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 3](#_bookmark0)

[Ход работы: 3](#_bookmark1)

[Чтение и отображение изображения 3](#_bookmark2)

[Сохранение изображений 4](#_bookmark3)

[Вращение изображений 5](#_bookmark4)

[Вращение на 45 градусов 6](#_bookmark5)

[Изменение размера изображения 7](#_bookmark6)

[Цветовые пространства Python OpenCV. ЧБ изображение 8](#_bookmark7)

[Арифметические операции 9](#_bookmark8)

[Суммирование изображений 10](#_bookmark9)

[Вычитание изображений 10](#_bookmark10)

[Побитовые операции над двоичным изображением 11](#_bookmark11)

[Побитовое И 12](#_bookmark12)

[Побитовое ИЛИ 12](#_bookmark13)

[Побитовое XOR 13](#_bookmark14)

[Побитовое NOT 14](#_bookmark15)

[Смещение изображений Python OpenCV 15](#_bookmark16)

[Обнаружение границ 16](#_bookmark17)

[Определение порога 17](#_bookmark18)

[Адаптивное пороговое значение 18](#_bookmark19)

[Пороговое значение Otsu 19](#_bookmark20)

[Размытие изображений 20](#_bookmark21)

[Двусторонняя фильтрация 21](#_bookmark22)

[Контуры изображения 22](#_bookmark23)

[Размывание и расширение 23](#_bookmark24)

[Сопоставление функций 24](#_bookmark25)

[Рисование на изображениях 26](#_bookmark26)

# Задание:

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

# Ход работы:

Используемое изображение:

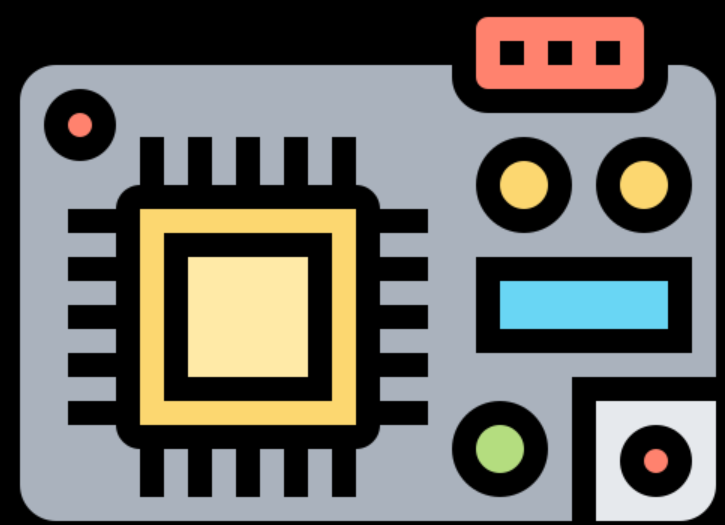


Рисунок 1 – Исходное изображение

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:

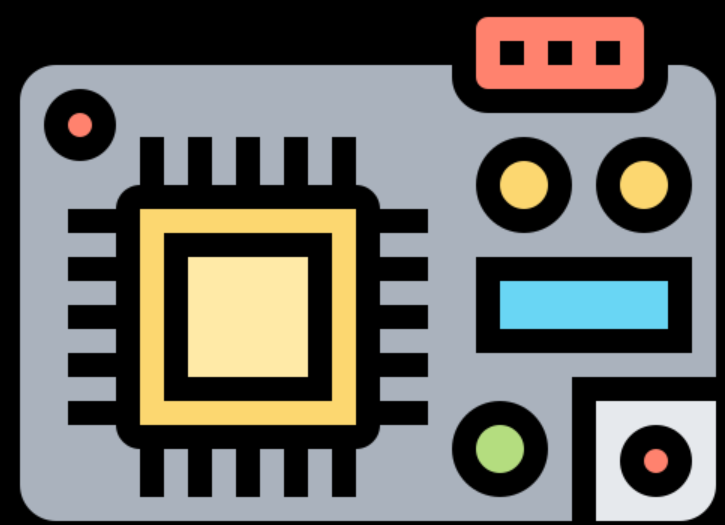


Рисунок 2 – Чтение и отображение

# 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

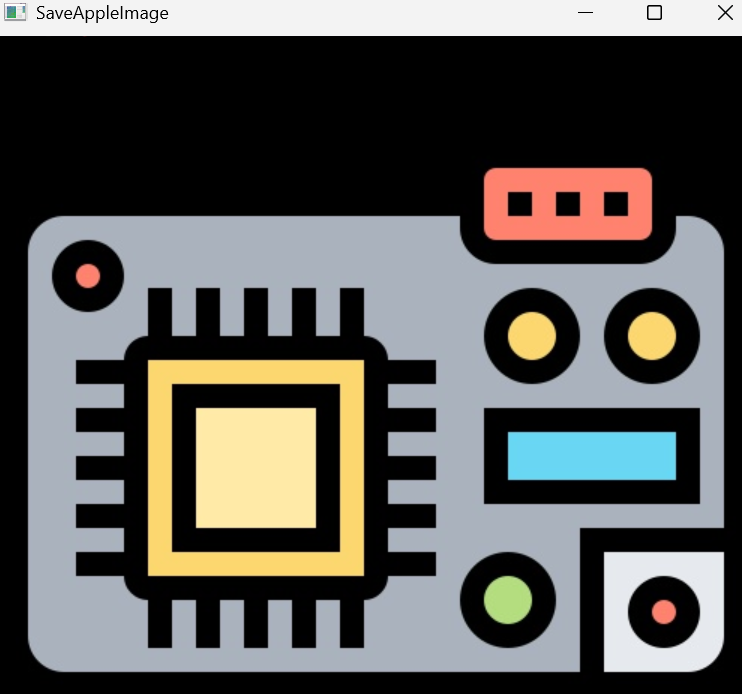


Рисунок 3 – Сохраненное изображение

# 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:

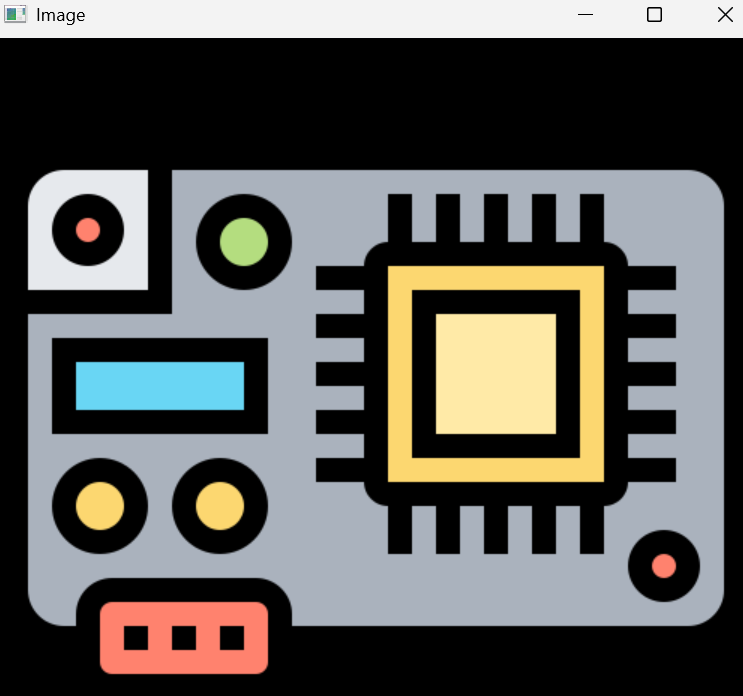


Рисунок 4 – Поворот на 180 градусов

# 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:

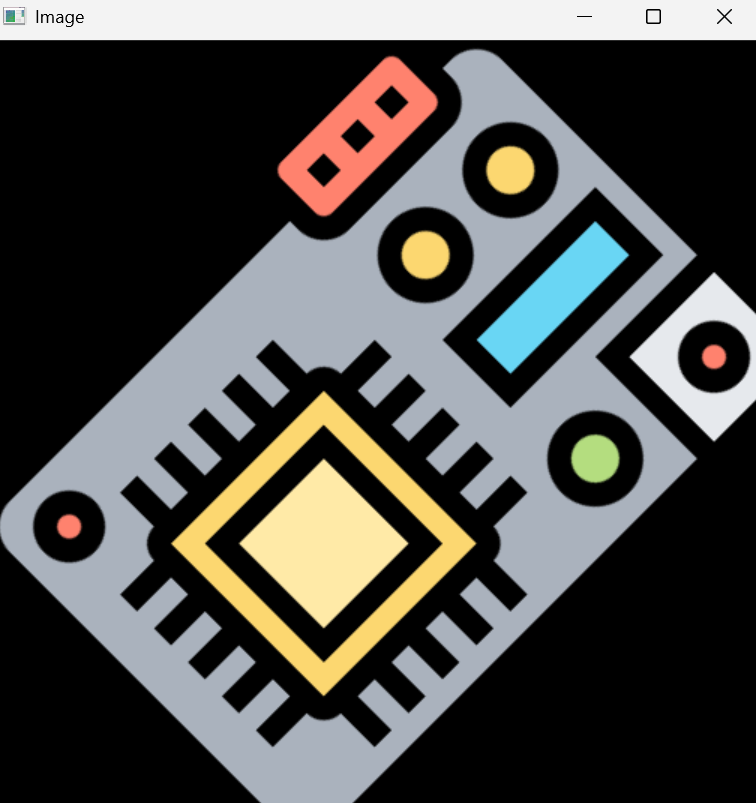


Рисунок 5 – Поворот на 45 градусов

# 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:

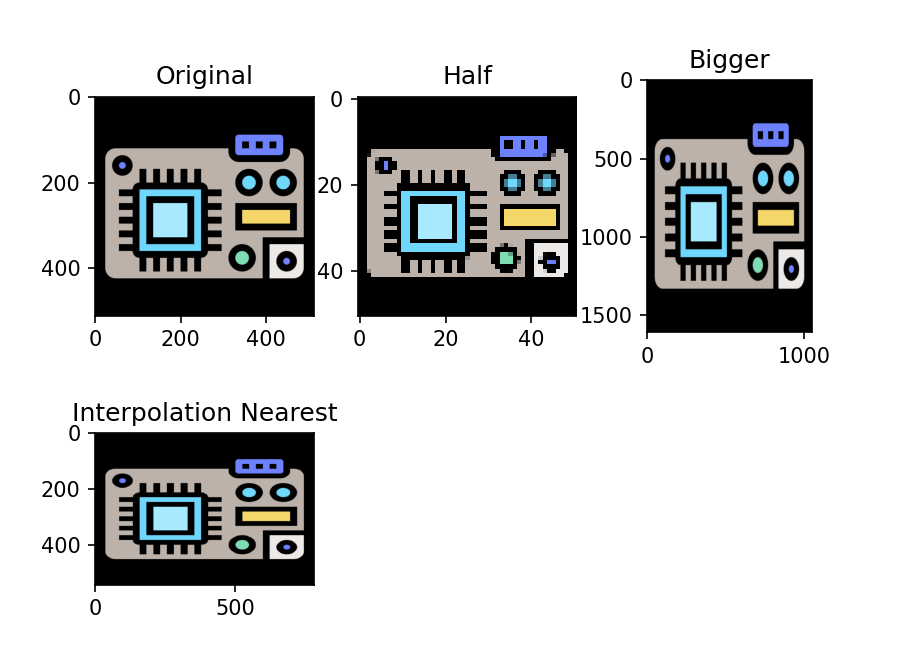


Рисунок 6 – Изменение размеров изображения

# 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:

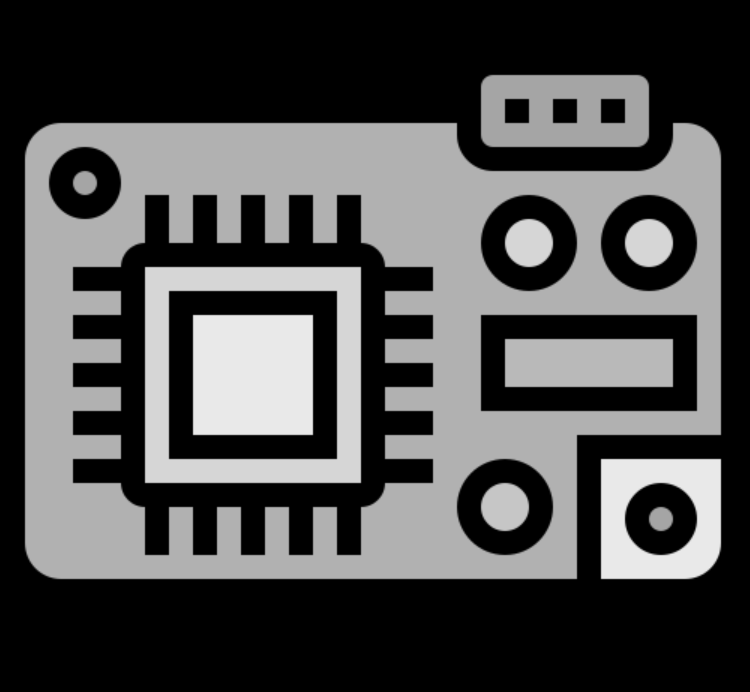


Рисунок 7 – Изменение цветовой гаммы изображения

# Арифметические операции

Ввод изображения 1:



Рисунок 8 – Изображение №1

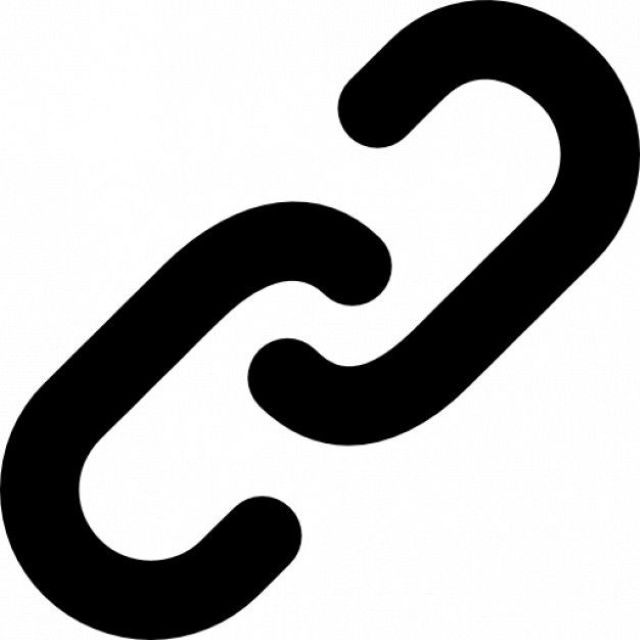
Ввод изображения 2

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

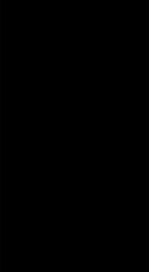


Рисунок 12 – Изображение №1

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:

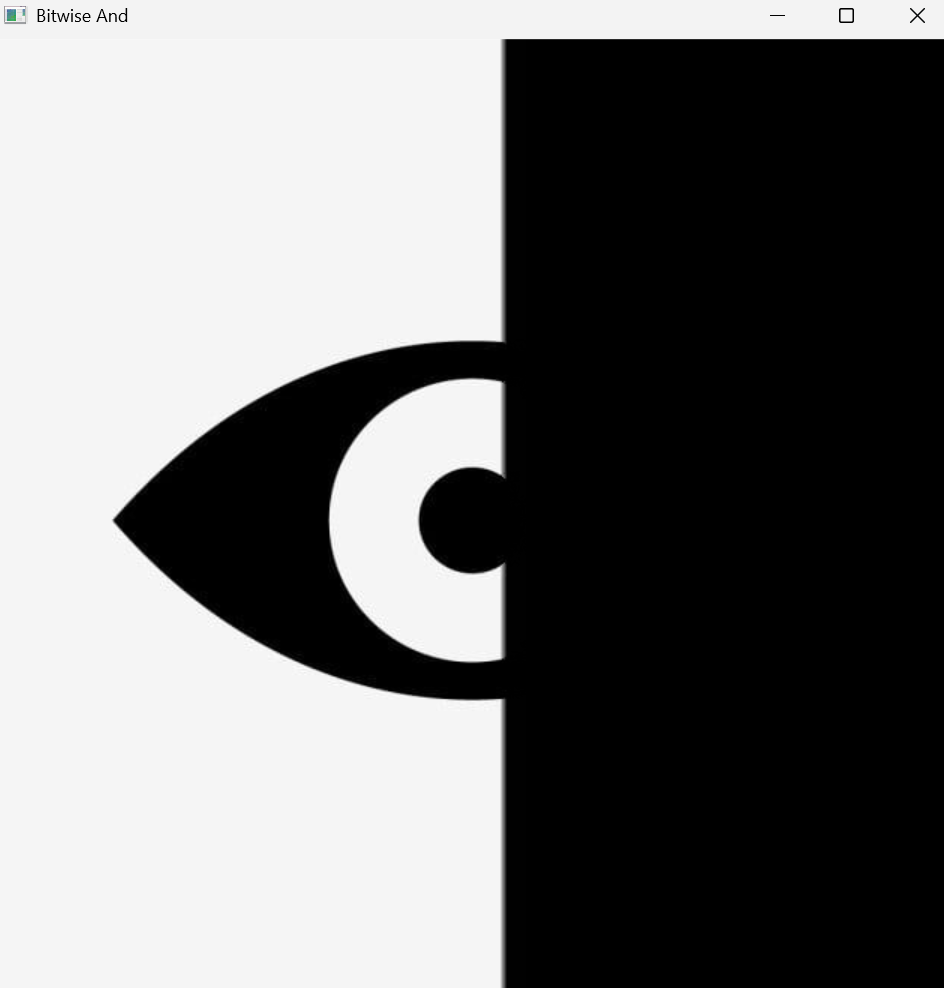


Рисунок 14 – Побитовое «И»

## 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:

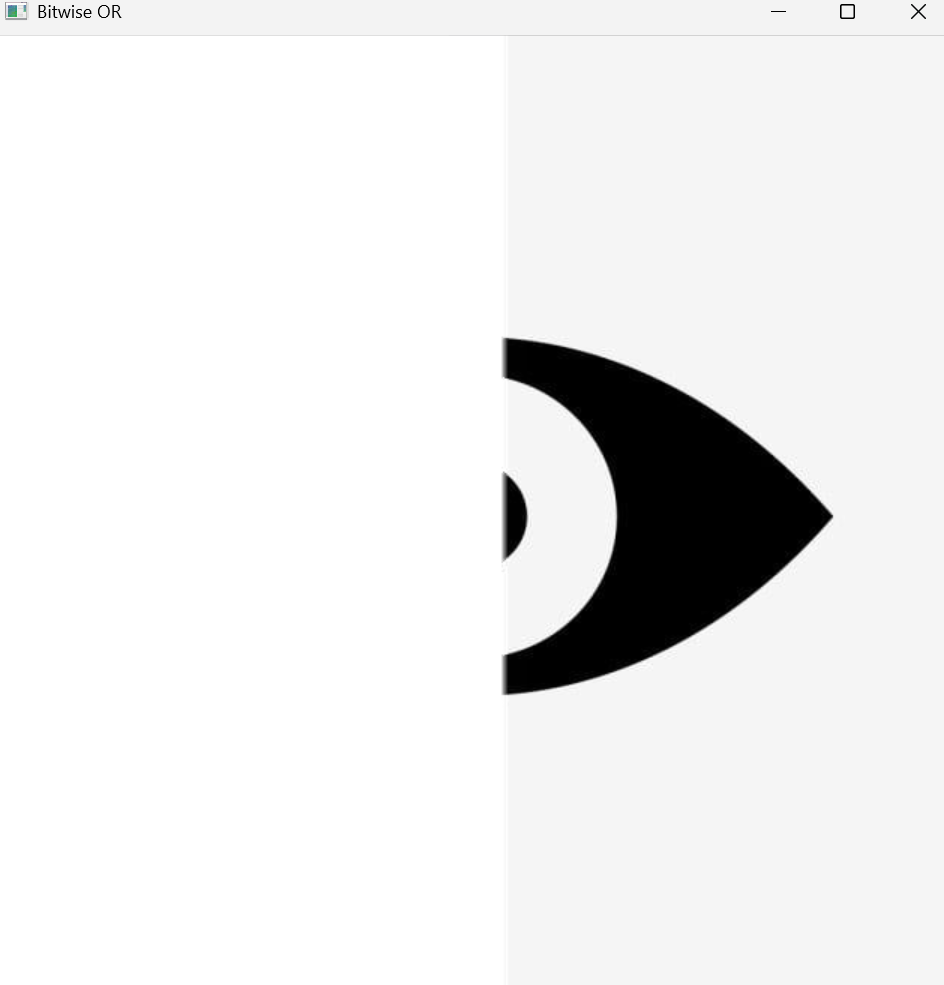


Рисунок 15 – Побитовое «ИЛИ»

## 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:

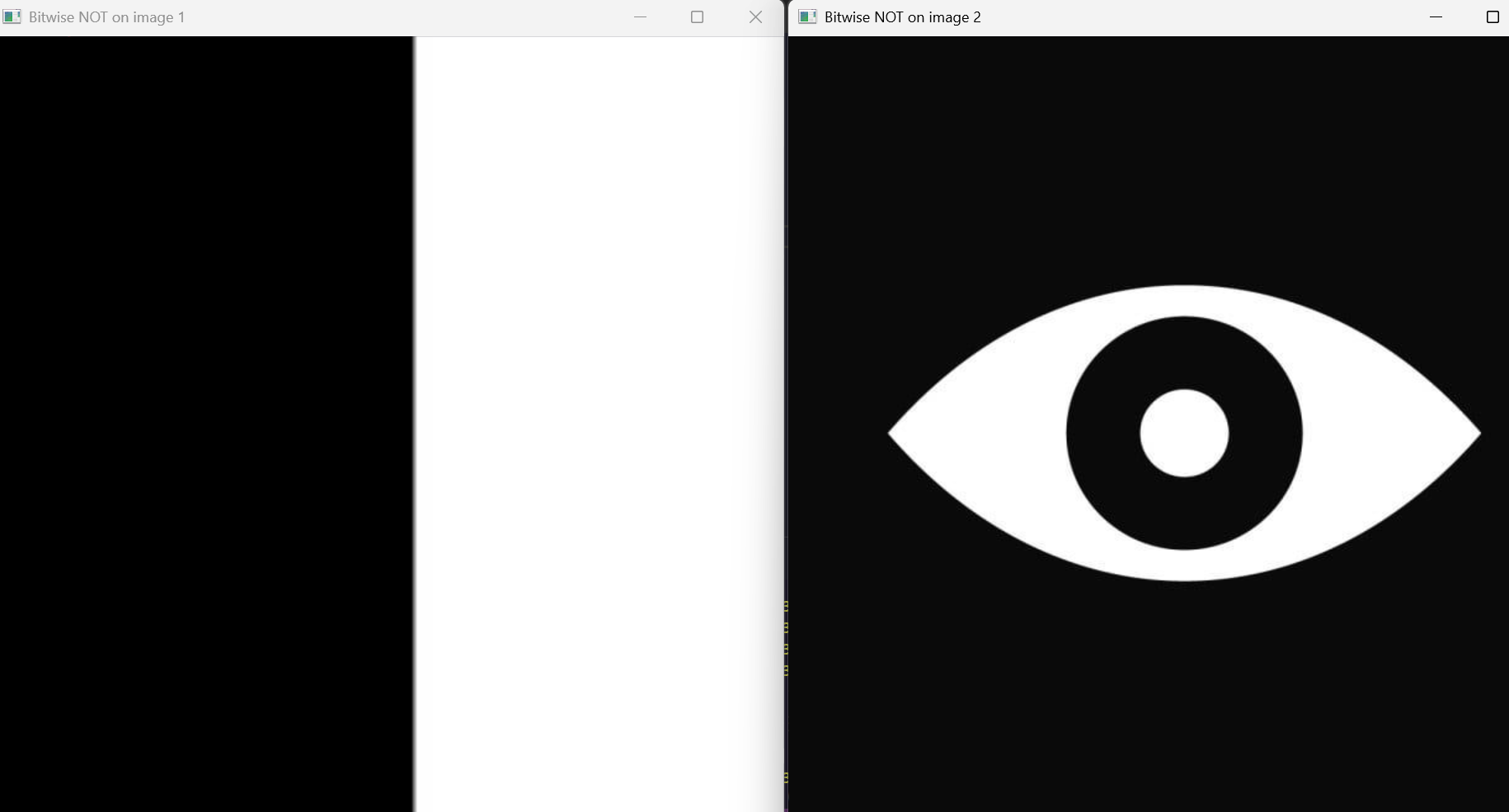


Рисунок 17 – Побитовое «NOT»

# 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:

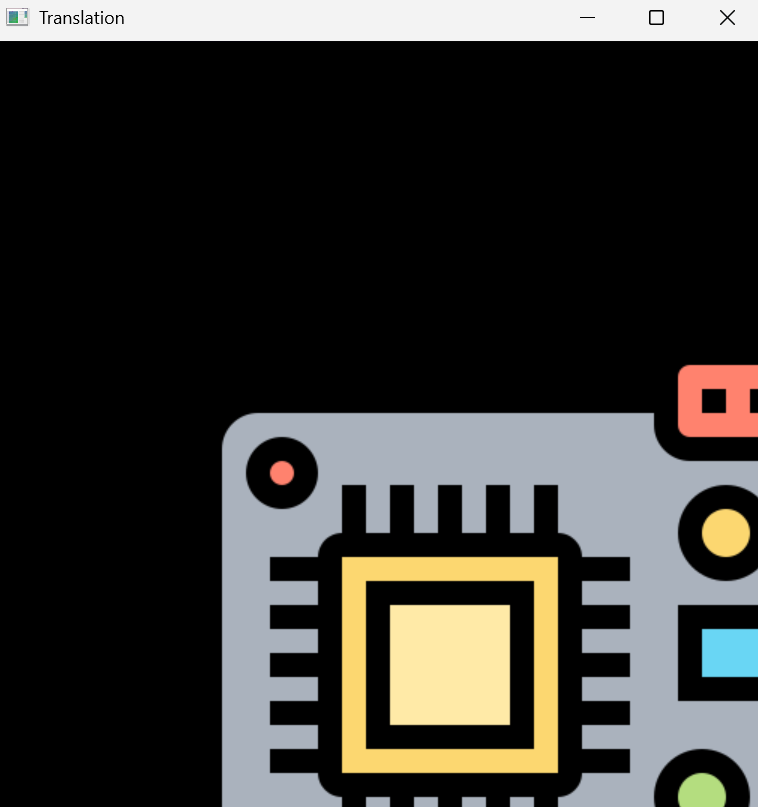


Рисунок 18 – Смещение изображения

# 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:

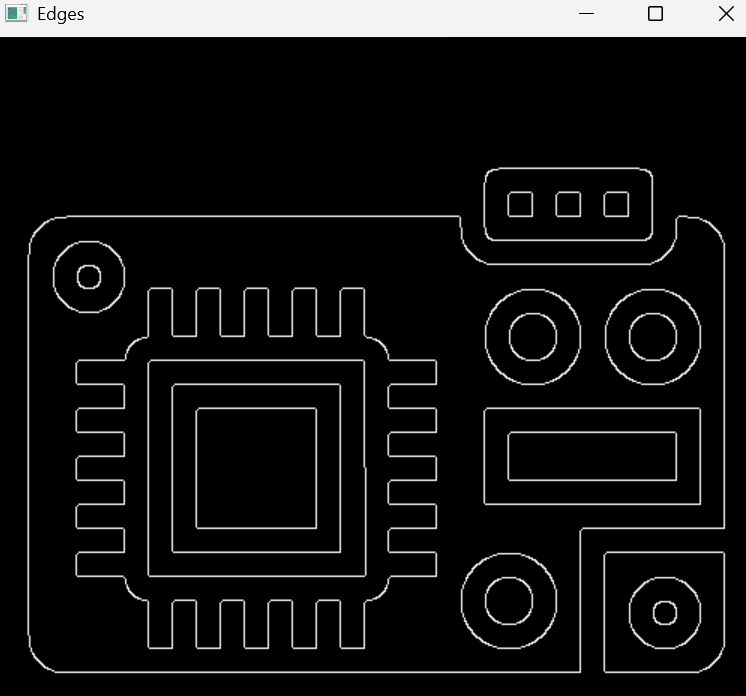


Рисунок 19 – Обнаружение границ изображения

# 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:

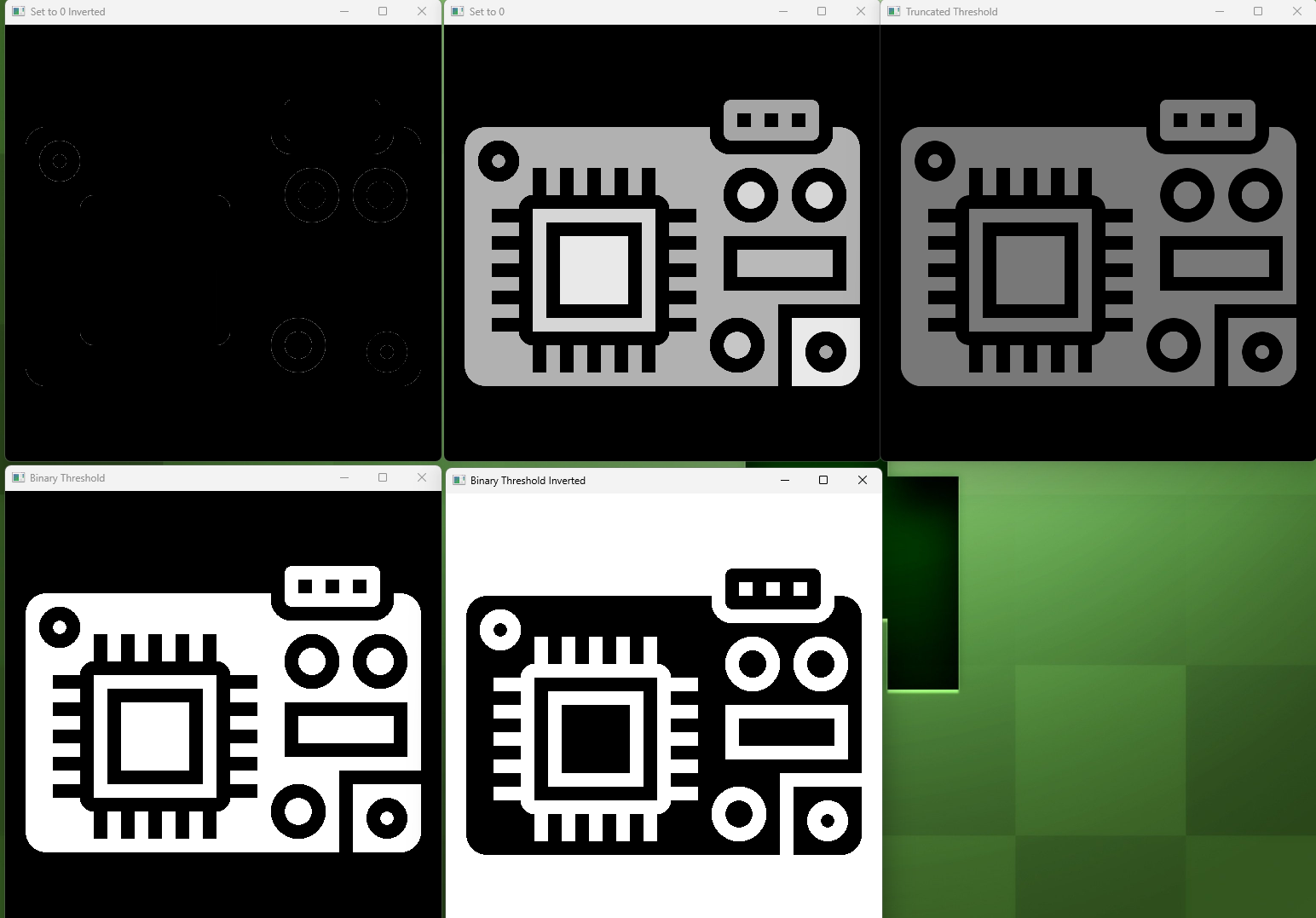


Рисунок 20 – Пороговая обработка изображения

# 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:

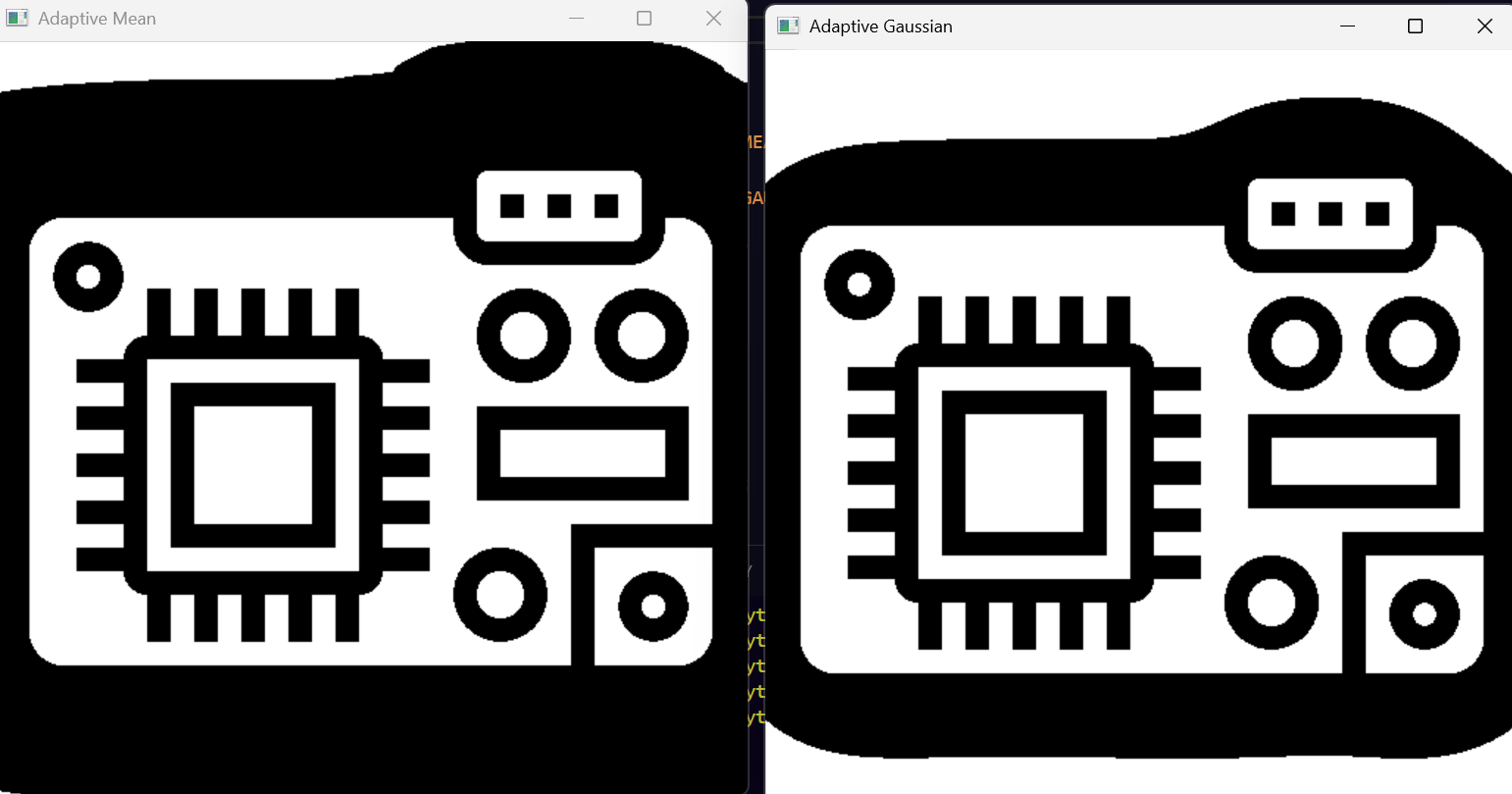


Рисунок 21 – Адаптивная пороговая обработка изображения

# 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:

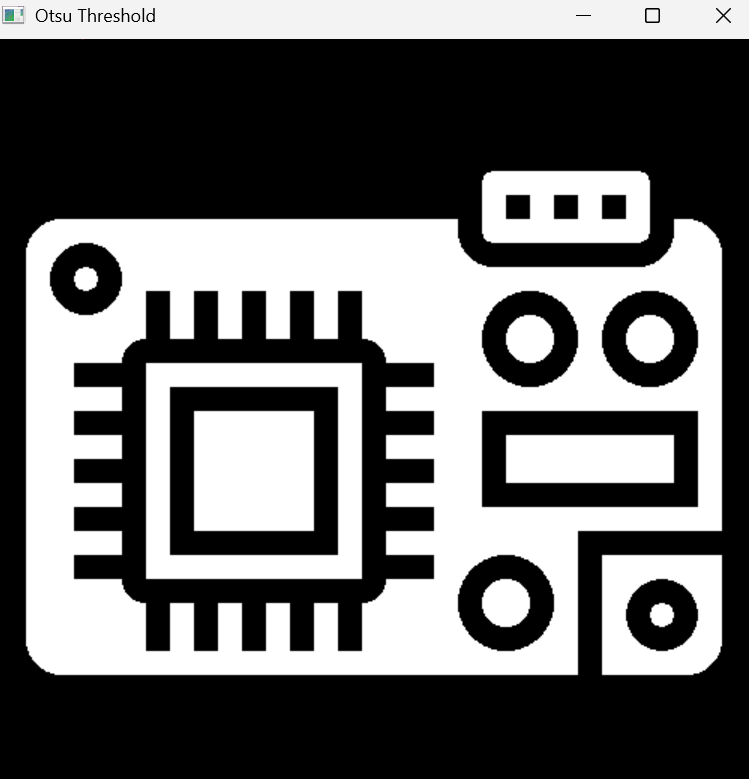


Рисунок 22 – Пороговое значение OTSU

# 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)

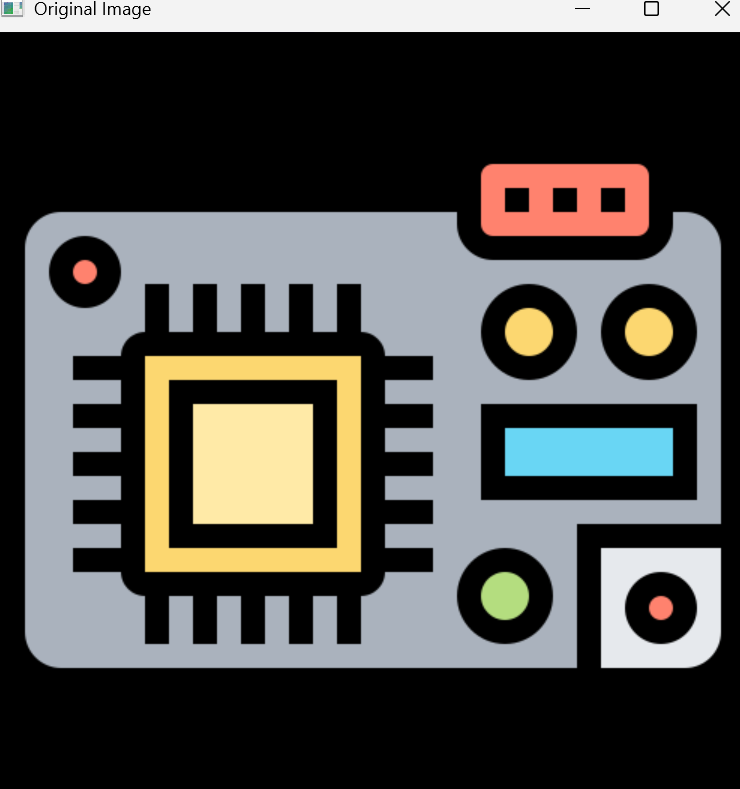
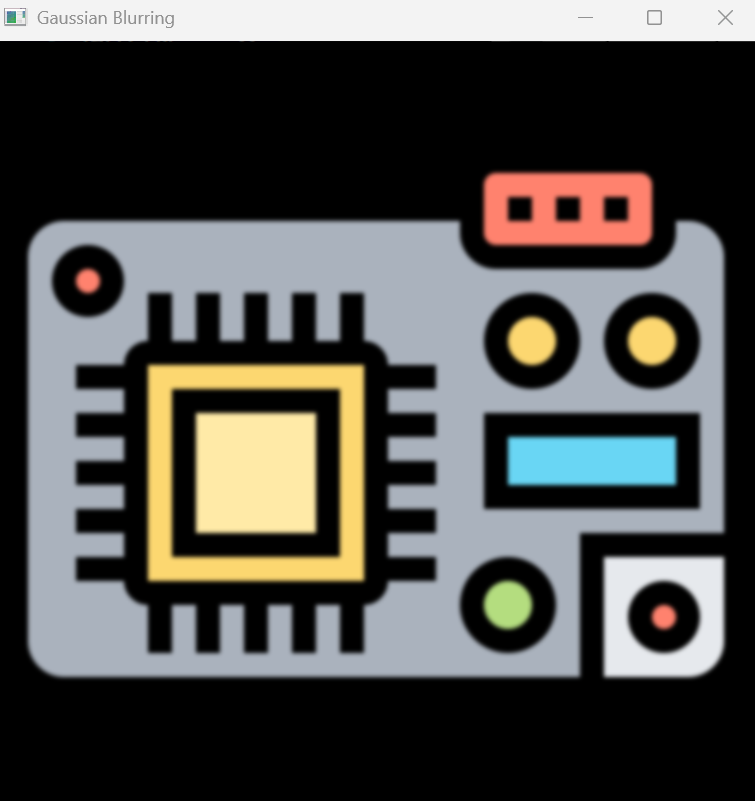
bilateral = cv2.bilateralFilter(image, 9, 75, 75)

cv2.imshow('Bilateral Blurring', bilateral)

cv2.waitKey(0)

cv2.destroyAllWindows()

Результат:

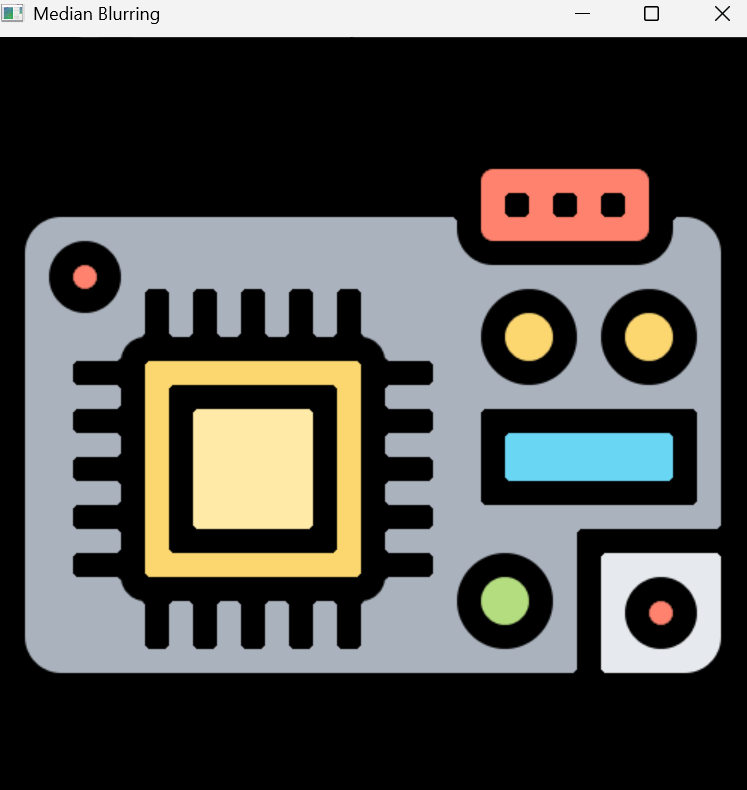
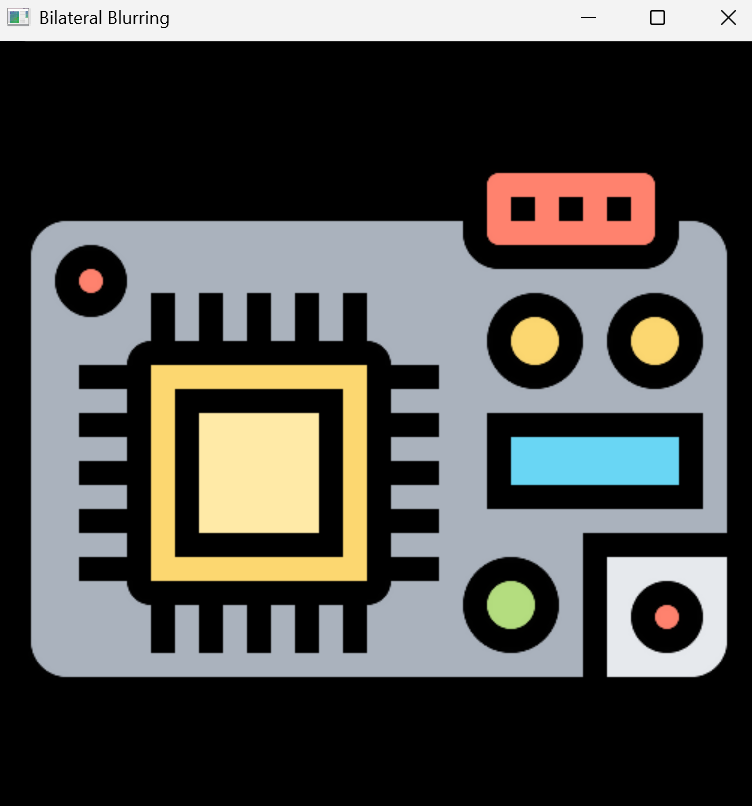
 

Рисунок 23 – Размытие изображения

# 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:

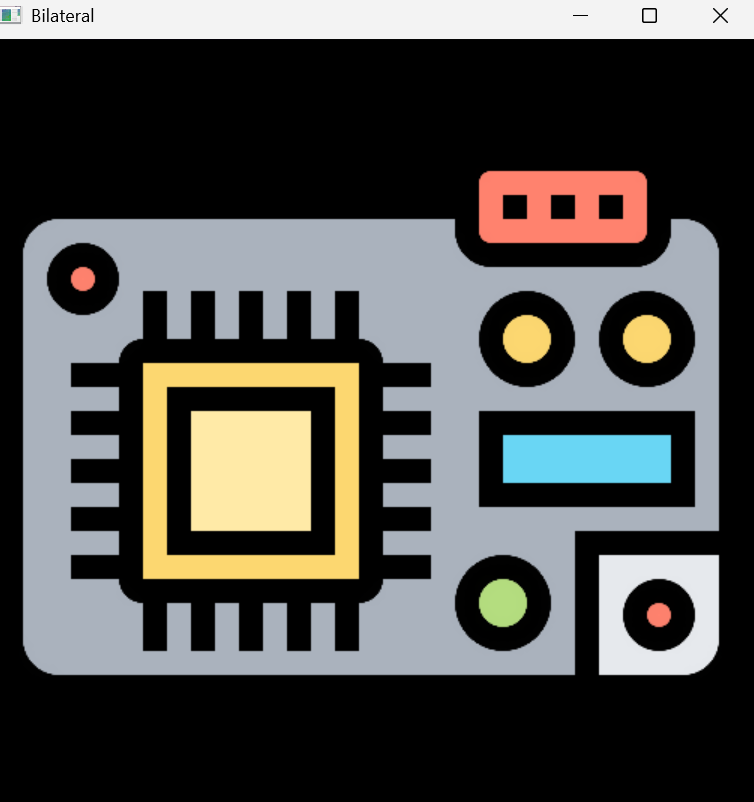


Рисунок 24 – Двусторонняя фильтрация

# 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:

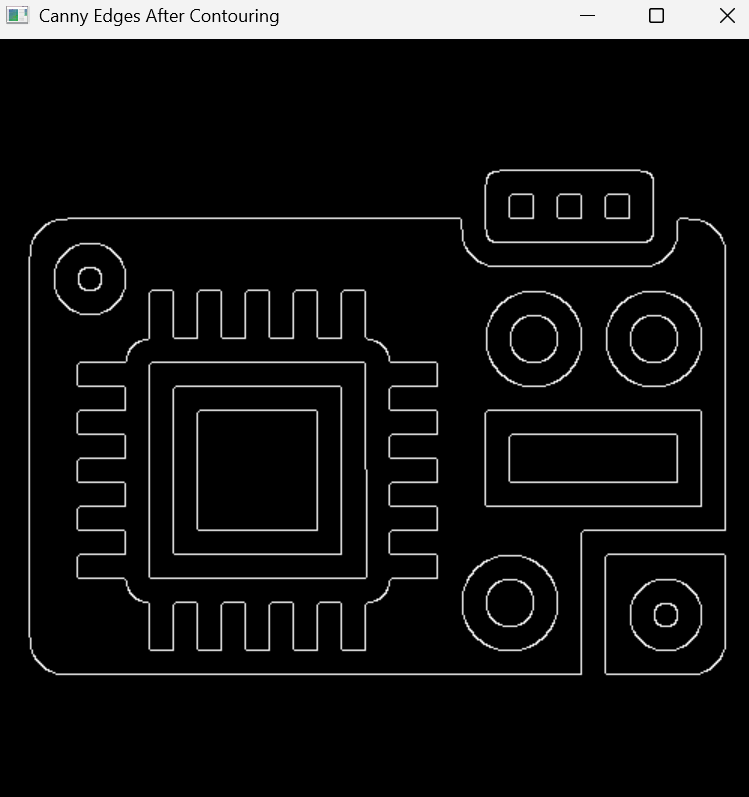
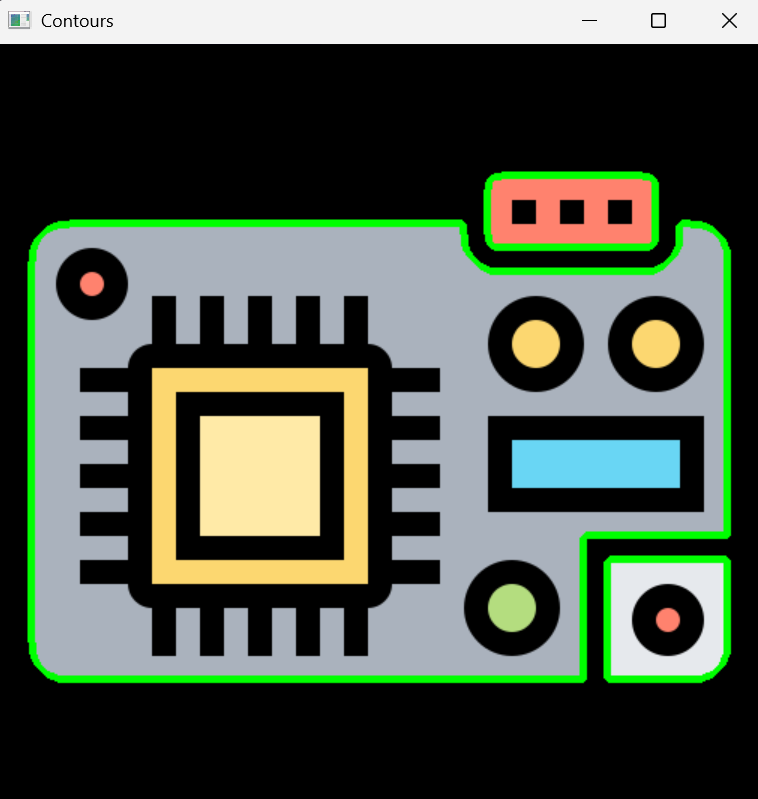
 

Рисунок 25 – Контуры изображения

# 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:

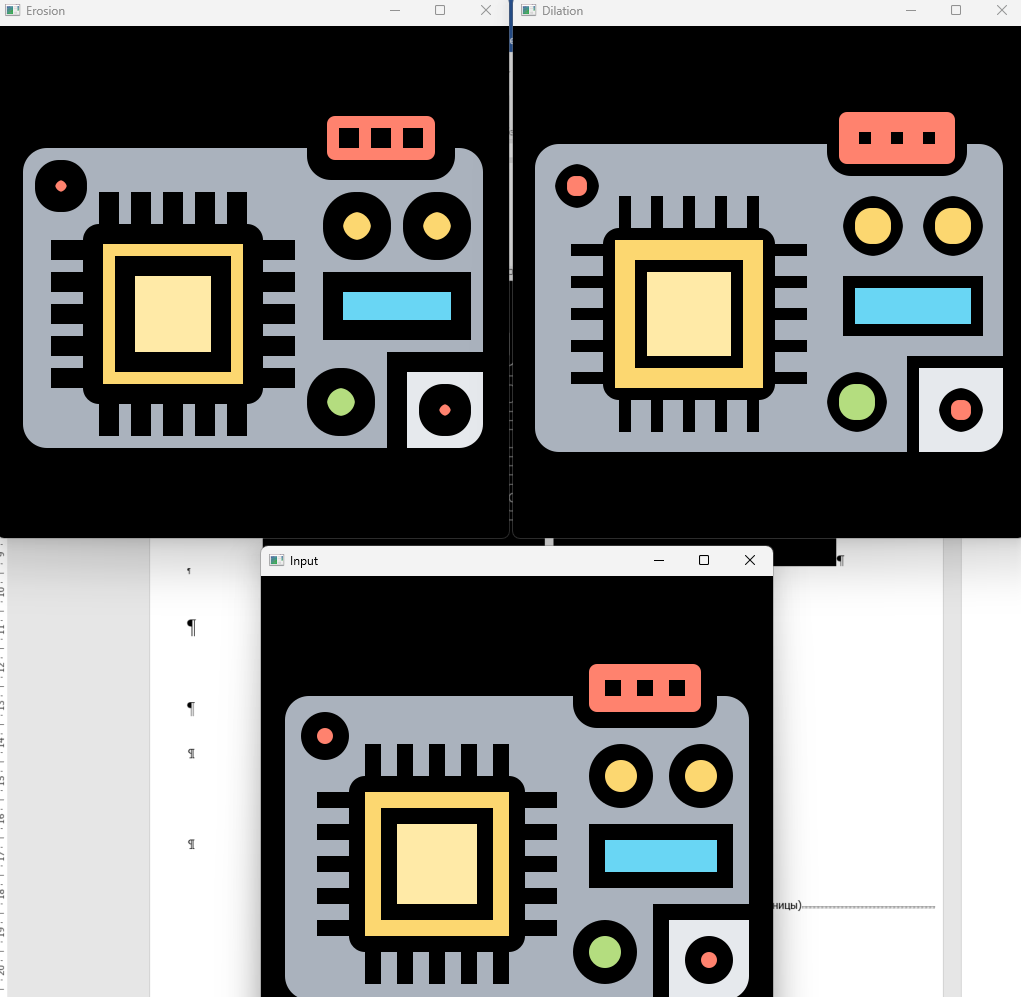


Рисунок 26 – Размывание и расширение

# 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:

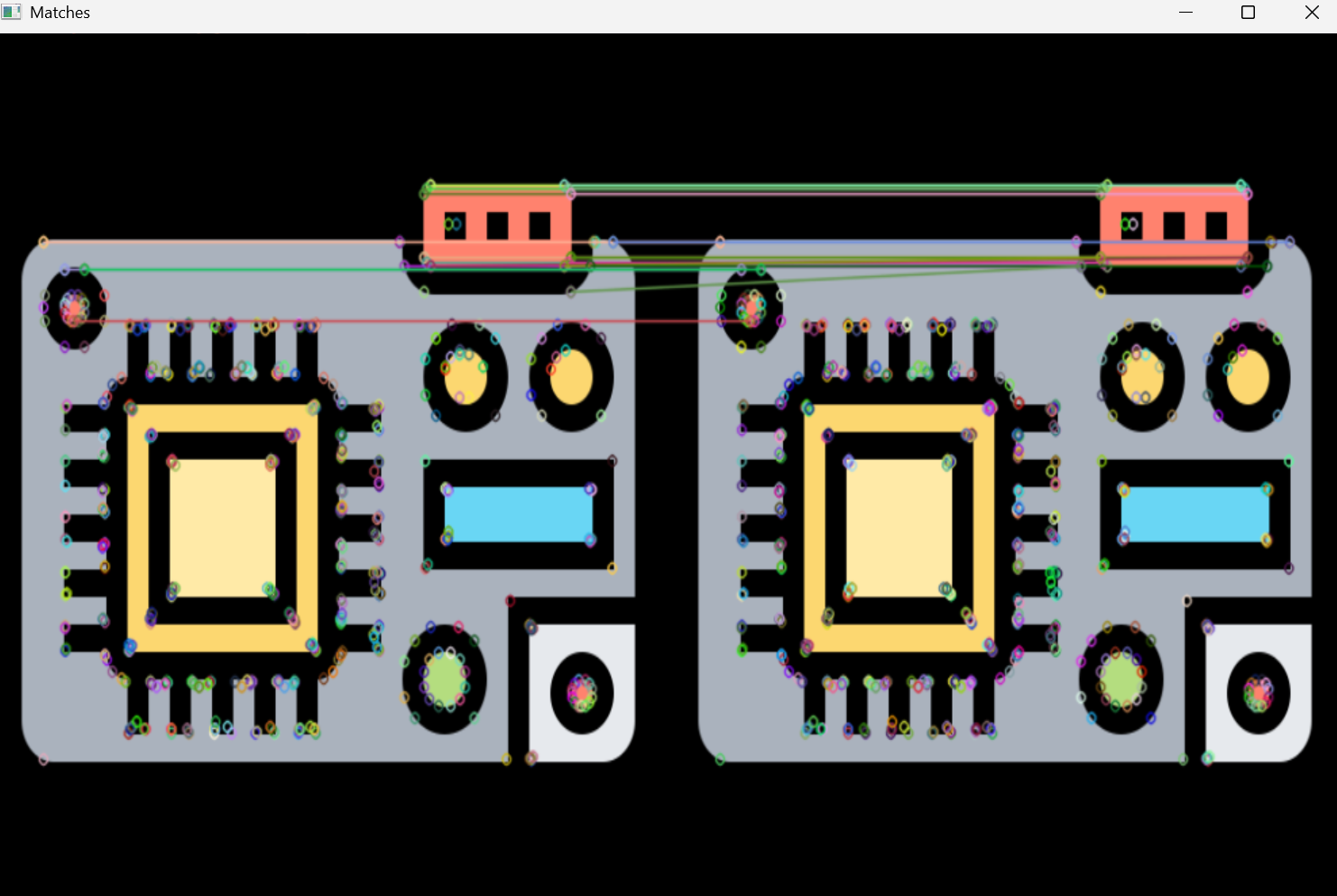


Рисунок 27 – Размывание и расширение

# 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:

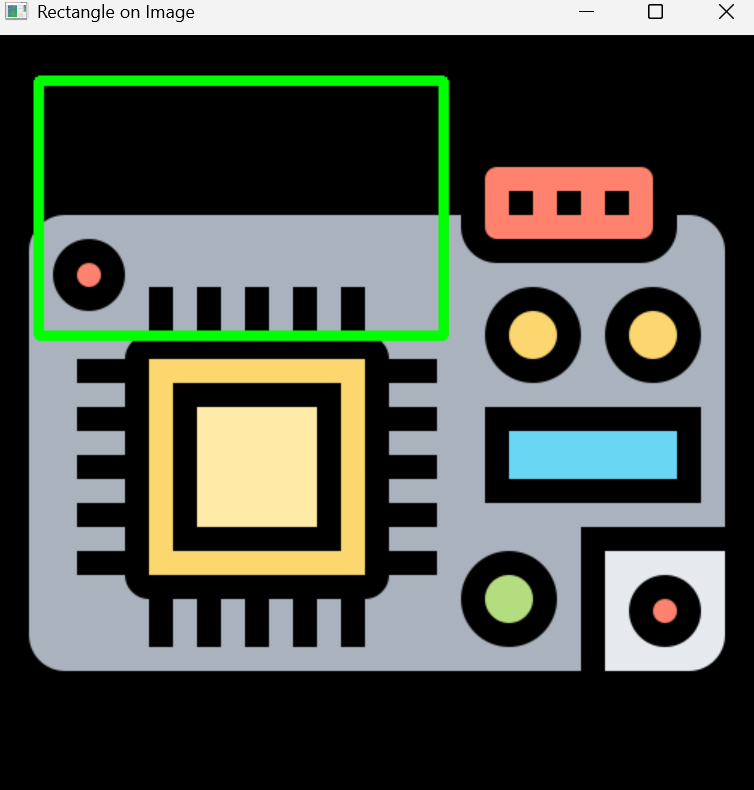


Рисунок 28 – Рисование на изображении