Optimization Methods in Computer Vision and Deep Learning

15 March 2023

Romen Samuel Wabina, M.Sc.
MSc Data Science and Artificial Intelligence

Student, PhD in Data Science for Healthcare and Clinical Informatics
Department of Clinical Epidemiology and Biostatistics, Faculty of Medicine Ramathibodi Hospital
Mahidol University, Thailand

About Me

Educational Background

BSc Applied Mathematics – Operations Research
Honorific Awardee | Consistent Dean's List Awardee
University of the Philippines
Year 2014 – 2018

GC Business Administration
Murdoch University
Perth, Australia | Singapore
Year 2019 - 2020

MSc Data Science and Artificial Intelligence
Asian Institute of Technology
Bangkok, Thailand
Year 2020 – 2022

PhD Data Science and AI in Healthcare and Clinical Informatics
Mahidol University
Faculty of Medicine – Ramathibodi Hospital
Year 2022 - Present

Professional Background

Data Analyst
August 2018 – August 2019
NerdHub | Nationlink Network

Lead Financial Analytics Officer
August 2019 – March 2020
Nationlink Network

Research Assistant
Department of Clinical Epidemiology and Biostatistics
Mahidol University

Recent Publications and Presentations

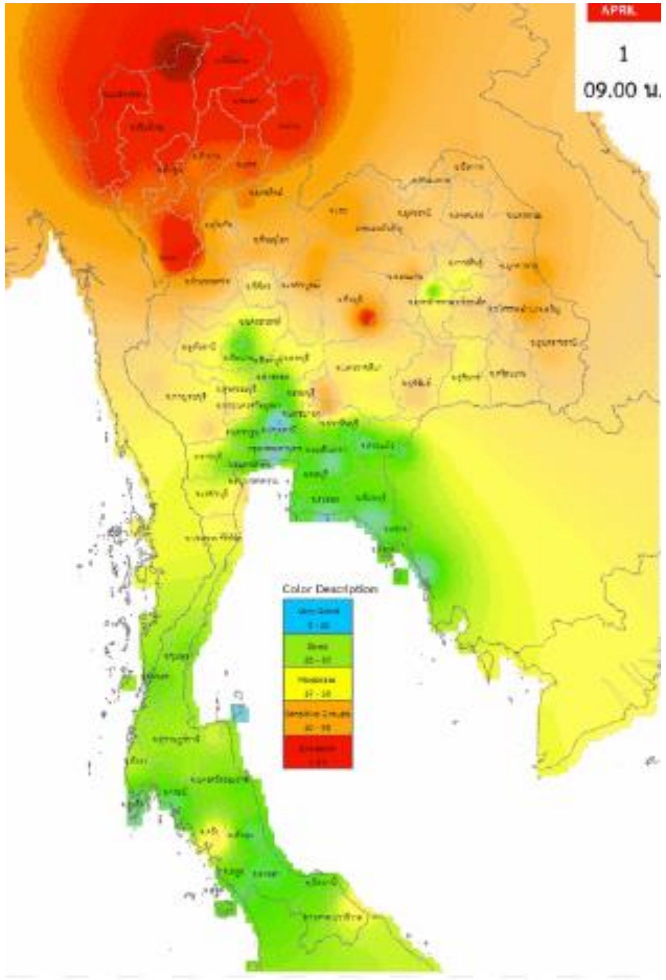
Wabina, R. S., & Silpasuwanchai, C. (2022). Neural stochastic differential equations network as uncertainty quantification method for EEG source localization. Biomedical Physics & Engineering Express.

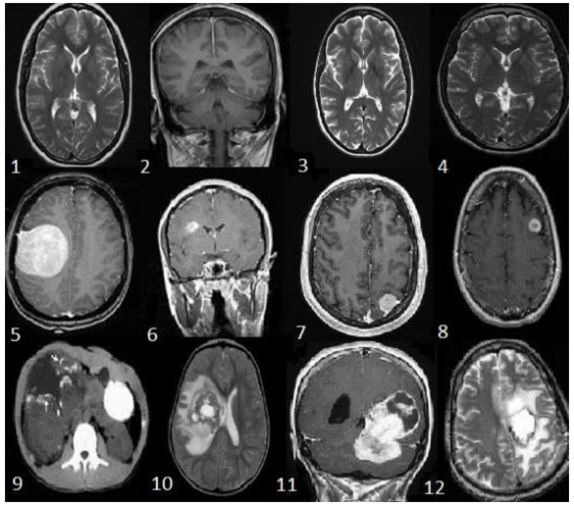
Wabina, R.S., & Anripa, N. (2022). Evaluation of Tobacco Control Policies in the Philippines and Indonesia in Strengthening Universal Health Care.

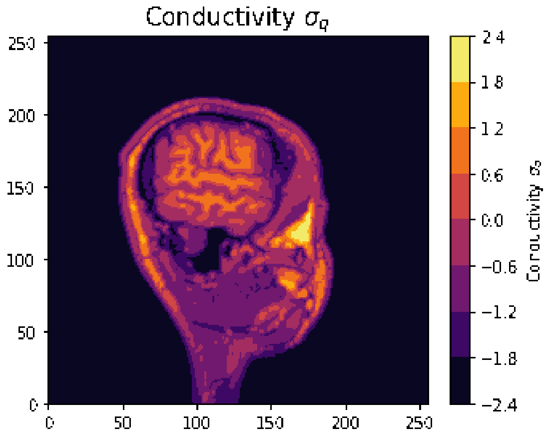
Research Interests

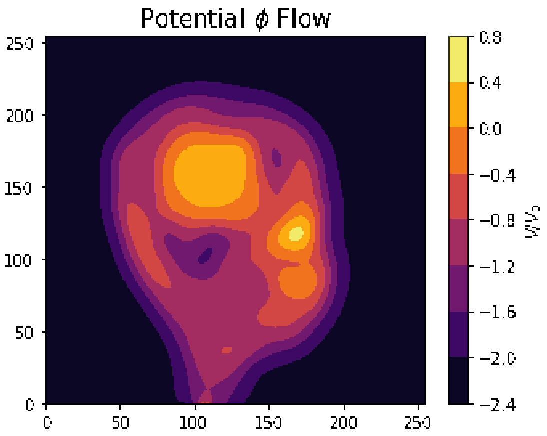
Specialty – Uncertainty Quantification, Bayesian Optimization, Dynamical Systems, Deep Learning, Computational Statistics
Areas of interest: Nucleotide Sequencing using Natural Language Processing, Deep Bayesian Optimization on Electrophysiological Models using Biomedical Volumetric Data

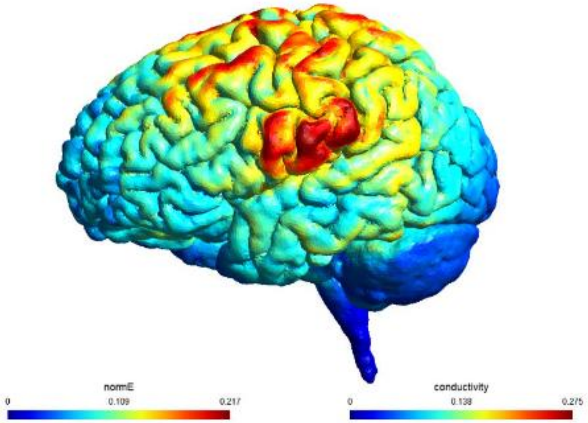
About Me

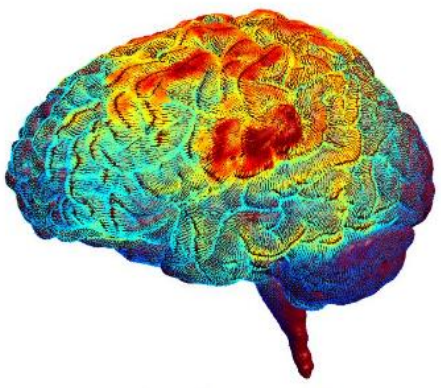











Electrical Propagation in VCM

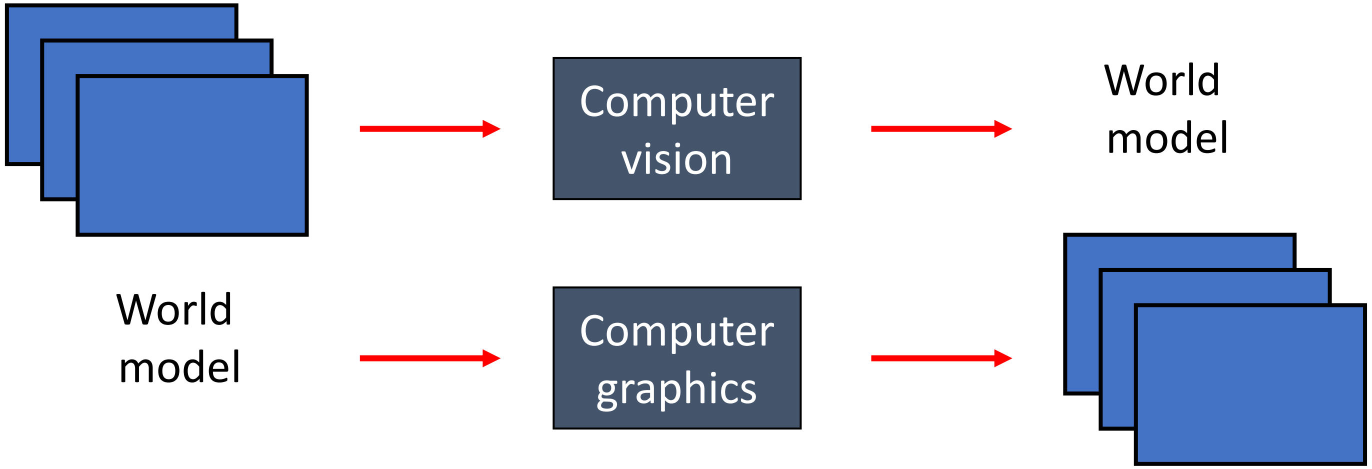
Model Optimization vs Data Optimization

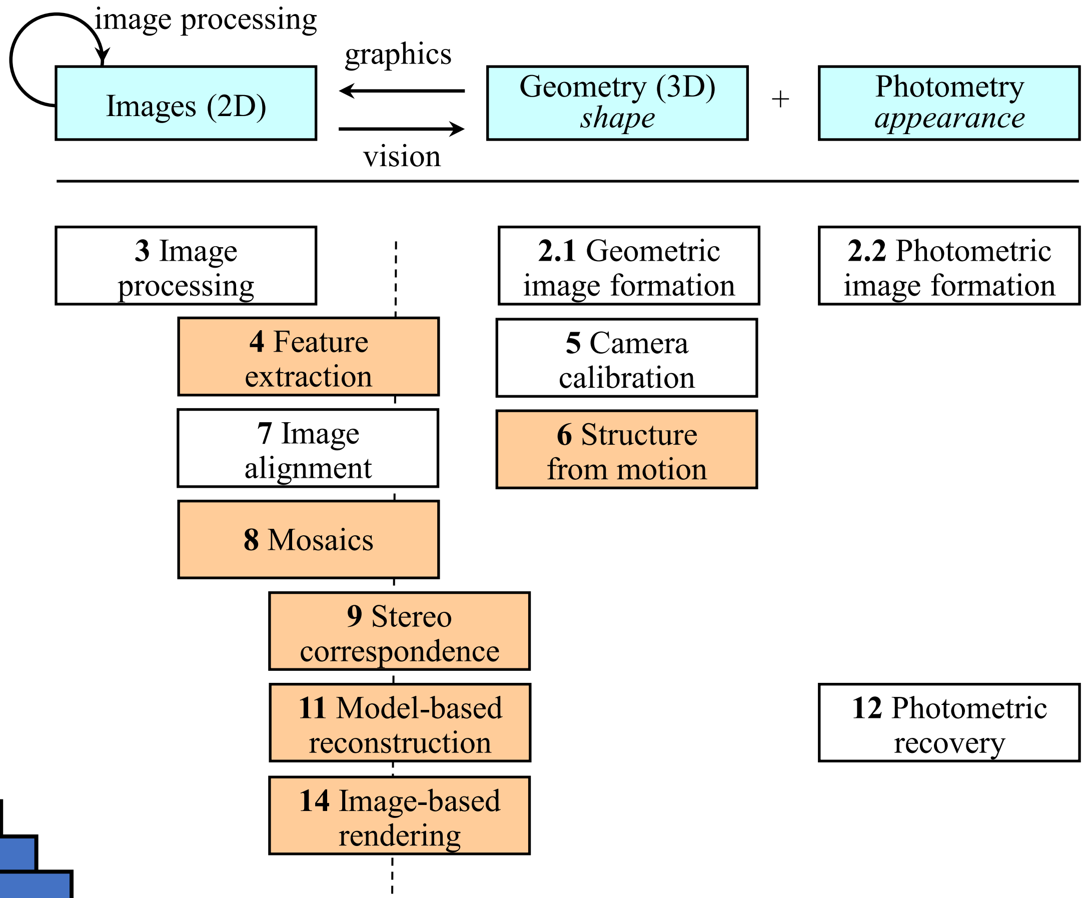
Computer Vision

Deals with the development of the **theoretical** and **algorithmic** basis by which useful information about the 3D world can be automatically extracted and analyzed from a single or multiple o 2D images of the world.

Its primary goals and applications include:

- Make computers **understand** images and video
- Image understanding (artificial intelligence, behaviour)
- Sensor modality for robotics and automated processes
- Computer emulation of human vision
- Inverse of computer graphics



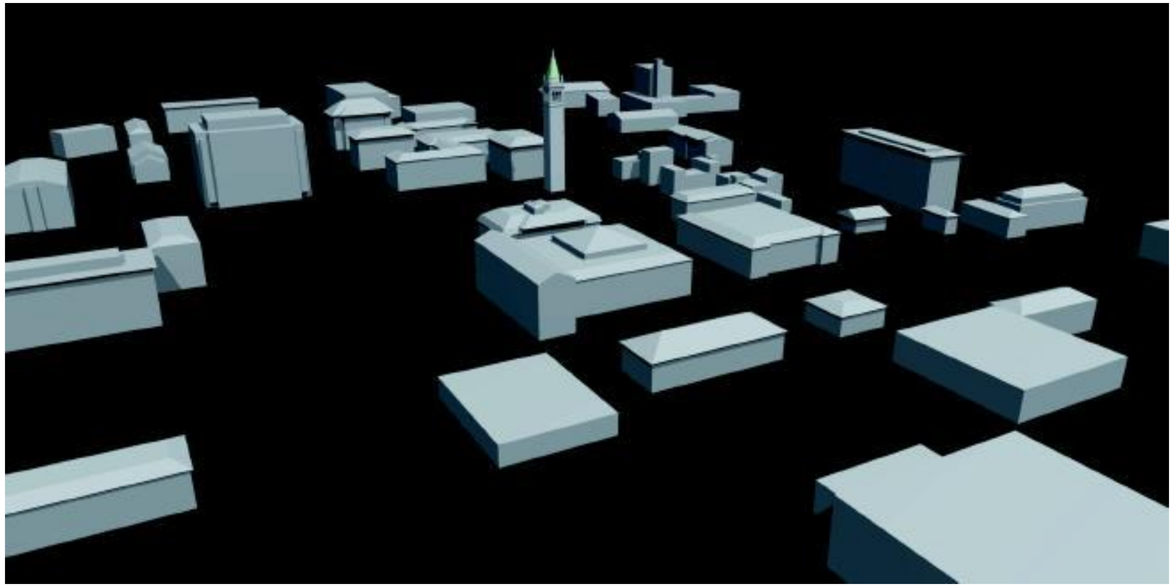


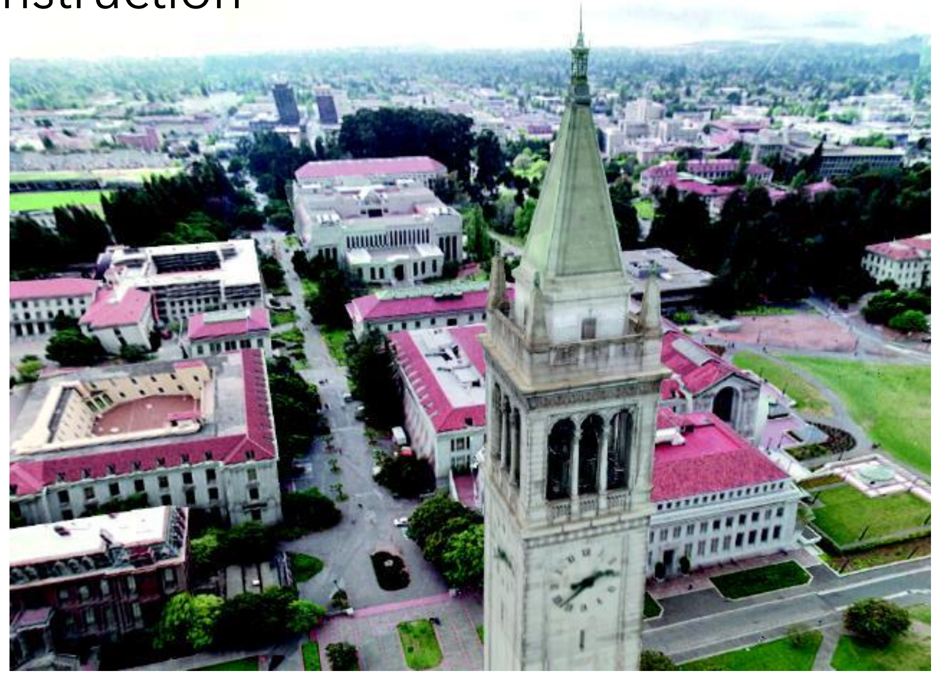
Every industry wants Computer Vision and Deep Learning

Cloud Service Provider	Medicine	Media & Entertainment	Security & Defense	Autonomous Machines
➢ Image/Video classification	➢ Cancer cell detection	➢ Video captioning	➢ Face recognition	➢ Pedestrian detection
➢ Speech recognition	➢ Diabetic grading	➢ Content based search	➢ Video surveillance	➢ Lane tracking
➢ Natural language processing	➢ Drug discovery	➢ Real time translation	➢ Cyber security	➢ Recognize traffic sign

Source: NVIDIA (www.slideshare.net/openomics/the-revolution-of-deep-learning)

3D Shape Reconstruction





Debevec, Taylor, and Malik, SIGGRAPH 1996

Edge Detection

Elder, J. H. and R. M. Goldberg. "Image Editing in the Contour Domain,"
Proc. IEEE: Computer Vision and Pattern Recognition, pp. 374-381, June 1998.

High Dynamic Range Photography

[Debevec et al.'97; Mitsunaga & Nayar'99]

Combine several different exposures together

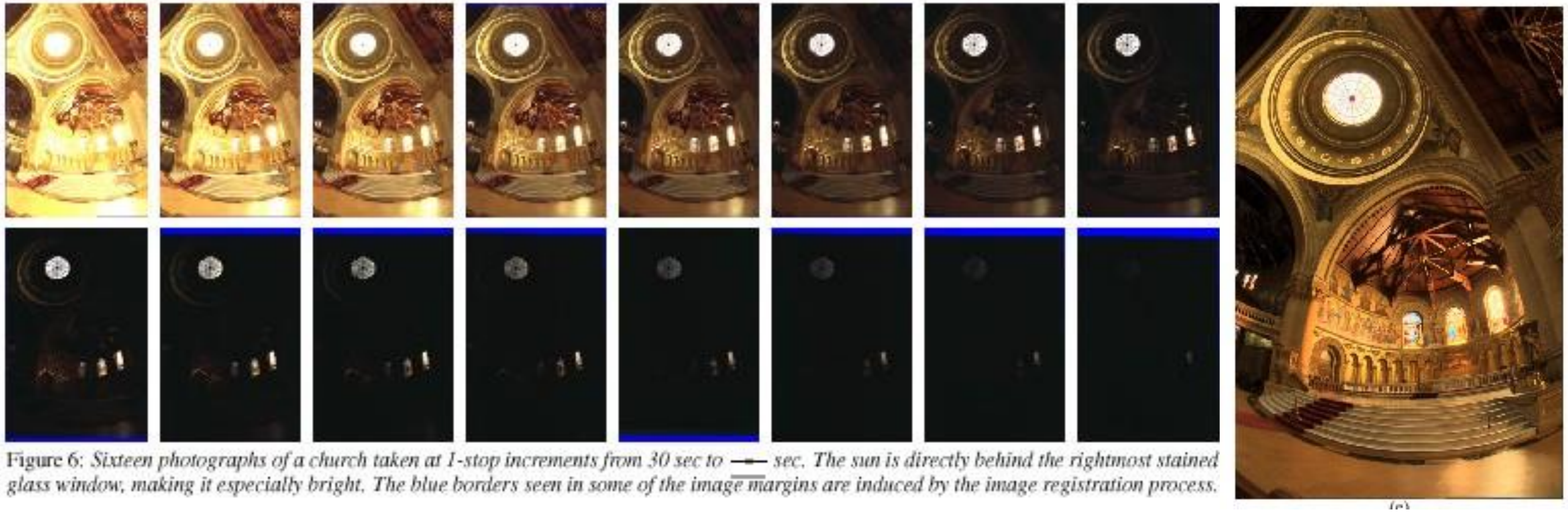
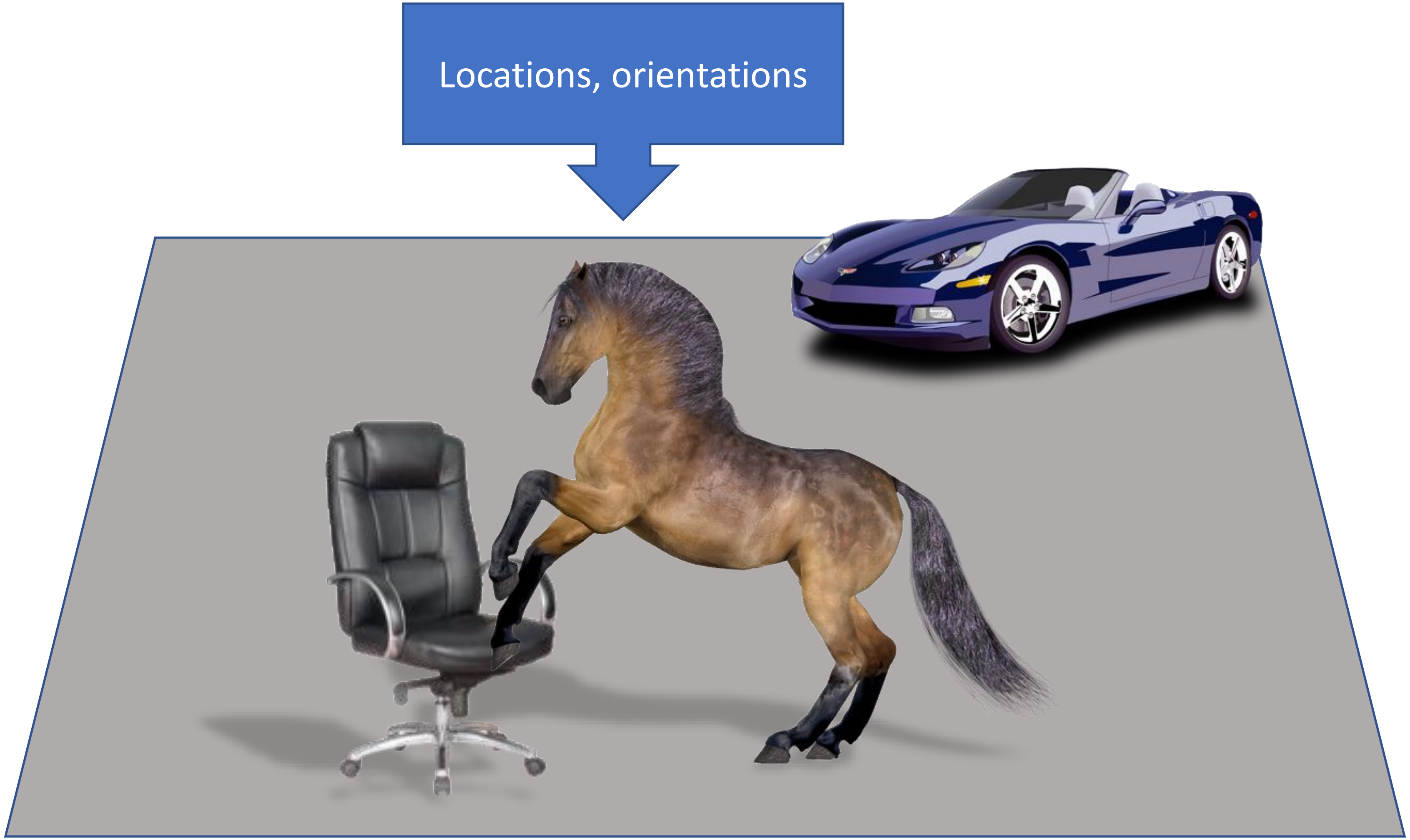
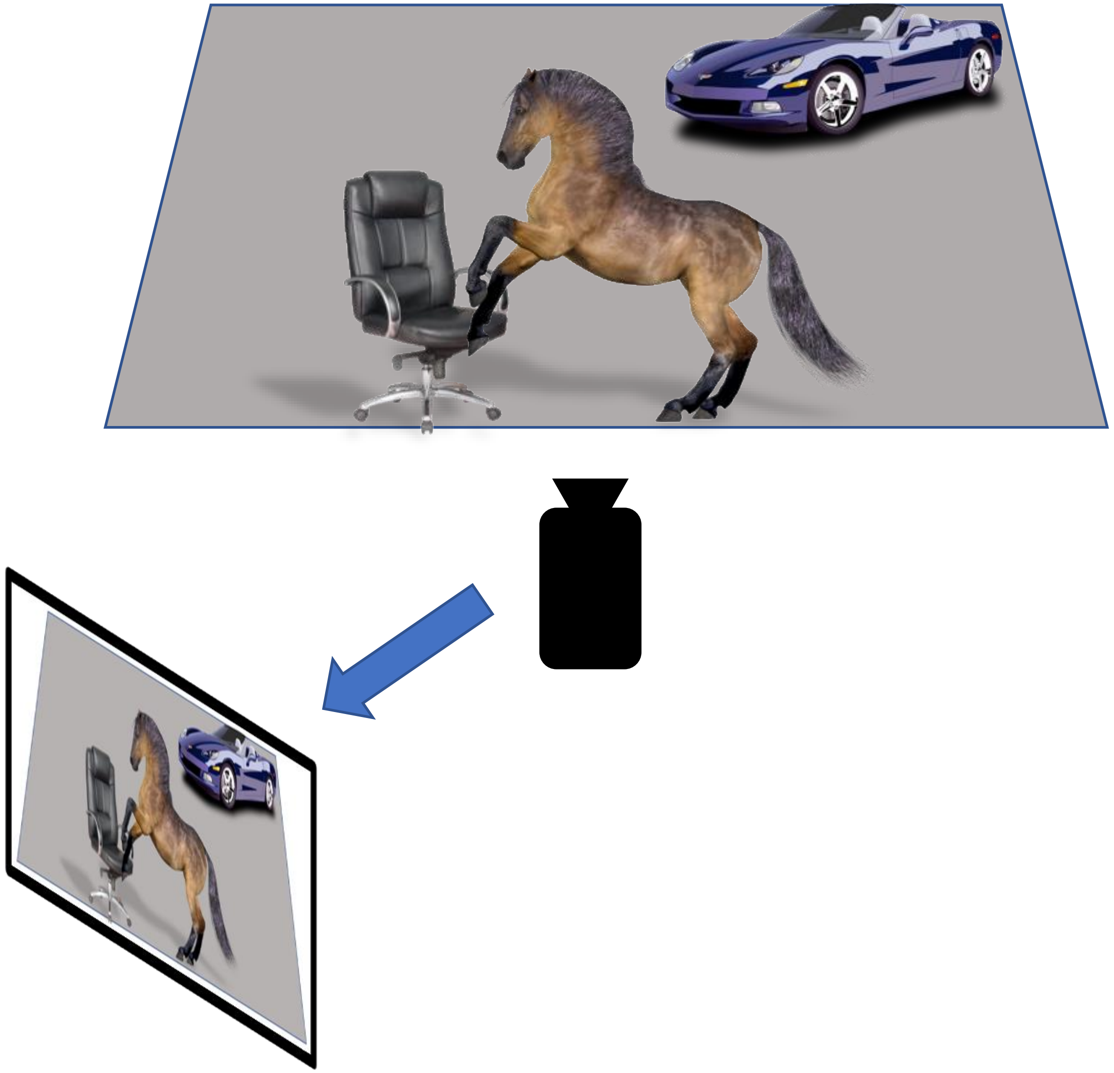


Figure 6: *Sixteen photographs of a church taken at 1-stop increments from 30 sec to [illegible] sec. The sun is directly behind the rightmost stained glass window, making it especially bright. The blue borders seen in some of the image margins are induced by the image registration process.*

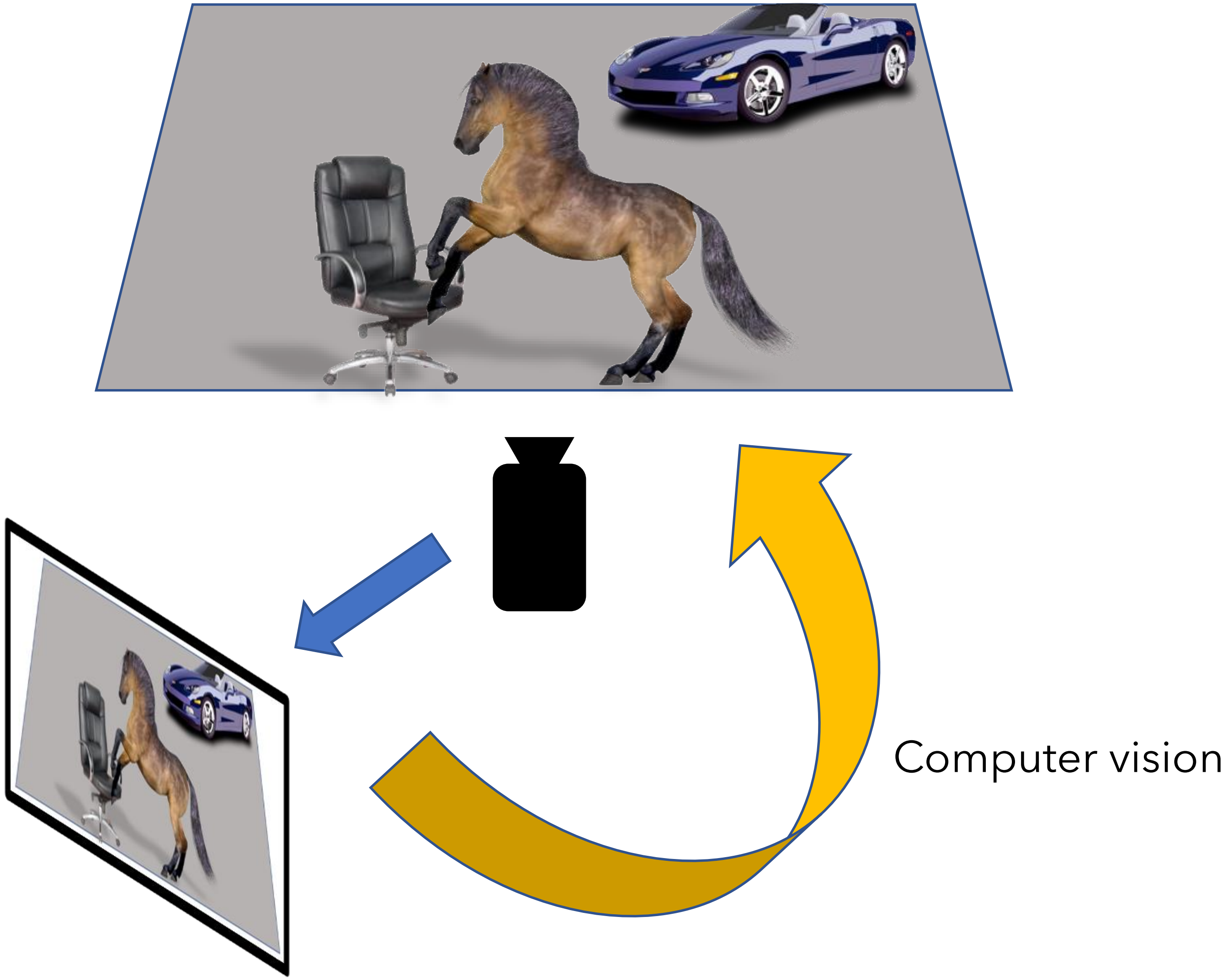
The *physical* world



The *physical* world into *images*

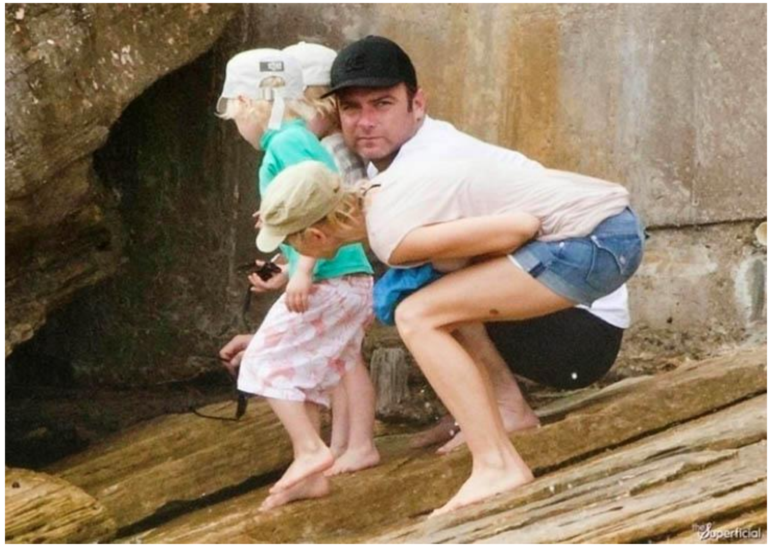


The *physical* world into *images*



Vision is **always** easy for humans

Vision is hard: Images are **ambiguous**



Viewpoint variation

Illumination

Scale

Shape variation

Occlusion

Background clutter

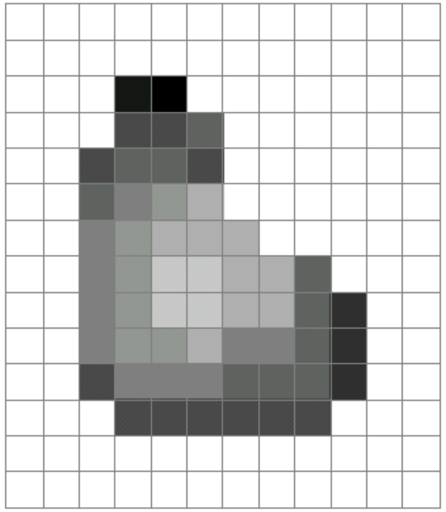
Local Ambiguity

Tenessee Warbler Orange Crowned Warbler

Two things might look very similar but be completely different.

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

A grid (matrix) of intensity values: 1 color or 3 colors

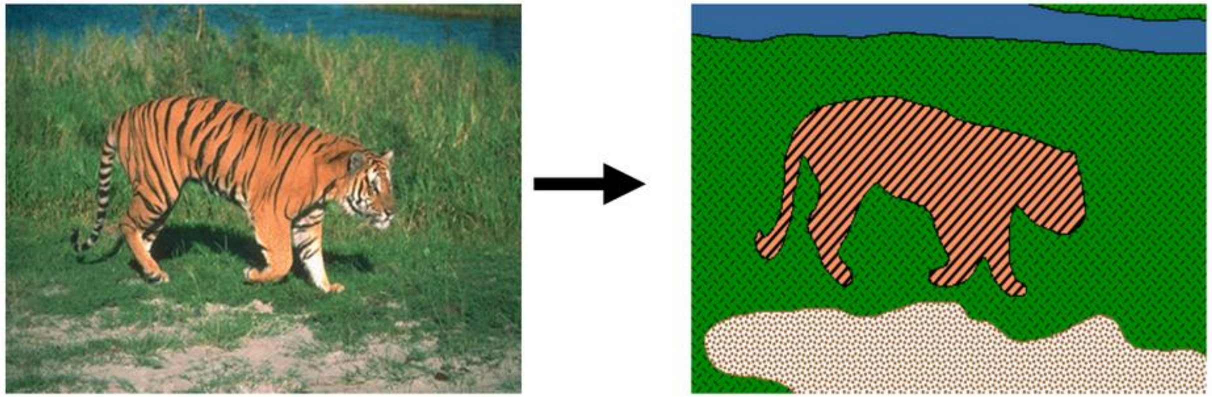


=

255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

Grouping ("Reorganization")

Convert from "pixels" to "objects": which groups of pixels correspond to objects?

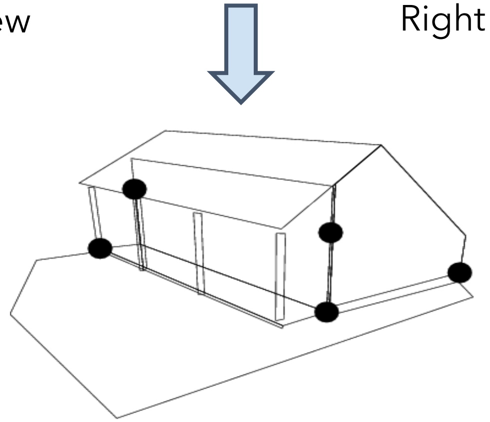


Reconstruction

Go from 2D arrays to 3D: what does every pixel correspond to in 3D

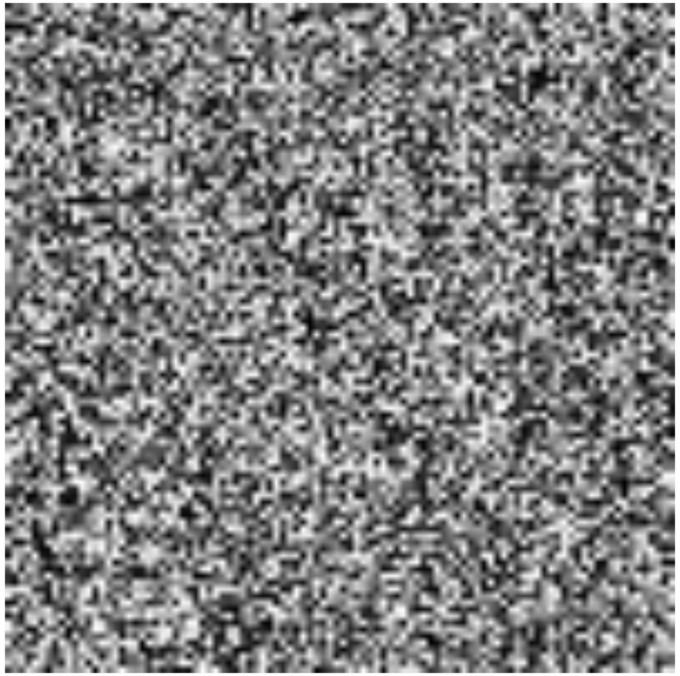
Left View

Right View

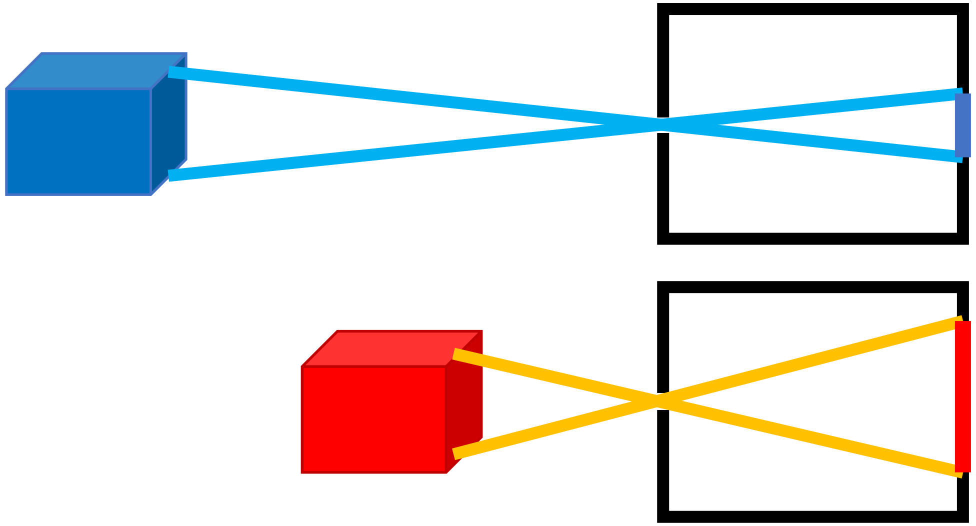


Not all 2D arrays are images

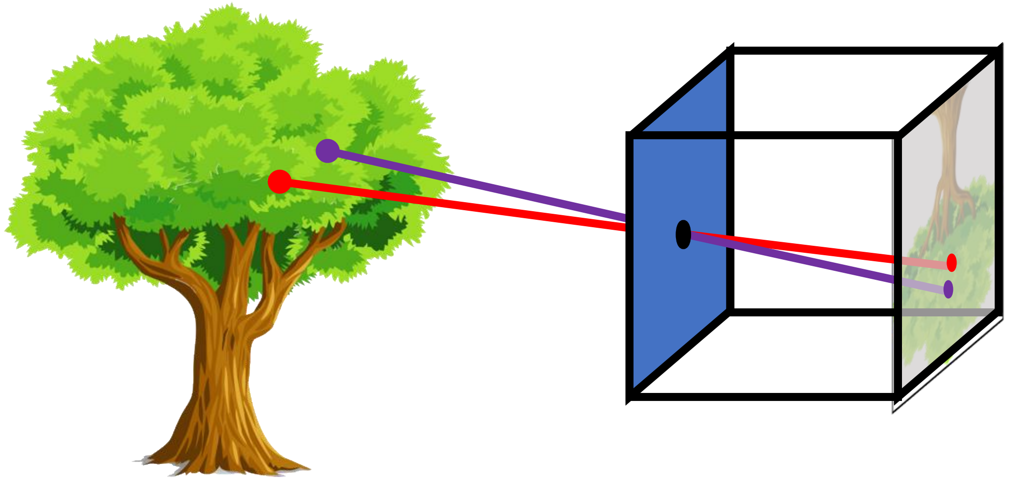
0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0



Consequences: Farther away objects appear smaller



Consequences: Nearby pixels are similar



Nearby pixels that are *not* similar tend to lie on *different* objects
Idea: To find where one object ends and another begins, look for *abrupt changes in color*

Places of color change might correspond to object boundaries
Object boundaries are a clue to *object shape*
Idea: *Use rough boundaries to recognize object(s)*

Key Takeaways from image consequences

- Natural images are ***not* arbitrary** 2D arrays
- They have properties resulting from physics / math of image formation
- Solving computer vision requires using these properties

Some methods:

Edge detection: identifying where pixels change color
- Cue to object boundary
- Cue to shape
- More resilient to lighting than pixel color

Zooming into or out of images
- Searching for both nearby and far-off objects

Matching patches from two different images
- First step in identifying 3D location

Image denoising

Let us assume noise at a pixel is

- independent of other pixels
- distributed according to a Gaussian distribution i.e., low noise values are more likely than high noise values
- "grainy images"

Use Noise Reduction

- Nearby pixels are likely to belong to same object thus likely to have similar color
- Replace each pixel by average of neighbors

Image denoising

$$S[f](m,n) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} {\color{red} w(i,j)} f(m+i, n+j)$$

0	0	0	0	0	0	0	0	0	0
0	0	0	10	10	10	0	0	0	0
0	0	10	20	20	20	10	40	0	0
0	10	20	30	0	20	10	0	0	0
0	10	0	30	40	30	20	10	0	0
0	10	20	30	40	30	20	10	0	0
0	10	20	10	40	30	20	10	0	0
0	10	20	30	30	20	10	0	0	0
0	0	10	20	20	0	10	0	20	0
0	0	0	10	10	10	0	0	0	0

0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

(0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 10)/9 = 1.11

Image denoising

$$S[f](m,n) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} f(m+i, n+j)/9$$

10	5	3
4	5	1
1	1	7

Local image data

f

	7	

Modified image data

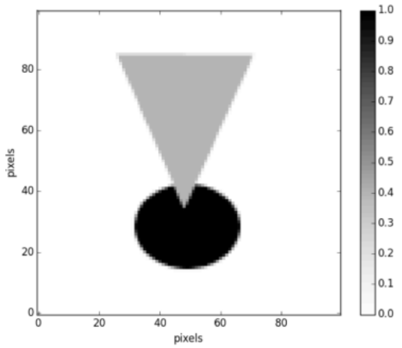
$S(f)$

Image denoising

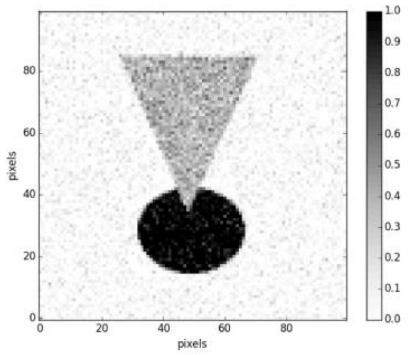
Gaussian Noise: A statistical noise with a probability density function (PDF) equal to the normal distribution, as indicated below, where z represents the gray level

$$p_G(z) = \frac{1}{\sigma\left(\sqrt{2\pi}\right)} e^{\frac{(z-\mu)^2}{2\sigma^2}}$$

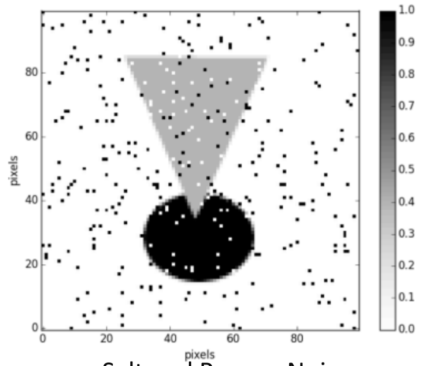
Sale-and-pepper noise: This noise can be caused by sharp and sudden disturbances in the image signal. It presents itself as sparsely occurring white and black pixels. (very grainy-looking image)



No Noise



Gaussian Noise



Salt and Pepper Noise

Image denoising

Poisson Noise: Similar with Gaussian Noise but with a probability density function (PDF) equal to the Poisson distribution, as indicated below, where λ represents the time interval and k is the number of appearances (i.e., multiple shots or jump shots)

$$p_P(z) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Speckle noise: A granular interference that inherently exists in and degrades the quality of the optical tomography images, radio wave images, and other biomedical images

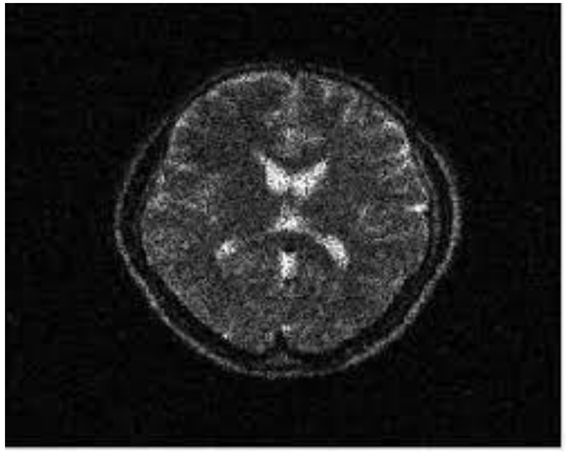


Image denoising

Gaussian Filter

Linear filtering technique by replacing each pixel with mean of neighboring pixels, sensitive to outliers so it edges are also filtered.

Median Filter

- A non-linear digital-filtering technique by replacing each pixel with the median of neighboring pixels
- Demonstrates better compared to Gaussian blur because <u>it removes noises but preserves the edges of the image</u>.

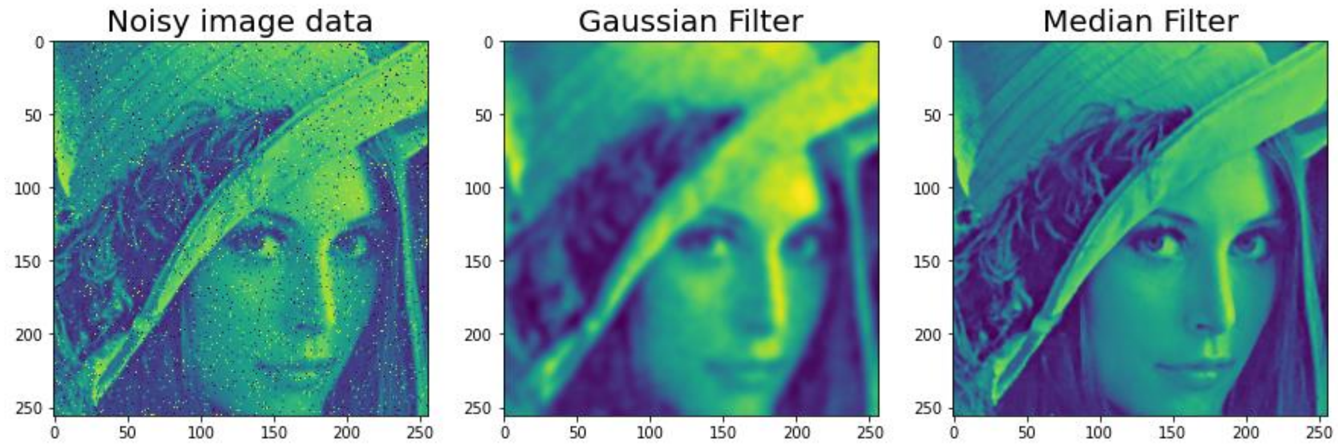
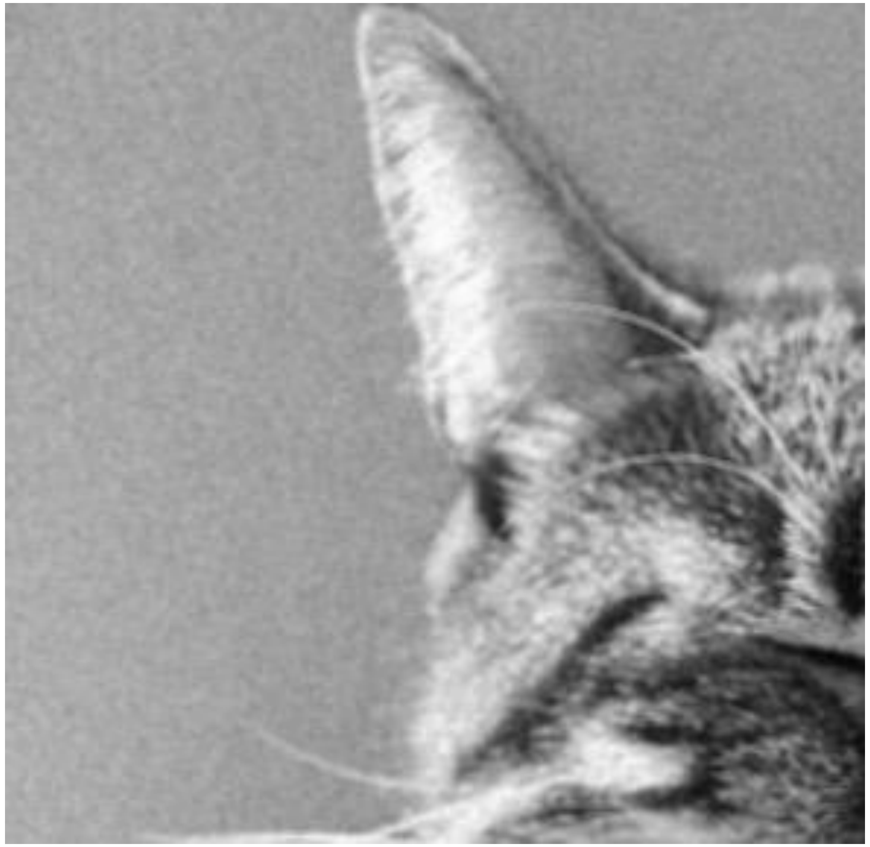


Image denoising using mean filter



Cross-correlation

$$S[f] = w \otimes f$$

$$S[f](m,n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i,j) f(m+i, n+j)$$

Convolution

$$S[f] = w * f$$

$$S[f](m,n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i,j) f(m-i, n-j)$$

Cross-correlation

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$$\sum_i w_i f_i = \vec{w} \cdot \vec{f}$$

$$\vec{w} = \begin{bmatrix} w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 & w_9 \end{bmatrix}$$

$$\vec{f} = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f_9 \end{bmatrix}$$

Dot-product

$$\vec{w} \cdot \vec{f} = \|\vec{w}\| \|\vec{f}\| \cos\theta$$

- cos θ indicates similarity
- can measure how much f "matches" w
  - Central component of "template matching"
  - But might need to divide by magnitude
  - Cosine distance
- cross-correlation ≈ template matching

Properties: *Linearity*

$$(w \otimes f)(m,n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i,j) f(m+i, n+j)$$

$$f'(m,n) = af(m,n)$$

$$(w \otimes f')(m,n) = a(w \otimes f)(m,n)$$

$$f' = af + bg$$

$$w \otimes f' = a(w \otimes f) + b(w \otimes g)$$

$$f' = af$$

$$(w \otimes f') = a(w \otimes f)$$

Properties: *Shift Invariance*

$$(w \otimes f)(m,n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i,j) f(m+i, n+j)$$

$$f'(m,n) = f(m-m_0, n-n_0)$$

f

f’

Properties: *Shift Invariance*

f $\quad$ f'

$$(w \otimes f)(m,n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i,j) f(m+i, n+j)$$

$$f'(m,n) = f(m - m_0, n - n_0)$$

$$(w \otimes f')(m,n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i,j) f'(m+i, n+j)$$

$$= \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i,j) f(m+i-m_0, n+j-n_0)$$

$$= (w \otimes f)(m - m_0, n - n_0)$$

Properties: *Shift Invariance*

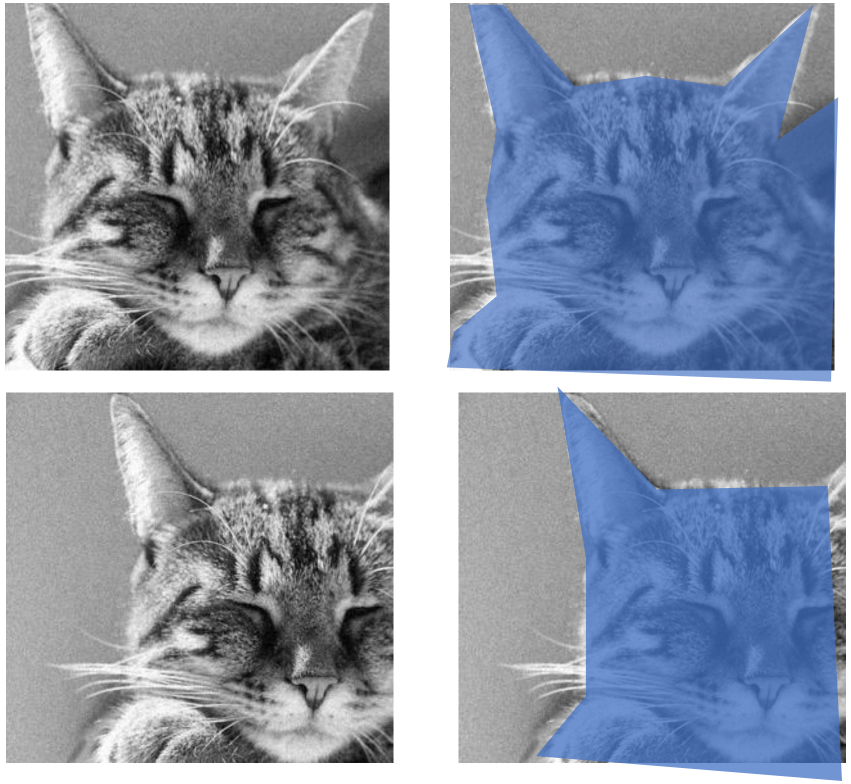
$$f'(m,n) = f(m - m_0, n - n_0)$$
$$(w \otimes f')(m,n) = (w \otimes f)(m - m_0, n - n_0)$$

Shift, then convolve = convolve, then shift

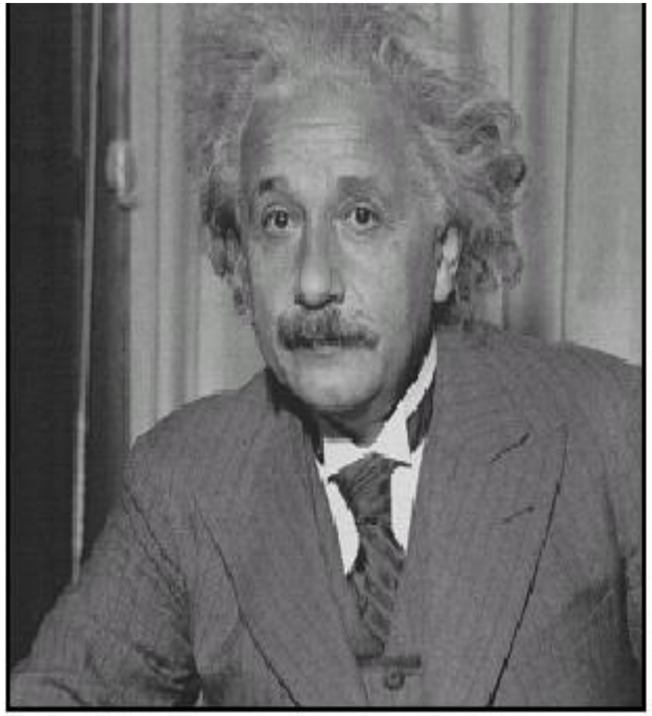
Output of convolution does not depend on where the pixel is

f

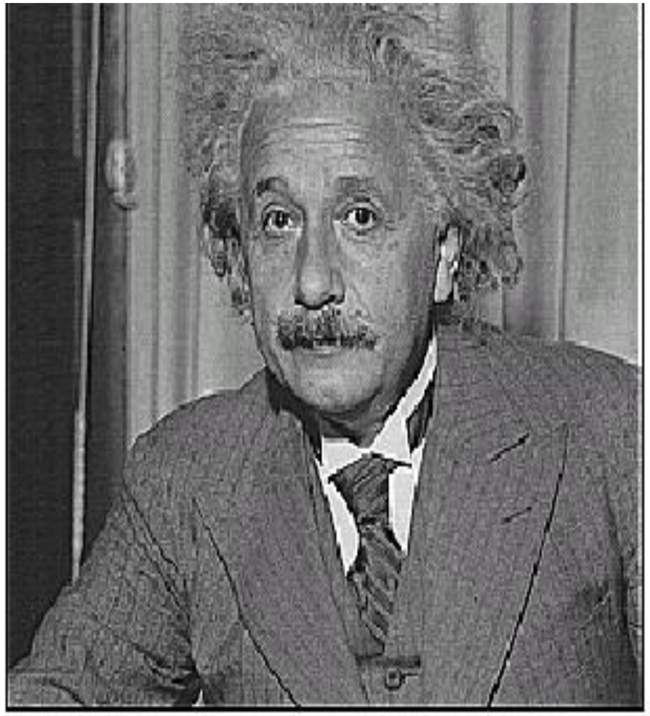
f’



Sharpening



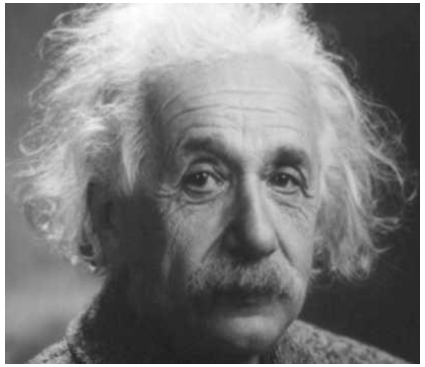
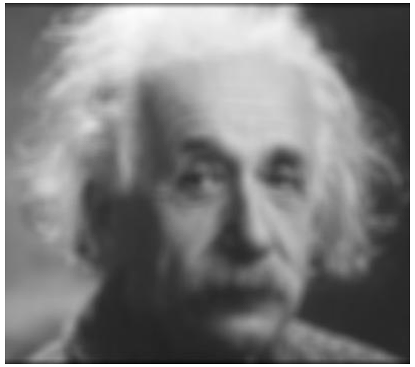
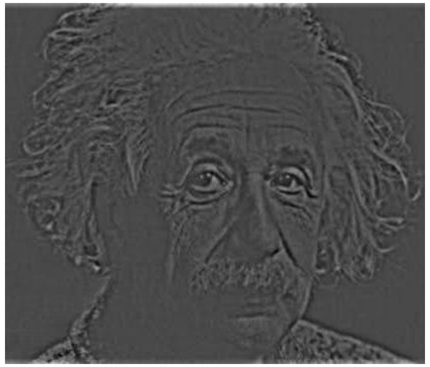
before

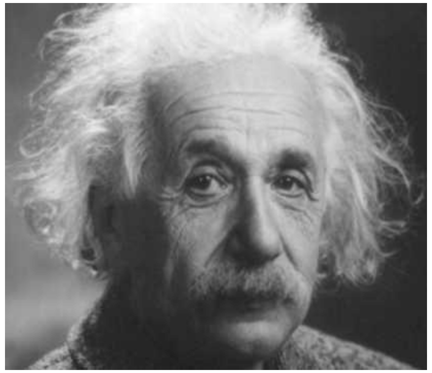
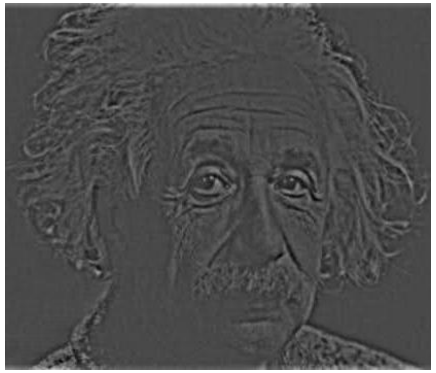
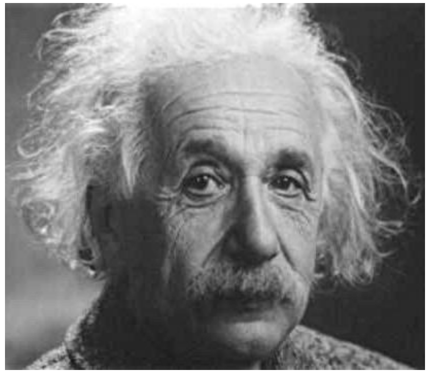


after

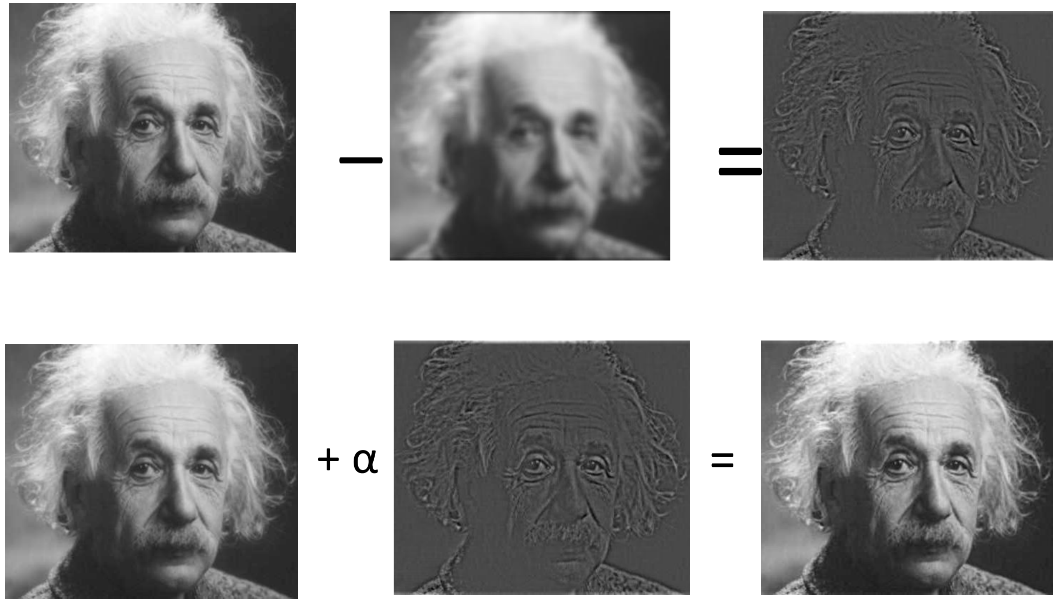
Sharpening

What does blurring take away?

 −  = 

 + α  = 

Sharpening



$$f_{sharp} = f + \alpha(f - f_{blur})$$

$$= (1+\alpha)f - \alpha f_{blur}$$

$$= (1+\alpha)(w * f) - \alpha(v * f)$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= ((1+\alpha)w - \alpha v) * f$$

Sharpening

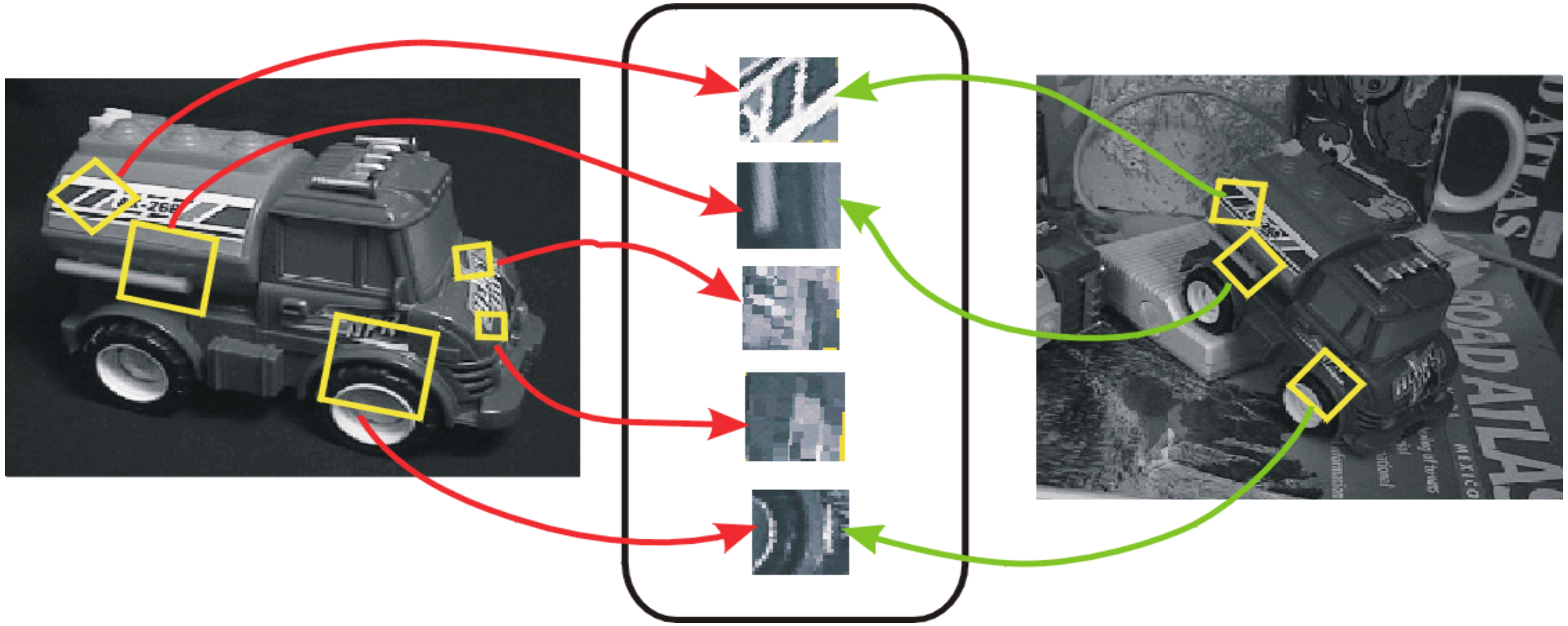
Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

Sharpening filter
(accentuates edges)

Invariant Local Features

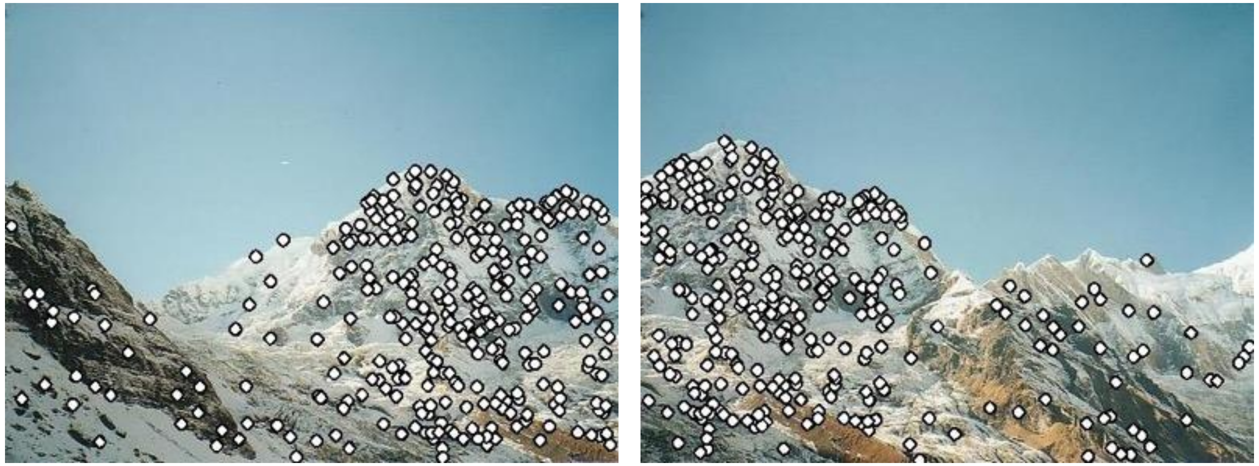
Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters.



The scale-invariant feature transform (SIFT) is a computer vision algorithm to detect, describe, and match local features in images (Lowe, 1999)


delicious vit-hydration to revive
the mind.
TDK
SANYO
GAP

Characteristics of good feature points



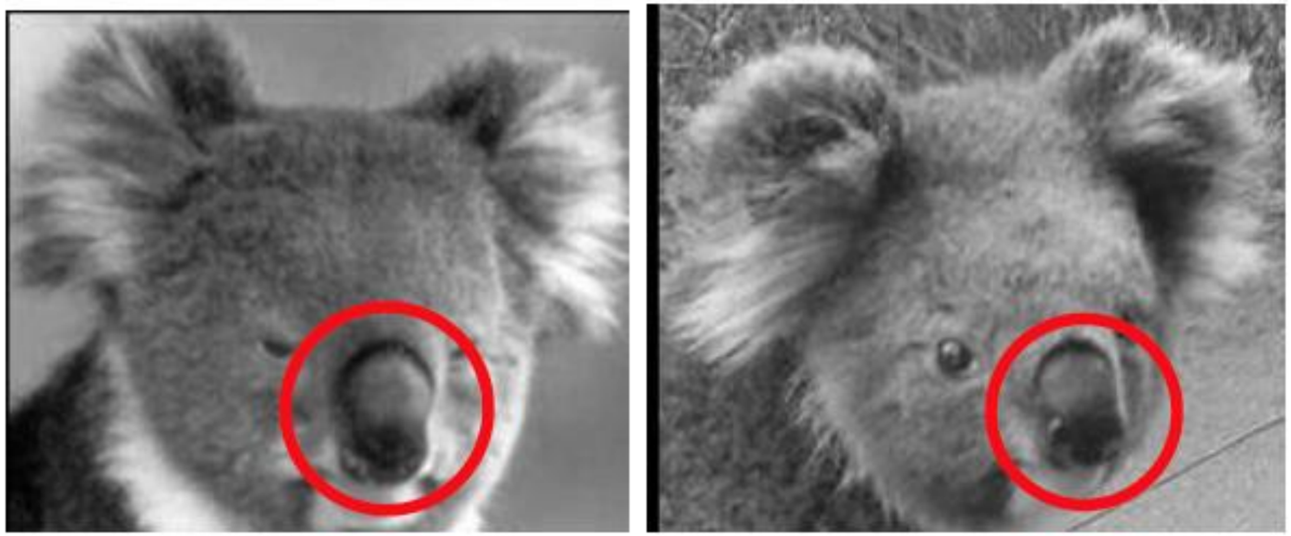
- Repeatability / invariance
  - The same feature point can be found in several images despite geometric and photometric transformations
- Saliency / distinctiveness
  - Each feature point is distinctive
  - Fewer ”false” matches

Goal: Repeatability

No chance to find true matches!

- We want to detect (at least some of) the same points in both images.
- Yet we must be able to run the detection procedure *independently* per image.

Invariance/Repeatability



The feature detector should “fire” at consistent places despite rotation, translation etc.

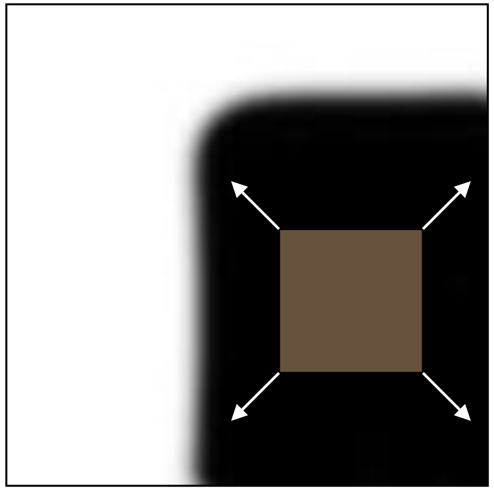
Goal: Distinctiveness

- The feature point should be distinctive enough that it is easy to match
- Should at least be distinctive from other patches nearby

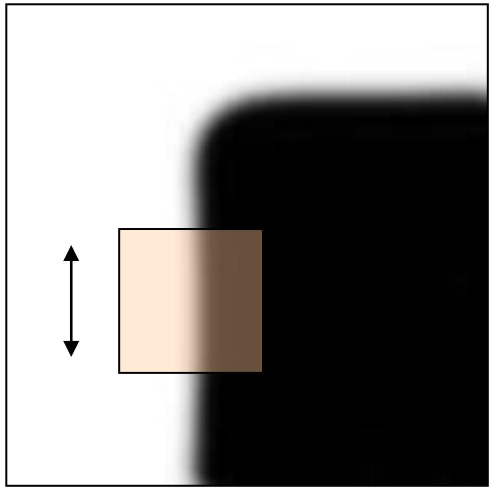
Corner Detection

We should easily recognize the point by looking through a small window

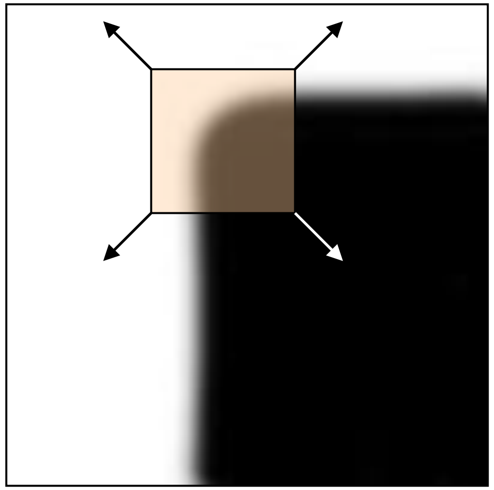
Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in all directions



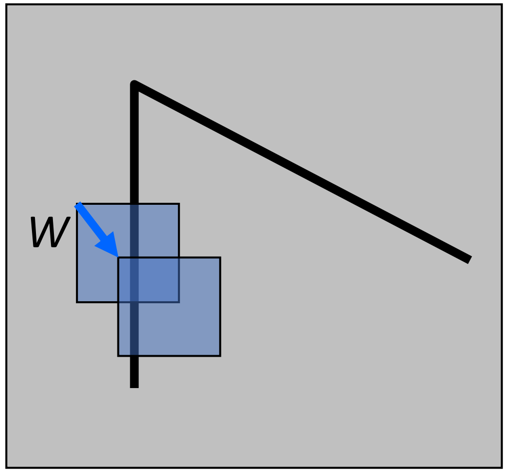
"edge":
no change along the edge direction



"corner":
significant change in all directions

Corner Detection

Consider shifting the window *W* by (*u*,*v*)
how do the pixels in W change?
Write pixels in window as a vector:



$$\phi_0 = [I(0,0), I(0,1), \ldots, I(n,n)]$$

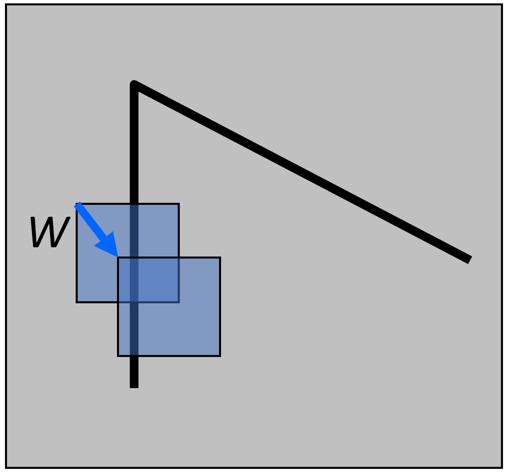
$$\phi_1 = [I(0+u, 0+v), I(0+u, 1+v), \ldots, I(n+u, n+v)]$$

$$\begin{aligned} E(u,v) &= \|\phi_0 - \phi_1\|_2^2 \\ &= \sum_{(x,y)\in W} (I(x,y) - I(x+u, y+v))^2 \end{aligned}$$

Corner Detection

Consider shifting the window *W* by (*u*,*v*)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD “error” E(u,v):



$$E(u,v) = \sum_{(x,y) \in W} \left[I(x+u, y+v) - I(x,y) \right]^2$$

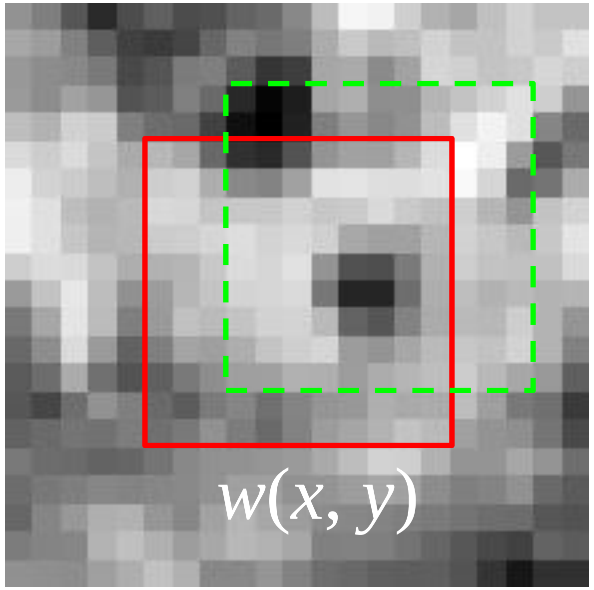
We want E(u,v) to be as high as possible for all u, v!

Corner Detection

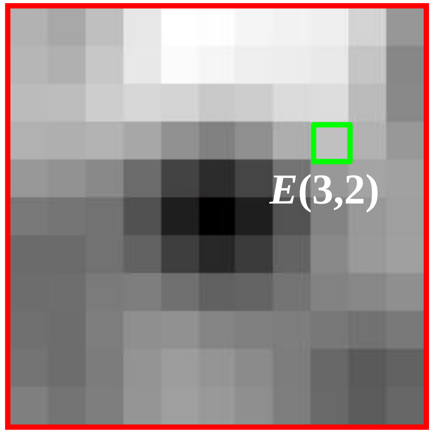
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

$I(x, y)$



$E(u, v)$



Corner Detection

Change in appearance of window $w(x,y)$ for the shift $[u,v]$:

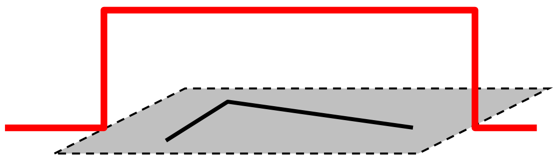
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

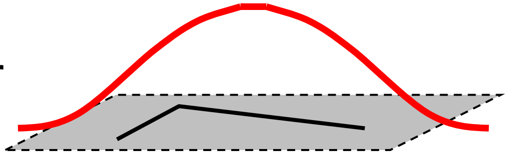
Shifted intensity

Intensity

Window function $w(x,y)$ =



or



1 in window, 0 outside

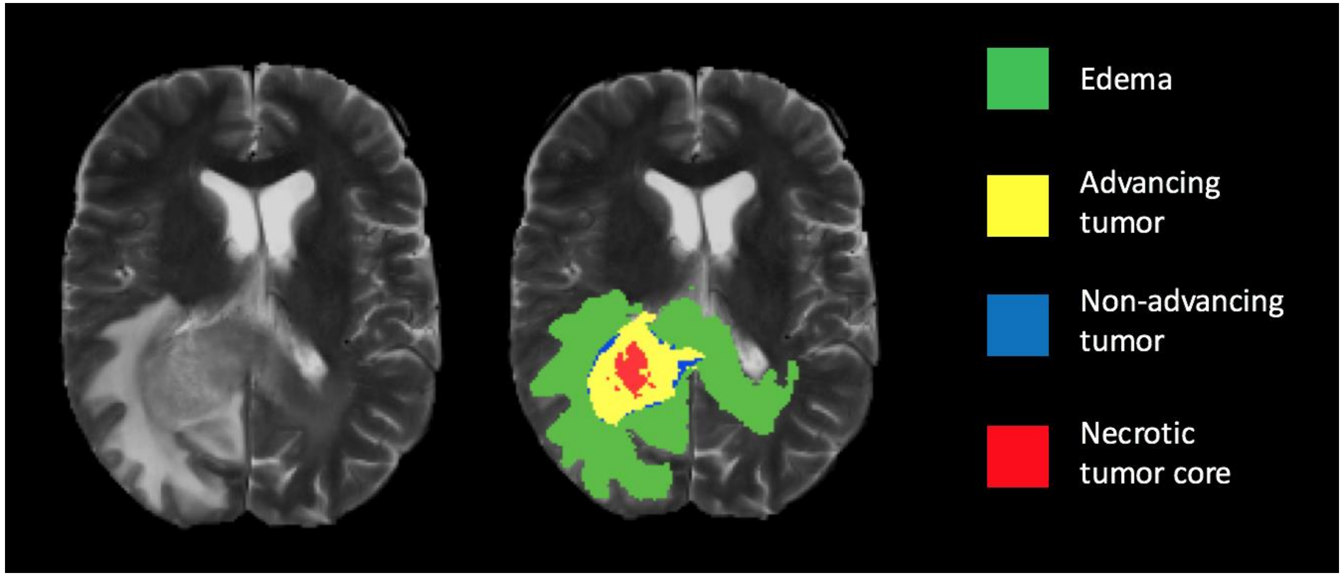
Gaussian

Image Segmentation

1. In computer vision, image segmentation is the process of partitioning an image into multiple segments.
2. The goal of segmenting an image is to change the representation of an image into something that is more meaningful and easier to analyze. It is usually used for locating objects and creating boundaries.

    1. Used in self-driving cars. Autonomous driving is not possible without object detection which involves segmentation.

    2. Used in the healthcare industry. Helpful in segmenting cancer cells and tumors using which their severity can be gauged.



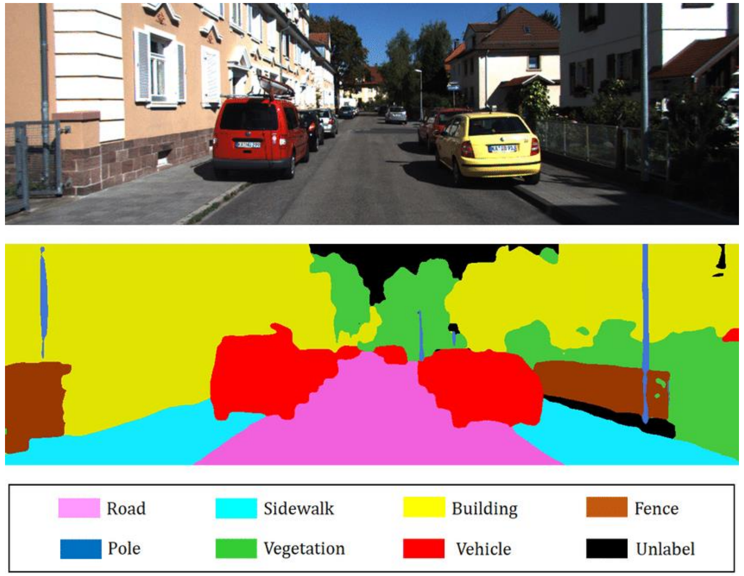


Image Segmentation

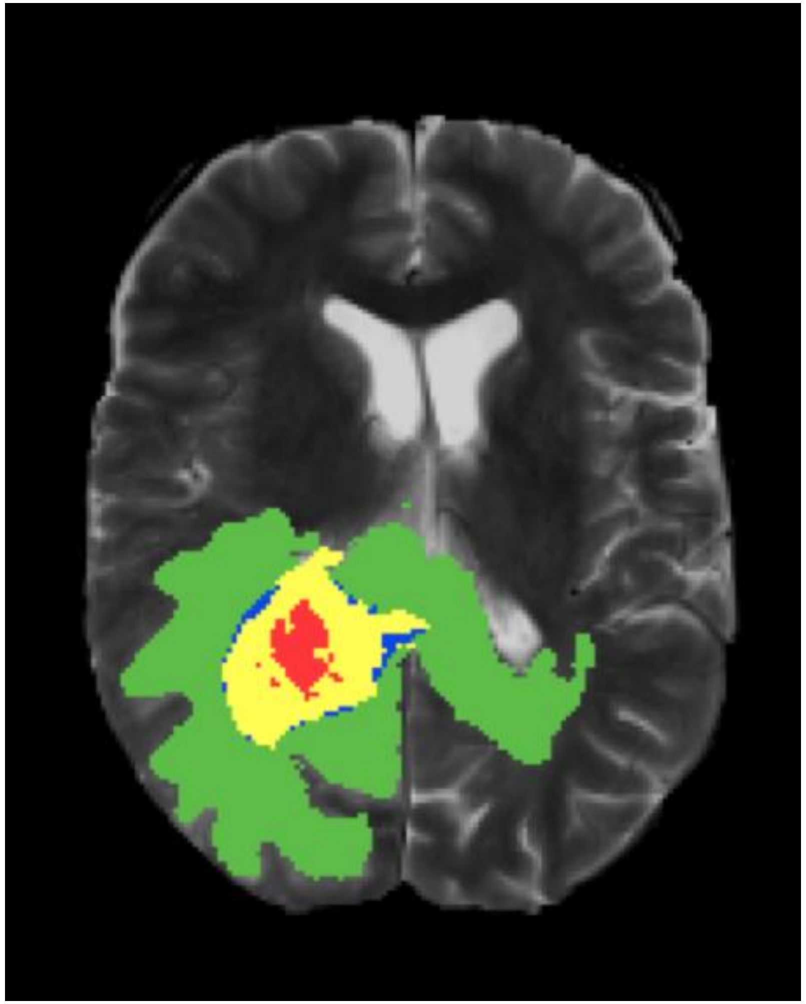


Image Segmentation

- Nowadays, AI offers advanced algorithms to segment images with similar characteristics. (i.e., deep learning models such as convolutional neural networks (CNN).

- Classical segmentation methods:

  - Thresholding (or Otsu's thresholding)
    Returns a single intensity threshold that separate pixels into two classes, foreground and background (i.e., black or white).

  - Edge Detection
    Finding the boundaries of objects within images. It works by detecting discontinuities in brightness.
    Application: Fingerprint recognition

  - Semantic Segmentation
    Assigns a label to every pixel in the image
    Application: Tumor segmentation, Autonomous vehicles

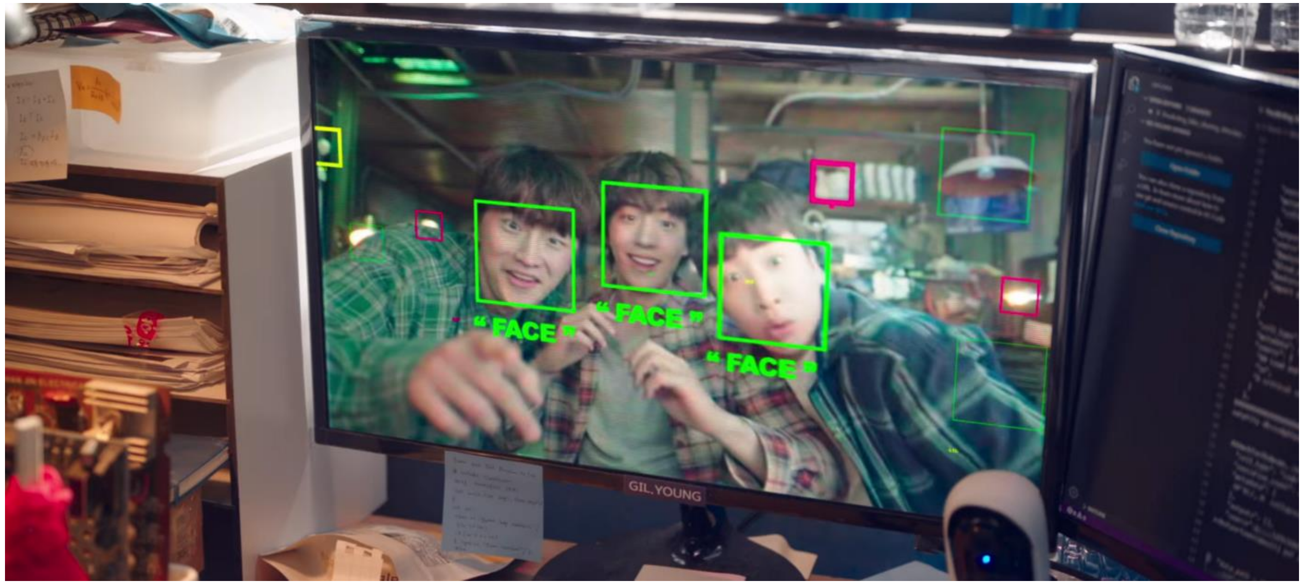
Image Segmentation

Thresholding

Edge Detection/Edge Segmentation

Semantic Segmentation

Image Segmentation



Screengrab from the Korean series ‘Startup (2020)’

Semantic segmentation