Image Representation, Processing, and Computer Vision

Ajad Chhatkuli and Bishesh Khanal

Dec 23, 2018

Contents

- What do we see ?

- What do we see ?
- Where are those located ?

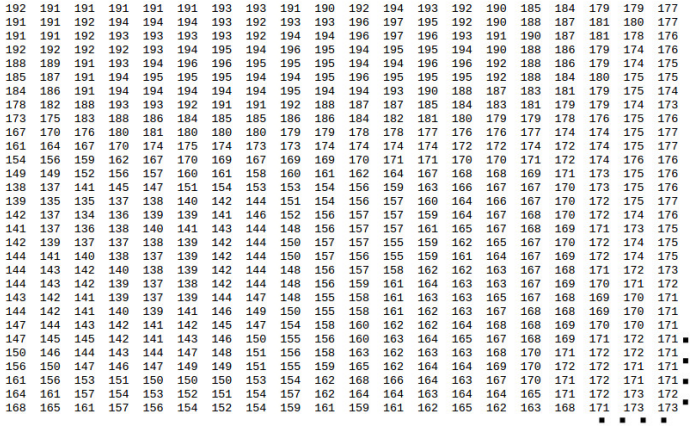
- What do we see ?
- Where are those located ?
- ...

Image Formation

What we see

Image Formation

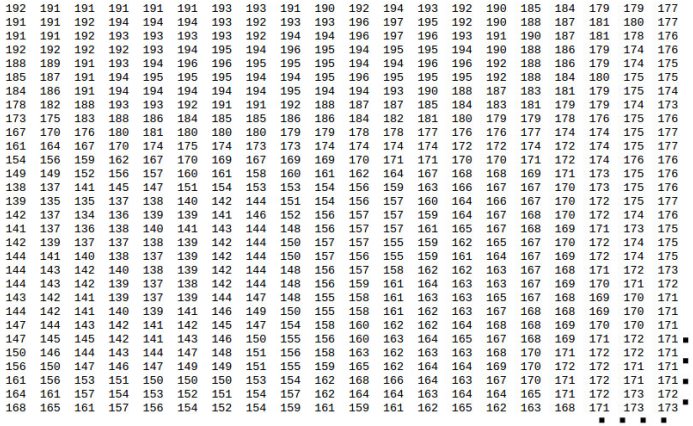
What we see



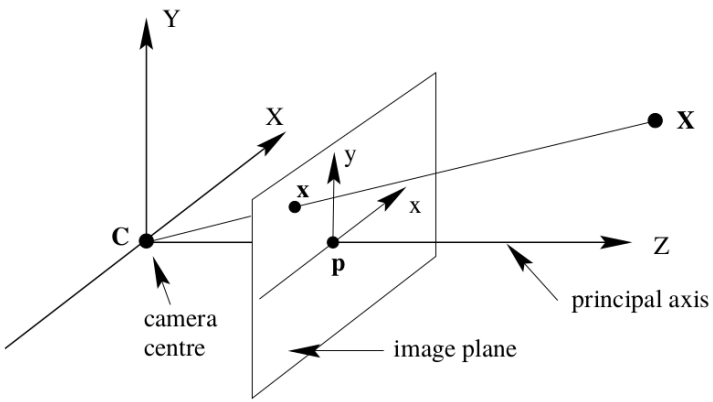
What machines see

Image Formation

What we see



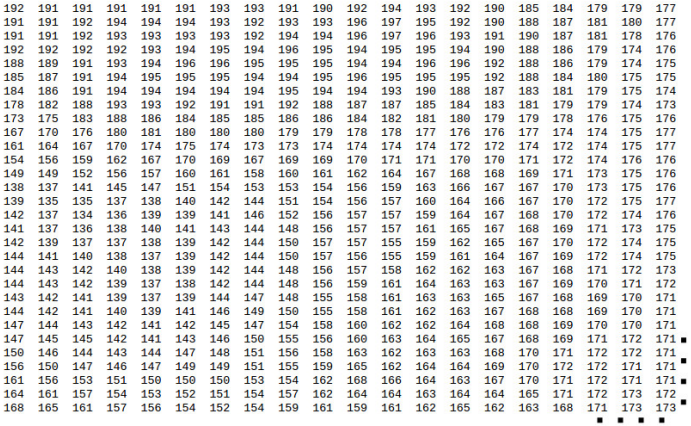
What machines see



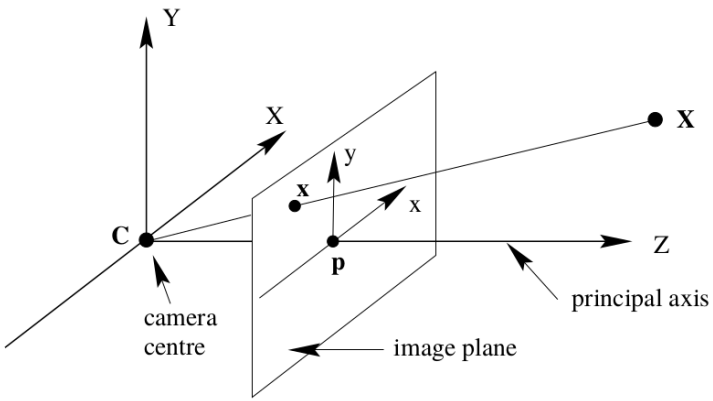
Source: Hartley, Richard, and Andrew Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003.

Image Formation

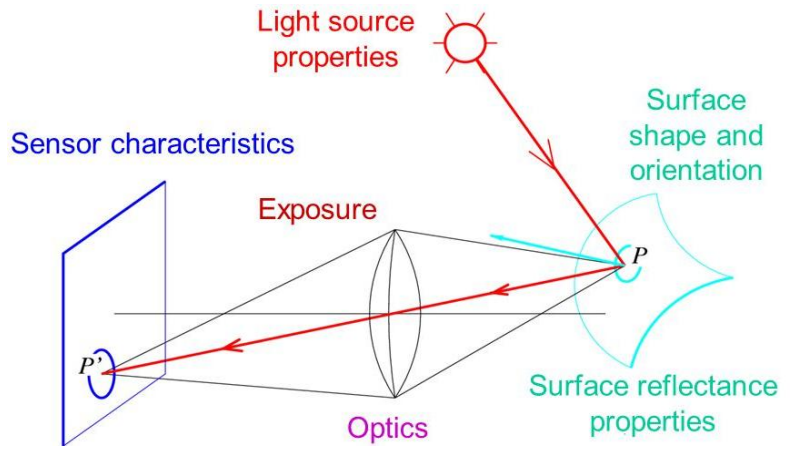
What we see



What machines see



Source: Hartley, Richard, and Andrew Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003.



Source: L. Fei Fei

Image: Function of Multiple Spatial Variables

- Computers just see numbers (in fact, a lot of numbers)
- 2D image: each discrete position (x, y) consist of intensity values
- 3D image: each discrete position (x, y, z) consist of intensity values
- grayscale images represented as $N \times M$ matrix
- color images (R, G, B) represented as $3 \times N \times M$ matrix

Two Closest images ?

Distance Between Images ?

Distance Between Images ?

$$A = \begin{pmatrix} 2 & 3 & 4 \\ 1 & 7 & 9 \end{pmatrix}$$

$$\mathrm{Vec}(A) = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 1 \\ 7 \\ 9 \end{pmatrix}$$

Distance Between Images ?

$$A = \begin{pmatrix} 2 & 3 & 4 \\ 1 & 7 & 9 \end{pmatrix}$$

$$\text{Vec}(A) = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 1 \\ 7 \\ 9 \end{pmatrix}$$

Remember distance between vectors in Euclidean space ?

Distance Between Images ?

$$A = \begin{pmatrix} 2 & 3 & 4 \\ 1 & 7 & 9 \end{pmatrix}$$

$$\text{Vec}(A) = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 1 \\ 7 \\ 9 \end{pmatrix}$$

Remember distance between vectors in Euclidean space ?
Sum of squared distance:
$\sum \left((x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2\right)$

Distance Between Images ?

$$A = \begin{pmatrix} 2 & 3 & 4 \\ 1 & 7 & 9 \end{pmatrix}$$

$$\text{Vec}(A) = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 1 \\ 7 \\ 9 \end{pmatrix}$$

Remember distance between vectors in Euclidean space ?
Sum of squared distance:
$\sum \left((x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2\right)$
Can you think of other distances ?

Distance Between Images ?

$$A = \begin{pmatrix} 2 & 3 & 4 \\ 1 & 7 & 9 \end{pmatrix}$$

$$\text{Vec}(A) = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 1 \\ 7 \\ 9 \end{pmatrix}$$

Remember distance between vectors in Euclidean space ?
Sum of squared distance:
$\sum \left((x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2\right)$
Can you think of other distances ?
Sum of absolute values of differences:
$\sum \left(|x_1 - y_1| + |x_2 - y_2| + ... + |x_n - y_n|\right)$

Distance Between Images ?

$$A = \begin{pmatrix} 2 & 3 & 4 \\ 1 & 7 & 9 \end{pmatrix}$$

$$\text{Vec}(A) = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 1 \\ 7 \\ 9 \end{pmatrix}$$

Remember distance between vectors in Euclidean space ?
Sum of squared distance:
$\sum \left((x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2\right)$
Can you think of other distances ?
Sum of absolute values of differences:
$\sum \left(|x_1 - y_1| + |x_2 - y_2| + ... + |x_n - y_n|\right)$
Norms: L_1, L_2, ...

Two Closest images ?

Two Closest images ?

Based on SSD, Kitten images are further apart!

Two Closest images ?

Based on SSD, Kitten images are further apart!
Euclidean space not suitable in most cases for images

Two Closest images ?

Based on SSD, Kitten images are further apart!
Euclidean space not suitable in most cases for images
Better representation ? what manifold do the images lie ?
Better metric "distance"

Dimensionality Reduced Representation, Feature Extraction

Dimensionality Reduced Representation, Feature Extraction

- An image of 600×400 size is a point in 240000 dimensional space

Dimensionality Reduced Representation, Feature Extraction

- An image of 600×400 size is a point in 240000 dimensional space
- Even with all the computational might that we have now, this high dimensional data is too large to process

Dimensionality Reduced Representation, Feature Extraction

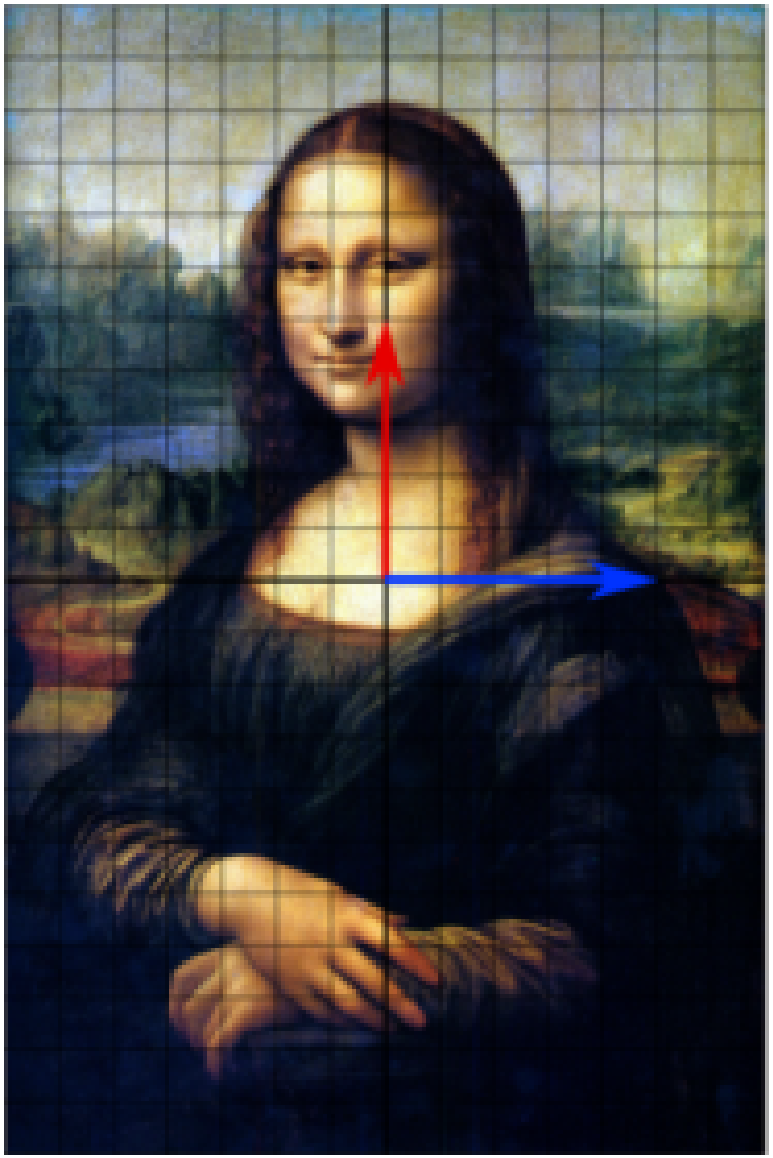
- An image of 600×400 size is a point in 240000 dimensional space
- Even with all the computational might that we have now, this high dimensional data is too large to process
- Can extract a small set of informative and non-redundant features

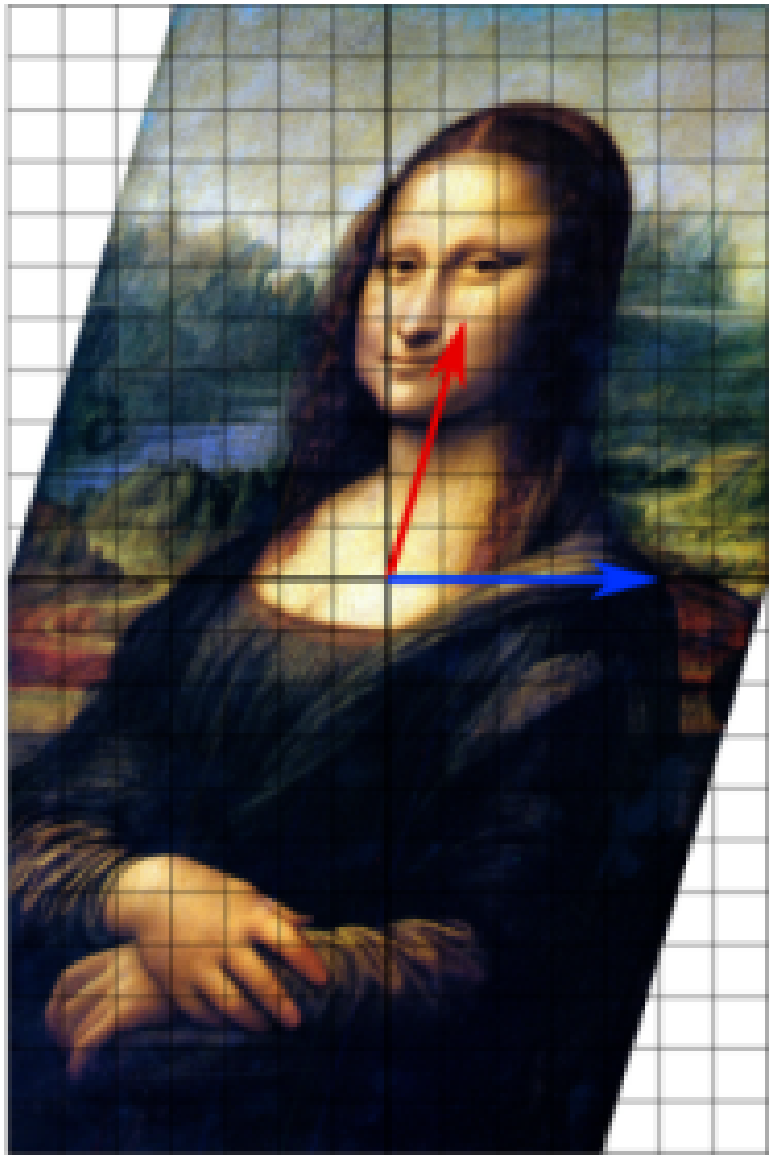
Dimensionality Reduced Representation, Feature Extraction

- An image of 600×400 size is a point in 240000 dimensional space
- Even with all the computational might that we have now, this high dimensional data is too large to process
- Can extract a small set of informative and non-redundant features
- Can represent image differently in a low dimensional space

Principal Component Analysis

Eigenvectors





Source: By TreyGreer62 https://commons.wikimedia.org/w/index.php?curid=12768508

Eigenvectors and Eigenvalues

A matrix A acts on vector $\vec{x}$ to give a new vector $\vec{b}$

$$\text{E.g. } A = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \quad \vec{x} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad A\vec{x} = \vec{b} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

- Generally, $\vec{x}$ and $\vec{b}$ are in different direction

Eigenvectors and Eigenvalues

A matrix A acts on vector $\vec{x}$ to give a new vector $\vec{b}$

$$\text{E.g. } A = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \quad \vec{x} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad A\vec{x} = \vec{b} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

- Generally, $\vec{x}$ and $\vec{b}$ are in different direction
- If $\vec{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\vec{b} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} = 2\vec{x}$
- $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 2 \\ 2 \end{bmatrix}$ in same direction

Eigenvectors and Eigenvalues

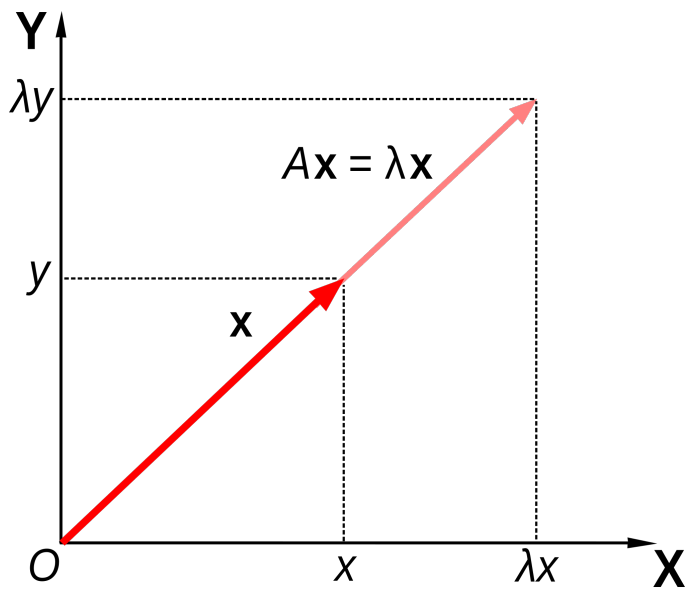
A matrix A acts on vector $\vec{x}$ to give a new vector $\vec{b}$

$$\text{E.g. } A = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \quad \vec{x} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad A\vec{x} = \vec{b} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

- Generally, $\vec{x}$ and $\vec{b}$ are in different direction
- If $\vec{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\vec{b} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} = 2\vec{x}$
- $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 2 \\ 2 \end{bmatrix}$ in same direction

If $A\vec{x} = \lambda\vec{x}$,

- $\vec{x}$: eigenvector and λ: eigenvalue



Source: Lyudmil Antonov Lantonov
https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors

Principal Component Analysis

Let's work through a code!

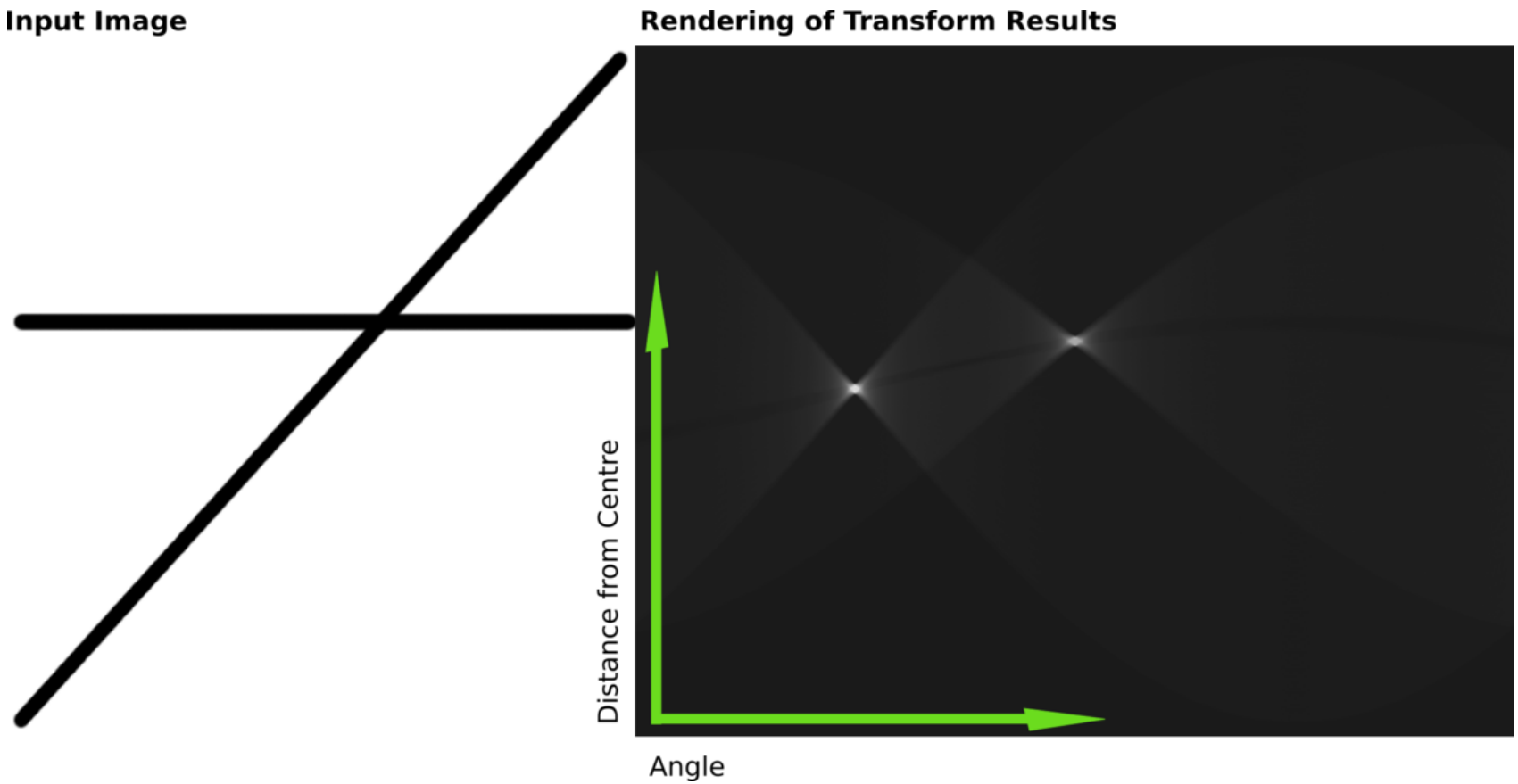
Feature Extraction : Hand Engineered

- Edge detection
- Shape detection
- Thresholding
- SIFT
- ...

Feature Extraction : Hand Engineered - Edge detection

Feature Extraction : Hand Engineered - Edge detection

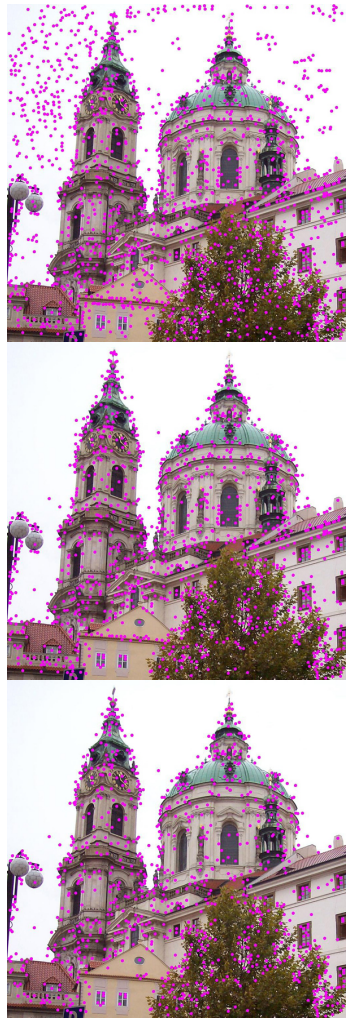
Feature Extraction : Hand Engineered - Shape detection



source: https://en.wikipedia.org/wiki/Hough_transform

Feature Extraction : Hand Engineered - Thresholding

Feature Extraction : Hand Engineered - SIFT

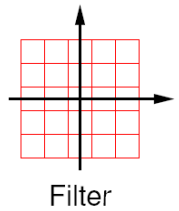


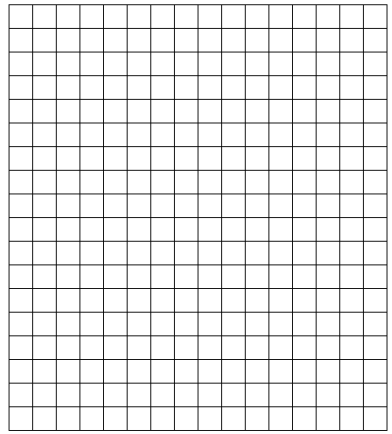
source: https://en.wikipedia.org/wiki/Scale-invariant_feature_transform

Linear Filters, Correlation and Convolution

Let $I(X, Y)$ a $n \times n$ image
Let $F(X, Y)$ a $m \times m$ image ("filter")

- Here $m = 5$. Let $k = m/2 = 2$





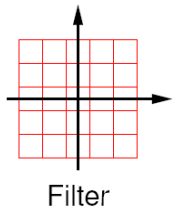
Compute a new image as:

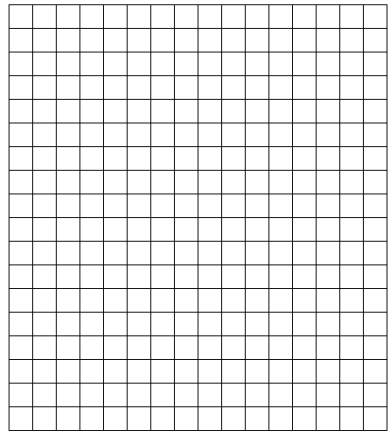
$$I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j) I(X + i, Y + j)$$

Linear Filters, Correlation and Convolution

Let $I(X, Y)$ a $n \times n$ image
Let $F(X, Y)$ a $m \times m$ image ("filter")

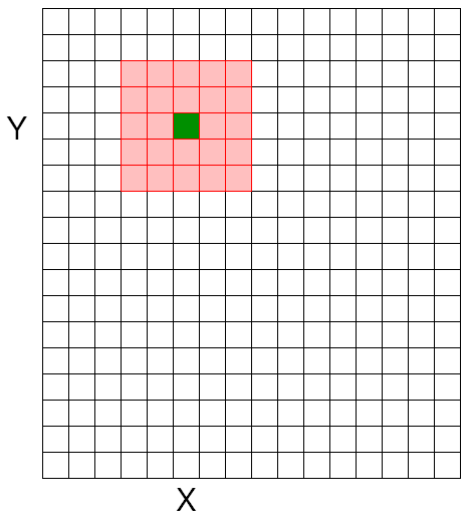
- Here $m = 5$. Let $k = m/2 = 2$

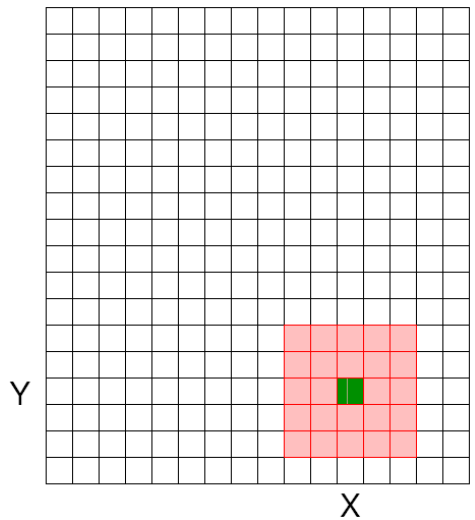




Compute a new image as:

$$I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j) I(X + i, Y + j)$$



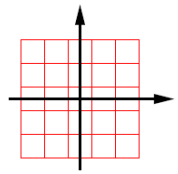


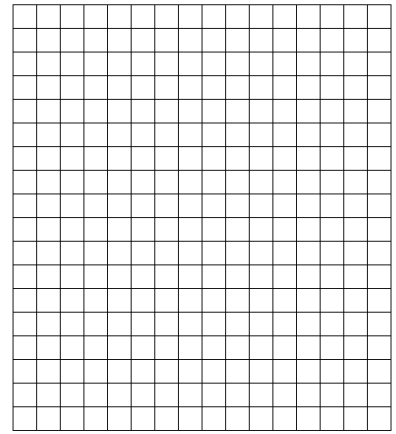
Source: Bob Woodham, UBC. CPSC 425: Computer Vision lecture notes

Linear Filters, Correlation and Convolution

Let $I(X, Y)$ a $n \times n$ image
Let $F(X, Y)$ a $m \times m$ image ("filter")

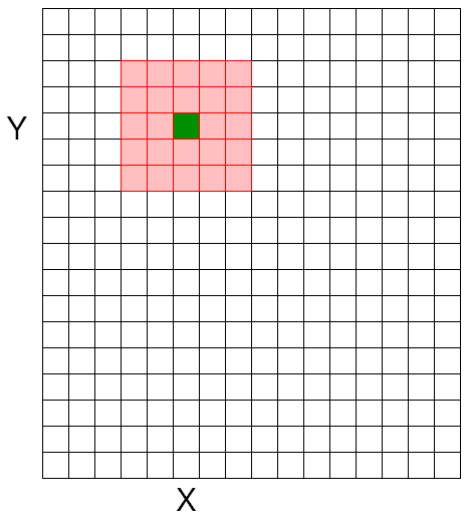
- Here $m = 5$. Let $k = m/2 = 2$

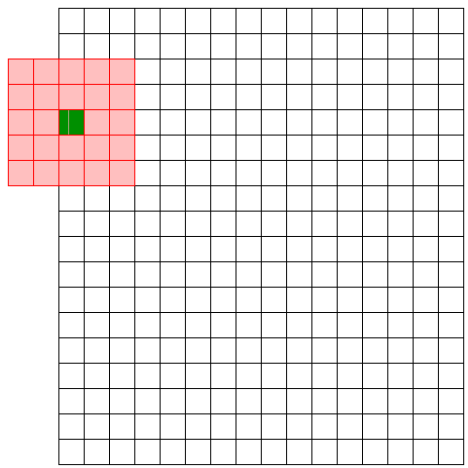




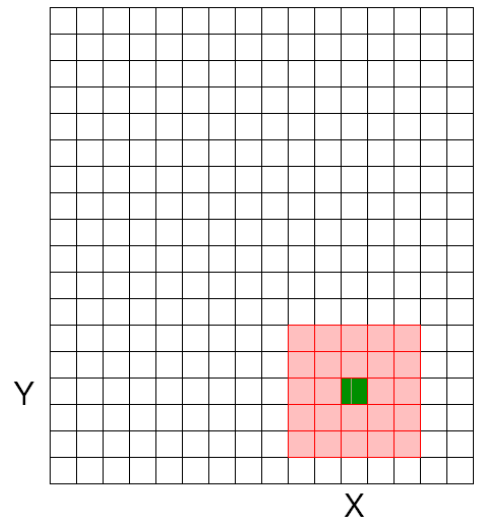
Compute a new image as:

$$I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j) I(X + i, Y + j)$$





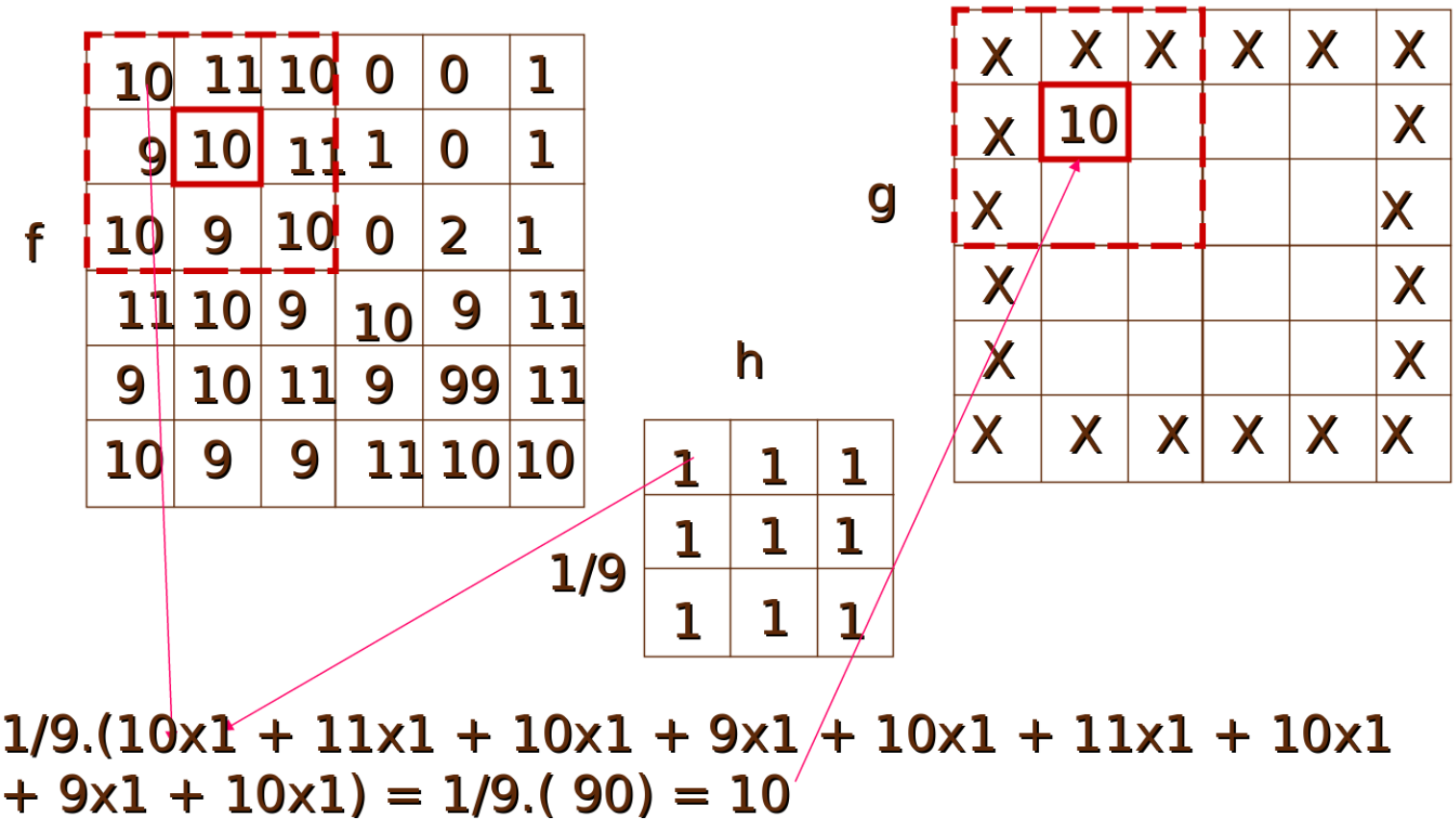
Boundary effects



Source: Bob Woodham, UBC. CPSC 425: Computer Vision lecture notes

Computation Example

$$g(X,Y) = \sum_{j=-1}^{1} \sum_{i=-1}^{1} h(i,j) f(X+i, Y+j)$$



Source: Jana Kosecka, Stanford CS223b Computer Vision, Winter 2007

Linear Filtering

Original

0	0	0
0	1	0
0	0	0

Filter

?

Result

Source: Bob Woodham, UBC. CPSC 425: Computer Vision lecture notes

Linear Filtering

0	0	0
0	1	0
0	0	0

Original

Filter

Result

(no change)

Source: Bob Woodham, UBC. CPSC 425: Computer Vision lecture notes

Linear Filtering

Original

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

Filter

(filter sums to 1)

?

Result

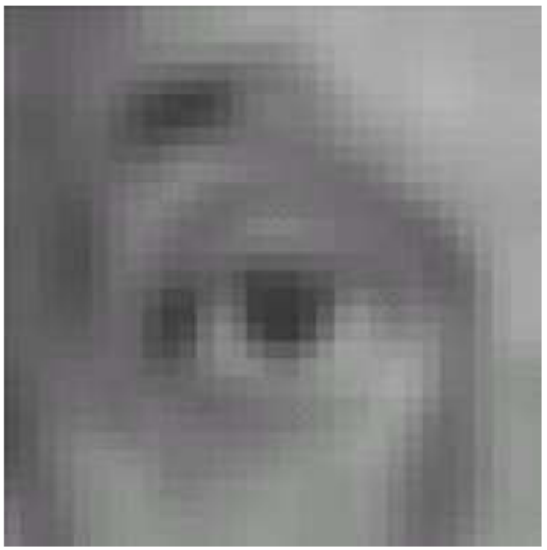
Source: Bob Woodham, UBC. CPSC 425: Computer Vision lecture notes

Linear Filtering

Original

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

Filter



Result

(blur with a box filter)

Source: Bob Woodham, UBC. CPSC 425: Computer Vision lecture notes

Correlation and Convolution

Correlation (previous slides)

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X+i, Y+j)$$

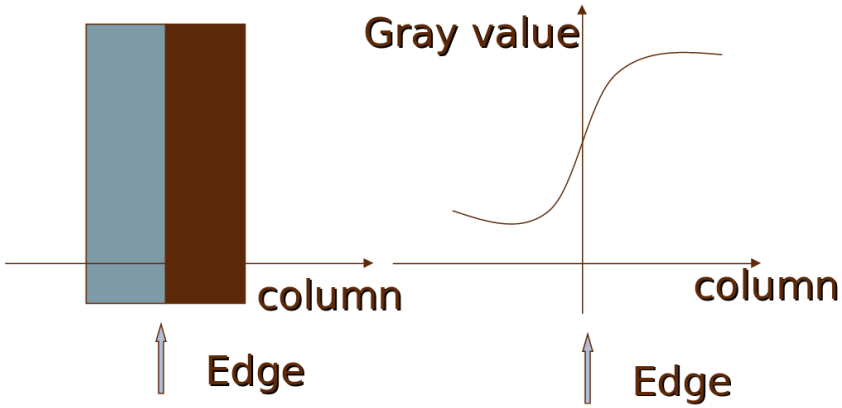
Convolution

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X-i, Y-j)$$

$$= \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(-i,-j) I(X+i, Y+j)$$

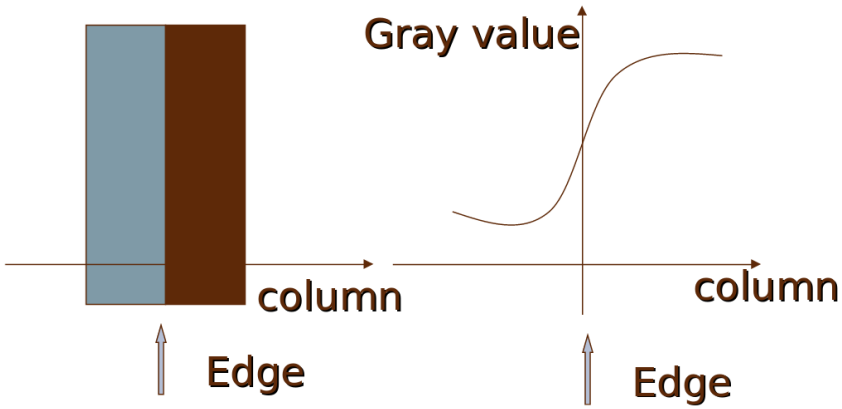
Note: If $F(X,Y) = F(-X,-Y)$ then correlation = convolution
i.e. if $F(X,Y)$ symmetric

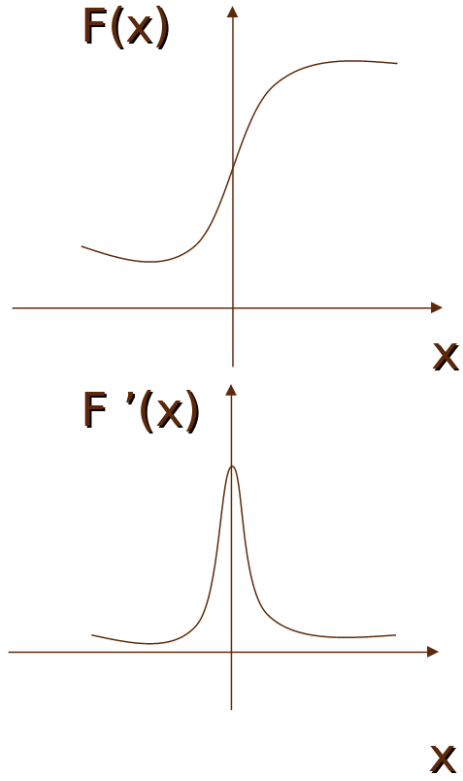
Edges in Images



Edge = Sharp variation in images

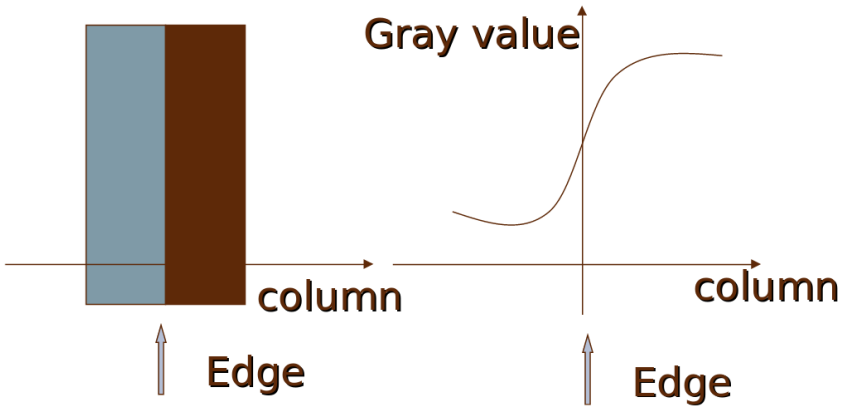
Edges in Images

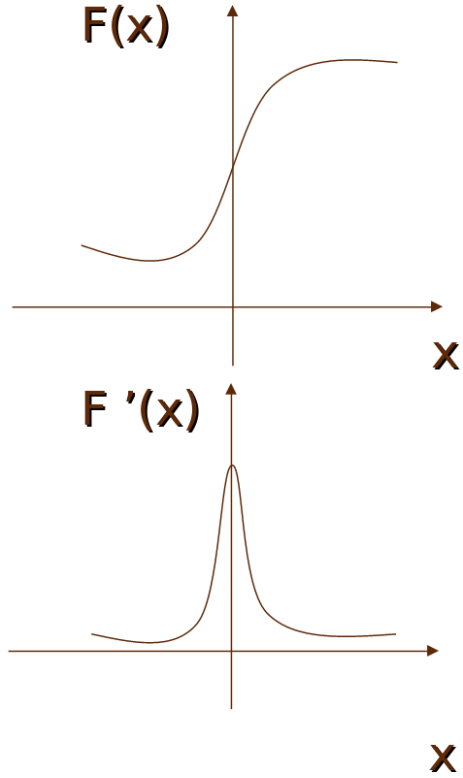




Edge = Sharp variation in images
Edge $\Rightarrow$ Large first derivative

Edges in Images



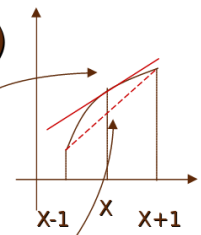


Edge = Sharp variation in images

Edge $\Rightarrow$ Large first derivative

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

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

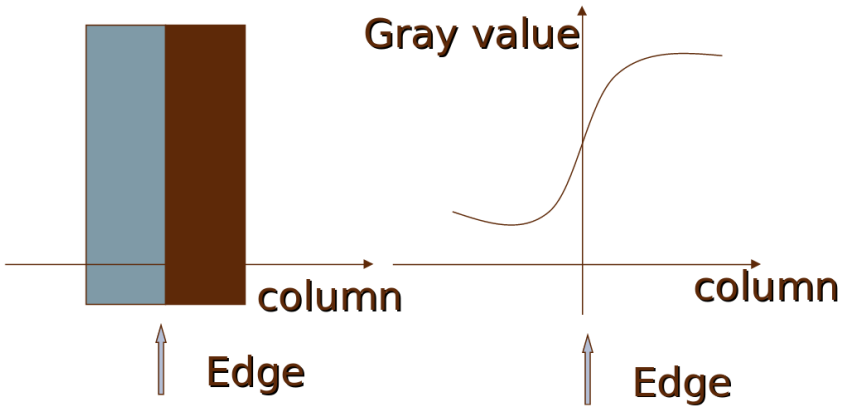


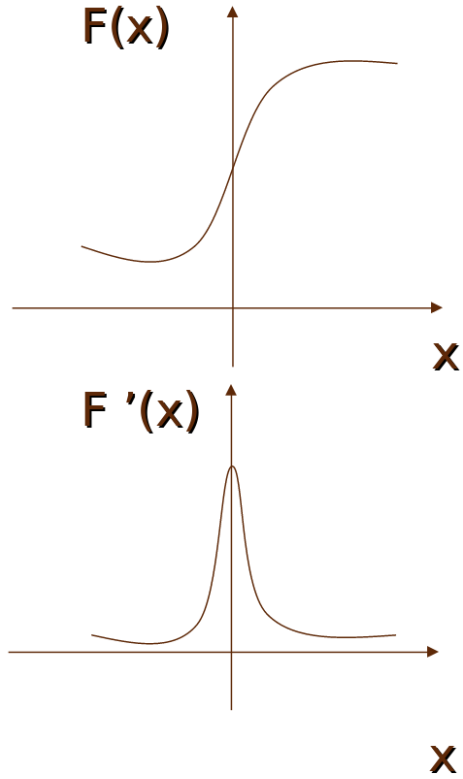
Convolve with

-1	0	1

Source: Jana Kosecka, Stanford CS223b Computer Vision, Winter 2007

Edges in Images

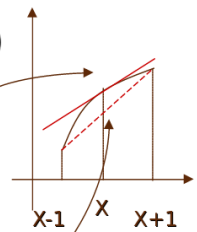




Edge = Sharp variation in images

Edge $\Rightarrow$ Large first derivative

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



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

Convolve with

-1	0	1

Source: Jana Kosecka, Stanford CS223b Computer Vision, Winter 2007

Vertical edges convolve with

-1	0	1

Horizontal edges convolve with

-1
0
1

Sobel Edge Detection

Input image I

Output image

Davidwkennedy,

http://en.wikipedia.org/wiki/File:Bikesgray.jpg

Sobel Edge Detection

Vertical edge detection

$$I \star \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \text{ gives } \frac{\partial I}{\partial x} = I_x$$

Horizontal edge detection:

$$I \star \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array} \text{ gives } \frac{\partial I}{\partial y} = I_y$$

Input image I

Output image

Davidwkennedy,

http://en.wikipedia.org/wiki/File:Bikesgray.jpg

Sobel Edge Detection

Vertical edge detection

$$I \star \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \text{ gives } \frac{\partial I}{\partial x} = I_x$$

Horizontal edge detection:

$$I \star \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array} \text{ gives } \frac{\partial I}{\partial y} = I_y$$

Each I_x and I_y is an image.

The image gradient:

$$\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$

Magnitude of gradient image:

$$||\nabla I|| = \sqrt{I_x^2 + I_y^2}$$

Input image I

Output image $||\nabla I||$

Davidwkennedy,

http://en.wikipedia.org/wiki/File:Bikesgray.jpg

Feature Extraction

Two types of feature

- Hand Engineered

Feature Extraction

Two types of feature

- Hand Engineered
  - easy to understand

Feature Extraction

Two types of feature

- Hand Engineered
  - easy to understand
  - if designed well, delivers good performance with small computation overhead

Feature Extraction

Two types of feature

- Hand Engineered
  - easy to understand
  - if designed well, delivers good performance with small computation overhead
  - requires a lot of expertise and domain knowledge

Feature Extraction

Two types of feature

- Hand Engineered
  - easy to understand
  - if designed well, delivers good performance with small computation overhead
  - requires a lot of expertise and domain knowledge
- Learned from data

Feature Extraction

Two types of feature

- Hand Engineered
  - easy to understand
  - if designed well, delivers good performance with small computation overhead
  - requires a lot of expertise and domain knowledge
- Learned from data
  - quite popular these days (Why ?)

Feature Extraction

Two types of feature

- Hand Engineered
  - easy to understand
  - if designed well, delivers good performance with small computation overhead
  - requires a lot of expertise and domain knowledge
- Learned from data
  - quite popular these days (Why ?)
  - requires a lot of data and computational might

Feature Extraction

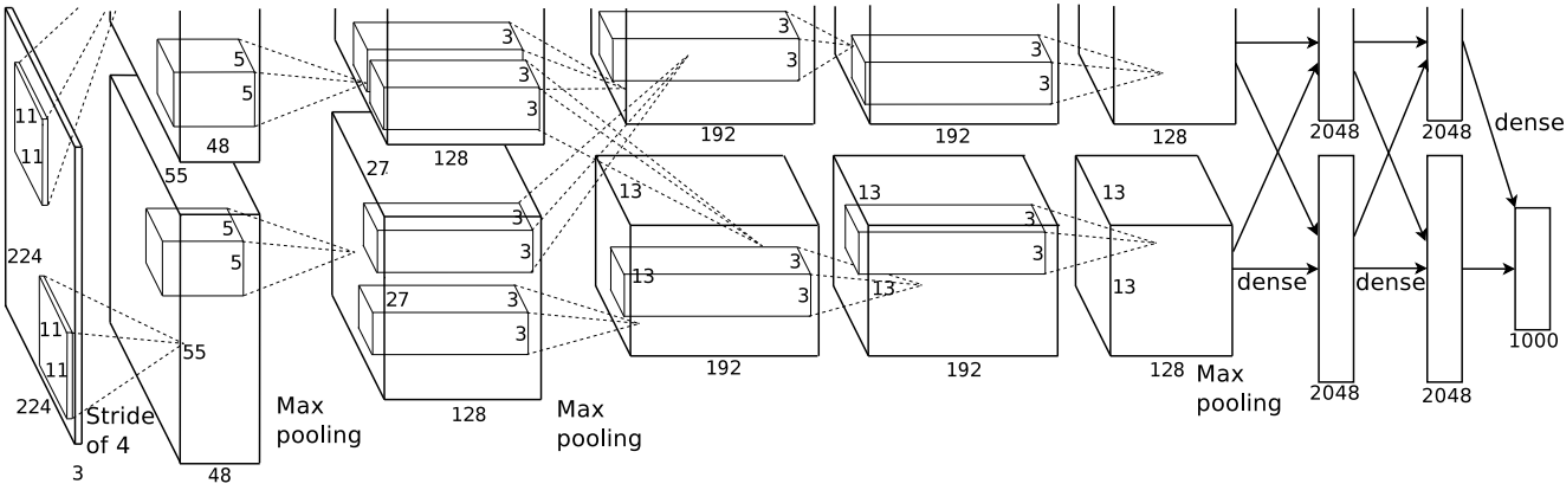
Two types of feature

- Hand Engineered
  - easy to understand
  - if designed well, delivers good performance with small computation overhead
  - requires a lot of expertise and domain knowledge
- Learned from data
  - quite popular these days (Why ?)
  - requires a lot of data and computational might
  - difficult (and sometime impossible) to understand (a black box)

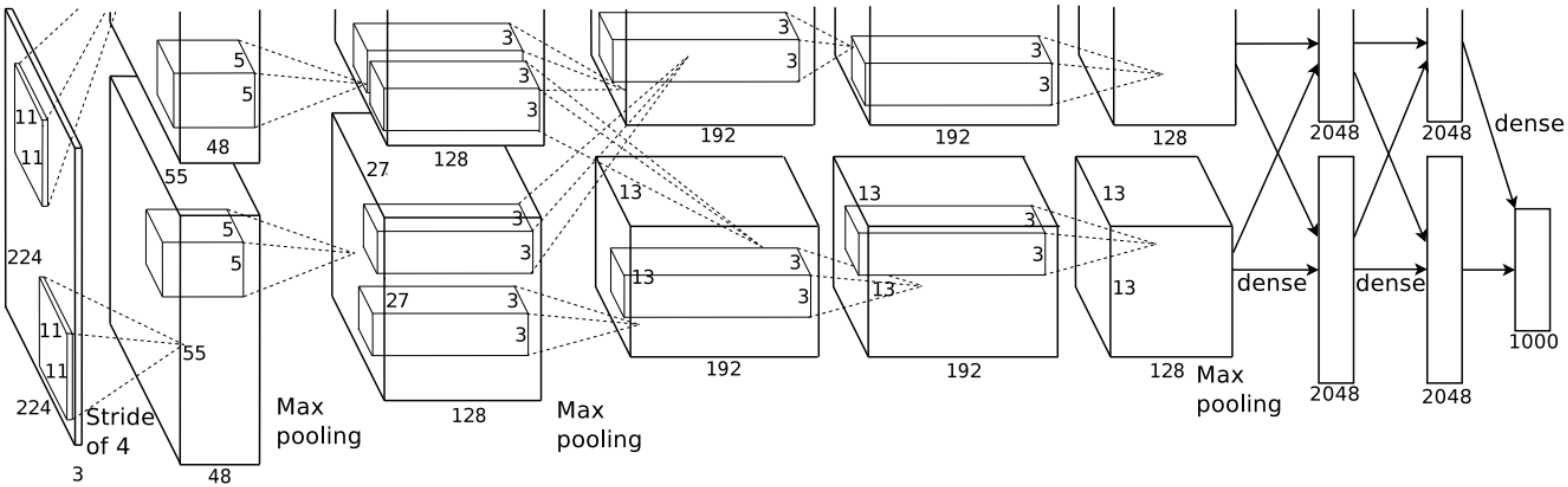
Feature Extraction : Learned from data

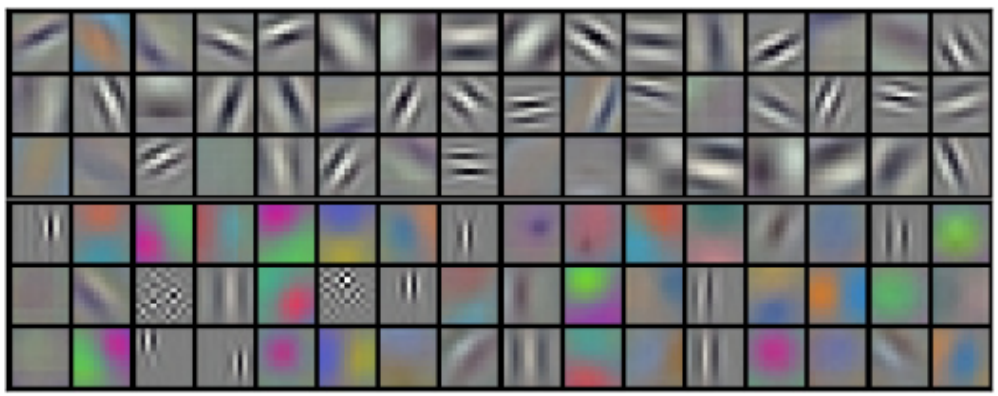
- Convolution Neural Network
- Auto-encoder
- ...

Feature Extraction : Learned from data - CNN



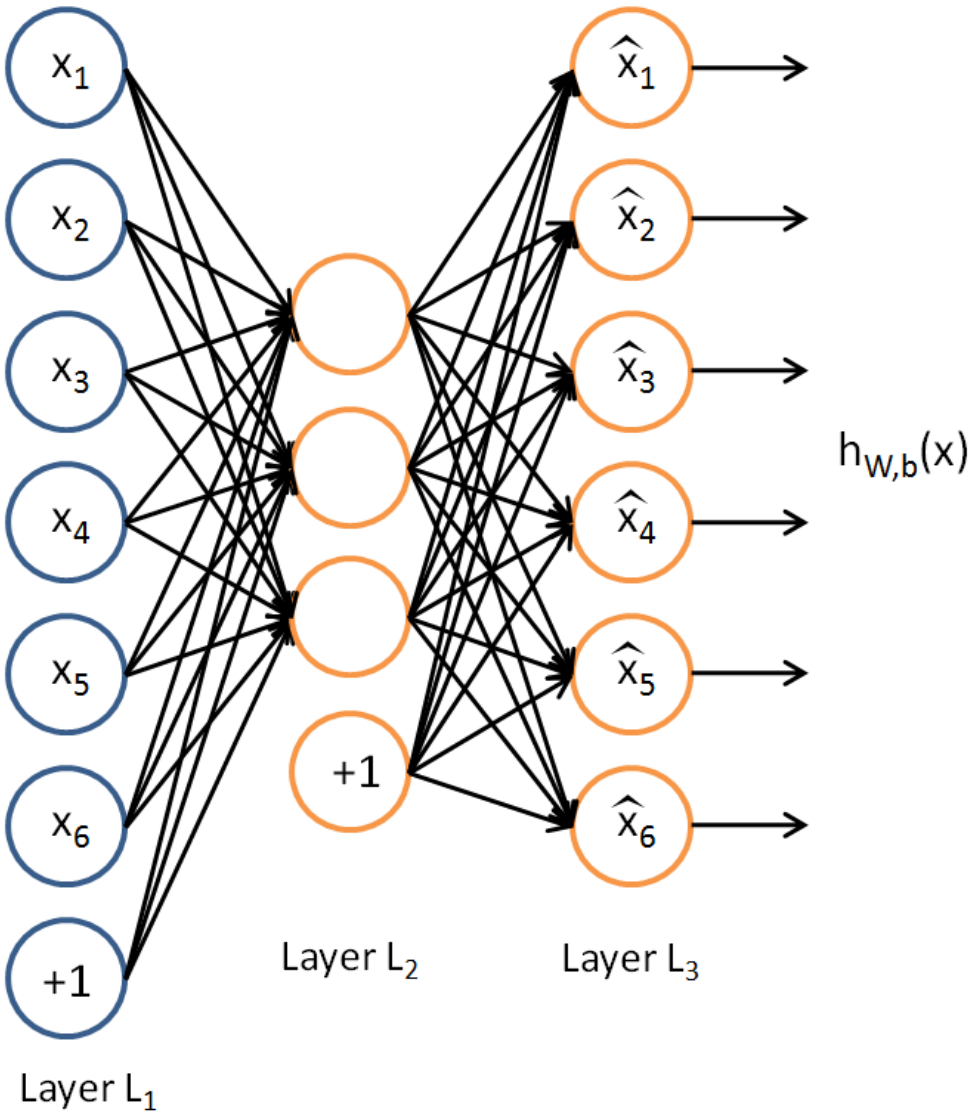
Feature Extraction : Learned from data - CNN





source: http://colah.github.io/posts/2014-07-Conv-Nets-Modular/

Feature Extraction : Learned from data - Autoencoder



source: http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/

Thank You