Computer Vision

Teaching cameras to “see”

Arjun Jain

What is **Computer Vision**?

- Automatic understanding of images and video
    - **Measurement** : Computing properties of the 3D world from visual data
    - **Perception and interpretation**: Algorithms and representations to allow a machine to recognize objects, people, scenes, and activities.

Computer Vision: Measurement

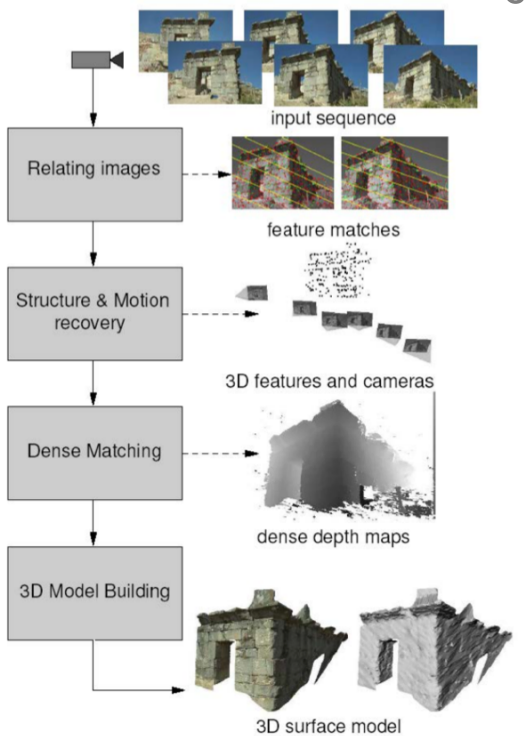
- Measurement

Real-time stereo

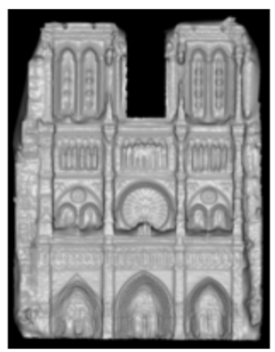
NASA Mars Rover

Pollefeys et al.

Structure-from-motion



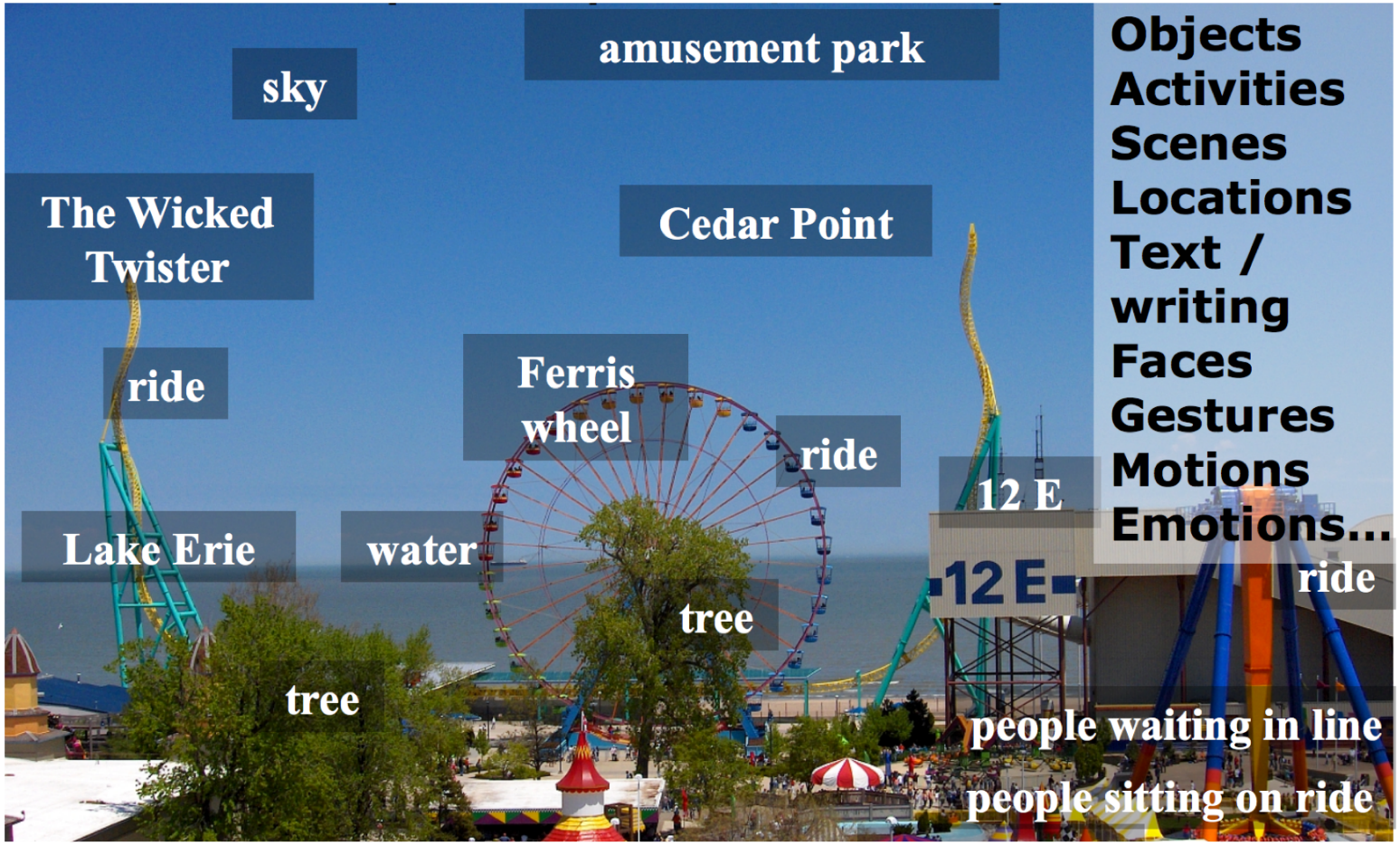
Multi-view stereo for community photo collections



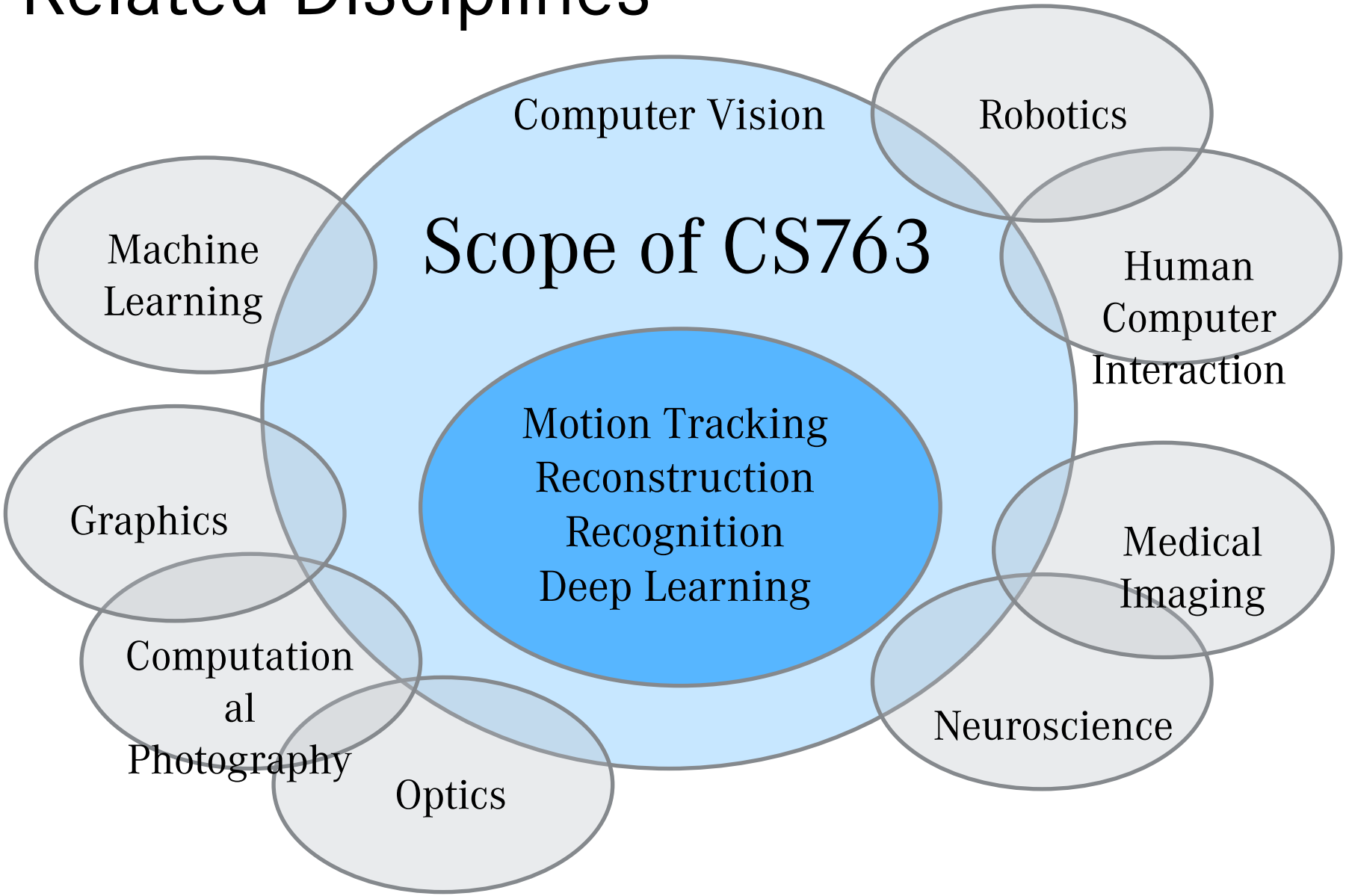
Goesele et al.

Computer Vision: Perception

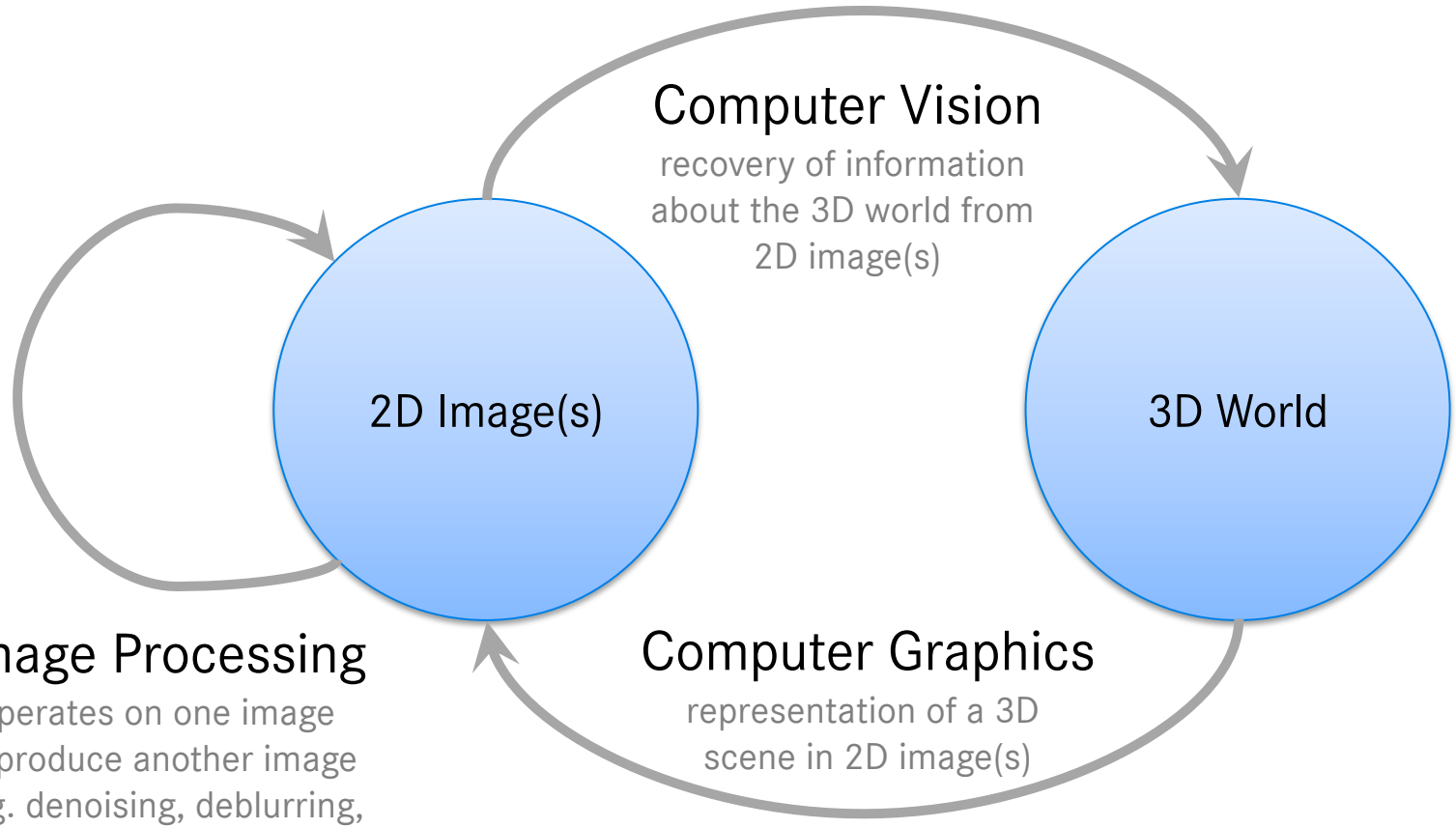
- Perception and interpretation



Related Disciplines



Computer Vision, Computer Graphics and Image Processing



Why do Computer Vision?

- As image sources multiply, so do applications
  - Relieve humans of boring, easy tasks
  - Enhance human abilities: human computer interaction, visualization
  - Perception for robotics / autonomous agents
  - Organize and give access to visual content

Why Computer Vision?

Images are everywhere!

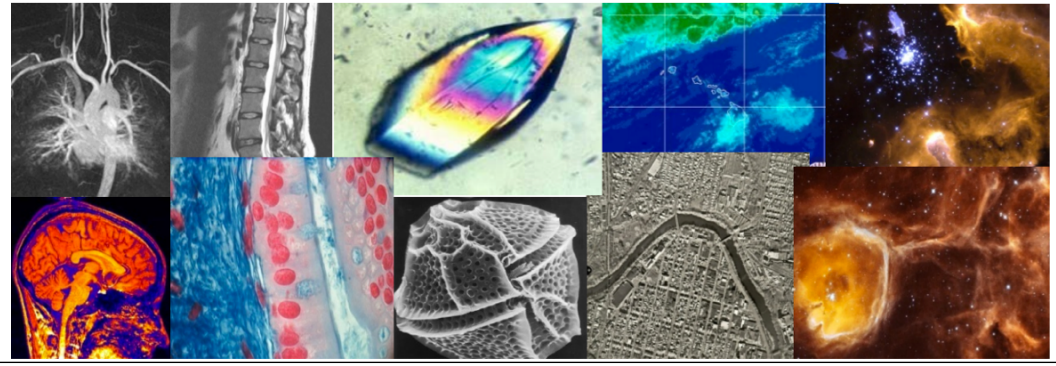
Personal Photo Albums

Movies, News, Sports

Google Image Search Picasa flickr webshots picsearch YouTube Broadcast Yourself

Surveillance and Security

Medical and Scientific Images



Why is Computer Vision so Hard?

Consider Image Classification: a core task in Computer Vision

(assume given set of discrete labels)

{dog, cat, truck, plane, ...}

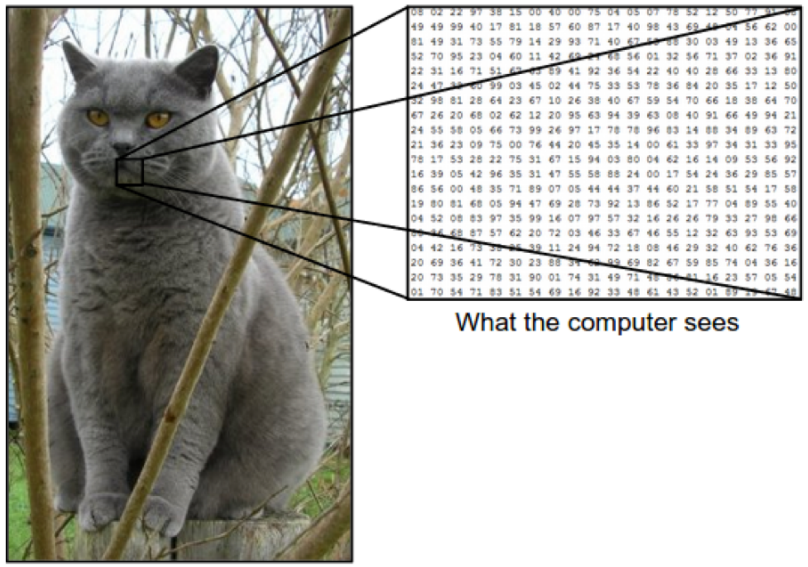
→ cat

Why is Computer Vision so Hard?

Images are represented as 3D arrays of numbers, with integers between [0, 255].

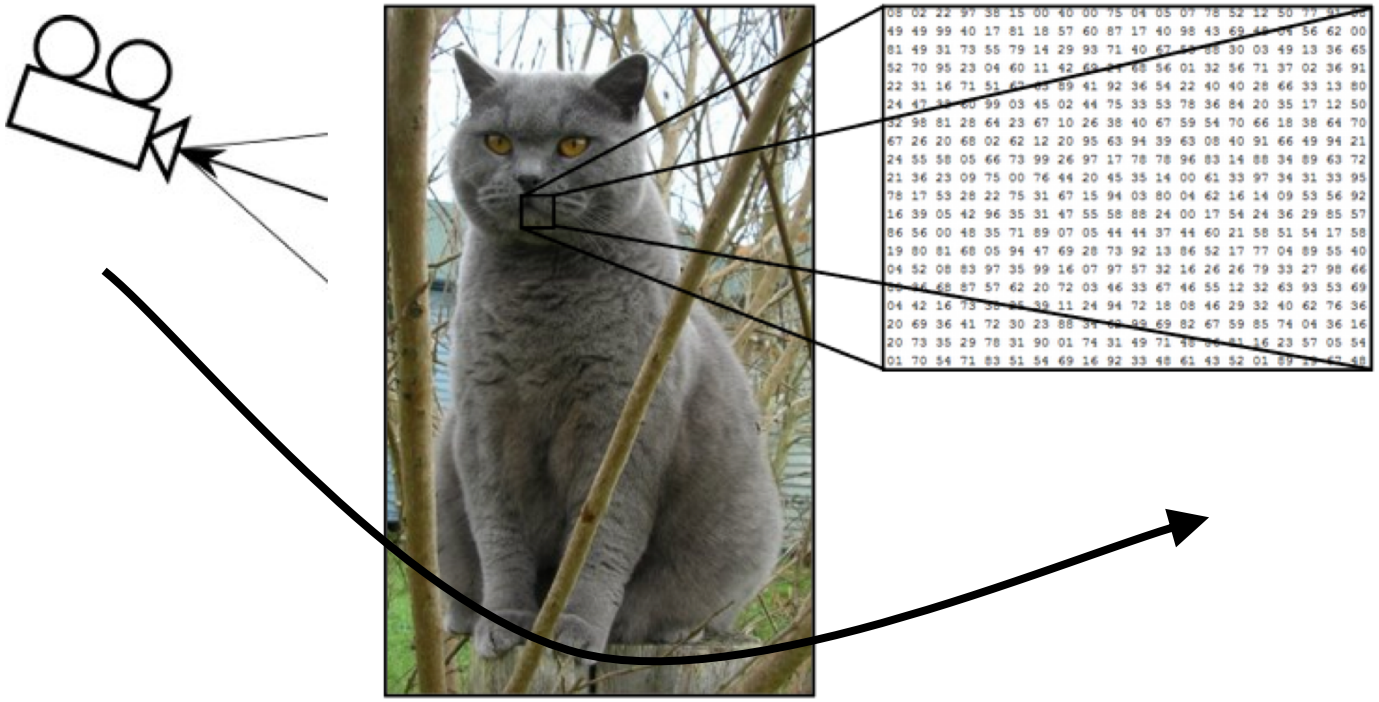
E.g.
300 x 100 x 3

(3 for 3 color channels RGB)

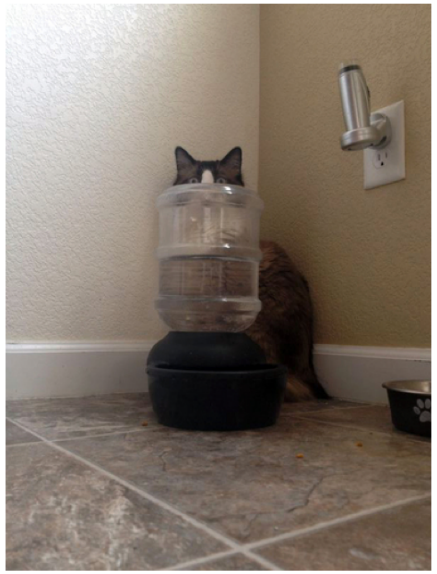


Challenges: Invariant to Illumination

Challenges: Invariant to Viewpoint



Challenges: Deal with Occlusion


Challenges: Invariant to Deformation

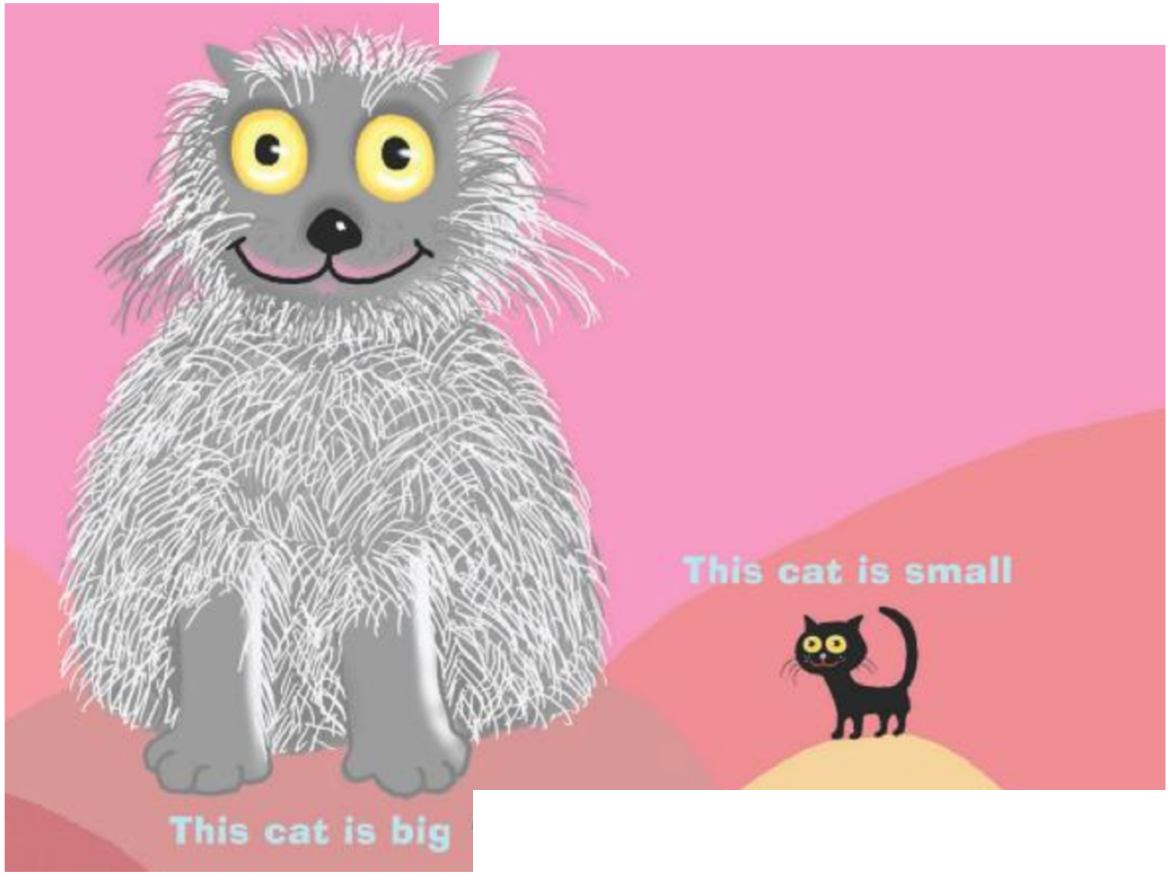
Challenges: Deal with Background Clutter

Challenges: Deal with Intra-class Variation






Challenges: Deal with Scale Changes



Challenges: Deal with Motion

Challenges or Opportunities?

- Images are confusing, but they also reveal the structure of the world through numerous cues

- Computer Vision: interpret the cues (the human visual system does this all the time!)

- E.g. we interpret depth in images using both **physiological** and **psychological** cues

    - **Physiological** cues require both eyes to be open (binocular)
    - Other cues are available also when looking at images with only one open eye (monocular). All **psychological** cues are monocular

Depth Cues: Linear Perspective

When looking down a straight level road we see the parallel sides of the road meet in the horizon

Depth Cues: Aerial Perspective

The mountains in the horizon look always slightly bluish or hazy. The reason for this are small water and dust particles in the air between the eye and the mountains. The farther the mountains, the hazier they look.

Depth Ordering Cues: Occlusion

When objects block each other out of our sight, we know that the object that blocks the other one is closer to us. The object whose outline pattern looks more continuous is felt to lie closer.

Depth Cues: Texture Gradient

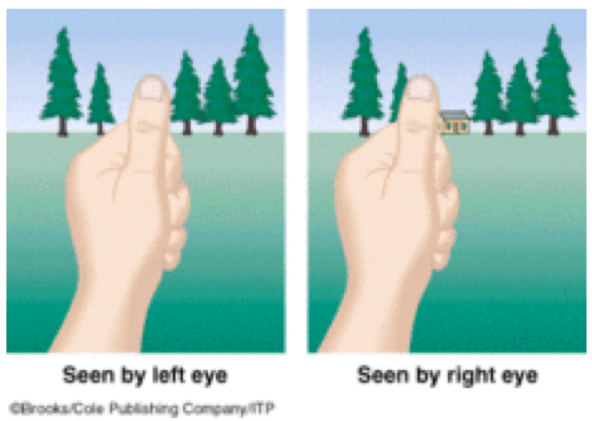
The closer we are to an object the more detail we can see of its surface texture. So objects with smooth textures are usually interpreted being farther away.

Depth Cues: Texture Gradient

When we know the location of a light source and see objects casting shadows on other objects, we learn that the object shadowing the other is closer to the light source.

Depth Cues

- **Binocular Parallax:** As our eyes see the world from slightly different locations, the images sensed by the eyes are slightly different. This difference in the sensed images is called binocular parallax. Human visual system is very sensitive to these differences, and binocular parallax is the most important depth cue for medium viewing distances. The sense of depth can be achieved using binocular parallax even if all other depth cues are removed.



The closer the object, the larger the disparity. Far away objects will seem almost the same by both eyes.

Depth Cues

- **Monocular Movement Parallax:** If we close one of our eyes, we can perceive depth by moving our head. This happens because human visual system can extract depth information in two similar images sensed after each other, in the same way it can combine two images from different eyes.

- **Retinal Image Size:** When the real size of the object is known, our brain compares the sensed size of the object to this real size, and thus acquires information about the distance of the object.

Grouping Cues

(color, texture, proximity)

Grouping Cues: “Common Fate”

Bottom Line

- Perception is an inherently ambiguous and ill posed problem:
  - Many different 3D scenes could have given rise to the same 2D picture

- Possible solutions: Bring in more constraints (more images)
- Use prior knowledge about the structure of the world
- Need a combination of different methods!

Every Picture Tells a Story

Goal of computer vision is to write computer programs that can interpret images

Computer Vision

- Has been around since the 1960s
- What has changed?

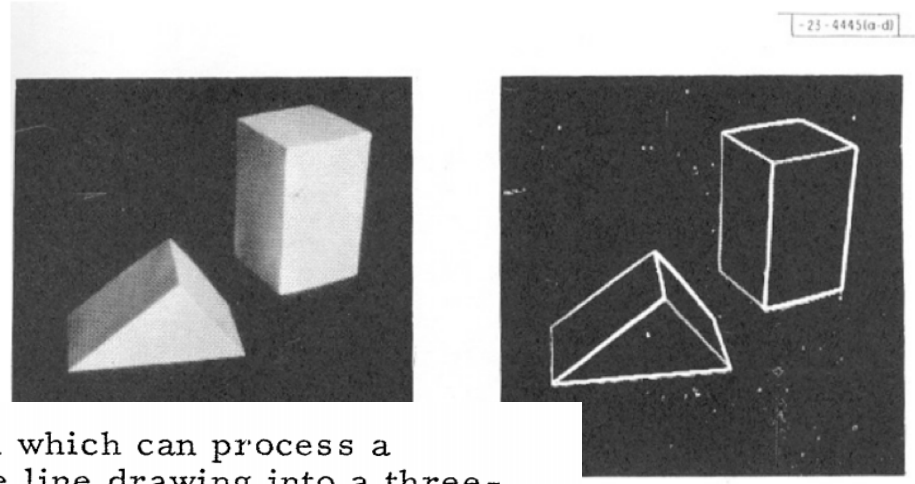
1. Increasing availability of cheap, powerful cameras (e.g. digital cameras, webcams) and other sensors

2. Increasing availability of massive amounts of labeled and unlabeled image and multimedia content on the web (e.g. face databases, etc.)

3. Increasing availability of cheap, powerful computers (processing speed and memory capacity - 10 Tflops by 1 Titan X!). Anyone heard of Titan V? Tesla V100?

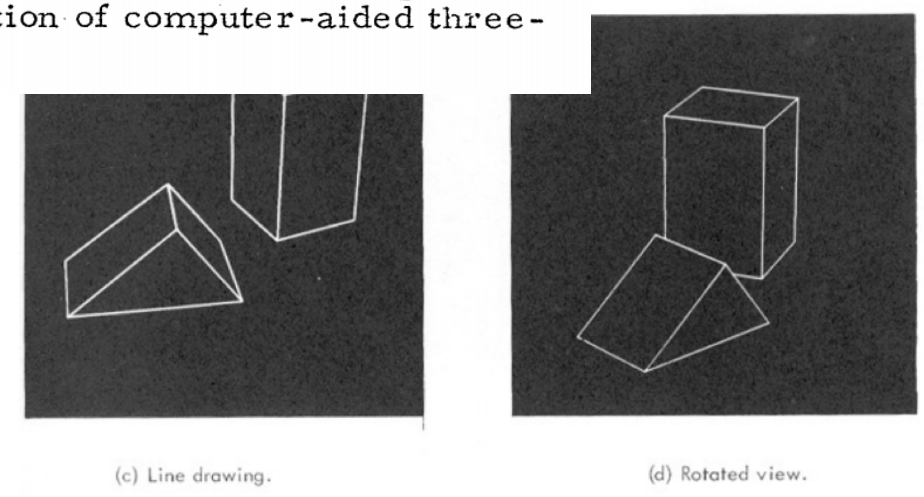
4. Techniques from machine learning and statistics which lead to more complex, data-driven models and algorithms (e.g. deep learning!)

L. G. Roberts, Machine Perception of Three Dimensional Solids, Ph.D. thesis, MIT Department of Electrical Engineering, 1963.



(b) Differentiated picture.

A computer program has been written which can process a photograph into a line drawing, transform the line drawing into a three-dimensional representation, and finally, display the three-dimensional structure with all the hidden lines removed, from any point of view. The 2-D to 3-D construction and 3-D to 2-D display processes are sufficiently general to handle most collections of planar-surfaced objects and provide a valuable starting point for future investigation of computer-aided three-dimensional systems.



(c) Line drawing.

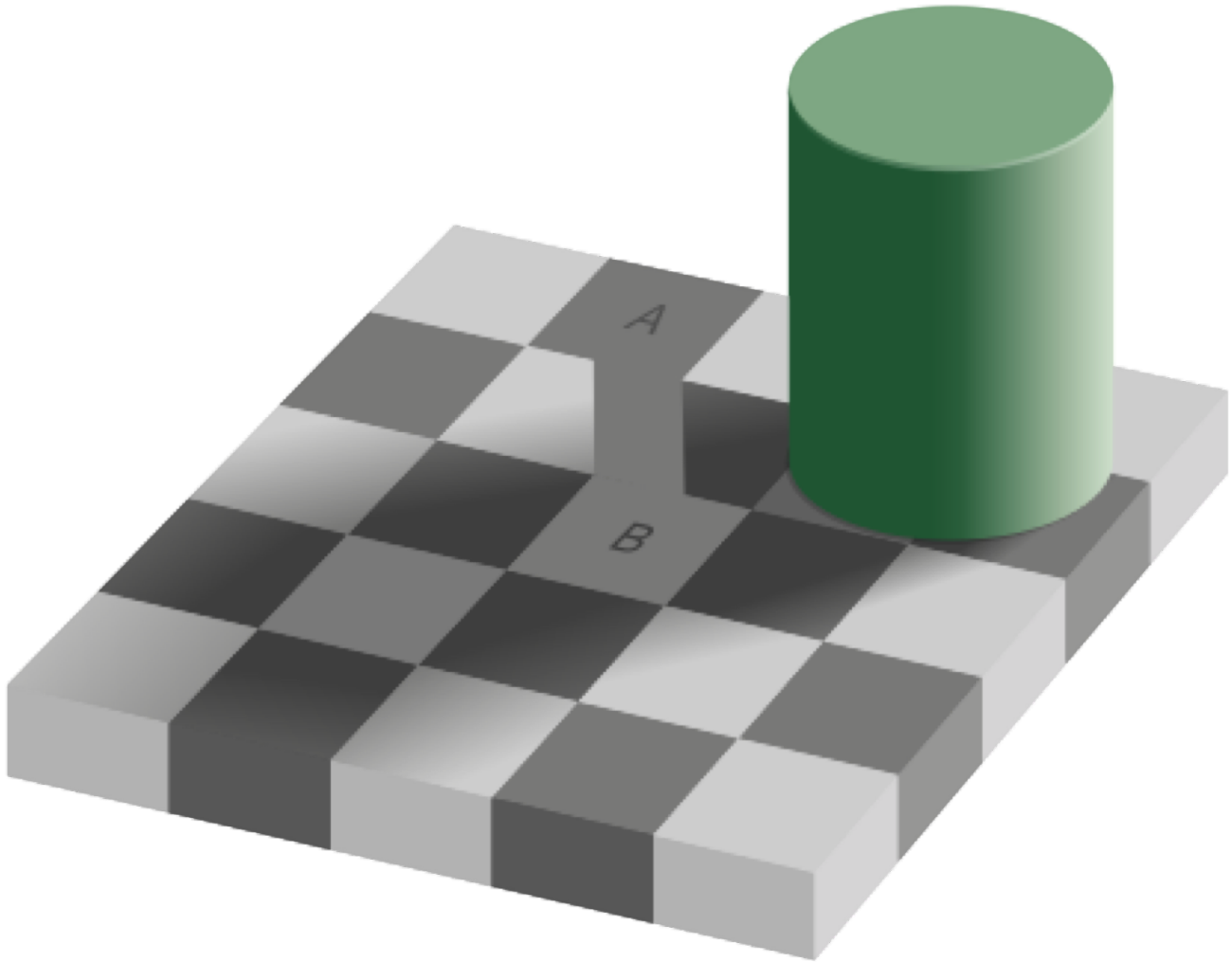
(d) Rotated view.

Human Perception Has Issues

Sinha and Poggio, Nature, 1996

- Absolutely identical in terms of nose, eyes, mouth and their spatial arrangements!

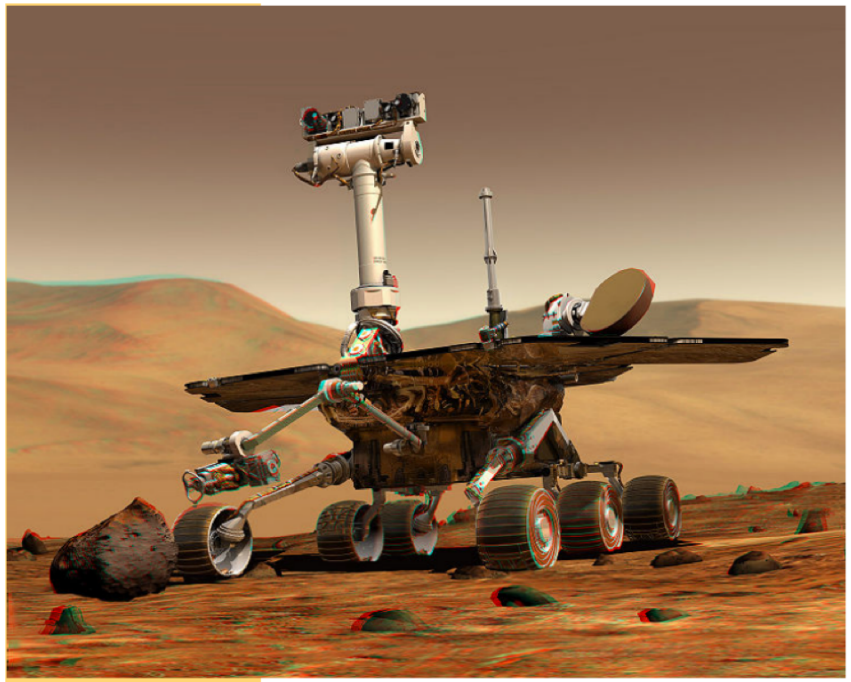
Human Perception Has Issues



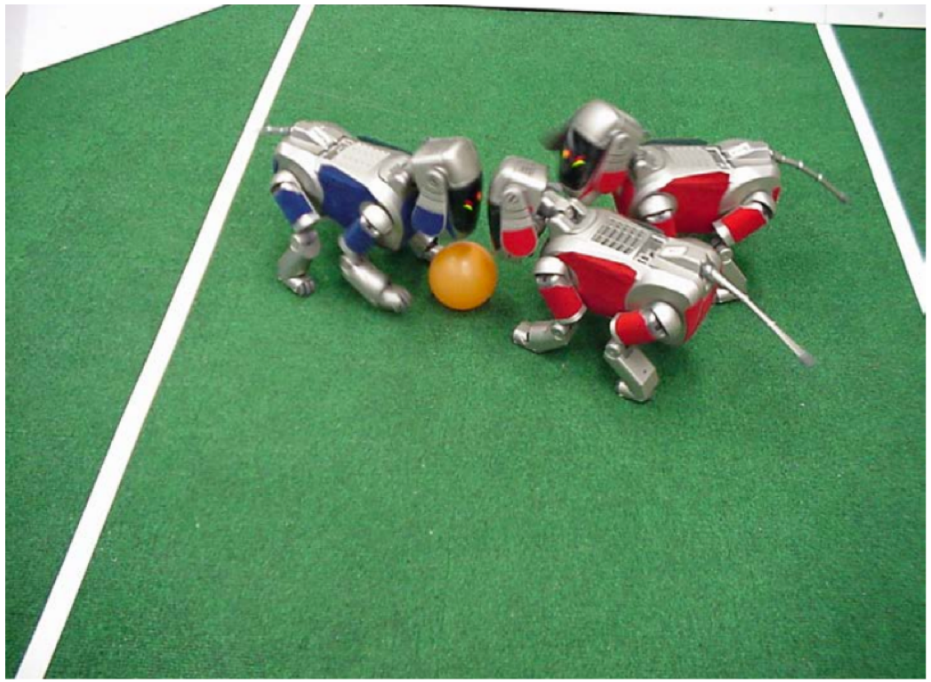
Can Computers Beat Human Vision?

- Yes and no
  - humans are usually much better at “generic” problems
  - computers can be better at “specific” problems

Vision in Robotics

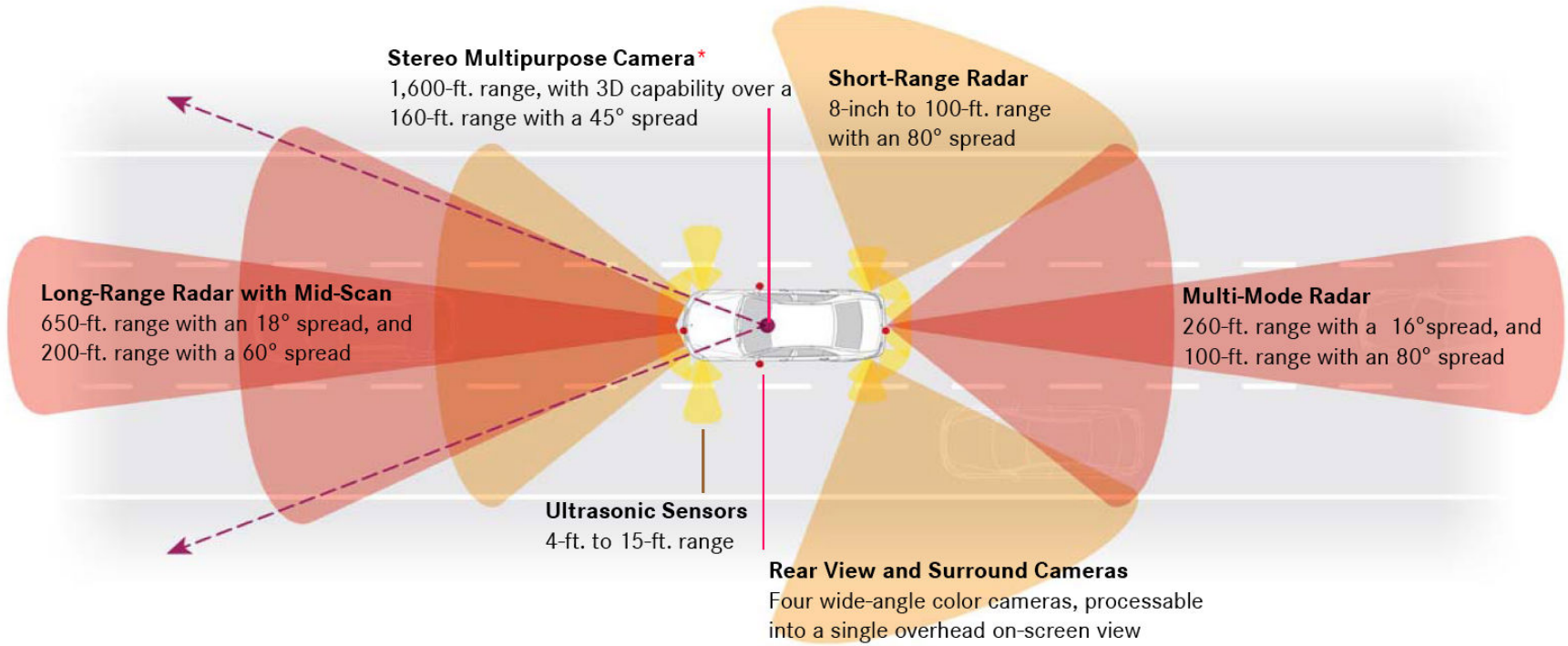


NASA's Mars Spirit Rover
http://en.wikipedia.org/wiki/Spirit_rover



http://www.robocup.org/

Vision in Autonomous Cars



AutoCars – Uber bought CMU's lab

Current State-of-the-Art

- Many of these applications are less than 5 years old
- This is a very active research area, and **rapidly** changing!
- Many new apps in the next 5 years

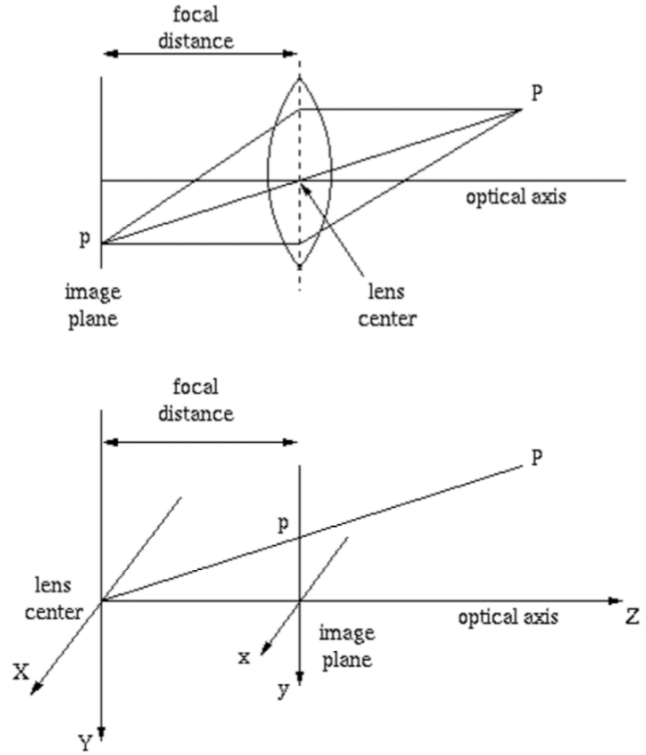
A few important topics in CV

1. Camera geometry
2. Shape from X
3. Motion Estimation
4. Machine learning in computer vision

A few important topics in CV

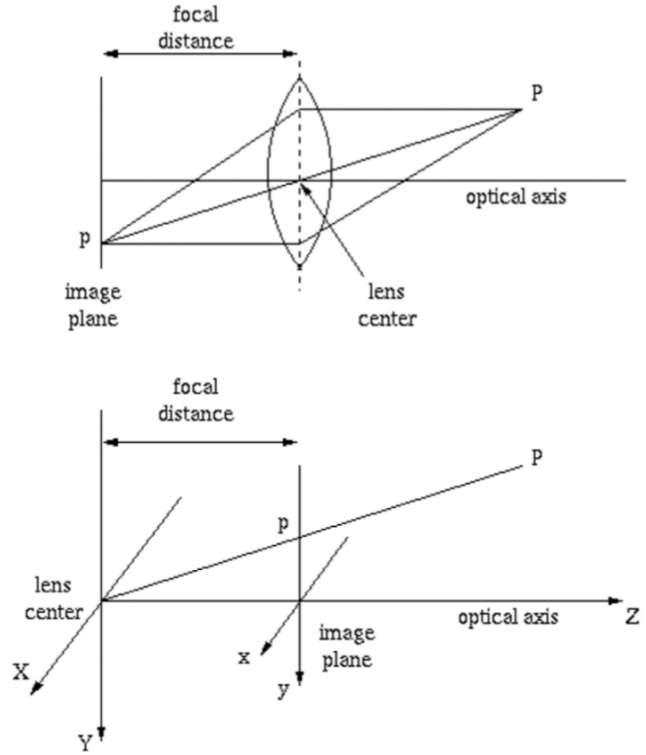
1. Camera geometry
2. Shape from X
3. Motion Estimation
4. Machine learning in computer vision

1. Camera Geometry



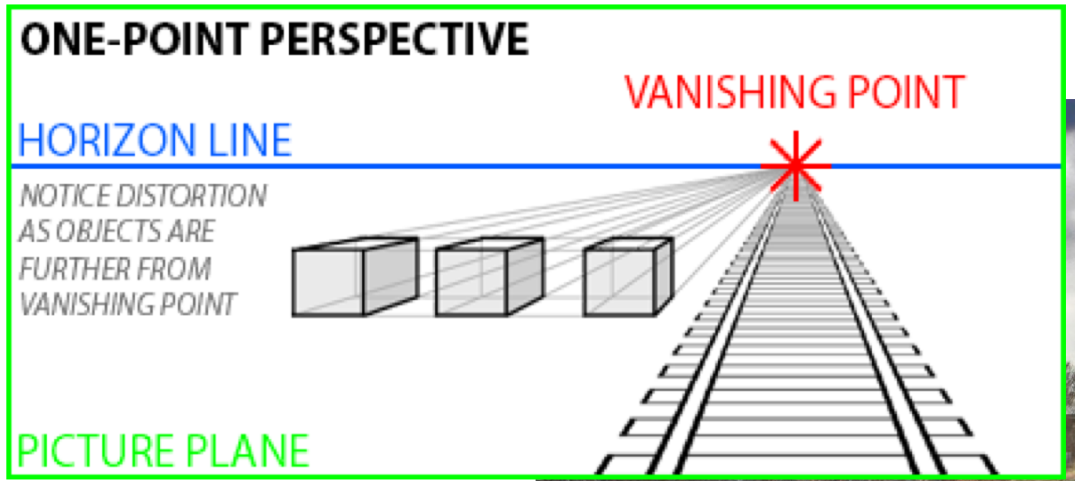
- Relationship between object coordinates (given by a vector **P** in 3D) and image coordinates (given by vector **p** in 2D)
- Effect of various intrinsic camera parameters (focal length of lens, nature of the lens, aspect ratio of sensor array, etc.) on image formation
- Effect of various extrinsic camera parameters on image formation

1. Camera Geometry

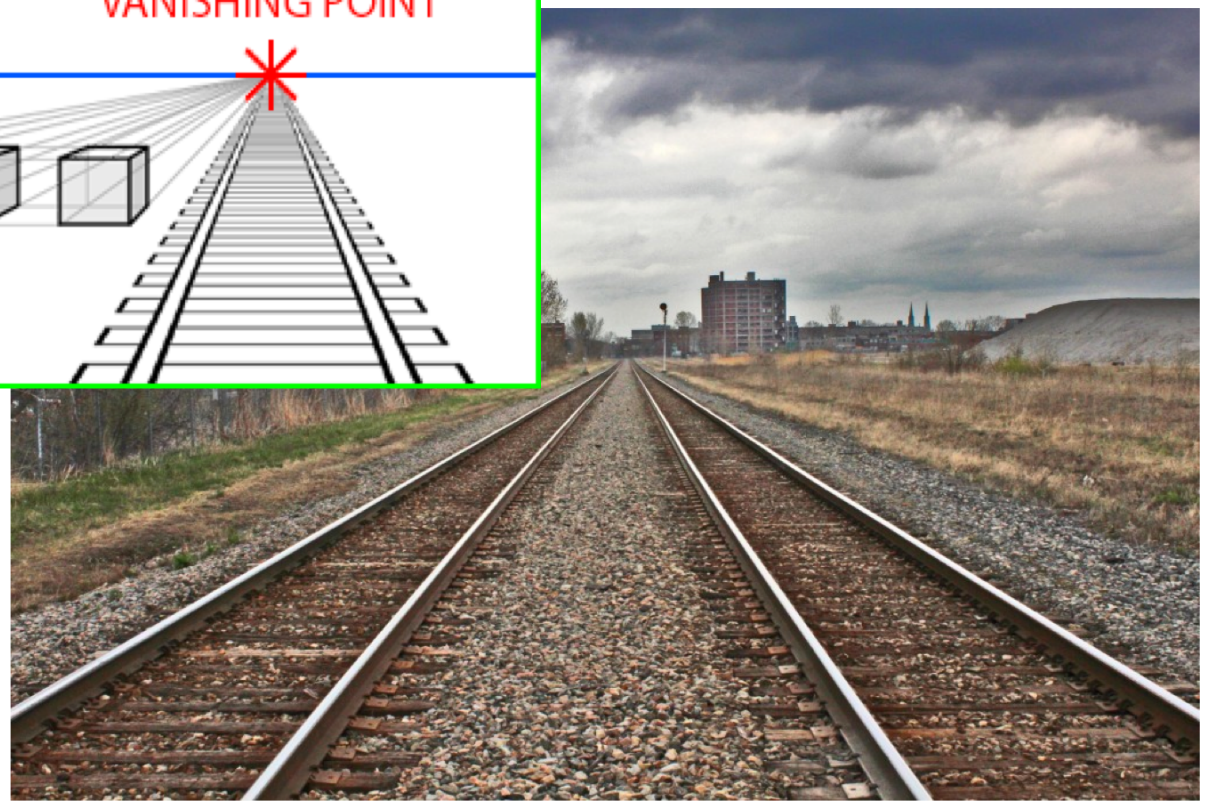


- Let's say we take a picture of a simple object of known geometry (example: chessboard, cube, etc.).
- Given the 3D coordinates of N points on the object, and their corresponding 2D coordinates in the image plane, can you determine the camera parameters such as focal length?
- Answer is **YES** we can! This process is called as camera calibration.

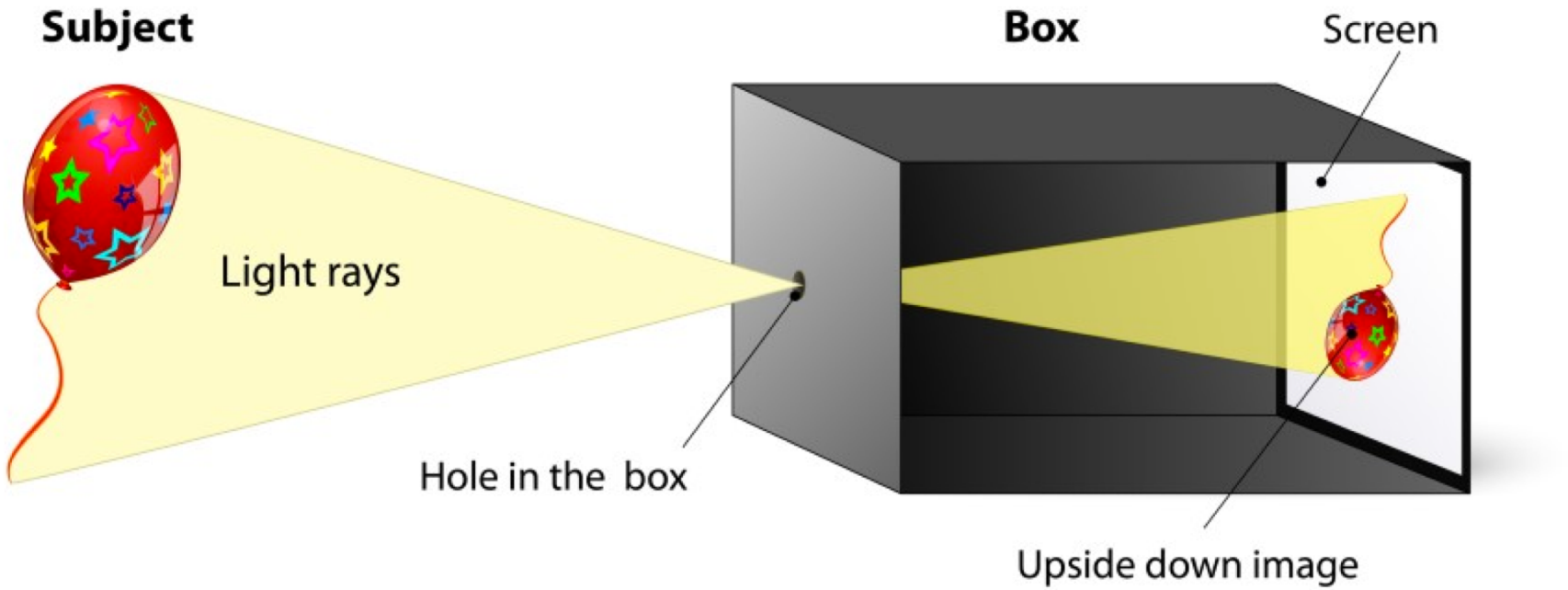
1. Camera Geometry (via Vanishing Points)



http://www.atpm.com/9.09/design.shtml



Pin Hole Camera Model



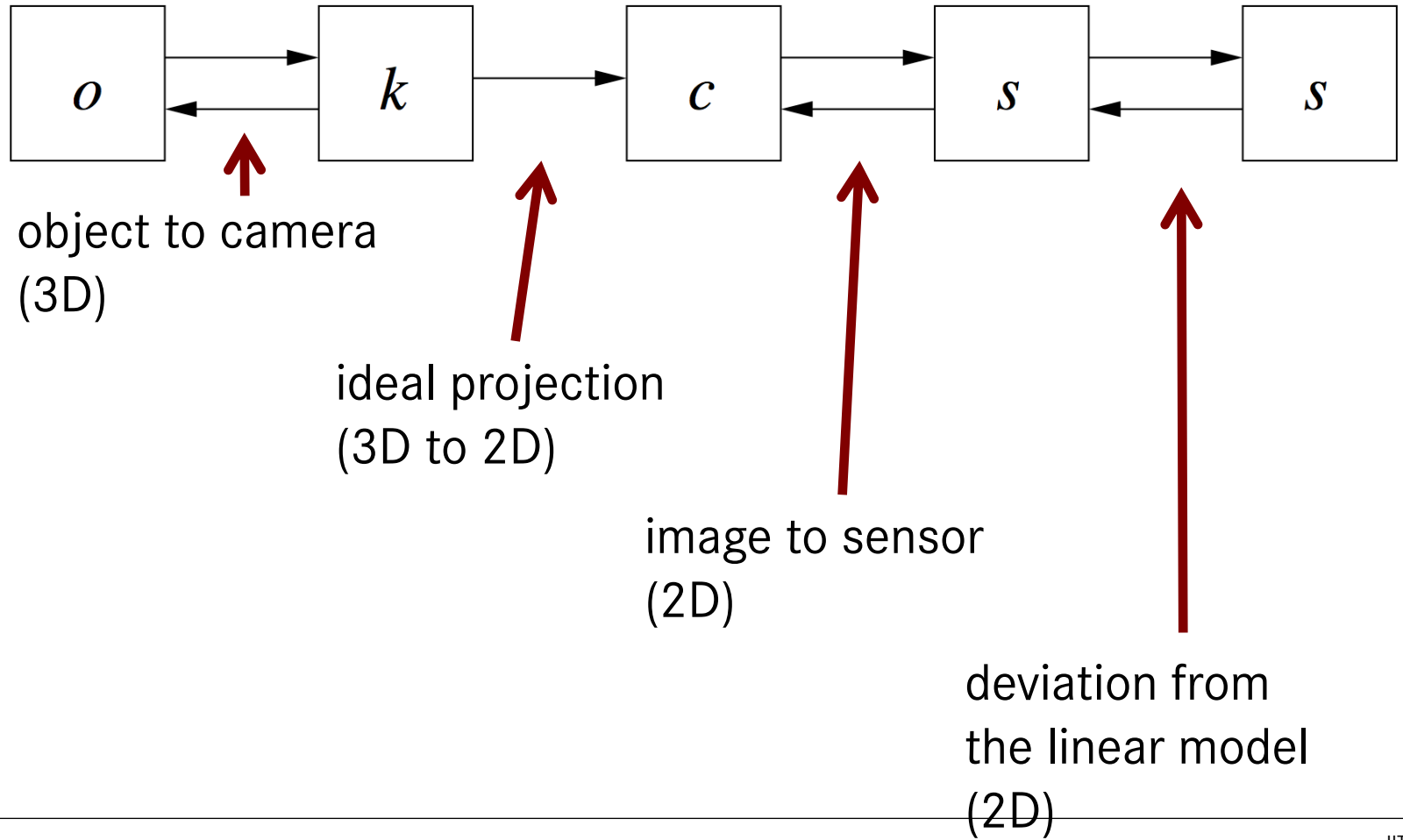
Transformation from 3D world to image (sensor)

We want to compute the mapping

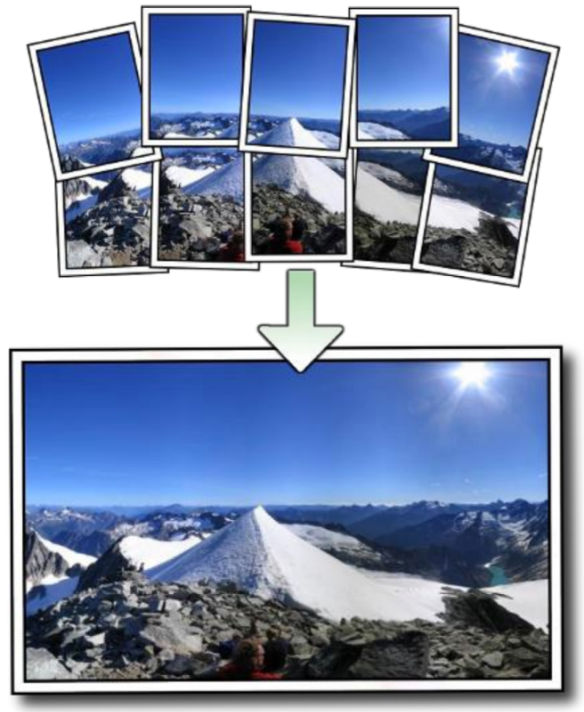
$$\begin{bmatrix} {}^{s}x \\ {}^{s}y \\ 1 \end{bmatrix} = {}^{s}\mathsf{H}_{c}\ {}^{c}\mathsf{P}_{k}\ {}^{k}\mathsf{H}_{o} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

in the sensor system	image plane to sensor	camera to image	object to camera	in the object system

From the World to the Sensor



1. Camera Geometry - Image Mosaicing/Panoramas



Generating a panorama out of a series of pictures of a scene from different viewpoints.

http://cs.bath.ac.uk/brown/autostitch/autostitch.html

A few topics in CV

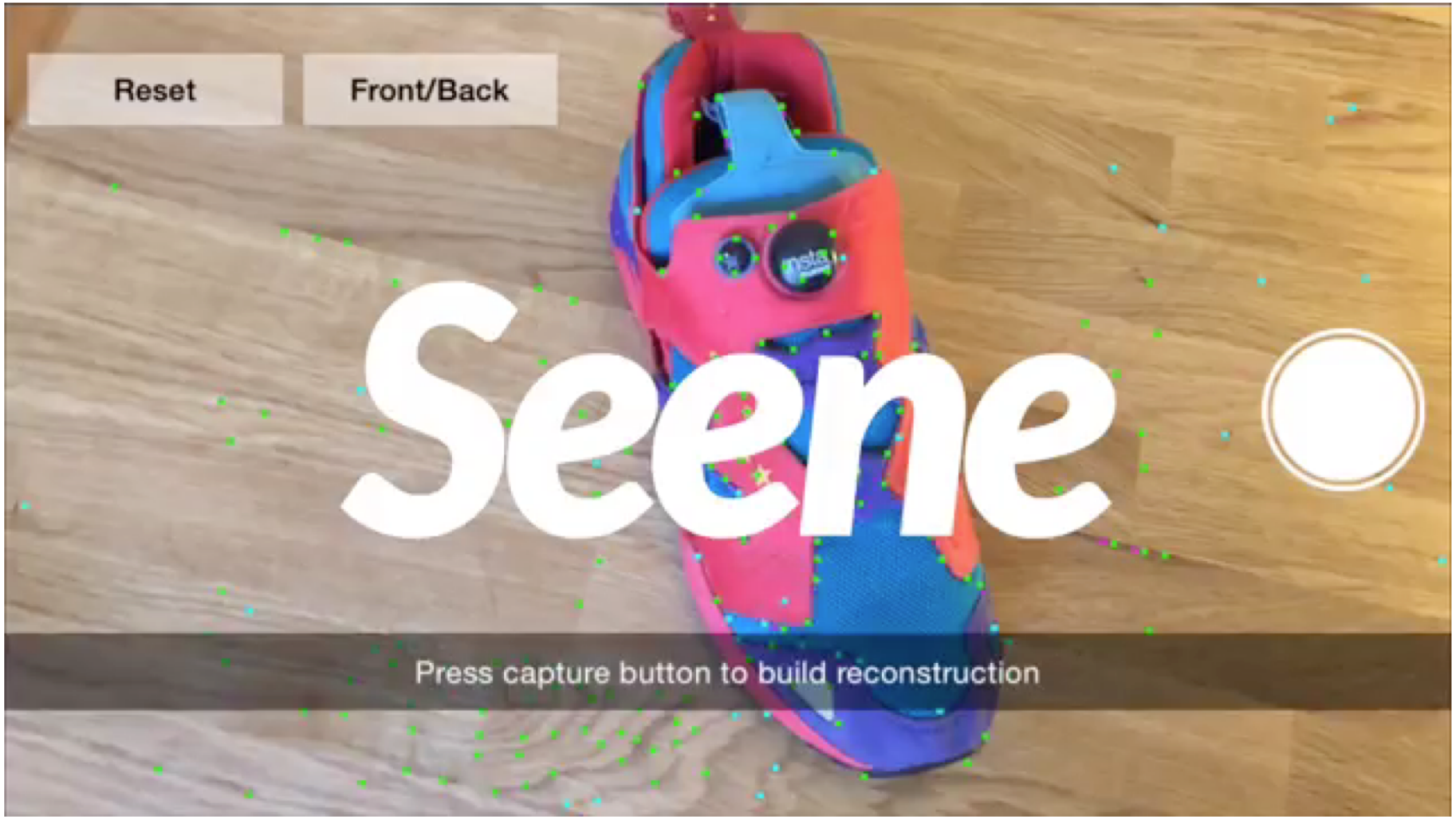
1. Camera geometry
2. Shape from X
3. Motion Estimation
4. Machine learning in computer vision

2. Shape from “X”

- An image is 2D. But most underlying objects are 3D.
- Can you guess something about the 3D structure of the underlying object just given the 2D image(s)?
- The human visual system does this all the time!
- We want to reproduce this effect computationally (the “holy grail” of computer vision)

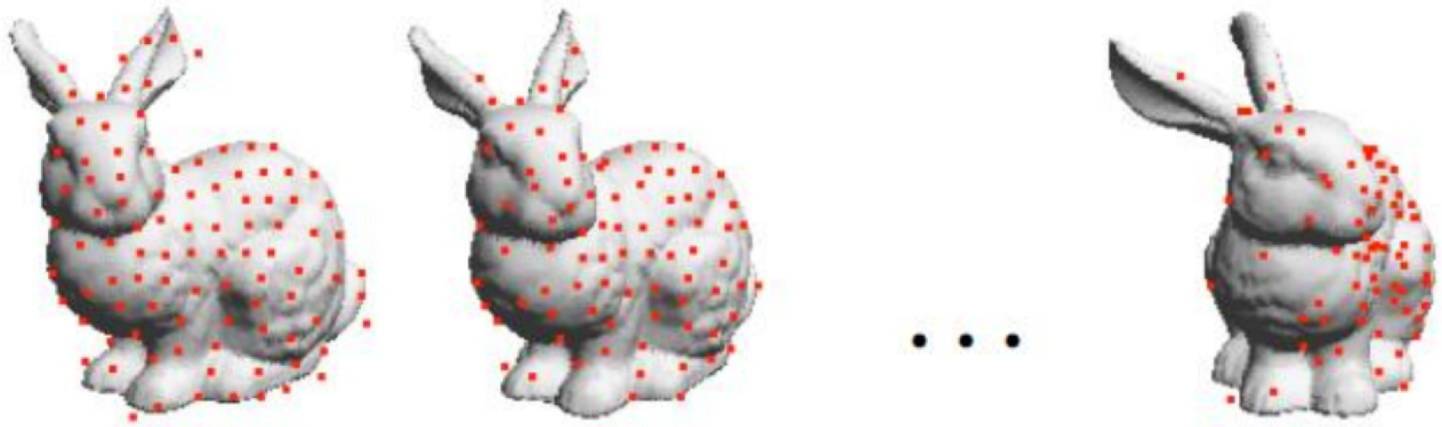
2. Shape from “X”

Structure from Motion



2. Shape from “X”

Structure from Motion



1. Input: Video sequence of moving (translating + rotating) object taken from a still camera
2. Solve: Tracks of some N 2D salient points from each frame of the video sequence (correspondence problem)
3. Outputs: 3D coordinates of each of those N points in each frame + 3D motion of the object!

2. Shape from “X”

Depth from Stereo and Disparity

http://3dstereophoto.blogspot.in/2015/06/

Left Image

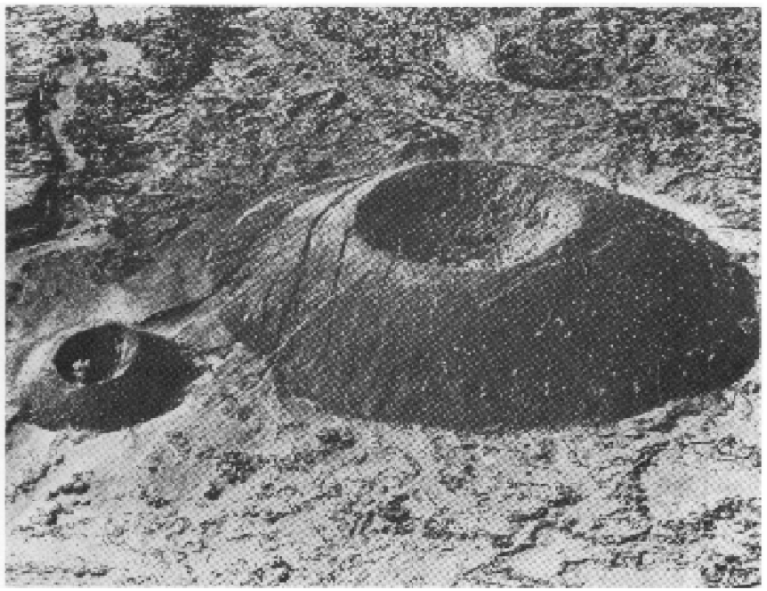
Right Image

Output Depth Map

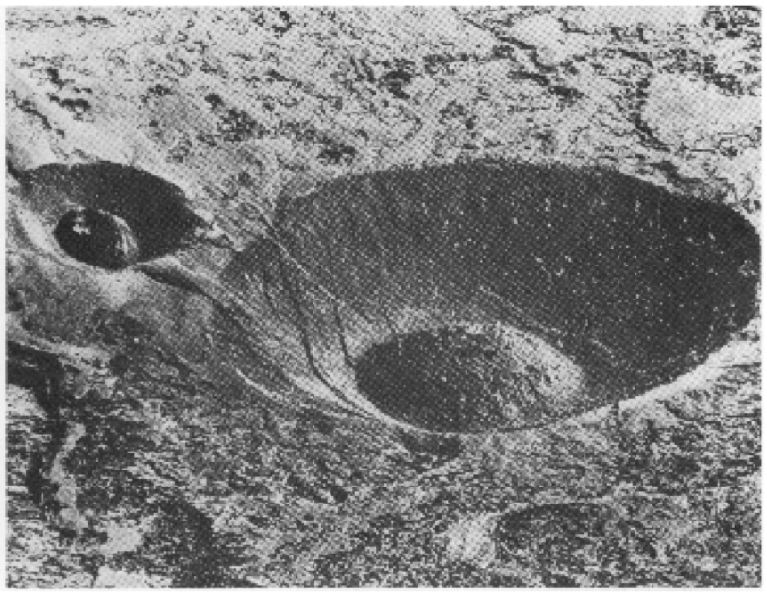
2. Shape from “X”

Shape from Shading

(a)



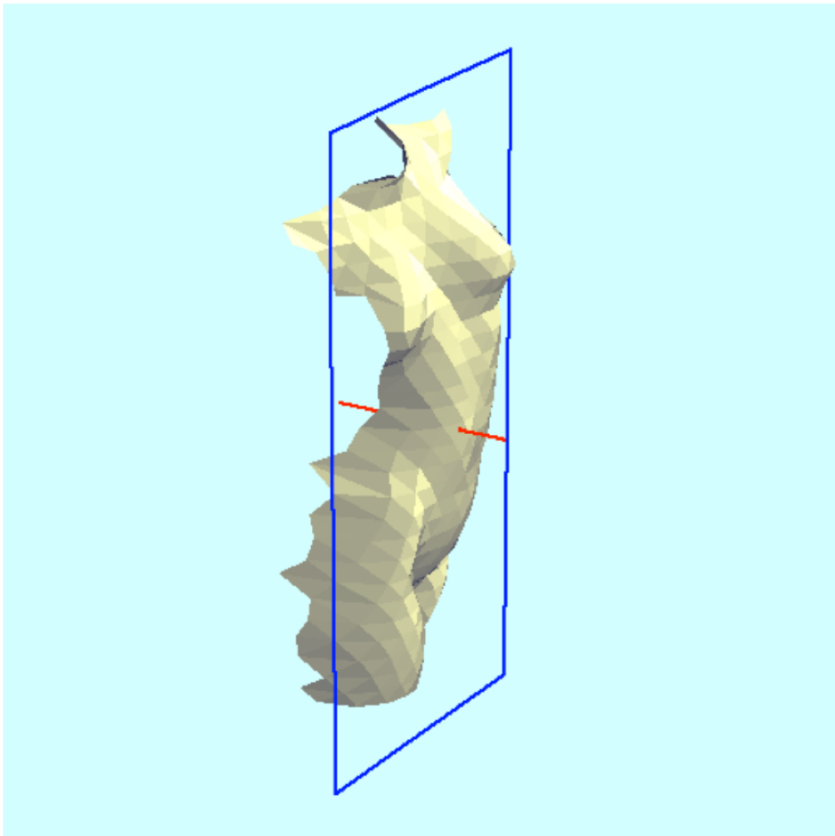
(b)



Shading influences shape. The image in (a) has the appearance of mound of dirt with a small indentation. The image in (b) appears to contain a crater with a mound at the top. Yet, the two images are the same except for an up-down flip

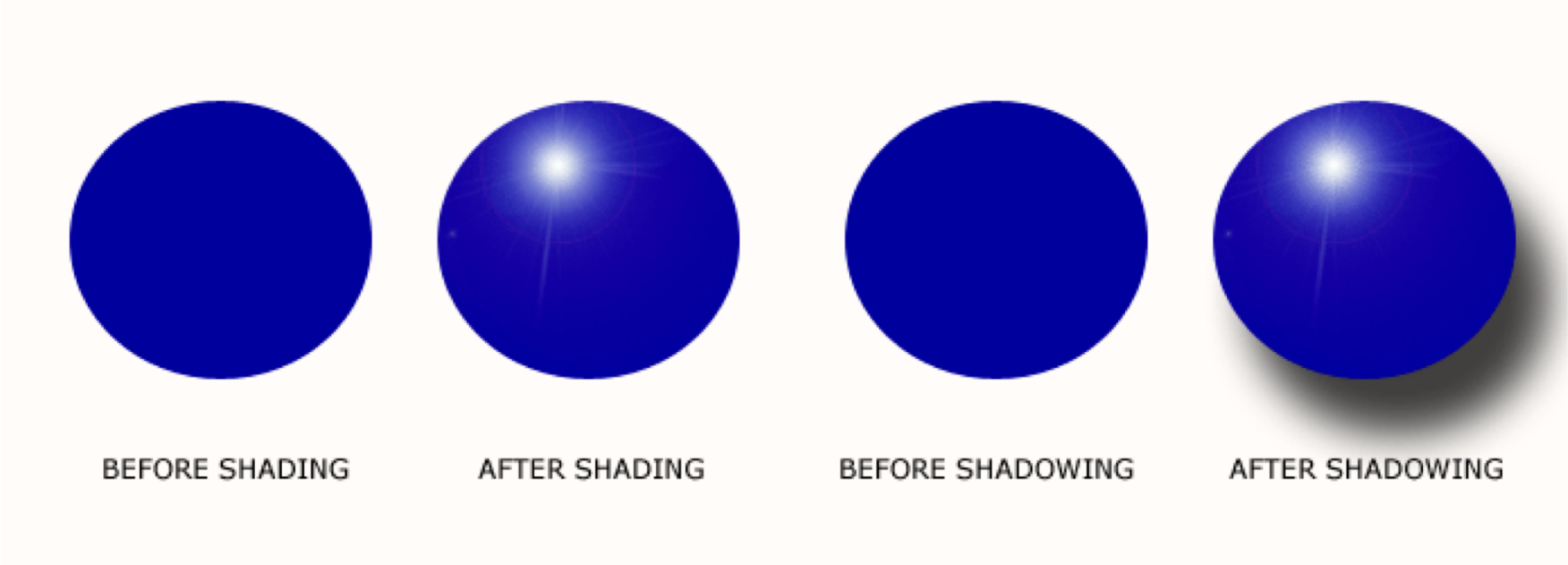
2. Shape from “X”

Shape from Shading



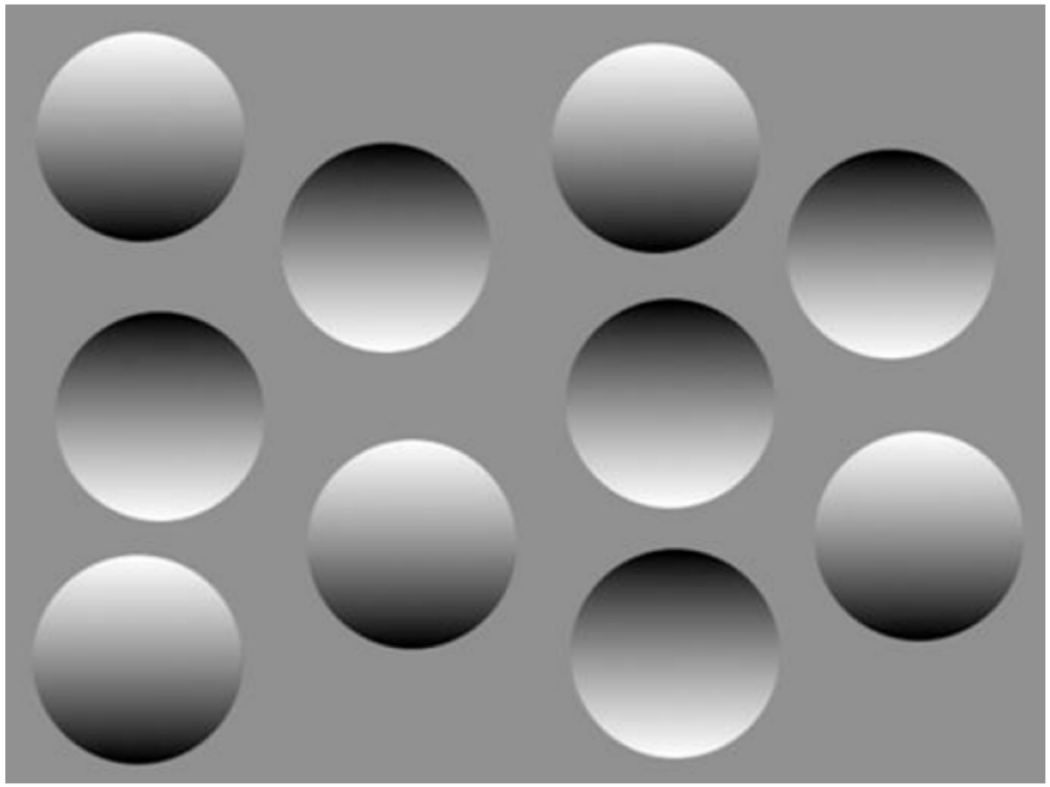
2. Shape from “X”

Shape from Shading



2. Shape from “X”

Shape from Shading



Crater vs mound?

A few topics in CV

1. Camera geometry
2. Shape from X
3. Motion Estimation
4. Machine learning in computer vision

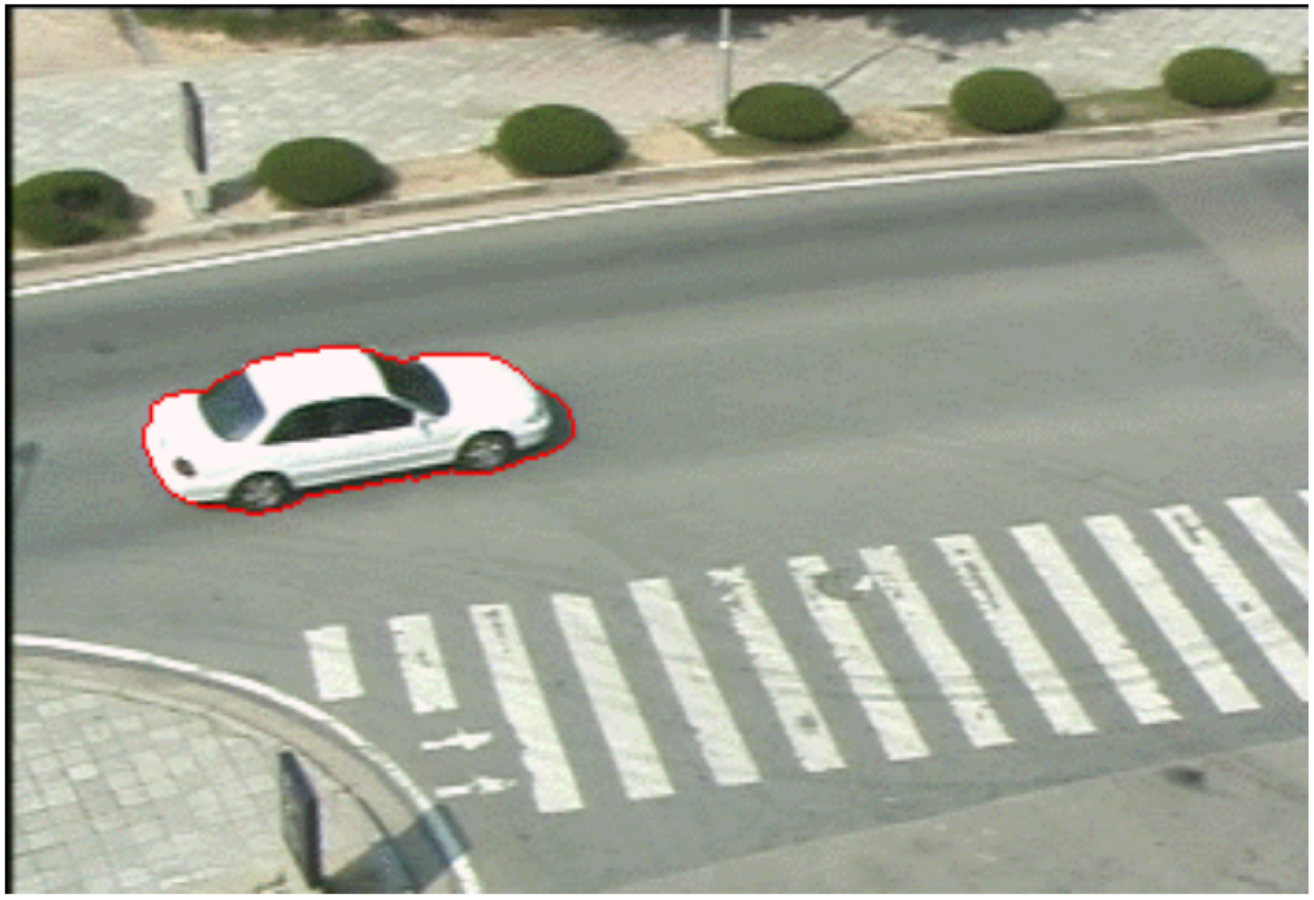
3. Motion Estimation

- A video sequence is very rich in information content
- Movement brings in most of this information
  - Movement allows objects identification
  - Image characteristics are coherent along motion trajectories
- Motion detection: binary decision (motion or no motion)
- Motion estimation: measure the movement

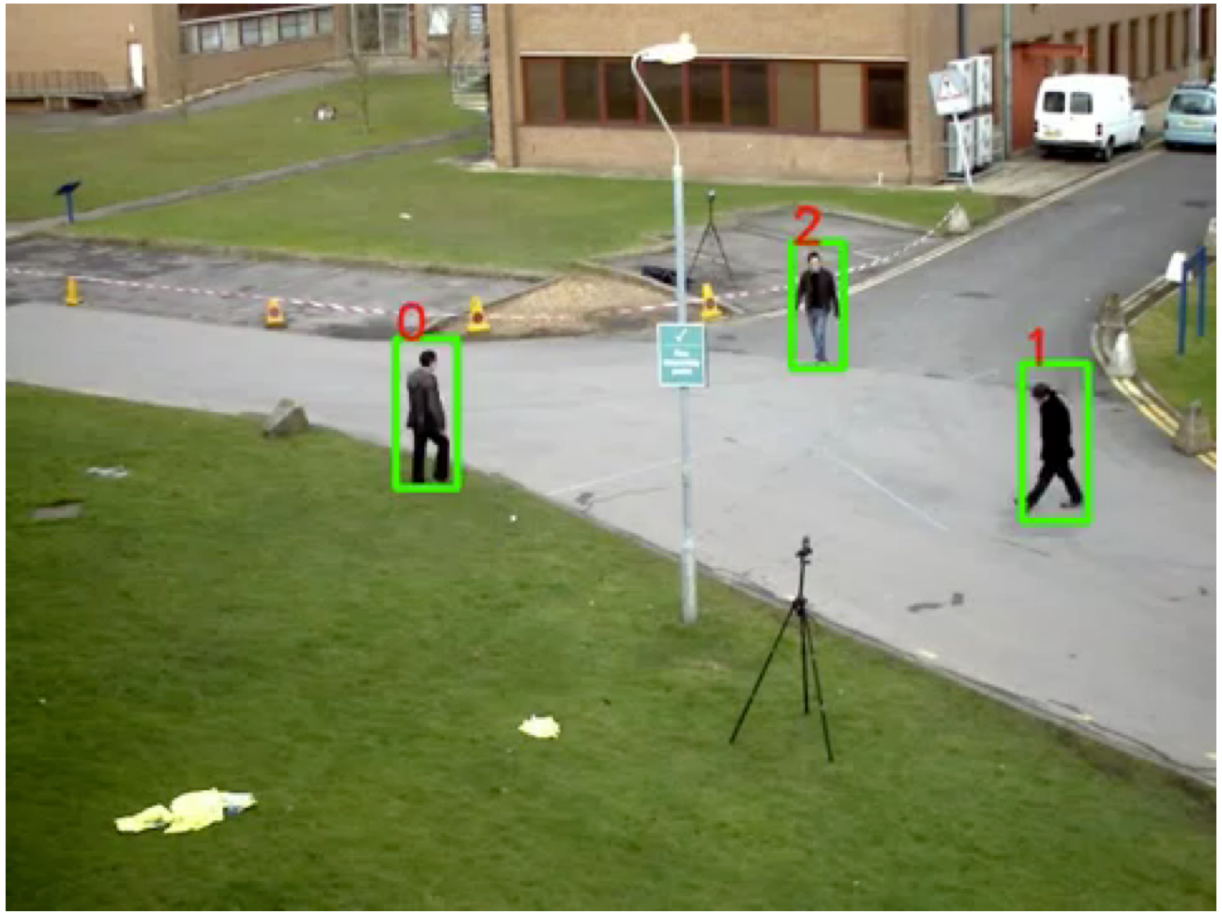
3. Motion Estimation

Other types of Motion Estimation:

1. Object Tracking

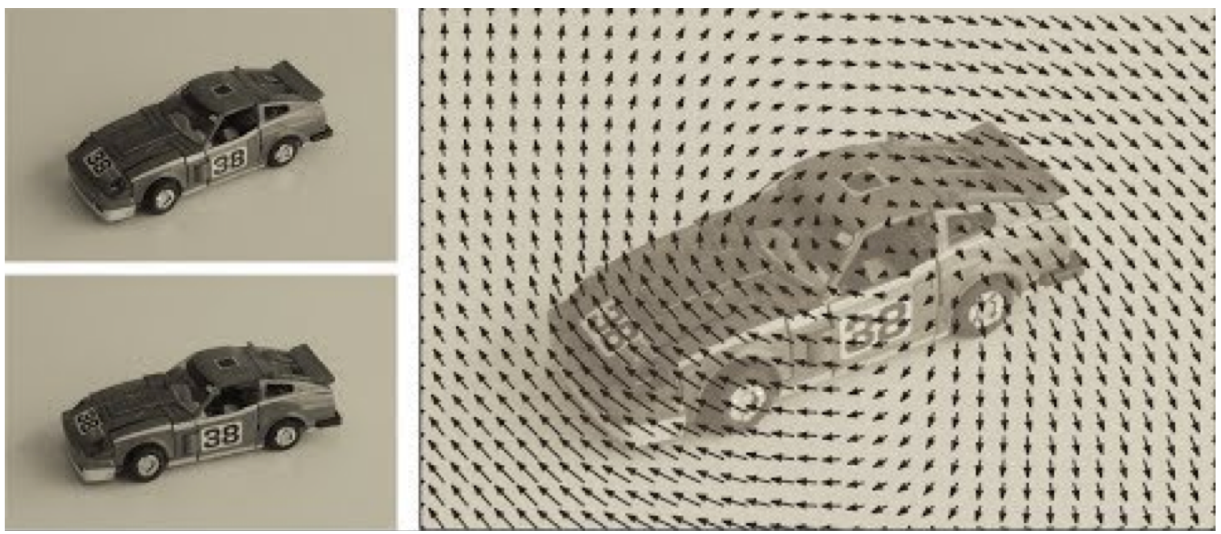


Other types of Motion Estimation:
2. Multiple Object Tracking



3. Motion Estimation

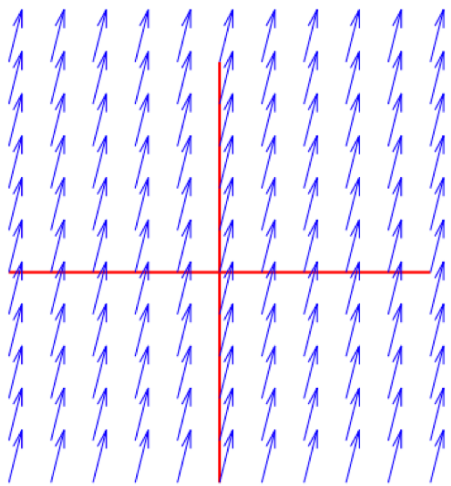
- **Input**: a video sequence
- **Target**: an estimate of the motion(2D) at all pixels in all frames
- Applications of such an algorithm: object tracking, video stabilization, etc.
- Typical assumptions: small motion between consecutive frames



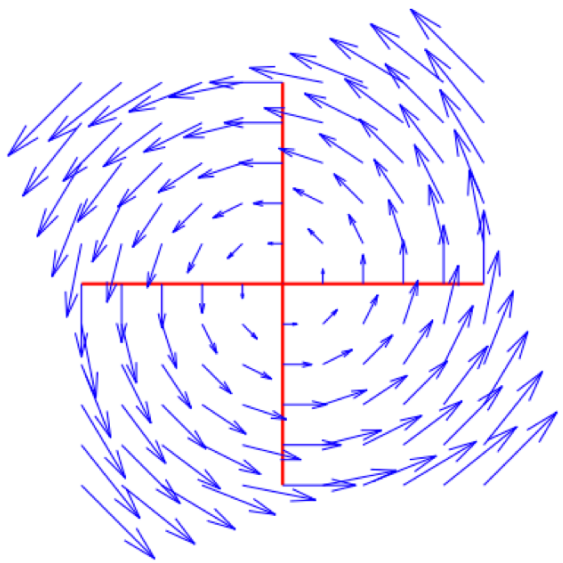
https://www.youtube.com/watch?v=KoMTYnlNNnc

3. Motion Estimation

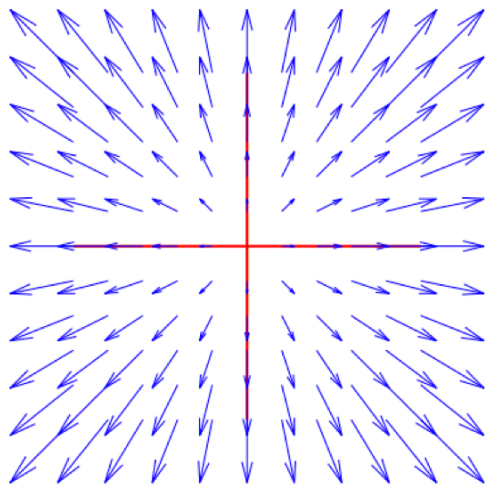
- Sometimes the motion between two images can be represented more compactly – e.g.: rotation, scaling, translation, etc.
- We will look at methods to estimate such “parametric motion”



Translation



Rotation



Scaling

A few topics in CV

1. Camera geometry
2. Shape from X
3. Motion Estimation
4. Machine learning in computer vision

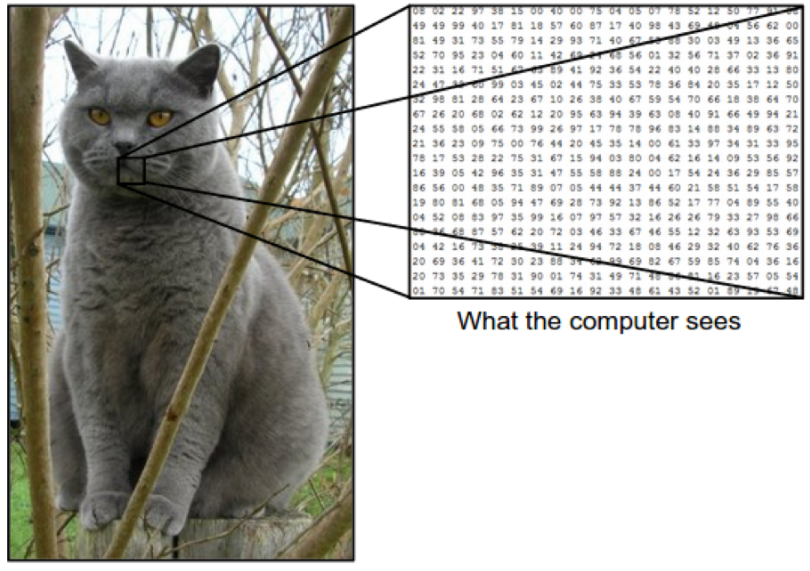
4. Machine Learning in Computer Vision

- Why do we need to do machine learning?

Images are represented as 3D arrays of numbers, with integers between [0, 255].

E.g.
300 x 100 x 3

(3 for 3 color channels RGB)



- Why do we need to do machine learning? **No way to hand code it!**

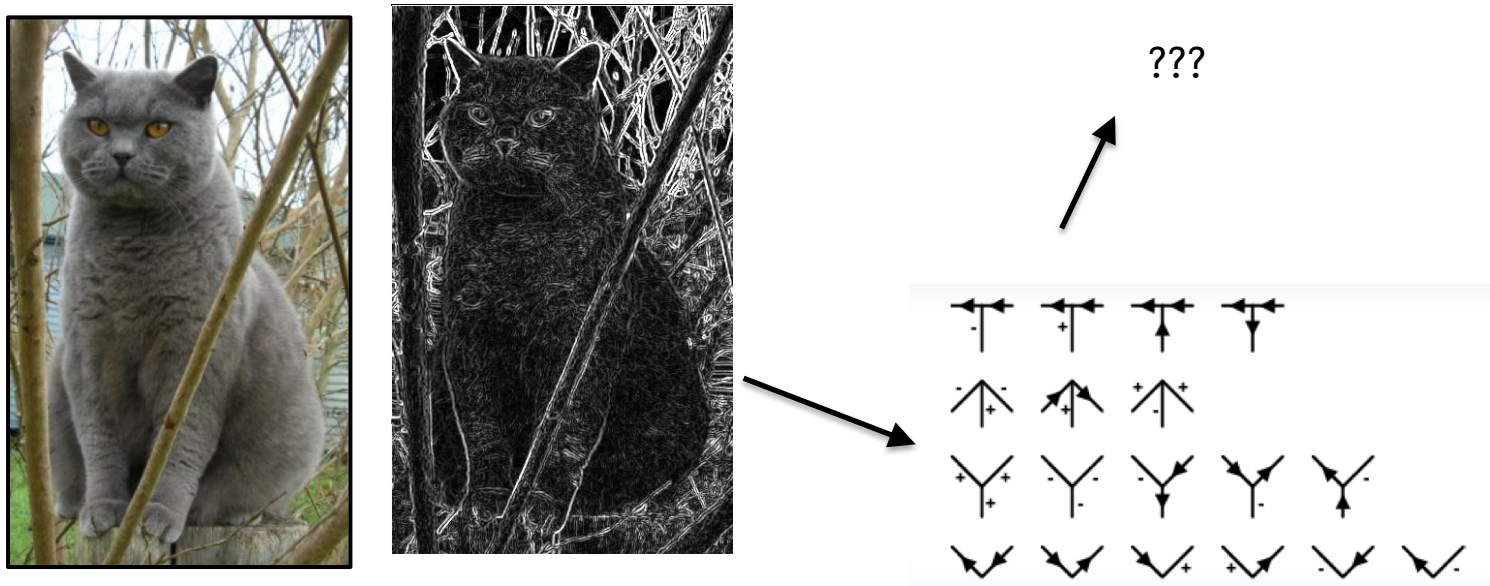
Image classification:

```
function predict(image)
  -- ????
  return class_label
end
```

- Unlike e.g. sorting a list of numbers
- No obvious way to hard-code the algorithm for recognizing a cat, or other classes

- People have attempted

- Image classification:

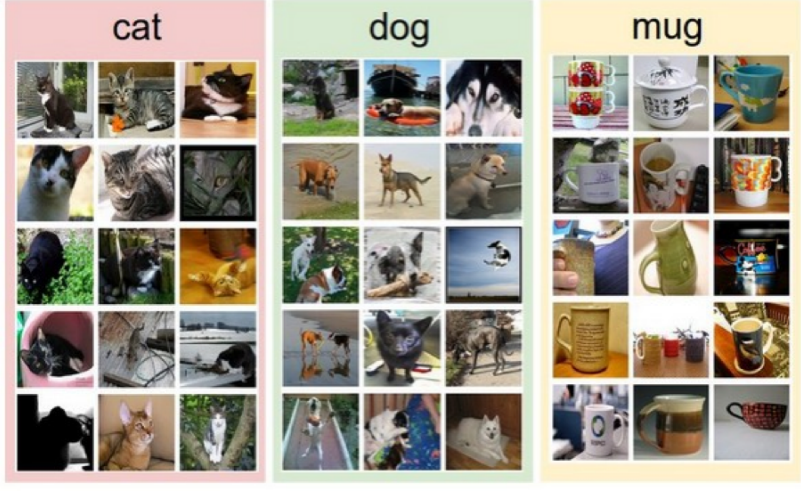


The Data Driven Paradigm

1. Collect a dataset of images and labels
2. Use Machine Learning to train an image classifier
3. Evaluate the classifier on a withheld set of test images

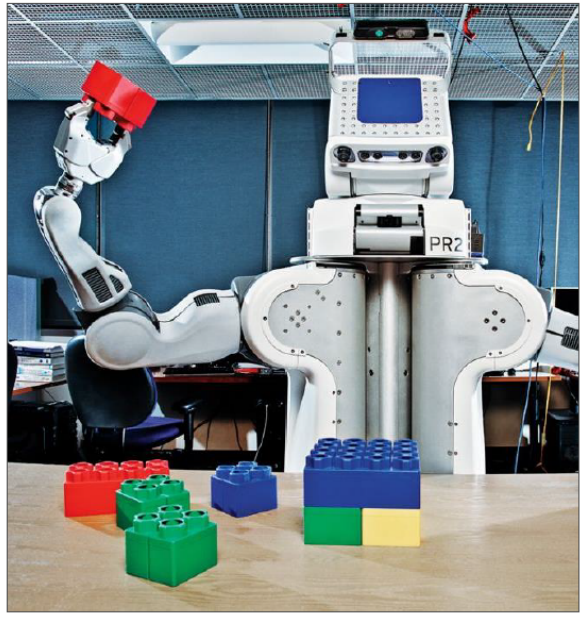
```
function train(train_images, train_labels)
  -- Build model: images -> labels
  return model
end


function predict(model, test_images)
  -- Predict test_labels using the model
  return test_labels
end
```

Example Training Set



4. Machine Learning in Computer Vision (Deep Learning)

- Deep Learning == AI



4. Machine Learning in Computer Vision (Deep Learning)

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

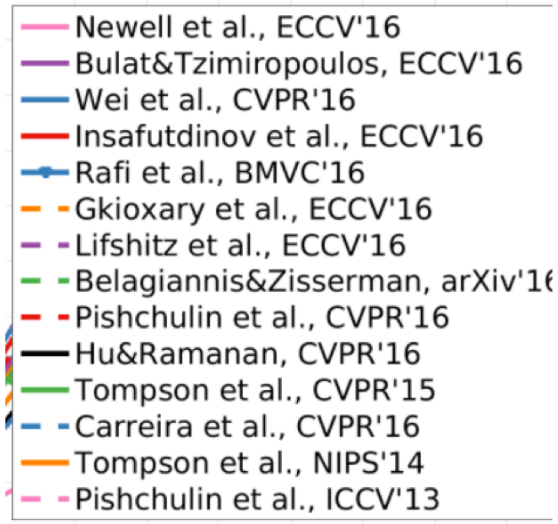
"two young girls are playing with lego toy."

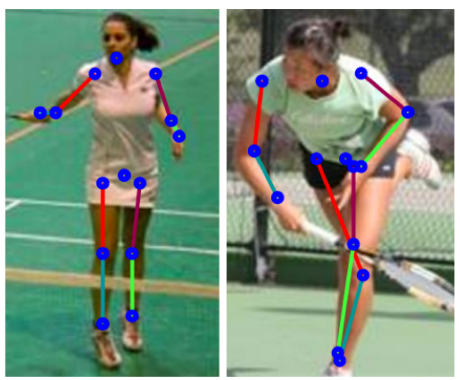
"girl in pink dress is jumping in air."

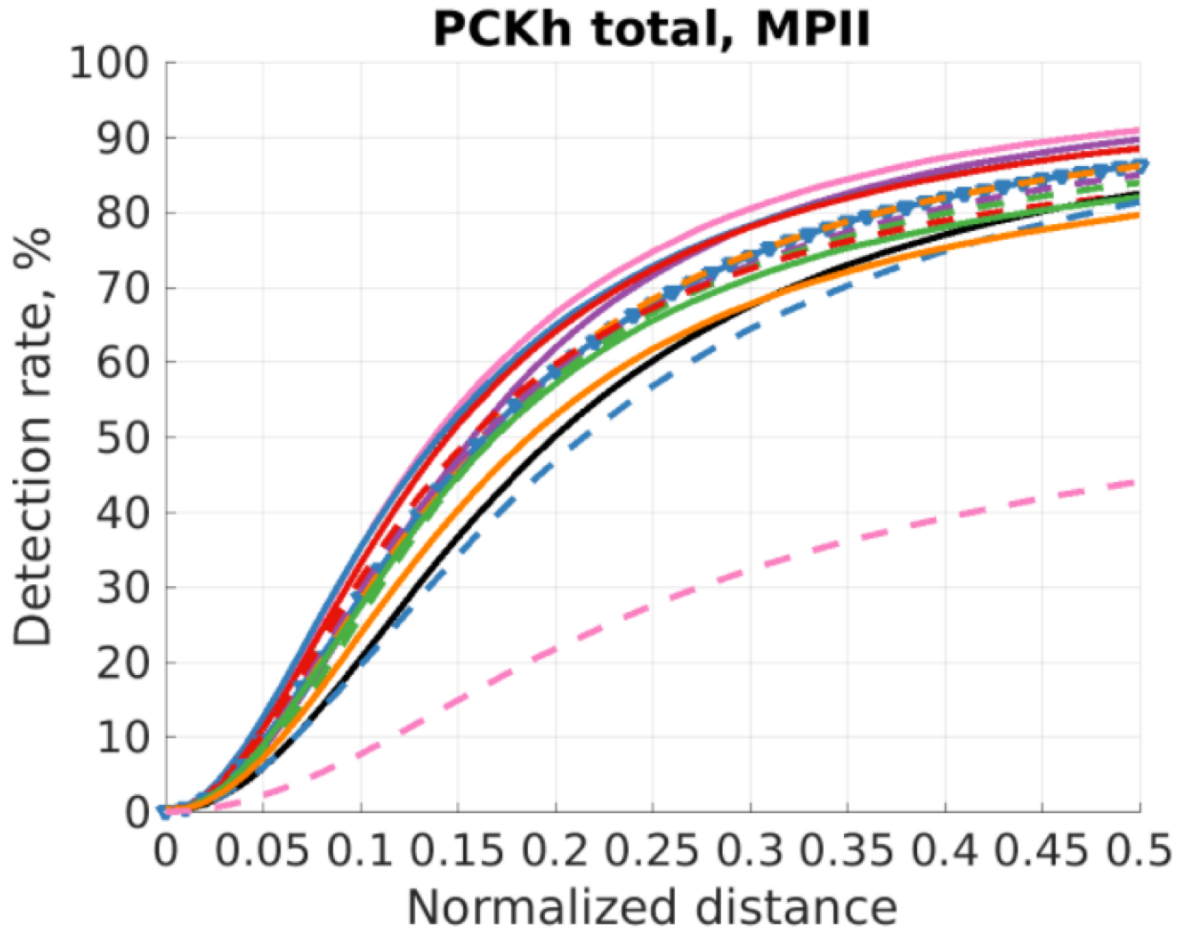
"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

4. Machine Learning in Computer Vision (Deep Learning)

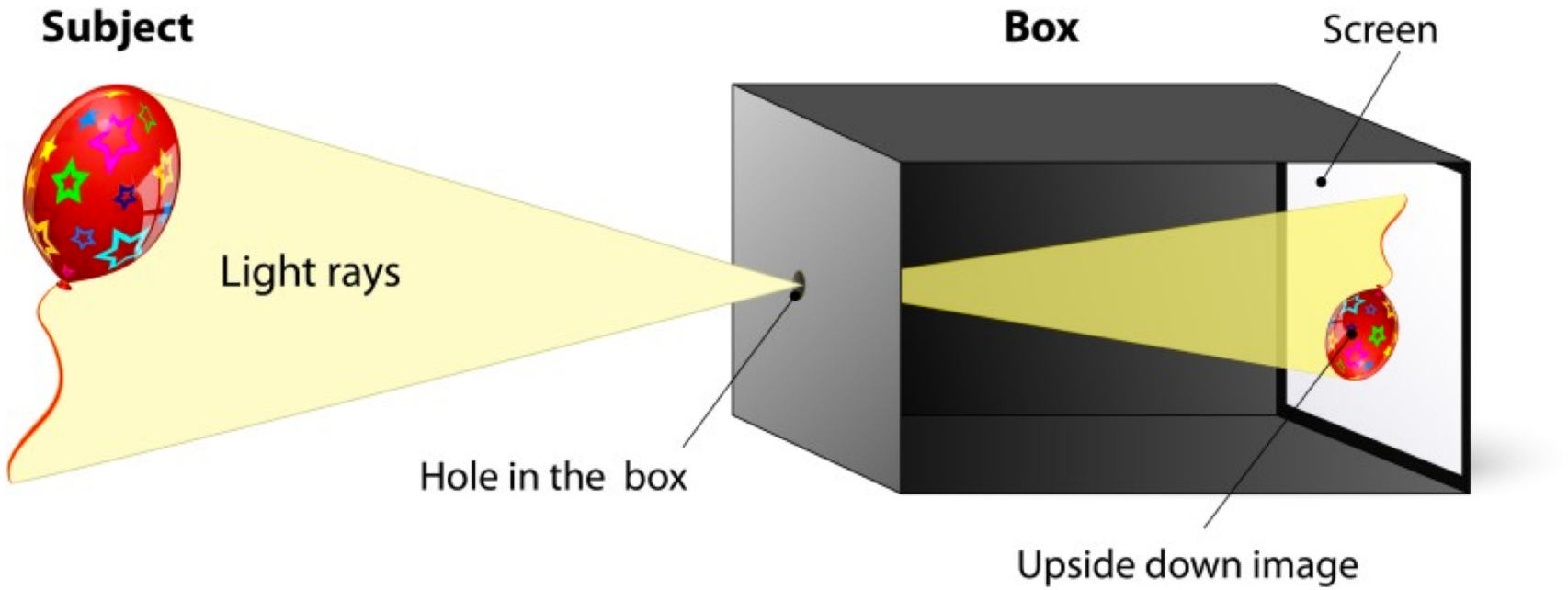






Digital Image Processing

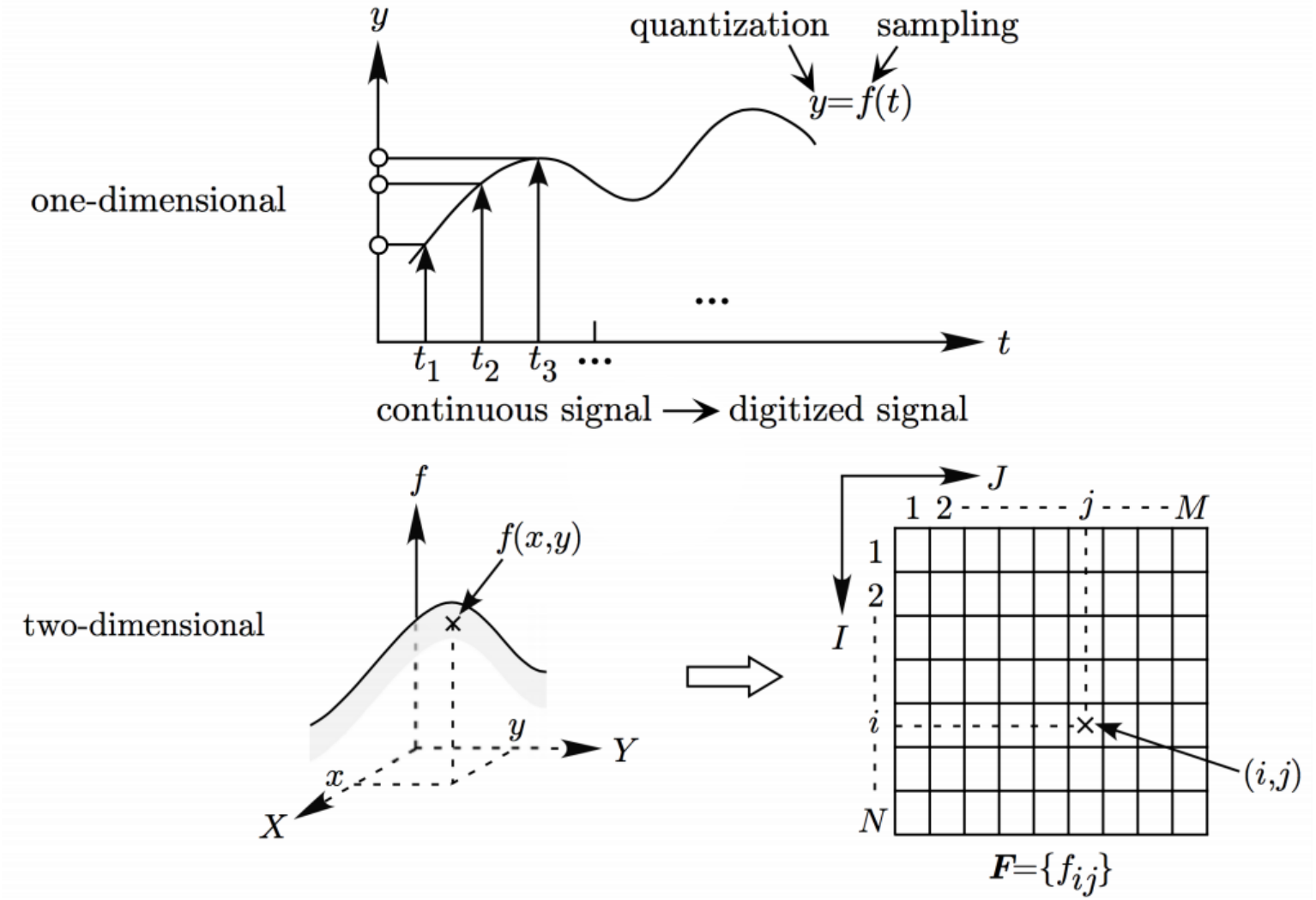
Pin Hole Camera Model



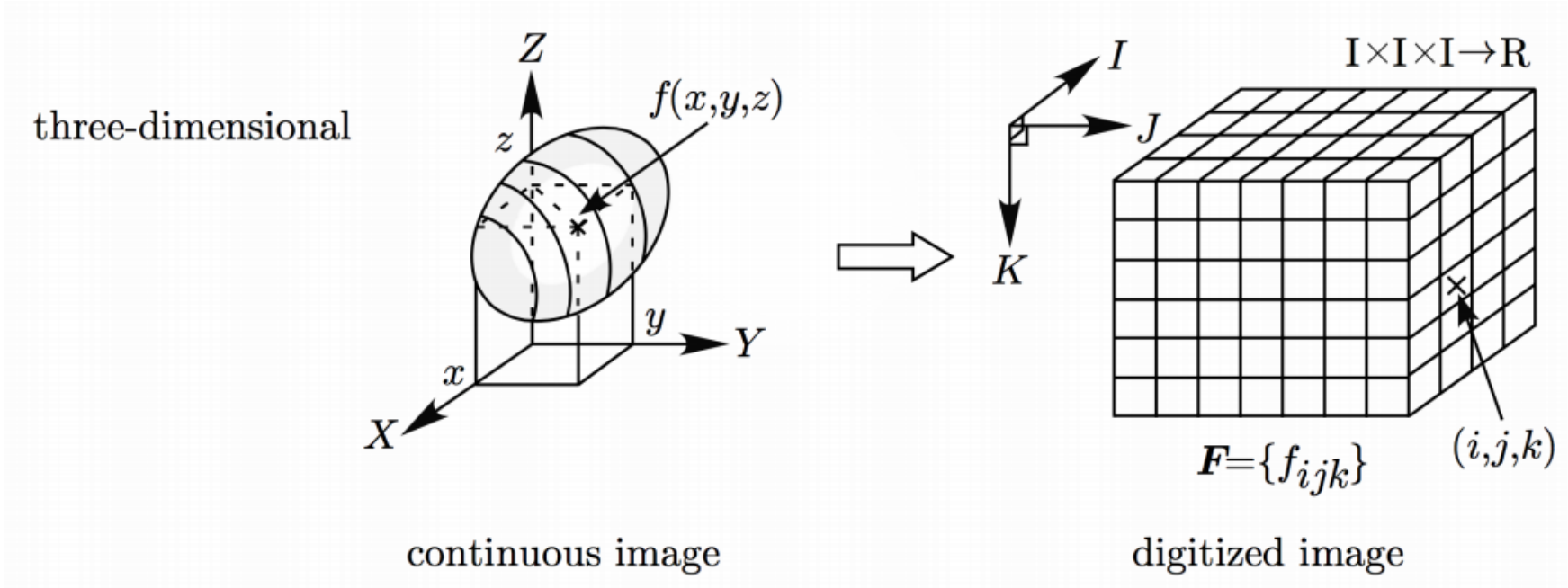
Digital Pictures

- **DIGITAL IMAGES** are electronic snapshots taken of a scene or scanned from documents, such as photographs, manuscripts, printed texts, and artwork.
- The digital image is sampled and mapped as a grid of dots or picture elements (pixels).
- The process transforming continuous space into discrete space is called digitization

1D/2D Digitization / Sampling

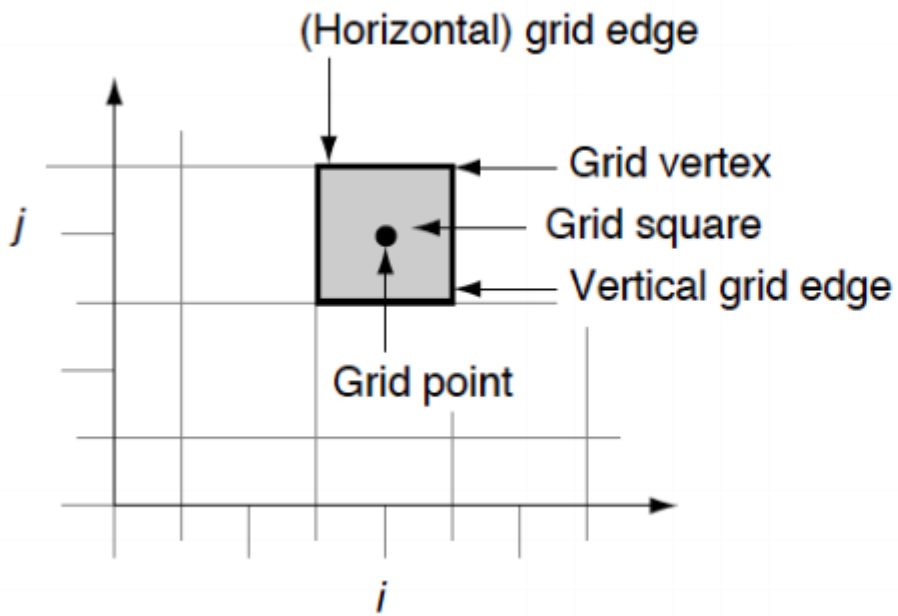


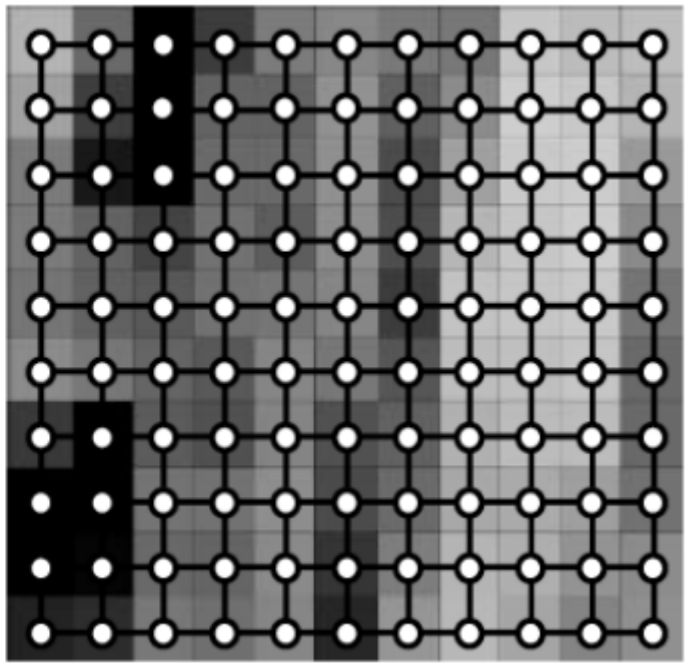
3D Digitization / Sampling



Definition

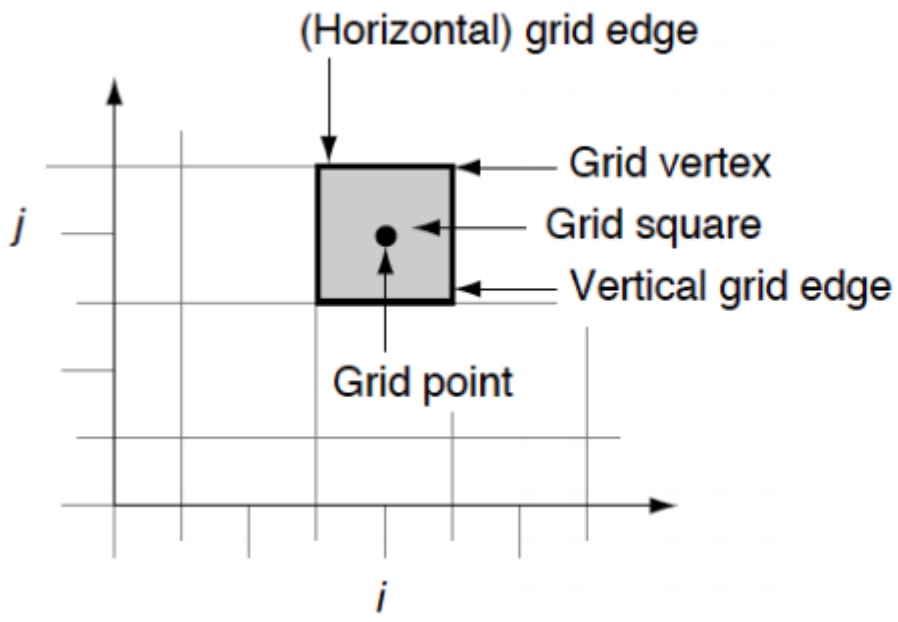
- A (2D) picture P is a function defined on a (finite) rectangular subset G of a regular planar orthogonal array. G is called (2D) grid, and an element of G is called pixel. P assigns a value of $P(p)$ to each $p \in G$

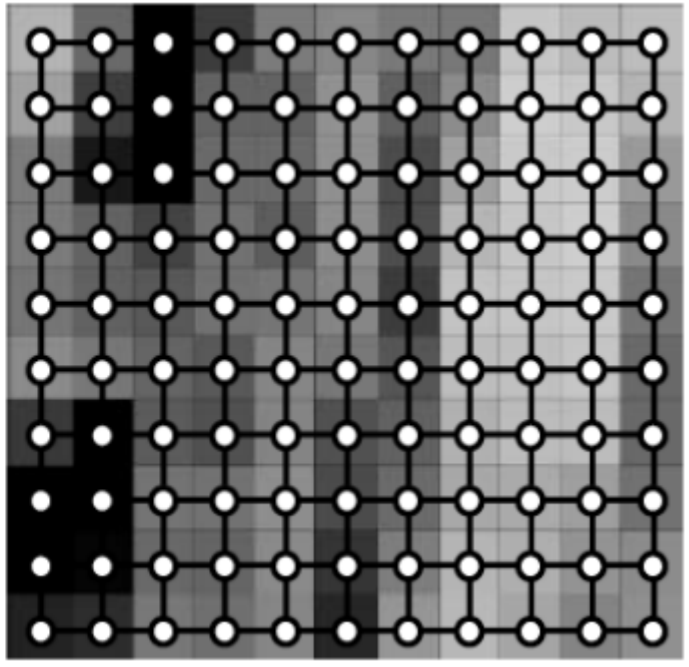




Definition

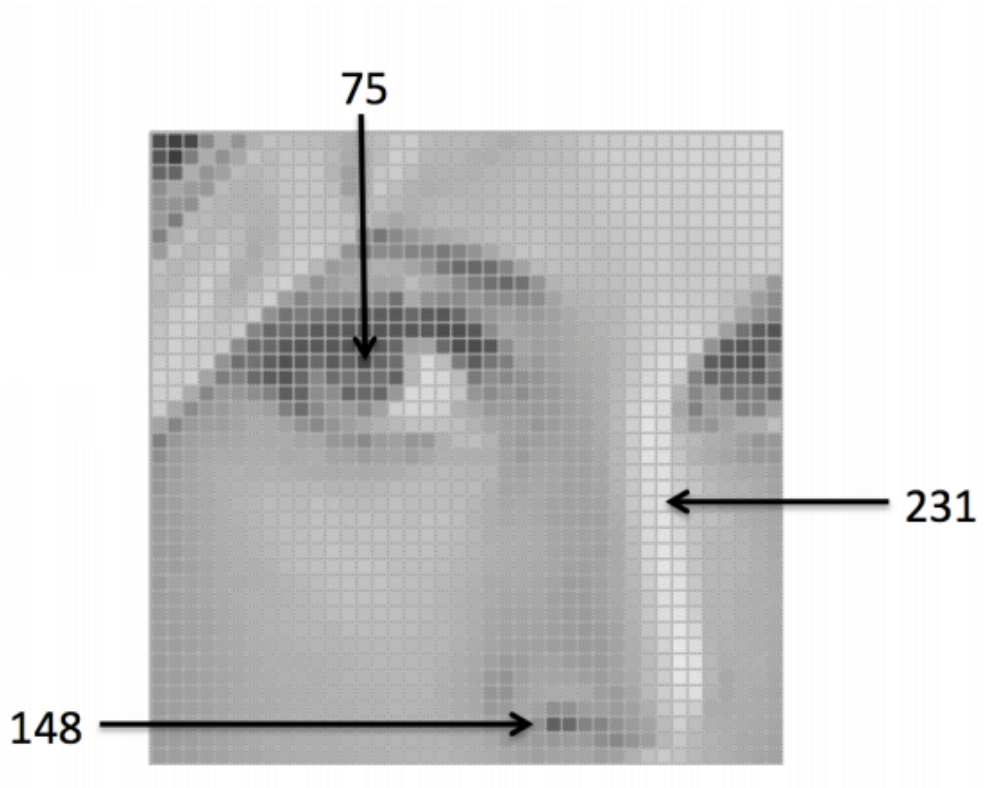
- Pictures are not only sampled, they are also quantized: they may have only a finite number of possible values (i.e., 0 to 255, 0-1, ...)





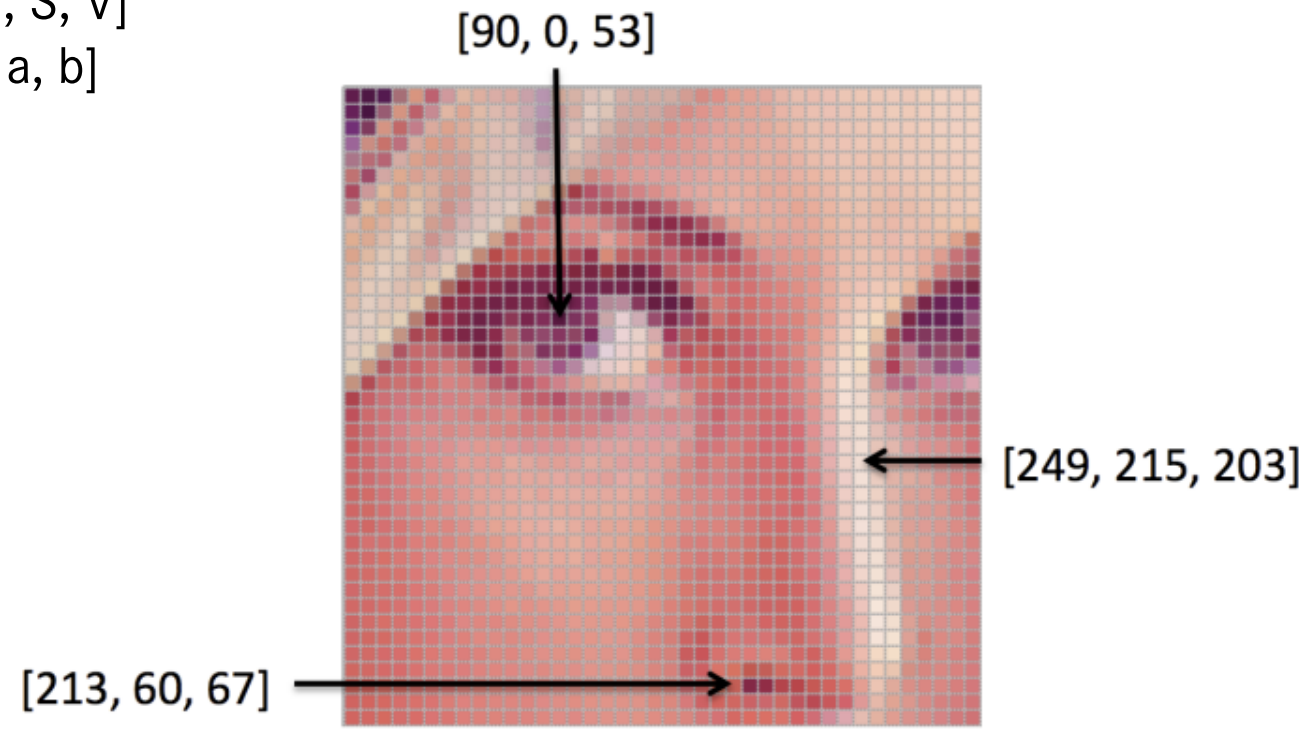
Example

- An image contains discrete number of pixels.
- In this example we have a “grayscale image”, with intensity values in [0,255]



Example

- An image contains discrete number of pixels.
- In this example we have a “color image”
  - RGB [R,G,B]
  - HSV [H, S, V]
  - Lab [L, a, b]



Example: RGB Channels of a Color Image

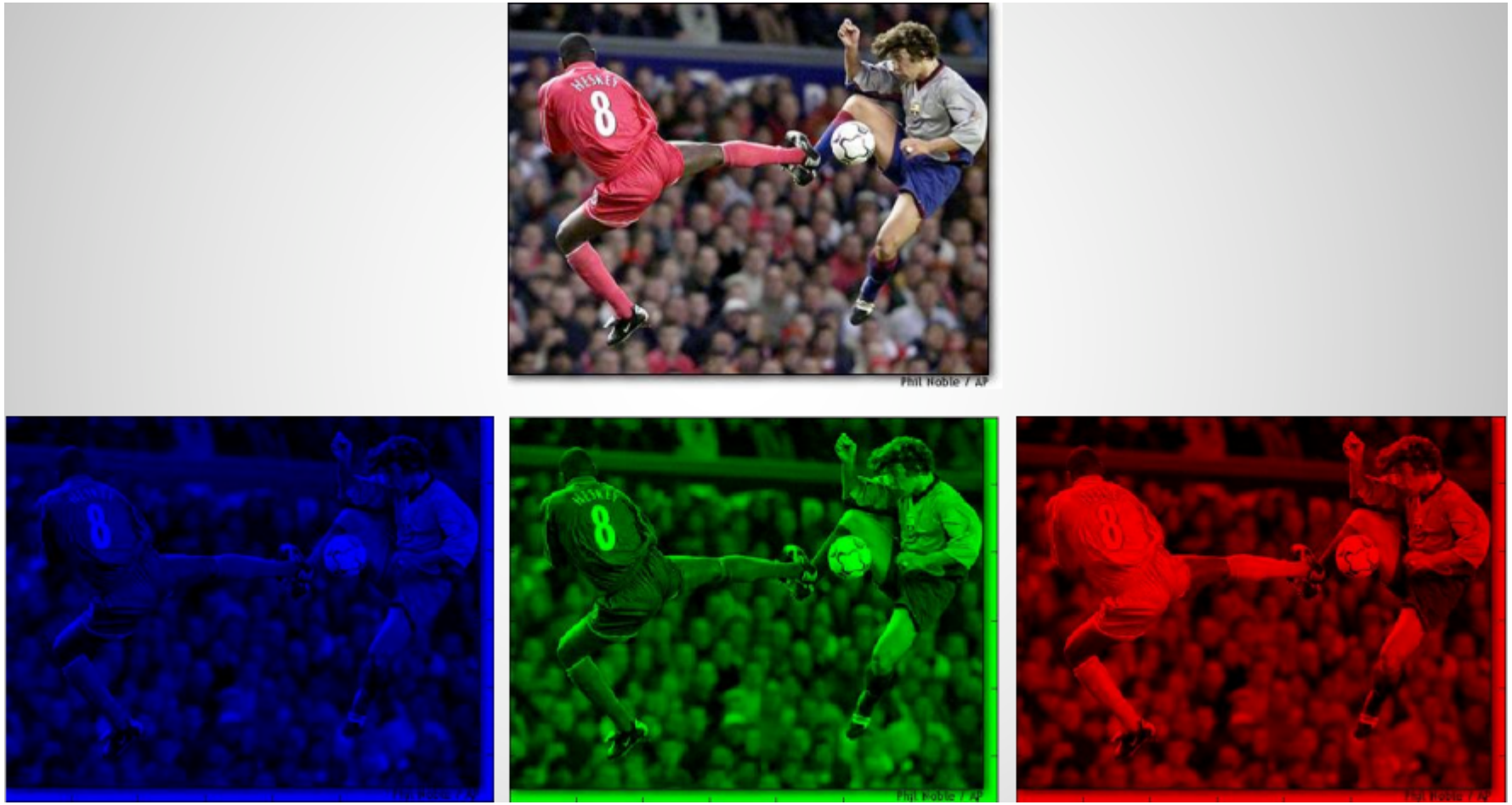


Image Derivatives and Filtering

Derivative

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x) = f_x$$

$v = \frac{ds}{dt}$ speed

$a = \frac{dv}{dt}$ acceleration

Discrete Derivative

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x)$$

$$\frac{df}{dx} = \frac{f(x) - f(x - 1)}{1} = f'(x)$$

$$\frac{df}{dx} = f(x) - f(x - 1) = f'(x)$$

Discrete Derivative
Finite Difference

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

Backward difference

$$\frac{df}{dx} = f(x) - f(x+1) = f'(x)$$

Forward difference

$$\frac{df}{dx} = \frac{f(x+1) - f(x-1)}{2} = f'(x)$$

Central difference

Example

$f(x) =$	10	15	10	10	25	20	20	20
$f'(x) =$	0	5	−5	0	15	−5	0	0
$f''(x) =$	0	5	−10	5	15	20	5	0

Derivative Masks

- Backward difference [-1 1]
- Forward difference [1 -1]
- Central difference [-0.5 0 0.5]

Derivatives in 2 Dimentions

Given function $f(x, y)$

Gradient vector $$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Gradient magnitude $$\left|\nabla f(x, y)\right| = \sqrt{f_x^2 + f_y^2}$$

Gradient direction $$\theta = \tan^{-1} \frac{f_x}{f_y}$$

Derivatives of Images

Derivative masl $f_x \Rightarrow \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ $f_y \Rightarrow \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Derivatives of Images

Derivative masl $f_x \Rightarrow \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ $f_y \Rightarrow \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Correlation

$$f \otimes h = \sum_k \sum_l f(k,l)h(i+k, j+l)$$

f = Image

f = Kernel

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$\otimes$

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$$f * h = f_1h_1 + f_2h_2 + f_3h_3 + f_4h_4 + f_5h_5 + f_6h_6 + f_7h_7 + f_8h_8 + f_9h_9$$

Convolution

$$f * h = \sum_k \sum_l f(k,l)h(i-k, j-l)$$

f = Image

h = Kernel

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$X - flip$

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

$Y - flip$

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

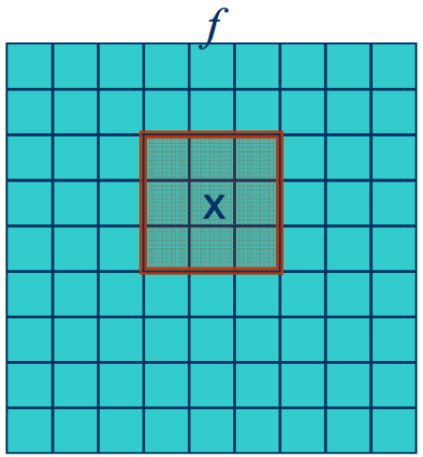
$\otimes$

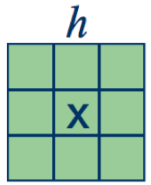
h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

$$f * h = f_1h_9 + f_2h_8 + f_3h_7 + f_4h_6 + f_5h_5 + f_6h_4 + f_7h_3 + f_8h_2 + f_9h_1$$

Convolution

$$f(x,y)*h = f(x+1,y+1)h(-1,-1) + f(x,y+1)h(0,-1) + f(x-1,y+1)h(1,-1) + f(x+1,y)h(-1,0) + f(x,y)h(0,0) + f(x-1,y)h(1,0) + f(x+1,y-1)h(-1,1) + f(x,y-1)h(0,1) + f(x-1,y-1)h(1,1)$$





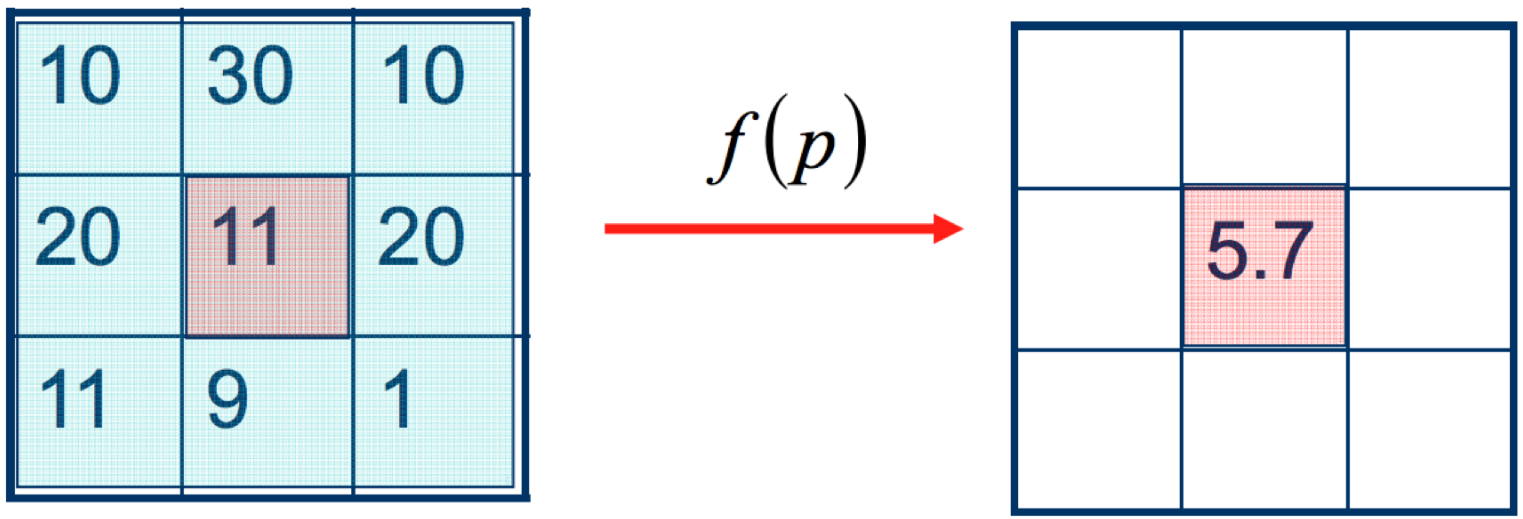
$$f*h = \sum_{i=-1}^{1}\sum_{j=-1}^{1} f(x-i,y-i)h(i,j)$$

Coordinates

-1,0	0,1	1,1
-1,0	0,0	1,0
-1,-1	0,-1	1,-1

Filtering

Modify pixels based on some function of the neighborhood



Linear Filtering

The output is the linear combination of the neighborhood pixels

1	3	0
2	10	2
4	1	1

Image

$\otimes$

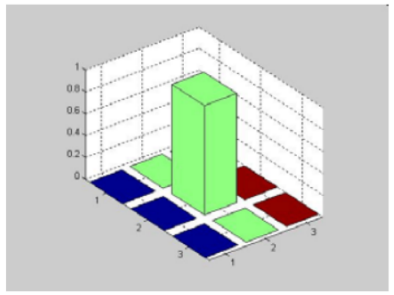
1	0	-1
1	0.1	-1
1	0	-1

Kernel

$=$

Filtering Examples

*

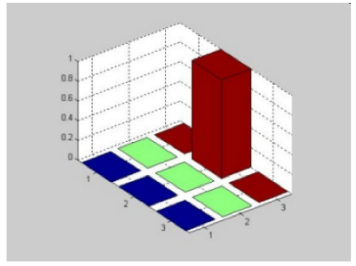


0	0	0
0	1	0
0	0	0

=

Filtering Examples

*



0	0	0
0	0	1
0	0	0

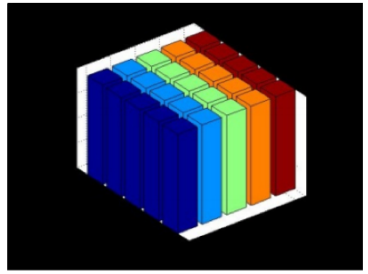
=

Filtering Examples

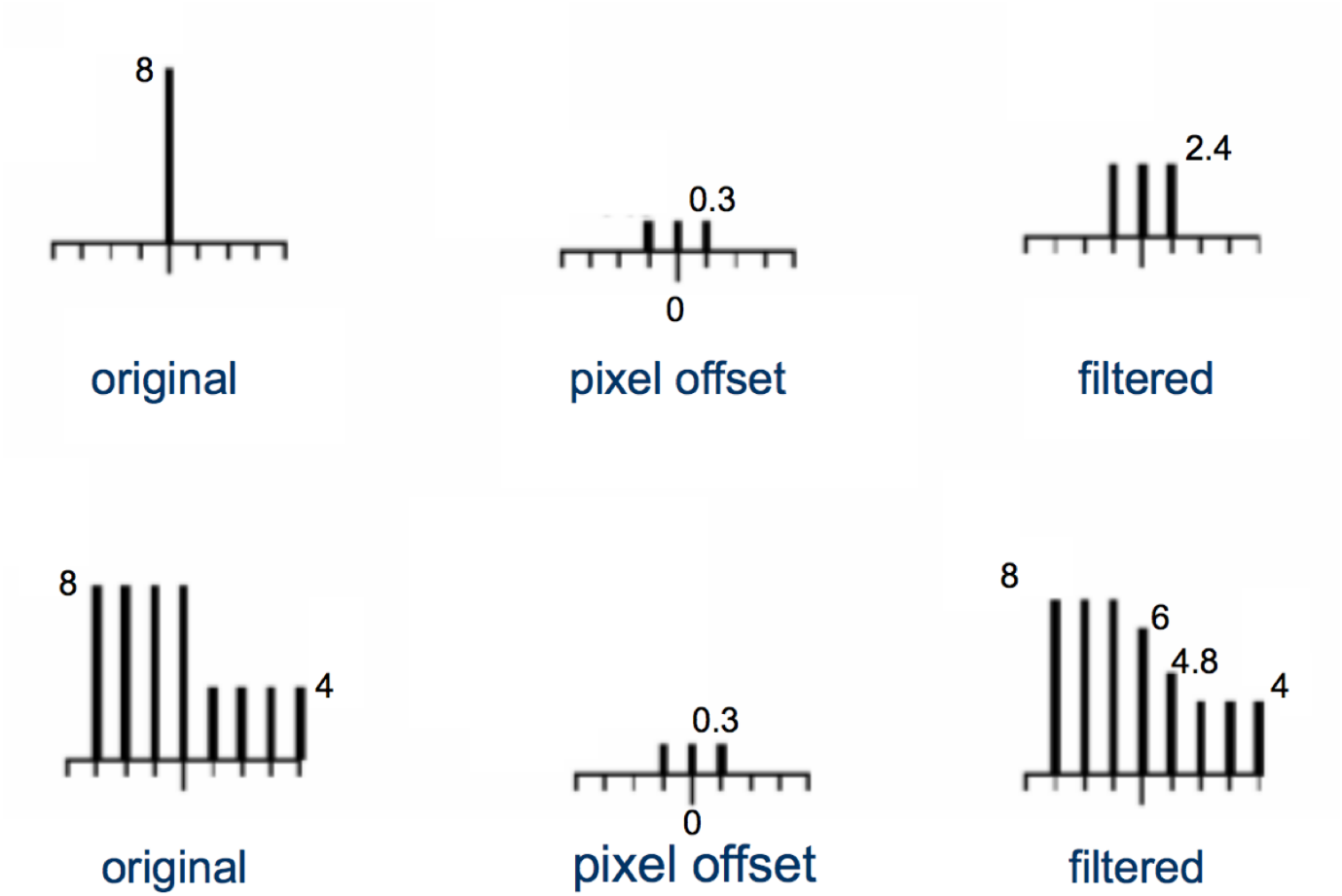
$$* \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

Filtering Examples

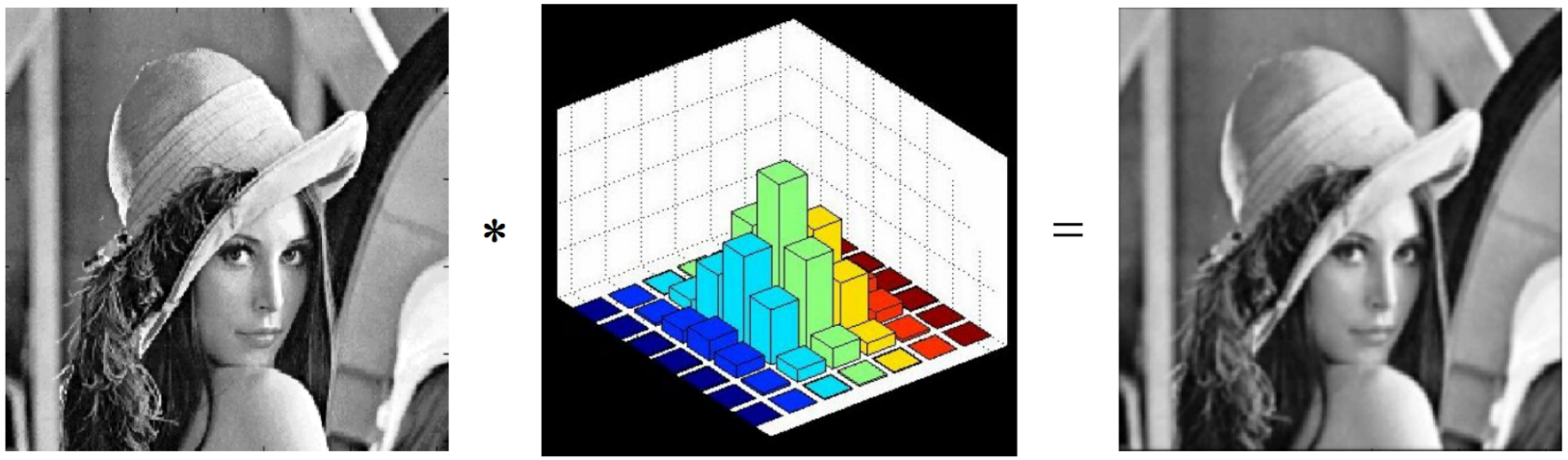
$$* \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$



Blurring Examples



Filtering Gaussian



Gaussian Filter vs. Smoothing

Gaussian Smoothing

Smoothing by Averaging

Noise Filtering

Gaussian Noise

After Averaging

After Gaussian Smoothing

Convolutional Neural Networks

A bit of history:

Hubel & Wiesel,

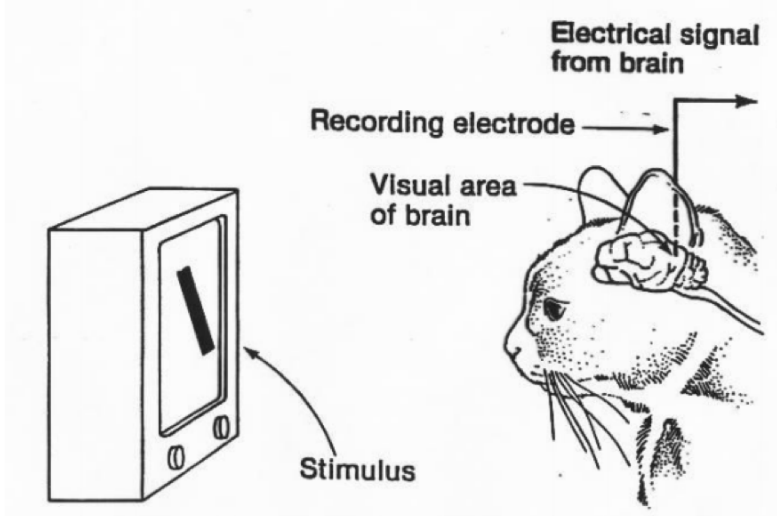
1959

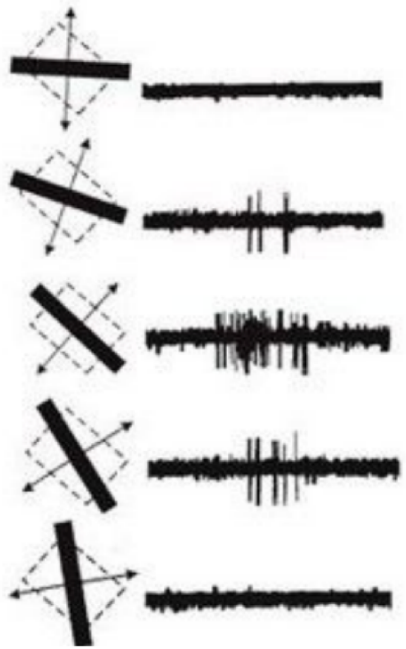
RECEPTIVE FIELDS OF SINGLE NEURONES IN
THE CAT'S STRIATE CORTEX

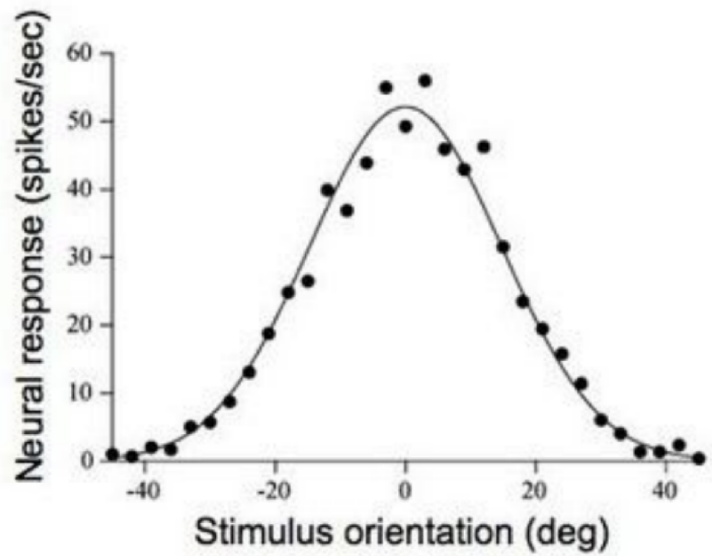
1962

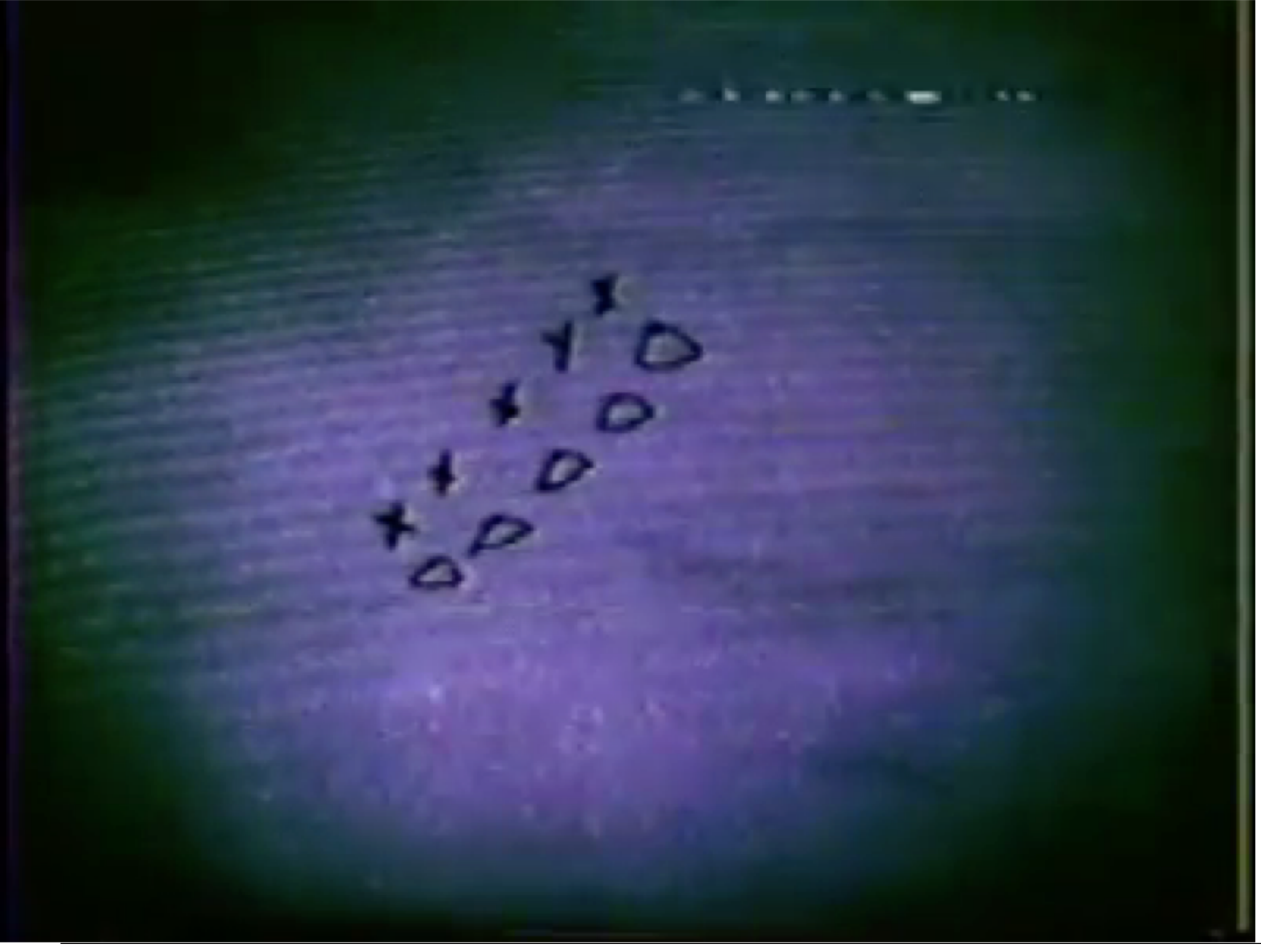
RECEPTIVE FIELDS, BINOCULAR INTERACTION
AND FUNCTIONAL ARCHITECTURE IN
THE CAT'S VISUAL CORTEX

1968...





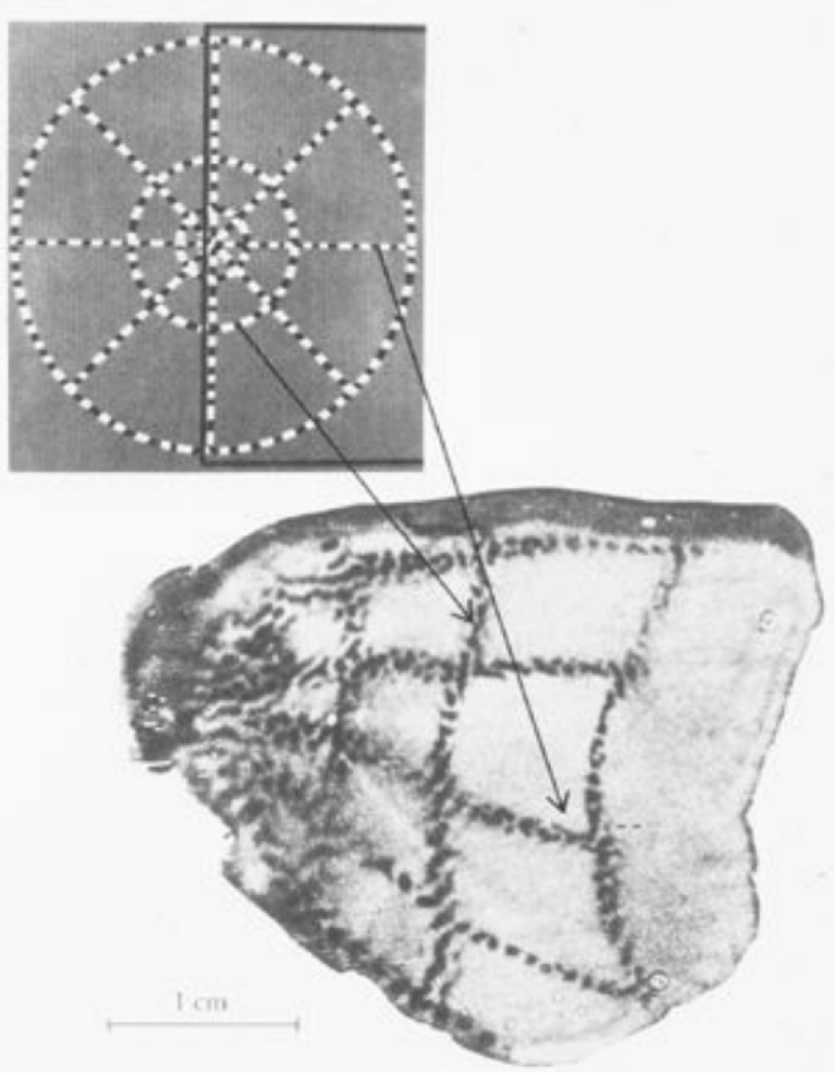




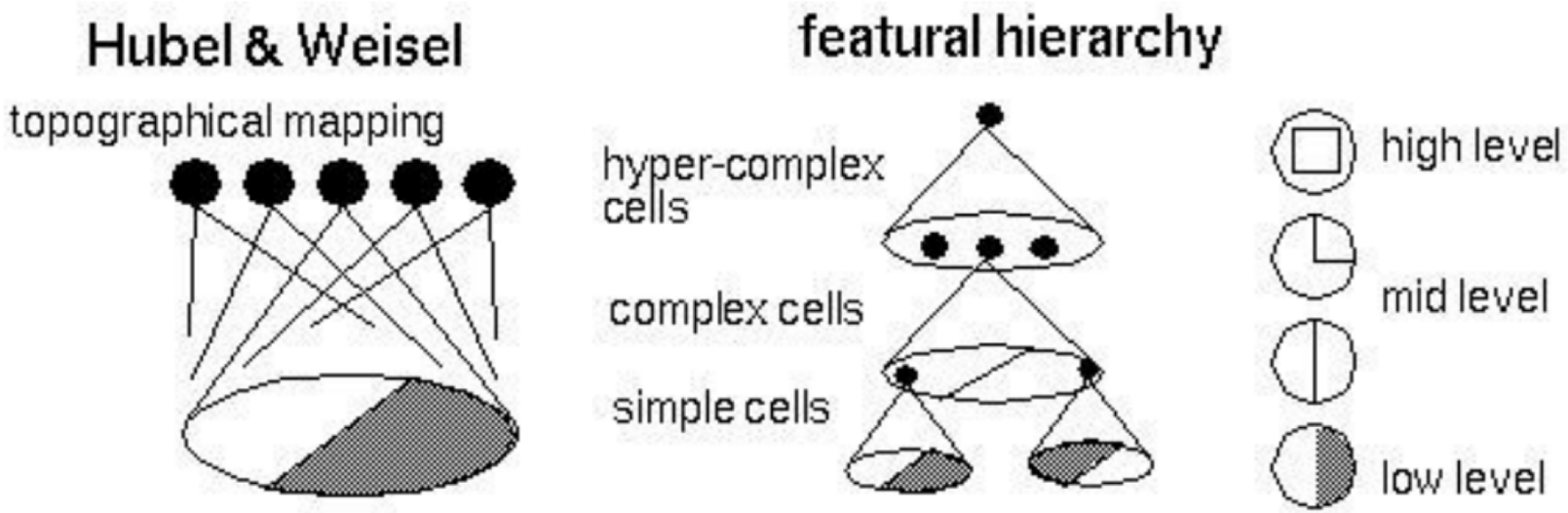
https://youtu.be/8VdFf3egwfg?t=1m10s

A bit of history

Topographical mapping in the cortex:
nearby cells in cortex represented
nearby regions in the visual field

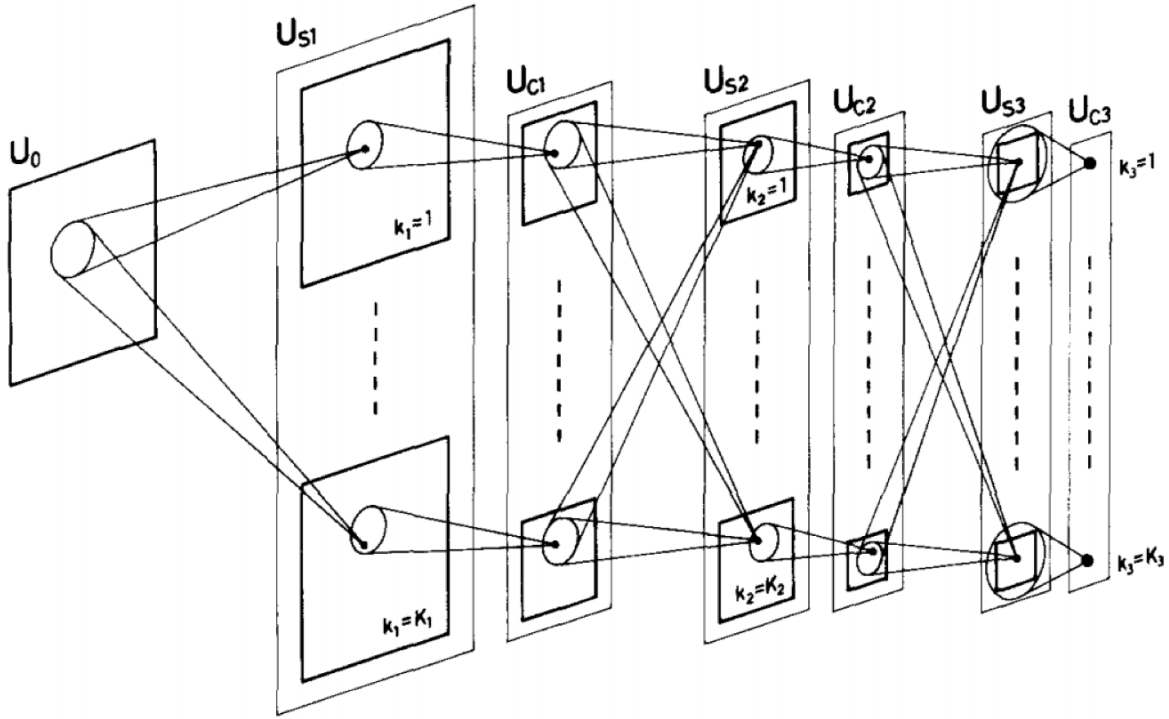


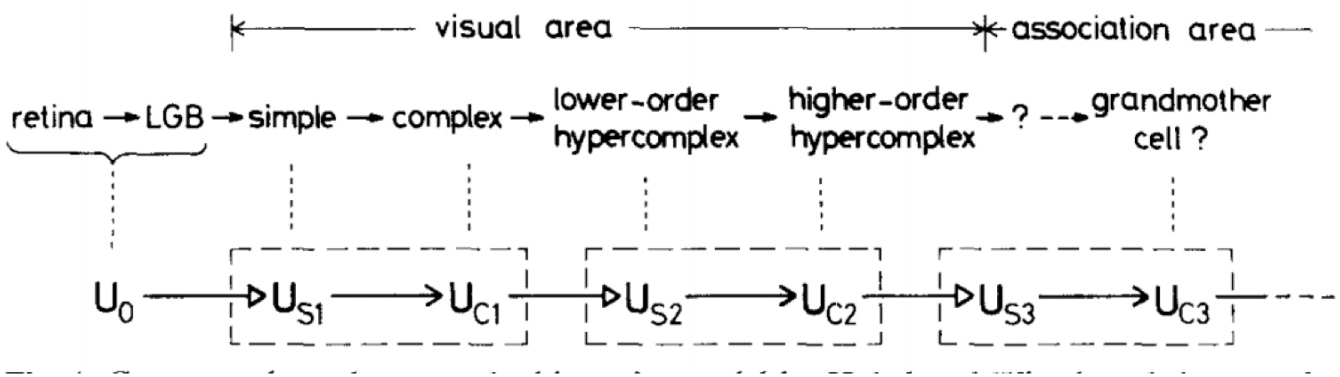
Hierarchical organization

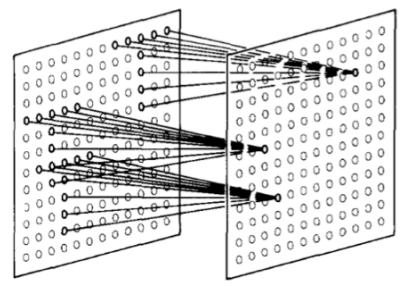


Brief History – The First ConvNet - 1980

- Neocognitron: multiple convolutional and pooling layers similar to modern networks, but the network was trained by using a reinforcement scheme
- Did not still use backpropagation
- Translational invariant





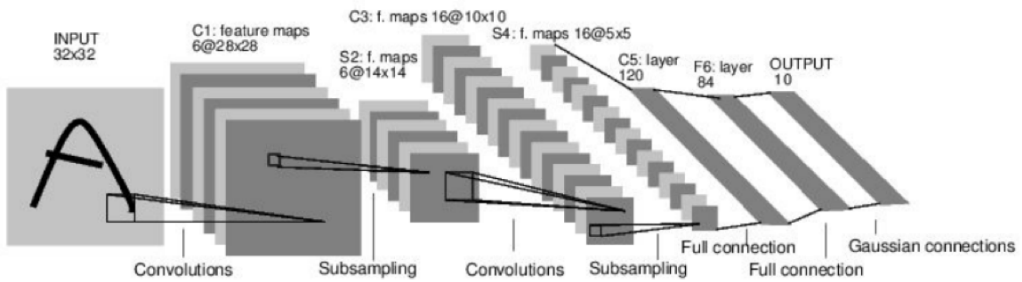


Kunihiko Fukushima

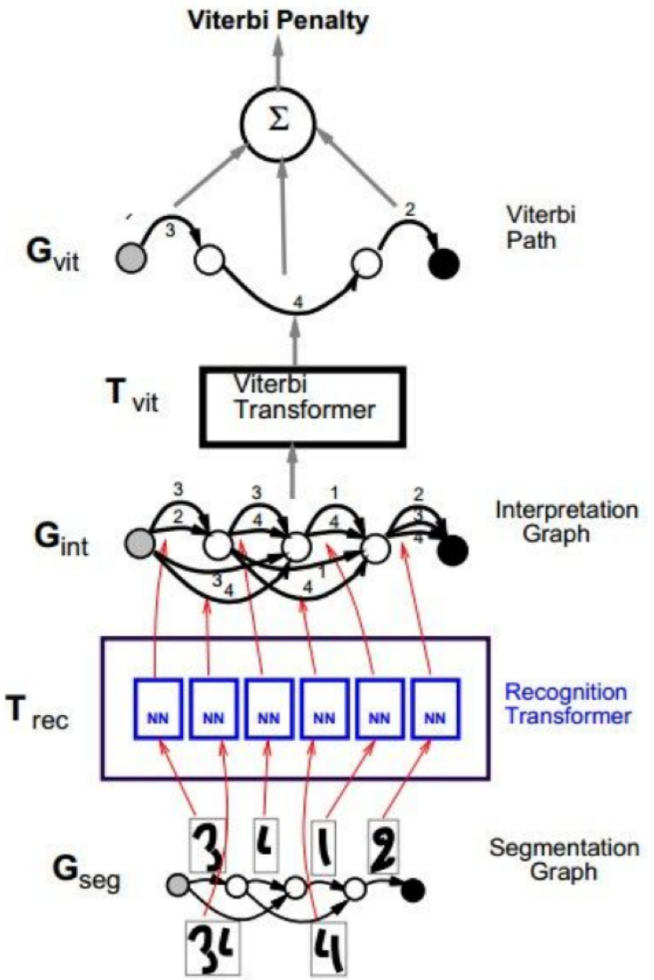
A bit of history:

Gradient-based learning applied to document recognition

[LeCun, Bottou, Bengio, Haffner 1998]

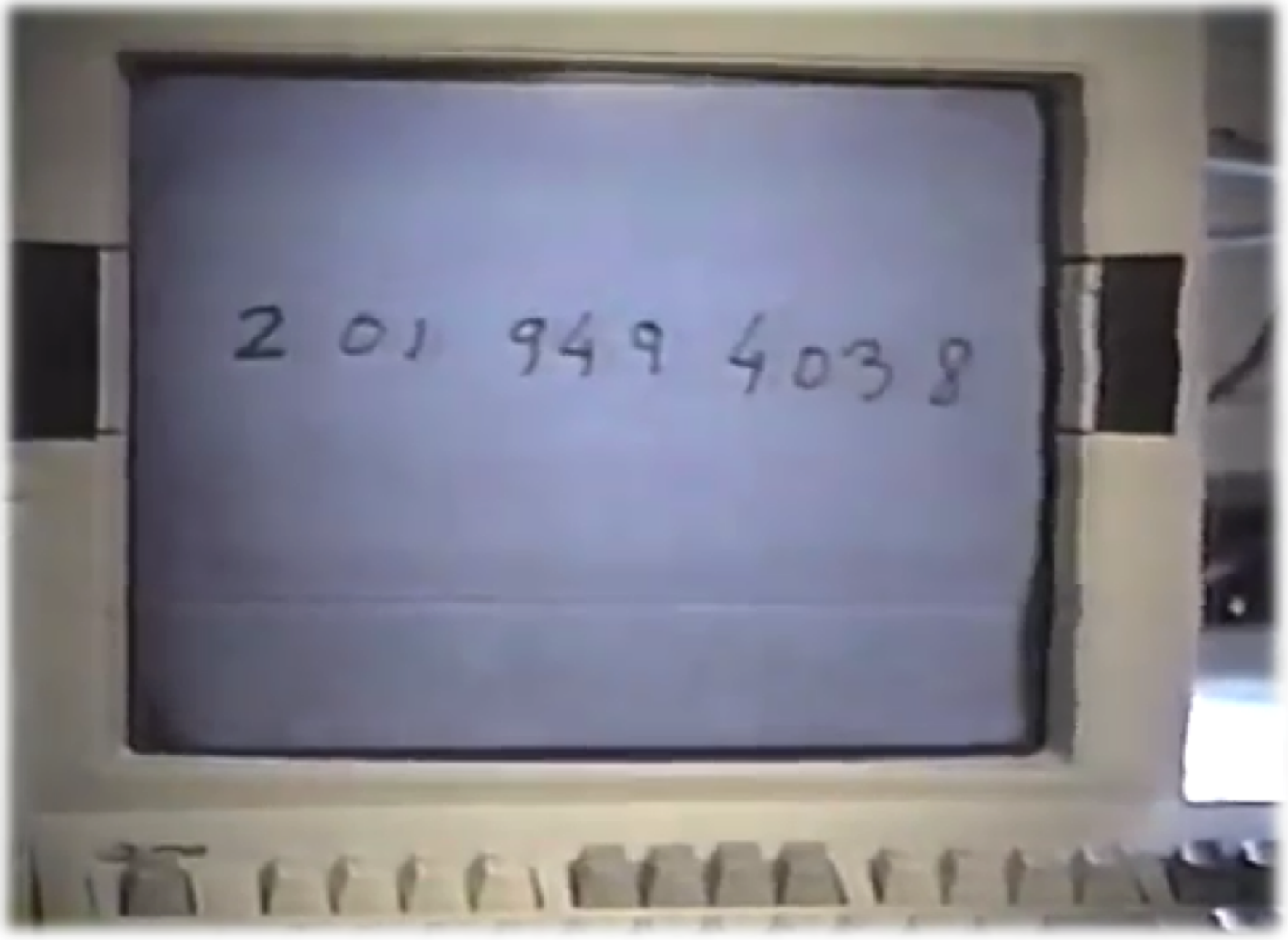


LeNet-5



Yann LeCun

Brief History – LeNet-5 In Action

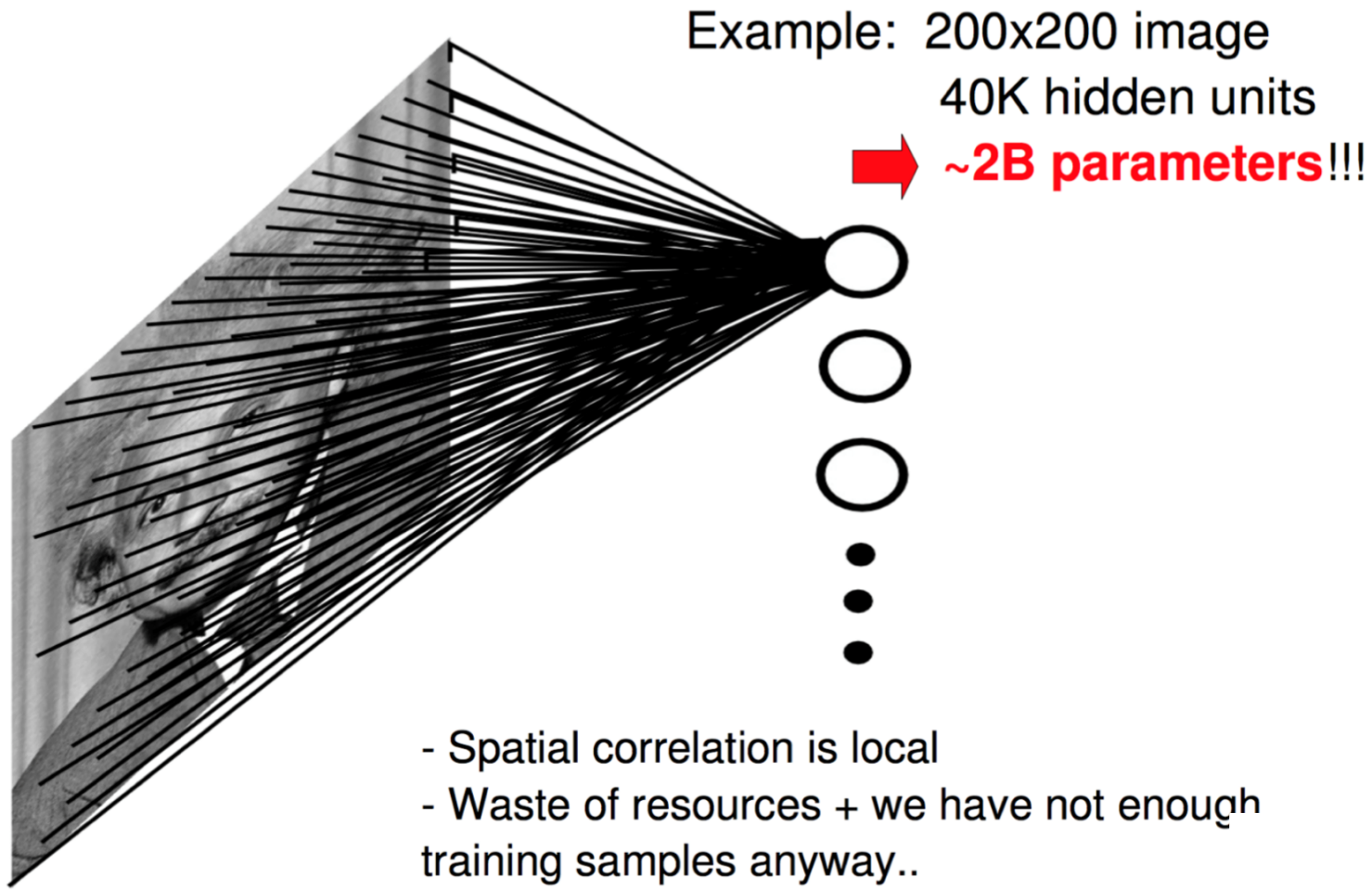


Brief History – So What Changed (since the 1970s)?

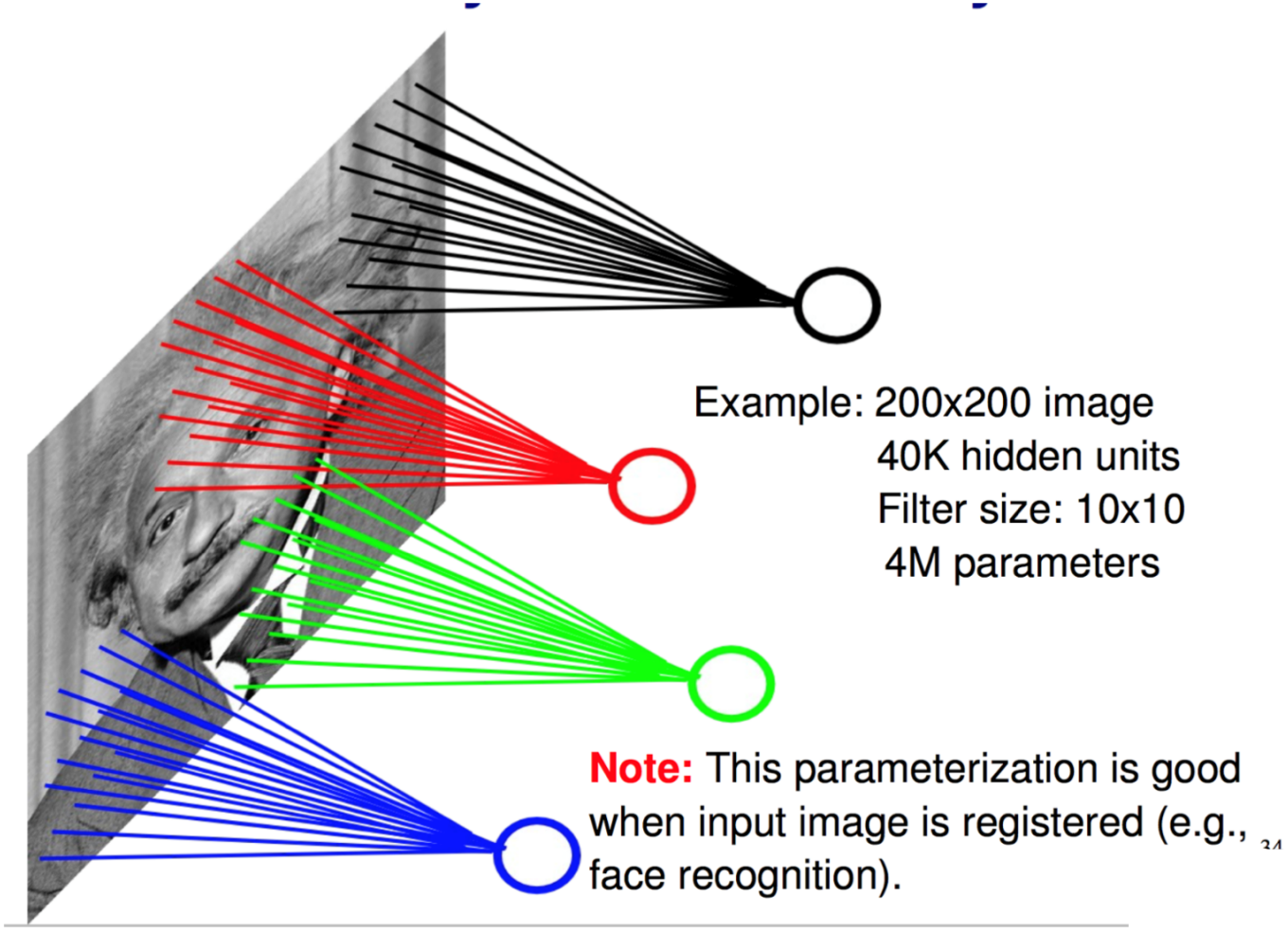
- Three things:
  - Availability of large amounts of labeled data e.g. ImageNet
  - Compute power – A single NVidia TITAN X card churns of 11 TFLOPS with ~3500 cores, **TITAN V**?
  - Algorithms:
    - ReLU - Found to decrease training time
    - Dropout – prevent overfitting to the training data

Building Blocks:
Convolution

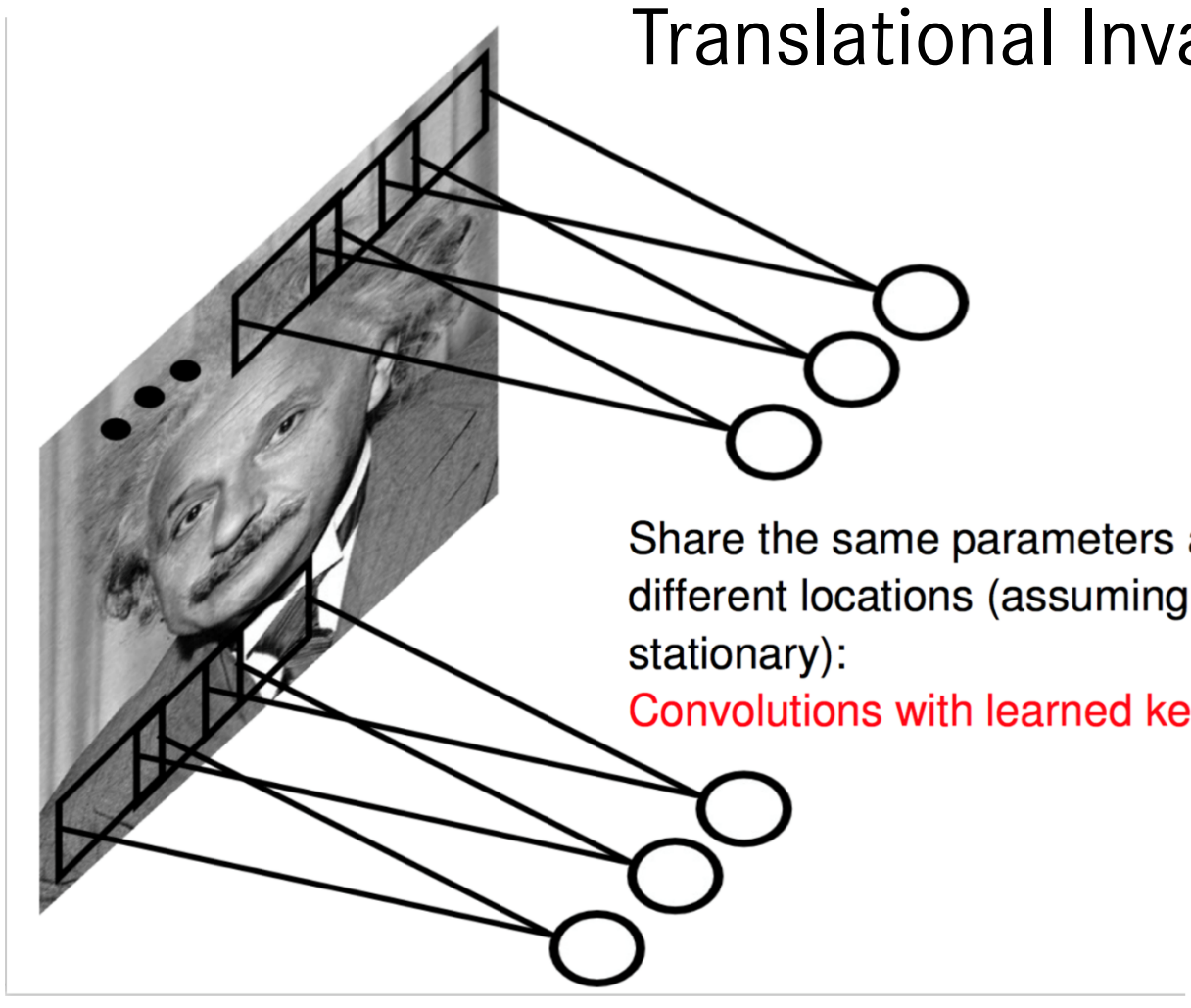
Building Blocks - Convolution



Building Blocks - Convolution

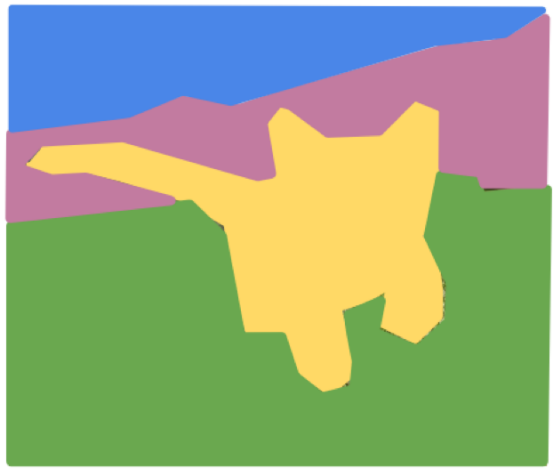


Building Blocks - Convolution



High Level Computer Vision Tasks

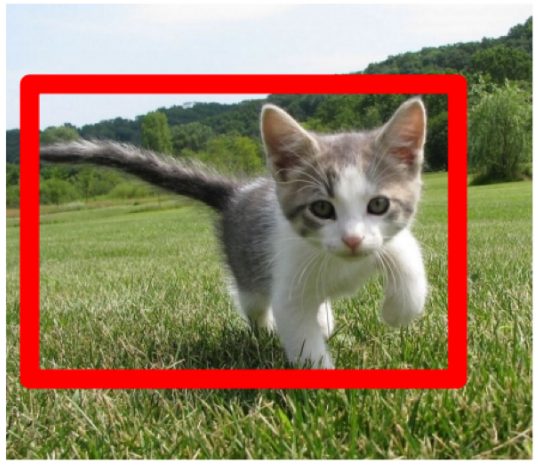
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

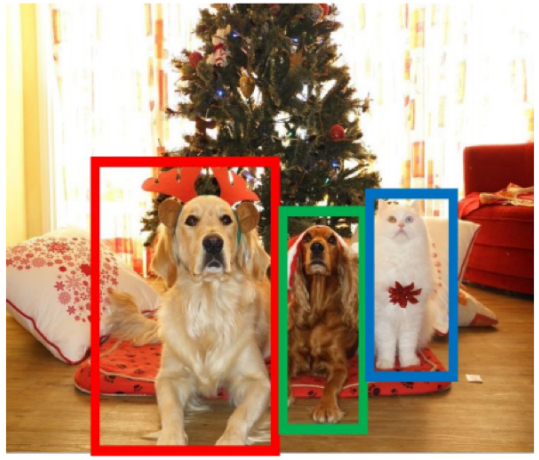
Classification + Localization



CAT

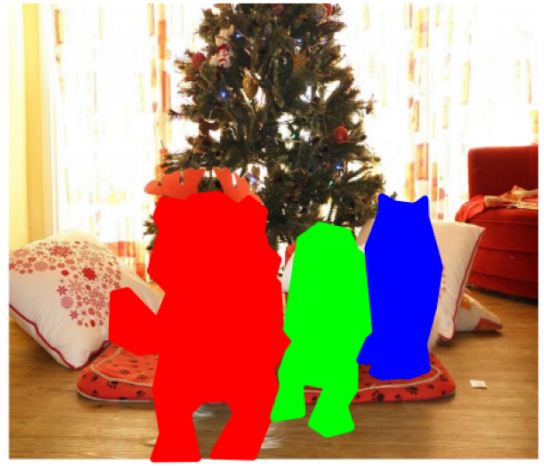
Single Object

Object Detection



DOG, DOG, CAT

Instance Segmentation

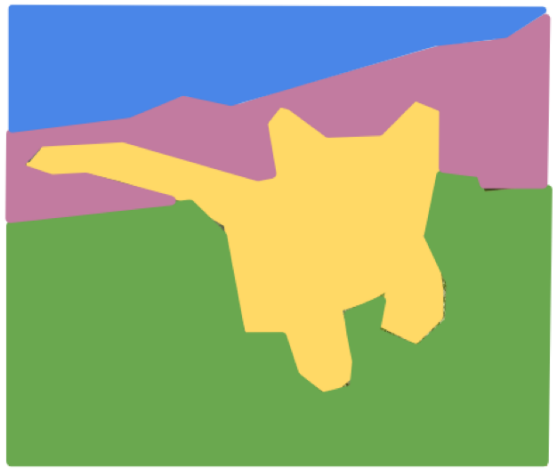


DOG, DOG, CAT

Multiple Object

Semantic Segmentation

Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Classification + Localization

CAT

Single Object

Object Detection

DOG, DOG, CAT

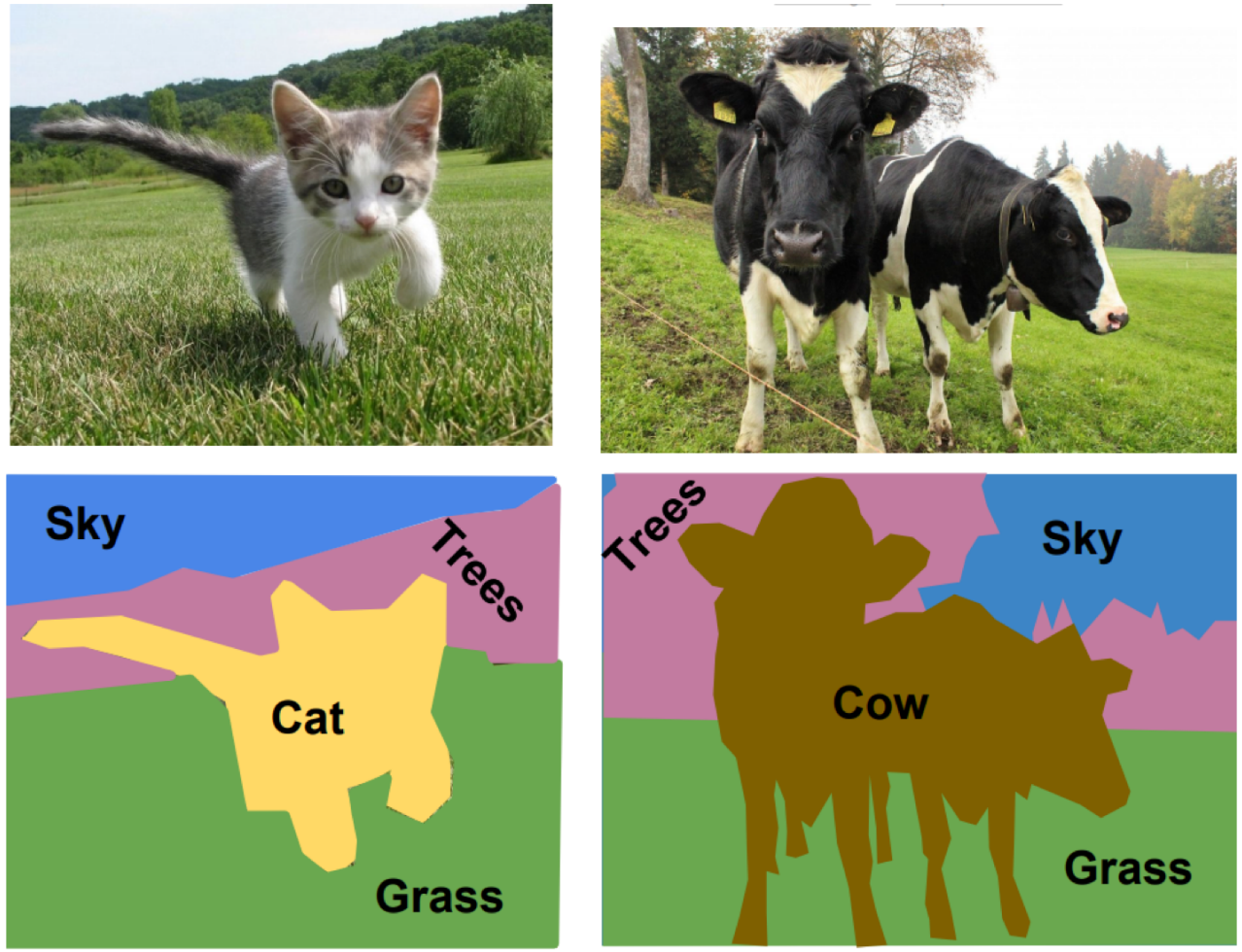
Instance Segmentation

DOG, DOG, CAT

Multiple Object

Semantic Segmentation

- Label each pixel in the image with a category label
- Don’t differentiate instances, only care about pixels



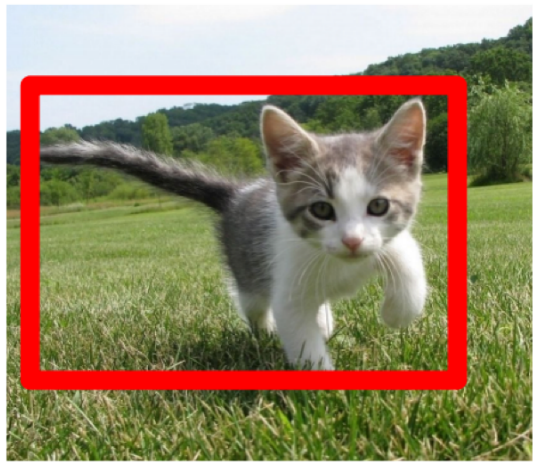
Classification + Localization

Semantic Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

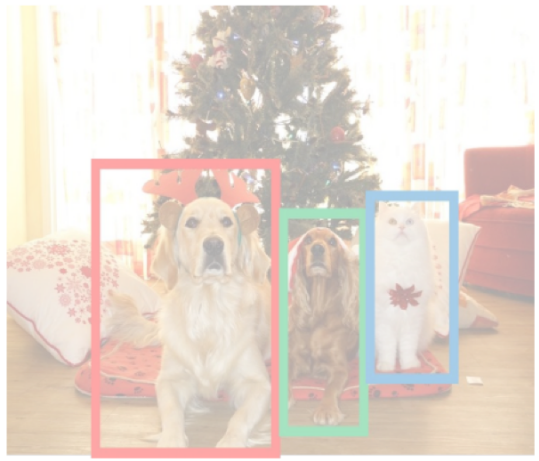
Classification + Localization



CAT

Single Object

Object Detection



DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

Multiple Object

Object Detection

Semantic Segmentation

GRASS, CAT, TREE, SKY

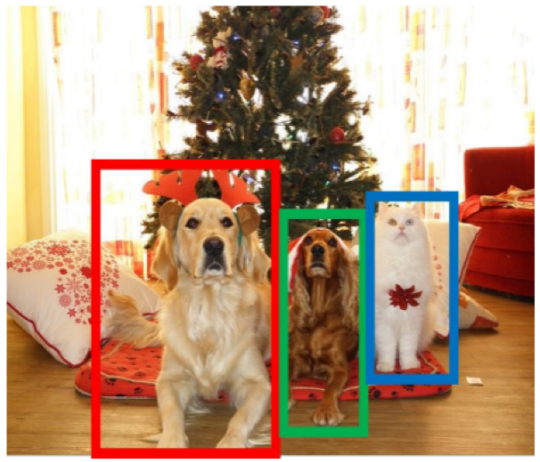
No objects, just pixels

Classification + Localization

CAT

Single Object

Object Detection



DOG, DOG, CAT

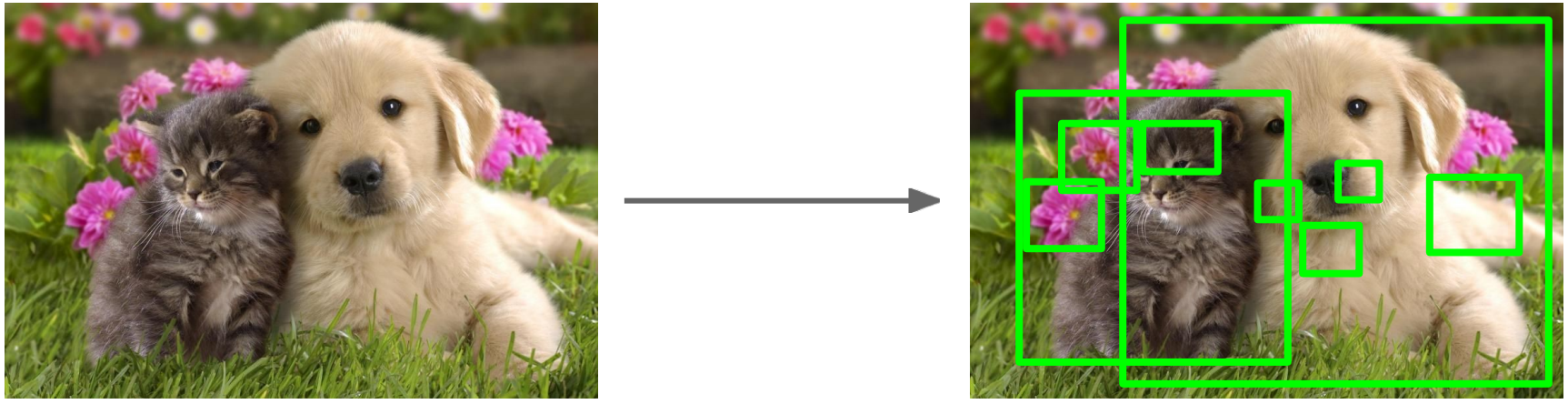
Instance Segmentation

DOG, DOG, CAT

Multiple Object

Region Proposals

- Find “blobby” image regions that are likely to contain objects – non DL based algorithm
- Relatively fast to run; e.g. Selective Search gives 1000 region proposals in a few seconds on CPU



Instance Segmentation

Semantic Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

Classification + Localization

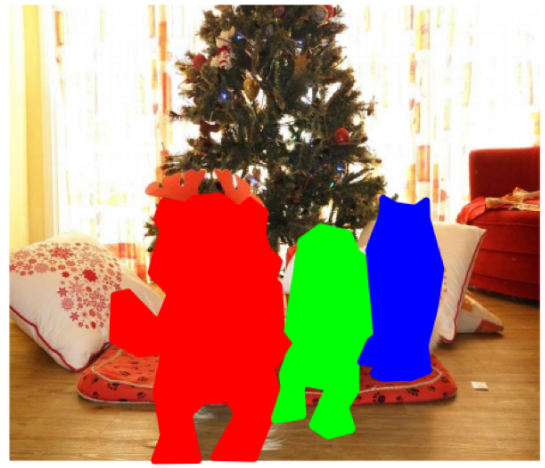
CAT

Single Object

Object Detection

DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Object

Sources

A lot of the material has been shamelessly and gratefully collected from:

- **http://cs231n.stanford.edu/**
- https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-history-training/
- https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html
- https://research.fb.com/learning-to-segment/
- https://research.fb.com/deep-learning-tutorial-at-cvpr-2014/
- https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/practicals/practical4.pdf
- http://torch.ch/docs/developer-docs.html
- https://github.com/torch/nn/blob/31d7d2bc86a914e2a9e6b3874c497c60517dc853/doc/module.md
- https://web.stanford.edu/group/pdplab/pdphandbook/handbookch6.html
- http://neuralnetworksanddeeplearning.com/chap2.html

Thank you!