Image Reconstruction

Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- In this lecture we describe image reconstruction:
  - Interpolation as convolution
  - Interpolation kernels for:
    - Nearest neighbor
    - Triangle filter
    - Cubic convolution
    - B-Spline interpolation
    - Windowed sinc functions

Introduction

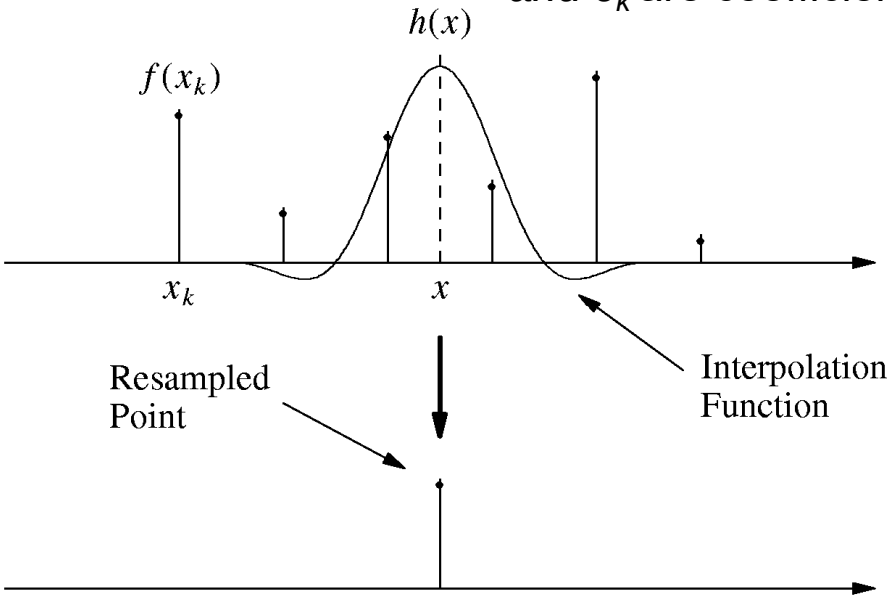
- Reconstruction is synonymous with interpolation.
- Determine value at position lying between samples.
- Strategy: fit a continuous function through the discrete input samples and evaluate at any desired set of points.
- Sampling generates infinite bandwidth signal.
- Interpolation reconstructs signal by smoothing samples with an interpolation function (kernel).

Interpolation

For equi-spaced data, interpolation can be expressed as a convolution:

$$f(x) = \sum_{k=0}^{K-1} c_k h(x - x_r)$$

where *K* is the number of neighborhood pixels *and* c_k are coefficients for kernel *h*



Kernel Samples: h(-d), h(-1-d), h(1-d), h(2-d)

If h is symmetric, h(d), h(1+d), h(1-d), h(2-d)

Wolberg: Image Processing Course Notes

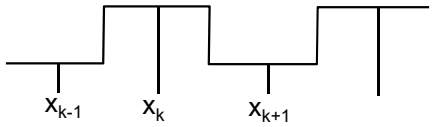
Interpolation Kernel

- Set of weights applied to neighborhood pixels
- Often defined analytically
- Usually symmetric: $h(x) = h(-x)$
- Commonly used kernels:
  - Nearest neighbor (pixel replication)
  - Triangle filter (linear interpolation)
  - Cubic convolution (smooth; used in digital cameras)
  - Windowed sinc functions (highest quality, more costly)

Nearest Neighbor

$$\text{Interpolating Polynomial: } f(x) = f(x_k) \quad \frac{x_{k-1} + x_k}{2} < x \le \frac{x_k + x_{k+1}}{2}$$

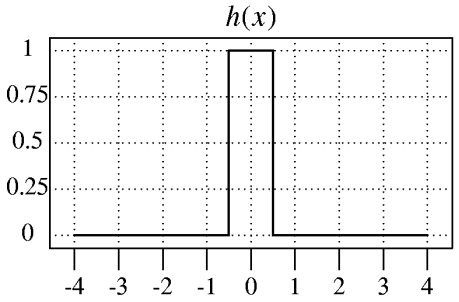
$$\text{Interpolating Kernel: } h(\mathrm{x}) = \begin{cases} 1 & 0 \le |\mathrm{x}| < 0.5 \\ 0 & 0.5 \le |x| \end{cases}$$

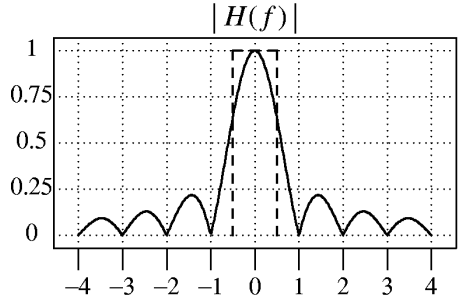


Other names: box filter, sample-and hold function, and Fourier window.

Poor stopband. NN achieves magnification by pixel replication. Very blocky.

Shift errors of up to 1/2 pixel are possible. Common in hardware zooms.





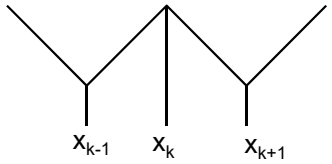
Triangle Filter

$$\text{Interpolating Polynomial: } f(x) = a_1 x + a_0$$

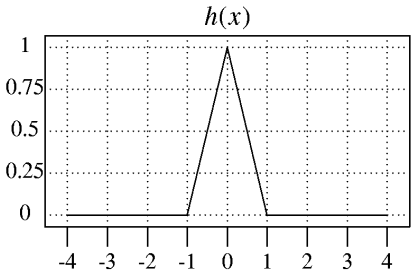
$$[f_0 \;\; f_1] = [a_1 \;\; a_0] \begin{bmatrix} x_0 & x_1 \\ 1 & 1 \end{bmatrix} \quad \textit{Solve for } a_1, a_0$$

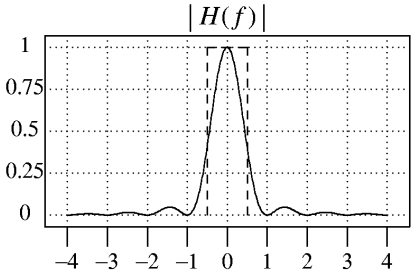
$$f(x) = f_0 + \left[\frac{x - x_0}{x_1 - x_0} \right] (f_1 - f_0)$$

$$\text{Interpolation Kernel: } h(x) = \begin{cases} 1 - |x| & 0 \le |x| < 1 \\ 0 & 1 \le |x| \end{cases}$$

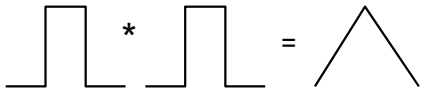


Other names for *h*: triangle filter, tent filter, roof function, chateau function, and Bartlett window.





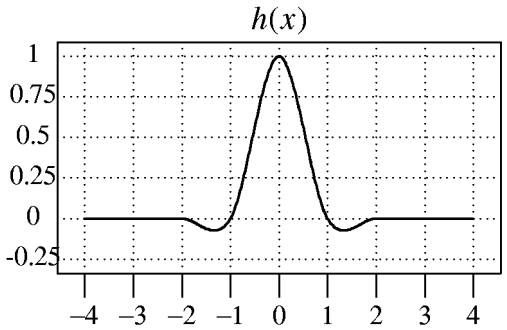
In the frequency domain:



Sinc x Sinc = Sinc^2

Cubic Convolution (1)

Third degree approximation to sinc. Its kernel is derived from constraints imposed on the general cubic spline interpolation formula.



$$h(x) = \begin{cases} a_{30}|x|^3 + a_{20}|x|^2 + a_{10}|x| + a_{00} & 0 \le |x| < 1 \\ a_{31}|x|^3 + a_{21}|x|^2 + a_{11}|x| + a_{01} & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases}$$

Determine coefficients by applying following constraints:

1. h(0) = 1 and h(x) = 0 for |x| = 1, 2
2. h must be continuous at |x| = 0, 1, 2
3. h must have a continuous first derivative at |x| = 0, 1, 2

Cubic Convolution (2)

Constraint (1) states that when *h* is centered on an input sample, the interpolation function is independent of neighboring samples.

First 2 constraints give 4 equations:

$$1 = h(0) = a_{00}$$

$$0 = h(1^-) = a_{30} + a_{20} + a_{10} + a_{00}$$

$$0 = h(1^+) = a_{31} + a_{21} + a_{11} + a_{01}$$

$$0 = h(2^-) = 8a_{31} + 4a_{21} + 2a_{11} + a_{01}$$

3 more equations are obtained from constraint (3):

$$-a_{10} = h'(0^-) = h'(0^+) = a_{10}$$

$$3a_{30} + 2a_{20} + a_{10} = h'(1^-) = h'(1^+) = 3a_{31} + 2a_{21} + a_{11}$$

$$12a_{31} + 4a_{21} + a_{11} = h'(2^-) = h'(2^+) = 0$$

Total :7 equations, 8 unknowns $\rightarrow$ free variable $(a = a_{31})$

$$h(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & 0 \le |x| < 1 \\ a|x|^3 - 5a|x|^2 + 8a\,|x| - 4a & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases}$$

Cubic Convolution (3)

How to pick a? Add some heuristics (make it resemble Sinc function):

$h''(0) = -2(a+3) < 0 \rightarrow a > -3$ *Concave downward at x = 0*

$h''(1) = -4a > 0$ *Concave upward at x = 1*

This bounds *a* to the [-3, 0] range.

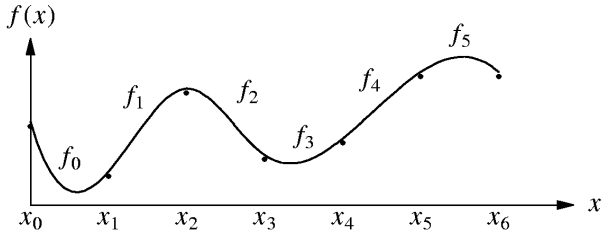
Common choices:

a = -1 matches the slope of sinc at x=1 (sharpens image)

a = -0.5 makes the Taylor series approximately agree in as many terms as possible with the original signal

a = -.75 sets the second derivative of the 2 cubic polynomials in h to 1 (continuous 2nd derivative at x = 1)

Cubic Splines (1)



$$f_k(x) = a_3(x - x_k)^3 + a_2(x - x_k)^2 + a_1(x - x_k) + a_0$$

6 polynomial segments, each of 3rd degree.
f_k's are joined at x_k (for k=1,…, n-2) such that f_k, f'_k, *and* f''_k are continuous.

$$a_0 = y_k$$
$$a_1 = y'_k \qquad where \quad \Delta y_k = y_{k+1} - y_k$$
$$a_2 = 3\Delta y_k - 2y'_k - y'_{k+1}$$
$$a_3 = -2\Delta y_k + y'_k + y'_{k+1}$$

(proof in App. 2)

Cubic Splines (2)

- The derivatives may be determined by solving the following system of linear equations:

$$\begin{bmatrix} 2 & 4 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & & \ddots & & \\ & & & & 1 & 4 & 1 \\ & & & & & 4 & 2 \end{bmatrix} \begin{bmatrix} y_0' \\ y_1' \\ y_2' \\ y_3' \\ \vdots \\ y_{n-2}' \\ y_{n-1}' \end{bmatrix} = \begin{bmatrix} -5y_0 + 4y_1 + y_2 \\ 3(y_2 - y_0) \\ 3(y_3 - y_1) \\ 3(y_4 - y_2) \\ \vdots \\ 3(y_{n-1} - y_n - 3) \\ -y_{n-3} - 4y_{n-2} + 5y_{n-1} \end{bmatrix}$$

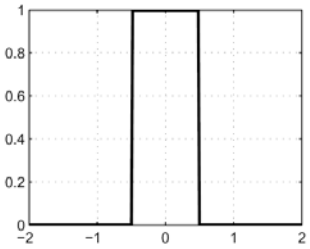
↑

global dependencies

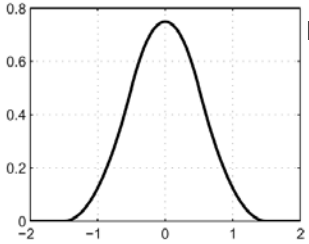
- Introduced by the constraints for continuity in the first and second derivatives at the knots.

B-Splines

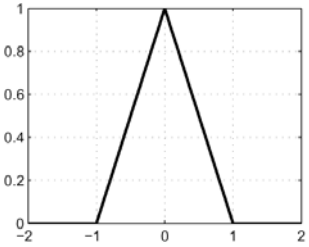
To analyze cubic splines, introduce cubic B-Spline interpolation kernel:



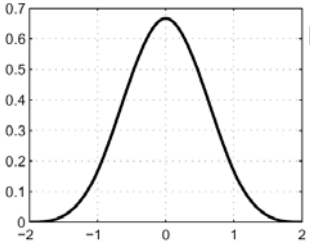
B_0 box filter



$B_2 = B_0 * B_0 * B_0$



$B_1 = B_0 * B_0$



$B_3 = B_0 * B_0 * B_0 * B_0$

$$h(x) = \frac{1}{6}\begin{cases} 3|x|^3 - 6|x|^2 + 4 & 0 \le |x| < 1 \\ -|x|^3 + 6|x|^2 - 12|x| + 8 & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases}$$

Parzen Window: Not interpolatory because it does not satisfy h(0) = 1, and h(1) = h(2) = 0. Indeed, it approximates the data.

Interpolatory B-Splines

$$f(x_j) = \sum_{k=j-2}^{j+2} c_k h(x_j - x_k)$$

Since $h(0) = \frac{4}{6},\ h(-1) = h(1) = \frac{1}{6},\ h(-2) = h(2) = 0$

we have $f(x_j) = \frac{1}{6}(c_{j-1} + 4c_j + c_{j+1})$

$$\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-2} \\ f_{n-1} \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 4 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & \ddots & & & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 4 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{bmatrix}$$

$$F = K C$$

$$C = K^{-1} F$$

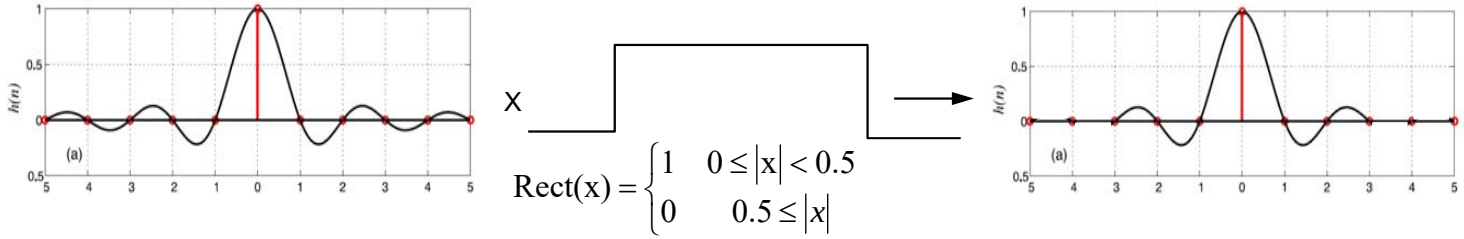
$K^{-1} \leftarrow$ inverse of tridiagonal matrix; Computation is O(n)

All previous methods used data values for c_k from $C = K^{-1}F$.

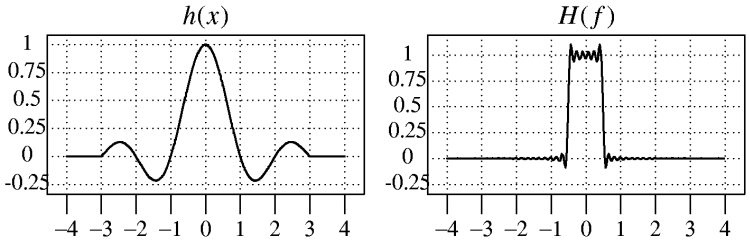
Truncated Sinc Function

- Alternative to previous kernels: use windowed sinc function.

Truncated Sinc



Truncating in spatial domain = convolving spectrum (box) with a Sinc function.



Ringing can be mitigated by using a smoothly tapering windowing function.

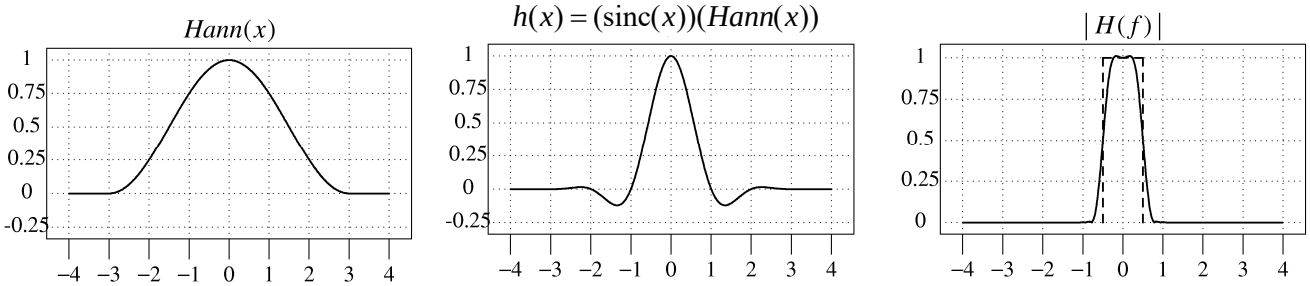
Popular window functions: Hann, Hamming, Blackman, Kaiser, and Lanczos.

Hann/Hamming Window

$$Hann/Hamming(x) = \begin{cases} \alpha + (1-\alpha)\cos\dfrac{2\pi x}{N-1} & |x| < \dfrac{N-1}{2} \\ 0 & o/w \end{cases}$$

N = number of samples in windowing function.

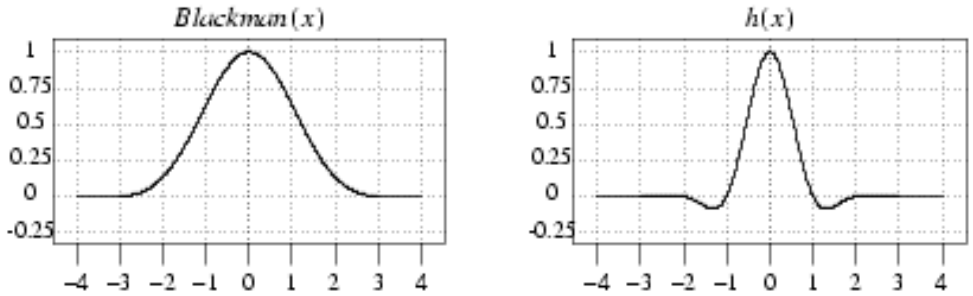
Hann: α = 0.5; Hamming: α = 0.54. Also known as raised cosine window.



|*H(f)*| is sinc+2 shifted sincs. These cancel the right and left side lobes of *Rect(x).*

Blackman Window

$$Blackman(x) = \begin{cases} .42 + .5\cos\frac{2\pi x}{N-1} + .08\cos\frac{4\pi x}{N-1} & |x| < \frac{N-1}{2} \\ 0 & o/w \end{cases}$$



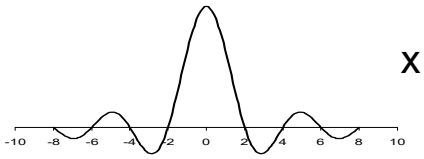
Lanczos Window (1)

$$Lanczos2(x) = \begin{cases} \mathrm{sinc}\left(\frac{\pi x}{2}\right) & 0 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases}$$

h(x) = sinc(x) Lanczos2(x)

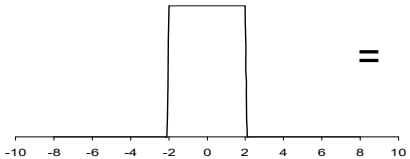
sinc(x) sinc(x/2) Rect(x/4) ← Spatial Domain

window



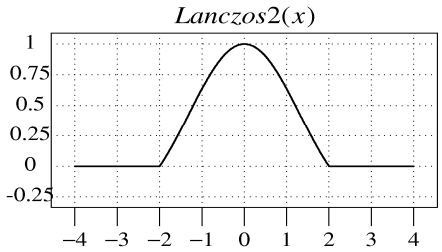
sinc(x/2)

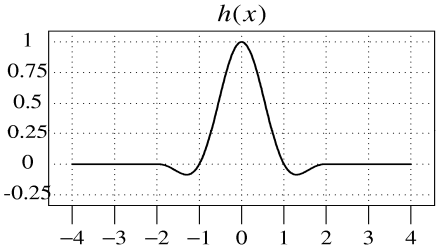
x



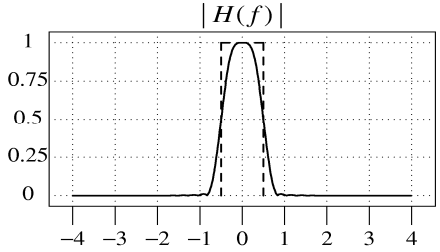
Rect(x/4)

=





h(x)=sinc(x)Lanczos2(x)



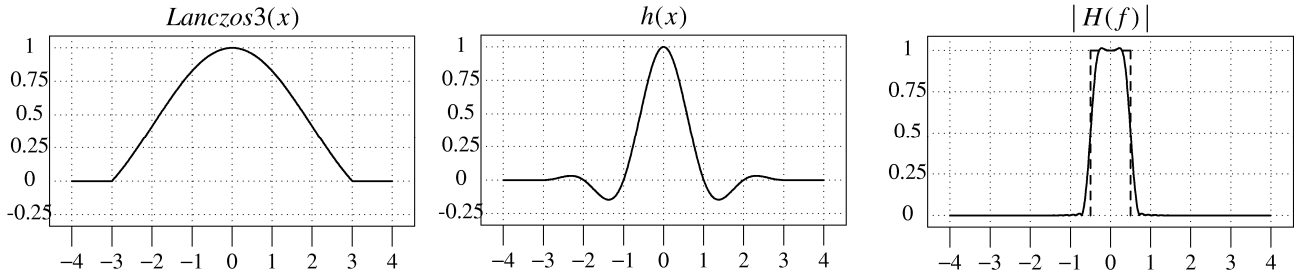
*H(f) = Rect(f) * Rect(2f) * sinc(4f)* ← Frequency Domain

Lanczos Window (2)

- Generalization to N lobes:

$$LanczosN(x) = \begin{cases} \operatorname{sinc}\left(\frac{\pi x}{N}\right) & 0 \le |x| < N \\ 0 & N \le |x| \end{cases}$$

- Let N = 3, this lets 3 lobes pass under the Lanczos window.



- Better passband and stopband response

Comparison of Interpolation Methods

NN, linear, cubic convolution, windowed sinc, sinc

poor ……..> ideal

(blocky, blurred, ringing, no artifacts)

Convolution Implementation

1. Position (center) kernel in input.
2. Evaluate kernel values at positions coinciding with neighbors.
3. Compute products of kernel values and neighbors.
4. Add products; init output pixel.

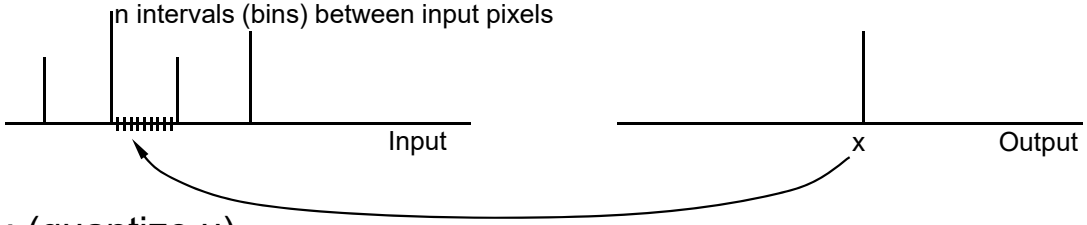
Step (1) can be simplified by incremental computation for space-invariant warps. (newpos = oldpos + inc).

Step (2) can be simplified by LUT.

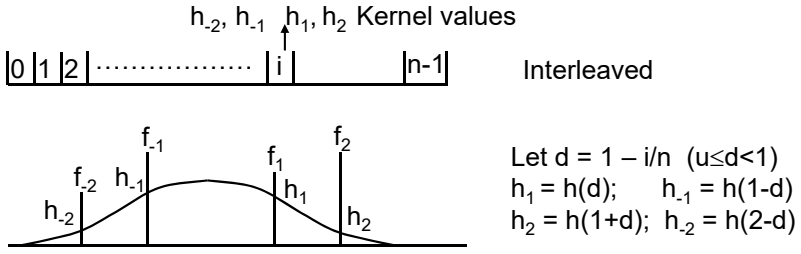
Interpolation with Coefficient Bins

Implementation #1: Interp. with Coefficient Bins (for space-invariant warps)

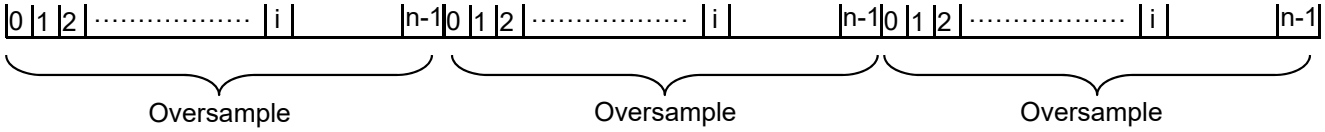
- **Strategy:** accelerate resampling by precomputing the input weights and storing them in LUT for fast access during convolution.



$u \rightarrow$ bin: (quantize u)

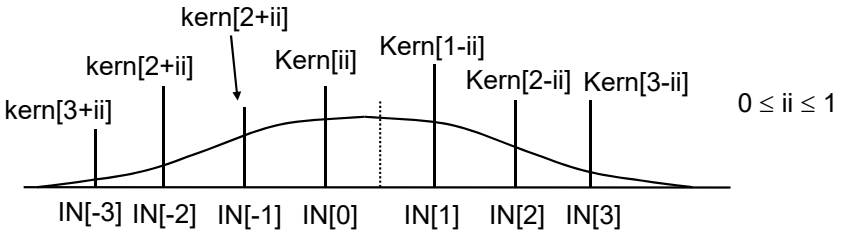


Uninterleaved Coefficient Bins




```
ii = bin #
0 ≤ ii < oversample
val =           IN[-2] * kern[2 * oversample + ii]
        +       IN[-1] * kern[1 * oversample + ii]
        +       IN[ 0] * kern[ii]
        +       IN[ 1] * kern[1 * oversample - ii]
        +       IN[ 2] * kern[2 * oversample - ii]
        +       IN[ 3] * kern[3 * oversample - ii];
if(ii == 0)val +=IN[-3] * kern[3 * oversample - ii];
```

Refer to code on p. 151

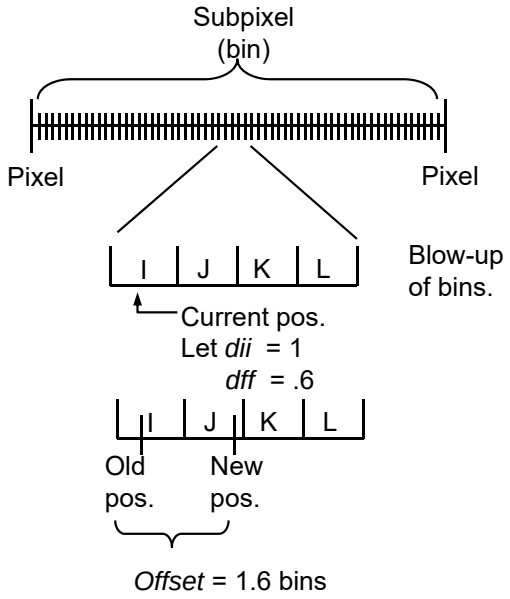


Kernel Position

- Since we are assuming space invariance, the new position for the kernel = oldpos + offset.

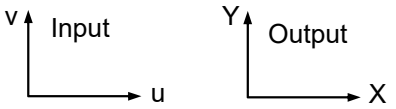
$$offset = dii + dff\,; \quad dii = \#\,\text{whole bins}\left(\frac{INlen * oversample}{OUTlen}\right)$$

$$dff = partial\ \text{bin}\ (INlen * oversample)\%OUTlen$$



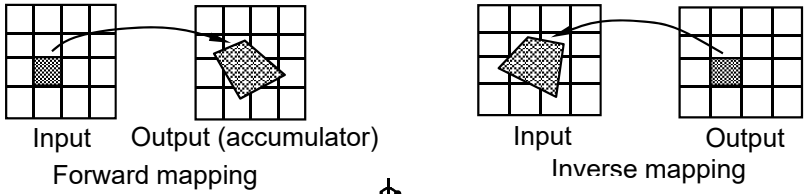
Offset must be accurate to avoid accrual of error in the incremental repositioning of the kernel.

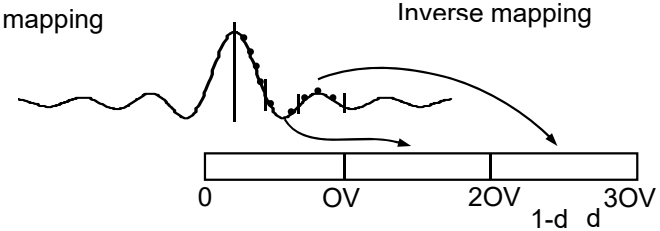
Forward vs. Inverse Mapping



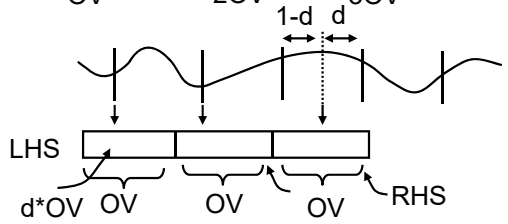
- Forward mapping: $x = X(u, v);\ y = Y(u, v)$
- Inverse mapping: $u = U(x,y);\ v = V(x, y)$

Ch. 3, Sec. 1



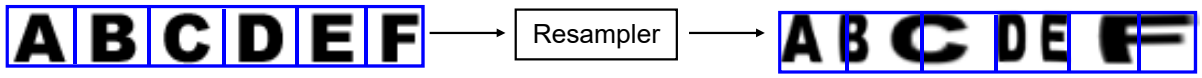


Coefficient Bins for kernel eval for fast Convolution for image reconstruction.



Fant's Algorithm

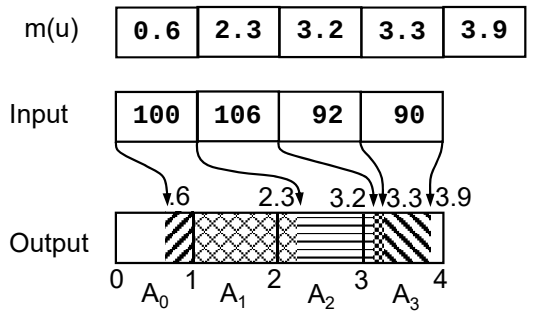
Implementation #2: Fant's Resampling Algorithm (for space-var. warps)

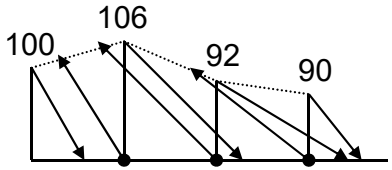


Input and output are streams of pixels that are consumed and generated at rate determined by the spatial mapping.
Three conditions per stream:
1) Current input pixel is entirely consumed without completing an output pixel.
2) The input is entirely consumed while completing the output pixel.
3) Output pixel computed without entirely consuming the current input pixel.

Algorithm uses linear interpolation for image reconstruction and box filtering (unweighted averaging) for antialiasing. Code on p.156.

Example





$$A_0 = (100)(.4) = 40$$

$$A_1 = \left[(100)\left(1-\frac{.4}{1.7}\right)+(106)\left(\frac{.4}{1.7}\right)\right](1) = 101$$

$$A_2 = \left[(100)\left(1-\frac{1.4}{1.7}\right)+(106)\left(\frac{1.4}{1.7}\right)\right](.3)+(106)(.7) = 106$$

$$A_3 = \left[(106)\left(1-\frac{.7}{.9}\right)+(92)\left(\frac{.7}{.9}\right)\right](.2)+(92)(.1)+(90)(.6) = 82$$