CS442/542b: Artificial Intelligence II
Prof. Olga Veksler

Lecture 12: Computer Vision

Introduction

Filtering

Most slides are from David Jacobs (UMD), David Lowe (UBC), Steve Seitz (UW), A.Efros (CMU), Xin Li (WVU)

Outline

- Very Brief Intro to Computer Vision
- Digital Images
- Image Filtering
  - noise reduction

Every picture tells a story

- Goal of computer vision is to write computer programs that can interpret images

Can computers match human perception?

- Yes and no (but mostly no!)
  - humans are much better at “hard” things
  - computers can be better at “easy” things

Why study Computer Vision?

- Images and movies are everywhere
- Fast-growing collection of useful applications
  - matching and modifying images from digital cameras
  - movie post-processing
  - building representations of the 3D world from pictures
  - medical imaging, household robots, security, traffic control, cell phone location, face finding, video game interfaces, ...
- Greater understanding of human vision and the brain
  - 25% of the human brain is devoted to vision

Low level processing

- Low level operations
  - Noise reduction, image enhancement, feature detection, region segmentation

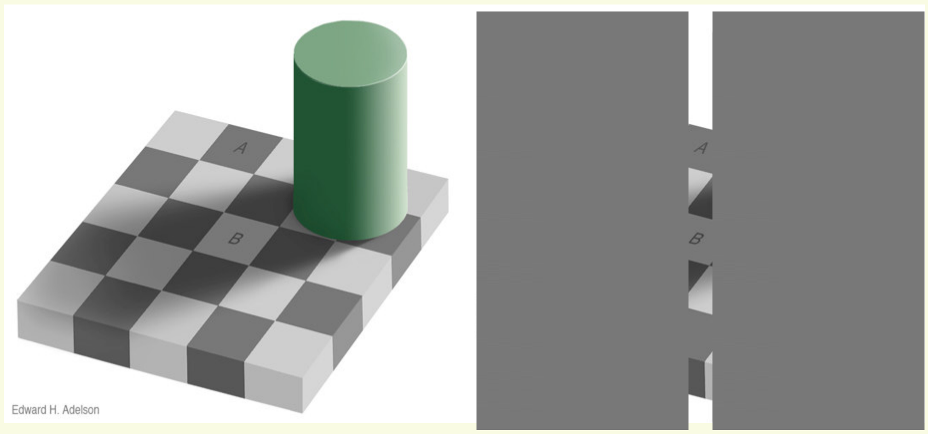
Mid level processing

- Mid level operations
  - 3D shape reconstruction, motion estimation

High level processing

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

Vision is inferential: Illumination



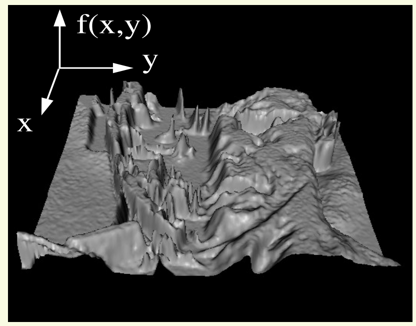
http://web.mit.edu/persci/people/adelson/checkershadow_illusion.html

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 a “vector-valued” function:

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

Images as functions



What is 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:

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

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

$j \rightarrow$, $i \downarrow$

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

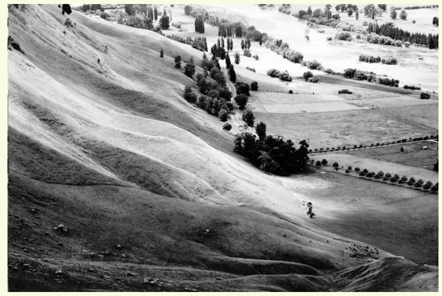
Image processing

- An **image processing** operation typically defines a new image g in terms of an existing image f
- We can transform either the domain or the range of f.
- **Range transformation**:
  - $g(x,y)=t\,(f\,(x,y)\,)$
- What's kinds of operations can this perform?

Image processing

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

Image processing



- One example: can improve the contrast in the picture

Image processing

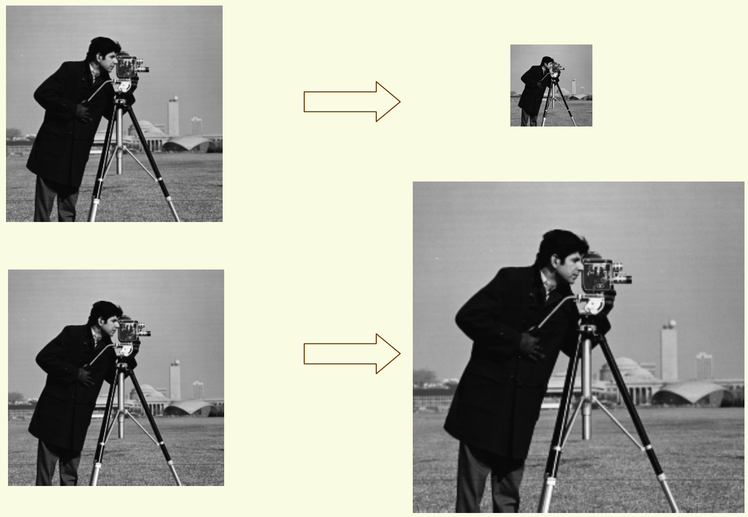
- Some operations preserve the range but change the domain of f :

$$g(x, y) = f(t_x(x, y), t_y(x, y))$$

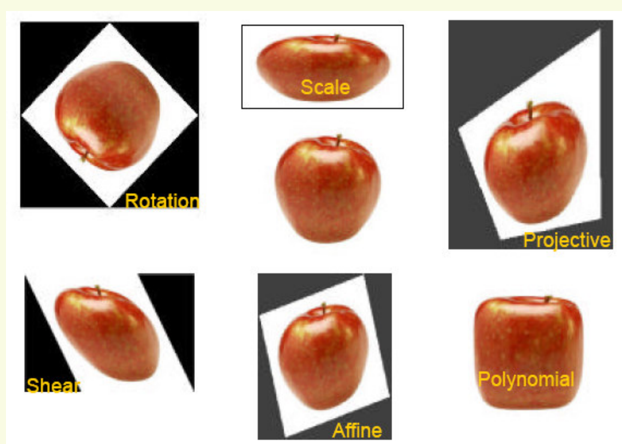
- What kinds of operations can this perform?

Image processing

$$g(x, y) = f(t_x(x, y), t_y(x, y))$$



Common Geometric Transformation



Widely used in computer graphics to generate special effects

MATLAB functions: griddata, interp2, maketform, imtransform

Image processing

- All sorts of other transformations

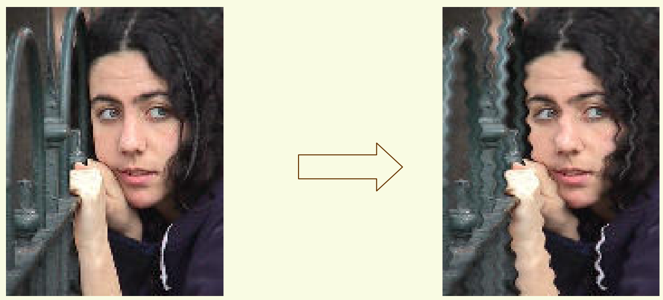
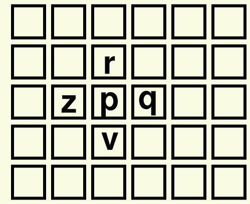


Image processing

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

Image Processing: Filtering



- Modifies pixels based on neighborhood
- Local methods are simplest
  - Change pixel based on its immediate neighbors
- Linear means linear combination of neighbors

$$g(p) = 2f(p) + \frac{1}{2}f(r) + \frac{1}{2}f(q) + \frac{1}{3}f(v) + \frac{1}{3}f(z)$$

- Linear methods simplest.
- Useful to:
  - Noise reduction
  - Integrate information over constant regions
  - Scale change
  - Detect changes

Filtering Application: Noise Reduction

- Image processing is useful for noise reduction...

Original — Salt and pepper noise — Impulse noise — Gaussian noise

- 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

Ideal noise reduction

- Given a camera and a still scene, how can you reduce noise?

Practical noise reduction

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

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	100	130	110	120	110	0	0
0	0	0	110	90	100	90	100	0	0
0	0	0	130	100	90	130	110	0	0
0	0	0	120	100	130	110	120	0	0
0	0	0	90	110	80	120	100	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

Mean filtering

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

Gaussian noise

Salt and pepper noise

3x3

5x5

7x7

Convolution

- Let's write this down as an equation. Assume the averaging window is (2k+1)x(2k+1):

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

- We can generalize this idea by allowing different weights for different neighboring pixels:

$$g[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] f[i-u, j-v]$$

- This is called a **convolution** operation and written:

$$g = H * f$$

- H is called the "filter," "kernel," or "mask."
- The above allows negative filter indices. When you implement need to use: H[u+k,v+k] instead of H[u,v]

Convolution

f

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

H

1/13

1	1	1
1	5	1
1	1	1

g

X	X	X	X	X	X
X	6.9				X
X					X
X					X
X					X
X	X	X	X	X	X

1/13.(10x1 + 11x1 + 10x1 + 9x1 + 10x5 + 11x1+10x1 +9x1+10x1) =
1/13.(90) ≈ 6.9

Slide credit: Christopher Rasmussen

Mean kernel (also called box filter)

- What's the 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 weight to pixels further from the center of the window

f(x,y)

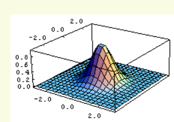
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$H[u, v]$

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

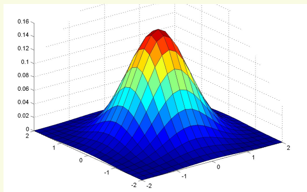
discrete Gaussian kernel

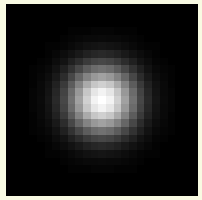
- This kernel is an approximation of a Gaussian function:



Gaussian Kernel

- Idea: 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

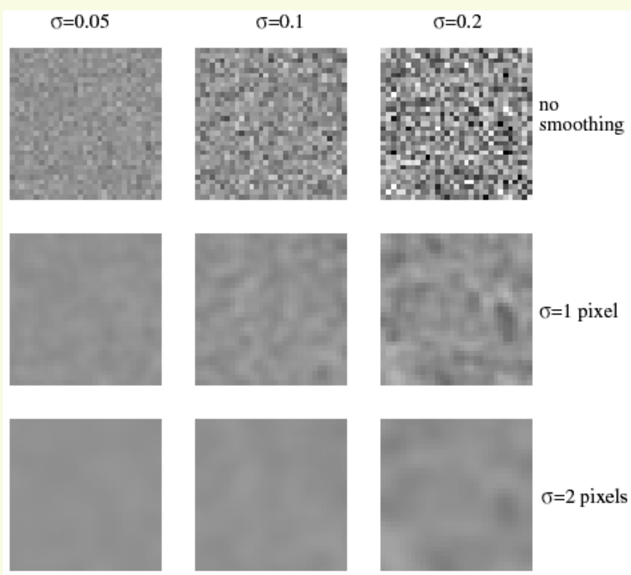
5 x 5, σ = 1

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

- constant factor at front makes volume sum to 1 (can be ignored, as we should normalize weights to sum to 1 in any case).
- What happens if you increase σ ?

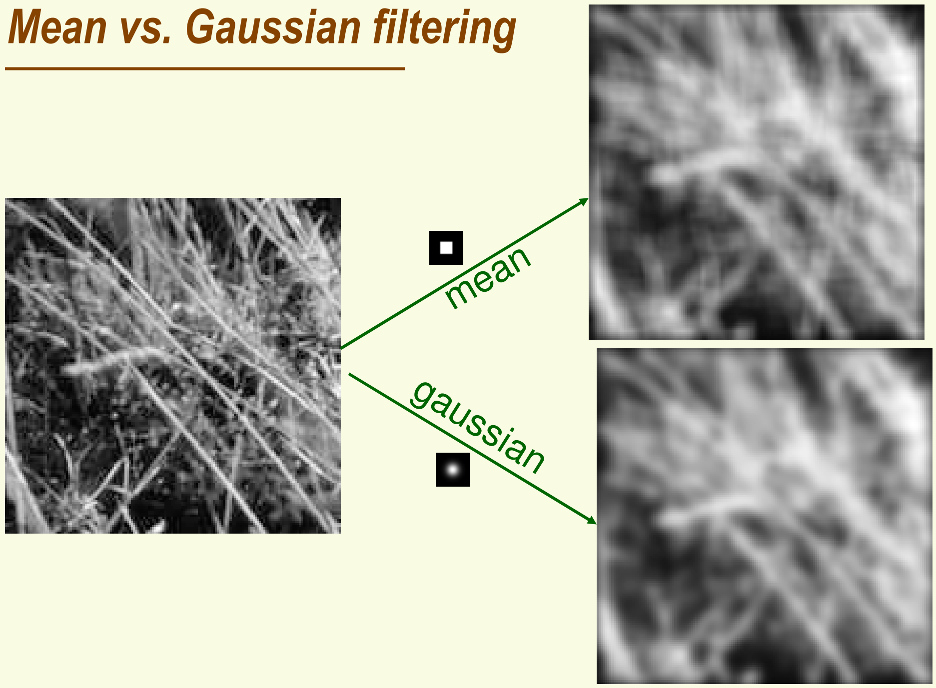
Slide credit: Christopher Rasmussen

Gaussian Filtering



The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realizations of an image of gaussian noise.

Mean vs. Gaussian filtering



Filtering an impulse

$H[u, v]$

a	b	c
d	e	f
g	h	i

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	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

g(x,y)

		i	h	g		
		f	e	d		
		c	b	a		

Efficient Implementation

- Both the BOX (mean) filter and the Gaussian filter are **separable** into two 1D convolutions:
  - First convolve each row with a 1D filter
  - Then convolve each column with a 1D filter.

Box Filter

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \end{bmatrix} \circ \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{bmatrix}$$

Gaussian Filter

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

$$g[i,j] = \sum_{u=-k}^{k}\sum_{v=-k}^{k} H[u,v] f[i-u,j-v] = \sum_{u=-k}^{k}\sum_{v=-k}^{k} exp\left(-\frac{(u^2+v^2)}{2\sigma^2}\right) f[i-u,j-v] =$$

$$\sum_{u=-k}^{k}\sum_{v=-k}^{k} exp\left(-\frac{u^2}{2\sigma^2}\right) exp\left(-\frac{v^2}{2\sigma^2}\right) f[i-u,j-v] =$$

$$\sum_{u=-k}^{k} exp\left(-\frac{u^2}{2\sigma^2}\right) \sum_{v=-k}^{k} exp\left(-\frac{v^2}{2\sigma^2}\right) f[i-u,j-v]$$

Gaussian Filter: Example

To convolve image with this kernel: $\frac{1}{115}$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

first convolve each row with: $\frac{1}{10.7}$

1.3	3.2	3.8	3.2	1.3

and then each column with: $\frac{1}{10.7}$

1.3	3.2	3.8	3.2	1.3

Gaussian Filter: Example

- Straightforward convolution with 5 by 5 kernel
  - 25 multiplications, 24 additions per pixel
- "Smart" convolution
  - 10 multiplications, 9 additions per pixel
- Savings are even larger for larger kernels
  - If kernel is n by n, straightforward convolution takes $O(n^2)$ time per pixel, "smart" convolution takes O(n) time per pixel

Median filters

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

1	2	25
3	24	22
20	21	23

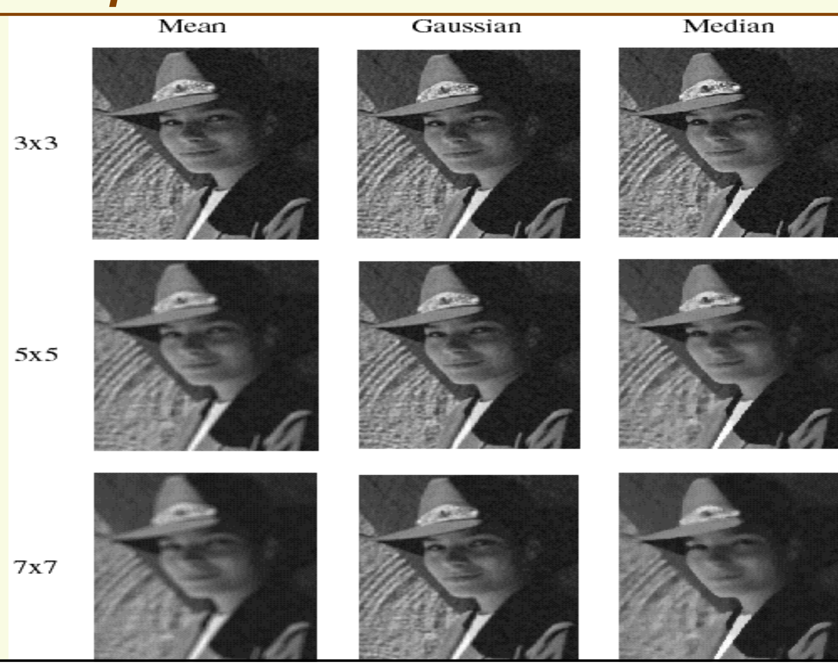
⇒

X	X	X
X	21	X
X	X	X

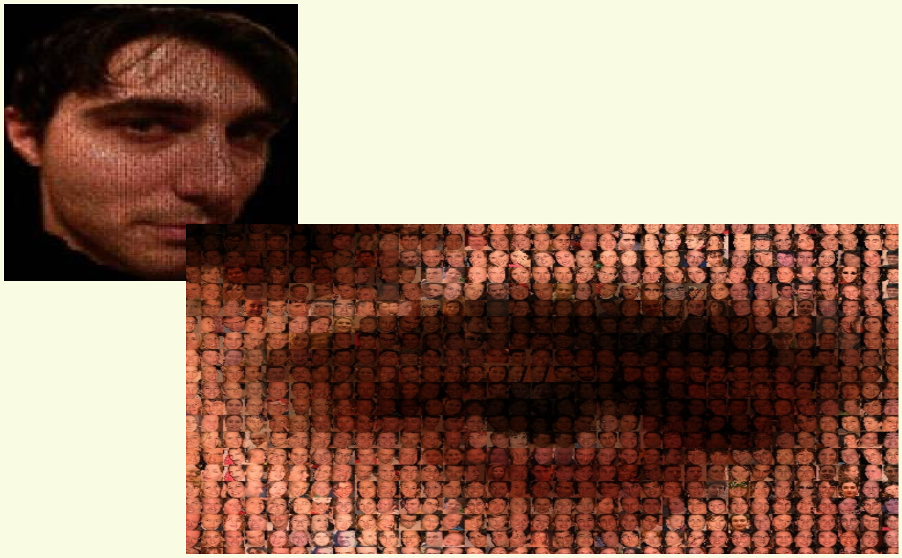
- A **Median Filter** operates over a window by selecting the median intensity in the window
- Median filter preserves sharp details better than mean filter, it is not so prone to oversmoothing
- Is a median filter a kind of convolution?

Comparison: salt and pepper noise

Comparison: Gaussian noise

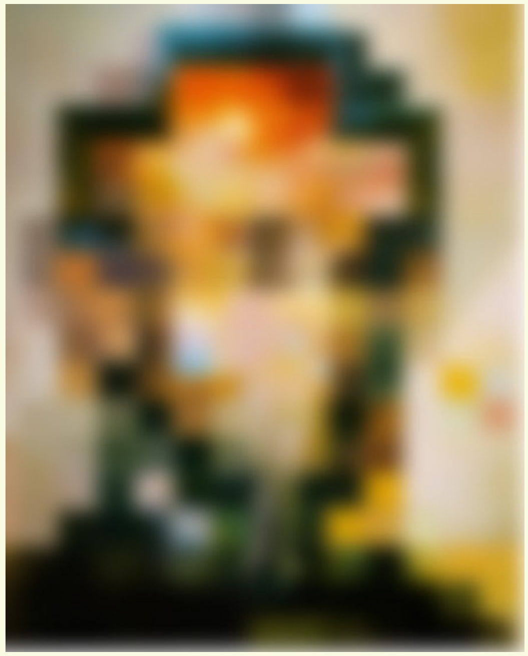


Alternative Filtering: Face of Faces



http://www.salle.url.edu/~ftorre/

Salvador Dali, *"Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln"*, 1976



Salvador Dali, "*Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln*", 1976