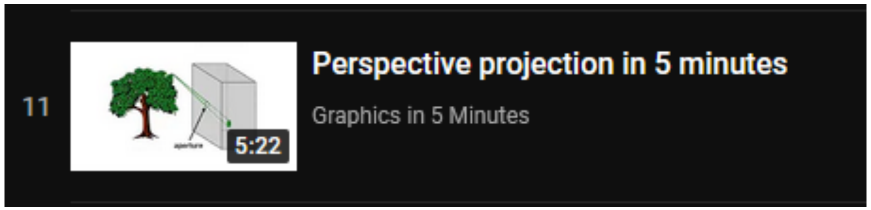
Image basics

What is an image?

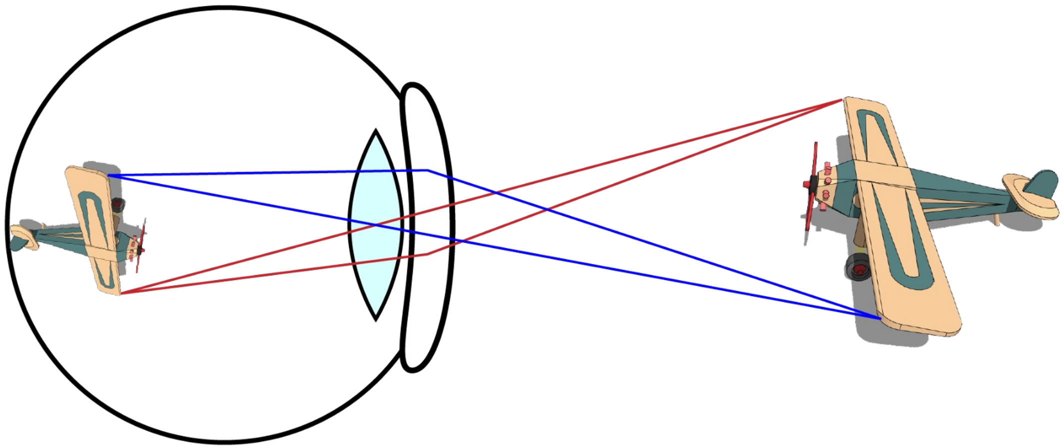
- slides copied and adjusted from: https://pjreddie.com/courses/computer-vision/

Graphics in 5 minutes

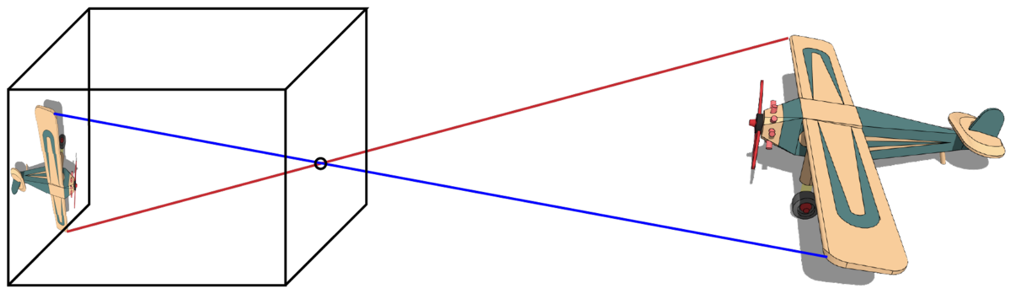
https://www.youtube.com/watch?v=F5WA26W4JaM



Eyes: projection onto retina



Model: pinhole camera



Model: pinhole camera

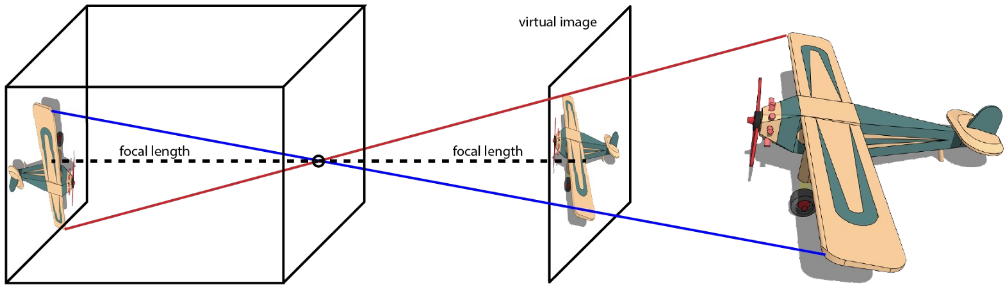


Image: 3D → 2D projection of the world

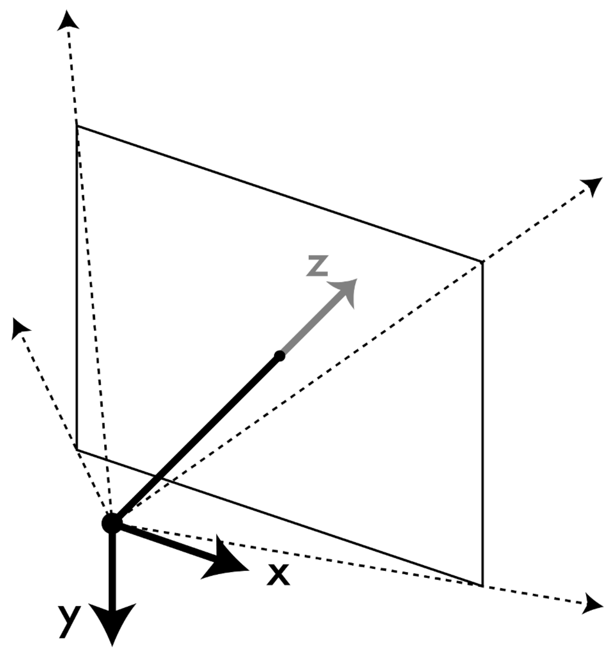


Image: 3D → 2D projection of the world

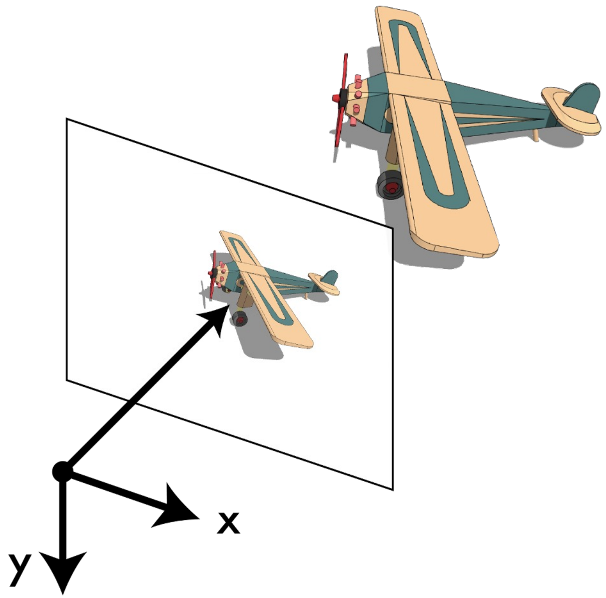


Image: 3D → 2D projection of the world

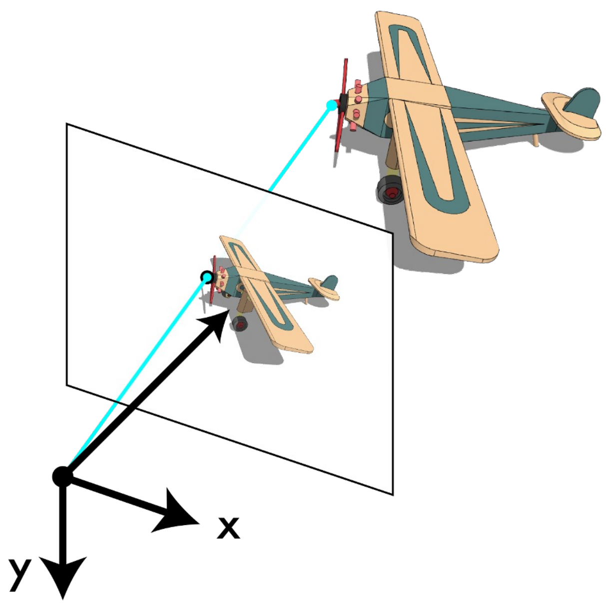


Image: 3D → 2D projection of the world

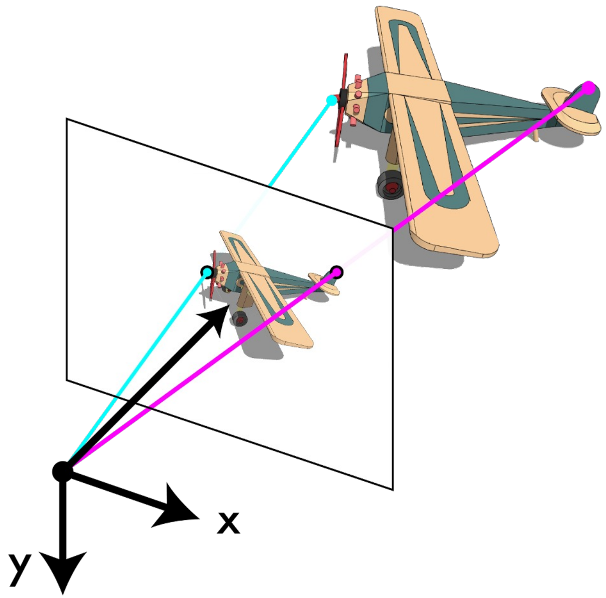


Image: 3D → 2D projection of the world

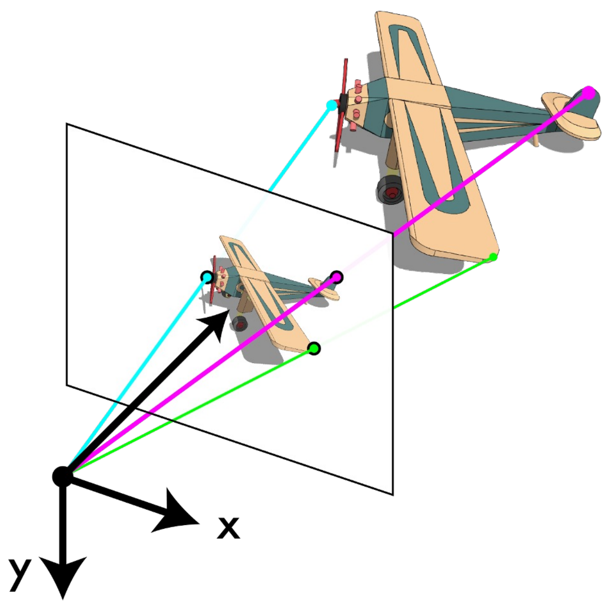
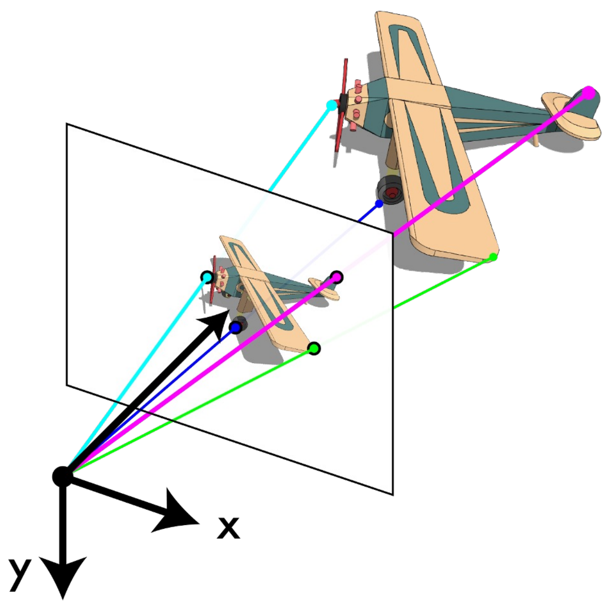
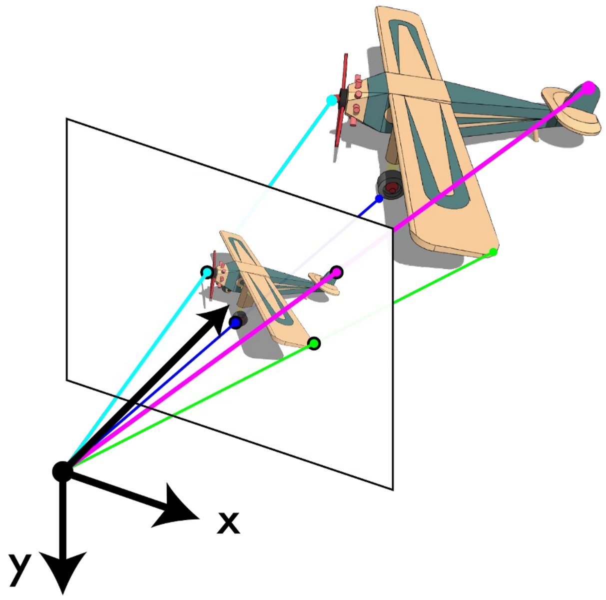


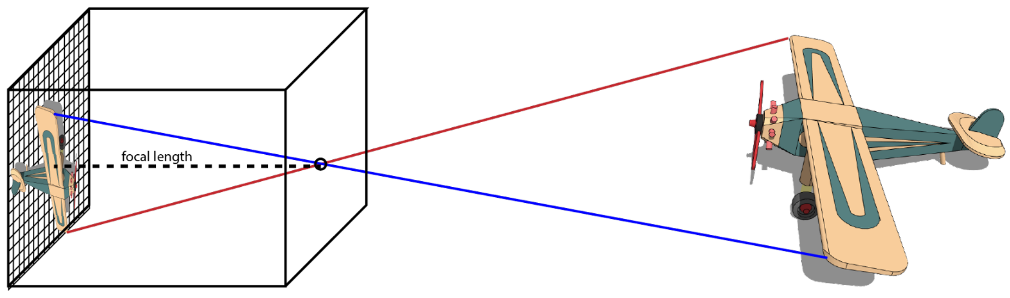
Image: 3D → 2D projection of the world



At each point we record incident light



At each point we record incident light

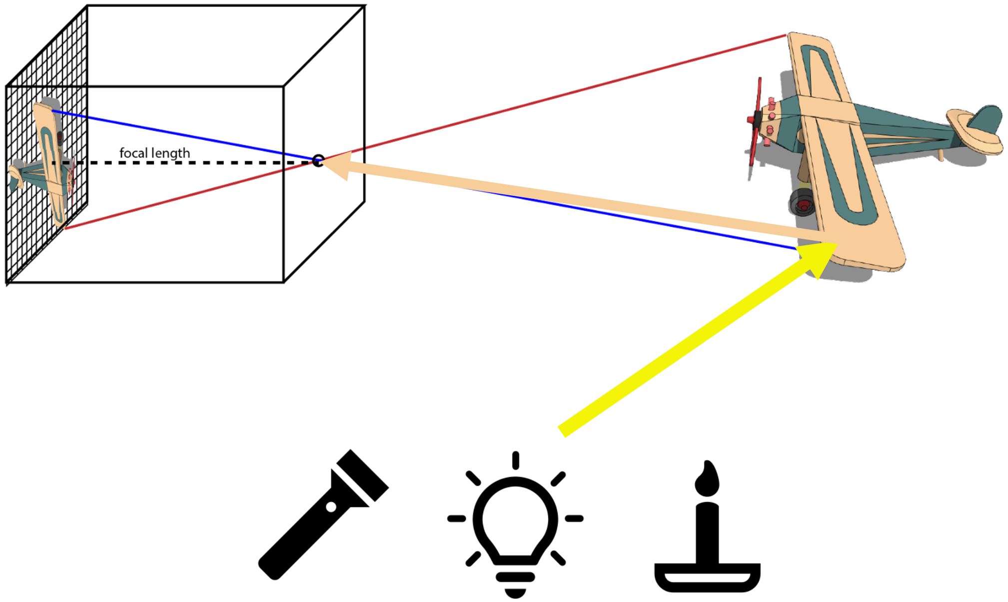


Which parameters influence the image?

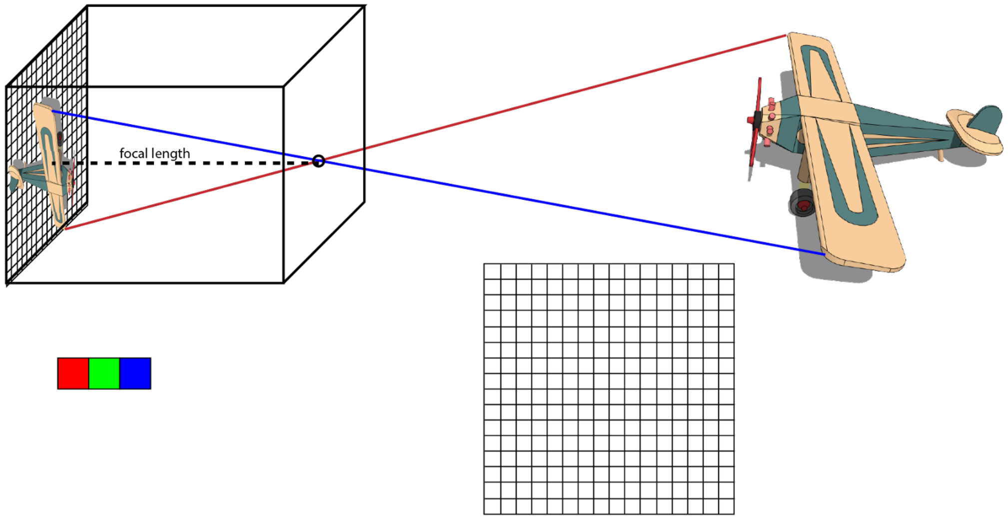
Light sources + material of scene objects and their reflection properties.

Geometry of scene

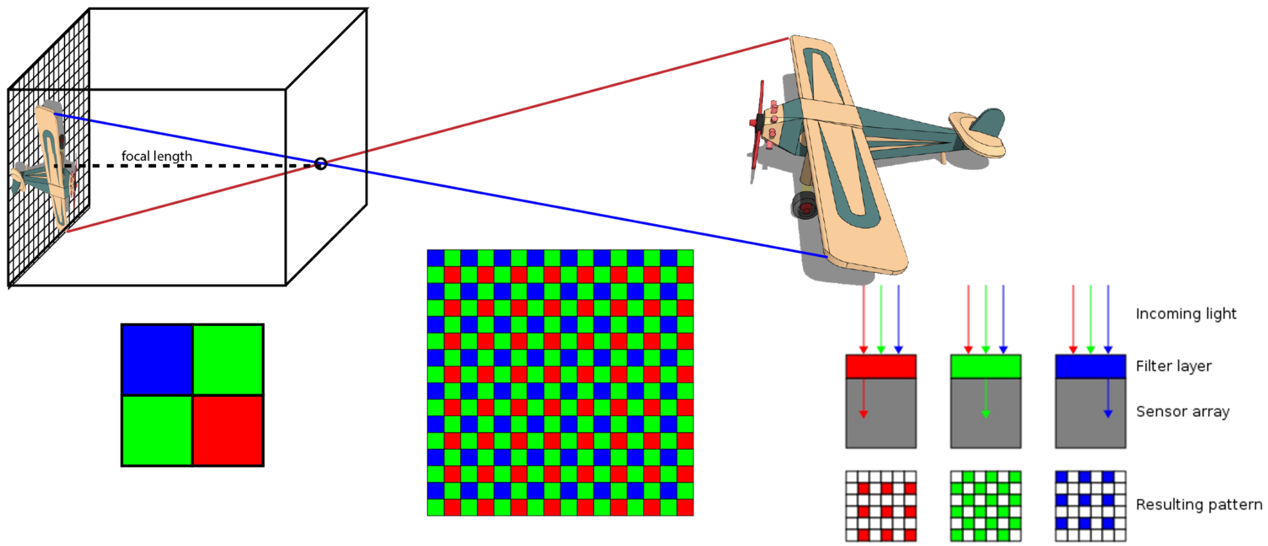
Camera settings/hardware



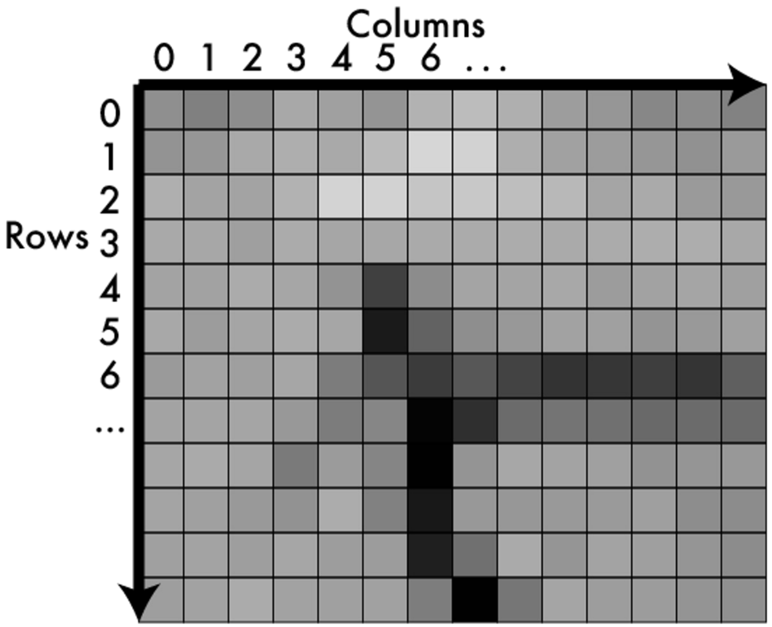
How do we record color?



Bayer pattern for CMOS sensors

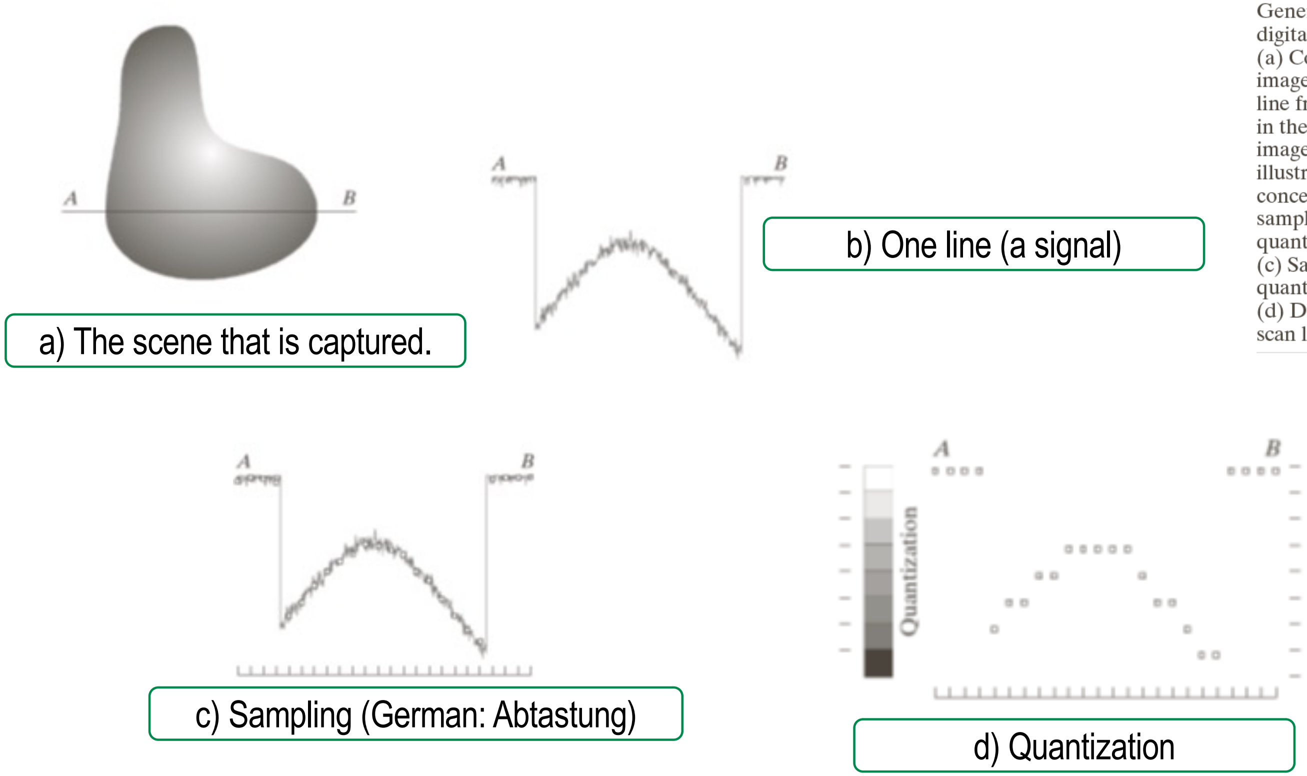


An image is a matrix of light



Digitalization = Sampling + Quantization

- **Sampling**: digitizing coordinate values
- **Quantization**: digitizing amplitude values



a b
c d

FIGURE 2.16 Generating a digital image. (a) Continuous image. (b) A scan line from *A* to *B* in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line.

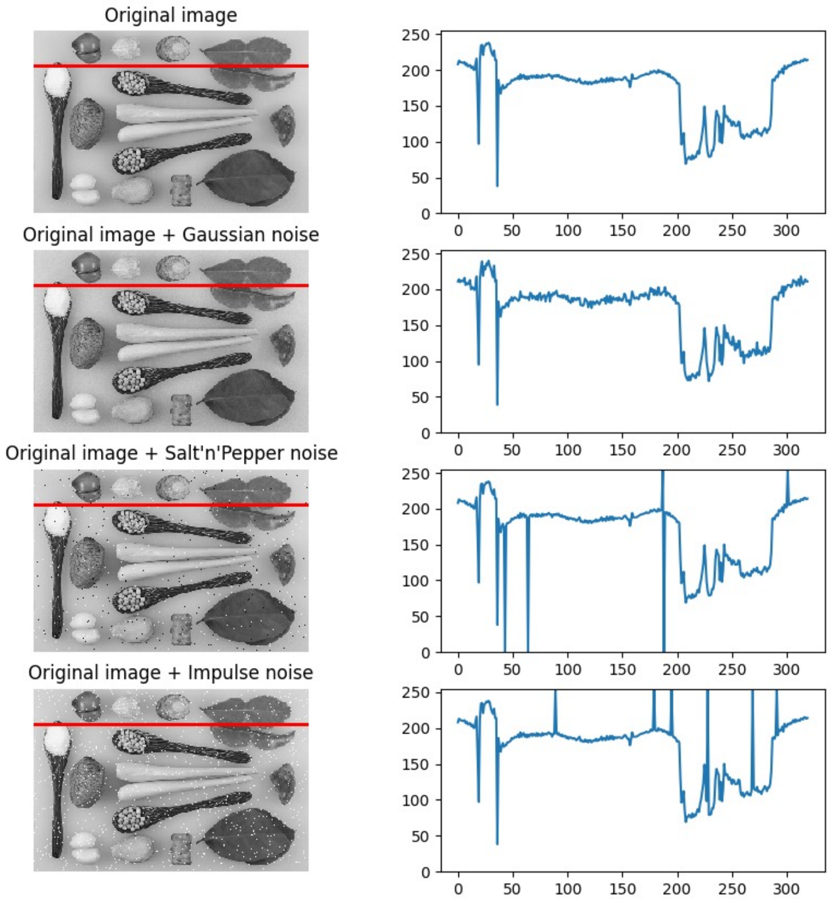
Image noise

Quantization error: 8 bit grayscale is usually enough, but when compression is needed, this error becomes visible.

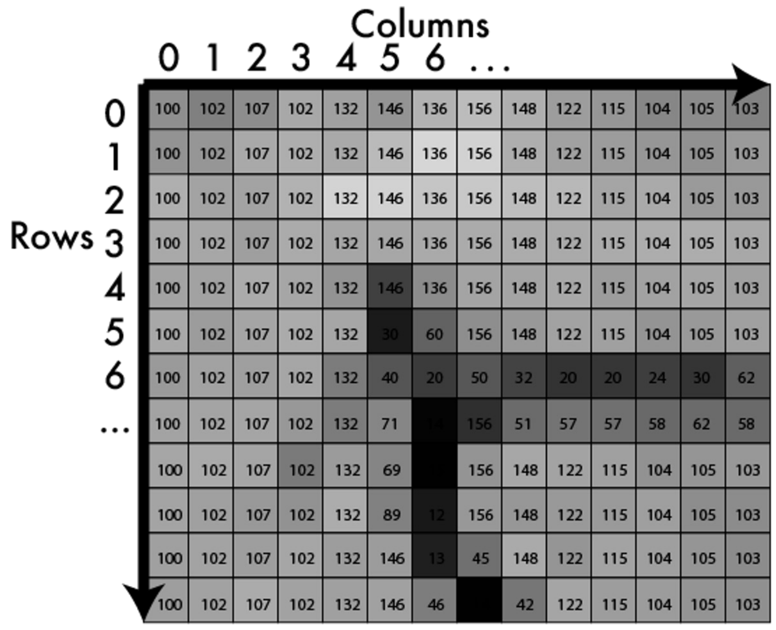
Gaussian noise: Intensity fluctuations originating from a Gaussian normal distribution

Impulse noise: Random occurrence of black and white pixels (aka salt and pepper noise)

See Wikipedia for details.

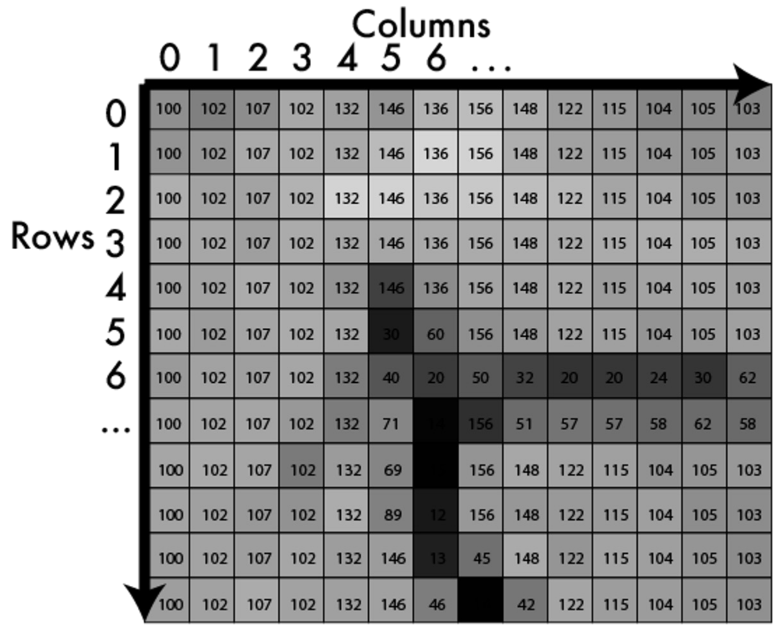


Values in matrix = how much light



Values in matrix = how much light

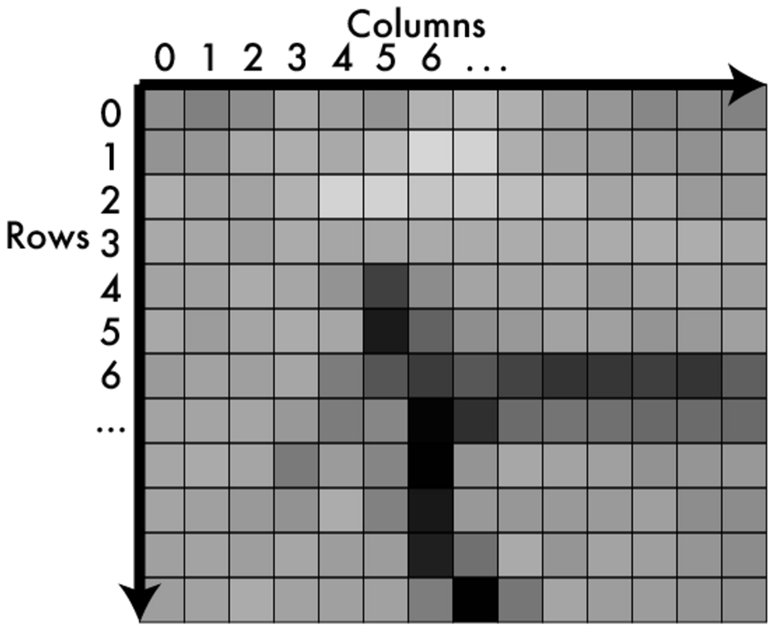
- Higher = more light
- Lower = less light
- Bounded
- No light = 0
- Sensor/device limit = max
- Typical ranges:
- [0-255] fits into byte
- [0-1] as floating point
- Called “pixels”



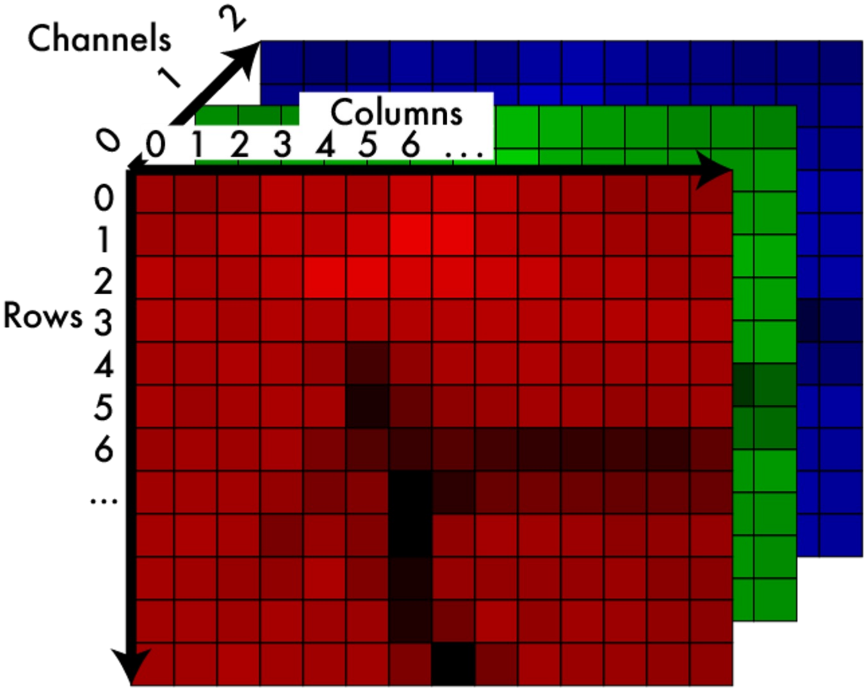
Addressing pixels

Ways to index:

- (x,y)
- Like cartesian coordinates
- (3,6) is column 3 row 6
- (r,c) or (i,j)
- Like matrix notation
- (3,6) is row 3 column 6
- For your future:
- Arbitrary
- Only thing that matters is consistency

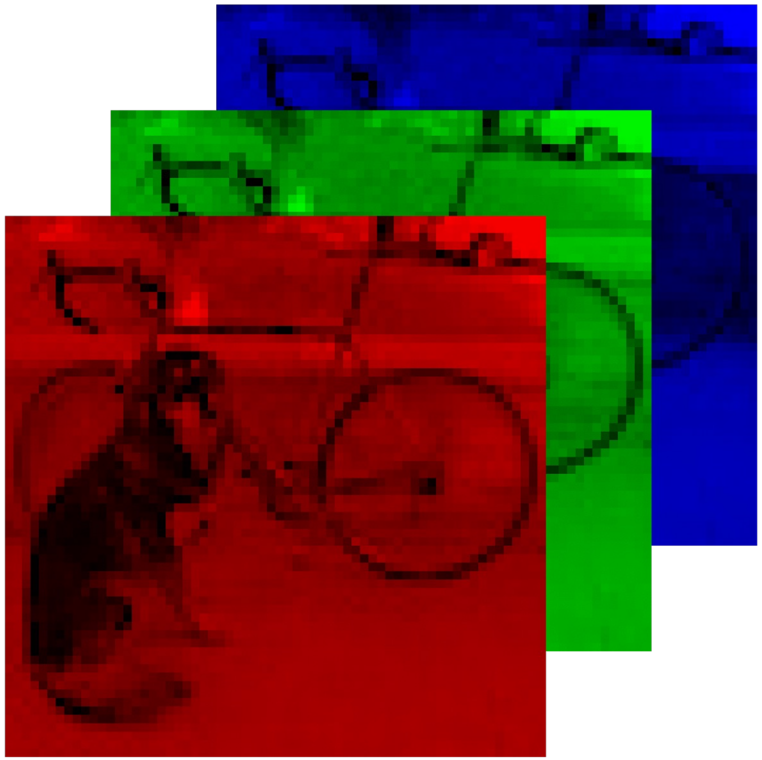


Color image: 3D tensor in colorspace



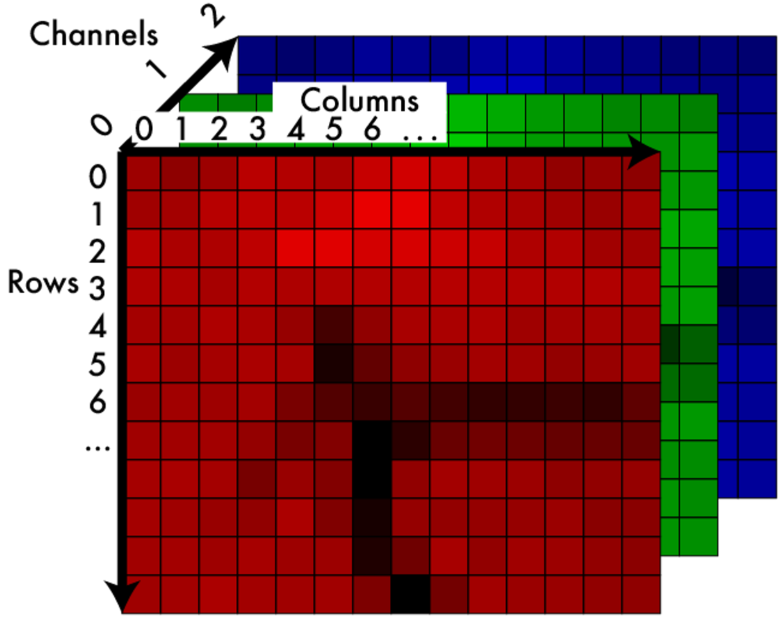
RGB information in separate “channels”

- Remember: we can match “real” colors using a mix of primaries.
- Each channel encodes one primary. Adding the light produced from each primary mimics the original color.

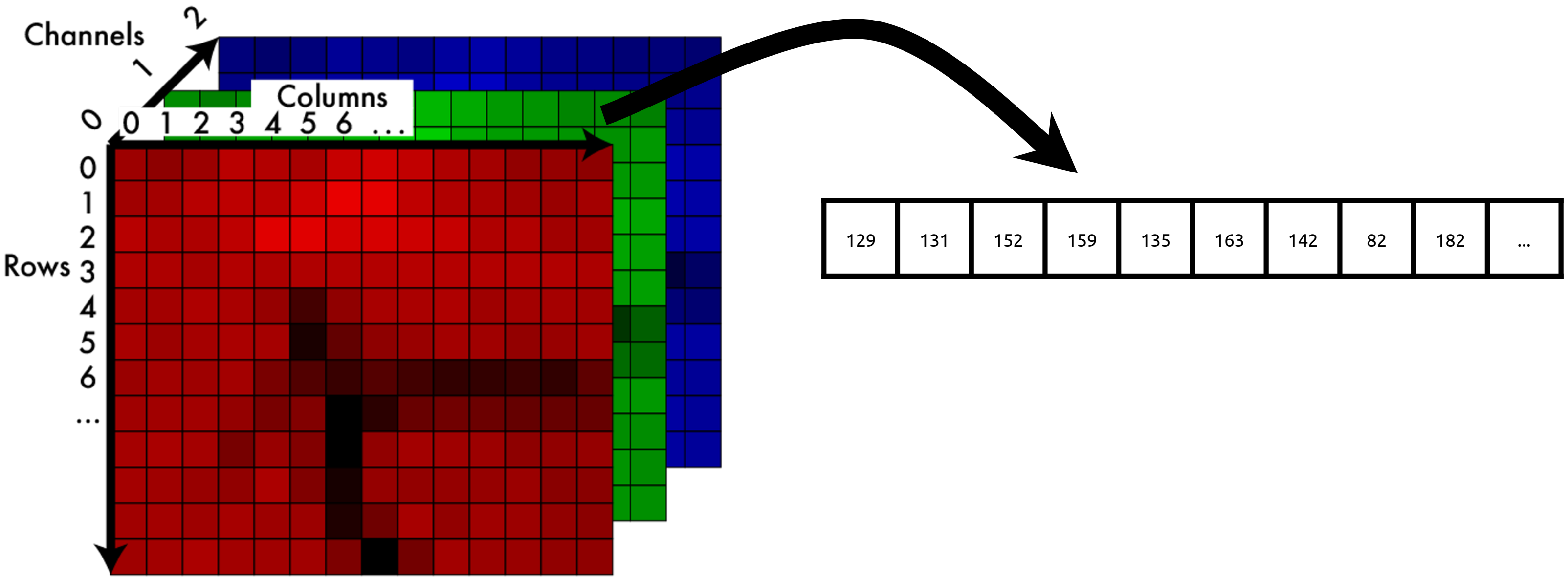


Addressing pixels

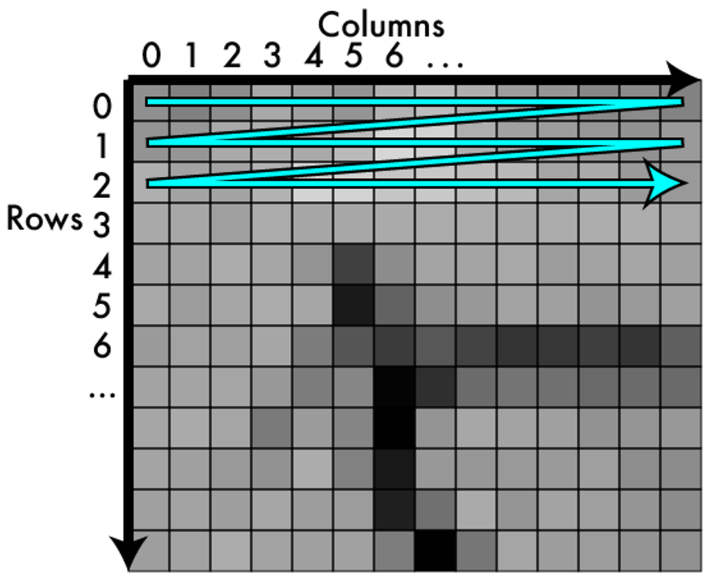
- If you use (x,y,c)
  - (1,2,0) means column 1, row 2, channel 0
- Still doesn't matter, just be consistent
- But know what you do for homeworks :-)
- Also for size:
- 1920 x 1080 x 3 image:
  - 1920 px wide (width)
  - 1080 px tall (height)
  - 3 channels

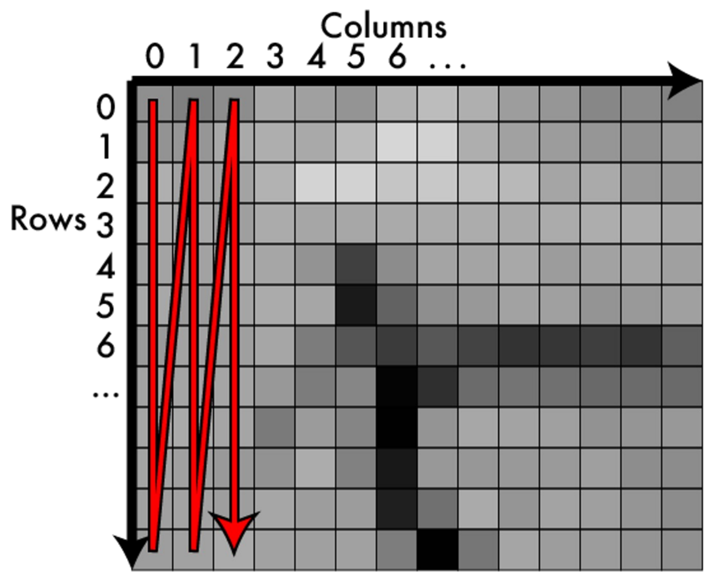


How do we store them?



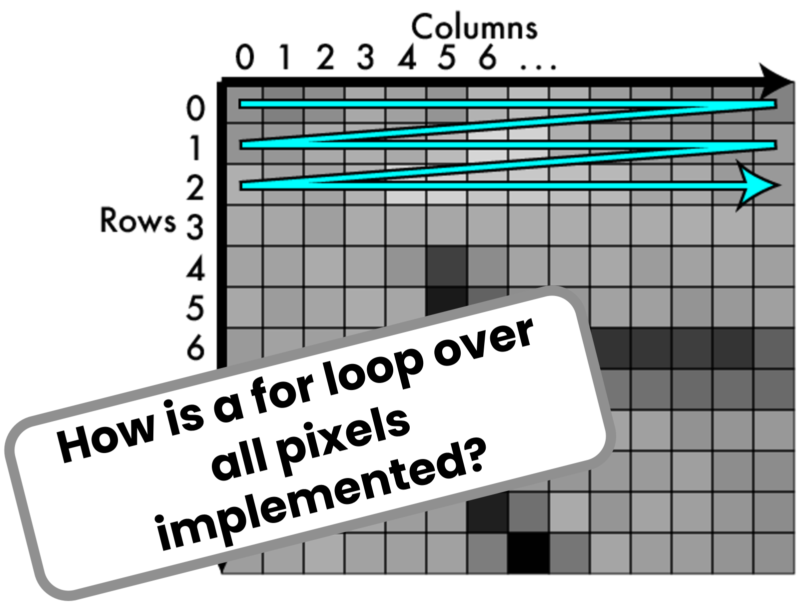
Storage: row major vs column major

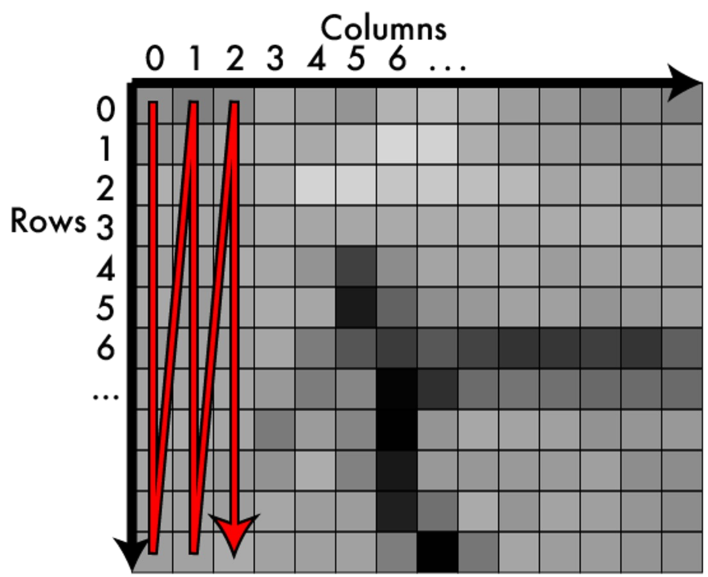




For loop quiz

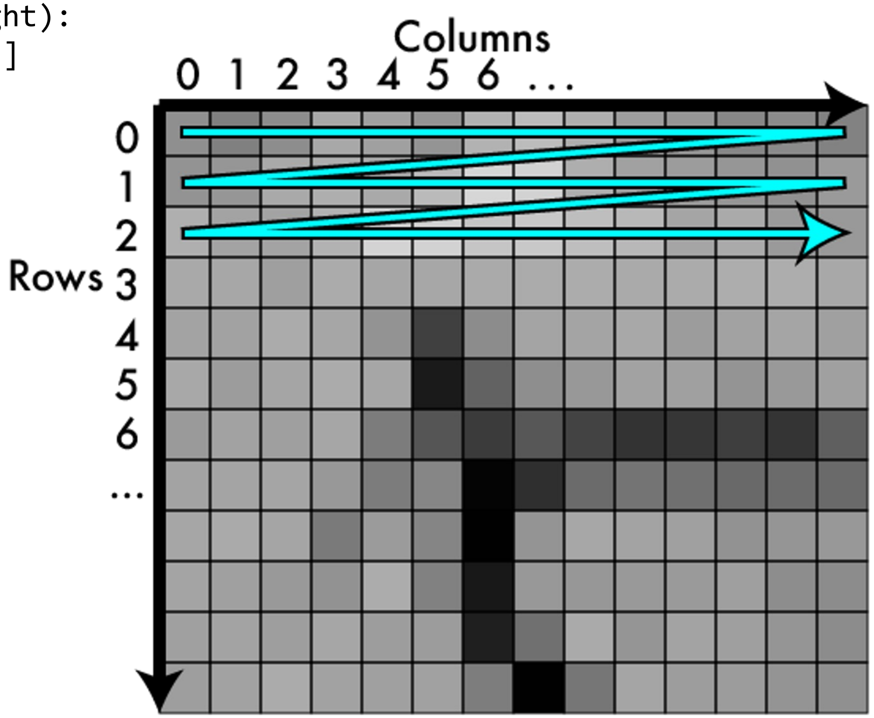
NumPy creates arrays in row-major order by default.



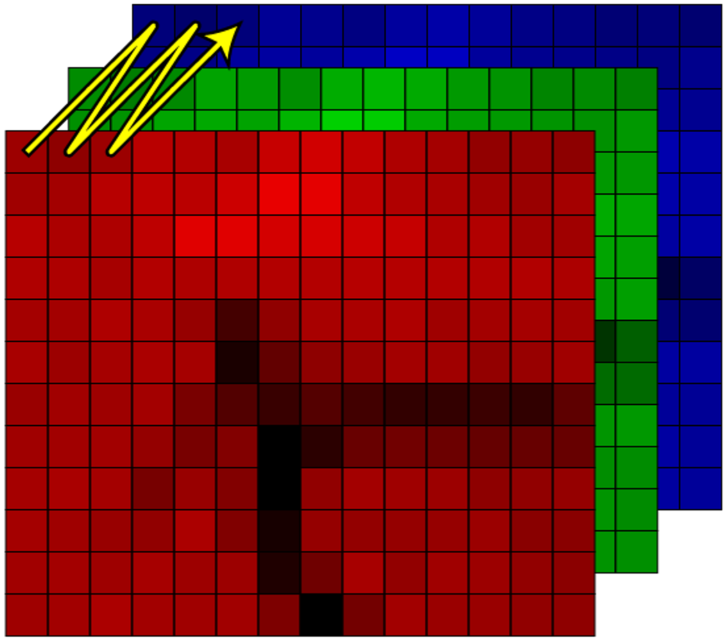


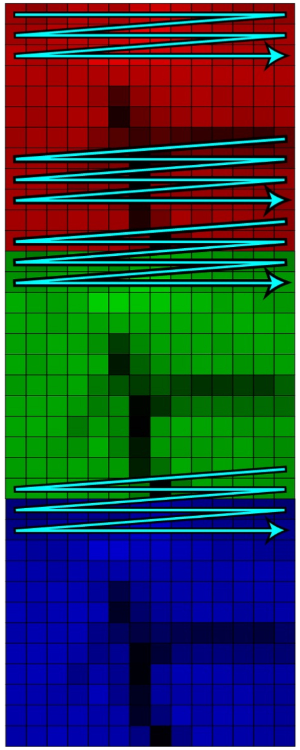
NumPy creates arrays in row-major order by default.

```
for i in range(width*height):
    curr_pix = img.flat[i]
```



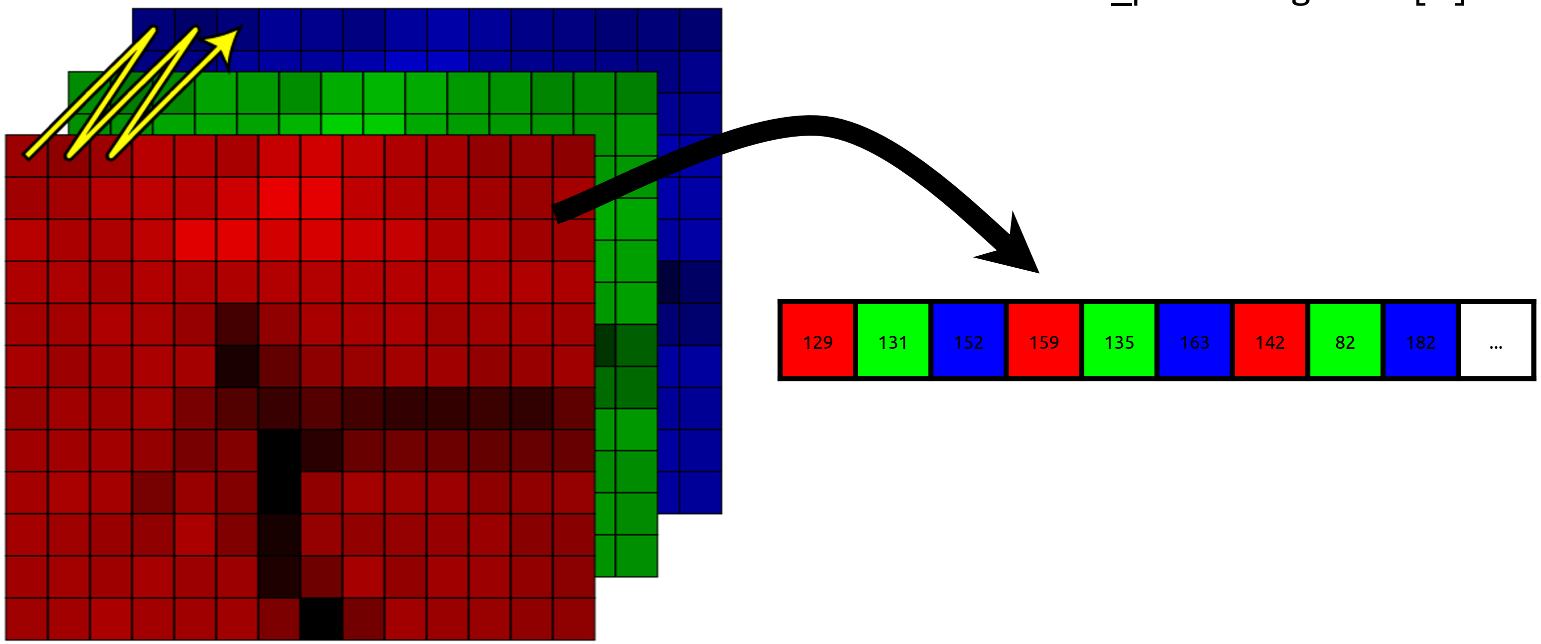
In 3D we have more choices!



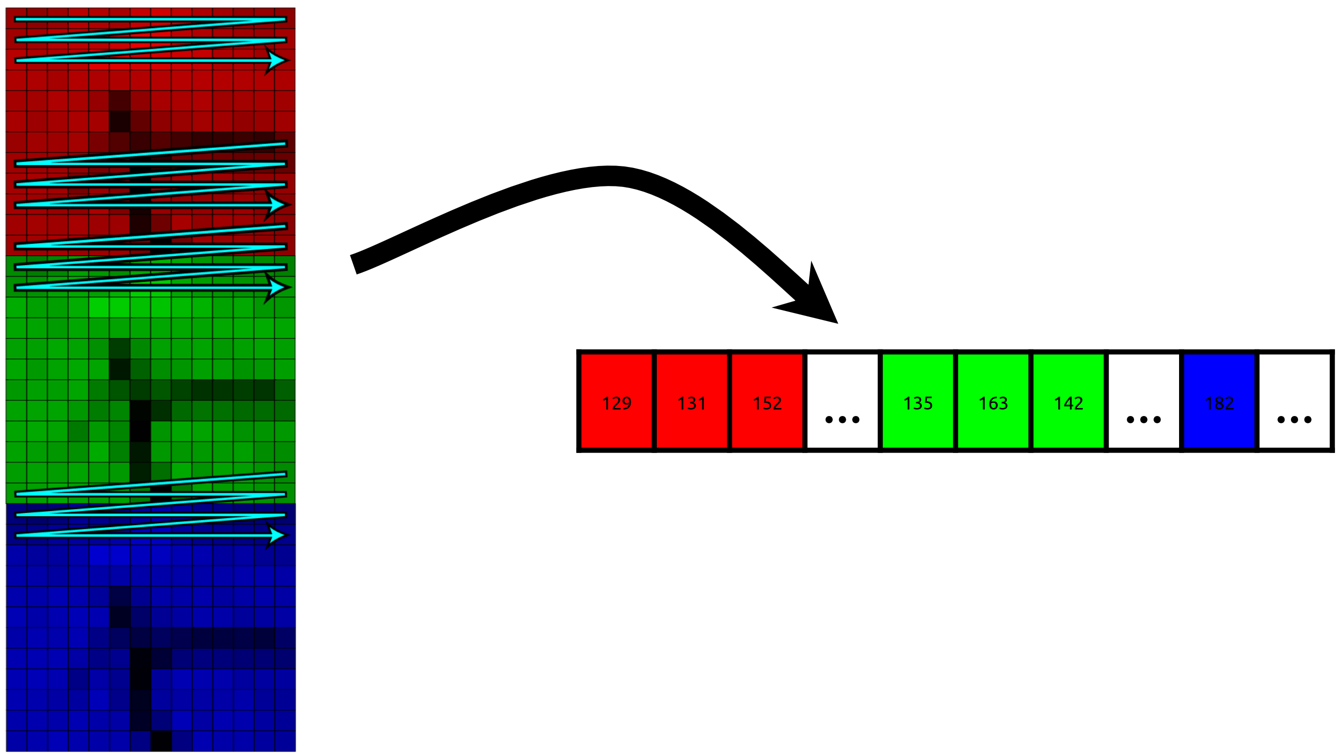


HWC: channels interleaved

```
for i in range(width*height*channels):
    curr_pix = img.flat[i]
```



CHW: channels separated



Numpy quiz


```
print(img.shape)
>>> (200, 280, 3)

# set some area of the image to zero
x = 200
y = 40
c = 1
img[y:y+20, x:x+20, c] = 0
```

How does the resulting image look like?