

Image Resampling

Arthur Goshtasby

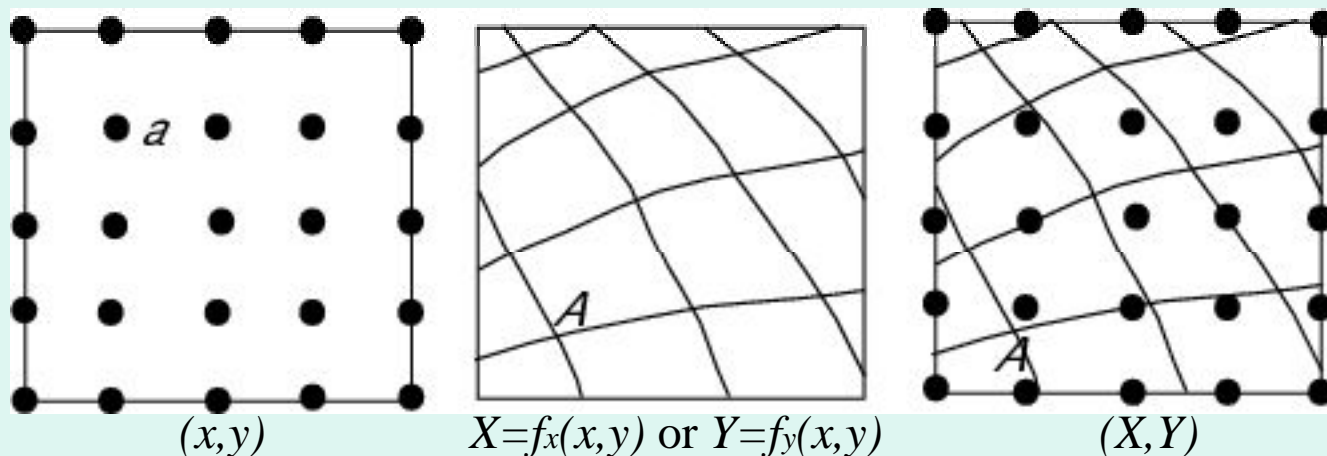
Wright State University

Image Fusion Systems Research

Problem description

Problem: Knowing image intensities at discrete coordinates, determine intensities at arbitrary coordinates.

Approach: Fit a surface to intensities at discrete coordinates and estimate the surface value at desired coordinates.



Resampling methods

- Nearest neighbor
- Bilinear interpolation
- Cubic convolution
- Cubic spline interpolation
- Radially symmetric kernels

Nearest-neighbor resampling

- Set the intensity at (X, Y) to the intensity of the pixel closest to it: $[\text{round}(X), \text{round}(Y)]$.
- This method is very fast, but it produces aliasing effects along edges.

Example:



Original



Resampled

Bilinear interpolation

- Assuming u and v are integer parts of X and Y , respectively, bilinear interpolation is defined by

$$I(X, Y) = W_{u,v}I(u, v) + W_{u+1,v}I(u+1, v) \\ + W_{u,v+1}I(u, v+1) + W_{u+1,v+1}I(u+1, v+1)$$

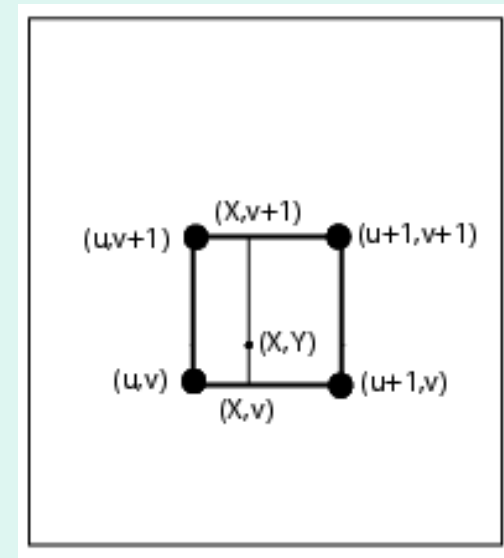
where

$$W_{u,v} = (u+1-X)(v+1-Y)$$

$$W_{u+1,v} = (X-u)(v+1-Y)$$

$$W_{u,v+1} = (u+1-X)(Y-v)$$

$$W_{u+1,v+1} = (X-u)(Y-v)$$



Bilinear example



Original



Resampled

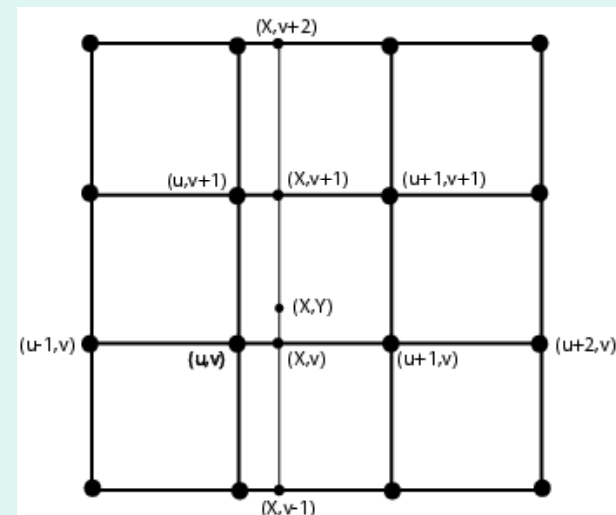
Cubic convolution

- Cubic convolution can be computed row-by-row and then column-by-column.
- Assuming intensities at $u-1$, u , $u+1$, $u+2$ are $I(u-1)$, $I(u)$, $I(u+1)$, $I(u+2)$, intensity at X is estimated from $f(X) = I(u-1)f_{-1} + I(u)f_0 + I(u+1)f_1 + I(u+2)f_2$

where

$$\begin{aligned} f_{-1} &= -\frac{1}{2}t^3 + t^2 - \frac{1}{2}t, \\ f_0 &= \frac{3}{2}t^3 - \frac{5}{2}t^2 + 1, \\ f_1 &= -\frac{3}{2}t^3 + 2t^2 + \frac{1}{2}t, \\ f_2 &= \frac{1}{2}t^3 - \frac{1}{2}t^2, \end{aligned}$$

and $t = X - u$.



Cubic convolution example



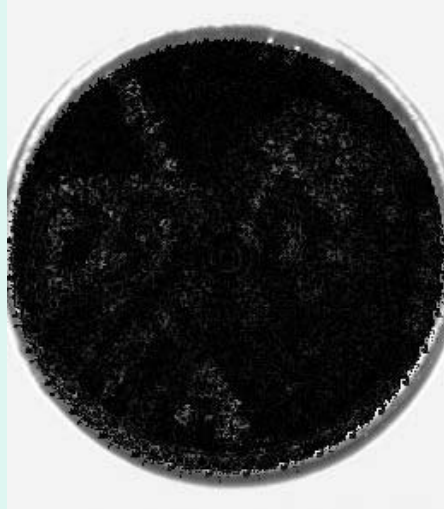
Original



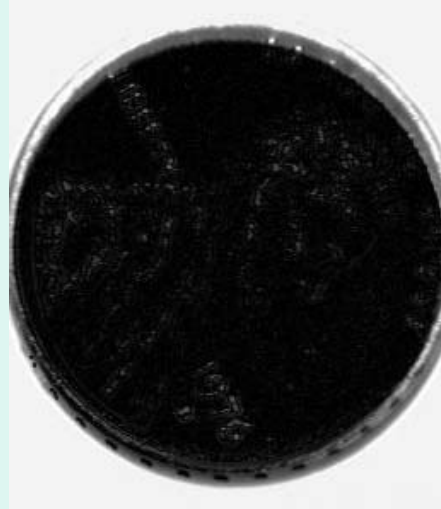
Resampled

Comparison

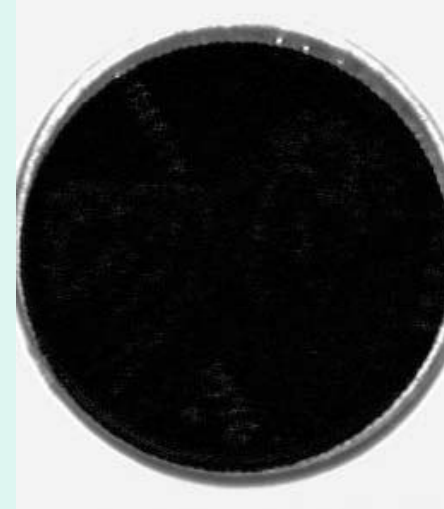
- Suppose an image is rotated with a 10-degree increment 36 times and then subtracted from the original image.
- Only nearest-neighbor preserves original image intensities. Bilinear and cubic convolution change image intensities.



Nearest-neighbor



Bilinear



Cubic convolution

Cubic spline interpolation

- In cubic spline also, computation is carried out row-by-row and then column-by-column.
- Assuming intensities at $i = -1, 0, 1, 2$ are I_{-1}, I_0, I_1, I_2 , the cubic B-spline curve approximating the intensities in the range $0 < u < 1$ is

$$f(u) = \sum_{i=-1}^2 I_i b_i(u)$$

where

$$\begin{aligned} b_{-1}(u) &= (-u^3 + 3u^2 - 3u + 1)/6, \\ b_0(u) &= (3u^3 - 6u^2 + 4)/6, \\ b_1(u) &= (-3u^3 + 3u^2 + 3u + 1)/6, \\ b_2(u) &= u^3/6, \end{aligned}$$

- For the curve to interpolate the intensities, it is required to determine new intensities I'_j s using I_i s.

$$I_i = \sum_{j=-1}^2 I'_j b_j(u_i) \quad i=-1, \dots, 2$$

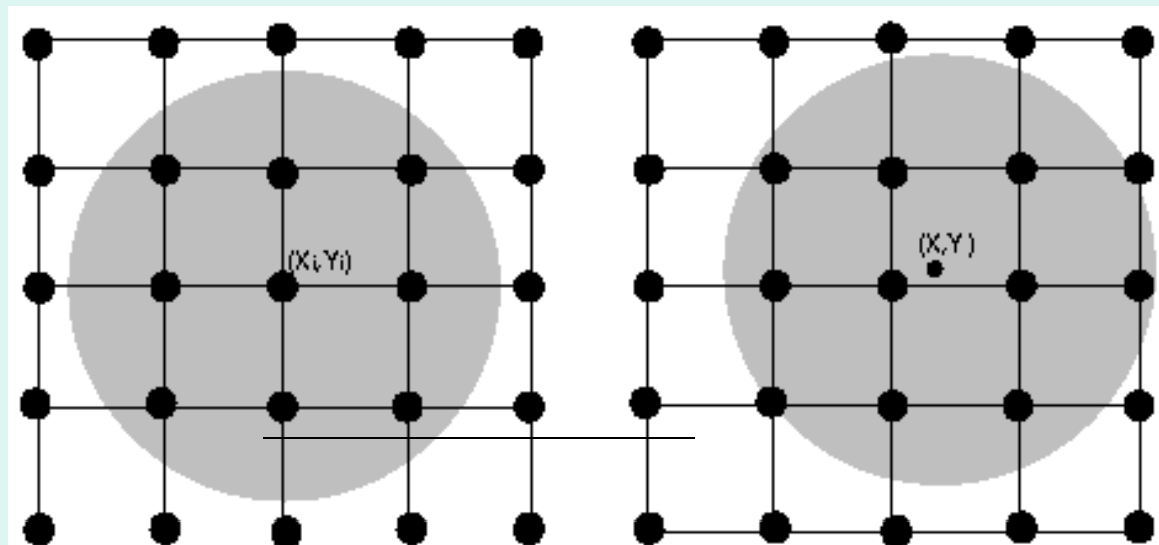
- Instead of solving a very large system of equations, inverse filtering may be used to find the I'_j s.

Radially symmetric kernels

- Cubic convolution and cubic spline methods are not rotationally invariant. To obtain a rotationally invariant resampling, radially symmetric kernels are needed.
- An example of a radially symmetric kernel:

$$f(r_i) = \begin{cases} 1 - 3r_i^2 + 2r_i^3, & 0 \leq r_i \leq 1, \\ 0, & r_i > 1, \end{cases}$$

- The influence of a pixel on interpolating function is radially symmetric.
- The resampled value at a point is obtained from the values of pixels within a circular area centered at the point.



Computational complexity

Type of Resampling	Computational Complexity
Nearest-Neighbor	$O(n^2)$
Bilinear Interpolation	$O(n^2)$
Cubic Convolution	$O(n^2)$
Cubic Spline, Direct Computation	$O(n^4)$
Cubic Spline, Using FFT	$O(n^3 \log n)$
Radial Functions with Local Support	$O(n^4)$
Gaussian, Using FFT	$O(n^3 \log n)$

For an $n \times n$ image.

Resampling references

1. E. G. Keys, Cubic convolution interpolation for digital image processing, *IEEE Trans. Acoustics, Speech, and Signal Processing*, **29**(6):1153–1160 (1981).
2. H. S. Hou and H. C. Andrews, Cubic splines for image interpolation and digital filtering, *IEEE Trans. Acoustics, Speech, and Signal Processing*, **26**(6):508–517 (1978).
3. L. A. Ferrari, P. V. Sankar, J. Sklansky, and S. Leeman, Efficient two-dimensional filters using B-spline functions, *Computer Vision, Graphics, and Image Processing*, **35**:152–169 (1986).
4. F. Cheng and A. Goshtasby, A parallel B-spline surface fitting algorithm, *ACM Trans. Graphics*, **8**(1):41–50 (1989).
5. A. Goshtasby, F. Cheng, and B. A. Barksy, B-spline curves and surface viewed as digital filters, *Computer Vision, Graphics, and Image Processing*, **52**:264–275 (1990).