EE421/521 Image Processing

Lecture 7

- GEOMETRIC TRANSFORMATIONS
- MULTIRESOLUTION REPRESENTATION

1

Recall: Image Resizing

400x600

500x400

2

Recall: Image Resizing

H↑5 → H-LPF: 1/5 → H↓4

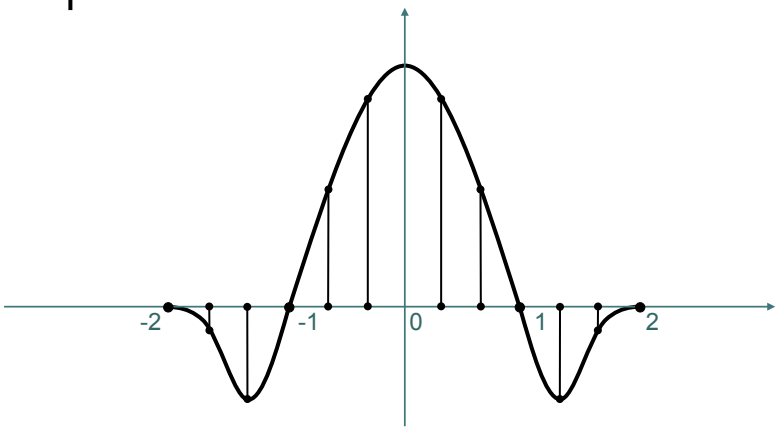
V↑2 → V-LPF: 1/3 → V↓3

400x600

500x400

3

Recall: Cubic Spline Kernel for 1/3 LPF



$$h[n] = \left\{-\frac{4}{27}, -\frac{8}{27}, 0, \frac{5}{9}, \frac{8}{9}, 1, \frac{8}{9}, \frac{5}{9}, 0, -\frac{8}{27}, -\frac{4}{27}\right\}$$

4

2D Geometric Transformations

5

2D Geometric Transformations (Global Warping)

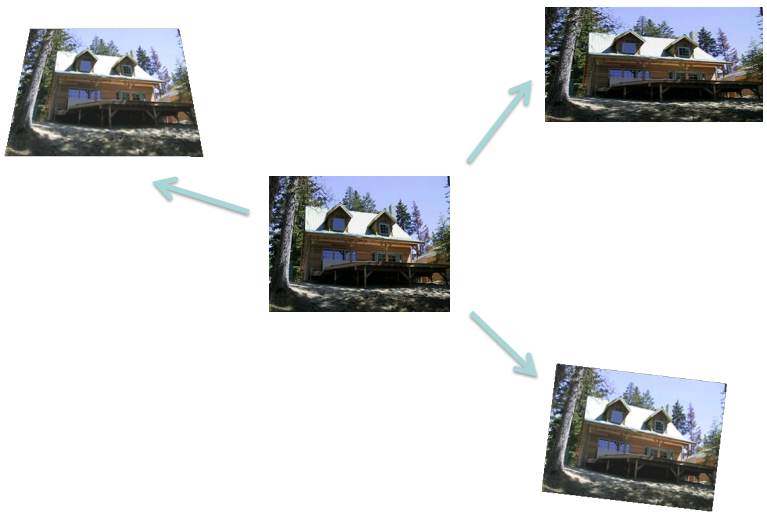
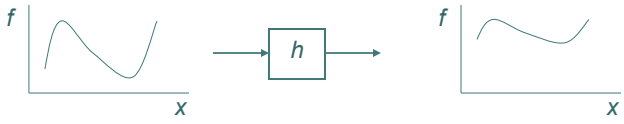


Image Filtering vs. Warping

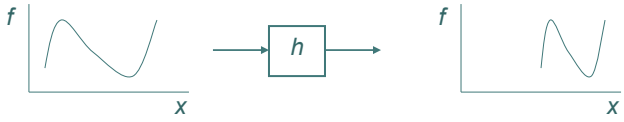
- image filtering: change *range* of image

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



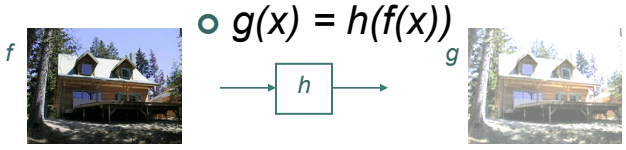
- image warping: change *domain* of image

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

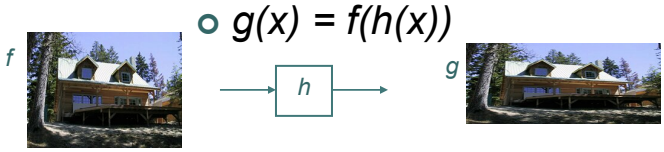


Filtering vs. Warping

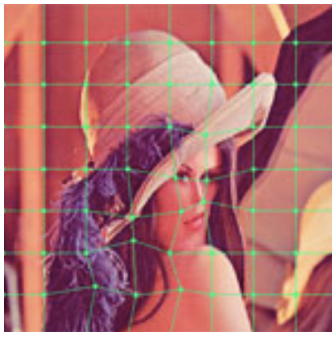
- image filtering: change *range* of image
  - $g(x) = h(f(x))$



- image warping: change *domain* of image
  - $g(x) = f(h(x))$



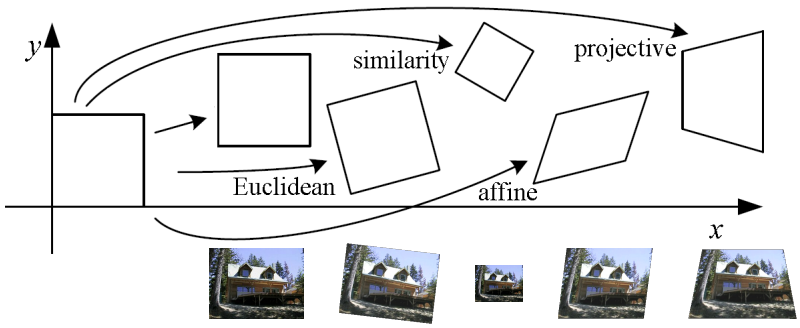
Local Warping



9

2D Geometric Transformations

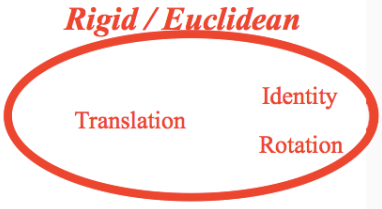
translation rotation scale shear perspective

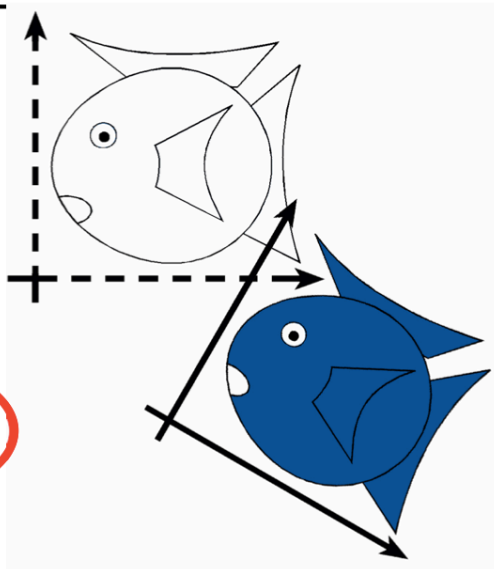


Example: Perspective Distortion

Rigid-Body / Euclidean Transforms

- Preserves distances
- Preserves angles

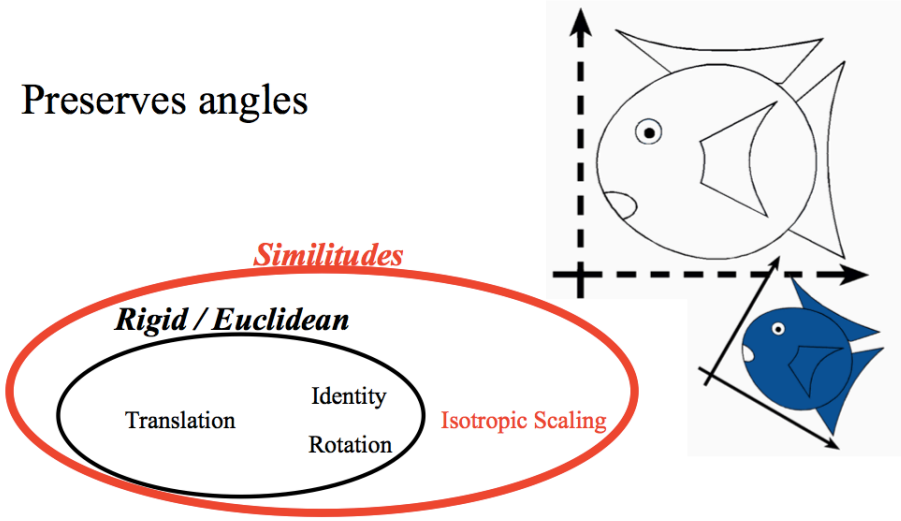




MIT EECS 6.837, Durand and Cutler

Similitudes / Similarity Transforms

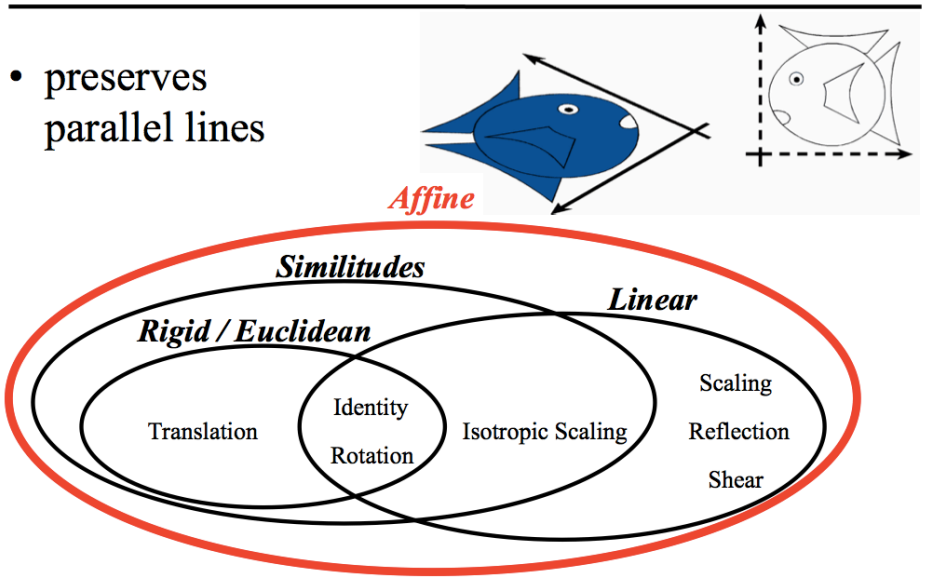
- Preserves angles



MIT EECS 6.837, Durand and Cutler

Affine Transformations

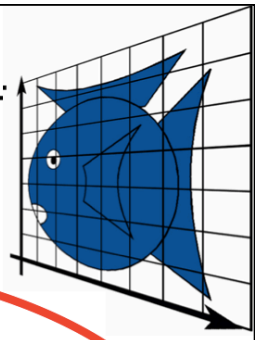
- preserves parallel lines

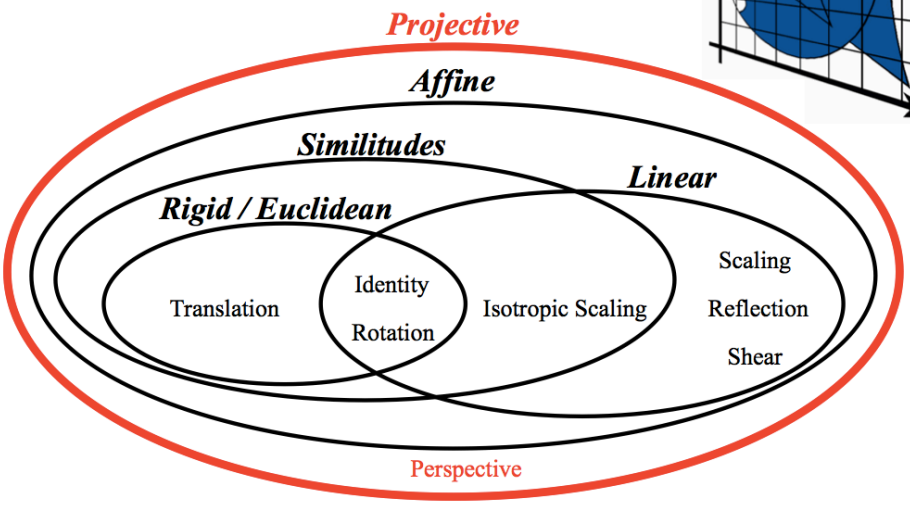


MIT EECS 6.837, Durand and Cutler

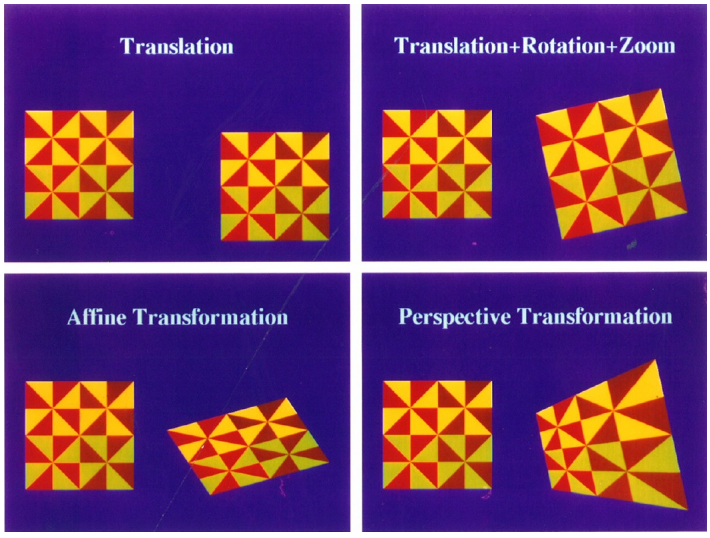
Projective Transformations

- preserves lines

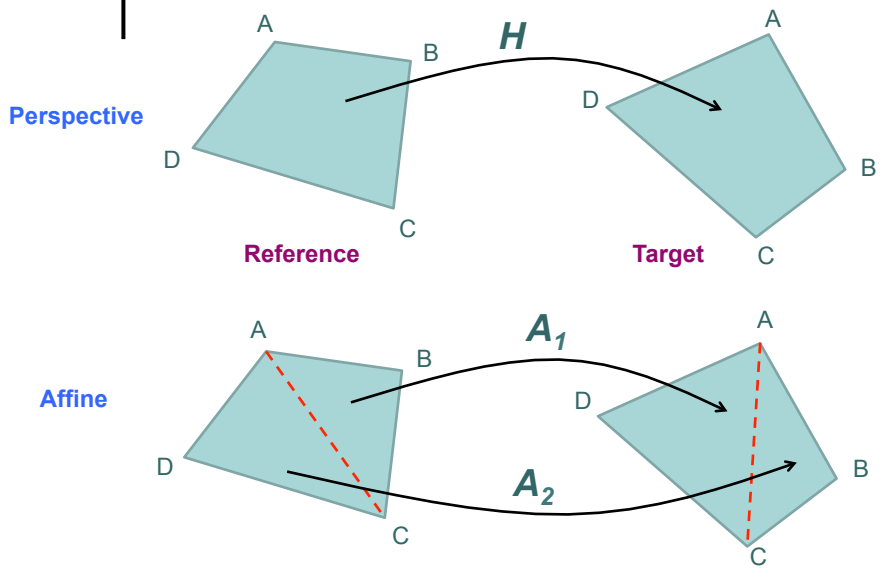




Summary of 2D Transformations

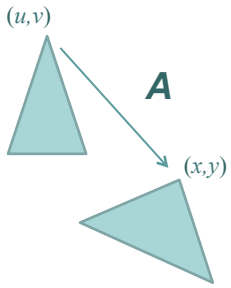


Affine vs. Perspective



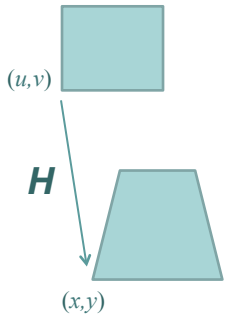
Affine Mapping Problem

- Given 3 matching corner locations (u,v)➔(x,y)
- Find the affine transformation ***A*** to map the triangles

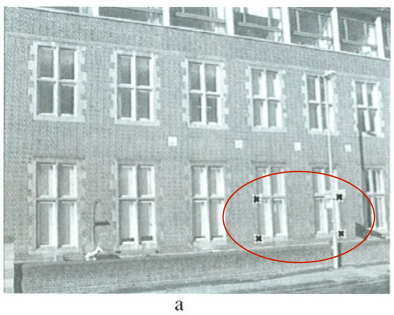


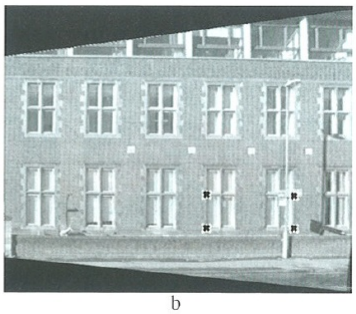
Perspective Mapping Problem

- Given 4 matching corner locations (u,v)➔(x,y)
- Find the perspective transformation ***H*** to map the quadrilaterals



Application: Removing Perspective Distortion


a


b

Finding Affine and Perspective Mapings

21

2D Affine Transformation

$$x = a_{11}u + a_{12}v + a_{13}$$
$$y = a_{21}u + a_{22}v + a_{23}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Subsets of Affine Transformation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

translation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

rotation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

scale

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & h_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

shear

Transformations Do Not Commute!

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Rotation followed by translation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x\cos\theta \\ \sin\theta & \cos\theta & t_y\cos\theta \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Translation followed by rotation

2D Perspective Transformation

$$x = \frac{a_{11}u + a_{12}v + a_{13}}{a_{31}u + a_{32}v + 1} = \frac{x'}{w}$$

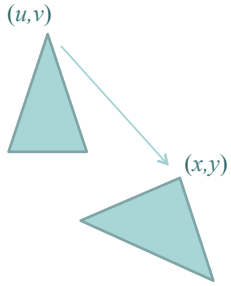
$$y = \frac{a_{21}u + a_{22}v + a_{23}}{a_{31}u + a_{32}v + 1} = \frac{y'}{w}$$

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Hierarchy of 2D Transformations

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

Finding Affine Parameters



$$x = a_{11}u + a_{12}v + a_{13}$$
$$y = a_{21}u + a_{22}v + a_{23}$$

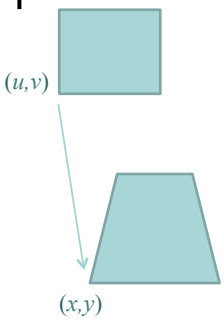
- 1 point correspondence provides 2 equations
- 6 parameters can be solved from 3 point correspondences

Equation to Solve for Affine Parameters

$$\begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_2 & v_2 & 1 \\ u_3 & v_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_3 & v_3 & 1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{bmatrix}$$

28

Finding Affine Parameters



$$x = \frac{a_{11}u + a_{12}v + a_{13}}{a_{31}u + a_{32}v + 1} = \frac{x'}{w}$$

$$y = \frac{a_{21}u + a_{22}v + a_{23}}{a_{31}u + a_{32}v + 1} = \frac{y'}{w}$$

- 1 point correspondence provides 2 equations
- 8 parameters can be solved from 4 point correspondences

$$a_{11}u + a_{12}v + a_{13} - a_{31}ux - a_{32}vx = x$$

$$a_{21}u + a_{22}v + a_{23} - a_{31}uy - a_{32}vy = y$$

Equation to Solve for Perspective Parameters

$$\begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -ux_1 & -vx_1 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -uy_1 & -vy_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -ux_2 & -vx_2 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -uy_2 & -vy_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -ux_3 & -vx_3 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -uy_3 & -vy_3 \\ u_4 & v_4 & 1 & 0 & 0 & 0 & -ux_4 & -vx_4 \\ 0 & 0 & 0 & u_4 & v_4 & 1 & -uy_4 & -vy_4 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ x_4 \\ x_4 \end{bmatrix}$$

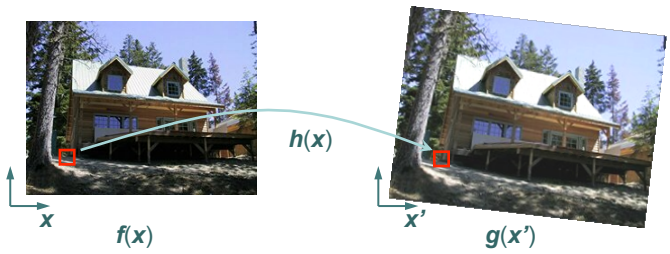
30

Calculating the Transformed Image

31

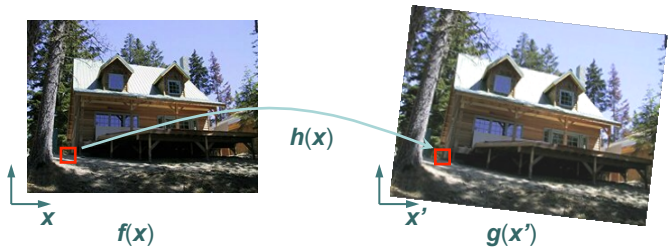
Image Warping

- Given a coordinate transform $\boldsymbol{x'} = \boldsymbol{h}(\boldsymbol{x})$ and a source image $\boldsymbol{f}(\boldsymbol{x})$, how do we compute a transformed image $\boldsymbol{g}(\boldsymbol{x'}) = \boldsymbol{f}(\boldsymbol{h}(\boldsymbol{x}))$?



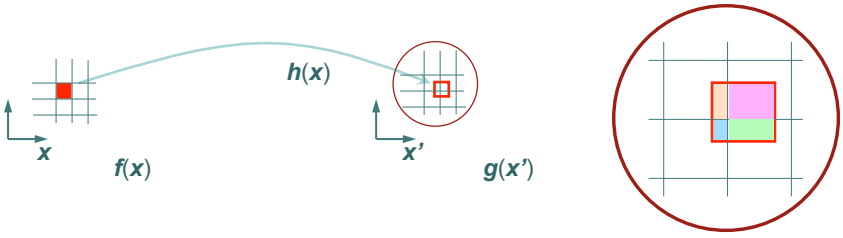
Forward Mapping

- Send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$
- What if pixel lands "between" two pixels?



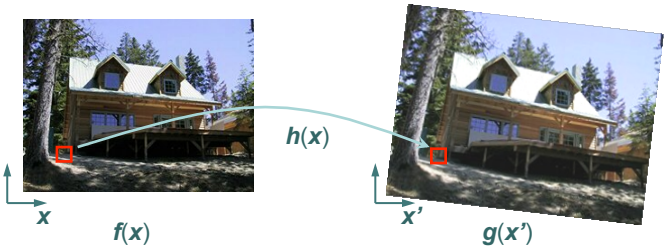
Forward Mapping

- Send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$
- What if pixel lands "between" two pixels?
- Answer: add *contribution* to several pixels, then normalize



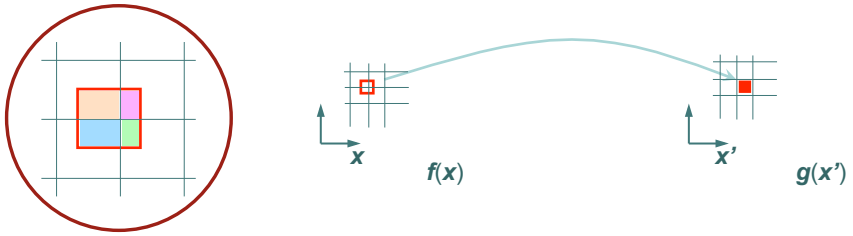
Backward Mapping

- Get each pixel ***g*(*x'*)** from its corresponding location ***x'* = *h*(*x*)** in ***f*(*x*)**
- What if pixel comes from "between" two pixels?

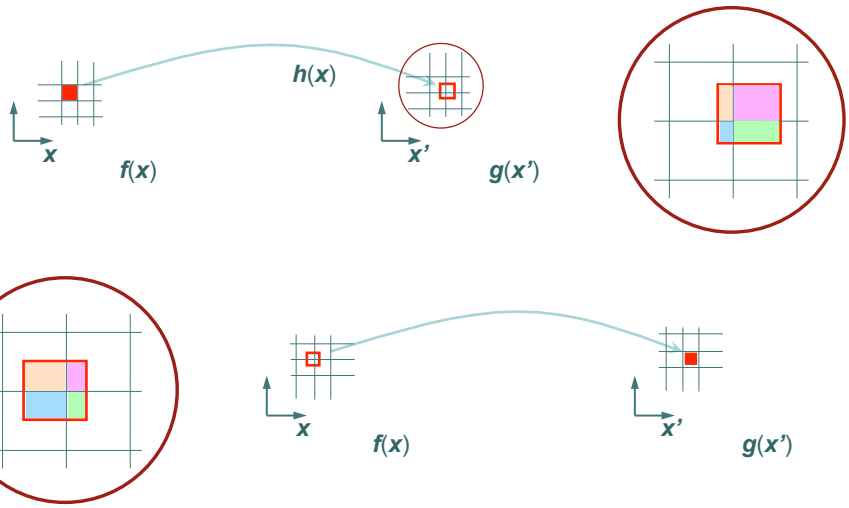


Backward Mapping

- Get each pixel ***g*(*x'*)** from its corresponding location ***x'* = *h*(*x*)** in ***f*(*x*)**
- What if pixel comes from "between" two pixels?
- Answer: *interpolate* color value from the source image



Forward vs. Backward Mapping



Interpolation

- Possible interpolation filters:
  - nearest neighbor
  - bilinear
  - bicubic
  - FIR sinc

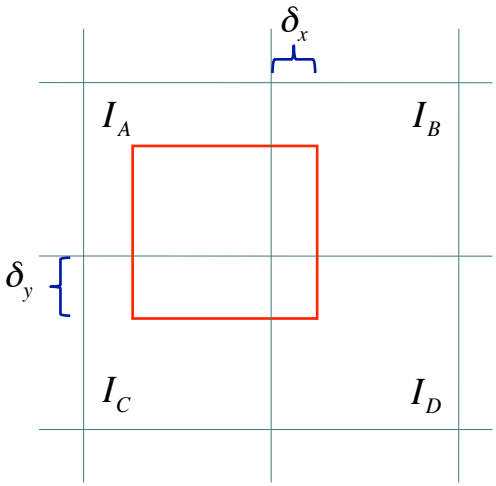
Needed to prevent “jaggines”

Bilinear vs. Nearest Neighbor Interpolation

original | nearest neighbour | bilinear

Bilinear Interpolation

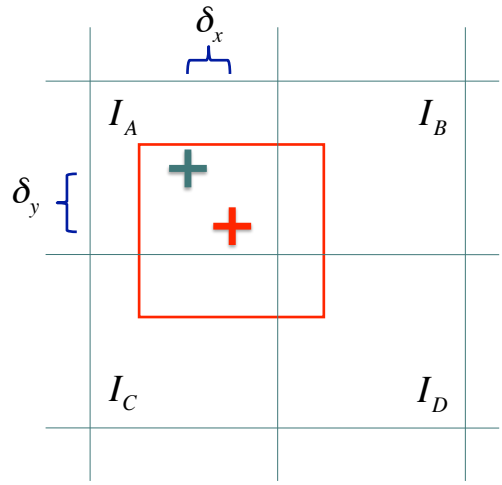
Interpolated value is a linear combination of the intensity values of the four closest pixels:



40

$$I = (1-\delta_x)(1-\delta_y)I_A + \delta_x(1-\delta_y)I_B + (1-\delta_x)\delta_y I_C + \delta_x\delta_y I_D$$

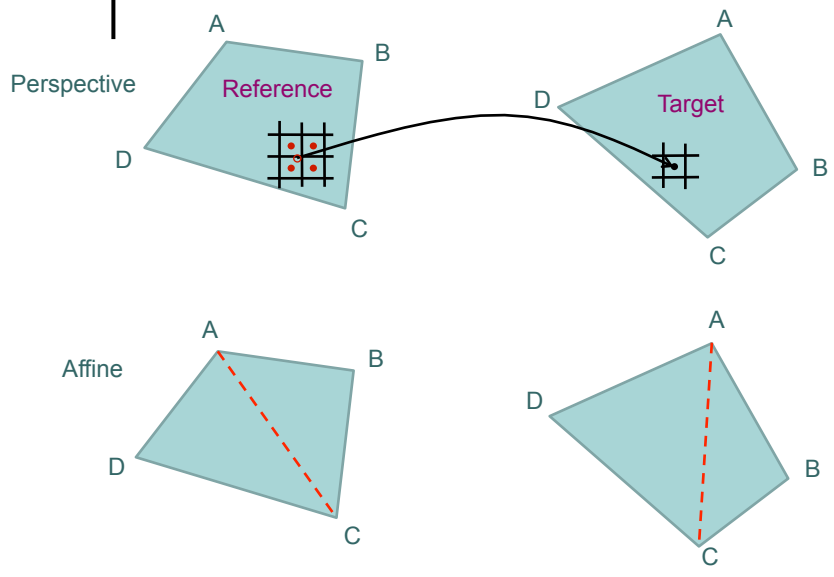
Bilinear Interpolation



41

$$I = (1-\delta_y)\left((1-\delta_x)I_A + \delta_x I_B\right) + \delta_y\left((1-\delta_x)I_C + \delta_x I_D\right)$$

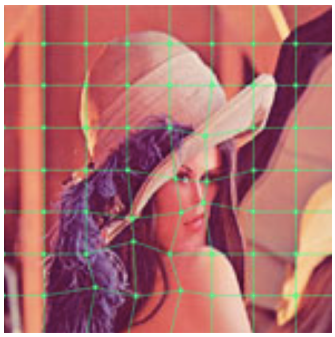
Mappings



Bilinear Maping

43

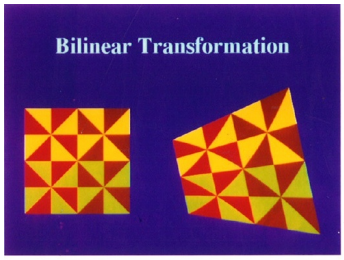
Local Warping



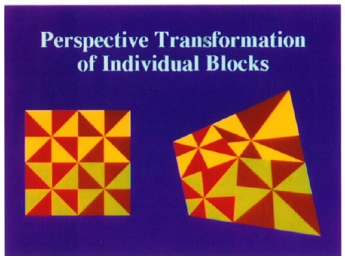
44

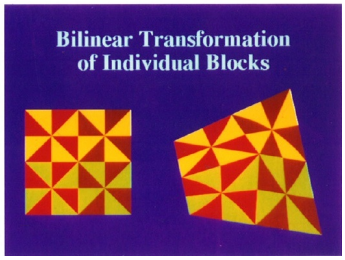
Perspective vs. Bilinear Mapping

(-) Diagonal lines map to quadratic curves



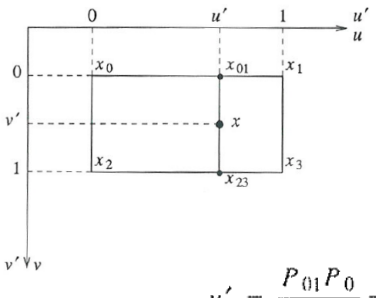
(+) Bilinear transformation provides continuous mapping



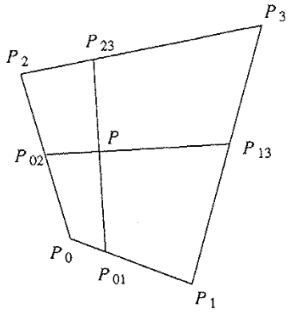


Bilinear Mapping

Reference image



Target image



$$u' = \frac{P_{01}P_0}{P_1P_0} = \frac{P_{23}P_2}{P_3P_2}$$

$$v' = \frac{P_{02}P_0}{P_2P_0} = \frac{P_{13}P_1}{P_3P_1}$$

Digital Image Warping
George Wolberg

Multiresolution Image Representation

47

Multiresolution Image Representation Techniques

- Gaussian Pyramid
- Laplacian Pyramid
- Subband Decomposition
- Wavelet Decomposition

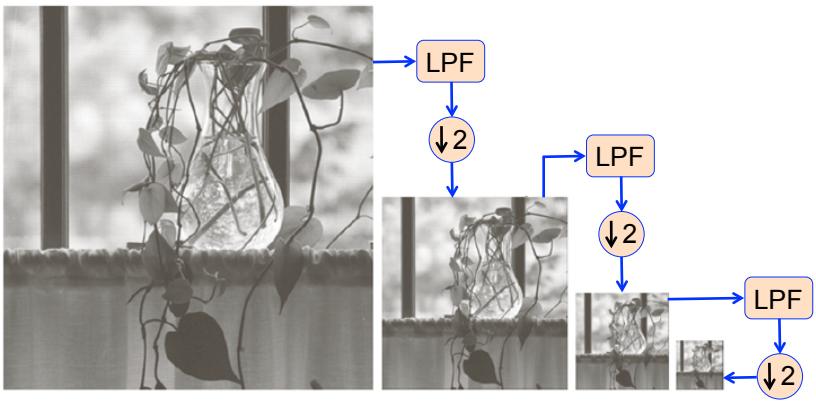
48

Gaussian Pyramid (MipMap)

Oversampled by 33% (max.)

49

Gaussian Pyramid (MipMap)



50

Application: Computer Graphics

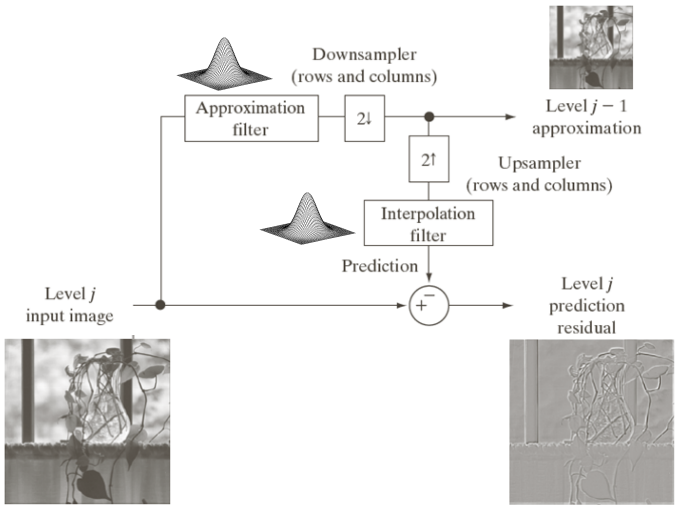
51

Laplacian Pyramid

Oversampled by 33% (max.)

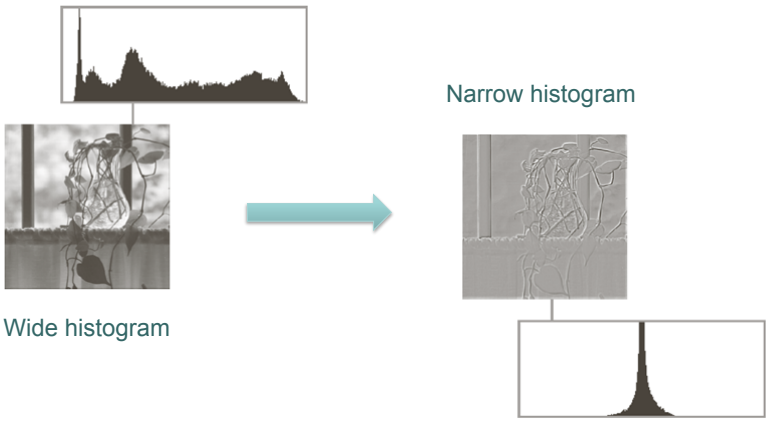
52

Laplacian Pyramid



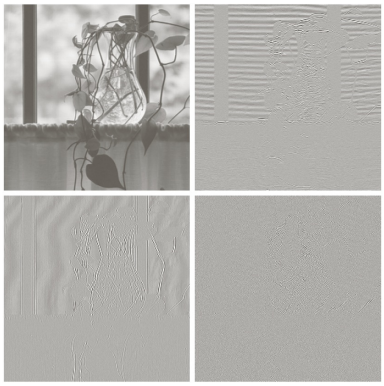
53

Application: Image Compression



54

Subband Representation: Uniform Decomposition

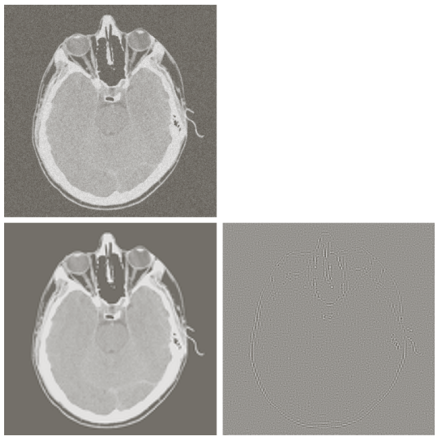


Critically sampled

55

Application: Noise Filtering

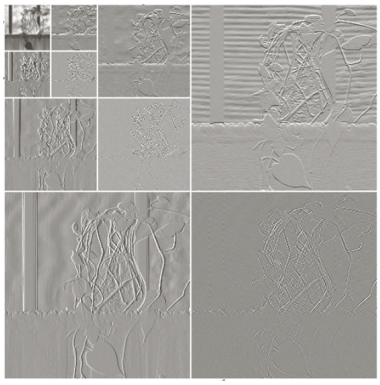
Threshold detail (high-frequency) coefficients



Detail removed

56

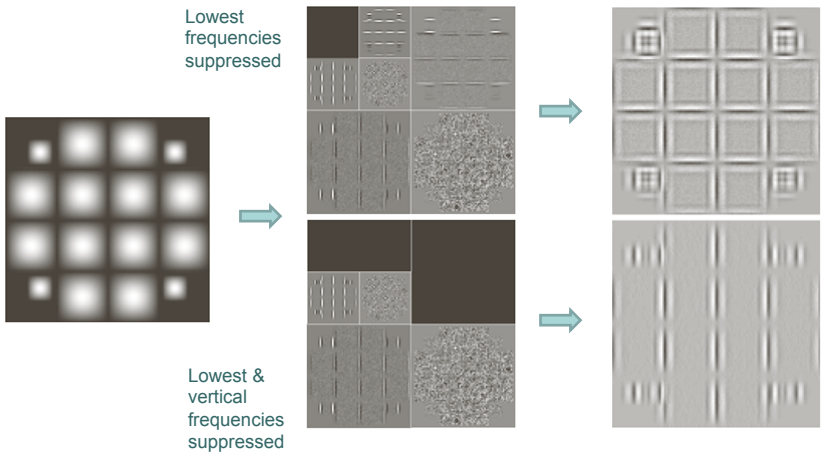
Subband Representation: Wavelet Decomposition



Critically sampled

57

Application: Edge Detection

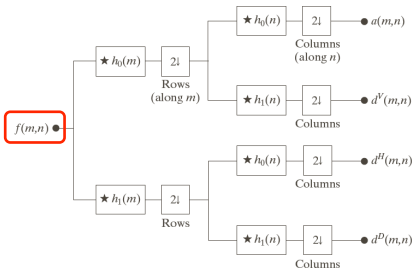


58

Subband Decomposition

59

Subband Decomposition (Separable)



$h_0(\cdot)$: low pass filter

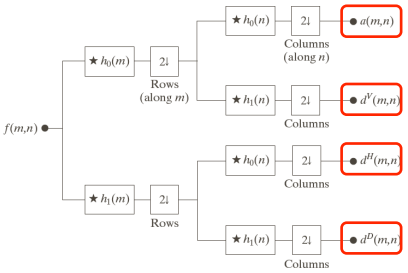
$h_1(\cdot)$: high pass filter

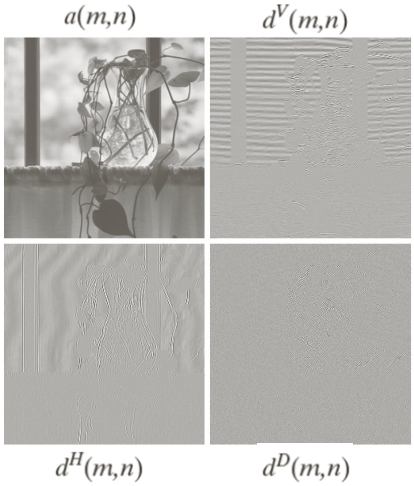
$2\downarrow$: downsample by 2

$2\uparrow$: upsample by 2

60

Subband Decomposition (Separable)



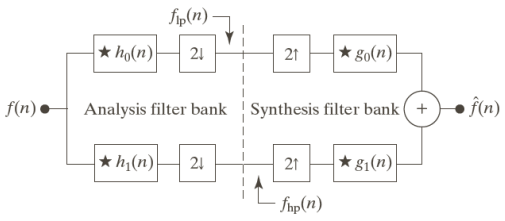


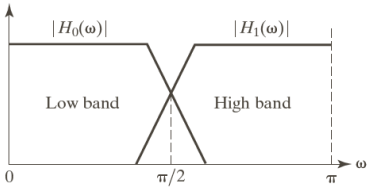
61

Analysis (Decomposition) and Synthesis Filters

h_0, h_1 : analysis filters

g_0, g_1 : synthesis filters





62

Fourier Domain Equations

aliasing

Analysis

$$F_{lp}(\omega) = \frac{1}{2}F(\omega/2)H_0(\omega/2) + \frac{1}{2}F(\pi+\omega/2)H_0(\pi+\omega/2)$$

$$F_{hp}(\omega) = \frac{1}{2}F(\omega/2)H_1(\omega/2) + \frac{1}{2}F(\pi+\omega/2)H_1(\pi+\omega/2)$$

Synthesis ⟶ $\hat{F}(\omega) = F_{lp}(2\omega)G_0(\omega) + F_{hp}(2\omega)G_1(\omega)$

63

Alias-Free Perfect Reconstruction Filters

aliasing

$$\hat{F}(\omega) = \left(\frac{1}{2}H_0(\omega)G_0(\omega) + \frac{1}{2}H_1(\omega)G_1(\omega) + H_0(\pi+\omega)G_0(\omega) + H_1(\pi+\omega)G_1(\omega)\right)F(\omega)$$

Alias Cancellation ⟹ $H_0(\pi+\omega)G_0(\omega) + H_1(\pi+\omega)G_1(\omega) = 0$

Perfect Reconstruction ⟹ $H_0(\omega)G_0(\omega) + H_1(\omega)G_1(\omega) = 2$

64

Orthonormal FIR Filter Banks

Prototype filter $\Rightarrow$ $h_0(m)$

$$h_1(m) = (-1)^m h_0(-m+1)$$ *modulation*

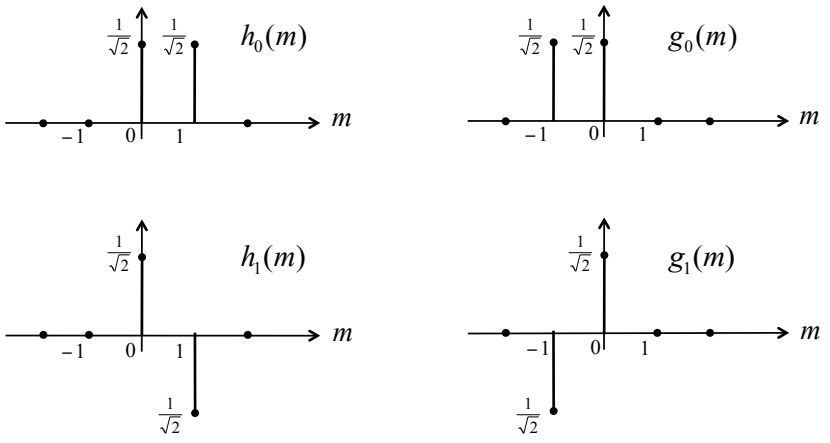
$$g_i(m) = h_i(-m), \quad i = 0,1$$ *time reversal*

$$\sum_m h_i(m) h_j(m+2n) = \begin{cases} 1 & i = j \;\&\; n = 0 \\ 0 & \text{otherwise} \end{cases}$$

Conservation of energy: Total distortion in the image is equal to the sum of distortions in the subbands

65

Example: Haar Filters



66

Haar Filters: Alias-Free Perfect Reconstruction

$$H(\omega) = \sum_n h(n) e^{-j\omega n}$$

$$H_0(\omega) = \frac{1 + e^{-j\omega}}{\sqrt{2}} \Rightarrow H_0(\pi + \omega) = \frac{1 - e^{-j\omega}}{\sqrt{2}}$$

$$H_1(\omega) = \frac{1 - e^{-j\omega}}{\sqrt{2}} \Rightarrow H_1(\pi + \omega) = \frac{1 + e^{-j\omega}}{\sqrt{2}}$$

$$G_0(\omega) = \frac{1 + e^{j\omega}}{\sqrt{2}}$$

$$G_1(\omega) = \frac{1 - e^{j\omega}}{\sqrt{2}}$$

$$\Rightarrow \hat{F}(\omega) = F(\omega)$$

67

Daubechies Orthonormal FIR Filters (normalized to 2)

D2 (Haar)	D4	D6	D8	D10
1	0.6830127	0.47046721	0.32580343	0.22641898
1	1.1830127	1.14111692	1.01094572	0.85394354
	0.3169873	0.650365	0.8922014	1.02432694
	-0.1830127	-0.19093442	-0.03957503	0.19576696
		-0.12083221	-0.26450717	-0.34265671
		0.0498175	0.0436163	-0.04560113
			0.0465036	0.10970265
			-0.01498699	-0.00882680
				-0.01779187
				4.71742793e-3

$h_0(m)$

$$h_1(m) = (-1)^n h_0(-m+1)$$

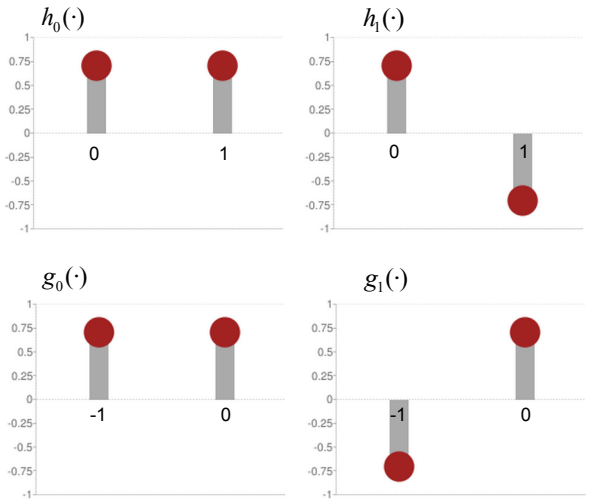
$$g_i(m) = h_i(-m), \quad i = 0,1$$

Orthogonality:

$$\sum_m h_i(m) h_j(m+2n) = \begin{cases} 1 & i = j \;\&\; n = 0 \\ 0 & \text{otherwise} \end{cases}$$

68

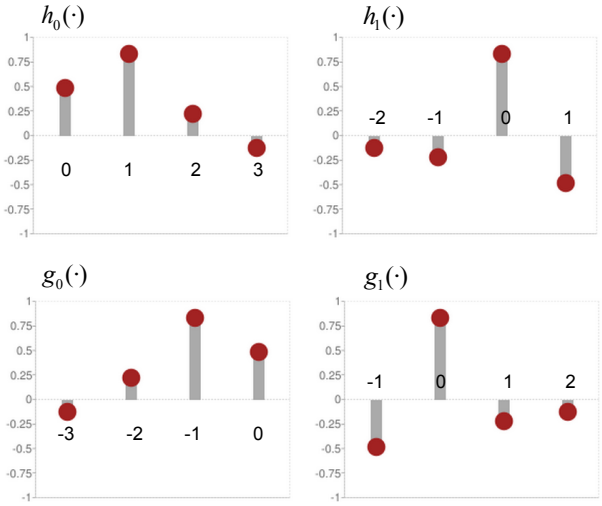
Daubechies 2 (Haar)



$$h_1(m) = (-1)^n h_0(-m+1)$$
$$g_i(m) = h_i(-m), \quad i = 0,1$$

69

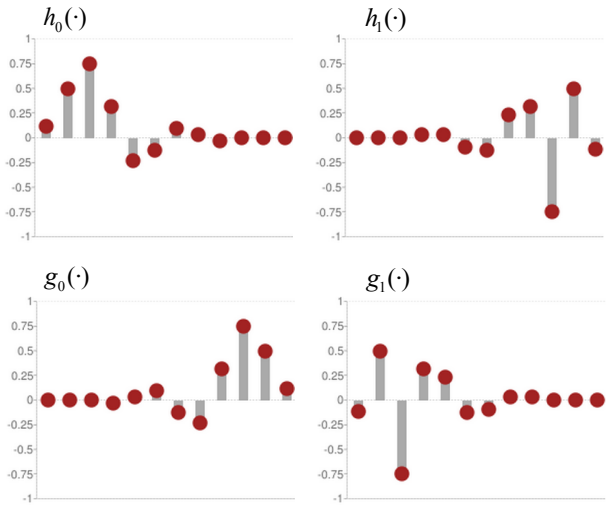
Daubechies 4



$$h_1(m) = (-1)^n h_0(-m+1)$$
$$g_i(m) = h_i(-m), \quad i = 0,1$$

70

Daubechies 12



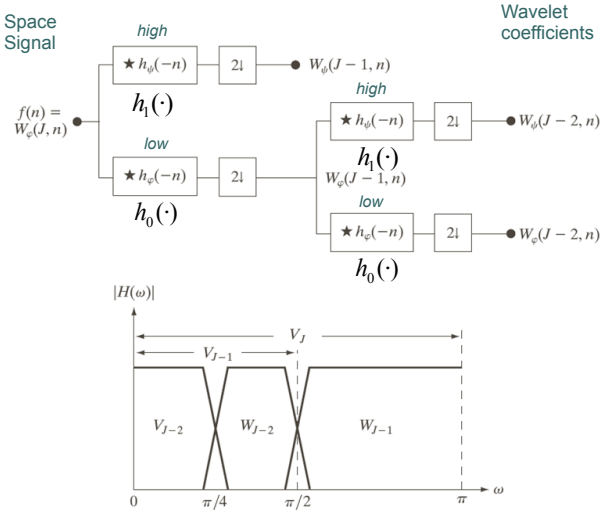
$$h_1(m) = (-1)^n h_0(-m+1)$$
$$g_i(m) = h_i(-m), \quad i = 0,1$$

71

Wavelet (Scale-Space) Decomposition

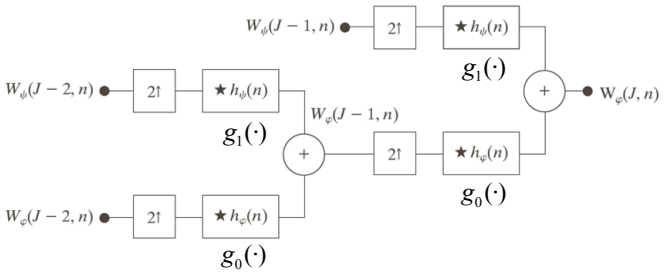
72

Wavelet Analysis



73

Wavelet Synthesis



74

Example

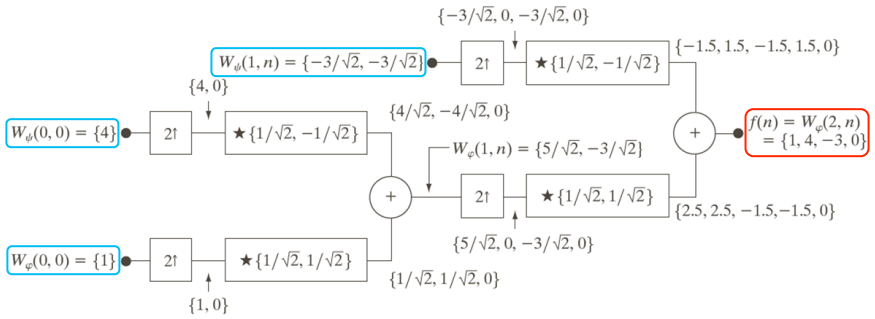
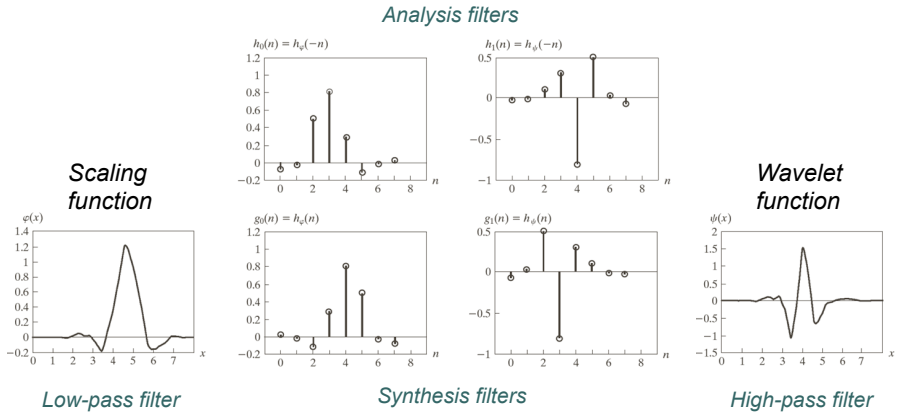


FIGURE 7.22 Computing a two-scale inverse fast wavelet transform of sequence $\{1, 4, -1.5\sqrt{2}, -1.5\sqrt{2}\}$ with Haar scaling and wavelet functions.

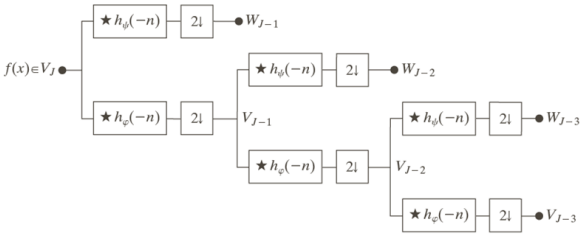
75

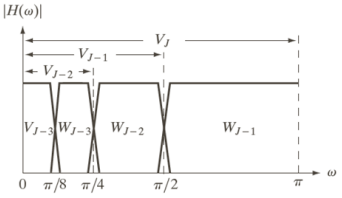
Scaling and Wavelet Functions



76

Three-Scale Wavelet Decomposition



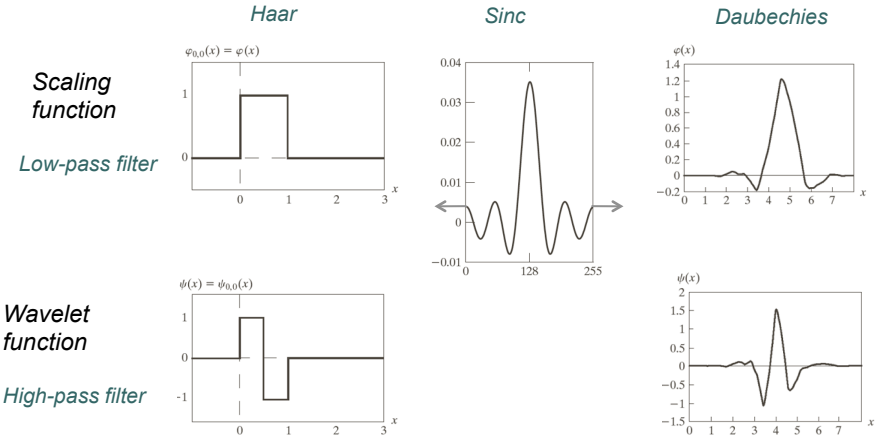


Low spatial resolution at low frequencies

High spatial resolution at high frequencies

77

Space-Frequency Representation



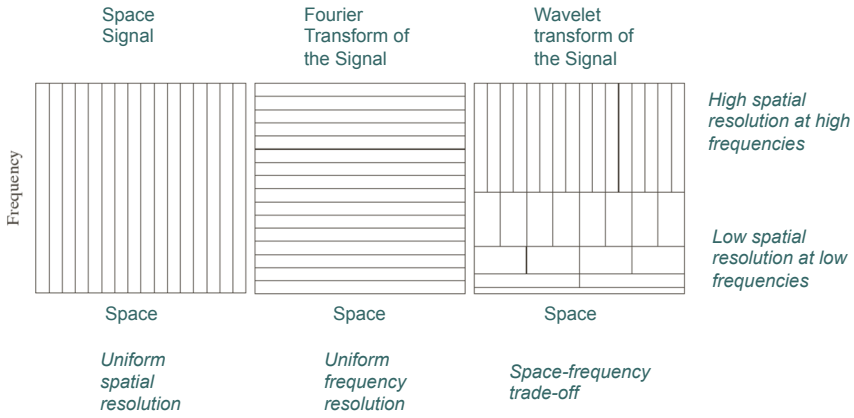
78

Space localization

Frequency localization

Space-frequency trade-off

Space-Frequency Localization



79

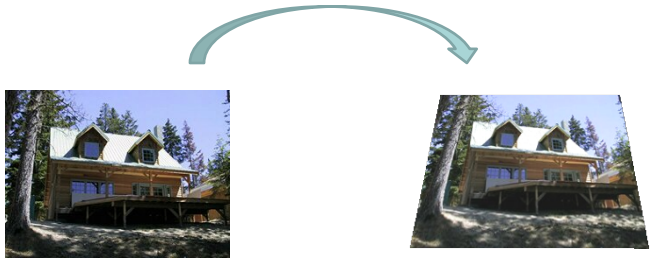
Project 1.7

Geometric Transform

Due 24.11.2013

80

Image Warping



81

Problem 1.7

1. Select an arbitrary color image.
2. Assuming that the original image support is (0,0), (1,0), (1,1), (0,1), warp the image into a quadrilateral support given by (0,0), (1,0), (0.8,1.3), (0.2,1.3), respectively, using perspective mapping. Use bilinear interpolation and backward mapping. Display the warped image.
3. Repeat Step 3 using affine mapping where the two triangles are separated by the diagonal line defined by the points (0,0) and (1,1). Again use bilinear interpolation and backward mapping. Display the warped image.
4. Calculate the RMSE between the two warped images obtained in Steps 2 and 3.
5. Comment on the RMSE value and any visual difference between the two warped images.

82

Root Mean Squared Error

$$RMSE = \left(\frac{1}{L} \sum_{m} \sum_{n} \left(s_{\text{ref}}[m,n] - \hat{s}_{\text{target}}[m',n'] \right)^2 \right)^{1/2}$$

where L is the total number of pixels used in the above double summation.

Next Lecture

- LINE PROCESSING

84