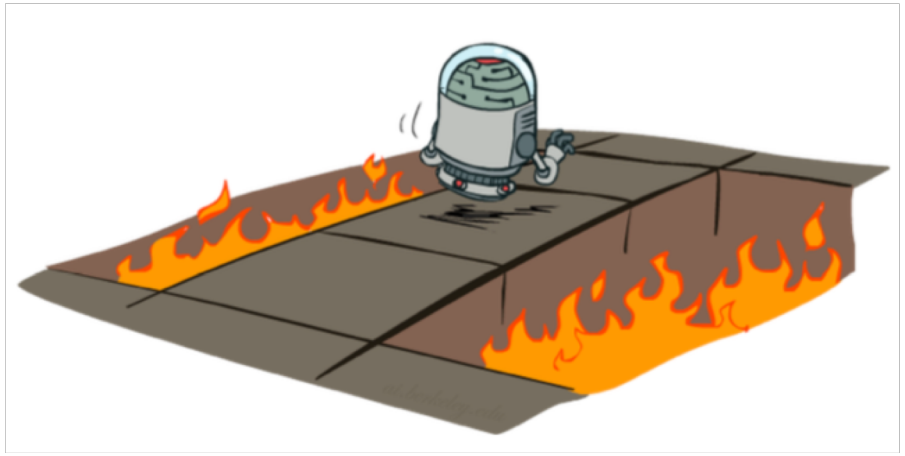
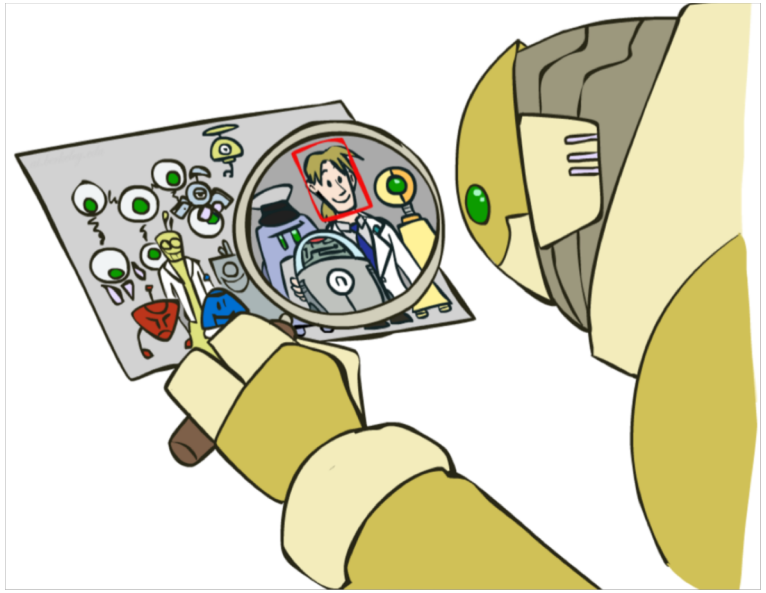
CS 1501: Intro to Robotics

Autonomy, AI, and Applications



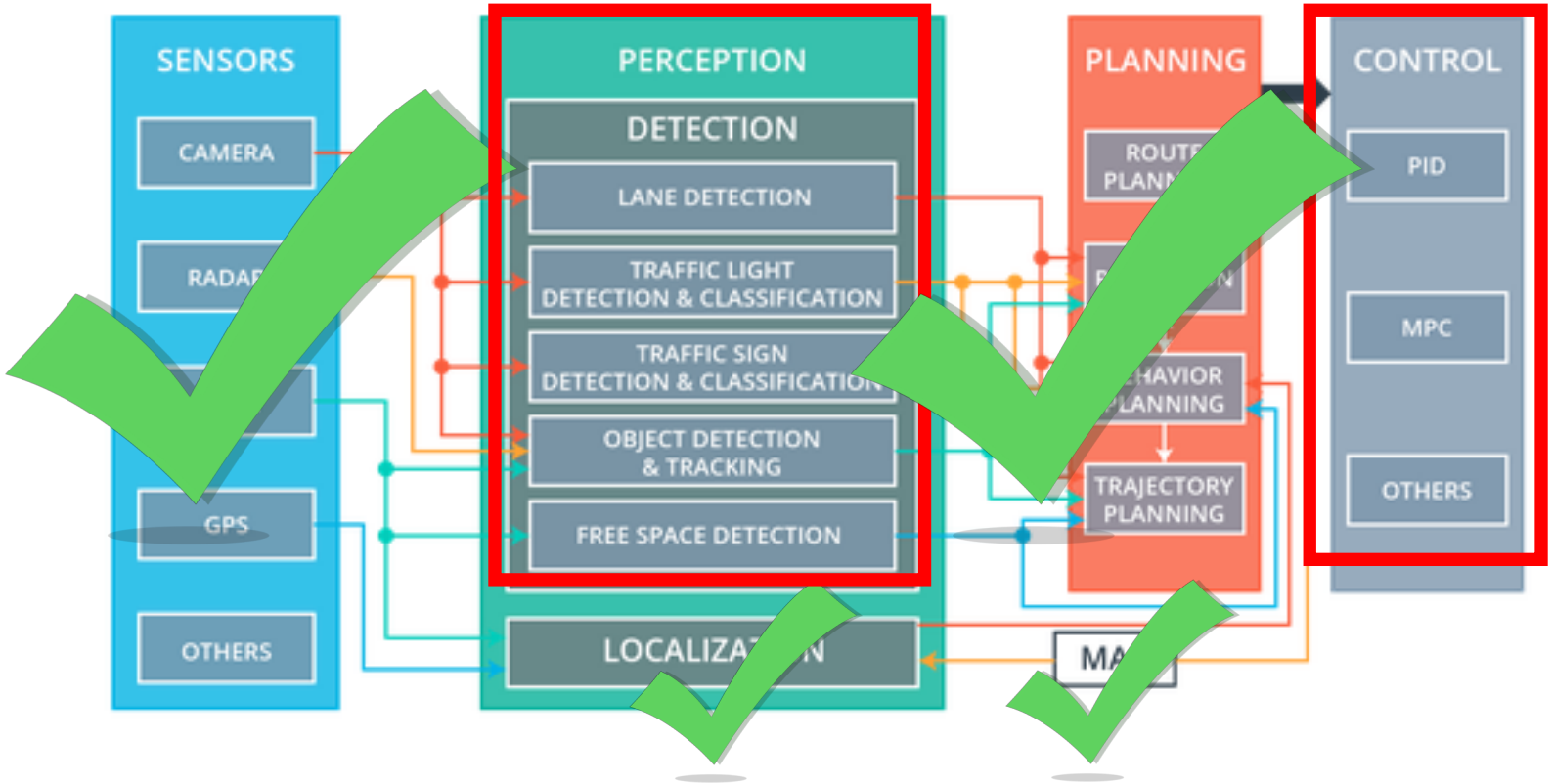
Motion Control II, Perception



Rohan Raval

Monday 1-1:50pm, MEC 213

Roadmap: See-Think-Act



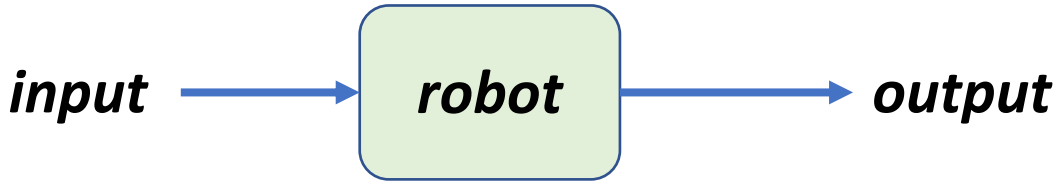
Recap: Control Theory

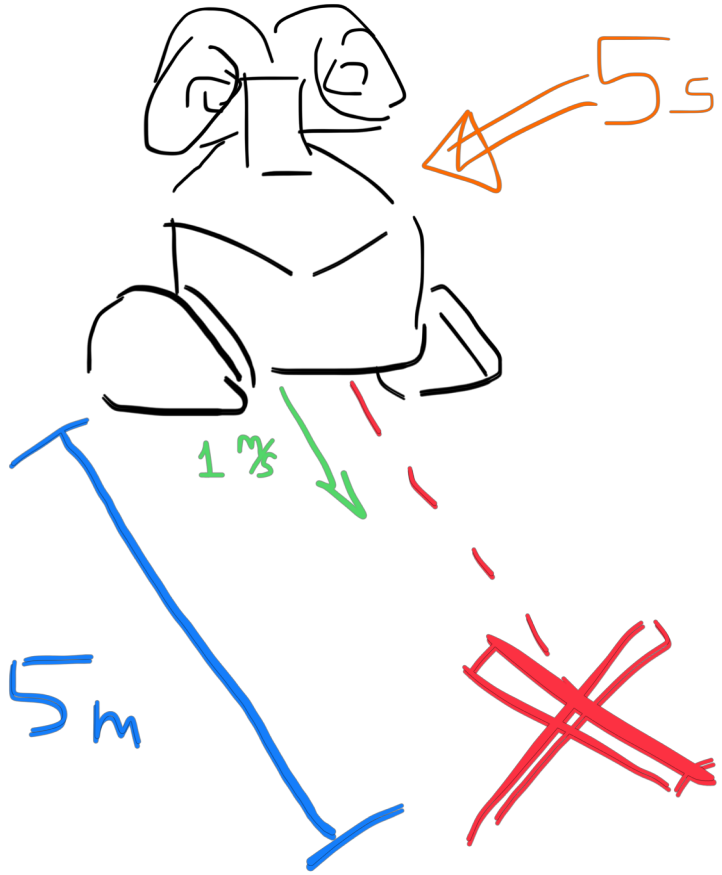
Task: We want our Robot to go to the Goal (red X)

Open-Loop Control

- Calculate distance and direction from Robot to Goal (5m forward)
- Given Robot's max speed (1 m/s)...
- Tell Robot to go **5** seconds in **forward** direction to reach goal

- Amount of time robot drives is not "adjusted"/corrected based on actual position of robot

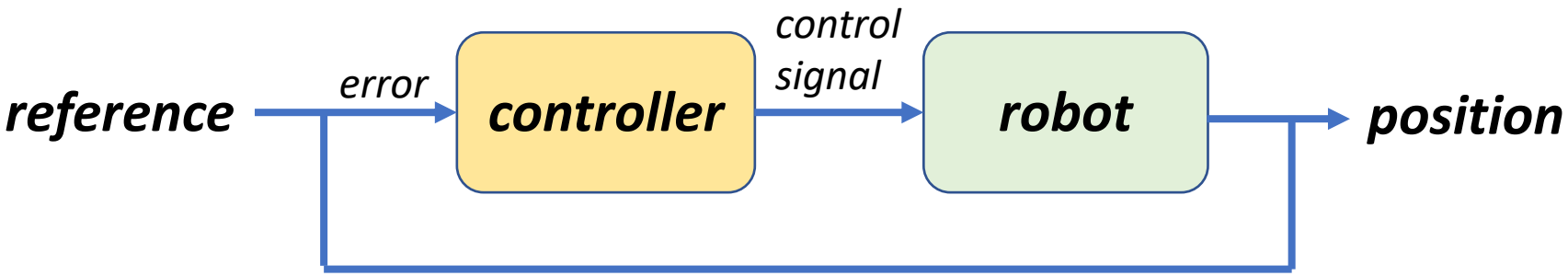


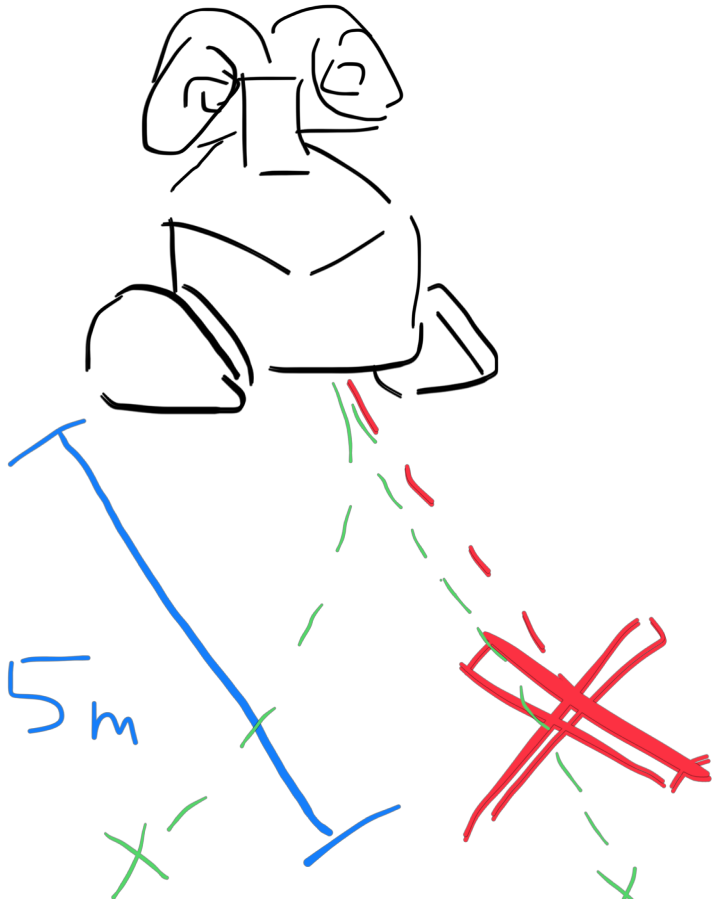


Recap: Control Theory

Closed-Loop (Feedback) Control

- What if the robot doesn't reach the goal?
  - Environmental factors (drift, ground, etc.)
  - Systematic biases (Steering wheels misaligned)
- Need to convey to robot that we are "off-track", and adjust accordingly
- Compare reference trajectory to current position = error
- Now just try to minimize that error!
- **Controller = converts error into a command (control signal *u*) to be sent to robot**





Recap: PID Controller

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau)d\tau + k_D \frac{de(t)}{dt}$$

Video: [Controlling Self Driving Cars]

Recap: PID Controller

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau)d\tau + k_D \frac{de(t)}{dt}$$

What does Proportional (k_P) do?

- *"Present"*: Contributes to stability, medium-rate responsiveness

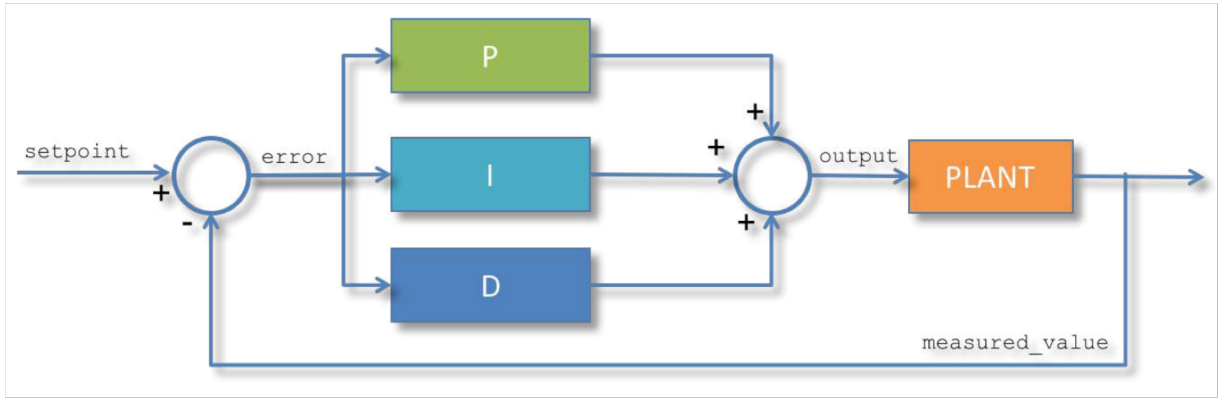
What does Integral (k_I) do?

- *"Past"*: Accumulates steady-state (past) errors
- Tracking and disturbance/bias rejection, slow-rate responsiveness

What does Derivative (k_D) do?

- *"Future"*: Anticipates errors based on rate of change
- Corrects Fast-rate responsiveness, sensitive to noise

Recap: PID Controller



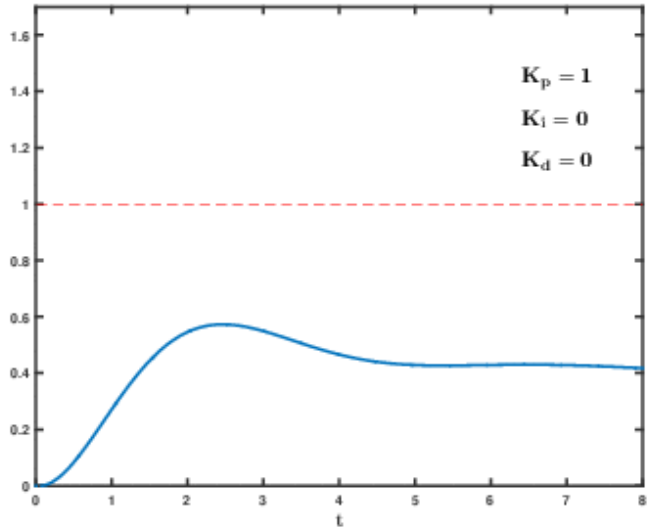
Pseudo-code:

```
previous_error = 0
integral = 0
start:
  error = setpoint - measured_value
  integral = integral + error*dt
  derivative = (error - previous_error)/dt
  output = Kp*error + Ki*integral + Kd*derivative
  previous_error = error
  wait(dt)
  goto start
```

Recap: Tuning a PID

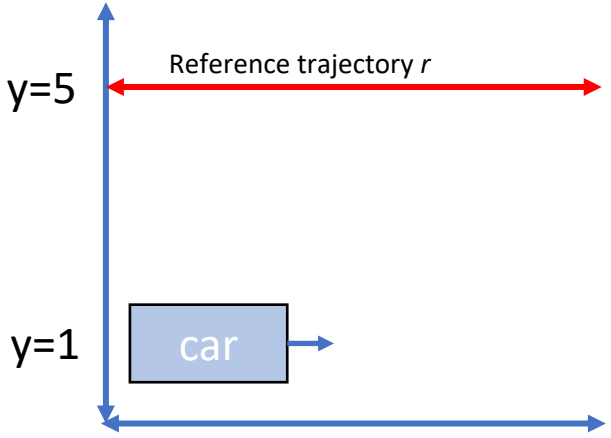
$$u(t) = k_P e(t) + k_I \int_0^t e(\tau)d\tau + k_D \frac{de(t)}{dt}$$

- **Steps to tune your k_P, k_I and k_D paramters:**
  - Start with k_P and increase until oscillations appear
  - Then move to k_I to reduce error
  - Finally move k_D to reduce oscillations
  - Go back to k_P and change until almost no oscillations (just a little overshoot)

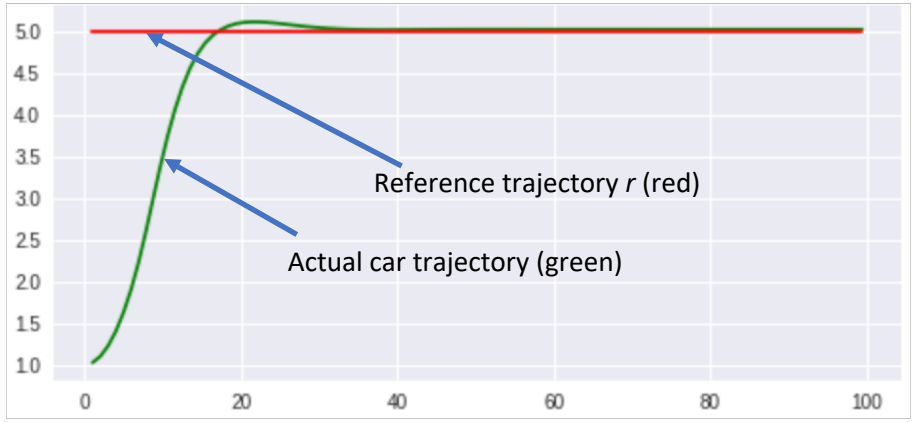


PID Controller: Exercise

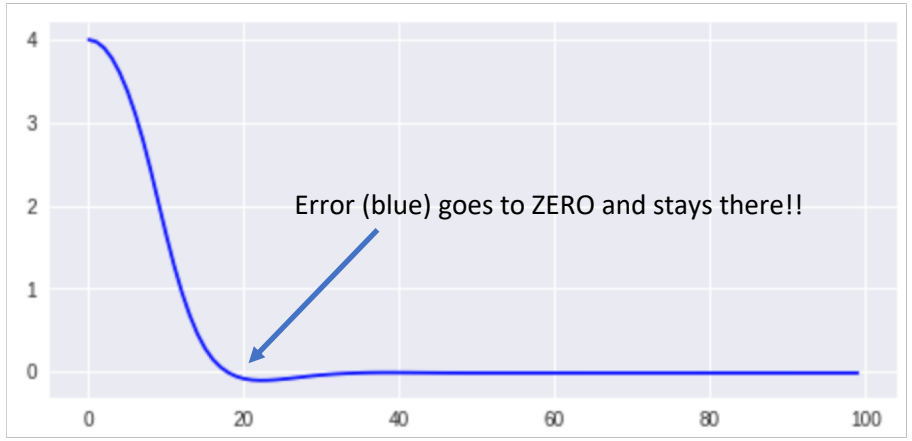
- **Task:**
  - Car starts at y=1
  - We want to follow a trajectory of y=5 (for all x)
  - So we need to steer car from y=1 to y=5, using a PID controller, & keep it there
  - Controller is on the *steering angle* (not velocity, like the previous slides!)
  - Fill in the code here: [bit.ly/cs1501-pid](bit.ly/cs1501-pid)



Depiction of task

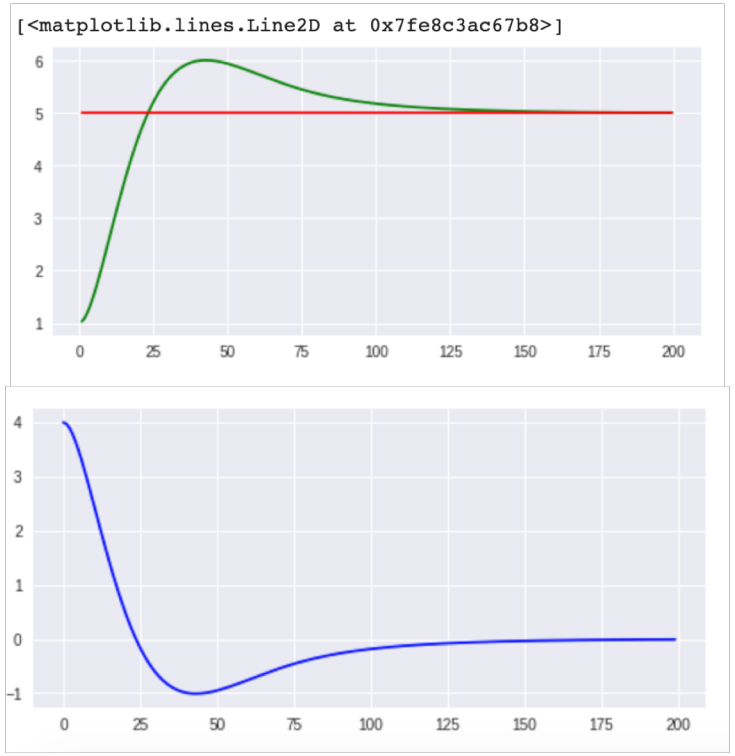


What we want to do



*What we want the **error** graph to look like*

PID Controller: Exercise Solution




```
k_p = 0.2
k_i = 0.004
k_d = 3.0
```

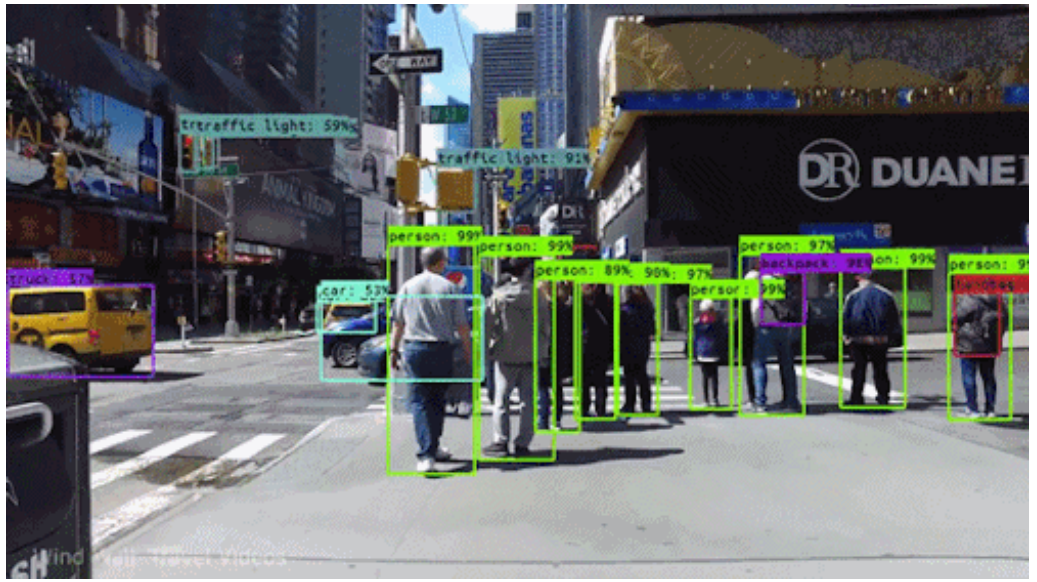
```python
def run(robot, k_p, k_i, k_d, n=100, speed=1.0, dt=1.0):

    x_trajectory = []
    y_trajectory = []
    errors = []

    prev_e = 0
    integral_e = 0

    for i in range(n):

        # error
        e = ref - robot.y

        # integral term
        integral_e += e * dt

        # derivative term
        diff_e = (e - prev_e) / dt
        prev_e = e

        # control input u (PID)
        steer_angle = (k_p * e) + (k_d * diff_e) + (k_i * integral_e)

        robot.move(steer_angle, speed)

        x_trajectory.append(robot.x)
        y_trajectory.append(robot.y)
        errors.append(e)

    return x_trajectory, y_trajectory, errors
```

Perception

Perception = How can a robot “see”?

What does “see”-ing mean for a robot?

- Camera Images (stereo)
- LiDAR Point Clouds



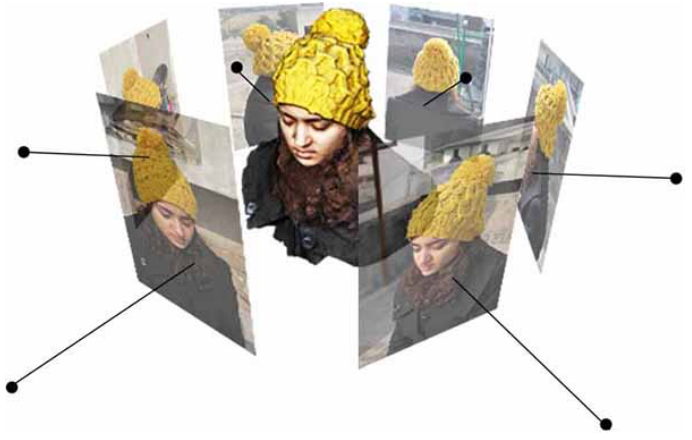
Why would a robot want to “see”?

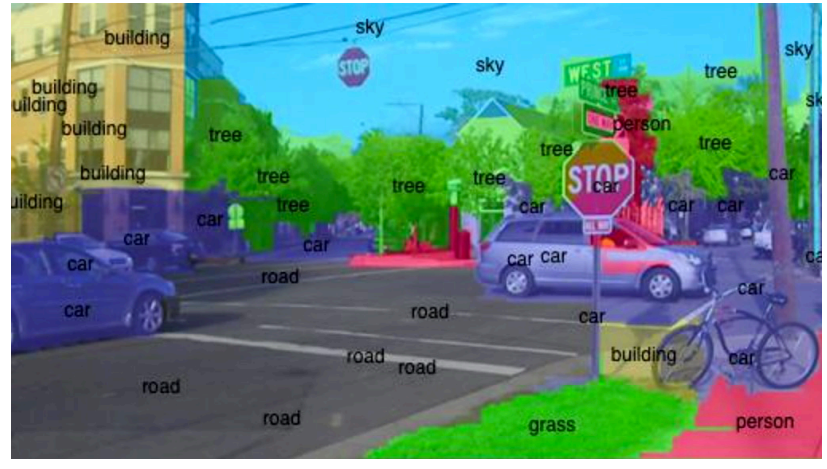
- Detect Pedestrians
- Stop Signs
- Traffic Lights
- Maps...

Computer Vision

Computer Vision = how can a computer make sense of images/videos?

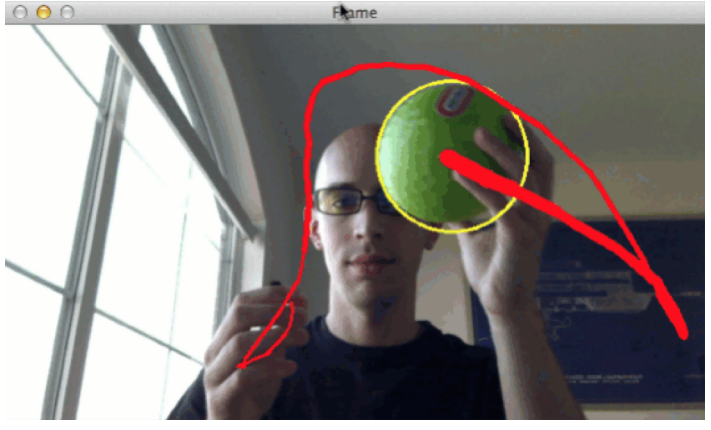
- Image → Knowledge



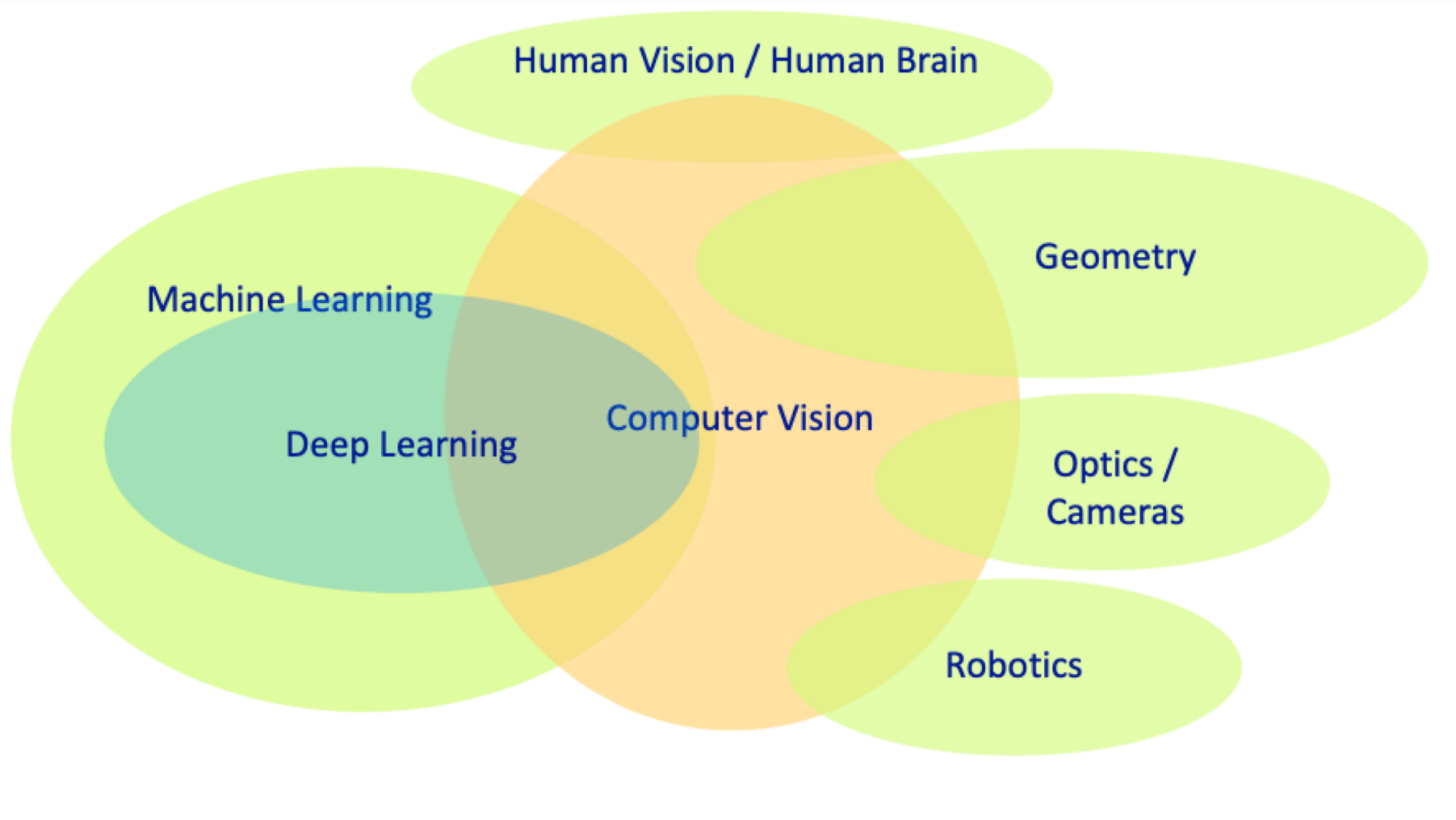


Purpose:

- Classification
- Object Detection
- 3D Reconstruction
- Pose Estimation
- Video Tracking
- So much more...!

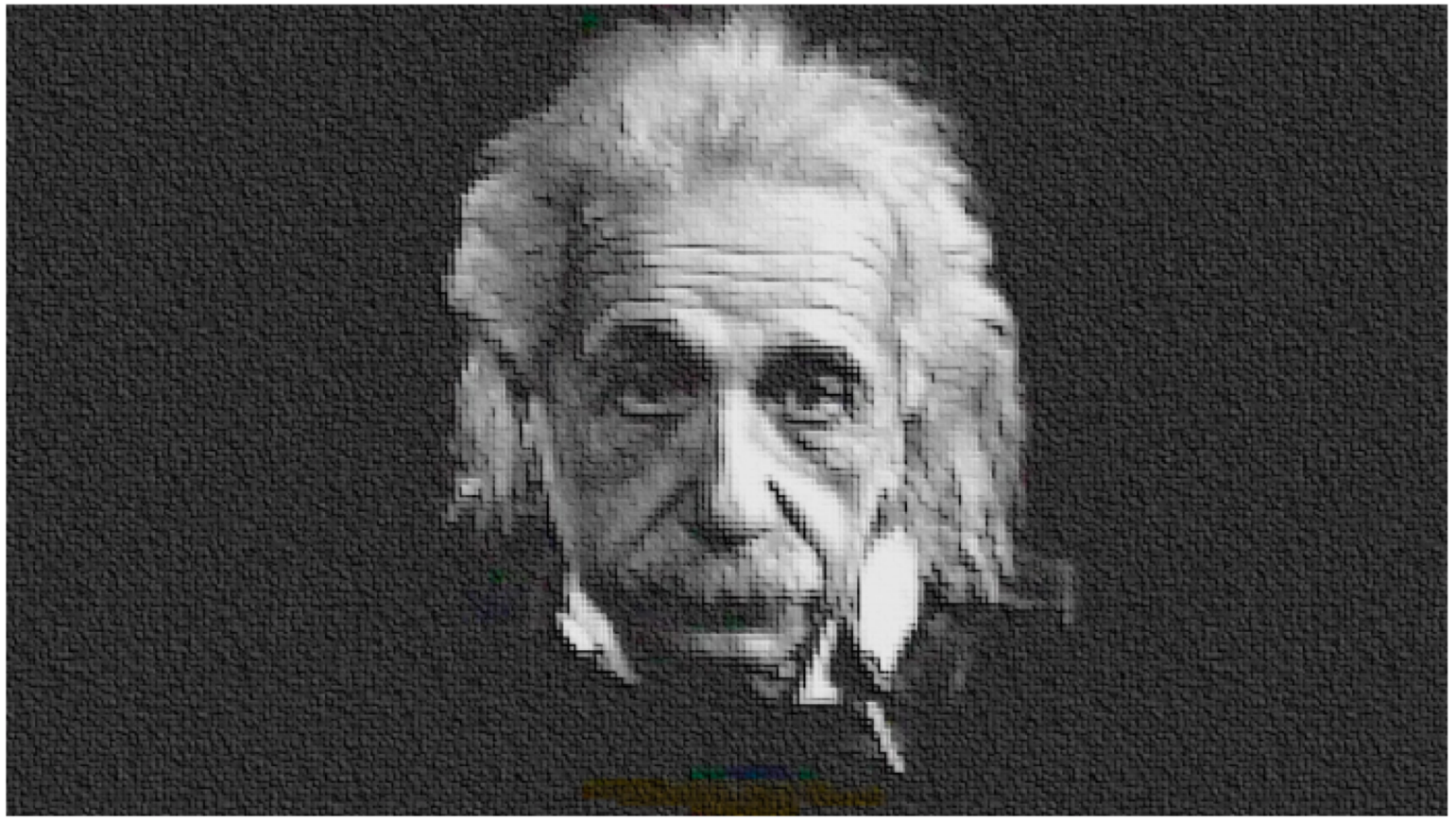


Computer Vision is a Big Field



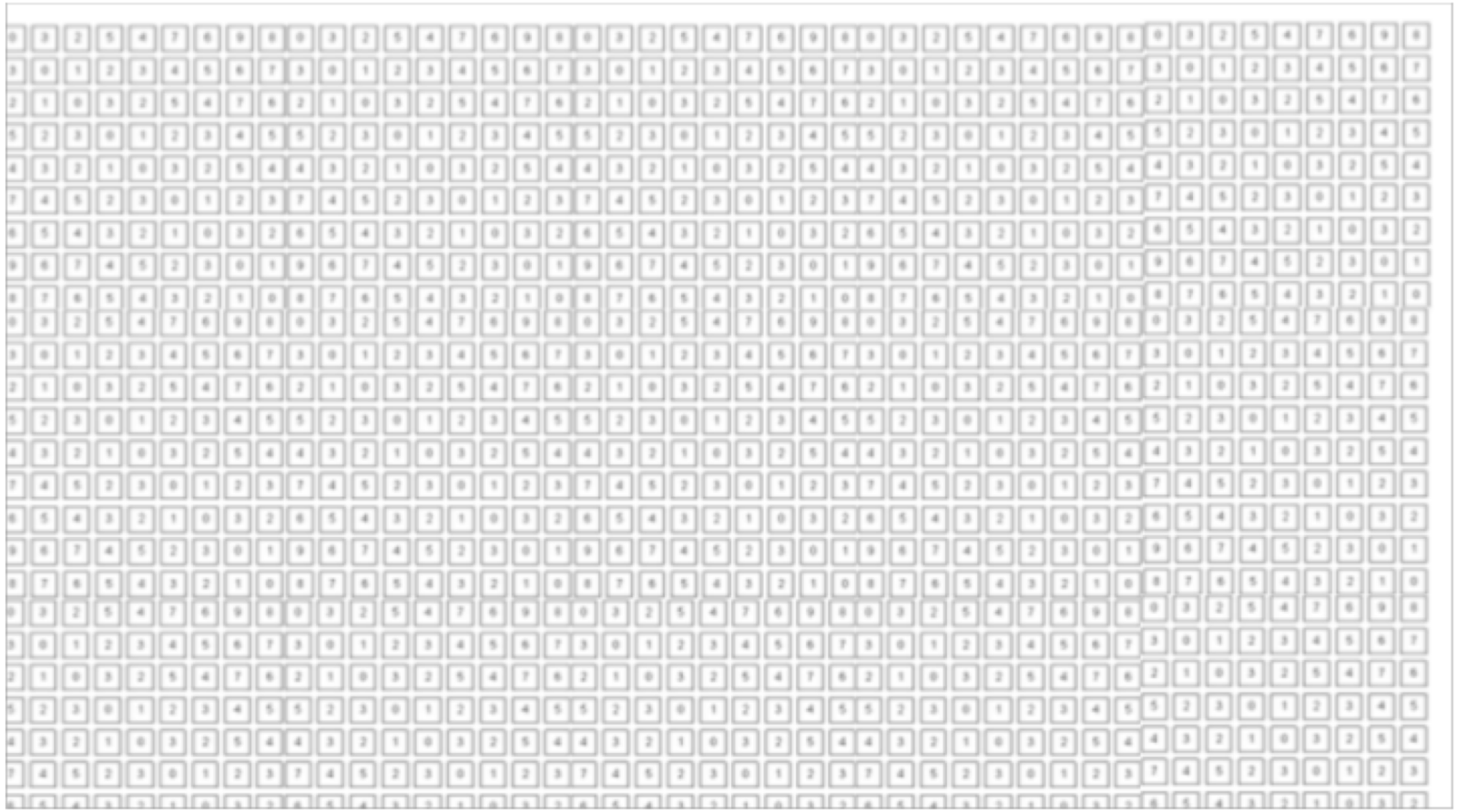
Computer Vision is Hard!

This is an image to us:



Computer Vision is Hard!

This is an image to a computer:



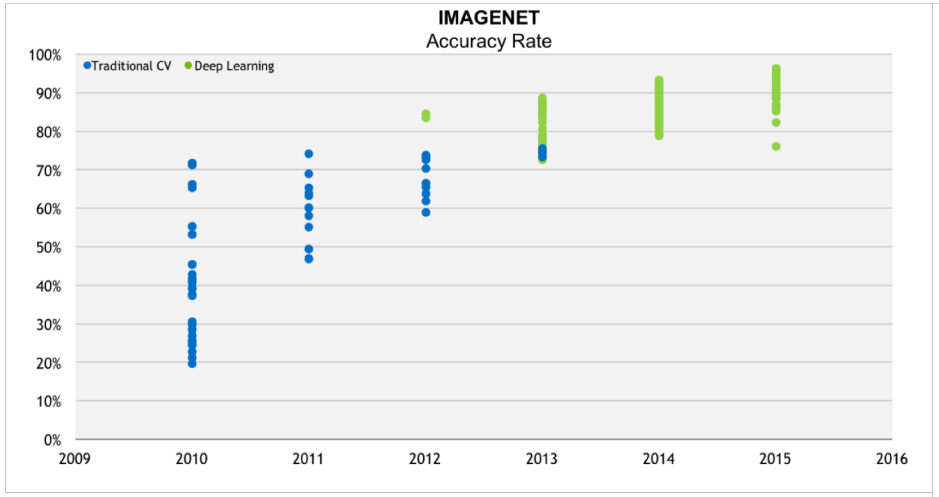
Types of Computer Vision

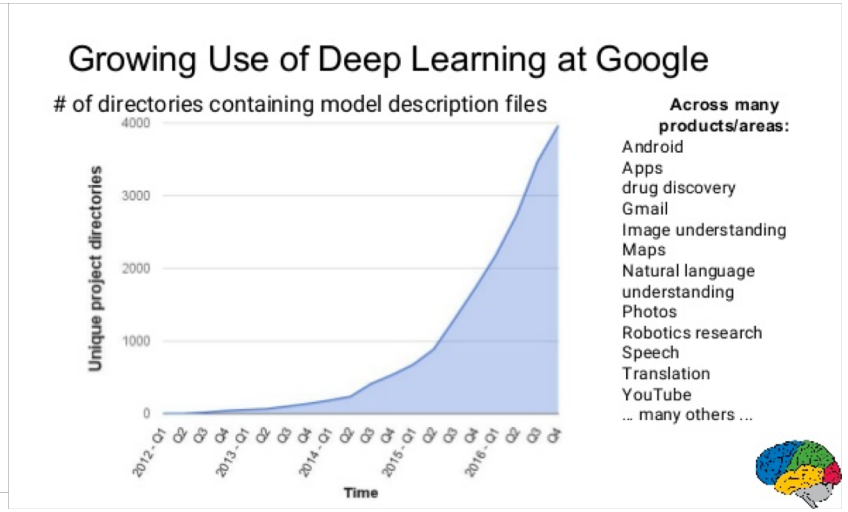
Two Main Types of Computer Vision:

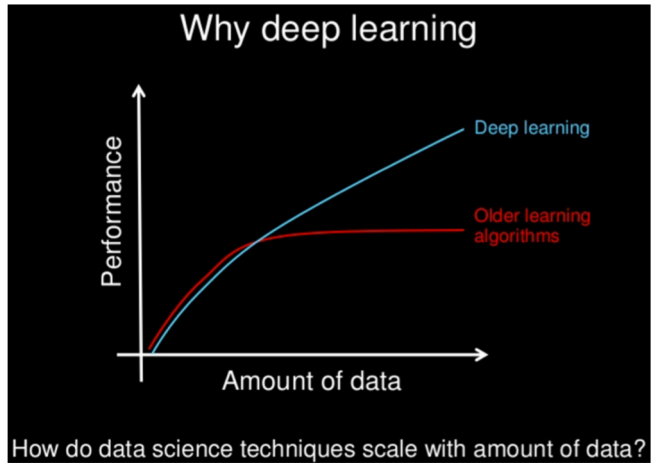
- Pre-ML (Classical)
- Post-ML (Deep Learning)

The adoption of Deep Learning (~2012) accelerated Computer Vision

https://twitter.com/olivercameron/status/1116671064600301568







Classical Computer Vision

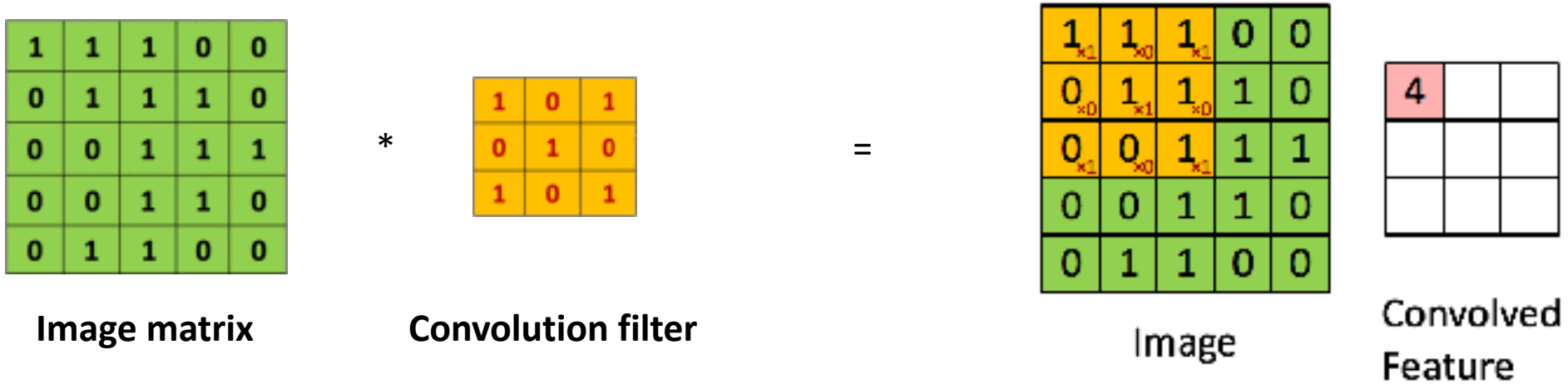
One big problem is Edge Detection!

- We want to detect all the edges in an image

Classical Computer Vision

Idea: "convolution" – apply a "kernel" on the image

- Image is a matrix (pixels)
- Kernel is a matrix
- Convolve them!



Classical Computer Vision

Sobel Operator: uses unique kernels to highlight edges in image

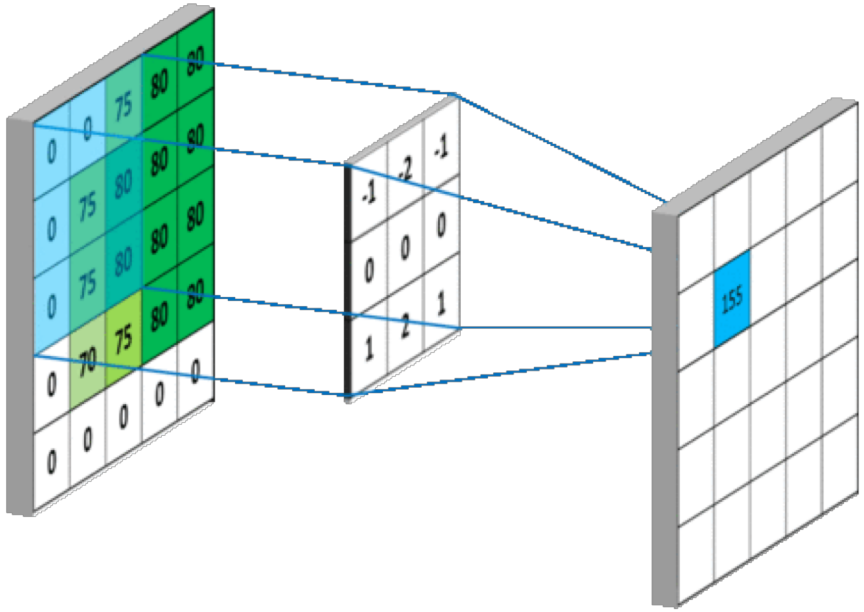
- Horizontal Edges
- Vertical Edges

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

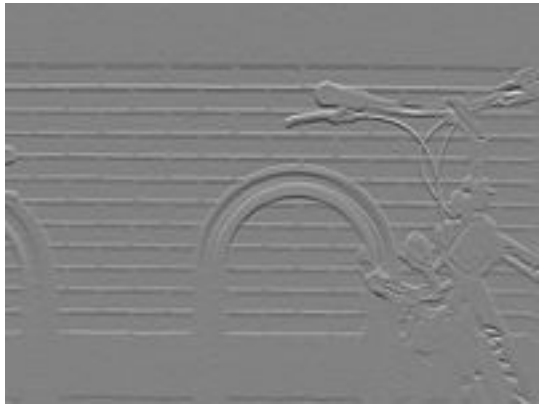


Classical Computer Vision

Sobel Operator: uses unique kernels to highlight edges in image

image

Sobel Vertical gradient



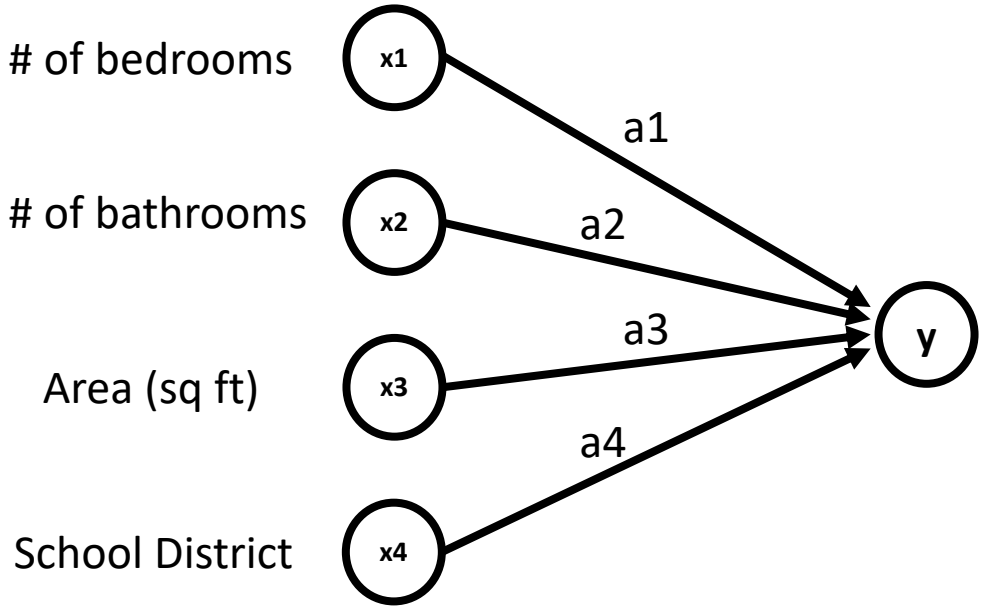
Sobel Horizontal gradient

Sobel combined

Deep Learning

What is a Neural Network?

Example: We are trying to predict the price of a house:



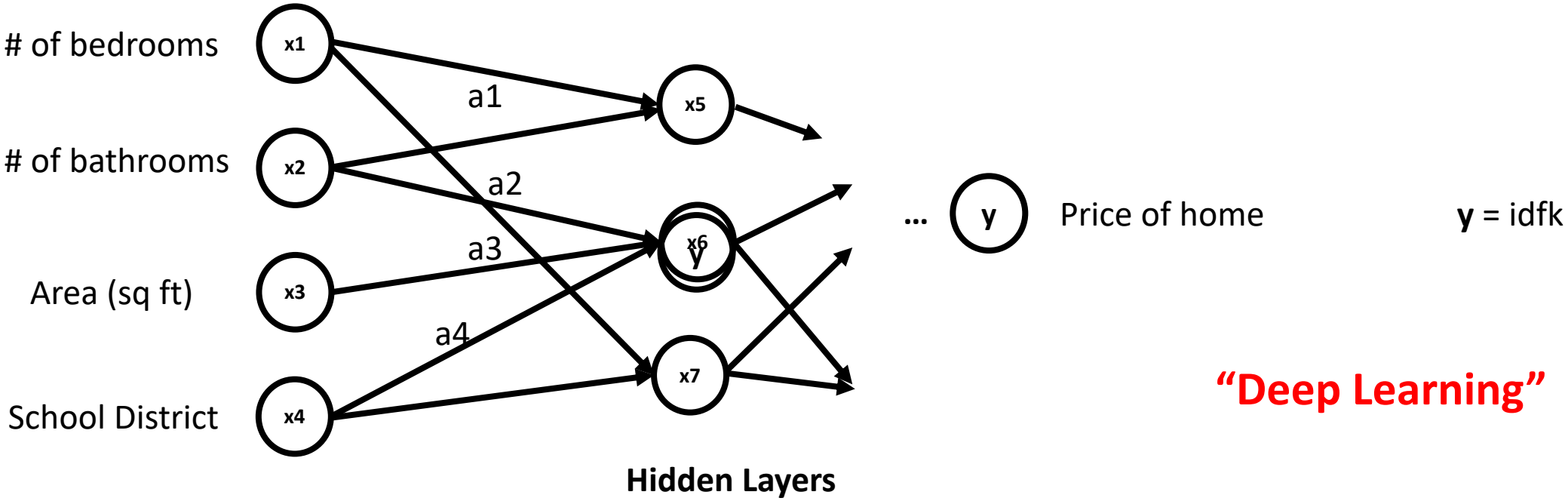
y = (a1*x1) + (a2*x2) + (a3*x3) + (a4*x4)

"Shallow Learning"

Deep Learning

What is a Neural Network?

Example: We are trying to predict the price of a house:

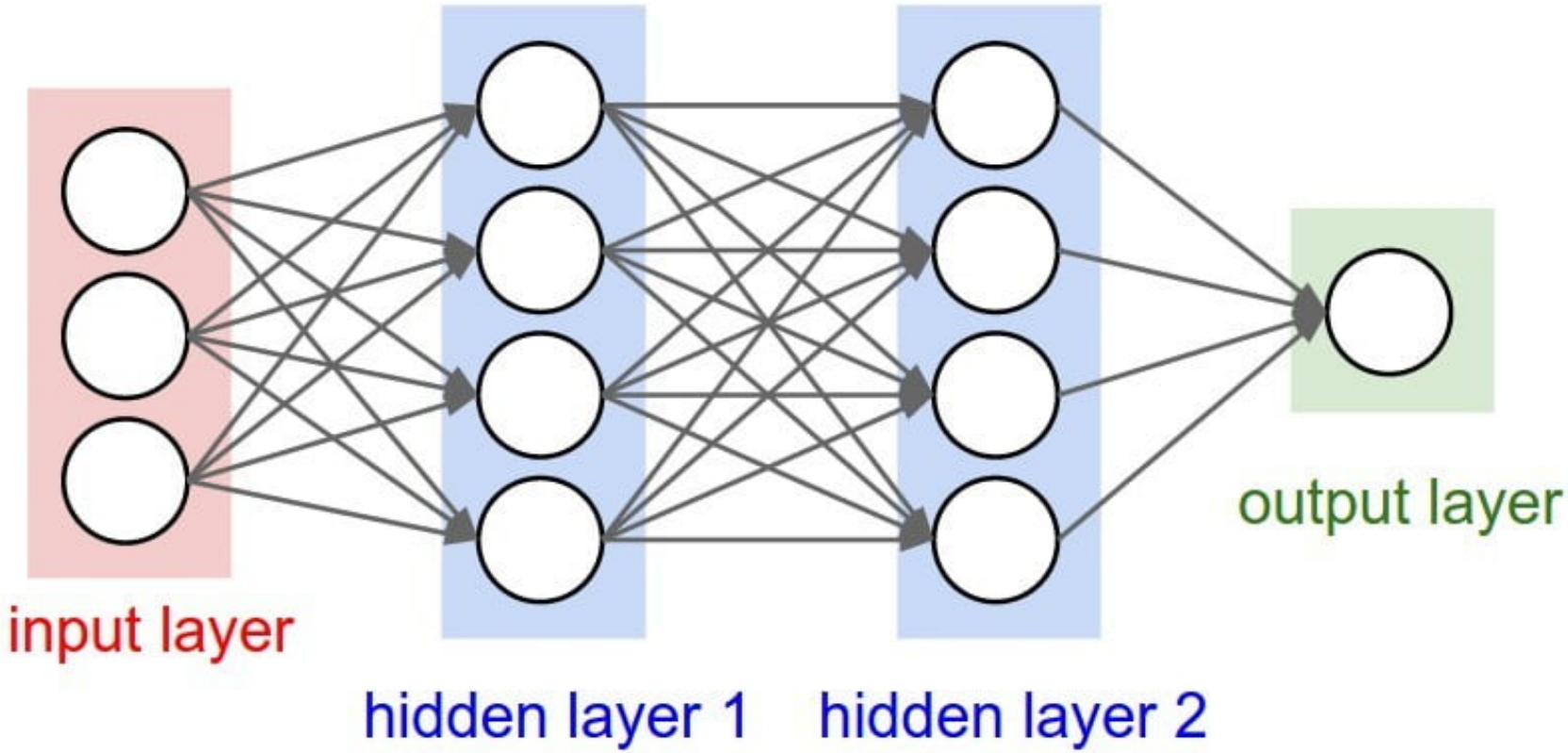


y = idfk

“Deep Learning”

Deep Learning

What is a Neural Network?

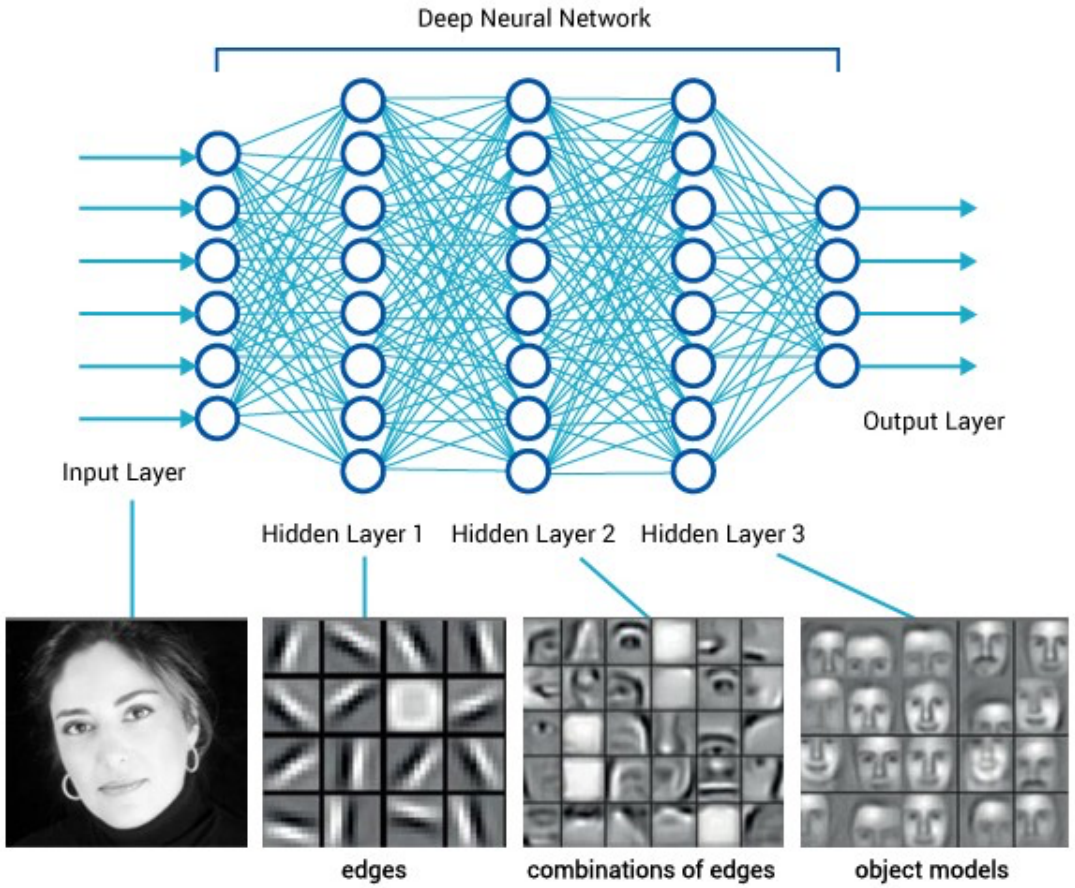


Deep Learning

What does this have to do with Computer Vision?

Deep Learning

What does this have to do with Computer Vision?



Deep Learning

Convolutional Neural Networks (CNNs)