Computer Vision

CS308

Feng Zheng
SUSTech CS Vision Intelligence and Perception

Week 4

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

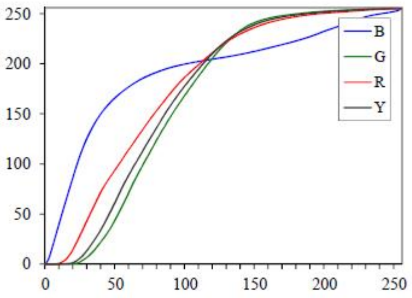
Content

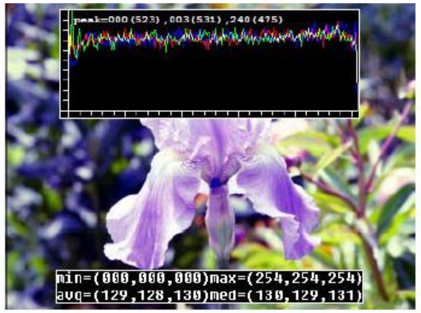
- Brief Review
- Thinking in Frequency
- Pyramids
- Geometric Transformations

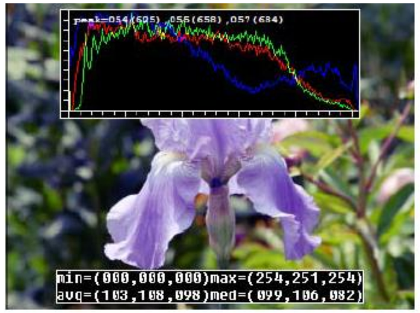
Brief Review

Review

- Point Operators







- Linear Filtering
  - ➢ Linear filtering is dot product at each position
    - ✓ Not a matrix multiplication
    - ✓ Can smooth, sharpen, translate (among many other uses)
  - ➢ Be aware of details
    - ✓ Filter size
    - ✓ Stride
    - ✓ Padding
    - ✓ Values to be fixed or **learned**

Auto ML

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$f(x,y)$

$*$

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$h(x,y)$

$=$

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

$g(x,y)$

Review

- More Neighborhood Operators

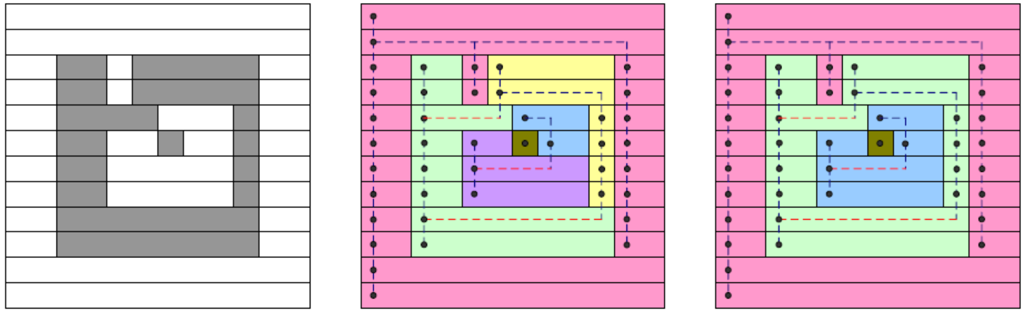
(a) (b) (c) (d) (e) (f)

1	2	1	2	4
2	1	3	5	8
1	3	7	6	9
3	4	8	6	7
4	5	7	8	9

(a) median = 4

1	2	1	2	4
2	1	3	5	8
1	3	7	6	9
3	4	8	6	7
4	5	7	8	9

(b) α-mean= 4.6



0	0	0	0	1	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	0	0

0	0	0	0	1	0	0
0	0	1	1	2	0	0
0	1	2	2	3	1	0
0	1	2	3			

0	0	0	0	1	0	0
0	0	1	1	2	0	0
0	1	2	2	3	1	0
0	1	2	2	1	1	0
0	1	2	1	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	0	0

0	0	0	0	1	0	0
0	0	1	1	1	0	0
0	1	2	2	2	1	0
0	1	2	2	1	1	0
0	1	2	1	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	0	0

Review

- Fill in the blanks

Filtering Operator

a) (G) = D * B

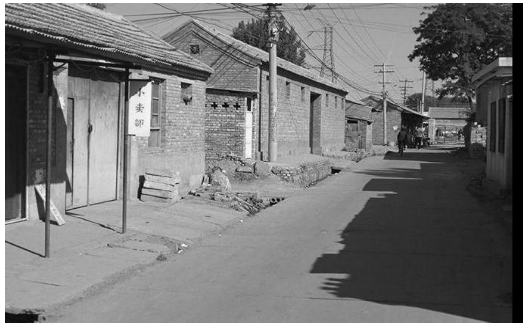
b) A = (C) * (B)

c) F = D * (E)

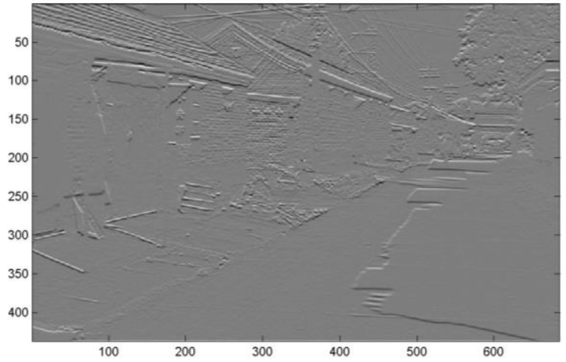
d) (I) = D * D

E

F

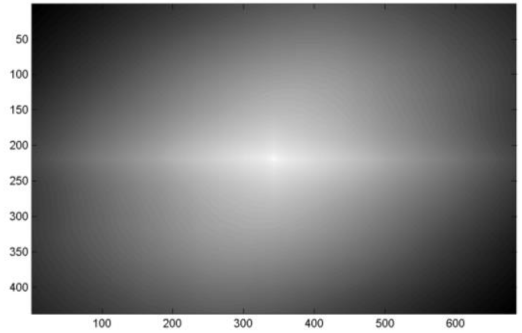


G

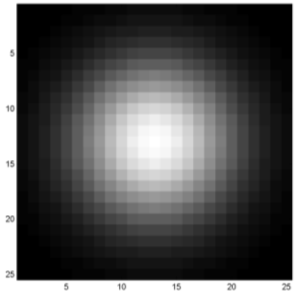


H

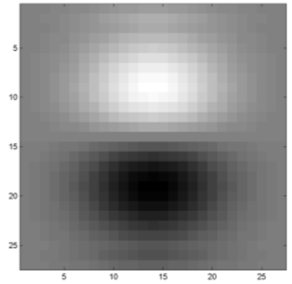
I



C

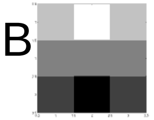


A



D

B

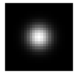


Slide: Hoiem

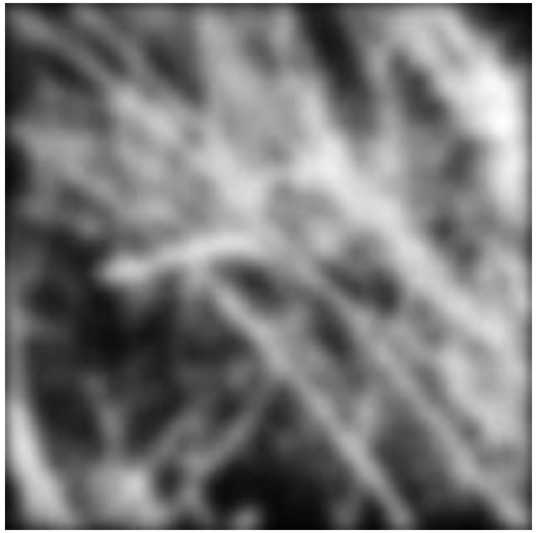
Thinking in Frequency

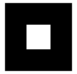
Thinking

- Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

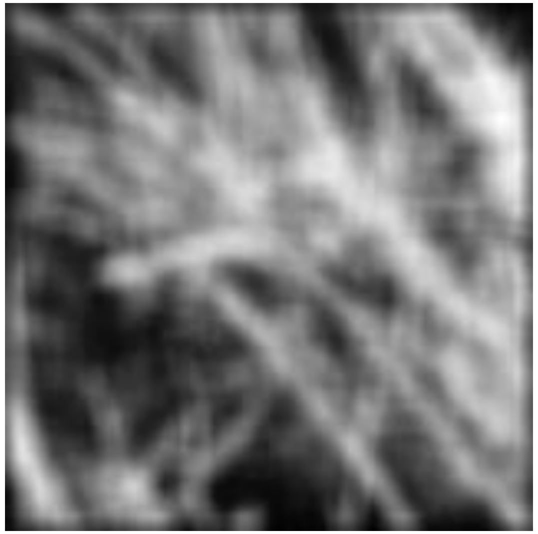


Gaussian





Box filter



- How is it that a 4MP image can be compressed to a few hundred KB without a noticeable change?

Thinking

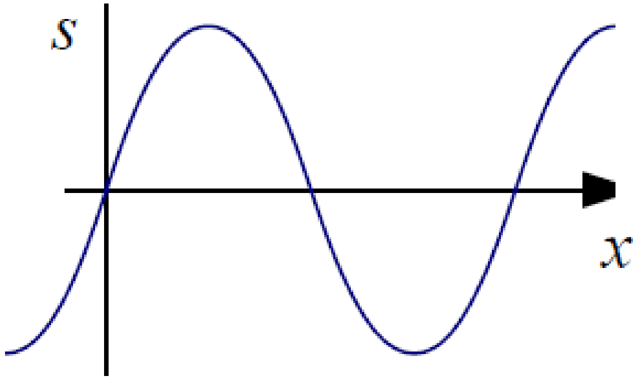
- Why does a lower resolution image still make sense to us?
- What do we lose?

Filtering and Fourier

- How can we analyze what a given filter does to high, medium, and low frequencies?
  - ➢ Pass a sinusoid of known frequency through the filter and to observe by how much it is attenuated

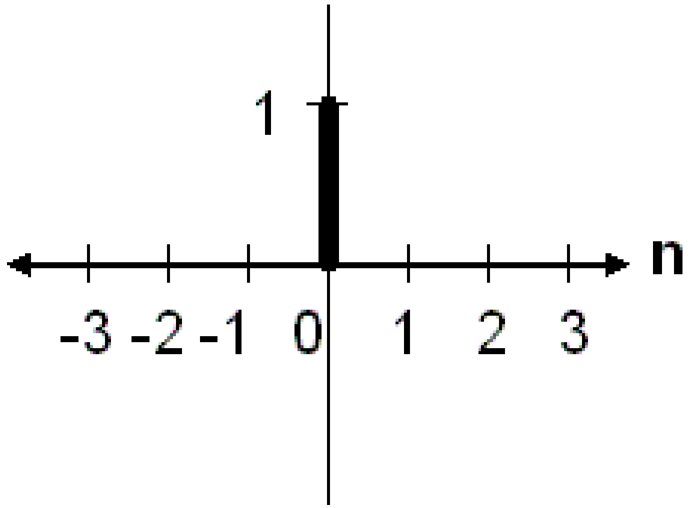
$$s(x) = \sin(2\pi f x + \phi_i) = \sin(\omega x + \phi_i)$$

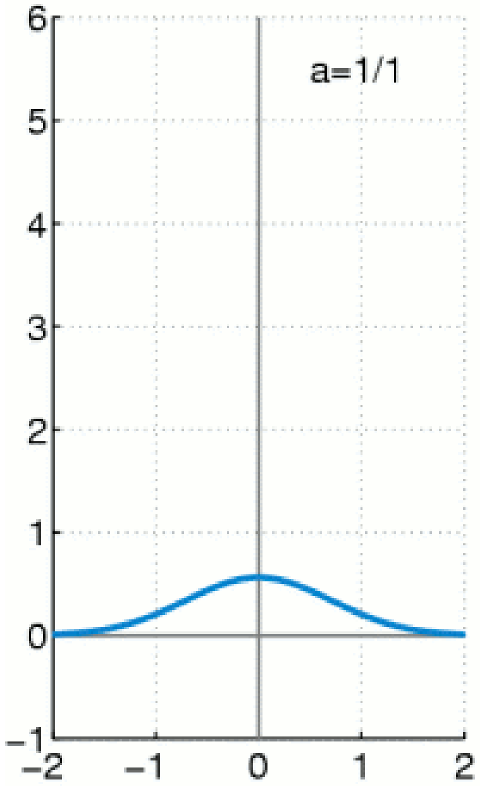
frequency angular frequency phase



Filtering and Fourier

- Impulse response: h(x)
  - Continuous function
  - Discrete function





  - The impulse function contains all frequencies
- If we convolve the signal with a filter whose impulse response is h(x), we get another sinusoid of the same frequency but different magnitude and phase

$$o(x) = h(x) * s(x) = A\sin(\omega x + \phi_o)$$

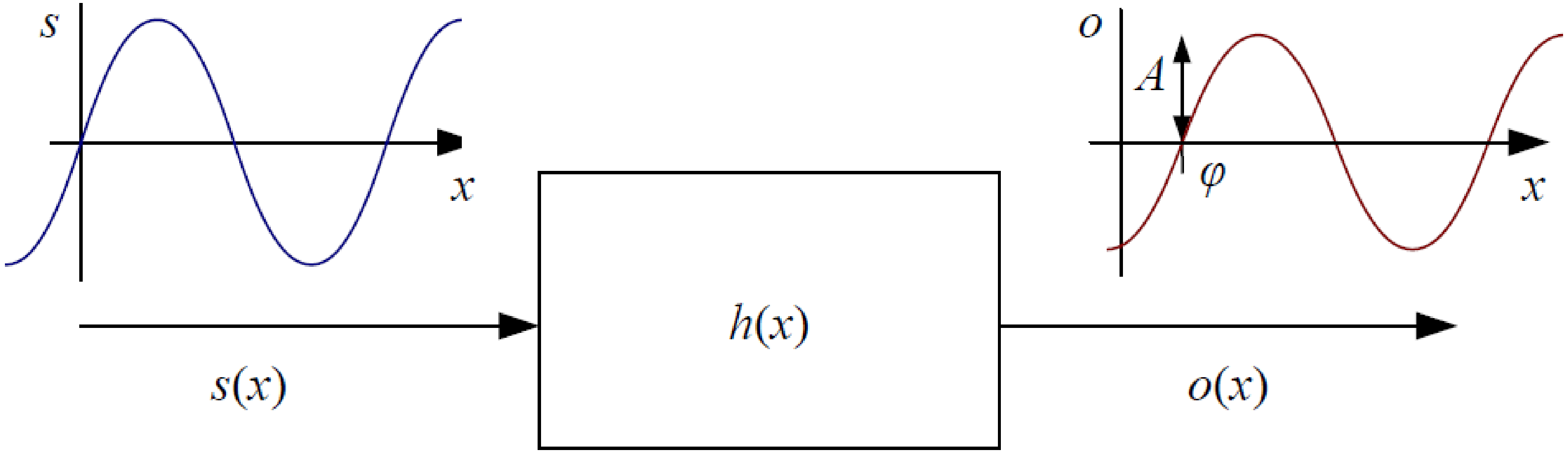
Filtering and Fourier

- Convolution

$$g(\boldsymbol{x}) = \int f(\boldsymbol{x} - \boldsymbol{u})h(\boldsymbol{u})d\boldsymbol{u}.$$

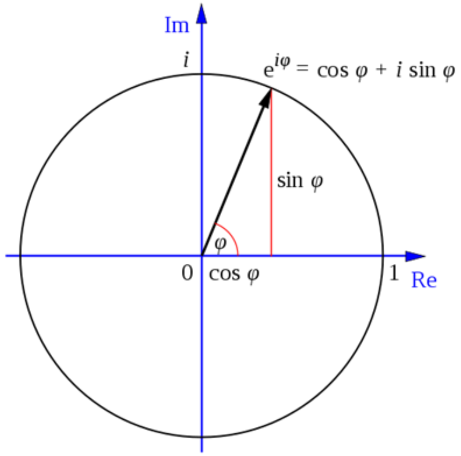
  - A weighted summation of shifted input signals (sinusoids)
  - The summation of a bunch of shifted sinusoids of the same frequency is just a single sinusoid at that frequency (same)

$$o(x) = h(x) * s(x) = A\sin(\omega x + \phi_o)$$



Filtering and Fourier

$$e^{ix} = cosx + isinx$$



- The complex-valued sinusoid is

$$s(x) = \underline{e^{j\omega x} = \cos\omega x + j\sin\omega x}$$

- The filtered sinusoid is

$$o(x) = h(x) * s(x) = Ae^{j\omega x + \phi}$$

- The Fourier transform in continuous domain

One frequency

$$H(\omega) = \int_{-\infty}^{\infty} h(x)e^{-j\omega x}dx$$

- The transform in discrete domain

length of the signal

$$H(k) = \frac{1}{N}\sum_{x=0}^{N-1} h(x)e^{-j\frac{2\pi kx}{N}}$$

Filtering and Fourier

- Some useful properties of Fourier transforms

Property	Signal	Transform
superposition	$f_1(x) + f_2(x)$	$F_1(\omega) + F_2(\omega)$
shift	$f(x - x_0)$	$F(\omega)e^{-j\omega x_0}$
reversal	$f(-x)$	$F^*(\omega)$
convolution	$f(x) * h(x)$	$F(\omega)H(\omega)$
correlation	$f(x) \otimes h(x)$	$F(\omega)H^*(\omega)$
multiplication	$f(x)h(x)$	$F(\omega) * H(\omega)$
differentiation	$f'(x)$	$j\omega F(\omega)$
domain scaling	$f(ax)$	$1/aF(\omega/a)$
real images	$f(x) = f^*(x) \quad \Leftrightarrow$	$F(\omega) = F(-\omega)$
Parseval's Theorem	$\sum_x [f(x)]^2 \quad =$	$\sum_\omega [F(\omega)]^2$

Filtering and Fourier

- Occurring filters and signals

Name	Signal		Transform
impulse	$\delta(x)$	$\Leftrightarrow$	1
shifted impulse	$\delta(x-u)$	$\Leftrightarrow$	$e^{-j\omega u}$
box filter	$\mathrm{box}(x/a)$	$\Leftrightarrow$	$a\mathrm{sinc}(a\omega)$
tent	$\mathrm{tent}(x/a)$	$\Leftrightarrow$	$a\mathrm{sinc}^2(a\omega)$
Gaussian	$G(x;\sigma)$	$\Leftrightarrow$	$\frac{\sqrt{2\pi}}{\sigma}G(\omega;\sigma^{-1})$
Laplacian of Gaussian	$(\frac{x^2}{\sigma^4}-\frac{1}{\sigma^2})G(x;\sigma)$	$\Leftrightarrow$	$-\frac{\sqrt{2\pi}}{\sigma}\omega^2 G(\omega;\sigma^{-1})$
Gabor	$\cos(\omega_0 x)G(x;\sigma)$	$\Leftrightarrow$	$\frac{\sqrt{2\pi}}{\sigma}G(\omega\pm\omega_0;\sigma^{-1})$
unsharp mask	$(1+\gamma)\delta(x)$ $-\gamma G(x;\sigma)$	$\Leftrightarrow$	$(1+\gamma)-$ $\frac{\sqrt{2\pi}\gamma}{\sigma}G(\omega;\sigma^{-1})$
windowed sinc	$\mathrm{rcos}(x/(aW))$ $\mathrm{sinc}(x/a)$	$\Leftrightarrow$	(see Figure 3.29)

Separable kernels

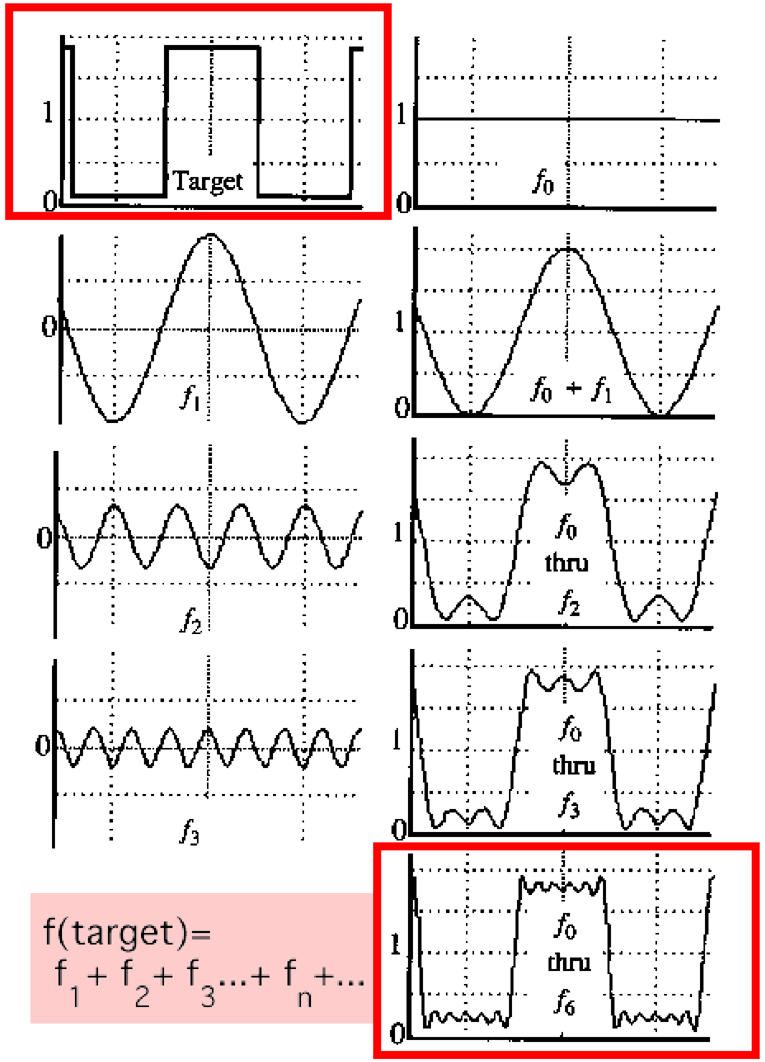
Name	Kernel	Transform	Plot
box-3	$\frac{1}{3}$ [1 | 1 | 1]	$\frac{1}{3}(1+2\cos\omega)$	
box-5	$\frac{1}{5}$ [1 | 1 | 1 | 1 | 1]	$\frac{1}{5}(1+2\cos\omega+2\cos 2\omega)$	
linear	$\frac{1}{4}$ [1 | 2 | 1]	$\frac{1}{2}(1+\cos\omega)$	
binomial	$\frac{1}{16}$ [1 | 4 | 6 | 4 | 1]	$\frac{1}{4}(1+\cos\omega)^2$	
Sobel	$\frac{1}{2}$ [−1 | 0 | 1]	$\sin\omega$	
corner	$\frac{1}{2}$ [−1 | 2 | −1]	$\frac{1}{2}(1-\cos\omega)$	

Aexample: A Sum of Sines

- Our building blocks:

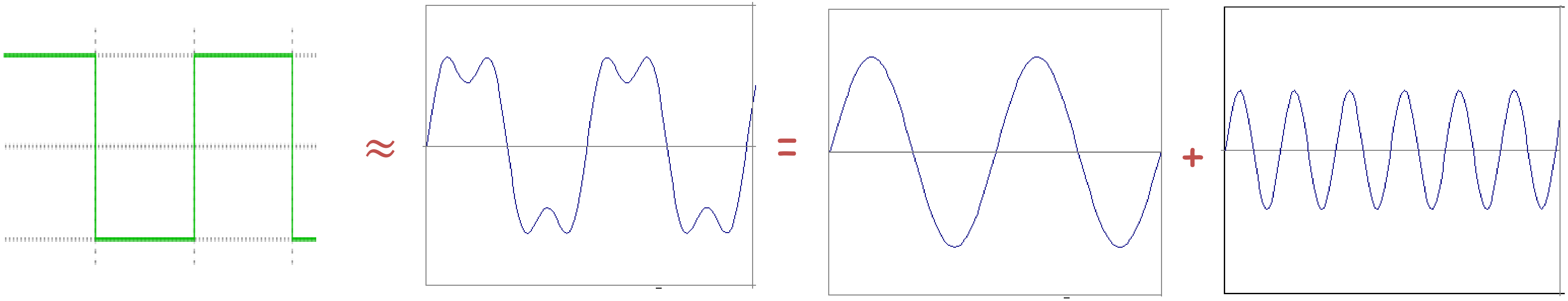
$$A\sin(\omega x + \phi)$$

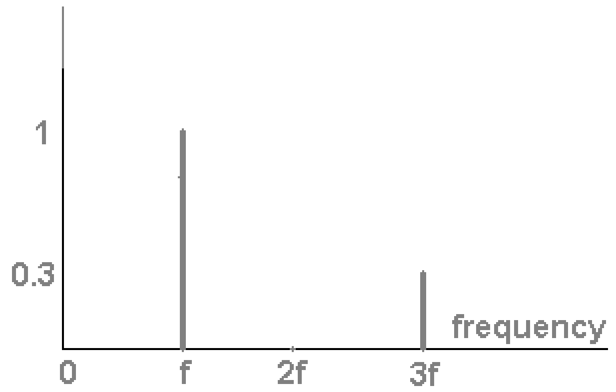
➢ Add enough of them to get any signal g(x) you want!



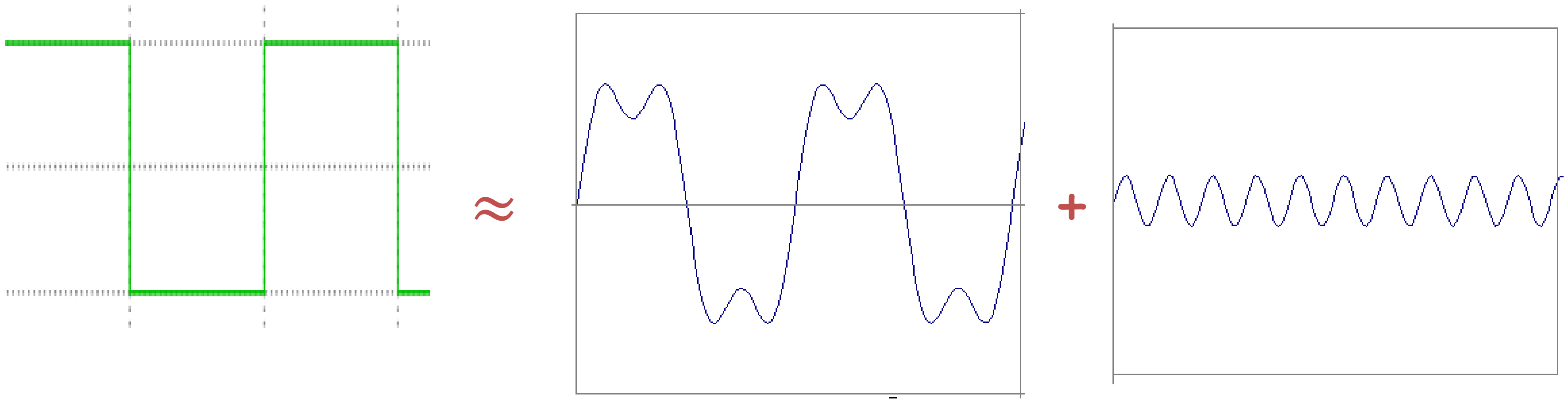
Frequency Spectra

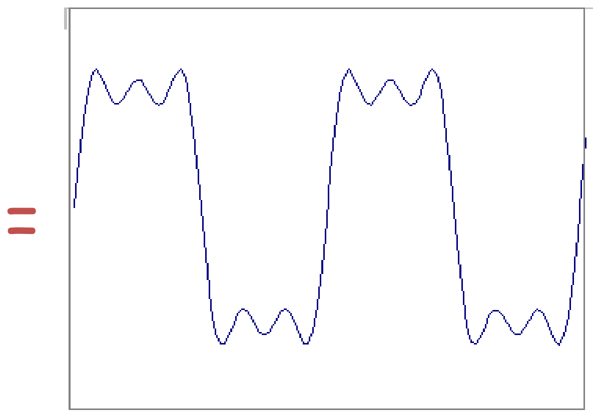
- Example: $g(t) = \sin(2\pi f\, t) + (1/3)\sin(2\pi(3f)\, t)$





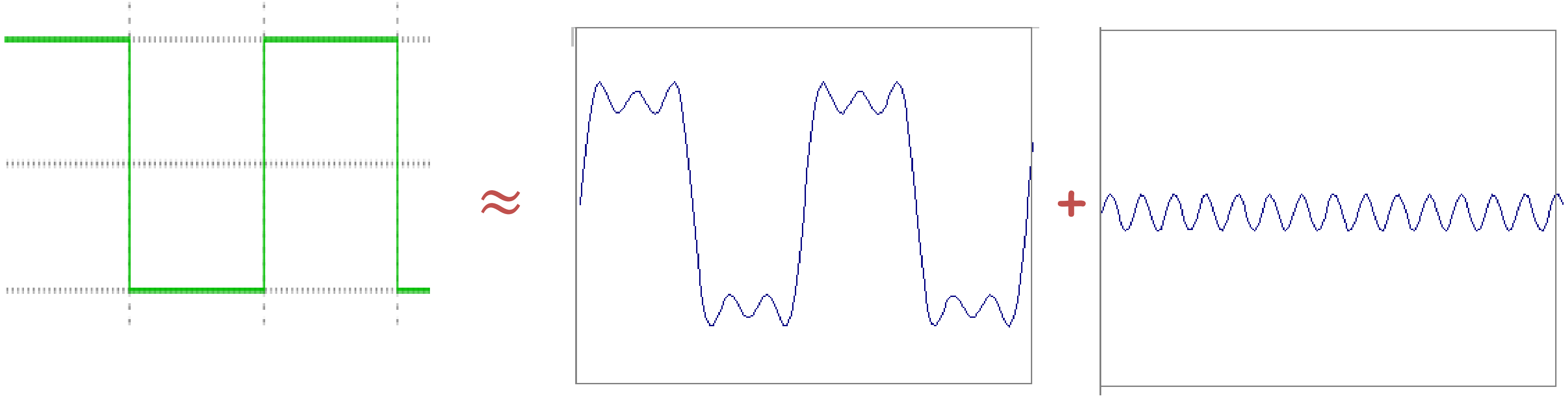
Frequency Spectra

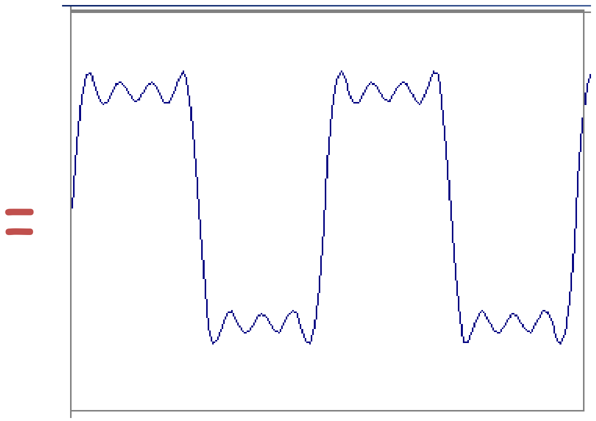




$(1/5)\sin(2\pi(5f)\,t)$

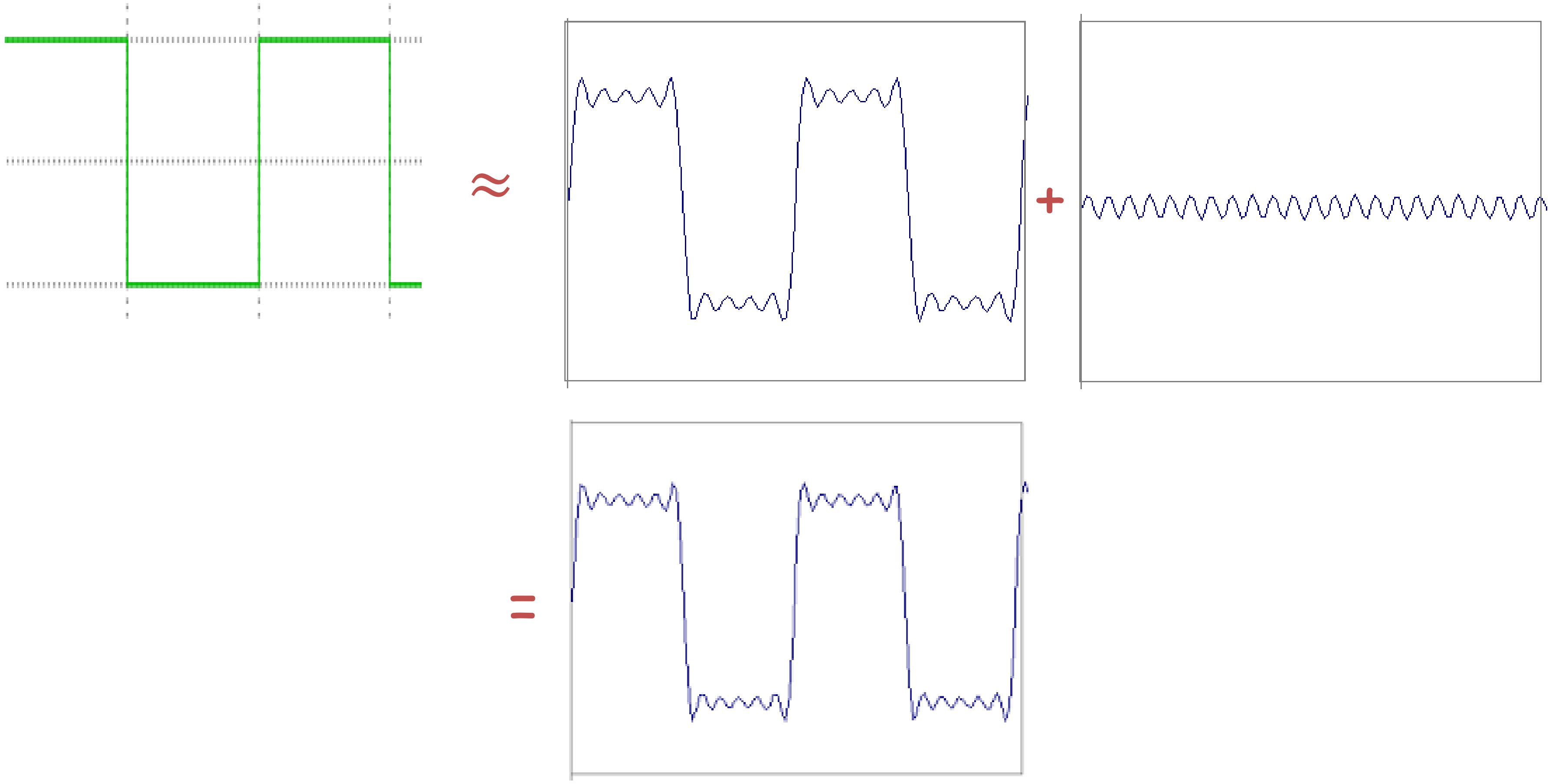
Frequency Spectra



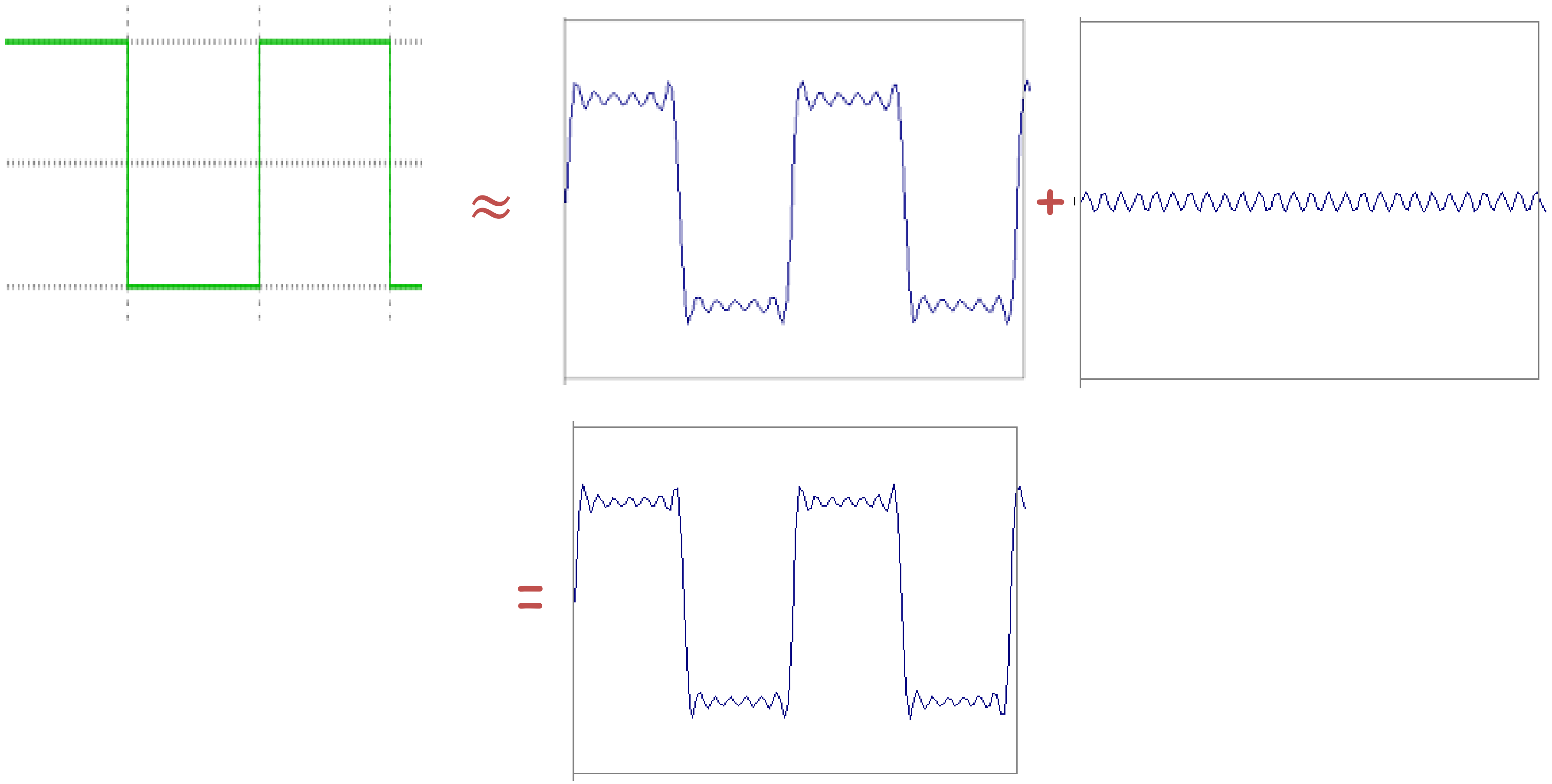


$(1/7)\sin(2\pi(7f)\ t)$

Frequency Spectra

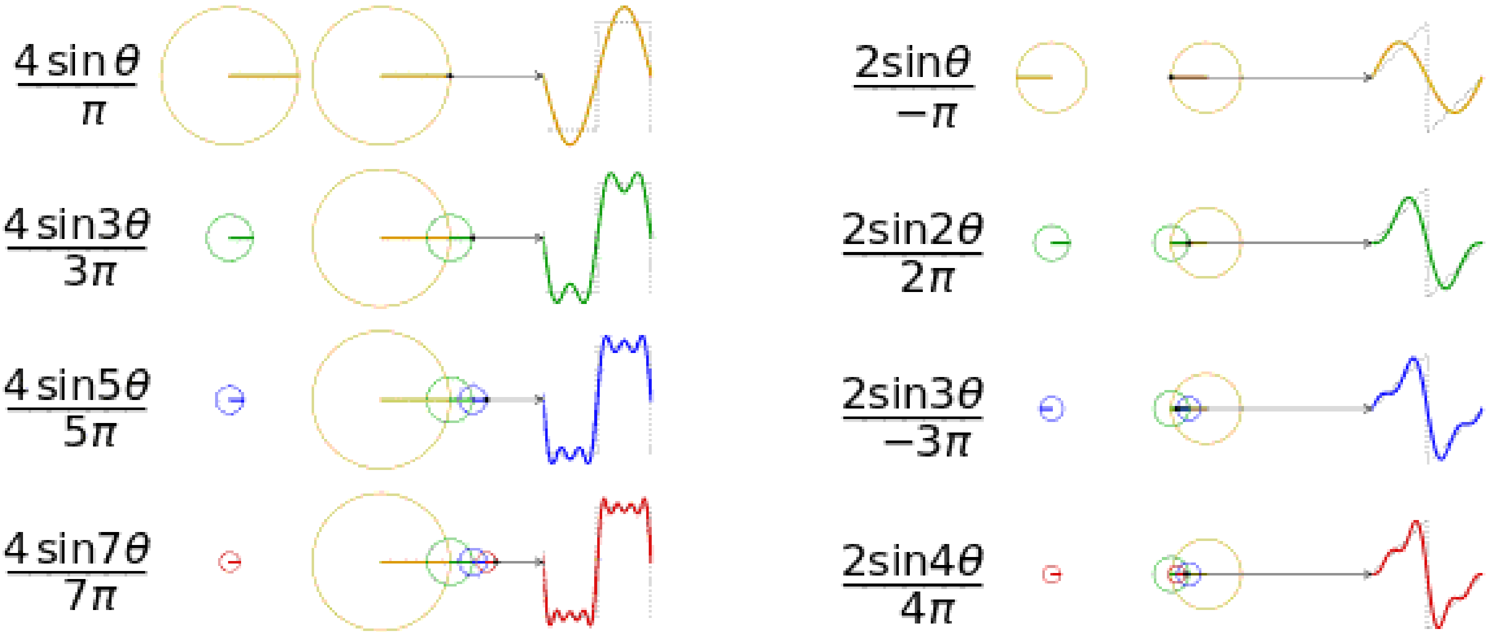


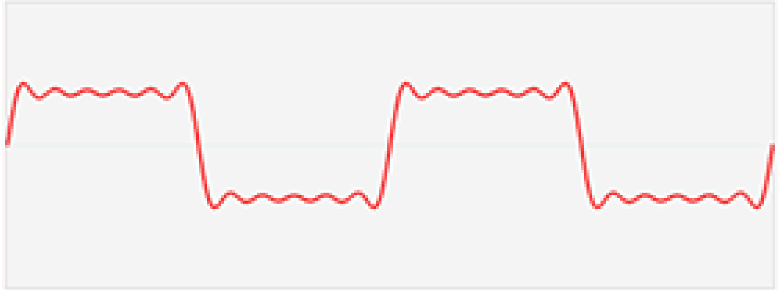
Frequency Spectra



Frequency Spectra

- Interesting explanations

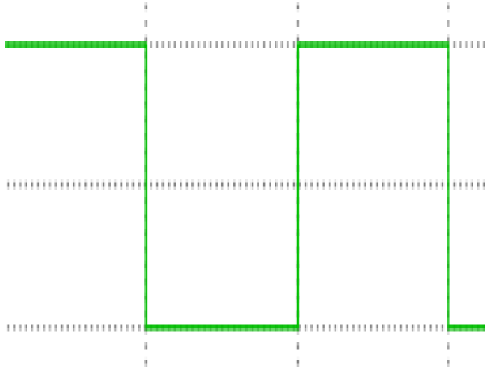




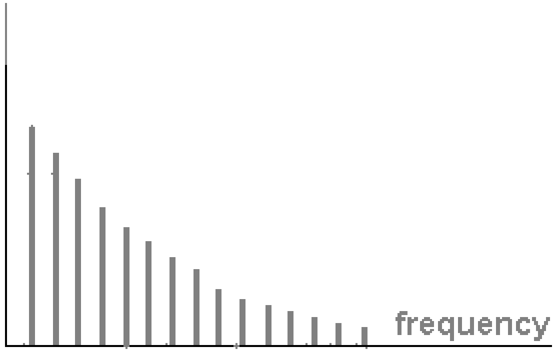
https://bl.ocks.org/jinroh/7524988

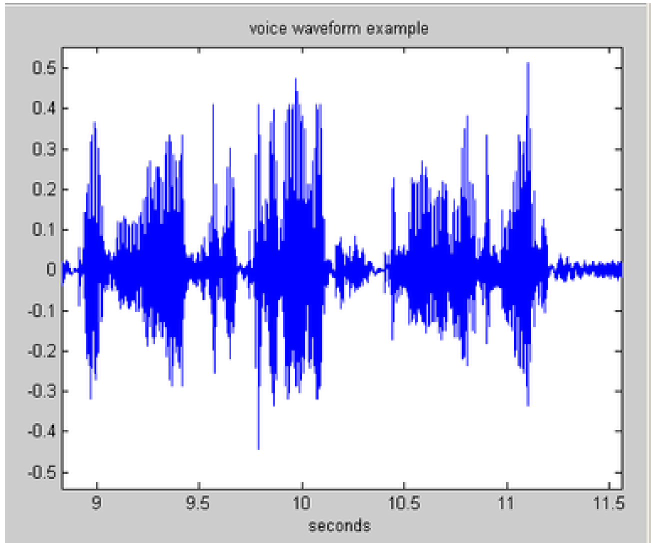
Frequency Spectra

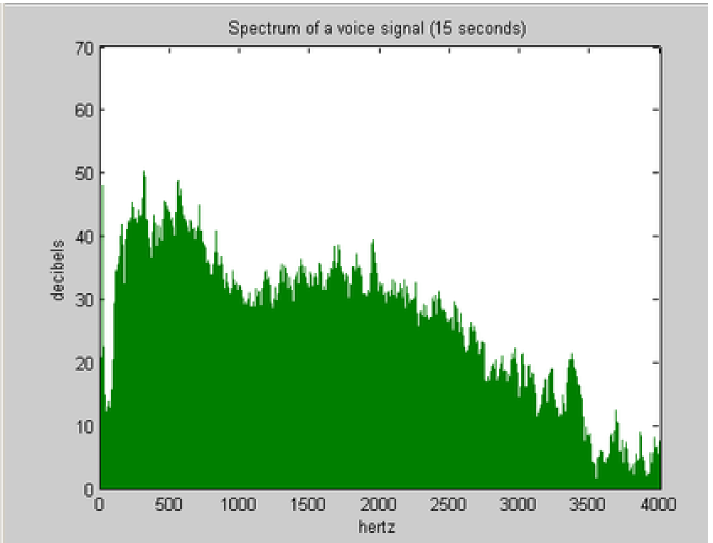
- We think of music in terms of frequencies at different magnitudes



$$= \boxed{A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kt)}$$







Two-dimensional Fourier Transforms

- An oriented sinusoid $s(x,y) = \sin(\omega_x x + \omega_y y)$

- The corresponding two-dimensional Fourier transforms

$$H(\omega_x, \omega_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x,y)e^{-j(\omega_x x + \omega_y y)}dx\,dy$$

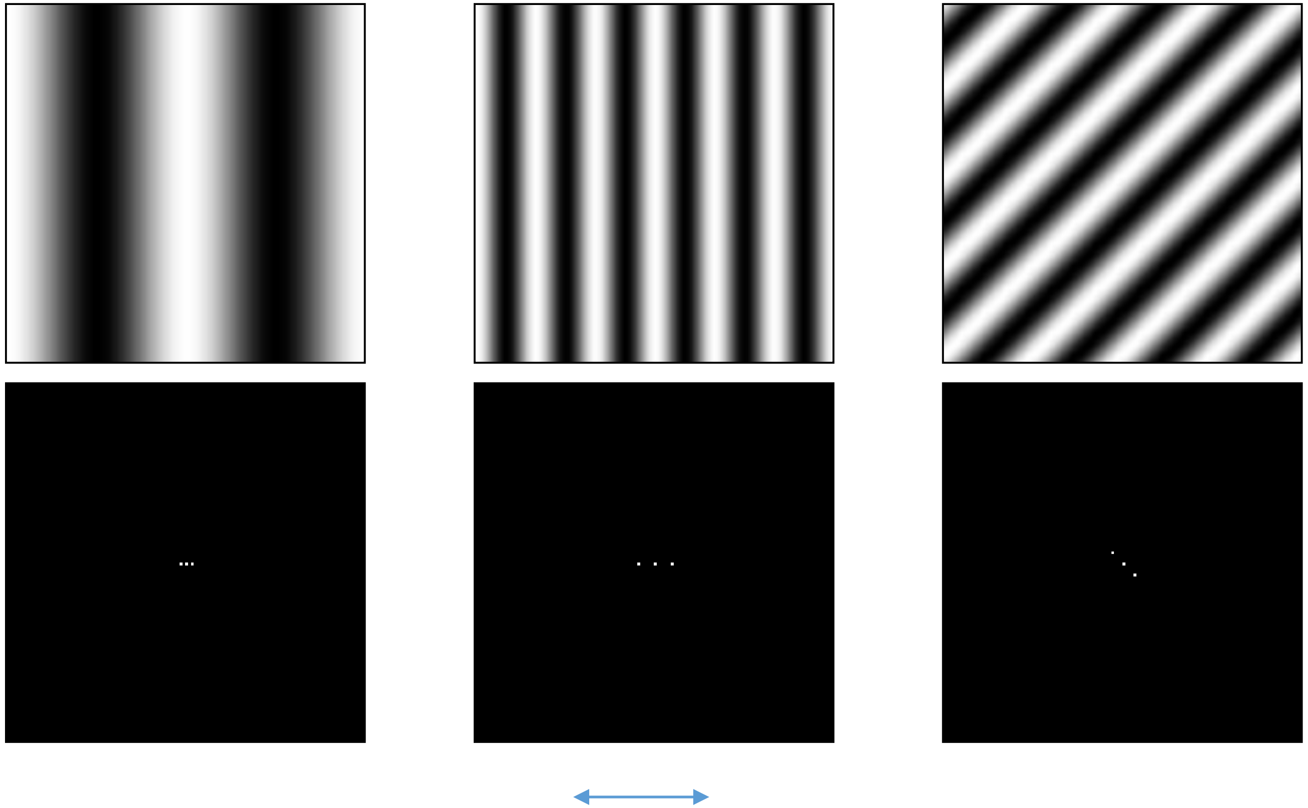
- In the discrete domain

$$H(k_x, k_y) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} h(x,y)e^{-j2\pi\frac{k_x x + k_y y}{MN}}$$

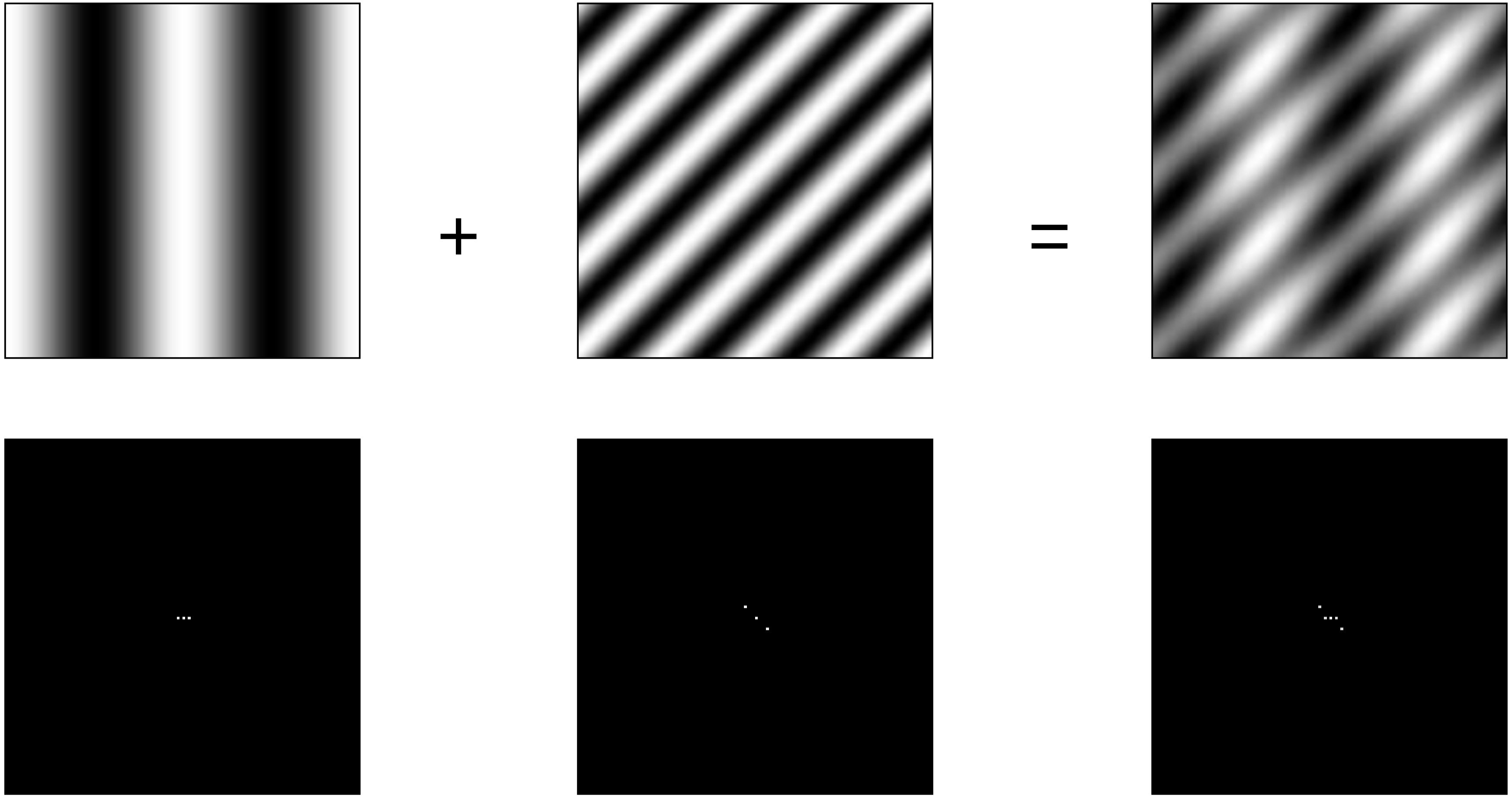
Fourier analysis in images

Intensity Image

Fourier Image

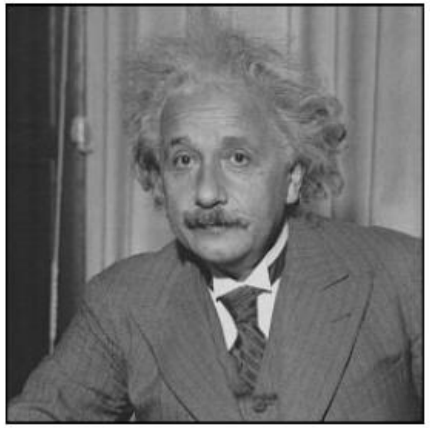
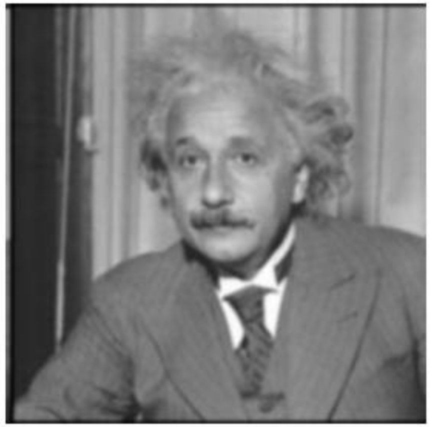


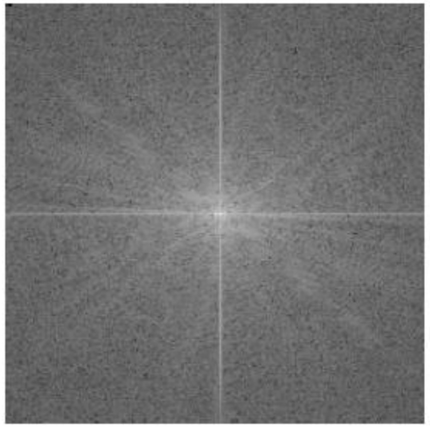
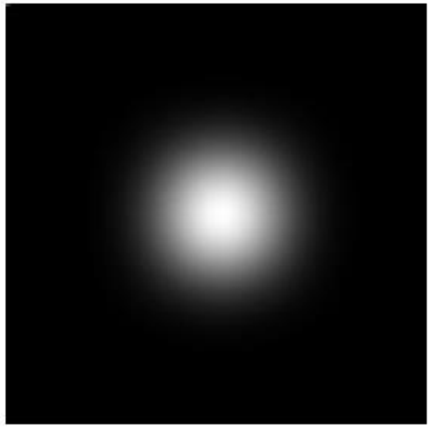
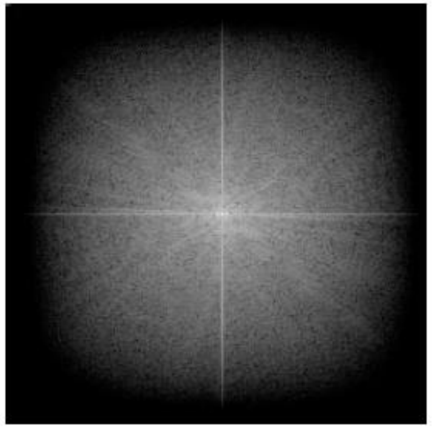
Signals can be composed: linear operation



Example of 2D Fourier Transform

- Smoothing

Fourier Transform

- Fourier transform stores the **magnitude** and phase at each frequency
  - ➢ Magnitude encodes how much signal there is at a particular frequency
  - ➢ Phase encodes spatial information (indirectly)
  - ➢ For mathematical convenience, this is often notated in terms of real and complex numbers
  - ➢ Fourier transform of a real signal is symmetric about the origin
- The Convolution Theorem

$$F[g * h] = F[g]F[h] \qquad g * h = F^{-1}[F[g]F[h]]$$

Pyramids

Pyramids

- What we have studied
  - All of the image transformations produce output images of the same size as the inputs
- Change the resolution of an image
  - Example: finding a face in an image
    - ✓ Accelerating the search for an object
    - ✓ Performing multi-scale editing operations
- Operations
  - Upsampling
  - Downsampling

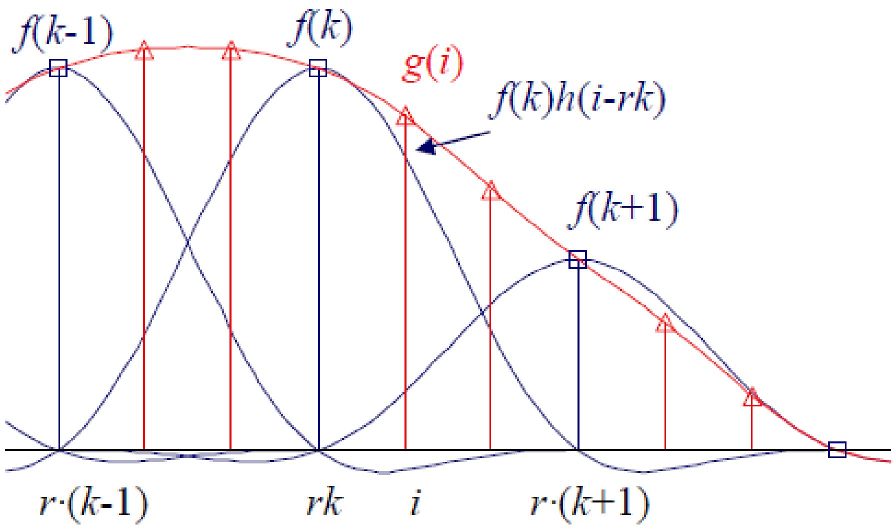
Interpolation

- Select some interpolation kernels to convolve the image

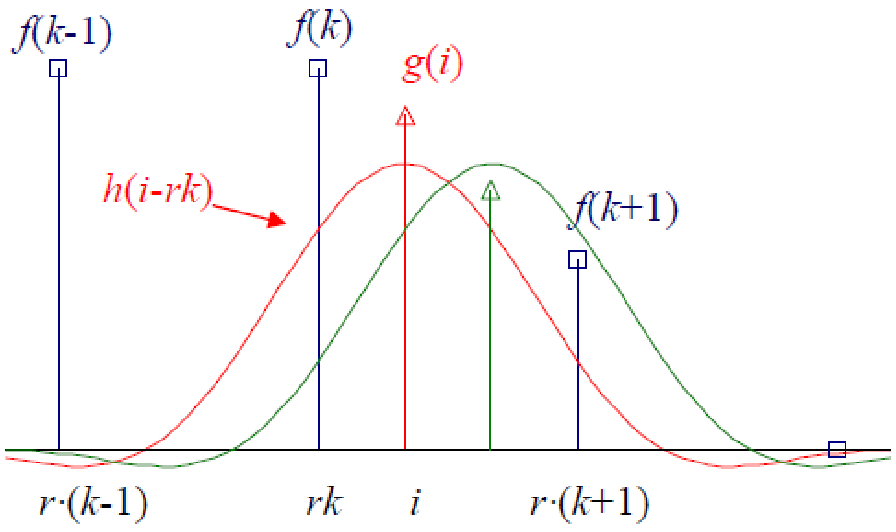
$$g(i,j) = \sum_{k,l} f(k,l)h(i-rk, j-rl)$$

the upsampling rate

$$\boxed{g(i) = \sum_k f(k)h(i-rk)}$$



weighted summation of input values

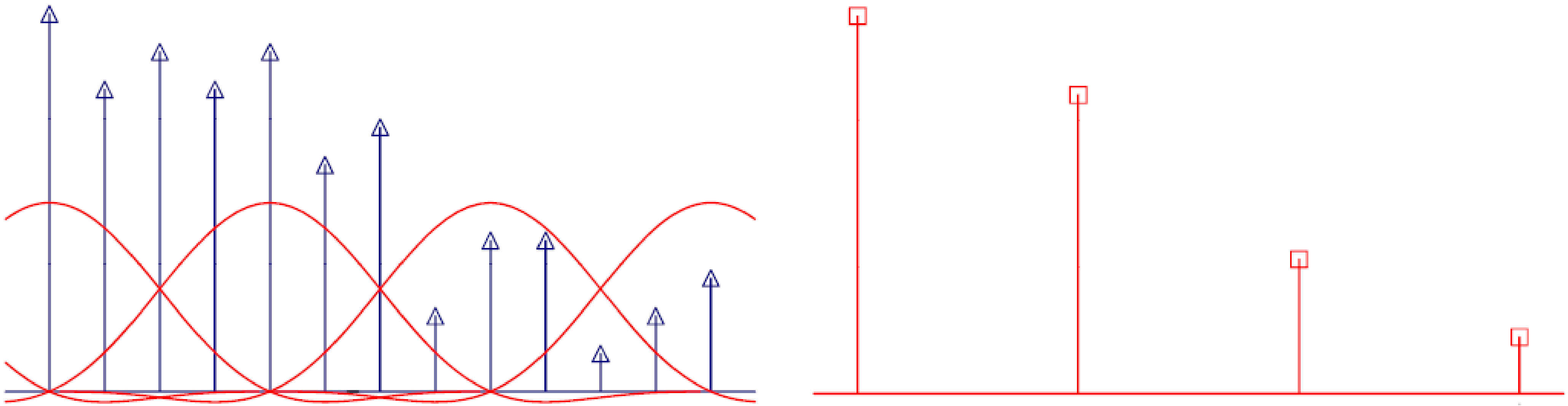


polyphase filter interpretation

Decimation

- Decimation (down-sampling) is required to reduce the resolution
  - ➢ Evaluate the convolution at every rth sample ()
    - ✓ First convolve the image with a low-pass filter
    - ✓ Then keep every rth sample

$$g(i,j) = \sum_{k,l} f(k,l)h(ri-k, rj-l)$$

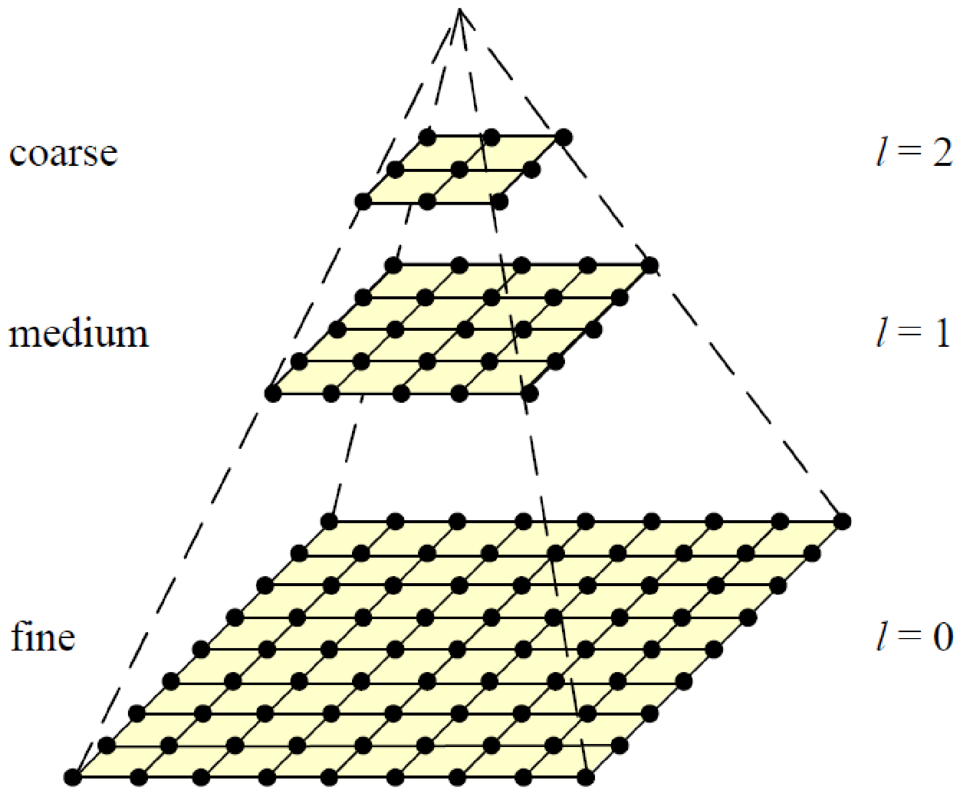


Multi-Resolution Representations

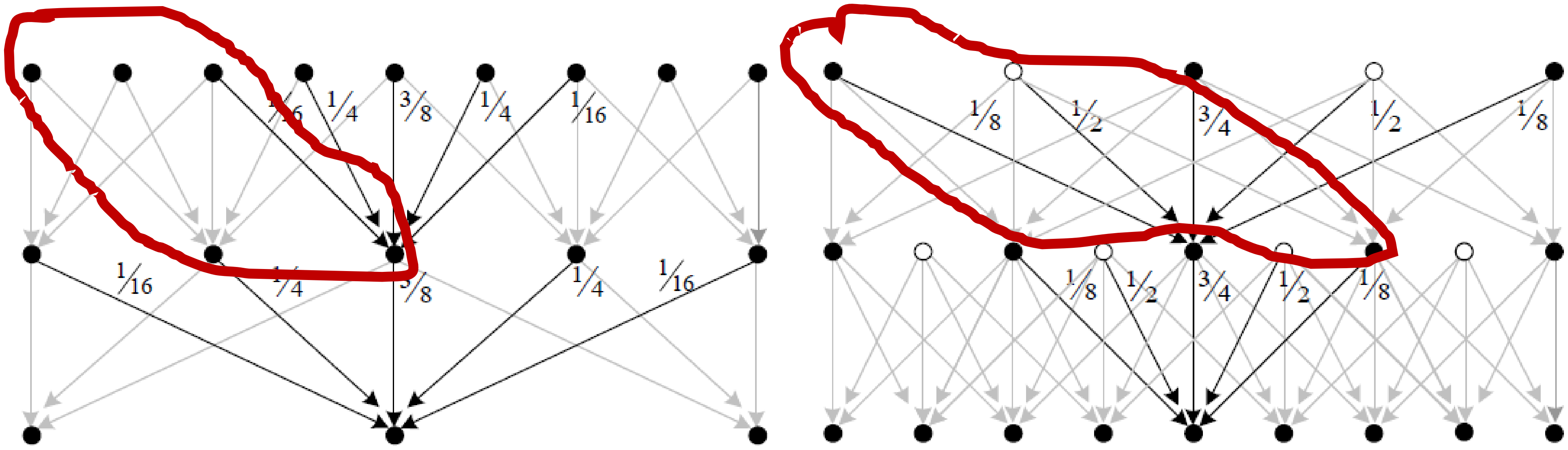
- Octave pyramids
  - ➢ Can be used to accelerate coarse-to-fine search algorithms

$$\frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline\end{array}$$

A traditional image pyramid



Multi-Resolution Representations



(a) (b)

Figure 3.33 The Gaussian pyramid shown as a signal processing diagram: The (a) analysis and (b) re-synthesis stages are shown as using similar computations. The white circles indicate zero values inserted by the ↑ 2 upsampling operation. Notice how the reconstruction filter coefficients are twice the analysis coefficients. The computation is shown as flowing down the page, regardless of whether we are going from coarse to fine or *vice versa*.

Gaussian Pyramid

- A Laplacian of Gaussian (LoG)

$$\mathrm{LoG}\{I;\sigma\} = \nabla^2(G_\sigma * I) = (\nabla^2 G_\sigma) * I,$$

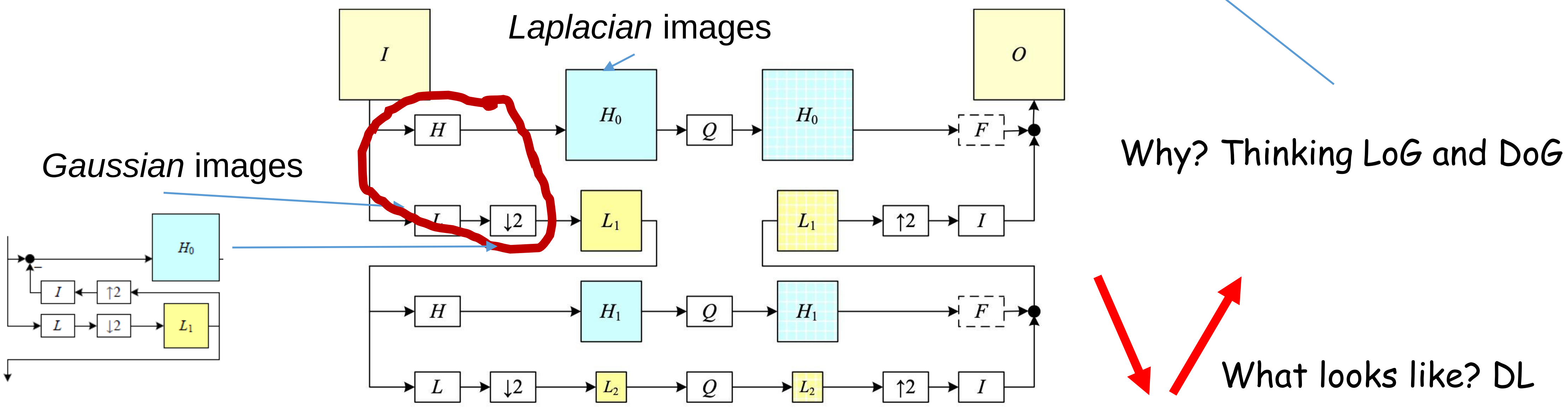
$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

- Difference of Gaussians (DoG)
  - ➢ To obtain an approximation of the Laplacian of Gaussian

$$\mathrm{DoG}\{I;\sigma_1,\sigma_2\} = G_{\sigma_1} * I - G_{\sigma_2} * I = (G_{\sigma_1} - G_{\sigma_2}) * I$$

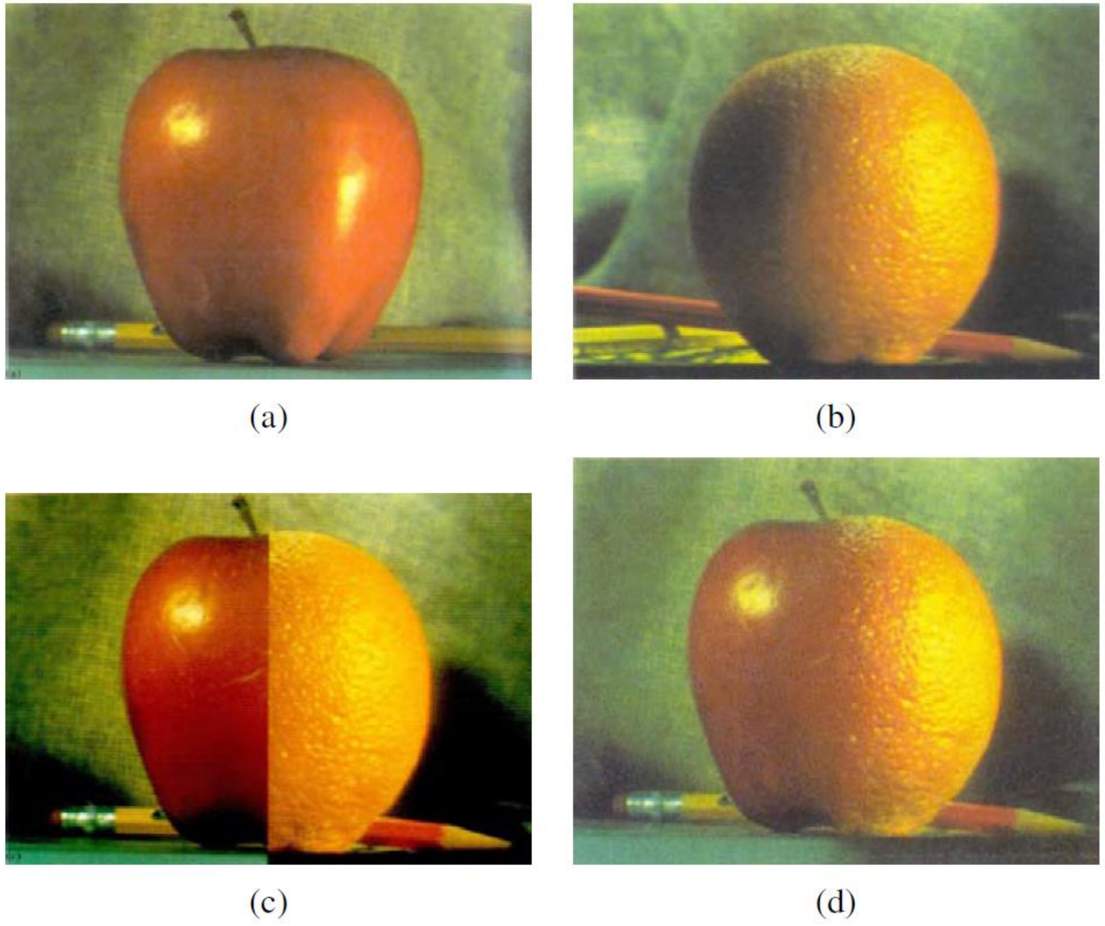
Multi-Resolution Representations

- To compute the Laplacian pyramid
  - First interpolate a lower resolution image to obtain a reconstructed low-pass (short-pass) version (high frequency)
  - Then subtract this low-pass version from the **original** to yield the band-pass "Laplacian" image



Application: Image Blending

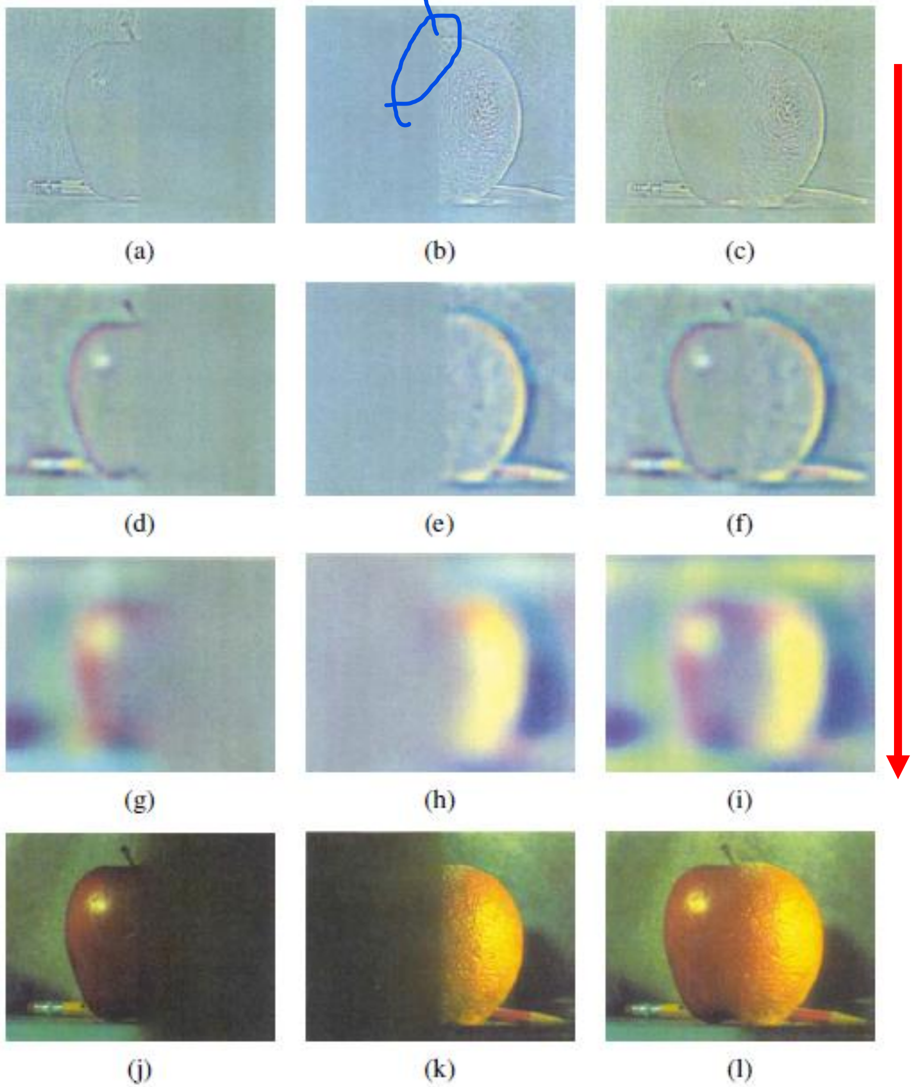
- Keys:
  - The low-frequency color variations between the red apple and the orange are smoothly blended
  - The higher-frequency textures on each fruit are blended more quickly to avoid "ghosting" effects when two textures are overlaid



(c) regular splice, (d) pyramid blend.

Application: Image Blending

- Steps:
  - Each source image is first decomposed into its own Laplacian pyramid
  - Each band is then multiplied by a smooth weighting function
    - ✓ Create these weights is to take a binary mask image
  - Each Laplacian pyramid image is then multiplied by its corresponding Gaussian mask
  - The sum of these two weighted pyramids is then used to construct the final image

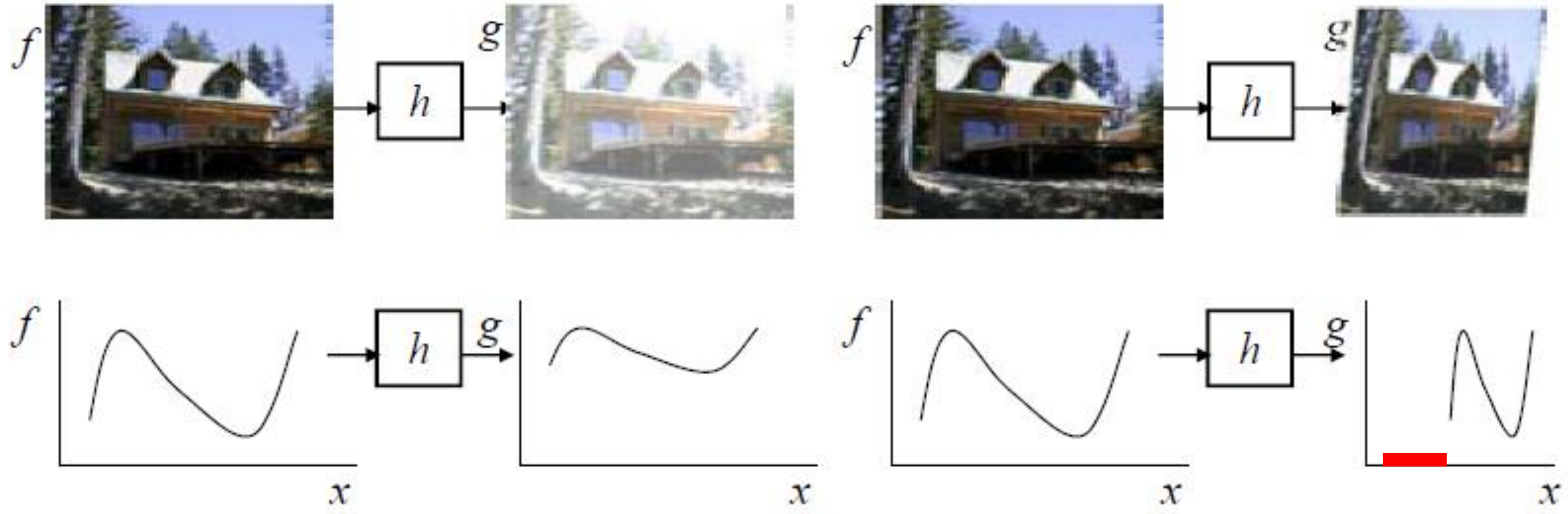


Geometric Transformations

Geometric Transformations

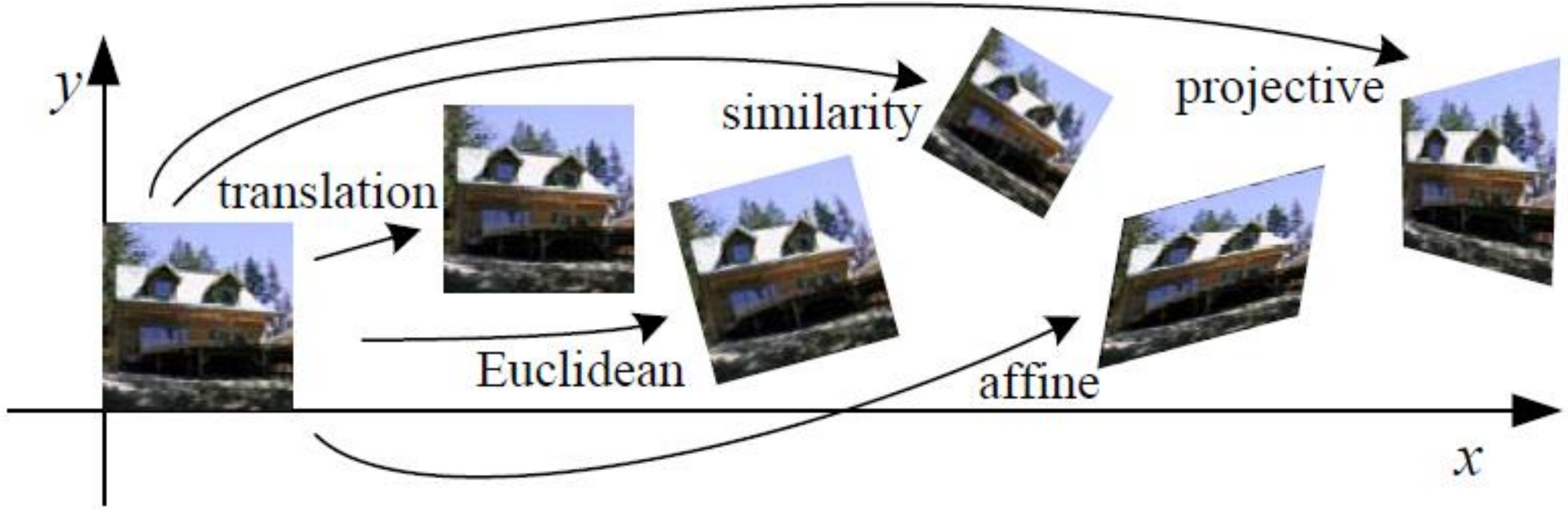
- General transformations
  - Image rotations
  - General warps

$$g(\boldsymbol{x}) = h(f(\boldsymbol{x}))$$



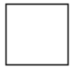
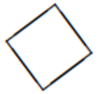
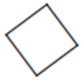
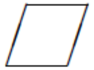
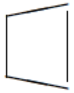
Geometric Transformations

- Basic set of 2D geometric image transformations



Parametric Transformations

- Apply a global deformation to an image
- Controlled by a small number of parameters

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\left[\begin{array}{c\|c} I & t \end{array}\right]_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\left[\begin{array}{c\|c} R & t \end{array}\right]_{2\times 3}$	3	lengths	
similarity	$\left[\begin{array}{c\|c} sR & t \end{array}\right]_{2\times 3}$	4	angles	
affine	$\left[\begin{array}{c} A \end{array}\right]_{2\times 3}$	6	parallelism	
projective	$\left[\begin{array}{c} \tilde{H} \end{array}\right]_{3\times 3}$	8	straight lines	

Forward Warping Algorithm

- Limitations:
  - Target location has a non-integer value
    - ✓ round the value
    - ✓ "distribute" the value
  - The second major problem with forward warping is the appearance of cracks and holes

procedure *forwardWarp*$(f, \boldsymbol{h},$ **out** $g)$:

For every pixel $\boldsymbol{x}$ in $f(\boldsymbol{x})$

1. Compute the destination location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$.
2. Copy the pixel $f(\boldsymbol{x})$ to $g(\boldsymbol{x}')$.

Forward Warping Algorithm

- An example

(a) (b)

Inverse Warping

- Resampling an image at non-integer locations is a well-studied problem
- High-quality filters that control aliasing can be used

procedure $inverseWarp(f, \boldsymbol{h}, \textbf{out } g)$:

For every pixel $\boldsymbol{x}'$ in $g(\boldsymbol{x}')$

1. Compute the source location $\boldsymbol{x} = \hat{\boldsymbol{h}}(\boldsymbol{x}')$
2. Resample $f(\boldsymbol{x})$ at location $\boldsymbol{x}$ and copy to $g(\boldsymbol{x}')$

Mesh-based Warping

- Local deformations with more degrees of freedom are often required
  - Changing the appearance of a face from a frown to a smile
  - Different amounts of motion are required in different parts of the image
- First approach
  - Specify a sparse set of corresponding **points**
  - Displacement of these points can then be interpolated to a dense displacement field
  - Alternative methods for interpolating a sparse set of displacements include moving nearby quadrilateral mesh vertices

Mesh-based Warping

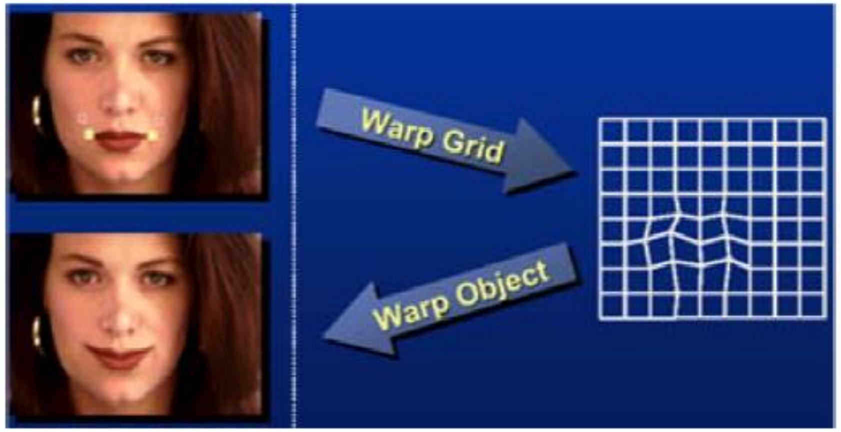
- A second approach
  - Specifying displacements for local deformations is to use corresponding oriented line
    - ✓ Pixels along each line segment are transferred from source to destination exactly as specified
    - ✓ Other pixels are warped using a smooth interpolation of these displacements
    - ✓ Each line segment correspondence specifies a translation, rotation, and scaling

- One possibility for specifying displacement fields is to use a mesh specifically adapted to the underlying image content

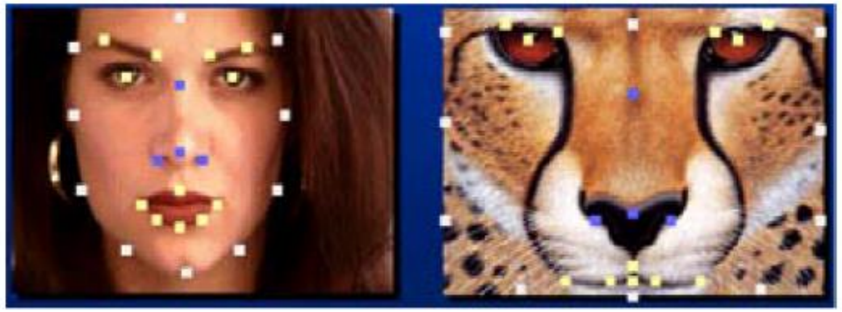
Mesh-based Warping

- Examples

(a) sparse control points->deformation grid;
(b) denser set of control point correspondences;
(c) oriented line correspondences;
(d) uniform quadrilateral grid.



(a)



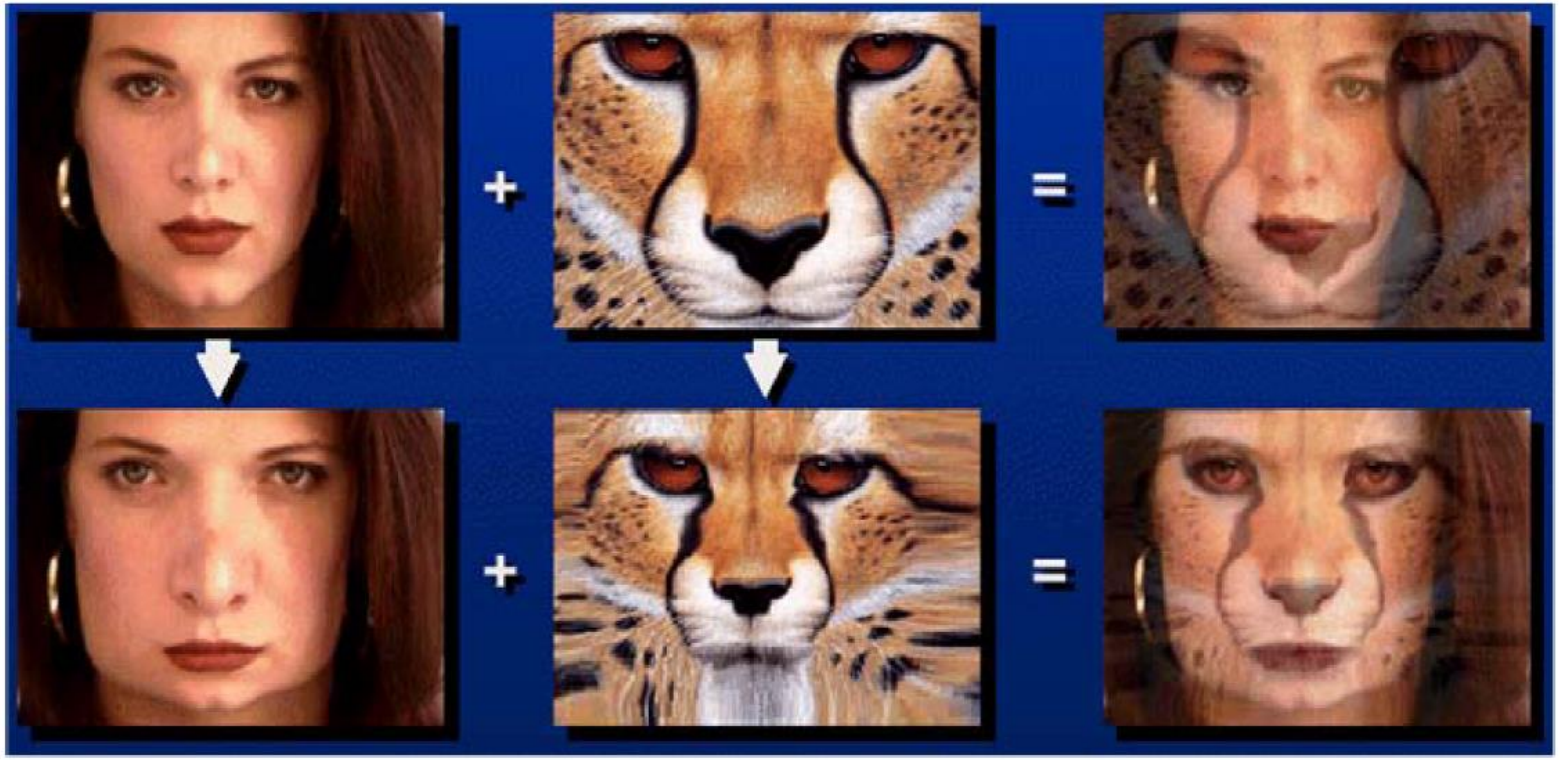
(b)

(c)

(d)

Image Morphing

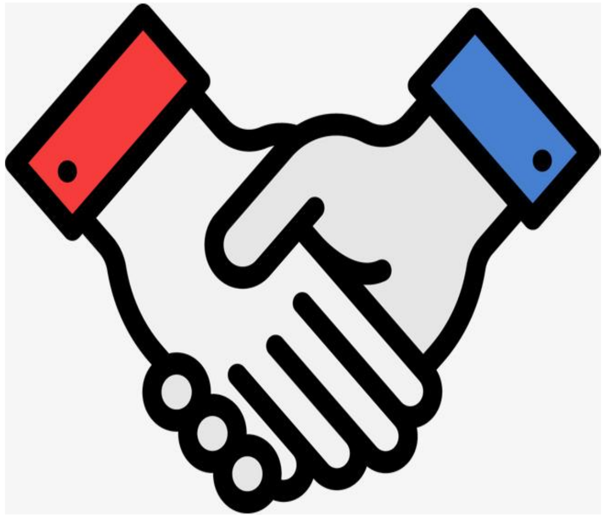
- Top row: if the two images are just blended, visible ghosting results
- Both images are first warped to the same intermediate location



Conclusions

Conclusions

- Thinking in Frequency
  - Remove noise (high frequency)
  - Theory
- Pyramids
  - Upsampling
  - Downsamping
- Geometric Transformations
  - Inverse warping (augmentation)
  - Mesh-based warping



Thanks

zhengf@sustc.edu.cn