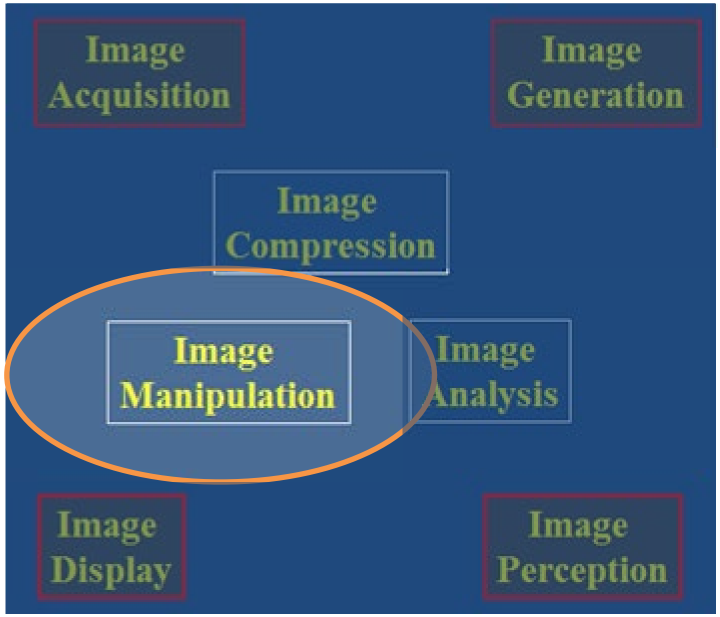
Warps, Filters, and Morph Interpolation



Brent M. Dingle, Ph.D. 2015
Game Design and Development Program
Mathematics, Statistics and Computer Science
University of Wisconsin - Stout

Material in this presentation is largely based on/derived from slides originally by Szeliski, Seitz and Efros

Lecture Objectives

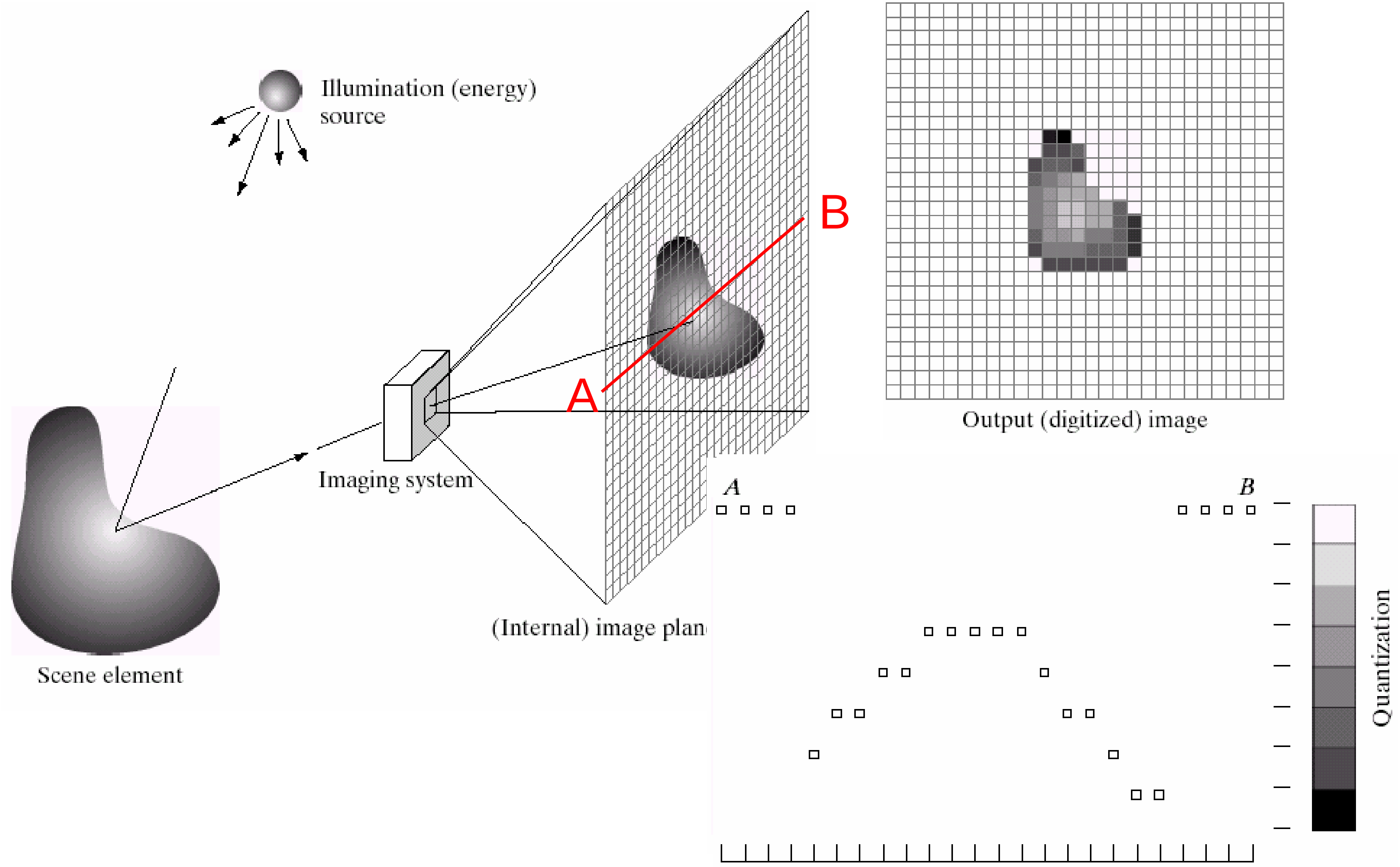
- Previously
  - Colors
  - Filtering
  - Interpolation
  - Warps

- Today
  - Review
  - Morphs

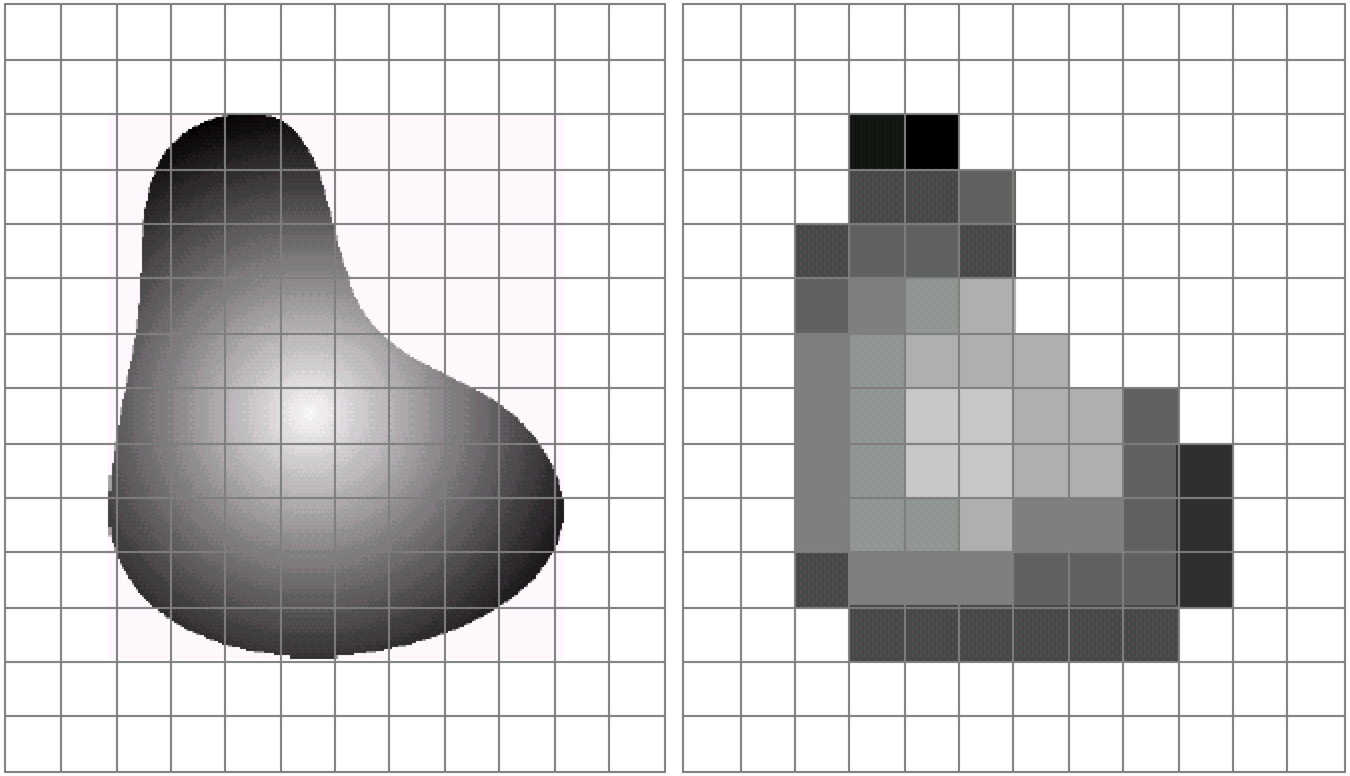
Outline

- Review: Image Fundamentals
- Review: Warping, Filtering, Interpolation
- Image Morphing

Image Formation

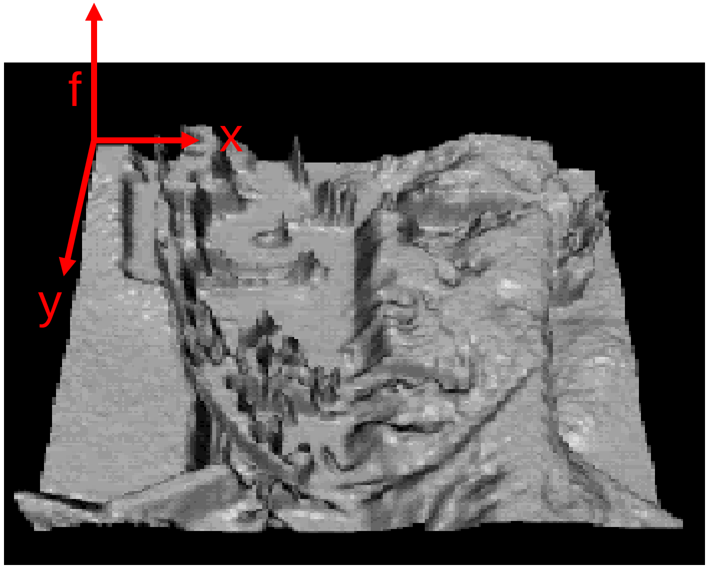


Sampling and Quantization



What is an Image?

- 2D Function or Array
  - f(x, y) gives the pixel intensity at position (x, y)
  - defined over a rectangle, with finite range
    - f: [a, b] x [c, d] → [0, 1]

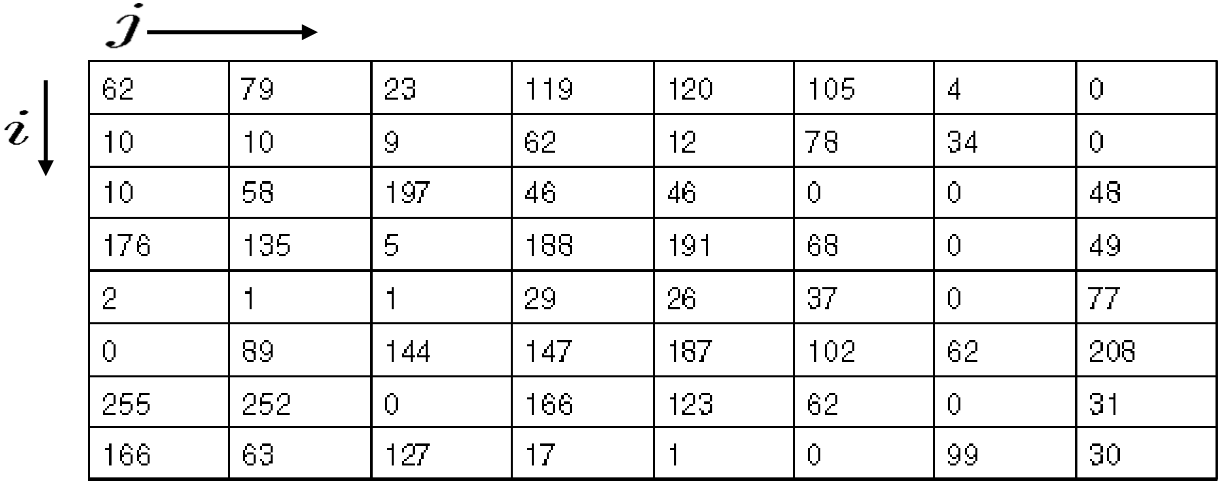


For color image f is a vector function:

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

Digital Image

- Digital Images
  - Sample the 2D space on a regular grid
  - Quantize each sample (to nearest integer)
  - Assume samples are D distance apart
    - f[i, j] = quantize(f(iD, jD))
  - Image is now a matrix of integer values:



62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Images may have Aliasing Artifacts

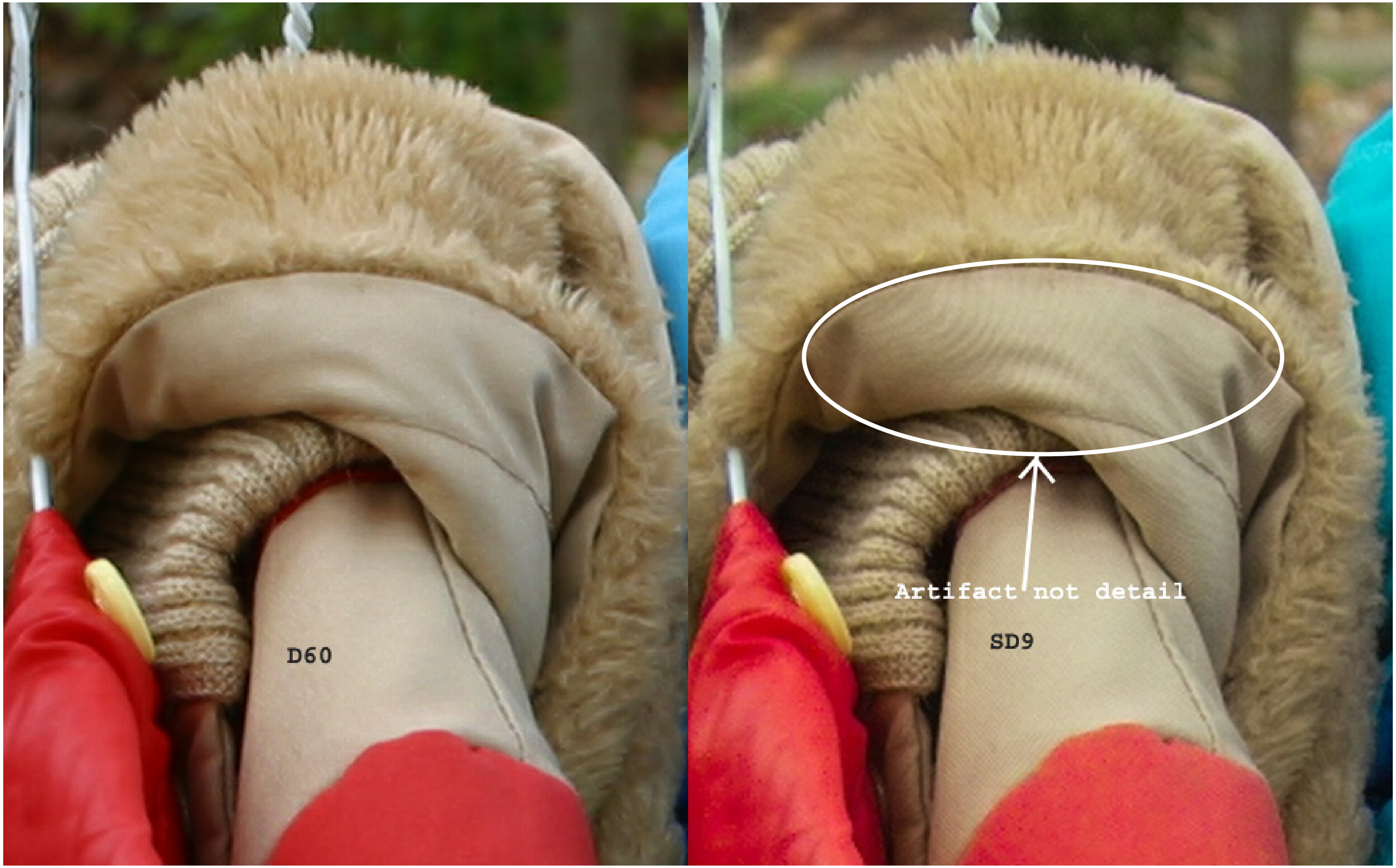
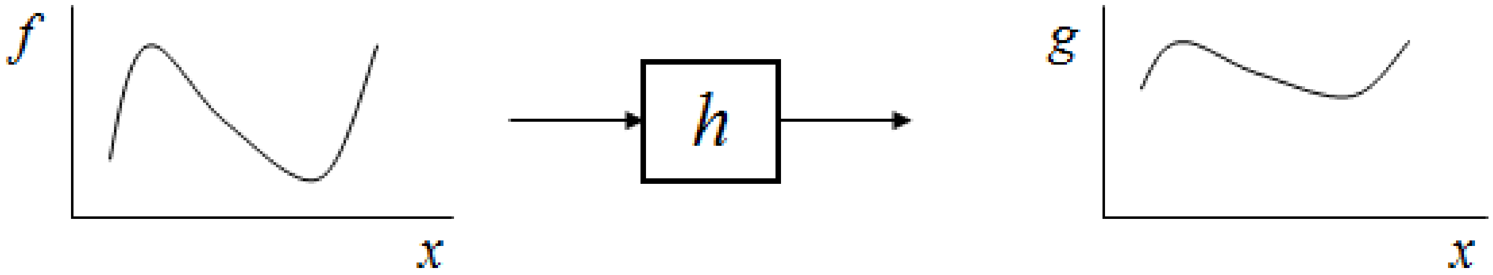
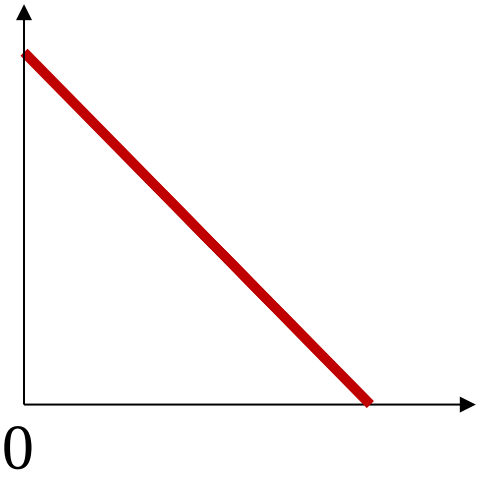


Image Manipulation

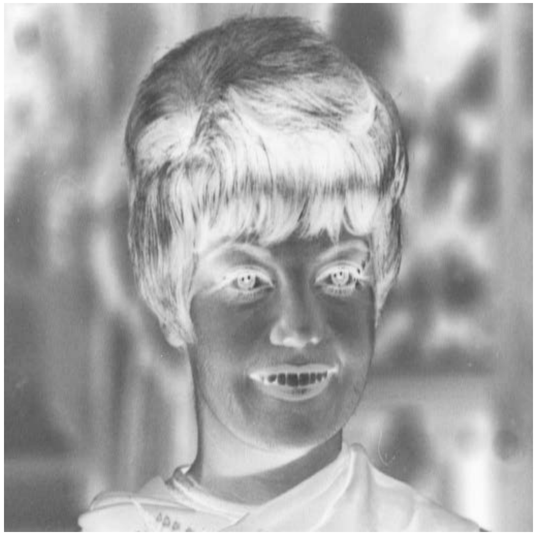


$$g(x) = h(f(x))$$

Point Processing Example

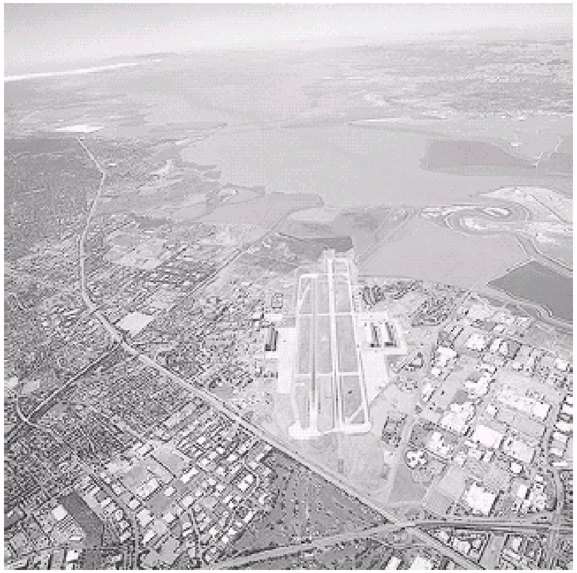


h(a) = 1 – a



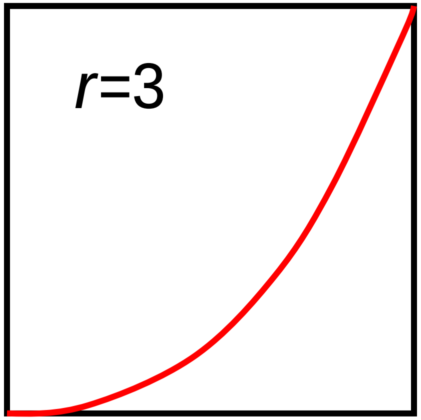
Negative

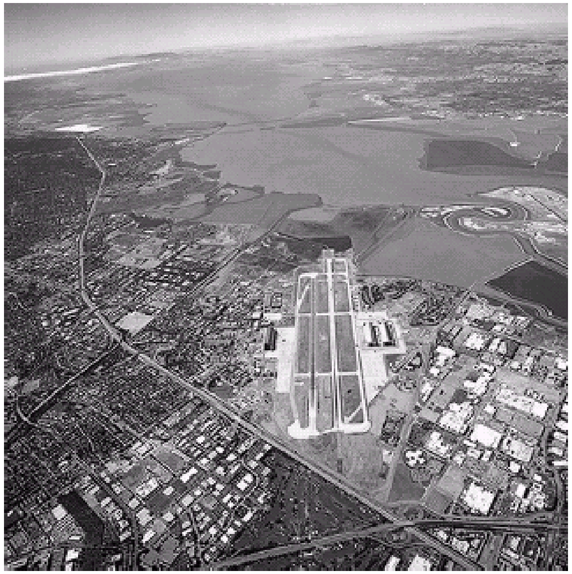
Image Enhancement Example

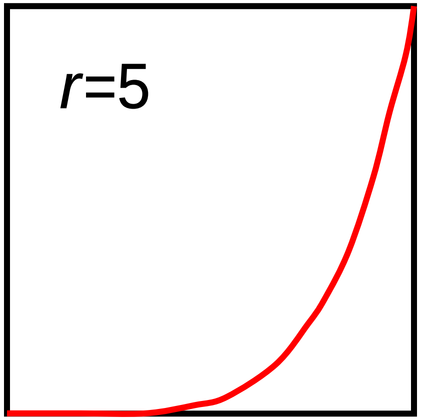


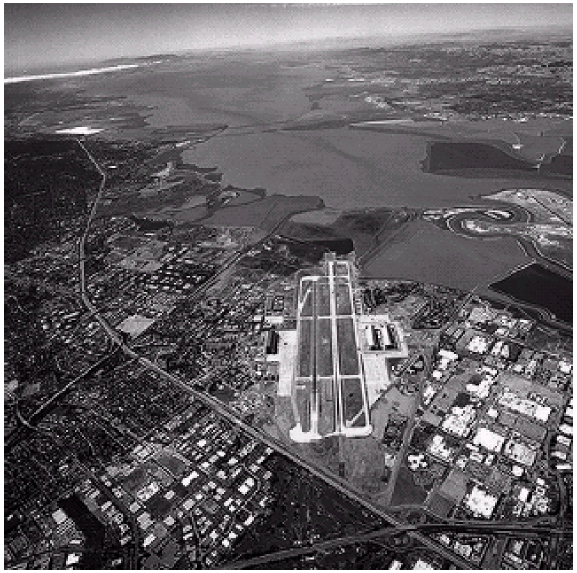
$h(a) = a^r$

gamma correction

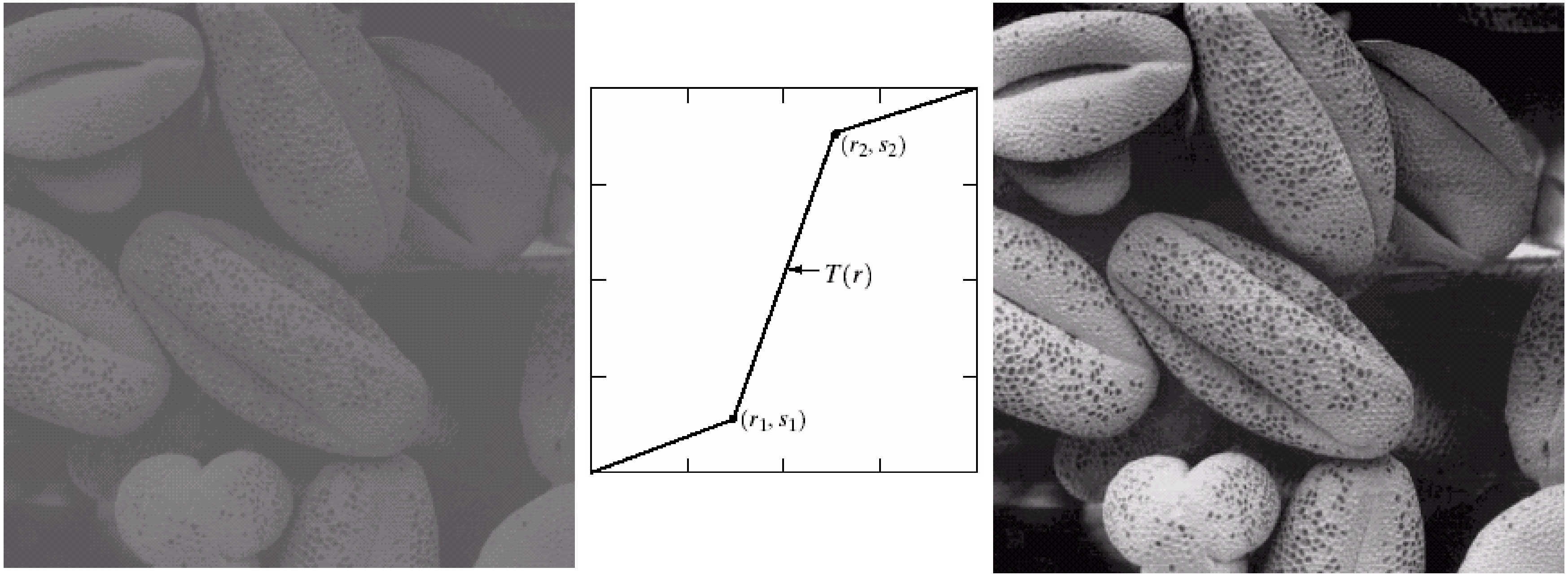




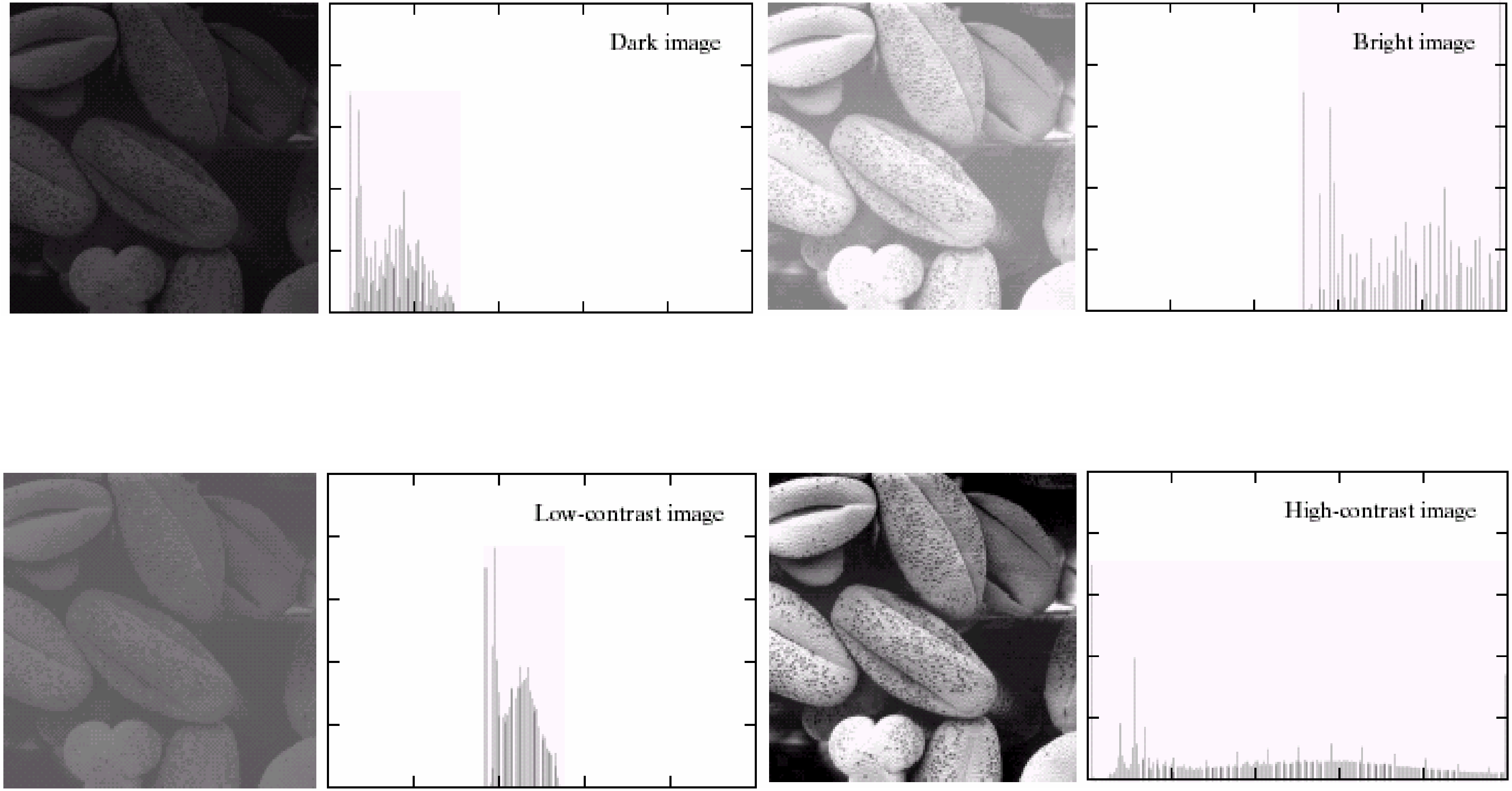




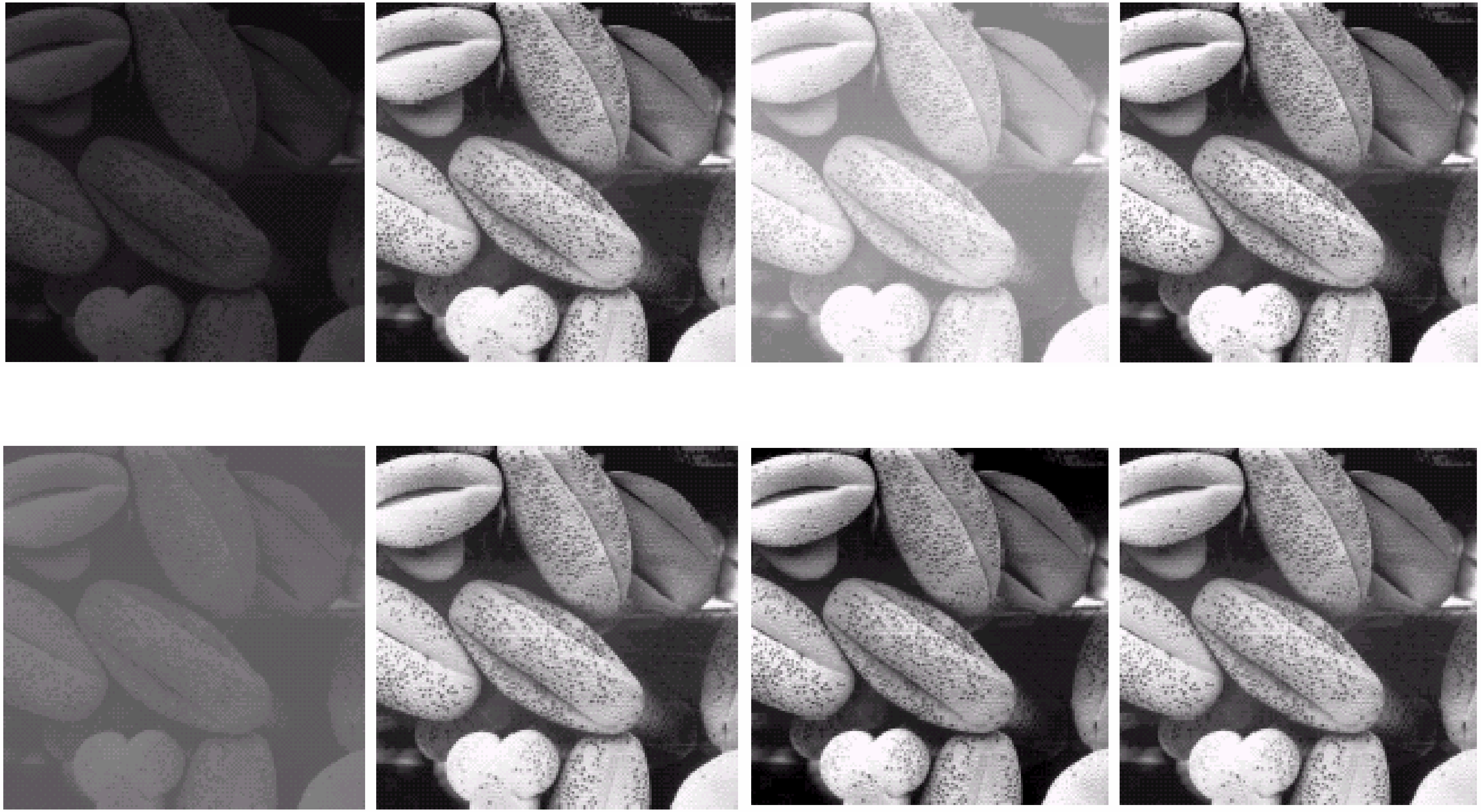
Contrast Stretching Example

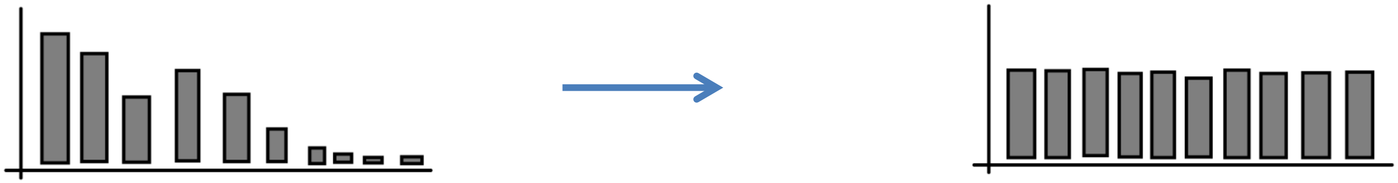


Histogram Analysis



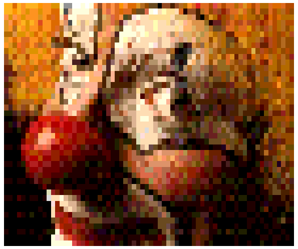
Histogram Equalization

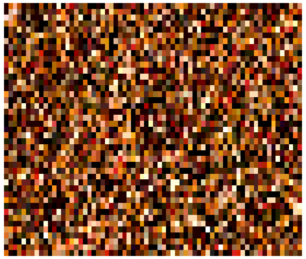




Filtering (neighborhood processing)

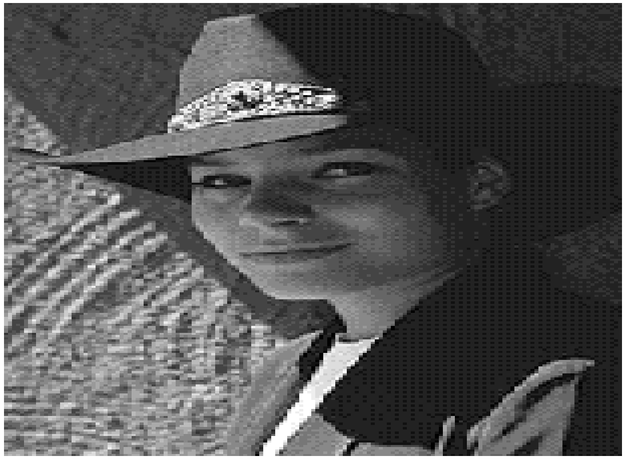
- ‘Global-ness’ of histogram may be too big
  - Example: Mix-up all the pixels



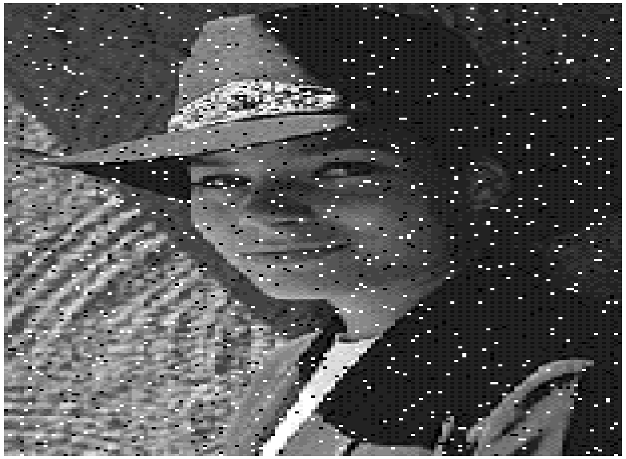


  - Histogram stays same
  - May need more locally spatial information for some operations to work as desired
    - such as noise removal

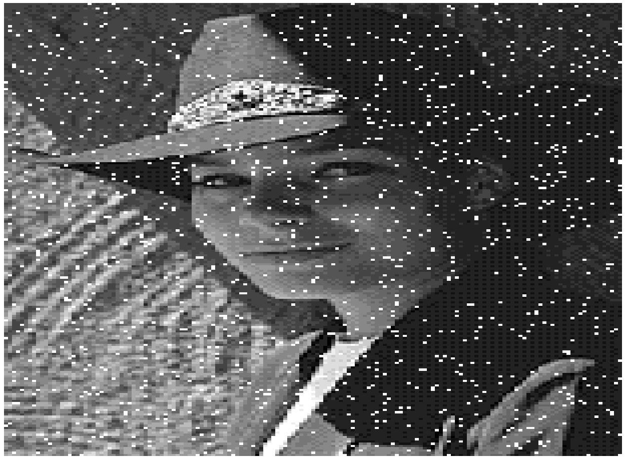
Noise Examples



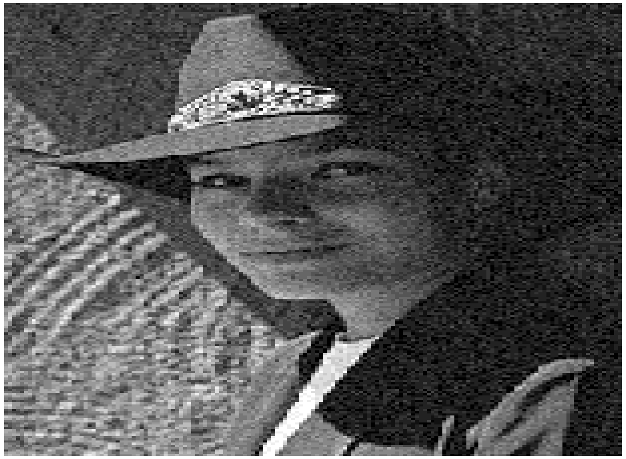
Original



Salt and pepper noise



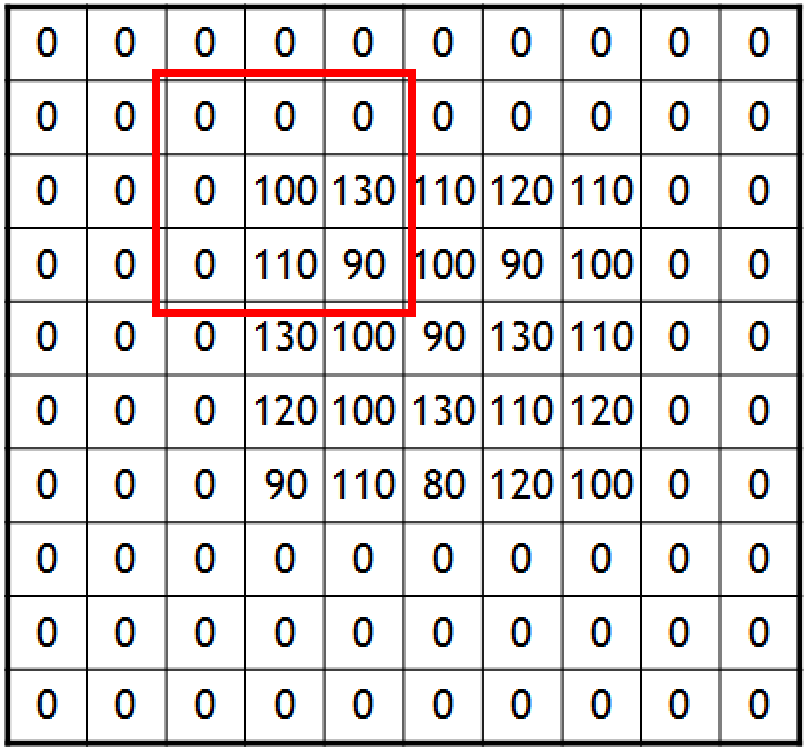
Impulse noise



Gaussian noise

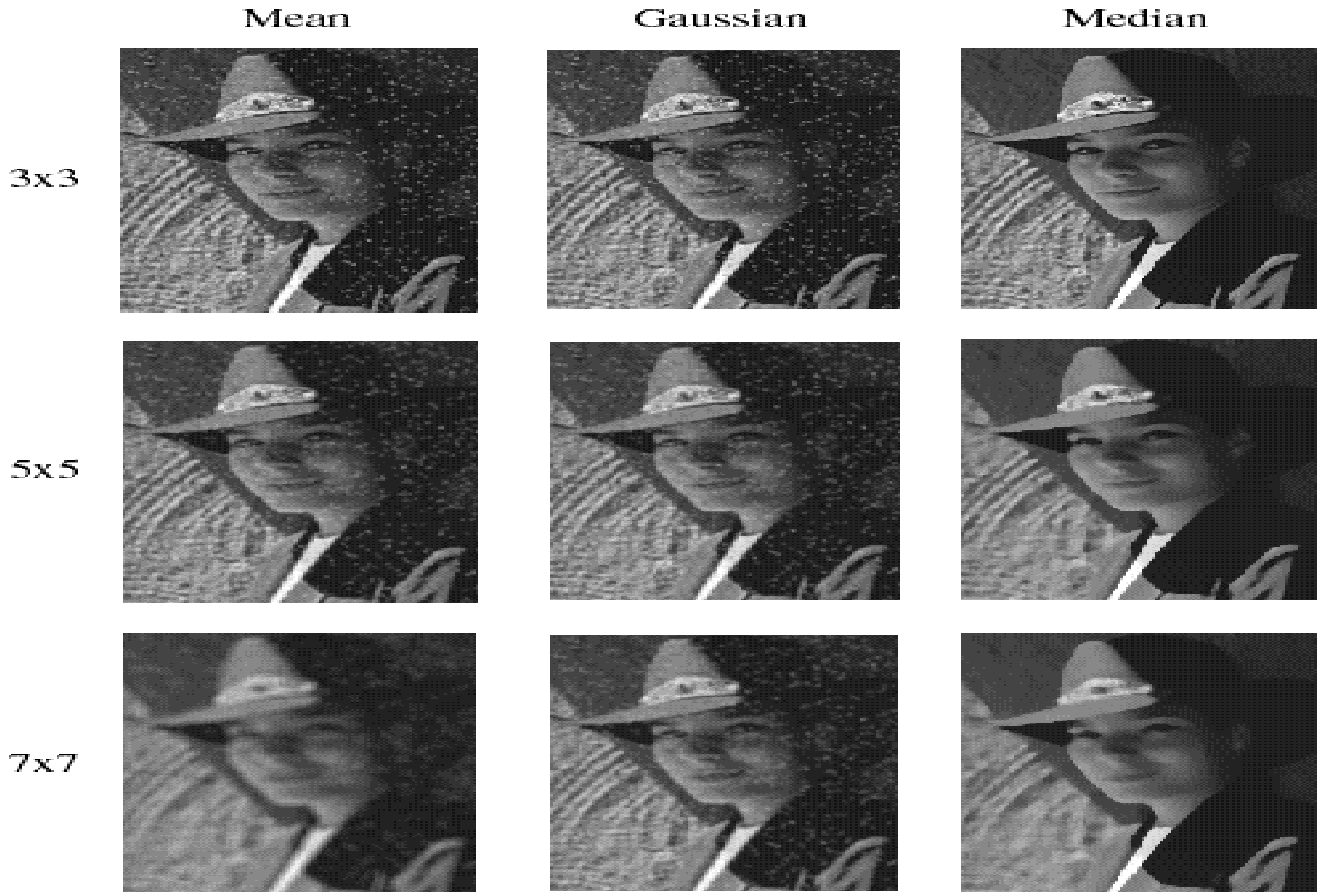
Noise Reduction

- Mean, Median, and Gaussian filters
  - – local neighborhoods



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	100	130	110	120	110	0	0
0	0	0	110	90	100	90	100	0	0
0	0	0	130	100	90	130	110	0	0
0	0	0	120	100	130	110	120	0	0
0	0	0	90	110	80	120	100	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Comparison: Salt and Pepper Noise



Outline

- Review: Image Fundamentals
- Review: Warping, Filtering, Interpolation
- Image Morphing

Filtering and Warping: Review

- Filtering
  - modify an image based on image color content without any intentional change in image geometry
    - resulting image essentially has the same size and shape as the original
- Warping
  - modification of an image that operates on the image’s geometric structure
- Review follows…

Image Filtering versus Warping

image Filtering: changes range of image

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

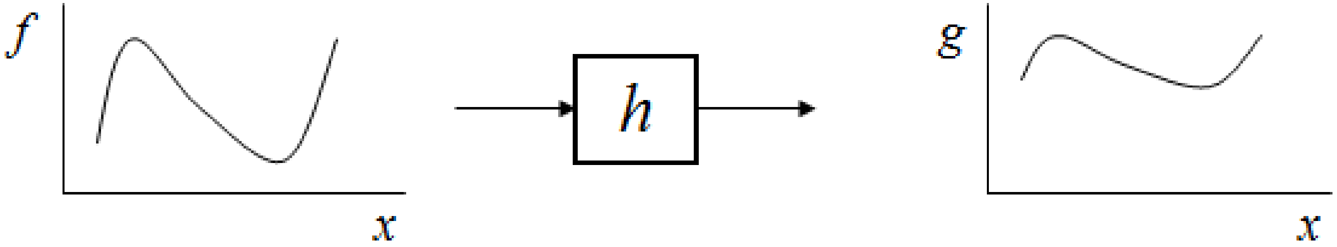


image Warping: changes domain of image

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

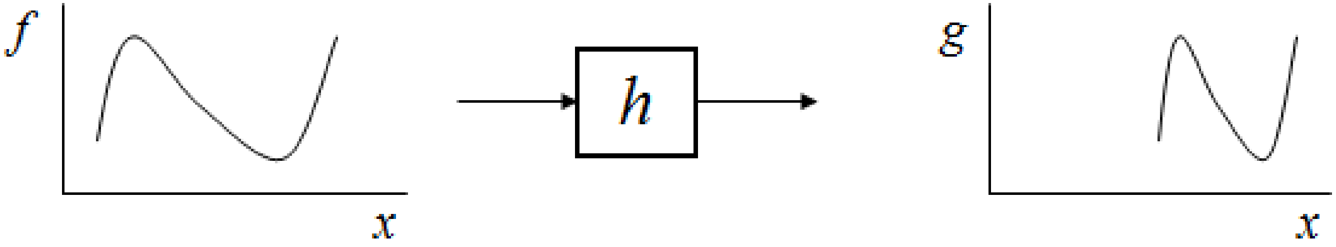


Image Filtering versus Warping

image Filtering: changes range of image

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

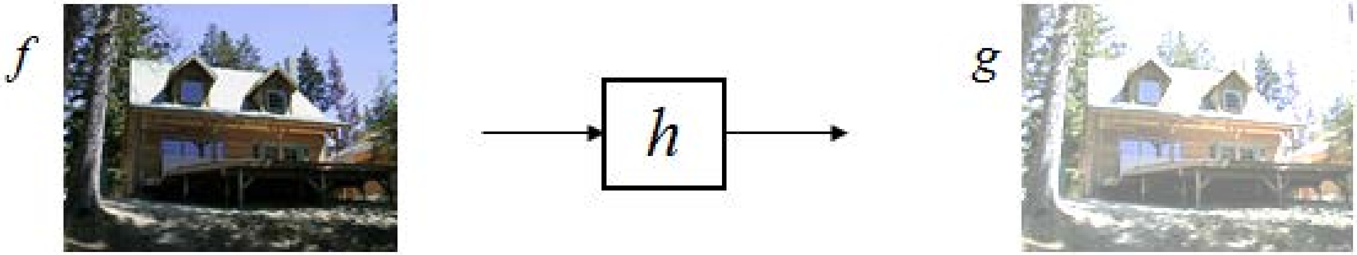
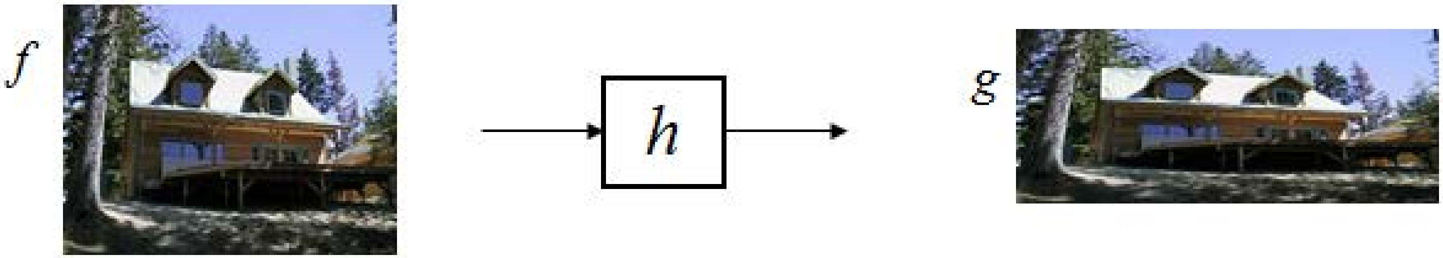


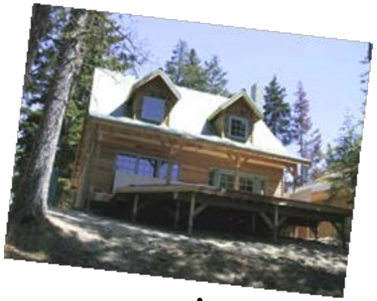
image Warping: changes domain of image

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

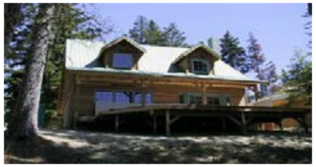


Parametric (global) warping

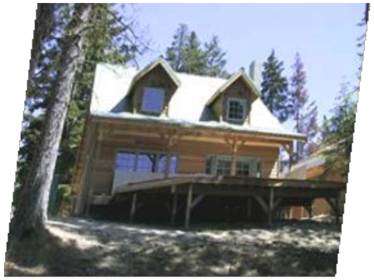
translation



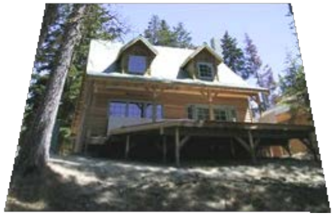
rotation



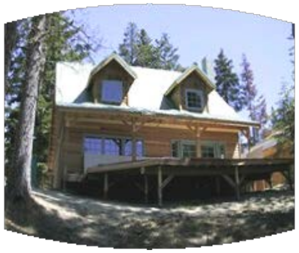
aspect



affine



perspective



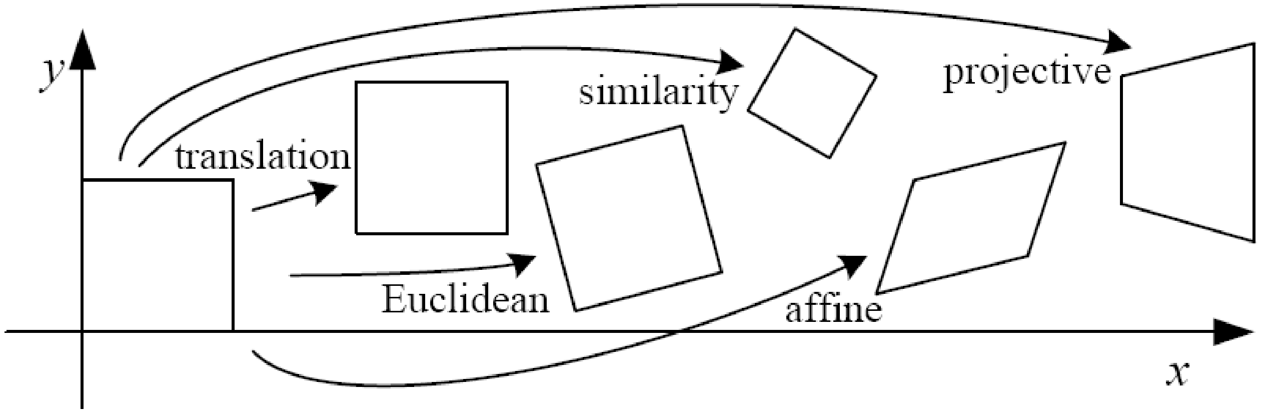
cylindrical

2D coordinate transforms

- translation: $x' = x + t$ $\quad x = (x,y)$
- rotation: $x' = R\,x + t$
- similarity: $x' = s\,R\,x + t$
- affine: $x' = A\,x + t$
- perspective: $x' \cong H\,x$ $\quad x = (x,y,1)$

  (x is a homogeneous coordinate)

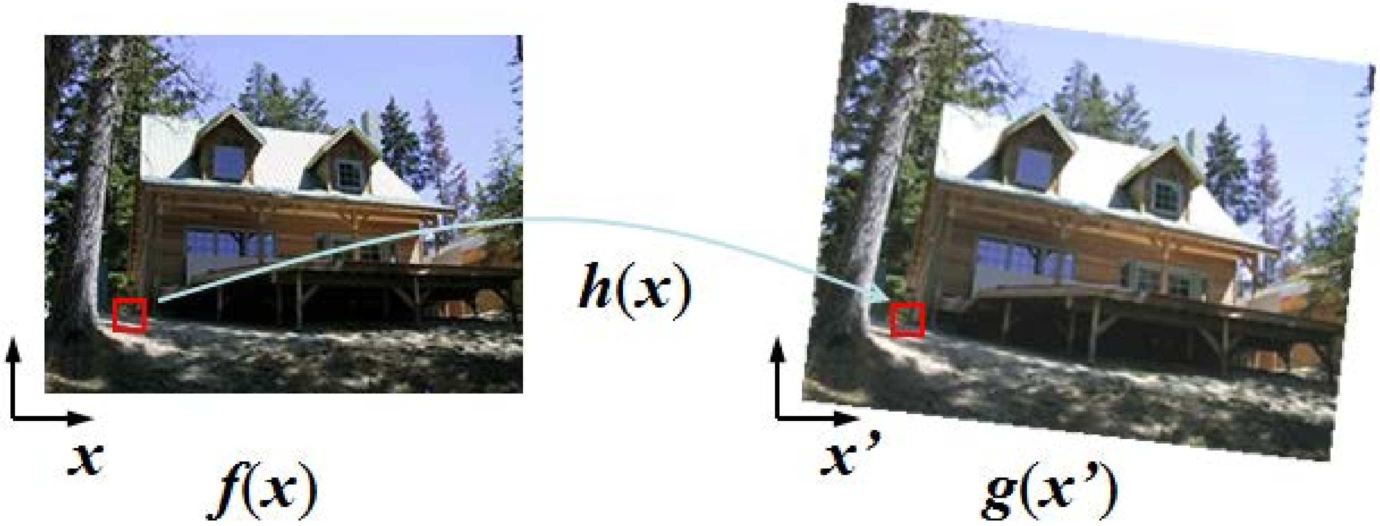
2D image transforms



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

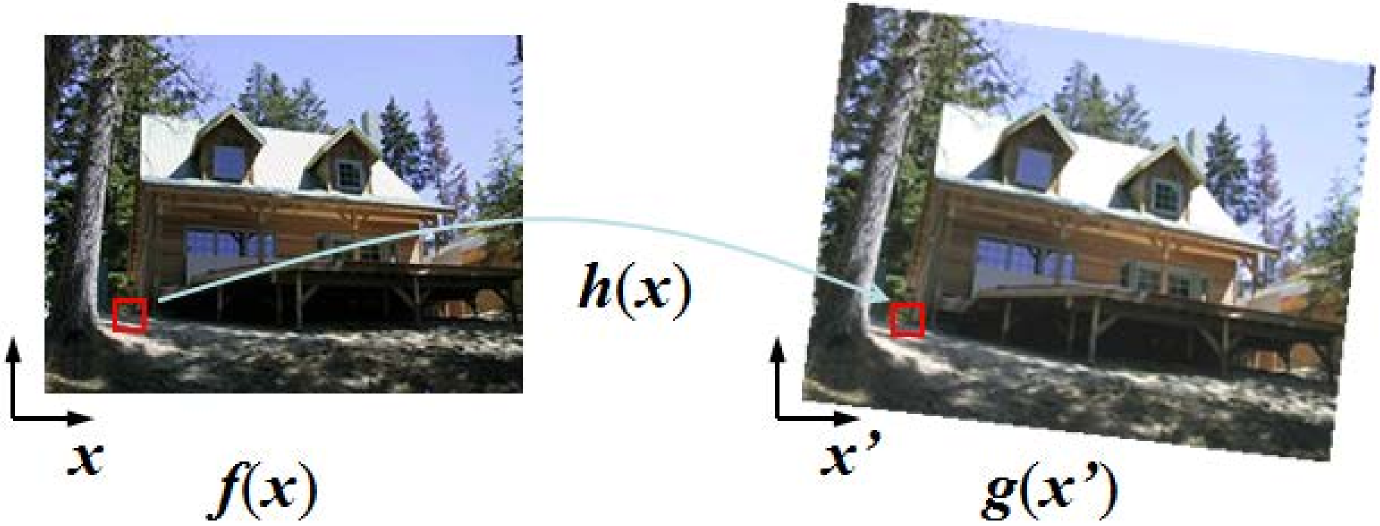
Image Warping

- Given a coordinate transform x’ = h(x) and a source image f(x), how do we compute a transformed image g(x’) = f(h(x))?



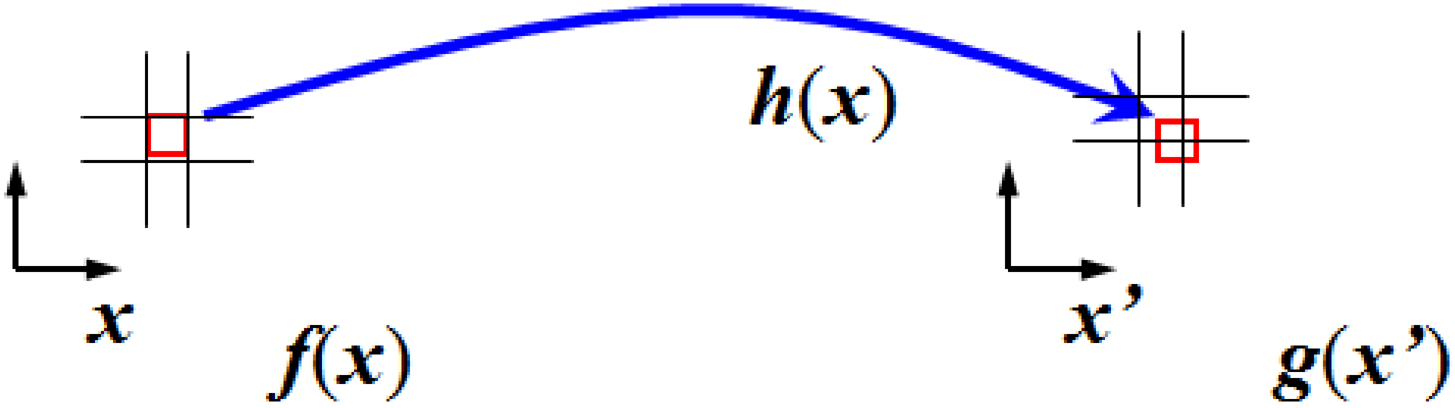
Forward Mapping

- Send each pixel f(x) to its corresponding location x’ = h(x) in g(x’)
  - What if pixel lands “between” two pixels?



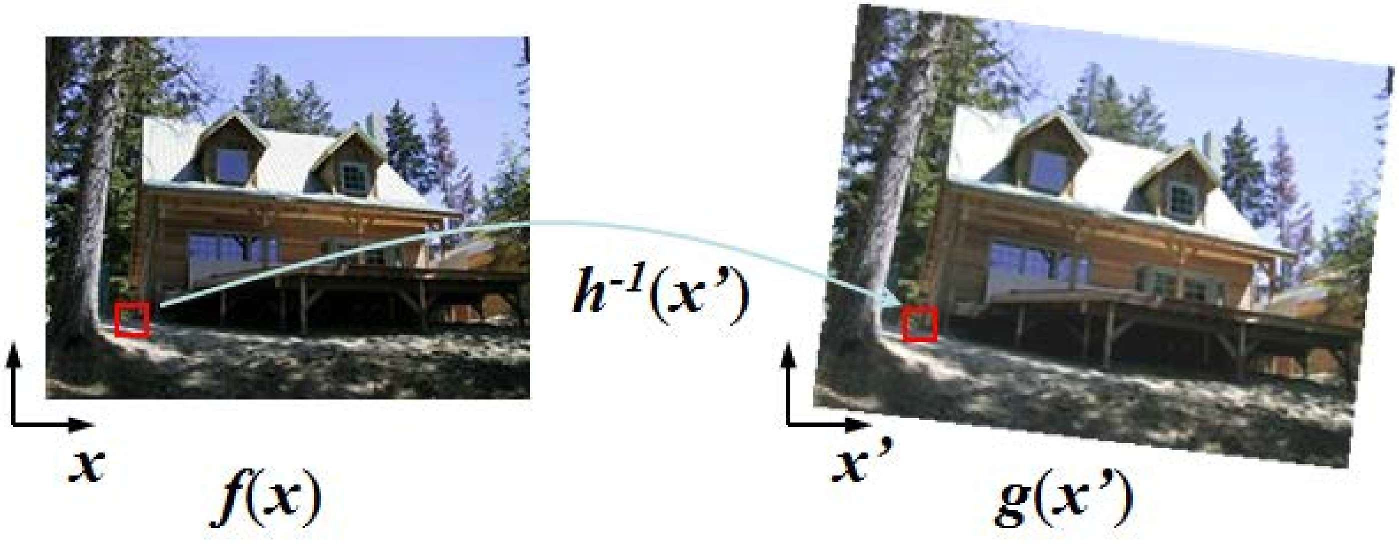
Forward Mapping

- 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, normalize later (splatting)



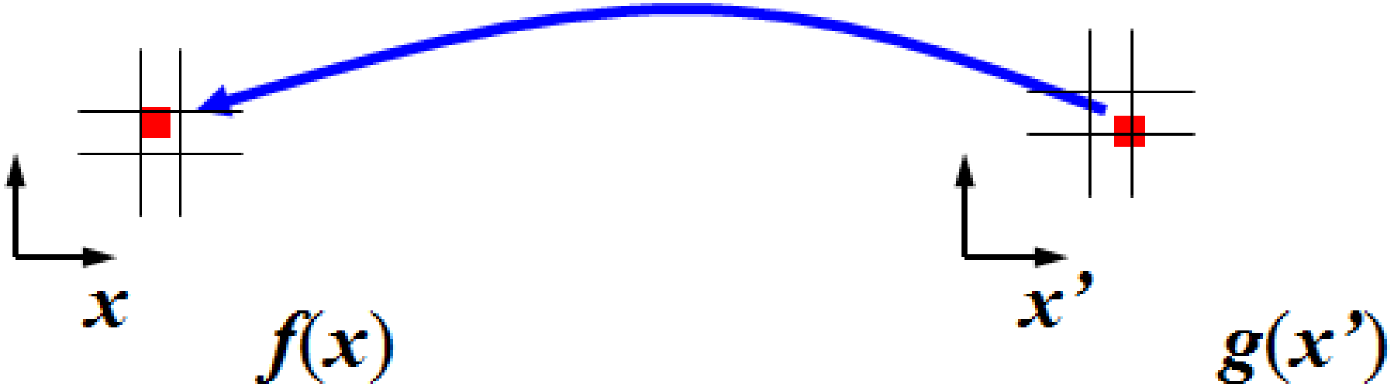
Inverse Mapping

- Get each pixel g(x’) from its corresponding location x = h^{-1}(x’) in f(x)
  - What if pixel comes from “between” two pixels?



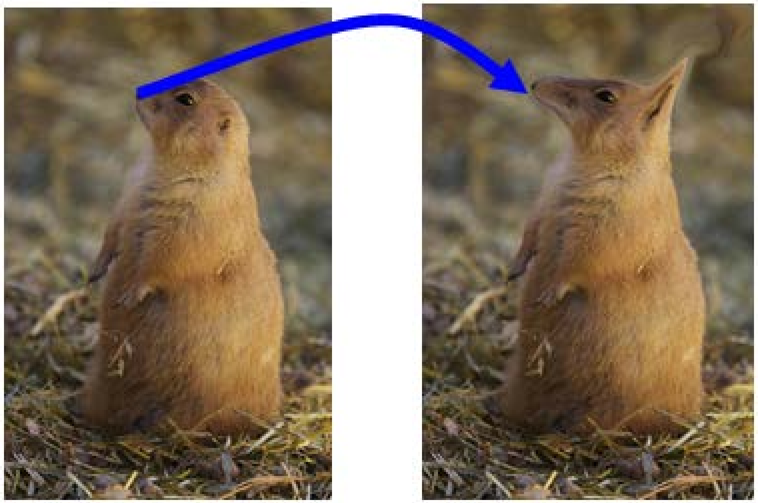
Inverse Mapping

- Get each pixel g(x') from its corresponding location $x = h^{-1}(x')$ in f(x)
  - What if pixel comes from "between" two pixels?
  - Answer: resample color value from interpolated (pre-filtered) source image

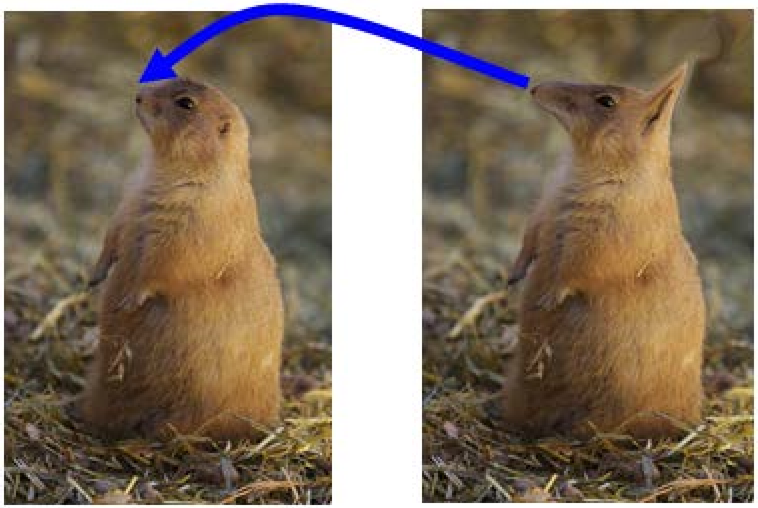


Forward versus Inverse

- Forward Map
  - Potential Gap Problems

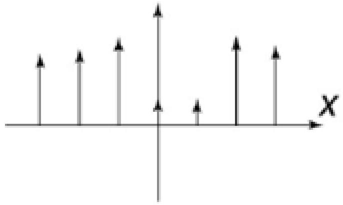


- Inverse Map
  - Most useful
  - For each output pixel
    - Lookup at inverse warp location in input image

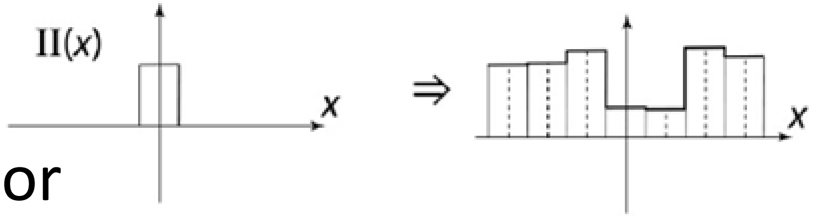


Interpolation

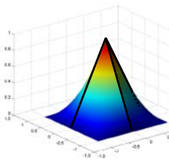
- Given



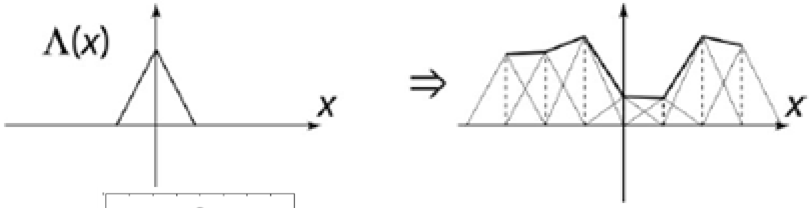
- Methods
  - nearest neighbor



Nearest-neighbor interpolation

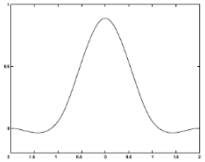


  - bilinear



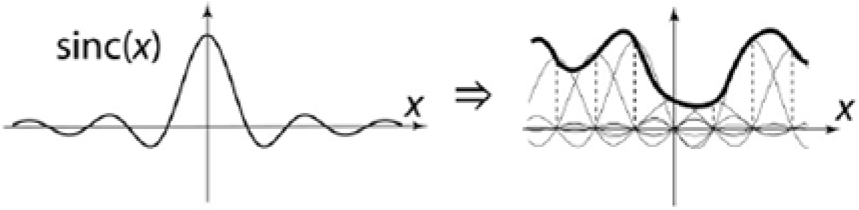
Linear interpolation

  - bicubic



Cubic interpolation

  - sinc



“Ideal” reconstruction

Nearest Neighbor

- Given pixel (i’ , j’) in the destination image
- Find the corresponding pixel in the source image (i, j)

EXAMPLE:

Assume source image has width = w, and height = h
Assume destination image has width = w’ and height = h’

Then a point in the destination is given by i’ = i * w’ / w

j’ = j * h’ / h

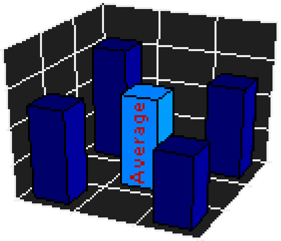
integer division so decimal portion is dropped

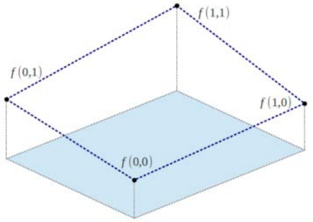
Note: if using inverse mapping idea,
then you know i’ and j’ and must solve for i and j
*So may be more useful to know i = i’ * w / w’*
*j = j’ * h / h’*

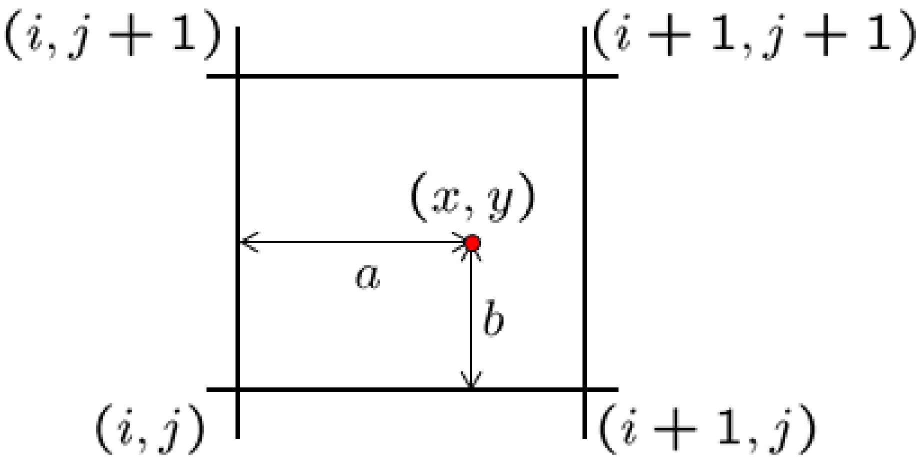
PROBLEM: Aliasing in both enlarging and reducing image size

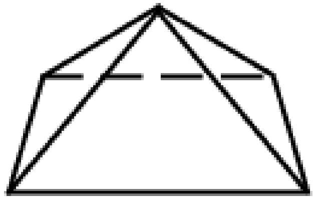
Slide based on: http://paulbourke.net/texture_colour/imageprocess/

Bilinear Interpolation







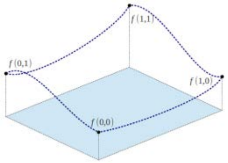


(i', j') =

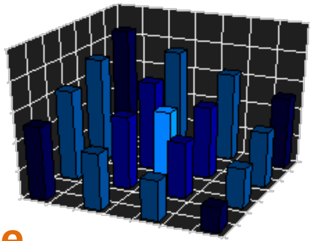
$$f(x,y) = \left.\begin{aligned} &(1-a)(1-b) && f[i,j] \\ &+a(1-b) && f[i+1,j] \\ &+ab && f[i+1,j+1] \\ &+(1-a)b && f[i,j+1] \end{aligned}\right\}$$

uses 4 pixel values from original image, f

Results are better than nearest neighbor
But there is still better

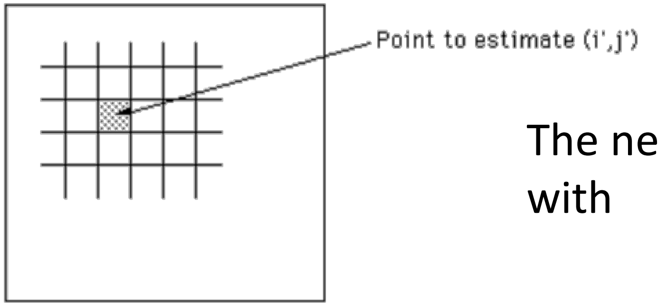


Bicubic Interpolation



Objective: Determine the color of every point (i’, j’) in the destination image

Point (i’, j’) of destination image corresponds to
a non-integer position in the source image → (x, y) = where x = i*w’ / w
y = j*h’ / h

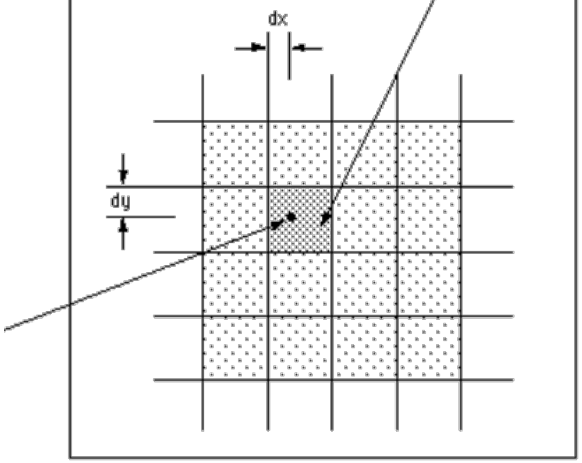


Final image

The nearest pixel coordinate (i, j) is the integer part of x and y
with dx = x – i and dy = y – j

the entire square is pixel at (i, j)
-- > 16 pixels are being used

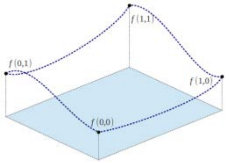
Transformed
position of (i’, j’)
i.e. (x, y)



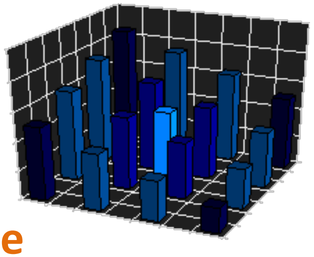
Original image

Recall Again
Inverse map idea:
*i = i’ * w / w’*
*j = j’ * h / h’*

Slide based on: http://paulbourke.net/texture_colour/imageprocess/

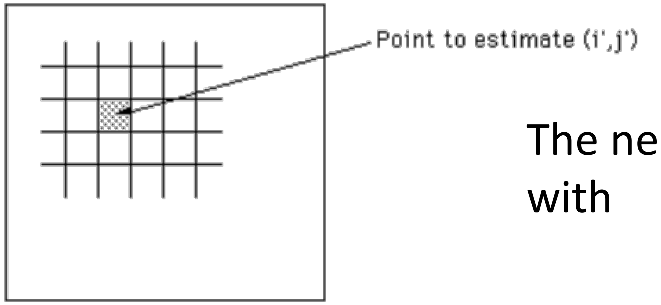


Bicubic Interpolation



Objective: Determine the color of every point (i’, j’) in the destination image

Point (i’, j’) of destination image corresponds to
a non-integer position in the source image → (x, y) = where x = i*w’ / w
y = j*h’ / h



Final image

The nearest pixel coordinate (i, j) is the integer part of x and y
with dx = x – i and dy = y – j

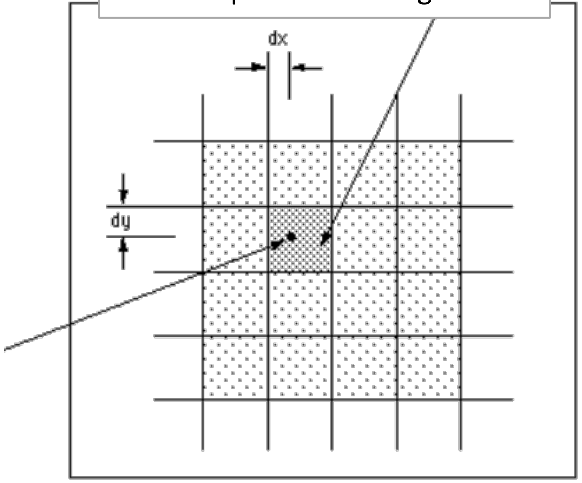
the entire square is pixel at (i, j)
-- > 16 pixels are being used

$$F(i',j') = \sum_{m=-1}^{2} \sum_{n=-1}^{2} F(i+m, j+n)\, R(m-dx)\, R(dy-n)$$

$$R(x) = \frac{1}{6}\left[P(x+2)^3 - 4\,P(x+1)^3 + 6\,P(x)^3 - 4\,P(x-1)^3 \right]$$

$$P(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Transformed
position of (i’, j’)
i.e. (x, y)



Original image

Recall Again
Inverse map idea:
*i = i’ * w / w’*
*j = j’ * h / h’*

Slide based on: http://paulbourke.net/texture_colour/imageprocess/

Outline

- Review: Image Fundamentals
- Review: Warping, Filtering, Interpolation
- Image Morphing
  - Cross Fading
  - Feature Correspondence
    - Warping Interpolation Options
      - Splines
      - Triangular Mesh
      - Radial Basis Functions (RBFs)

More Recap: Morphing

- Morphing is a special effect in animation that changes one image into another through a seamless transition
  - Early methods used cross-fading techniques on film
  - More common now
    - Is a combination of generalized image warping with a cross dissolve between image elements

Morph: Cross Fading/Dissolving

- Averaging Images (an “easy” morph)
  - Using the compositing equation
    - $C = \alpha F + (1 - \alpha)B$

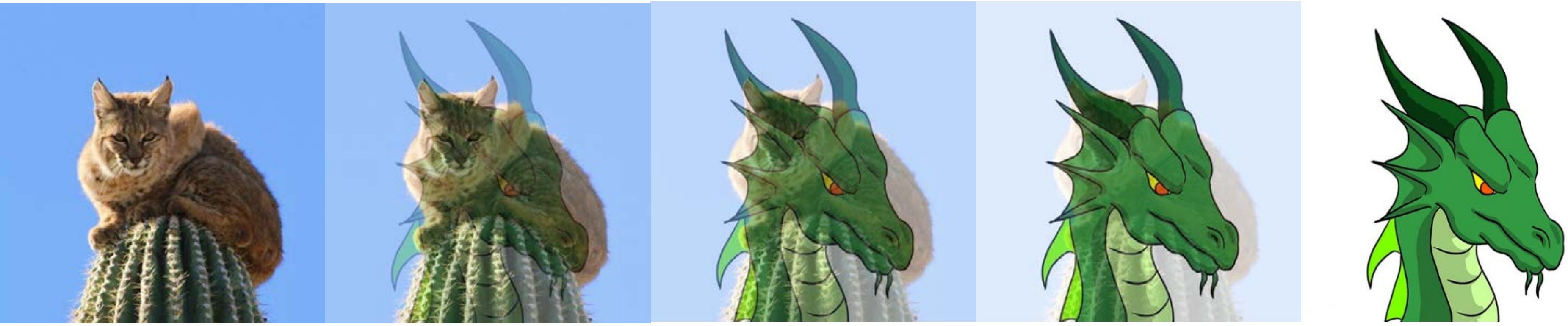


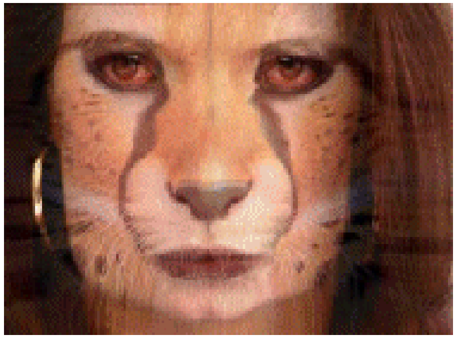
Image Morphing WITHOUT feature correspondence

Example code for this should be online: *c015_crossfading*

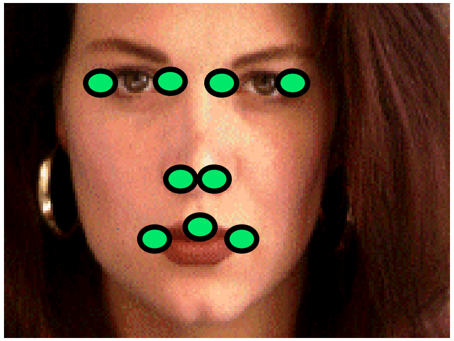
Morphing: Feature Correspondence

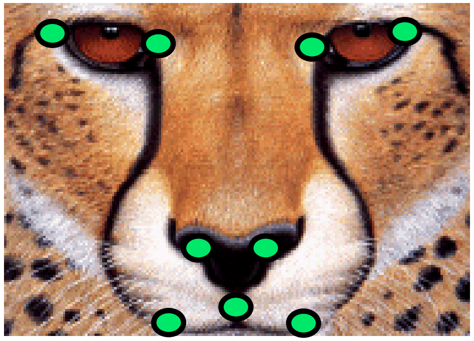
- Input 2 images I_0 and I_N:

- Output is image sequence I_i, with i = 1..N-1



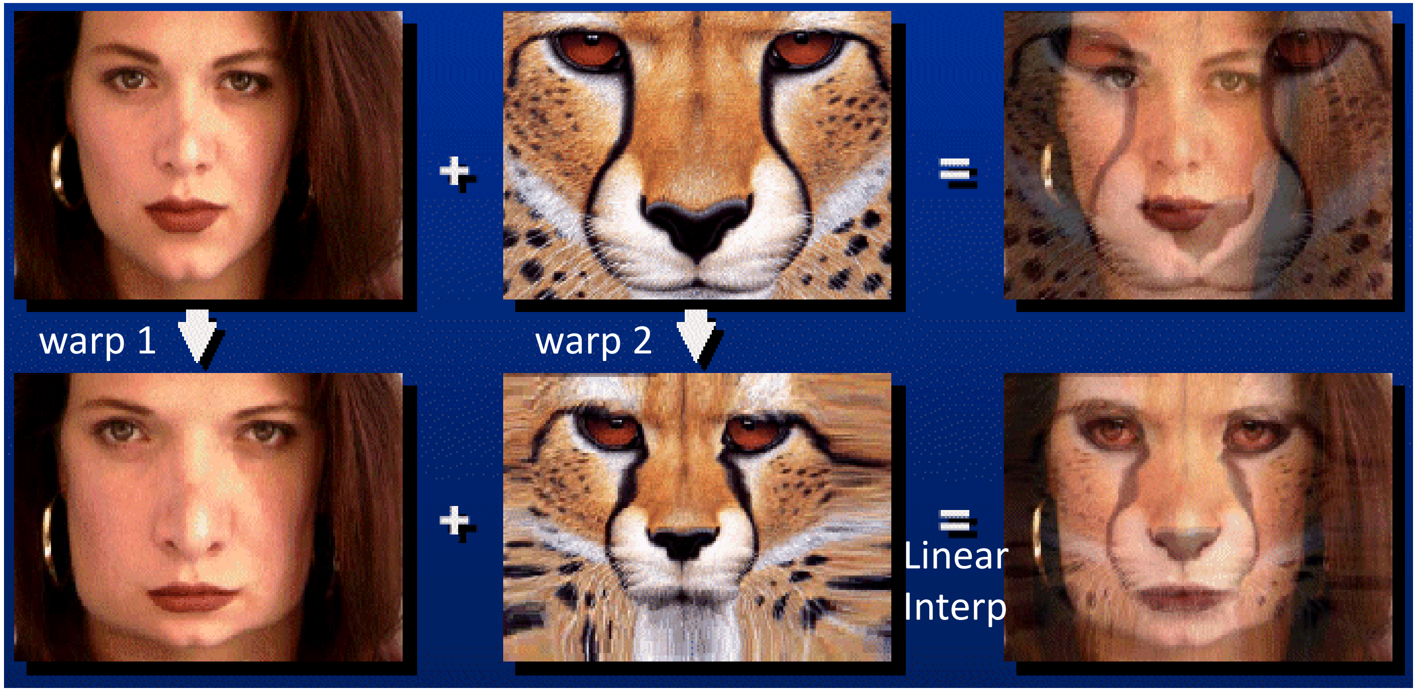
- User specifies sparse correspondences on the images
  - Vector pairs: { (P_j^0 , P_j^N) }





Morphing: Feature Correspondence

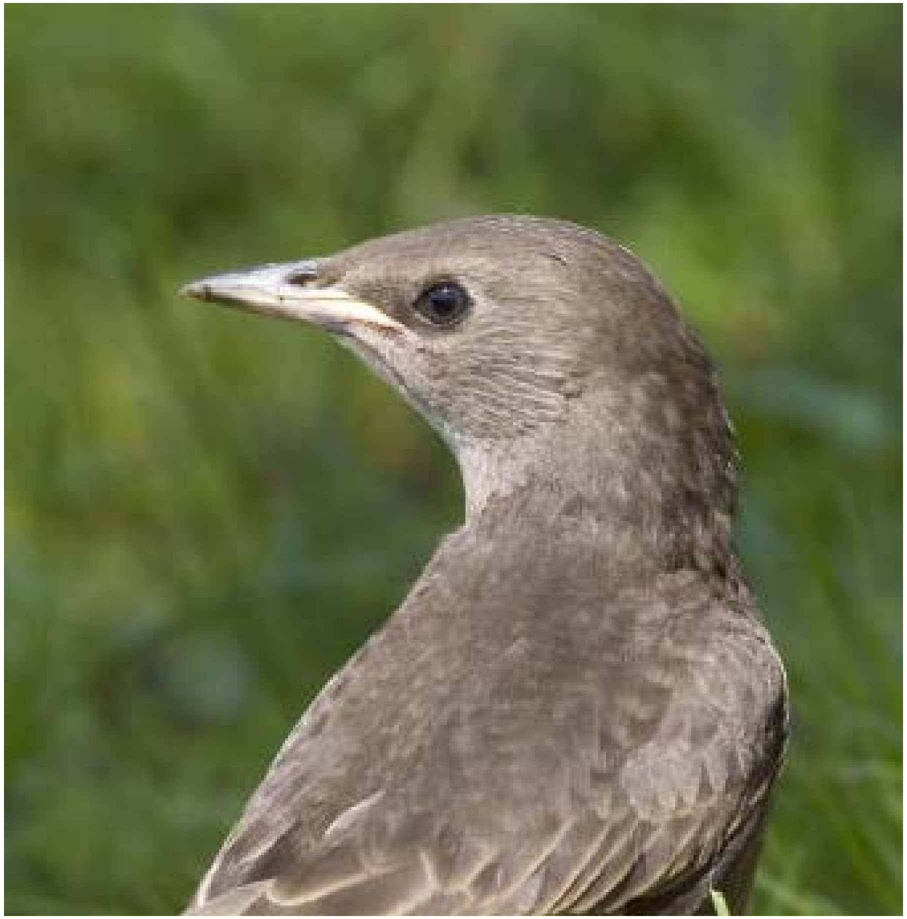
- For each intermediate frame I_t
  - Interpolate feature locations $P^t_i = (1- t)\ P^0_i + t\ P^1_i$
  - Perform **two warps**: one for I_0, one for I_1
    - Deduce a dense warp field from the pairs of features
    - Warp the pixels
  - Linearly interpolate the two warped images

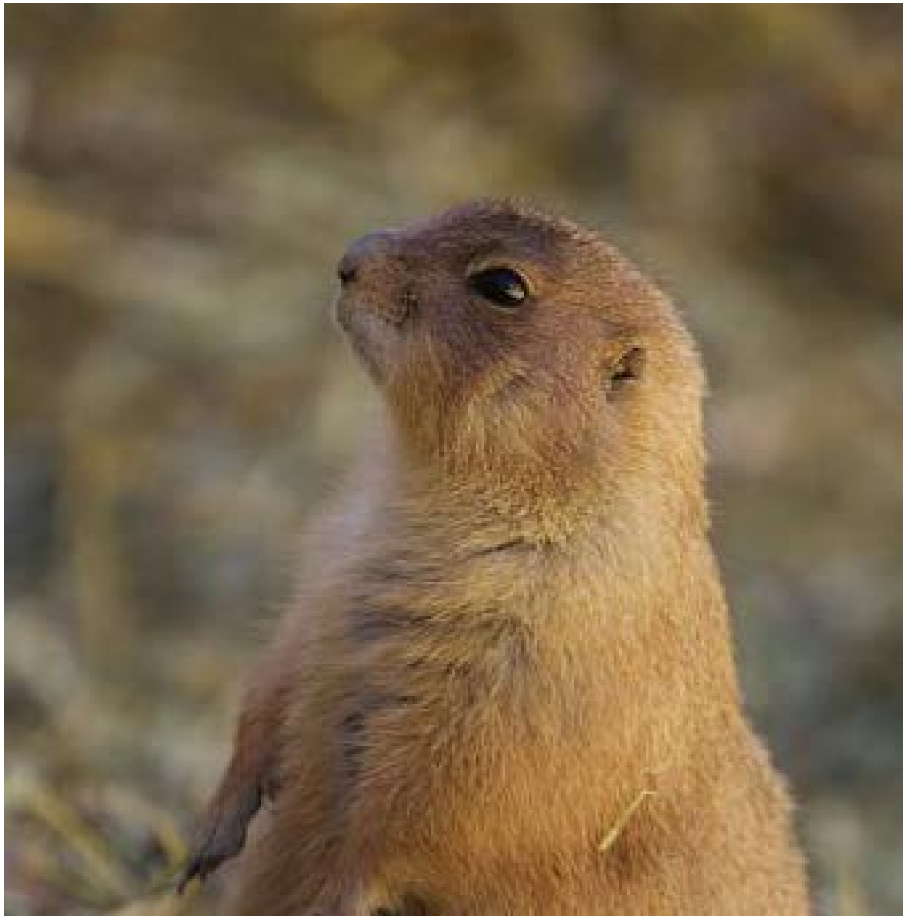


crossfade

morph with feature correspondence

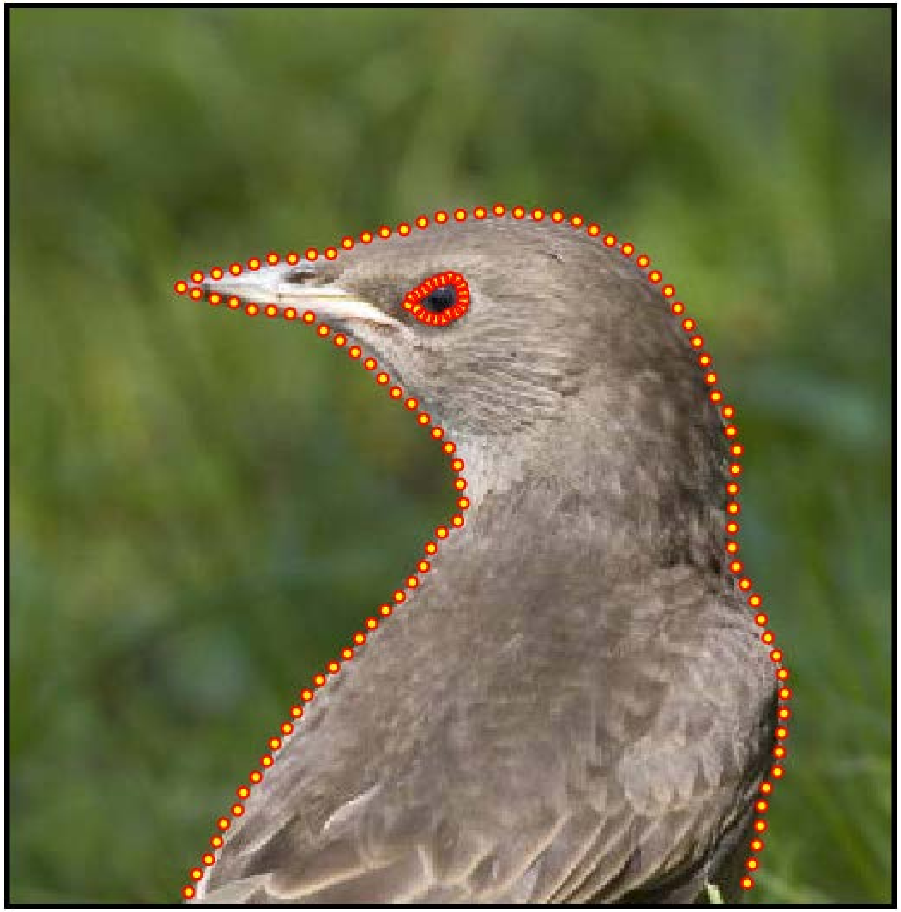
Example: Input Images

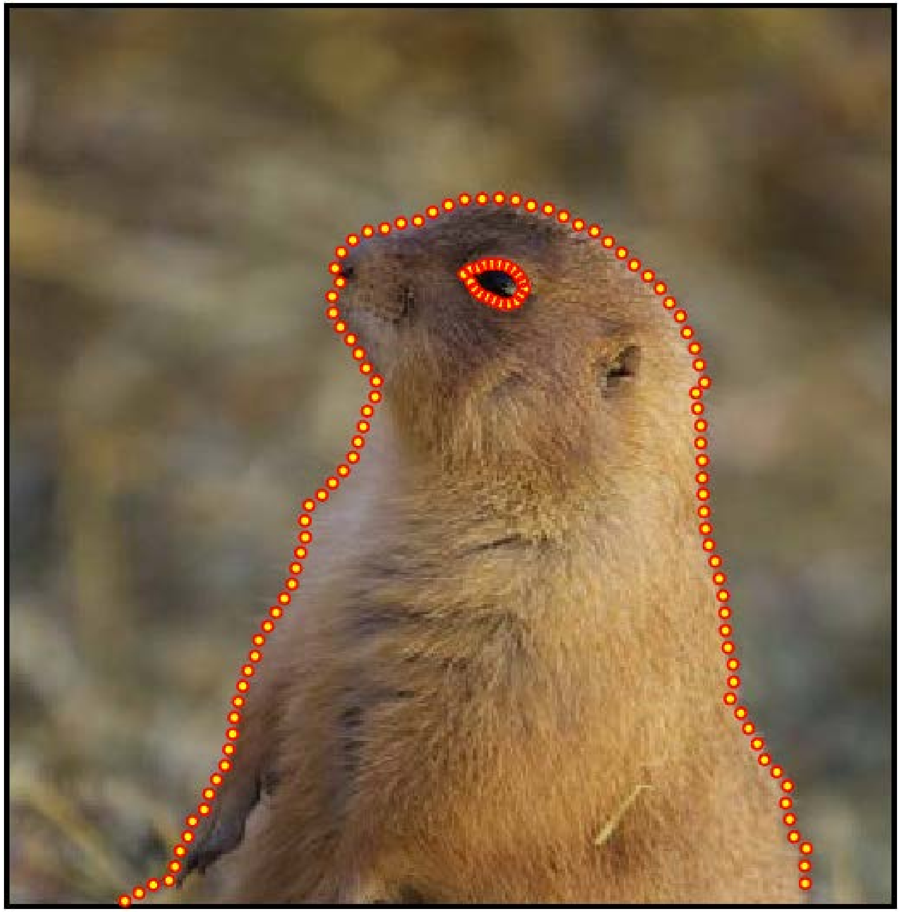




Slide derived/taken from presentation of Fredo Durand and Bill Freeman

Feature Correspondences

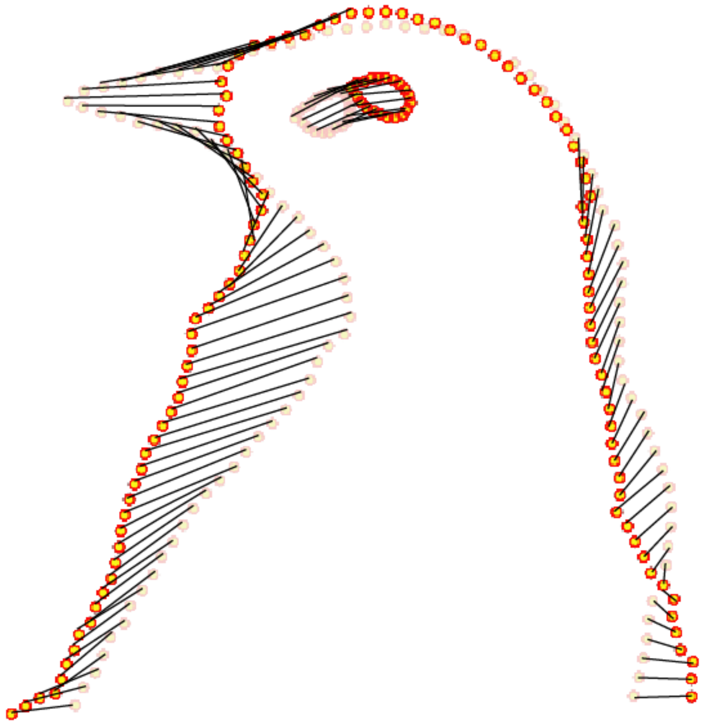




- Feature locations are the 2D vector points: y_i.

Slide derived/taken from presentation of Fredo Durand and Bill Freeman

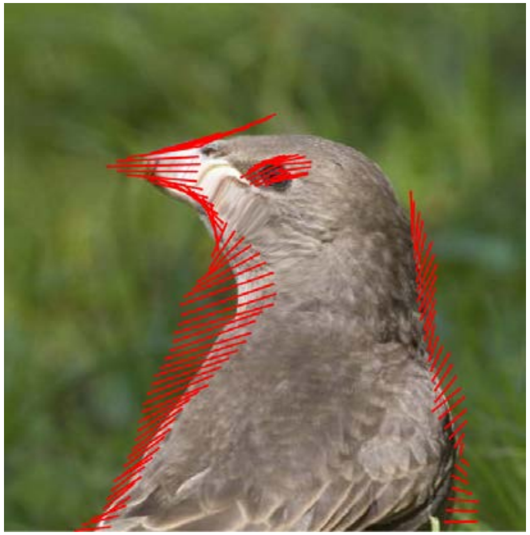
Interpolate Feature Locations



- Interpolate between the 2D vectors
  - – Provides the 2D vector points: x_i .

Slide derived/taken from presentation of Fredo Durand and Bill Freeman

Warp Each Image to Intermediate Location

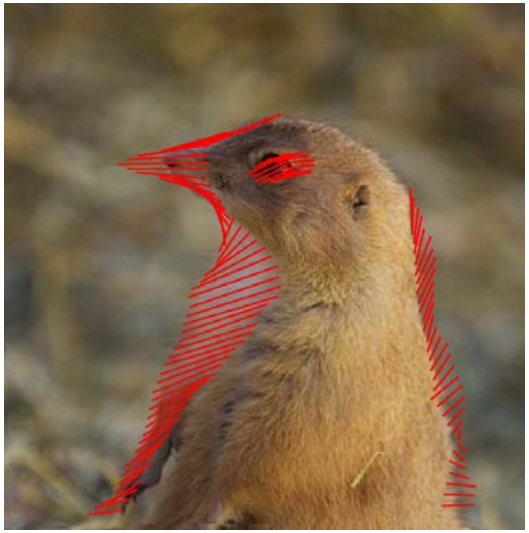


Two DIFFERENT warps
Same target location
but different source location

The x_i are the same
(intermediate locations)
The y_i are different
(source feature locations)

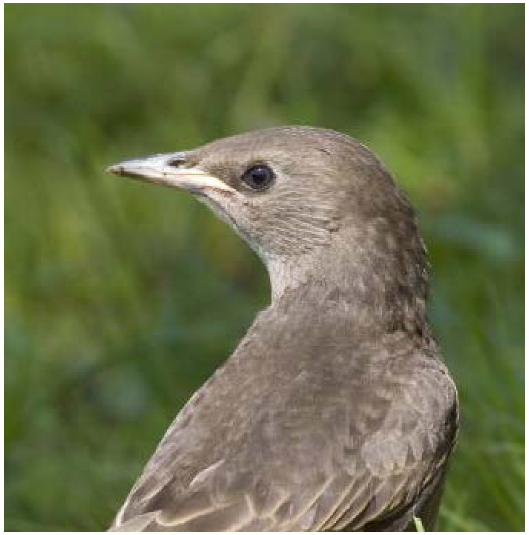
The y_i do NOT change throughout the animation
BUT
the x_i are different for each intermediate image

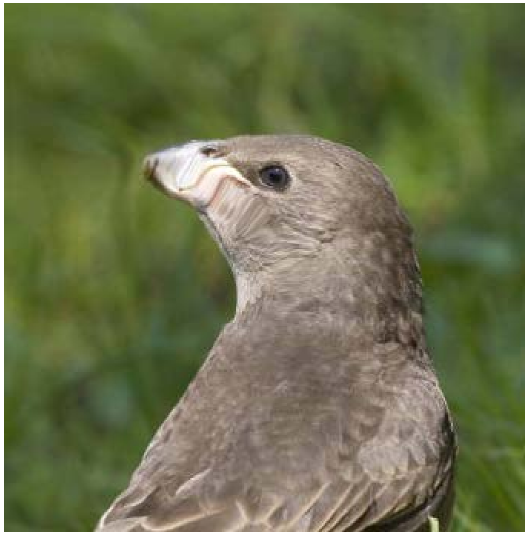
Images shown are for t = 0.5
-- the y_i are in the middle

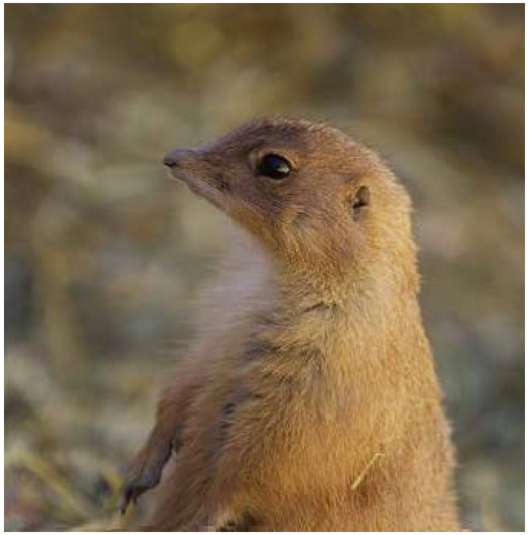


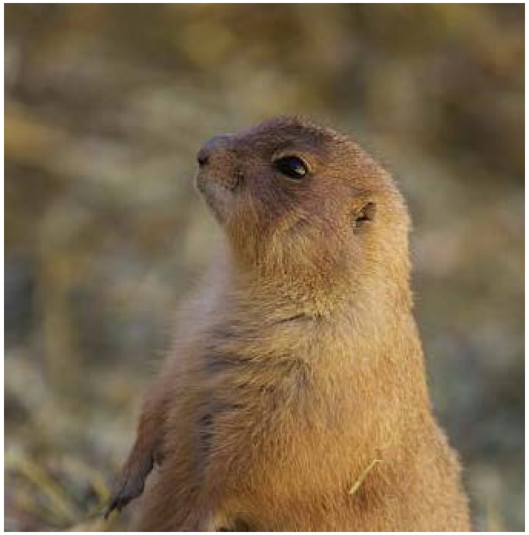
Slide derived/taken from presentation of Fredo Durand and Bill Freeman

Warp each image to Intermediate Locations



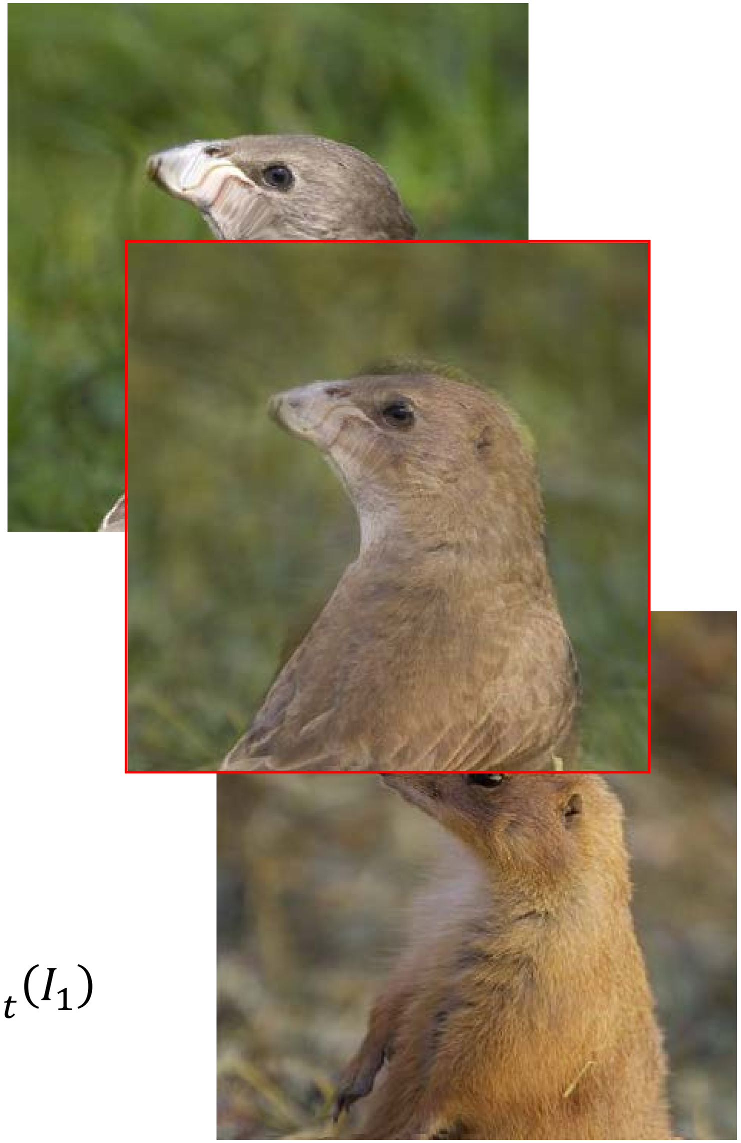






Slide derived/taken from presentation of Fredo Durand and Bill Freeman

Linearly Interpolate Colors



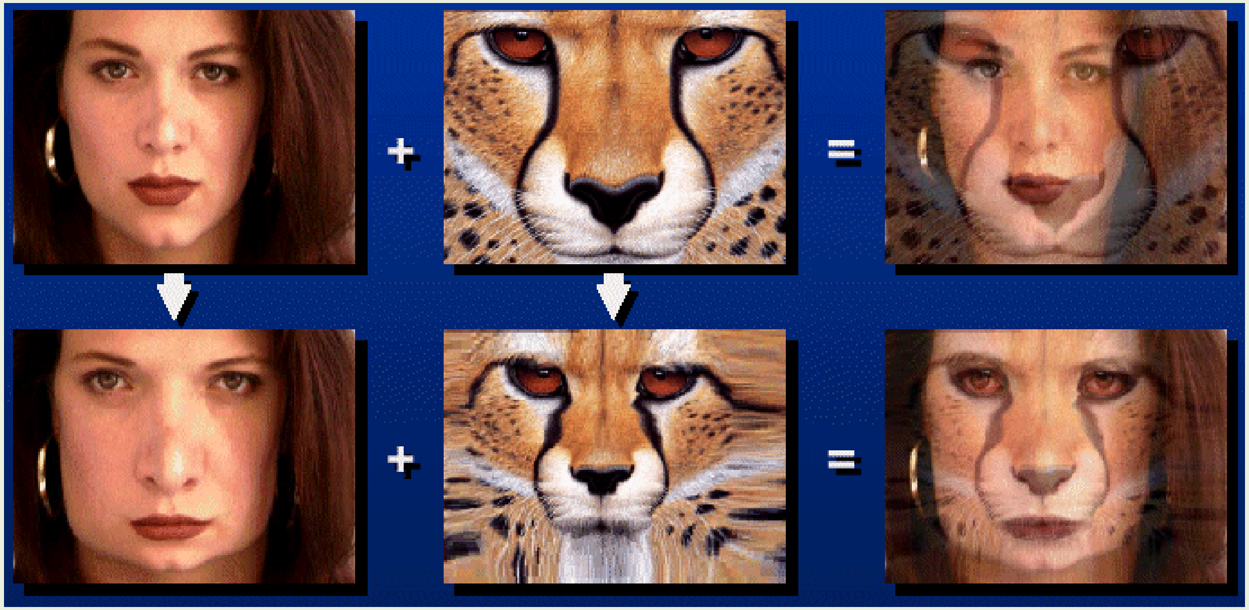
Interpolation weights
are a function of time

$$C = (1 - t){f^0}_t(I_0) + t{f^1}_t(I_1)$$

Slide derived/taken from presentation of Fredo Durand and Bill Freeman

Feature Summary So Far

- For each intermediate frame I_t :
  - Interpolate feature locations
    - $y^t_i = (1\text{-}t)x^0_i + tx^1_i$.
  - Perform two warps: one for I_0 and one for I_1
    - Calculate a warp field from the feature pairs
    - Warp the pixels
  - Linearly interpolate the two warped images

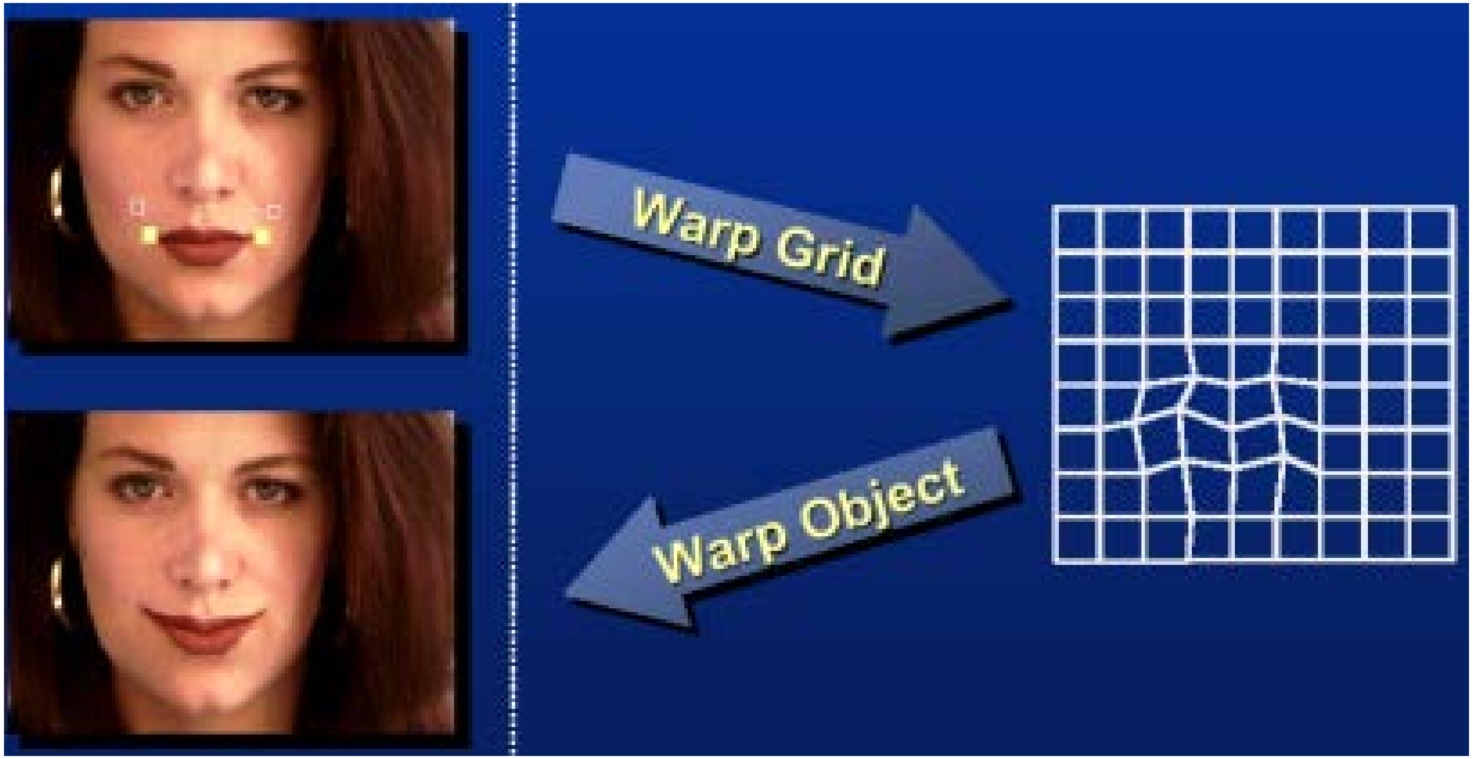


Warp Options

- Feature Point to Feature Point
  - But how exactly do those warps work?

Warp Options: non-parametric

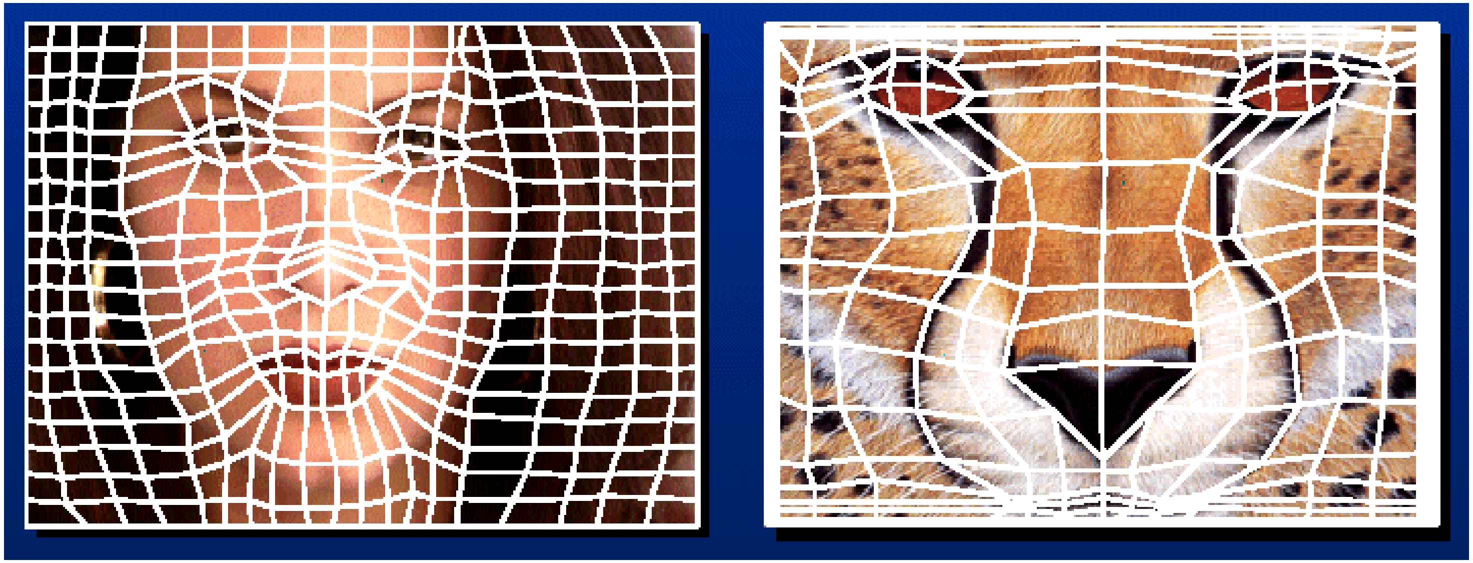
- Move control points to specify a spline warp
- Spline produces a smooth vector field



Slide Alyosha Efros

Warp Specification – too dense

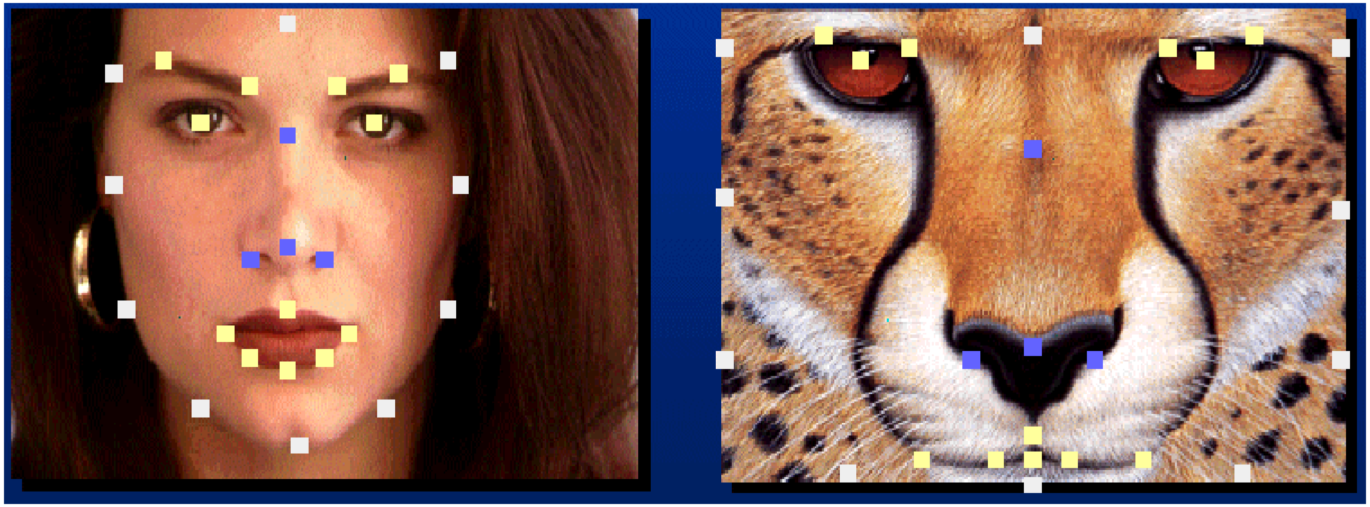
- We could specify all the spline control points
  - Interpolate to complete a warping function



But we really want to specify only a few points – not the entire grid

Warp Specification – too dense

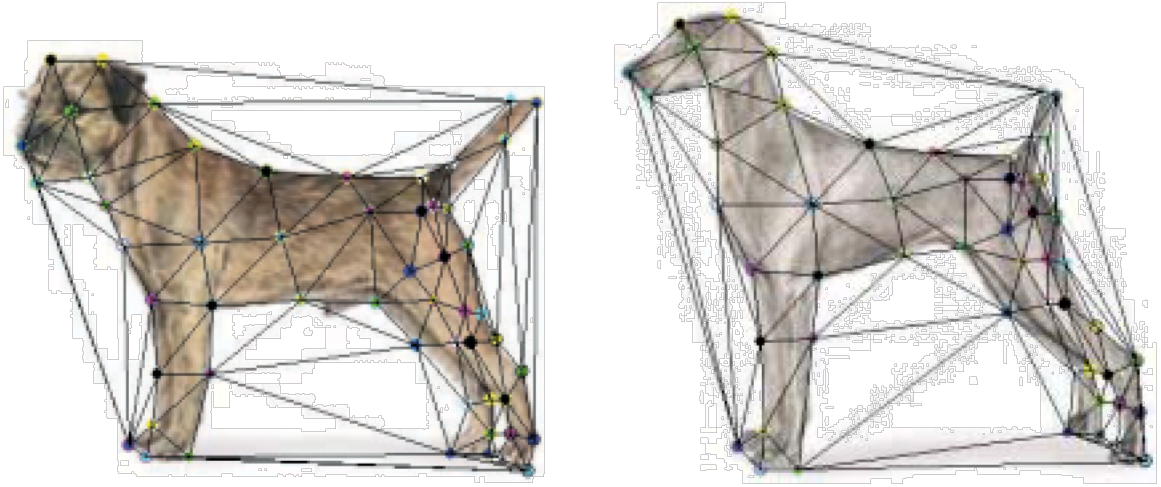
- We should instead **specify corresponding points**
  - **Interpolate to complete** a warping function



How do we do it?
How do we go from feature points to pixels?

Triangular Mesh of Images

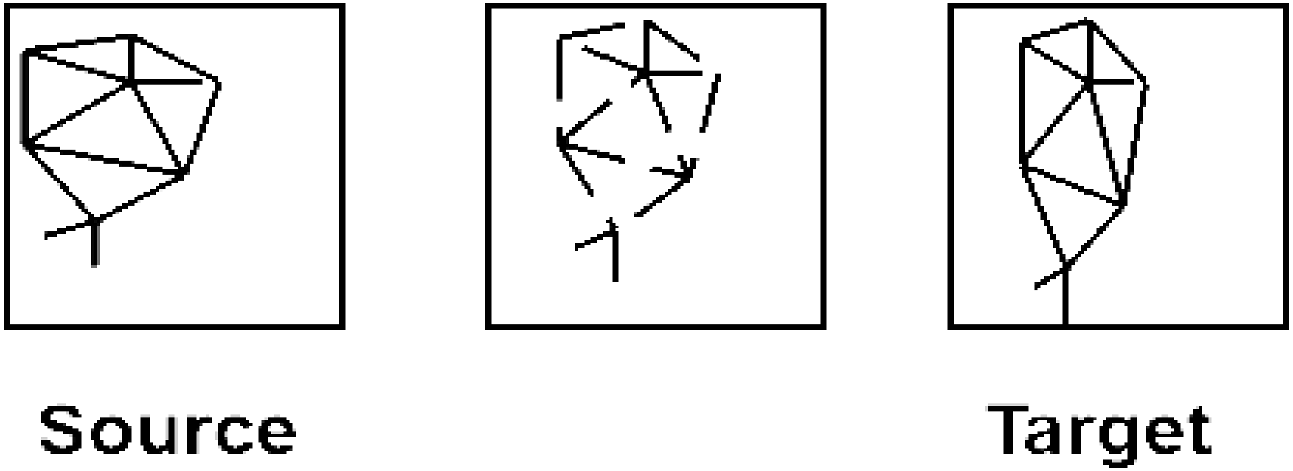
- Input correspondences at key feature points
- Define a triangular mesh over the points
  - Use SAME MESH for both images
  - Provides triangle-to-triangle correspondences
- Warp each triangle separately from source to destination



Slide Alyosha Efros

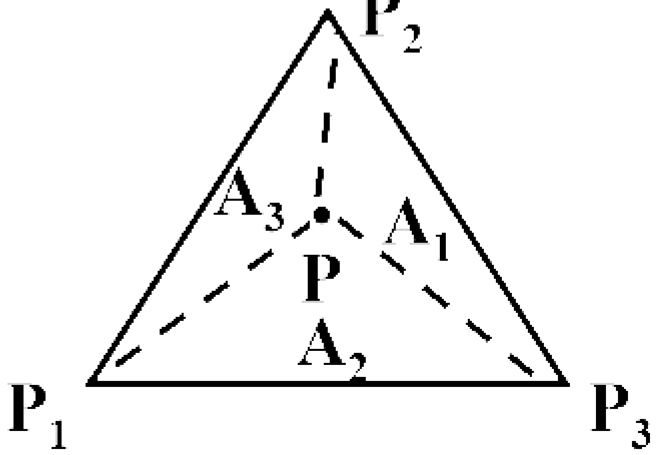
Morphing: Triangle Method

- Feature points are marked on source and target images
  - i.e. correspondence points identified
- Points are use to form triangles
- Triangulation is interpolated for intermediate frames
- Images are warped based on the interpolation of points
- Colors are blended (as in crossfade)



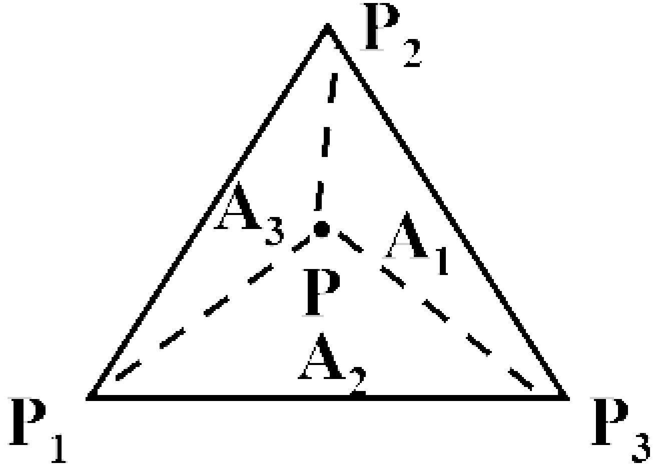
Morphing: Triangle Method

- Interpolation is in the triangular domain
  - How is P related to P_1, P_2, and P_3 ?



Morphing: Triangle Method

- Interpolation is in the triangular domain
  - – How is P related to P_1, P_2, and P_3 ?



$$P = uP_1 + vP_2 + wP_3\ .$$

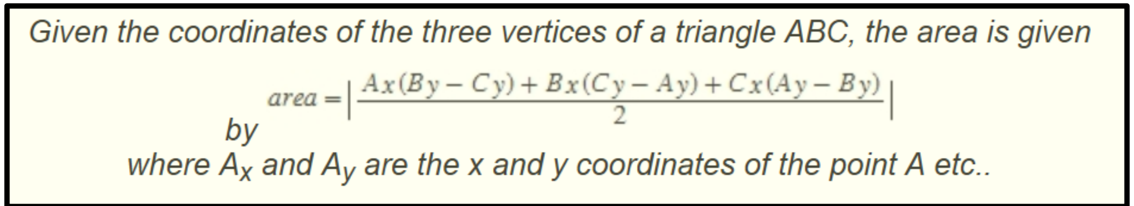
A = area of triangle

$$u = A_1 / A$$

$$v = A_2 / A$$

$$w = A_3 / A$$

u, v, w: Barycentric coordinates



Given the coordinates of the three vertices of a triangle ABC, the area is given by

$$area = \left| \frac{Ax(By - Cy) + Bx(Cy - Ay) + Cx(Ay - By)}{2} \right|$$

where A_x and A_y are the x and y coordinates of the point A etc..

Triangulation Morphing: Problems

- Not very continuous – only C^0.

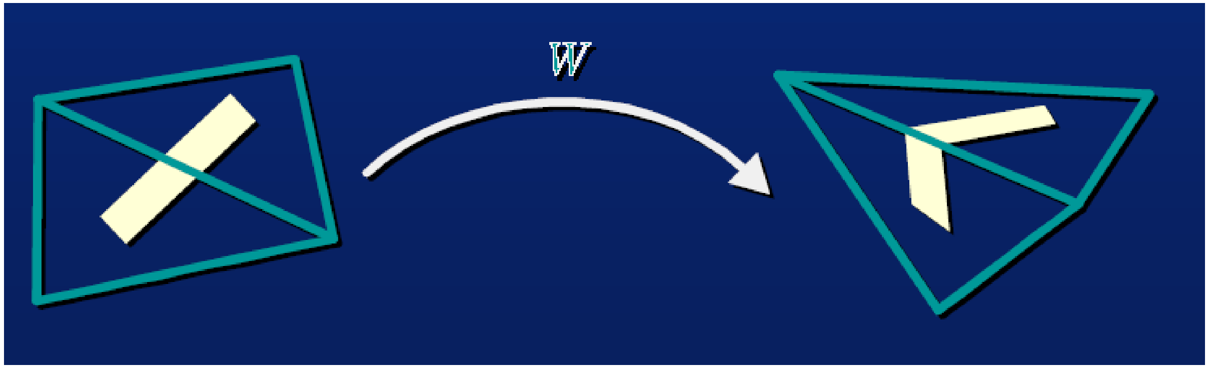
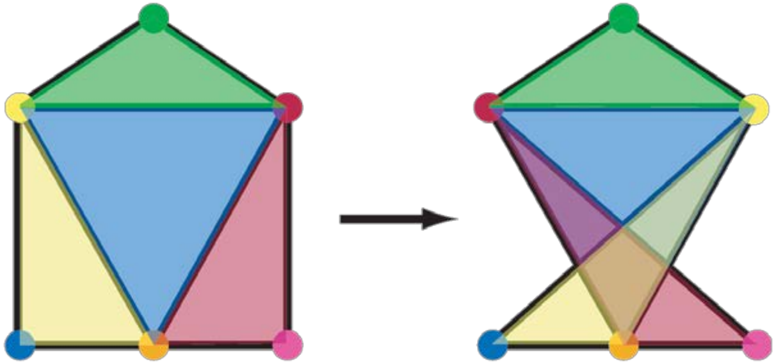


Fig. L. Darsa

- Folding problems



Desires of Warp Interpolation

- Looking for a warping field
  - A function that given a 2D point returns a warped 2D point

- Only have a sparse number of correspondence points
  - These points specify values of the warping field

- This is an interpolation problem
  - Given sparse data, find a smooth function

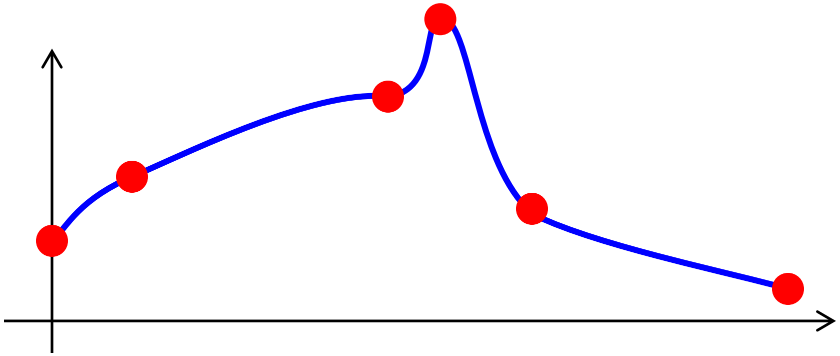
Interpolation in 1D

- Looking for a function *f*

- *Have N data points: x_i, y_i .*
  - Scattered points → spacing between x_i is non-uniform

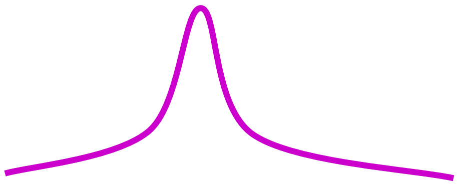
- Desire f such that
  - For each i, *f*(x_i) = y_i
  - *f* is smooth

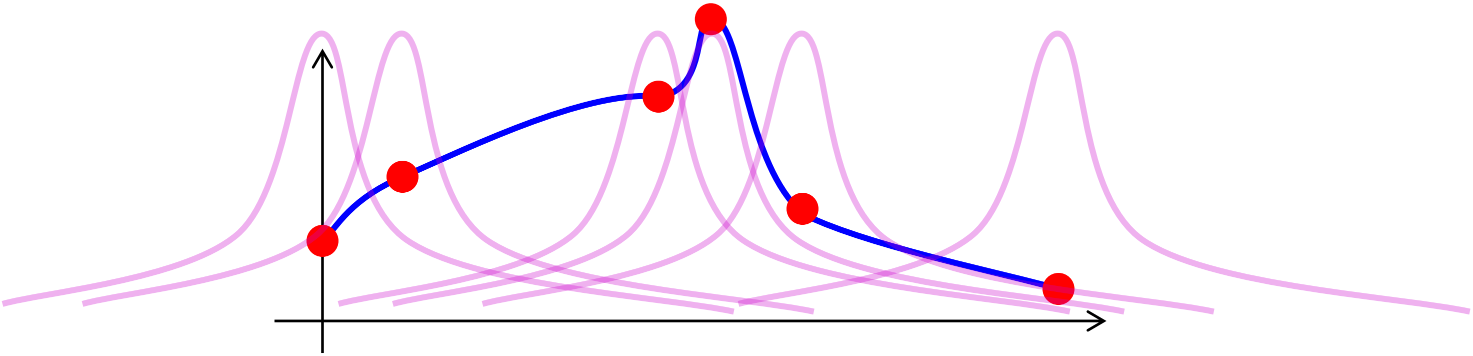
- What "smooth" means can yield different *f*



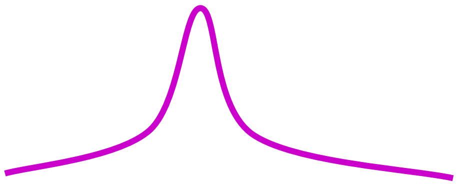
Radial Basis Functions (RBF)

- Place a smooth kernel R centered on each data point x_i

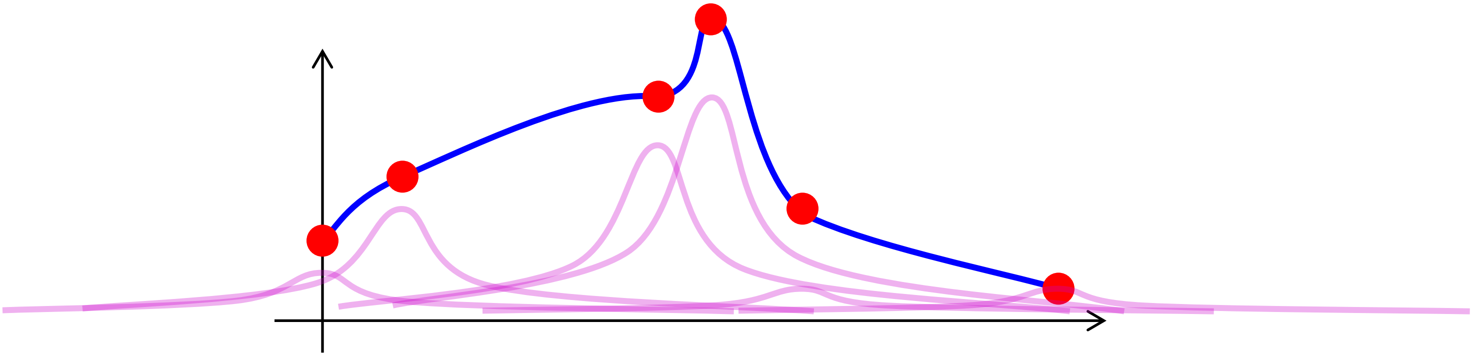




Radial Basis Functions (RBF)



- Place a smooth kernel R centered on each data point x_i
- $f(z) = \Sigma\alpha_i\ R(z, x_i\ )$
  - Find weights α_i to so $f(x_i) = y_i$ for all i



Kernel Choices

- Many choices for what kernel function to use

An option to use is an inverse multi-quadric

$$R(z, x_i) = \frac{1}{\sqrt{c + \|z - x_i\|^2}}$$

Notice ***c*** controls falloff

Easy choice is to pick a constant ***c*** for every kernel

Better option
For each kernel select a unique ***c*** such that
c is the squared distance to the nearest other $\boldsymbol{x_j}$.

Other Kernel Options

- Gaussians

$$e^{-r^2/2\sigma}$$

- Thin plate splines

$$r^2 \log r$$

» *Aside:*

- *May want or need to add a global polynomial term*

Applying the Interpolation

- $f(z) = \Sigma\alpha_i\ R(z, x_i)$
- N equations
  - for each j, $f(x_j) = y_j$

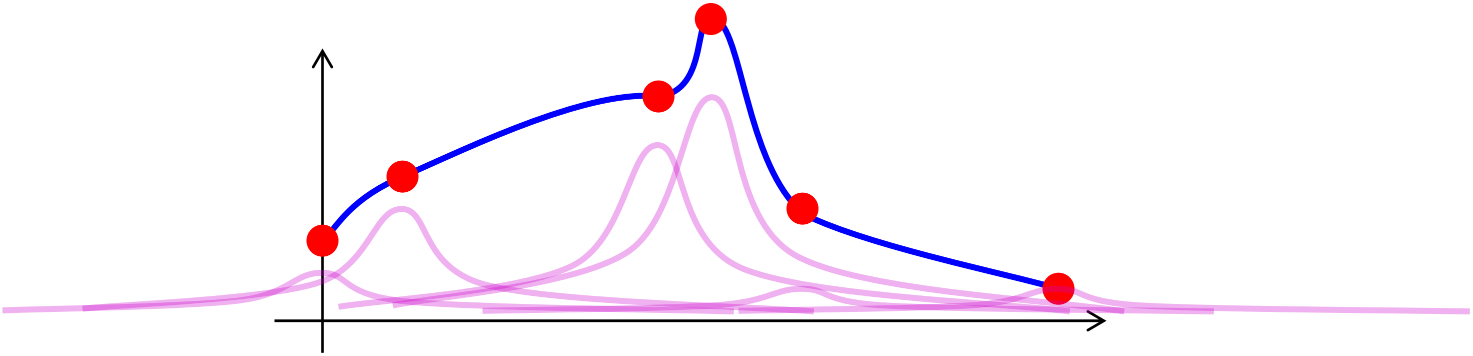
    $y_j = \Sigma\alpha_i\ R(z, x_i)$
- N unknowns: α_i .
  - Invert the matrix

Differences of Methods

- $f(z) = \Sigma \alpha_i R(z, x_i)$

Note, at a given data point
the influence of each function is NON-ZERO EVERYWHERE
→ the values of the other bases are not zero

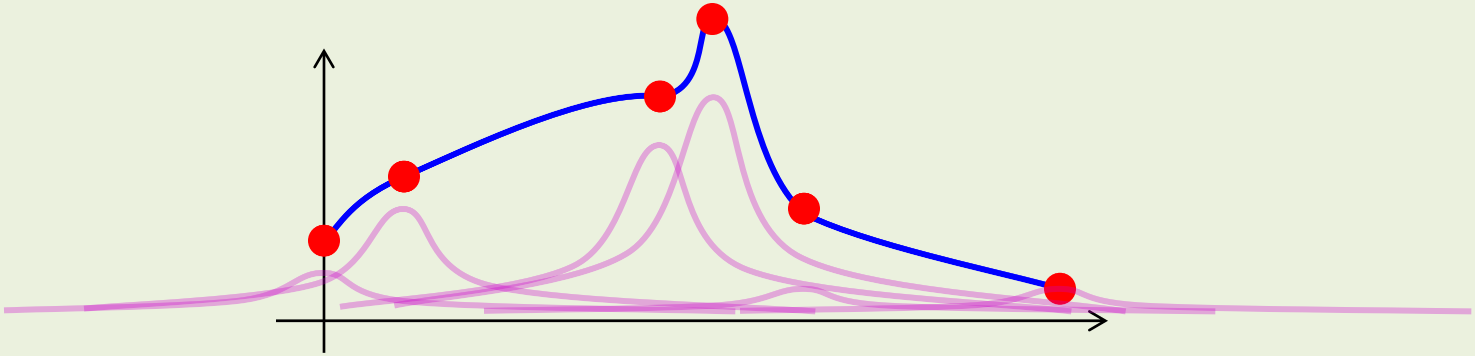
This is different from interpolation splines
→ know the neighborhood of influence



Recap: 1D Scattered Data Interpolation

- Sparse input/output pairs: x_i, y_i.
  - non-uniform sampling

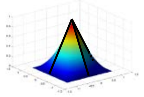
- Radial Basis Functions (RBFs)
  - Weighted sum of kernels R centered at data points
    - $f(z) = \Sigma\alpha_i R(z, x_i)$
  - Compute the weights α_i, by enforcing interpolation
    - $f(x_j) = y_j$.
    - Simple linear system



RBF warping: 2D case

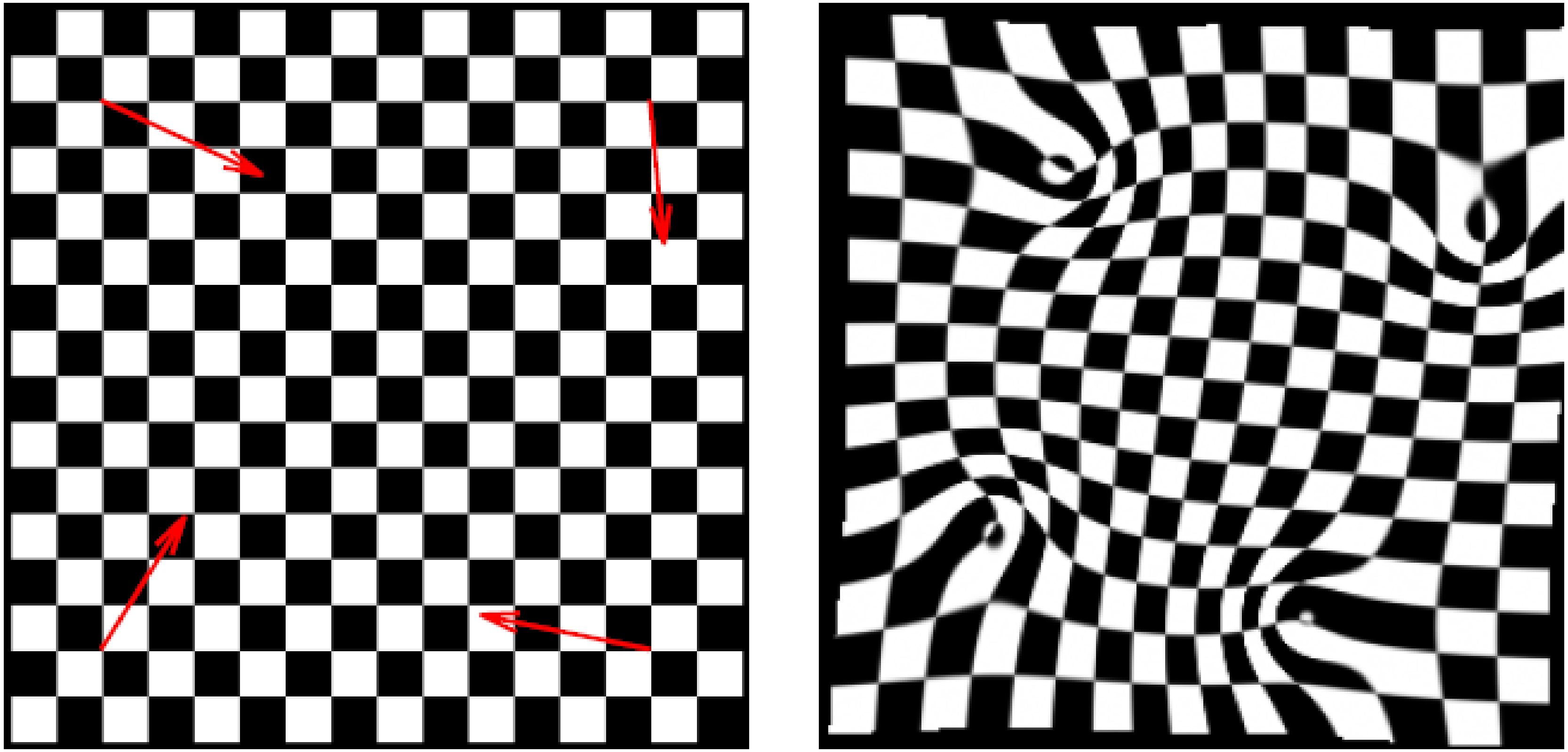
- Think vector functions
  - f was R→R, is now $R^2 \rightarrow R^2$.

- Still have N data points
  - x_i and y_i are now 2D vectors
  - Use 2D kernels at each data point *(think cone-like)*
  - The weights, α_i, are also 2D vectors

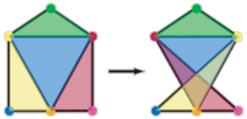


- Solve a linear system of 2N equations and 2N unknowns

Example



Still may have folding problems…



RBF: Further Investigation

- Students are encouraged to perform further investigations using RBFs
  - – Contrasting with the spline interpolation methods
  - – Discover other applications/uses of RBFs

de Boor, C. (1978).
A practical guide to splines, New York: Springer Verlag.

Brenner, S and Scott, L (1994).
The mathematical theory of finite elements Springer, New York.

Bishop, C.M. (1995).
Neural networks for pattern recognition, Oxford: Clarendon Press.

Powell, M J D (1997).
A new iterative algorithm for thin plate spline interpolation in two dimensions, *Annals of Numerical Mathematics* 4: 519-527.

Davis, P J (1975).
Interpolation and Approximation Dover, New York.

Turk,G.and O’Brien, J. 1999. Shape transformation using variational implicit functions. Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 1999), 335–342.

Witkin,A.P. And Heckbert, P.S. 1994. Using particles to sample and control implicit surfaces. Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 94), 269–278.

Buhmann, Martin D. (2003),
Radial Basis Functions: Theory and Implementations, Cambridge University Press

Many others…

Outline

- Review: Image Fundamentals
- Review: Warping, Filtering, Interpolation
- Image Morphing
  - Cross Fading
  - Feature Correspondence
    - Warping Interpolation Options
      - Splines
      - Triangular Mesh
      - Radial Basis Functions (RBFs)
      - Other options

From Points to Lines

- Point based features can triangulate image areas
  - Triangle interpolation has folding issues
  - RBF interpolation also has folding issues

- Other options?
  - Line based features also can be used

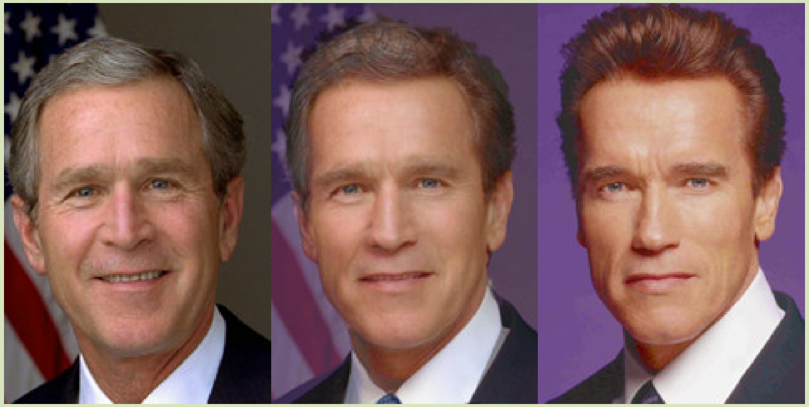
Beier, T & Neely, S. (1992). Feature-based image metamorphosis
Computer Graphics 26 (2): 35-42. doi: 10.1145/133994.134003

See also a contrast paper at:
http://airccse.org/journal/sipij/papers/2411sipij20.pdf

Bhatt Bhumika, G (2011).
Comparative Study of Triangulation based and Feature based Image Morphing
Signal & Image Processing: An International Journal (SIPIJ) Vol.2, No.4, Dec 2011.

So Again: Morphing

- **Morphing** is a special effect in animation that changes one image into another through a seamless transition.
  - Early methods used cross-fading techniques on film
  - More common now
    - **Is a combination of generalized image warping with a cross dissolve between image elements**

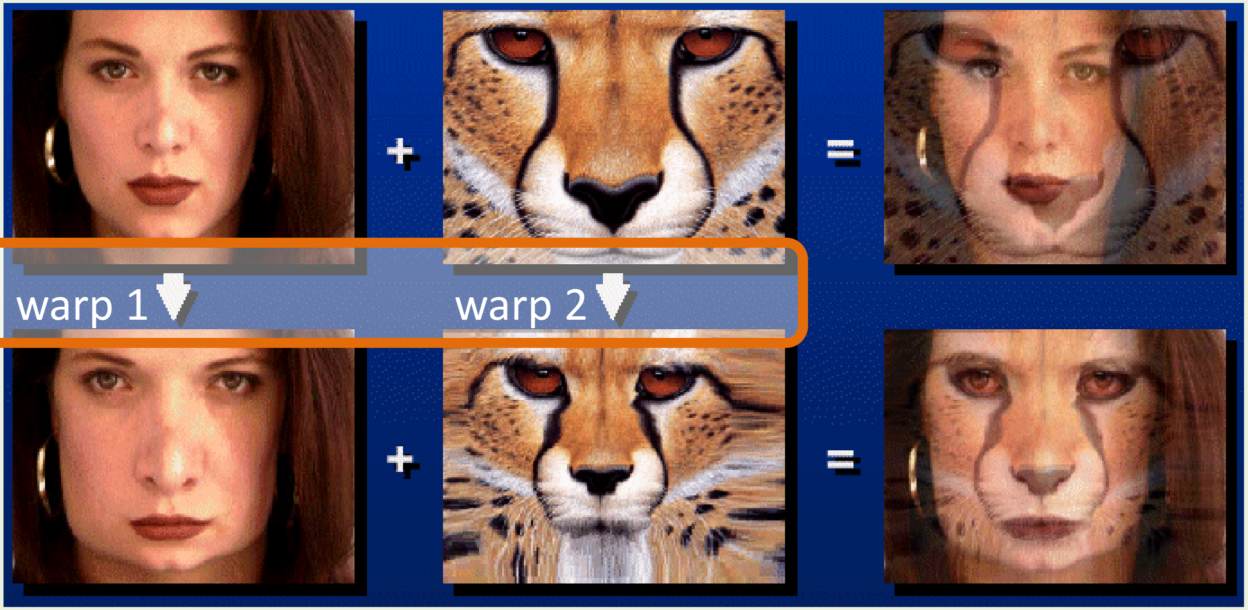


public domain images from wikimedia commons
https://commons.wikimedia.org/wiki/File:Bush-Arnie-morph.jpg

Check out the book: http://www-cs.engr.ccny.cuny.edu/~wolberg/diw.html

Feature Summary So Far

- For each intermediate frame I_t :
  - Interpolate feature locations
    - $y^t_i = (1\text{-}t)x^0_i + tx^1_i$.
  - Performa two warps: one for I_0 and one for I_1
    - Calculate a warp field from the feature pairs
    - Warp the pixels
  - Linearly interpolate the two warped images

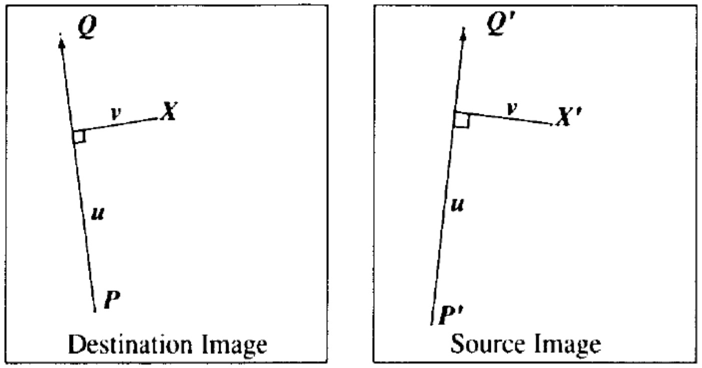


These warps require attention

Line Pair Option

- May also morph using **line pairs**
  - instead of point pairs

Beier, T & Neely, S. (1992). Feature-based image metamorphosis
Computer Graphics 26 (2): 35-42. doi: 10.1145/133994.134003



```
For each pixel $X$ in the destination image
    find the corresponding $u,v$
    find the $X'$ in the source image for that $u,v$
    destinationImage($X$) = sourceImage($X'$)
```

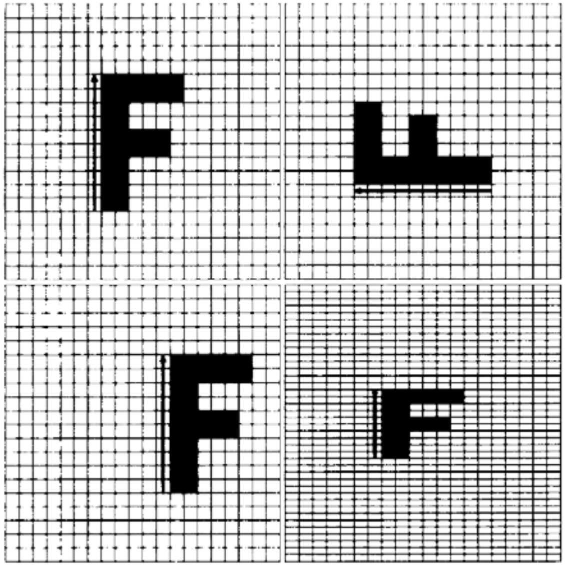
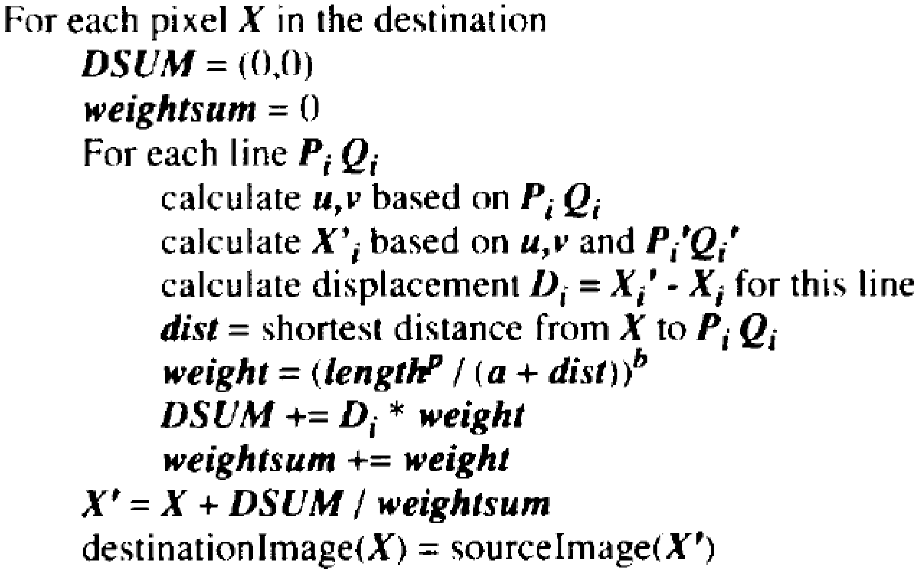
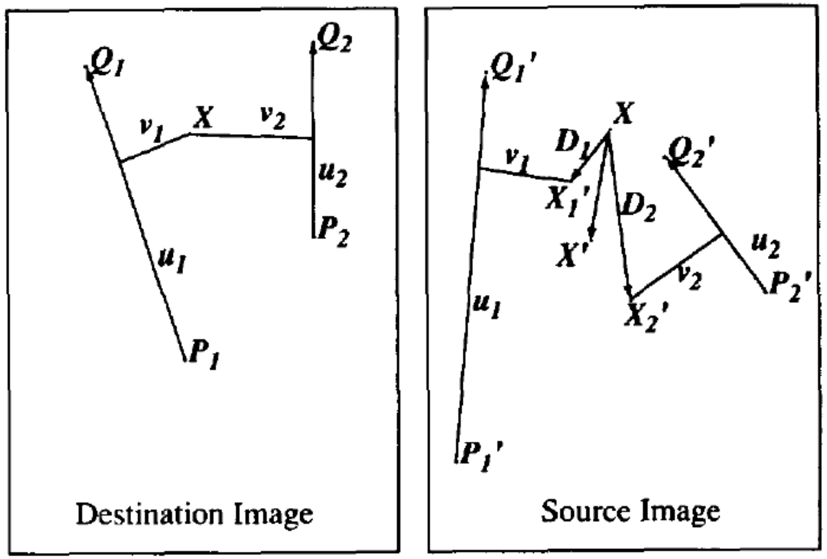


Figure 2: Single line pair examples

Multiple Line Pairs



```
For each pixel $X$ in the destination
    $DSUM$ = (0,0)
    $weightsum$ = 0
    For each line $P_i Q_i$
        calculate $u,v$ based on $P_i Q_i$
        calculate $X'_i$ based on $u,v$ and $P_i'Q_i'$
        calculate displacement $D_i = X_i' - X_i$ for this line
        $dist$ = shortest distance from $X$ to $P_i Q_i$
        $weight = (length^p / (a + dist))^b$
        $DSUM$ += $D_i$ * $weight$
        $weightsum$ += $weight$
    $X' = X + DSUM / weightsum$
    destinationImage($X$) = sourceImage($X'$)
```



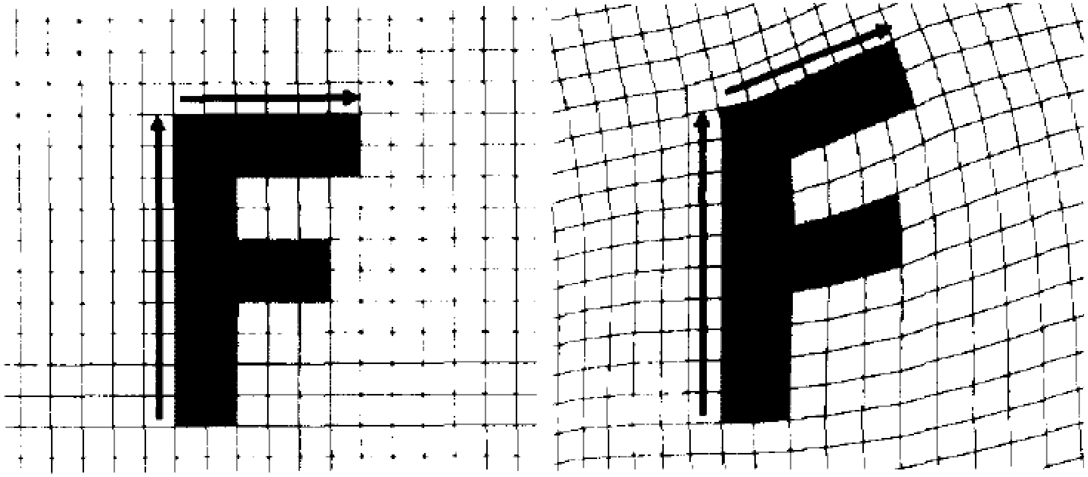


Figure 4: Multiple line pair example

See also: http://www.hammerhead.com/thad/morph.html

Challenge

- Challenge 1
  - Implement a program to crossfade 2 input images
    - Display crossfade animation

- Challenge 2
  - Implement a program to allow the user to select features on 2 different images to morph between
    - Display morphing animation
      - Suggest using line-based algorithm first
      - Then try point-based (with triangulation then RBF)

Outline

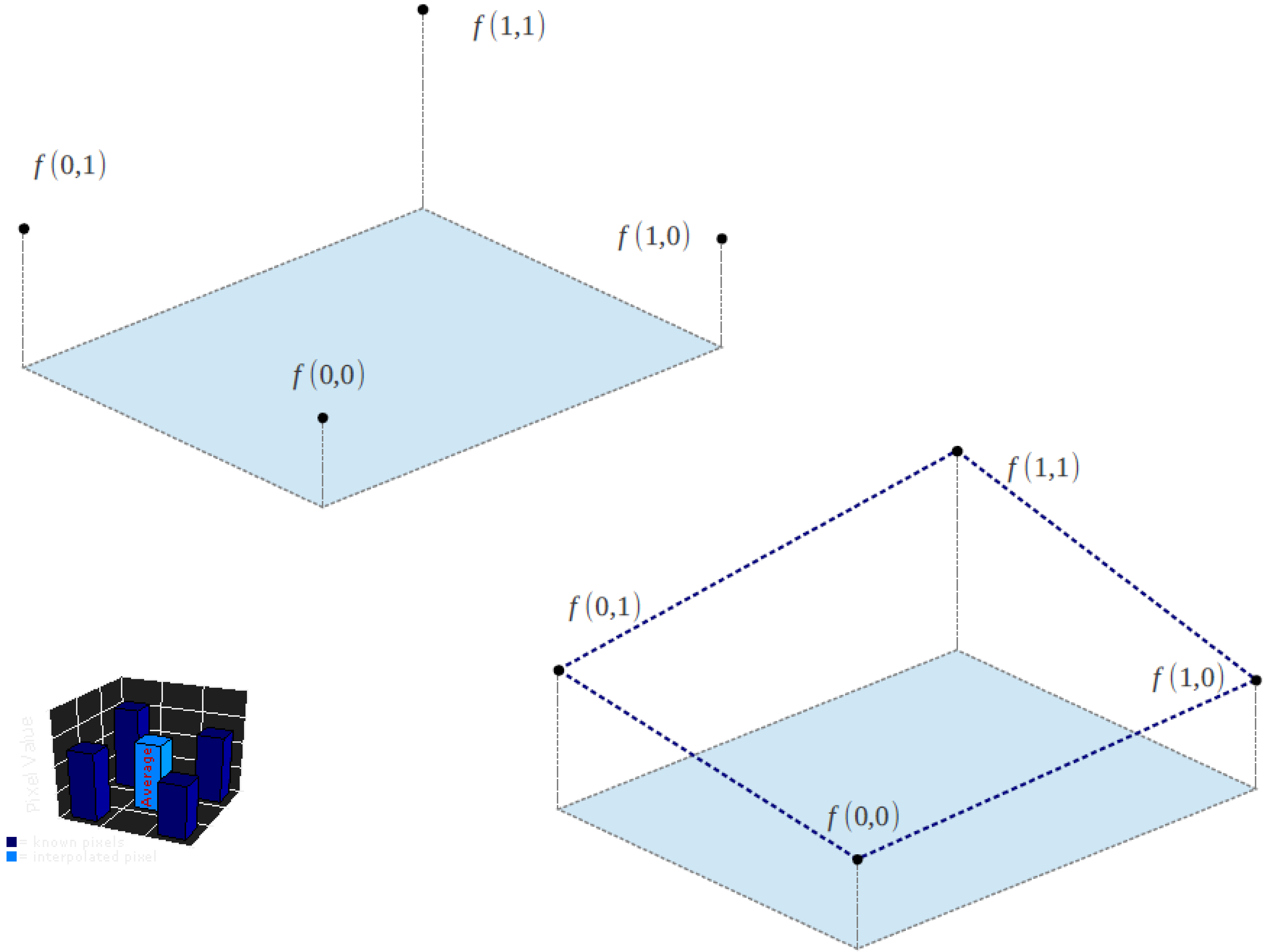
- Review: Image Fundamentals
- Review: Warping, Filtering, Interpolation
- Image Morphing
  - Cross Fading
  - Feature Correspondence
    - Warping Interpolation Options
      - Splines
      - Triangular Mesh
      - Radial Basis Functions (RBFs)
      - Other options

Questions?

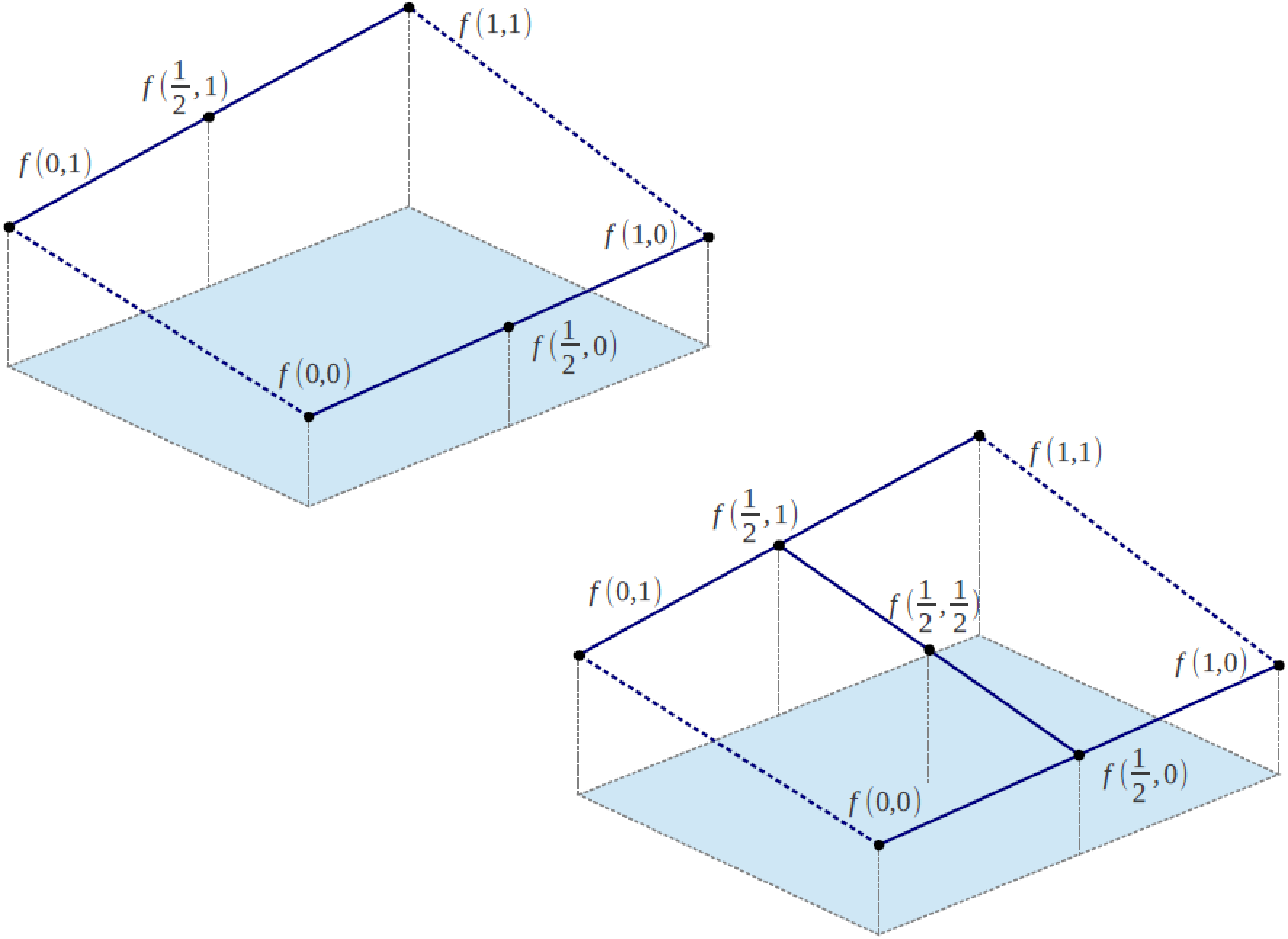
- Beyond D2L
  - Examples and information can be found online at:
    - *http://docdingle.com/teaching/cs.html*

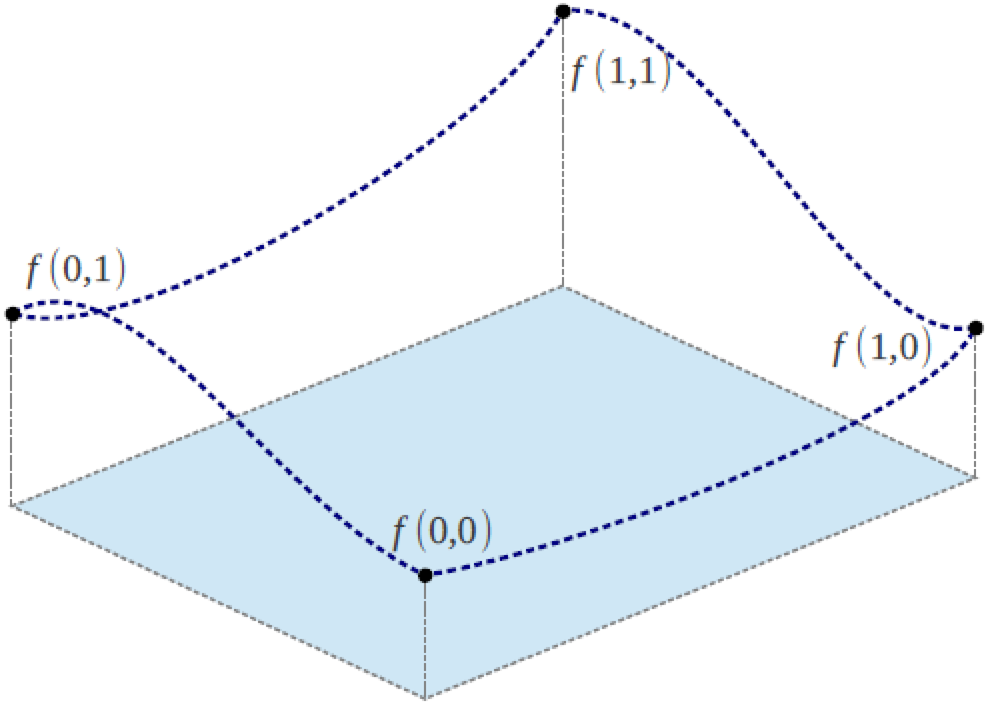
    - *Continue to more stuff as needed*

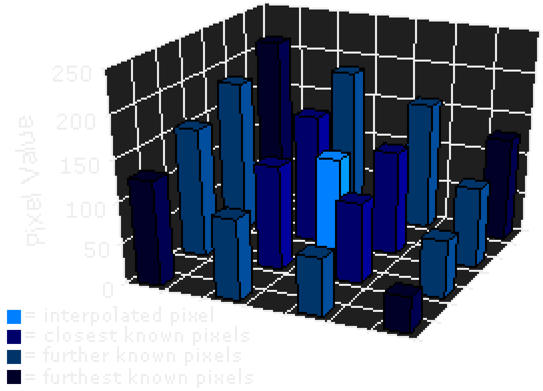
Extra Reference Stuff Follows

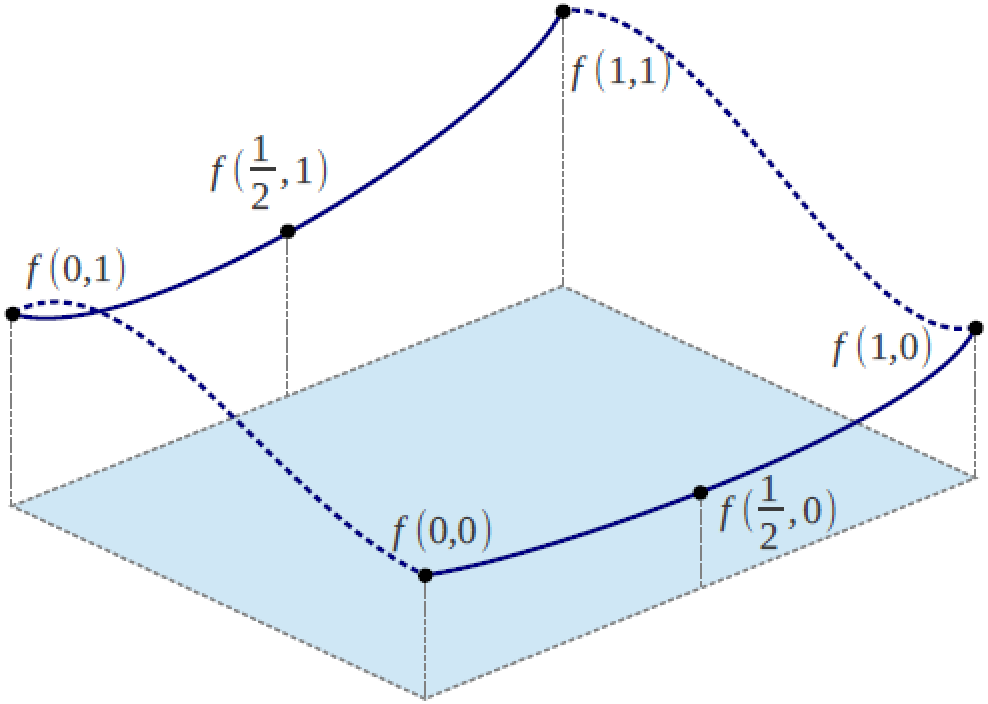


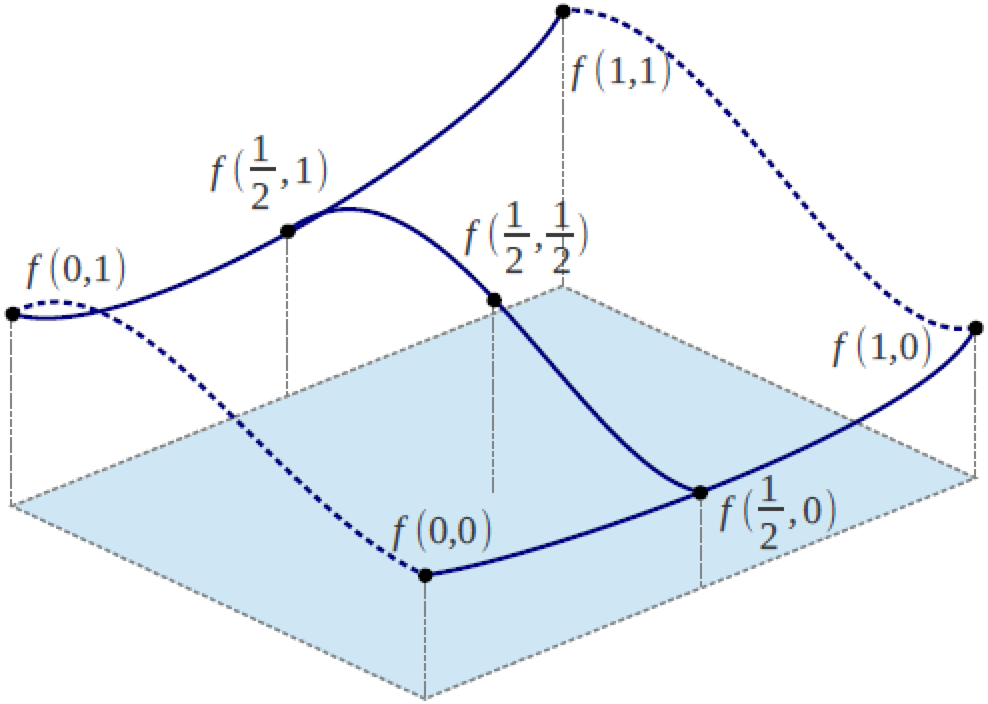
$f(1,1)$
$f(0,1)$
$f(1,0)$
$f(0,0)$
Pixel Value
Average
■ = known pixels
■ = interpolated pixel
$f(1,1)$
$f(0,1)$
$f(1,0)$
$f(0,0)$


$f(1,1)$
$f(\frac{1}{2},1)$
$f(0,1)$
$f(1,0)$
$f(\frac{1}{2},0)$
$f(0,0)$
$f(1,1)$
$f(\frac{1}{2},1)$
$f(0,1)$
$f(\frac{1}{2},\frac{1}{2})$
$f(1,0)$
$f(\frac{1}{2},0)$
$f(0,0)$

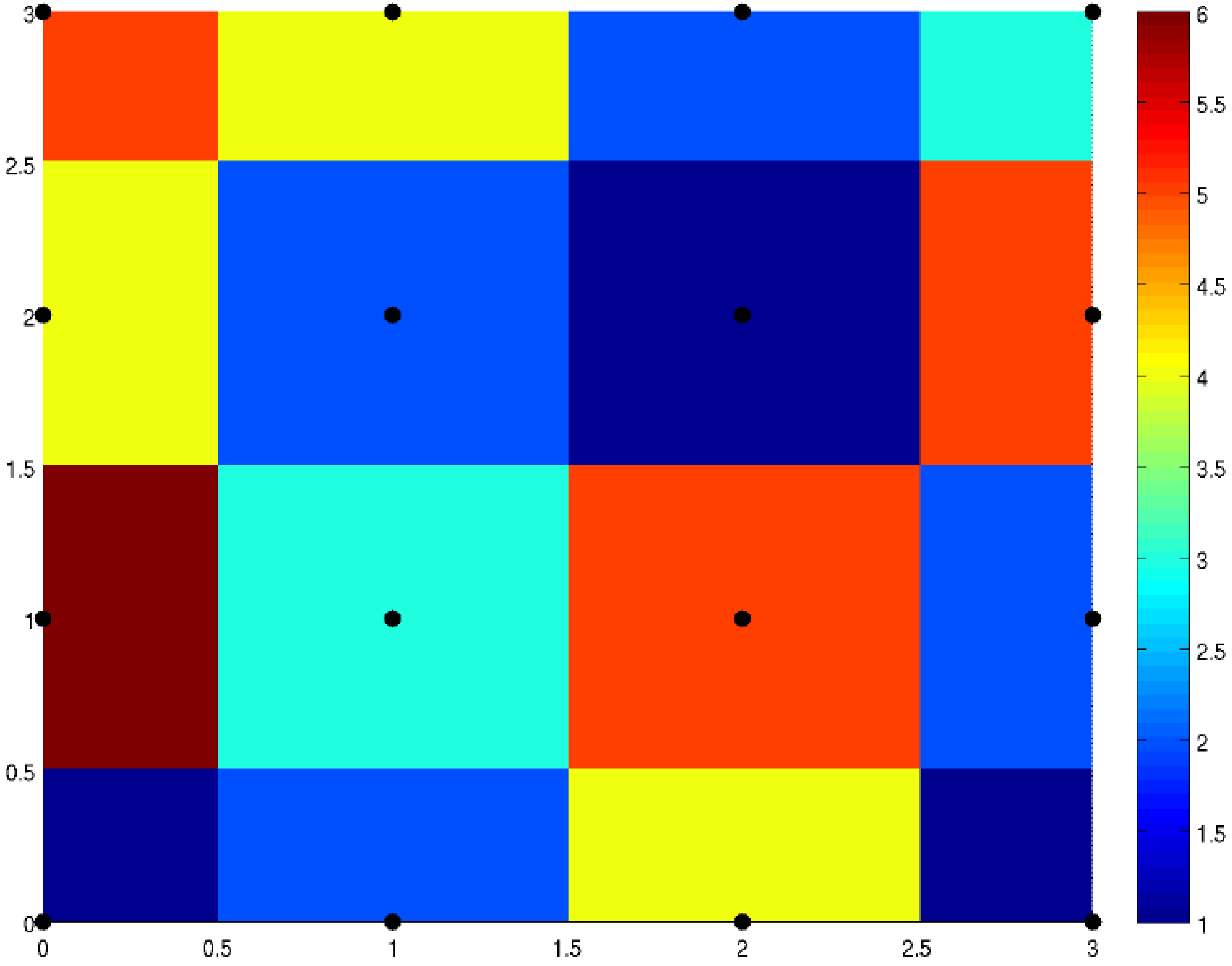

$f(1,1)$
$f(0,1)$
$f(1,0)$
$f(0,0)$


Pixel Value
250
200
150
100
50
0
= interpolated pixel
= closest known pixels
= further known pixels
= furthest known pixels

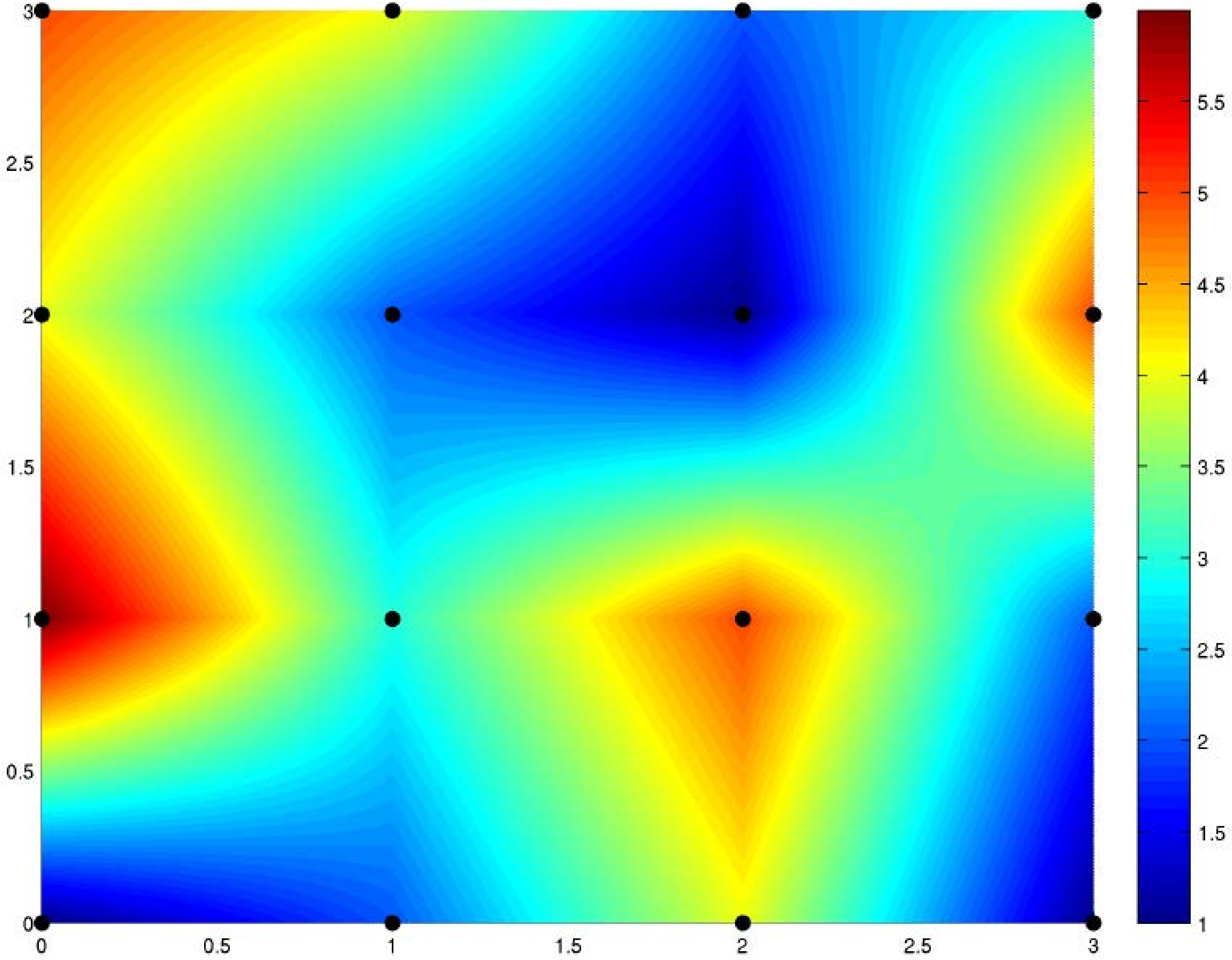

$f(1,1)$
$f(\frac{1}{2},1)$
$f(0,1)$
$f(1,0)$
$f(0,0)$
$f(\frac{1}{2},0)$


$f(1,1)$
$f(\frac{1}{2},1)$
$f(\frac{1}{2},\frac{1}{2})$
$f(0,1)$
$f(1,0)$
$f(\frac{1}{2},0)$
$f(0,0)$

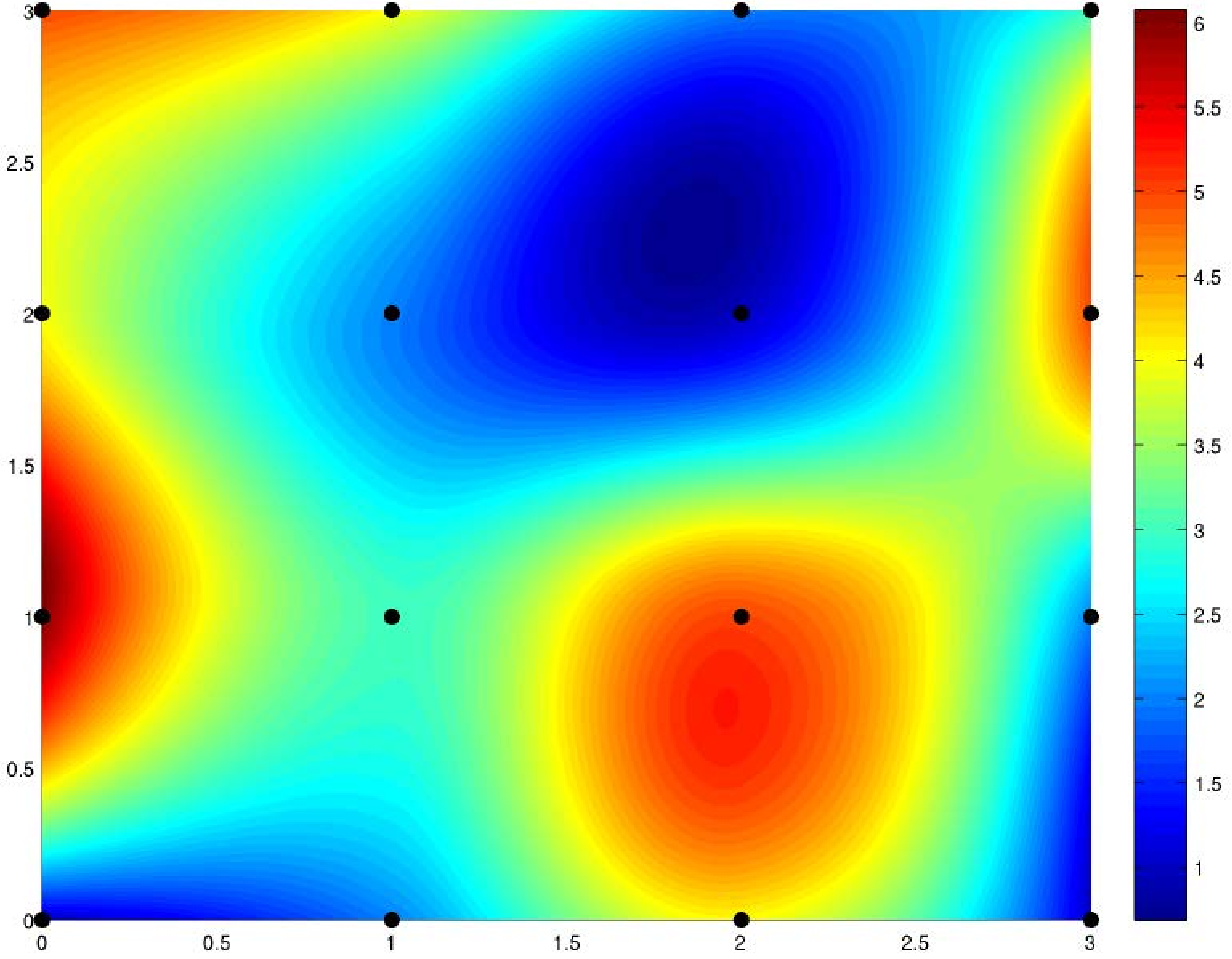
Nearest Nhbr Interp



Bilinear Interp



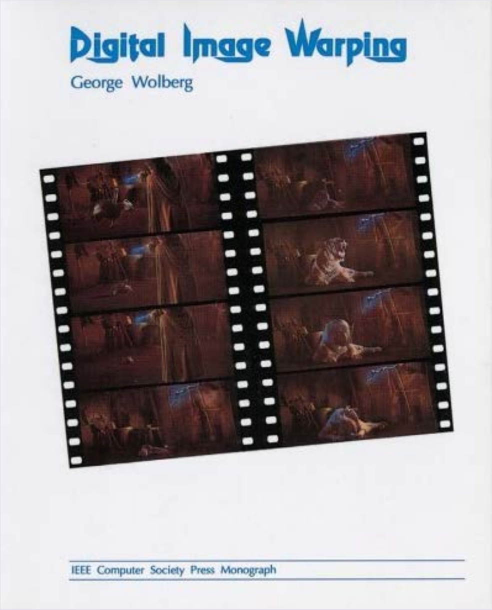
Bicubic Interp



Credits

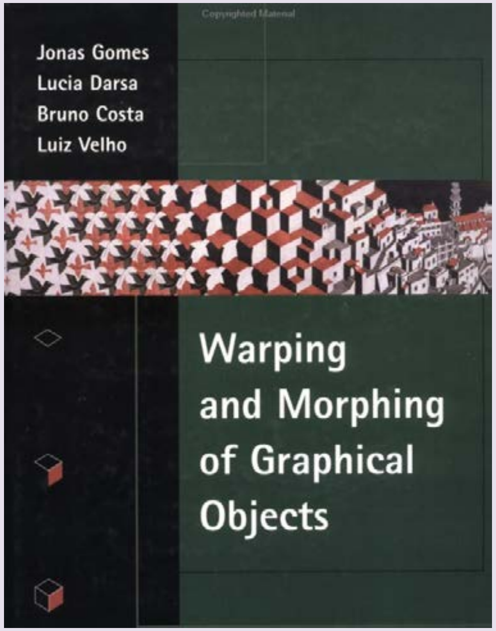
- Much of the content derived/based on slides for use with the book:
  - *Digital Image Processing,* Gonzalez and Woods

- Some layout and presentation style derived/based on presentations by
  - Donald House, Texas A&M University, 1999
  - Bernd Girod, Stanford University, 2007
  - Shreekanth Mandayam, Rowan University, 2009
  - Igor Aizenberg, TAMUT, 2013
  - Xin Li, WVU, 2014
  - George Wolberg, City College of New York, 2015
  - Yao Wang and Zhu Liu, NYU-Poly, 2015
  - Sinisa Todorovic, Oregon State, 2015


Digital Image Warping

George Wolberg

IEEE Computer Society Press Monograph


Jonas Gomes
Lucia Darsa
Bruno Costa
Luiz Velho

Warping and Morphing of Graphical Objects