SUV Color Space & Filtering

Computer Vision I

CSE252A

Lecture 9

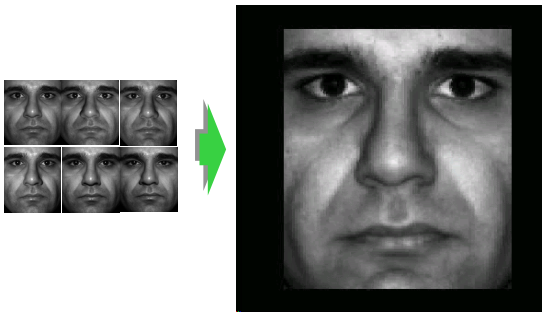
Announcement

- HW2 assigned

SUV Color Space for Photometric Stereo of Glossy Objects

Note: You'll use this in HW2.

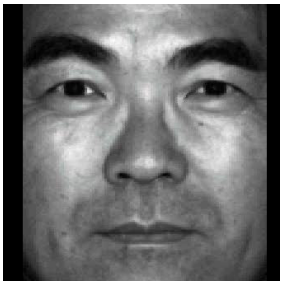
Lambertian Photometric Stereo



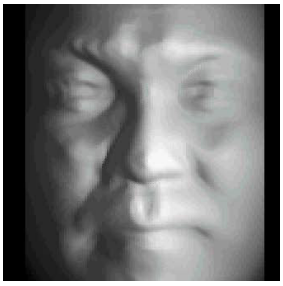
Reconsstruction with Albedo Map

Without the albedo map

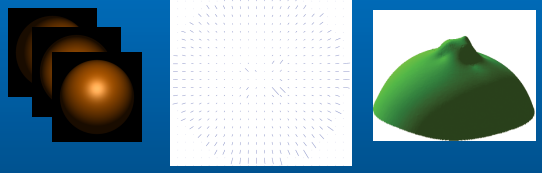
Another person



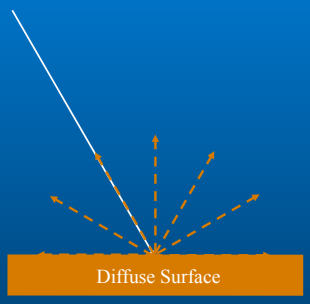
No Albedo map

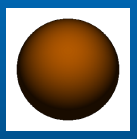


Motivation: Lambertian Algorithm Applied to NonLambertian Surface: Photometric Stereo



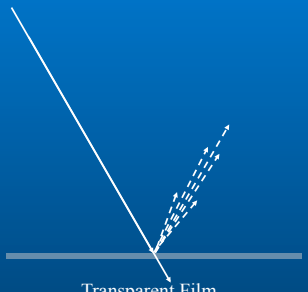
Dichromatic Reflection Model





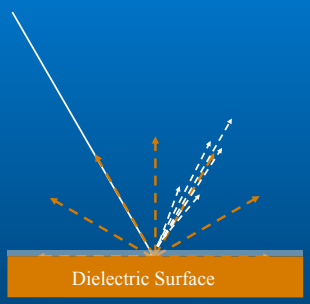
Color depends on light source color and diffuse color

Dichromatic Reflection Model



Color of light source

Dichromatic Reflection Model



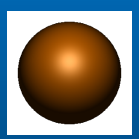


Image Formation: Color Channel *k*

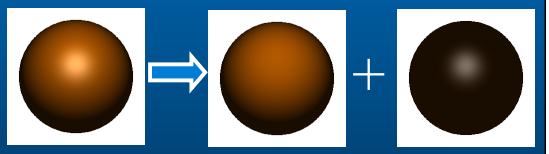
$L(\lambda)$ = Spectral Power Distribution of light source

$C_k(\lambda)$ = Camera Sensitivity

$S_k = \int C_k(\lambda) L(\lambda) d\lambda.$ Specular Color

$D_k = \int C_k(\lambda) L(\lambda) g_d(\lambda) d\lambda$ Diffuse Color

$$I_k = (D_k f_d + S_k f_s(\theta))\, \hat{\mathbf{n}} \cdot \hat{\mathbf{l}},$$



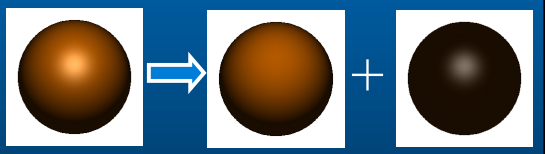
Where f_d and f_s are the diffuse and specular BRDF)

Image formation: Color Channel *k*

$$I_k = (D_k f_d + S_k f_s(\boldsymbol{\theta}))\,\hat{\mathbf{n}} \cdot \hat{\mathbf{l}},$$

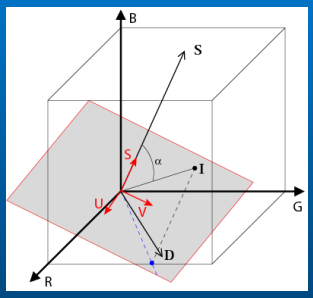
$$\begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix} = (f_d \mathbf{n} \cdot l) \begin{bmatrix} D_r \\ D_g \\ D_b \end{bmatrix} + (f_s(\theta) \mathbf{n} \cdot l) \begin{bmatrix} S_r \\ S_g \\ S_b \end{bmatrix}$$

Image color lies in span of diffuse color ***D*** and specular color ***S***.



David Kriegman

Data-dependent SUV Color Space



$$\mathbf{I}_{SUV} = [R]\mathbf{I}_{RGB}$$

[R] = [**S** | **U** | **V**]^t

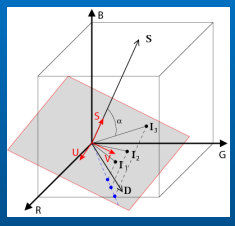
$$[R] \in SO(3)$$

First row of R is specular color S. Other rows are orthogonal to S

U,V spans a plane orthogonal to **S**

David Kriegman

Properties of SUV



- **Data-dependent.**
- **Rotational (hence, linear) Transformation.**
- **The S channel encodes the entire specular component and an unknown amount of diffuse component.**
- **Shading information is preserved in u and v channels.**

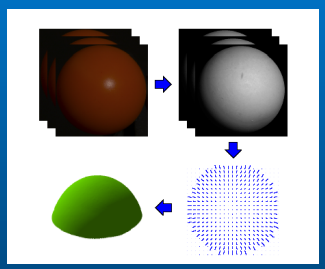
$$I_U = \mathbf{r}_2^\top \mathbf{D} f_d \hat{\mathbf{n}} \cdot \hat{\mathbf{l}}$$
$$I_V = \mathbf{r}_3^\top \mathbf{D} f_d \hat{\mathbf{n}} \cdot \hat{\mathbf{l}}.$$

David Kriegman

Example

David Kriegman

Multi-channel Photometric Stereo



David Kriegman

Multi-channel Photometric Stereo

$\mathbf{J} = [I_U \quad I_V]^\top$
$\mathbf{J}^k$: 2-channel color vector under the k^{th} light source.
$\hat{\mathbf{l}}^k$: The k^{th}three light source directions.
$\boldsymbol{\rho}$: 2-channel UV albedo.

$$\mathbf{J}^k = [I_U^k, I_V^k]^\top = (\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}^k)\boldsymbol{\rho},$$

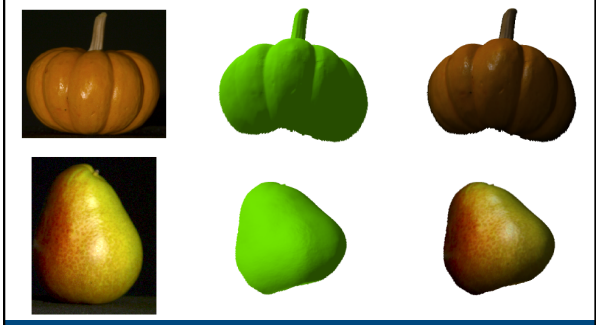
Shading vector $= \mathbf{F} = [f^1, f^2, f^3]^\top = [\hat{\mathbf{l}}^1\ \hat{\mathbf{l}}^2\ \hat{\mathbf{l}}^3]^\top \hat{\mathbf{n}}$

Intensity matrix $= [J] = \begin{bmatrix} J_1^1 & J_2^1 \\ J_1^2 & J_2^2 \\ J_1^3 & J_2^3 \end{bmatrix} = \begin{bmatrix} f^1\rho_U & f^1\rho_V \\ f^2\rho_U & f^2\rho_V \\ f^3\rho_U & f^3\rho_V \end{bmatrix} = \mathbf{F}\boldsymbol{\rho}^\top.$

The least squares estimate of the shading vector $\mathbf{F}$ is the principal eigenvector of $[J][J]^\top$. Once the shading vector is known, the surface normal is found by solving the matrix equation $\mathbf{F} = [\hat{\mathbf{l}}^1\ \hat{\mathbf{l}}^2\ \hat{\mathbf{l}}^3]^\top \hat{\mathbf{n}}$.

David Kriegman

Qualitative Results



David Kriegman

Quantitative Results

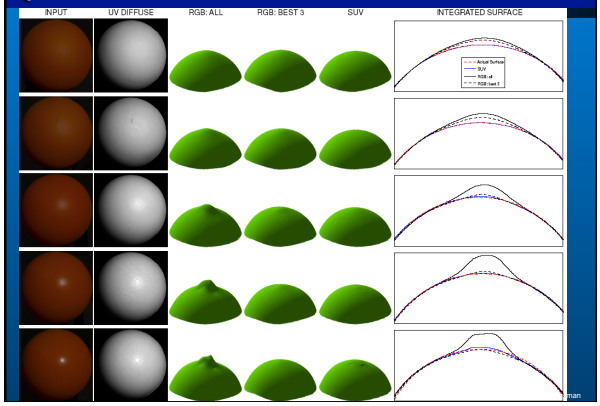
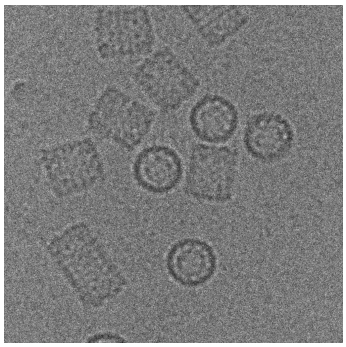
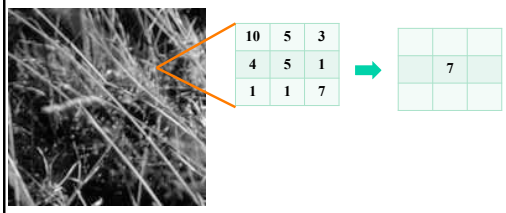


Image Filtering



What is image filtering?

Producing a new image where the value at a pixel in the output image is a function of a neighborhood of the pixel location in the input image.



Smoothing by Averaging

Kernel: ■

Linear Filters

- General process:
  - Form new image whose pixels are a weighted sum of original pixel values, using the same set of weights at each point.
- Properties
  - Output is a linear function of the input
  - Output is a shift-invariant function of the input (i.e. shift the input image two pixels to the left, the output is shifted two pixels to the left)
- Example: smoothing by averaging
  - form the average of pixels in a neighborhood
- Example: smoothing with a Gaussian
  - form a weighted average of pixels in a neighborhood
- Example: finding a derivative

Linear functions

- Simplest: linear filtering.
  - Replace each pixel by a linear combination of its neighbors.
- The prescription for the linear combination is called the "convolution kernel".

10	5	3
4	5	1
1	1	7

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel weights

	7	

Modified image data 11

(Freeman)

Convolution

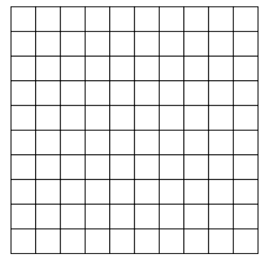


Image (I)

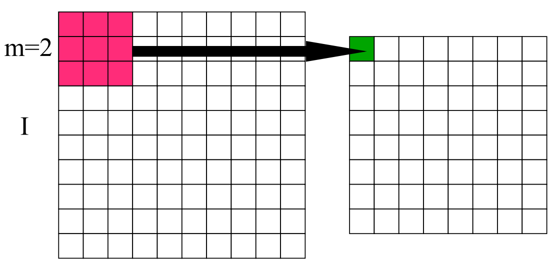
*

1	2	1
-1	-2	-1

Kernel (K)

Note: Typically Kernel is relatively small in vision applications.

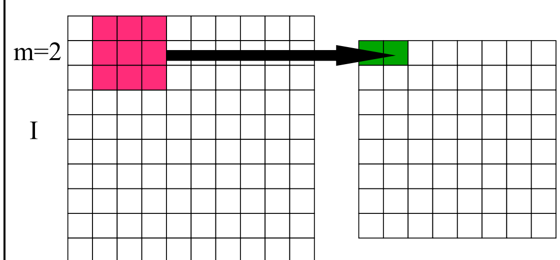
Convolution: R= K*I



Kernel size is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

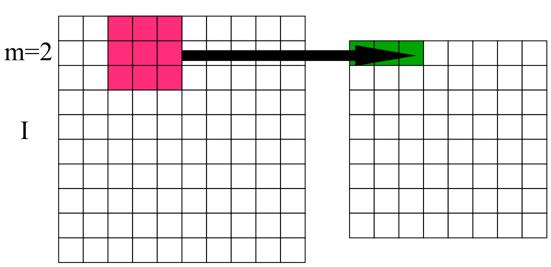
Convolution: R= K*I



Kernel size is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

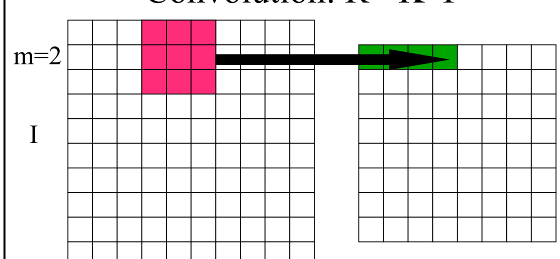
Convolution: R= K*I



Kernel size is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

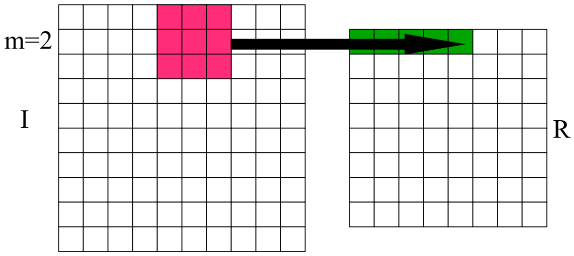
Convolution: R= K*I



Kernel size is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

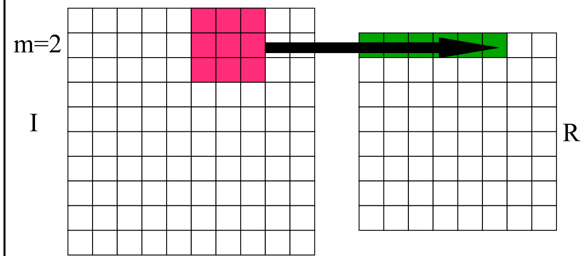
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j)=\sum_{h=-m/2}^{m/2}\sum_{k=-m/2}^{m/2}K(h,k)I(i-h,j-k)$$

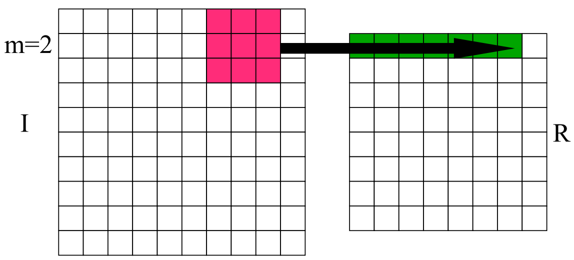
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j)=\sum_{h=-m/2}^{m/2}\sum_{k=-m/2}^{m/2}K(h,k)I(i-h,j-k)$$

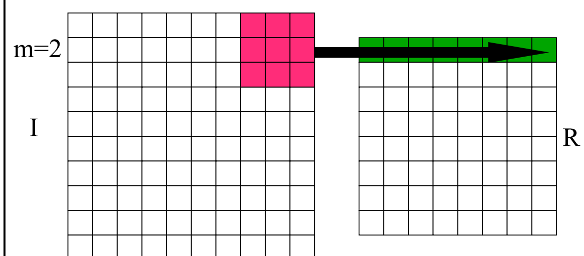
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j)=\sum_{h=-m/2}^{m/2}\sum_{k=-m/2}^{m/2}K(h,k)I(i-h,j-k)$$

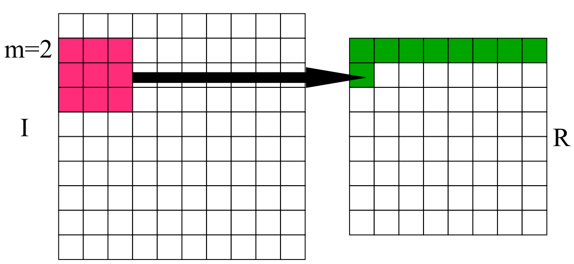
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j)=\sum_{h=-m/2}^{m/2}\sum_{k=-m/2}^{m/2}K(h,k)I(i-h,j-k)$$

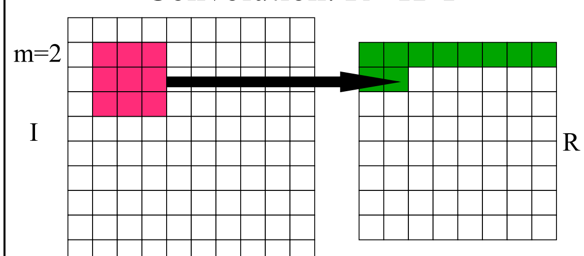
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j)=\sum_{h=-m/2}^{m/2}\sum_{k=-m/2}^{m/2}K(h,k)I(i-h,j-k)$$

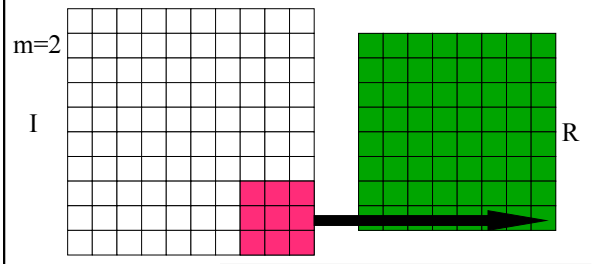
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j)=\sum_{h=-m/2}^{m/2}\sum_{k=-m/2}^{m/2}K(h,k)I(i-h,j-k)$$

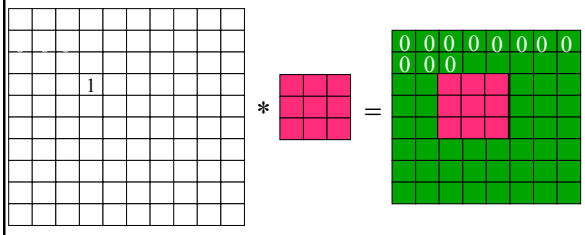
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

Impulse Response



Convolutional Neural Networks

- Core operation is, not surprisingly, convolution.
- During training of a CNN, the weights of the convolution kernels are learned.

Linear filtering (warm-up slide)

original

0	0	0
0	1	0
0	0	0

?

Linear filtering (warm-up slide)

original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Linear filtering

original

0	0	0
0	0	1
0	0	0

Pixel offset

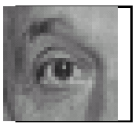
?

shift

original

0	0	0
0	0	1
0	0	0

Pixel offset



Shifted one
Pixel to the left

Linear filtering

original

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

?

Blurring

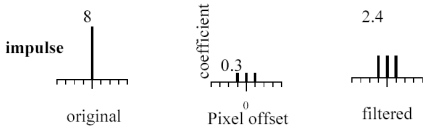
original

$\frac{1}{9}$

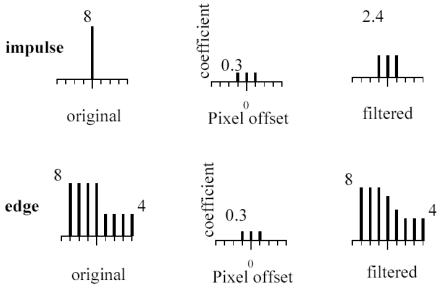
1	1	1
1	1	1
1	1	1

Blurred (filter
applied in both
dimensions).

Blur Examples
1-Dimensional



Blur Examples
1-Dimensional



Practice with linear filters

Original

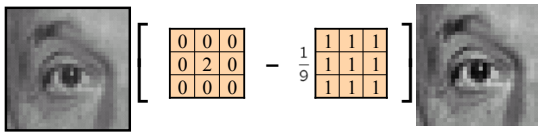
0	0	0
0	2	0
0	0	0

$-\frac{1}{9}$

1	1	1
1	1	1
1	1	1

?

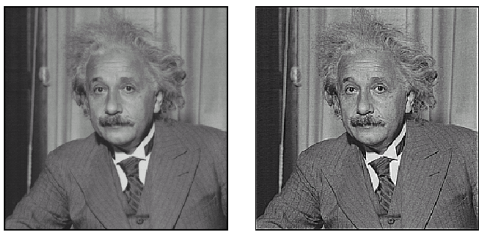
Practice with linear filters



Original

Sharpening filter
- Accentuates differences with local average

Sharpening



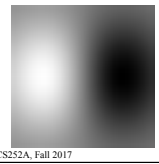
before **after**

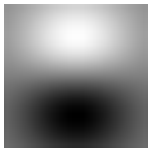
Smoothing by Averaging

Kernel: ■

Filters are templates

- Applying a filter at some image location can be seen as taking a dot-product between a neighborhood and the kernel.
- Filtering the image is like taking a dot product at each location.
- Insight
  - filters look like the effects they are intended to find
  - filters find effects they look like





Properties of Continuous Convolution

(Holds for discrete too)

Let f,g,h be images and * denote convolution

$$f * g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-u, y-v)g(u,v)\,du\,dv$$

- Commutative: f*g=g*f
- Associative: f*(g*h)=(f*g)*h
- Linear: for scalars a & b and images f,g,h

  (af+bg)*h=a(f*h)+b(g*h)
- Differentiation rule

$$\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g = f * \frac{\partial g}{\partial x}$$

Filtering to reduce noise

- Noise is what we're not interested in.
  - We'll discuss simple, low-level noise today: Light fluctuations; Sensor noise; Quantization effects; Finite precision
  - Not complex: shadows; extraneous objects.
- A pixel's neighborhood contains information about its intensity.
- Averaging noise reduces its effect.

Additive noise

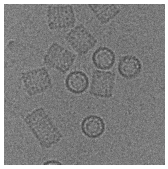
- I = S + N. Noise doesn't depend on signal.
- We'll consider:

$I_i = s_i + n_i$ with $E(n_i) = 0$

s_i deterministic. n_i a random var.

n_i, n_j independent for $i \neq j$

n_i, n_j identically distributed



Guassian Noise: sigma=1

Guassian Noise: sigma=16

Averaging Filter

- Mask with positive entries, that sum 1.
- Replaces each pixel with an average of its neighborhood.
- If all weights are equal, it is called a BOX filter.

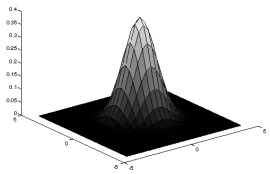
F

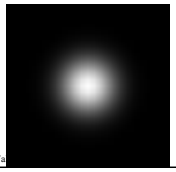
1/9

1	1	1
1	1	1
1	1	1

(Camps)

An Isotropic Gaussian





- The picture shows a smoothing kernel proportional to

$$e^{-\frac{x^2+y^2}{2\sigma^2}}$$

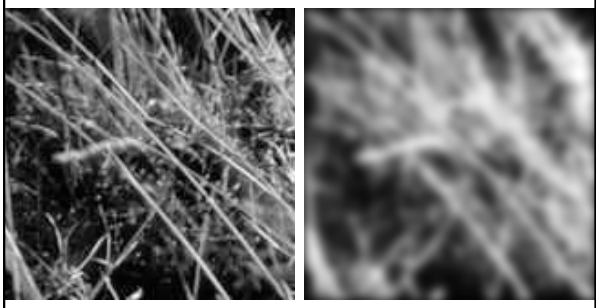
(which is a reasonable model of a circularly symmetric fuzzy blob)

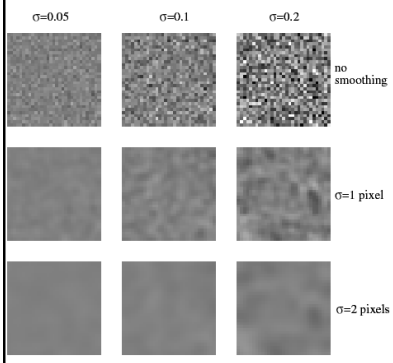
Smoothing by Averaging

Kernel: ■

Smoothing with a Gaussian

Kernel: 





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

Efficient Implementation

- Both, the BOX filter and the Gaussian filter are separable:
  - First convolve each row with a 1D filter
  - Then convolve each column with a 1D filter.

r(u)*c(v)*I(x,y)=(r(u)*c(v))*I(u,v)
=(c(u)*r(v))*I(u,v)
=c(u)*r(v)*I(x,y)

Fourier Transform

- 1-D transform (signal processing)
- 2-D transform (image processing)
- Consider 1-D

  Time domain ←→ Frequency Domain

  Real ←→Complex
- Consider time domain signal to be expressed as weighted sum of sinusoid. A sinusoid cos(ut+ϕ) is characterized by its phase ϕ and its frequency u
- The Fourier transform of the signal is a function giving the weights (and phase) as a function of frequency u.

Fourier Tansform

Discrete Fourier Transform (DFT) of I[x,y]

$$F[u,v] \equiv \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} I[x,y]e^{\frac{-2\pi j}{N}(xu+yv)}$$

Inverse DFT

$$I[x,y] \equiv \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F[u,v]e^{\frac{+2\pi j}{N}(ux+vy)}$$

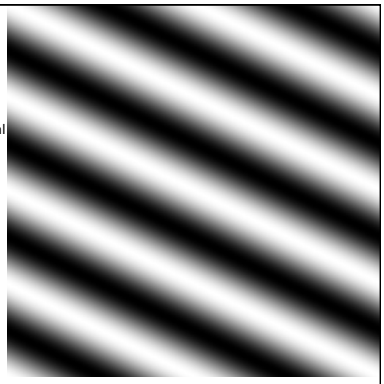
x,y: spatial domain
u,v: frequence domain
Implemented via the "Fast Fourier Transform" algorithm (FFT)

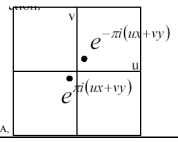
Fourier basis element

$e^{-i2\pi(ux+vy)}$

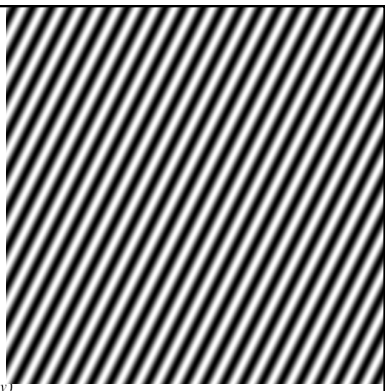
Transform is sum of orthogonal basis functions

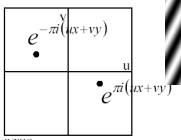
Vector (u,v)
- Magnitude gives frequency
- Direction gives orientation.



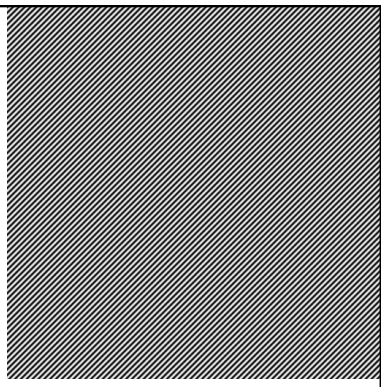


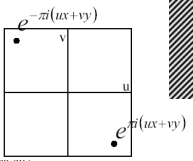
Here u and v are larger than in the previous slide.





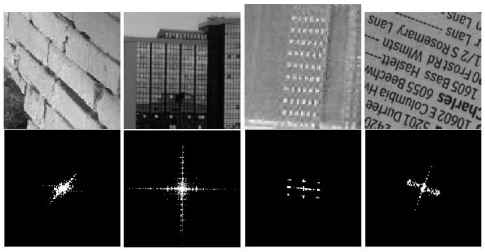
And larger still...





Using Fourier Representations

Dominant Orientation



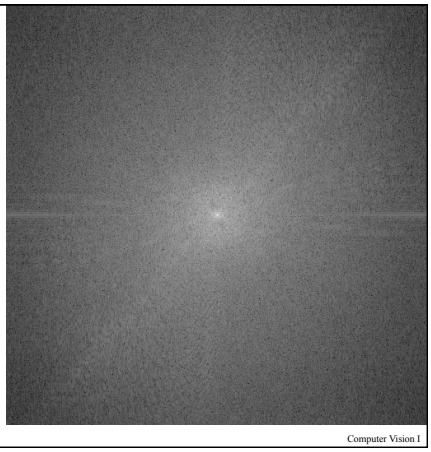
Limitations: not useful for local segmentation

Phase and Magnitude

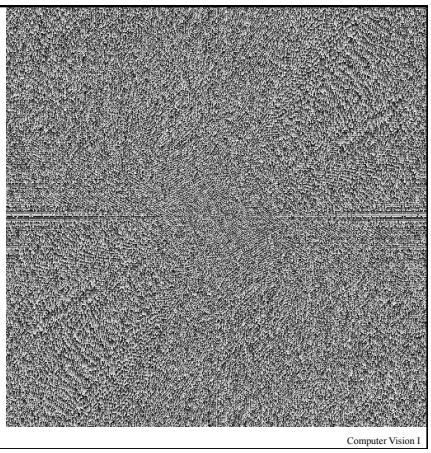
$e^{i\theta} = \cos\theta + i \sin\theta$

- Fourier transform of a real function is complex
  - difficult to plot, visualize
  - instead, we can think of the phase and magnitude of the transform
- Phase is the phase of the complex transform
- Magnitude is the magnitude of the complex transform
- Curious fact
  - all natural images have about the same magnitude transform
  - hence, phase seems to matter, but magnitude largely doesn't
- Demonstration
  - Take two pictures, swap the phase transforms, compute the inverse - what does the result look like?

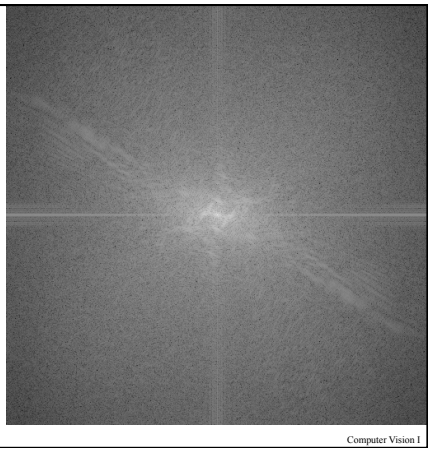
This is the magnitude transform of the cheetah pic



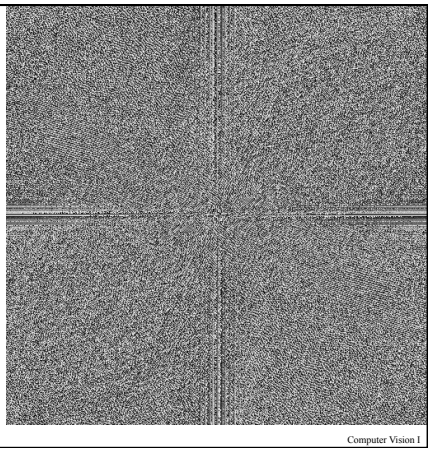
This is the phase transform of the cheetah pic



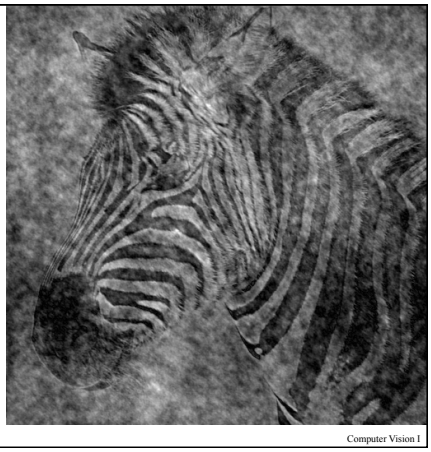
This is the magnitude transform of the zebra pic



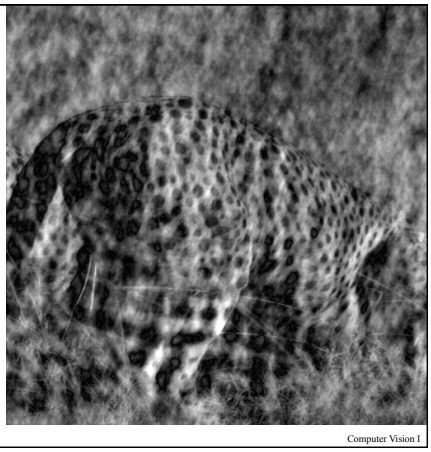
This is the phase transform of the zebra pic



Reconstruction with zebra phase, cheetah magnitude



Reconstruction with cheetah phase, zebra magnitude



The Fourier Transform and Convolution

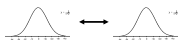
- If H and G are images, and F(.) represents Fourier transform, then

$$F(H*G) = F(H)F(G)$$

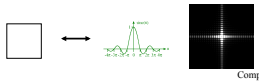
- Thus, one way of thinking about the properties of a convolution is by thinking of how it modifies the frequencies of the image to which it is applied.
- In particular, if we look at the power spectrum, then we see that convolving image H by G attenuates frequencies where G has low power, and amplifies those which have high power.
- This is referred to as the **Convolution Theorem**

Various Fourier Transform Pairs

- Important facts
  - scale function down ⇔ scale transform up
    i.e. high frequency = small details
  - The FT of a Gaussian is a Gaussian.



compare to box function transform



Other Types of Noise

- Impulsive noise
  - randomly pick a pixel and randomly set ot a value
  - saturated version is called salt and pepper noise
- Quantization effects
  - Often called noise although it is not statistical
- Unanticipated image structures
  - Also often called noise although it is a real repeatable signal.

Some other useful filtering techniques

- Median filter
- Anisotropic diffusion

Median filters : principle

Method :

1. rank-order neighbourhood intensities
2. take middle value

- non-linear filter
- no new grey levels emerge...

Median filters: Example for window size of 3

1,1,1,7,1,1,1,1

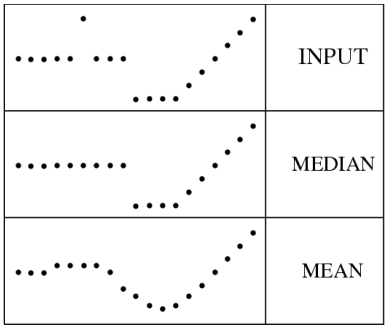
↓

?,1,1,1.1,1,1,?

advantage of this type of filter is that it
Eliminates spikes (salt & peper noise).

Median filters : example

filters have width 5 :



Median filters : analysis

median completely discards the spike,
linear filter always responds to all aspects

median filter preserves discontinuities,
linear filter produces rounding-off effects

DON'T become all too optimistic

Median filter : images

3 x 3 median filter :

sharpens edges, destroys edge cusps
and protrusions

Median filters : Gauss revisited

Comparison with Gaussian :

e.g. upper lip smoother, eye better preserved

Example of median

10 times 3 X 3 median

patchy effect
important details lost (e.g. ear-ring)