Aplicação do operador gradiente na detecção de bordas

Giovanni Mandel Martignago, João Pedro Garcia Guedes, Matheus Schiochet

[1]Departamento de Ciência da Computação – Universidade do Estado de Santa Catarina (UDESC)
Caixa Postal 631 – 89.219-710 – Joinville – SC – Brazil

1. Introdução

A detecção de bordas é uma etapa fundamental na análise de imagens digitais, pois permite identificar regiões onde ocorrem variações significativas na intensidade dos pixels. Essas variações normalmente indicam os limites entre diferentes objetos ou áreas dentro da imagem, evidenciando formas e destacando elementos do fundo ou de outras estruturas.

Para detectar essas transições, são utilizados operadores baseados em gradiente, que avaliam a variação de intensidade luminosa em direções específicas, como horizontal e vertical. Esse processo é realizado por meio da aplicação de máscaras, como as de Sobel e Prewitt, que são pequenas matrizes responsáveis por realçar as bordas conforme a direção do gradiente. A técnica utilizada, chamada convolução, percorre a imagem aplicando essas máscaras sobre cada pixel, resultando nos componentes horizontal e vertical do gradiente.

Com esses componentes, é possível calcular a magnitude e a direção do gradiente, que indicam a força e a orientação das bordas. Após essa etapa, aplica-se a supressão de não-máximos, um processo que remove pixels irrelevantes para manter apenas os contornos mais marcantes, deixando as bordas mais finas e precisas. Por fim, os resultados obtidos com a implementação manual são comparados com os produzidos por ferramentas da biblioteca OpenCV, utilizando o índice de similaridade estrutural (SSIM) como métrica para avaliar a fidelidade e a qualidade da detecção de bordas em relação à imagem original.

2. Descrição do problema

Quando olhamos para uma imagem em tons de cinza, cada ponto dela representa um valor de brilho. Em algumas partes da imagem, esse brilho muda de forma mais rápida e essas mudanças geralmente mostram onde começa ou termina um objeto. Ou seja, é nessas áreas que conseguimos identificar as bordas.

Para encontrar essas bordas, usamos uma ferramenta chamada operador gradiente, que mede o quanto o brilho varia nas direções horizontal e vertical. Isso é feito aplicando filtros específicos, como os de Sobel, Prewitt ou Scharr, que ajudam a calcular essas mudanças.

Antes de usar esses operadores, no entanto, é importante suavizar a imagem para diminuir o ruído. Isso é feito com um filtro passa-baixa, que reduz pequenas variações indesejadas nos pixels, mas sem apagar os contornos importantes. Essa etapa evita que o ruído seja confundido com bordas de verdade.

Depois de aplicar os filtros de borda, obtemos duas imagens: uma mostrando as mudanças na horizontal e outra na vertical. A partir delas, calculamos a intensidade da

borda (magnitude) e a direção em que ela aparece. Mesmo assim, nem toda mudança forte representa uma borda real. Por isso, usamos mais uma etapa chamada supressão de não-máximos, que mantém apenas os pontos mais importantes e que realmente formam uma borda. Dessa forma, fazemos a comparação entre os resultados obtidos manualmente com os resultados da biblioteca OpenCV, para ver o quanto os dois métodos se parecem.

3. Objetivo

Esta tarefa tem como principal objetivo implementar manualmente operadores de gradiente — especificamente os filtros de Sobel e Prewitt — para detectar bordas em imagens. O processo envolve desde o carregamento e pré-processamento das imagens até a aplicação dos operadores e o refinamento das bordas com a supressão de não-máximos. Após isso, os resultados obtidos manualmente são comparados com aqueles gerados por funções do OpenCV, utilizando a métrica SSIM (Índice de Similaridade Estrutural) para avaliar a qualidade e a precisão das bordas detectadas entre cada método.

4. Metodologia

A metodologia adotada neste trabalho foi estruturada em quatro etapas principais, com o objetivo de detectar bordas em imagens digitais por meio da aplicação de operadores de gradiente. As etapas são:

1. **Pré-processamento da imagem**: visando à melhoria da qualidade dos resultados, as imagens são inicialmente convertidas para escala de cinza e passam por técnicas de redução de ruído, como o uso de filtros suavizantes. Esse passo é essencial para minimizar falsos positivos na detecção de bordas.
2. **Aplicação dos operadores de gradiente**: foram implementados manualmente os operadores de Sobel e Prewitt, que atuam por meio da convolução de máscaras nas direções horizontal e vertical. Esses operadores são responsáveis por estimar a variação de intensidade entre pixels vizinhos, destacando regiões de transição brusca — típicas de contornos.
3. **Cálculo da magnitude e da direção do gradiente**: a partir dos componentes horizontal (G_x) e vertical (G_y) obtidos nas convoluções, é calculada a magnitude do gradiente ($|\nabla f|$) para indicar a intensidade das bordas, e a direção para eventual uso em etapas posteriores de refinamento.
4. **Supressão de não-máximos**: com base na direção do gradiente, essa técnica é utilizada para manter apenas os pixels que representam possíveis bordas locais, eliminando redundâncias e afinando o resultado.

Ao final do processo, os resultados da implementação manual dos operadores foram comparados, de forma quantitativa, com os obtidos por meio das funções nativas de bibliotecas especializadas. Essa comparação visa avaliar a fidelidade da implementação e a efetividade dos operadores na detecção de bordas.

5. Implementação

A implementação deste projeto foi realizada utilizando a linguagem Python, com auxílio das bibliotecas OpenCV, NumPy e scikit-image, para aplicar e comparar operadores de detecção de bordas nas imagens. O código realiza diversas etapas, descritas a seguir:

5.1. Pré-processamento das Imagens

Inicialmente, as imagens foram carregadas em escala de cinza utilizando *cv2.imread*. Posteriormente, foi aplicado um filtro de mediana para remoção de ruídos.

A função *filtro_mediana()* percorre a imagem utilizando uma janela 3×3 e substitui o valor do pixel central pela mediana dos valores da vizinhança, o que ajuda a preservar bordas e eliminar ruídos isolados. As imagens resultantes da filtragem são salvas para visualização posterior.

5.2. Operadores de Gradiente

Para a detecção das bordas, foram utilizadas máscaras de convolução dos operadores Sobel e Prewitt. As bordas horizontais e verticais foram extraídas com a aplicação de convolução em duas direções: x (horizontal) e y (vertical).

O código possui duas abordagens:

- Automática, utilizando as funções *cv2.Sobel* e *cv2.filter2D* da OpenCV.
- Manual, por meio da função *aplicar_convolucao()* que percorre cada pixel da imagem e aplica a operação de convolução com a máscara correspondente.

As máscaras utilizadas para detectar bordas nas direções X e Y foram definidas como:

Sobel:

X: [-1, 0, 1], [-2, 0, 2], [-1, 0, 1]

Y: [-1, -2, -1], [0, 0, 0], [1, 2, 1]

Prewitt:

X: [-1, 0, 1], [-1, 0, 1], [-1, 0, 1]

Y: [-1, -1, -1], [0, 0, 0], [1, 1, 1]

5.3. Magnitude e Direção do Gradiente

Após o cálculo dos gradientes horizontais (gx) e verticais (gy), foi utilizada a função *calcula_magnitude_direcao()* para calcular a magnitude do gradiente e a direção da borda. A magnitude representa a intensidade da borda, enquanto a direção indica sua orientação. Os ângulos são normalizados para o intervalo de 0 a 180 graus.

5.4. Supressão de Não-Máximos com Critério de Comparação Relativa

Para refinar os contornos detectados, é aplicada uma técnica de supressão de não-máximos modificada, que considera um critério de comparação relativa com um fator K. Essa técnica verifica se o valor da magnitude do gradiente de um pixel é maior que o de seus vizinhos ao longo da direção do gradiente. Se a condição for satisfeita, o pixel é mantido como borda. A função *maximoLocal()* realiza esse procedimento. Esse passo melhora a definição das bordas e reduz ruídos.

5.5. Armazenamento dos Resultados

As imagens processadas nas diversas combinações (Sobel e Prewitt, manual e OpenCV) são salvas em arquivos separados para posterior análise visual e comparação.

5.6. Similaridade Estrutural (SSIM)

Com o objetivo de avaliar a qualidade dos resultados, a função *calcular_ssim()* determina o Índice de Similaridade Estrutural (SSIM) entre as bordas obtidas pelos métodos manual e OpenCV. Para isso, a função carrega ambas as imagens a serem comparadas, utiliza a função ssim da biblioteca skimage.metrics para calcular o índice e retorna o valor da similaridade.

6. Resultados

Nesta seção, será realizada uma análise dos resultados obtidos no experimento. Serão apresentadas as imagens originais utilizadas, bem como suas versões após a aplicação do filtro da mediana e da detecção de bordas. Além disso, será investigada a influência do parâmetro K nesse processo. Por fim, será feita uma comparação entre os métodos de detecção de bordas desenvolvidos manualmente e aqueles disponíveis na biblioteca OpenCV.

6.1. Imagens Originais

Estas são as imagens de exemplo originais utilizadas:

Figure 1. moedas

Figure 2. Lua1_gray

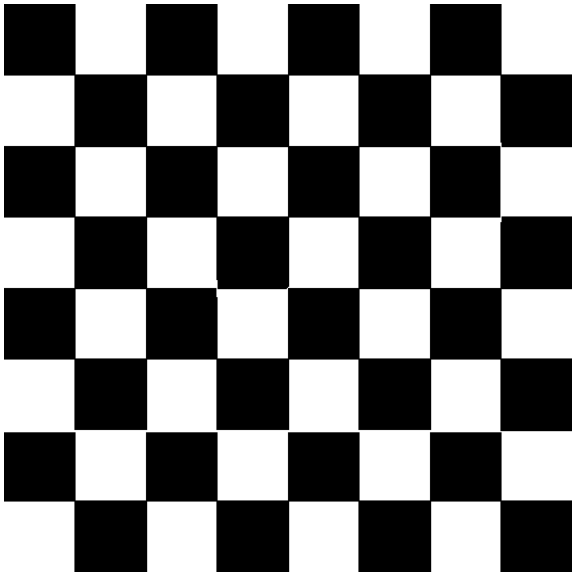

Figure 3. chessboard_inv

Figure 4. img02

6.2. Filtragem Mediana

Estas são as imagens de exemplo após a aplicação do filtro da mediana:

Figure 5. moedas filtrada

Figure 6. Lua1_gray filtrada

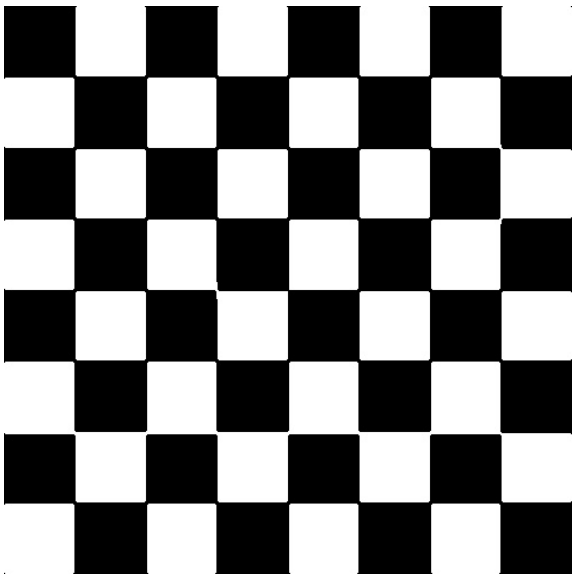

Figure 7. chessboard_inv filtrada

Figure 8. img02 filtrada

6.3. Parâmetro K

Durante a etapa de refinamento das bordas, foi utilizada uma técnica de supressão de não-máximos com limiar adaptativo. Nesse processo, o parâmetro K exerce um papel crucial na definição da sensibilidade à presença de bordas.

O parâmetro K é um fator de multiplicação aplicado à magnitude dos pixels vizinhos na direção do gradiente. Um pixel só é mantido como borda se sua magnitude for estritamente maior que a de seus vizinhos multiplicada por esse fator. Assim:

- $K < 1$: torna o critério menos rigoroso, permitindo que mais pixels sejam considerados como borda. Isso pode resultar em bordas mais espessas ou ruidosas, incluindo falsas bordas.
- $K = 1$: representa um equilíbrio entre sensibilidade e precisão, mantendo apenas os picos locais de maior intensidade.
- $K > 1$: torna o critério mais seletivo, resultando em bordas mais finas e limpas, mas com risco de perder detalhes sutis ou bordas de baixo contraste.

6.3.1. Parâmetro $K < 1$

Resultados da detecção de bordas com $K < 1$, usando tanto a implementação manual quanto OpenCV. Foi escolhido K = 0.5.

Utilizando a implementação manual:

Figure 9. moedas Prewitt manual

Figure 10. moedas Sobel manual

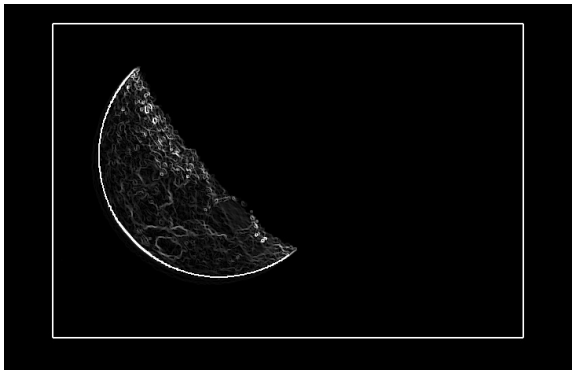

Figure 11. Lua1_gray Prewitt manual

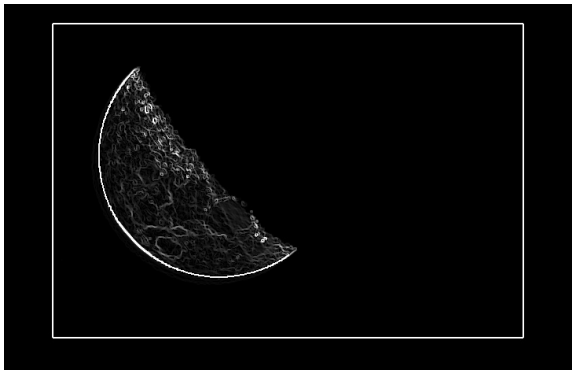

Figure 12. Lua1_gray Sobel manual

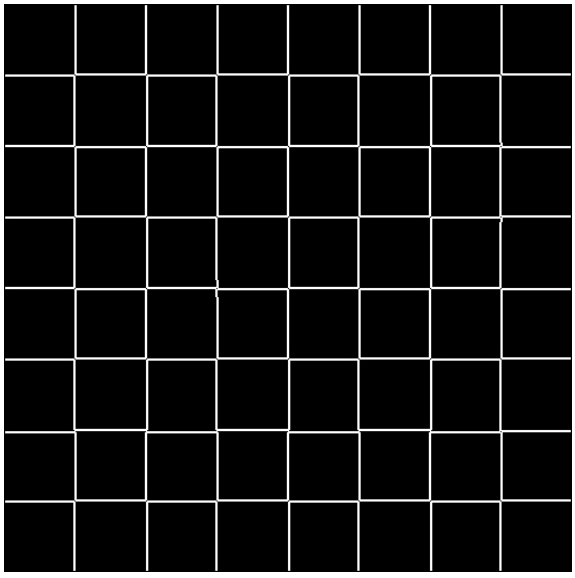

Figure 13. chessboard_inv Prewitt manual

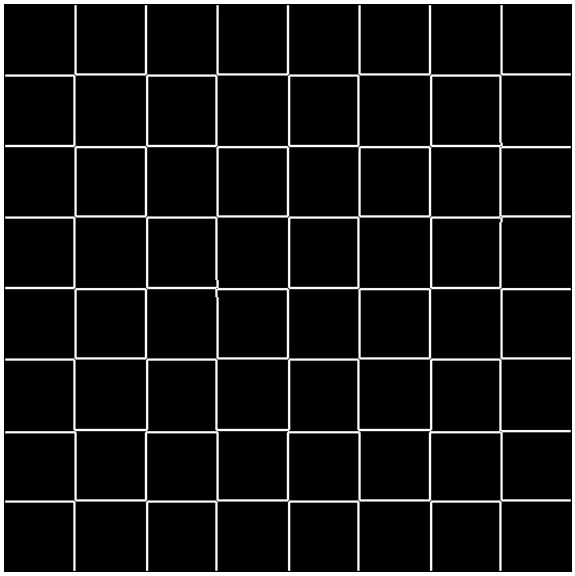

Figure 14. chessboard_inv Sobel manual

Figure 15. img02 Prewitt manual

Figure 16. img02 Sobel manual

Utilizando a OpenCV:

Figure 17. moedas Prewitt OpenCV

Figure 18. moedas Sobel OpenCV

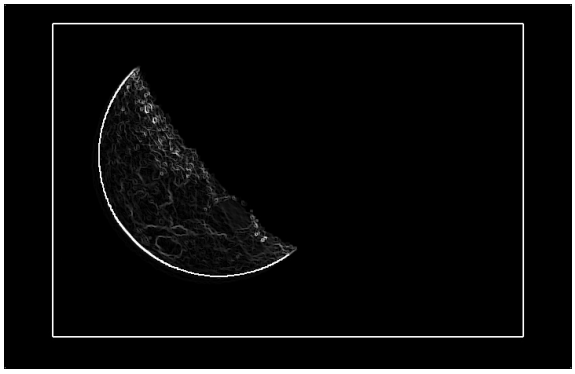

Figure 19. Lua1_gray Prewitt OpenCV

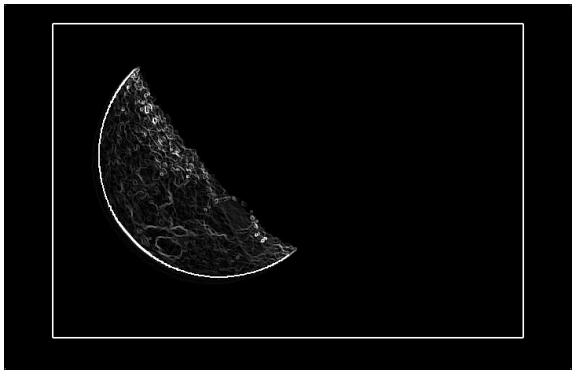

Figure 20. Lua1_gray Sobel OpenCV

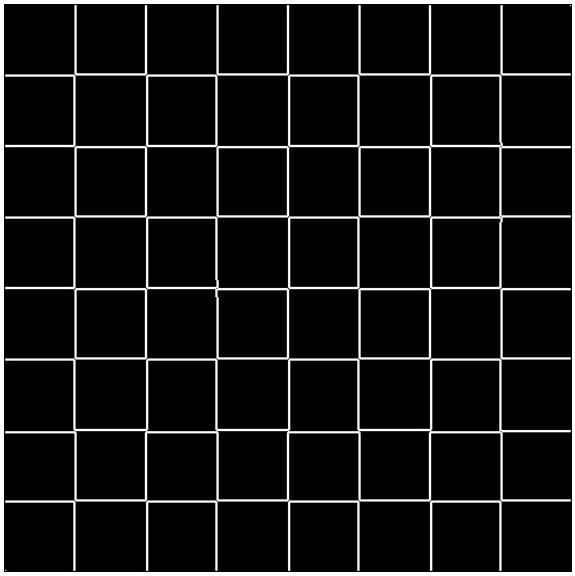

Figure 21. chessboard_inv Prewitt OpenCV

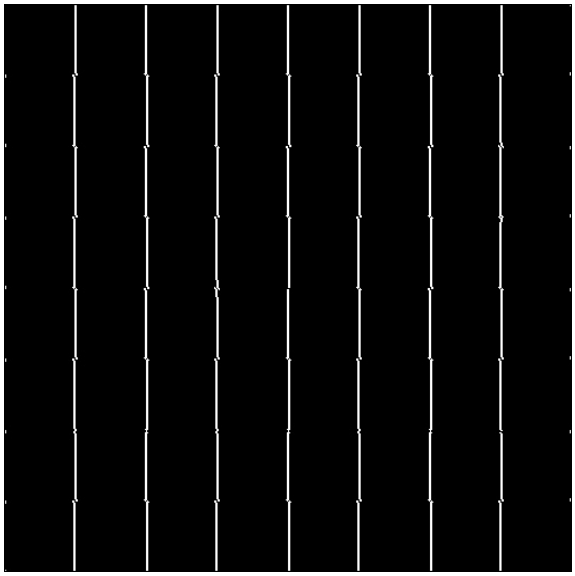

Figure 22. chessboard_inv Sobel OpenCV

Figure 23. img02 Prewitt OpenCV

Figure 24. img02 Sobel OpenCV

6.3.2. Parâmetro $K = 1$

Resultados da detecção de bordas com $K = 1$, usando tanto a implementação manual quanto OpenCV.

Utilizando a implementação manual:

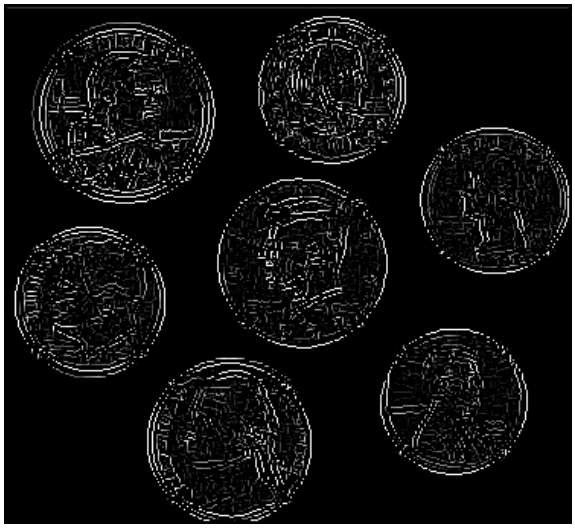

Figure 25. moedas Prewitt manual

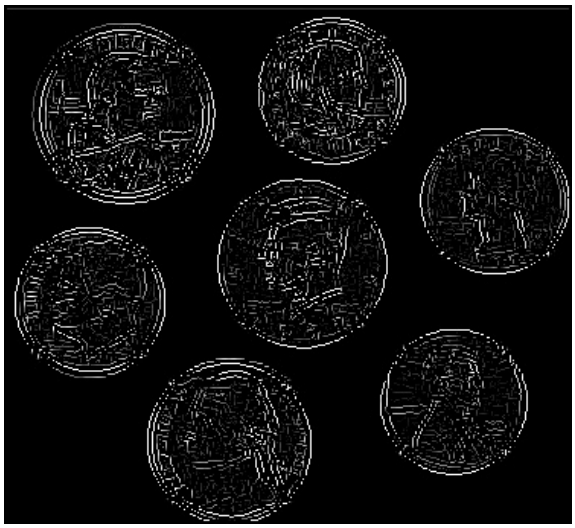

Figure 26. moedas Sobel manual

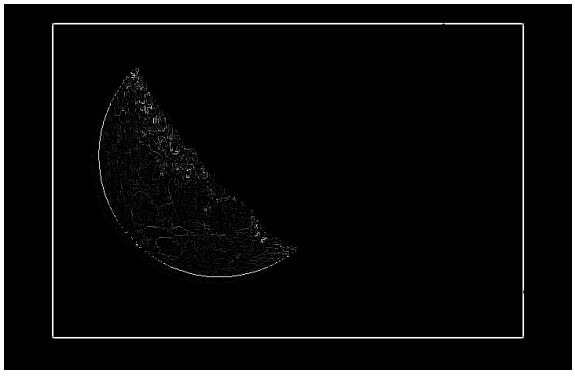

Figure 27. Lua1_gray Prewitt manual

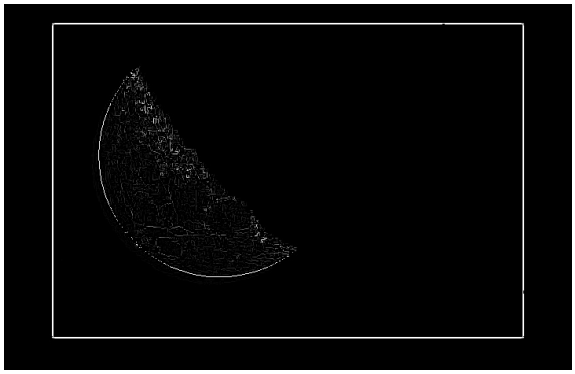

Figure 28. Lua1_gray Sobel manual

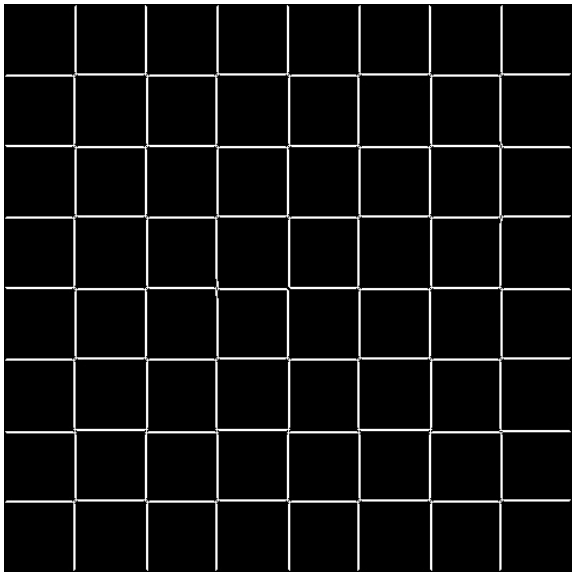

Figure 29. chessboard_inv Prewitt manual

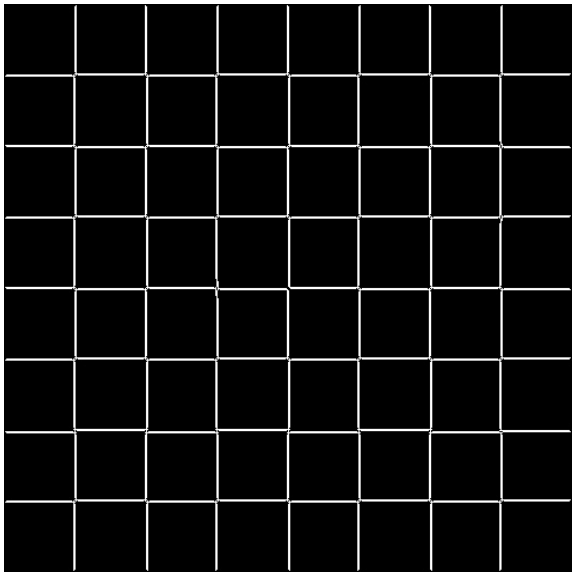

Figure 30. chessboard_inv Sobel manual

Figure 31. img02 Prewitt manual

Figure 32. img02 Sobel manual

Utilizando a OpenCV:

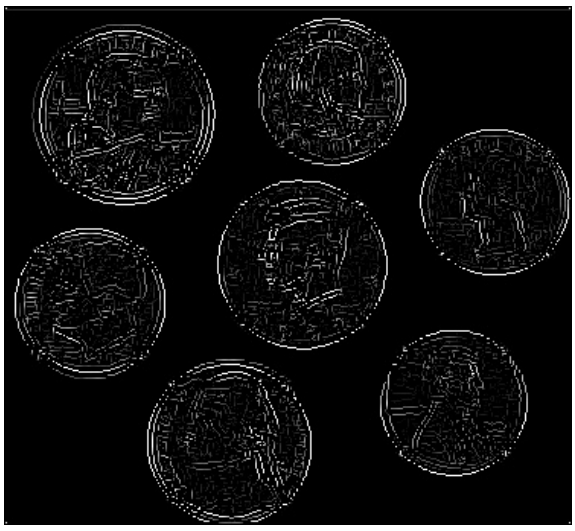

Figure 33. moedas Prewitt OpenCV

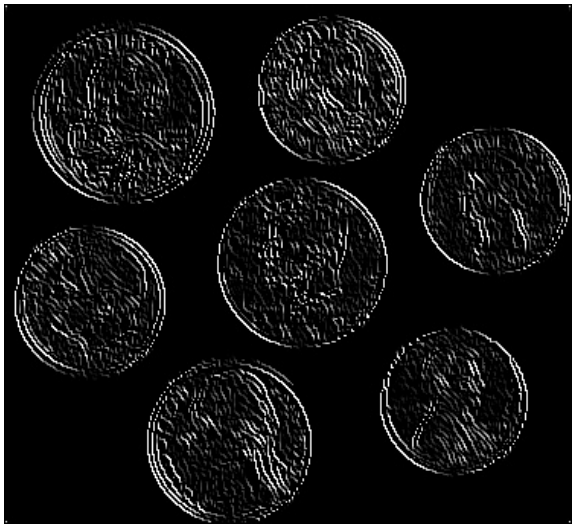

Figure 34. moedas Sobel OpenCV

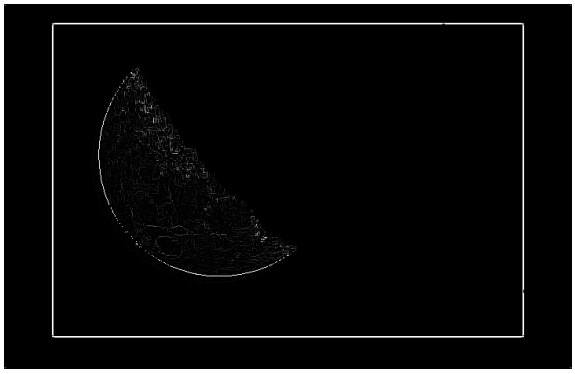

Figure 35. Lua1_gray Prewitt OpenCV

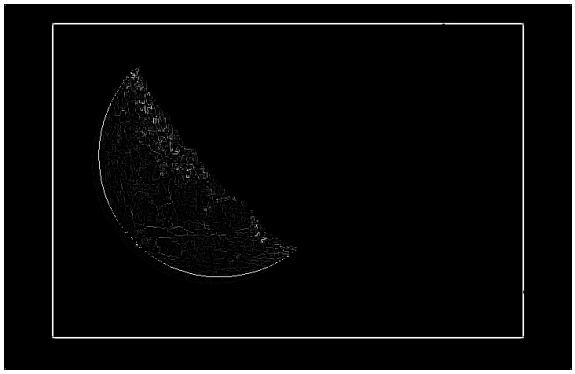

Figure 36. Lua1_gray Sobel OpenCV

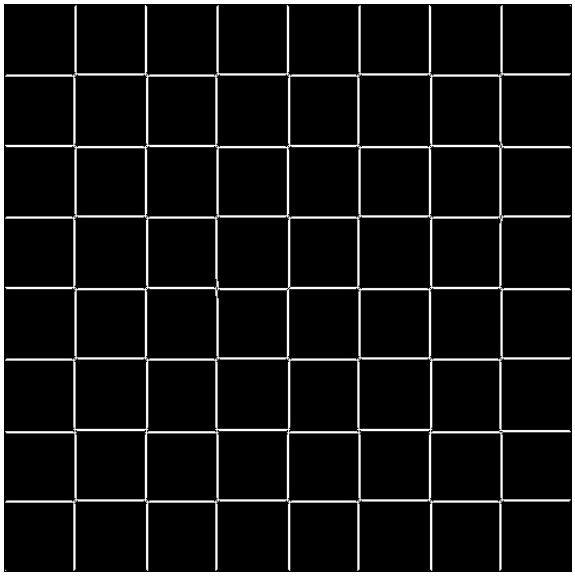

Figure 37. chessboard_inv Prewitt OpenCV

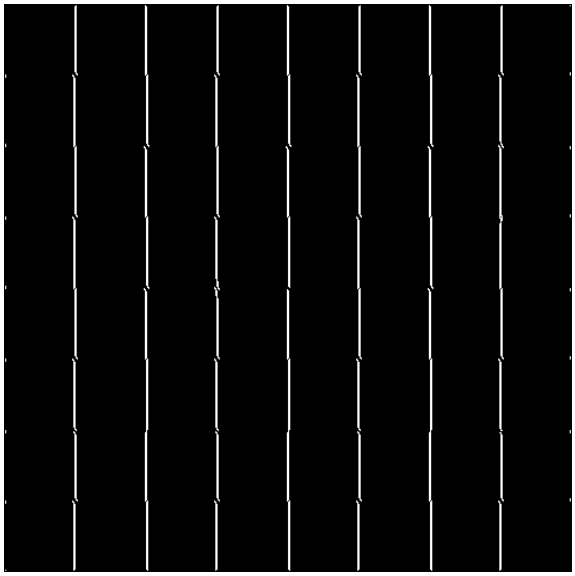

Figure 38. chessboard_inv Sobel OpenCV

Figure 39. img02 Prewitt OpenCV

Figure 40. img02 Sobel OpenCV

6.3.3. Parâmetro $K > 1$

Resultados da detecção de bordas com $K > 1$, usando tanto a implementação manual quanto OpenCV. Foi escolhido K = 1.5.

Utilizando a implementação manual:

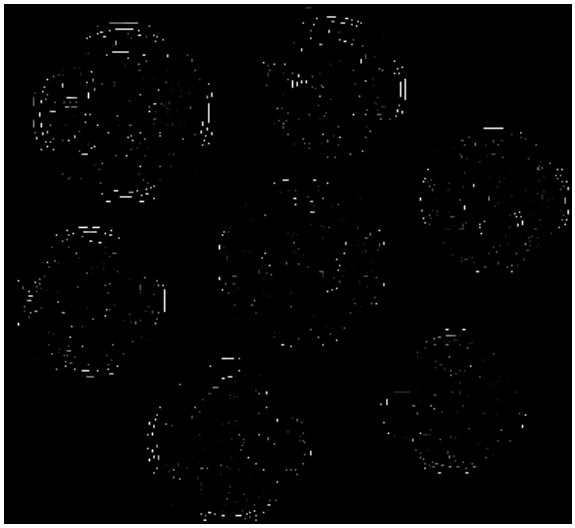

Figure 41. moedas Prewitt manual

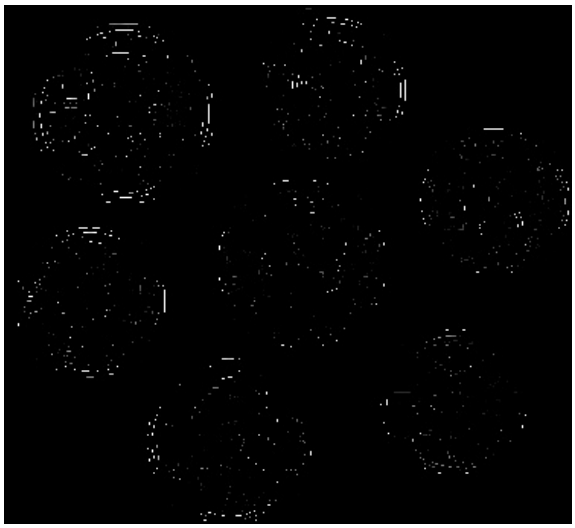

Figure 42. moedas Sobel manual

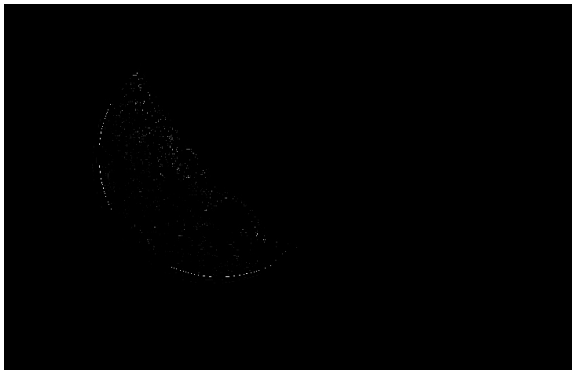

Figure 43. Lua1_gray Prewitt manual

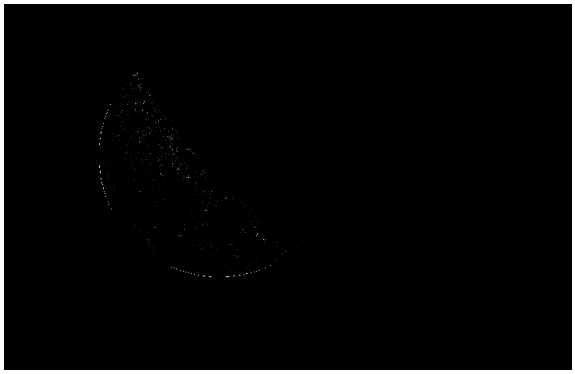

Figure 44. Lua1_gray Sobel manual

Figure 45. chessboard_inv Prewitt manual

Figure 46. chessboard_inv Sobel manual

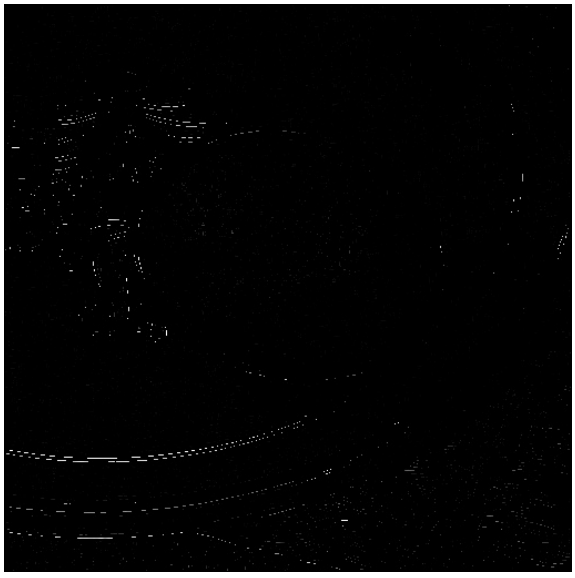

Figure 47. img02 Prewitt manual

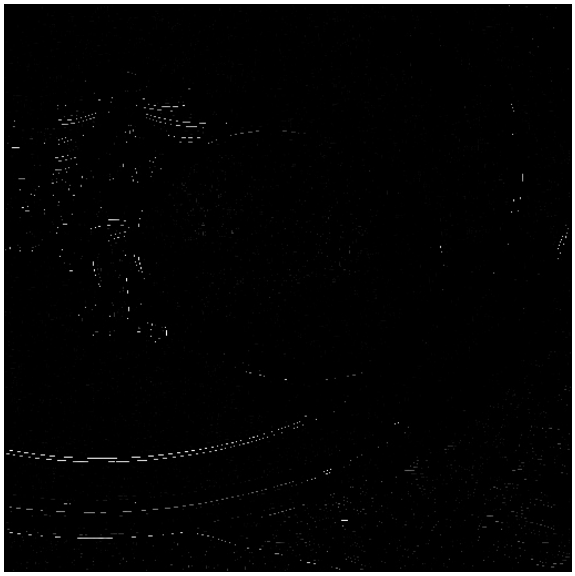

Figure 48. img02 Sobel manual

Utilizando a OpenCV:

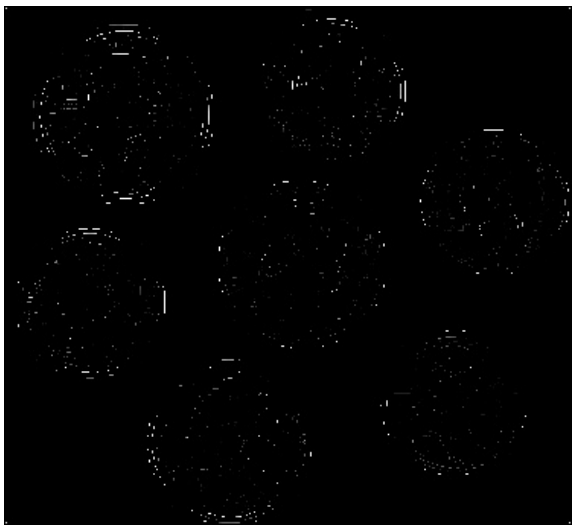

Figure 49. moedas Prewitt OpenCV

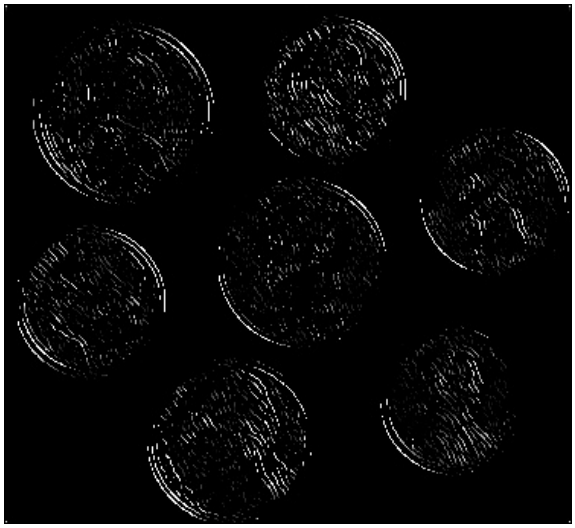

Figure 50. moedas Sobel OpenCV

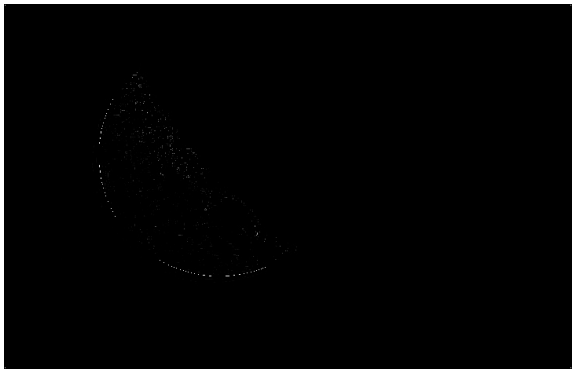

Figure 51. Lua1_gray Prewitt OpenCV

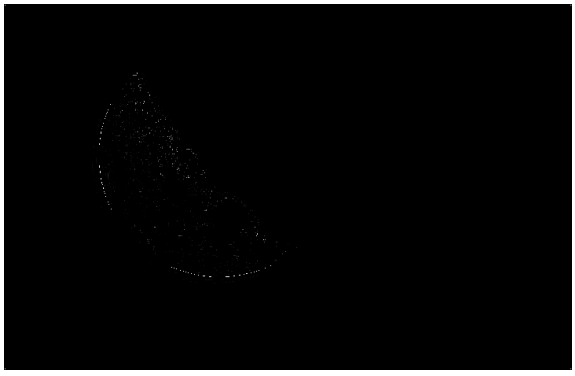

Figure 52. Lua1_gray Sobel OpenCV

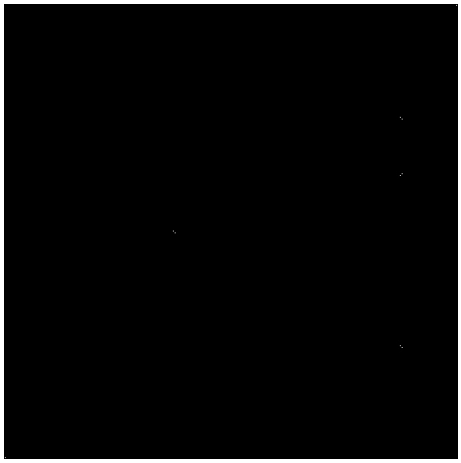

Figure 53. chessboard_inv Prewitt OpenCV

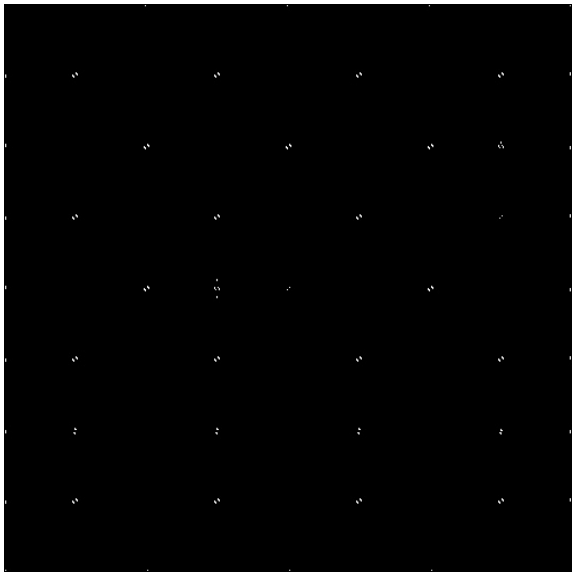

Figure 54. chessboard_inv Sobel OpenCV

Figure 55. img02 Prewitt OpenCV

Figure 56. img02 Sobel OpenCV

6.4. SSIM

Comparação do Índice de Similaridade Estrutural (SSIM) entre a implementação manual e OpenCV:

- SSIM com K = 0.5:
  SSIM entre Lua Sobel CV vs Manual: 0.9999
  SSIM entre Lua Prewitt CV vs Manual: 0.9877
  SSIM entre Xadrez Sobel CV vs Manual: 0.8949
  SSIM entre Xadrez Prewitt CV vs Manual: 1.0000
  SSIM entre Prato Sobel CV vs Manual: 0.7134

SSIM entre Prato Prewitt CV vs Manual: 0.9028
SSIM entre Moedas Sobel CV vs Manual: 0.8050
SSIM entre Moedas Prewitt CV vs Manual: 0.9469

- SSIM com K = 1:
  SSIM entre Lua Sobel CV vs Manual: 0.9999
  SSIM entre Lua Prewitt CV vs Manual: 0.9830
  SSIM entre Xadrez Sobel CV vs Manual: 0.8930
  SSIM entre Xadrez Prewitt CV vs Manual: 0.9999
  SSIM entre Prato Sobel CV vs Manual: 0.5662
  SSIM entre Prato Prewitt CV vs Manual: 0.8798
  SSIM entre Moedas Sobel CV vs Manual: 0.6395
  SSIM entre Moedas Prewitt CV vs Manual: 0.9196

- SSIM com K = 1.5:
  SSIM entre Lua Sobel CV vs Manual: 0.9999
  SSIM entre Lua Prewitt CV vs Manual: 0.9835
  SSIM entre Xadrez Sobel CV vs Manual: 0.9861
  SSIM entre Xadrez Prewitt CV vs Manual: 1.0000
  SSIM entre Prato Sobel CV vs Manual: 0.5824
  SSIM entre Prato Prewitt CV vs Manual: 0.9384
  SSIM entre Moedas Sobel CV vs Manual: 0.5590
  SSIM entre Moedas Prewitt CV vs Manual: 0.9169

7. Discussão

Os resultados obtidos demonstram que a implementação manual dos operadores de Sobel e Prewitt é capaz de detectar as bordas de forma eficaz, evidenciando as transições de intensidade na imagem conforme esperado. A comparação quantitativa com as funções nativas da biblioteca OpenCV revelou que, apesar da implementação manual apresentar desempenho satisfatório, as funções otimizadas da biblioteca possuem maior eficiência computacional e podem lidar melhor com imagens de maior resolução e ruído.

A supressão de não-máximos contribuiu significativamente para o refinamento das bordas detectadas, reduzindo falsas detecções e tornando o resultado mais preciso. Contudo, algumas limitações foram observadas, como a sensibilidade a ruídos residuais no pré-processamento, que ainda podem gerar bordas espúrias. Além disso, o método depende do ajuste adequado dos parâmetros, como o limiar para a magnitude do gradiente, o que pode variar conforme a natureza das imagens analisadas.

Para trabalhos futuros, sugere-se a integração de técnicas adicionais, como a suavização adaptativa e a utilização de operadores de gradiente mais robustos, além da implementação de métodos de limiarização automática. A aplicação do operador gradiente em combinação com algoritmos de aprendizado de máquina também pode ser explorada para melhorar a detecção em contextos mais complexos.

Em suma, a metodologia proposta demonstra ser uma base sólida para a detecção de bordas, com resultados compatíveis com as soluções existentes, e oferece flexibilidade para aprimoramentos e adaptações conforme a necessidade das aplicações.

8. Conclusão

Este trabalho apresentou a implementação manual dos operadores de gradiente de Sobel e Prewitt para a detecção de bordas em imagens digitais, com destaque para o cálculo da magnitude do gradiente e a aplicação da supressão de não-máximos utilizando um critério relativo com fator K. A metodologia adotada permitiu identificar eficientemente as transições de intensidade, resultando em bordas nítidas e bem definidas.

A comparação com as funções nativas da biblioteca OpenCV evidenciou que a implementação manual, apesar de menos otimizada, é capaz de fornecer resultados satisfatórios e comparáveis, demonstrando a viabilidade do método para aplicações acadêmicas e de pesquisa. Além disso, a supressão de não-máximos contribuiu significativamente para o refinamento dos contornos, reduzindo ruídos e melhorando a qualidade das bordas detectadas.

Futuros trabalhos poderão explorar técnicas de pré-processamento mais avançadas, ajustes dinâmicos do fator K e a combinação do operador gradiente com métodos baseados em aprendizado de máquina para aumentar a robustez e a adaptabilidade da detecção de bordas em diferentes contextos e tipos de imagens.