Midterm Review

15-463: Computational Photography

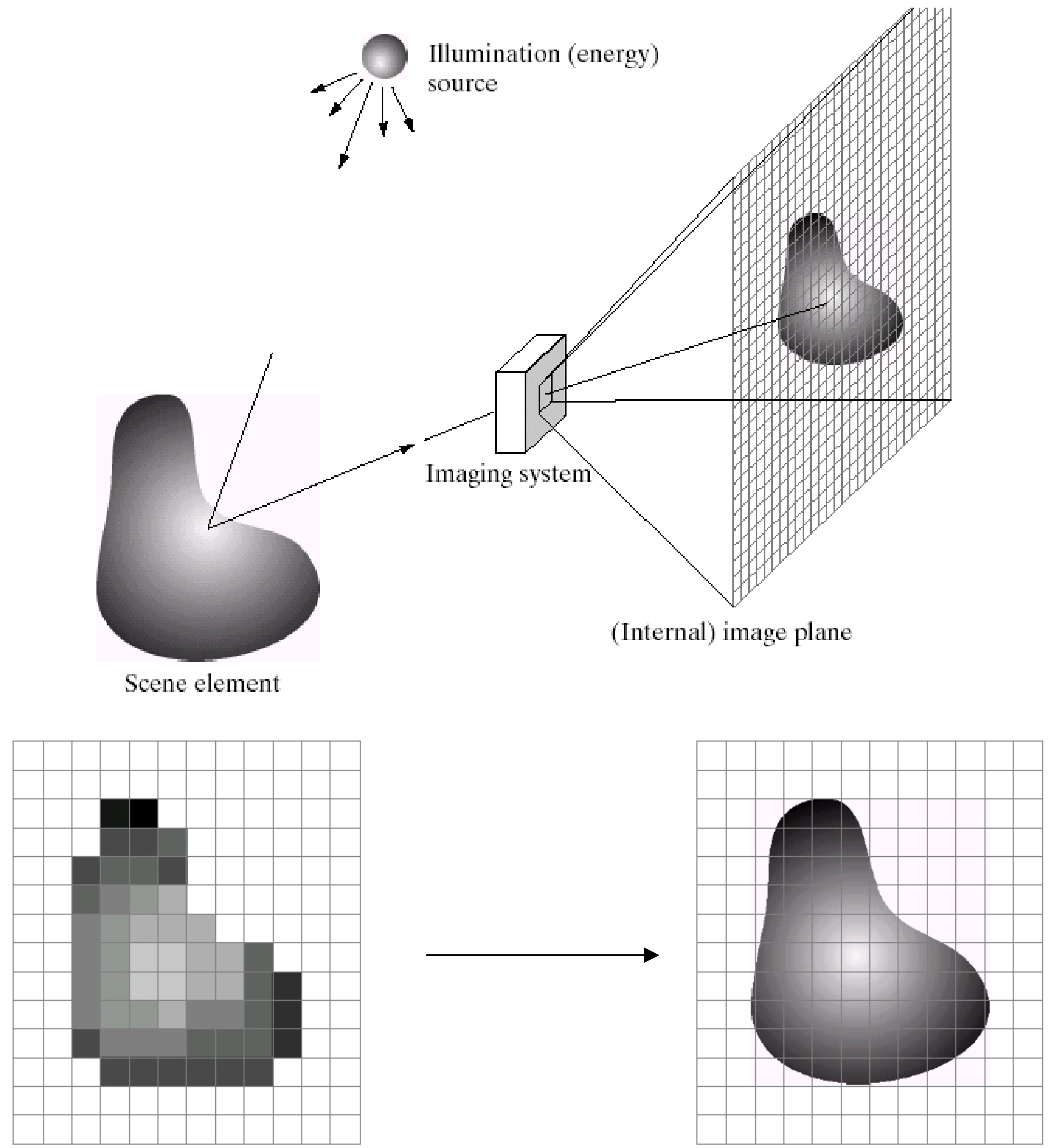
Review Topics

- Sampling and Reconstruction
- Frequency Domain and Filtering
- Blending
- Warping
- Data-driven Methods
- Camera
- Homographies
- Modeling Light

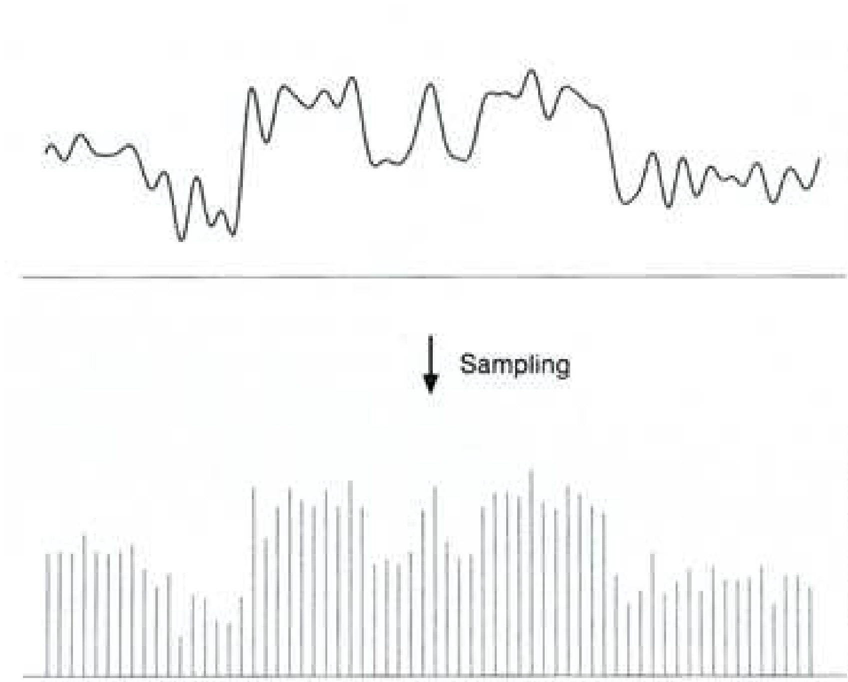
Review Topics

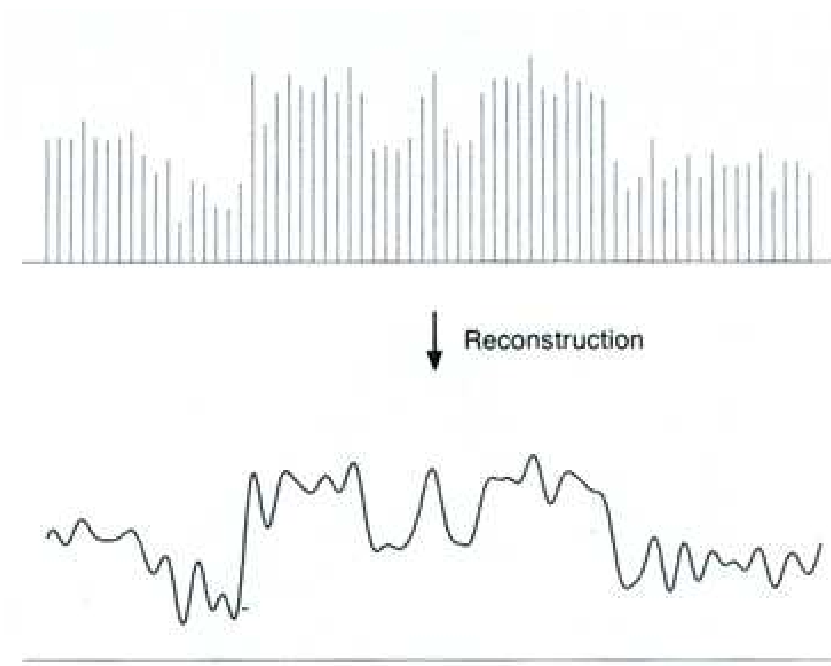
- Sampling and Reconstruction
- Frequency Domain and Filtering
- Blending
- Warping
- Data-driven Methods
- Camera
- Homographies
- Modeling Light

Sampling and Reconstruction



Sampling and Reconstruction

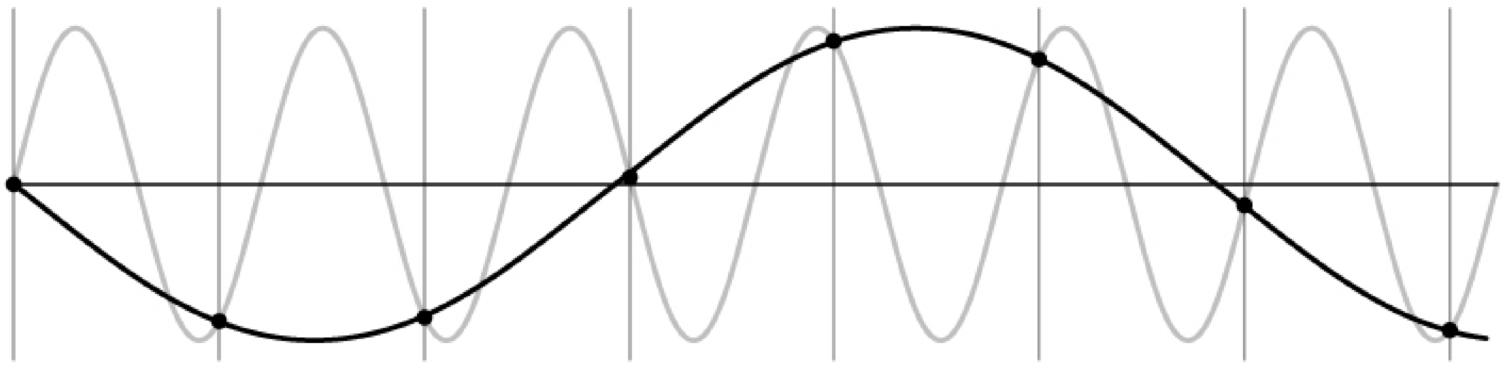


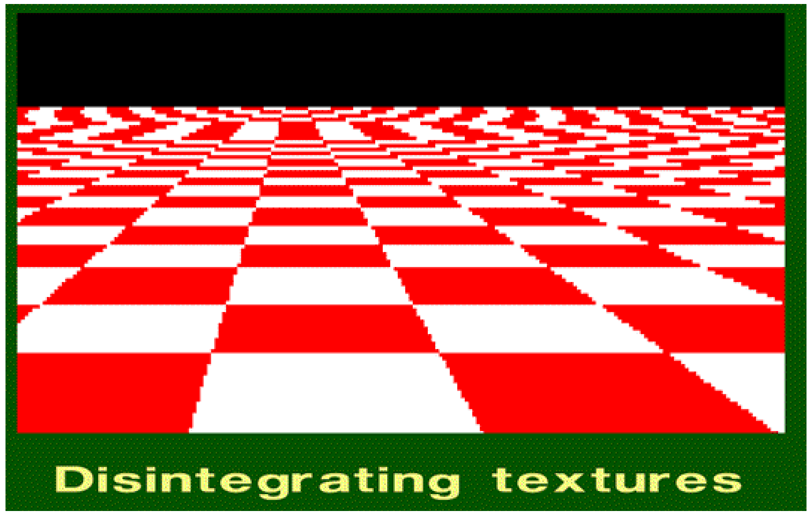


Mathematically guess what happens in between

Sampling and Reconstruction

- Effects of Undersampling
  - Lost information
  - High frequency signals get indistinguishable from low frequency ones (aliasing)





Sampling and Reconstruction

- How to avoid aliasing?
  - Sample more often
  - Low pass filter the signal (anti-aliasing)
- Filters work by convolution

Sampling and Reconstruction

- Examples of filters
  - Moving average
  - Weighted moving average
    - Equal weights
    - Gaussian weights
  - Sobel

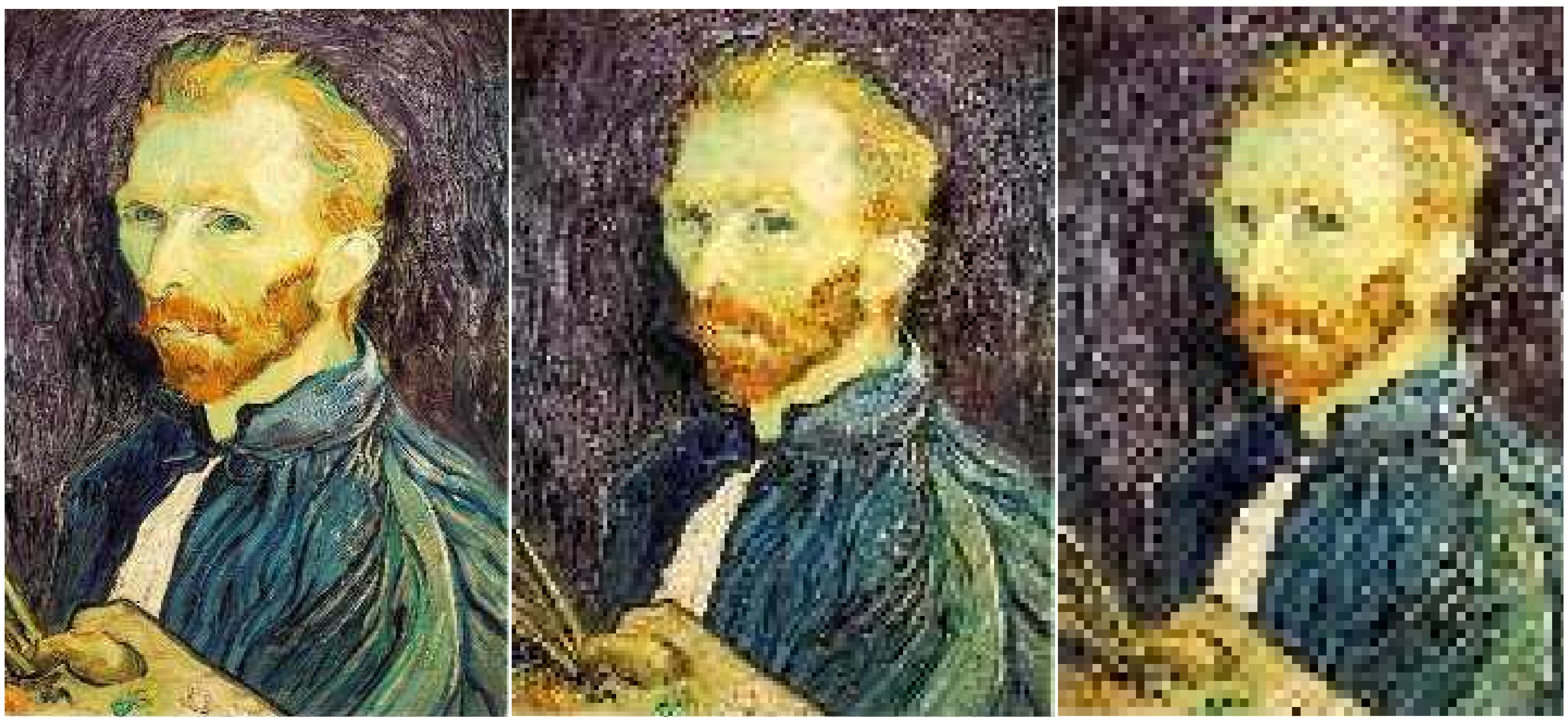
Sampling and Reconstruction

- Gaussian Filters
  - Smoothe out images
  - Convolution of two Gaussians each with standard deviation σ, gives Gaussian with standard deviation σ/2

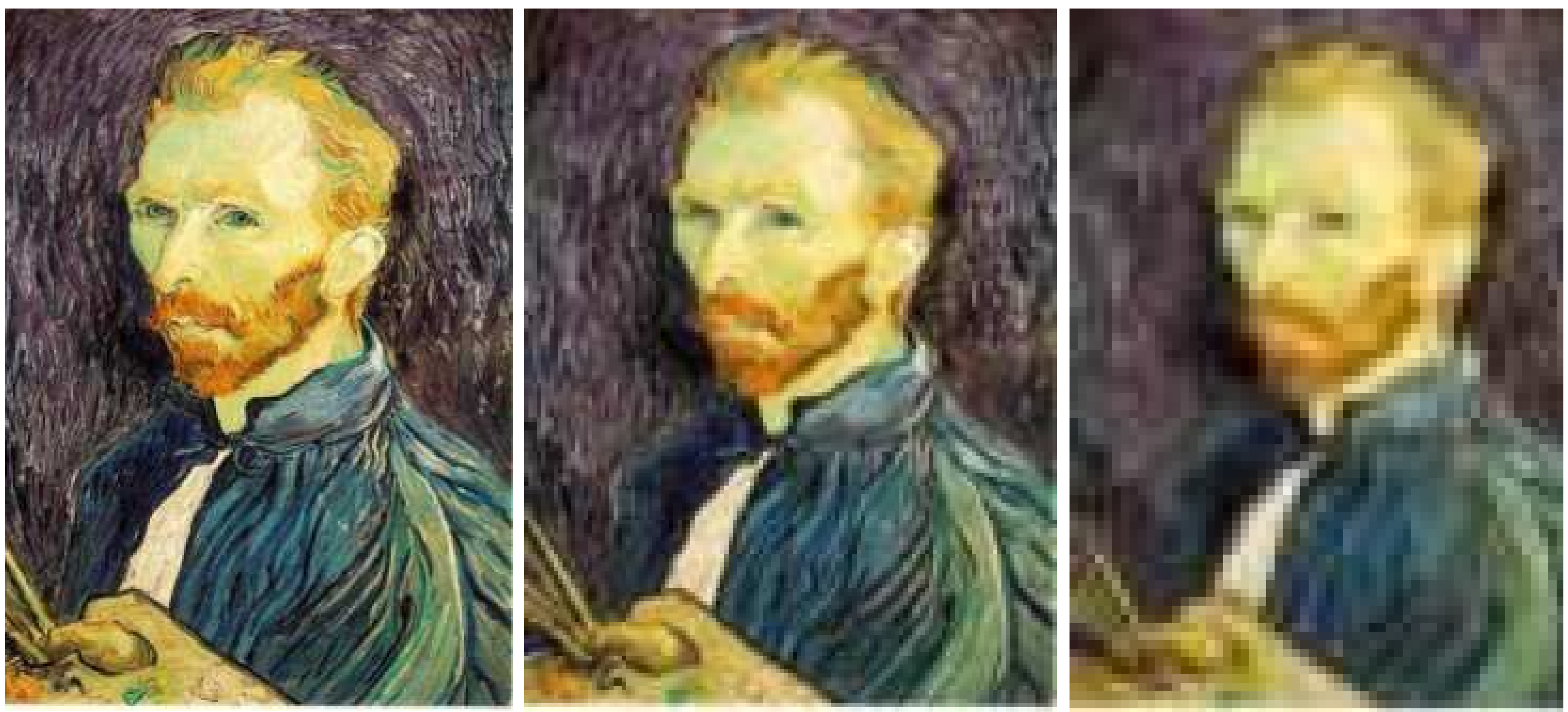
Sampling and Reconstruction

- Matching
  - Use normalized-cross correlation or SSH over patches
- Subsampling
  - Filter with Gaussian then subsample
  - Double filter size with every half-sizing
    - Forms image pyramids

Sampling and Reconstruction



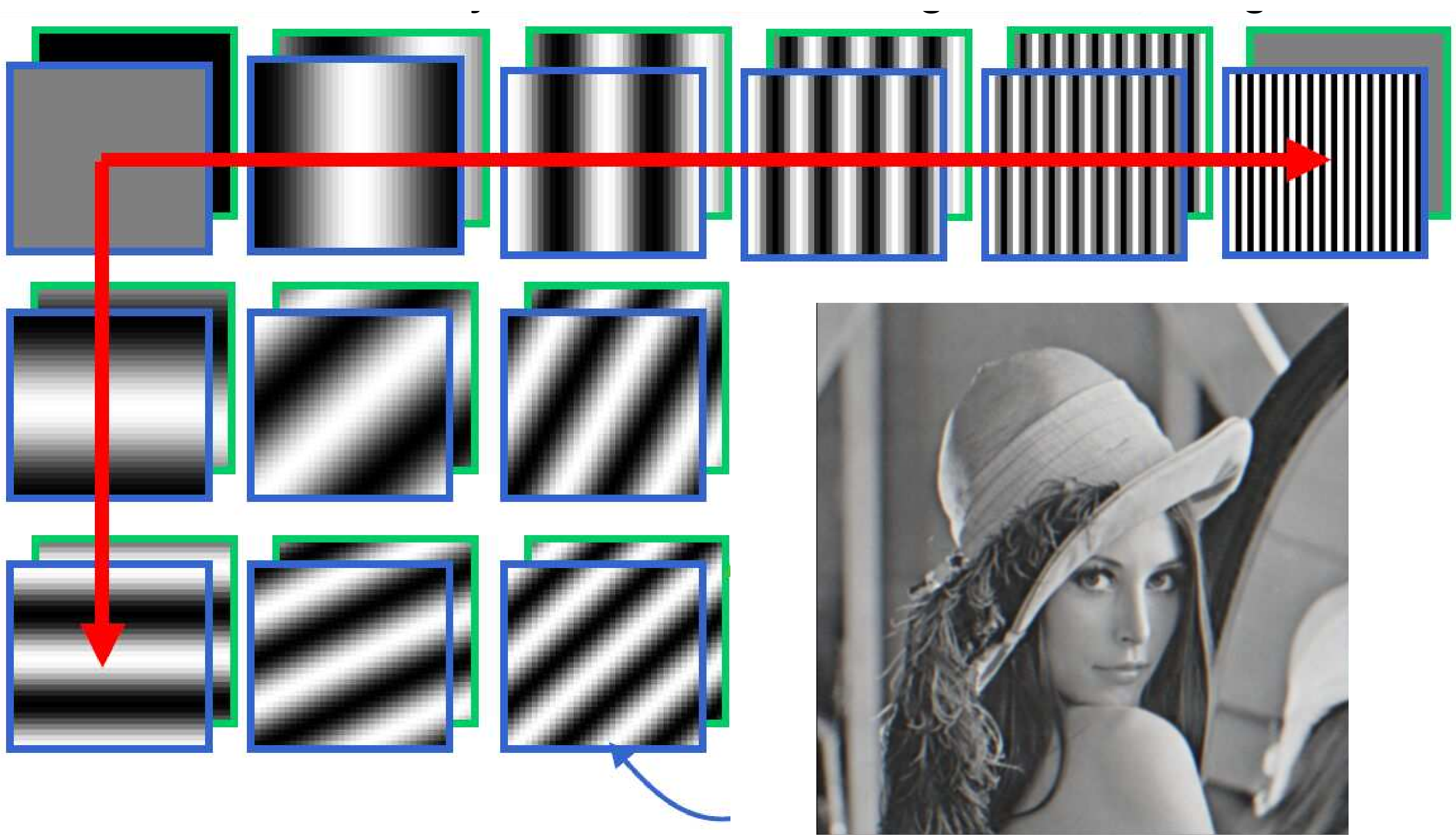
Sampling and Reconstruction



Review Topics

- Sampling and Reconstruction
- Frequency Domain and Filtering
- Blending
- Warping
- Data-driven Methods
- Camera
- Homographies
- Modeling Light

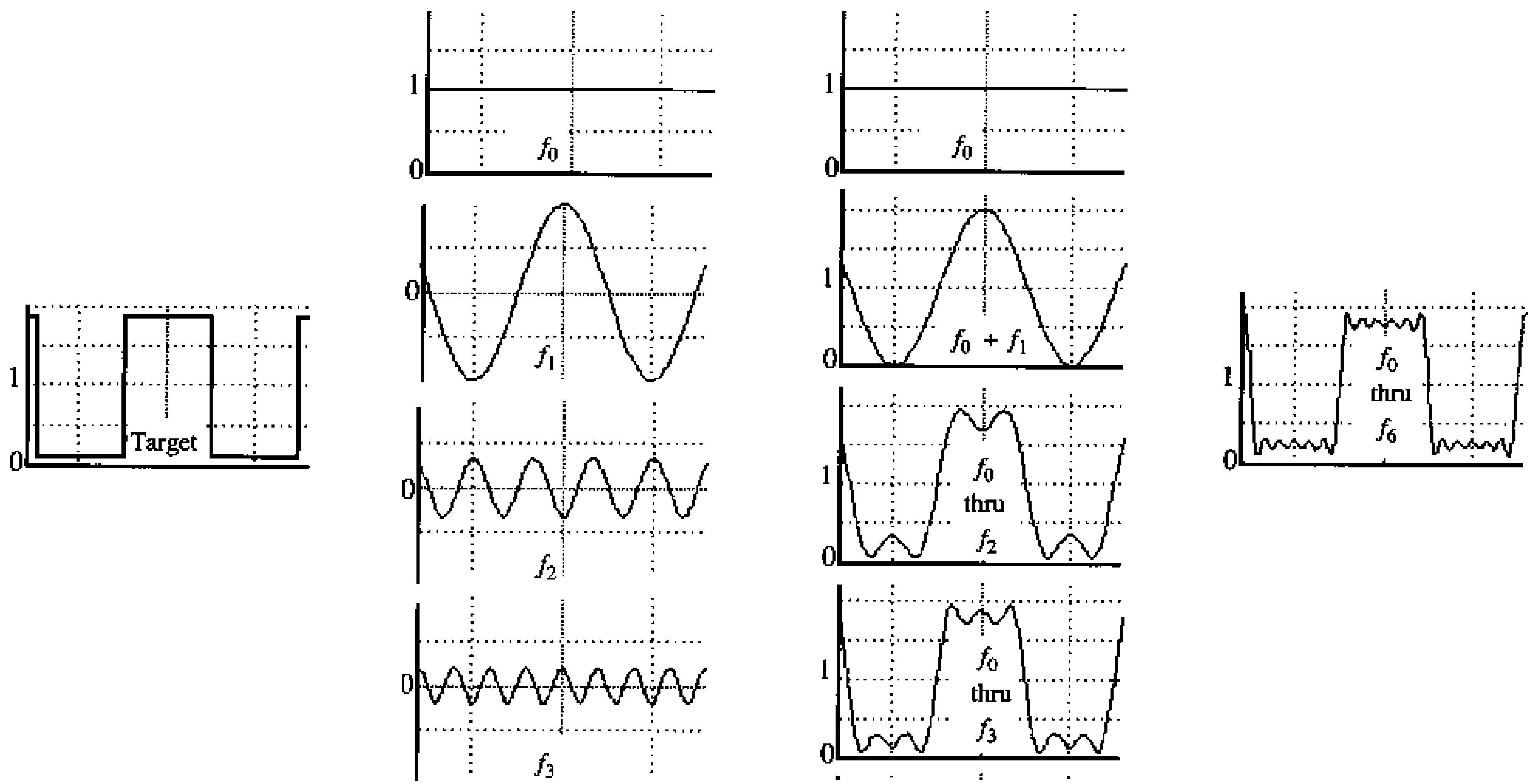
Frequency Domain



Decompose signal into different frequencies

Frequency Domain and Filtering

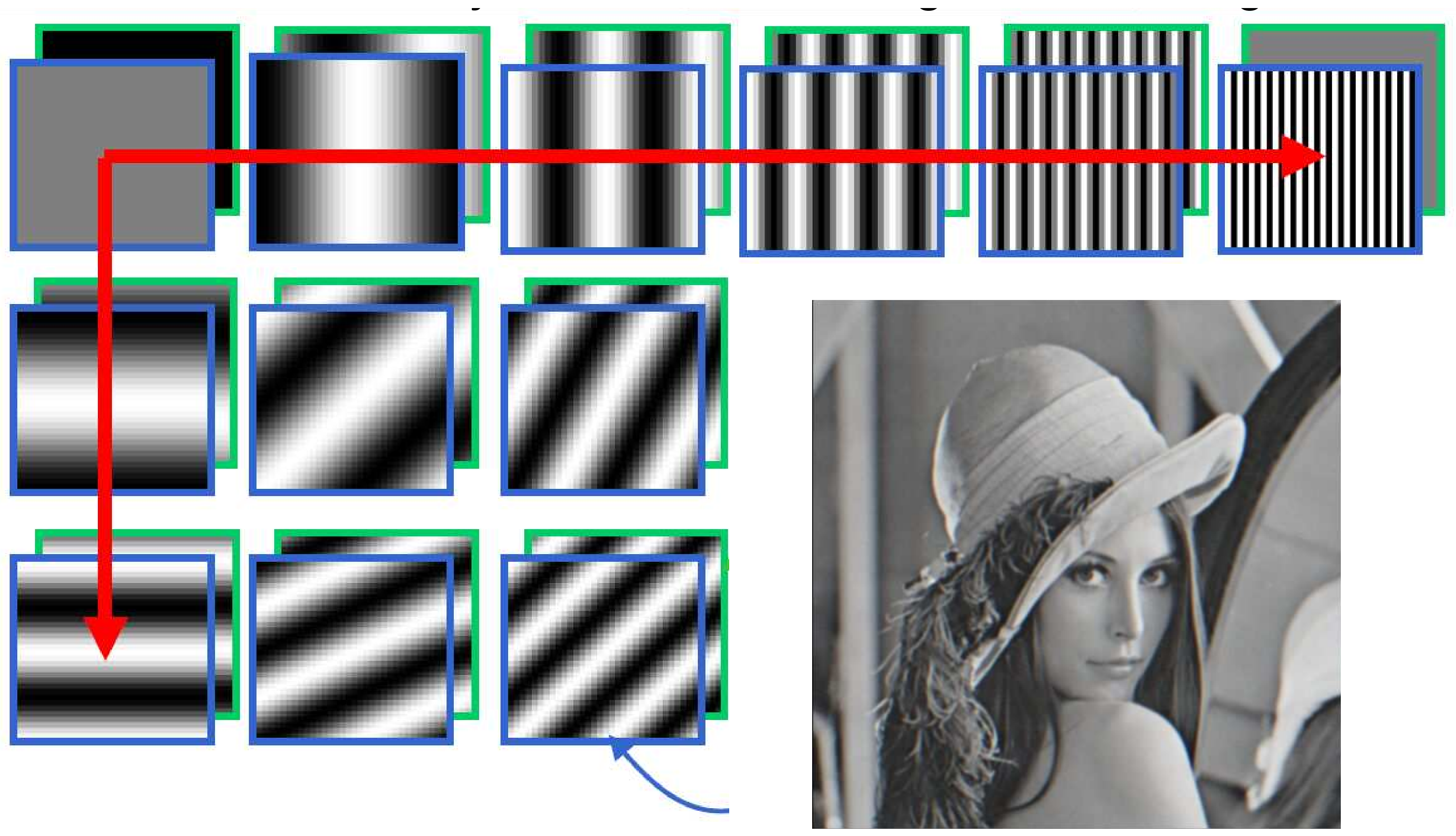
Sum of sine waves of different frequencies



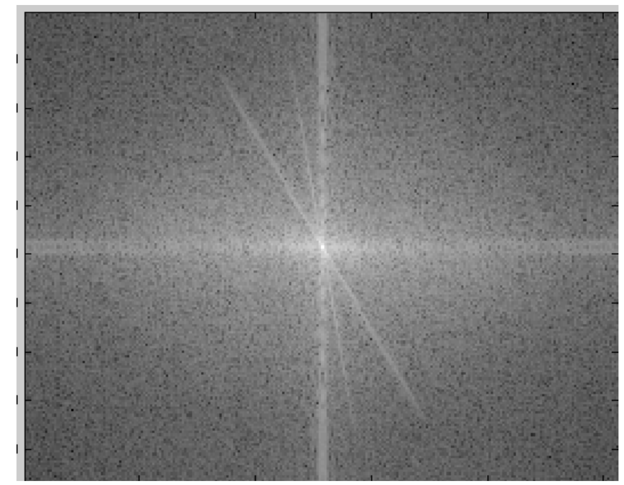
$f(x) \longrightarrow$ Fourier Transform $\longrightarrow F(\omega)$

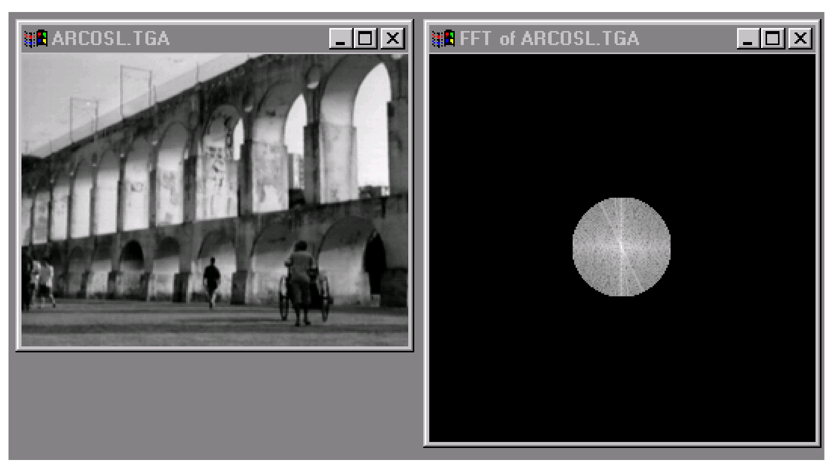
Frequency Domain and Filtering

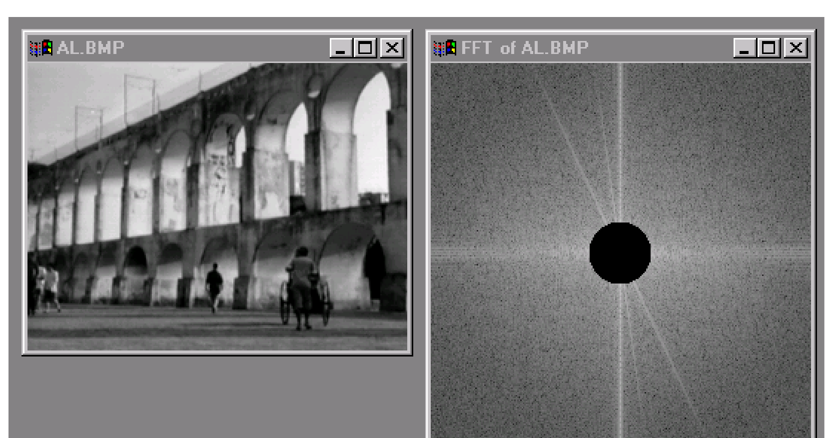
In 2D



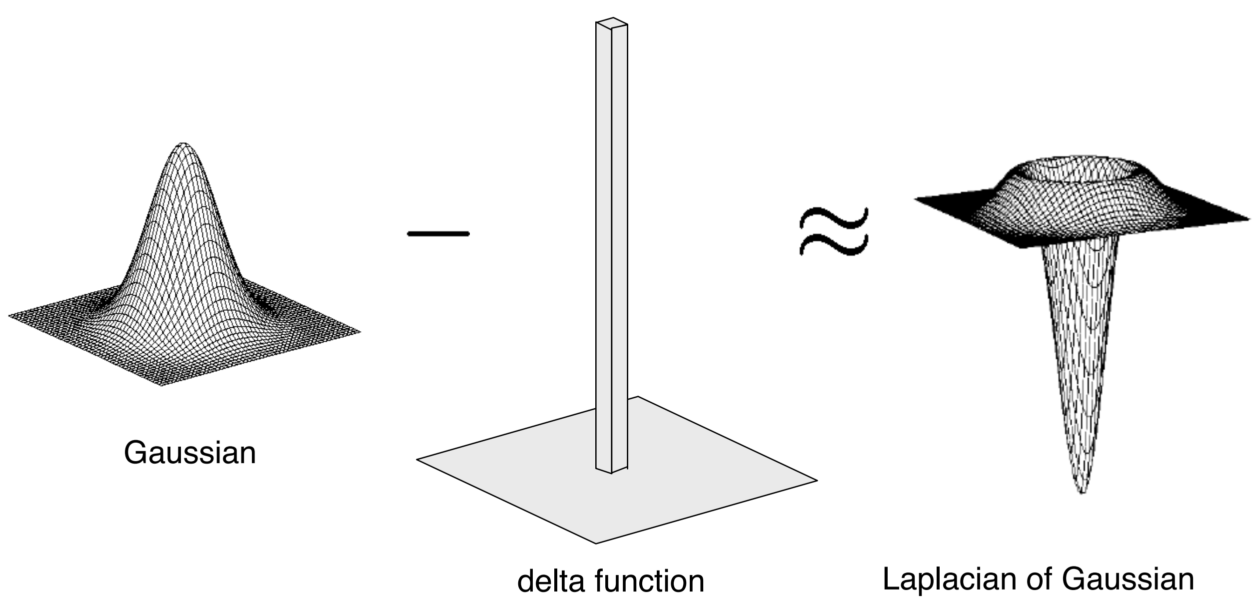
Frequency Domain and Filtering



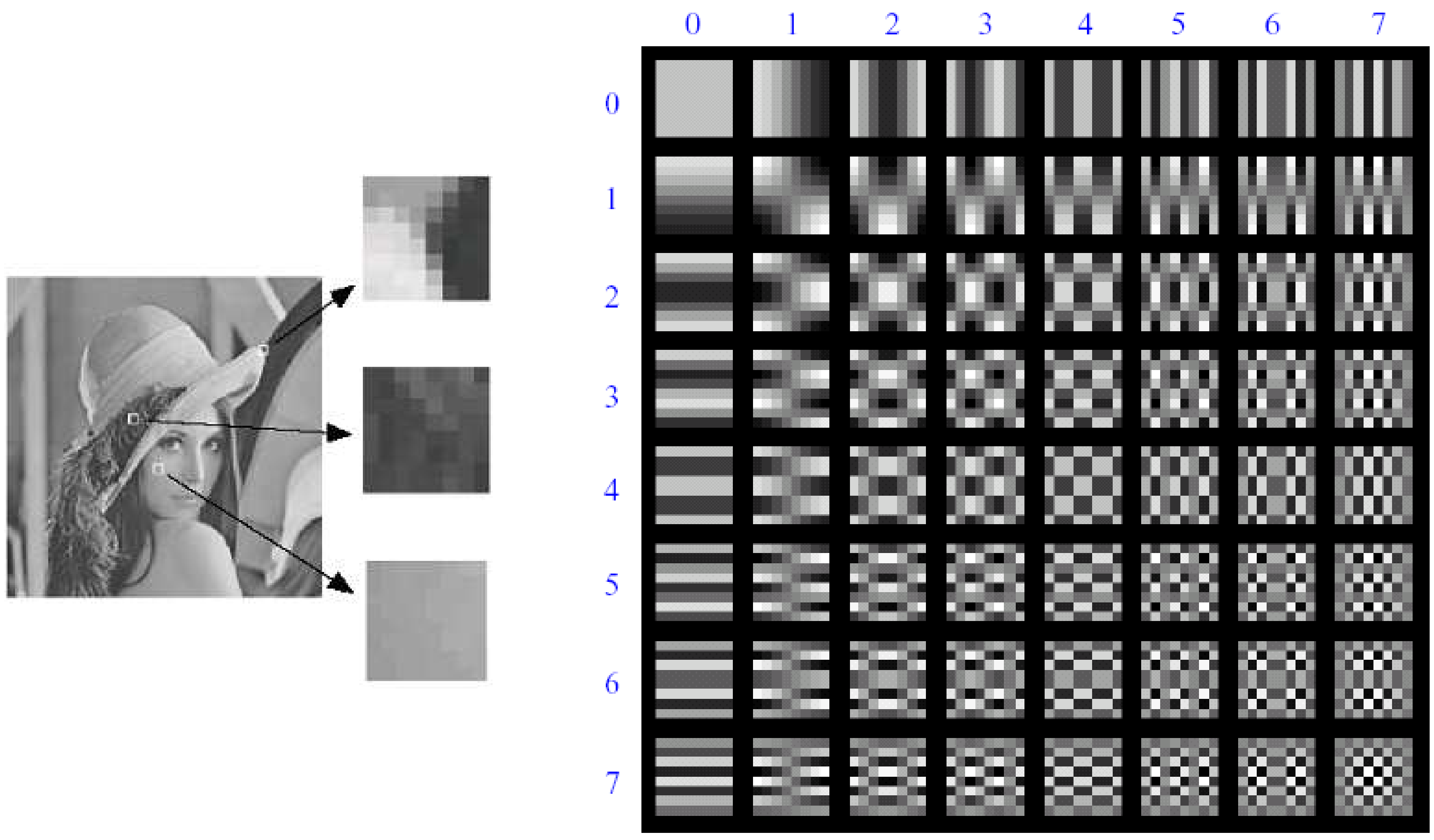




Frequency Domain and Filtering



Frequency Domain and Filtering



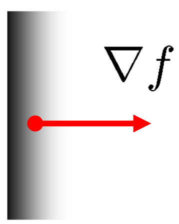
Block-based Discrete Cosine Transform (DCT)

Frequency Domain and Filtering

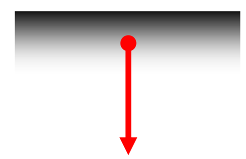
The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

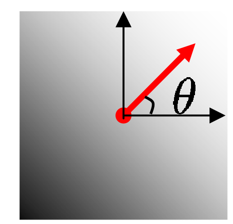
The gradient points in the direction of most rapid change in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

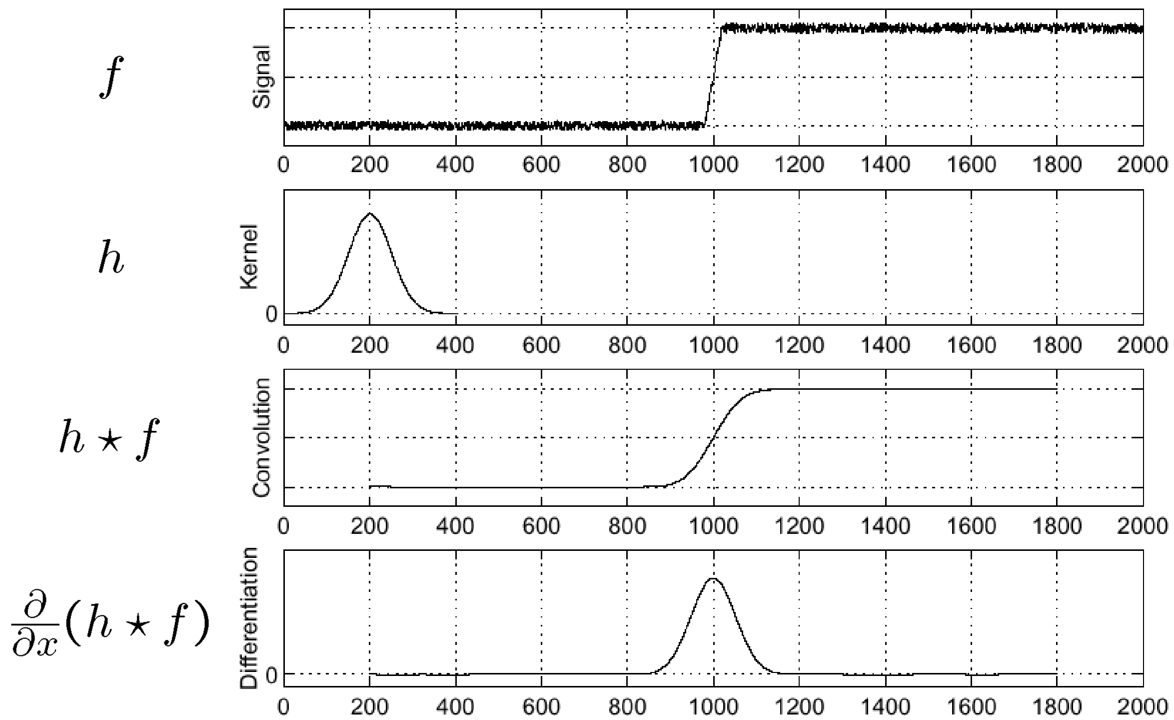


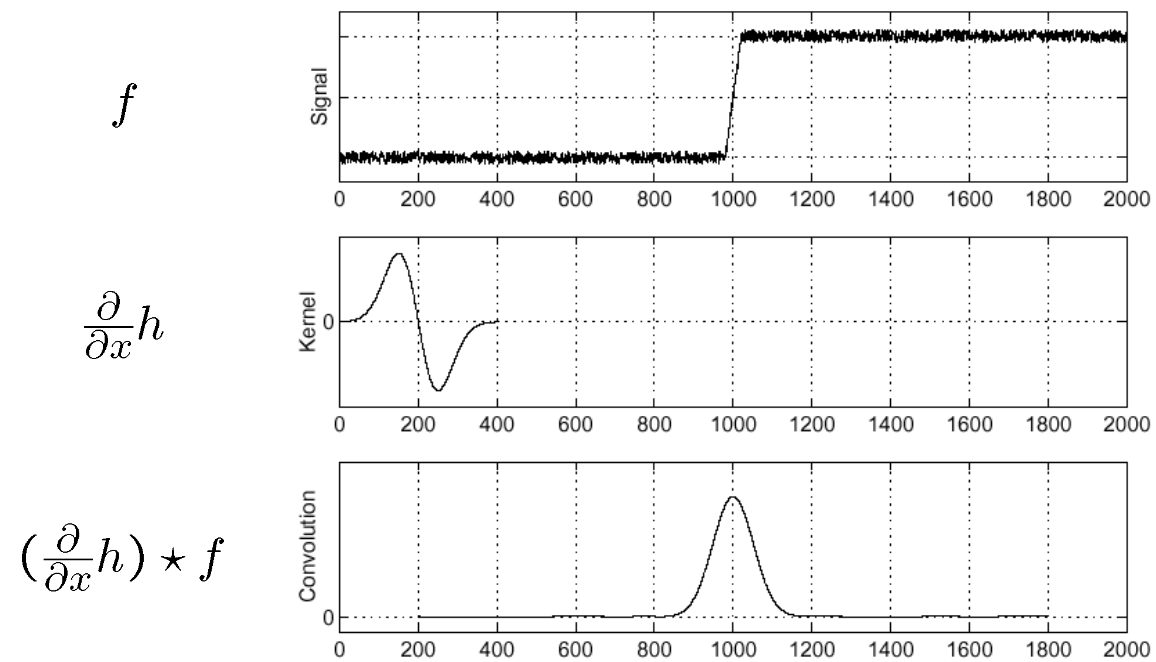
$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Frequency Domain and Filtering

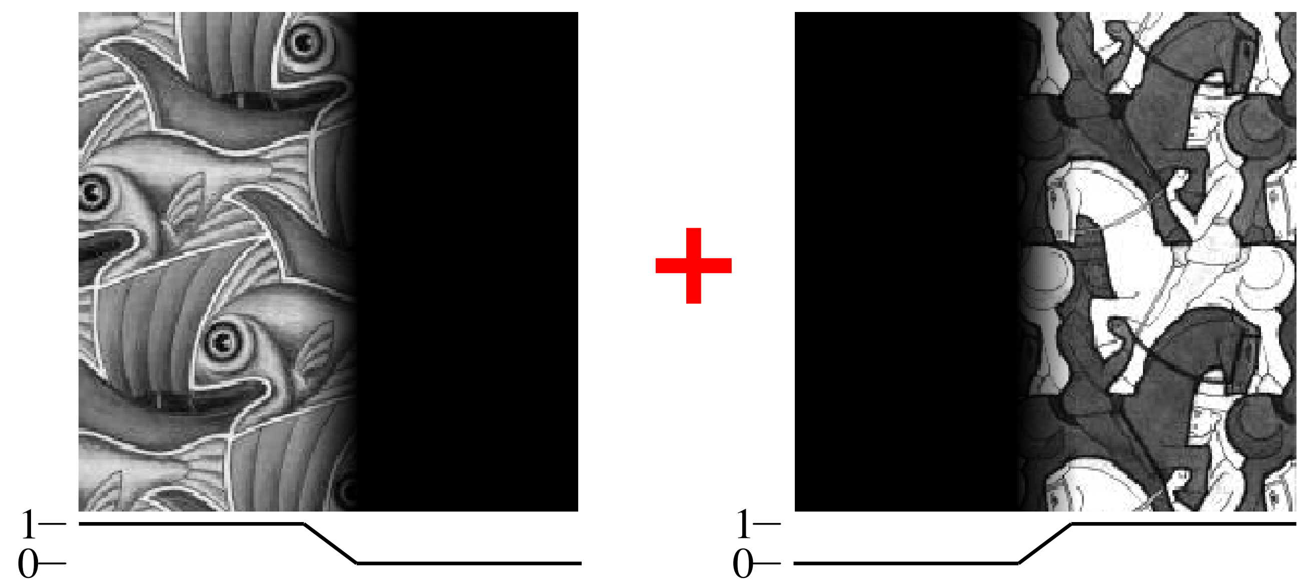




Review Topics

- Sampling and Reconstruction
- Frequency Domain and Filtering
- Blending
- Warping
- Data-driven Methods
- Camera
- Homographies
- Modeling Light

Blending



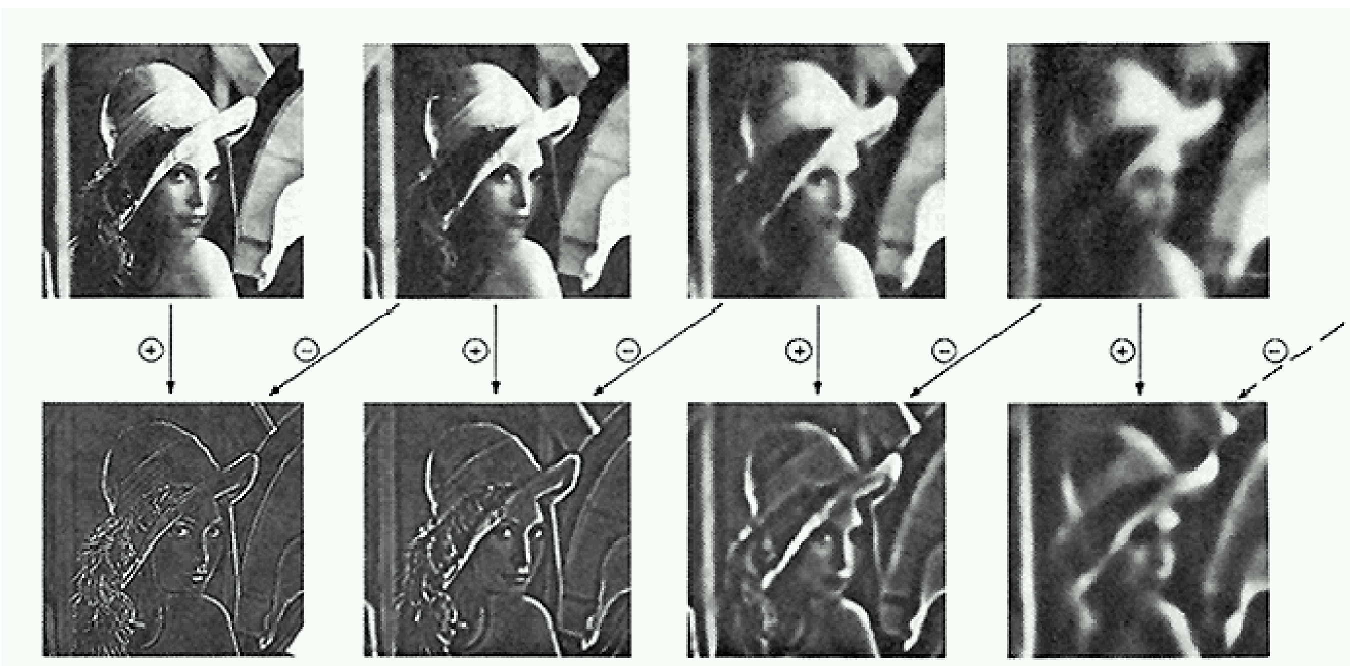
$$I_{blend} = \alpha I_{left} + (1-\alpha) I_{right}$$

- Window size = size of largest feature (to avoid strong seams)
- Window size <= 2 * size of smallest feature (to avoid ghosting)

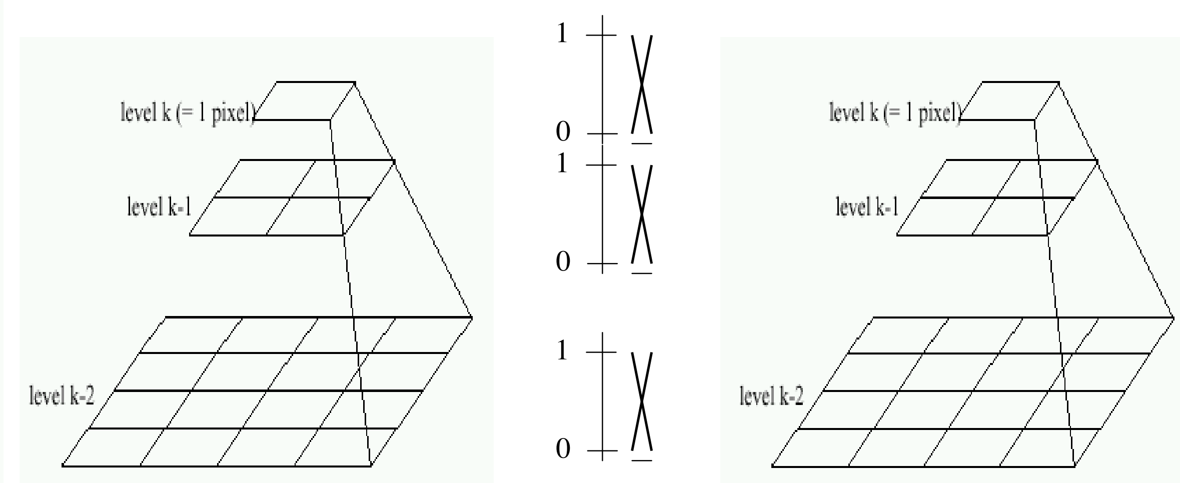
Blending

Pyramid Blending

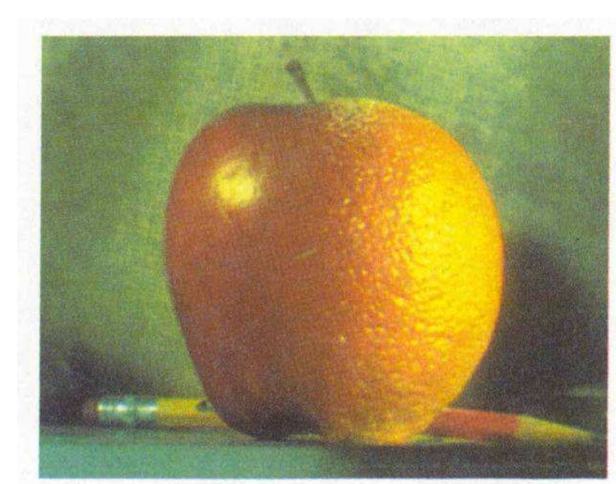
Lowpass Images



Bandpass Images



Left pyramid blend Right pyramid



Blending

Gradient Domain

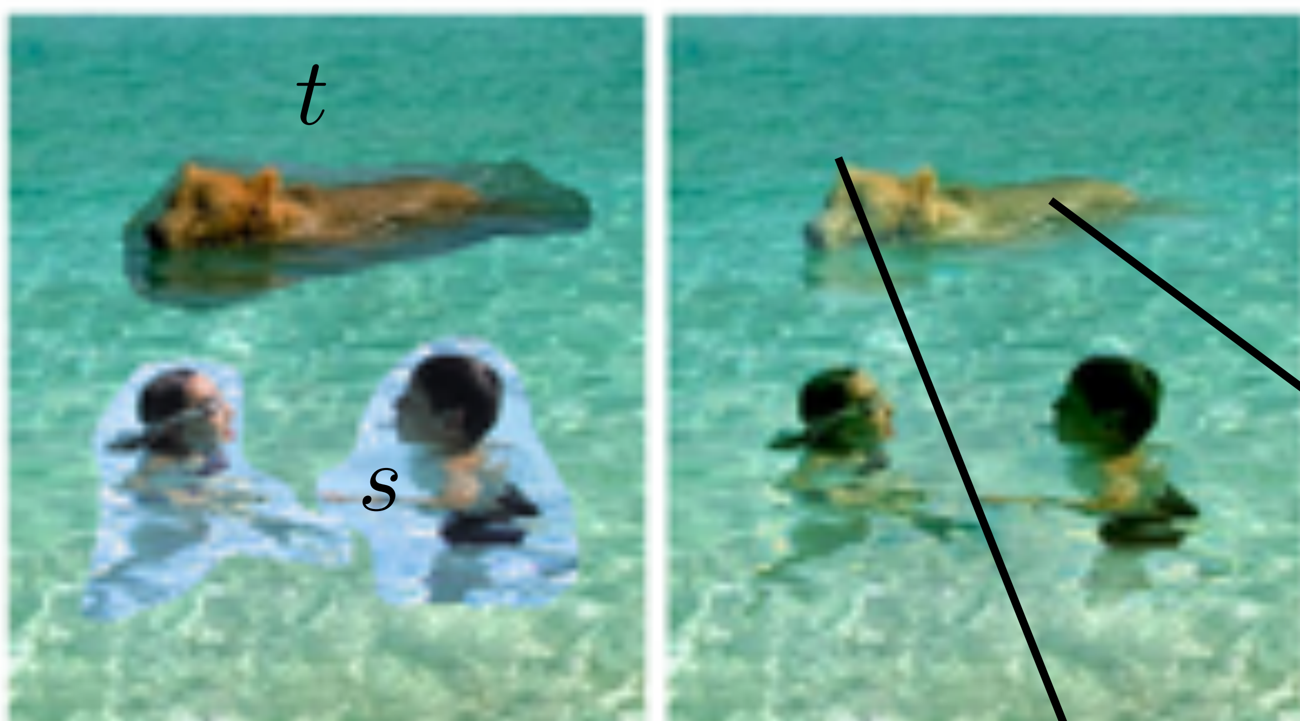
- Result image: f
  Gradients: f_x, f_y
- Want f to 'look like' some prespecified d, and
  f_x, f_y to 'look like' some prespecified g^x, g^y

$$\min_f \; w^x(f_x - g^x)^2 + w^y(f_y - g^y)^2 + w^d(f - d)^2$$

- Weights specify **per-pixel** importance of how much you want f close to d, f_x close to g^x, f_y close to g^y

Blending

Gradient Domain



$$f_x = s_x, f_y = s_y$$

$$f(x, y) - t(x-1, y) = s_x, f(x, y) - t(x, y-1) = s_y$$

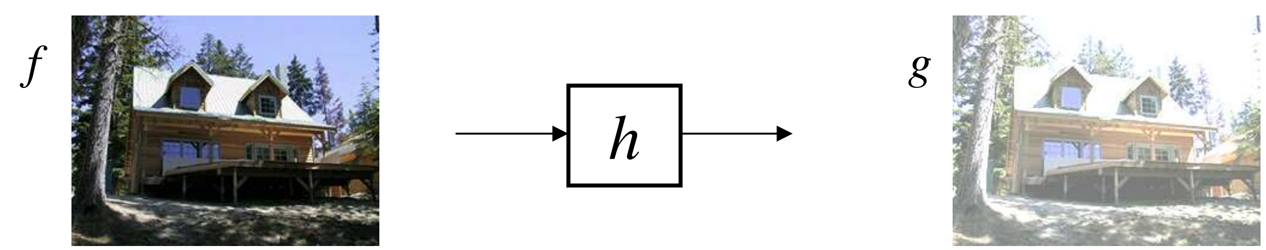
Review Topics

- Sampling and Reconstruction
- Frequency Domain and Filtering
- Blending
- Warping
- Data-driven Methods
- Camera
- Homographies
- Modeling Light

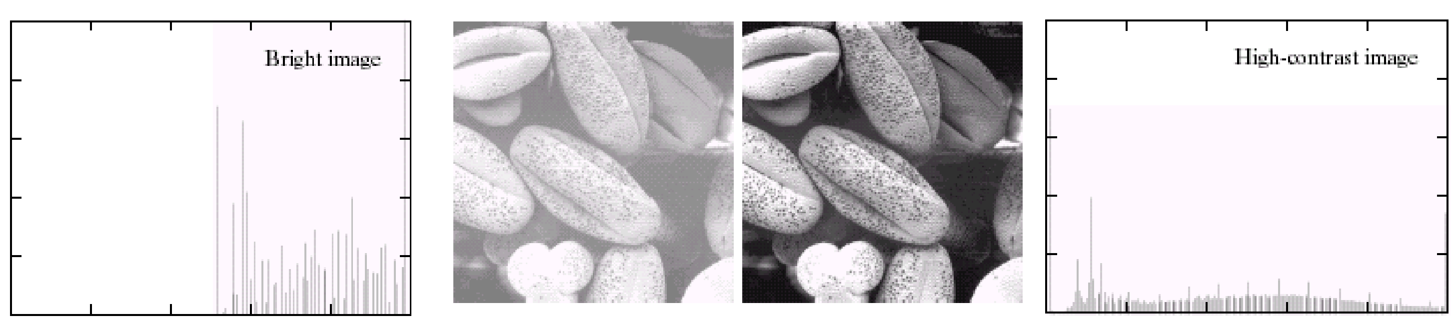
Point Processing

Change range of image

$$g(x) = h(f(x))$$



Example: $g(x)=f(x)+.3$



Histogram Equalization

Warping

Change domain of image

f

$g(x) = f(h(x))$

$$\longrightarrow \boxed{h} \longrightarrow$$

g

Example: $g(x)=f(x/2)$

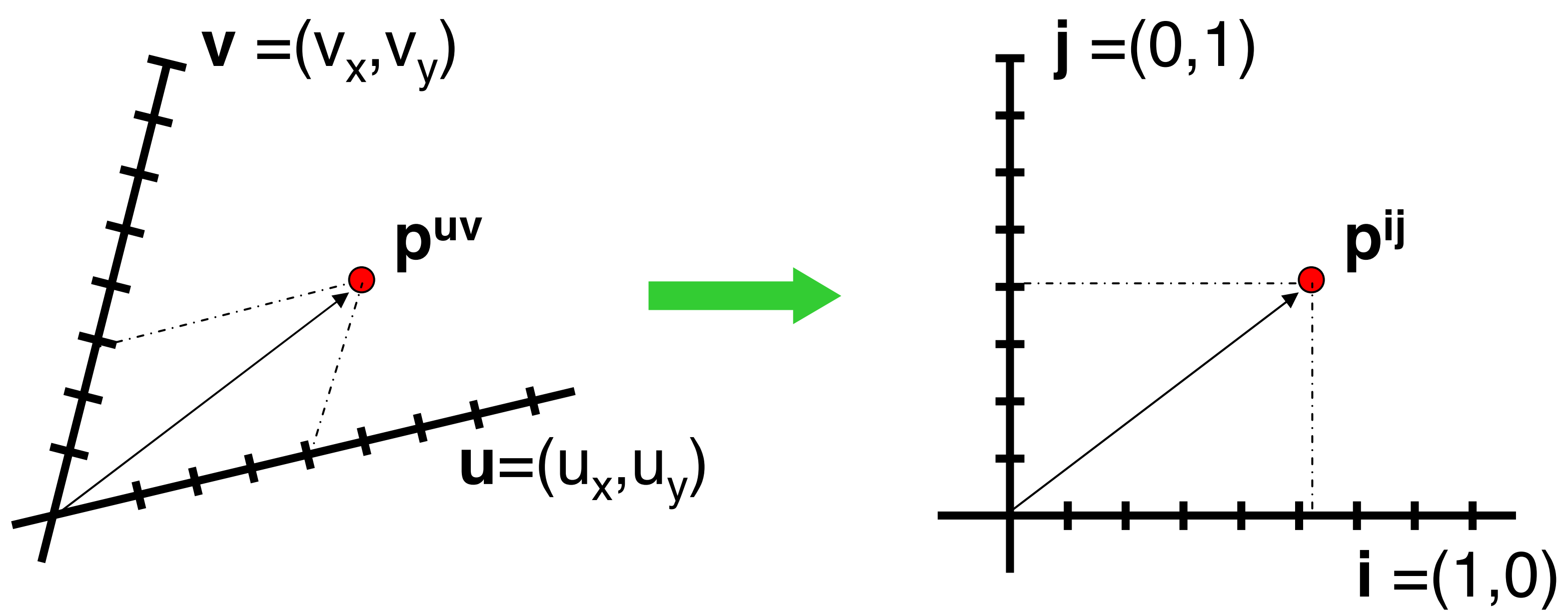
Warping

- 2D Transformations
  - Translate
  - Rotate
  - Scale
  - Similarity
  - Affine
  - Projective

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

Warping

Change of Basis



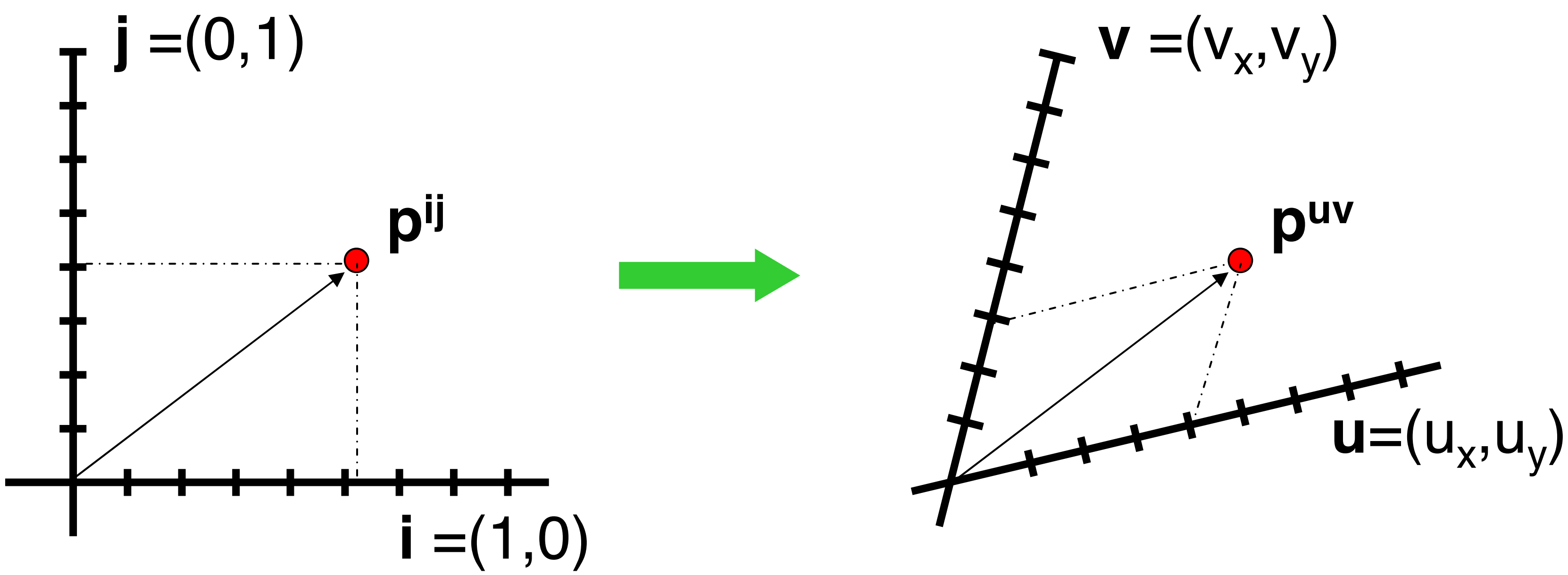
$\mathbf{p}^{\mathbf{uv}} = (4,3)$

$\mathbf{p}^{\mathbf{ij}} = 4\mathbf{u} + 3\mathbf{v}$

$$\mathbf{p}^{\mathbf{ij}} = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} \mathbf{p}^{\mathbf{uv}}$$

Warping

Change of Basis: Inverse Transform



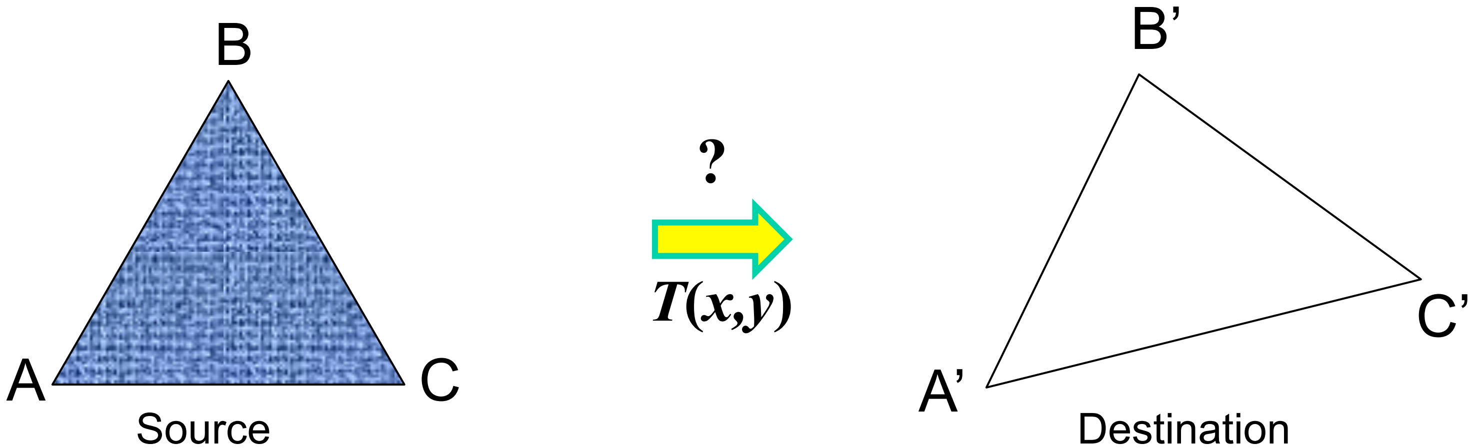
$\mathbf{p}^{\mathbf{ij}} = (5,4) = p_x\mathbf{u} + p_y\mathbf{v}$

$\mathbf{p}^{\mathbf{uv}} = (p_x,p_y) = ?$

$$\mathbf{p}^{\mathbf{uv}} = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix}^{-1} \begin{bmatrix} 5 \\ 4 \end{bmatrix} = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix}^{-1} \mathbf{p}^{\mathbf{ij}}$$

Warping

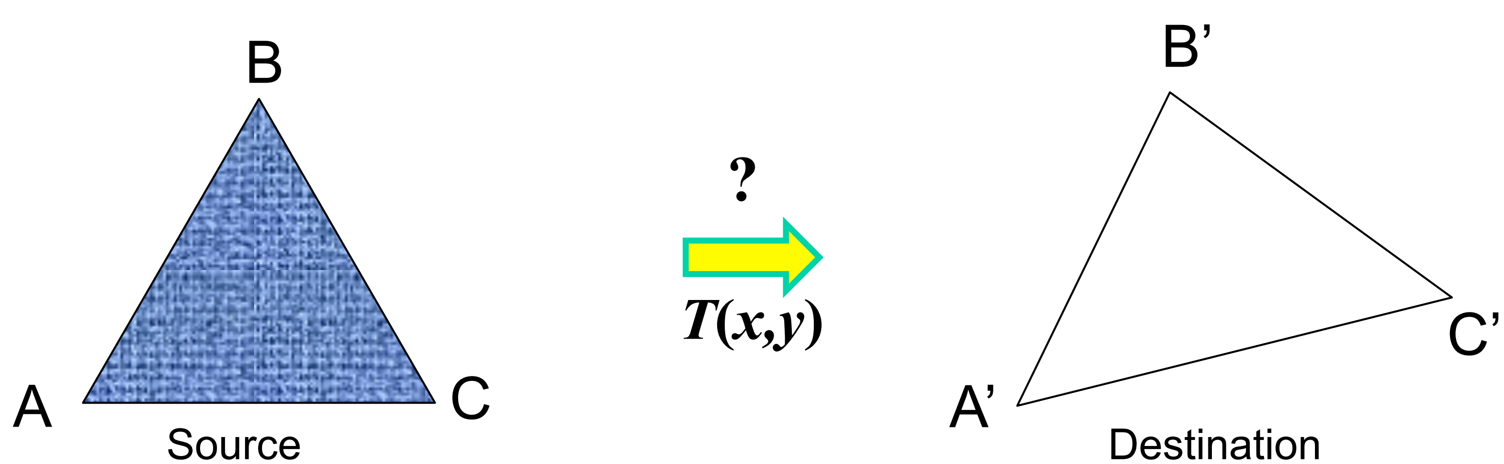
- Affine Warp
  - Need 3 correspondences



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Warping

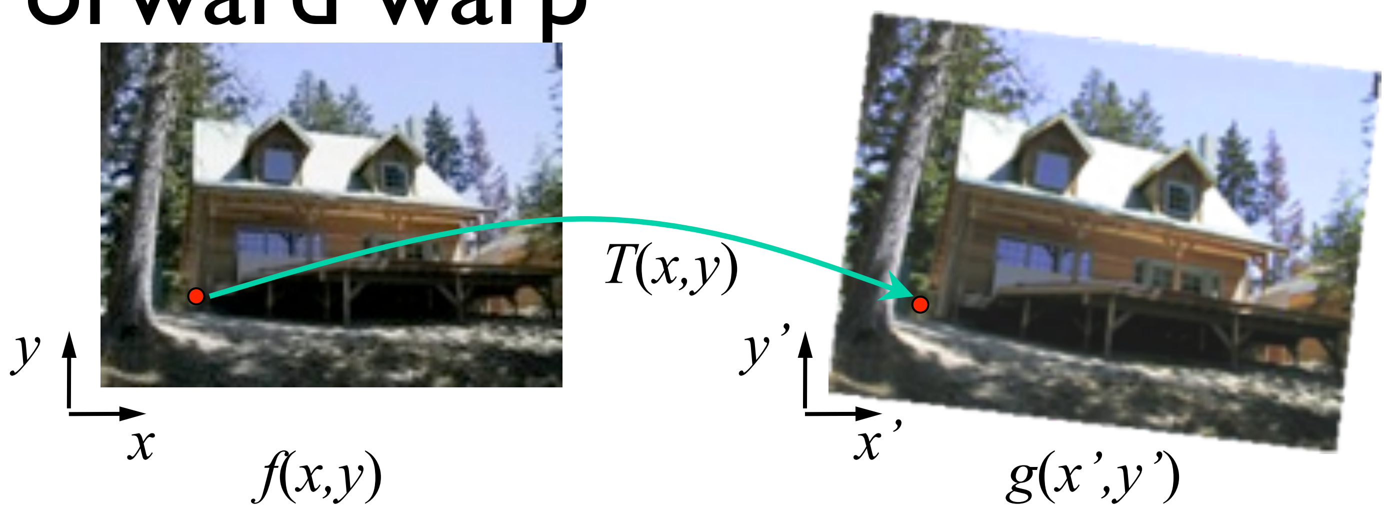
- Many ways to find affine matrix
  - Warp Source to [0,0], [1,0], [0,1], and then to Destination
  - Pose as system of equations in [a;b;c;d;e;f]



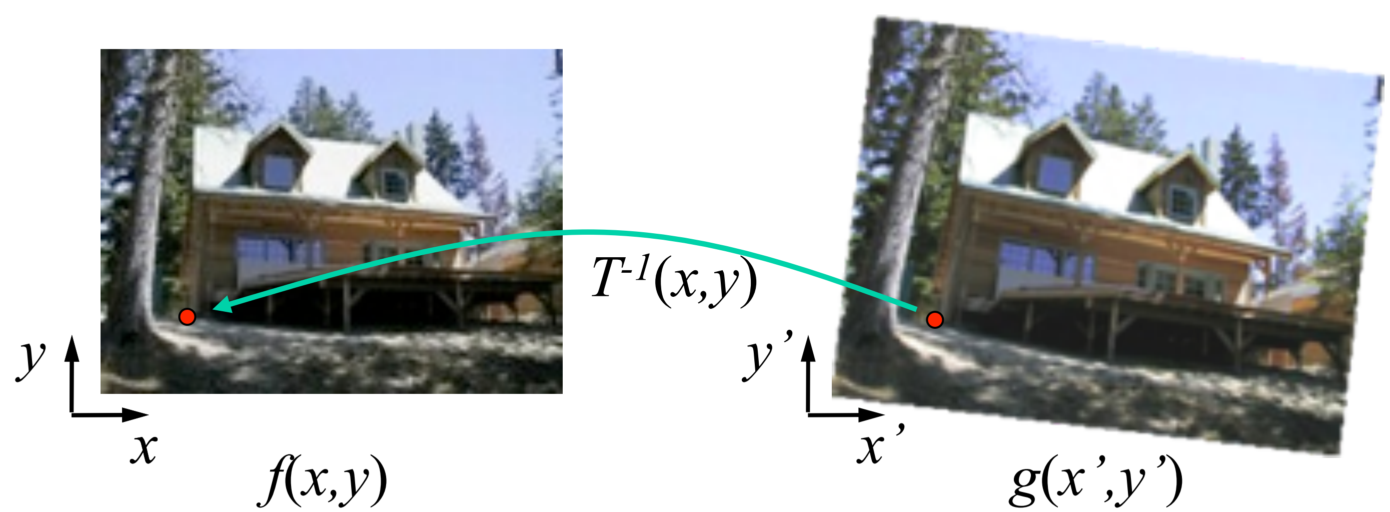
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Warping

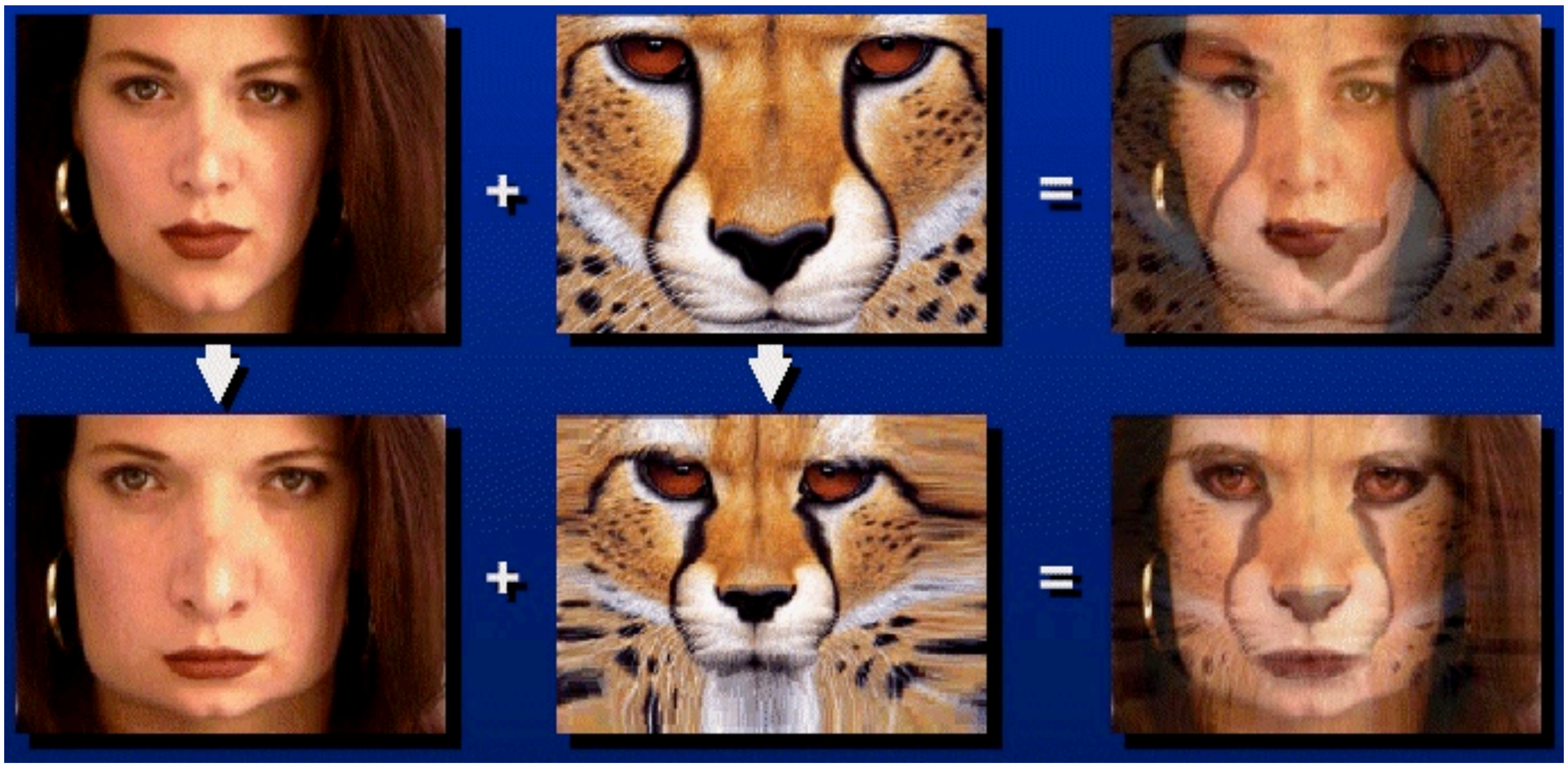
- Forward warp



- Inverse warp



Morphing

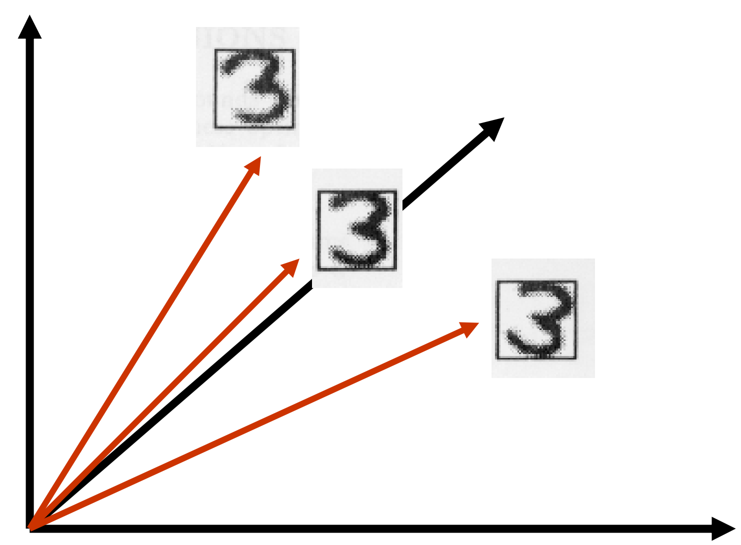


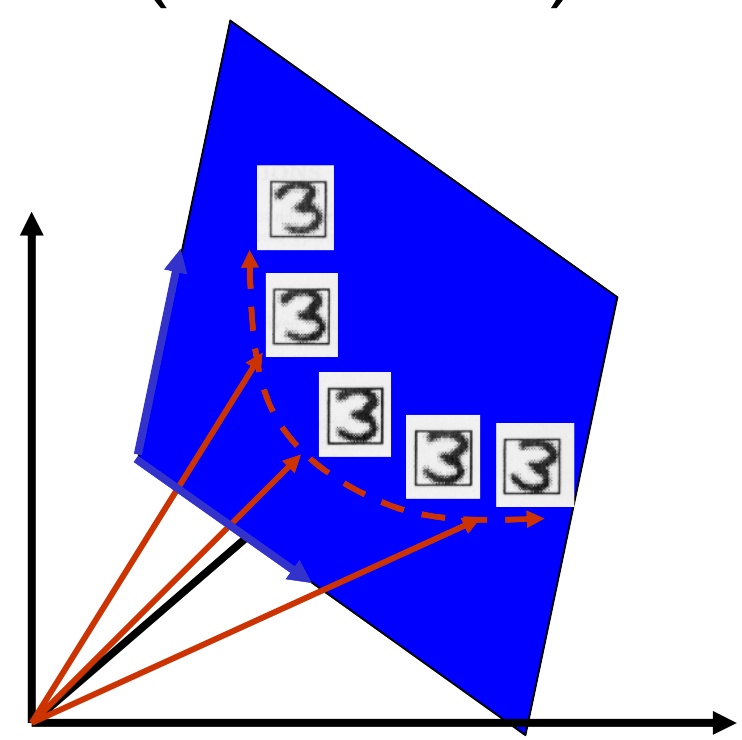
Review Topics

- Sampling and Reconstruction
- Frequency Domain and Filtering
- Blending
- Warping
- Data-driven Methods
- Camera
- Homographies
- Modeling Light

Data-Driven Methods

Subspaces Methods (ex: Faces)





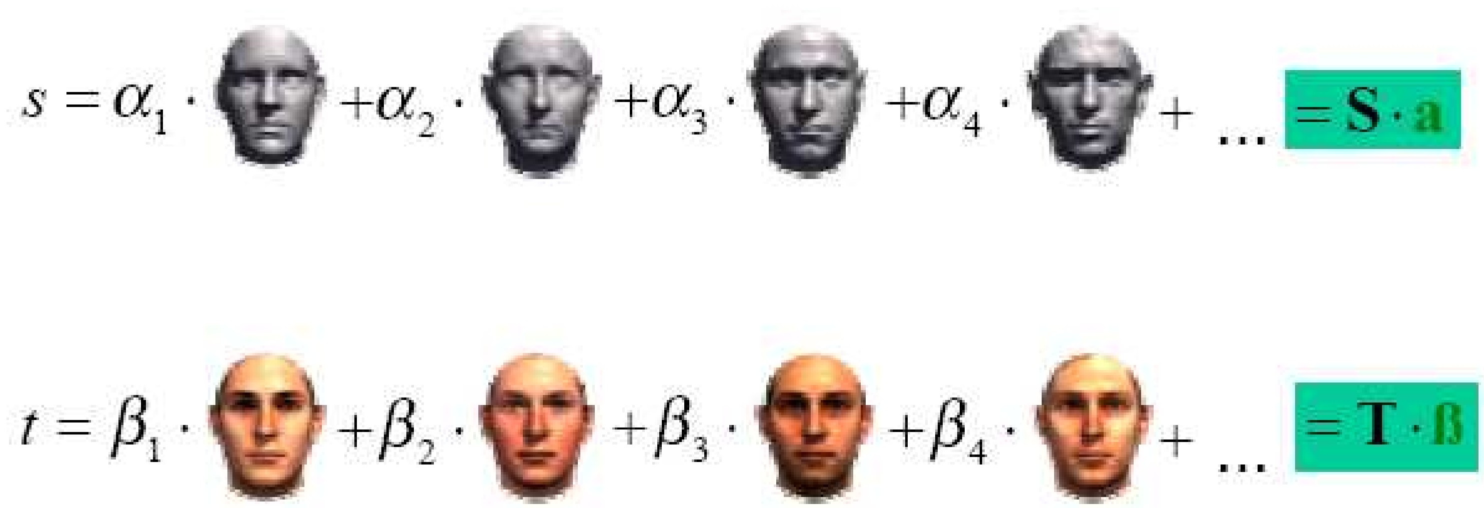
Write an image as linear combination of basis images

$$X = \sum_{i=1}^{m} a_i X_i$$

Data-Driven Methods

Subspaces Methods (ex: Faces)

$$\mathbf{S}_{model} = \sum_{i=1}^{m} a_i \mathbf{S}_i \qquad \mathbf{T}_{model} = \sum_{i=1}^{m} b_i \mathbf{T}_i$$

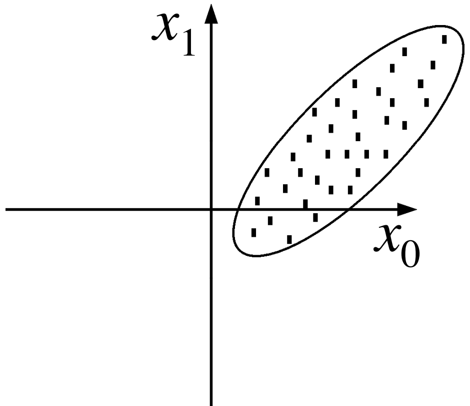


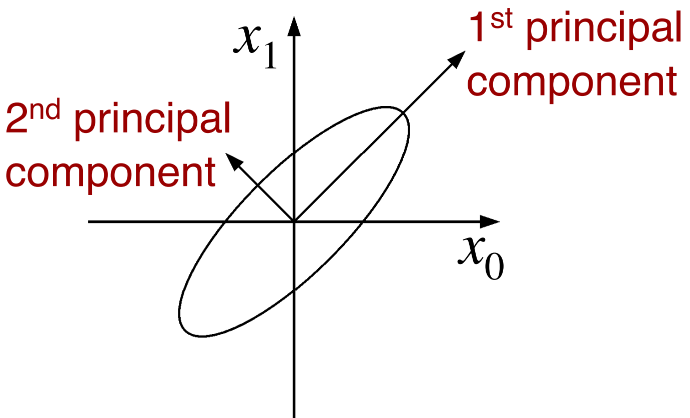
Shape and Appearance Models

Data-Driven Methods

Subspaces Methods (ex: Faces)

- How to get basis?
  - How many basis images to use?
  - How to get images that capture important variations?
- Use PCA (principal component analysis)
  - Keep those principal components whose eigenvalues are above a threshold





Data-Driven Methods

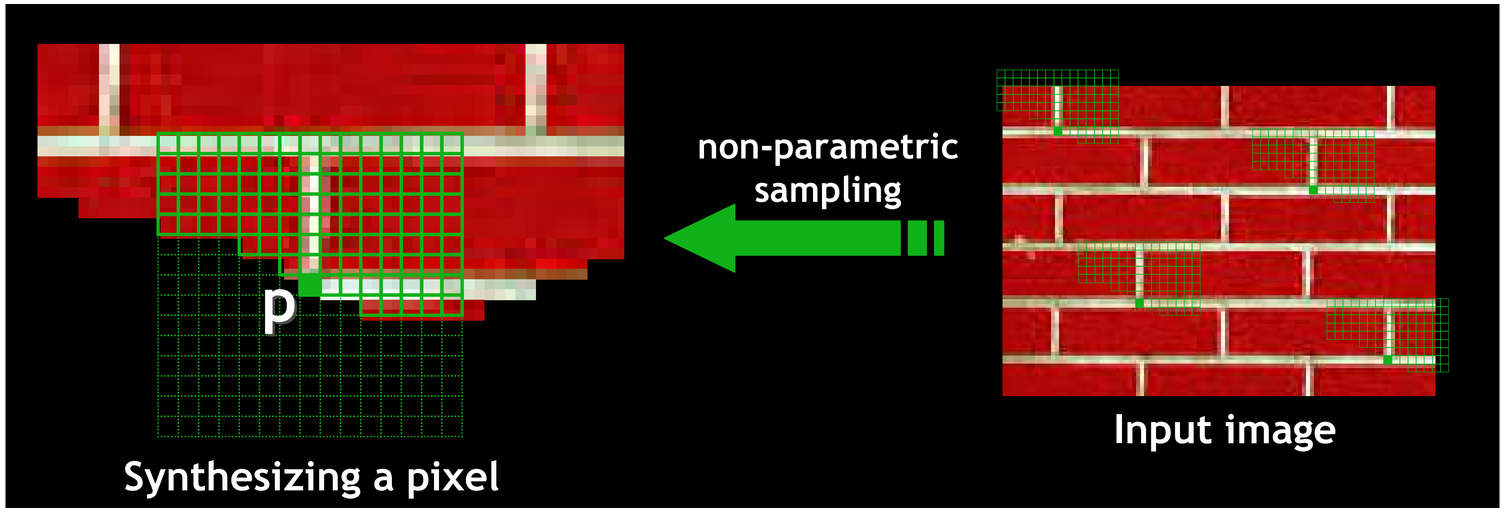
Video Textures

- Compute SSD between frames
- At frame i, transit either to
  - frame i+1
  - frame j (if SSD(j, i+1) is small)
- Decide to go from i to j or i+1 by tossing a weighted coin.

$$P_{i \to j} \sim \exp\left(- C_{i \to j} / \sigma^2 \right)$$

Data-Driven Methods

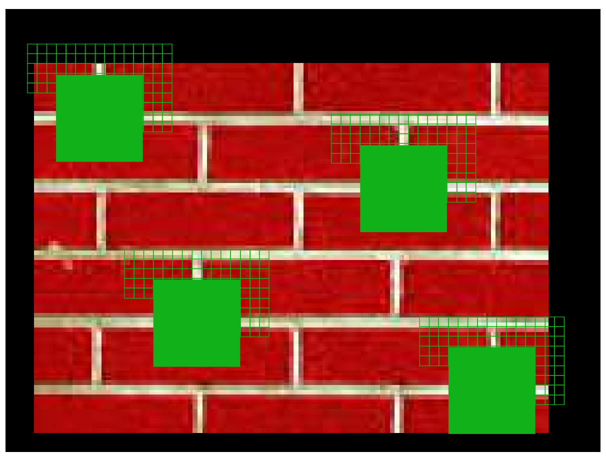
Texture Synthesis

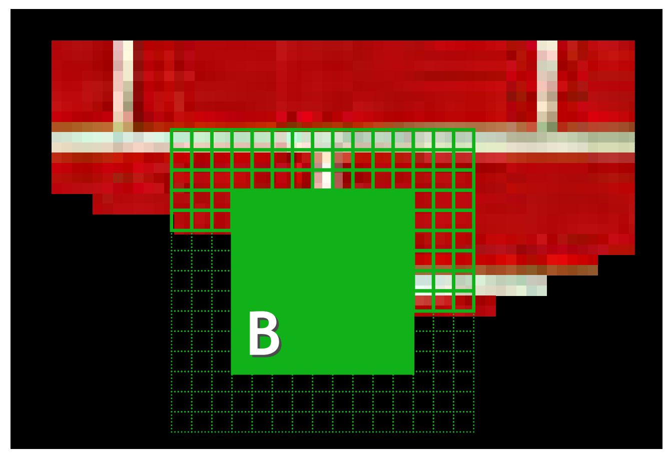


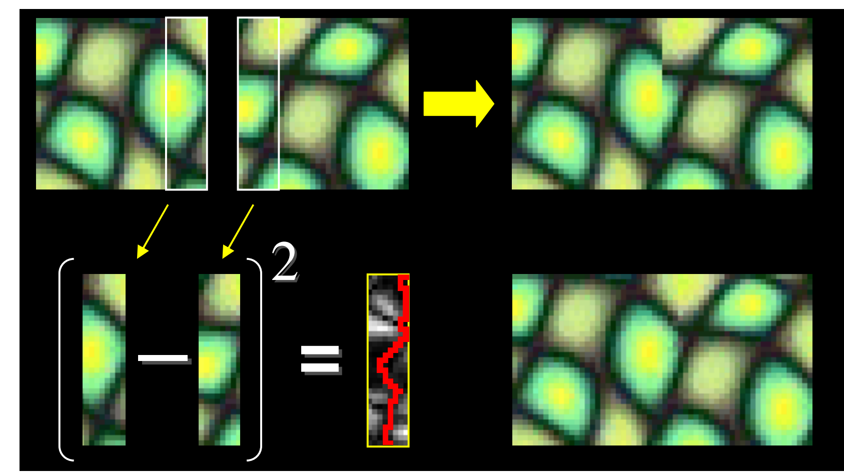
- Search input image for similar neighborhoods
  - Use Gaussian weighted SSD for search to emphasize central pixel
  - Sample one neighborhood at random
- Grow texture

Data-Driven Methods

Blocked Texture Synthesis







- Search input image for similar neighborhoods around block
- Grow texture by synthesizing blocks
    - Find boundary with minimum error (seam carving)

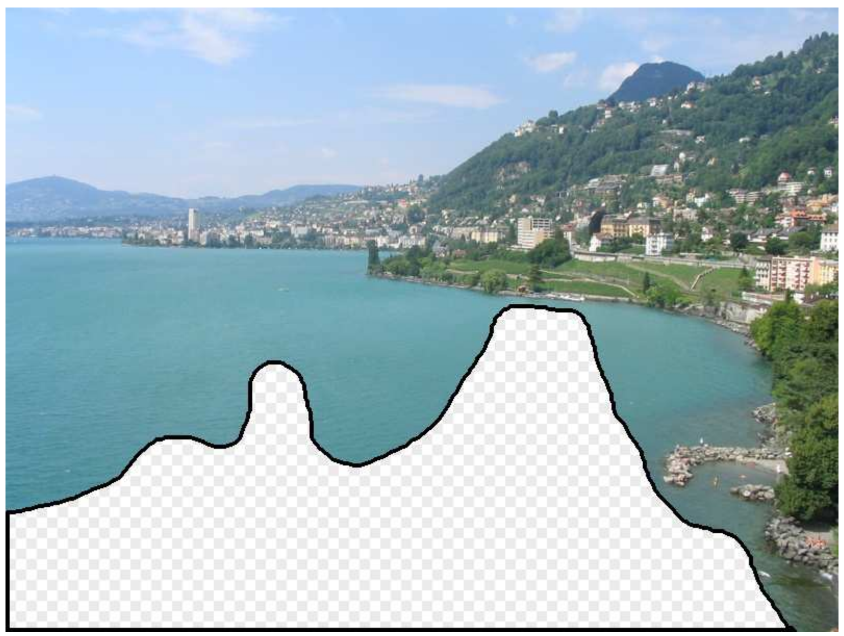
Data-Driven Methods

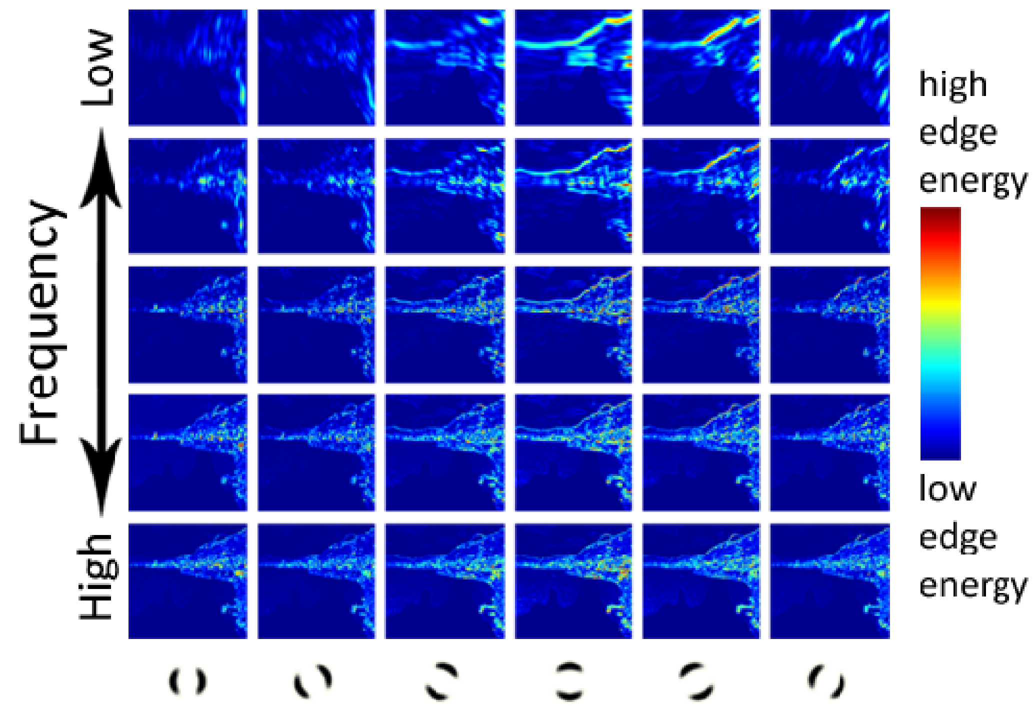
Lots of Data

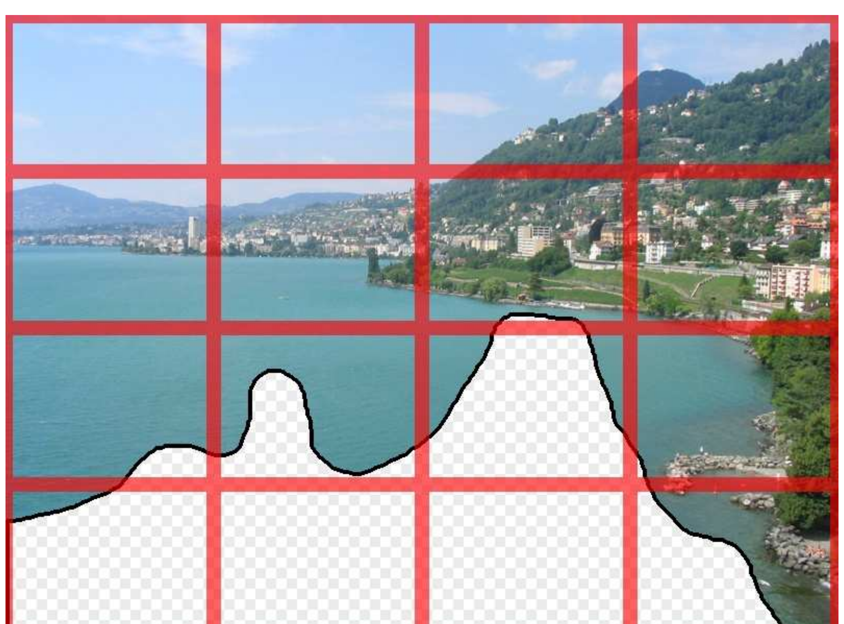
- Ex: Scene completion
    - Search millions of images on the Internet to find a patch that will complete your image

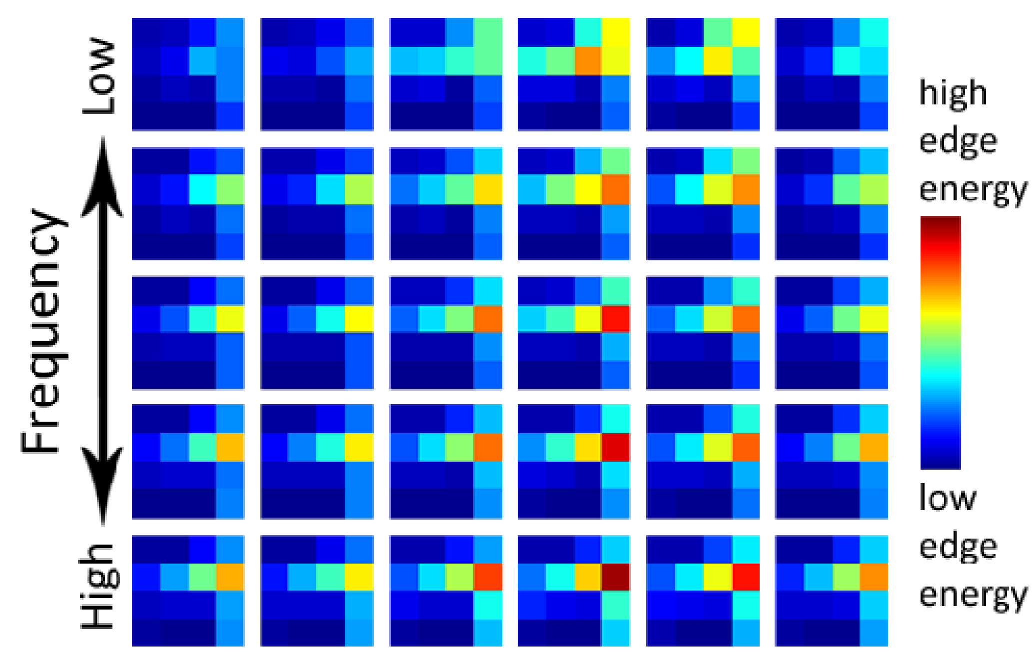
Data-Driven Methods

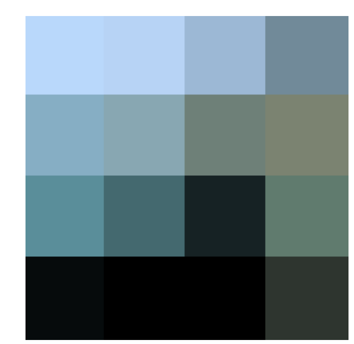
Scene Completion (GIST descriptor)







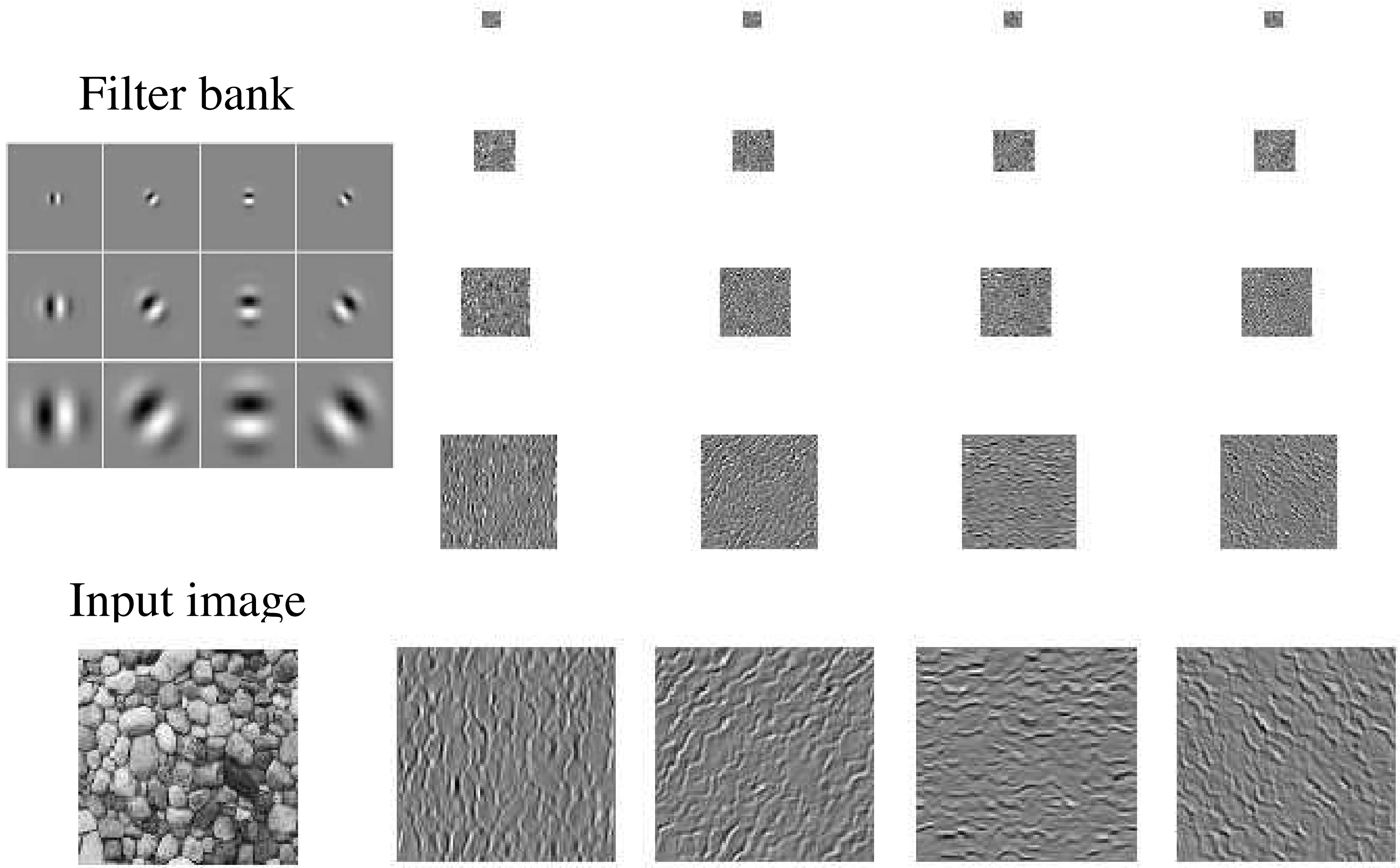




$$\chi^2(h_i,h_j)=\frac{1}{2}\sum_{m=1}^{K}\frac{[h_i(m)-h_j(m)]^2}{h_i(m)+h_j(m)}$$

Histogram matching distance

Data-Driven Methods



Data-Driven Methods

Lots of Data

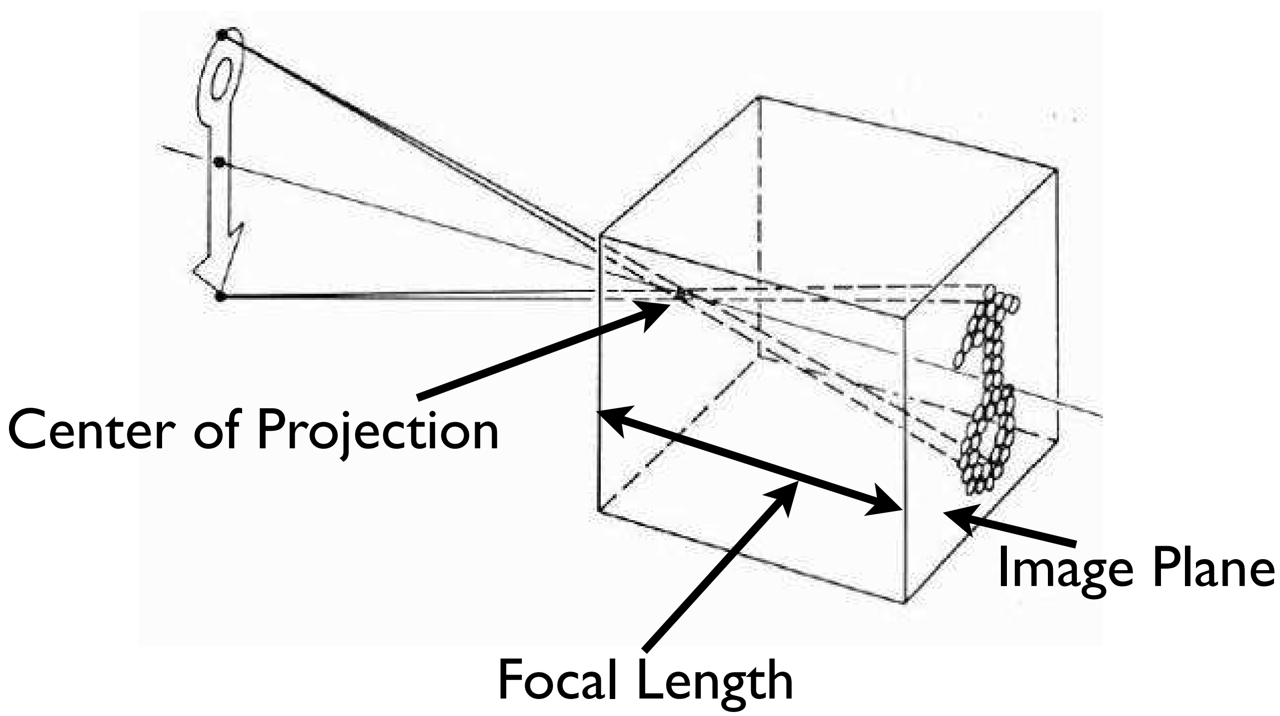
- Issues with Data
  - Sampling Bias
  - Photographer Bias
- Reduce Bias
  - Use autonomous techniques to capture data
  - StreetView, satellite, webcam etc.

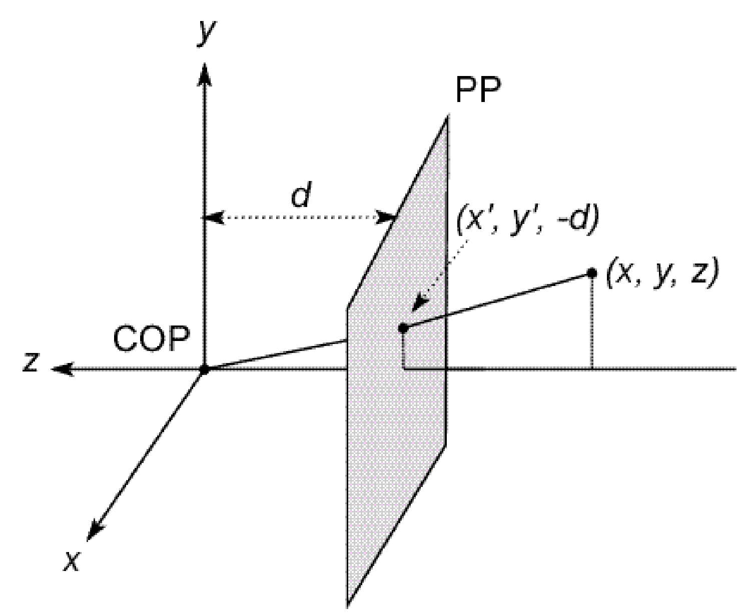
Review Topics

- Sampling and Reconstruction
- Frequency Domain and Filtering
- Blending
- Warping
- Data-driven Methods
- Camera
- Homographies
- Modeling Light

Camera

Pinhole Model





Perspective Projection

$$(x, y, z) \rightarrow \left(-d\frac{x}{z},\ -d\frac{y}{z},\ \cancel{-d}\right)$$

Camera

Pinhole Model

Perspective Projection (Matrix Representation)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \ -d\frac{y}{z})$$

Homogeneous Coordinates

Camera

Pinhole Model

Orthographic Projection (Matrix Representation)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

Camera

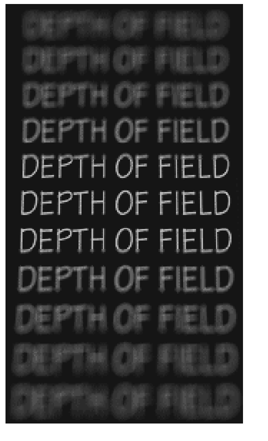
Pinhole Model

- Pinhole camera aperture
  - Large aperture --- blurry image
  - Small aperture --- not enough light, diffraction effects
- Lenses create sharp images with large aperture
  - Trade-off: only at a certain focus

Camera

Pinhole Model

- Depth of field
  - Distance over which objects are in focus
- Field of view
  - Angle of visible region



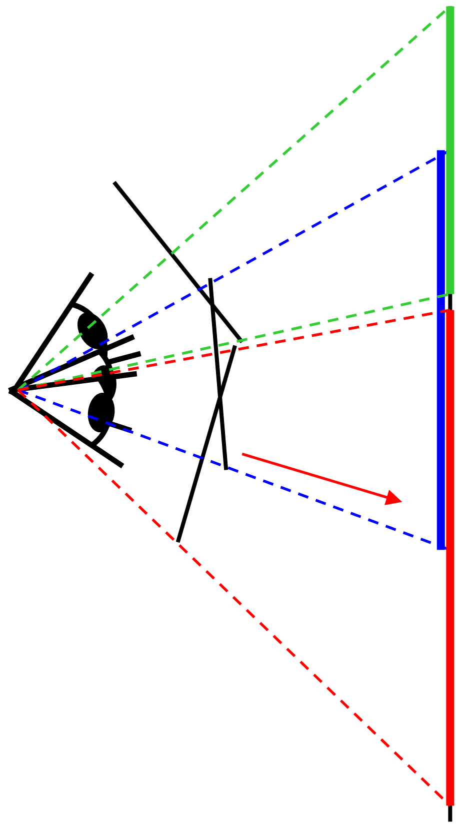
Review Topics

- Sampling and Reconstruction
- Frequency Domain and Filtering
- Blending
- Warping
- Data-driven Methods
- Camera
- Homographies
- Modeling Light

Homographies

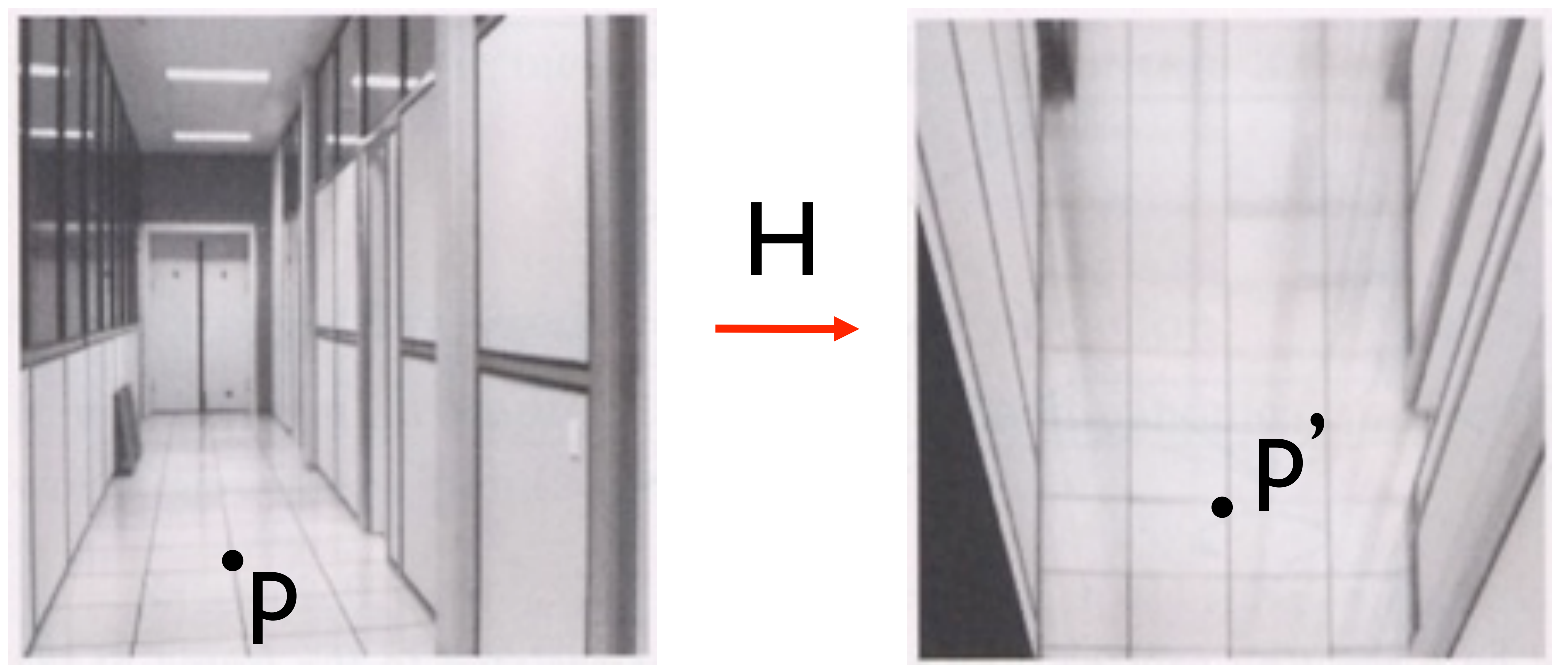
- Panorama
  - Reproject images onto a common plane
  - Images should have same center of projection

Mosaic:
Synthetic wide-angle camera



Projective Warp
(Homography)

Homographies



$$\begin{bmatrix} x'' \\ y'' \\ w'' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, x' = \frac{x''}{w''}, y' = \frac{y''}{w''}$$

Homographies

$$\begin{bmatrix} x'' \\ y'' \\ w'' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, x' = \frac{x''}{w''}, y' = \frac{y''}{w''}$$

- Expand equations and rewrite as

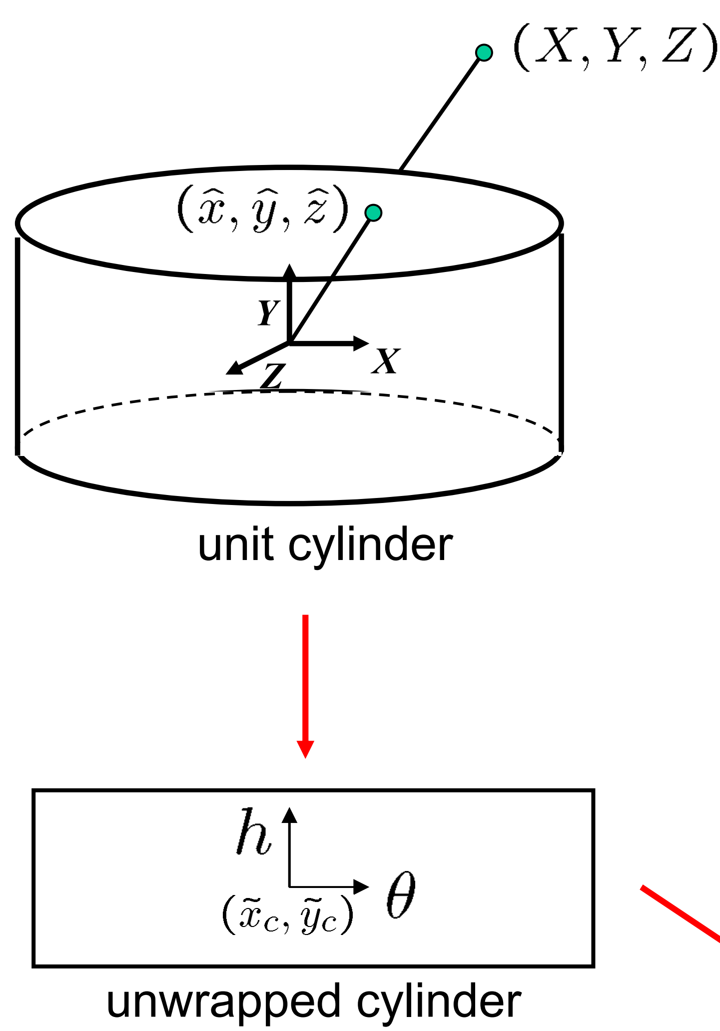
$$\mathbf{P}\mathbf{h} = \mathbf{q}$$

$$\mathbf{h} = \begin{bmatrix} a & b & c & d & e & f & g & h \end{bmatrix}^T$$

- Solve using least-squares (**h = P\q**)

Other Projection Models

Cylindrical Projection



- Map 3D point (X,Y,Z) onto cylinder

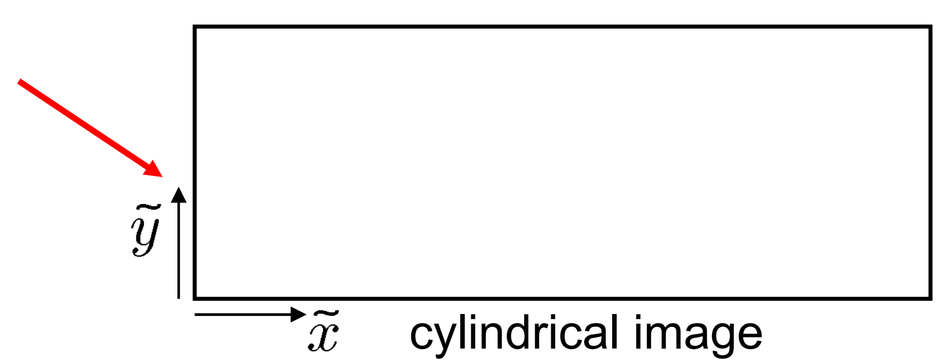
$$(\hat{x}, \hat{y}, \hat{z}) = \frac{1}{\sqrt{X^2+Z^2}}(X, Y, Z)$$

- Convert to cylindrical coordinates

$$(sin\theta, h, cos\theta) = (\hat{x}, \hat{y}, \hat{z})$$

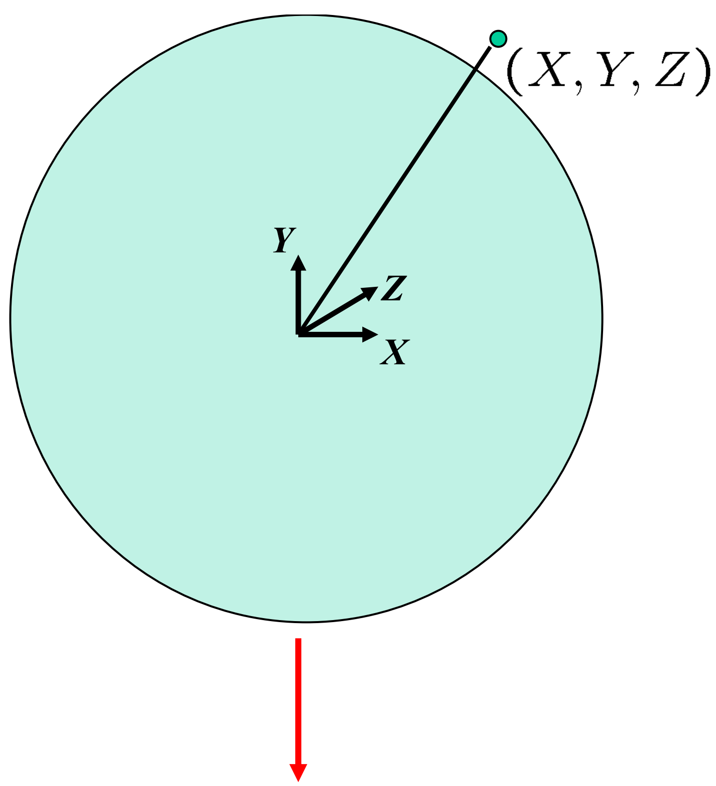
- Convert to cylindrical image coordinates

$$(\tilde{x}, \tilde{y}) = (f\theta, fh) + (\tilde{x}_c, \tilde{y}_c)$$



Other Projection Models

Spherical Projection



- Map 3D point (X,Y,Z) onto sphere

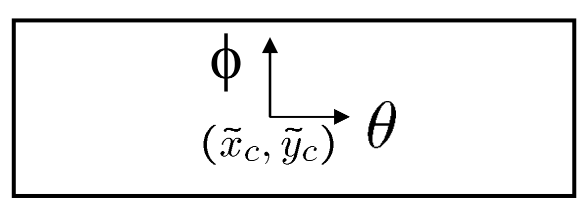
$$(\hat{x}, \hat{y}, \hat{z}) = \frac{1}{\sqrt{X^2 + Y^2 + Z^2}}(X, Y, Z)$$

- Convert to spherical coordinates

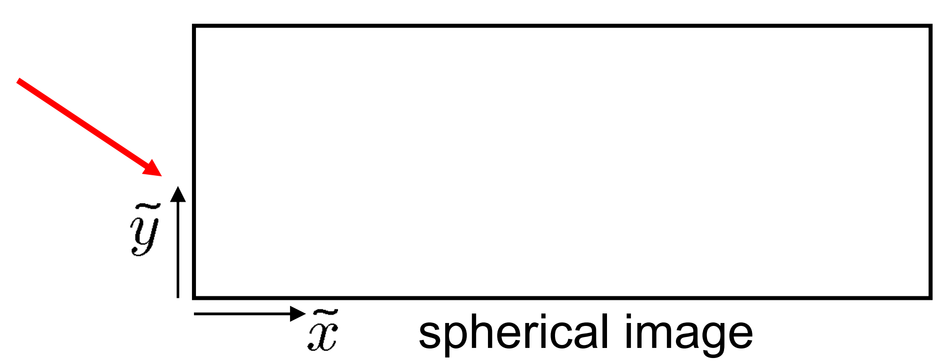
$$(\sin\theta\cos\phi, \sin\phi, \cos\theta\cos\phi) = (\hat{x}, \hat{y}, \hat{z})$$

- Convert to spherical image coordinates

$$(\tilde{x}, \tilde{y}) = (f\theta, fh) + (\tilde{x}_c, \tilde{y}_c)$$



unwrapped sphere



spherical image

Review Topics

- Sampling and Reconstruction
- Frequency Domain and Filtering
- Blending
- Warping
- Data-driven Methods
- Camera
- Homographies
- Modeling Light

Modeling Light

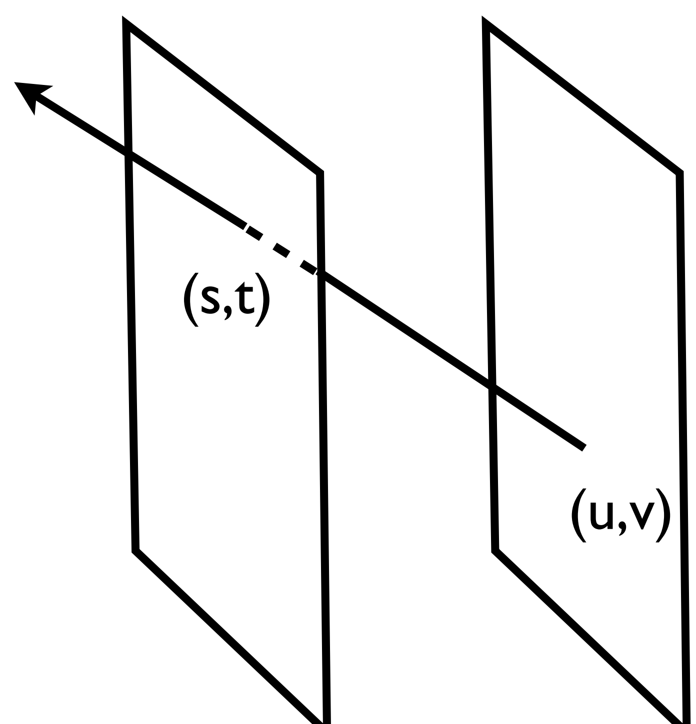
(The Omnipotent) Plenoptic Function

- Intensity of Light:
  - From all directions: θ, φ
  - At all wavelengths: λ
  - At all times: t
  - Seen from any viewpoint: V_x, V_y, V_z
- $P(\theta, \varphi, \lambda, t, V_x, V_y, V_z)$

Modeling Light

Lumigraph (Lightfield)

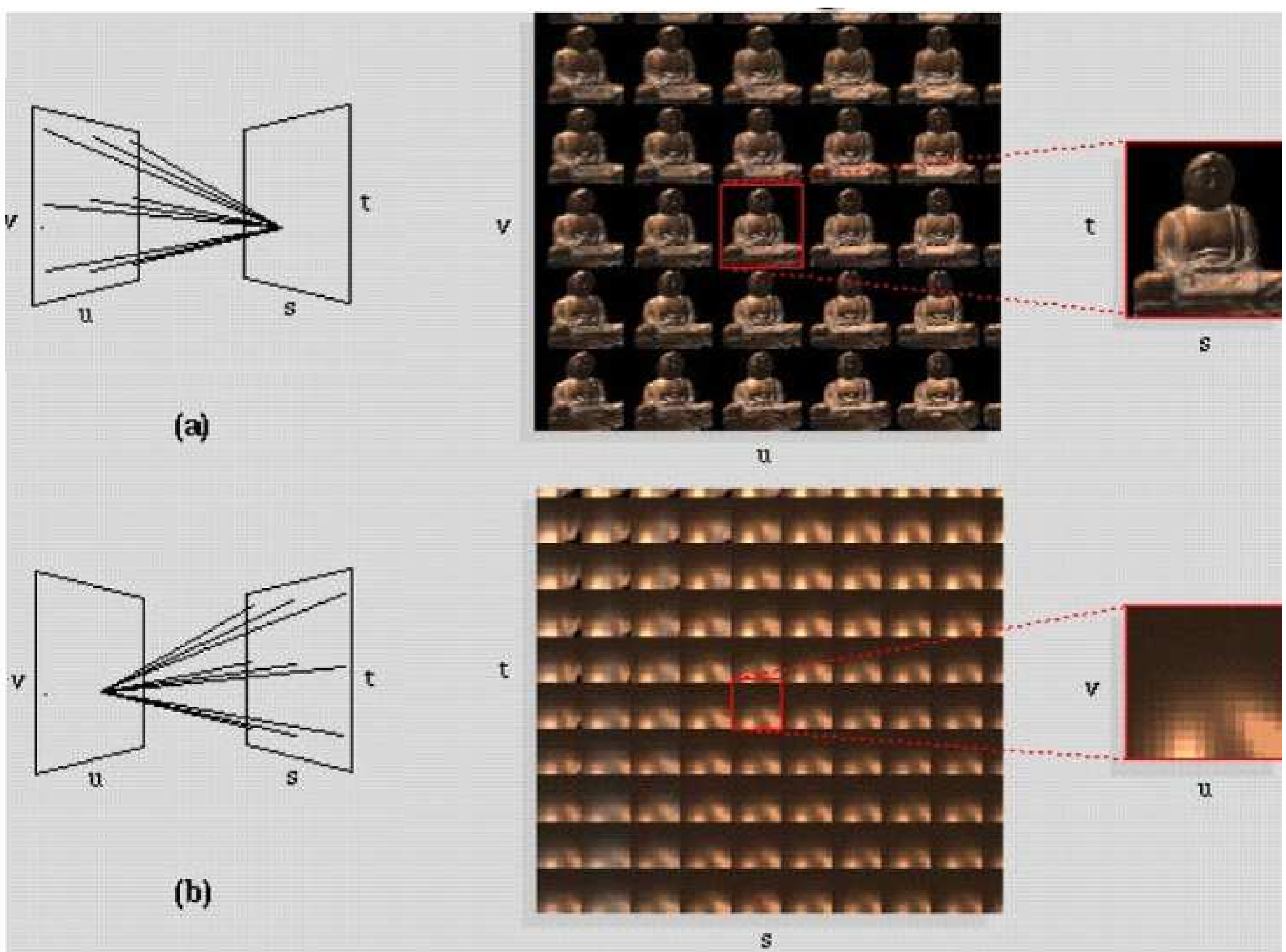
- Intensity along all lines
  - For all views (i.e. s,t), gives intensity at all points (i.e. u,v)
  - Captures to some extent $P(\theta, \varphi, V_x, V_y, V_z)$



For all (s,t,u,v)

Modeling Light

Lumigraph (Lightfield)



Modeling Light

Acquiring Lightfield

- Move camera in known steps over (s,t) using gantry
- Move camera anywhere over (s,t) and recover optimal field
- Use microlens array after main lens

Good Luck!!