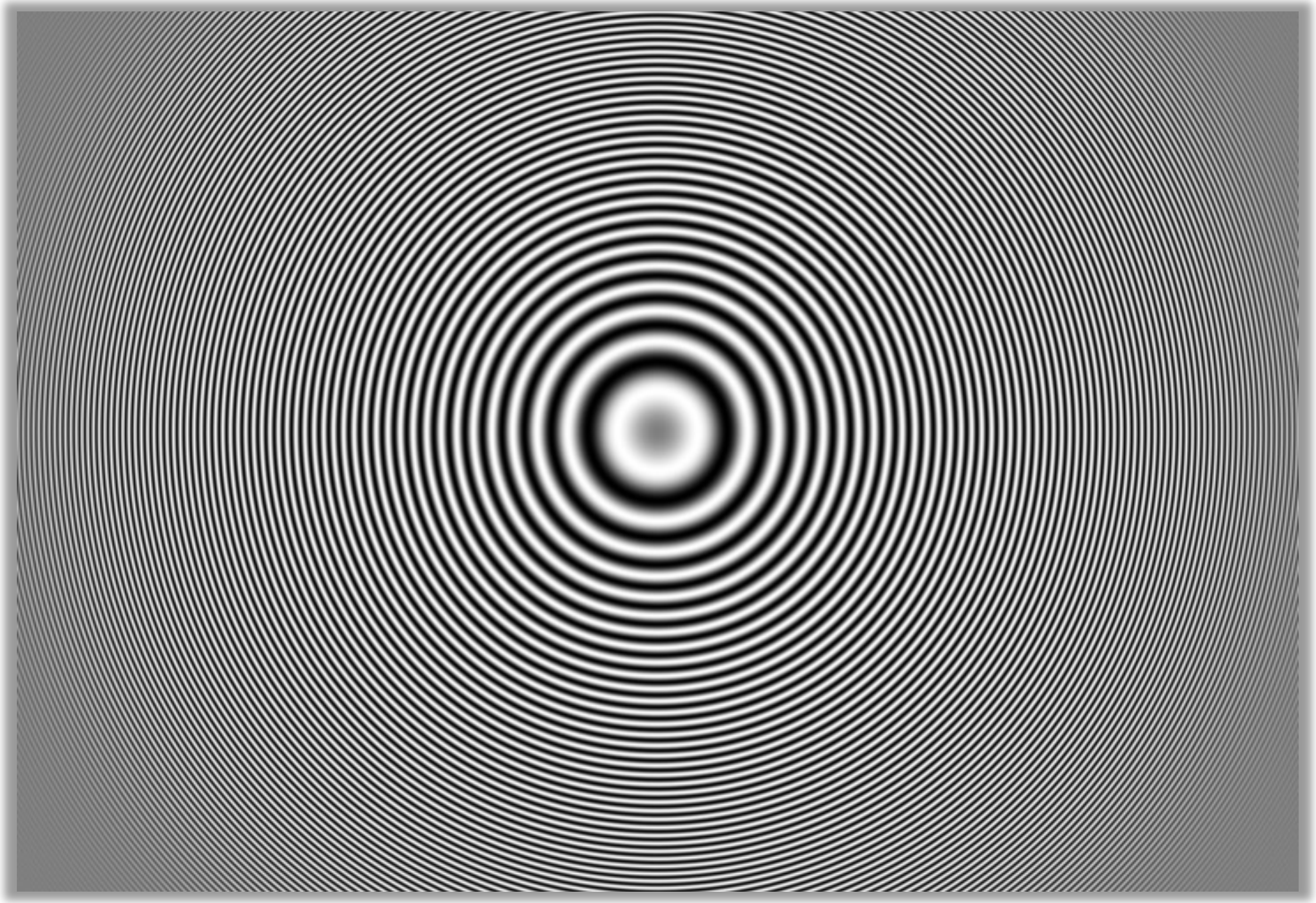
Image Pyramids

Aliasing, Anti-Aliasing, Interpolation



CSC420
David Lindell
University of Toronto
cs.toronto.edu/~lindell/teaching/420
Slide credit: Babak Taati ←Ahmed Ashraf ←Sanja Fidler

TCIG Toronto Computational Imaging Group

Logistics

- HW1 is due Friday!
- Turn in on Markus
- Post on Piazza, go to TA office hours if you need help

Finding Waldo

- Let's revisit the problem of finding Waldo
- This time he is on the road

image

Template(filter)

Finding Waldo

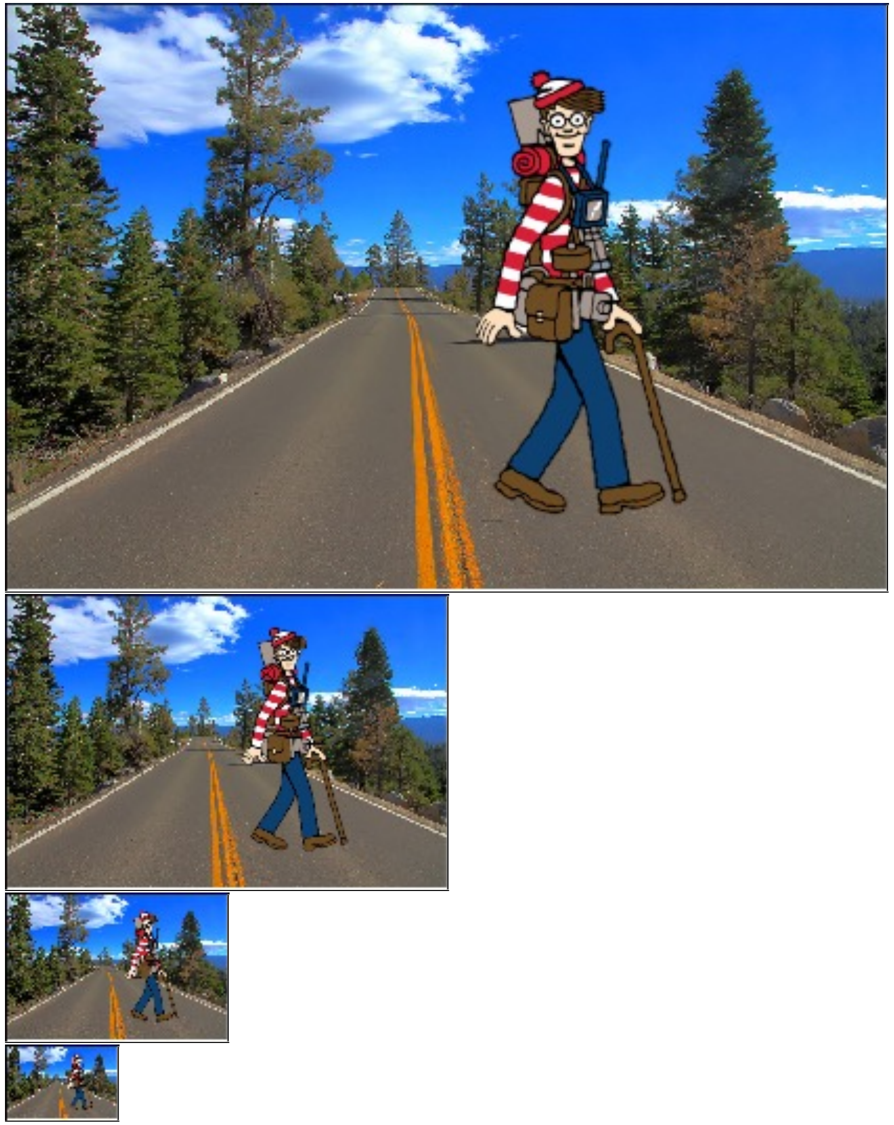
- He comes closer but our filter doesn't know that
- How can we find Waldo?

image

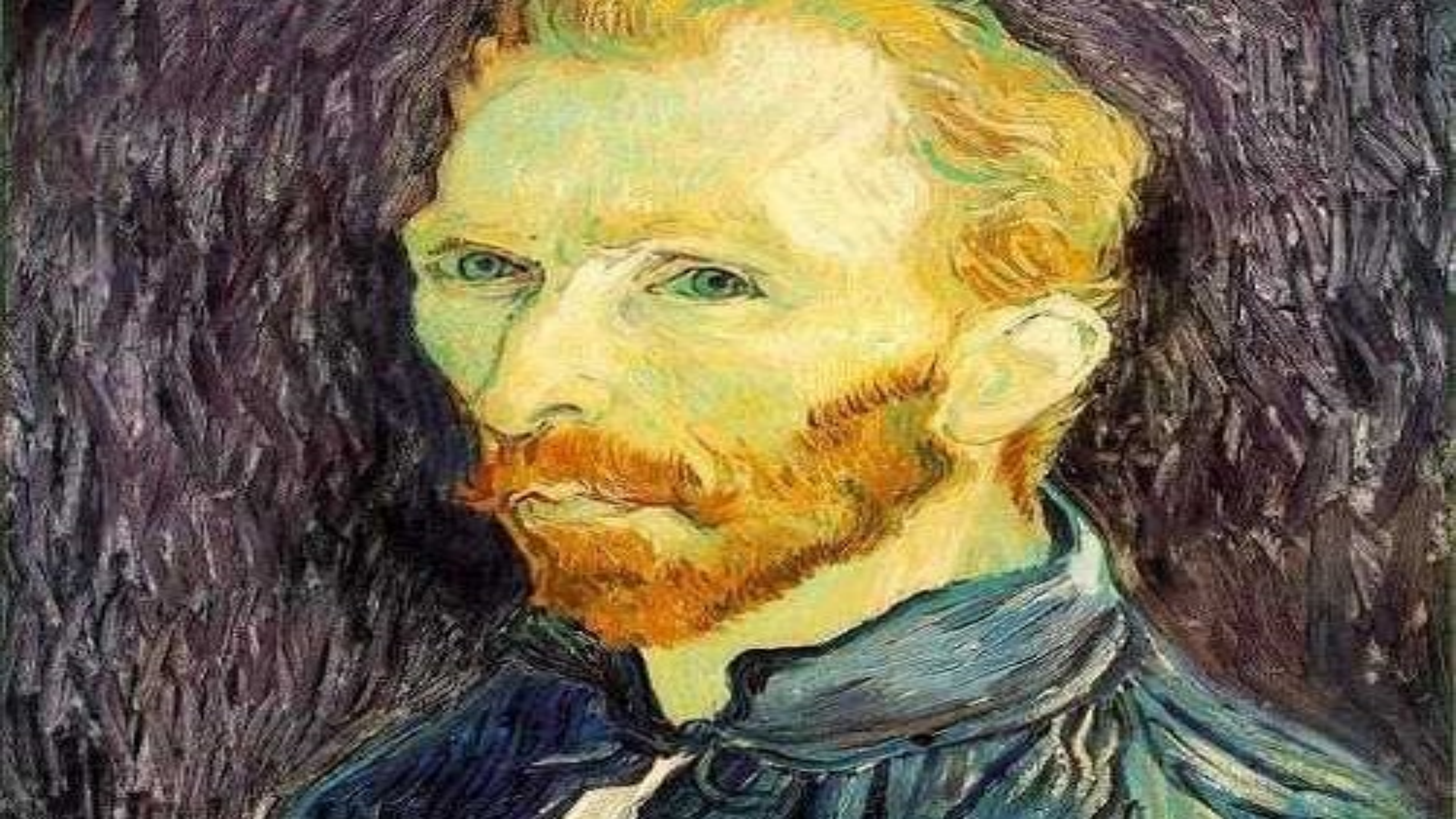
Template(filter)

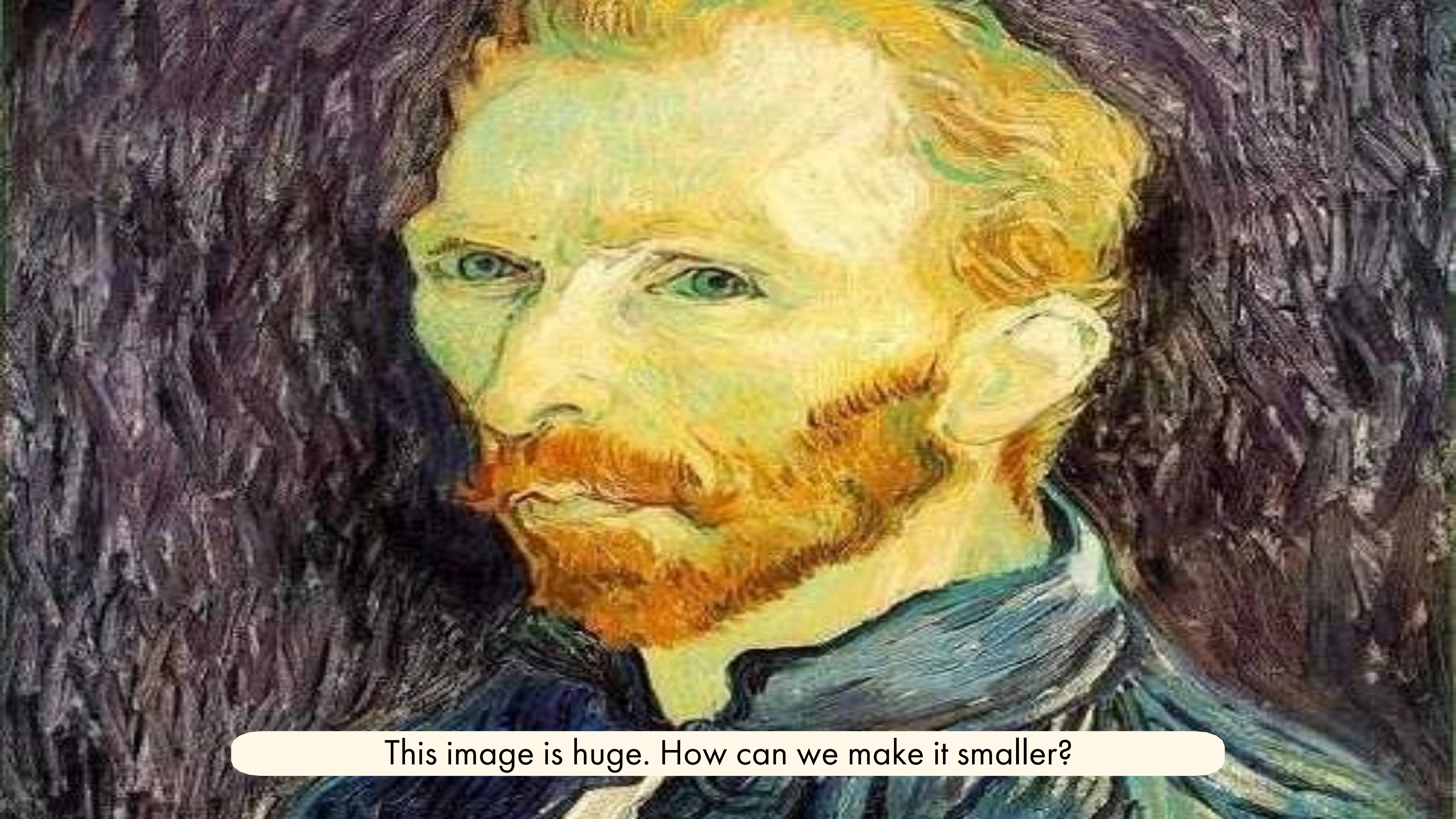
Idea: Re-size Image

- Re-scale the image multiple times! Do correlation on every size!



Template(filter)

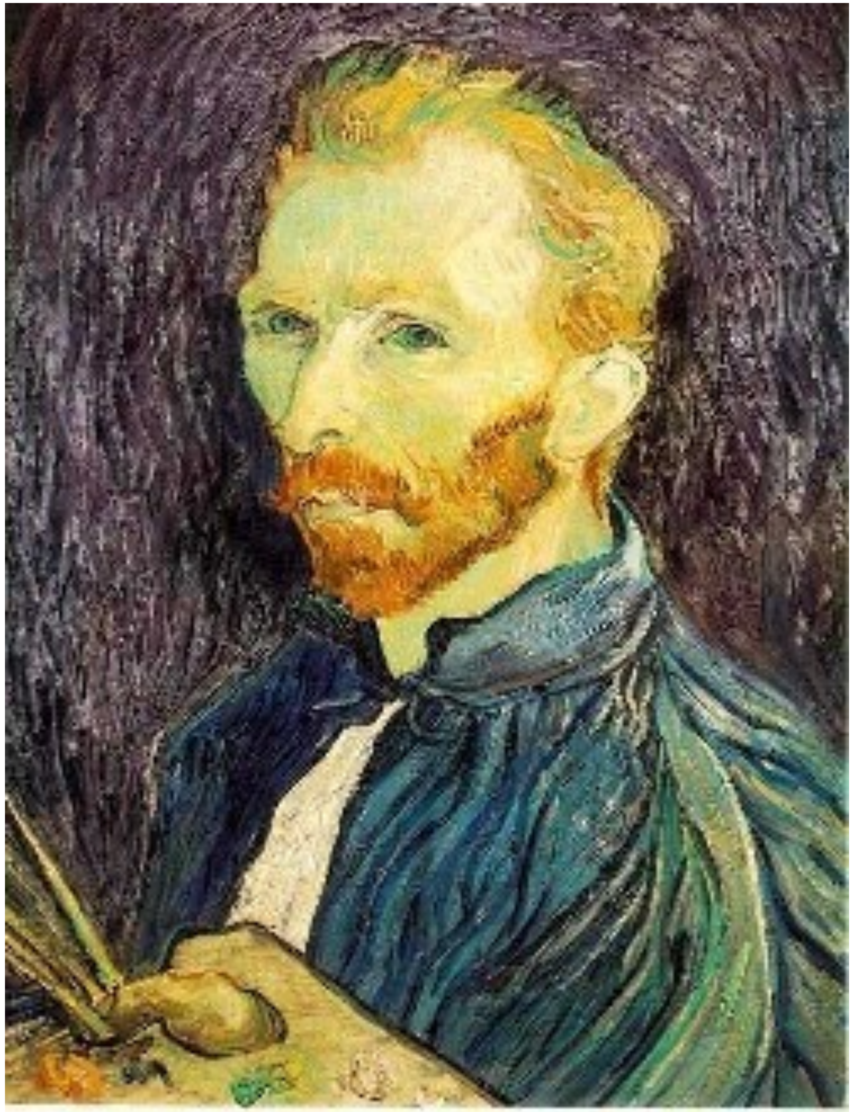




This image is huge. How can we make it smaller?

Image Sub-Sampling

- Idea: Throw away every other row and column to create a 1/2 size image



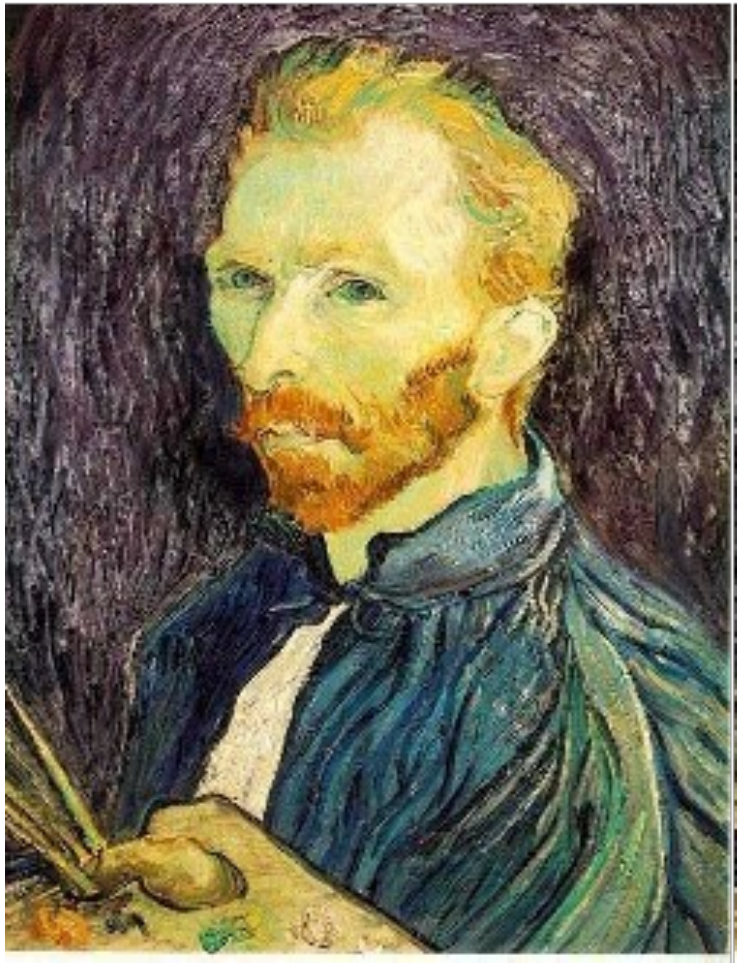
1/4

1/8

[Source: S. Seitz]

Image Sub-Sampling

- Why does this look so crufty?



1/2

1/4 (2x Zoom)

1/8 (4x Zoom)

[Source: S. Seitz]

Even worse for synthetic images

- I want to resize my image by factor 2
- And I take every other column and every other row (1st, 3rd, 5th, etc)

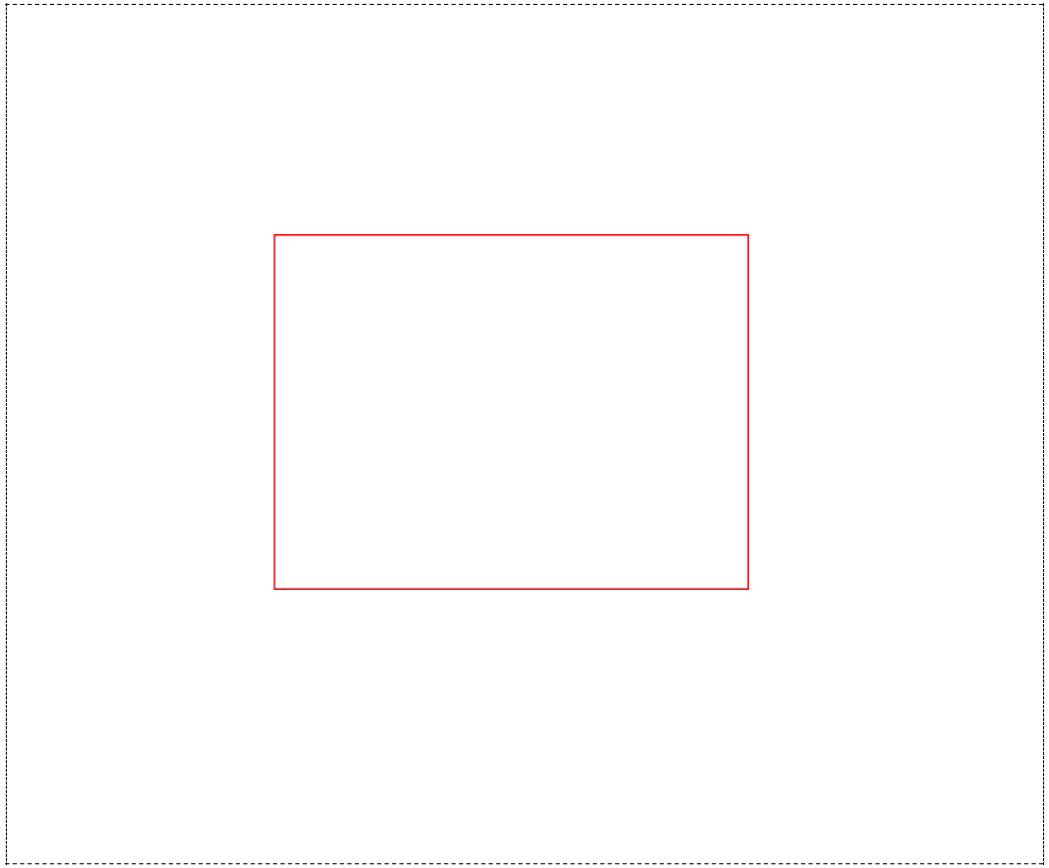


Figure: Dashed line denotes the border of the image (it’s not part of the image)

Even worse for synthetic images

- I want to resize my image by factor 2
- And I take every other column and every other row (1st, 3rd, 5th, etc)
- Where is the rectangle!

Figure: Dashed line denotes the border of the image (it’s not part of the image)

Even worse for synthetic images

- What's in the image?
- Now I want to resize my image by half in the width direction
- And I take every other column (1st, 3rd, 5th, etc)

Even worse for synthetic images

- What's in the image?
- Now I want to resize my image by half in the width direction
- And I take every other column (1st, 3rd, 5th, etc)

Even worse for synthetic images

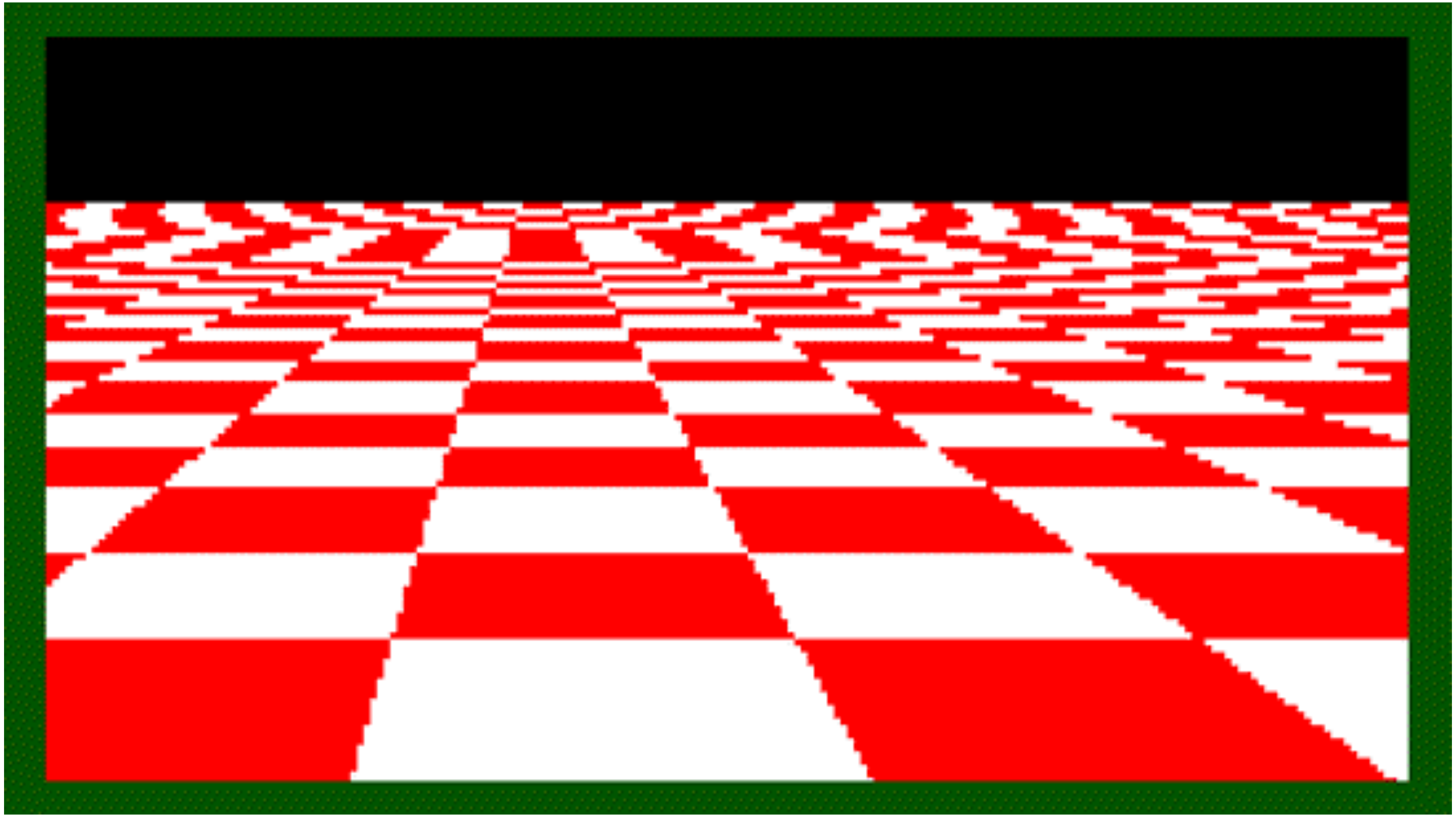
- What’s in the image?
- Now I want to resize my image by half in the width direction
- And I take every other column (1st, 3rd, 5th, etc)
- Where is the chicken!

Image Sub-Sampling

[Source: F. Durand]

Even worse for synthetic images

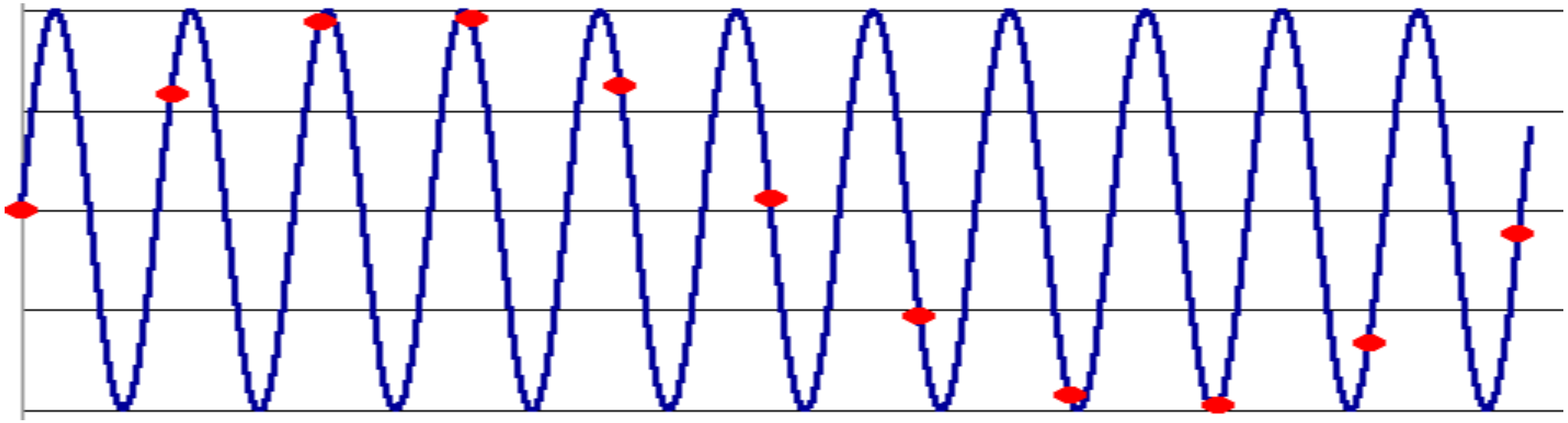
- What's happening?



[Source: L. Zhang]

Aliasing

- Occurs when your sampling rate is not high enough to capture the amount of detail in your image

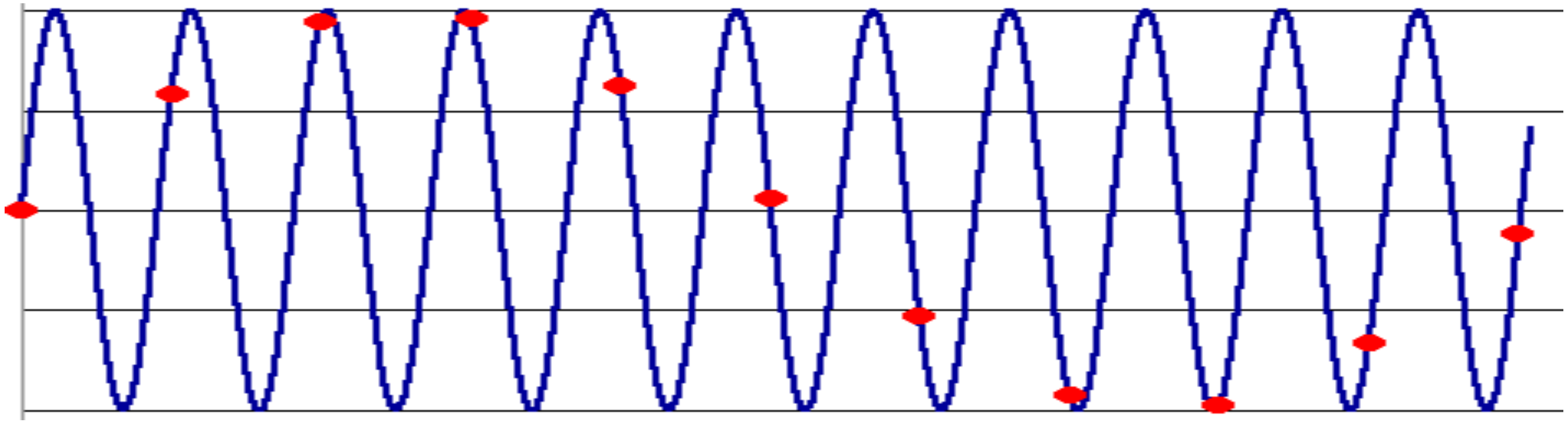


- To do sampling right, need to understand the structure of your signal/image

[Source: R. Urtasun]

Aliasing

- Occurs when your sampling rate is not high enough to capture the amount of detail in your image

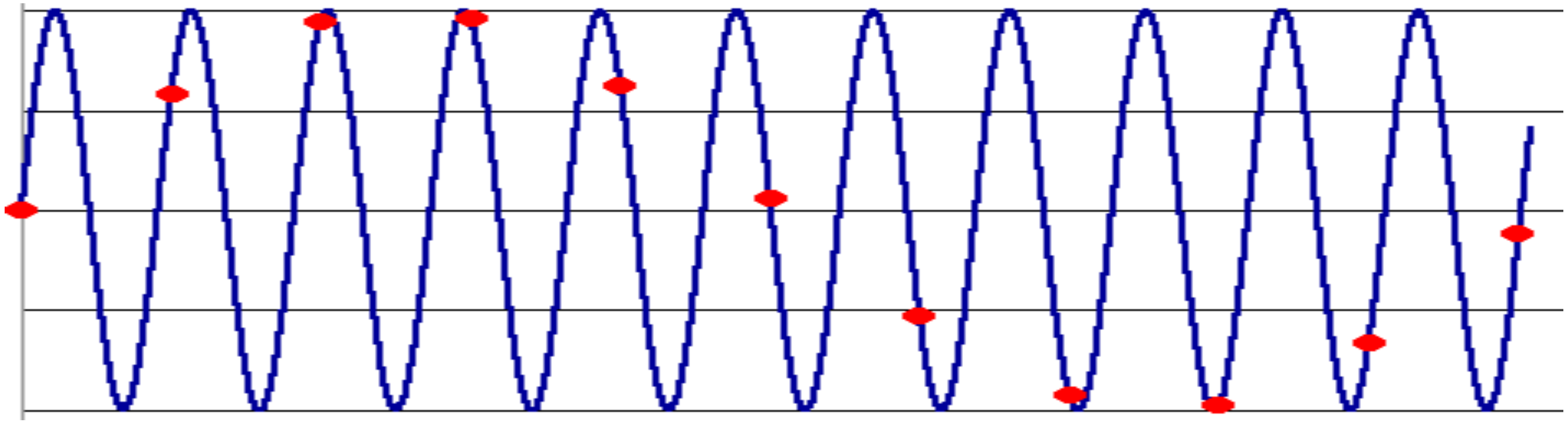


- To do sampling right, need to understand the structure of your signal/image
- The minimum sampling rate is called the Nyquist rate

[Source: R. Urtasun]

Aliasing

- Occurs when your sampling rate is not high enough to capture the amount of detail in your image



- To do sampling right, need to understand the structure of your signal/image
- The minimum sampling rate is called the Nyquist rate

[Source: R. Urtasun]

Examples of Aliasing: Temporal Aliasing

- wagon wheel effect

youtube.com/watch?v=jHS9JGkEOmA

Examples of Aliasing: Temporal Aliasing

https://www.youtube.com/watch?v=yr3ngmRuGUc

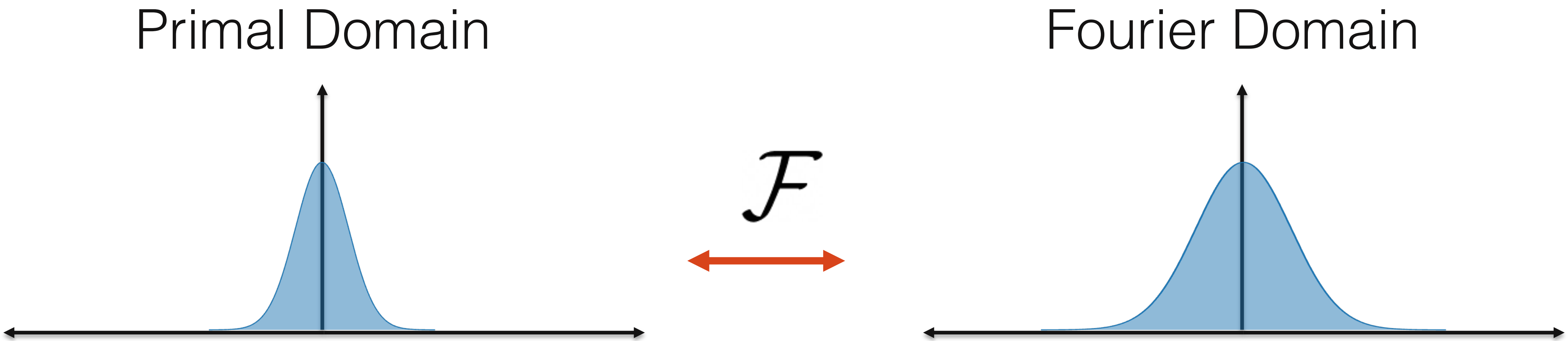
Examples of Aliasing: Temporal Aliasing

https://www.youtube.com/shorts/eTW0rNgMcKk

Mr. Nyquist

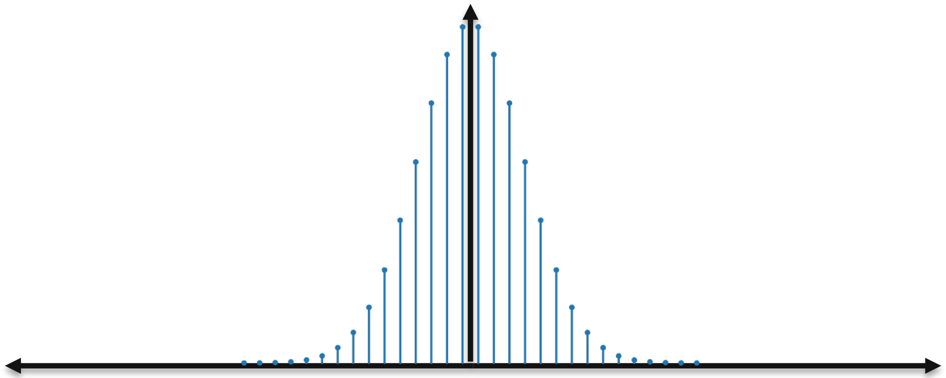
- Harry Nyquist says that one should look at the frequencies of the signal.
- One should find the highest frequency (via Fourier Transform)
- To sample properly you need to sample with at least twice that frequency
- For those interested: http://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem

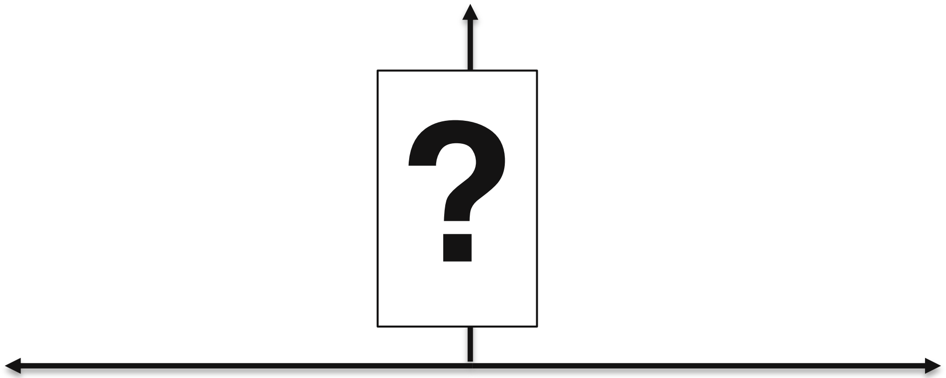
Sampling



Sampling

Primal Domain

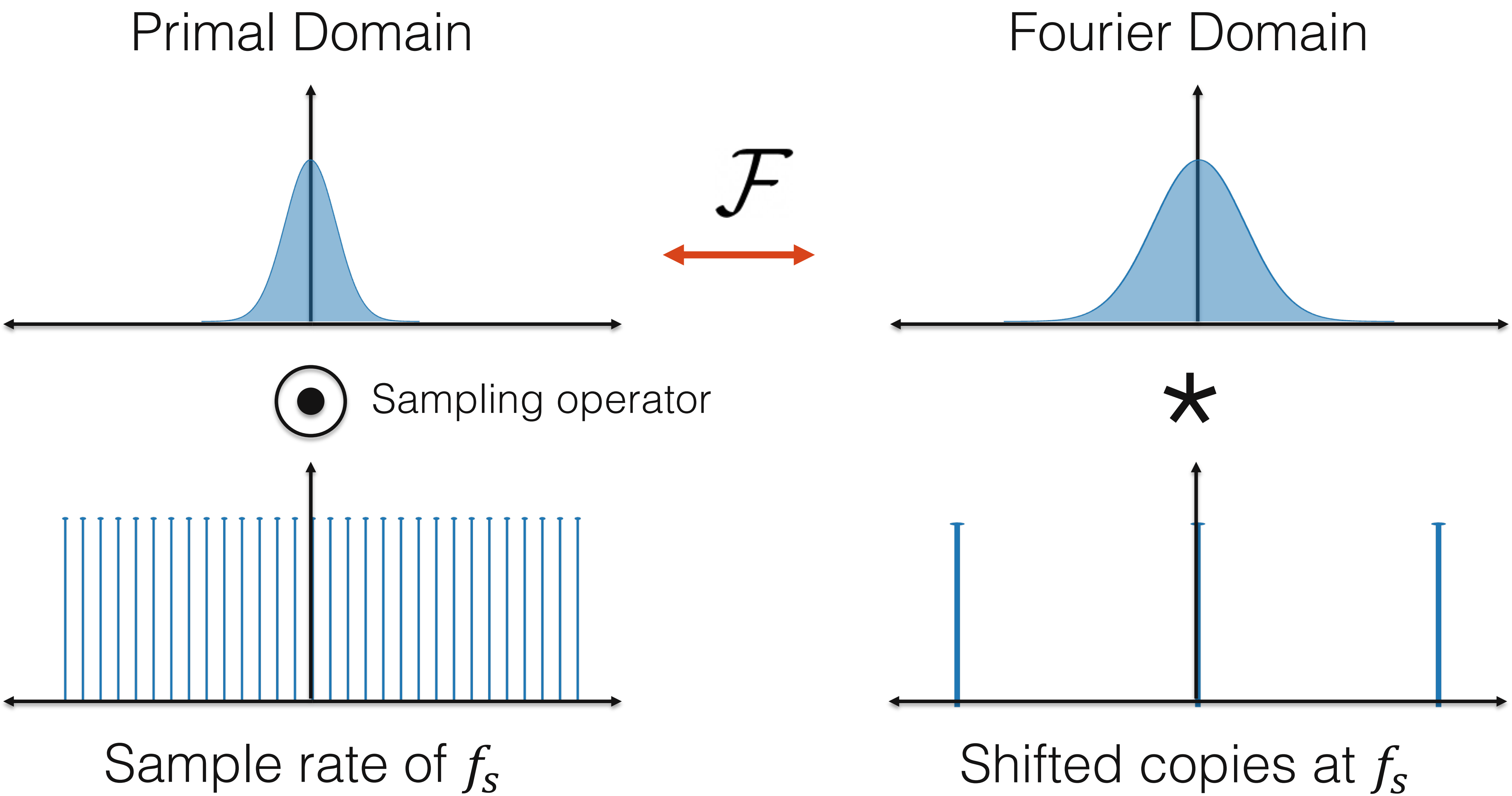




Fourier Domain

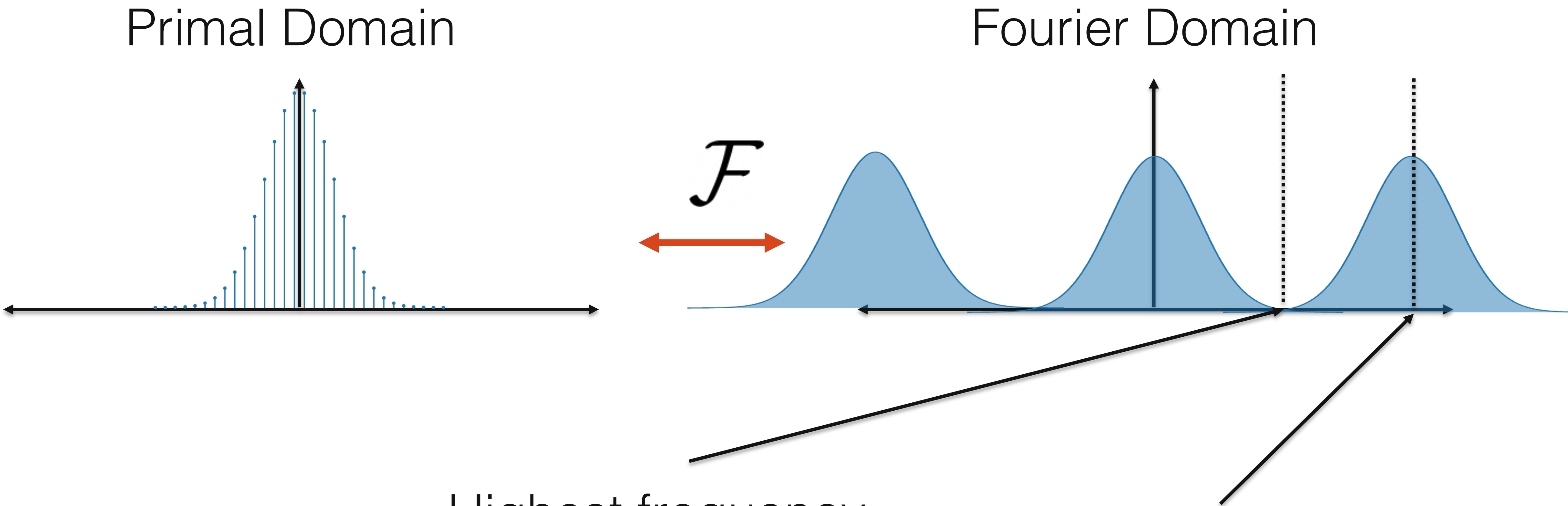
discrete sampled signal

Sampling



Primal Domain
Fourier Domain
$\mathcal{F}$
Sampling operator
*
Sample rate of f_s
Shifted copies at f_s

Sampling

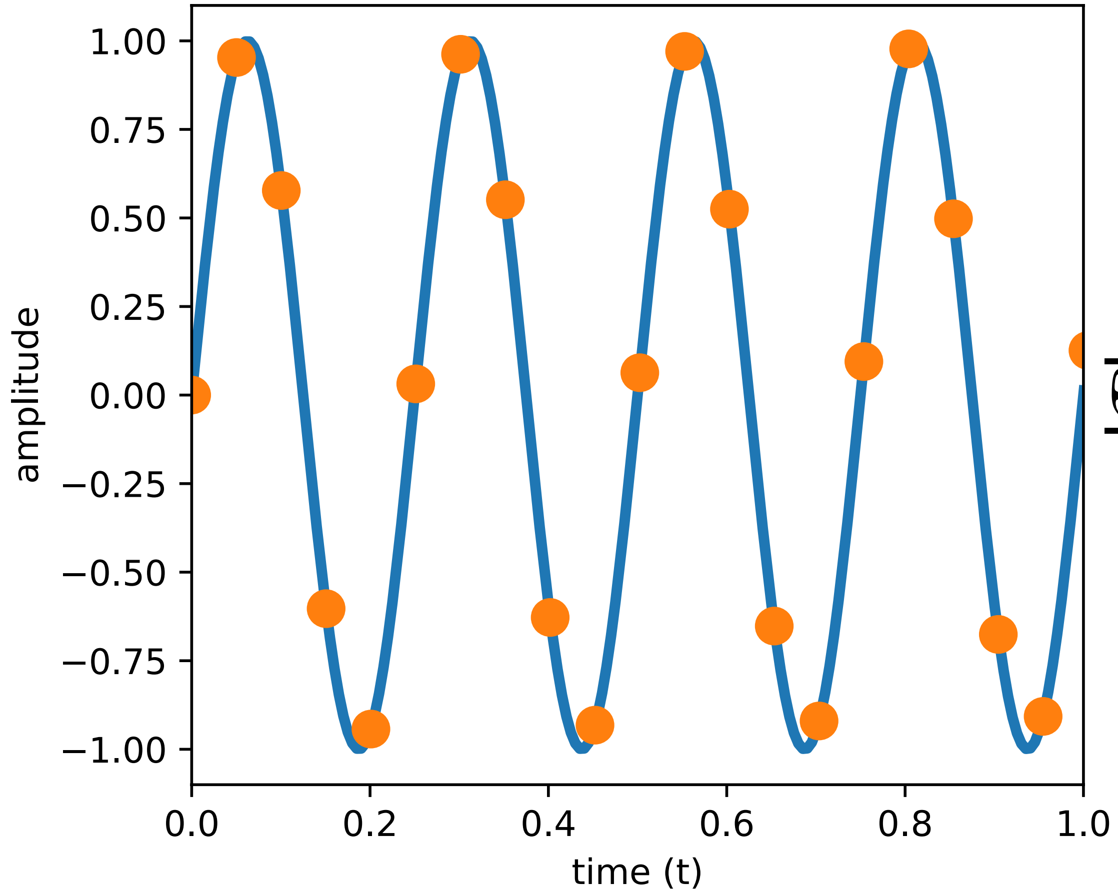


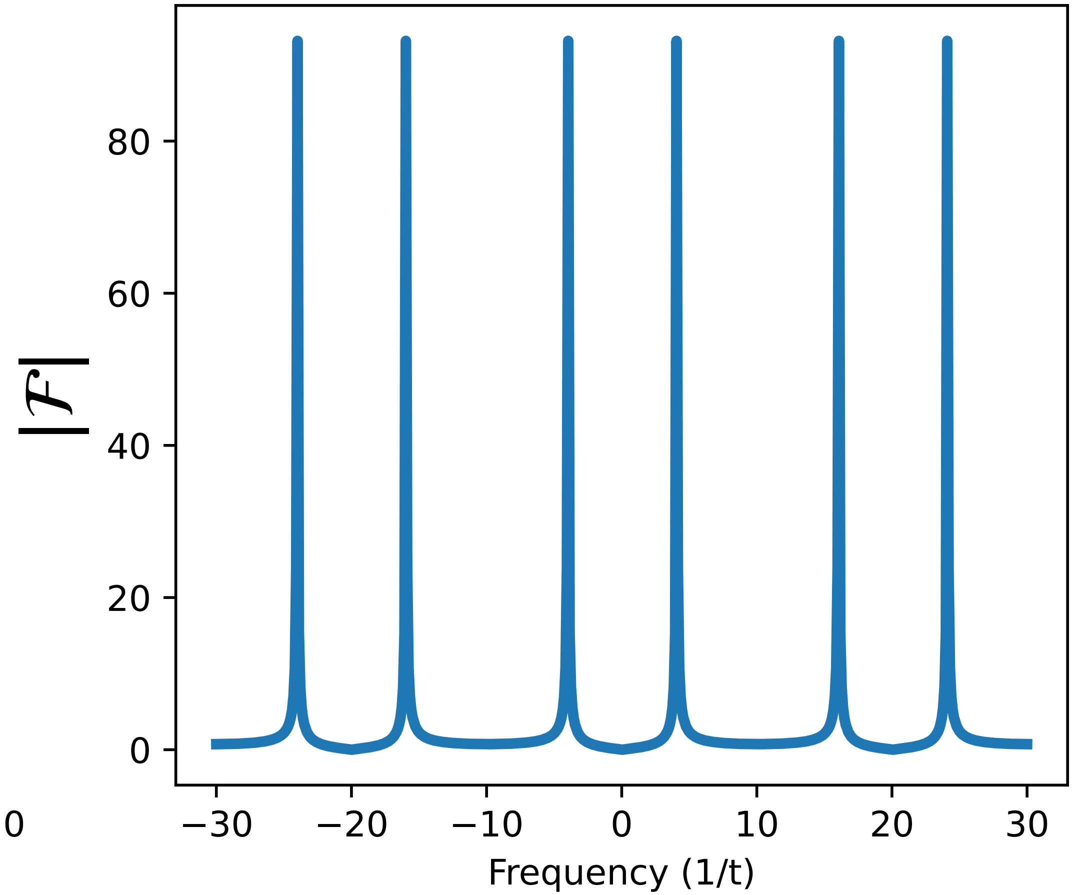
Highest frequency

Sample rate should be twice the highest frequency to avoid aliasing!

Sampling exercise

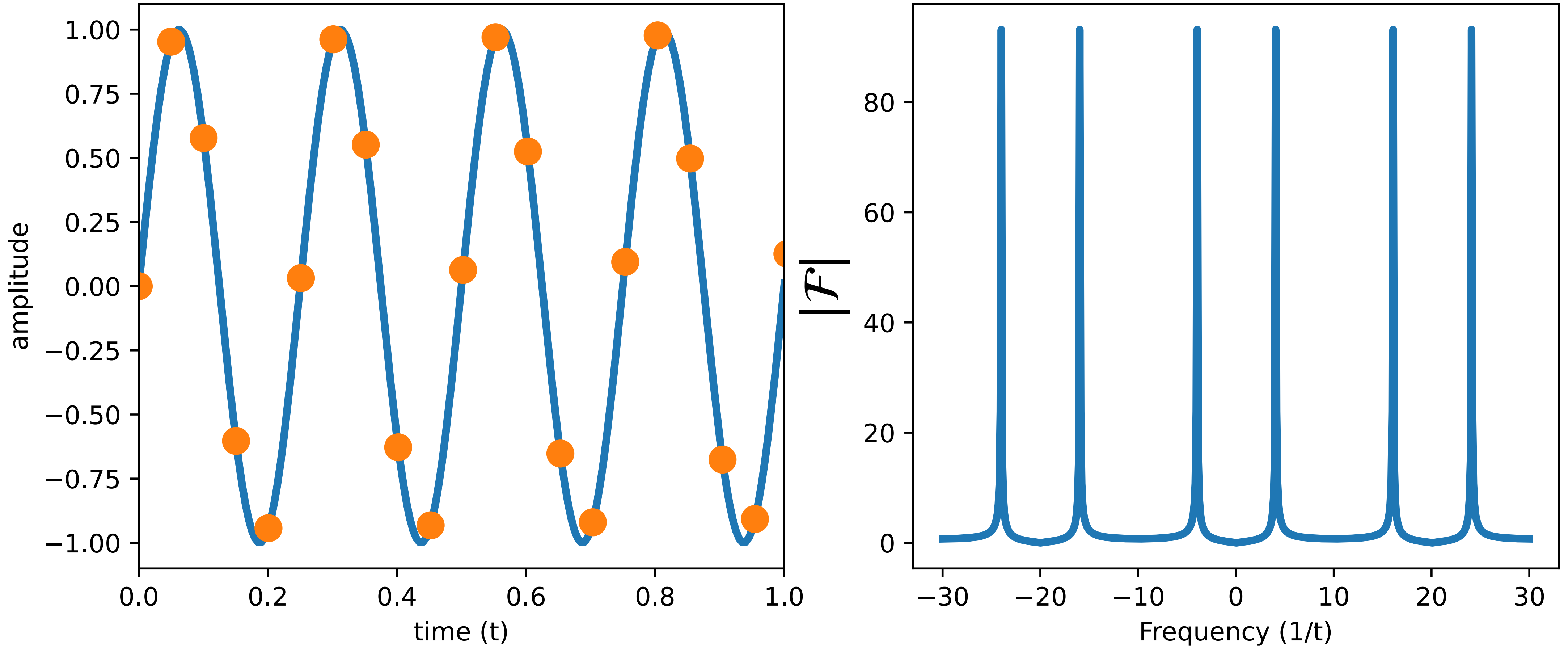
Sample frequency: ?
Signal frequency: ?





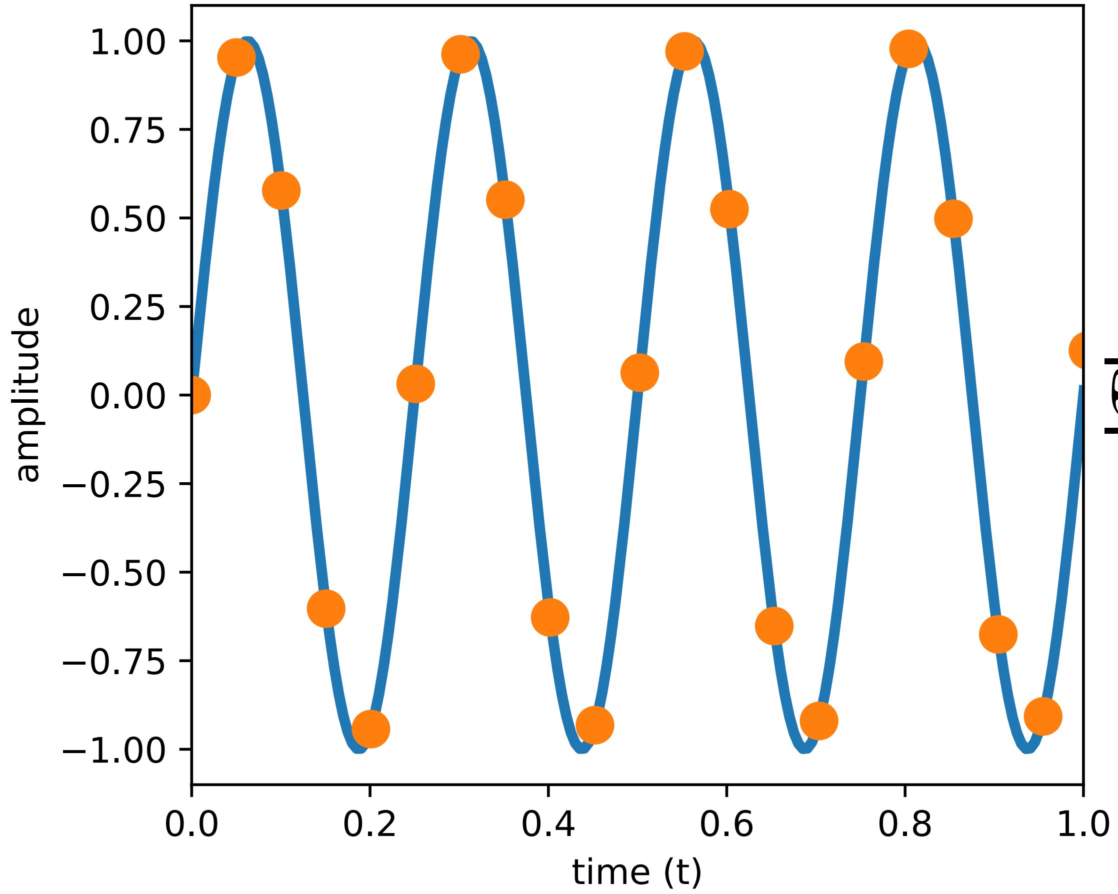
Sampling exercise

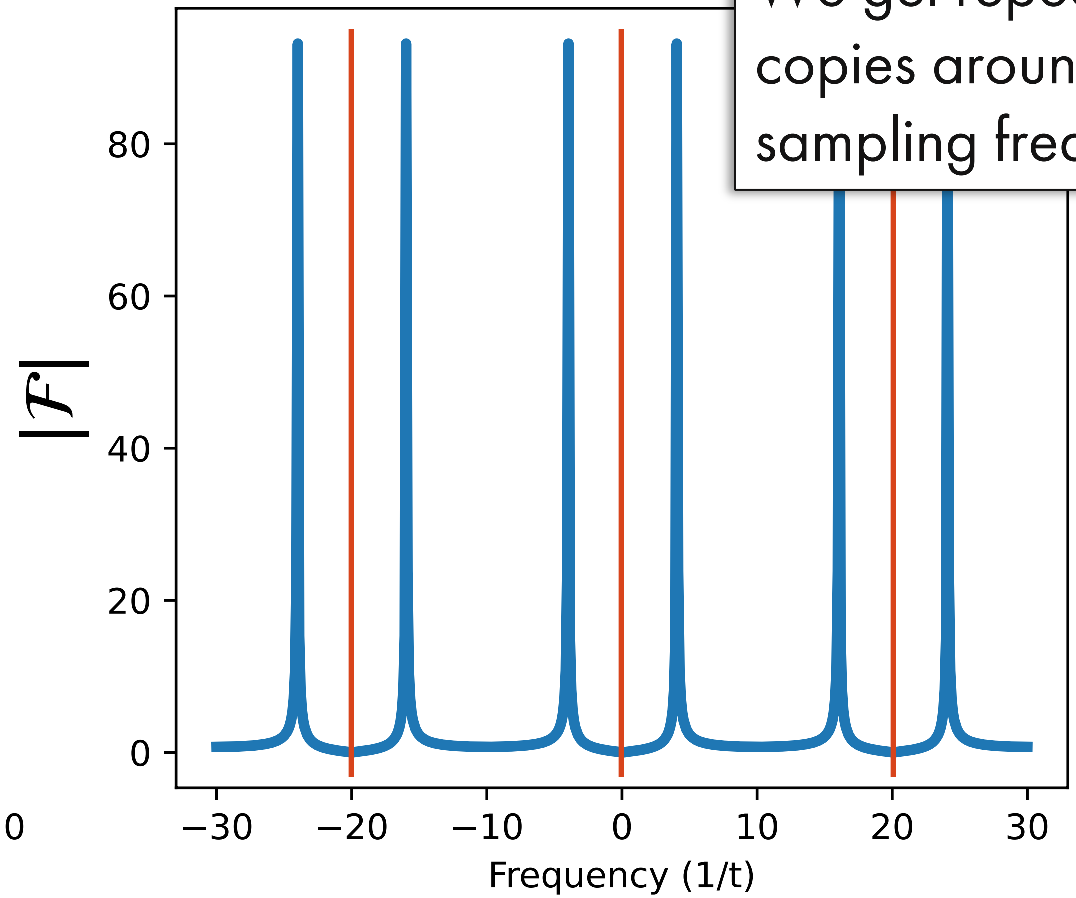
Sample frequency: 20 Hz
Signal: 4 Hz



Sampling exercise

Sample frequency: 20 Hz
Signal: 4 Hz

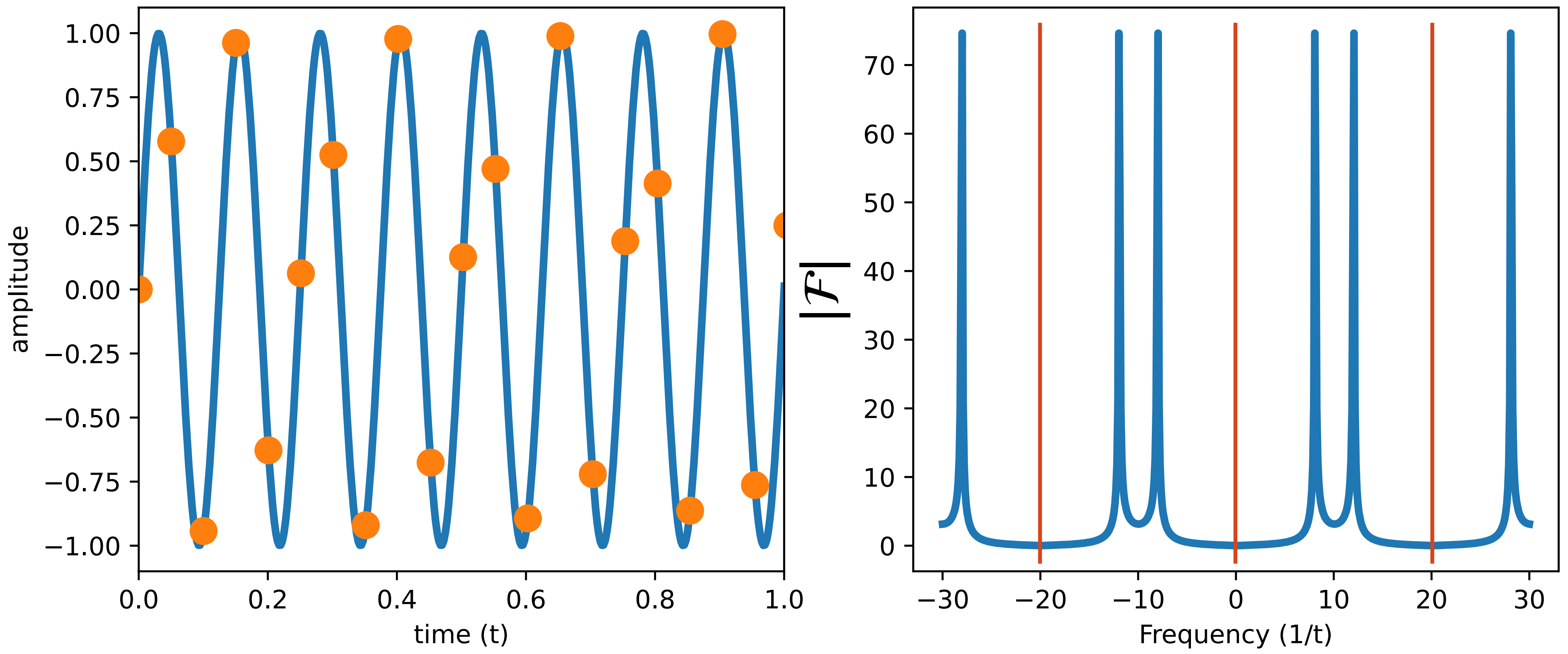




We get repeated copies around the sampling frequency!

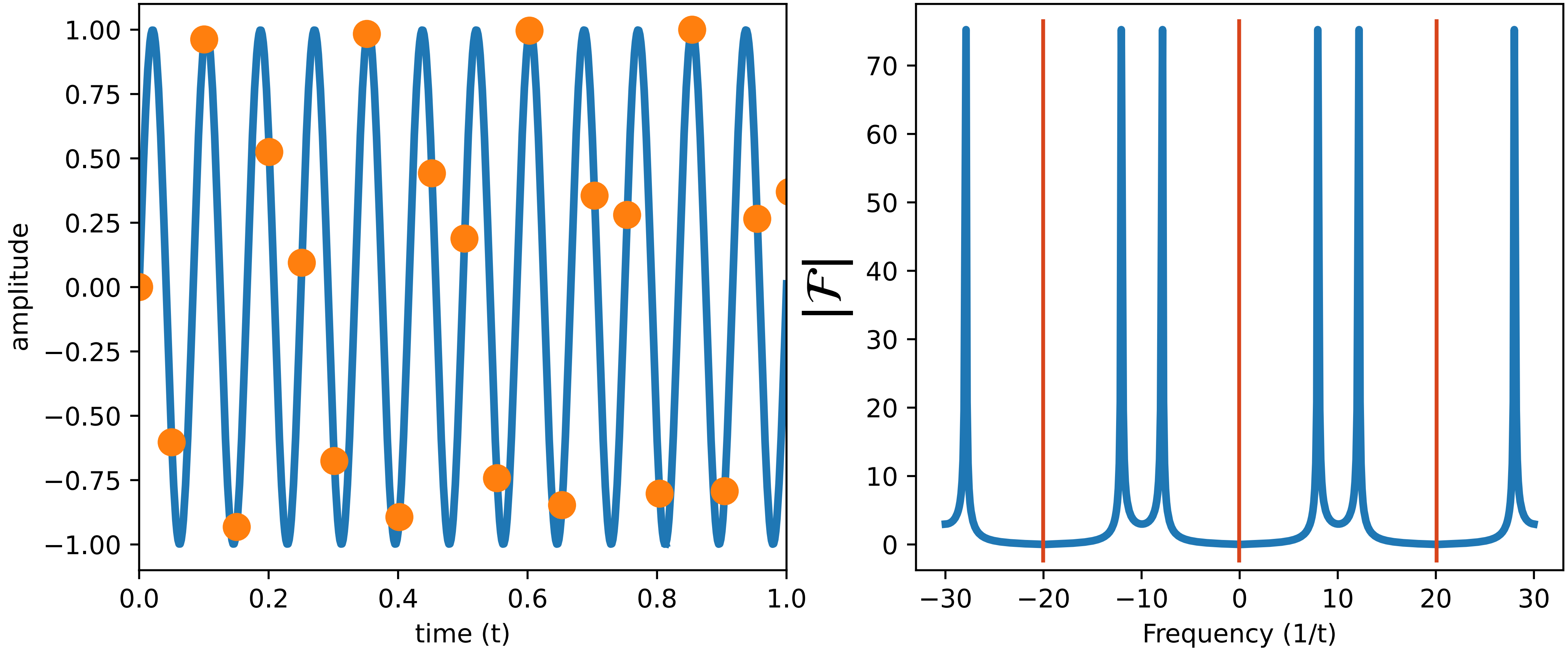
Sampling exercise

Sample frequency: 20 Hz
Signal: 8 Hz



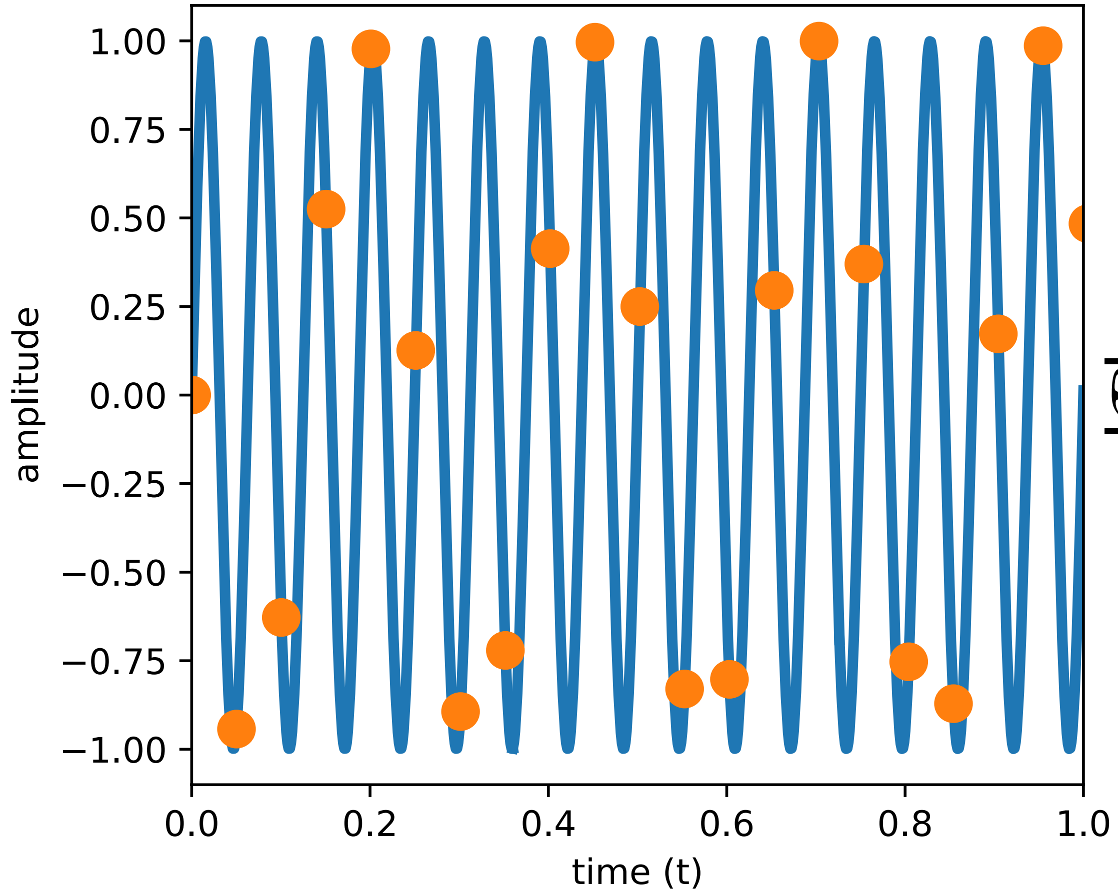
Sampling exercise

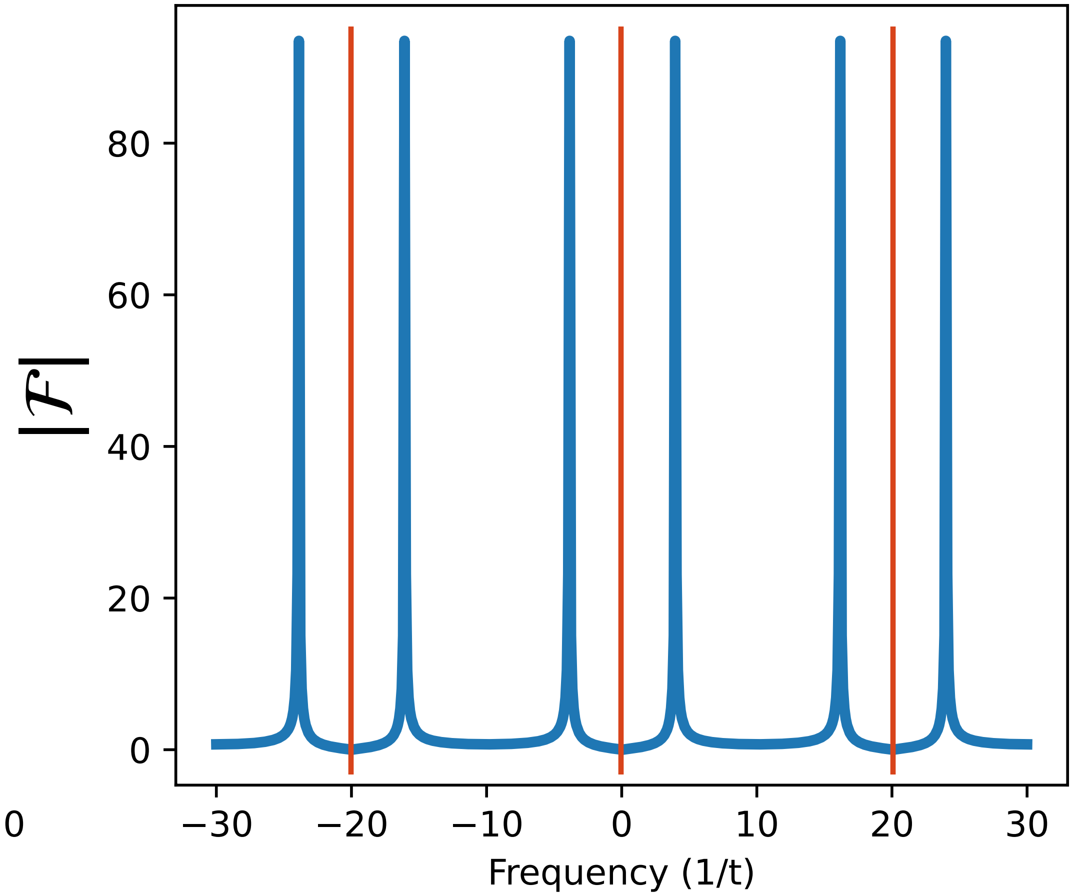
Sample frequency: 20 Hz
Signal: 12 Hz



Sampling exercise

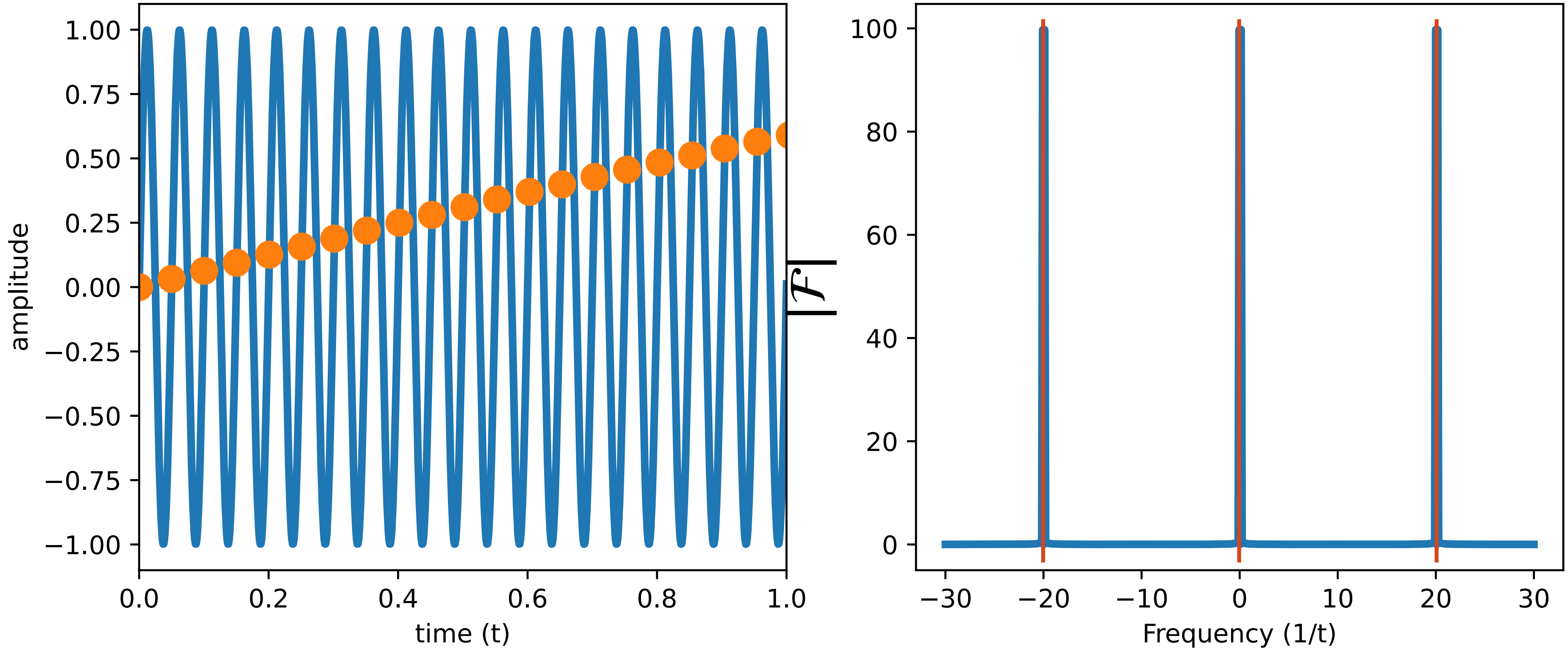
Sample frequency: 20 Hz
Signal: 16 Hz





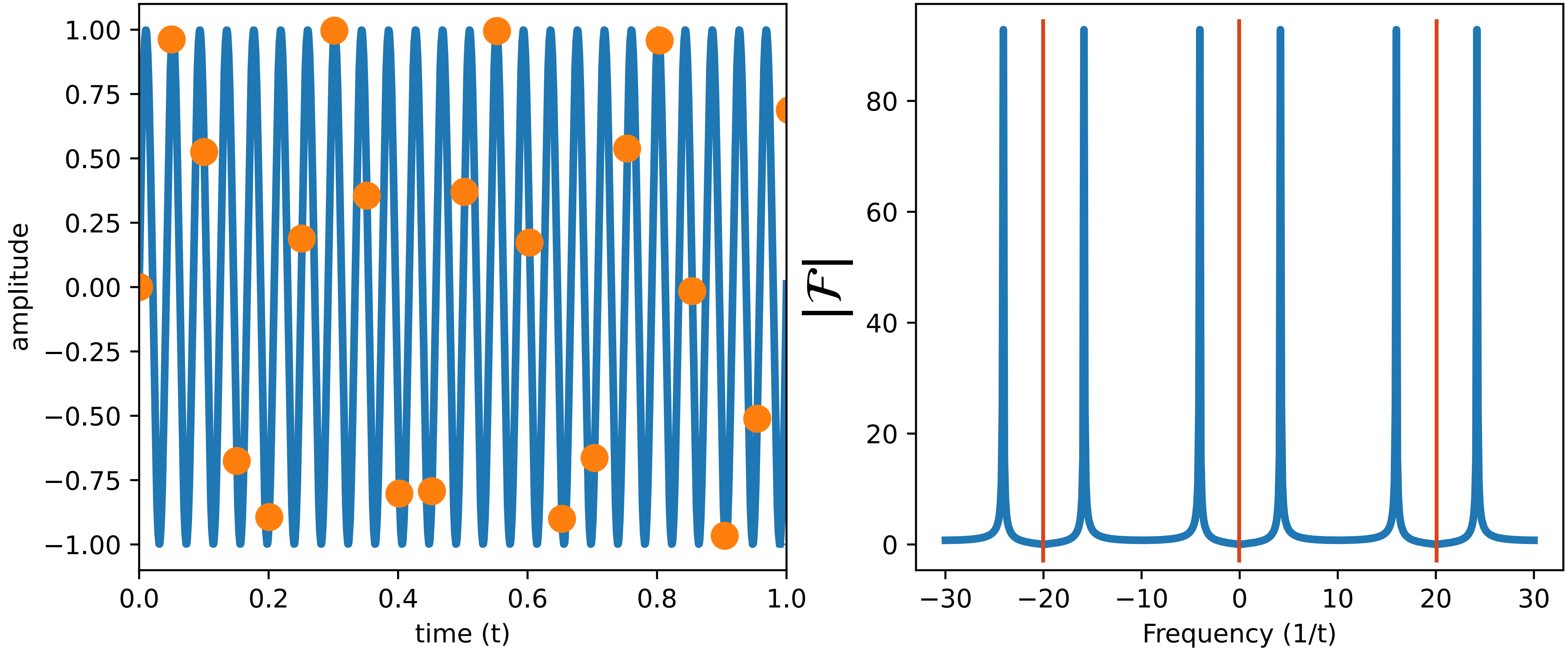
Sampling exercise

Sample frequency: 20 Hz
Signal: 20 Hz



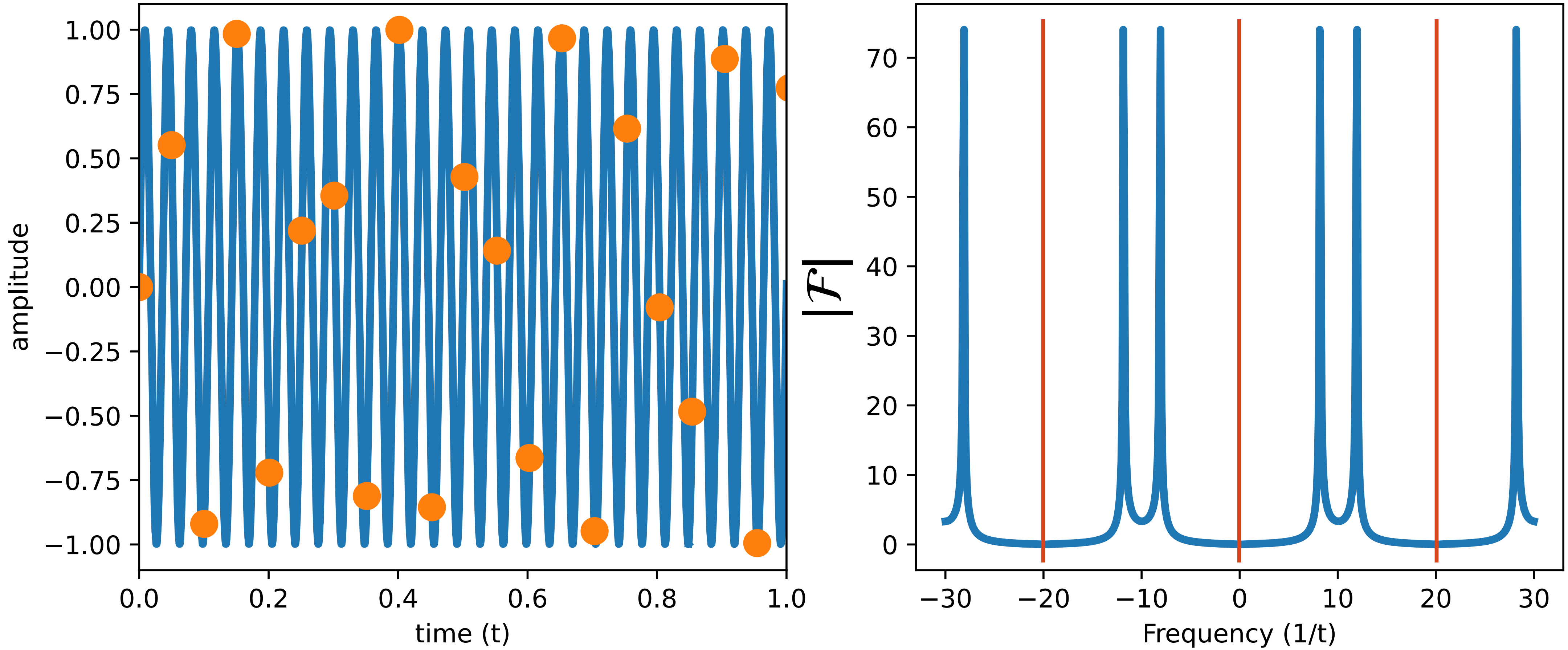
Sampling exercise

Sample frequency: 20 Hz
Signal: 24 Hz



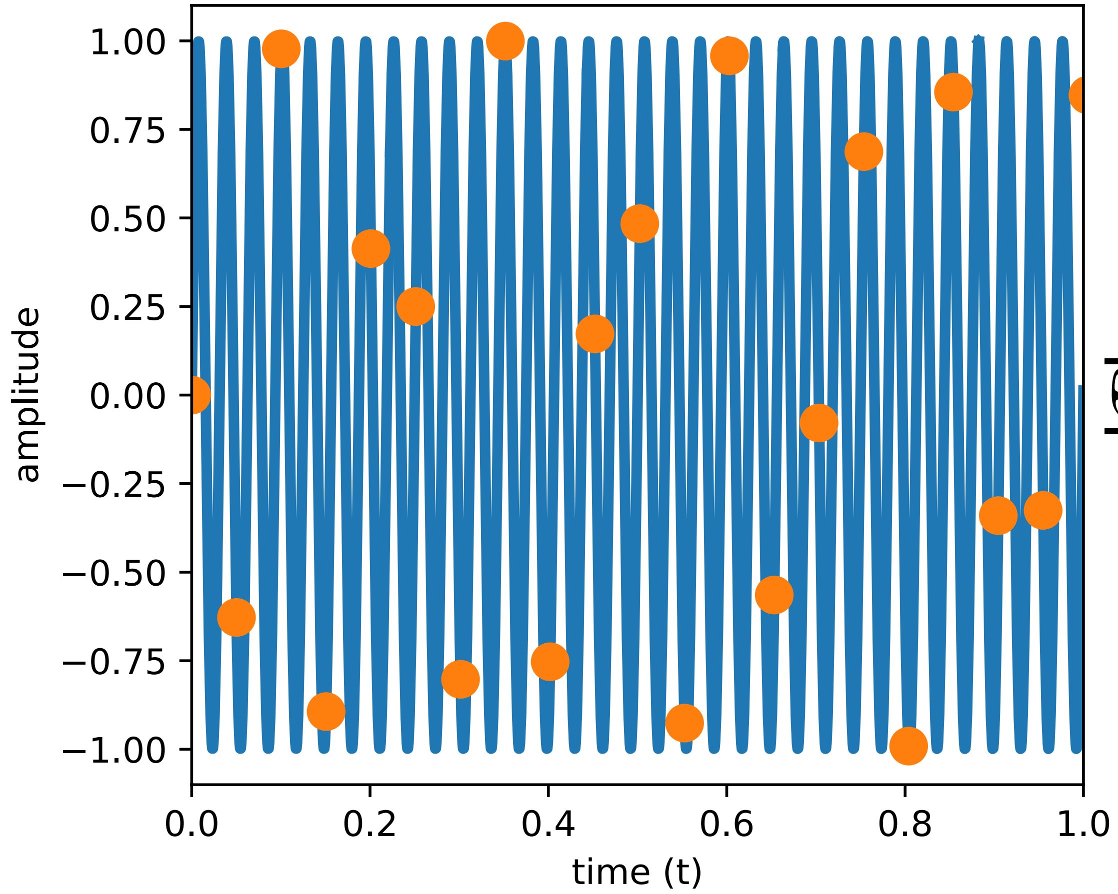
Sampling exercise

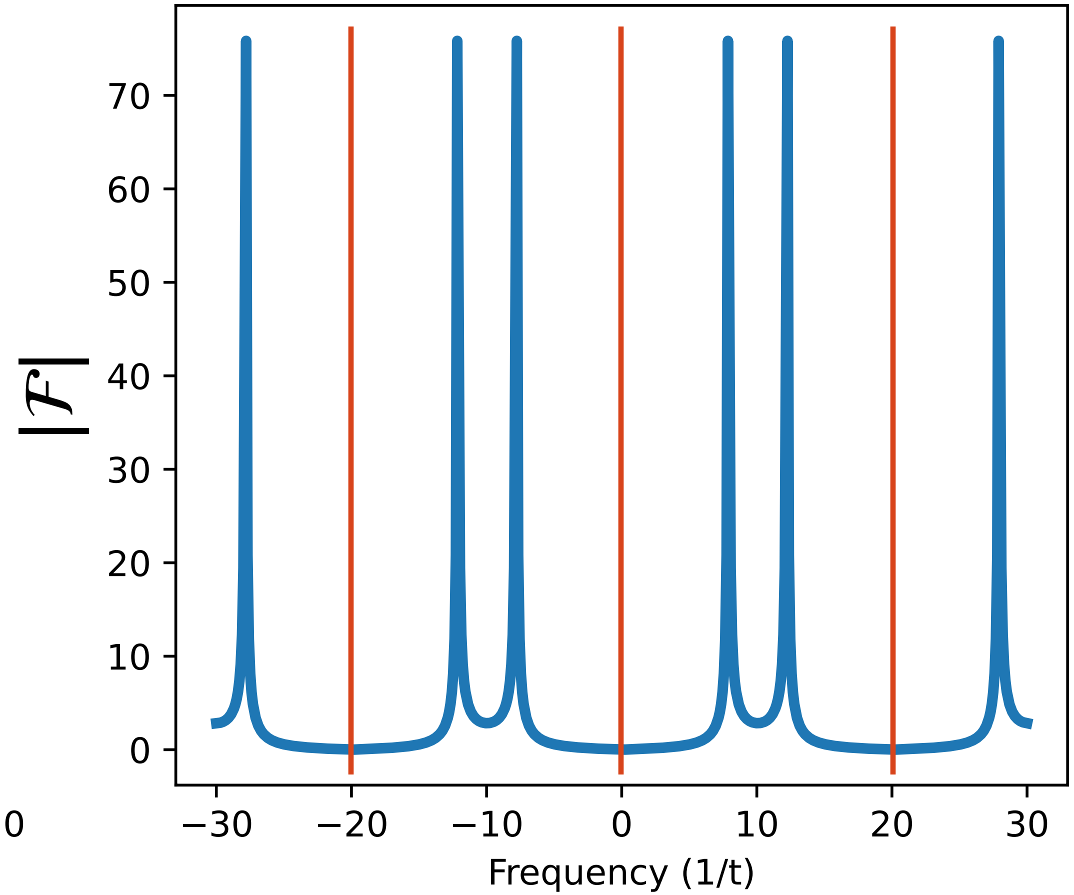
Sample frequency: 20 Hz
Signal: 28 Hz



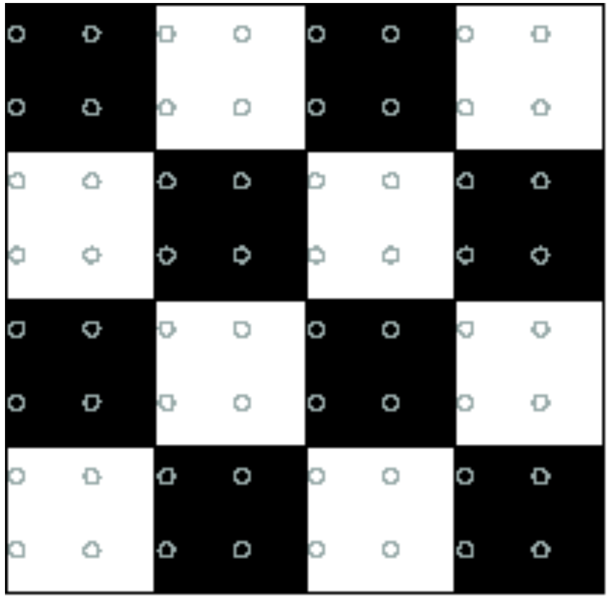
Sampling exercise

Sample frequency: 20 Hz
Signal: 32 Hz





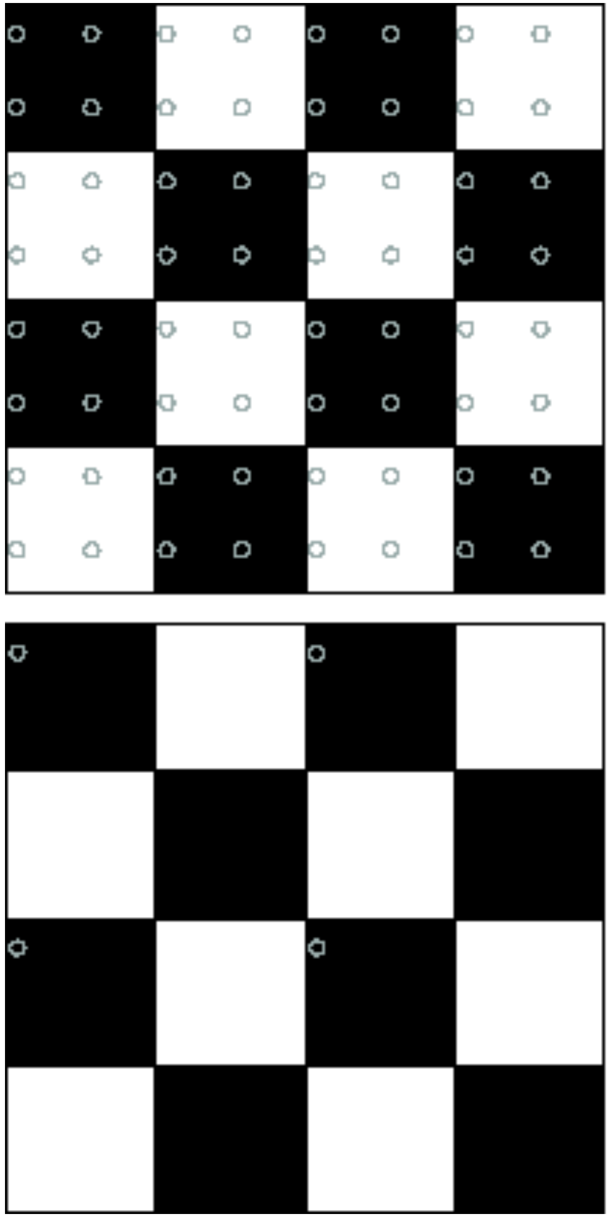
2D example



above or
below
Nyquist?

[Source: N. Snavely]

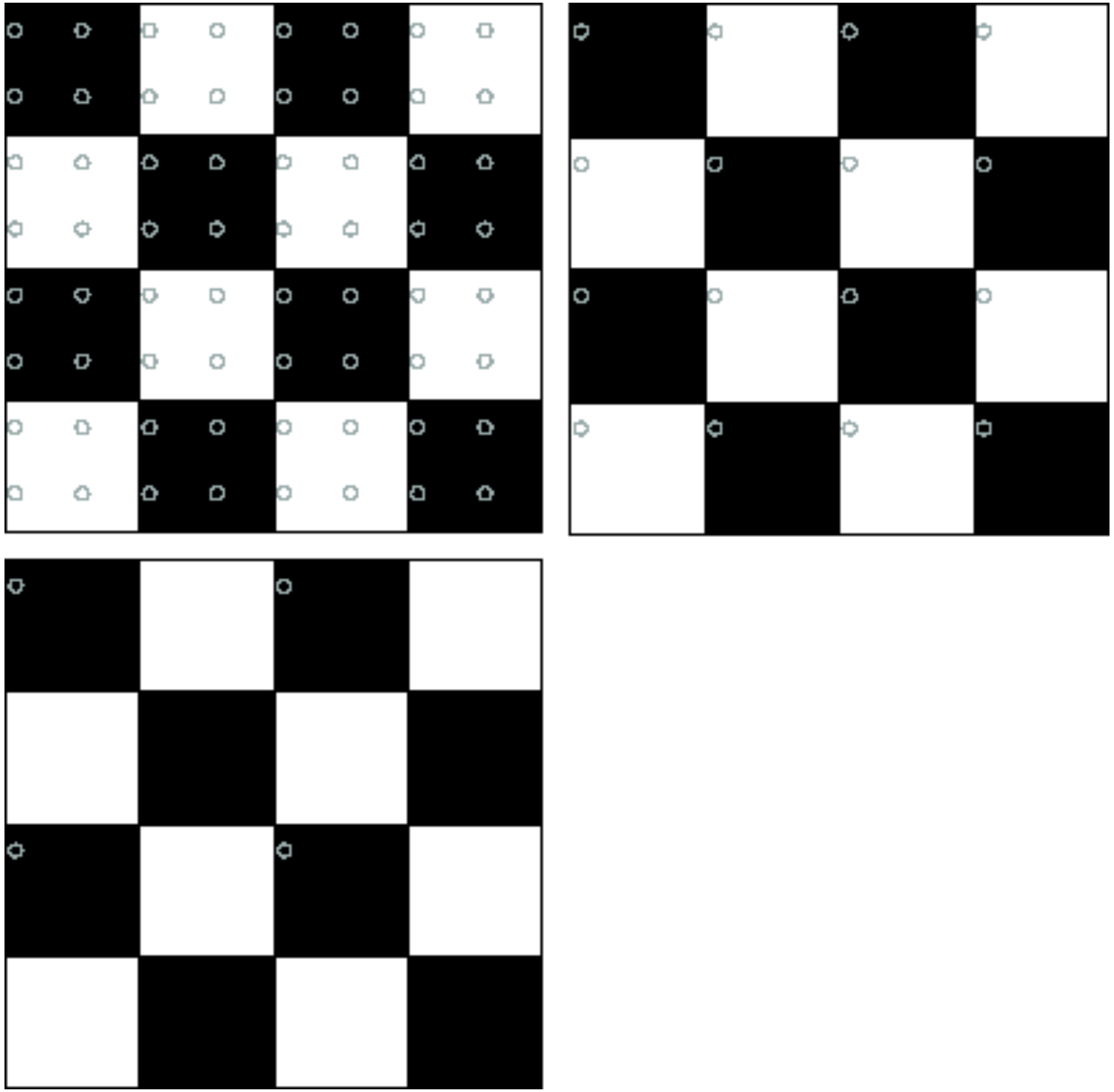
2D example



above or below Nyquist?

[Source: N. Snavely]

2D example

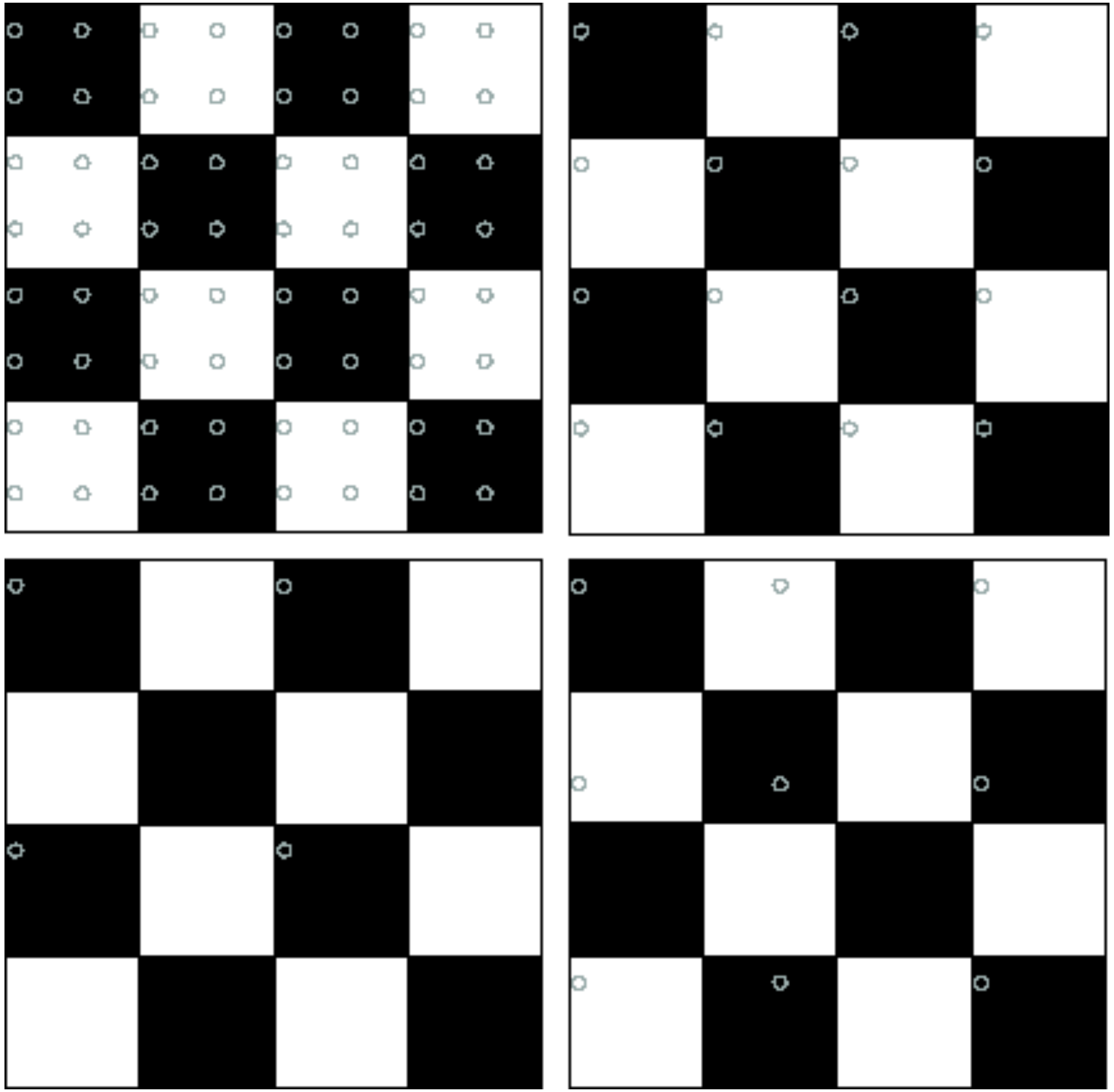


above or
below
Nyquist?

[Source: N. Snavely]

2D example

above or below Nyquist?



[Source: N. Snavely]

Going back to Downsampling ...

- When downsampling by a factor of two, the original image has frequencies that are too high
- High frequencies are caused by sharp edges
- How can we fix this?

[Adopted from: R. Urtasun]

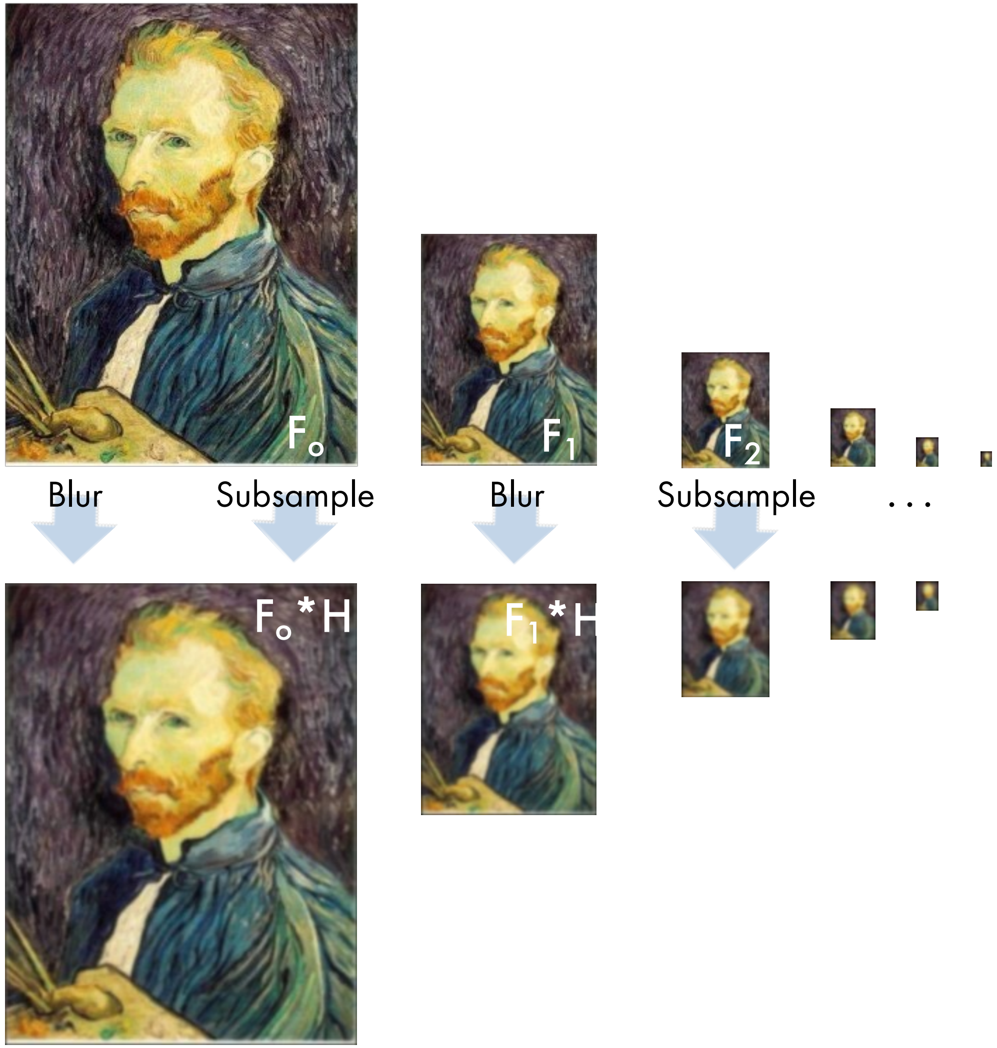
Going back to Downsampling ...

- When downsampling by a factor of two, the original image has frequencies that are too high
- High frequencies are caused by sharp edges
- How can we fix this?

[Adopted from: R. Urtasun]

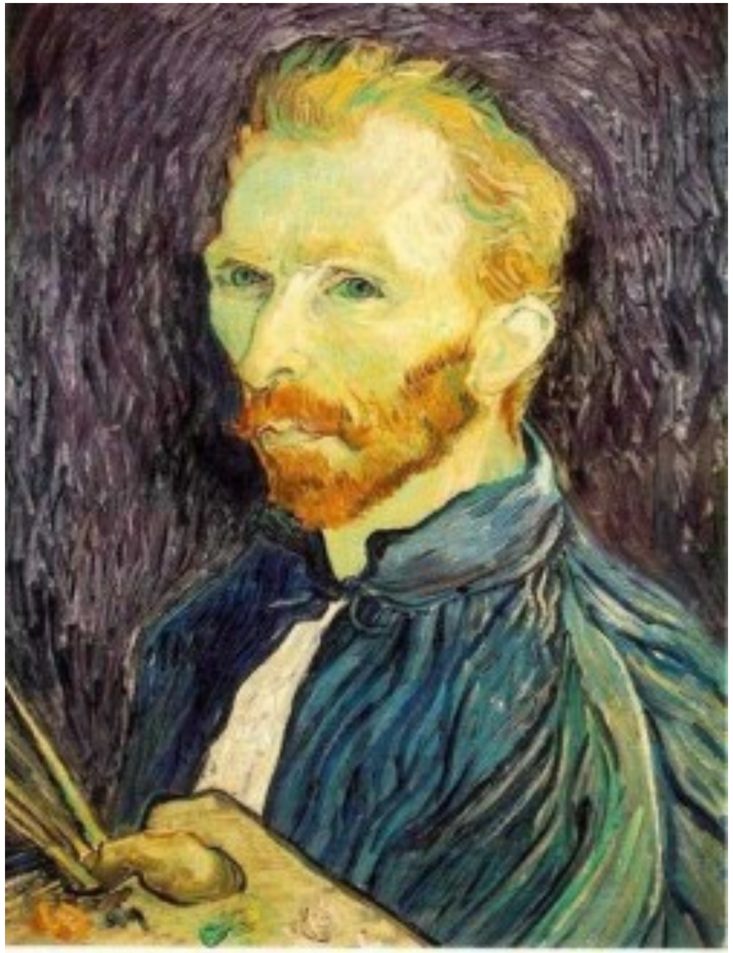
Gaussian pre-filtering

- Solution: Filter out the higher frequency data. Blur the image via Gaussian, then subsample. Very simple!



[Source: N. Snavely]

Subsampling with Gaussian pre-filtering



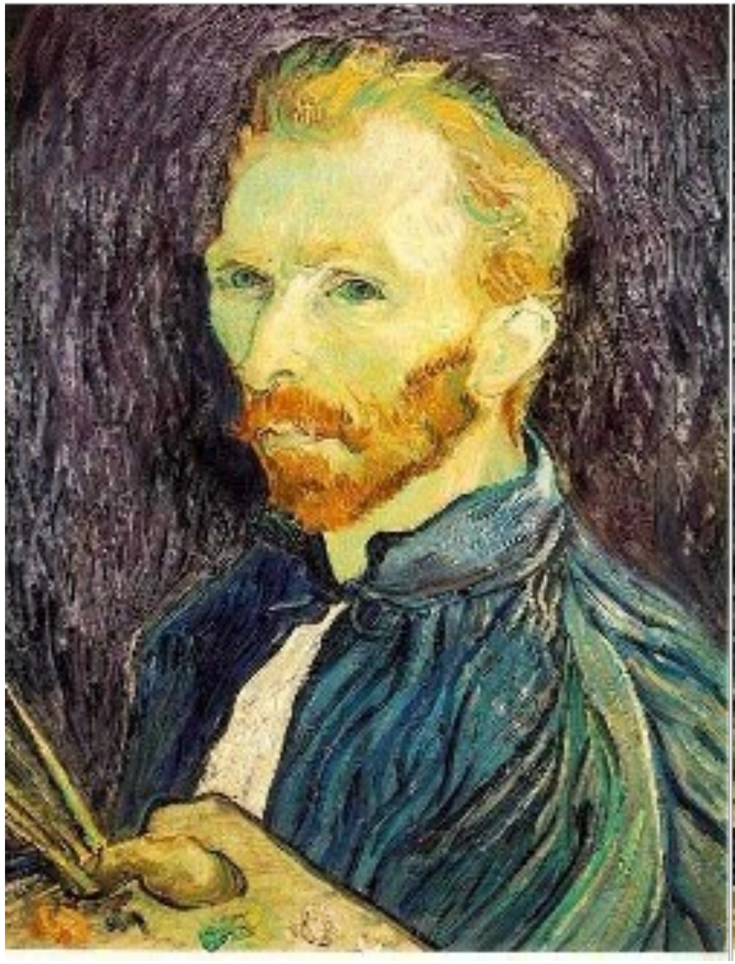
Gaussian 1/2

G 1/4

G 1/8

[Source: S. Seitz]

Subsampling with Gaussian pre-filtering



1/2

¼ (2x Zoom)

1/8 (4x Zoom)

[Source: S. Seitz]

Where is the Rectangle?

- My image

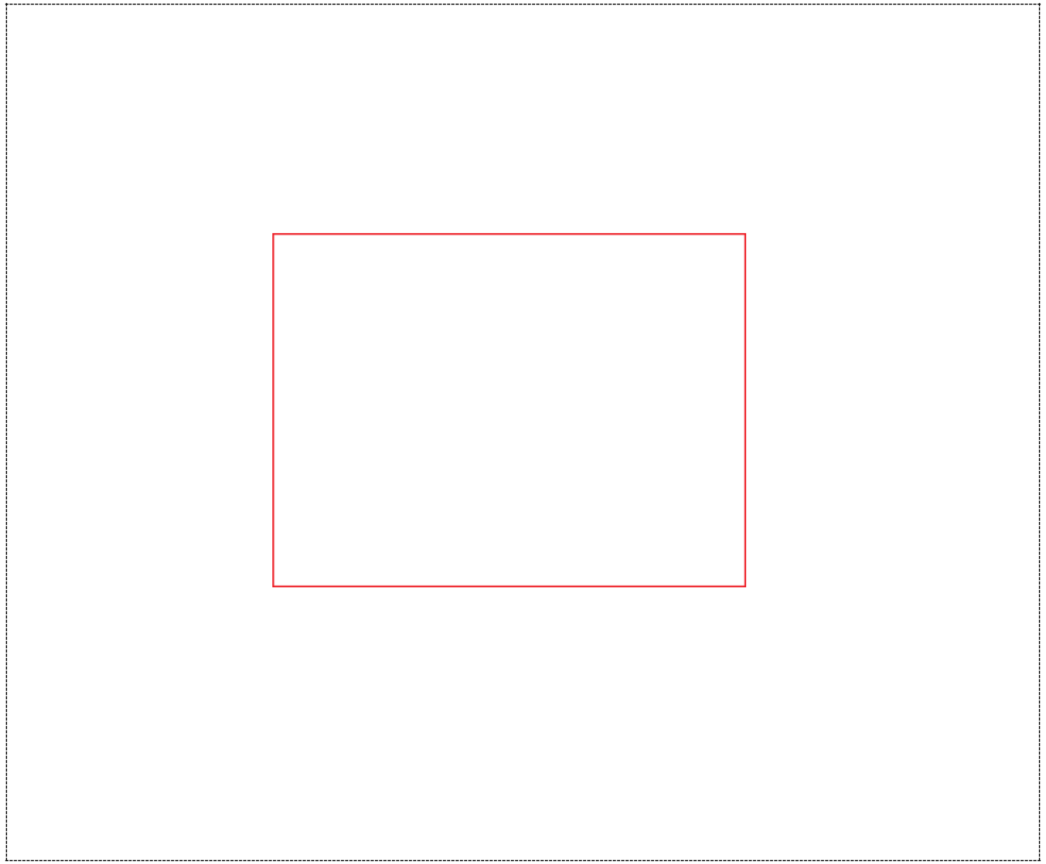


Figure: Dashed line denotes the border of the image (it's not part of the image)

Where is the Rectangle?

- My image
- Let's blur

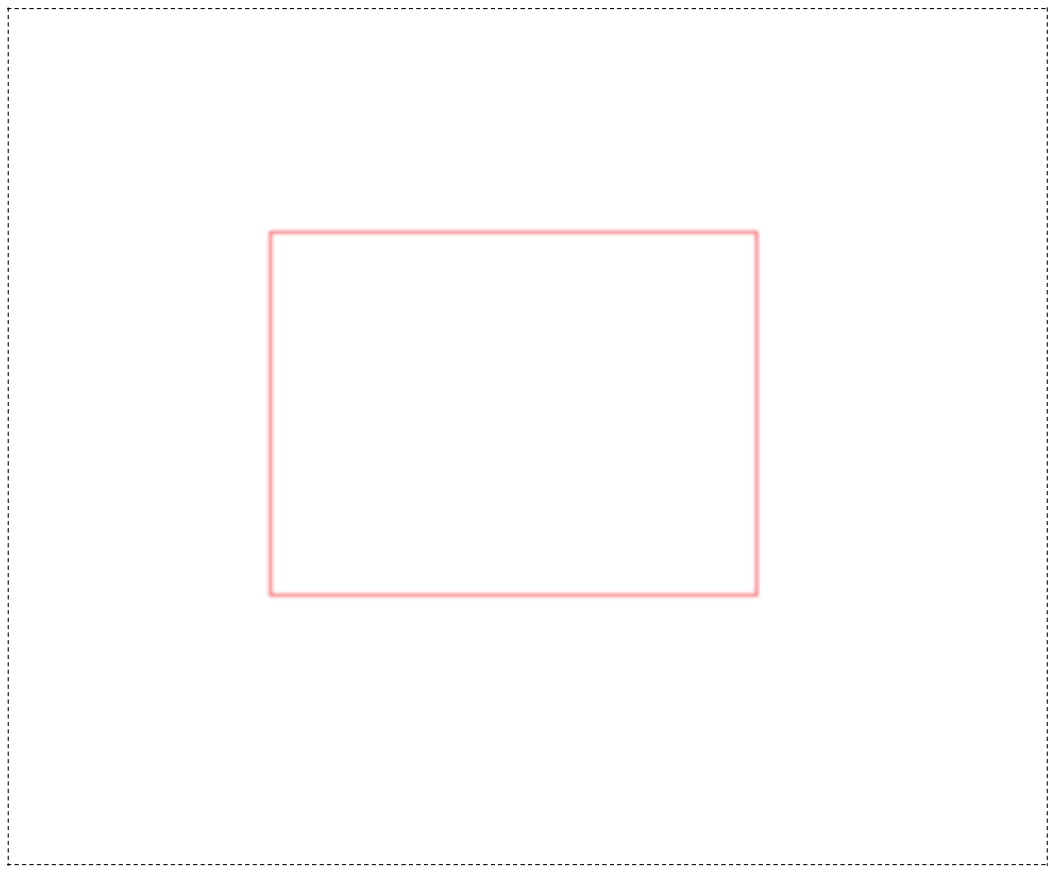


Figure: Dashed line denotes the border of the image (it's not part of the image)

Where is the Rectangle?

- My image
- Let's blur
- And now take every other row and column

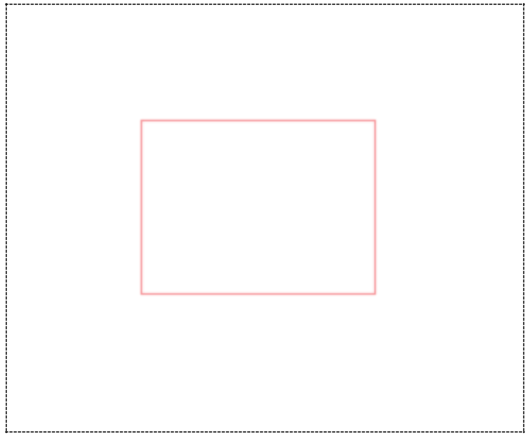


Figure: Dashed line denotes the border of the image (it's not part of the image)

Where is the Chicken?

- My image

Where is the Chicken?

- My image
- Let's blur

Where is the Chicken?

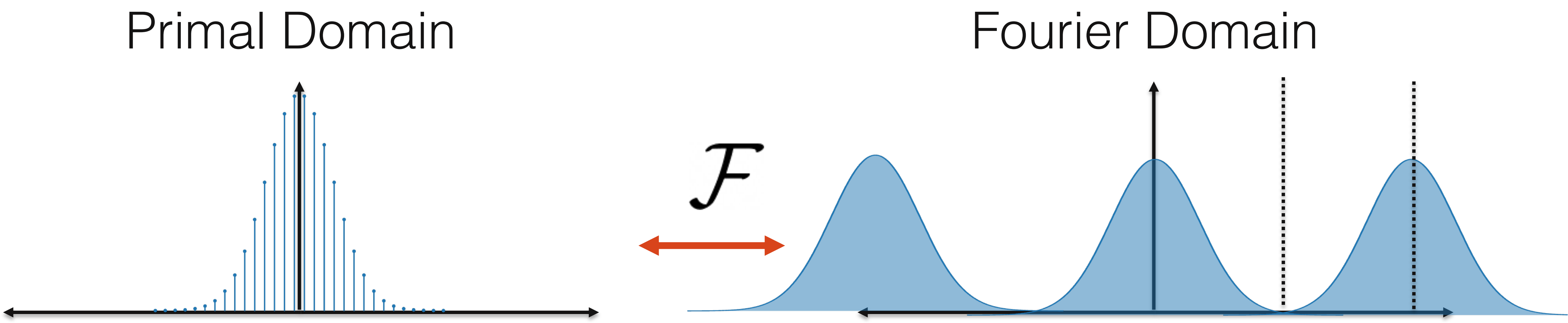
- My image
- Let's blur
- And now take every other column

Why does this work?

- What does blurring do in the frequency domain?

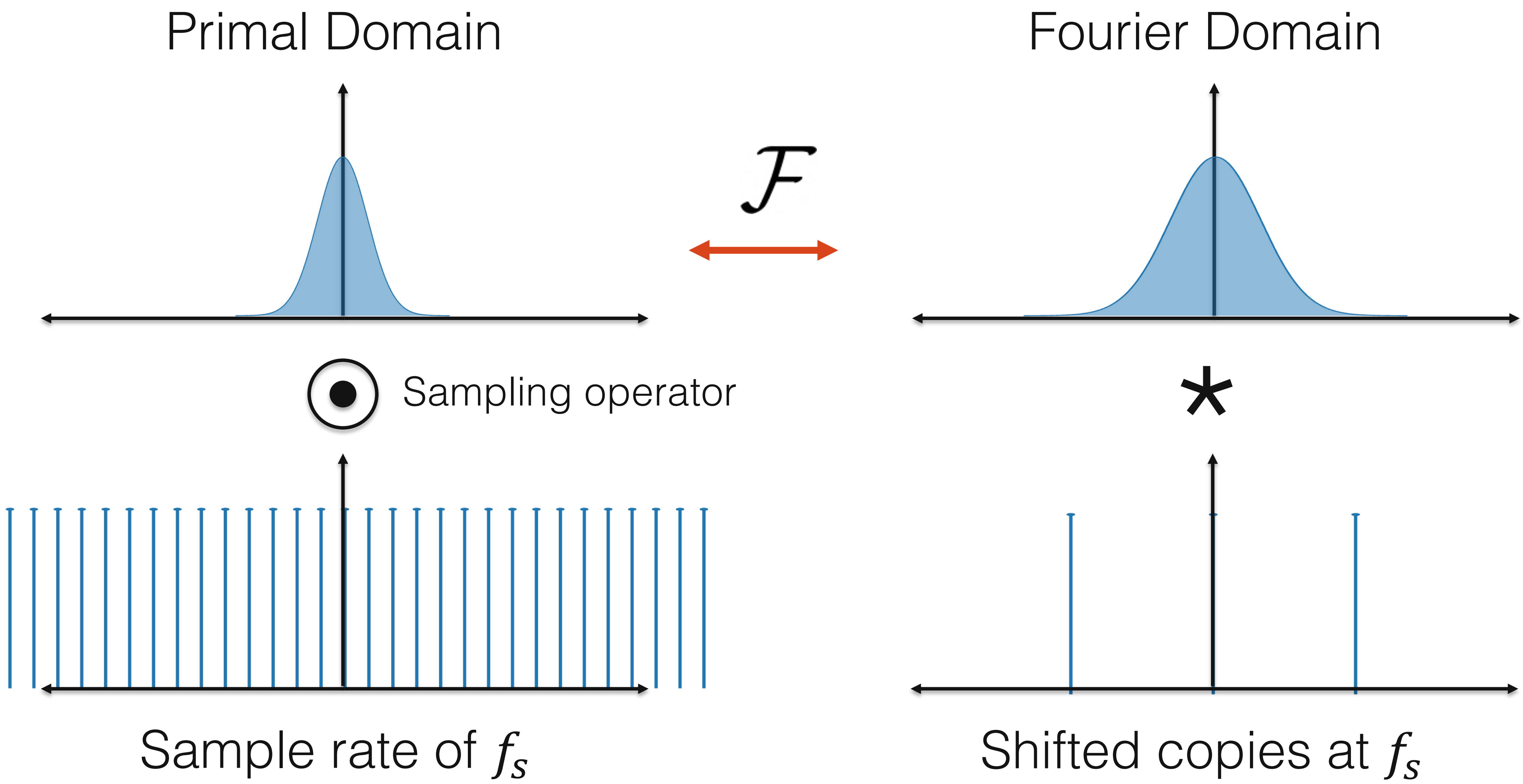
- How does that fix the aliasing problem?

Sampling



What happens if we subsample in the primal domain?

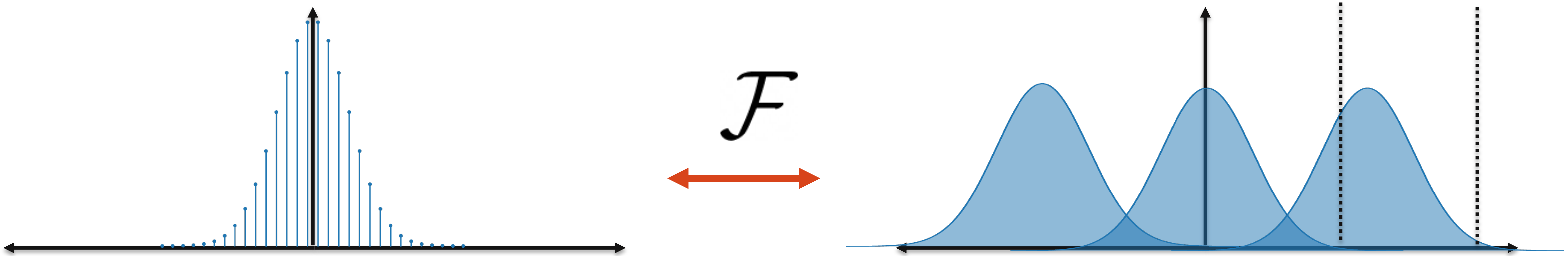
Sampling


Primal Domain
Fourier Domain
$\mathcal{F}$
Sampling operator
*
Sample rate of f_s
Shifted copies at f_s

Sampling

Primal Domain

Fourier Domain

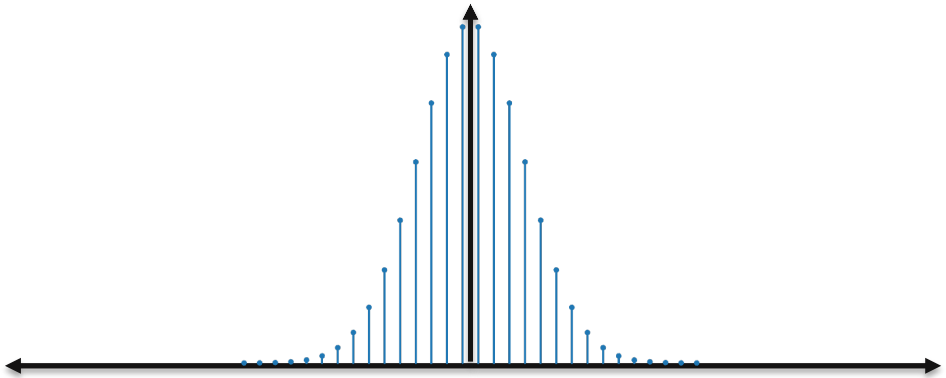


What happens if we subsample in the primal domain?

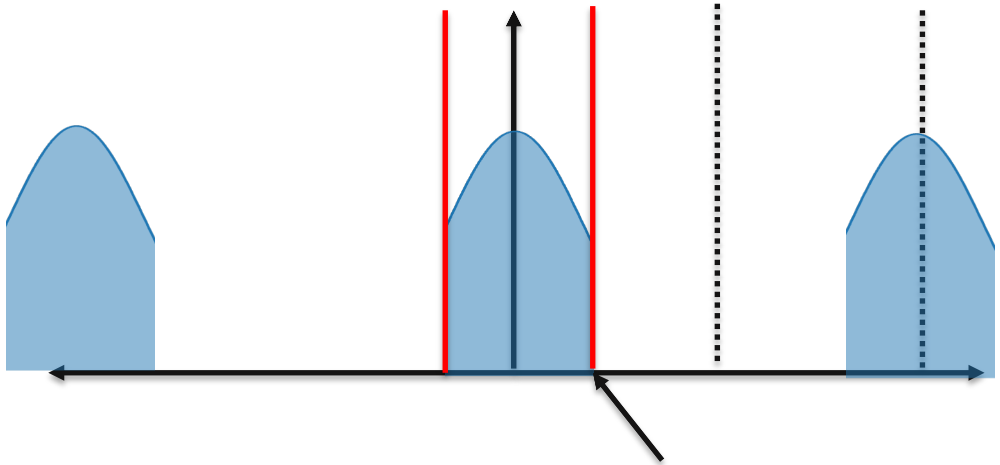
- **Shifted copies start to overlap! High frequencies *alias* into lower frequencies**

Sampling

Primal Domain



Fourier Domain



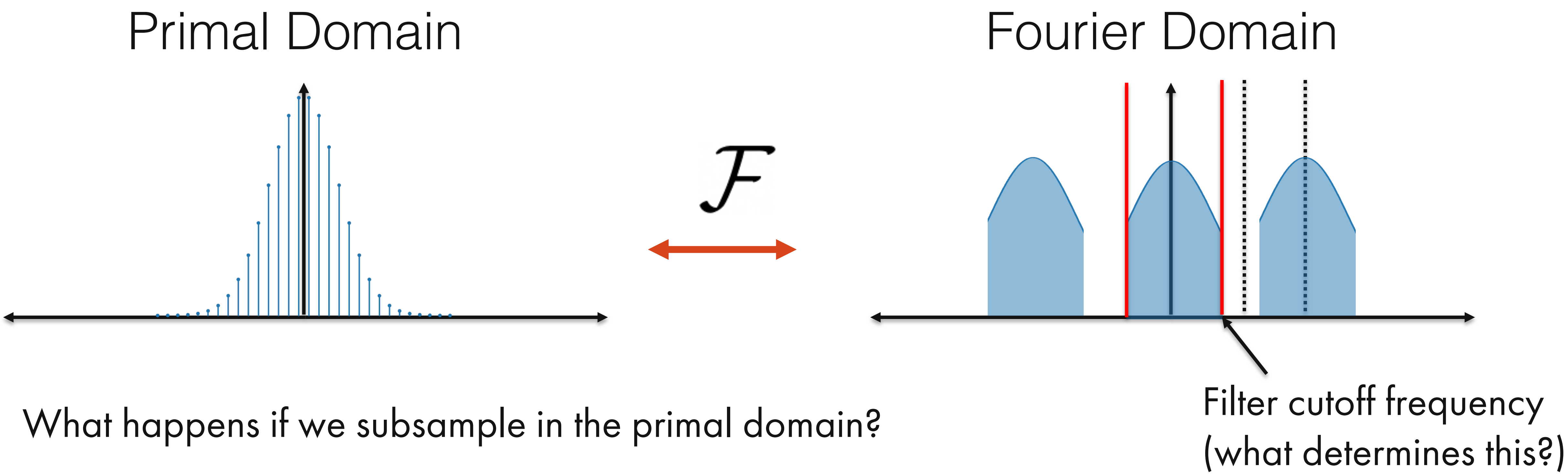
Filter cutoff frequency (what determines this?)

What happens if we subsample in the primal domain?

- **Shifted copies start to overlap! High frequencies *alias* into lower frequencies**

- To solve: first low-pass filter

Sampling



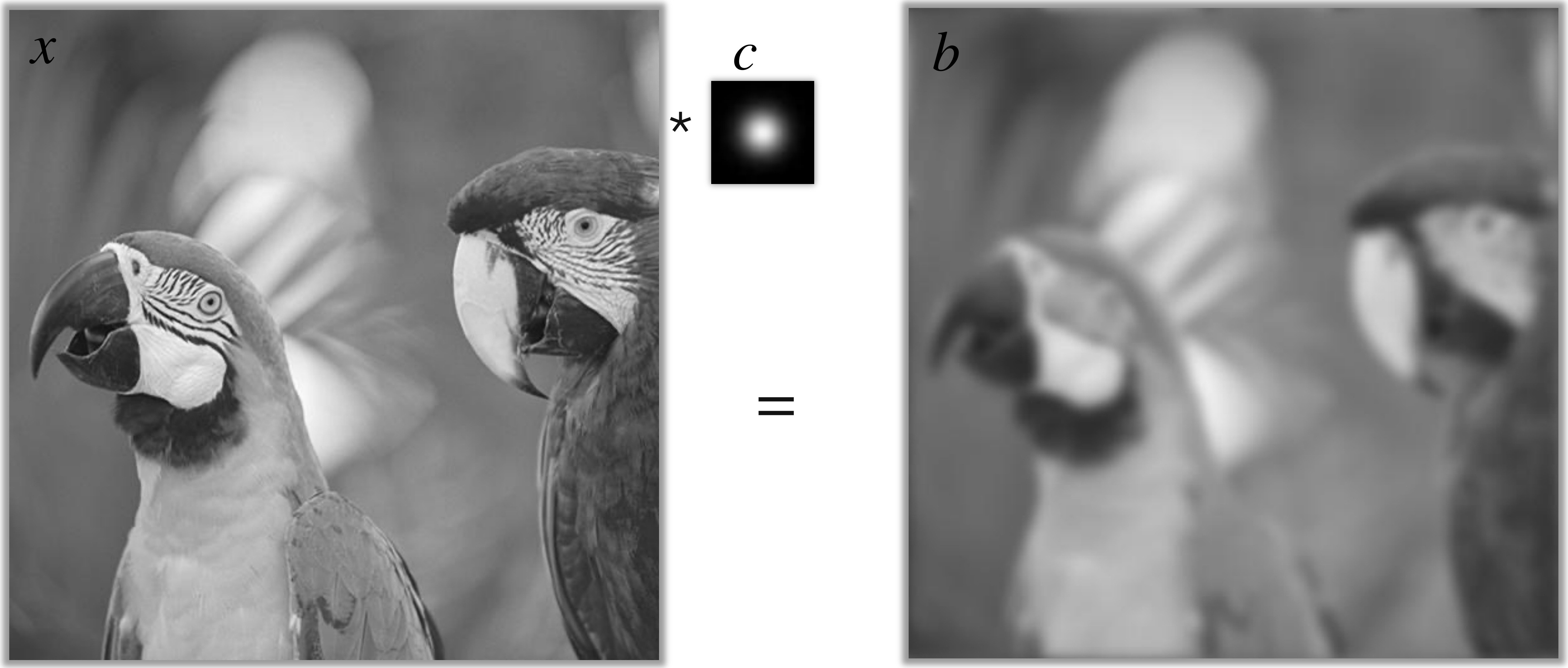
What happens if we subsample in the primal domain?

- **Shifted copies start to overlap! High frequencies *alias* into lower frequencies**

- To solve: first low-pass filter
- Then no aliasing after downsampling!

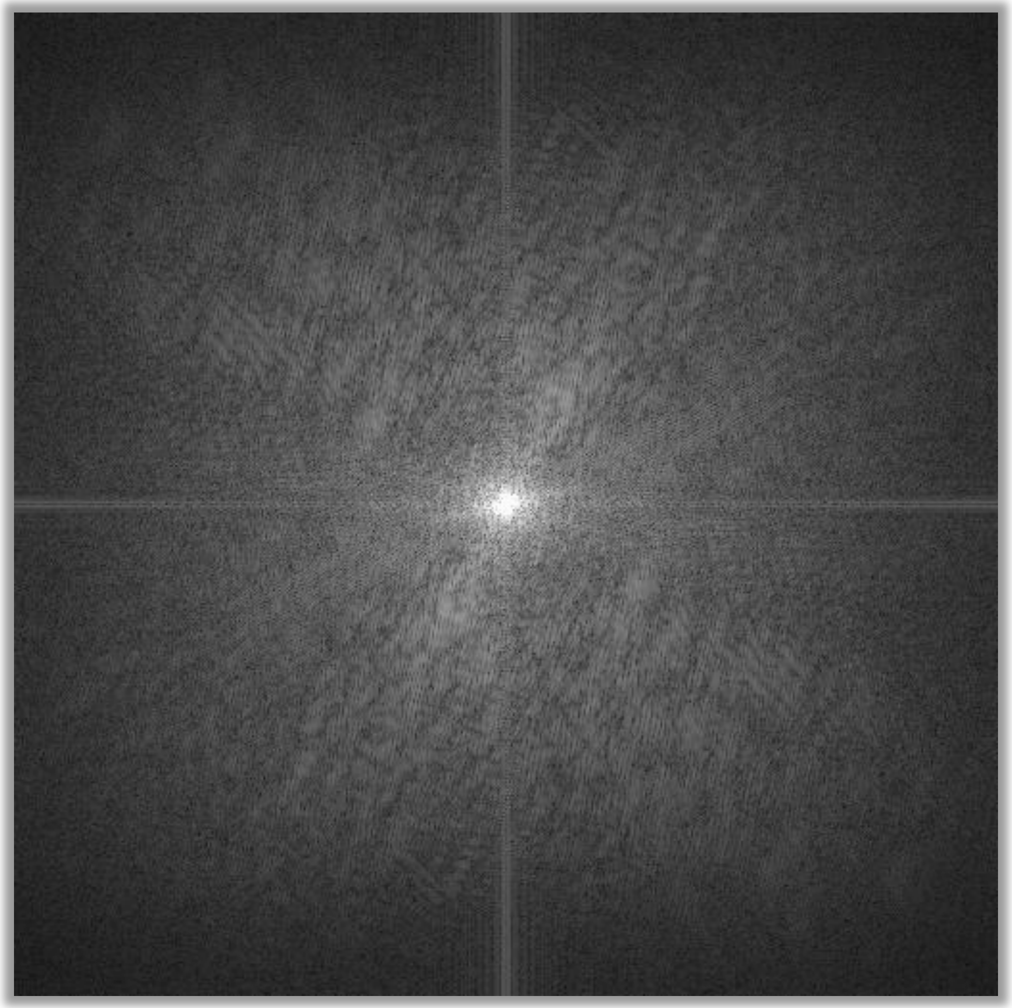
Filtering – Low-pass Filter

- low-pass filter: convolution in primal domain $b = x * c$
- convolution kernel c is also known as point spread function (PSF)

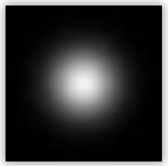


Filtering – Low-pass Filter

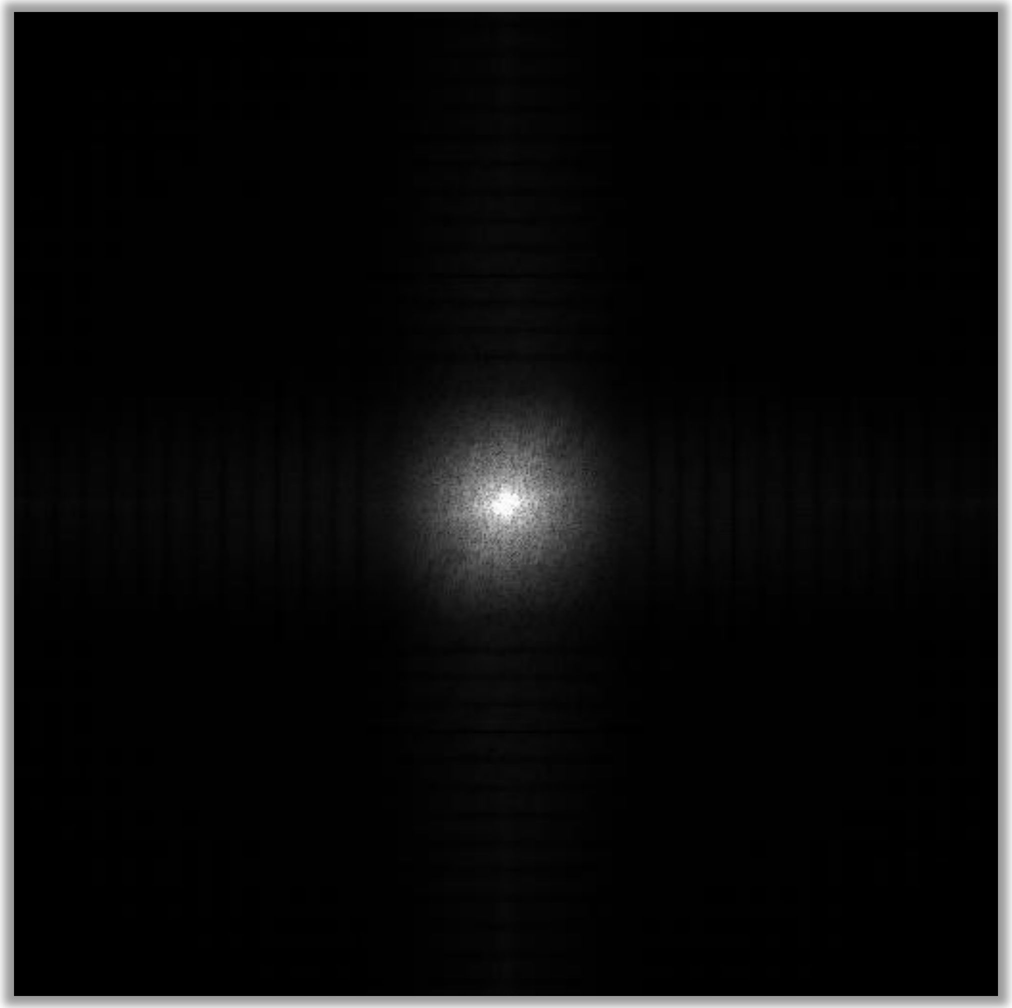
- low-pass filter: multiplication in frequency domain $F\{b\} = F\{x\} \cdot F\{c\}$



$\cdot$

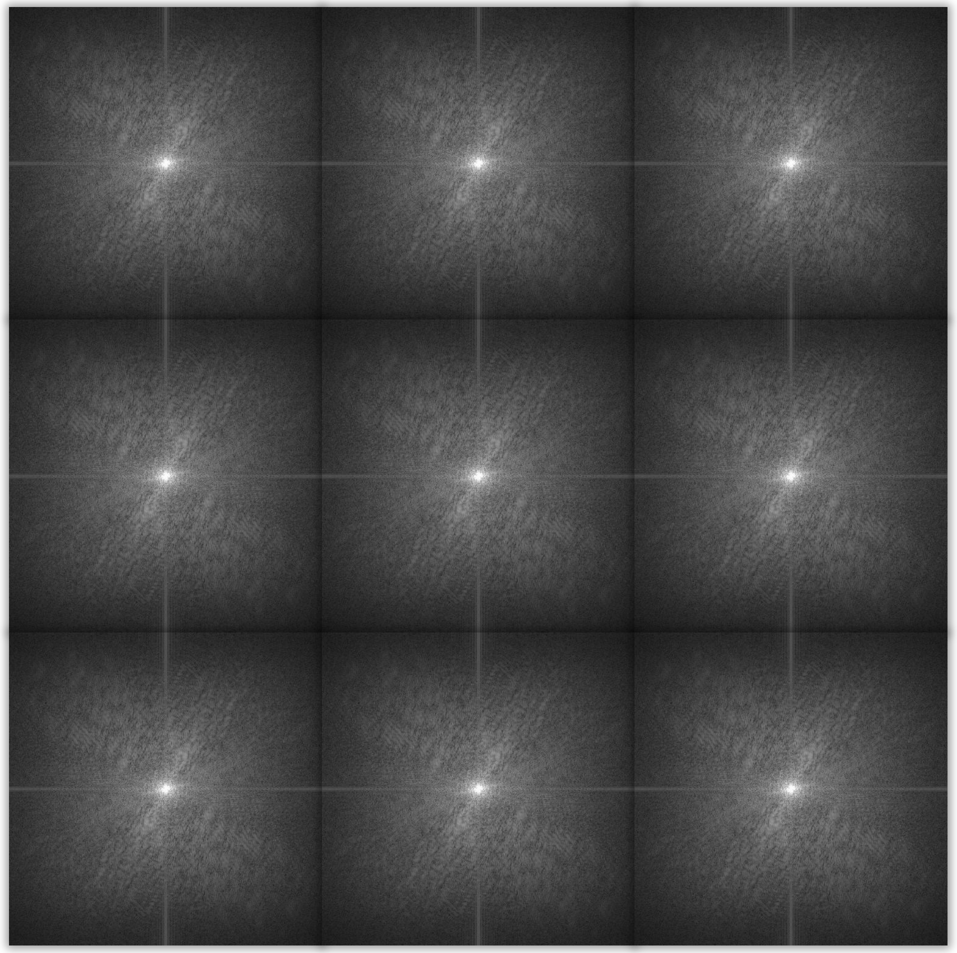


$=$



What's the picture in 2D?

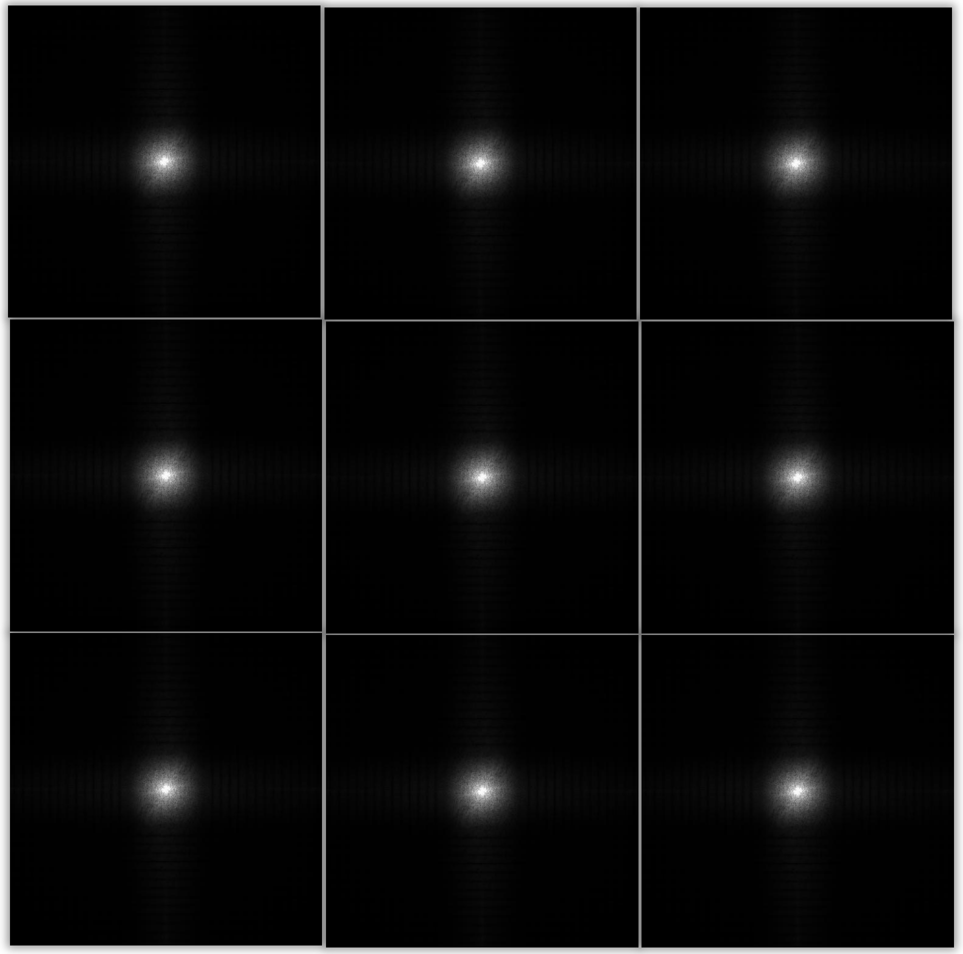
Periodic spectral copies in the frequency domain



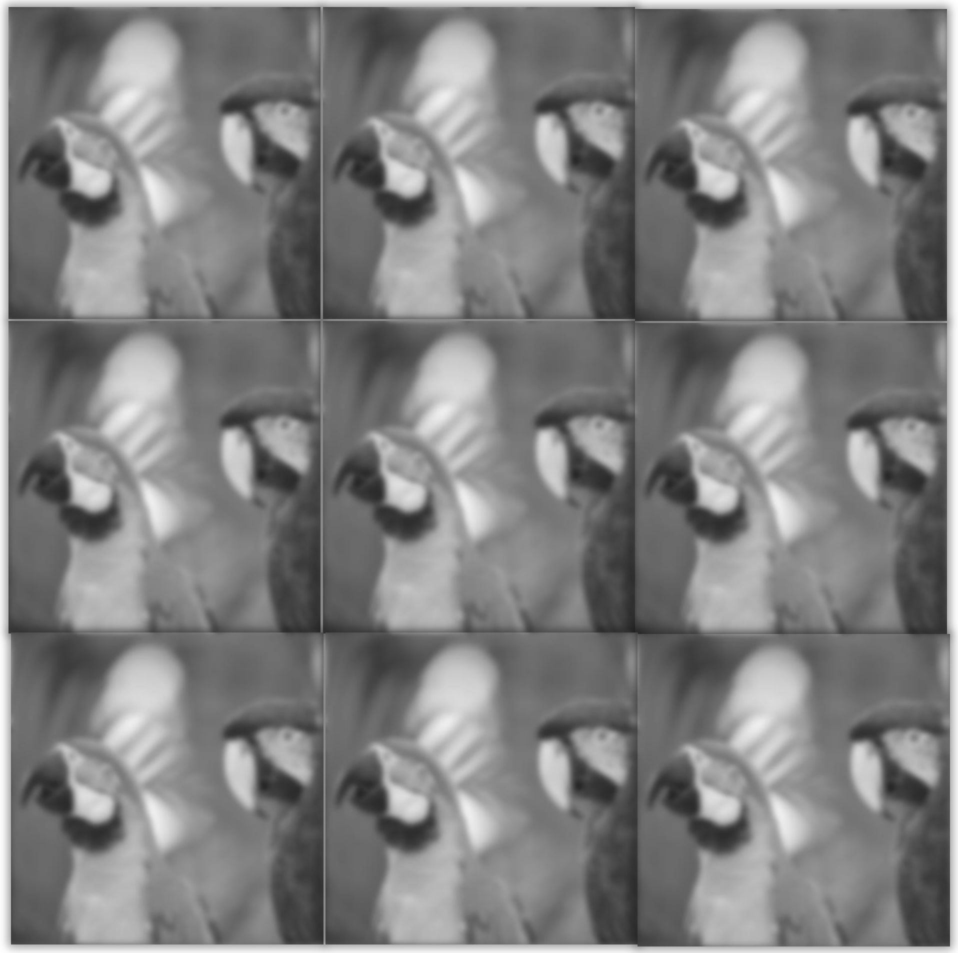
Periodic primal domain signal

What's the picture in 2D?

Periodic spectral copies in the frequency domain



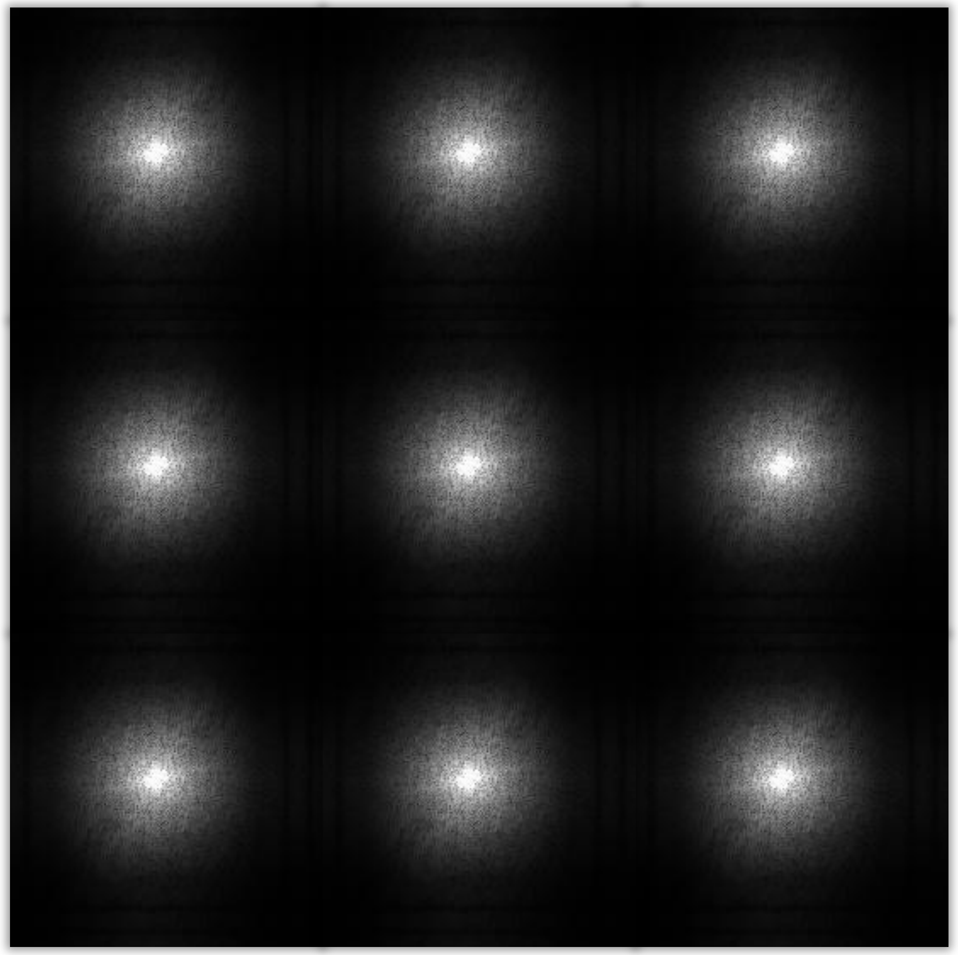
Periodic primal domain signal



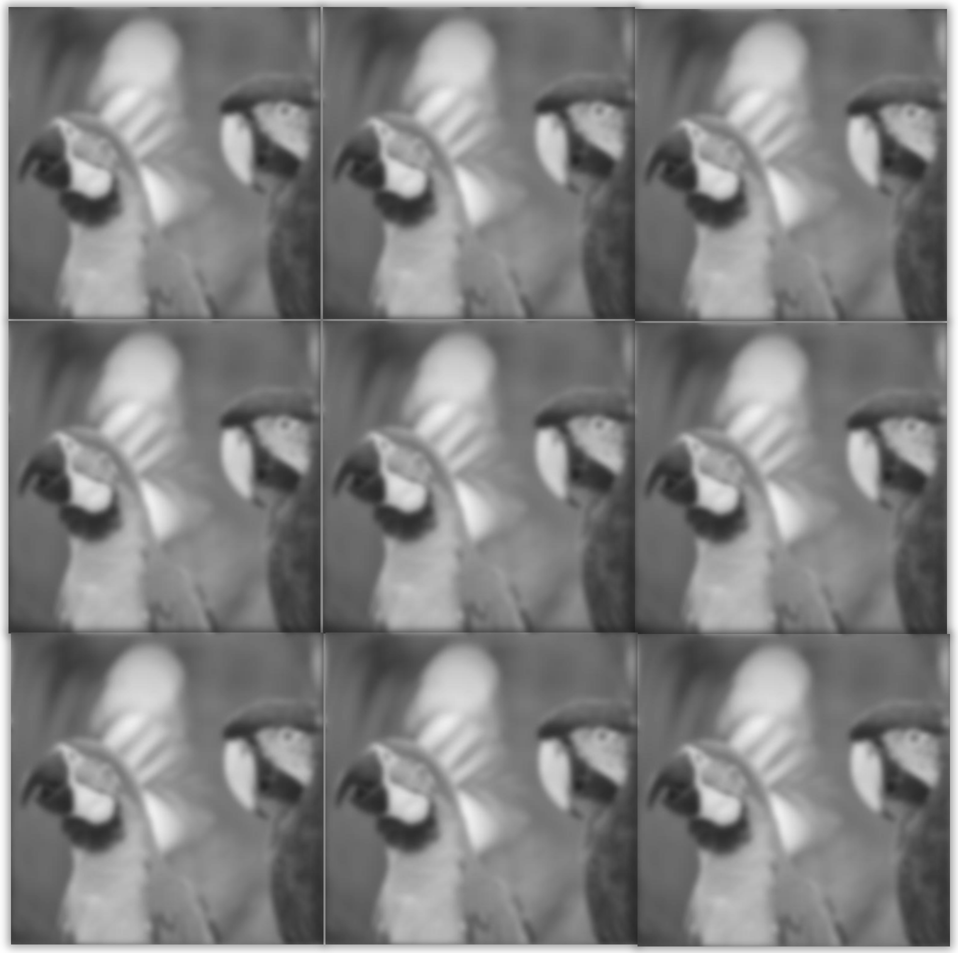
After low-pass filtering

What's the picture in 2D?

Periodic spectral copies in the frequency domain



Periodic primal domain signal



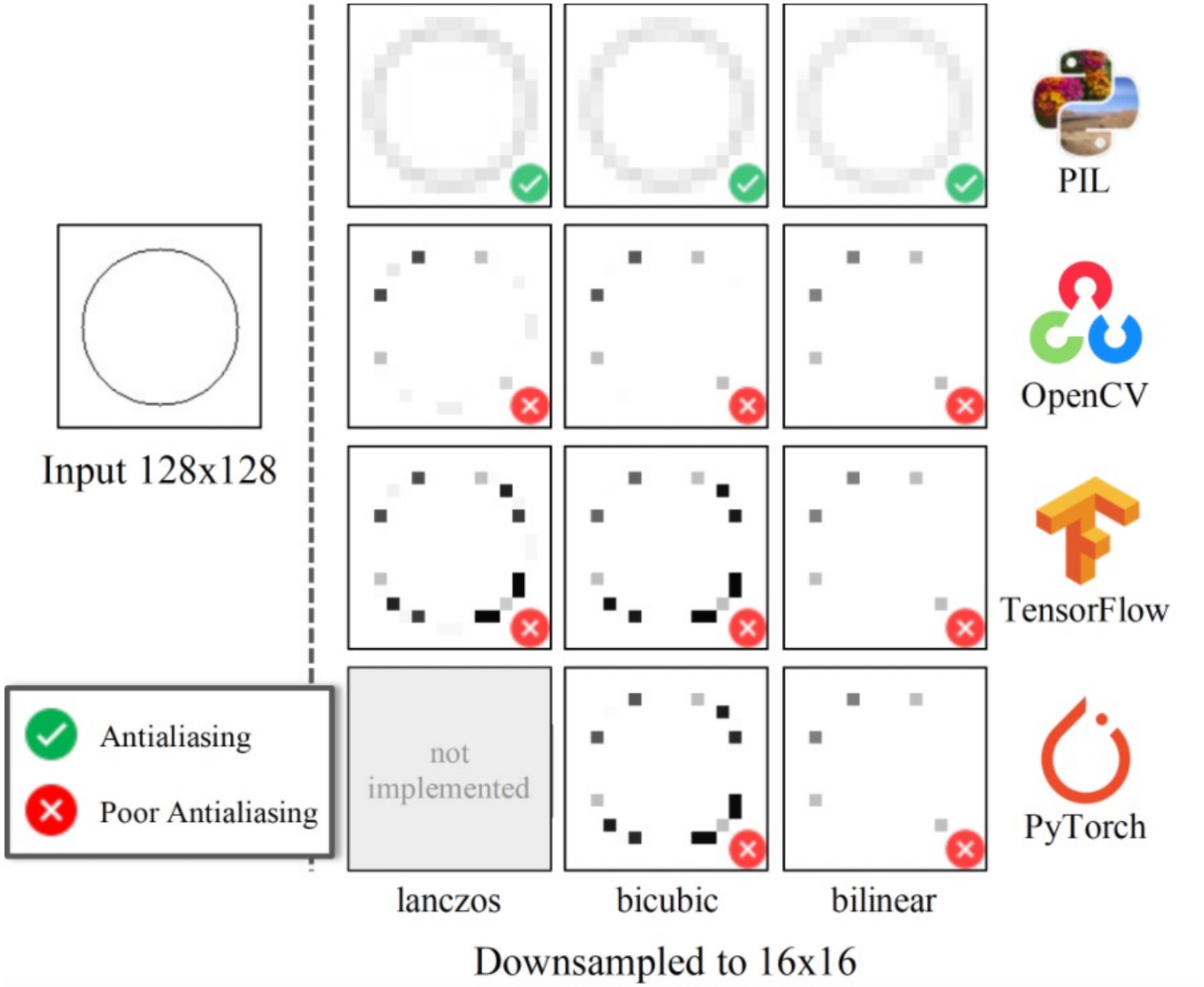
After sub-sampling (no aliasing!)

Image Downsampling (& Upsampling)

- "anti-aliasing" → **before** re-sampling, apply appropriate filter!

- how much filtering? Shannon-Nyquist sampling theorem:

$$f_s \geq 2 f_{\max}$$

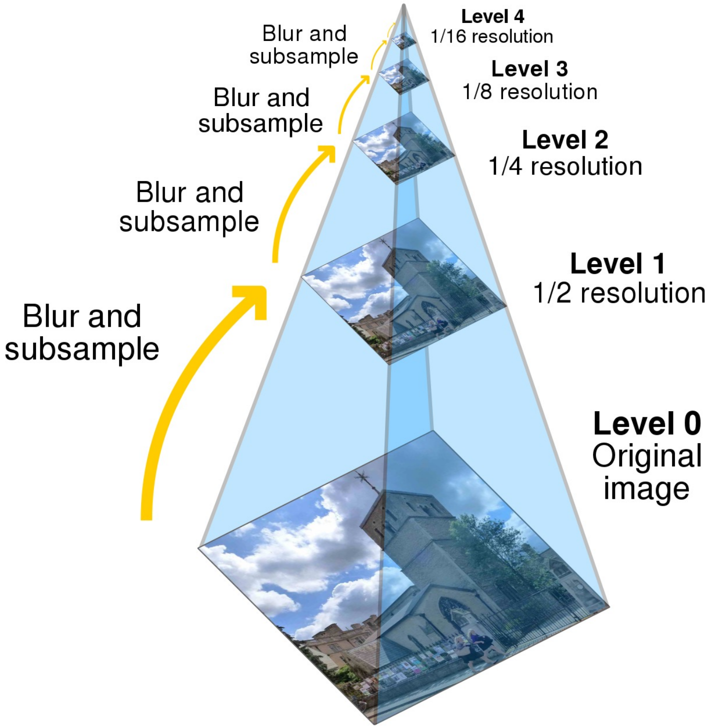


Parmar et al. 2021

Gaussian Pyramids [Burt and Adelson, 1983]

- A sequence of images created with Gaussian blurring and downsampling is called a Gaussian Pyramid
- In computer graphics, a mip map [Williams, 1983]

Idea: Represent NxN image as a "pyramid" of 1×1, 2×2, 4×4,..., $2^k \times 2^k$ images (assuming $N=2^k$)



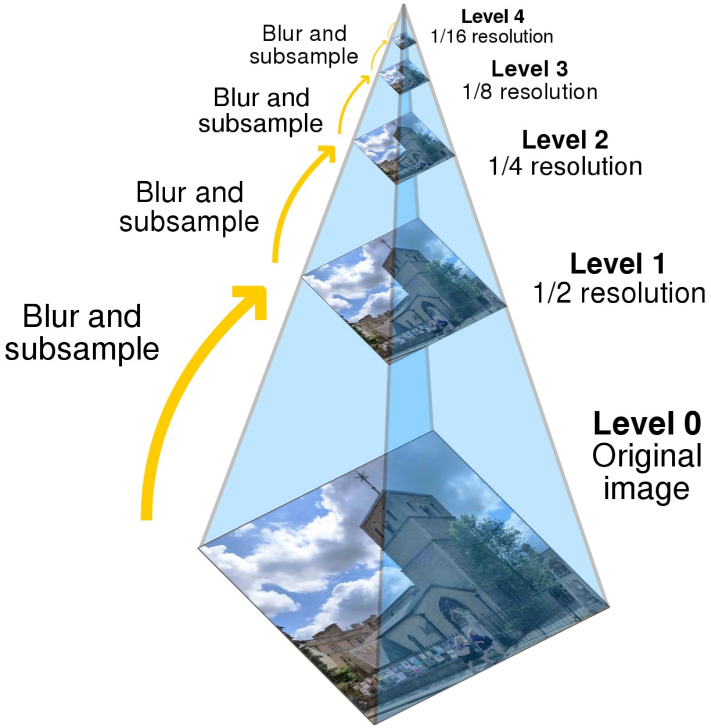
[Source: S. Seitz]

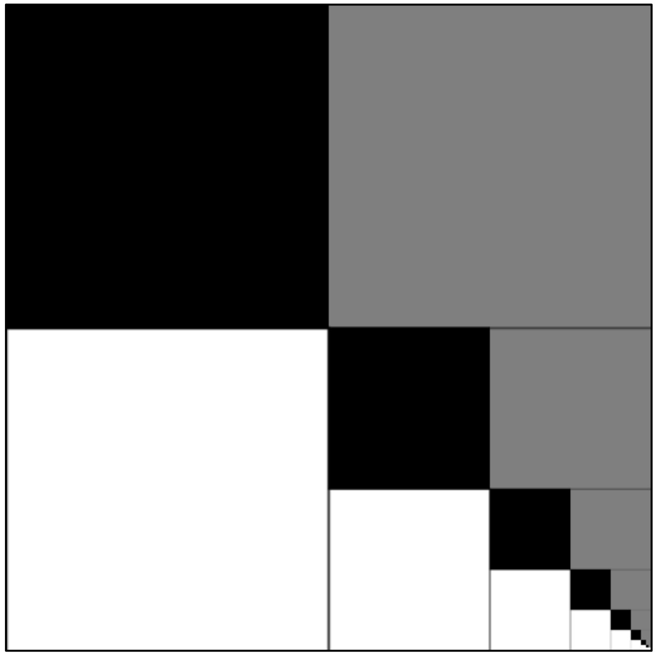
How much space does a Gaussian pyramid take compared to original image?

Gaussian Pyramids [Burt and Adelson, 1983]

- A sequence of images created with Gaussian blurring and downsampling is called a Gaussian Pyramid
- In computer graphics, a mip map [Williams, 1983]

Idea: Represent NxN image as a "pyramid" of 1×1, 2×2, 4×4,…, $2^k\times2^k$ images (assuming $N=2^k$)



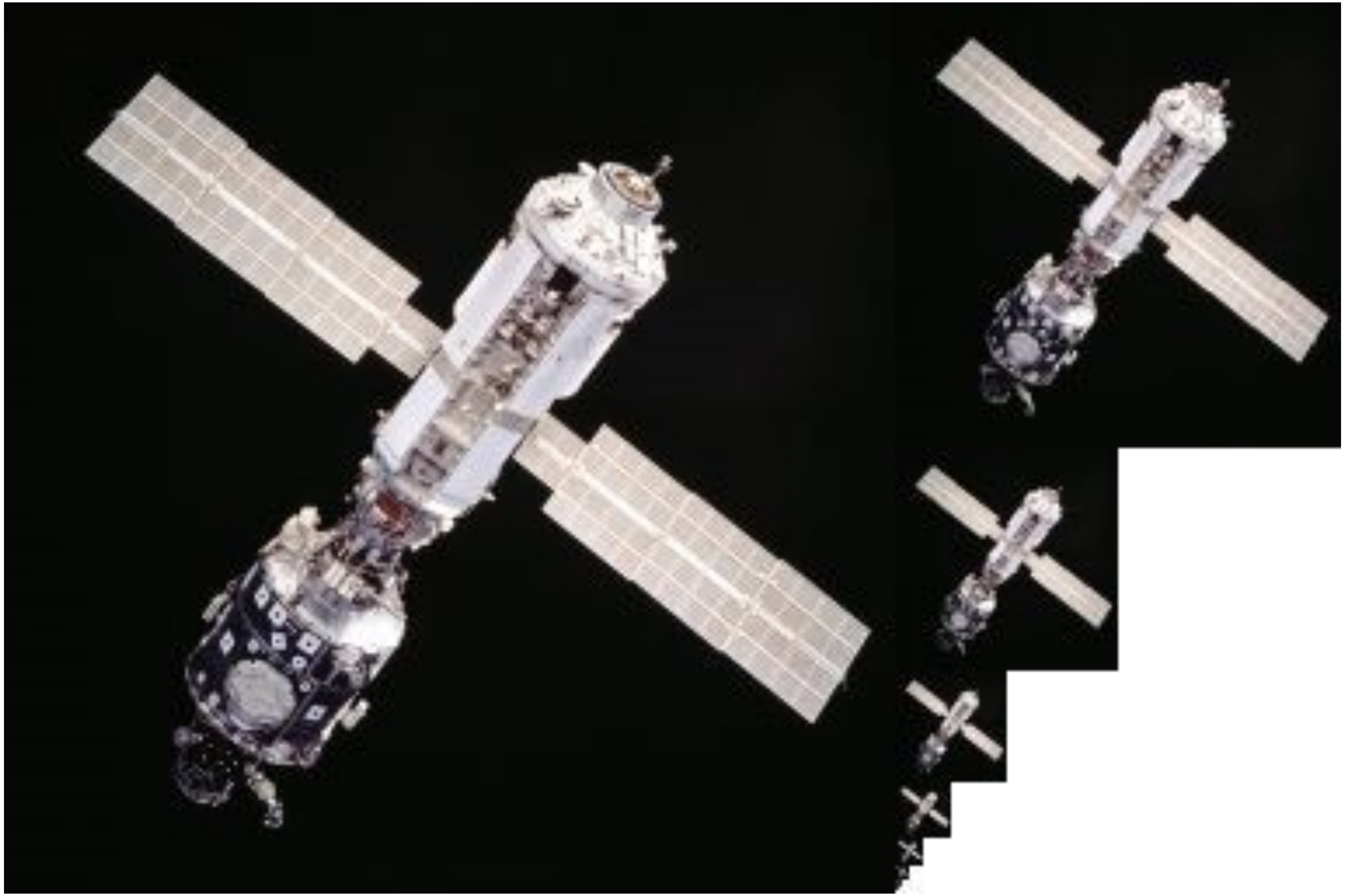


$$\sum_{n=1}^{\infty} \frac{1}{4^n} = \frac{1/4}{1-1/4} = \frac{1}{3}$$

[Source: S. Seitz]

How much space does a Gaussian pyramid take compared to original image?

Example of Gaussian Pyramid



[Source: N. Snavely]

Image Up-Sampling

- This image is too small, how can we make it 10 times as big?

[Source: N. Snavely, R. Urtasun]

Image Up-Sampling

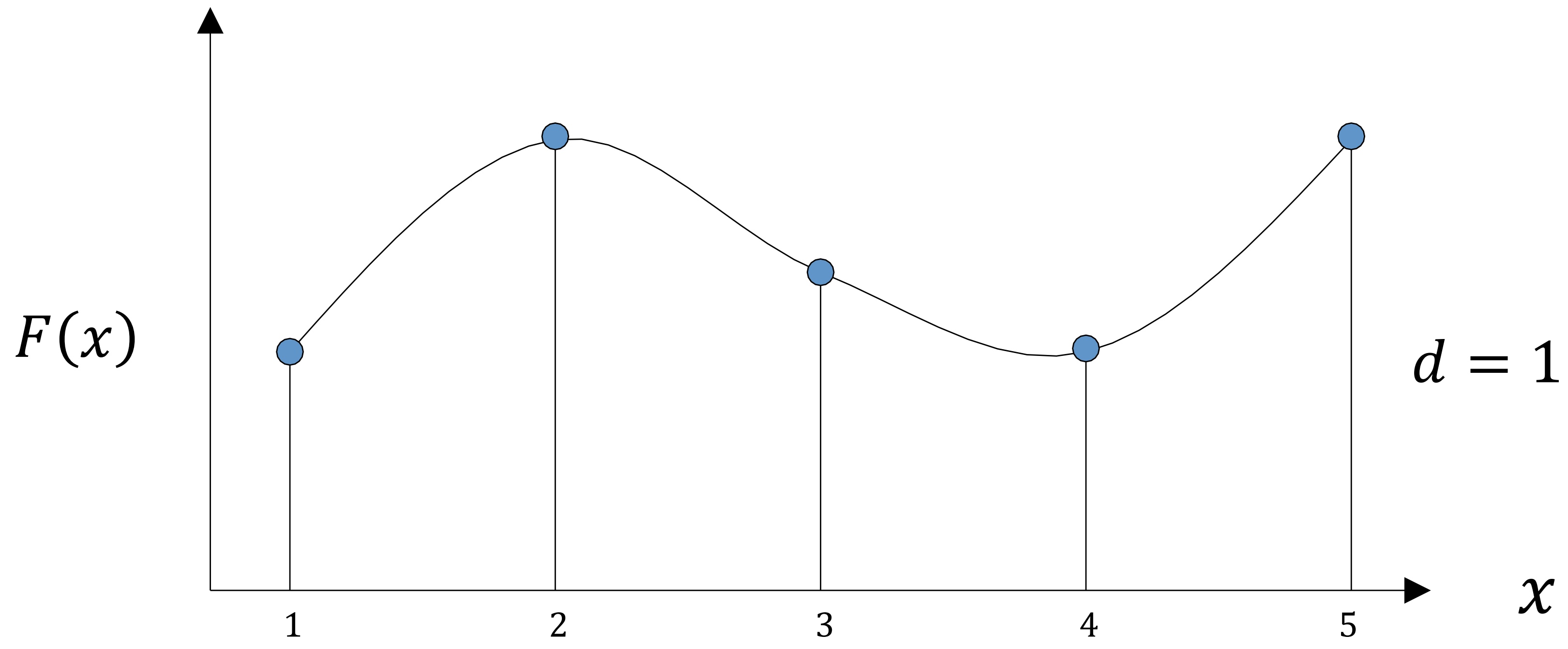
- This image is too small, how can we make it 10 times as big?

- Simplest approach: repeat each row and column 10 times

[Source: N. Snavely, R. Urtasun]

Interpolation

Interpolation



$d = 1$ *in this example*

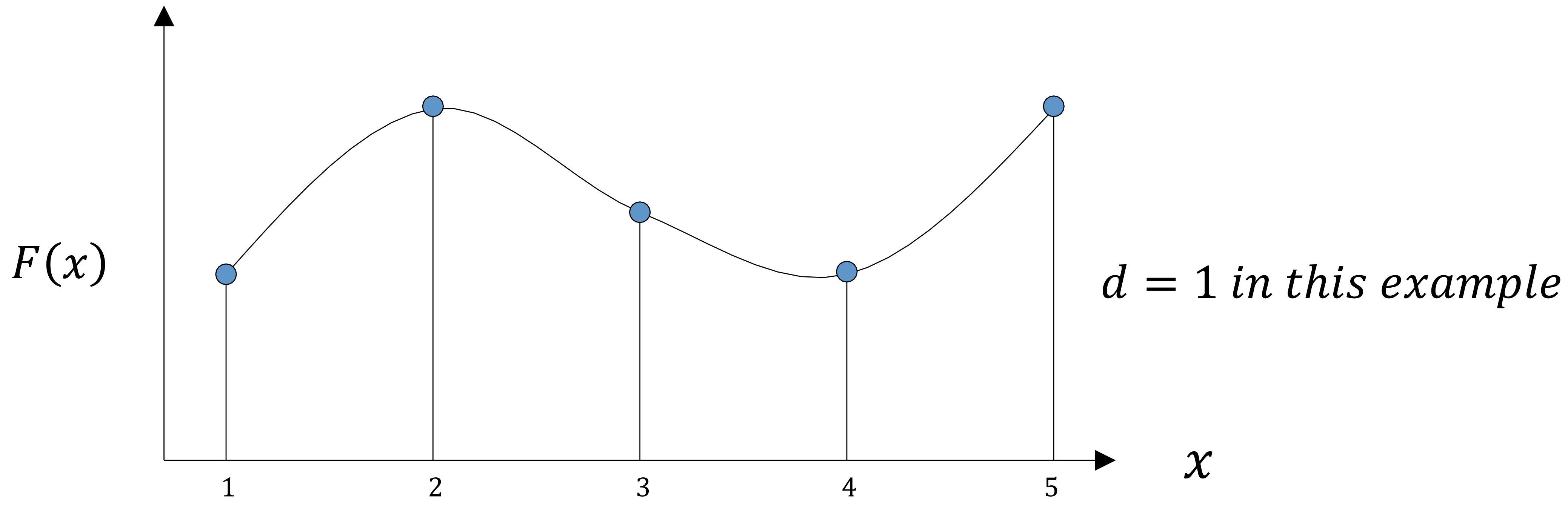
- Recall how digital image is formed

$$F[x, y] = quantize\ \{f\left(\frac{x}{d}, \frac{y}{d}\right)\}$$

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

[Source: N. Snavely, S. Seitz]

Interpolation



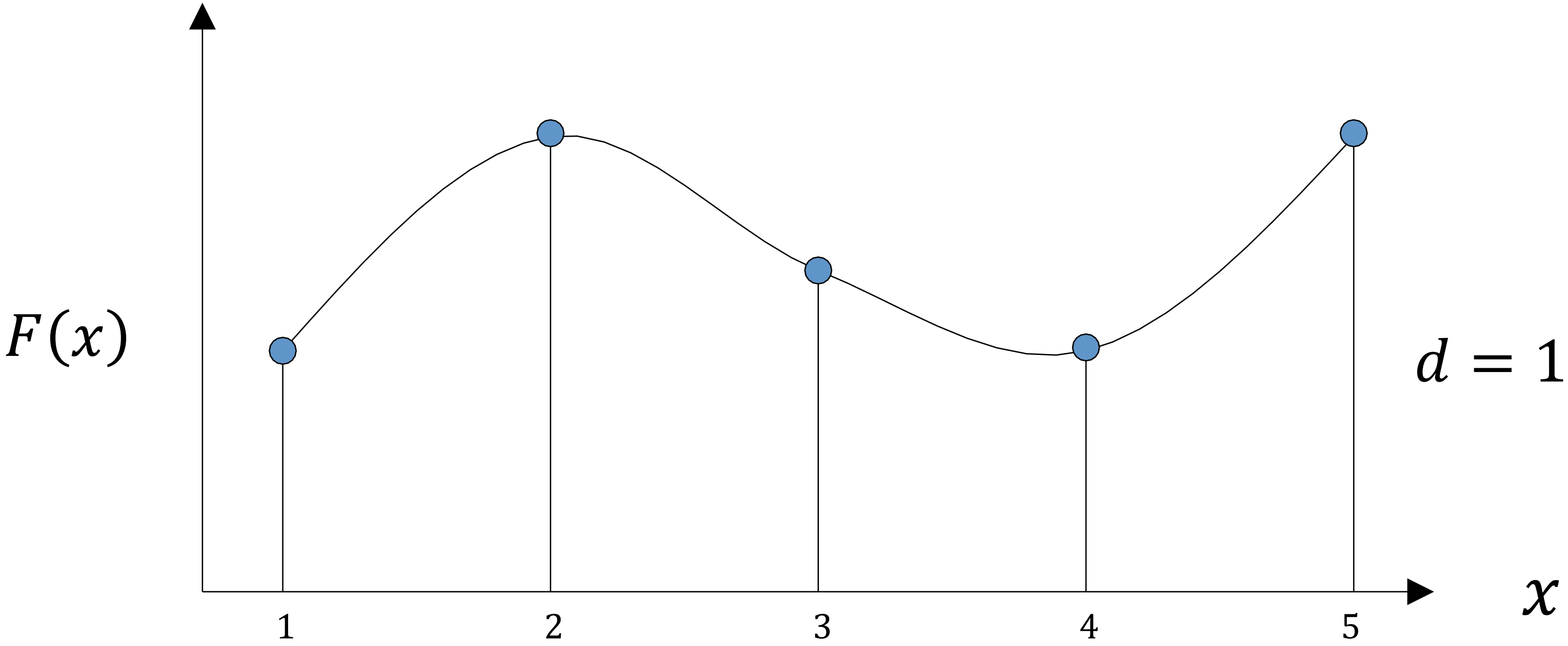
- Recall how digital image is formed

$$F[x, y] = quantize\left\{f\left(\frac{x}{d}, \frac{y}{d}\right)\right\}$$

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

[Source: N. Snavely, S. Seitz]

Interpolation



$d = 1 \; in \; this \; example$

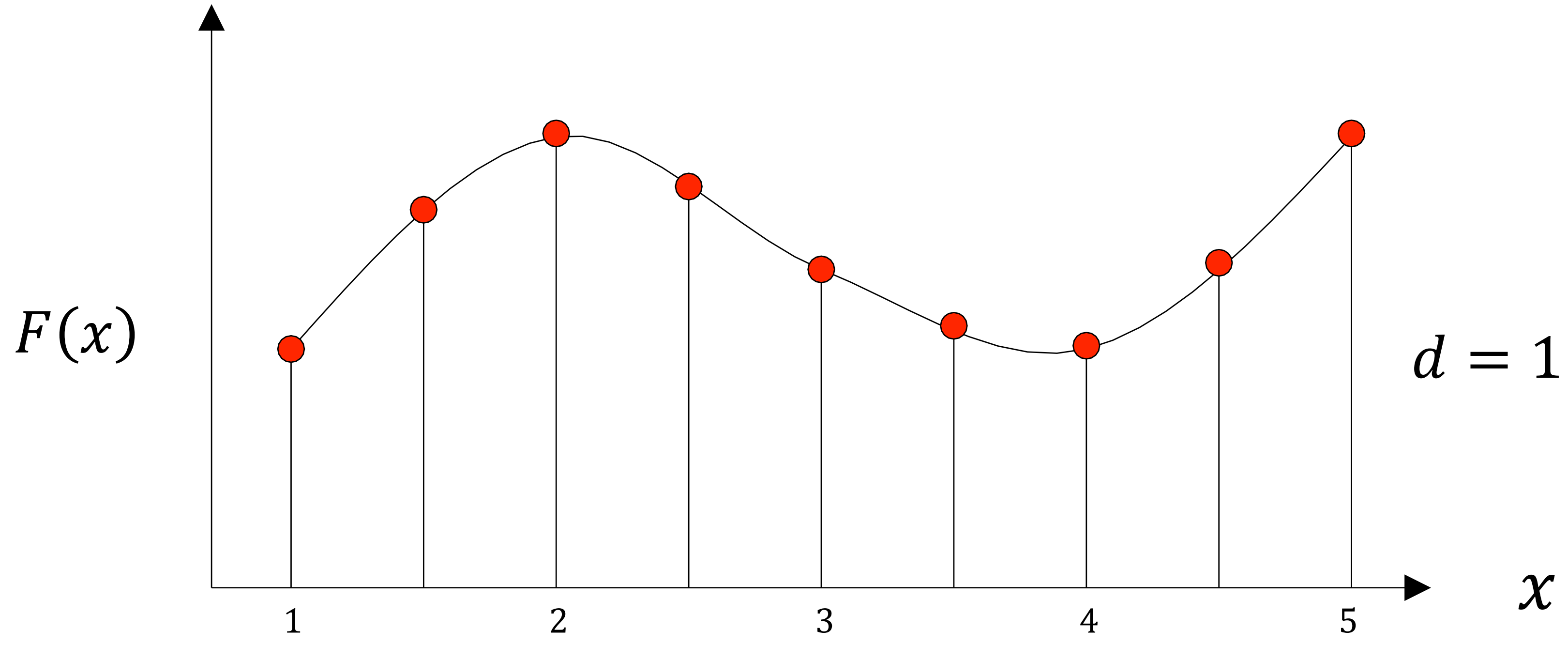
- Recall how digital image is formed

$$F[x, y] = quantize\left\{f\left(\frac{x}{d}, \frac{y}{d}\right)\right\}$$

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

[Source: N. Snavely, S. Seitz]

Interpolation



$d = 1$ *in this example*

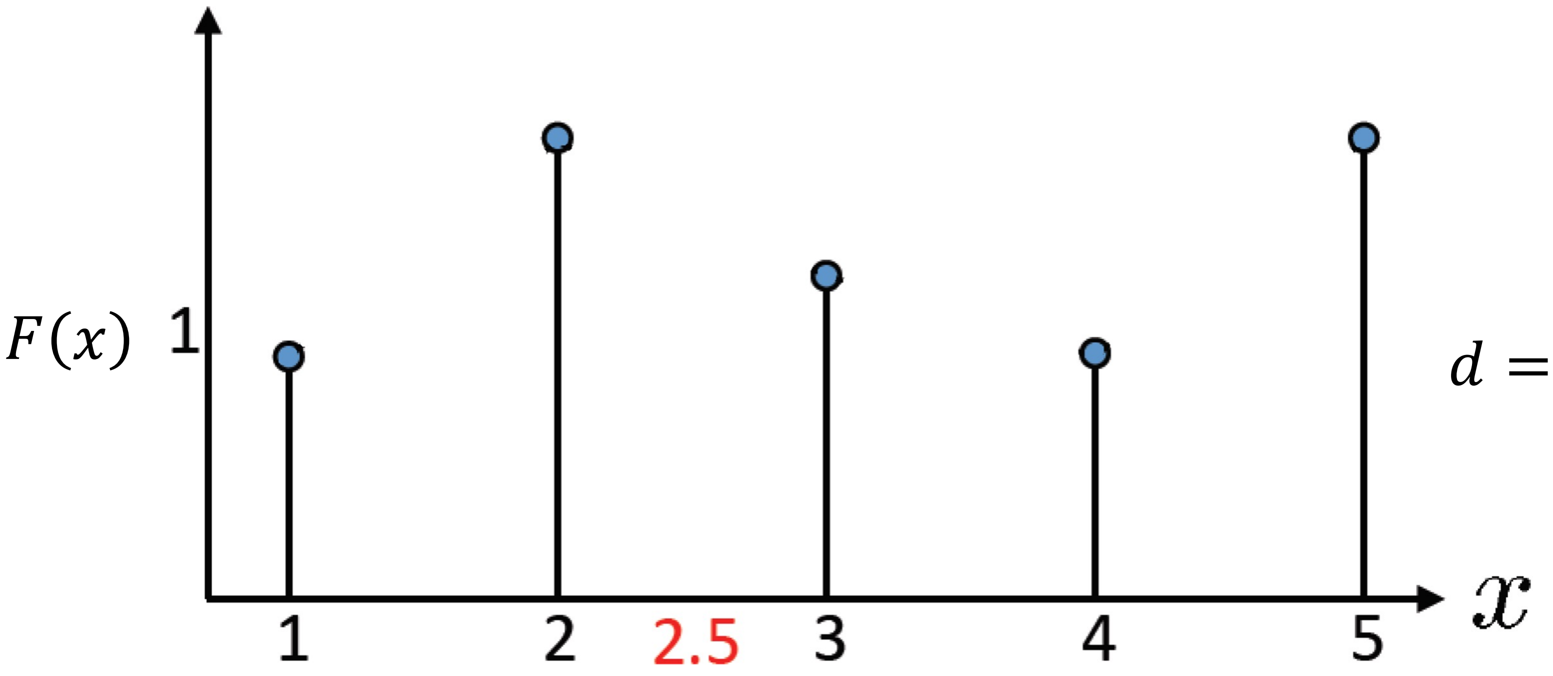
- Recall how digital image is formed

$$F[x,y] = quantize\left\{f\left(\frac{x}{d},\frac{y}{d}\right)\right\}$$

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

[Source: N. Snavely, S. Seitz]

Interpolation

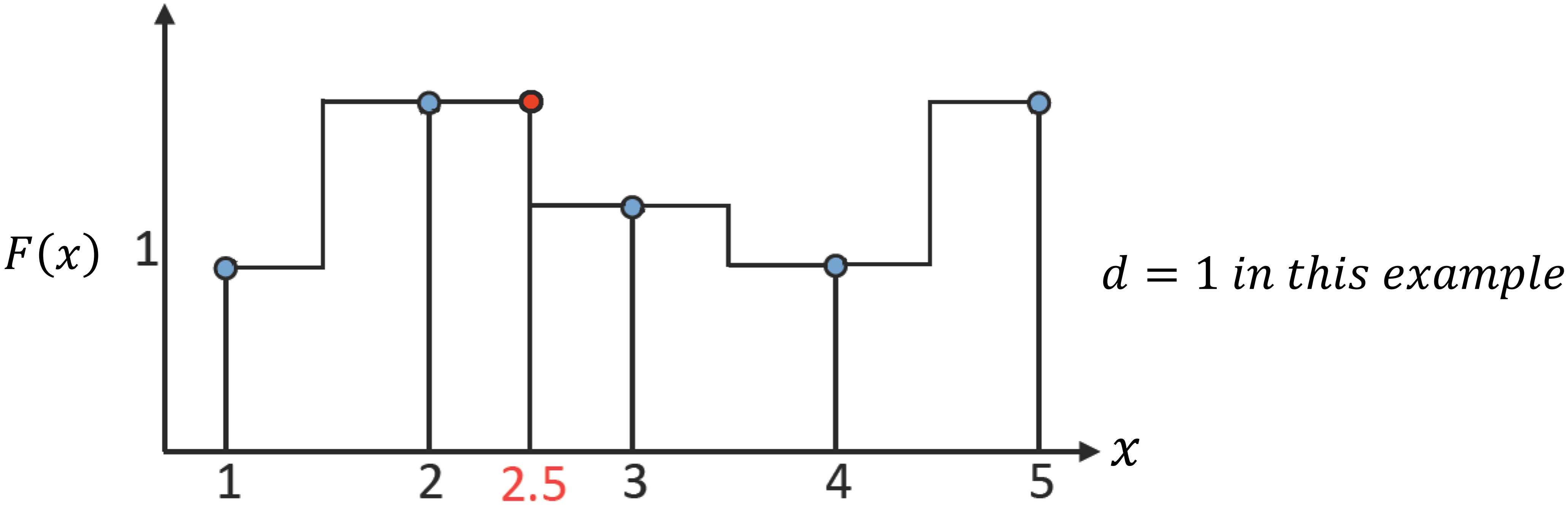


$d = 1$ *in this example*

- What if we don't know f?

[Source: N. Snavely, S. Seitz]

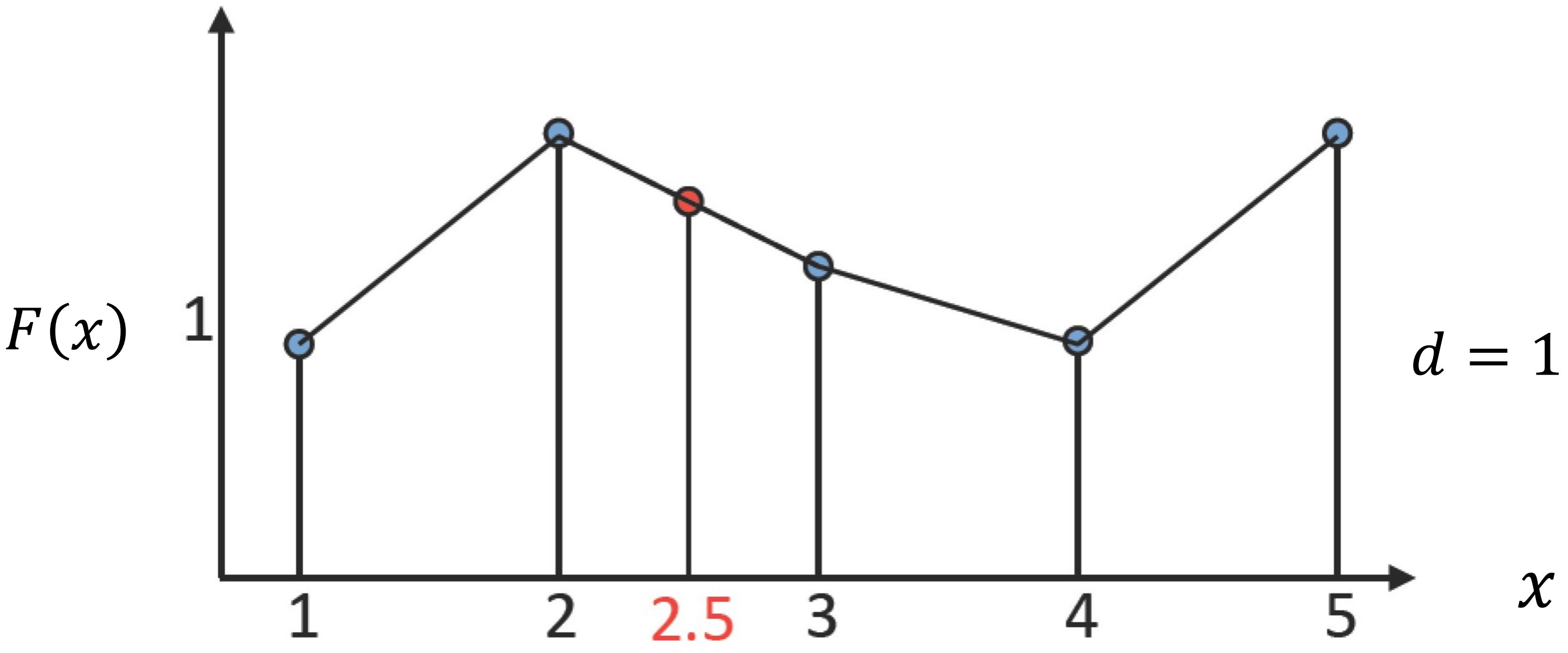
Interpolation



- What if we don't know f?
  - Guess an approximation: for example nearest-neighbor

[Source: N. Snavely, S. Seitz]

Interpolation

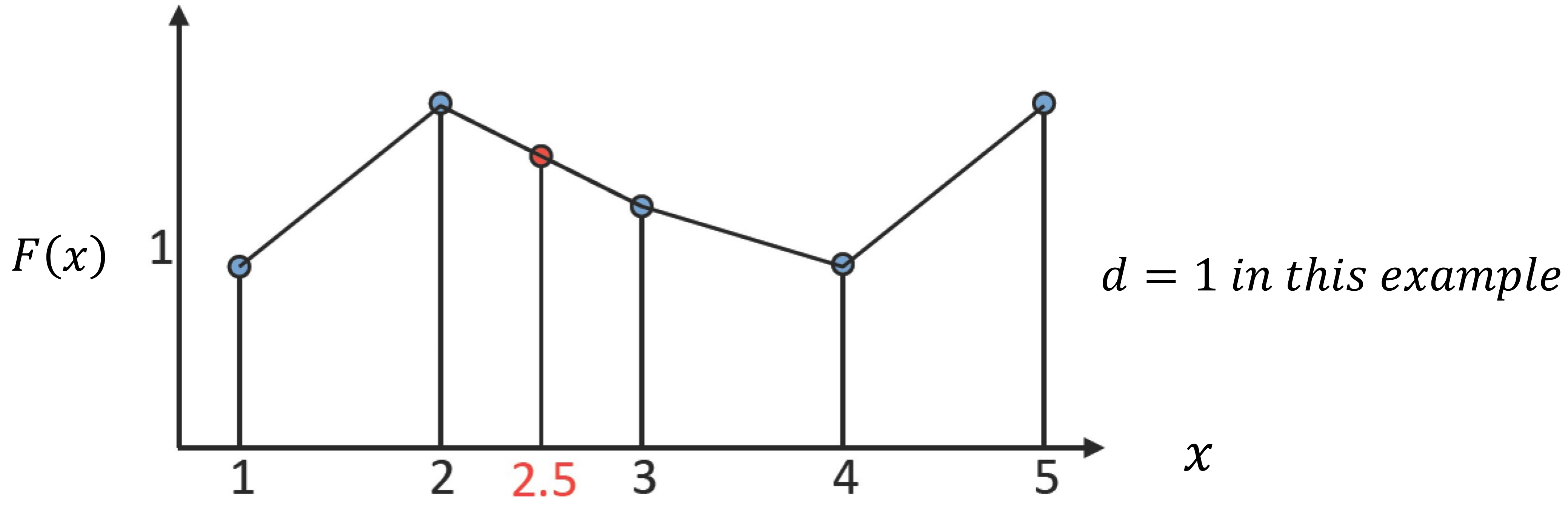


$d = 1$ *in this example*

- What if we don't know f?
  - Guess an approximation: for example nearest-neighbor
  - Guess an approximation: for example linear

[Source: N. Snavely, S. Seitz]

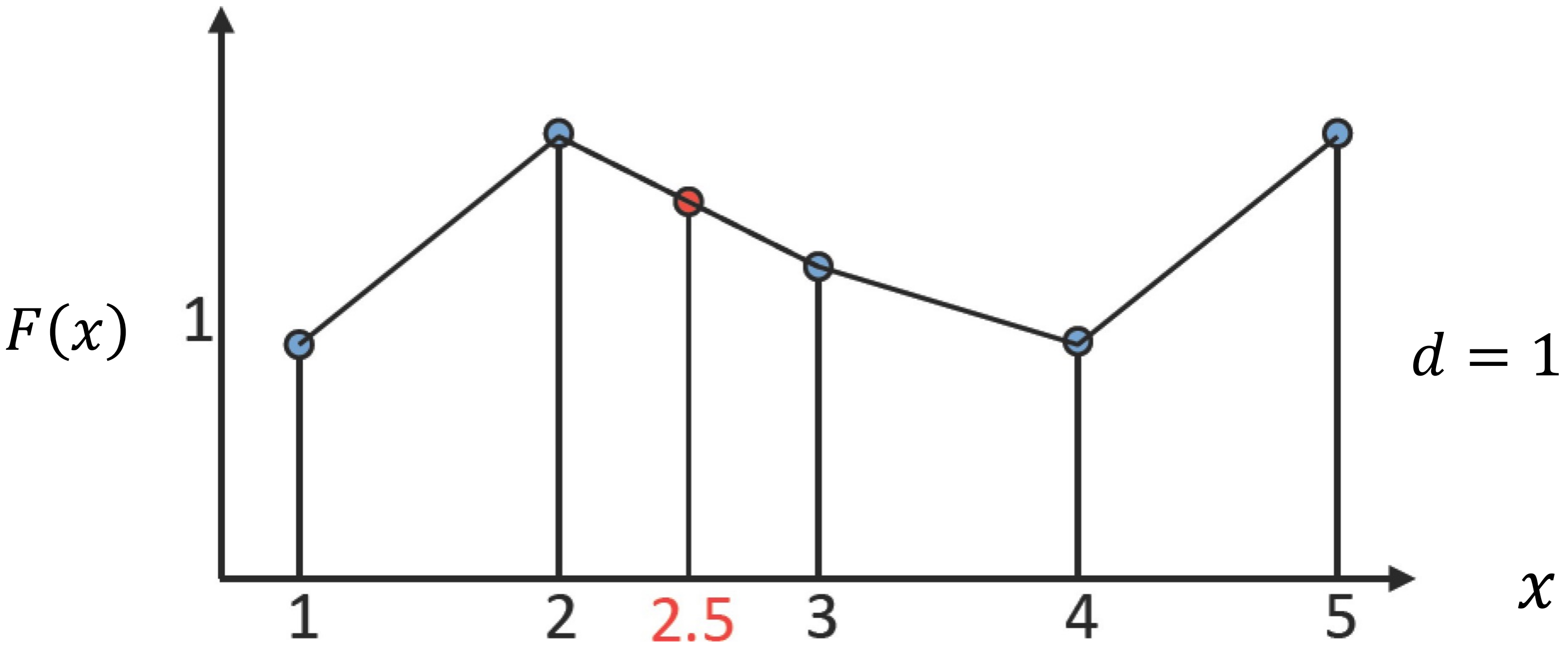
Interpolation



- What if we don't know f?
  - Guess an approximation: for example nearest-neighbor
  - Guess an approximation: for example linear
  - More complex approximations: cubic, B-splines

[Source: N. Snavely, S. Seitz]

Interpolation

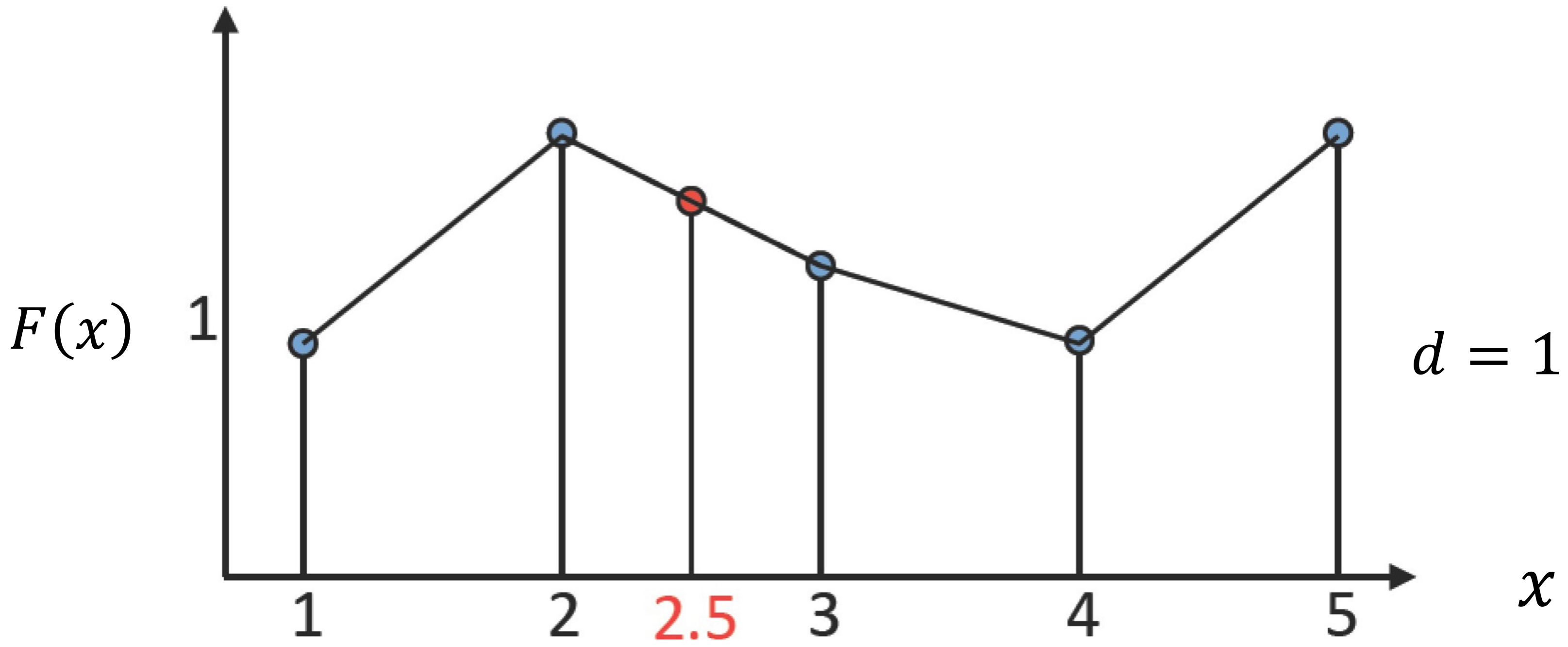


$d = 1 \ in \ this \ example$

- What if we don't know f?
  - Guess an approximation: for example nearest-neighbor
  - Guess an approximation: for example linear
  - More complex approximations: cubic, B-splines
  - But more isn't always better!

[Source: N. Snavely, S. Seitz]

Linear Interpolation



$d = 1 \ in \ this \ example$

- Linear interpolation from our discretized F :

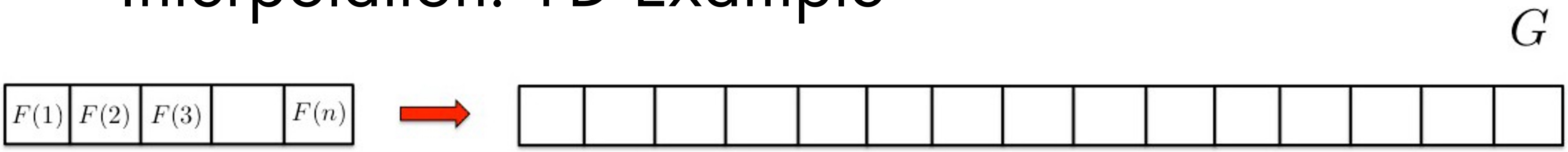
$$G(x) = \frac{x_2 - x}{x_2 - x_1} F(x_1) + \frac{x - x_1}{x_2 - x_1} F(x_2)$$

Interpolation: 1D Example

$F(1)$	$F(2)$	$F(3)$		$F(n)$

- Let's upsample by a factor of d=3

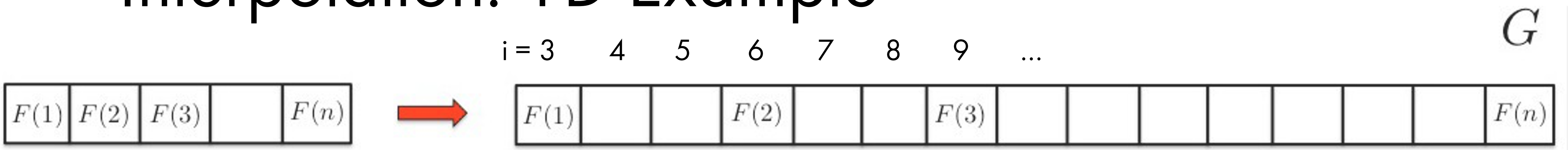
Interpolation: 1D Example



Make a vector G with d times the size of F

- Let's upsample by a factor of d=3

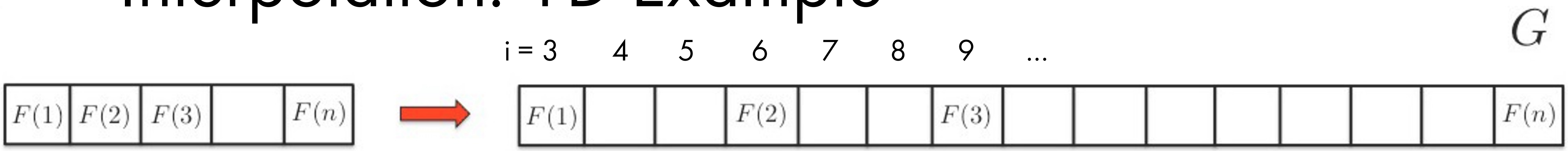
Interpolation: 1D Example



$$\text{if } \frac{i}{d} \text{ integer: } G(i) = F(i/d)$$

- Let's upsample by a factor of d=3
- if i/d is integer, just copy over the value

Interpolation: 1D Example



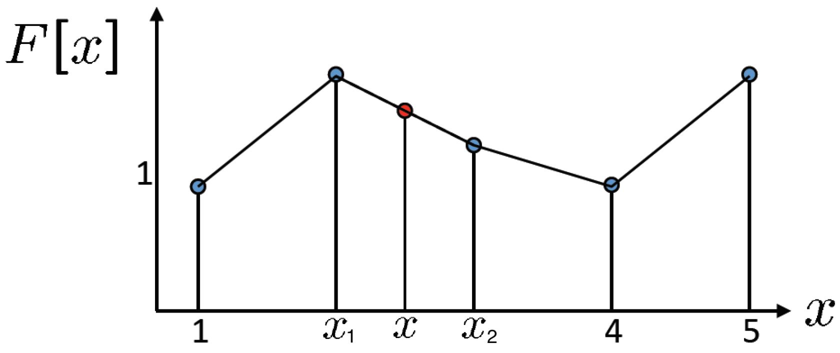
$$\text{if } \frac{i}{d} \text{ integer: } G(i) = F(i/d)$$

$$\text{otherwise: } G(i) = \frac{x_2 - x}{x_2 - x_1} F(x_1) + \frac{x - x_1}{x_2 - x_1} F(x_2)$$

$$\text{where} \quad \begin{aligned} x &= i/d \\ x_1 &= \lfloor i/d \rfloor \\ x_2 &= \lceil i/d \rceil \end{aligned}$$

- Let's upsample by a factor of d=3
- if i/d is integer, just copy over the value
- otherwise, use interpolation formula

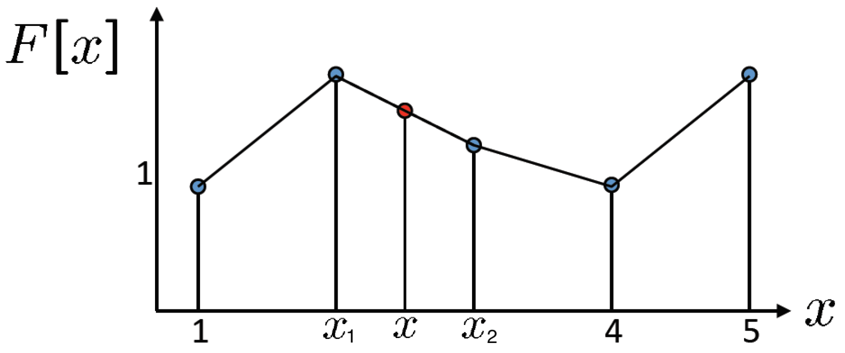
Linear Interpolation via Convolution



- Linear interpolation:

$$G(x) = \frac{x_2 - x}{x_2 - x_1} F(x_1) + \frac{x - x_1}{x_2 - x_1} F(x_2)$$

Linear Interpolation via Convolution

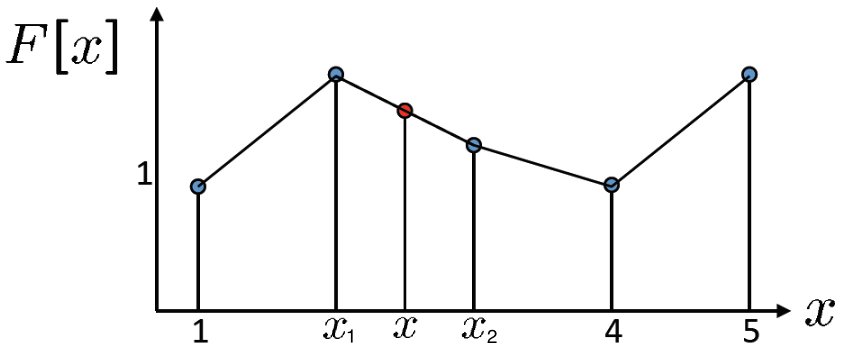


- Linear interpolation:

$$G(x) = \frac{x_2 - x}{x_2 - x_1} F(x_1) + \frac{x - x_1}{x_2 - x_1} F(x_2)$$

- how can we do this with a convolution?

Linear Interpolation via Convolution



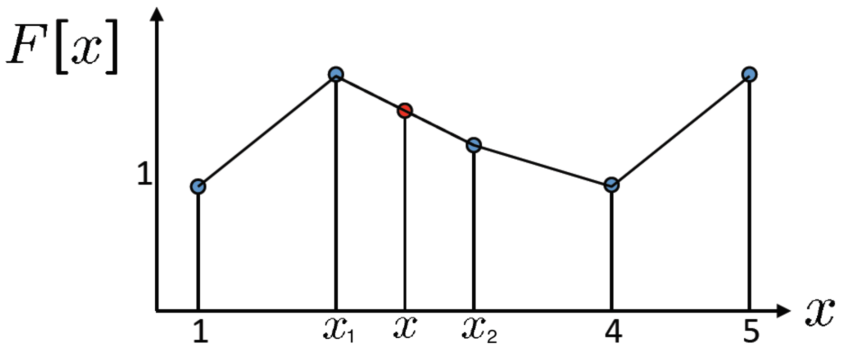
- Linear interpolation:

$$G(x) = \frac{x_2 - x}{x_2 - x_1} F(x_1) + \frac{x - x_1}{x_2 - x_1} F(x_2)$$

- how can we do this with a convolution?

- what should be the input?

Linear Interpolation via Convolution



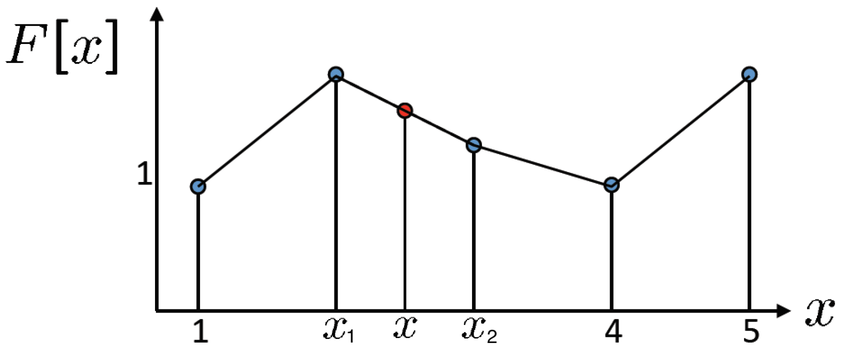
- Linear interpolation:

$$G(x) = \frac{x_2 - x}{x_2 - x_1} F(x_1) + \frac{x - x_1}{x_2 - x_1} F(x_2)$$

- how can we do this with a convolution?

- what should be the input? (original signal interleaved with zeroes)

Linear Interpolation via Convolution



- Linear interpolation:

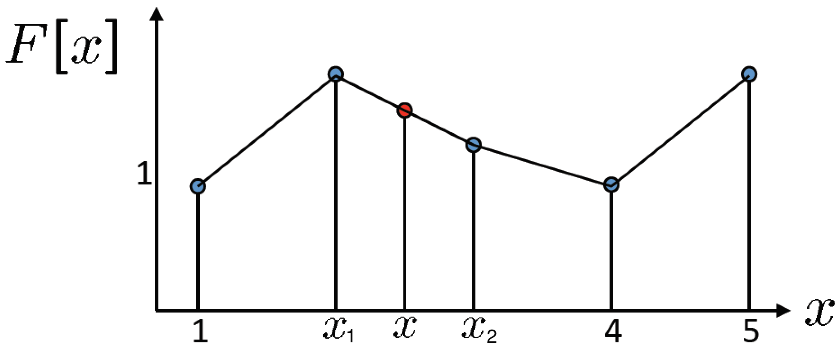
$$G(x) = \frac{x_2 - x}{x_2 - x_1} F(x_1) + \frac{x - x_1}{x_2 - x_1} F(x_2)$$

- how can we do this with a convolution?

- what should be the input? (original signal interleaved with zeroes)

- what should be the filter?

Linear Interpolation via Convolution



- Linear interpolation:

$$G(x) = \frac{x_2 - x}{x_2 - x_1} F(x_1) + \frac{x - x_1}{x_2 - x_1} F(x_2)$$

- how can we do this with a convolution?

- what should be the input? (original signal interleaved with zeroes)

- what should be the filter?

$$G_{interpolated}(x_i) = \left[\frac{1}{2}, 1, \frac{1}{2}\right] * G'$$

Interpolation via Convolution: 1D Example

$F(1)$	$F(2)$	$F(3)$		$F(n)$

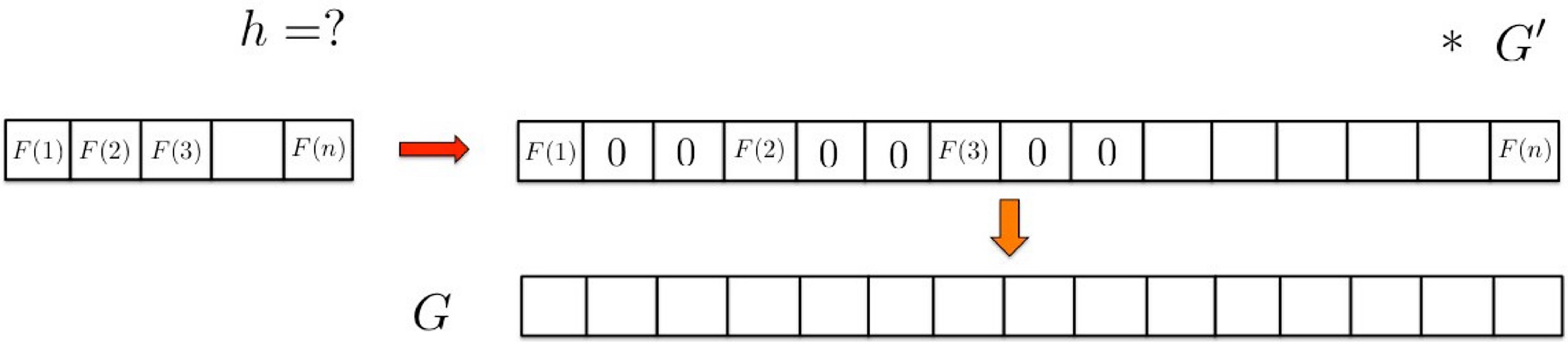
- Let's make this signal triple length (what will this look like?)

Interpolation via Convolution: 1D Example

$h = ?$

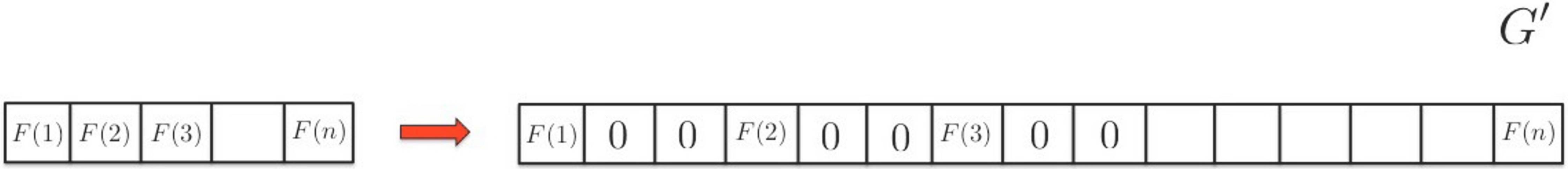
$* \quad G'$

Interpolation via Convolution: 1D Example



- how does this get copied to the output?

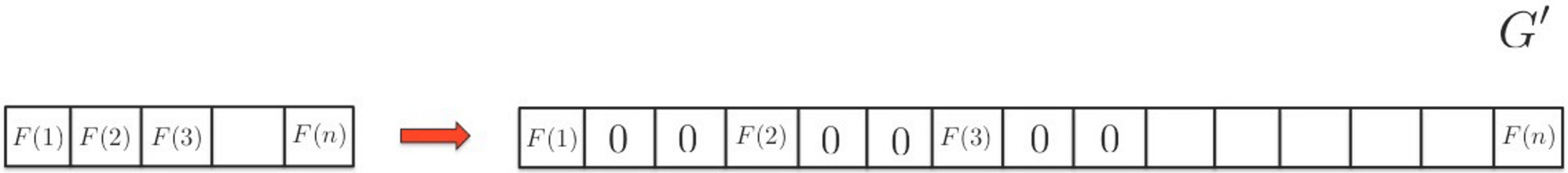
Interpolation via Convolution: 1D Example



$$\text{if } \frac{i}{d} \text{ integer: } G'(i) = F(i/d)$$

$$\text{otherwise: } 0$$

Interpolation via Convolution: 1D Example



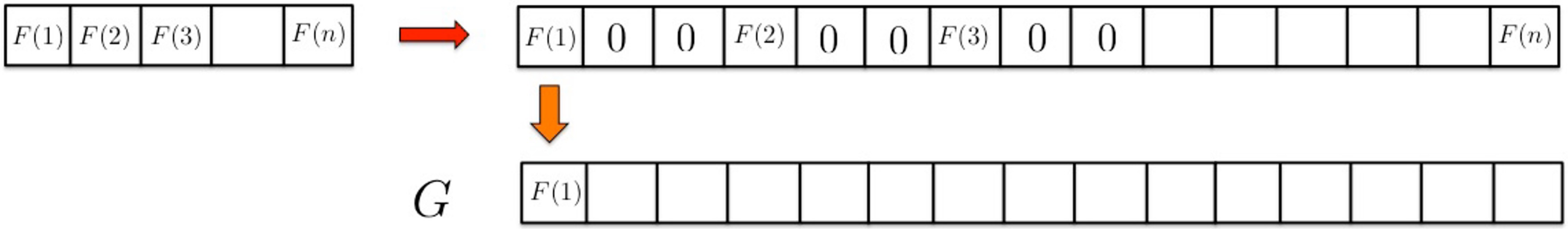
$$\text{if } \frac{i}{d} \text{ integer: } G'(i) = F(i/d)$$

$$\text{otherwise: } 0$$

- what does the convolution kernel look like?

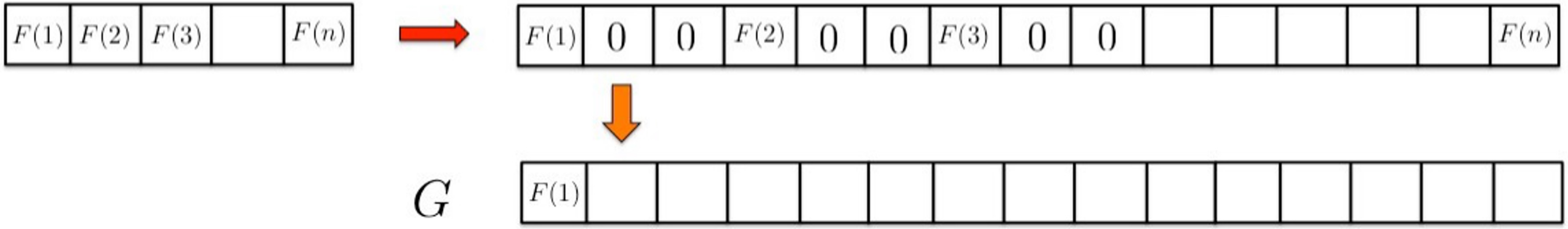
Interpolation via Convolution: 1D Example

$$h = \left[0, \frac{1}{3}, \frac{2}{3}, 1, \frac{2}{3}, \frac{1}{3}, 0\right] \qquad * \; G'$$



Interpolation via Convolution: 1D Example

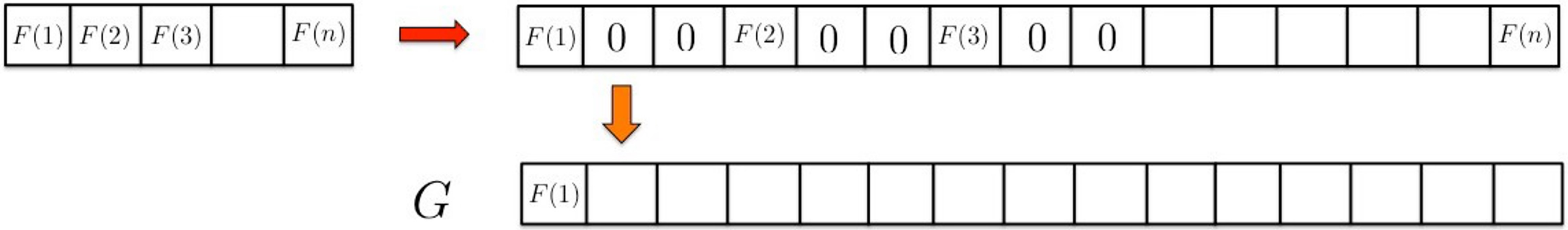
$$h = \left[0, \frac{1}{3}, \frac{2}{3}, 1, \frac{2}{3}, \frac{1}{3}, 0\right] \quad * \quad G'$$



- how do we calculate this entry?

Interpolation via Convolution: 1D Example

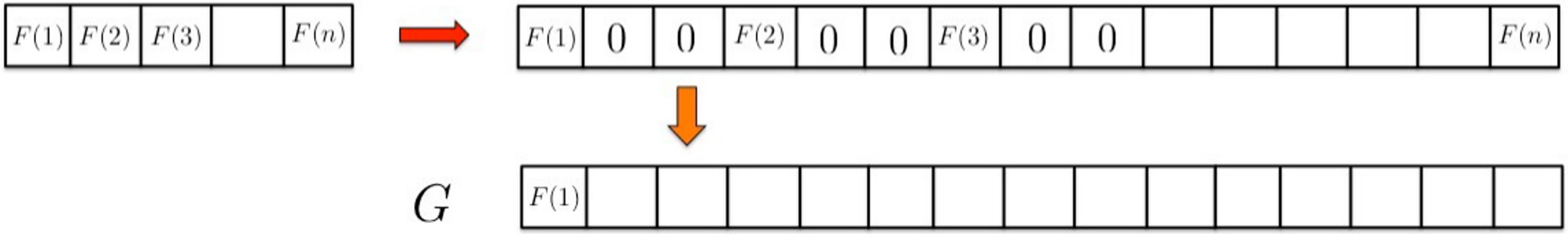
$$h = \left[0, \frac{1}{3}, \frac{2}{3}, 1, \frac{2}{3}, \frac{1}{3}, 0\right] \qquad * \; G'$$



$$\frac{2}{3}F(1) + \frac{1}{3}F(2)$$

Interpolation via Convolution: 1D Example

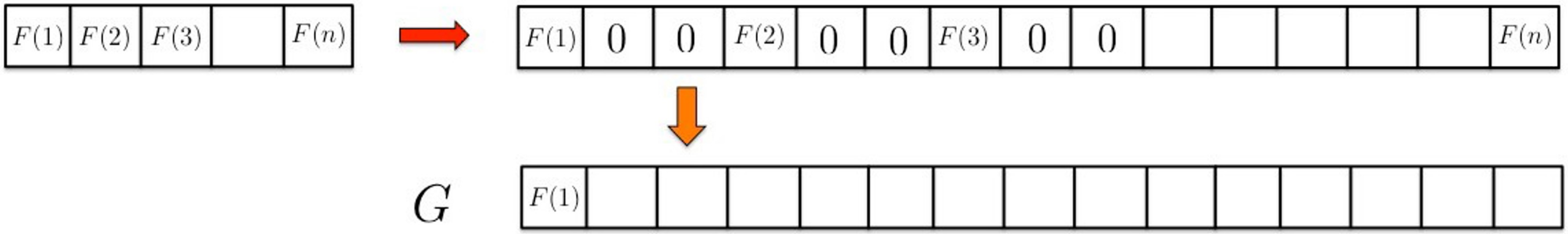
$$h = \left[0, \frac{1}{3}, \frac{2}{3}, 1, \frac{2}{3}, \frac{1}{3}, 0\right] \quad * \; G'$$



- how do we calculate this entry?

Interpolation via Convolution: 1D Example

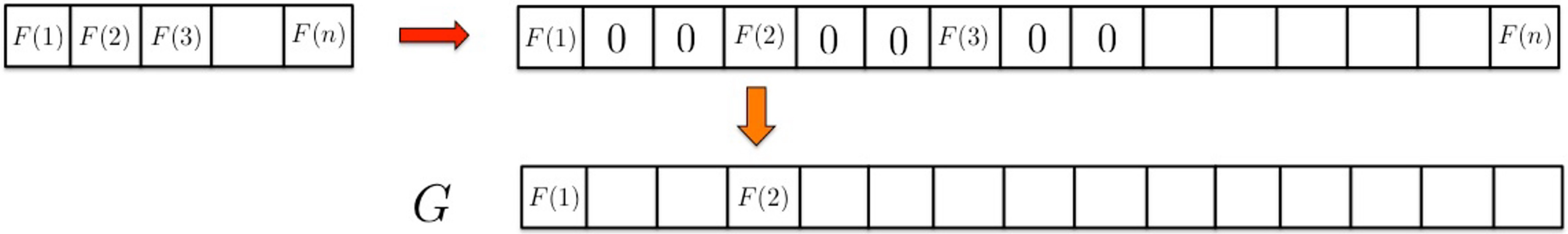
$$h = \left[0, \frac{1}{3}, \frac{2}{3}, 1, \frac{2}{3}, \frac{1}{3}, 0\right] \quad * \ G'$$



$$\frac{1}{3}F(1) + \frac{2}{3}F(2)$$

Interpolation via Convolution: 1D Example

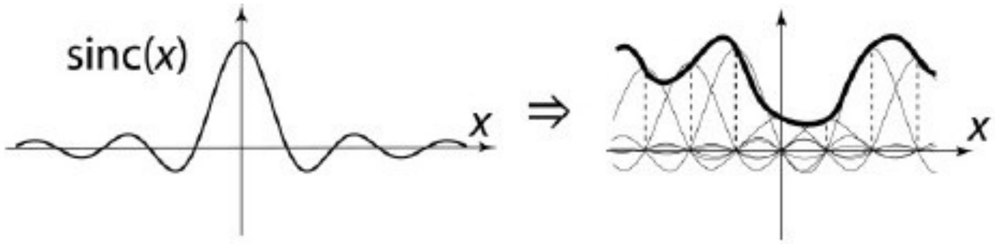
$$h = \left[0, \frac{1}{3}, \frac{2}{3}, 1, \frac{2}{3}, \frac{1}{3}, 0\right] \quad * \quad G'$$

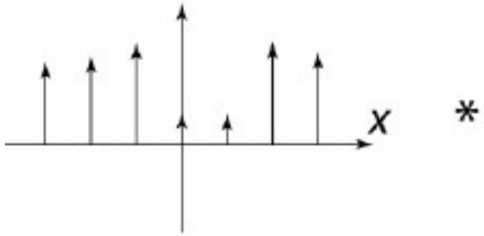


- ... and so on and so forth

Interpolation via Convolution (1D)

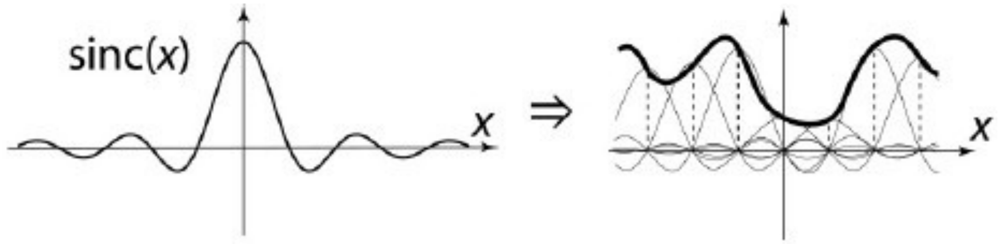
overview of
interpolation
kernels



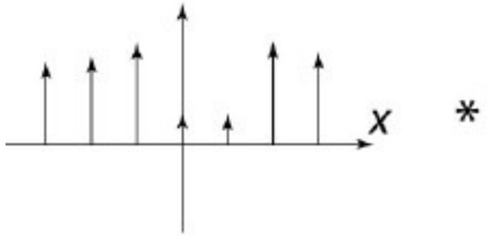


Interpolation via Convolution (1D)

overview of interpolation kernels

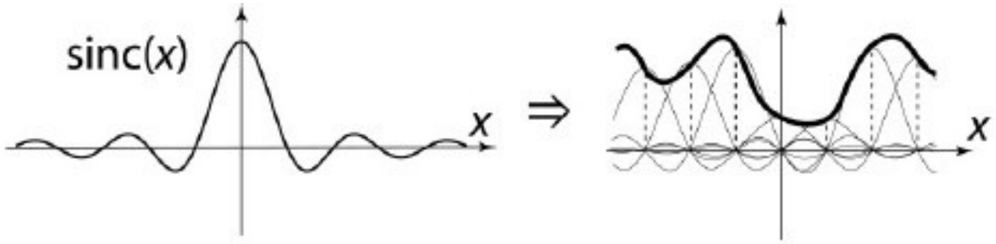


"Ideal" reconstruction

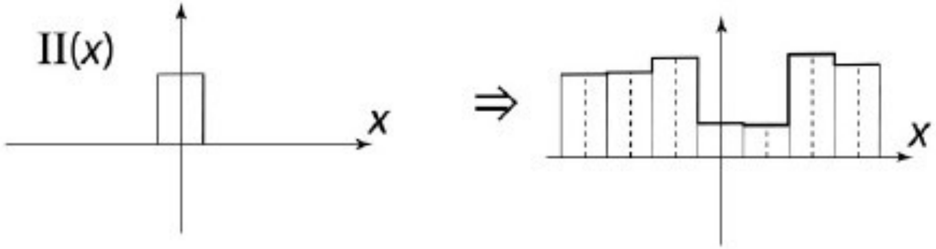


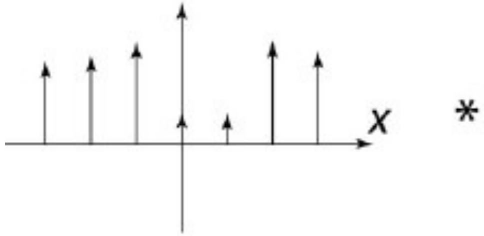
Interpolation via Convolution (1D)

overview of interpolation kernels



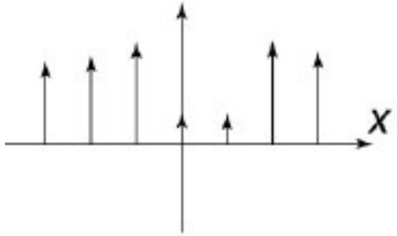
"Ideal" reconstruction



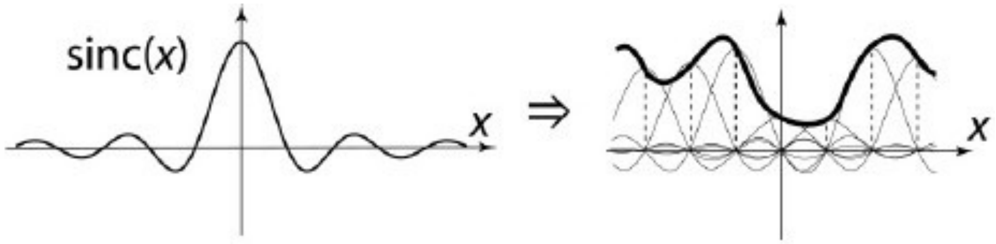


Interpolation via Convolution (1D)

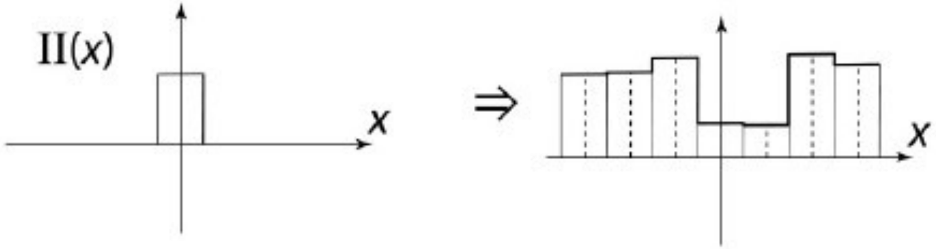
overview of interpolation kernels



*



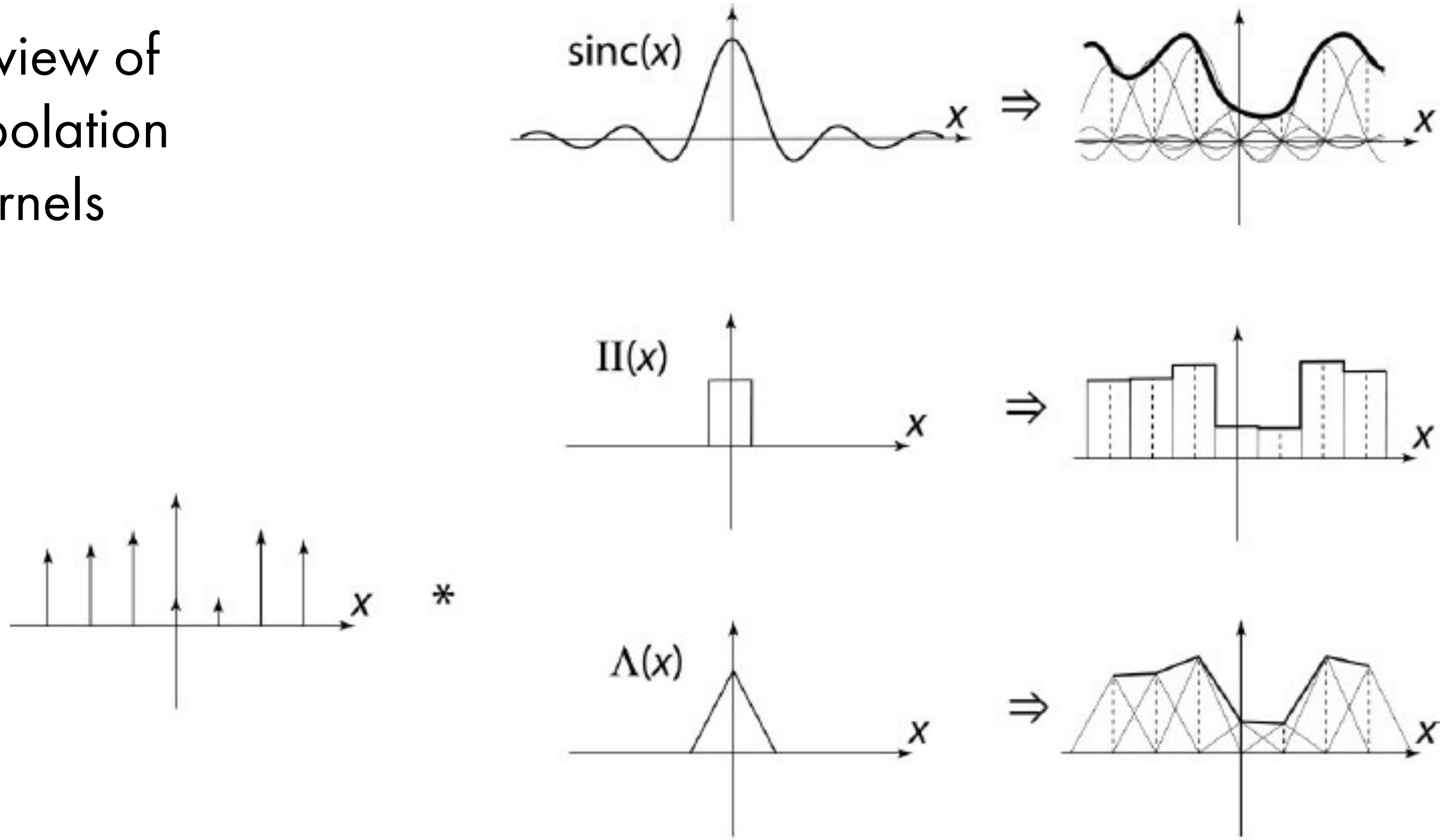
"Ideal" reconstruction



Nearest-Neighbor Interpolation

Interpolation via Convolution (1D)

overview of interpolation kernels

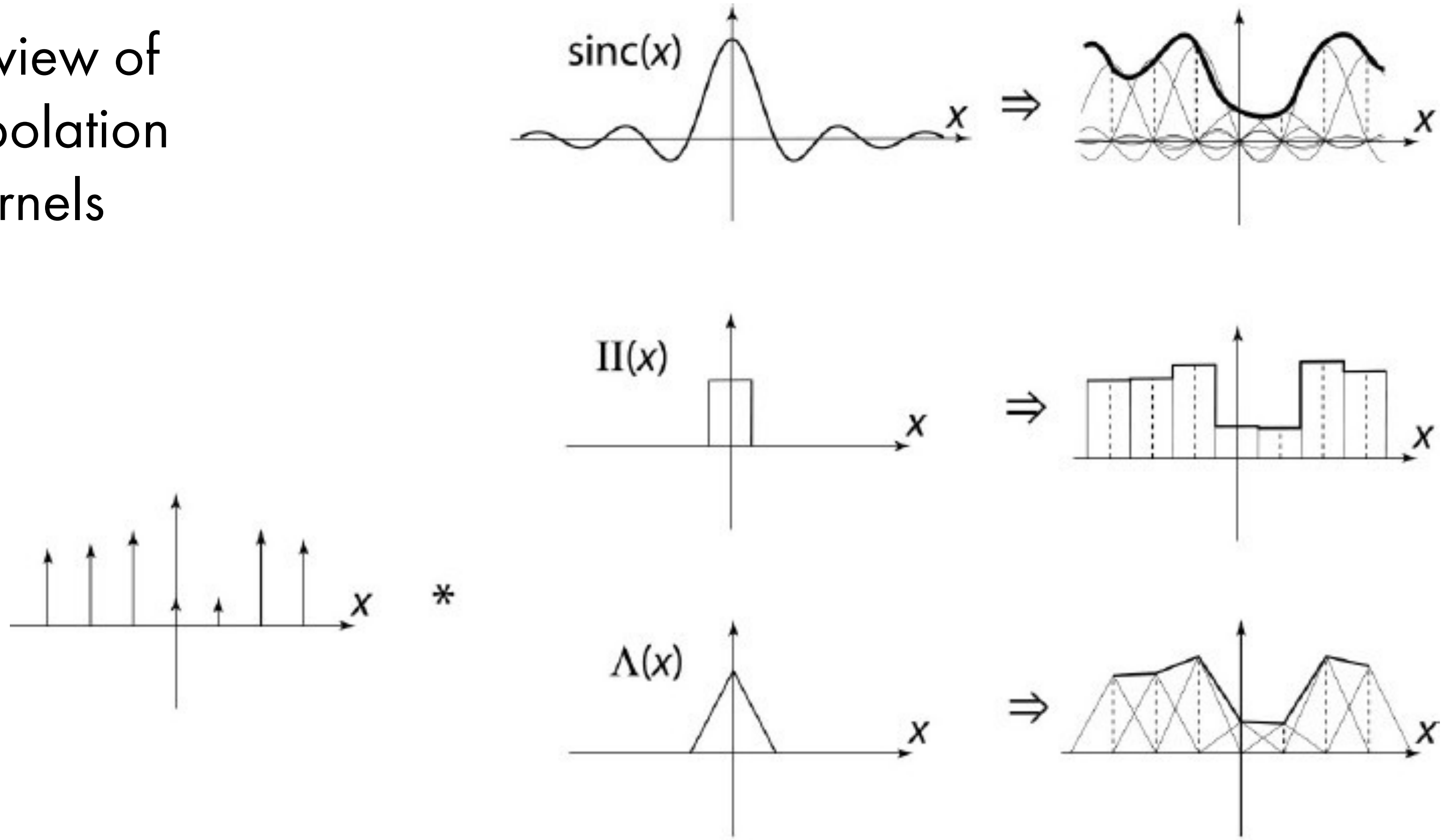


“Ideal” reconstruction

Nearest-Neighbor Interpolation

Interpolation via Convolution (1D)

overview of interpolation kernels



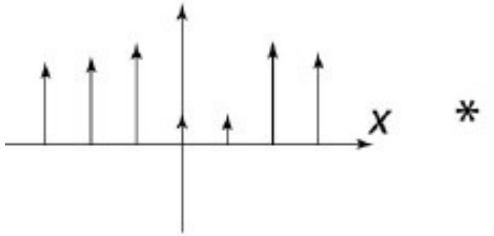
"Ideal" reconstruction

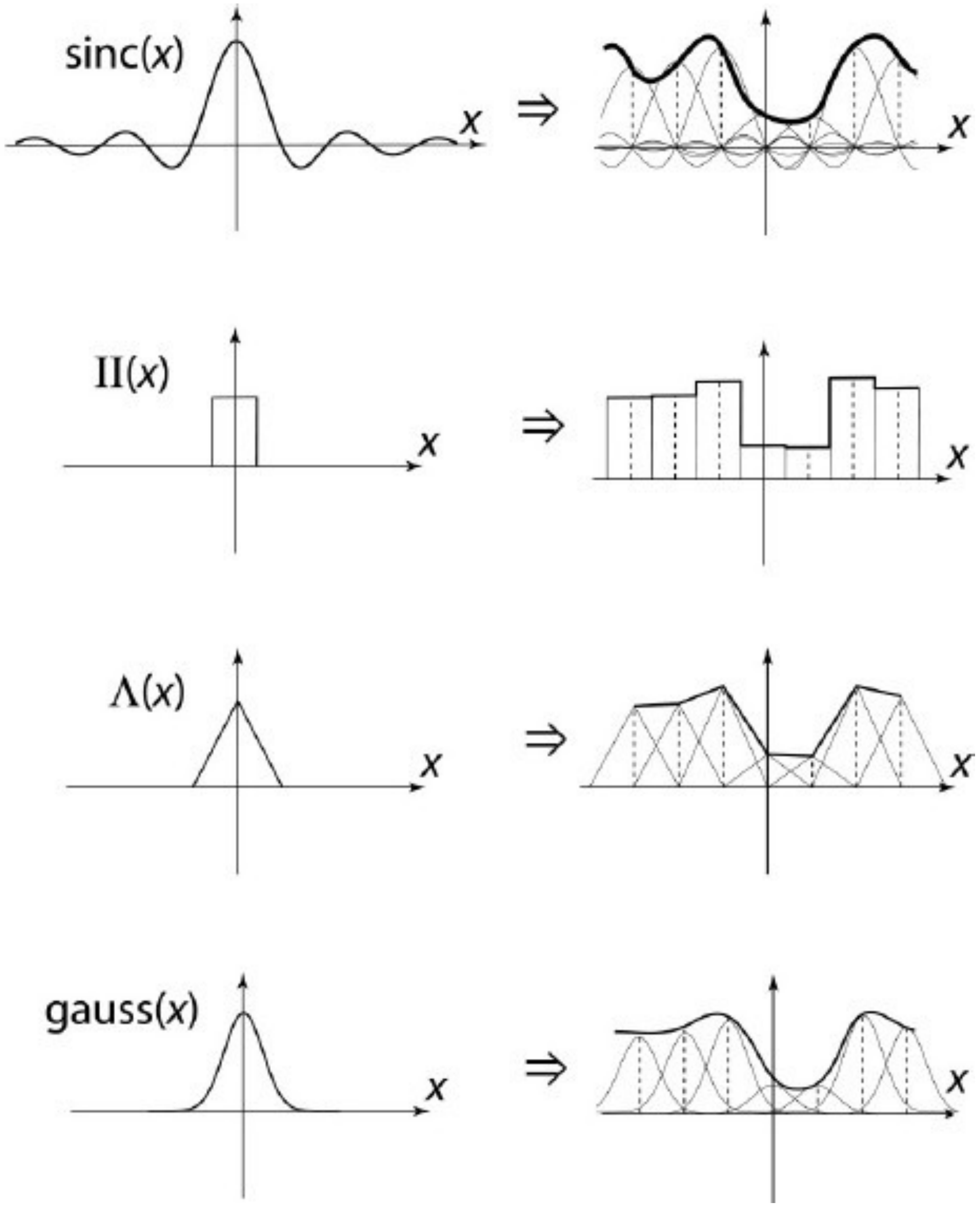
Nearest-Neighbor Interpolation

Linear Interpolation

Interpolation via Convolution (1D)

overview of interpolation kernels





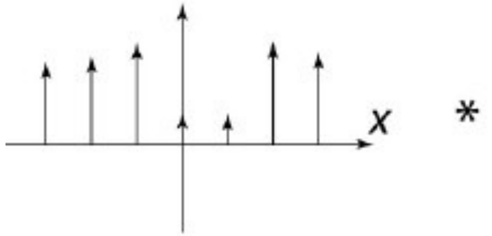
“Ideal” reconstruction

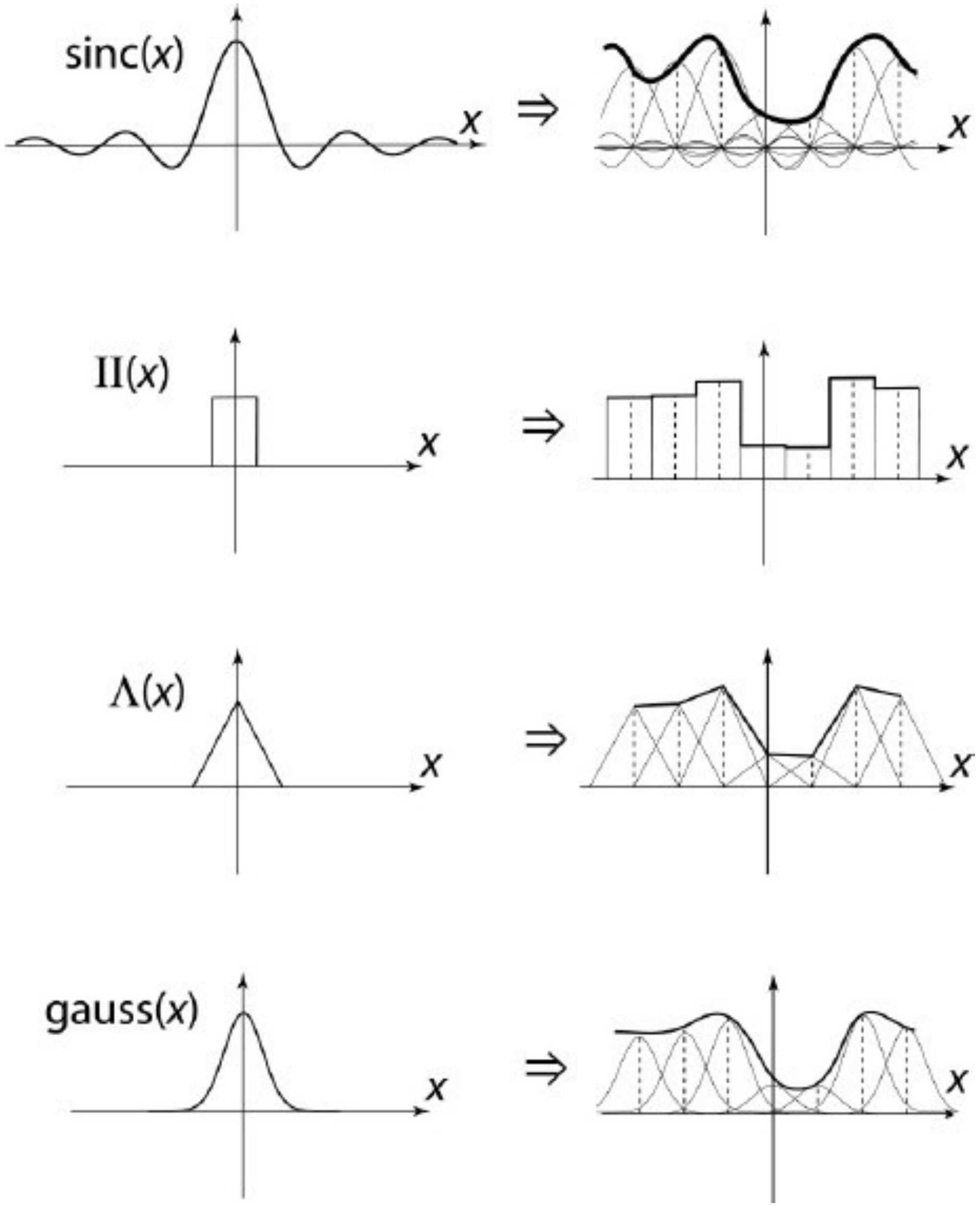
Nearest-Neighbor Interpolation

Linear Interpolation

Interpolation via Convolution (1D)

overview of
interpolation
kernels





"Ideal" reconstruction

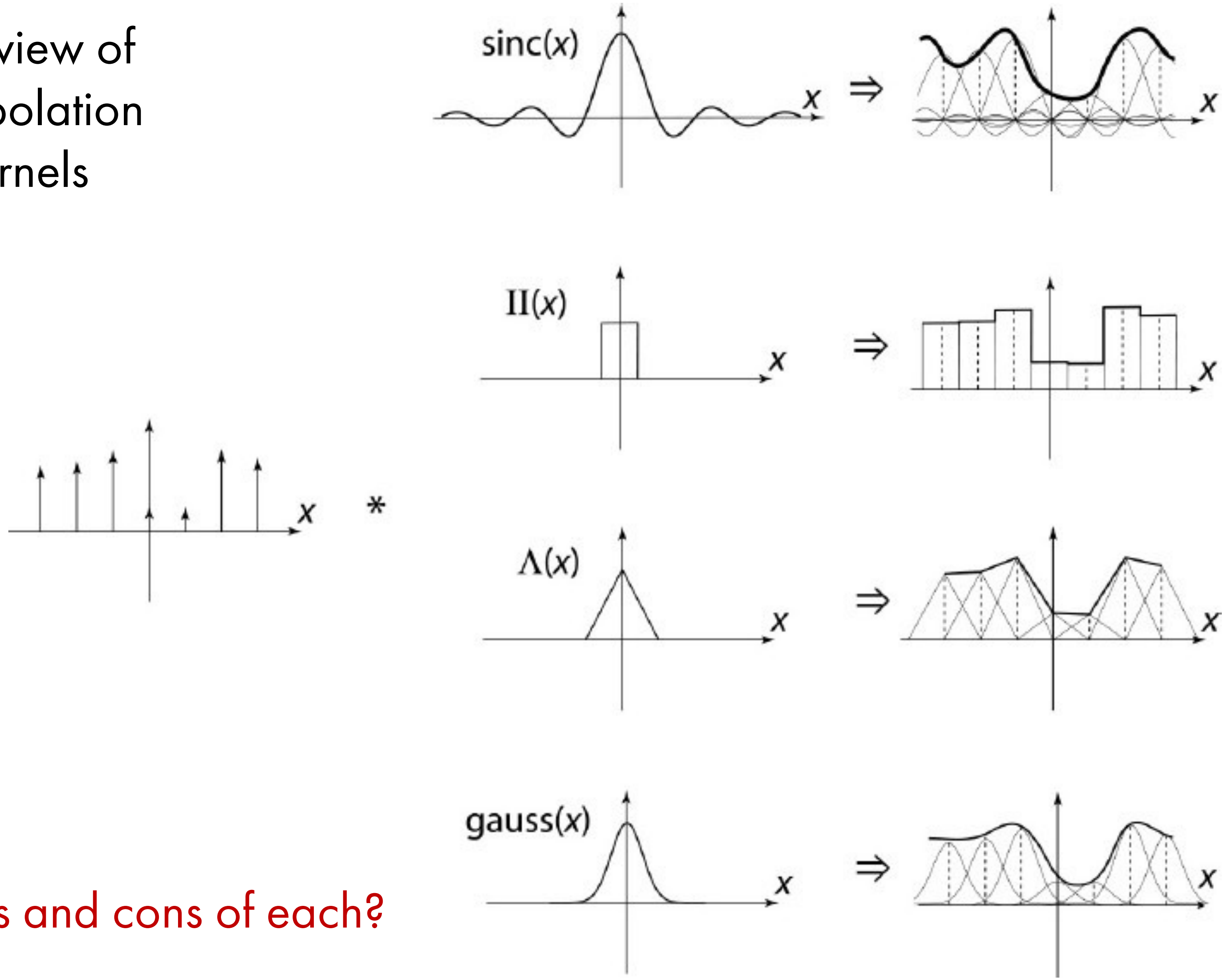
Nearest-Neighbor Interpolation

Linear Interpolation

Gaussian reconstruction

Interpolation via Convolution (1D)

overview of interpolation kernels



“Ideal” reconstruction

Nearest-Neighbor Interpolation

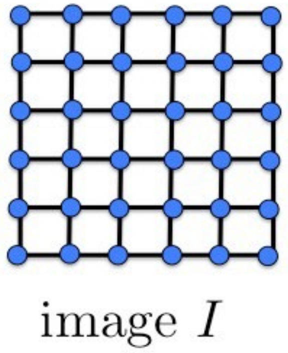
Linear Interpolation

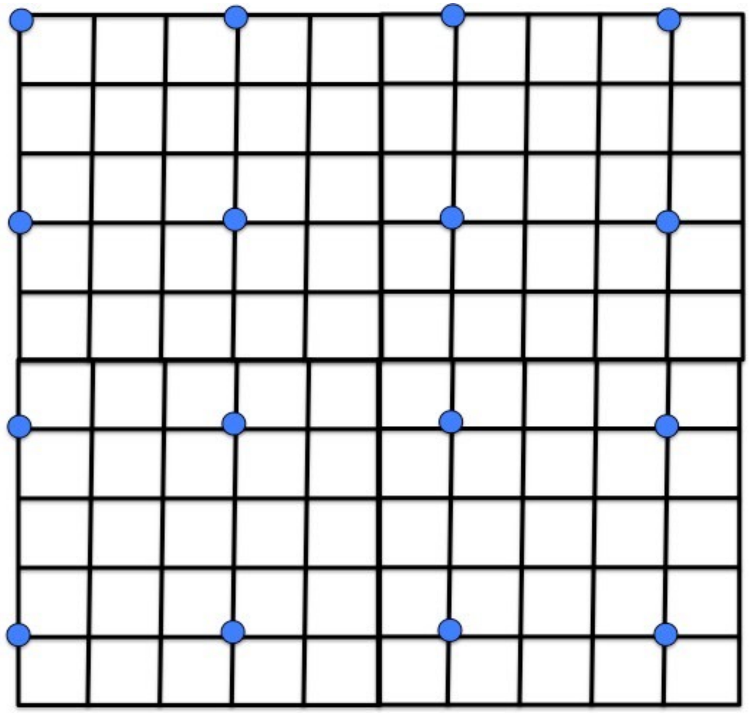
Gaussian reconstruction

pros and cons of each?

Image Interpolation (2D)

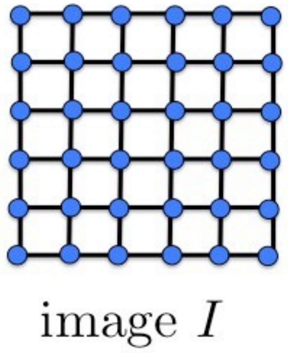
Image Interpolation (2D)

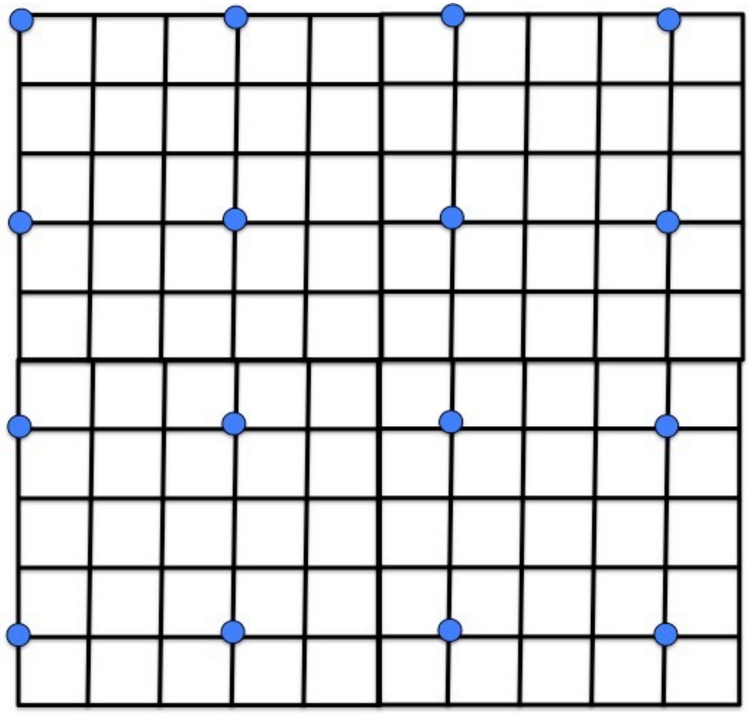




- Let's make this image triple size

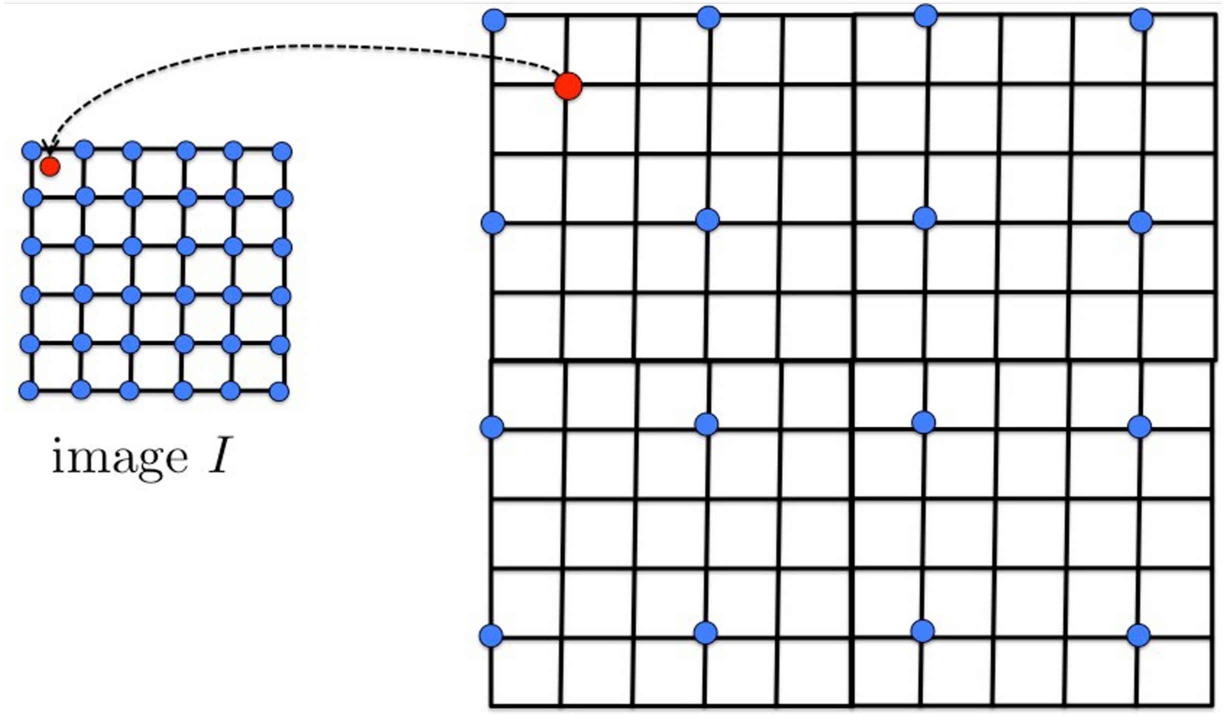
Image Interpolation (2D)





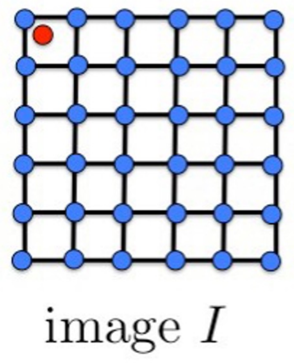
- Let's make this image triple size
- Copy image in every third pixel.

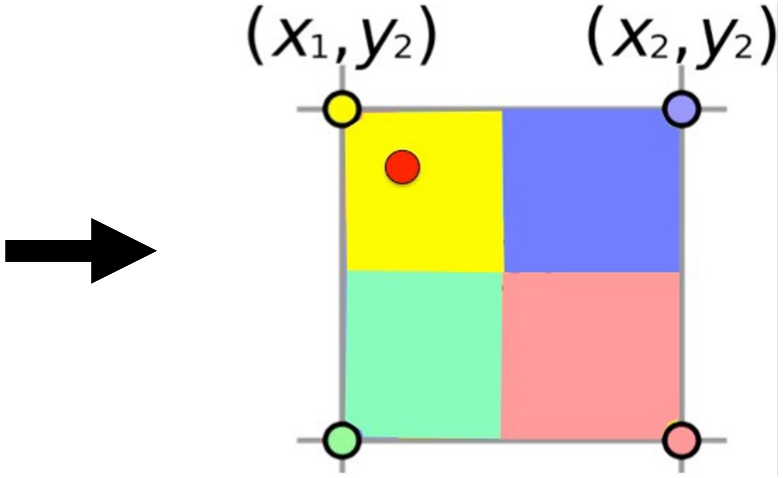
Image Interpolation (2D)



- Let's make this image triple size
- Copy image in every third pixel.
- What about the remaining pixels in G? (how would we compute this value?)

Image Interpolation (2D)



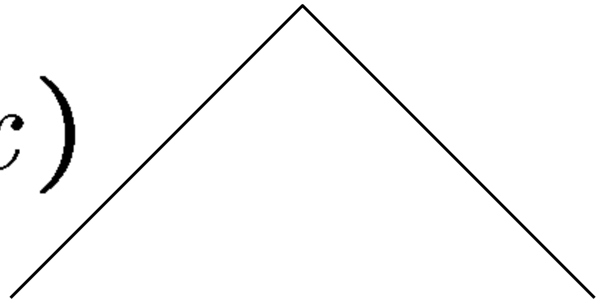


- Let's make this image triple size
- Copy image in every third pixel.
- What about the remaining pixels in G? (how would we compute this value?)
    - *bilinear interpolation (linear interpolation in x and y, resulting in quadratic interpolation)*
    - Check out details: http://en.wikipedia.org/wiki/Bilinear_interpolation

Reconstruction Filters

- What does the 2D version of this hat function look like?

$h(x)$



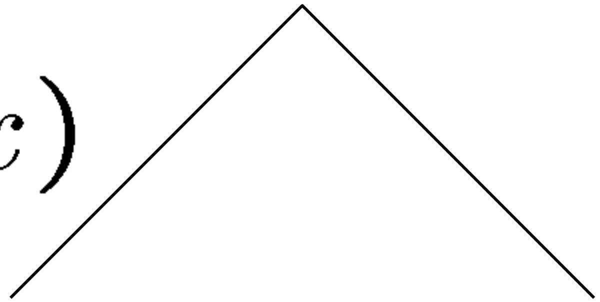
Performs Linear Interpolation

$h(x, y)$?

Reconstruction Filters

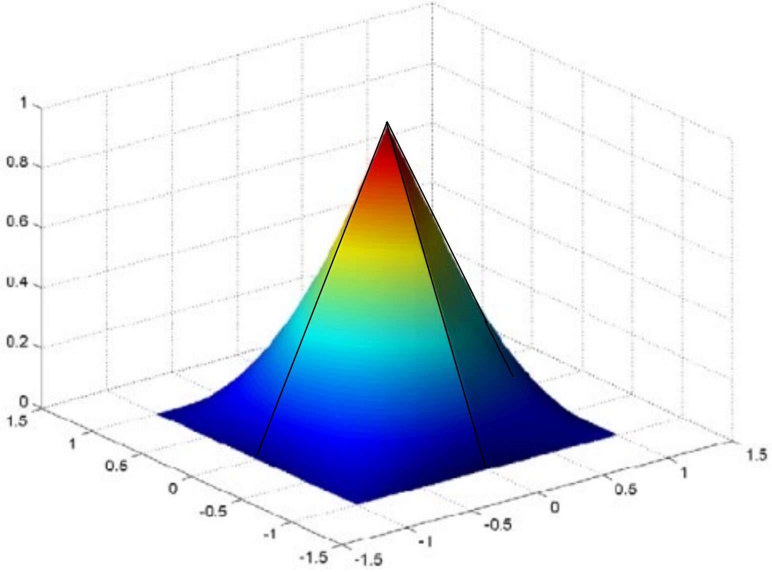
- What does the 2D version of this hat function look like?

$h(x)$



Performs Linear Interpolation

$h(x, y)$

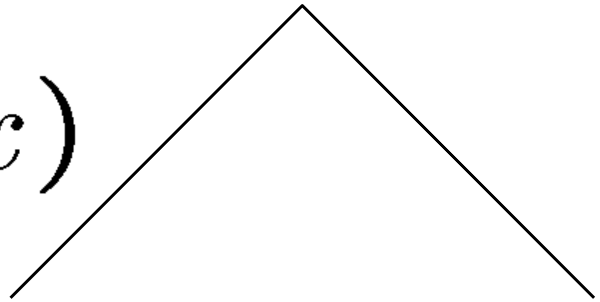


(tent function) **Performs bilinear Interpolation**

Reconstruction Filters

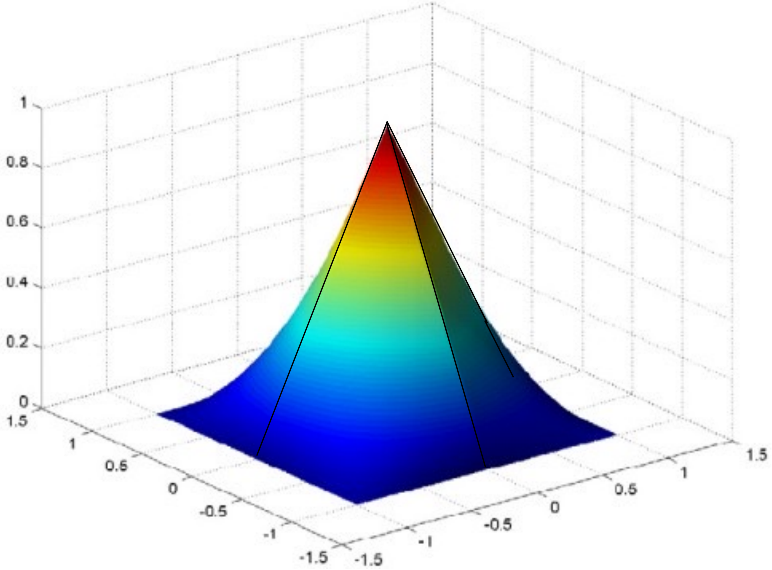
- What does the 2D version of this hat function look like?

$h(x)$



Performs Linear Interpolation

$h(x, y)$



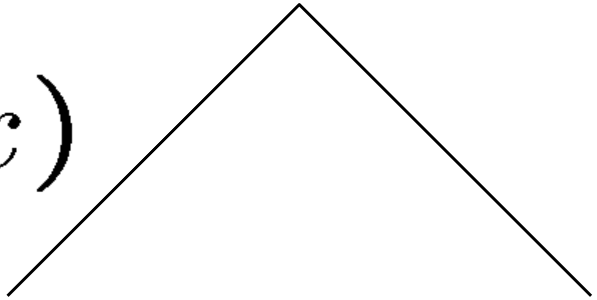
(tent function) **Performs bilinear Interpolation**

- And filter for nearest neighbor interpolation?

Reconstruction Filters

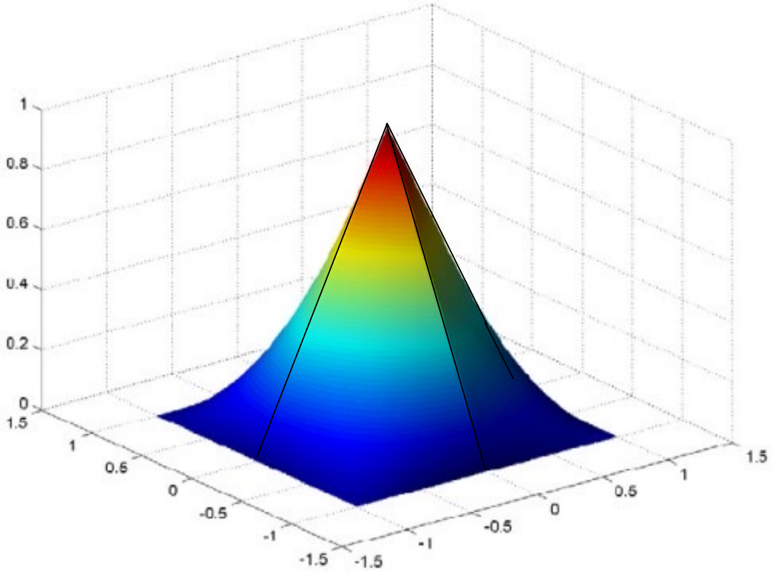
- What does the 2D version of this hat function look like?

$h(x)$



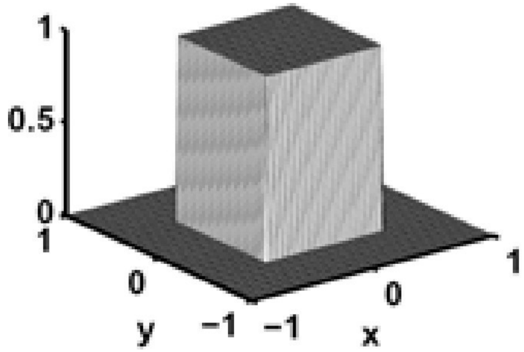
Performs Linear Interpolation

$h(x, y)$



(tent function) **Performs bilinear Interpolation**

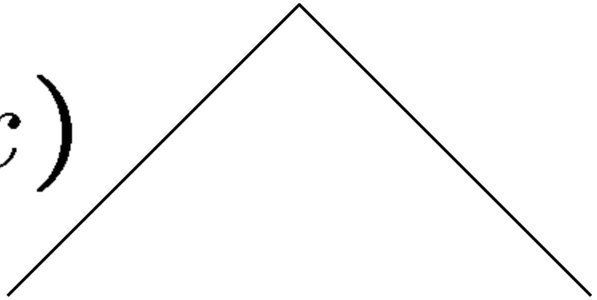
- And filter for nearest neighbor interpolation?



Reconstruction Filters

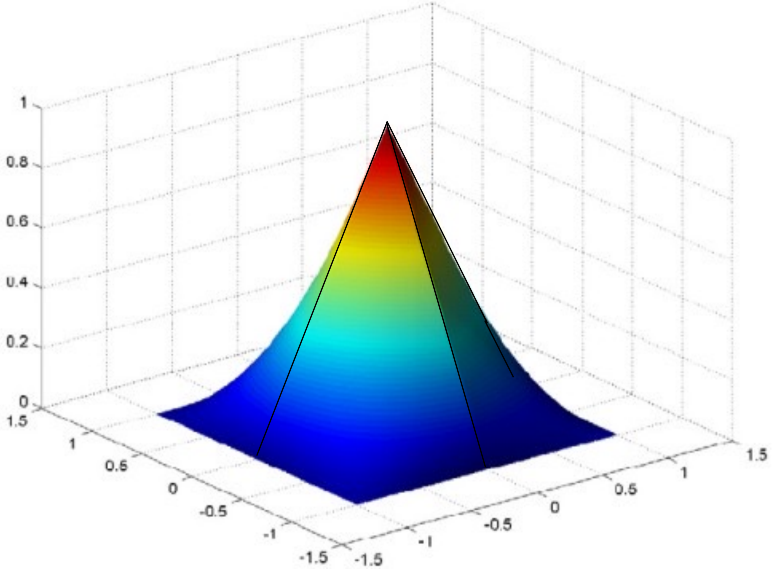
- What does the 2D version of this hat function look like?

$h(x)$



Performs Linear Interpolation

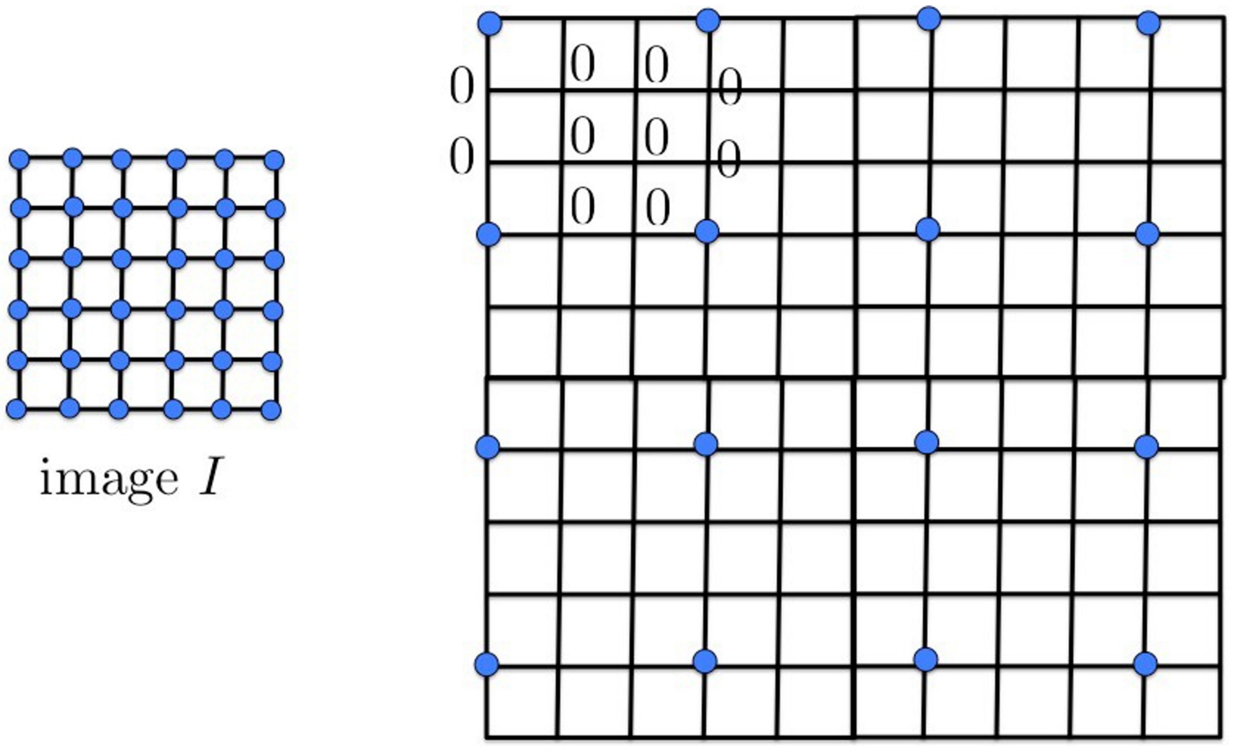
$h(x, y)$



(tent function) **Performs bilinear Interpolation**

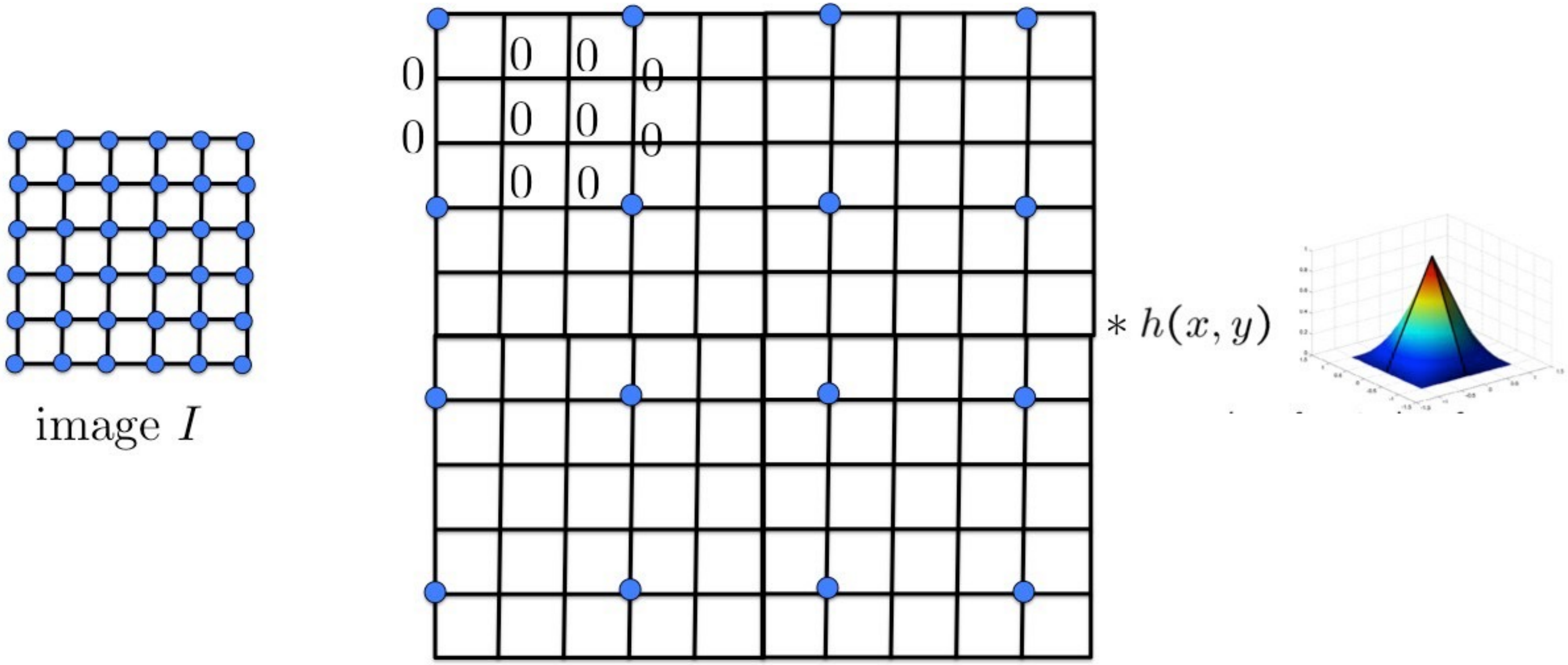
- higher order filters can give better resampled images: Bicubic is a common choice

Image Interpolation via Convolution (2D)



- Let’s make this image triple size: copy image values in every third pixel, place zeros everywhere else
- what filter do we use?

Image Interpolation via Convolution (2D)



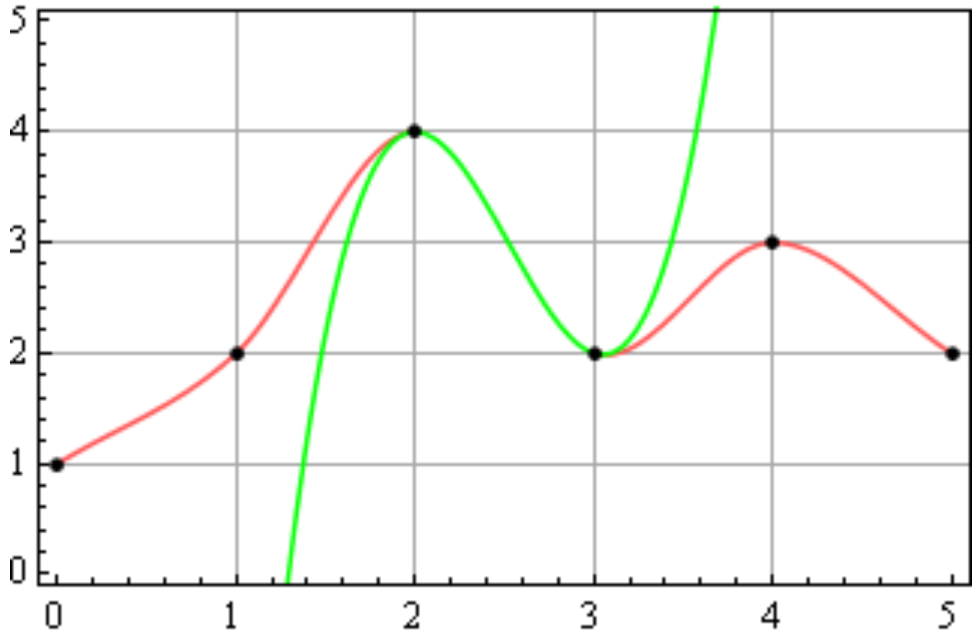
- Let's make this image triple size: copy image values in every third pixel, place zeros everywhere else
- Convolution with a reconstruction filter (e.g., bilinear) and you get the interpolated image

Cubic interpolation

- Uses third degree polynomial

$$f(x) = ax^3 + bx^2 + cx + d$$

$$f'(x) = 3ax^2 + 2bx + c$$



https://www.paulinternet.nl/?page=bicubic

Cubic interpolation

- Uses third degree polynomial

$$f(x) = ax^3 + bx^2 + cx + d$$

$$f'(x) = 3ax^2 + 2bx + c$$

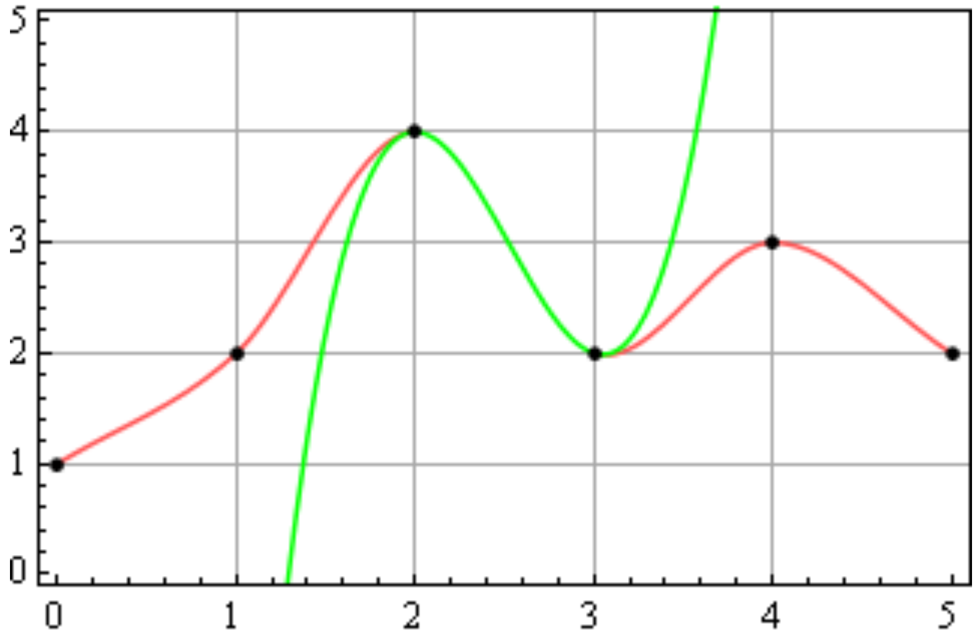
- plug in values at x=0 and x=1(pick two points)

$$f(0) = d$$

$$f(1) = a + b + c + d$$

$$f'(0) = c$$

$$f'(1) = 3a + 2b + c$$



https://www.paulinternet.nl/?page=bicubic

Cubic interpolation

- Uses third degree polynomial

$$f(x) = ax^3 + bx^2 + cx + d$$

$$f'(x) = 3ax^2 + 2bx + c$$

- plug in values at x=0 and x=1(pick two points)

$$f(0) = d$$

$$f(1) = a + b + c + d$$

$$f'(0) = c$$

$$f'(1) = 3a + 2b + c$$

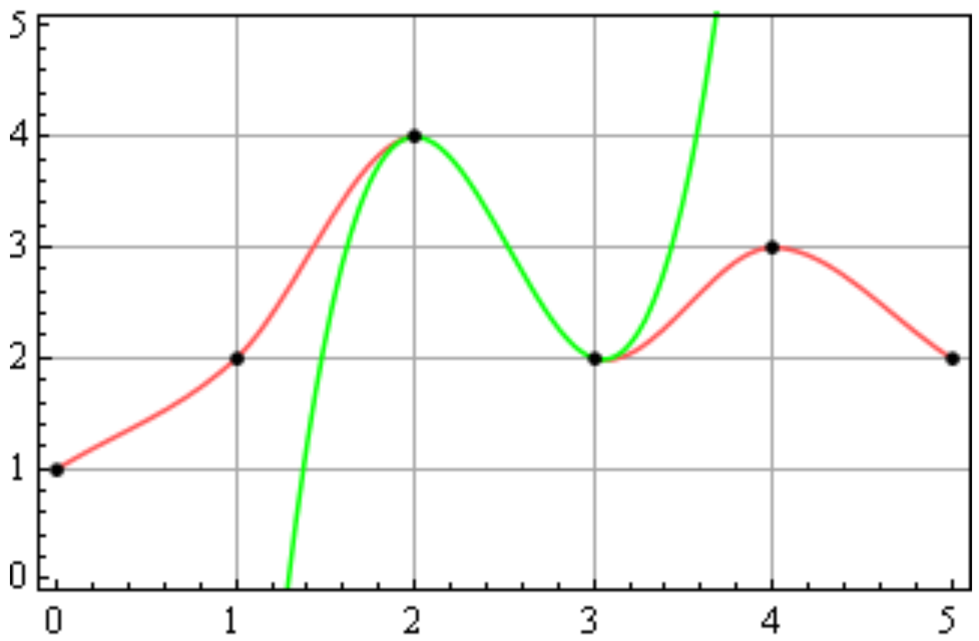
- rewrite the equations as

$$a = 2f(0) - 2f(1) + f'(0) + f'(1)$$

$$b = -3f(0) + 3f(1) - 2f'(0) - f'(1)$$

$$c = f'(0)$$

$$d = f(0)$$

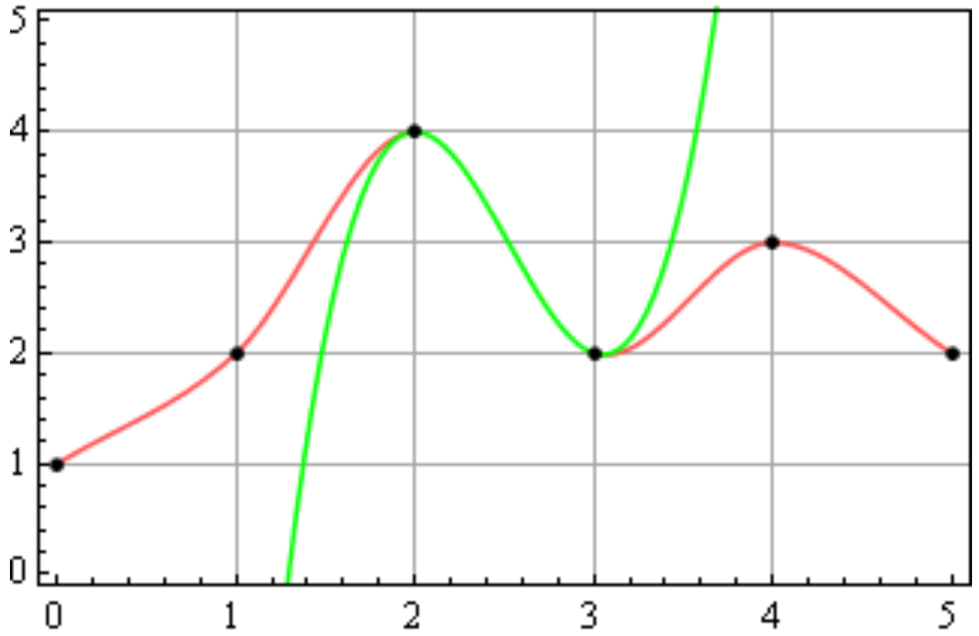


https://www.paulinternet.nl/?page=bicubic

Cubic interpolation

- rewrite the equations as

$$a = 2f(0) - 2f(1) + f'(0) + f'(1)$$
$$b = -3f(0) + 3f(1) - 2f'(0) - f'(1)$$
$$c = f'(0)$$
$$d = f(0)$$



- plug in known values for *f*

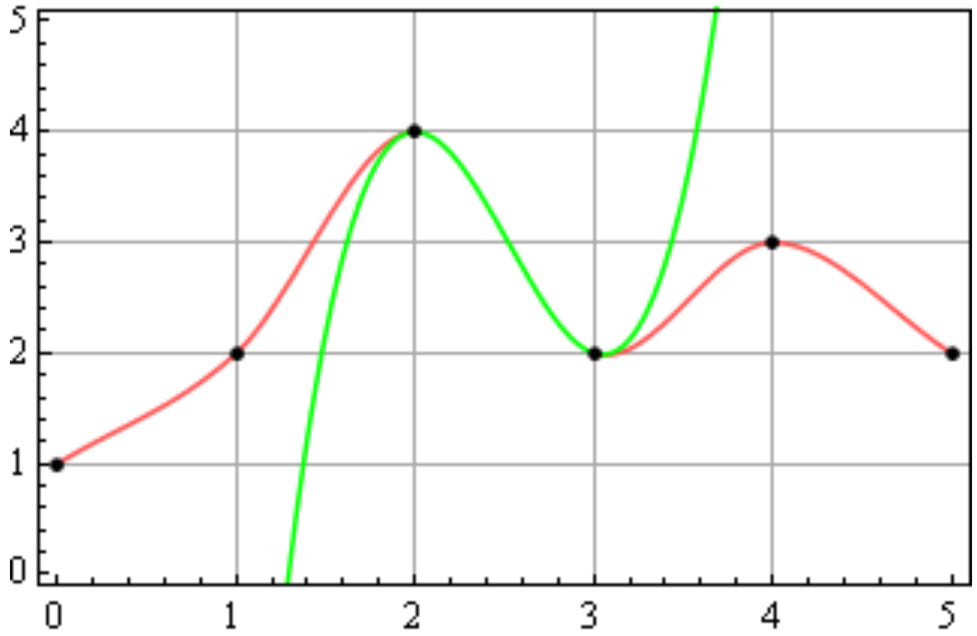
$$f(0) = p_1$$
$$f(1) = p_2$$
$$f'(0) = \frac{p_2 - p_0}{2}$$
$$f'(1) = \frac{p_3 - p_1}{2}$$

- plug back in!

$$a = -\tfrac{1}{2}p_0 + \tfrac{3}{2}p_1 - \tfrac{3}{2}p_2 + \tfrac{1}{2}p_3 \qquad b = p_0 - \tfrac{5}{2}p_1 + 2p_2 - \tfrac{1}{2}p_3$$
$$c = -\tfrac{1}{2}p_0 + \tfrac{1}{2}p_2 \qquad d = p_1$$

https://www.paulinternet.nl/?page=bicubic

Cubic interpolation



The whole formula looks like:

$$f(p_0, p_1, p_2, p_3, x) = \left(-\frac{1}{2}p_0 + \frac{3}{2}p_1 - \frac{3}{2}p_2 + \frac{1}{2}p_3\right)x^3 + \left(p_0 - \frac{5}{2}p_1 + 2p_2 - \frac{1}{2}p_3\right)x^2 + \left(-\frac{1}{2}p_0 + \frac{1}{2}p_2\right)x + p_1$$

Smoother interpolation between points (can constrain both values and derivatives at each point)

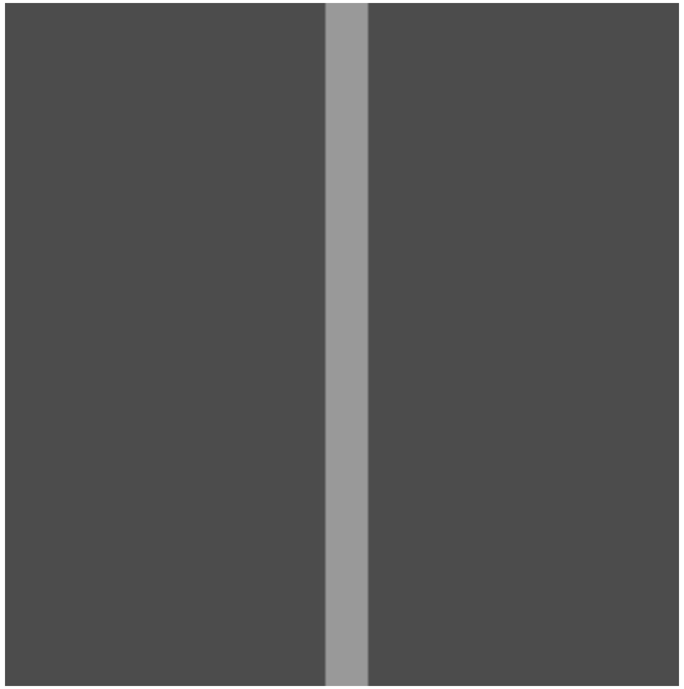
https://www.paulinternet.nl/?page=bicubic

Bicubic interpolation

- What happens to the global minimum/maximum of discrete samples after cubic interpolation?

Bicubic interpolation

- What happens to the global minimum/maximum of discrete samples after cubic interpolation?
- Nearest neighbor



Input image

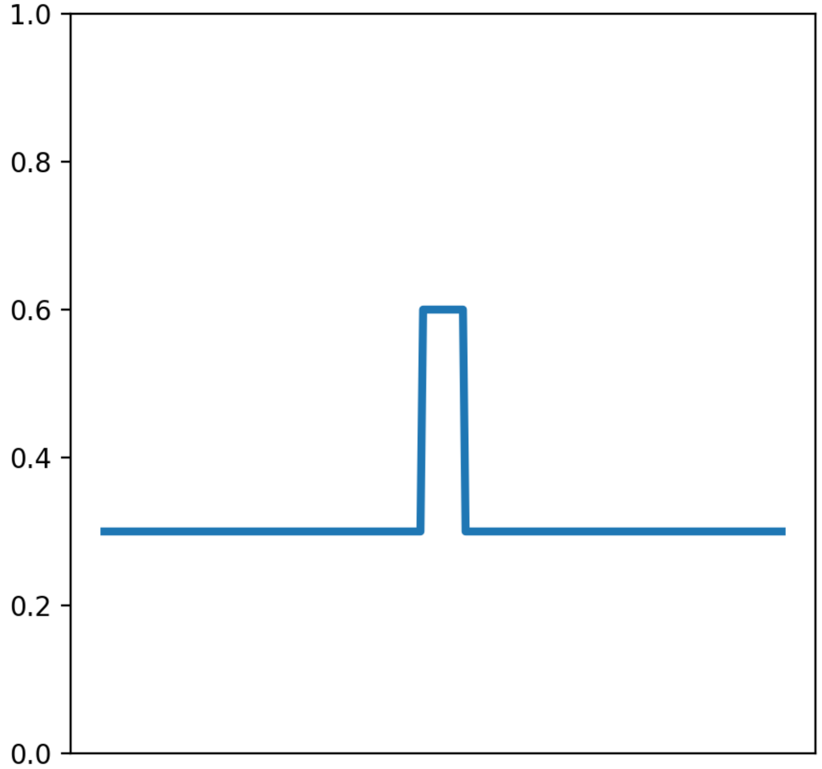


image slice

https://www.paulinternet.nl/?page=bicubic

Bicubic interpolation

- What happens to the global minimum/maximum of discrete samples after cubic interpolation?
- Bilinear interpolation

Input image

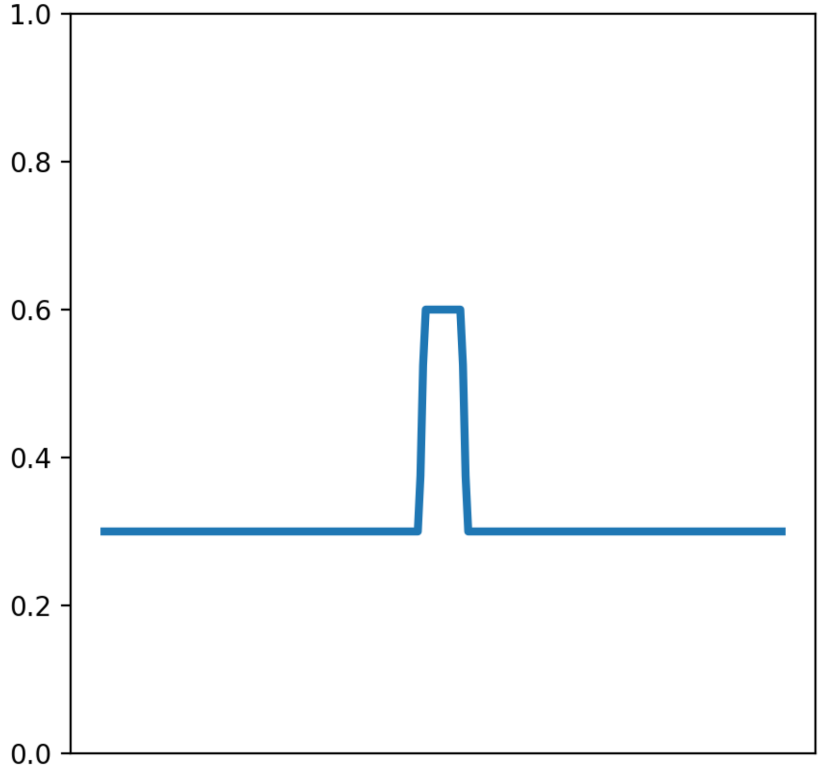
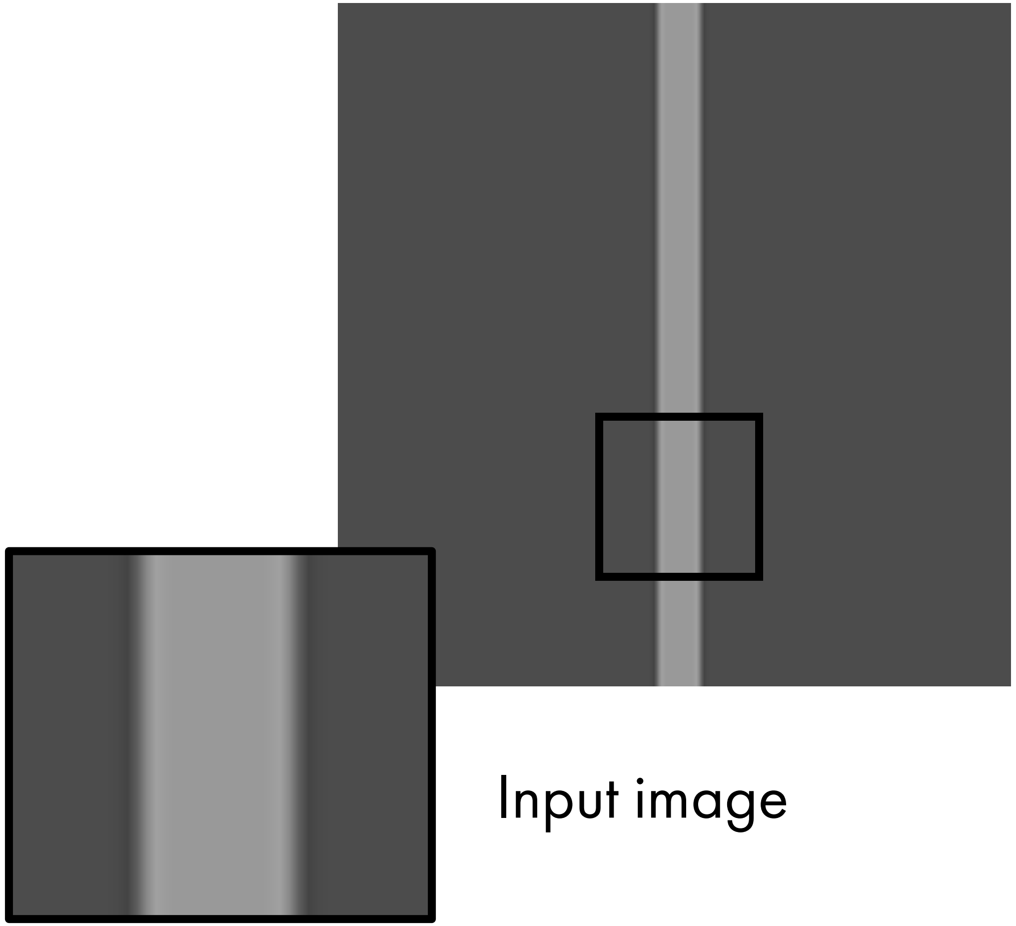


image slice

https://www.paulinternet.nl/?page=bicubic

Bicubic interpolation

- What happens to the global minimum/maximum of discrete samples after cubic interpolation?
- Bicubic interpolation



Input image

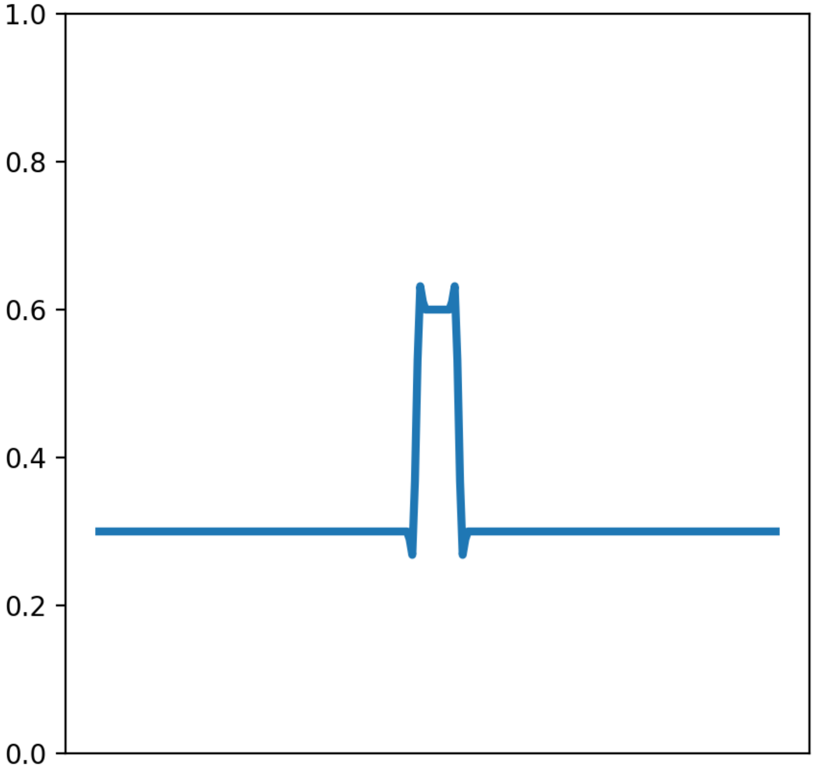


image slice

https://www.paulinternet.nl/?page=bicubic

Image Interpolation

Original image

Nearest-Neighbor Interpolation

Bilinear interpolation

Bicubic interpolation

[Source: N. Snavely]

Summary – Stuff You Should Know

- To down-scale an image: blur it with a small Gaussian (e.g., σ = 1.4) and downsample
- What is aliasing in the Fourier domain? What is anti-aliasing?
- To up-scale an image: interpolation (nearest neighbor, bilinear, bicubic, etc)
- Gaussian pyramid: Blur with Gaussian filter, downsample result by factor 2, blur it with the Gaussian, downsample by 2...

<u>OpenCV functions</u>:

- imresize: with interpolation options INTER_LINEAR, INTER_CUBIC, ...
- pyrDown: Blurs an image and downsamples it (1/2 size)
- pyrUp: Upsamples an image and then blurs it ($\times$2 size)
- img[::2,::2,:]: takes every second row and column