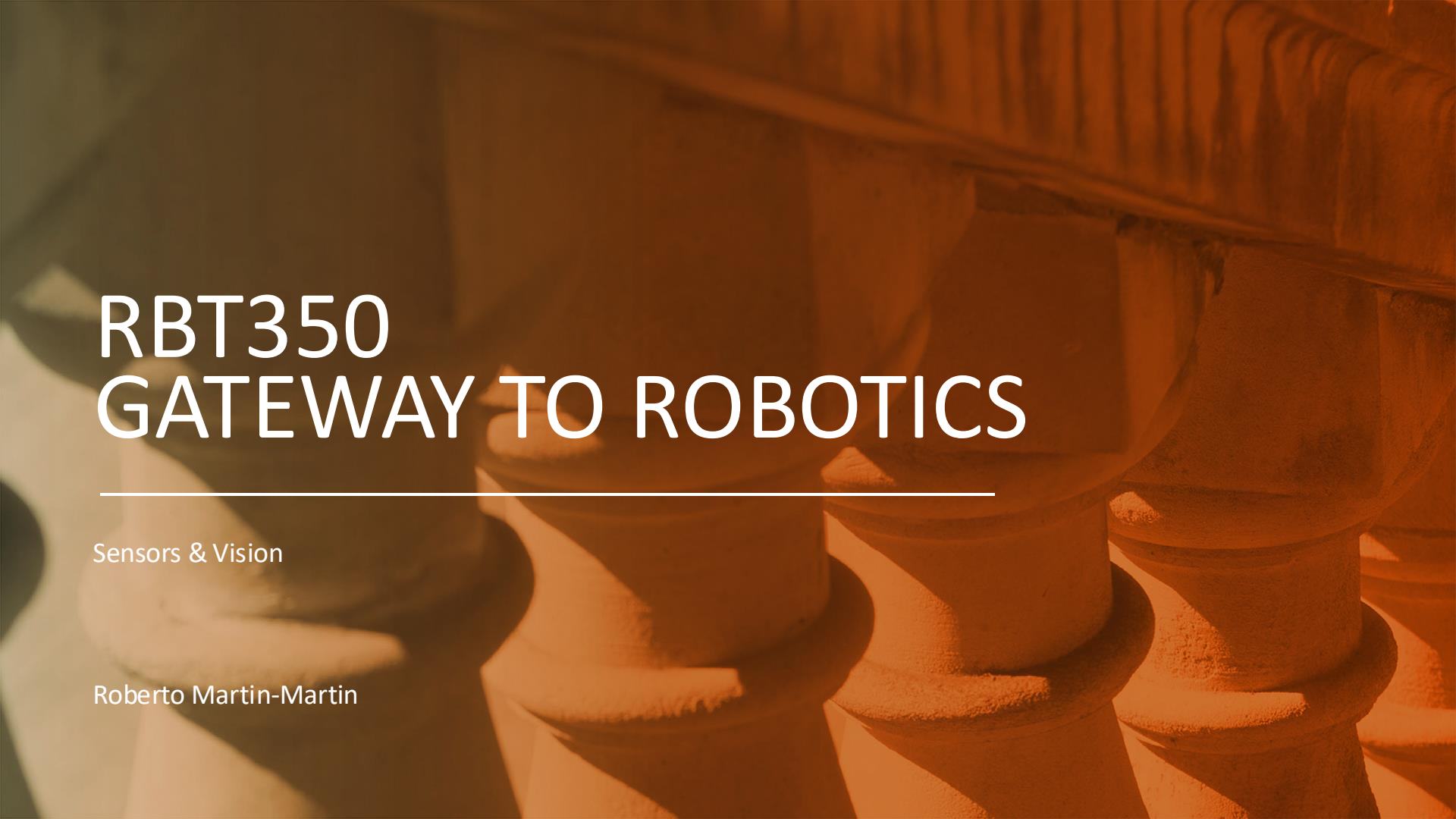
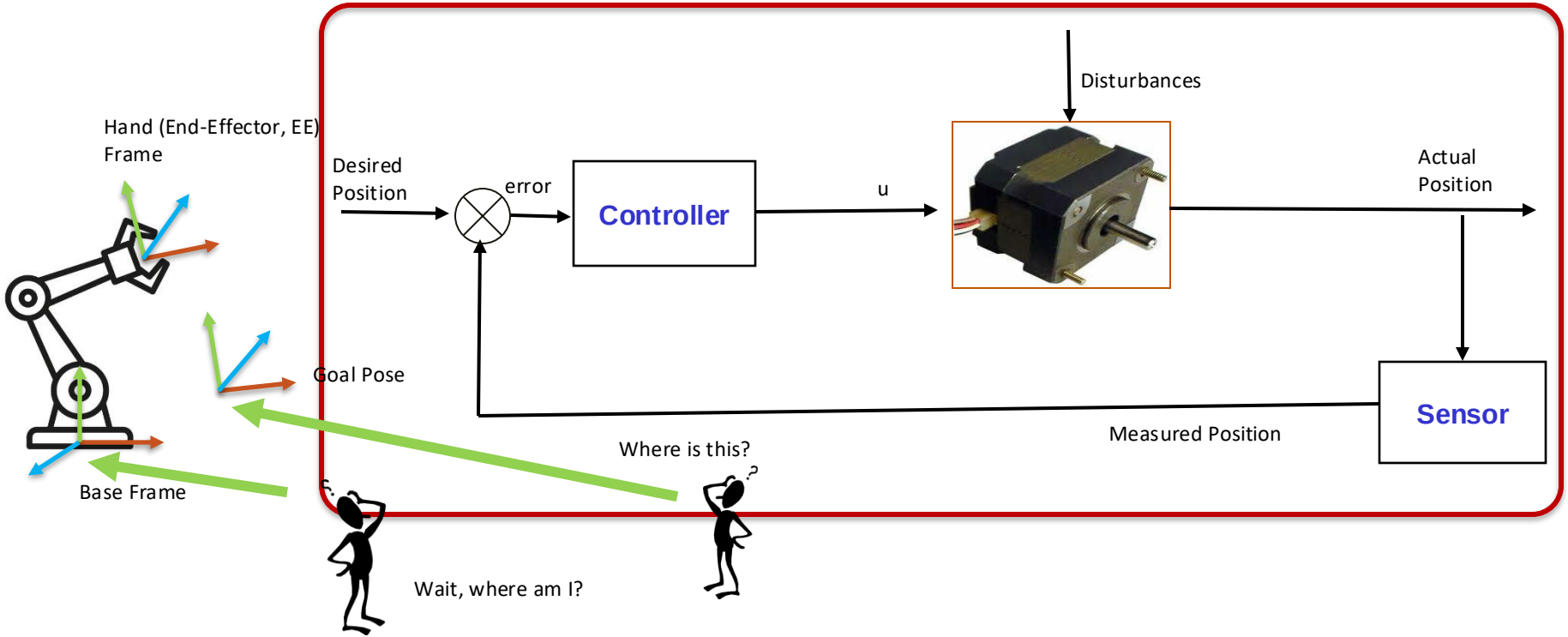


RBT350
GATEWAY TO ROBOTICS

Sensors & Vision

Roberto Martin-Martin

How do we know where we are and where do we need to move?

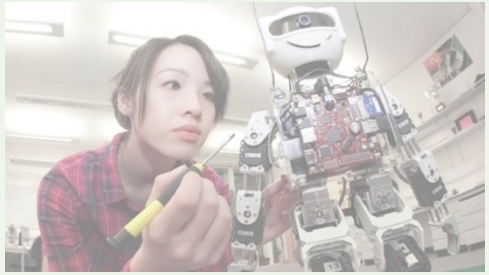


We have everything to move a robot, but…

But to move it where?

Course Content

How to build a robot



weeks 1-4

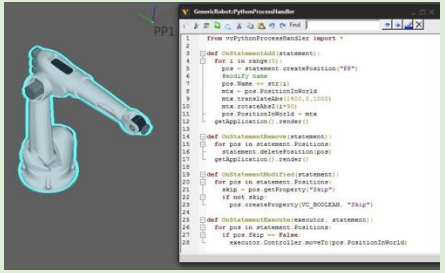
- Mechanics of Materials
- Electronics (A/D)
- State Machines
- Mechatronics

How to make a robot move

weeks 5-9

- PID Control
- Physics of Movement
- Transformations & Poses
- F/I Kinematics
- Dynamics

How to program a robot



weeks 10-13

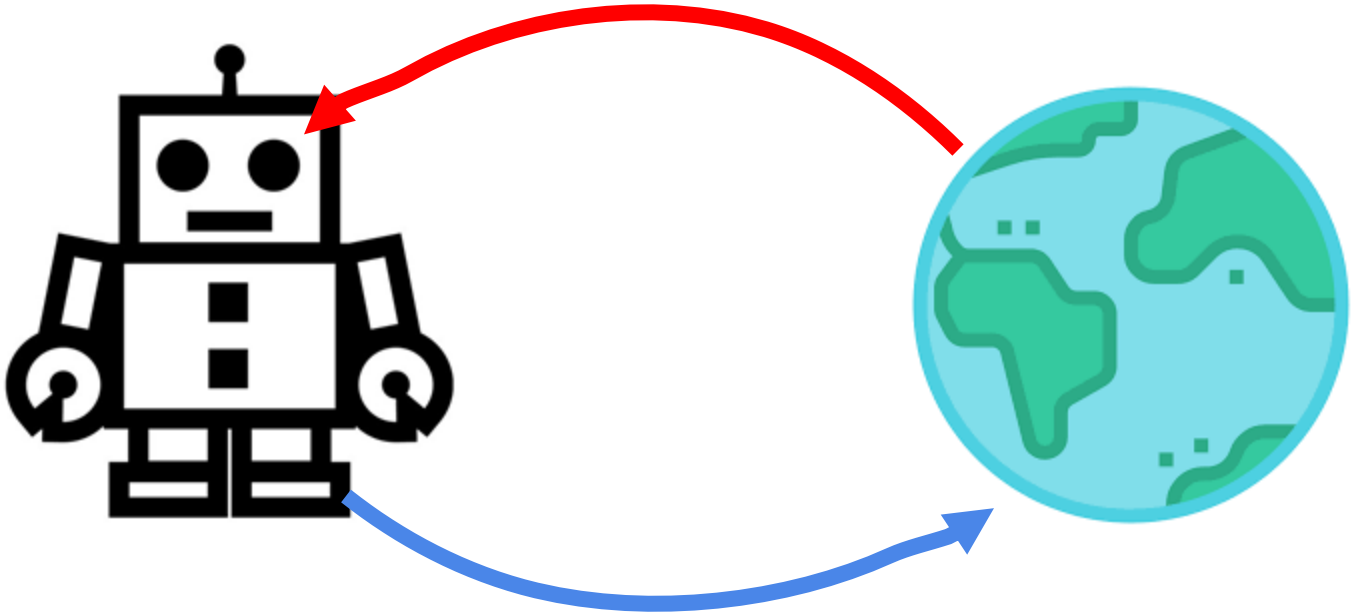
- Vision
- Graph Search
- Motion Planning
- Machine Learning

How to program our robot?

- We have:
  - a definition of the task
  - a set of methods to move the robot to desired configurations
- We need to:
  - Tell the robot where and how to move to achieve the task (what configurations?)
    - Motion planning
  - We would like for the robot to do this in an autonomous way
    - Perception
    - Machine learning

What will you learn today?

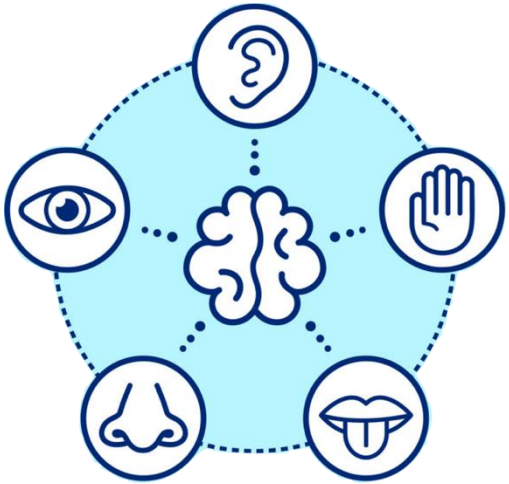
- Robot Sensing
- Basics of Computer Vision
  - Cameras
  - Image formation
  - Projective Geometry
  - Calibration



A robot acts in the world and **observes it**

Robotic Sensors

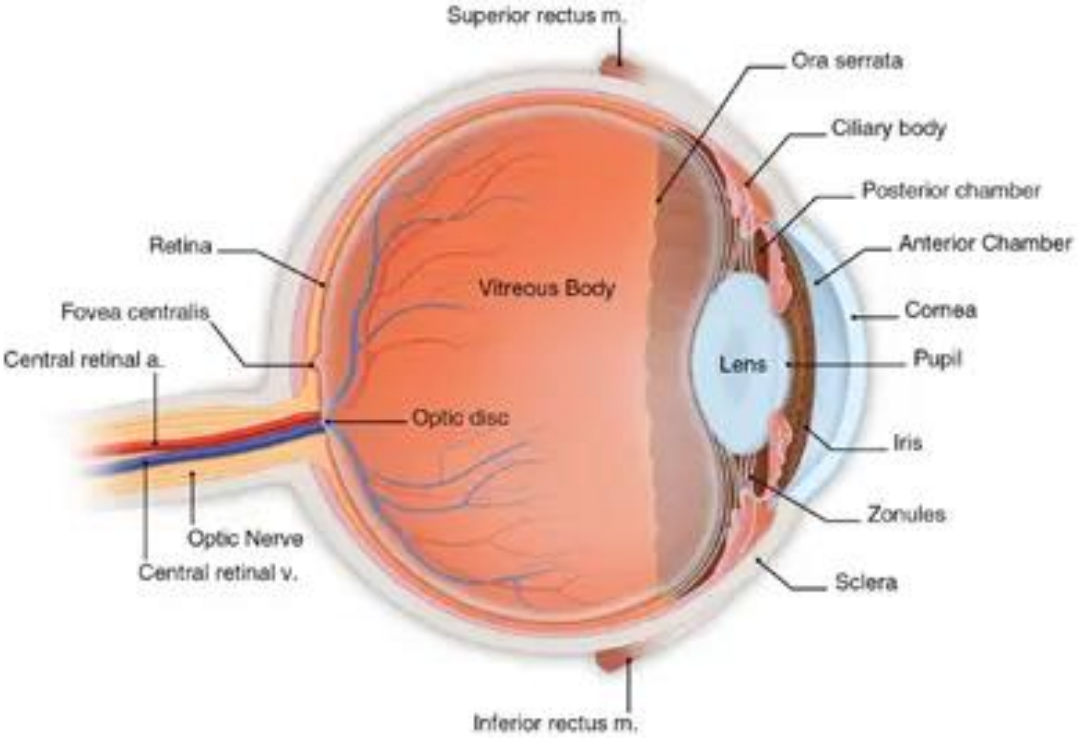
Observing the physical world through multimodal senses

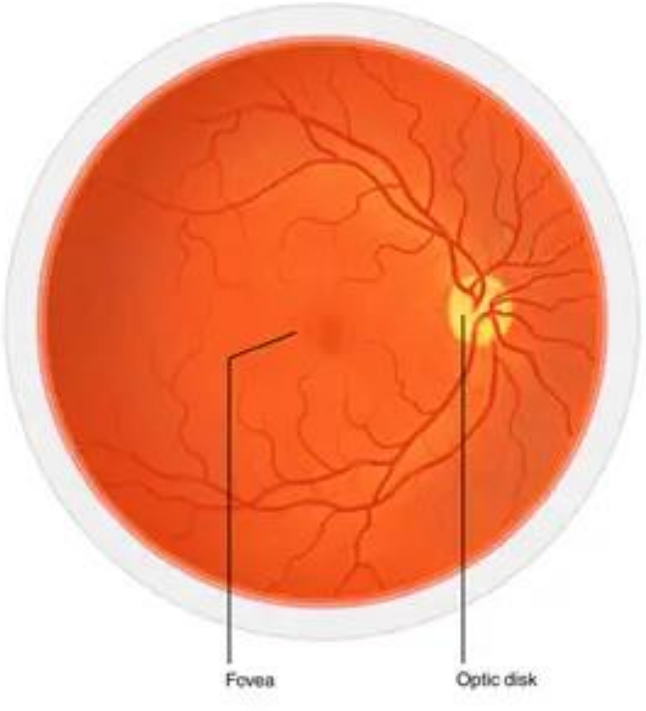


Some Vocabulary

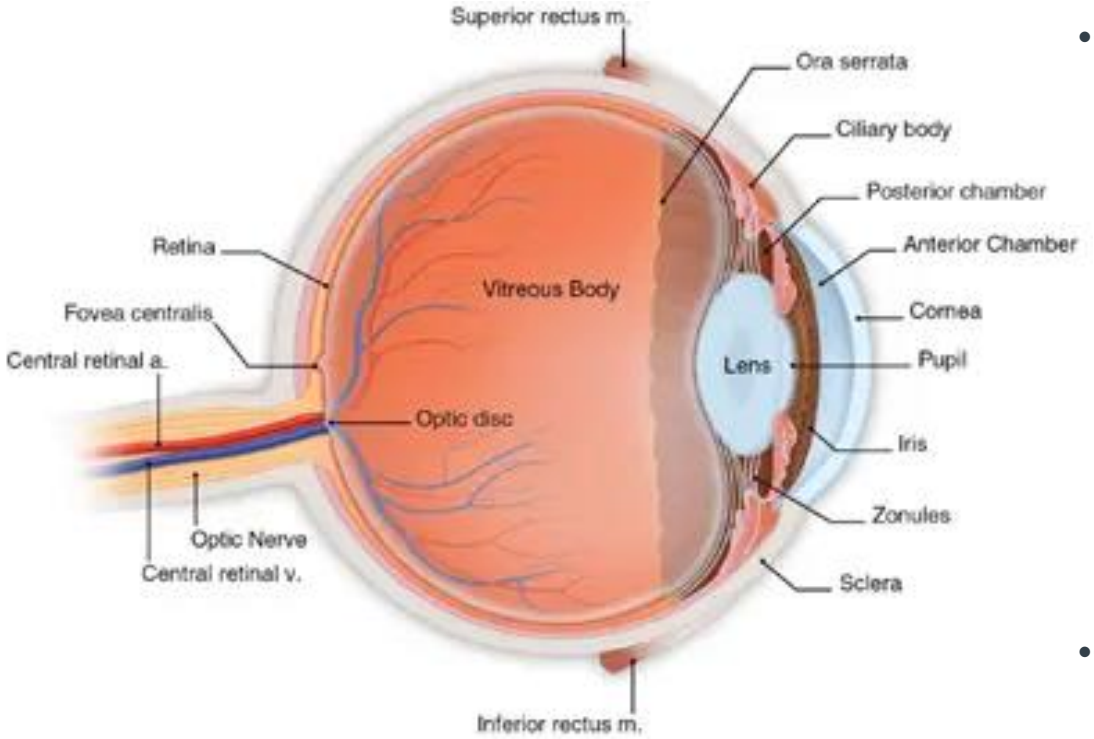
- **Sensor**: Converter of a measured physical quantity to a signal readable by an observer.
- What is measured?
  - **Proprioceptive**: Measure values internal to the robot
    - **Examples**: motor speed, wheel load, robot heading, battery status, etc.
  - **Exteroceptive**: Information from the robot's environment
    - **Examples**: distance to object, image data, sound data, gas detection, radiation, etc.
    - **Contact**: Determine shape, size, weight, etc. by touching.
    - **Non-contact**: Sense and indicate presence without physical contact.
- How is it measured?
  - **Passive**: *Energy* from the environment
    - temperature probe, microphone, RGB camera, Geiger Counter, IMU, etc.
  - **Active**: Emit energy and measure the response
    - LiDAR (light), Sonar (sound), Ultrasonics (sound), Capacitive sensors (electric field), GPS (digital!), etc.

Example: Human Visual System



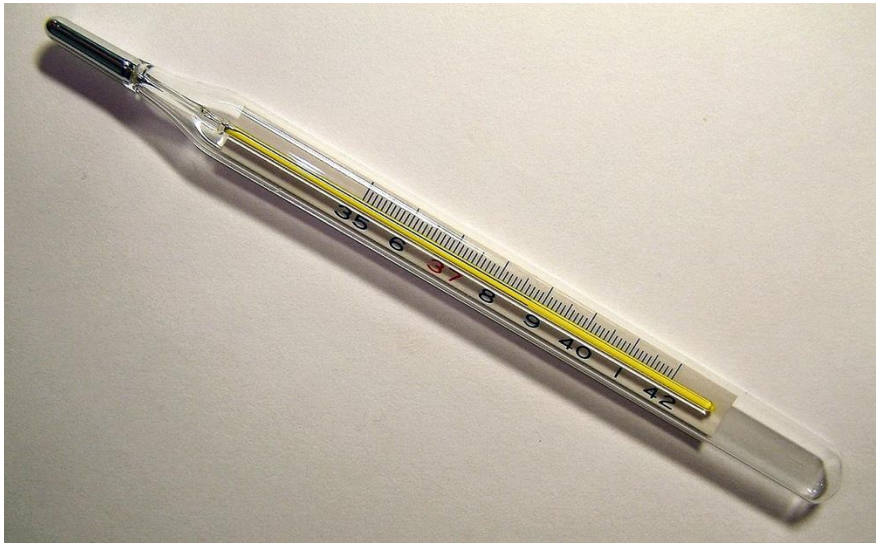


Example: Human Visual System



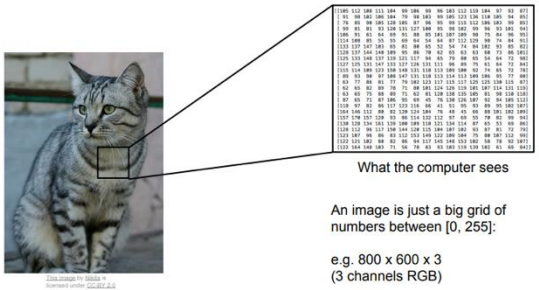
- Covered with light-sensitive receptors
  - rods
    - sensitive to a broad spectrum of light.
    - primarily for night vision & detecting motion
    - can't discriminate between colors
    - can sense intensity and shades of gray
  - cones
    - used to sense color
- Center of retina (fovea) has most cones

Another example: Thermometer

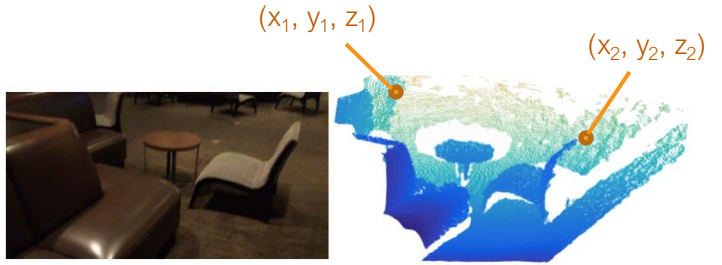


- Mercury expands or contracts with changes in temperature which we can measured with a standard scale.
- **Key point**: all sensors measure energy (from the environment or returned to the sensor) and converts it to a form that the observer (robot or human or both) can digest.

Robot Perception: **Modalities**

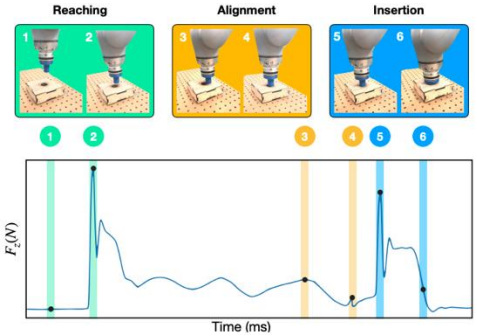


Pixels (from RGB cameras)



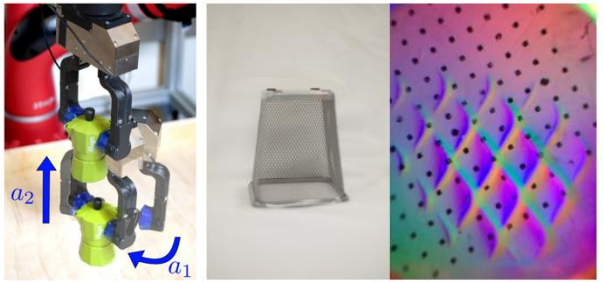
[Source: PointNet++; Qi et al. 2016]

Point cloud (from structure sensors)



[Source: Lee*, Zhu*, et al. 2018]

Time series (from F/T sensors)

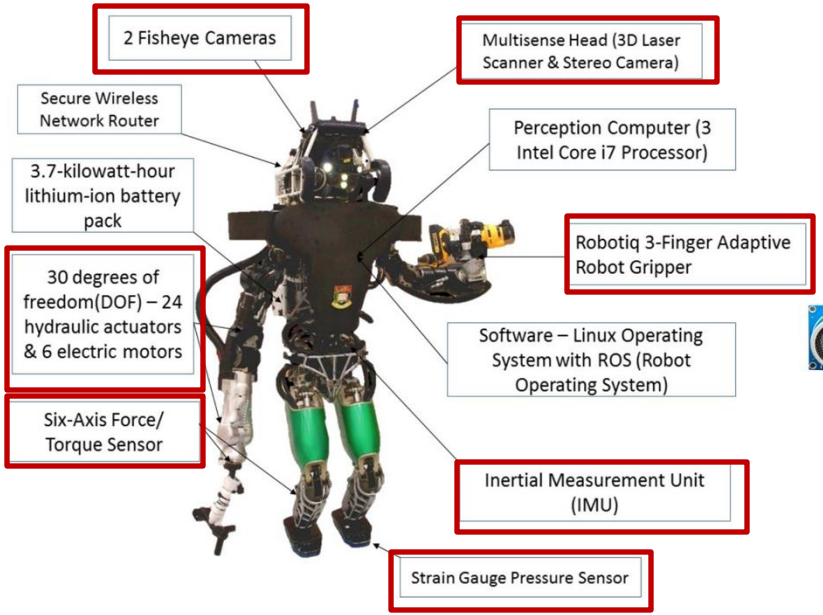


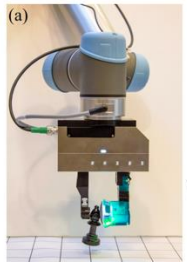
[Source: Calandra et al. 2018]

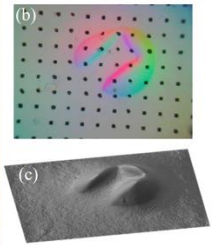
Tactile data (from the GelSights sensors)

Robotic Sensors: We often use many.

Observing the physical world through multimodal senses



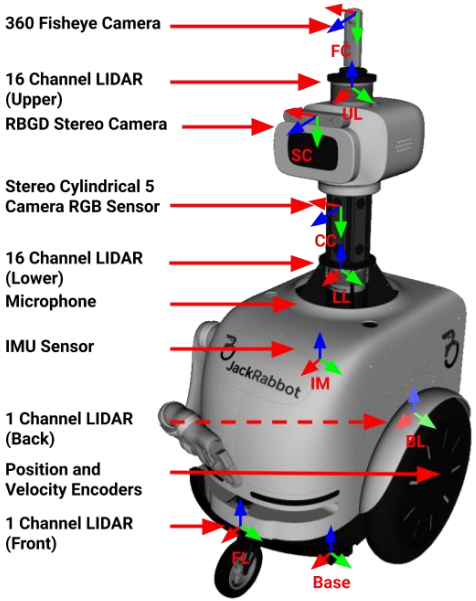




[Source: HKU Advanced Robotics Laboratory]

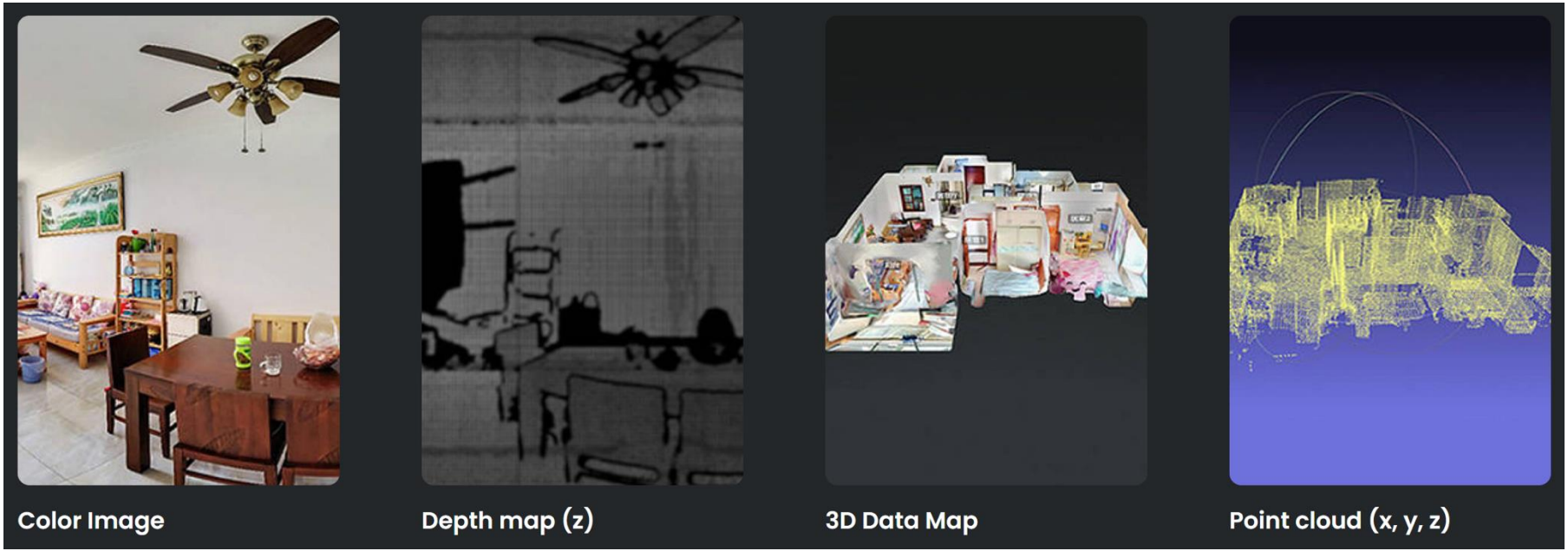
Robotic Sensors: We often use many

Observing the physical world through multimodal senses

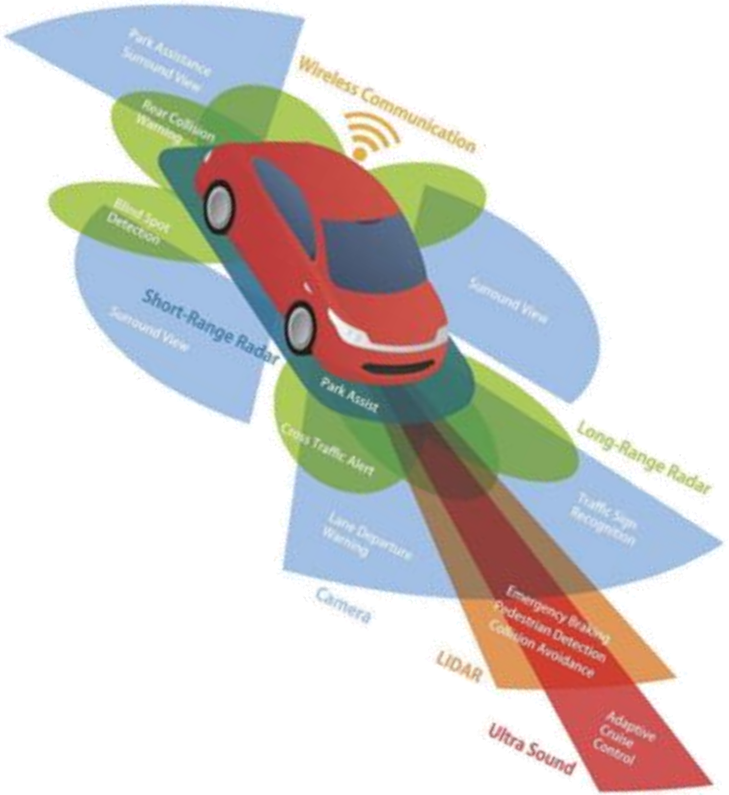


[Source: JackRabbot, Stanford]

Some More Vocabulary



Sensor Fusion



- Combining Sensor data to:
  - Get a more accurate reading
  - Create a more complete picture
  - Infer additional detail
    - Example, combine LiDAR, thermal sensors, and CO_2 sensors to identify which objects are humans.
  - Fusion Types
    - **Early (Low-Level)**: Combine raw data from multiple sensors (point clouds from LiDAR and pixels from cameras and *then* doing object recognition.
    - **Late (Mid-Level)**: The algorithm uses the different sensor data streams to decide. (Camera data and thermal data to identify humans)

Why Sensor Fusion: A Classic Example

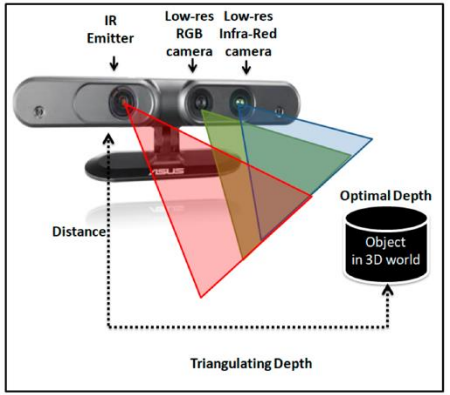
How can we learn to fuse **multiple sensory modalities** together?

Is seeing believing?

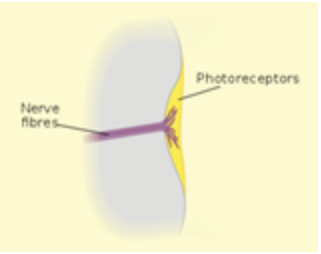
[The McGurk Effect, BBC]

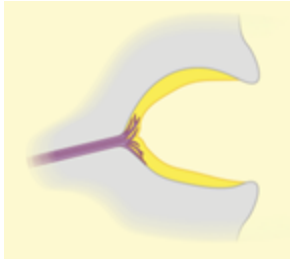
https://www.youtube.com/watch?v=2k8fHR9jKVM

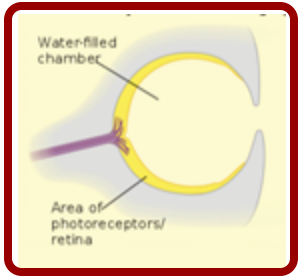
VIP: Vision (Cameras, RGBD Cameras, and LiDAR)

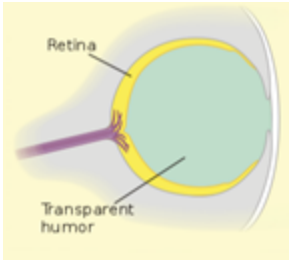


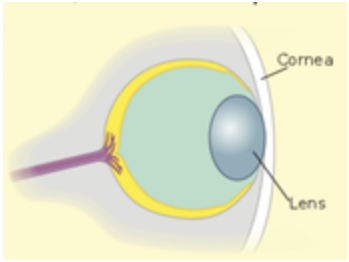
Evolution of the Eye

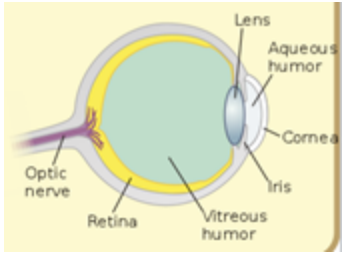






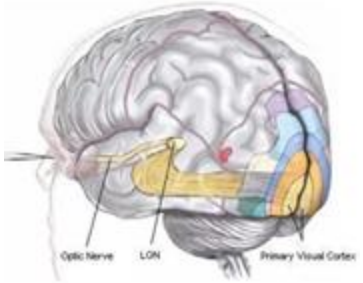




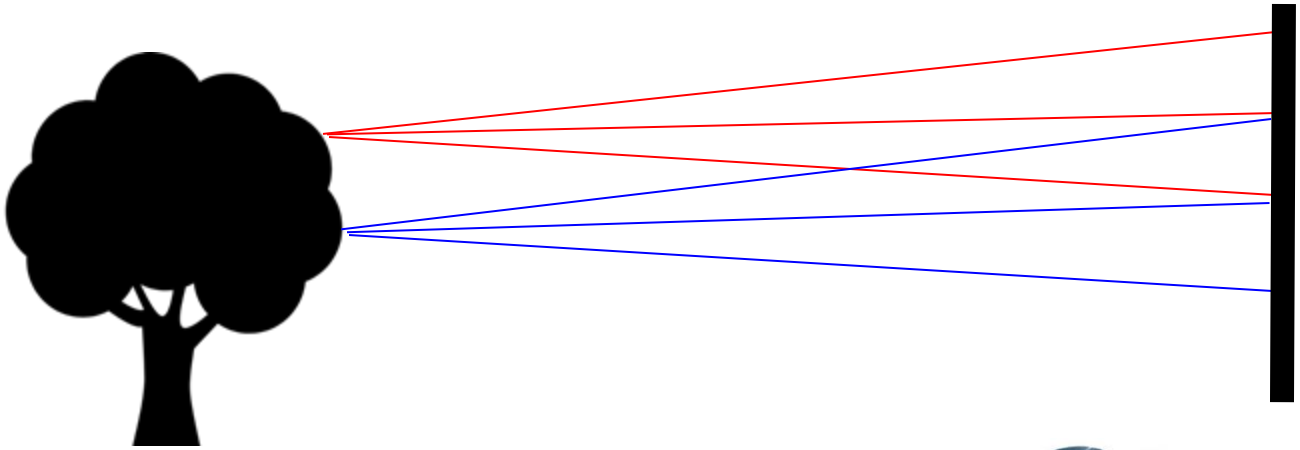


Pin Hole Model

+ More than 50% of the human cortex “involved” in vision!



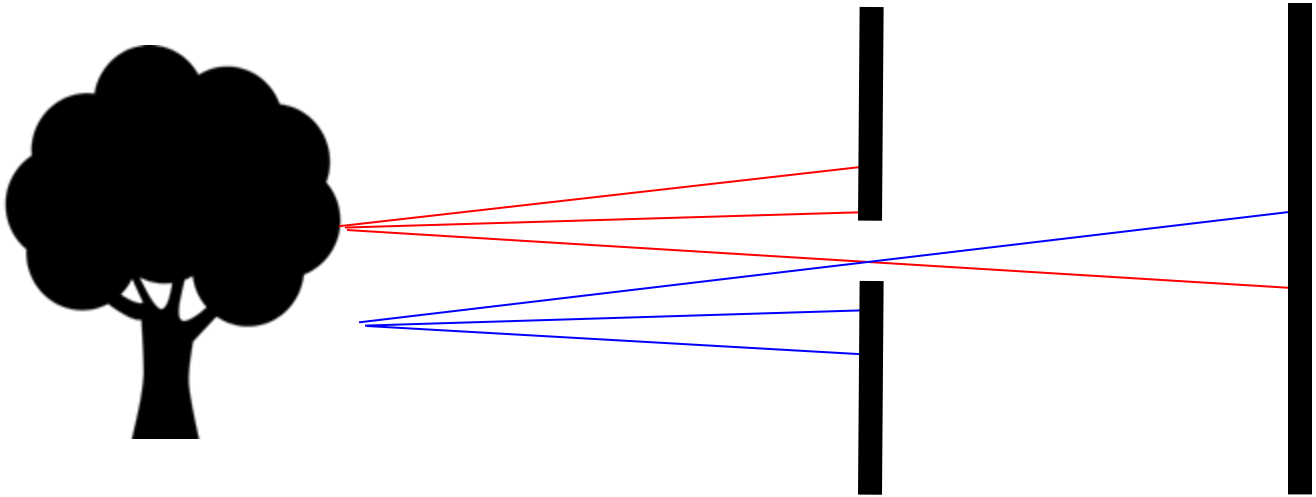
How do we see the world? Let's Design a camera.



- Put a piece of photosensitive film in front of an object.
- Do we get a reasonable image?

slides credit to Prof. Savarese

Pinhole Camera

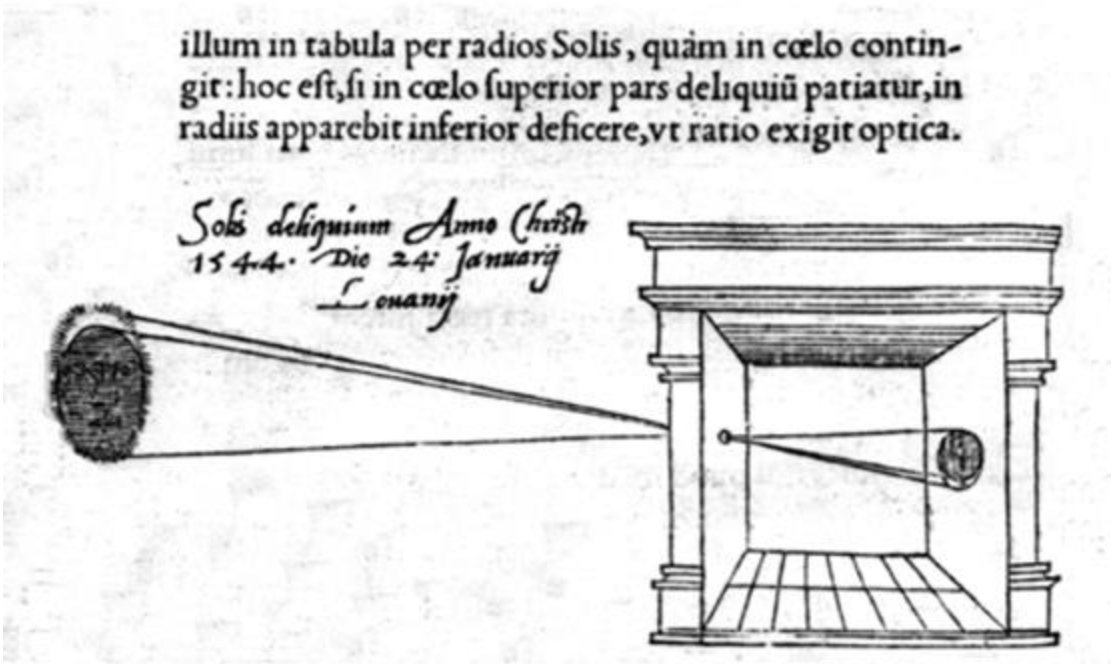


- Add a barrier to block most of the rays
  - Reduces blurring
  - Opening in barrier is known as the aperture.

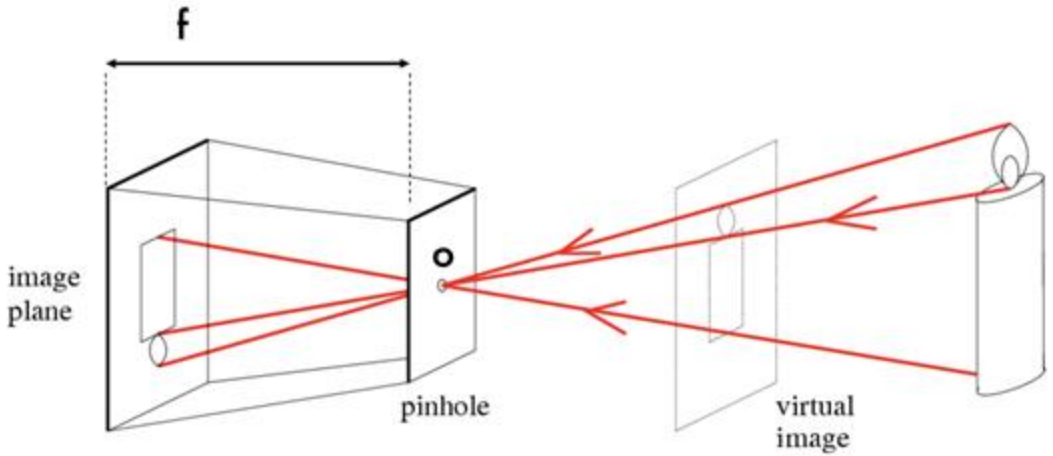
slides credit to Prof. Savarese

First one to do it (that we know about…)

Leonardo da Vinci
(1452-1519)



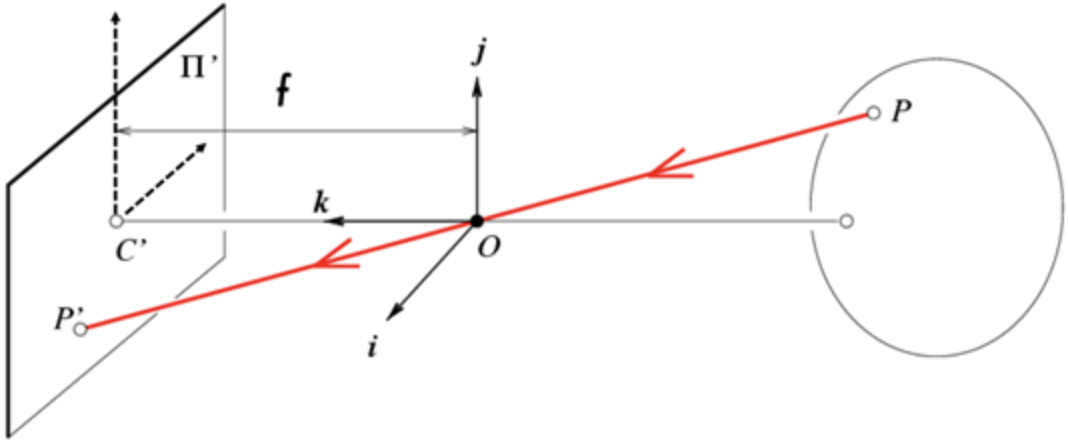
Pinhole Camera Model

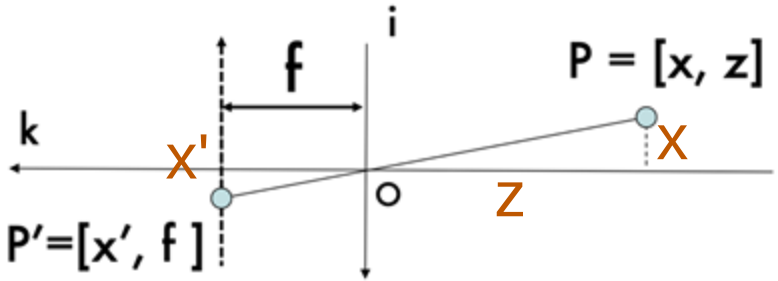


- Capture all rays passing through a single ‘point’
  - known as the Center of Projection or focal point
  - Forms 2D image know as the image plane.

slides credit to Prof. Savarese

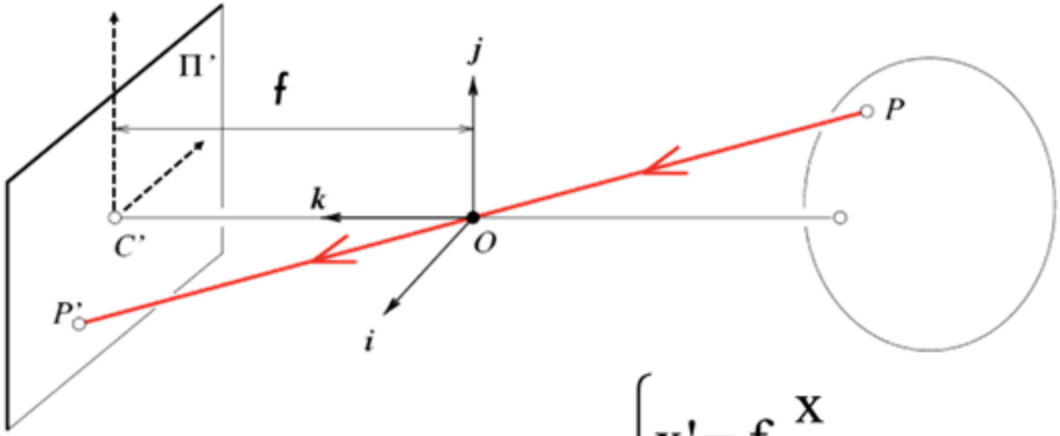
Pinhole Camera Model





$$\frac{x'}{f} = \frac{x}{z}$$

Pinhole Camera Model



$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

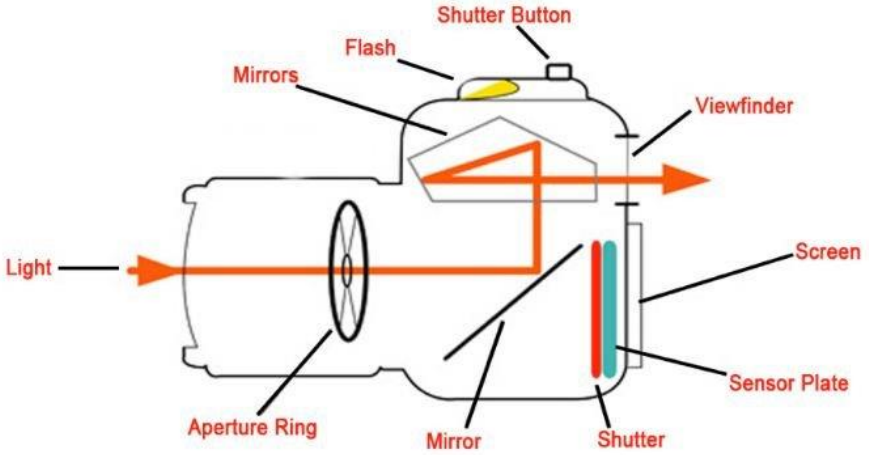
$$\begin{cases} x' = f\,\dfrac{x}{z} \\ y' = f\,\dfrac{y}{z} \end{cases}$$

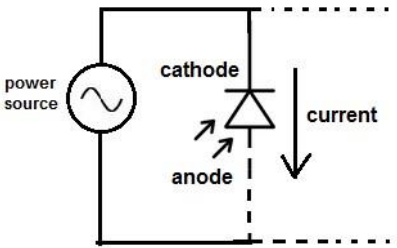
Derived using similar triangles

- 3D → 2D

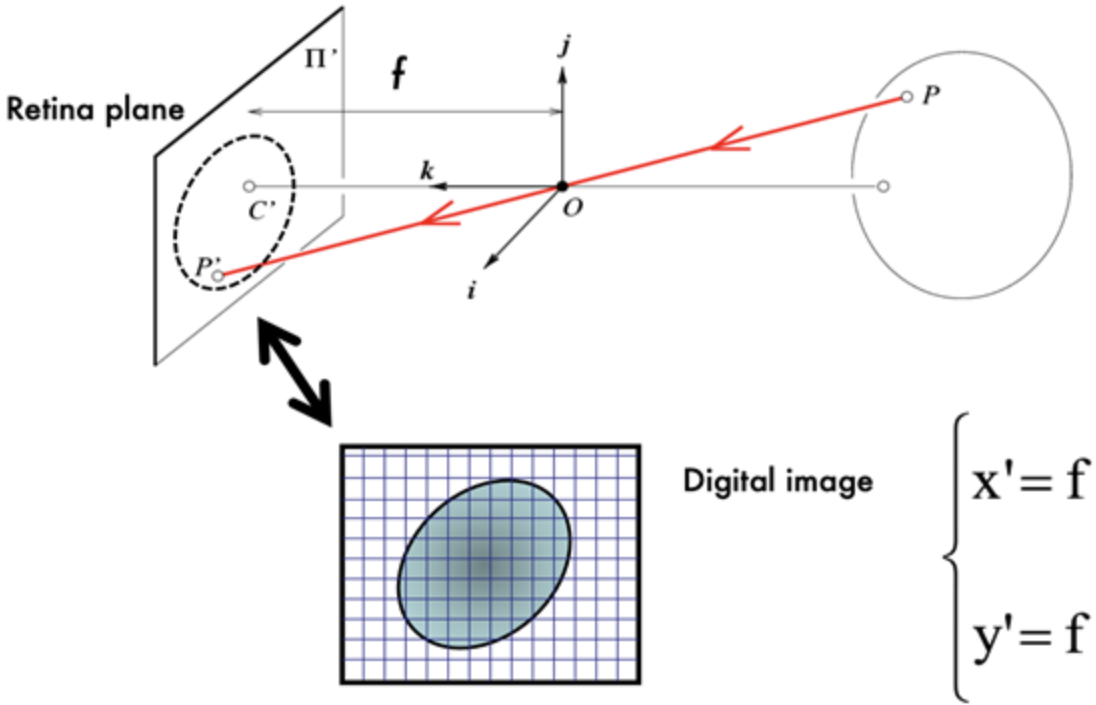
Digital Camera

- Photosensitive sensitive sensor (array of individual photosensitive diodes) gets exposed to light rays
- Digital measurements are collected from each sensor
- We obtain a 2D array with the measurements
- Different sensors for different colors





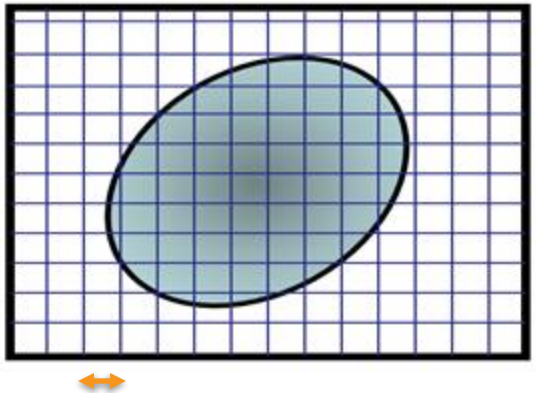
Digital Image – From "metric" to "pixel space"



$$\begin{cases} x' = f\,\dfrac{x}{z} \\ y' = f\,\dfrac{y}{z} \end{cases}$$

- If f is in m (or mm), x' and y' will be in m (or mm)
- But we want to know the pixel "number"!

Digital Image – From "metric" to "pixel space"



height of a pixel [m/pixel]

$$\begin{cases} x' = f\dfrac{x}{z} \text{ [m]} \\ y' = f\dfrac{y}{z} \text{ [m]} \end{cases}$$

$$x'[pixels] = \frac{x[m]}{pixel_{width}\left[\frac{m}{pixel}\right]}$$

$$y'[pixels] = \frac{y[m]}{pixel_{height}\left[\frac{m}{pixel}\right]}$$

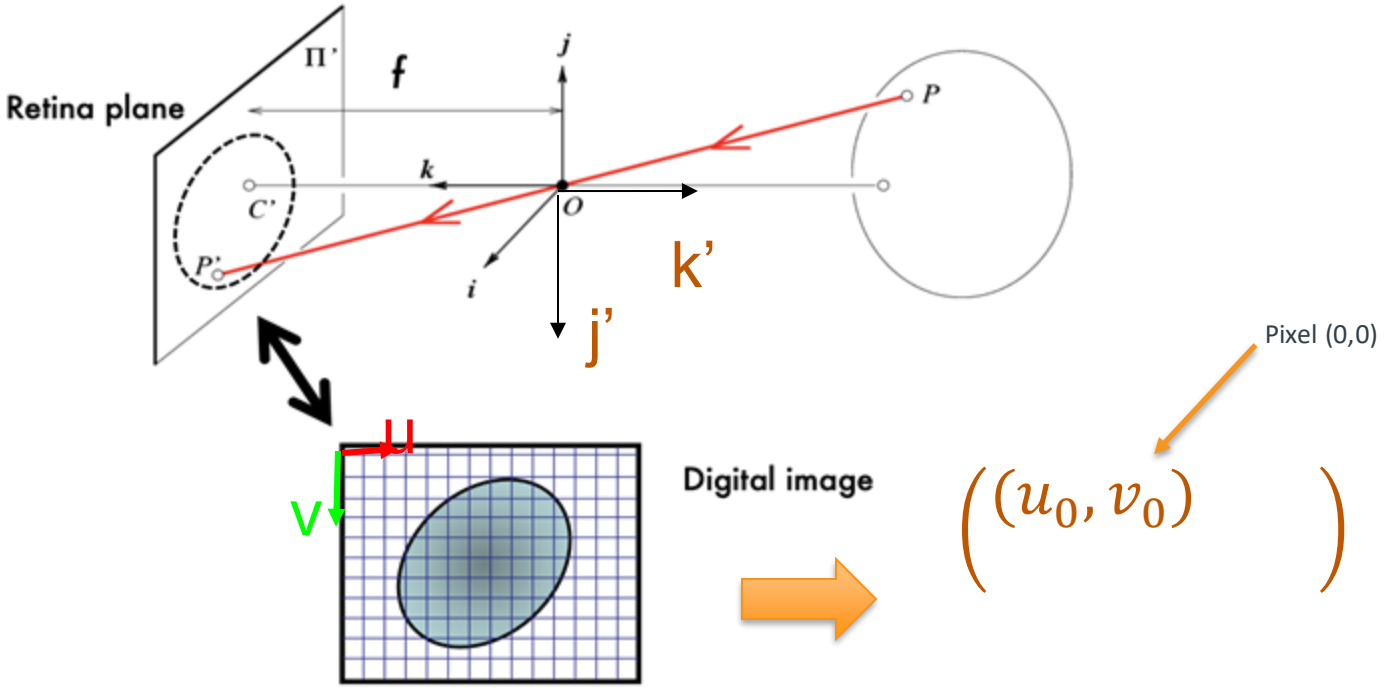
width of a pixel [m/pixel]

- In practice, we use fx and fy in pixels, that combine both operations
- $f_x = \frac{f}{pixel_{width}} \rightarrow x' = f_x \frac{x}{z}$
- $f_y = \frac{f}{pixel_{height}} \rightarrow y' = f_y \frac{y}{z}$

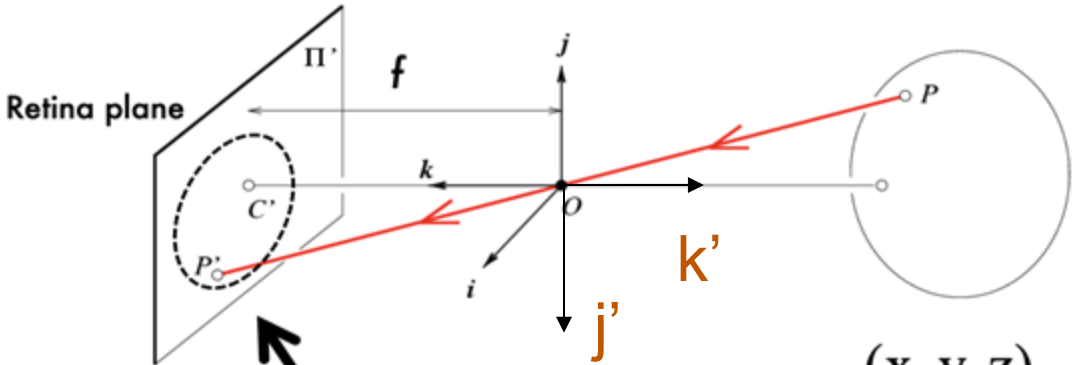
In pixels!

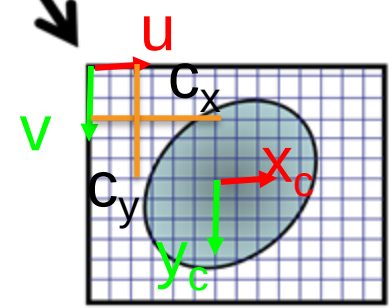
We round to the smallest integer → floor operation:
floor(x) = greatest integer smaller or equal than x)

Digital Image – referring to top-left corner



Offset to Image Center





Digital image

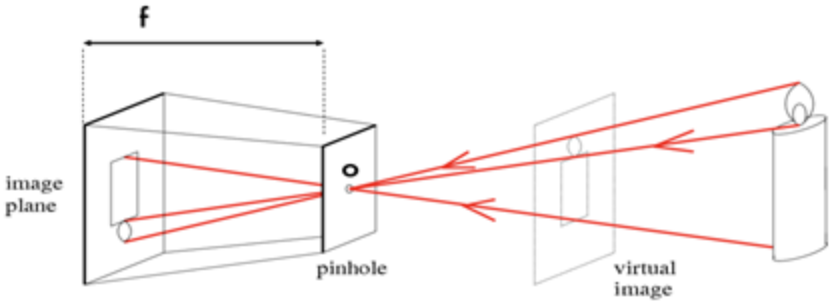
Both in pixels or in metric

$$(x, y, z) \rightarrow (f\frac{x}{z} + c_x, f\frac{y}{z} + c_y)$$

Projective Transformation

Homogeneous Coordinates

- We previously used these to transform a point with a pose
  - Now we will use them to map 3D points to 2D (image space)



E→H

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous scene coordinates

Projective Transformation with Homogeneous Coordinates

$$P_h' = \begin{bmatrix} f_x x + c_x z \\ f_y y + c_y z \\ z \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad P_h$$

Also called P

$$K = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Sometimes we omit it

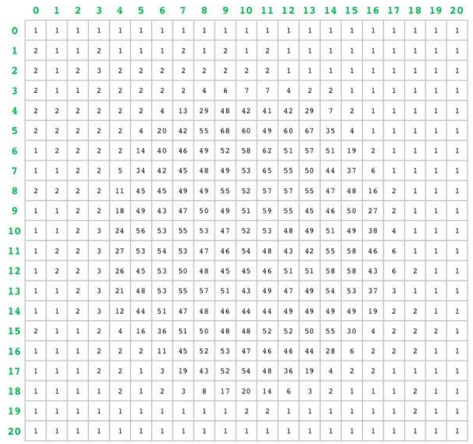
Homogenous → Euclidian

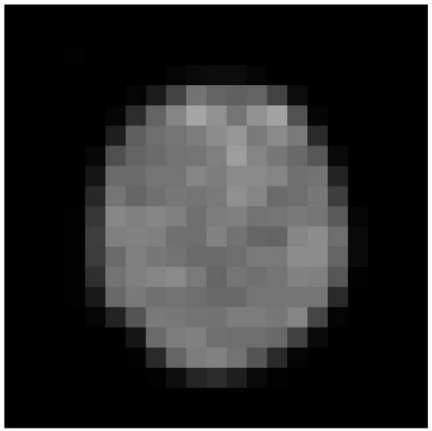
$$P_h' \rightarrow P' = (f_x \frac{x}{z} + c_x, f_y \frac{y}{z} + c_y)$$

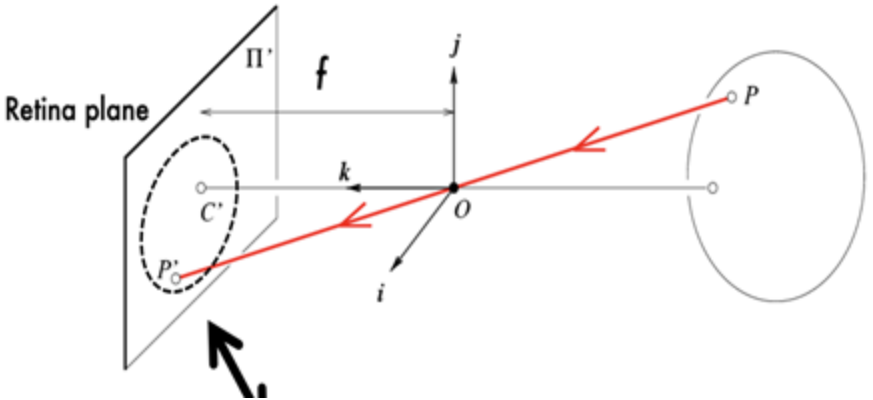
Intrinsic matrix

Digital Image

- f in pixels
- f_x and f_y are often the same (not exactly, depends on calibration)
- c_x and c_y are the image width/2 and height/2 (not exactly, depends on calibration)
- In practice, how do we go from f in mm to f in pixels?
  - Knowing the physical size of the photosensitive sensor and the resolution of the image (cam specs)







3mm and 480 pixels

Example camera matrix in real camera (ROS)

```
$ sudo nano ./ros/camera_info/head_camera.yaml

OUTPUT-

image_width: 640
image_height: 480
camera_name: head_camera
camera_matrix:
  rows: 3
  cols: 3
  data: [729.1016874494248, 0, 277.5382540847794, 0,
728.3444988325477, 218.6703499234103, 0, 0, 1]
```

https://pollev.com/robertomartinmartin739

Exercise

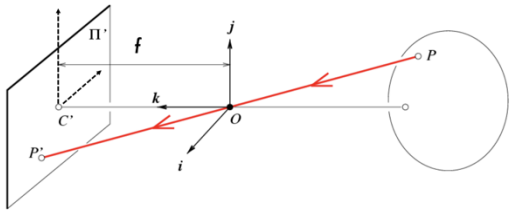
Given the internal camera matrix (in mm)

$$K = \begin{pmatrix} 1.5 & 0 & 0.5 \\ 0 & 1.5 & 0.5 \\ 0 & 0 & 1 \end{pmatrix}$$

and the 3D point (in m)

$$p = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}$$

estimate the projection of the point (pixel coordinates) of the point in the image



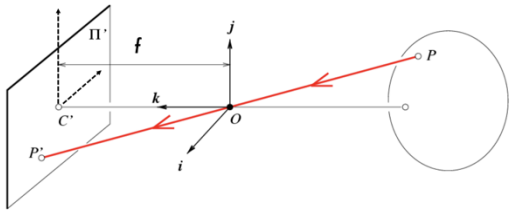
https://pollev.com/robertomartinmartin739

Exercise

Given the internal camera matrix (in pixels) $K = \begin{pmatrix} 500 & 0 & 320 \\ 0 & 500 & 240 \\ 0 & 0 & 1 \end{pmatrix}$

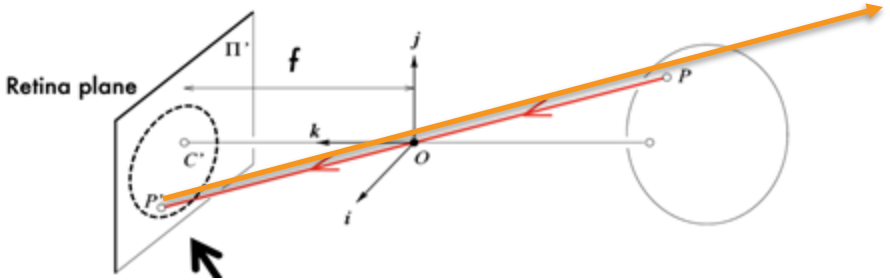
and the 3D points (in m) $p_1 = \begin{pmatrix} 1 \\ -0.5 \\ 3 \end{pmatrix}$ and $p_2 = \begin{pmatrix} -10 \\ 2 \\ 2 \end{pmatrix}$

estimate the projection of the points (pixel coordinates) in the image



Can we recover the 3D location of a point from an image?

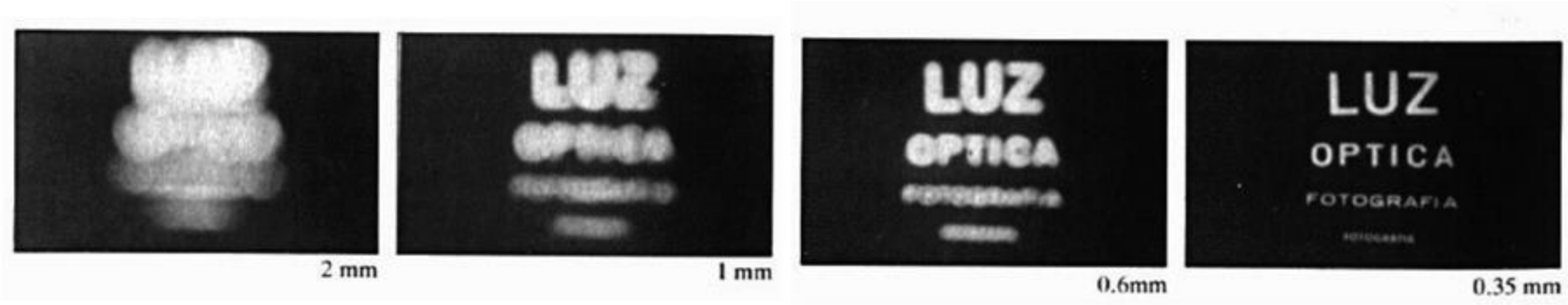
- NO!
- If we know the camera intrinsics, we can recover the "ray" on which the point will be, **but** not the depth
- Projective geometry "removes" one dimension: we go from 3D to 2D, but we can't go from 2D to 3D



So, I made my pinhole camera!

- Why is it so blurry?

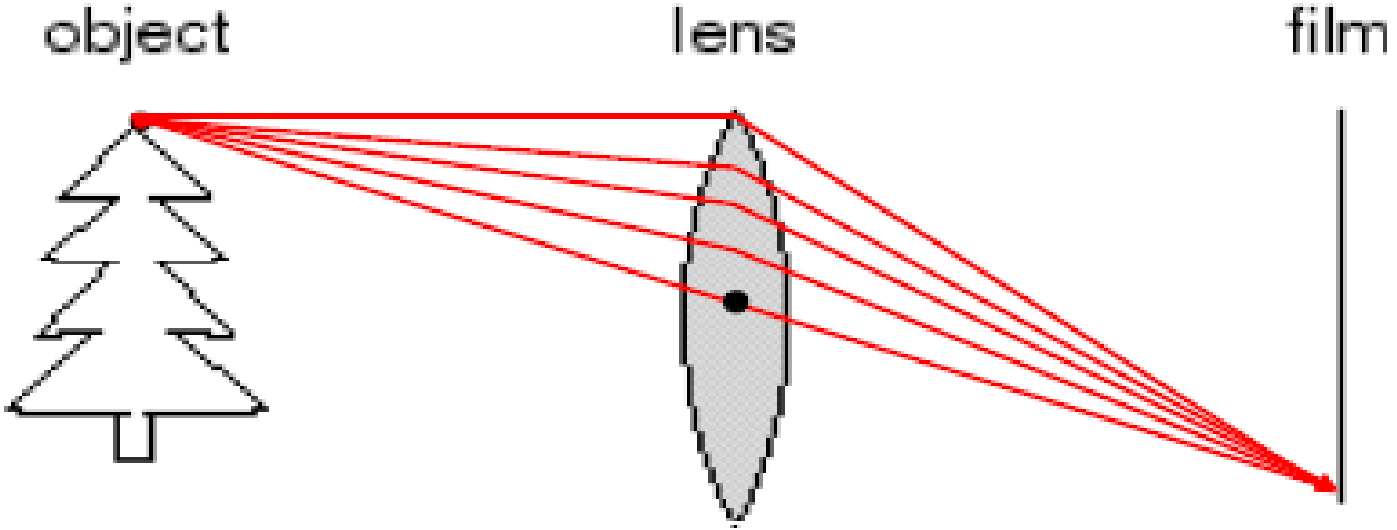
Size of the Aperture



- So let's make the aperture as small as possible…
  - But…

- Diffraction becomes an issue
- Or lack of enough light

Camera Lenses



- A lens focuses light on the film
  - Larger aperture without blurriness

slides credit to Prof. Savarese

New Problems: Lens Distortion

- Images are distorted due to lens, film, and their locations.
  - lens imperfections, finite aperture, alignment errors, etc.
  - Most pronounced in wide angle lens →
- Radial Distortion
  - distortion in objects due to distance from the principal point.
- Tangential Distortion
  - Optical axis is not perfectly aligned with the sensor plane
- Center of perspective projection
  - Center of distortion is not aligned with center of perspective.

Modeling Distortion: Plumb Bob Model

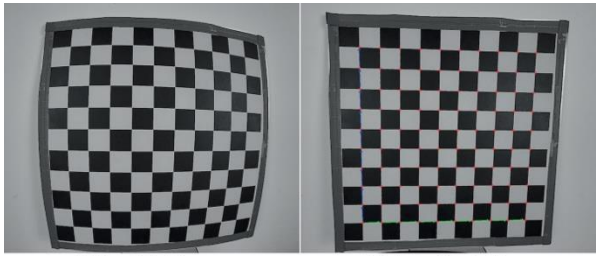
$$p_c = \begin{pmatrix} x_c \\ y_c \end{pmatrix} = \begin{pmatrix} f_x \frac{x}{z} \\ f_y \frac{y}{z} \end{pmatrix}$$

$$p'_c = p_c \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + \begin{pmatrix} 2t_1 x_c y_c + t_2 (r^2 + 2x_c^2) \\ t_1 (r^2 + 2y_c^2) + 2k_2 x_c y_c \end{pmatrix}$$

Radial distance $r^2 = x_c^2 + y_c^2$

Distortion parameters $d = (k_1, k_2, k_3, t_1, t_2)$

We can “undistort” an image!!!



(a) An original image (b) An undistorted image

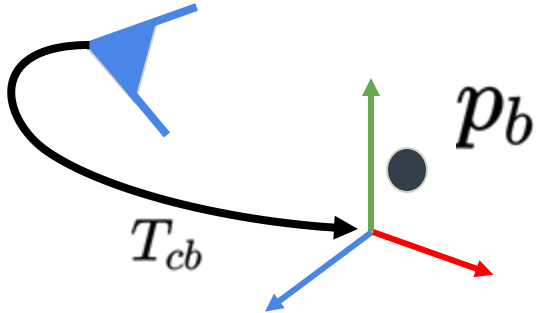
Example distortion in real camera (ROS)

```
$ sudo nano ./ros/camera_info/head_camera.yaml

OUTPUT-

image_width: 640
image_height: 480
camera_name: head_camera
camera_matrix:
  rows: 3
  cols: 3
  data: [729.1016874494248, 0, 277.5382540847794, 0,
728.3444988325477, 218.6703499234103, 0, 0, 1]
distortion_model: plumb_bob
distortion_coefficients:
  rows: 1
  cols: 5
  data: [-0.00898402793551297, -0.04505947514194832,
-0.006922247877124037, -0.01114510192485125, 0]
```

Internal and External Camera Parameters

Internal Camera Parameters: (scaling, inversion, distortion correction, etc.) K

External Camera Parameters: T_{cb}



$$p_c = K \cdot T_{cb} \cdot p_s$$

4x4 → We add the last column of zeros to K for the dimensions to match

https://pollev.com/robertomartinmartin739

Exercise

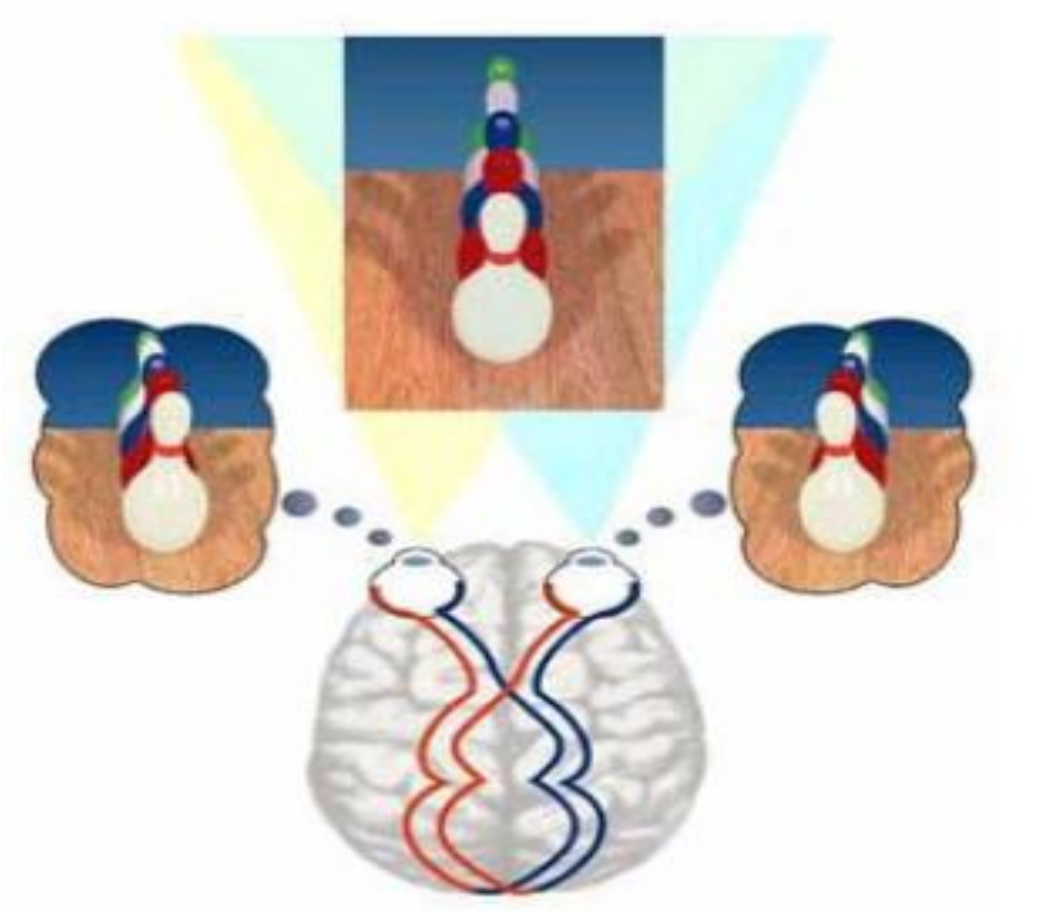
Given the internal camera matrix $K = \begin{pmatrix} 5 & 0 & 10 & 0 \\ 0 & 4 & 10 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

and the 3D point $p_b = (2, 2, 2)$

on an object placed at $T_{cb} = \begin{pmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

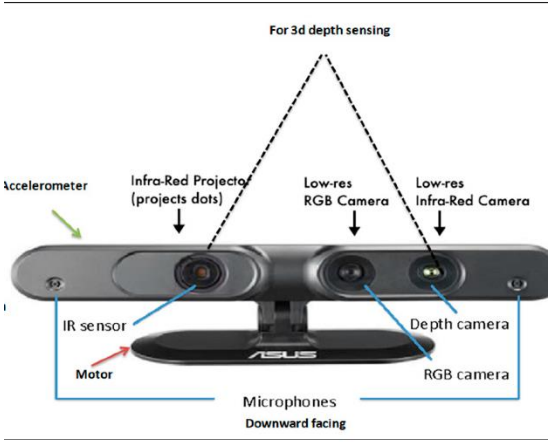
Wrt. the camera, estimate the projection of the point into the camera.

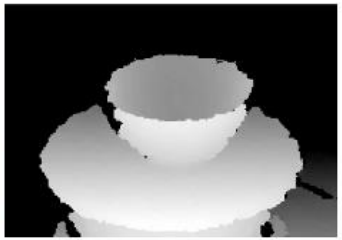
But what about depth?



RGB-D Cameras

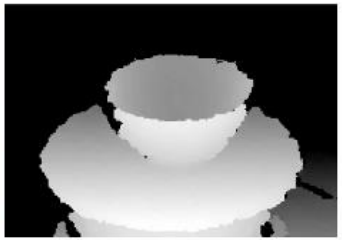
- Can we recover the 3D position of a point?
  - YES!
  - We have the distance between the sensor and the point, we know where the point is along the ray





RGB-D Cameras

- Given a pixel (u,v), what is its 3D location?

$$p' = (u, v) = (f_x \frac{x}{z} + c_x, f_y \frac{y}{z} + c_y)$$

$$x = \frac{(u - c_x)z}{f_x} \qquad y = \frac{(v - c_y)z}{f_y}$$

We get z from the depth map!

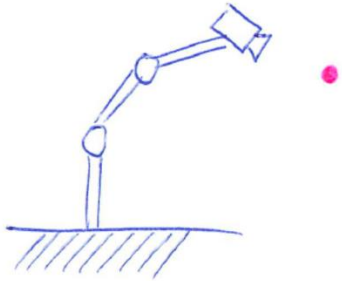

31 12 2010

Estimating Camera Parameters, **K**: Camera Calibration

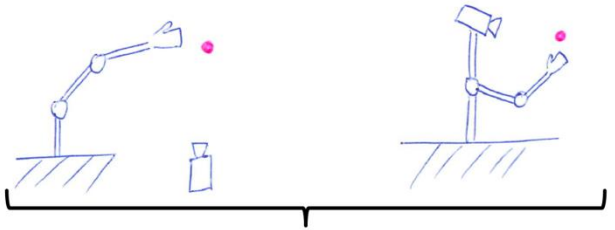
- Move known pattern (size) in front of the camera and collect images
- Detect point-corners on the pattern
  - Set of images -> set of corresponding points
- Estimate:
  - Camera-to-pattern poses T_{cb}^{i}
  - Camera parameters that minimize the reprojection error K, d

Controlling a robot directly with images

- Control the robot motion based directly on vision
- Types of systems depending on where the camera is placed



eye-in-hand system



eye-to-hand system

endpoint closed-loop (ECL)
or
endpoint open-loop (EOL)

Summary

- Way too many sensors to cover in two lectures
  - Active vs Passive
  - Proprioceptive vs Exteroceptive
- Vision is arguably the most important
- Image formation uses a pinhole camera model
  - Gives us the coordinates (pixels) of a point in the image
- In general, we cannot know the 3D point that correspond to a given pixel (many points along a ray would fall in the same pixel)
  - Except if we have a depth sensor! → we know exactly which point on the line
- Camera calibration is the process of finding the internal parameters of the camera
  - Focal length
  - Image center
  - Distortion params
- Robots uses camera-to-hand or camera-in-hand settings