Image Pyramids

Finding Waldo

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

image

template (filter)

Finding Waldo

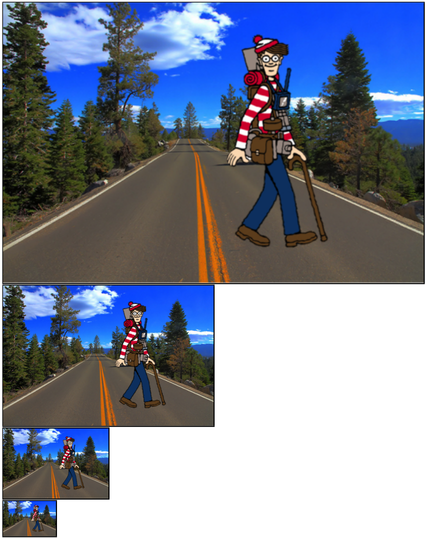
- He comes closer but our filter doesn't know that
- How can we find Waldo?

image

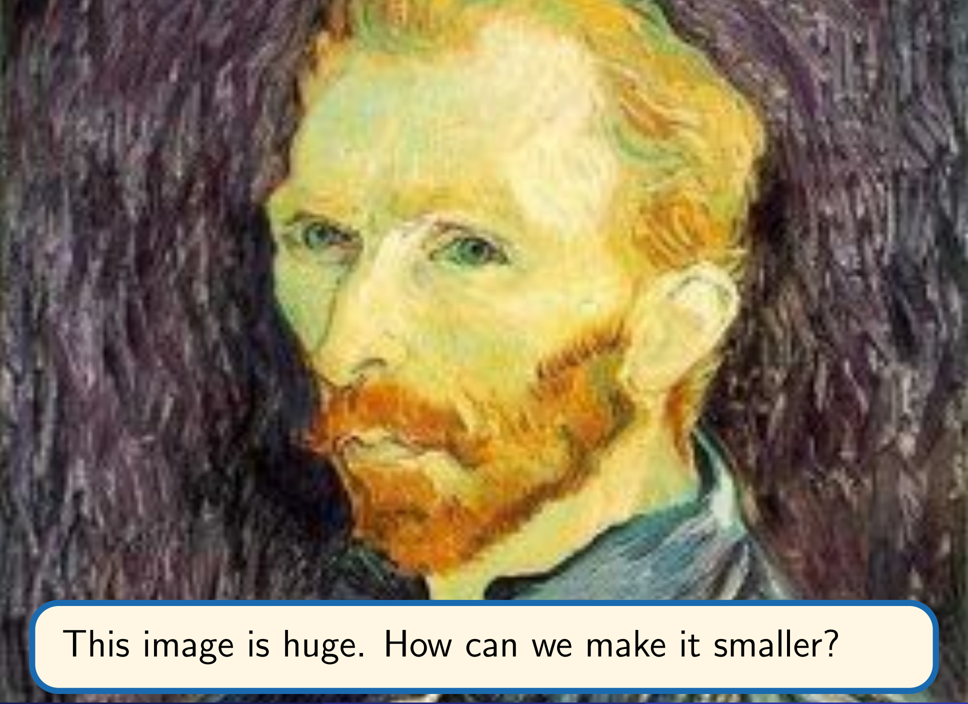
template (filter)

Idea: Re-size Image

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



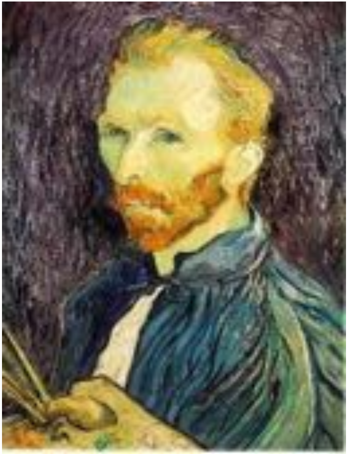
template (filter)



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



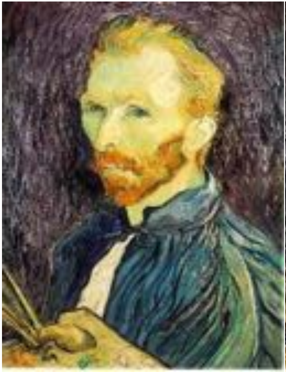
1/4

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)

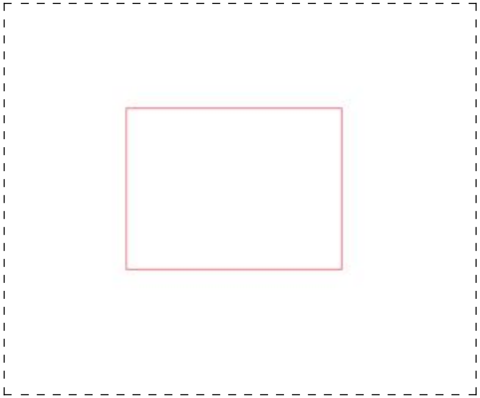


Figure: Dashed line denotes the border of the image (it's not part of the image)

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

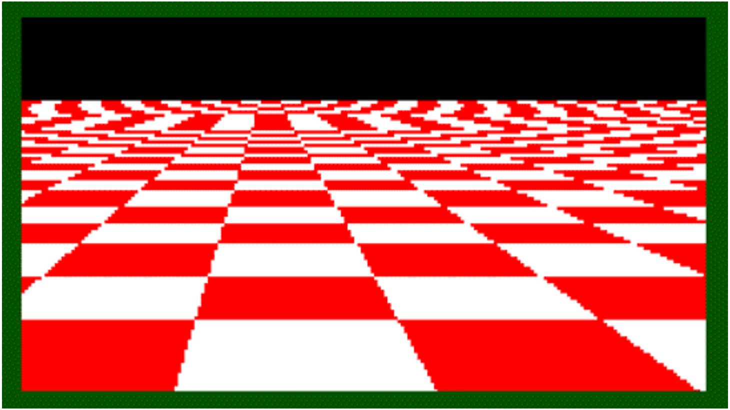
- What's in the image?
- Now I want to resize my image by half in the width direction
- And I take every other column (1st, 3rd, 5th, etc)
- Where is the chicken!

Image Sub-Sampling


[Source: F. Durand]

Even worse for synthetic images

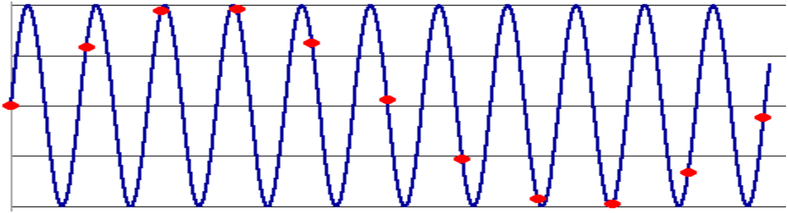
- What's happening?



[Source: L. Zhang]

Aliasing

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

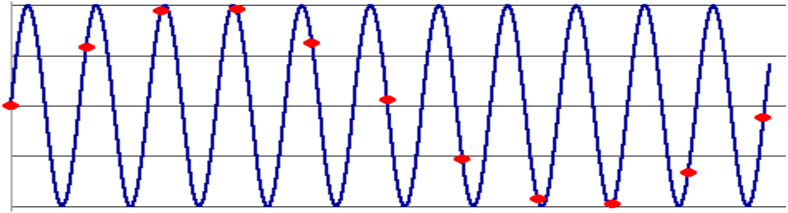


- To do sampling right, need to understand the structure of your signal/image

[Source: R. Urtasun]

Aliasing

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

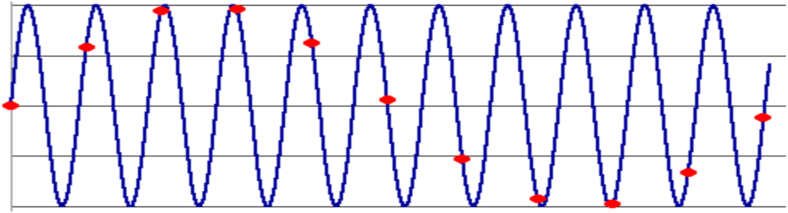


- To do sampling right, need to understand the structure of your signal/image
- The minimum sampling rate is called the **Nyquist rate**

[Source: R. Urtasun]

Aliasing

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



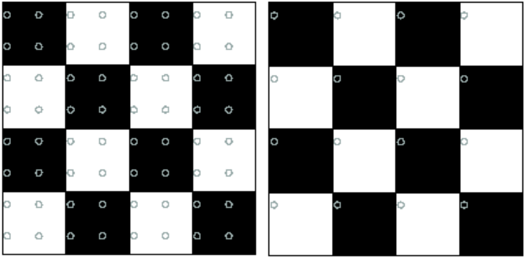
- To do sampling right, need to understand the structure of your signal/image
- The minimum sampling rate is called the **Nyquist rate**

[Source: R. Urtasun]

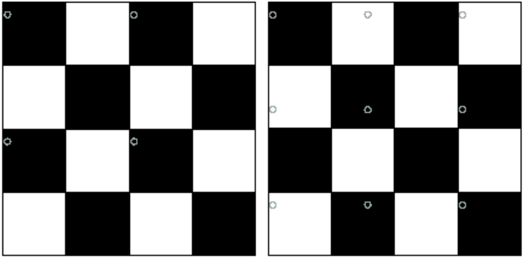
Mr. Nyquist

- 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`
- He looks like a smart guy, we'll just believe him

2D example



Good sampling



Bad sampling

[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]

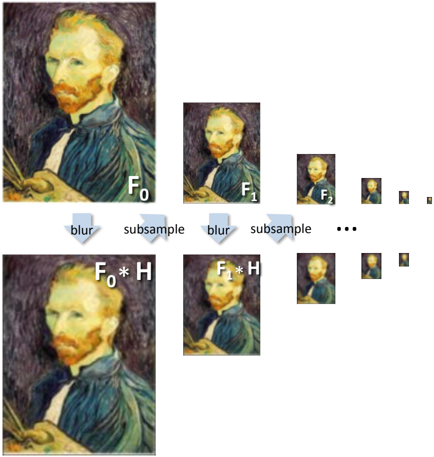
Going back to Downsampling ...

- 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]

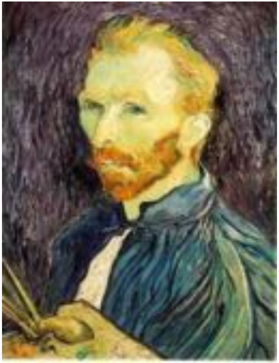
Gaussian pre-filtering

- Solution: Blur the image via Gaussian, then subsample. Very simple!



[Source: N. Snavely]

Subsampling with Gaussian pre-filtering



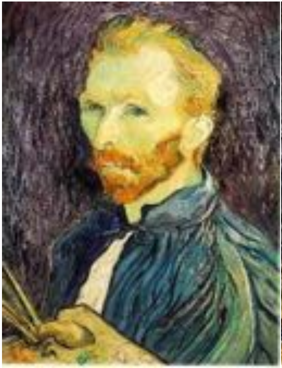
Gaussian 1/2

G 1/4

G 1/8

[Source: S. Seitz]

Compare with ...



1/2

1/4 (2x zoom)

1/8 (4x zoom)

[Source: S. Seitz]

Where is the Rectangle?

- My image

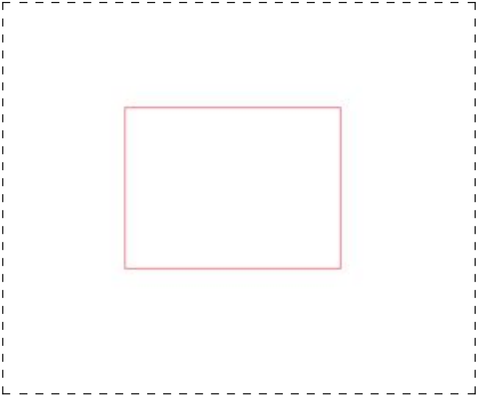


Figure: Dashed line denotes the border of the image (it's not part of the image)

Where is the Rectangle?

- My image
- Let's blur

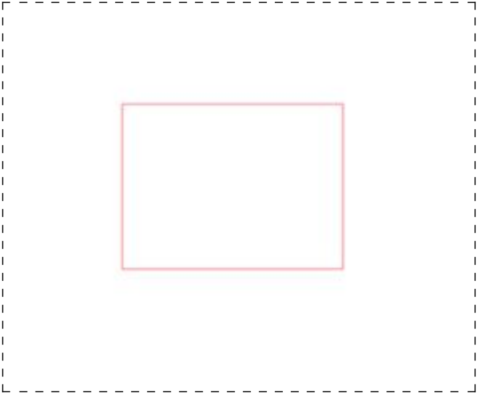


Figure: Dashed line denotes the border of the image (it's not part of the image)

Where is the Rectangle?

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

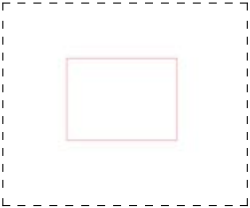


Figure: Dashed line denotes the border of the image (it's not part of the image)

Where is the Chicken?

- My image

Where is the Chicken?

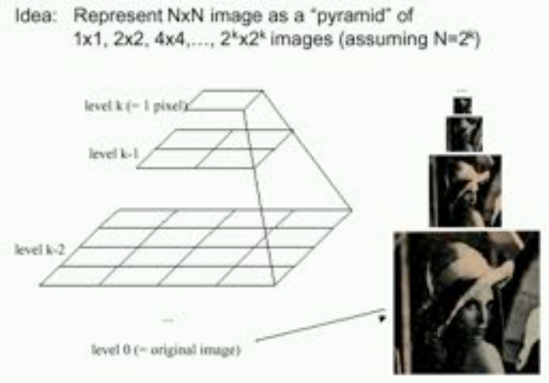
- My image
- Let's blur

Where is the Chicken?

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

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]

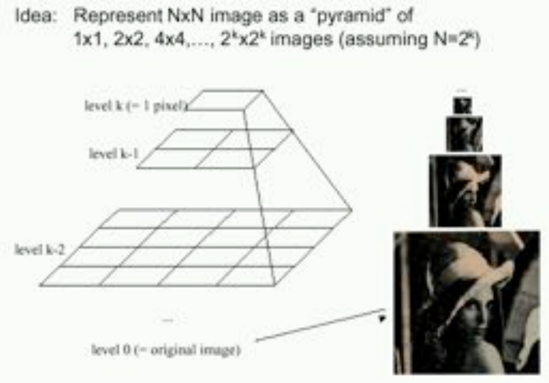


- How much space does a Gaussian pyramid take compared to original image?

[Source: S. Seitz]

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]



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

[Source: S. Seitz]

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]

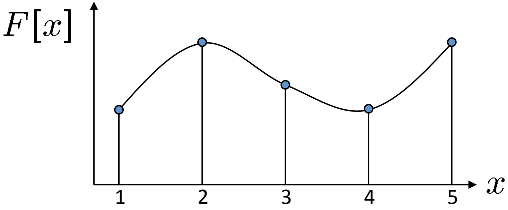
Image Up-Sampling

- 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



d = 1 in this example

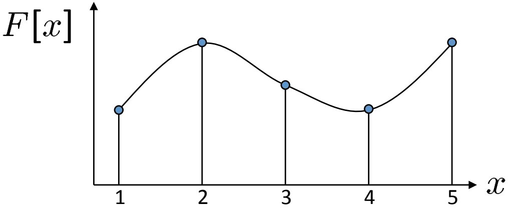
Recall how a digital image is formed

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

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

[Source: N. Snavely, S. Seitz]

Interpolation



d = 1 in this example

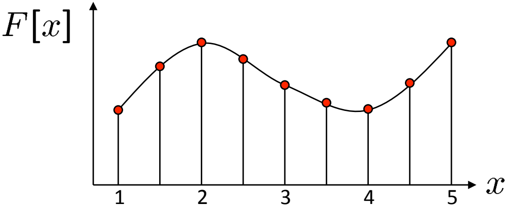
Recall how a digital image is formed

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

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

[Source: N. Snavely, S. Seitz]

Interpolation



d = 1 in this example

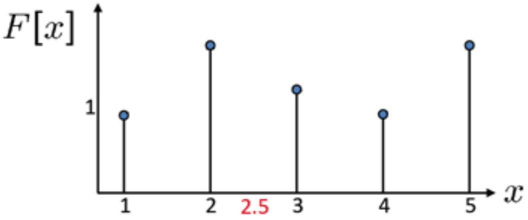
Recall how a digital image is formed

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

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

[Source: N. Snavely, S. Seitz]

Interpolation

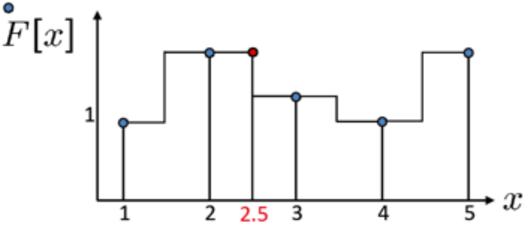


d = 1 in this example

What if we don't know f?

[Source: N. Snavely, S. Seitz]

Interpolation



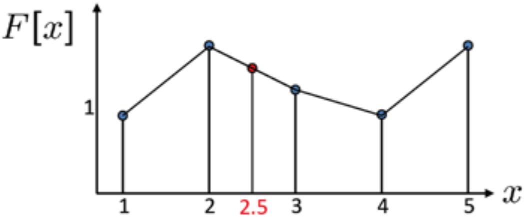
d = 1 in this example

What if we don't know f?

- Guess an approximation: for example nearest-neighbor

[Source: N. Snavely, S. Seitz]

Interpolation



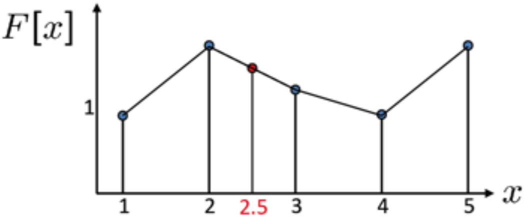
d = 1 in this example

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



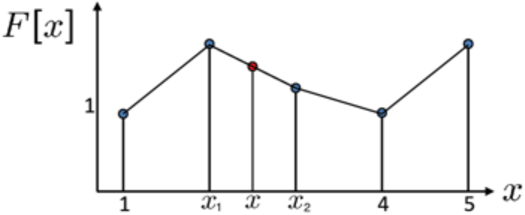
d = 1 in this example

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

[Source: N. Snavely, S. Seitz]

Linear Interpolation



d = 1 in this example

- Linear interpolation:

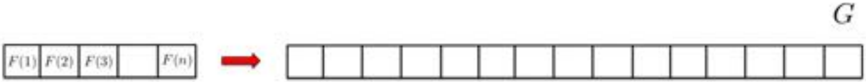
$$G(x) = \frac{x_2 - x}{x_2 - x_1} F(x_1) + \frac{x - x_1}{x_2 - x_1} F(x_2)$$

Interpolation: 1D Example

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

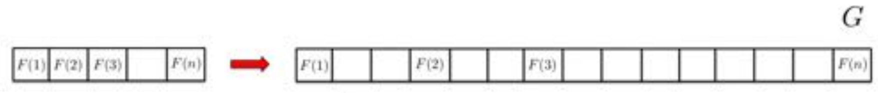
- Let's make this signal triple length

Interpolation: 1D Example



Make a vector G with d times the size of F

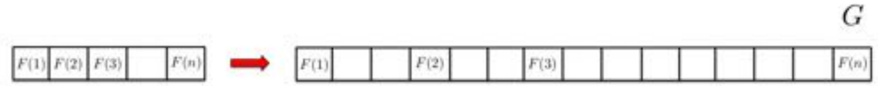
- Let's make this signal triple length ($d = 3$)



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

- Let's make this signal triple length ($d = 3$)
- If i/d is an integer, just copy from the signal

Interpolation: 1D Example



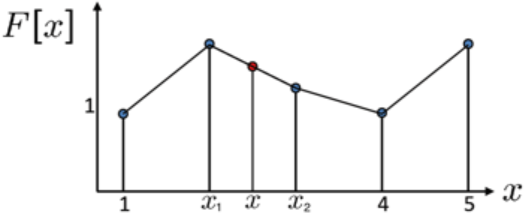
$$\text{if } \frac{i}{d} \text{ integer: } G(i) = F(i/d)$$

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

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

- Let's make this signal triple length ($d = 3$)
- If i/d is an integer, just copy from the signal
- Otherwise use the interpolation formula

Linear Interpolation via Convolution



d = 1 in this example

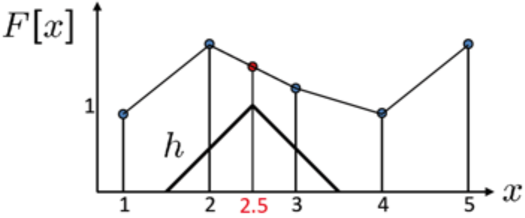
- Linear interpolation:

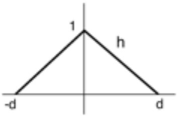
$$G(x) = \frac{x_2 - x}{x_2 - x_1} F(x_1) + \frac{x - x_1}{x_2 - x_1} F(x_2)$$

- With $t = x - x_1$ and $d = x_2 - x_1$ we can get:

$$G(x) = \frac{d - t}{d} F(x - t) + \frac{t}{d} F(x + d - t)$$

Linear Interpolation via Convolution





- Linear interpolation:

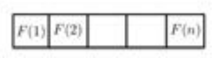
$$G(x) = \frac{x_2 - x}{x_2 - x_1} F(x_1) + \frac{x - x_1}{x_2 - x_1} F(x_2)$$

- With $t = x - x_1$ and $d = x_2 - x_1$ we can get:

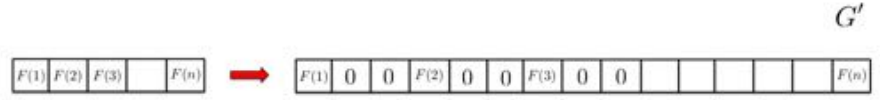
$$G(x) = \frac{d - t}{d} F(x - t) + \frac{t}{d} F(x + d - t)$$

(Kind of looks like convolution: $G(x) = \sum_t h(t)F(x - t)$))

Interpolation via Convolution: 1D Example



- Let's make this signal triple length

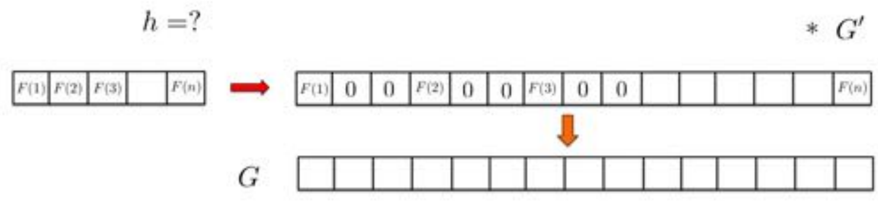


if $\frac{i}{d}$ integer: $G'(i) = F(i/d)$

otherwise: 0

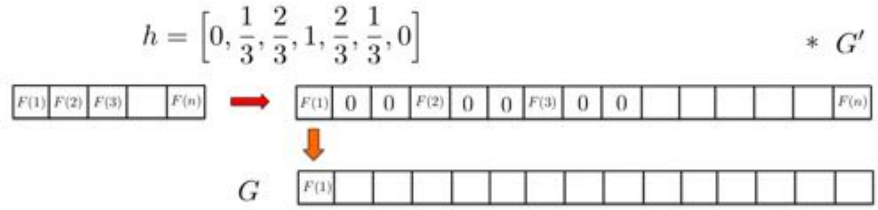
- Let's make this signal triple length ($d = 3$)

Interpolation via Convolution: 1D Example



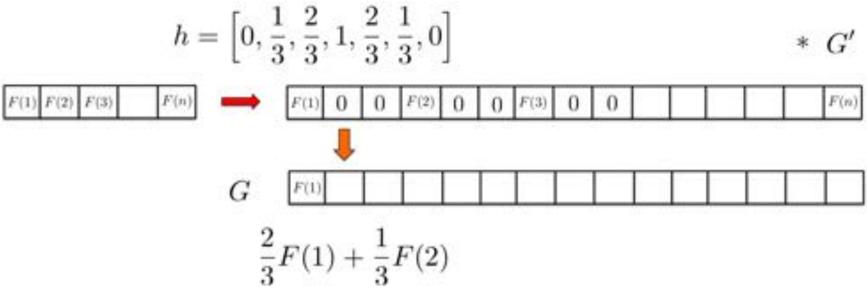
- Let's make this signal triple length ($d = 3$)
- What should be my "reconstruction" filter h (such that $G = h * G'$)?

Interpolation via Convolution: 1D Example



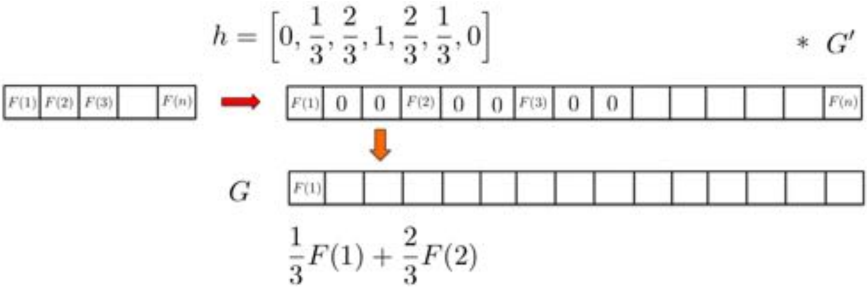
- Let's make this signal triple length ($d = 3$)
- What should be my "reconstruction" filter h (such that $G = h * G'$)?
- $h = [0, \frac{1}{d}, \ldots, \frac{d-1}{d}, 1, \frac{d-1}{d}, \ldots, \frac{1}{d}, 0]$, where d my upsampling factor

Interpolation via Convolution: 1D Example



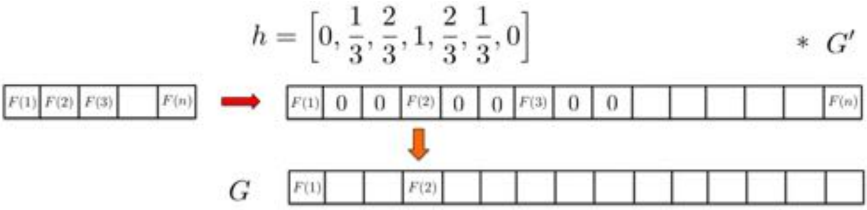
- Let's make this signal triple length ($d = 3$)
- What should be my "reconstruction" filter h (such that $G = h * G'$)?
- $h = [0, \frac{1}{d}, \ldots, \frac{d-1}{d}, 1, \frac{d-1}{d}, \ldots, \frac{1}{d}, 0]$, where d my upsampling factor

Interpolation via Convolution: 1D Example



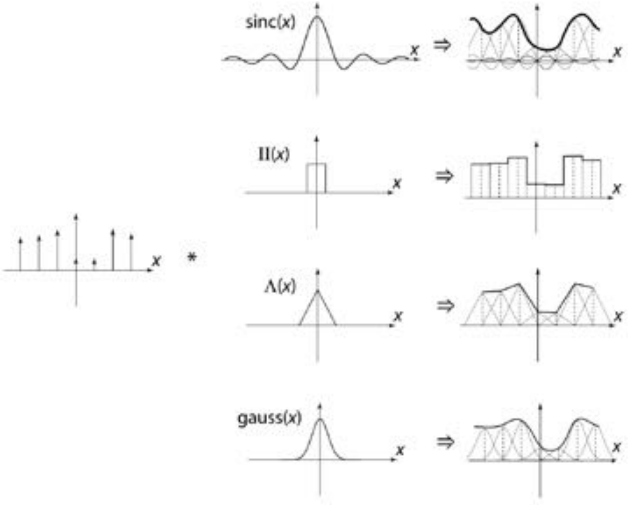
- Let's make this signal triple length ($d = 3$)
- What should be my "reconstruction" filter h (such that $G = h * G'$)?
- $h = [0, \frac{1}{d}, \ldots, \frac{d-1}{d}, 1, \frac{d-1}{d}, \ldots, \frac{1}{d}, 0]$, where d my upsampling factor

Interpolation via Convolution: 1D Example



- Let's make this signal triple length ($d = 3$)
- What should be my "reconstruction" filter h (such that $G = h * G'$)?
- $h = [0, \frac{1}{d}, \ldots, \frac{d-1}{d}, 1, \frac{d-1}{d}, \ldots, \frac{1}{d}, 0]$, where d my upsampling factor

Interpolation via Convolution (1D)



"Ideal" reconstruction

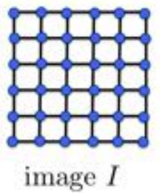
Nearest-neighbor interpolation

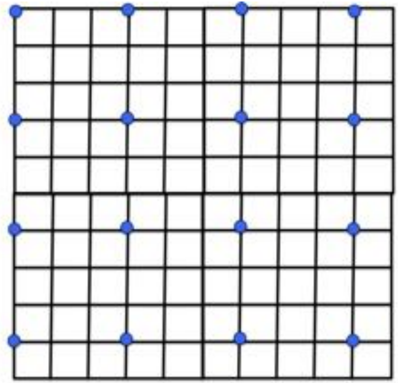
Linear interpolation

Gaussian reconstruction

Source: B. Curless

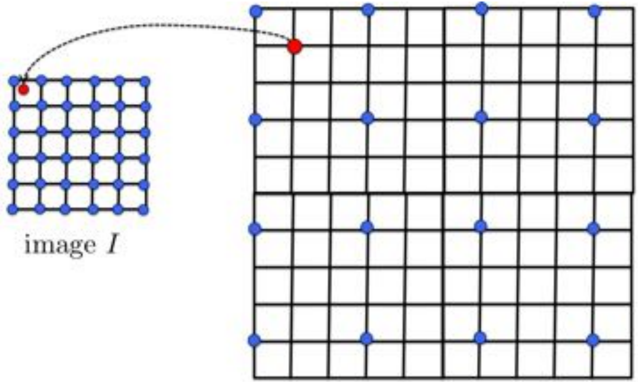
Image Interpolation (2D)





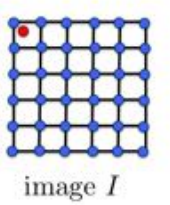
- Let's make this image triple size
- Copy image in every third pixel. What about the remaining pixels in G?

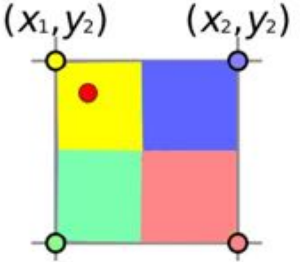
Image Interpolation (2D)



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

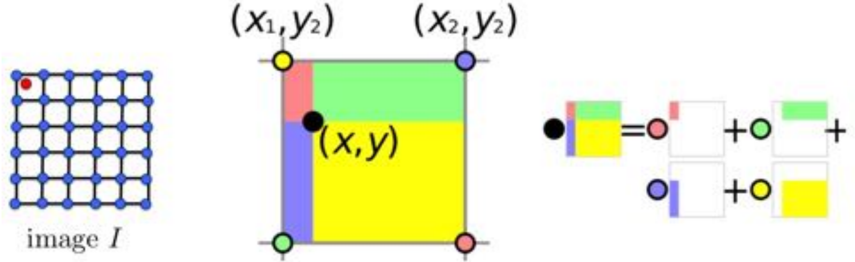
Image Interpolation (2D)





- Let's make this image triple size
- Copy image in every third pixel. What about the remaining pixels in G?
- One possible way: nearest neighbor interpolation

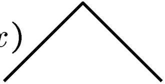
Image Interpolation (2D)



- Let's make this image triple size
- Copy image in every third pixel. What about the remaining pixels in G?
- Better: bilinear interpolation (check out details: http://en.wikipedia.org/wiki/Bilinear_interpolation)

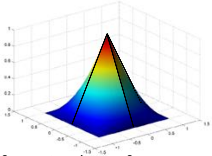
- What does the 2D version of this hat function look like?

$h(x)$



performs
linear interpolation

$h(x, y)$

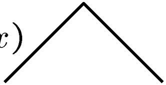


(tent function) performs
bilinear interpolation

Reconstruction Filters

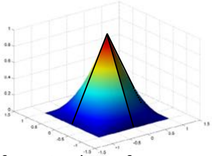
- What does the 2D version of this hat function look like?

$h(x)$



performs
linear interpolation

$h(x, y)$



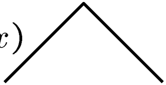
(tent function) performs
bilinear interpolation

- And filter for nearest neighbor interpolation?

Reconstruction Filters

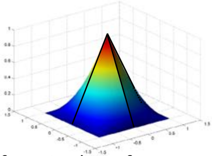
- What does the 2D version of this hat function look like?

$h(x)$



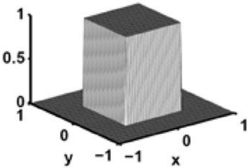
performs
linear interpolation

$h(x, y)$



(tent function) performs
bilinear interpolation

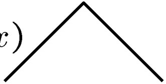
- And filter for nearest neighbor interpolation?



Reconstruction Filters

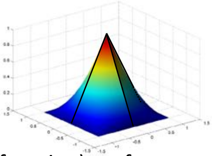
- What does the 2D version of this hat function look like?

$h(x)$



performs
linear interpolation

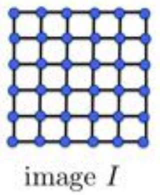
$h(x, y)$

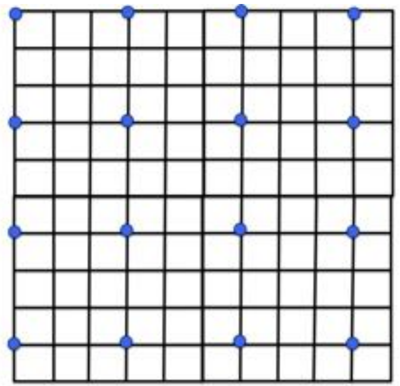


(tent function) performs
bilinear interpolation

- Better filters give better resampled images: Bicubic is a common choice

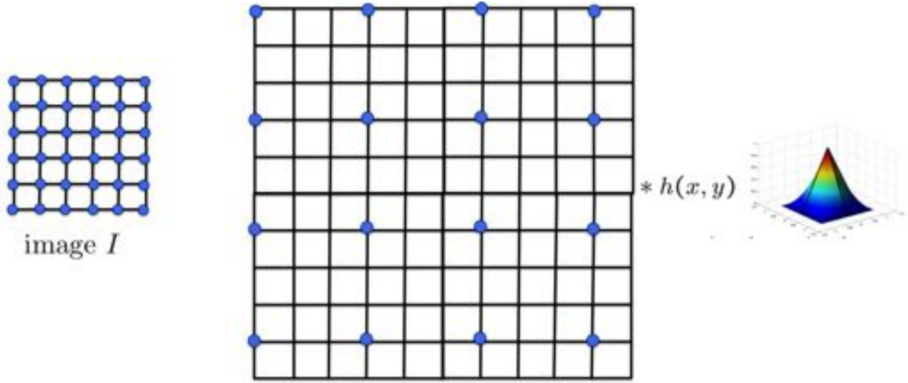
Image Interpolation via Convolution (2D)





- Let's make this image triple size: copy image values in every third pixel, place zeros everywhere else

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

Image Interpolation

Original image

Interpolation results

Nearest-neighbor interpolation

Bilinear interpolation

Bicubic interpolation

[Source: N. Snavely]

Deep Learning for Image Superresolution

- You can use DL to increase image resolution!

https://www.youtube.com/watch?v=pZXFXtfd-Ak

Pic credit:
https://medium.com/beyondminds/an-introduction-to-super-resolution-using-deep-learning-f60aff9a499d

Deep Learning Super Sampling (DLSS)

- You can use DL to increase image resolution!

https://news.developer.nvidia.com/dlss-three-things-you-need-to-know/

Summary – Stuff You Should Know

- To down-scale an image: blur it with a small Gaussian (e.g., $\sigma = 1.4$) and downsample
- 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...

Functions:

- IMRESIZE(IMAGE, SCALE, METHOD): Matlab's function for resizing the image, where METHOD="nearest", "bilinear", "bicubic" (works for downsampling and upsampling)
- SKIMAGE.TRANSFORM.RESIZE and SKIMAGE.TRANSFORM.RESCALE: Python's function for resizing, where ORDER is in the range 0-5 with the following semantics: 0: Nearest-neighbor 1: Bi-linear (default) 2: Bi-quadratic 3: Bi-cubic