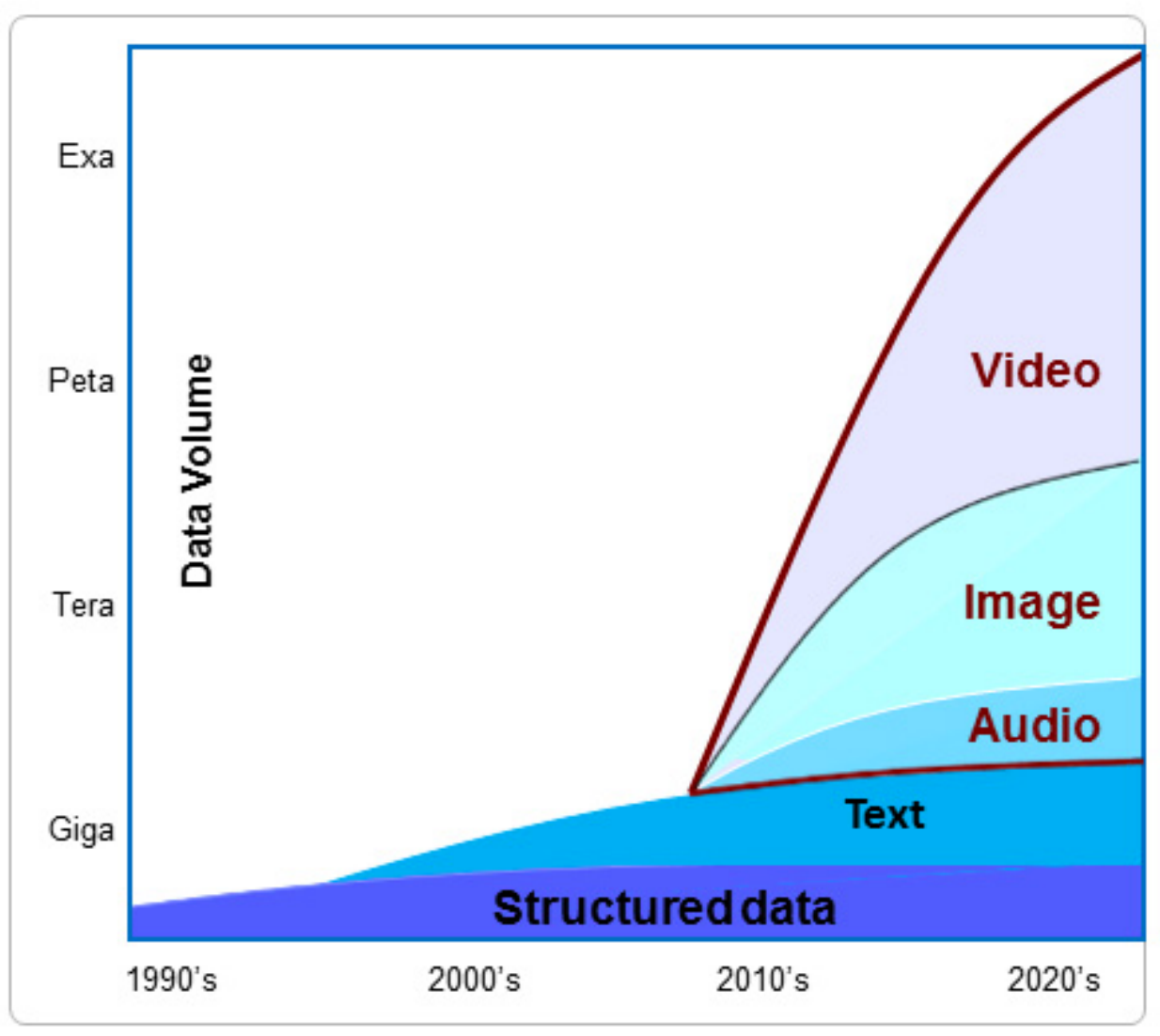
Image Processing

Goals

- Motivation and Applications
- Tools and Data pipeline
- Basic computer vision techniques

Image analysis is hard

Motivation


Exa
Peta
Tera
Giga
Data Volume
Video
Image
Audio
Text
Structured data
1990's
2000's
2010's
2020's

Application

Framework

- Read
- Resize
- Transformations
- Featurize

Python Libraries

- Scikit-image (Skimage)
- OpenCV (Based on C++)
  - ⋆ Face detection

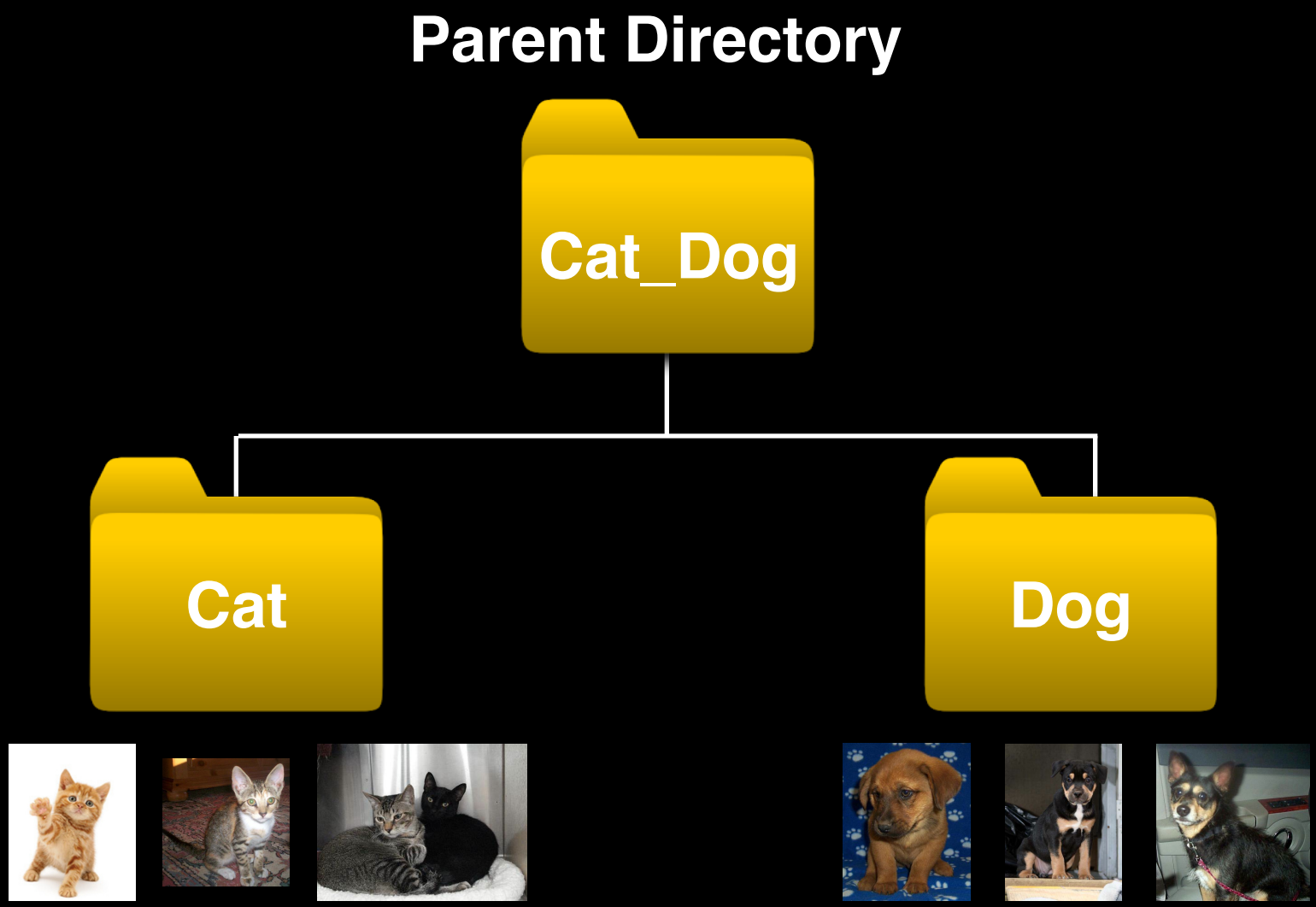
Framework

- Read
- Resize
- Transformations
- Featurize

Read

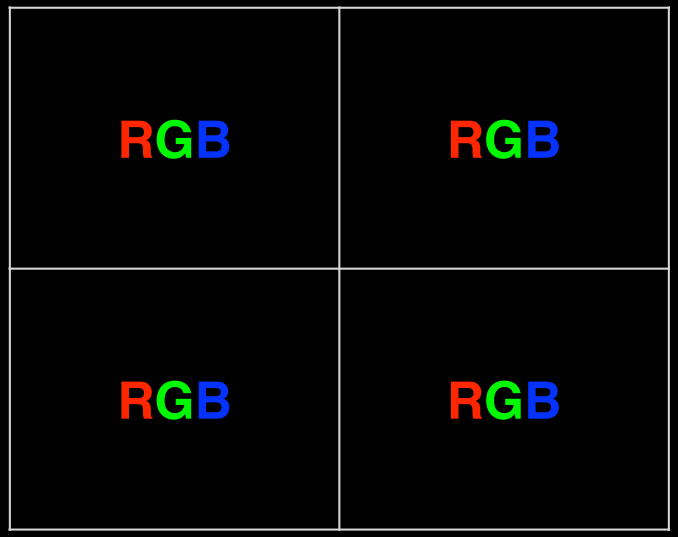
- Read an image into a matrix of numbers
- For image classification, read in sub-directories of images
- Each sub-directory is named after the label

Directory Structure



[[  ], [  ]]

Image Tensor




```
array([      R  G  B       R  G  B
        [ [108,50,13], [111,55,18] ],
          R  G  B       R  G  B
        [ [115,61,23], [130,129,127] ]

      ]
```

Framework

- Read
- Resize
- Transformations
- Featurize

Resize

- Make image a specified shape
- Not cropping
- Uniform feature matrix downstream

  (See Featurization later)

Resize Cont.

- For colored images:

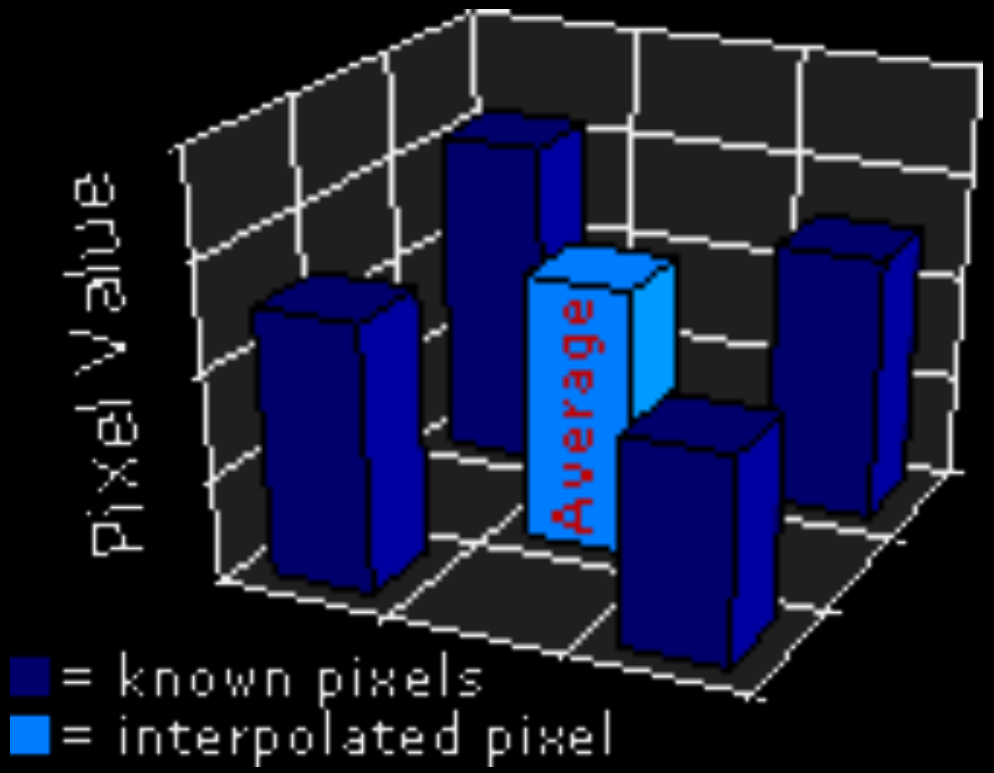
```
shape = (width, height, 3)
```

- For grayscale images:

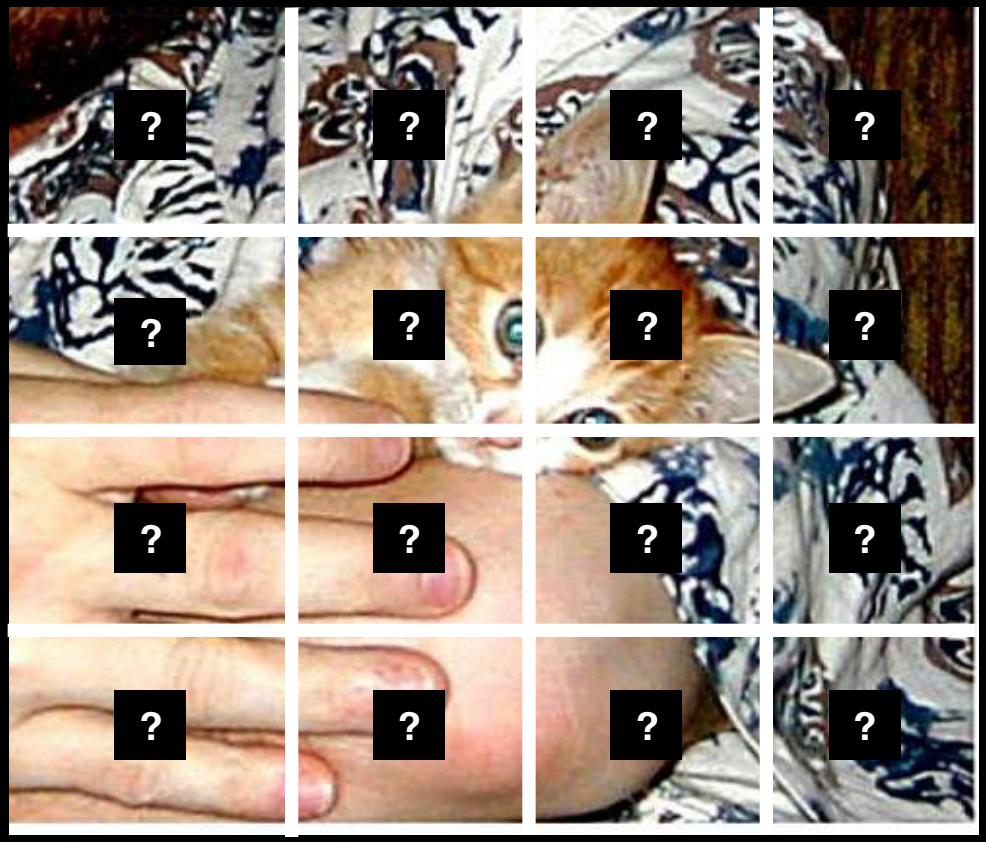
```
shape = (width, height)
```

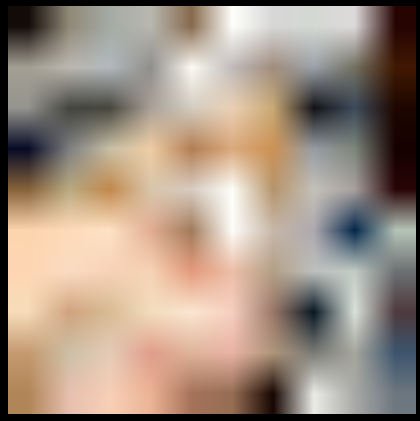
Interpolation

Bi-linear Interpolation



Down-Sample Interpolation





Up-Sample Interpolation

Down-sampling

- Reduce size / dimensions
- Less processing time downstream
- Lose resolution
- Visually verify

Framework

- Read
- Resize
- Transformations
- Featurize

Image Transformations

- Convert an image from one domain to another

Grayscale

- Compute luminance of the **RGB** image
- Y = 0.2125 **R** + 0.7154 **G** + 0.0721 **B**
- Normalize to range from 0 to 1
- 3D tensor matrix → 2D matrix

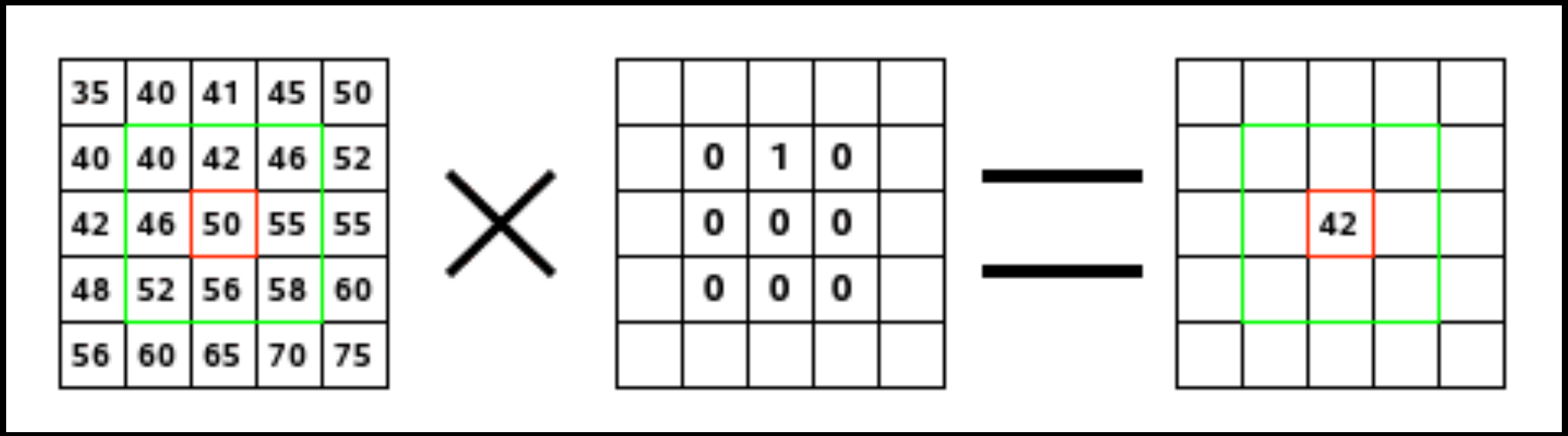
Denoise

- Removing unnecessary details of an image
- Better generalization of the class of image


LIBERTY
IN GOD
WE TRUST
2000
S


LIBERTY

Method 1: Convolution



Input Kernel Output

Gaussian Kernel

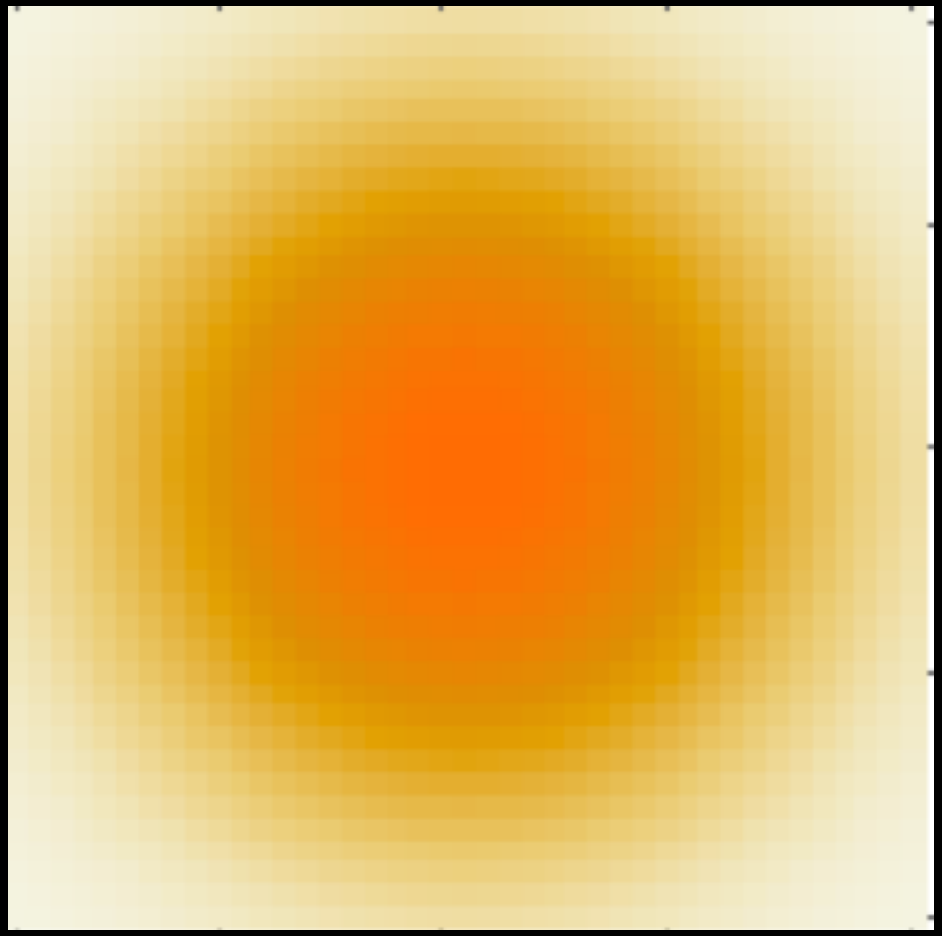
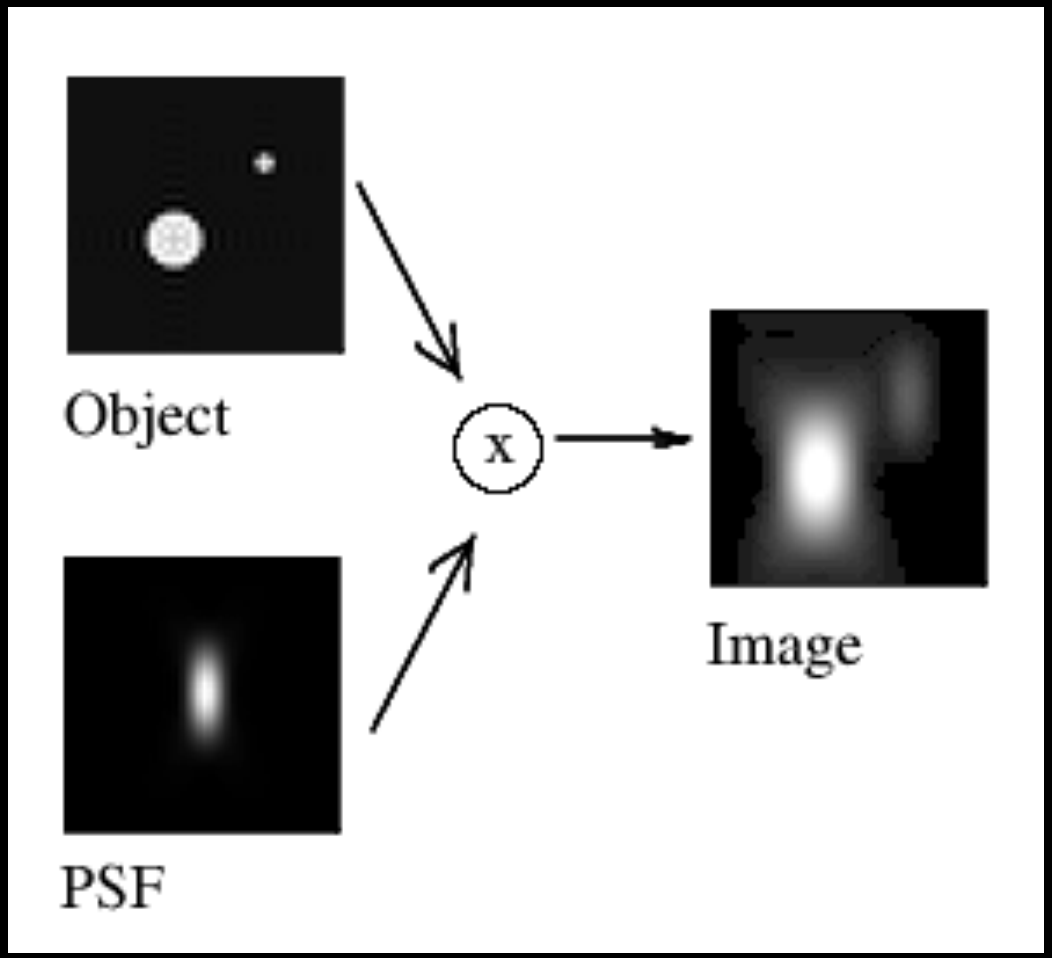


Image Convolution



Bi-lateral Denoise

- **Bilateral:** *Spatial* and *Pixel value*
- Convolution with a kernel
- Usually Gaussian Kernel

Bi-lateral Denoise Cont.

- Higher weights for spatially close pixel
- Higher weights for pixel value similarity

Method 2: Variation Minimization

Fidelity: $E(x,y) = \frac{1}{2}\sum_n (x_n - y_n)^2$

Variation: $V(y) = \sum_n |y_{n+1} - y_n|$

Minimize: $E(x,y) + \lambda V(y)$

How much to denoise

$\mathbf{x_n}$ is the original data

$\mathbf{y_n}$ is denoised data

Total Variation Denoise

- Takes much longer if the image is noisy
- Can adjust λ

Edge Detection

- Only works with grayscale images
- **Edge**:
  - ⋆ Large change of color between neighboring pixels

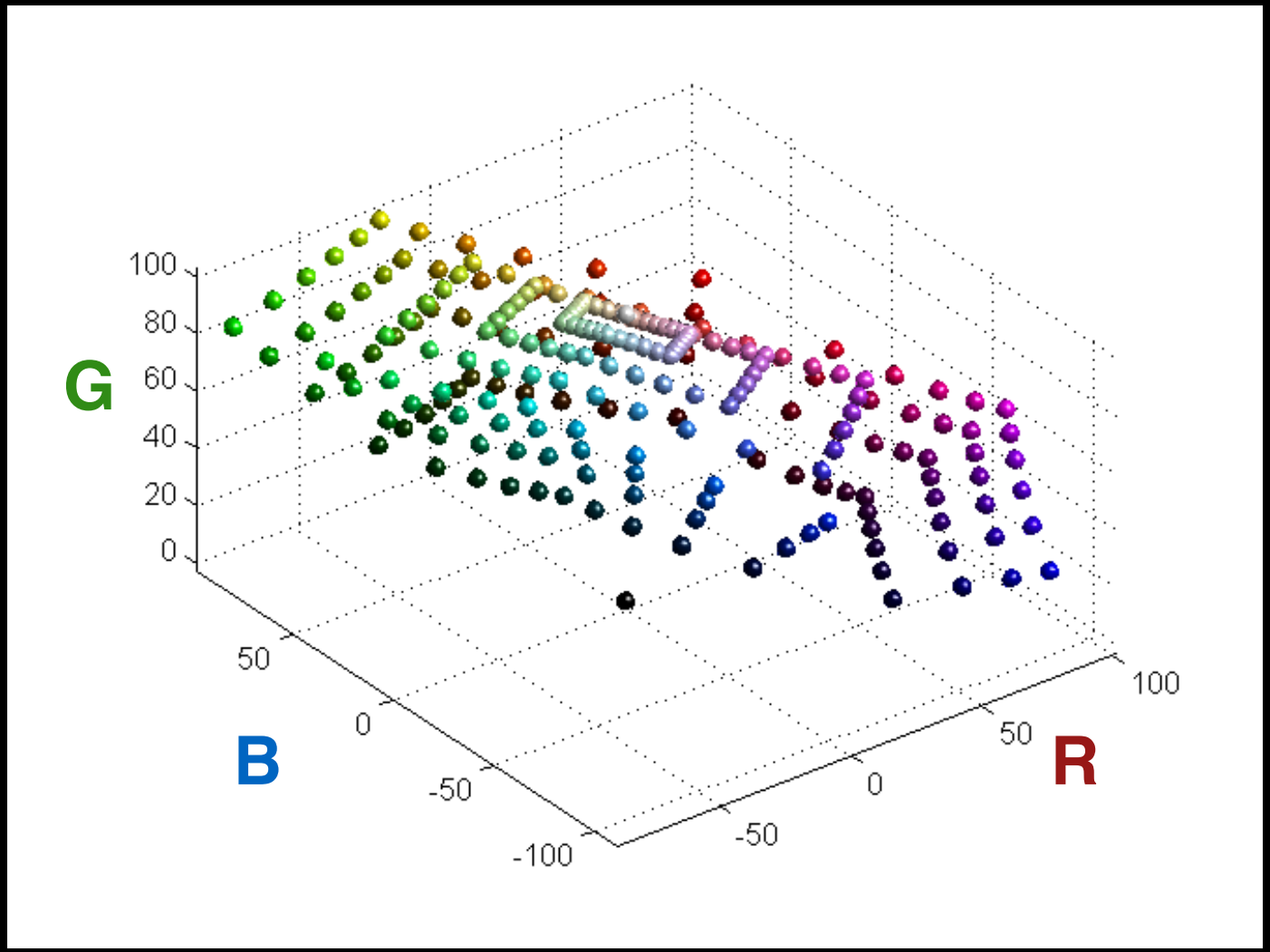
Canny

Sobel

Dominant Colors

- Most common colors in the image
- KMeans to cluster the RGB pixels
- Centriods represent the dominant colors

K-Means of RGB pixels

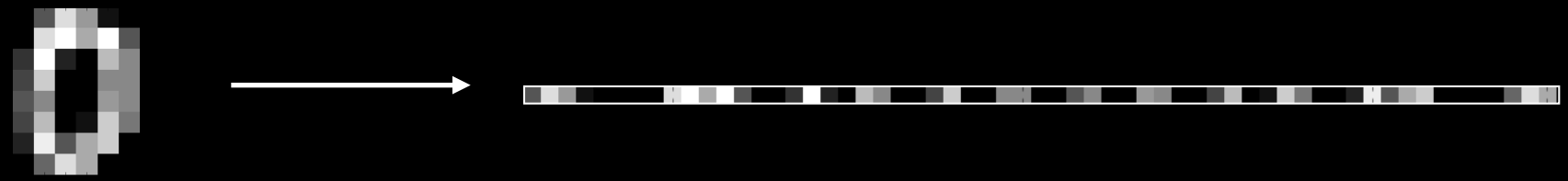


Framework

- Read
- Resize
- Transformations
- Featurize

Vectorize Image

`numpy.ravel()`



`numpy.ravel()`