

# Image and Video Processing

## Image Resizing and Warping

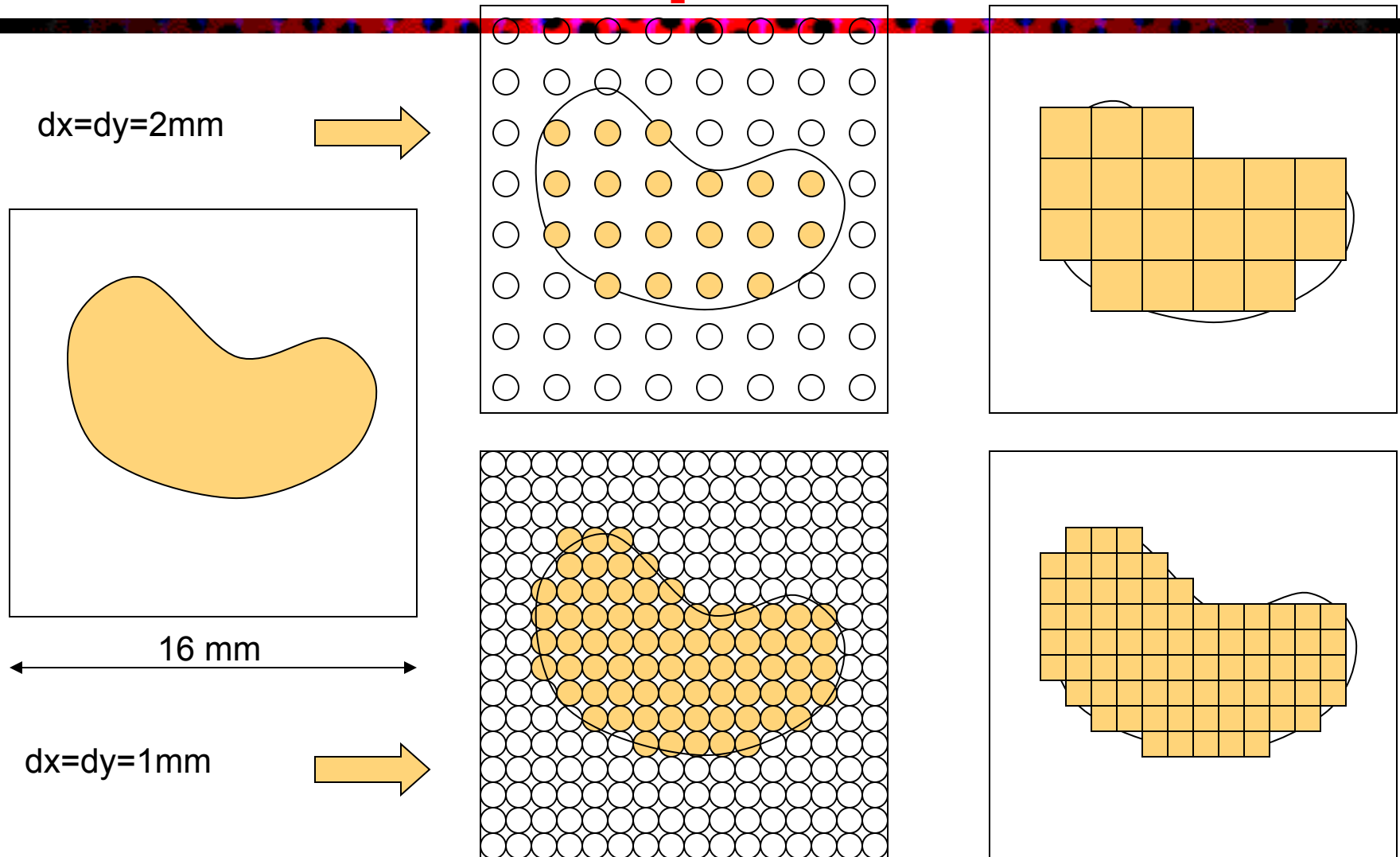
Yao Wang  
Tandon School of Engineering, New York University

# Sampling and Resizing

---

- Nyquist sampling and interpolation theorem
- Common sampling and interpolation filters
- Sampling rate conversion of discrete images (image resizing)

# Illustration of Image Sampling and Interpolation



# Uniform Sampling

- $f(x,y)$  represents the original continuous image,  $f_s(m,n)$  the sampled image, and  $\hat{f}(x,y)$  the reconstructed image.
- Uniform sampling

$$f_s(m,n) = f(m\Delta x, n\Delta y),$$
$$m = 0, \dots, M-1; n = 0, \dots, N-1.$$

- $\Delta x$  and  $\Delta y$  are vertical and horizontal sampling intervals.  $f_{s,x} = 1/\Delta x$ ,  $f_{s,y} = 1/\Delta y$  are vertical and horizontal sampling frequencies.

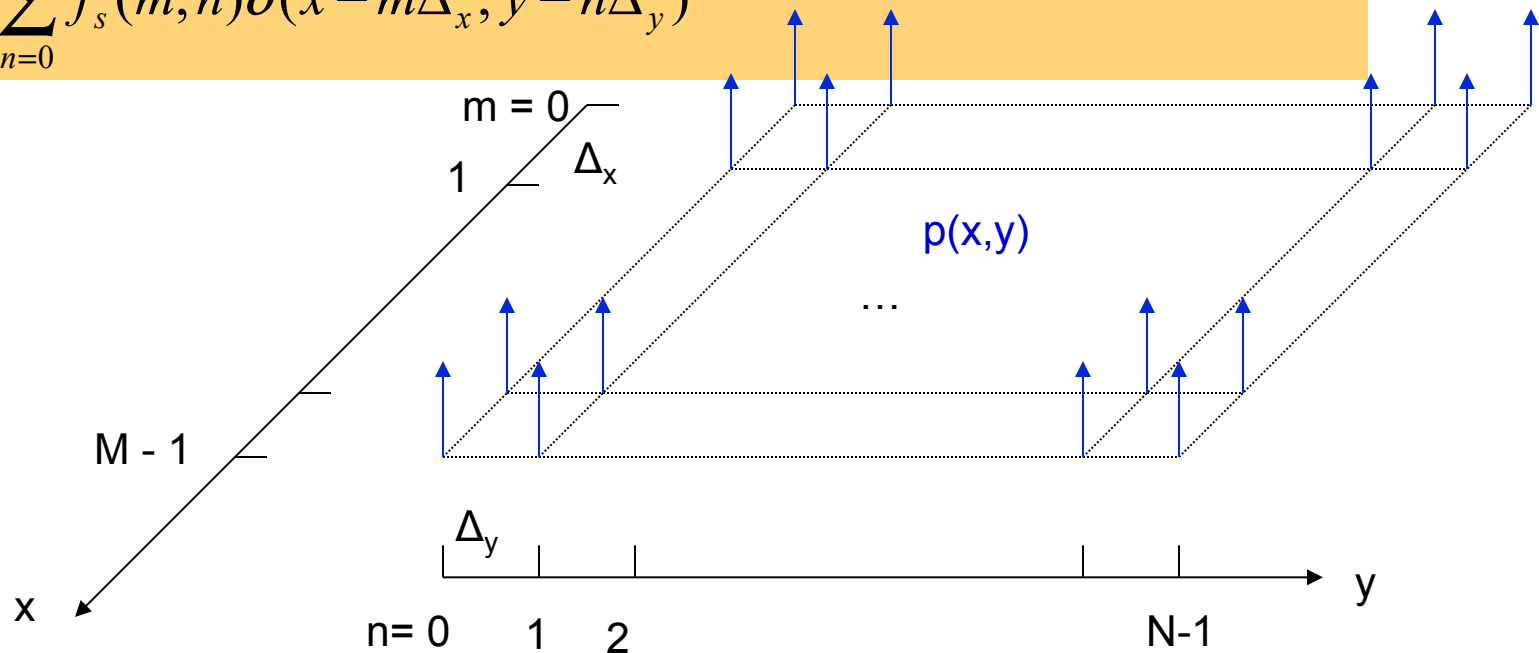
# Image Sampling as Product with Impulse Train

- Periodic impulse sequence

$$p(x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \delta(x - m\Delta_x, y - n\Delta_y)$$

$$\tilde{f}_s(x, y) = f(x, y) \times p(x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m\Delta_x, n\Delta_y) \delta(x - m\Delta_x, y - n\Delta_y)$$

$$= \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f_s(m, n) \delta(x - m\Delta_x, y - n\Delta_y)$$



# Fourier Transform of Impulse Train

- 1D

$$p(t) = \sum_{m,n} \delta(t - n\Delta t) \Leftrightarrow P(u) = \frac{1}{\Delta t} \sum_n \delta(u - nf_s)$$

$$\text{where } f_s = \frac{1}{\Delta t}$$

- 2D

$$p(x, y) = \sum_{m,n} \delta(x - m\Delta x, y - n\Delta y) \Leftrightarrow P(u, v) = \frac{1}{\Delta x \Delta y} \sum_{m,n} \delta(u - mf_{s,x}, v - nf_{s,y})$$

$$\text{where } f_{s,x} = \frac{1}{\Delta x}, f_{s,y} = \frac{1}{\Delta y}$$

# Frequency Domain Interpretation of Sampling

- Sampling is equivalent to multiplication of the original signal with a sampling pulse sequence.

$$f_s(x, y) = f(x, y)p(x, y)$$

$$\text{where } p(x, y) = \sum_{m,n} \delta(x - m\Delta x, y - n\Delta y)$$

- In frequency domain

$$F_s(u, v) = F(u, v) * P(u, v)$$

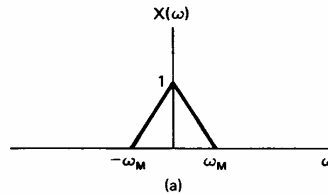
$$P(u, v) = \frac{1}{\Delta x \Delta y} \sum_{m,n} \delta(u - mf_{s,x}, v - nf_{s,y}) \Rightarrow F_s(u, v) = \frac{1}{\Delta x \Delta y} \sum_{m,n} F(u - mf_{s,x}, v - nf_{s,y})$$

$$\text{where } f_{s,x} = \frac{1}{\Delta x}, f_{s,y} = \frac{1}{\Delta y}$$

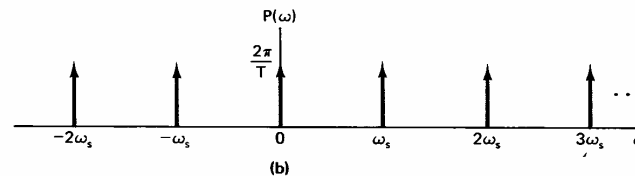
# Frequency Domain

## Interpretation of Sampling in 1D

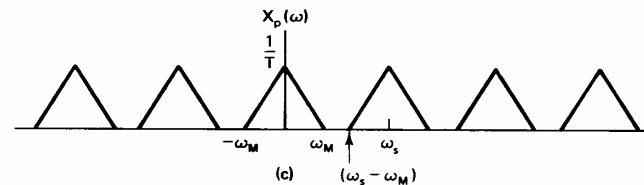
Original signal



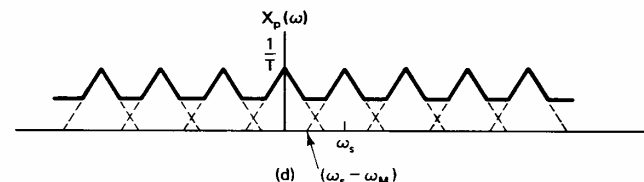
Sampling  
impulse train



Sampled signal  
 $f_s > 2f_m$



Sampled signal  
 $f_s < 2f_m$   
(Aliasing effect)



The spectrum of the sampled signal includes the original spectrum and its aliases (copies) shifted to  $k f_s$ ,  $k = \pm 1, 2, 3, \dots$

The reconstructed signal from samples has the frequency components upto  $f_s/2$ .

*When  $f_s < 2f_m$ , aliasing occur.*



# Sampling of 1D Sinusoid Signals

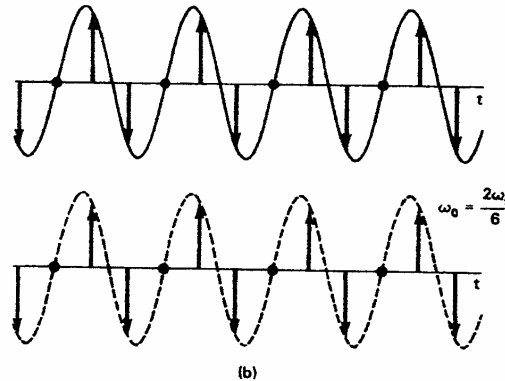
Sampling above

Nyquist rate

$$w_s = 3w_m > w_{s0}$$

Reconstructed

= original



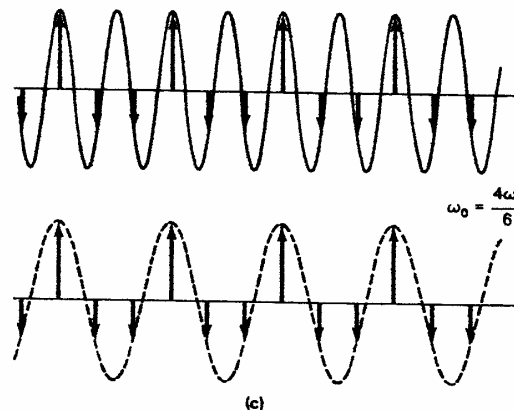
Sampling under

Nyquist rate

$$w_s = 1.5w_m < w_{s0}$$

Reconstructed

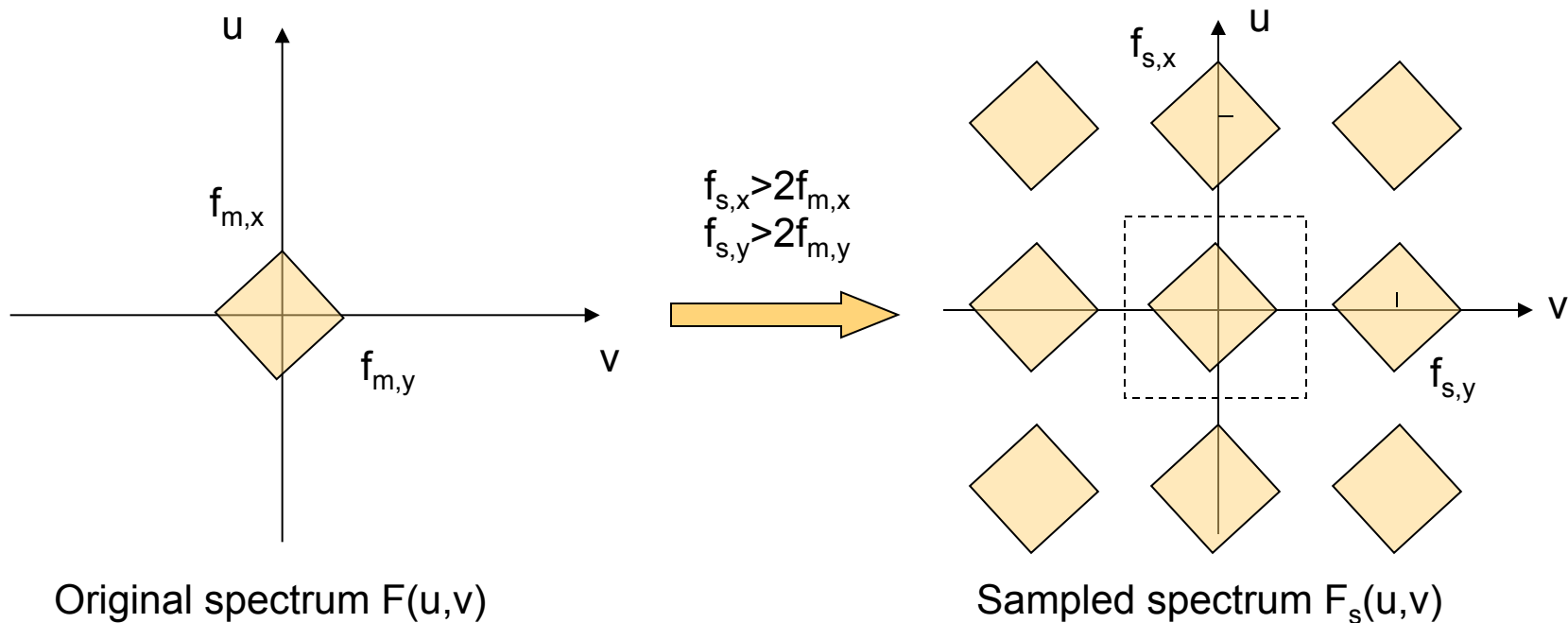
!= original



Aliasing: The reconstructed sinusoid has a lower frequency than the original!

# Frequency Domain Interpretation of Sampling in 2D

- The sampled signal contains replicas of the original spectrum shifted by multiples of sampling frequencies.



# Nyquist Sampling and Reconstruction Theorem

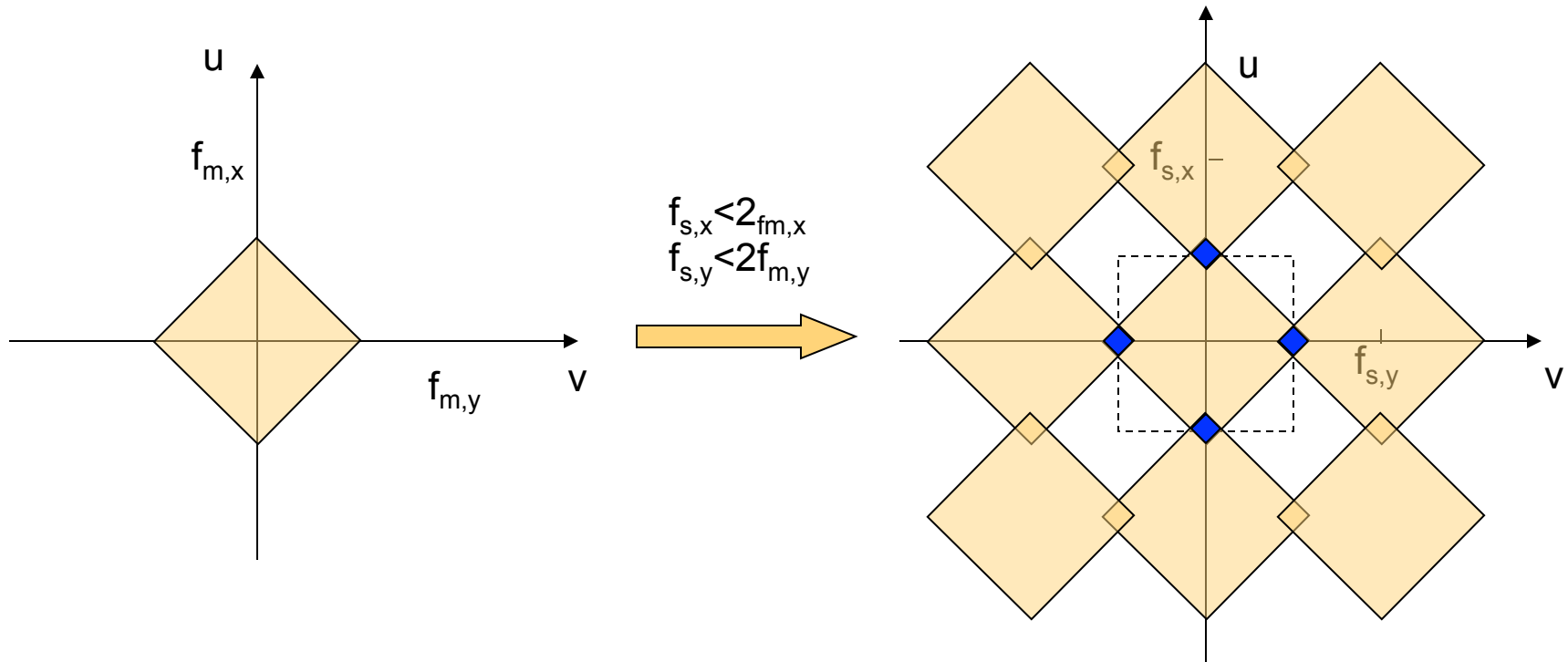
- A band-limited image with highest frequencies at  $f_{m,x}$ ,  $f_{m,y}$  can be reconstructed perfectly from its samples, provided that the sampling frequencies satisfy:  $f_{s,x} > 2f_{m,x}$ ,  $f_{s,y} > 2f_{m,y}$
- The reconstruction can be accomplished by the ideal low-pass filter with cutoff frequency at  $f_{c,x} = f_{s,x}/2$ ,  $f_{c,y} = f_{s,y}/2$ , with magnitude  $\Delta x \Delta y$ .

$$H(u, v) = \begin{cases} \Delta x \Delta y & |u| \leq \frac{f_{s,x}}{2}, |v| \leq \frac{f_{s,y}}{2} \\ 0 & \text{otherwise} \end{cases} \Leftrightarrow h(x, y) = \frac{\sin \pi f_{s,x} x}{\pi f_{s,x} x} \cdot \frac{\sin \pi f_{s,y} y}{\pi f_{s,y} y}$$

- The interpolated image

$$\hat{f}(x, y) = \sum_m \sum_n f_s(m, n) \frac{\sin \pi f_{s,x} (x - m\Delta x)}{\pi f_{s,x} (x - m\Delta x)} \frac{\sin \pi f_{s,y} (y - m\Delta y)}{\pi f_{s,y} (y - m\Delta y)}$$

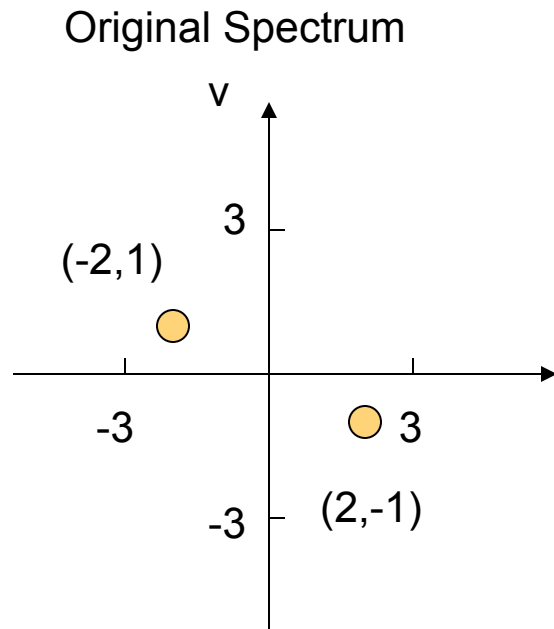
# What Happens When You Under-sample?



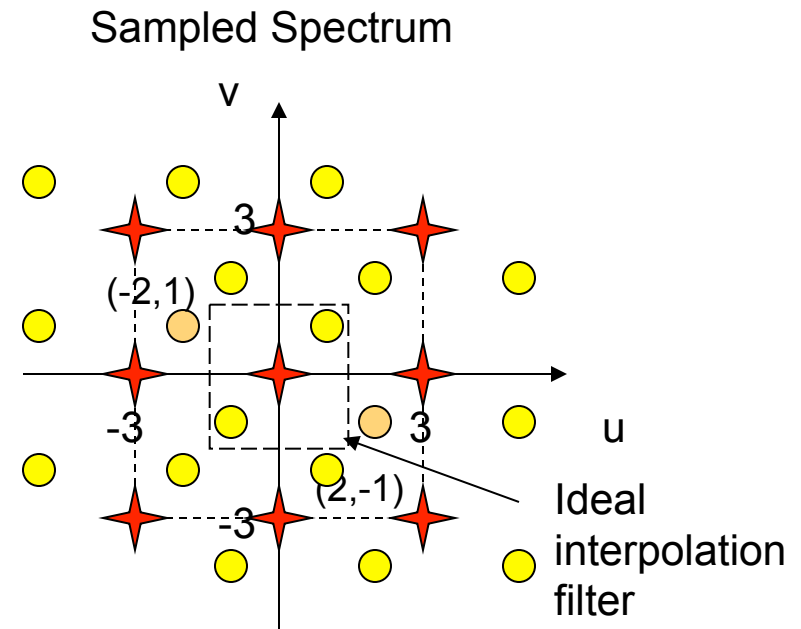
# Sampling a Sinusoidal Signal

$$f(x, y) = \cos(4\pi x - 2\pi y) \Leftrightarrow F(u, v) = \frac{1}{2} [\delta(u - 2, v + 1) + \delta(u + 2, v - 1)]$$

Sampled at  $\Delta x = \Delta y = 1/3$   $f_{s,x} = f_{s,y} = 3$



Original pulse



Replicated pulse

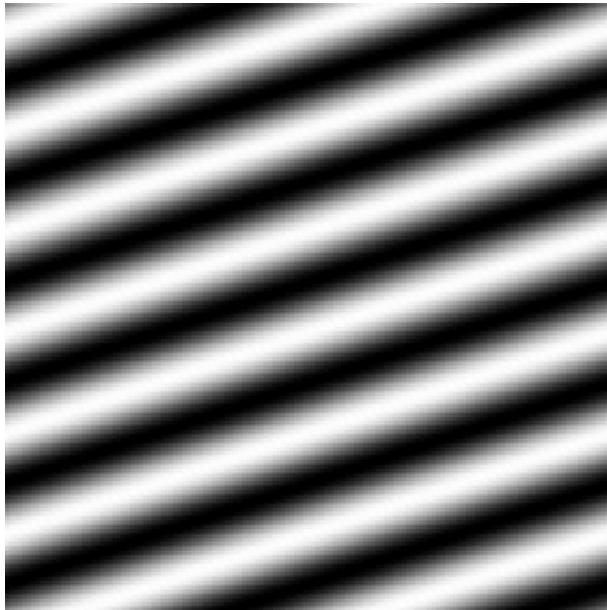
Replication center

$$\hat{f}(x, y) = \cos(2\pi x + 2\pi y)$$

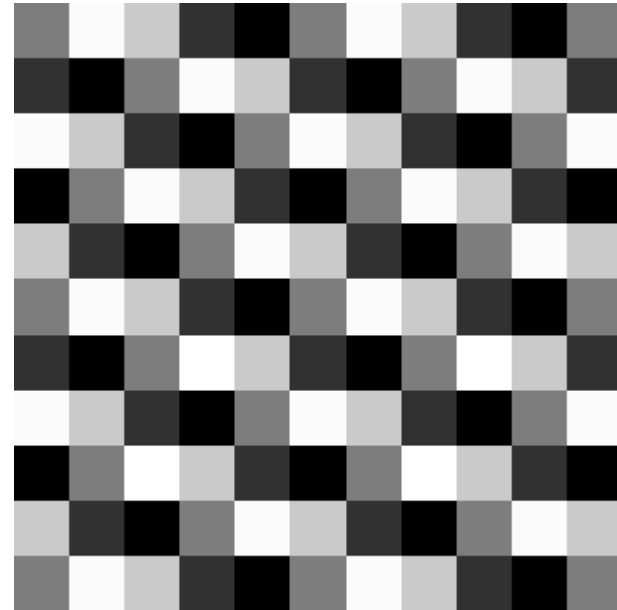
Replication center

# Sampling in 2D:

## Sampling a 2D Sinusoidal Pattern



$f(x,y)=\sin(2*\pi*(3x+y))$   
Sampling:  $dx=0.01, dy=0.01$   
Satisfying Nyquist rate  
 $f_{x,max}=3, f_{y,max}=1$   
 $f_{s,x}=100>6, f_{s,y}=100>2$

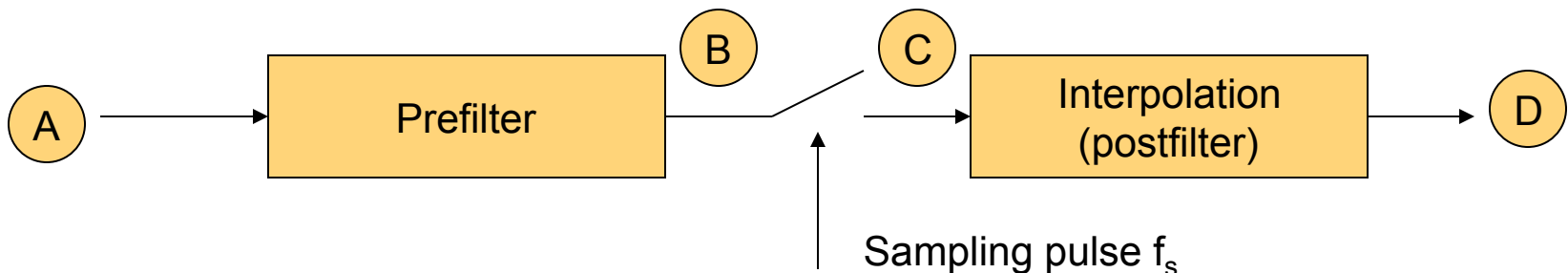


$f(x,y)=\sin(2*\pi*(3x+y))$   
Sampling:  $dx=0.2, dy=0.2$   
(Displayed with pixel replication)  
Sampling at a rate lower than Nyquist rate

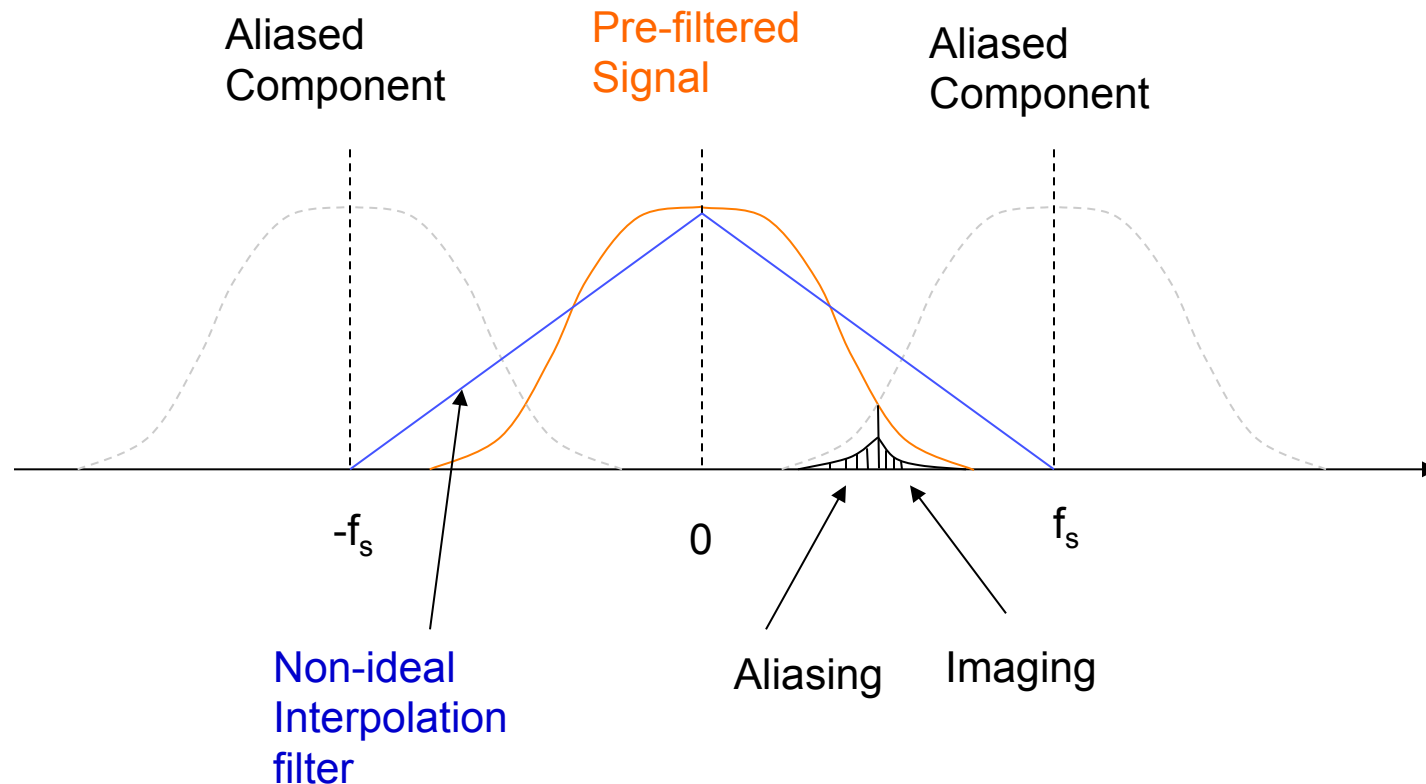
Aliasing causes changes in both frequency and direction of line patterns!

# Applying Nyquist Theorem

- Two issues
  - The signals are not bandlimited.
    - A bandlimiting filter with cutoff frequency  $f_c = f_s/2$  needs to be applied before sampling. This is called *prefilter* or *sampling filter*.
  - The sinc filter is not realizable.
    - Shorter, finite length filters are usually used in practice for both prefilter and interpolation filter.
- A general paradigm



# Non-ideal Sampling and Interpolation



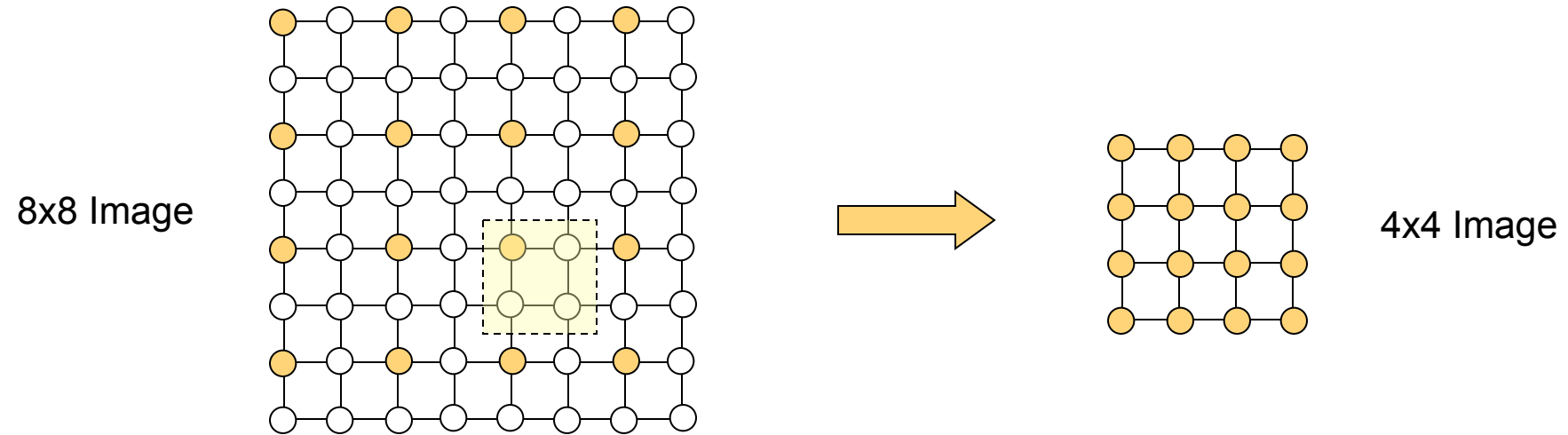
Non ideal prefiltering causes **Aliasing**  
Non ideal interpolation filter causes **Imaging**



# From Sampling to Resizing

- Image resizing:
  - Enlarge or reduce the image size (number of pixels) of sampled pictures
  - Equivalent to
    - First reconstruct the continuous image from samples
    - Then Resample the image at a different sampling rate
  - Can be done w/o reconstructing the continuous image explicitly
- Image down-sampling (resample at a lower rate)
- Image up-sampling (resample at a higher rate)

# Down Sampling by a Factor of Two



- Without Pre-filtering (simple approach)

$$f_d(m, n) = f(2m, 2n)$$

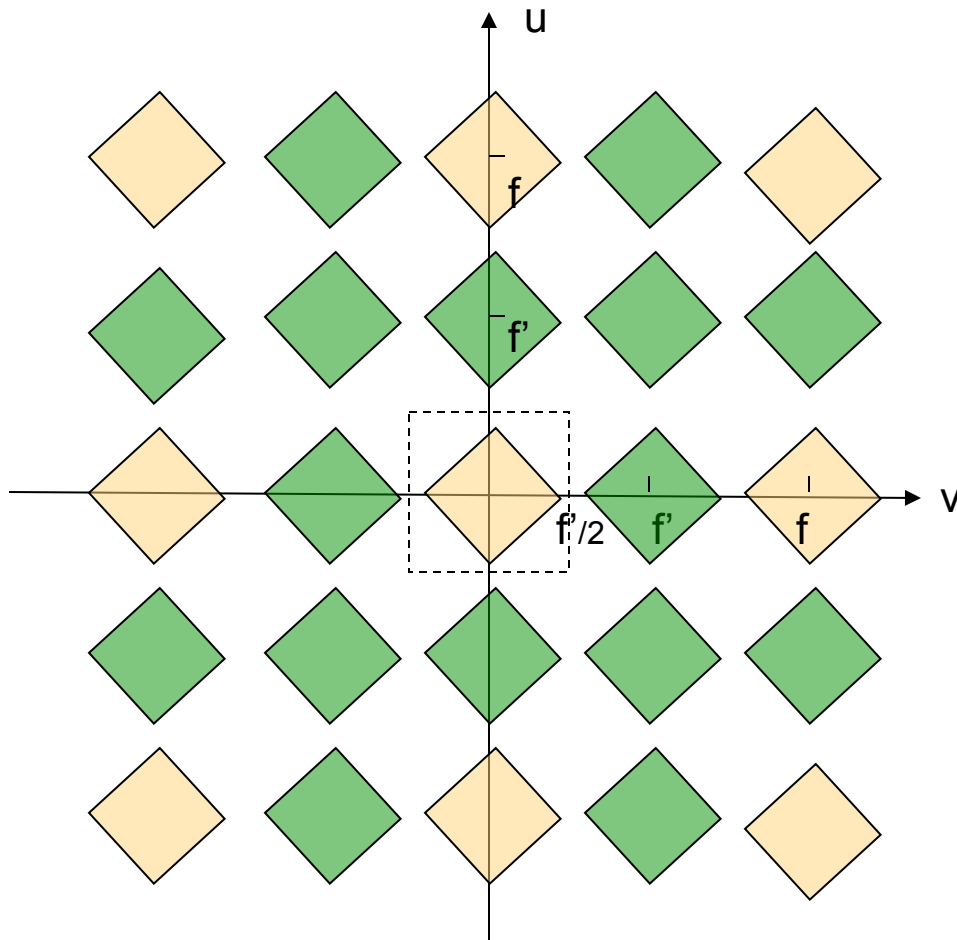
- Averaging Filter



$$f_d(m, n) = [f(2m, 2n) + f(2m, 2n + 1) + f(2m + 1, 2n) + f(2m + 1, 2n + 1)] / 4$$

# Problem of Simple Approach

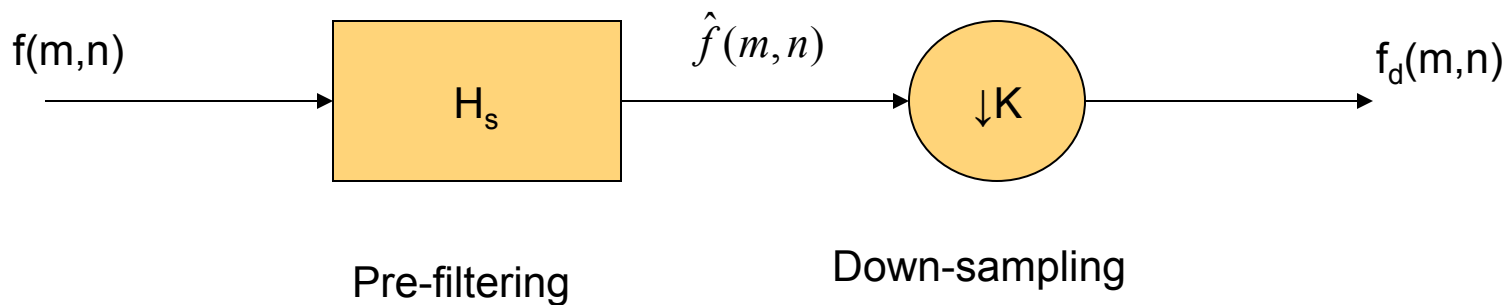
- Aliasing if the effective new sampling rate  $< 2 \times$  highest frequency in the original continuous signal
- We need to prefilter the signal before down-sampling
- Ideally the prefilter should be a low-pass filter with a cut-off frequency half of the new sampling rate.
  - In digital frequency of the original sampled image, highest freq=1/2, the cutoff frequency is 1/4.
- In practice, we may use simple averaging filter

# Spectrum Under Different Sampling Rates



-  Original sampling rate:  $f$
-  New sampling rate:  $f'$
- Need to remove components outside  $f'/2 = f/4$  to avoid aliasing
- Low-pass filtering on original signal, with cutoff at digital freq =  $1/4$

# Down Sampling by a Factor of K



$$f_d(m,n) = \hat{f}(K_1 m, K_2 n)$$
$$F_d(u,v) = \frac{1}{K_1 K_2} \sum_{i=0}^{K_1-1} \sum_{j=0}^{K_2-1} F\left(\frac{u-i}{K_1}, \frac{v-j}{K_2}\right)$$

For factor of K down sampling, the prefilter should be low pass filter with cutoff at  $1/(2K)$  in each direction

Can use MATLAB filter design function, e.g., `fir1(11,1/2)`.

Note in MATLAB, 1.0 represents the highest digital freq of  $1/2$ .

# Example: Image Down-Sample



Without  
prefiltering



With  
averaging

# Down-Sampling Using Matlab

- Without prefiltering
  - If  $f(m,n)$  is an  $M \times N$  image, down-sampling by a factor of  $K$  can be done simply by
    - `>> g=f(1:K:M,1:K:N)`
- With prefiltering
  - First convolve the image with a desired filter
    - Low pass filter with digital cutoff frequency  $1/(2K)$ 
      - In matlab,  $1/2$  is normalized to 1
  - Then subsample
    - `>> h=fir1(N, 1/K)`
      - %design a lowpass filter with cutoff at  $1/2K$  and length  $N$ .
    - `>> fp=conv2(f,h,'same')`
    - `>> g=fp(1:K:M,1:K:N)`

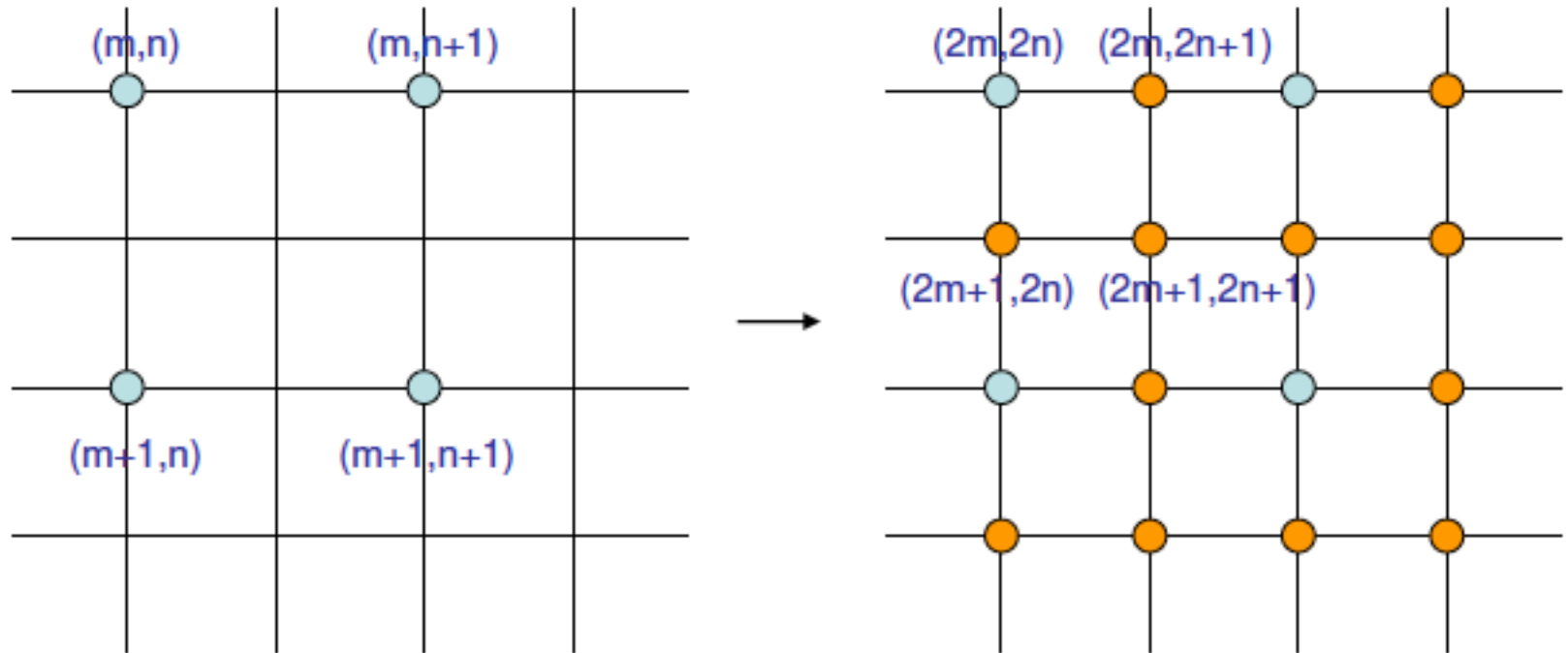
# Image Up-Sampling

- Produce a larger image from a smaller one
  - Eg. 512x512 -> 1024x1024
  - More generally we may up-sample by an arbitrary factor L
- Questions:
  - How should we generate a larger image?
  - Does the enlarged image carry more information?
- Connection with Interpolation of a continuous image from discrete image
  - First interpolate to continuous image, then sampling at a higher sampling rate, Lfs
  - Ideally using the sinc filter!

$$\hat{f}(x, y) = \sum_m \sum_n f_s(m, n) \frac{\sin \pi f_{s,x}(x - m\Delta x)}{\pi f_{s,x}(x - m\Delta x)} \frac{\sin \pi f_{s,y}(y - m\Delta y)}{\pi f_{s,y}(y - m\Delta y)}$$

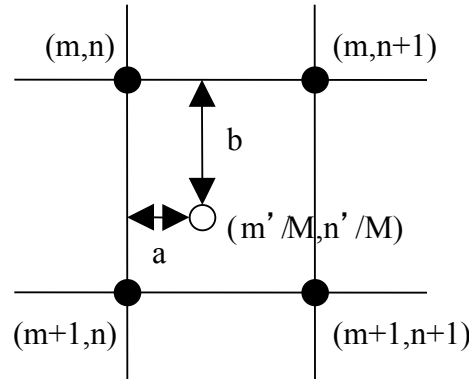


# Example: Factor of 2 Up-Sampling



Green samples are retained in the interpolated image;  
Orange samples are estimated from surrounding green samples.

# Nearest Neighbor Interpolation (pixel replication)



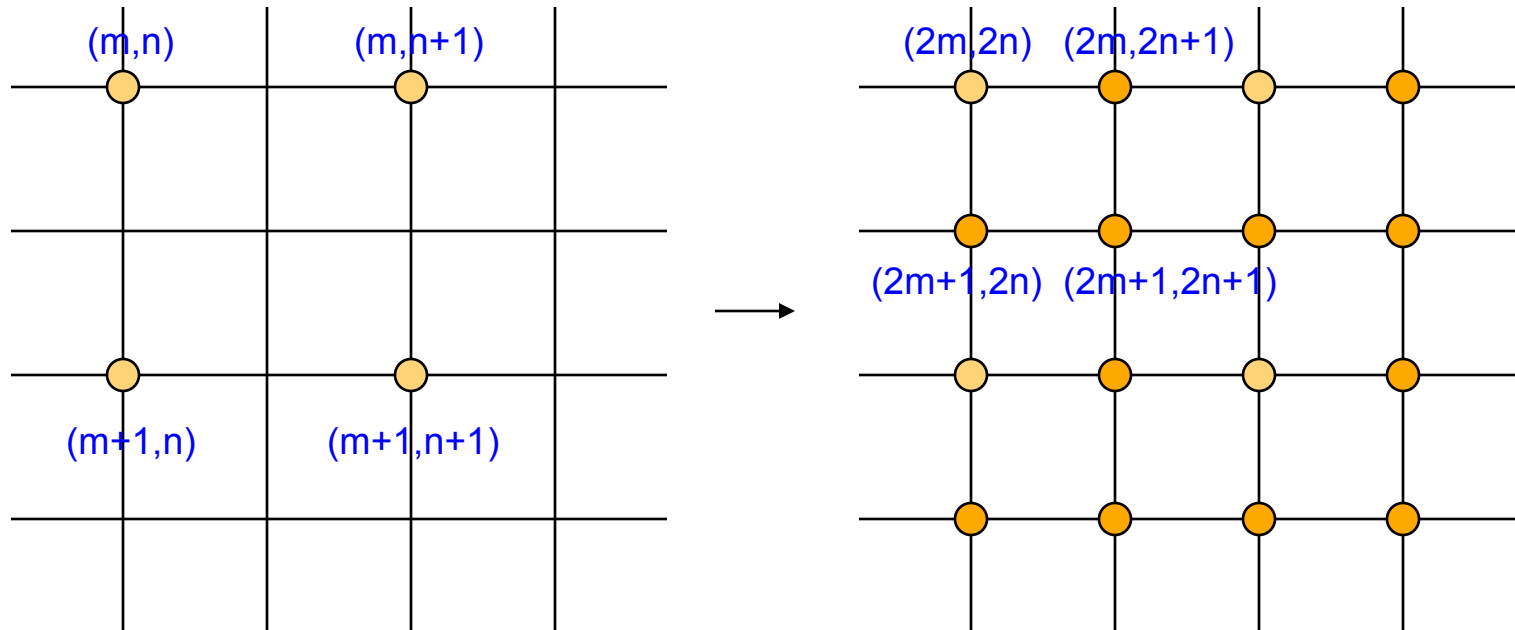
$O[m', n']$  (the resized image) takes the value of the sample nearest to  $(m'/M, n'/M)$  in  $I[m, n]$  (the original image):

$$O[m', n'] = I[(\text{int})(m + 0.5), (\text{int})(n + 0.5)], \quad m = m'/M, \quad n = n'/M.$$

Also known as pixel replication: each original pixel is replaced by  $M \times M$  pixels of the sample value

Equivalent to using the sample-and-hold interpolation filter.

# Special Case: $M=2$



Nearest Neighbor:

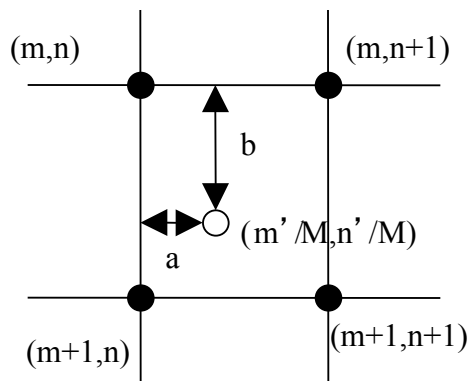
$$O[2m,2n] = I[m,n]$$

$$O[2m,2n+1] = I[m,n]$$

$$O[2m+1,2n] = I[m,n]$$

$$O[2m+1,2n+1] = I[m,n]$$

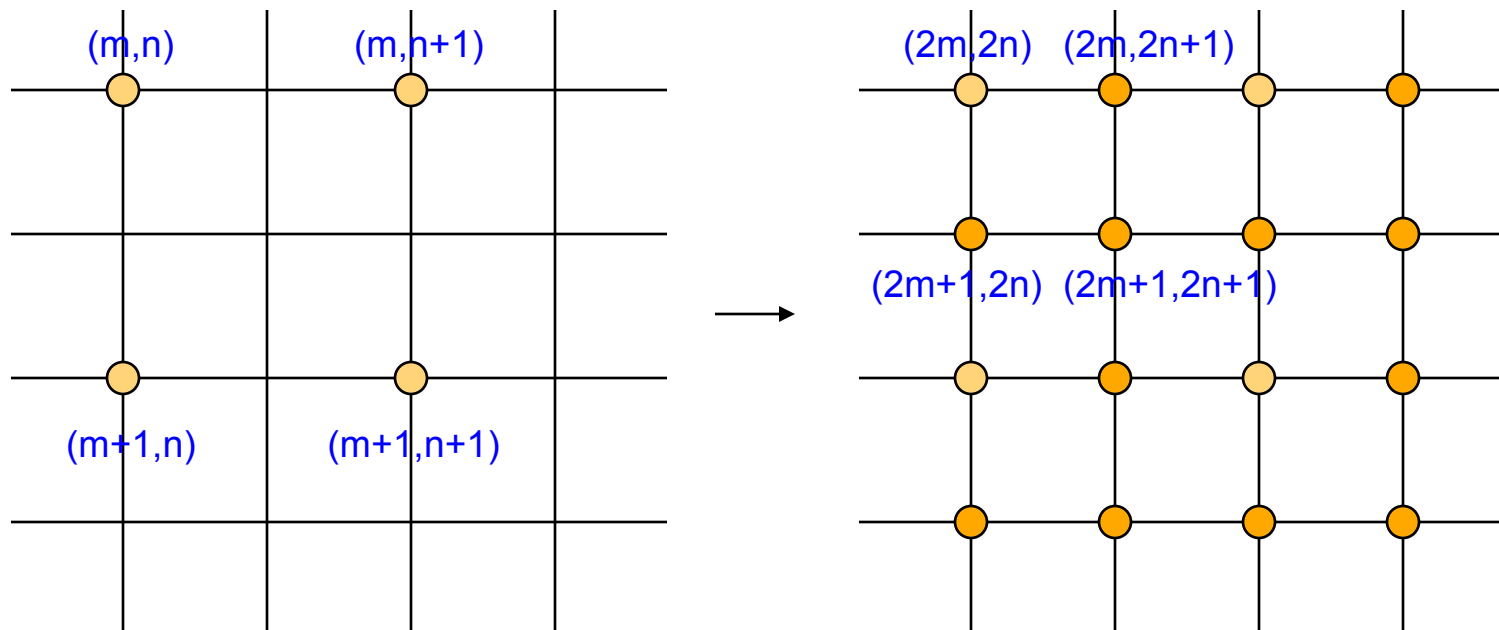
# Bilinear Interpolation



- $O(m', n')$  takes a weighted average of 4 samples nearest to  $(m'/M, n'/M)$  in  $I(m, n)$ .
- **Separable interpolation:**
  - interpolate along each row  $y$ :  $F[m, n'] = (1-a) \cdot I[m, n] + a \cdot I[m, n+1]$   
“a” determined by assuming the samples at  $(m, n)$  and  $(m, n+1)$  form a straight line
  - interpolate along each column  $x'$ :  $O[m', n'] = (1-b) \cdot F[m', n] + b \cdot F[m'+1, n]$
- **Direct interpolation:** each new sample takes 4 multiplications:  

$$O[m', n'] = (1-a) \cdot (1-b) \cdot I[m, n] + a \cdot (1-b) \cdot I[m, n+1] + (1-a) \cdot b \cdot I[m+1, n] + a \cdot b \cdot I[m+1, n+1]$$

# Special Case: M=2



Bilinear Interpolation:

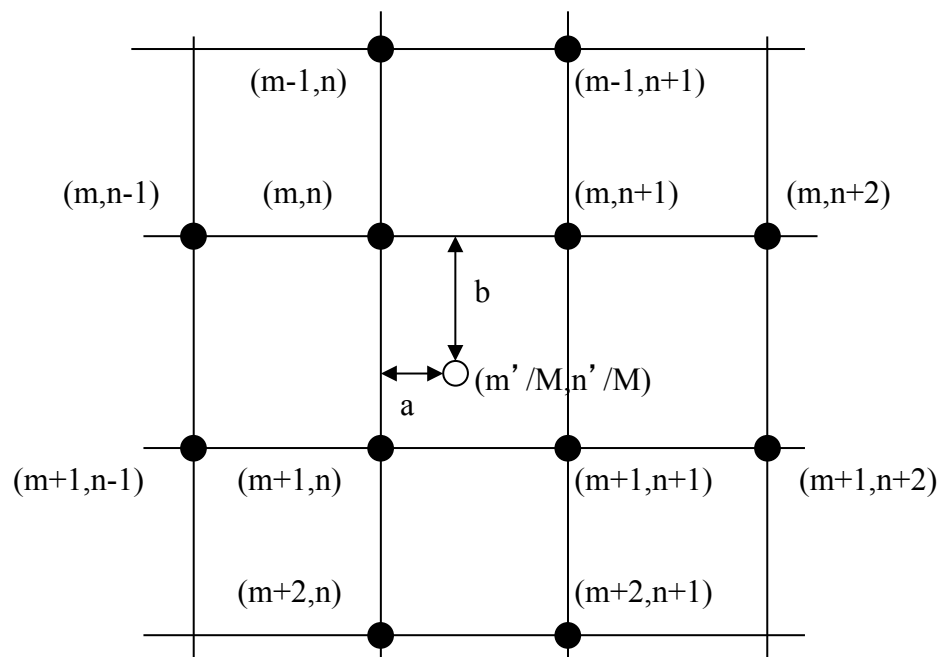
$$O[2m,2n] = I[m,n]$$

$$O[2m,2n+1] = (I[m,n] + I[m,n+1]) / 2$$

$$O[2m+1,2n] = (I[m,n] + I[m+1,n]) / 2$$

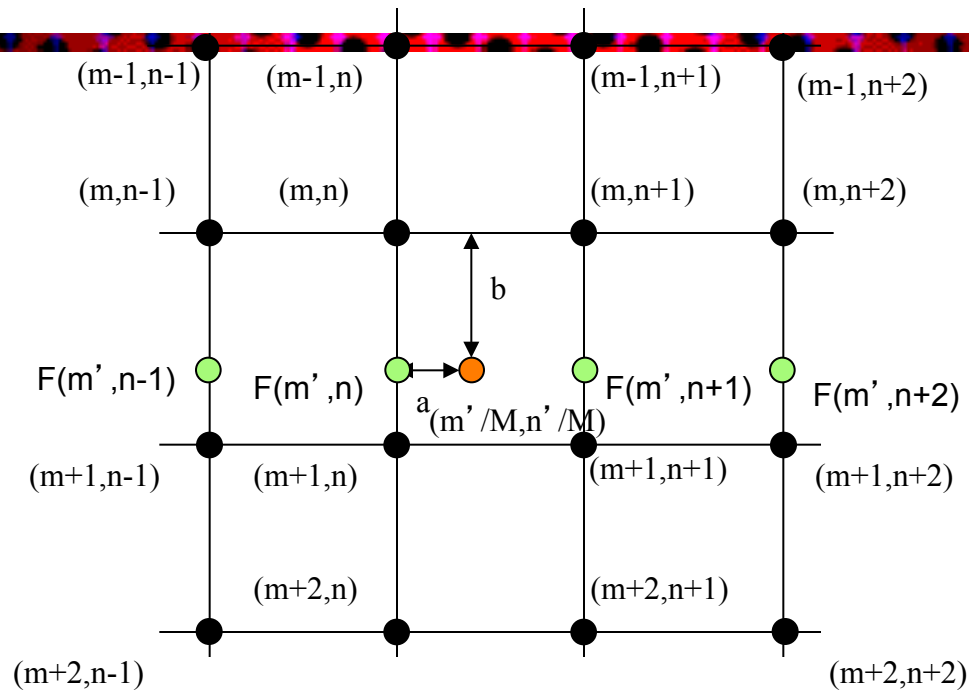
$$O[2m+1,2n+1] = (I[m,n] + I[m,n+1] + I[m+1,n] + I[m+1,n+1]) / 4$$

# Bicubic Interpolation



- $O(m', n')$  is interpolated from **16 samples** nearest to  $(m' / M, n' / M)$  in  $I(m, n)$ .
- **Separable interpolation:**
  - i) interpolate along each row  $y$ :  $I[m, n] \rightarrow F[m, n']$  (from 4 samples)  
Coefficients on these samples determined by fitting the known 4 samples into a cubic polynomial
  - ii) interpolate along each column  $x'$ :  $F[m, n'] \rightarrow O[m', n']$  (from 4 samples)
- **Direct interpolation:** each new sample takes 16 multiplications

# Interpolation Formula



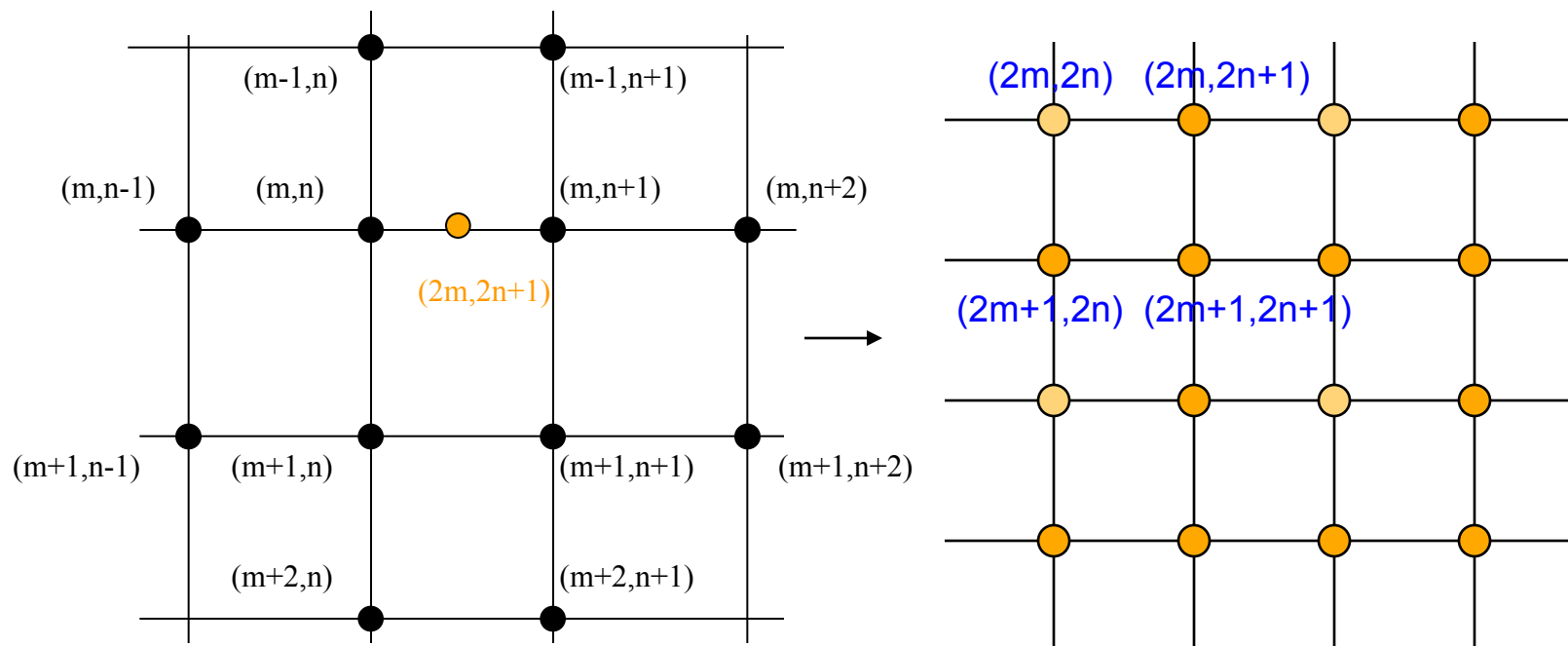
$$F[m', n] = -b(1-b)^2 I[m-1, n] + (1-2b^2 + b^3) I[m, n] + b(1+b-b^2) I[m+1, n] - b^2(1-b) I[m+2, n],$$

where  $m = (\text{int}) \frac{m'}{M}$ ,  $b = \frac{m'}{M} - m$

$$O[m', n'] = -a(1-a)^2 F[m', n-1] + (1-2a^2 + a^3) F[m', n] + a(1+a-a^2) F[m', n+1] - a^2(1-a) F[m', n+2],$$

where  $n = (\text{int}) \frac{n'}{M}$ ,  $a = \frac{n'}{M} - n$

# Special Case: M=2



Bicubic interpolation in Horizontal direction

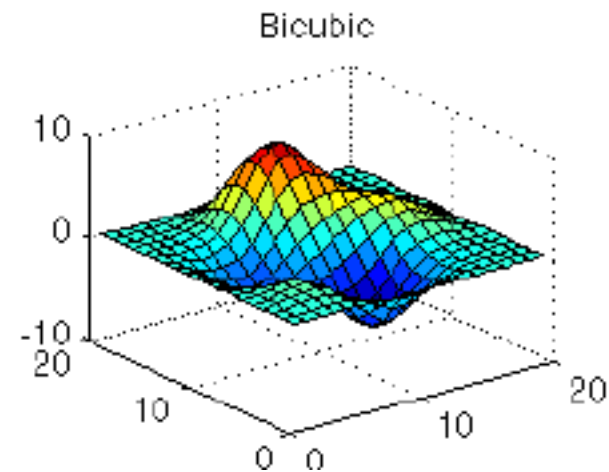
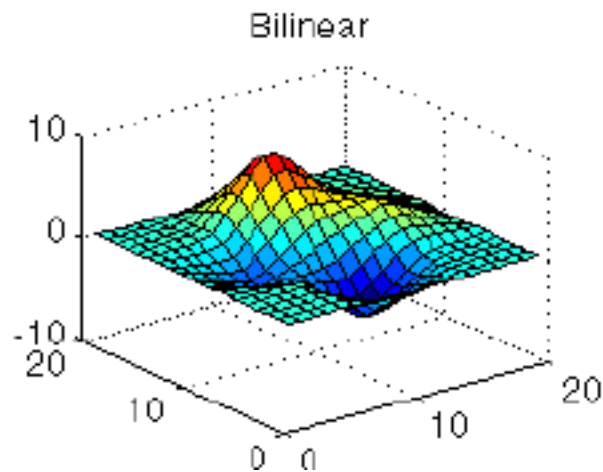
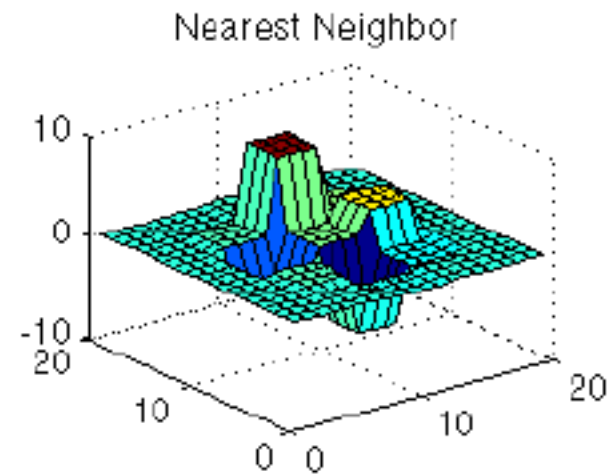
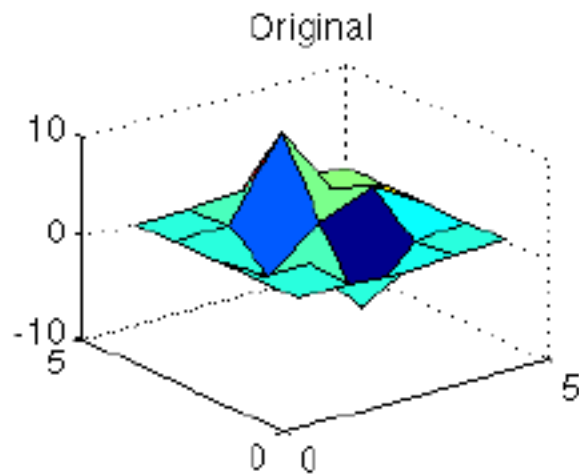
$$F[2m, 2n] = I[m, n]$$

$$F[2m, 2n+1] = -(1/8)I[m, n-1] + (5/8)I[m, n] + (5/8)I[m, n+1] - (1/8)I[m, n+2]$$

Same operation then repeats in vertical direction



# Comparison of Interpolation Methods



Resize\_peak.m

# Up-Sampled from w/o Prefiltering

Original



Nearest  
neighbor



Bilinear



Bicubic



# Up-Sampled from with Prefiltering

Original



Nearest  
neighbor



Bilinear



Bicubic

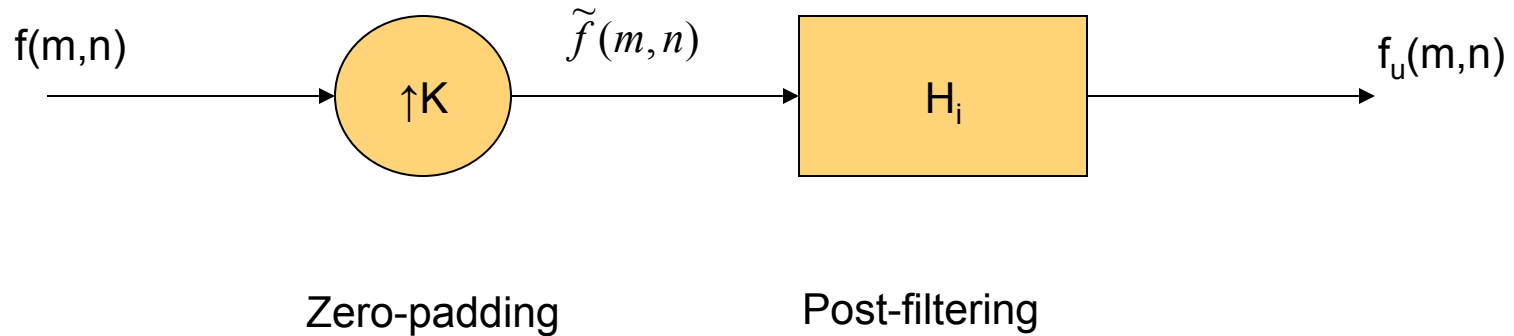




# Matlab for Image Resizing

```
[img]=imread('fruit.jpg','jpg');  
%downsampling without prefiltering  
img1=imresize(img,0.5,'nearest');  
%upsampling with different filters:  
img2rep=imresize(img1,2,'nearest');  
img2lin=imresize(img1,2,'bilinear');  
img2cubic=imresize(img1,2,'bicubic');  
  
%down sampling with filtering  
img1=imresize(img,0.5,'bilinear',11);  
%upsampling with different filters  
img2rep=imresize(img1,2,'nearest');  
img2lin=imresize(img1,2,'bilinear');  
img2cubic=imresize(img1,2,'bicubic');
```

# Filtering View: Up Sampling by a Factor of K

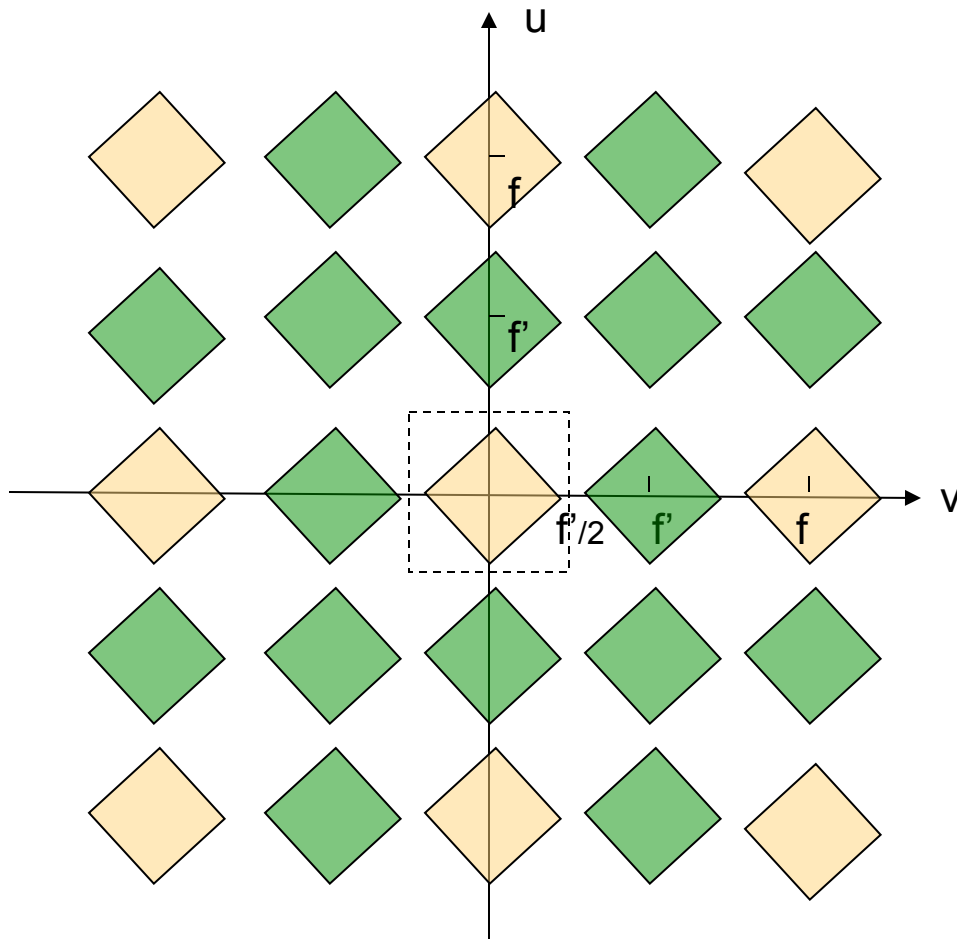


$$\tilde{f}(m,n) = \begin{cases} f(m/K, n/K) & \text{if } m,n \text{ are multiple of } K \\ 0 & \text{otherwise} \end{cases}$$

$$f_u(m,n) = \sum_{k,l} h(k,l) \tilde{f}(m-k, n-l)$$

Ideally H should be a low pass filter with cutoff at  $1/2K$  in digital frequency, or  $f_s/2K$  in continuous frequency

# Spectrum Under Different Sampling Rates

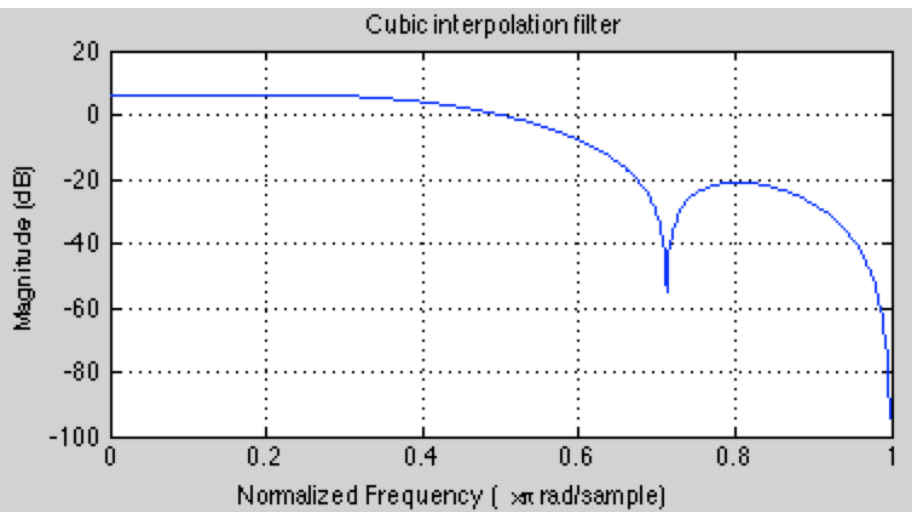
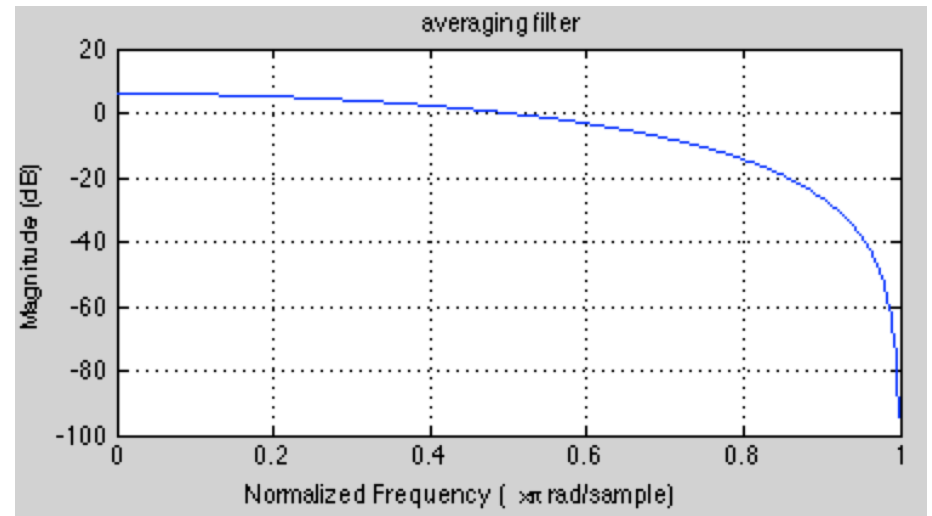
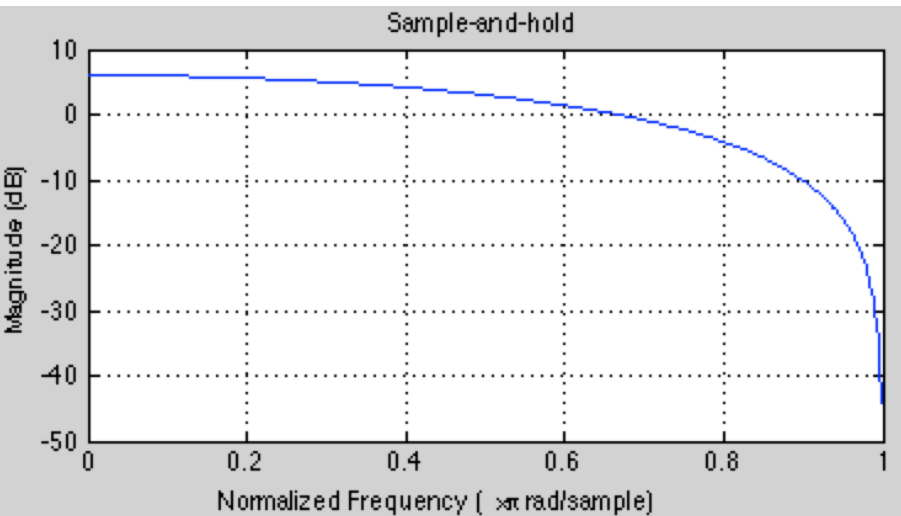


- Original sampling rate:  $f'$  (include all green and yellow replicas)
- New sampling rate:  $f$  (should include only yellow replicas)
- Need to remove green replicated components corresponding to  $f'$  but not  $f$
- Low-pass filtering on zero-filled signal at sampling rate of  $f$ , with cutoff at  $f'/2 = f/4$ , of digital freq =  $1/4$

# Interpolation Filters Corresponding to Different 1D Interpolation Methods for Factor of 2 Upsampling

- The interpolation filter is defined over all pixels in the up-sampled resolution
  - Known samples keep original value ( $h(0)=1$ ,  $h(Ln)=0$ , for factor-of  $L$  interpolation)
  - Sum of filter coefficients (except  $h(0)$ )=1
  - Known as Nyquist filter
- Sample-and-hold
  - $h=[1,1]$
- Linear averaging
  - $h=[0.5, 1, 0.5]$
- Cubic interpolation
  - $h=[-1/8, 0, 5/8, 1, 5/8, 0, -1/8]$

# Freq. Response of Popular Interpolation Filters



In MATLAB: “1” represents  $\frac{1}{2}$  in digital frequency



# Summary on Sampling and Resizing

- Nyquist sampling theorem
  - Direct extension from 1D to 2D to higher dimension
  - Prefiltering to prevent aliasing
  - Interpolation
  - Ideal filter for both, practical filters
  - Separable filtering along each direction
- Image resizing
  - Think of sampling/reconstructing continuous image
  - Think of directly down-sampling or interpolating
  - Practical filters for down-sampling: averaging
  - Practical filters for up-sampling: bilinear and cubic spline
  - Tradeoff between blurring and aliasing

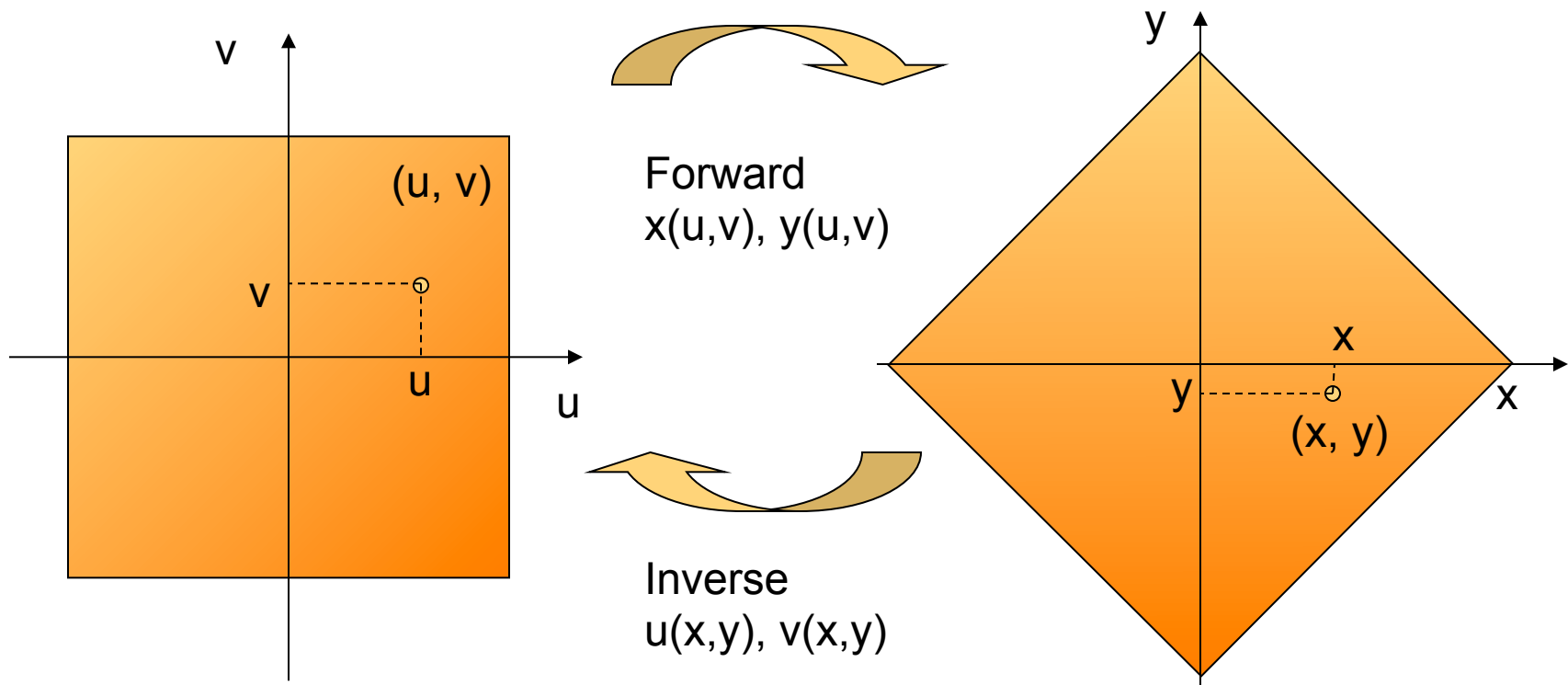
# Image Warping

- Geometric Transformation (Image warping)
- Image registration
- Image morphing

# What is Geometric Transformation?

- So far, the image processing operations we have discussed modify the **color values** of pixels in a given image
- With geometric transformation, we modify the **positions** of pixels in a image, but keep their colors unchanged
  - To create special effects
  - To register two images taken of the same scene at different times or frame different view angles
  - To morph one image to another

# Illustration of Forward and Inverse Mapping Functions



$$\begin{cases} g(x, y) = f(u(x, y), v(x, y)) \\ f(u, v) = g(x(u, v), y(u, v)) \end{cases}, \text{ or } \begin{cases} g(\mathbf{x}) = f(u(\mathbf{x})) \\ f(\mathbf{u}) = g(x(\mathbf{u})) \end{cases}$$

# Translation

- **Translation** is defined by the following mapping functions:

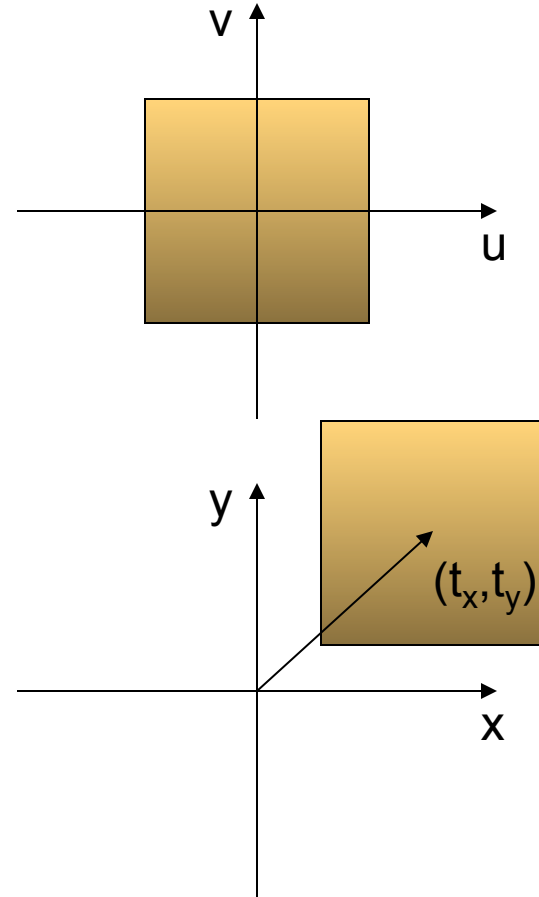
$$\begin{cases} x = u + t_x \\ y = v + t_y \end{cases} \quad \text{and} \quad \begin{cases} u = x - t_x \\ v = y - t_y \end{cases}$$

- In matrix notation

$$\mathbf{x} = \mathbf{u} + \mathbf{t}, \quad \mathbf{u} = \mathbf{x} - \mathbf{t}$$

where

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}.$$



# Scaling

- **Scaling** is defined by

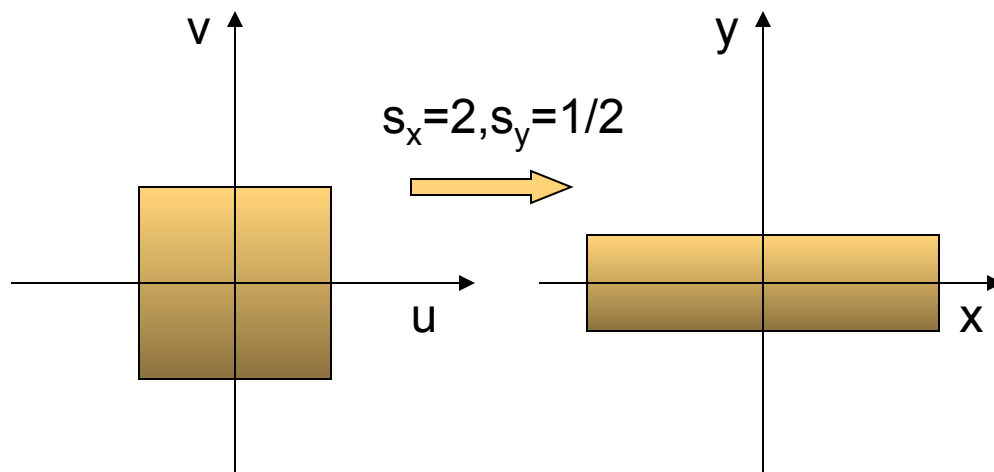
$$\begin{cases} x = s_x u \\ y = s_y v \end{cases} \quad \text{and} \quad \begin{cases} u = x / s_x \\ v = y / s_y \end{cases}$$

- **Matrix notation**

$$\mathbf{x} = \mathbf{S}\mathbf{u}, \quad \mathbf{u} = \mathbf{S}^{-1}\mathbf{x}$$

where

$$\mathbf{S} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$



- If  $s_x < 1$  and  $s_y < 1$  -> **shrinking**,
- if  $s_x > 1$  and  $s_y > 1$ , -> **zoom**.

# Rotation

- Rotation by an angle of  $\theta$  is defined by

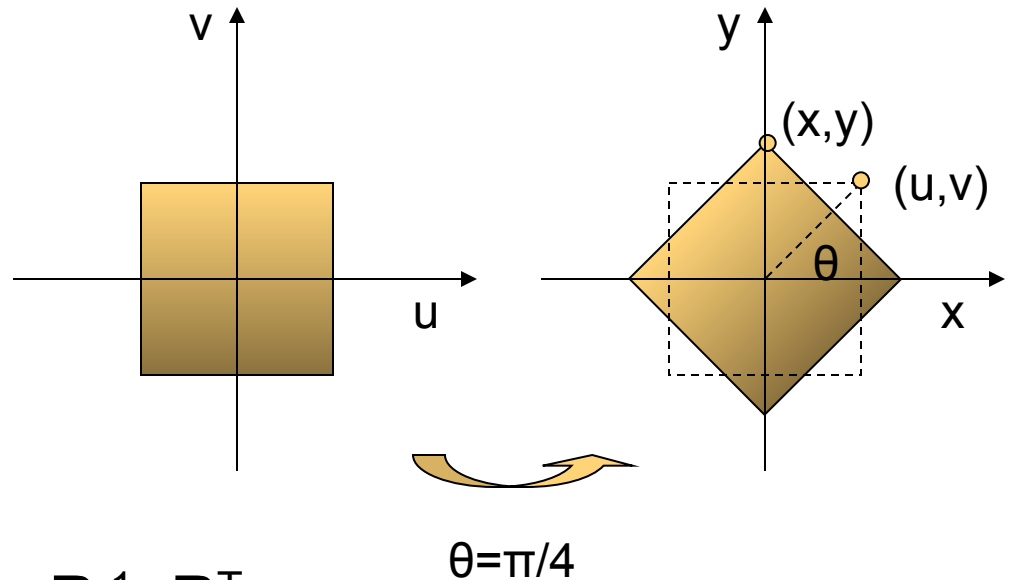
$$\begin{cases} x = u \cos \theta - v \sin \theta \\ y = u \sin \theta + v \cos \theta \end{cases} \quad \text{and} \quad \begin{cases} u = x \cos \theta + y \sin \theta \\ v = -x \sin \theta + y \cos \theta \end{cases}$$

- In matrix format

$$\mathbf{x} = \mathbf{R}\mathbf{u}, \quad \mathbf{u} = \mathbf{R}^T \mathbf{x}$$

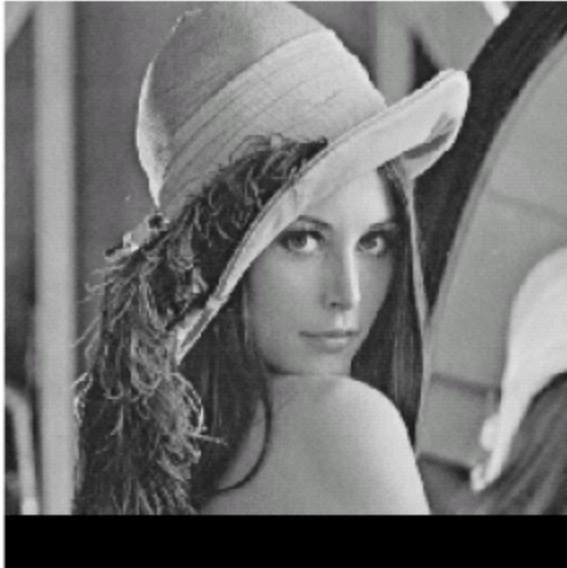
where

$$\mathbf{R} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$



- $\mathbf{R}$  is a **unitary matrix**:  $\mathbf{R}^{-1} = \mathbf{R}^T$

B translation



B rotation



Translation:  $x(k, l) = k + 50; y(k, l) = l;$

Rotation:  $x(k, l) = (k - x_0)\cos(\theta) + (l - y_0)\sin(\theta) + x_0;$   
 $y(k, l) = -(k - x_0)\sin(\theta) + (l - y_0)\cos(\theta) + y_0;$

$x_0 = y_0 = 256.5$  the center of the image **A**,  $\theta = \pi/6$

By Onur Guleyuz



# Affine Mapping

- Affine Mapping includes translation, scaling and rotation as special cases:

$$\begin{cases} x = a_0 + a_1u + a_2v \\ y = b_0 + b_1u + b_2v \end{cases} \quad or \quad \mathbf{x} = \mathbf{A}\mathbf{u} + \mathbf{b}$$

$$\mathbf{A} = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} a_0 \\ b_0 \end{bmatrix}$$

- When  $\mathbf{A}$  is an orthonormal matrix, it corresponds to a rotation matrix, and the corresponding affine mapping reduces to rotation plus translation.

# Relevant MATLAB Functions

- `G = MAKETFORM('affine',T)` builds a TFORM struct `G` for affine transformation. `T` defines a forward transformation

In MATLAB notation

$$T = \begin{bmatrix} a_1 & b_1 & 0 \\ a_2 & b_2 & 0 \\ a_0 & b_0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A}^T & 0 \\ \mathbf{b}^T & 1 \end{bmatrix}$$

- `B = IMTRANSFORM(A,TFORM, INTERP)` transforms the image `A` according to the 2-D spatial transformation defined by `TFORM`; `INTERP` specifies the interpolation filter
- Example 1
- -----
- Apply a horizontal shear to an intensity image.
- 
- `I = imread('cameraman.tif');`
- `tform = maketform('affine',[1 0 0; .5 1 0; 0 0 1]);`
- `J = imtransform(I,tform);`
- `figure, imshow(I), figure, imshow(J)`

# Horizontal Shear Example



`tform = maketform('affine',[1 0 0; .5 1 0; 0 0 1]);`  
In MATLAB, 'affine' transform is defined by:  
`[a1,b1,0;a2,b2,0;a0,b0,1]`

With notation used in this lecture note

$$\mathbf{A} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Note in this example, first coordinate indicates horizontal position, second coordinate indicate vertical

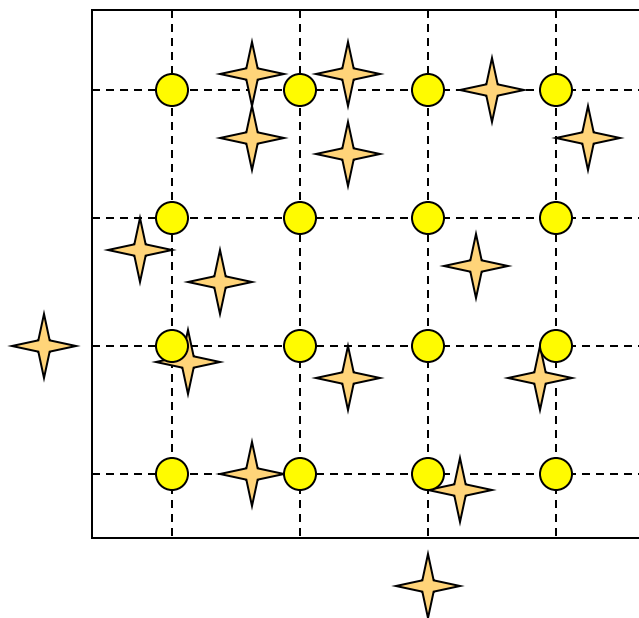
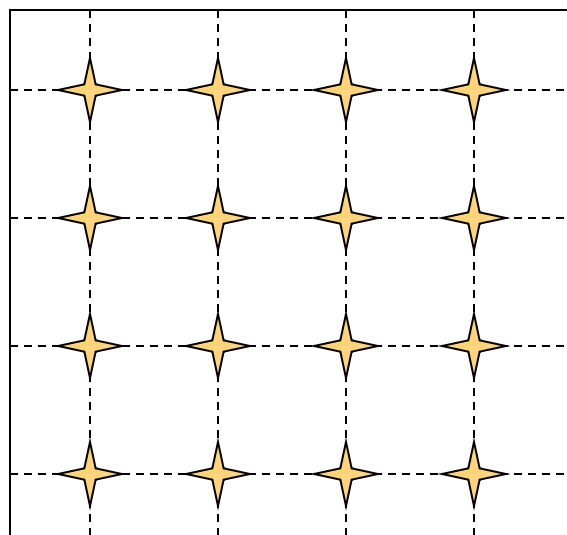
# Bilinear Mapping

$$\begin{cases} x = a_0 + a_1u + a_2v + a_3uv \\ y = b_0 + b_1u + b_2v + b_3uv \end{cases}$$

- Mapping a square to a quadrangle.
- More generally mapping a quadrangle to another quadrangle.
- 8 parameters.
- Can be completely determined from how the four corners moved.

# Image Warping by Forward Mapping

- Mapping image  $f(u, v)$  to  $g(x, y)$  based on a given mapping function:  $x(u, v)$ ,  $y(u, v)$ .
- Forward Mapping
  - For each point  $(u, v)$  in the original image, find the corresponding position  $(x, y)$  in the deformed image by the forward mapping function, and let  $g(x, y) = f(u, v)$ .
  - What if the mapped position  $(x, y)$  is not an integer sample in the desired image?

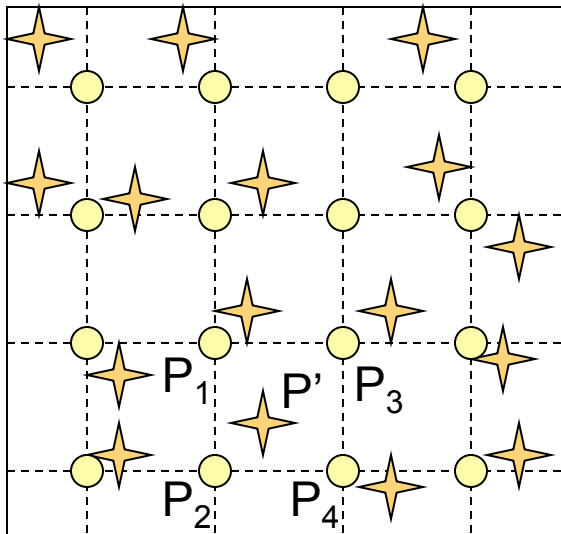


Warping points  
are often non-  
integer samples

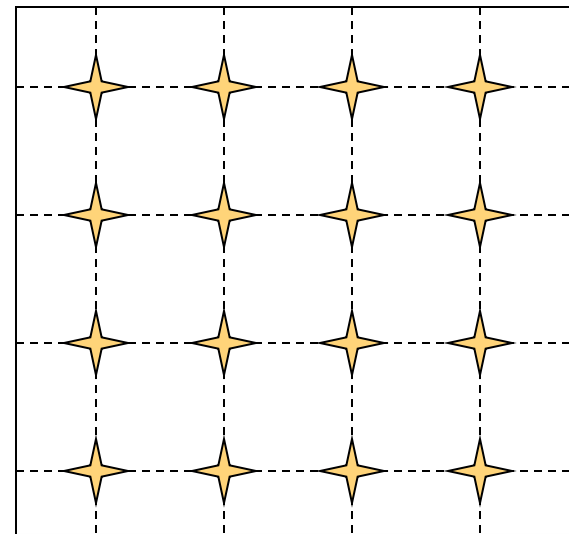
Many integer  
samples "o"  
are not assigned  
Values

# Image Warping by Inverse Mapping

- For each point  $(x, y)$  in the image to be obtained, find its corresponding point  $(u, v)$  in the original image using the inverse mapping function, and let  $g(x, y) = f(u, v)$ .
- What if the mapped point  $(u, v)$  is not an integer sample?
  - Interpolate from nearby integer samples!



$P'$  will be interpolated  
from  $P_1, P_2, P_3,$  and  $P_4$



# Interpolation Method

- Nearest neighbor:
  - Round  $(u,v)$  to the nearest integer samples
- Bilinear interpolation:
  - find four integer samples nearest to  $(u,v)$ , apply bilinear interpolation
- Other higher order interpolation methods can also be used
  - Requiring more than 4 nearest integer samples!



# MATLAB function: interp2

- `ZI = INTERP2(X,Y,Z,XI,YI, METHOD)` interpolates to find `ZI`, the values of the underlying 2-D function `Z` at the points defined by matrices `XI` and `YI`.
  - Matrices `X` and `Y` specify the points at which the data `Z` is given.
  - `METHOD` specifies interpolation filter
    - 'nearest' - nearest neighbor interpolation
    - 'linear' - bilinear interpolation
    - 'spline' - spline interpolation
    - 'cubic' - bicubic interpolation as long as the data is uniformly spaced, otherwise the same as 'spline'

# Using 'interp2' to realize image warping

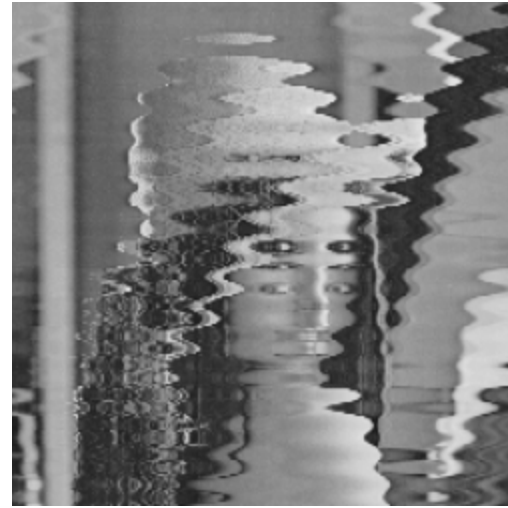
- Use inverse mapping
- Step 1: For all possible pixels in output image (x,y), find corresponding points in the input image (u,v)
  - $(X,Y)=\text{meshgrid}(1:M,1:N)$
  - Apply inverse mapping function to find corresponding (u,v), for every (x,y), store in (UI,VI)
    - Can use `tforminv( )` function if you derived the transformation using `maketform()`.
    - Or write your own code using the specified mapping
- Step 2: Use `interp2` to interpret the value of the input image at (UI,VI) from their values at regularly sampled points (X,Y)
  - `Outimg=interp2(X,Y,inimg,UI,VI,'linear');`

# Example of Image Warping (1)

WAVE1



WAVE2



wave1: $x(u,v)=u+20\sin(2\pi v/128)$ ;  $y(u,v)=v$ ;  
wave2: $x(u,v)=u+20\sin(2\pi u/30)$ ;  $y(u,v)=v$ .

By Onur Guleyuz

# Example of Image Warping (2)

WARP



SWIRL



WARP

$$x(u, v) = \text{sign}(u - x_0) * (u - x_0)^2 / x_0 + x_0; y(u, v) = v$$

SWIRL

$$\begin{aligned} x(u, v) &= (u - x_0) \cos(\theta) + (v - y_0) \sin(\theta) + x_0; \\ y(u, v) &= -(u - x_0) \sin(\theta) + (v - y_0) \cos(\theta) + y_0; \\ r &= ((u - x_0)^2 + (v - y_0)^2)^{1/2}, \theta = \pi r / 512. \end{aligned}$$

By Onur Guleyuz

# Image Registration

- Suppose we are given **two images** taken at different times of the **same scene**. To observe the changes between these two images, we need to make sure that they are aligned properly. To obtain this goal, we need to find the correct **mapping function** between the two. The determination of the mapping functions between two images is known as the **registration problem**.
- Once the mapping function is determined, the alignment step can be accomplished using the warping methods.

# How to find the mapping function?

- Assume the mapping function is a polynomial of order N
- Step 1: Identify  $K \geq N$  corresponding points between two images, i.e.

$$(u_i, v_i) \leftrightarrow (x_i, y_i), i = 1, 2, \dots, K.$$

- Step 2: Determine the coefficients  $a_i, b_i, i = 0, \dots, N-1$  by solving

$$\begin{cases} x(u_i, v_i) = a_0 + a_1 u_i + a_2 v_i + \dots = x_i, \\ y(u_i, v_i) = b_0 + b_1 u_i + b_2 v_i + \dots = y_i, \end{cases} \quad i = 1, 2, \dots, K$$

- How to solve this?

# How to Solve the Previous Equations?

- Convert to matrix equation:

$$\mathbf{A}\mathbf{a} = \mathbf{x}, \quad \mathbf{A}\mathbf{b} = \mathbf{y}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & u_1 & v_1 & \cdots \\ 1 & u_2 & v_2 & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ 1 & u_K & v_K & \cdots \end{bmatrix}, \mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{bmatrix}$$

If  $K = N$ , and the matrix  $\mathbf{A}$  is non-singular, then

$$\mathbf{a} = \mathbf{A}^{-1}\mathbf{x}, \quad \mathbf{b} = \mathbf{A}^{-1}\mathbf{y}$$

If  $K > N$ , then we can use a least square solution

$$\mathbf{a} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{x}, \quad \mathbf{b} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y}$$

If  $K < N$ , or  $\mathbf{A}$  is singular, then more corresponding feature points must be identified.

# Examples

- If we want to use an affine mapping to register to images, we need to find 3 or more pairs of corresponding points
- If we have only 3 pairs, we can solve the mapping parameters exactly as before
- If we have more than 3 pairs, these pairs may not all be related by an affine mapping. We find the “least squares fit” by solving an over-determined system of equations



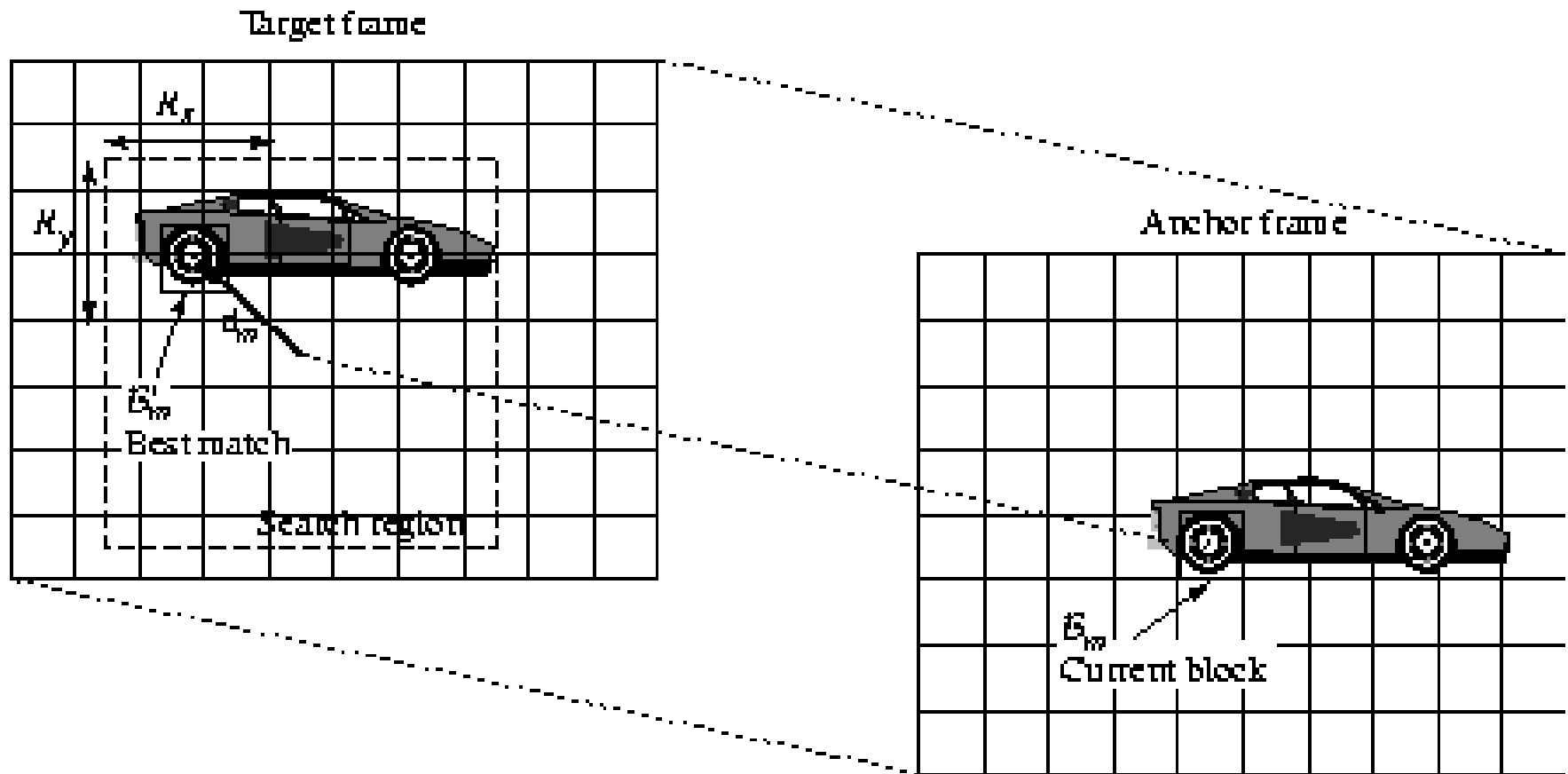
# How to find corresponding points in two images?

- Which points to select in one image (image 1)?
  - Ideally choose “interesting points”: corners, special features (known as feature point selection)
  - Can also use points over a regular grid
  - Popular method for selecting feature points automatically:
    - SIF features
    - Corner detectors
- How to find the corresponding point in the other image (image 2)?
  - Put a small block around the point in image 1
  - Find a block in image 2 that matches the block pattern the best
  - Exhaustive search within a certain range.

# MATLAB function for manually selecting control points

- CPSELECT(INPUT,BASE) returns control points in CPSTRUCT. INPUT is the image that needs to be warped to bring it into the coordinate system of the BASE image.
- Example
- `cpselect('westconcordaerial.png','westconcordorthophoto.png')`

# Exhaustive Block Matching Algorithm (to be discussed later)



# MATLAB function: `cp2tform()`

`TFORM=CP2TFORM(INPUT_POINTS,BASE_POINTS,TRANSFORMTYPE)`

- returns a TFORM structure containing a spatial transformation.
- INPUT\_POINTS is an M-by-2 double matrix containing the X and Y coordinates of control points in the image you want to transform.
- BASE\_POINTS is an M-by-2 double matrix containing the X and Y coordinates of control points in the base image.
- TRANSFORMTYPE can be 'nonreflective similarity', 'similarity', 'affine', 'projective', 'polynomial', 'piecewise linear' or 'lwm'.
- Example: Create an affine transformation that maps the triangle with vertices (0,0), (6,3), (-2,5) to the triangle with vertices (-1,-1), (0,-10), (4,4):
  - `inpt=[0,0;6,3;-2,5],`
  - `outpt=[-1,-1;0,-10;4,4]`
  - `tform = cp2tform(inpt,outpt,'affine');`

# MATLAB function for image warping

- `B = IMTRANSFORM(A,TFORM, INTERP)` transforms the image A according to the 2-D spatial transformation defined by TFORM
- INTERP specifies the interpolation filter
- Example 1
- -----
- Apply a horizontal shear to an intensity image.
- 
- `I = imread('cameraman.tif');`
- `tform = maketform('affine',[1 0 0; .5 1 0; 0 0 1]);`
- `J = imtransform(I,tform);`
- `figure, imshow(I), figure, imshow(J)`

# MATLAB Example

- Register an aerial photo to an orthophoto.

```
unregistered = imread('westconcordaerial.png');  
figure, imshow(unregistered)  
figure, imshow('westconcordorthophoto.png')  
load westconcordpoints % load some points that were already  
picked  
t_concord = cp2tform(input_points,base_points,'projective');  
info = imfinfo('westconcordorthophoto.png');  
registered = imtransform(unregistered,t_concord,...  
                        'XData',[1 info.Width], 'YData',[1 info.Height]);  
figure, imshow(registered)
```

# Examples of Image Morphing

Cross  
Dissolve

$$I(t) = (1-t)*S + t*T$$



Mesh  
based



*George Wolberg, "Recent Advances in Image Morphing",  
Computer Graphics Intl. '96, Pohang, Korea, June 1996.*

# Image Morphing

- Image morphing has been widely used in movies and commercials to create special visual effects. For example, changing a beauty gradually into a monster.
- The fundamental techniques behind image morphing is image warping.
- Let the original image be  $f(\mathbf{u})$  and the final image be  $g(\mathbf{x})$ . In image warping, we create  $g(\mathbf{x})$  from  $f(\mathbf{u})$  by changing its shape. In image morphing, we use a combination of both  $f(\mathbf{u})$  and  $g(\mathbf{x})$  to create a series of intermediate images.



# Image Morphing Method

- Suppose the mapping function between the two end images is given as  $\mathbf{x}=\mathbf{u}+\mathbf{d}(\mathbf{u})$ .  $\mathbf{d}(\mathbf{u})$  is the displacement between corresponding points in these two images.
- In image morphing, we create a series of images, starting with  $f(\mathbf{u})$  at  $k=0$ , and ending at  $g(\mathbf{x})$  at  $k=K$ . The intermediate images are a linear combination of the two end images:

$$h_k(\mathbf{u} + s_k \mathbf{d}) = (1 - s_k) f(\mathbf{u}) + s_k g(\mathbf{u} + \mathbf{d}(\mathbf{u})), \quad k = 0, 1, \dots, K,$$

where  $s_k = k / K$ .

- Perform two image warpings:
  - From  $f(\mathbf{u})$  to  $h(\mathbf{u}+s \mathbf{d})$  (to realize by using inverse mapping to obtain  $h_1(\mathbf{x})$  from  $f(\mathbf{x}-s\mathbf{d})$ )
  - From  $g(\mathbf{u}+\mathbf{d}(\mathbf{u}))$  to  $h(\mathbf{u}+s \mathbf{d})$  (to realize by using inverse mapping to obtain  $h_2(\mathbf{x})$  from  $g(\mathbf{x}+(1-s)\mathbf{d})$ )
  - Weighted averaging  $h_1(\mathbf{x})$  and  $h_2(\mathbf{x})$

# Summary: Image Warping

- Basic image warping operation
  - Realizing image warping using inverse mapping
- Image registration
  - Determining warping function from feature correspondence
  - Feature detection and feature matching (not covered in this lecture)
  - Multiple applications: align images taken in different angles/times; generating image panorama.
- Image morphing
  - Manually selecting corresponding features
  - Produces a set of intermediate images
  - Determining warping functions for creating each intermediate image

# Reading Assignment

---

- Lecture Note for this class.
- Wang, Chap 3 and 4 on sampling and sampling rate conversion. You can focus on rectangular sampling only.
- George Wolberg, Digital Image Warping, Wiley-IEEE Computer Society Press, 1990

# Written Homework (1)

1. Consider a function  $f(x, y) = \cos 2\pi(4x + 2y)$  sampled with a sampling period of  $\Delta x = \Delta y = \Delta = 1/6$  or sampling frequency  $f_s = 1/\Delta = 6$ .

a) Assume that it is reconstructed with an ideal low-pass filter with cut-off frequency  $f_{cx} = f_{cy} = 1/2f_s$ . Illustrate the spectra of the original, sampled, and reconstructed signals. Give the spatial domain function representation of the reconstructed signal. Is the result as expected?

b) If the reconstruction filter has the following impulse response:

$$h(x, y) = \begin{cases} 1 & -\Delta/2 < x, y < \Delta/2 \\ 0 & \text{otherwise} \end{cases}$$

Illustrate the spectra of the reconstructed signal in the range  $-f_s \leq u, v \leq f_s$ . Give a spatial domain function representation of the reconstructed signal if the reconstruction filter is band-limited to  $(-f_s \leq u, v \leq f_s)$ . (i.e., this filter remains the same for the frequency range  $-f_s \leq u, v \leq f_s$ , and is set to 0 outside this range.)

2. You are given two pictures of the same scene, taken at different times. In order to align the two pictures, you need to find a mapping function between the two pictures based on some common feature points. Suppose you were able to extract  $N$  ( $N \geq 4$ ) feature points in both images that correspond to the same set of object features, with image coordinates given as  $(u_k, v_k)$  and  $(x_k, y_k), k=1, 2, \dots, N$ . Also, suppose you want to use an bilinear mapping to approximate the actual unknown mapping function. How would you determine the affine mapping parameters?

3. Suppose you as a doctor took x-rays of the same patient in two different visits and you would like to see the difference between the two images clearly. Propose a method to do that. List all the steps involved.

# Computer Assignments

1. Write your own program or programs which can: a) Down sample an image by a factor of 2, without using a prefilter, and with using a filter you designed using MATLAB; b) Up-sample the previously down-sampled images by a factor of 2, using the bilinear interpolation and cubic interpolation methods, respectively. You should have a total of 4 interpolated images, with different combination of down-sampling and interpolation methods. Your program could either directly display on screen the processed images during program execution, or save the processed images as computer files for display after program execution. Run your program with the image *Barbara*. Comment on the quality of the down/up sampled images obtained with different methods.

Note: you should not use the "resize" function in Matlab to do this assignment. But you are encouraged to compare results of your program with "resize".

2. Computer assignment: Write a matlab program that implements the following steps: i) Read in two images that are taken of the same scene but at slightly different angles; ii) select corresponding feature points (more than 3) in these two images (You can use `cpselect()` function; iii) determine the affine transform between the two images; iii) apply affine transform to one image so that it is aligned with the other image. Please note that you should not use the 'cp2tform' or 'imtransform()' function in MATLAB. You should write your own program, which can call "interp2()". Compare your results with that obtained using the 'cp2tform' and 'imtransform()'.

3 (optional) Implement a program for image morphing.