Jump into filtering

IMAGE PROCESSING IN PYTHON

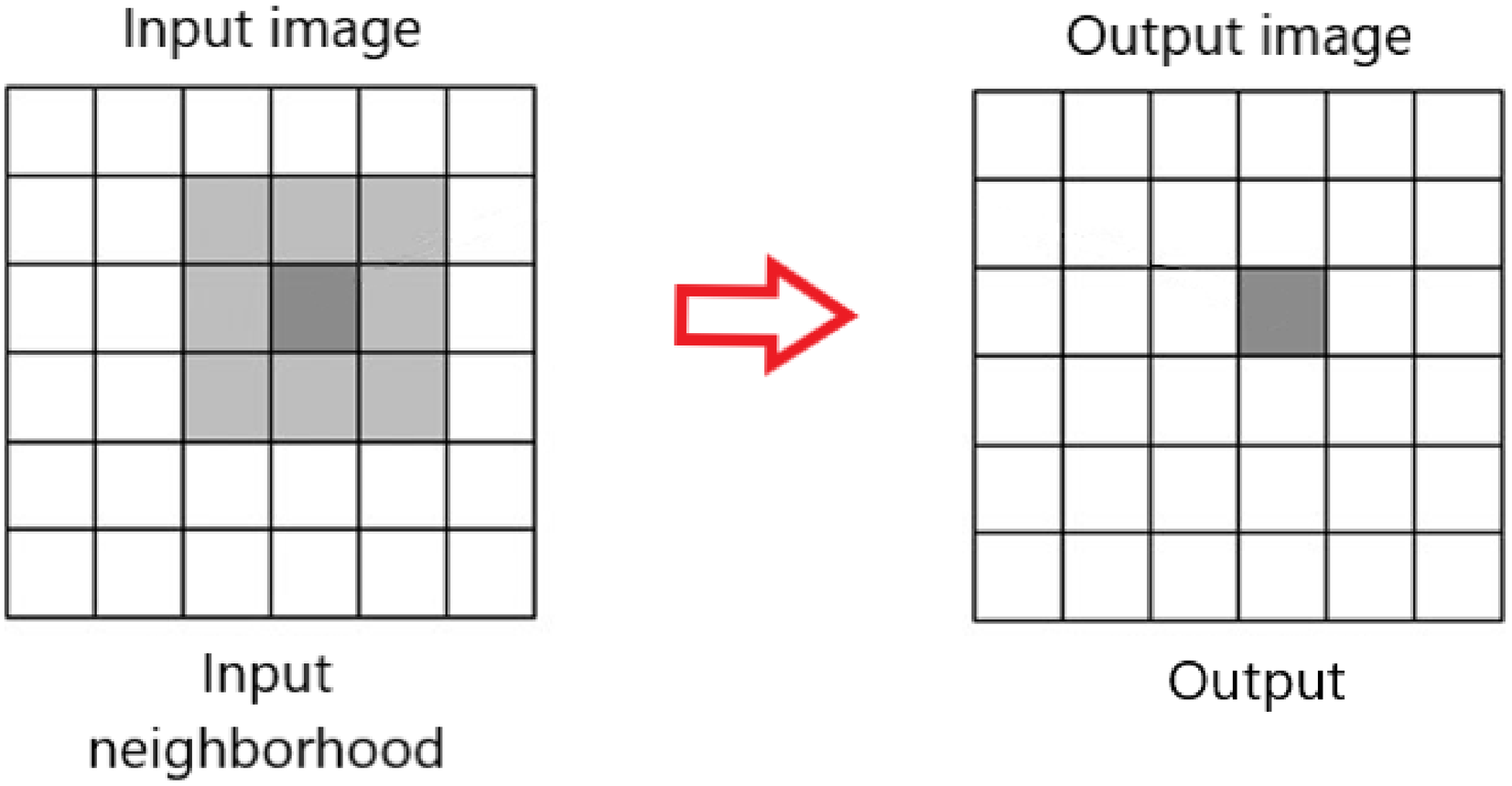
Rebeca Gonzalez
Data Engineer

datacamp

Filters

- Enhancing an image
- Emphasize or remove features
- Smoothing
- Sharpening
- Edge detection

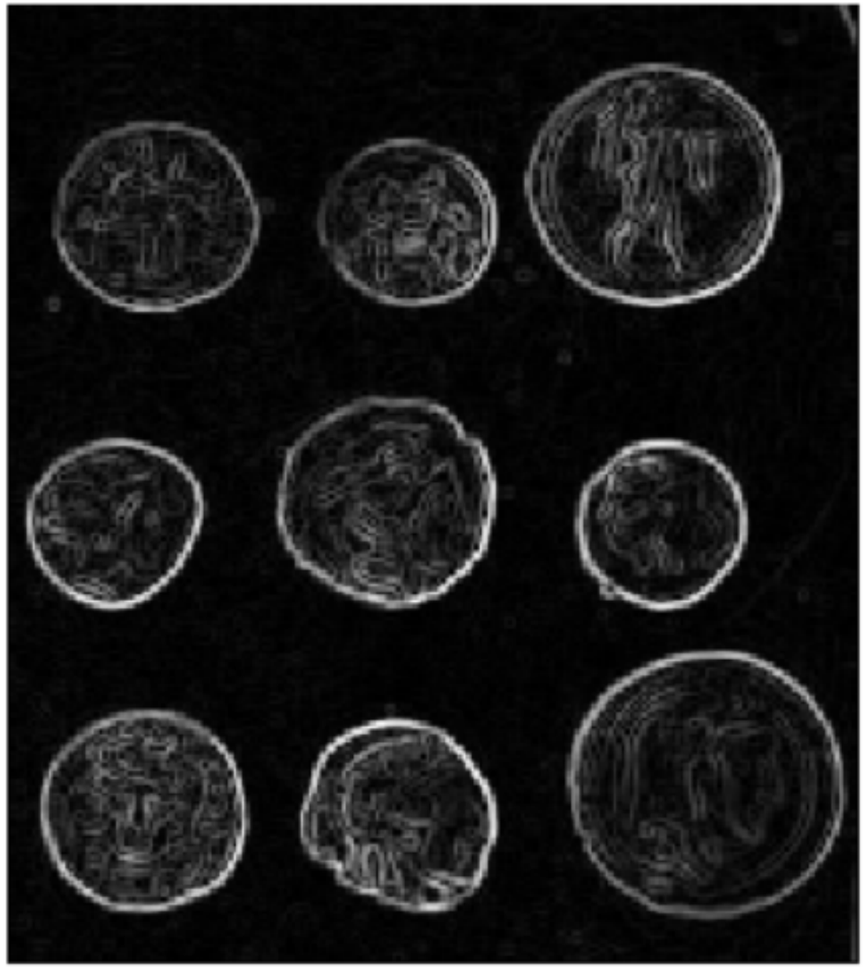
Neighborhoods



Edge detection

Original

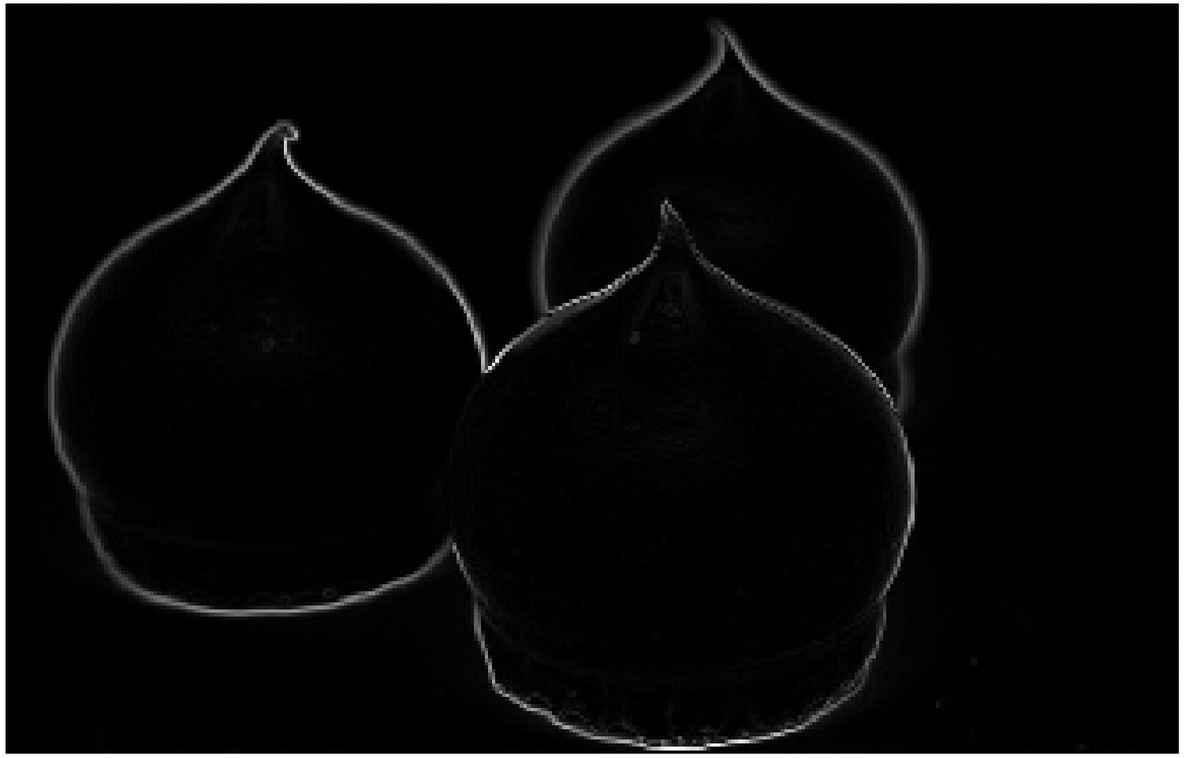
Edges with Sobel



Edge detection

Original chocolate kisses

Edges with Sobel



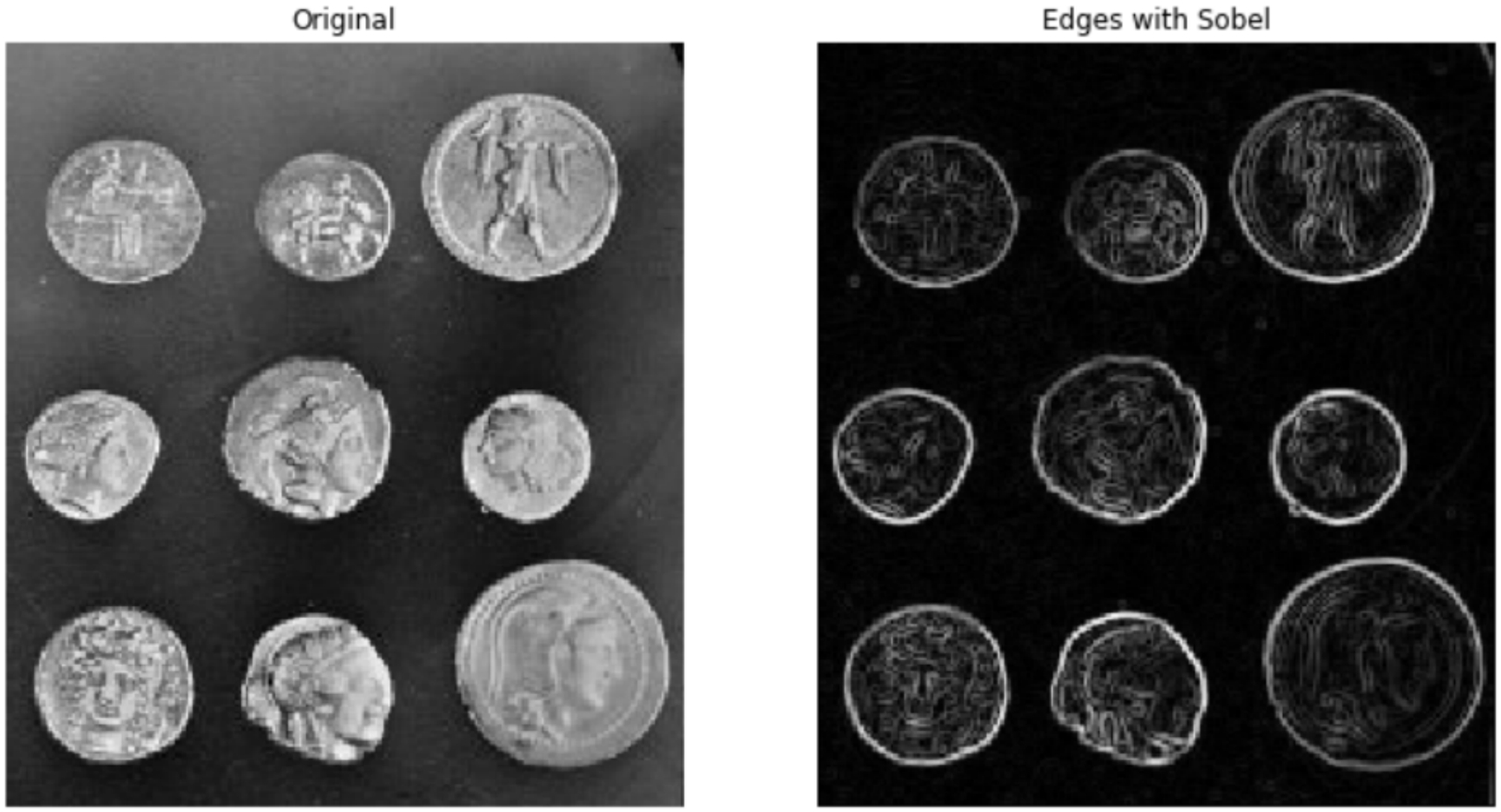
Edge detection

Sobel

```
# Import module and function
from skimage.filters import sobel

# Apply edge detection filter
edge_sobel = sobel(image_coins)

# Show original and resulting image to compare
plot_comparison(image_coins, edge_sobel, "Edge with Sobel")
```

Edge detection

Sobel



Comparing plots

```
def plot_comparison(original, filtered, title_filtered):

    fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(8, 6), sharex=True,
                                   sharey=True)
    ax1.imshow(original, cmap=plt.cm.gray)
    ax1.set_title('original')
    ax1.axis('off')
    ax2.imshow(filtered, cmap=plt.cm.gray)
    ax2.set_title(title_filtered)
    ax2.axis('off')
```

Gaussian smoothing

Original

Blurred with Gaussian filter

Gaussian smoothing

Gaussian smoothing

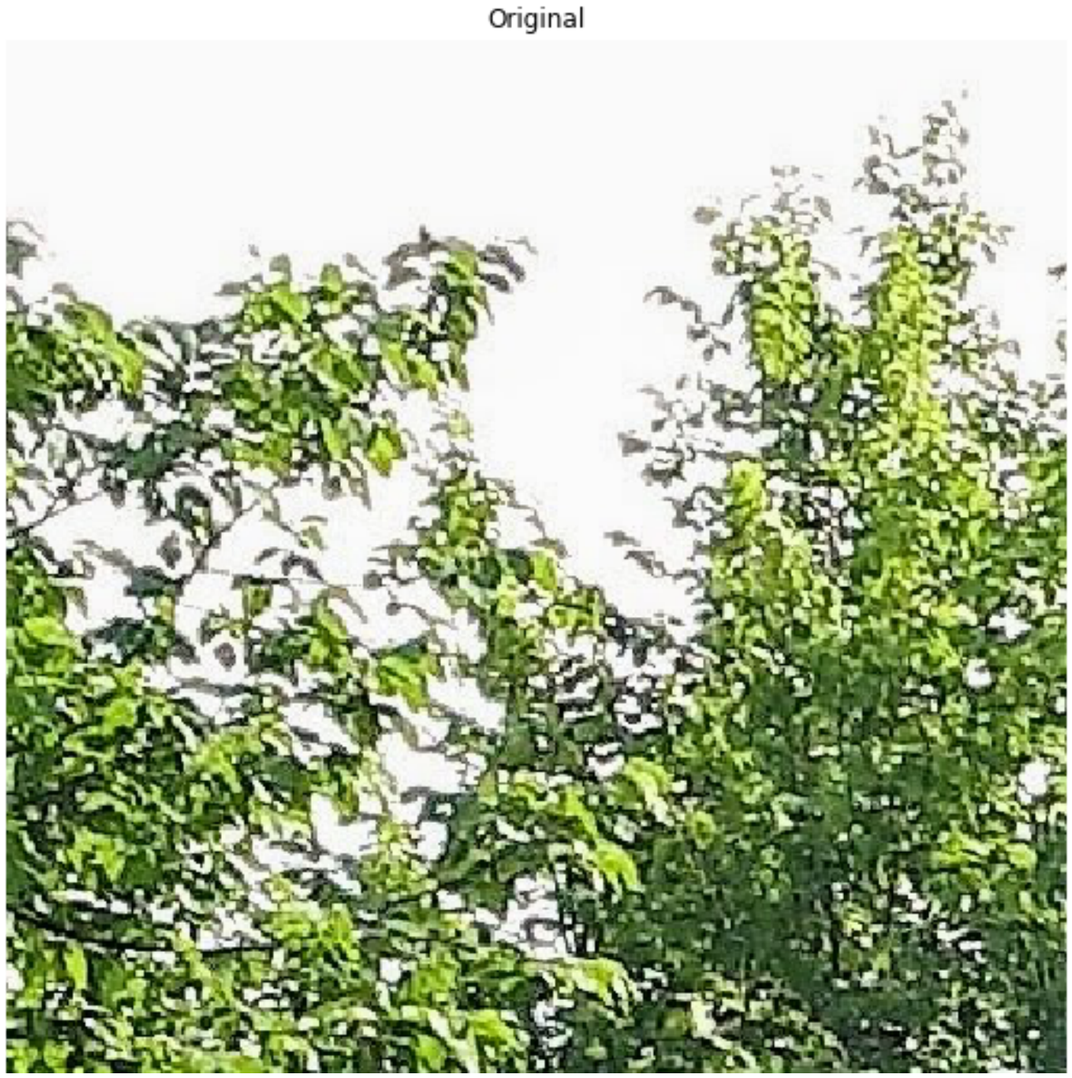
```
# Import the module and function
from skimage.filters import gaussian

# Apply edge detection filter
gaussian_image = gaussian(amsterdam_pic, multichannel=True)

# Show original and resulting image to compare
plot_comparison(amsterdam_pic, gaussian_image, "Blurred with Gaussian filter")
```

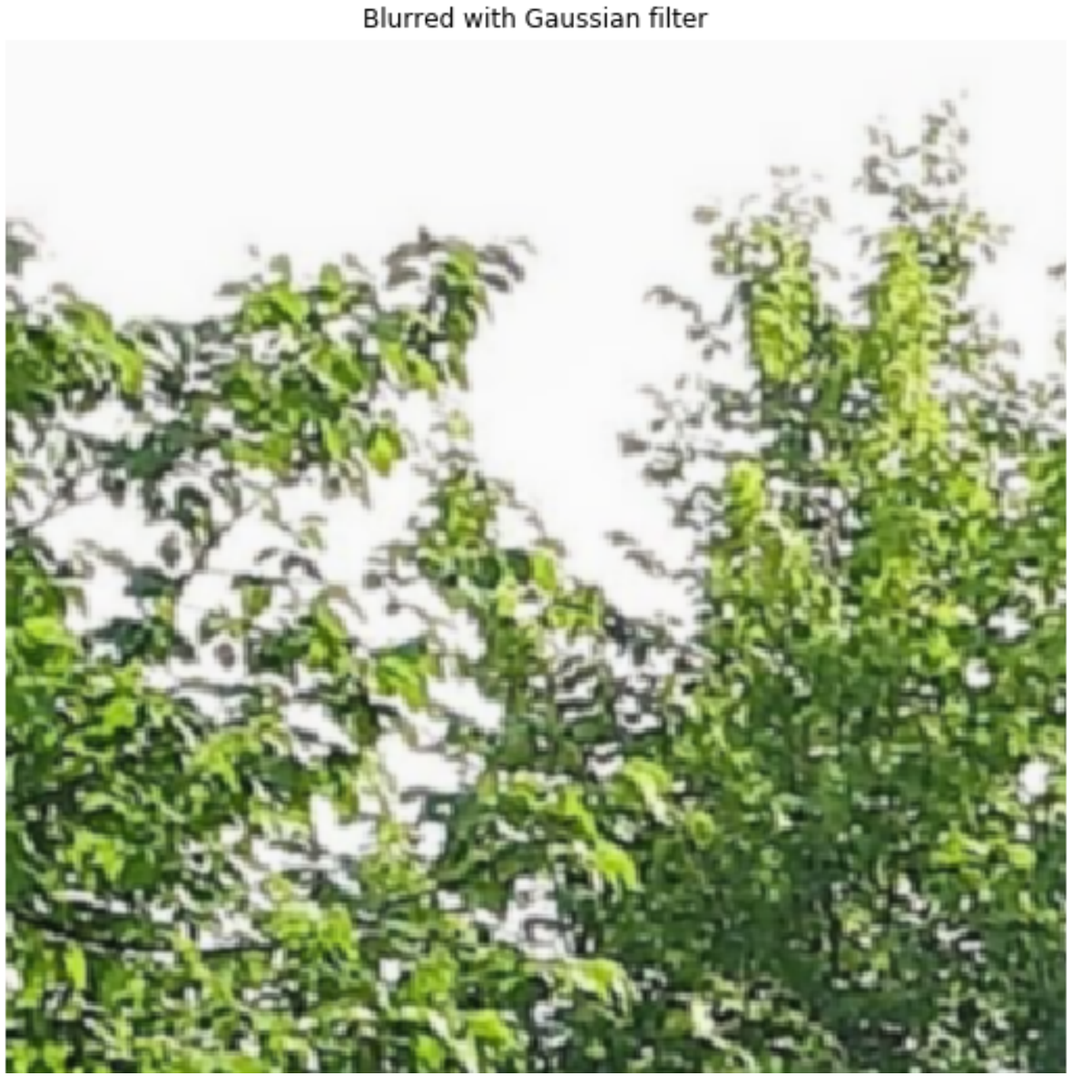
Gaussian smoothing

Original

Blurred with Gaussian filter

Gaussian smoothing





Let's practice!

IMAGE PROCESSING IN PYTHON

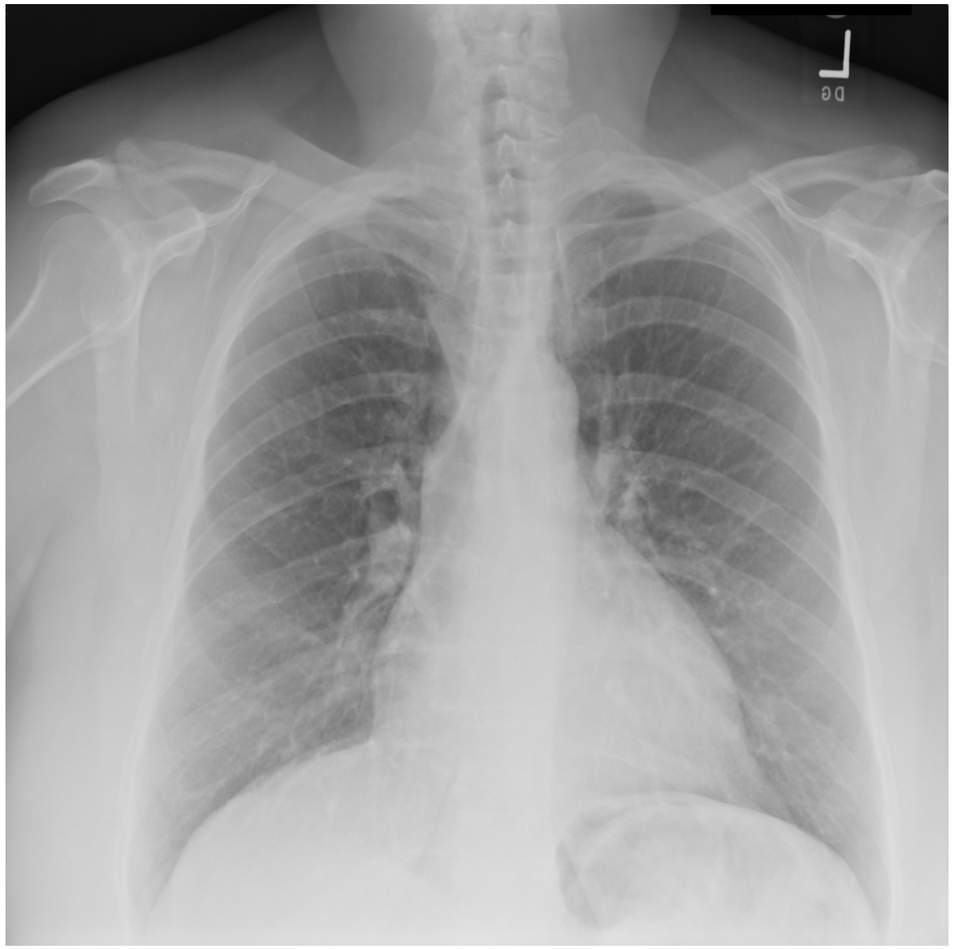
Contrast enhancement

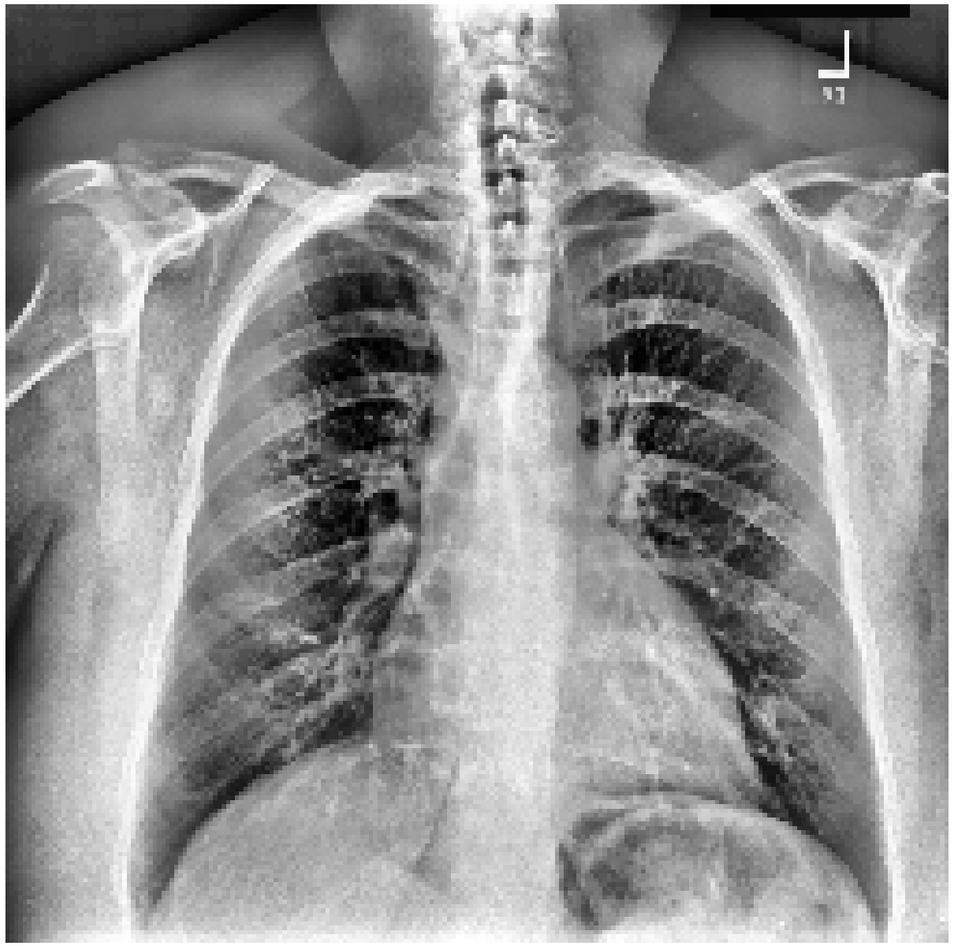
IMAGE PROCESSING IN PYTHON

Rebeca Gonzalez
Data engineer

datacamp

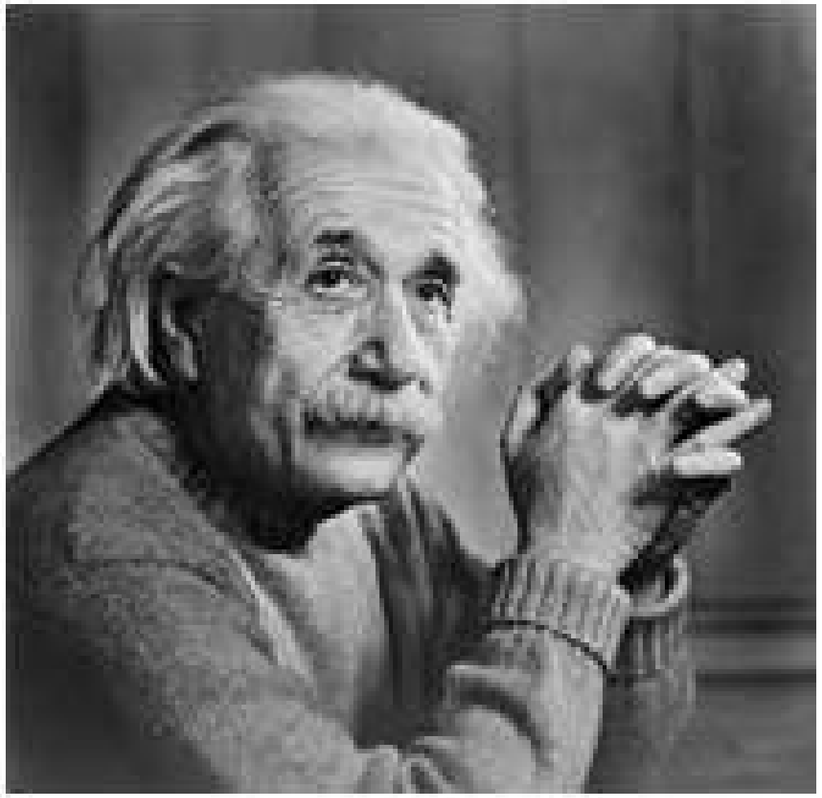
Contrast enhancement

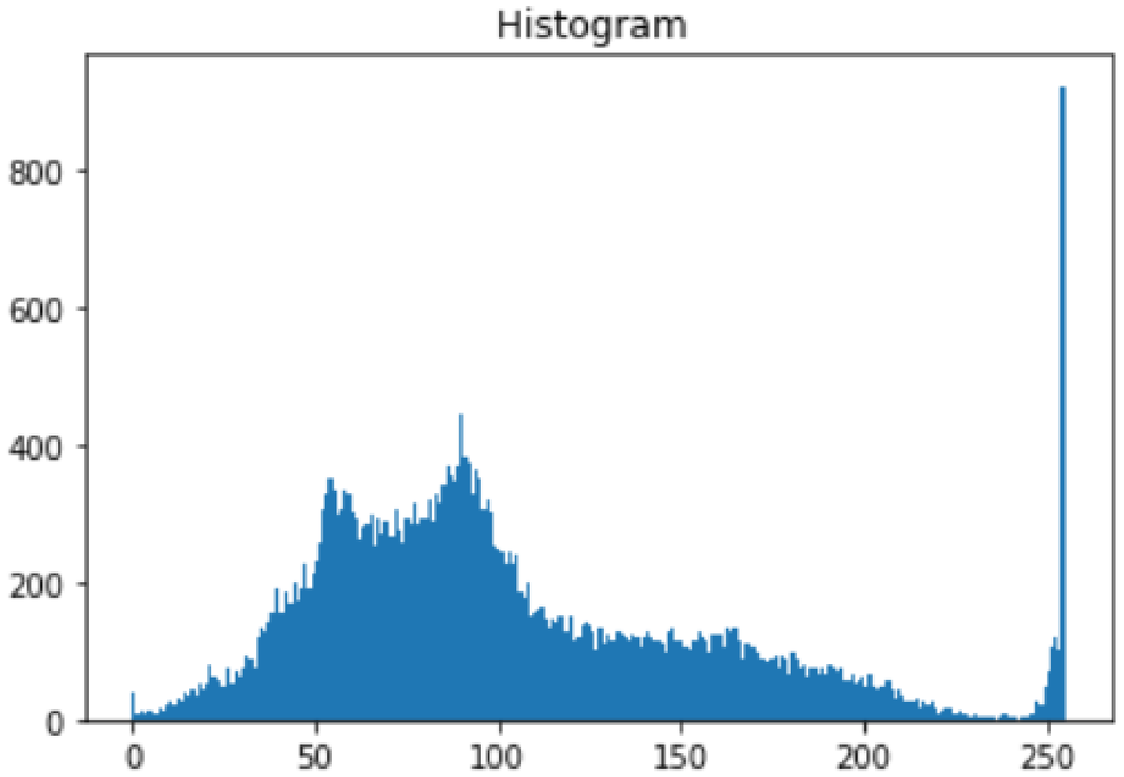




Contrast

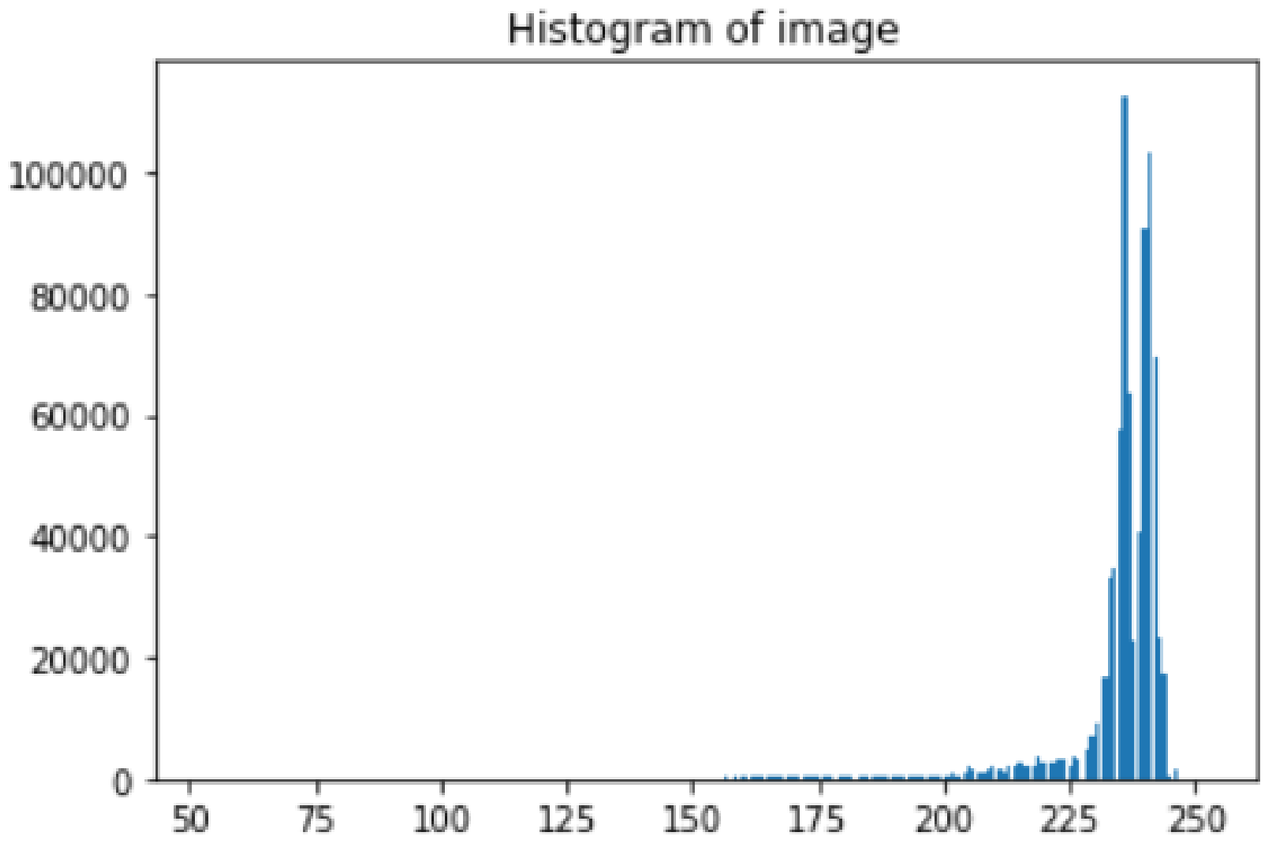
Histograms for contrast enhancement





Contrast


Low contrast image - light


Histogram of image
100000
80000
60000
40000
20000
0
50
75
100
125
150
175
200
225
250

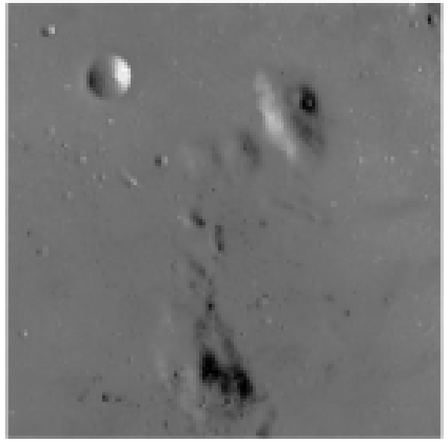
Enhance contrast

- Contrast stretching
- Histogram equalization

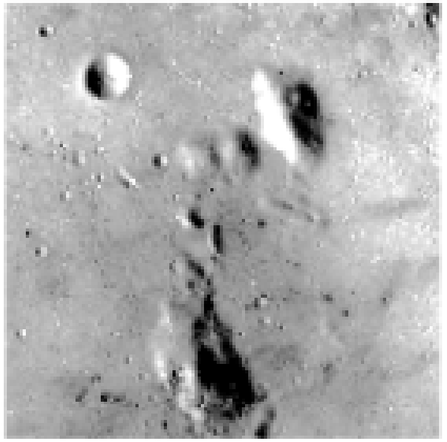
Types

- Histogram equalization
- Adaptive histogram equalization
- Contrast Limited Adaptive Histogram Equalization (CLAHE)

Low contrast image



Contrast stretching

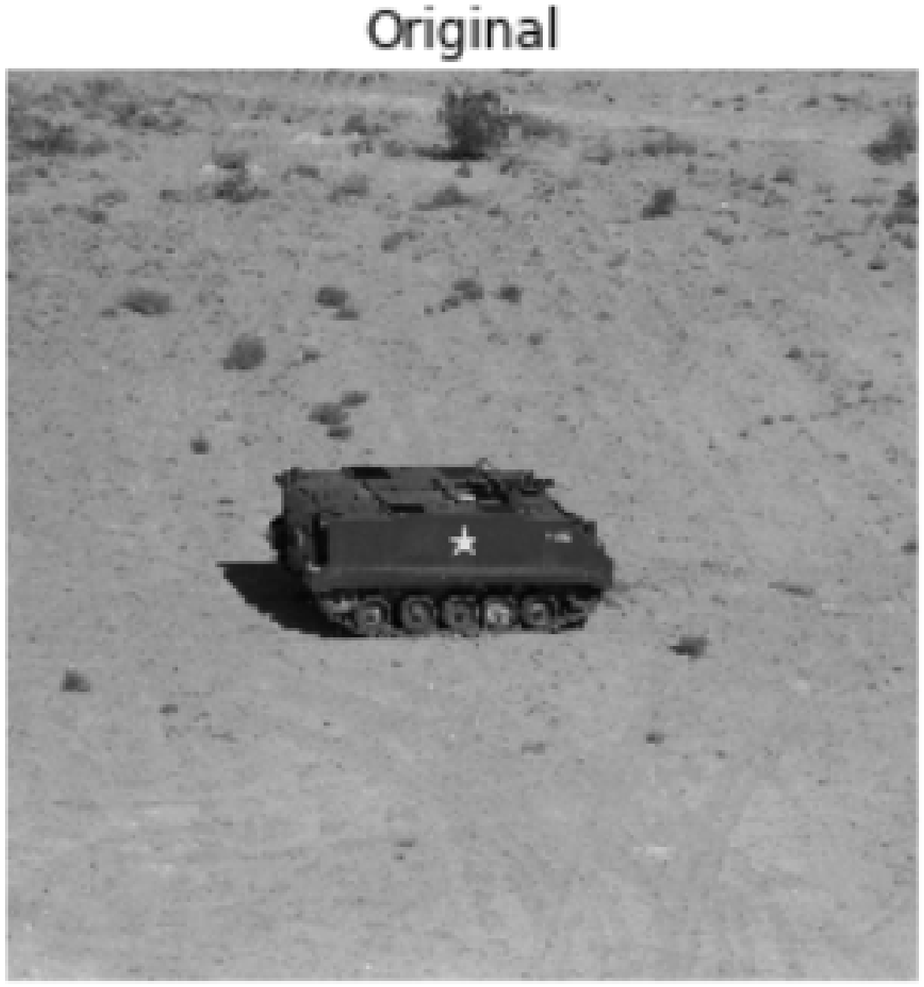


Histogram equalization

Adaptive equalization

Histogram equalization

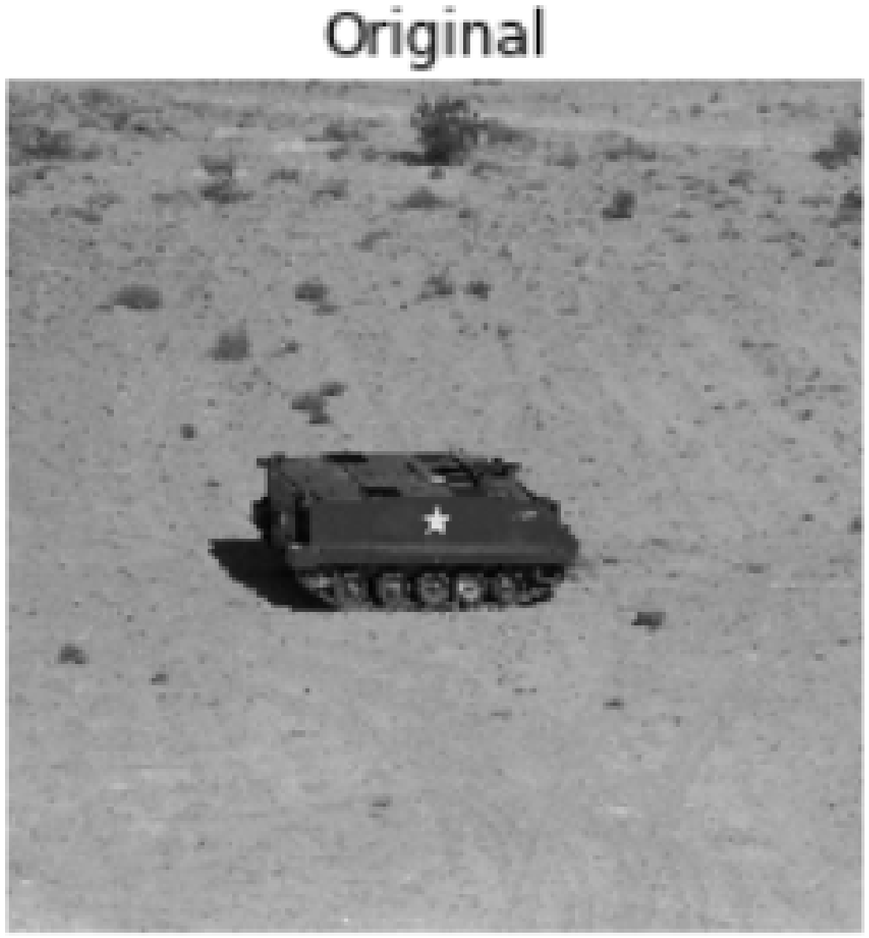
Histogram equalization

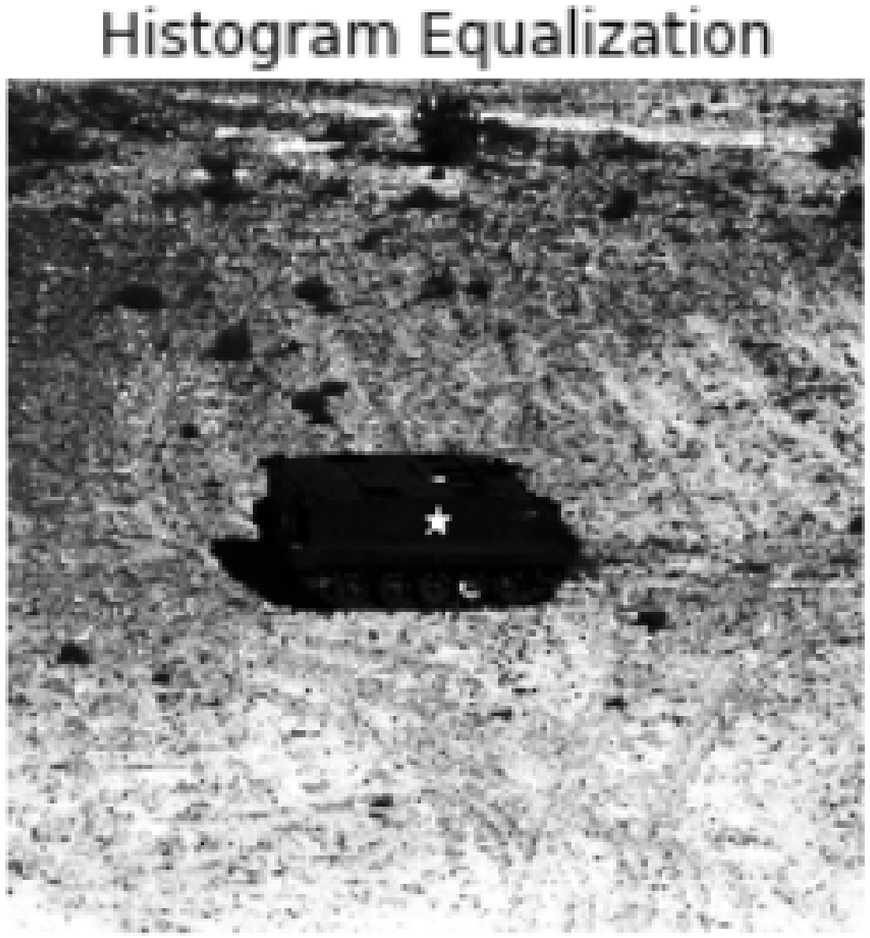


Histogram equalization

```
from skimage import exposure

# Obtain the equalized image
image_eq = exposure.equalize_hist(image)

# Show original and result
show_image(image, 'Original')
show_image(image_eq, 'Histogram equalized')
```

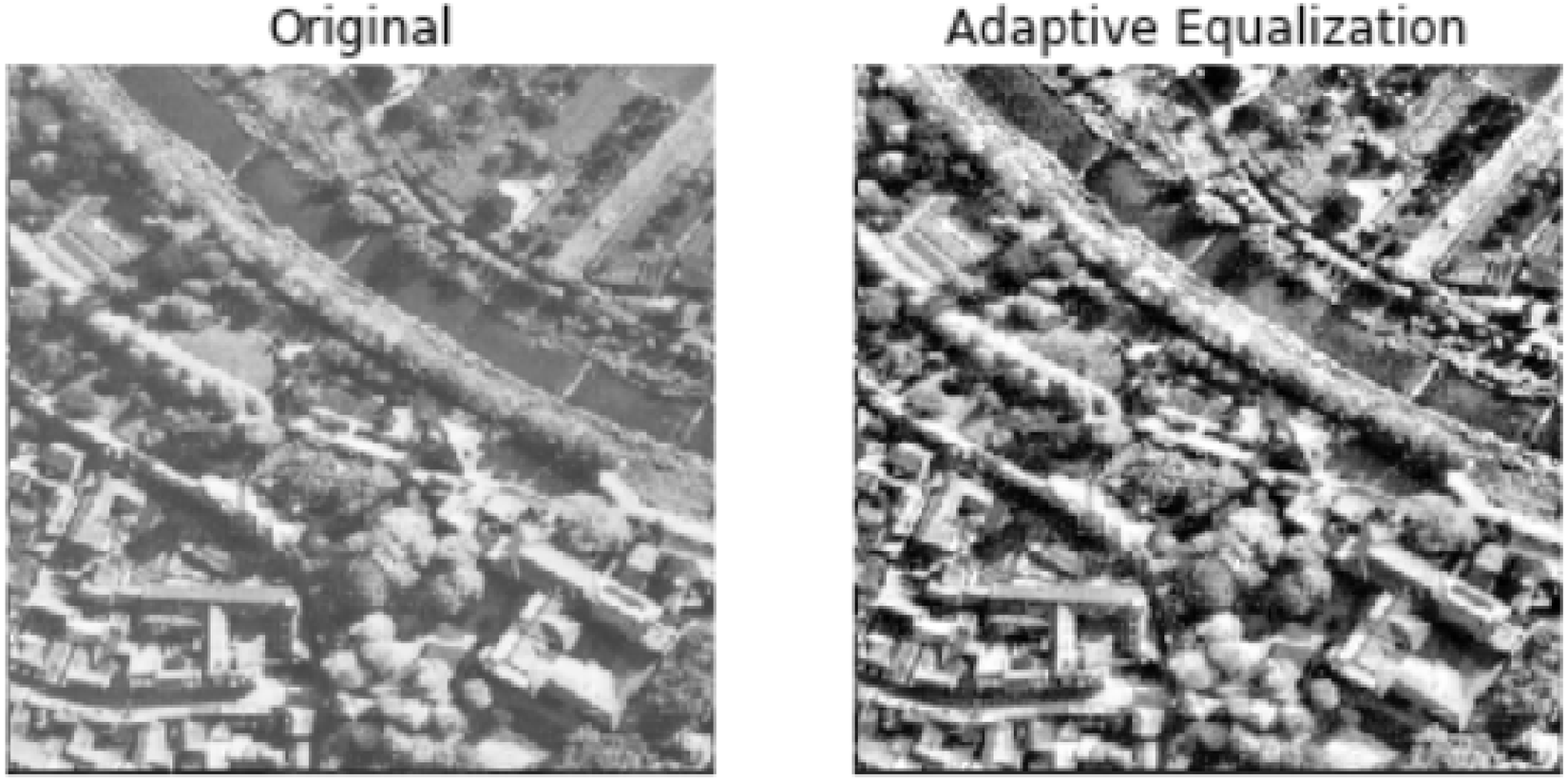
Histogram equalization





Adaptive Equalization

- Contrastive Limited Adaptive Histogram Equalization



Contrastive Limited Adaptive Equalization

Original

Histogram Equalization

Adaptive Equalization

CLAHE in scikit-image

```
from skimage import exposure

# Apply adaptive Equalization
image_adapteq = exposure.equalize_adapthist(image, clip_limit=0.03)

# Show original and result
show_image(image, 'Original')
show_image(image_adapteq, 'Adaptive equalized')
```

CLAHE in scikit-image

Original

Adaptive Equalization

Let's practice!

IMAGE PROCESSING IN PYTHON

Transformations

IMAGE PROCESSING IN PYTHON

Rebeca Gonzalez
Data Engineer

datacamp

Why transform images?

- Preparing images for classification Machine Learning models
- Optimization and compression of images
- Save images with same proportion

Rotating

Original

Rotated 90 degrees clockwise

Rotating

Original

Rotated 90 degrees anticlockwise

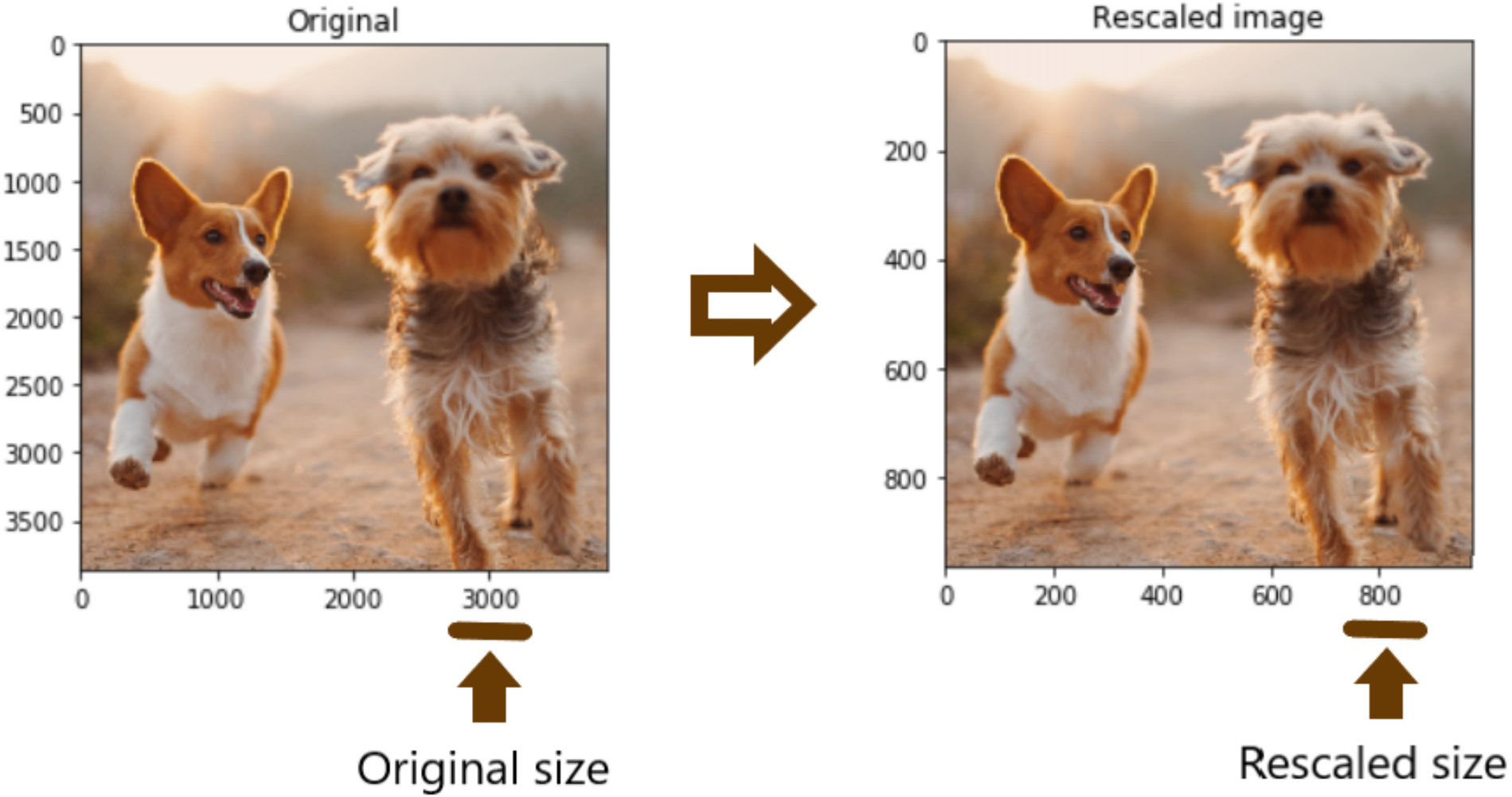
Rotating clockwise

```
from skimage.transform import rotate

# Rotate the image 90 degrees clockwise
image_rotated = rotate(image, -90)

show_image(image, 'Original')
show_image(image_rotated, 'Rotated 90 degrees clockwise')
```

Rotating anticlockwise

```
from skimage.transform import rotate

# Rotate an image 90 degrees anticlockwise
image_rotated = rotate(image, 90)

show_image(image, 'Original')
show_image(image_rotated, 'Rotated 90 degrees anticlockwise')
```

Rescaling

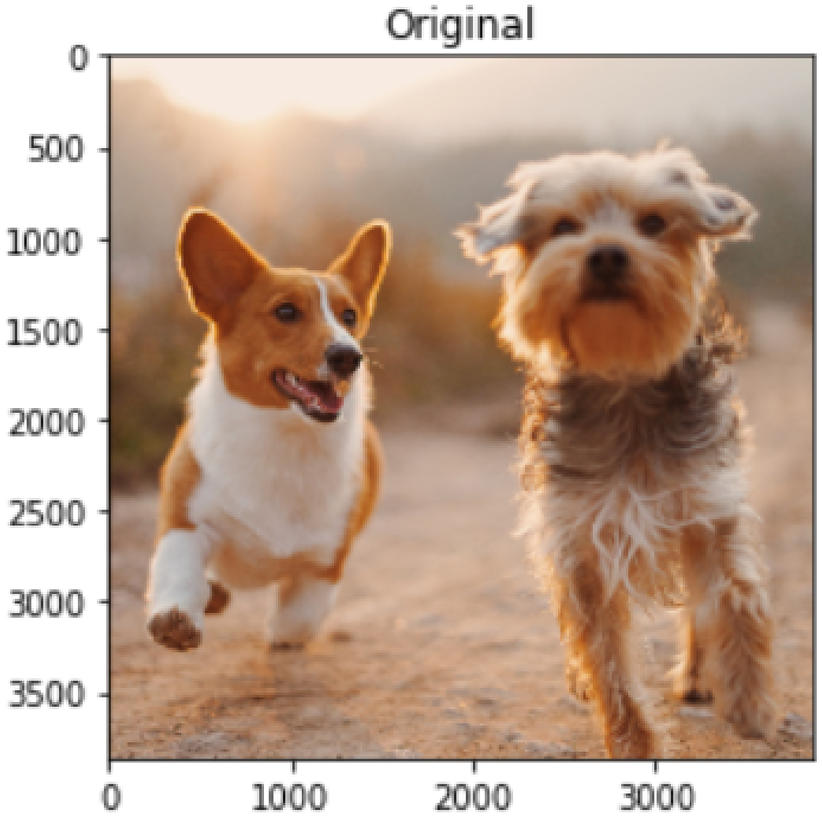

Original
0
500
1000
1500
2000
2500
3000
3500
0
1000
2000
3000
Original size
Rescaled image
0
200
400
600
800
0
200
400
600
800
Rescaled size

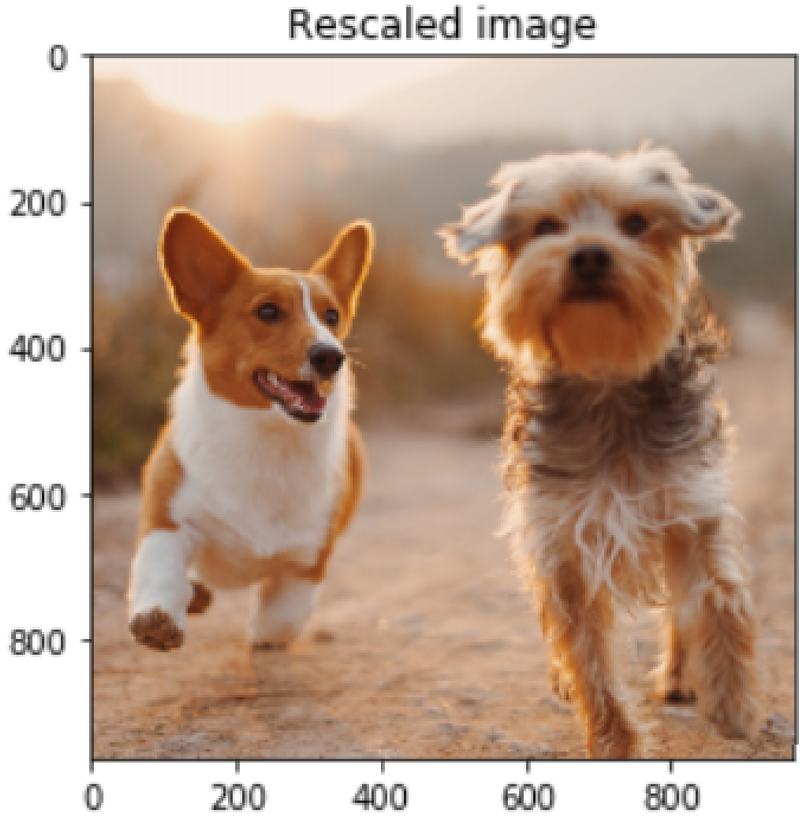
Rescaling

Downgrading

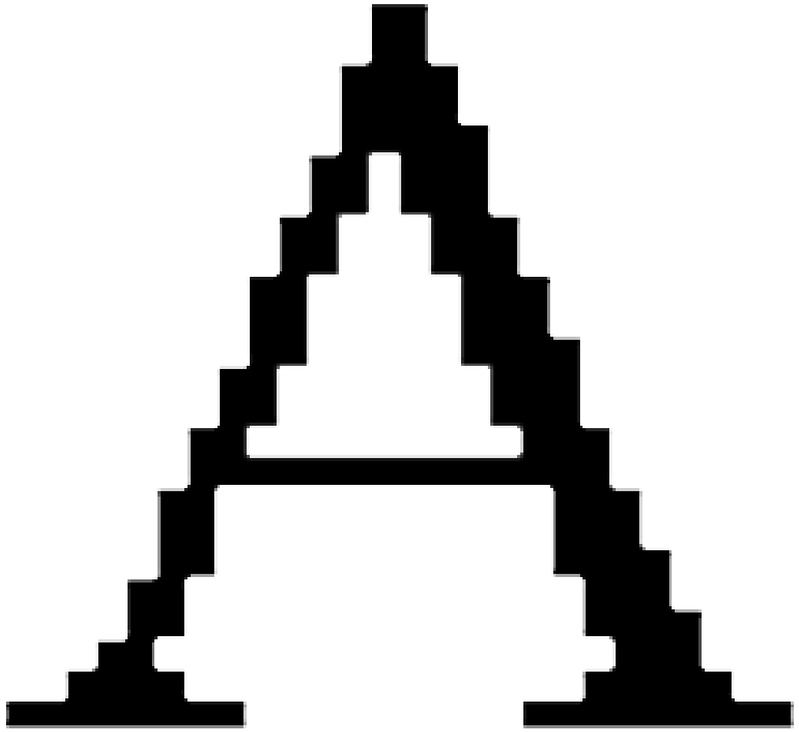
```
from skimage.transform import rescale

# Rescale the image to be 4 times smaller
image_rescaled = rescale(image, 1/4, anti_aliasing=True, multichannel=True)

show_image(image, 'Original image')
show_image(image_rescaled, 'Rescaled image')
```

Rescaling

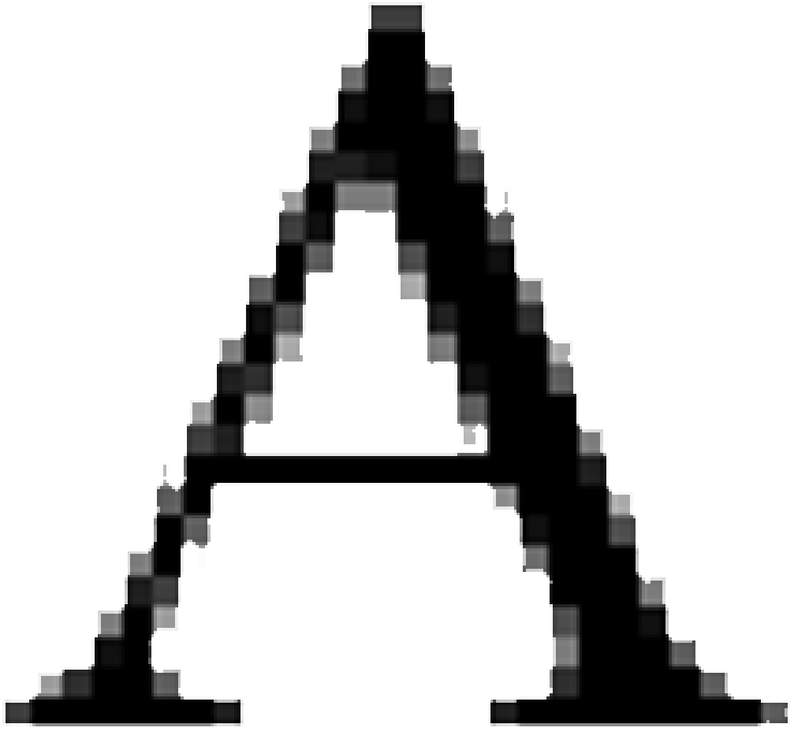

Original
0
500
1000
1500
2000
2500
3000
3500
0
1000
2000
3000


Rescaled image
0
200
400
600
800
0
200
400
600
800

Aliasing in digital images

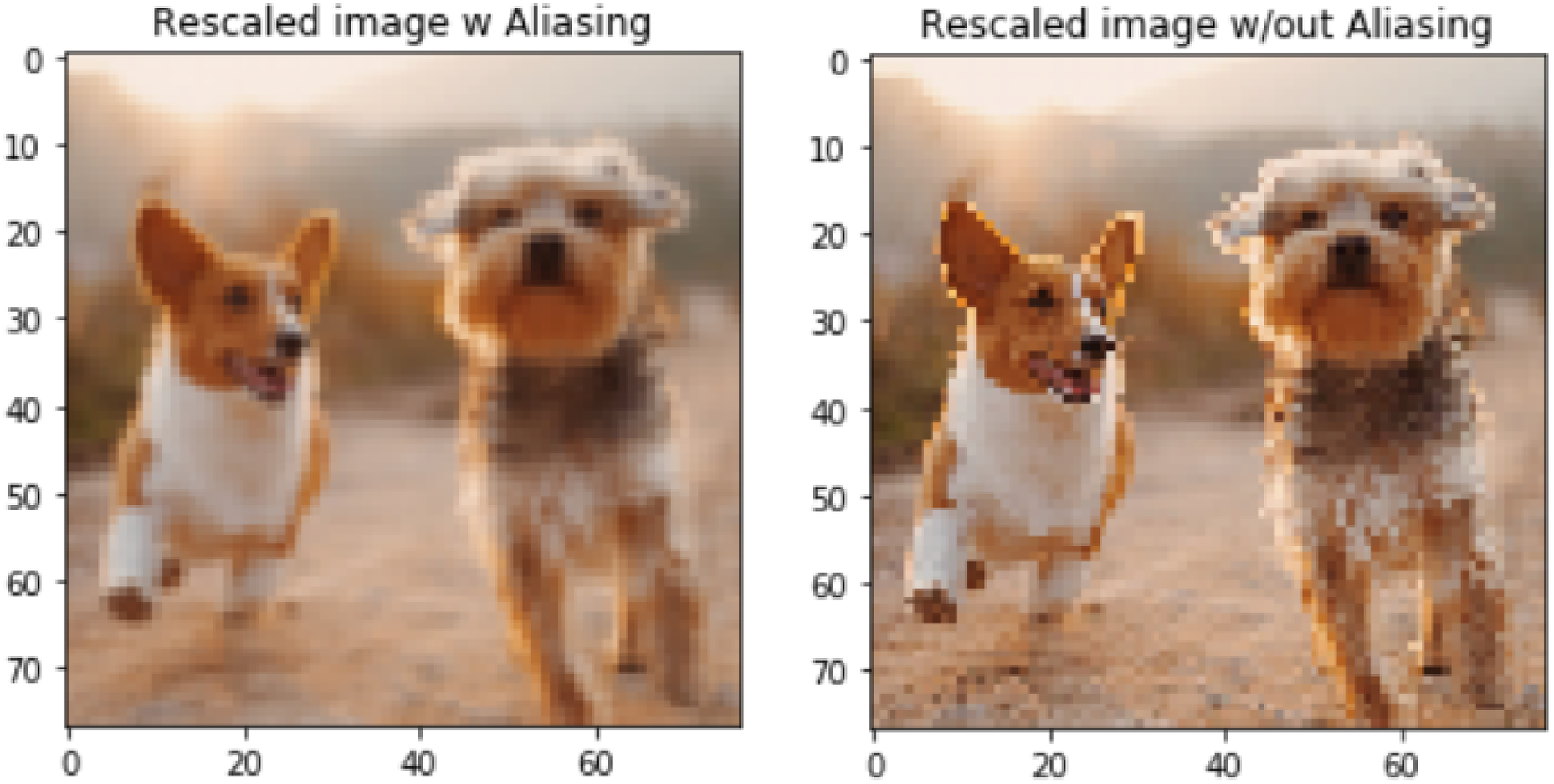


Original letter A

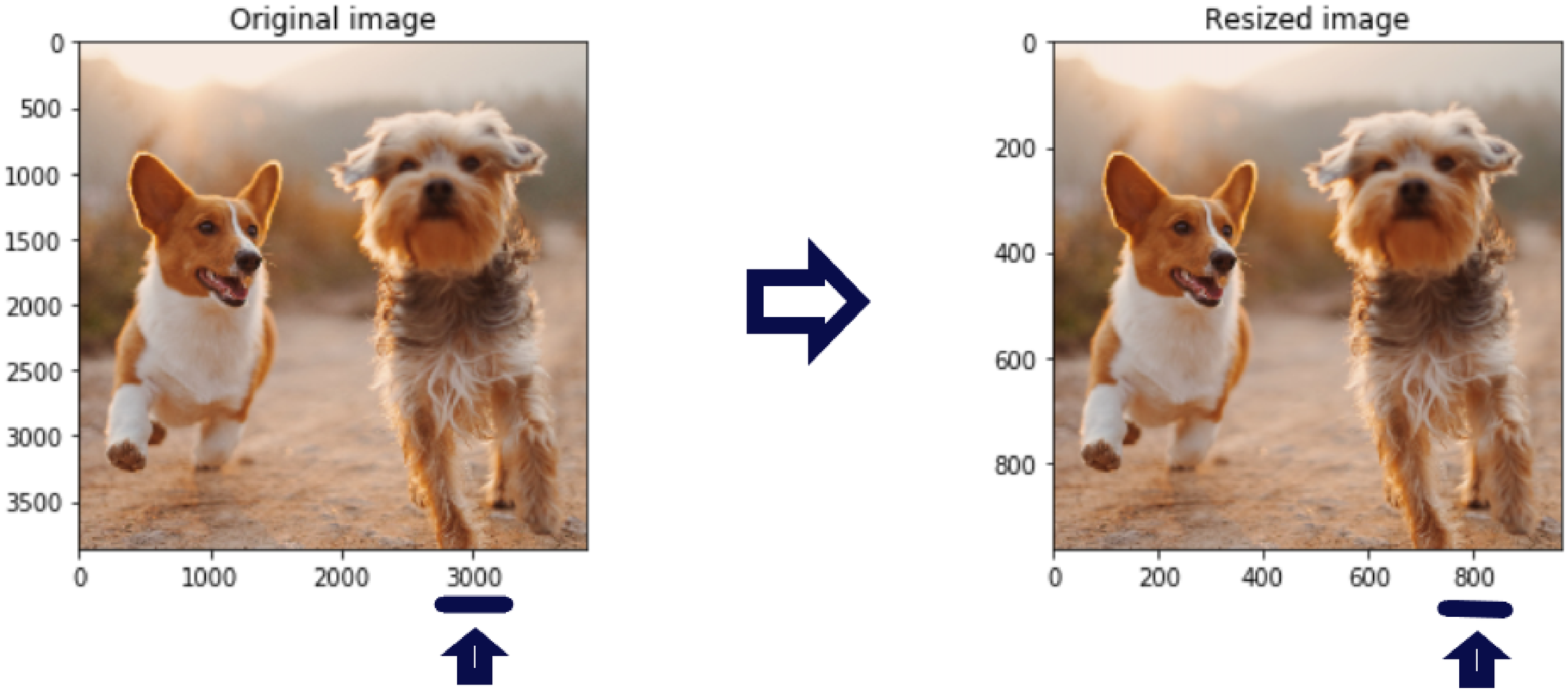


Aliased letter A

Aliasing in digital images



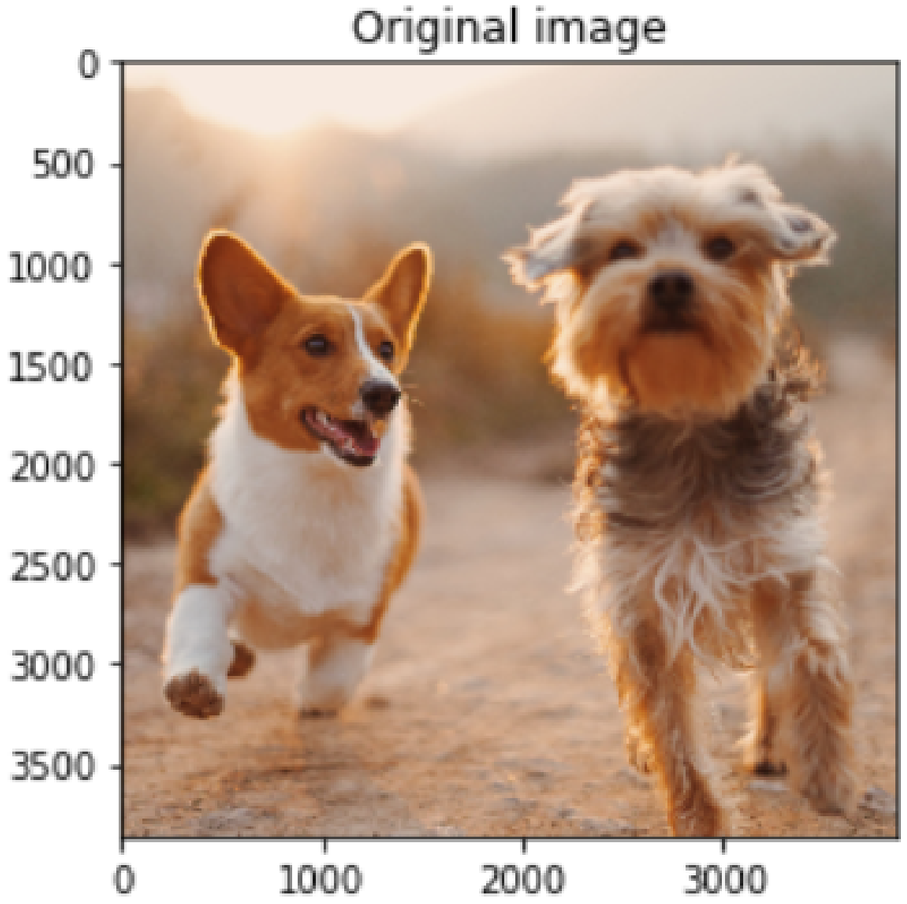
Resizing

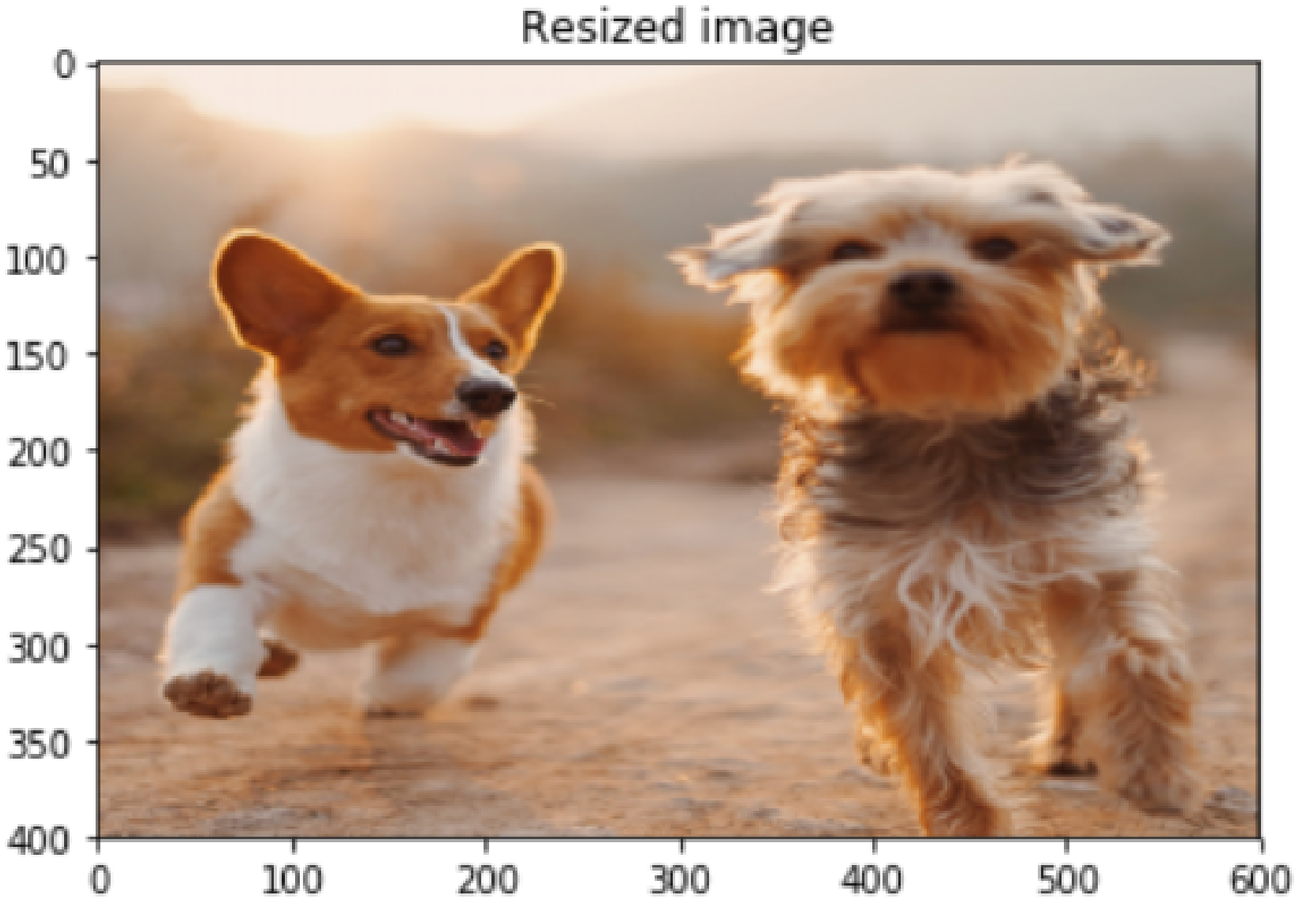

Original image
0
500
1000
1500
2000
2500
3000
3500
0
1000
2000
3000
Resized image
0
200
400
600
800
0
200
400
600
800

Resizing

```
from skimage.transform import resize

# Height and width to resize
height = 400
width = 500

# Resize image
image_resized = resize(image, (height, width), anti_aliasing=True)

# Show the original and resulting images
show_image(image, 'Original image')
show_image(image_resized, 'Resized image')
```

Resizing

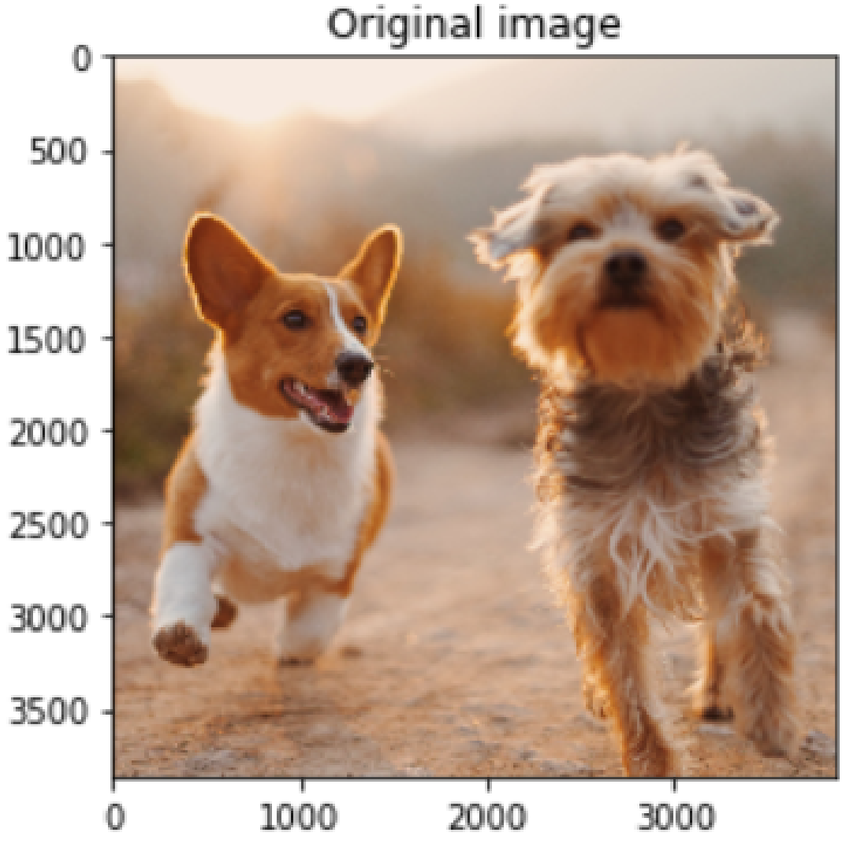

Original image
0
500
1000
1500
2000
2500
3000
3500
0
1000
2000
3000

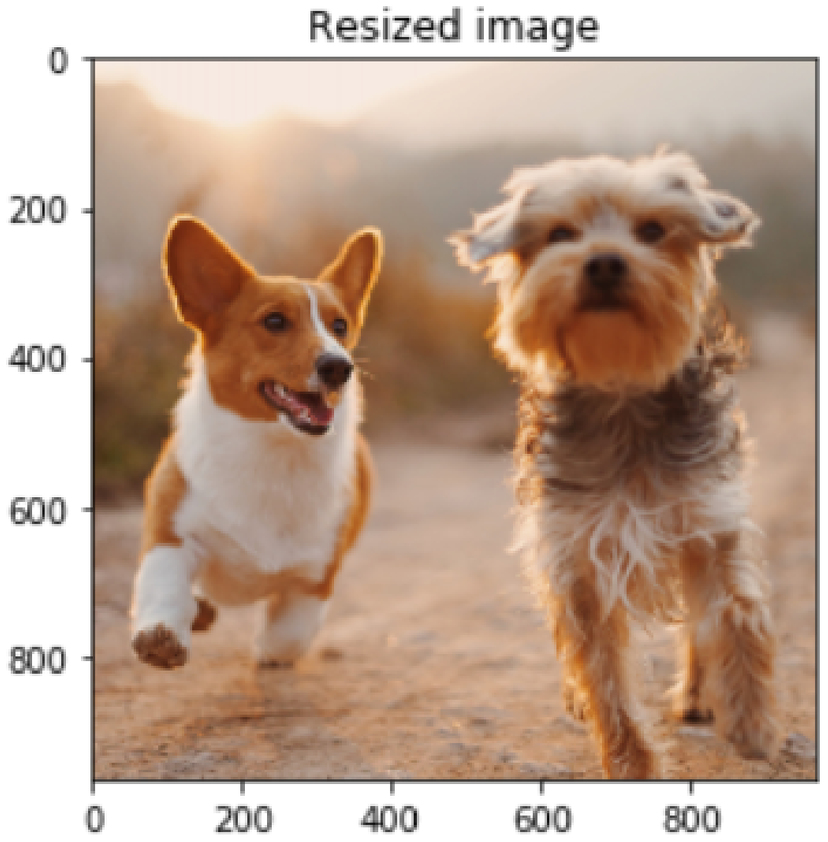

Resized image
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600

Resizing proportionally

```
from skimage.transform import resize

# Set proportional height so its 4 times its size
height = image.shape[0] / 4
width = image.shape[1] / 4

# Resize image
image_resized = resize(image, (height, width), anti_aliasing=True)

show_image(image_resized, 'Resized image')
```

Resizing proportionally





Let's practice!

IMAGE PROCESSING IN PYTHON

Morphology

IMAGE PROCESSING IN PYTHON

Rebeca Gonzalez
Data Engineer

datacamp

Binary images

Morphological filtering

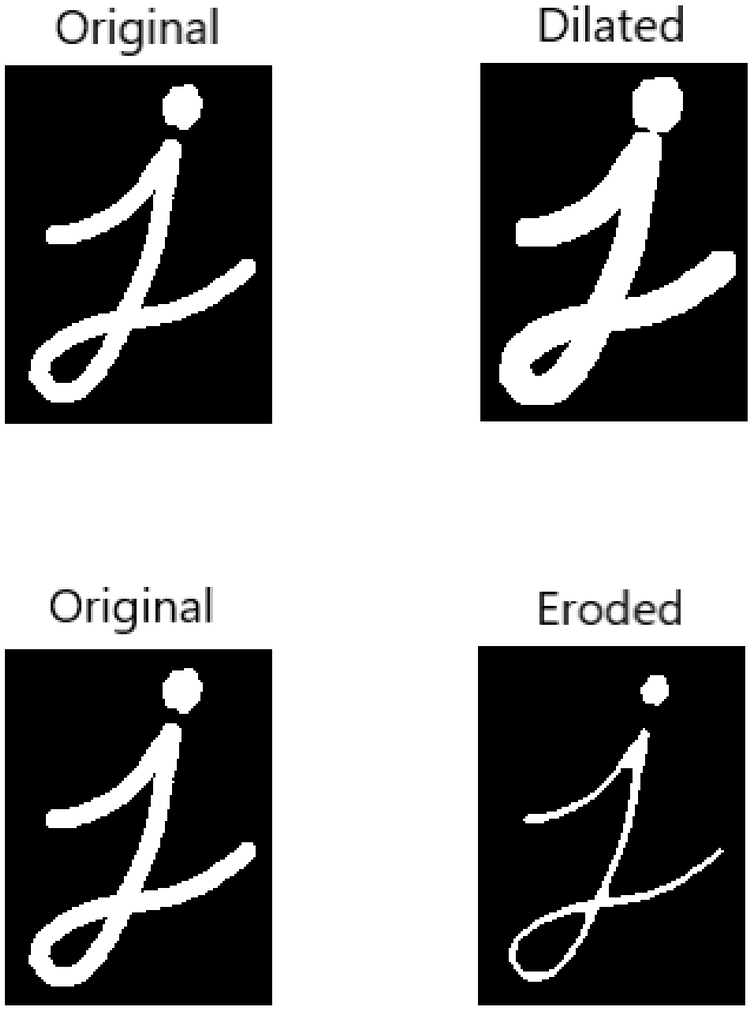
- Better for binary images
- Can extend for grayscale

Binary image

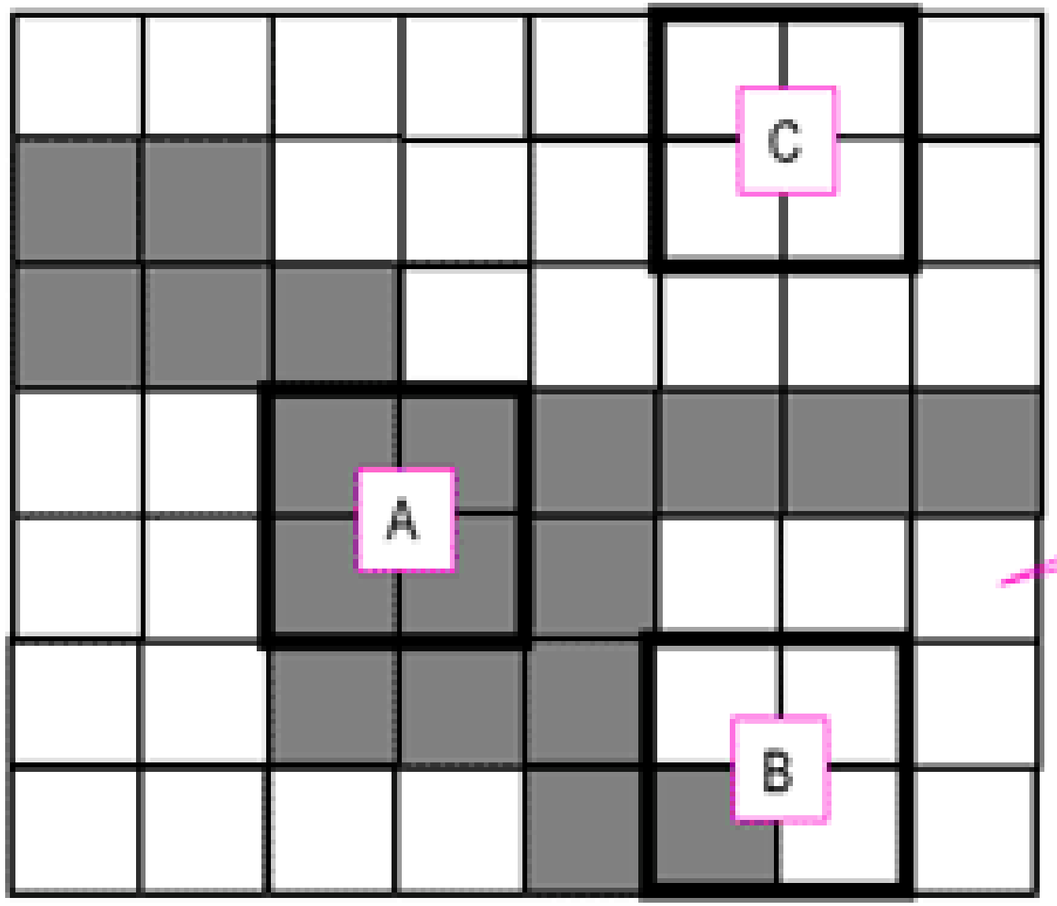
Grayscale

Morphological operations

- Dilation
- Erosion

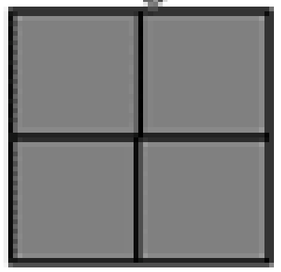


Structuring element

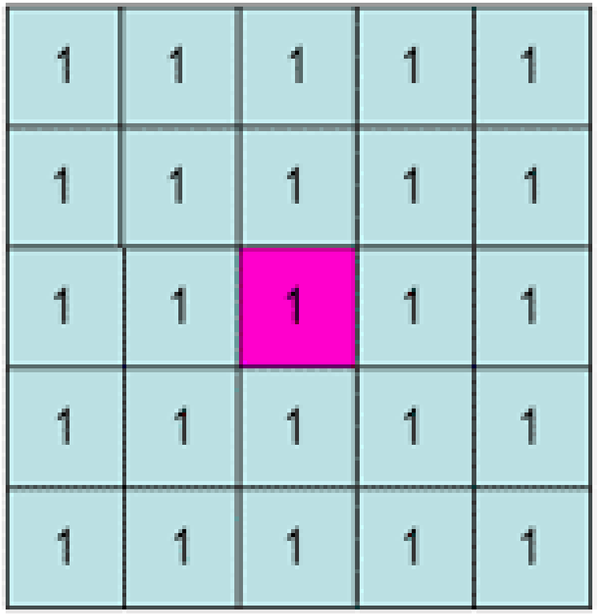


A - the structuring element fits the image
B - the structuring element hits (intersects) the image
C - the structuring element neither fits, nor hits the image

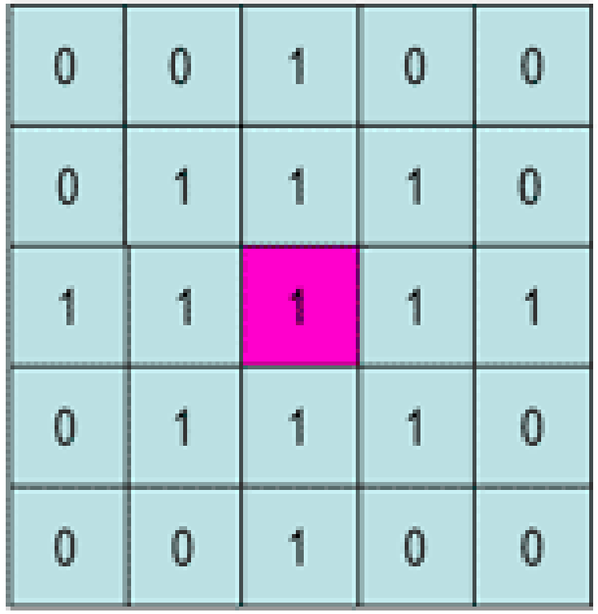
Structuring element



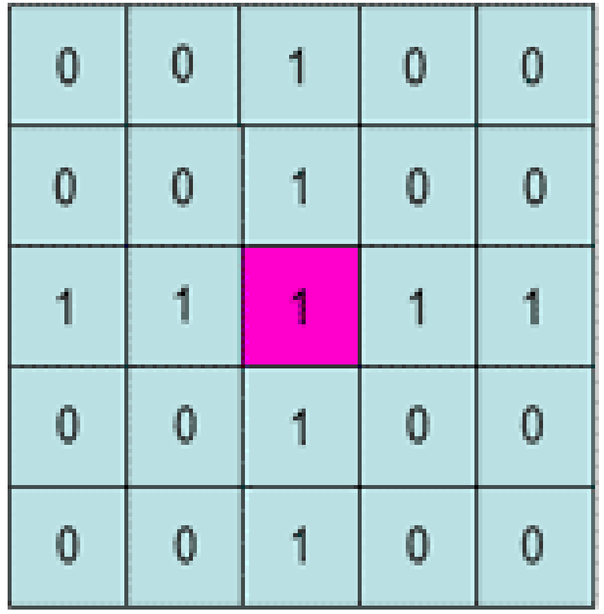
Structuring element



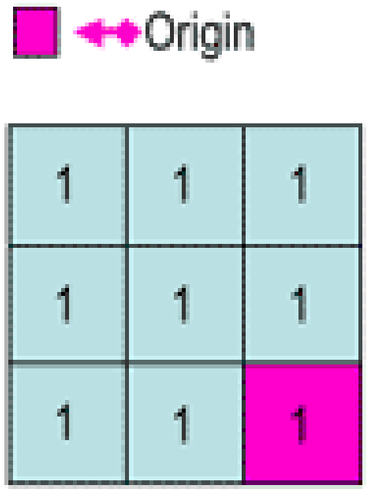
Square 5x5 element



Diamond-shaped 5x5 element



Cross-shaped 5x5 element



Square 3x3 element

Shapes in scikit-image

```
from skimage import morphology

square = morphology.square(4)
```

```
[[1 1 1 1]
 [1 1 1 1]
 [1 1 1 1]
 [1 1 1 1]]
```

```
rectangle = morphology.rectangle(4, 2)
```

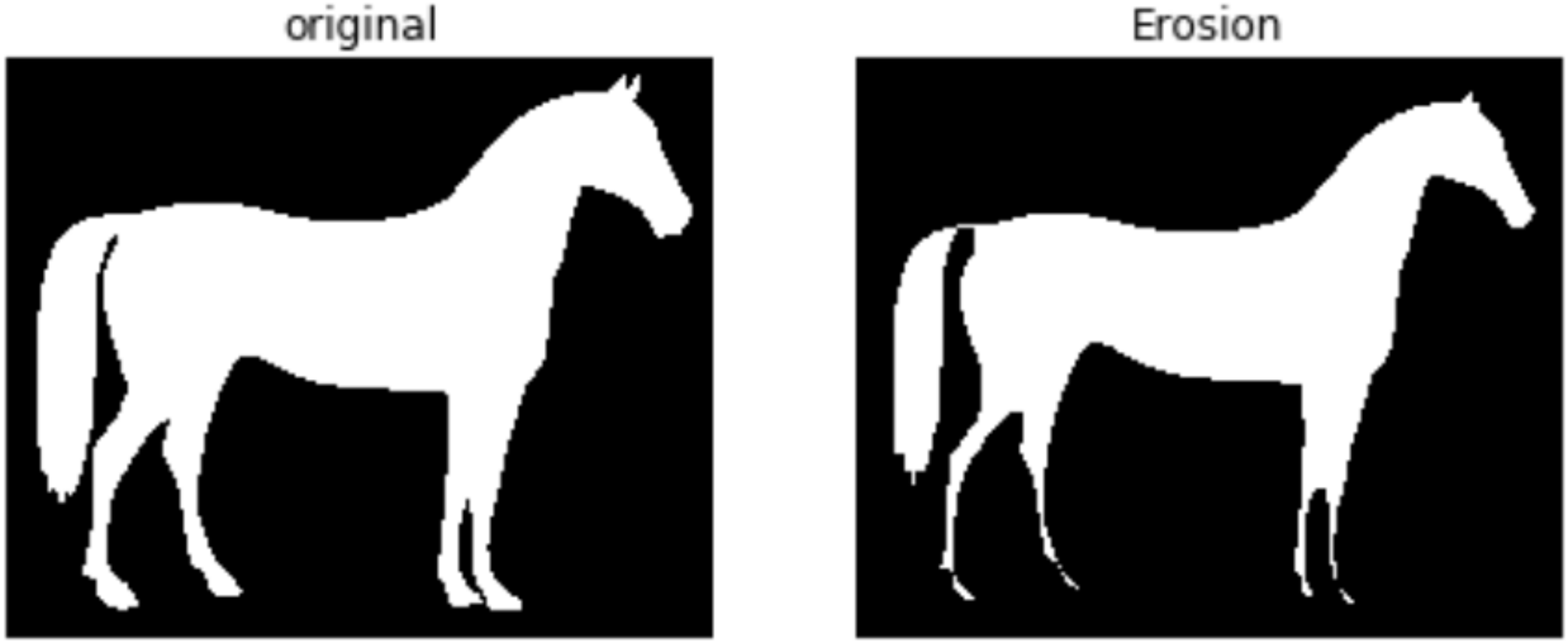
```
[[1 1]
 [1 1]
 [1 1]
 [1 1]]
```

Erosion in scikit-image

```
from skimage import morphology

# Set structuring element to the rectangular-shaped
selem = rectangle(12,6)

# Obtain the erosed image with binary erosion
eroded_image = morphology.binary_erosion(image_horse, selem=selem)
```

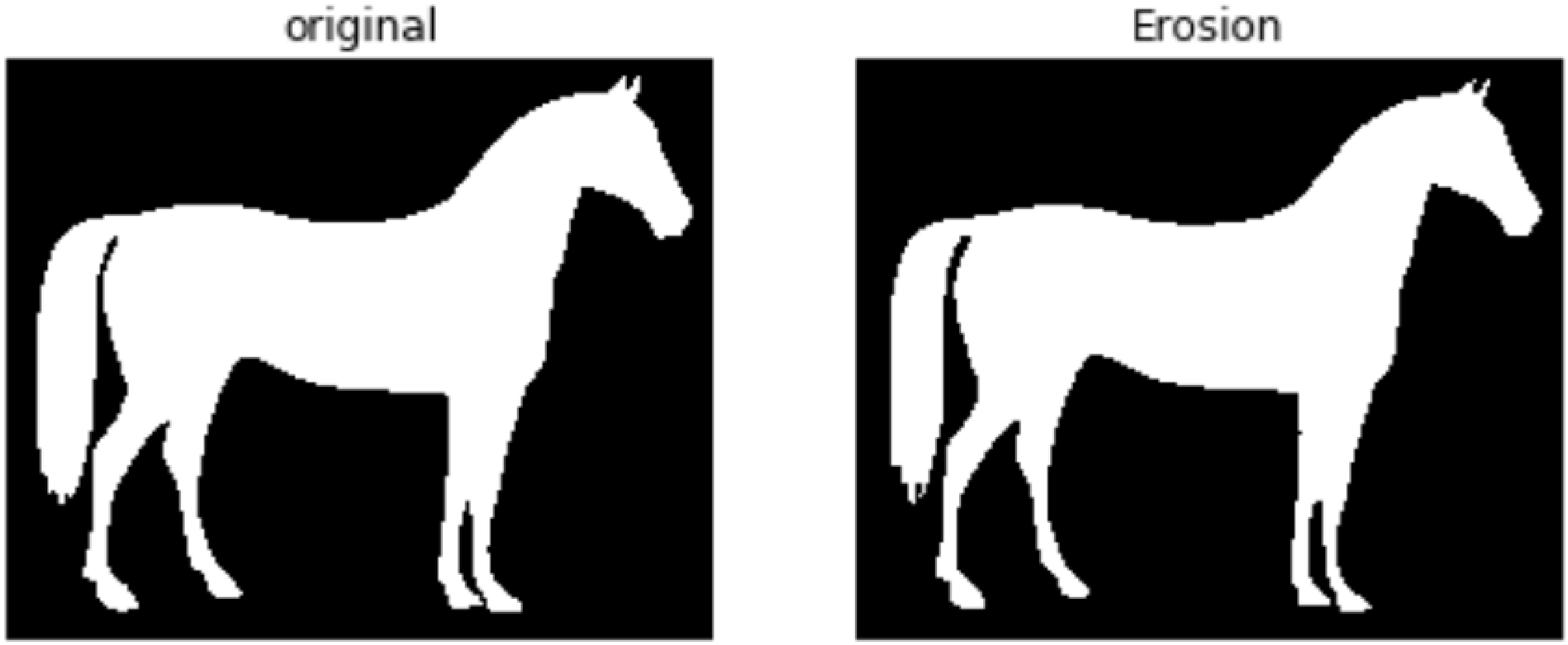
Erosion in scikit-image

```
# Show result
plot_comparison(image_horse, eroded_image, 'Erosion')
```



Binary erosion with default selem

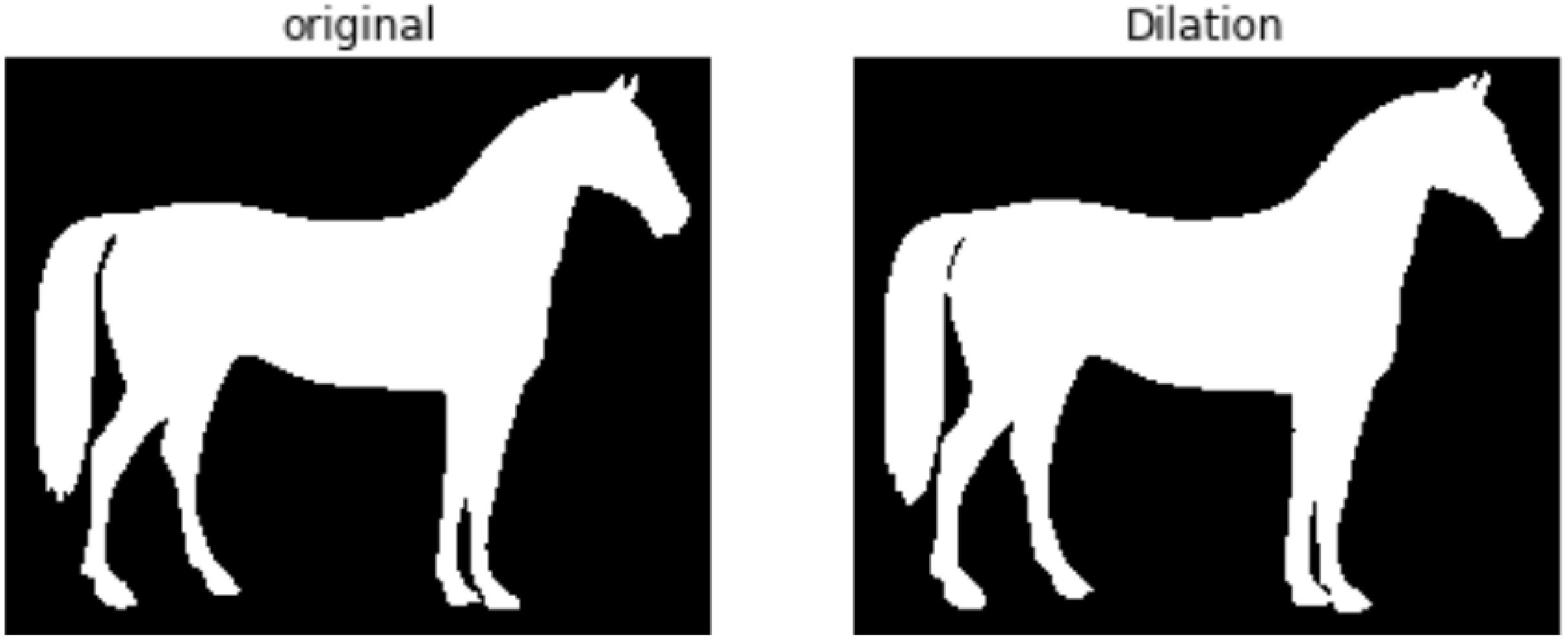
```
# Binary erosion with default selem
eroded_image = morphology.binary_erosion(image_horse)
```



Dilation in scikit-image

```
from skimage import morphology

# Obtain dilated image, using binary dilation
dilated_image = morphology.binary_dilation(image_horse)

# See results
plot_comparison(image_horse, dilated_image, 'Erosion')
```

Dilation in scikit-image



Let's practice!

IMAGE PROCESSING IN PYTHON