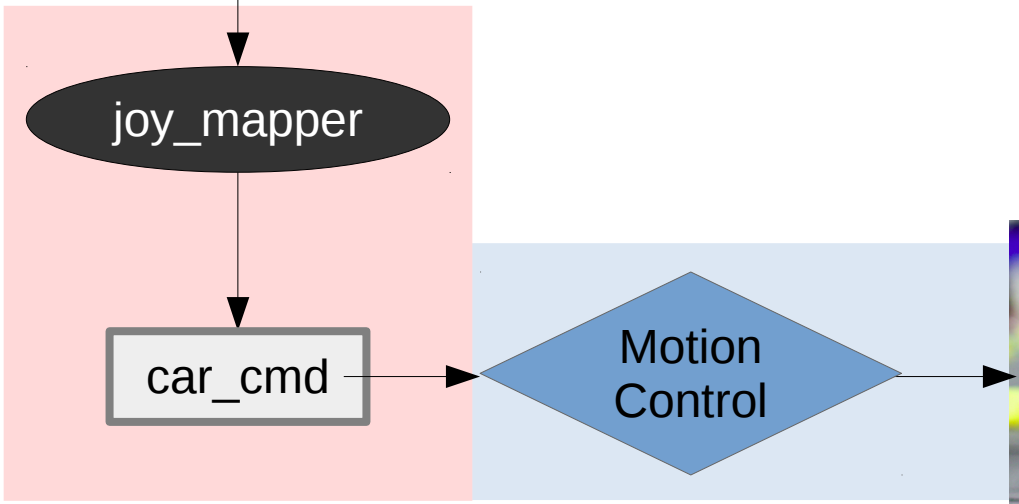
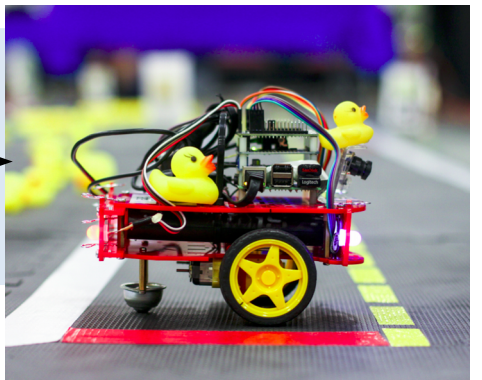
Following lanes

Moose must advance autonomously on the road, staying in the middle of the right lane.

Remember the joystick?

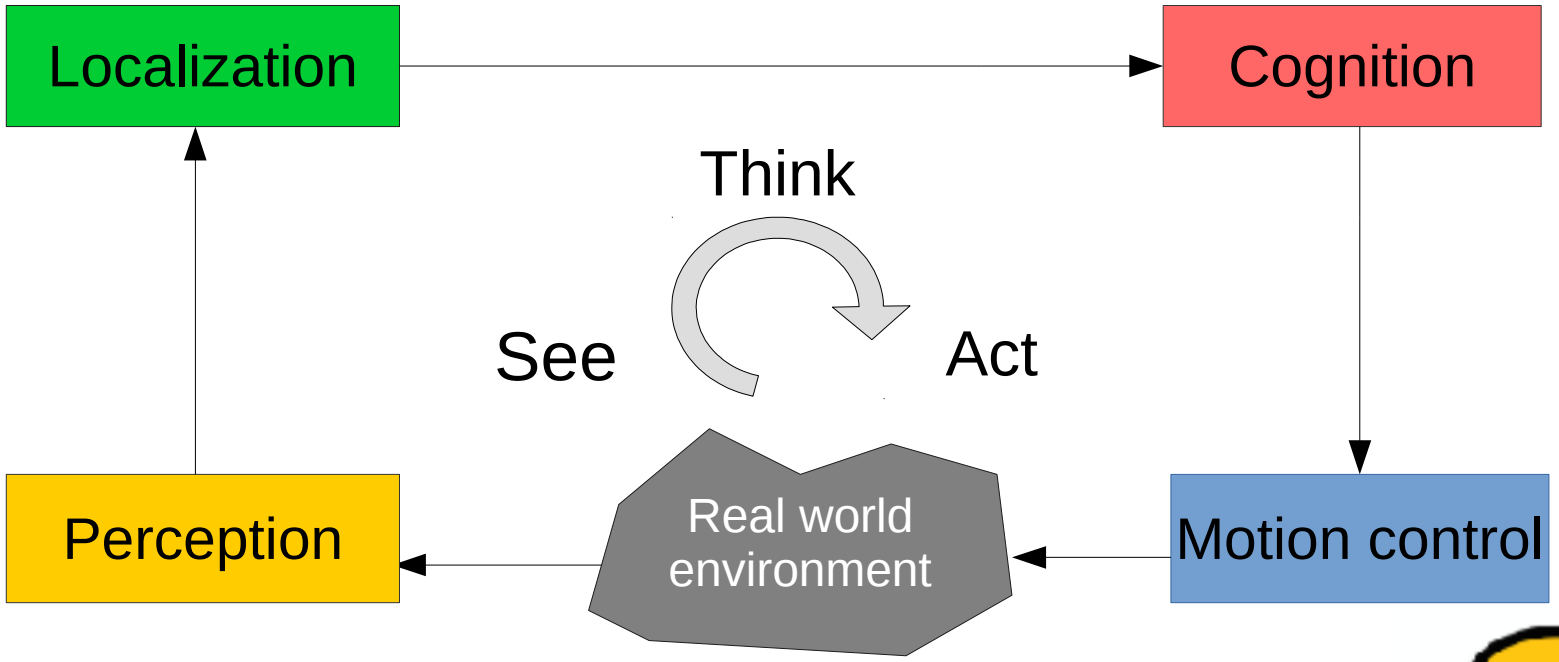
Was Moose autonomous then?





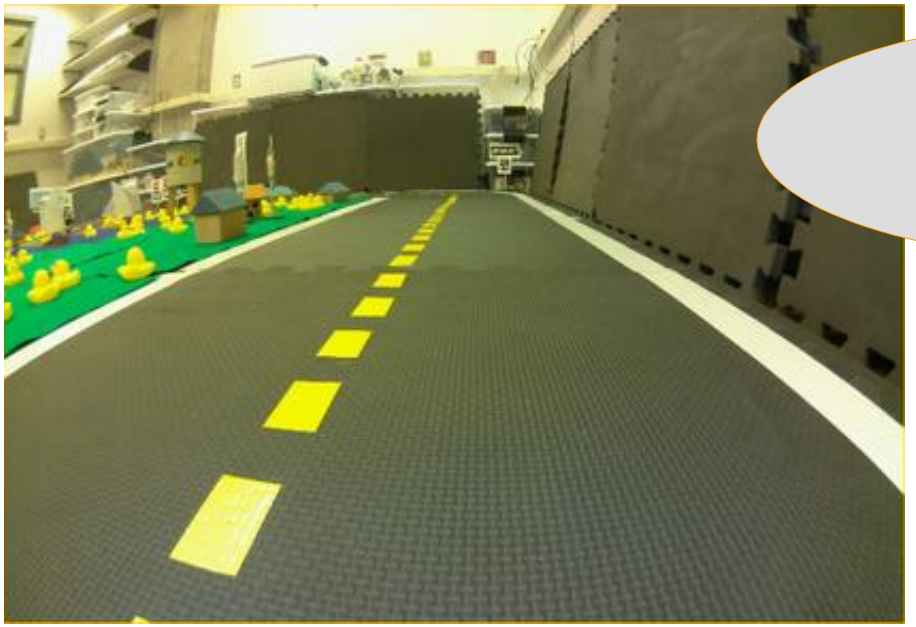
Autonomous mobile robot

See – think – act

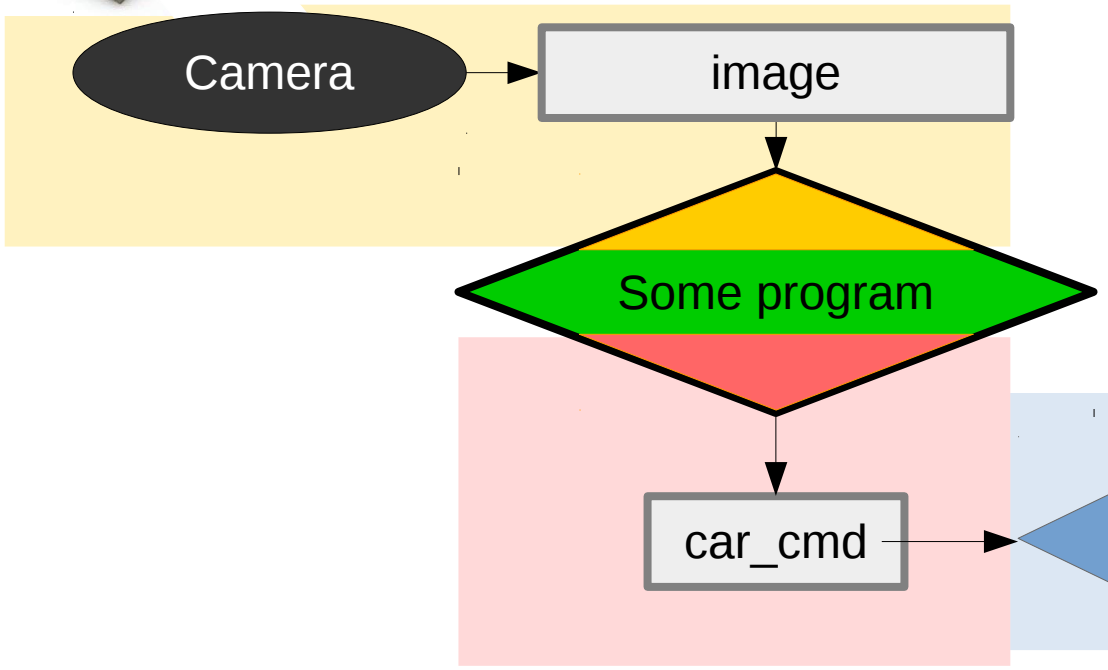


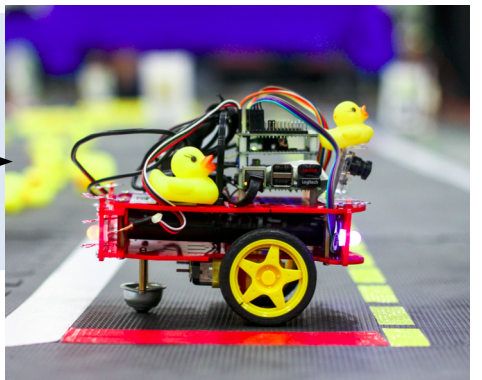
THIS IS THE MOST IMPORTANT SLIDE OF THE WEEK!

Following lanes – Overview

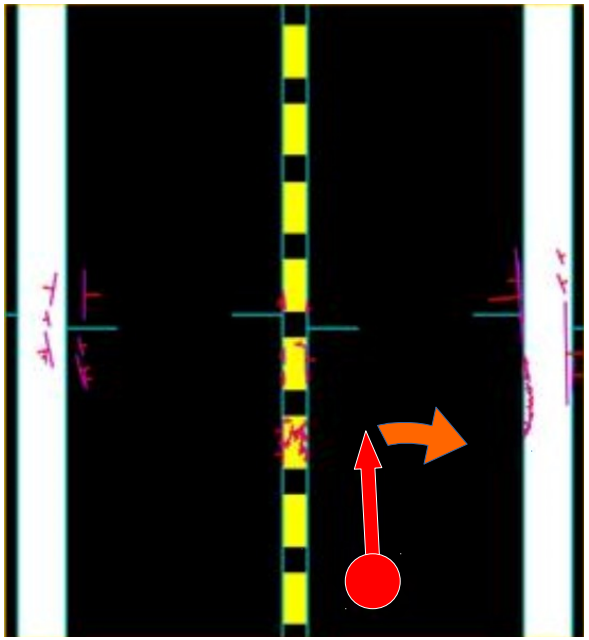


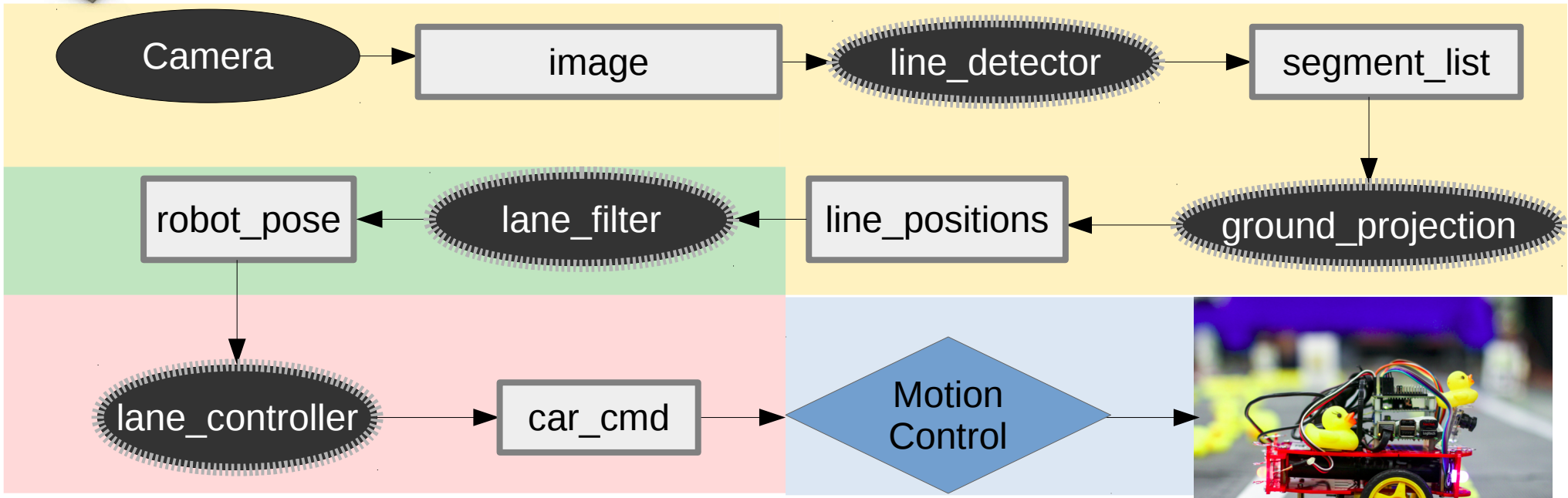
How would you do this program?





Following lanes – Overview





Perception : Line detection

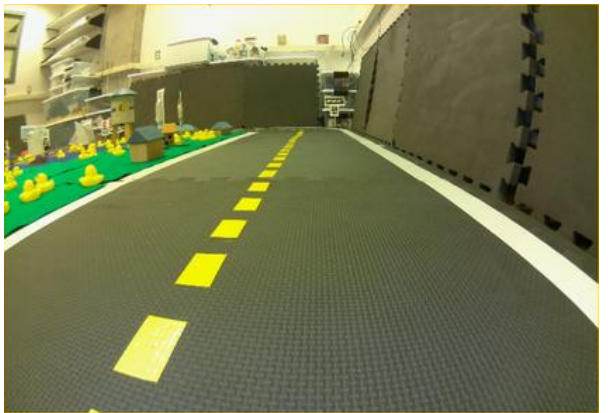
Sensing

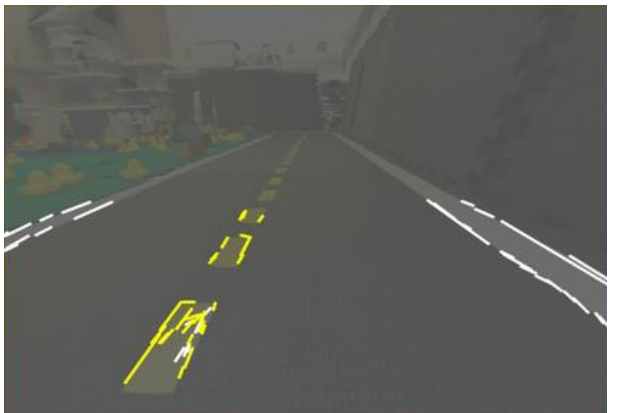
Raw data

Information Extraction

It’s a pillar!

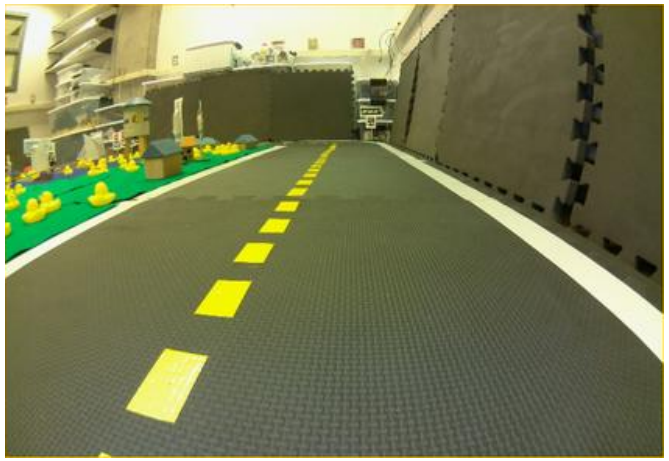
Here are the lines





Line detector

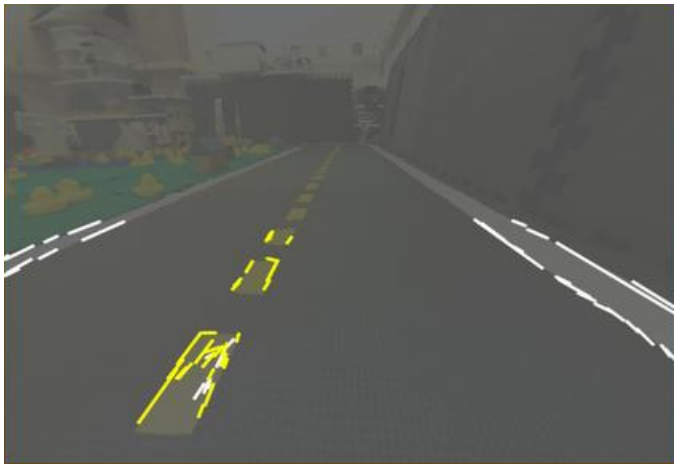
What we have:
A BGR image from the camera on the car.



What we do:
A line detector!

What we want:
A list of segments of color, given by the coordinates of the extremity points.

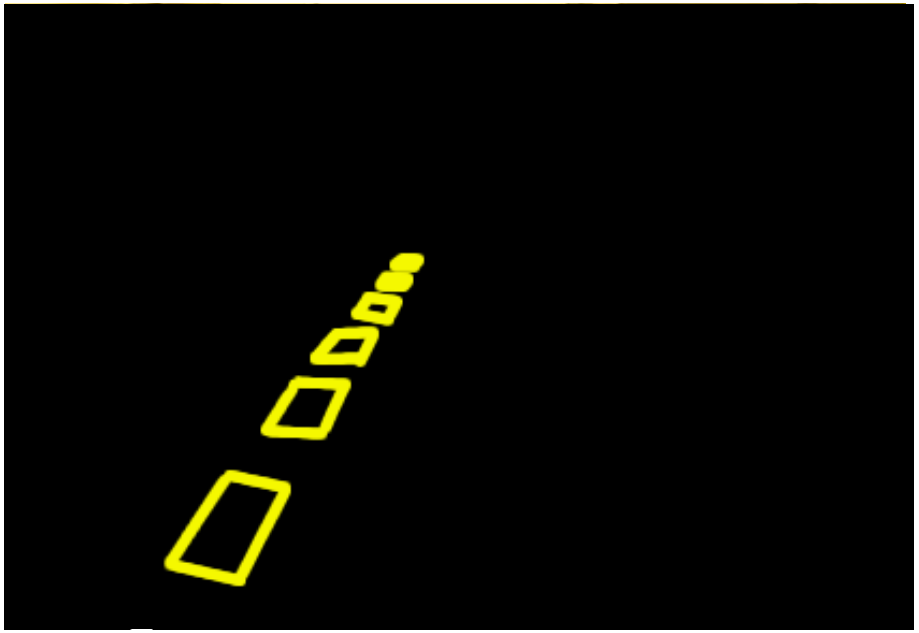
Line 1: (78, 259) to (103, 221)
Line 2: (103, 221) to (129, 224)
Line 3: (129, 224) to (105, 267)
Line 4: ...
...



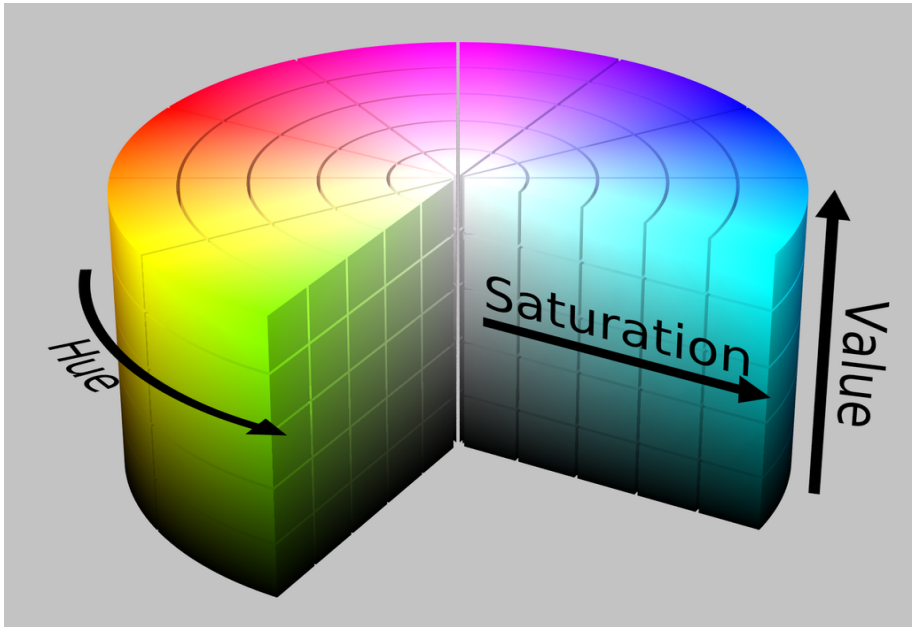
Line detector

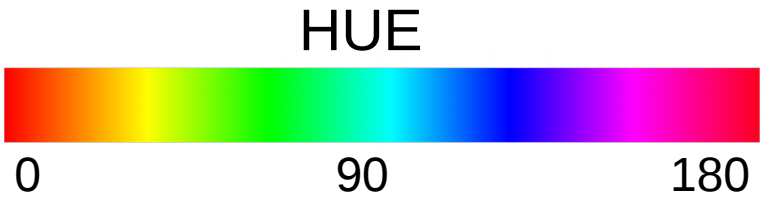
How do we detect lines of color in an image?

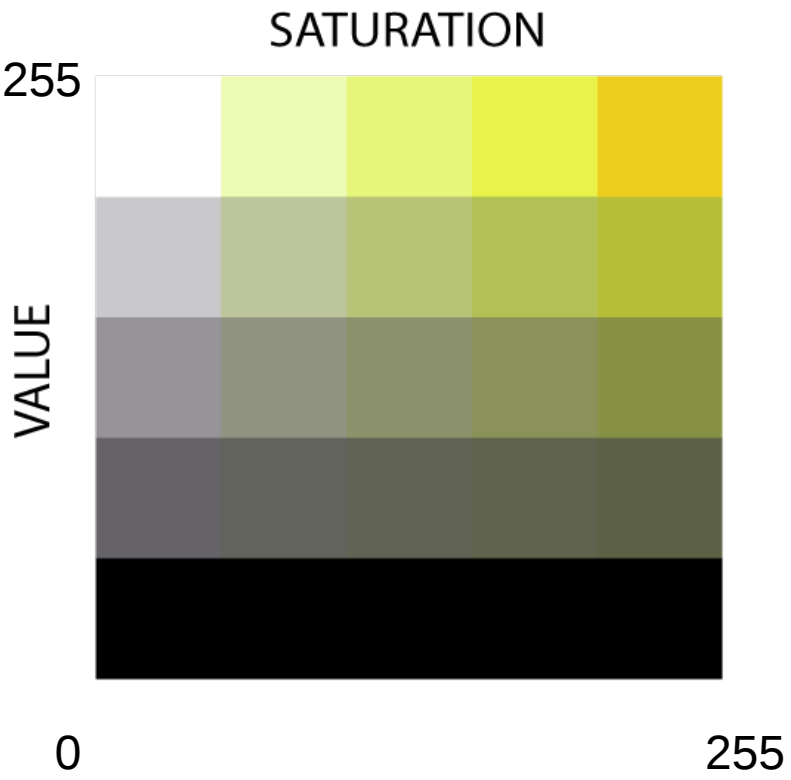
1. Convert image from Red-Blue-Green to Hue-Saturation-Value.
2. Filter color to only have the one wanted (white, yellow or red)
3. Get edges with contrast
4. Get lines through Hough transform



1. Convert an image from BGR to HSV

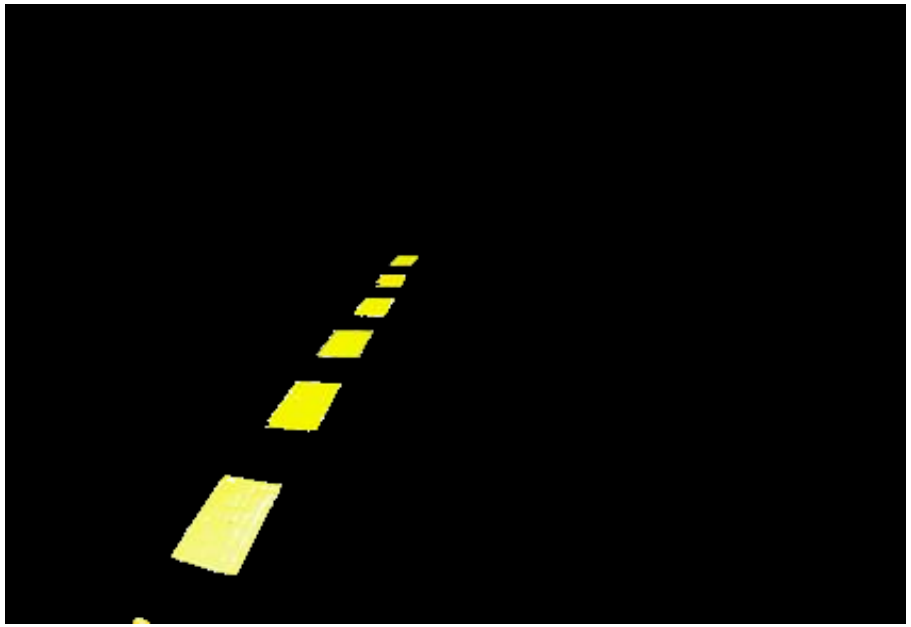


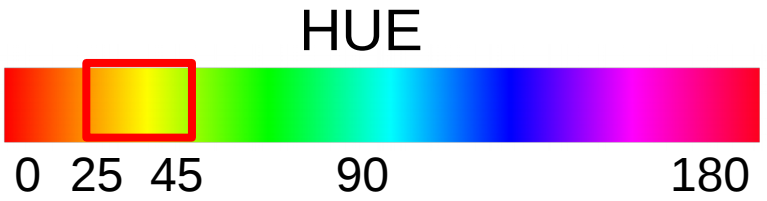


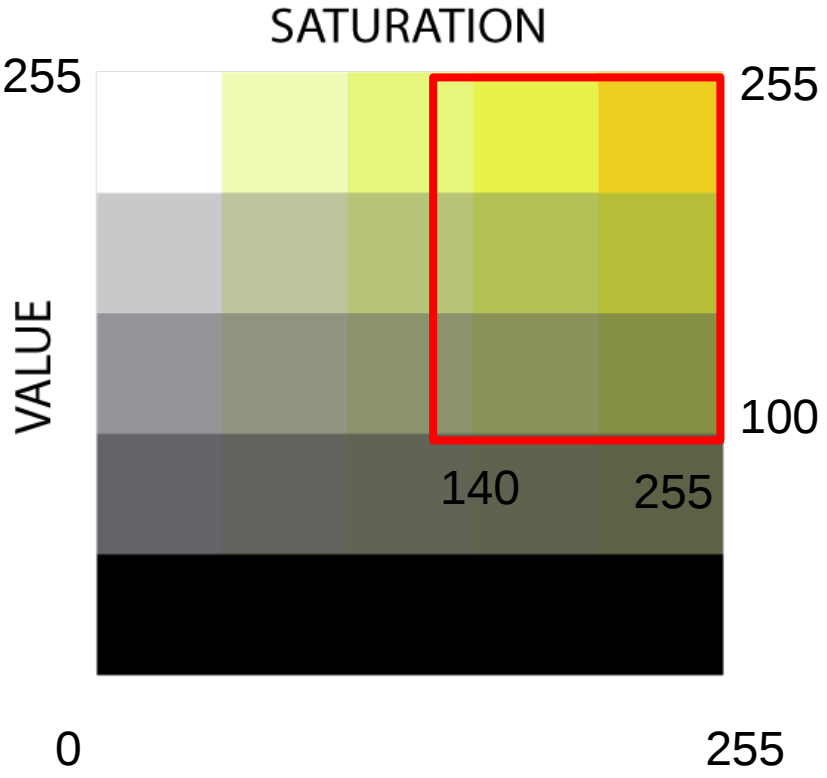


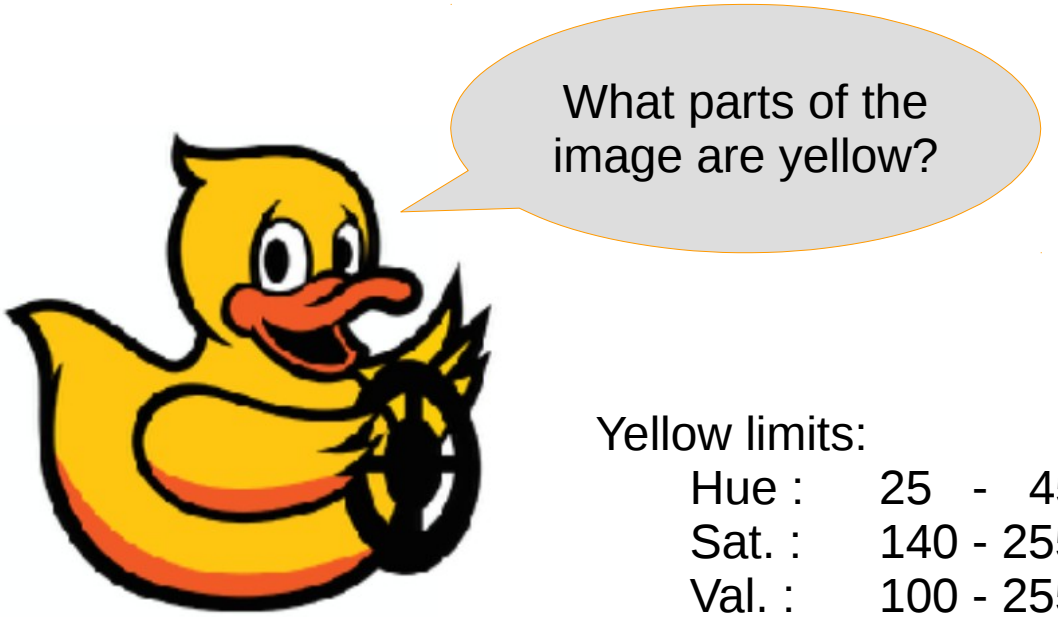
According to you, why do we need the image to be in HSV?

2. Filter color: we only want the yellow!





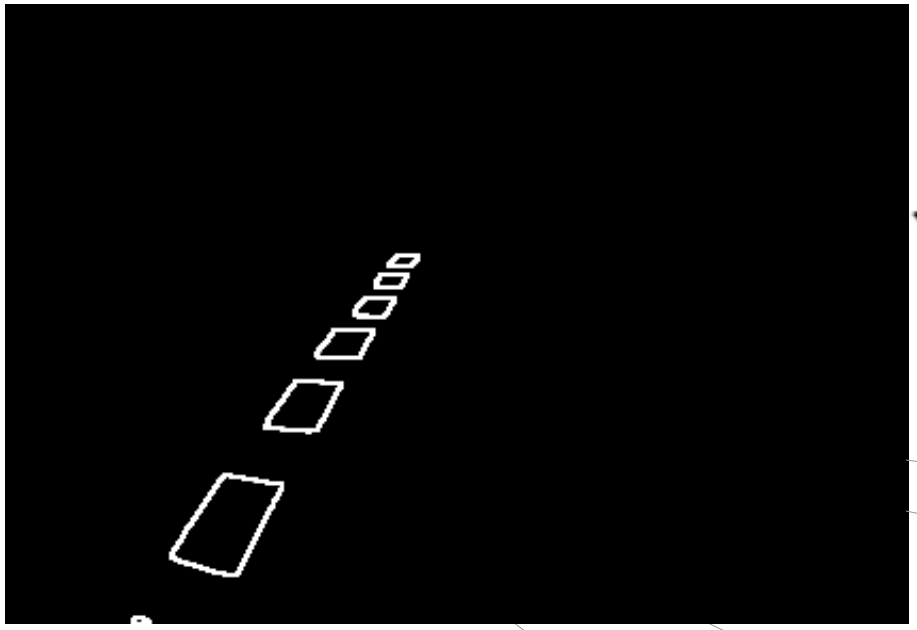




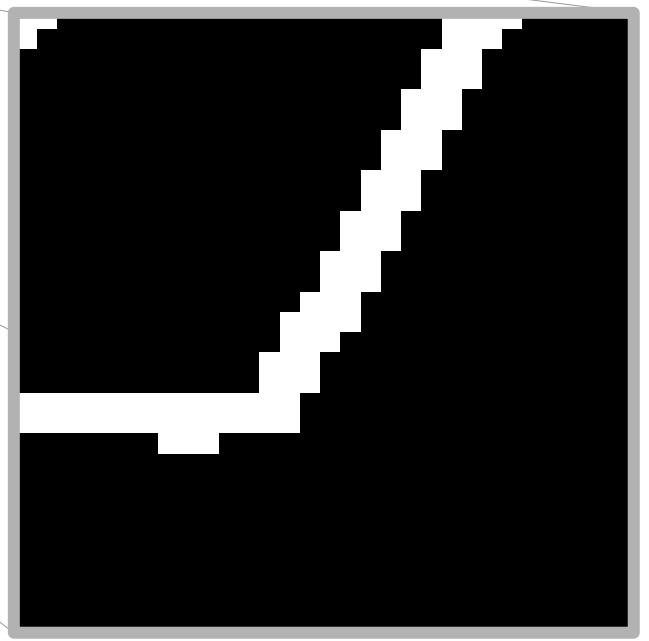
Yellow limits:

Hue :	25	- 45
Sat. :	140	- 255
Val. :	100	- 255

3. Detecting the edges



How would you detect the edges?



Answer:

1st : turn into black and white

2nd: look for the contrast!

The real process is a bit more complex and it is called: **Canny edge detection.**

4. Detecting lines

What do we have?

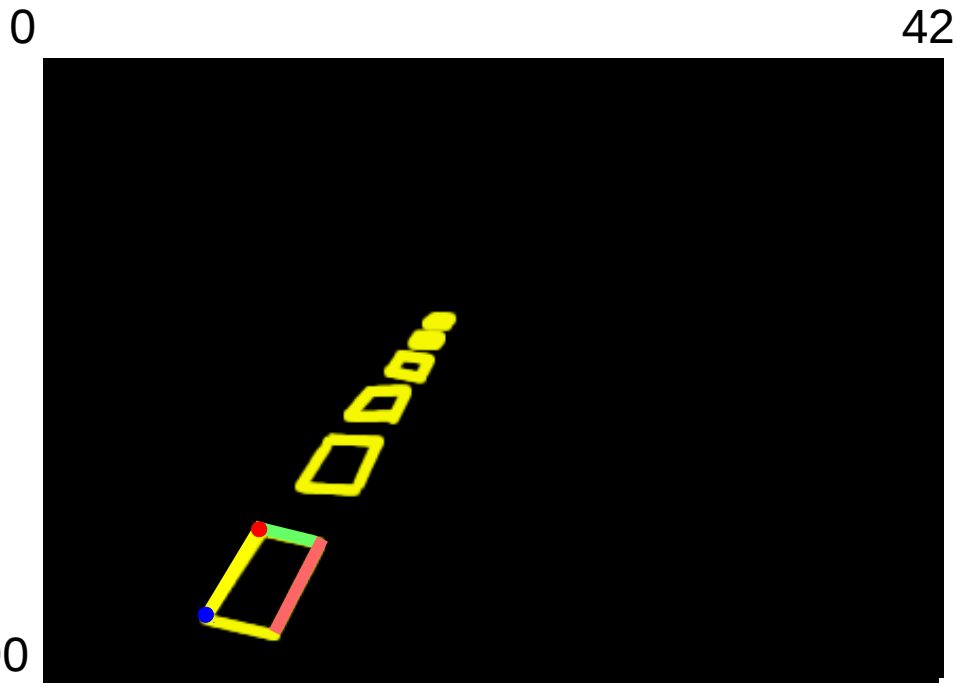
- An image with white pixels forming lines.

What do we want?

- A list of lines

How do we do this?

- **Hough transform**!



Line 1: (78, 259) to (103, 221)
Line 2: (103, 221) to (129, 224)
Line 3: (129, 224) to (105, 267)
Line 4: (,) to (,)
Line 5: (,) to (,)
Line 6: (,) to (,)
Line 7: (,) to (,)
Line 8: (,) to (,)
Line 9: (,) to (,)
Line 10: (,) to (,)
Line 11: (,) to (,)
Line 12: (,) to (,)
Line 13: (,) to (,)
Line 14: (,) to (,)

Ouf! We made it!

But now, it is your time to act!

Vincent is in trouble. He is trying to make Moose follow a lane but it does not work.

He needs to look into the line detection code to see where is the bug.

But he realizes that there are 4 different line detectors codes! And the one that Moose is currently using is totally empty!

Somebody made a mess in there...

Could you help Vincent choose which of the 4 line detectors is the one Moose should use?

A, B, C, or D?

1) What we have: BGR image from the camera

```
self.detector = line_detector_X()
```

2) What we do: “line_detectorX.py”

```
# Set the image to be detected
self.detector.setImage(image_cv_corr)

# Detect lines and normals

white = self.detector.detectLines('white')
yellow = self.detector.detectLines('yellow')
red = self.detector.detectLines('red')
```

3) What we get: list of white, yellow and red segments in the image.

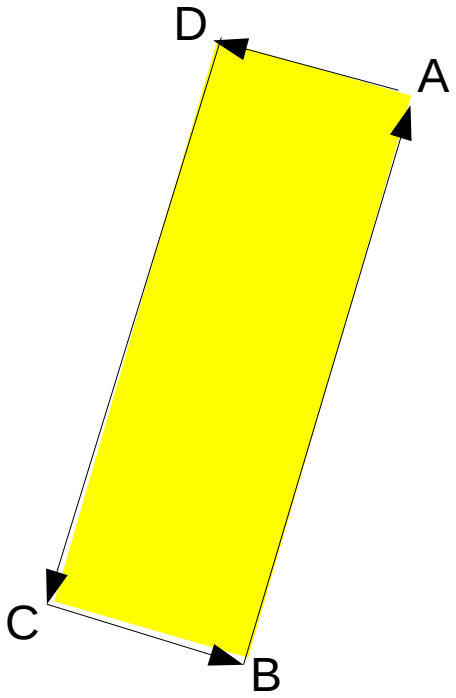
You found it? Great!

Now that we have the segments, we need to do a bit more work:

- Set the segment direction.
- Compute the normals and centres of the segments.

Convention

Where is the line with respect to the yellow, white or red band?



What are the 4 lines we can see here?

AB, or BA? BC or CB? CD or DC? AD or DA?

We need a convention:

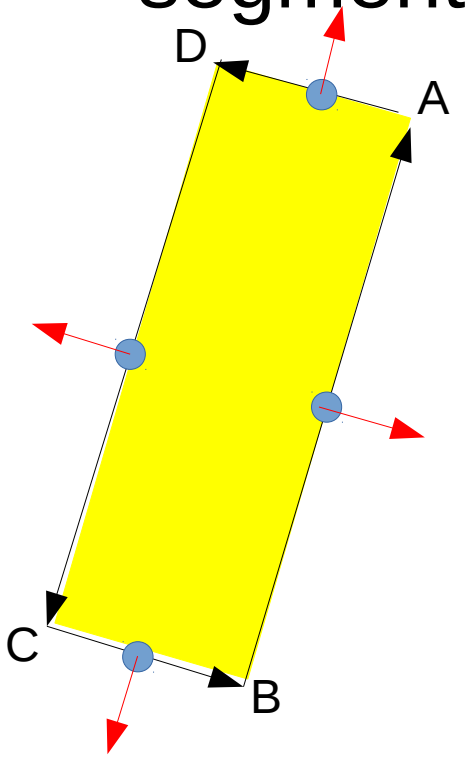
The color has to be on the LEFT of the segment from P1 to P2.

Therefore, we want the segments to be:

BA, AD, DC, CB.

Center and norm

We also need the center and norm of each segment.



What is the center?

- the middle point of the segment

What is the norm?

- an unit vector perpendicular to the segment.
It always goes from the color to outside.
As a vector, it has two members: *x, y*

Task of the day

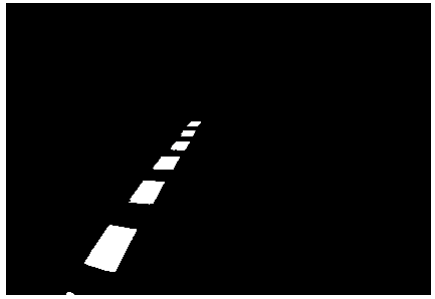
In line_detector1.py, write function self._findNormal(bw, lines)

Inputs:

bw is a black and white image of the color filter.
lines is a matrix of lines

Outputs:

centers is a vector of centers
norms is a matrix of norms
lines follows the convention



bw

Line 1 A	Line 1 B
Line 2 A	Line 2 D
Line 3 D	Line 3 C
Line 4 B	Line 4 C

lines

_findNormal

Center line 1
Center line 2
Center line 3
Center line 4

centers

Norm1 x	Norm1 y
Norm2 x	Norm2 y
Norm3 x	Norm3 y
Norm4 x	Norm4 y

norms

Line 1 B	Line 1 A
Line 2 A	Line 2 D
Line 3 DC	Line 3 P2
Line 4 C	Line 4 B

lines

You made it? Awesome!

Now, let’s see it working in real life, on Moose!