МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Вычислительной техники

ОТЧЁТ

по лабораторной работе № 1

«Начало работы с библиотекой OpenCV»

по дисциплине: «Математические методы распознавания образов»

Выполнили:

Студенты гр. АПИМ-24, АВТФ:

Чернощеков Владимир Сергеевич

Преподаватель:

Ильиных Сергей Петрович

Новосибирск, 2025

Оглавление

Задание:

1. Установка Python
2. Установка библиотек OpenCV-Python, Numpy и Matplotlib
3. Тестирование функционала библиотеки OpenCV

Ход работы:

Используемое изображение:

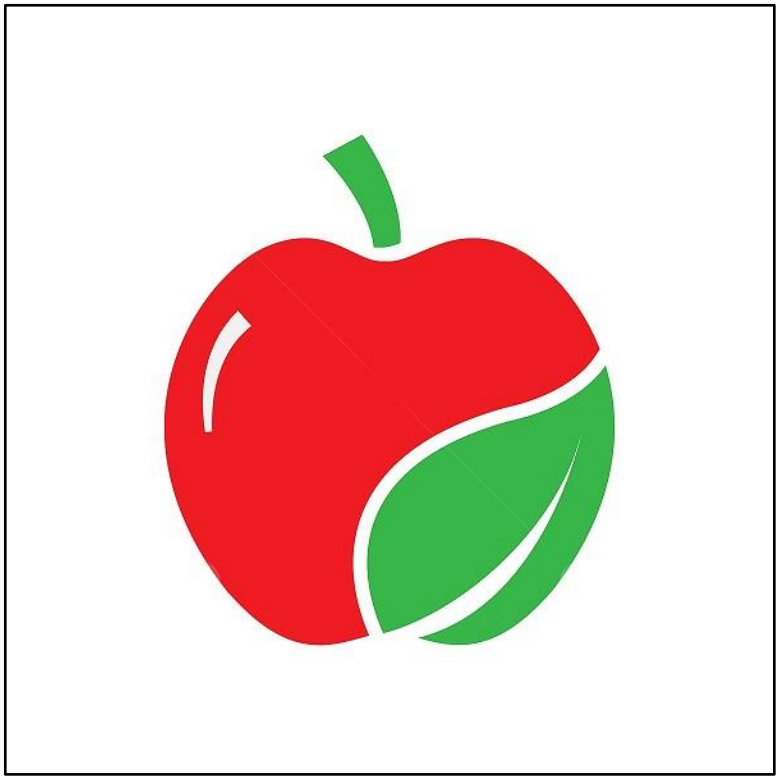

Рисунок 2 – Исходное изображение

Чтение и отображение изображения

Код программы:

```
# Python code to read image
import cv2

img = cv2.imread("AppleImage.jpg", cv2.IMREAD_COLOR)

cv2.imshow("AppleImage", img)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Результат:

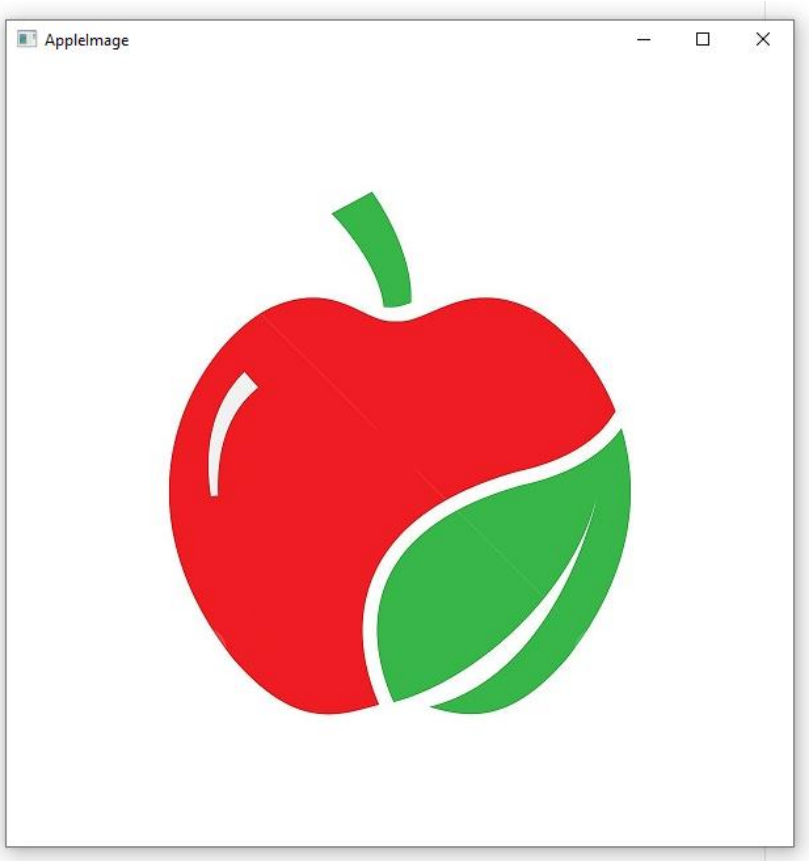

Рисунок 2 – Чтение и отображение

Сохранение изображений

Код программы:

```
import cv2

img = cv2.imread("AppleImage.jpg", cv2.IMREAD_COLOR)

# Filename
filename = 'savedImage.jpg'

cv2.imwrite(filename, img)
img = cv2.imread(filename)
cv2.imshow("SaveAppleImage", img)

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

Результат:

Рисунок 2 – Чтение и отображение

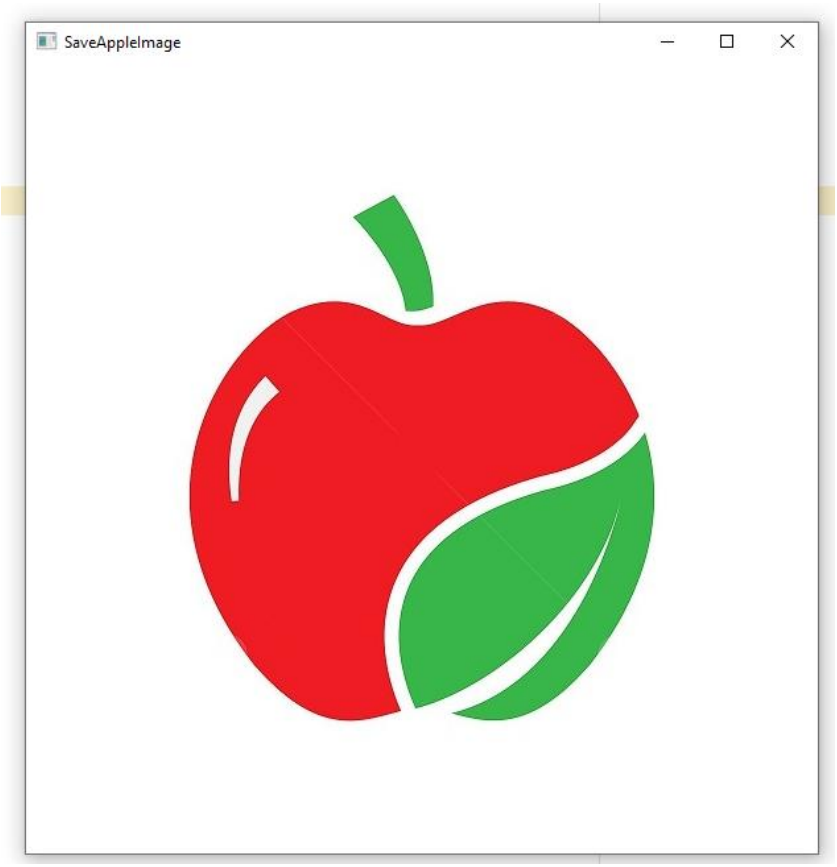

Рисунок 3 – Сохраненное изображение

Вращение изображений

Код программы:

```
# importing cv2
import cv2

# path
path = 'AppleImage.jpg'

# Reading an image in default mode
src = cv2.imread(path)

# Window name in which image is displayed
window_name = 'Image'

image = cv2.rotate(src, cv2.ROTATE_180)

# Displaying the image
cv2.imshow(window_name, image)
cv2.waitKey(0)
```

Результат:

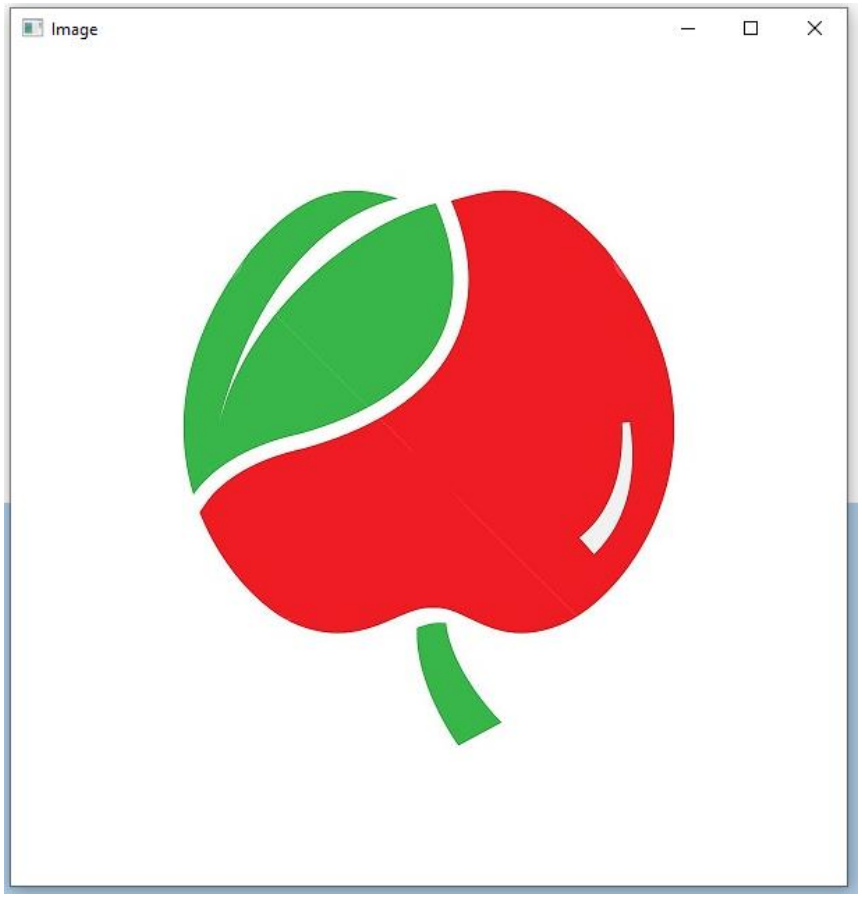

Рисунок 4 – Поворот на 180 градусов

Вращение на 45 градусов

Код программы:

```
import cv2
import numpy as np

FILE_NAME = 'AppleImage.jpg'

# Read image from the disk.
img = cv2.imread(FILE_NAME)

# Shape of image in terms of pixels.
(rows, cols) = img.shape[:2]

M = cv2.getRotationMatrix2D((cols / 2, rows / 2), 45, 1)
res = cv2.warpAffine(img, M, (cols, rows))

cv2.imshow("AppleImage", res)

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

Результат:

Рисунок 5 – Поворот на 45 градусов

Изменение размера изображения

Код программы:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread("AppleImage.jpg", 1)
# Loading the image

half = cv2.resize(image, (0, 0), fx = 0.1, fy = 0.1)
bigger = cv2.resize(image, (1050, 1610))

stretch_near = cv2.resize(image, (780, 540),
 interpolation = cv2.INTER_NEAREST)


Titles =["Original", "Half", "Bigger", "Interpolation Nearest"]
images =[image, half, bigger, stretch_near]
count = 4

for i in range(count):
 plt.subplot(2, 3, i + 1)
 plt.title(Titles[i])
 plt.imshow(images[i])

plt.show()
```

Результат:

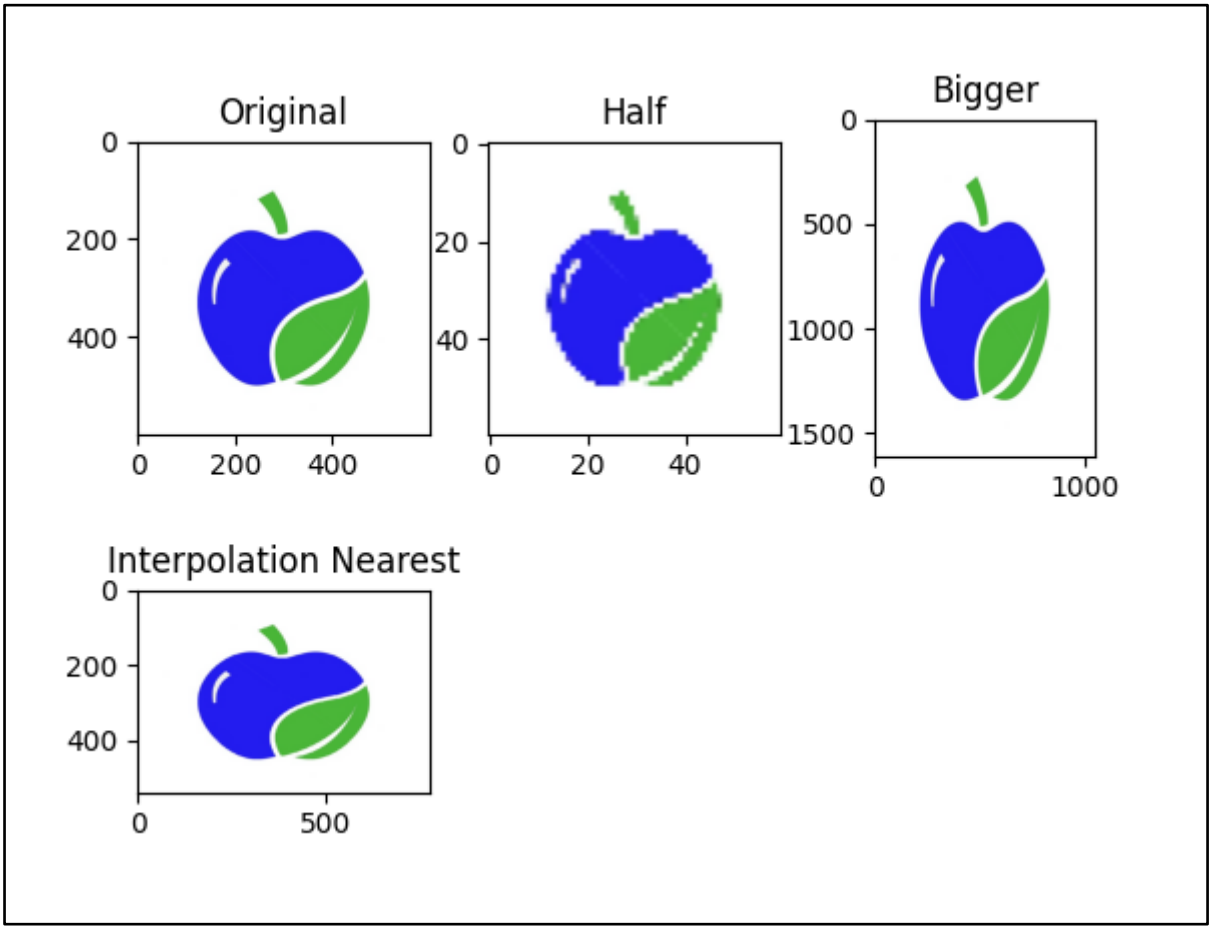

Рисунок 6 – Изменение размеров изображения

Цветовые пространства Python OpenCV. ЧБ изображение.

Код программы:

```
# importing cv2
import cv2

# path
path = 'AppleImage.jpg'

# Reading an image in default mode
src = cv2.imread(path)

window_name = 'AppleImage'

image = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY )

# Displaying the image
cv2.imshow(window_name, image)

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

Результат:

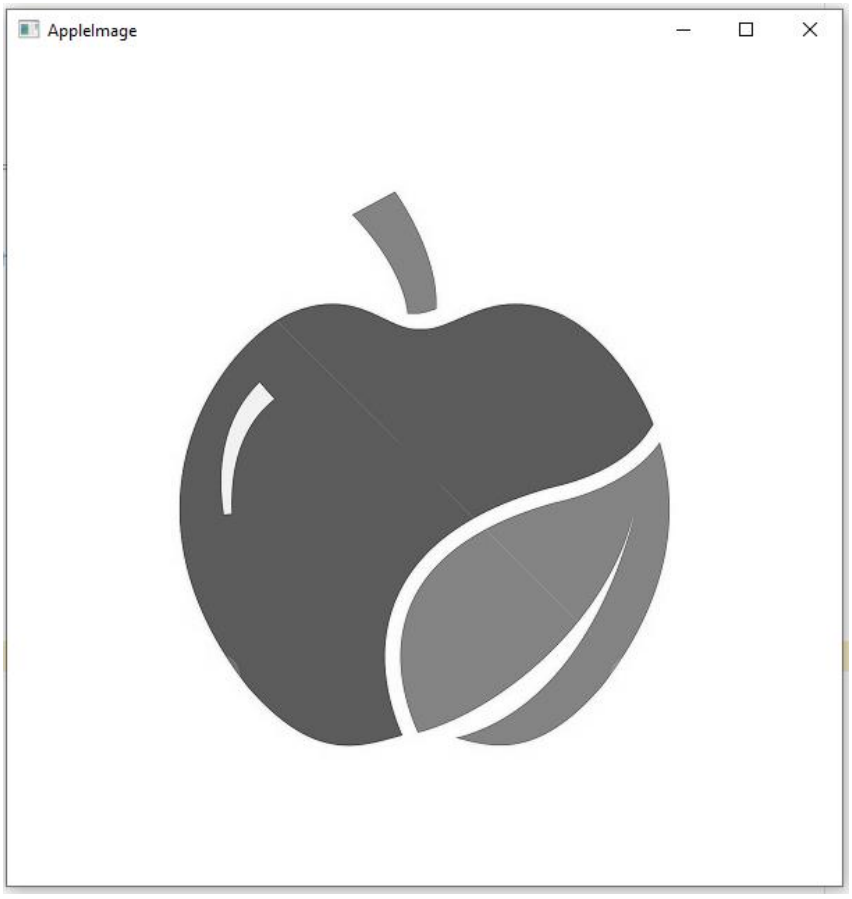

Рисунок 7 – Изменение цветовой гаммы изображения

Арифметические операции

Ввод изображения 1:

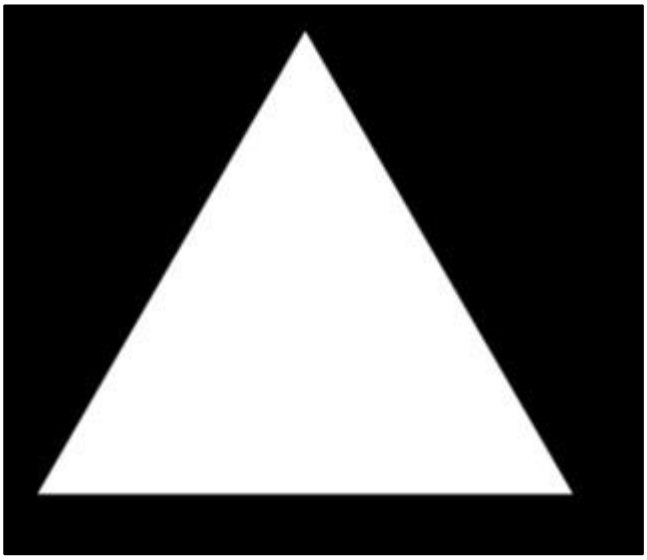

Рисунок 8 – Изображение №1

Ввод изображения 2:

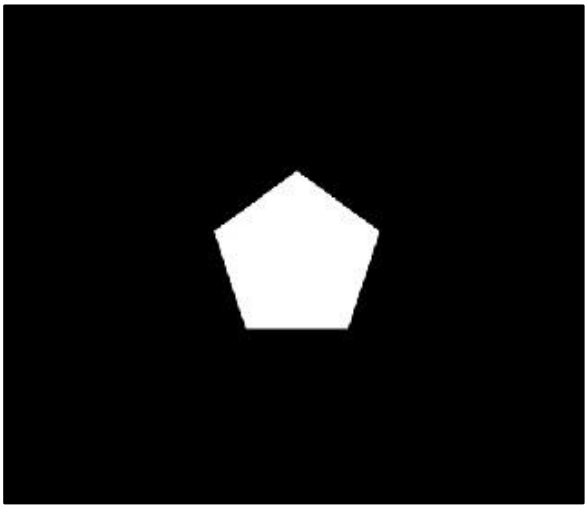

Рисунок 9 – Изображение №2

Суммирование изображений:

Код программы:

```
import cv2
import numpy as np

image1 = cv2.imread('Figure_1.jpg')
image2 = cv2.imread('Figure_2.jpg')

weightedSum = cv2.addWeighted(image1, 0.5, image2, 0.4, 0)

cv2.imshow('Weighted Image', weightedSum)

if cv2.waitKey(0) & 0xff == 27:
 cv2.destroyAllWindows()
```

Результат:

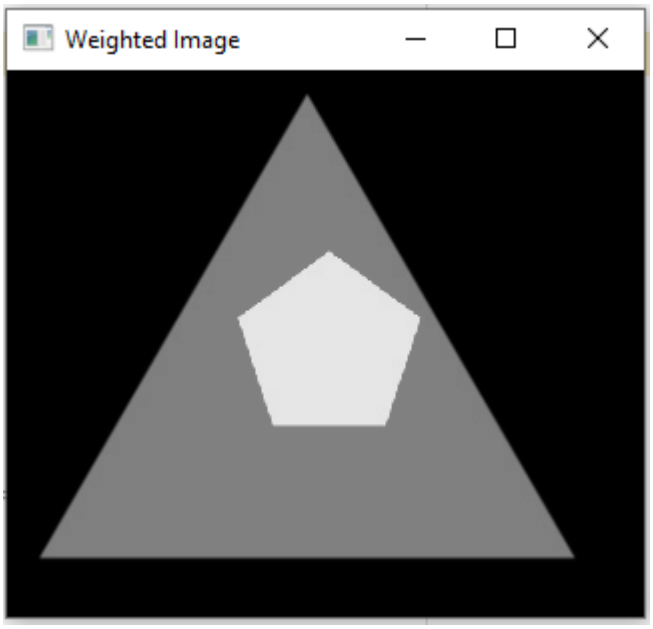

Рисунок 10 – Сочетание изображений

Вычитание изображений:

Код программы:

```
# organizing imports
import cv2
import numpy as np

# path to input images are specified and
# images are loaded with imread command
image1 = cv2.imread('Figure_1.jpg')
image2 = cv2.imread('Figure_2.jpg')

# cv2.subtract is applied over the
# image inputs with applied parameters
sub = cv2.subtract(image1, image2)

# the window showing output image
# with the subtracted image
cv2.imshow('Subtracted Image', sub)
```

```
# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
 cv2.destroyAllWindows()
```

Результат:

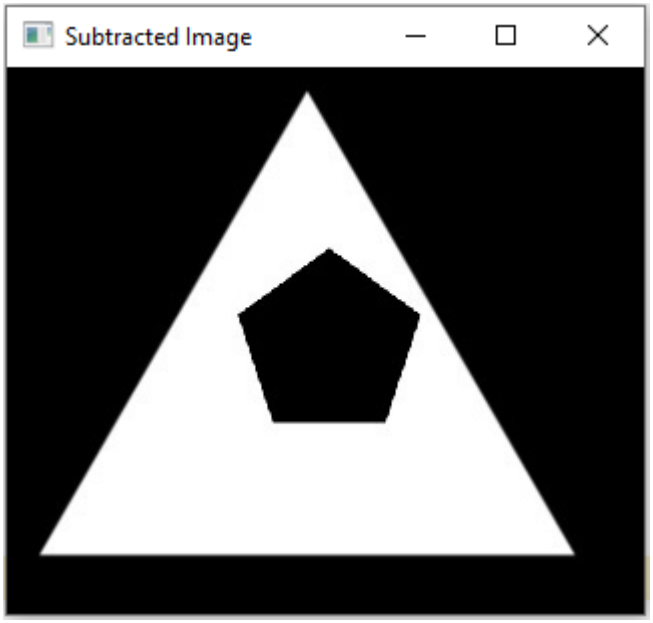

Рисунок 11 – Вычитание изображений

Побитовые операции над двоичным изображением

Входное изображение 1

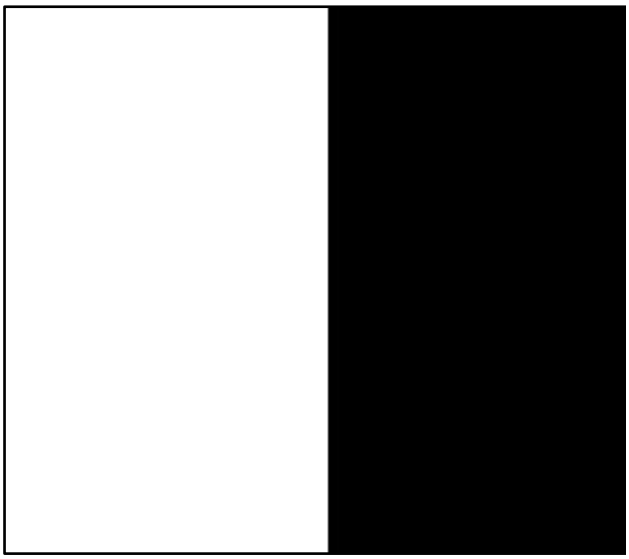

Рисунок 12 – Изображение №1

Входное изображение 2

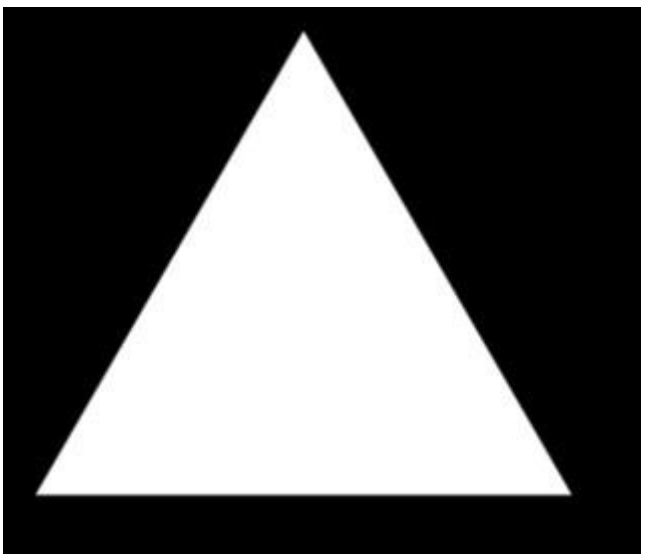

Рисунок 13 – Изображение №1

Побитовое И

Код программы:

```
# organizing imports
import cv2
import numpy as np

# path to input images are specified and
# images are loaded with imread command
img1 = cv2.imread('Figure_3.jpg')
img2 = cv2.imread('Figure_1.jpg')

# cv2.bitwise_and is applied over the
# image inputs with applied parameters
dest_and = cv2.bitwise_and(img2, img1, mask = None)

# the window showing output image
# with the Bitwise AND operation
# on the input images
cv2.imshow('Bitwise And', dest_and)

# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
 cv2.destroyAllWindows()
```

Результат:

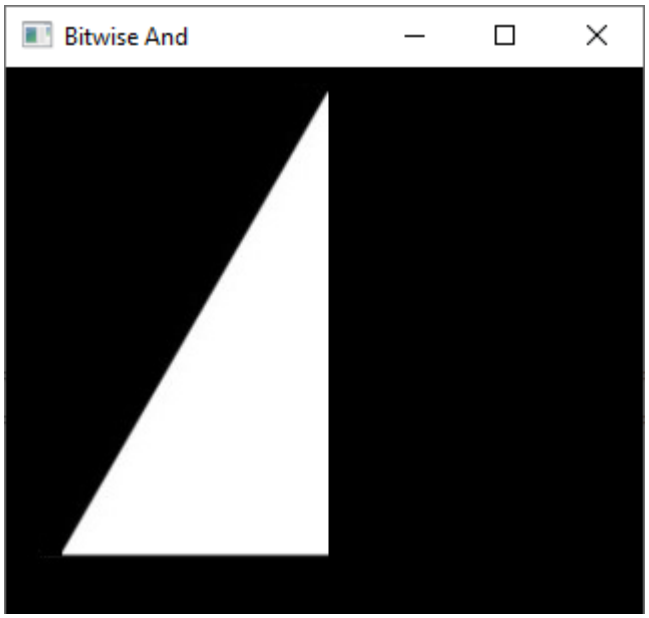

Рисунок 14 – Побитовое «И»

Побитовое ИЛИ

Код программы:

```
import cv2
import numpy as np
img1 = cv2.imread('Figure_3.jpg')
img2 = cv2.imread('Figure_1.jpg')
# cv2.bitwise_or is applied over the
# image inputs with applied parameters
dest_or = cv2.bitwise_or(img2, img1, mask = None)
```

```
# the window showing output image
# with the Bitwise OR operation
# on the input images
cv2.imshow('Bitwise OR', dest_or)

# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
 cv2.destroyAllWindows()
```

Результат:

Рисунок 15 – Побитовое «ИЛИ»

Побитовое XOR

Код программы:

```
import cv2
import numpy as np
img1 = cv2.imread('Figure_3.jpg')
img2 = cv2.imread('Figure_1.jpg')

# cv2.bitwise_xor is applied over the
# image inputs with applied parameters
dest_xor = cv2.bitwise_xor(img1, img2, mask = None)

# the window showing output image
# with the Bitwise XOR operation
# on the input images
cv2.imshow('Bitwise XOR', dest_xor)

# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
 cv2.destroyAllWindows()
```

Результат:

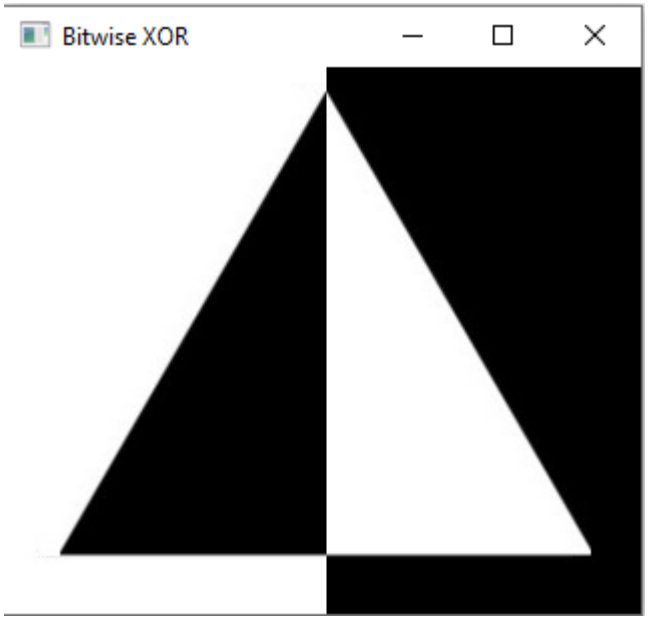

Рисунок 16 – Побитовое «XOR»

Побитовое NOT

Код программы:

```
import cv2
import numpy as np

img1 = cv2.imread('Figure_3.jpg')
img2 = cv2.imread('Figure_1.jpg')

dest_not1 = cv2.bitwise_not(img1, mask=None)
dest_not2 = cv2.bitwise_not(img2, mask=None)

# the windows showing output image
# with the Bitwise NOT operation
# on the 1st and 2nd input image
cv2.imshow('Bitwise NOT on image 1', dest_not1)
cv2.imshow('Bitwise NOT on image 2', dest_not2)

# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
  cv2.destroyAllWindows()
```

Результат:

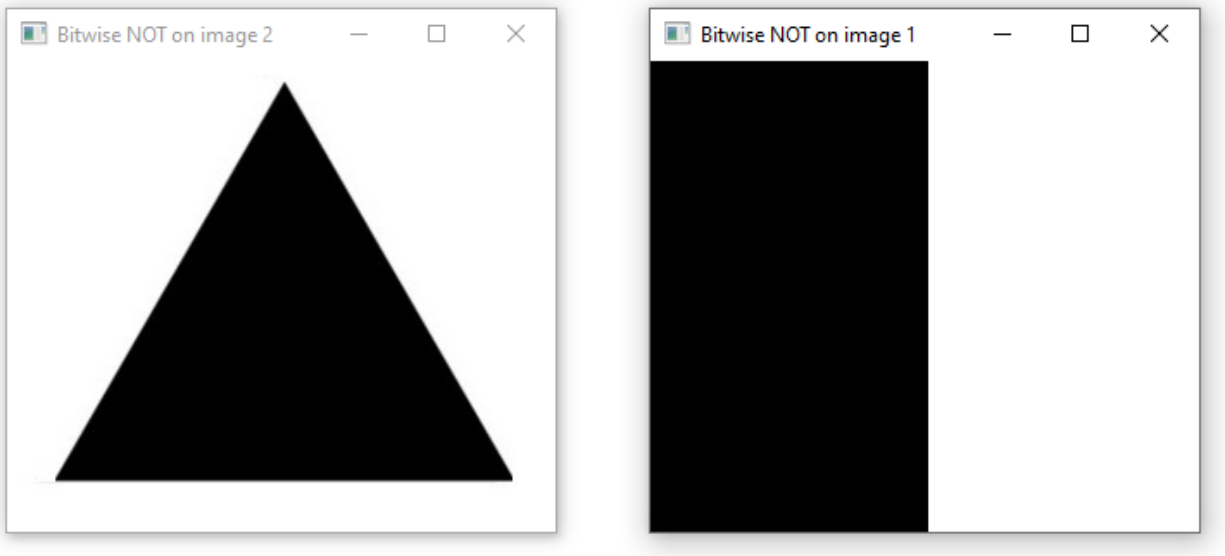

Рисунок 17 – Побитовое «NOT»

Смещение изображений Python OpenCV

Код программы:

```
import cv2
import numpy as np

image = cv2.imread('AppleImage.jpg')

# Store height and width of the image
height, width = image.shape[:2]

quarter_height, quarter_width = height / 4, width / 4

T = np.float32([[1, 0, quarter_width], [0, 1, quarter_height]])

# We use warpAffine to transform
# the image using the matrix, T
img_translation = cv2.warpAffine(image, T, (width, height))

cv2.imshow('Translation', img_translation)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Результат:

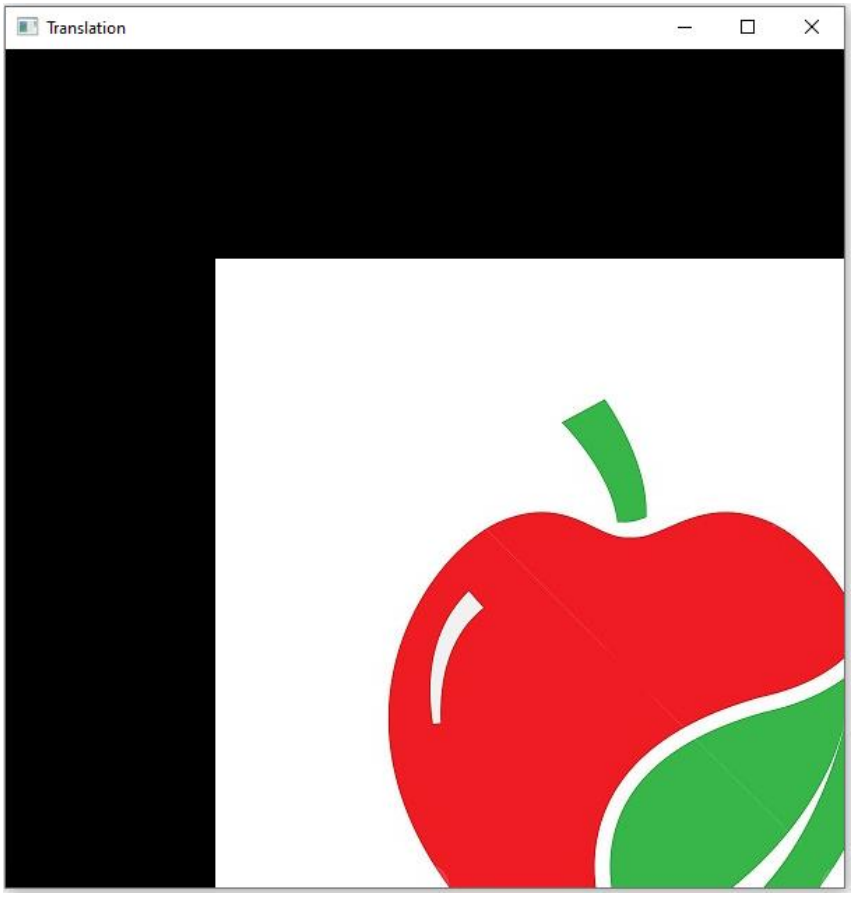

Рисунок 18 – Смещение изображения

Обнаружение границ

Код программы:

```
import cv2

FILE_NAME = 'AppleImage.jpg'

# Read image from disk.
img = cv2.imread(FILE_NAME)

# Canny edge detection.
edges = cv2.Canny(img, 100, 200)

# Write image back to disk.
cv2.imshow('Edges', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Результат:

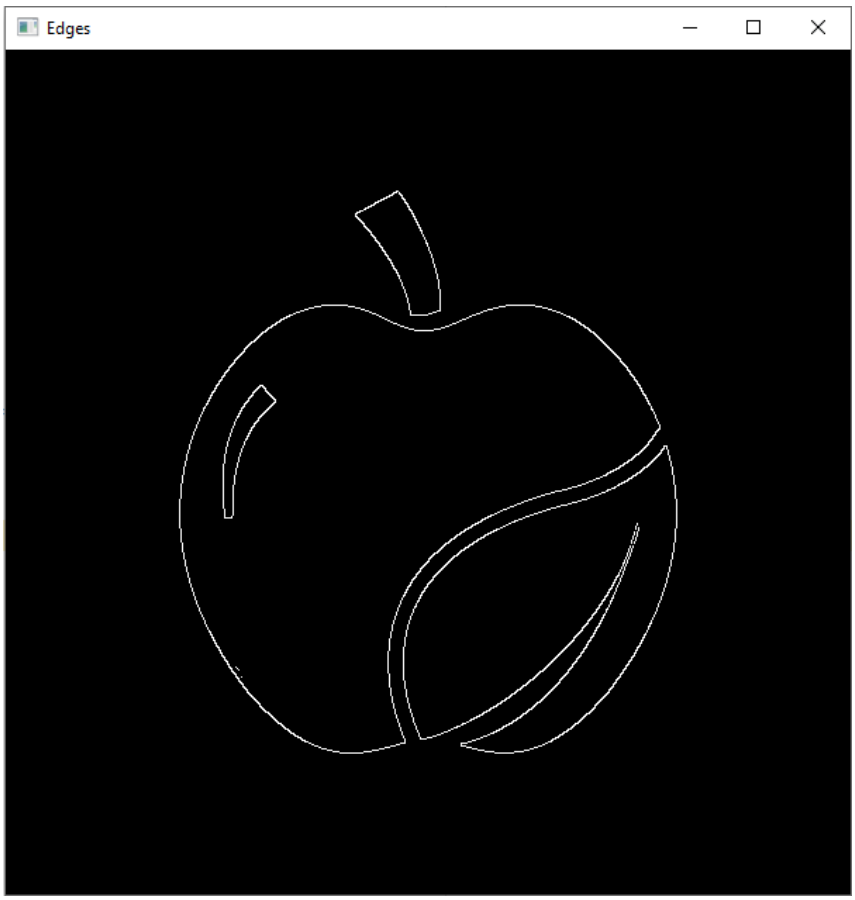

Рисунок 19 – Обнаружение границ изображения

Определение порога

Код программы:

```
# Python program to illustrate
# simple thresholding type on an image

# organizing imports
import cv2
import numpy as np

# path to input image is specified and
# image is loaded with imread command
image1 = cv2.imread('AppleImage.jpg')

img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)

ret, thresh1 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(img, 120, 255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(img, 120, 255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(img, 120, 255, cv2.THRESH_TOZERO_INV)


cv2.imshow('Binary Threshold', thresh1)
cv2.imshow('Binary Threshold Inverted', thresh2)
cv2.imshow('Truncated Threshold', thresh3)
cv2.imshow('Set to 0', thresh4)
cv2.imshow('Set to 0 Inverted', thresh5)
```

```
# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
 cv2.destroyAllWindows()
```

Результат:

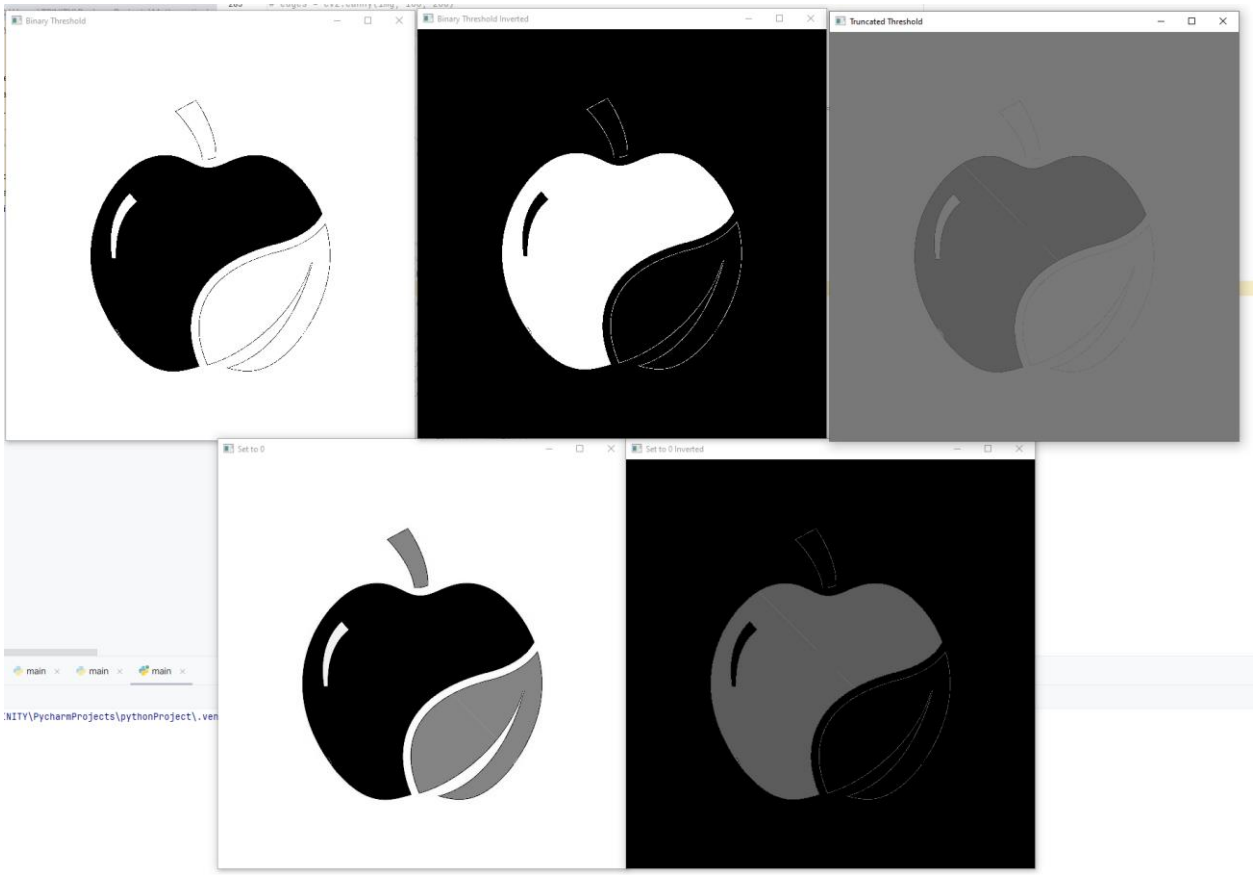

Рисунок 20 – Пороговая обработка изображения

Адаптивное пороговое значение

Код программы:

```
import cv2
import numpy as np

image1 = cv2.imread('AppleImage.jpg')

img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)

# techniques on the input image
thresh1 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
 cv2.THRESH_BINARY, 199, 5)

thresh2 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
 cv2.THRESH_BINARY, 199, 5)

# techniques applied to the input image
cv2.imshow('Adaptive Mean', thresh1)
cv2.imshow('Adaptive Gaussian', thresh2)

# De-allocate any associated memory usage
```

```
if cv2.waitKey(0) & 0xff == 27:
 cv2.destroyAllWindows()
```

Результат:

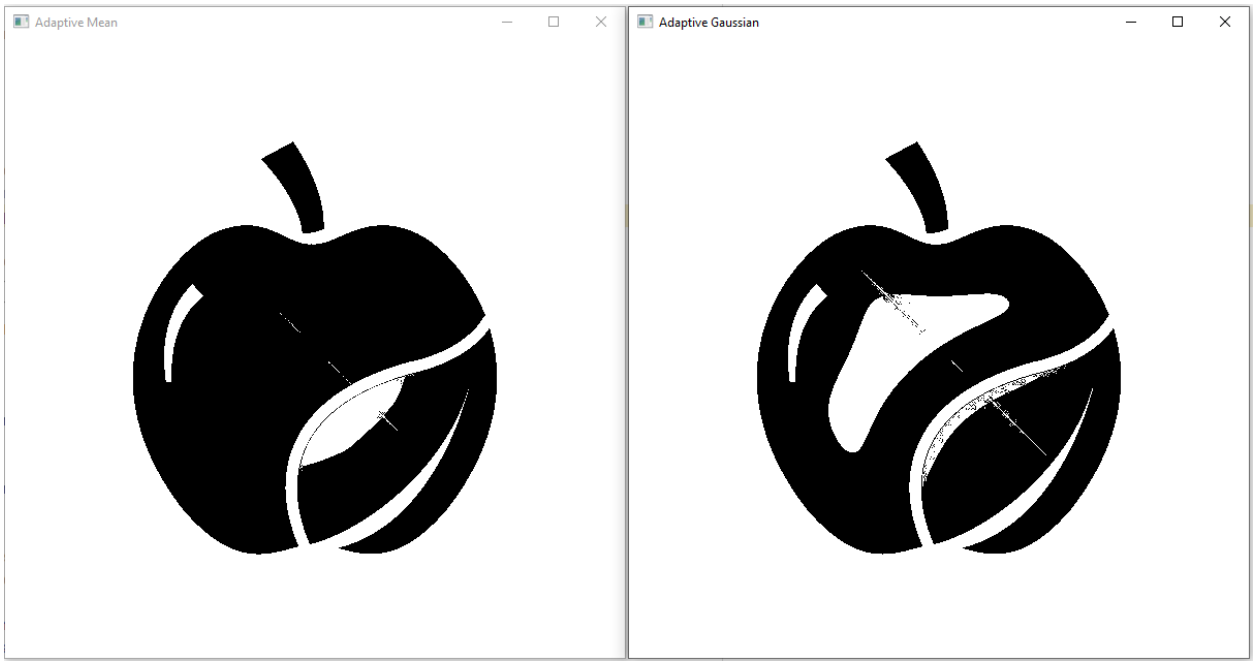

Рисунок 21 – Адаптивная пороговая обработка изображения

Пороговое значение Otsu

Код программы:

```
import cv2
import numpy as np

# path to input image is specified and
# image is loaded with imread command
image1 = cv2.imread('AppleImage.jpg')

img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)

ret, thresh1 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY +
 cv2.THRESH_OTSU)

cv2.imshow('Otsu Threshold', thresh1)

# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
 cv2.destroyAllWindows()
```

Результат:

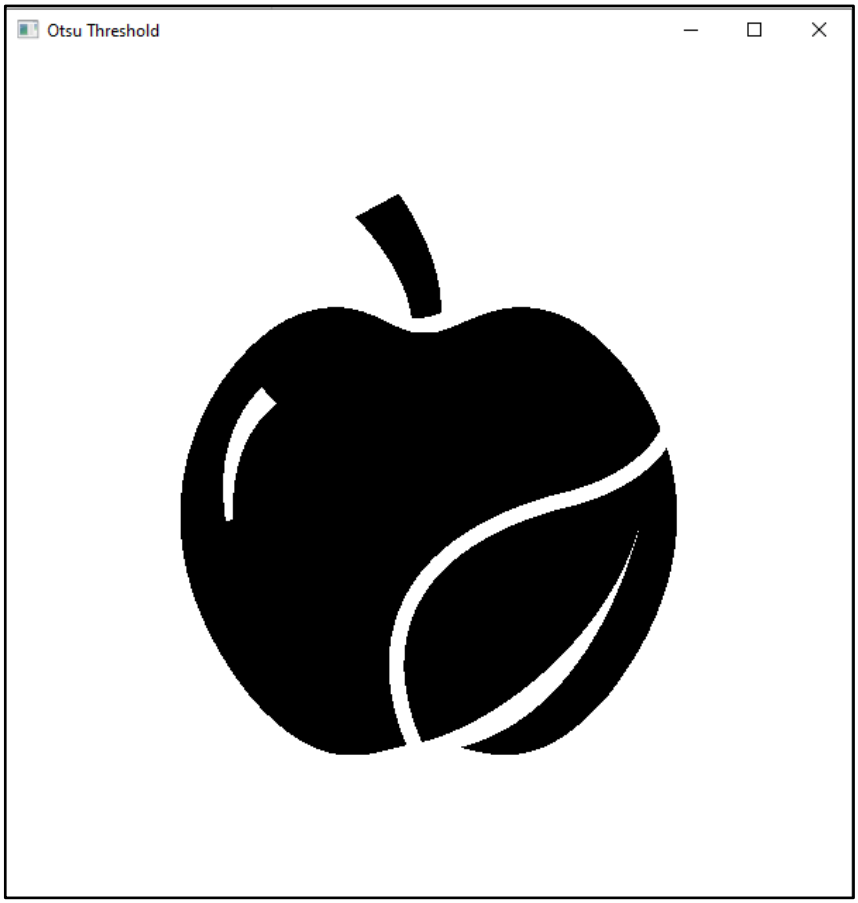

Рисунок 22 – Пороговое значение OTSU

Размытие изображений

Код программы:

```
# importing libraries
import cv2
import numpy as np

image = cv2.imread('AppleImage.jpg')

cv2.imshow('Original Image', image)
cv2.waitKey(0)

# Gaussian Blur
Gaussian = cv2.GaussianBlur(image, (7, 7), 0)
cv2.imshow('Gaussian Blurring', Gaussian)
cv2.waitKey(0)

# Median Blur
median = cv2.medianBlur(image, 5)
cv2.imshow('Median Blurring', median)
cv2.waitKey(0)


# Bilateral Blur
bilateral = cv2.bilateralFilter(image, 9, 75, 75)
cv2.imshow('Bilateral Blurring', bilateral)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Результат:

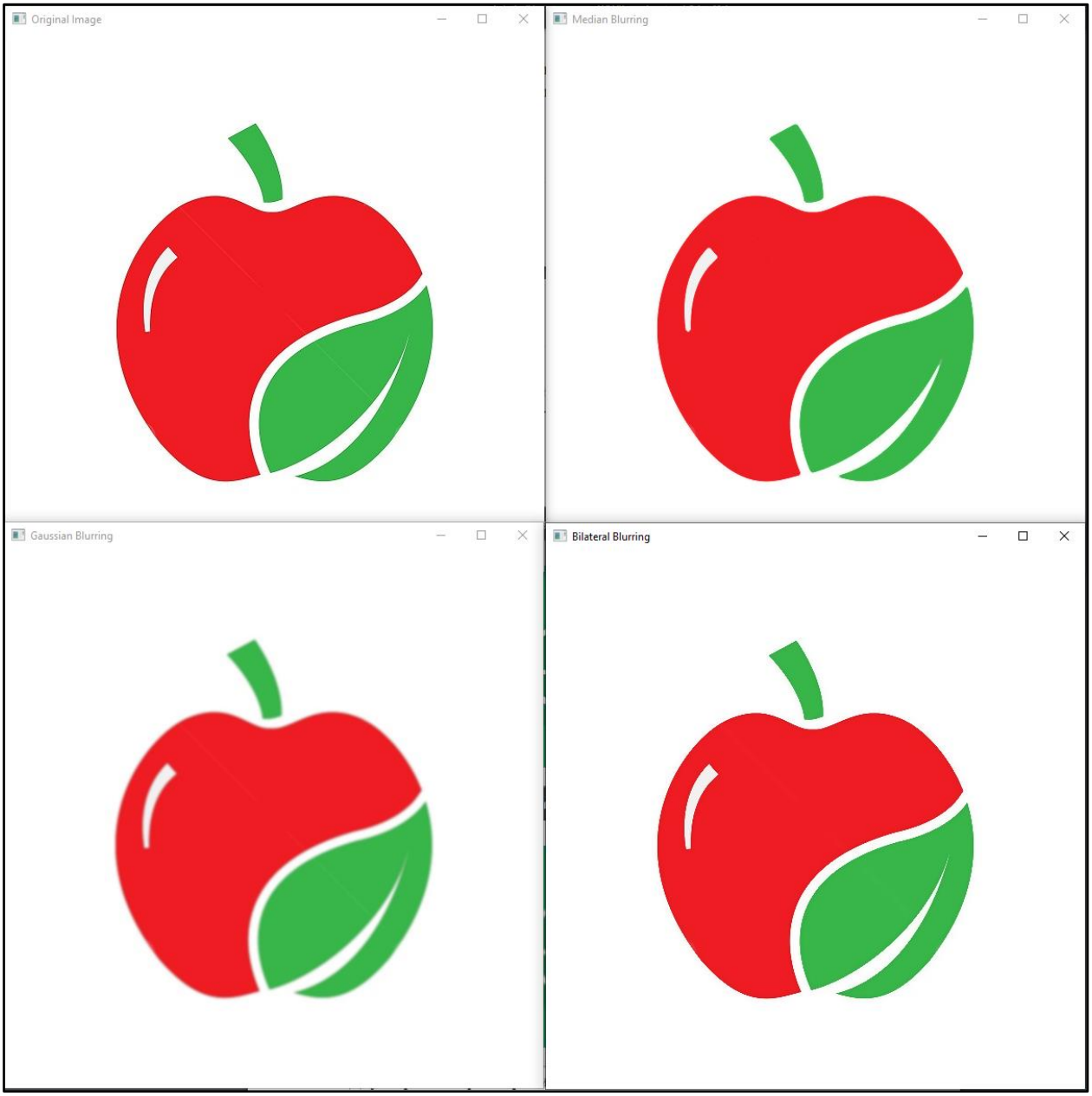

Рисунок 23 – Размытие изображения

Двусторонняя фильтрация

Код программы:

```
import cv2

# Read the image
img = cv2.imread('AppleImage.jpg')

# Apply bilateral filter with d = 30,
# sigmaColor = sigmaSpace = 100
bilateral = cv2.bilateralFilter(img, 15, 100, 100)

# Save the output
cv2.imshow('Bilateral', bilateral)
```

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

Результат:

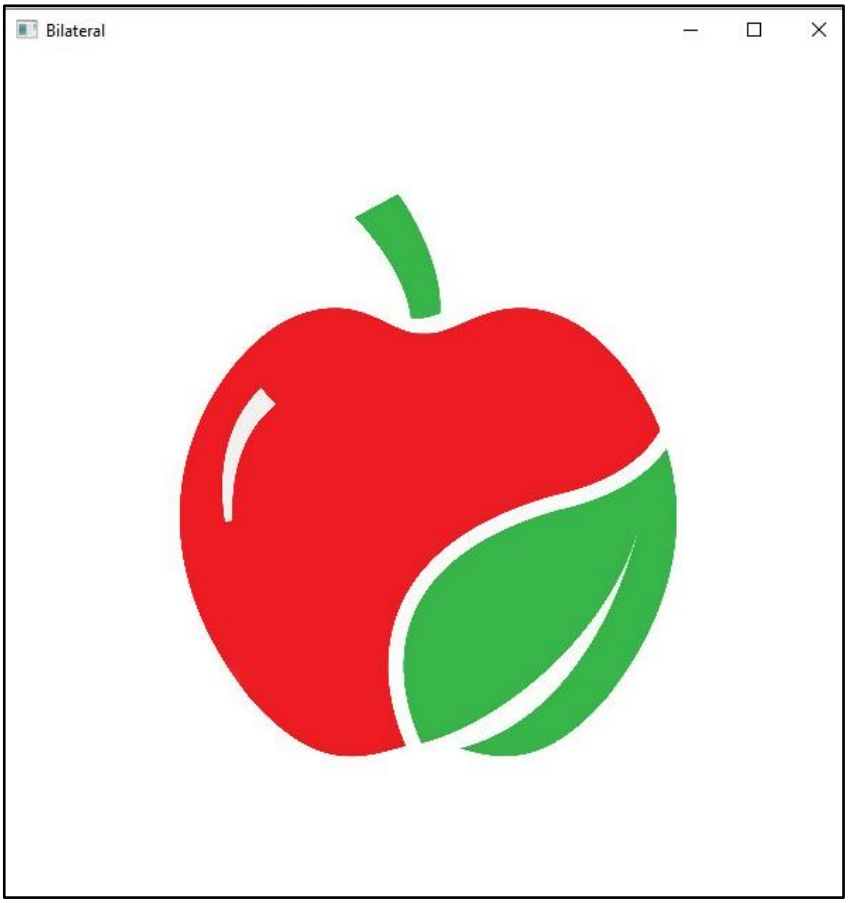

Рисунок 24 – Двусторонняя фильтрация

Контуры изображения

Код программы:

```
import cv2
import numpy as np

# Let's load a simple image with 3 black squares
image = cv2.imread('AppleImage.jpg')
cv2.waitKey(0)

# Grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Find Canny edges
edged = cv2.Canny(gray, 30, 200)
cv2.waitKey(0)

# Finding Contours
# Use a copy of the image e.g. edged.copy()
# since findContours alters the image
contours, hierarchy = cv2.findContours(edged,
 cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
```

```
cv2.imshow('Canny Edges After Contouring', edged)
cv2.waitKey(0)

print("Number of Contours found = " + str(len(contours)))

# Draw all contours
# -1 signifies drawing all contours
cv2.drawContours(image, contours, -1, (0, 255, 0), 3)

cv2.imshow('Contours', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Результат:

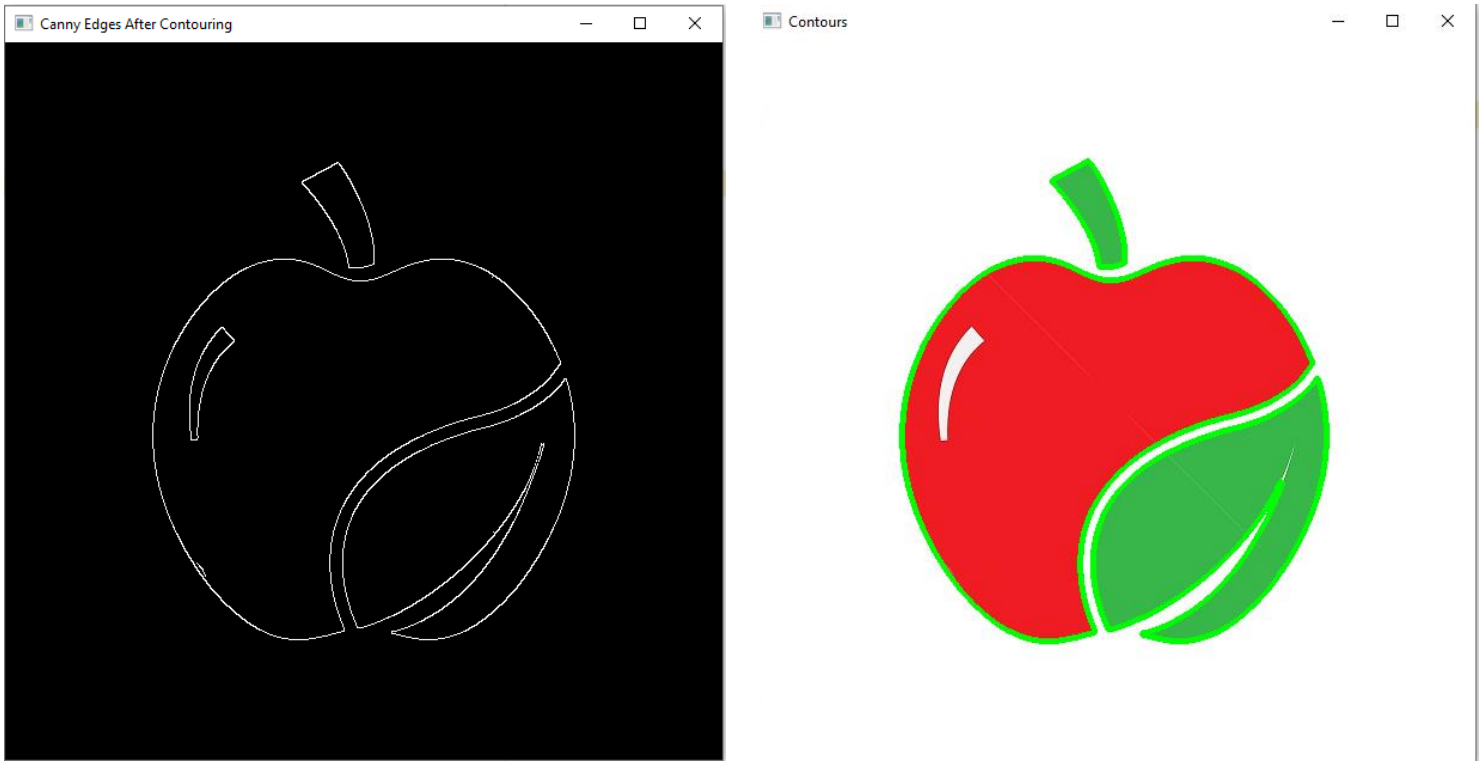

Рисунок 25 – Контуры изображения

Размывание и расширение

Код программы:

```
import cv2
import numpy as np

# Reading the input image
img = cv2.imread('AppleImage.jpg')

# Taking a matrix of size 5 as the kernel
kernel = np.ones((5,5), np.uint8)

img_erosion = cv2.erode(img, kernel, iterations=1)
img_dilation = cv2.dilate(img, kernel, iterations=1)

cv2.imshow('Input', img)
```

```
cv2.imshow('Erosion', img_erosion)
cv2.imshow('Dilation', img_dilation)

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

Результат:

Рисунок 26 – Размывание и расширение

Сопоставление функций

Код программы:

```
i import numpy as np
import cv2

query_img = cv2.imread('AppleImage.jpg')
```

```
train_img = cv2.imread('AppleImage.jpg')

# Convert it to grayscale
query_img_bw = cv2.cvtColor(query_img,cv2.COLOR_BGR2GRAY)
train_img_bw = cv2.cvtColor(train_img, cv2.COLOR_BGR2GRAY)

# Initialize the ORB detector algorithm
orb = cv2.ORB_create()

# and train image
queryKeypoints, queryDescriptors = orb.detectAndCompute(query_img_bw,None)
trainKeypoints, trainDescriptors = orb.detectAndCompute(train_img_bw,None)

# keypoints
matcher = cv2.BFMatcher()
matches = matcher.match(queryDescriptors,trainDescriptors)


# its train image
final_img = cv2.drawMatches(query_img, queryKeypoints,
train_img, trainKeypoints, matches[:20],None)

final_img = cv2.resize(final_img, (1000,650))

# Show the final image
cv2.imshow("Matches", final_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Результат:

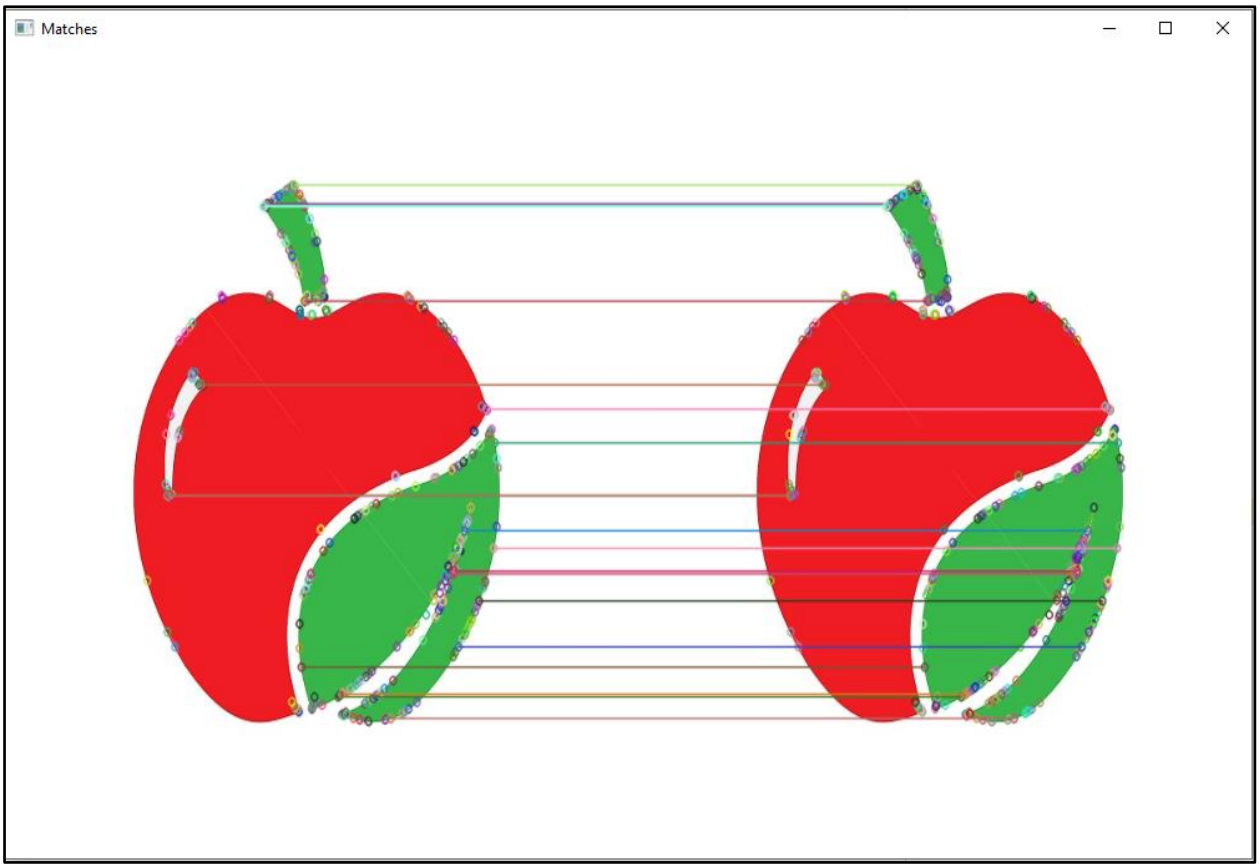

Рисунок 27 – Размывание и расширение

Рисование на изображениях

Код программы:

```
# Python3 program to draw rectangle
# shape on solid image
import numpy as np
import cv2

# Creating a black image with 3
# channels RGB and unsigned int datatype
img = np.zeros((400, 400, 3), dtype = "uint8")

# Creating rectangle
cv2.rectangle(img, (30, 30), (300, 200), (0, 255, 0), 5)

cv2.imshow('dark', img)

# Allows us to see image
# until closed forcefully
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Результат:

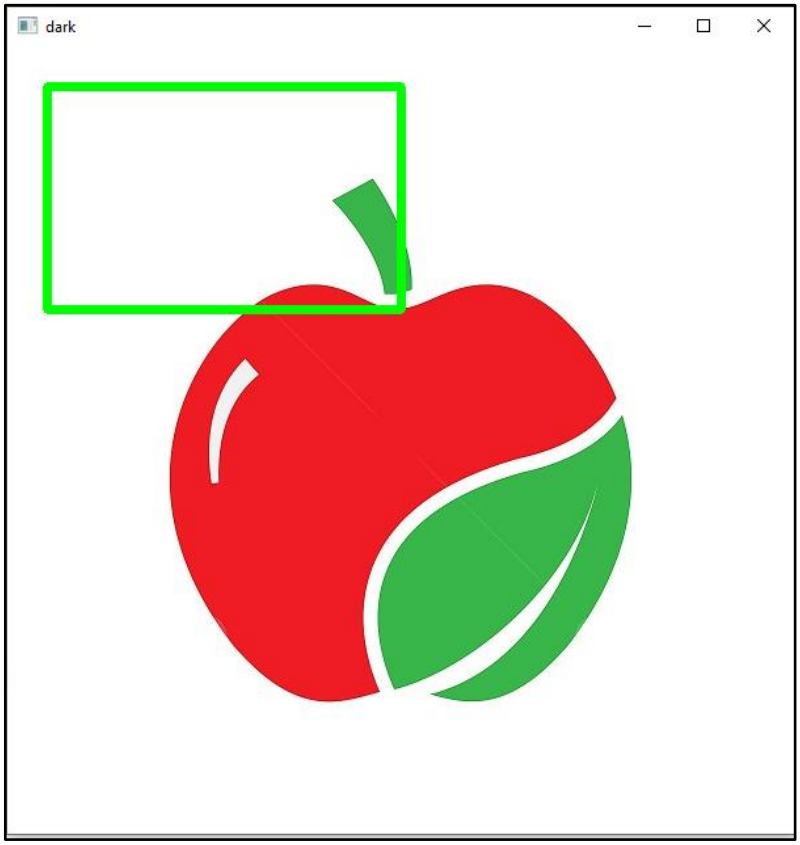

Рисунок 28 – Размывание и расширение