2.4. Morphological

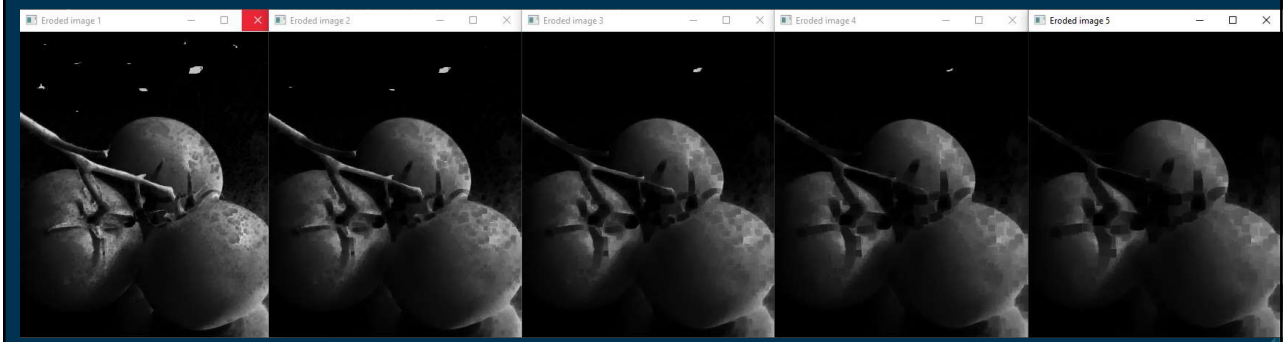
Example: remove the gray spots from the image

93

93

2.4. Morphological

Example: remove the gray spots from the image



94

94

2.4. Morphological

Solution:

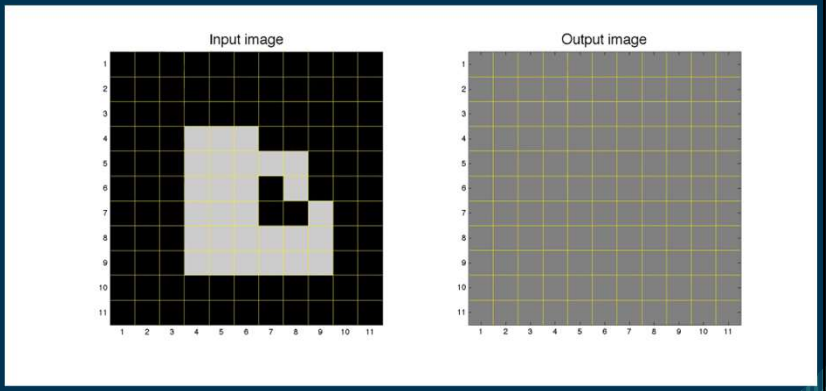
```
import cv2
import numpy as np

# Đọc ảnh nhiễu
img = cv2.imread('C:/Users/quang/OneDrive/Desktop/tomato_noise.jpg')

gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


# Lặp lại quá trình erosion với số lần lặp khác nhau
for i in range(0, 5):
    eroded_img = cv2.erode(gray_img.copy(), None, iterations=i + 1)
    cv2.imshow("Eroded image {}".format(i + 1), eroded_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

95

95

2.4. Morphological

Dilation is a morphological operation used to expand the white regions (foreground) in a binary or grayscale image and to shrink the black regions (background). It is typically used to fill small gaps, connect broken regions, and enhance features in the image.

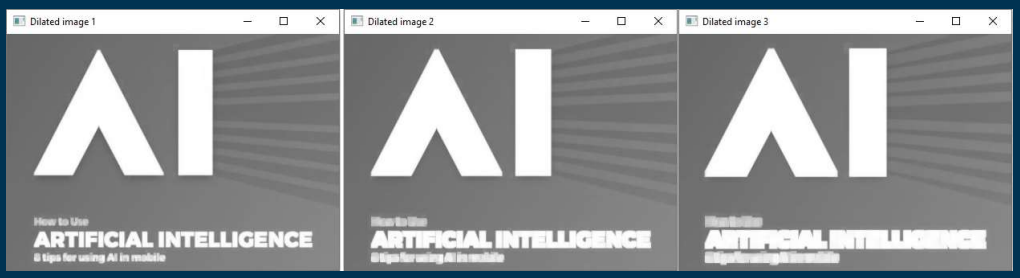


96

96

2.4. Morphological

- Dilation: cv2.dilate

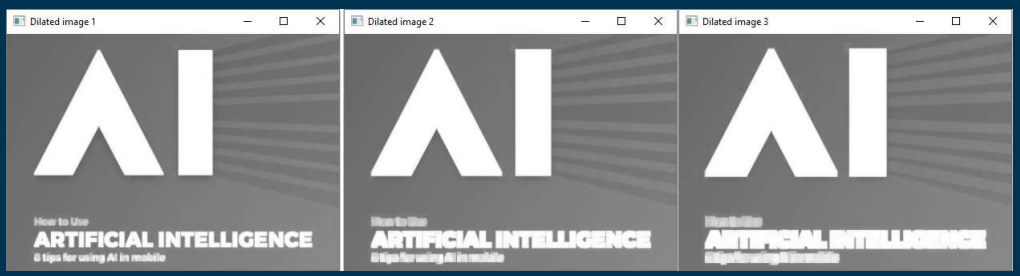


97

97

2.4. Morphological

- Dilation: cv2.dilate




```
for i in range(0, 3):
    dilated_img = cv2.dilate(gray_img.copy(), None, iterations=i + 1)
    cv2.imshow("Dilated image {} ".format(i + 1), dilated_img)
    cv2.waitKey(0)
```

98

98

2.4. Morphological

Exercise: Remove cracks/blemishes in the text "Mechatronics".

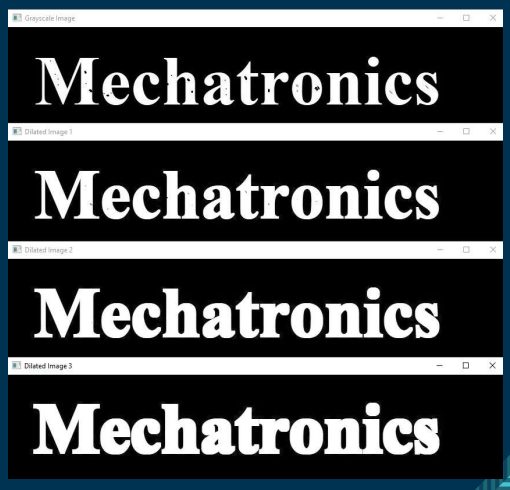
99

99

2.4. Morphological

Solution:

```
import cv2

img = cv2.imread("C:/Users/quang/OneDrive/Desktop/dilation.jpg")
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow("Grayscale Image", gray_img)

for i in range(0, 3):
    dilated_img = cv2.dilate(gray_img.copy(), None, iterations = i + 1)
    cv2.imshow("Dilated Image {}".format(i+1), dilated_img)
    cv2.waitKey(0)
```



100

100

2.4. Morphological

Opening - Closing

Opening is a combination of two operations: erosion followed by dilation. Opening is used to remove small objects or minor noise details in an image.

Closing is a combination of two operations: dilation followed by erosion. Closing is used to fill small gaps and connect broken regions in an image.

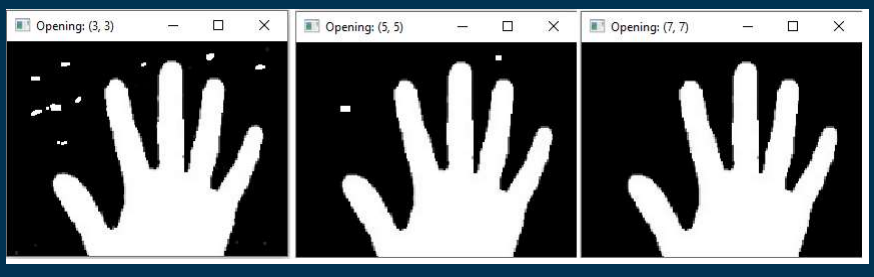
101

101

Opening

Syntax: opening = cv2.morphologyEx(gray_img, cv2.MORPH_OPEN, kernel)

Process: Erosion -> Dilation

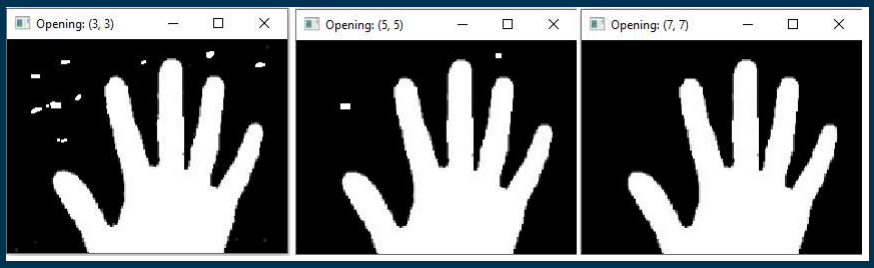


102

102

2.4. Morphological

- Opening: Erosion -> Dilation




```
kernelSizes = [(3, 3), (5, 5), (7, 7)]
for kernelSize in kernelSizes:
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, kernelSize)
    opening = cv2.morphologyEx(gray_img, cv2.MORPH_OPEN, kernel)
    cv2.imshow("Opening: ({}, {})".format(kernelSize[0], kernelSize[1]), opening)
    cv2.waitKey(0)
```

103

2.4. Morphological

```
kernelSizes = [(3, 3), (5, 5), (7, 7)]
for kernelSize in kernelSizes:
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, kernelSize)
    opening = cv2.morphologyEx(gray_img, cv2.MORPH_OPEN, kernel)
    cv2.imshow("Opening: ({}, {})".format(kernelSize[0], kernelSize[1]), opening)
    cv2.waitKey(0)
```

kernelSizes: A tuple representing the size of the kernel with dimensions (3, 3), (5, 5), and (7, 7).

kernel = cv2.getStructuringElement(cv2.MORPH_RECT, kernelSize) creates a rectangular kernel with the given size.

opening = cv2.morphologyEx(gray_img, cv2.MORPH_OPEN, kernel) applies the opening operation to the image.

104

Opening

Example: remove the gray spots from the image using opening

105

105

Closing

- Syntax: closing = cv2.morphologyEx(gray_img, cv2.MORPH_CLOSE, kernel)
- Process: Dilation -> Erosion

Closing is the combination of two operations: dilation followed by erosion. Closing is used to fill small holes and connect broken areas in an image.

106

Closing

- Closing: Dilation -> Erosion


```
for kernelSize in kernelSizes:
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, kernelSize)
    closing = cv2.morphologyEx(gray_img, cv2.MORPH_CLOSE, kernel)
    cv2.imshow("Closing: ({}, {})".format(kernelSize[0], kernelSize[1]), closing)
    cv2.waitKey(0)
```

107

107

Closing

Example: remove the gray spots from the image using Closing

108

108

Closing

Results:

109

109

Closing

Code:

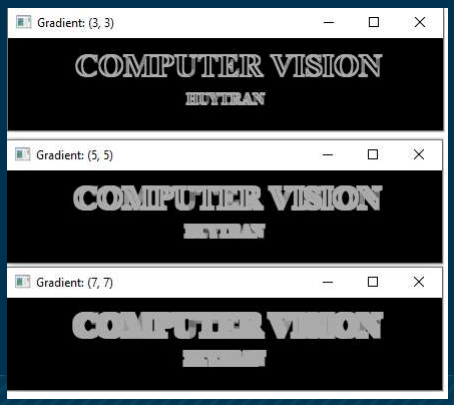
```
import cv2

img = cv2.imread("C:/Users/quang/OneDrive/Desktop/closing.jpg")
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow("Grayscale Image", gray_img)
kernelSizes = [(3, 3), (5, 5), (7, 7)]
for kernelSize in kernelSizes:
    kernel  = cv2.getStructuringElement(cv2.MORPH_RECT, kernelSize)
    closing = cv2.morphologyEx(gray_img, cv2.MORPH_CLOSE, kernel)
    cv2.imshow("closing: {} {}".format(kernelSize[0], kernelSize[1]), closing)
    cv2.waitKey(0)
```

110

110

Morphological gradient

- Morphological gradient: gradient = cv2.morphologyEx(gray_img, cv2.MORPH_GRADIENT, kernel)



111

111

Morphological gradient

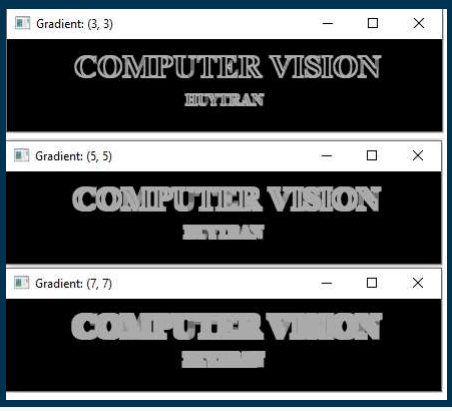
- Morphological gradient is an image processing technique used to highlight the edges or boundaries in an image.
- This technique is computed by taking the difference between the image after dilation and the image after erosion.
- Syntax: gradient = cv2.morphologyEx(gray_img, cv2.MORPH_GRADIENT, kernel)

112

112

Morphological gradient

Example:



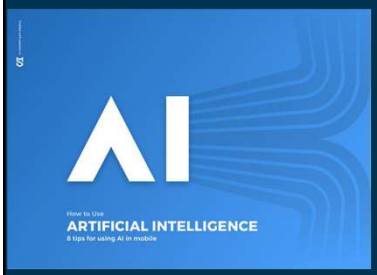

```
for kernelSize in kernelSizes:
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, kernelSize)
    gradient = cv2.morphologyEx(gray_img, cv2.MORPH_GRADIENT, kernel)
    cv2.imshow("Gradient: ({}, {})".format(kernelSize[0], kernelSize[1]), gradient)
    cv2.waitKey(0)
```

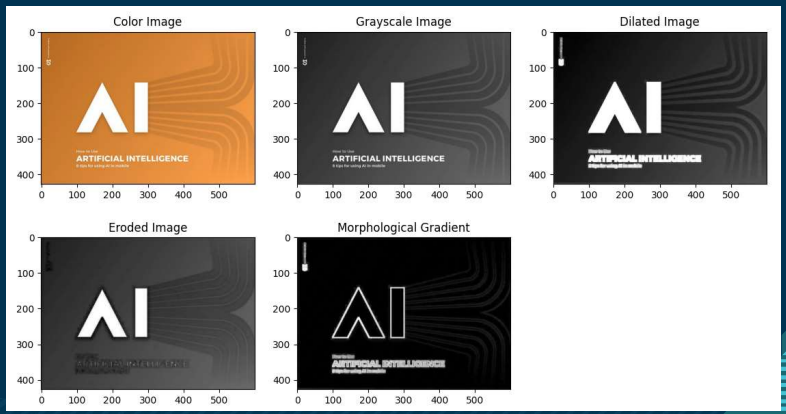
113

113

Morphological gradient

Example: apply Morphological gradient and Matplotlib to achieve the following result





114

114

Morphological gradient

- Apply matplotlib:

  import matplotlib.pyplot as plt

- Create new blank figure with dimensions of 20 x 12 inches (width x height)

  plt.figure(figsize=(20, 12))

- Make a frame with 2 rows and 3 columns. The selected image is in the first position.

  plt.subplot(2, 3, 1)

- Set title

  plt.title('Color Image')

- Display images

  plt.imshow(img, cmap='gray')

115

115

Morphological gradient

```
import cv2
import matplotlib.pyplot as plt

# Load image
img = cv2.imread("C:/Users/quang/Downloads/AI.png")
image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Create structuring element
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))

# Apply dilation and erosion
dilated = cv2.dilate(image, kernel)
eroded = cv2.erode(image, kernel)

# Apply morphological gradient
gradient = cv2.morphologyEx(image, cv2.MORPH_GRADIENT, kernel)
```

116

116

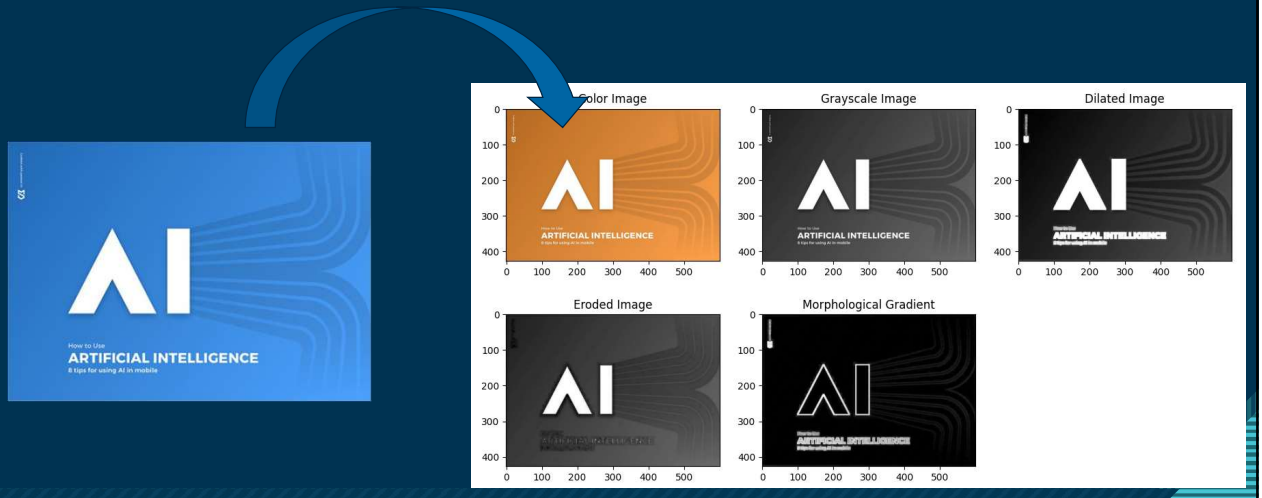
Morphological gradient

```
# Display images
plt.figure(figsize=(20, 12))

plt.subplot(2, 3, 1)
plt.title('Color Image')
plt.imshow(img, cmap='gray')

plt.subplot(2, 3, 2)
plt.title('Grayscale Image')
plt.imshow(image, cmap='gray')

plt.subplot(2, 3, 3)
plt.title('Dilated Image')
plt.imshow(dilated, cmap='gray')

plt.subplot(2, 3, 4)
plt.title('Eroded Image')
plt.imshow(eroded, cmap='gray')

plt.subplot(2, 3, 5)
plt.title('Morphological Gradient')
plt.imshow(gradient, cmap='gray')

plt.show()
```


117

Morphological gradient

Question: why does the original image change its color?

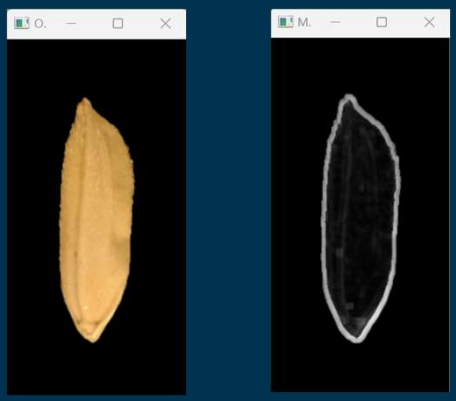


Đáp án: ***img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)***

118

Morphological gradient

Example 1:



119

119

Morphological gradient

Example: apply Morphological gradient

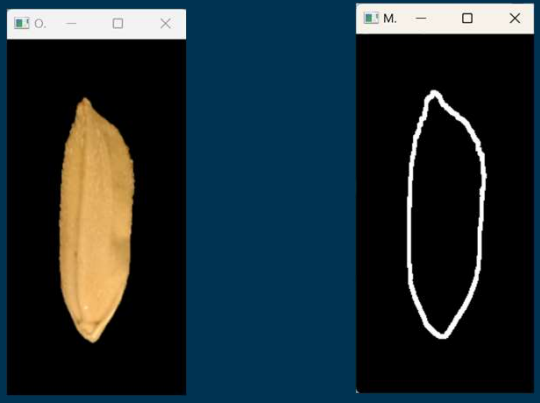
```
import cv2
import numpy as np
# Read the image
img = cv2.imread("C:/Users/quang/OneDrive/Desktop/photo/leaf.PNG")
# Convert the image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Create a kernel for morphological operations
kernel = np.ones( shape: (5,5), np.uint8)
# Apply the Morphological Gradient operation (calculating the object's edges)
gradient = cv2.morphologyEx(gray, cv2.MORPH_GRADIENT, kernel)
# Display the result
cv2.imshow( winname: 'Original Image', img)
cv2.imshow( winname: 'Morphological Gradient', gradient)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

120

120

Morphological gradient

Example 2:



121

121

Morphological gradient

Solution: apply Morphological gradient and thresholding

```
import cv2
import numpy as np
# Read the image
img = cv2.imread("C:/Users/quang/OneDrive/Desktop/photo/leaf.PNG")
# Convert the image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Apply binary thresholding to highlight the leaf
_, thresh = cv2.threshold(gray, thresh: 128, maxval: 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
# Create a kernel for morphological operations
kernel = np.ones( shape: (5,5), np.uint8)
# Apply the Morphological Gradient operation (calculating the object's edges)
gradient = cv2.morphologyEx(thresh, cv2.MORPH_GRADIENT, kernel)
# Display the result
cv2.imshow( winname: 'Original Image', img)
cv2.imshow( winname: 'Morphological Gradient', gradient)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

122

122