

# **INSTITUTO FEDERAL**

## Ceará

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ**

**IFCE *CAMPUS* FORTALEZA**

**PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

**MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO**

**FELIPE NOGUEIRA NERES**

**TRABALHO 1 - MÉTODOS BÁSICOS DE VISÃO COMPUTACIONAL**

**FORTALEZA**

**2024**

FELIPE NOGUEIRA NERES

TRABALHO 1 - MÉTODOS BÁSICOS DE VISÃO COMPUTACIONAL

Trabalho apresentado ao curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do *Campus* Fortaleza do Instituto Federal de Educação, Ciência e Tecnologia do Ceará

Orientador: Prof. Dr. Pedro Pedrosa Rebouças Filho.

FORTALEZA

2024

## **RESUMO**

O presente trabalho visa a reprodução das operações e técnicas básicas em Processamento Digital de Imagens, utilizando como base os conhecimentos obtidos nas aulas da disciplina e o livro Processamento Digital de Imagens, 3ª Edição, de Woods e González.

**Palavras-chave:** Processamento. Imagens. Filtros.

## **ABSTRACT**

The present work aims to reproduce the basic operations and techniques in Digital Image Processing, using as a basis the knowledge obtained in the subject classes and in the book Digital Image Processing, 3rd Edition, by Woods and González.

**Keywords:** Processing. Image. Filters.

## LISTA DE FIGURAS

Figura 1 – Questões 1 e 2 (Código) . . . . .	7
Figura 2 – Questões 1 e 2 (Resultado) . . . . .	8
Figura 3 – Questão 4 (Código) . . . . .	9
Figura 4 – Questão 4 (Resultado) . . . . .	10
Figura 5 – Questão 6 (Código) . . . . .	11
Figura 6 – Questão 6 (Resultado) . . . . .	12
Figura 7 – Questão 8 (Código) . . . . .	13
Figura 8 – Questão 8 (Resultado) . . . . .	14
Figura 9 – Questão 9 (Código) . . . . .	15
Figura 10 – Questão 9 (Resultado) . . . . .	16

## SUMÁRIO

1	INTRODUÇÃO . . . . .	5
2	MÉTODOS BÁSICOS DE VISÃO COMPUTACIONAL . . . . .	6
3	TRABALHO 1 - QUESTÕES 1 E 2 . . . . .	7
4	TRABALHO 1 - QUESTÃO 4 . . . . .	9
5	TRABALHO 1 - QUESTÃO 6 . . . . .	11
6	TRABALHO 1 - QUESTÃO 8 . . . . .	13
7	TRABLHO 1 - QUESTÃO 9 . . . . .	15
8	CONCLUSÃO . . . . .	17

## 1 INTRODUÇÃO

O processamento digital de imagens desempenha um papel fundamental em uma variedade de campos, desde medicina até entretenimento, fornecendo ferramentas poderosas para a análise, melhoria e compreensão de imagens digitais. Neste contexto, este trabalho de mestrado explora as operações básicas do processamento digital de imagens, visando aprimorar e transformar imagens digitais de diversas fontes. Compreender as operações fundamentais é essencial para construir uma base sólida neste campo. Este estudo concentra-se em explorar e aplicar uma série de operações básicas, incluindo filtragem, segmentação, morfologia matemática e transformações geométricas. Através da aplicação dessas técnicas, o objetivo é melhorar a qualidade visual das imagens, extrair informações relevantes e prepará-las para análises mais avançadas.

Além disso, este trabalho discute a importância do processamento digital de imagens em várias aplicações do mundo real, desde o diagnóstico médico até a automação industrial. Ao entender as operações básicas e suas aplicações, os profissionais deste campo podem contribuir significativamente para o avanço de tecnologias e soluções em uma ampla gama de setores.

Por meio da aplicação prática das operações básicas, este trabalho busca fornecer insights valiosos sobre os princípios subjacentes do processamento digital de imagens, bem como suas aplicações potenciais e limitações. Ao final, espera-se que este estudo ofereça uma base sólida para futuras pesquisas e desenvolvimentos neste campo dinâmico e em constante evolução.

## 2 MÉTODOS BÁSICOS DE VISÃO COMPUTACIONAL

O processamento digital de imagens envolve uma série de operações básicas e filtros que são aplicados a imagens digitais para melhorar sua qualidade, extrair informações relevantes ou prepará-las para análises posteriores. Essas operações desempenham um papel crucial em diversas áreas, incluindo medicina, processamento de imagem e vídeo, automação industrial e muitas outras.

Os filtros espaciais são utilizados para modificar a aparência de uma imagem digital, geralmente aplicando-se uma transformação local a cada pixel da imagem. Isso pode incluir operações como suavização (para reduzir ruídos), realce de bordas (para destacar transições de intensidade) e nitidez (para aumentar a definição da imagem).

Abaixo, segue a lista de exercícios cujo objetivo é aplicar os métodos solicitados.

### **Filtros passa-baixa:**

- Média [Questão 1]
- Mediana [Questão 2]
- Gaussiano [Questão 3]

### **•Filtros passa-alta:**

- Laplaciano [Questão 4]
- Prewit [Questão 5]
- Sobel [Questão 6]

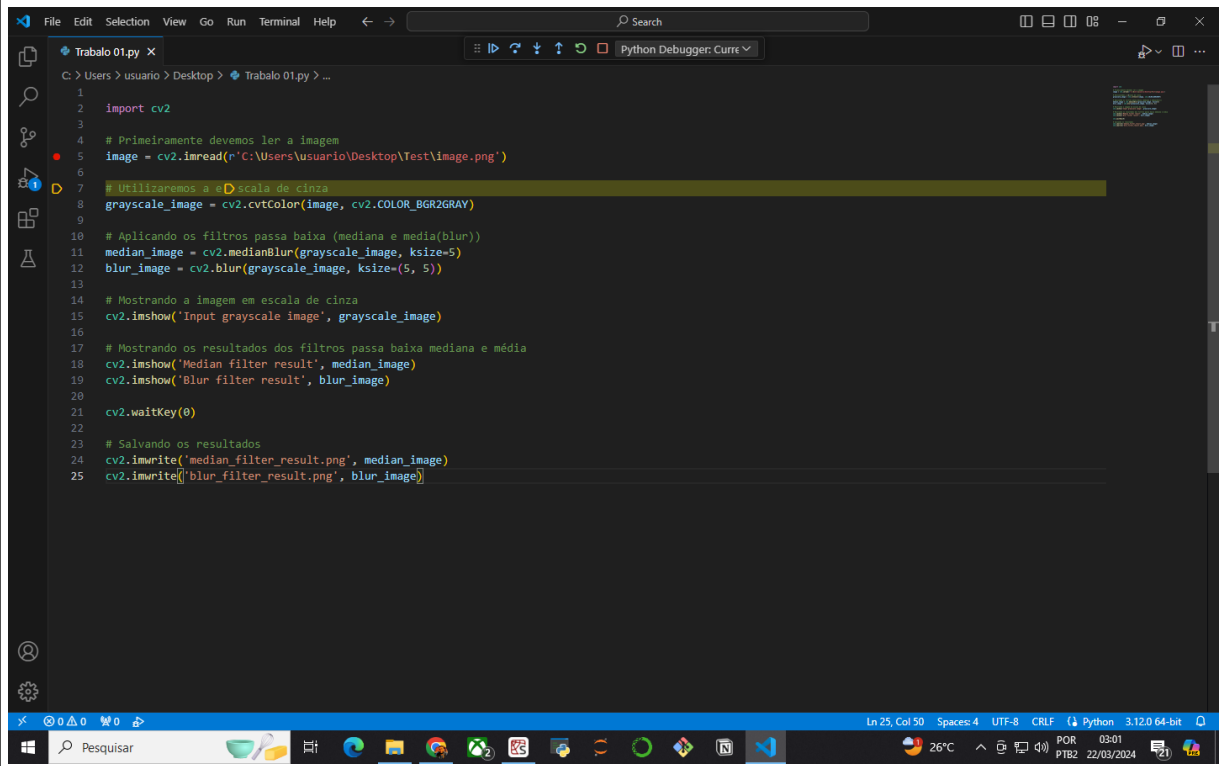
### **Outras operações**

- Cálculo e apresentação do histograma [Questão 7]
- Equalização do histograma [Questão 8]
- Limiarização [Questão 9]
- Multilimiarização [Questão 10]



### 3 TRABALHO 1 - QUESTÕES 1 E 2

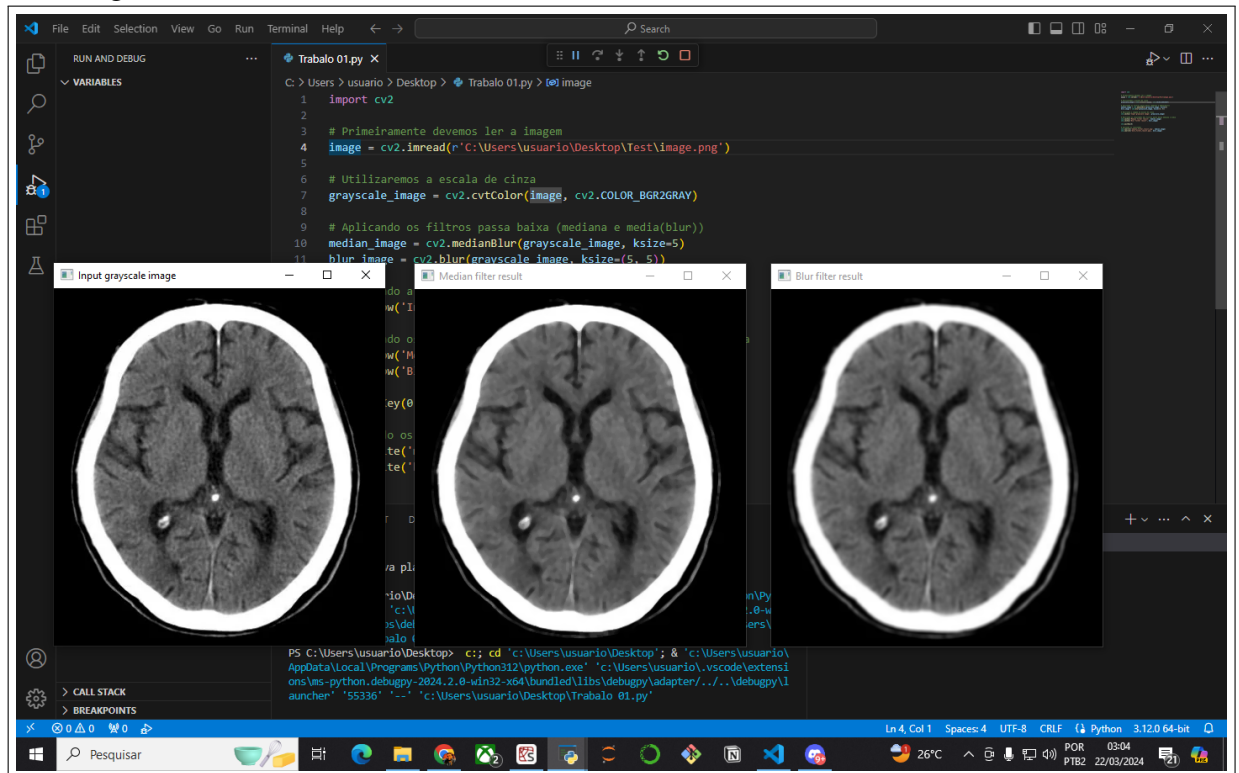
Figura 1 – Questões 1 e 2 (Código)



```
1 import cv2
2
3 # Primeiramente devemos ler a imagem
4 image = cv2.imread('C:\Users\usuario\Desktop\Test\image.png')
5
6 # Utilizaremos a escala de cinza
7 grayscale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
8
9 # Aplicando os filtros passa baixa (mediana e media(blur))
10 median_image = cv2.medianBlur(grayscale_image, ksize=5)
11 blur_image = cv2.blur(grayscale_image, ksize=(5, 5))
12
13 # Mostrando a imagem em escala de cinza
14 cv2.imshow('Input grayscale image', grayscale_image)
15
16 # Mostrando os resultados dos filtros passa baixa mediana e média
17 cv2.imshow('Median filter result', median_image)
18 cv2.imshow('Blur filter result', blur_image)
19
20 cv2.waitKey(0)
21
22 # Salvando os resultados
23 cv2.imwrite('median_filter_result.png', median_image)
24 cv2.imwrite('blur_filter_result.png', blur_image)
```

Fonte: Elaborado pelo autor

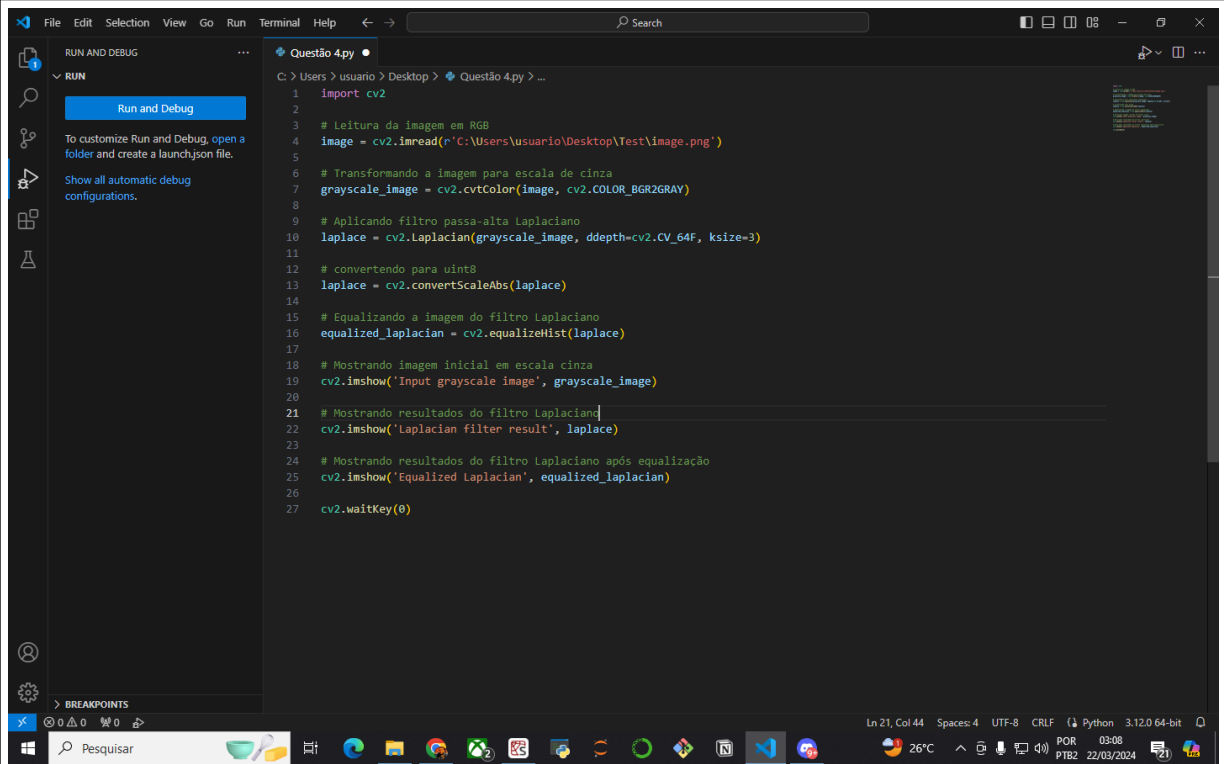
Figura 2 – Questões 1 e 2 (Resultado)



Fonte: Elaborado pelo autor

## 4 TRABALHO 1 - QUESTÃO 4

Figura 3 – Questão 4 (Código)

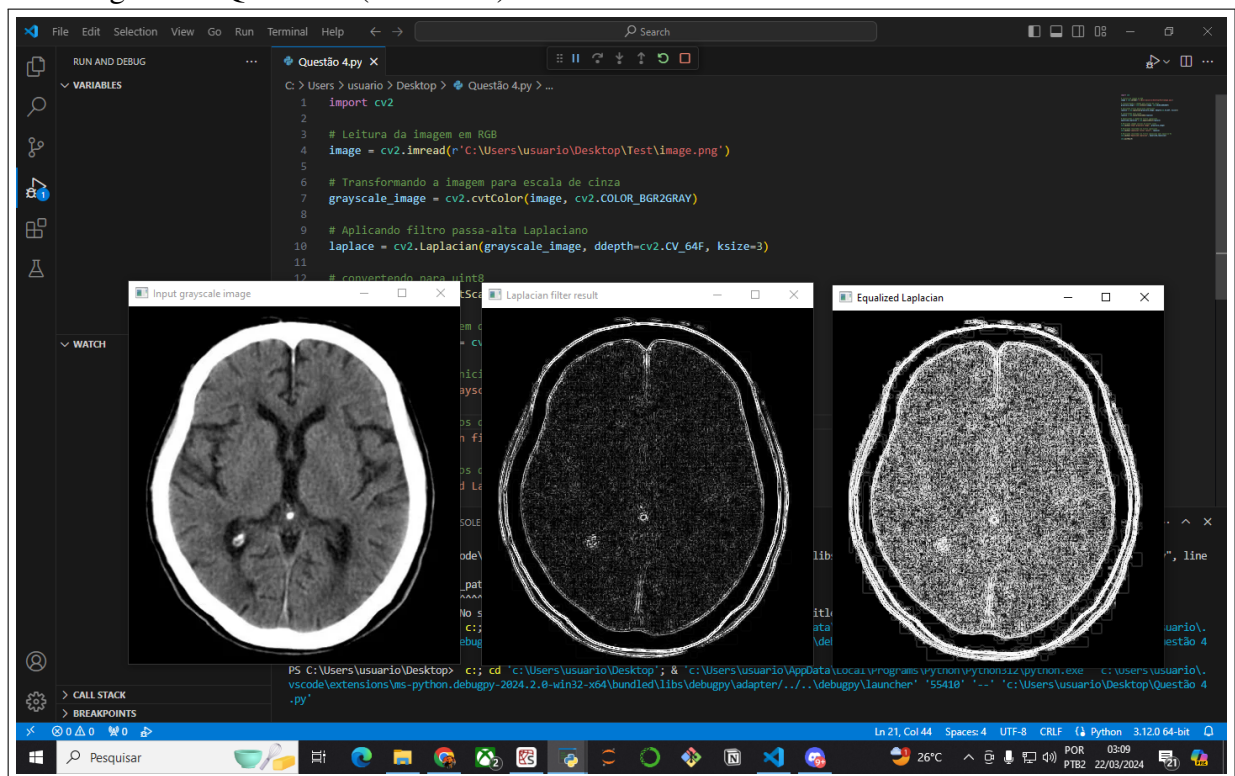


```
1 import cv2
2
3 # Leitura da imagem em RGB
4 image = cv2.imread(r'C:\Users\usuario\Desktop\Test\image.png')
5
6 # Transformando a imagem para escala de cinza
7 grayscale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
8
9 # Aplicando filtro passa-alta Laplaciano
10 laplace = cv2.Laplacian(grayscale_image, ddepth=cv2.CV_64F, ksize=3)
11
12 # convertendo para uint8
13 laplace = cv2.convertScaleAbs(laplace)
14
15 # Equalizando a imagem do filtro Laplaciano
16 equalized_laplace = cv2.equalizeHist(laplace)
17
18 # Mostrando imagem inicial em escala cinza
19 cv2.imshow('Input grayscale image', grayscale_image)
20
21 # Mostrando resultados do filtro Laplaciano
22 cv2.imshow('Laplacian filter result', laplace)
23
24 # Mostrando resultados do filtro Laplaciano após equalização
25 cv2.imshow('Equalized Laplacian', equalized_laplace)
26
27 cv2.waitKey(0)
```

The screenshot shows the Visual Studio Code interface. On the left, the 'RUN AND DEBUG' sidebar is visible with a 'Run and Debug' button and instructions. The main editor area displays the Python code for 'Questão 4.py'. The code performs the following steps: imports cv2, reads an image from 'C:\Users\usuario\Desktop\Test\image.png', converts it to grayscale using cv2.cvtColor, applies a Laplacian filter using cv2.Laplacian with ddepth=cv2.CV\_64F and ksize=3, converts the result to uint8 using cv2.convertScaleAbs, equalizes the histogram using cv2.equalizeHist, and displays three images using cv2.imshow: the input grayscale image, the Laplacian filter result, and the equalized Laplacian result. The script ends with cv2.waitKey(0). The status bar at the bottom indicates the file is at line 21, column 44, with 4 spaces, UTF-8 encoding, CRLF line endings, and is using Python 3.12.0 64-bit.

Fonte: Elaborado pelo autor

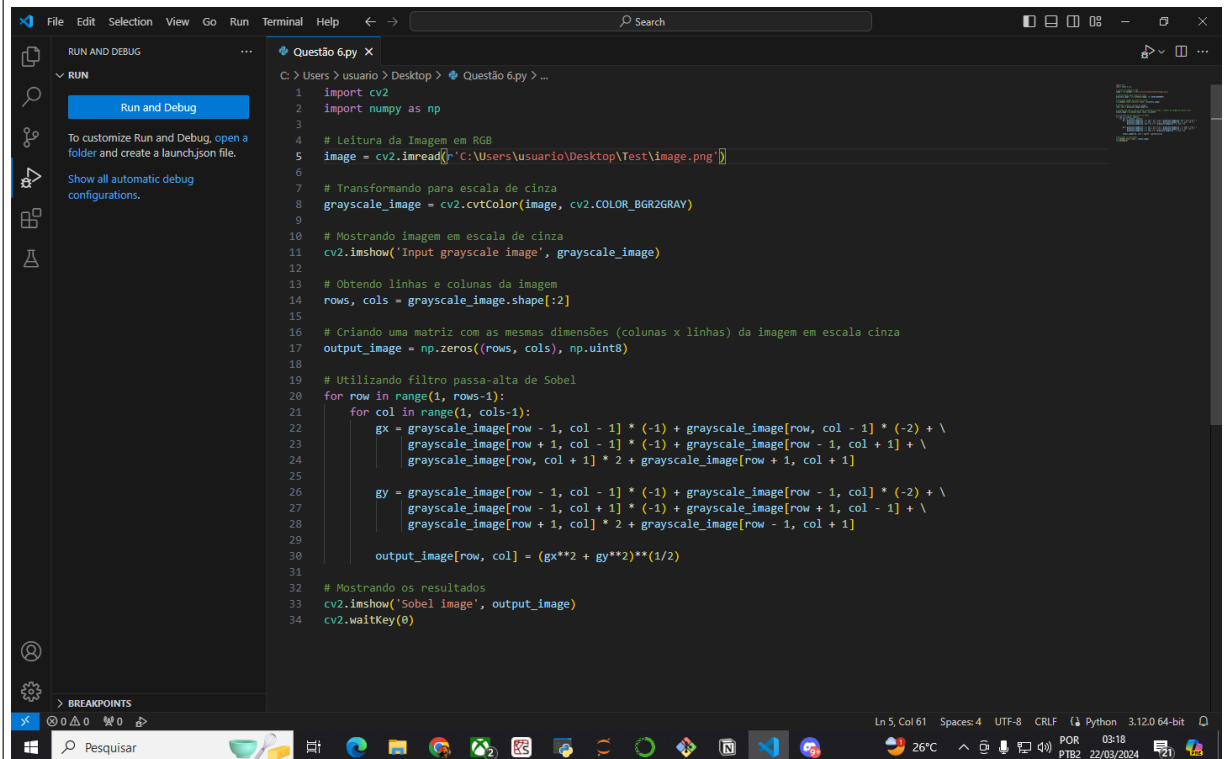
Figura 4 – Questão 4 (Resultado)



Fonte: Elaborado pelo autor

## 5 TRABALHO 1 - QUESTÃO 6

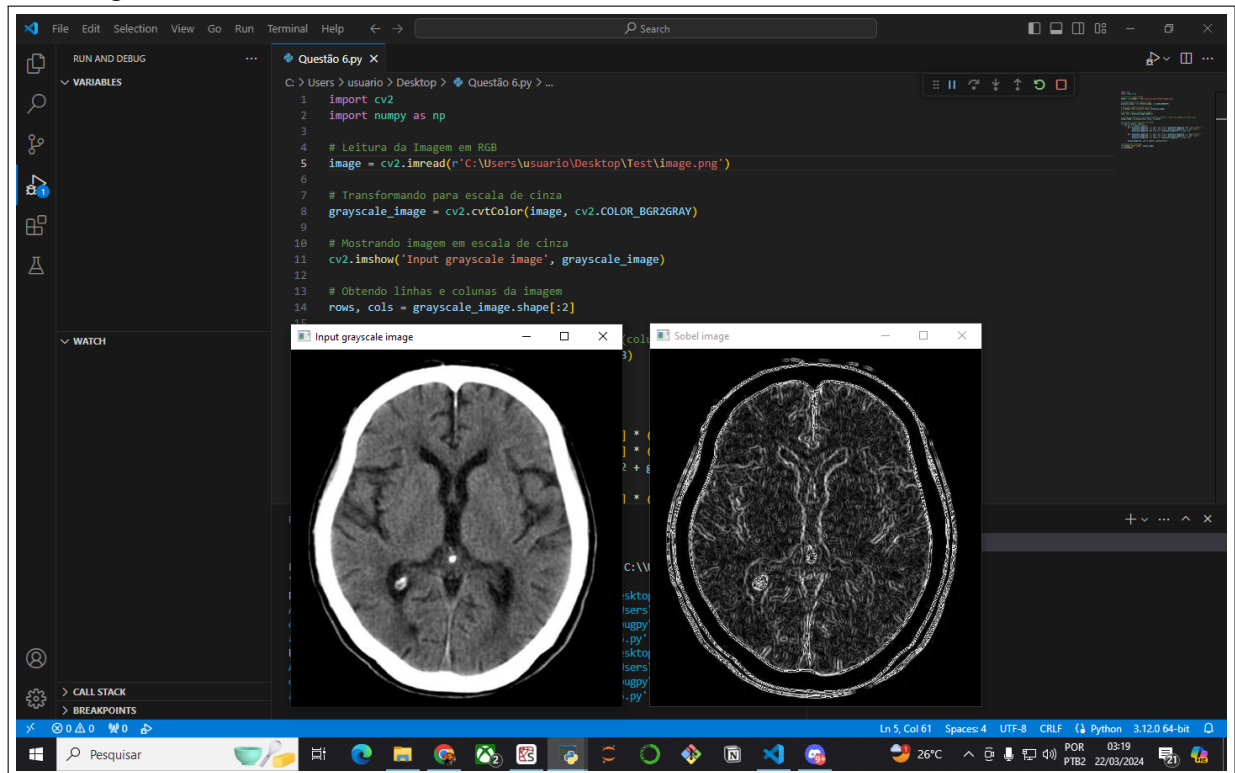
Figura 5 – Questão 6 (Código)



```
1 import cv2
2 import numpy as np
3
4 # Leitura da Imagem em RGB
5 image = cv2.imread(r"C:\Users\usuario\Desktop\Test\image.png")
6
7 # Transformando para escala de cinza
8 grayscale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
9
10 # Mostrando imagem em escala de cinza
11 cv2.imshow('Input grayscale image', grayscale_image)
12
13 # Obtendo linhas e colunas da imagem
14 rows, cols = grayscale_image.shape[:2]
15
16 # Criando uma matriz com as mesmas dimensões (colunas x linhas) da imagem em escala cinza
17 output_image = np.zeros((rows, cols), np.uint8)
18
19 # Utilizando filtro passa-alta de Sobel
20 for row in range(1, rows-1):
21     for col in range(1, cols-1):
22         gx = grayscale_image[row - 1, col - 1] * (-1) + grayscale_image[row, col - 1] * (-2) + \
23             grayscale_image[row + 1, col - 1] * (-1) + grayscale_image[row - 1, col + 1] + \
24             grayscale_image[row, col + 1] * 2 + grayscale_image[row + 1, col + 1]
25
26         gy = grayscale_image[row - 1, col - 1] * (-1) + grayscale_image[row - 1, col] * (-2) + \
27             grayscale_image[row - 1, col + 1] * (-1) + grayscale_image[row + 1, col - 1] + \
28             grayscale_image[row + 1, col] * 2 + grayscale_image[row + 1, col + 1]
29
30         output_image[row, col] = (gx**2 + gy**2)**(1/2)
31
32 # Mostrando os resultados
33 cv2.imshow('Sobel image', output_image)
34 cv2.waitKey(0)
```

Fonte: Elaborado pelo autor

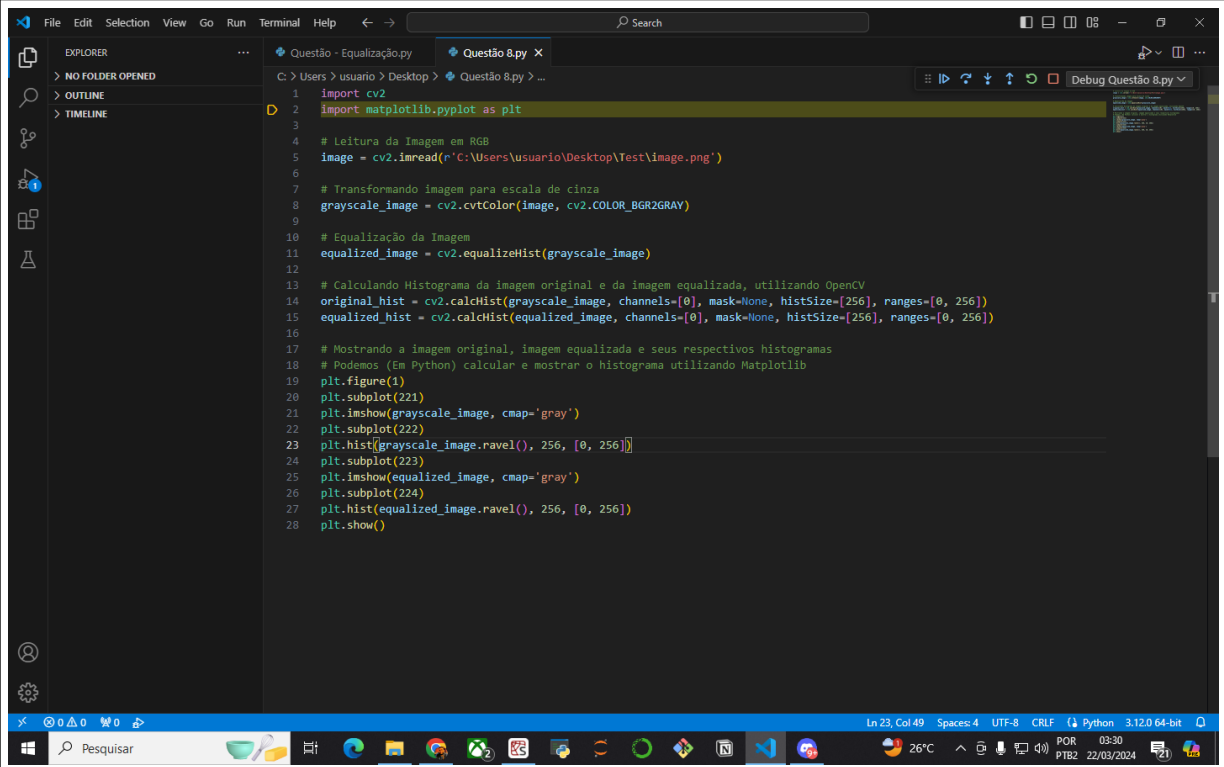
Figura 6 – Questão 6 (Resultado)



Fonte: Elaborado pelo autor

## 6 TRABALHO 1 - QUESTÃO 8

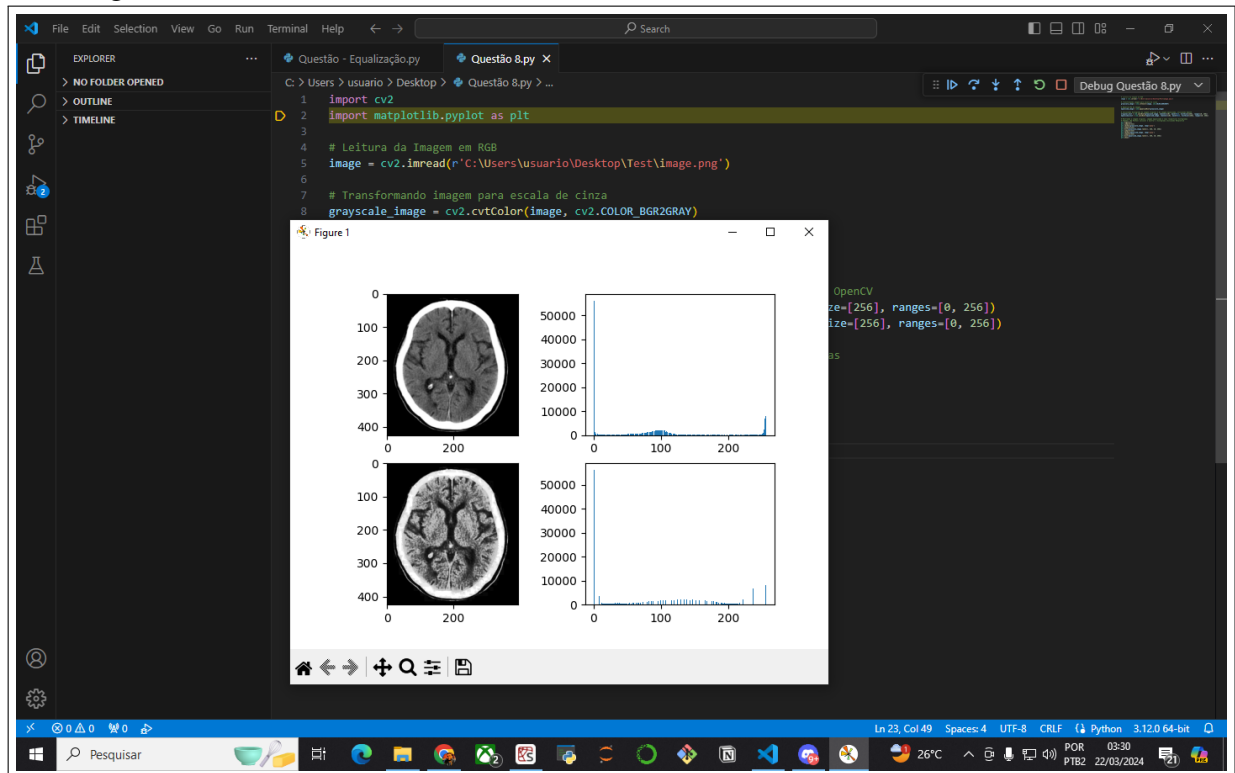
Figura 7 – Questão 8 (Código)



```
1 import cv2
2 import matplotlib.pyplot as plt
3
4 # Leitura da Imagem em RGB
5 image = cv2.imread(r'C:\Users\usuario\Desktop\Test\image.png')
6
7 # Transformando imagem para escala de cinza
8 grayscale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
9
10 # Equalização da Imagem
11 equalized_image = cv2.equalizeHist(grayscale_image)
12
13 # Calculando Histograma da imagem original e da imagem equalizada, utilizando OpenCV
14 original_hist = cv2.calcHist([grayscale_image], [0], None, [256], [0, 256])
15 equalized_hist = cv2.calcHist([equalized_image], [0], None, [256], [0, 256])
16
17 # Mostrando a imagem original, imagem equalizada e seus respectivos histogramas
18 # Podemos (Em Python) calcular e mostrar o histograma utilizando Matplotlib
19 plt.figure(1)
20 plt.subplot(221)
21 plt.imshow(grayscale_image, cmap='gray')
22 plt.subplot(222)
23 plt.hist(grayscale_image.ravel(), 256, [0, 256])
24 plt.subplot(223)
25 plt.imshow(equalized_image, cmap='gray')
26 plt.subplot(224)
27 plt.hist(equalized_image.ravel(), 256, [0, 256])
28 plt.show()
```

Fonte: Elaborado pelo autor

Figura 8 – Questão 8 (Resultado)

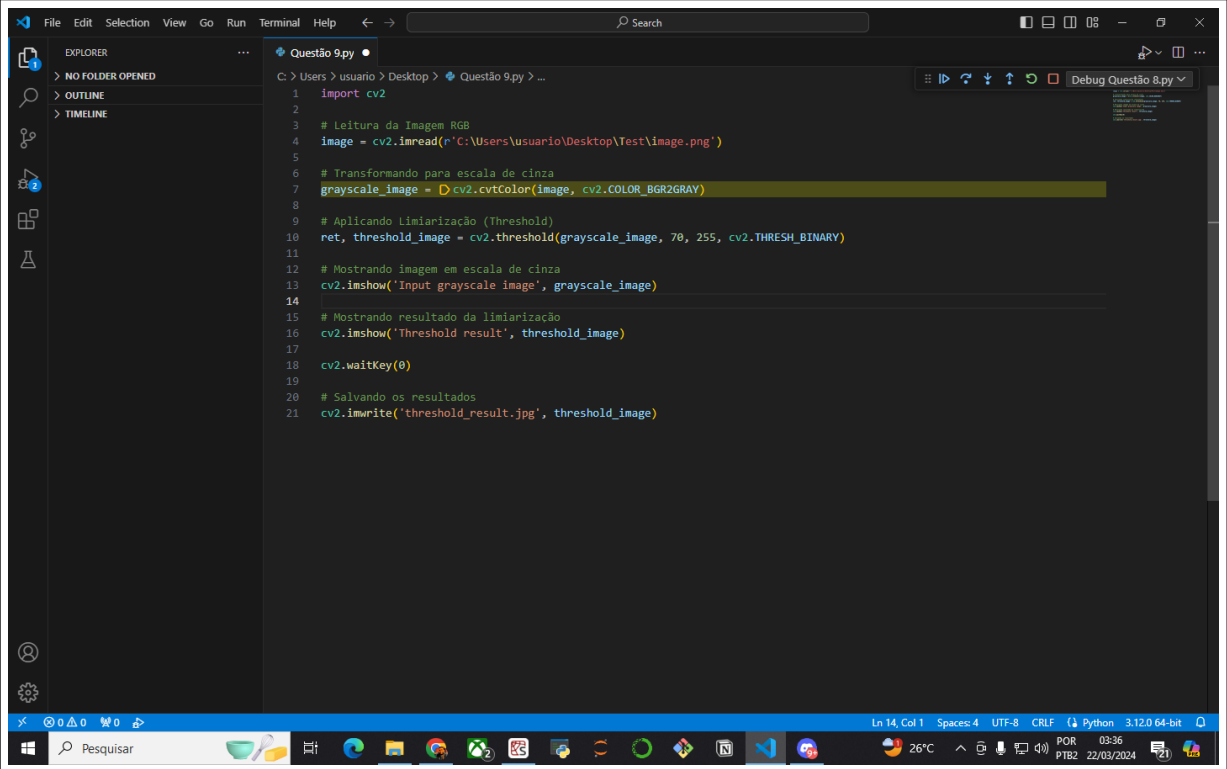


Fonte: Elaborado pelo autor



## 7 TRABALHO 1 - QUESTÃO 9

Figura 9 – Questão 9 (Código)

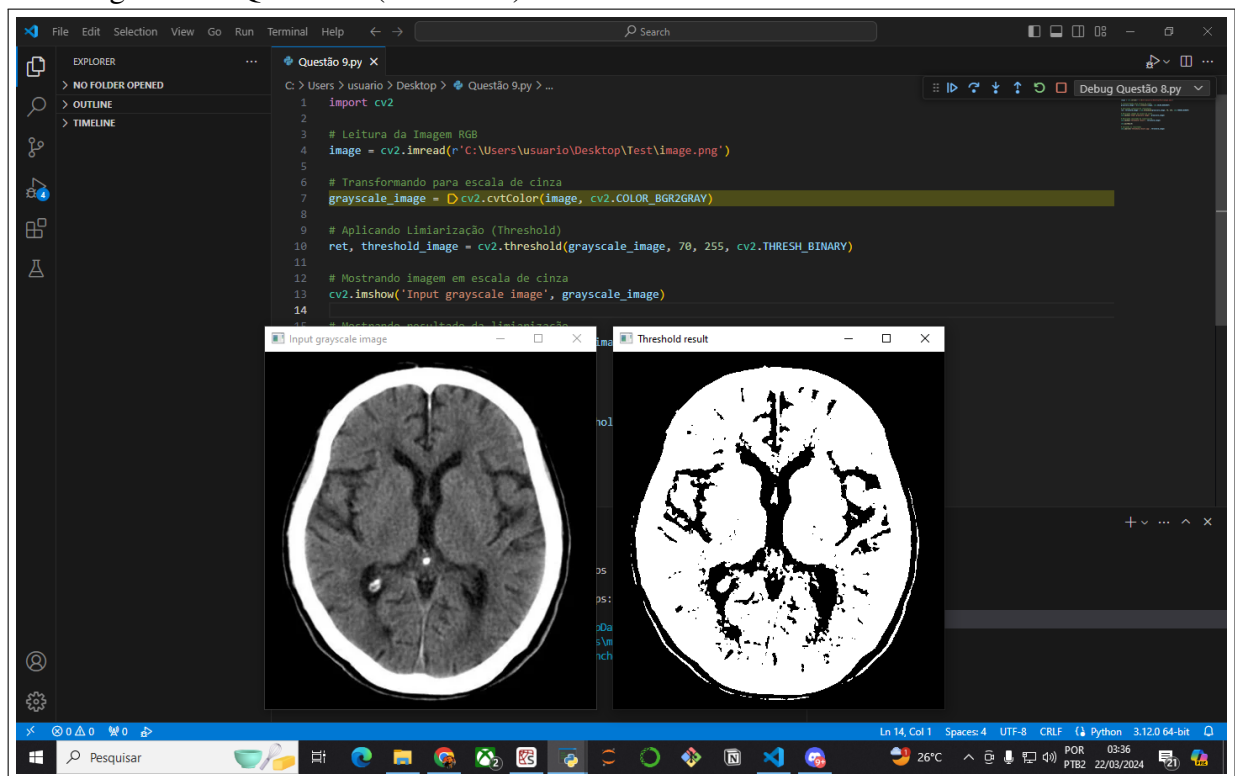


The image shows a screenshot of the Visual Studio Code editor interface. The Explorer panel on the left shows a project named 'Questão 9.py'. The main editor area displays a Python script for image processing using OpenCV. The script reads an image, converts it to grayscale, applies a binary threshold, and saves the result. The status bar at the bottom indicates the file is at line 14, column 1, with 4 spaces, in UTF-8 encoding with CRLF line endings, using Python 3.12.0 64-bit.

```
1 import cv2
2
3 # Leitura da Imagem RGB
4 image = cv2.imread(r'C:\Users\usuario\Desktop\Test\image.png')
5
6 # Transformando para escala de cinza
7 grayscale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
8
9 # Aplicando limiarização (Threshold)
10 ret, threshold_image = cv2.threshold(grayscale_image, 70, 255, cv2.THRESH_BINARY)
11
12 # Mostrando imagem em escala de cinza
13 cv2.imshow('Input grayscale image', grayscale_image)
14
15 # Mostrando resultado da limiarização
16 cv2.imshow('Threshold result', threshold_image)
17
18 cv2.waitKey(0)
19
20 # Salvando os resultados
21 cv2.imwrite('threshold_result.jpg', threshold_image)
```

Fonte: Elaborado pelo autor

Figura 10 – Questão 9 (Resultado)



Fonte: Elaborado pelo autor

## 8 CONCLUSÃO

Neste trabalho, foram abordados seis exercícios de processamento digital de imagens, que proporcionaram uma oportunidade valiosa para aplicar e compreender conceitos fundamentais deste campo. Embora não tenhamos conseguido abordar todos os dez exercícios propostos, os resultados obtidos e as lições aprendidas são significativos e merecem ser destacados.

Através da resolução desses exercícios, pudemos explorar uma variedade de técnicas e algoritmos de processamento de imagens, incluindo filtragem, segmentação e transformações geométricas. Estas operações básicas desempenham um papel crucial na análise e manipulação de imagens digitais, e sua compreensão é essencial para o desenvolvimento de soluções eficazes em diversas áreas de aplicação.