CS440/ECE448 Lecture 26 Computer Vision

Mark Hasegawa-Johnson, 4/2021

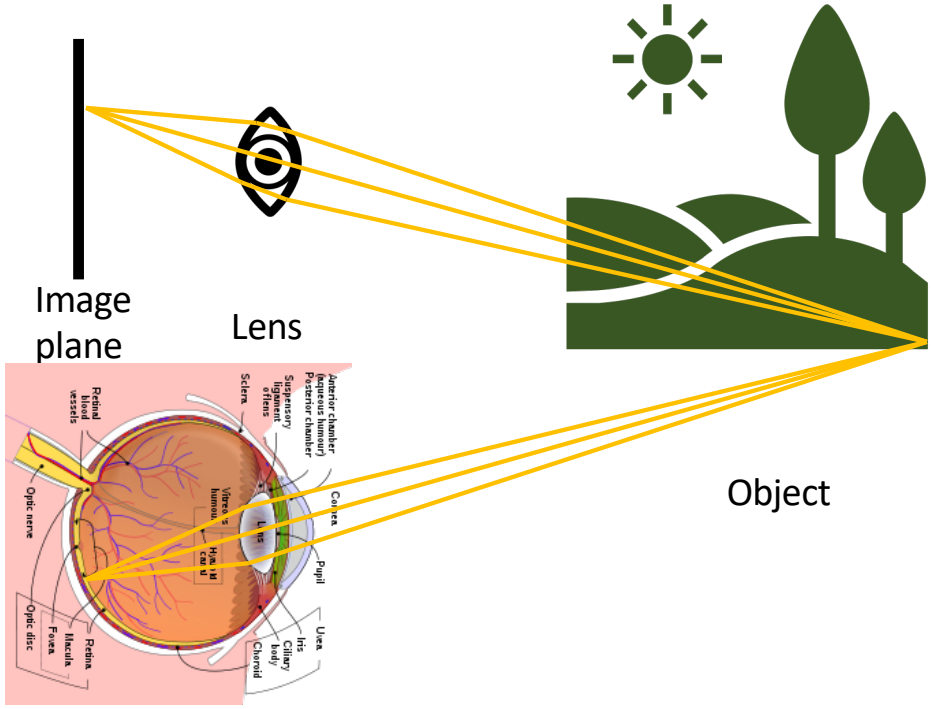
Slides may be redistributed under CC-BY 4.0. Most images are from other published sources; citations are provided inline.

Outline

- Cameras
  - Pinhole camera equations
  - Vanishing point
- Convolutions
  - Blur
  - Edge detection
  - Convolutional neural net
  - Imagenet

Lenses and focus

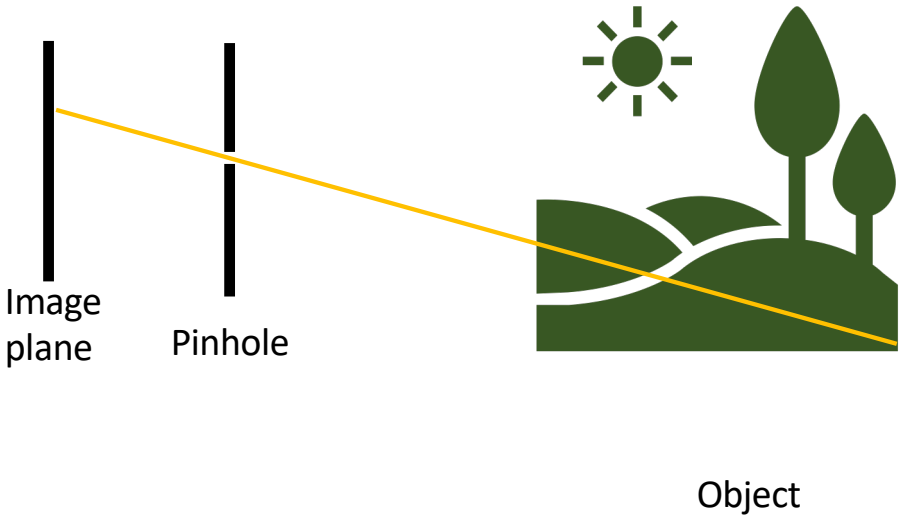
- The lens in your eye collects light.
- Light that passes directly through the center of the lens is not bent.
- Light that passes above center is bent back toward center, and vice versa, so that it can all be collected in the same point on the image plane.



By Rhcastilhos. And Jmarchn. - Schematic_diagram_of_the_human_eye_with_English_annotations.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1597930

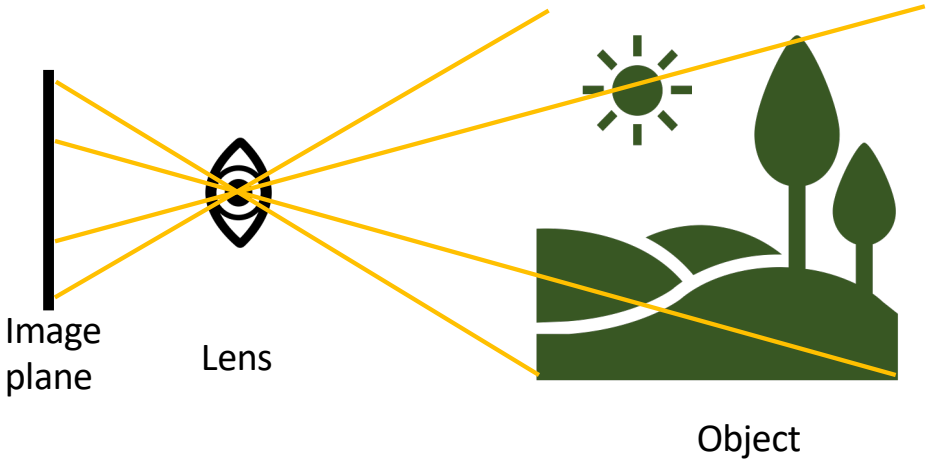
The “pinhole camera” approximation

- A “pinhole camera” is a camera that only allows light through a very small hole.
- Disadvantage: A pinhole camera gets much less light than a lens (because the hole is smaller).
- Advantage: A pinhole camera focuses on all objects, at every distance, simultaneously.



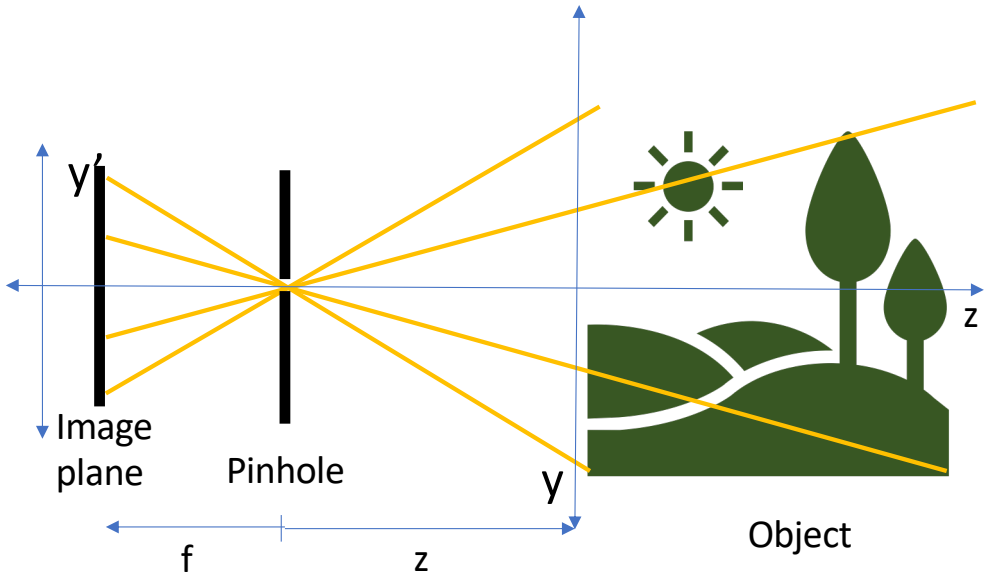
Converting a 3D world to a 2D picture

- Different spots in the real world are projected onto different points in the image plane.
- Light that passes through the center of the lens is not bent.
- Therefore, we can use the pinhole camera approximation to analyze the relationship between real world position (x,y,z) and position on the image plane (x',y').



The pinhole camera equations

- Define the origin (0,0,0) to be the pinhole.
- Define (x,y,z) as position of the object: x is horizontal (into the slide), y is vertical (upward), z is away from the camera.
- Define (x’,y’) as the position on the image plane where the light strikes (upside down).
- Define f as the distance from the pinhole to the image plane.



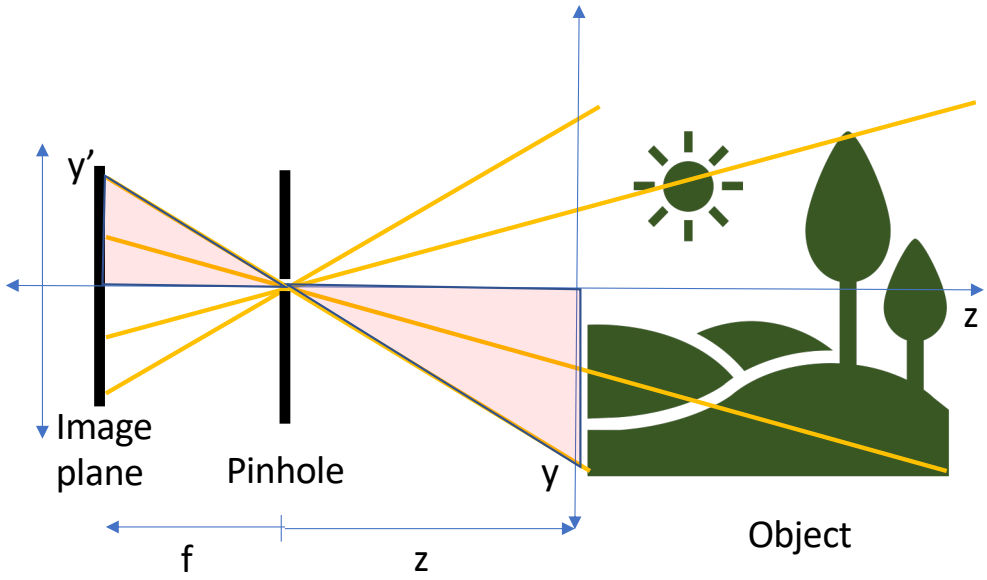
The pinhole camera equations

- These are similar triangles! So

$$\frac{y'}{f} = \frac{-y}{z}, \qquad \frac{x'}{f} = \frac{-x}{z}$$

- Solving for (x',y'), we get the pinhole camera equations:

$$x' = \frac{-fx}{z}, \qquad y' = \frac{-fy}{z}$$



Outline

- Cameras
  - Pinhole camera equations
  - Vanishing point
- Convolutions
  - Blur
  - Edge detection
  - Convolutional neural net
  - Imagenet

Vanishing point

- When you take a picture, lines that are parallel in the real world appear to converge.
- The point toward which they converge is called the vanishing point. It lies on the horizon.
- The “horizon” is a line in the image, where a plane parallel to the ground passes through the pinhole.

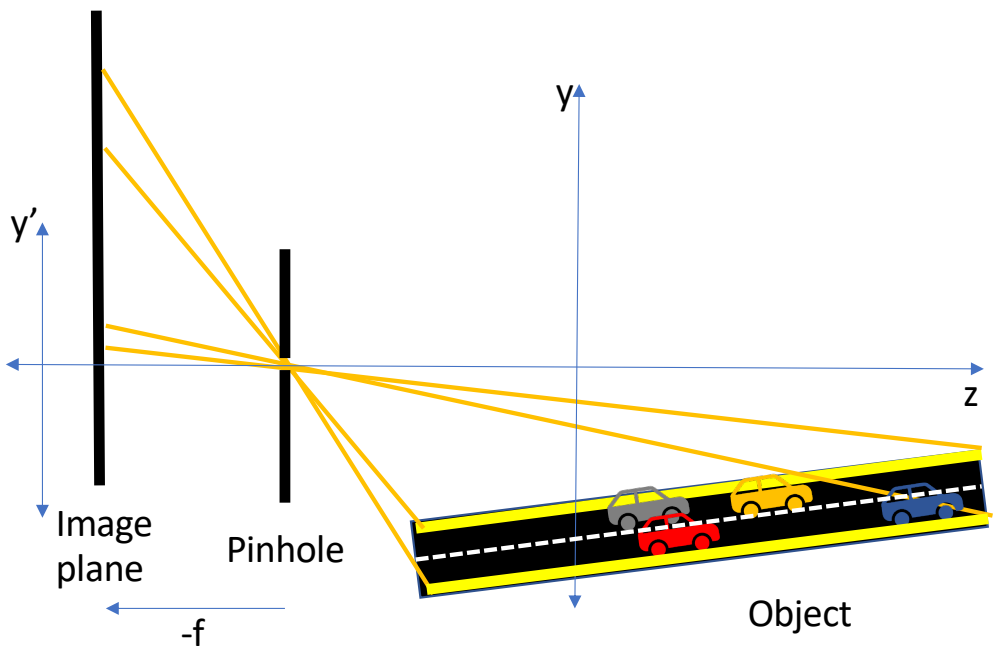
Vanishing point

- Suppose we are imaging a couple of parallel lines, for which x and y depend on z:

Line 1: $ax_1 + z = b_1, cy_1 + z = d_1$

Line 2: $ax_2 + z = b_2, cy_2 + z = d_1$

- These are parallel lines, so they have the same slopes, a, c.
- They are slightly offset from one another, so they have different offsets b_1, b_2, d_1, d_2



Vanishing point

Plugging in the similar triangle equations

$$\frac{y'}{f} = \frac{-y}{z}, \qquad \frac{x'}{f} = \frac{-x}{z}$$

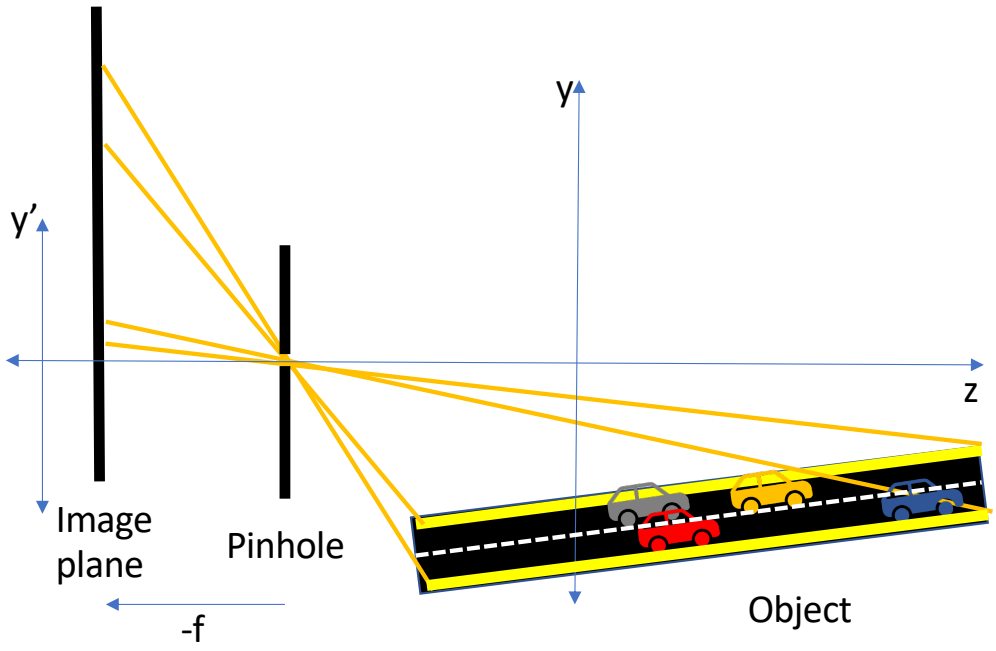
we get

Line 1: $-a\frac{x_1'}{f} + 1 = \frac{b_1}{z}$,

$$-c\frac{y_1'}{f} + 1 = \frac{d_1}{z}$$

Line 2: $-a\frac{x_2'}{f} + 1 = \frac{b_2}{z}$

$$-c\frac{y_1'}{f} + 1 = \frac{d_1}{z}$$



Vanishing point

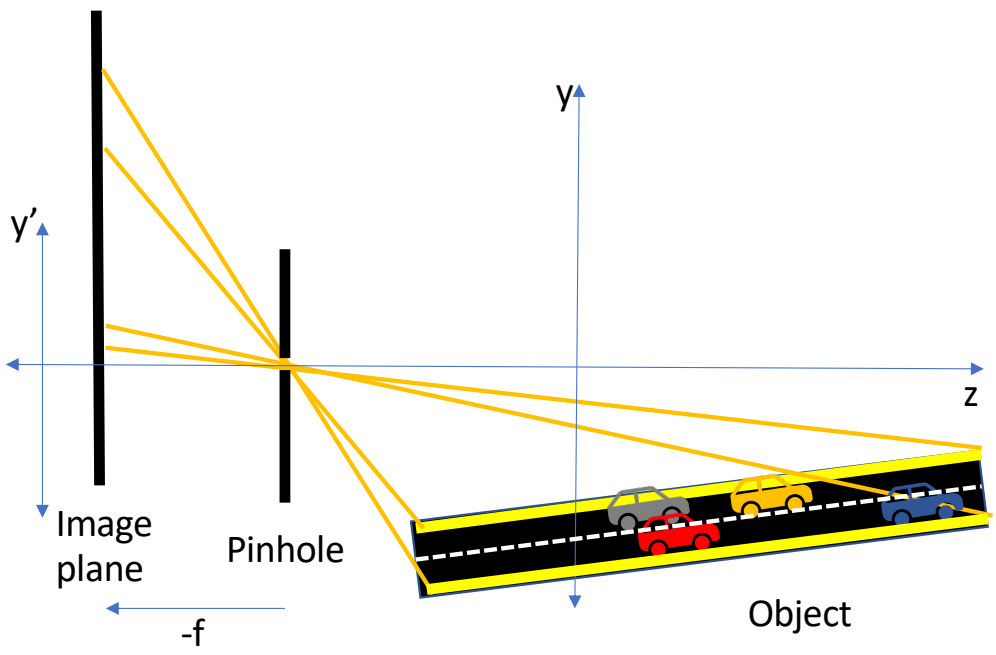
In the limit as $z \to \infty$:

Line 1: $-a\frac{x_1'}{f'} + 1 = \frac{b_1}{z} \to 0,$

$$-c\frac{y_1}{f} + 1 = \frac{d_1}{z} \to 0$$

Line 2: $-a\frac{x_2'}{f'} + 1 = \frac{b_2}{z} \to 0$

$$-c\frac{y_1}{f} + 1 = \frac{d_1}{z} \to 0$$



Vanishing point

So

Line 1: $x_1' \to \frac{f}{a}, y_1' \to \frac{f}{c}$

Line 2: $x_2' \to \frac{f}{a}, y_2' \to \frac{f}{c}$

Outline

- Cameras
  - Pinhole camera equations
  - Vanishing point
- Convolutions
  - Blur
  - Edge detection
  - Convolutional neural net
  - Imagenet

- An RGB image is a signal in three dimensions: $f[i, j, k] =$ intensity of the signal in the ith row, j^{th} column, and k^{th} color.
- $f[i, j, k]$, for each (i, j, k), is either stored as an integer or a floating point number:
  - Floating point: usually $x \in [0, 1]$, so $x = 0$ means dark, $x = 1$ means bright.
  - Integer: usually $x \in \{0, \ldots, 255\}$, so $x = 0$ means dark, $x = 255$ means bright.
- The three color planes are usually:
  - $k = 0$: Red
  - $k = 1$: Blue
  - $k = 2$: Green

- "Local averaging" means that we create an output image, $y[i,j,k]$, each of whose pixels is an **average** of nearby pixels in $f[i,j,k]$.
- For example, if we average along the rows:

$$y[i,j,k] = \frac{1}{2M+1} \sum_{j'=j-M}^{j+M} f[i,j',k]$$

- If we average along the columns:

$$y[i,j,k] = \frac{1}{2M+1} \sum_{i'=i-M}^{i+M} f[i',j,k]$$

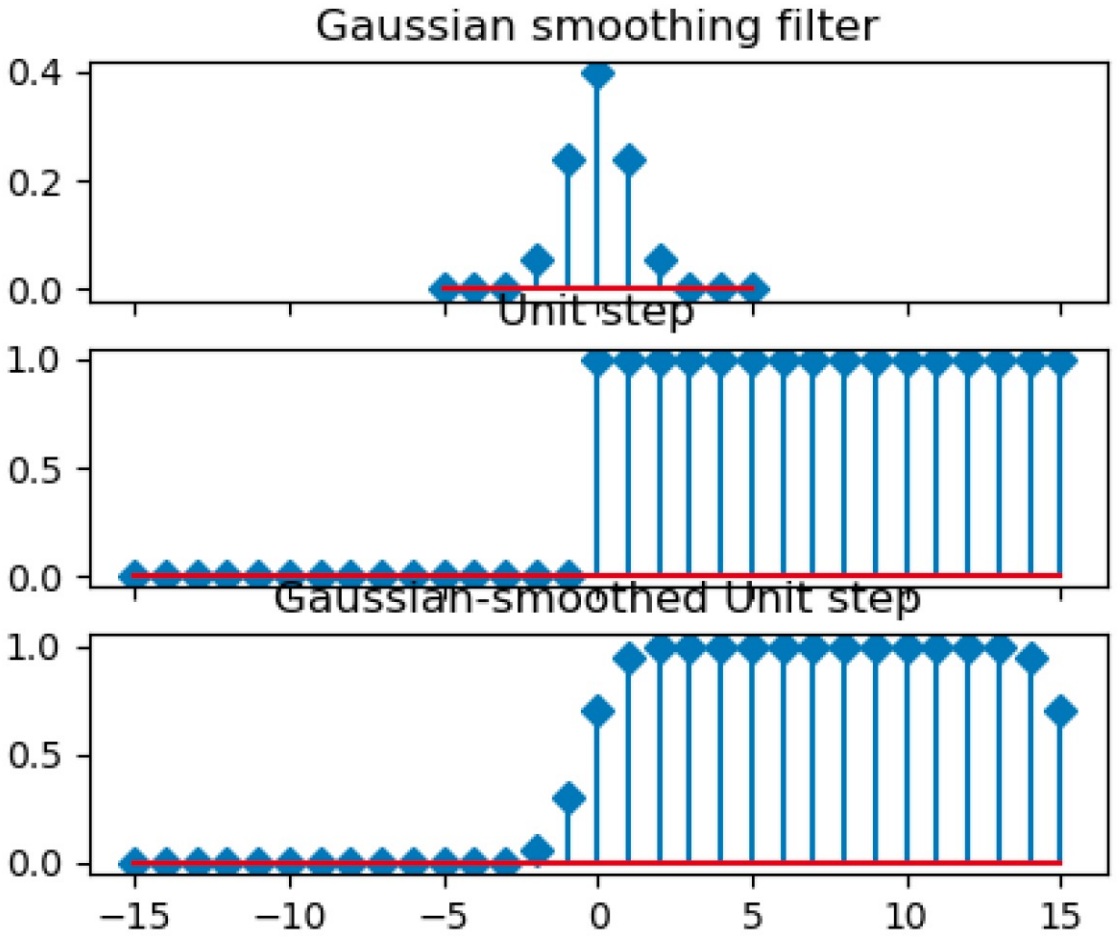
- Suppose we don't want the edges quite so abrupt. We could do that using "weighted local averaging:" each pixel of $y[i,j,k]$ is a **weighted average** of nearby pixels in $f[i,j,k]$, with some averaging weights $g[n]$.
- For example, if we average along the rows:

$$y[i,j,k] = \sum_{m=j-M}^{j+M} g[j-m]f[i,m,k]$$

- If we average along the columns:

$$y[i,j,k] = \sum_{i'=i-M}^{i+M} g[i-m]f[m,j,k]$$

The top row are the averaging weights, $g[n]$. The middle row shows the input, $f[n]$. The bottom row shows the output, $y[n]$.



Gaussian blur

Weighted averaging is a little better than unweighted averaging. We just need to make sure that the weights add up to one. For example:

$$y[i,j] = \sum_{m=-5}^{5} \sum_{n=-5}^{5} g[m,n] f[i-m, j-n]$$

$$g[m,n] = \frac{1}{2\pi\sigma^2} e^{-\left(\left(\frac{m}{\sigma}\right)^2 + \left(\frac{n}{\sigma}\right)^2\right)}$$

The Gaussian blur filter $h(m,n)$ results in different degrees of smoothness of $g(x',y')$, depending on how we set the hyperparameter σ (the StDev):

By IkamusumeFan - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=41790217

- A **convolution** is exactly the same thing as a **weighted local average**. We give it a special name, because we will use it very often. It's defined as:

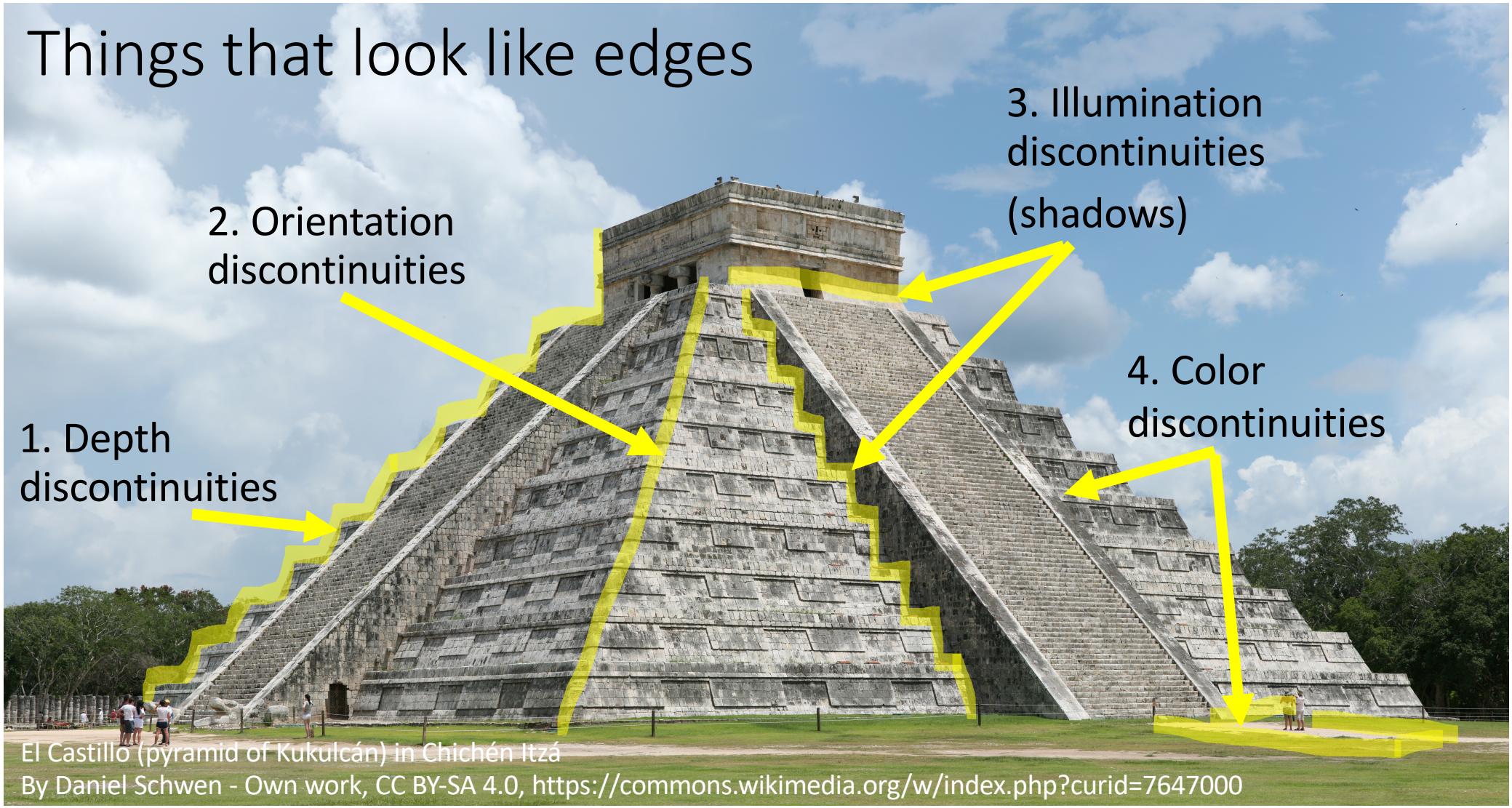
$$y[n] = \sum_m g[m]f[n-m] = \sum_m g[n-m]f[m]$$

- We use the symbol $*$ to mean "convolution:"

$$y[n] = g[n] * f[n] = \sum_m g[m]f[n-m] = \sum_m g[n-m]f[m]$$

Outline

- Cameras
  - Pinhole camera equations
  - Vanishing point
- Convolutions
  - Blur
  - Edge detection
  - Convolutional neural net
  - Imagenet



El Castillo (pyramid of Kukulcán) in Chichén Itzá

By Daniel Schwen - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=7647000

Edge detection by subtraction

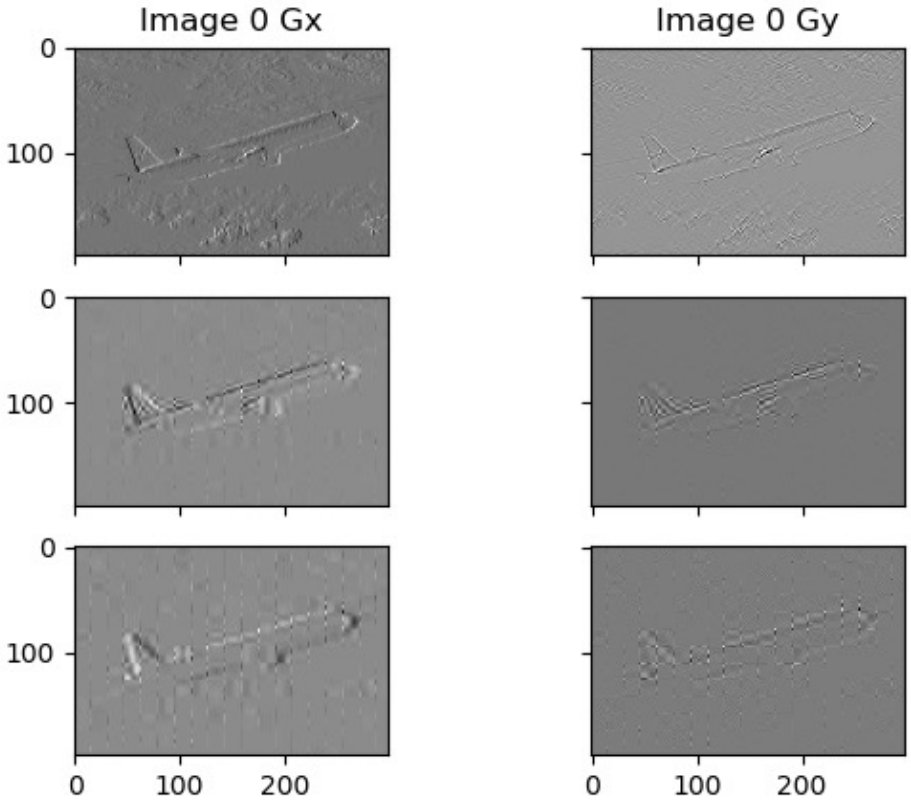
Subtract neighboring pixels, to compute an "image gradient:"

X gradient:

$$\nabla_x f[i,j] \approx \frac{f[i,j+1] - f[i,j-1]}{2}$$

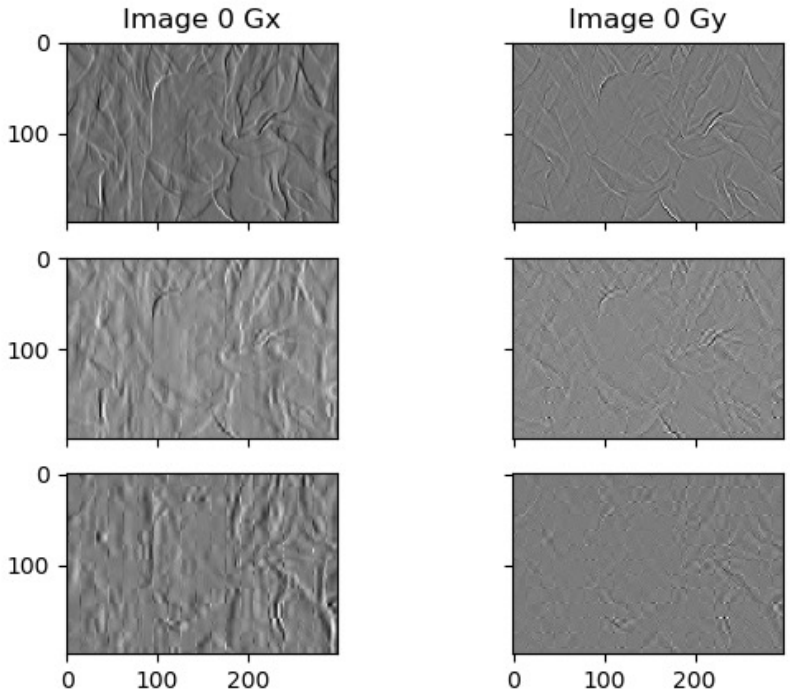
Y gradient:

$$\nabla_y f[i,j] \approx \frac{f[i+1,j] - f[i-1,j]}{2}$$



A problem with “edge detection by subtraction”

It tends to exaggerate noise.



Solving the noise problem

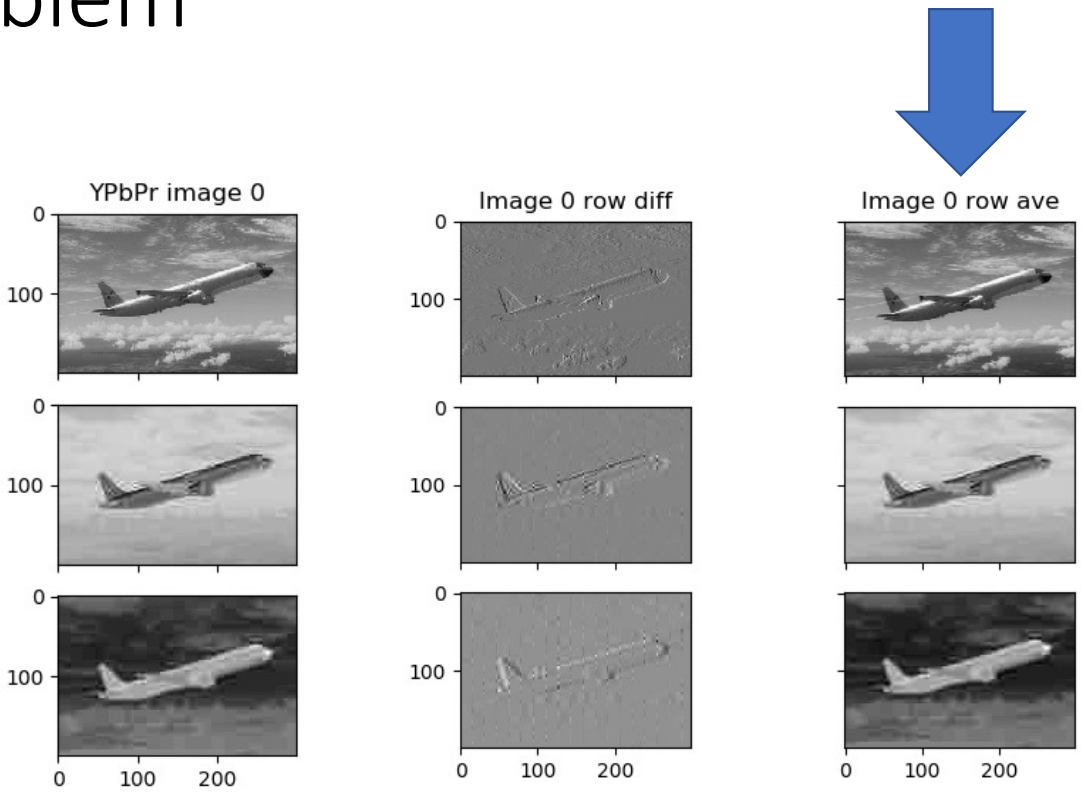
We can solve the noise problem by smoothing the image first, then taking the difference.

Smoothing is done by taking a local average, e.g.,

$$s[i,j] = \sum_{m=-5}^{5} \sum_{n=-5}^{5} \left(\frac{1}{(11)^2} \right) f[i-m, j-n]$$

$$\nabla_x f[i,j] \approx \frac{s[i,j+1] - s[i,j-1]}{2}$$

$$\nabla_y f[i,j] \approx \frac{s[i+1,j] - s[i-1,j]}{2}$$



Gaussian blur

Weighted averaging is a little better than unweighted averaging. We just need to make sure that the weights add up to one. For example:

$$s[i,j] = \sum_{m=-5}^{5} \sum_{n=-5}^{5} h[m,n] f[i-m, j-n]$$

$$h[m,n] = \frac{1}{2\pi\sigma^2} e^{-\left(\left(\frac{m}{\sigma}\right)^2 + \left(\frac{n}{\sigma}\right)^2\right)}$$

$$\nabla_x f[i,j] \approx \frac{s[i,j+1] - s[i,j-1]}{2}$$

$$\nabla_y f[i,j] \approx \frac{s[i+1,j] - s[i-1,j]}{2}$$

The Gaussian blur filter $h(m,n)$ results in different degrees of smoothness of $S(x',y')$, depending on how we set the hyperparameter σ (the StDev):

By IkamusumeFan - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=41790217

Difference of Gaussians

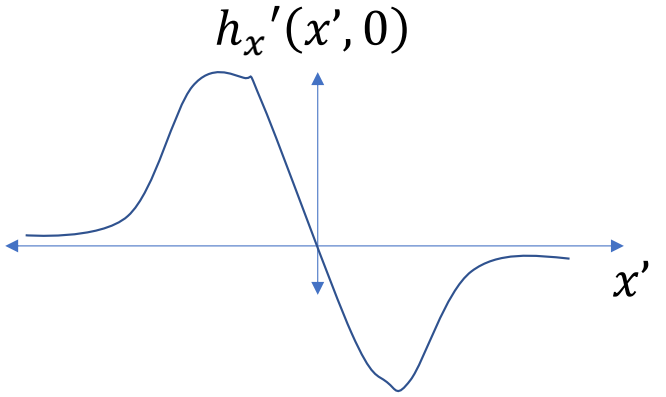
A "difference-of-Gaussians" filter is created by subtracting two Gaussian-blur filters, like this:

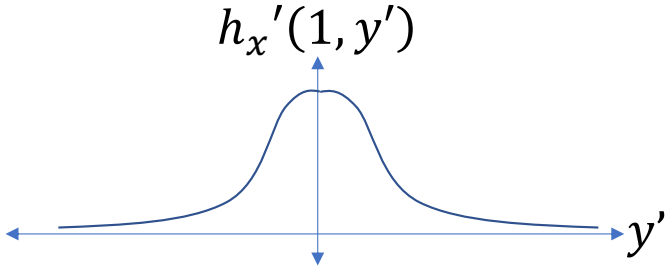
$$h[m,n] = \frac{1}{2\pi\sigma^2} e^{-\left(\left(\frac{m}{\sigma}\right)^2 + \left(\frac{n}{\sigma}\right)^2\right)}$$

$$h_x'[m,n] = \frac{h[m,n+1] - h[m,n-1]}{2}$$

$$h_y'[m,n] = \frac{h[m+1,n] - h[m-1,n]}{2}$$

A "difference-of-Gaussians" filter looks kind of like this:





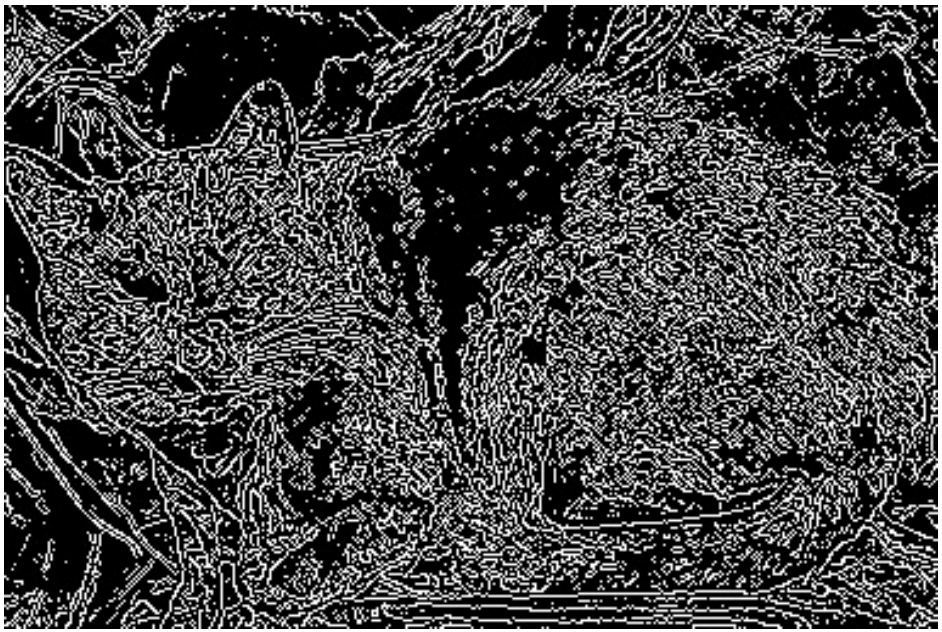
Difference of Gaussians

If we pre-compute the difference-of-Gaussians filters, then we can combine the weighted-average and the subtraction into just one operation, to save computation:

$$\nabla_x f[i,j] = \sum_{m=-5}^{5} \sum_{n=-5}^{5} h_x'[m,n] f[i-m, j-n]$$

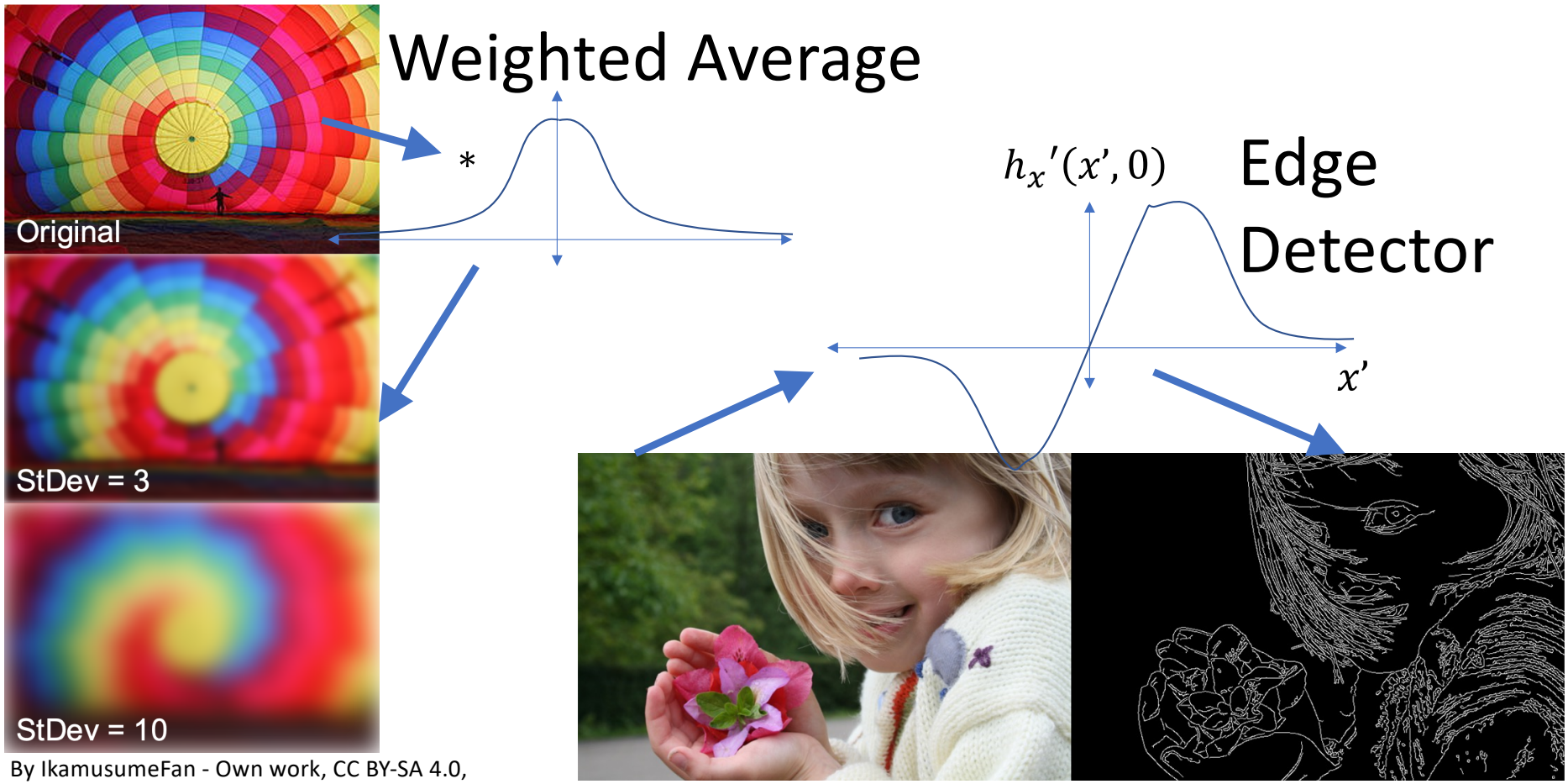
$$\nabla_y f[i,j] = \sum_{m=-5}^{5} \sum_{n=-5}^{5} h_y'[m,n] f[i-m, j-n]$$

The difference-of-Gaussians filters, $h_x'[m,n]$ and $h_y'[m,n]$, detect more or less edges, depending on how we set the hyperparameter σ. Here is $\sqrt{\nabla_x f[i,j]^2 + \nabla_y f[i,j]^2}$, thresholded to make it black and white:



By Overremorto - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=10581259

Example Convolutions: Weighted Average, Edge Detector



By IkamusumeFan - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=41790217

Jon McLoone, CC-SA 3.0,https://commons.wikimedia.org/wiki/File:%C3%84%C3%A4retuvastuse_n%C3%A4ide.png

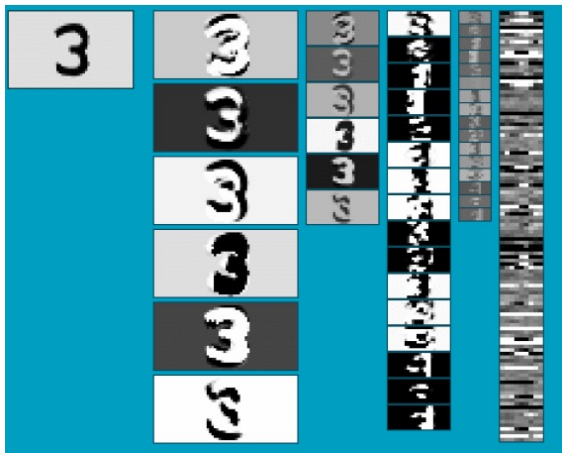
Outline

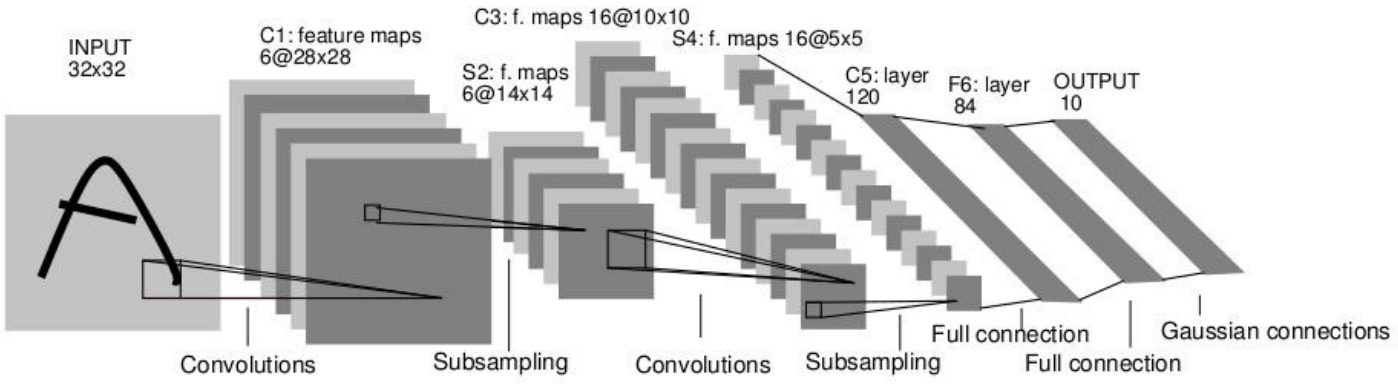
- Cameras
  - Pinhole camera equations
  - Vanishing point
- Convolutions
  - Blur
  - Edge detection
  - Convolutional neural net
  - Imagenet

Convolutional Neural Networks

Slide by Svetlana Lazebnik, 9/2017

- Neural network with specialized connectivity structure
- Stack multiple stages of feature extractors (trainable convolutions!!)
- Higher stages compute more global, more invariant features
- Classification layer at the end



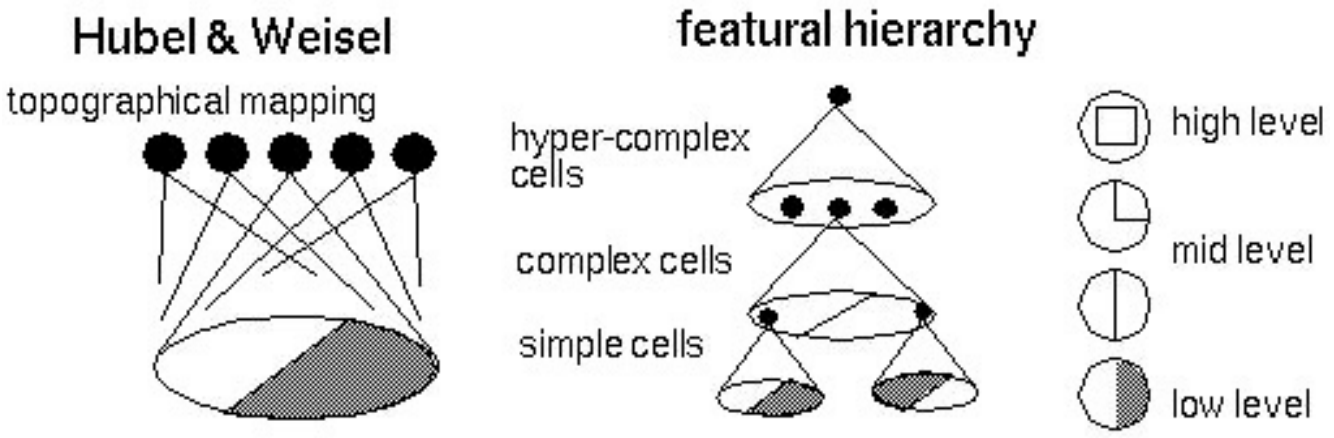


Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.

Biological inspiration

Slide by Svetlana Lazebnik, 9/2017

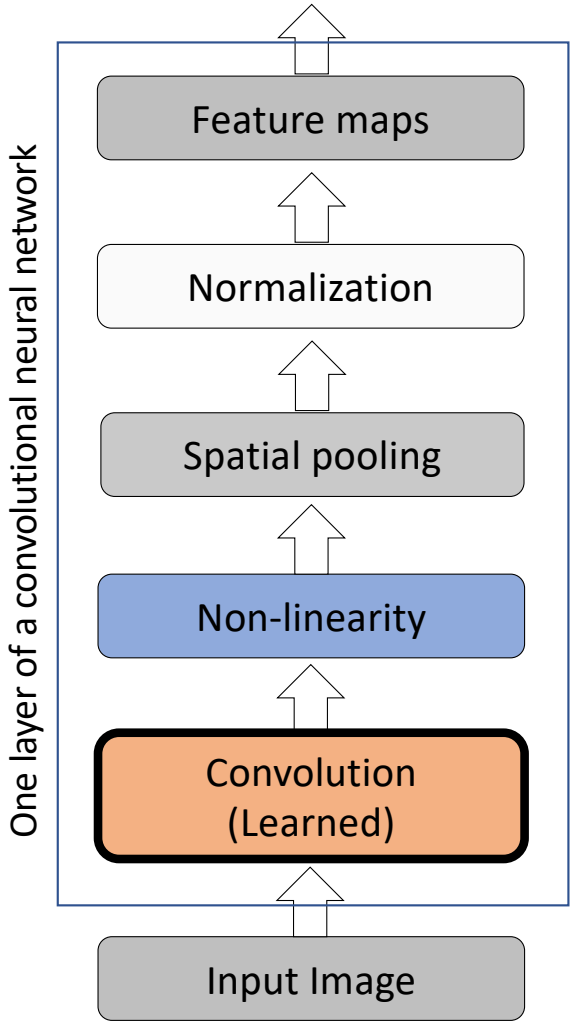
- D. Hubel and T. Wiesel (1959, 1962, Nobel Prize 1981)
  - Visual cortex consists of a hierarchy of *simple*, *complex*, and *hyper-complex* cells



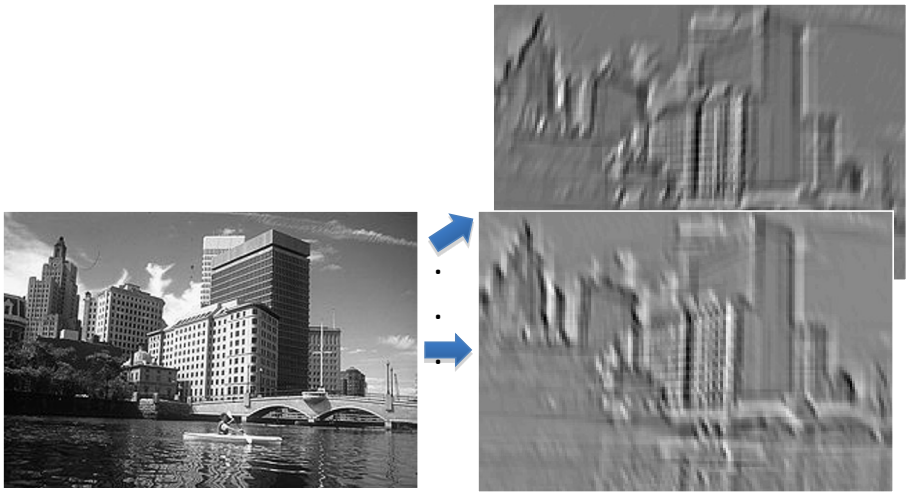
Source

Convolutional Neural Networks

Slide by Svetlana Lazebnik, 9/2017

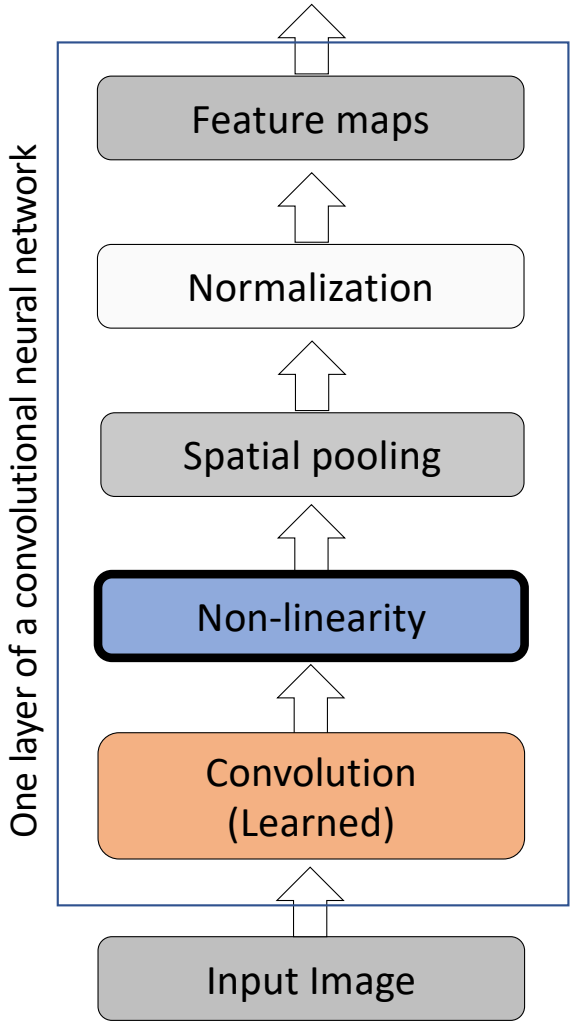


$$D(x', y') = h(x', y') * Y(x', y')$$

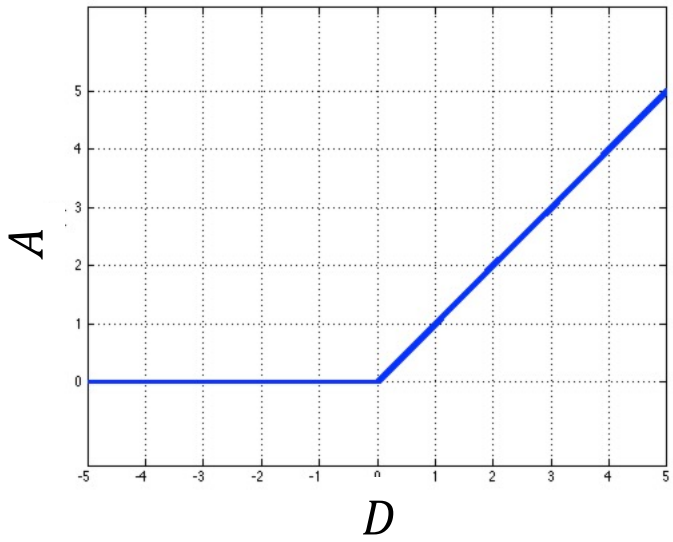


Convolutional Neural Networks

Slide by Svetlana Lazebnik, 9/2017

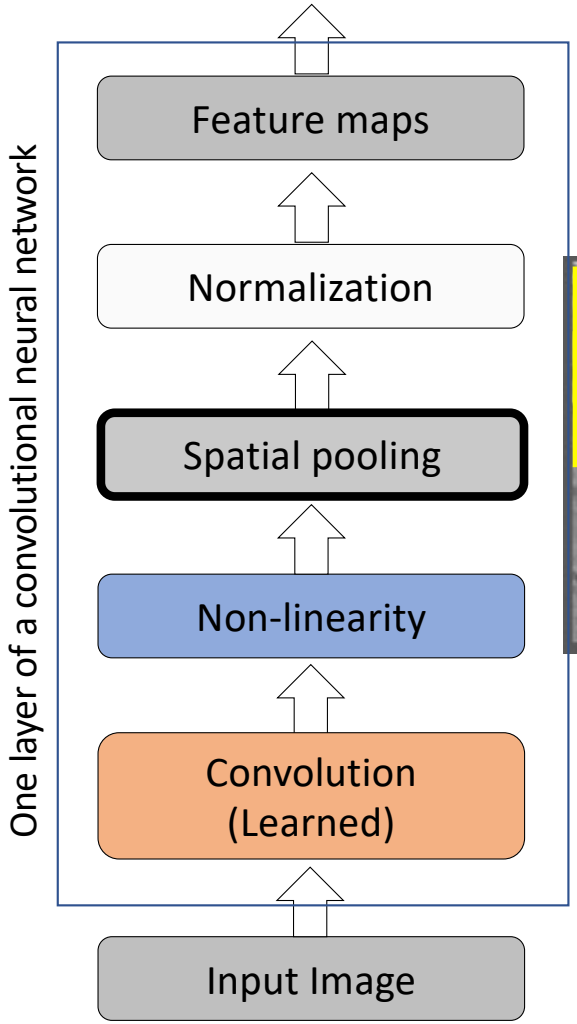


$$A = g(D)$$

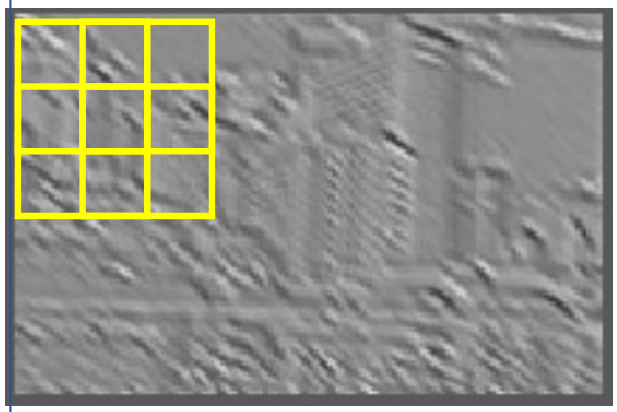


Convolutional Neural Networks

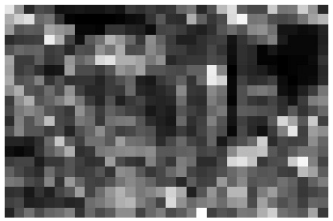
Slide by Svetlana Lazebnik, 9/2017



$\max_{x',y'} A(x', y')$, within some local region.

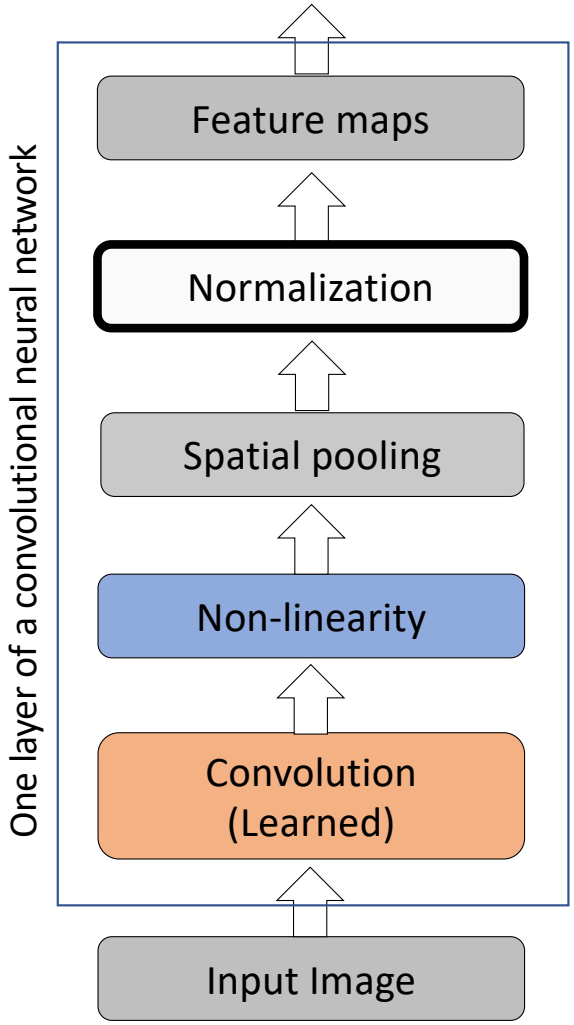


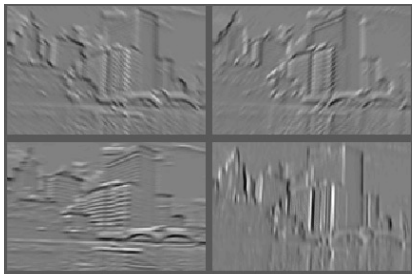
Max



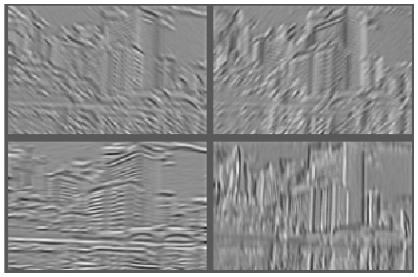
Convolutional Neural Networks

Slide by Svetlana Lazebnik, 9/2017





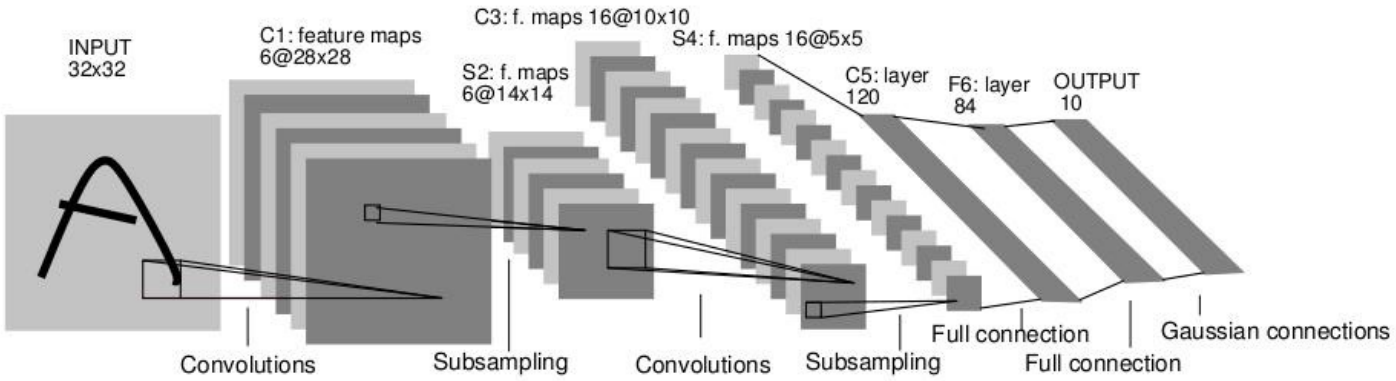
Feature Maps



Feature Maps
After Contrast
Normalization

LeNet, 1998

- (Convolution, Nonlinearity, Maximum-Subsampling, Normalization) X 2 layers
- After the second subsampling, the resulting features are concatenated to form a vector, which is then classified.



Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.

Outline

- Cameras
  - Pinhole camera equations
  - Vanishing point
- Convolutions
  - Blur
  - Edge detection
  - Convolutional neural net
  - Imagenet

ImageNet Challenge

IMAGENET

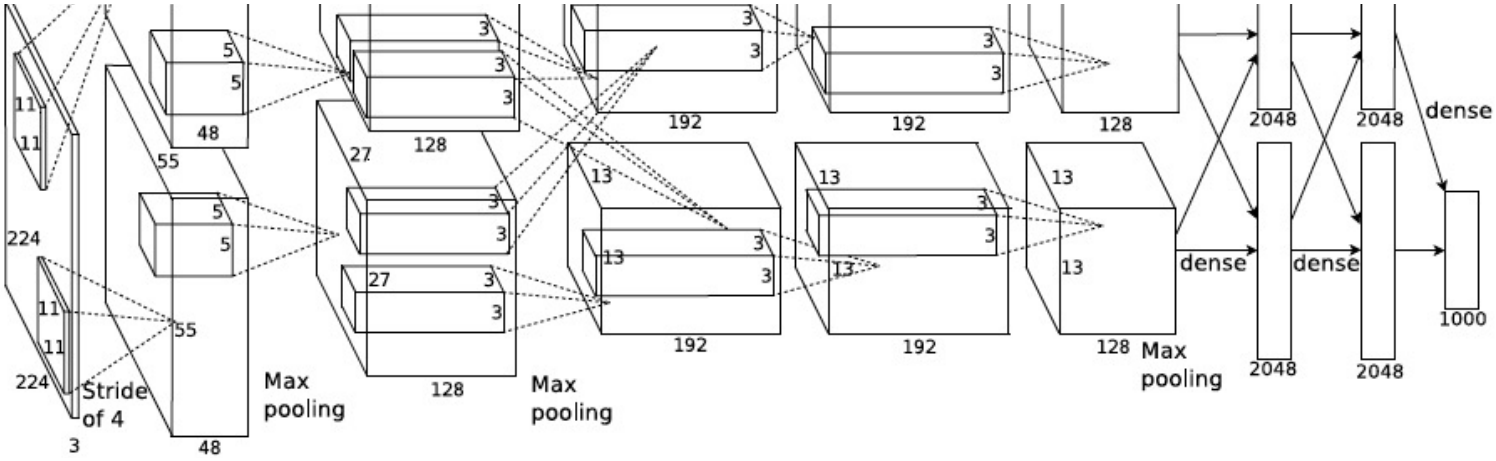
[Deng et al. CVPR 2009]

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- Challenge: 1.2 million training images, 1000 classes

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

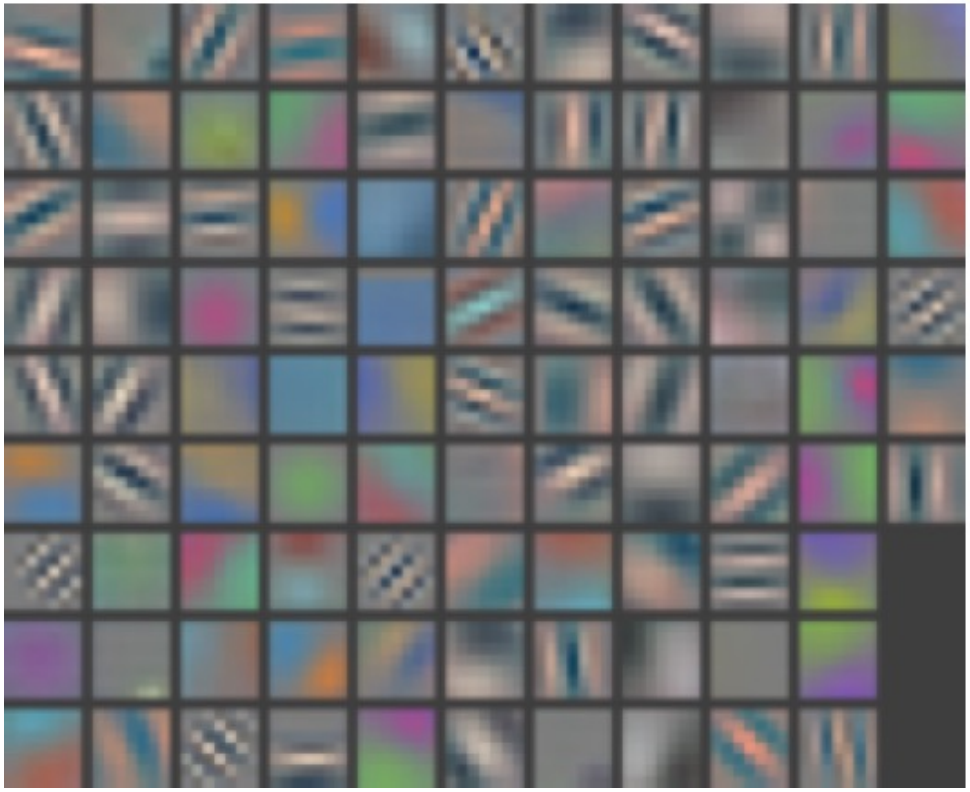
AlexNet

- Similar framework to LeCun'98 but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data (10^6 vs. 10^3 images)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week



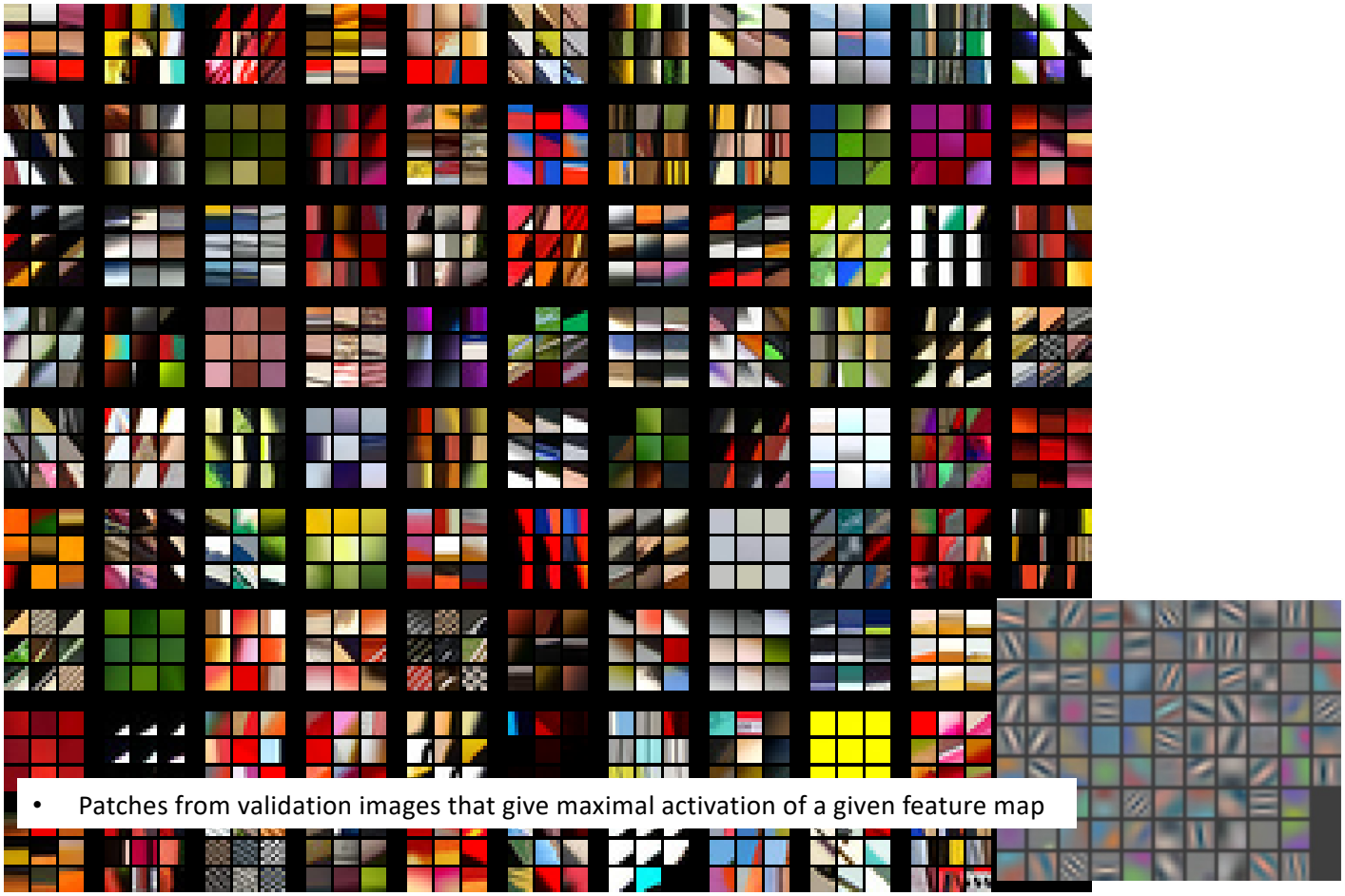
A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Layer 1 Filters



M. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, arXiv preprint, 2013

Layer 1: Top-9 Patches



- Patches from validation images that give maximal activation of a given feature map

Layer 2: Top-9 Patches

- Patches from validation images that give maximal activation of a given feature map

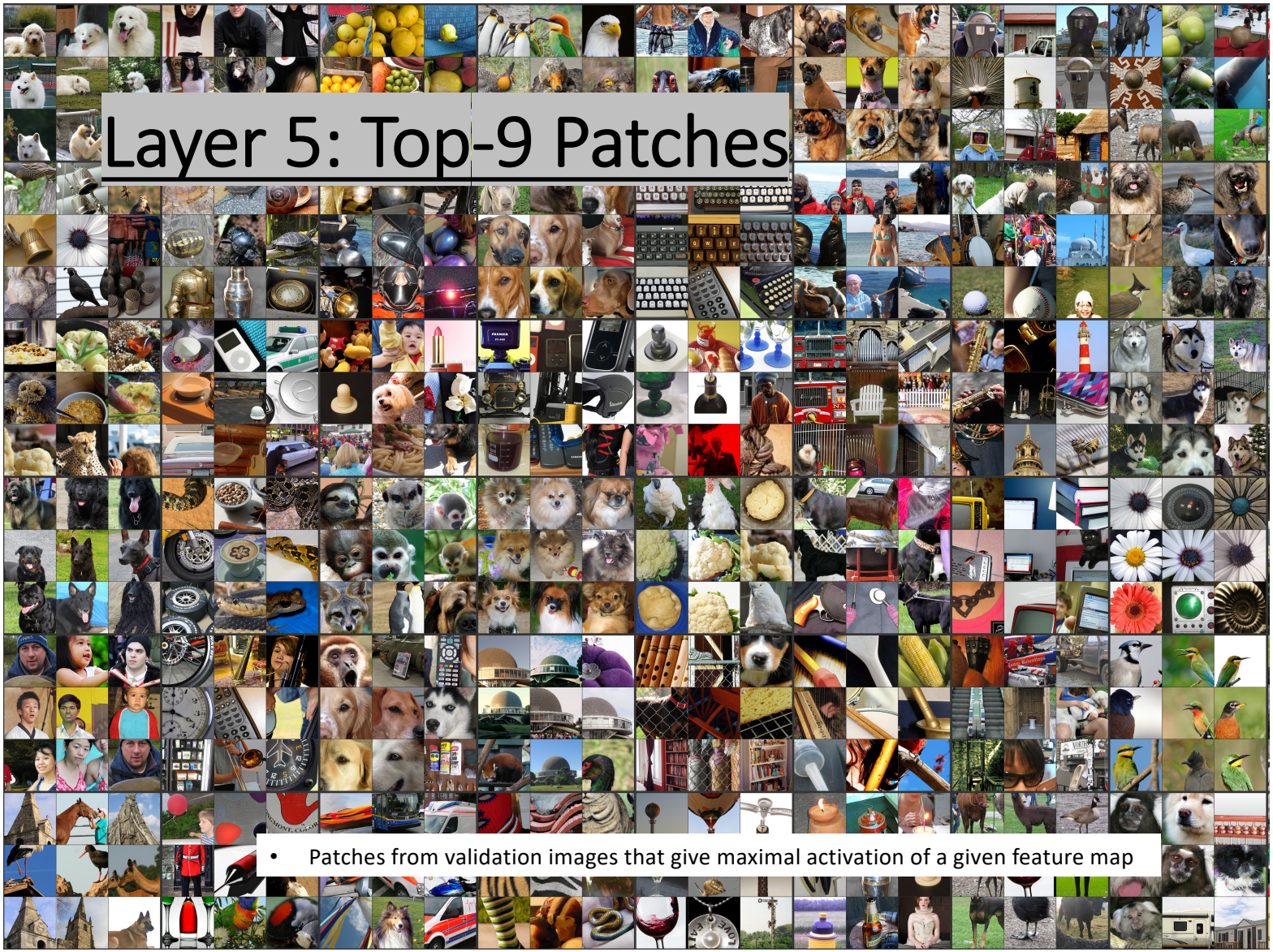
Layer 3: Top-9 Patches

- Patches from validation images that give maximal activation of a given feature map

Layer 4: Top-9 Patches

- Patches from validation images that give maximal activation of a given feature map

Layer 5: Top-9 Patches



- Patches from validation images that give maximal activation of a given feature map

Conclusions

- Pinhole camera equations tell you the relationship between the position on the image, (x',y'), and the position in the real world, (x,y,z). In particular, they tell you why parallel lines seem to converge at the vanishing point.
- Weighted averaging (e.g., Gaussian blur) and edge detection are two examples of convolution.
- Edges are caused by discontinuities of depth, orientation, illumination, and color. It's useful to detect where such things happen! Subtracting pixels is noisy, so convolve with a difference-of-gaussians filter instead.
- A convolutional neural network is a series of layers, each of which contains a convolution, followed by a nonlinearity, followed by a local maximum.
- ImageNet is a database with millions of images, showing examples of over 1000 different objects. It has permitted people to train extremely complicated and highly accurate neural nets.