Artificial Intelligence II

Part 2: Lecture 1

Brent Davis, with

thanks to

Yalda Mohsenzadeh

Winter 2022

Part 2 Syllabus

- Introduction to Computer Vision:
  - Filtering, Edge detection, Edge types, Image gradients, Canny Edge detector
  - Image segmentation (perceptual grouping, pixel clustering, histogram based methods)
  - Motion: Motion estimation, motion field, optical flow, Methods for optical flow estimation and motion tracking
- Neural Networks
  - Brief history, basic formulation, optimization with gradient descent, layer types (linear, point-wise, nonlinearity), linear classification with perceptron, Tensorflow, Regularizers, Normalization
- Deep learning
  - Batch processing, stochastic gradient descent, backpropagation
- Neural networks for images,
  - convolutional neural networks (multiple channels, pooling, strides), receptive fields, unit visualization, important network architectures (ex: Transformer architecture) and their tricks
- Representational learning, unsupervised/self-supervised learning with neural network

What is AI?

- What is not AI?
- How do we draw the line between what is AI and what is not?

What is AI?

- ‘Sci-Fi’ Term – Artificial General Intelligence
- How do we differentiate between kinds of intelligence?

What is AI?

- Hard problems of AI –
  - What is consciousness?
  - ‘Turing Test’
  - Embodied vs Disembodied Intelligence
  - Distributed vs Centralized Intelligence
  - Mistake correcting

What is AI?

- Symbolic Representations
- ‘Bias Free’ Learning, i.e., all from scratch
- Brain model vs Bacteria model

Kinds of AI

- Independent AI
  - Fully autonomous systems
- Dependent AI
  - Intelligence Augmentation
- Interdependent AI
  - Cyborg(?)

Computer Vision

Introduction

Filtering

A simple Visual World

To discover from images what is present in the world, where things are, what actions are taking place, to predict and anticipate events in the world

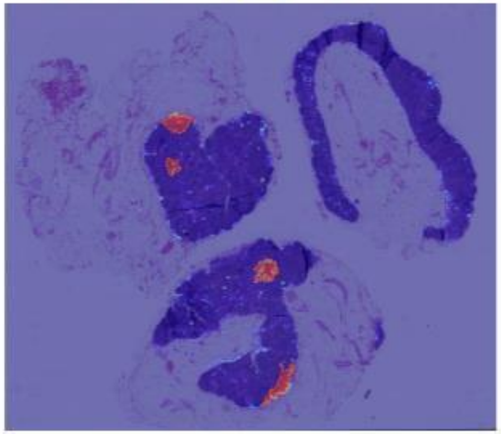
What is Computer Vision?

- The ability of computers to see
  - Image Understanding
  - Machine Vision
  - Robot Vision
  - Image Analysis
  - Video Understanding

Exciting Time for Computer Vision

Robotics

Medical applications



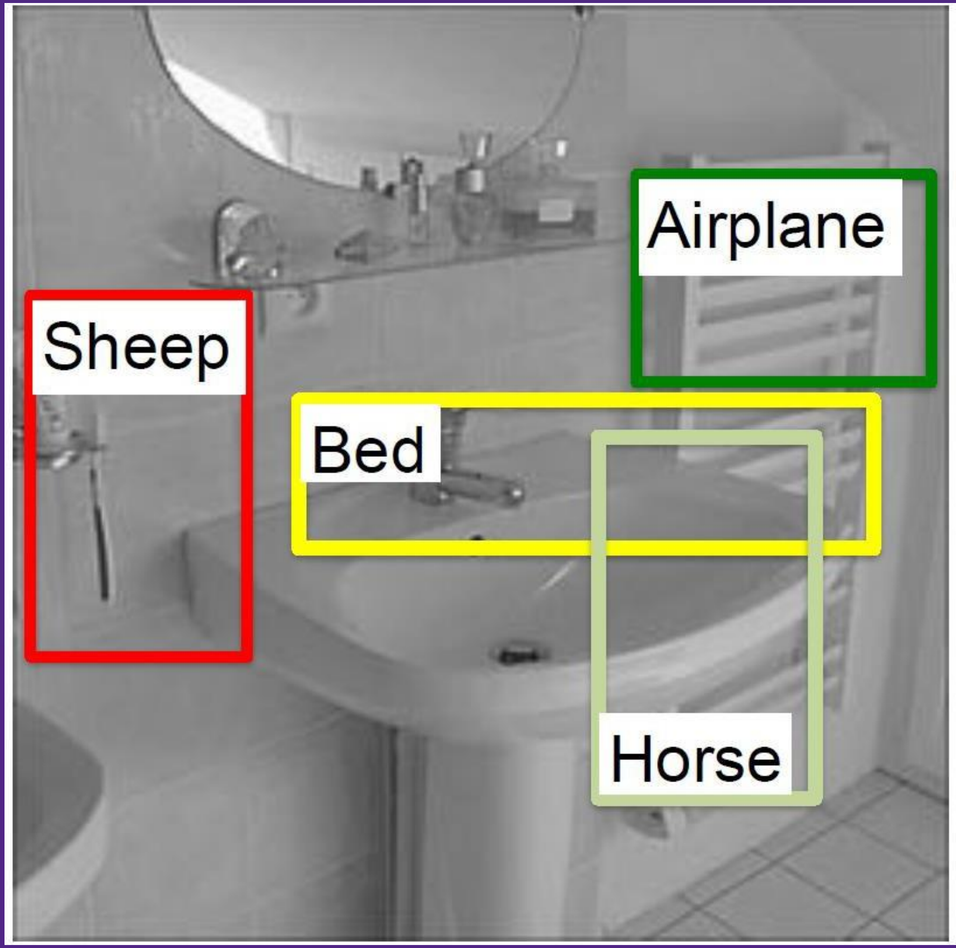
Gaming

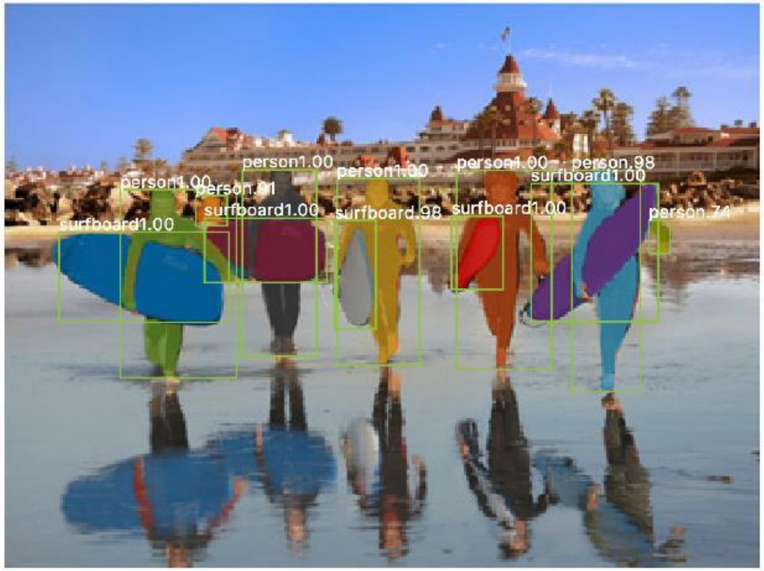
Driving

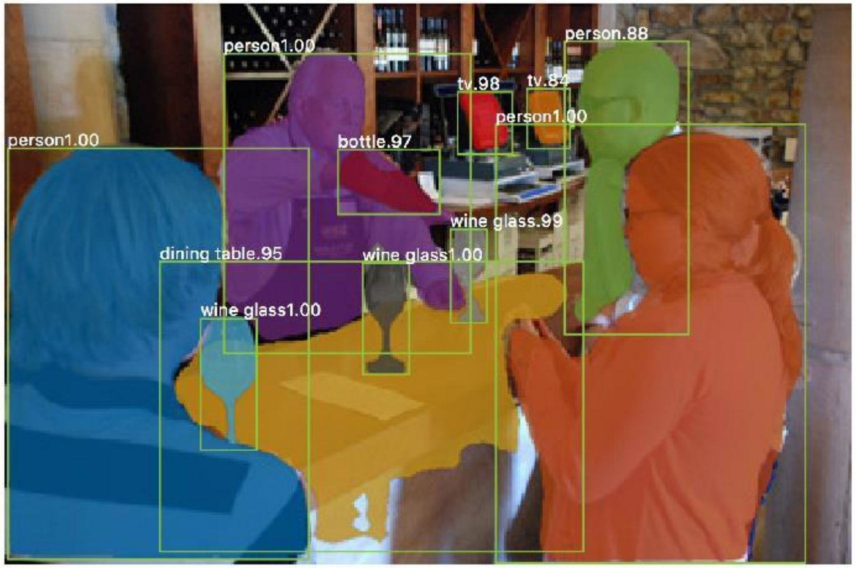
Mobile devices

Accessibility

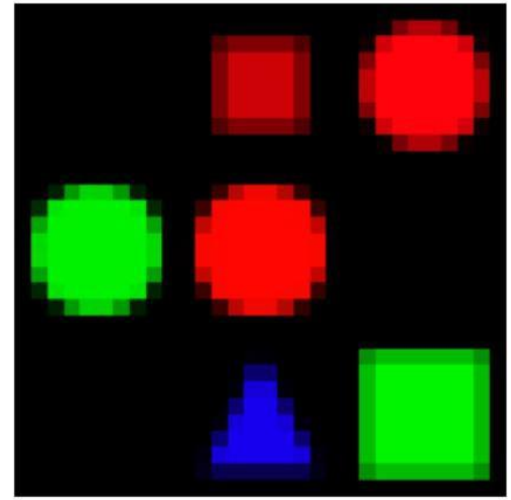
Just a few years ago







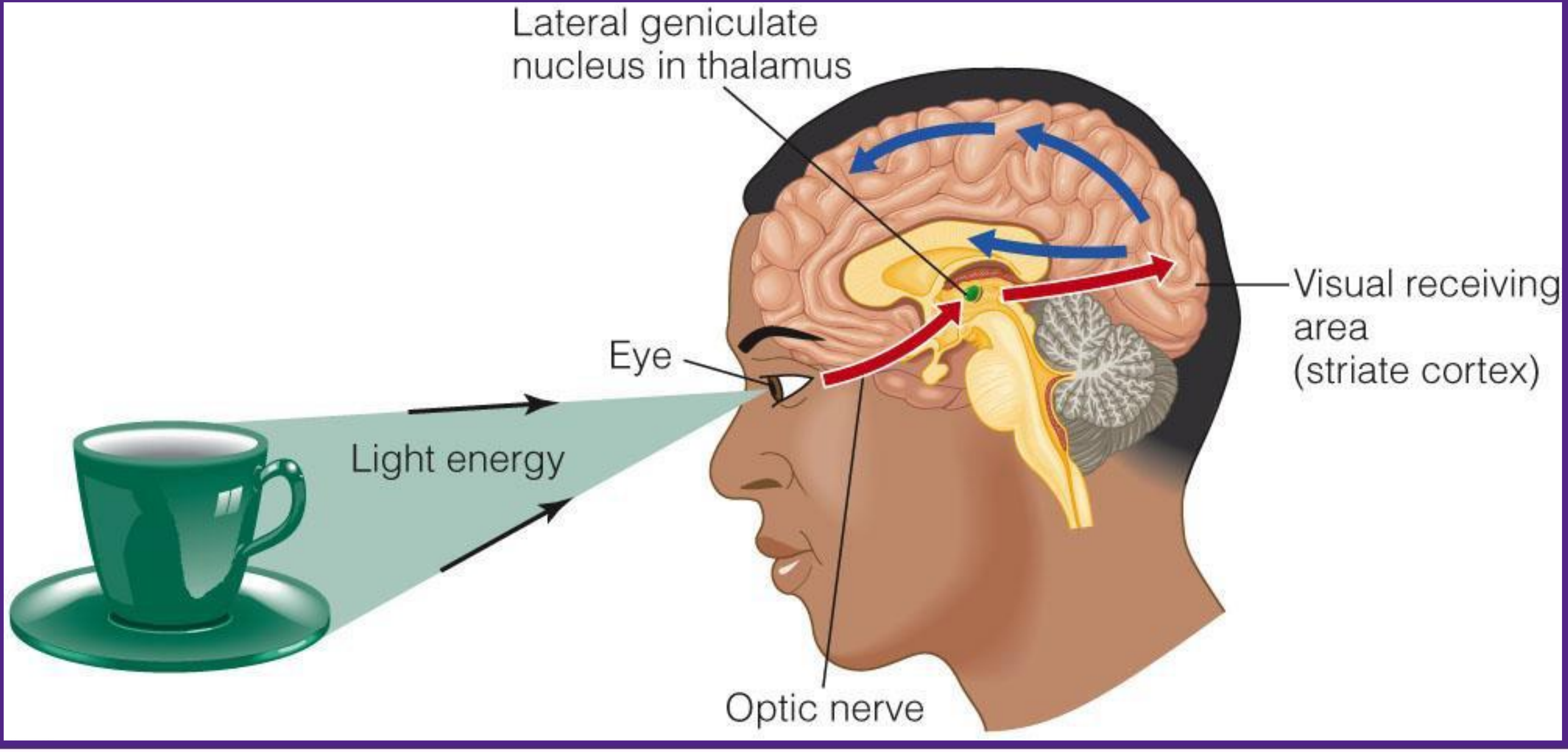
Mask RCNN, He et al. 2017

<table>
<tr>
<td>
what color is the vase?</td>
<td>
is the bus full of passengers?</td>
<td>
is there a red shape above a circle?</td>
</tr>
<tr>
<td><code>classify[color](
  attend[vase])</code></td>
<td><code>measure[is](
  combine[and](
    attend[bus],
    attend[full])</code></td>
<td><code>measure[is](
  combine[and](
   attend[red],
   re-attend[above](
    attend[circle])))</code></td>
</tr>
<tr>
<td>green (green)</td>
<td>yes (yes)</td>
<td>no (no)</td>
</tr>
</table>

Neural Module Networks, Andreas et al. 2017

Human Brain: A View of Visual System

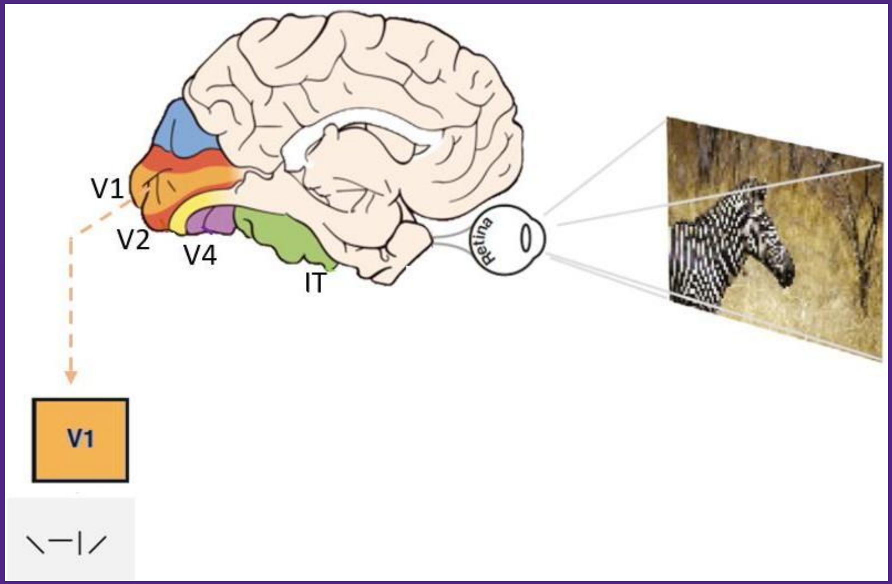
- Vision starts with the eyes, but truly takes place in the brain



Goldstein and Brockmole, 2016

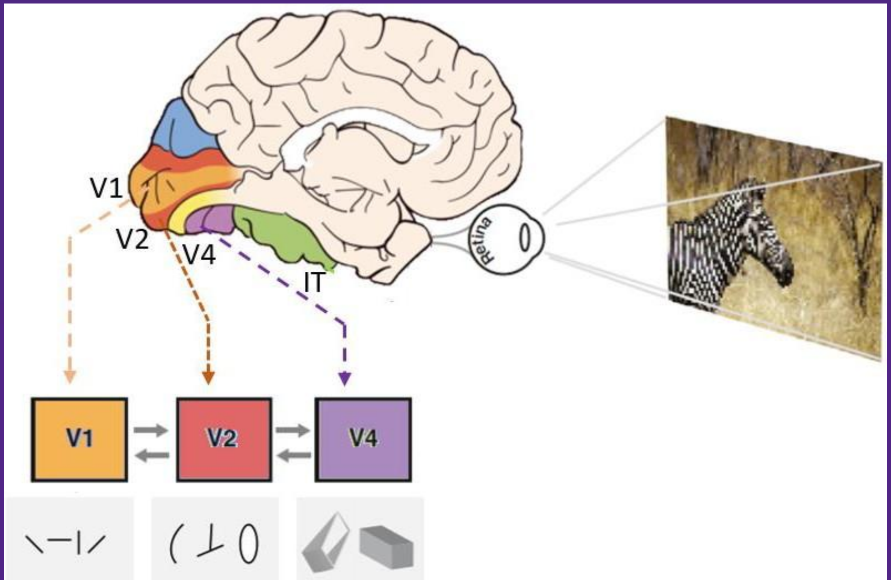
Low level processing
Primary Visual Cortex

- Low level operations
- Filtering, edge detection



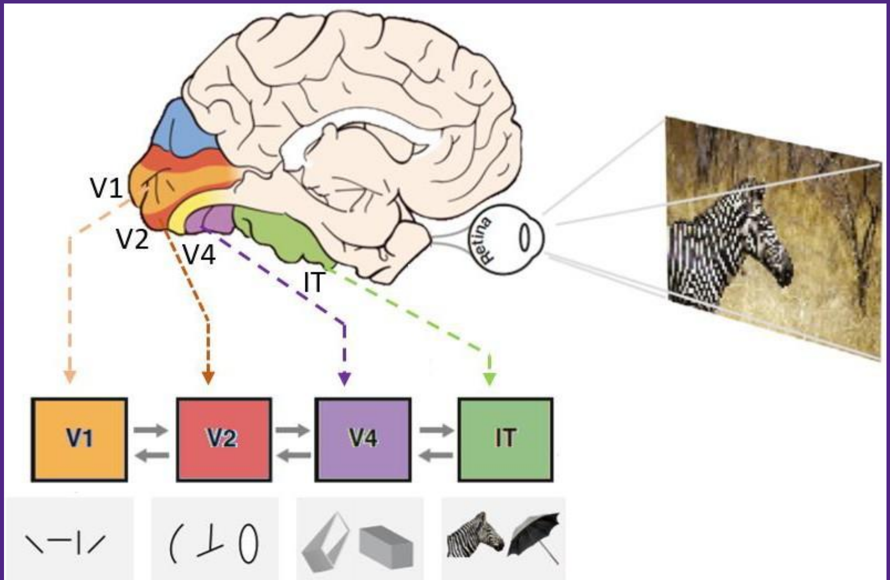
Mid level processing

- Mid level operations
- Shape formation, 3D shape reconstruction, …

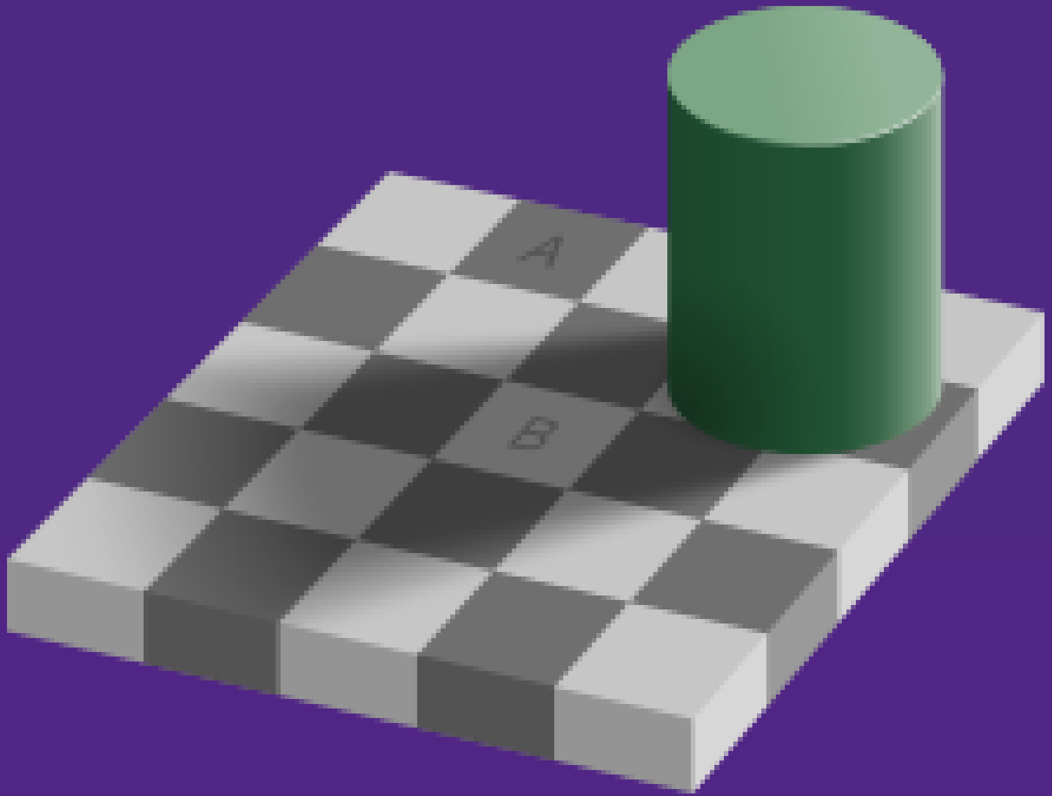


High level processing

- High level operations
- Recognition of objects, people, places, events

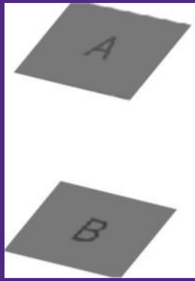


Perception versus measurement



Edward Adelson

Perception versus measurement



Western Science

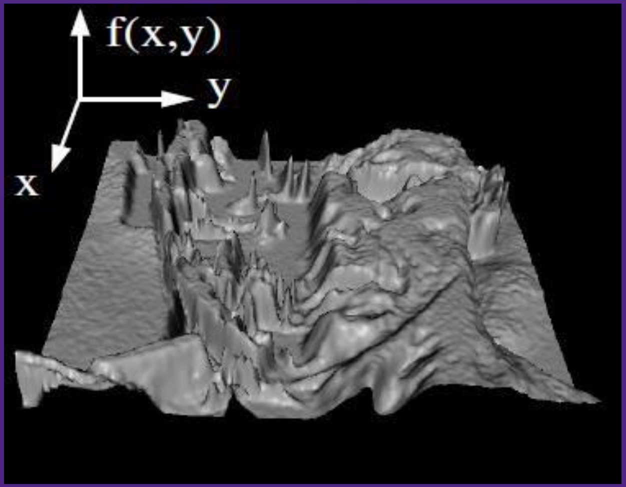
Image

- 2-D array of numbers (intensity values, gray levels)
- gray level 0 (black) to 255 (white)
- Color images are 3 2-D arrays of numbers
  - Red
  - Green
  - Blue
- Resolution (number of rows and columns)
  - 128 x 128
  - 256 x 256
  - 640 x 480

Images as functions

- We can think of an image as a
- function, f, from R^2 to R:
- *f(x, y)* gives the intensity at position
- *(x,y)*
- *f(x,y)* is proportional to the brightness at *(x,y)*
- Realistically, we expect the image only to be defined over a rectangle, with a finite range:
  - *f: [a,b] x[c,d] -> [0, 255]*
    - Standard range for gray scale images is
  - *(0, 1, 2, …, 255)*
- A color image is just three functions pasted together. We can write this as vector-valued function

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$



A digital image

- In computer vision we usually operate on **digital** (**discrete**) images:
- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)
- If our samples are Δ apart, we can write this as:

$$f[i,j] = Quantize\{f(i\Delta, j\Delta)\}$$

- The image can now be represented as a matrix of integer values

12	15	120	128	128	128	130
240	120	18	120	121	128	128
252	248	22	13	112	133	133
255	243	230	11	20	128	125
24	32	251	255	26	127	123
10	15	252	253	18	120	128
8	14	18	176	154	128	127
129	110	120	127	128	128	130

Image processing

- **Image processing operation**: defining a new image g in terms of an existing image f
- We can transform either the domain or the range of f.
- **Range transformation:**
- $g(x, y) = t\,(f(x, y))$

Image processing

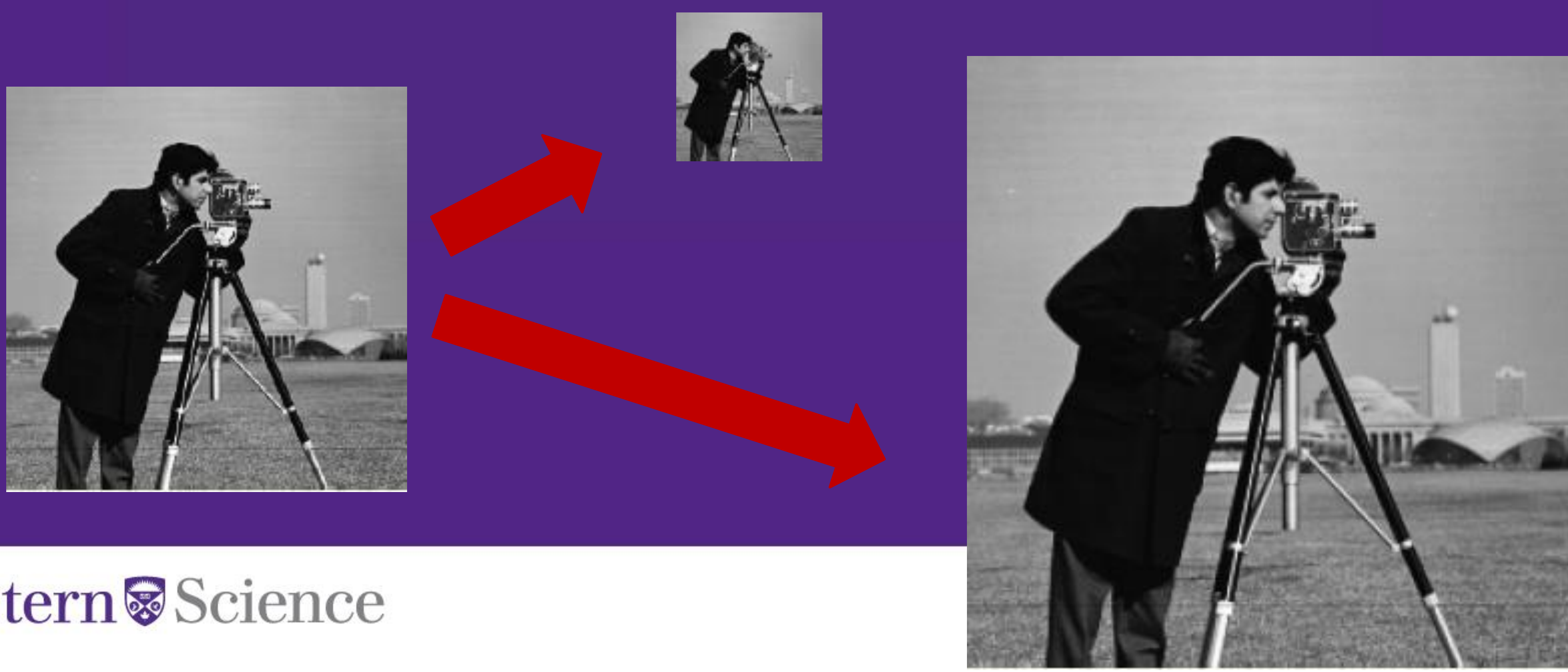
- Digital negative
- $g(x, y) = 255 - f(x, y)$

Image processing

- Improving the contrast in the picture

Image processing

- Some operations preserve the range but change the domain of f:
- $g(x, y) = f(t_x(x, y), t_y(x, y))$



Common Geometric Transformation

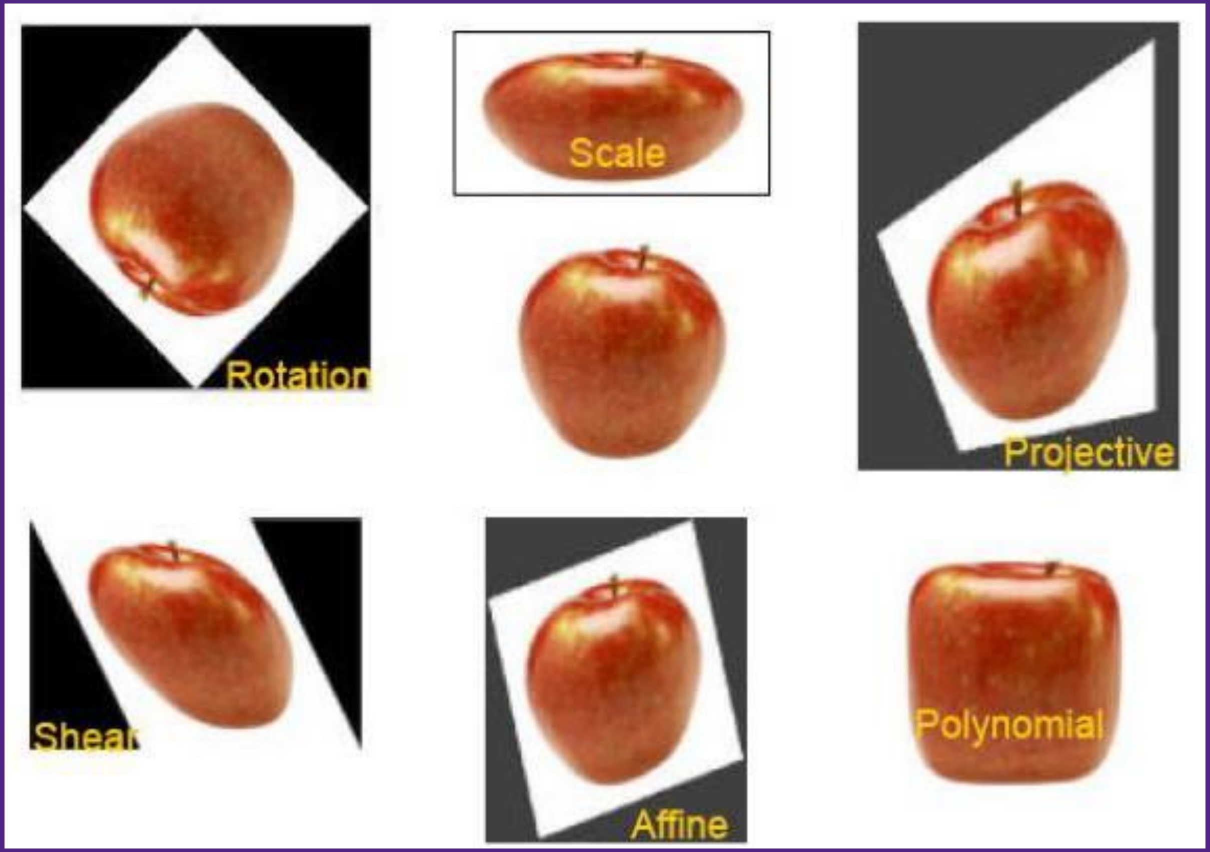
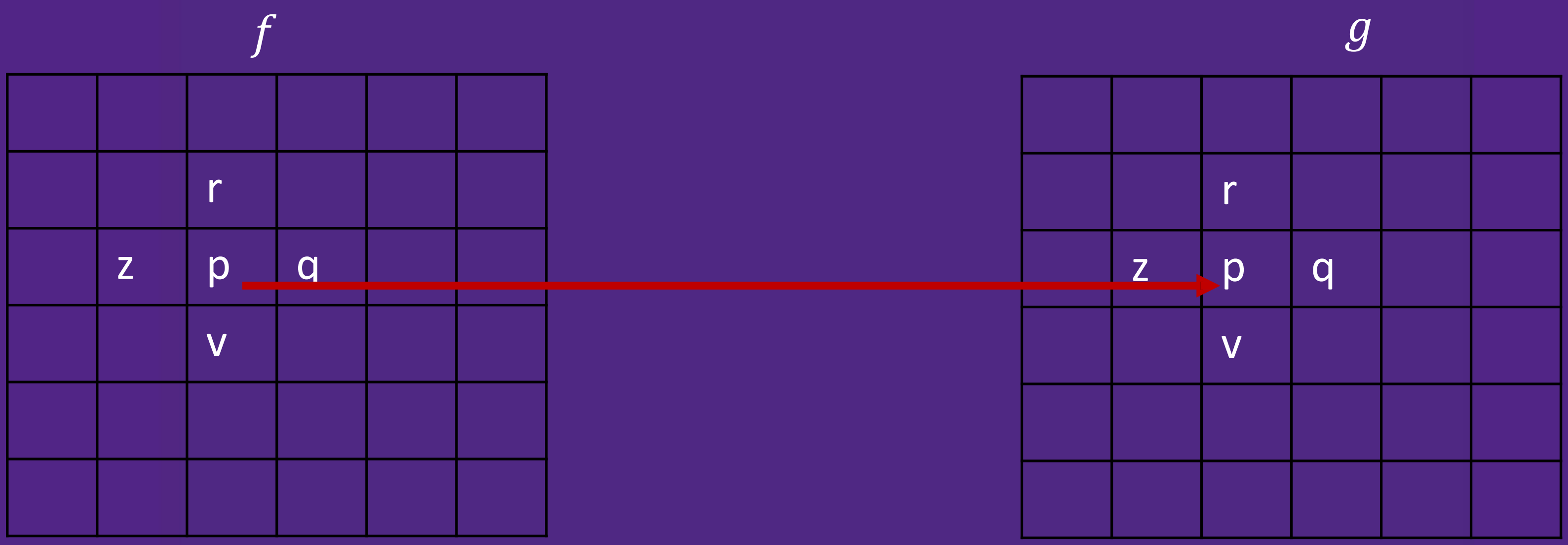


Image Processing

- Still other operations operate on both domain and the range of f

Image Processing: Filtering

- Modifies pixels based on neighborhood

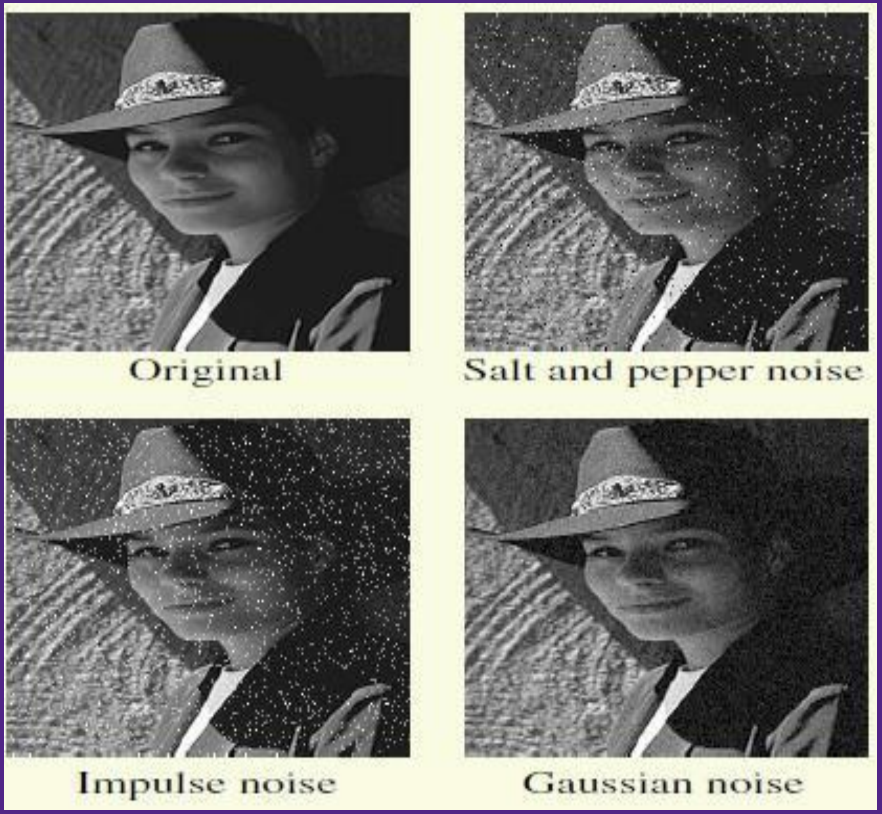


$$g(p) = 2f(p) + 0.5f(r) + 0.5f(q) + 0.3f(v) + 0.3f(z)$$

- Useful to:
  - Noise reduction, integrate information over constant regions, scale change, detect changes

Filtering Application: Noise Reduction

- Common types of noise:
  - **Salt and pepper noise**: contains random occurrences of black and white pixels
  - **Impulse noise**: contains random occurrences of white pixels
  - **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution



- Image processing is useful for noise reduction

Noise Reduction by Mean Filtering

- How can we smooth away noise in a single image?

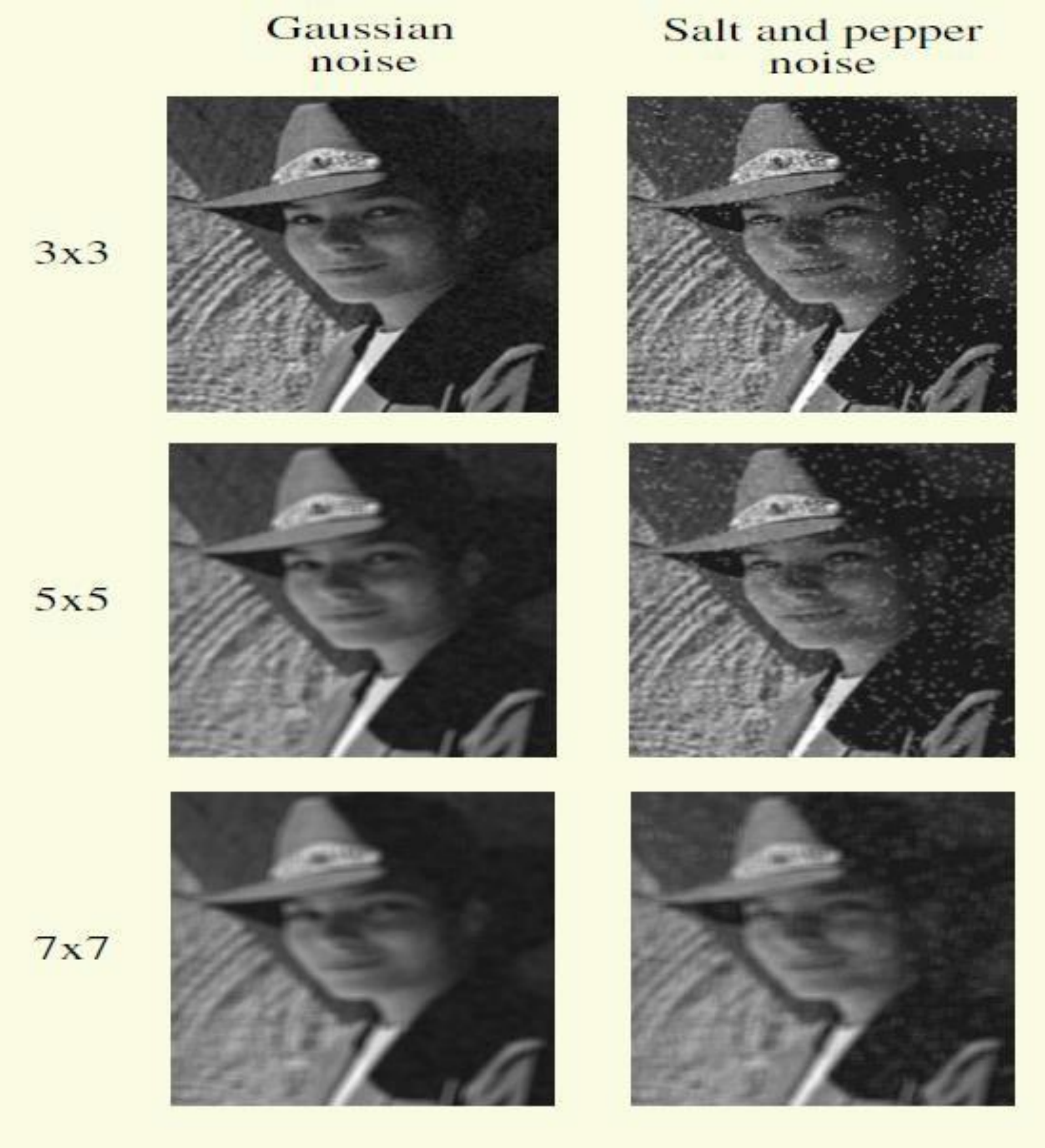
$f(x, y)$

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

$g(x, y)$

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

Effect of mean filters



Convolution

- Assume the averaging window as (2k+1) x (2k+1):

$$g[i,j] - \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} f[i-u, j-v]$$

  - Let's generalize the idea by allowing different weights for different neighboring pixels:

$$g[i,j] - \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] f[i-u, j-v]$$

  - This is called a convolution:

$$g = h * f$$

  - *h* is called the "filter", "kernel", or "mask".

Convolution

$$\underset{g}{\begin{bmatrix} 1 & 3 & 2 & 1 \\ 2 & 9 & 1 & 1 \\ 1 & 3 & 2 & 3 \\ 5 & 6 & 1 & 2 \end{bmatrix}} * \underset{h}{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}} = \underset{f}{\begin{bmatrix} -1 & 10 \\ 4 & 12 \end{bmatrix}}$$

$$g(i,j) = h * f = \sigma_{u,v}\, h(u,v)\, f(i-u, j-v)$$

Mean Kernel (also called box filter)

- Kernel for a 3x3 mean filter:

$f[x, y]$

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

$h[u, v]$

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

Gaussian Filtering

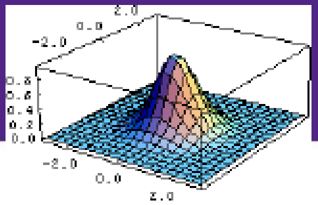
- A Gaussian kernel gives less weights to pixels further from the center of the window

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

$h[u,v]$

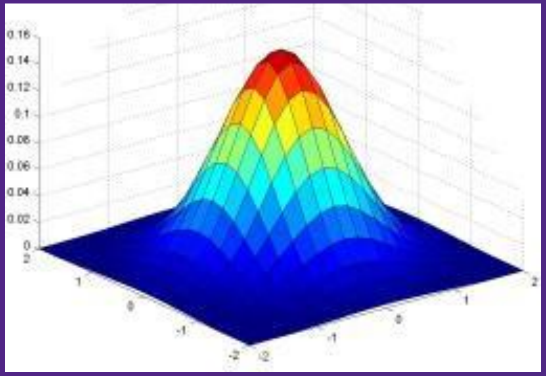
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

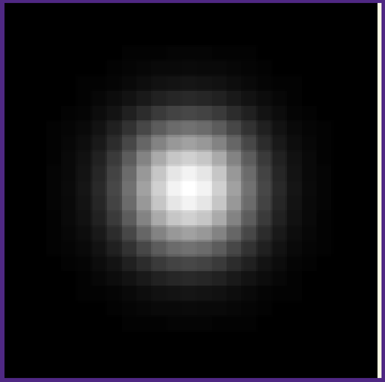
Discrete Gaussian kernel



Gaussian Kernel

- Weight contributions of neighboring pixels by nearness



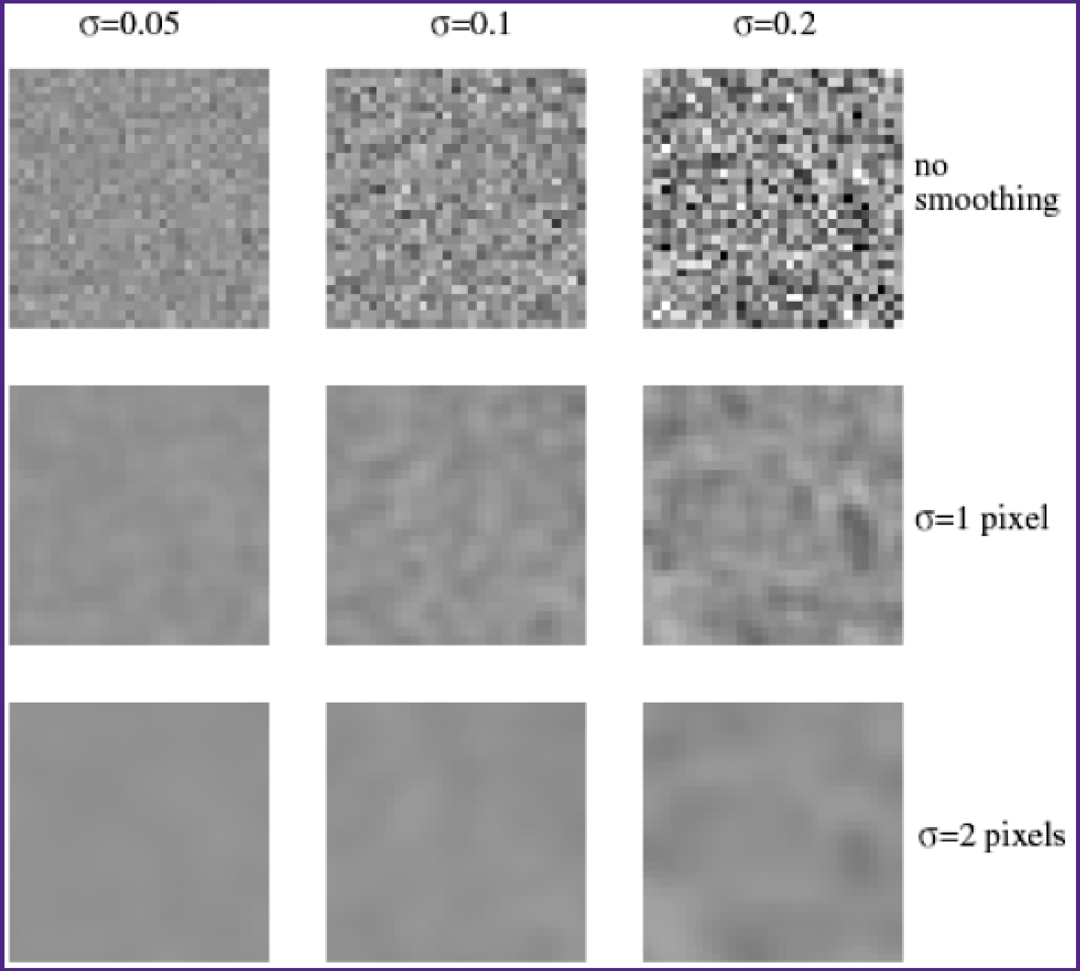


0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

$$G_{\sigma(x,y)} = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

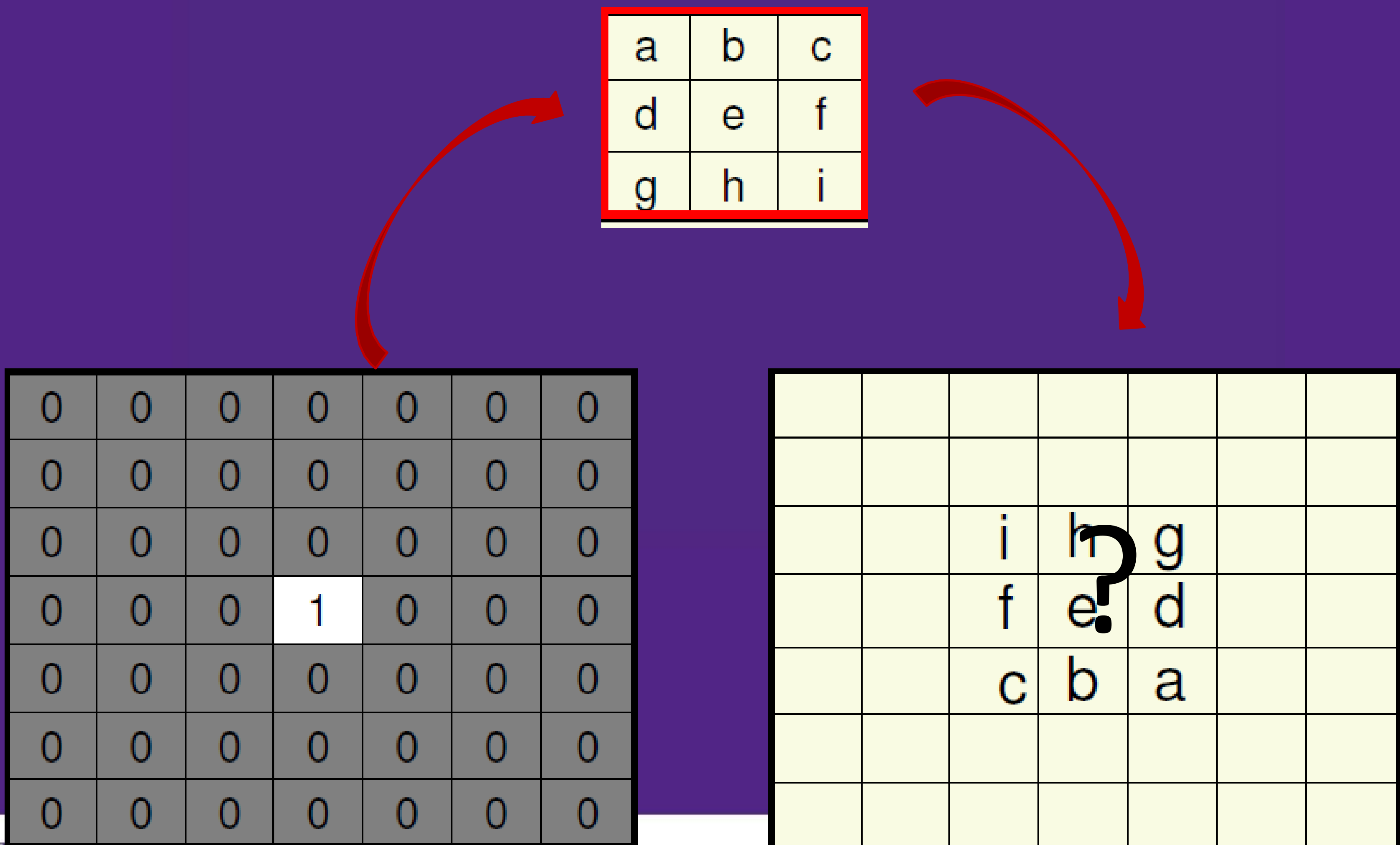
- Constant factor at front makes volume sum to 1 (we should normalize weights to sum to 1 in any case)
- What happens if you increase σ?

Gaussian Filtering



- Each row shows smoothing with Gaussians of different width
- Each column shows different realizations of an image of Gaussian noise.

Filtering an impulse



Median Filter

- Median of {1, 2, 25, 3, 24, 22, 20, 21, 23}

  ={1, 2, 3, 20, 21, 22, 23, 24, 25} =21

1	2	25
3	24	22
20	21	23

→

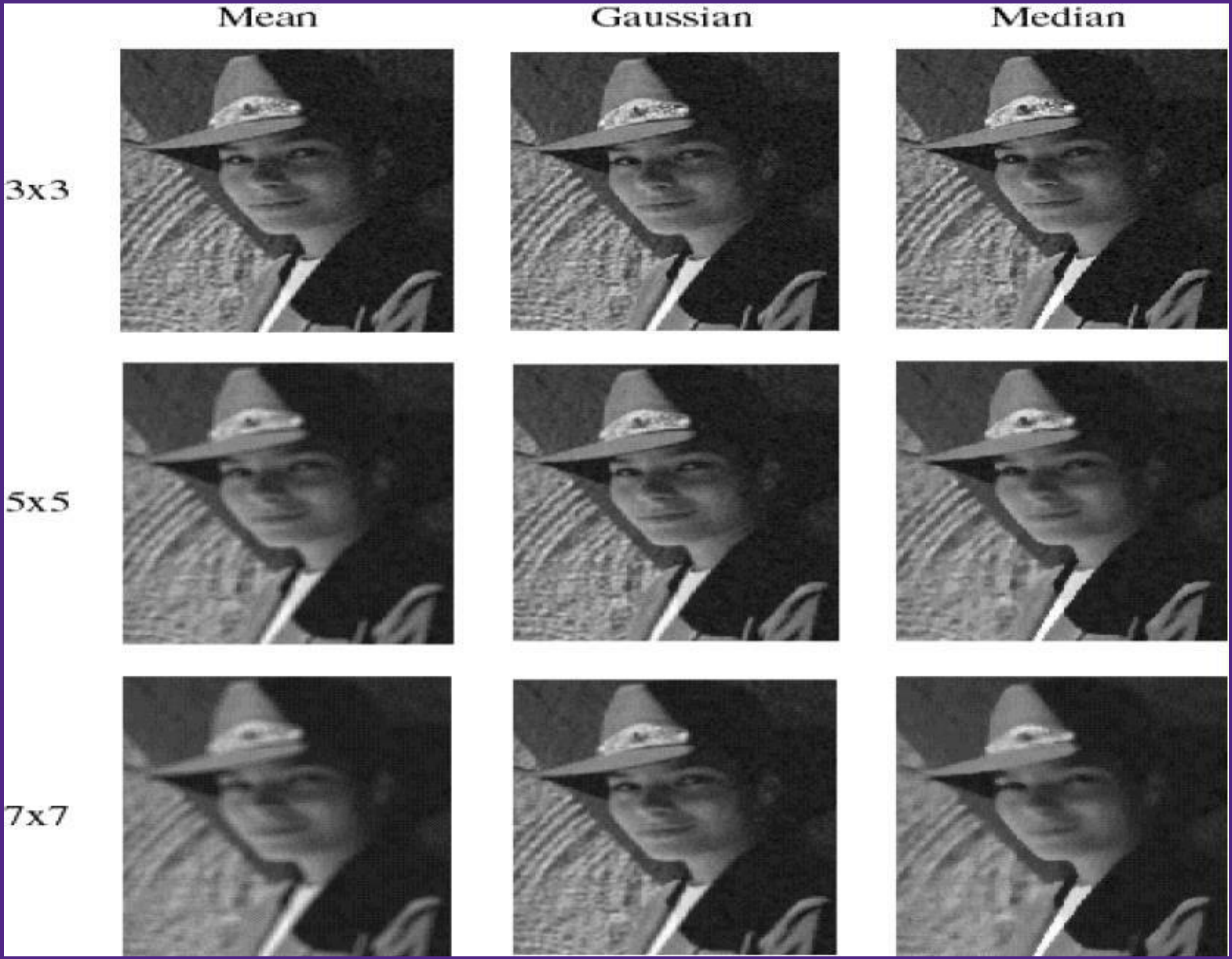
x	x	x
x	21	x
x	x	x

- Median filter selects the Median intensity over a window.
- Median filter preserves sharp detail better than mean filter, it is not so prone to over-smoothing.
- Is a median filter a kind of convolution?

Salt and Pepper Noise

- Comparison

Gaussian Noise



Face of faces