Image processing basics

Andy Warhol, *Marilyn Diptych*, 1962 (source)

Many slides adapted from
Alyosha Efros, Derek Hoiem

Image processing basics: Outline

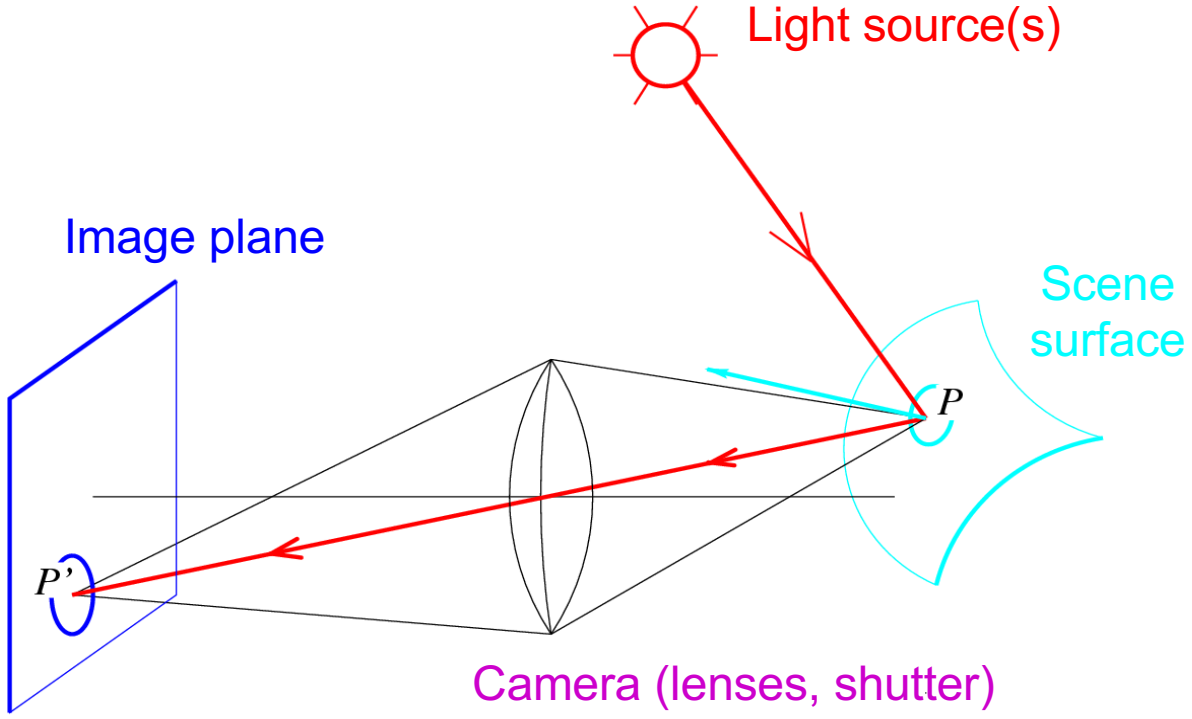
- Images as sampled functions
- Sampling and reconstruction, aliasing
- Image resampling, interpolation
- Image transformations

Image formation (preview)

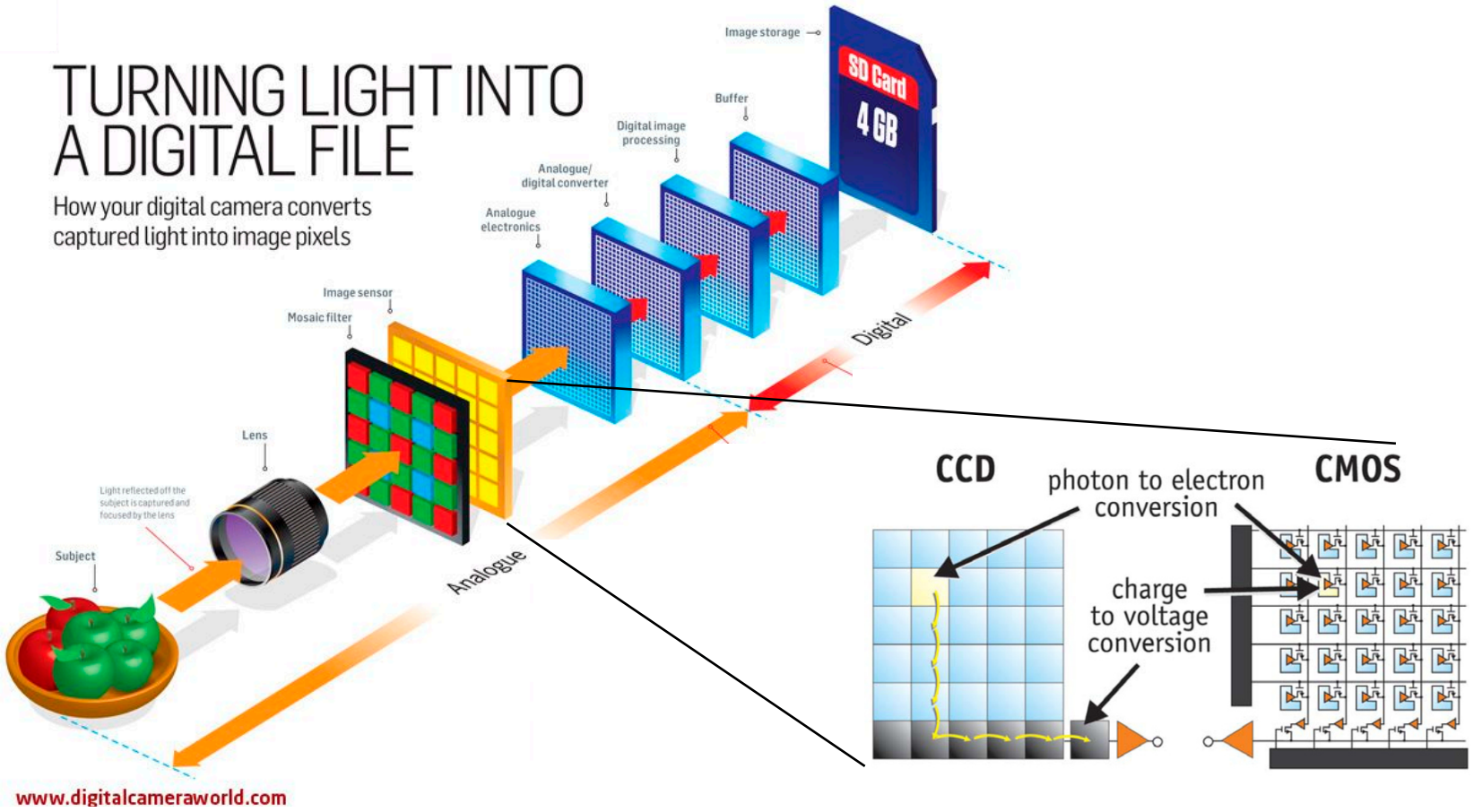
- What determines the brightness of an image pixel?

Light energy arriving at P' per unit time is quantified by ***irradiance*** (Watts per sq. meter)

Strictly speaking, we need ***spectral irradiance*** *to account for light across different wavelengths*



Images as sampled functions



Digital color image

Images in Python

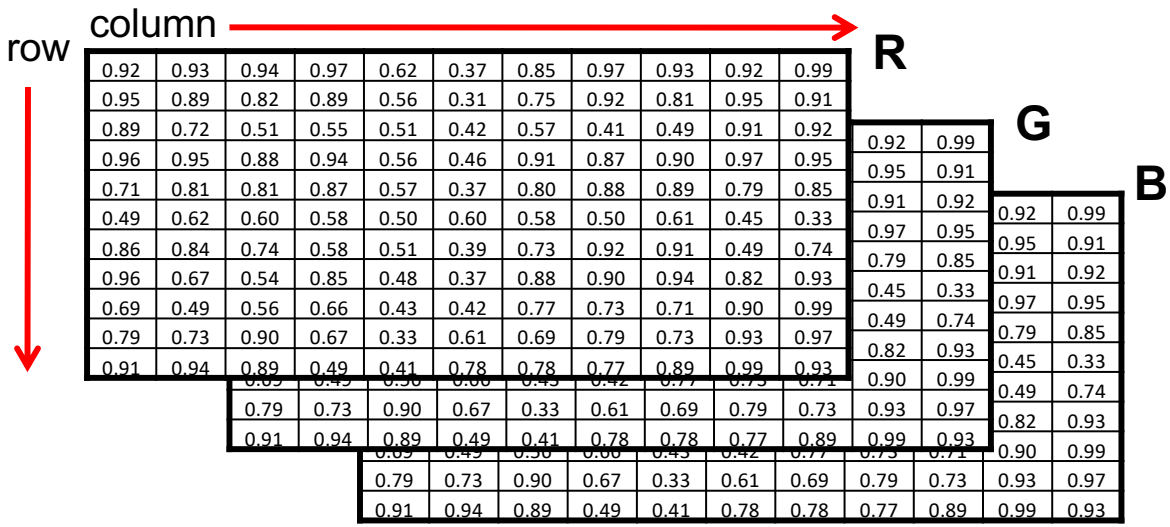
```
im = cv2.imread(filename)                    # read image
im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)     # order channels as RGB
im = im / 255                                # values range from 0 to 1
```

RGB image `im` is a H x W x 3 matrix (numpy.ndarray)

`im[0,0,0]` is the top-left pixel value in R-channel

`im[y, x, c]` is the value y+1 pixels down, x+1 pixels to right in the c^{th} channel

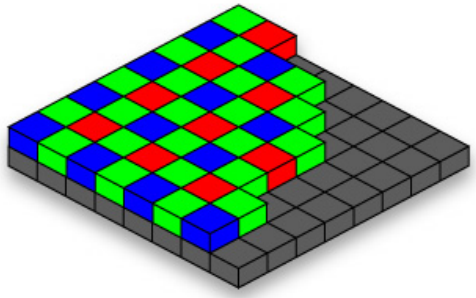
`im[H-1, W-1, 2]` is the bottom-right pixel in B-channel

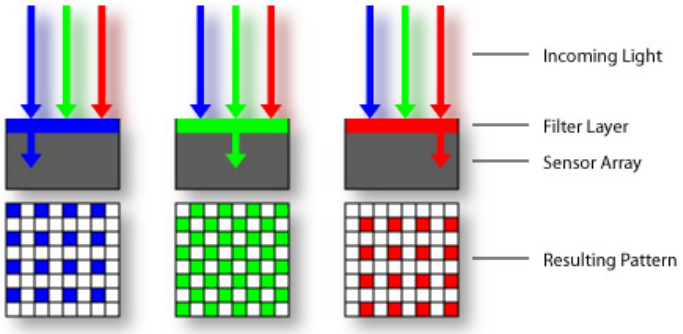


How are the three color channels acquired?

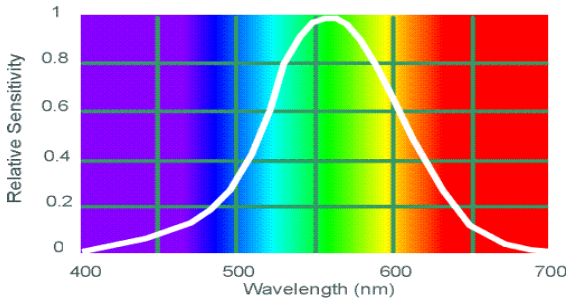
How are the three channels acquired?

Bayer grid (1976)





Why more green?



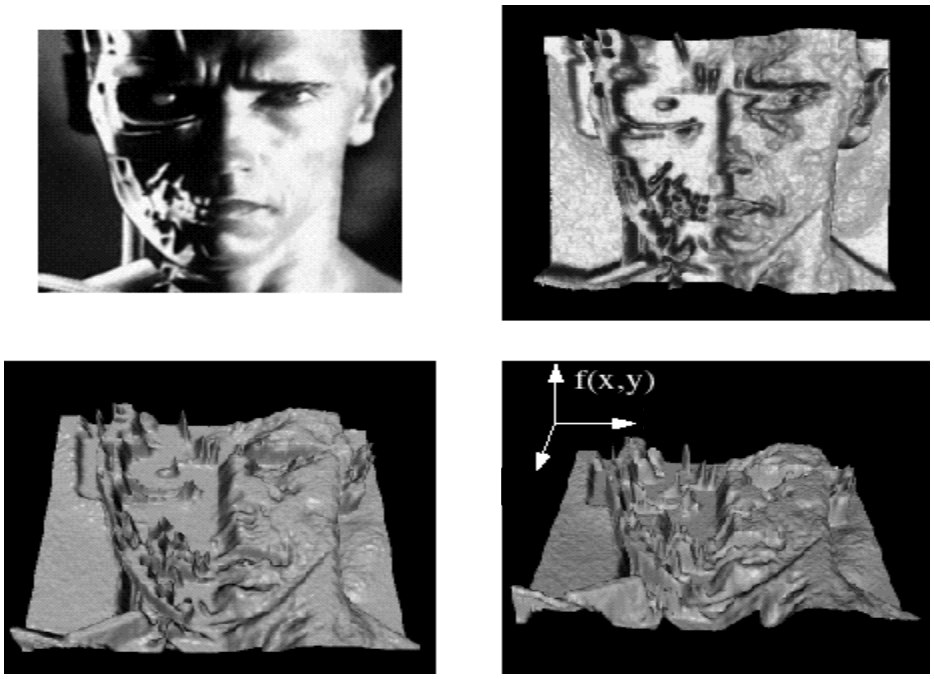
Human Luminance Sensitivity Function

Demosaicing:
Estimation of missing components from neighboring values

Source: Wikipedia

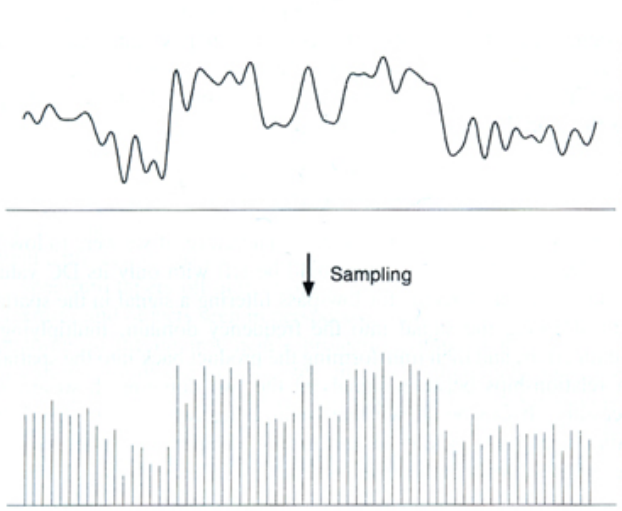
Images as sampled functions

- We like to think of a digital image as a sampled representation of a continuous function $f(x, y)$ defined over a continuous 2D domain

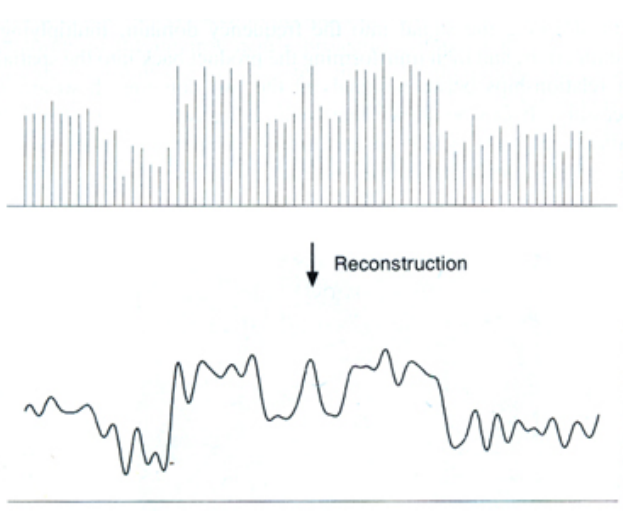


Sampling and reconstruction

- Sampling: recording the function's values at a discrete set of locations



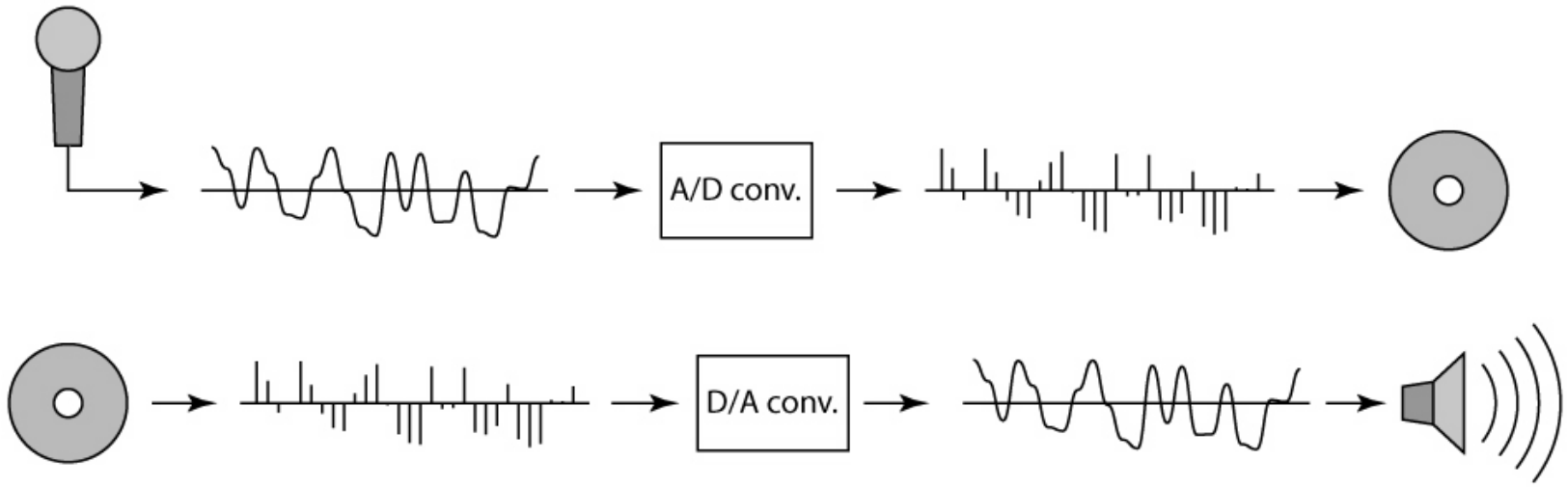
- Reconstruction: converting a sampled representation back into a continuous function by "guessing" what happens between the samples



Source: S. Marschner (via A. Efros)

1D example: Digital audio

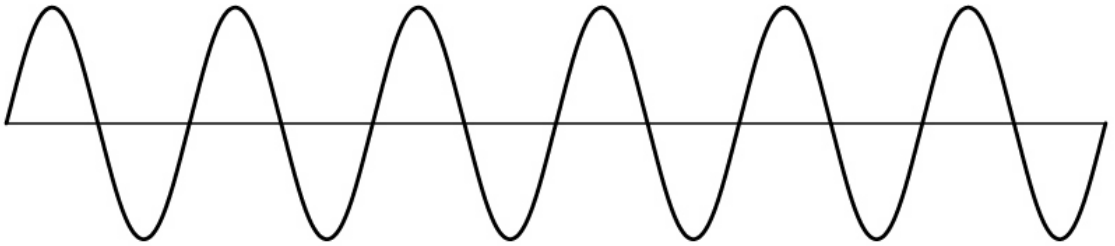
- Recording: sound to analog to samples to disc
- Playback: disc to samples to analog to sound again



Source: S. Marschner (via A. Efros)

Sampling and reconstruction

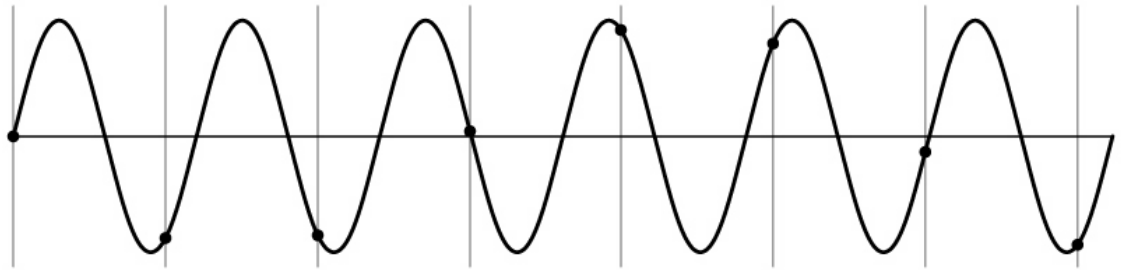
- Simple example: a sine wave



Source: S. Marschner (via A. Efros)

Sampling and reconstruction

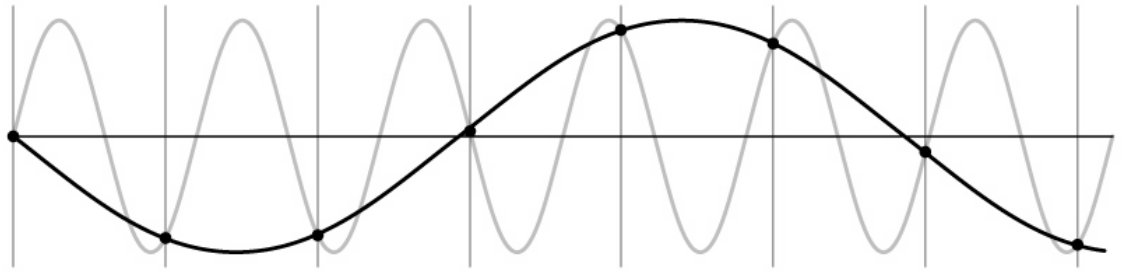
- Simple example: a sine wave
- What if we “missed” things between the samples?
  - Unsurprising result: information is lost



Source: S. Marschner (via A. Efros)

Sampling and reconstruction

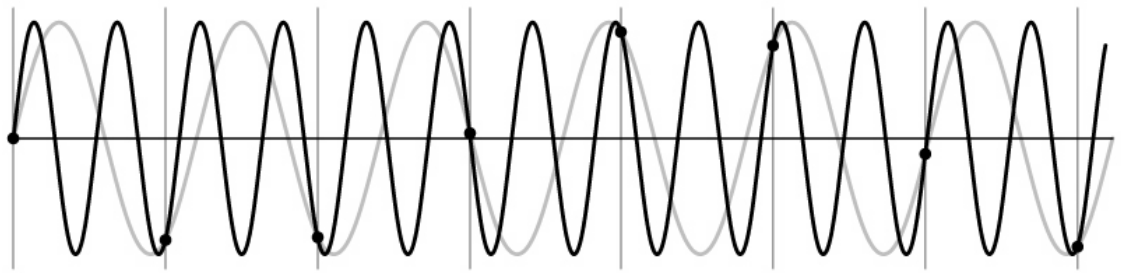
- Simple example: a sine wave
- What if we “missed” things between the samples?
  - Unsurprising result: information is lost
  - Surprising result: indistinguishable from lower frequencies



Source: S. Marschner (via A. Efros)

Sampling and reconstruction

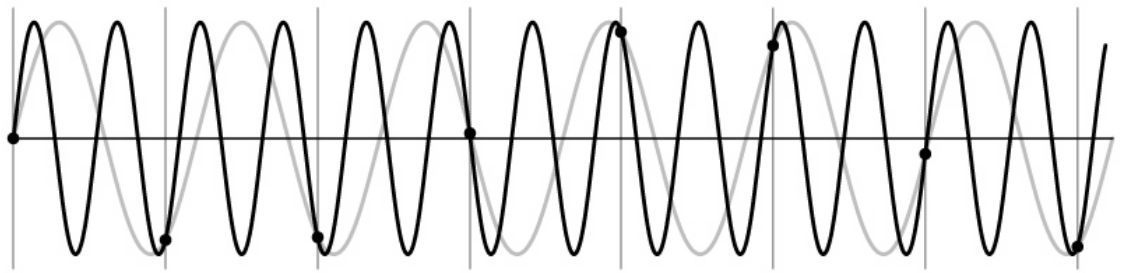
- Simple example: a sine wave
- What if we “missed” things between the samples?
  - Unsurprising result: information is lost
  - Surprising result: indistinguishable from lower frequencies (or even higher frequencies)



Source: S. Marschner (via A. Efros)

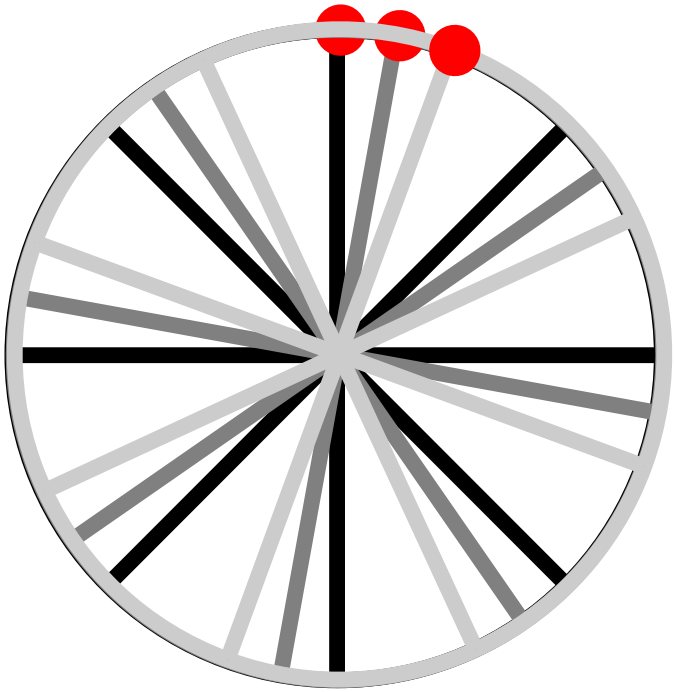
Sampling and reconstruction

- Simple example: a sine wave
- What if we “missed” things between the samples?
  - Unsurprising result: information is lost
  - Surprising result: indistinguishable from lower frequencies (or even higher frequencies)
  - *Aliasing*: signal “traveling in disguise” as other frequencies



Source: S. Marschner (via A. Efros)

Wagon wheel effect

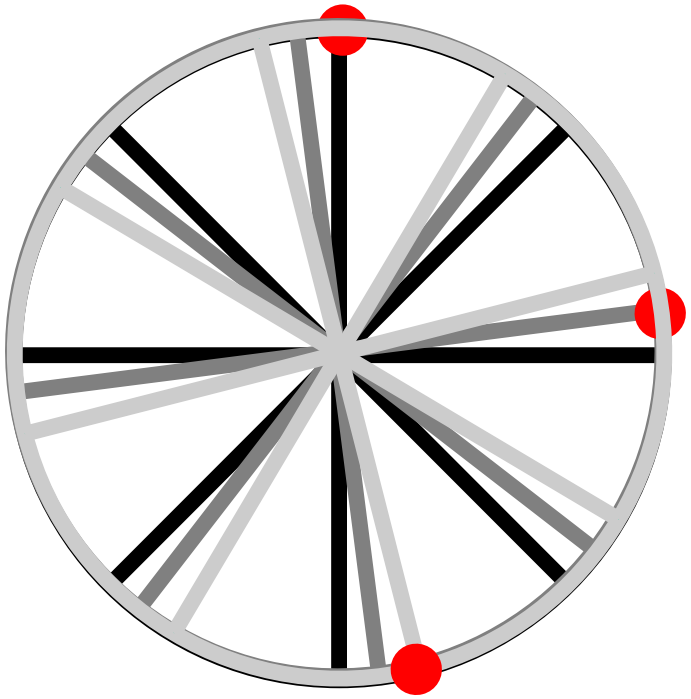


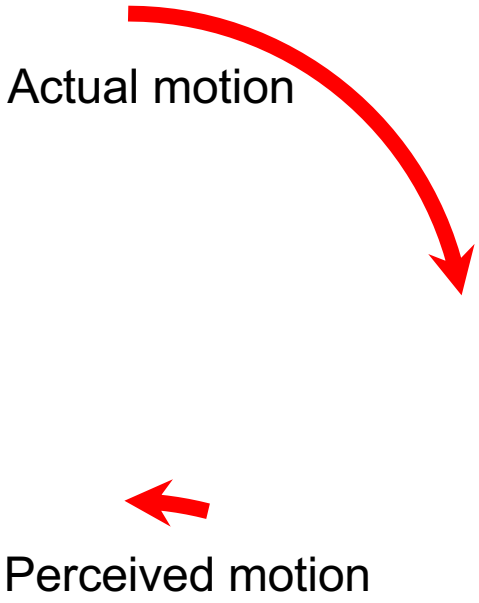
Actual motion

Perceived motion

https://en.wikipedia.org/wiki/Wagon-wheel_effect
https://maycontainmaths.wordpress.com/2015/02/22/the-wagon-wheel-effect/

Wagon wheel effect

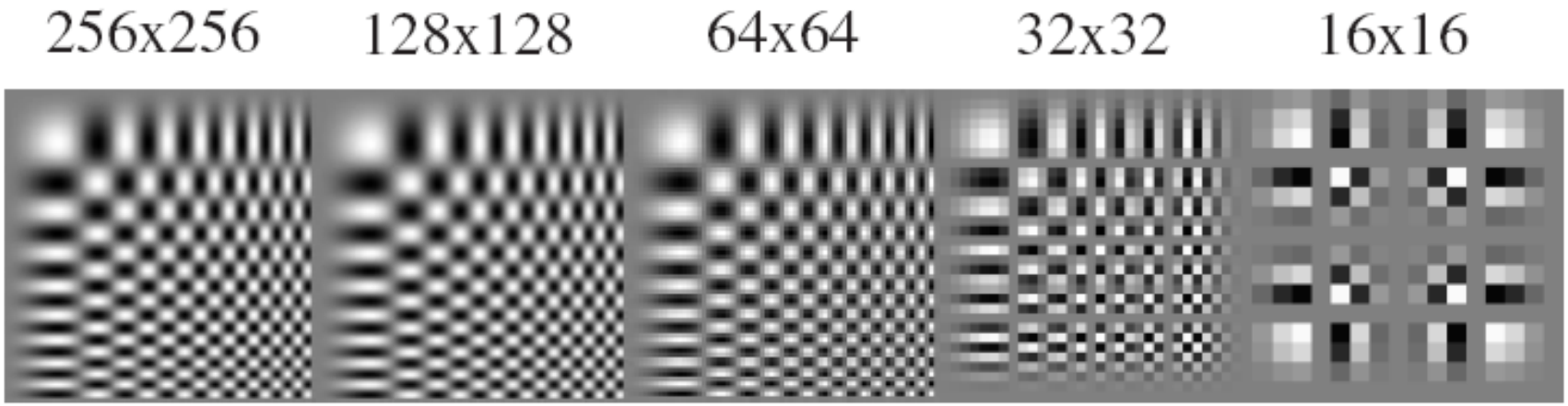




Perceived motion

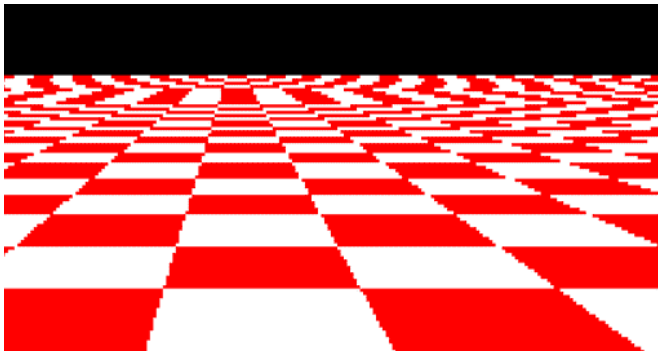
https://en.wikipedia.org/wiki/Wagon-wheel_effect
https://maycontainmaths.wordpress.com/2015/02/22/the-wagon-wheel-effect/

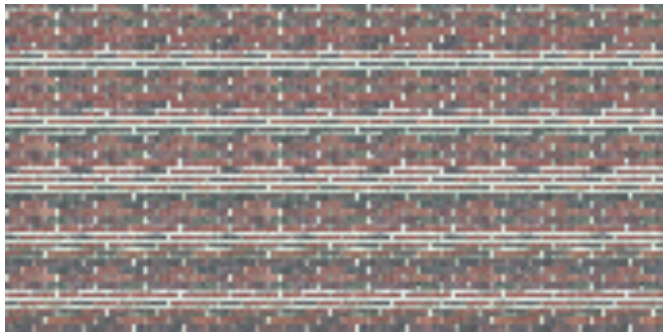
Aliasing in images



Aliasing “in the wild”

Disintegrating textures

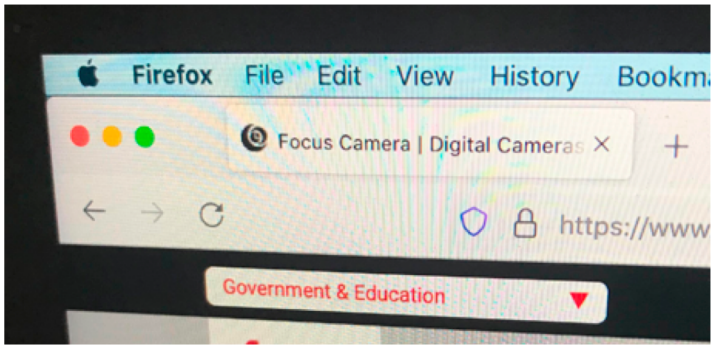




Source

Moire patterns, false color

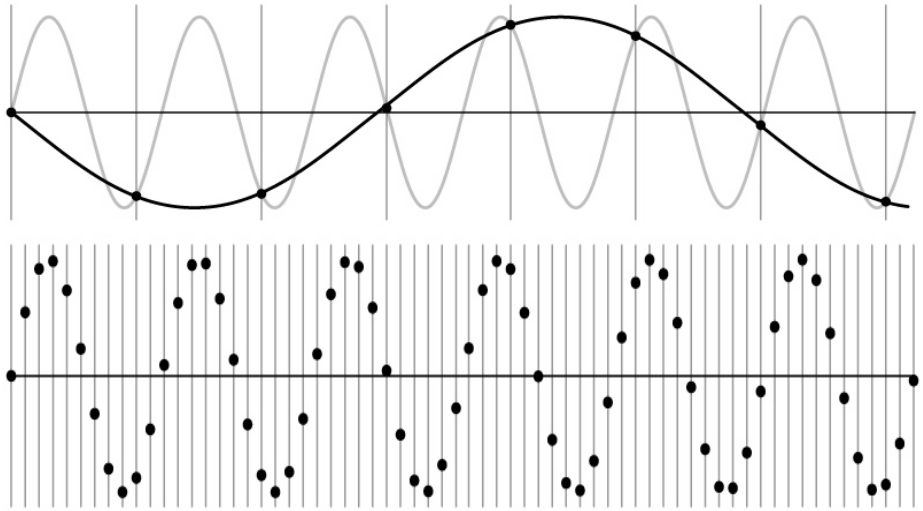
Source



Source

Nyquist-Shannon sampling theorem

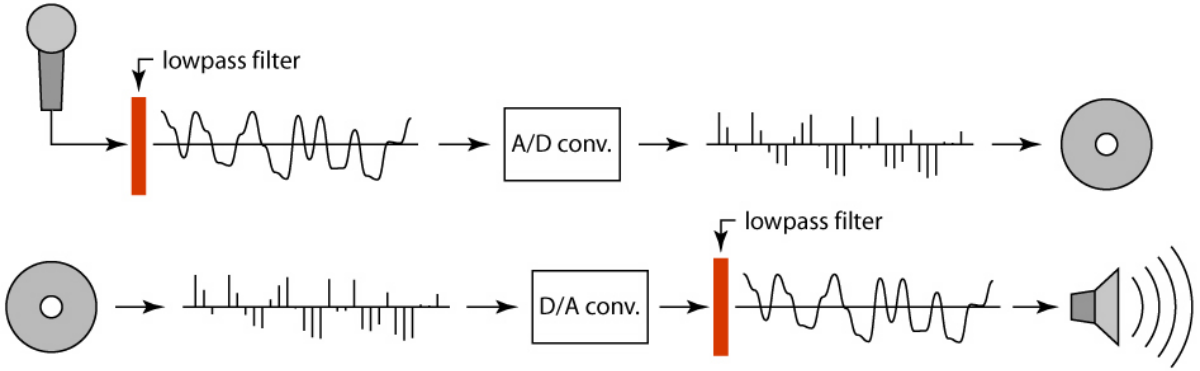
- When sampling a signal at discrete intervals, the sampling frequency must be at least *twice* the maximum frequency of the input signal to allow us to reconstruct the original perfectly from the sampled version



https://en.wikipedia.org/wiki/Nyquist-Shannon_sampling_theorem

Anti-aliasing

- What are possible solutions?
  - Sample more often (if you can)
  - Get rid of all frequencies that are greater than half the new sampling frequency
    - Will lose information, but that's better than aliasing
    - How to get rid of high frequencies?
      - Apply a smoothing or *low-pass* filter (later)



Why should you care about anti-aliasing?

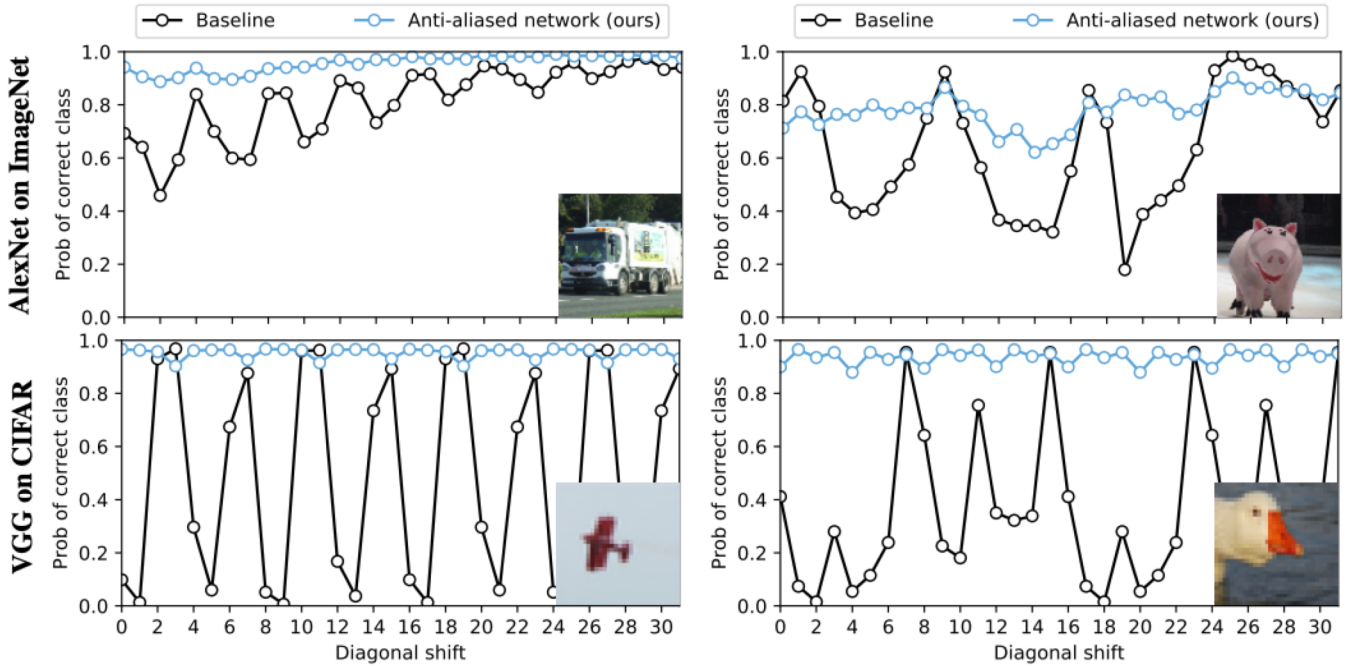


Figure 1. **Classification stability for selected images.** Predicted probability of the correct class changes when shifting the image. The baseline (black) exhibits chaotic behavior, which is stabilized by our method (blue). We find this behavior across networks and datasets. Here, we show selected examples using AlexNet on ImageNet **(top)** and VGG on CIFAR10 **(bottom)**. Code and anti-aliased versions of popular networks are available at `https://richzhang.github.io/antialiased-cnns/`.

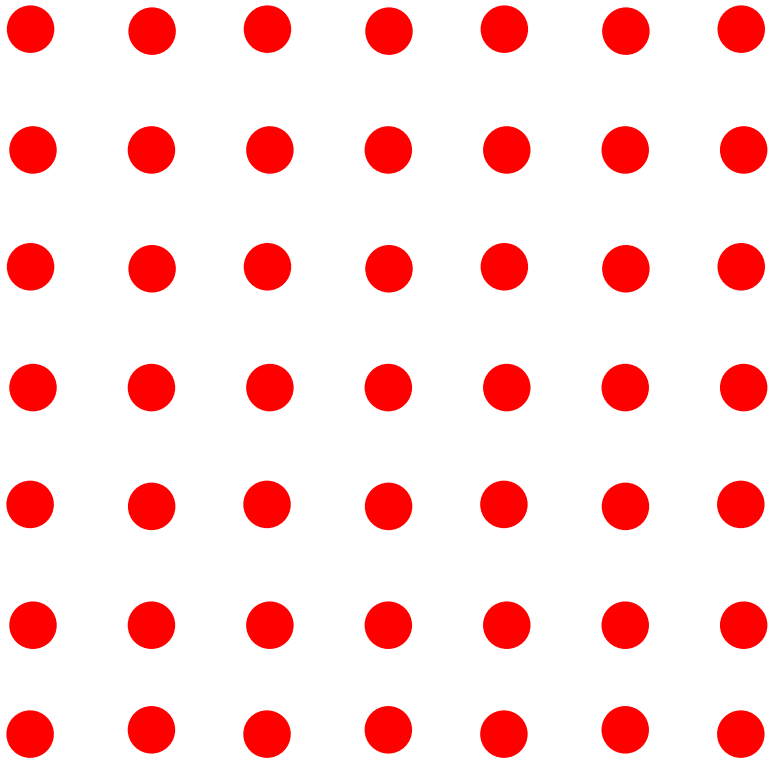
R. Zhang. Making convolutional networks shift-invariant again. ICML 2019

Image processing basics: Outline

- Images as sampled functions
- Sampling and reconstruction, aliasing
- Image resampling, interpolation

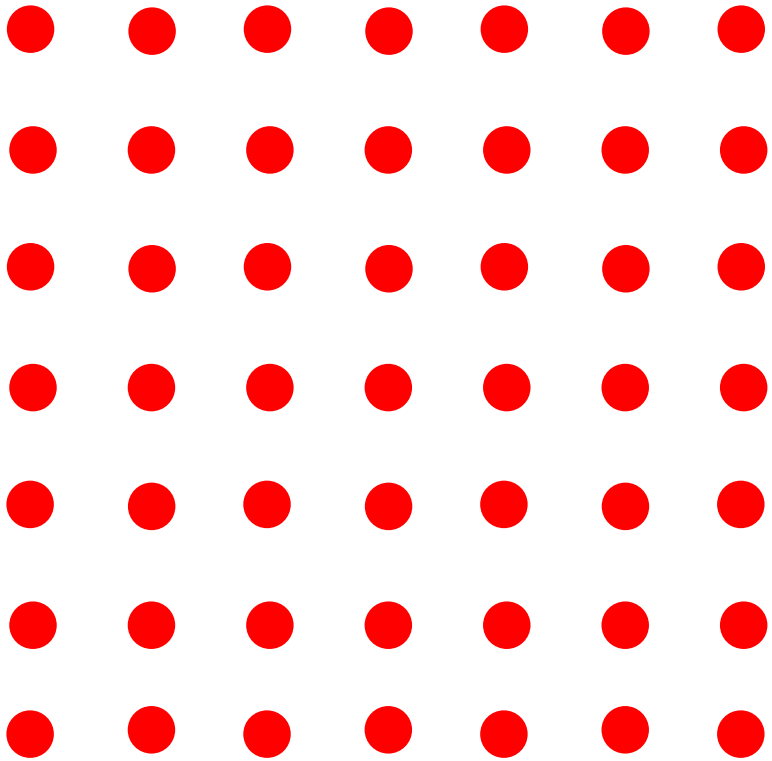
Subsampling an image

- How do we reduce the size of an image by a factor of two?



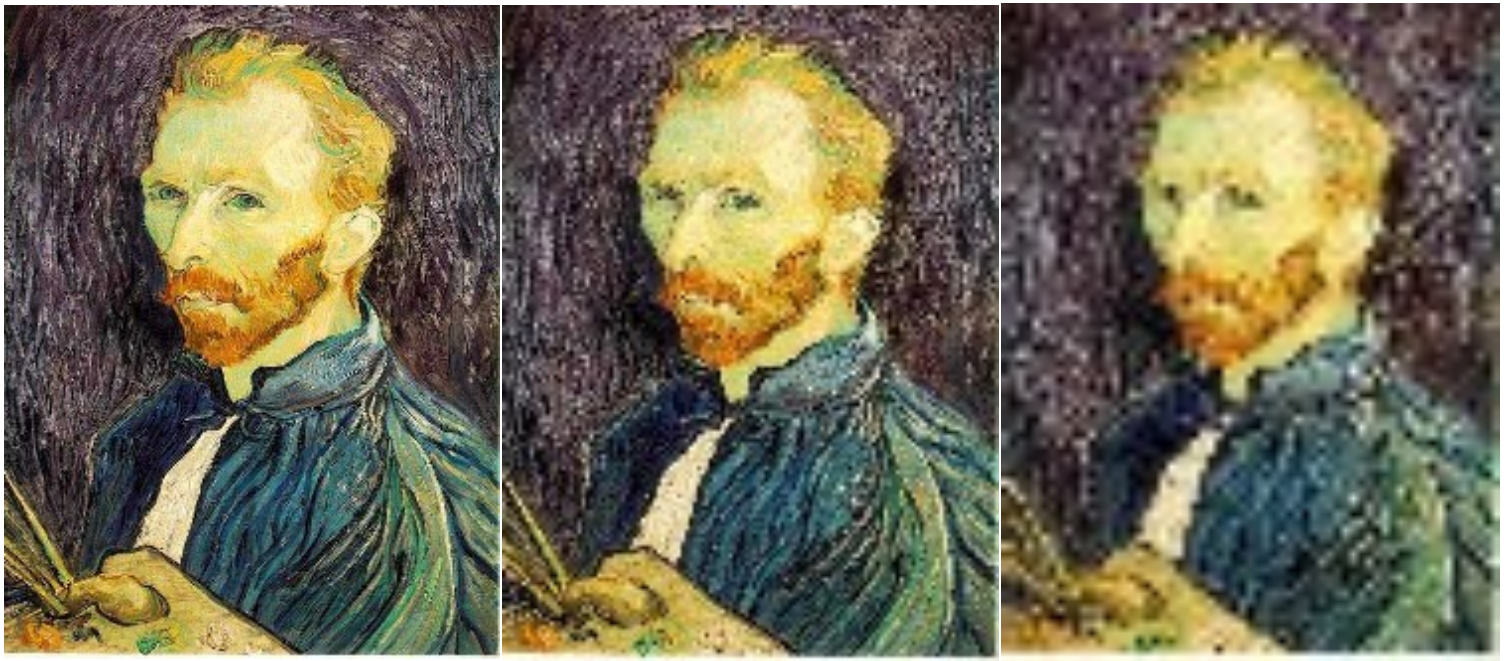
Subsampling an image

- How do we reduce the size of an image by a factor of two?



How about throwing away every other row and column to create a half-size image?

Subsampling without pre-filtering



1/2 1/4 (2x zoom) 1/8 (4x zoom)

Source: S. Seitz (via D. Hoiem)

Subsampling with pre-filtering

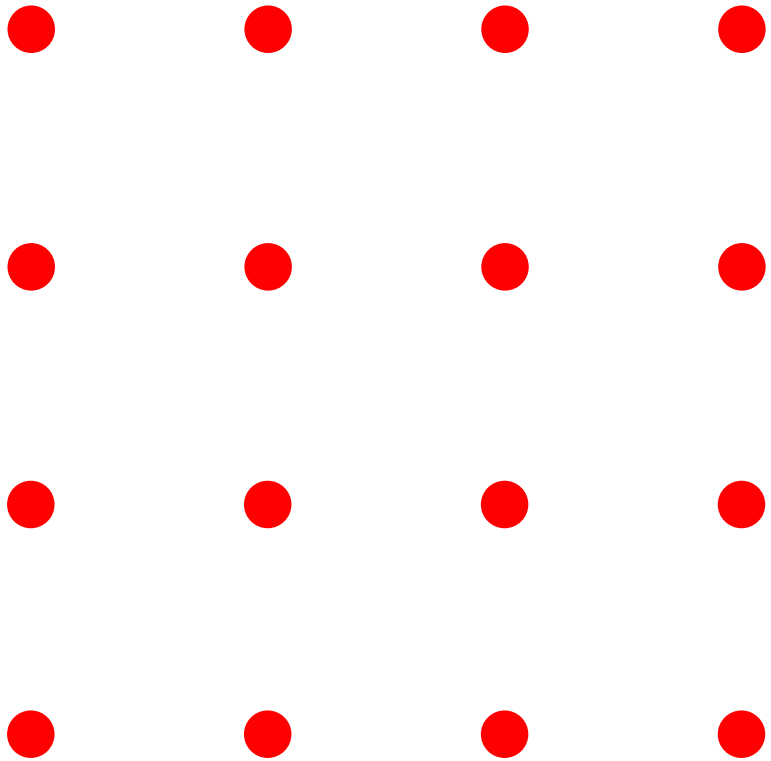
1/2 1/4 1/8

- Image is smoothed with a *Gaussian filter* before subsampling

Source: S. Seitz (via D. Hoiem)

Upsampling an image

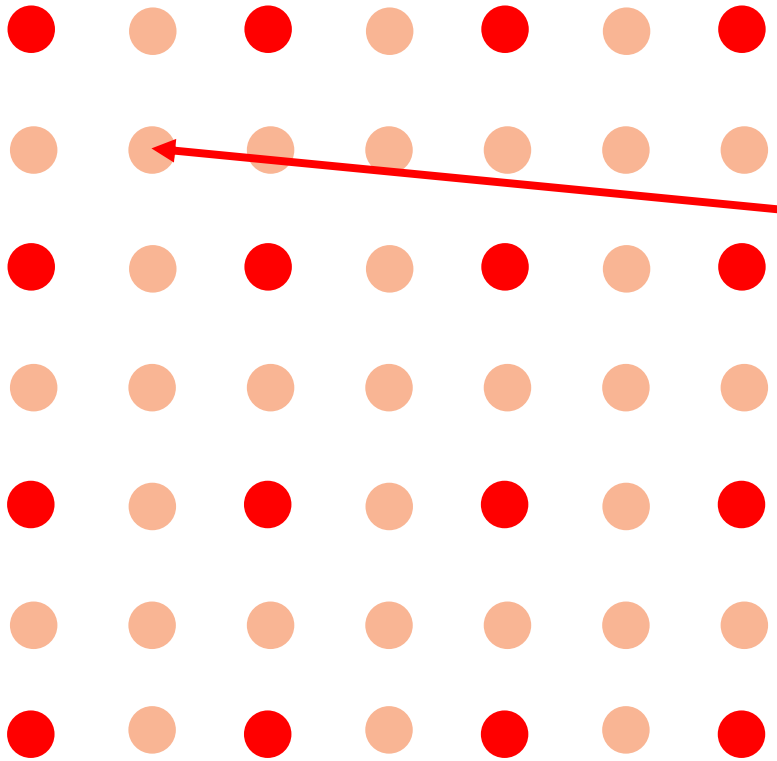
- How do we *increase* the size of an image by a factor of two?



Let's increase the resolution of the sampling grid!

Upsampling an image

- How do we *increase* the size of an image by a factor of two?

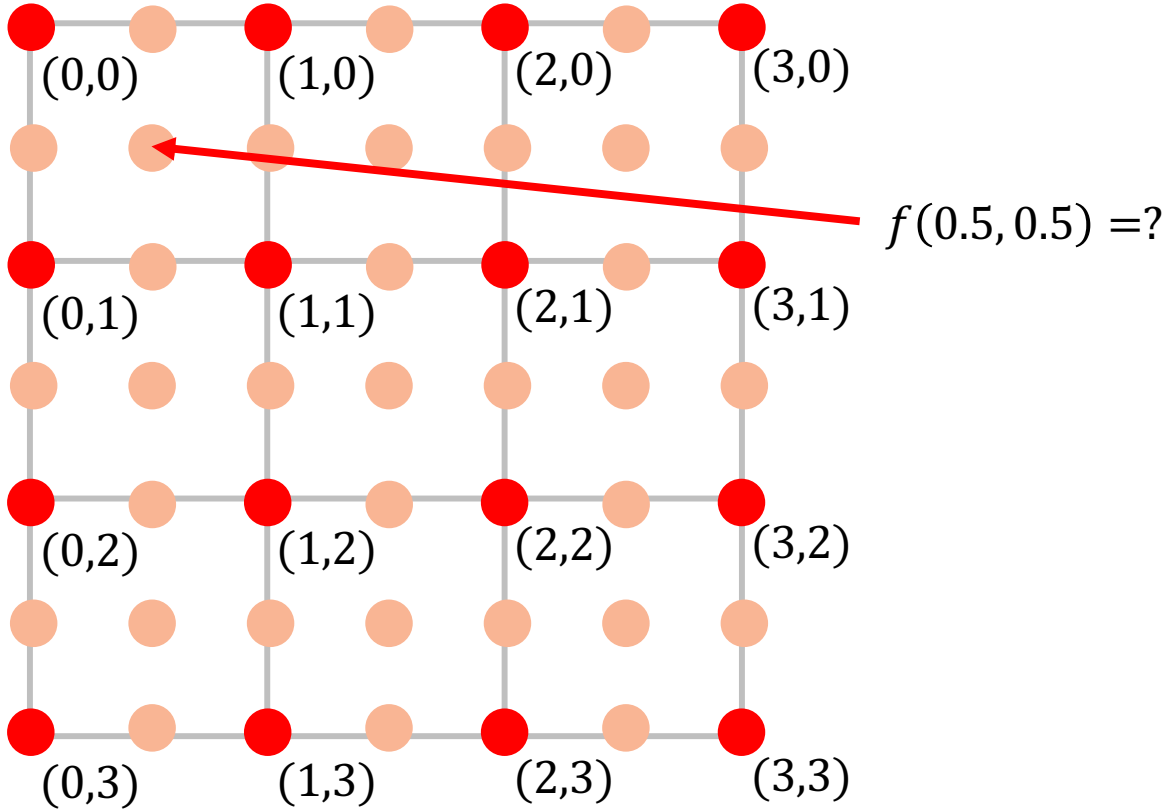


What should this value be?

Need to *interpolate*!

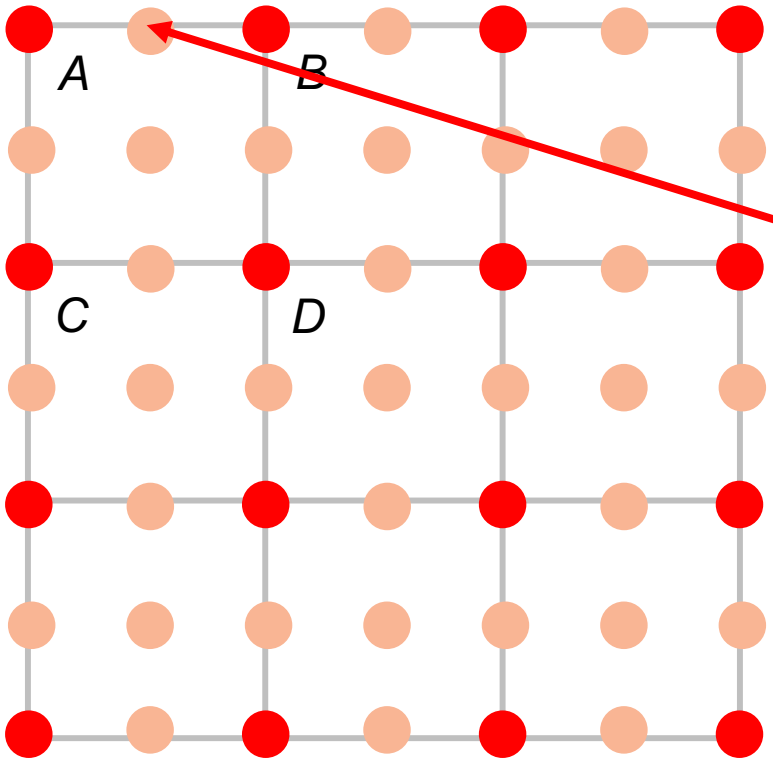
Upsampling an image

- How do we *increase* the size of an image by a factor of two?



Bilinear interpolation

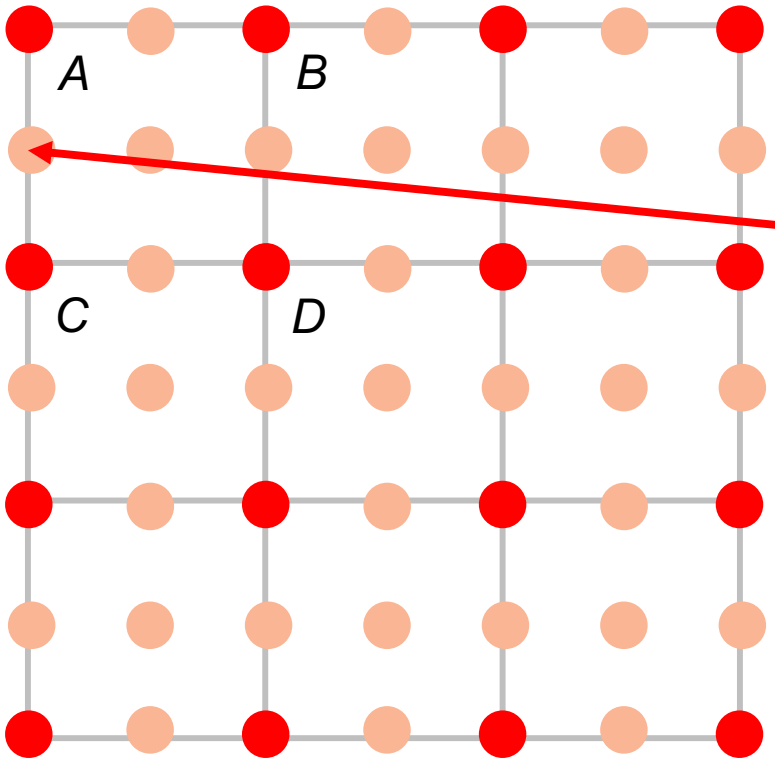
- Let $f(0,0) = A, f(1,0) = B, f(0,1) = C, f(1,1) = D$



$$f(0.5, 0) = \frac{1}{2}(A + B)$$

Bilinear interpolation

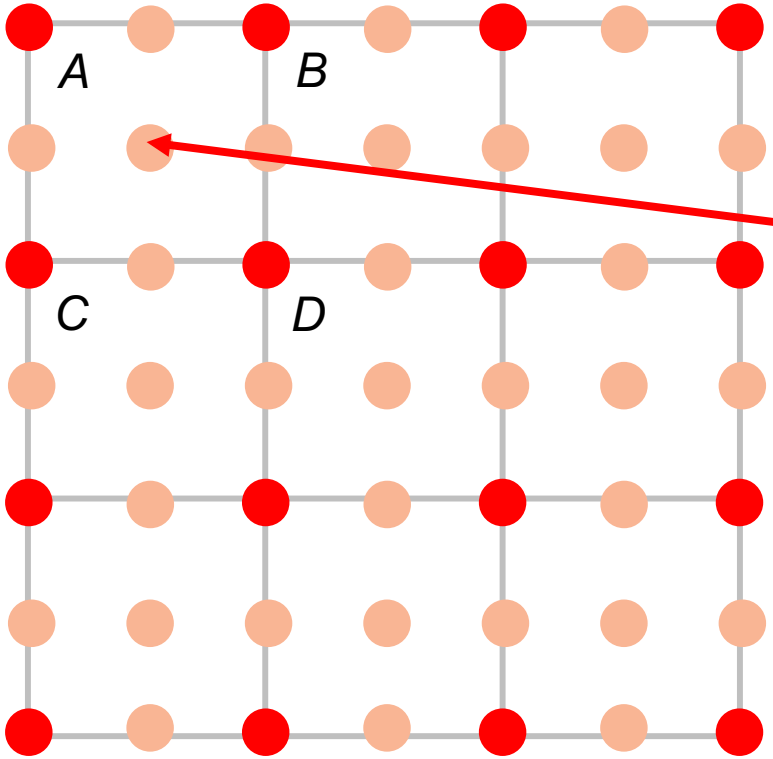
- Let $f(0,0) = A, f(1,0) = B, f(0,1) = C, f(1,1) = D$



$$f(0, 0.5) = \frac{1}{2}(A + C)$$

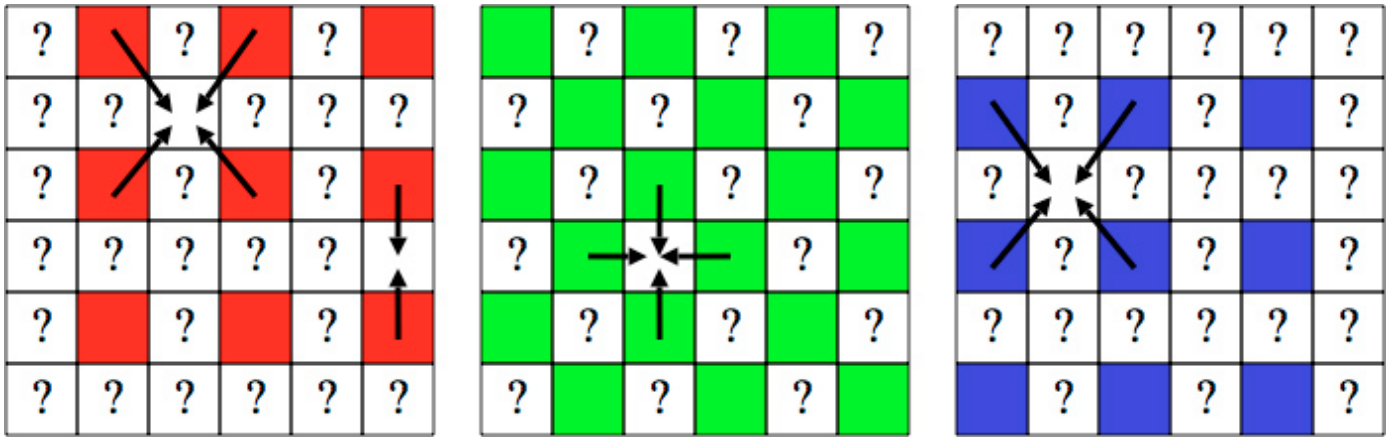
Bilinear interpolation

- Let $f(0,0) = A, f(1,0) = B, f(0,1) = C, f(1,1) = D$

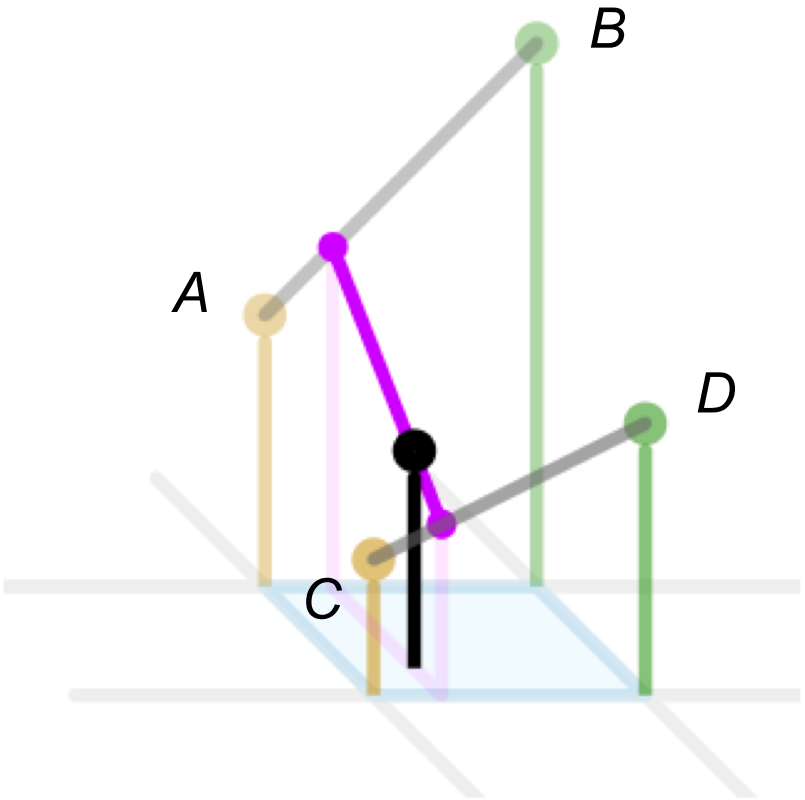


$$f(0.5, 0.5) = \frac{1}{4}(A + B + C + D)$$

Application: Demosaicing

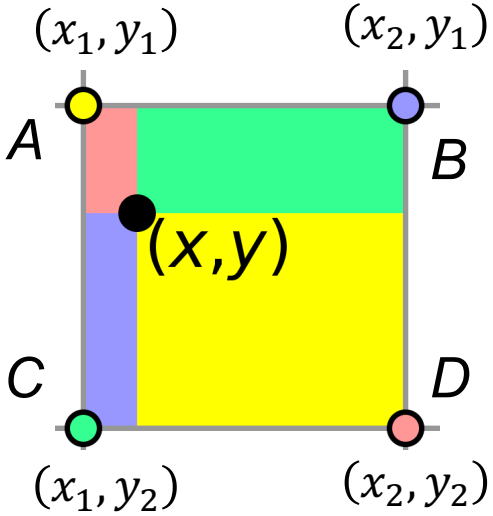


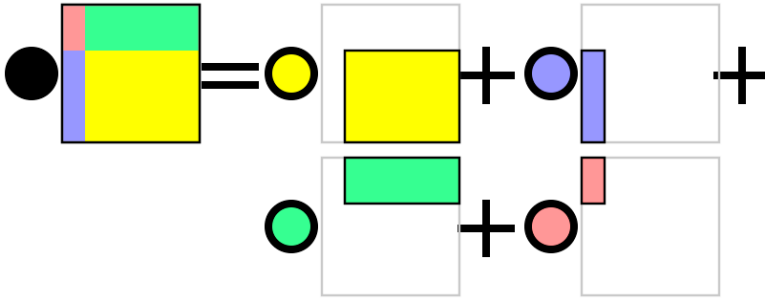
Bilinear interpolation more generally



http://en.wikipedia.org/wiki/Bilinear_interpolation

Bilinear interpolation more generally





$$f(x,y) = w_{11}A + w_{21}B + w_{12}C + w_{22}D$$

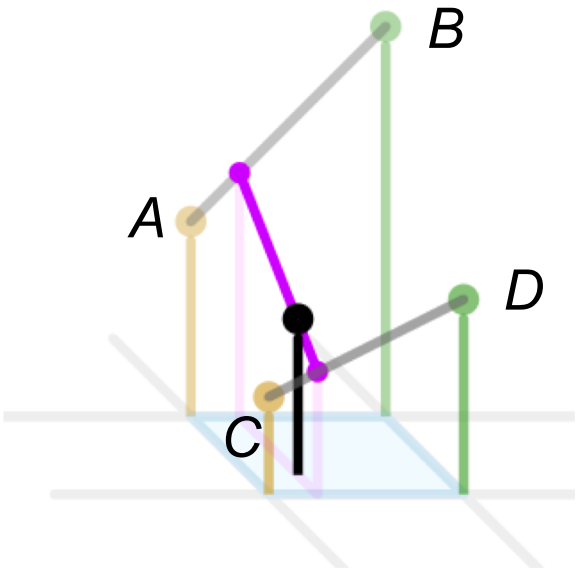
$$w_{11} = \frac{(x_2 - x)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} \qquad w_{21} = \frac{(x - x_1)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)}$$

$$w_{12} = \frac{(x_2 - x)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} \qquad w_{22} = \frac{(x - x_1)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)}$$

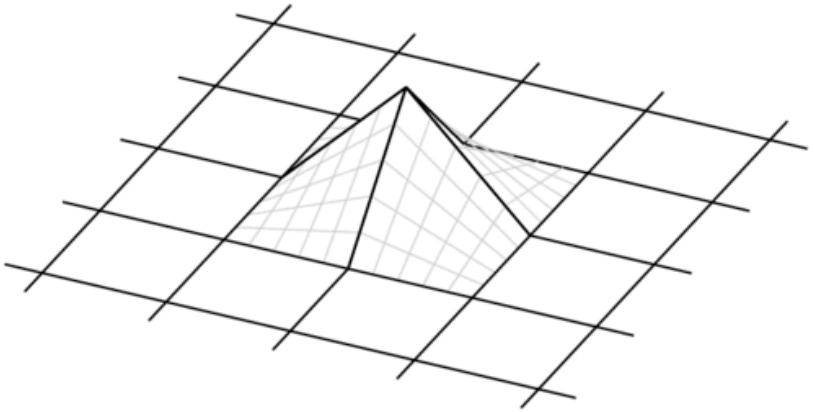
http://en.wikipedia.org/wiki/Bilinear_interpolation

Bilinear interpolation: Basis function view

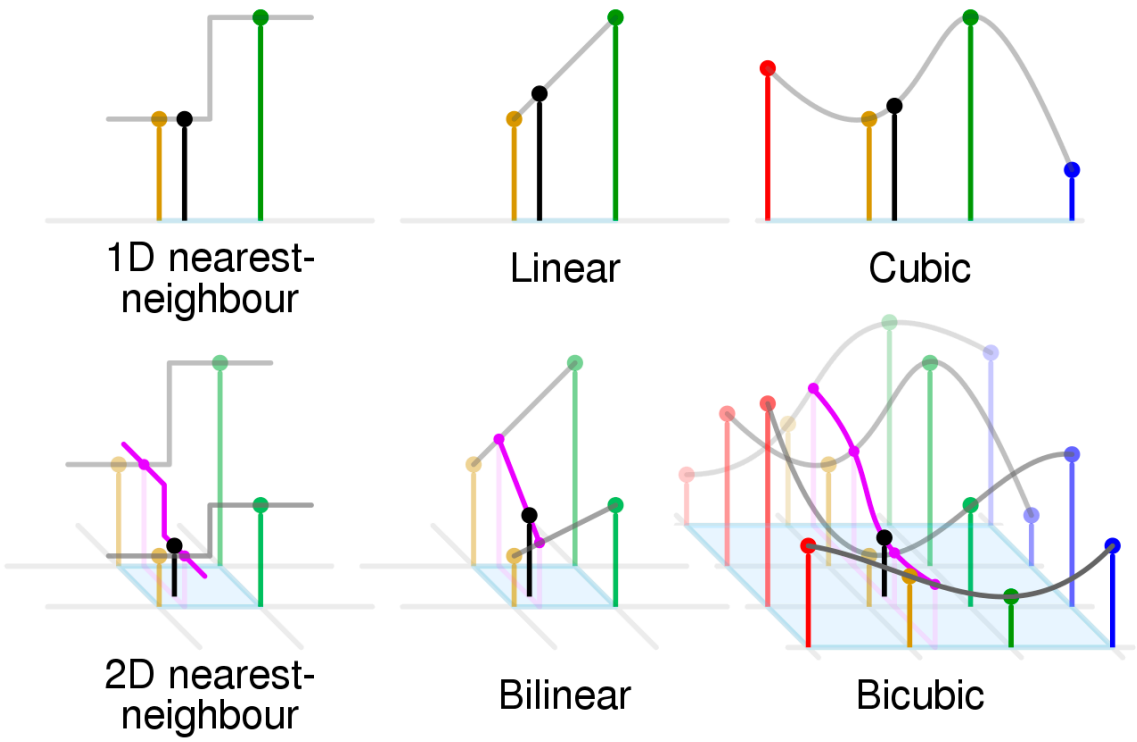
- Interpolated function is sum of basis functions or “bumps” centered at the four adjacent grid points, weighted by the image values at the corresponding points



Bilinear basis function



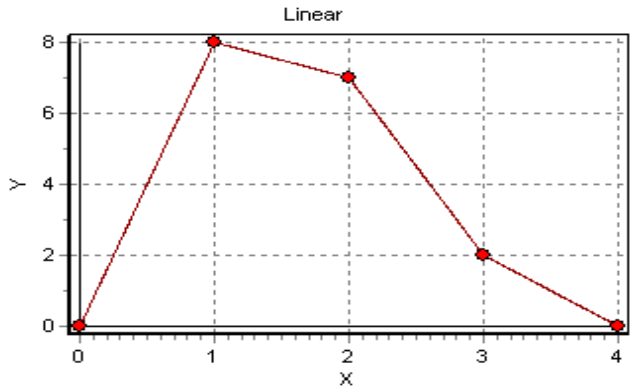
Other kinds of interpolation



Source: Wikipedia

Interpolation and function extrema

- When you use linear interpolation, extrema of the image function can only occur at the original sample points
- What about nonlinear interpolation?



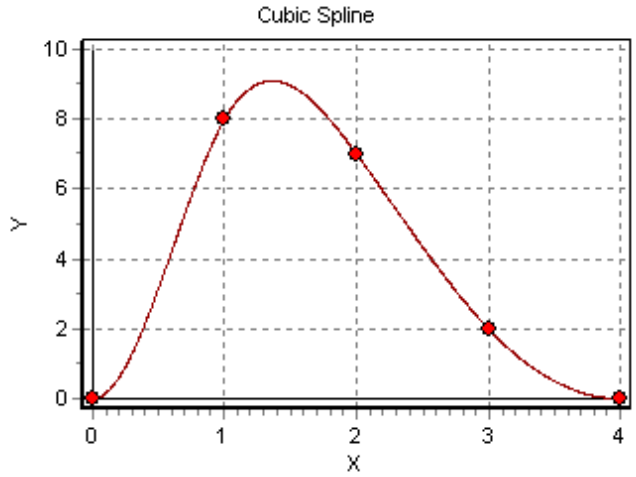


Image source

Image processing basics: Outline

- Images as sampled functions
- Sampling and reconstruction, aliasing
- Image resampling, interpolation
- Image transformations

Image transformations

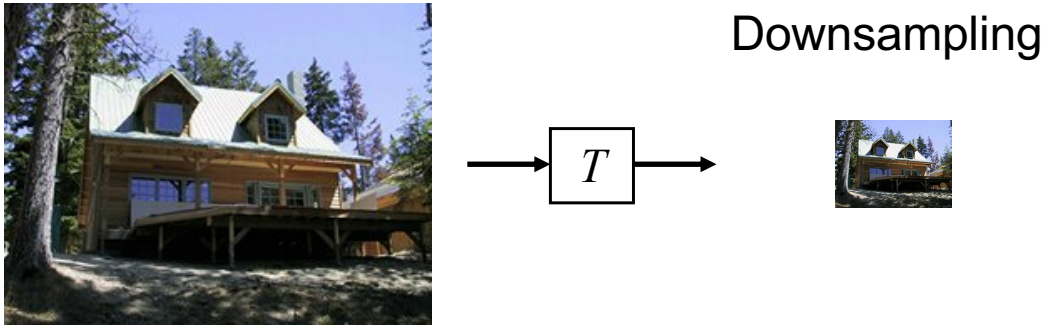


Image transformations

Upsampling

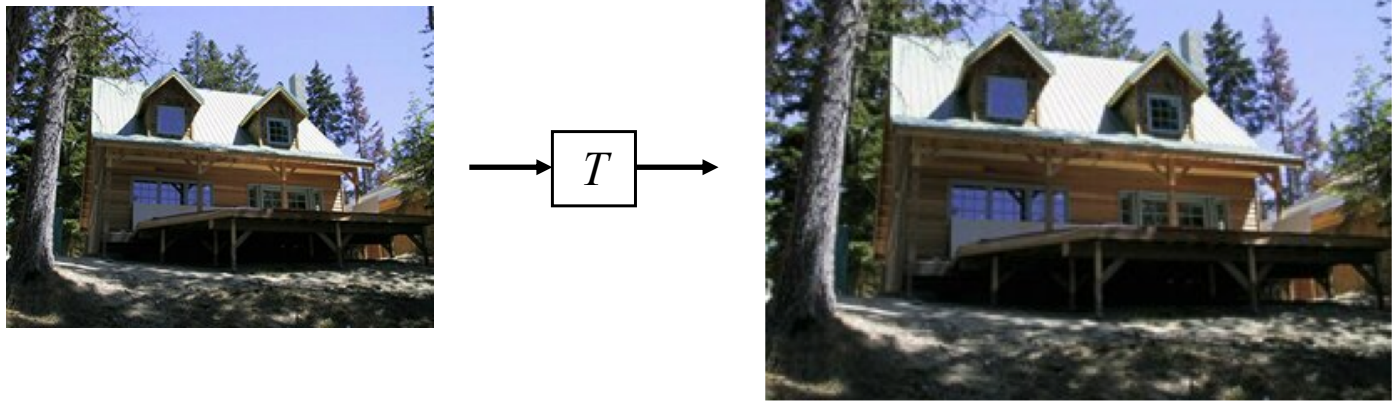


Image transformations

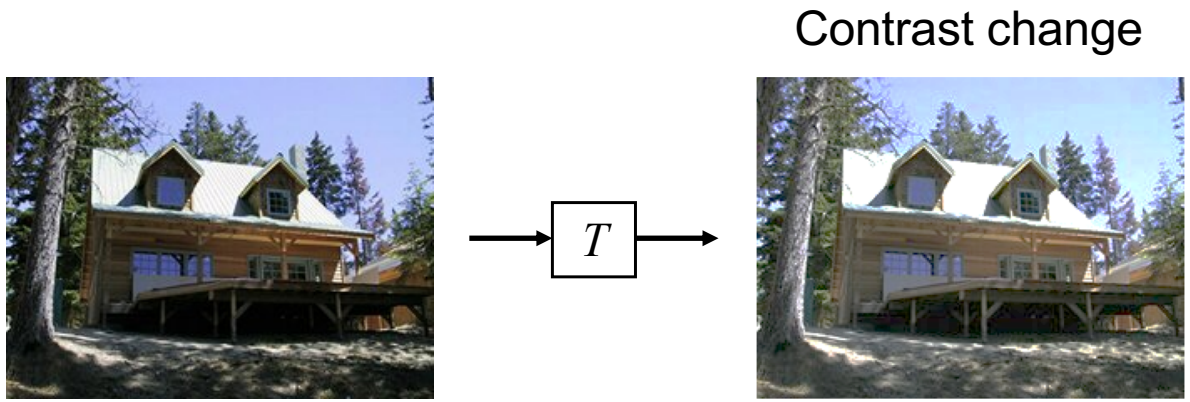


Image transformations

Blurring/sharpening

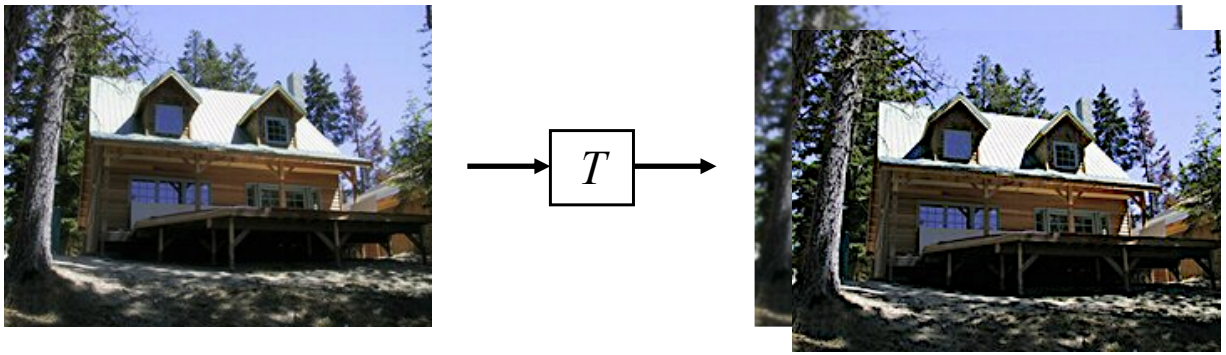
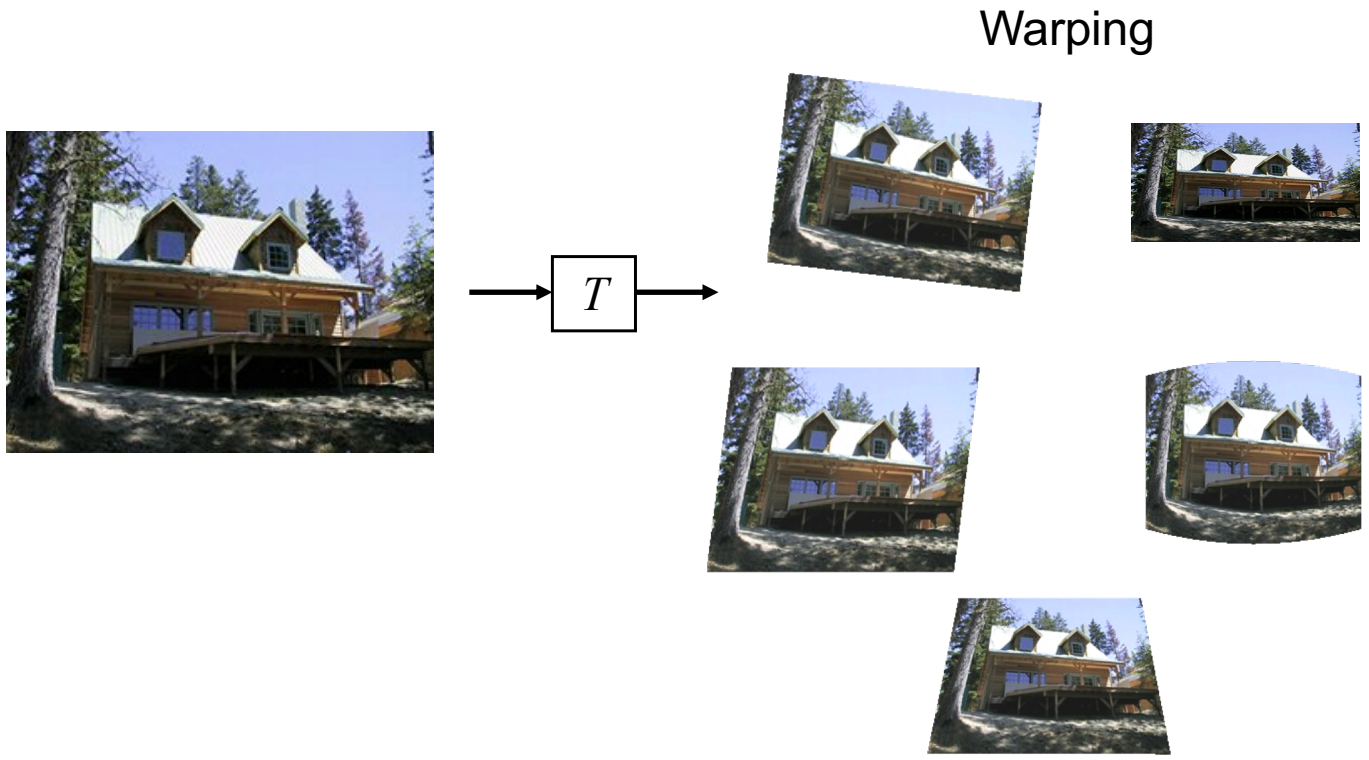
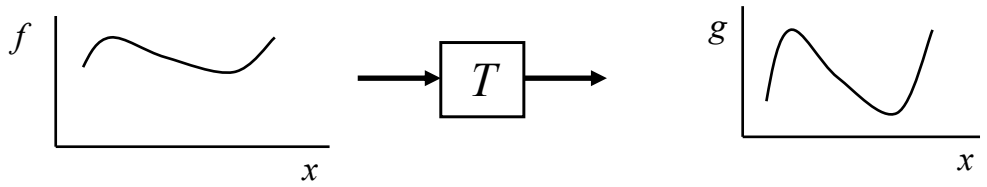


Image transformations



Point processing

- Change ***range*** of image: $g = T(f)$

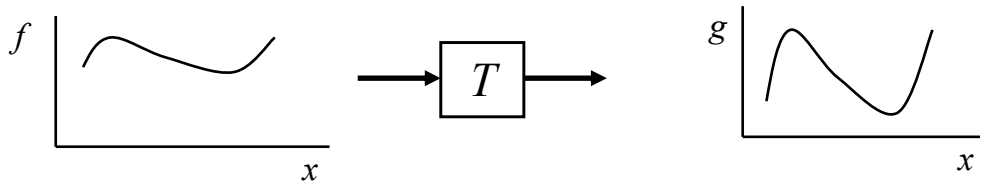


- *Negative:* $g = 1 - f$

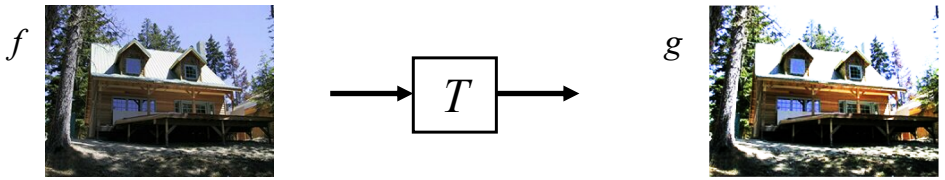
Image source

Point processing

- Change ***range*** of image: $g = T(f)$

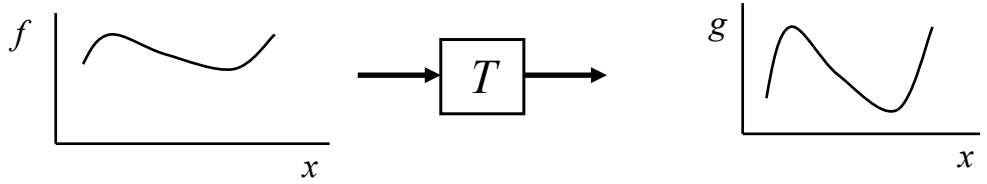


- *Affine* contrast adjustment: $g = af + b$

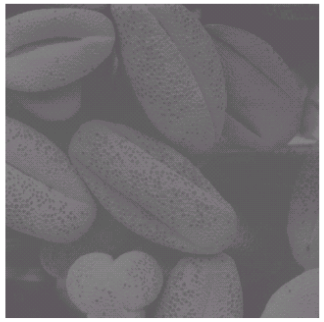


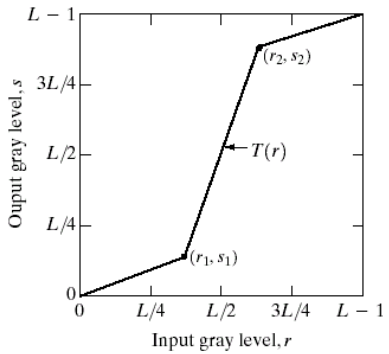
Point processing

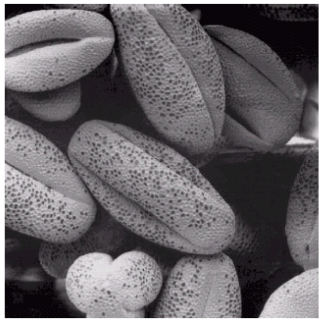
- Change ***range*** of image: $g = T(f)$



- *Piecewise-linear* contrast adjustment:

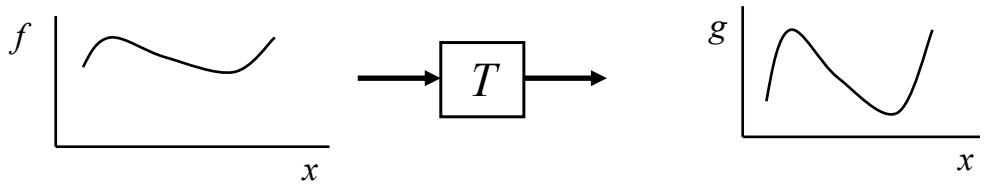






Point processing

- Change ***range*** of image: $g = T(f)$



- *Gamma correction:* $g = af^{\gamma}$

$\gamma = 1/2$ | Original | $\gamma = 2$

Source

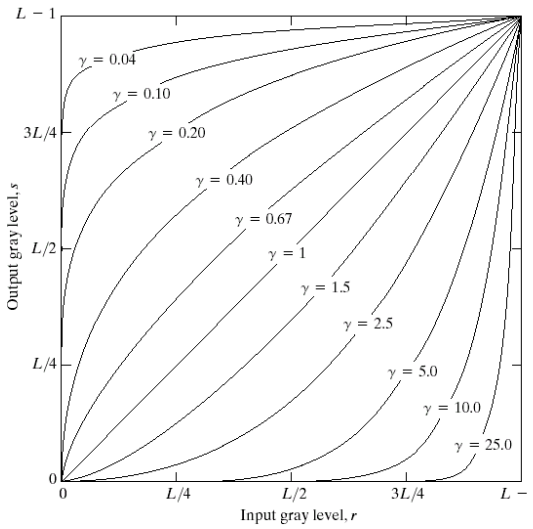


Image filtering

- Roughly speaking, replace image value at x with some function of values in its spatial neighborhood $N(x)$:

$$g(x) = T(f(N(x)))$$

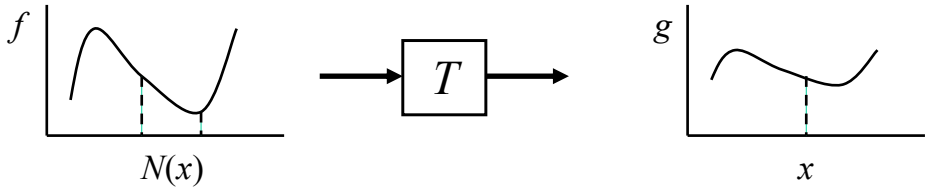
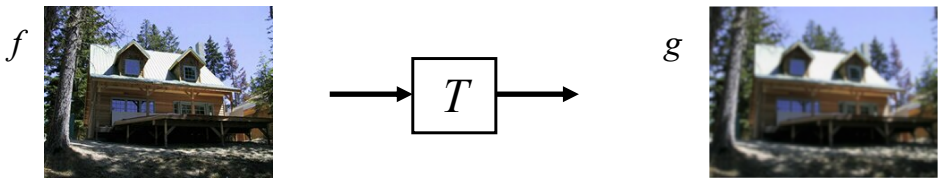


Image filtering

- Roughly speaking, replace image value at x with some function of values in its spatial neighborhood $N(x)$:

$$g(x) = T(f(N(x)))$$



- Examples: smoothing, sharpening, edge detection, etc.

Image warping

- Change ***domain*** of image:

$$x' = T(x), \qquad g(x') = f(x)$$

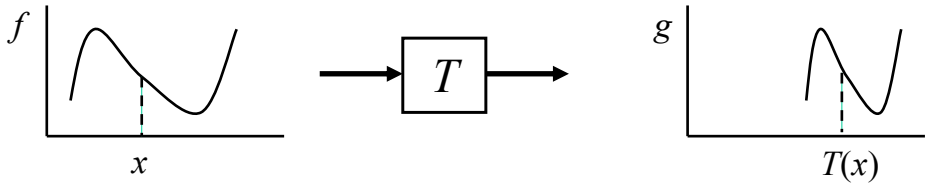


Image warping

- Change ***domain*** of image:

$$x' = T(x), \qquad g(x') = f(x)$$

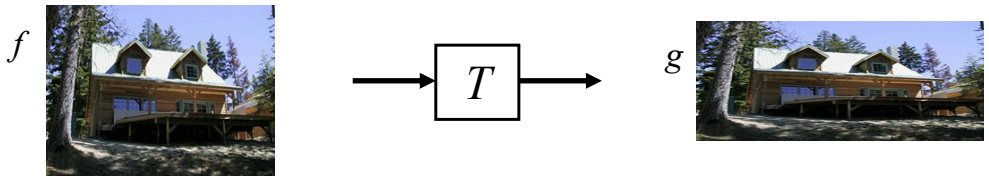


Image warping

- Examples of *global parametric* warps:

translation

rotation

scaling (uniform or non-uniform)

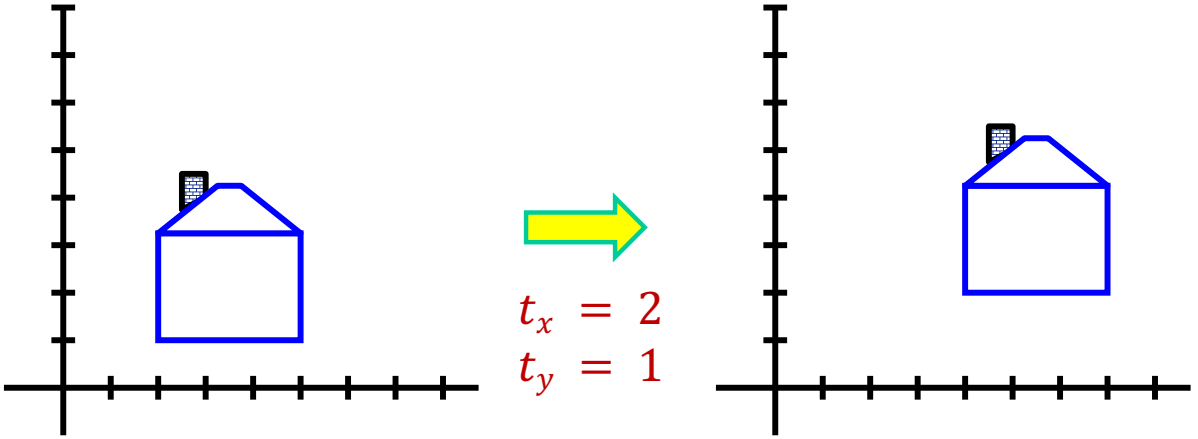
affine

perspective
homography

cylindrical
(not covered
in this class)

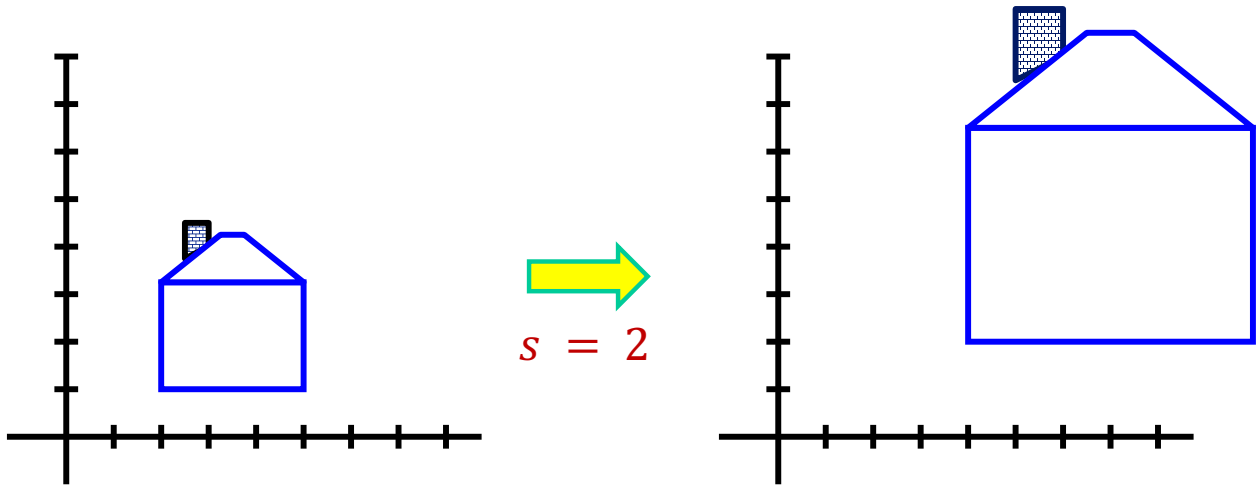
Translation

$$x' = x + t_x$$
$$y' = y + t_y$$



Uniform scaling

$$x' = s * x$$
$$y' = s * y$$

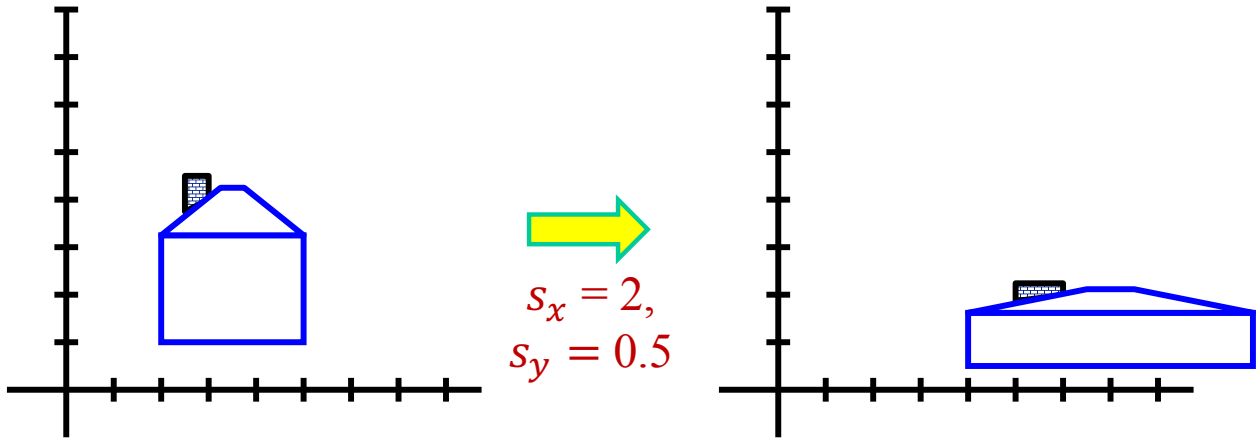


Non-uniform scaling

$$x' = s_x * x$$
$$y' = s_y * y$$

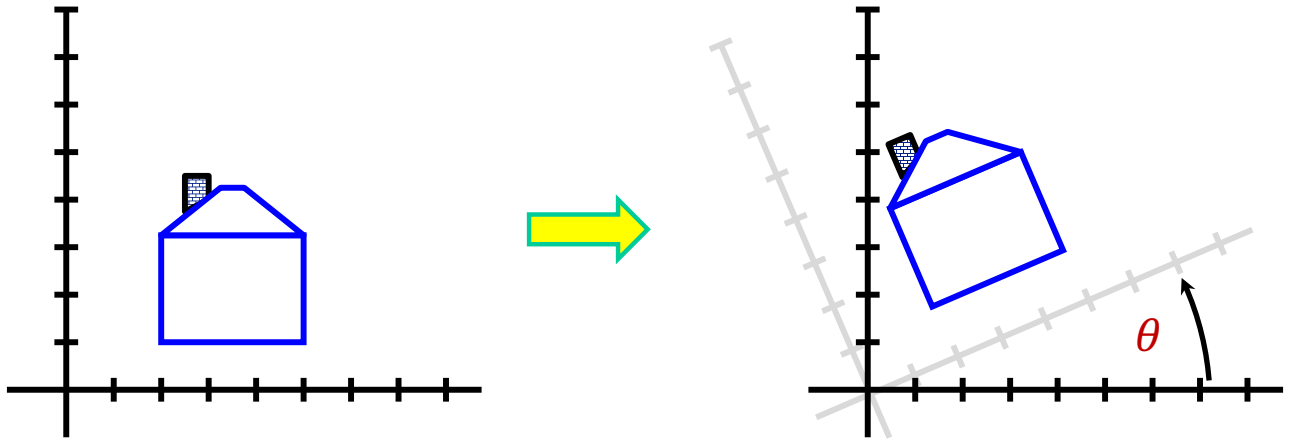
In matrix form:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



Rotation

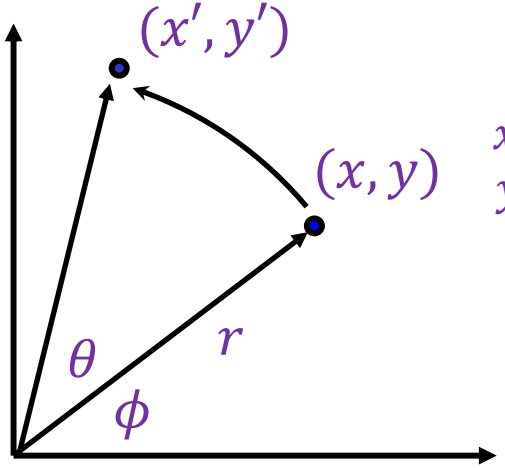
- Rotate the image by an angle of θ about the origin:



Rotation

- Rotate the image by an angle of θ about the origin:

$$x' = r\cos(\phi+\theta) = r\cos(\phi)\cos(\theta) - r\sin(\phi)\sin(\theta) = x\cos(\theta) - y\sin(\theta)$$
$$y' = r\sin(\phi+\theta) = r\sin(\phi)\cos(\theta) + r\cos(\phi)\sin(\theta) = x\sin(\theta) + y\cos(\theta)$$



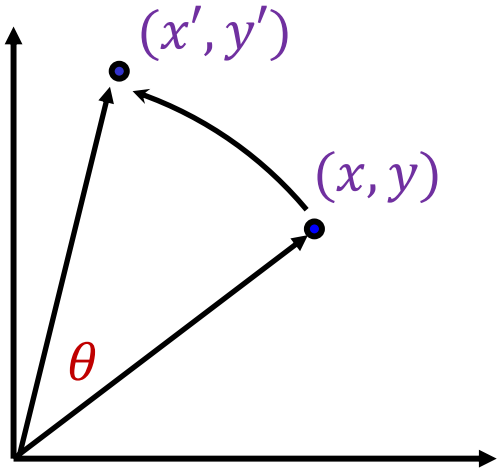
Apply trig identities

Substitute

$$x = r\cos(\phi)$$
$$y = r\sin(\phi)$$

Rotation

- Rotate the image by an angle of θ about the origin:



$$x' = x\cos(\theta) - y\sin(\theta)$$
$$y' = x\sin(\theta) + y\cos(\theta)$$

In matrix form:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Rotation

- 2D rotation in matrix form:

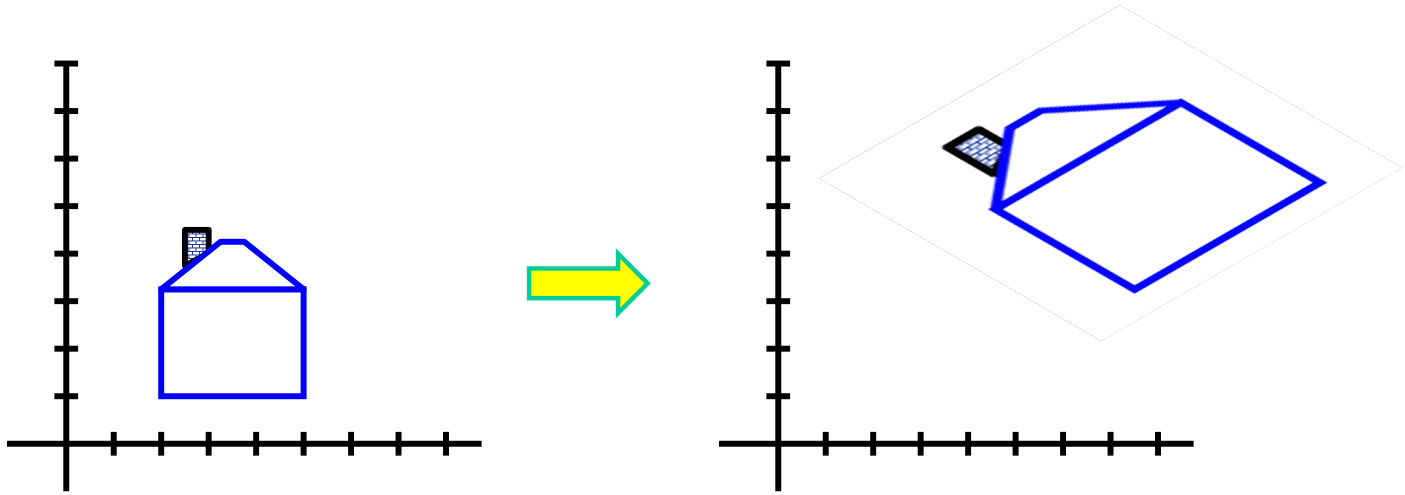
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{R(\theta)} \begin{pmatrix} x \\ y \end{pmatrix}$$

- Note: even though $\cos(\theta)$ and $\sin(\theta)$ are *nonlinear* functions of θ, x' and y' are *linear* combinations of x and y
- What is the inverse transformation?
  - Rotation by $-\theta$
  - For rotation matrices, $R^{-1} = R^T$

Affine transformation

- Combination of translation, non-uniform scaling, rotation, and *shear*

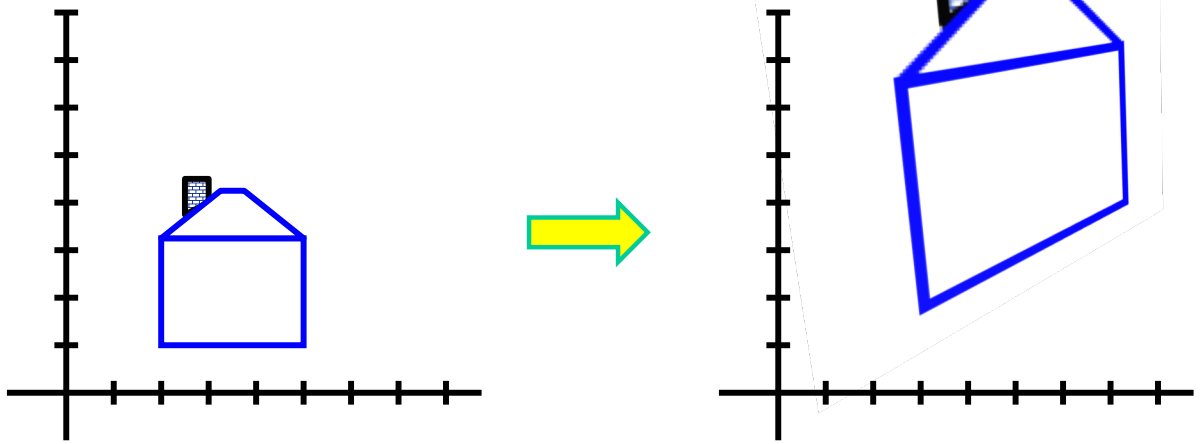
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$



Projective homography

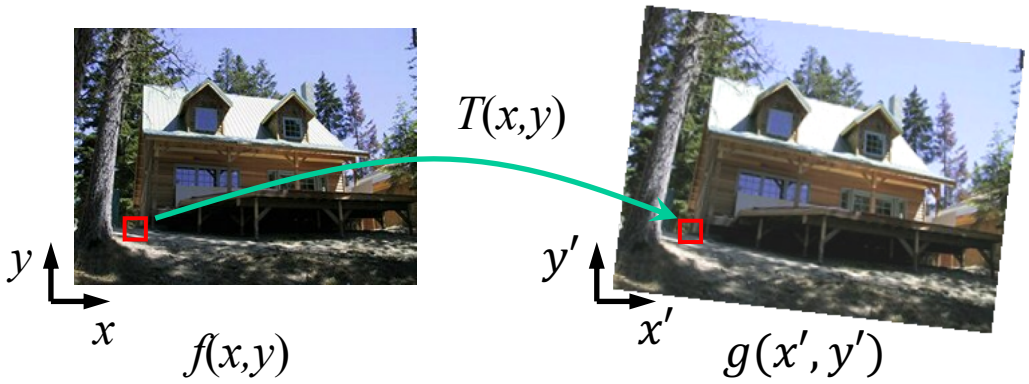
- A transformation that preserves straight lines, but not parallelism

$$x' = \frac{ax + by + c}{gx + hy + i}, \qquad y' = \frac{dx + ey + f}{gx + hy + i}$$



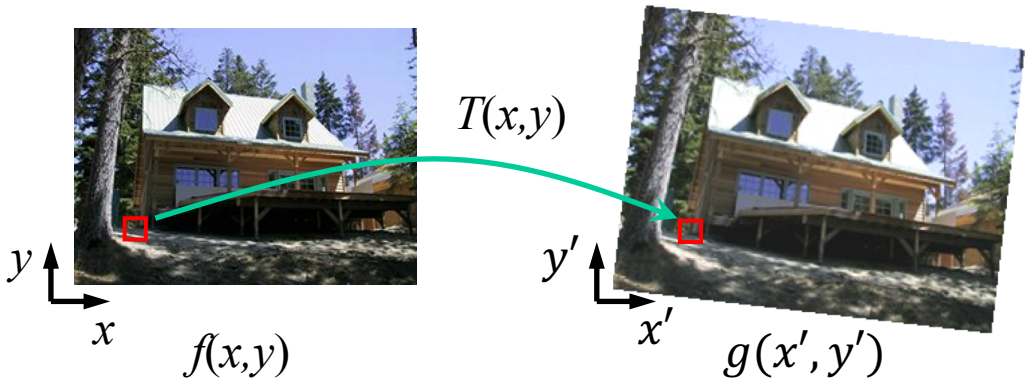
Warping pixels

- Given a coordinate transform $(x', y') = T(x, y)$ and a source image $f(x, y)$, how do we compute a transformed image $g(x', y') = f(T(x, y))$?



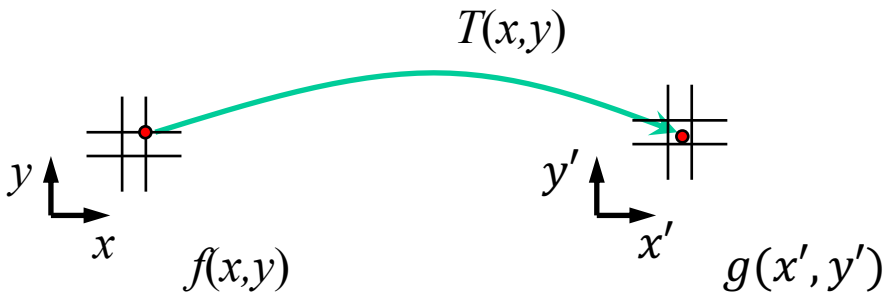
Forward warping

- Send each pixel $f(x, y)$ to its corresponding location $(x', y') = T(x, y)$ in the second image



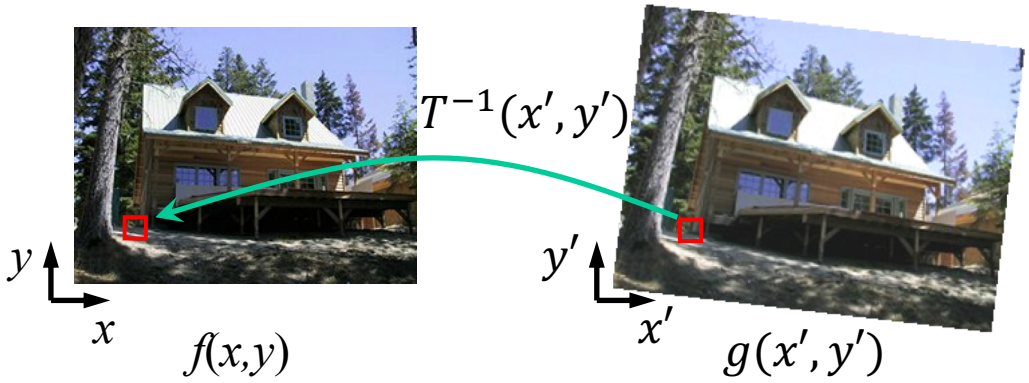
Forward warping

- Send each pixel $f(x, y)$ to its corresponding location $(x', y') = T(x, y)$ in the second image
  - What if (x', y') is not an integer location on the pixel grid?
  - We can "distribute" colors among the neighboring grid locations – known as *splatting*



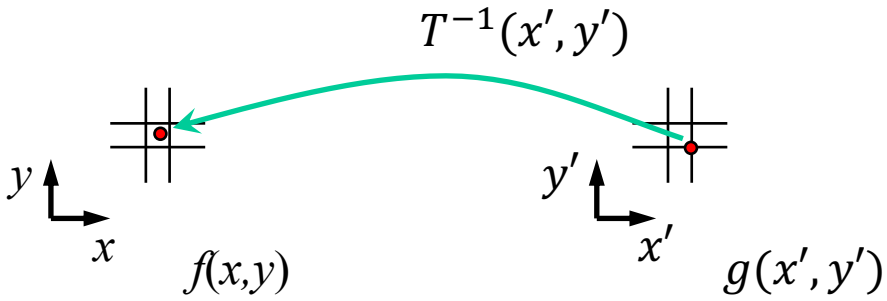
Inverse warping

- For each pixel grid location (x', y') in the second image, get the color from its corresponding location $(x, y) = T^{-1}(x', y')$ in the first image



Inverse warping

- For each pixel grid location (x', y') in the second image, get the color from its corresponding location $(x, y) = T^{-1}(x', y')$ in the first image
  - What if (x, y) is not an integer location on the original pixel grid?
  - Interpolate!



Forward vs. inverse warping

- Which is better?
  - Usually inverse: more efficient, doesn’t have a problem with holes
  - However, it requires an invertible warp function, which is not always possible