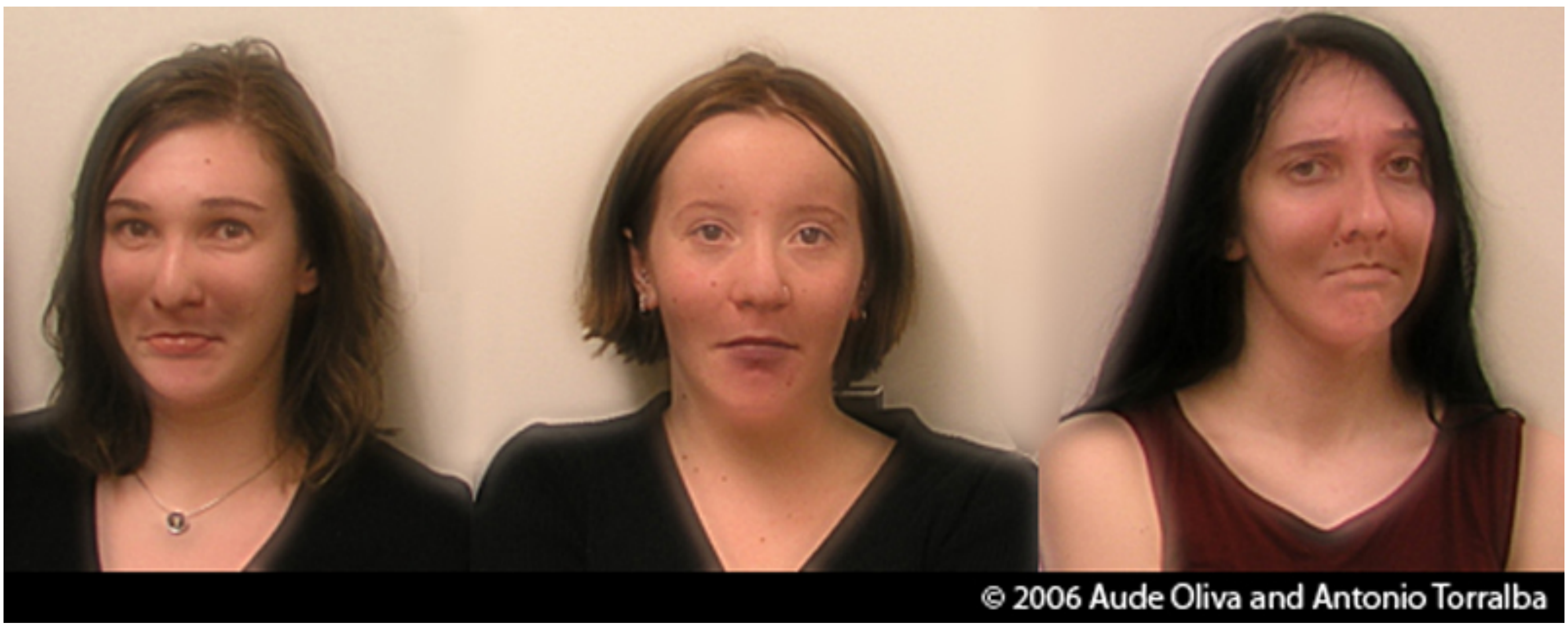
CS4670/5670: Intro to Computer Vision

Kavita Bala

Lecture 1: Images and image filtering

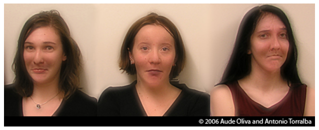


Hybrid Images, Oliva et al., http://cvcl.mit.edu/hybridimage.htm

CS4670: Computer Vision

Kavita Bala

Lecture 1: Images and image filtering

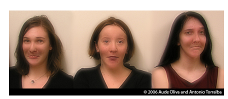


Hybrid Images, Oliva et al., http://cvcl.mit.edu/hybridimage.htm

CS4670: Computer Vision

Kavita Bala

Lecture 1: Images and image filtering

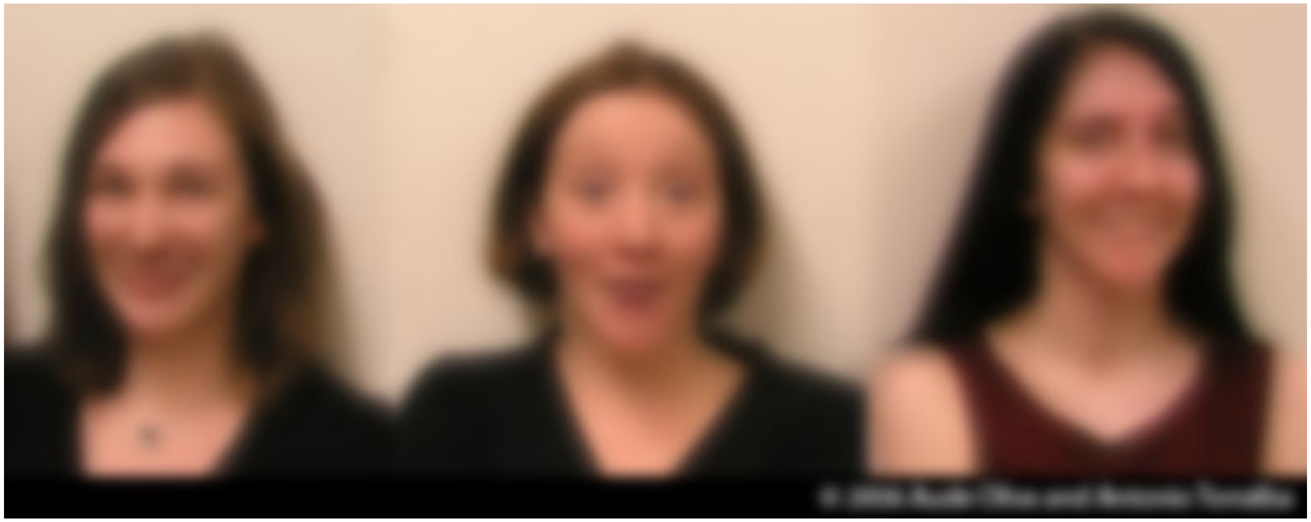


Hybrid Images, Oliva et al., http://cvcl.mit.edu/hybridimage.htm

CS4670: Computer Vision

Kavita Bala

Lecture 1: Images and image filtering



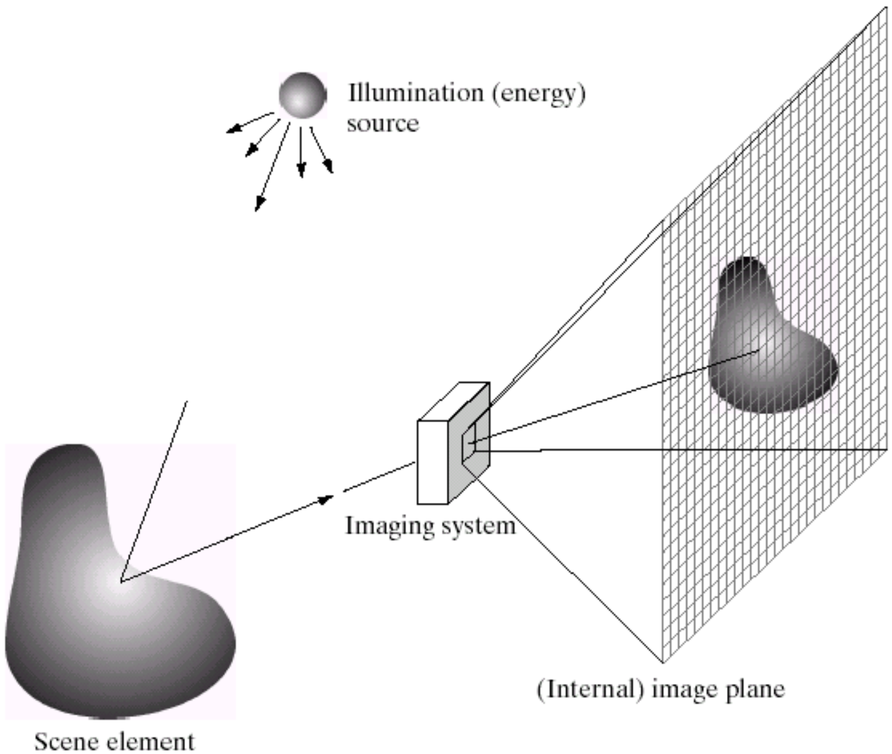
Hybrid Images, Oliva et al., http://cvcl.mit.edu/hybridimage.htm

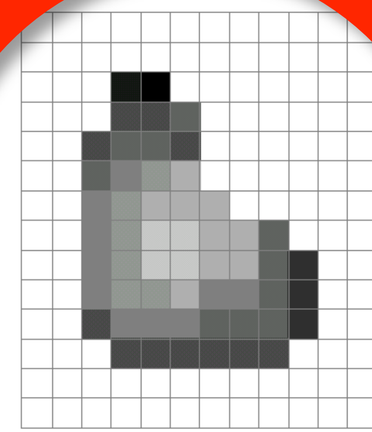
Reading and Announcements

- Szeliski, Chapter 3.1-3.2
- For CMS, wait and then contact Randy Hess (rbhess@cs.cornell.edu)

What is an image?

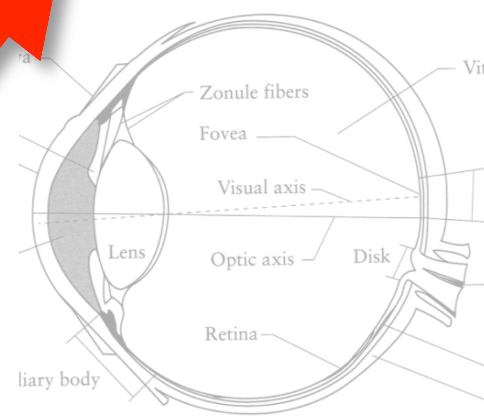
What is an image?





Digital Camera

We'll focus on these in this class



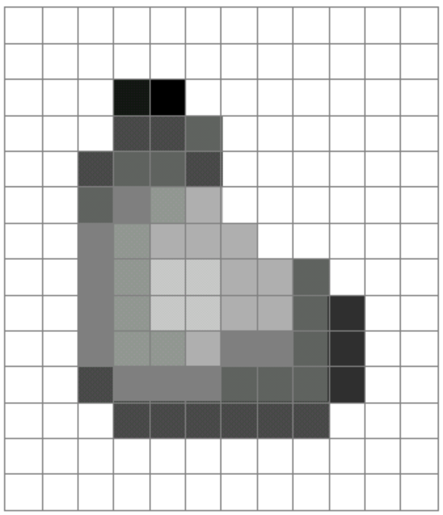
The Eye

(More on this process later)

Source: A. Efros

What is an image?

- A grid (matrix) of intensity values



=

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

(common to use one byte per value: 0 = black, 255 = white)

Images as functions

- An image contains discrete numbers of pixels

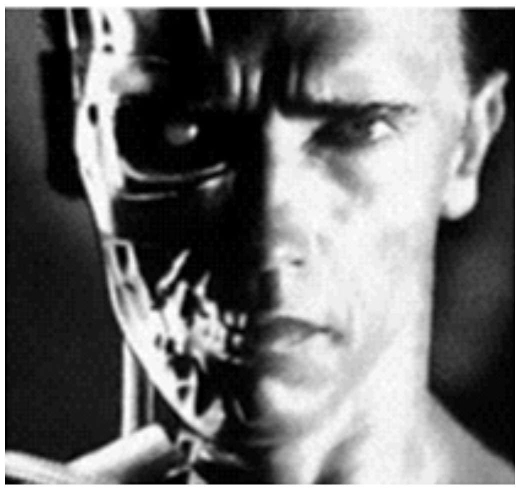
- Pixel value
  - grayscale/intensity
    - [0,255]
  - Color
    - RGB [R, G, B], where [0,255] per channel
    - Lab [L, a, b]: Lightness, a and b are color-opponent dimensions
    - HSV [H, S, V]: Hue, saturation, value

Images as functions

- Can think of image as a **function**, f, from R^2 to R or R^M:
  - Grayscale: $f(x,y)$ gives **intensity** at position (x,y)
    - f: [a,b] x [c,d] ➔ [0,255]
  - Color: $f(x,y) = [r(x,y), g(x,y), b(x,y)]$

What is an image?

A **digital** image is a discrete (**sampled**, **quantized**) version of this function



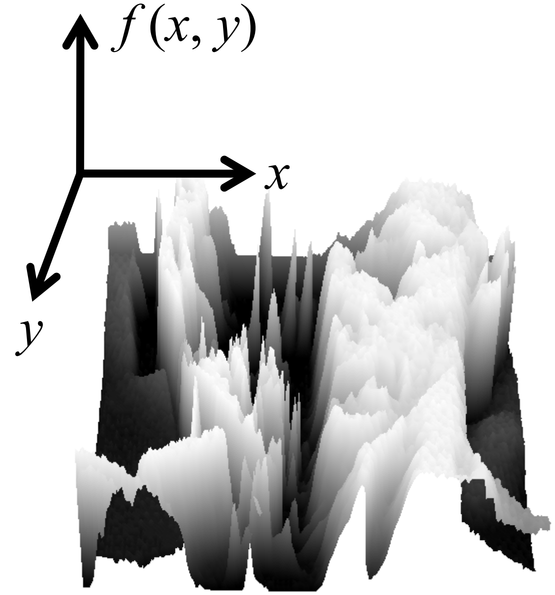
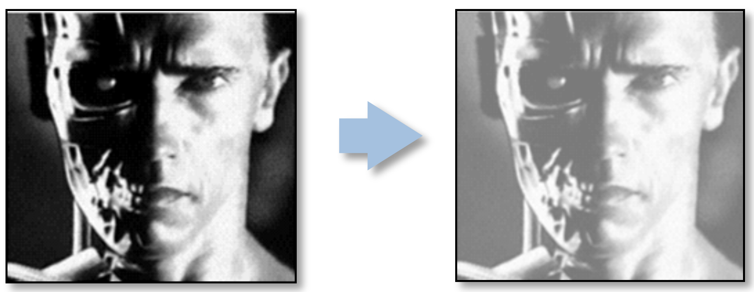
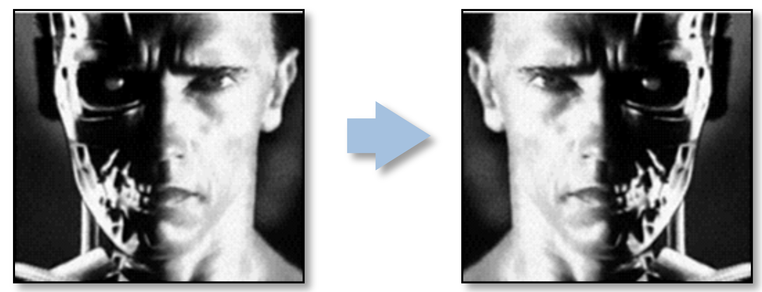


Image transformations

- As with any function, we can apply operators to an image



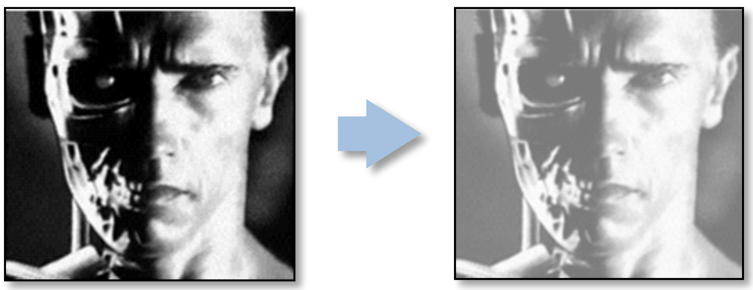
$g(x,y) = f(x,y) + 20$



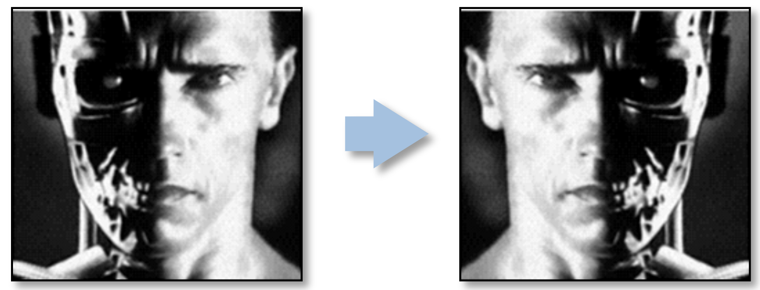
$g(x,y) = f(-x,y)$

Image transformations

- As with any function, we can apply operators to an image



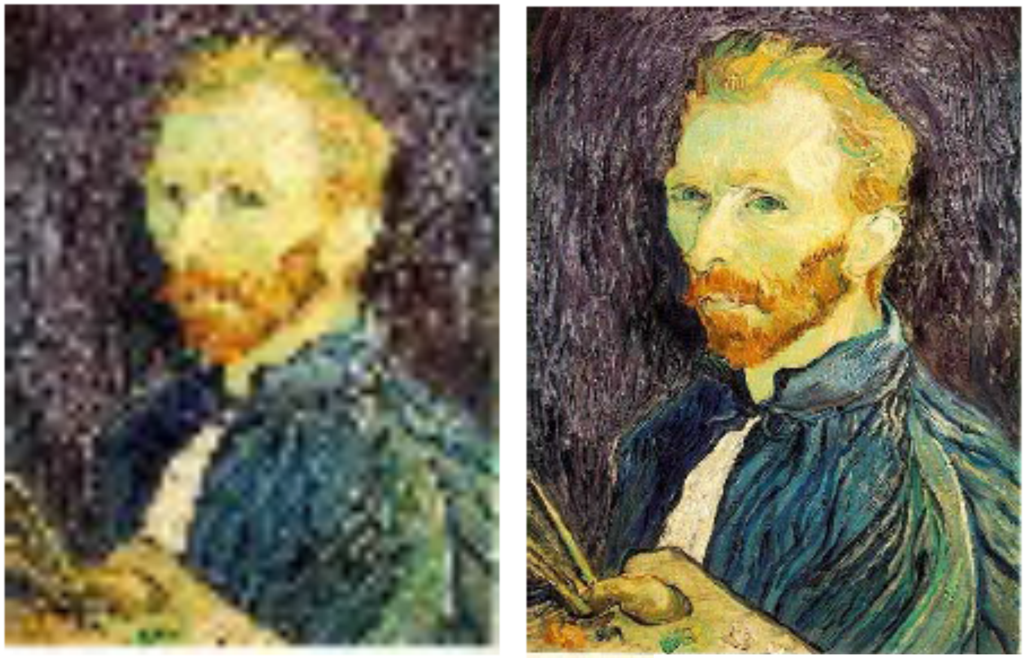
$g(x,y) = f(x,y) + 20$



$g(x,y) = f(-x,y)$

Filters

- Filtering
  - Form a new image whose pixels are a combination of the original pixels
- Why?
  - To get useful information from images
    - E.g., extract edges or contours (to understand shape)
  - To enhance the image
    - E.g., to blur to remove noise
    - E.g., to sharpen to “enhance image” a la CSI



Super-resolution

Noise reduction

Noise reduction

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

Take lots of images and average them!

How to formulate as filtering?

Source: S. Seitz

Image filtering

- Modify the pixels in an image based on some function of a local neighborhood of each pixel

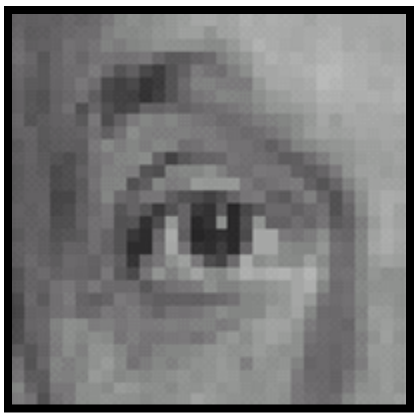
Local image data

Modified image data

$$f[m,n] \rightarrow S \rightarrow g[m,n]$$

Source: L. Zhang

Filters: examples



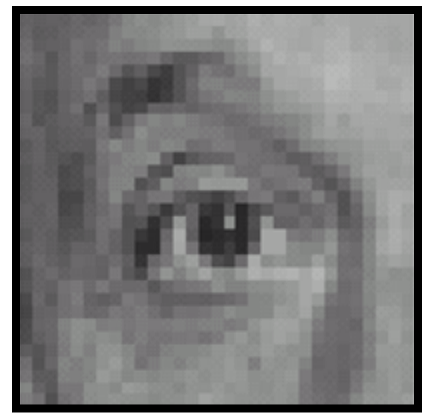
Original (f)

*

0	0	0
0	1	0
0	0	0

Kernel (k)

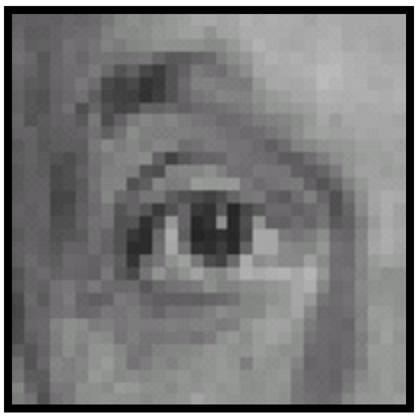
=



Identical image (g)

Source: D. Lowe

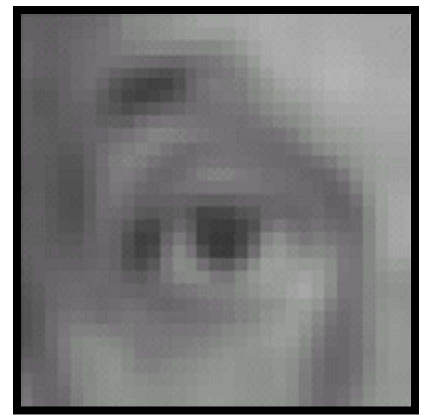
Filters: examples



Original (f)

$$* \quad \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad =$$

Kernel (k)



Blur (with a mean filter) (g)

Source: D. Lowe

Mean filtering

H

$*$

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

F

Mean filtering/Moving average

$F[x, y]$

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							

Mean filtering/Moving average

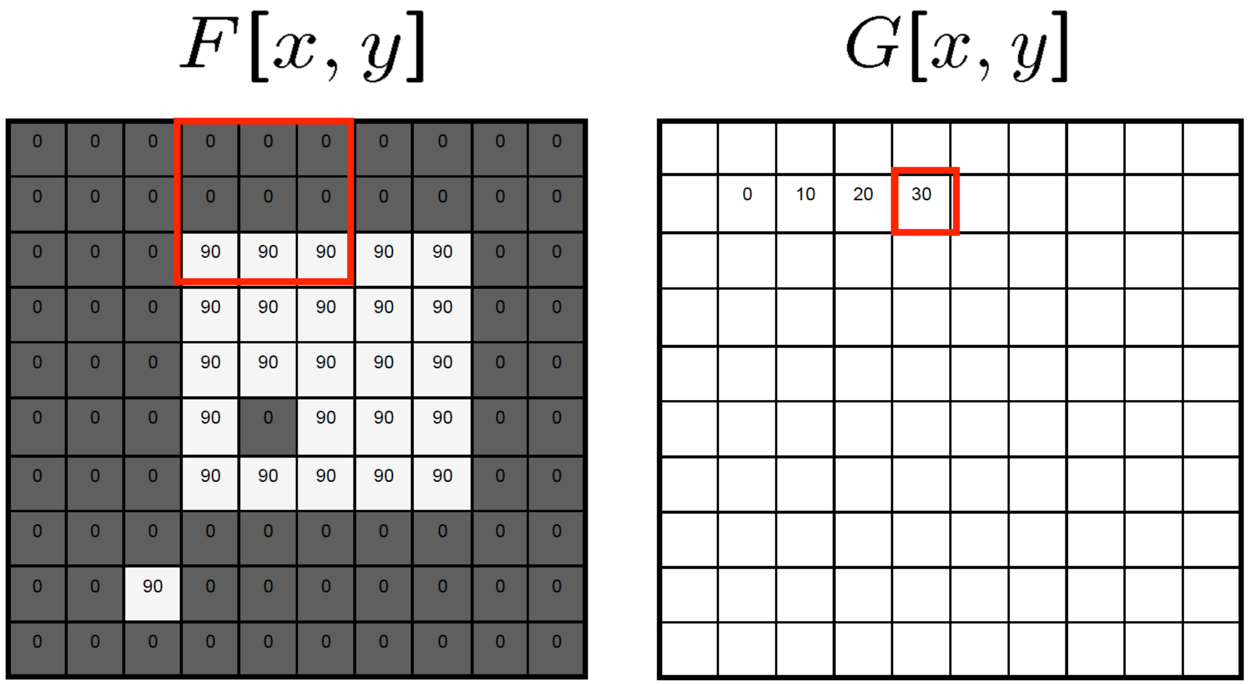
$F[x, y]$

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						

Mean filtering/Moving average



Mean filtering/Moving average

$F[x, y]$

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				

Mean filtering/Moving average

$F[x, y]$

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	

Mean filtering

H * F = G

H

F

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

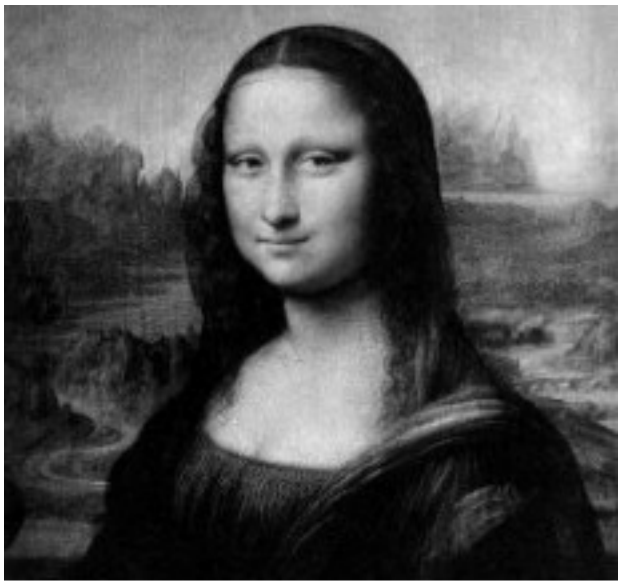
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Mean filtering/Moving Average

- Replace each pixel with an average of its neighborhood
- Achieves smoothing effect
  - Removes sharp features

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

Filters: Thresholding



$$g(m,n) = \begin{cases} 255, & f(m,n) > A \\ 0 & otherwise \end{cases}$$

Linear filtering

- One simple version: linear filtering
  - Replace each pixel by a linear combination (a weighted sum) of its neighbors
  - Simple, but powerful
  - Cross-correlation, convolution
- The prescription for the linear combination is called the “kernel” (or “mask”, “filter”)

10	5	3
4	6	1
1	1	8

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

	8	

Modified image data

Source: L. Zhang

Filter Properties

- Linearity
  - Weighted sum of original pixel values
  - Use same set of weights at each point
  - $S[f + g] = S[f] + S[g]$
  - $S[k f + m g] = k S[f] + m S[g]$

Linear Systems

- Is mean filtering/moving average linear?
  - Yes

- Is thresholding linear?
  - No

Filter Properties

- Linearity
  - Weighted sum of original pixel values
  - Use same set of weights at each point
  - S[f + g] = S[f] + S[g]
  - S[p f + q g] = p S[f] + q S[g]
- Shift-invariance
  - If $f[m,n] \overset{S}{\rightarrow} g[m,n]$, then $f[m-p,n-q] \overset{S}{\rightarrow} g[m-p, n-q]$
  - The operator behaves the same everywhere

Cross-correlation

Let F be the image, H be the kernel (of size 2*k*+1 x 2*k*+1), and G be the output image

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i+u, j+v]$$

This is called a **cross-correlation** operation:

$$G = H \otimes F$$

- Can think of as a "dot product" between local neighborhood and kernel for each pixel

Convolution

- Same as cross-correlation, except that the kernel is "flipped" (horizontally and vertically)

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

This is called a **convolution** operation:

$$G = H * F$$

- Convolution is **commutative** and **associative**

Cross-correlation

Let F be the image, H be the kernel (of size 2*k*+1 x 2*k*+1), and G be the output image

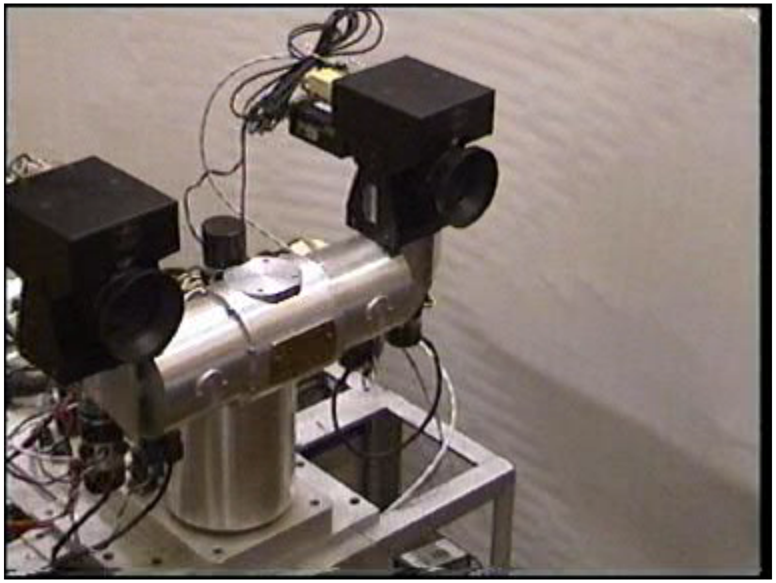
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

This is called a **cross-correlation** operation:

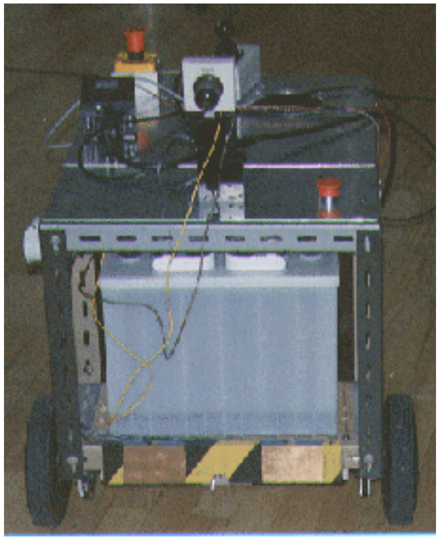
$$G = H \otimes F$$

- Can think of as a "dot product" between local neighborhood and kernel for each pixel

Stereo head



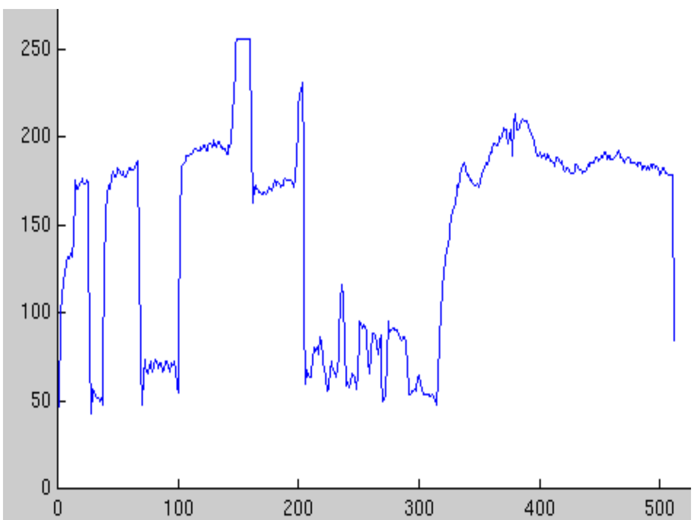
Camera on a mobile vehicle

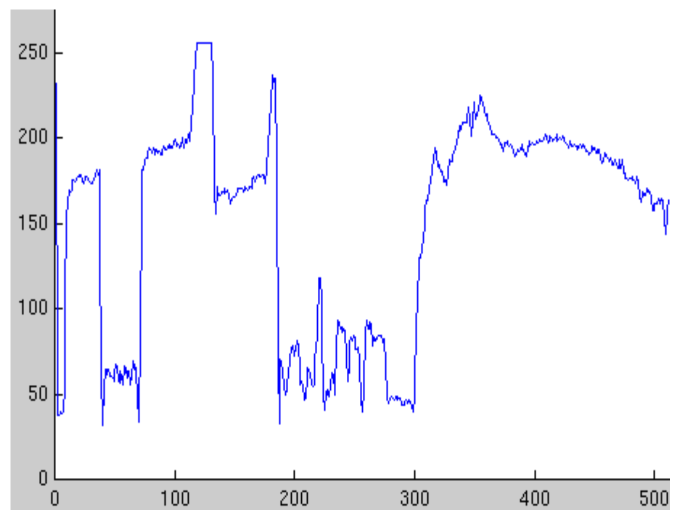


(COURTESY SONY)

Example image pair – parallel cameras

Intensity profiles





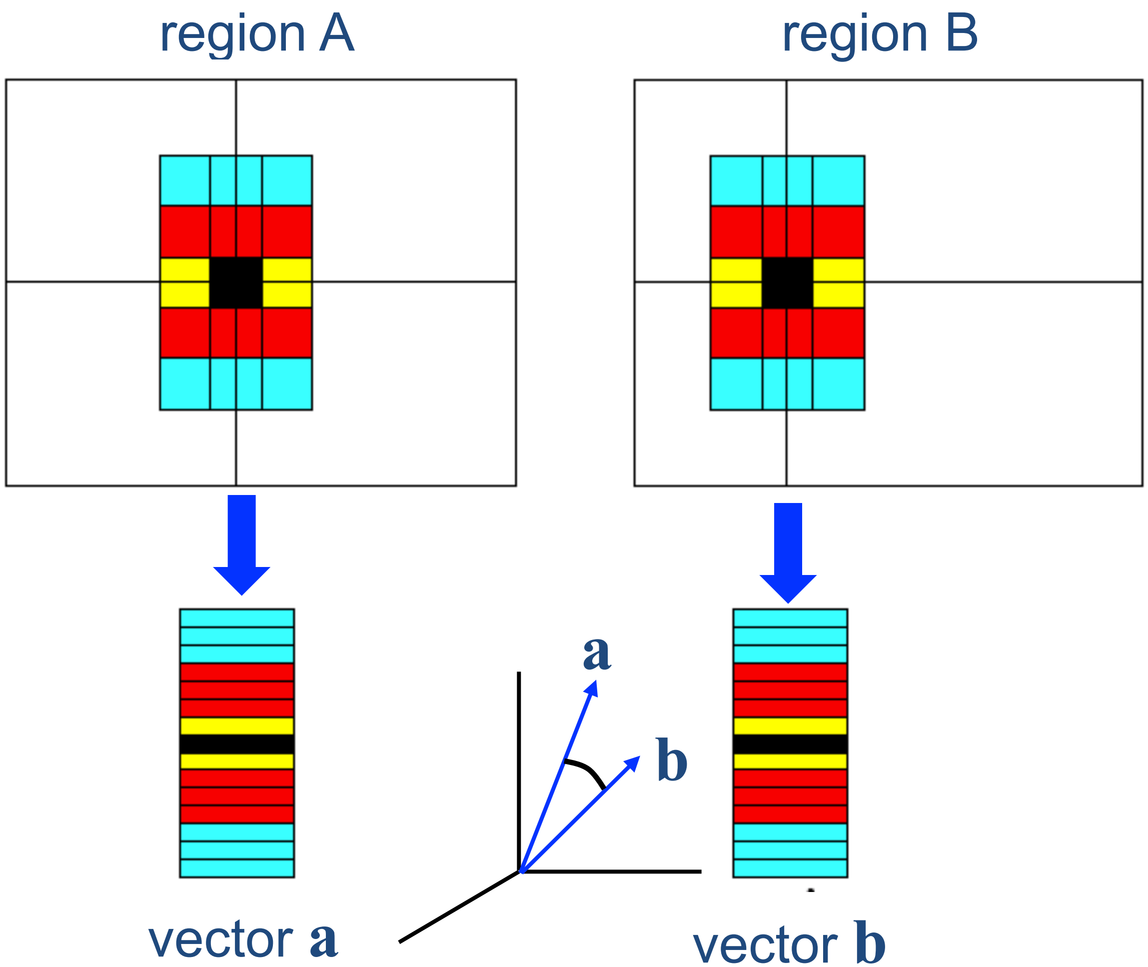
- Clear correspondence between intensities, but also noise and ambiguity

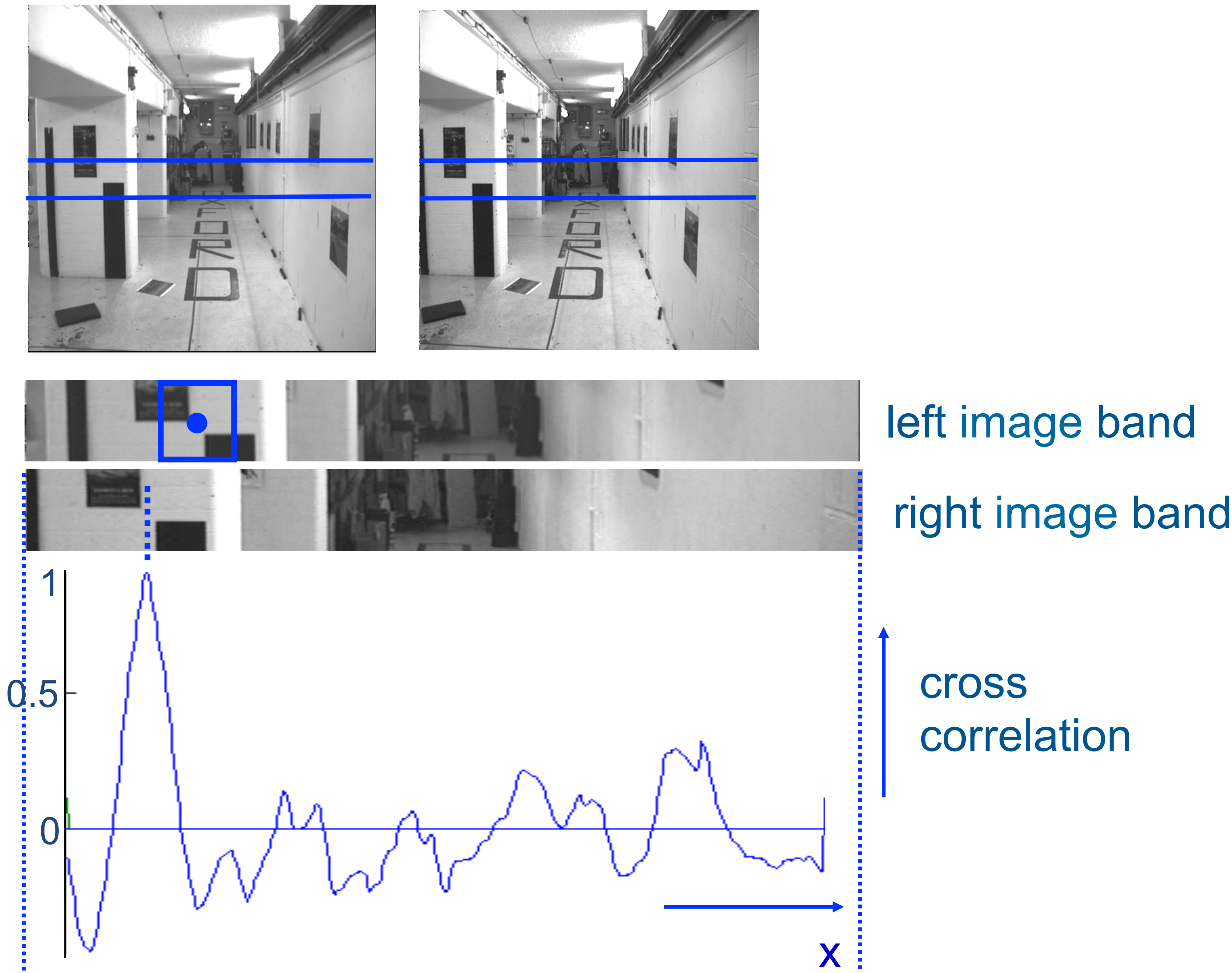
Normalized Cross Correlation

write regions as vectors

$$\mathtt{A} \rightarrow \mathbf{a}, \;\; \mathtt{B} \rightarrow \mathbf{b}$$

$$\mathrm{NCC} = \frac{\mathbf{a}.\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}$$





left image band
right image band
1
0.5
0
cross
correlation
x

Convolution

- Same as cross-correlation, except that the kernel is "flipped" (horizontally and vertically)

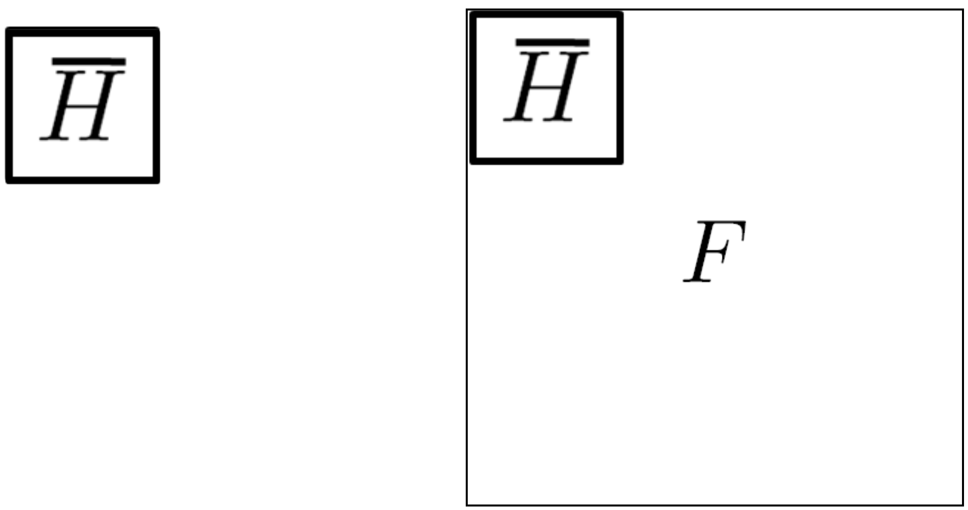
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

This is called a **convolution** operation:

$$G = H * F$$

- Convolution is **commutative** and **associative**

Convolution



Adapted from F. Durand