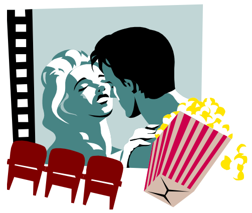
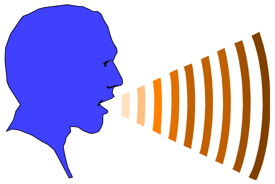



Image and Video Processing

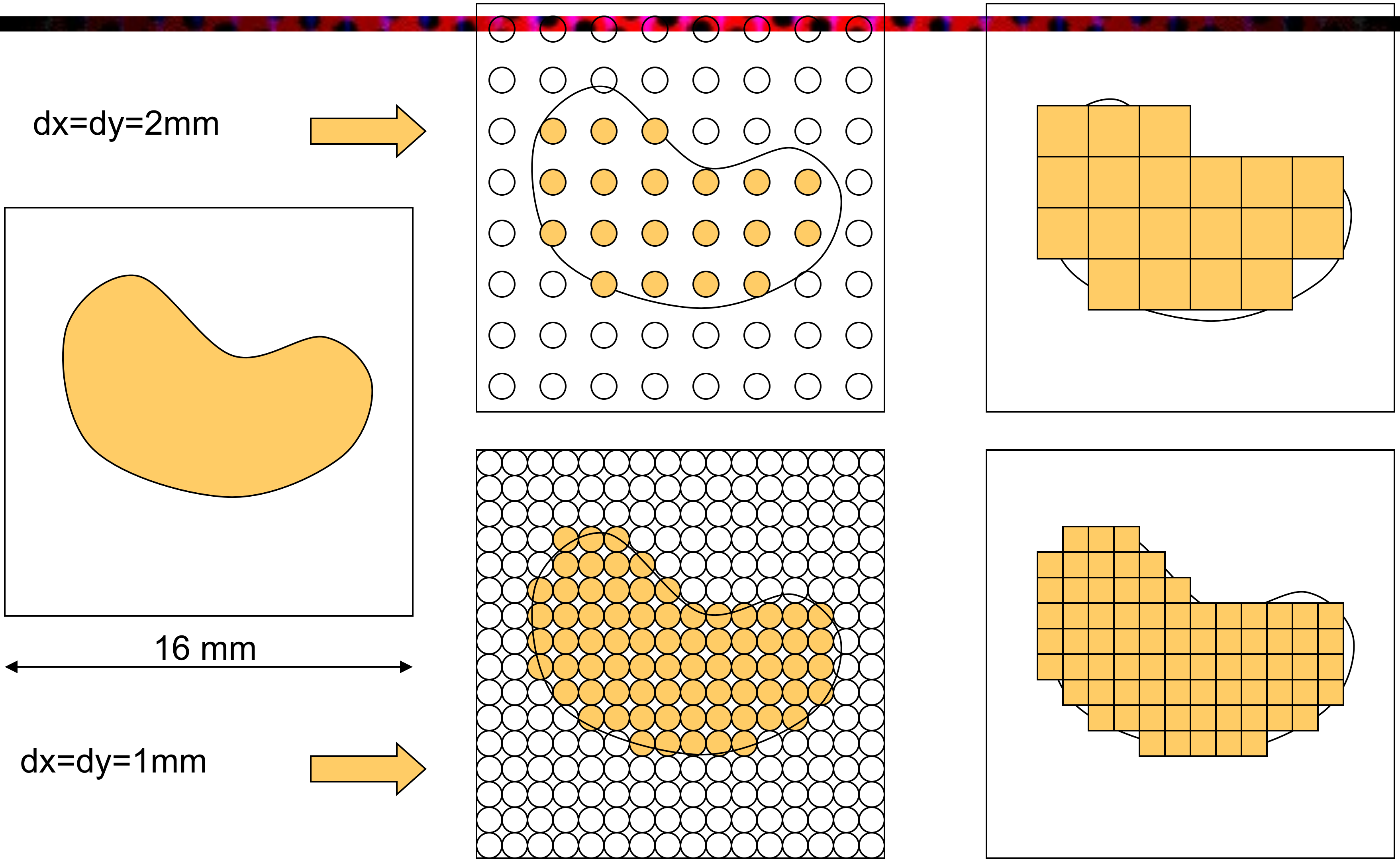
Sampling and Resizing

Yao Wang
Tandon School of Engineering, New York University

Sampling and Resizing

- Nyquist sampling and interpolation theorem: 1D->2D
- Common sampling and interpolation filters
- Human visual system's frequency responses
- Necessary sampling rates and common video formats
- 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

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

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

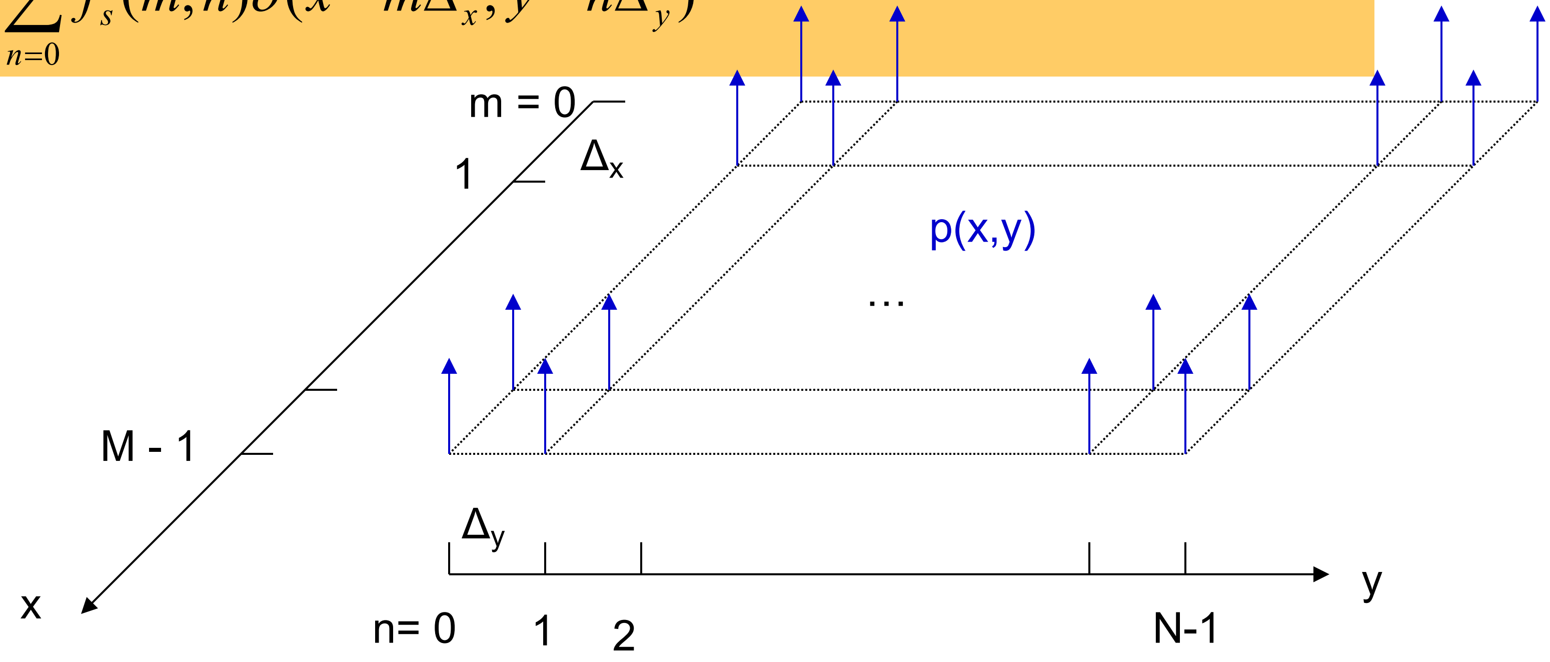
Image Sampling as Product with Impulse Train

- 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

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

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

- 2D

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

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

- Important properties (convolution with δ functions)

$$f(x,y) * \delta(x - x_0, y - y_0) = f(x - x_0, y - y_0)$$
$$F(u,v) * \delta(u - u_0, v - v_0) = F(u - u_0, v - v_0)$$

Frequency Domain Interpretation of Sampling

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

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

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

- In frequency domain

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

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

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

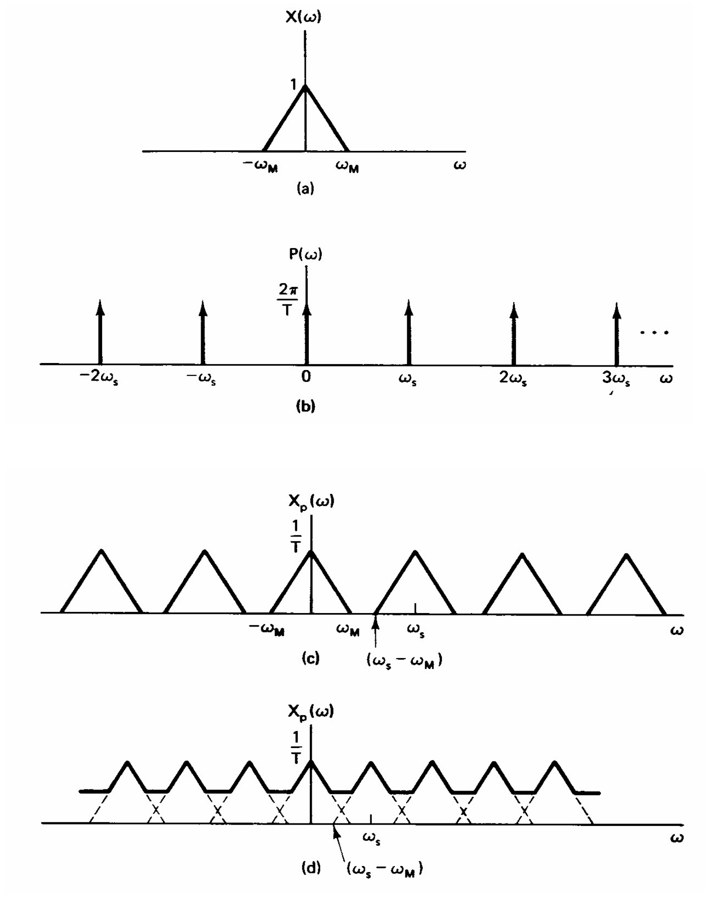
Frequency Domain Interpretation of Sampling in 1D

Original signal

Sampling impulse train

Sampled signal $f_s > 2f_m$ (no aliasing)

Sampled signal $f_s < 2f_m$ (with aliasing)



The spectrum of the sampled signal includes the original spectrum and its aliases (copies) shifted to *k* f_s, *k=+/- 1,2,3,…*

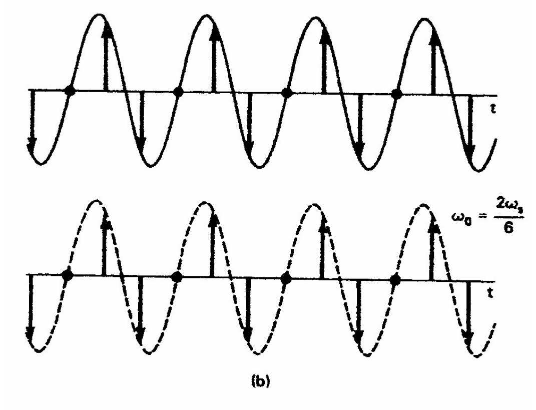
When $f_s < 2f_m$, aliasing occur (the original spectrum and some aliased versions overlap)

Nyquist frequency = 2 maximum freq.*

Sampling of 1D Sinusoid Signals

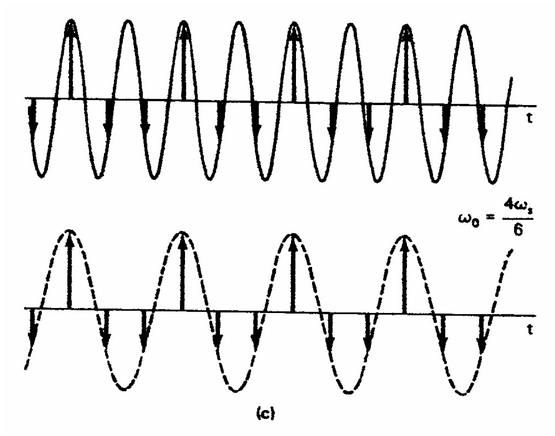
Sampling above Nyquist rate
$\omega_s=3\omega_m>\omega_{s0}$

Reconstructed =original



Sampling under Nyquist rate
$\omega_s=1.5\omega_m<\omega_{s0}$

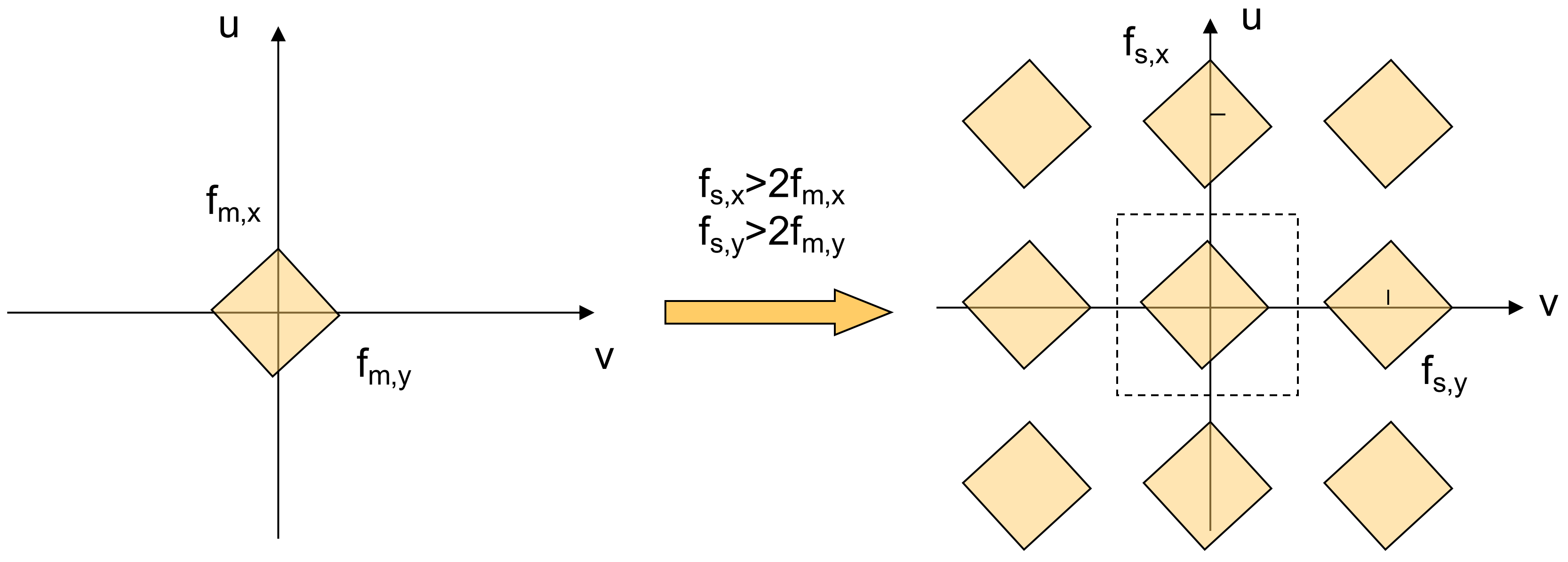
Reconstructed != original



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

Frequency Domain Interpretation of Sampling in 2D

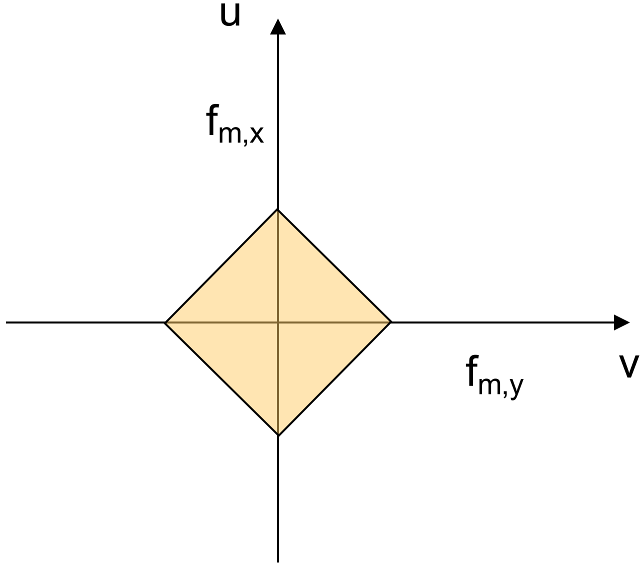
- The sampled signal contains replicas of the original spectrum shifted by multiples of sampling frequencies.



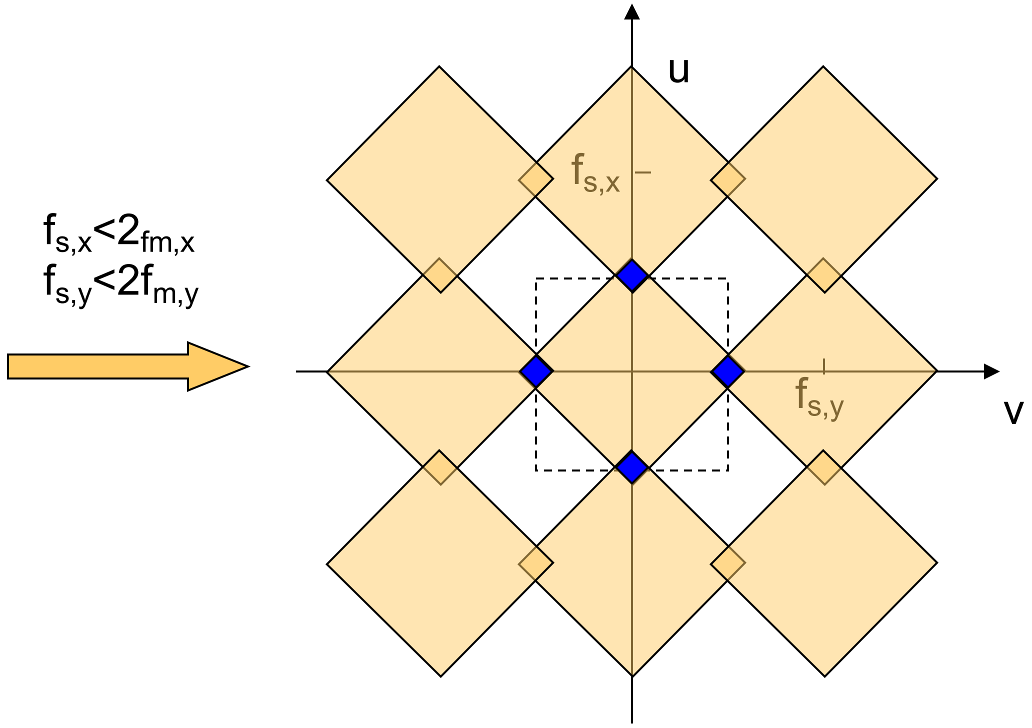
Original spectrum F(u,v)

Sampled spectrum $F_s(u,v)$

What Happens When You Under-sample?



$f_{s,x}<2f_{m,x}$
$f_{s,y}<2f_{m,y}$



Original spectrum F(u,v)

Sampled spectrum $F_s(u,v)$

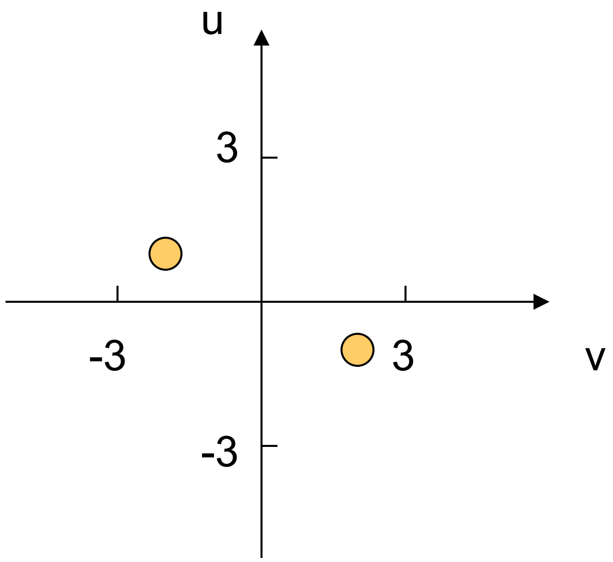
◆ Parts with aliasing

Sampling a Sinusoidal Signal

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

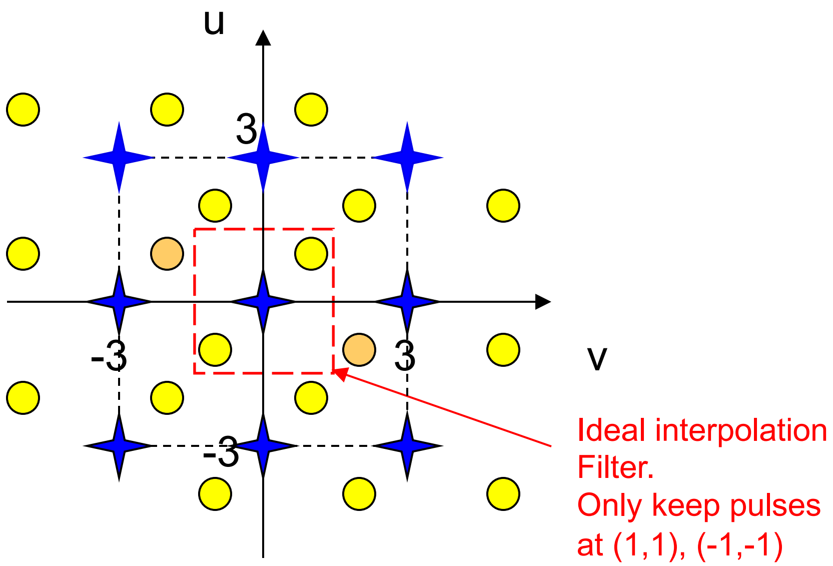
Sampled at $\Delta x = \Delta y = 1/3$ $\quad f_{s,x} = f_{s,y} = 3$

Original Spectrum



● Original pulse

Sampled Spectrum


u
3
-3
3
v
-3
Ideal interpolation Filter.
Only keep pulses at (1,1), (-1,-1)

● Replicated pulse

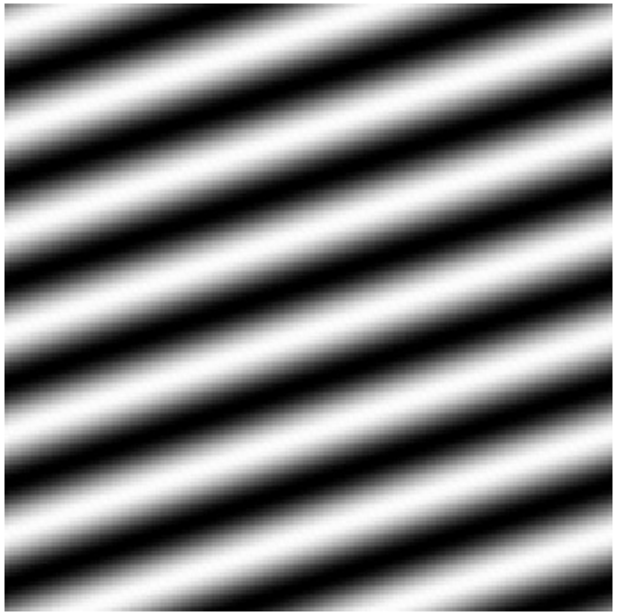
✦ Replication center, m f_{sx} , n f_{sy}

Reconstructed signal:

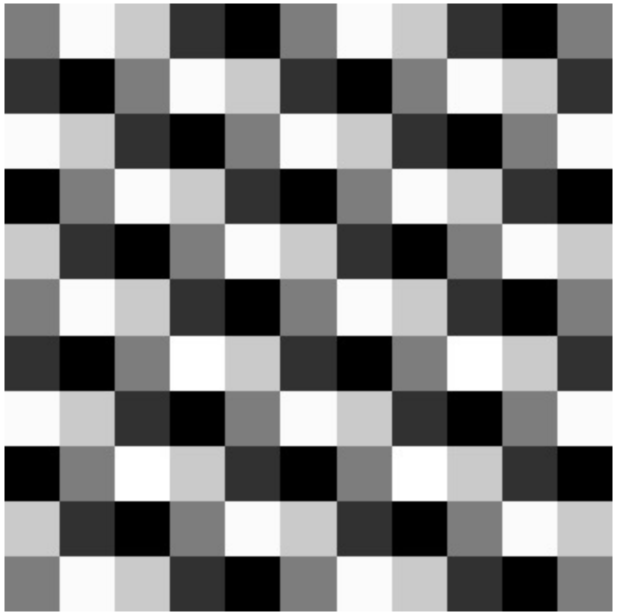
$$\hat{f}(x,y) = \cos(2\pi x + 2\pi y)$$

Different frequency and orientation from original!

Sampling in 2D: Sampling a 2D Sinusoidal Pattern



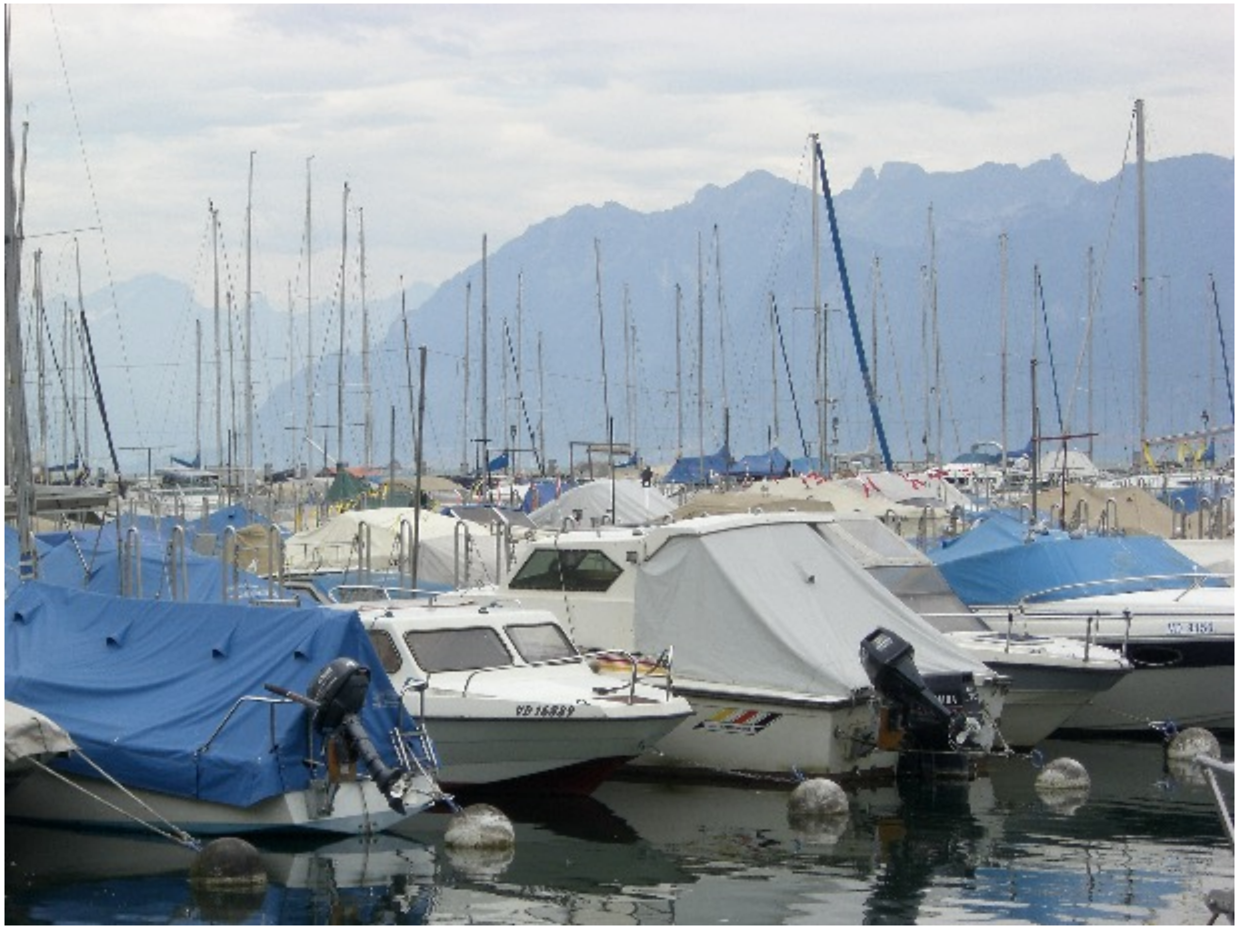
$f(x,y)=\sin(2*\pi*(3x+y))$
Sampling: $dx=0.01, dy=0.01$
$f_{x,max}=3$, $f_{y,max}=1$
$f_{s,x}=100 > 2\, f_{x,max}$, $f_{s,y}=100 > 2\, f_{y,max}$
Satisfying Nyquist rate
No aliasing!



$f(x,y)=\sin(2*\pi*(3x+y))$
Sampling: $dx=0.5, dy=0.5$
(Displayed with pixel replication)
$f_{s,x}=2 < 2\, f_{x,max}$, $f_{s,y}=2 = 2f_{y,max}$
Sampling at a rate lower than Nyquist rate
Aliasing!

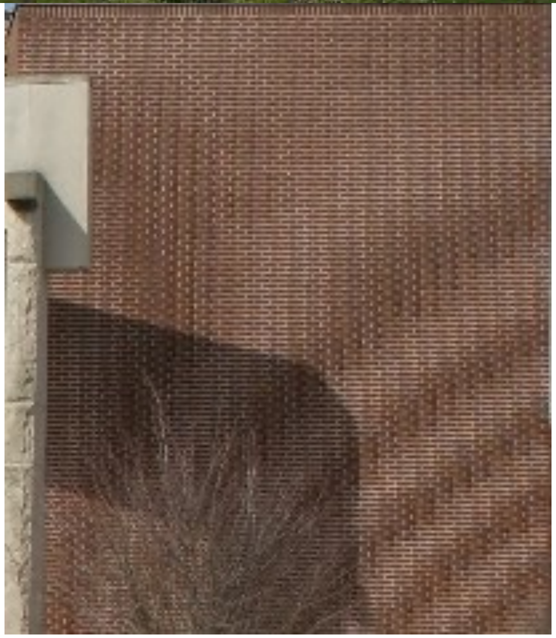
Aliasing causes changes in both frequency and direction of line patterns!

Aliasing in 2D



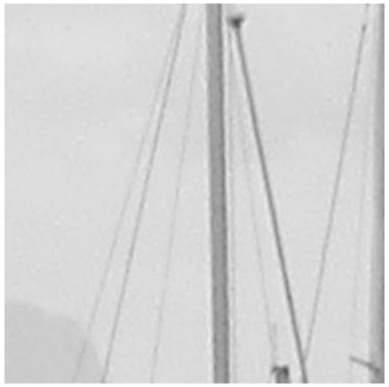
Example provided by Amy Reibman

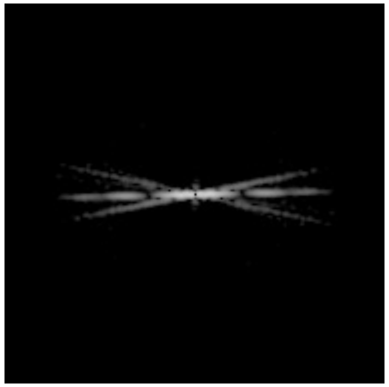
Pictures off the web



Example provided by Amy Reibman

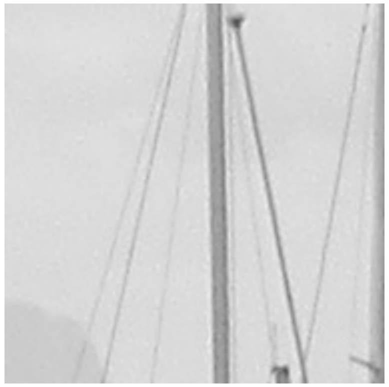
Fourier examples of aliasing: jaggies

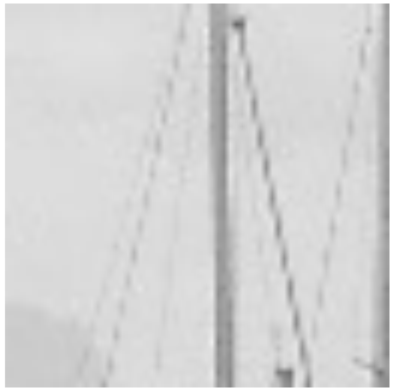


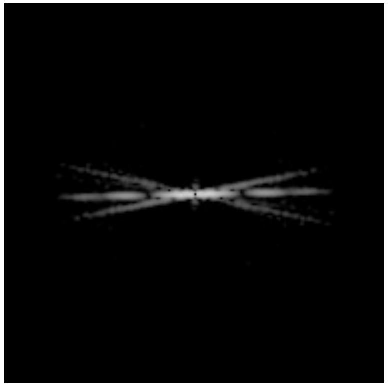


Example provided by Amy Reibman

Fourier examples of aliasing: jaggies

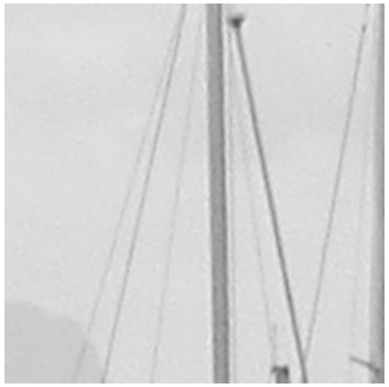


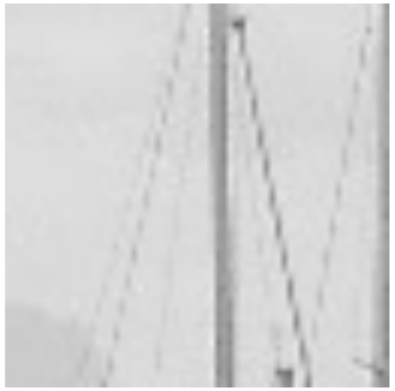


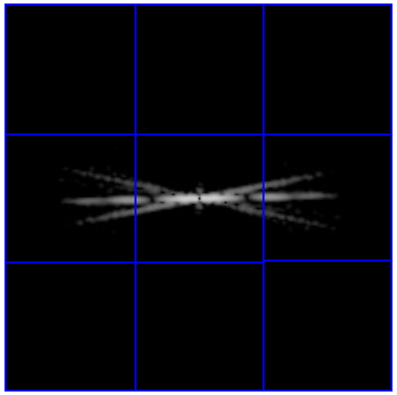


Example provided by Amy Reibman

Fourier examples of aliasing: jaggies

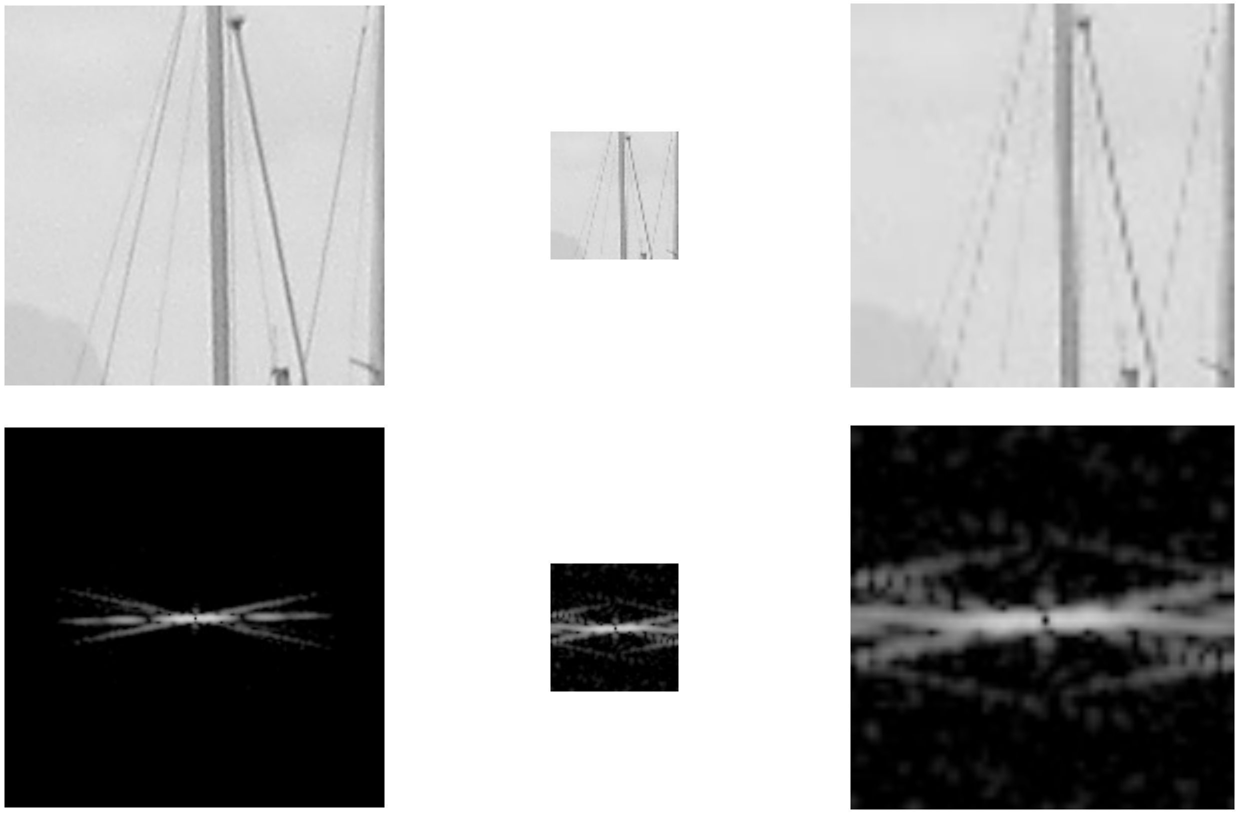






Example provided by Amy Reibman

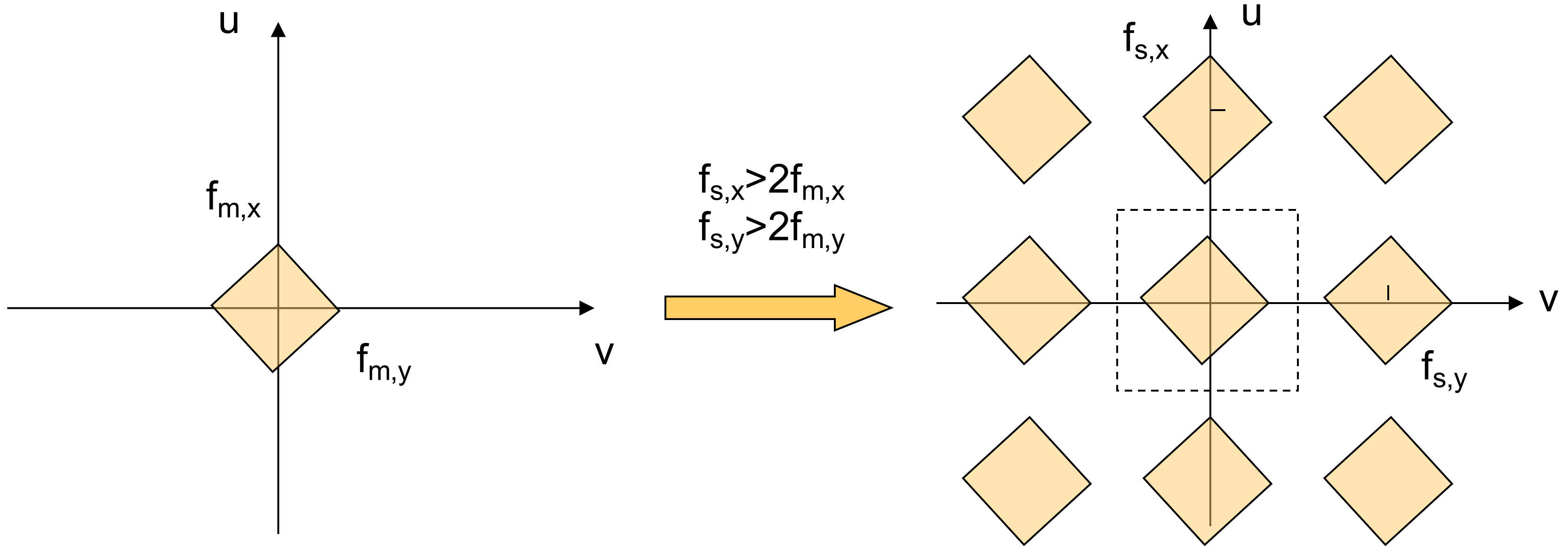
Fourier examples of aliasing: jaggies



Example provided by Amy Reibman

Frequency Domain Interpretation of Sampling in 2D

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



Original spectrum F(u,v)

Sampled spectrum $F_s(u,v)$

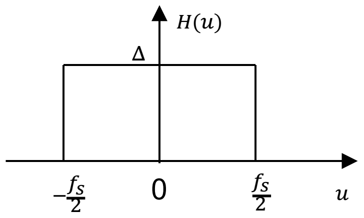
Nyquist Sampling and Reconstruction Theorem in Frequency Domain

- 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:

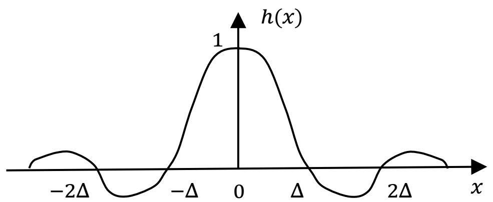
  $f_{s,x} > 2f_{m,x},\ f_{s,y} > 2f_{m,y}$

- 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 $\Delta x \Delta y$.

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



$f_s = \frac{1}{\Delta}$



Reconstruction in Spatial Domain

- The sampled signal can be treated as a continuous signal:

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

- Frequency domain multiplication = spatial domain convolution

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

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

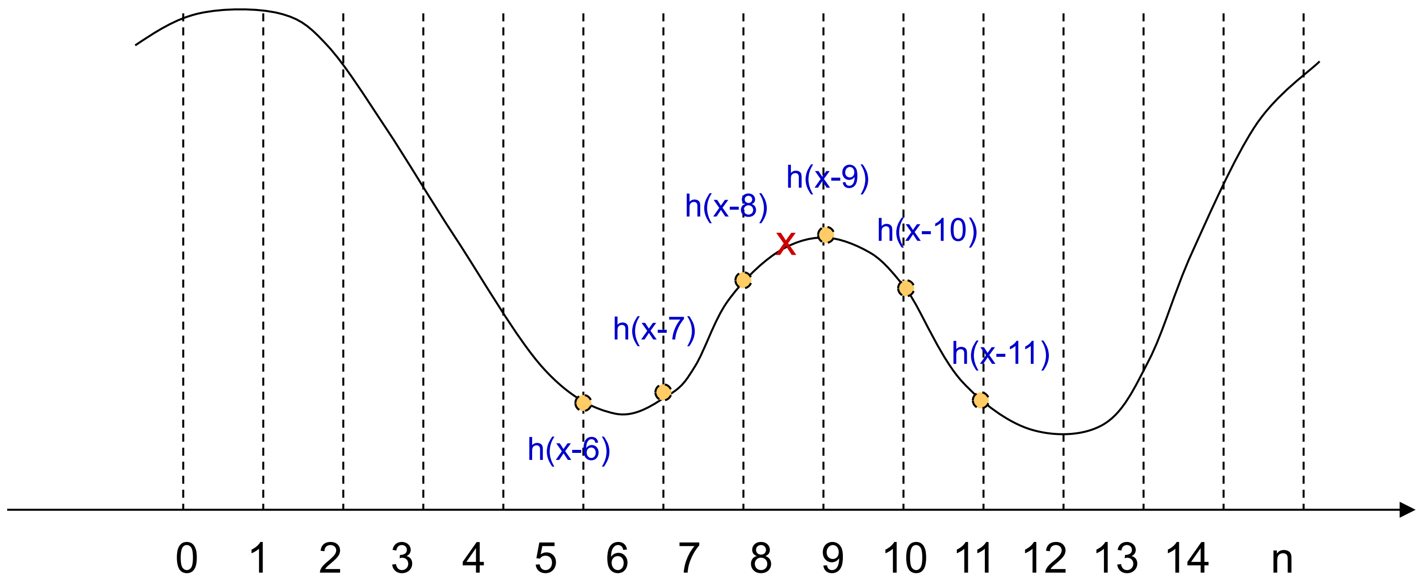
$$h(x,y)=\frac{\sin(\pi f_{s,x}x)}{\pi f_{s,x}x}\frac{\sin(\pi f_{s,y}y)}{\pi f_{s,y}y}$$

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

Interpretation as a weighted average of sample values

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

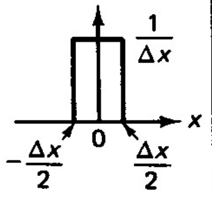
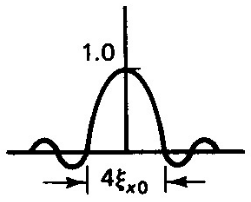
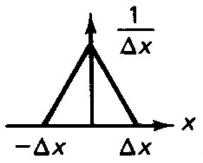
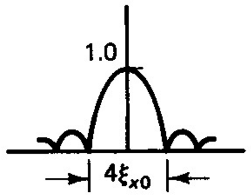
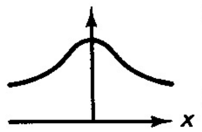
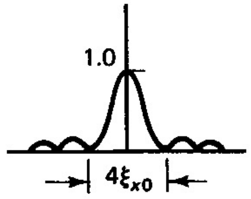
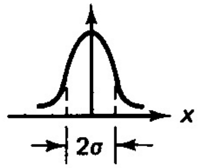
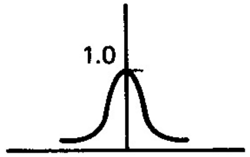
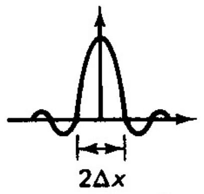
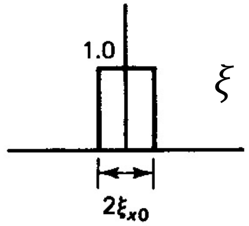
- The value at arbitrary point (x, y) is estimated from a weighted sum of the sample values at $(m\Delta_x, n\Delta_y)$ with weights $h(x-m\Delta_x, y-n\Delta_y)$



Desirable Properties of the Interpolation Kernel

- Should be a decreasing function of the distance
  - Higher weight for nearby samples
- Should be an even function of the distance
  - Left neighbor and right neighbor of same distance have the same weight
  - $h(x,y)=h(-x,-y)=h(-x,y)=h(x,-y)$
- Separable (to reduce computational complexity):
  - $h(x,y)=h_1(x)\,h_1(y)$
- To retain the original sample values, should have
  - $h(0,0)=1$, $h(m\Delta_x, n\Delta_y)=0$
- Should be close to ideal low-pass in the frequency domain!
- Such filters are called Nyquist Filters!
- Ideal filter (sinc function):
  - Satisfy all above properties

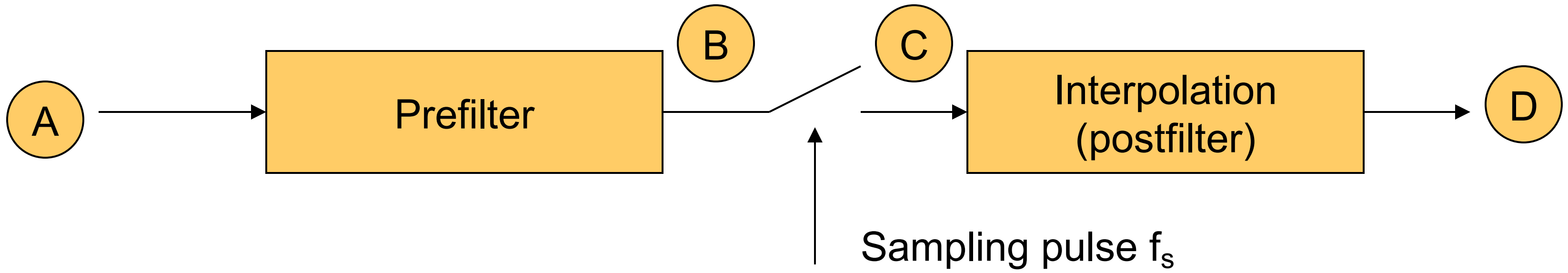
$$h(x,y)=\frac{\sin(\pi f_{s,x}x)}{\pi f_{s,x}x}\frac{\sin(\pi f_{s,y}y)}{\pi f_{s,y}y}$$

One-dimensional interpolation function	Diagram	Definition $p(x)$	Two-dimensional interpolation function $p_d(x, y) = p(x)p(y)$	Frequency response $P_d(\xi_1, \xi_2)$	$P_d(\xi_1, 0)$
Rectangle (zero-order hold) ZOH $p_o(x)$	 $\frac{1}{\Delta x}$ x 0 $-\frac{\Delta x}{2}$ $\frac{\Delta x}{2}$	$\frac{1}{\Delta x}\,\mathrm{rect}\left(\frac{x}{\Delta x}\right)$	$p_o(x)p_o(y)$	$\mathrm{sinc}\left(\frac{\xi_1}{2\xi_{x0}}\right)\mathrm{sinc}\left(\frac{\xi_2}{2\xi_{y0}}\right)$	
Triangle (first-order hold) FOH $p_1(x)$		$\frac{1}{\Delta x}\,\mathrm{tri}\left(\frac{x}{\Delta x}\right)$ $p_o(x) \circledast p_o(x)$	$p_1(x)p_1(y)$	$\left[\mathrm{sinc}\left(\frac{\xi_1}{2\xi_{x0}}\right)\mathrm{sinc}\left(\frac{\xi_2}{2\xi_{y0}}\right)\right]^2$	
nth-order hold $n = 2$, quadratic $n = 3$, cubic splines $p_n(x)$		$p_o(x) \circledast \cdots \circledast p_o(x)$ n convolutions	$p_n(x)p_n(y)$	$\left[\mathrm{sinc}\left(\frac{\xi_1}{\xi_{x0}}\right)\mathrm{sinc}\left(\frac{\xi_2}{\xi_{y0}}\right)\right]^{n+1}$	
Gaussian $p_g(x)$		$\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{x^2}{2\sigma^2}\right]$	$\frac{1}{2\pi\sigma^2}\exp\left[-\frac{(x^2+y^2)}{2\sigma^2}\right]$	$\exp\left[-2\pi^2\sigma^2(\xi_1^2+\xi_2^2)\right]$	
Sinc		$\frac{1}{\Delta x}\,\mathrm{sinc}\left(\frac{x}{\Delta x}\right)$	$\frac{1}{\Delta x \Delta y}\,\mathrm{sinc}\left(\frac{x}{\Delta x}\right)\mathrm{sinc}\left(\frac{x}{\Delta y}\right)$	$\mathrm{rect}\left(\frac{\xi_1}{2\xi_{x0}}\right)\mathrm{rect}\left(\frac{\xi_2}{2\xi_{y0}}\right)$	

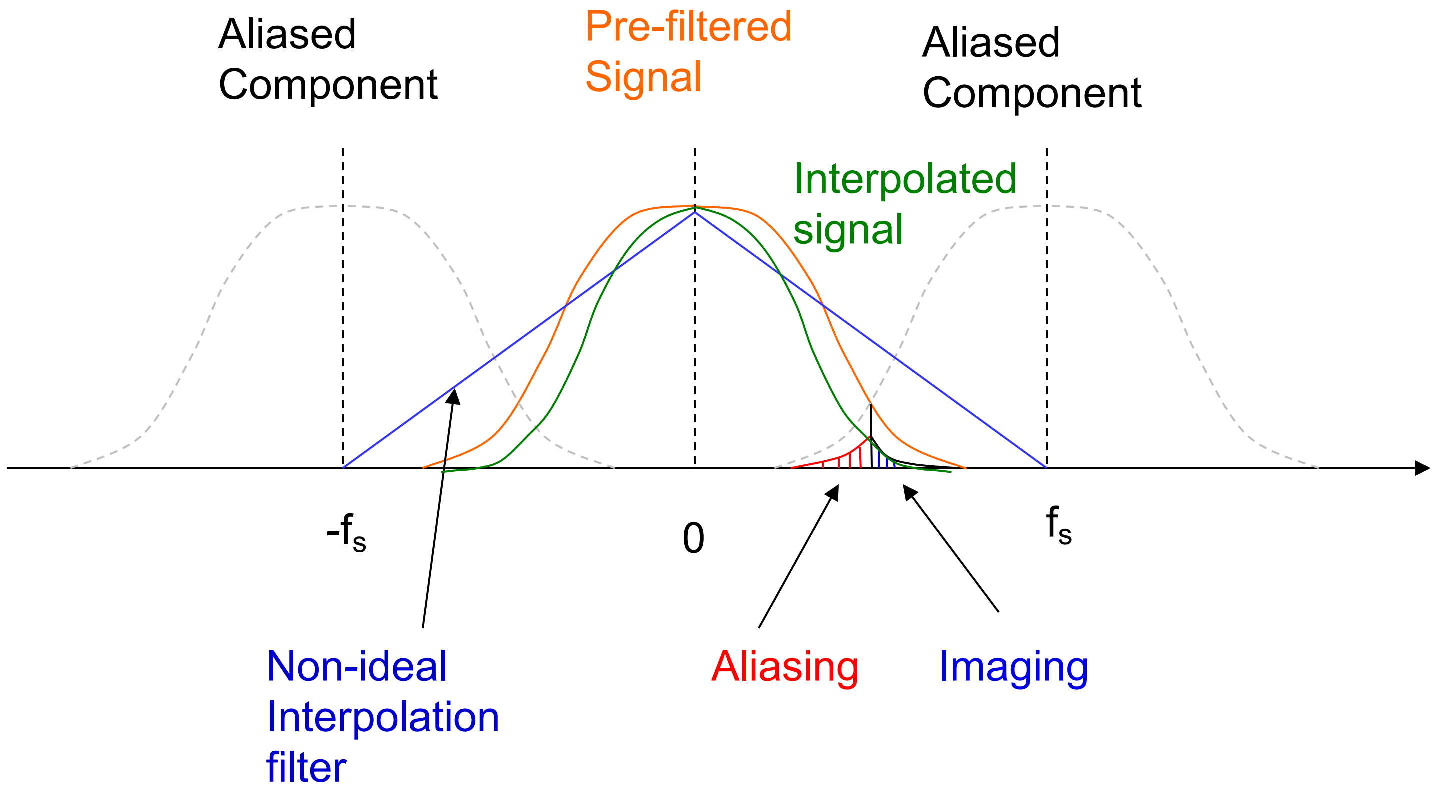
$\xi = \frac{f_S}{2}$

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

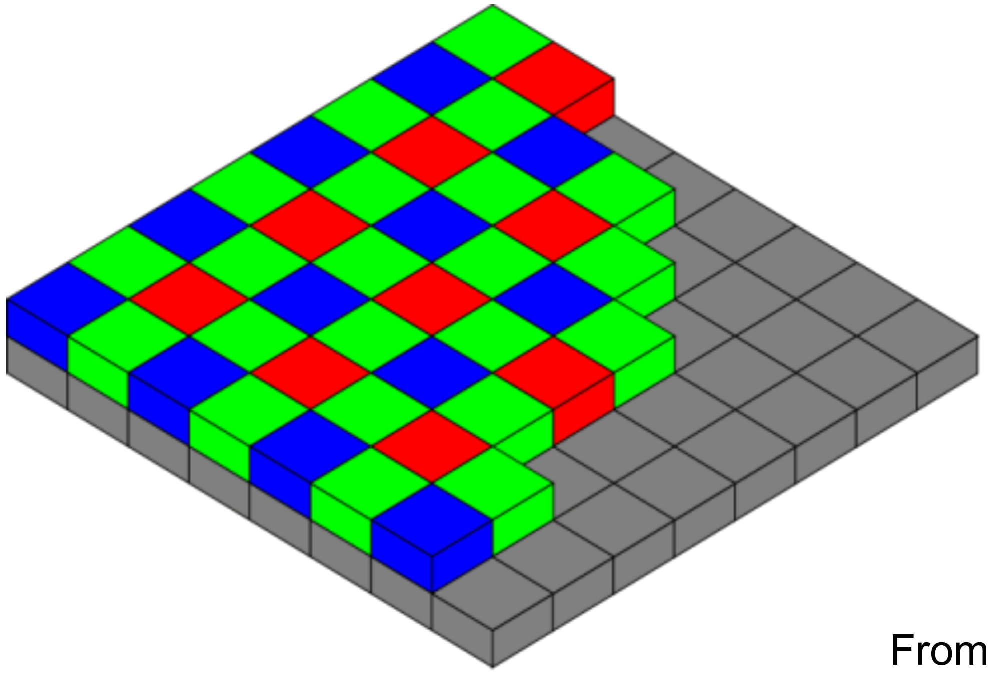
What happens in digital cameras?

- Consider a Black and White camera first!
- The CCD sensor array contains a sensor for each pixel
- The value recorded for each pixel is roughly the sum of the light entering the sensor (after filtering using the corresponding color filter)
  - Each pixel value = integration of the light over the sensor aperture (roughly a small square area)
- Combining prefiltering and sampling in one step!
- Prefilter = ?
  - A averaging filter over sampling interval

- What about color cameras?

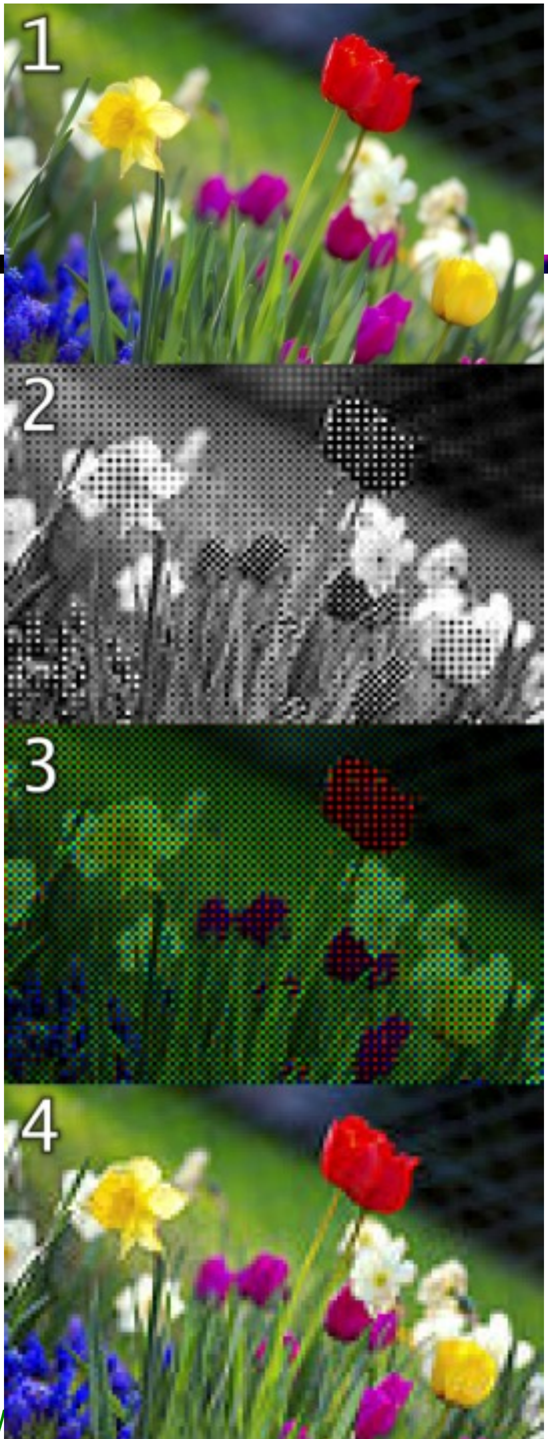
Color Imaging Using Color Filter Arrays

- Single sensor array, with different color filters to separate the RGB primary components
- Sensors: CCD, CMOS



Bayer RGB Pattern

From http://en.wikipedia.org/wiki/Bayer_filter



From http://en.wikipedia.org/wiki/Bayer_filter

1: original scene
2: output of a 120x80 pixel sensor with a Bayer filter
3: output color coded
4: Reconstructed image after interpolating missing colors (demosaicing)

What about video?

- Sampling in 3D!
- Frame rate = sampling freq. in time
- Nyquist sampling theorem applies in each dimension
- Must sample sufficiently fast to avoid temporal aliasing!

Aliasing in Video

- Aliasing makes faster moving objects appear slower and moving in different directions!
- Example: Fast moving wheels appear turning backward
- Other example:
- https://www.red.com/red-101/cinema-temporal-aliasing
  - Containing other interesting facts and solutions!

Who does the interpolation when a a digital image or video is displayed?

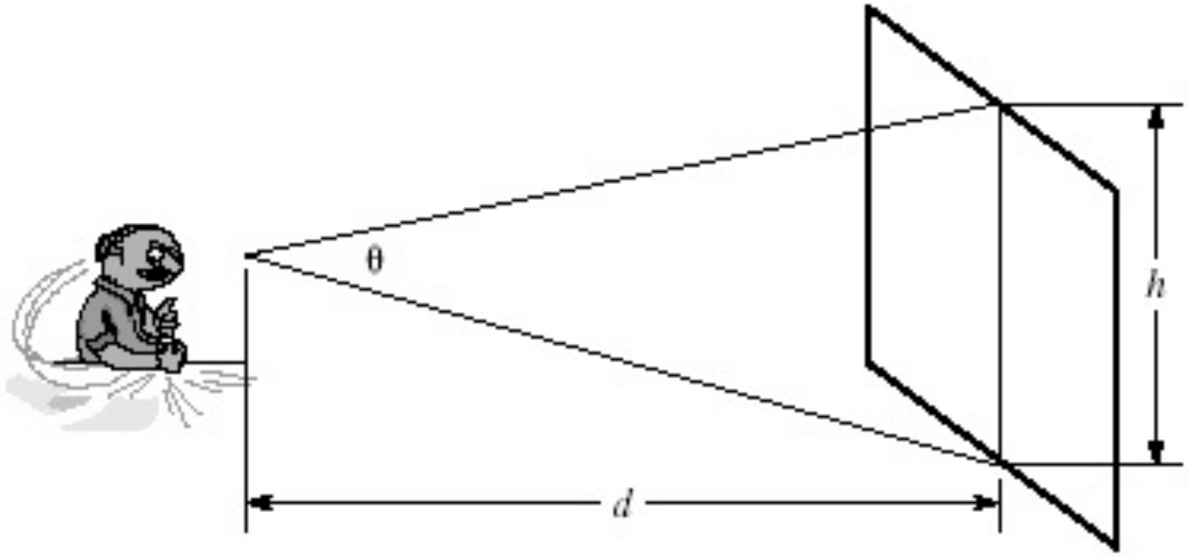
- Our eye!
- We can blend in discrete pixels as if there are covering a continuous plane if the pixels are not too far away!
- Our eye/brain is essentially applying the low pass filtering to interpolate a continuous 2D function from discrete pixels
- We will see pixilation effect if the pixels are too far apart (i.e. the image has low resolution).
- What about video?
- Videos are captured and displayed in separate frames
- Our eye blend in adjacent frames so that they look to appear continuously, if the frame rate is sufficiently high!

Video Cameras

- Sampling mechanism
  - All perform sampling in time (characterized by frame rate=frames/s)
  - Old film cameras capture continuous frames on film
  - Old analog video cameras sample in vertical but not horizontal direction, using either progressive or interlaced scan.
  - Digital cameras sample in both horizontal and vertical direction, yielding pixels with discrete 3-D coordinates
- How to determine necessary frame rate, line rate, and pixels/line?
  - Depending on display size, viewing distance, maximum frequency in the underlying signal, the maximum frequency that the HVS can detect (visual freq. thresholds), as well as technical feasibility and cost
  - If technically feasible, should sample at twice of visual freq. threshold!

Angular Frequency

Perceived spatial frequency (cycles/viewing-angle or cpd) depends on viewing distance



$$\theta = 2\arctan(h/2d)(\text{radian}) \approx 2\text{h}/2\text{d}(\text{radian}) = \frac{180}{\pi}\frac{h}{d}(\text{degree})$$

$$\text{f}_\theta = \frac{\text{f}_s}{\theta} = \frac{\pi}{180}\frac{d}{h}\text{f}_s(\text{cycle/degree})$$

Frequency Response of the HVS (Human Sensitivity to Different Frequency Components)

- Temporal frequency response and flicker
- Spatial frequency response
- Spatio-temporal response
- Smooth pursuit eye movement

Spatial Contrast Sensitivity Function

- Display the following signal (vertical sinusoidal bars)

$$\psi(x,y,t)=B(1+m\cos 2\pi fx)$$

- B brightness, f=spatial frequency, m=modulation level
- Given f, what is minimum modulation level m_{min} at which sinusoidal grating is visible?
- 1/ m_{min} at a given frequency f is the *spatial contrast sensitivity at f*
- Contrast sensitivity function is also known as the Modulation Transfer Function (MTF) of the human eye

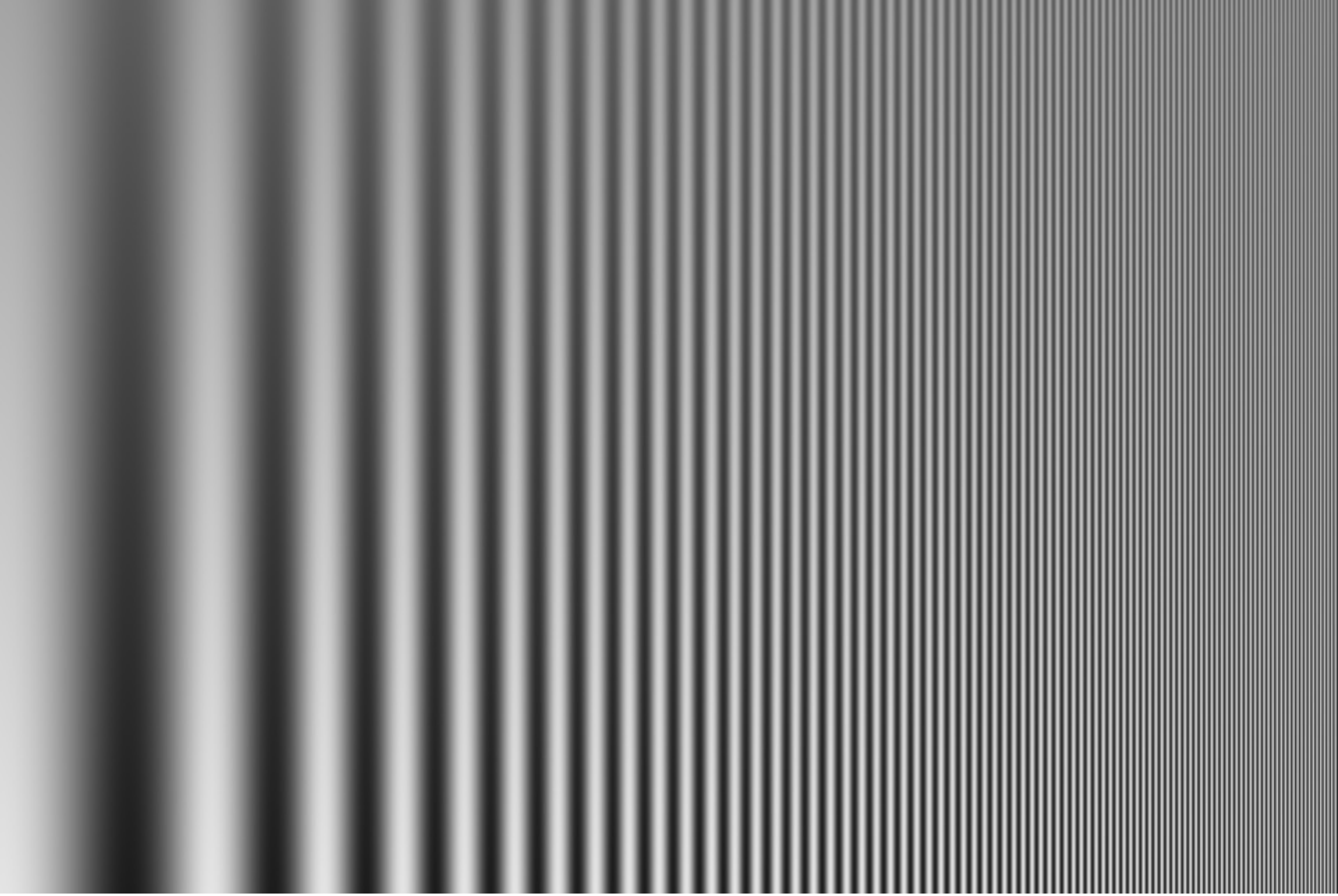
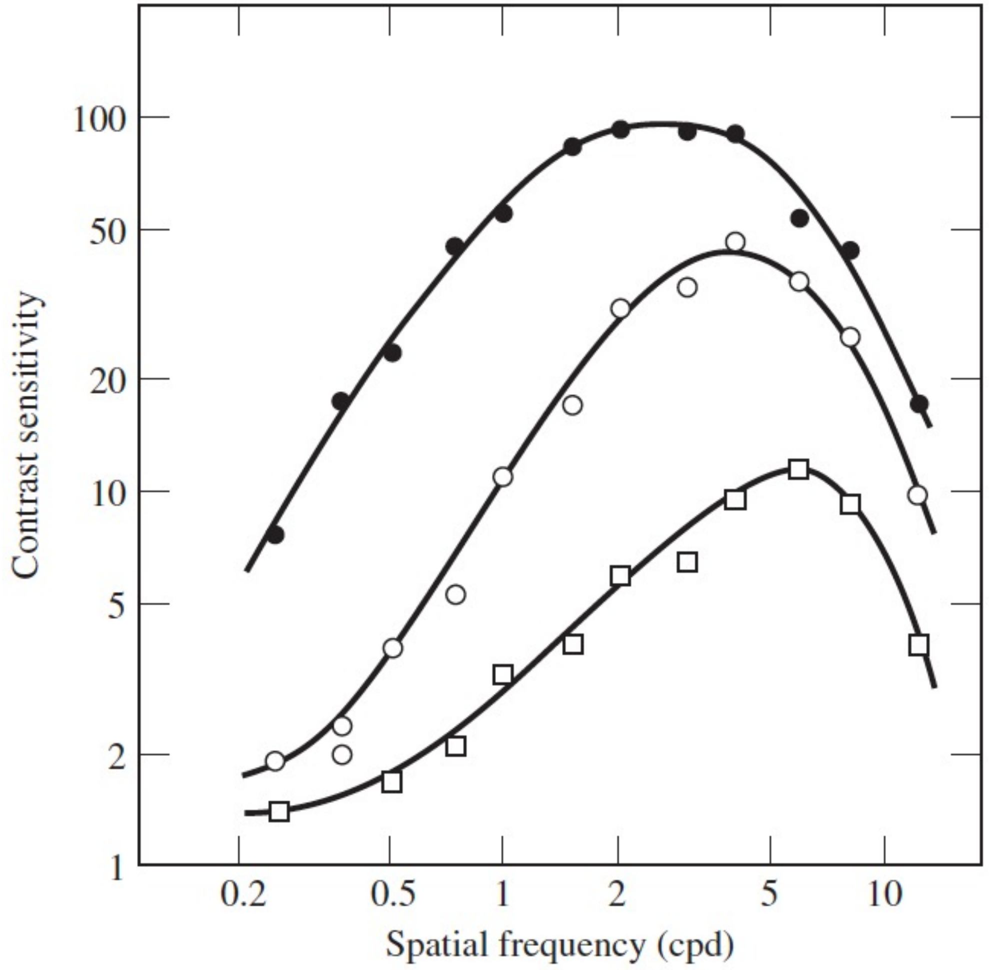


Image provided by Amy Reibman

Spatial Frequency Response of HVS



From [Wang2002]

Figure 2.6 The spatial frequency response of the HVS, obtained by a visual experiment. The three curves result from different stabilization settings used to remove the effect of saccadic eye movements. Filled circles were obtained under normal, unstablized conditions; open squares, with optimal gain setting for stabilization; open circles, with the gain changed about 5 percent. Reprinted from D. H. Kelly, Motion and vision. I. Stabilized images of stationary gratings, *J. Opt. Soc. Am.* (1979), 69:1266–74, by permission of the Optical Society of America.

HVS is most sensitive around 3-5 cpd, and is sensitive up to 30 cpd.

Temporal Contrast Sensitivity Function

- Display the following signal

$$\psi(t,x,y) = B(1 + m\cos 2\pi f t)$$

- B brightness, f =temporal frequency, m=modulation level
- What is minimum modulation level at which sinusoidal grating is visible?
- $1/m_{\min}$ at a given frequency f is the *temporal contrast sensitivity at f*

Temporal Frequency Response of HVS (depends on mean display brightness)

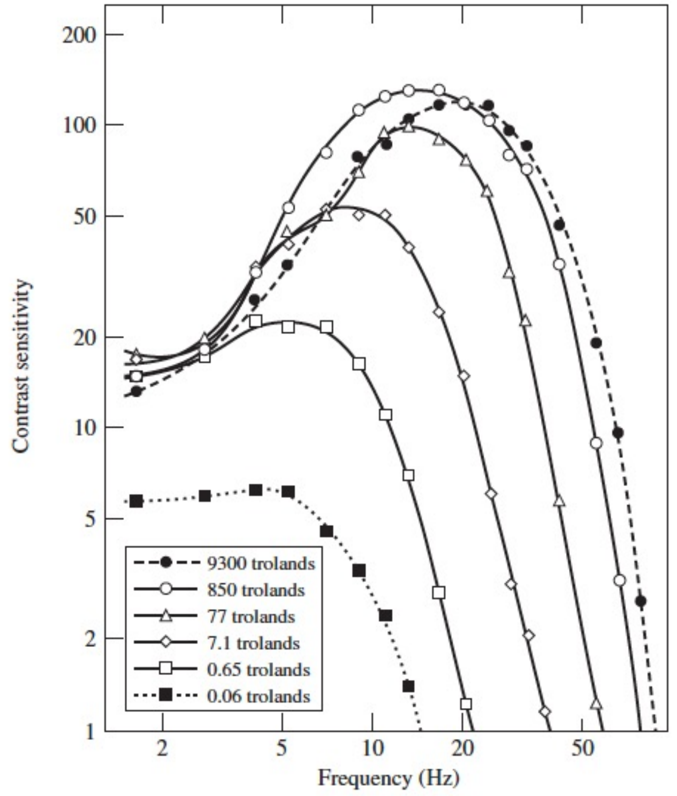


Figure 2.5 The temporal frequency response of the HVS obtained by a visual experiment. Different curves represent the responses obtained with different mean brightness levels, *B*, measured in trolands. The horizontal axis represents the flicker frequency *f*, measured in Hz. Reprinted from D. H. Kelly, Visual responses to time-dependent stimuli. I. Amplitude sensitivity measurements, *J. Opt. Soc. Am.* (1961) 51:422–29, by permission of the Optical Society of America.

Critical flicker frequency: The lowest frame rate at which the eye does not perceive flicker.

Provides guideline for determining the frame rate when designing a video system.

Critical flicker frequency depends on the mean brightness of the display:

60 Hz is typically sufficient for watching TV.

Watching a movie needs lower frame rate than TV because movie theater uses less bright display

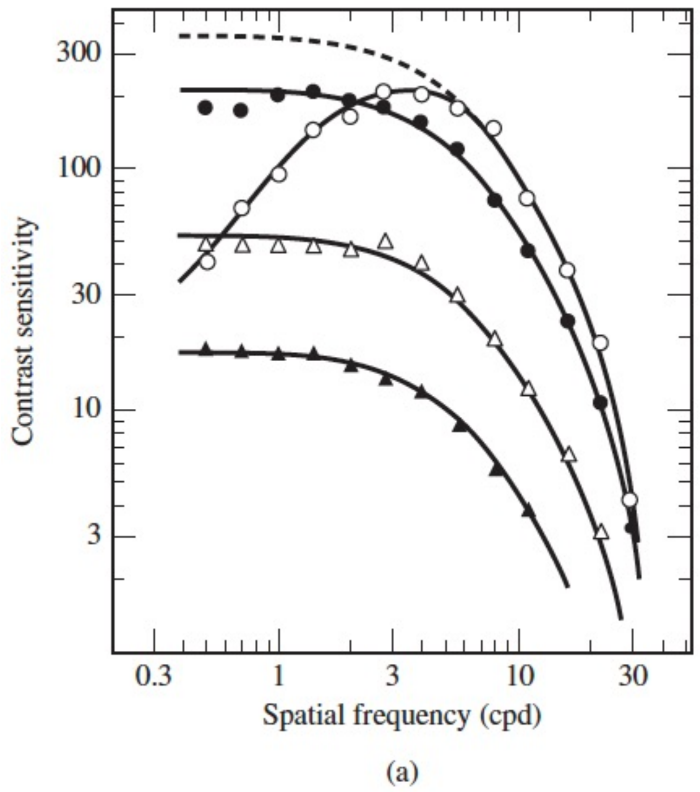
HVS is most sensitive around 10-20 Hz, and is sensitive up to 75 Hz.

Spatio-Temporal Contrast Sensitivity

- Repeat the previous experiment using the following signal, determine minimal m for each f_x and f_y

$$\psi\left(x,y,t\right)=B\left(1+m\cos(2\pi f_x x)\cos(2\pi f_t t)\right)$$

Spatiotemporal Response



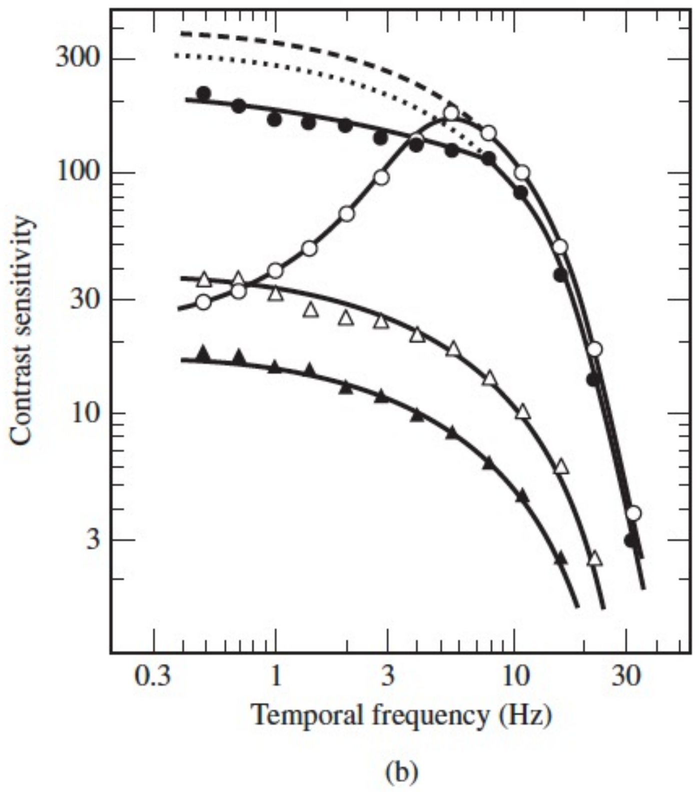


Figure 2.7 Spatiotemporal frequency response of the HVS. (a) Spatial frequency responses for different temporal frequencies of 1 Hz (open circles), 6 Hz (filled circles), 16 Hz (open triangles), and 22 Hz (filled triangles). (b) Temporal frequency responses for different spatial frequencies of 0.5 cpd (open circles), 4 cpd (filled circles), 16 cpd (open triangles), and 22 cpd (filled triangles). Reprinted from J. G. Robson, Spatial and temporal contrast sensitivity functions of the visual systems, *J. Opt. Soc. Am.* (1966), 56:1141–42, by permission of the Optical Society of America.

Temporal masking (a):

When temporal frequency is high (moving fast), we have lower sensitivity to spatial details.

Exploited in Interlaced scan in analog TV system:

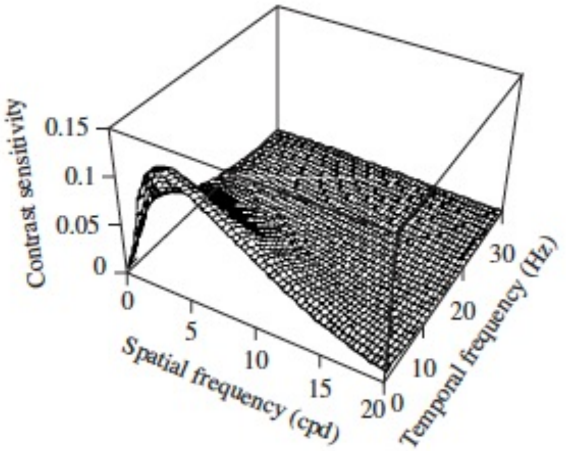
Stationary scene enjoys high vertical resolution

Moving scene has lower vertical resolution

Smooth Pursuit Eye Movement

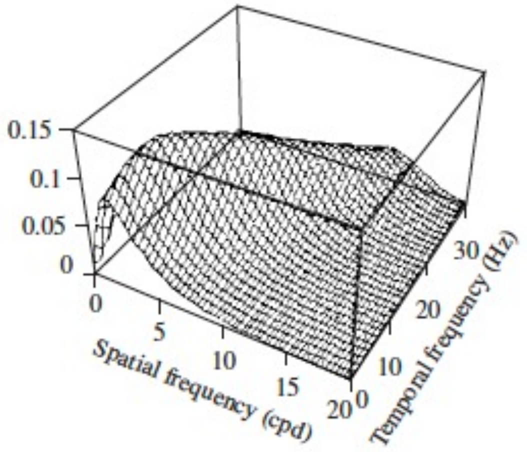
- Smooth Pursuit: the eye tracks moving objects
- Net effect: reduce the velocity of moving objects on the retinal plane, so that the eye can perceive much higher raw temporal frequencies than indicated by the temporal frequency response.

No SPEM

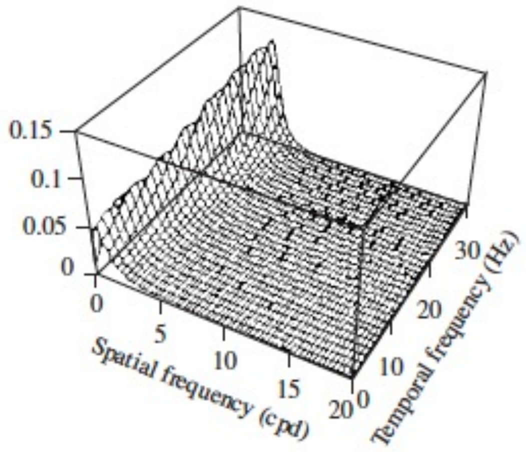


(a)

SPEM
2 deg/s



(b)



(c)

SPEM
10 deg/s

Figure 2.8 Spatiotemporal response of the HVS under smooth pursuit eye movements: (a) without smooth pursuit eye movement; (b) with eye velocity of 2 deg/s; (c) with eye velocity of 10 deg/s. Reprinted from Girod, B. "Motion compensation: visual aspects, accuracy, and fundamental limits." In Sezan, M. I., and R. L. Lagendijk, eds., *Motion Analysis and Image Sequence Processing,* Boston: Kluwer Academic Publishers, 1993, 126–52, by permission of Kluwer Academic Publishers.

Necessary Sampling Rates

- Frame rate (frames/sec):
  - Human eye can see up to 60 Hz
  - Even higher to account for smooth pursuit eye movement
  - Fast moving objects can lead to temporal freq. >60 cycles/s
  - Ideally video should be captured at >=120 Hz
- Pixels/line:
  - Should portrait at least 30 cpd in angular frequency, requiring sampling at >=2*30 = 60 samples/viewing degree! (Retina display is defined as >=53 cpd, mostly >70)
  - Depends on the viewing angle span (which depends on the display monitor width, typical viewing distance)
- Lines/frame:
  - Depends on the desired aspect ratio (=ratio of width to height)
  - Wide aspect ratio (16:9, HDTV) is preferred over standard (4:3, SDTV, analog TV)

Examples

$$\theta = 2\arctan(h/2d)(\text{radian}) \approx 2\text{h}/2\text{d}(\text{radian}) = \frac{180}{\pi}\frac{h}{d}(\text{degree})$$

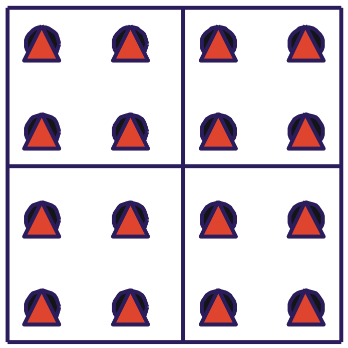
$$\text{f}_\theta = \frac{\text{f}_s}{\theta} = \frac{\pi}{180}\frac{d}{h}\text{f}_s(\text{cycle/degree})$$

- Example 1 (watching TV), 1 m wide screen, viewed from 3m away
  - Horizontal view angle = 180 * 1/ (3.14*3) = 19 (degree)
  - To display up to 30 cpd, require fs=60 * 19= 1200 pixels/width
- Example 2 (using computer), 1 ft wide screen, viewed from 1 ft
  - Horizontal view angle = 180 * 1/ (3.14*1) = 57 (degree)
  - To display up to 30 cpd, require fs=60 * 57 ~= 3600 pixels/width (~Apple Retina Display!)

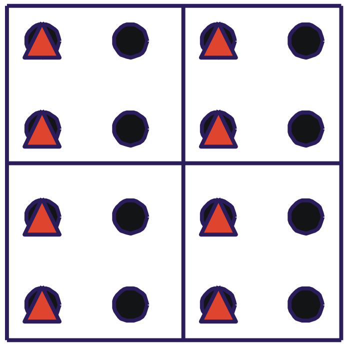
Example Video Formats

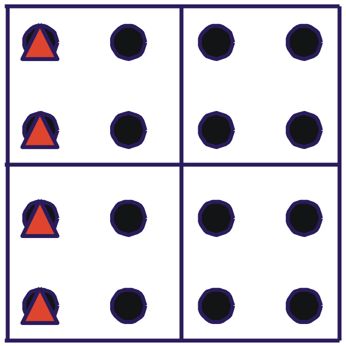
- Old analog TV system:
  - NTSC: 60 fields/sec (60i), 525 lines/frame (only 480 active lines)
  - PAL/SECAM: 50 fields/sec (50i), 625 lines/frame (only 576 active lines)
- BT601 Digital video (digitized version of NTSC/PAL)
  - NTSC->60i, 720x480, data rate=720*480*30*24=249Mbps
  - PAL/SECAM-> 50i, 720x576, data rate=720*576*50*24=249Mbps
- SD video format
  - 4CIF: 60 frames/sec, 720x480/frame, data rate=720*480*60*24=498Mbps
- HDTV (16:9 aspect ratio)
  - 1080p: 60 frames/sec, 1920x1080/frame, data rate=3 Gbps
- Ultra HD:
  - 4K: 60 frames/sec, 4096x2180/frame, data rate=12.9 Gbps
- Above assumes the three color components (e.g. YCbCr) are represented with the same resolution (4:4:4 format)
- Because the human eye has reduced sensitivity to chrominance components, we can downsample the CbCr components to reduce the data rate!

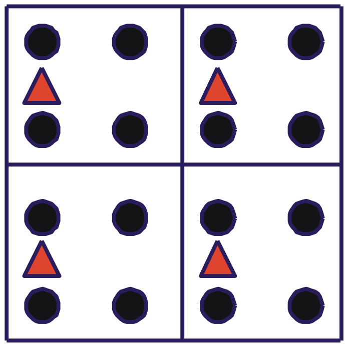
Chrominance Subsampling Formats


4:4:4
For every 2x2 Y Pixels
4 Cb & 4 Cr Pixel
(No subsampling)

Professional video


4:2:2
For every 2x2 Y Pixels
2 Cb & 2 Cr Pixel
(Subsampling by 2:1 horizontally only)


4:1:1
For every 4x1 Y Pixels
1 Cb & 1 Cr Pixel
(Subsampling by 4:1 horizontally only)


4:2:0
For every 2x2 Y Pixels
1 Cb & 1 Cr Pixel
(Subsampling by 2:1 both horizontally and vertically)

Consumer video

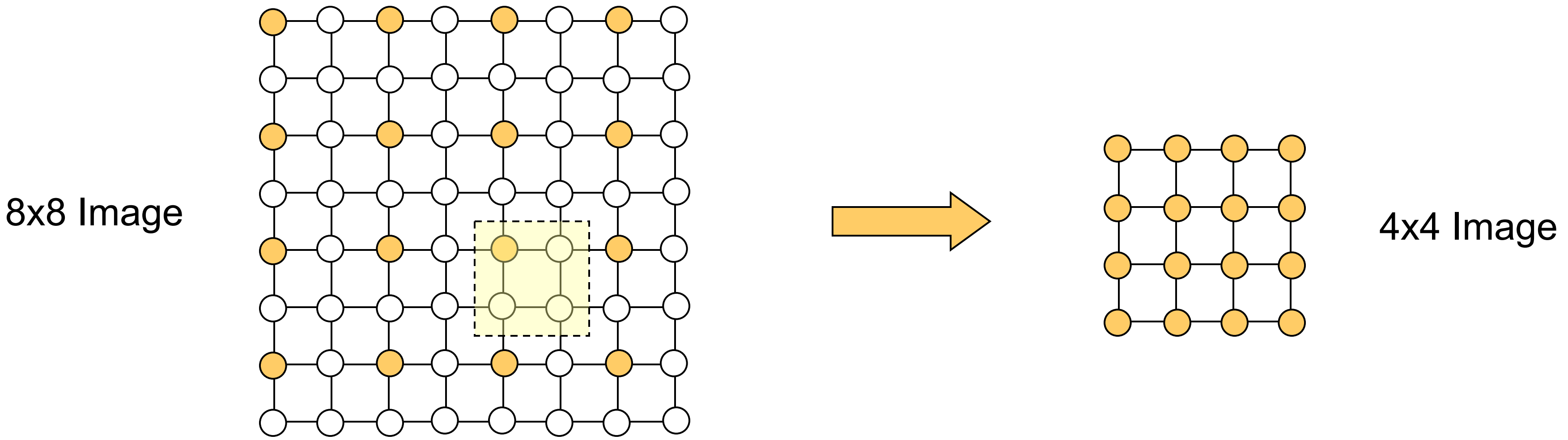
● Y Pixel

▲ Cb and Cr Pixel

From Sampling to Resizing

- Sampling deals with converting a continuous space/time signal to a discrete space/time signal
- Resizing refers to changing the sampling rate of a discrete space/time signal
- Image down-sampling (resample to a lower rate)
- Image up-sampling (resample to a higher rate)
- Frame rate conversion (resample in time)
- Necessary for scaling of image sizes
- Necessary for displaying videos on devices using different frame rate than the video camera

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 * 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 ($f_{s,new}/2$).
  - In continuous domain, the cutoff frequency is $f_{s,old}/4$
  - In digital frequency, $f_{s,old}$ corresponds to 1, the cutoff frequency is ¼.
- The ideal low pass filter is not realizable. We can design filters to approximate the ideal filter.
- In practice, for image down sampling, we use relatively short filters

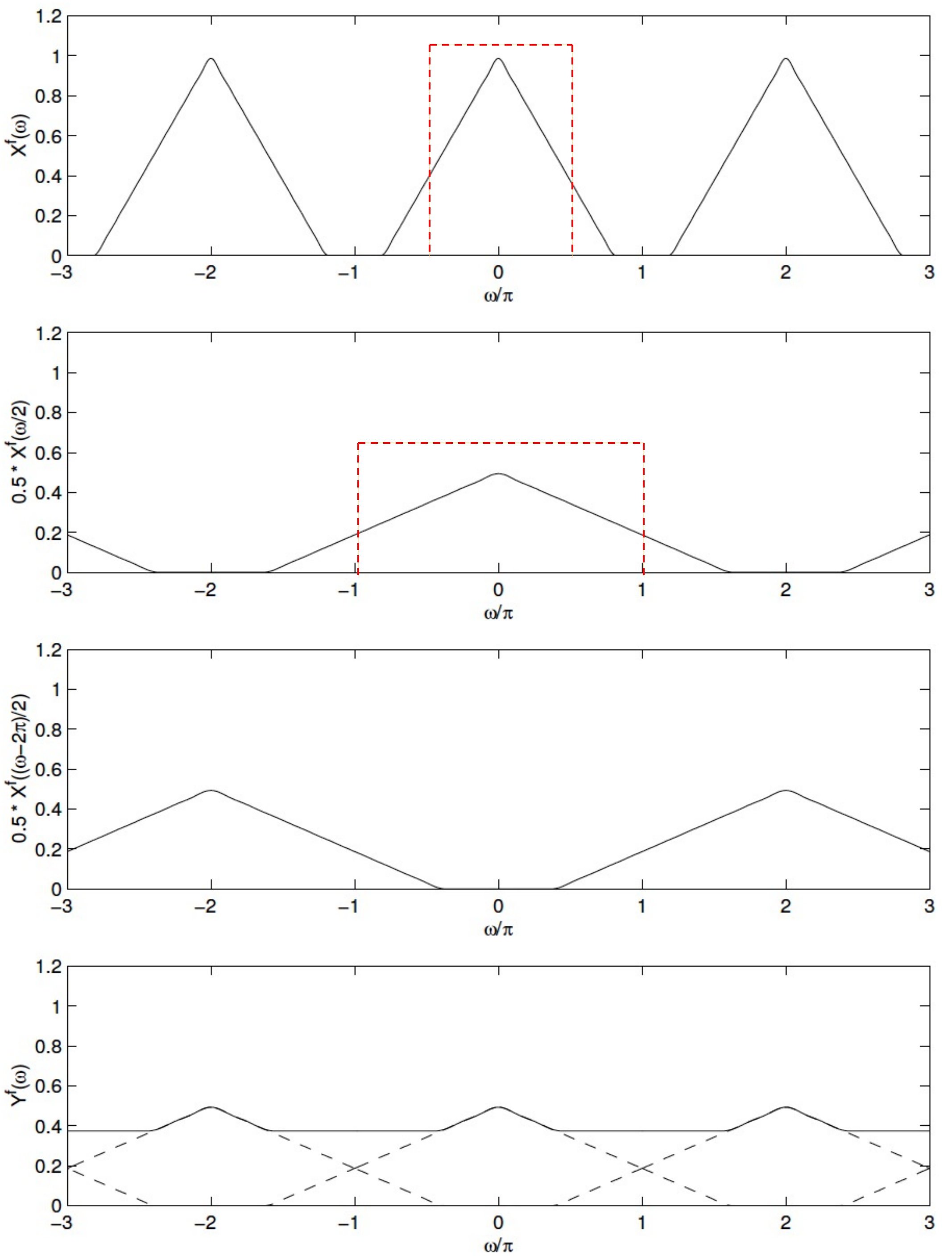
Frequency Domain Interpretation (1D)

$$y(n) = [\downarrow 2]\, x(n)$$

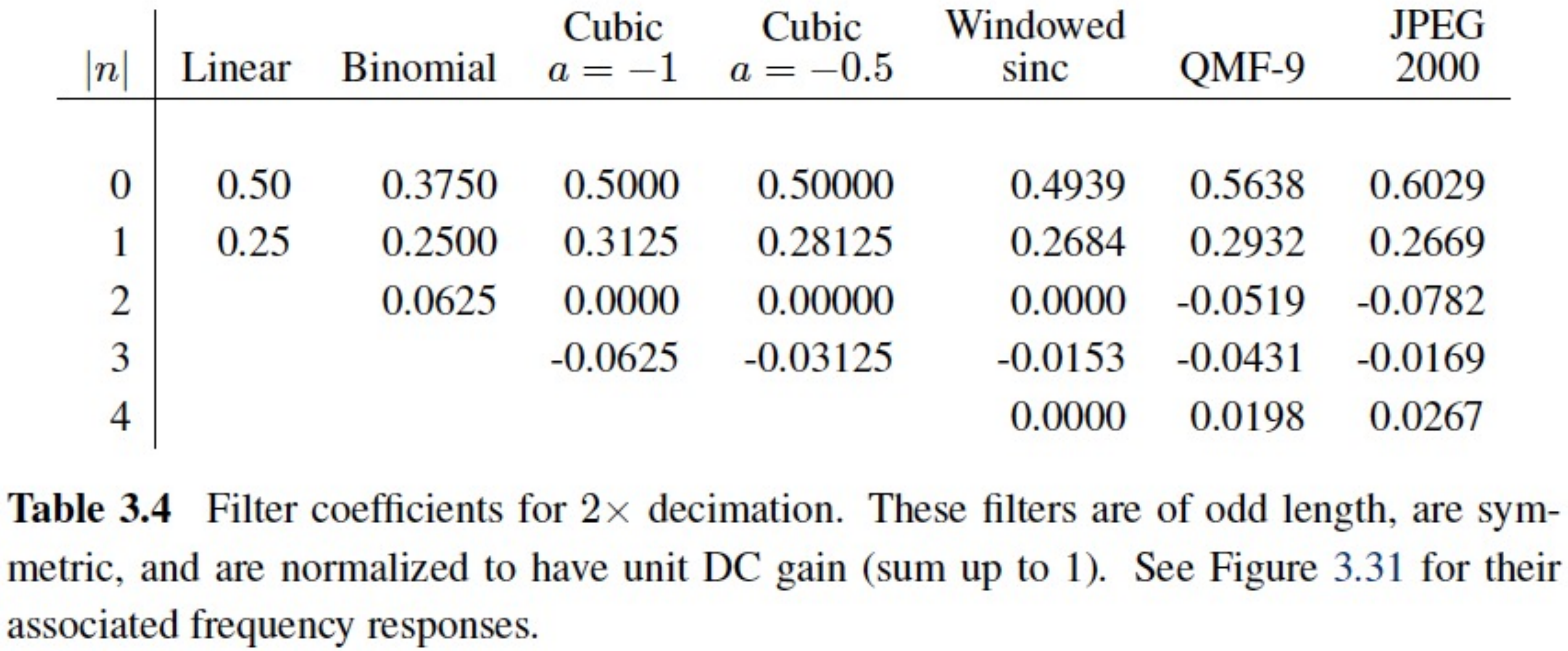
$$Y(z) = \frac{1}{2}\left[X(z^{\frac{1}{2}}) + X(-z^{\frac{1}{2}})\right]$$

$$Y^f(\omega) = \frac{1}{2}\left[X^f\left(\frac{\omega}{2}\right) + X^f\left(\frac{\omega - 2\pi}{2}\right)\right]$$

To avoid aliasing, should apply a half band filter before downsampling!



Common Prefilters for Downsampling by 2



			Cubic	Cubic	Windowed		JPEG
$\lvert n \rvert$	Linear	Binomial	$a = -1$	$a = -0.5$	sinc	QMF-9	2000
0	0.50	0.3750	0.5000	0.50000	0.4939	0.5638	0.6029
1	0.25	0.2500	0.3125	0.28125	0.2684	0.2932	0.2669
2		0.0625	0.0000	0.00000	0.0000	-0.0519	-0.0782
3			-0.0625	-0.03125	-0.0153	-0.0431	-0.0169
4					0.0000	0.0198	0.0267

Table 3.4 Filter coefficients for $2\times$ decimation. These filters are of odd length, are symmetric, and are normalized to have unit DC gain (sum up to 1). See Figure 3.31 for their associated frequency responses.

Ex: Binomial filter is [0.0625, 0.2500, 0.3750, 0.2500, 0.0625] with origin at the center.

From [Szeliski2012]

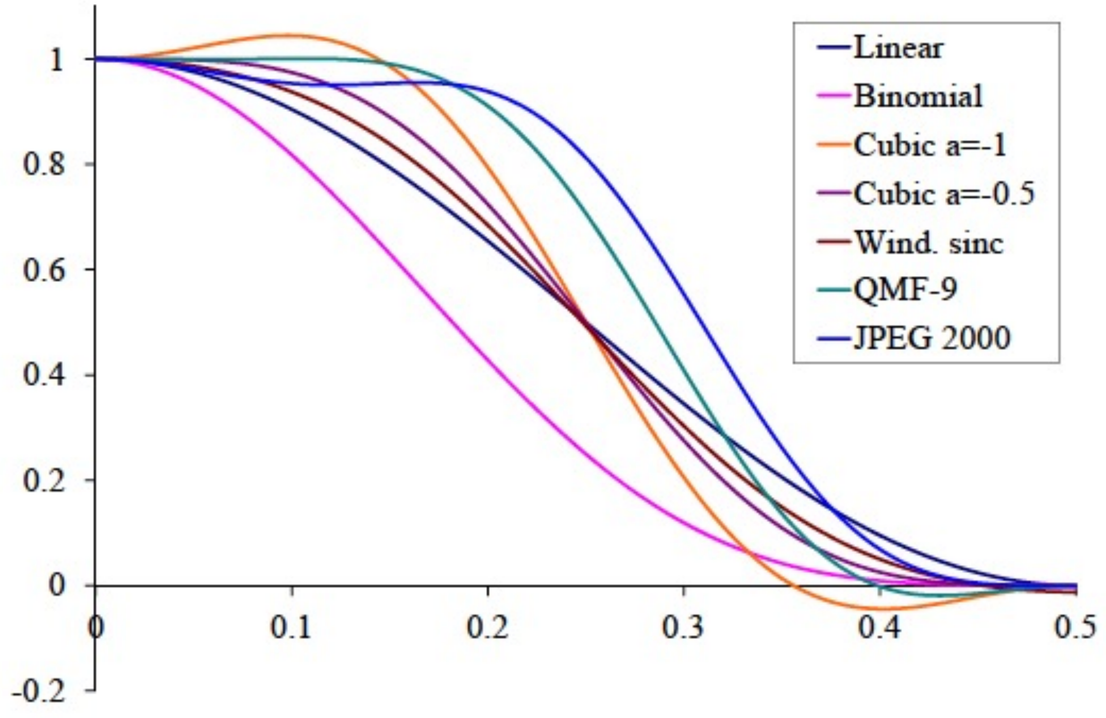
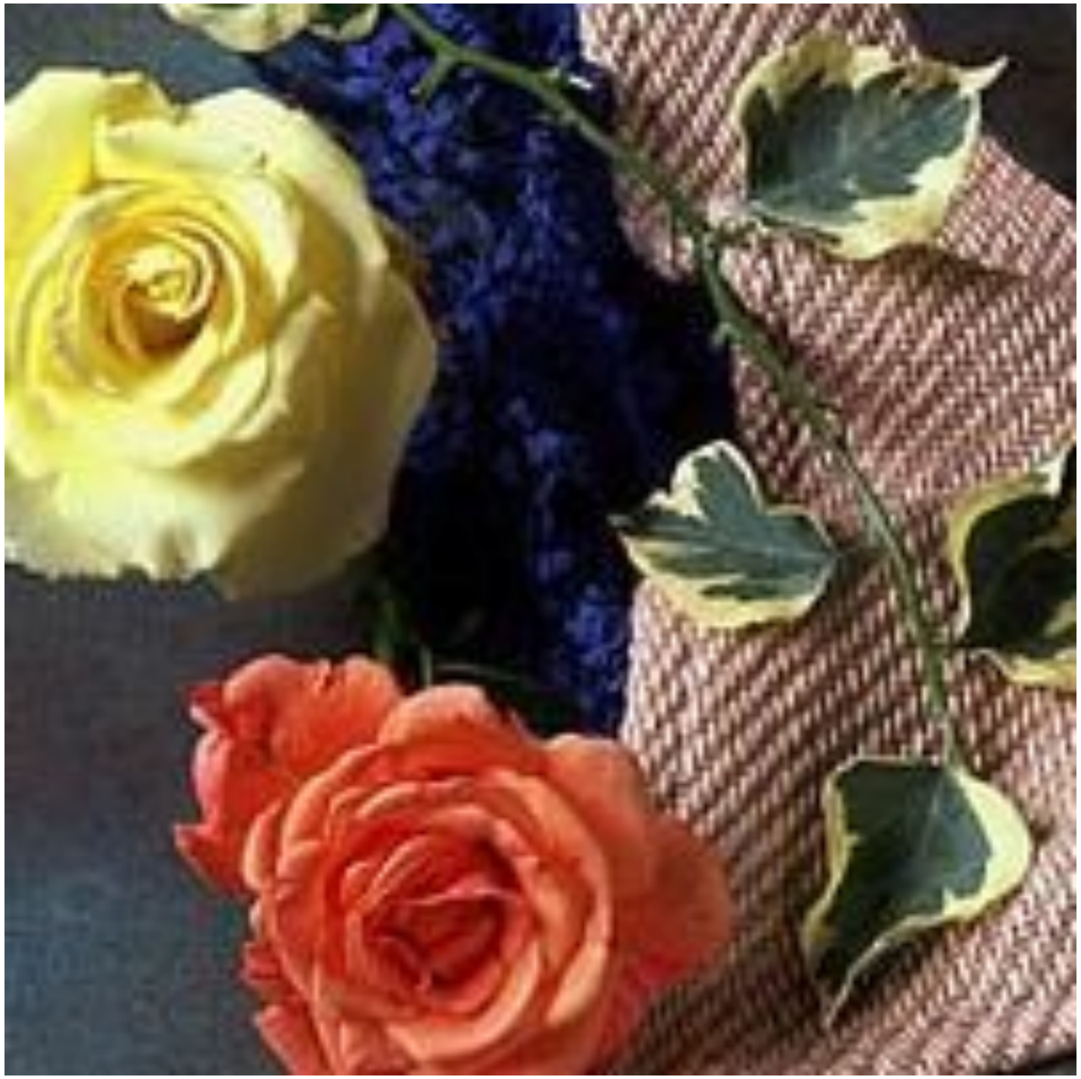
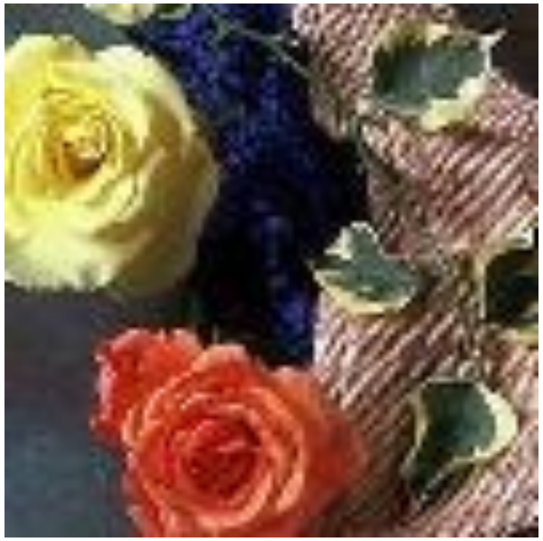


Figure 3.31 Frequency response for some 2× decimation filters. The cubic $a = -1$ filter has the sharpest fall-off but also a bit of ringing; the wavelet analysis filters (QMF-9 and JPEG 2000), while useful for compression, have more aliasing.

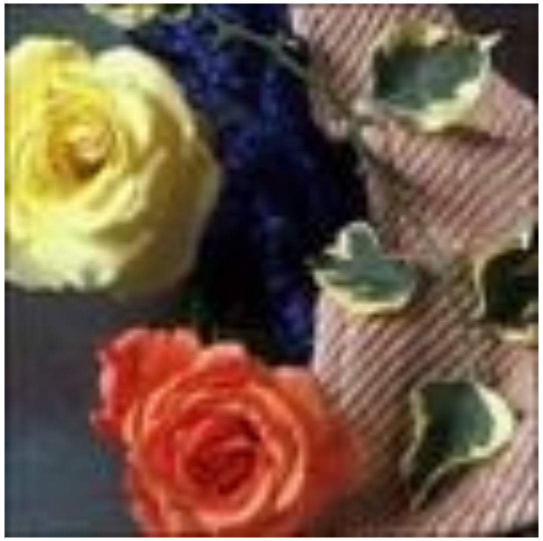
From [Szeliski2012]

Example: Image Down-Sample



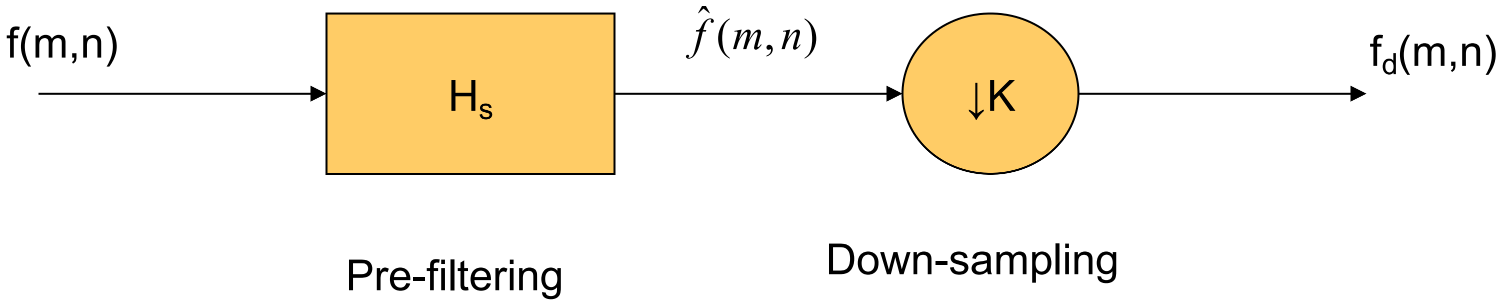


Without prefiltering



With averaging (2x2)

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., fdesign.nyquist( )

Note in MATLAB, 1.0 represents the highest digital freq of ½. So for a desired cut off of ¼, you should specify ½ as the cut-off frequency.

https://www.mathworks.com/help/dsp/examples/fir-nyquist-l-th-band-filter-design.html

Down-Sampling Using Matlab

- Without prefiltering
  - If f(m,n) is an MxN image, down-sampling by a factor of K can be done simply by

```
>> g=f(1:K:M,1:K:N)
```

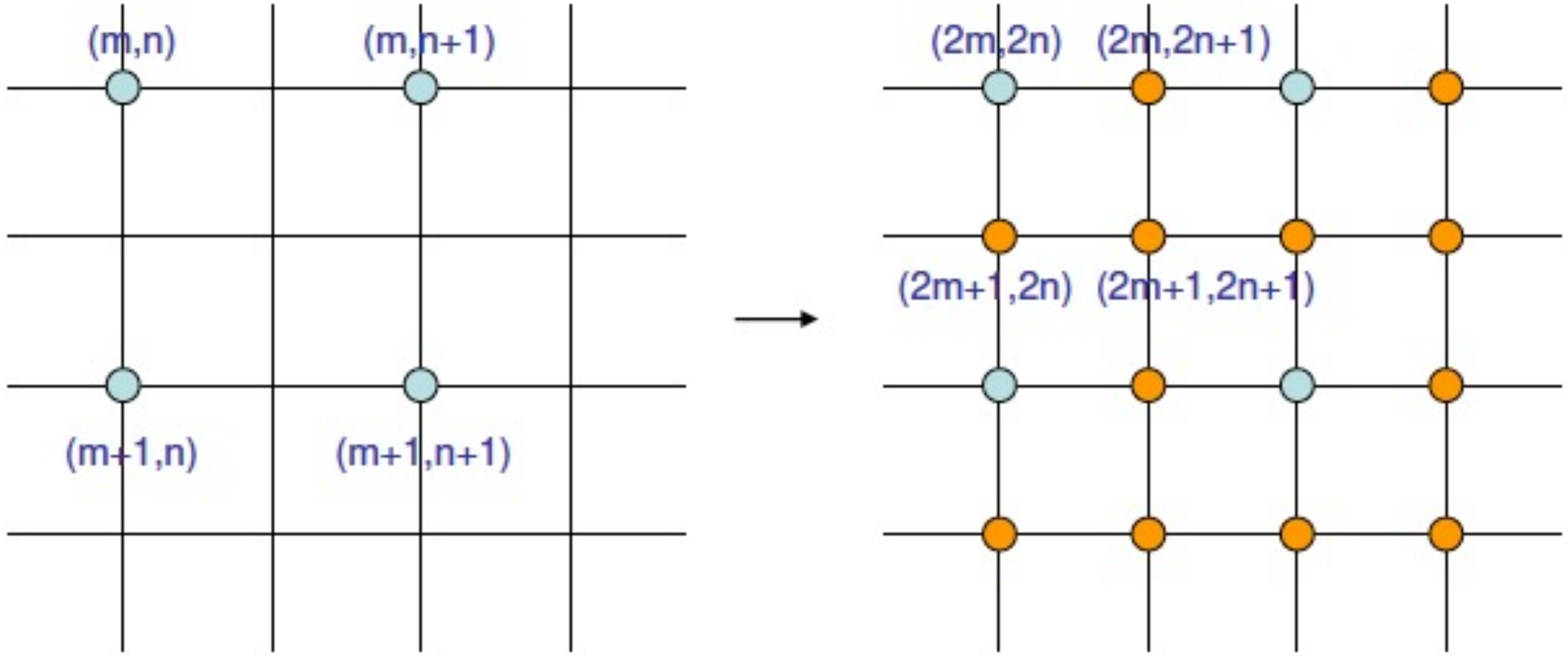
- With prefiltering
  - First convolve the image with a desired filter
    - Low pass filter with digital cutoff frequency 1/(2K)
      - In matlab, 1/2 is represented by 1, so the cutoff is 1/K
  - Then subsample

```
>> h=fir1(N, 1/K);
   %design a lowpass filter with cutoff at 1/2K and length N.
  % or directly specify a filter from the previous table
>> h2=h'*h; %form 2D filters from 1D filter
>> fp=conv2(f,h,'same')
>> g=fp(1:K:M,1:K:N)
```

Image Up-Sampling

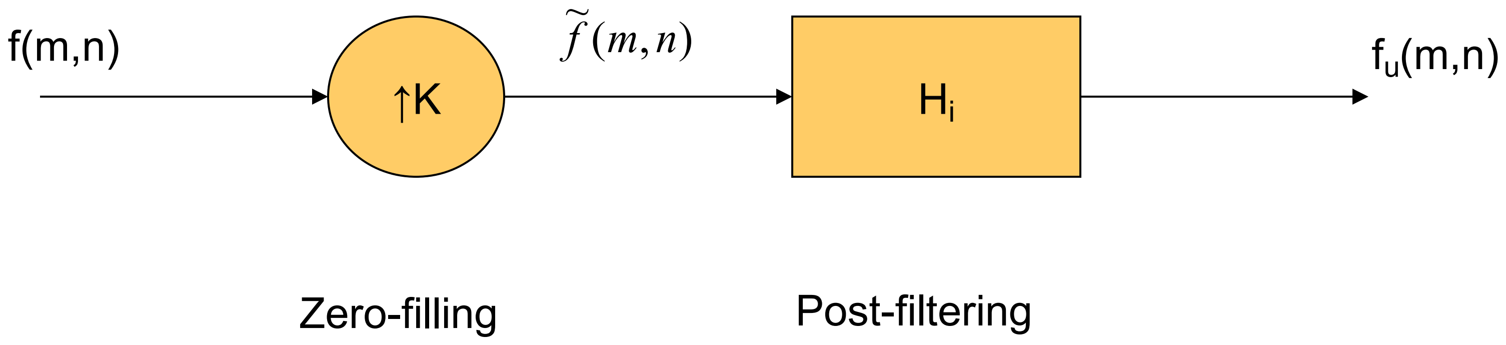
- Produce a larger image from a smaller one
  - Eg. 512x512 -> 1024x1024
  - More generally we may up-sample by an arbitrary factor L
- How should we generate a larger image?
  - First up-sample by filling in zeros
  - Then filter the zero-filled signal
  - Separable operation

Factor of 2 Up-Sampling



Green samples are retained in the interpolated image;
Orange samples are estimated from surrounding green samples, initially set to 0.

Filtering View: Up Sampling by a Factor of K



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

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

Ideally H should be a low pass filter with cutoff at 1/2K in digital frequency. Often the filter h(k,l) is separable, so that h(k,l)=g(k) g(l).

Frequency Domain Interpretation (1D)

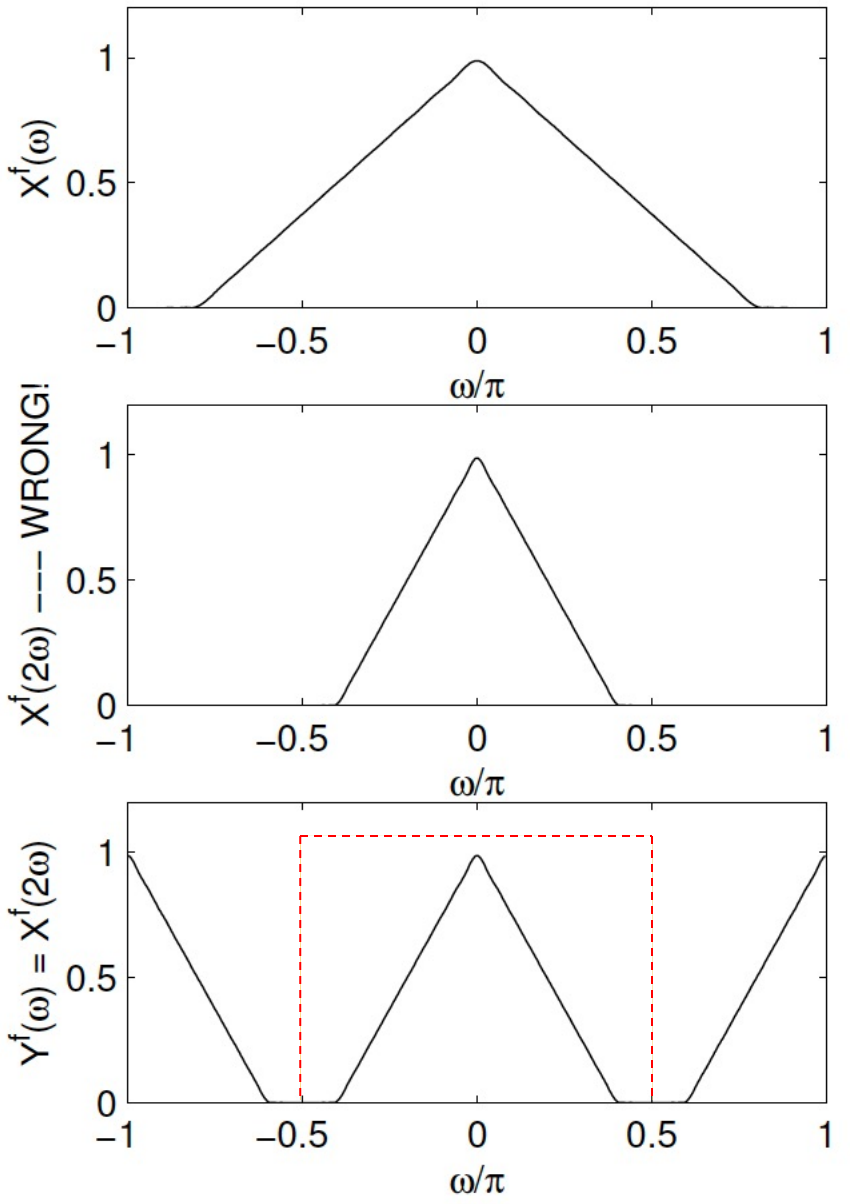
$$y(n) = [\uparrow 2]\, x(n)$$

$$Y(z) = X(z^2).$$

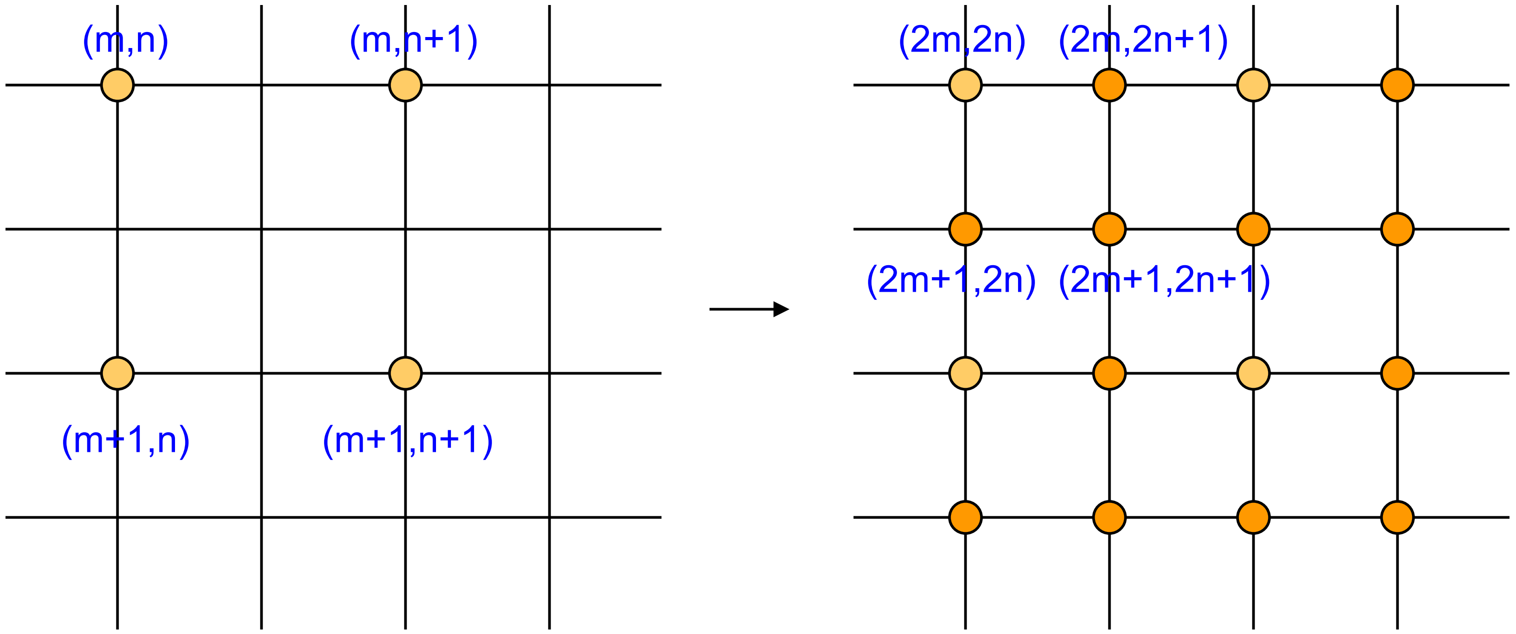
$$Y^f(\omega) = X^f(2\omega).$$

Should apply a half-band filter to zero-filled signal to remove the aliased copy!

Nyquist filter (to guarantee the original sample values do not change): h(0)=1, h(Kn)=0, symmetric



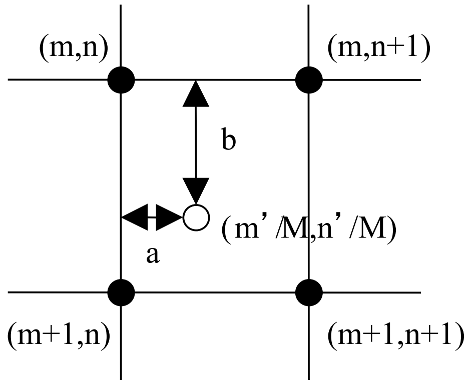
Nearest Neighbor Interpolation (Pixel Replication) (M=2)



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]

Equivalent 1D filter:
h=[1, 1]
O'(n)=I(n/2) for n=2m, =0 otherwise
O(n)=O'(n)*1+O'(n-1)*1

Nearest Neighbor Interpolation (pixel replication) (General Case)

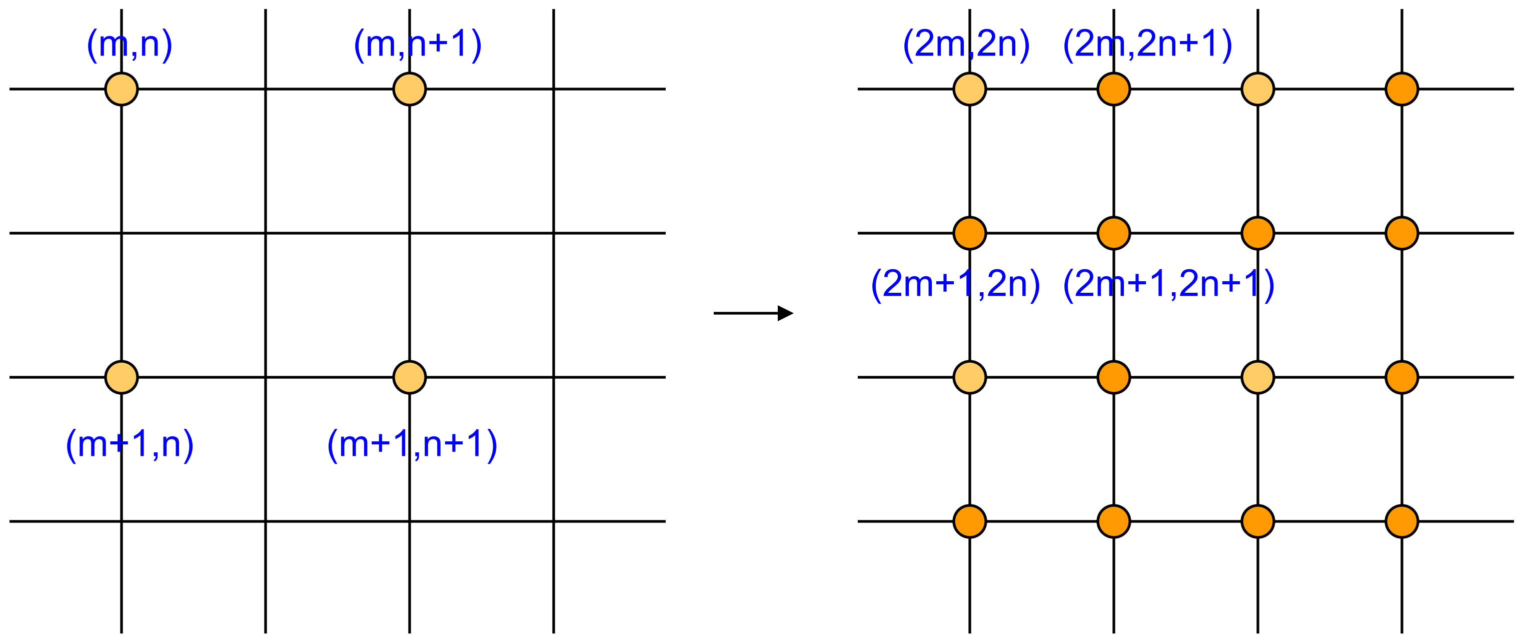


O[m',n'] (the resized image) takes the value of the sample nearest to (m'/M,n'/M) in I[m,n] (the original image):

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

Each original pixel is replaced by MxM pixels of the sample value
Equivalent to using the sample-and-hold interpolation filter.

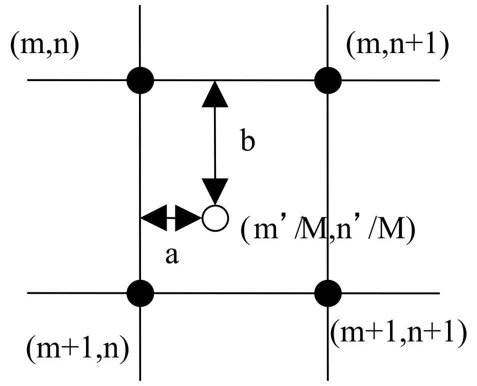
Bilinear Interpoation (M=2)



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

Equivalent 1D Filter
h=[0.5, 1, 0.5]

Bilinear Interpolation (General Case, Not Required)

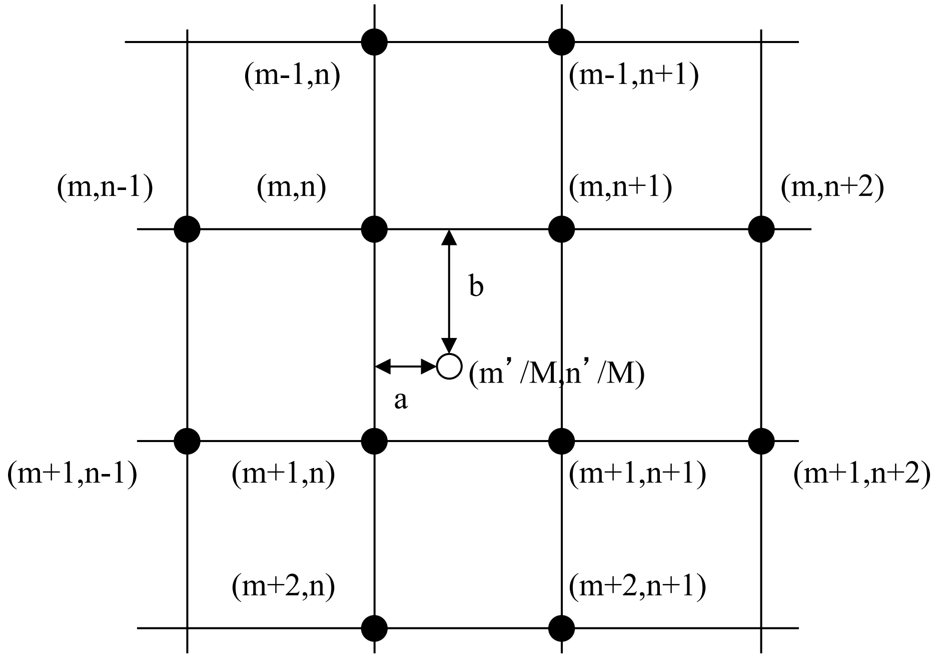


- O(m’,n’) takes a weighted average of 4 samples nearest to (m’/M,n’/M) in I(m,n).

- Separable interpolation:
  i) interpolate along each row y: F[m,n’]=(1-a)*I[m,n]+a*I[m,n+1], a=n’-n.
  Linear interpolation: assume samples at (m,n) and (m,n+1) form a straight line
  ii) interpolate along each column x’: O[m’,n’]=(1-b)*F[m’,n]+b*F[m’+1,n], b=m’-m

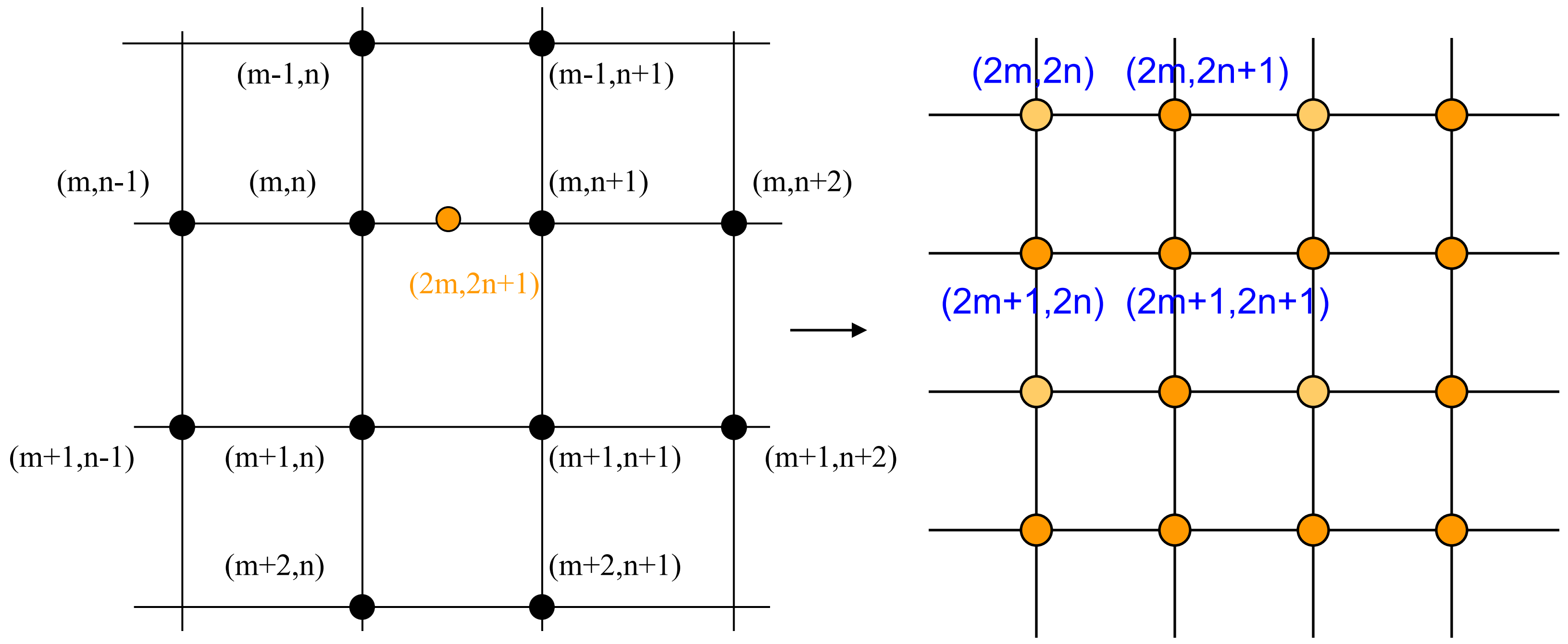
- Direct interpolation: each new sample takes 4 multiplications:
  O[m’,n’]=(1-a)*(1-b)*I[m,n]+a*(1-b)*I[m,n+1]+(1-a)*b*I[m+1,n]+a*b*I[m+1,n+1]

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]->F[m,n']
  - (from 4 samples, 2 on the left, 2 on the right)
  - ii) interpolate along each column x': F[m,n']-> O[m',n'] (from 4 samples)
- Direct interpolation: each new sample takes 16 multiplications

Special Case: M=2



Bicubic: cubic interpolation in both horizontal and vertical 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)

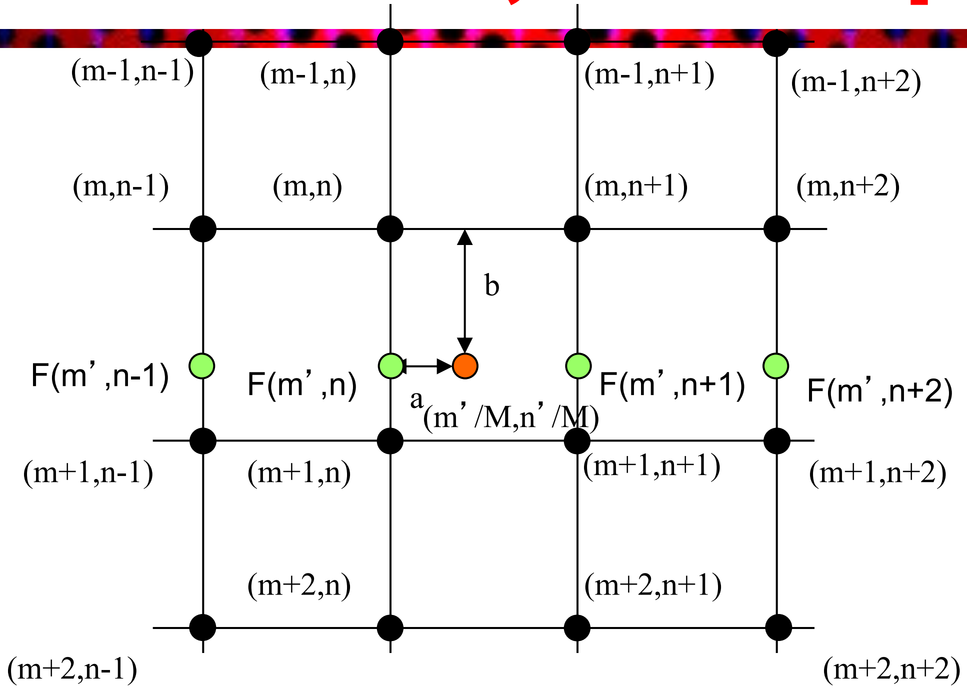
Equivalent 1D filter: h=[-1/8, 0, 5/8,1, 5/8, 0, -1/8]

(satisfying Nyquist filter condition!)

Same operation then repeats in vertical direction

Coefficients determined so that resulting signal can be modeled by a third order polynomial in each dimension.

Interpolation Formula (General Case, Not Required)



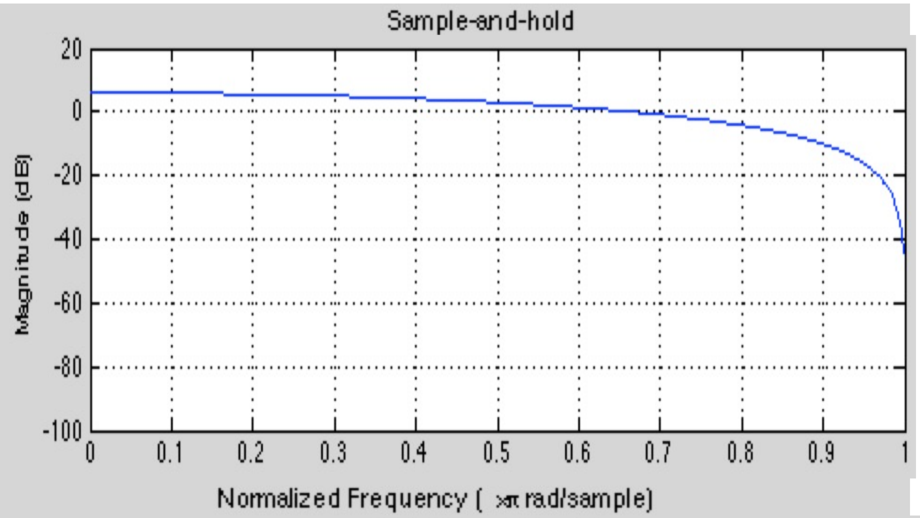
$$F[m',n] = -b(1-b)^2 I[m-1,n] + (1-2b^2+b^3) I[m,n] + b(1+b-b^2) I[m+1,n] - b^2(1-b) I[m+2,n],$$

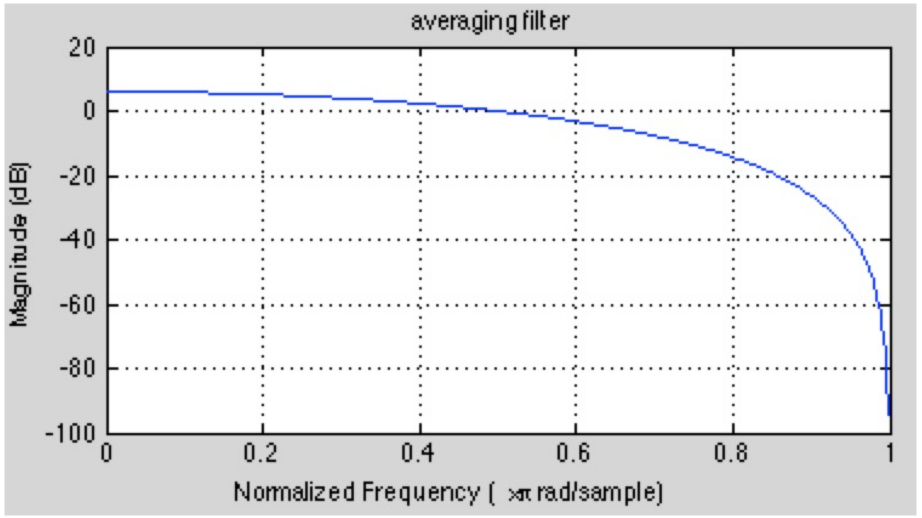
$$\text{where} \quad m = (\text{int})\frac{m'}{M}, b = \frac{m'}{M} - m$$

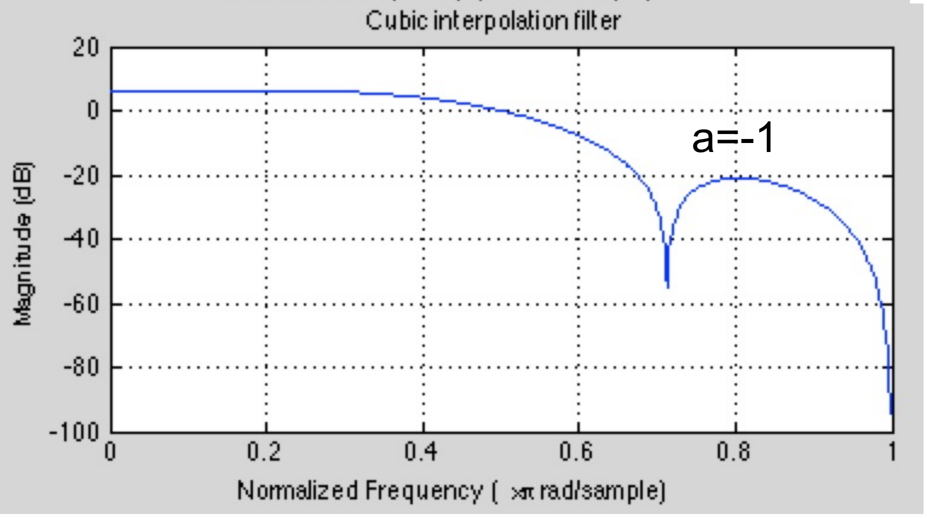
$$O[m',n'] = -a(1-a)^2 F[m',n-1] + (1-2a^2+a^3) F[m',n] + a(1+a-a^2) F[m',n+1] - a^2(1-a) F[m',n+2],$$

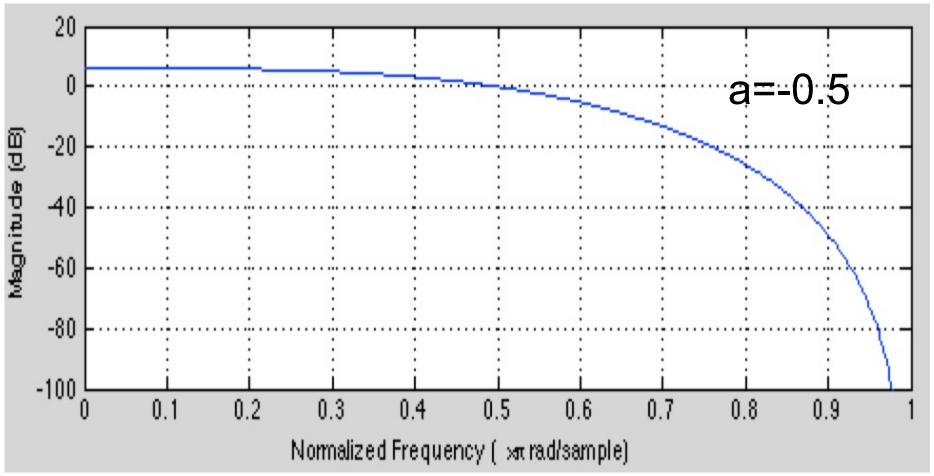
$$\text{where} \quad n = (\text{int})\frac{n'}{M}, a = \frac{n'}{M} - n$$

Freq. Response of Some Interpolation Filters









In MATLAB: “1” represents ½ in

Interpretation in Continuous Domain (Optional)

- Original signal sampled at fs, with interval Δ
- Reconstruct continuous space signal from these samples using interpolation filter h(x)

$$\hat{f}(x) = \sum_m f_s(m) h(x - m\Delta)$$

- Upsample by factor of K is equivalent to sample the interpolated signal with sampling rate of Kfs, or at interval Δ/K

$$\hat{f}\left(l\frac{\Delta}{K}\right) = \sum_m f_s(m) h\left(l\frac{\Delta}{K} - m\Delta\right)$$

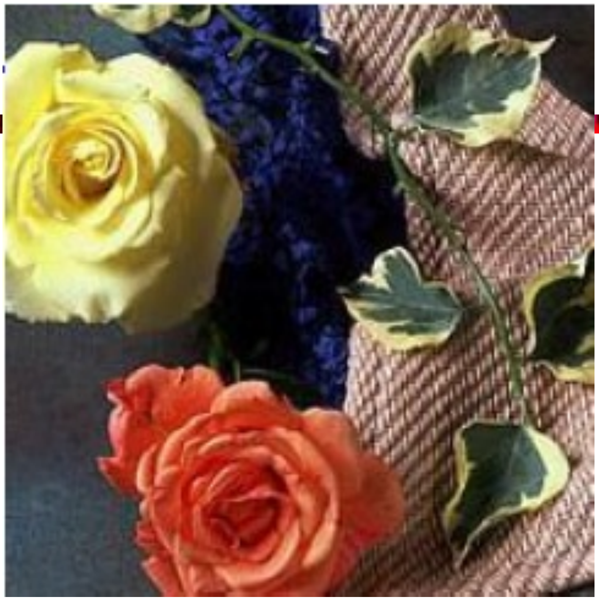
- Let $\frac{\Delta}{K}$=1, the upsampled signals can be written as

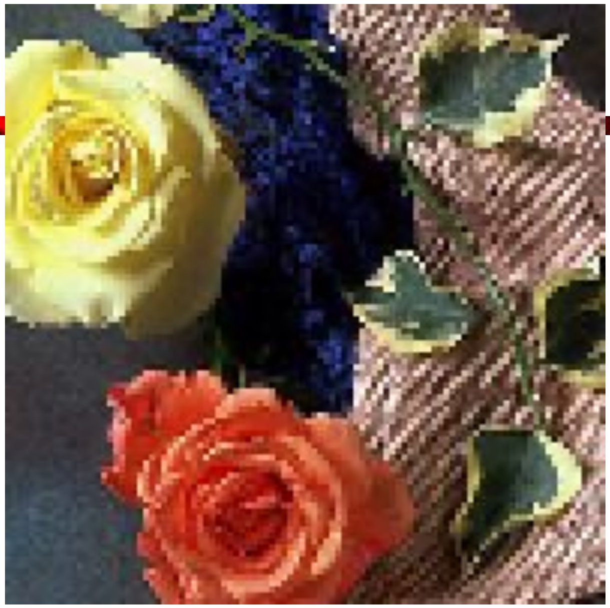
$$\hat{f}(l) = \sum_m f_s(m) h'(l - mK) = \sum_n \tilde{f}(n) h'(l - n) = \tilde{f}(l) * \mathrm{h}'(\mathrm{l})$$

$\tilde{f}(n)$ being the zero-padded signal; h'(n) is the rescaled interpolation kernel with Δ=K

Up-Sampled from w/o Prefiltering

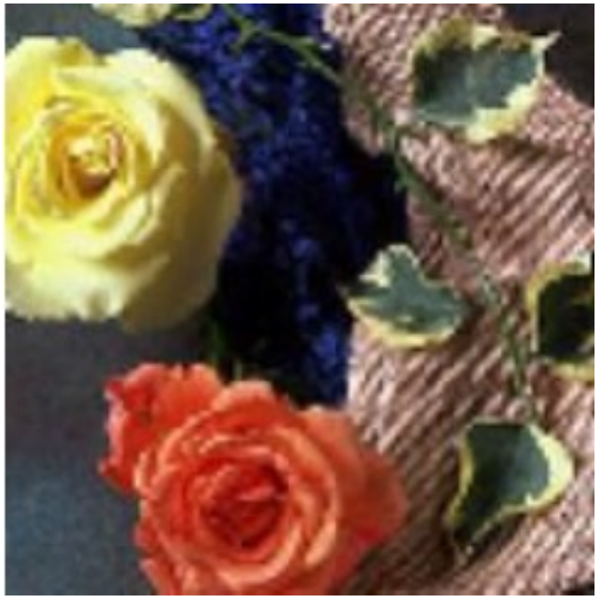
Original

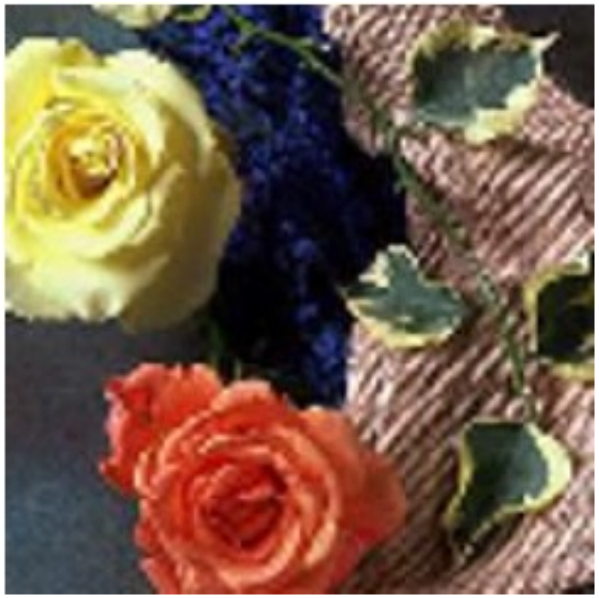




Nearest neighbor

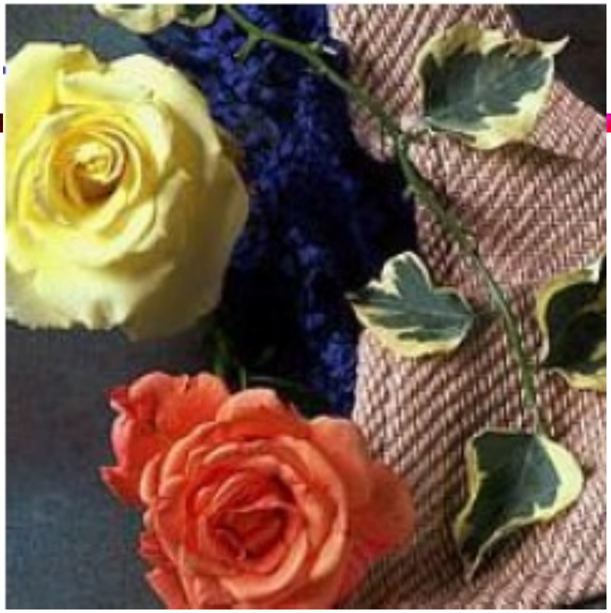
Bilinear

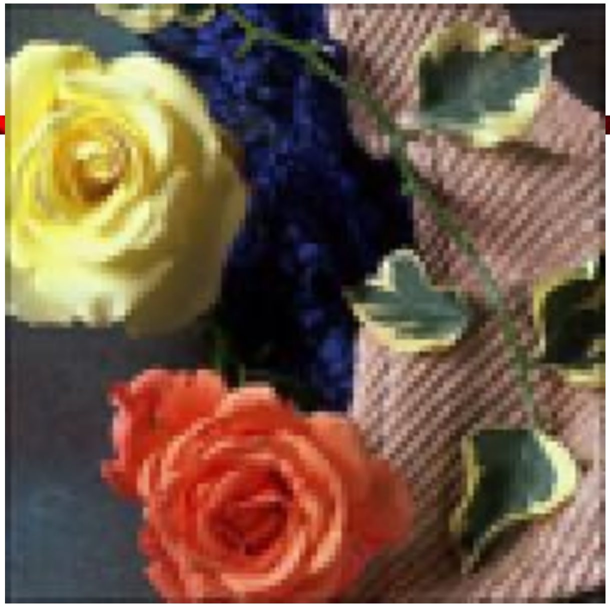


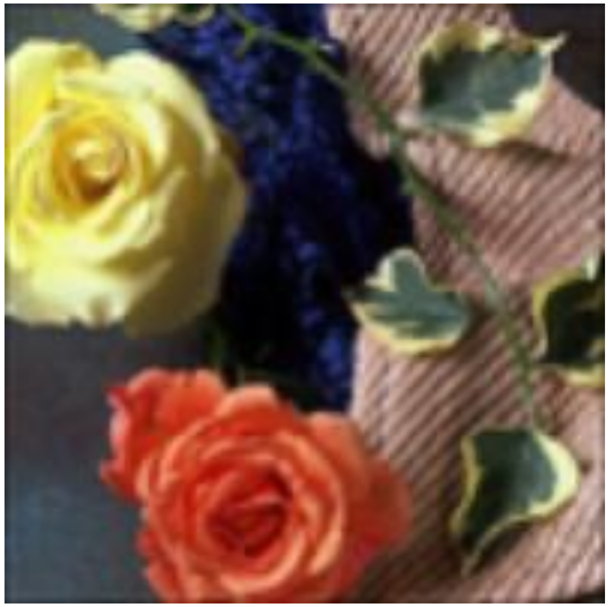


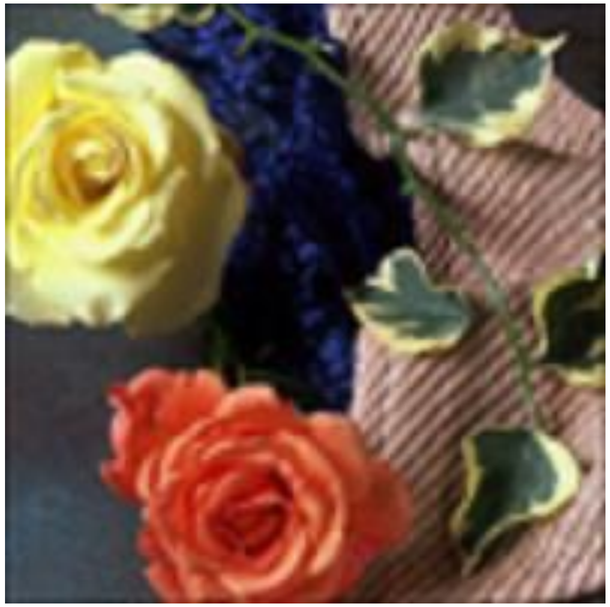
Bicubic

Up-Sampled from with Prefiltering


Original


Nearest neighbor

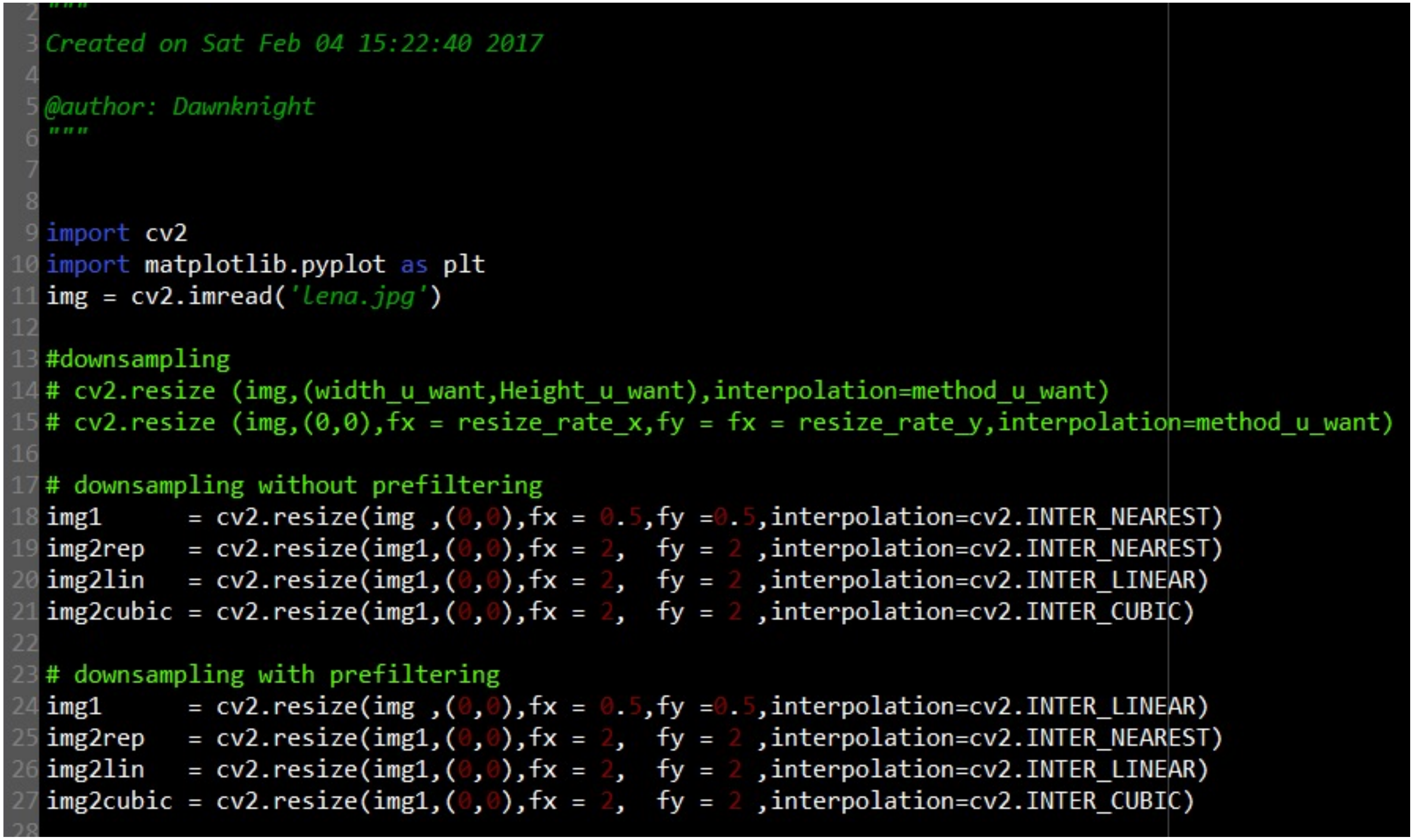

Bilinear


Bicubic

Matlab for Image Resizing

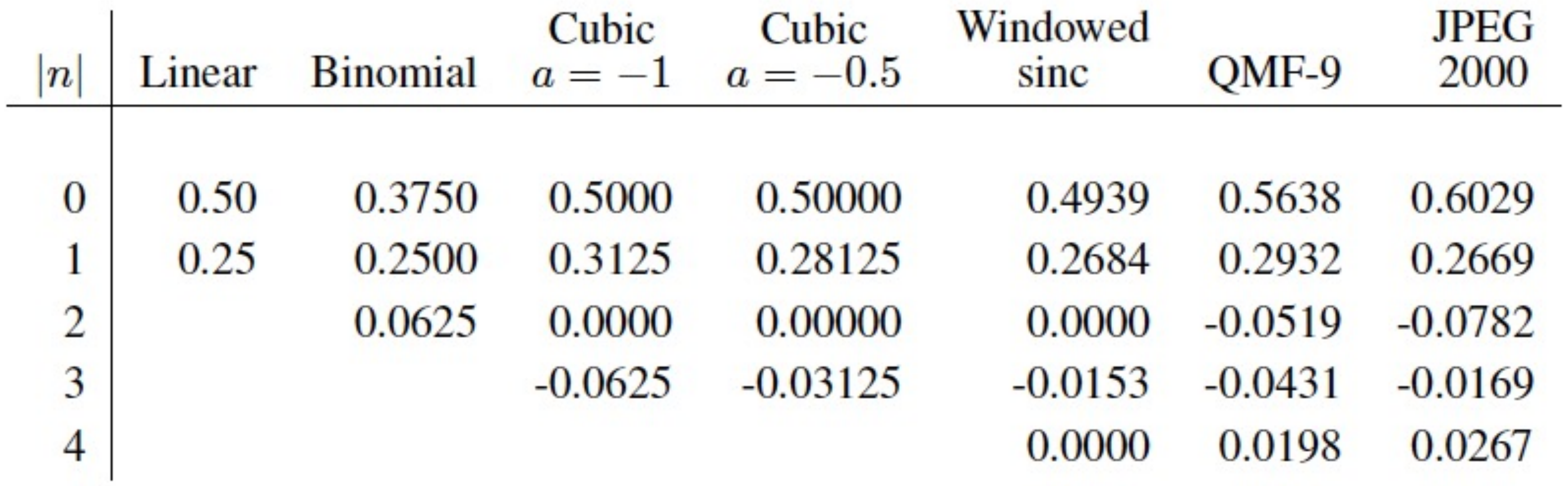
```
[img]=imread('fruit.jpg','jpg');
%downsampling without prefiltering
img1=imresize(img,0.5,'nearest');
%upsampling with different filters:
img2rep=imresize(img1,2,'nearest');
img2lin=imresize(img1,2,'bilinear');
img2cubic=imresize(img1,2,'bicubic');

%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');
```

Python for Image Resizing



```
"""
Created on Sat Feb 04 15:22:40 2017

@author: Dawnknight
"""


import cv2
import matplotlib.pyplot as plt
img = cv2.imread('Lena.jpg')

#downsampling
# cv2.resize (img,(width_u_want,Height_u_want),interpolation=method_u_want)
# cv2.resize (img,(0,0),fx = resize_rate_x,fy = fx = resize_rate_y,interpolation=method_u_want)

# downsampling without prefiltering
img1      = cv2.resize(img ,(0,0),fx = 0.5,fy =0.5,interpolation=cv2.INTER_NEAREST)
img2rep   = cv2.resize(img1,(0,0),fx = 2,  fy = 2 ,interpolation=cv2.INTER_NEAREST)
img2lin   = cv2.resize(img1,(0,0),fx = 2,  fy = 2 ,interpolation=cv2.INTER_LINEAR)
img2cubic = cv2.resize(img1,(0,0),fx = 2,  fy = 2 ,interpolation=cv2.INTER_CUBIC)

# downsampling with prefiltering
img1      = cv2.resize(img ,(0,0),fx = 0.5,fy =0.5,interpolation=cv2.INTER_LINEAR)
img2rep   = cv2.resize(img1,(0,0),fx = 2,  fy = 2 ,interpolation=cv2.INTER_NEAREST)
img2lin   = cv2.resize(img1,(0,0),fx = 2,  fy = 2 ,interpolation=cv2.INTER_LINEAR)
img2cubic = cv2.resize(img1,(0,0),fx = 2,  fy = 2 ,interpolation=cv2.INTER_CUBIC)
```

Other filters for interpolation by 2



$\lvert n \rvert$	Linear	Binomial	Cubic $a=-1$	Cubic $a=-0.5$	Windowed sinc	QMF-9	JPEG 2000
0	0.50	0.3750	0.5000	0.50000	0.4939	0.5638	0.6029
1	0.25	0.2500	0.3125	0.28125	0.2684	0.2932	0.2669
2		0.0625	0.0000	0.00000	0.0000	-0.0519	-0.0782
3			-0.0625	-0.03125	-0.0153	-0.0431	-0.0169
4					0.0000	0.0198	0.0267

Table 3.4 Filter coefficients for $2\times$ decimation. These filters are of odd length, are symmetric, and are normalized to have unit DC gain (sum up to 1). See Figure 3.31 for their associated frequency responses.

From [Szeliski2012]

The above filters, when multiplied by 2 in coefficient values, are commonly used interpolation filters for 2x interpolation, to be used on zero-filled up-sampled signals. Note that Binomial, QMF-9, and JPEG are not Nyquist Filters.

Summary

- Nyquist sampling theorem (continuous->discrete->continuous)
  - Direct extension from 1D to 2D to 3D to even higher dimension
  - Prefiltering to prevent aliasing
  - Interpolation to remove aliased component
  - Ideal filter for both, practical filters
  - Separable filtering along each direction
- Image resizing (discrete->discrete)
  - Think of reconstructing continuous image and resampling
  - Or Think of directly down-sampling or interpolating
  - Practical filters for down-sampling: averaging, weighted average, binomial
  - Practical filters for up-sampling: bilinear and cubic spline
  - Tradeoff between blurring and aliasing

Summary (Cnt'd)

- Properties of human visual system
  - Concept of angular frequency
  - Methods to measure visual spatiotemporal thresholds
  - Spatial masking and temporal masking, smooth pursuit
  - Design of practical cameras and displays based on visual thresholds and masking effect
  - Reduced sensitivity to chrominance components
    - Chrominance subsampling formats

Reading Assignment

- [Szeliski2012] Richard Szeliski, Computer Vision: Algorithms and Applications. 2012. Sec. 3.5.1, 3.5.2.
- Lecture Note for this class.
- Optional:
- [Wang2002] Wang et al, Digital video processing and communications. Sec. 2.3, 2.4: Frequency response of visual systems; Chap 3 and 4: sampling and sampling rate conversion. You can focus on rectangular sampling only.
- Ivan Selesnick, Lecture note on Multirate Systems.

Written Homework (1)

1. Consider a function $f(x, y) = \cos 2\pi(2x + 4y)$ 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 $f_{cx} = f_{cy} = 1/2 f_s$. 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:

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

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

2. You are asked to determine the necessary sampling rate of a digital video display based on the knowledge of the human visual thresholds in terms of the maximum temporal and spatial frequency that the human can perceive properly. Suppose that the visual temporal threshold is 60 Hz, and the visual spatial threshold is 30 cycles/degree (cpd). The display size is 3 feet (width) by 2 feet (height). Assume that the viewer typically sits 4 feet from the display. What are the minimal frame rate (frames/sec), line rate (lines/frame) and pixels/line that you should use?

3. (Optional) Suppose you want to increase the image size by a factor of 3. Describe the steps you would use to accomplish this. What is the desired filter response for the interpolation filter? What would be the equivalent filter for linear interpolation?

Computer Exercises (Optional)

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 given in Table 3.4 in [Szeliski2012]; b) Up-sample the previously down-sampled images by a factor of 2, using the bilinear interpolation and cubic interpolation methods (a=-1), 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 ”cv2.resize” function in Python to do this assignment. But you are encouraged to compare results of your program with ”cv2.resize”. For saving image you can use cv2.imwrite(’filename.file_extension’, img),

Hint :common image file extension : jpg, png, bmp… etc