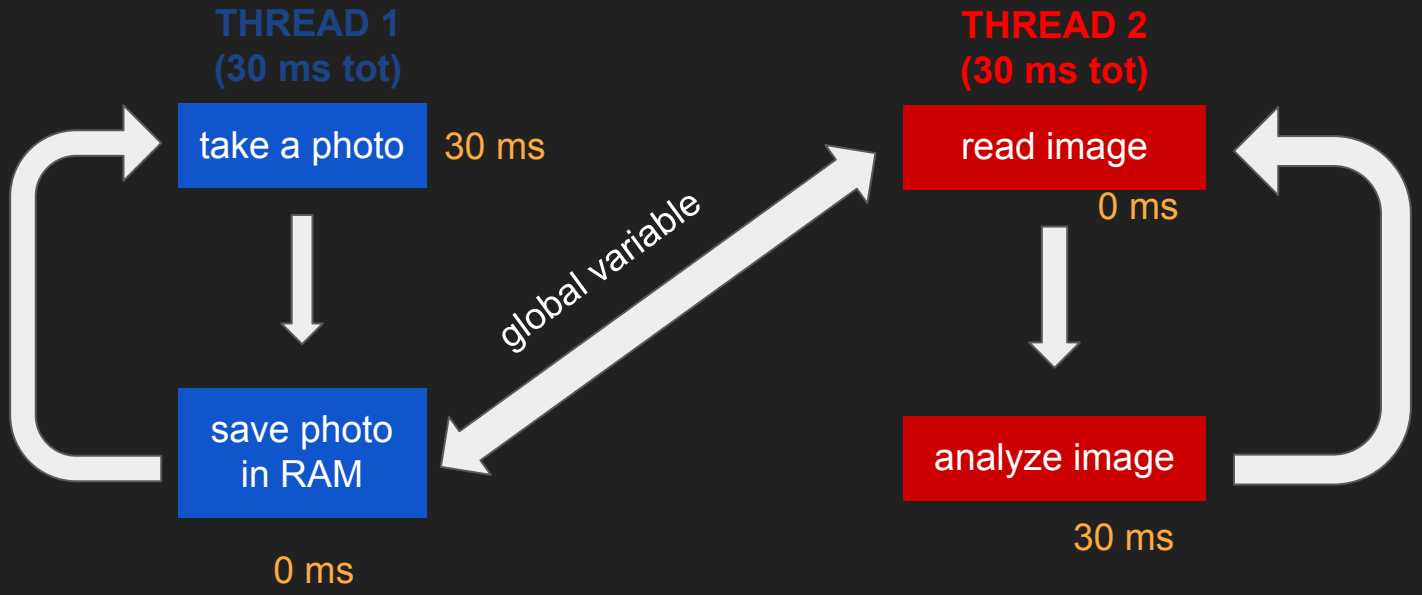
-taking a photo takes around 30 ms (at 30 fps)
-analysing the photo (30 ms)

60 ms total

1/0.06 = 17 fps

back to flowchart

multi threading



If the robot uses 2 threads it can have one cycle that is updated every 30 ms to take photos, so it can get access to the last photo shooted in a really short time

-read a photo (0 ms)
-analysing the photo (30 ms)

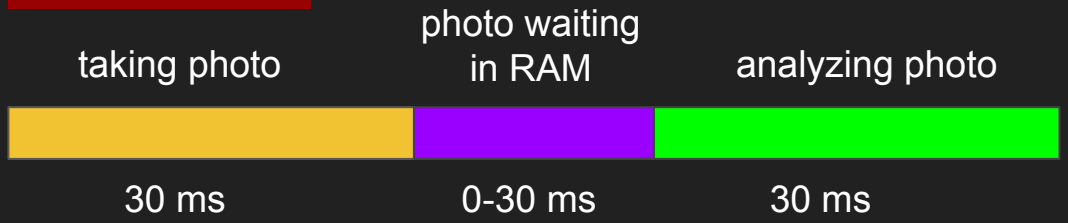
30 ms total

1/0.03 = 33 fps

back to flowchart

single threading

multi threading

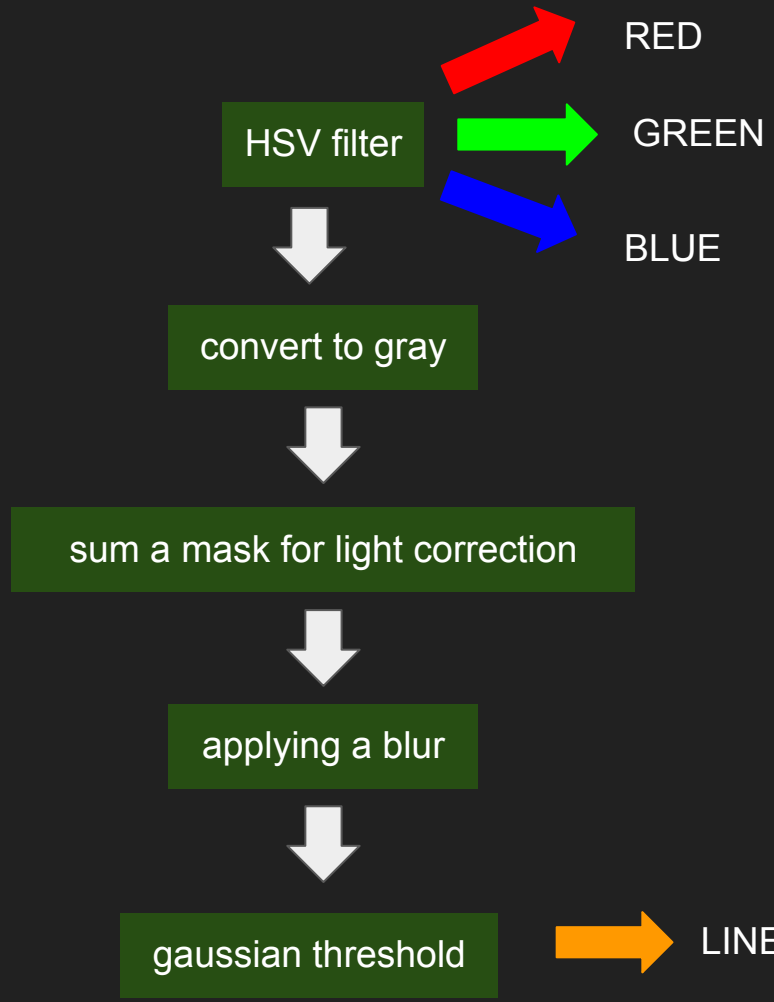


AVERAGE INPUT LAG

Using multi threading the average input lag increases by up to 50%, but the average frame rate doubles so it’s worth.

back to flowchart

make some adjustment for better reading



back to flowchart

applying a blur

We apply a blur to make the image "smoother" and to remove lines between tiles.

```
img2=cv2.medianBlur(img2,5)
```

pre blur

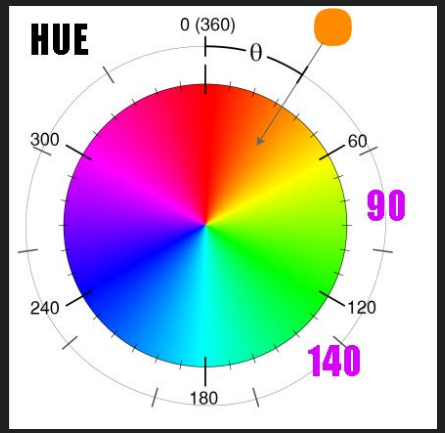
post blur

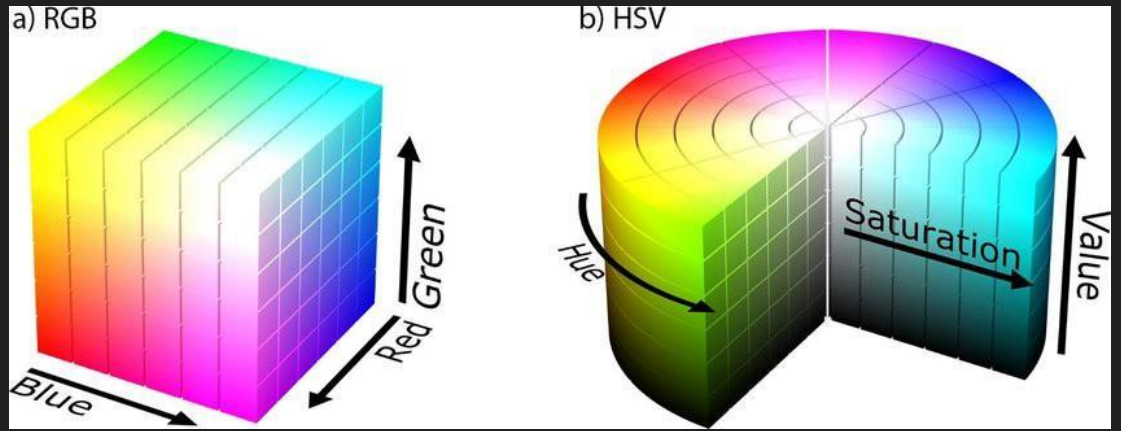
back to flowchart

HSV filter

In order to detect a color (for example green) it is easier to use HSV colors instead of RGB. For example, if the robot needs to detect green, it just needs to keep HUE between 90 and 140.

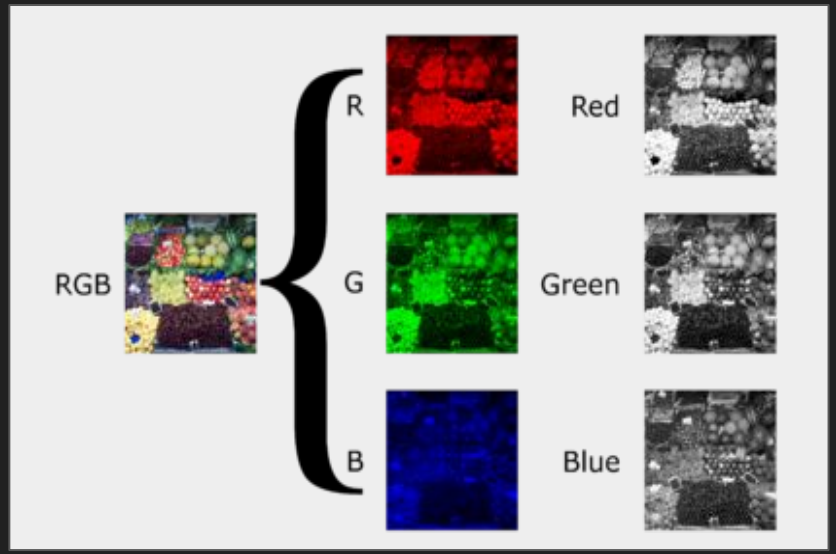
```
6  hsv = cv2.cvtColor(im, cv2.COLOR_BGR2HSV)
7
8  lower = np.array([0, 0, 0], np.uint8)
9  upper = np.array([180, 255, 200], np.uint8)
10
11 mask = cv2.inRange(hsv, lower, upper)
```





back to flowchart

convert to gray



A colored image has 3 times more pixels than a gray image, but when it needs to detect only a line; the robot doesn't need this information, and analyzing a gray image is quicker, so it converts it.

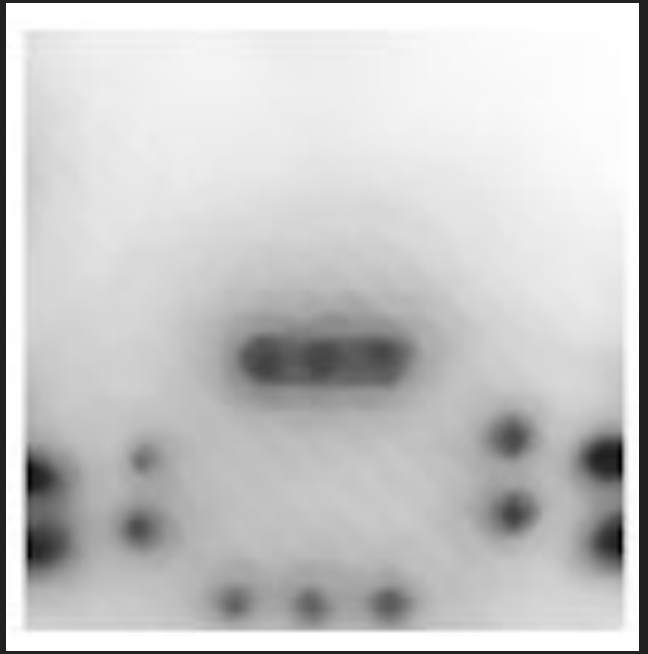
```
109    img2=cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY)
```

back to flowchart

sum a mask for light correction

We have some leds in the robot. Strong light sources are not good for taking a photo, the general image quality is actually better without leds because other light sources like room light are softer and farther.
However, external light sources are unstable and always different, whereas LEDs are always in the same position; this means that we can track it and correct the image.

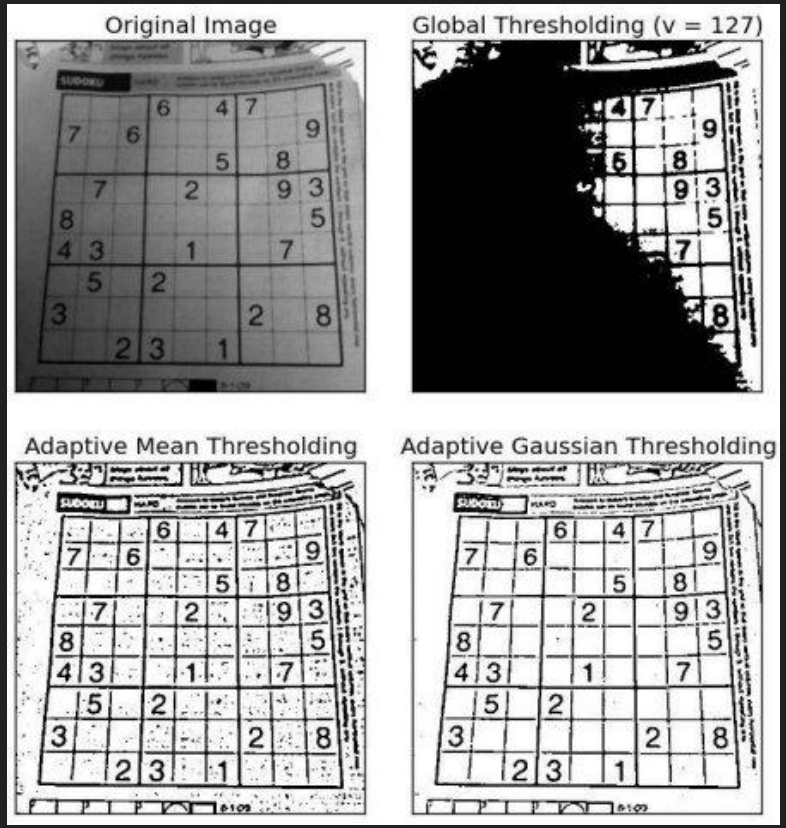
correcting mask



back to flowchart

gaussian threshold

In order to split black and white the easiest way is to apply a threshold: if a pixel value is higher than the threshold, it is white, otherwise it is black.
However, this does not work well with pictures that have shadows, so we use a gaussian threshold; in a gaussian threshold every pixel has a different threshold that depends on the pixels around it.




```
th2 = cv2.adaptiveThreshold(img2,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,61,10)
```

back to flowchart

check if gap

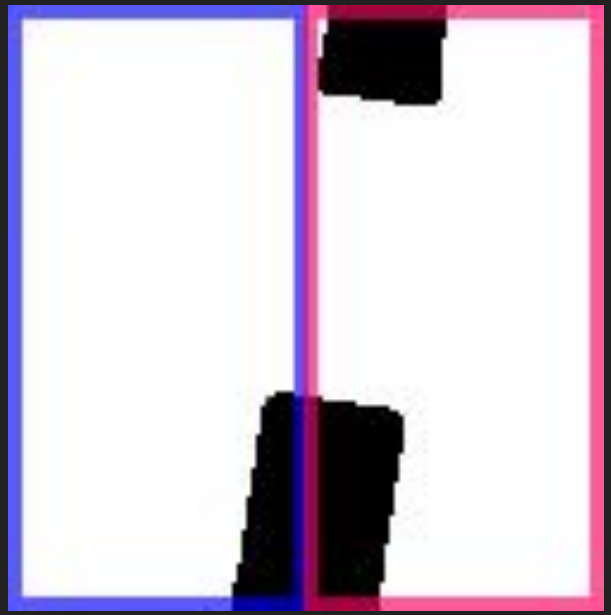
check if I cross

In order to detect if the robot is seeing a gap or a cross the number of white areas must be counted:

- 1 area: gap
- 2 areas: normal line
- 3 areas: “T” cross
- 4 areas: “X” cross

back to flowchart

difference between left and right black pixels



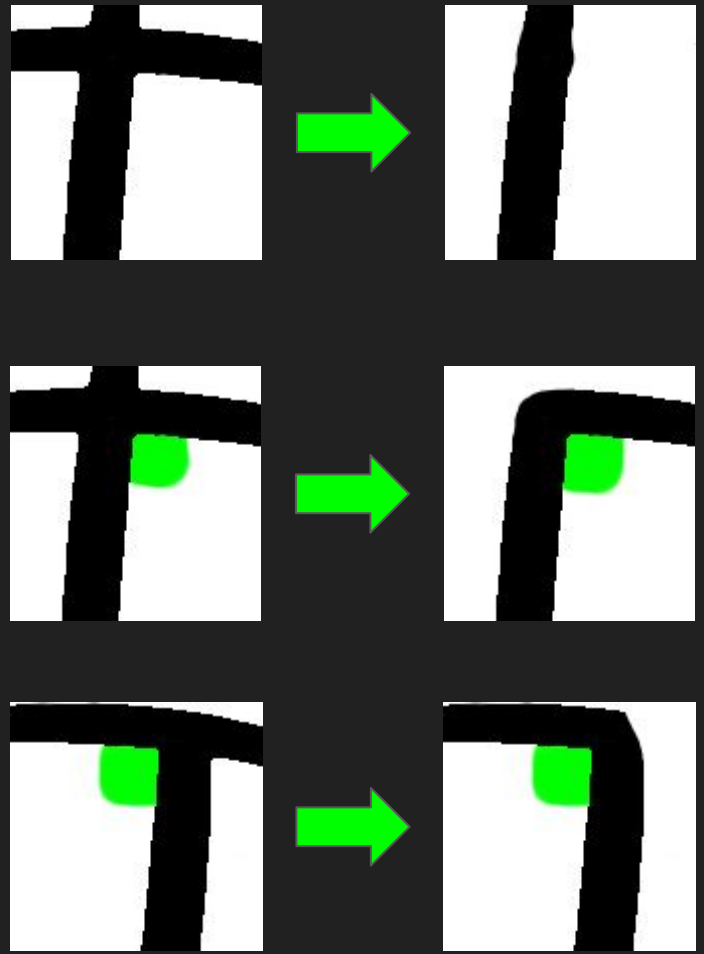
The robot makes the difference between the numbers of left and right black pixels, and it uses the value it gets in a proportional control for moving motors.

back to flowchart

removing wrong directions

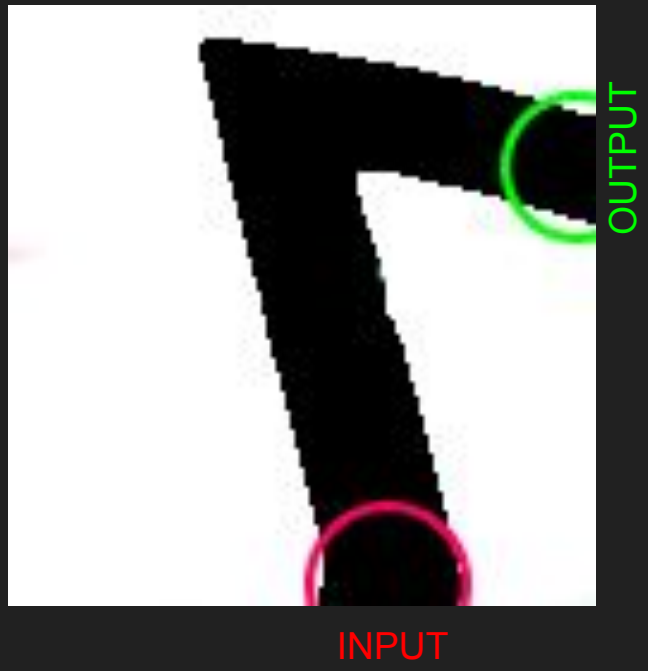
The algorithm that analyzes a cross and decides the right direction is the most complicated in this project... i can't explain it only with text and images, so check this video for a full explanation

VIDEO: https://youtu.be/Njx3aAMHUKc



back to flowchart

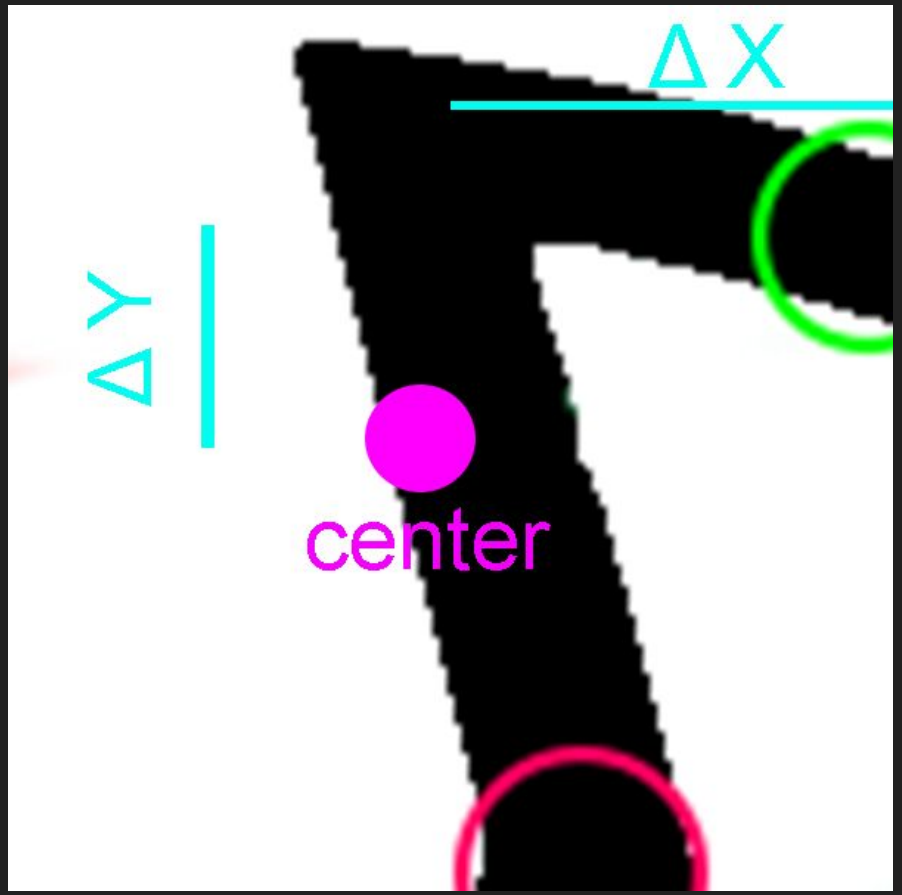
line in/out tracking



The robot tracks the input and output of the line. In order to achieve such a goal, it detects the black pixels near the position of the line input or output in the previous frame.

back to flowchart

2d control



It makes the difference between the center and the output.

```
846
847         deltaOutX=cX-64
848         #deltaInX=64-FX(pos)
849
850         deltaOutY=cY-64
851         #deltainY=128-FY(pos)
```

It uses these 2 values to give the correct speed to the motors (in case of gap it uses only ΔX)

back to flowchart

obstacle

In order to detect the obstacle we use HC-SR04 ultrasonic sensors: one is on the front of the robot, and two are on the left and on the right; as a result, they can stay close to the obstacle while turning around it.
While the robot is turning around the obstacle, it checks if it is seeing a line with the camera, and if it sees it, it interrupts the obstacle routine and continues to follow the line.

back to flowchart