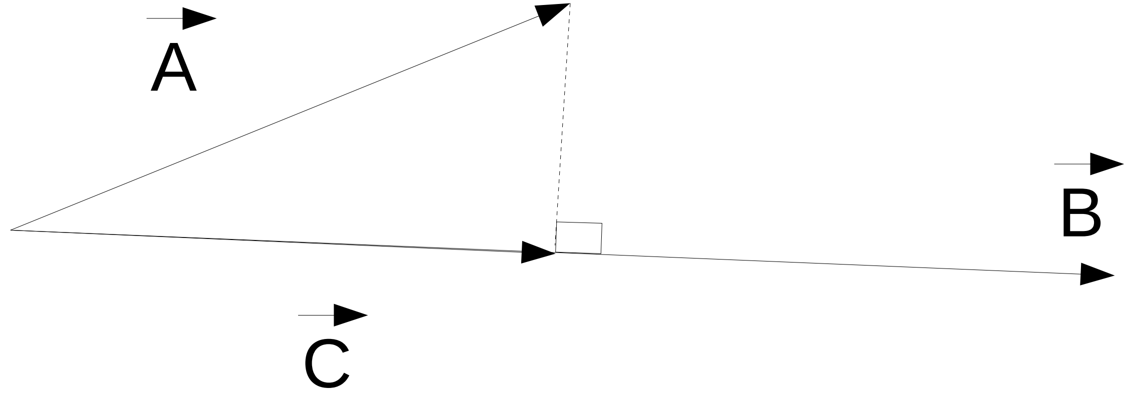
More maths!

The dot product and projections: another way to see it

Dot product : projections



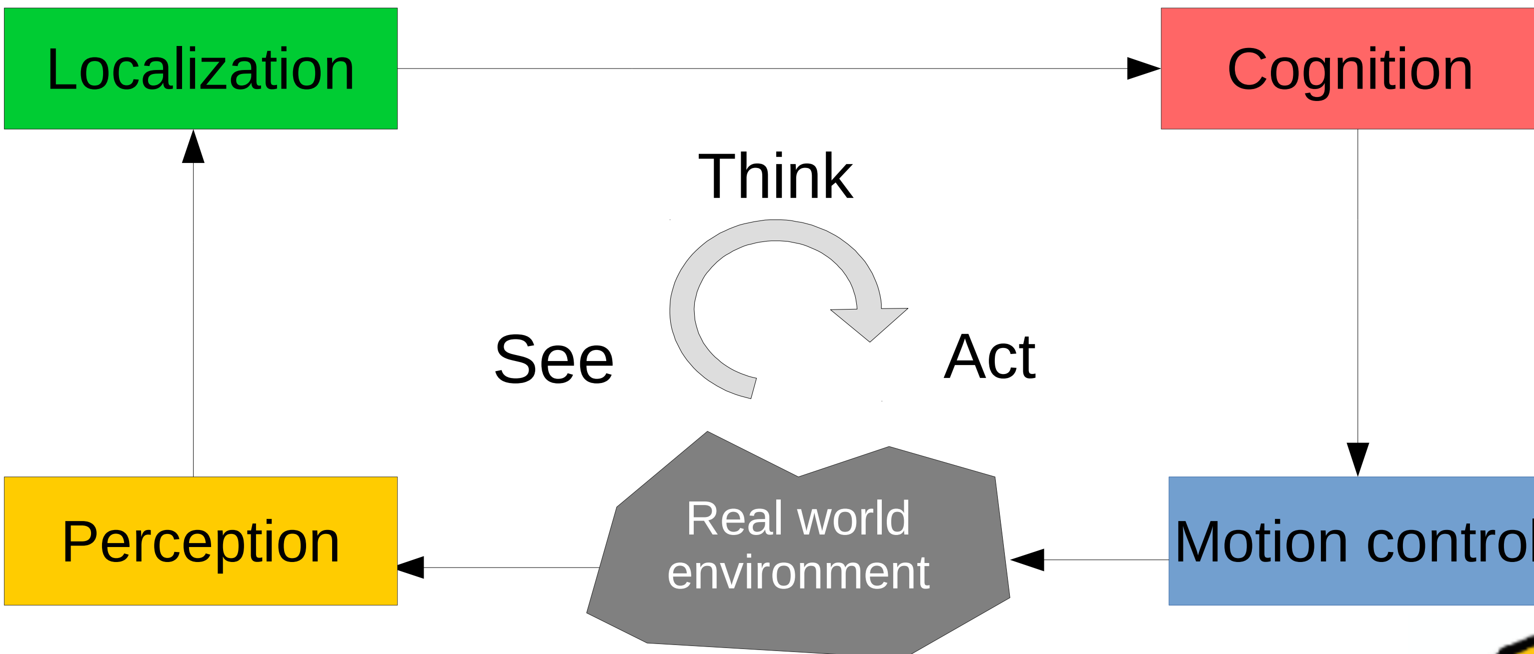
$$\vec{C} = \vec{A} \cdot \hat{B}$$

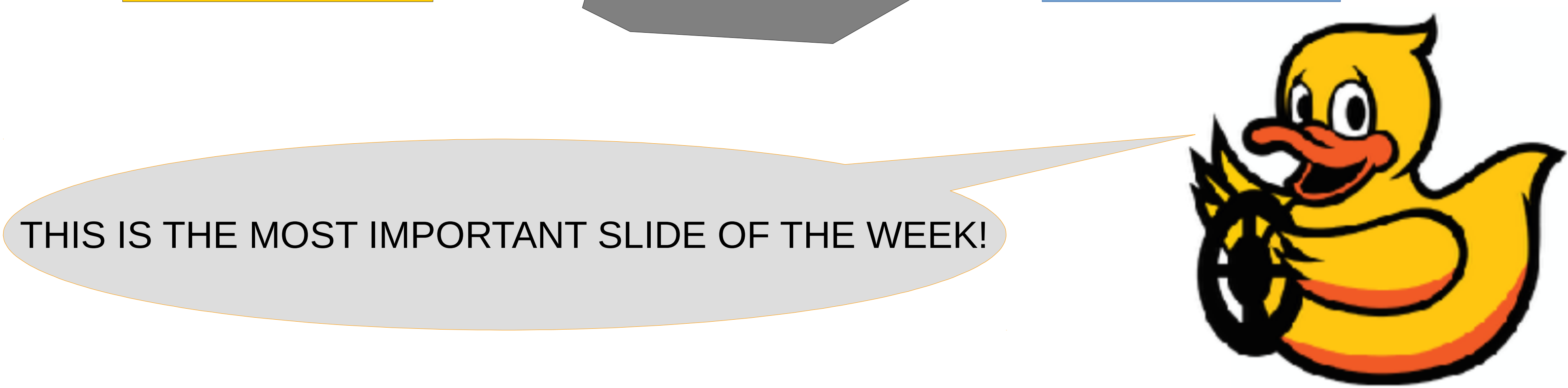
Where $\hat{B}$ is the unit vector of $\vec{B}$

Lane filter

Autonomous mobile robot

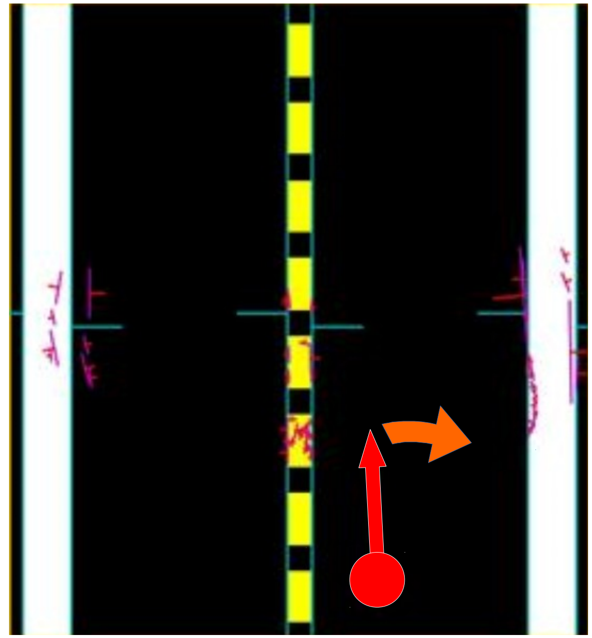
See – think – act

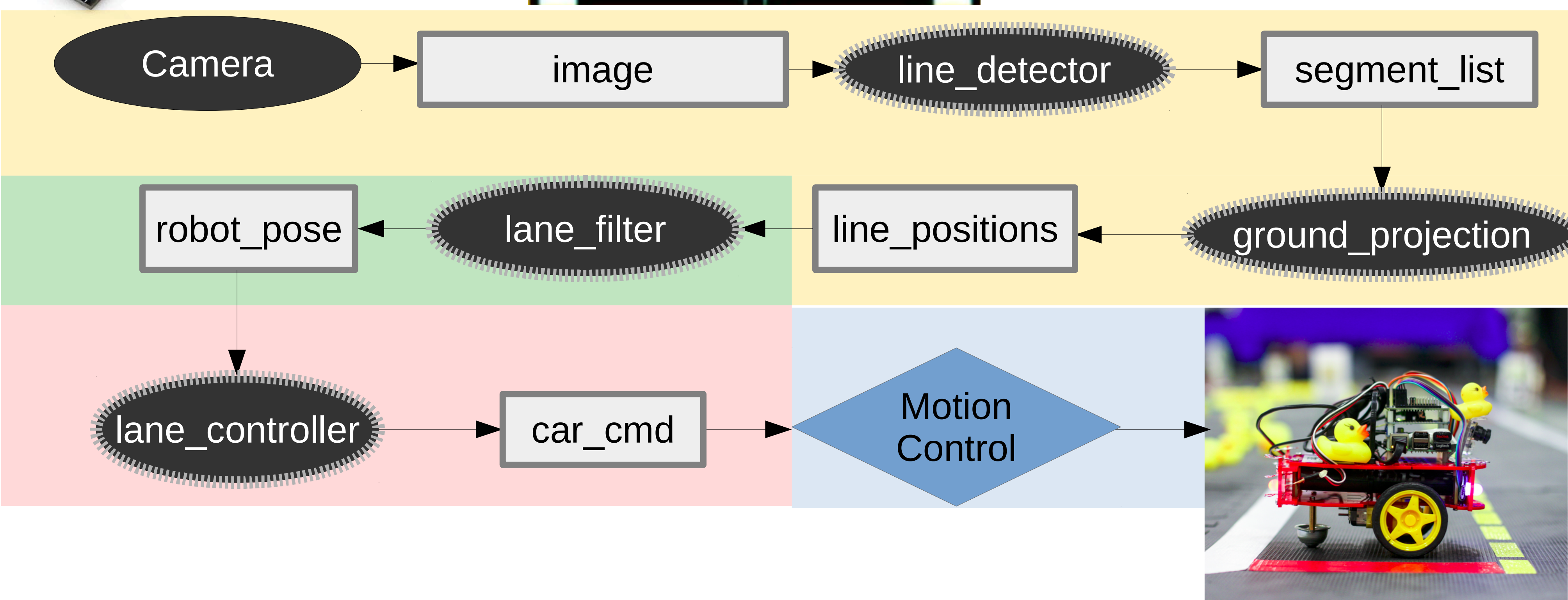




Autonomous Mobile Robots – R. Siegwart, I. Nourbakhsh, D. Scaramuzza

Following lanes – Overview





Localization

Data base

Information → Position

Movement

Data base: where are the pillars.

I see a pillar

There are 3 places I can see a pillar from

I moved forward

This is where I can be now

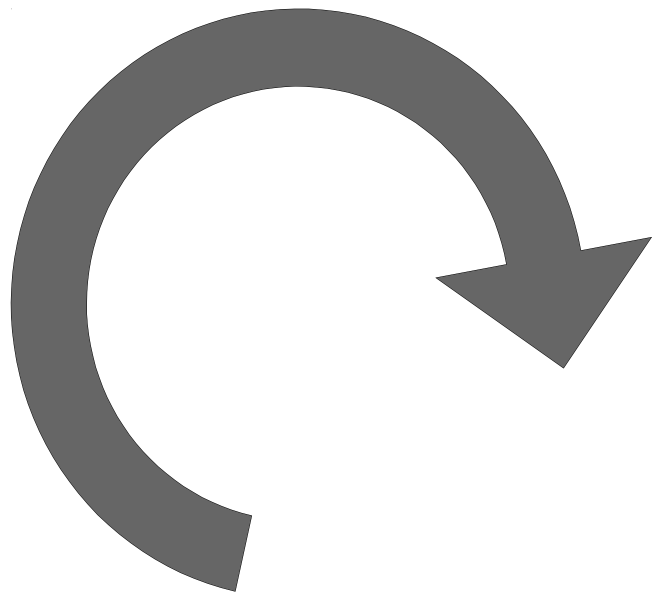
I see a pillar again

There are 3 places I can see a pillar from

Therefore I must be here now

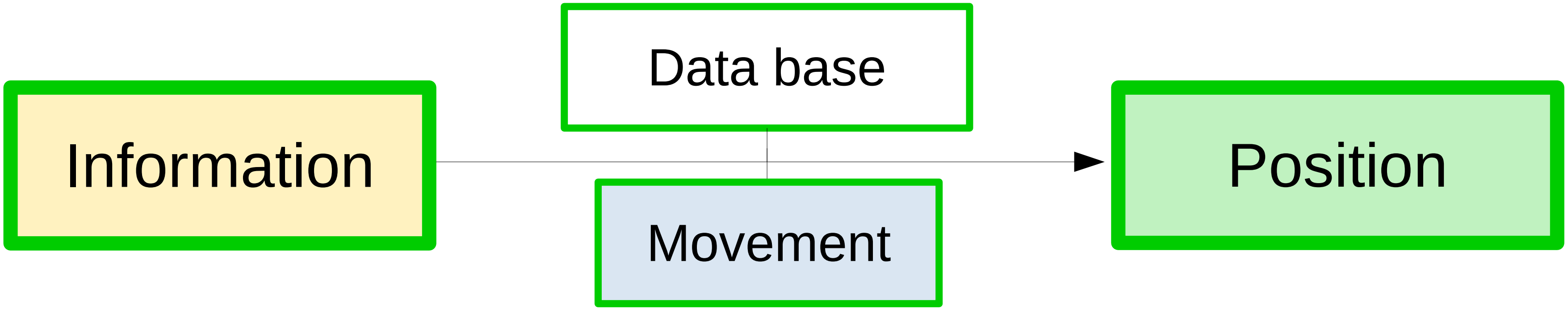
Lane filter

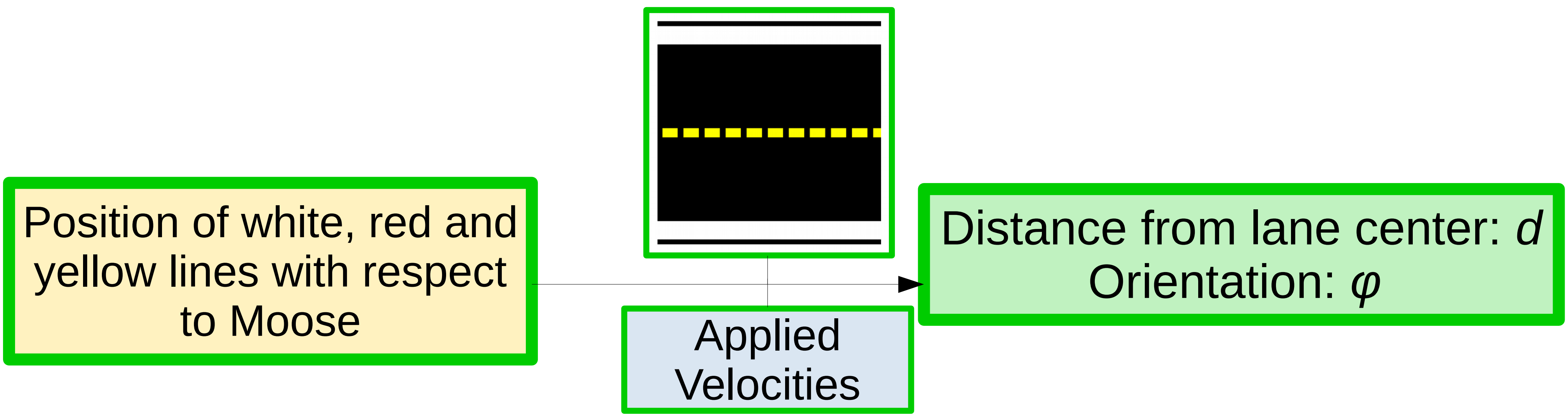
Predict
With movement



Update
With measurements
and database

Localization





What do we want to estimate?

Distance *d*

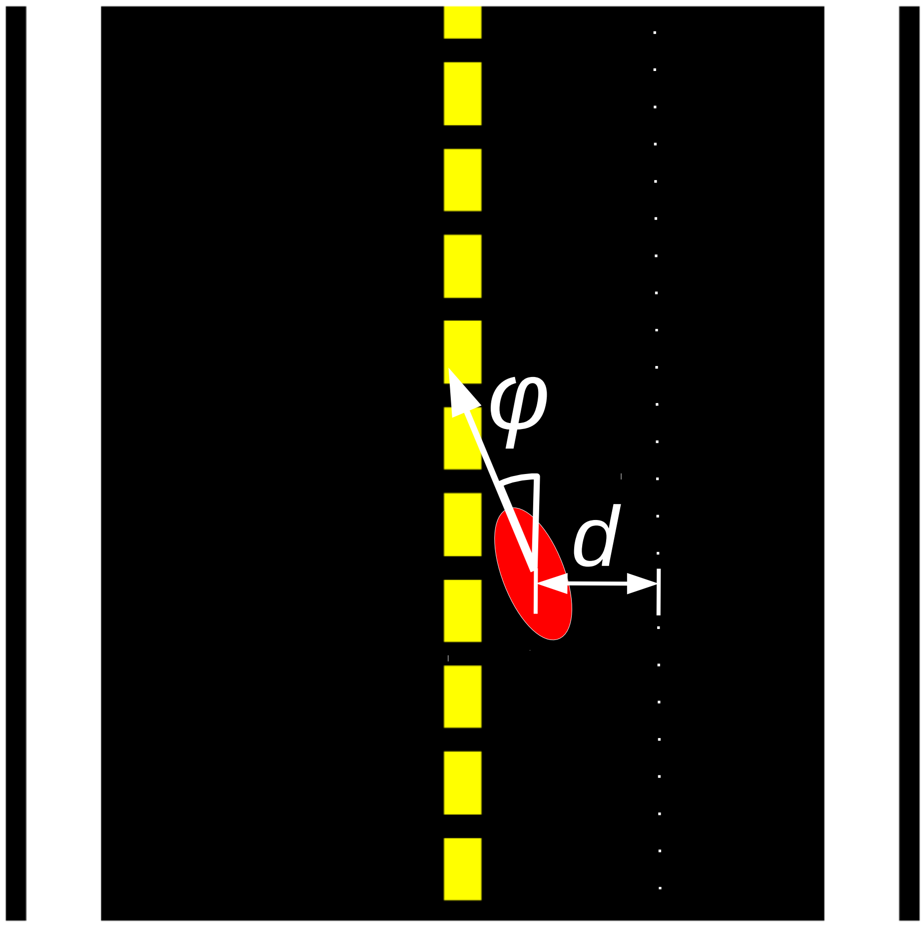
from center of lane

d > 0 : too much on the left

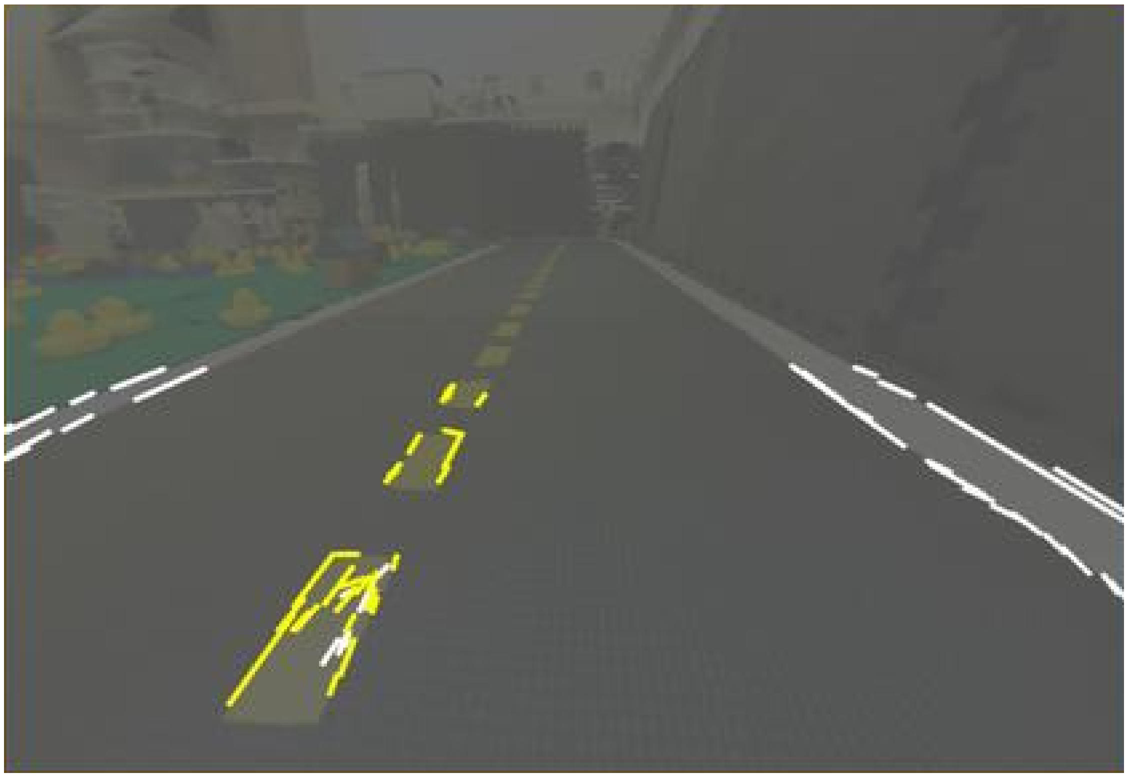
Orientation *φ*

from straight direction

φ > 0 : looking too much to the left



Can we be sure?

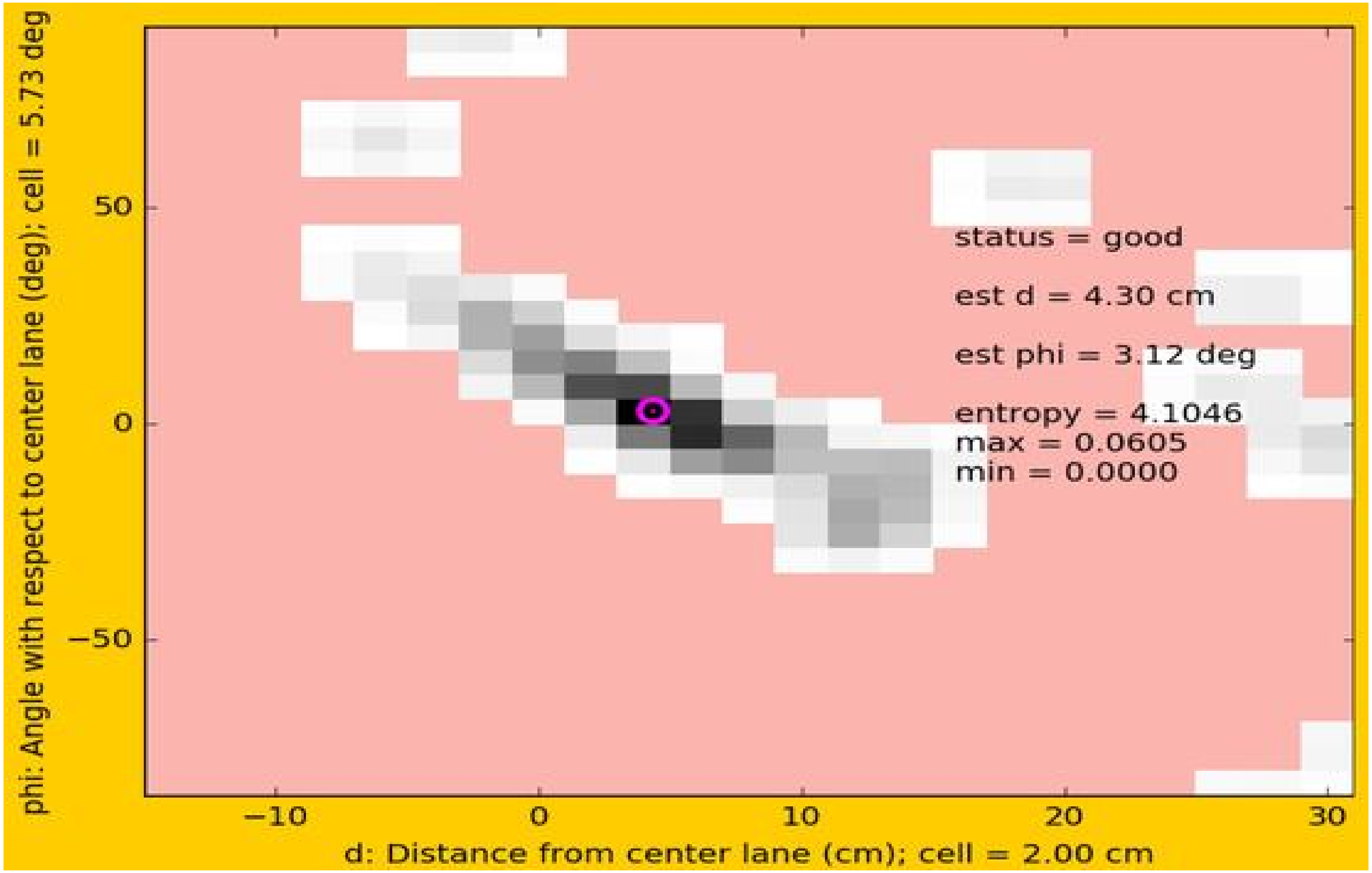


Do you see anything interesting on this picture?

Representing belief

p(d, φ)

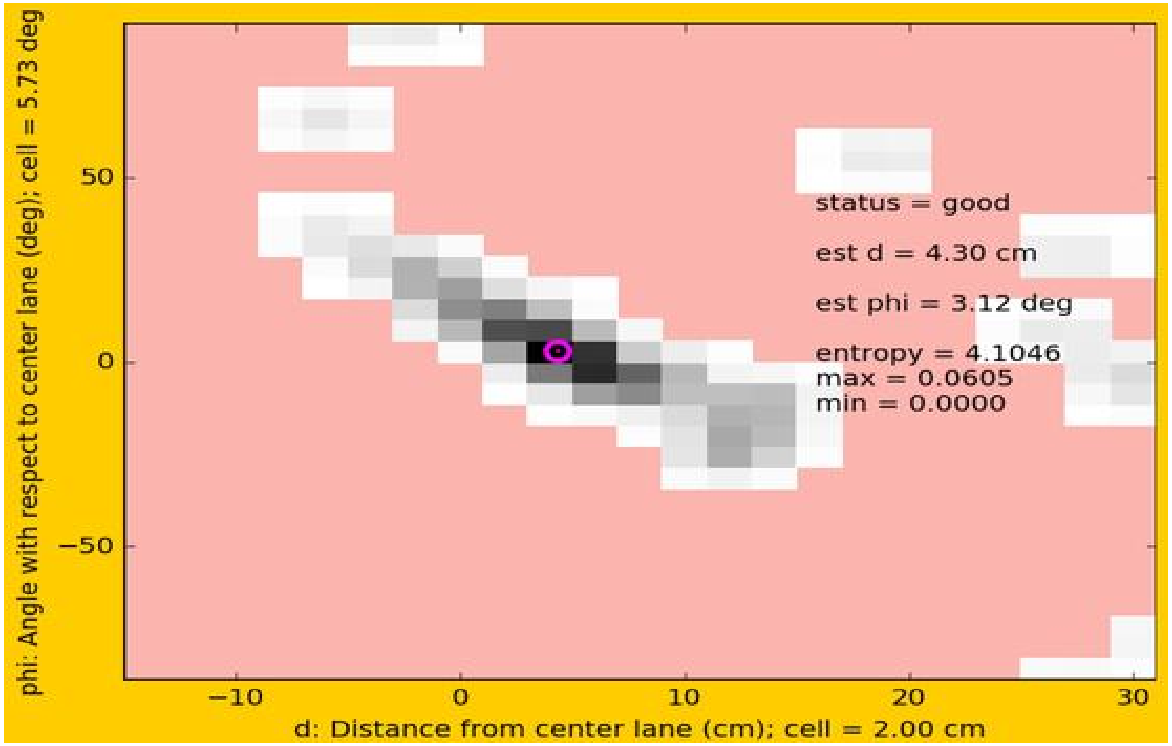
φ



d

Representing belief

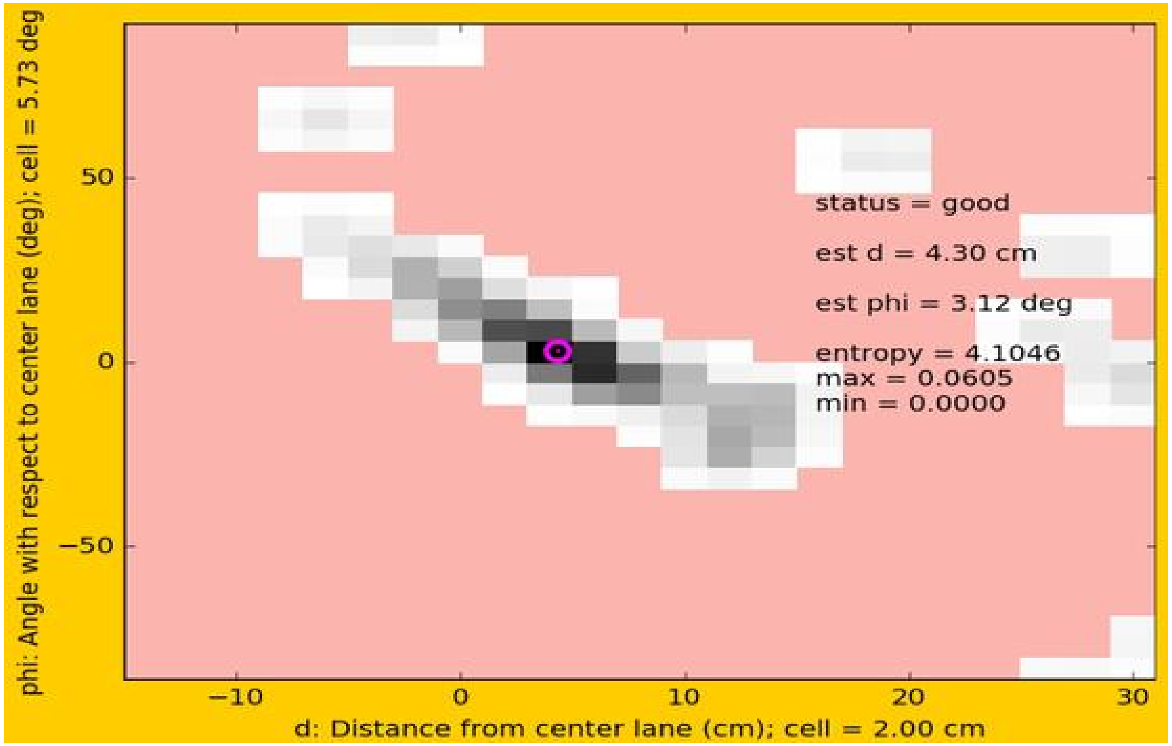
$p(d, \varphi)$



This is a probability map. The sum of all elements is 1.

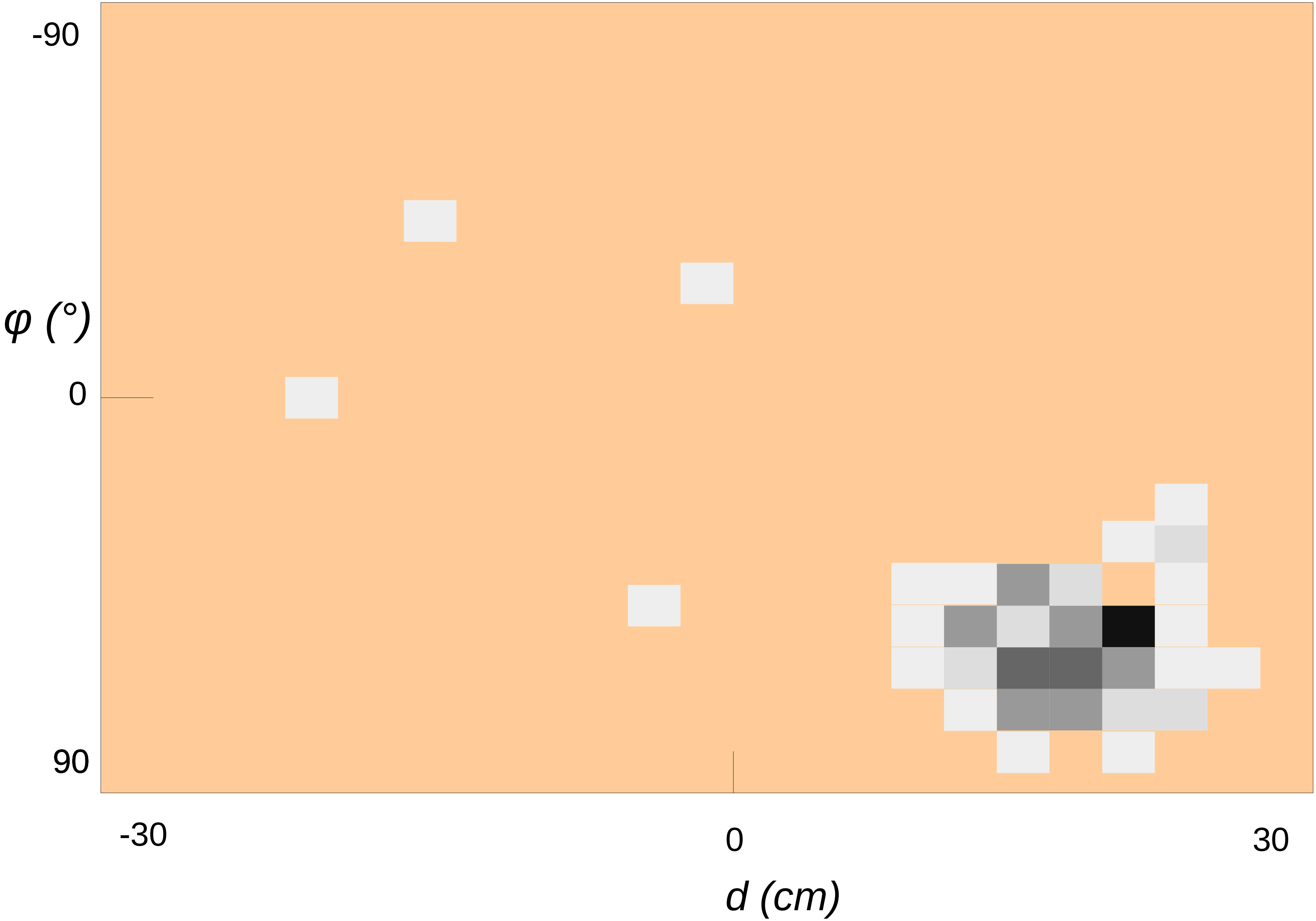
Representing belief

p(d, φ)



We can estimate the current position and orientation by taking the coordinates of the maximum in the map!

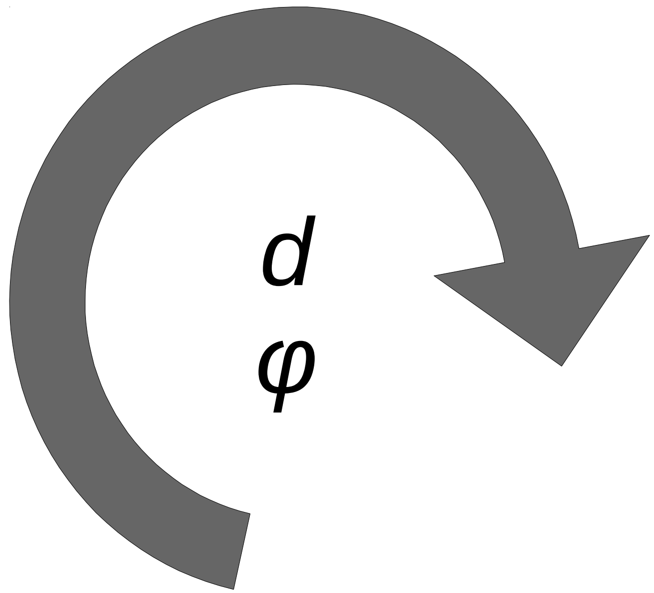
Your turn!



Lane filter

Predict

With applied velocities

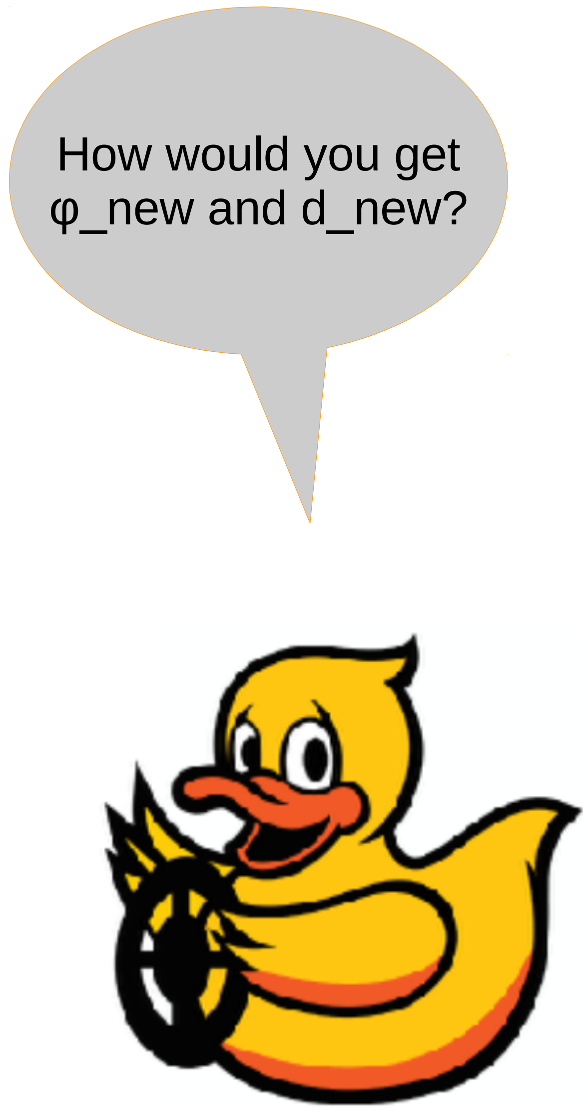


Update

With position of lanes

Predicting

φ_{old}	the previous orientation estimation
d_{old}	the previous position estimation
ω	the angular velocity
v	the linear velocity
δt	the time since previous estimation



$$\varphi_{new} = ?$$
$$d_{new} = ?$$

Your turn! Find a way to estimate φ_new and d_new.

φ_{old}	the previous orientation estimation
d_{old}	the previous position estimation
ω	the angular velocity
v	the linear velocity
δt	the time since previous estimation

$$\varphi_{new} = ?$$
$$d_{new} = ?$$

Predicting

Angular velocity ω is the derivative of orientation φ!

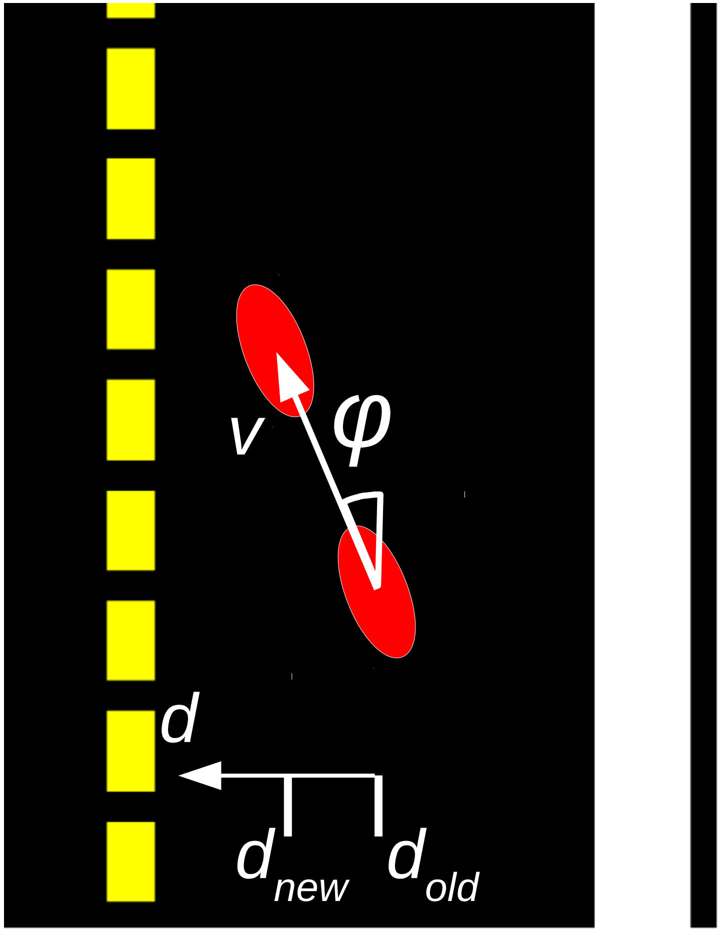
$$\omega = \frac{\varphi(t + \delta t) - \varphi(t)}{\delta t} = \frac{\varphi_{new} - \varphi_{old}}{\delta t}$$

Therefore:

$$\varphi_{new} = \varphi_{old} + \omega \cdot \delta t$$

Predicting

Linear velocity *v* is the derivative of position... in 2 dimensions!

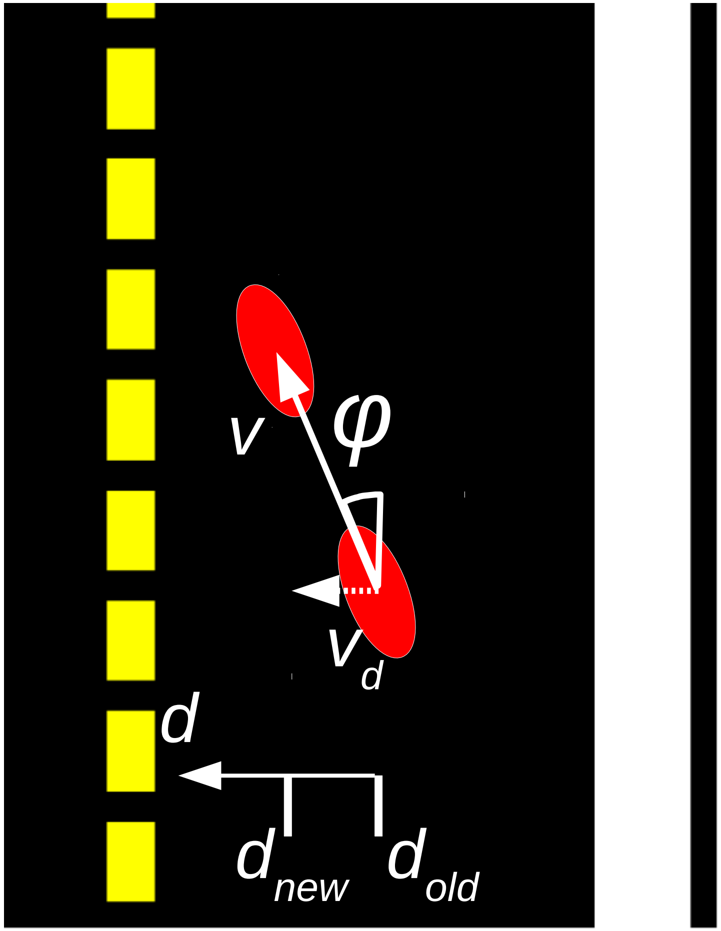


Predicting

Linear velocity *v* is the derivative of position... in 2 dimensions!

We need to project *v* in the *d* direction.

$$v_d = v \cdot \sin(\varphi)$$



$$d_{new} = d_{old} + v_d \cdot \delta t$$
$$= d_{old} + v \cdot \sin(\varphi_{old}) \cdot \delta t$$

Predicting

Dynamic equations:

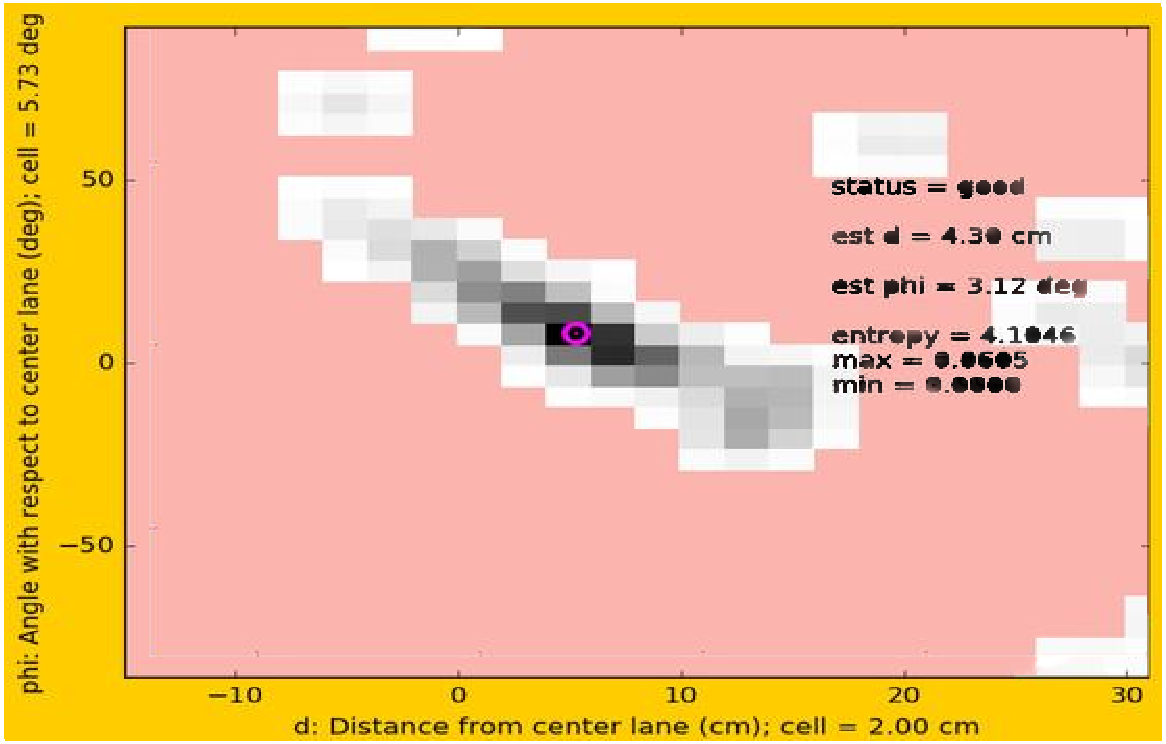
$$\varphi_{new} = \varphi_{old} + \omega \cdot \delta t$$

$$d_{new} = d_{old} + v \cdot \sin(\varphi_{old}) \cdot \delta t$$

Propagate prediction on belief

$$\varphi_{new} = \varphi_{old} + \omega \cdot \delta t$$

$$d_{new} = d_{old} + v \cdot \sin(\varphi_{old}) \cdot \delta t$$

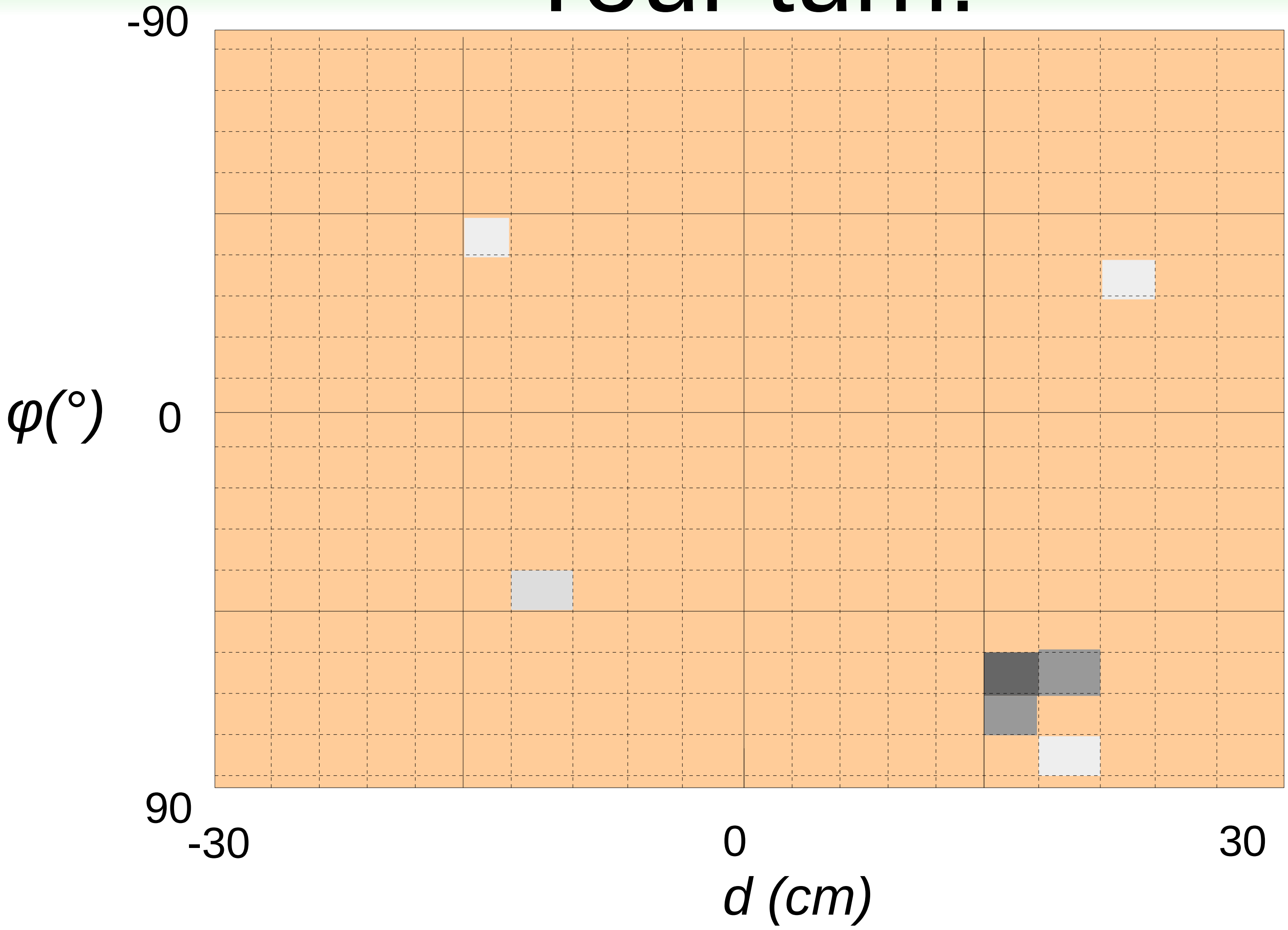


Move of each point with a

$\omega \cdot \delta t$ in the φ direction

$v \cdot \sin(\varphi) \cdot \delta t$ in the d direction

Your turn!

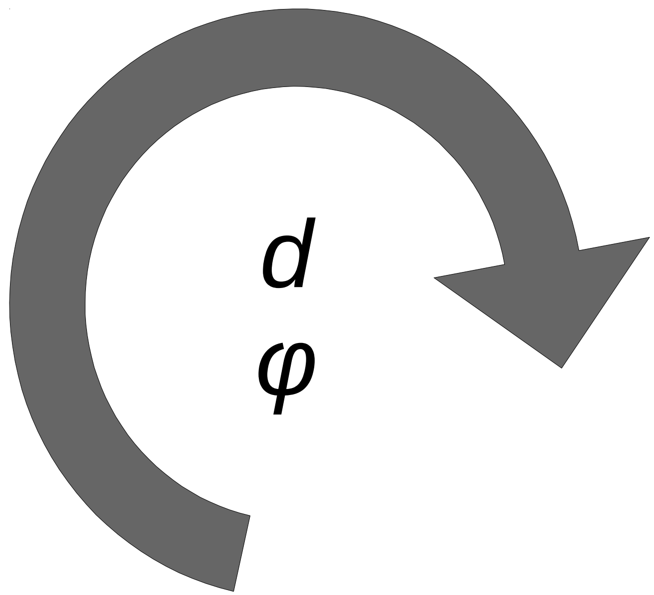


V = 5 m/s   ω = -5°/s   dt = 1 s

Give me the d and phi coordinates of each possible location after the update

Lane filter

Predict
With applied velocities



Update
With position of lanes

Updating

- Idea: combine
  - Our current idea of where we are: the ***prior***

    $$p(d, \varphi)$$

  - The probability of observations m: the ***likelihood***

    $$p(m \mid d, \varphi)$$

- To get the ***posterior***

  $$p(d, \varphi \mid m)$$

Probabilities...

This is Bayes law!

We want this (posterior)

We can evaluate this (likelihood)

We have this (prior)

$$p(d,\varphi \mid m) = \frac{p(m \mid d,\varphi) \cdot p(d,\varphi)}{p(m)}$$

We don't need this (normalization constant)

Probabilities...

We want this

We can compute this

We have this (prior)

$$p(d,\varphi \mid m) = \frac{p(m \mid d,\varphi) \cdot p(d,\varphi)}{p(m)}$$

We don’t need this because we can normalize:

$$\sum_{d} \sum_{\varphi} p(d,\varphi \mid m) = 1$$

Probabilities...

How do we compute the likelihood?

$$p(d,\varphi \mid m) = \frac{p(m \mid d,\varphi) \cdot p(d,\varphi)}{p(m)}$$

Computing the likelihood

$p(m \mid d, \varphi)$ is computed by voting.

1) For each line in *m*, we compute the corresponding *d* and *φ*

2) We add +1 in the corresponding square of a new belief map

3) We normalize the new belief map when we have seen all the lines.

What do we want to estimate?

Distance *d*

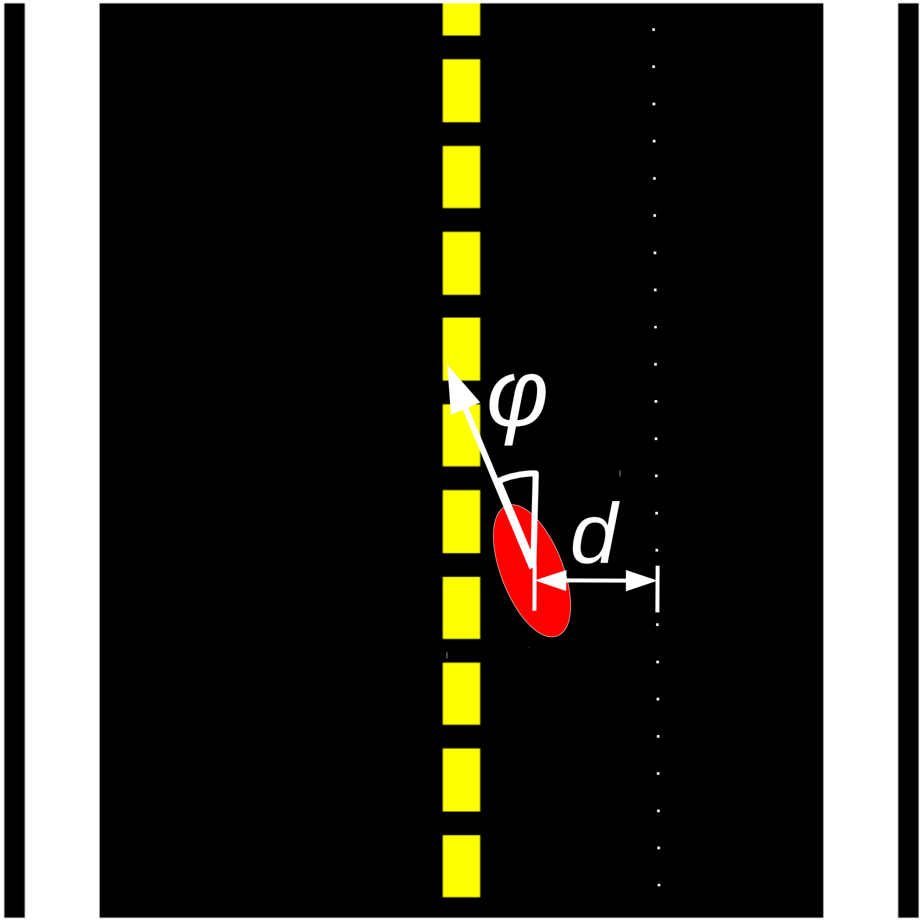
from center of lane

d > 0 : too much on the left

Orientation *φ*

from straight direction

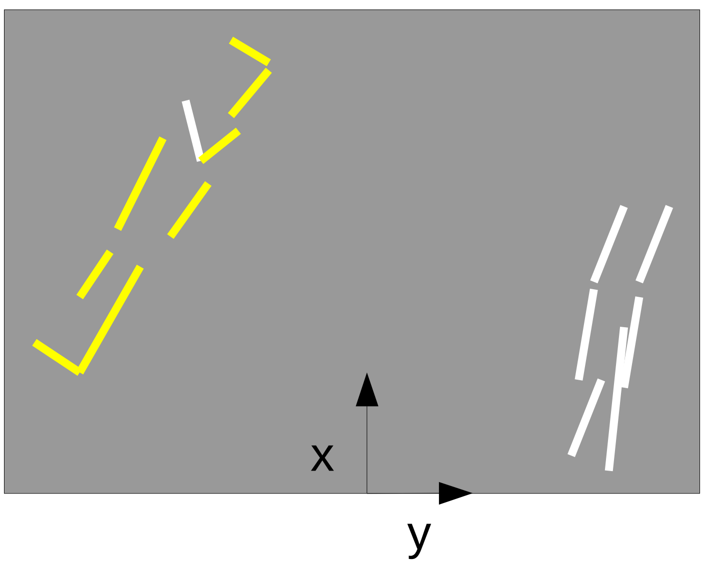
φ > 0 : looking too much to the left

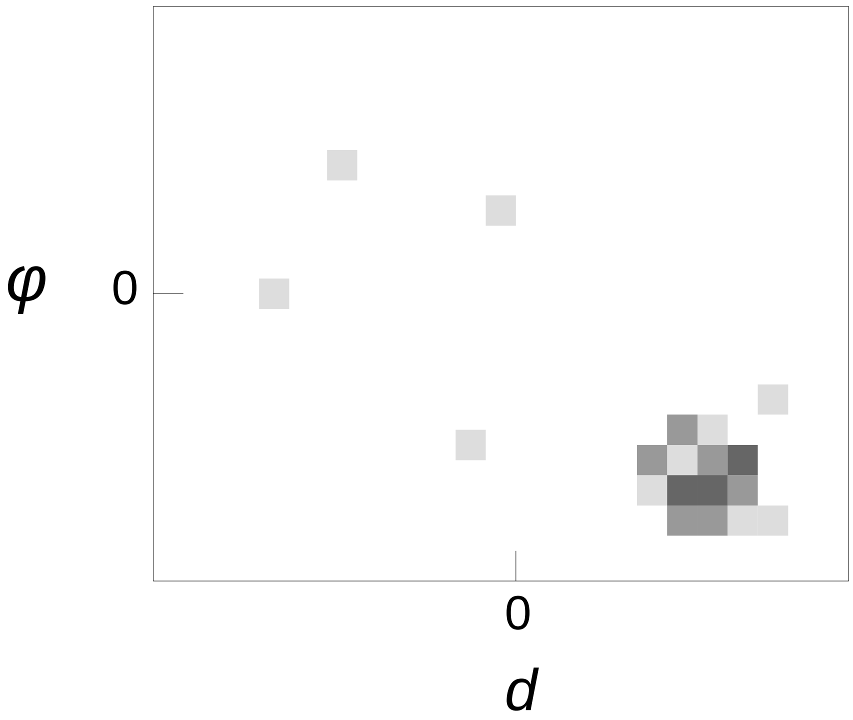


Computing the likelihood

1) For each line in *m*, we compute the corresponding *d* and *φ*
2) We add +1 in the corresponding square of a new belief map
3) We normalize the new belief map when we have seen all the maps.

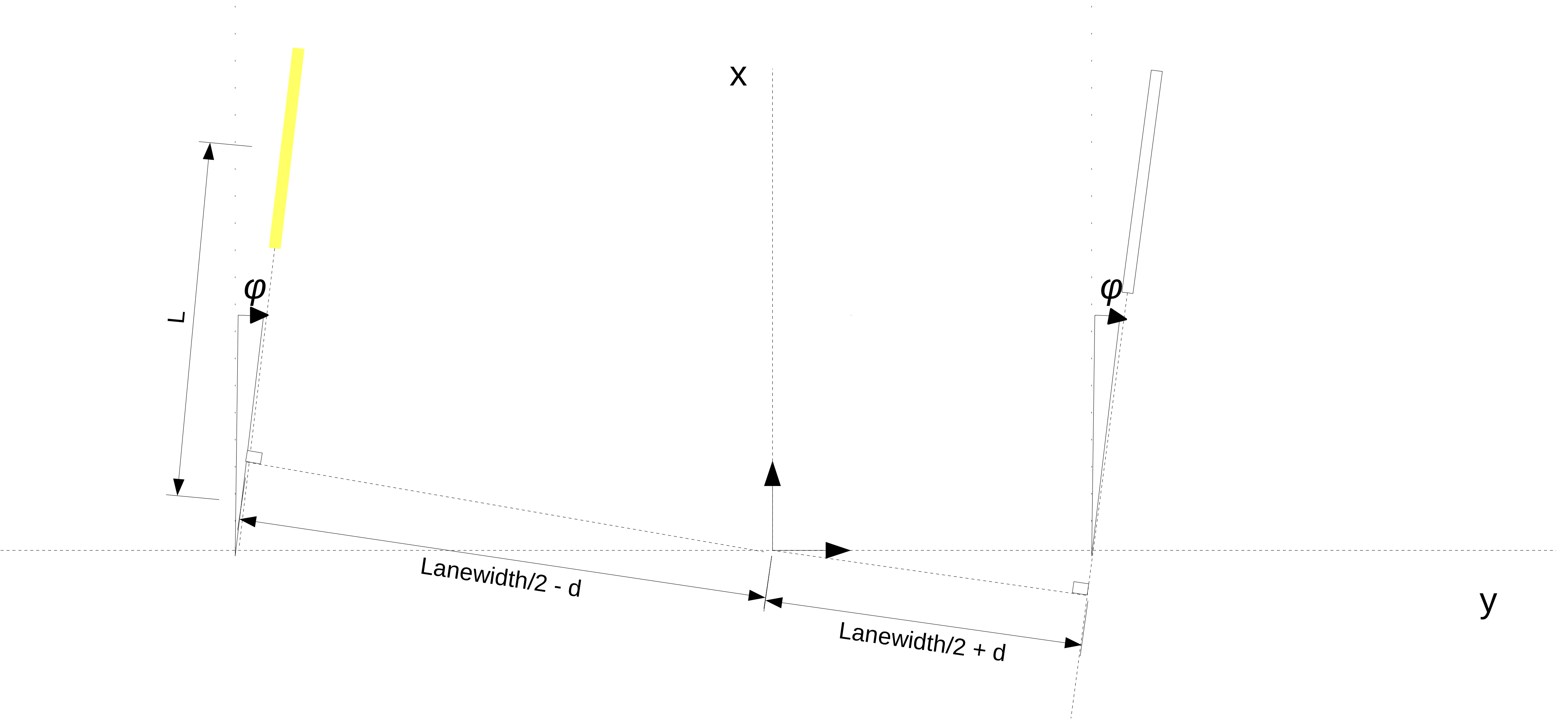
Line position





Computing d and φ for each line

1) For each line in *m*, we compute the corresponding *d*, φ and *L*



Your turn!

If the line is white, it has to be on the right.

If the line is yellow, it has to be on the left.

If the line is outside the color band, we have to reduce the computed d by the width of the lane (white or yellow).

Your turn!

How to do this?

Draw, think, figure out the maths
and implement it!

In function def generateVote(self,segment), compute d_i, phi_i and l_i.
You have access to:

- segment.points[0].x, segment.points[0].y, segment.points[1].x, segment.points[1].y are the x and y coordinates of points 0 and 1 of the segment.

- segment.color = segment.WHITE or segment.YELLOW

- self.lanewidth is the width of the lane

- self.linewidth_yellow and self.linewidth_white are the witdths of yellow and white lines

Your turn!

In function def generateVote(self,segment), compute d_i, phi_i and l_i.
You have access to:

- segment.points[0].x, segment.points[0].y, segment.points[1].x, segment.points[1].y are the x and y coordinates of points 0 and 1 of the segment.

- segment.color = segment.WHITE or segment.YELLOW

- self.lanewidth is the width of the lane

- self.linewidth_yellow and self.linewidth_white are the witdths of yellow and white lines

Computing the likelihood

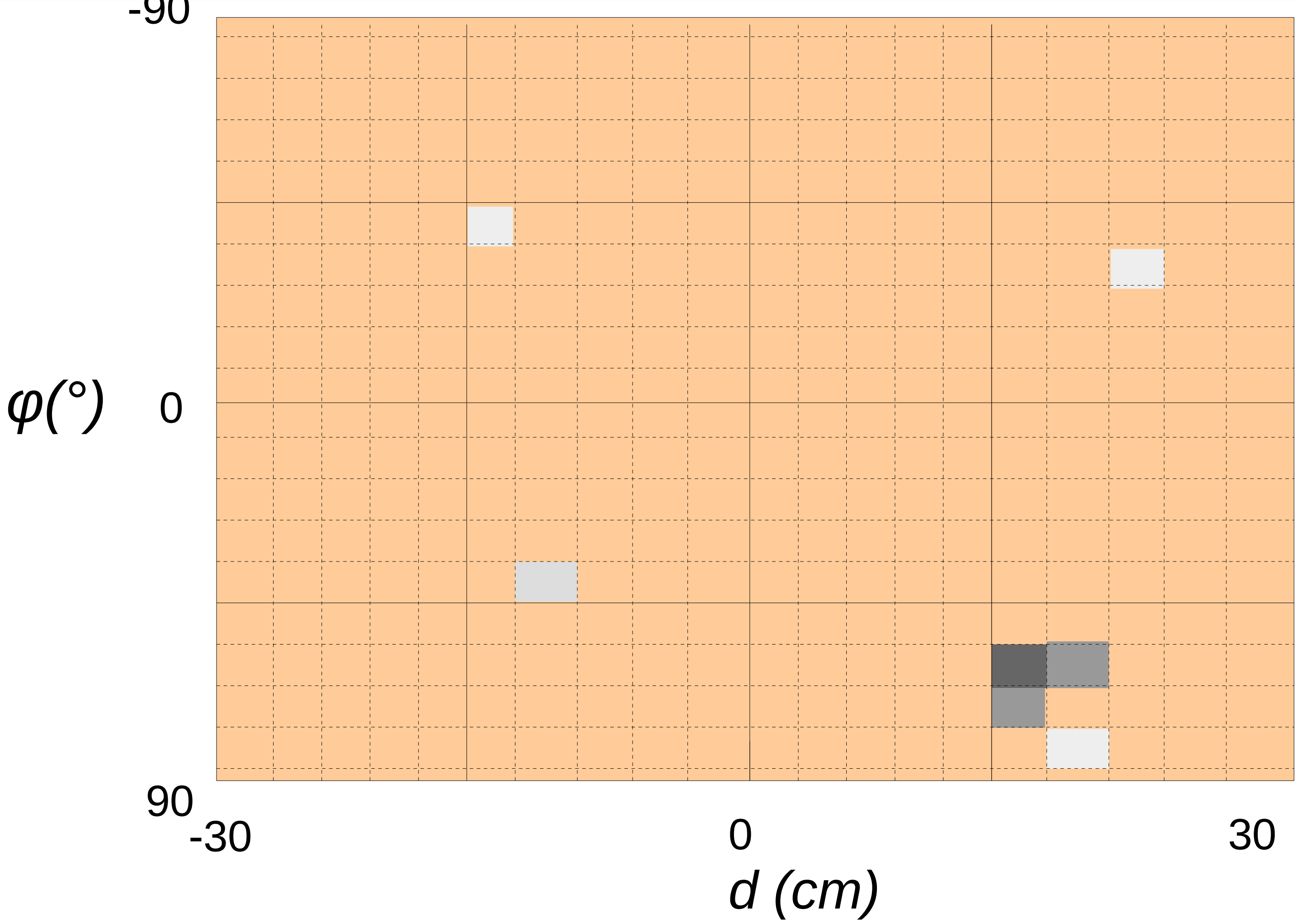
$p(m \mid d, \varphi)$ is computed by voting.

1) For each line in *m*, we compute the corresponding *d* and *φ*

2) We add +1 in the corresponding square of a new belief map

3) We normalize the new belief map when we have seen all the maps.

Your turn!



1) Pink: 0 White: 1 Light gray: 2 Dark gray: 3 Black: 4

What values do we have in the belief map after normalizing?

Probabilities...

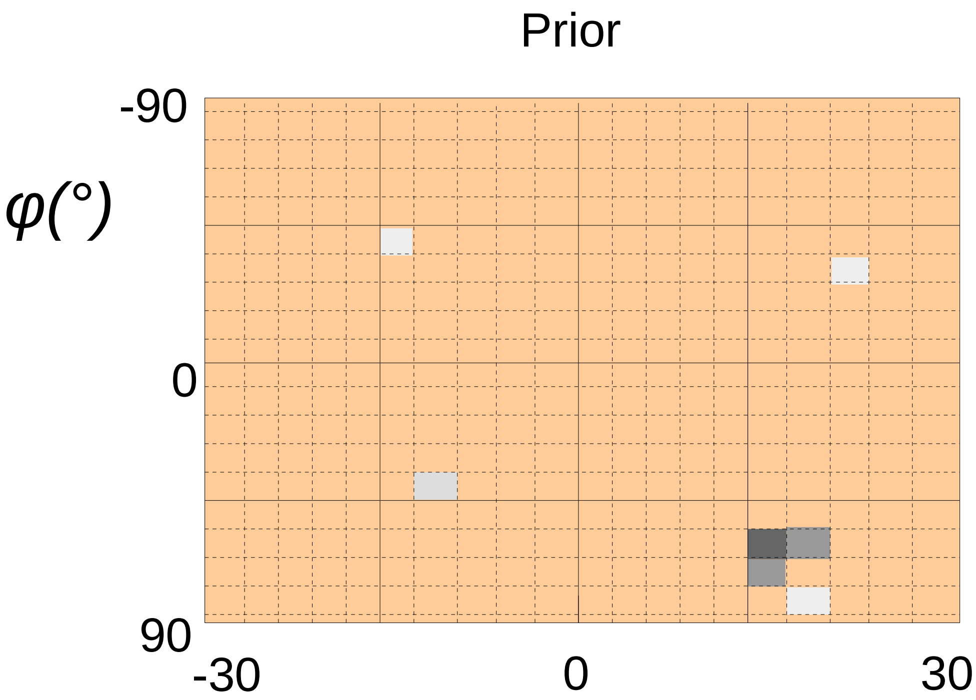
Now we have this

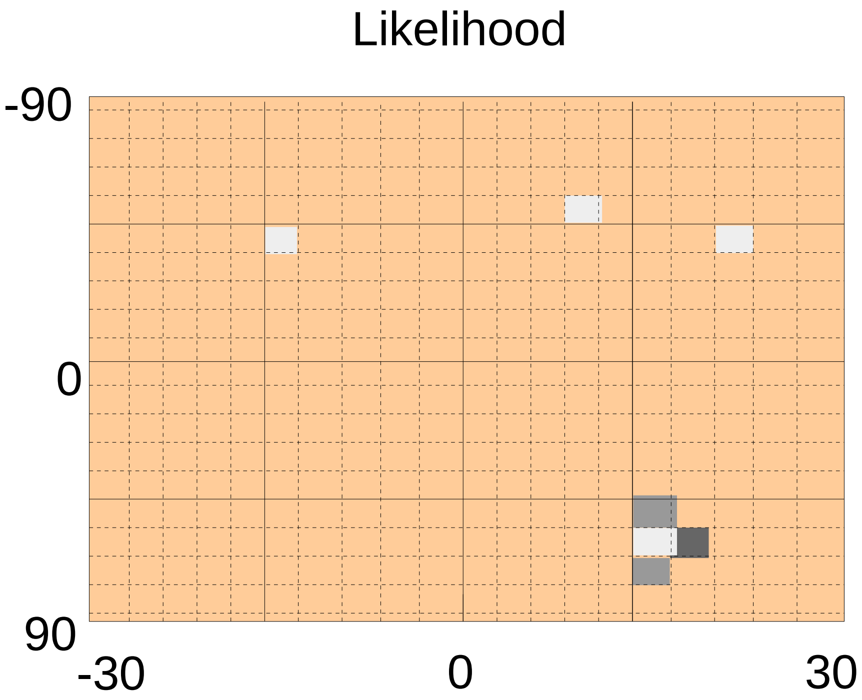
And this (which is the previous belief)

$$p(d,\varphi \mid m) = \frac{p(m \mid d,\varphi) \cdot p(d,\varphi)}{p(m)}$$

How do we get the new d, φ map?

Your turn!





What does the posterior belief map look like?