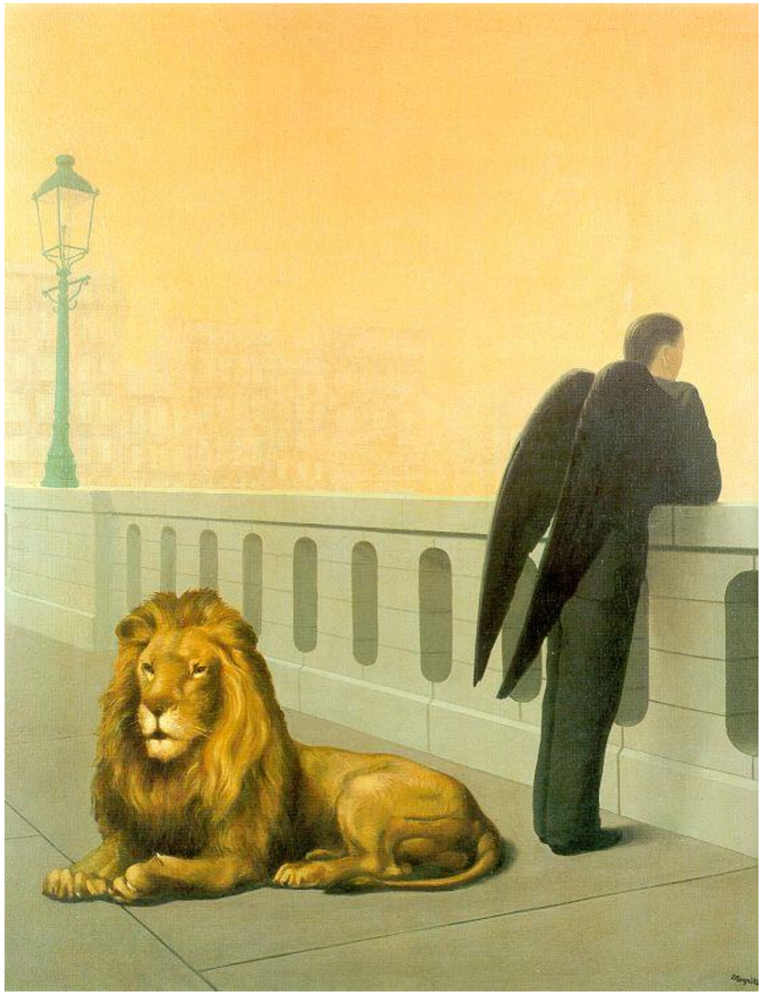
11/17/15

Midterm Review


Magritte, *Homesickness*

Computational Photography

Derek Hoiem, University of Illinois

Major Topics

- **Linear Filtering**
  - How it works
  - Template and Frequency interpretations
  - Image pyramids and their applications
  - Sampling (Nyquist theorem, application of low-pass filtering)
- **Light and color**
  - Lambertian shading, shadows, specularities
  - Color spaces (RGB, HSV, LAB)
  - Image-based lighting
- Techniques
  - Finding boundaries: intelligent scissors, graph cuts, where to cut and why
  - Texture synthesis: idea of sampling patches to synthesize, filling order
  - Compositing and blending: alpha compositing, Laplacian blending, Poisson editing
- **Warping**
  - Transformation matrices, homogeneous coordinates, solving for parameters via system of linear equations
- Modeling shape
  - Averaging and interpolating sets of points

Major Topics

- **Camera models and Geometry**
  - Pinhole model: diagram, intrinsic/extrinsic matrices, camera center (or center of projection), image plane
  - Focal length, depth of field, field of view, aperture size
  - Vanishing points and vanishing lines (what they are, how to find them)
  - Measuring relative lengths based on vanishing points and horizon
- Interest points
  - Trade-offs between repeatability and distinctiveness for detectors and descriptors
  - Harris (corner) detectors and Difference of Gaussian (blob) detectors
  - SIFT representation: what transformations is it robust to or not robust to
- Image stitching
  - Solving for homography
  - RANSAC for robust detection of inliers
- Object recognition and search
  - Use of "visual words" to speed search
  - Idea of geometric verification to check that points have consistent geometry
- Other camera systems
  - Kinect, lightfield camera, synthetic aperture --- how do they work?

Studying for Midterm

- Roughly 80% is related to topics in bold (linear filtering, warping, camera models, lighting, and geometry)
- No book, no notes, no computers/calculators allowed
- Bring a pencil or pen

Today’s review

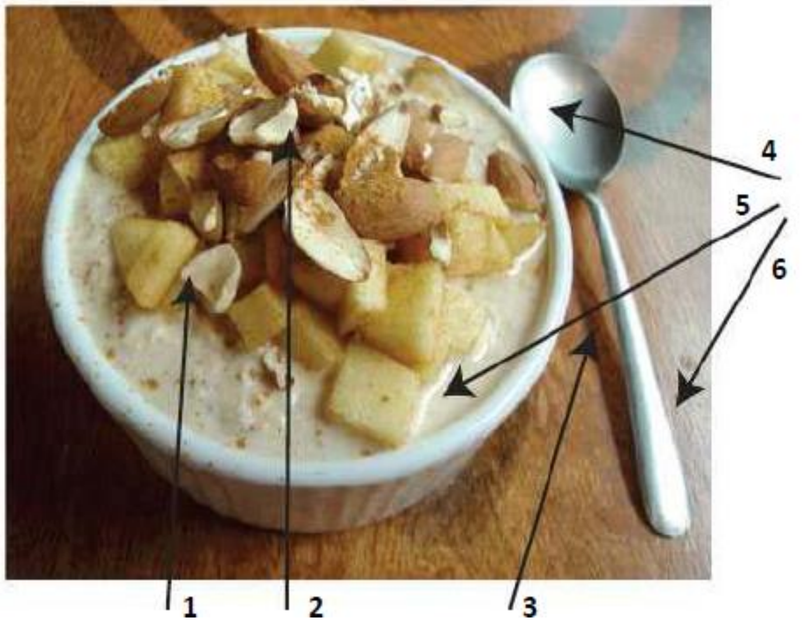
1. Light
2. Camera capture and geometry
3. Image filtering
4. Region selection and compositing
5. Solving for transformations

Purposes

- Remind you of key concepts
- Chance for you to ask questions

1. Light and color

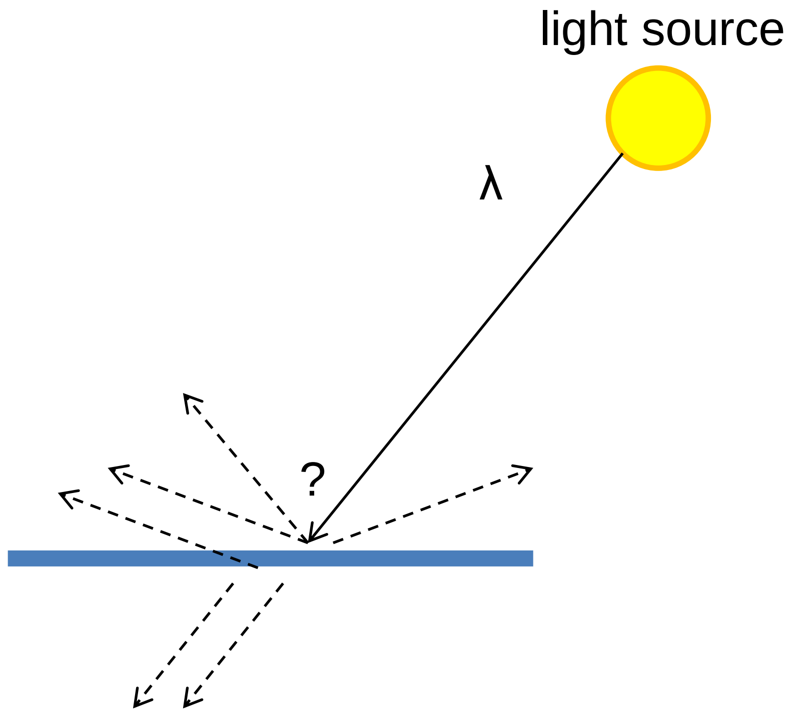
- Lighting
  - Lambertian shading, shadows, specularities
  - Color spaces (RGB, HSV, LAB)



How is light reflected from a surface?

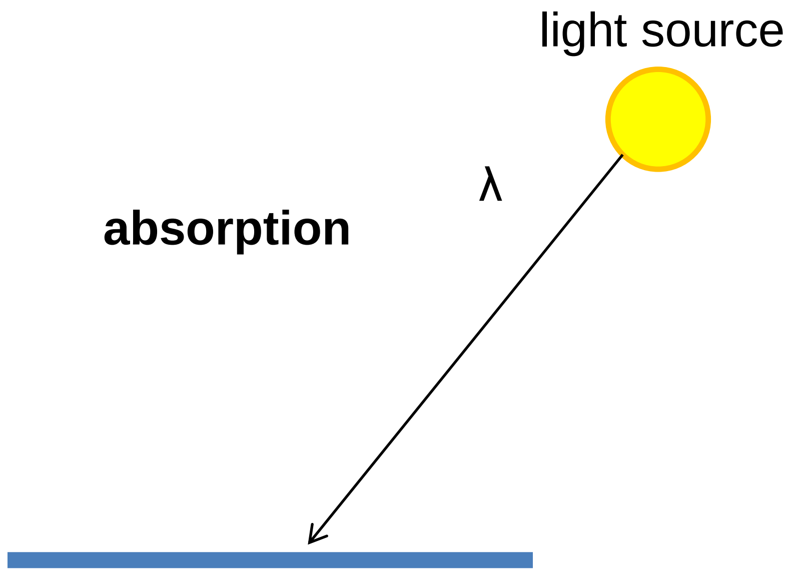
Depends on

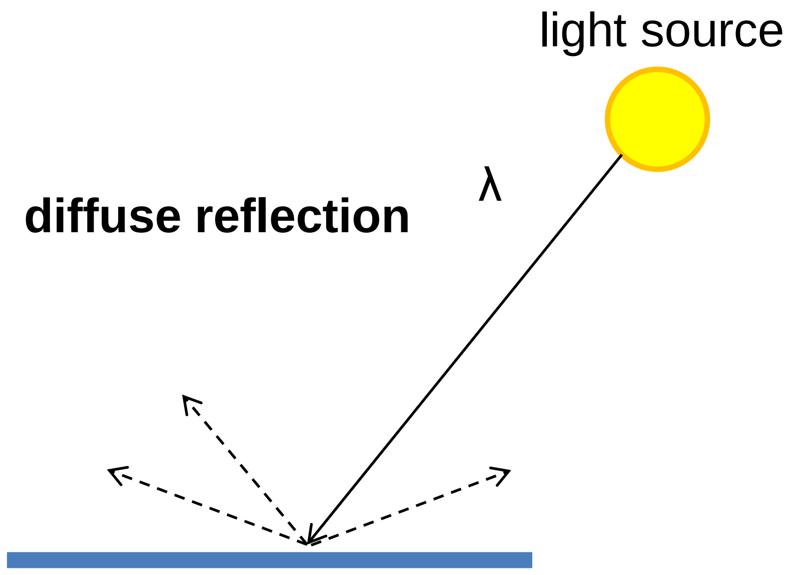
- Illumination properties: wavelength, orientation, intensity
- Surface properties: material, surface orientation, roughness, etc.



Lambertian surface

- Some light is absorbed (function of albedo)
- Remaining light is reflected equally in all directions (diffuse reflection)
- Examples: soft cloth, concrete, matte paints





Diffuse reflection

Intensity *does* depend on illumination angle because less light comes in at oblique angles.

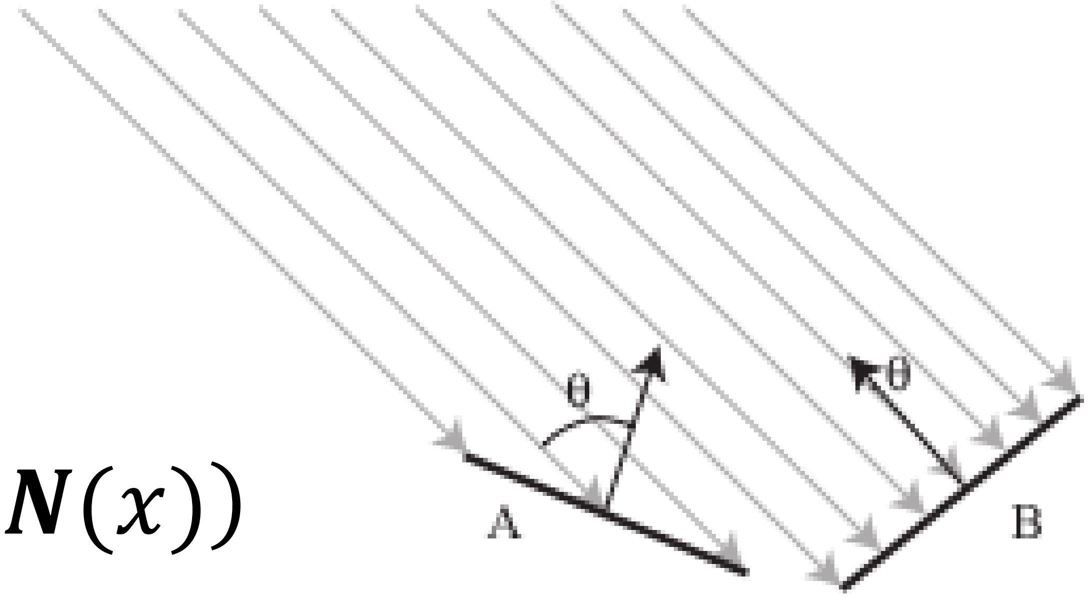
$\rho =$ albedo

$\boldsymbol{S} =$ directional source

$\boldsymbol{N} =$ surface normal

$\mathrm{I} =$ image intensity

$$I(x) = \rho(x)(\boldsymbol{S} \cdot \boldsymbol{N}(x))$$



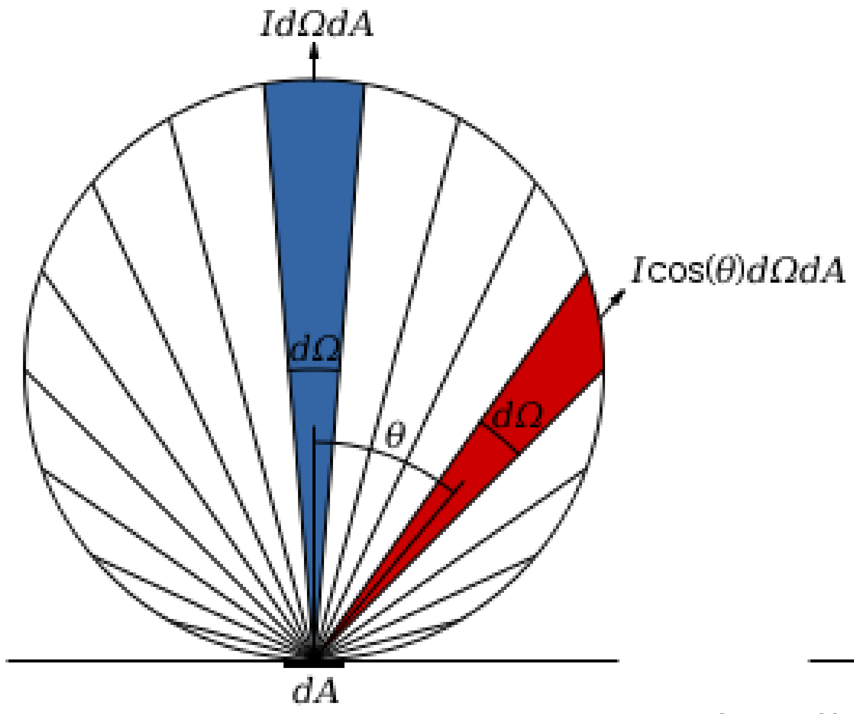
Slide: Forsyth

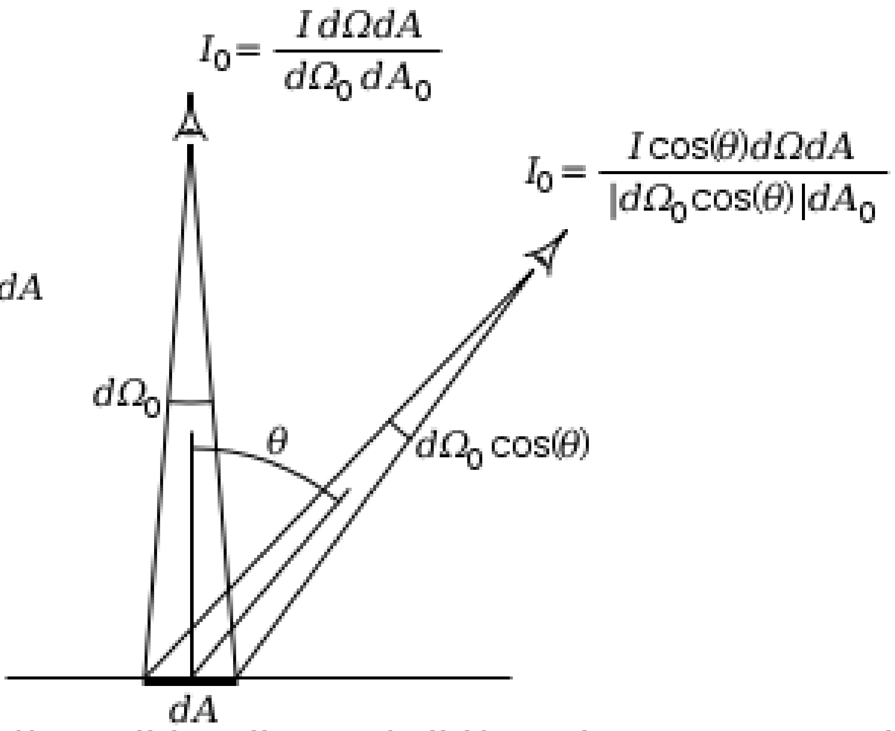
1
2

Diffuse reflection

Perceived intensity does *not* depend on viewer angle.

- Amount of reflected light proportional to cos(theta)
- Visible solid angle also proportional to cos(theta)



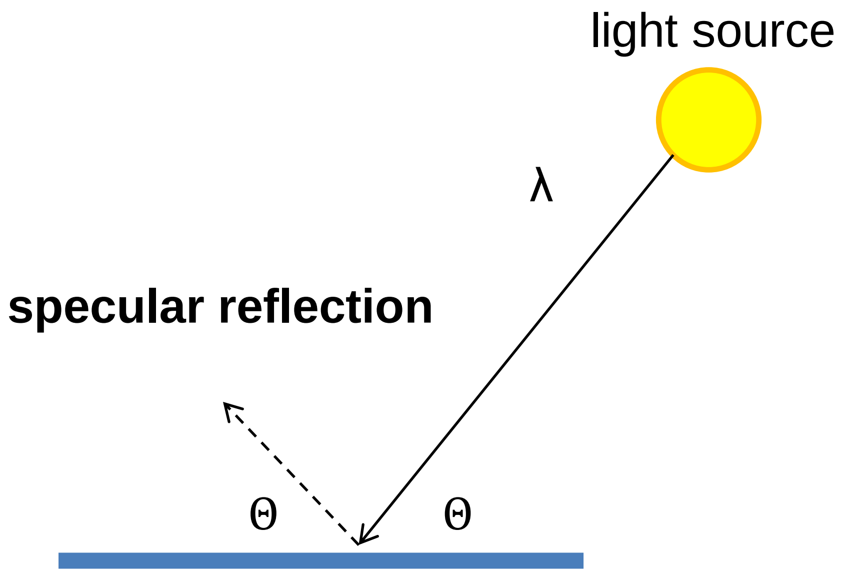


$I_0 = \frac{I\, d\Omega dA}{d\Omega_0\, dA_0}$
$I_0 = \frac{I\cos(\theta) d\Omega dA}{|d\Omega_0 \cos(\theta)| dA_0}$
$d\Omega_0$
θ
$d\Omega_0 \cos(\theta)$
dA

http://en.wikipedia.org/wiki/Lambert%27s_cosine_law

Specular Reflection

- Reflected direction depends on light orientation and surface normal
- E.g., mirrors are mostly specular



Flickr, by suzysputnik

Flickr, by piratejohnny

Many surfaces have both specular and diffuse components

- Specularity = spot where specular reflection dominates (typically reflects light source)

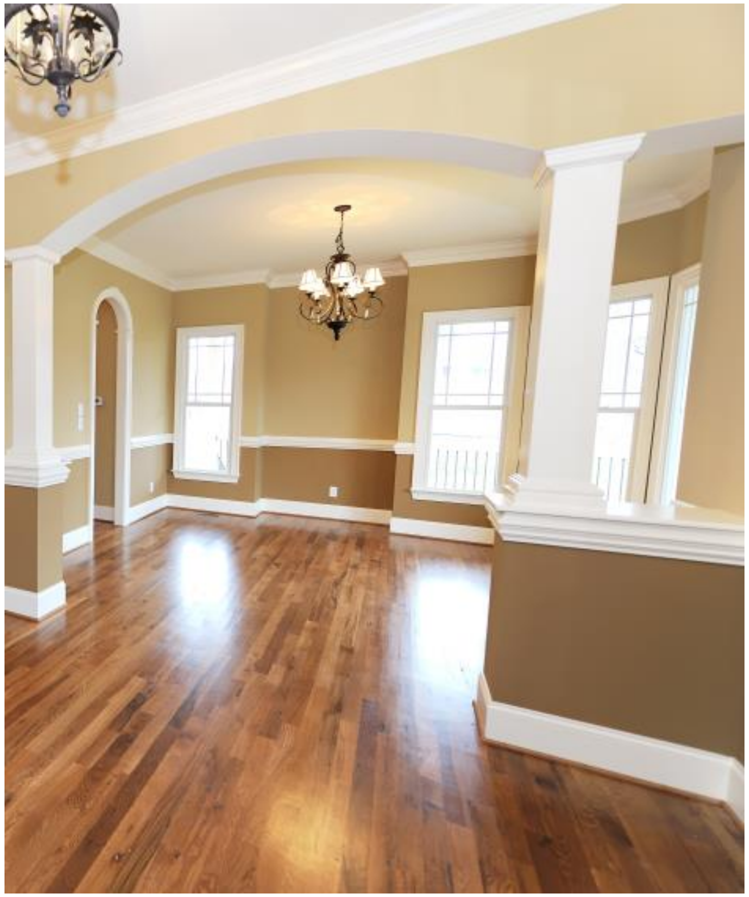
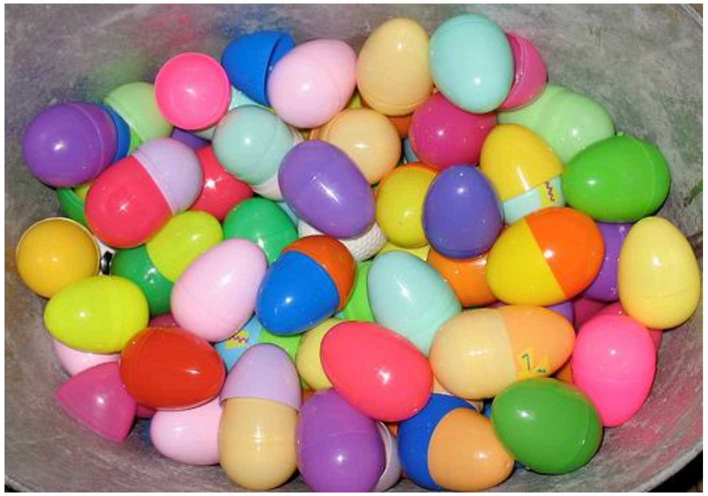
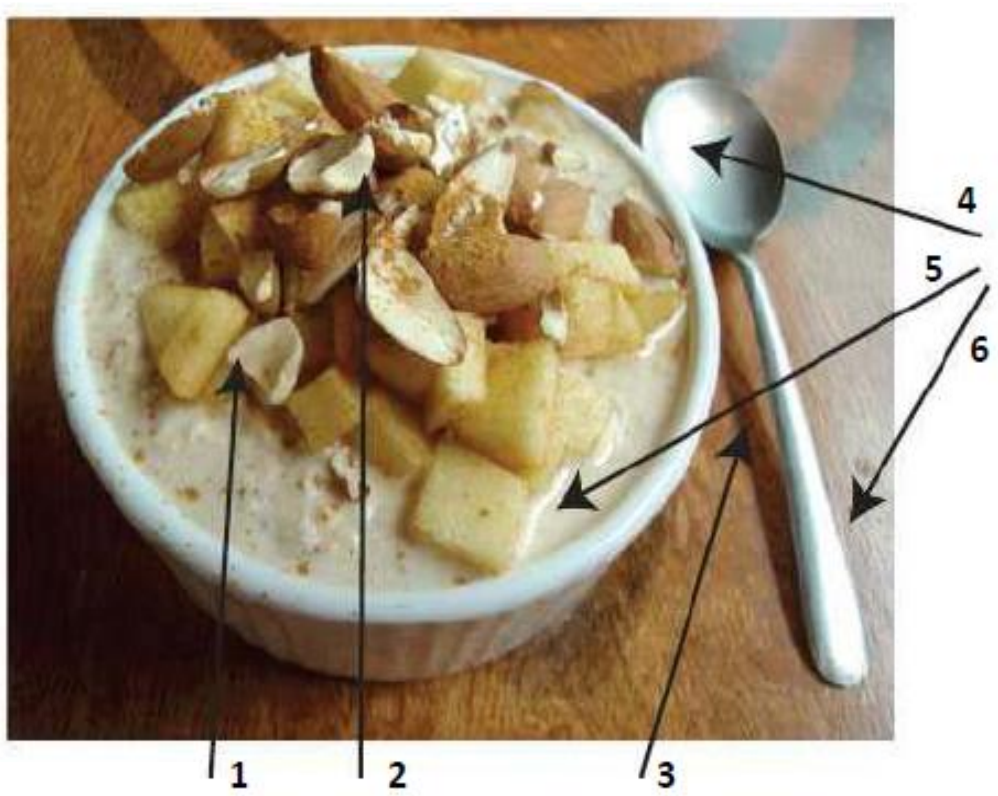


Photo: northcountryhardwoodfloors.com



Questions

1.

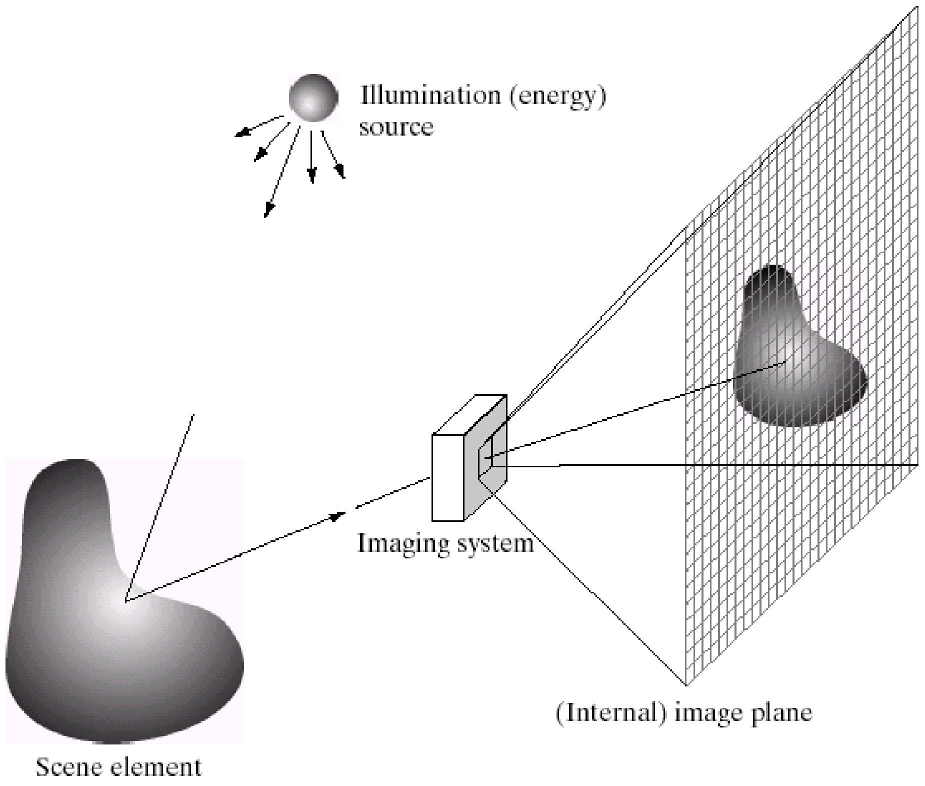


A. For each of the arrows in the above image, name the reasons the pixel near the end of the arrow has its brightness value and explain very briefly. The arrow pointing to milk is pointing to the thin bright line at the edge of the piece of apple; the arrow pointing to the spoon handle is pointing to the bright area on the handle.

Possible factors: albedo, shadows, texture, specularities, curvature, lighting direction

Discretization

- Because pixel grid is discrete, pixel intensities are determined by a range of scene points



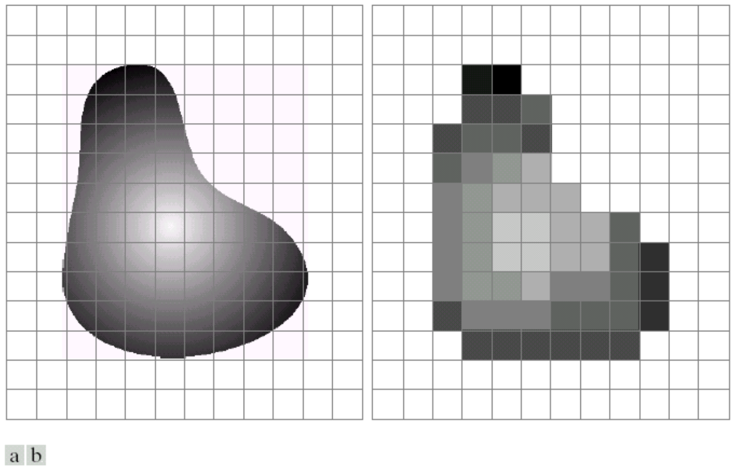
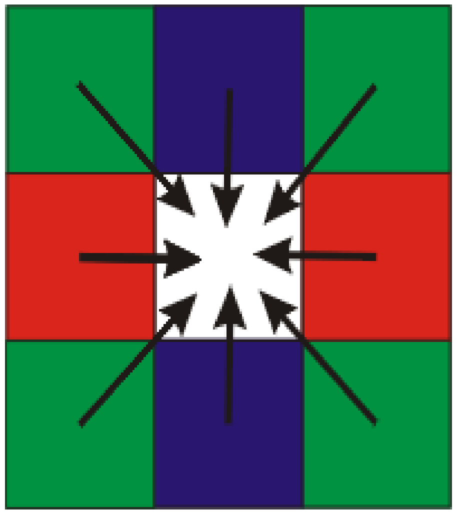
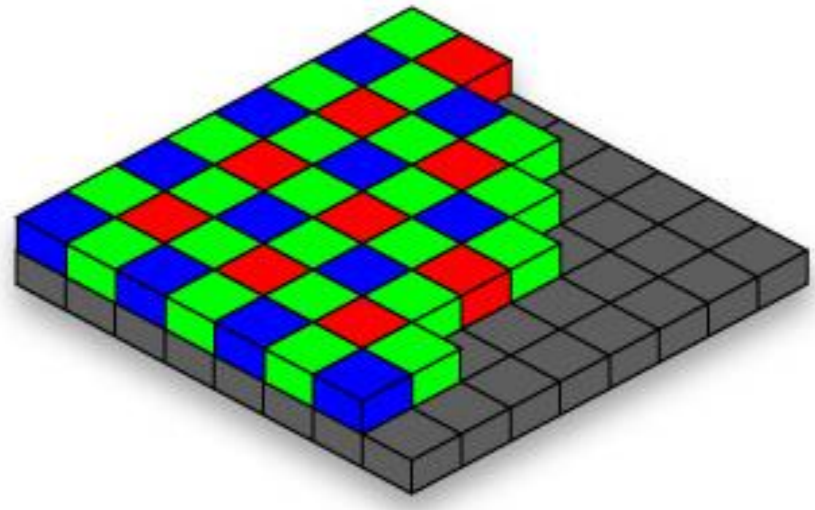
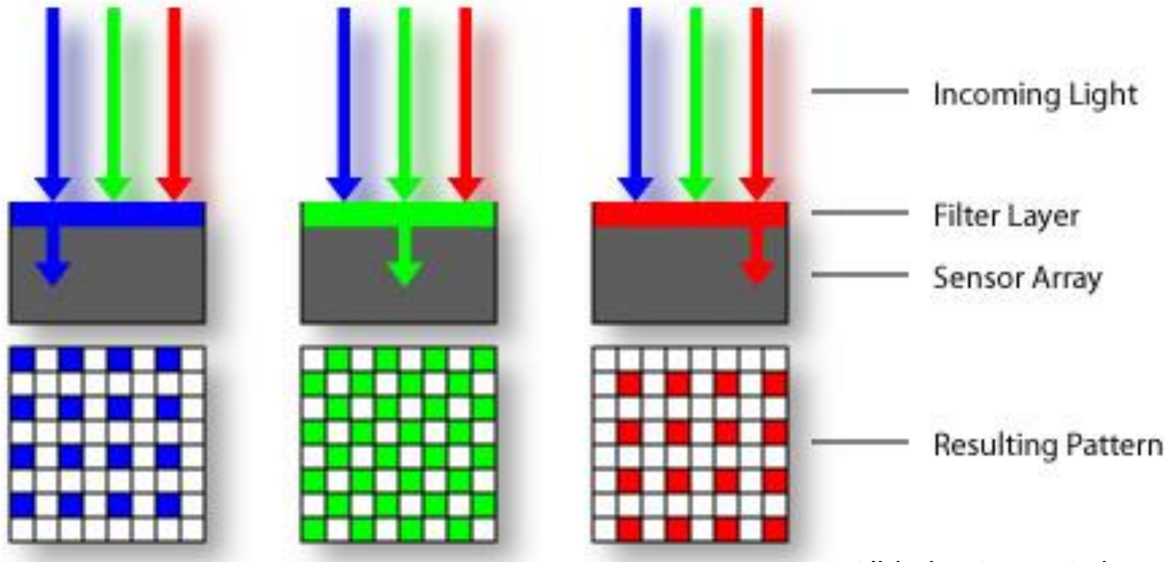


FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.

Color Sensing: Bayer Grid







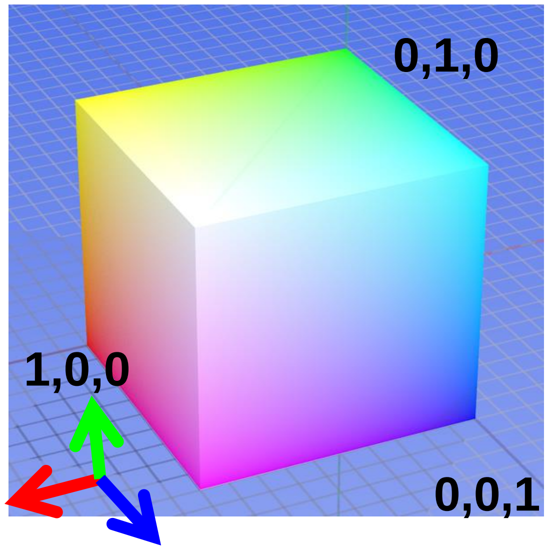
- Estimate RGB at each cell from neighboring values

http://en.wikipedia.org/wiki/Bayer_filter

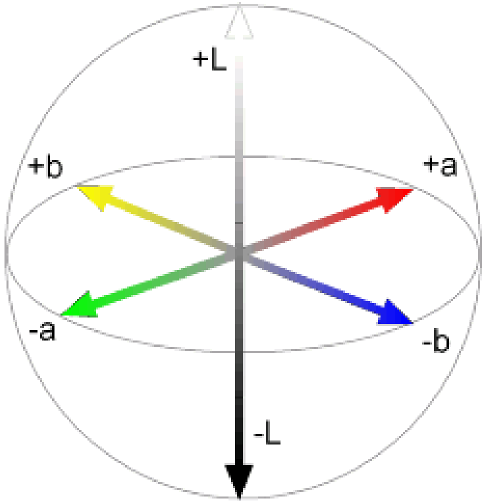
Slide by Steve Seitz

Color spaces

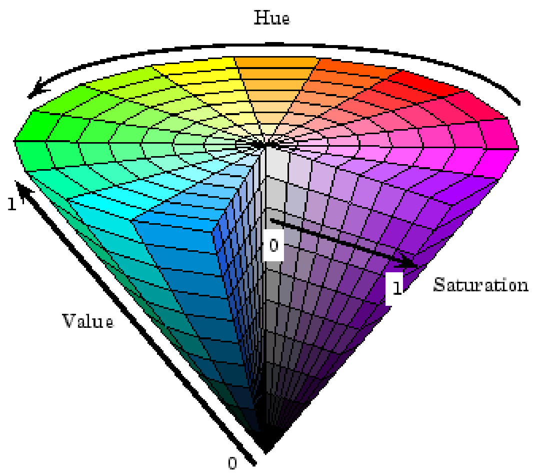
RGB



LAB



HSV



YCbCr

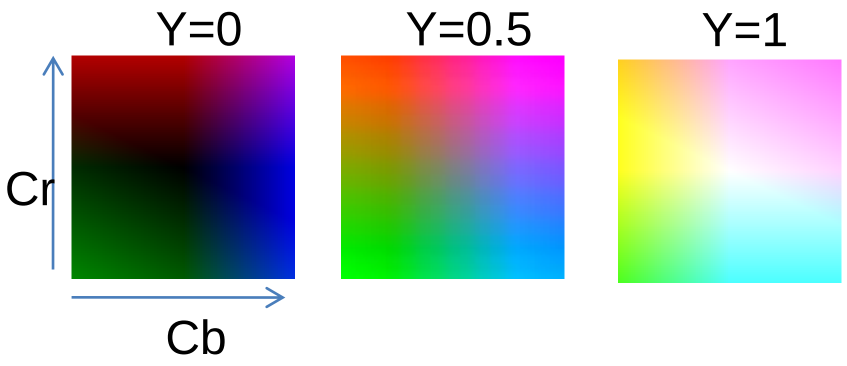
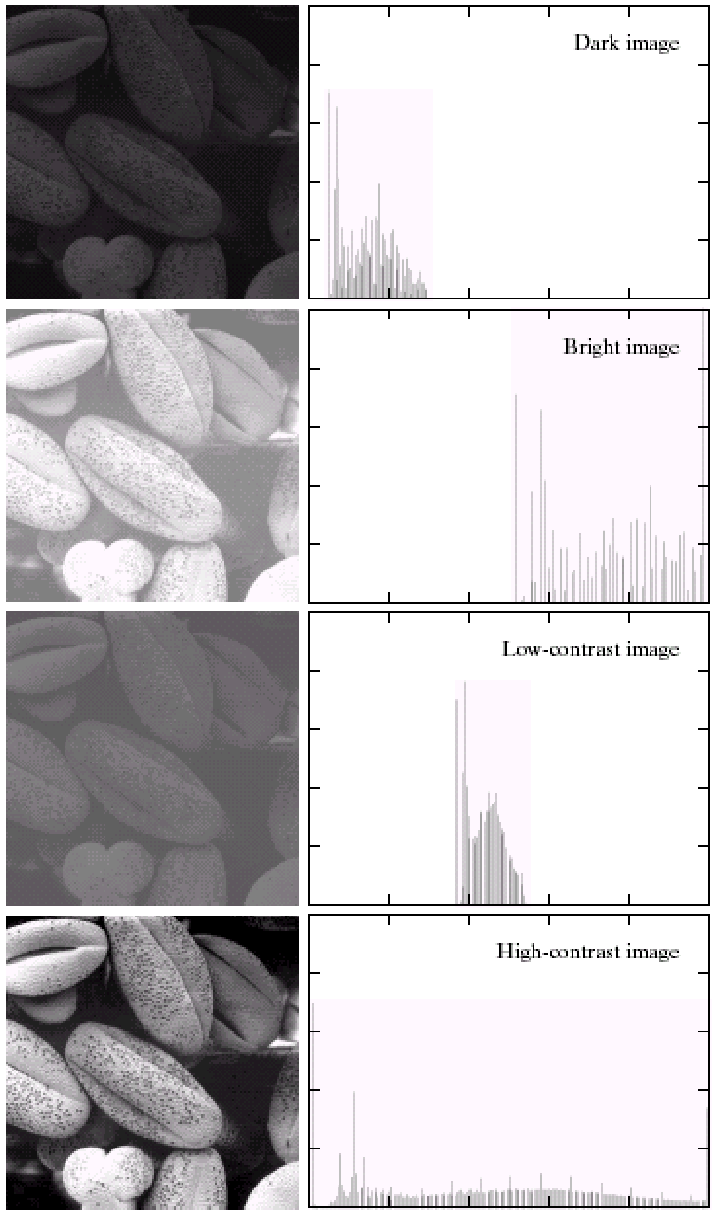


Image Histograms



a b

FIGURE 3.15 Four basic image types: dark, light, low contrast, high contrast, and their corresponding histograms. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

Contrast enhancement / balancing

- Gamma correction

$$\mathrm{v}_{\mathrm{new}} = v_{old}{}^{\gamma}$$

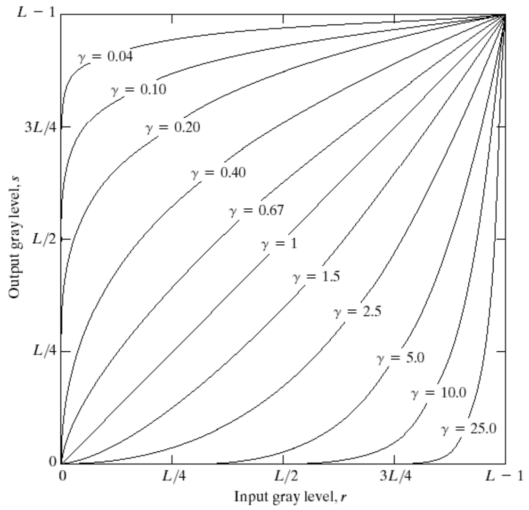
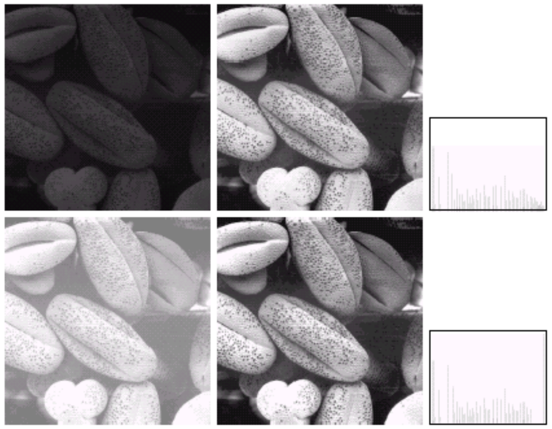
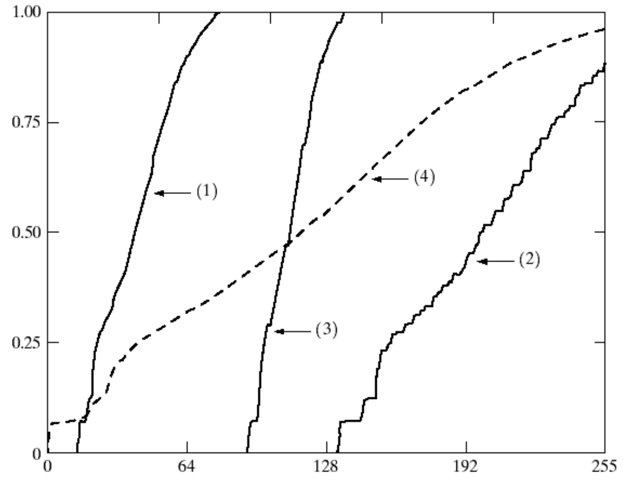


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

- Histogram equalization

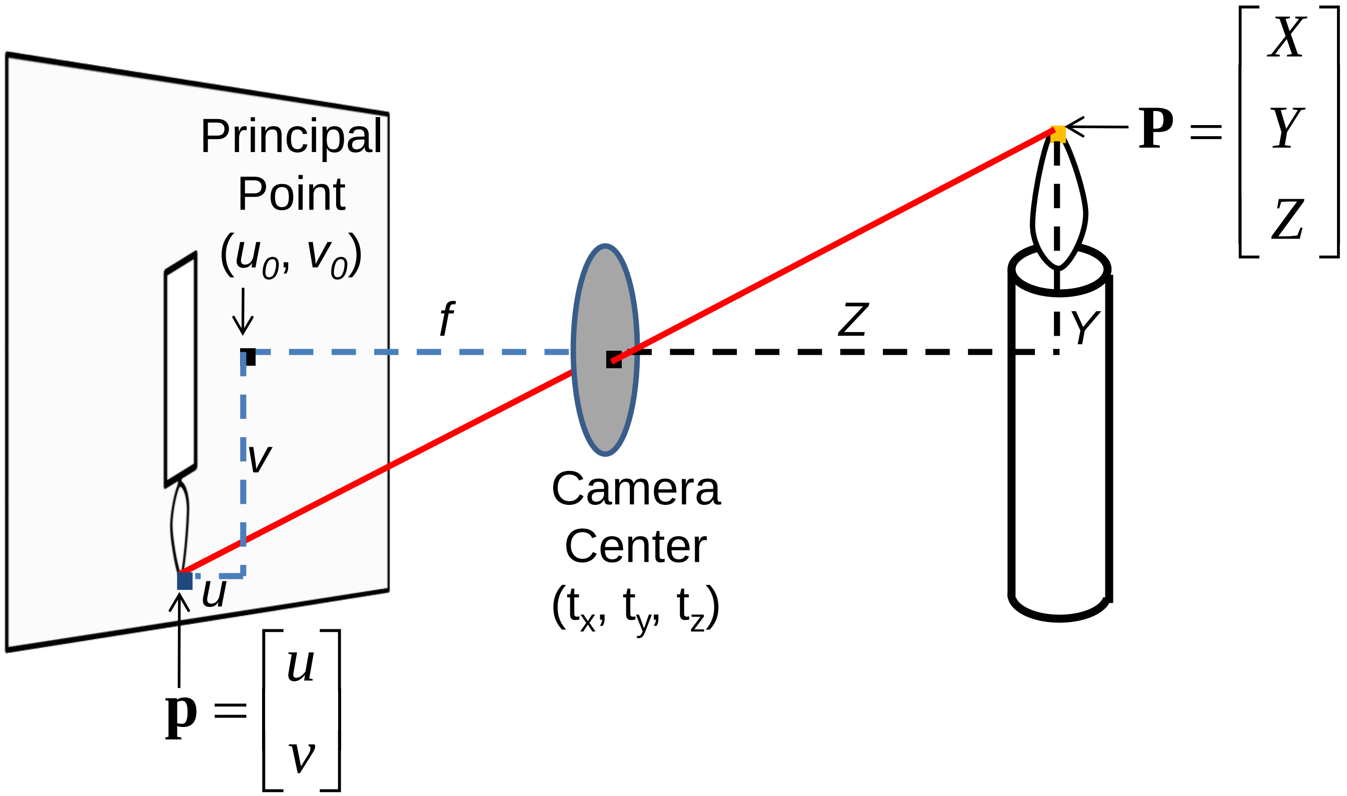




Cumulative Histograms

2. Camera Capture and Geometry

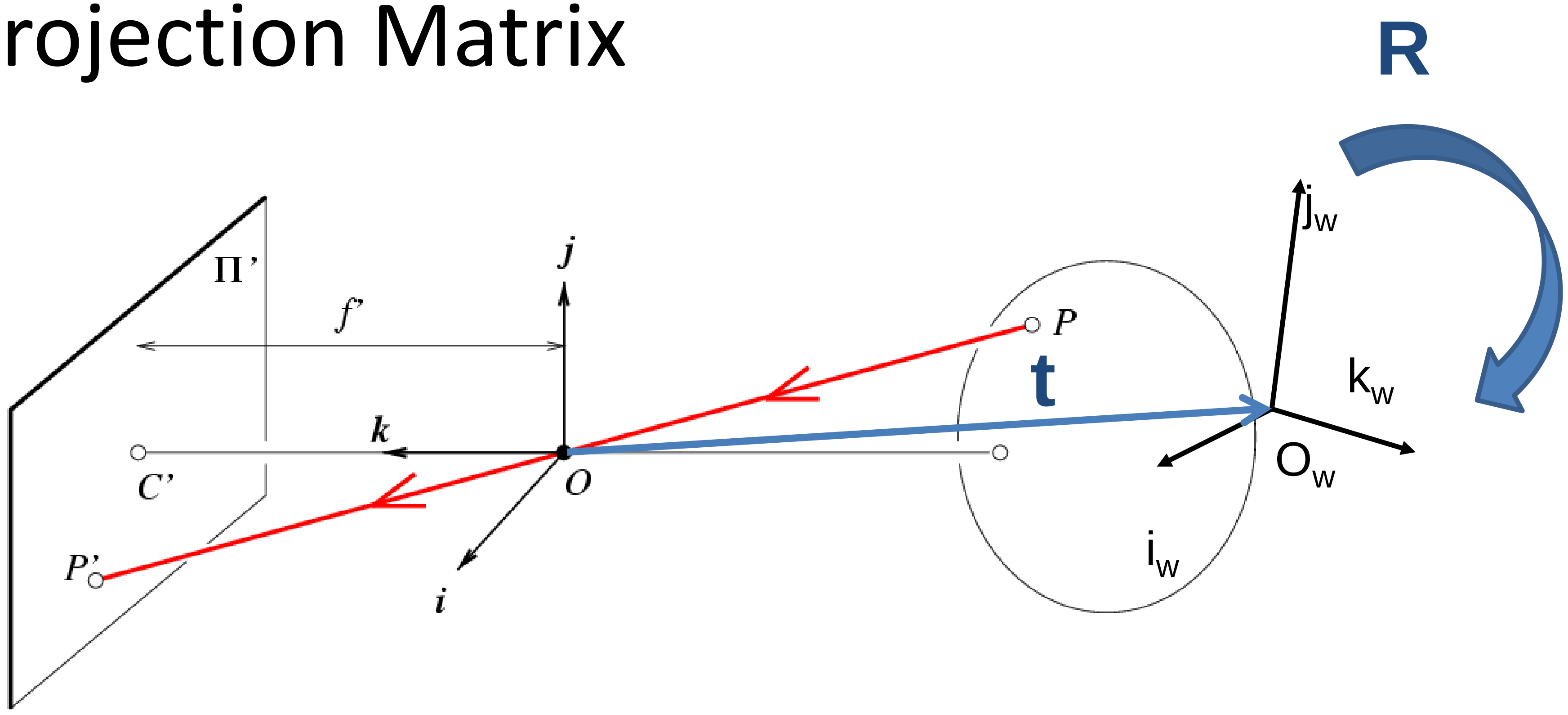
Pinhole Camera



Principal Point
(u_0, v_0)
f
v
u
Z
Y
Camera Center
(t_x, t_y, t_z)
$\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$
$\mathbf{p} = \begin{bmatrix} u \\ v \end{bmatrix}$

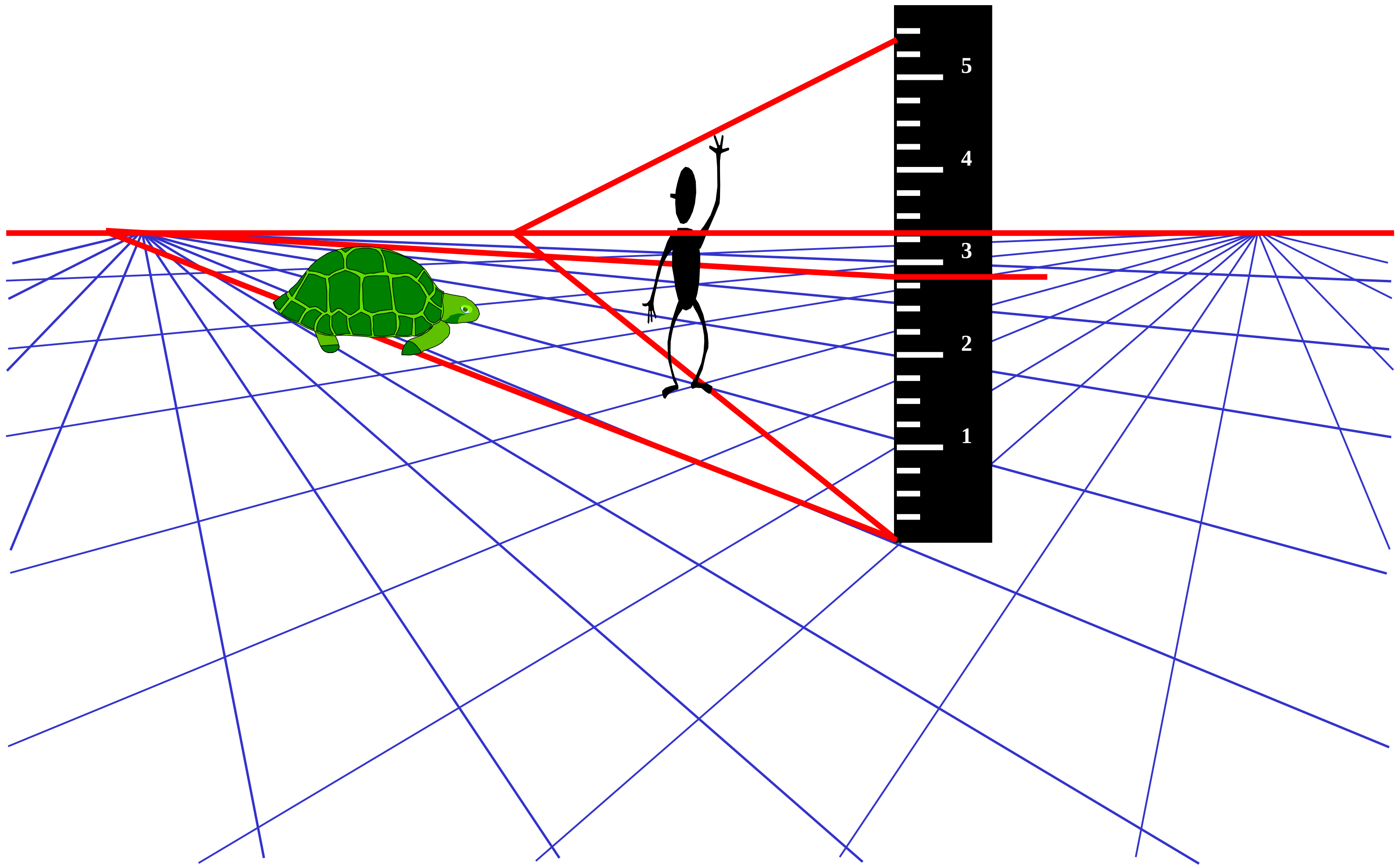
Useful figure to remember

Projection Matrix



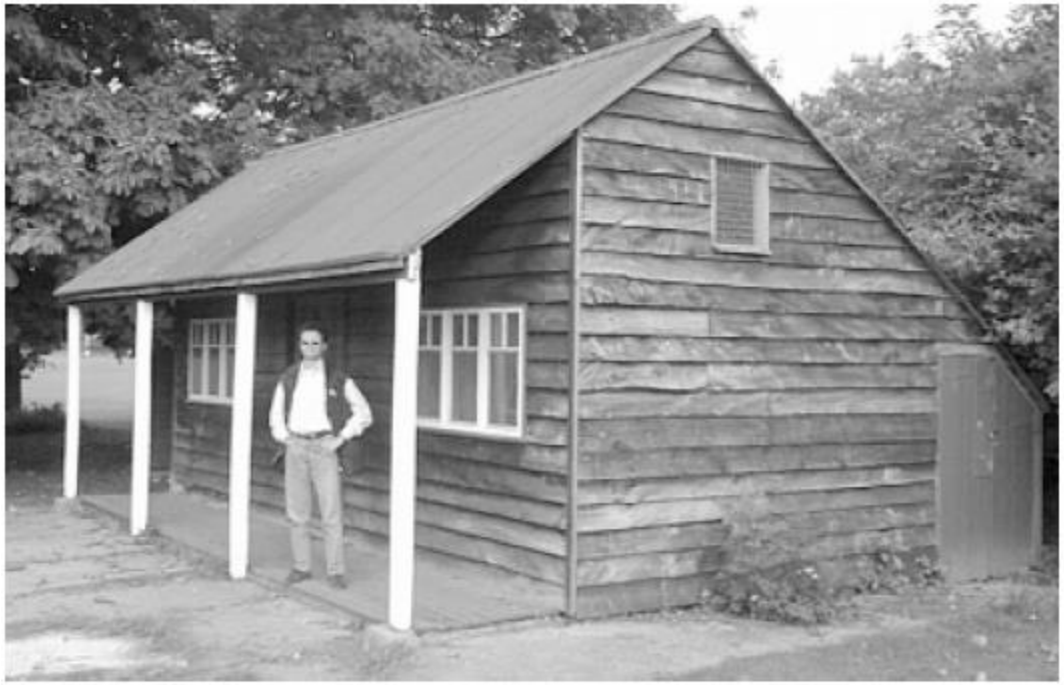
$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}f & s & u_0\\0 & \alpha f & v_0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_x\\r_{21} & r_{22} & r_{23} & t_y\\r_{31} & r_{32} & r_{33} & t_z\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

Single-view metrology



Assume the man is 6 ft tall.

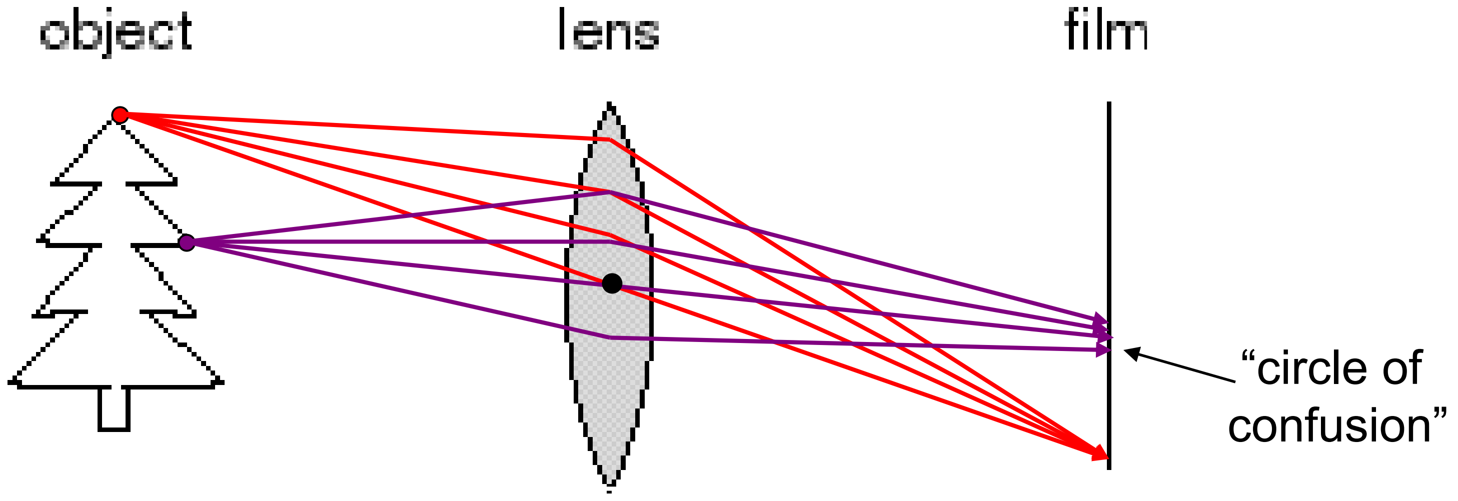
- What is the height of the building?
- How long is the right side of the building compared to the small window on the right side of the building?



cross-ratio

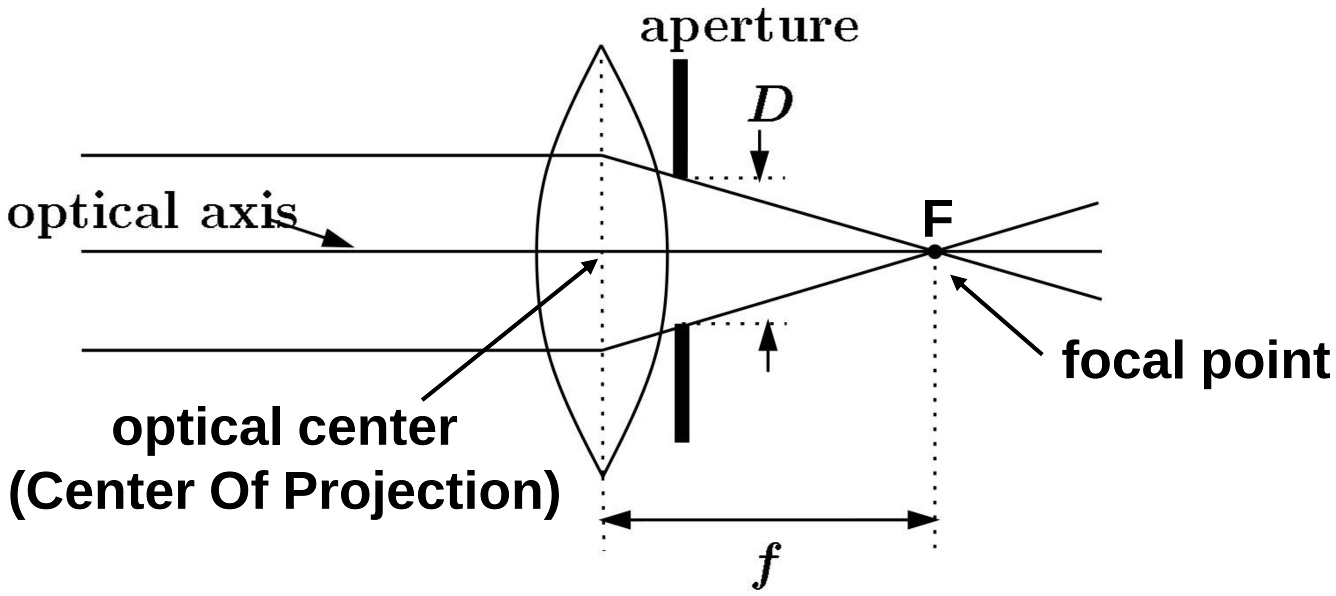
$$\frac{\|\mathbf{P}_3 - \mathbf{P}_1\| \, \|\mathbf{P}_4 - \mathbf{P}_2\|}{\|\mathbf{P}_3 - \mathbf{P}_2\| \, \|\mathbf{P}_4 - \mathbf{P}_1\|}$$

Adding a lens



- A lens focuses light onto the film
  - There is a specific distance at which objects are "in focus"
    - other points project to a "circle of confusion" in the image
  - Changing the shape of the lens changes this distance

Focal length, aperture, depth of field



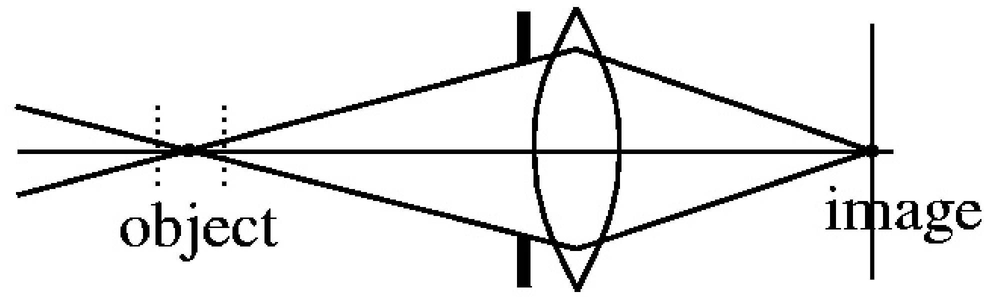
A lens focuses parallel rays onto a single focal point

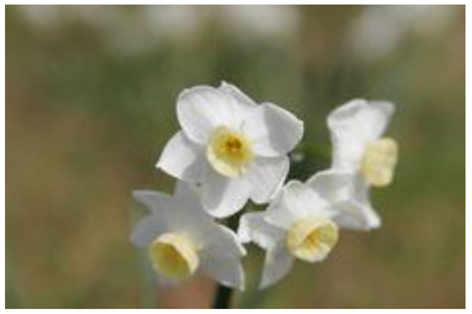
- focal point at a distance *f* beyond the plane of the lens
- Aperture of diameter D restricts the range of rays

Slide source: Seitz

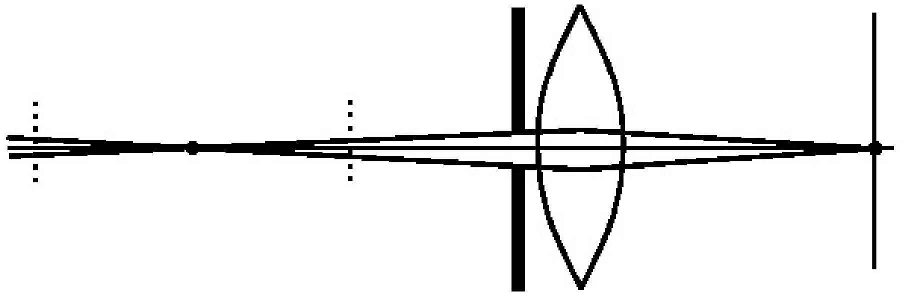
Slide source: Seitz

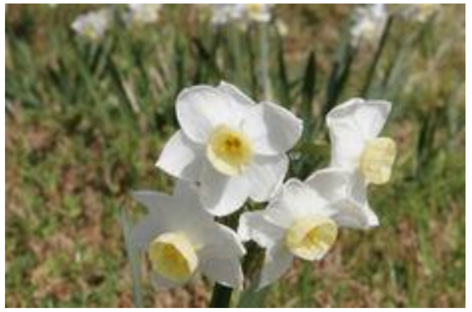
The aperture and depth of field





f / 5.6





f / 32

Main way to increase depth of field: Decrease aperture size

F-number (f/#) =focal_length / aperture_diameter

- E.g., f/16 means that the focal length is 16 times the diameter
- When you set the f-number of a camera, you are setting the aperture

Flower images from Wikipedia http://en.wikipedia.org/wiki/Depth_of_field

The Photographer's Great Compromise

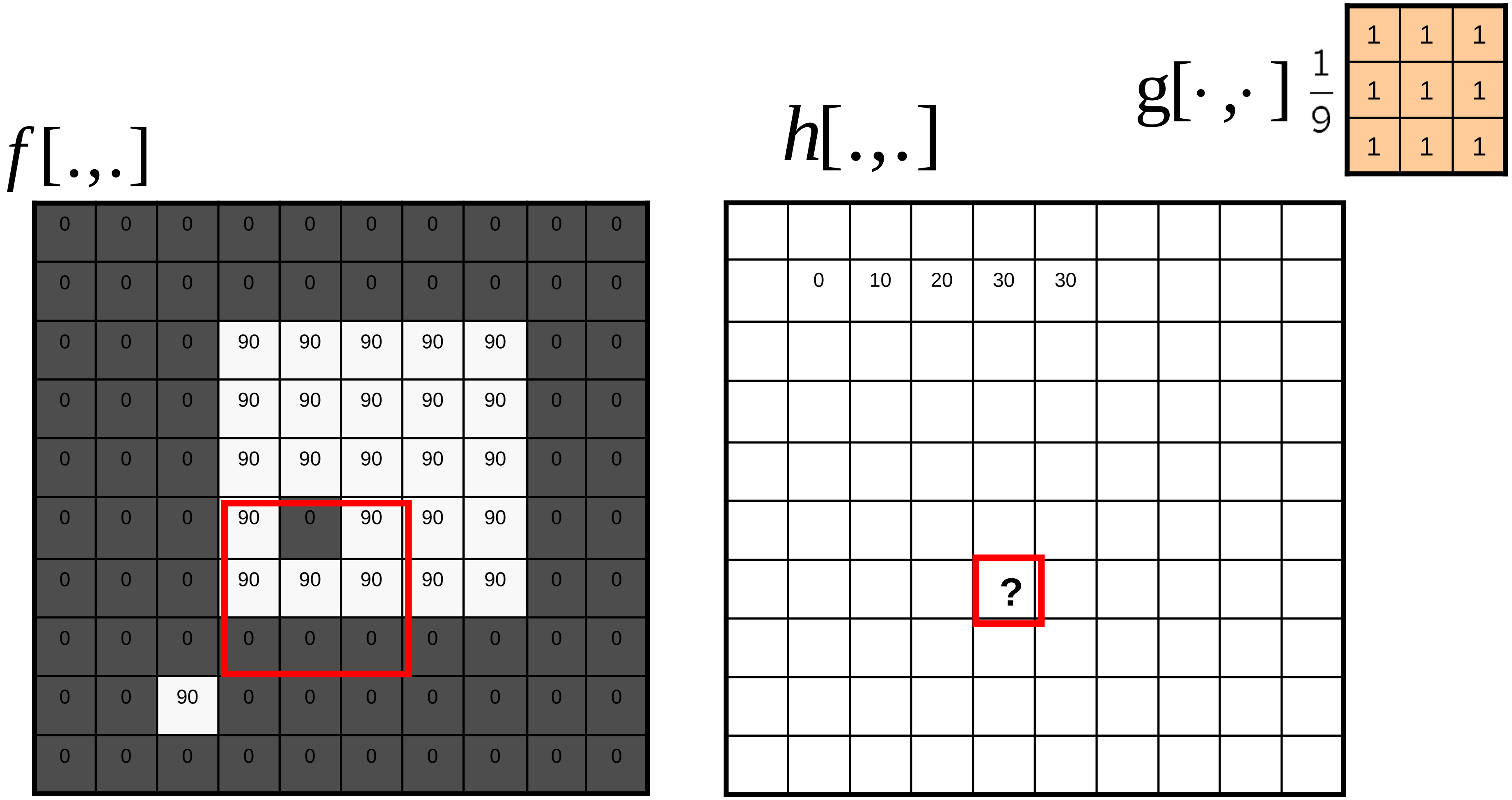
$$\begin{tikzcd}
\textbf{What we want} & \textbf{How we get it} & \textbf{Cost} \\
\text{More spatial resolution} \arrow[r] \arrow[dddr] & \text{Increase focal length} & \text{Light, FOV} \\
 & \text{Decrease focal length} & \text{DOF} \\
\text{Broader field of view} \arrow[ur] & & \\
\text{More depth of field} \arrow[r] & \text{Decrease aperture} & \text{Light} \\
 & \text{Increase aperture} & \text{DOF} \\
\text{More temporal resolution} \arrow[r] & \text{Shorten exposure} & \text{Light} \\
 & \text{Lengthen exposure} & \text{Temporal Res} \\
\text{More light} \arrow[uuur] \arrow[ur] & &
\end{tikzcd}$$

3. Linear filtering

- Can think of filtering as
  - A function in the spatial domain (e.g., compute average of each 3x3 window)
  - Template matching
  - Modifying the frequency of the image

Filtering in spatial domain

- Slide filter over image and take dot product at each position



$f[.,.]$

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

$h[.,.]$

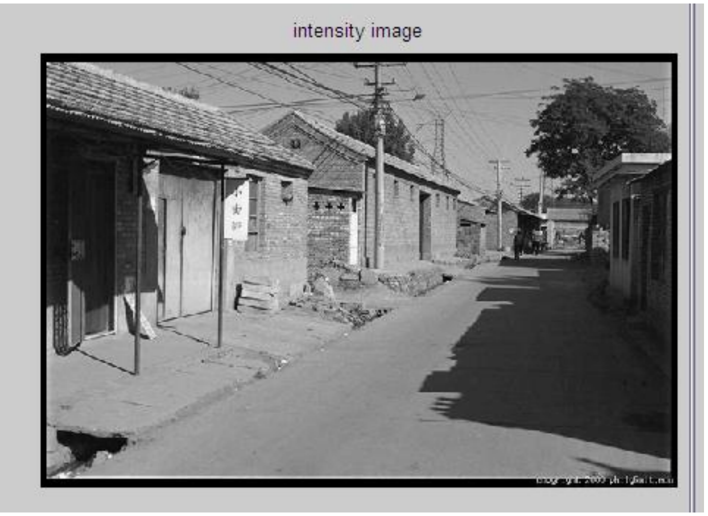
	0	10	20	30	30				
				?					

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

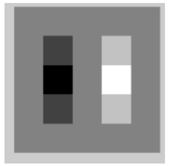
1	1	1
1	1	1
1	1	1

Filtering in spatial domain

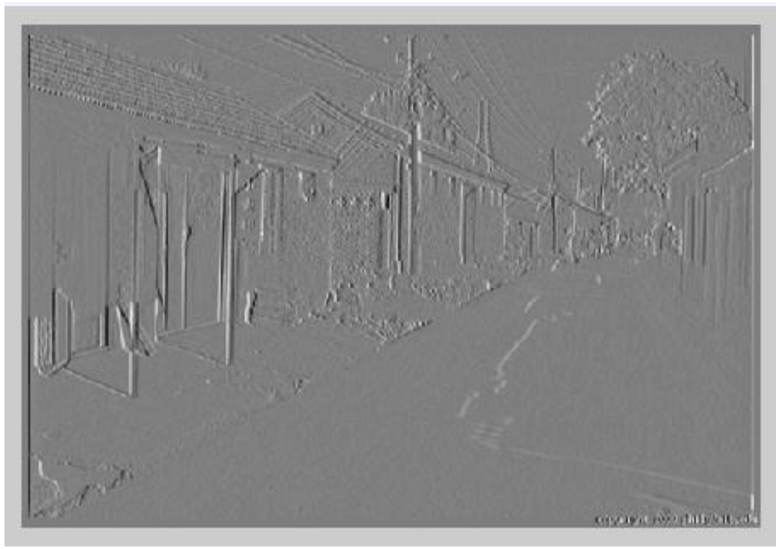
1	0	-1
2	0	-2
1	0	-1



$*$



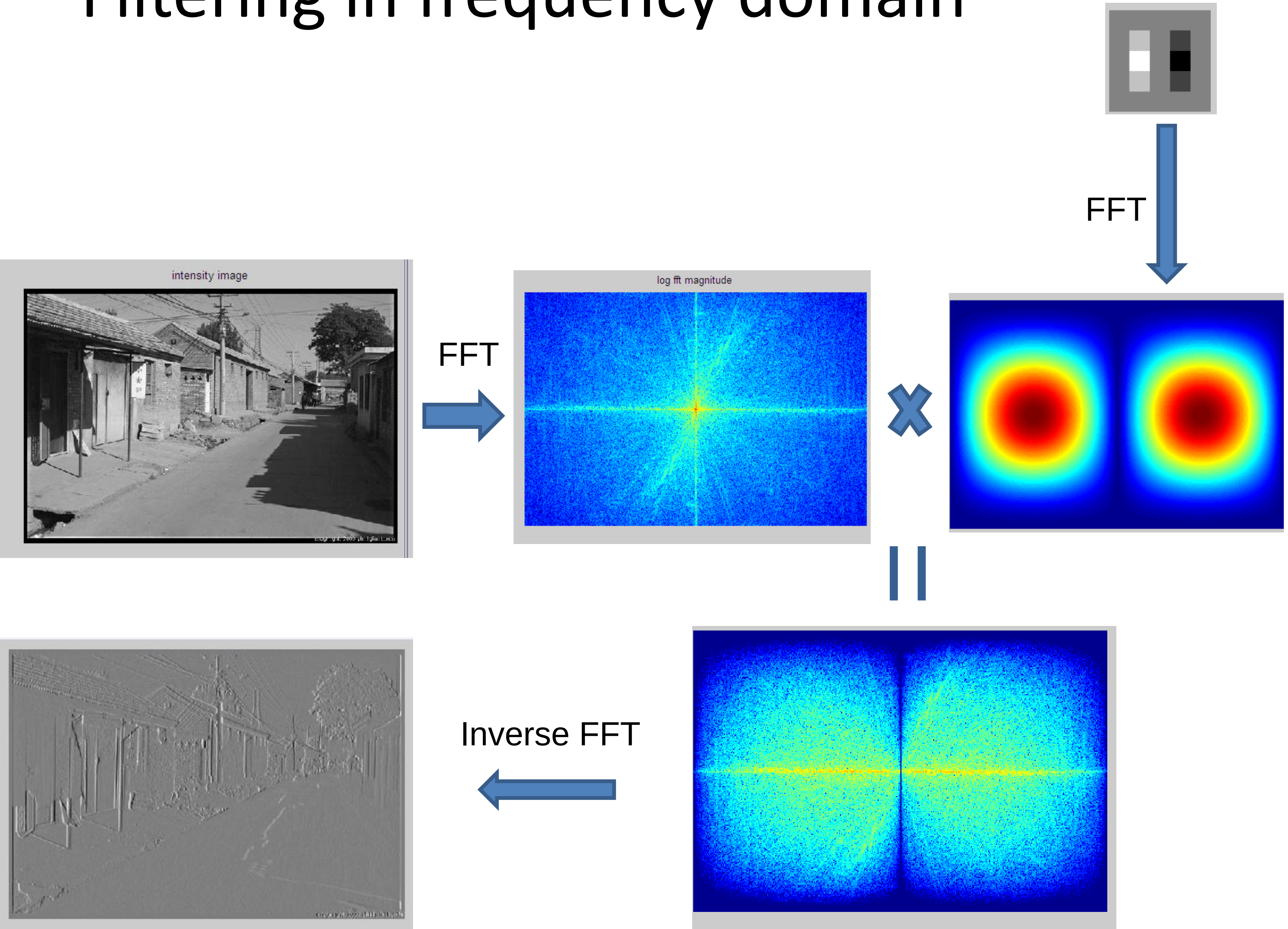
$=$



Filtering in frequency domain

- Can be faster than filtering in spatial domain (for large filters)
- Can help understand effect of filter
- Algorithm:
    1. Convert image and filter to fft (fft2 in matlab)
    2. Pointwise-multiply ffts
    3. Convert result to spatial domain with ifft2

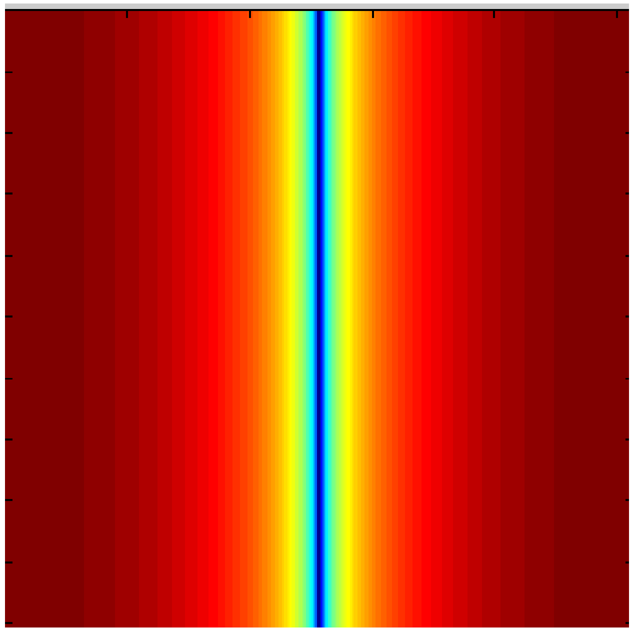
Filtering in frequency domain

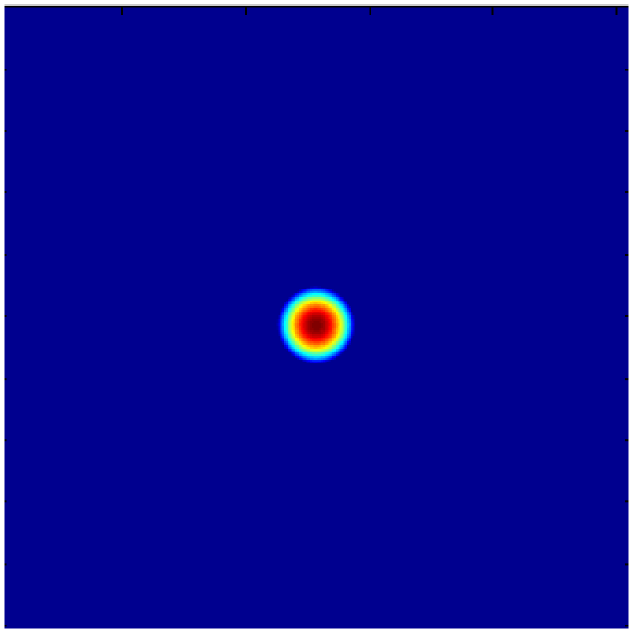


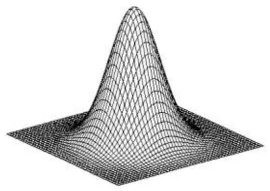
Filtering in frequency domain

- Linear filters for basic processing
  - Edge filter (high-pass)
  - Gaussian filter (low-pass)

[-1 1]

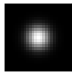

FFT of Gradient Filter

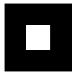

FFT of Gaussian

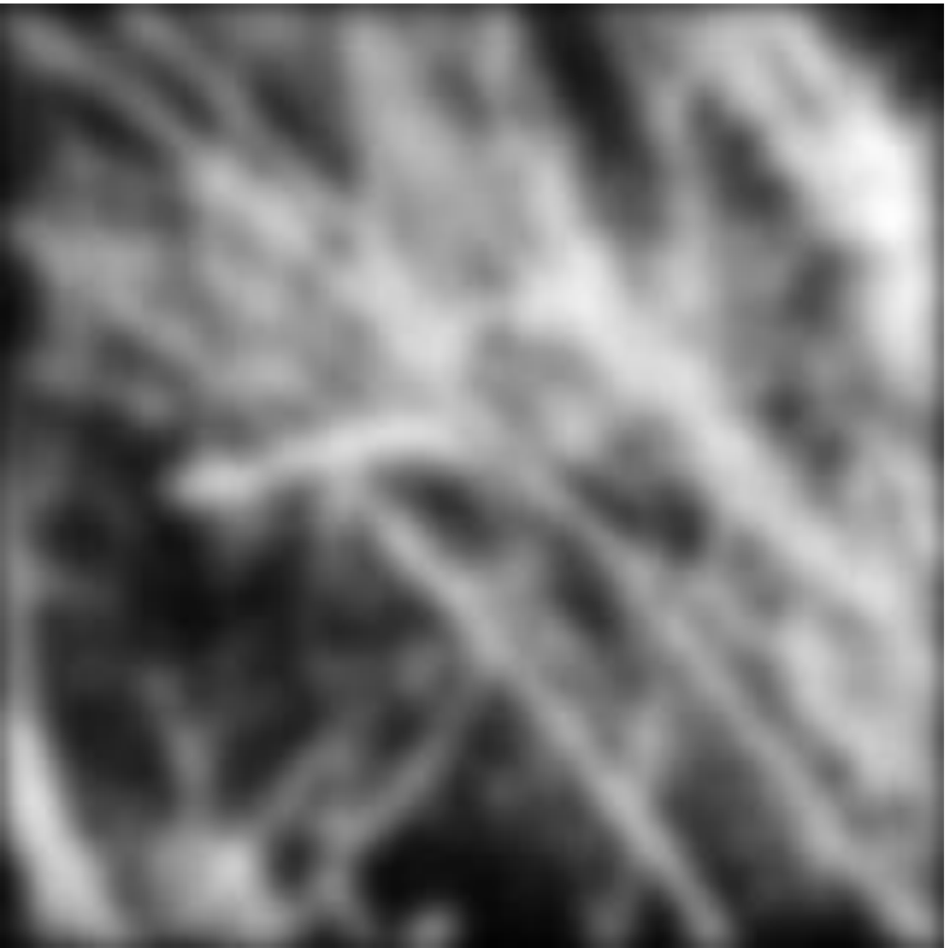

Gaussian

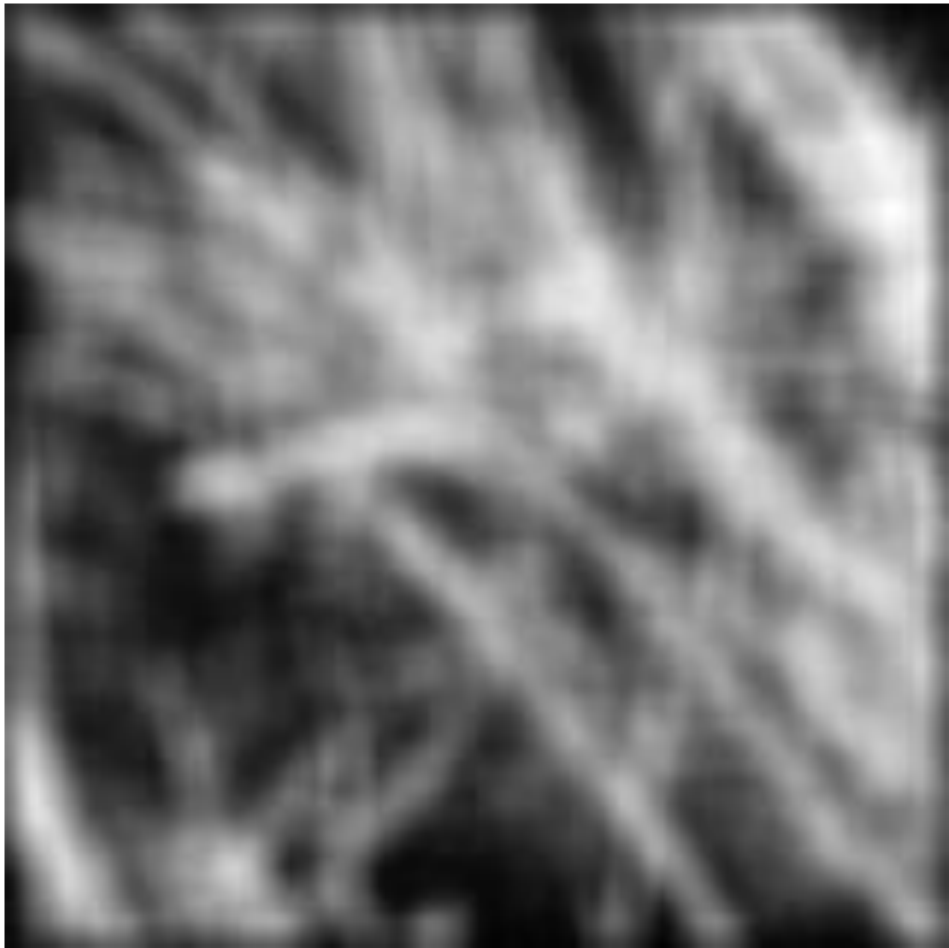
Filtering

Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

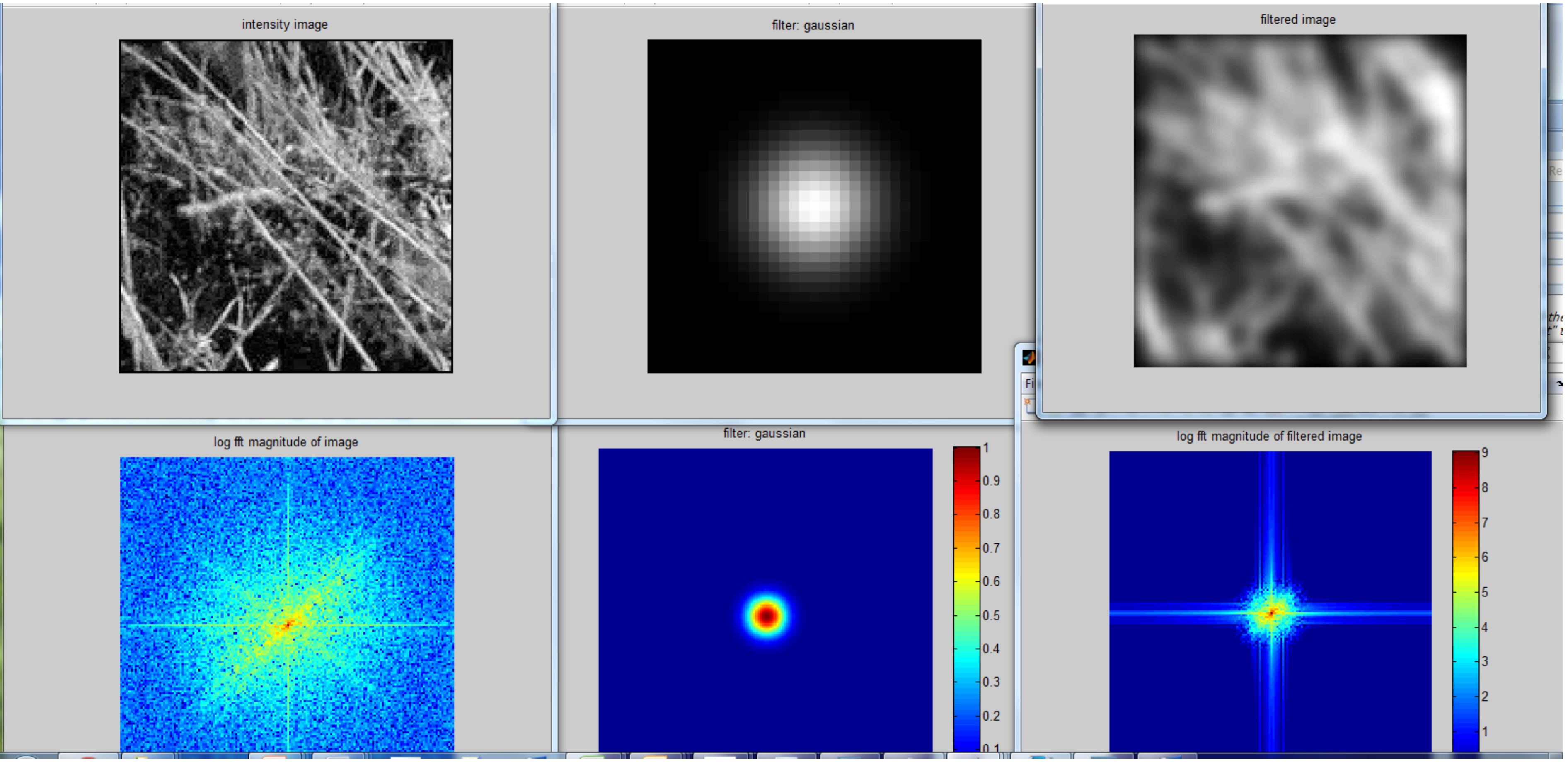
Gaussian 

Box filter 

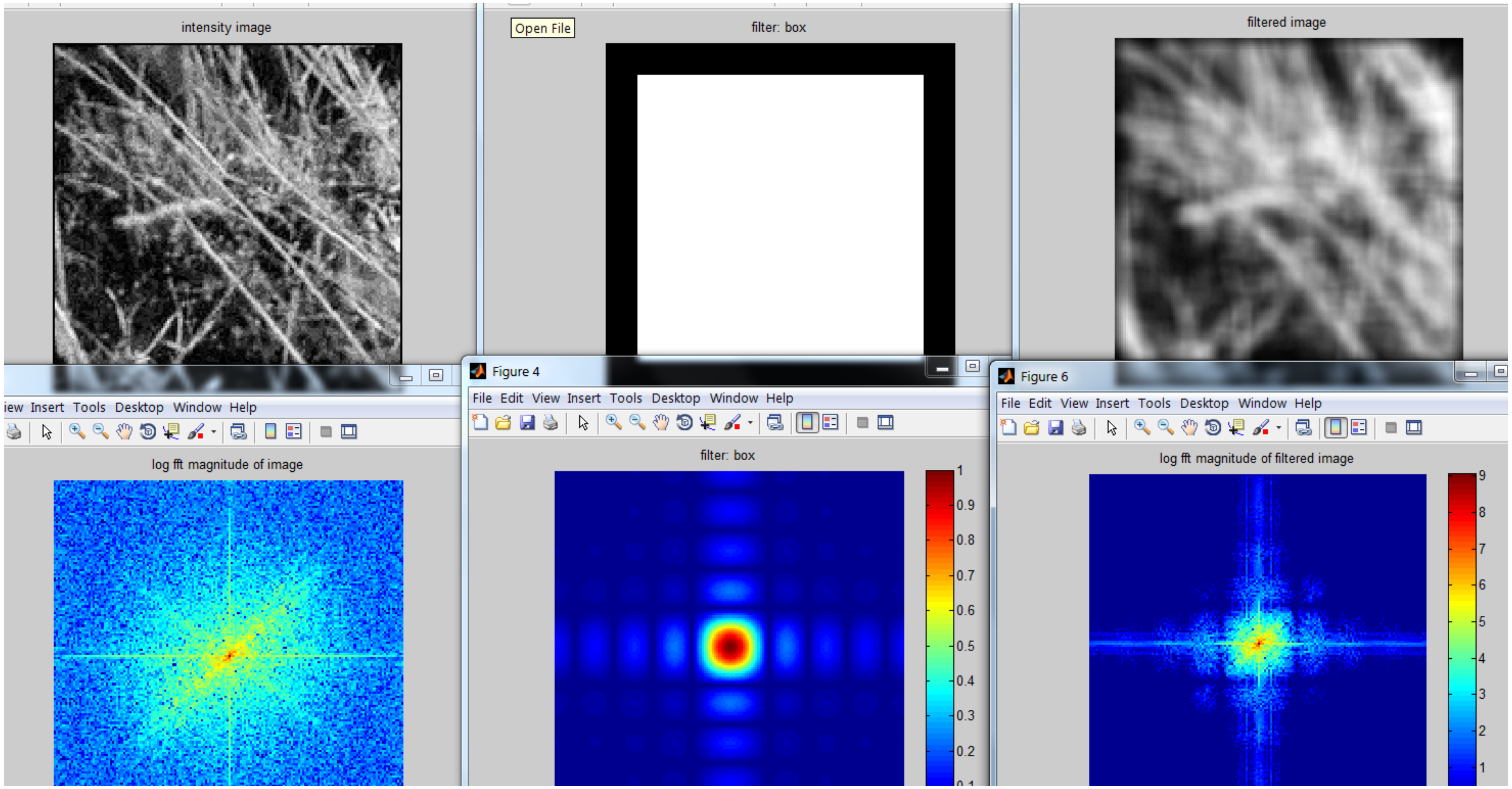




Gaussian


intensity image
filter: gaussian
filtered image
log fft magnitude of image
filter: gaussian
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
log fft magnitude of filtered image
9
8
7
6
5
4
3
2
1

Box Filter


intensity image
Open File
filter: box
filtered image
Figure 4
Figure 6
File Edit View Insert Tools Desktop Window Help
log fft magnitude of image
filter: box
log fft magnitude of filtered image
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
9
8
7
6
5
4
3
2
1

Question

1. Use filtering to find pixels that have at least three white pixels among the 8 surrounding pixels (assume a binary image)

2. Write down a filter that will compute the following function

```
fil(y,x) = -0.5*im(y+1,x)+im(y,x)-0.5*im(y-1,x)
                                   for each x, y
```

Question

Filtering Operator

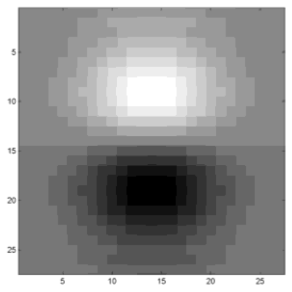
3. Fill in the blanks:

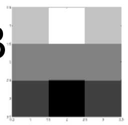
a) _ = D * B

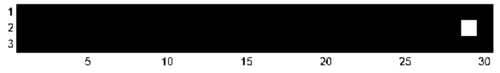
b) A = _ * _

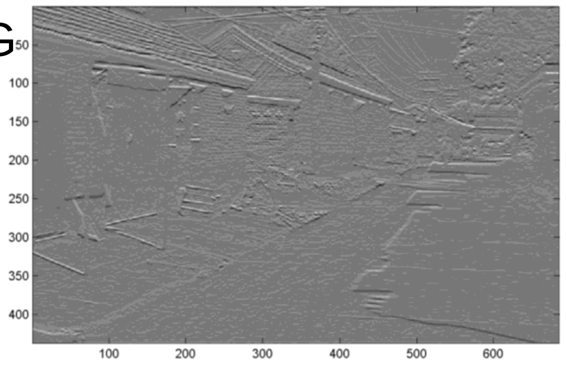
c) F = D * _

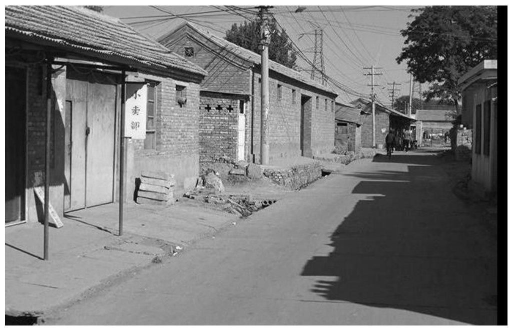
d) _ = D * C

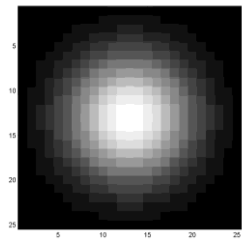
A


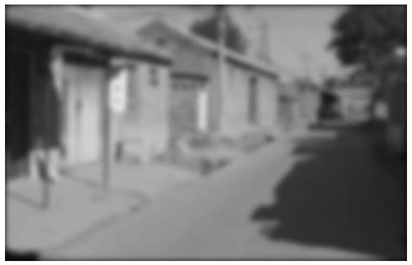
B


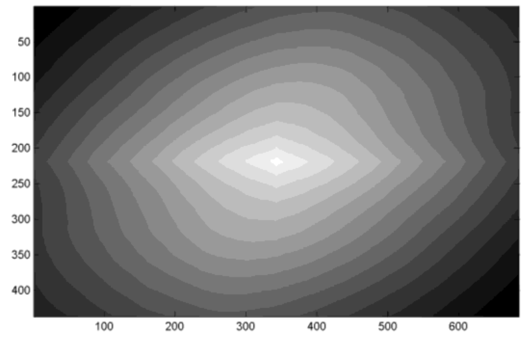
E


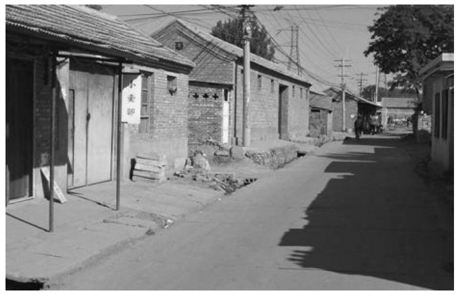
G


F


C


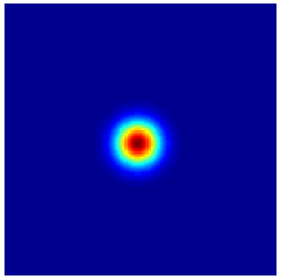
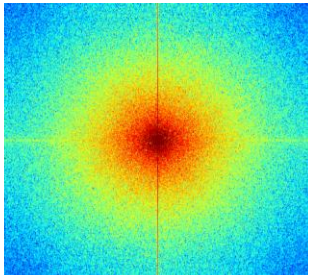
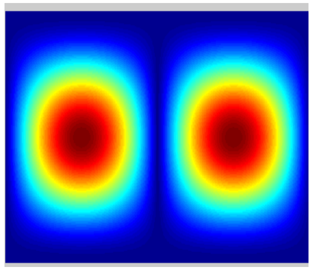
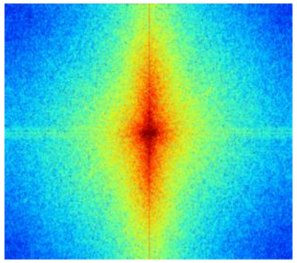
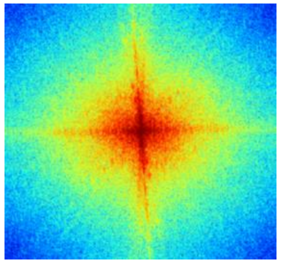
H


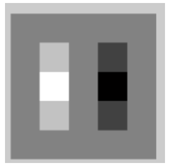
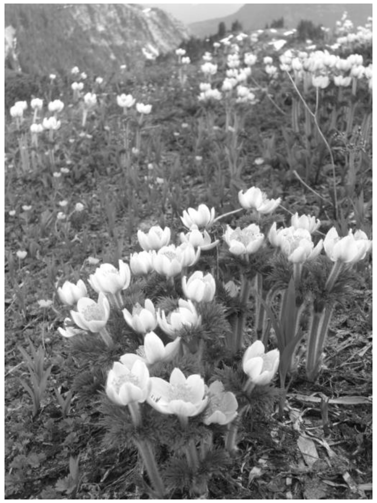
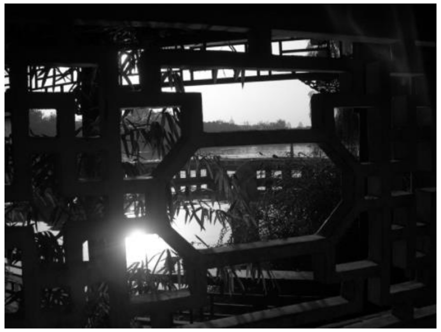
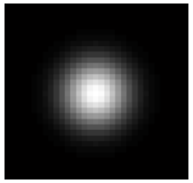
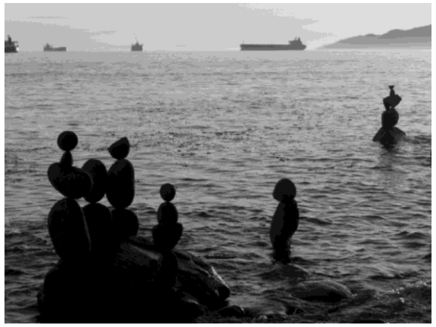
I


D


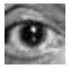
Question

1. Match the spatial domain image to the Fourier magnitude image

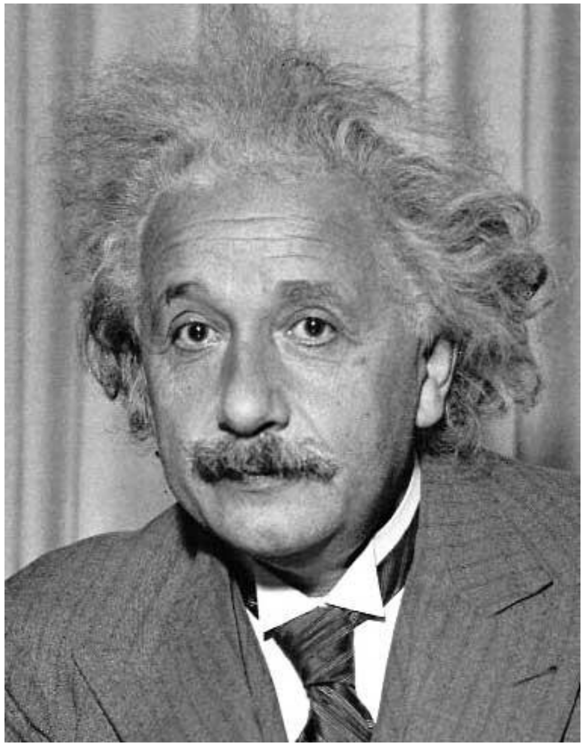
1

2

3

4

5


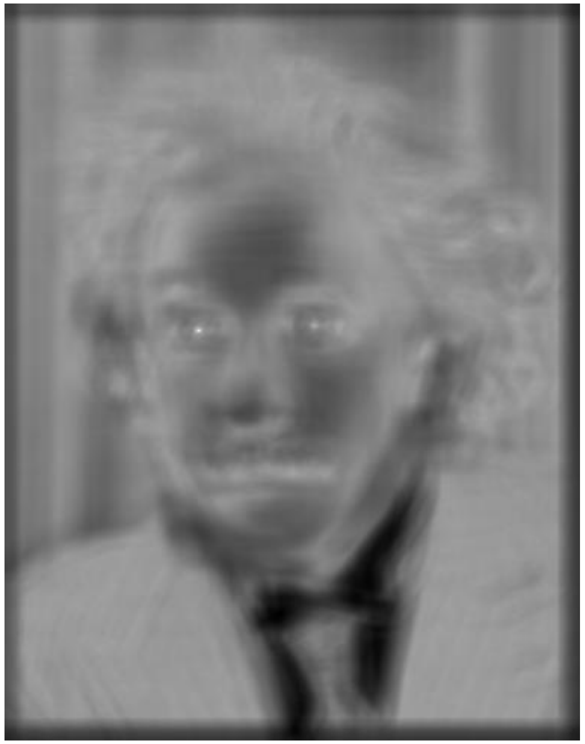
A

B

C

D

E


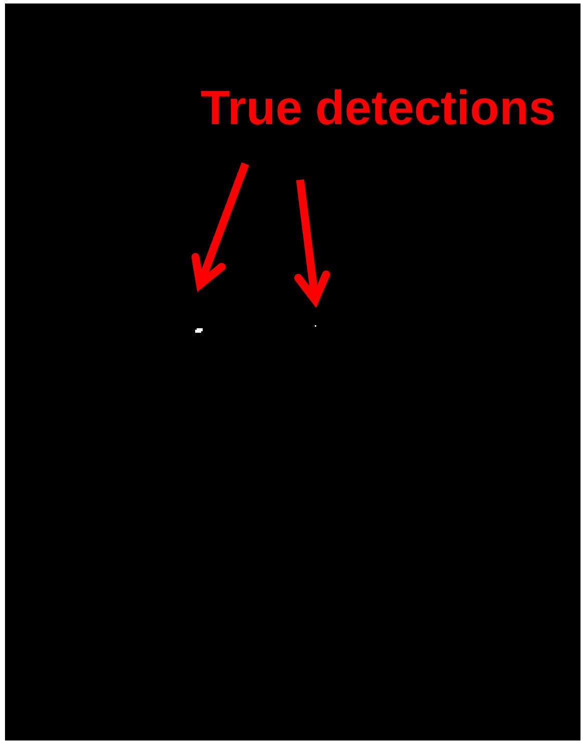
Matching with filters

- Goal: find  in image
- Method 2: SSD

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$


Input


1- sqrt(SSD)



Thresholded Image

Matching with filters

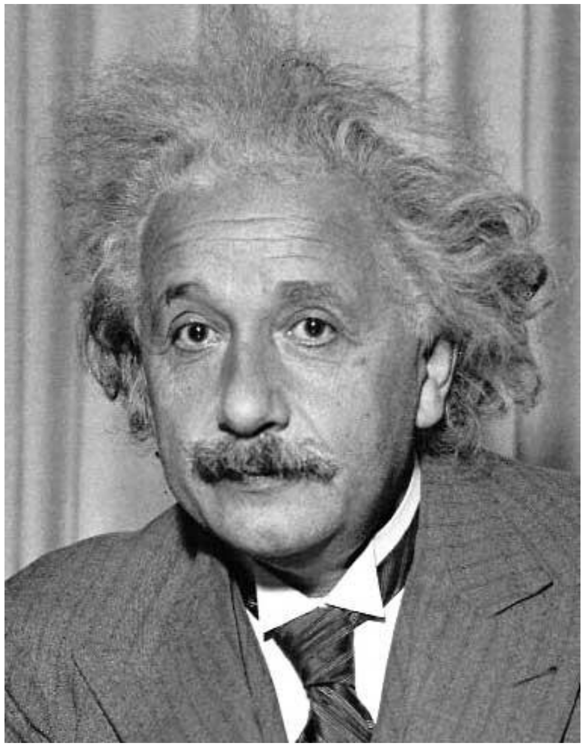
- Goal: find in image
- Method 3: Normalized cross-correlation

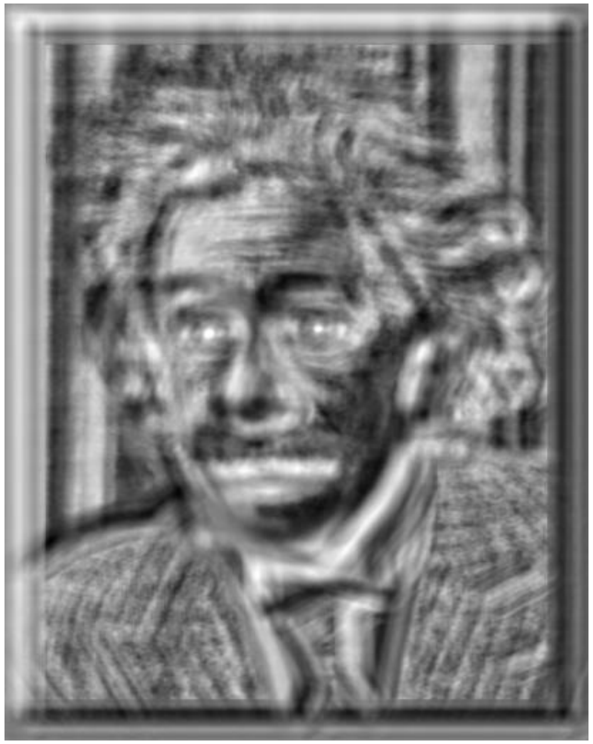
$$h[m,n] = \frac{\sum_{k,l} (g[k,l] - \overset{\text{mean template}}{\overset{\downarrow}{\bar{g}}})(f[m-k,n-l] - \overset{\text{mean image patch}}{\overset{\downarrow}{\bar{f}_{m,n}}})}{\left(\sum_{k,l} (g[k,l] - \bar{g})^2 \sum_{k,l} (f[m-k,n-l] - \bar{f}_{m,n})^2 \right)^{0.5}}$$

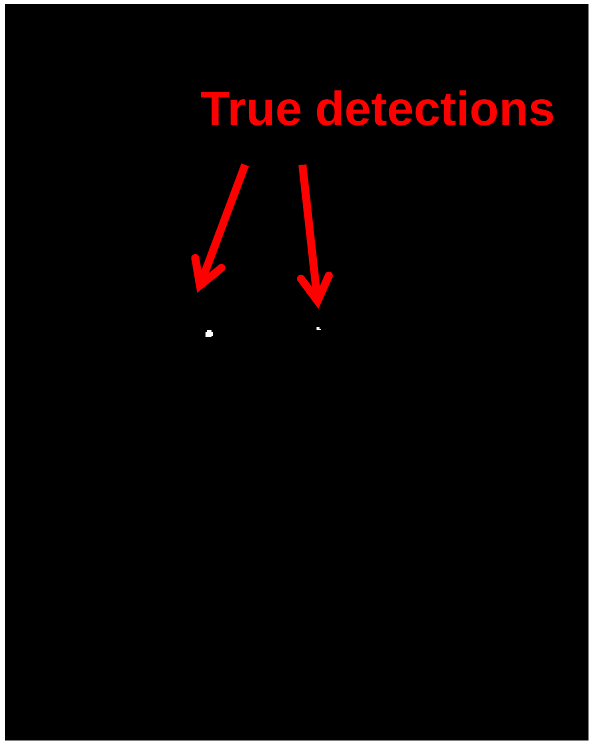
Matlab: `normxcorr2(template, im)`

Matching with filters

- Goal: find in image
- Method 3: Normalized cross-correlation

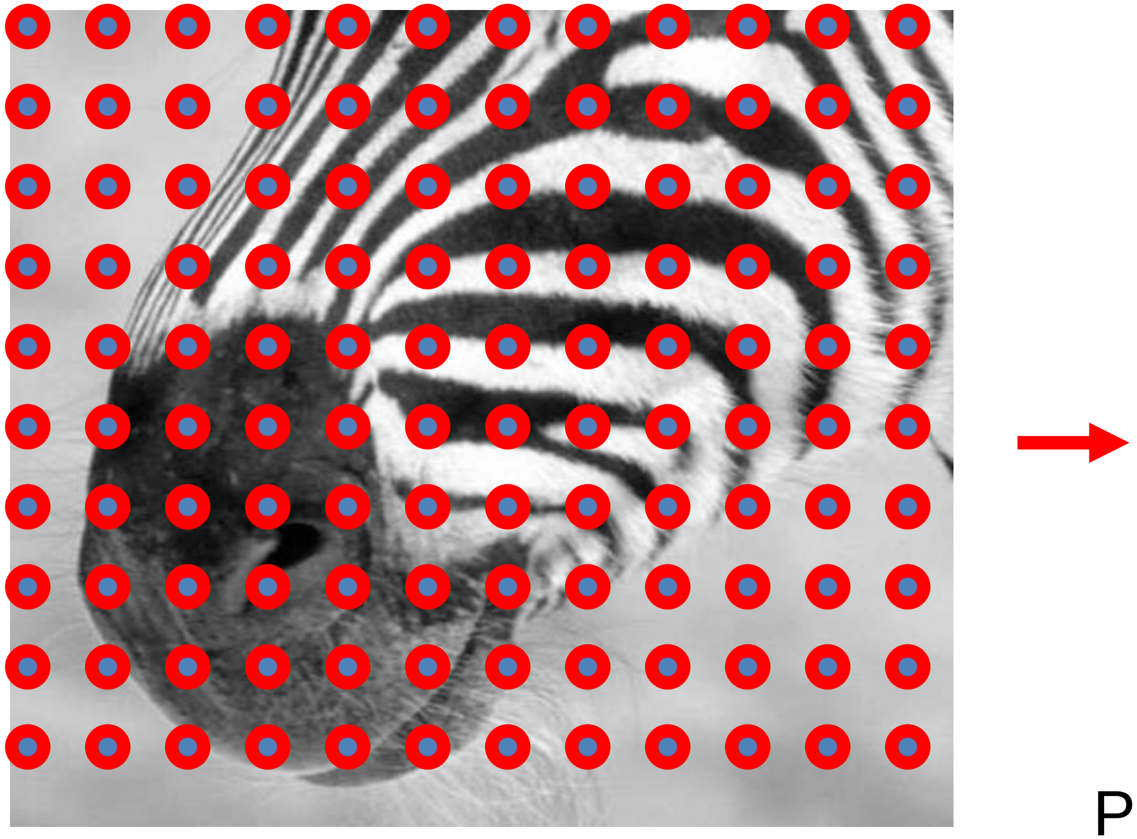

Input


Normalized X-Correlation



Thresholded Image

Subsampling by a factor of 2

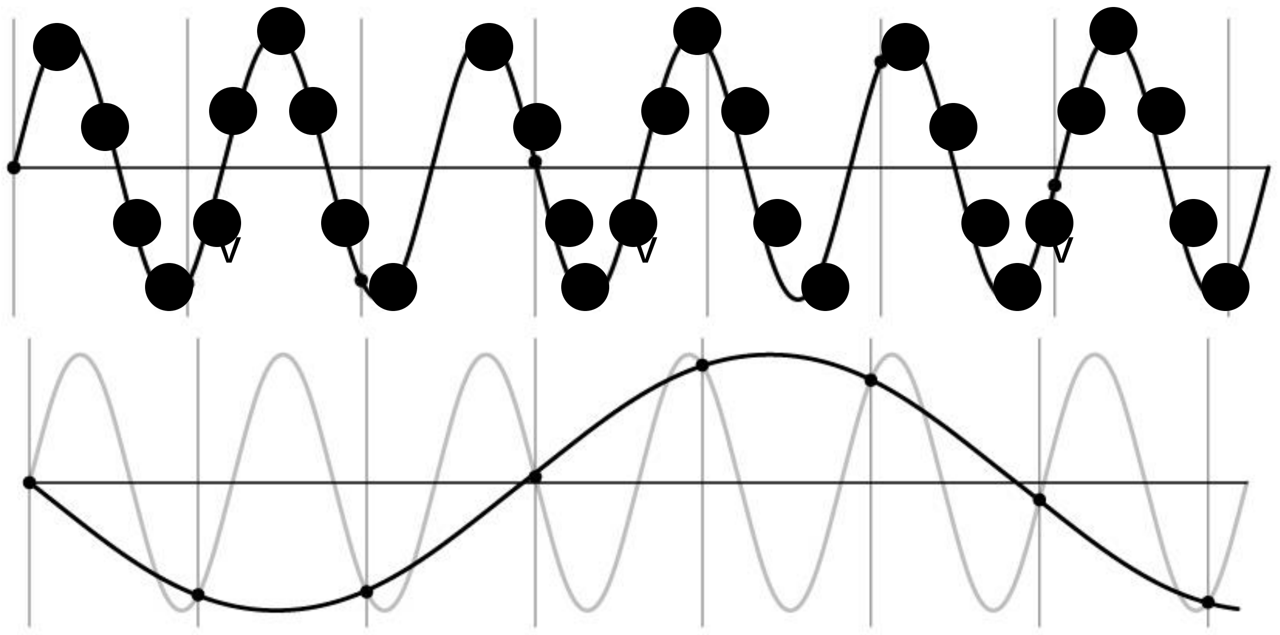


Throw away every other row and column to create a 1/2 size image

Problem: This approach causes "aliasing"

Nyquist-Shannon Sampling Theorem

- When sampling a signal at discrete intervals, the sampling frequency must be $\geq 2 \times f_{max}$
- f_{max} = max frequency of the input signal
- This will allows to reconstruct the original perfectly from the sampled version



Algorithm for downsampling by factor of 2

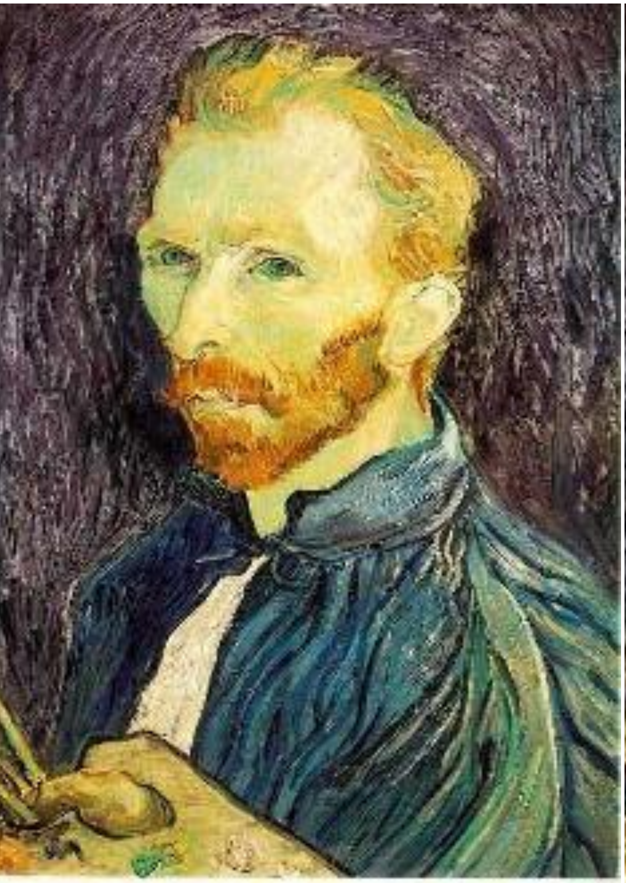
1. Start with image(h, w)
2. Apply low-pass filter

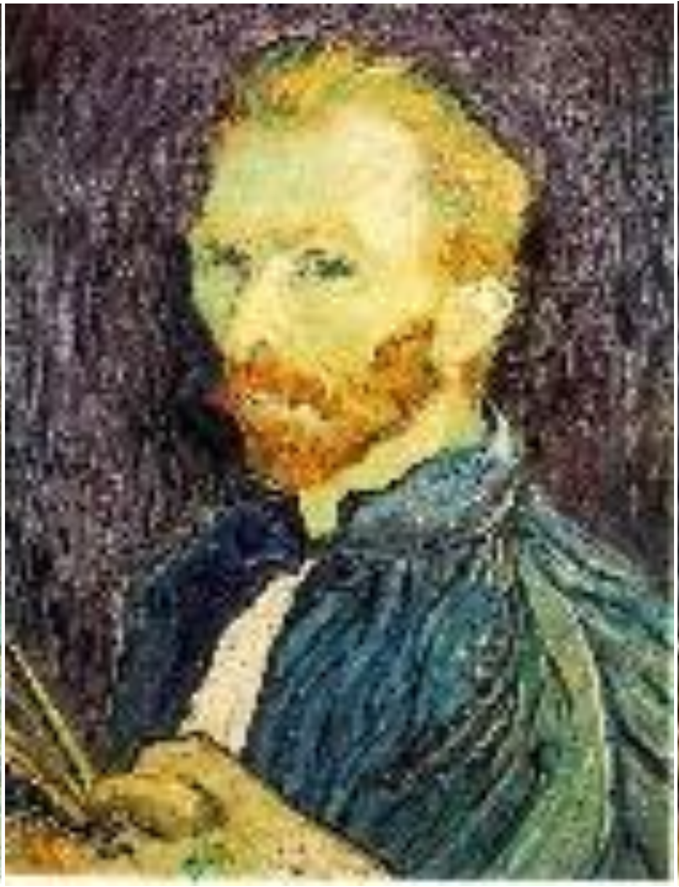
   ```
   im_blur = imfilter(image, fspecial(‘gaussian’, 13, 2))
   ```

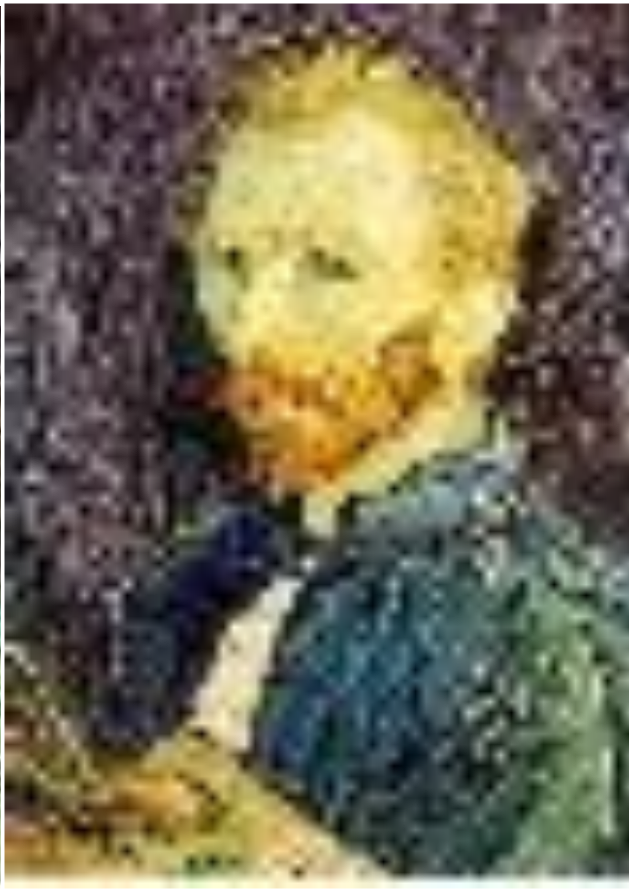
3. Sample every other pixel

   ```
   im_small = im_blur(1:2:end, 1:2:end);
   ```

Subsampling without pre-filtering

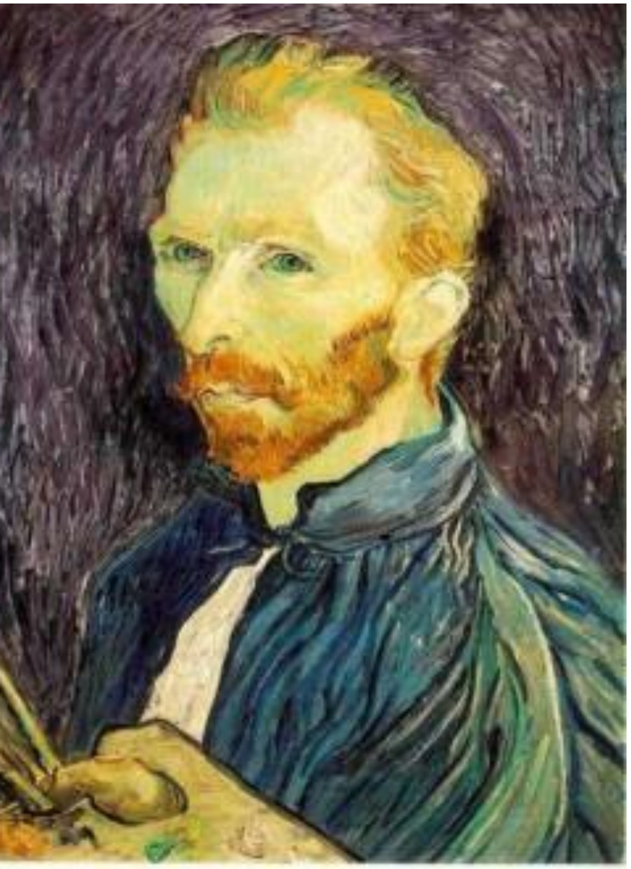

1/2

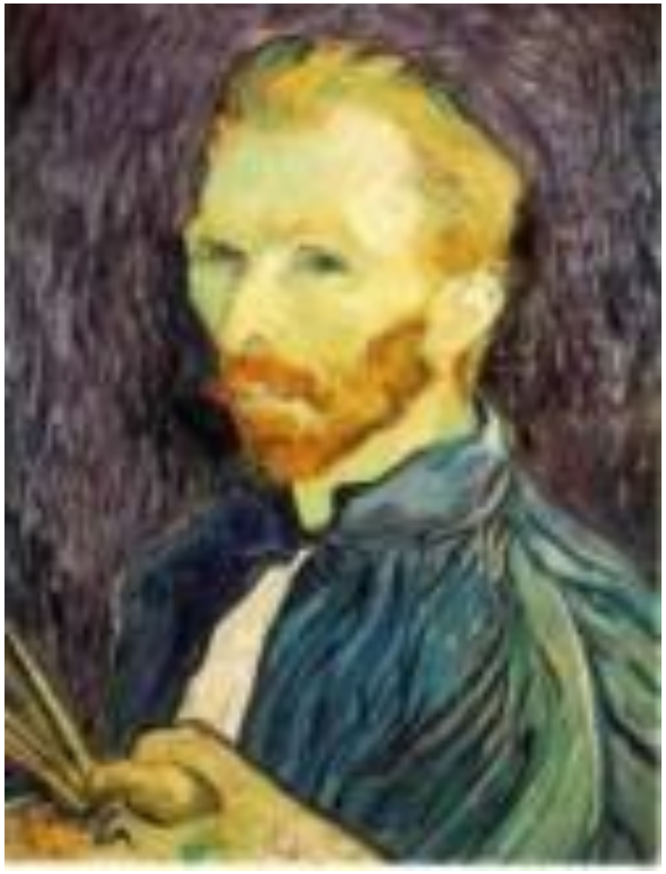

1/4 (2x zoom)

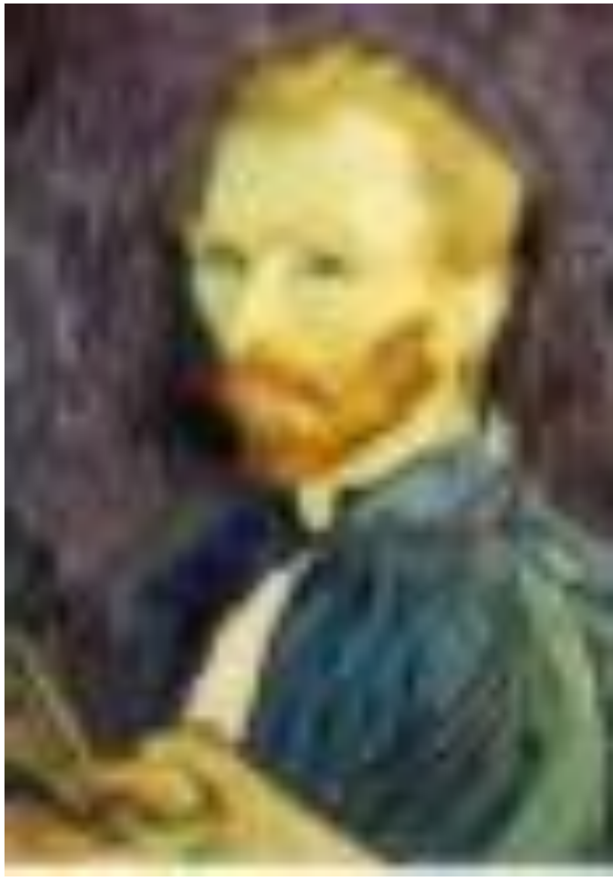

1/8 (4x zoom)

Slide by Steve Seitz

Subsampling with Gaussian pre-filtering


Gaussian 1/2


G 1/4


G 1/8

Slide by Steve Seitz

Sampling

Why does a lower resolution image still make sense to us? What do we lose?

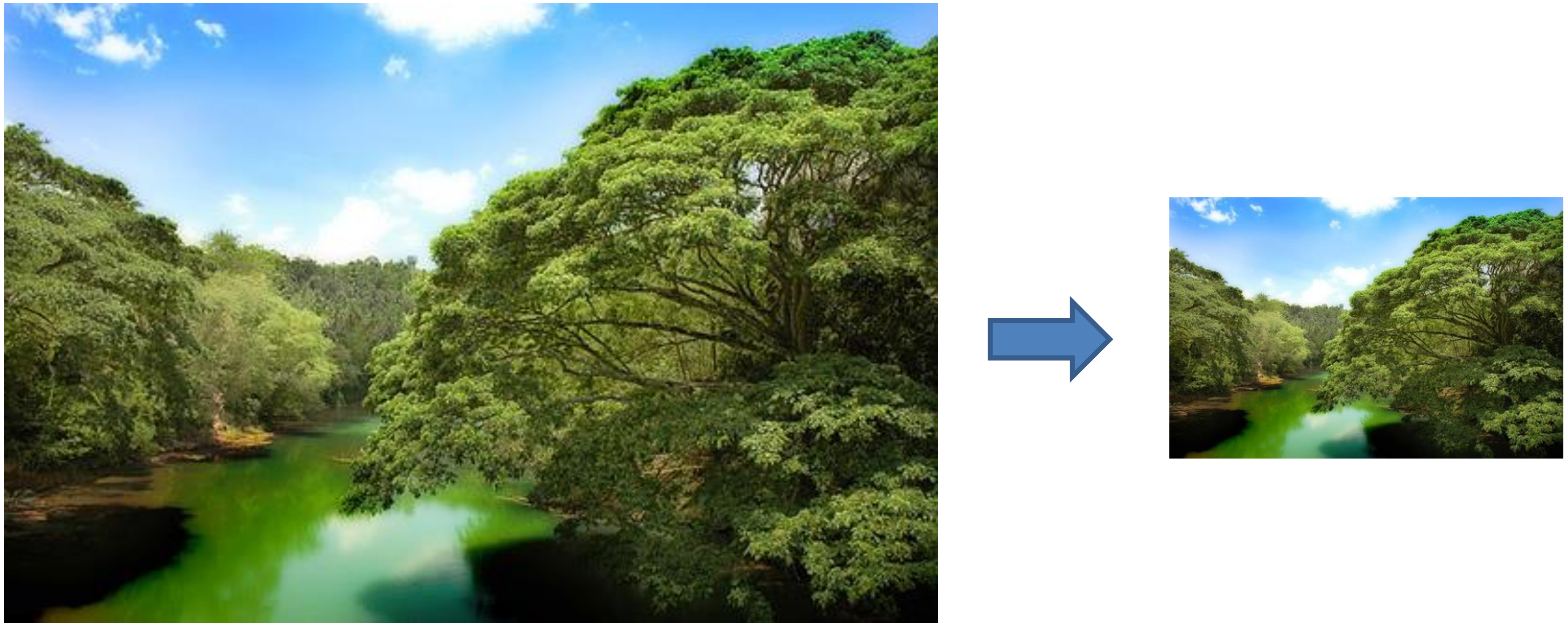
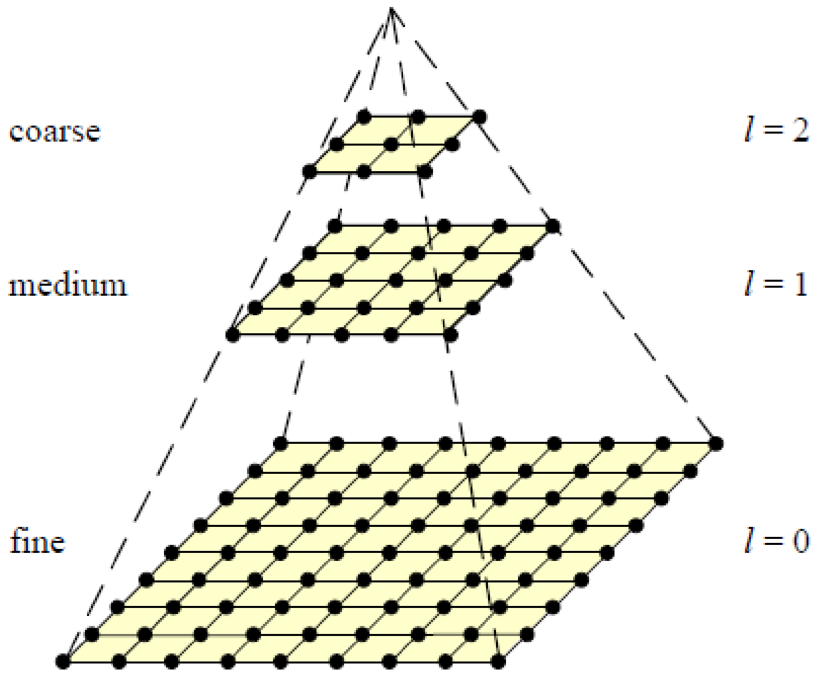


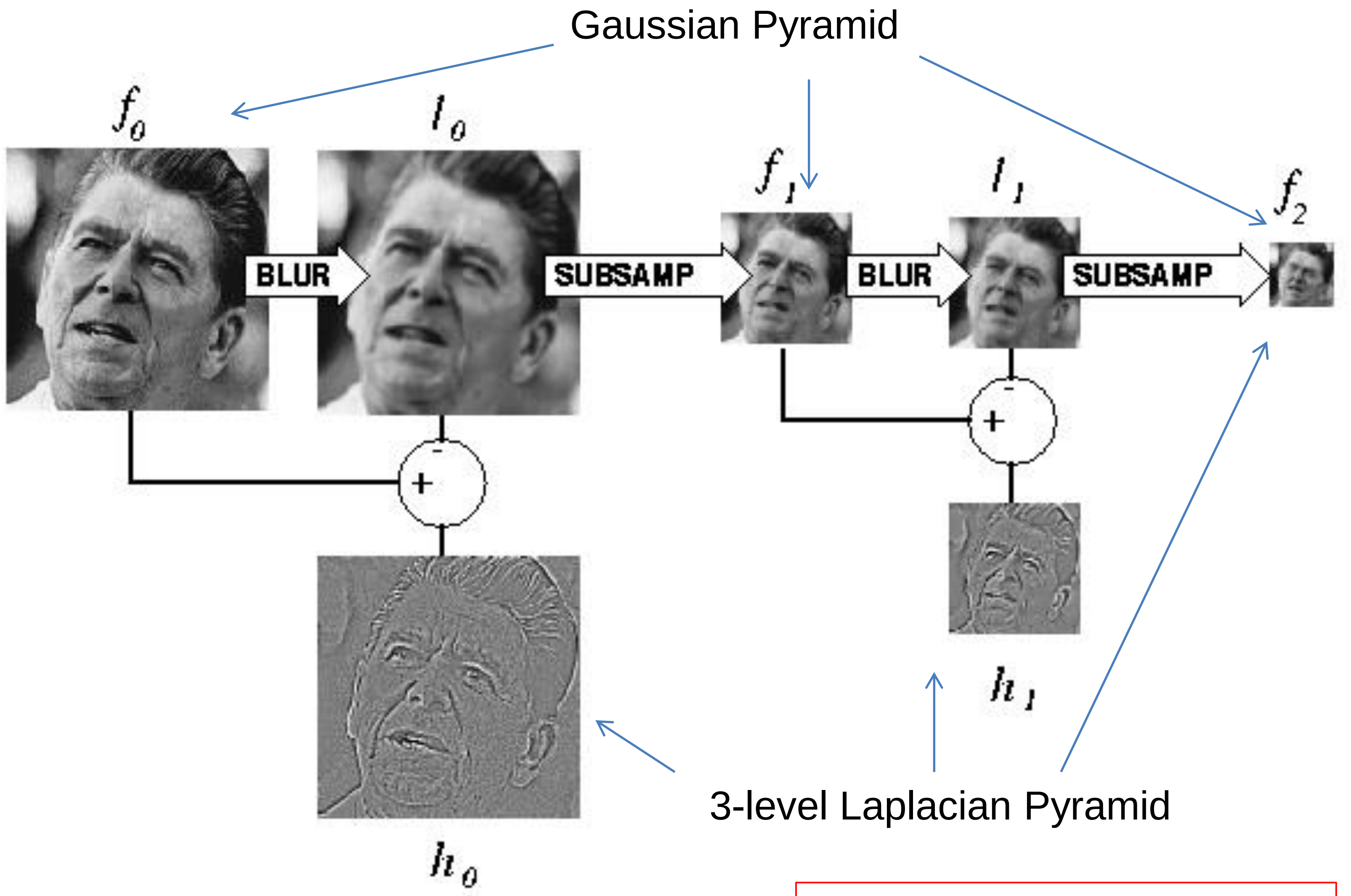
Image: http://www.flickr.com/photos/igorms/136916757/

Gaussian pyramid

- Useful for coarse-to-fine matching
- Applications include multi-scale object detection, image alignment, optical flow, point tracking



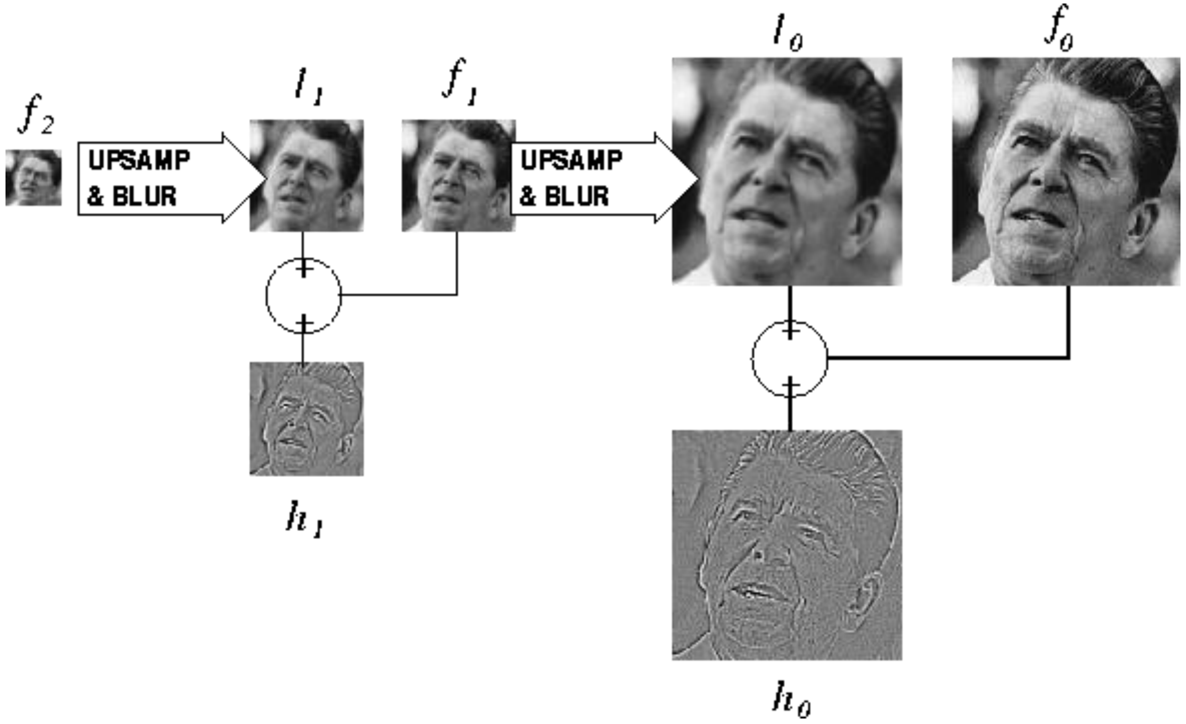
Computing Gaussian/Laplacian Pyramid



Useful figure to remember

http://sepwww.stanford.edu/~morgan/texturematch/paper_html/node3.html

Image reconstruction from Laplacian pyramid

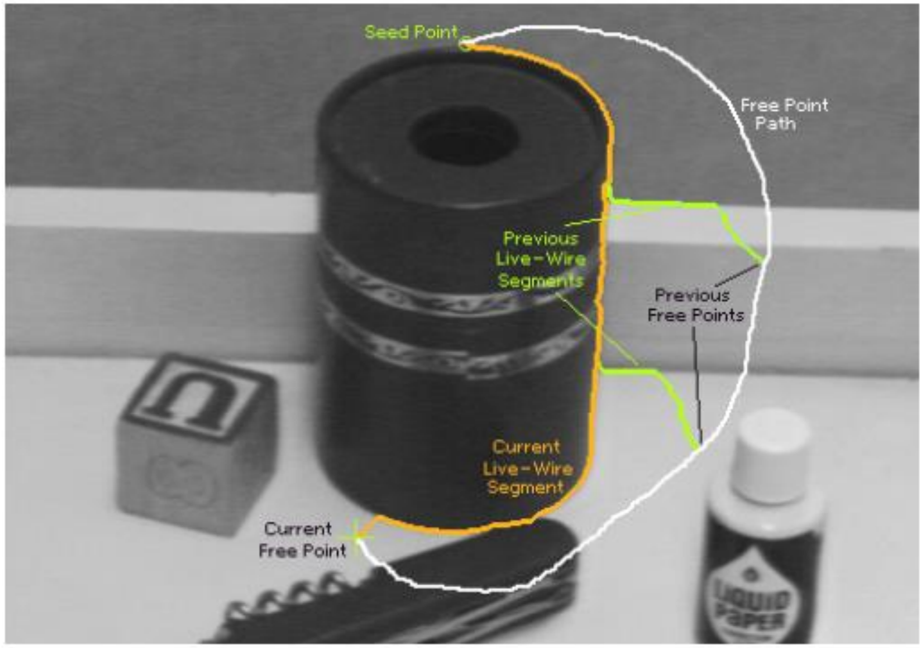


4. Region selection and compositing

- Selecting image regions
  - Intelligent scissors
  - Graph cuts
- Compositing
  - Alpha masks
  - Feathering
  - Laplacian pyramid blending
  - Poisson blending

Intelligent Scissors

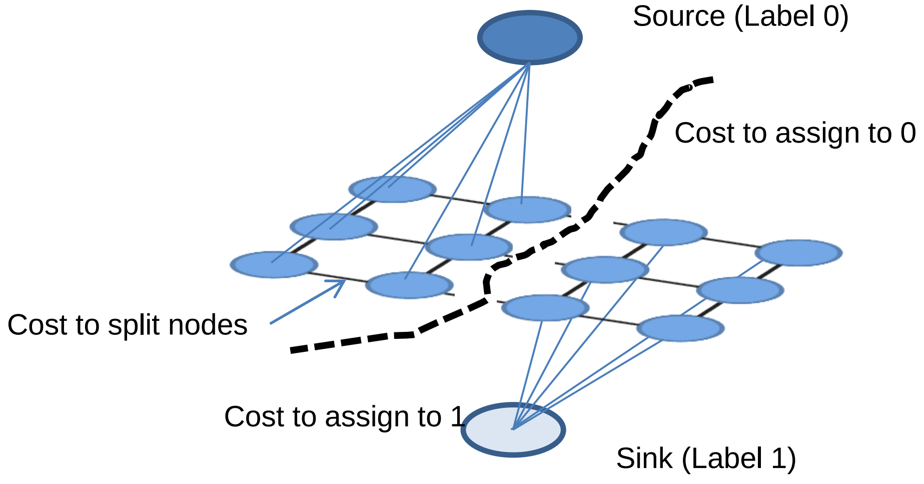
- You can treat the image as a graph
  - Nodes = pixels, edges connect neighboring pixels

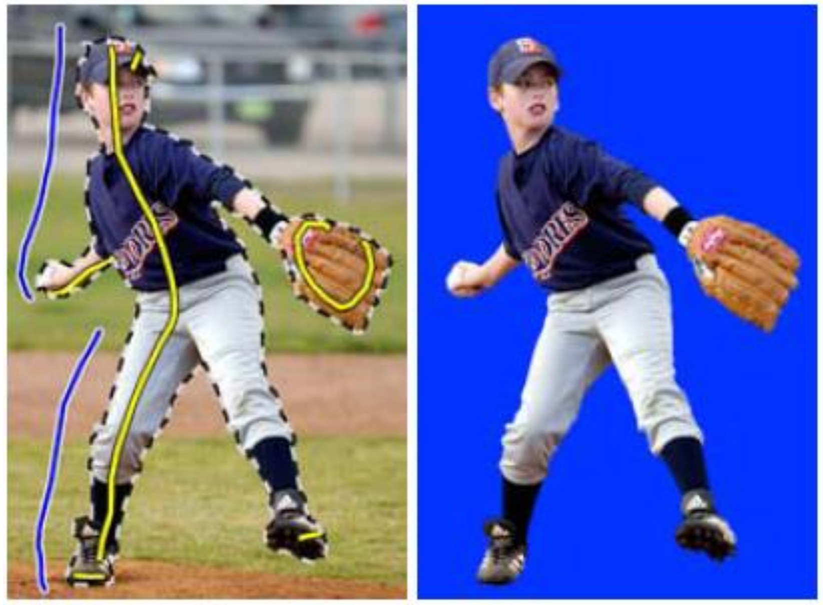


Intelligent Scissors: Good boundaries are a short (high gradient) path through the graph

Graph Cuts

- You can treat the image as a graph
  - Nodes = pixels, edges connect neighboring pixels





Graph cut: Good boundaries are a cheap cut, where some pixels want to be foreground, and some to be background

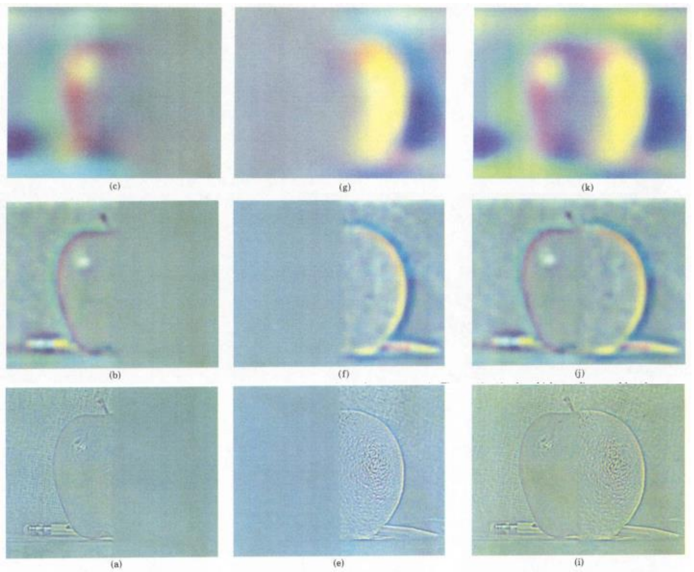
Compositing and Blending

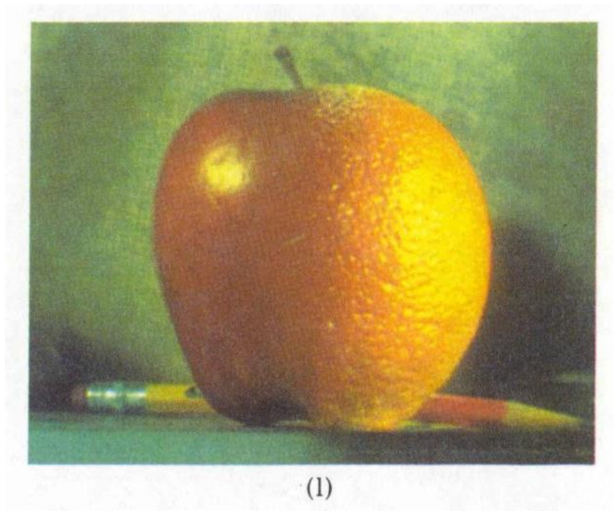
- Feathering: blur mask around its edges
- Laplacian blending: blend low-frequency slowly, high frequency quickly
  - Blend with alpha mask values ranging from 0 to 1

laplacian level 4

laplacian level 2

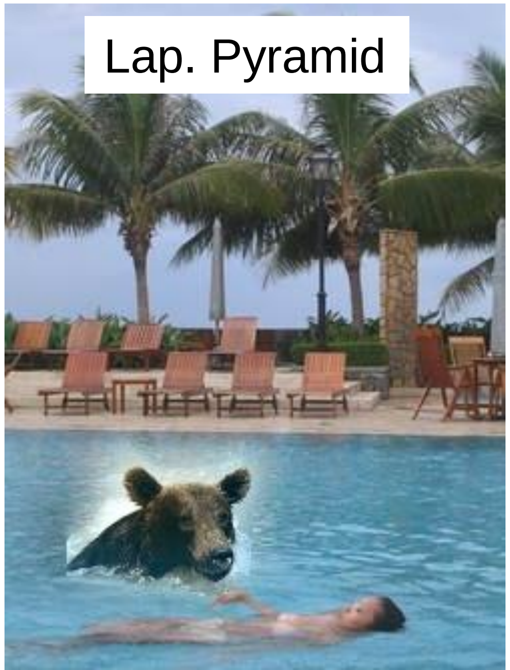
laplacian level 0

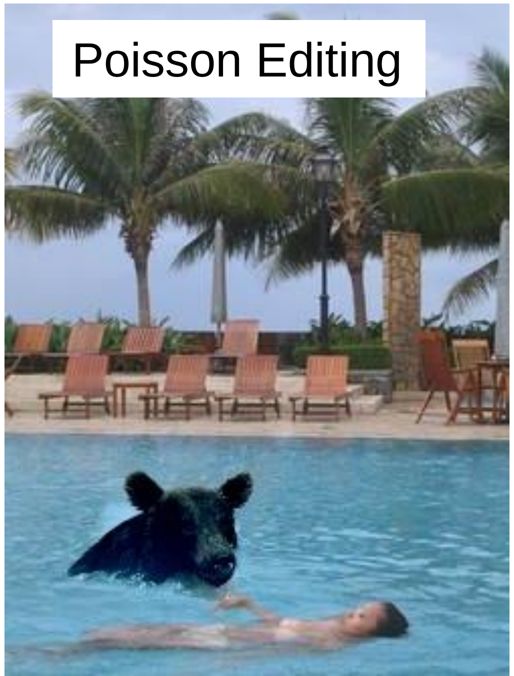


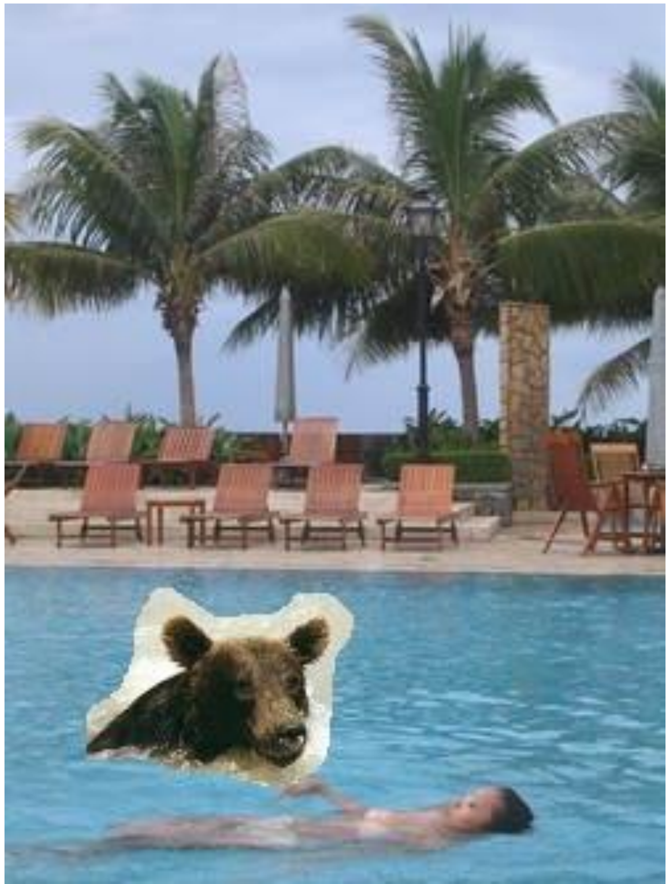


Question

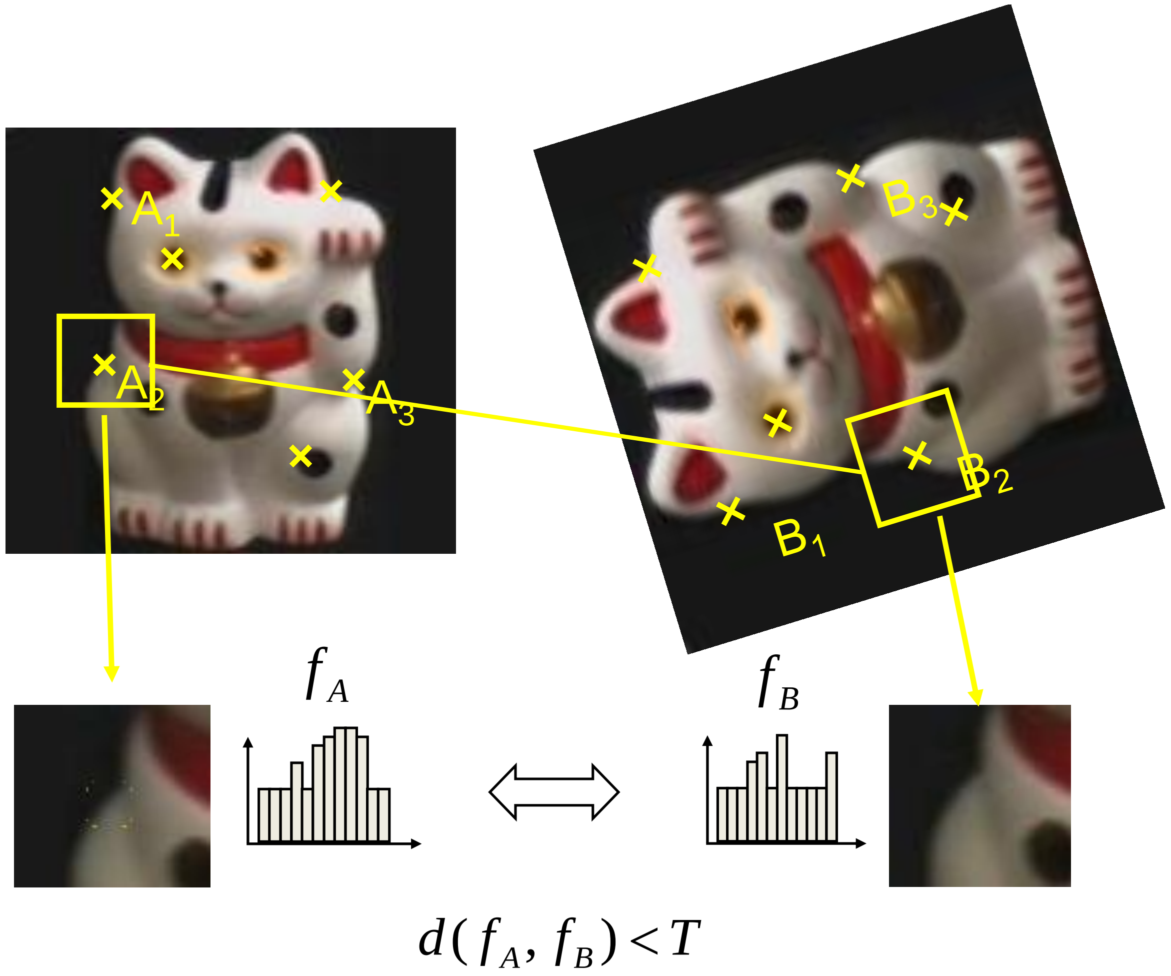
1) I am trying to blend this bear into this pool. What problems will I have if I use:
   a) Alpha compositing with feathering
   b) Laplacian pyramid blending
   c) Poisson editing?







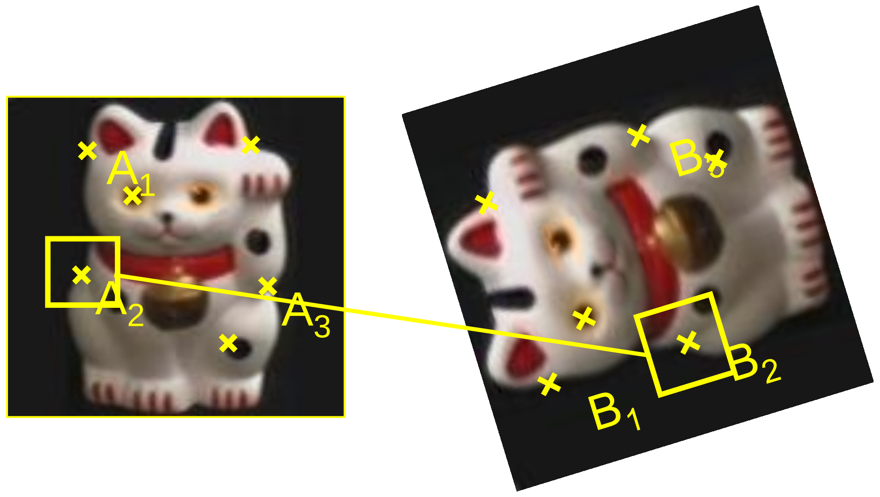
Keypoint Matching



1. **Find a set of distinctive key-points**
2. **Define a region around each keypoint**
3. **Extract and normalize the region content**
4. **Compute a local descriptor from the normalized region**
5. **Match local descriptors**

K. Grauman, B. Leibe

Key trade-offs



Localization

More Points

Robust to occlusion
Works with less texture

More Repeatable

Robust detection
Precise localization

Description

More Robust

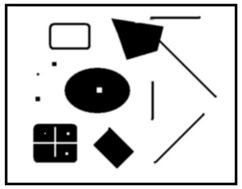
Deal with expected variations
Maximize correct matches

More Selective

Minimize wrong matches

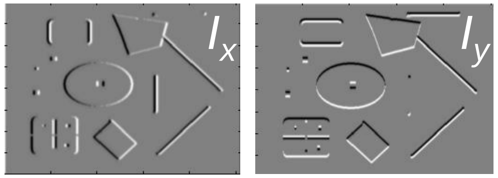
Harris Detector [Harris88]

- Second moment matrix

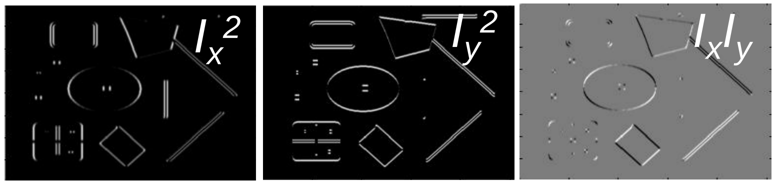


$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives

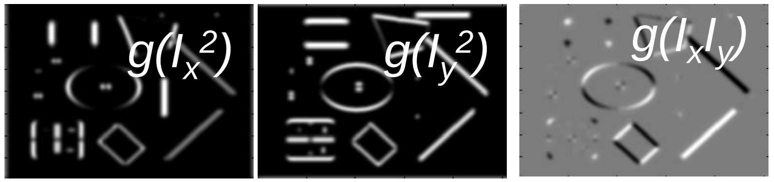


2. Square of derivatives



$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

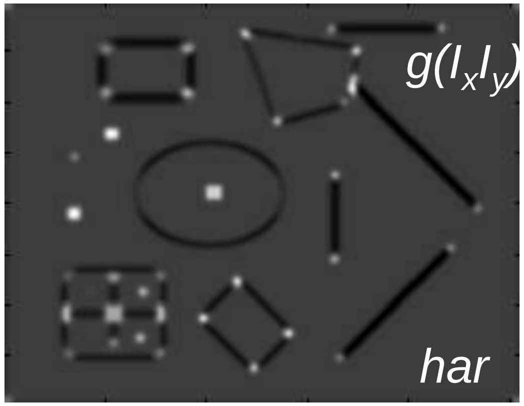
3. Gaussian filter $g(\sigma_I)$



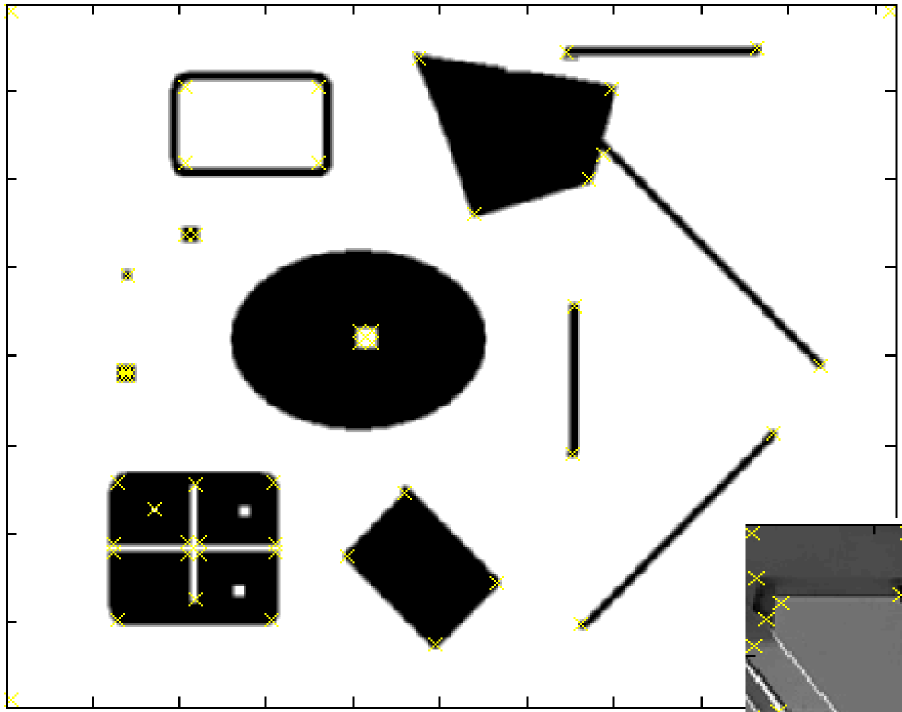
4. Cornerness function – both eigenvalues are strong

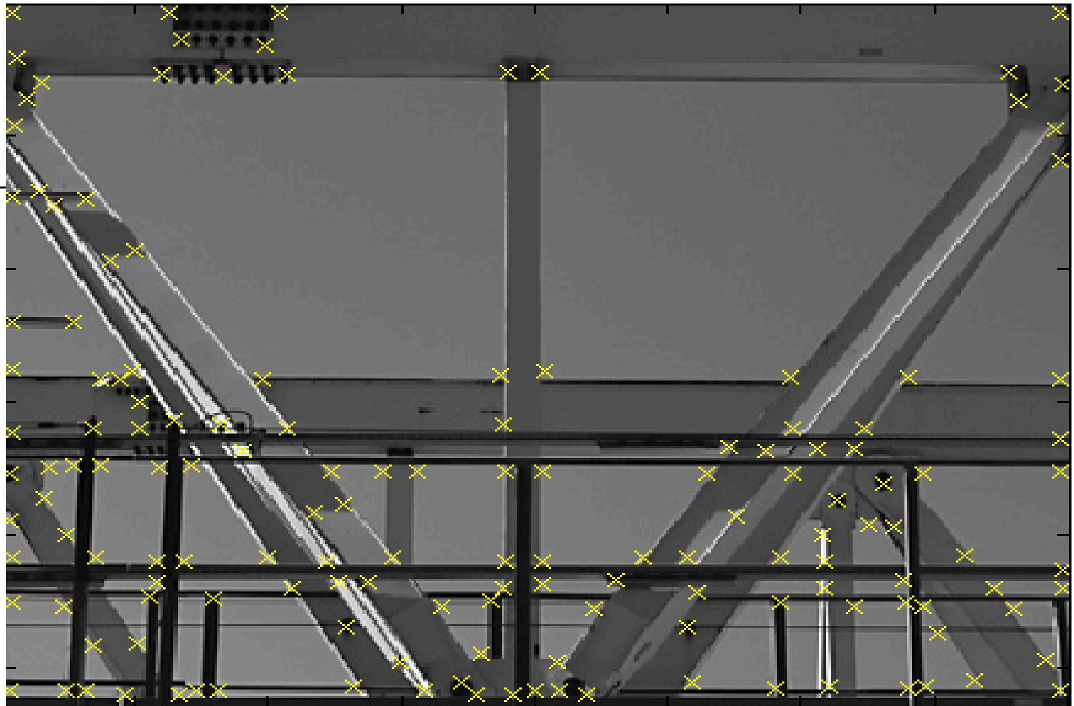
$$har = \det[\mu(\sigma_I, \sigma_D)] - \alpha[\operatorname{trace}(\mu(\sigma_I, \sigma_D))^2] =$$
$$g(I_x^2)g(I_y^2) - [g(I_xI_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

5. Non-maxima suppression



Harris Detector – Responses [Harris88]





Effect: **A very precise corner detector.**

Applications

- Image stitching
  - Matching keypoints
  - Solving for homography
  - RANSAC

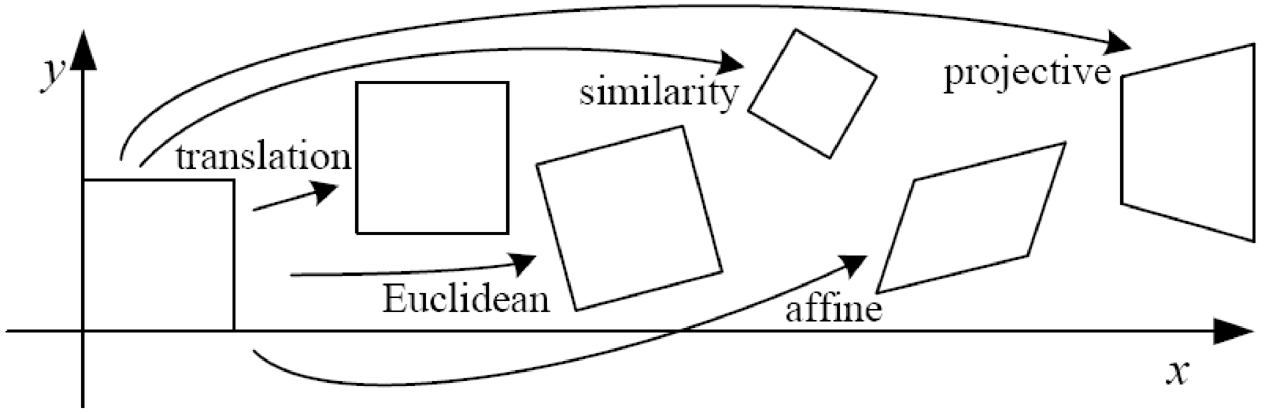
- Object recognition
  - Clustering keypoints and creating tf-idf tables for fast retrieval
  - Geometric verification

5. Solving for transformations

- Map between 2D coordinates using linear projection

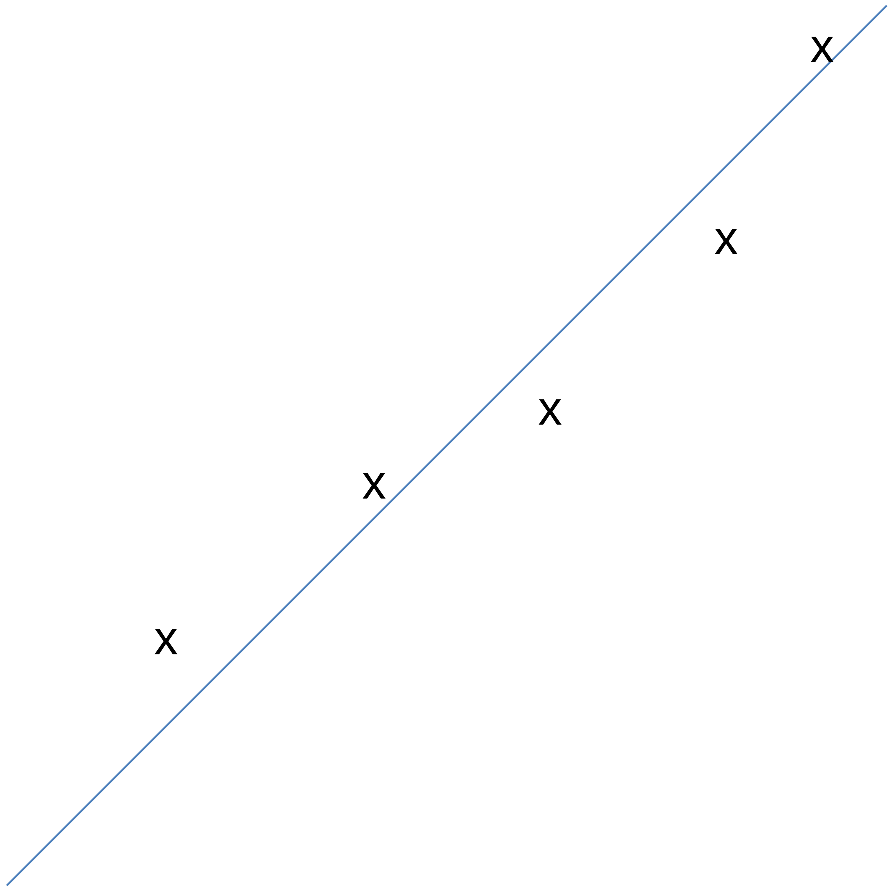
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation $+\cdots$	□
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths $+\cdots$	◇
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles $+\cdots$	◇
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6	parallelism $+\cdots$	▱
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	⏢

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$



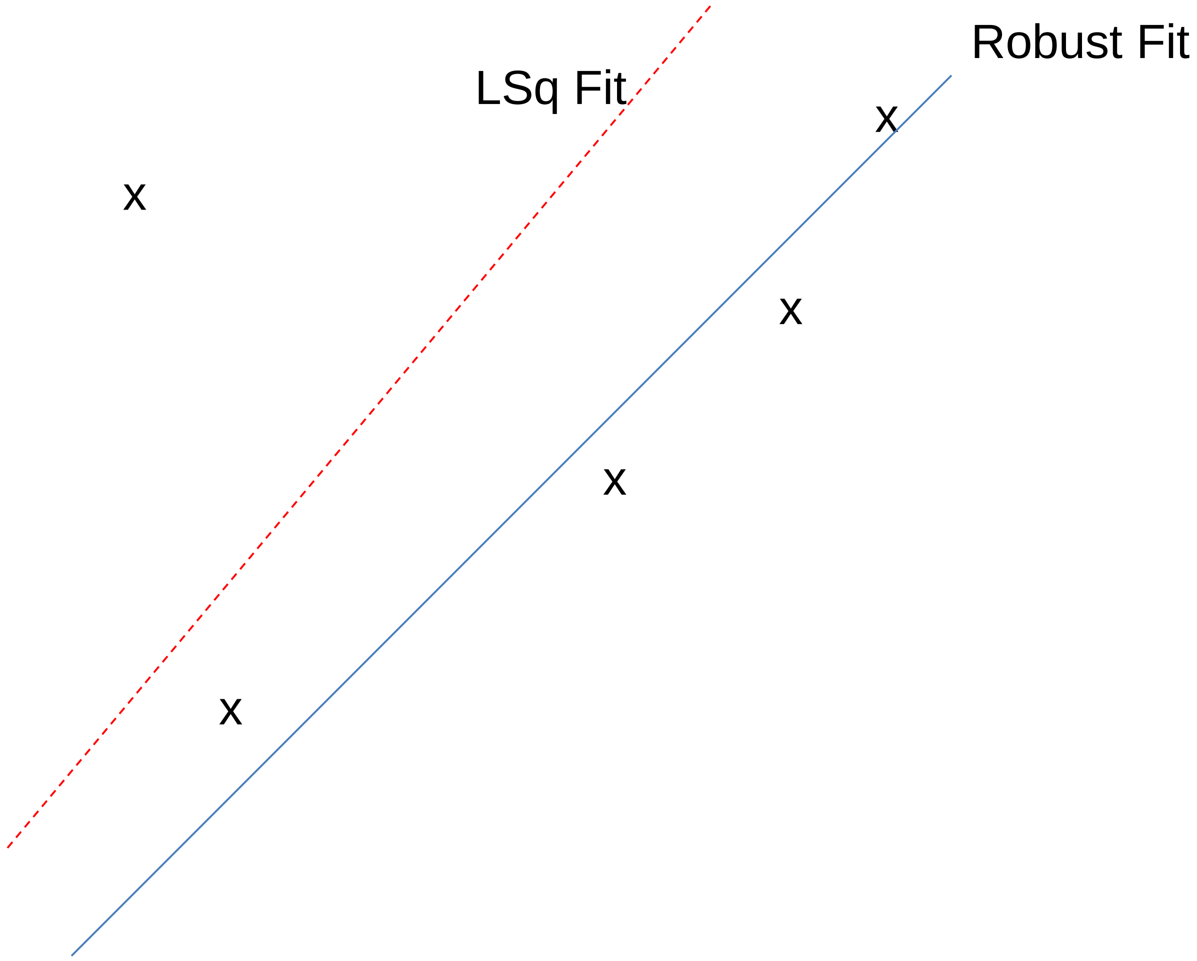
Least-squares Solving

- If all points (or correspondences) fit the model with some noise, better to use all for least squares estimate



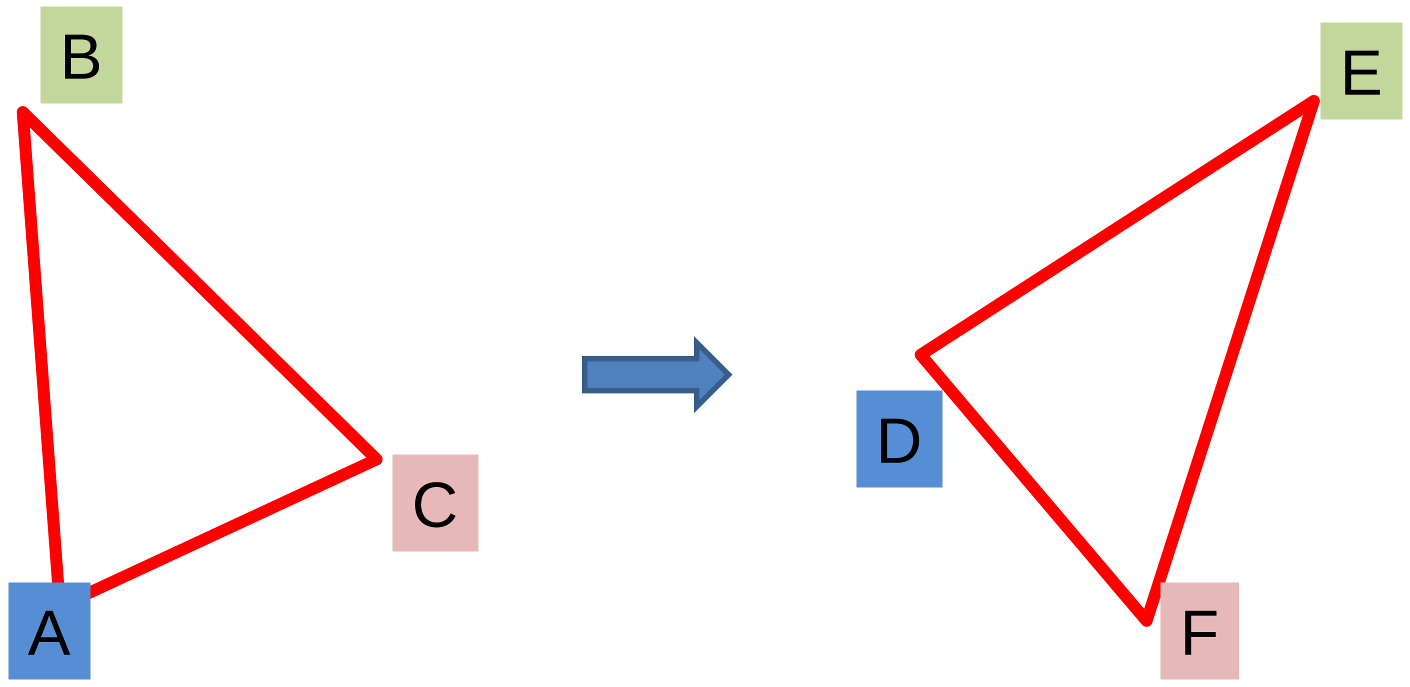
Least-squares Solving

- If some points are outliers, robust approach such as RANSAC is needed



Question

Suppose we have two triangles, ABC and DEF, related by a general affine transformation. Solve for the transformation.



Good luck!

- Questions?

Take-home questions

2) How would you make a sharpening filter using gradient domain processing? What are the constraints on the gradients and the intensities?

Take-home question

Suppose you have estimated three vanishing points corresponding to orthogonal directions. How can you recover the rotation matrix that is aligned with the 3D axes defined by these points?

- Assume that intrinsic matrix K has three parameters
- Remember, in homogeneous coordinates, we can write a 3d point at infinity as (X, Y, Z, 0)

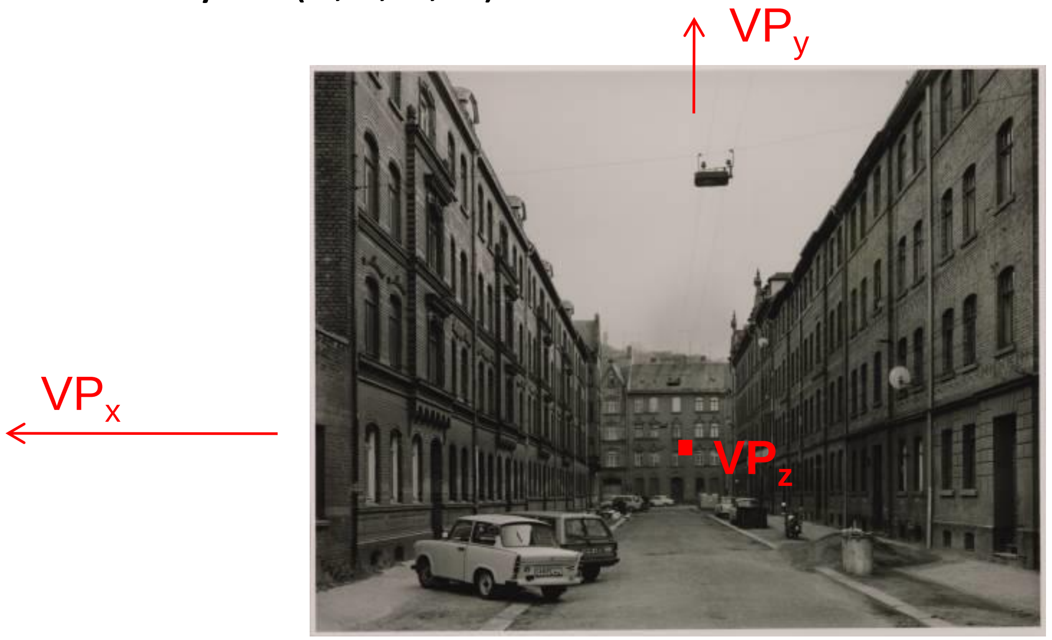


Photo from online Tate collection