Image interpolation

A reinterpretation of low-pass filtering

Image Interpolation

- Introduction
  - What is image interpolation? (D-A conversion)
  - Why do we need it?
- Interpolation Techniques
  - 1D zero-order, first-order, third-order
  - 2D = two sequential 1D (divide-and-conquer)
  - Directional(Adaptive) interpolation*
- Interpolation Applications
  - Digital zooming (resolution enhancement)
  - Image inpainting (error concealment)
  - Geometric transformations (where your imagination can fly)

Introduction

- What is image interpolation?
  - An image f(x,y) tells us the intensity values at the integral lattice locations, i.e., when x and y are both integers
  - Image interpolation refers to the "guess" of intensity values at missing locations, i.e., x and y can be arbitrary
  - Note that it is just a guess (Note that all sensors have finite sampling distance)

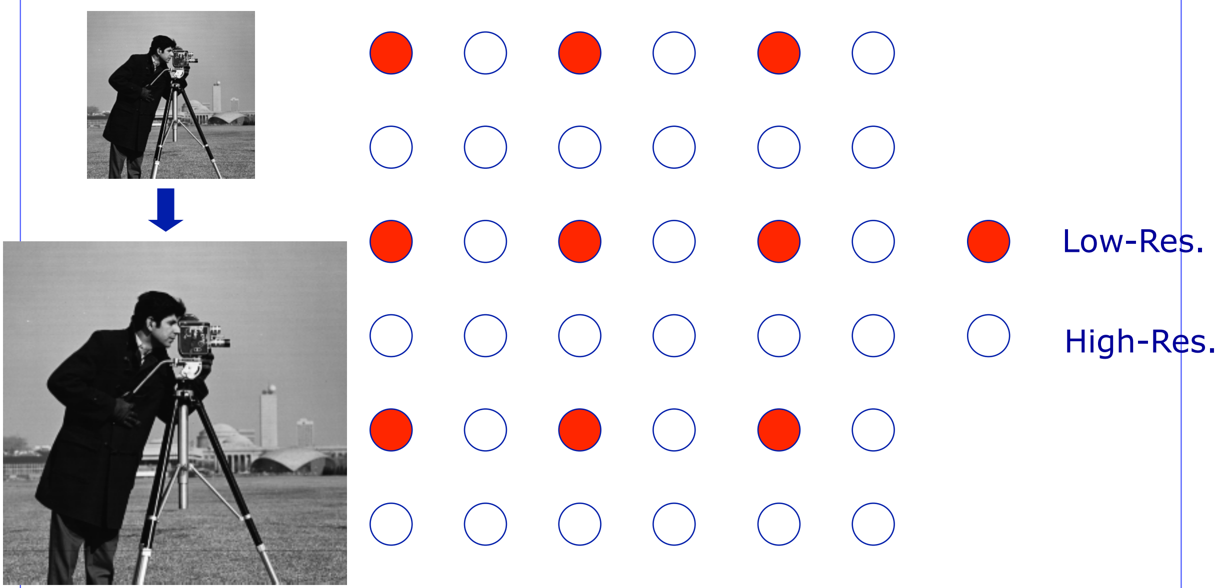
A Sentimental Comment

- Haven't we just learned from discrete sampling (A-D conversion)?
- Yes, image interpolation is about D-A conversion
- Recall the gap between biological vision and artificial vision systems
  - Digital: camera + computer
  - Analog: retina + brain

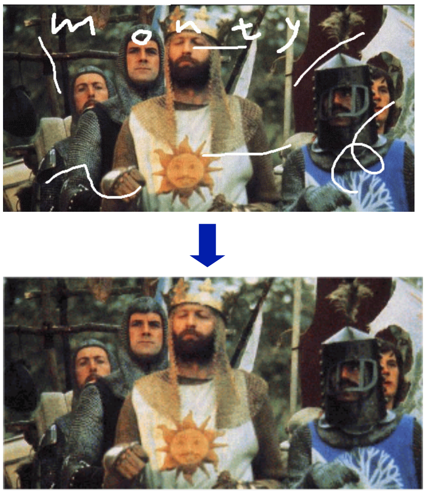
Engineering Motivations

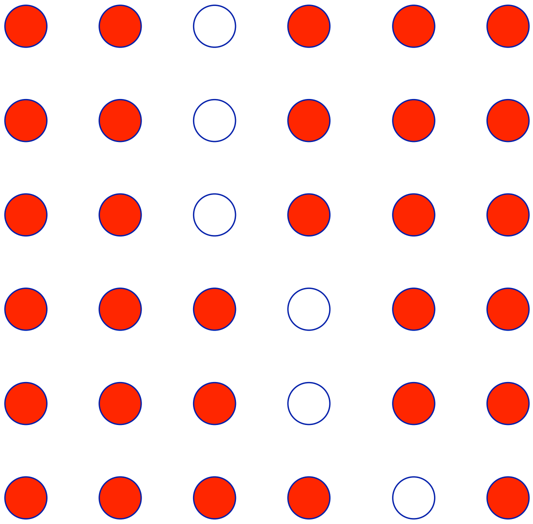
- Why do we need image interpolation?
  - We want BIG images
    - When we see a video clip on a PC, we like to see it in the full screen mode
  - We want GOOD images
    - If some block of an image gets damaged during the transmission, we want to repair it
  - We want COOL images
    - Manipulate images digitally can render fancy artistic effects as we often see in movies

Scenario I: Resolution Enhancement



Scenario II: Image Inpainting





Non-damaged Damaged

Scenario III: Image Warping

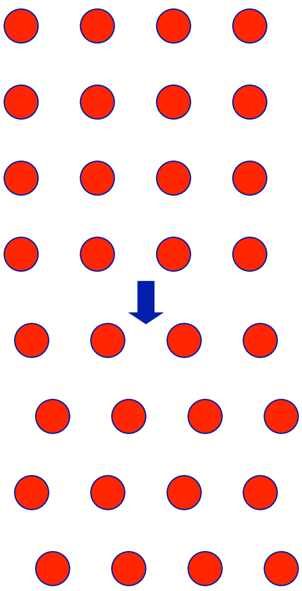


Image Interpolation

- Introduction
  - What is image interpolation?
  - Why do we need it?
- Interpolation Techniques
  - 1D linear interpolation (elementary algebra)
  - 2D = 2 sequential 1D (divide-and-conquer)
  - Directional (adaptive) interpolation
- Interpolation Applications
  - Digital zooming (resolution enhancement)
  - Image inpainting (error concealment)
  - Geometric transformations

Upsampling

- This image is too small for this screen:

- How can we make it 10 times as big?
- Simplest approach:

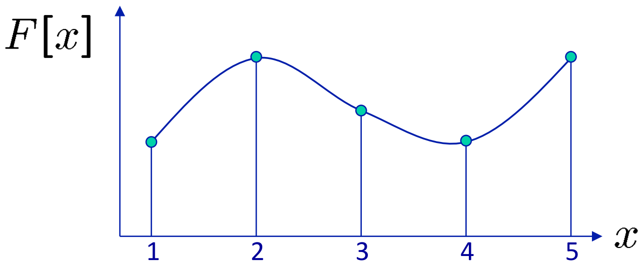
  repeat each row

  and column 10 times

- ("Nearest neighbor

  interpolation")

Image interpolation



d = 1 in this example

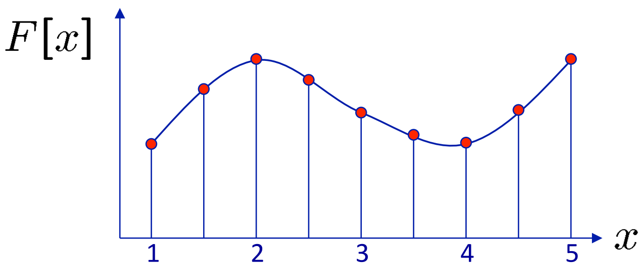
Recall how a digital image is formed

$$F[x,y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Adapted from: S. Seitz

Image interpolation



d = 1 in this example

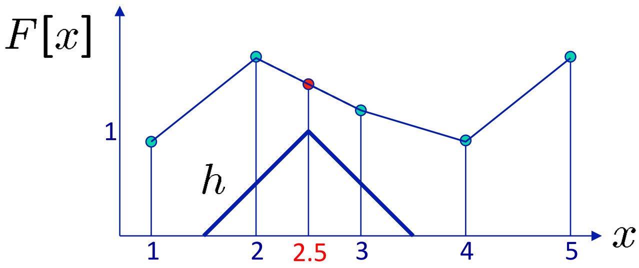
Recall how a digital image is formed

$$F[x,y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Adapted from: S. Seitz

Image interpolation

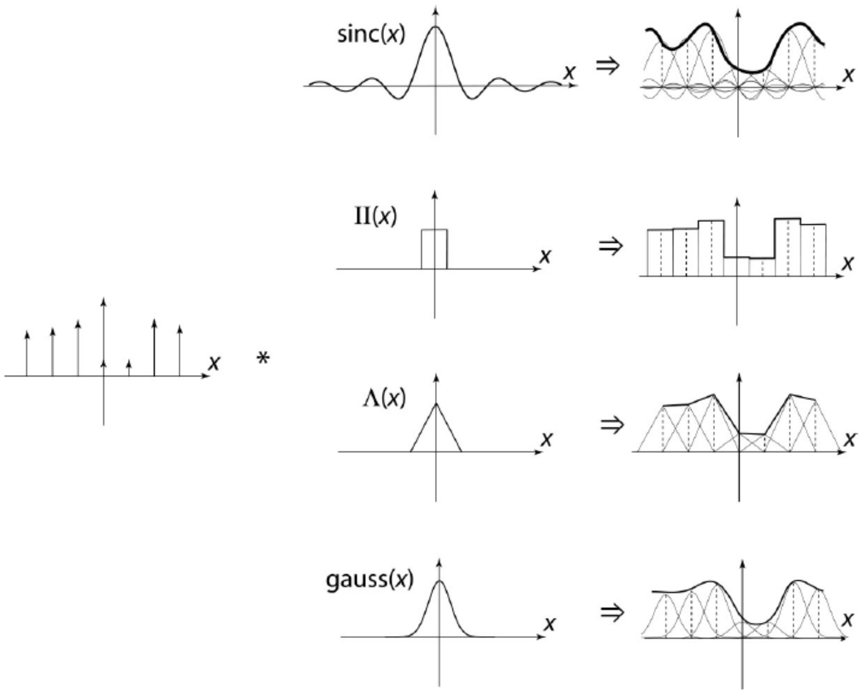


d = 1 in this example

- What if we don't know f ?
  - Guess an approximation: $\tilde{f}$
  - Can be done in a principled way: filtering
  - Convert F to a continuous function:
    $f_F(x) = F(\frac{x}{d})$ when $\frac{x}{d}$ is an integer, 0 otherwise
  - Reconstruct by convolution with a *reconstruction filter, h*
    $\tilde{f} = h * f_F$

Adapted from: S. Seitz

Image interpolation



"Ideal" reconstruction

Nearest-neighbor interpolation

Linear interpolation

Gaussian reconstruction

Source: B. Curless

Ideal reconstruction

3.13 IMAGE INTERPOLATION ALGORITHMS

A digital image $f(n_1, n_2)$ may be thought of as a sampled region of an analog image $f(x, y)$ having continuous coordinates x, y:

$$f(n_1, n_2) = f(x, y)\,|_{x=n_1T_1, y=n_2T_2} \tag{3.13.1}$$

as has already been described in Chapter 1. T_1, T_2 are the sampling intervals along the x, y axes. If the analog image $f(x, y)$ is band-limited:

$$F(\Omega_1, \Omega_2) = 0 \text{ for } |\,\Omega_1\,| \geq \tfrac{\pi}{T_1}, |\,\Omega_2\,| \geq \tfrac{\pi}{T_2} \tag{3.13.2}$$

and the sampling frequencies are above the Nyquist frequencies, it can be recovered from the sampled image $f(n_1, n_2)$ by using the sinc interpolation formula [DUD84]:

$$f(x, y) = \sum^{\infty} \sum^{\infty} f(n_1, n_2) \frac{\sin \frac{\pi}{T_1}(x - n_1T_1)}{\frac{\pi}{T_1}(x - n_1T_1)} \frac{\sin \frac{\pi}{T_2}(y - n_2T_2)}{\frac{\pi}{T_2}(y - n_2T_2)} \tag{3.13.3}$$

Ideal reconstruction

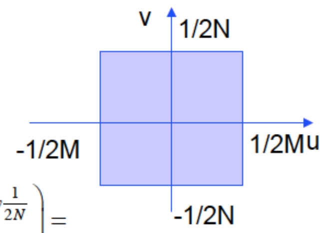
$$h(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} H(u,v)e^{j2\pi(ux+vy)}dudv = \int_{-\frac{1}{2N}}^{\frac{1}{2N}}\int_{\frac{-1}{2M}}^{\frac{1}{2M}} MNe^{j2\pi(ux+vy)}dudv$$

$$= \int_{\frac{-1}{2M}}^{\frac{1}{2M}} Me^{j2\pi ux}du \int_{\frac{-1}{2N}}^{\frac{1}{2N}} Ne^{j2\pi vy}dv$$

$$= MN\frac{1}{j2\pi x}\left(e^{j2\pi x\frac{1}{2M}} - e^{-j2\pi y\frac{1}{2M}}\right) \times \frac{1}{j2\pi y}\left(e^{j2\pi y\frac{1}{2N}} - e^{-j2\pi y\frac{1}{2N}}\right) =$$

$$= \frac{1}{\frac{\pi}{M}x}\frac{1}{2j}\left(e^{j\pi x\frac{1}{M}} - e^{-j\pi x\frac{1}{M}}\right) \times \frac{1}{\frac{\pi}{N}y}\frac{1}{2j}\left(e^{j\pi y\frac{1}{N}} - e^{-j\pi y\frac{1}{N}}\right) =$$

$$= \frac{\sin\left(\frac{\pi}{M}x\right)}{\frac{\pi}{M}x} \times \frac{\sin\left(\frac{\pi}{N}y\right)}{\frac{\pi}{N}y}$$



$$\sin(x) = \frac{1}{2j}\left(e^{jx} - e^{-jx}\right)$$

Ideal reconstruction

- The ideal reconstruction filter is a square window in the F-domain and a sinc function in the signal domain
- Its implementation is unpractical since it is prone to truncation artifacts and has relatively high computational complexity.
- Other low-pass filters can be used instead

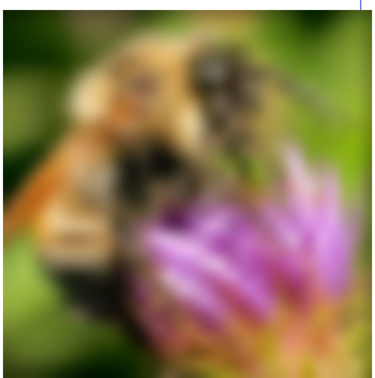
Interpolation: LP filtering

Image interpolation

Original image:  x 10

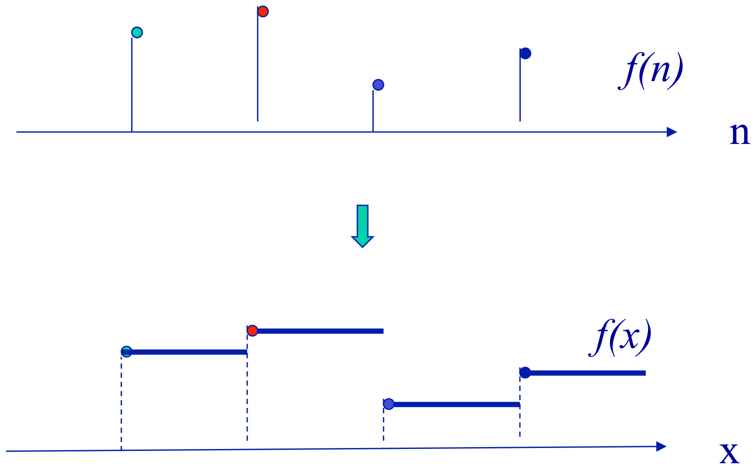
Nearest-neighbor interpolation

Bilinear interpolation

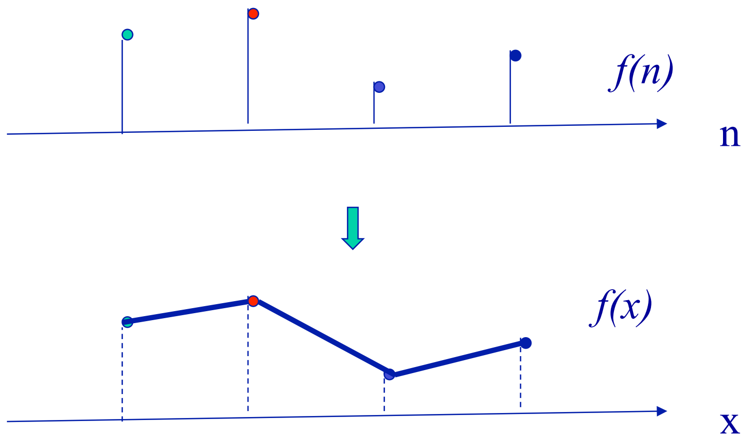


Bicubic interpolation

1D Zero-order (Replication)

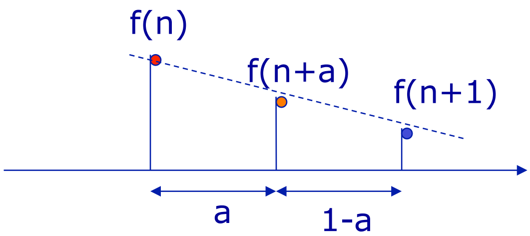


1D First-order Interpolation (Linear)



Linear Interpolation Formula

Heuristic: the closer to a pixel, the higher weight is assigned
Principle: line fitting to polynomial fitting (analytical formula)



$$f(n+a)=(1-a)\times f(n)+a\times f(n+1),\ 0<a<1$$

Note: when a=0.5, we simply have the average of two

Numerical Examples

f(n)=[0,120,180,120,0]

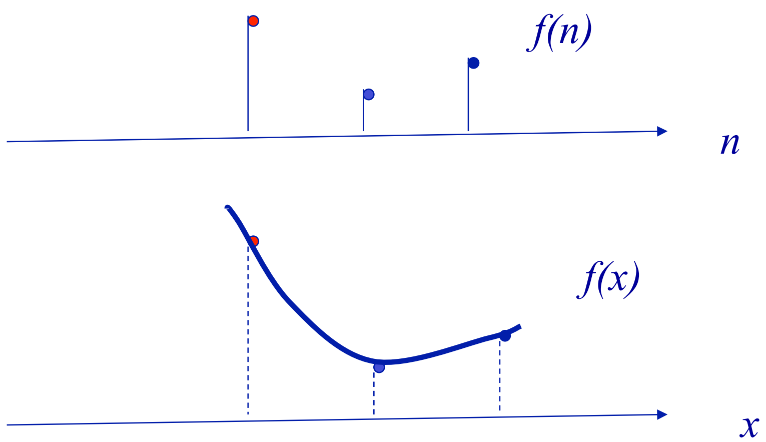
↓ Interpolate at 1/2-pixel

f(x)=[0,60,120,150,180,150,120,60,0], x=n/2

↓ Interpolate at 1/3-pixel

f(x)=[0,20,40,60,80,100,120,130,140,150,160,170,180,…], x=n/6

1D Third-order Interpolation (Cubic)*



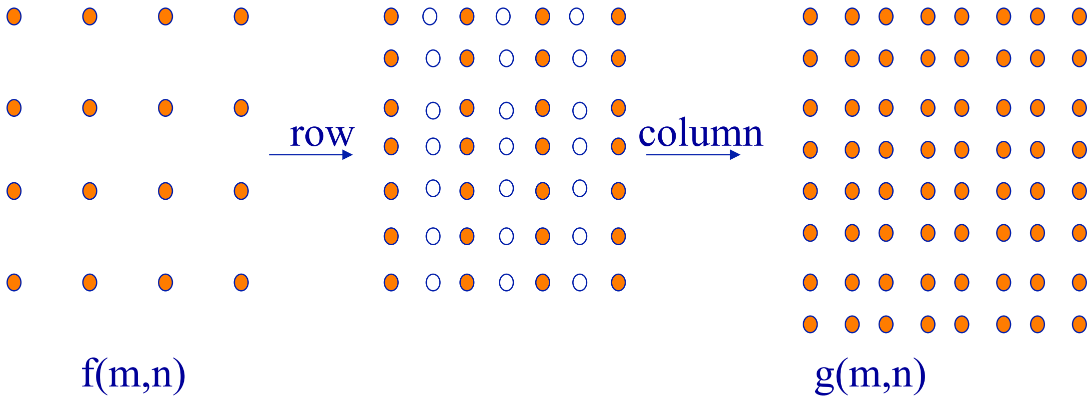
Cubic spline fitting

From 1D to 2D

- Engineers' wisdom: divide and conquer
  - 2D interpolation can be decomposed into two sequential 1D interpolations.
  - The ordering does not matter (row-column = column-row)
  - Such separable implementation is not optimal but enjoys low computational complexity

"If you don't know how to solve a problem, there must be a related but easier problem you know how to solve. See if you can reduce the problem to the easier one." - rephrased from G. Polya's "How to Solve It"

Graphical Interpretation of Interpolation at Half-pel



Numerical Examples

a b

c d

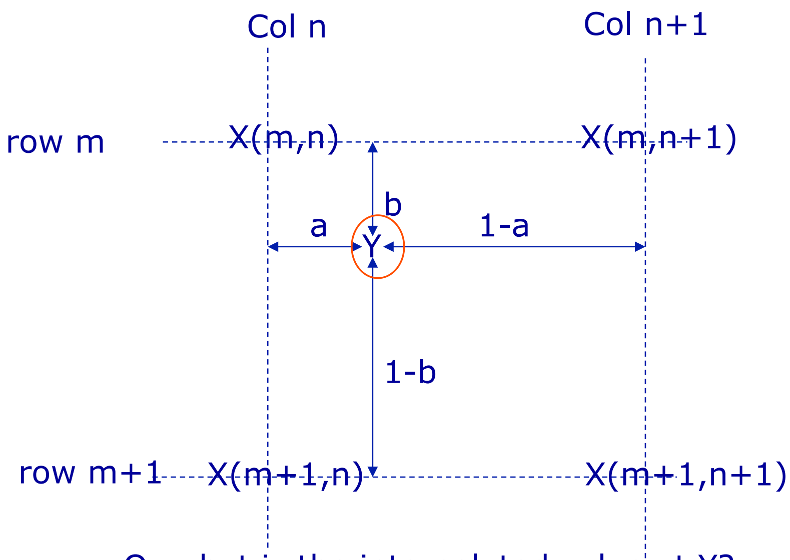
zero-order

a	a	b	b
a	a	b	b
c	c	d	d
c	c	d	d

first-order

a	(a+b)/2	b
(a+c)/2	(a+b+c+d)/4	(b+d)/2
c	(c+d)/2	d

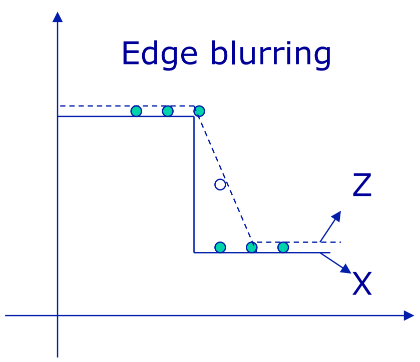
Numerical Examples (Con't)



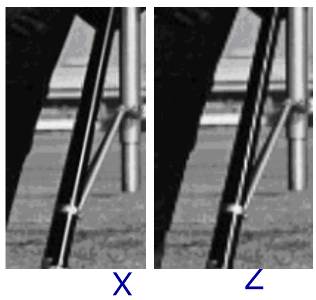
Q: what is the interpolated value at Y?
Ans.: (1-a)(1-b)X(m,n)+(1-a)bX(m+1,n)
+a(1-b)X(m,n+1)+abX(m+1,n+1)

Limitation with bilinear/bicubic

- Edge blurring
- Jagged artifacts

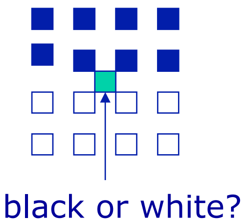


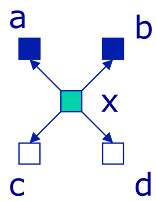
Jagged artifacts



Edge-Sensitive Interpolation

Step 1: interpolate the missing pixels along the diagonal

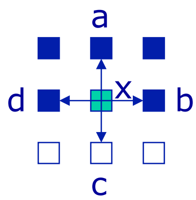




Since $|a-c|=|b-d|$

x has equal probability of being black or white

Step 2: interpolate the other half missing pixels



Since $|a-c|>|b-d|$

$x=(b+d)/2=$black

Image Interpolation

- Introduction
- Interpolation Techniques
  - 1D zero-order, first-order, third-order
  - 2D zero-order, first-order, third-order
  - Directional interpolation*
- Interpolation Applications
  - Digital zooming (resolution enhancement)
  - Image inpainting (error concealment)
  - Geometric transformations (where your imagination can fly)

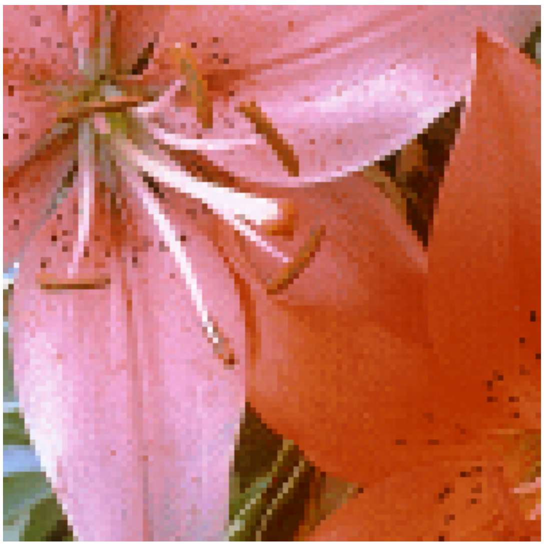
Pixel Replication

low-resolution
image (100×100)

high-resolution
image (400×400)

Bilinear Interpolation

low-resolution
image (100×100)



high-resolution
image (400×400)

Bicubic Interpolation

low-resolution
image (100×100)

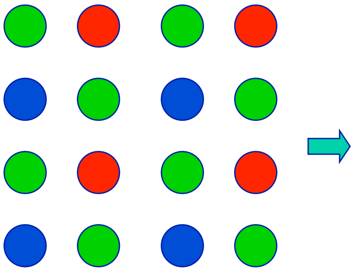
high-resolution
image (400×400)

Edge-Directed Interpolation (Li&Orchard'2000)

low-resolution
image (100×100)

high-resolution
image (400×400)

Image Demosaicing
(Color-Filter-Array Interpolation)



Bayer Pattern

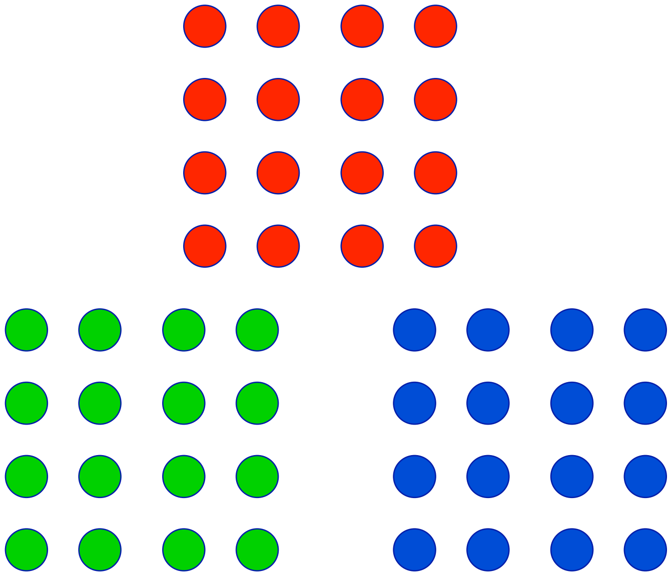
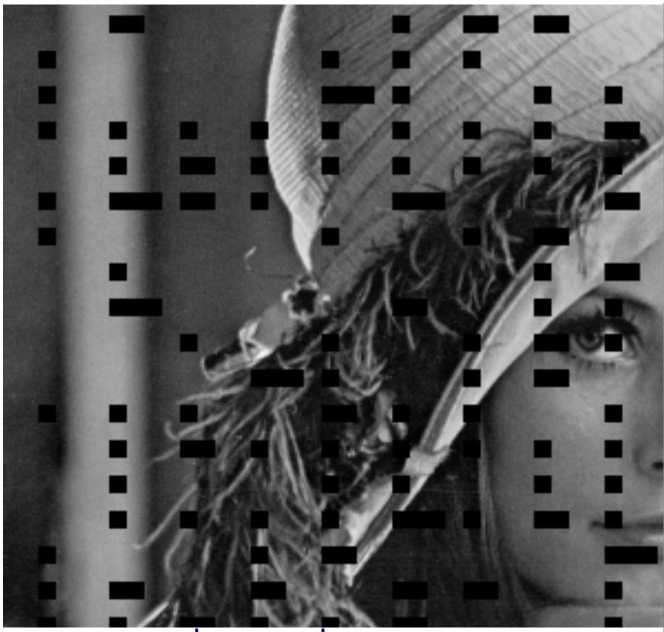


Image Example

Ad-hoc CFA Interpolation

Advanced CFA Interpolation

Error Concealment*



damaged

interpolated

Image Inpainting

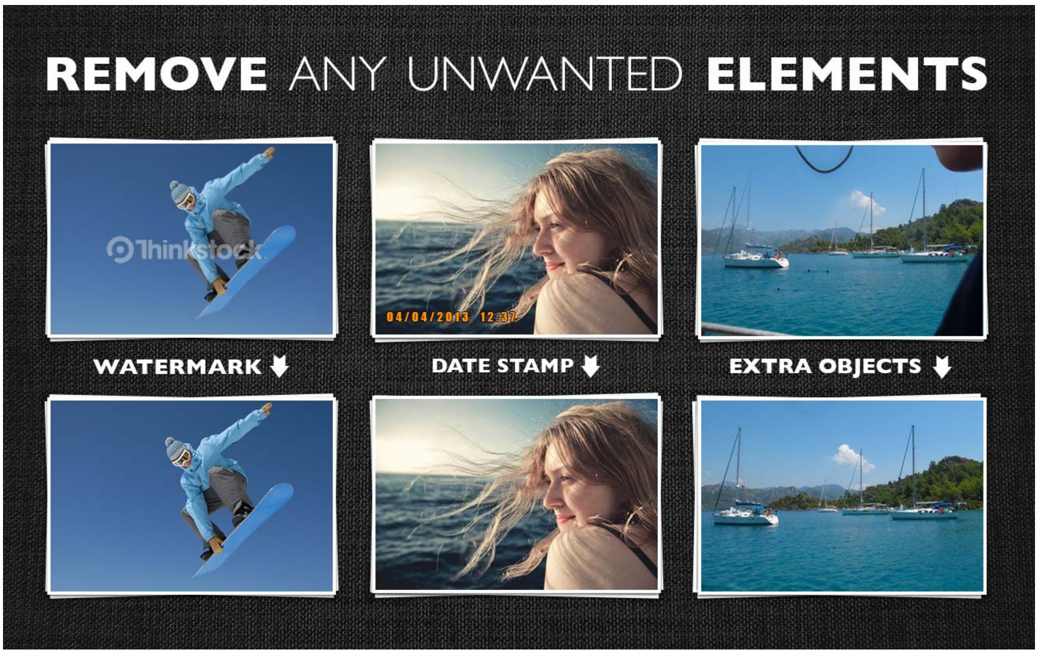
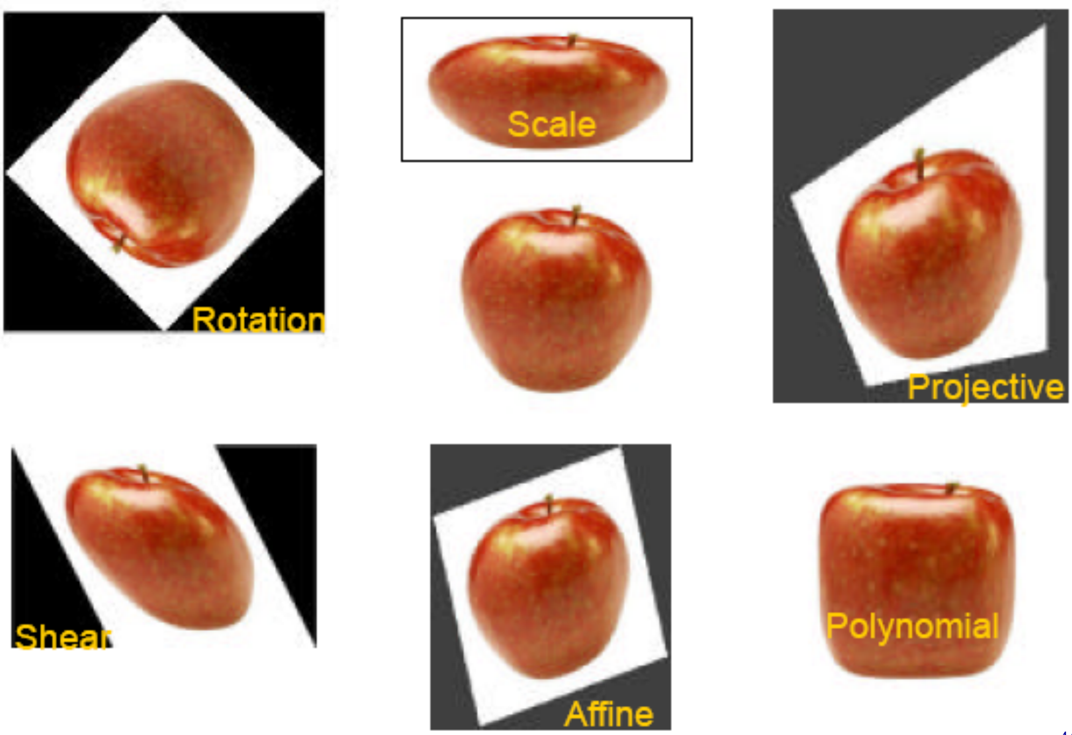


Image mosaicing

Geometric Transformation



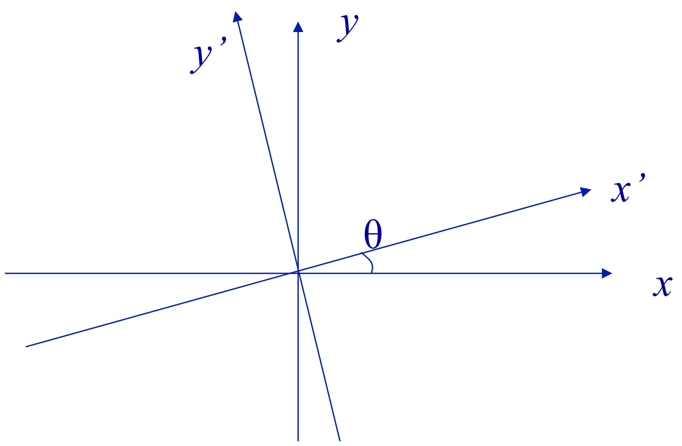
MATLAB functions: griddata, interp2, maketform, imtransform

Basic Principle

- (x,y) → (x',y') is a geometric transformation
- We are given pixel values at (x,y) and want to interpolate the unknown values at (x',y')
- Usually (x',y') are not integers and therefore we can use linear interpolation to guess their values

MATLAB implementation: interp2

Rotation



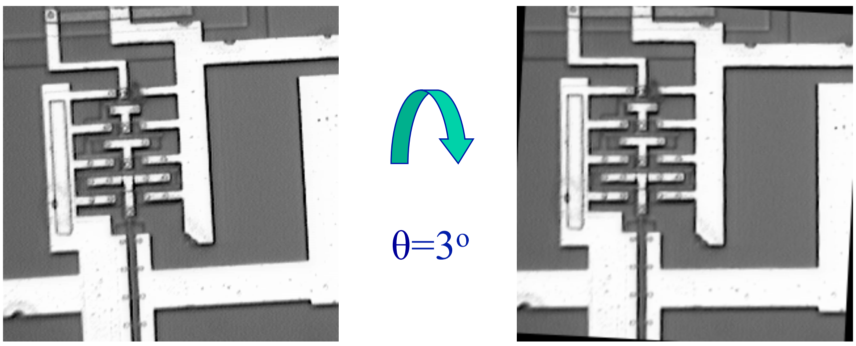
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

MATLAB Example

```
z=imread('cameraman.tif');
% original coordinates
[x,y]=meshgrid(1:256,1:256);

% new coordinates
a=2;
for i=1:256;for j=1:256;
x1(i,j)=a*x(i,j);
y1(i,j=y(i,j)/a;
end;end
% Do the interpolation
z1=interp2(x,y,z,x1,y1,'cubic');
```


Rotation Example



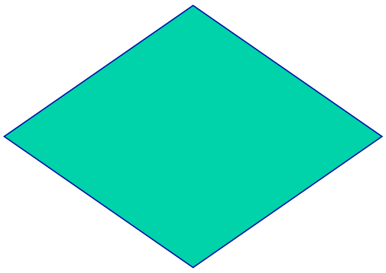
Scale

a=1/2

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & 1/a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Affine Transform

square



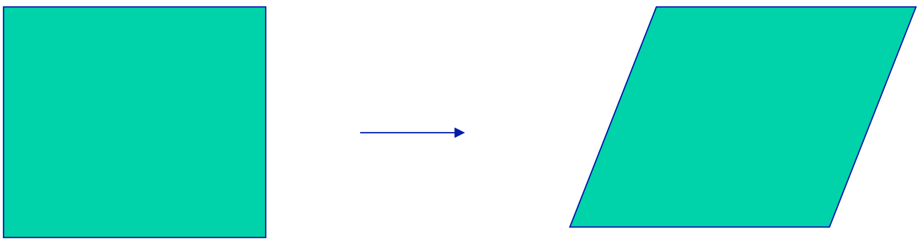
parallelogram

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

Affine Transform Example

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} .5 & 1 \\ .5 & -2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Shear



square

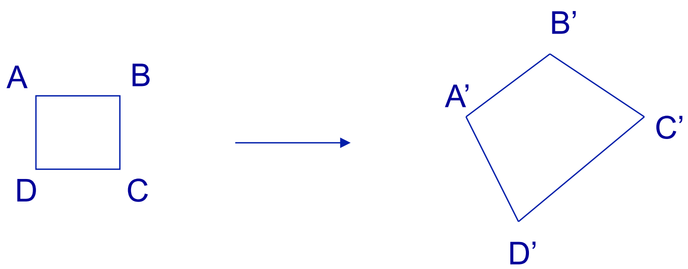
parallelogram

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

Shear Example

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ .5 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Projective Transform

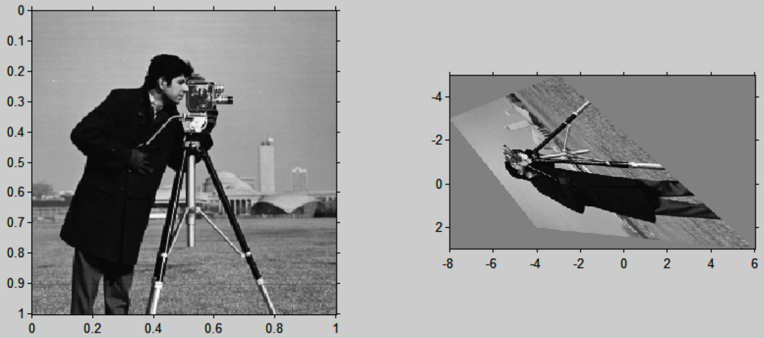


square quadrilateral

$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$

$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$

Projective Transform Example



[0 0; 1 0; 1 1; 0 1] ⟶ [-4 2; -8 -3; -3 -5; 6 3]

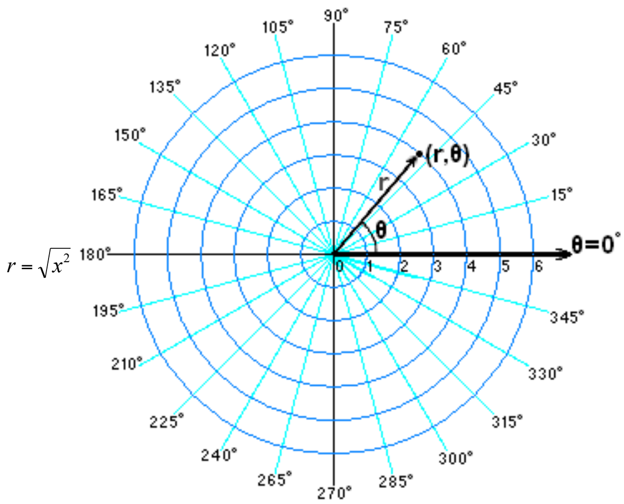
Polar Transform

$$r = \sqrt{x^2 + y^2}$$

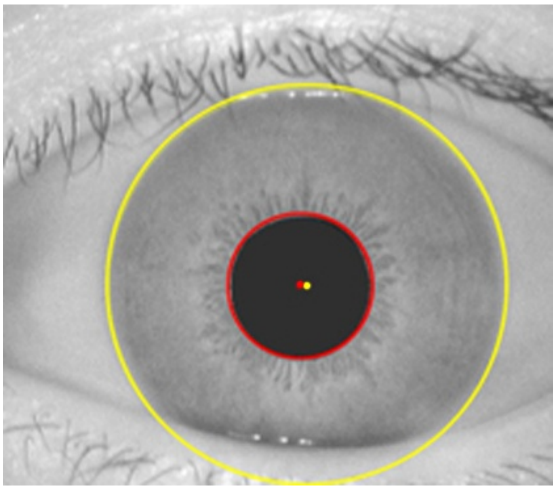
$$\theta = \tan^{-1} \frac{y}{x}$$

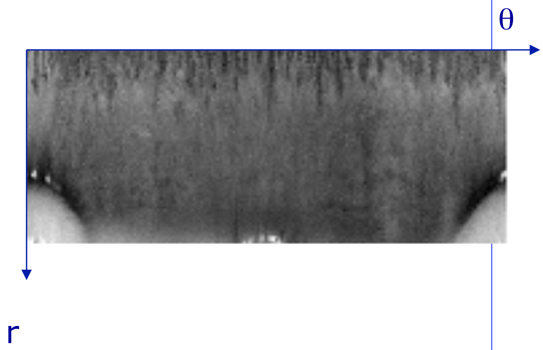
$$x = r \cos \vartheta$$

$$y = r \sin \vartheta$$



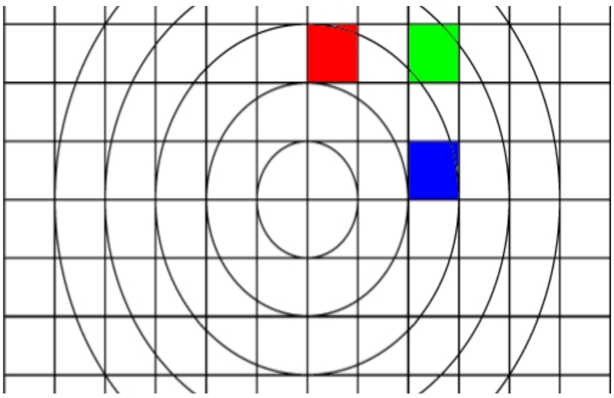
Iris Image Unwrapping

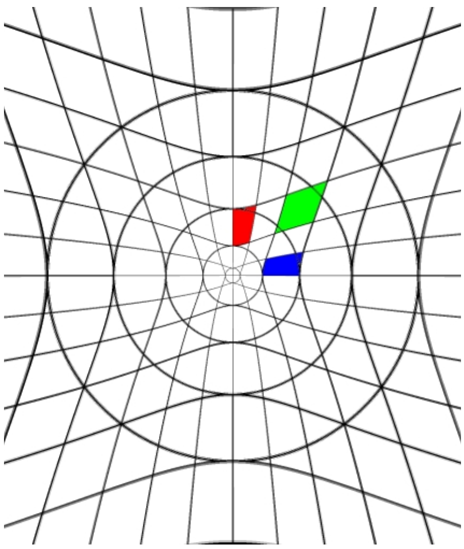




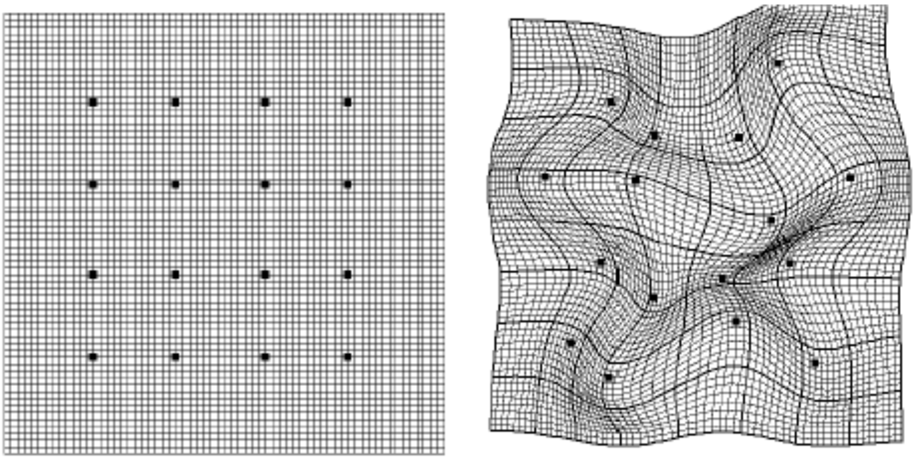
Use Your Imagination

r -> sqrt(r)



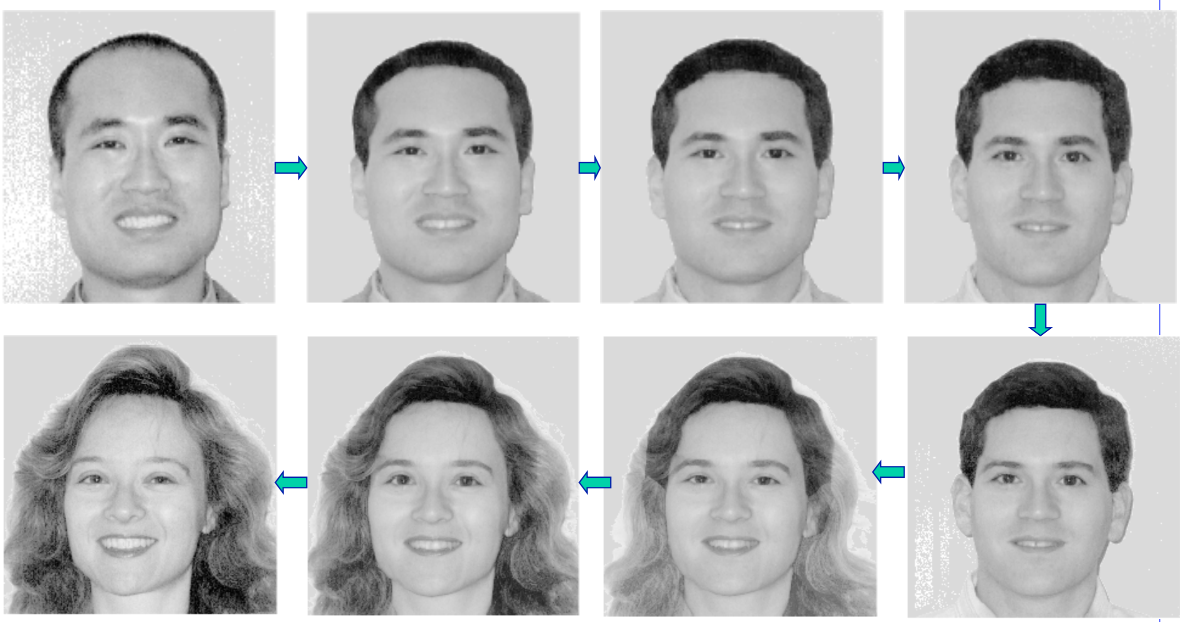


Free Form Deformation



Seung-Yong Lee et al., "Image Metamorphosis Using Snakes and Free-Form Deformations,"*SIGGRAPH'1985,* Pages 439-448

Application into Image Metamorphosis



Summary of Image Interpolation

- A fundamental tool in digital processing of images: bridging the continuous world and the discrete world
- Wide applications from consumer electronics to biomedical imaging
- Remains a hot topic after the IT bubbles break