Computer Vision

Lec 5: Image DownSampling

Dr. Pratik Mazumder

Course Project Instructions

- Minimum size of group is 4 and Maximum size is 5

- Topic has to be related to Computer Vision
    - **OCR based projects are not allowed.**
    - Topics involving Audio, Text, and other non Image data are **not allowed. Primary task has to be based on image data.**

    - Every group has to apply **atleast two different approaches** to the selected topic.
        - Ideal case: 4 approaches - 1 for each member

    - Project has to be deployed on online platforms such as streamlit.
    - Deep Learning methods are allowed but the topic has to be related to computer vision.
    - New Findings/Analysis will be appreciated.

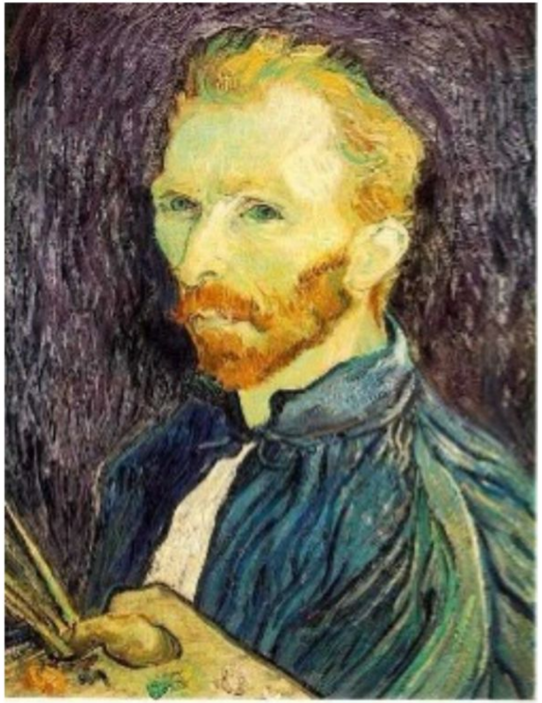
- More Instructions Later

Image downsampling

- Reducing the size of an image **WITHOUT LOSING MUCH INFORMATION**
- Why
    - Storage
    - Computing Power
    - Transmission Bandwidth

Naive Image downsampling

Throw away half the rows and columns



1/2

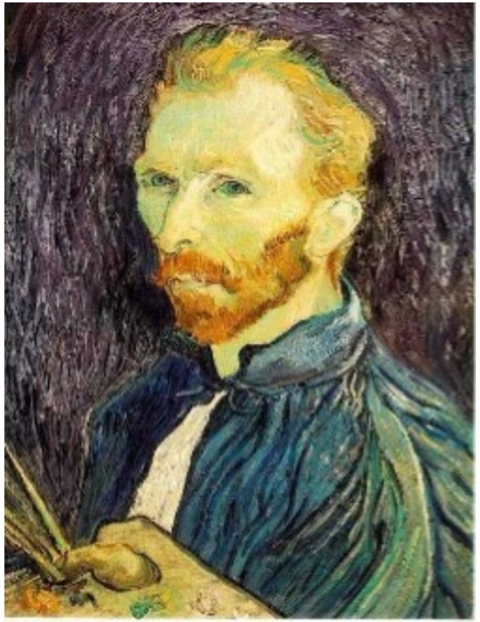
delete even rows
delete even columns

1/4

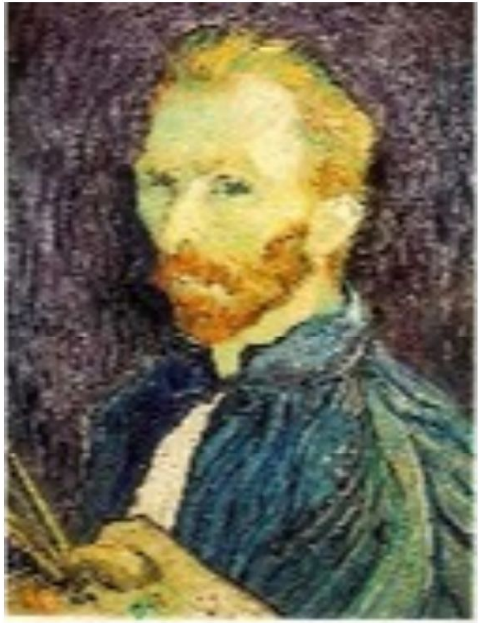
delete even rows
delete even columns

1/8

Naive Image downsampling



1/2

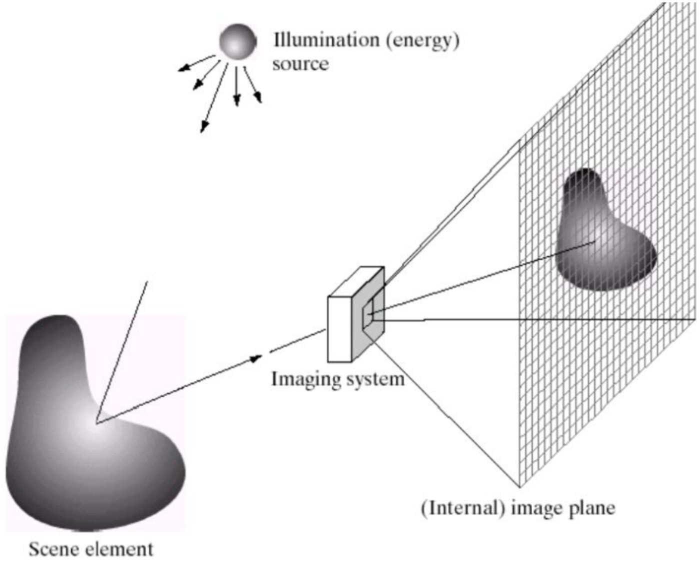


1/4 (2x zoom)

1/8 (4x zoom)

The ⅛ image is highly pixelated. Some artificial artifacts appear. This effect is called aliasing.
Phenomenon where a sampled representation of an image exhibits undesirable artifacts or distortions.

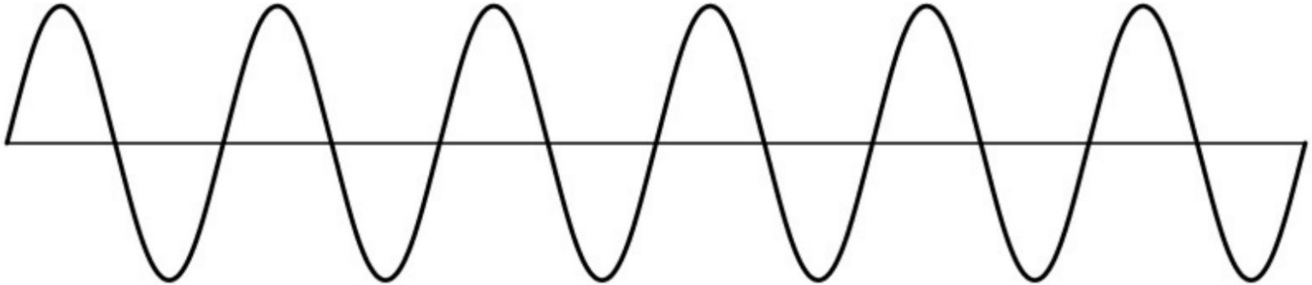
Reminder: Digital Image



Digital Images are a discrete, or sampled, representation of a continuous world

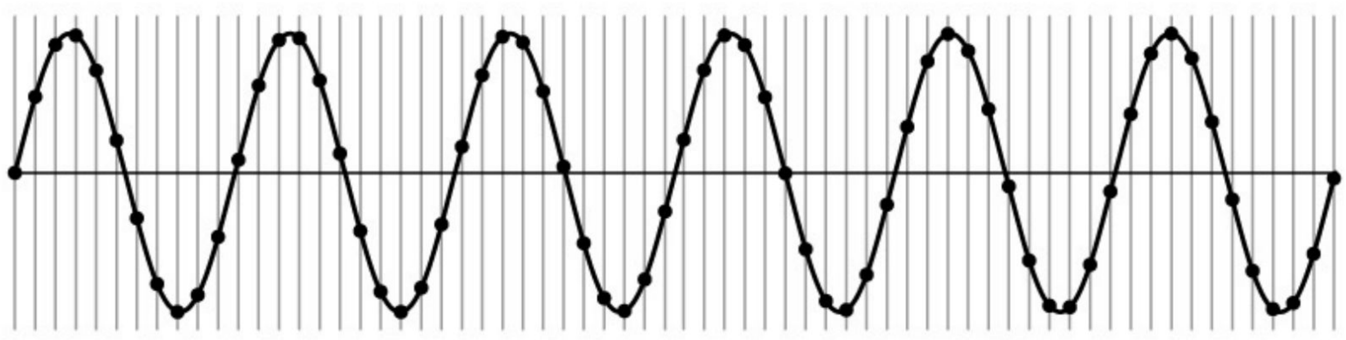
Sampling

Very simple example: a sine wave

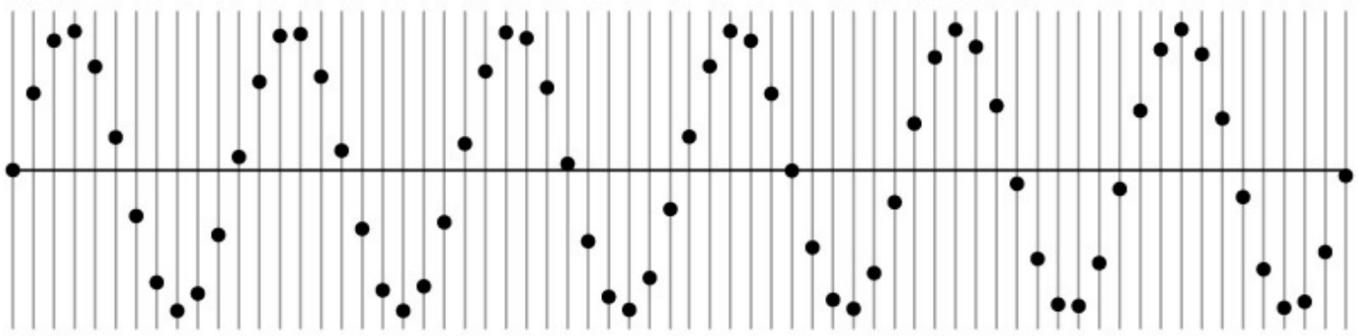


How would you discretize this signal?

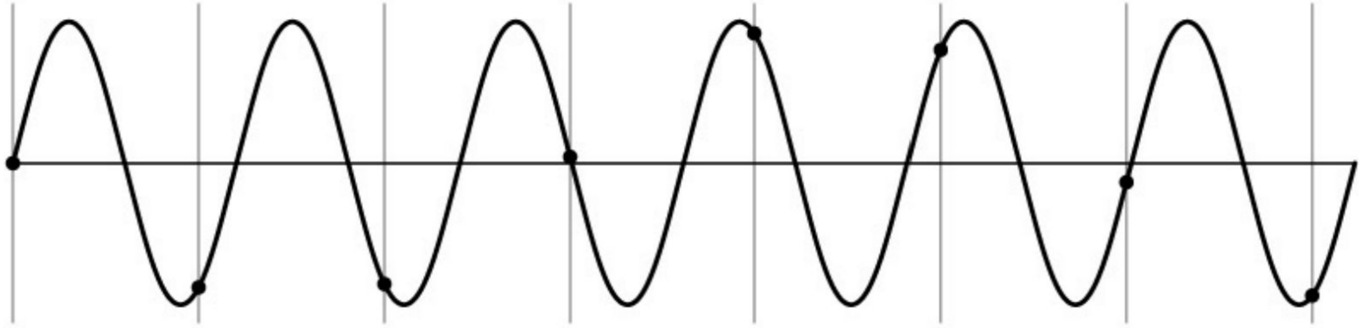
Sampling



Sampling

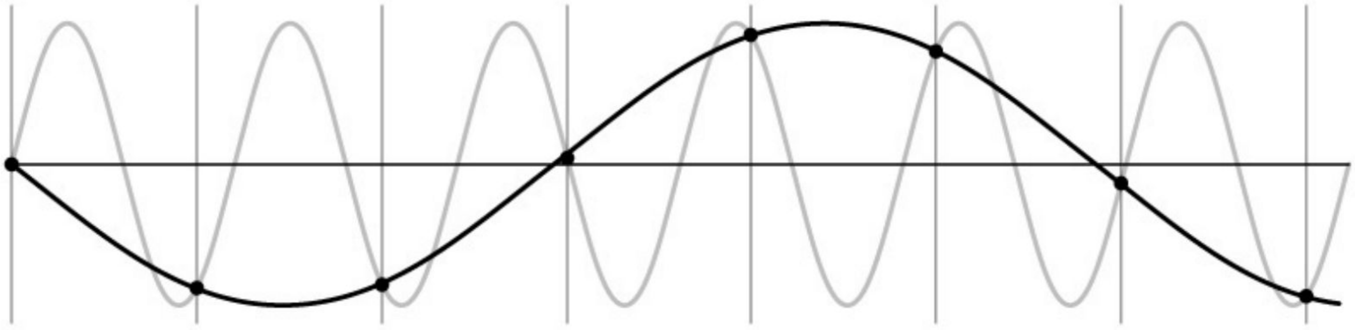


Undersampling



Taking very few samples: information is lost

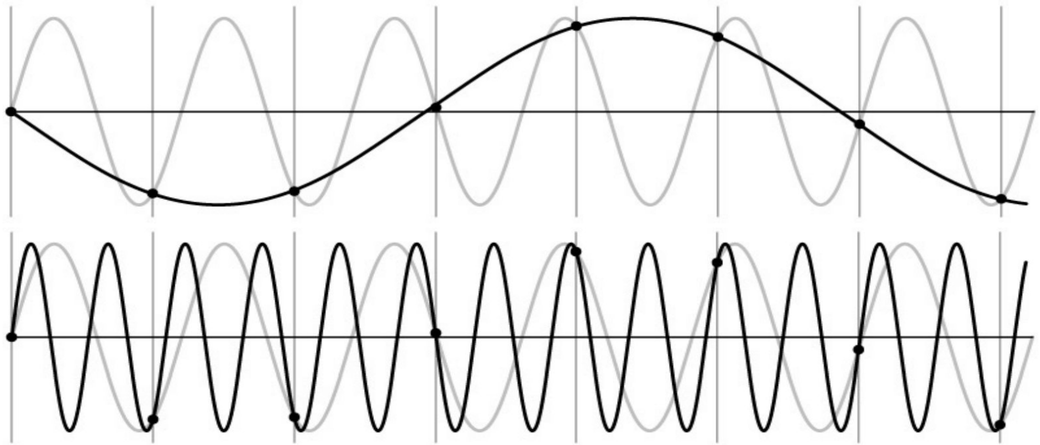
Undersampling



Taking very few samples: information is lost

Aliasing

- Aliasing refers to the distortion or artifacts that occur when a signal (such as an image or sound) is sampled or represented at a resolution that is insufficient to capture its details.
- It happens because high-frequency components of the signal are misinterpreted as lower-frequency components due to undersampling.
- Undersampling can disguise a signal as one of lower frequency.
- Can also be confused as a signal with higher frequency but low sampling rate



Aliasing in textures

Aliasing in photographs

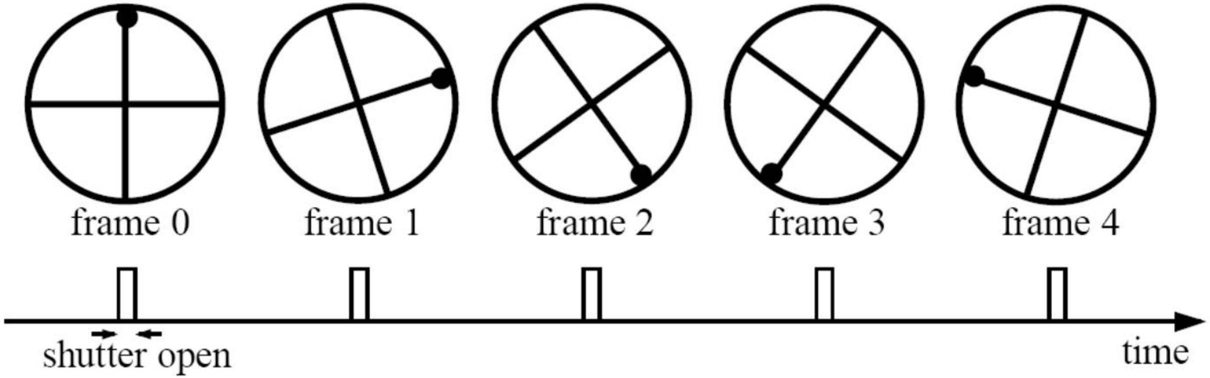
This is also known as "moire"

The moire patterns often appear as wavy or zigzag lines, curves, or other irregular shapes that were not present in the original scene.

Temporal aliasing

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

Wagon Wheel Effect

A visual phenomenon that occurs in film, animation, or real-life situations where a spoked wheel appears to rotate in a direction opposite to its actual rotation or appears to stand still, even though it is moving.

Two common scenarios leading to the wagon wheel effect:

- Under-Sampled Motion:
  - If the rate of capturing frames is not high enough and the wheel rotates quickly, the captured frames may not adequately represent the continuous motion. Instead, it might seem as if the wheel is moving more slowly or even in the opposite direction.

- Sampling Rate Matching Rotation Speed:
  - When the rotational speed of the wheel aligns with the frame capture rate, the spokes of the wheel might coincide with the captured frames, creating the illusion of the wheel standing still or moving slowly.

Wagon Wheel Effect

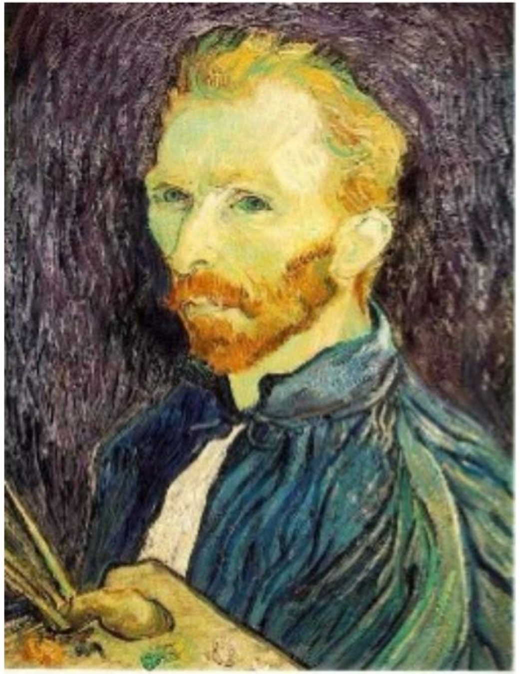
Anti-aliasing

Dealing with Aliasing

- Approach 1: Oversample the signal
- Approach 2: Smooth the signal
  - Remove some of the detail effects that cause aliasing.
  - Lose information, but better than aliasing artifacts

Better image downsampling

Apply a smoothing filter first, then throw away half the rows and columns



1/2

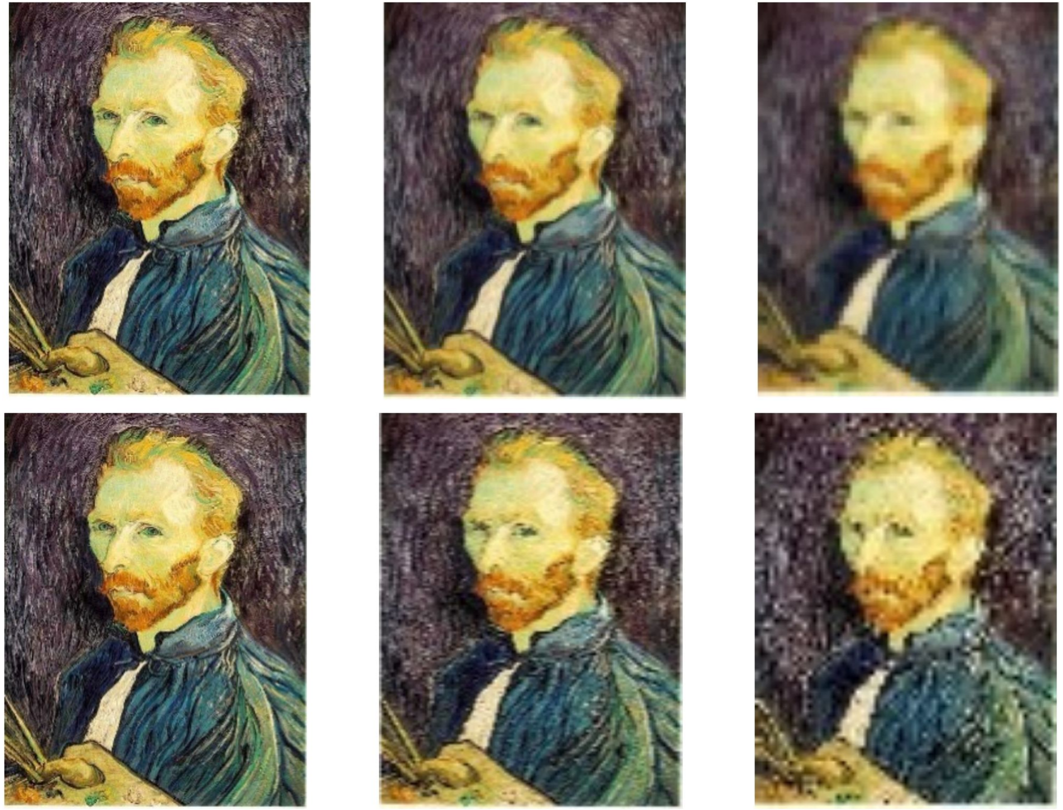
Gaussian filter
delete even rows
delete even columns

1/4

Gaussian filter
delete even rows
delete even columns

1/8

Better image downsampling

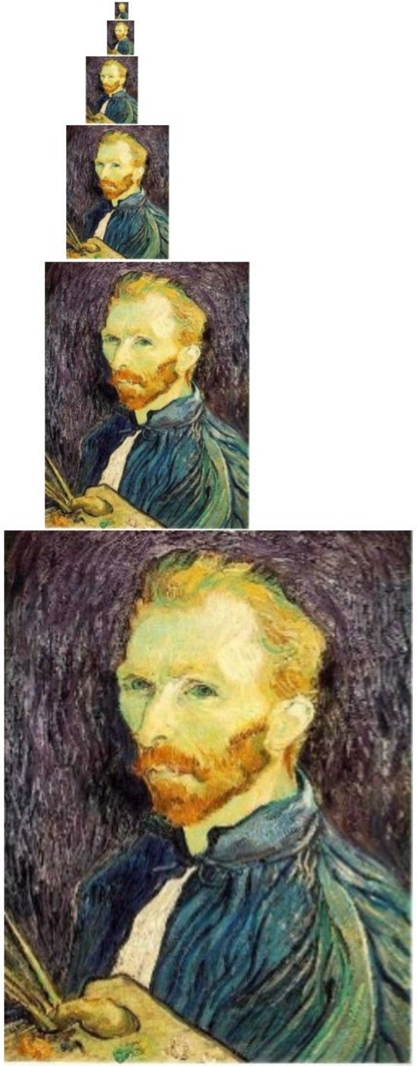


Naive

1/2 1/4 (2x zoom) 1/8 (4x zoom)

Gaussian image pyramid

- A multi-scale representation of an image that is created using a series of smoothed and downsampled versions of the original image.



Constructing a Gaussian pyramid

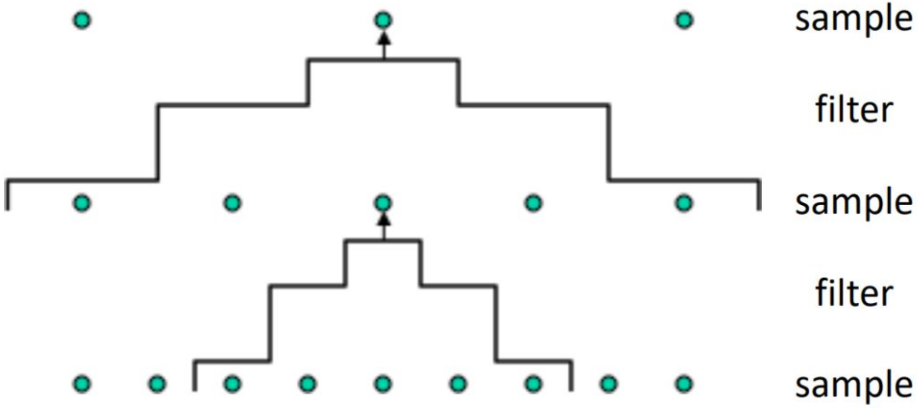
Algorithm

repeat:

filter

subsample

until min resolution reached



Question: How much bigger than the original image is the whole pyramid?

Constructing a Gaussian pyramid

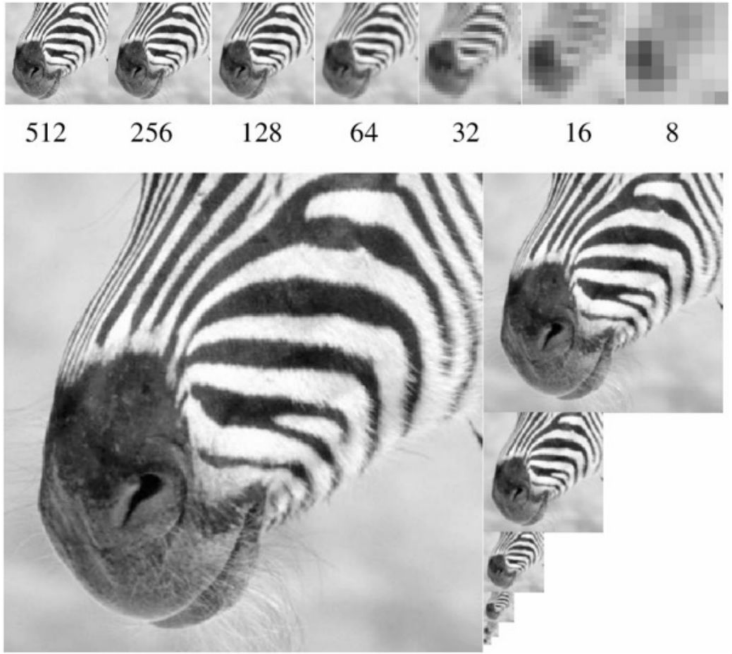
Let W be the width and H be the height of the original image.

After applying the Gaussian filter and downsampling by a factor of ½ the width and height become W/2 and H/2. Size reduction by a factor of 4

Next image, will have a size reduction by a factor of 16 compared to the original image and so on

$1 + 1/4 + 1/16 + 1/64 + \ldots = ?$

Constructing a Gaussian pyramid



What happens to the details of the image?

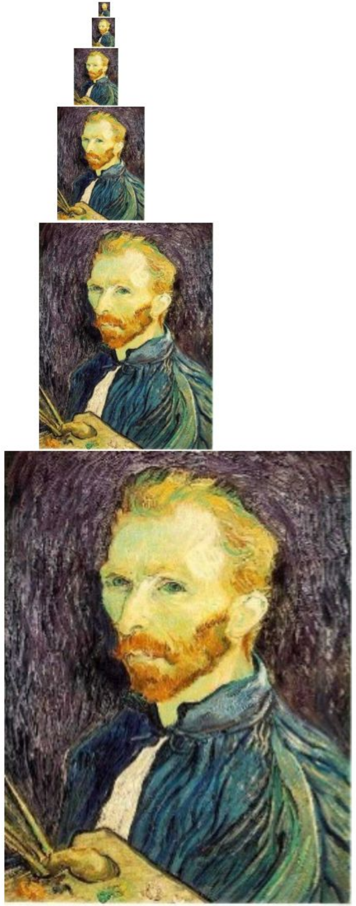
- They get smoothed out as we move to higher levels.

What is preserved at the higher levels?

- Mostly large, uniform regions in the original image

Reconstruction of the original image from the image at the upper level?

- Not good



Blurring is lossy

level 0

level 1 (before downsampling)

What does the residual look like?

Blurring is lossy

level 0

-

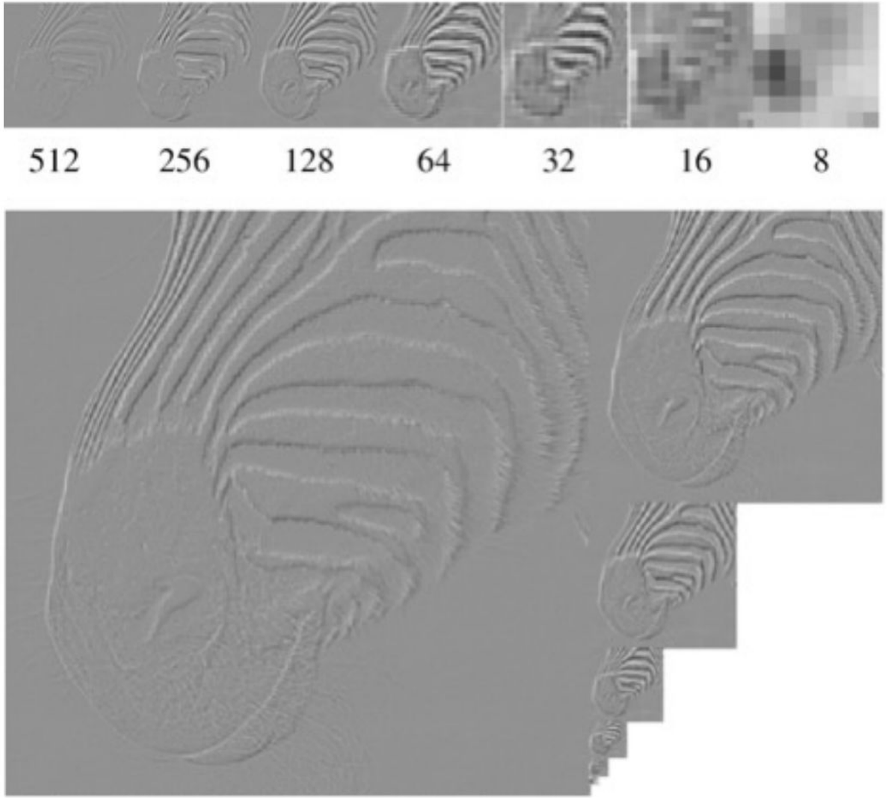
level 1 (before downsampling)

=

residual

Can we make a pyramid that is lossless?

Laplacian image pyramid



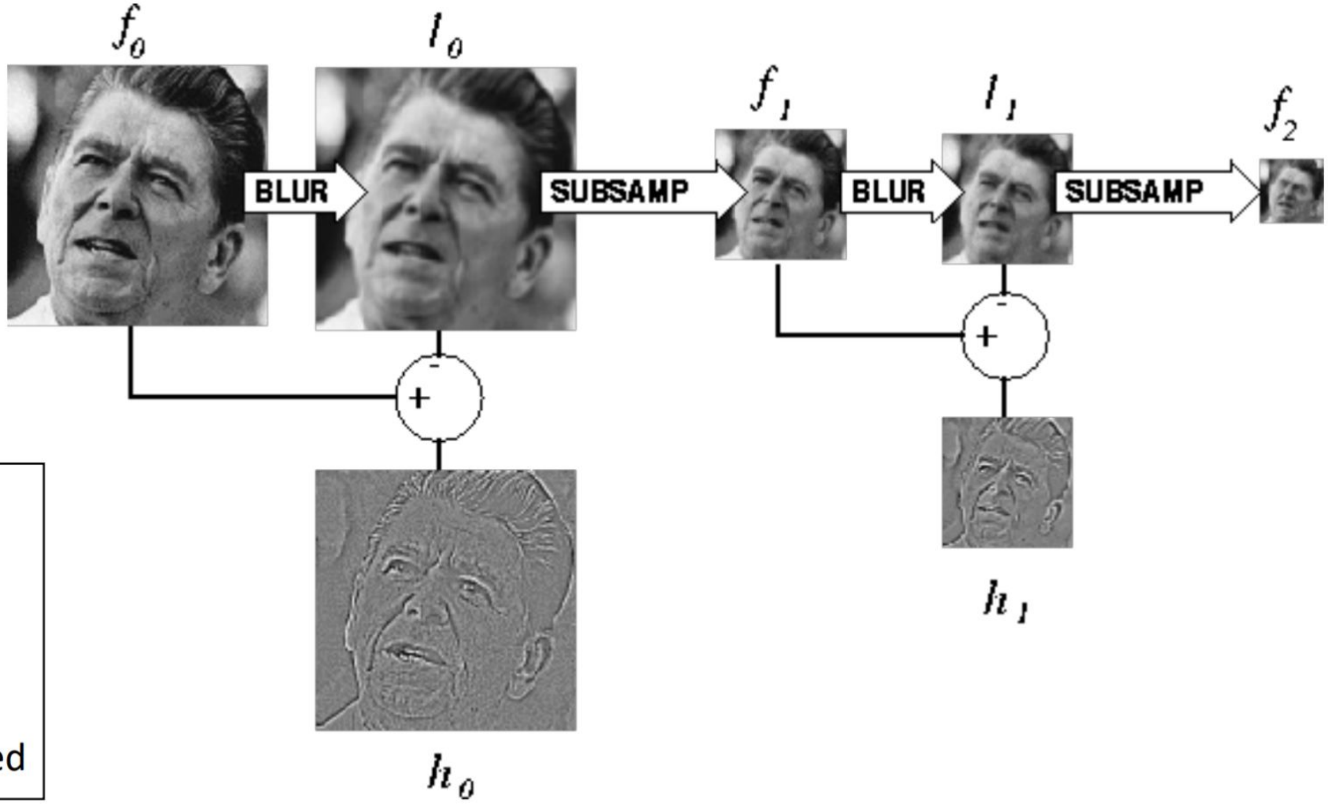
- At each level, retain the residuals instead of the blurred images themselves.
- Can we reconstruct the original image using the pyramid?
  - Yes we can!
- What do we need to store to be able to reconstruct the original image?

Let’s start by looking at just one level

level 0 = level 1 (upsampled) + residual

Does this mean we need to store both residuals and the blurred copies of the original?

Let's start by looking at just one level



Algorithm

repeat:

filter

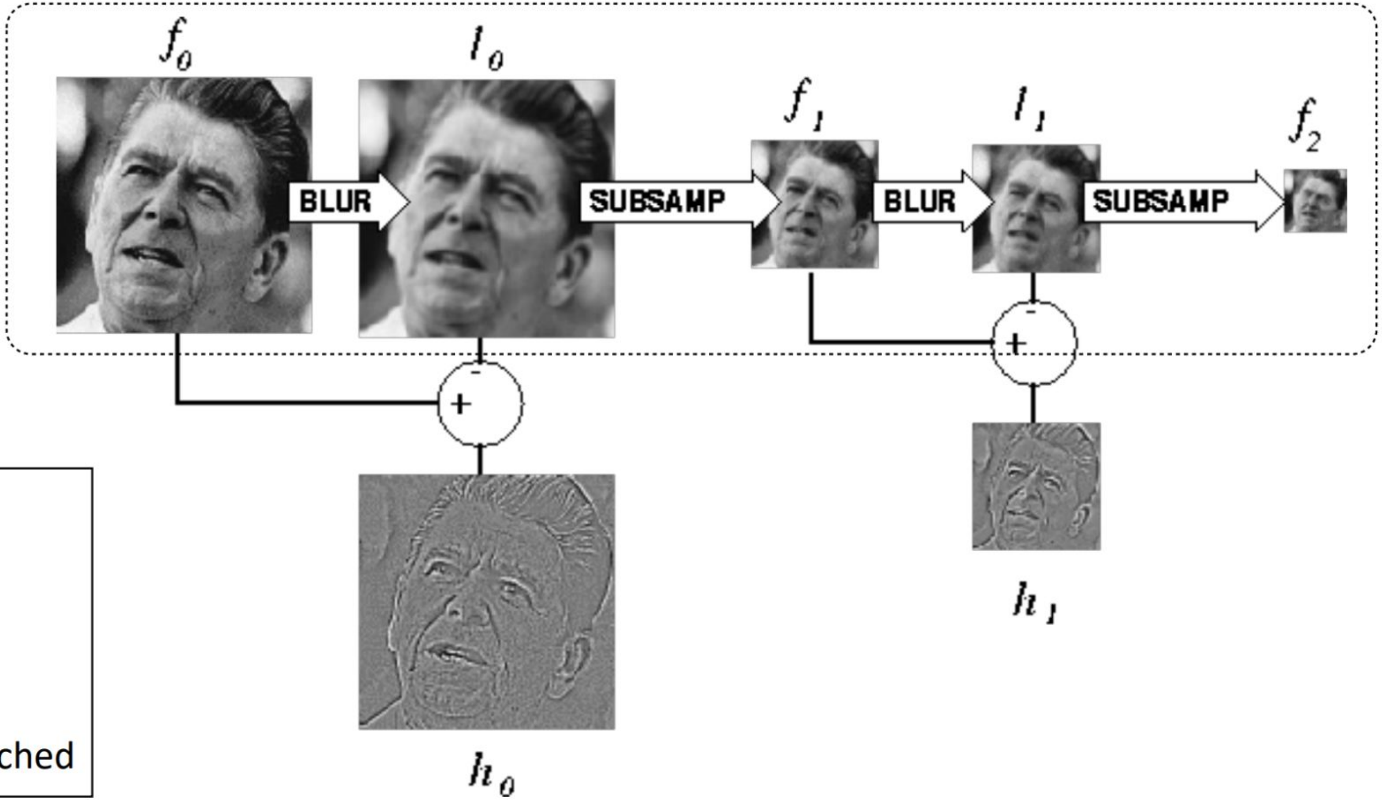
compute residual

subsample

until min resolution reached

Let's start by looking at just one level

What is this part?



Algorithm

repeat:

filter

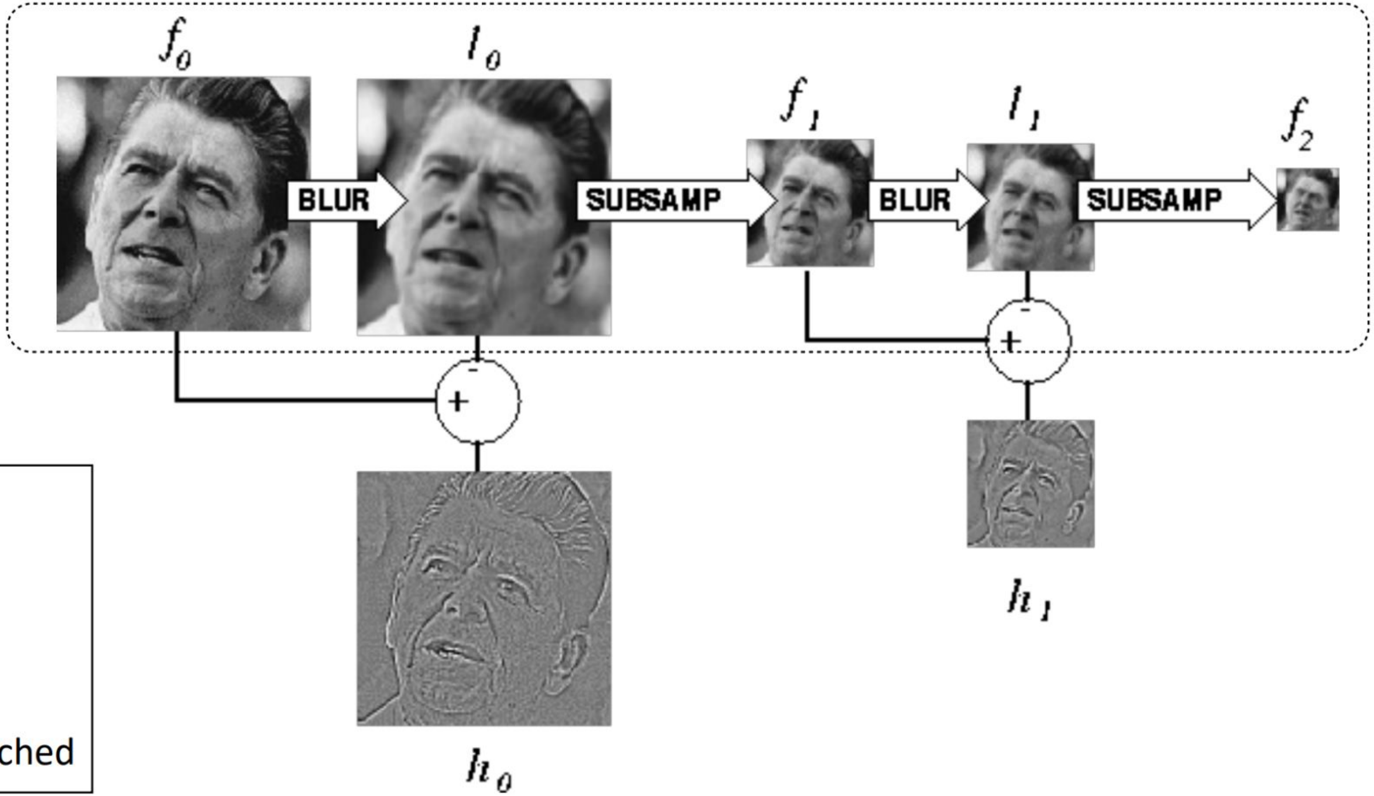
compute residual

subsample

until min resolution reached

Let's start by looking at just one level

It's a Gaussian pyramid.



Algorithm

repeat:

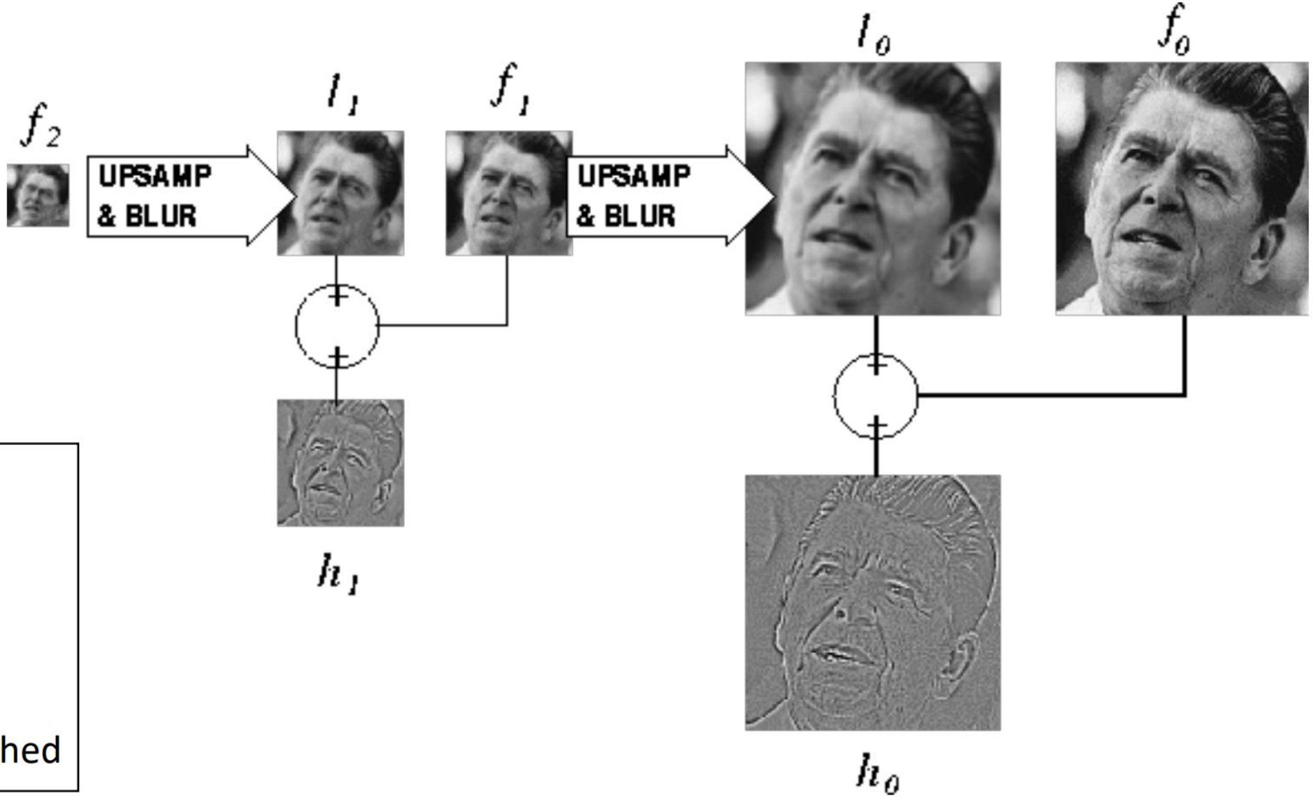
filter

compute residual

subsample

until min resolution reached

Reconstructing the original image



Algorithm

repeat:

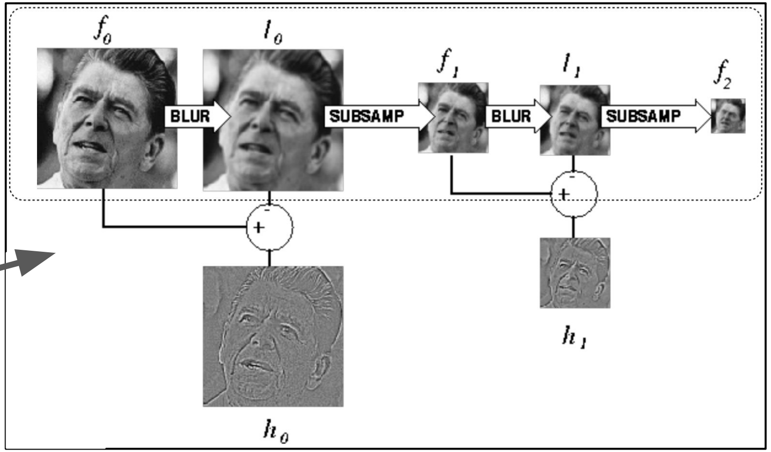
upsample

sum with residual

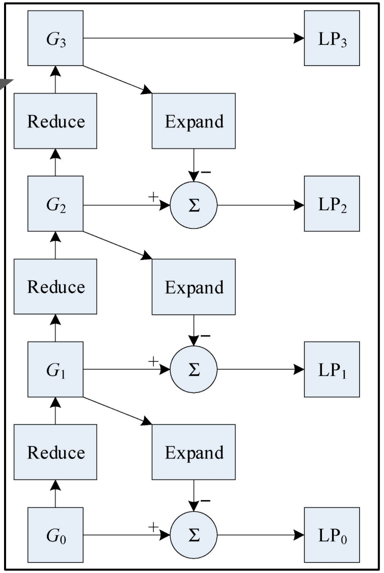
until orig resolution reached

Laplacian Pyramid Construction Approaches

1. Can be created in parallel with the Gaussian Pyramid (demonstrated in the previous slides).

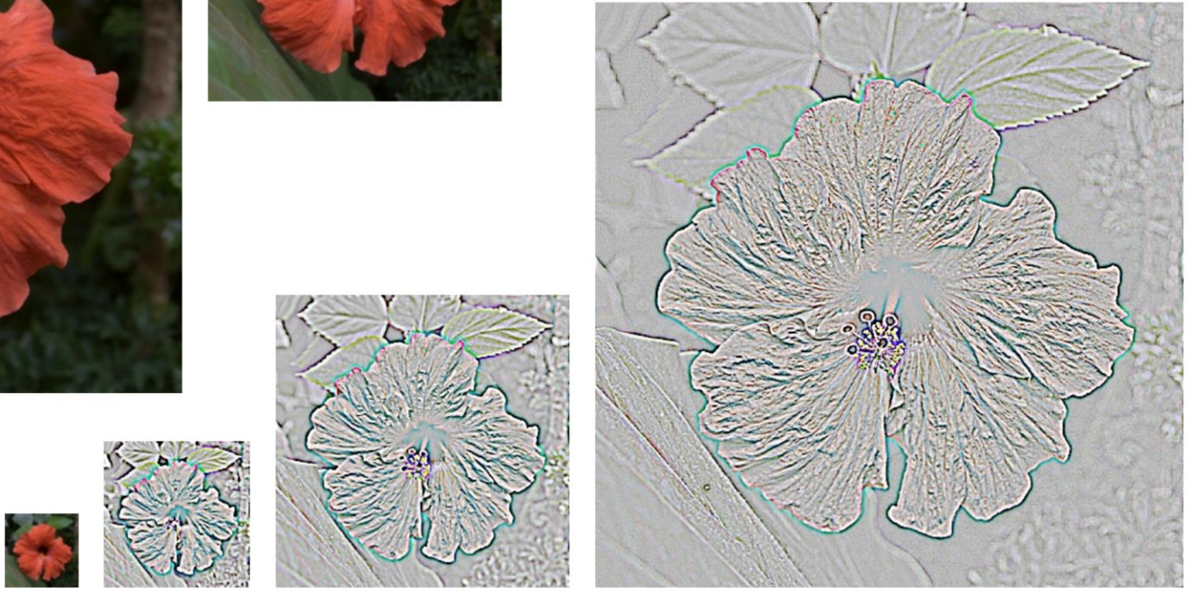


2. Can also be created from the Gaussian Pyramid starting from the last level of the Gaussian pyramid
   - Involves upsampling and blurring the image from Level L to make it the same size as the previous level (L-1) Gaussian pyramid image.
   - The difference can then be calculated between the upsampled and blurred image of level L with the level L-1 Gaussian pyramid image.



Gaussian vs Laplacian Pyramid

Shown in opposite order for space.



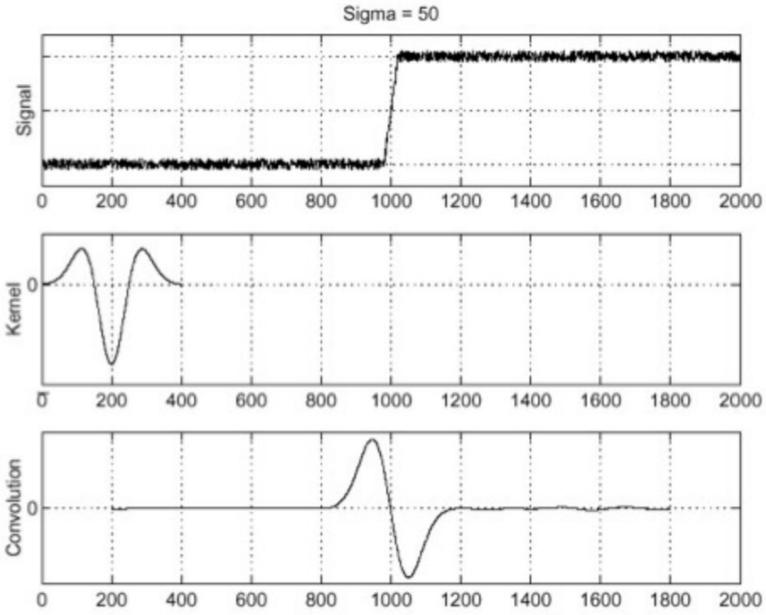
Reminder: Laplacian of Gaussian (LoG) filter

As with derivative, we can combine Laplace filtering with Gaussian filtering

input

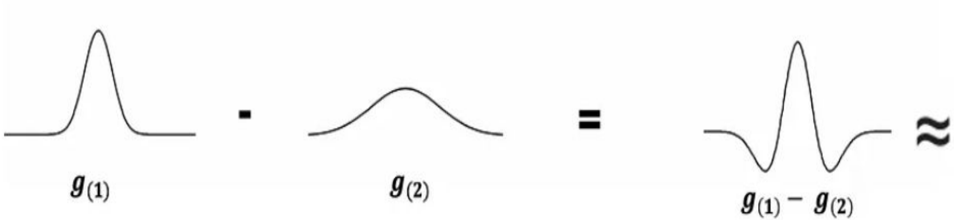
Laplacian of Gaussian

output

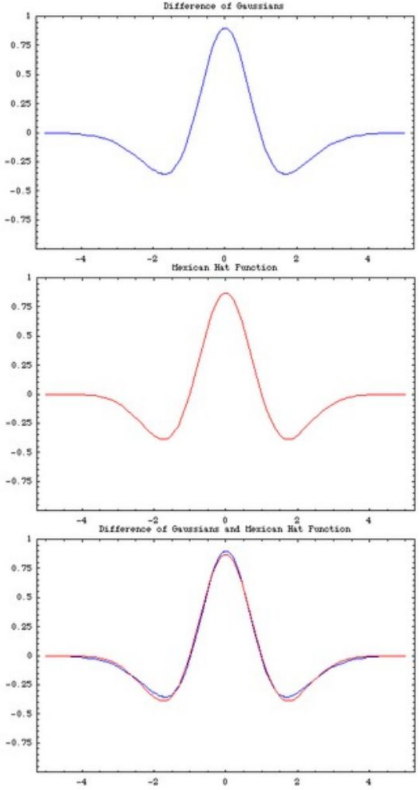


"zero crossings" at edges

Why is it called a Laplacian pyramid?

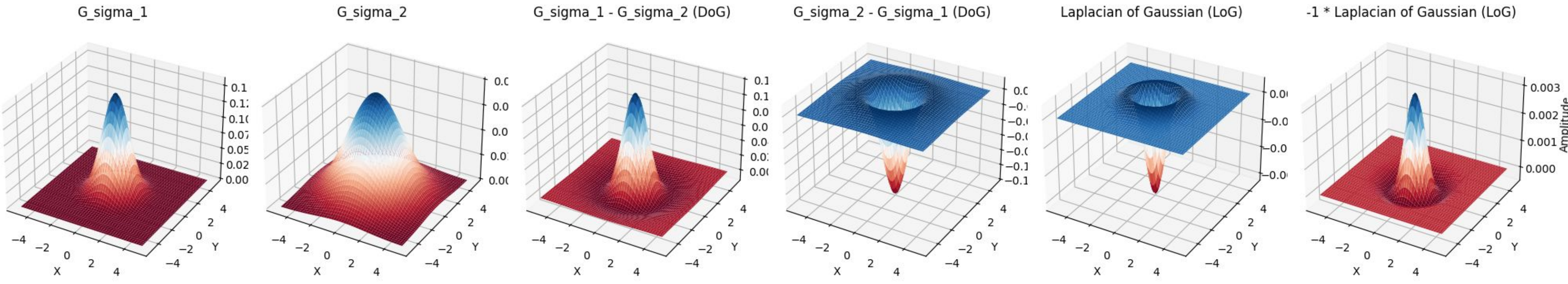


≈ **Laplacian of Gaussian (LOG)**



Difference of Gaussians approximates the Laplacian

Why is it called a Laplacian pyramid?



What are image pyramids used for?

image compression

multi-scale texture mapping

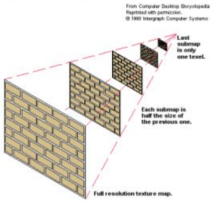
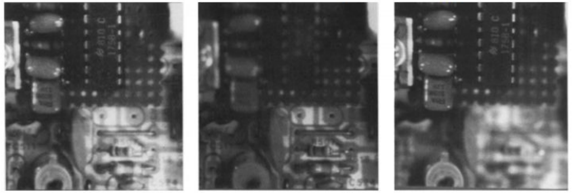
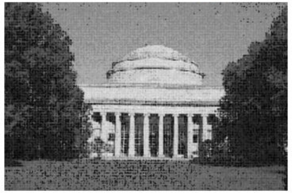


image blending

focal stack compositing

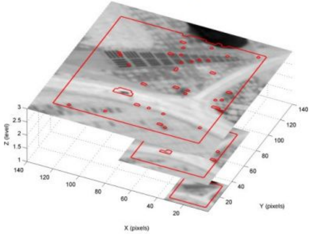


denoising



multi-scale detection

multi-scale registration



Simple Image UpSampling: Nearest Neighbor

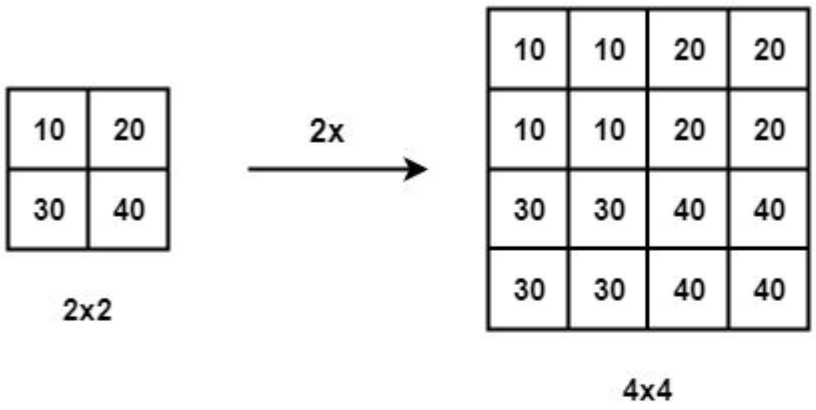


Image Blending using Laplacian Pyramid

- Input Images: Start with two images, A and B, that you want to blend. Ensure they have the same size. If not, resize them to match.

- Define a Mask: Create a binary or grayscale mask M of the same size as the input images.
  - This mask determines how much of each image will contribute to the blend.
  - For example:
    - If M[i,j] = 1, only image A contributes at pixel (i, j).
    - If M[i,j] = 0, only image B contributes at pixel (i, j).
  - Intermediate values smoothly blend the two images.

- Construct Gaussian Pyramids:
  - Build the Gaussian pyramids for the two images (A and B) and the mask (M).
  - A Gaussian pyramid is constructed by:
  - Starting with the original image.
  - Iteratively applying a Gaussian blur and downsampling by a factor of 2.
  - Let GA, GB, and GM represent the Gaussian pyramids of A, B, and M, respectively.

Image Blending using Laplacian Pyramid

- Construct Laplacian Pyramids:
    - Build the Laplacian pyramids for the two images (A and B).
    - A Laplacian pyramid is constructed by:
        - Taking the difference between two consecutive levels of a Gaussian pyramid (Will need upsampling).
        - At the final level, keep the coarsest Gaussian level directly.
    - Let LA and LB represent the Laplacian pyramids of A and B.

- Blend the Laplacian Pyramids:
    - Combine the Laplacian pyramids of the two images using the Gaussian pyramid of the mask:
        - For each level l:
            - $L^l_{\text{blend}} = GM^l \,.\, \text{LA}^l + (1 - GM^l) \,.\, \text{LB}^l$
        - This ensures that the regions where the mask is 1 will come from A, and where the mask is 0 will come from B.

- Reconstruct the Blended Image:
    - Reconstruct the final blended image by collapsing the blended Laplacian pyramid.
    - This involves:
        - Starting from the coarsest level.
        - Upsampling it and adding it to the next finer level iteratively until you reach the original resolution.

Image Blending using Laplacian Pyramid

Using the original images directly : Bad Blending

Hough Transform

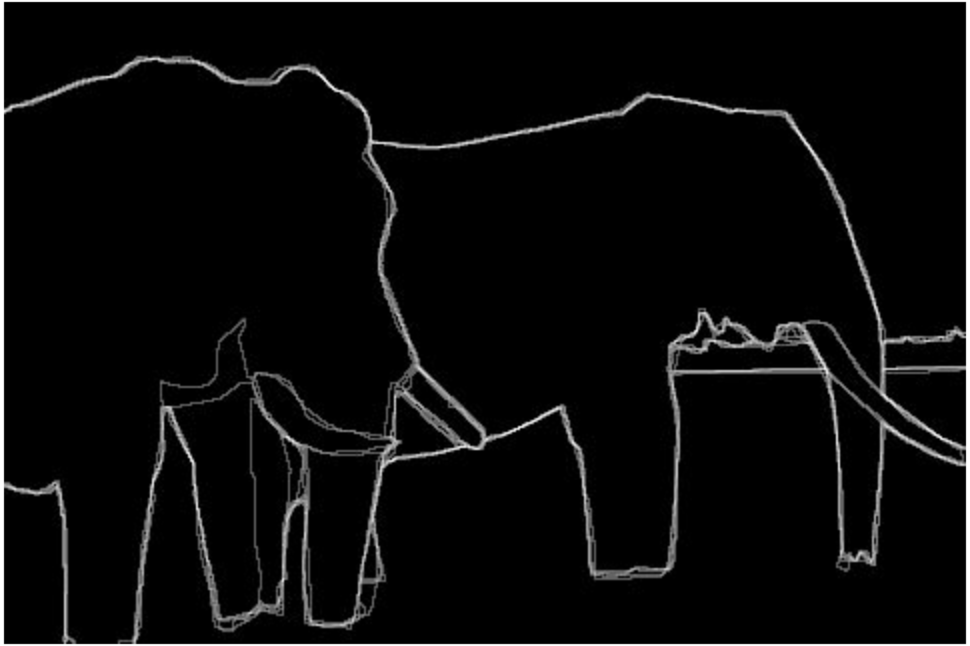
Finding boundaries

- We have already methods to detect edges.
- Edge pixels may not always correspond to the object boundaries.
- We need to find object boundaries in the image.

Finding boundaries

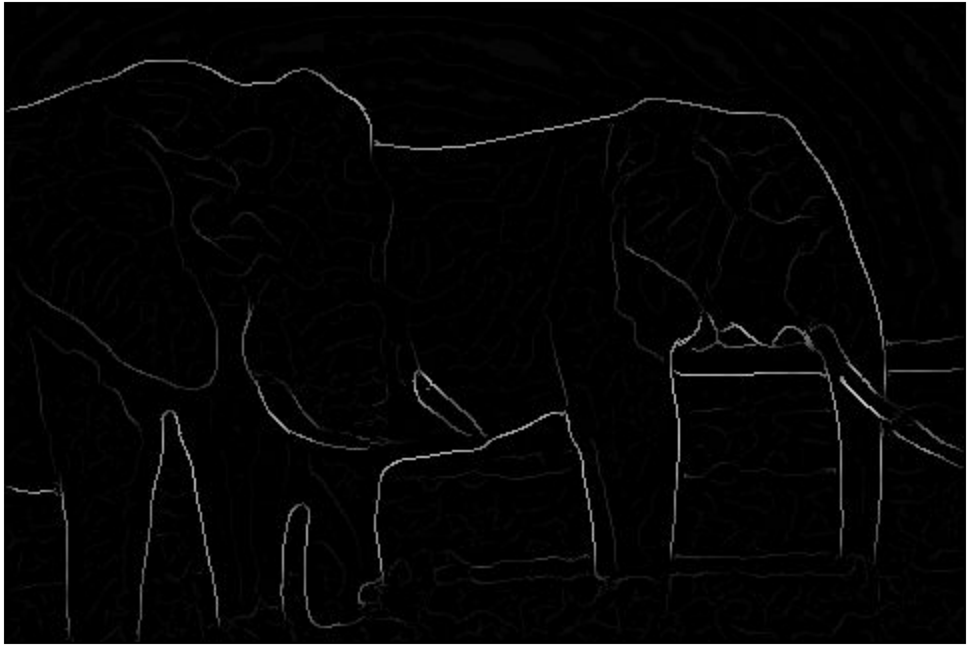
Where are the object boundaries?

Finding boundaries



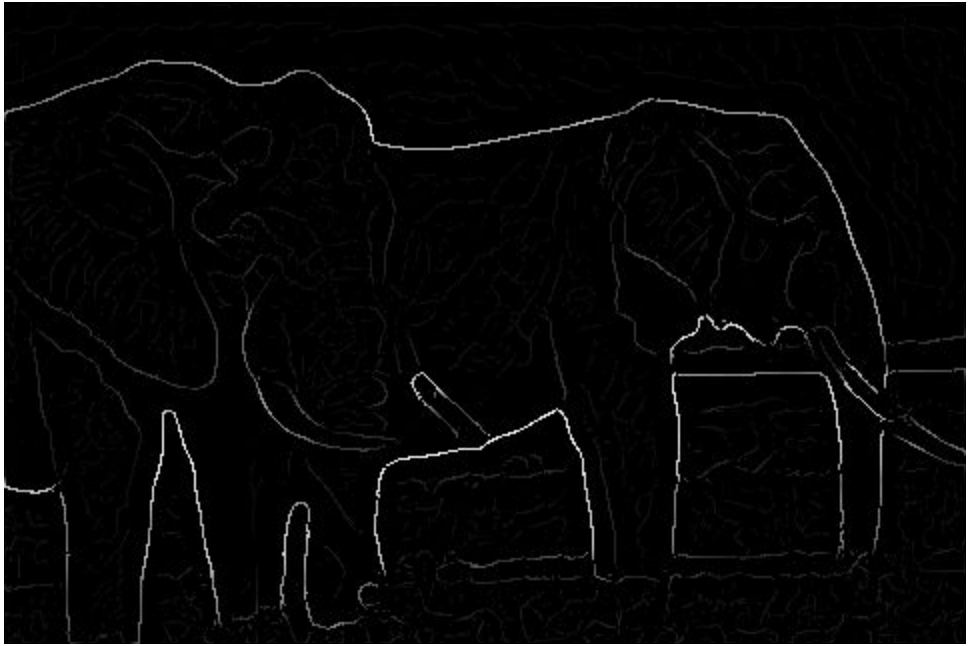
Human annotated boundaries

Finding boundaries



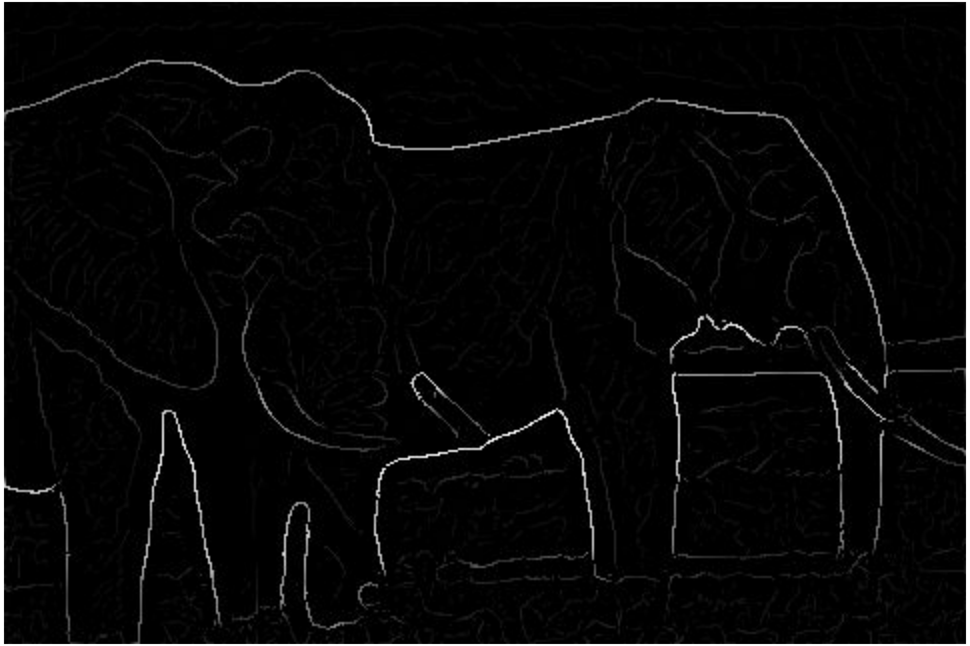
Edge Detection

Finding boundaries



Multi-scale edge detection

Finding boundaries

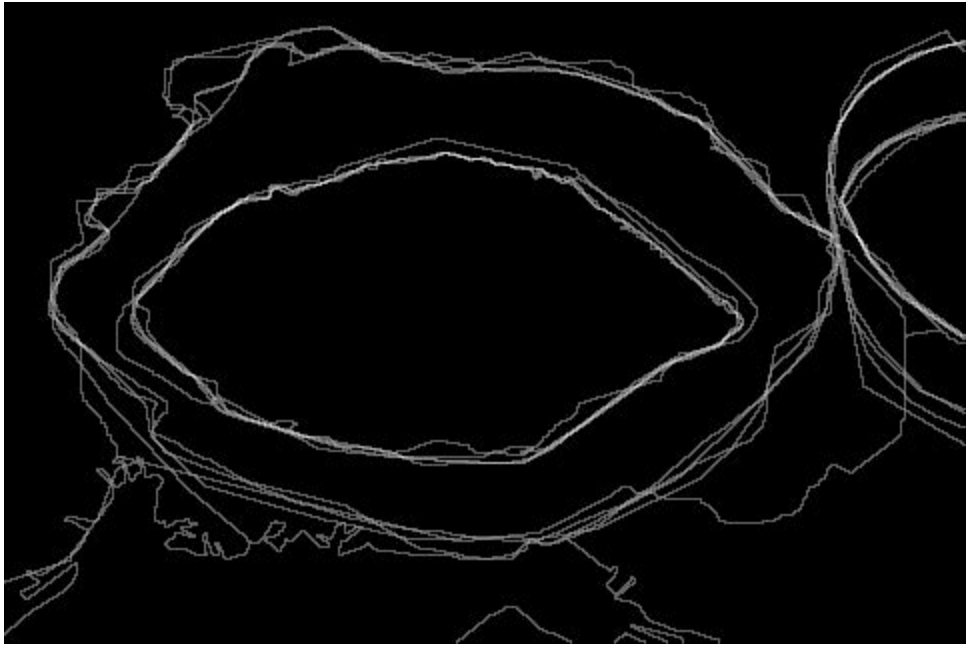


Edge strength does not necessarily correspond to our perception of boundaries

Finding boundaries

Where are the object boundaries?

Finding boundaries



Human annotated boundaries

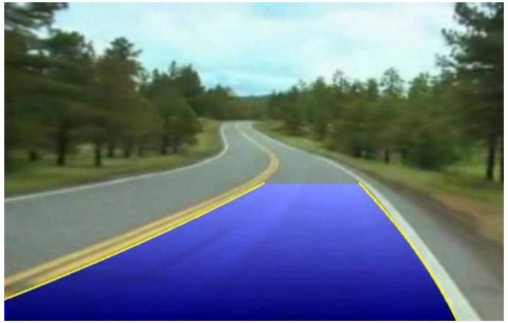
Finding boundaries

Edge Detection

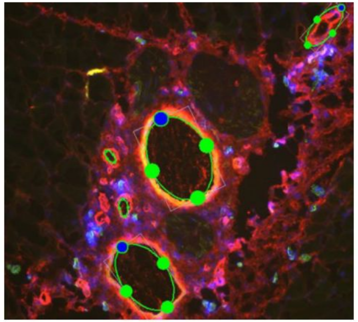
Finding boundaries

Defining boundaries are hard for us too

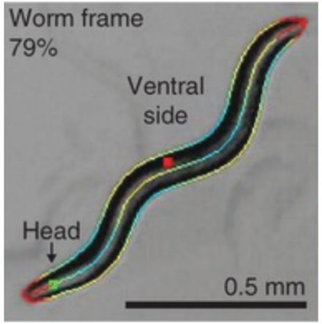
Applications



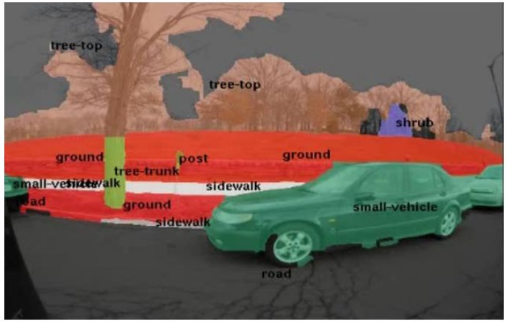
Autonomous Vehicles
(lane line detection)



tissue engineering
(blood vessel counting)



behavioral genetics
(earthworm contours)



Autonomous Vehicles
(semantic scene segmentation)

Computational Photography
(image inpainting)

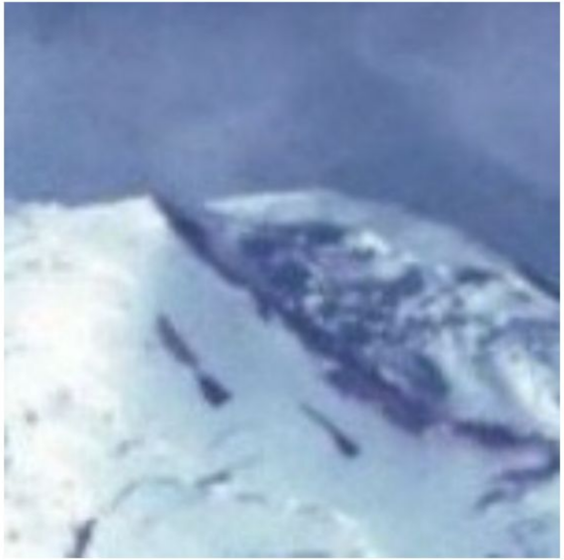
Finding boundaries

- We already have edges in the images.
- Try to fit Line to Edges

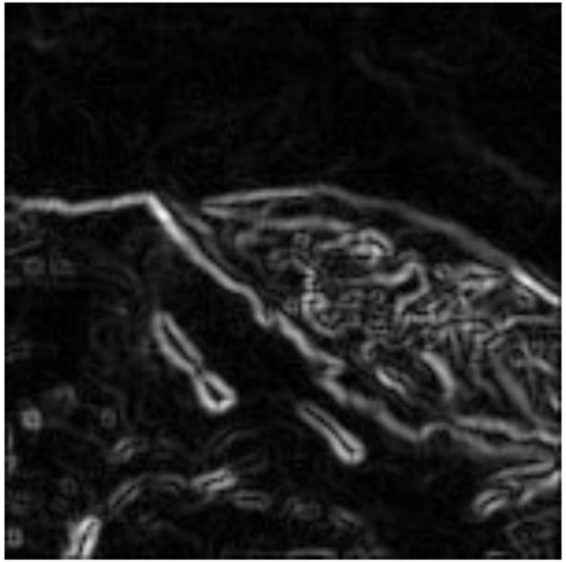
Finding boundaries

Where is the boundary of the mountain top?

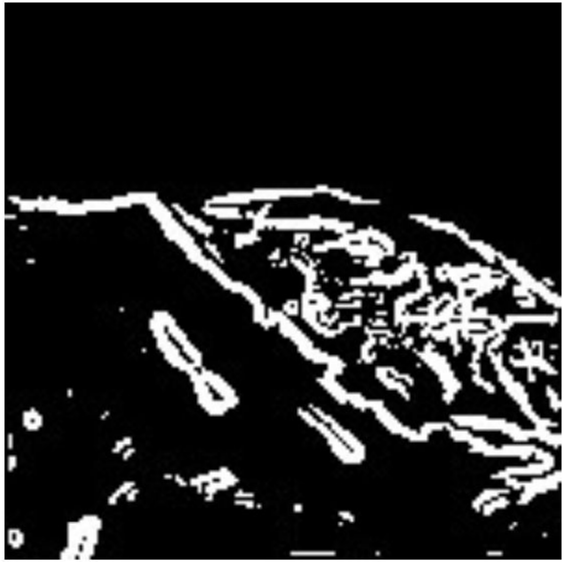
Lines are hard to find



Original image



Edge detection



Thresholding

Noisy edge image
Incomplete boundaries

Line fitting

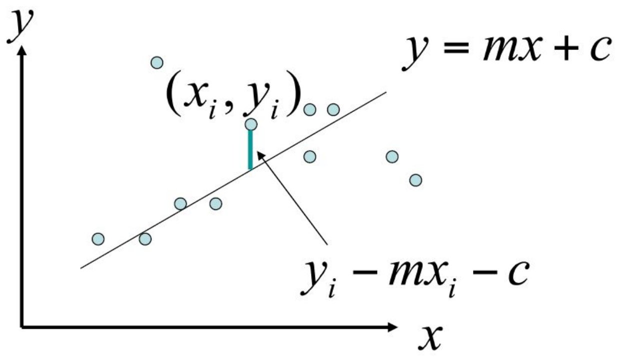
Line fitting

Given: Many (x_i, y_i) pairs

Find: Parameters (m, c)

Minimize: Average square distance:

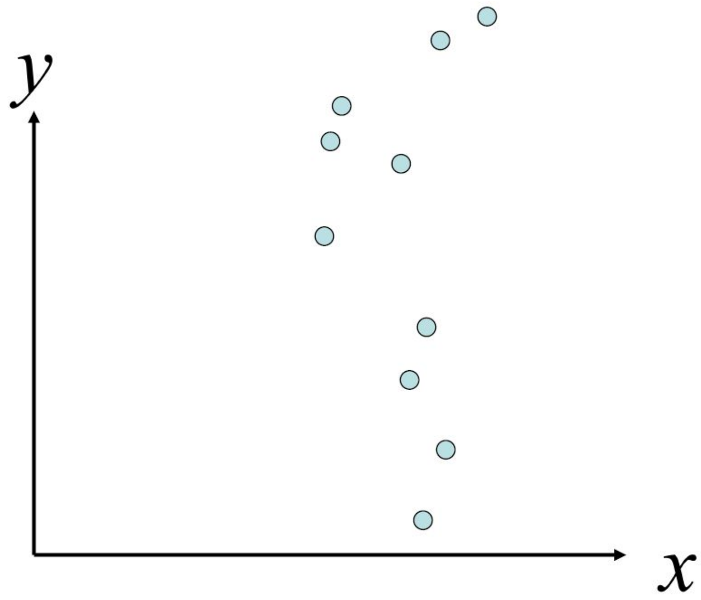
$$E = \sum_i \frac{(y_i - mx_i - c)^2}{N}$$



Problems with parameterizations

Where is the line that minimizes E?

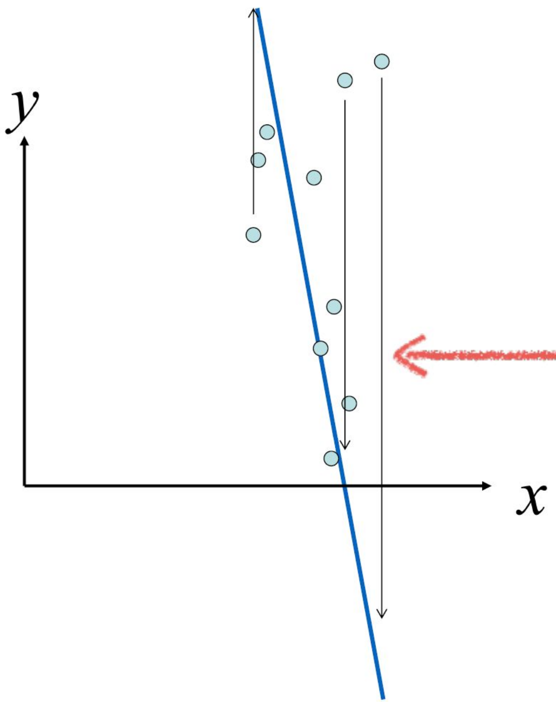
$$E = \sum_{i=1}^{n} (y_i - m x_i - b)^2$$



Problems with parameterizations

Where is the line that minimizes E?

$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

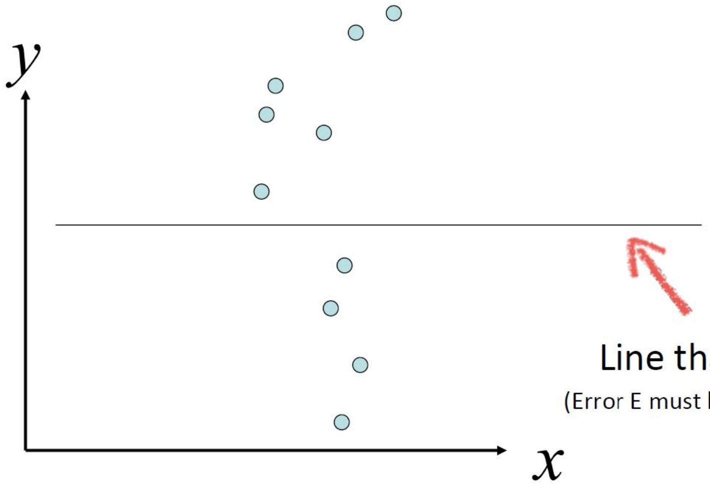


Huge E!

Problems with parameterizations

Where is the line that minimizes E?

$$\boxed{E=\sum_{i=1}^{n}(y_i-mx_i-b)}$$

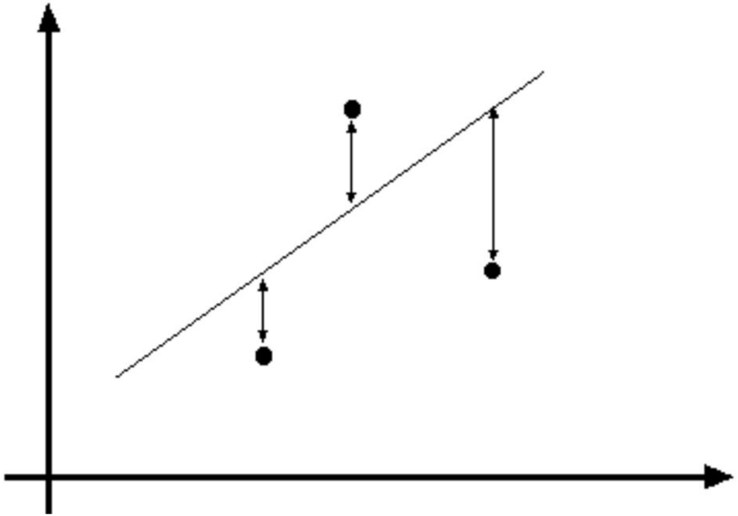


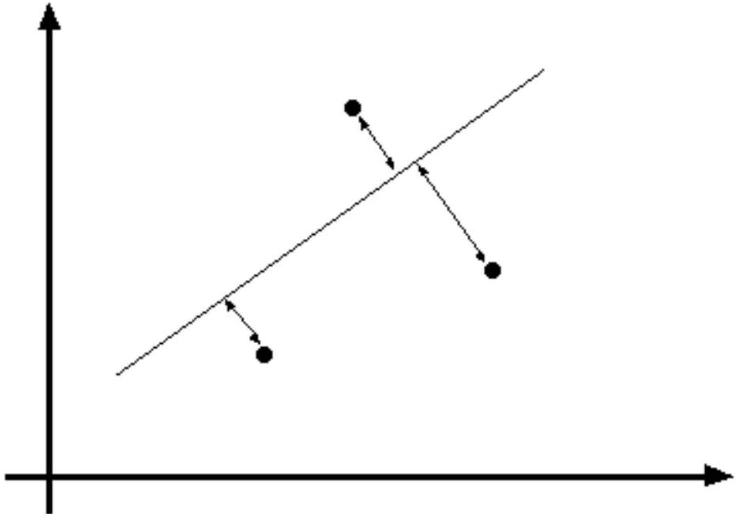
Line that minimizes E!!

(Error E must be formulated carefully!)

Line Fitting

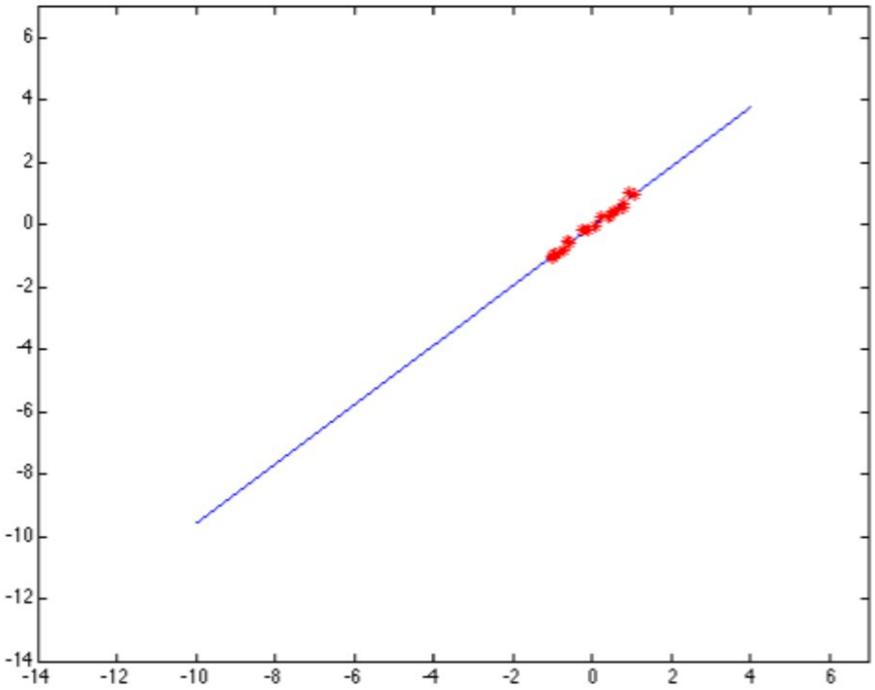
Line fitting is easily setup as a maximum likelihood problem
... but choice of model is important



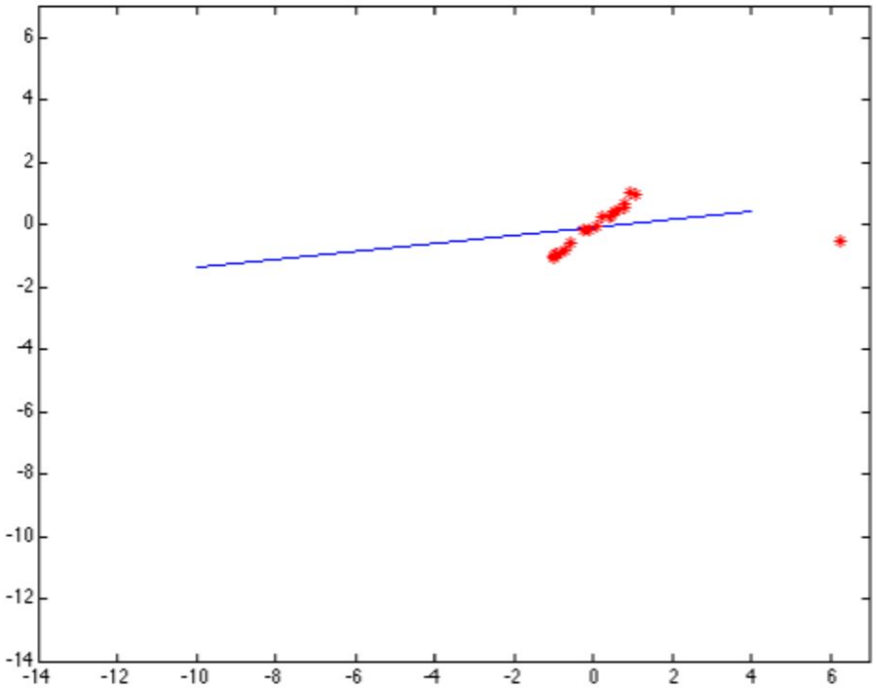


$$\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$$

Problems with noise



Least-squares error fit



Squared error heavily penalizes outliers