**Code:**

import cv2

import numpy as np

# Load the grayscale image

image = cv2.imread(r"D:\DSIP\ex1\_1.png", cv2.IMREAD\_GRAYSCALE)

if image is None:

print("Error: Could not open or find the image.")

exit()

# Image operations

def image\_negation(input\_image):

return 255 - input\_image

def image\_thresholding(input\_image, threshold\_value):

\_, thresholded\_image = cv2.threshold(input\_image, threshold\_value, 255, cv2.THRESH\_BINARY)

return thresholded\_image

def image\_gamma\_correction(input\_image, gamma):

gamma\_corrected = np.power(input\_image / 255.0, gamma) \* 255.0

return np.uint8(gamma\_corrected)

# Resize helper to match sizes

def resize\_to\_common\_size(images, width=300, height=300):

resized = []

for img in images:

resized.append(cv2.resize(img, (width, height), interpolation=cv2.INTER\_LINEAR))

return resized

# Perform operations

negated\_image = image\_negation(image)

thresholded\_image = image\_thresholding(image, 102)

gamma\_corrected\_image = image\_gamma\_correction(image, -0.5)

# Resize all images to same size

resized\_images = resize\_to\_common\_size([

image,

negated\_image,

thresholded\_image,

gamma\_corrected\_image

])

# Add labels (optional)

def add\_label(image, text):

return cv2.putText(image.copy(), text, (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,

0.8, (255), 2, cv2.LINE\_AA)

labeled\_images = [

add\_label(resized\_images[0], "Original"),

add\_label(resized\_images[1], "Negated"),

add\_label(resized\_images[2], "Thresholded"),

add\_label(resized\_images[3], "Gamma")

]

# Stack images horizontally

combined\_image = cv2.hconcat(labeled\_images)

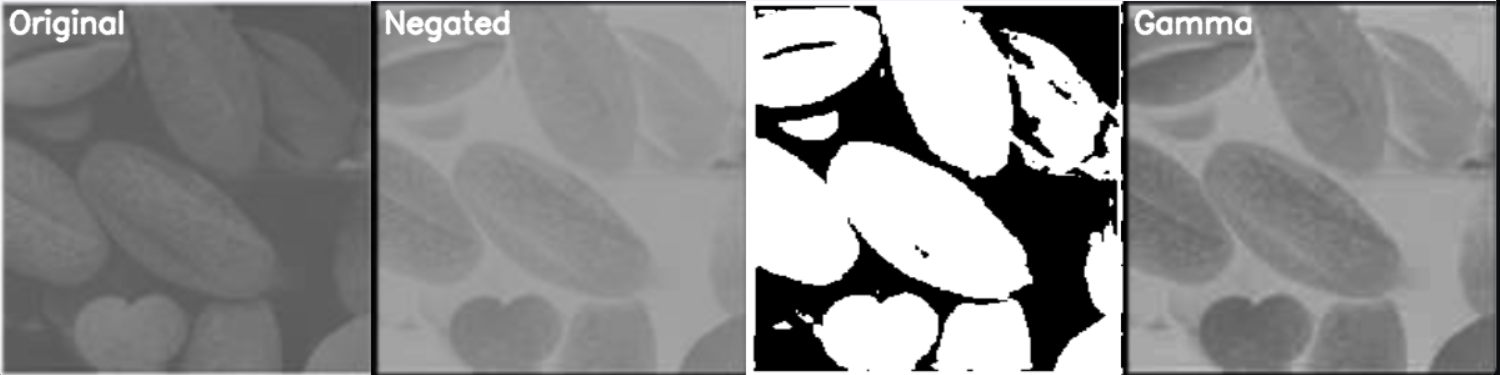
# Show the combined image

cv2.imshow("Image Processing Results", combined\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****

**CODE:**

import cv2

import numpy as np

*# Load the grayscale image*

image = cv2.imread(r"D:\DSIP\ex1\_2.png", cv2.IMREAD\_GRAYSCALE)

if image is None:

    print("Error: Could not open or find the image.")

    exit()

*# Image operations*

def image\_negation(input\_image):

    return 255 - input\_image

def image\_thresholding(input\_image, threshold\_value):

    \_, thresholded\_image = cv2.threshold(input\_image, threshold\_value, 255, cv2.THRESH\_BINARY)

    return thresholded\_image

def image\_gamma\_correction(input\_image, gamma):

    gamma\_corrected = np.power(input\_image / 255.0, gamma) \* 255.0

    return np.uint8(gamma\_corrected)

*# Resize helper to match sizes*

def resize\_to\_common\_size(images, width=300, height=300):

    resized = []

    for img in images:

        resized.append(cv2.resize(img, (width, height), interpolation=cv2.INTER\_LINEAR))

    return resized

*# Perform operations*

negated\_image = image\_negation(image)

thresholded\_image = image\_thresholding(image, 118)

gamma\_corrected\_image = image\_gamma\_correction(image, 1)

*# Resize all images to same size*

resized\_images = resize\_to\_common\_size([

    image,

    negated\_image,

    thresholded\_image,

    gamma\_corrected\_image

])

*# Add labels (optional)*

def add\_label(image, text):

    return cv2.putText(image.copy(), text, (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,

                       0.8, (255), 2, cv2.LINE\_AA)

labeled\_images = [

    add\_label(resized\_images[0], "Original"),

    add\_label(resized\_images[1], "Negated"),

    add\_label(resized\_images[2], "Thresholded"),

    add\_label(resized\_images[3], "Gamma")

]

*# Stack images horizontally*

combined\_image = cv2.hconcat(labeled\_images)

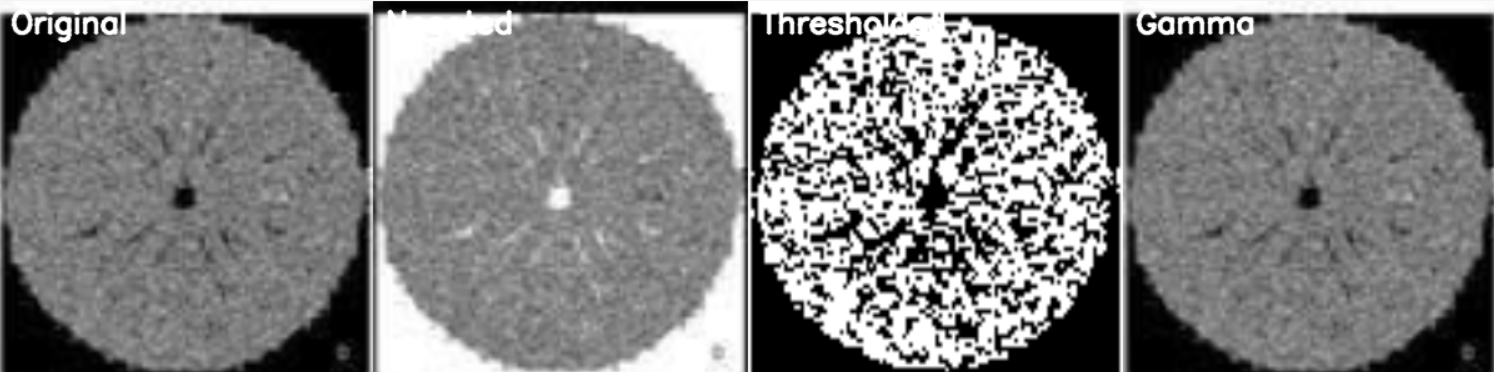
*# Show the combined image*

cv2.imshow("Image Processing Results", combined\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()\

**OUTPUT:**

****

**CODE:**

import cv2

import numpy as np

*# Load the grayscale image*

image = cv2.imread(r"D:\DSIP\ex1\_3.png",cv2.IMREAD\_GRAYSCALE)

if image is None:

    print("Error: Could not open or find the image.")

    exit()

*# Image operations*

def image\_negation(input\_image):

    return 255 - input\_image

def image\_thresholding(input\_image, threshold\_value):

    \_, thresholded\_image = cv2.threshold(input\_image, threshold\_value, 255, cv2.THRESH\_BINARY)

    return thresholded\_image

def image\_gamma\_correction(input\_image, gamma):

    gamma\_corrected = np.power(input\_image / 255.0, gamma) \* 255.0

    return np.uint8(gamma\_corrected)

*# Resize helper to match sizes*

def resize\_to\_common\_size(images, width=300, height=300):

    resized = []

    for img in images:

        resized.append(cv2.resize(img, (width, height), interpolation=cv2.INTER\_LINEAR))

    return resized

*# Perform operations*

negated\_image = image\_negation(image)

thresholded\_image = image\_thresholding(image, 25)

gamma\_corrected\_image = image\_gamma\_correction(image, 0.8)

*# Resize all images to same size*

resized\_images = resize\_to\_common\_size([

    image,

    negated\_image,

    thresholded\_image,

    gamma\_corrected\_image

])

*# Add labels (optional)*

def add\_label(image, text):

    return cv2.putText(image.copy(), text, (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,

                       0.8, (255), 2, cv2.LINE\_AA)

labeled\_images = [

    add\_label(resized\_images[0], "Original"),

    add\_label(resized\_images[1], "Negated"),

    add\_label(resized\_images[2], "Thresholded"),

    add\_label(resized\_images[3], "Gamma")

]

*# Stack images horizontally*

combined\_image = cv2.hconcat(labeled\_images)

*# Show the combined image*

cv2.imshow("Image Processing Results", combined\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****

**CODE:**

import cv2

import numpy as np

*# Load the grayscale image*

image = cv2.imread(r"D:\DSIP\ex1\_4.png", cv2.IMREAD\_GRAYSCALE)

if image is None:

    print("Error: Could not open or find the image.")

    exit()

*# Image operations*

def image\_negation(input\_image):

    return 255 - input\_image

def image\_thresholding(input\_image, threshold\_value):

    \_, thresholded\_image = cv2.threshold(input\_image, threshold\_value, 255, cv2.THRESH\_BINARY)

    return thresholded\_image

def image\_gamma\_correction(input\_image, gamma):

    gamma\_corrected = np.power(input\_image / 255.0, gamma) \* 255.0

    return np.uint8(gamma\_corrected)

*# Resize helper to match sizes*

def resize\_to\_common\_size(images, width=300, height=300):

    resized = []

    for img in images:

        resized.append(cv2.resize(img, (width, height), interpolation=cv2.INTER\_LINEAR))

    return resized

*# Perform operations*

negated\_image = image\_negation(image)

thresholded\_image = image\_thresholding(image, 1.2)

gamma\_corrected\_image = image\_gamma\_correction(image, -1.15)

*# Resize all images to same size*

resized\_images = resize\_to\_common\_size([

    image,

    negated\_image,

    thresholded\_image,

    gamma\_corrected\_image

])

*# Add labels (optional)*

def add\_label(image, text):

    return cv2.putText(image.copy(), text, (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,

                       0.8, (255), 2, cv2.LINE\_AA)

labeled\_images = [

    add\_label(resized\_images[0], "Original"),

    add\_label(resized\_images[1], "Negated"),

    add\_label(resized\_images[2], "Thresholded"),

    add\_label(resized\_images[3], "Gamma")

]

*# Stack images horizontally*

combined\_image = cv2.hconcat(labeled\_images)

*# Show the combined image*

cv2.imshow("Image Processing Results", combined\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****

**CODE:**

import cv2

import numpy as np

*# Load the grayscale image*

image = cv2.imread(r"D:\DSIP\ex1\_5.png", cv2.IMREAD\_GRAYSCALE)

if image is None:

    print("Error: Could not open or find the image.")

    exit()

*# Image operations*

def image\_negation(input\_image):

    return 255 - input\_image

def image\_thresholding(input\_image, threshold\_value):

    \_, thresholded\_image = cv2.threshold(input\_image, threshold\_value, 255, cv2.THRESH\_BINARY)

    return thresholded\_image

def image\_gamma\_correction(input\_image, gamma):

    gamma\_corrected = np.power(input\_image / 255.0, gamma) \* 255.0

    return np.uint8(gamma\_corrected)

*# Resize helper to match sizes*

def resize\_to\_common\_size(images, width=300, height=300):

    resized = []

    for img in images:

        resized.append(cv2.resize(img, (width, height), interpolation=cv2.INTER\_LINEAR))

    return resized

*# Perform operations*

negated\_image = image\_negation(image)

thresholded\_image = image\_thresholding(image, 2)

gamma\_corrected\_image = image\_gamma\_correction(image, -1)

*# Resize all images to same size*

resized\_images = resize\_to\_common\_size([

    image,

    negated\_image,

    thresholded\_image,

    gamma\_corrected\_image

])

*# Add labels (optional)*

def add\_label(image, text):

    return cv2.putText(image.copy(), text, (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,

                       0.8, (255), 2, cv2.LINE\_AA)

labeled\_images = [

    add\_label(resized\_images[0], "Original"),

    add\_label(resized\_images[1], "Negated"),

    add\_label(resized\_images[2], "Thresholded"),

    add\_label(resized\_images[3], "Gamma")

]

*# Stack images horizontally*

combined\_image = cv2.hconcat(labeled\_images)

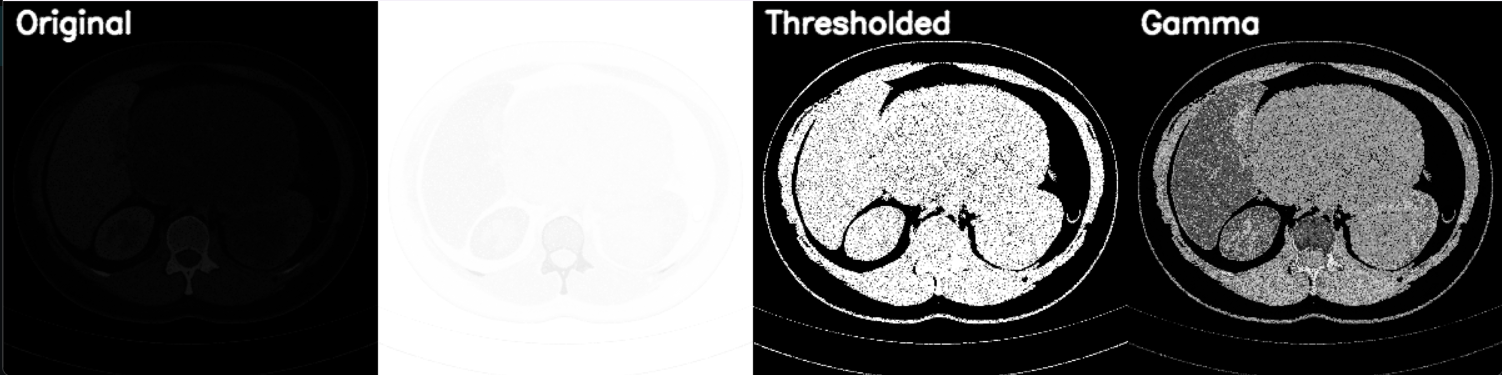
*# Show the combined image*

cv2.imshow("Image Processing Results", combined\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****