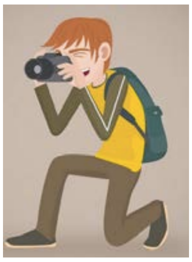
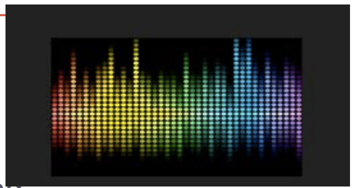
DIGITAL IMAGE PROCESSING





Lecture 2

Basics of Image Processing

Tammy Riklin Raviv
Electrical and Computer Engineering
Ben-Gurion University of the Negev

The Makeover of My First Image

The Makeover of My First Image

The Makeover of My First Image

The Makeover of My First Image

The Makeover of My First Image

The Makeover of My First Image

The Makeover of My First Image

The Makeover of My First Image

The Makeover of My First Image

The Makeover of My First Image

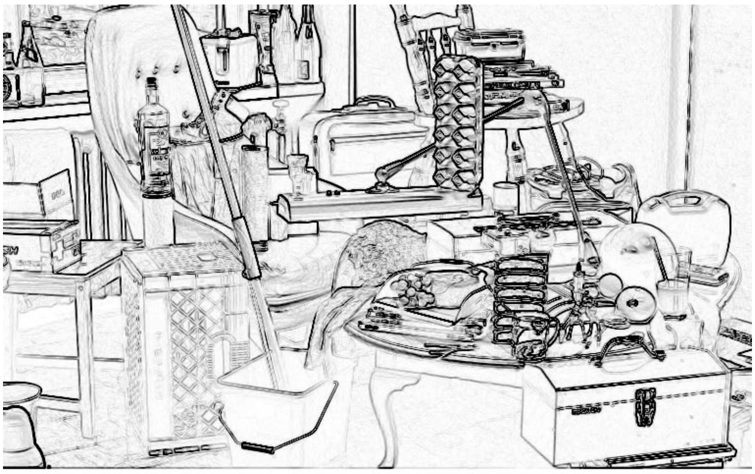


Image processing vs. Image detection

Image processing paradigm

image → filter → image

In contrast the image detection paradigm is

image → filter → features

for example, features might be edges, lines, points, faces, …
i.e. geometry or objects (but not an image)

This class

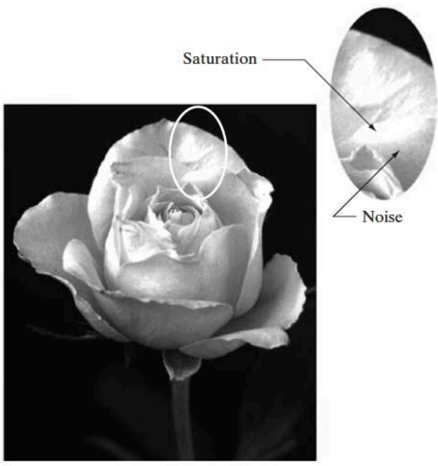
image → operation → image

- Image basics
  - Two types of quantization
- Point operations
  - Histogram equalization
  - Temporal averaging
- Spatial filters (linear)
  - Noise reduction, sharpening
  - Separability

Zisserman

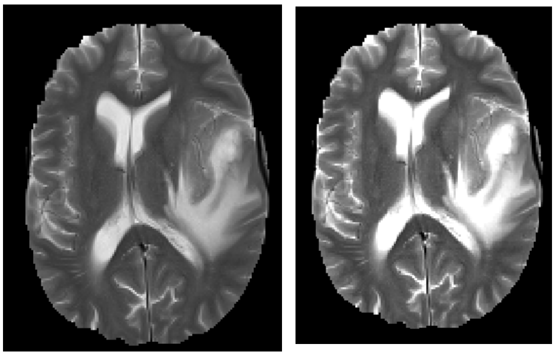
Image Processing – Why?

- Main Objective: Image Enhancement
- Why?
  - Further processing
  - Aesthetic



What is image enhancement?

- How to improve contrast ?



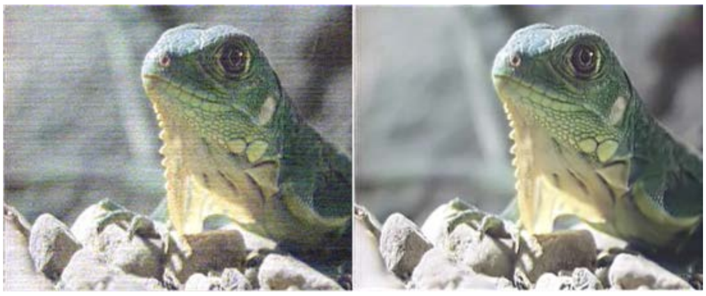
What is image enhancement?

- How to improve contrast ?
- How to sharpen edges?

Original image: http://www.rd.com/advice/pets/how-to-decode-your-cats-behavior/

What is image enhancement?

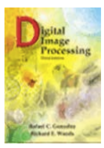
- How to improve contrast ?
- How to sharpen edges?
- How to reduce noise?



https://leegihan.wordpress.com/category/the-best-noise-reduction/

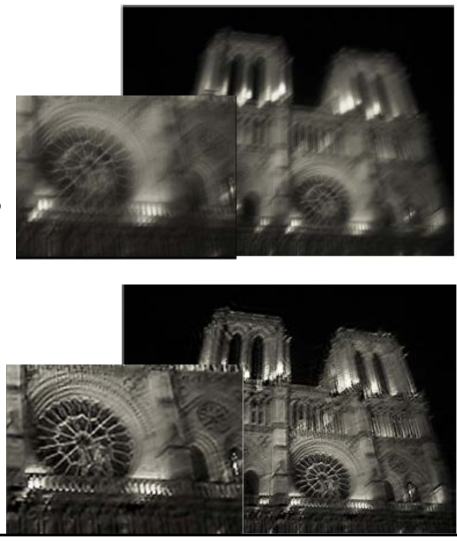
What is image enhancement?

- How to improve contrast ?
- How to sharpen edges?
- How to reduce noise?
- How to remove shadows ?



What is image enhancement?

- How to improve contrast ?
- How to sharpen edges?
- How to reduce noise?
- How to remove shadows ?
- How to do deblurring?



What is image enhancement?

- How to improve contrast ?
- How to sharpen edges?
- How to reduce noise?
- How to remove shadows ?
- How to do deblurring?
- How to do impainting?

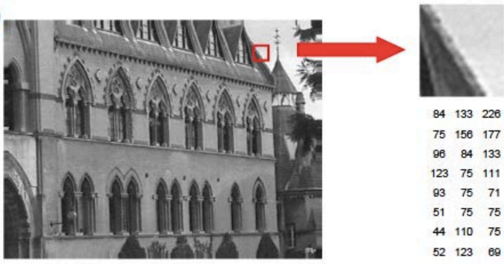
Sampling and Quantization

A monochrome image is an array of values.

There are two types of discretization involved:

1. Spatial sampling (pixels -‘picture elements’), and
2. Intensity quantization (grey level value).

(0,0)

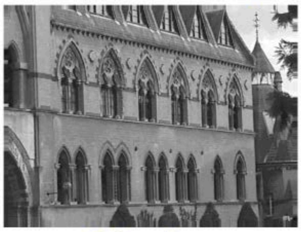


84	133	226	212	218	218	222	212	218	222	226	218
75	156	177	218	212	218	218	218	218	222	218	218
96	84	133	203	218	218	218	222	212	218	222	218
123	75	111	156	212	218	212	212	218	218	218	226
93	75	71	133	185	231	226	226	222	212	218	218
51	75	75	75	156	206	218	218	218	222	212	222
44	110	75	65	143	194	231	218	218	218	218	218
52	123	69	84	60	156	199	231	231	222	226	226
52	75	84	81	65	69	150	231	231	226	231	231
36	36	84	93	84	71	156	160	240	240	231	231
36	40	113	75	69	75	71	133	194	240	240	240
52	52	105	85	69	75	75	123	111	222	231	231

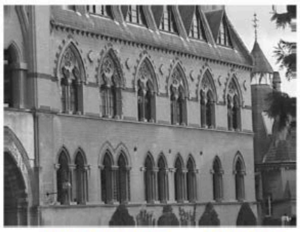
Intensity quantization

2 levels - binary

4 levels



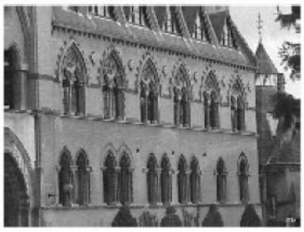
8 levels



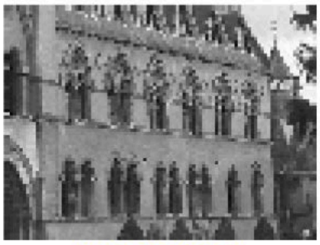
256 levels – 1 byte

Spatial sampling

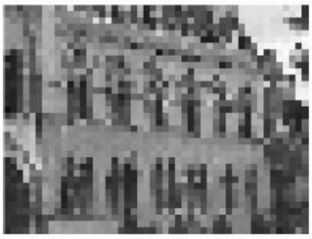
384 x 288 pixels



192 x 144 pixels



92 x 72 pixels



48 x 36 pixels

Some numbers

Quantization:

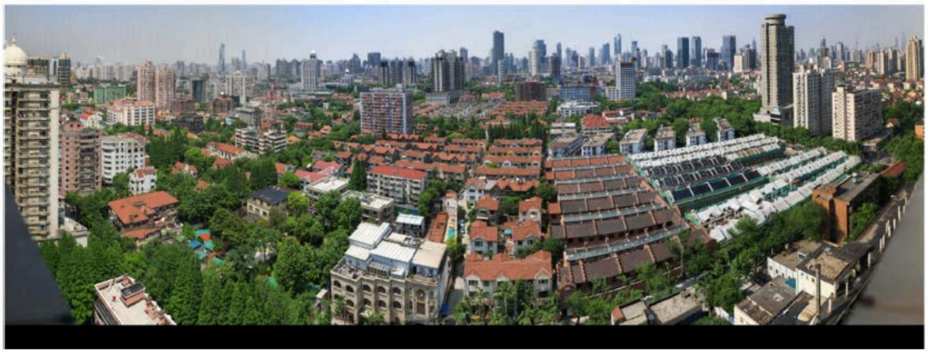
- Often 8 bits per pixel (0-255, 2^8 = 256 levels) for monochrome
- 24 bits per pixel for colour (8 bits for each of Red Green Blue)
- Medical images 12 bits (4096 levels) or 16 bits (65536 levels)

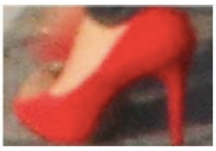
Size

- Cameras typically 4K x 3K pixels or far more
- Satellite images 10 -100K pixels width

- 3D images
  - e.g. Magnetic Resonance Images
  - Videos (2D + time)

Shanghai Skyline - Stitched from 12,000 photos





273 G pixels

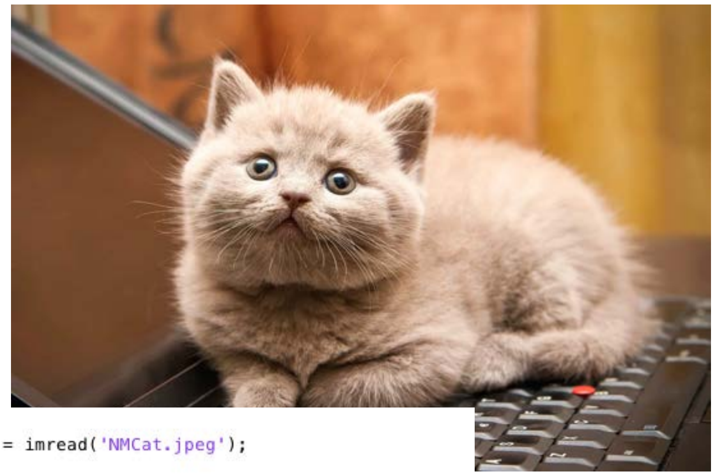
http://gigapan.com

Google Art Project

resolution 30,000 × 23,756 pixels

https://www.google.com/culturalinstitute/about/artproject/

Not-My-Cat Image



```
>>
>> Irgb = imread('NMCat.jpeg');
>> whos
  Name          Size                   Bytes  Class

  Irgb       1600x2400x3            11520000  uint8
```

Images as functions

- We can think of an **image** as a function, f, from $\mathbb{R}^2 \to \mathbb{R}$:
  - $f(x, y)$ gives the **intensity** at position (x, y)
  - Realistically, we expect the image only to be defined over a rectangle, with a finite range:

$$f: [a, b] \times [c, d] \to [0, 1]$$

- A color image is just three functions pasted together. We can write this as a "vector-valued" function:

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

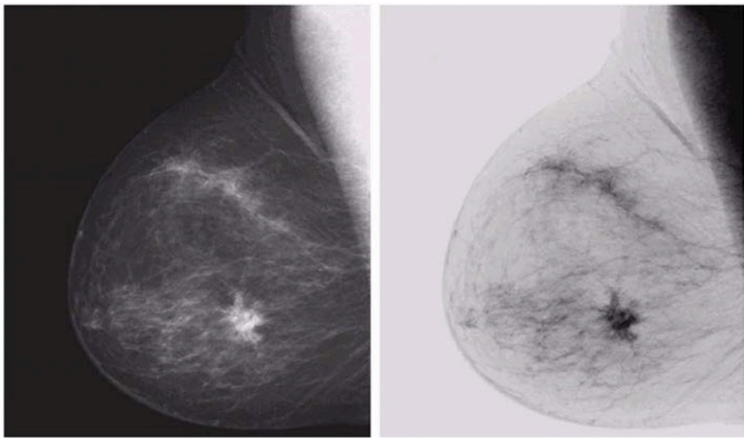
Point (Pixelwise) Operation

$$i_{\mathrm{n}} = \mathrm{function}(i)$$

new intensity

original intensity

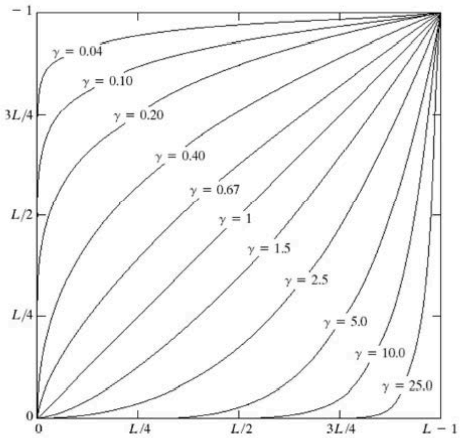
Pixelwise operation: Negative



Pixelwise operation: power low transformation

$$i_n = i^{\gamma}$$

i_n

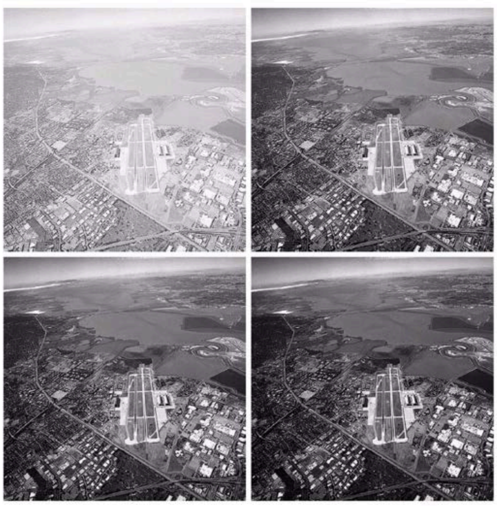


i

Image Enhancement

original

$\gamma = 3$

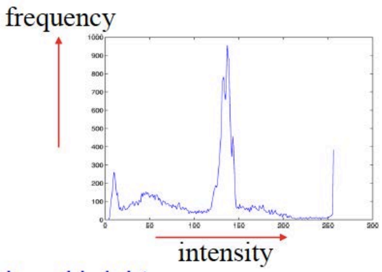


$\gamma = 4$

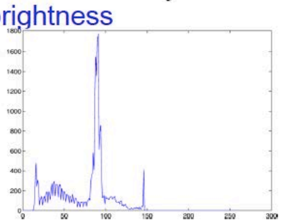
$\gamma = 5$

Point operation: Histograms

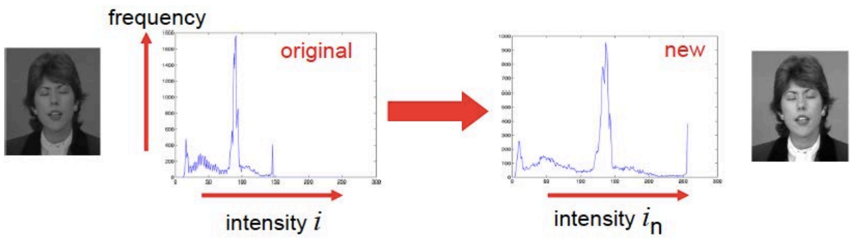
- image intensity histogram



- histogram for image with reduced brightness

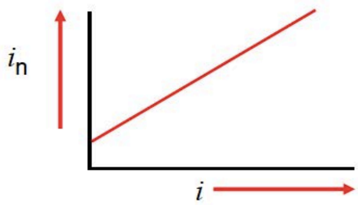


Point Operation: Intensity Map

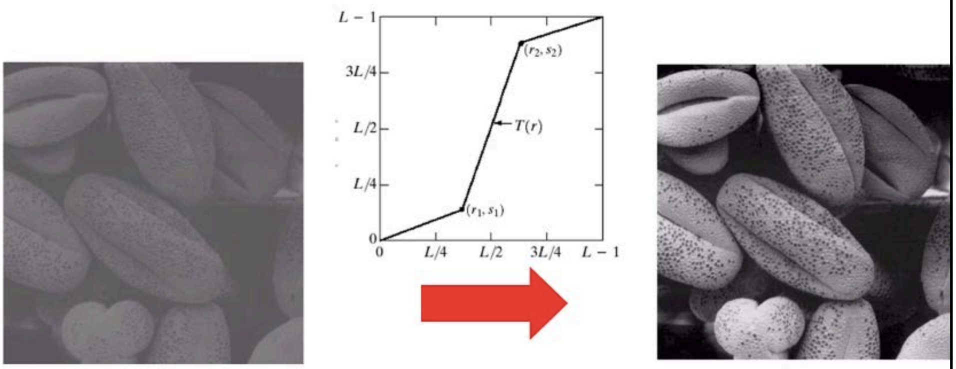


- change contrast and brightness by transforming pixel intensities
- assign the same new intensity value to all pixels having a given original value
- e.g. an affine (linear) map:

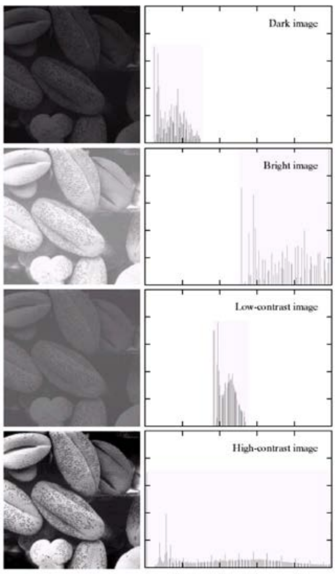
$$i_n = \alpha i + \beta$$

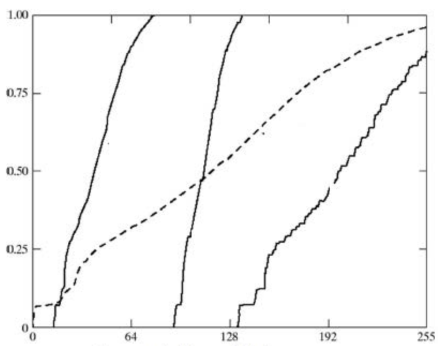


Contrast Stretching



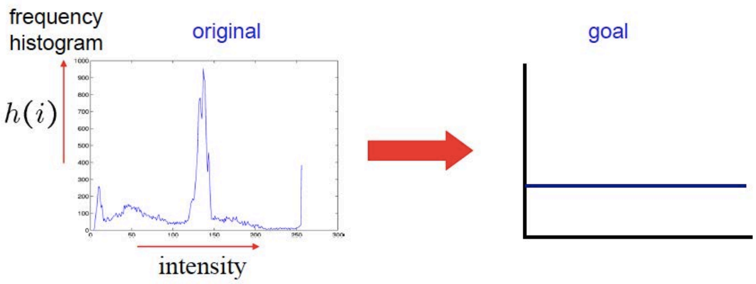
Cumulative Histograms



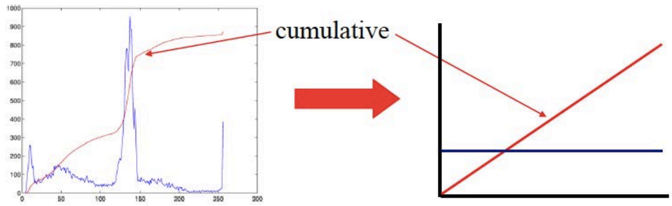


Cumulative Histograms

Histogram Equalization

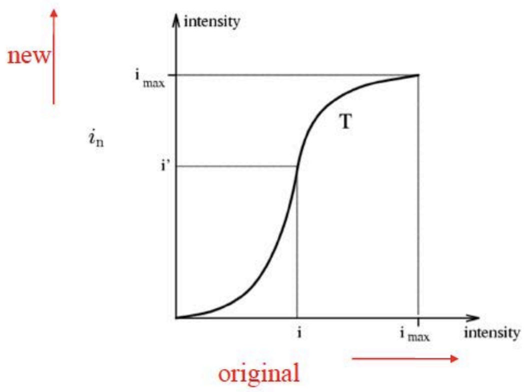


- use cumulative distribution $C(i) = \int_0^i h(u)\, du$ as a measure



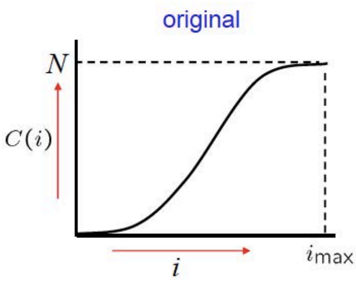
Histogram Equalization

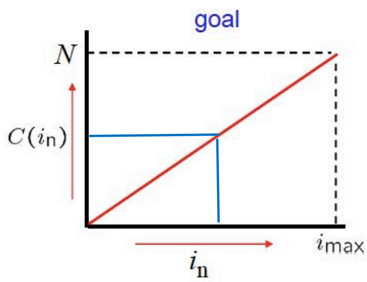
Objective: apply a monotonic map *T(i)* to the intensities so that the intensity histogram is less peaked (flattened)



Histogram Equalization (flatening)

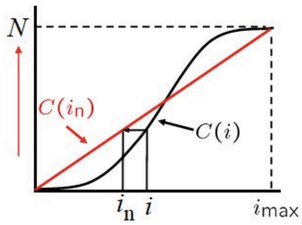
$$C(i) = \int_0^i h(u)\,du \qquad C(i_{\text{max}}) = \#\text{pixels} = N$$





$$C(i_n) = \frac{N}{i_{\text{max}}} i_n$$

Histogram Equalization



$$C(i) = C(i_n) = \frac{N}{i_{\text{max}}} i_n$$

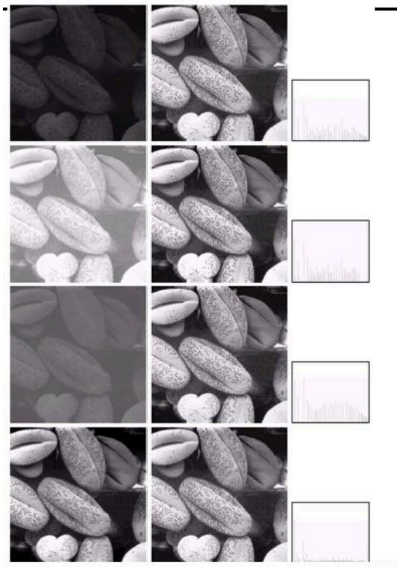
$$i_n = \frac{i_{\text{max}}}{N} C(i)$$

Algorithm:

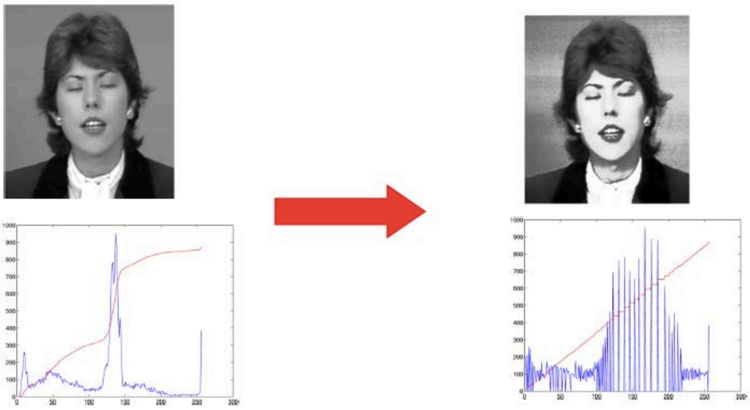
- compute the cumulative probability distribution $C(i)$ from the intensity histogram
- map pixel intensities as

$$i_n = T(i) \quad \text{where} \quad T(i) = \frac{i_{\text{max}}}{N} C(i)$$

Histogram Equalization

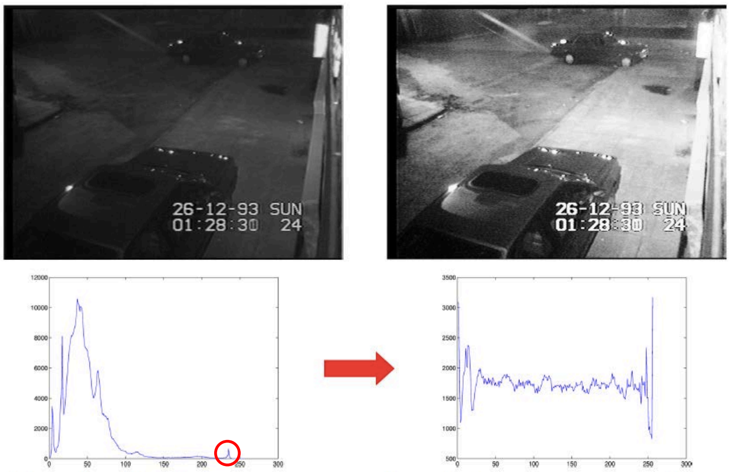


Histogram Equalization



- some values which are quite close together are spread out so that minor variations are more visible
- some values which were originally different are reassigned to the same value, so information is lost
- note, histogram is not flat

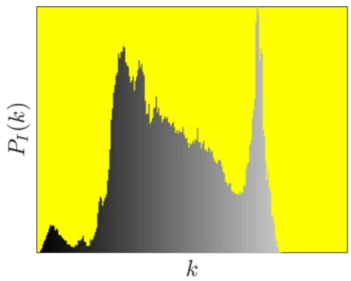
Histogram Equalization



• intervals without many corresponding pixels are compressed

Normalized Intensity Histogram

$I : \Omega \rightarrow \{0, \ldots, K-1\}$

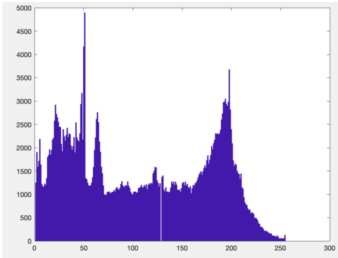


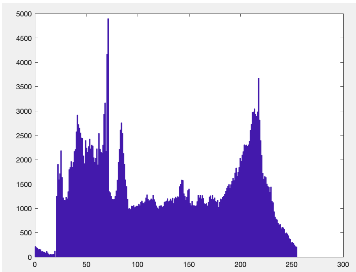
Natural image: $\mathbf{x} \in \Omega,\ I(\mathbf{x}) = k,\ k \in \{0, 1, \ldots, K-1\}$

$$P_I(k) \triangleq Pr(I(\mathbf{x}) = k)$$

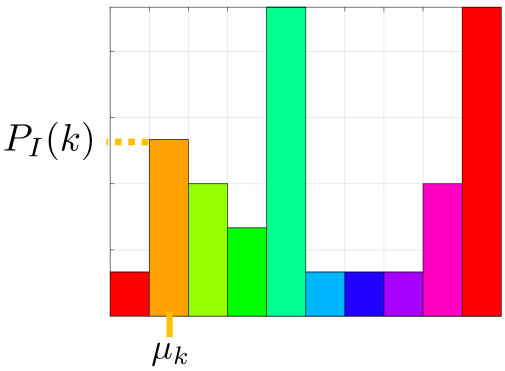
Mor Avi-Aharon et al

Difference between histograms





Histogram Representation

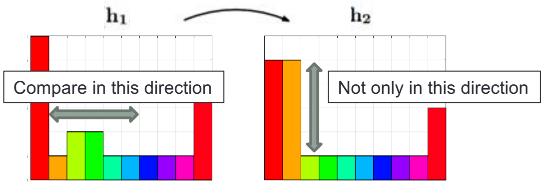


$$\mathbf{h} = \{\mu_k, P_I(k)\}_{k=0}^{K-1}$$

$$P_I(k) = \frac{1}{N}\sum_{\mathbf{x}\in\Omega}\Pi_k(I(\mathbf{x}))$$

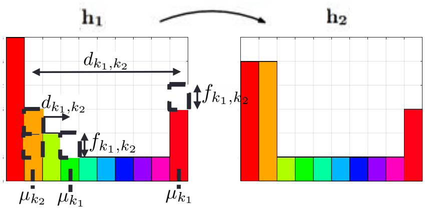
Mor Avi-Aharon et al

Earth Mover's Distance (Wasserstein Distance)



Mor Avi-Aharon et al

Earth Mover's Distance (Wasserstein Distance)



$$\mathcal{W}(\mathbf{h_1}, \mathbf{h_2}, F) = \sum_{k_1=0}^{K-1} \sum_{k_2=0}^{K-1} d_{k_1,k_2} f_{k_1,k_2}$$

EMD constraints:

$$f_{k_1,k_2} \geq 0 \text{ for each } k_1, k_2$$

$$\sum_{k_2=0}^{K-1} f_{k_1,k_2} \leq P_{I_1}(k_1) \text{ for all } k_1$$

$$\sum_{k_1=0}^{K-1} f_{k_1,k_2} \leq P_{I_2}(k_2) \text{ for all } k_2$$

$$\sum_{k_1=0}^{K-1} \sum_{k_2=0}^{K-1} f_{k_1,k_2} = \min\left\{\sum_{k_1=0}^{K-1} P_{I_1}(k_1), \sum_{k_2=0}^{K-1} P_{I_2}(k_2)\right\}$$

$\mathbf{D}$ – ground distance matrix
d_{k_1,k_2} distance between μ_{k_1}, μ_{k_2}

$\mathbf{F}$ – flow matrix
f_{k_1,k_2} mass transported between μ_{k_1}, μ_{k_2}

Rubner, Tomasi, and Guibas, ICCV, 1998.

Mor Avi-Aharon et al

Earth Mover's Distance (Wasserstein Distance)

- The EMD between $\mathbf{h}_1, \mathbf{h}_2$ is the **minimum cost** of work that satisfies the **constraints** normalized by the total flow:

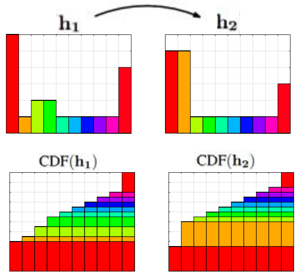
$$\mathcal{D}_{\text{EMD}}(\mathbf{h_1}, \mathbf{h_2}) = \inf_{\mathbf{F}} \frac{W(\mathbf{h_1}, \mathbf{h_2}, \mathbf{F})}{\sum_{k_1=0}^{K-1} \sum_{k_2=0}^{K-1} f_{k_1,k_2}}$$

- $\boxed{\sum_{k_1=0}^{K-1} P_{I_1}(k_1) = \sum_{k_2=0}^{K-1} P_{I_2}}$ $\boxed{\mathbf{h}_1, \mathbf{h}_2 \in \mathbb{R}^{1\times K}}$

  → EMD is equivalent to Mallows distance:

$$\mathcal{D}_{\text{EMD}}(\mathbf{h_1}, \mathbf{h_2}) = \left(\frac{1}{K}\right)^{\frac{1}{l}} \|\text{CDF}(\mathbf{h_1}) - \text{CDF}(\mathbf{h_2})\|_l$$

  where, CDF(·) is the cumulative density function.



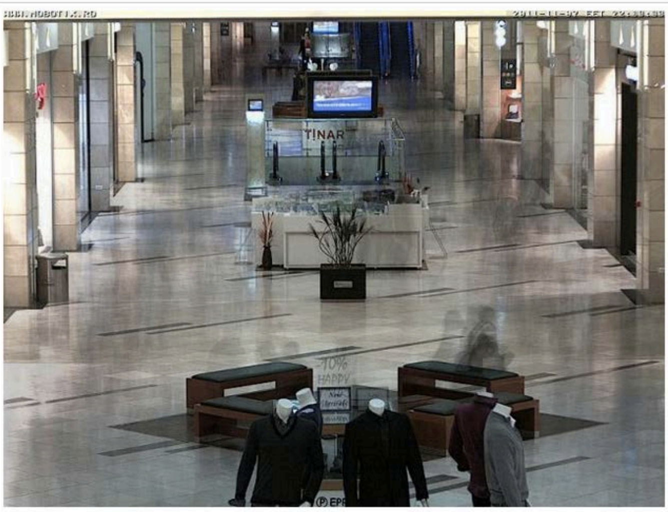
Levina and Bickel. ICCV 2001.

Mor Avi-Aharon et al

Temporal Averaging

Temporal Averaging

Temporal Averaging



Temporally Averaged from 70 Images

Spatial Operation

$$i_n(x, y) = \text{function}(i \in \text{spatial neighbourhood}(x, y))$$

new intensity

original intensity

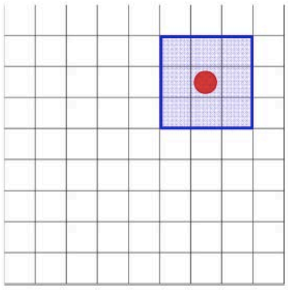


Image Filtering

- Image filtering: compute function of local neighborhood at each position
- Linear filtering: function is a weighted sum/difference of pixel values
- Many applications:
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching

Image filtering

- Compute function of local neighborhood at each position

h=output f=filter I=image

$$h[m,n]=\sum_{k,l} f[k,l]\, I[m+k,n+l]$$

James Hays

Image filtering

Example: box filter

$$f[\cdot\,,\cdot\,]$$

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

Slide credit: David Lowe (UBC)

Image Filtering

$f[\cdot,\cdot]\,\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$I[.,.]$

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

$h[.,.]$

$$h[m,n]=\sum_{k,l} f[k,l]\,I[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$f[\cdot,\cdot]\,\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$I[.,.]$

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

$h[.,.]$

	0	10							

$$h[m,n]=\sum_{k,l} f[k,l]\,I[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$f[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$I[.,.]$

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

$h[.,.]$

	0	10	20						

$$h[m,n]=\sum_{k,l} f[k,l]\, I[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$f[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$I[.,.]$

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

$h[.,.]$

	0	10	20	30					

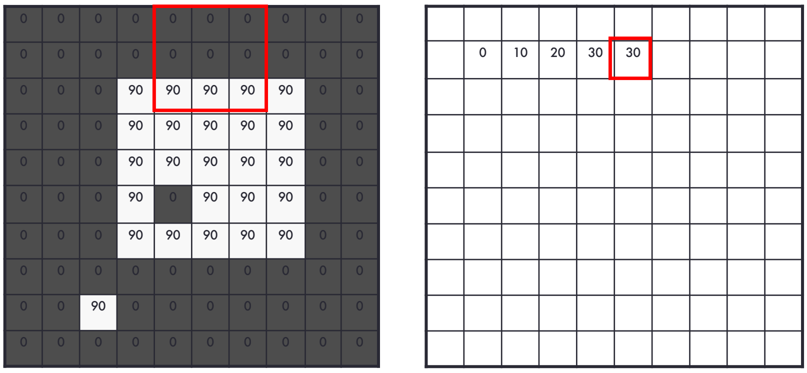
$$h[m,n]=\sum_{k,l} f[k,l]\, I[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$f[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$

$I[.,.]$ $h[.,.]$



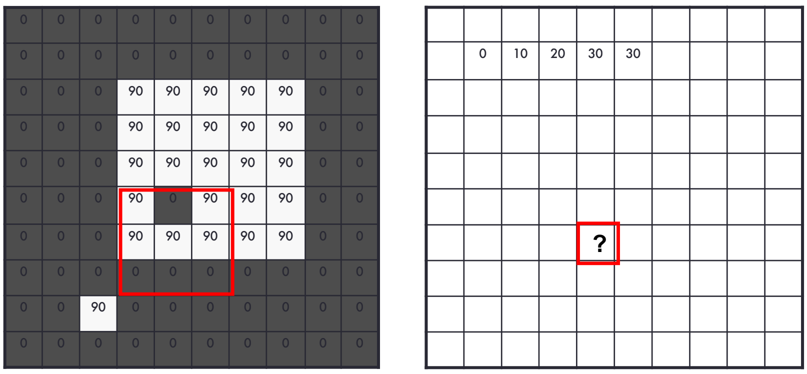
$$h[m,n]=\sum_{k,l} f[k,l]\, I[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$f[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$

$I[.,.]$ $h[.,.]$



$$h[m,n]=\sum_{k,l} f[k,l]\, I[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$f[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$I[.,.]$

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

$h[.,.]$

	0	10	20	30	30				
						?			
				50					

$$h[m,n]=\sum_{k,l} f[k,l]\, I[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$f[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$I[.,.]$

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

$h[.,.]$

	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	

$$h[m,n]=\sum_{k,l} f[k,l]\, I[m+k,n+l]$$

Credit: S. Seitz

Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

$$f[\cdot\,,\cdot\,]$$

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

Slide credit: David Lowe (UBC)

Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)
- Why does it sum to one?

$$f[\cdot\,,\cdot\,]$$

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

Slide credit: David Lowe (UBC)

James Hays

Smoothing with box filter

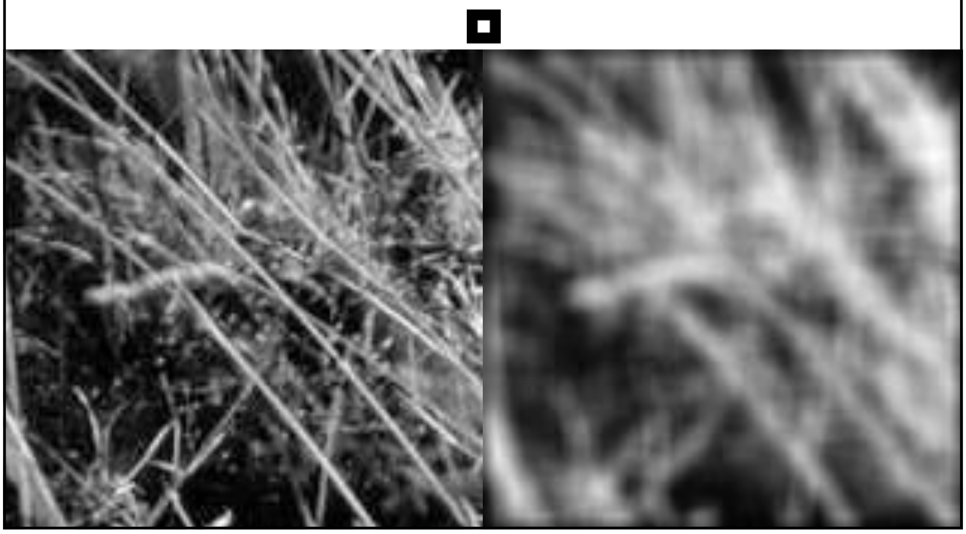


Image filtering

- Compute function of local neighborhood at each position

$$h[m,n]=\sum_{k,l} f[k,l]\, I[m+k,n+l]$$

- Really important!
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching

James Hays

Linear Filters

1. $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

2. $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$

3. $\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$

4. $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

1. Practice with linear filters

Original

$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

?

Source: D. Lowe

1. Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Source: D. Lowe

2. Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

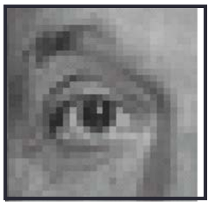
?

Source: D. Lowe

2. Practice with linear filters

Original

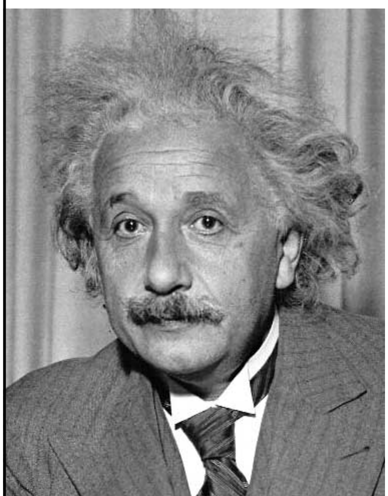
0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

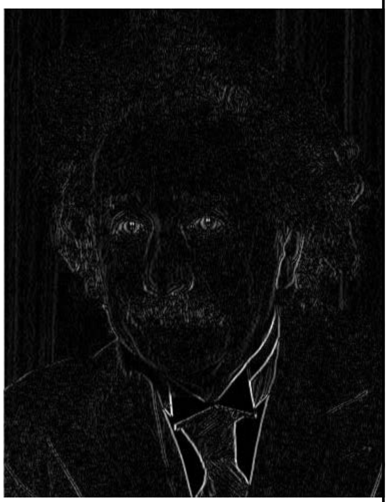
Source: D. Lowe

3. Practice with linear filters



1	0	-1
2	0	-2
1	0	-1

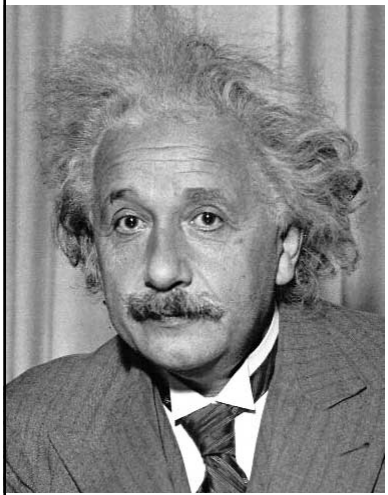
Sobel



Vertical Edge
(absolute value)

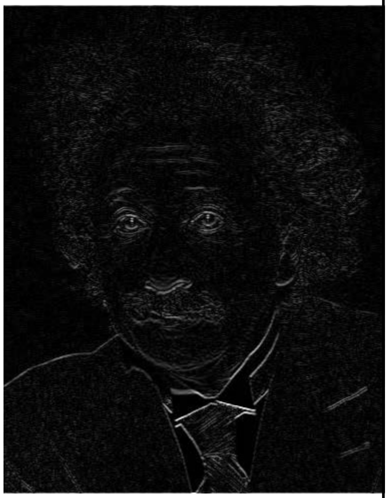
David Lowe

3. Practice with linear filters



1	2	1
0	0	0
-1	-2	-1

Sobel



Horizontal Edge
(absolute value)

David Lowe

4. Practice with linear filters

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

?

(Note that filter sums to 1)

Source: D. Lowe

4. Practice with linear filters

Original

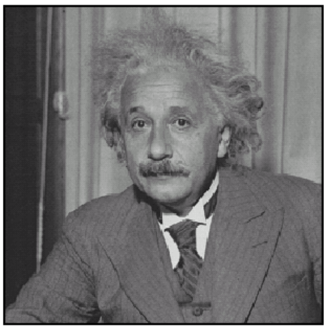
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

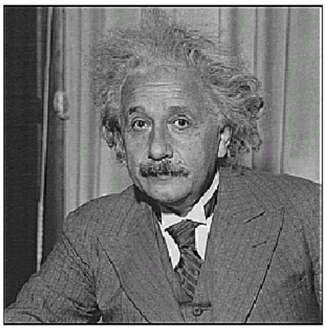
- Accentuates differences with local average

Source: D. Lowe

4. Practice with linear filters



before



after

Sharpening

Source: D. Lowe

Filtering: Correlation vs. Convolution

- 2d correlation

`h=filter2(f,I);` or `h=imfilter(I,f);`

$$h[m,n] = \sum_{k,l} f[k,l]\, I[m+k, n+l]$$

James Hays

Filtering: Correlation vs. Convolution

- 2d correlation

`h=filter2(f,I);` or `h=imfilter(I,f);`

$$h[m,n] = \sum_{k,l} f[k,l]\, I[m+k, n+l]$$

- 2d convolution

`h=conv2(f,I);`

$$h[m,n] = \sum_{k,l} f[k,l]\, I[m-k, n-l]$$

`conv2(I,f)` is the same as `filter2(rot90(f,2),I)`

Correlation and convolution are identical when the filter is symmetric.

James Hays

Key properties of linear filters

Linearity:

```
imfilter(I, f1 + f2) =
   imfilter(I,f1) + imfilter(I,f2)
```

Shift invariance: same behavior regardless of pixel location

```
imfilter(I,shift(f)) = shift(imfilter(I,f))
```

Any linear, shift-invariant operator can be represented as a convolution

Source: S. Lazebnik

Convolution properties

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
  - But particular filtering implementations might break this equality, e.g., image edges
- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$
  - Correlation is _not_ associative (rotation effect)
  - Why important?

Source: S. Lazebnik

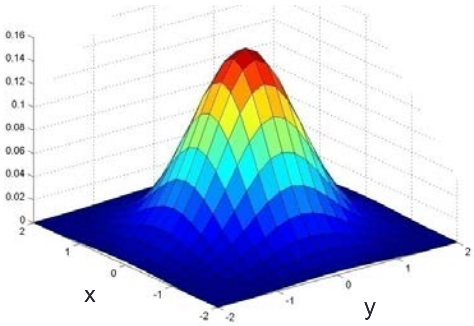
Convolution properties

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
  - But particular filtering implementations might break this equality, e.g., image edges
- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$
  - Correlation is _not_ associative (rotation effect)
  - Why important?
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k (a * b)$
- Identity: unit impulse $e = [0, 0, 1, 0, 0]$, $a * e = a$

Source: S. Lazebnik

Important filter: Gaussian

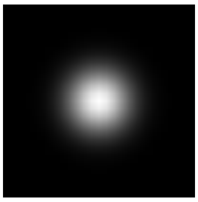
- Weight contributions of neighboring pixels by nearness



x

y					
	0.003	0.013	0.022	0.013	0.003
	0.013	0.059	0.097	0.059	0.013
y	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, $\sigma = 1$

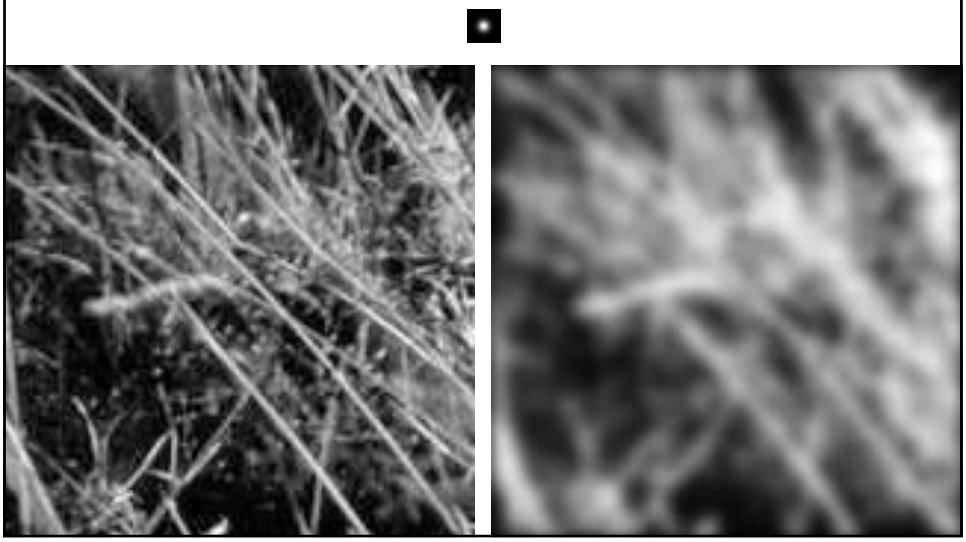


$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Slide credit: Christopher Rasmussen

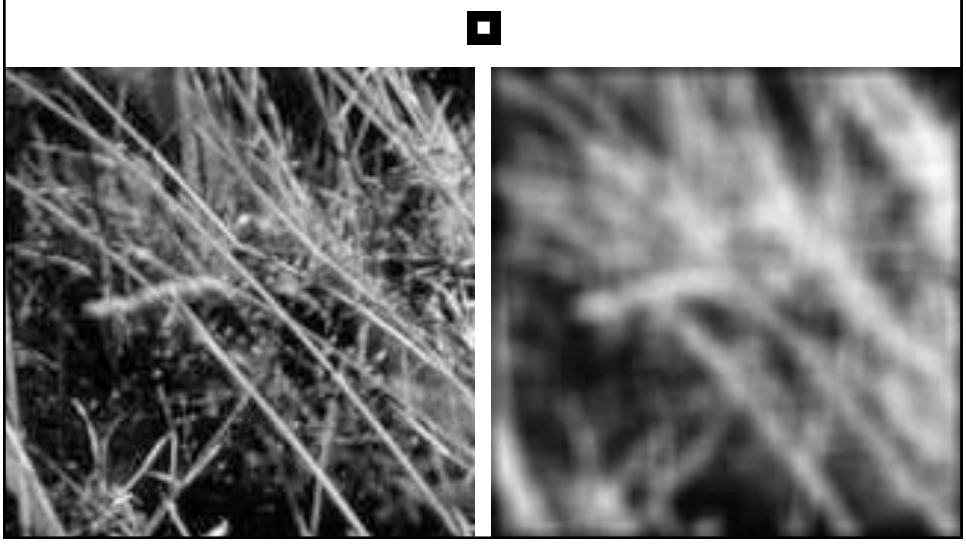
James Hays

Smoothing with Gaussian filter



James Hay

Smoothing with box filter



Gaussian filters

- Remove "high-frequency" components from the image (low-pass filter)
  - Images become more smooth
- Convolution with self is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians

Source: K. Grauman

Separability of the Gaussian filter

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

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

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

Source: D. Lowe

Separability example

2D convolution (center location only)

$$\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} * \begin{bmatrix}2&3&3\\3&5&5\\4&4&6\end{bmatrix}$$

65

The filter factors into a product of 1D filters:

$$\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} = \begin{bmatrix}1\\2\\1\end{bmatrix} \times \begin{bmatrix}1&2&1\end{bmatrix}$$

Perform convolution along rows:

$$\begin{bmatrix}1&2&1\end{bmatrix} * \begin{bmatrix}2&3&3\\3&5&5\\4&4&6\end{bmatrix} = \begin{bmatrix}\ &11&\ \\\ &18&\ \\\ &18&\ \end{bmatrix}$$

Followed by convolution along the remaining column:

$$\begin{bmatrix}1\\2\\1\end{bmatrix} * \begin{bmatrix}\ &11&\ \\\ &18&\ \\\ &18&\ \end{bmatrix} = \begin{bmatrix}\ &\ &\ \\\ &65&\ \\\ &\ &\ \end{bmatrix}$$

Source: K. Grauman

Separability

- Why is separability useful in practice?

Separability

- Why is separability useful in practice?
- If K is width of convolution kernel:
  - 2D convolution = K^2 multiply-add operations
  - 2x 1D convolution: 2K multiply-add operations

Practical matters

How big should the filter be?

- Values at edges should be near zero
- Gaussians have infinite extent…
- Rule of thumb for Gaussian: set filter half-width to about 3 σ

James Hays

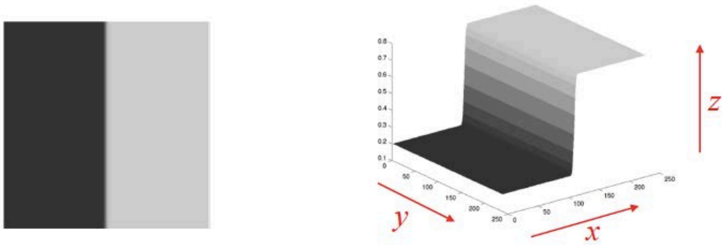
Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

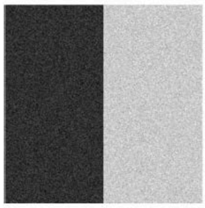
Computing Derivatives using Linear Filters

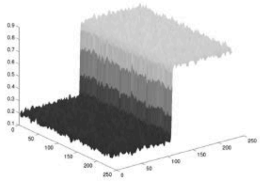
Think of the image as a surface with $z = f(x,y)$



Computing Derivatives using Linear Filters

Objective: compute gradient $\nabla f(x,y) = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})$ of the image "surface"





- e.g. as a method to find the edge in the image
- there is the problem of noise ….

Computing Derivatives using Linear Filters

1D

Want $f'(x)$, use central difference

$$f'(i) = \frac{f_{i+1} - f_{i-1}}{2}$$

which is equivalent to the molecule $[-\frac{1}{2}, 0, \frac{1}{2}]$.

For a noisy signal the derivative amplifies the noise.

Solution?

Computing Derivatives using Linear Filters

1D

Want $f'(x)$, use central difference

$$f'(i) = \frac{f_{i+1} - f_{i-1}}{2}$$

which is equivalent to the molecule $[-\frac{1}{2}, 0, \frac{1}{2}]$.

For a noisy signal the derivative amplifies the noise.

Solution

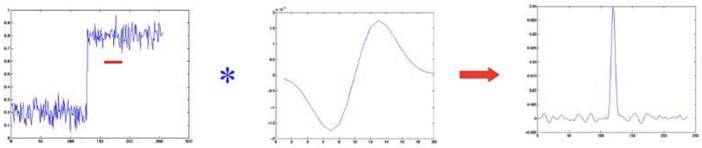
- smooth with a Gaussian filter
- then differentiate

Computing Derivatives using Linear Filters

Differentiate smoothed signal $G(x) * f(x)$

$$\begin{aligned}\frac{d(G(x) * f(x))}{dx} &= G(x) * \frac{df(x)}{dx} \\ &= \frac{dG(x)}{dx} * f(x) \\ &= \left(\frac{-x}{\sqrt{2\pi}\sigma^3} e^{-x^2/2\sigma^2}\right) * f(x)\end{aligned}$$

Convolution with a derivative of Gaussian filter



e.g. for $\sigma = 1$ the molecule is $[-0.0133, -0.1080, -0.2420, 0.0000, 0.2420, 0.1080, 0.0133]$.

Computing Derivatives using Linear Filters

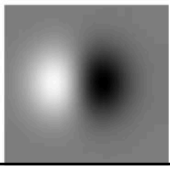
2D

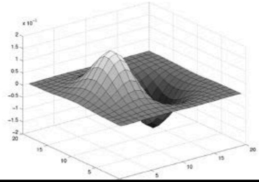
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-r^2/2\sigma^2}$$

$$\frac{\partial}{\partial x} G(x,y) * f(x,y) = \left(\frac{-x}{2\pi\sigma^4} e^{-(x^2+y^2)/2\sigma^2} \right) * f(x,y)$$

$$= \left(\frac{-x}{\sqrt{2\pi}\sigma^3} e^{-x^2/2\sigma^2} \right) \times \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-y^2/2\sigma^2} \right) * f(x,y)$$

Filtering with a 1D derivative of Gaussian filter in x and a 1D Gaussian filter in y – it is a separable filter

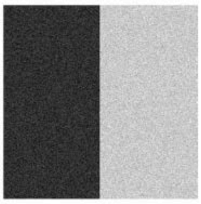




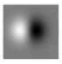
Computing Derivatives using Linear Filters

Example

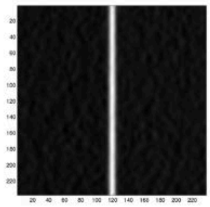
filter with x and y derivatives of Gaussian to obtain directional image derivatives



$*$

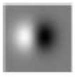


$$I_x = G_x(x,y) * I(x,y)$$

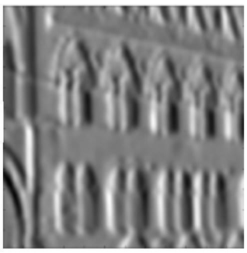


Computing Derivatives using Linear Filters

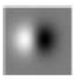
x-deriv

* 

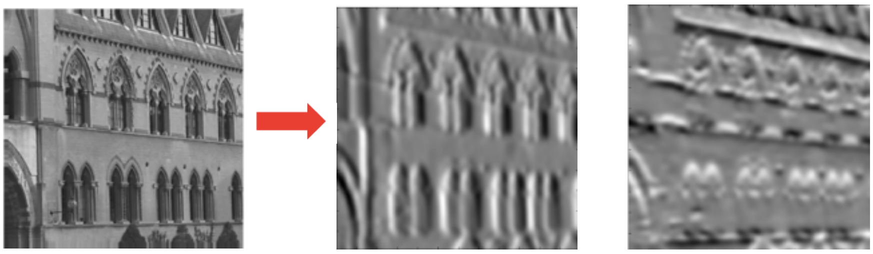
Computing Derivatives using Linear Filters



x-deriv

* 

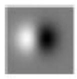
Computing Derivatives using Linear Filters



x-deriv

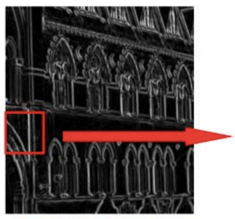
y-deriv

$*$



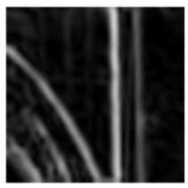
Computing Derivatives using Linear Filters

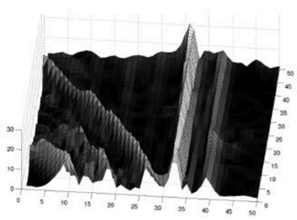
image



gradient magnitude

$$|\nabla(G(x,y) * I(x,y))|$$





Filtering: summary

Linear filtering is a weighted sum/difference of pixel values

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

- Can smooth, sharpen, translate (among many other uses)
- Filtering in Matlab, e.g. to filter image f with h

```
g = filter2( h, f );
```

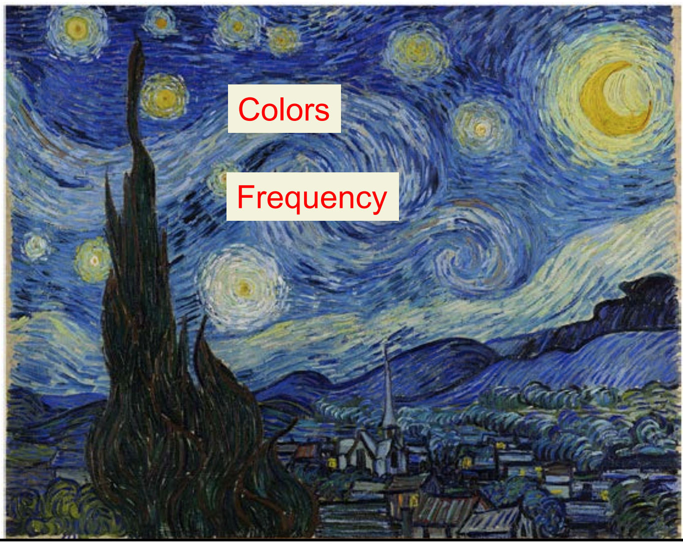
h=filter f=image

e.g. h = fspecial('gaussian');

Bonus Question: Image Enhancement

- Take an image (any image, but preferably one's that needs enhancement) and enhance it.
- Use what learned in this class to do so
- Plot the "before" and "after"
- Plot its derivatives before and after
- Matlab code is needed
- 3 Best works in class get 1 bonus point

Next Class



Links to Some Last Year Projects

- Supermarket
- AirDrums
- BallBounce
- PizzaPlanner
- VirtualShooting