CS4501: Introduction to Computer Vision

Course Recap

Various slides from previous courses by:

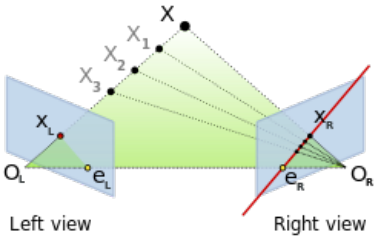
D.A. Forsyth (Berkeley / UIUC), I. Kokkinos (Ecole Centrale / UCL). S. Lazebnik (UNC / UIUC), S. Seitz (MSR / Facebook), J. Hays (Brown / Georgia Tech), A. Berg (Stony Brook / UNC), D. Samaras (Stony Brook) . J. M. Frahm (UNC), V. Ordonez (UVA), Steve Seitz (UW).

Today's Class

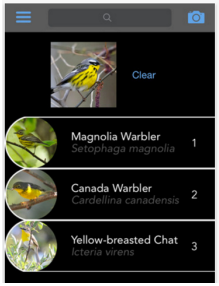
- Course Recap

kaggle

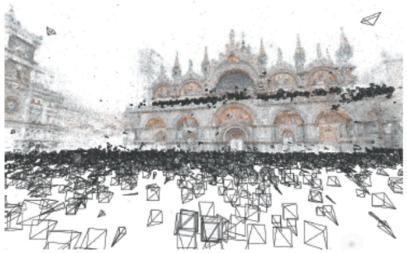
Create an algorithm to distinguish dogs from cats



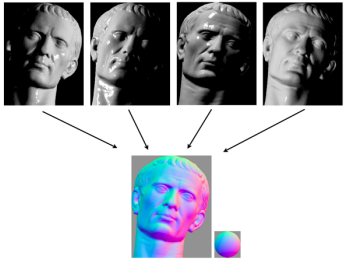
Birdsnap

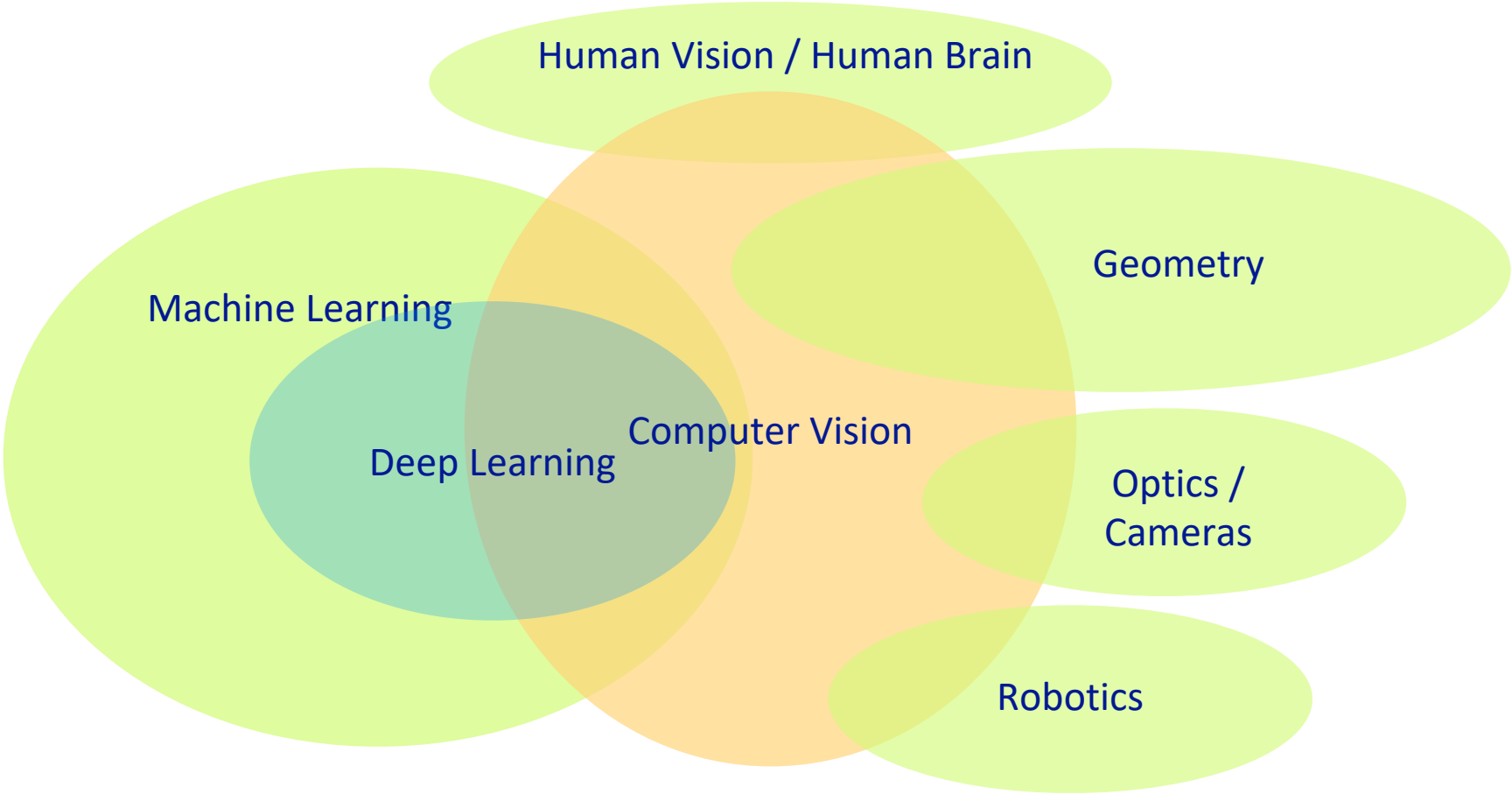


Computer Vision



Face Detection in Cameras




Human Vision / Human Brain
Machine Learning
Geometry
Deep Learning
Computer Vision
Optics /
Cameras
Robotics

Objectives

- Develop Intuitions Between Human Vision and Computer Vision
- Understanding the Basics of 2D and 3D Computer Vision
- Become familiar with the technical approaches in computer vision such as registration, matching, and recognition
- Obtain practical experience in the implementation of computer vision applications.

Introduction to Computer Vision

Foundations

- Cameras
- Human Vision
- Image Processing
- Image Filtering
- Edge Detection
- Corner Detection
- Line Detection
- Interest Points

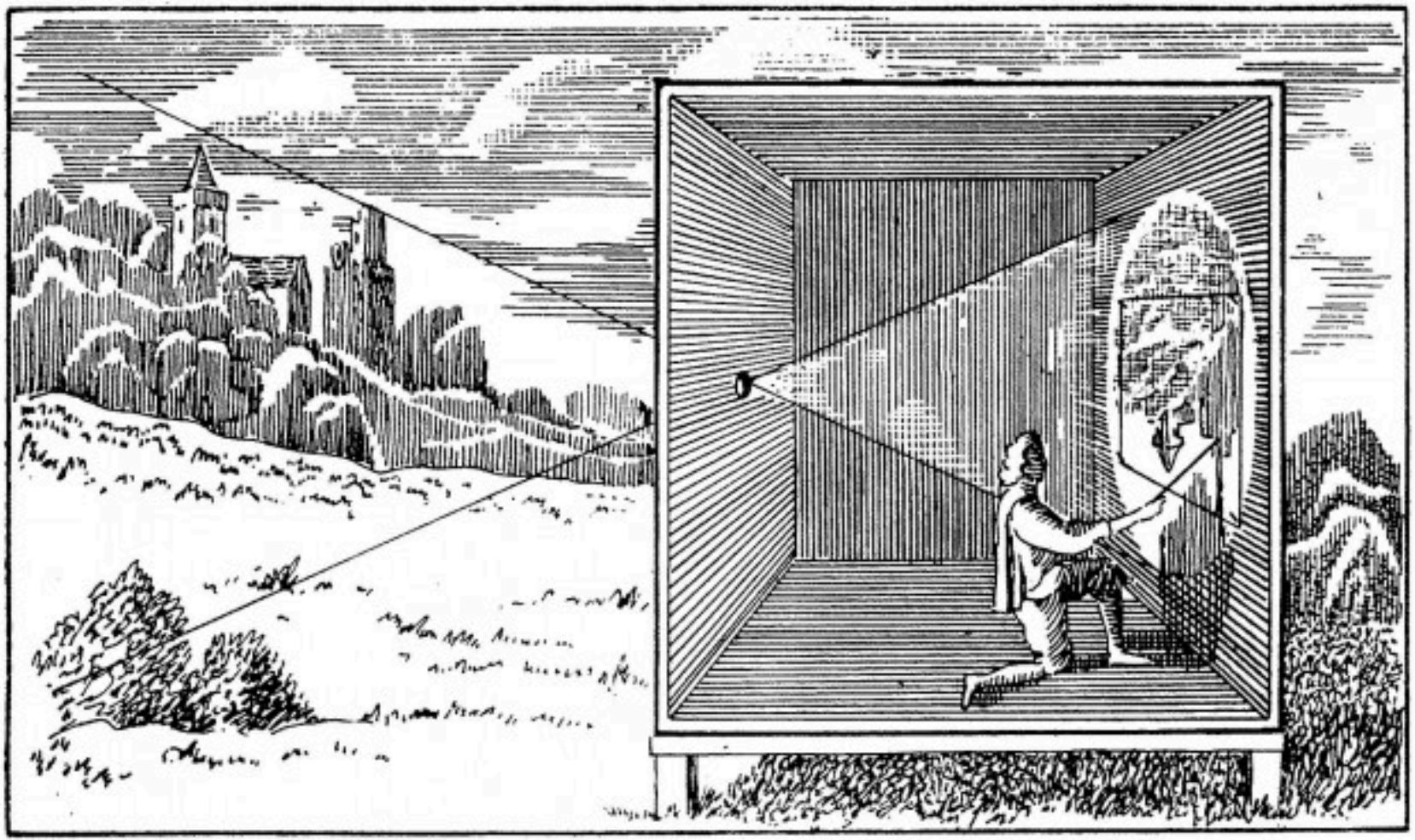
Geometry

- Point Matching
- RANSAC
- Homography Estimation
- Image Stitching
- Camera Calibration
- Dense Stereo
- Epipolar Geometry

Deep Learning

- Machine Learning
- Neural Networks
- Classification / Regression
- Object Detection
- Image Segmentation
- Generative Adversarial Networks
- Video and Optical Flow

Cameras



https://pixsylated.com/blog/stepping-inside-havana-camera-obscura/

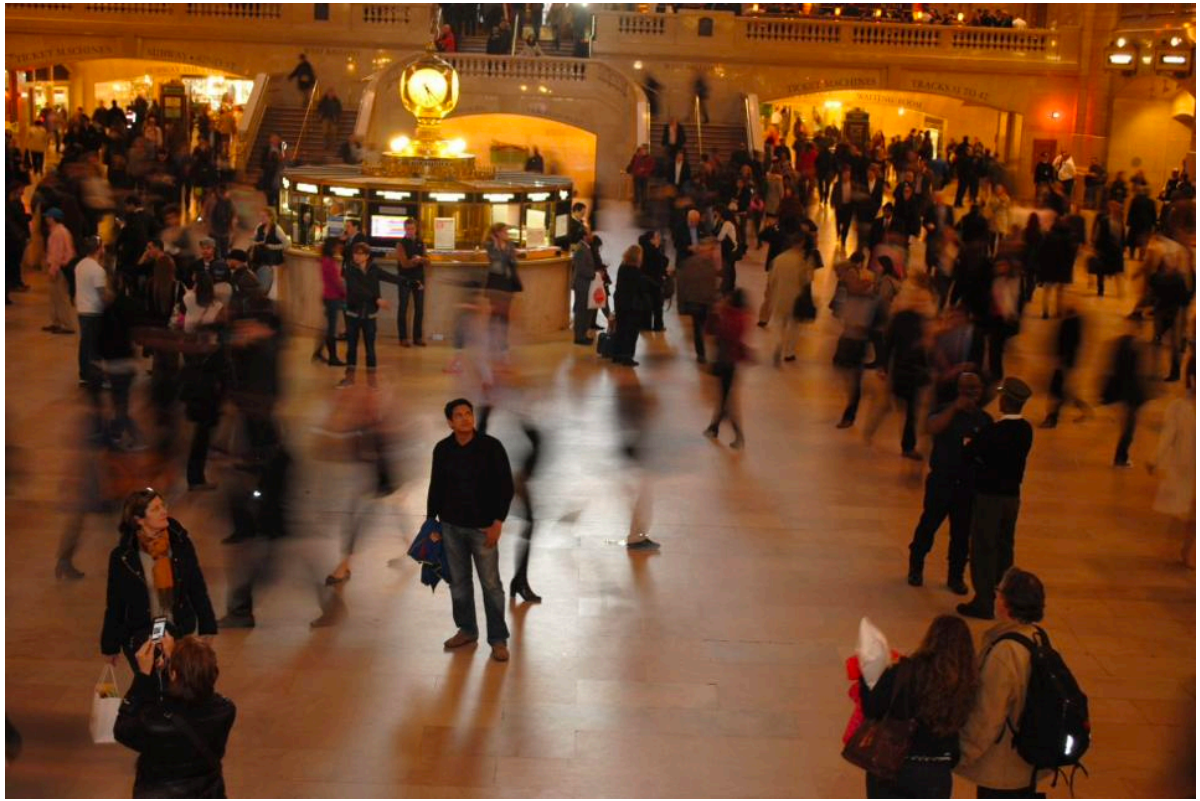
How to Shoot Photos in Manual?

- Shutter time
- Aperture
- ISO
- Focus / Auto-focus (Yes, you can shoot in manual and also probably should focus in manual)

Small Shutter Time / Speed

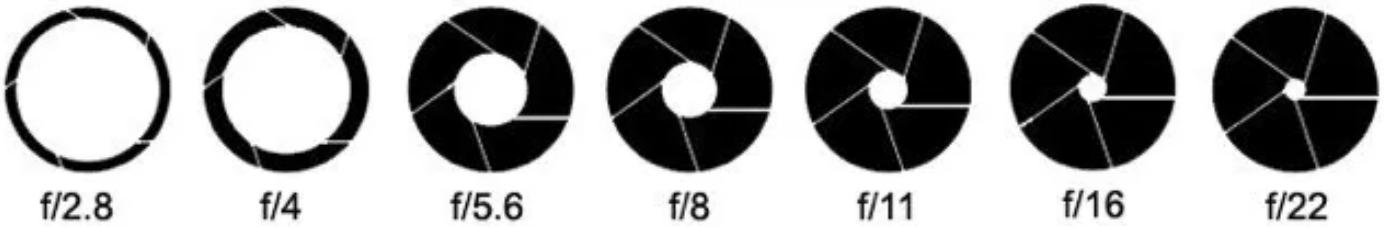
http://www.photographymad.com/pages/view/shutter-speed-a-beginners-guide

Long Shutter Time



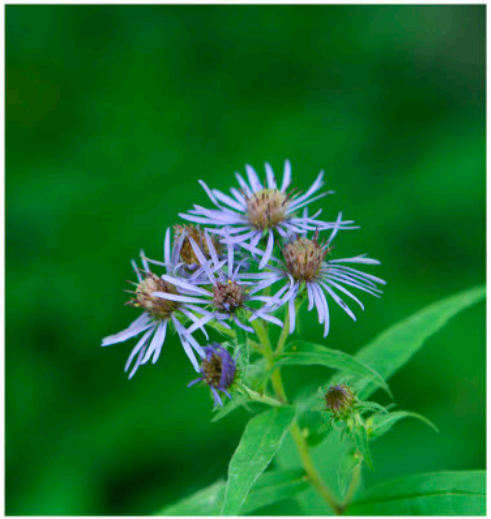
Long Shutter Time

Aperture

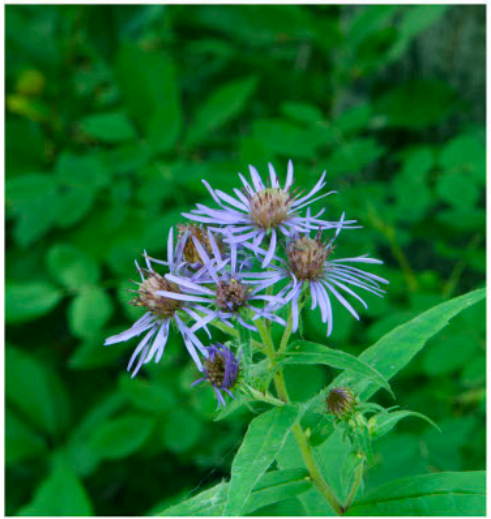


Large vs Small Aperture + Focus Control

Large Aperture (F4.0)
Background nicely blurred

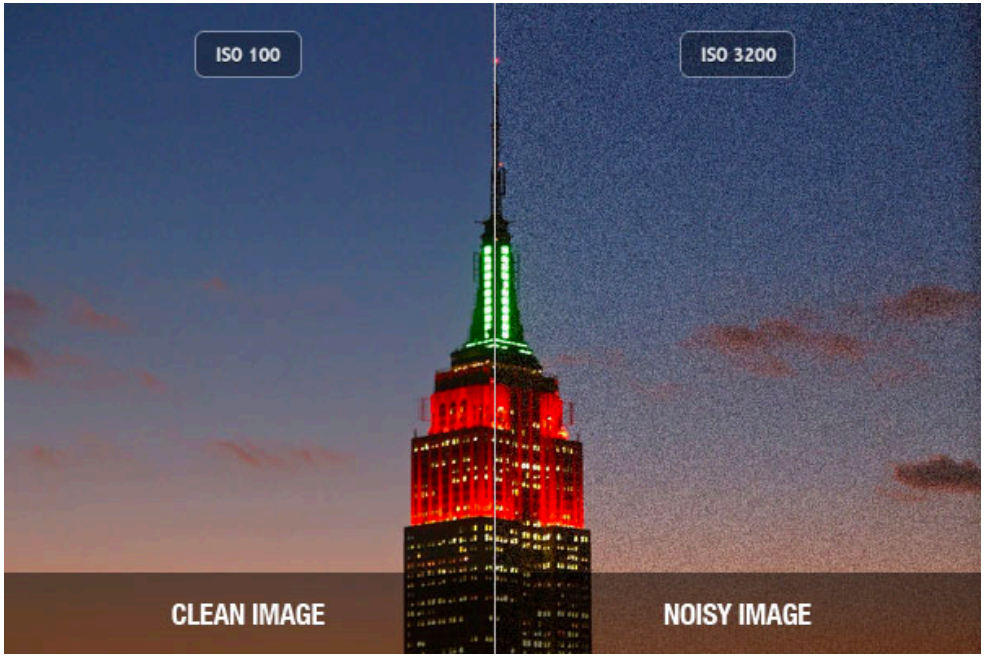


Small Aperture (F22)
Background is distracting



http://www.pgphotoclub.com/articles/aperture.html

ISO – Should be small ideally



https://www.exposureguide.com/iso-sensitivity/

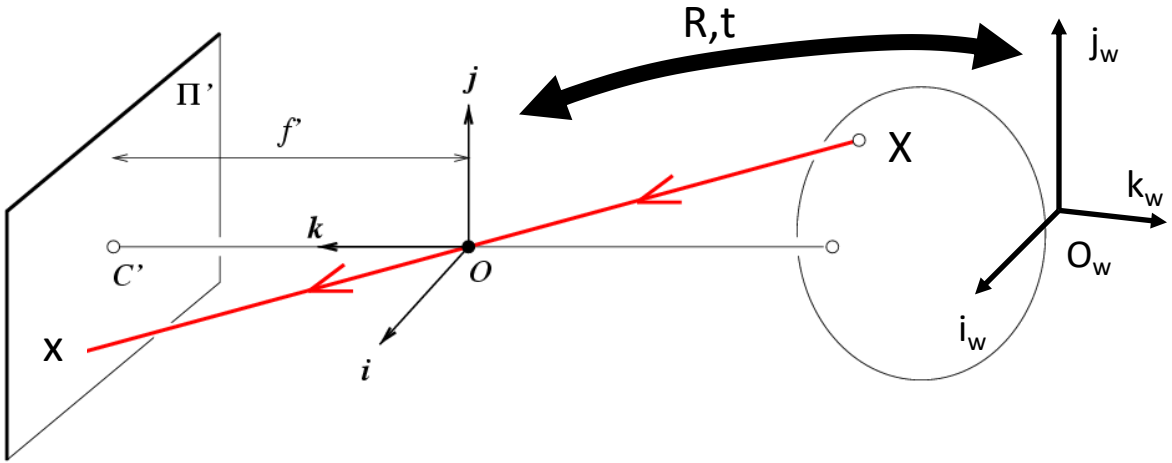
Trade-offs (We need light to capture a photo)

- Small Aperture leads to less light
  (but allows more focus on objects)

- Small Shutter speed leads to less light
  (but allows capturing fast moving objects)

- Small ISO leads to less light
  (but produces less noisy “grainy” output)

Projection matrix (World Coordinates to Image Coordinates)



$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

x: Image Coordinates: (u,v,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1)

Intrinsic Camera Properties: **K**

Extrinsic Camera Properties: **[R t]**

Slide Credit: Savarese

Reflection

Body Reflection:

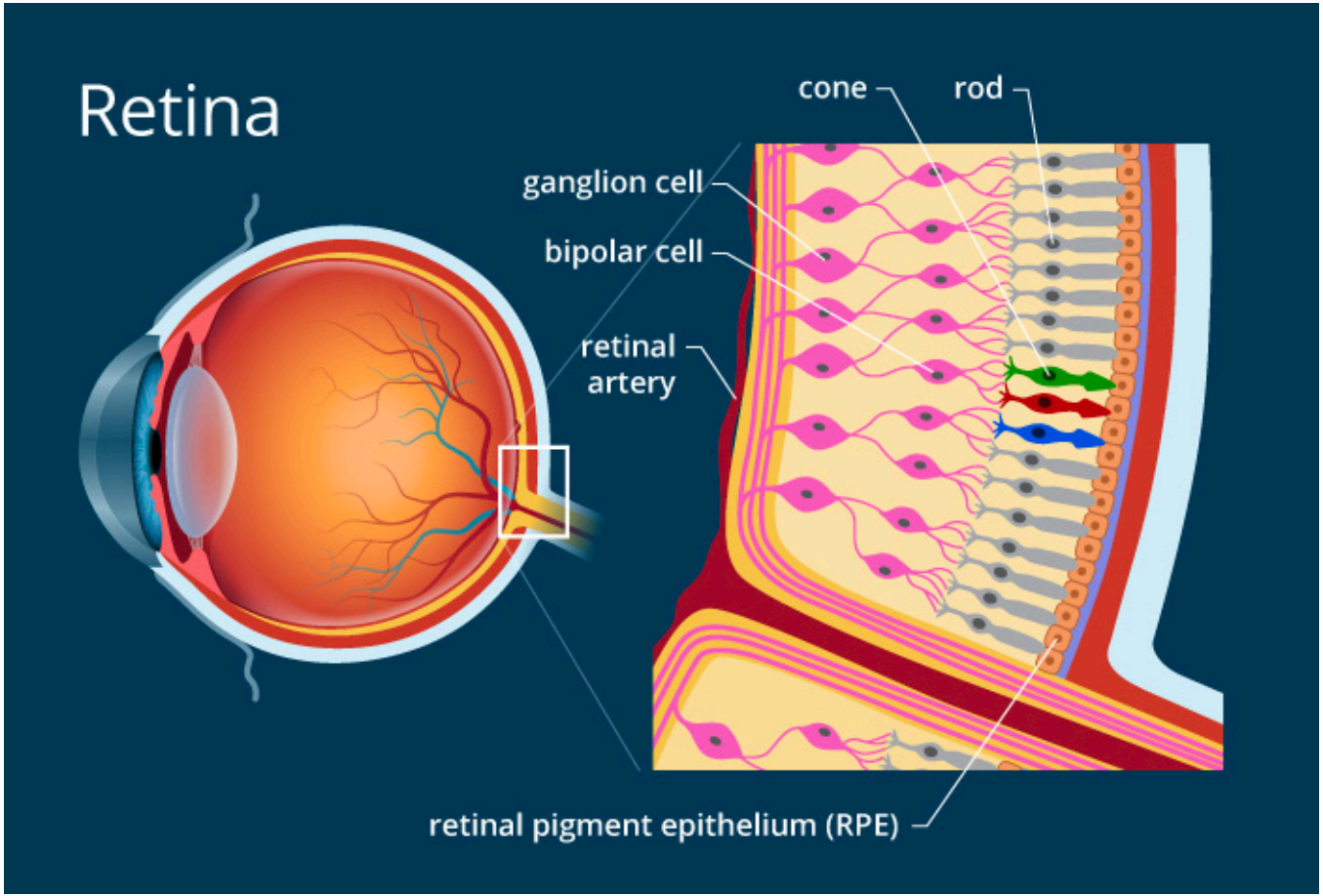
Diffuse Reflection
Matte Appearance
Non-Homogeneous Medium
Clay, paper, etc

Surface Reflection:

Specular Reflection
Glossy Appearance
Highlights
Dominant for Metals

Many materials exhibit
both Reflections:

Slide by Aaron Bobick



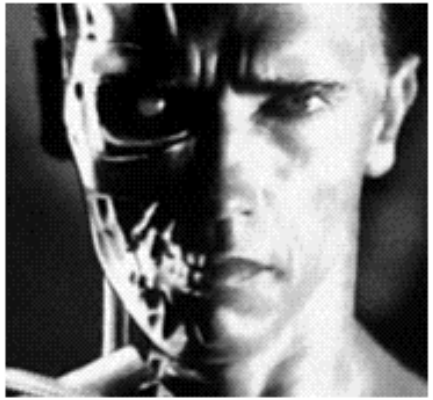
https://www.findlight.net/blog/2018/03/16/artificial-photoreceptors/

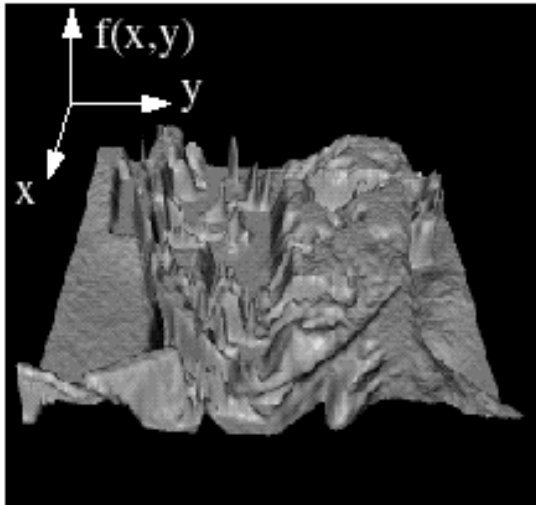
What the Frog's Eye Tells the Frog's Brain*

J. Y. LETTVIN†, H. R. MATURANA‡, W. S. McCULLOCH||, SENIOR MEMBER, IRE, AND W. H. PITTS||

Images as Functions

$$z = f(x, y)$$



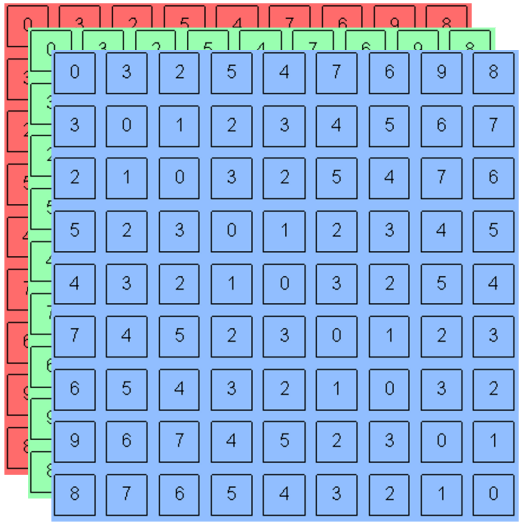


- The domain of x and y is [0, img-width) and [0 and img-height)
- x, and y are discretized into integer values.

Images as Matrices

0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

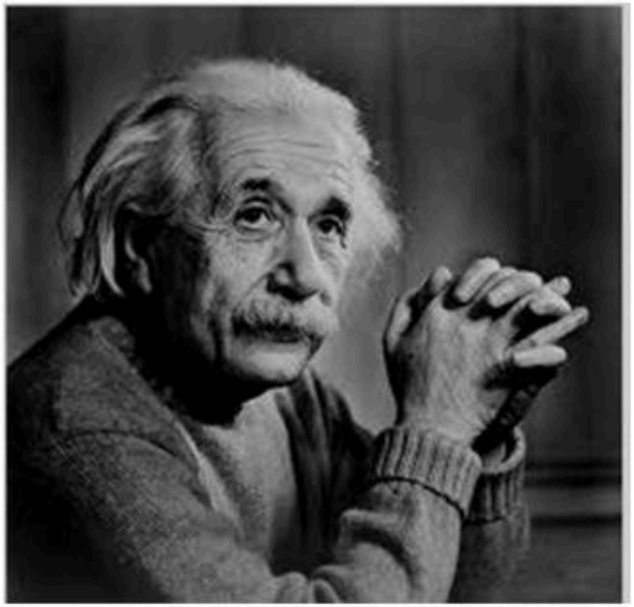
Color Images as Tensors



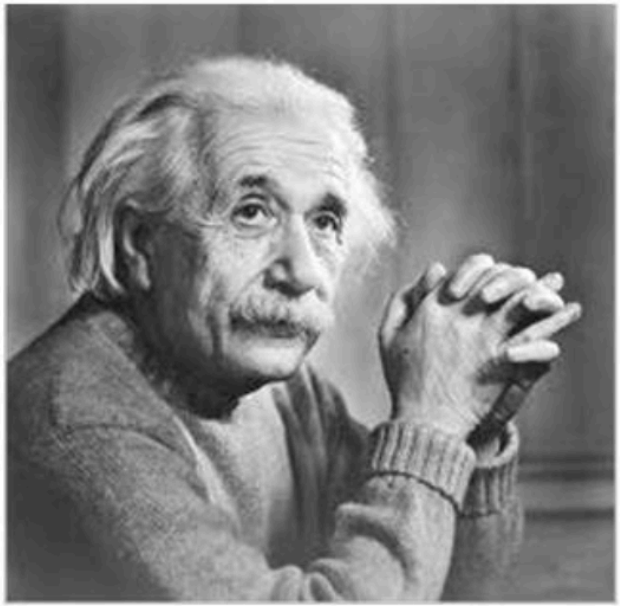
$channel\ x\ height\ x\ width$

Basic Image Processing

I



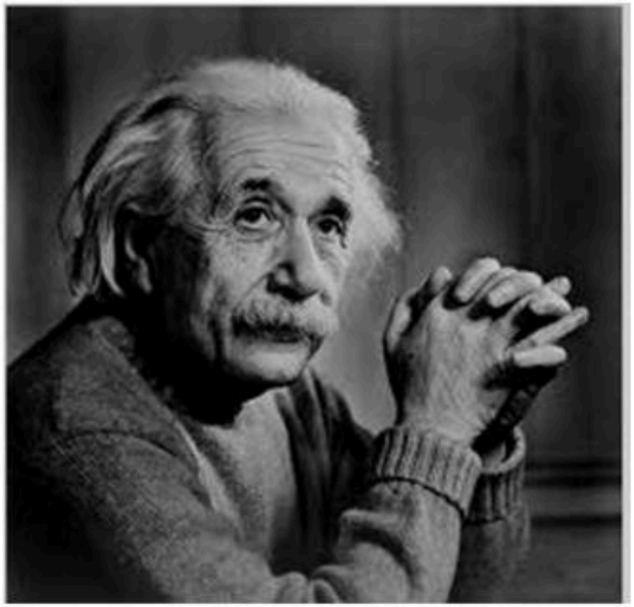
αI



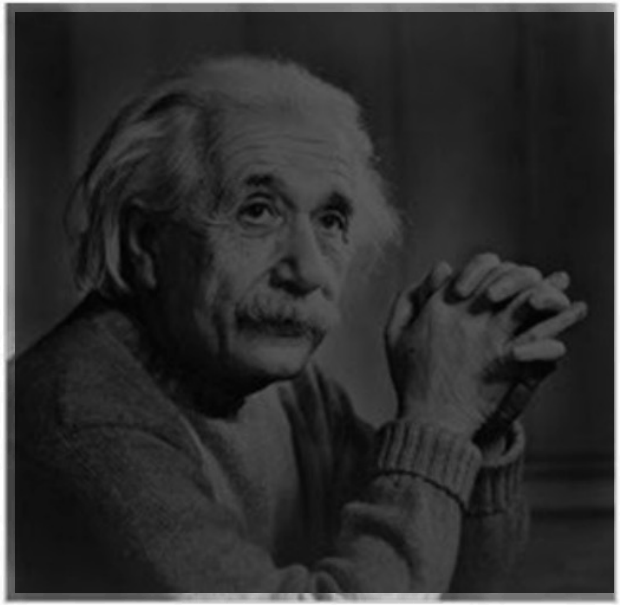
$\alpha > 1$

Basic Image Processing

I

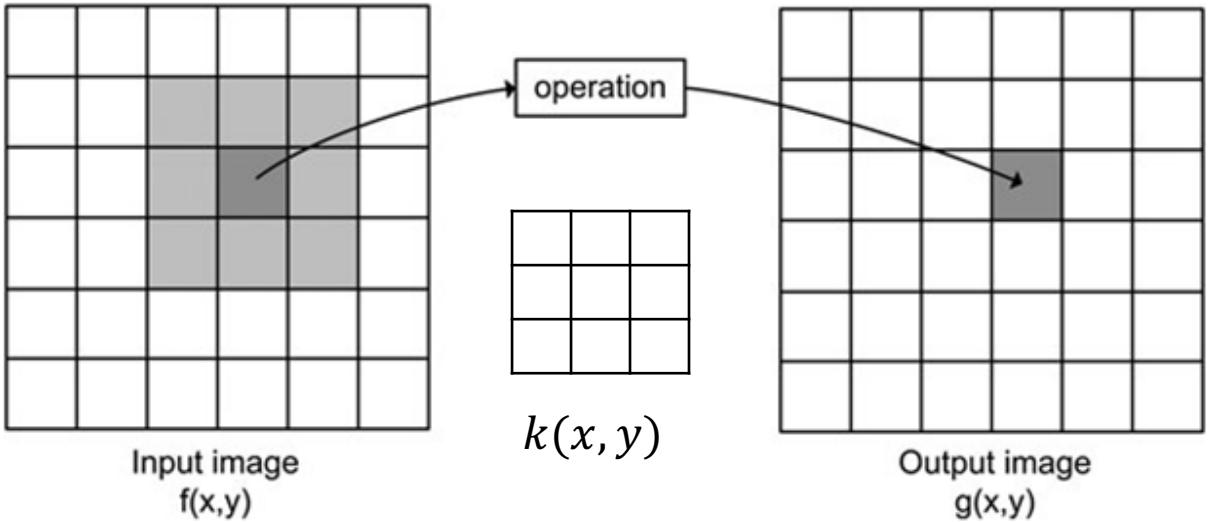


αI



$0 < \alpha < 1$

Image filtering: Convolution operator



$$g(x,y) = \sum_v \sum_u k(u,v) f(x - u, y - v)$$

Image Credit: http://what-when-how.com/introduction-to-video-and-image-processing/neighborhood-processing-introduction-to-video-and-image-processing-part-1/

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10							

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20						

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30					

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
				?					

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
						?			
				50					

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

Credit: S. Seitz

Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood

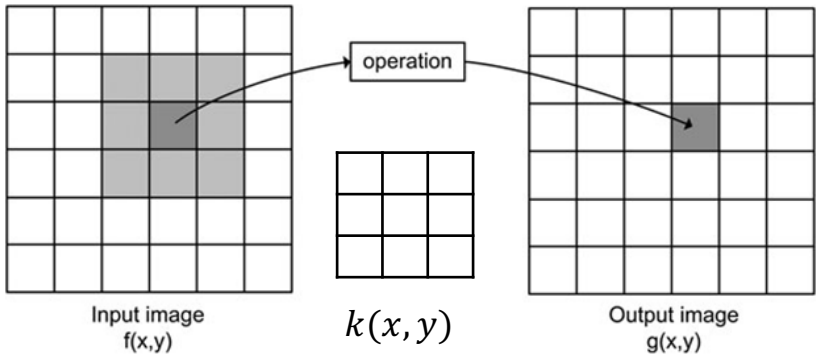
- Achieve smoothing effect (remove sharp features)

$g[\cdot,\cdot]$

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

Slide credit: David Lowe (UBC)

Image filtering: Convolution operator
Important filter: gaussian filter (gaussian blur)



$k(x,y)$ =

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

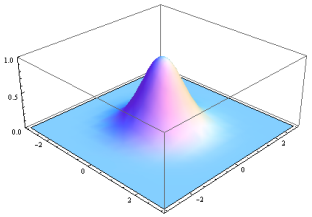
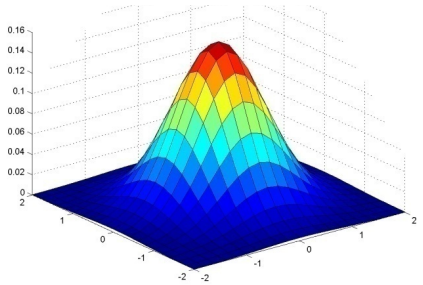
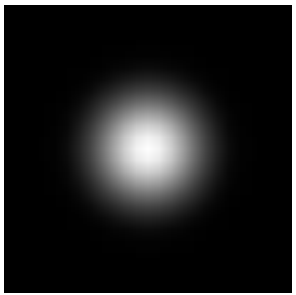


Image Credit: http://what-when-how.com/introduction-to-video-and-image-processing/neighborhood-processing-introduction-to-video-and-image-processing-part-1/

Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness





0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

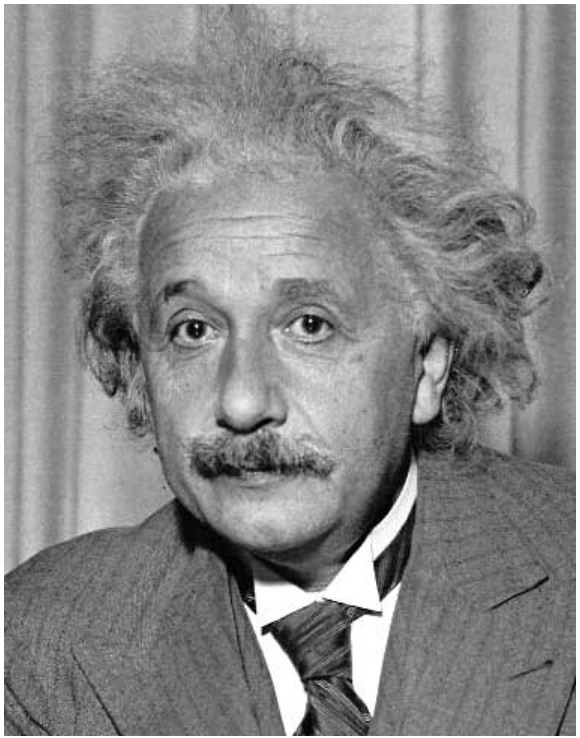
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Slide credit: Christopher Rasmussen

Image filtering: Convolution operator e.g. gaussian filter (gaussian blur)

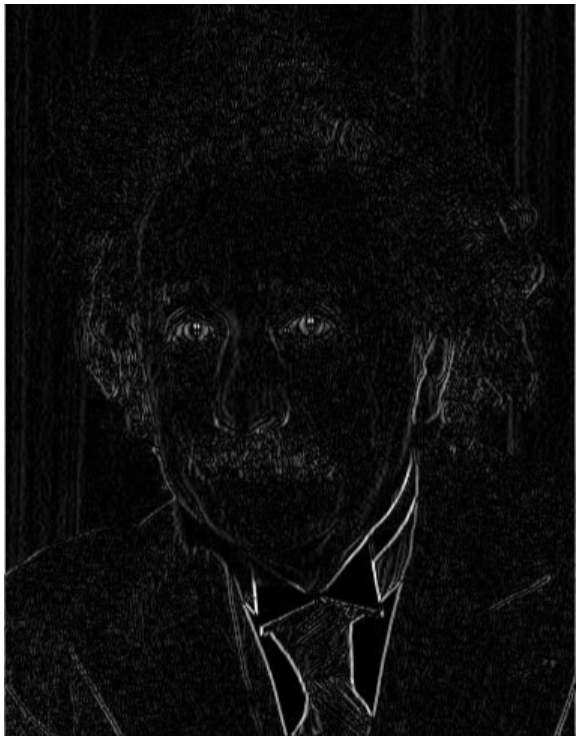
Image Credit: http://what-when-how.com/introduction-to-video-and-image-processing/neighborhood-processing-introduction-to-video-and-image-processing-part-1/

Other filters



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

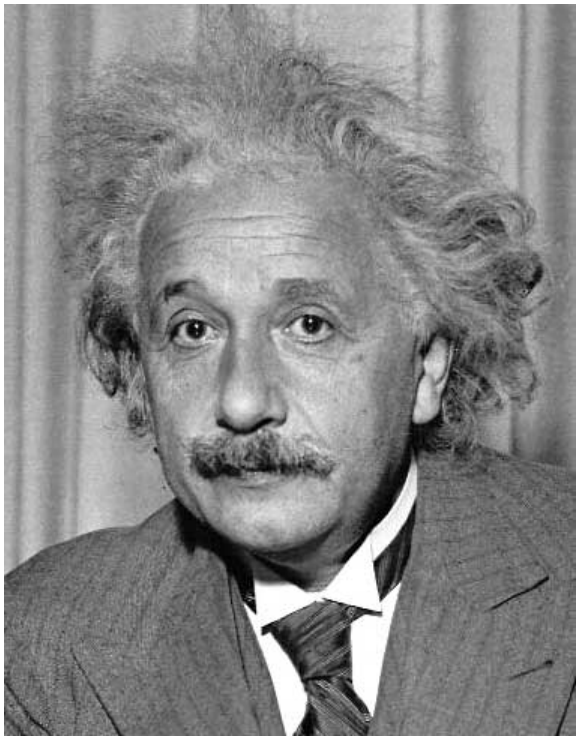
Sobel



Vertical Edge
(absolute value)

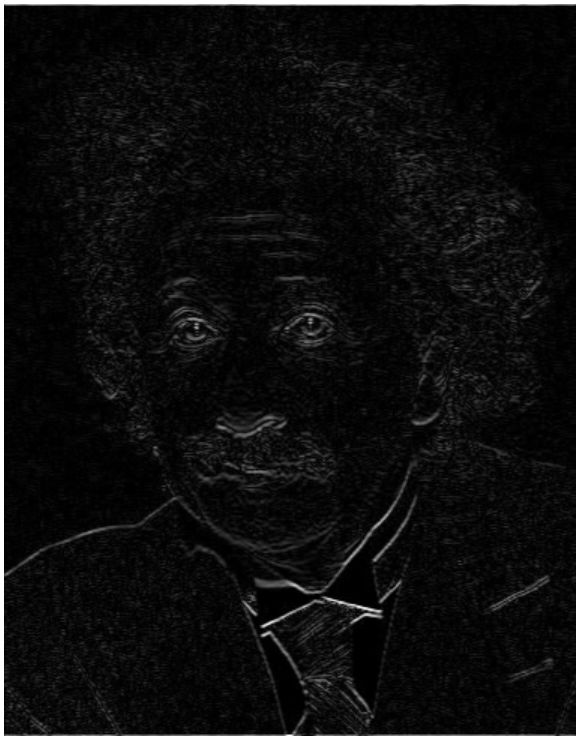
Slide by James Hays

Other filters



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

Sobel



Horizontal Edge
(absolute value)

Slide by James Hays

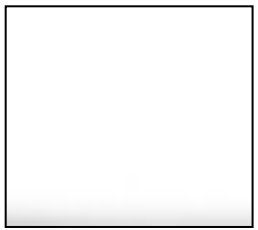
Harris Corner Detection

- Compute the following matrix of squared gradients for every pixel.

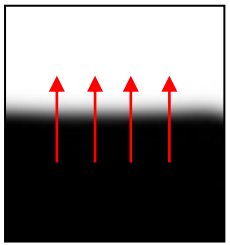
$$M = \sum_{patch} \begin{bmatrix} {I_x}^2 & I_x I_y \\ I_y I_x & {I_y}^2 \end{bmatrix}$$

I_x and I_y are gradients computed using Sobel or some other approximation.

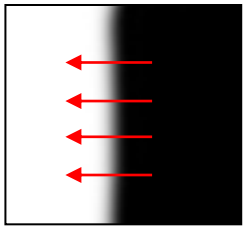
$$M = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$



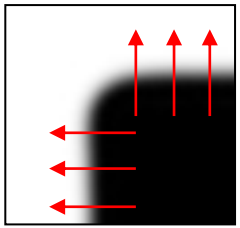
$$M = \begin{bmatrix} 0 & 0 \\ 0 & b \end{bmatrix}$$



$$M = \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix}$$

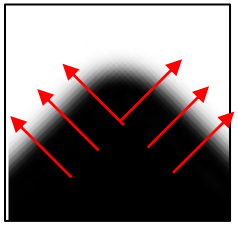


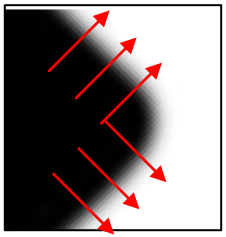
$$M = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$



Harris Corner Detection!

Works for these corners!





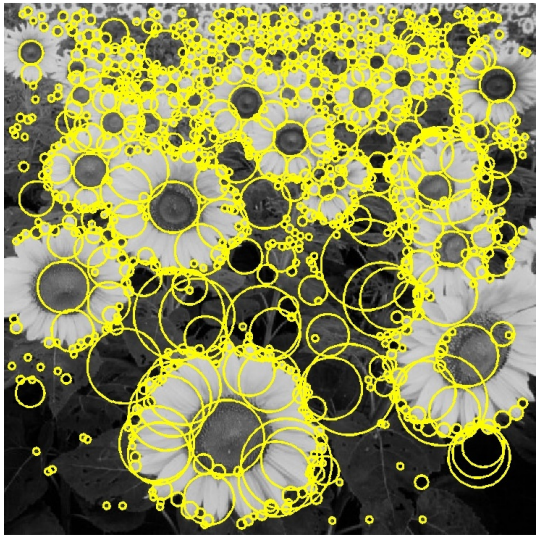
$$M = \sum_{patch} \begin{bmatrix} {I_x}^2 & I_x I_y \\ I_y I_x & {I_y}^2 \end{bmatrix} = \begin{bmatrix} a & c \\ c & b \end{bmatrix}$$

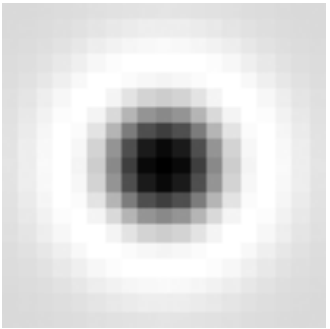
Use the following criteria to decide if it is a corner instead

$$\det(M) - 0.06\ trace(M)^2 > \tau$$

Basic idea

- Convolve the image with a “blob filter” at multiple scales and look for extrema of filter response in the resulting *scale space*

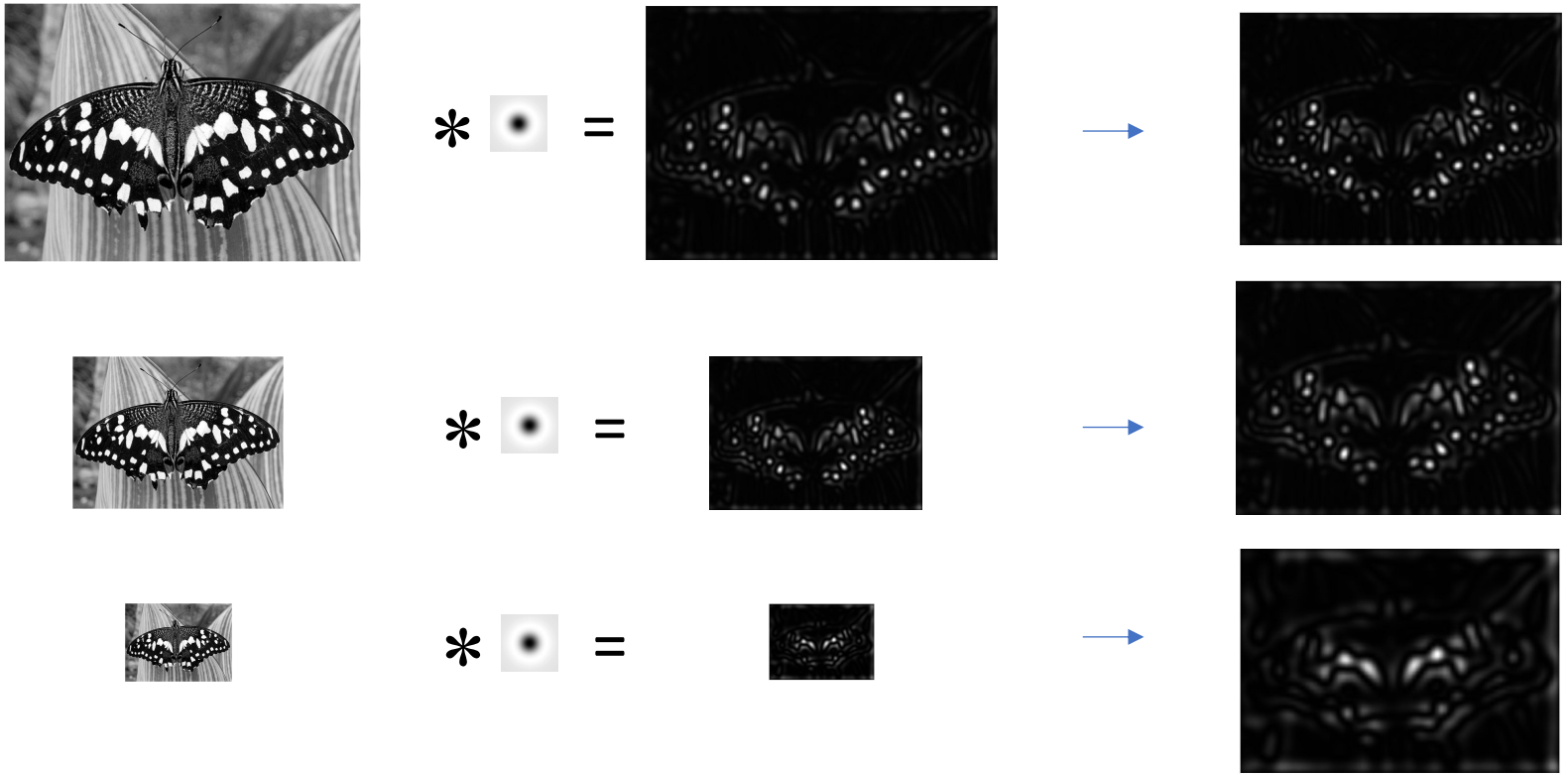




T. Lindeberg. Feature detection with automatic scale selection. *IJCV* 30(2), pp 77-116, 1998.

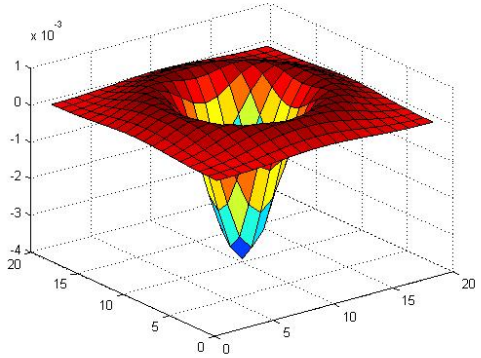
Slide by Svetlana Lazebnik

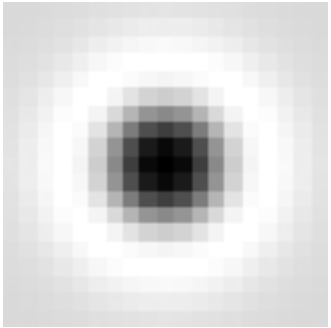
Blob at multiple scales – Option 1



Blob filter

- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

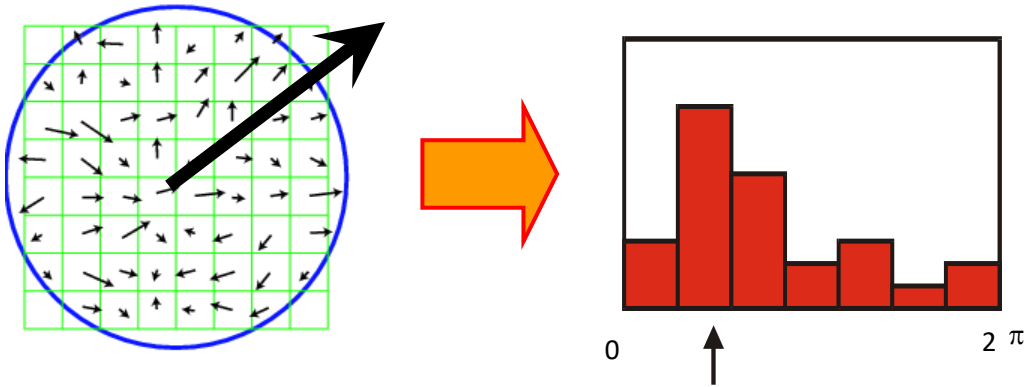




$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

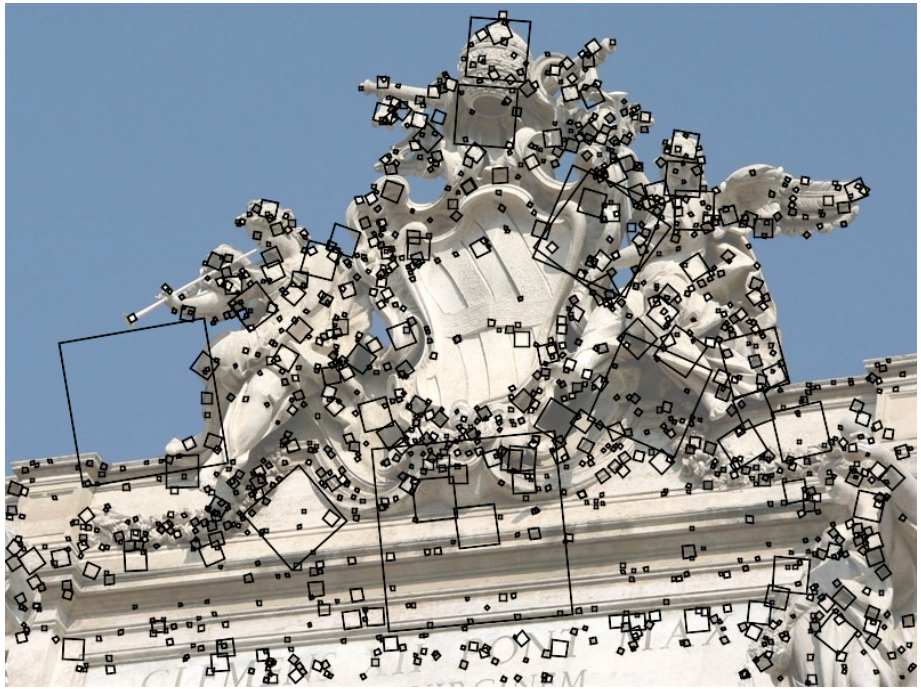
Eliminating rotation ambiguity

- To assign a unique orientation to circular image windows:
  - Create histogram of local gradient directions in the patch
  - Assign canonical orientation at peak of smoothed histogram



Slide by Svetlana Lazebnik

Locations + Scales + Orientations



D. Lowe, **Distinctive image features from scale-invariant keypoints**, *IJCV* 60 (2), pp. 91-110, 2004.

From keypoint detection to keypoint representation (feature descriptors)

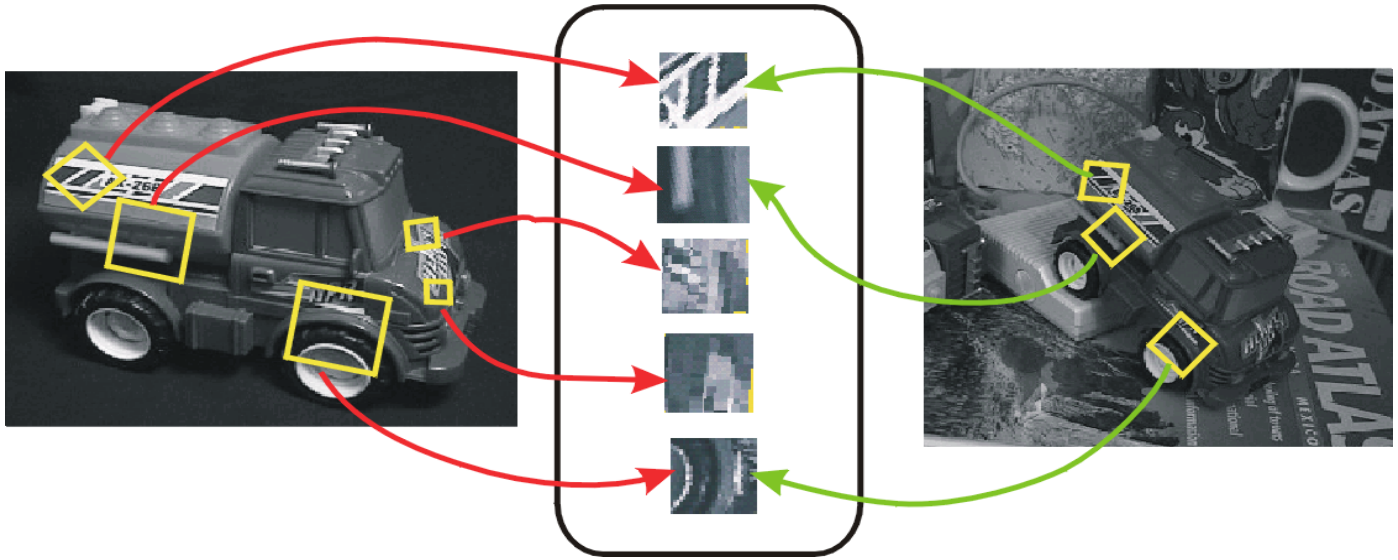
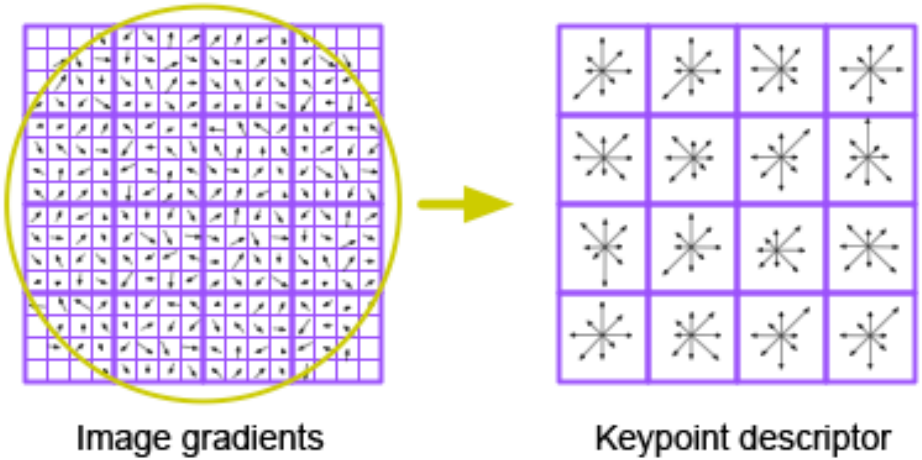


Figure by Svetlana Lazebnik

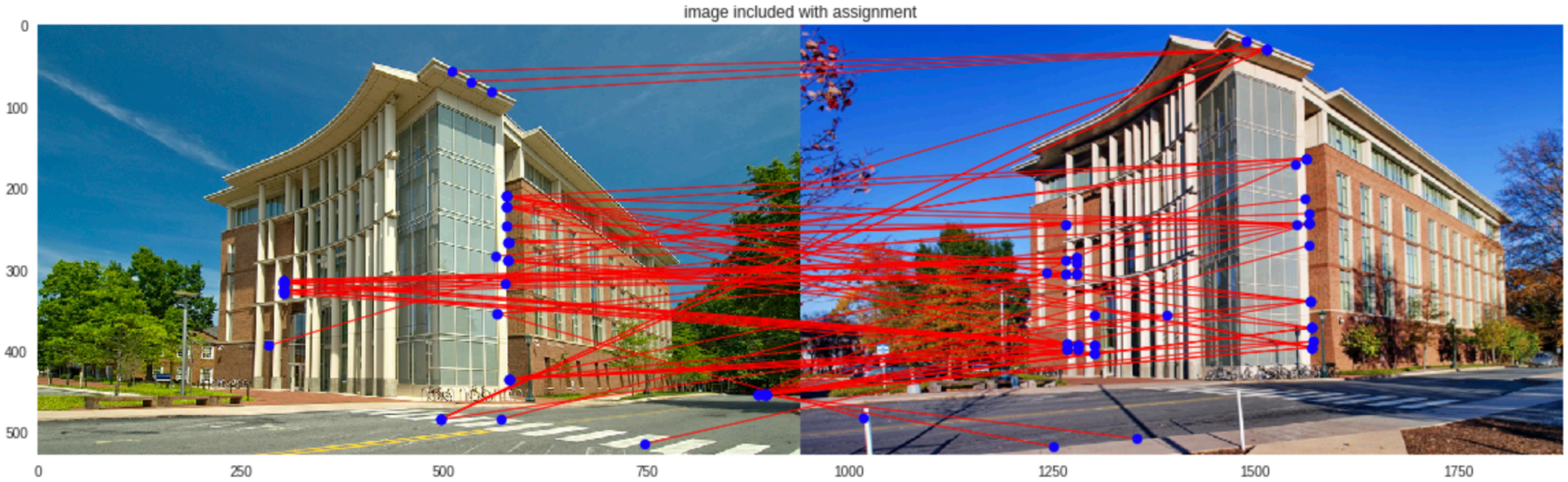
SIFT descriptors

- Inspiration: complex neurons in the primary visual cortex



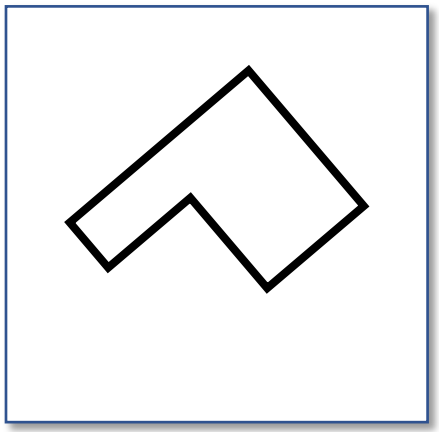
D. Lowe. **Distinctive image features from scale-invariant keypoints.** *IJCV* 60 (2), pp. 91-110, 2004.

SIFT Feature Matching

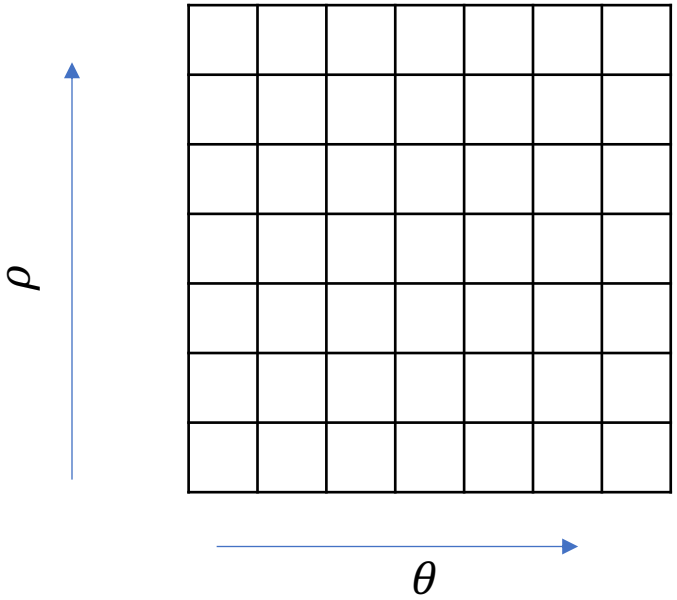


Rice Hall at UVA

Line Detection – Hough Transform



$$x \cos\theta + y \sin\theta = \rho$$

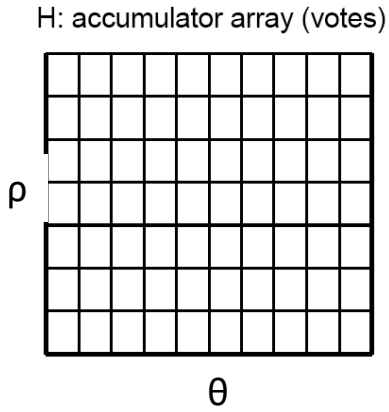


Count all points intersecting with all lines with rho = (-diagonal, diagonal), theta = [0, 180]

Hough Transform Algorithm outline

- Initialize accumulator H to all zeros

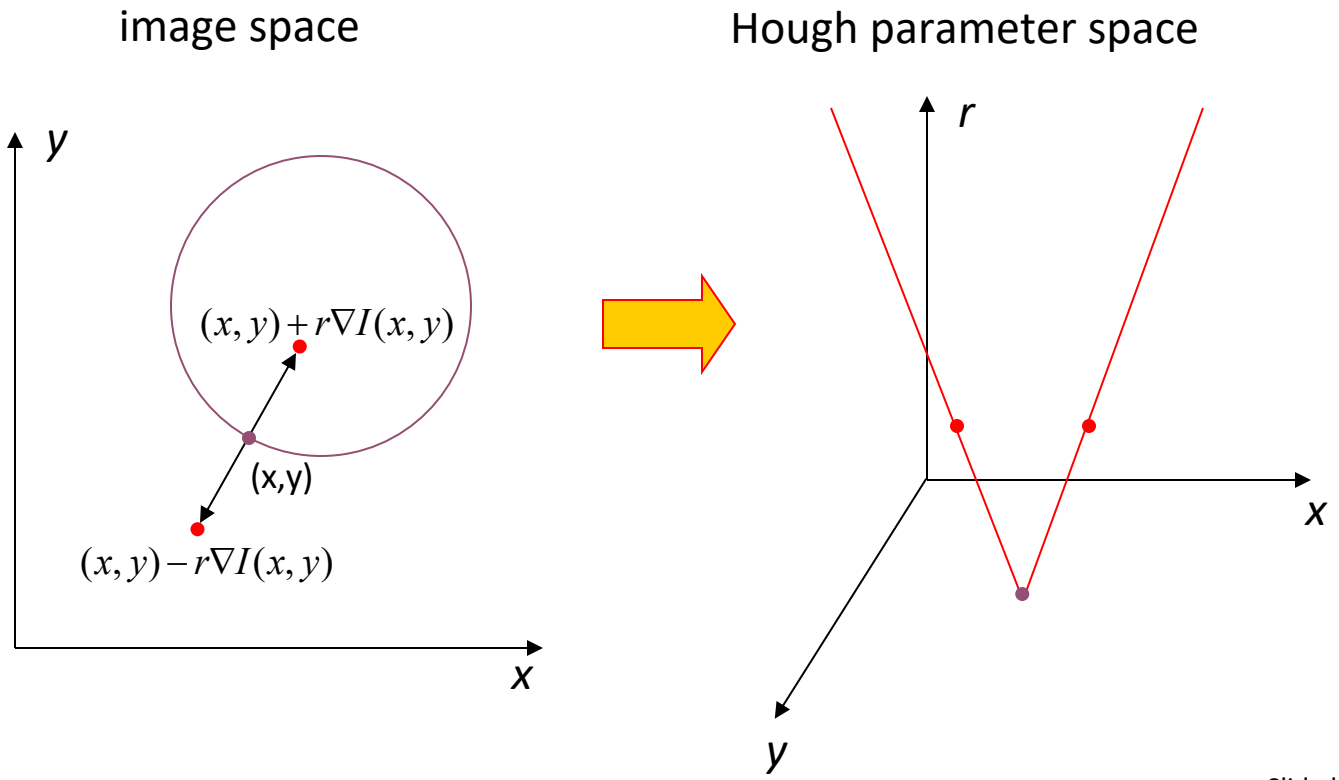
- For each feature point (x,y)
  in the image
    For θ = 0 to 180
      ρ = x cos θ + y sin θ
      H(θ, ρ) = H(θ, ρ) + 1
    end
  end



- Find the value(s) of (θ, ρ) where H(θ, ρ) is a local maximum
  - The detected line in the image is given by
    ρ = x cos θ + y sin θ

Slide by Svetlana Lazebnik

Hough transform for circles



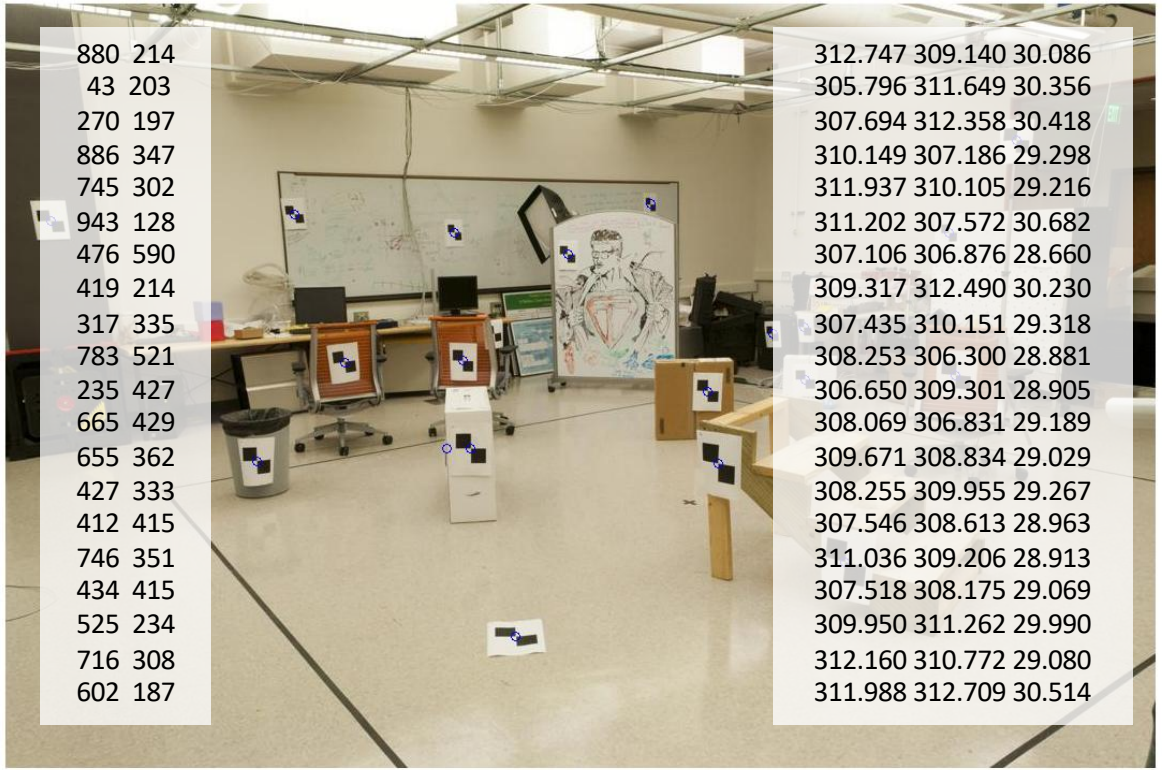
Slide by Svetlana Lazebnik

How do we calibrate a camera?

Slide Credit: James Hays

Known 2d image coords

880 214
43 203
270 197
886 347
745 302
943 128
476 590
419 214
317 335
783 521
235 427
665 429
655 362
427 333
412 415
746 351
434 415
525 234
716 308
602 187



Known 3d locations

312.747 309.140 30.086
305.796 311.649 30.356
307.694 312.358 30.418
310.149 307.186 29.298
311.937 310.105 29.216
311.202 307.572 30.682
307.106 306.876 28.660
309.317 312.490 30.230
307.435 310.151 29.318
308.253 306.300 28.881
306.650 309.301 28.905
308.069 306.831 29.189
309.671 308.834 29.029
308.255 309.955 29.267
307.546 308.613 28.963
311.036 309.206 28.913
307.518 308.175 29.069
309.950 311.262 29.990
312.160 310.772 29.080
311.988 312.709 30.514

Camera Calibration

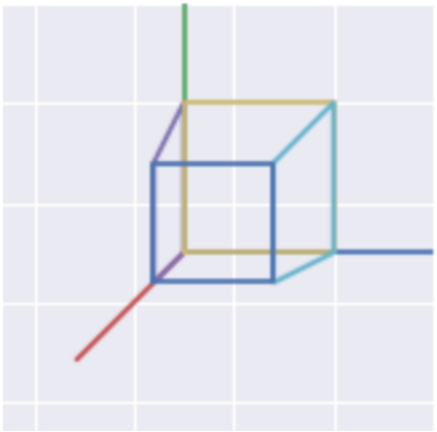
$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

$\mathbf{X} =$

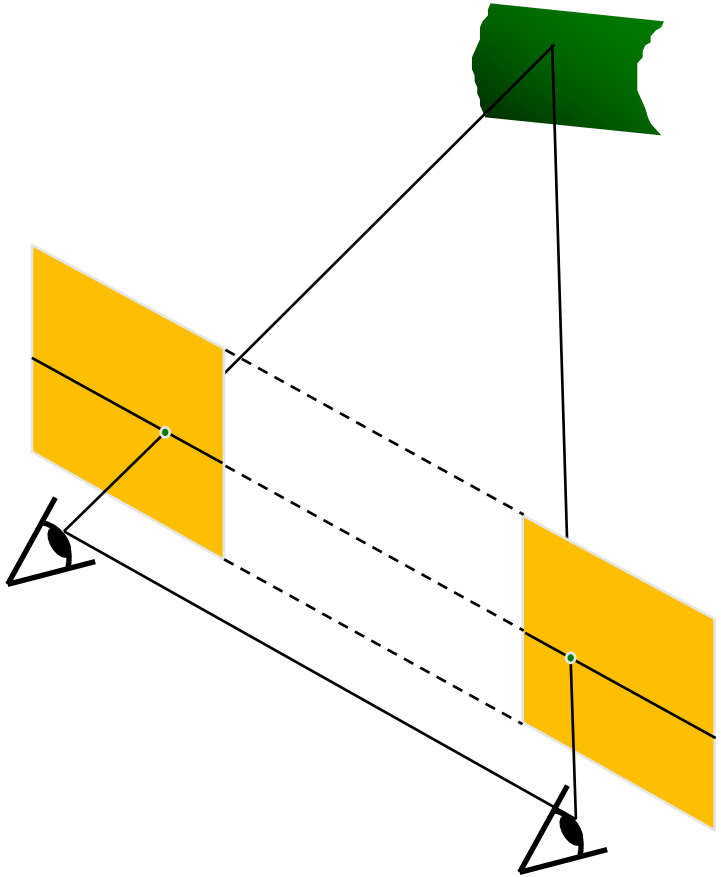
```
# Definition of the faces of the cube.
cube_pts = np.array(
                [[[0,0,0], [0,0,1], [0,1,1], [0,1,0], [0,0,0]],   # Face 1.
                 [[0,0,0], [0,1,0], [1,1,0], [1,0,0], [0,0,0]],   # Face 2.
                 [[1,0,0], [1,0,1], [1,1,1], [1,1,0], [1,0,0]],   # Face 3.
                 [[0,0,1], [0,1,1], [1,1,1], [1,0,1], [0,0,1]]])  # Face 4.
```

Goal: Find $\mathbf{K}[\mathbf{R}\ \mathbf{t}]$

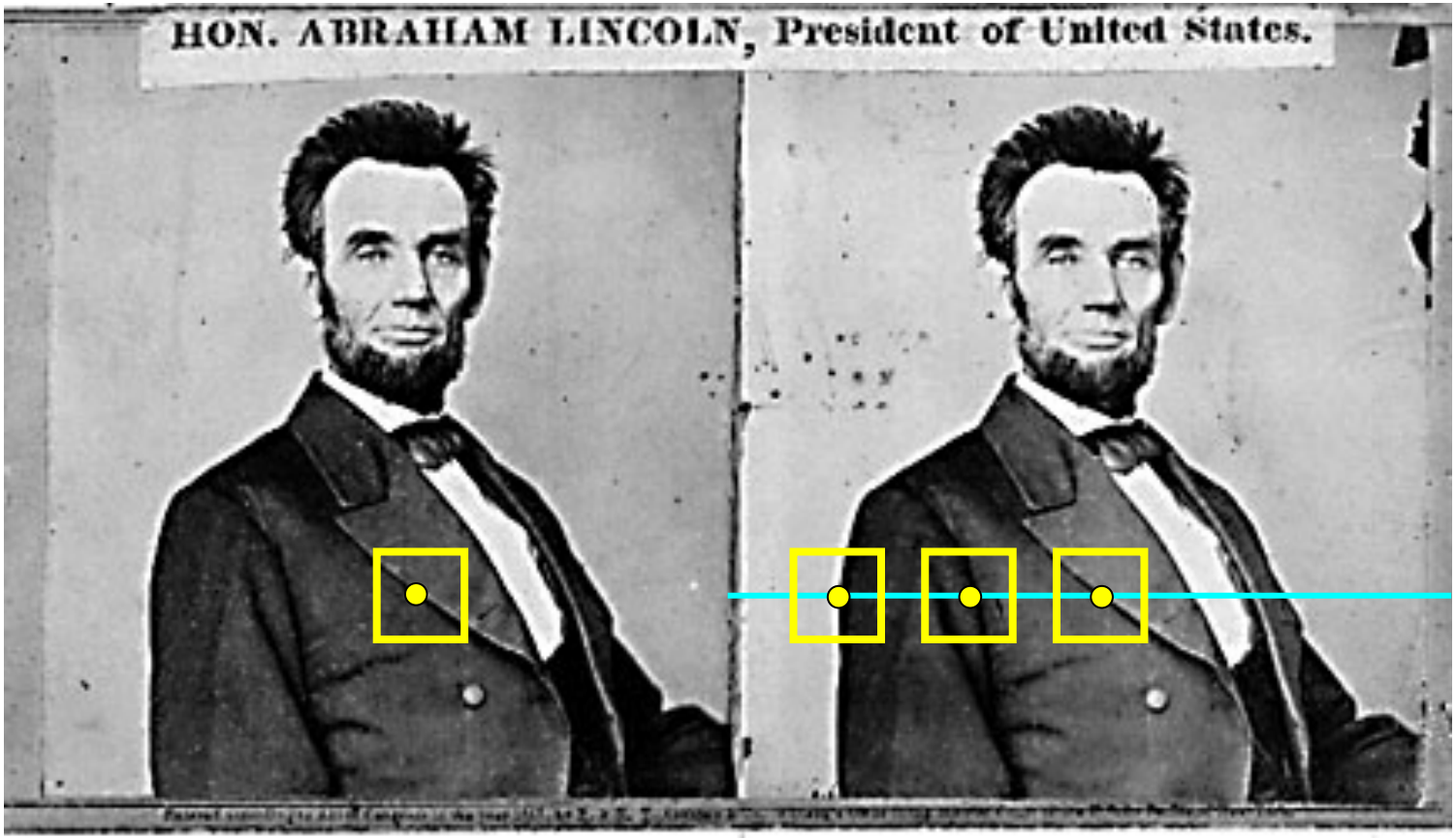
$\mathbf{x} =$

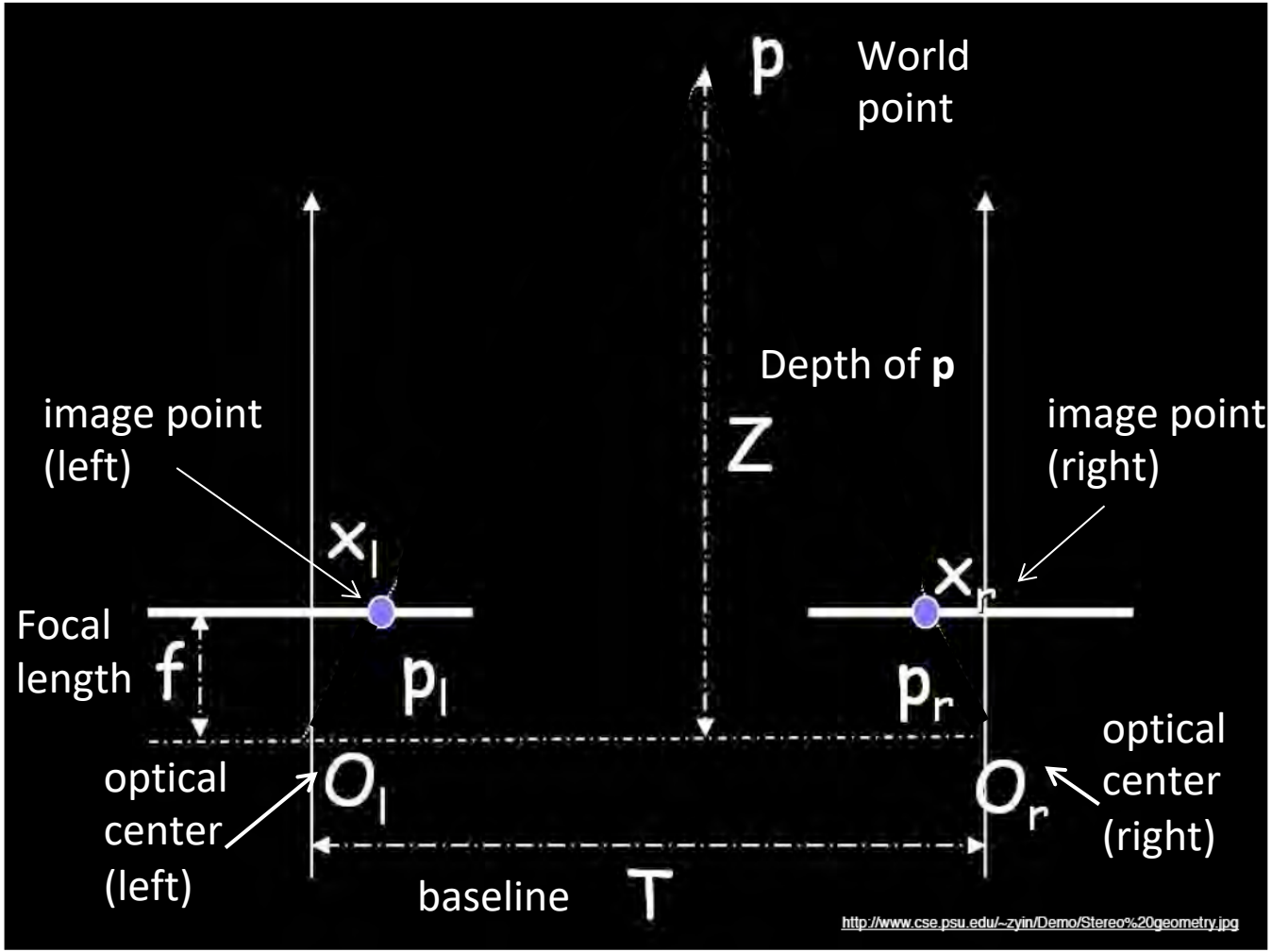


Stereo Vision



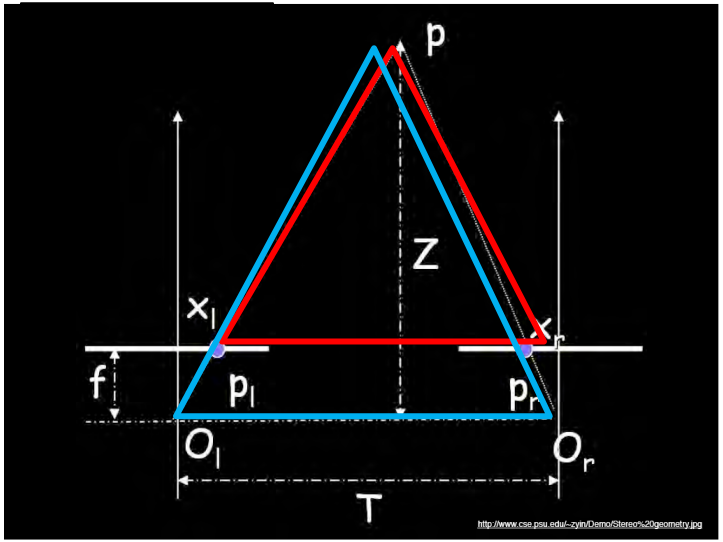
- Image planes of cameras are parallel to each other and to the baseline
- Camera centers are at same height
- Focal lengths are the same
- Then epipolar lines fall along the horizontal scan lines of the images


HON. ABRAHAM LINCOLN, President of United States.


p
World point
Depth of p
Z
image point (left)
image point (right)
x_l
x_r
Focal length
f
p_l
p_r
optical center (left)
O_l
O_r
optical center (right)
baseline
T
http://www.cse.psu.edu/~zyin/Demo/Stereo%20geometry.jpg

Geometry for a simple stereo system

- Assume parallel optical axes, known camera parameters (i.e., calibrated cameras). **What is expression for Z?**



Similar triangles (p_l, P, p_r) and (O_l, P, O_r):

$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

$$Z = f\frac{T}{x_r - x_l}$$

disparity

Depth from disparity

image I(x,y)

Disparity map D(x,y)

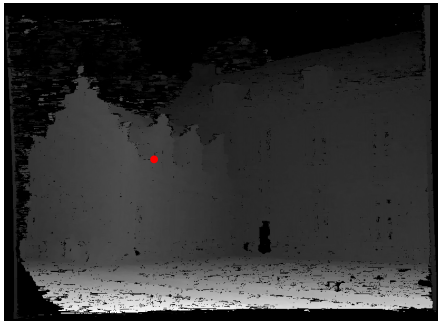
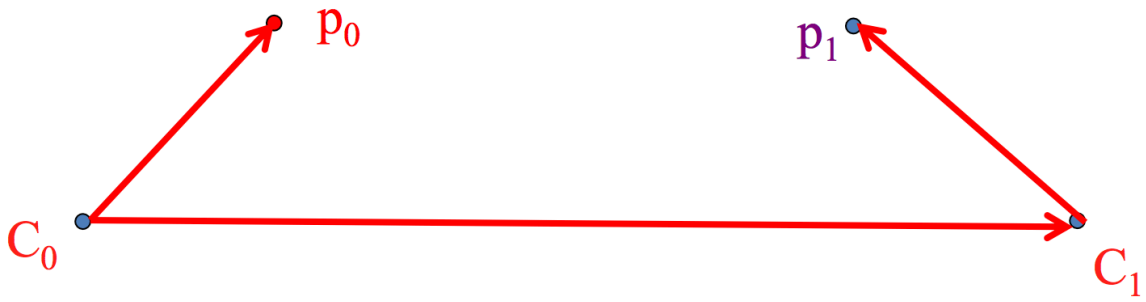


image I´(x´,y´)

$(x´,y´)=(x+D(x,y), y)$

So if we could find the **corresponding points** in two images, we could **estimate relative depth**…

Back to the General Problem

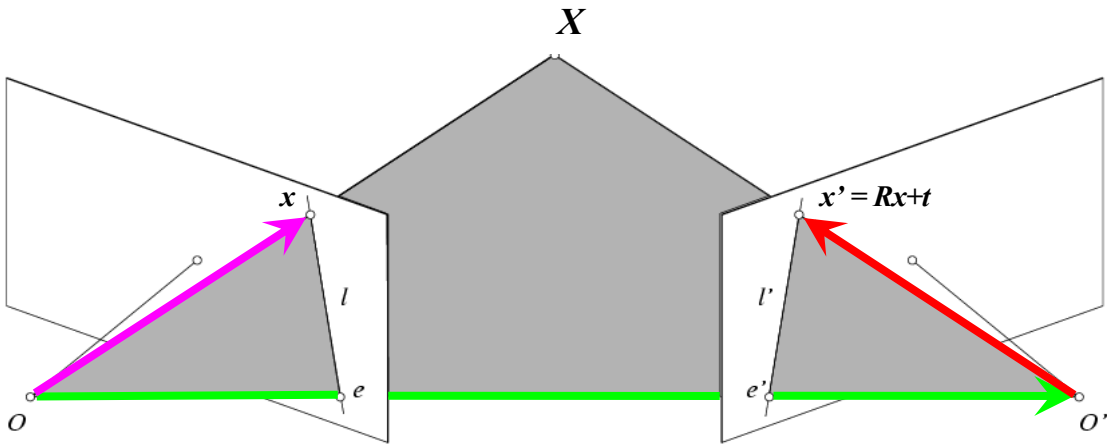


- Another way to write the fact they are co-planar is

$$\overrightarrow{\mathbf{C}_0\mathbf{p}_0} \cdot \left(\overrightarrow{\mathbf{C}_0\mathbf{C}_1} \times \overrightarrow{\mathbf{C}_1\mathbf{p}_1}\right) = 0$$

Credit: William Hoff, Colorado School of Mines

Epipolar constraint: Calibrated case

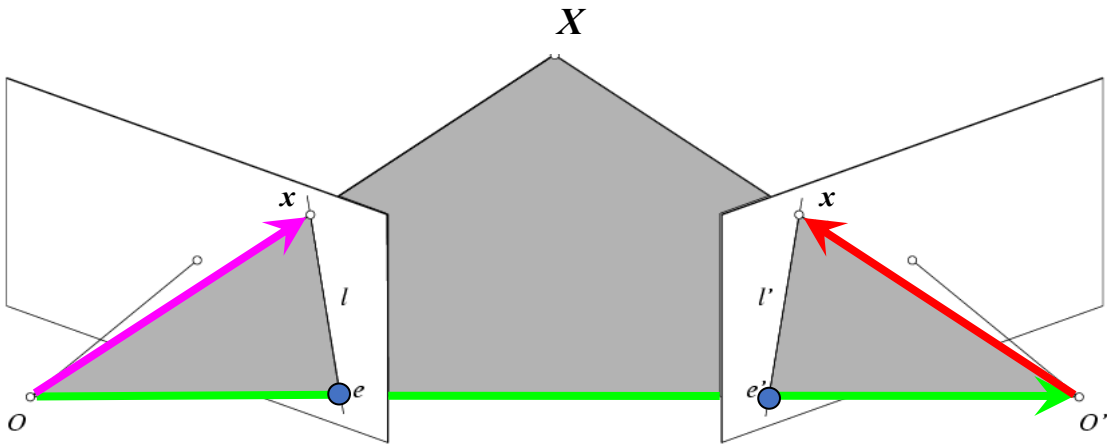


$$\boldsymbol{x}' \cdot [\boldsymbol{t} \times (\boldsymbol{R}\boldsymbol{x})] = 0 \quad \Rightarrow \quad \boldsymbol{x}'^{T} [\boldsymbol{t}_{\times}] \boldsymbol{R}\boldsymbol{x} = 0 \quad \Rightarrow \quad \boldsymbol{x}'^{T} \boldsymbol{E}\boldsymbol{x} = 0$$

Essential Matrix
(Longuet-Higgins, 1981)

The vectors ***Rx***, ***t***, and ***x'*** are coplanar

Epipolar constraint: Uncalibrated case



$$\hat{x}'^T E \hat{x} = 0 \quad \Rightarrow \quad x'^T F x = 0 \quad \text{with} \quad F = K'^{-T} E K^{-1}$$

$$\hat{x} = K^{-1} x$$

$$\hat{x}' = K'^{-1} x'$$

Fundamental Matrix
(Faugeras and Luong, 1992)

Supervised Learning –Softmax Classifier

Extract features

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$

Run features through classifier

$$g_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c$$

$$g_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d$$

$$g_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b$$

$$f_c = e^{g_c}/(e^{g_c}+e^{g_d} + e^{g_b})$$

$$f_d = e^{g_d}/(e^{g_c}+e^{g_d} + e^{g_b})$$

$$f_b = e^{g_b}/(e^{g_c}+e^{g_d} + e^{g_b})$$

Get predictions

$\hat{y}_i = [f_c \quad f_d \quad f_b]$

(mini-batch) Stochastic Gradient Descent (SGD)

$\lambda = 0.01$

Initialize w and b randomly

$$l(w,b) = \sum_{i \in B} -\log f_{i,label}(w,b)$$

for e = 0, num_epochs **do**

for b = 0, num_batches **do**

Compute: $dl(w,b)/dw$ and $dl(w,b)/db$

Update w: $w = w - \lambda\, dl(w,b)/dw$

Update b: $b = b - \lambda\, dl(w,b)/db$

Print: $l(w,b)$ // Useful to see if this is becoming smaller or not.

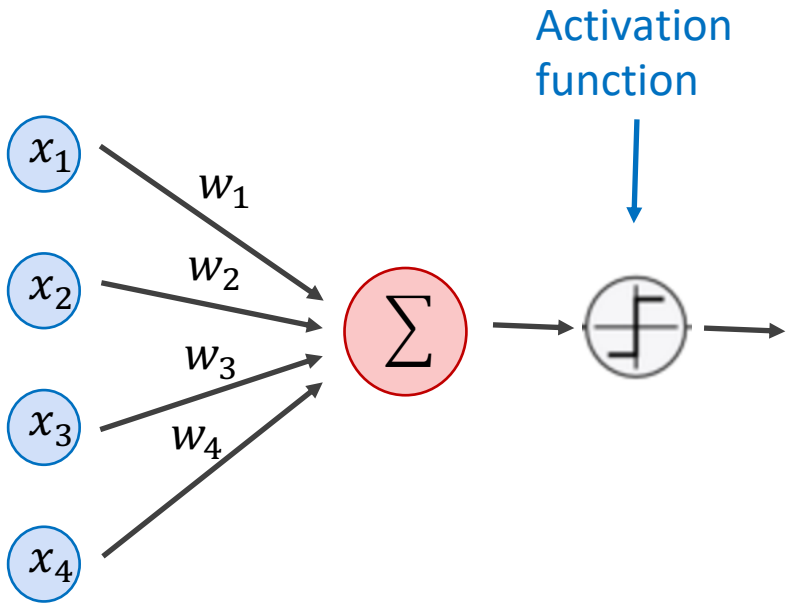
end

end

Perceptron Model

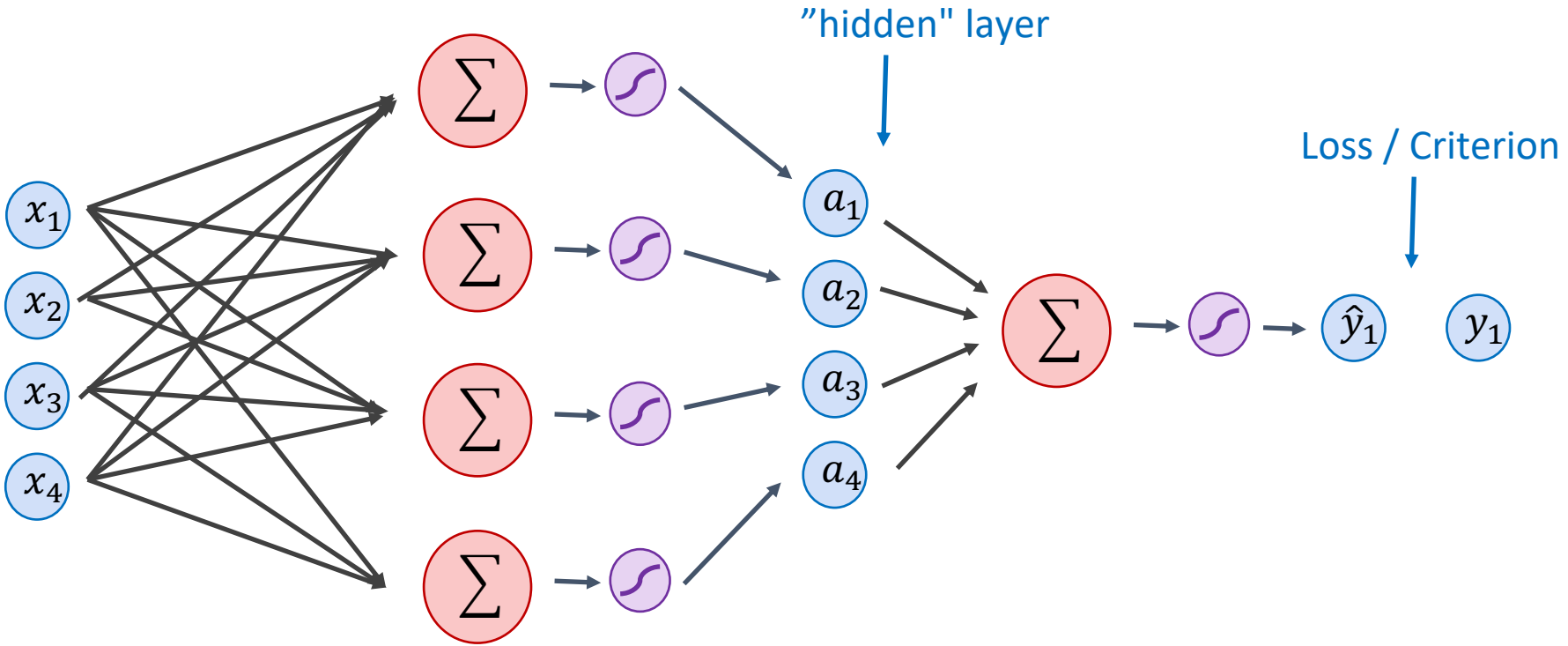
Frank Rosenblatt (1957) - Cornell University

$$f(x) = \begin{cases} 1, & \text{if } \sum_{i=0}^{n} w_i x_i + b > 0 \\ 0, & \text{otherwise} \end{cases}$$

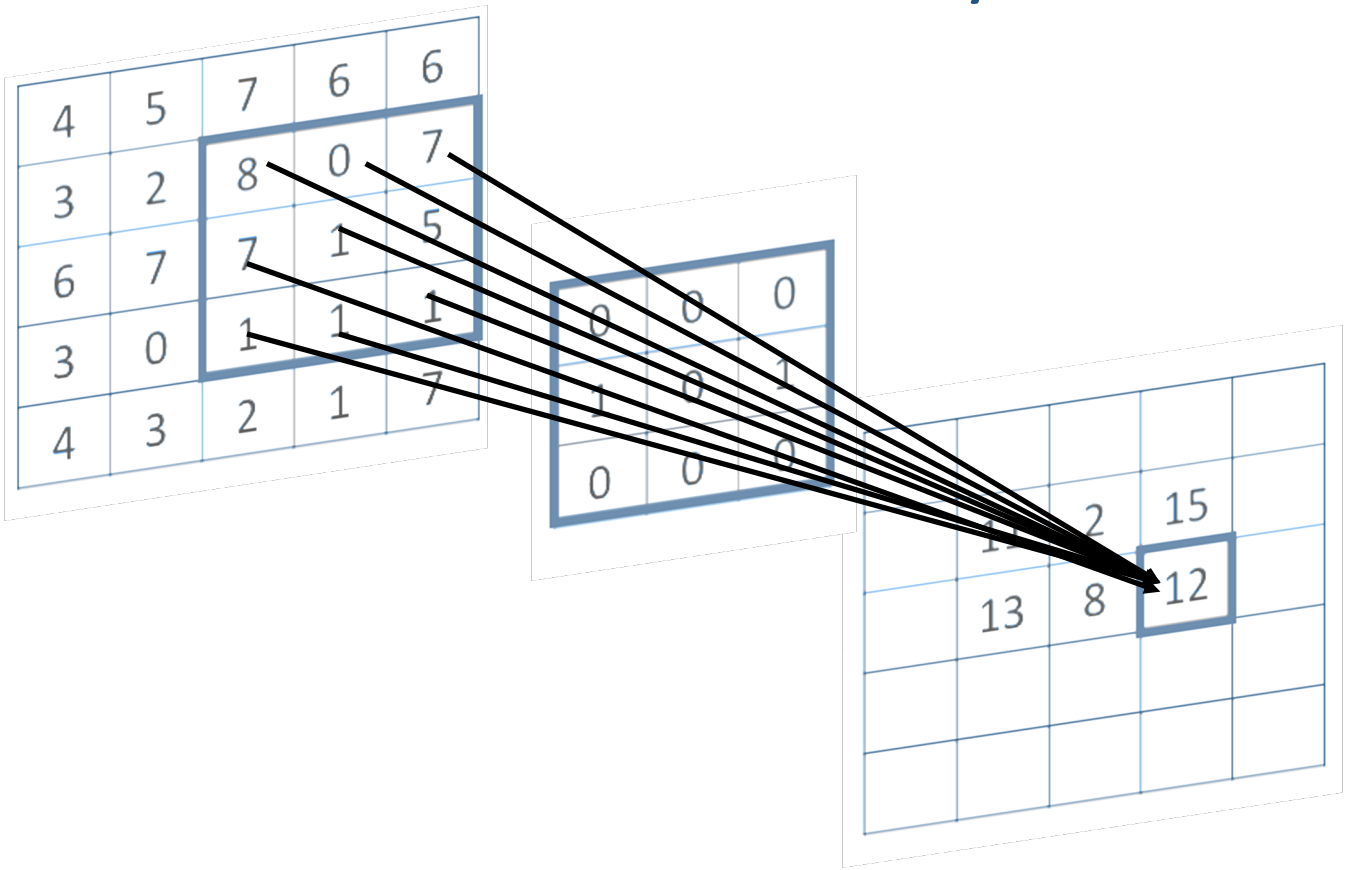


More: https://en.wikipedia.org/wiki/Perceptron

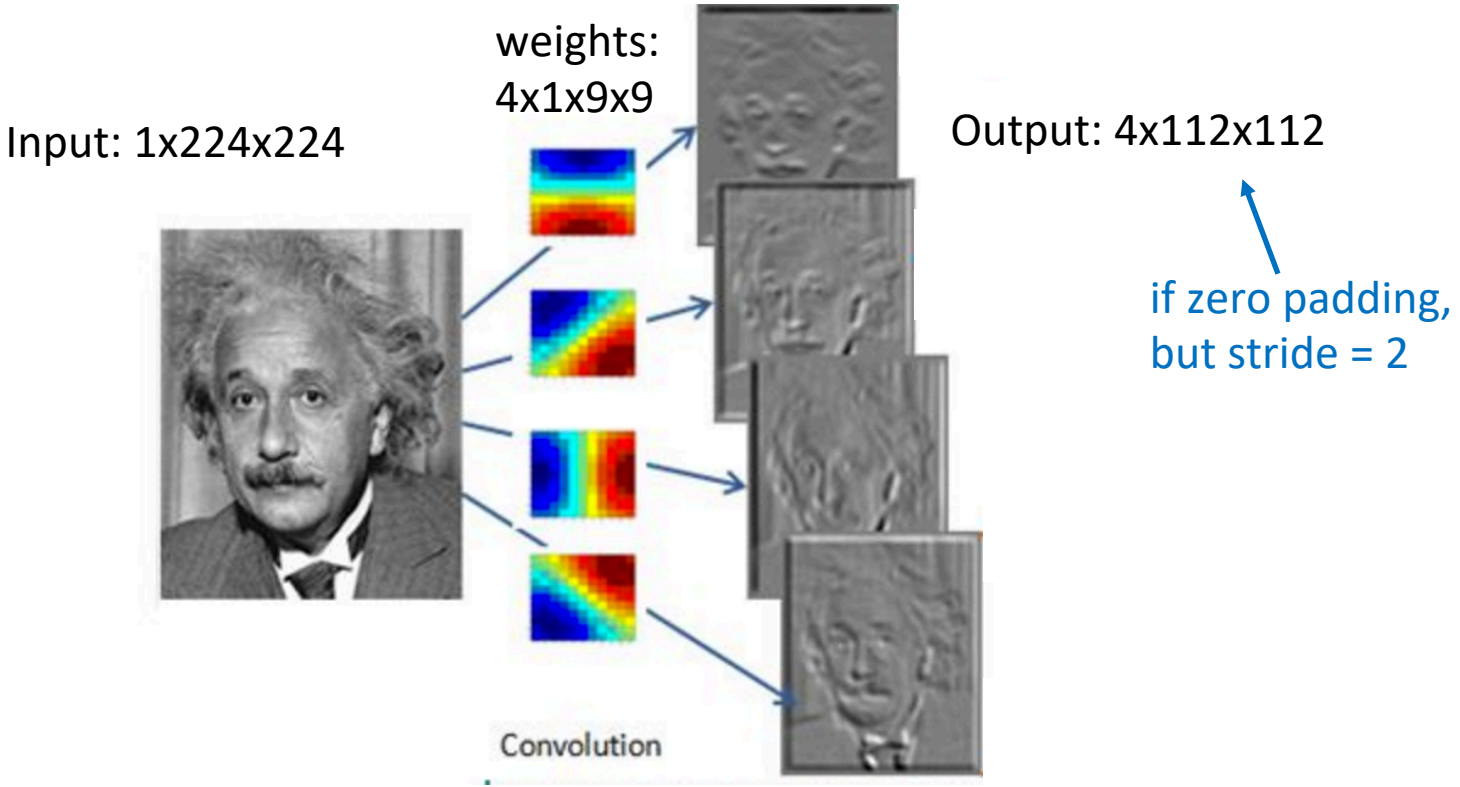
Two-layer Multi-layer Perceptron (MLP)



Convolutional Layer

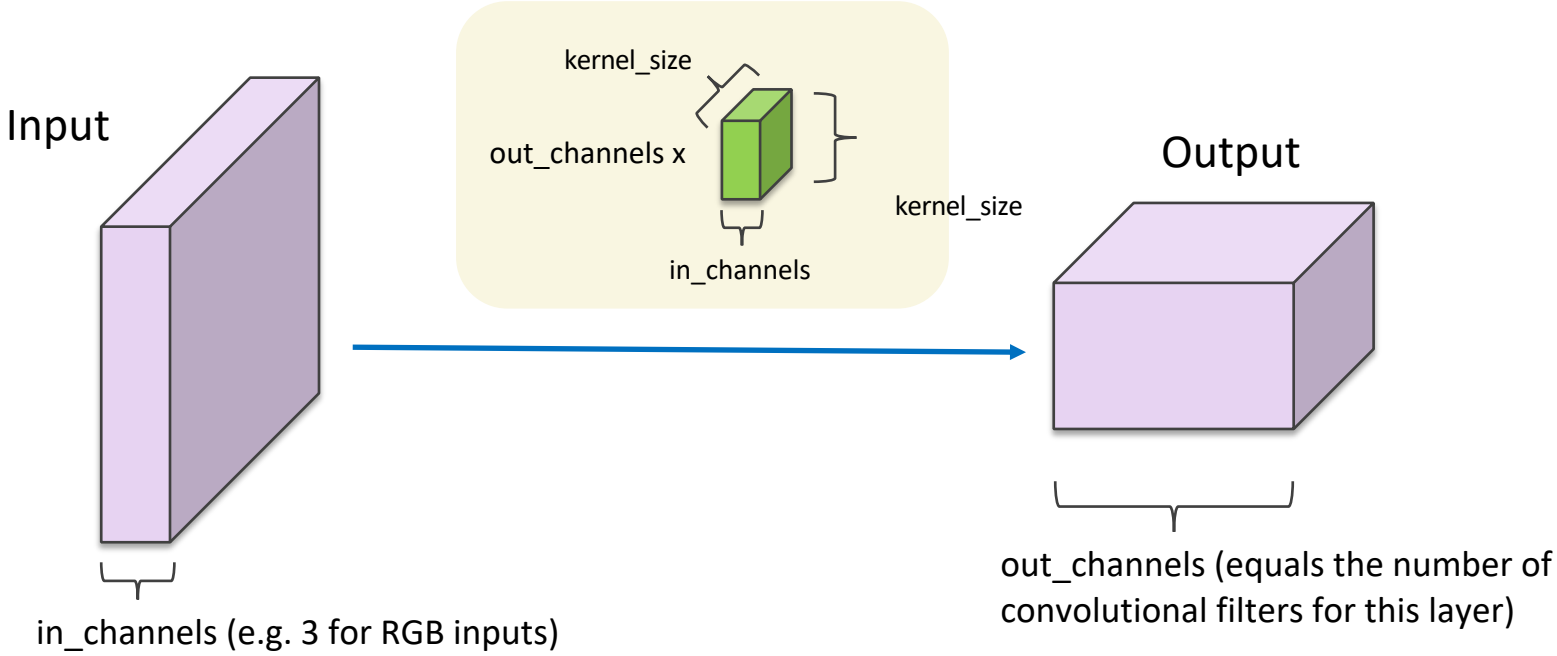


Convolutional Layer (with 4 filters)

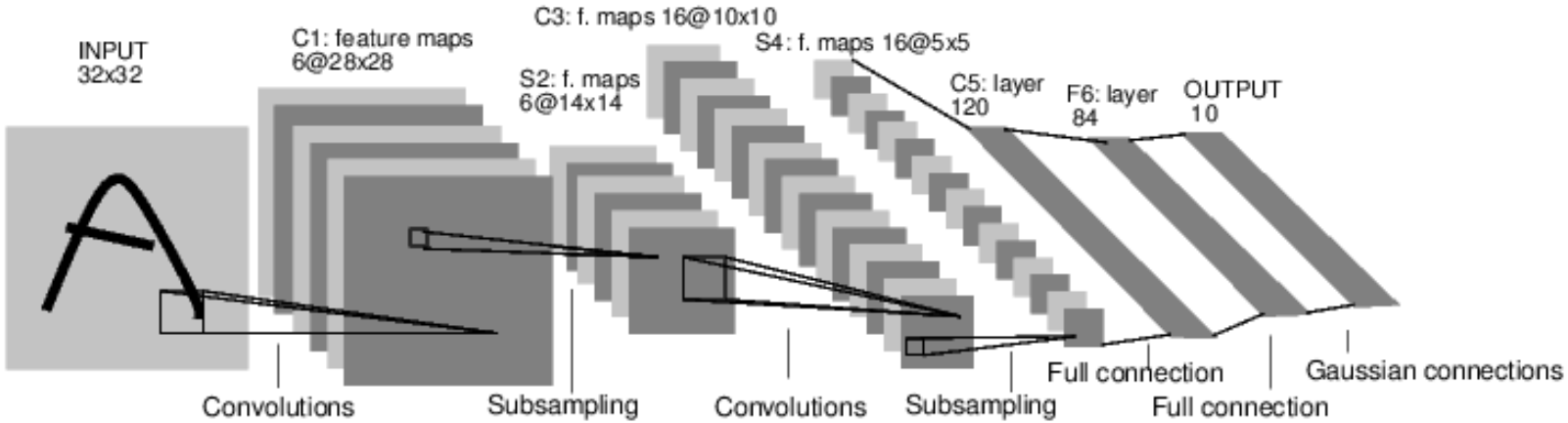


Convolutional Layer in pytorch

```
class torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1,
groups=1, bias=True)  [source]
```



Convolutional Network: LeNet



Yann LeCun

TITLE	CITED BY	YEAR
Gradient-based learning applied to document recognition Y LeCun, L Bottou, Y Bengio, P Haffner Proceedings of the IEEE 86 (11), 2278-2324	11736	1998

LeNet in Pytorch

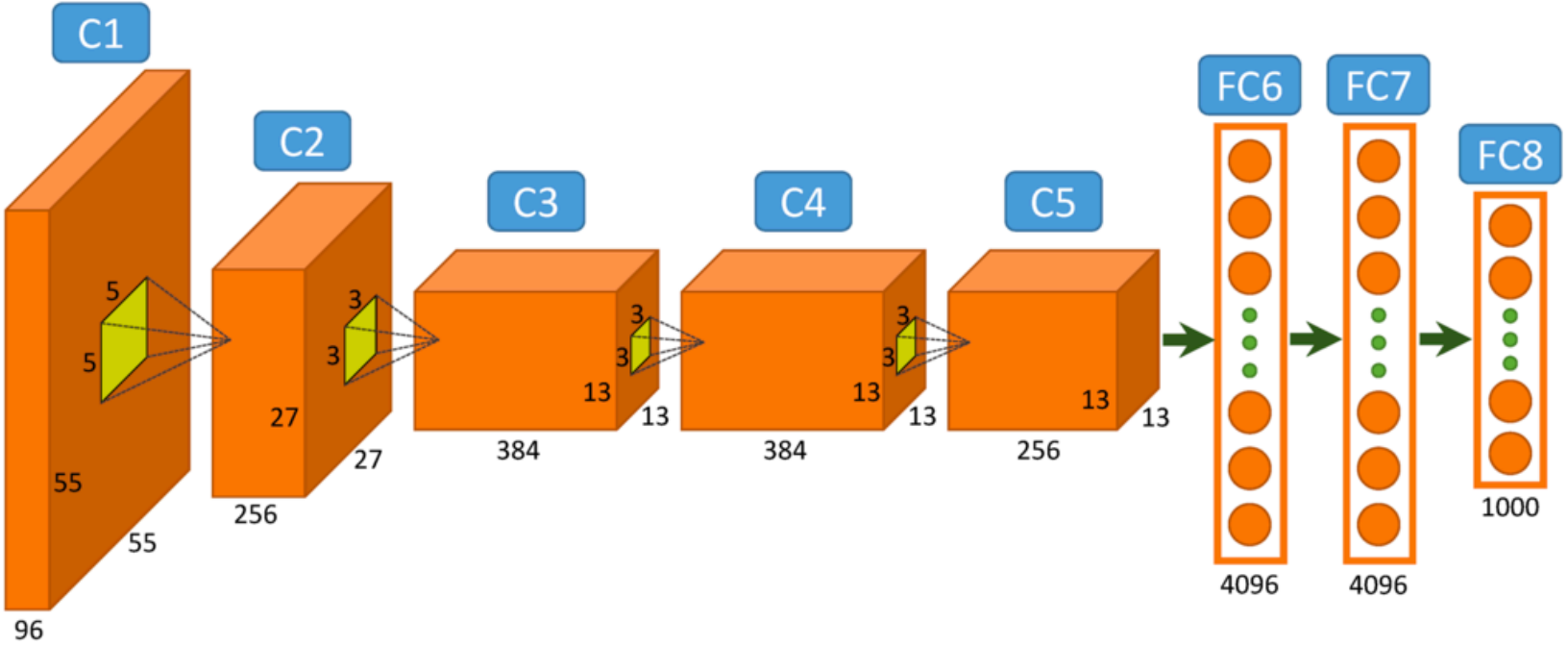
```
# LeNet is French for The Network, and is taken from Yann Lecun's 98 paper
# on digit classification http://yann.lecun.com/exdb/lenet/
# This was also a network with just two convolutional layers.
class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        # Convolutional layers.
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)

        # Linear layers.
        self.fc1 = nn.Linear(16*5*5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        out = F.relu(self.conv1(x))
        out = F.max_pool2d(out, 2)
        out = F.relu(self.conv2(out))
        out = F.max_pool2d(out, 2)

        # This flattens the output of the previous layer into a vector.
        out = out.view(out.size(0), -1)
        out = F.relu(self.fc1(out))
        out = F.relu(self.fc2(out))
        out = self.fc3(out)
        return out
```

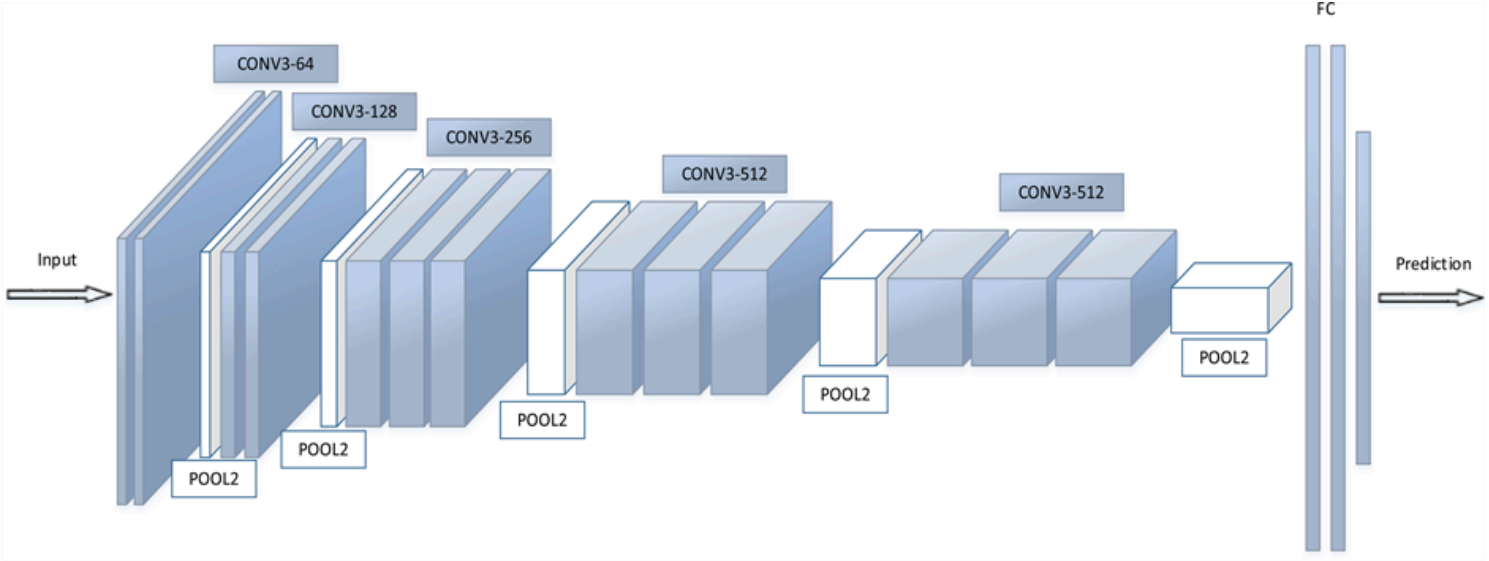
Alexnet



https://www.saagie.com/fr/blog/object-detection-part1

VGG Network

Top-5:

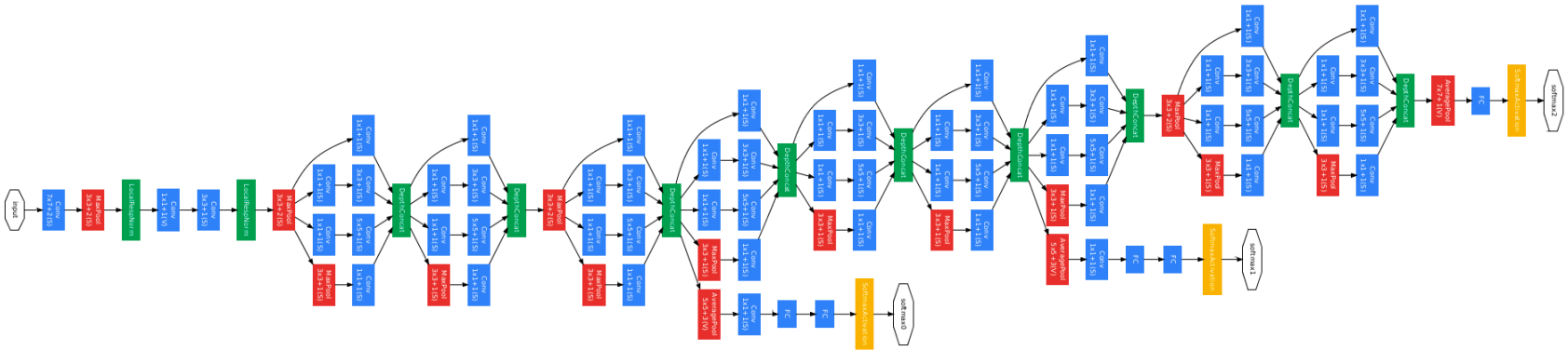


https://github.com/pytorch/vision/blob/master/torchvision/models/vgg.py

Simonyan and Zisserman, 2014.

https://arxiv.org/pdf/1409.1556.pdf

GoogLeNet

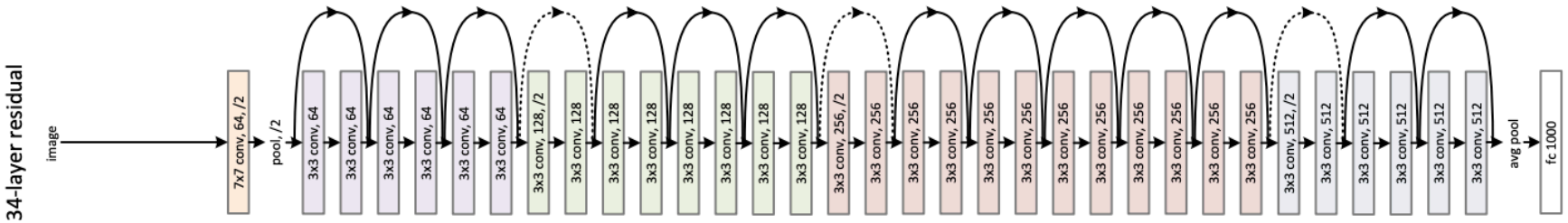


https://github.com/kuangliu/pytorch-cifar/blob/master/models/googlenet.py

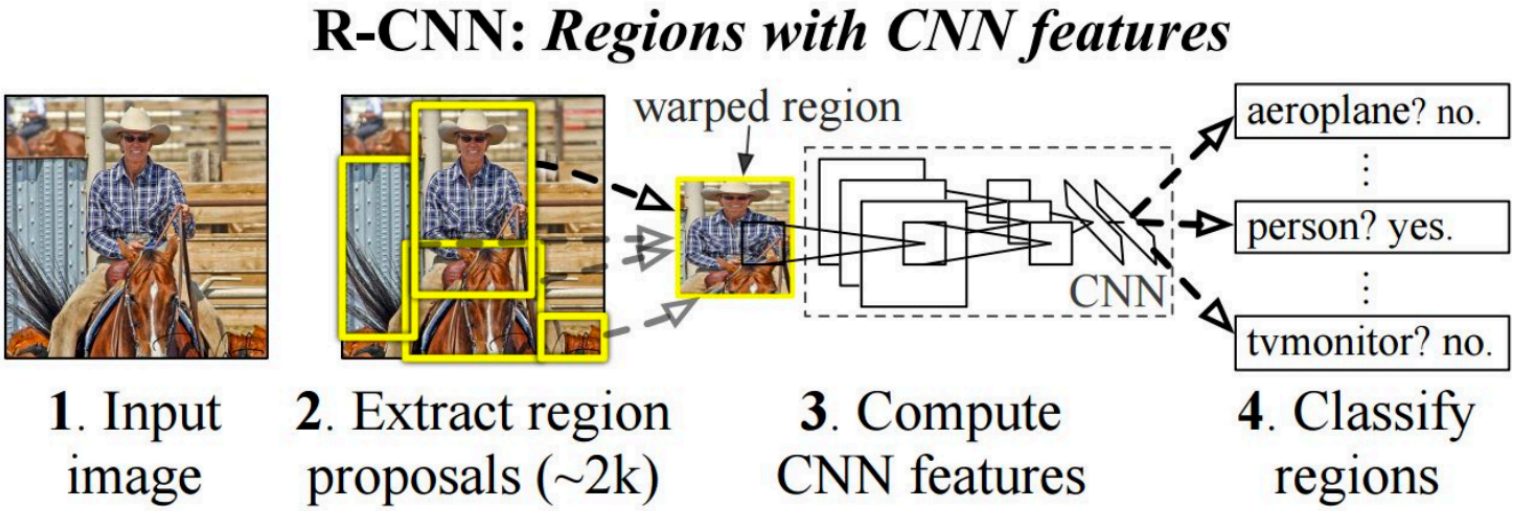
Szegedy et al. 2014

https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf

ResNet


34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000

RCNN

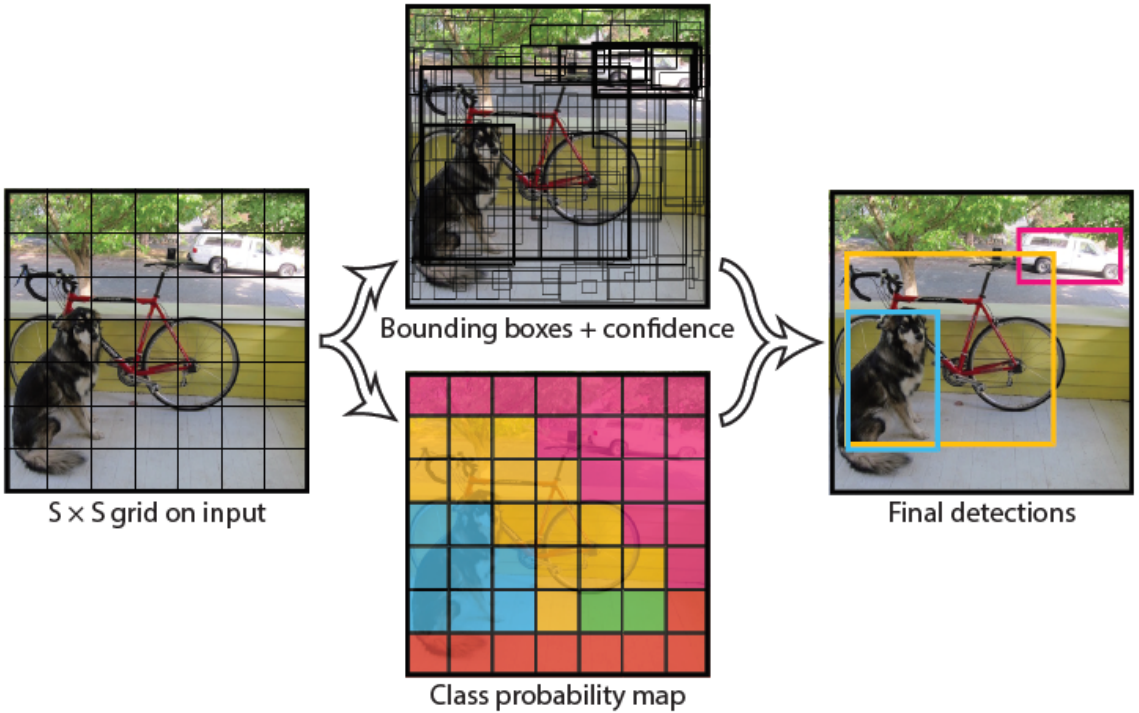


https://people.eecs.berkeley.edu/~rbg/papers/r-cnn-cvpr.pdf
Rich feature hierarchies for accurate object detection and semantic segmentation.
Girshick et al. CVPR 2014.

YOLO- You Only Look Once

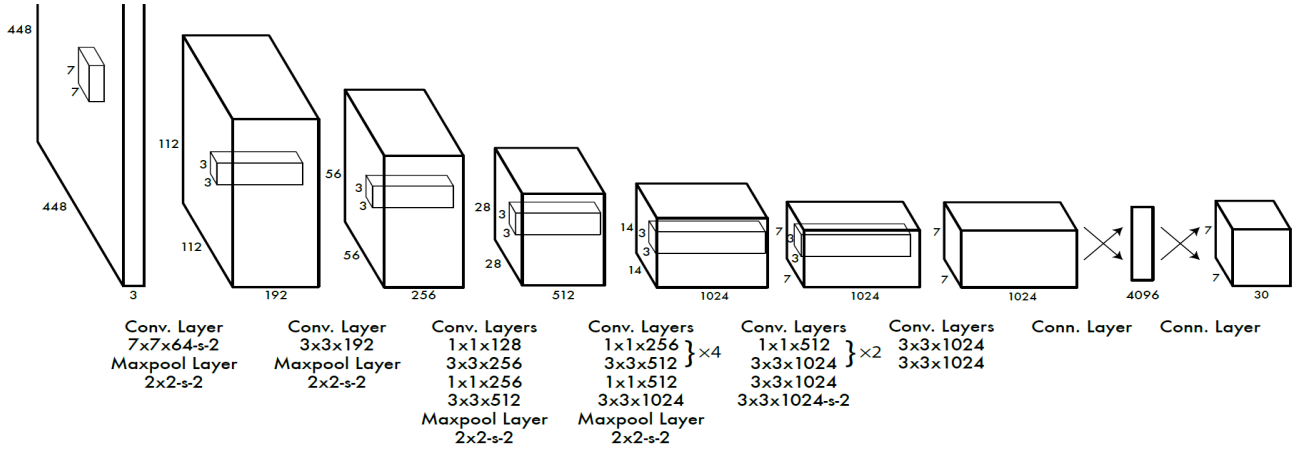
Idea: No bounding box proposal. A single regression problem, straight from image pixels to bounding box coordinates and class probabilities.

- extremely fast
- reason globally
- learn generalizable representations



https://arxiv.org/abs/1506.02640
Redmon et al. CVPR 2016.

YOLO- You Only Look Once



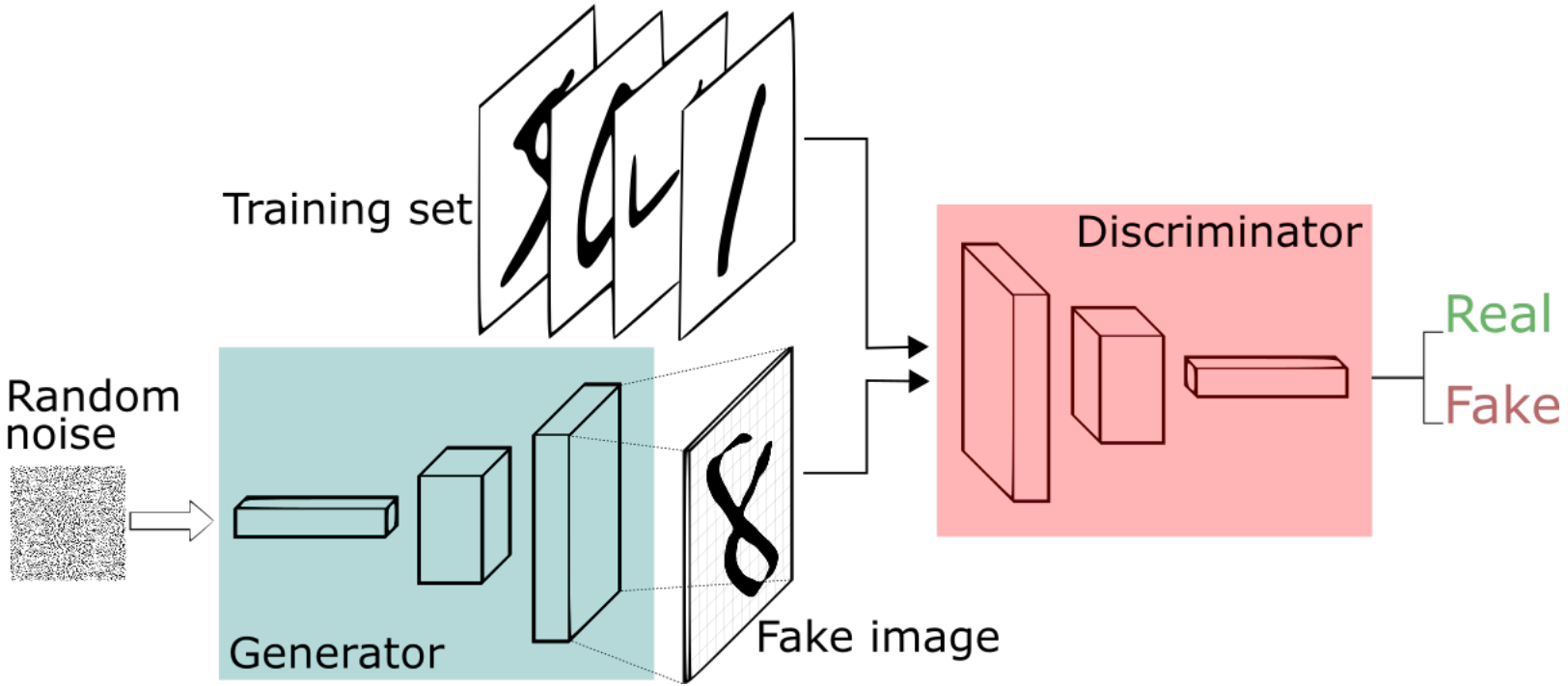
Divide the image into 7x7 cells.

Each cell trains a detector.

The detector needs to predict the object's class distributions.

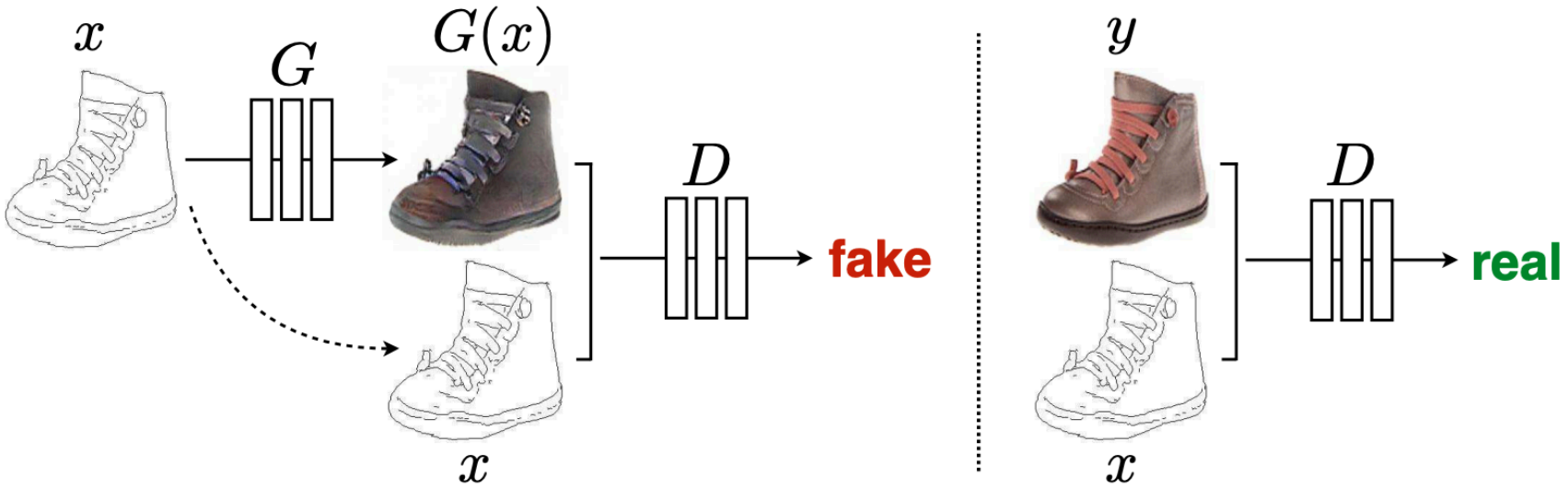
The detector has 2 bounding-box predictors to predict bounding-boxes and confidence scores.

Generative Adversarial Networks (GAN) [Goodfellow et al.]



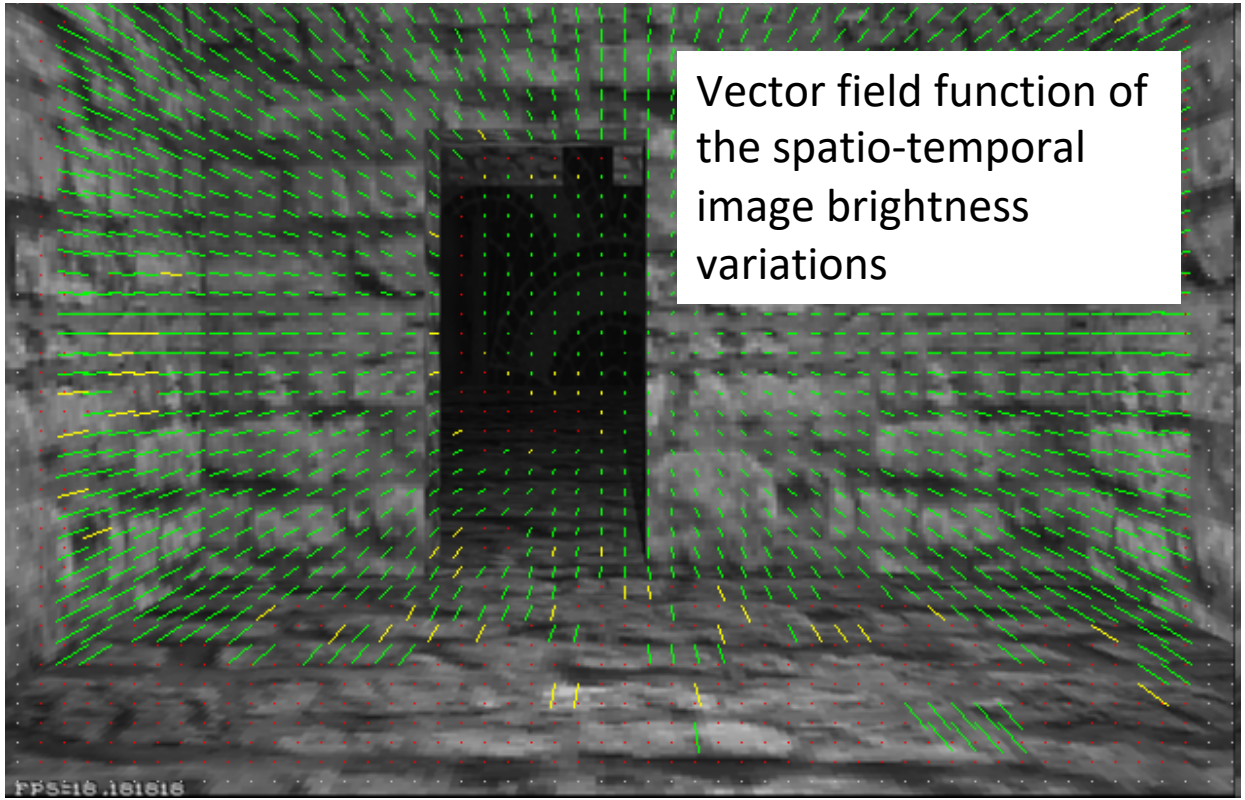
https://deeplearning4j.org/generative-adversarial-network

Conditional GANs: Input is not just Noise



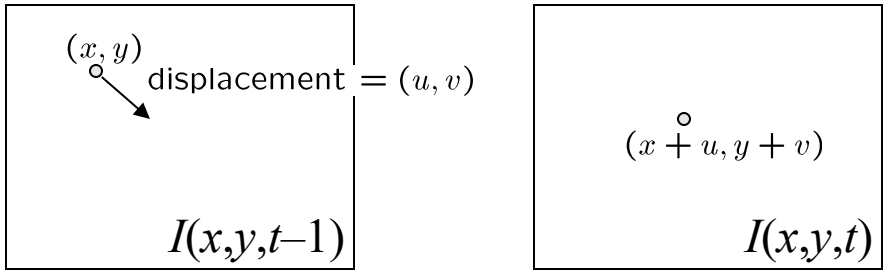
Isola et al. CVPR 2017: Image-to-Image Translation with Conditional Adversarial Networks

Optical flow



Picture courtesy of Selim Temizer - Learning and Intelligent Systems (LIS) Group, MIT

The brightness constancy constraint



- Brightness Constancy Equation:

$$I(x,y,t-1) = I(x+u(x,y), y+v(x,y), t)$$

Linearizing the right side using Taylor expansion:

Image derivative along x

$$I(x+u,y+v,t) \approx I(x,y,t-1) + I_x \cdot u(x,y) + I_y \cdot v(x,y) + I_t$$

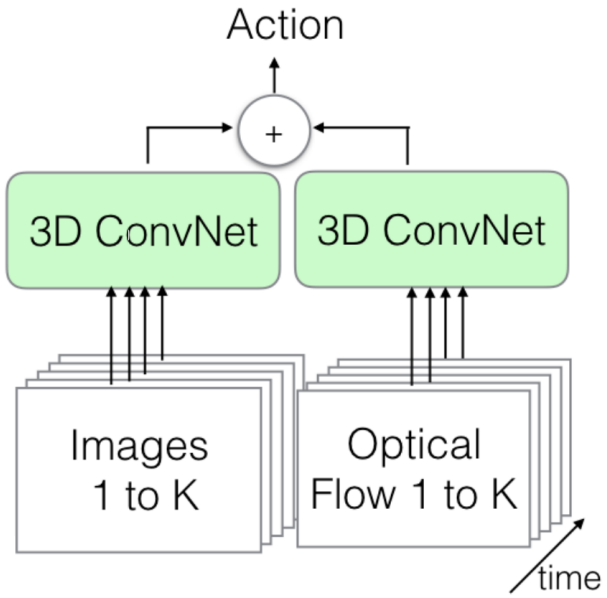
$$I(x+u,y+v,t) - I(x,y,t-1) = I_x \cdot u(x,y) + I_y \cdot v(x,y) + I_t$$

Hence, $I_x \cdot u + I_y \cdot v + I_t \approx 0 \quad \rightarrow \nabla I \cdot \begin{bmatrix} u & v \end{bmatrix}^T + I_t = 0$

Source: Silvio Savarese

Action Classification from Video

Two Stream 3D CNN: Images + Flow Map



Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset

João Carreira† joaoluis@google.com
Andrew Zisserman†,* zisserman@google.com
†DeepMind *Department of Engineering Science, University of Oxford

Figure from Carreira & Zisserman, 2018

Introduction to Computer Vision

Foundations

- Cameras
- Human Vision
- Image Processing
- Image Filtering
- Edge Detection
- Corner Detection
- Line Detection
- Interest Points

Geometry

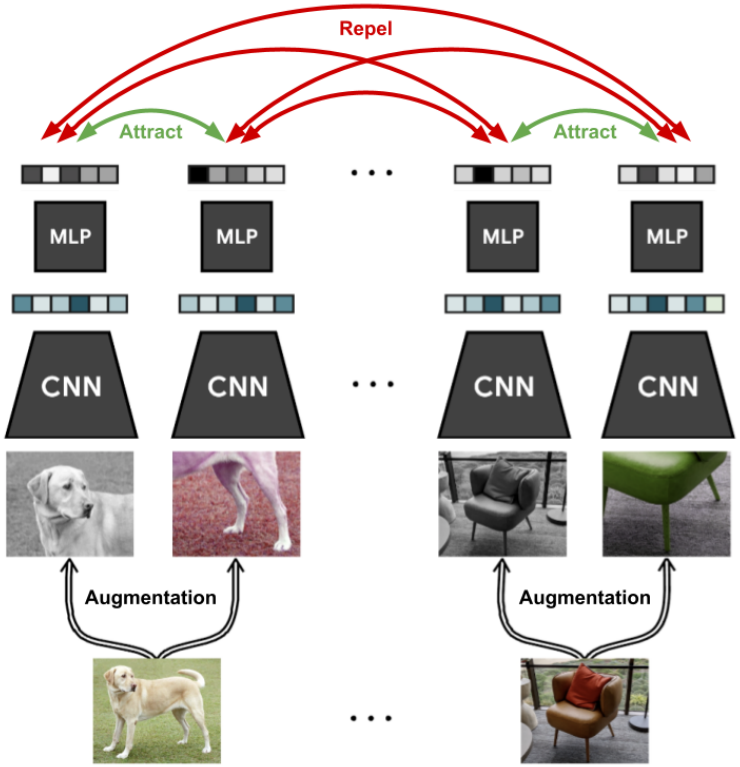
- Point Matching
- RANSAC
- Homography Estimation
- Image Stitching
- Camera Calibration
- Dense Stereo
- Epipolar Geometry

Deep Learning

- Machine Learning
- Neural Networks
- Classification / Regression
- Object Detection
- Image Segmentation
- Generative Adversarial Networks
- Video and Optical Flow

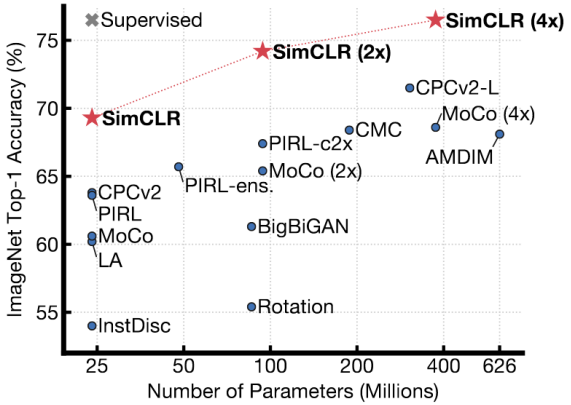
The Future?

Learning without Labeled Images
“Self-supervised”



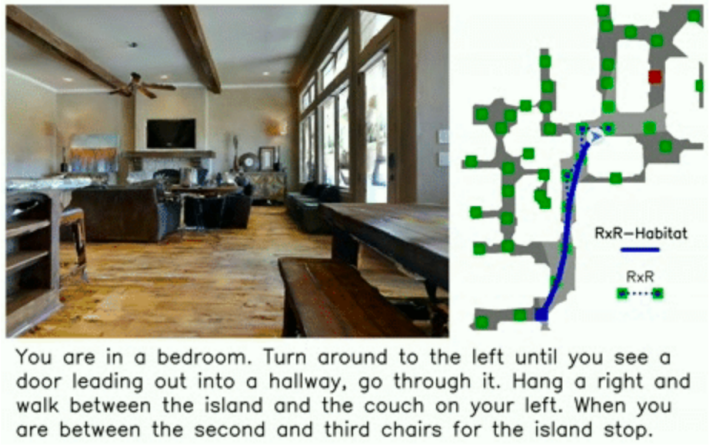
A Simple Framework for Contrastive Learning of Visual Representations

Ting Chen [1] Simon Kornblith [1] Mohammad Norouzi [1] Geoffrey Hinton [1]



Learning from Interactions

Learning from Interactions and Language



RxR-Habitat Competition

ALFRED Challenge

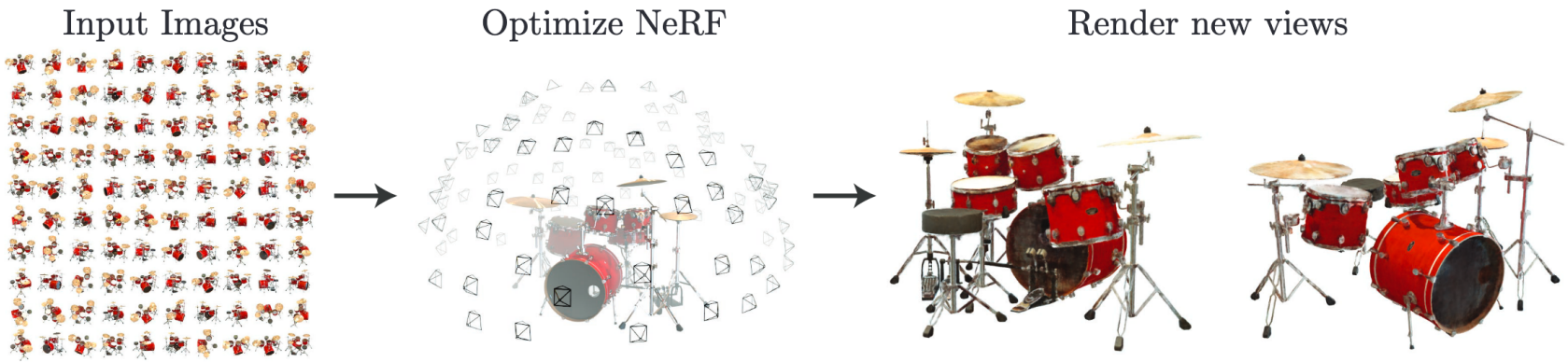
Machine Learning and 3D

NeRF

Representing Scenes as Neural Radiance Fields for View Synthesis

ECCV 2020 Oral - Best Paper Honorable Mention

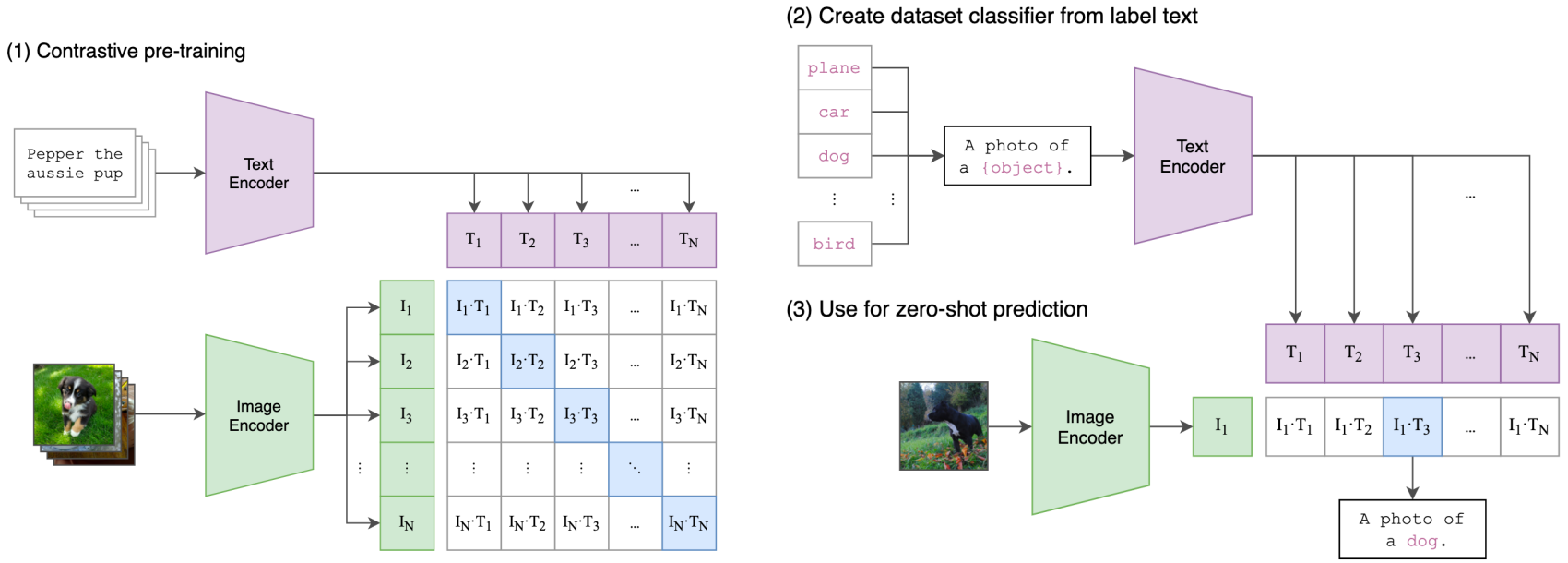
Ben Mildenhall* UC Berkeley
Pratul P. Srinivasan* UC Berkeley
Matthew Tancik* UC Berkeley
Jonathan T. Barron Google Research
Ravi Ramamoorthi UC San Diego
Ren Ng UC Berkeley



https://www.matthewtancik.com/nerf

Massively Trained Models on Noisy Data – OpenAI CLIP

- 400 million images with text from the Internet



Radford et al. Learning Transferable Visual Models From Natural Language Supervision. 2021

Thanks to all of you!