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»

Выполнил:

Студенты гр. АПИМ-24, АВТФ:

Разуваве Владислав Валерьевич

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:

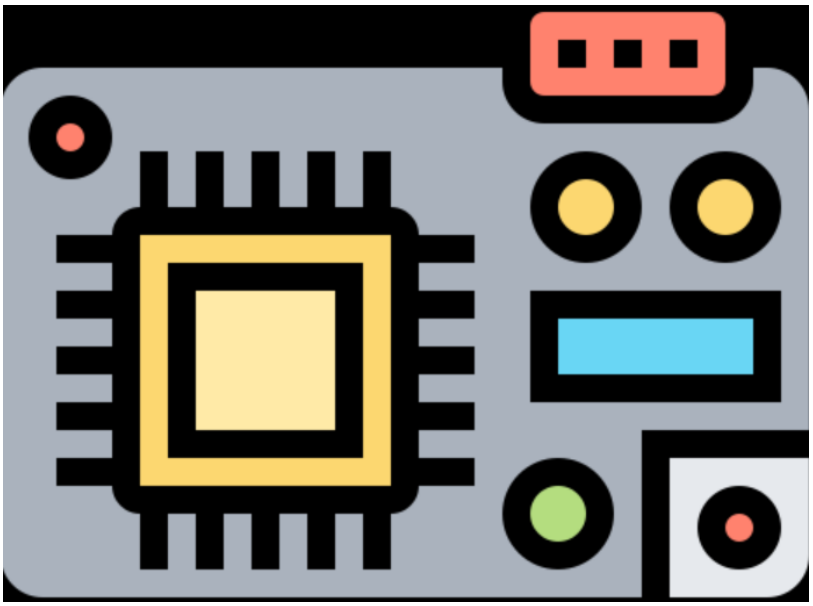

Рисунок 1 – Исходное изображение

Код программы:

```
import cv2
img = cv2.imread("pain.png", cv2.IMREAD_COLOR)
cv2.imshow("pain", img)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Чтение и отображение изображения

Результат:

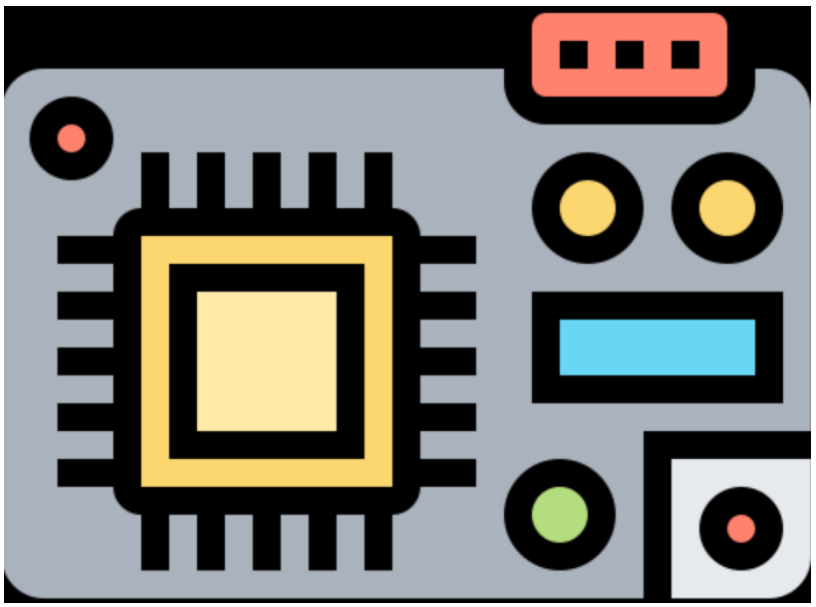

Рисунок 2 – Чтение и отображение

Сохранение изображений

Код программы:

```
import cv2

img = cv2.imread("pain.png", 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

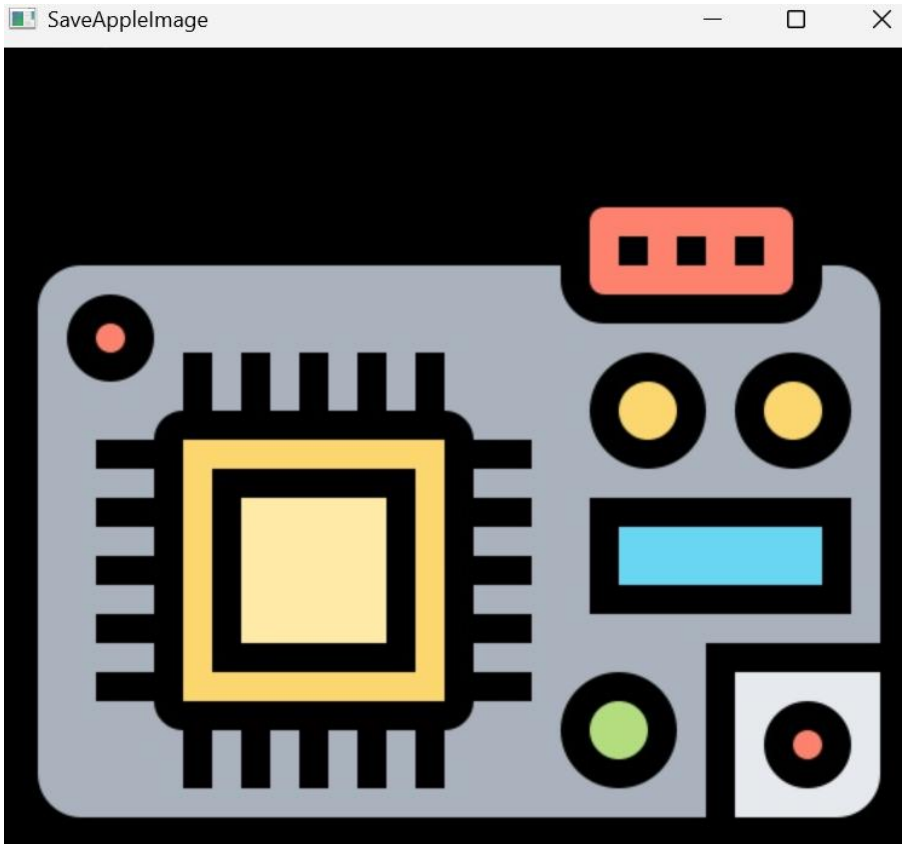

Рисунок 3 – Сохраненное изображение

Вращение изображений

Код программы:

```
import cv2

path = "pain.png"
window_name = "Image"

src = cv2.imread(path)

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

cv2.imshow(window_name, img)
cv2.waitKey(0)
```

Результат:

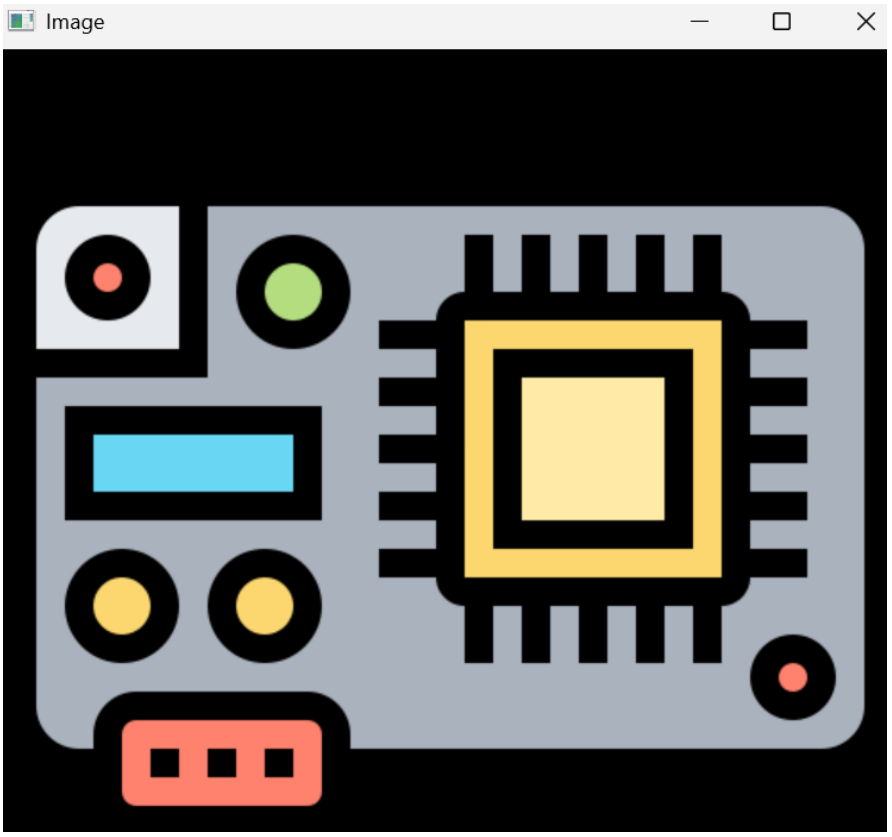

Рисунок 4 – Поворот на 180 градусов

Вращение на 45 градусов

Код программы:

```
import cv2
import numpy as np

path = "pain.png"
window_name = "Image"

src = cv2.imread(path)

(rows, cols) = src.shape[:2]

m = cv2.getRotationMatrix2D((cols/2, rows/2), 45, 1)
res = cv2.warpAffine(src, m, (cols, rows))



cv2.imshow(window_name, res)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Результат:

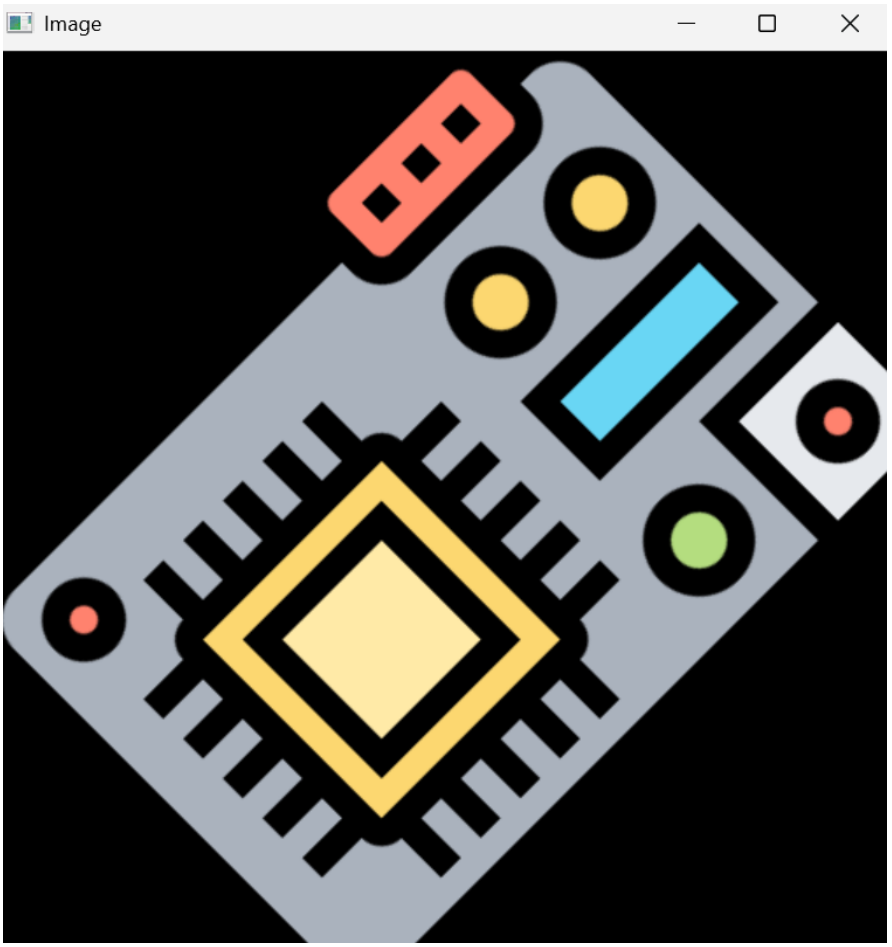

Рисунок 5 – Поворот на 45 градусов

Изменение размера изображения

Код программы:

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

path = "pain.png"
window_name = "Image"

img = cv2.imread(path)

half = cv2.resize(img, (0,0), fx = 0.1, fy = 0.1)
bigger = cv2.resize(img, (1050, 1610))
stretch_near = cv2.resize(img, (780, 540), interpolation =
cv2.INTER_NEAREST)
Titles =["Original", "Half", "Bigger", "Interpolation Nearest"]
images =[img, 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:

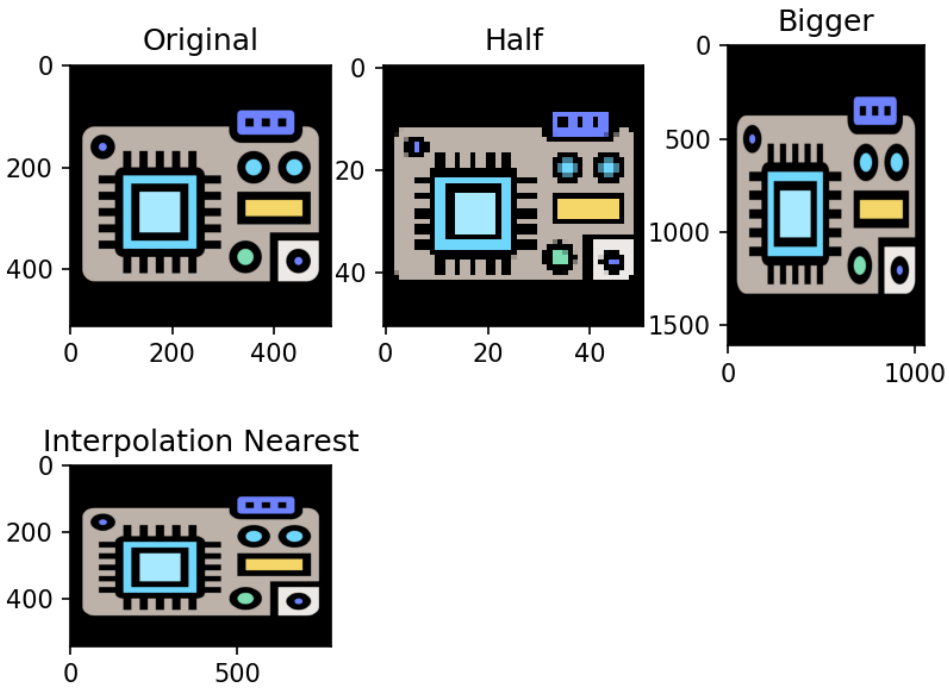

Рисунок 6 – Изменение размеров изображения

Цветовые пространства Python OpenCV. ЧБ изображение.

Код программы:

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

path = "pain.png"
window_name = "Image"

img = cv2.imread(path)

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

cv2.imshow(window_name, img_color)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Результат:

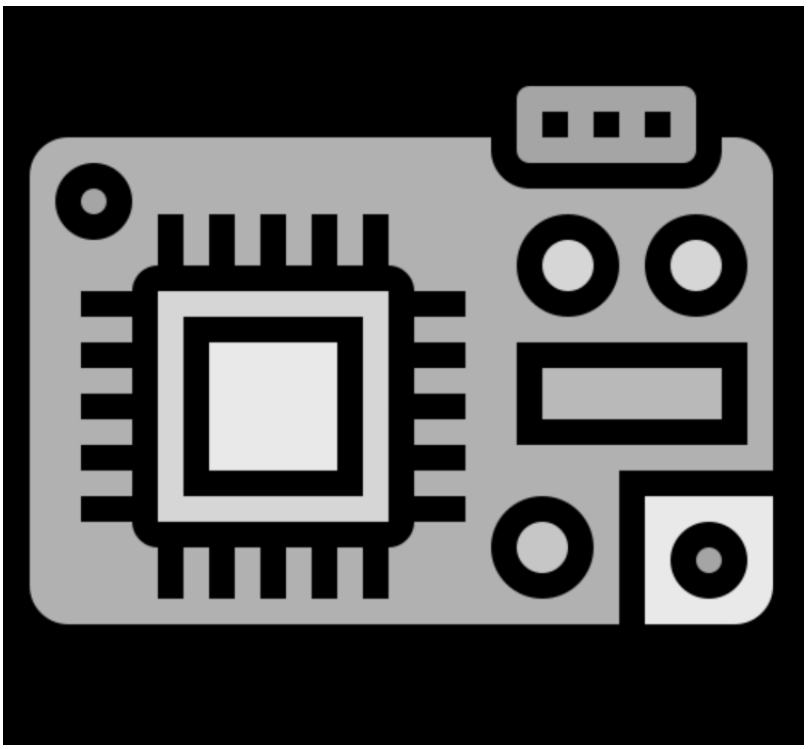

Рисунок 7 – Изменение цветовой гаммы изображения

Арифметические операции

Ввод изображения 1:

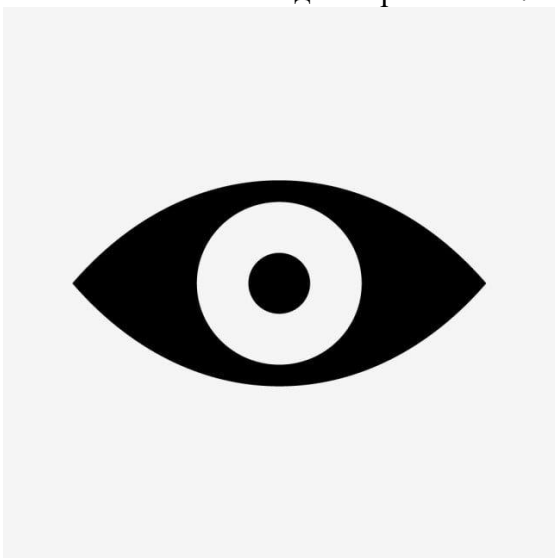

Рисунок 8 – Изображение №1

Ввод изображения 2

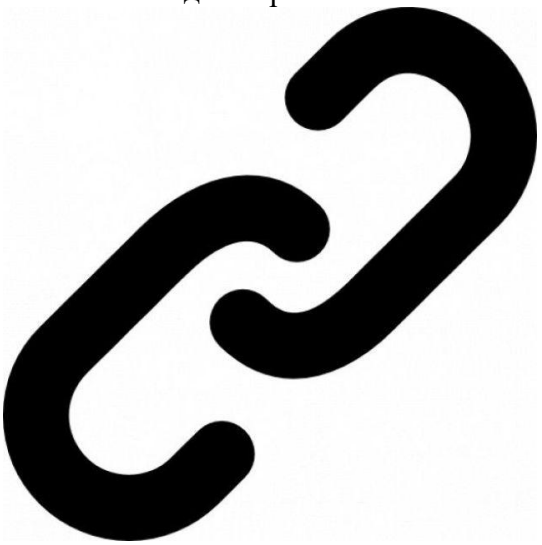

Рисунок 9 – Изображение №2

Суммирование изображений:

Код программы:

```
import cv2
import numpy as np
image1 = cv2.imread('1.jpg')
image2 = cv2.imread('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:

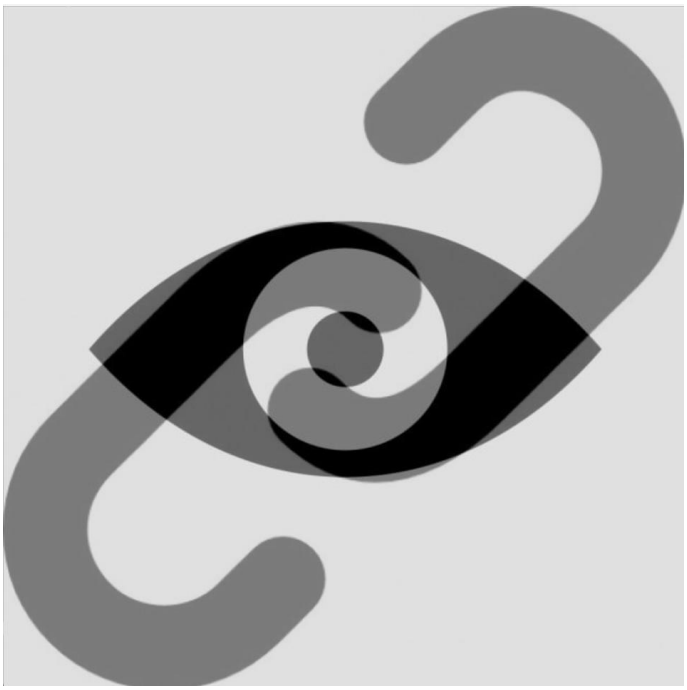

Рисунок 10 – Сочетание изображений

Вычитание изображений:

Код программы:

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

# path to input images are specified and # images are loaded with
imread command
image1 = cv2.imread('1.jpg')
image2 = cv2.imread('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

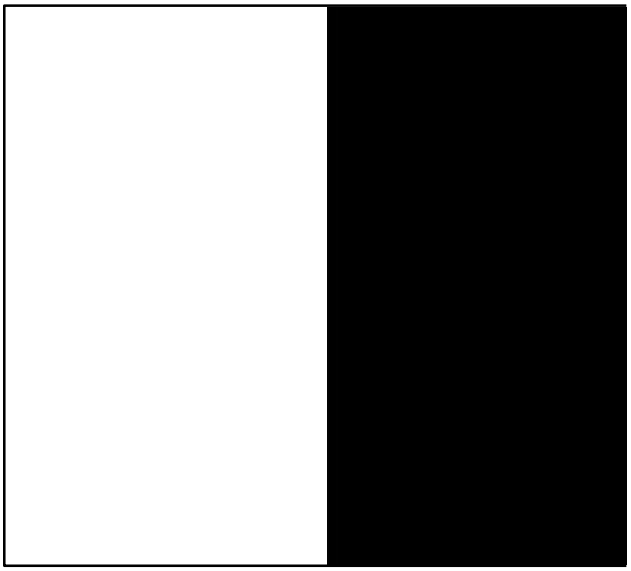

Рисунок 12 – Изображение №1

Входное изображение 2

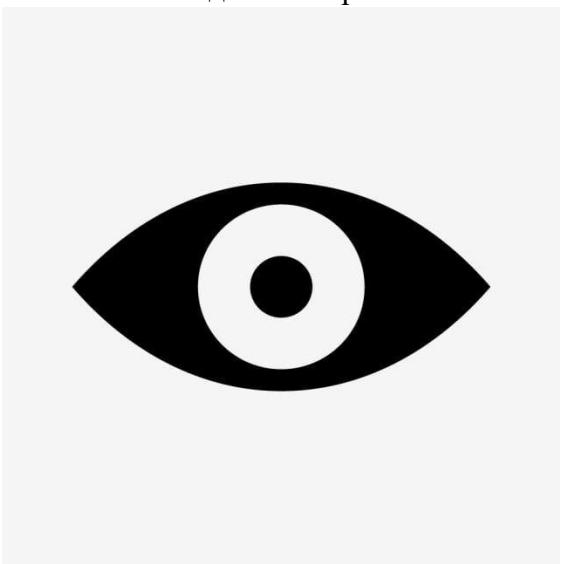

Рисунок 13 – Изображение №1

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('3.jpg')
  img2 = cv2.imread('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('3.jpg')
  img2 = cv2.imread('1.jpg')
  dest_or = cv2.bitwise_or(img2, img1, mask = None)
```

```
cv2.imshow('Bitwise OR', dest_or)
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```

Результат:

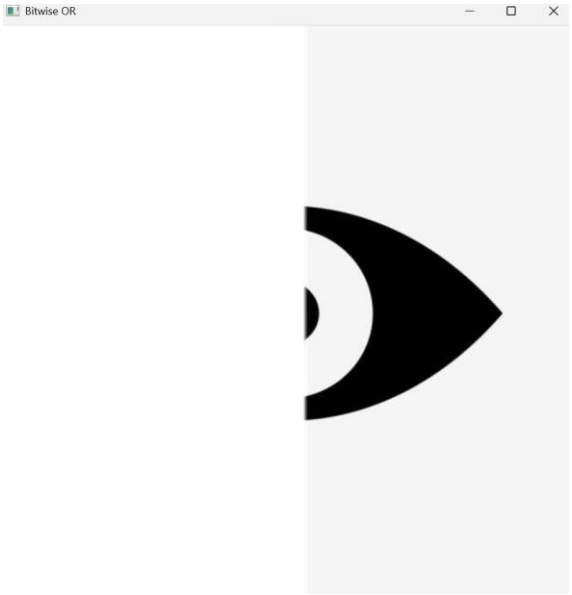

Рисунок 15 – Побитовое «ИЛИ»

Побитовое XOR

Код программы:

```
import cv2

img1 = cv2.imread('3.jpg')
img2 = cv2.imread('1.jpg')
dest_xor = cv2.bitwise_xor(img2, img1, mask = None)

cv2.imshow('Bitwise XOR', dest_xor)
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```

Результат:

Рисунок 16 – Побитовое «XOR»

Побитовое NOT

Код программы:

```
import cv2

img1 = cv2.imread('3.jpg')
img2 = cv2.imread('1.jpg')
dest_not1 = cv2.bitwise_not(img1, mask = None)
dest_not2 = cv2.bitwise_not(img2, mask = None)

cv2.imshow('Bitwise NOT on image 1', dest_not1)
cv2.imshow('Bitwise NOT on image 2', dest_not2)
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```

Результат:

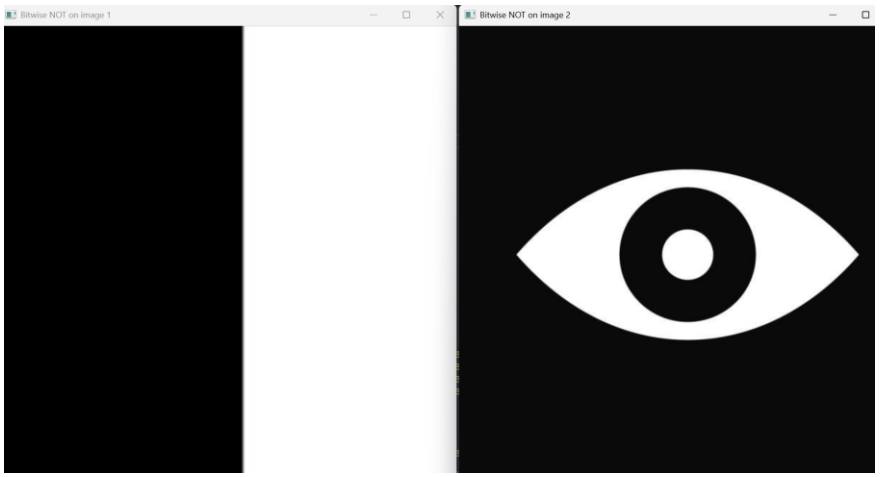

Рисунок 17 – Побитовое «NOT»

Смещение изображений Python OpenCV

Код программы:

```
import cv2
import numpy as np

image = cv2.imread('pain.png')

height, width = image.shape[:2]
quarter_height, quarter_width = height / 4, width / 4

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

img_translation = cv2.warpAffine(image, T, (width, height))

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

cv2.destroyAllWindows()
```

Результат:

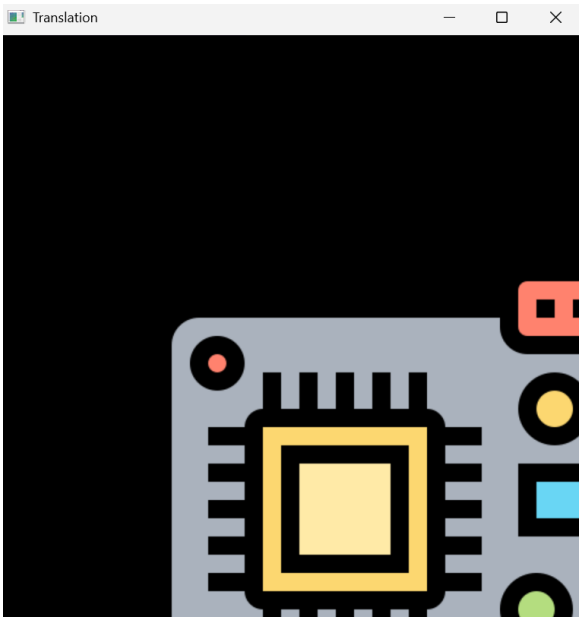

Рисунок 18 – Смещение изображения

Обнаружение границ

Код программы:

```
import cv2

FILE_NAME = 'pain.png'

img = cv2.imread(FILE_NAME)

edges = cv2.Canny(img, 100, 200)
cv2.imshow('Edges', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Результат:

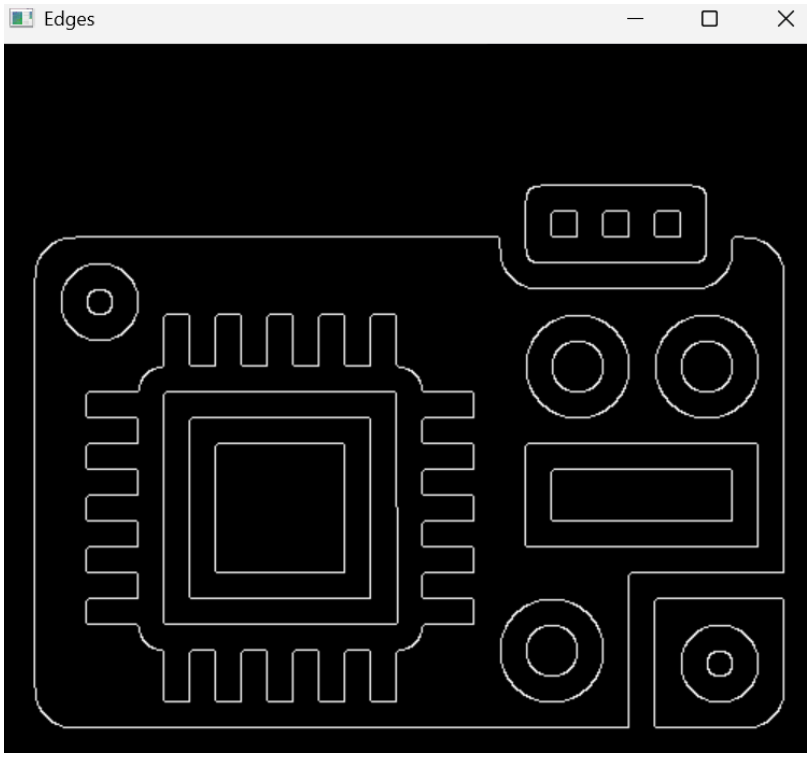

Рисунок 19 – Обнаружение границ изображения

Определение порога

Код программы:

```
import cv2
import numpy as np

image1 = cv2.imread('pain.png')

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)

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

Результат:

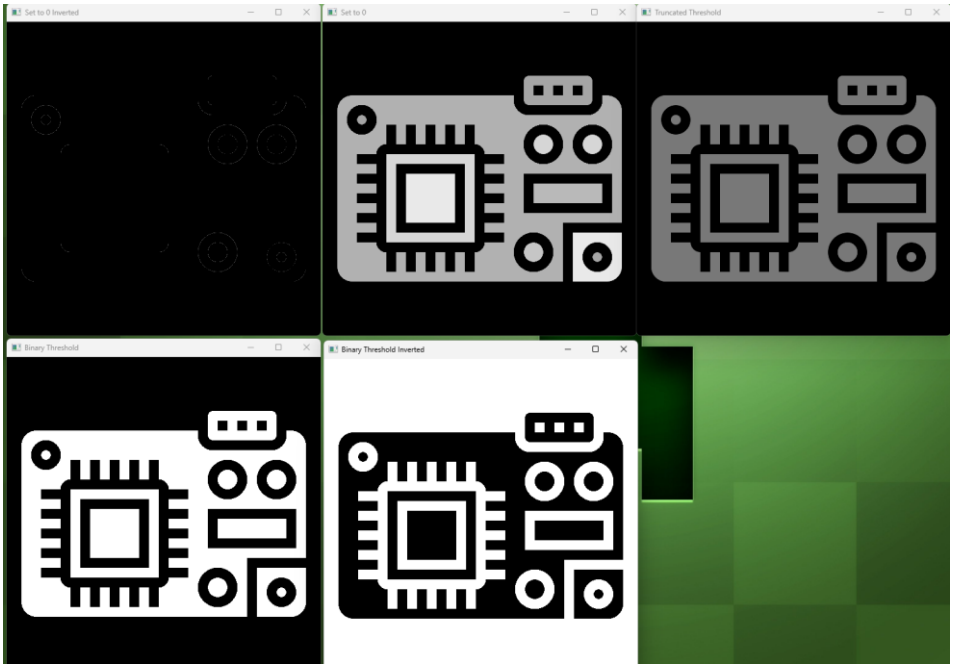

Рисунок 20 – Пороговая обработка изображения

Адаптивное пороговое значение

Код программы:

```
import cv2

image1 = cv2.imread('pain.png')
img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
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)

cv2.imshow('Adaptive Mean', thresh1)
cv2.imshow('Adaptive Gaussian', thresh2)

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

Результат:

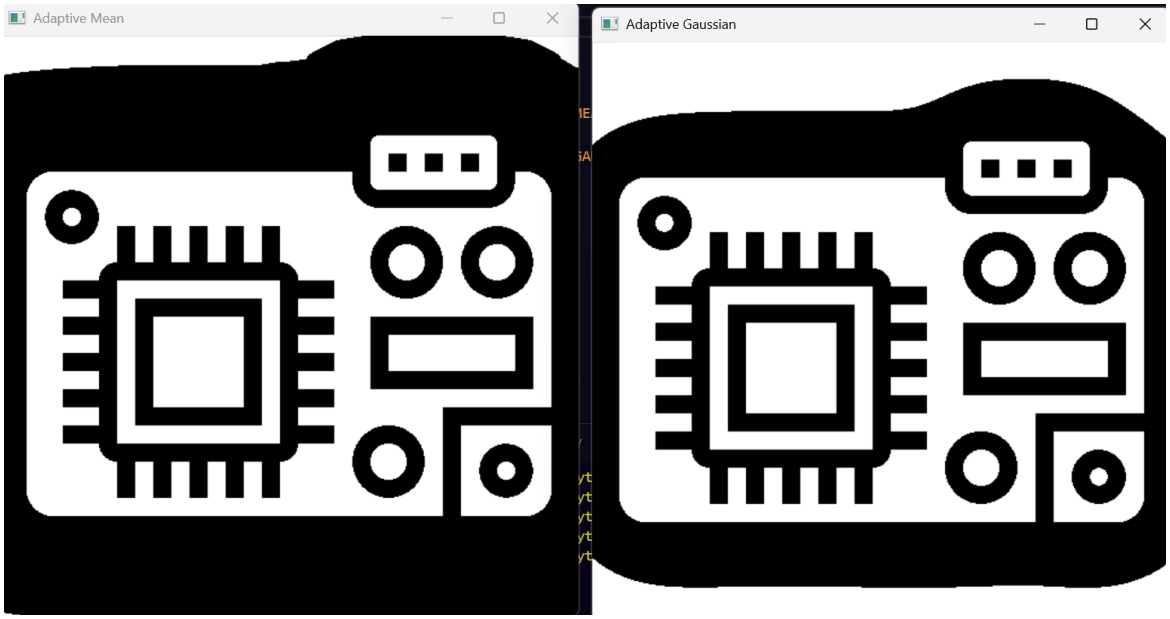

Рисунок 21 – Адаптивная пороговая обработка изображения

Пороговое значение Otsu

Код программы:

```
import cv2

image1 = cv2.imread('pain.png')

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)


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

Результат:

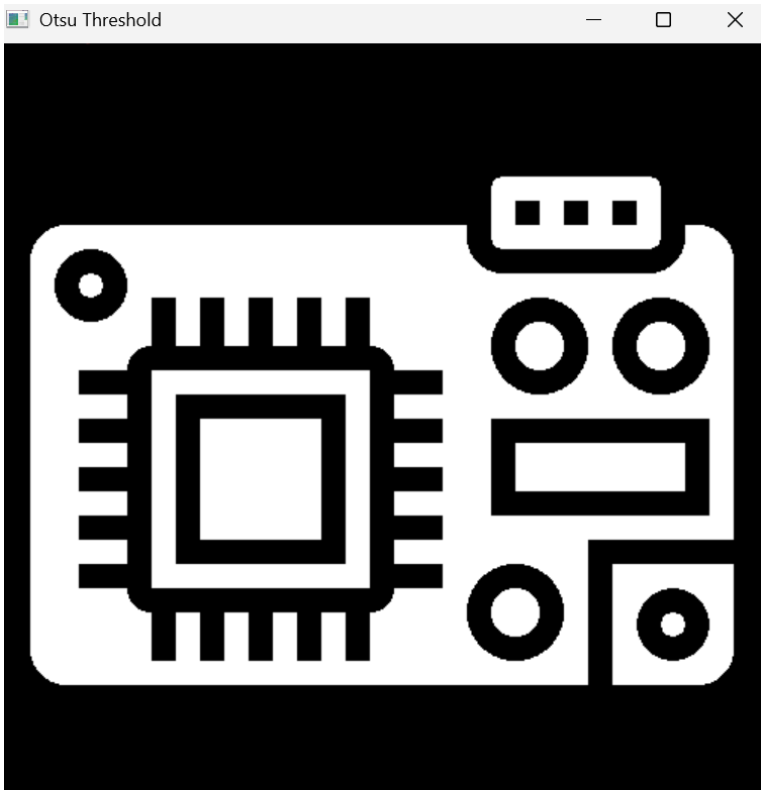

Рисунок 22 – Пороговое значение OTSU

Размытие изображений

Код программы:

```
import cv2
image = cv2.imread('pain.png')

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

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

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

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

Результат:

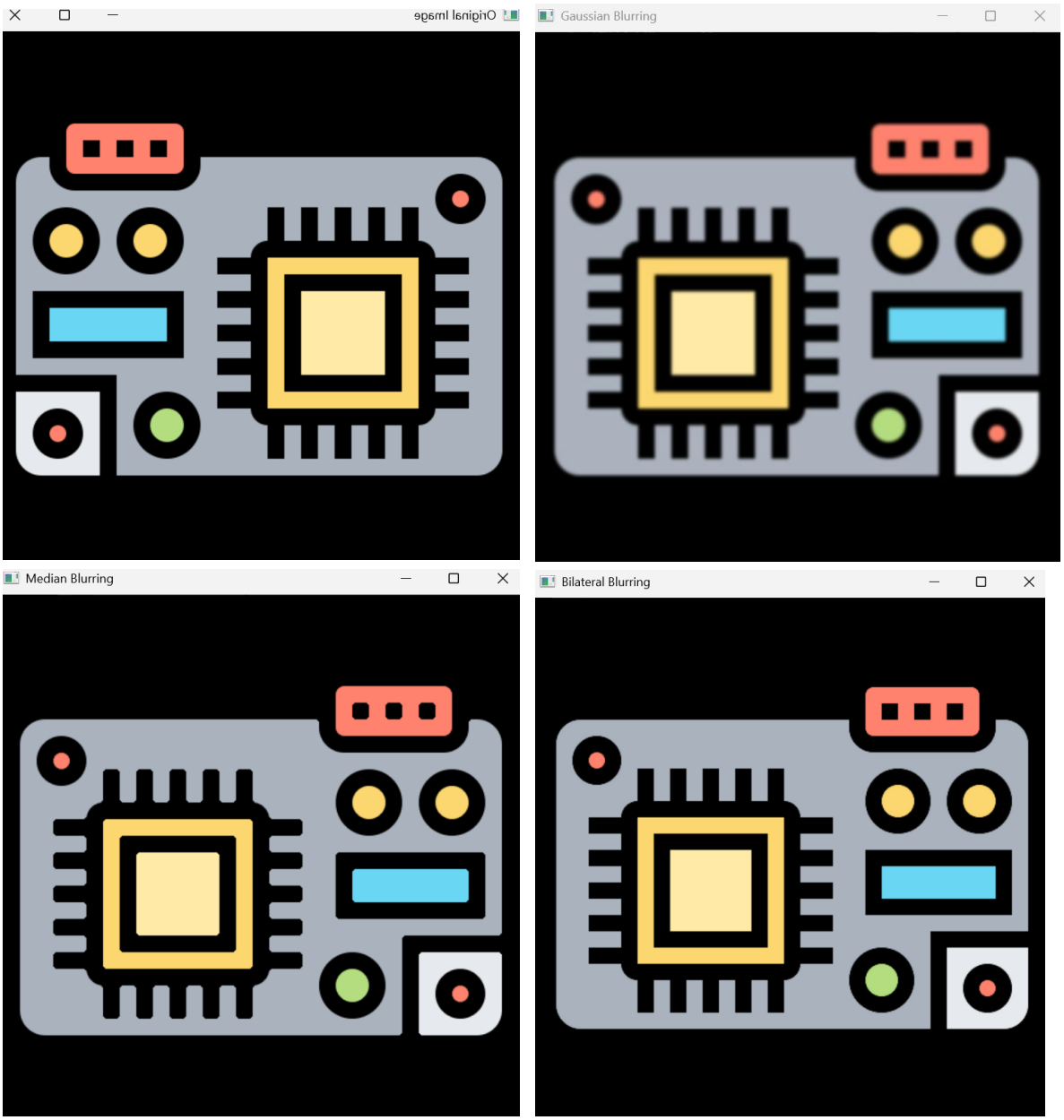

Рисунок 23 – Размытие изображения

Двусторонняя фильтрация

Код программы:

```
import cv2

img = cv2.imread('pain.png')
bilateral = cv2.bilateralFilter(img, 15, 100, 100)

cv2.imshow('Bilateral', bilateral)

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

Результат:

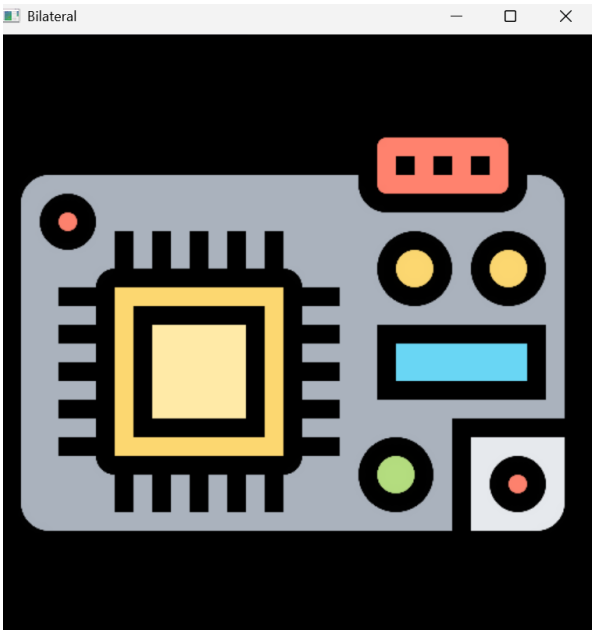

Рисунок 24 – Двусторонняя фильтрация

Контуры изображения

Код программы:

```
import cv2
import numpy as np
image = cv2.imread('pain.png')
cv2.waitKey(0)

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

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

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)))

cv2.drawContours(image, contours, -1, (0, 255, 0), 3)
cv2.imshow('Contours', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Результат:

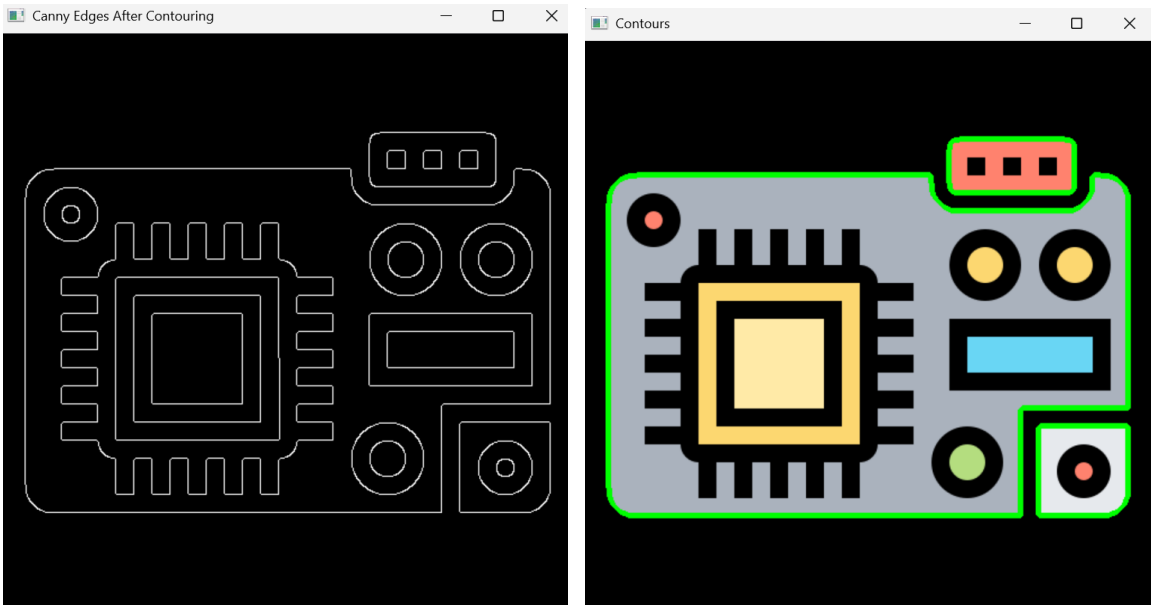

Рисунок 25 – Контуры изображения

Размывание и расширение

Код программы:

```
import cv2
import numpy as np
img = cv2.imread('pain.png')

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:

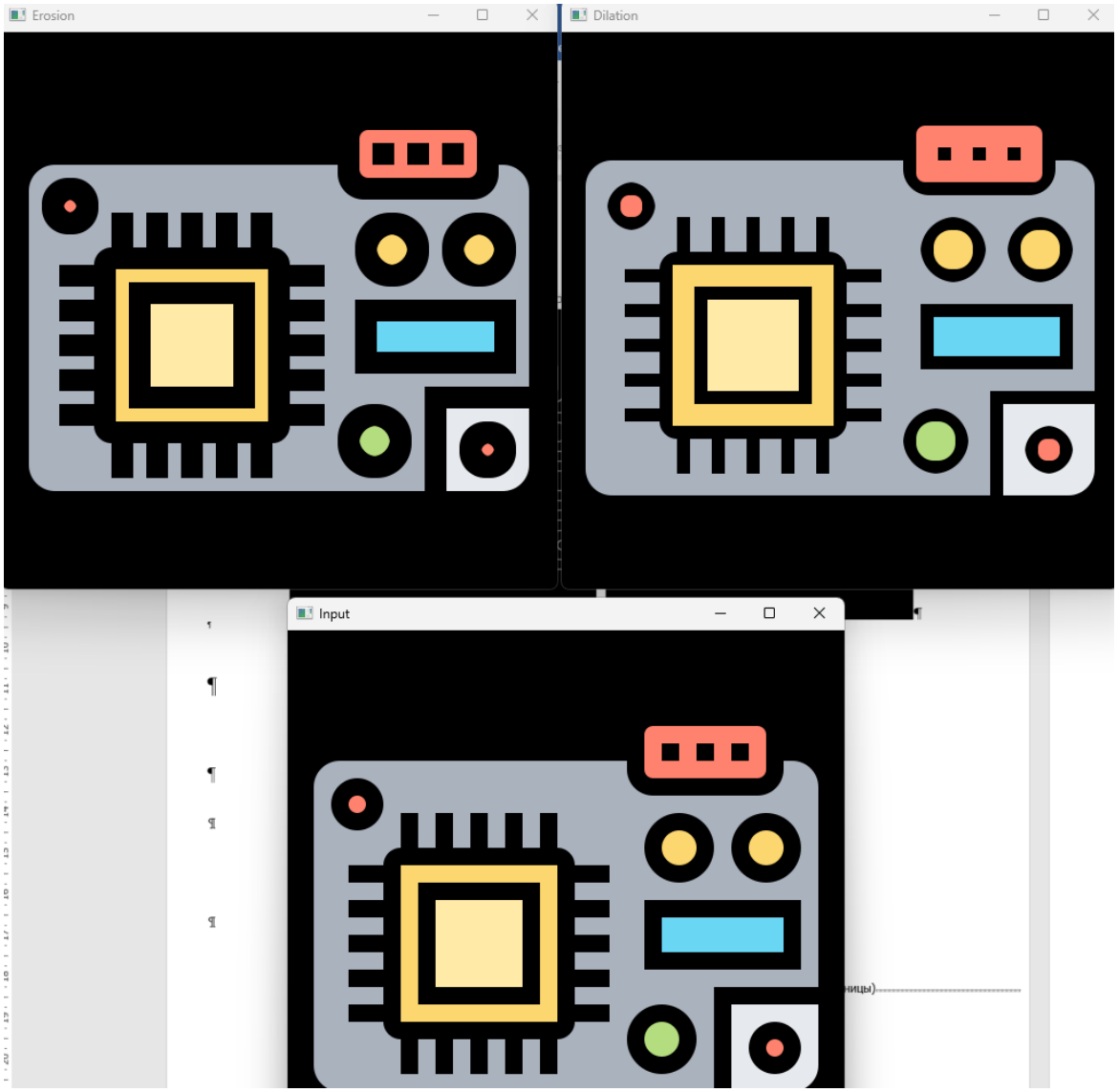

Рисунок 26 – Размывание и расширение

Сопоставление функций

Код программы:

```
import numpy as np
import cv2

query_img = cv2.imread('pain.png')

train_img = cv2.imread('pain.png')

query_img_bw = cv2.cvtColor(query_img,cv2.COLOR_BGR2GRAY)
train_img_bw = cv2.cvtColor(train_img, cv2.COLOR_BGR2GRAY)

orb = cv2.ORB_create()

queryKeypoints, queryDescriptors =
orb.detectAndCompute(query_img_bw,None)
trainKeypoints, trainDescriptors =
orb.detectAndCompute(train_img_bw,None)
```

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

    final_img = cv2.drawMatches(query_img, queryKeypoints, train_img,
trainKeypoints, matches[:20],None)
    final_img = cv2.resize(final_img, (1000,650))
    cv2.imshow("Matches", final_img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Результат:

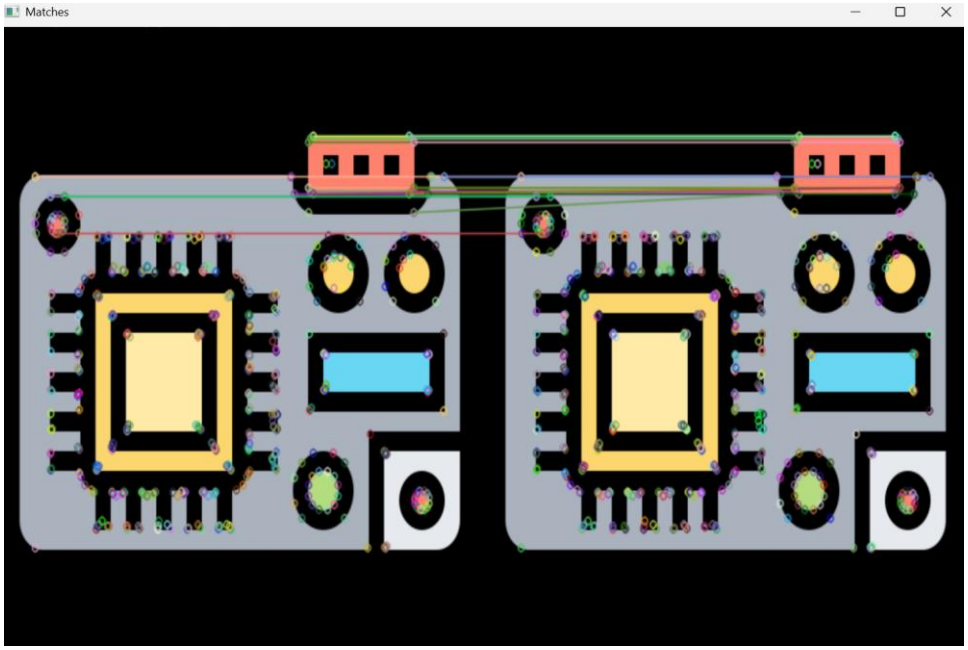

Рисунок 27 – Размывание и расширение

Рисование на изображениях

Код программы:

```
import cv2

src = cv2.imread('pain.png')

cv2.rectangle(src, (30, 30), (300, 200), (0, 255, 0), 5)

cv2.imshow('Rectangle on Image', src)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Результат:

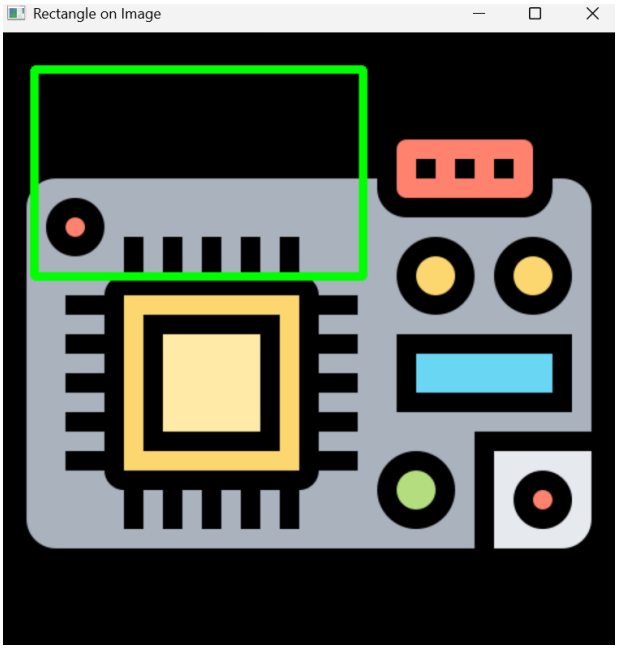

Рисунок 28 – Рисование на изображении