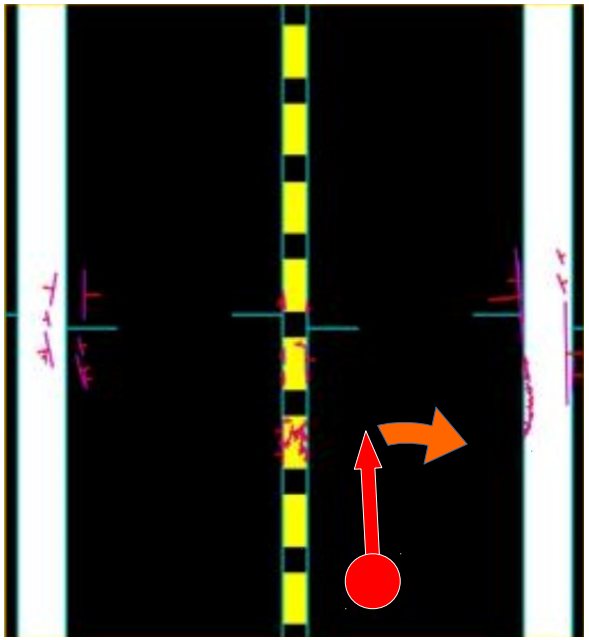
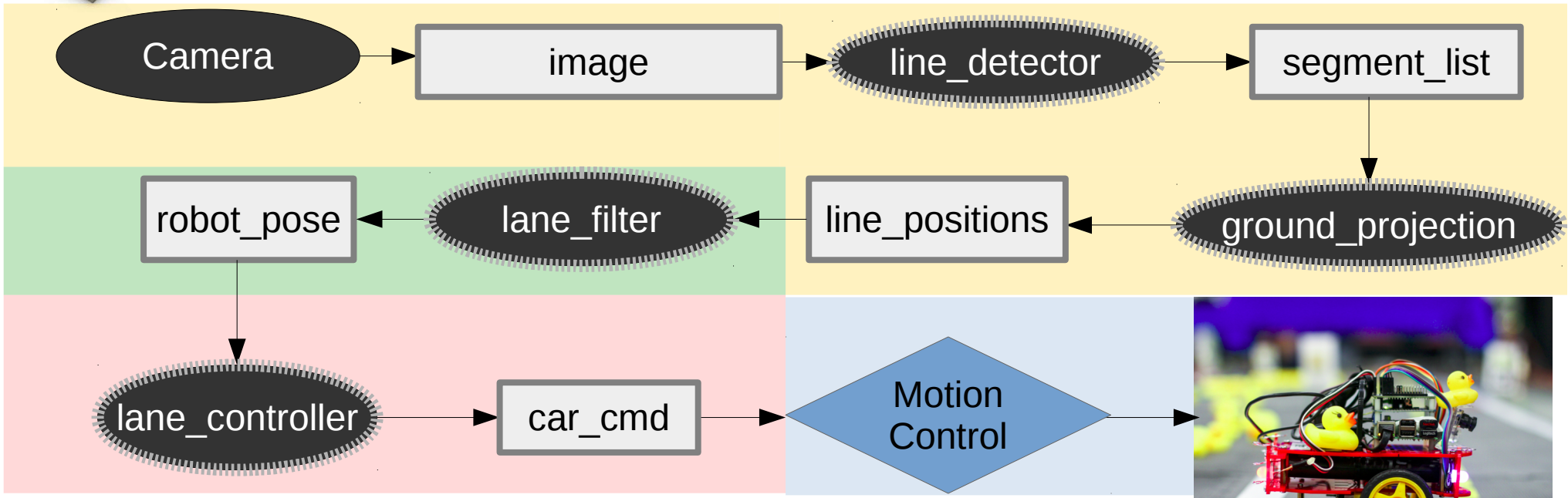
Ground projection

Following lanes – Overview





Perception : Line detection

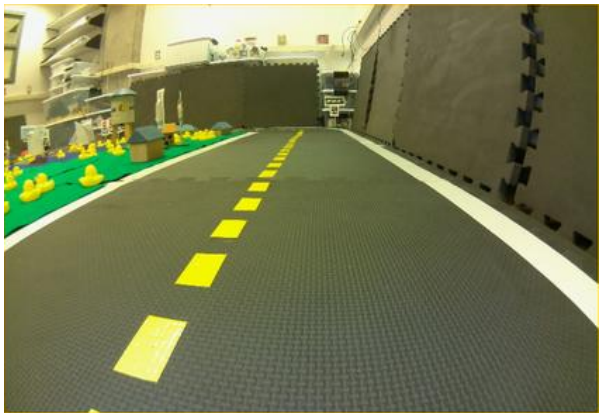
Sensing

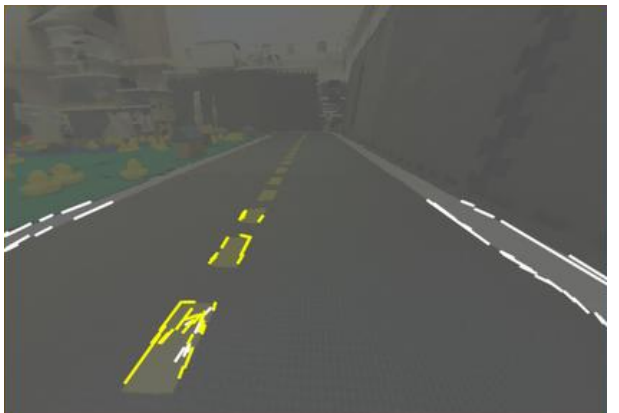
Raw data

Information Extraction

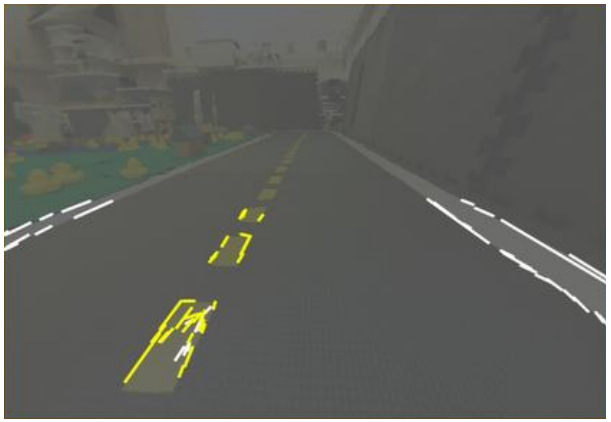
It’s a pillar!

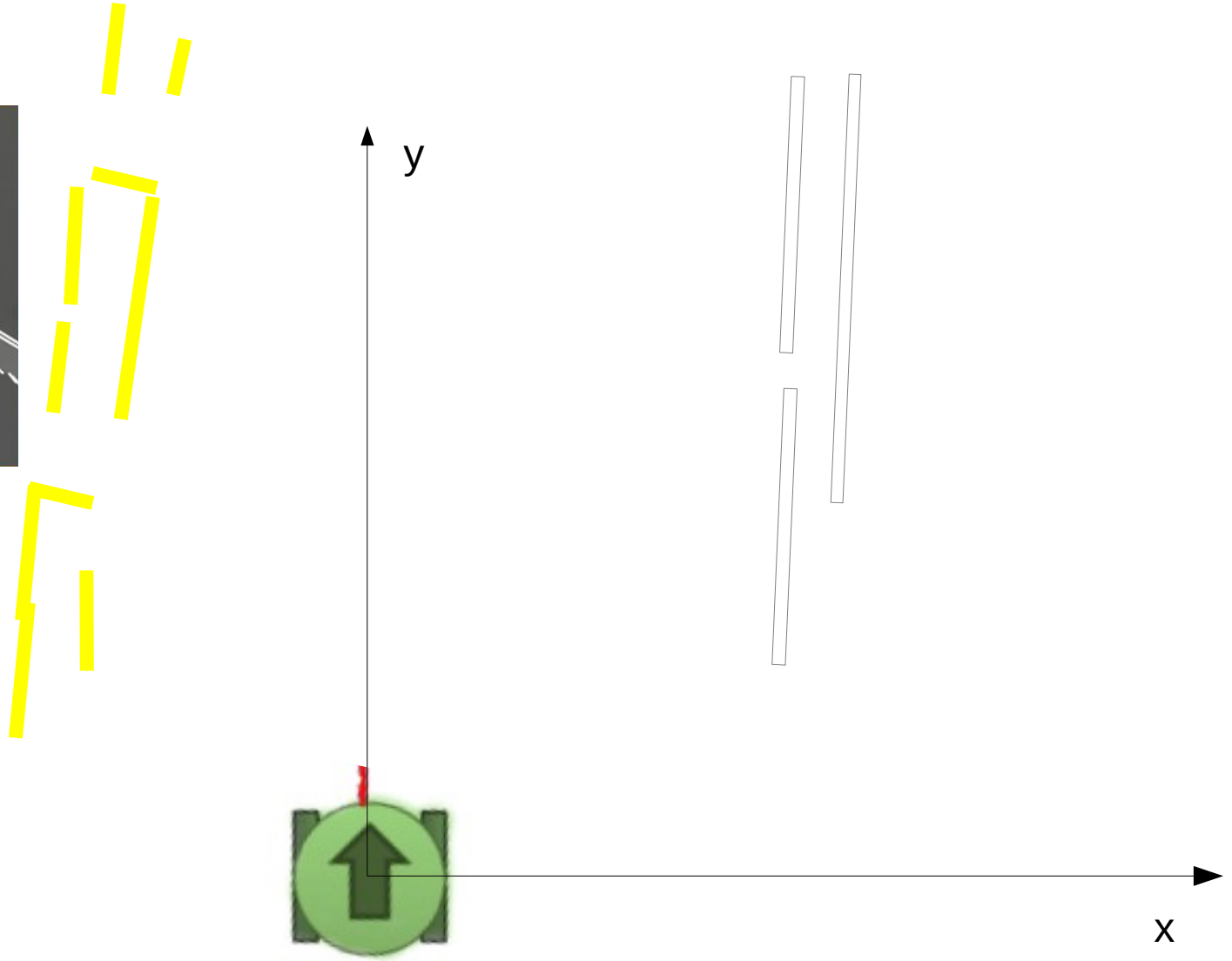
Here are the lines





Perception : Ground projection



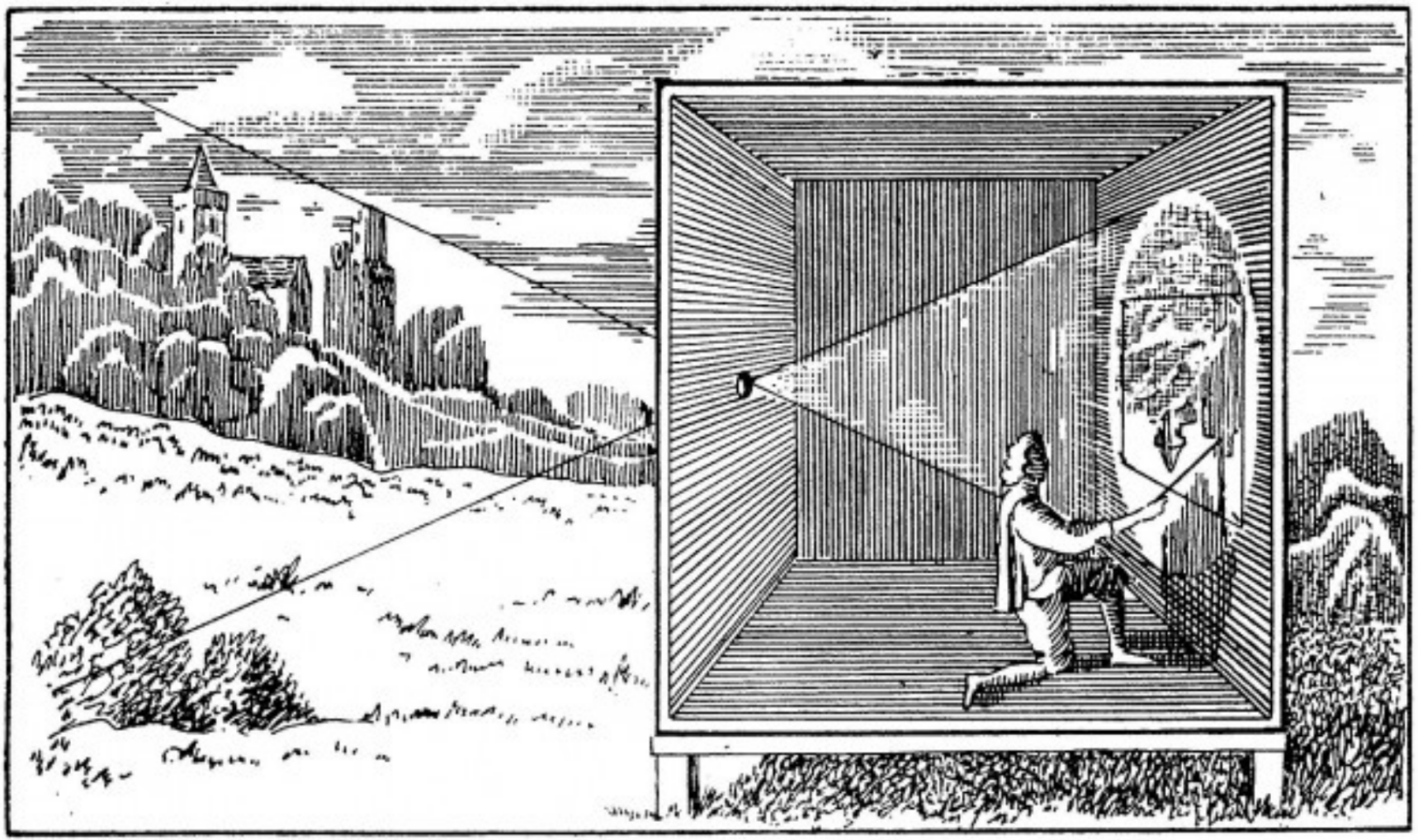


Where are the lines with respect to me?

How does a camera work?

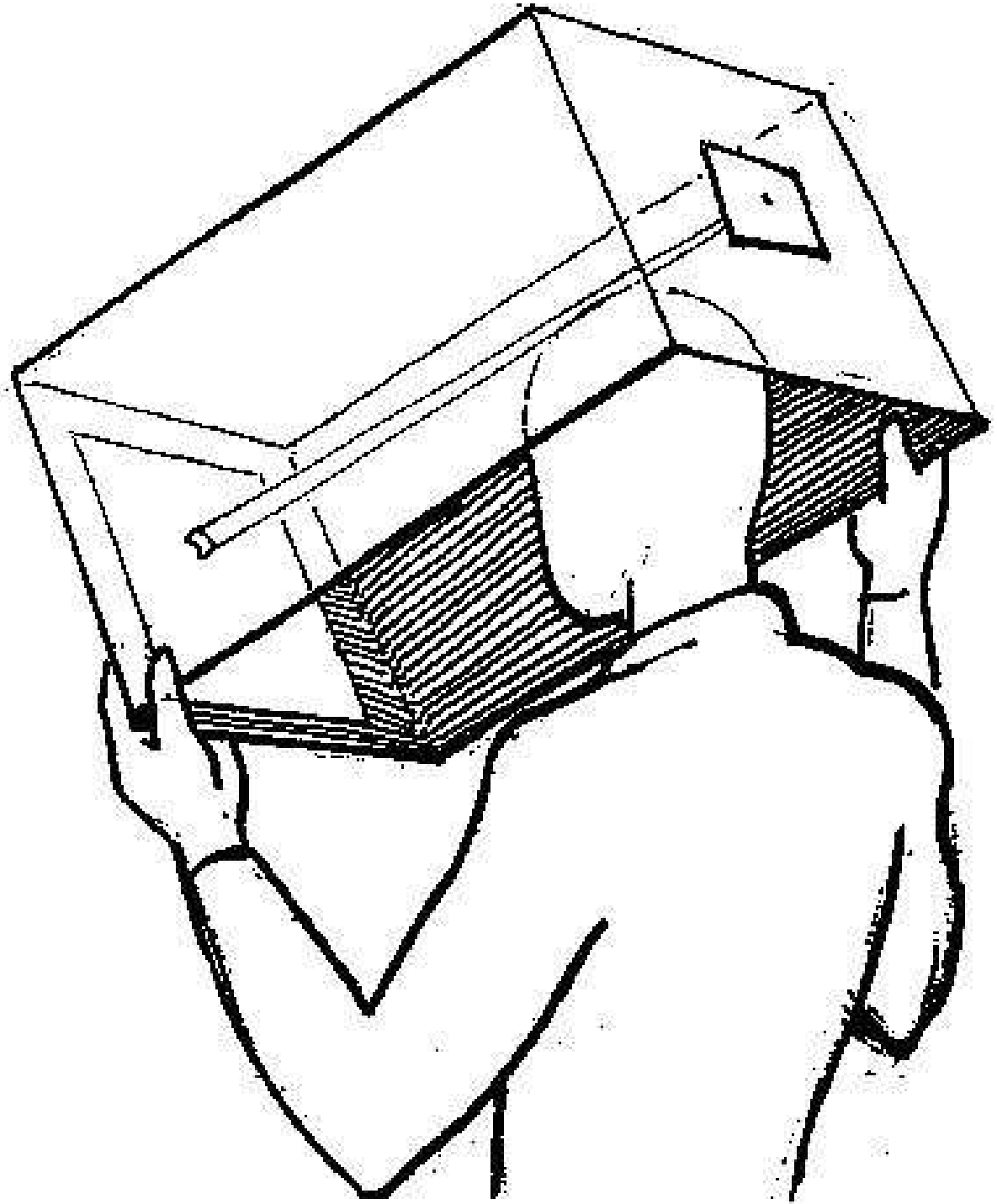
A sensor (CMOS or CCD) : a matrix of a lot of pixels. Each pixel detects the light and colors hitting it.

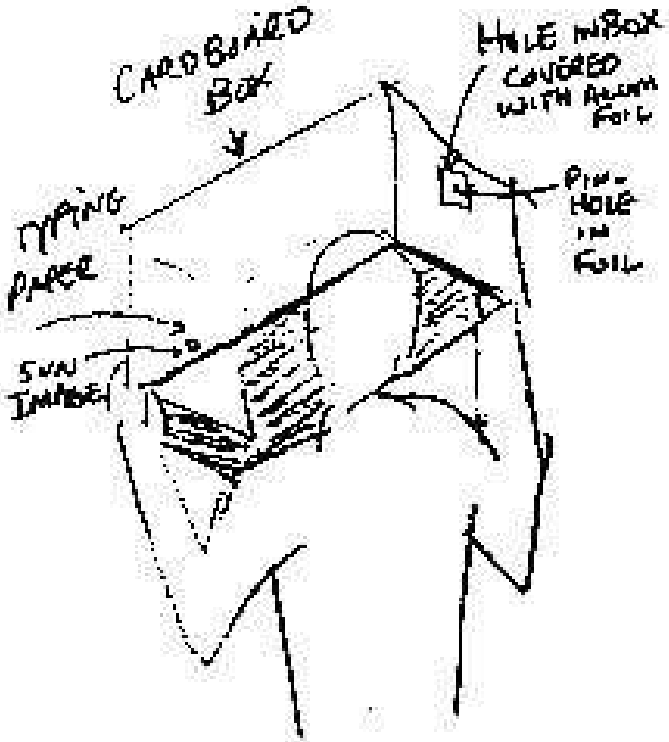
Camera obscura



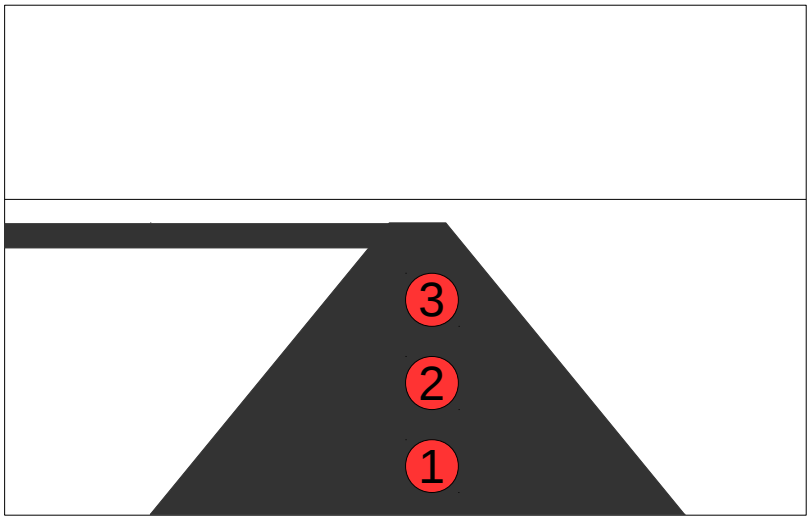
Drawing aid for artists
(da Vinci, 1452–1519)

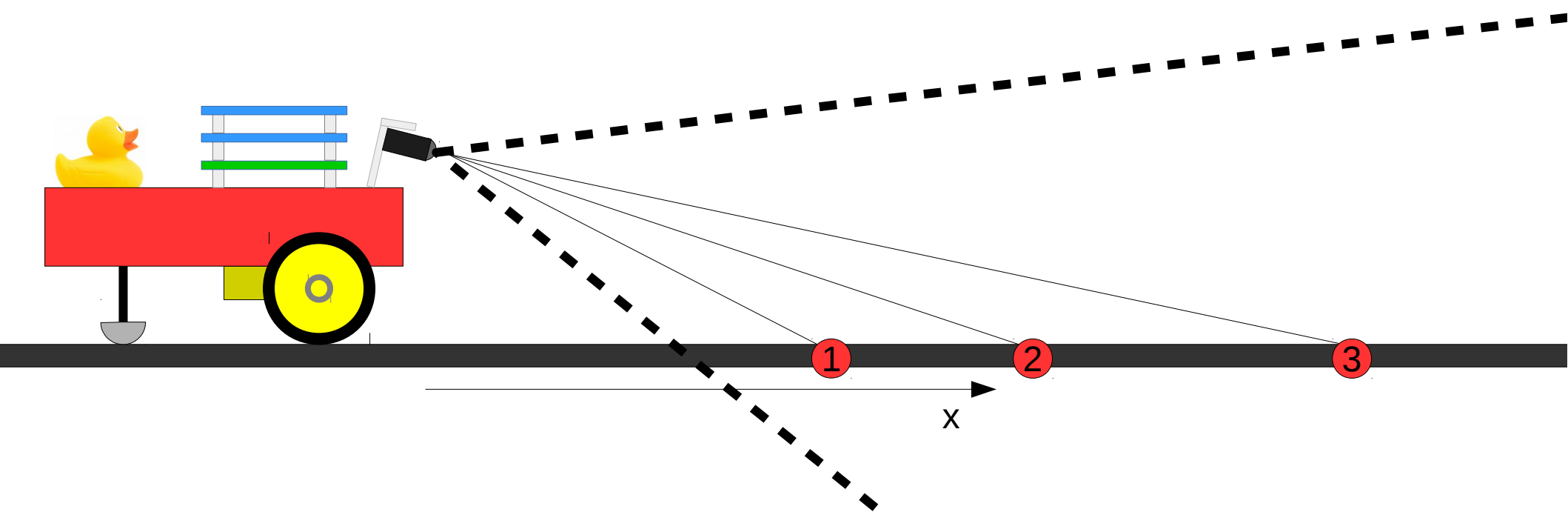
SAFE WAY TO VIEW ECLIPSE





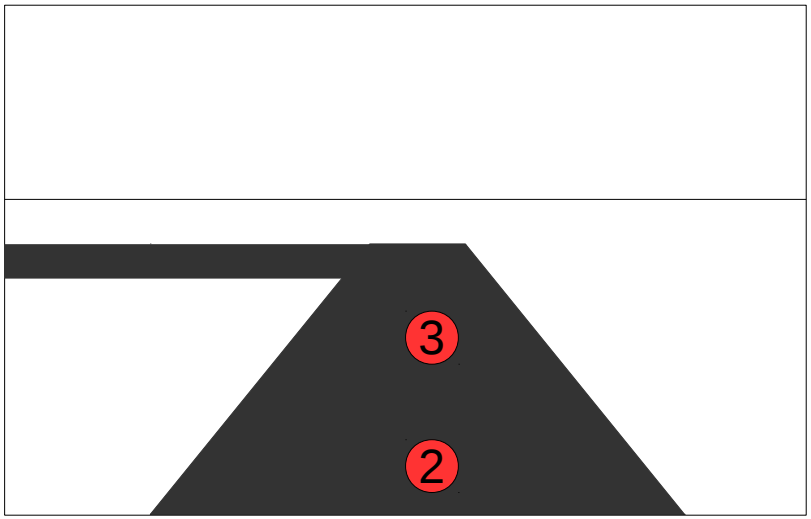
World to image

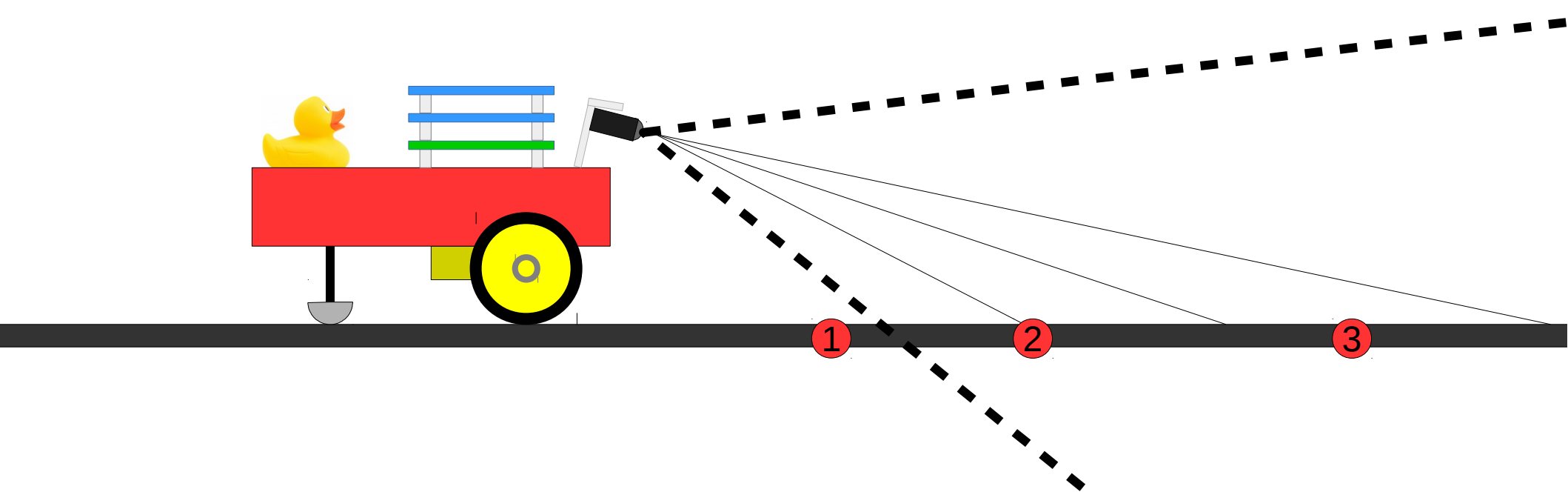




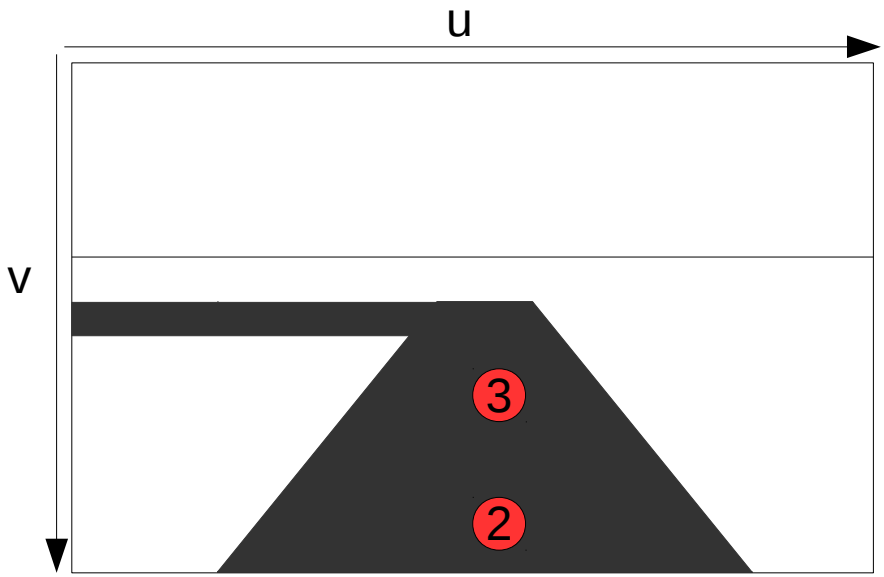
World to image

Notice that point 2 is now at the same pixel that point 1 was before. Normal: they are at the same position with respect to the camera!





What does this mean?



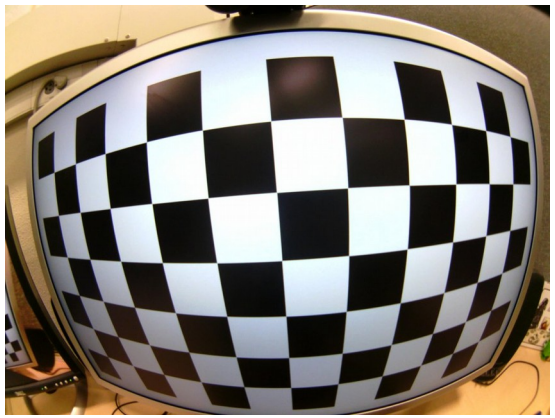
If a point (x, y) is on the ground (z = 0), there is a unique function

$$(u, v) = f(x, y)$$

such that (u, v) represent the pixels of the point on the camera image.

$$(u, v) = f(x, y)$$

According to you, what influences on this function?



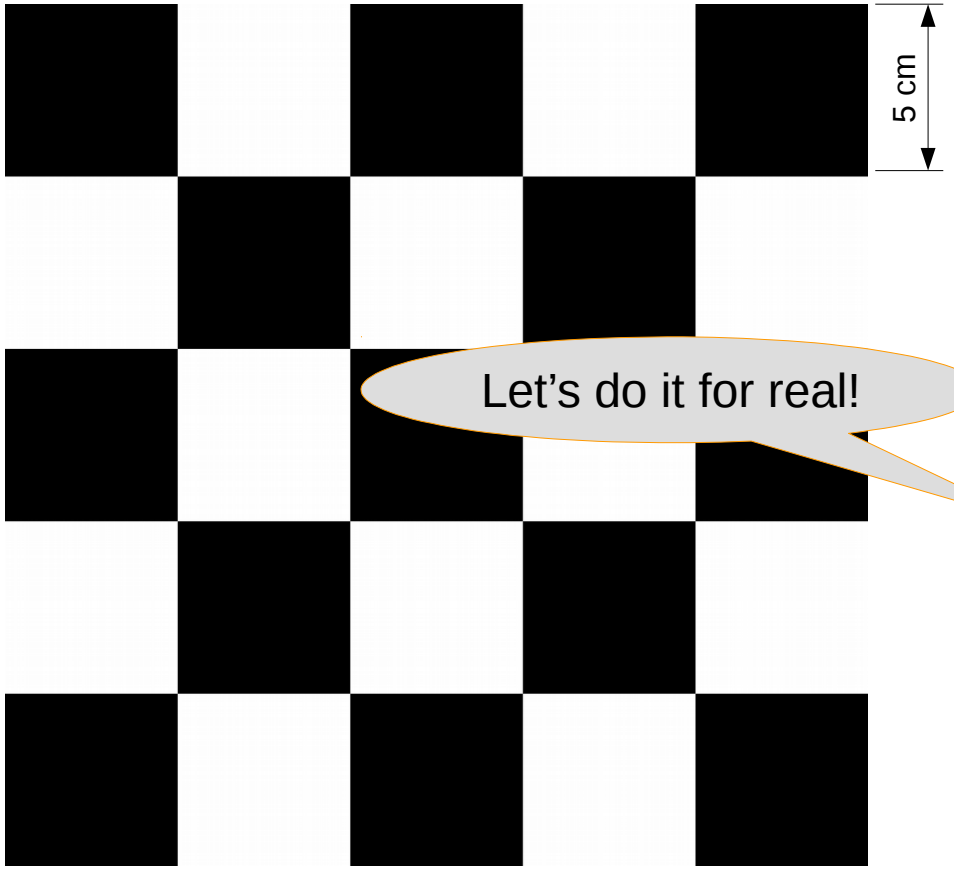
Intrinsic camera parameters
(lens effect, opening angle, ...)

Position/orientation of the camera with respect to the ground and the DuckieBot

How do we get this function?

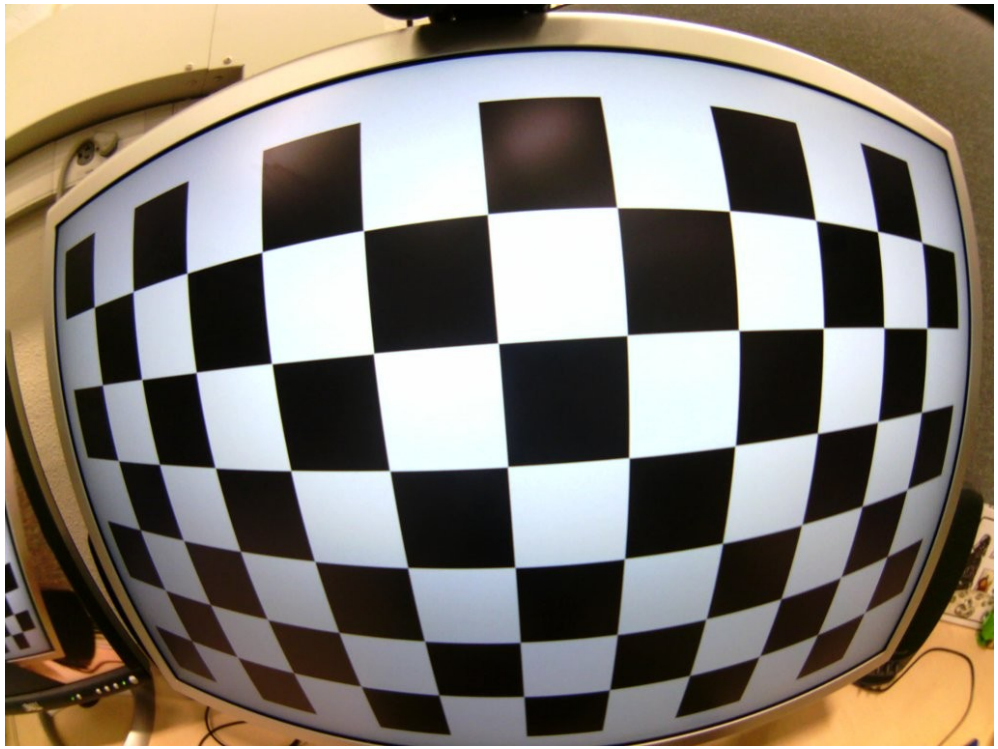
We calibrate!

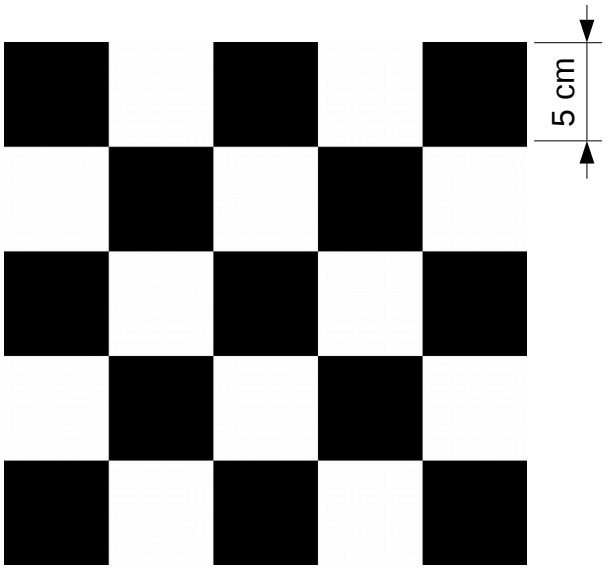
Calibration: take something we KNOW and see what does the camera give us.
Example: a checkboard.



Step 1: Internal camera parameters

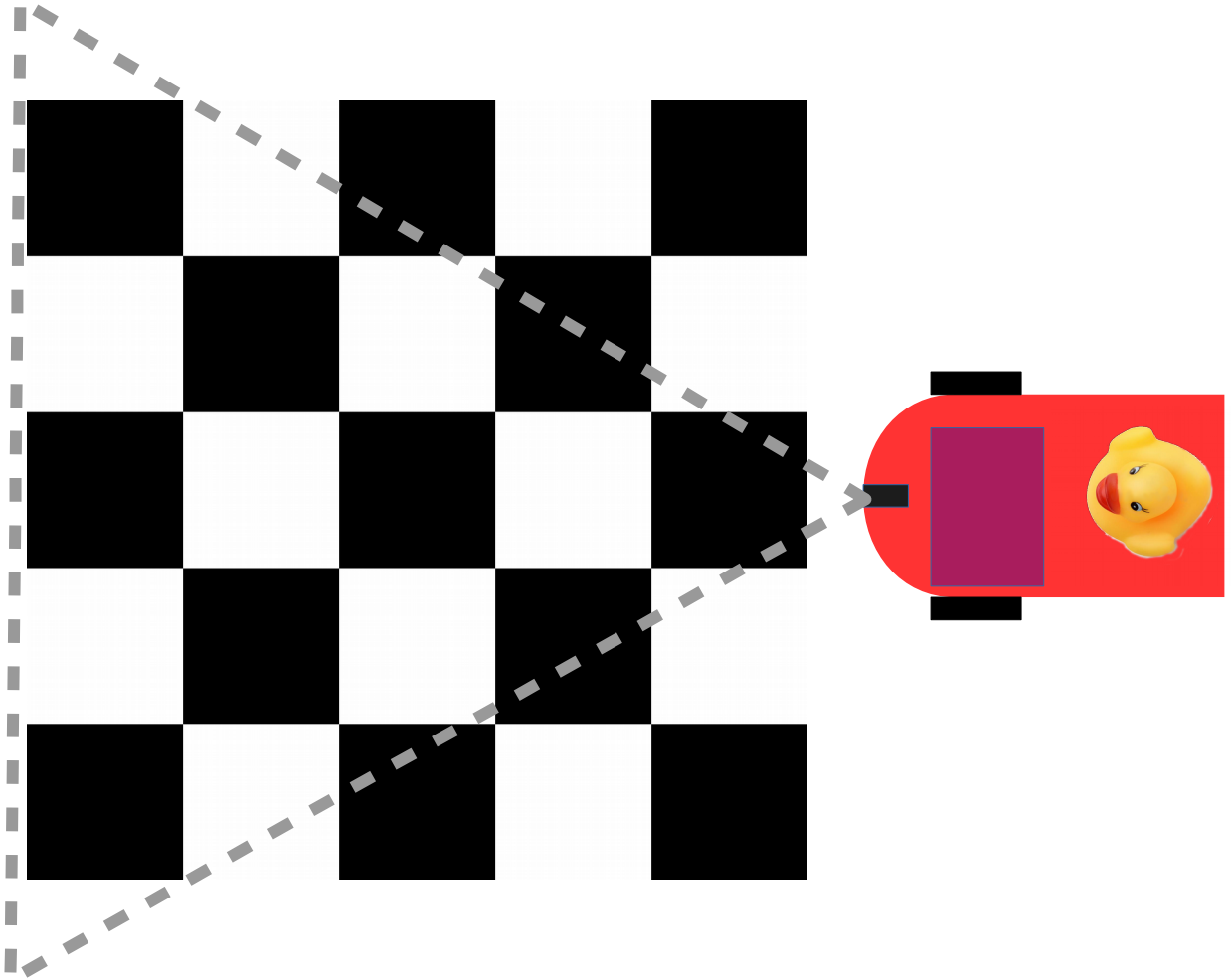
Move the checkboard in all directions (up-down, left-right, close-far, turn it).
This allows to correct the weird effects on the sides!





Step 2: Position and orientation of the camera

Put the bot in a known position in front of the checkboard.
This allows to know exactly the projection of each pixel on the ground!



Alright, all this is done.

What did we get from this?

$$(u, v) = f(x, y)$$

Great!
But... why did we want this again?

Remember what we are looking for?

What we have is *(u, v)*: position of lines on the image.
What we want is *(x, y)*: position of lines on the ground.

$$(u, v) = f(x, y)$$

Knowing *f* allows us to do the inverse operation.

$$(x, y) = f^{-1}(u, v)$$

How does it work?

$$(x, y) = f^{-1}(u, v)$$

The calibration gives us the “homographic” matrix *H*.
We can describe f^{-1} *in* 2 steps:

1)
$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = H \times \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

2)
$$x = a/c$$
$$y = b/c$$
$$z = 0$$

Your turn!

Implement 1) and 2) in the the pixel2ground function of GroundProjection.py

1)
$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = H \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

2)
$$x = a/c$$
$$y = b/c$$
$$z = 0$$

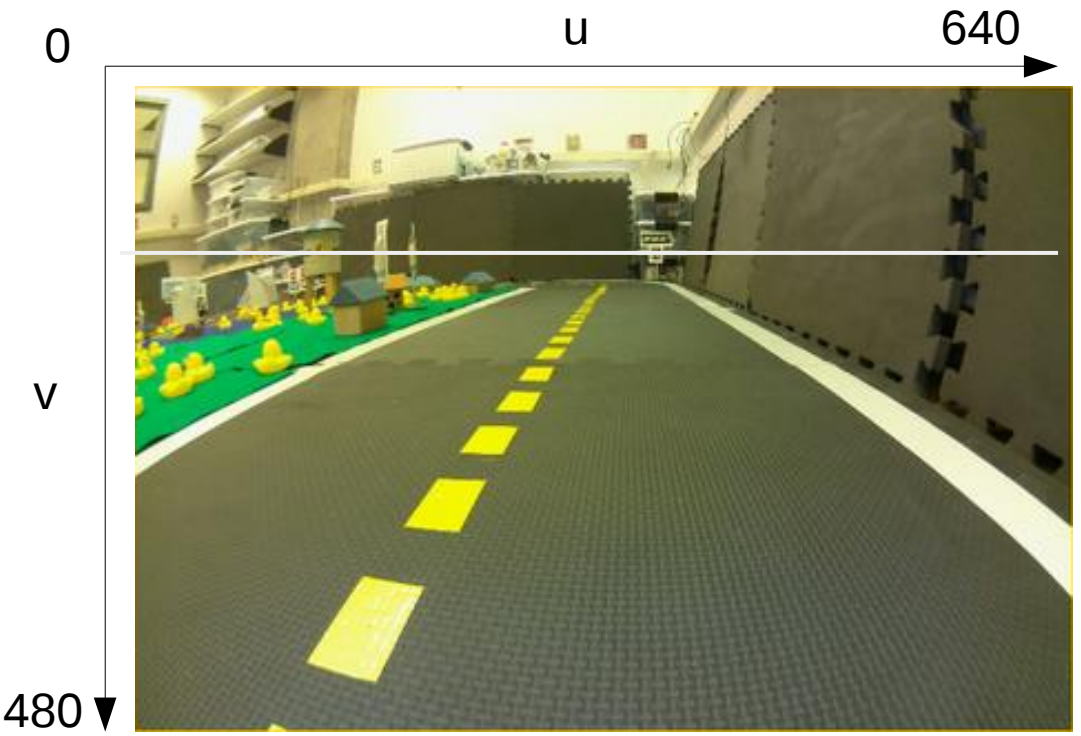
Still your turn!

H that has been computed for a very similar camera.

Experiment with it to tell me: from which v is the horizon?

H

$$\begin{pmatrix} -0.0000127373 & -0.0002421778 & -0.1970125 \\ 0.001029818 & -0.00001578045 & -0.337324 \\ -0.000108811 & -0.007584862 & 1 \end{pmatrix}$$



Think further

If you did not know the point was
on the ground, what would you be able to know from *u* and *v*?