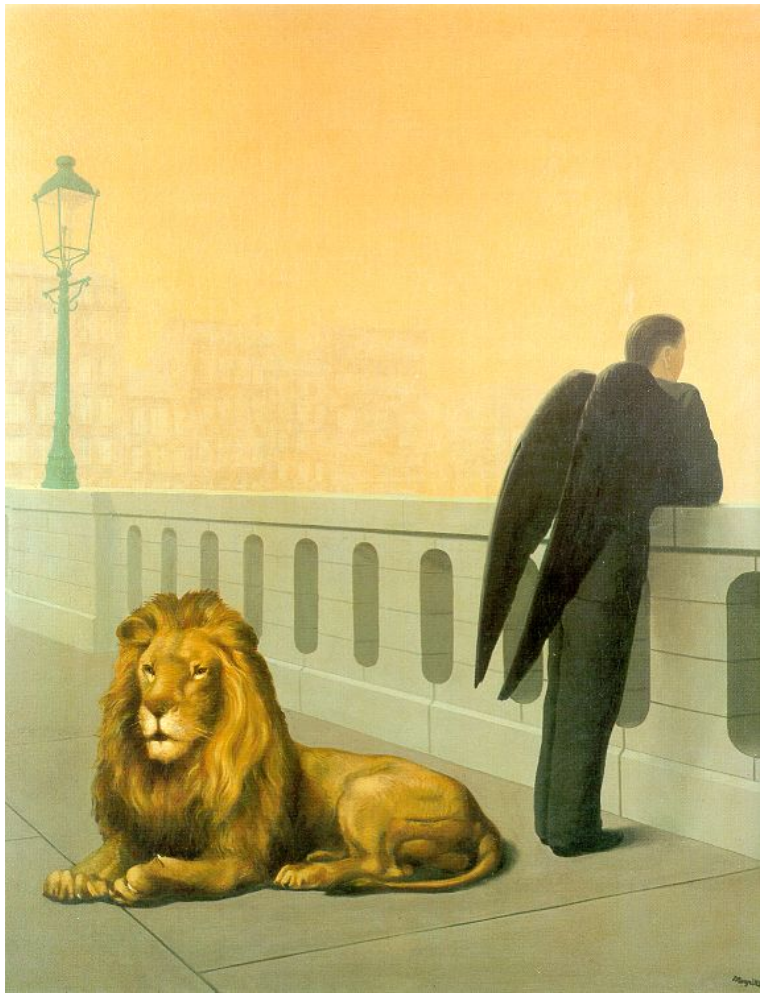
Midterm Review



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?

Preparing for the Exam

Topics include:

- **Filtering** (inc. apply, design, interpret simple filters, frequency interpretation, properties of filters)
- **Lighting** (inc. lighting and material models)
- **Camera models** (inc. pinhole model and its applications, effects of lens and camera controls)
- **Single view geometry** (inc. relative measurements within an image)
- **Interest points and correspondence** (inc. interest point detectors, matching, retrieval based on interest points)
- **Alignment/transformations** (inc. global transformations, setting up matrix to solve for a transformation, RANSAC)
- **Color spaces, pyramids, and sampling**
- **"Topics of Interest"** and other

- Each bullet has 10-20 points allocated (out of 100)
- There are 10 questions.
- Questions about special topics typically require that you understand the key concepts that are presented.

- **Bring a photo ID**
- **You cannot use notes or calculator**

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)

a surface appears equally bright from any viewing angle, but its brightness decreases as it is tilted away from a light source

Some light is absorbed (function of albedo)
remaining light is reflected in all directions (diffuse reflection)
examples of lambertian surface: soft cloth, concrete, matte paints

perceptually uniform
L = luminance = brightness
A = difference between red and green] = chrominance = color
B = difference between yellow and blue]
There is more information in intensity than color

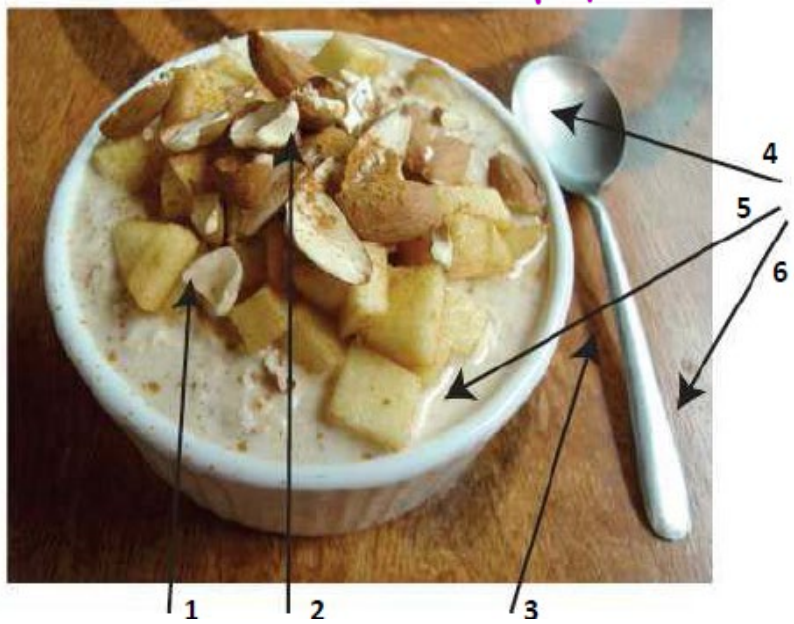
= bright spots on a shiny object

a point that can't see the source at all is in shadow
shading and shadows give major cues to shape and position

the different ways of representing visible colors on a screen

the default color space
has some drawbacks like strongly correlated color channels, and it is non-perceptual

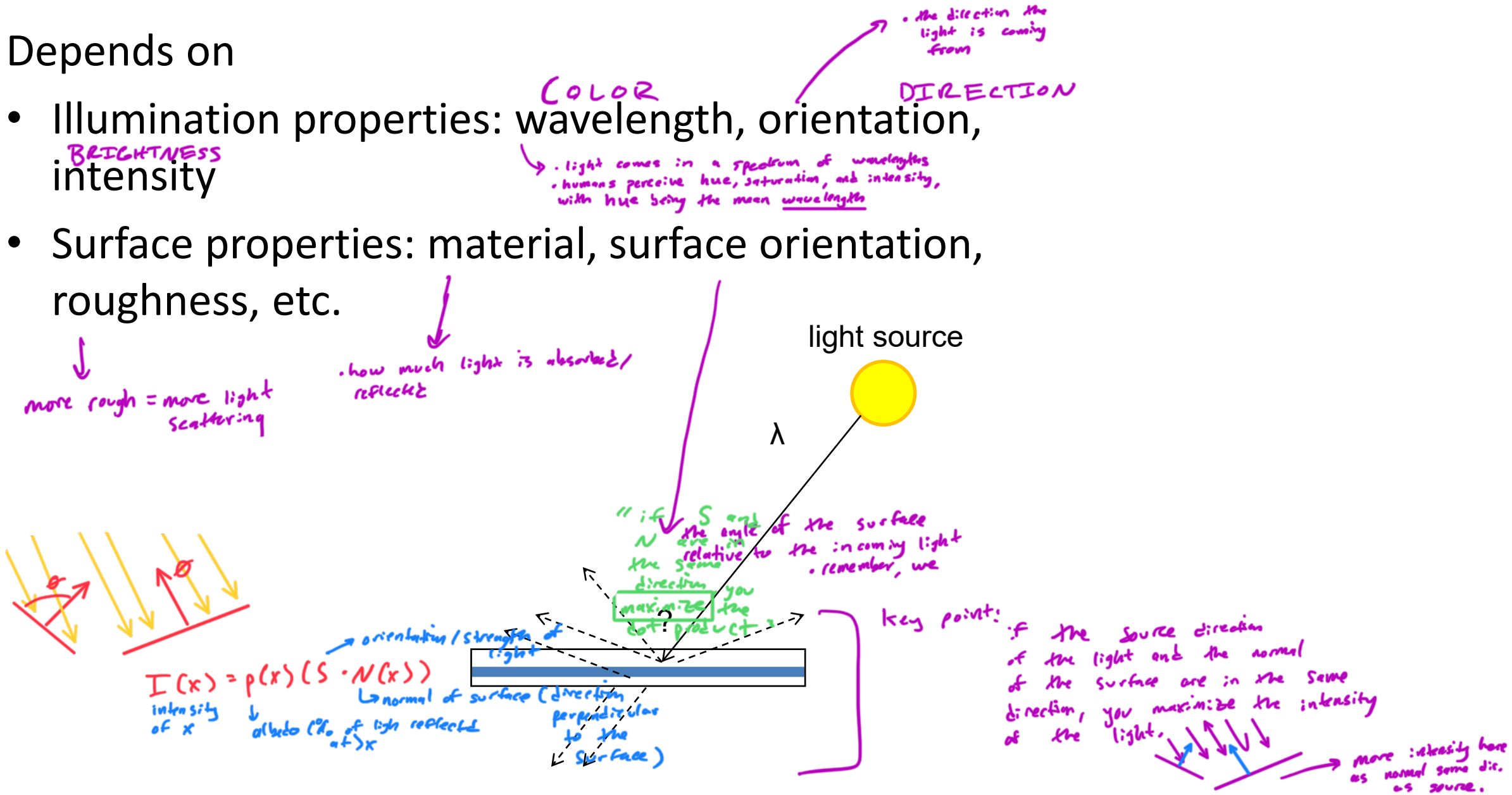
A more perceptual color space, although still not showing perceptual uniformity
"hue, saturation, and value" → non-linear function of RGB space for hue and saturation
hue = color of rainbow, saturation = brightness of the color, value = intensity image



How is light reflected from a surface?

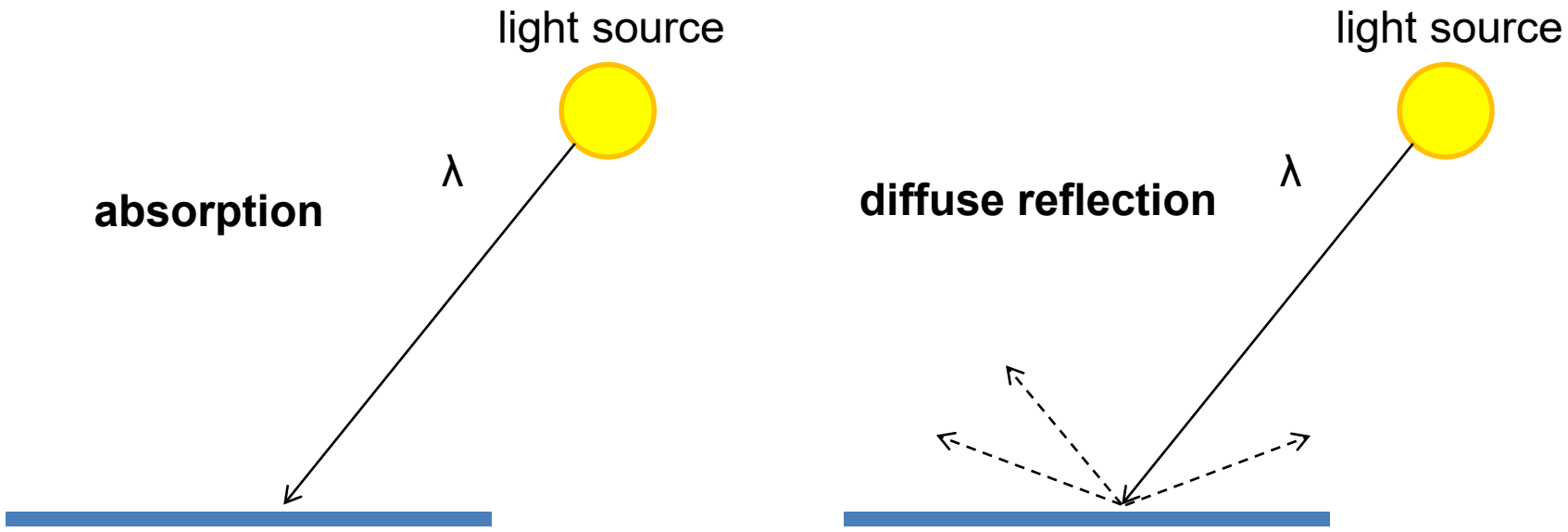
Depends on

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



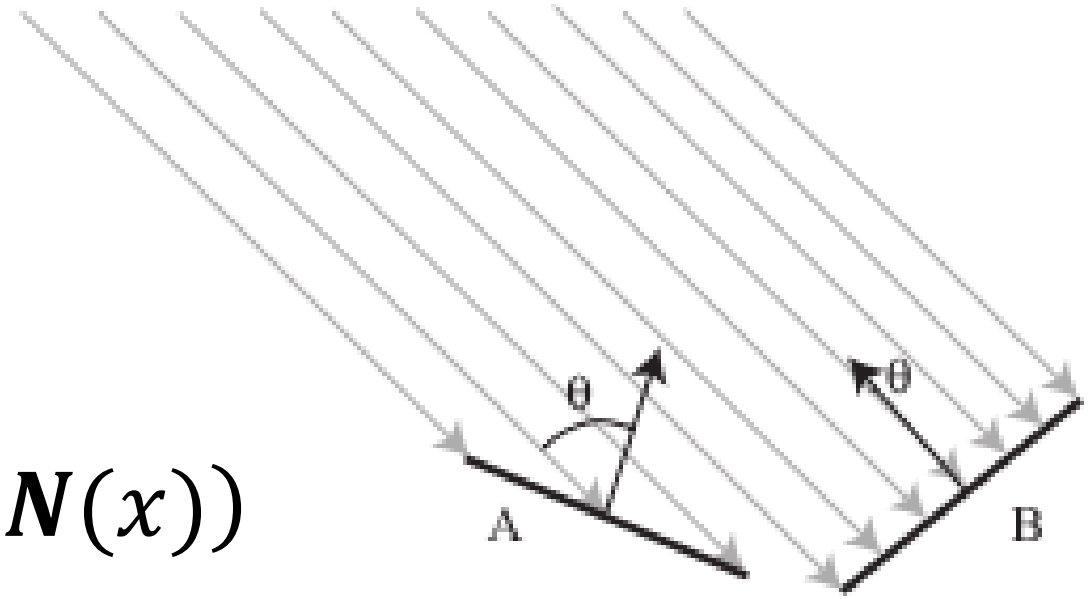
Lambertian surface

- 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.



ρ = albedo

$\boldsymbol{S}$ = directional source

$\boldsymbol{N}$ = surface normal

I = image intensity

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

Slide: Forsyth

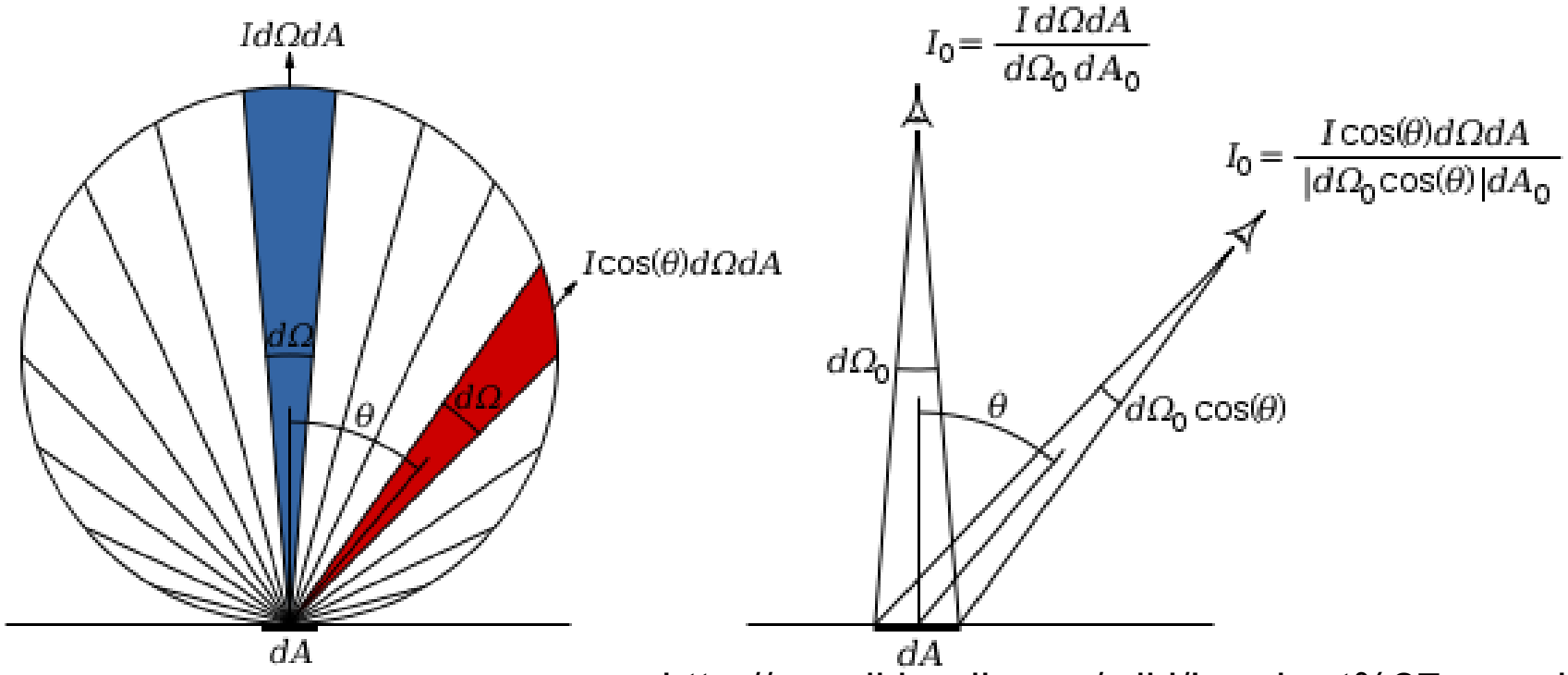

1
2

Diffuse reflection

Intensity does not depend on viewer angle.

– but does depend on illumination angle

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



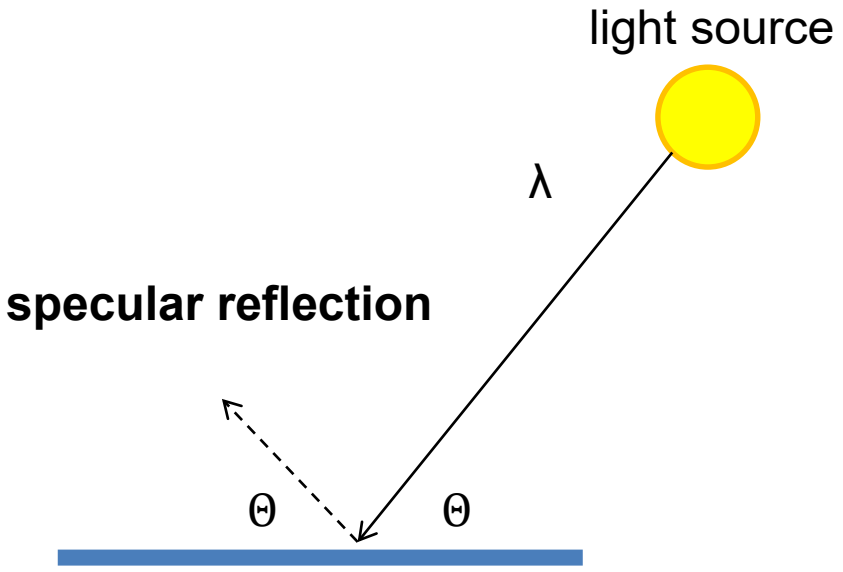
http://en.wikipedia.org/wiki/Lambert%27s_cosine_law

Specular Reflection

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

Flickr, by suzysputnik

Flickr, by piratejohnny



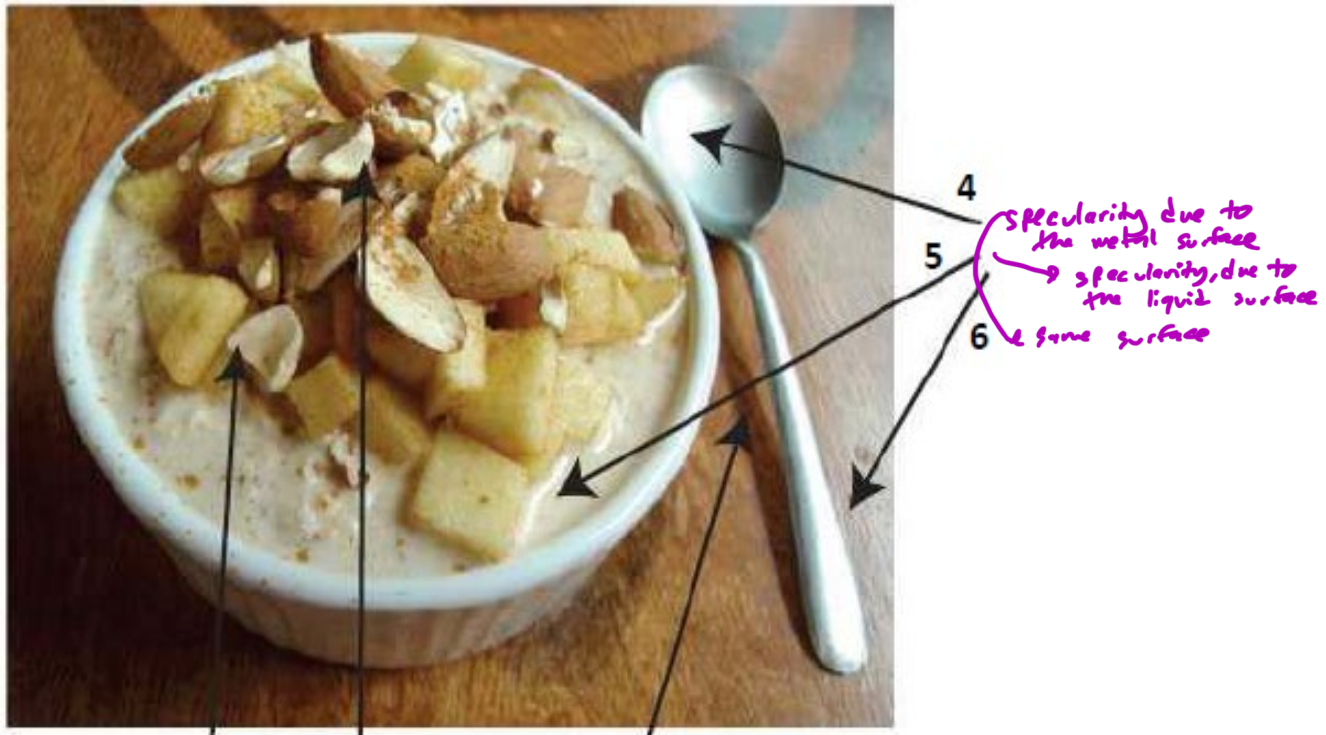
Many surfaces have both specular and diffuse components

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

Photo: northcountryhardwoodfloors.com

Questions

1.



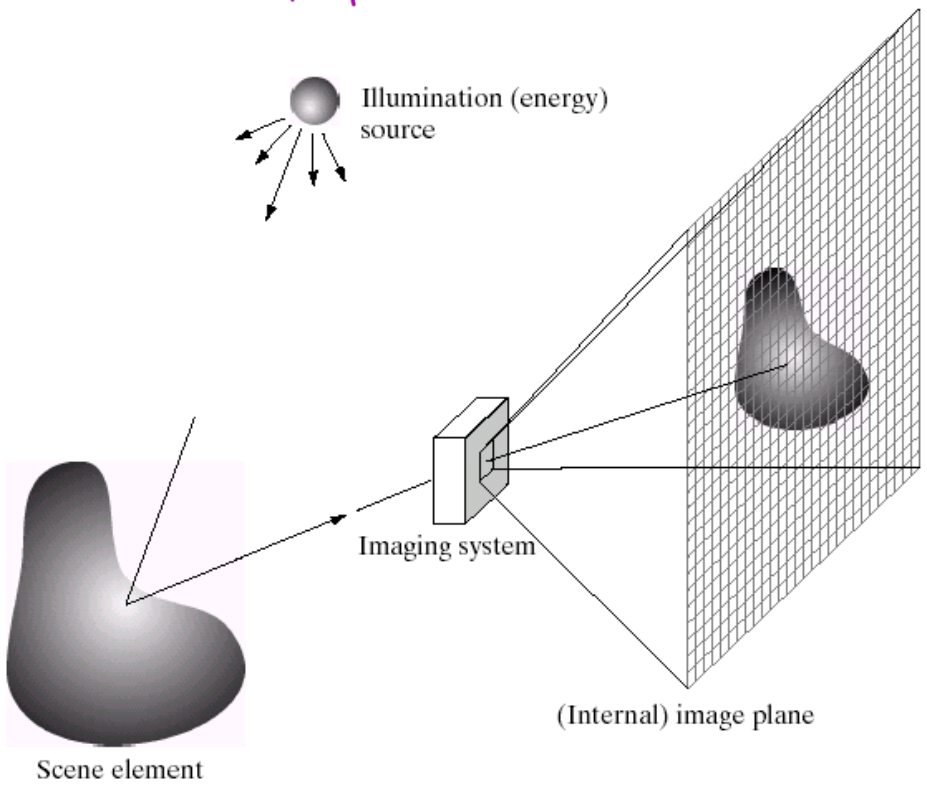
A. For each of the arrows in the above image, name the reasons the pixel near the end of the arrow has its brightness value and explain very briefly. The arrow pointing to milk is pointing to the thin bright line at the edge of the piece of apple; the arrow pointing to the spoon handle is pointing to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

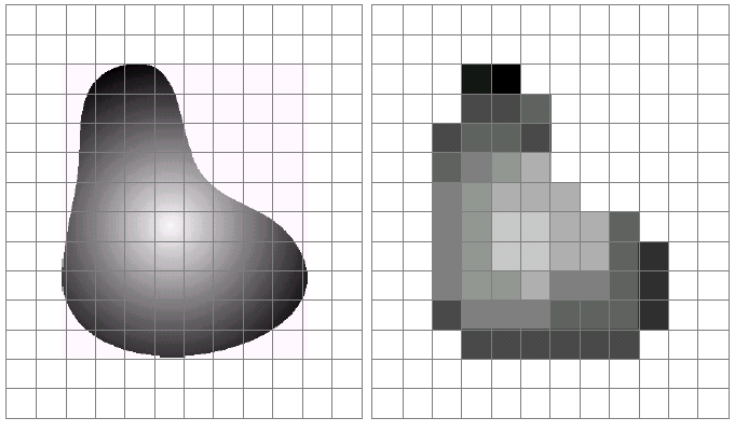
Possible factors: albedo, shadows, texture, specularities, curvature, lighting direction

Discretization

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

- Yep, this makes sense. But what happens if a value to be mapped from the real-world falls between two values in the pixel grid?
  - interpolation (bilinear interpolation)



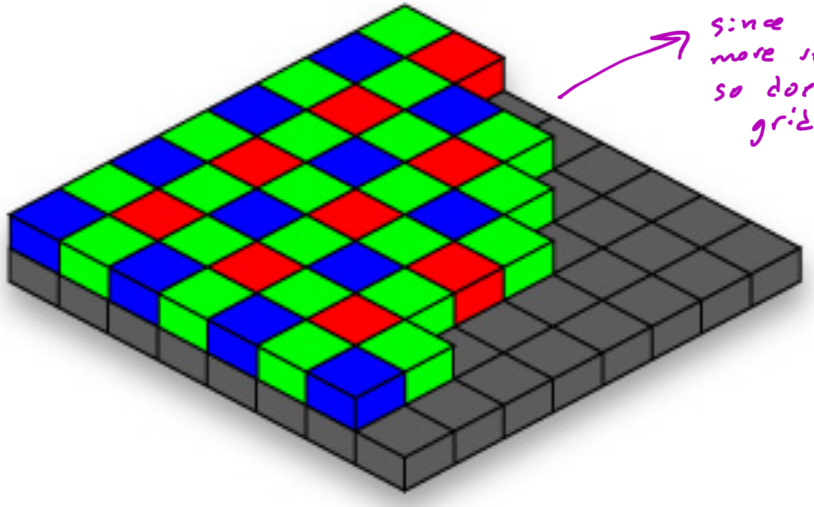


a b

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

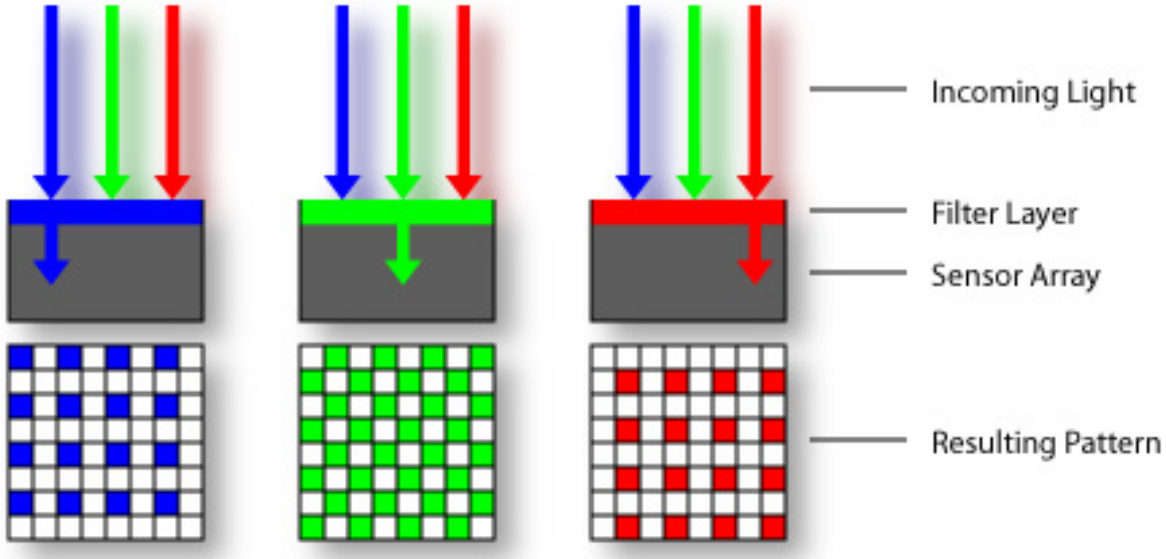
Color Sensing: Bayer Grid

put a set of filters in front of photo receptor cells
- each pixel measures amount of red or blue or green light
- each cell gets a range, pixels will be in red, blue, or green



since eyes have more red and green, so does the bayer grid

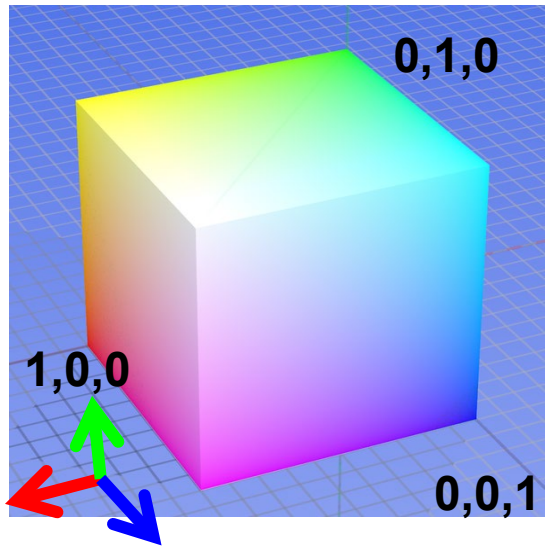
Estimate RGB at each cell from neighboring values



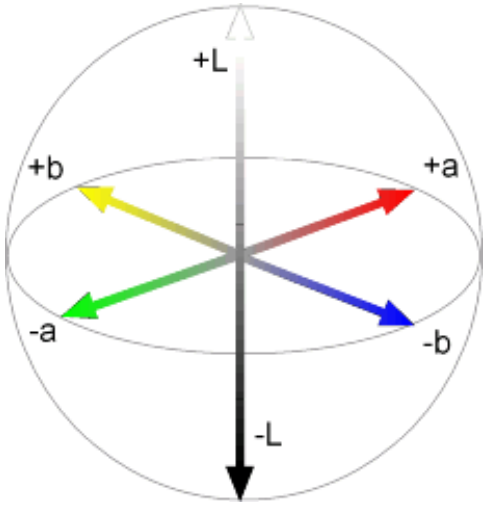
http://en.wikipedia.org/wiki/Bayer_filter

Color spaces

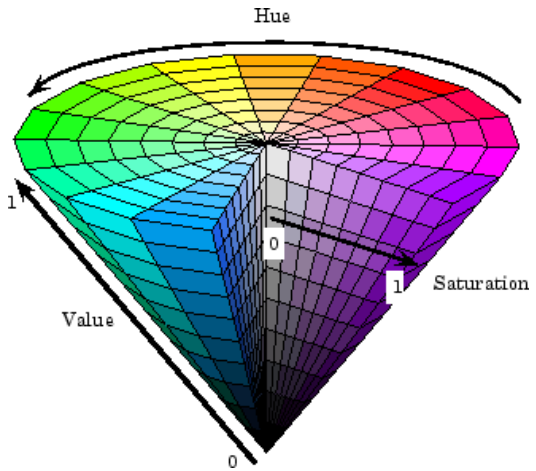
RGB ✓



LAB ✓



HSV ✓



YCbCr

- one of the more perceptual (than RGB) color spaces
- used by TV communication, good for compression
- similar to LAB, 1 Lum, 2 chrom. channels
- can be computed with a linear operation

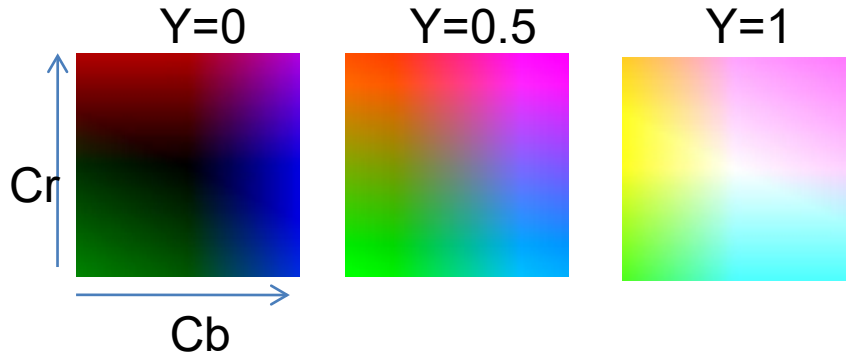
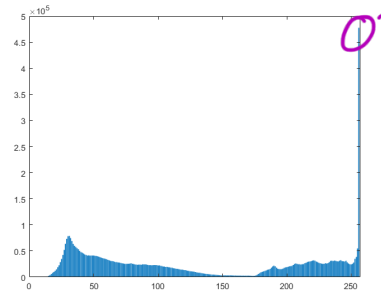
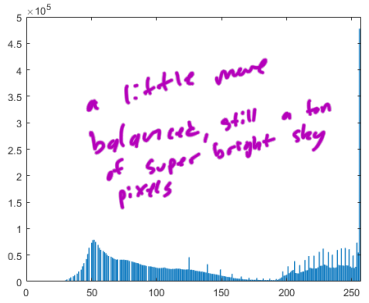
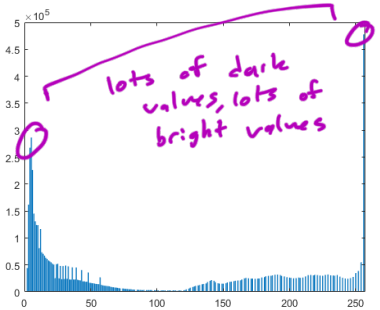
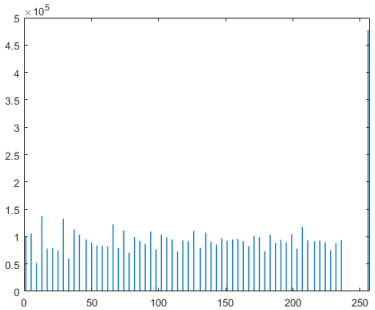


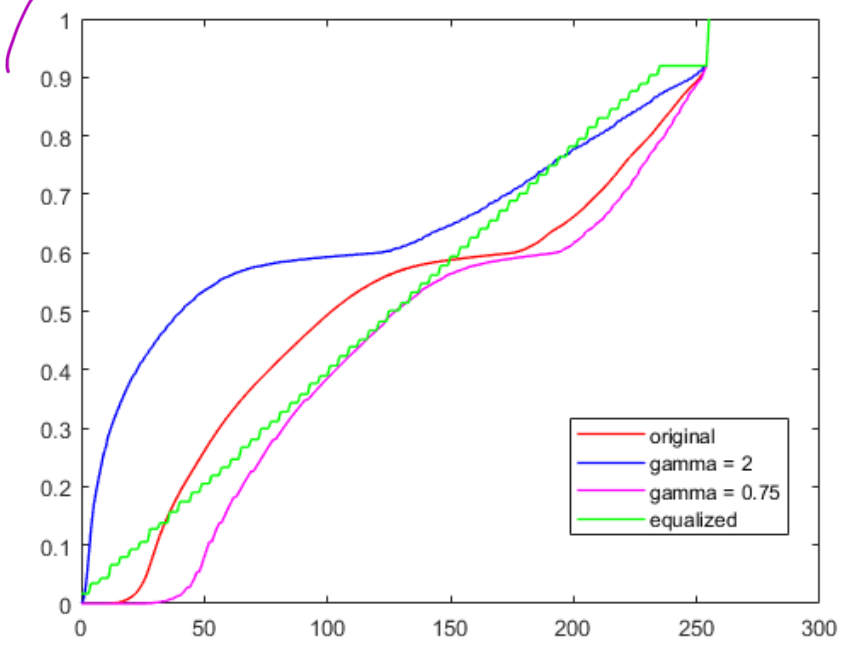
Image Histograms









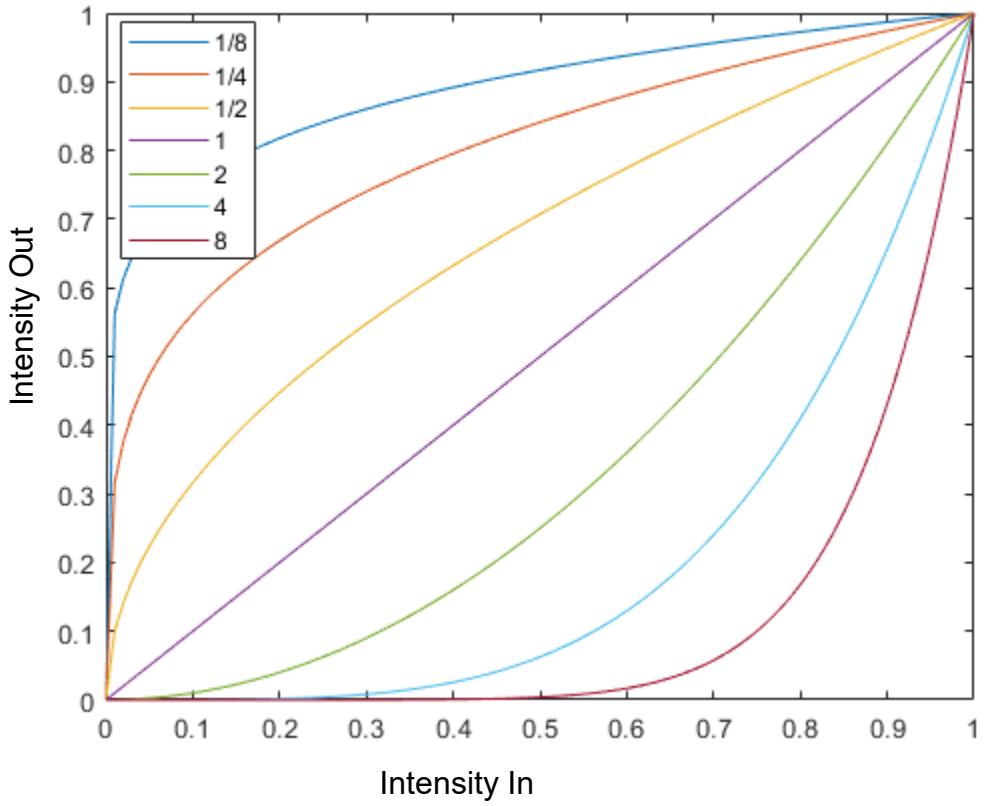


Cumulative Histograms

Gamma adjustment

- for each pixel value, we take that pixel raised to the power of Gamma
- low gamma = brighter
- high gamma = darker

$$i_{out} = i_{in}^{\gamma}$$

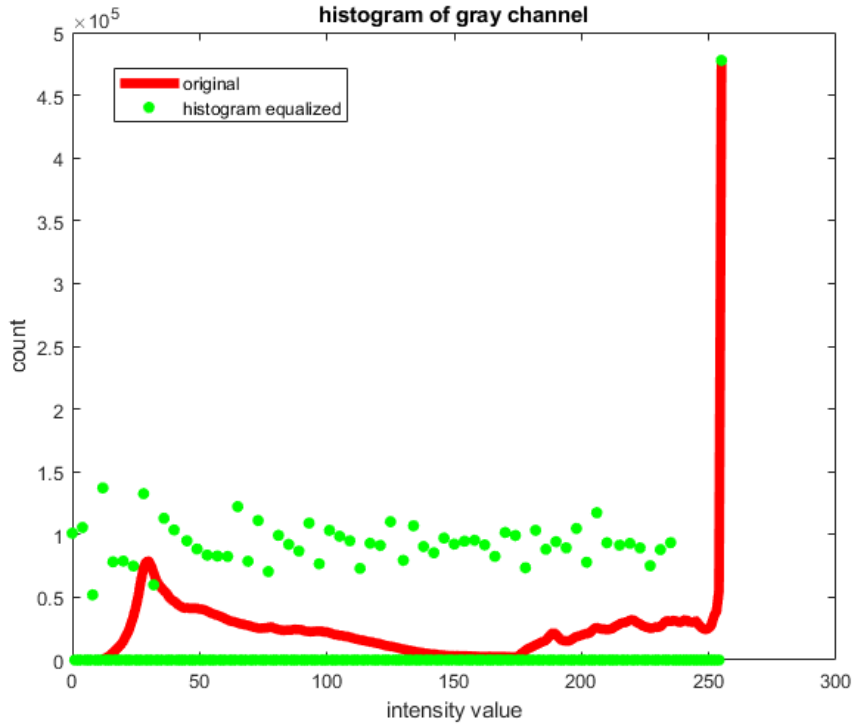


γ = 0.5

γ = 1

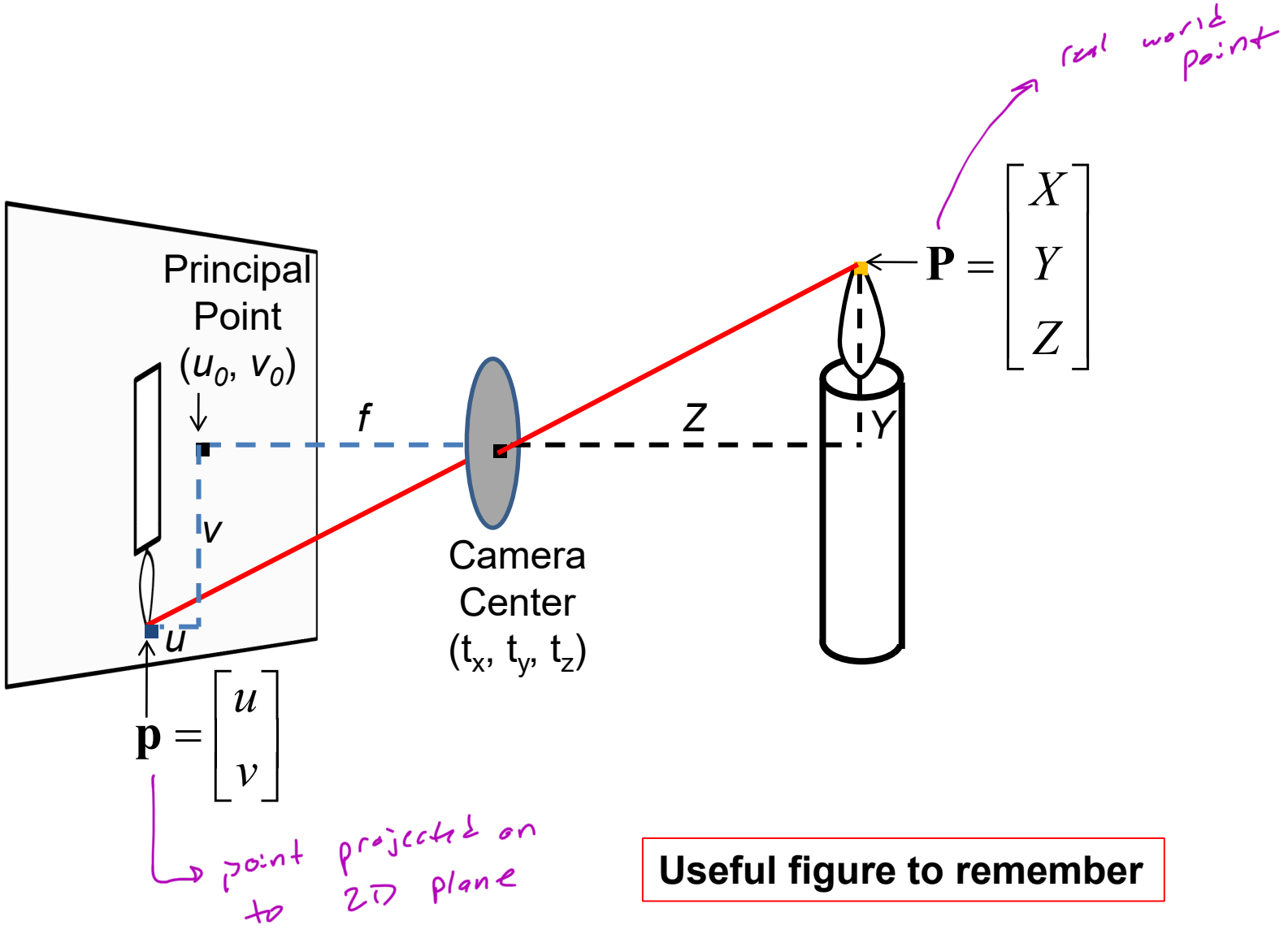
γ = 2

Histogram Equalization



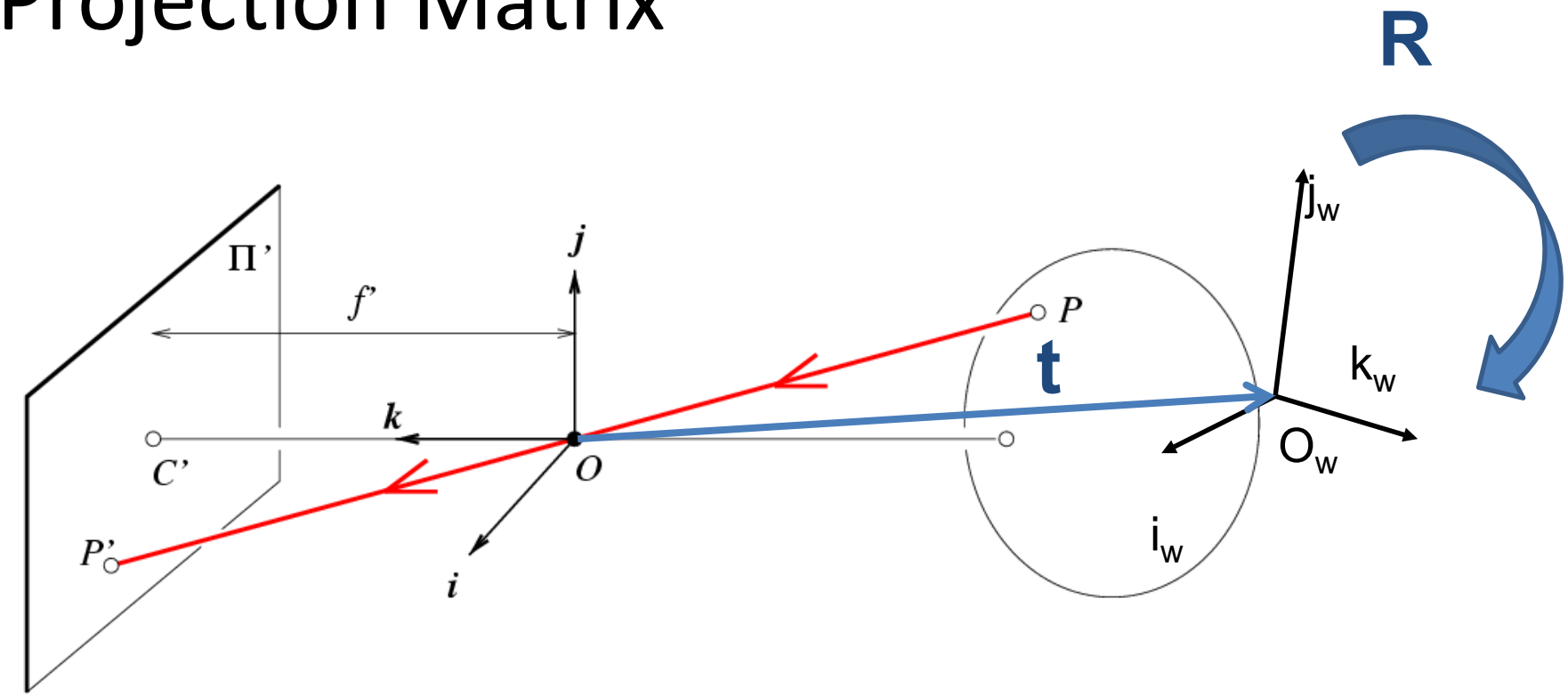
2. Camera Capture and Geometry

Pinhole Camera



Useful figure to remember

Projection Matrix



intrinsic

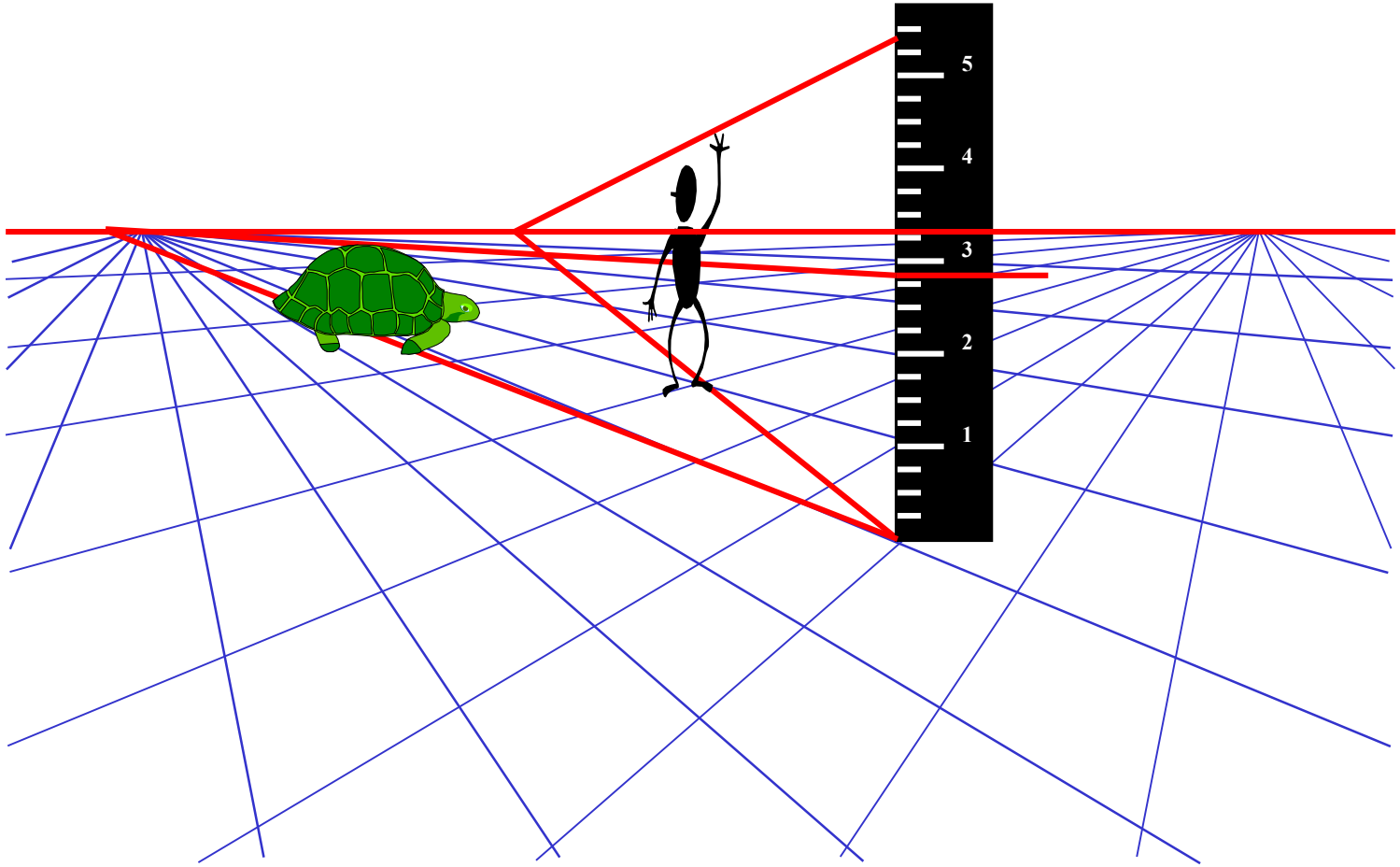
Rotation and translation = extrinsic matrix

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

real world coordinates

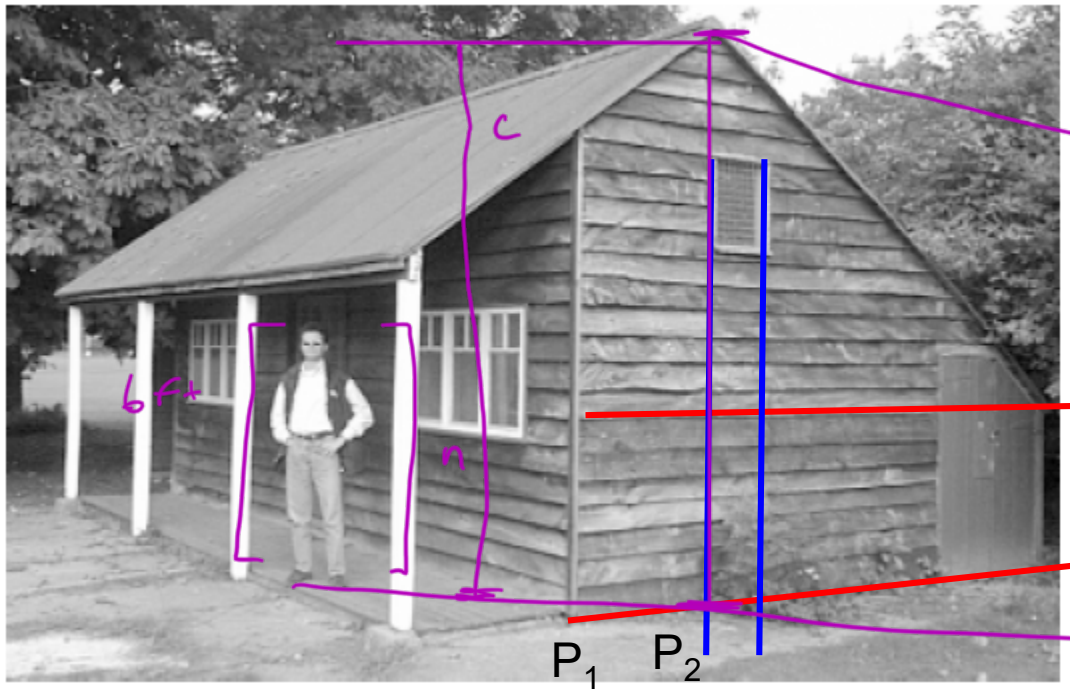
Single-view metrology

→ see quizzes... solve these



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?



So, we basically know here that

cross-ratio

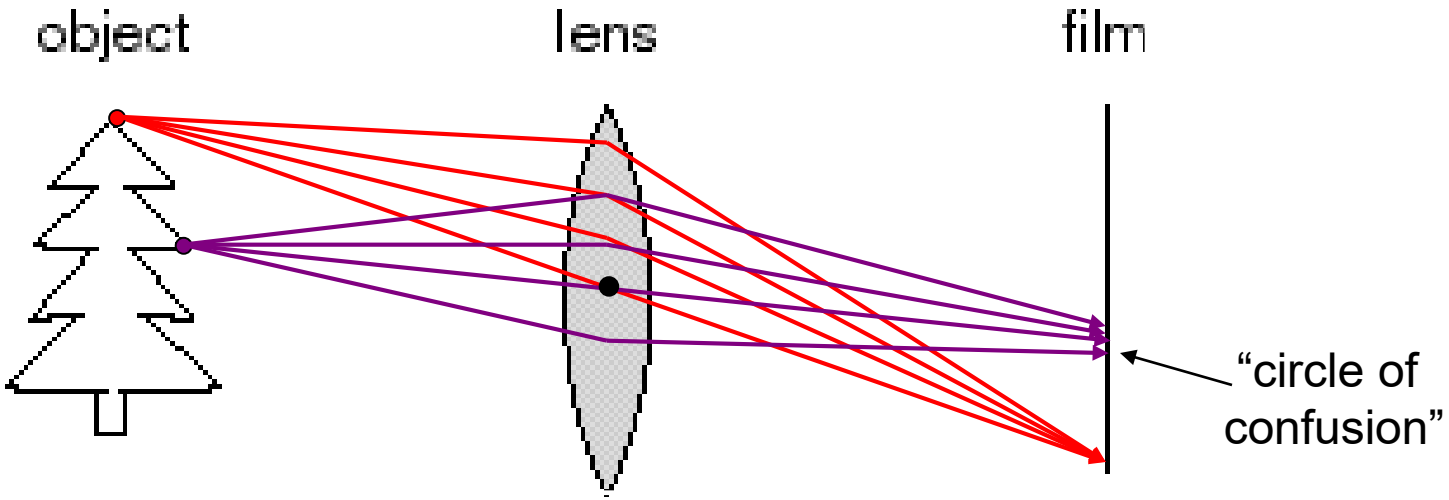
$$\frac{\|\mathbf{P}_3-\mathbf{P}_1\|\,\|\mathbf{P}_4-\mathbf{P}_2\|}{\|\mathbf{P}_3-\mathbf{P}_2\|\,\|\mathbf{P}_4-\mathbf{P}_1\|}$$

P_3

P_4

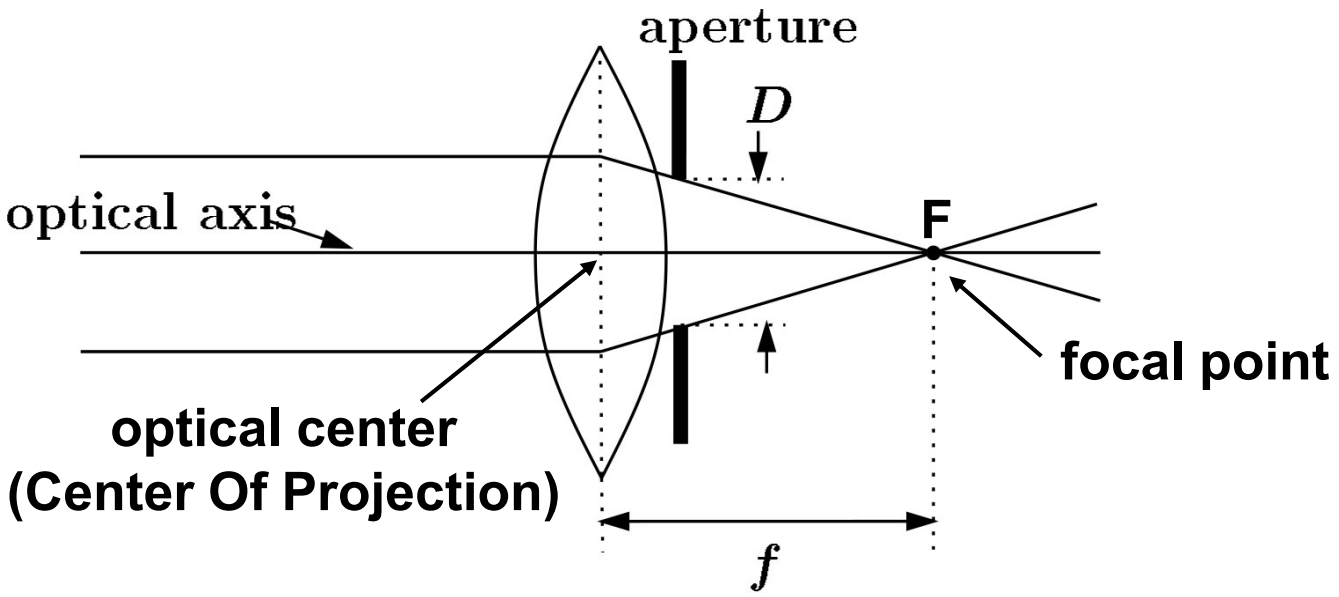
$\left(\frac{c}{n}\right)6$ = building height ?

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

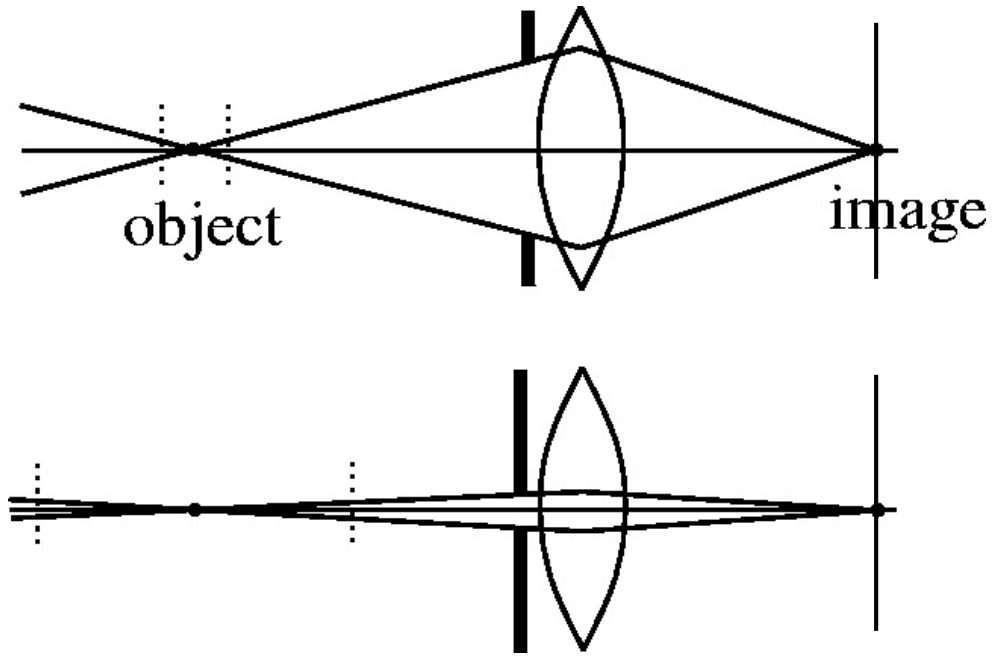
Focal length, aperture, depth of field



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

The aperture and depth of field



f / 5.6

f / 32

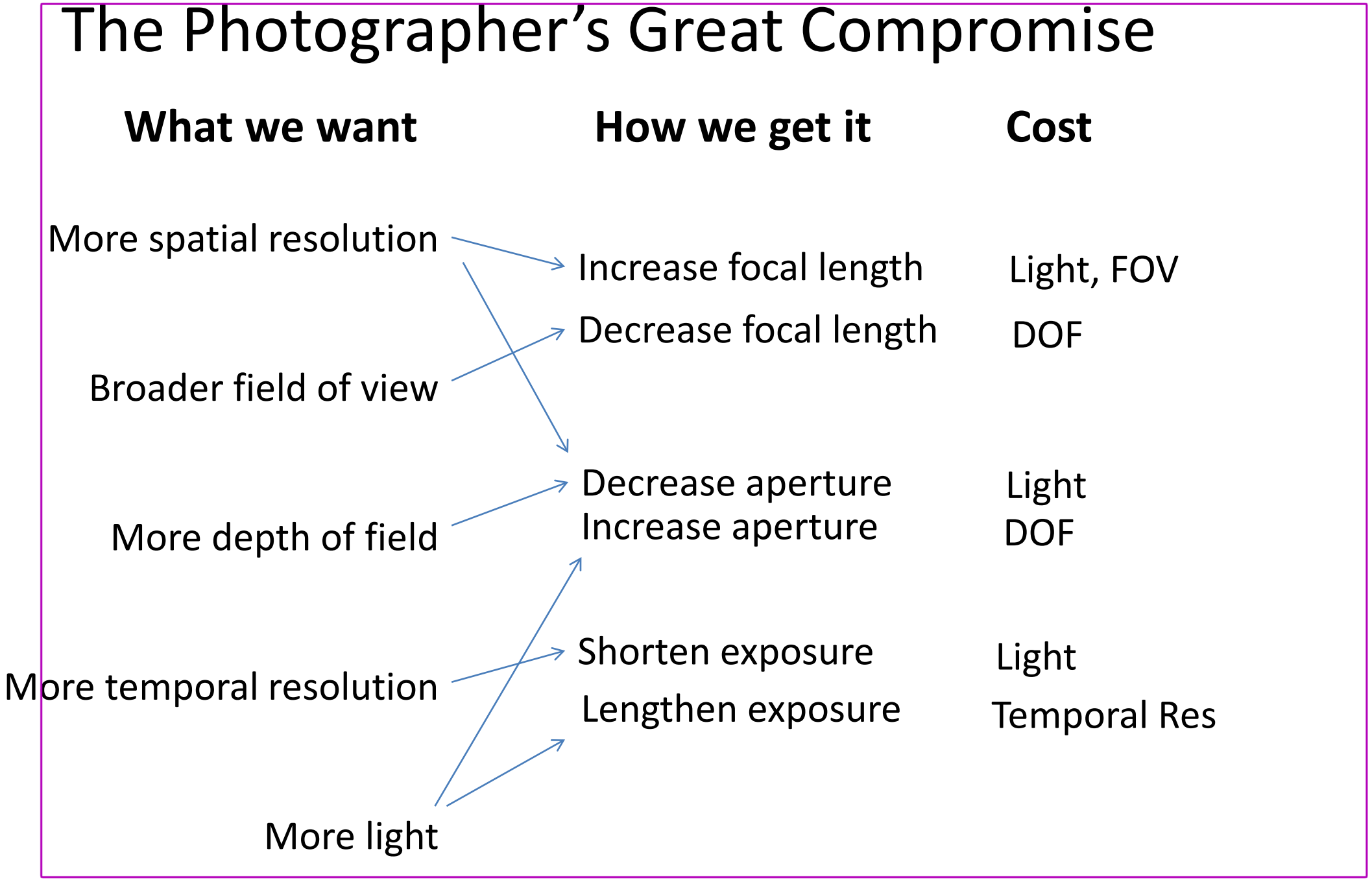
Main way to increase depth of field: Decrease aperture size

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



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

↳ ... "convolution"

$f[.,.]$

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

1. Filters

a) Compute the center four values of the filter response, using the filter and image given below.

Filter

-1	0	-1
0	4	0
-1	0	-1

*

Input Image

0	1	0	0
0	2	6	0
5	3	2	1
1	2	1	0

→

Filter Response

x	x	x	x
x	1	19	x
x	4	4	x
x	x	x	x

0 0 -6
0 12 0
-1 0 -1

-2 0 0
0 8 0
-2 0 -1

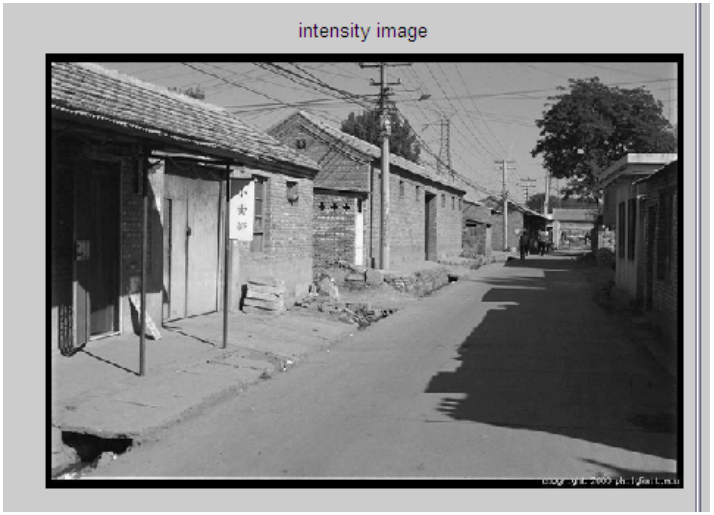
0 0 0
0 8 0
-5 0 -2

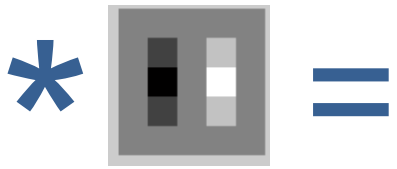
0 0 0
0 24 0
-3 0 -1

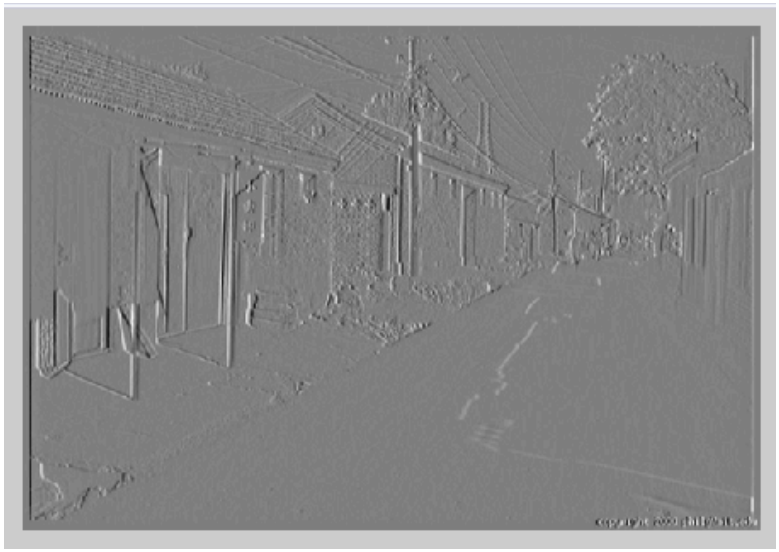
Filtering in spatial domain

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

edge detection filter? ✓







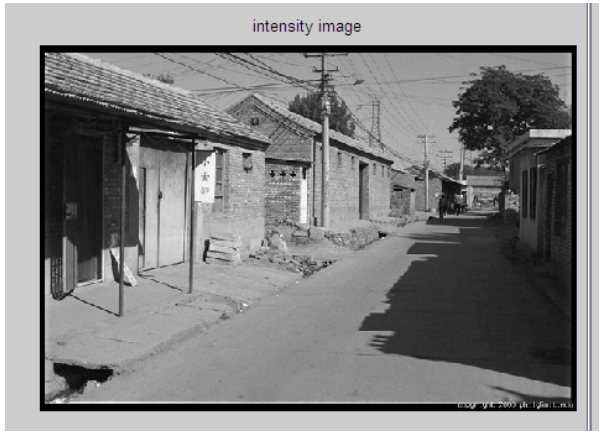
Filtering in frequency domain

- Can be faster than filtering in spatial domain (for large filters) ✓

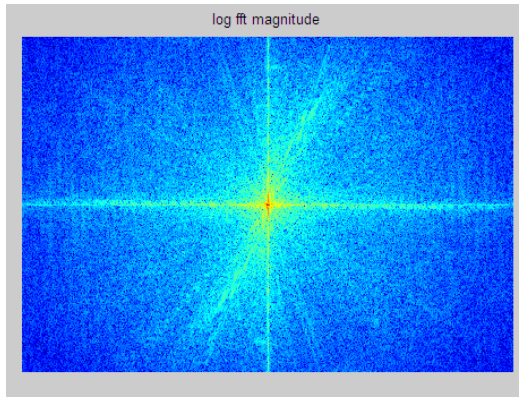
- Can help understand effect of filter

- Algorithm:
  1. Convert image and filter to fft (fft2 in matlab)
  2. Pointwise-multiply ffts
  3. Convert result to spatial domain with ifft2

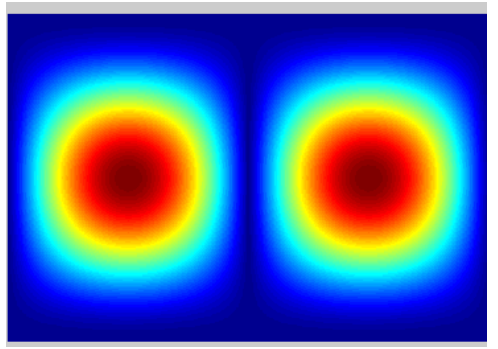
Filtering in frequency domain



FFT

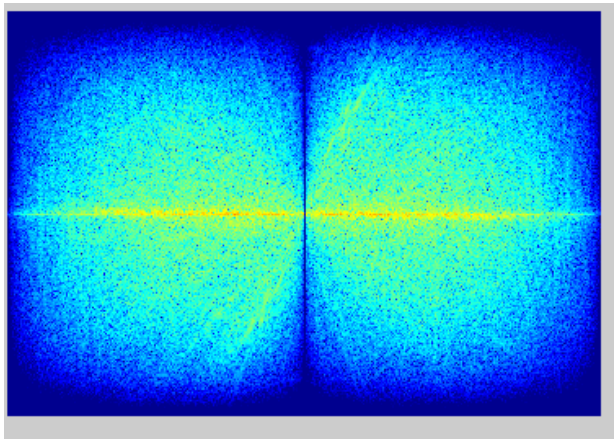


×

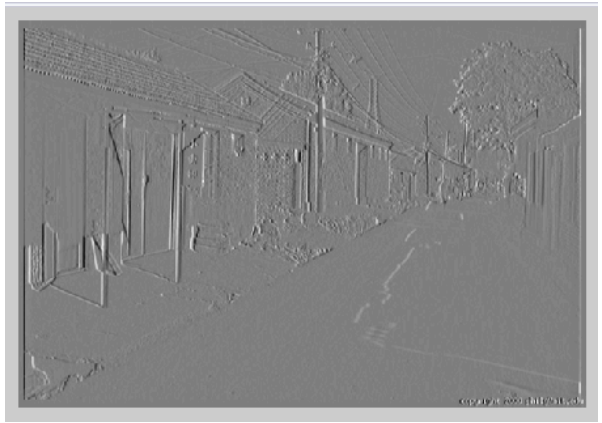


FFT

=



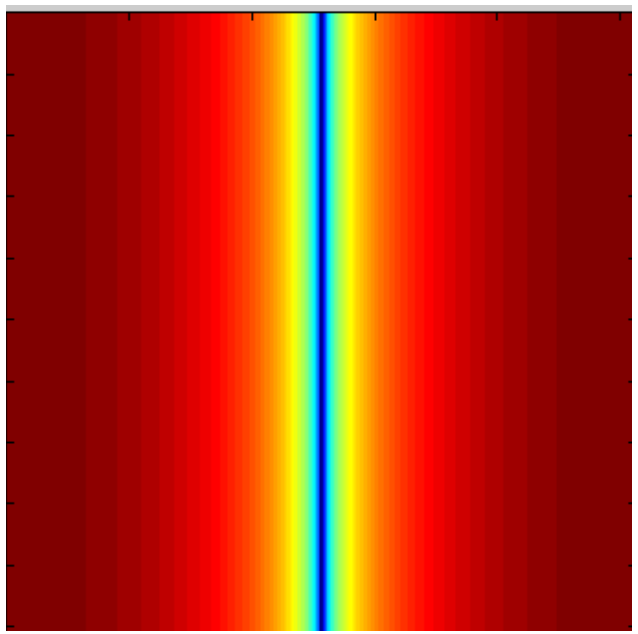
Inverse FFT



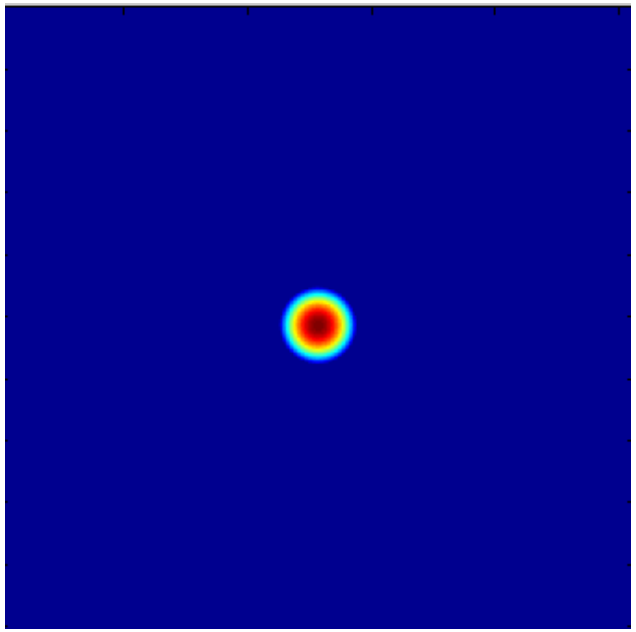
Filtering in frequency domain

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

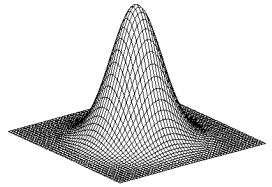
[-1 1]



FFT of Gradient Filter



FFT of Gaussian



Gaussian

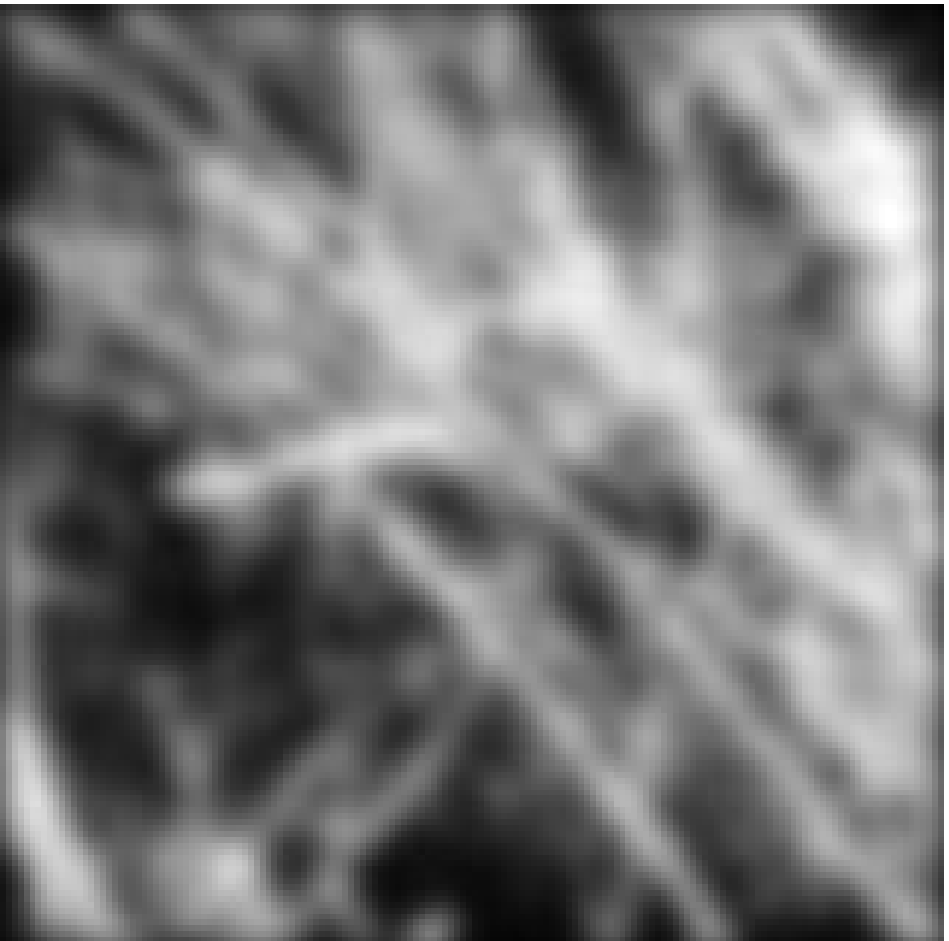
Filtering

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

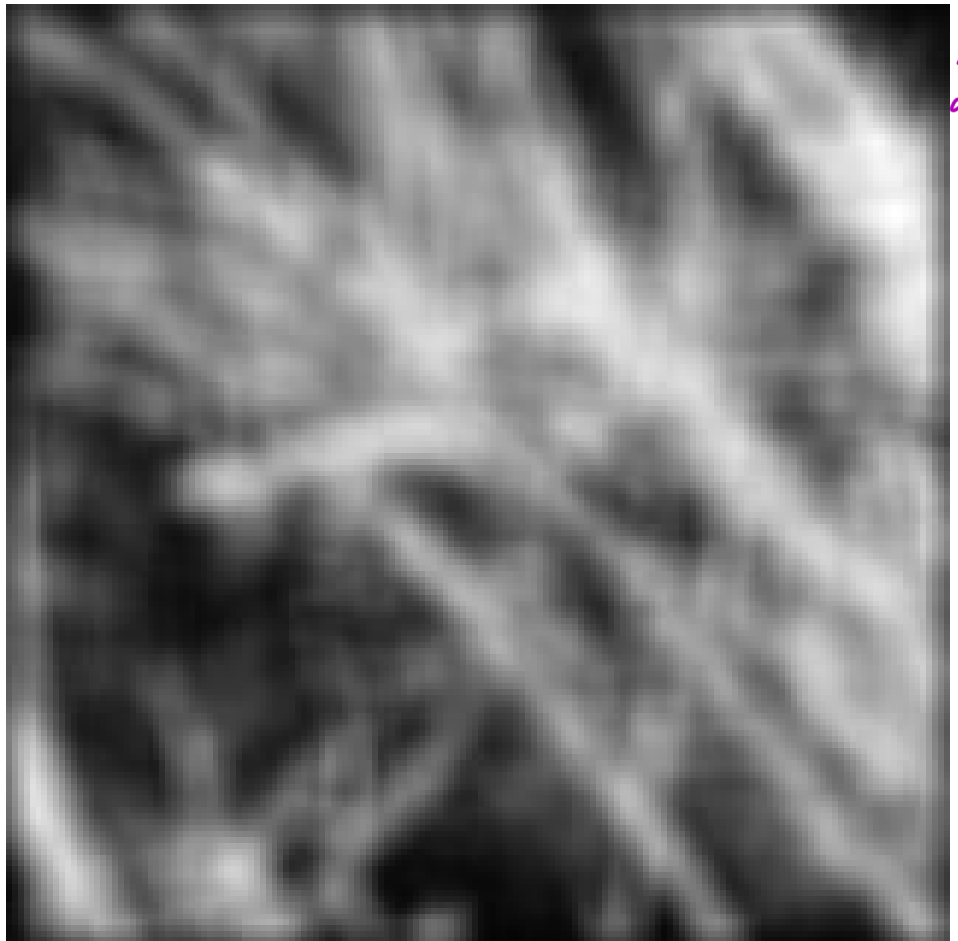
because the Gaussian assigns more weight to pixels closer to the center. The box gives all pixels in the window an equal weight.

Thus, when there is a harsh change in intensity, such as at and edge the box filter does not return as smooth, or as representative of an image as the Gaussian

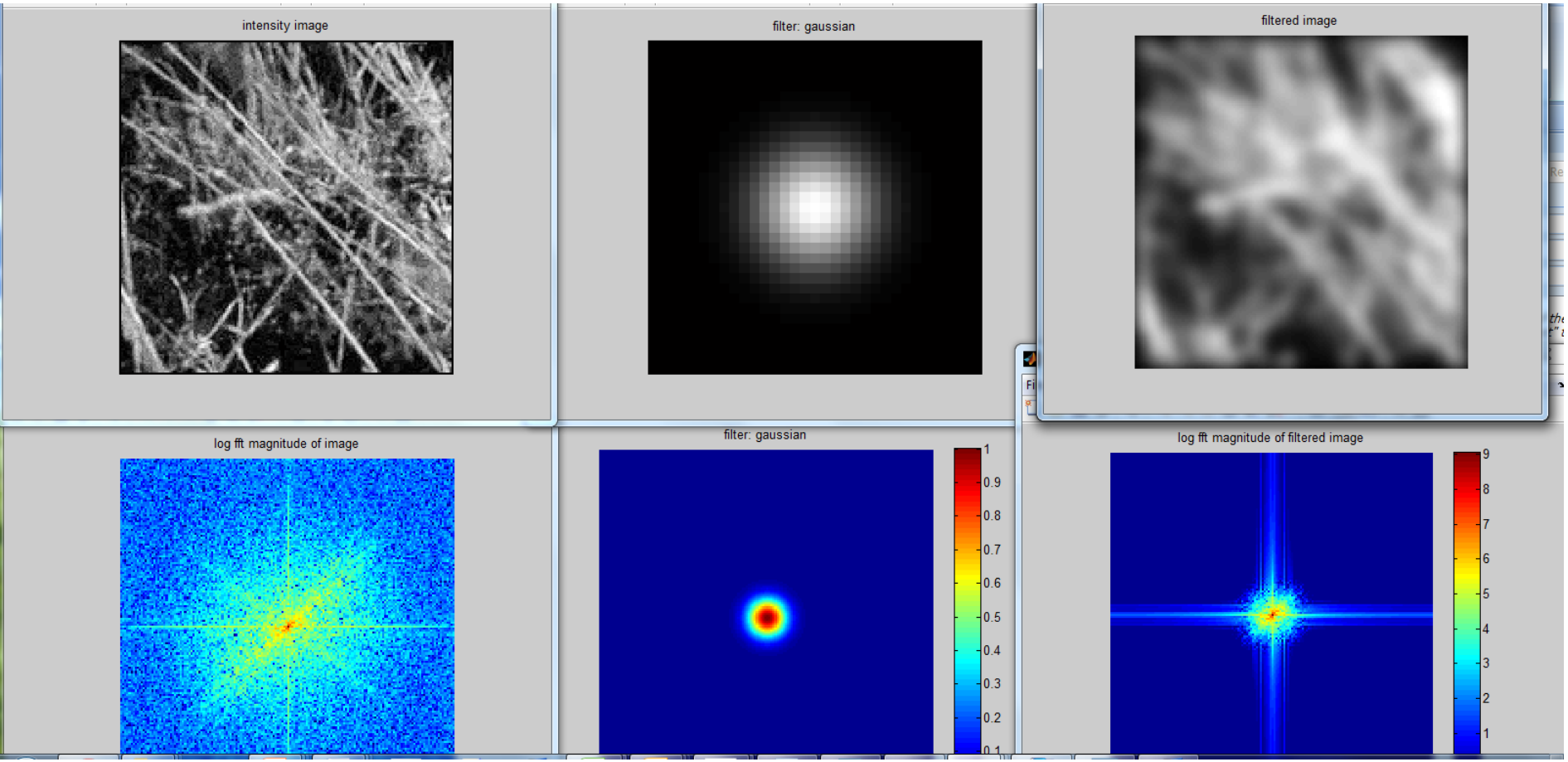
Gaussian 



Box filter 

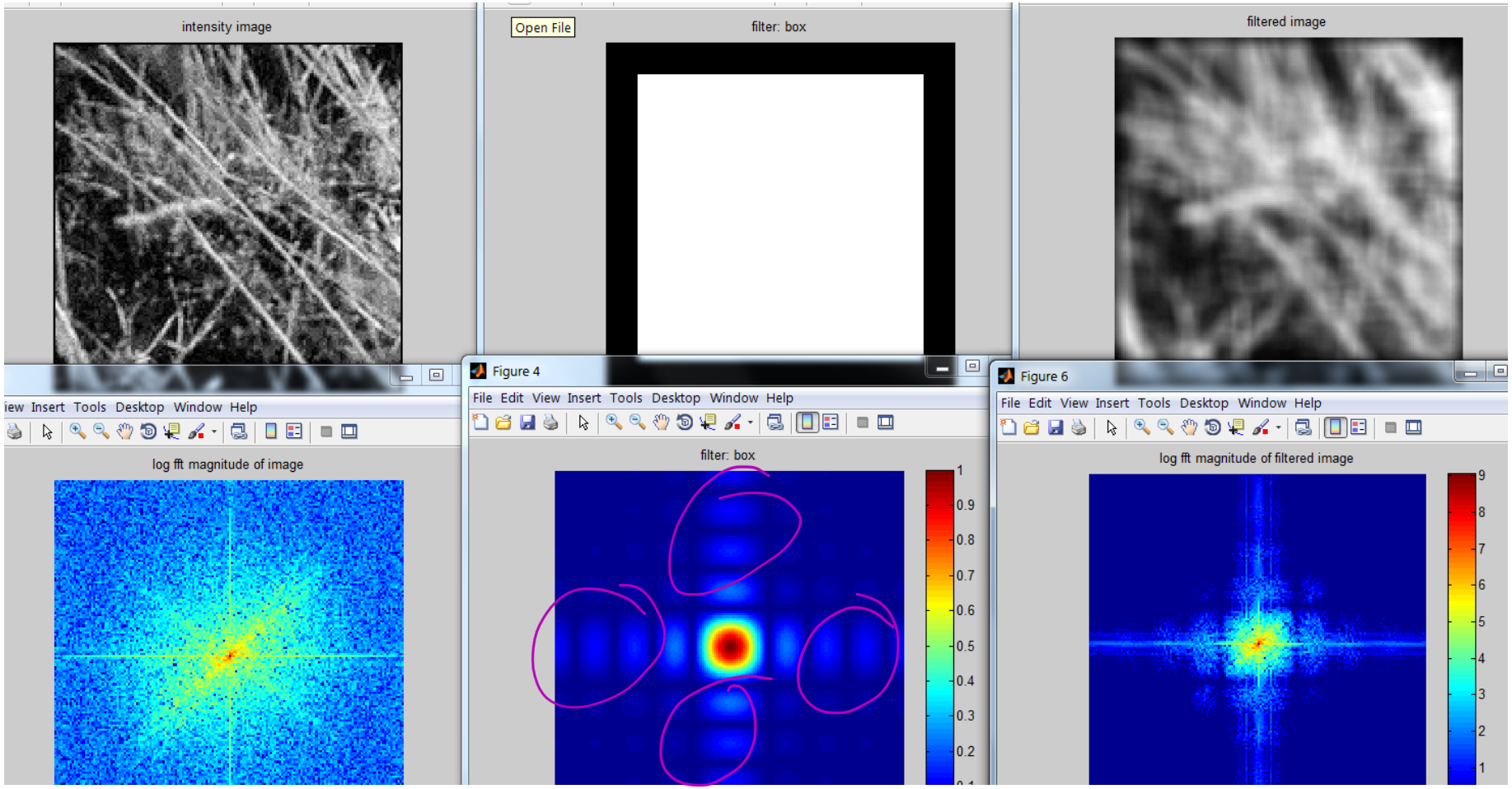


Gaussian


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

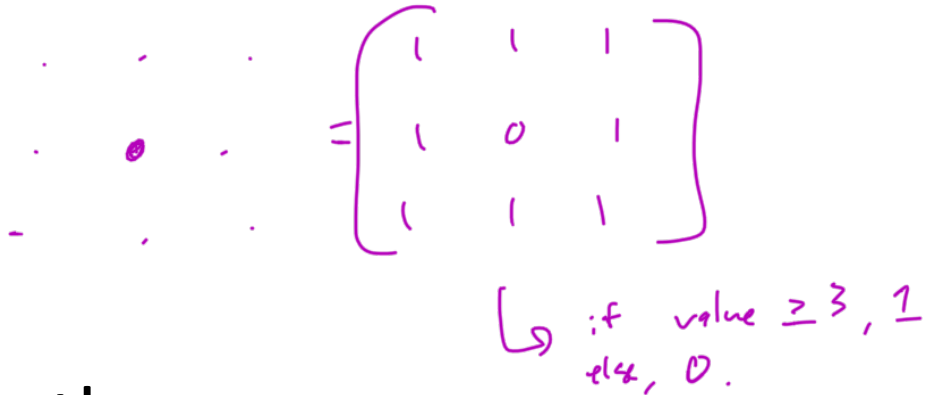
END of PART 1

Box Filter



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
```

given pixel

below center

above center

Question

3. Fill in the blanks:

Filtering Operator

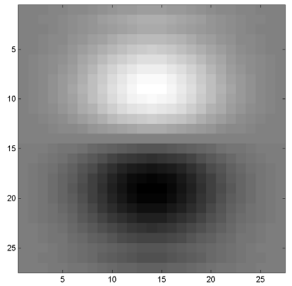
a) G = D * B

b) A = C * B

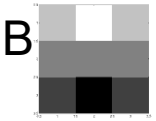
?c) F = D * _

d) H = D * C

A

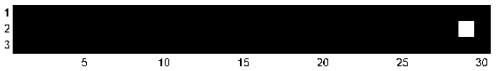


B

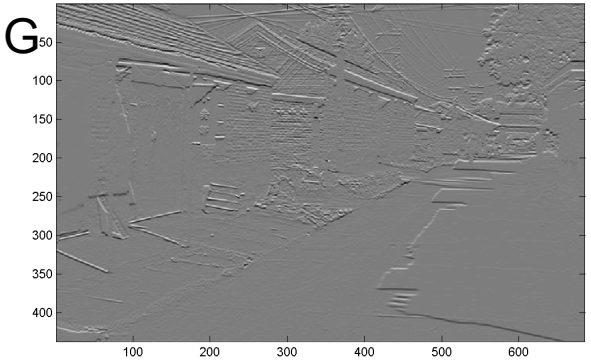


→ a warper?

E

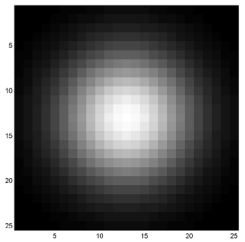


G



Gaussian

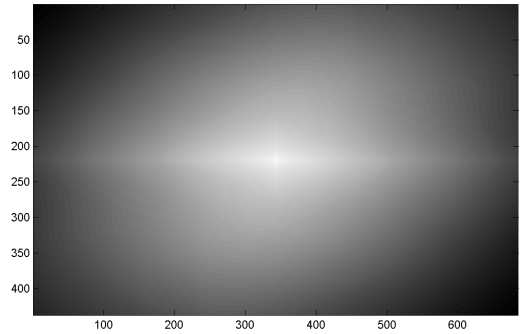
C



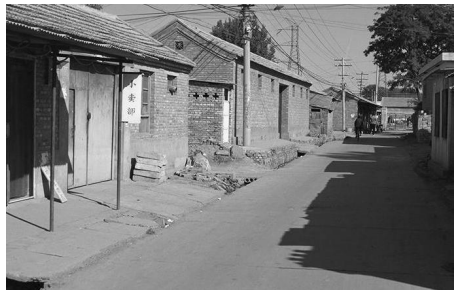
F

H

I

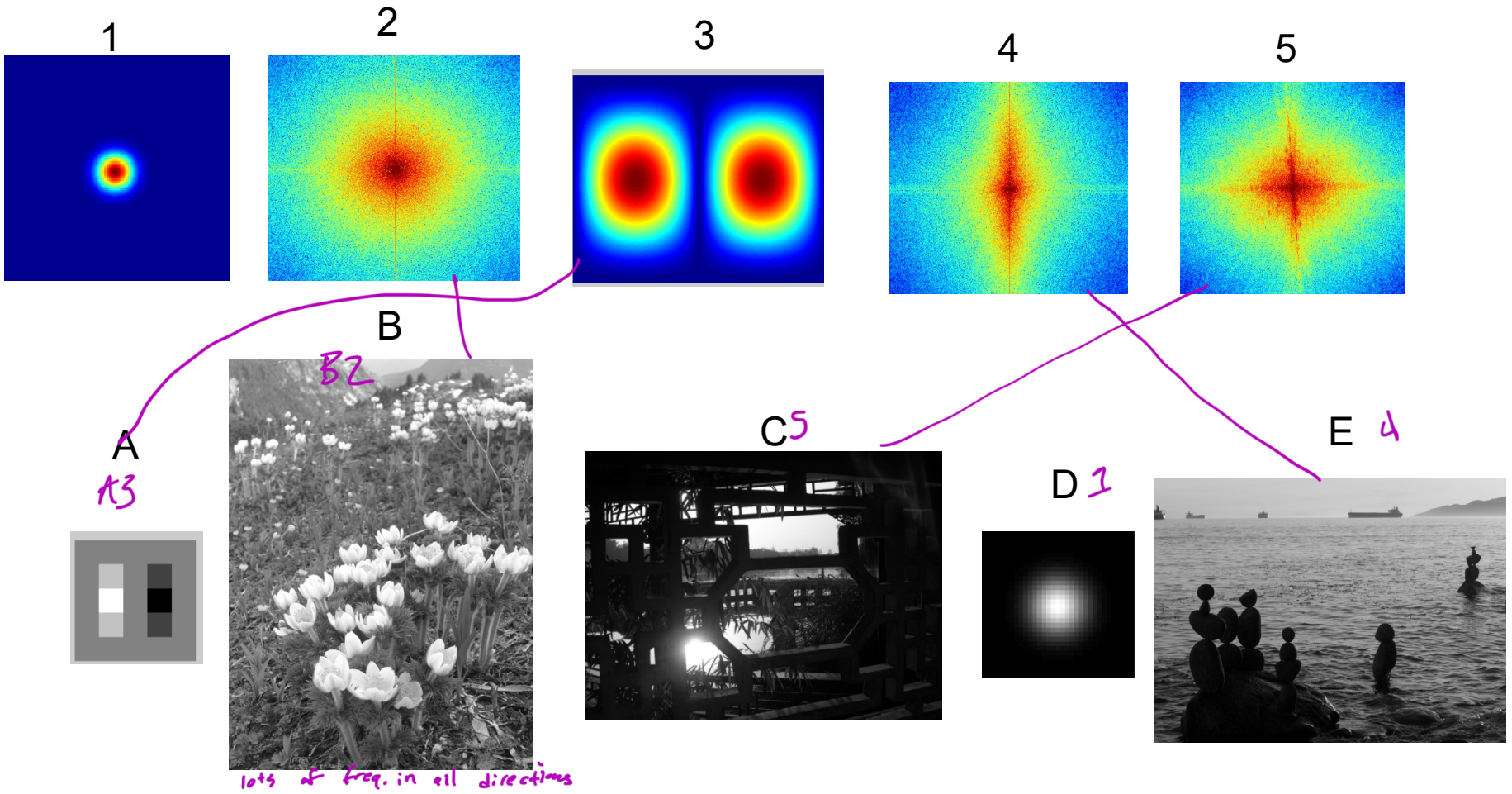


D



Question

Match the spatial domain image to the Fourier magnitude image



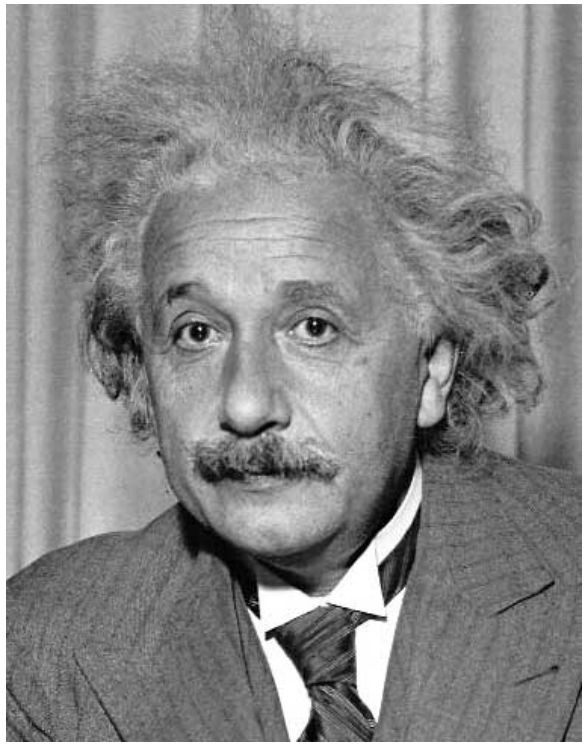
Matching with filters

→ Zero mean filter is the fastest, but SSD is the next fastest

- Goal: find  in image
- Method 2: SSD

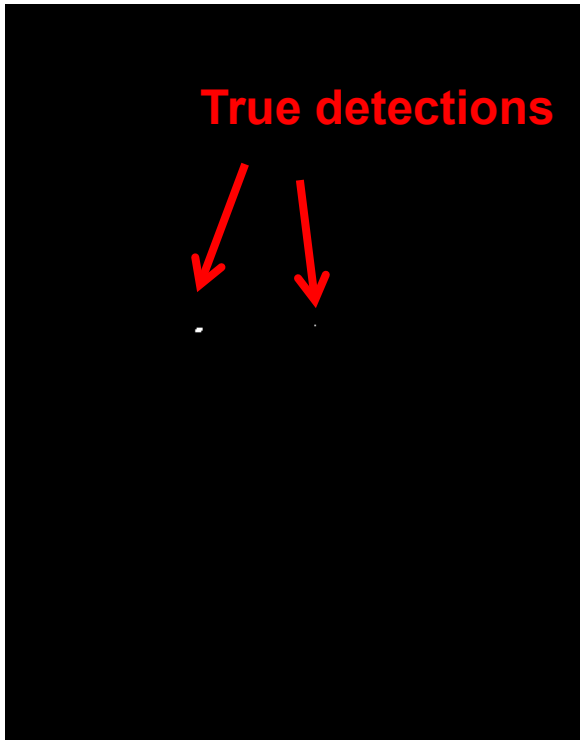
area in the image — corresponding template values — after filtering in that region

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



Input

1- sqrt(SSD)



Thresholded Image

Matching with filters

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

slowest

kind of like correlation, but we adjust for average intensity of template, average intensity of image patch, and contrast

mean template

mean image patch

$$h[m,n] = \frac{\sum_{k,l}(g[k,l]-\bar{g})(f[m-k,n-l]-\bar{f}_{m,n})}{\left(\sum_{k,l}(g[k,l]-\bar{g})^2 \sum_{k,l}(f[m-k,n-l]-\bar{f}_{m,n})^2\right)^{0.5}}$$

→ such that adding any constant value will have no effect

→ makes it such that multiplying our image intensities by a constant also cancels out

std of template and std of image patch

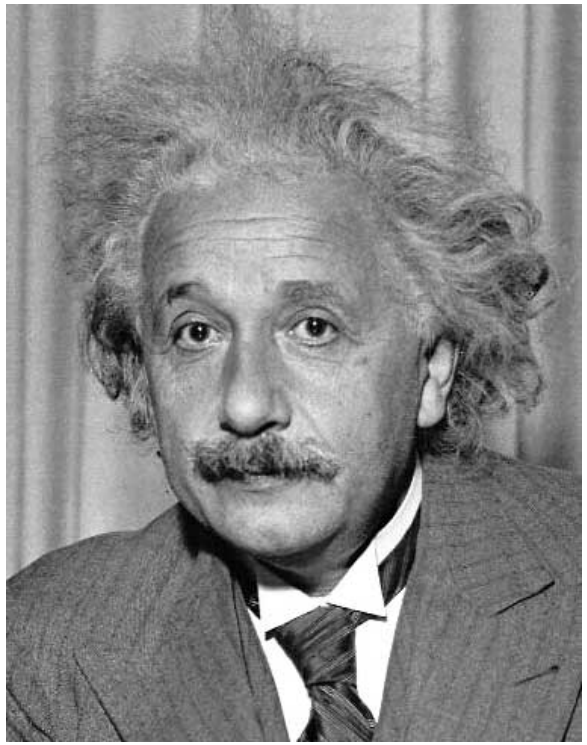
Matlab: `normxcorr2(template, im)`

Matching with filters

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

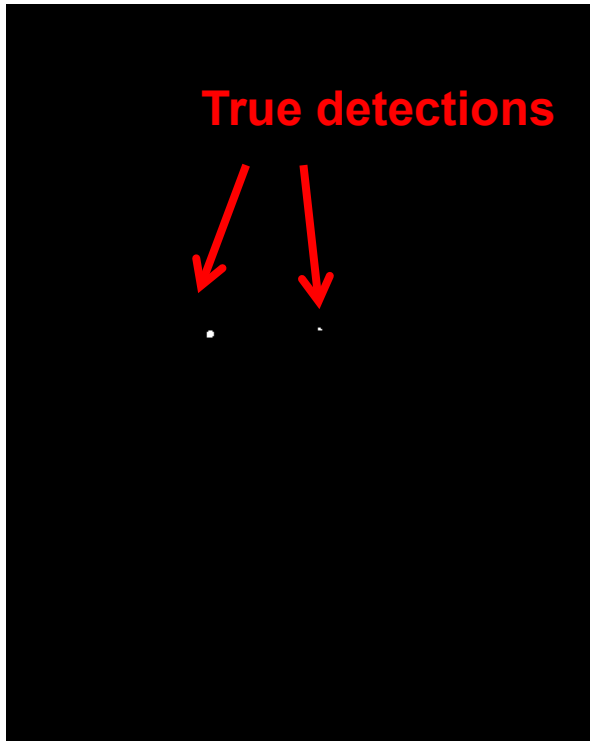
Key idea: the slowest method, and cannot be implemented with linear filters, as we are subtracting a different mean value for each image patch... but, is invariant to local average intensity

Unlike SSD, which is very sensitive to overall SSD



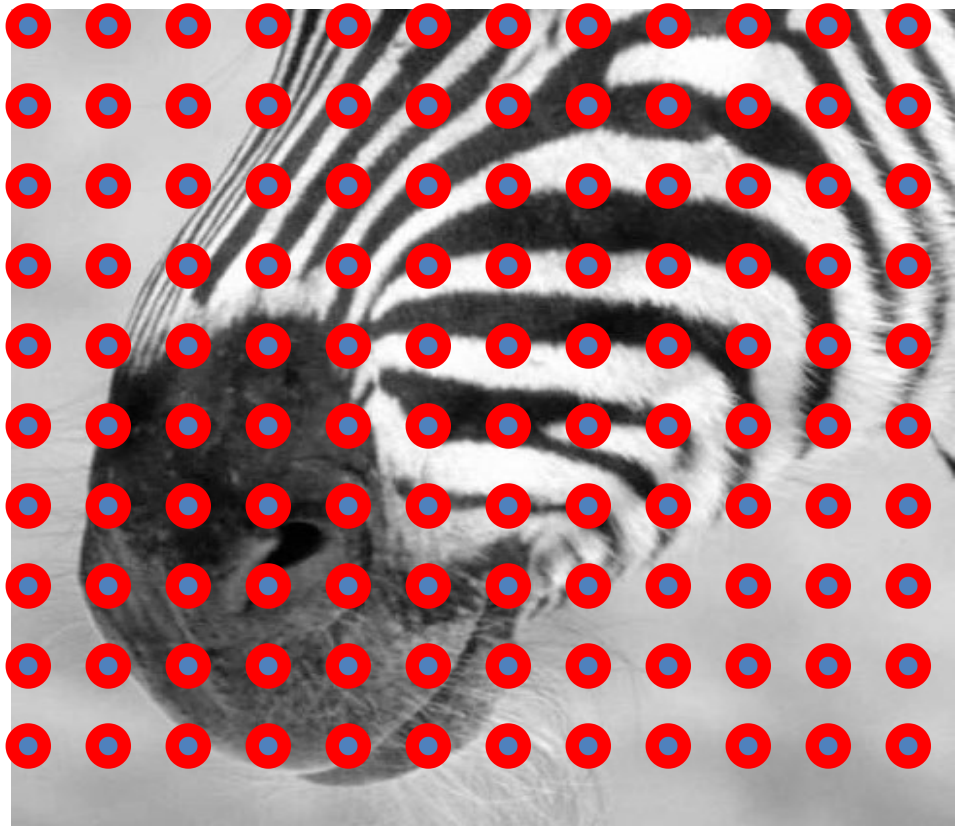
Input

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

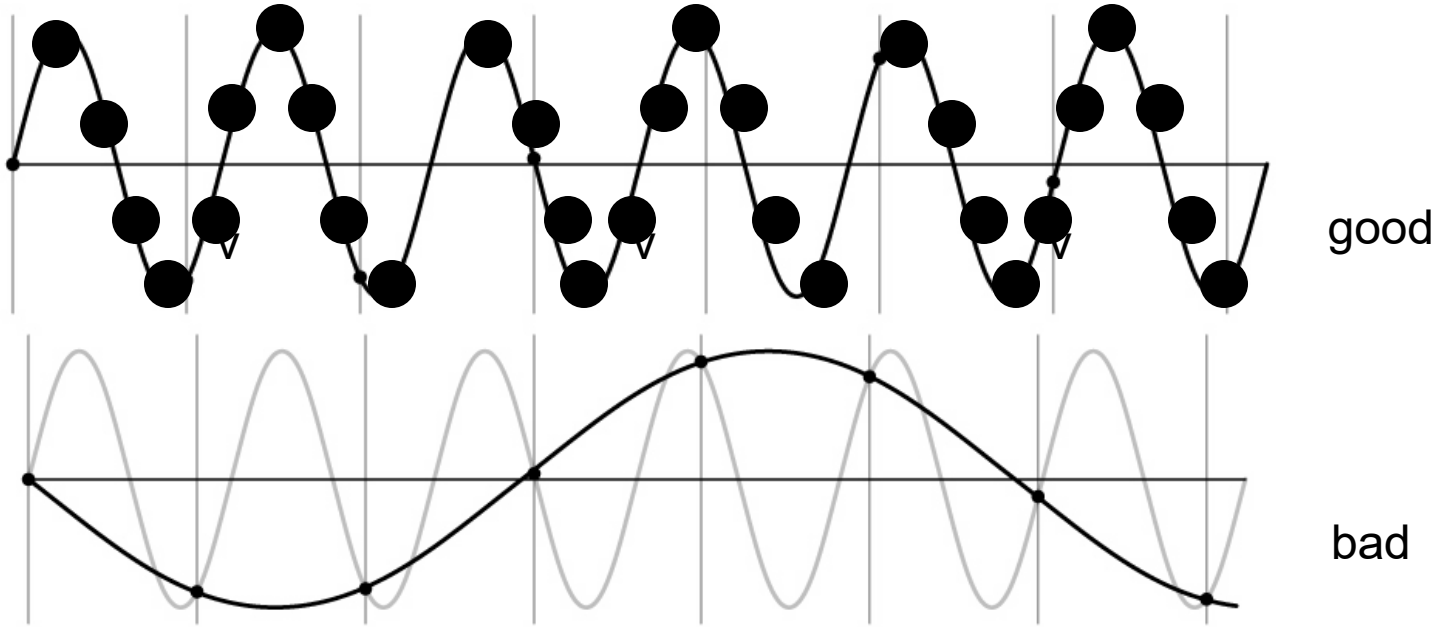
Problem: This approach causes “aliasing”

lots of artifacts..
- wagon wheels rolling in the wrong direction in movies
- checkerboards disintegrate in ray tracing
- striped shirts look funny on TV

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 ✓

In notes



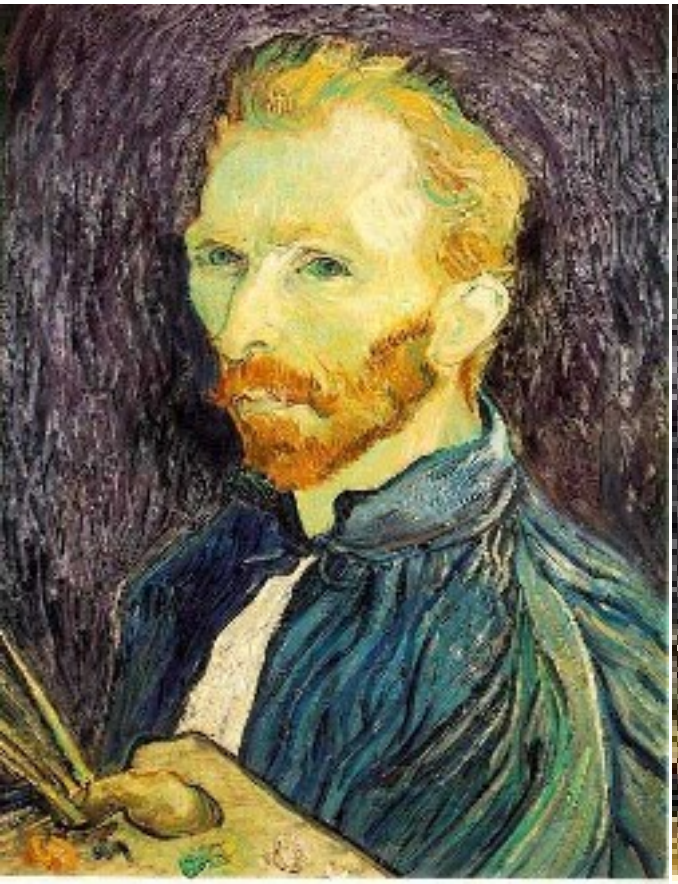
Algorithm for downsampling by factor of 2 ✓

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

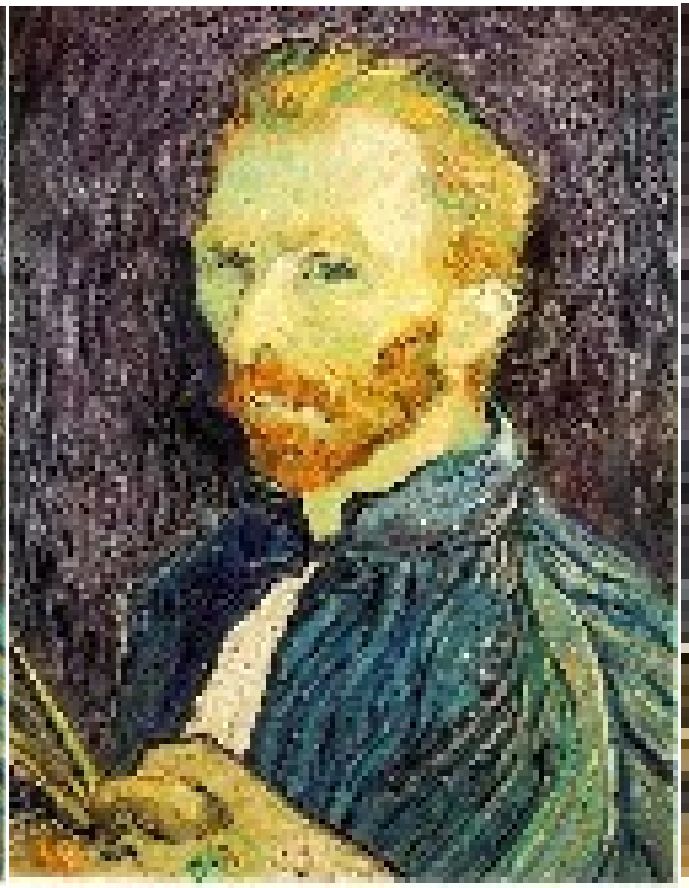
   im_blur = imfilter(image, fspecial('gaussian', 13, 2))
3. Sample every other pixel

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

Subsampling without pre-filtering



1/2

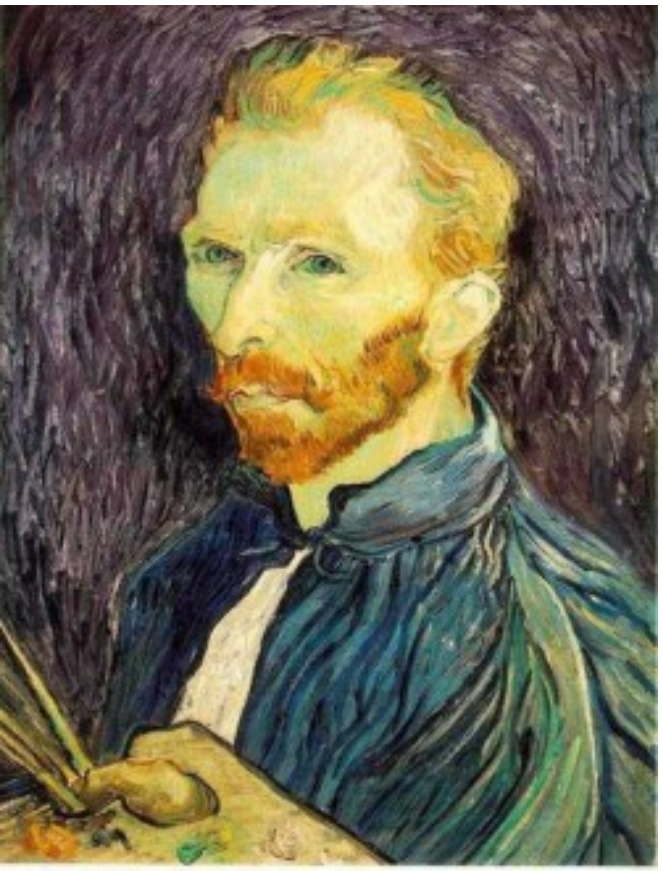


1/4 (2x zoom)

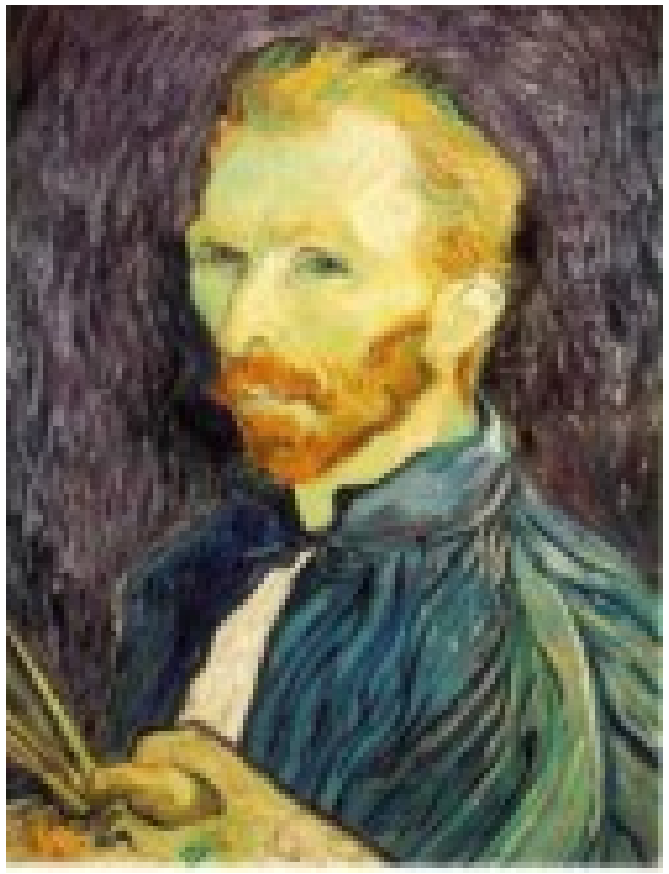
1/8 (4x zoom)

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?

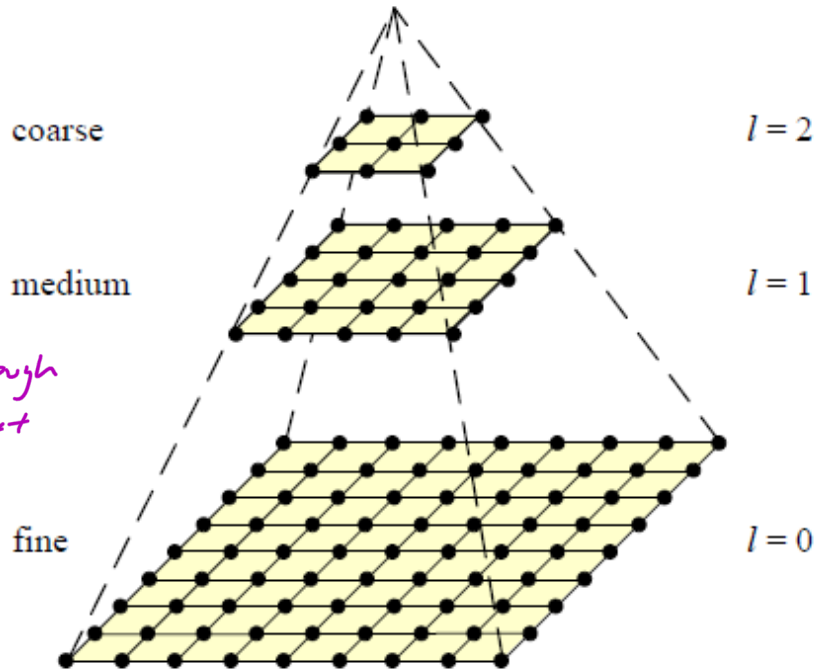
↳ in notes...
- "because it preserves low frequencies, where the majority of image information lies"

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

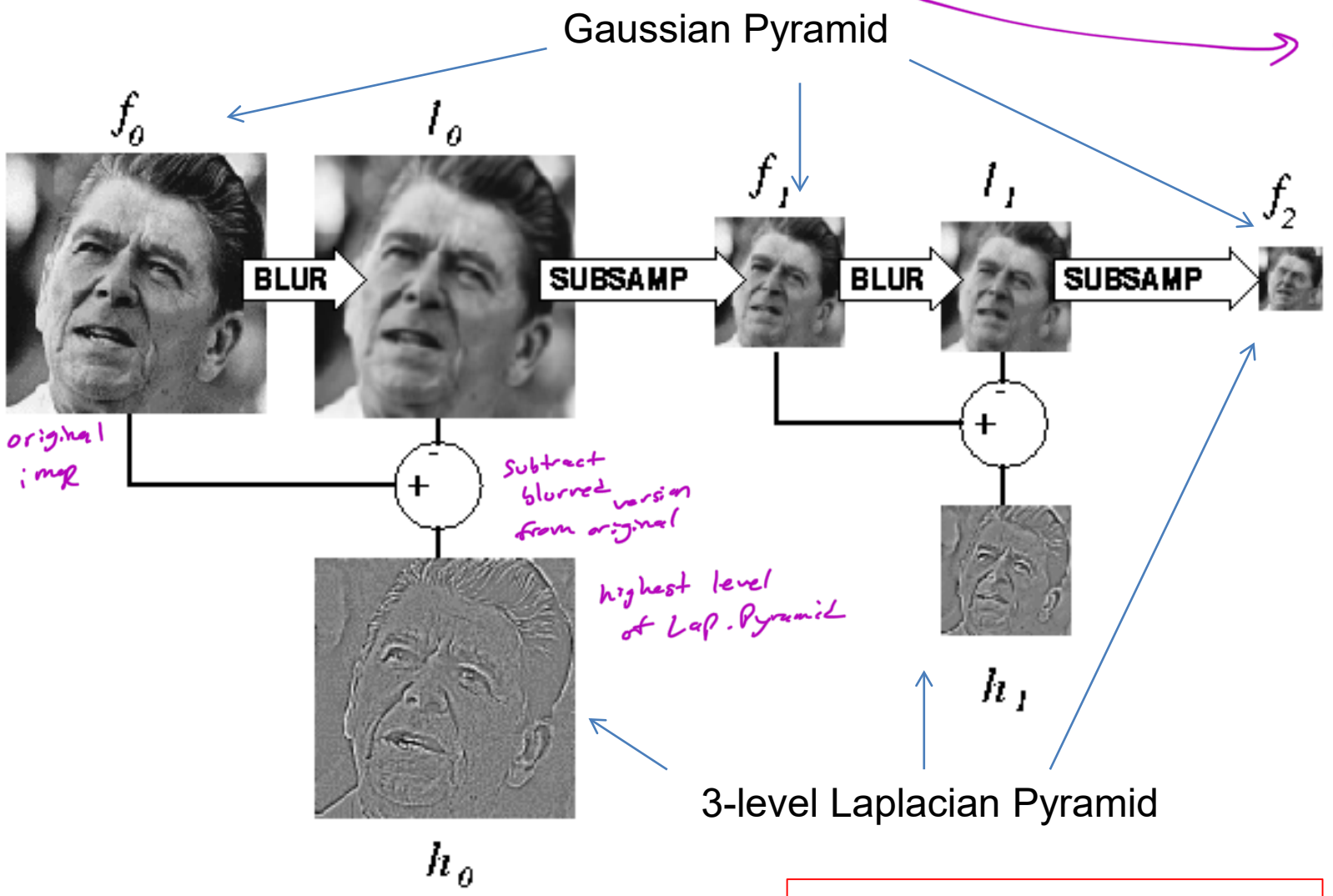
Gaussian pyramid

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

just keep smoothing and downsampling to get a pyramid of low resolution to high resolution images...
↳ then, to detect something, you could run a template through the whole pyramid = eyes at different scales



Computing Gaussian/Laplacian Pyramid



unit impulse - Gaussian

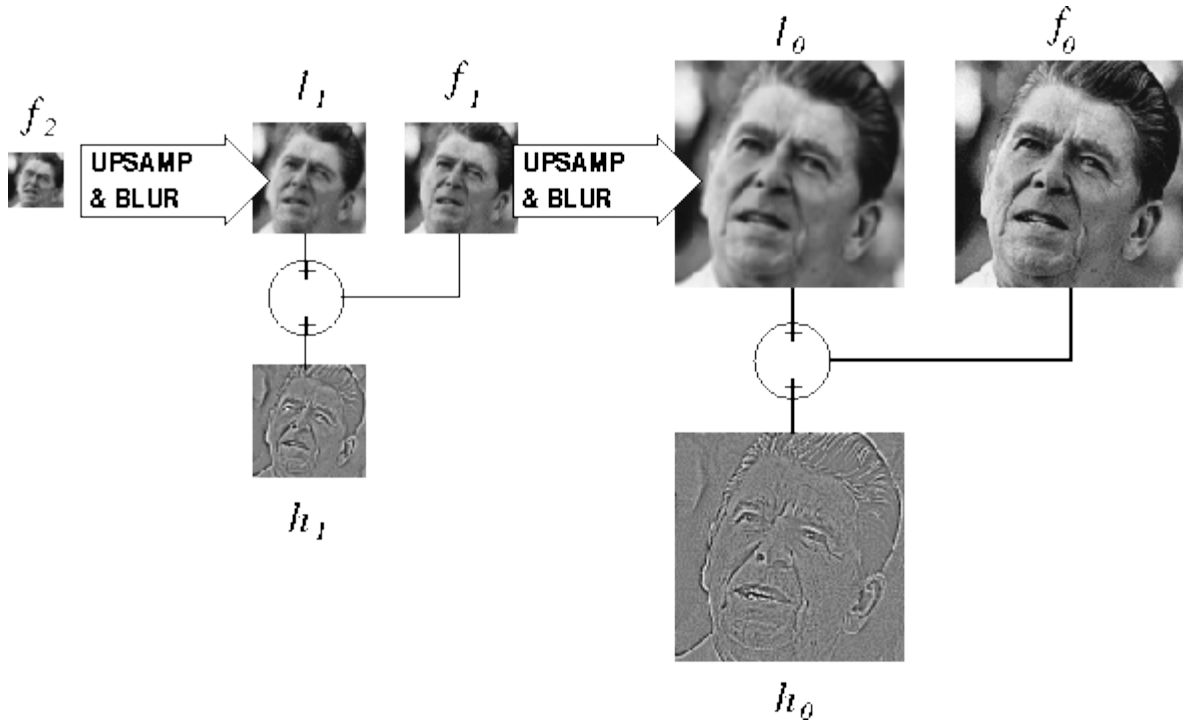
lowest level of Laplacian pyramid = smallest, subsampled image

Useful figure to remember

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

Image reconstruction from Laplacian pyramid

makes sense

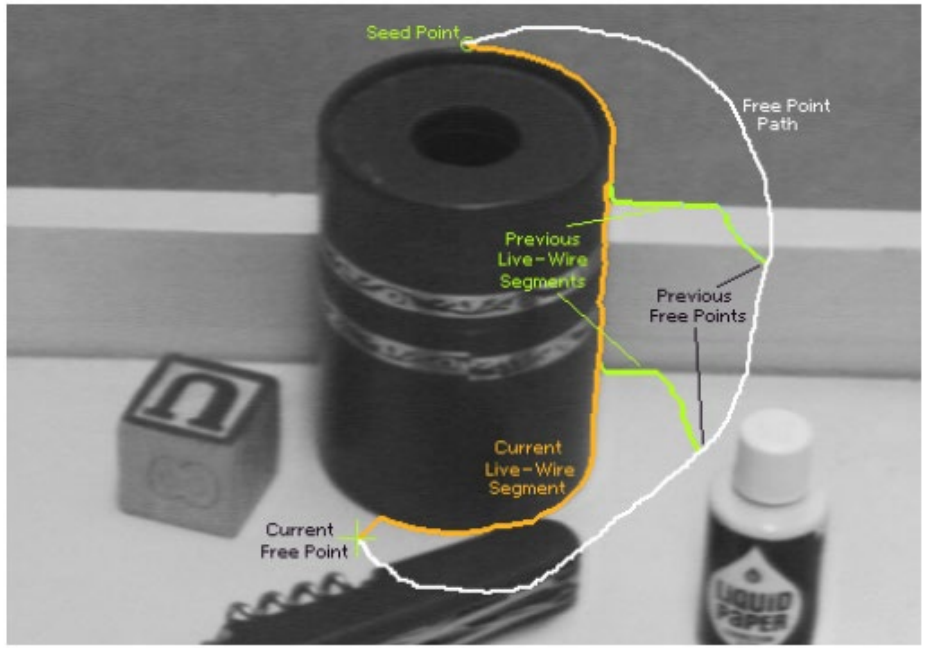


4. Region selection and compositing

- 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

want user to be able to form long boundaries

↳ How do you define what a good boundary is?

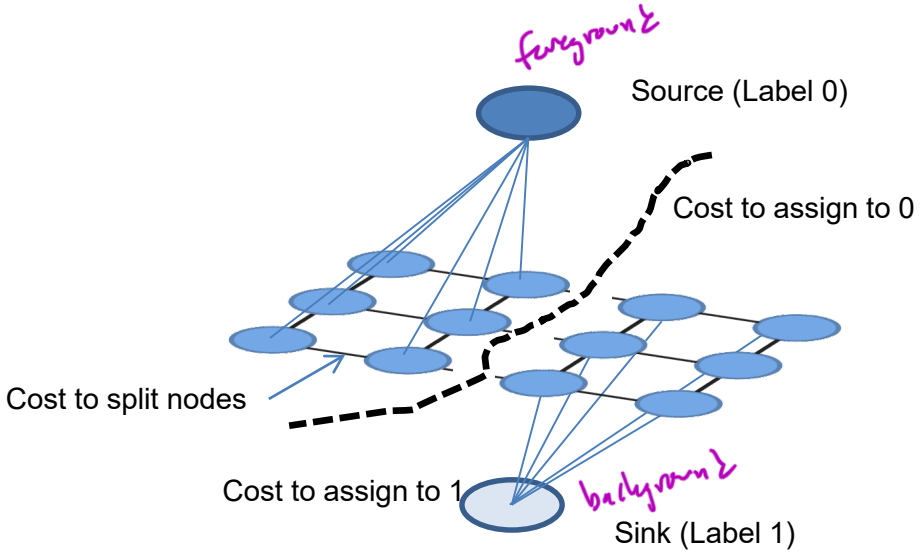
↳ boundary cost between neighboring pixels

↳ ① lower if edge present
② lower if gradient is strong
③ lower if gradient is in direction of boundary

3 things that obviously would make for a good boundary

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

Combines magic wand and intelligent scissors a bit to do something that will use both user input and find a boundary with the background

Assign some pixels to background, some to foreground. Find a cut that goes through as few edges as possible to make the partition.

① Define Graph

② Set weights for foreground and background using probability of going to fore or back given color } unary potential

③ Set weights for edges between pixels

cost
- k_1 = always pay some cost for splitting two pixels. Ensures we get a short, roundish boundary
- k_2 = encoding gradients. If the colors of two pixels are similar, the cost will be close to 0... thus k_2 if they are very different, then the second term will be very small

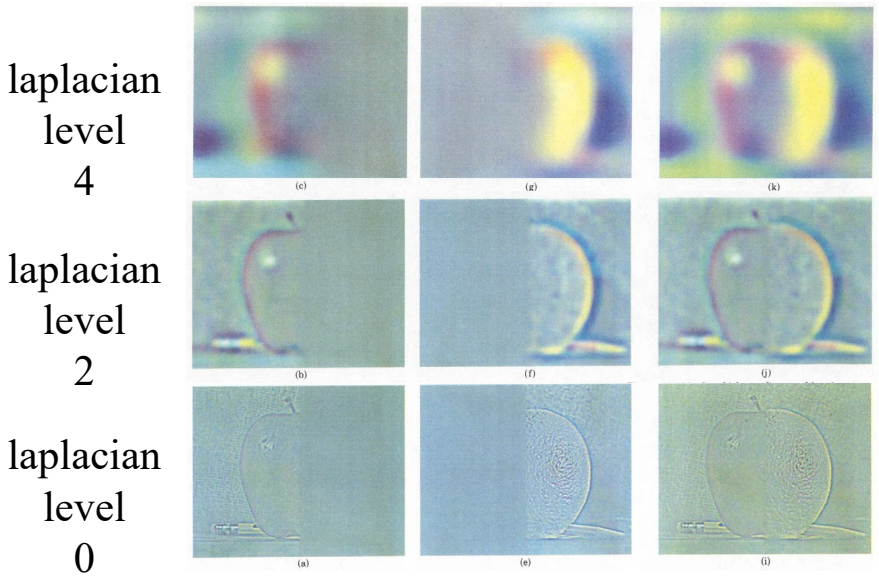
④ Apply min-cut/max-flow algorithm

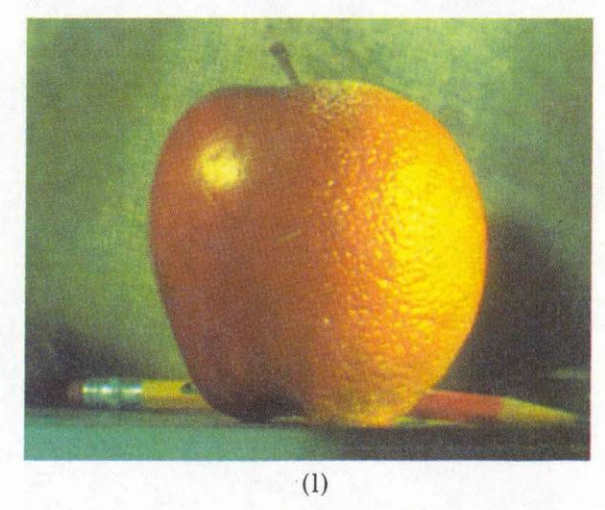
⑤ Return to step 2, with current labels to compute foreground, background labels

2 to 3 iterations give convergence

Compositing and Blending

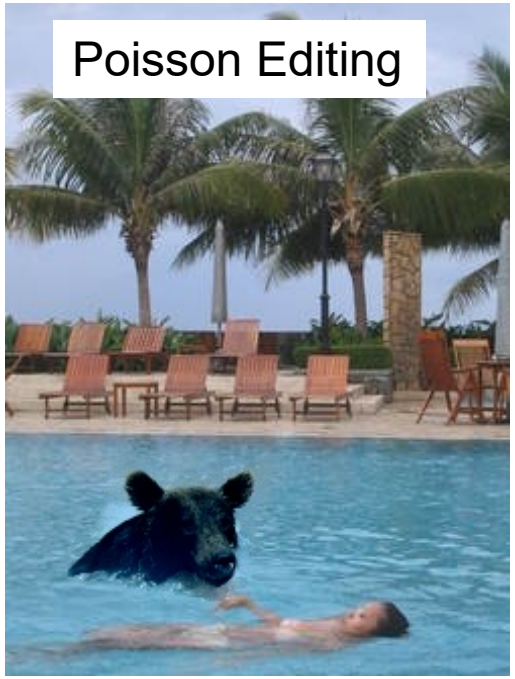
- Feathering: blur mask around its edges
- Laplacian blending: blend low-frequency slowly, high frequency quickly
  - Blend with alpha mask values ranging from 0 to 1



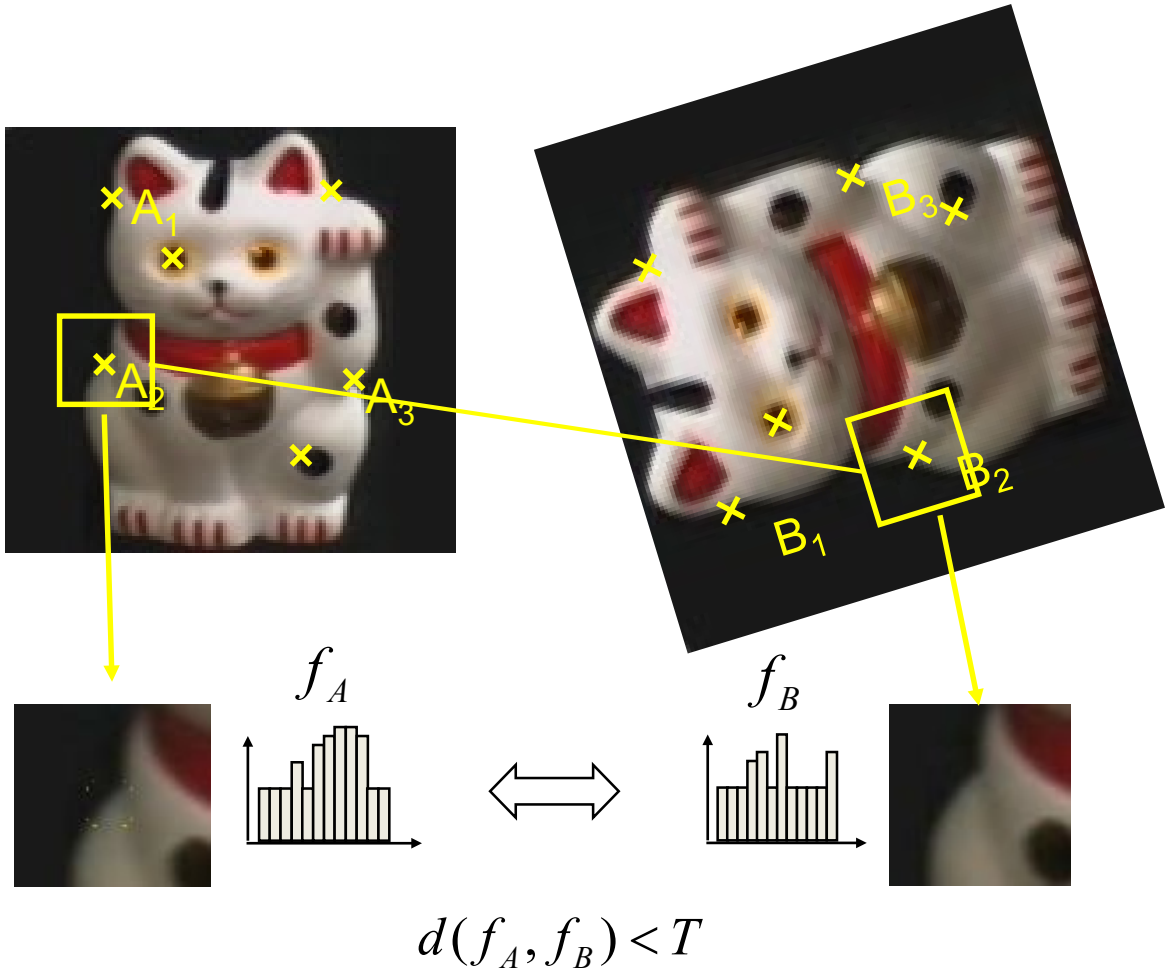


Question

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?

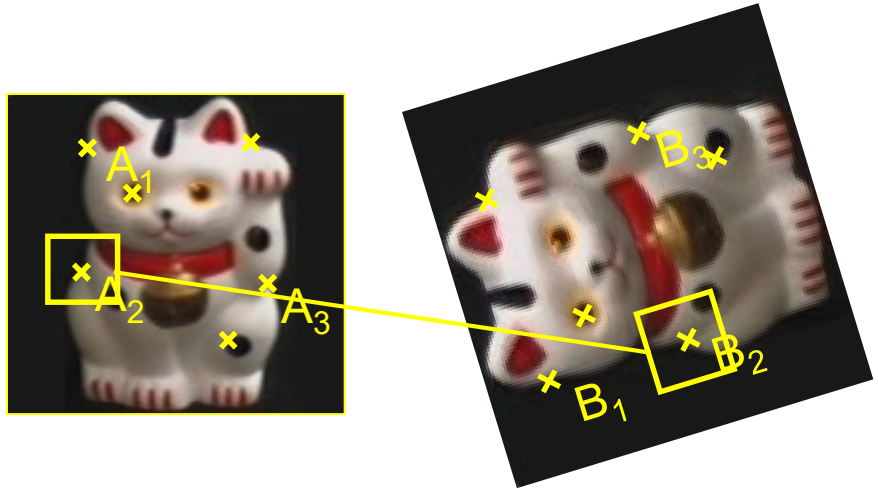


Keypoint Matching



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**

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

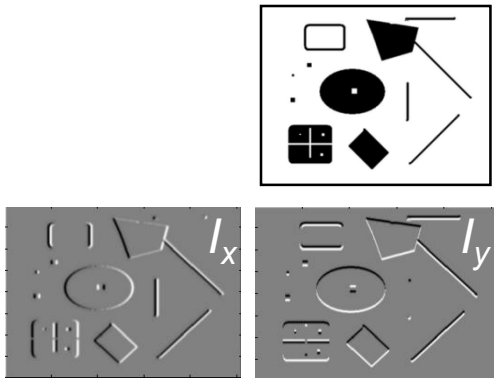
Minimize wrong matches

Harris Detector [Harris88]

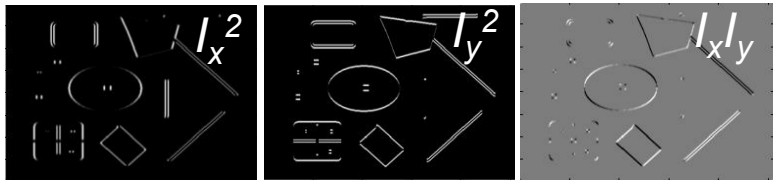
- Second moment matrix

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

1. Image derivatives



2. Square of derivatives

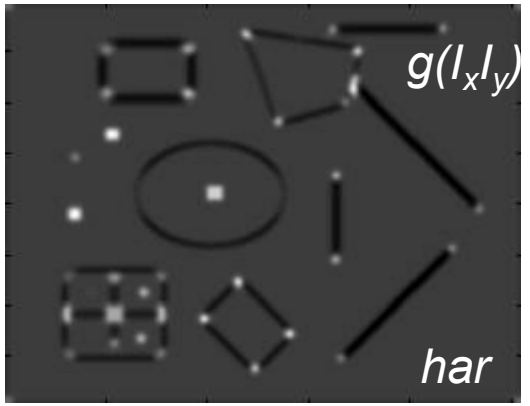


3. Gaussian filter $g(\sigma_I)$

$$\det M = \lambda_1 \lambda_2$$
$$\text{trace}\, M = \lambda_1 + \lambda_2$$

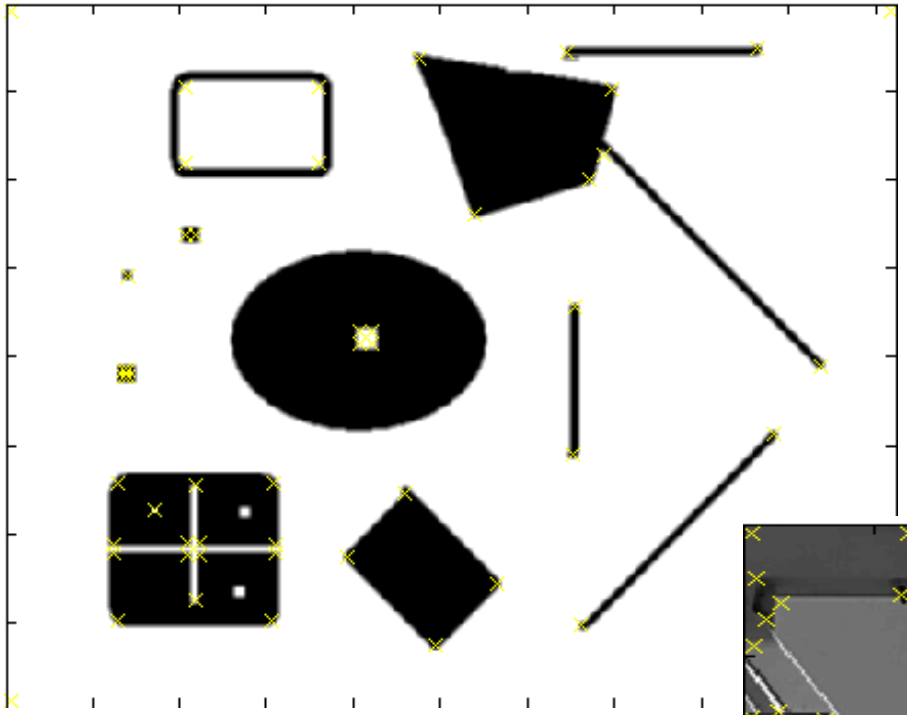
4. Cornerness function – both eigenvalues are strong

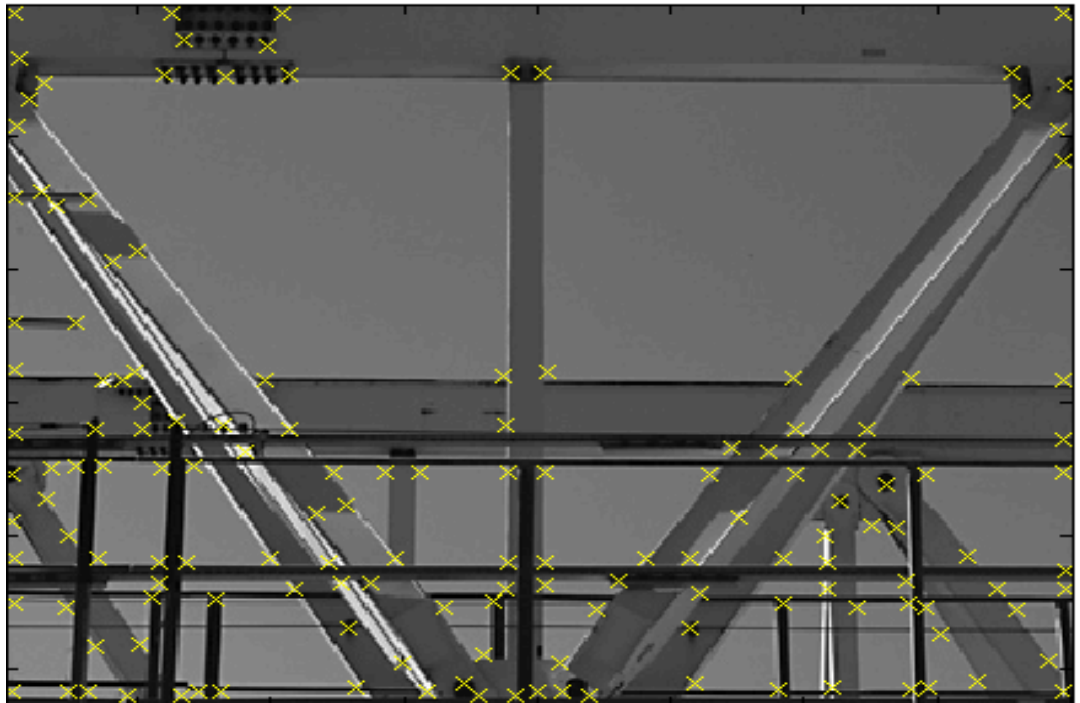
$$har = \det[\mu(\sigma_I, \sigma_D)] - \alpha[\text{trace}(\mu(\sigma_I, \sigma_D))^2] =$$
$$g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$



5. Non-maxima suppression

Harris Detector – Responses [Harris88]





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

Applications

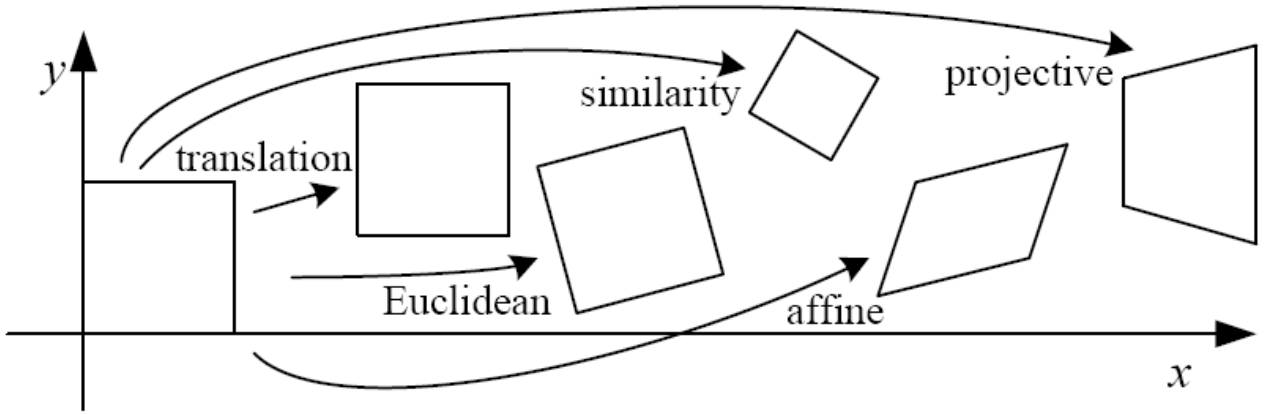
- Image stitching
  - Matching keypoints
  - Solving for homography
  - RANSAC

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

5. Solving for transformations

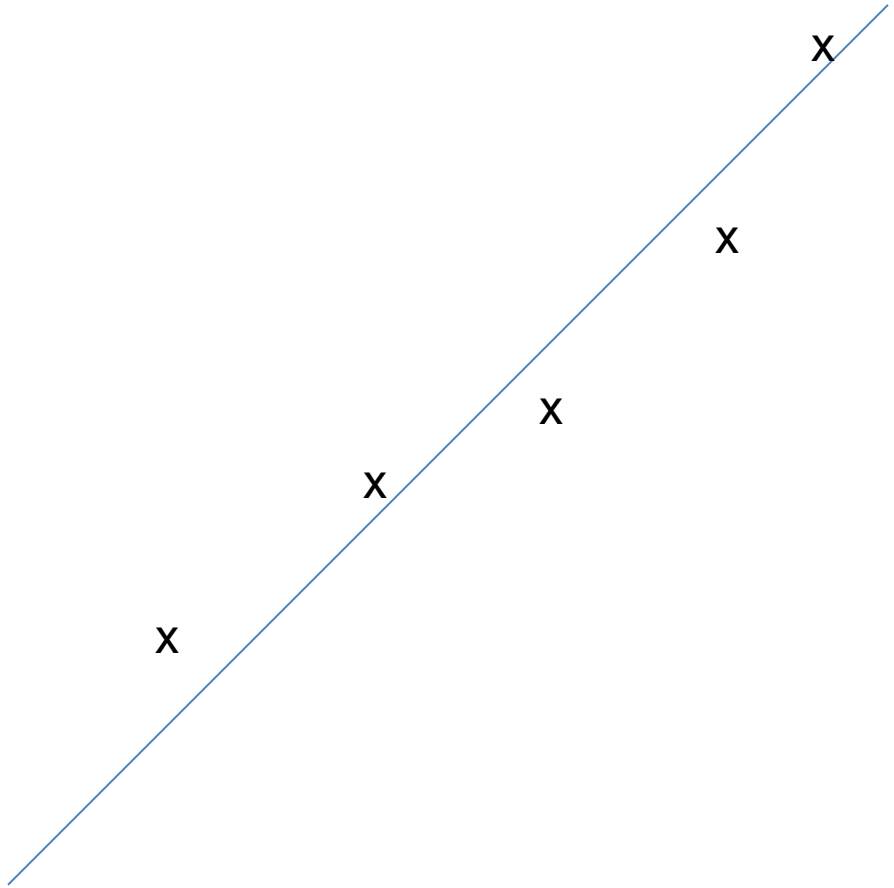
- Map between 2D coordinates using linear projection

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	4	angles $+\cdots$	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times3}$	6	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times3}$	8	straight lines	



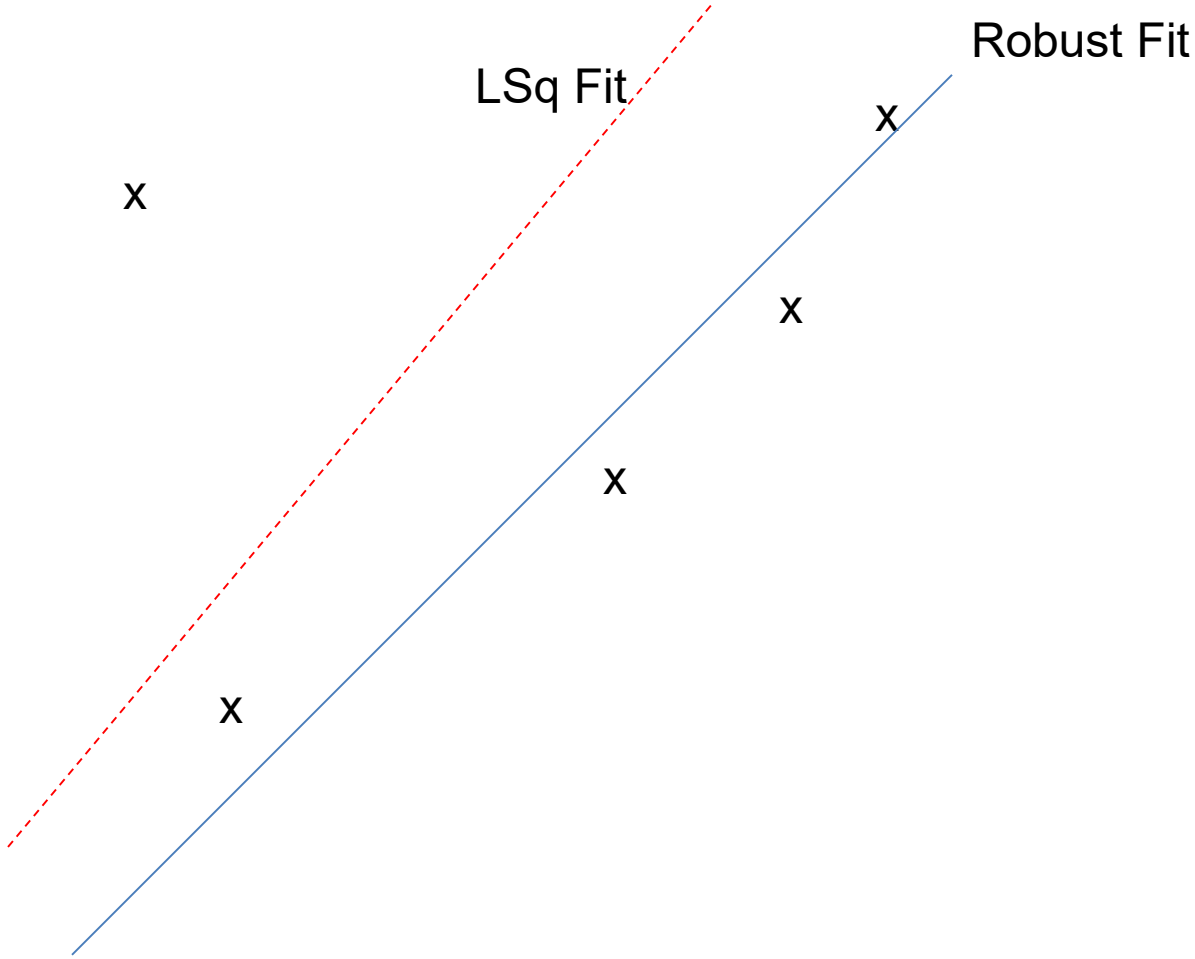
Least-squares Solving

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



RANSAC: RANdom SAmple Consensus

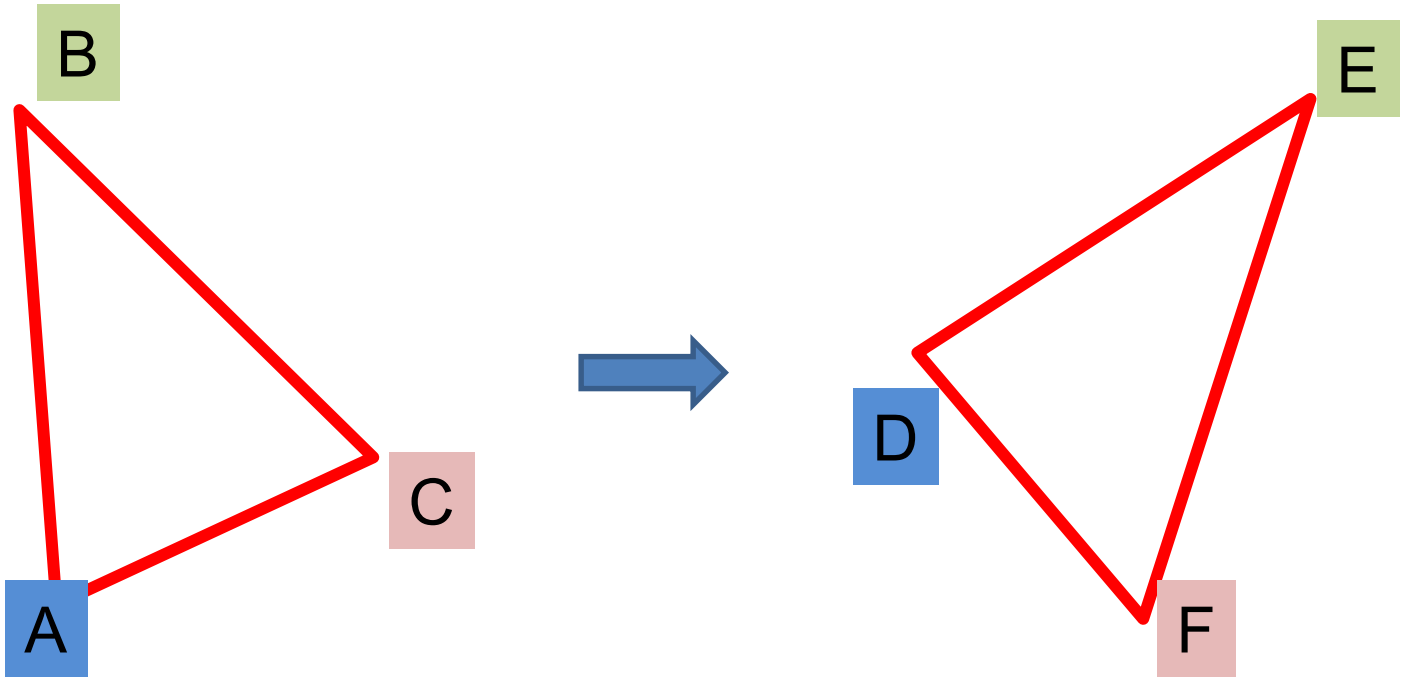
RANSAC Algorithm

- Repeat N times
  1. Randomly select a sample
  – Select just enough points to recover the parameters
  2. Fit the model with random sample
  3. See how many other points agree
- Best estimate is one with most agreement
  – can use agreeing points to refine estimate

Example of 2D line fitting

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)

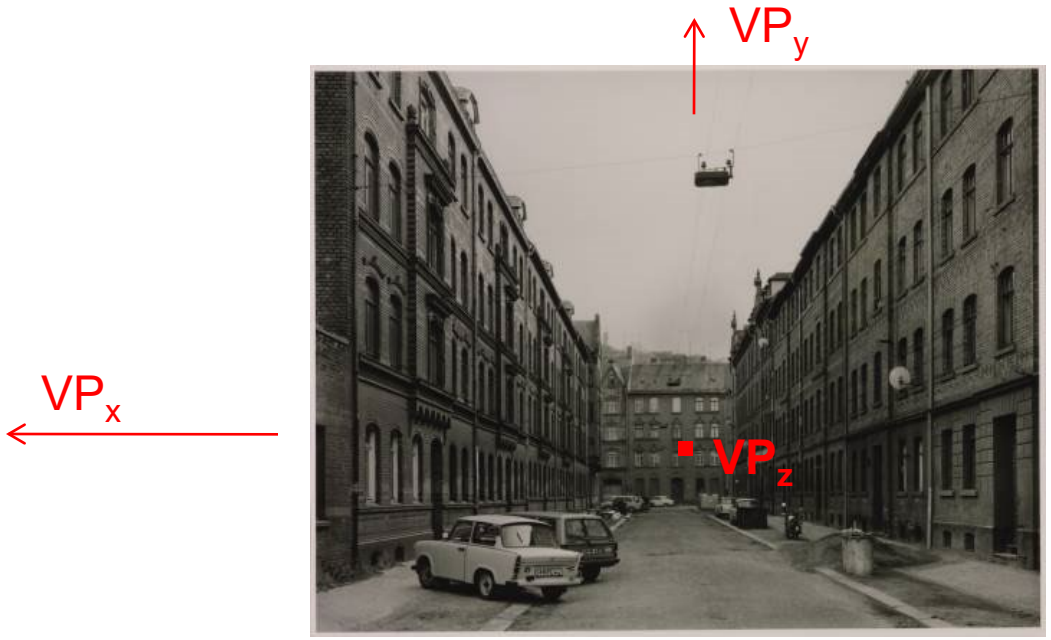


Photo from online Tate collection