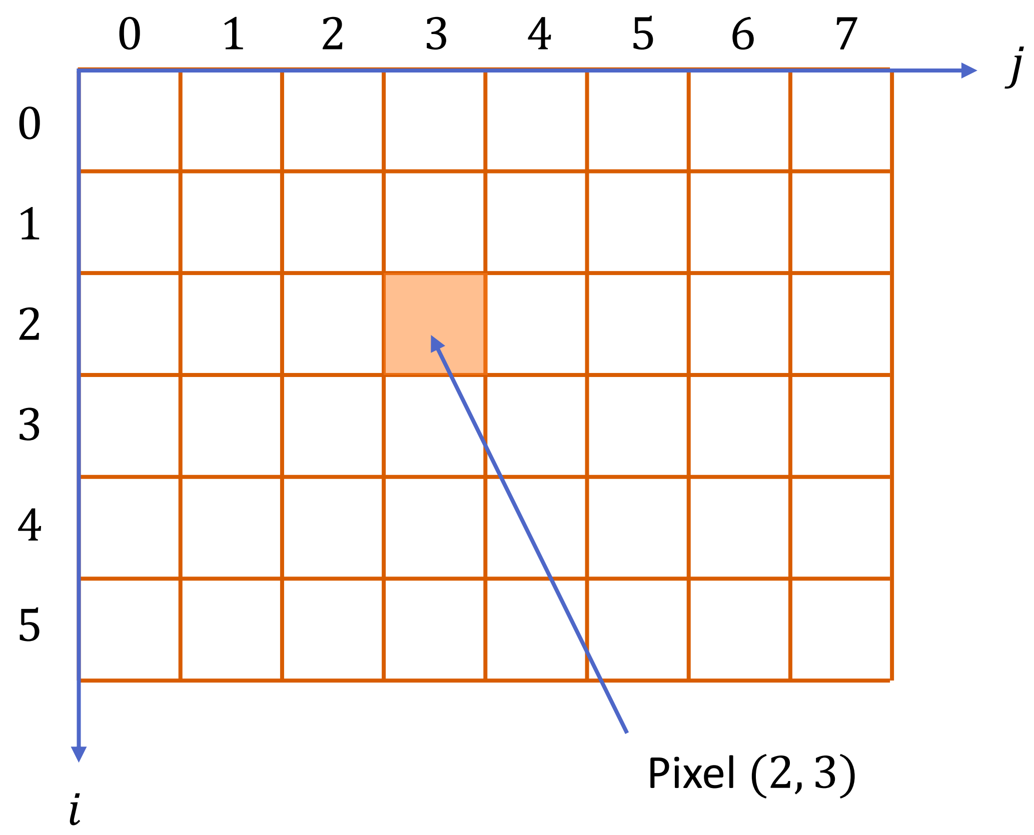
EN3160 L02
Basics and Point Operations

What is a Digital Image?

- A grayscale digital image is a rectangular array of numbers which represent pixels.
- Each pixel can take an integer value in the range [0, 255] , for an 8-bit image.
- If the image is a color image, then there would be three such arrays.
- The size of this array is actually the resolution of the image, e.g., example, 3712×5568.
- We take the top-left pixel as the (0, 0) pixel and vertical axis as the i axis.



What is a Color Digital Image?

- The grayscale image that we considered as a two-dimensional array, or a single plane.
- A color image has three such planes, one for blue, one for green and one for red. We call such an image an BGR image.
- If we access a pixel location such as (2, 3), we will get three values, B, G, and R.
- Each value is in [0, 255]. In this context, $2^8 \times 2^8 \times 2^8$ different colors are possible.

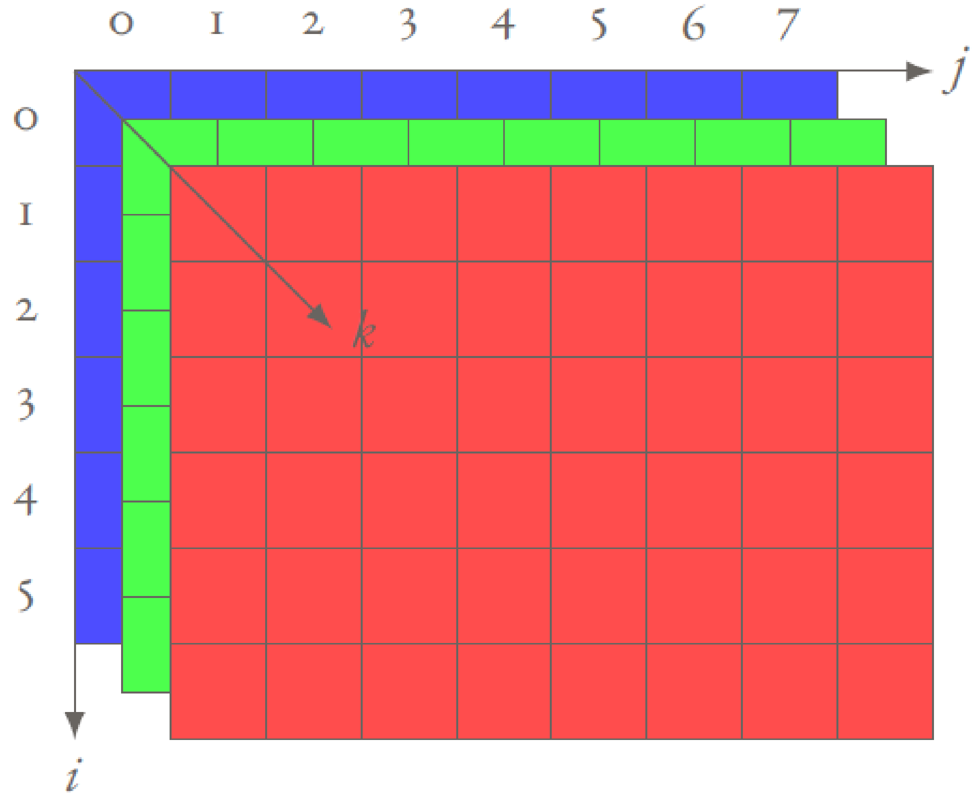
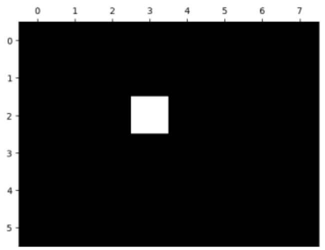


Image Resolution and DPI

- Image Resolution:
  - Number of pixels in an image (width × height)
  - Example: 1920×1080 (Full HD) = 2,073,600 pixels
  - Impacts the image detail and sharpness, file size and processing time
- Dots Per Inch (DPI):
  - Number of printed dots in one inch, i.e., physical point density
  - Typical values: screen: ~72–96 DPI, print: 300 DPI or higher

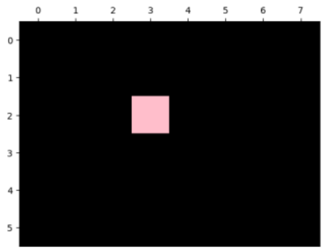
Creating a 6 × 8 Image

A Grayscale Image




```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
im = np.zeros((6,8), dtype=np.uint8)
im[2,3] = 255
fig, ax = plt.subplots(1, 1, figsize=(6, 8))
ax.imshow(im, cmap='gray', vmin=0, vmax=255)
ax.xaxis.set_ticks_position('top')
plt.show()
```

A Color Image




```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
im = np.zeros((6,8,3), dtype=np.uint8)
im[2,3] = (255, 190, 203)
fig, ax = plt.subplots(1, 1, figsize=(6, 8))
ax.imshow(im)
ax.xaxis.set_ticks_position('top')
plt.show()
```

Image Opening and Displaying

Displaying Using Matplotlib

```python
import cv2 as cv
import matplotlib.pyplot as plt
im = cv.imread('images/jolie.png')
fig, ax = plt.subplots(1, 1, figsize=(6, 8))
ax.imshow(cv.cvtColor(im, cv.COLOR_BGR2RGB))
ax.xaxis.set_ticks_position('top')
plt.show()
```

Displaying Using OpenCV

```python
import cv2 as cv
im = cv.imread('images/jolie.png')
cv.namedWindow('Image', cv.WINDOW_AUTOSIZE)
cv.imshow('Image', im)
cv.waitKey(0)
cv.destroyAllWindows()
```

Q:Why do we need to cv.cvtColor only when displaying using Matplotlib?

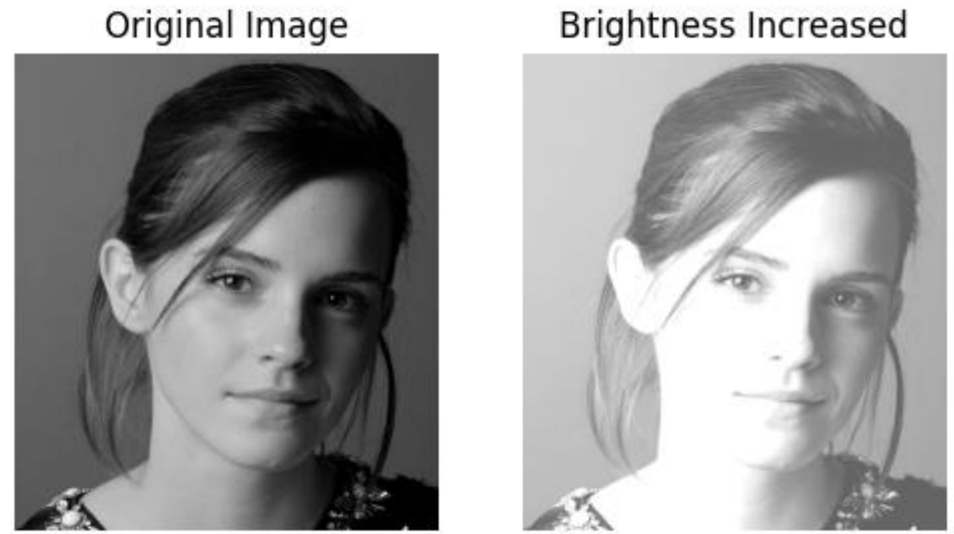
Displaying Image Properties

```
import cv2 as cv
import matplotlib.pyplot as plt
im = cv.imread('images/ryan.jpg')
fig, ax = plt.subplots(1, 1, figsize=(6, 8))
ax.imshow(cv.cvtColor(im, cv.COLOR_BGR2RGB))
ax.xaxis.set_ticks_position('top')
plt.show()
print('Image Shape:', im.shape)
print('Image Data Type:', im.dtype)
print('Image Size:', im.size)
```

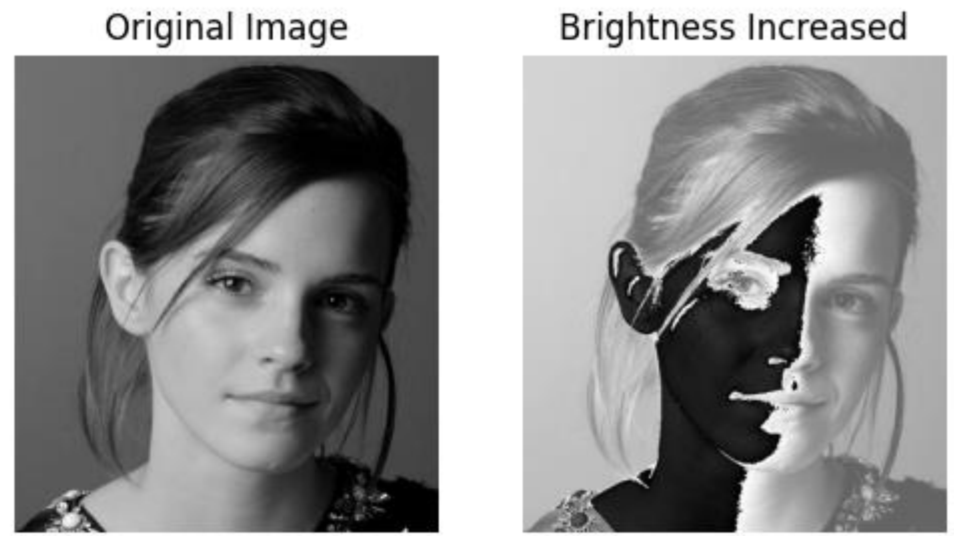
```
Image Shape: (2641, 1761, 3)
Image Data Type: uint8
Image Size: 13952403
```

Increasing the Brightness

```python
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
im1 = cv.imread('images/emma_gray.jpg', 
cv.IMREAD_GRAYSCALE)
im2 = cv.add(im1, 100)
fig, ax = plt.subplots(1, 2, figsize=(6, 8))
ax[0].imshow(im1, cmap='gray', vmin=0, vmax=255)
ax[0].set_title('Original Image')
ax[1].imshow(im2, cmap='gray', vmin=0, vmax=255)
ax[1].set_title('Brightness Increased')
for a in ax:
    a.axis('off')
plt.show()
```



What is wrong with this? `im2 = im1 + 100`

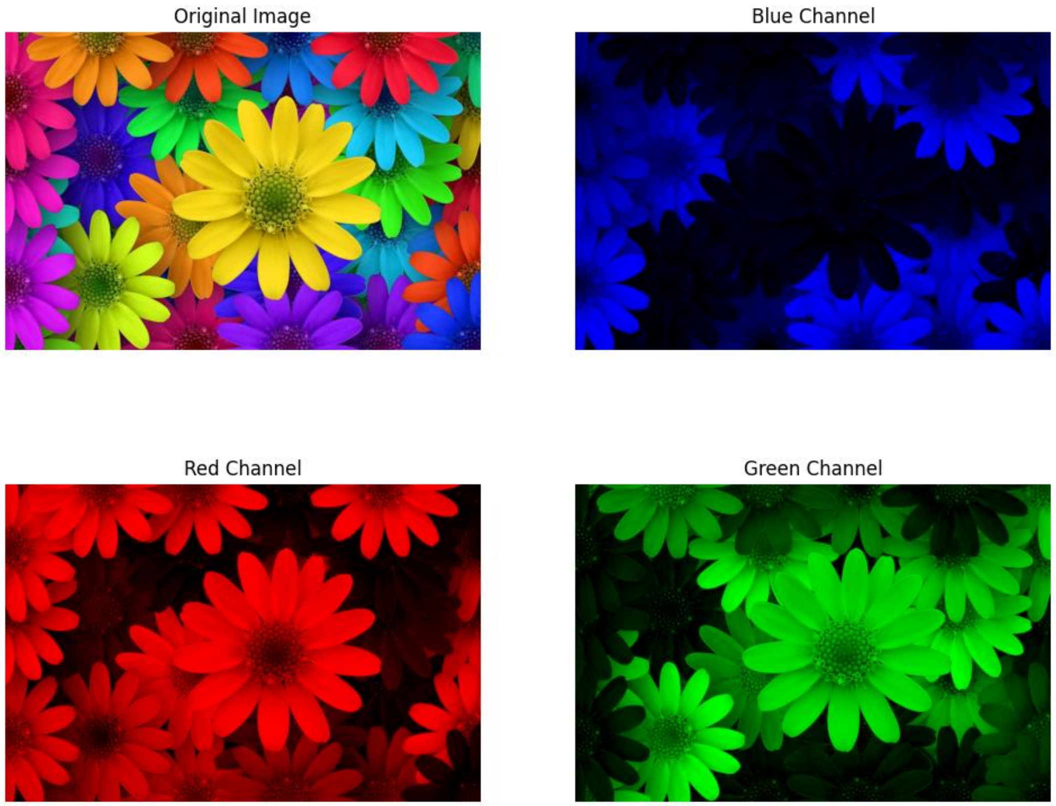


Increasing the Brightness Using Loops (Slow)

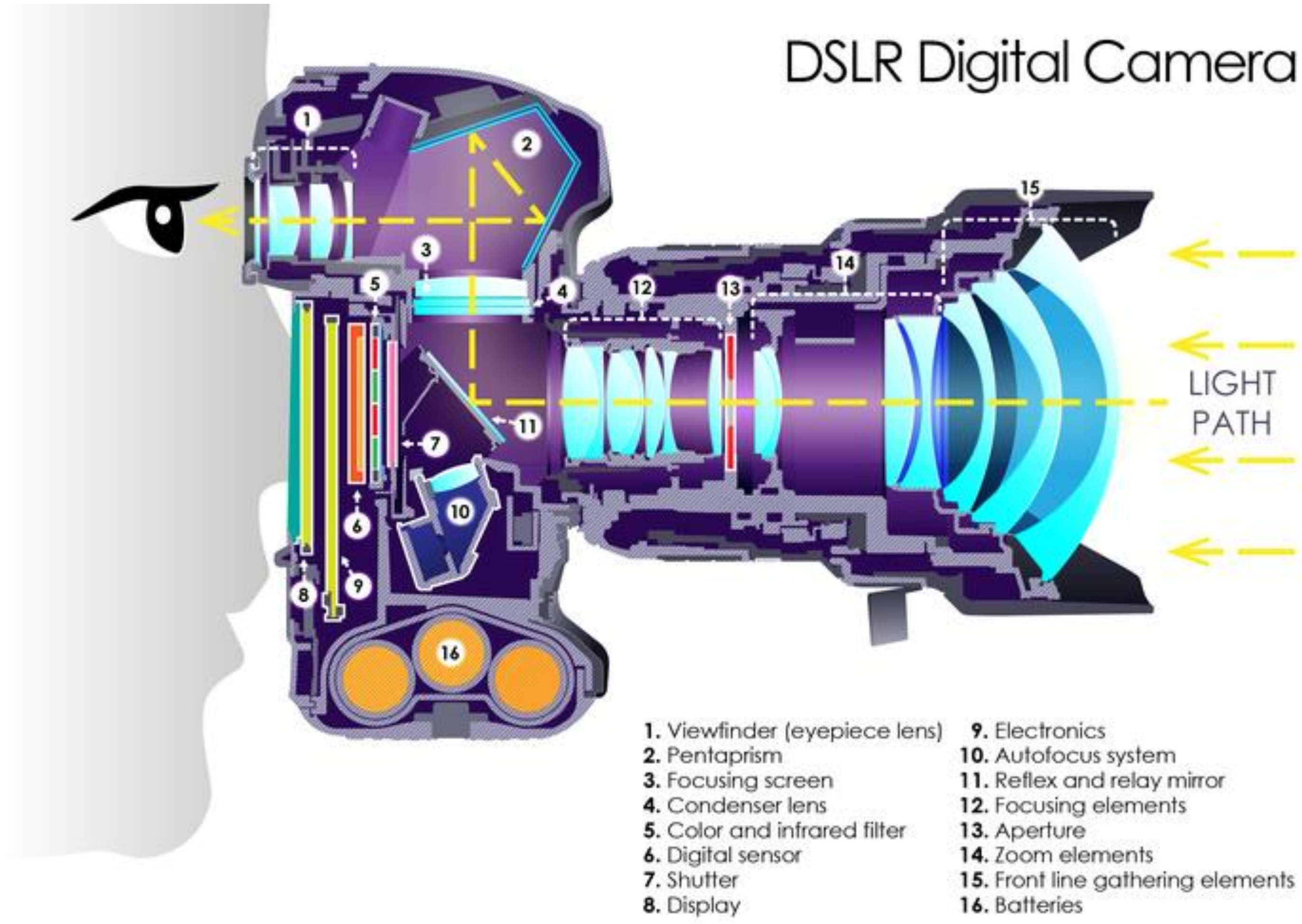
```python
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
im1 = cv.imread('images/emma_gray.jpg', cv.IMREAD_GRAYSCALE)
im2 = np.zeros_like(im1)
for i in range(im1.shape[0]):
    for j in range(im1.shape[1]):
        im2[i,j] = im1[i,j] + 100
fig, ax = plt.subplots(1, 2, figsize=(6, 8))
ax[0].imshow(im1, cmap='gray', vmin=0, vmax=255)
ax[0].set_title('Original Image')
ax[1].imshow(im2, cmap='gray', vmin=0, vmax=255)
ax[1].set_title('Brightness Increased')
for a in ax:
    a.axis('off')
plt.show()
```

Obtaining One Color Plane

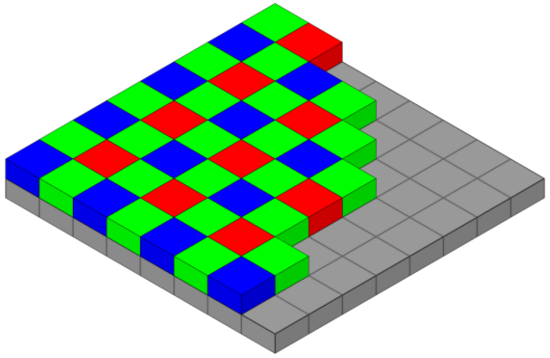
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
im = cv.imread('images/rgb_flowers.jpg')
im_blue = im.copy()
im_blue[:,:,1] = 0
im_blue[:,:,2] = 0
fig, ax = plt.subplots(1, 2, figsize=(12, 8))
ax[0].imshow(cv.cvtColor(im, cv.COLOR_BGR2RGB))
ax[0].set_title('Original Image')
ax[1].imshow(cv.cvtColor(im_blue, cv.COLOR_BGR2RGB))
ax[1].set_title('Blue Channel')
for a in ax:
    a.axis('off')
plt.show()
```

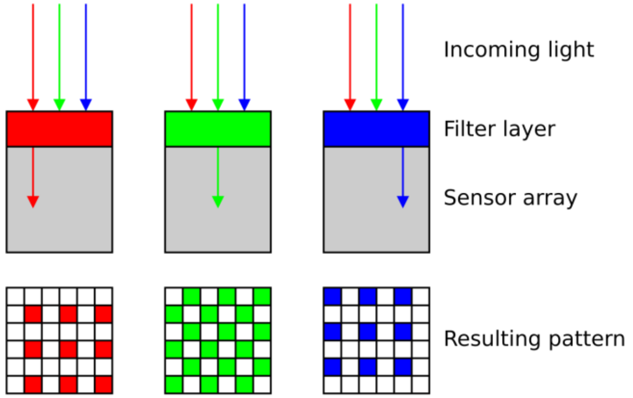


Working of a Camera



Bayer arrangement of color filters

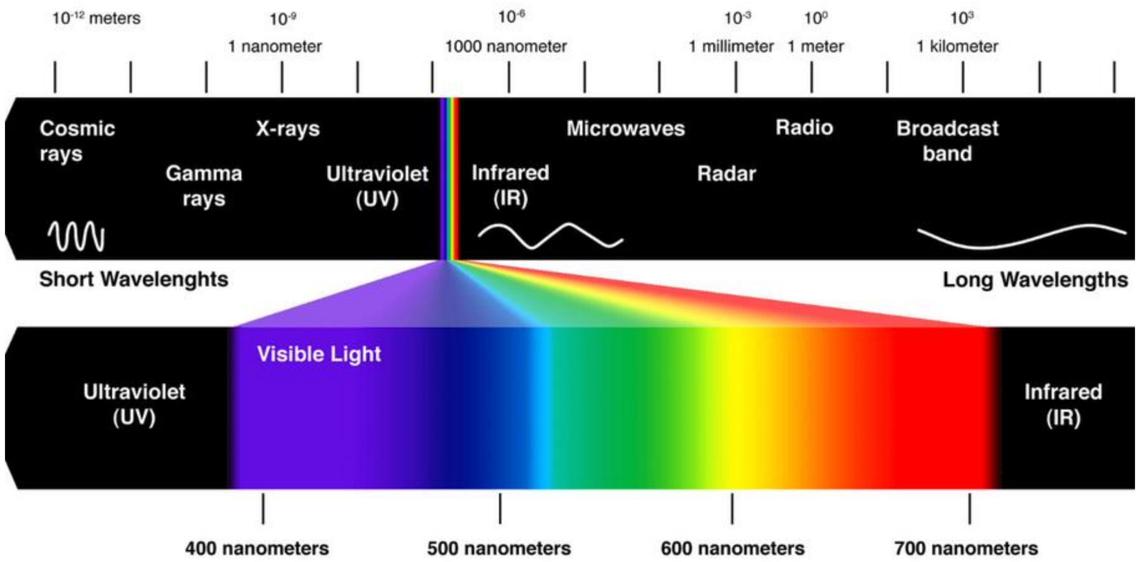




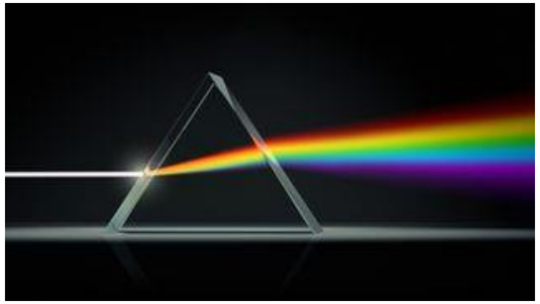
Demosaicing: Estimation of missing components from neighboring values

https://en.wikipedia.org/wiki/Bayer_filter
https://commons.wikimedia.org/wiki/File:Cross_section_of_a_DSLR_camera.png

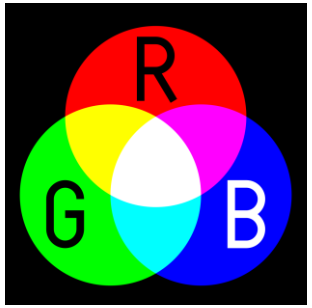
Color



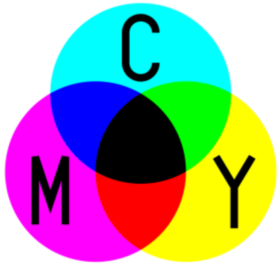
Wavelengths comprising the visible range of the electromagnetic spectrum.



Color spectrum seen by passing white light through a prism.



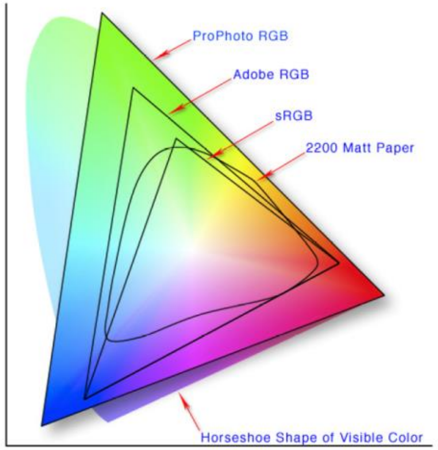
Additive mixing

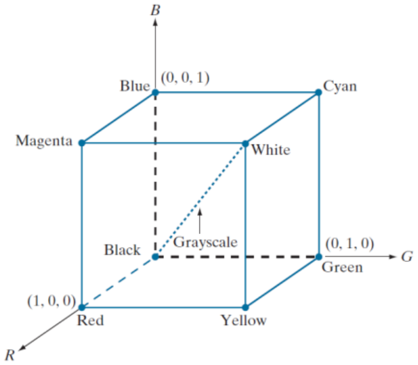


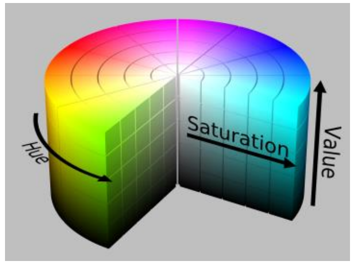
Subtractive mixing

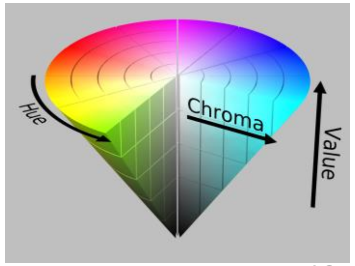
A **color model** (color space or color system) facilitates the specification of colors: (1) a coordinate system, and (2) a subspace within that system, such that each color in the model is represented by a single point contained in that subspace.

RGB: image capturing in a camera, wavelength-based
CMYK (Cyan, Magenta, Yellow, Black): for printing
HSV (Hue, Saturation, Value): how humans describe color, decouples the color and gray-scale information









Billmeyer and Saltzman's Principles of Color Technology
Gonzalez and Woods, chapter 6, https://en.wikipedia.org/wiki/Color_space
https://en.wikipedia.org/wiki/HSL_and_HSV, https://www.livescience.com/50678-visible-light.html
https://www.quora.com/What-is-visible-light-and-how-much-of-the-spectrum-does-it-cover-compared-to-the-others

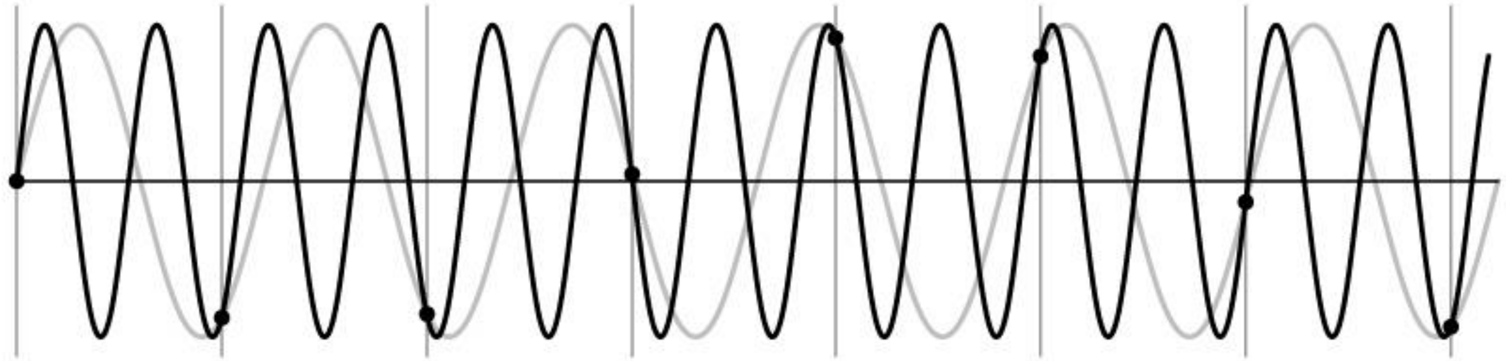
Summary

1. A grayscale digital image is an array of 8-bit unsigned integers in [0, 255]. We call each integer a pixel.
2. Color images have three such arrays (planes), one for red, one for green, and one for blue.
3. We can manipulate an image using OpenCV function or a pair of for-loops to access each pixel (slower).

Sampling and Reconstruction

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)

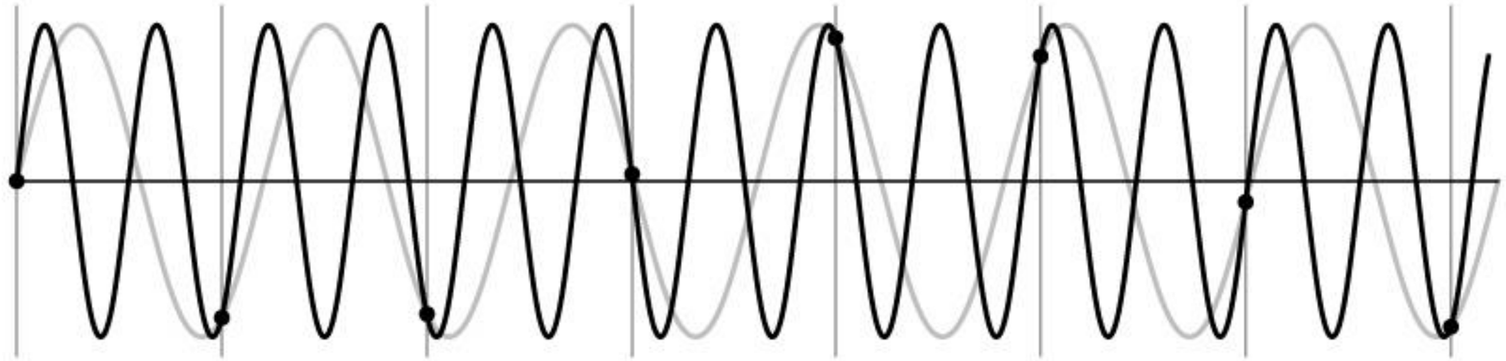


Source: S. Marschner (via A. Efros)

Source: Lazeknik

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)

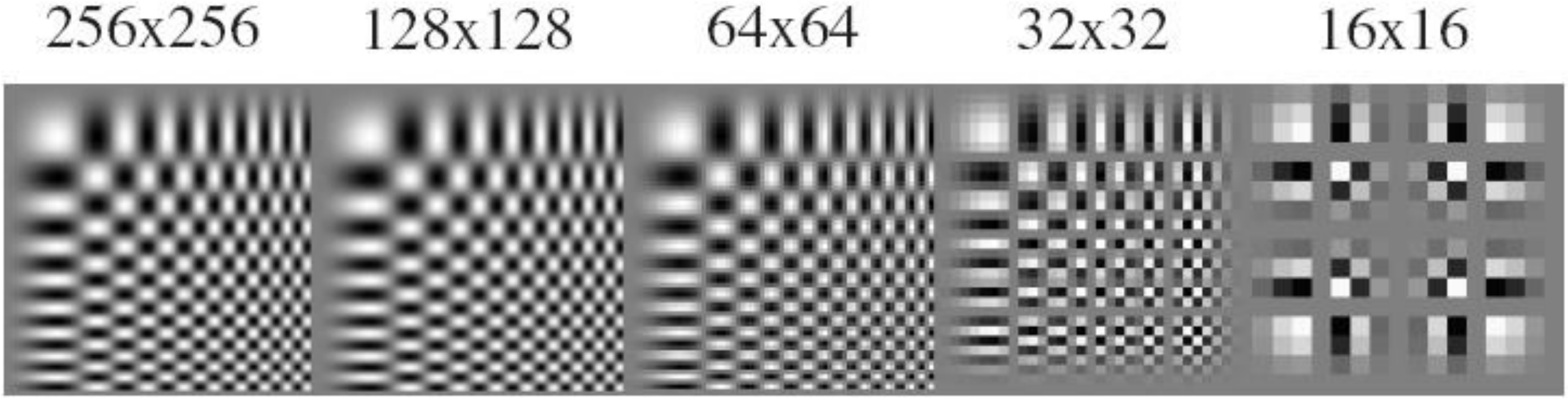
Source: Lazeknik

Wagon Wheel Effect

https://www.youtube.com/watch?v=6XwgbHjRo30

Source: Lazeknik

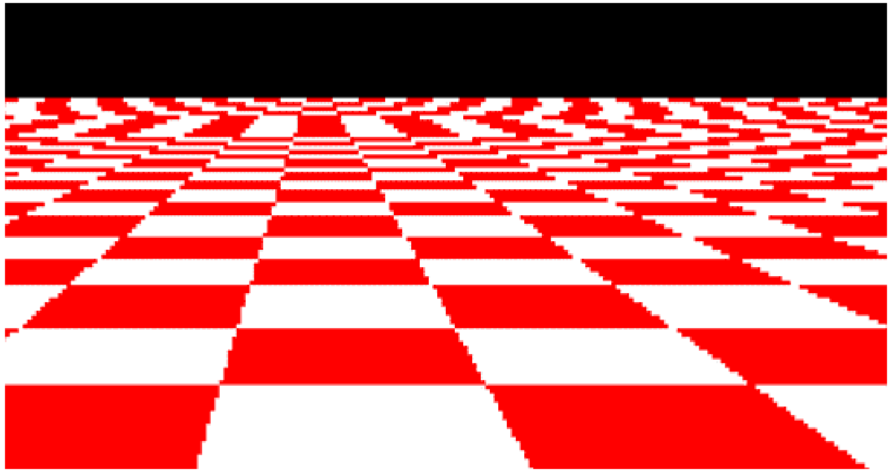
Aliasing in Images

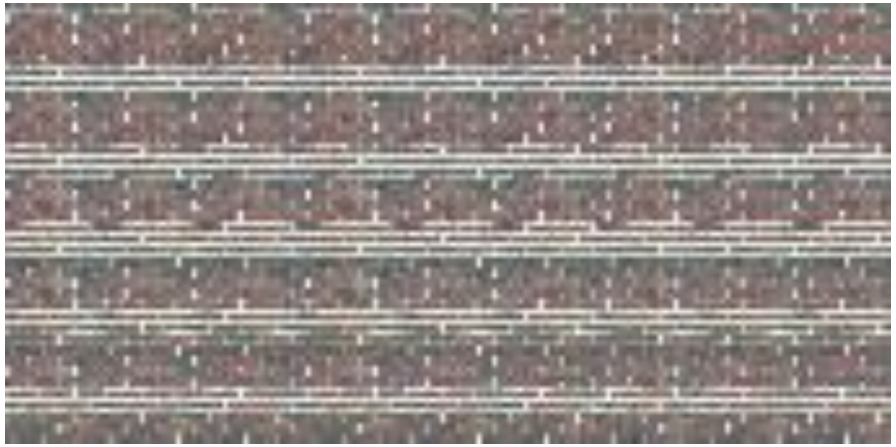


Source: Lazeknik

Aliasing “in the Wild”

Disintegrating textures

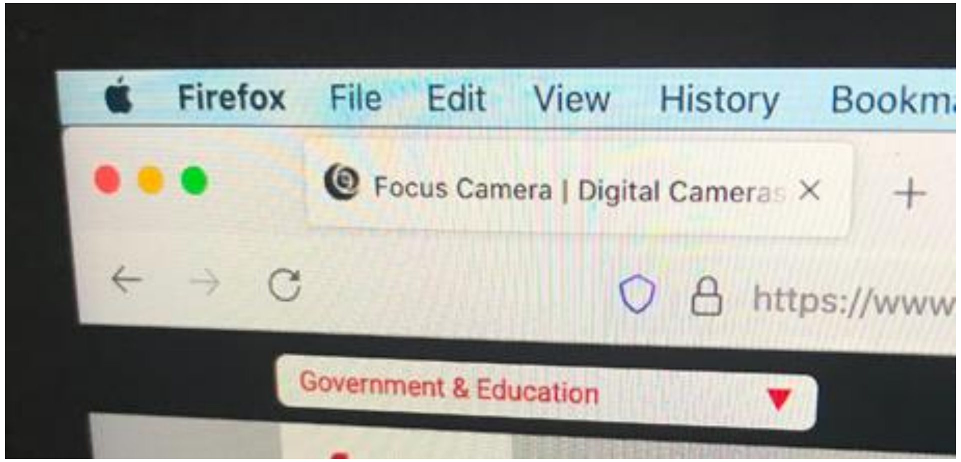




Source

Moire patterns, false color

Source



Source

Source: Lazeknik

Aliasing in Neural Networks

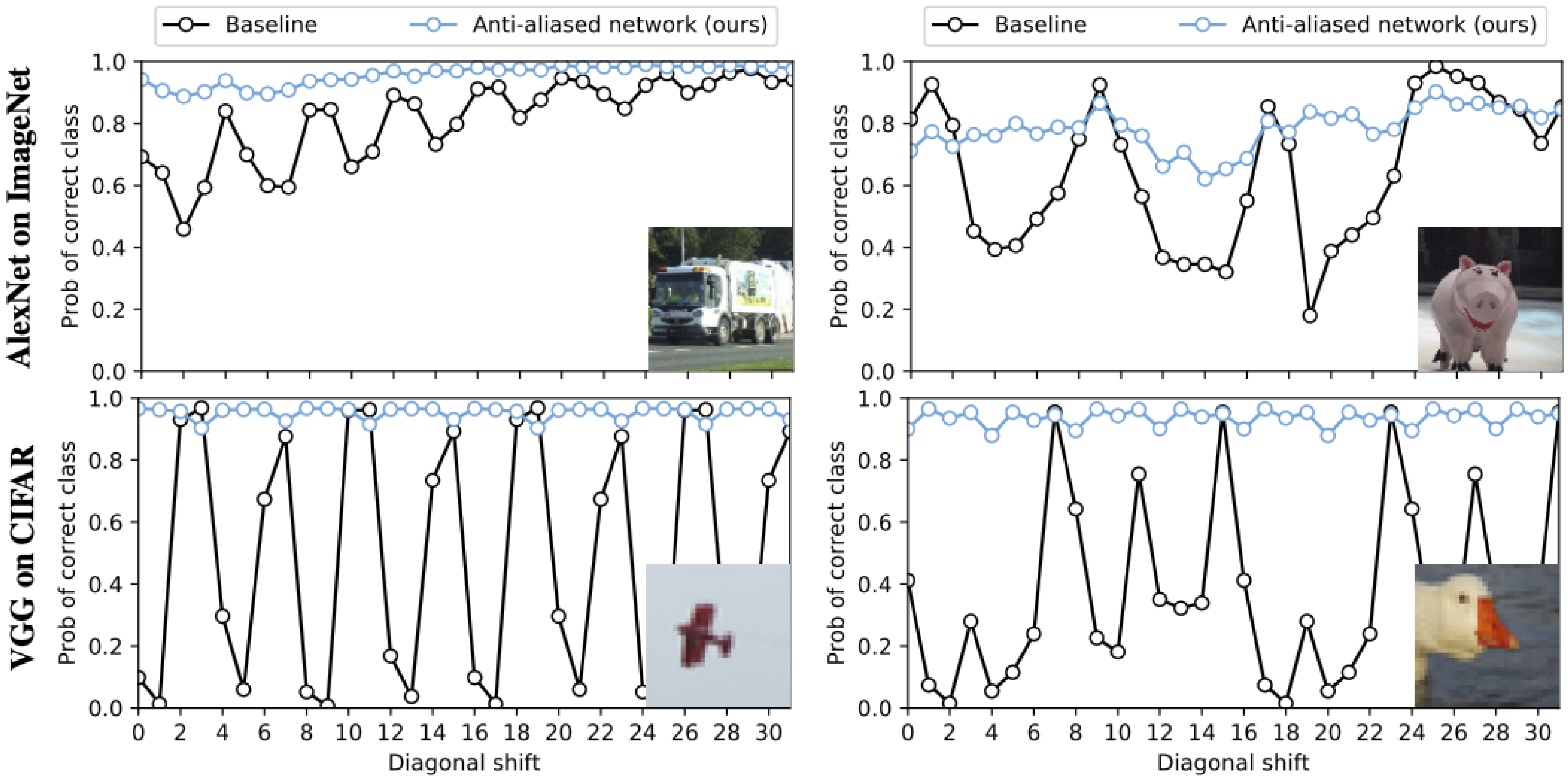
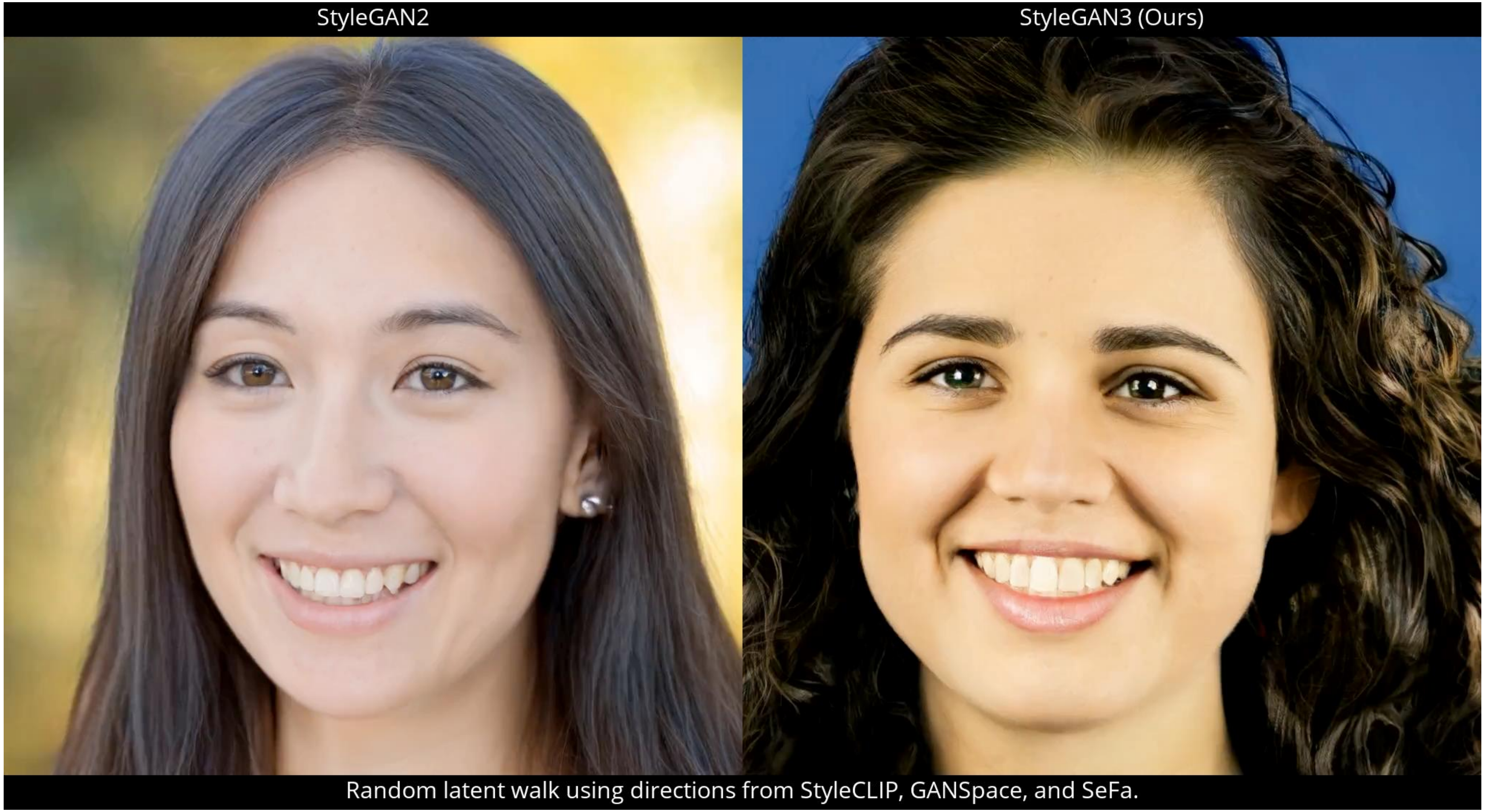


Figure 1. **Classification stability for selected images.** Predicted probability of the correct class changes when shifting the image. The baseline (black) exhibits chaotic behavior, which is stabilized by our method (blue). We find this behavior across networks and datasets. Here, we show selected 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

Source: Lazeknik

Aliasing in neural networks

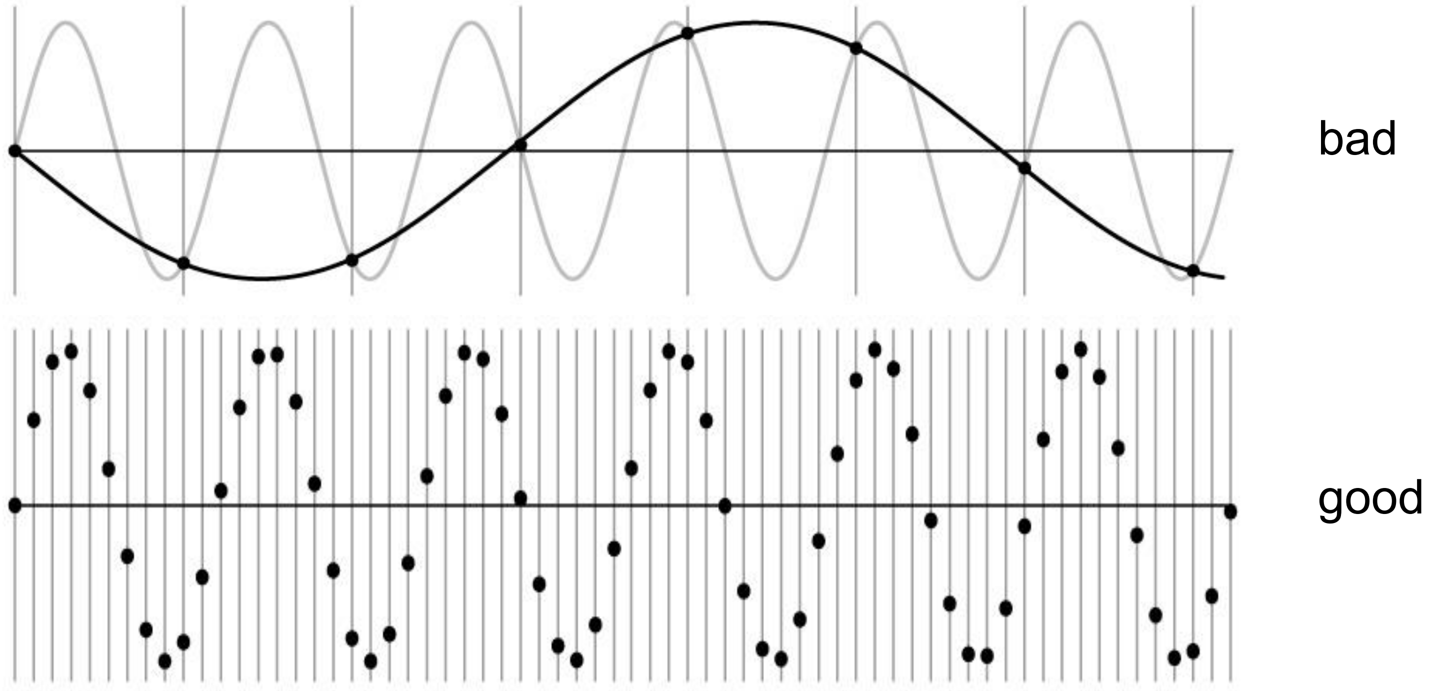


T. Karras et al. Alias-Free Generative Adversarial Networks. NeurIPS 2021

Videos

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

- How can we get rid of aliasing?
  - 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 **low-pass filter** (to be covered later)

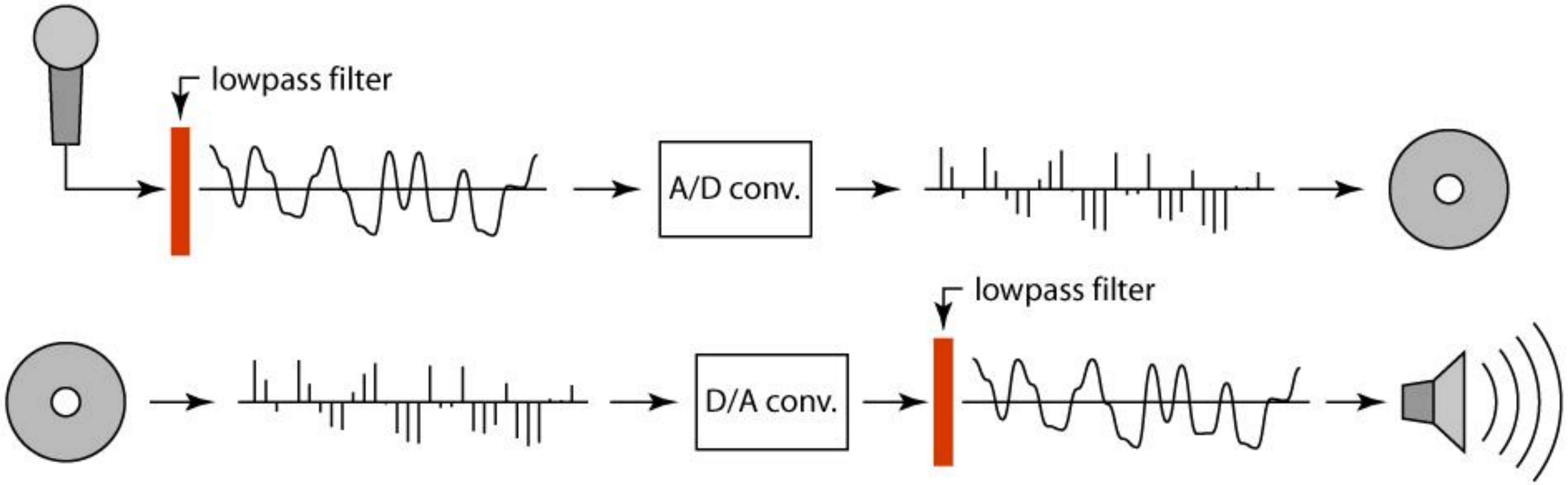
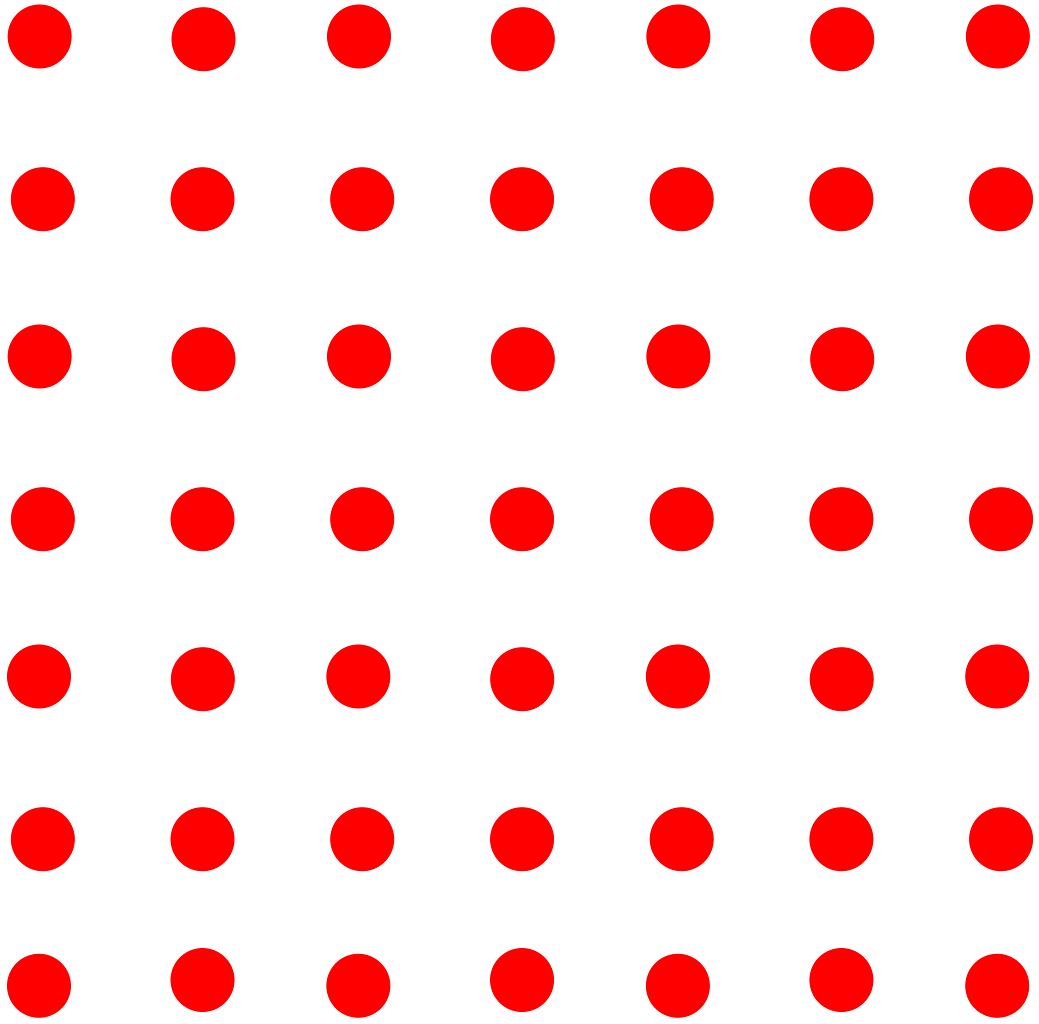


Image Resampling and Interpolation

Subsampling an Image

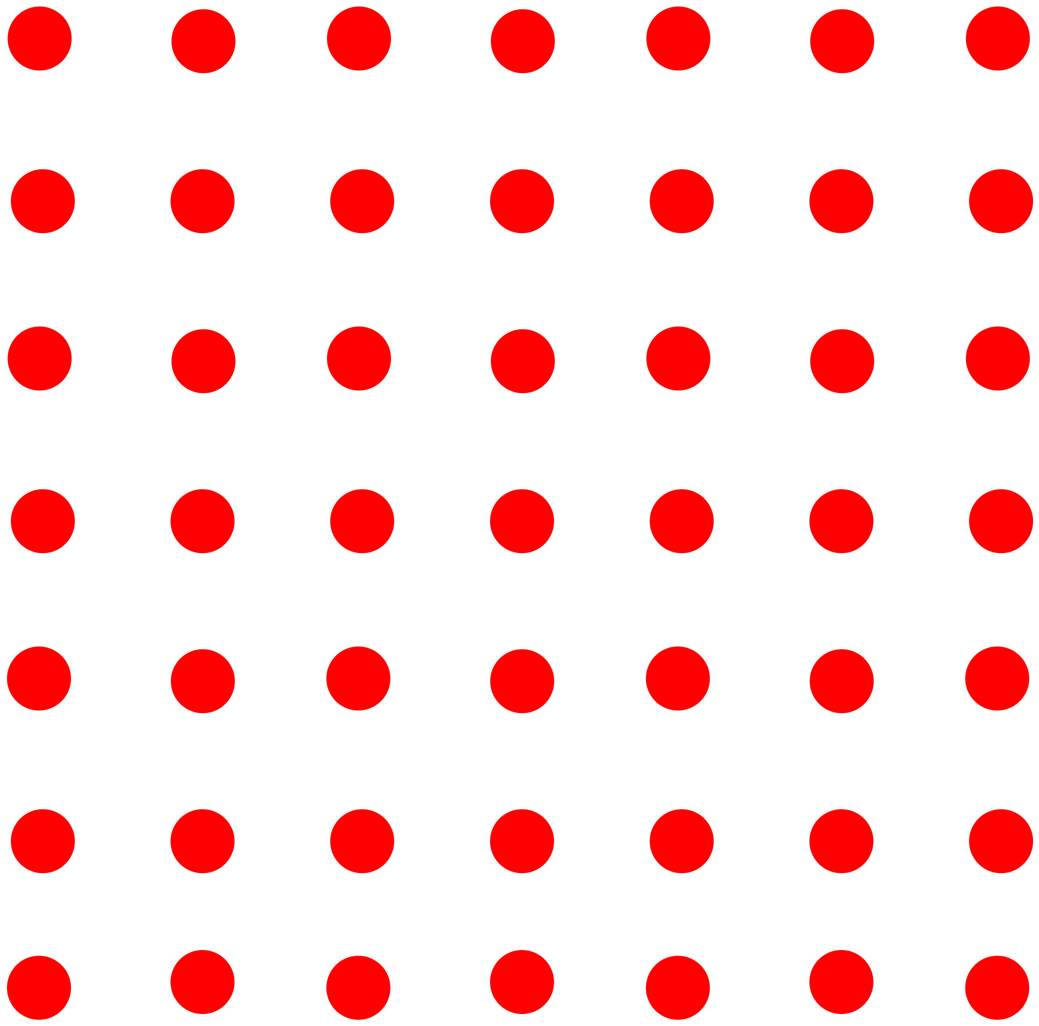
How do we reduce the size of an image by a factor of two?



Source: Lazeknik

Subsampling an image

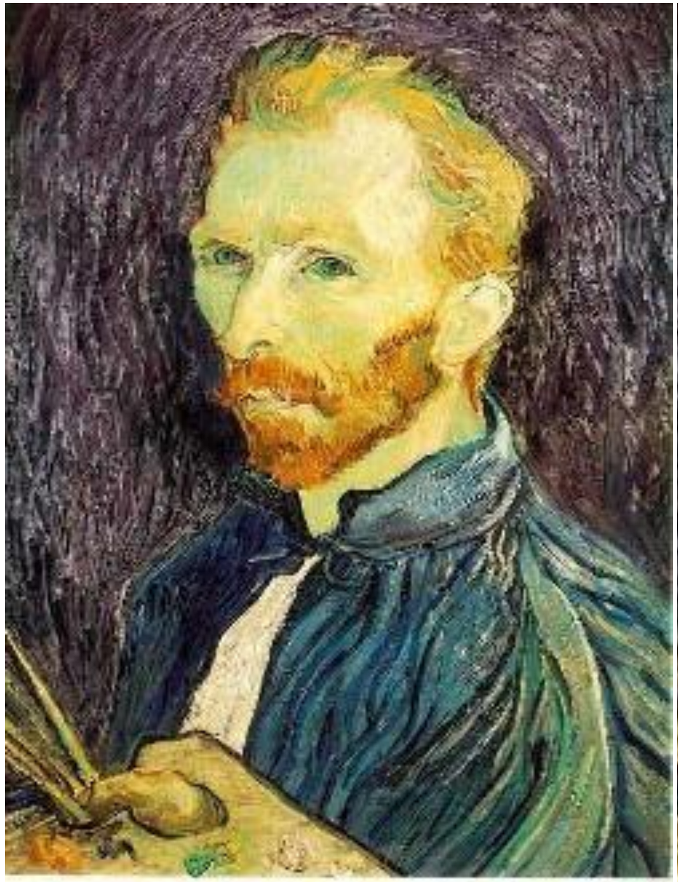
How do we reduce the size of an image by a factor of two?



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

Source: Lazeknik

Subsampling without Pre-filtering



1/2

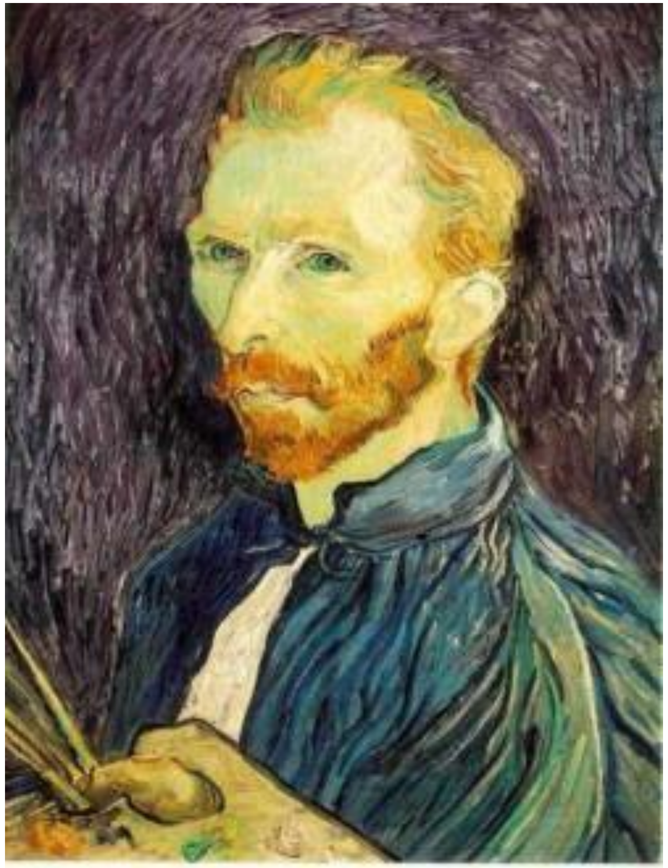
1/4 (2x zoom)

1/8 (4x zoom)

Source: S. Seitz (via D. Hoiem)

Source: Lazeknik

Subsampling with Pre-filtering



1/2 1/4 1/8

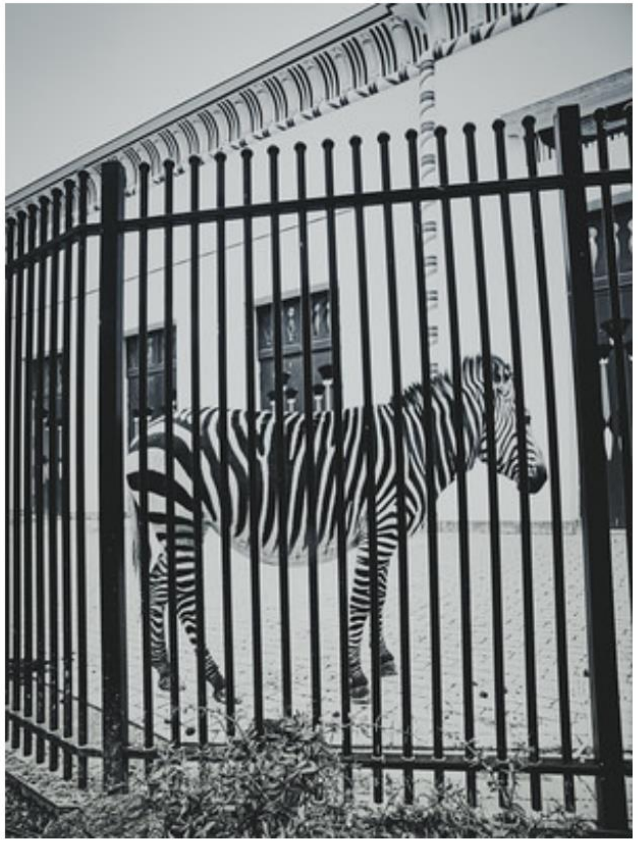
Image is smoothed with a *Gaussian filter* before subsampling

Source: Lazeknik

Source: S. Seitz (via D. Hoiem)

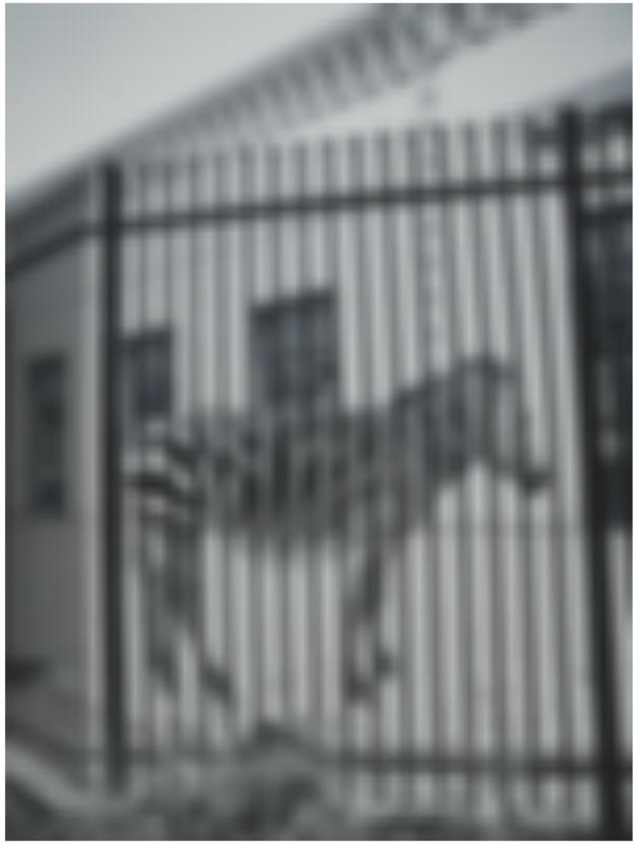
Subsampling with Pre-filtering

Image



Downsampled by 4

Smoothed then downsampled by 4

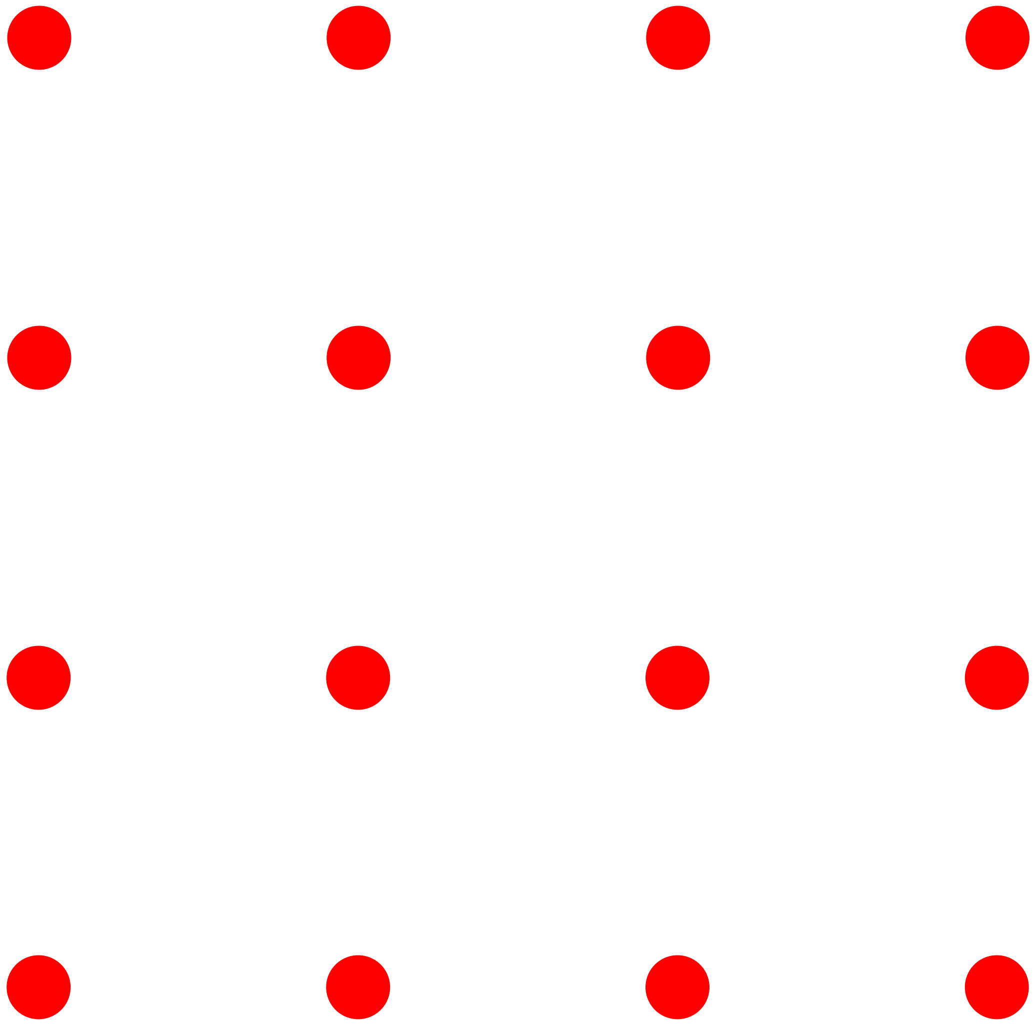


Source: Lazeknik

Source: D. Forsyth

Upsampling an Image

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

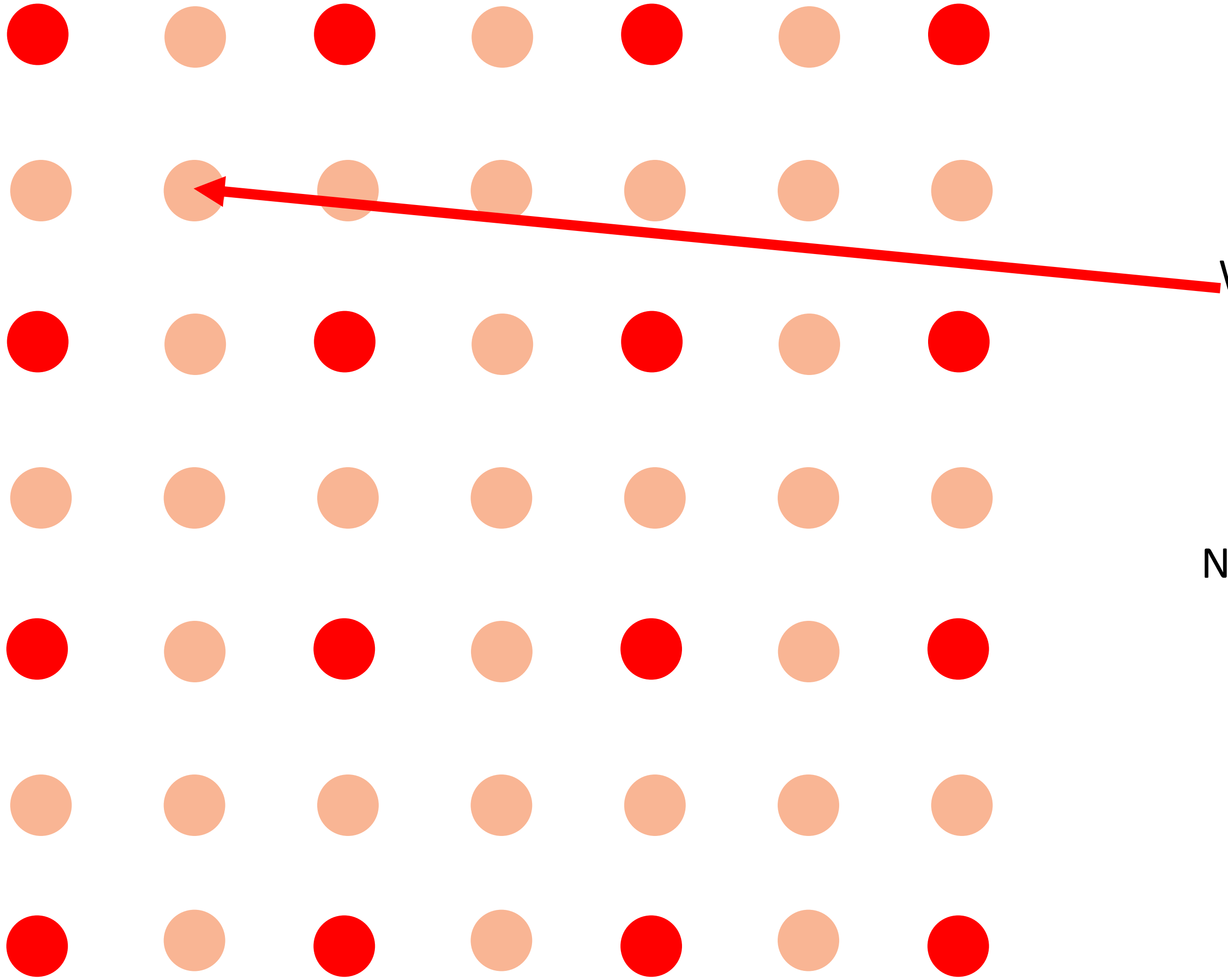


Let's increase the resolution of the sampling grid!

Source: Lazeknik

Upsampling an Image

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



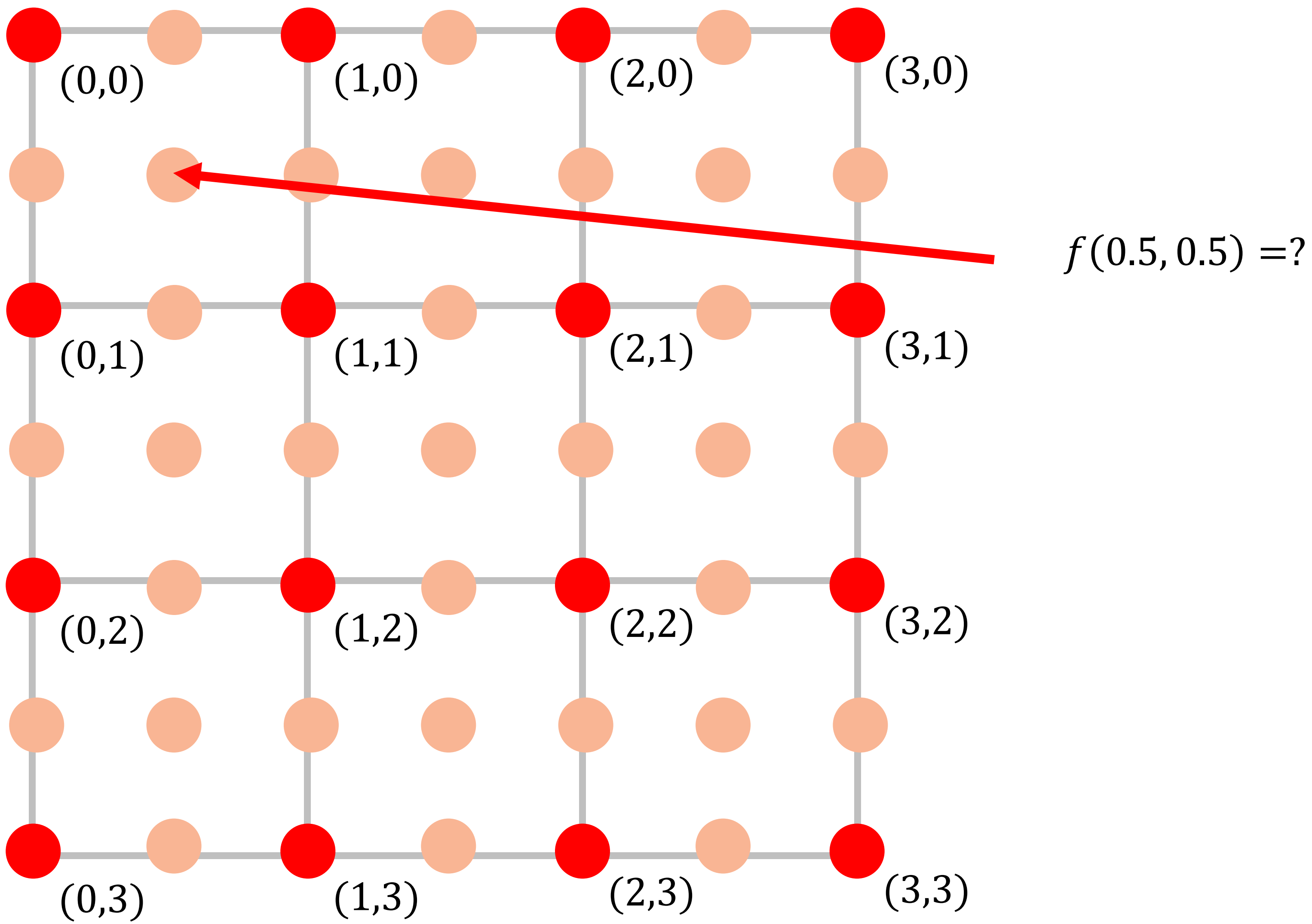
What should this value be?

Need to *interpolate*!

Source: Lazeknik

Upsampling an Image

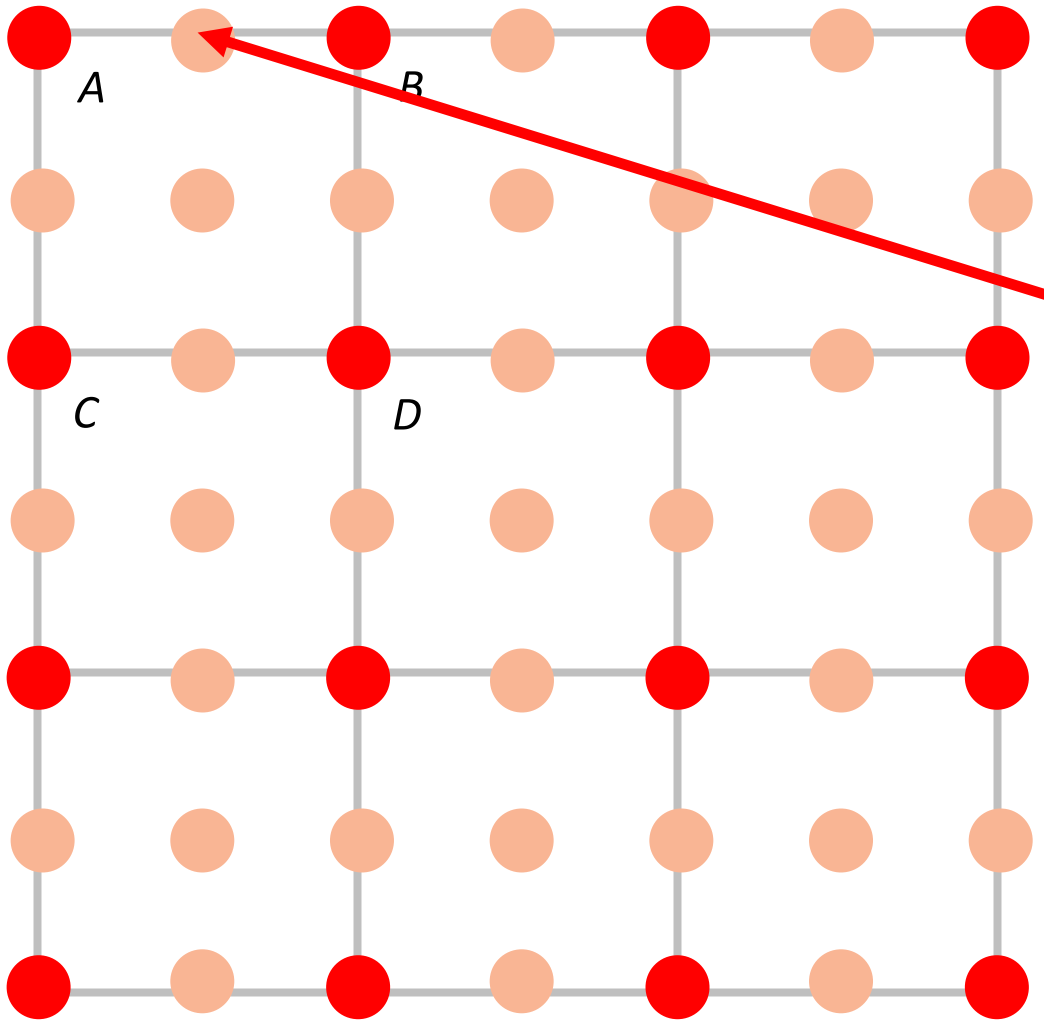
- How do we *increase* the size of an image by a factor of two?



Source: Lazeknik

Bilinear Interpolation

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

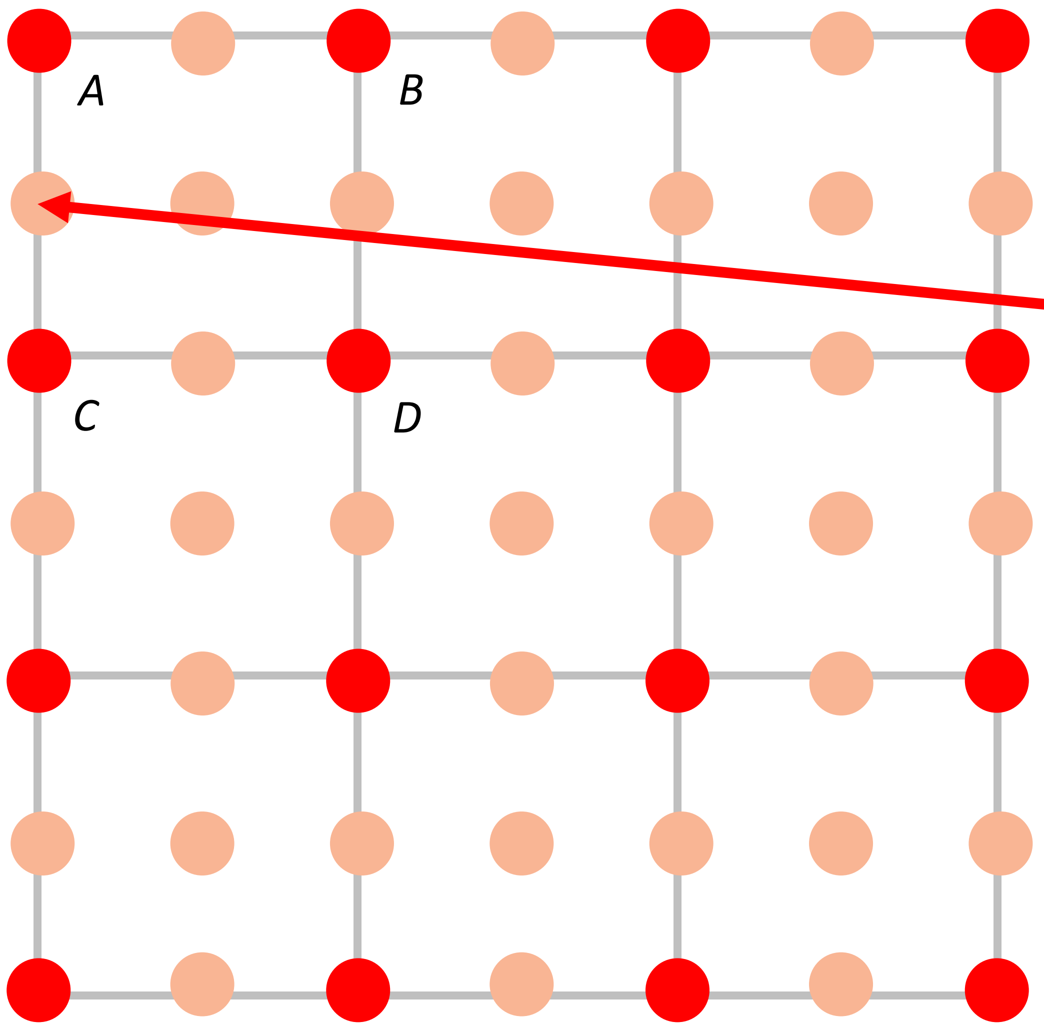


$f(0.5, 0) = \frac{1}{2}(A + B)$

Source: Lazeknik

Bilinear Interpolation

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

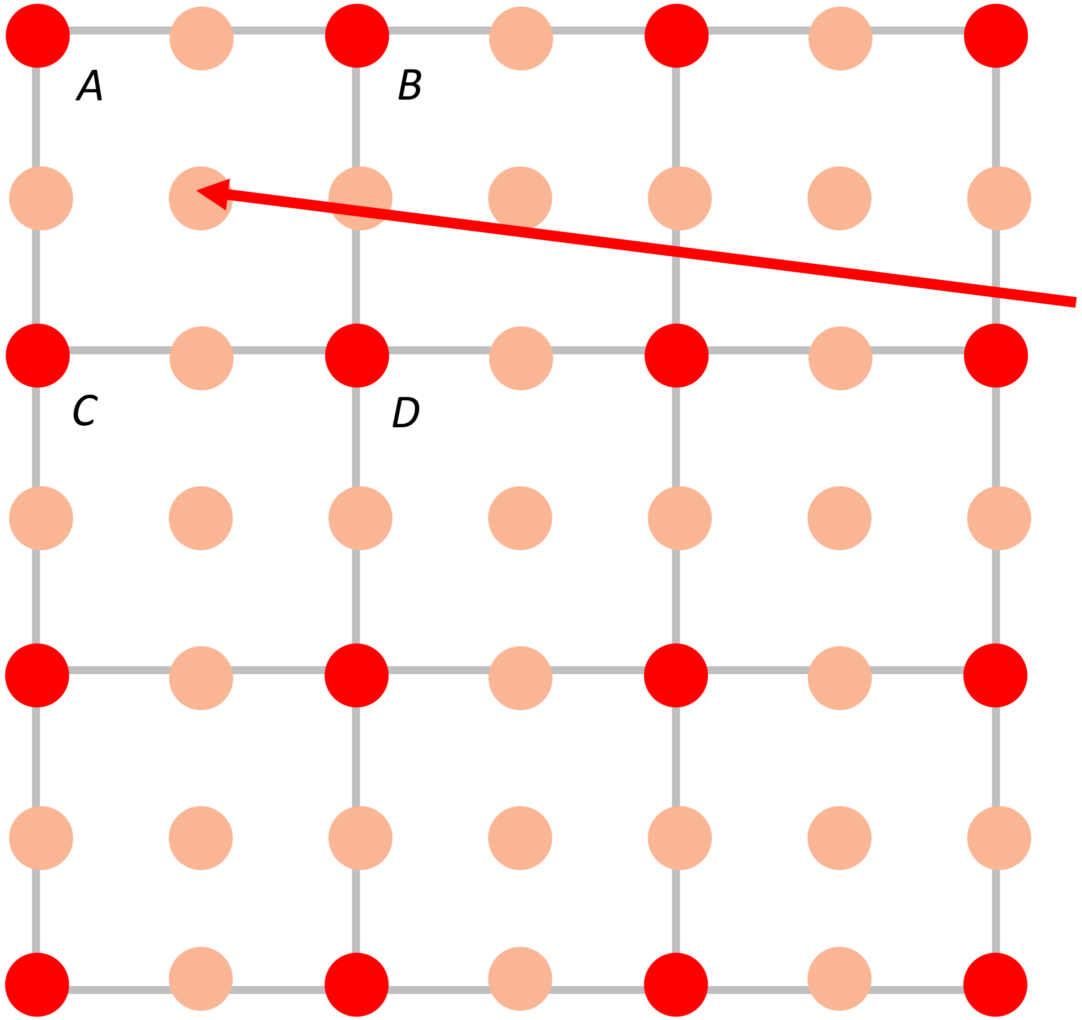


$$f(0, 0.5) = \frac{1}{2}(A + C)$$

Source: Lazeknik

Bilinear Interpolation

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

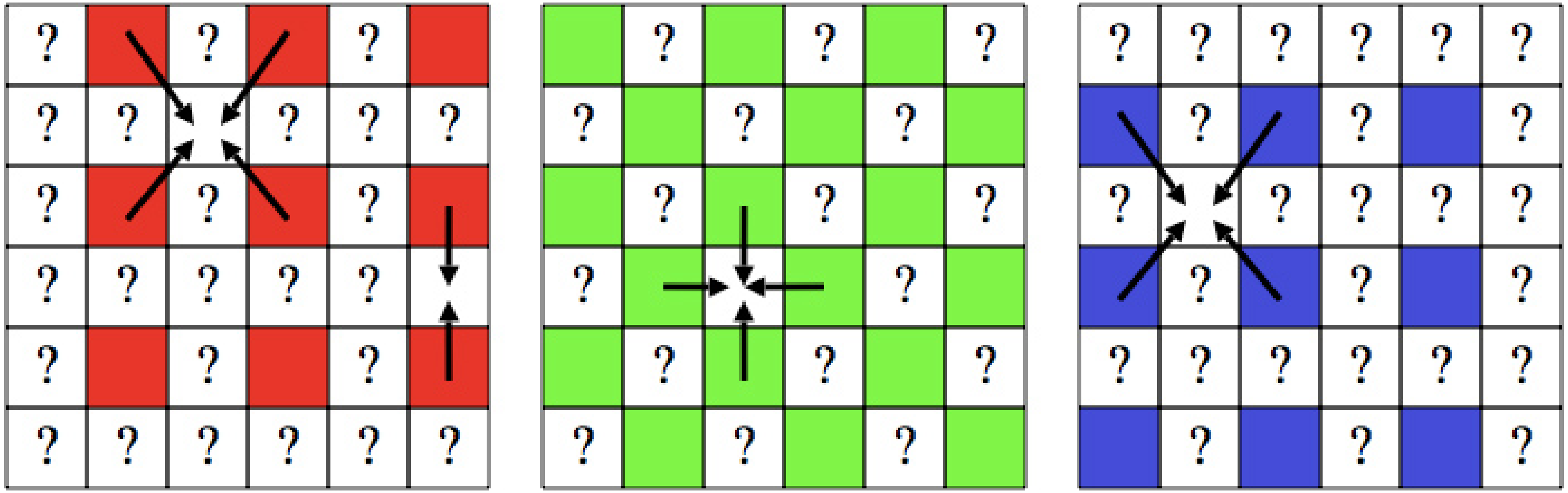


$f(0.5, 0.5) =$

$\frac{1}{4}(A + B + C + D)$

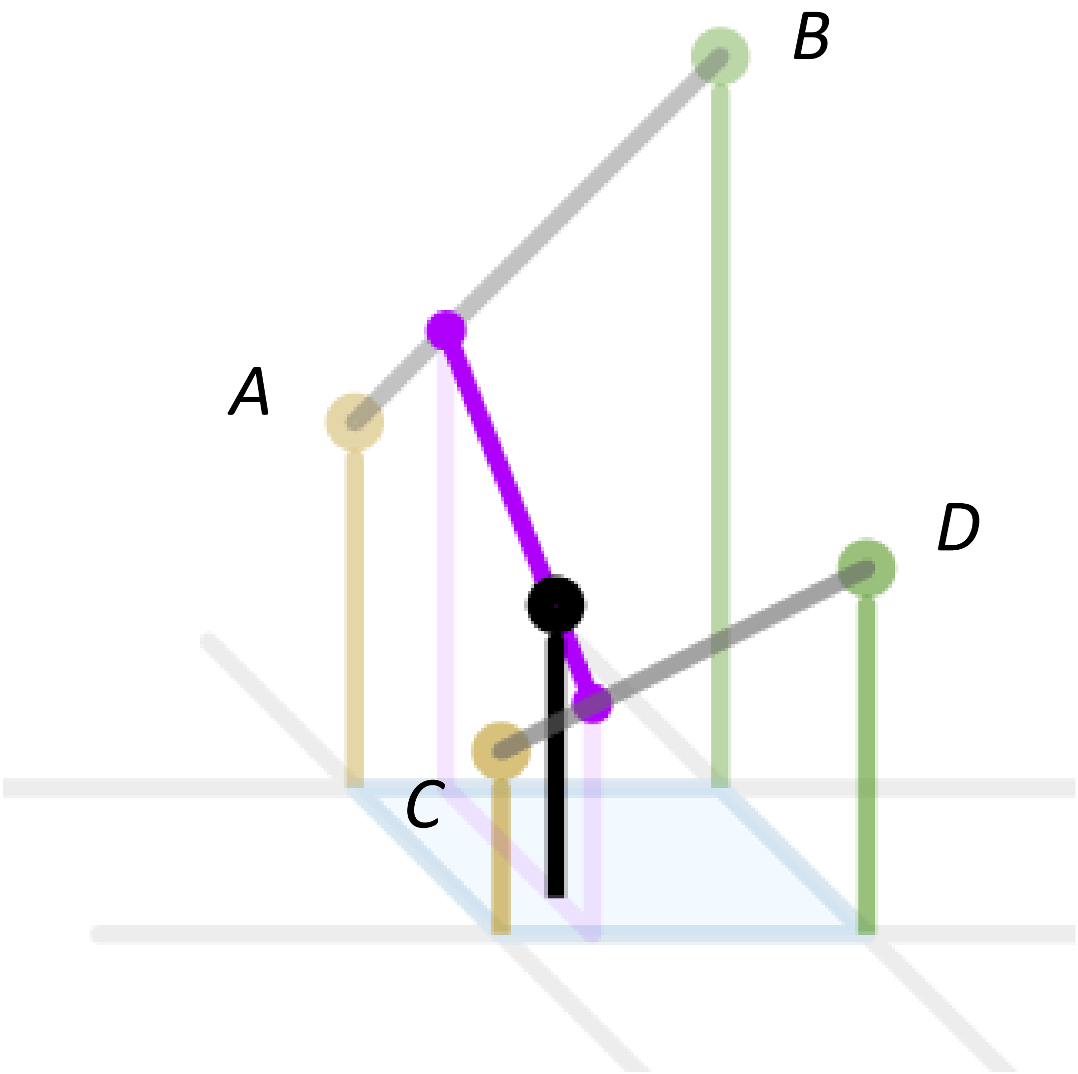
Source: Lazeknik

Application: Demosaicing



Source: Lazeknik

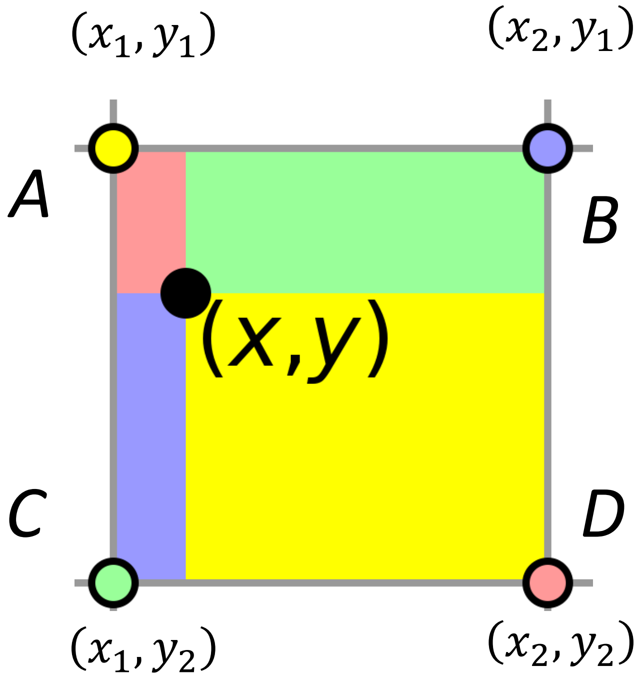
Bilinear Interpolation More Generally

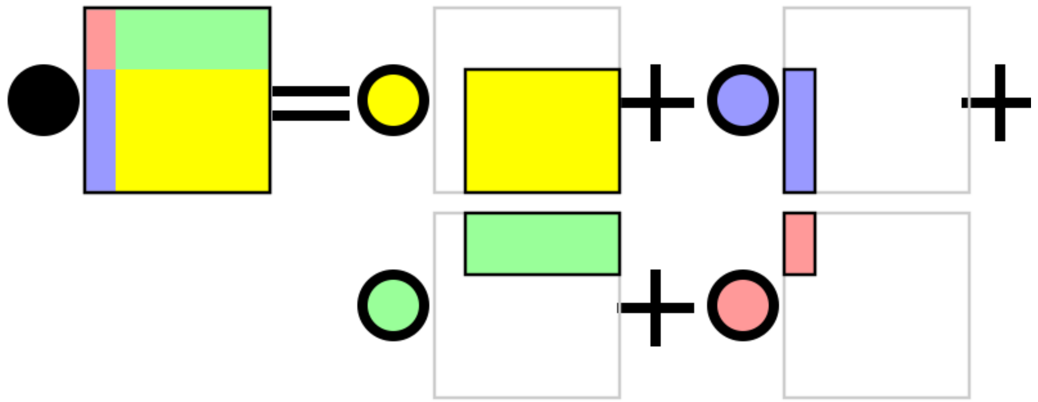


http://en.wikipedia.org/wiki/Bilinear_interpolation

Source: Lazeknik

Bilinear Interpolation More Generally





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

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

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

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

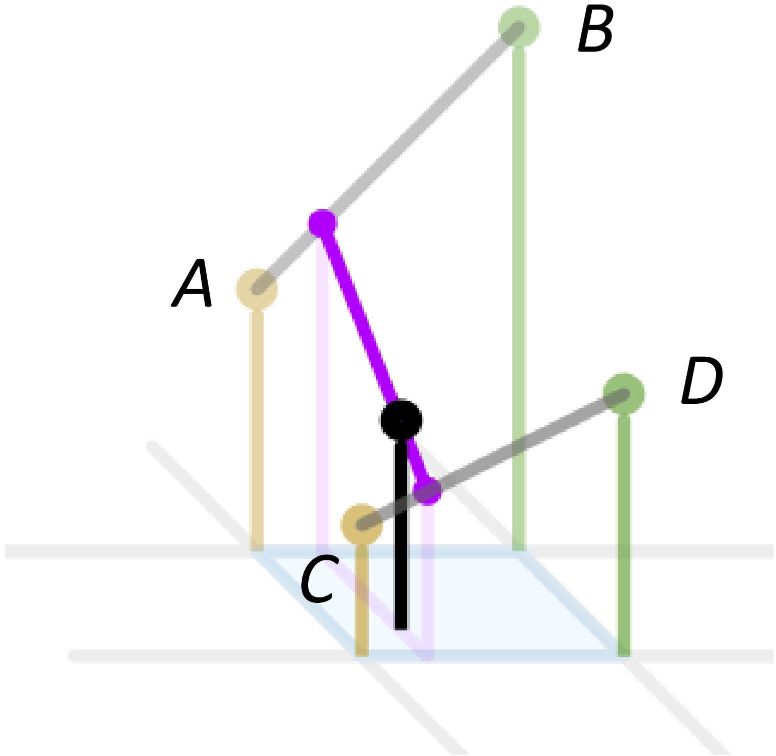
$$w_{22} = \frac{(x - x_1)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)}$$

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

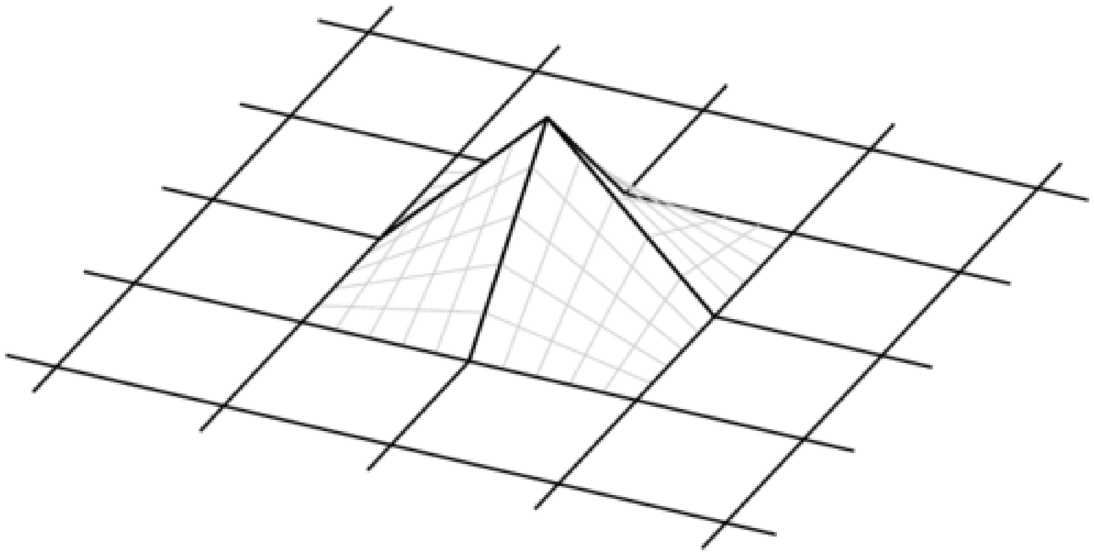
Source: Lazeknik

Bilinear interpolation: Basis function view

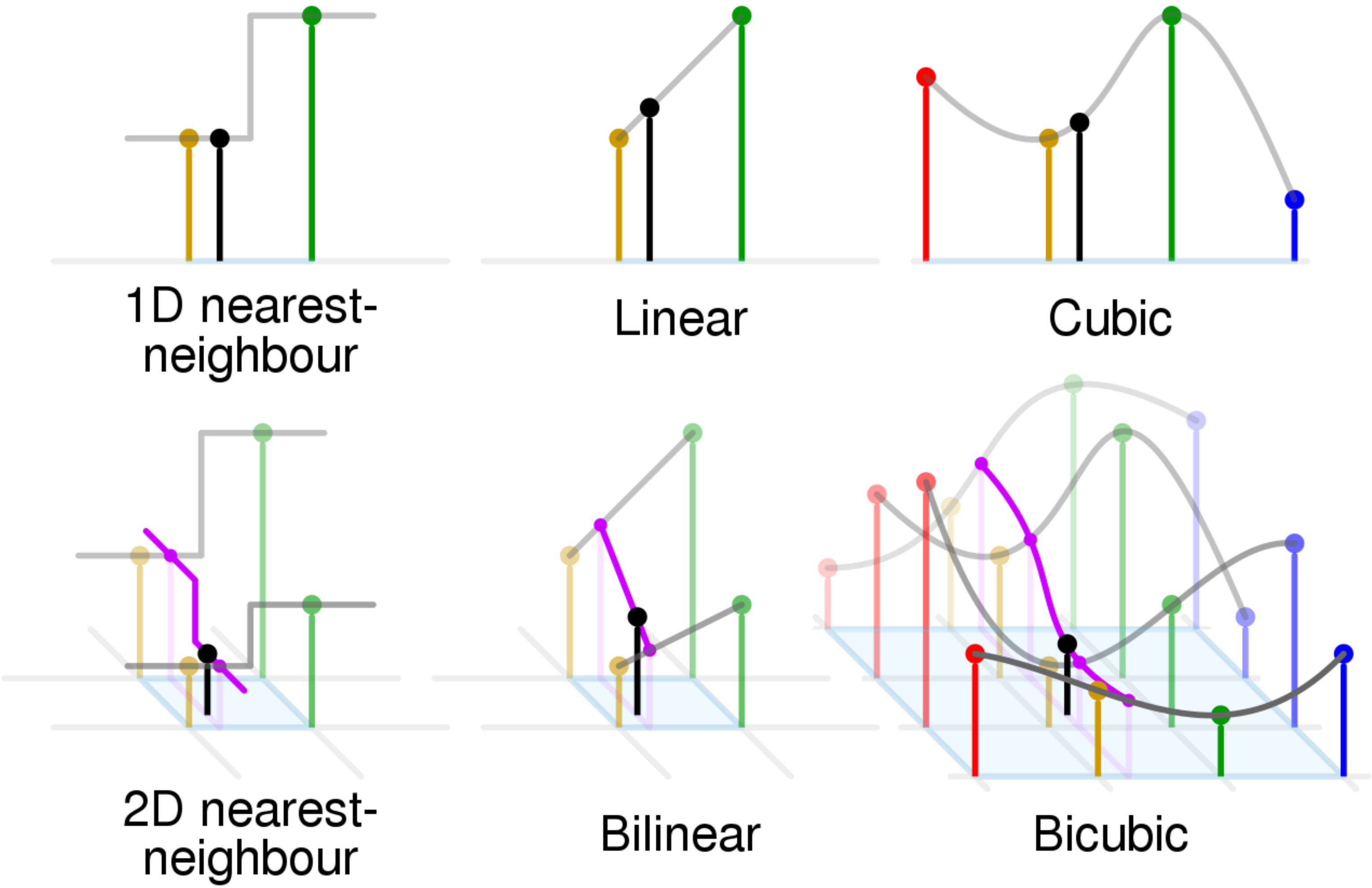
Interpolated function is sum of basis functions or “bumps” centered at the four adjacent grid points, weighted by the image values at the corresponding points



Bilinear basis function



Other kinds of interpolation

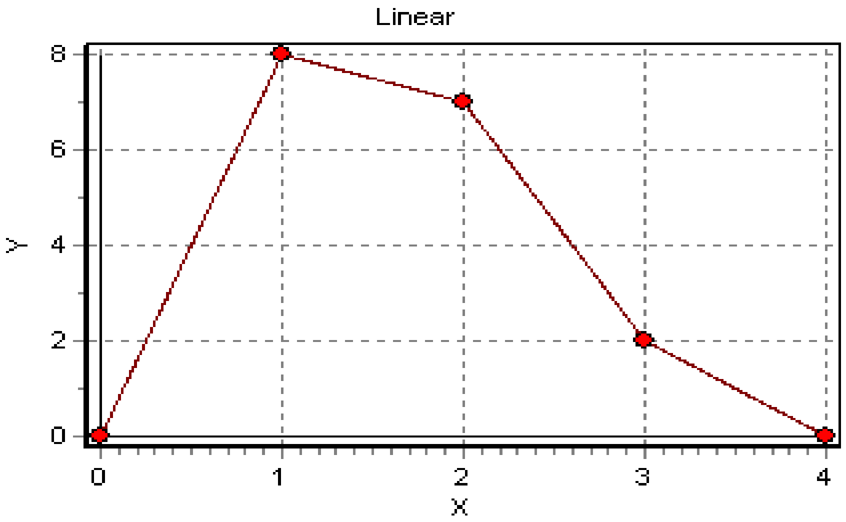


Source: Wikipedia

Source: Lazeknik

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?



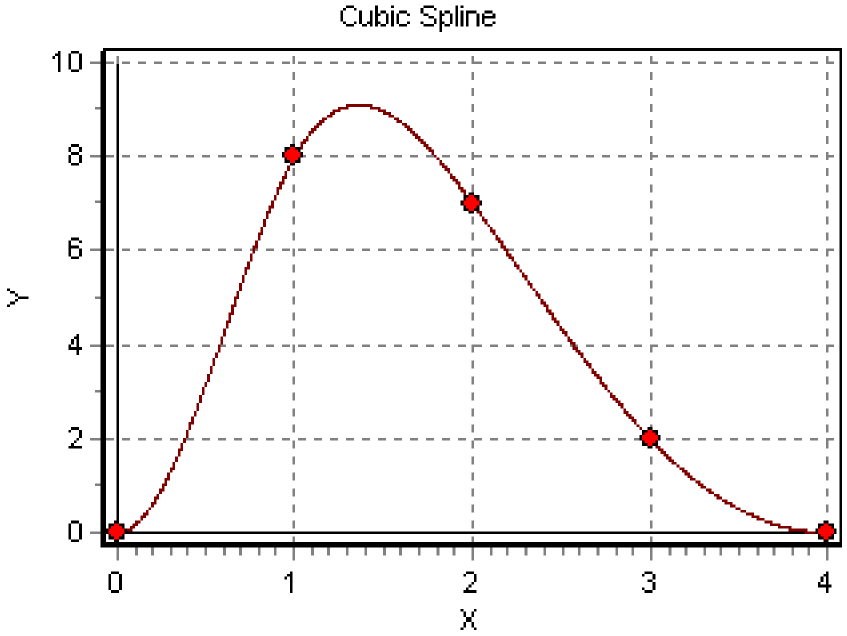


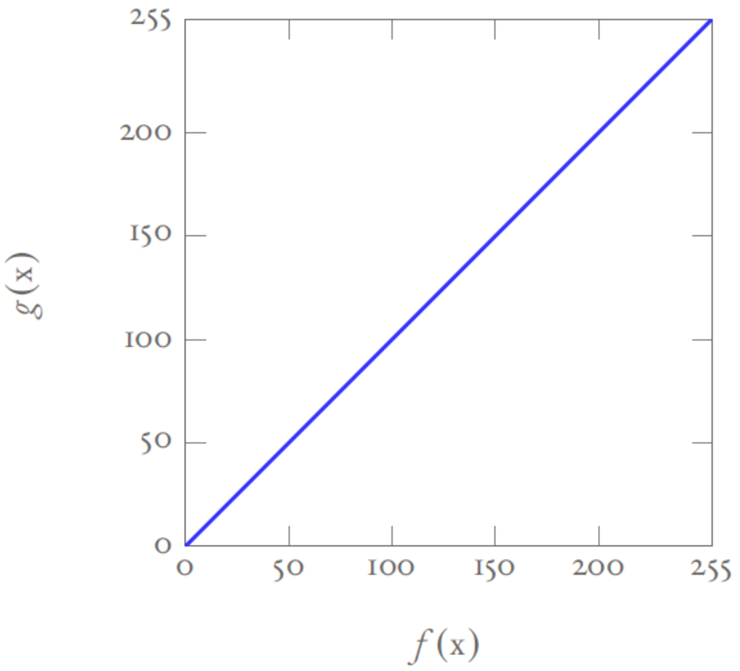
Image source

Intensity Transformations

Intensity Transformations: Introduction

In intensity transformations, the output value of the pixel depends only on the input values of that pixel, not it neighbors.

Input image: $f(\boldsymbol{x})$
Output image: $g(\boldsymbol{x})$
Intensity transform $g(\boldsymbol{x}) = T(f(\boldsymbol{x}))$

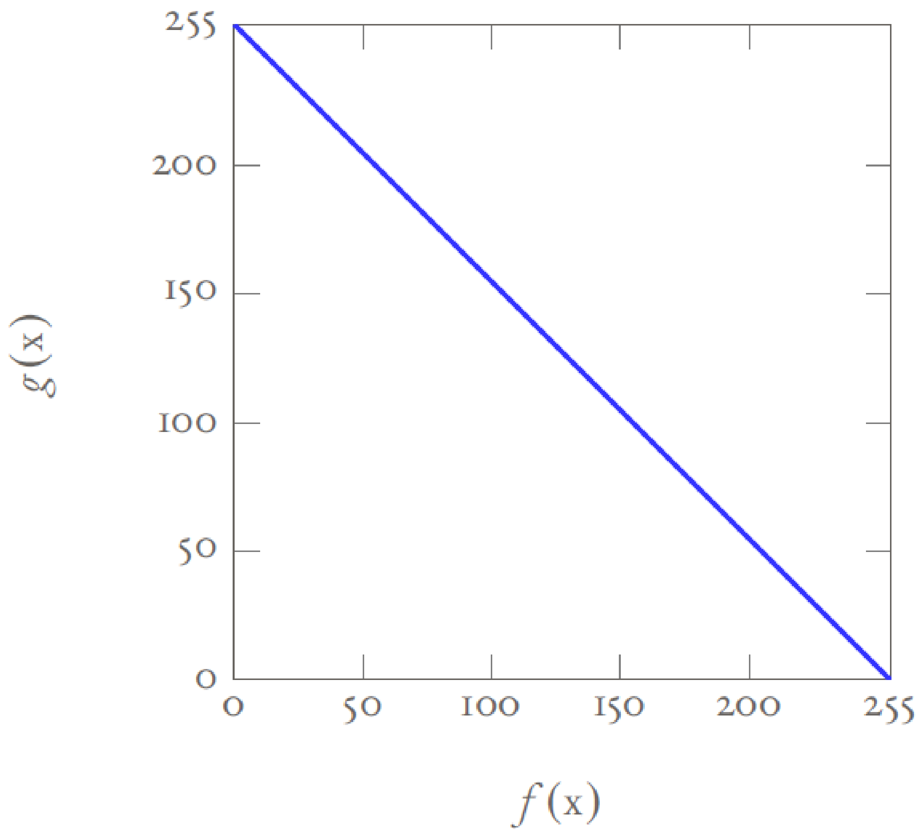


E.g., identity transform $\boldsymbol{T}(.) = \mathbf{1}$

```
import cv2 as cv
import numpy as np
f = cv.imread('images/emma_gray.jpg',
cv.IMREAD_GRAYSCALE)
t = np.arange(256, dtype=np.uint8)
g = t[f]
```

1. Explain the line g = cv.LUT(f, t).
2. This can be done using NumPy only as g= t[f]. Explain this operation.
3. What is $T()$ here?

Example



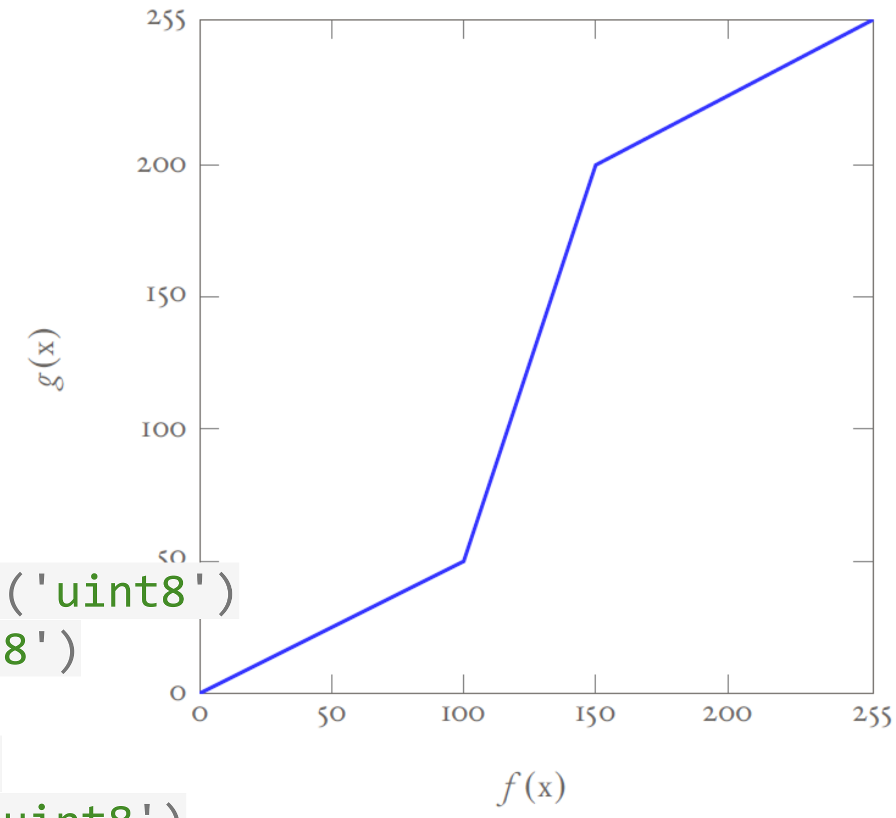
g(x) = 255 – f (x)

Implementation:

```
t = np.arange(255, -1, -1,  dtype=np.uint8)
```

Intensity Windowing

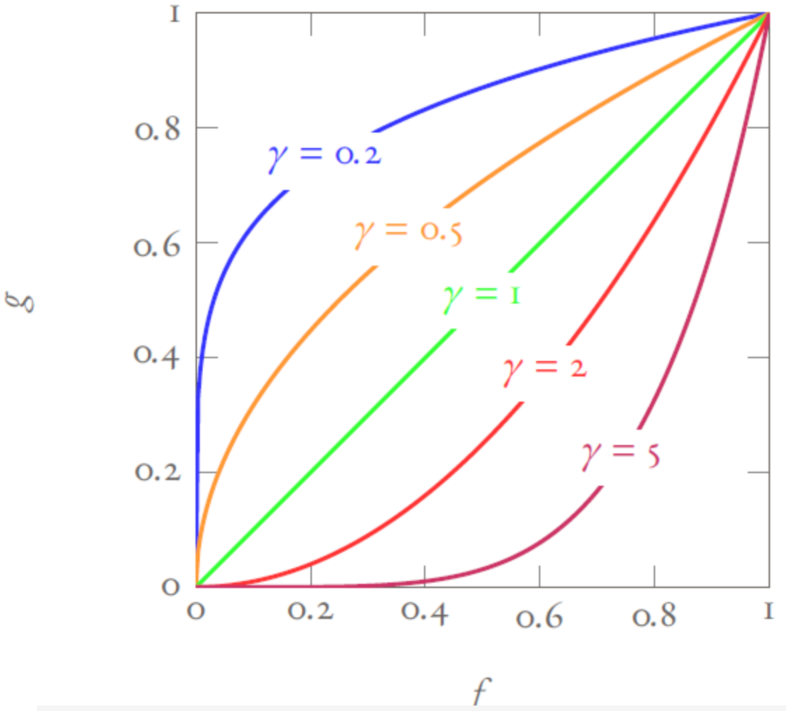
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

c = np.array([(100, 50), (150, 200)])

t1 = np.linspace(0, c[0,1], c[0,0] + 1 - 0).astype('uint8')
t2 = np.linspace(c[0,1] + 1, c[1,1], c[1,0] - c[0,0]).astype('uint8')
t3 = np.linspace(c[1,1] + 1, 255, 255 - c[1,0]).astype('uint8')

transform = np.concatenate((t1, t2), axis=0).astype('uint8')
transform = np.concatenate((transform, t3), axis=0).astype('uint8')
print(len(transform))

img_orig = cv.imread('images/katrina.jpg', cv.IMREAD_GRAYSCALE)
image_transformed = cv.LUT(img_orig, transform)
```



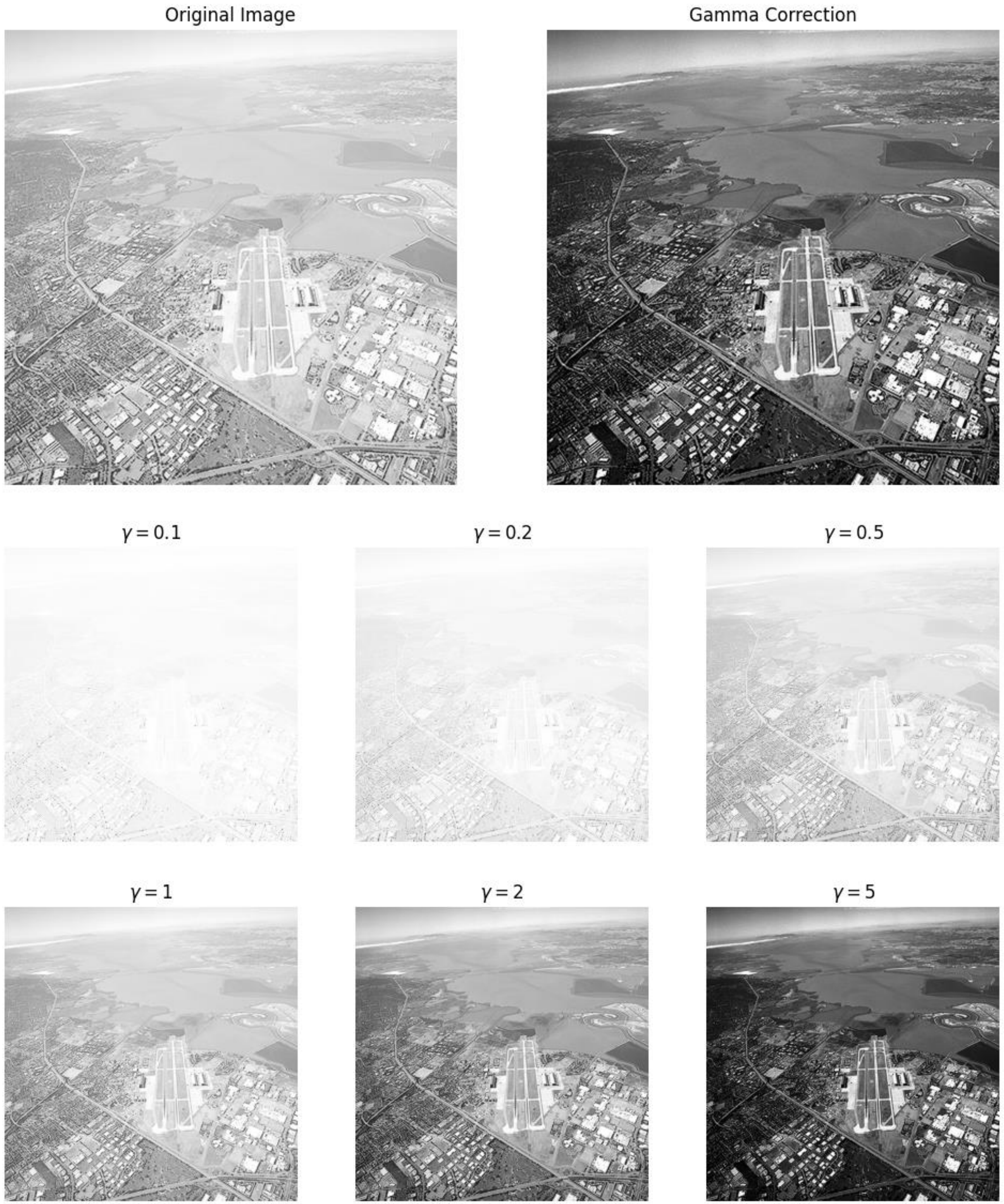
Gamma Correction

$$g = f^{\gamma}, \qquad f \in [0,1]$$

Values of γ such that $0 < \gamma < 1$ map a narrow range of dark pixels to a wider range of dark pixels.
$\gamma > 0$ has the opposite effect.
$\gamma = 1$ gives the identity transform.




```python
t = np.array([(i/255.0)**(gamma)*255 for i in
np.arange(0,256)]).astype(np.uint8)
g = cv.LUT(f, t)
```



Histograms

1. We can represent the intensity distribution over the range of intensities [0, 255], using a histogram.
2. If h is the histogram of a particular image, $h(r_k)$ gives us how many pixels have the intensity r_k.
3. The histogram of a digital image with gray values in the range $[0, L - 1]$ is a discrete function $h(r_k) = n_k$ where r_k is the kth gray level and n_k is the number of pixels having gray level r_k.
4. We can normalize the histogram by dividing by the total number of pixels n. Then we have an estimate of the probability of occurrence of level r_k , i.e., $p(r_k) = n_k/n$.
5. The histogram that we described above has L bins. We can construct a coarser histogram by selecting a smaller number of bins than L. Then several adjacent values of k will be counted for a bin.

Exercise

The figure shows a 3×4 image. The range of intensities that this image has is $[0, 7]$. Compute its histogram.

6	5	5	3
7	6	6	4
2	3	5	4

Image Properties though the Histogram

1. If the image is dark histogram will have many values in the left region, that correspond to dark pixels.
2. If the image is bright histogram will have many values in the right region, that correspond to bright pixels.
3. If a significant number of pixels are dark and a significant number of pixels a bright, the histogram will have two modes, one in the left region and the other in the right region.
4. A flat histogram signifies that the image has a uniform distribution of all intensities. Then, the contrast is high, and image will look vibrant.

Histogram Equalization

1. Photographers like to shoot pictures with a flat histogram, as such pictures are vibrant.
2. Histogram equalization is a gray-level transformation that results in an image with a more or less flat histogram.
3. We can take and image and make its histogram flat by using the operation called histogram equalization.

$$s = T(r) = (L-1)\int_0^r p_r(w)dw$$

$$s = T(r) = \frac{L-1}{MN}\sum_{j=0}^{k} n_j\,, \qquad k = 0,..,L-1$$

Example:
Suppose that a 3-bit image (L = 8) of size 64×64 pixels ($MN = 4096$) has the intensity distribution shown in Table 4, where the intensity levels are in the range $[0, L-1] = [0,7]$. Carry out histogram equalization.

r_k	n_k	$p_r(r_k) = n_k/MN$	$\sum_{j=0}^{k} n_j$	$\frac{(L-1)}{MN}\sum_{j=0}^{k} n_j$	Rounded
$r_0 = 0$	790	0.19			
$r_1 = 1$	1023	0.25			
$r_2 = 2$	850	0.21			
$r_3 = 3$	656	0.16			
$r_4 = 4$	329	0.08			
$r_5 = 5$	245	0.06			
$r_6 = 6$	122	0.03			
$r_7 = 7$	81	0.02			

r_k	n_k	$p_r(r_k) = n_k/MN$	$\sum_{j=0}^{k} n_j$	$\frac{(L-1)}{MN} \sum_{j=0}^{k} n_j$	Rounded
$r_0 = 0$	790	0.19	790		
$r_1 = 1$	1023	0.25			
$r_2 = 2$	850	0.21			
$r_3 = 3$	656	0.16			
$r_4 = 4$	329	0.08			
$r_5 = 5$	245	0.06			
$r_6 = 6$	122	0.03			
$r_7 = 7$	81	0.02			

r_k	n_k	$p_r(r_k) = n_k/MN$	$\sum_{j=0}^{k} n_j$	$\frac{(L-1)}{MN} \sum_{j=0}^{k} n_j$	Rounded
$r_0 = 0$	790	0.19	790		
$r_1 = 1$	1023	0.25	1813		
$r_2 = 2$	850	0.21	2663		
$r_3 = 3$	656	0.16	3319		
$r_4 = 4$	329	0.08	3648		
$r_5 = 5$	245	0.06	3893		
$r_6 = 6$	122	0.03	4015		
$r_7 = 7$	81	0.02	4096		

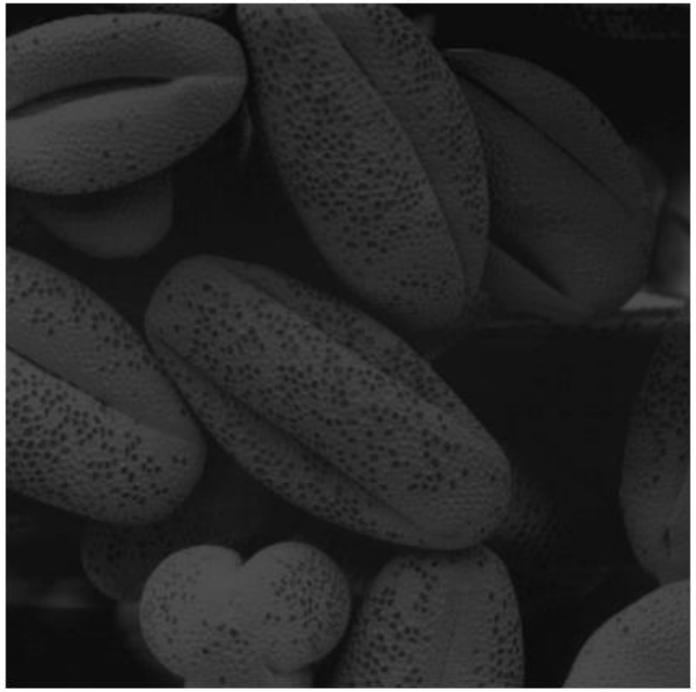
r_k	n_k	$p_r(r_k) = n_k/MN$	$\sum_{j=0}^{k} n_j$	$\frac{(L-1)}{MN}\sum_{j=0}^{k} n_j$	Rounded
$r_0 = 0$	790	0.19	790	1.350	1
$r_1 = 1$	1023	0.25	1813	3.098	3
$r_2 = 2$	850	0.21	2663	4.551	5
$r_3 = 3$	656	0.16	3319	5.672	6
$r_4 = 4$	329	0.08	3648	6.234	6
$r_5 = 5$	245	0.06	3893	6.653	7
$r_6 = 6$	122	0.03	4015	6.862	7
$r_7 = 7$	81	0.02	4096	7.000	7

Do the quiz.
Meeting at 2:30

Histogram Equalization Using OpenCV

```python
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

f = cv.imread('images/shells.tif', cv.IMREAD_GRAYSCALE)
g = cv.equalizeHist(f)
```

Original Image



Histogram Equalization

Histogram Equalization Using the Formula

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

f = cv.imread('images/shells.tif', cv.IMREAD_GRAYSCALE)

t = np.array([(L-1)/(M*N)*cdf[k] for k in range(256)], 
dtype=np.uint8)
g = t[f]

g = cv.equalizeHist(f)
fig, ax = plt.subplots(1, 2, figsize=(12, 8))
ax[0].imshow(f, cmap='gray', vmin=0, vmax=255)
ax[0].set_title('Original Image')
ax[0].axis('off')
ax[1].imshow(g, cmap='gray', vmin=0, vmax=255)
ax[1].set_title('Histogram Equalization')
ax[1].axis('off')
plt.show()
```

Summary

1. Basics: digital image, color images, creating an image in Python, displaying images, increasing brightness, image planes
2. Intensity transformations:
$$g = t[f]$$
    1. Identity, negative, intensity windowing
    2. Gamma correction
    3. Histograms, histogram equalization