

Trabalho 1

MO443 - Introdução ao Processamento de Imagem Digital

Diego Alysson Braga Moreira

RA:230640

diegoalyssonbm@gmail.com / d230640@dac.unicamp.br

UNICAMP

I. INTRODUÇÃO

O principal objetivo do trabalho 1, aqui apresentado, é observar o comportamento de diferentes máscaras (filtros) propostos nos enunciados e como estes filtros agem para a composição da imagem final. Aqui é apresentado e implementado os problemas dados, utilizando principalmente python3, opencv-python e os conceitos introduzidos na aula e no trabalho 0.

Outro fator também considerado por este trabalho é a múltiplas formas de se tratar bordas de imagens ao se aplicar diferentes tipos de máscaras através da convolução realizada na imagem. Estes filtros serão comparados entre si, distinguindo-se o comportamento de algumas propostas.

Cada um desses problemas serão abordados e discutidos nas próximas seções, com instruções de execução, modelos e decisões de implementação, saídas dos dados, resultados e conclusões.

II. EXECUÇÃO

Para a execução padrão deste projeto, o arquivo de implementação dos códigos, imagens e pastas foram enviadas em conjunto. Abaixo segue a lista dos componentes que podem ser encontrados neste trabalho:

- Pasta Códigos: Arquivo de Execução do Código (trabalho1.py)
- Pasta Imagens: baboon.png, butterfly.png, city.png, house.png, seagull.png, jangada.png, fortaleza.png, igreja.png e forteNs.png
- Pasta Raiz: Pasta Código, Pasta Imagens e Relatório em PDF

Onde, as estão imagens em uma subpasta chamada Images, a qual deve ser utilizada como padrão de localização das imagens utilizadas. As pastas e seu conteúdo (arquivos .py e imagens) estão na pasta Raiz do projeto, juntamente com o relatório em pdf. O arquivo .py necessário para a execução e observação dos resultados é o arquivo trabalho1.py.

Nesta seção serão apresentadas todas as instruções de execução para ao código enviado, suas dependências, funcionalidades, parâmetros, decisões de implementação e saídas de dados.

O projeto deve ser executado com uma linha básica de chamada no terminal, como exemplo:

`python3 -i <nome_entrada.png>-o <nome_saida.png>`

A. Dependências

Para a execução correta dos códigos em anexo, as seguintes dependências são essenciais:

- opencv-python: Utilizado para importação, exportação, exibição das imagens e convoluções.
- NumPy: Manipulação de vetores e matrizes. Como métodos de rotações e conversões.
- sys: Utilizada para aquisição dos argumentos em linha de comando.
- scipy: Utilizado para alguns operações com matrizes e convoluções.

B. Funcionalidades

Nesta seção será descrito as funcionalidades disponíveis no código em anexo, os parâmetros recebidos por este código e alguns exemplos de execução. Detalhes de implementação e técnicas são descritas na seção III.

Entre as funcionalidades disponíveis no trabalho 1, contidas no arquivos trabalho1.py estão, a seleção de diferentes máscaras para convolução de imagens. Os filtros utilizados foram propostos na descrição do trabalho, todos foram implementados e testados nas imagens de exemplo. Também disponibiliza-se opções de alteração dos tipos de composição utilizados para o comportamento das bordadas das imagens, uma vez que, ao operar estas bordas com a máscara, é necessário realizar um tratamento, visto que parte deste filtro estará fora da imagem.

C. Parâmetros

Para a correta execução dos códigos em anexo, deve-se seguir o seguinte modelo de forma geral:

`python3 trabalho1.py [INPUT] [OUTPUT] [OPCOES]`

A partir do padrão acima, os elementos em parênteses são obrigatórios, os elementos entre colchetes podem expandir suas possibilidades de acordo com as necessidades de cada questão, com utilização opcional, que serão explicadas a seguir.

- 1) python3: Palavra chave para utilização da linguagem python, 3 edição, no sistema linux.
- 2) trabalho1.py: Parâmetro obrigatório, define o arquivo python a ser executado com os códigos para a solução dos problemas, neste caso, trabalho1.py é o arquivo em questão.
- 3) INPUT: Especifica o arquivo imagem que será tratado através dos algoritmos. Também parâmetro obrigatório.

Deve-se explicitar o comando Input através da palavra -i ou --ifile, o nome da imagem também deve especificar a extensão do arquivo. Neste trabalho, apenas arquivos PNG (.png) devem ser utilizados como entrada. Como exemplo:

- python3 trabalho1.py -i baboon.png

4) OUTPUT: Este parâmetro opcional é utilizado para especificar o nome do arquivo da imagem gerada após o processamento do algoritmos. O nome deve conter também o formato de saída, que neste trabalho abrange somente formatos PNG (.png). Caso não informado, a imagem resultante terá o nome no formato, <Filtro><Borda>.png, que representa o Filtro utilizado e a Borda escolhida pelo usuário. Para definir o valor deste parâmetro, deve-se utilizar a palavra -o ou --ofile. Como possível caso de utilização, tem-se:

- python3 trabalho1.py -i baboon.png -o saida.png

5) OPCOES: Este parâmetro representa as múltiplas opções disponibilizadas.

- Parâmetros: -k ou --kernel, -b ou --borderType,
- Descrição:

O primeiro parâmetro opcional, kernel, define qual será a filtro utilizada para a convolução. As opções disponível para escolha a partir deste parâmetro reflete cada item requisitado no texto do trabalho 1. O usuário pode escolher as opções, h1, h2, h3, h4, h5, h6, h7, h8, h9, h10 e h11, que representam as 11 máscaras requisitadas no trabalho. Caso não seja disponibilizado este parâmetro, será utilizado o filtro h1 como padrão. A palavra de seleção para este atributo é -k x ou --kernel x, onde x é um valor entre {h1 e h12}.

- python3 trabalho1 -i baboon.png -k h1
- python3 trabalho1 -i city.png --kernel 8

Um outro parâmetro que pode ser definido é o tipo de política de borda que será aplicada na imagem. Estes tipos são definidos de acordo com a teoria apresentada em aulas e alguns dos tipos disponíveis para utilização, estão: constant (1), replicate (2), reflect (3), reflect101 (4) e isolated (5). Para selecionar o tipo de borda, deve-se escrever -b ou --borderType seguido de uma das bordas selecionadas, indicadas pelo nome ou número de seleção. Como exemplo, para escolher bordas refletidas, utiliza-se -b reflect ou -b 3.

- python3 trabalho1 -i baboon.png -b replicate
- python3 trabalho1 -i city.png --borderType 4

Caso não seja definido o tipo de bordar a ser utilizado, a borda selecionada como padrão é a reflect101.

D. Exemplos de Execução

Para facilitar a execução dos códigos, segue alguns exemplos de linhas de código que pode ser utilizadas no terminal e

TABLE I
RESUMO DOS POSSÍVEIS VALORES PARA PARÂMETROS

Parâmetros	Palavra	Possibilidades	Opcionalidade
(INPUT)	-i --ifile	Nome da imagem de entrada (.png)	Obrigatório
[OUTPUT]	-o --ofile	Nome da imagem de saída	Opcional
Opções	-k --kernel	Filtro desejado: h1, h2, h3, h4, h5, h6, h7, h8, h9, h10 e h11.	Opcional
	-b --borderType	Tipo política de borda: constant (1), replicate (2), reflect (3), reflect101 (4) e isolated (5)	Opcional

algumas utilizações de parâmetros opcionais.

Filtro h1 e Borda tipo reflexão:

- python3 trabalho1.py -i baboon.png -k h1 -b reflect

Filtro h3 e Borda tipo isolada:

- python3 trabalho1.py -i city.png --kernel h3 -b 5

Filtro h1 e Borda tipo reflexão 101:

- python3 trabalho1.py -i baboon.png --borderType 4

Filtro h1 e Borda tipo reflexão 101:

- python3 trabalho1.py -i butterfly.png

Filtro h4 e Bordar tipo isolada:

- python3 trabalho1.py -i butterfly.png --kernel h4 --borderType isolated

III. IMPLEMENTAÇÃO E DECISÕES TOMADAS

Nesta seção será descrito o funcionamento dos algoritmos do trabalho 0 e alguns aspectos mais específicos de implementação e algumas decisões importantes para esta implementação específica.

De forma geral algumas decisões foram tomadas:

- Todas as entradas aceitam imagens coloridas, porém estas imagens são convertidas para escala de cinza.
- Imagens de diferentes tamanhos podem ser utilizadas. Não necessariamente matrizes quadradas ou com valores pares em linhas e colunas
- Os valores dos parâmetros opcionais, quando não declarados, exibem a imagem com valores padrões.
- Sempre que possível buscou-se utilizar métodos e equações em formato vetorizado.
- Para implementação deste trabalho, utilizou-se os métodos de aplicação de filtros da biblioteca opencv-python.

A. Filtros

Como comentado anteriormente os filtros utilizados neste trabalho foram aplicados com biblioteca opencv-python, através do método cv2.filter2D(), porém cada aspecto de variação da imagem e utilização das aplicações foram estudadas e consideradas.

Todos os filtros propostos foram implementados em forma de matriz e disponibilizados para o método cv2.filter2D(), considerando-se as rotações necessárias para convolução no opencv, juntamente com os parâmetros de borda, descritos na seção Tipos de Bordas (III-B). Os efeitos encontrados e observados serão descritos na seção de Resultados (IV)

B. Tipos de Bordas

Um dos desafios deste trabalho é definir e entender como aplicar filtros em uma imagem. Entre as dificuldades que podem ser encontradas, tem-se como lidar com os limites da imagem e a política que será adotada para trabalhar com estes valores indefinidos, uma vez que o filtro terá uma de suas partes fora da imagem em questão, como pode ser observado na imagem

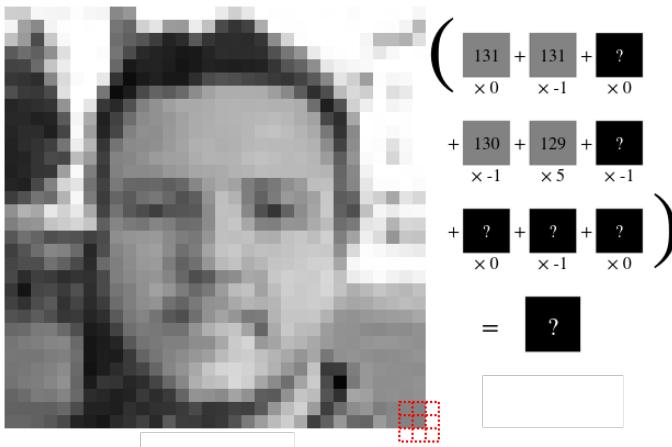


Fig. 1. Máscara nas Bordas de uma Imagem

Para solucionar este problema, utilizou-se algumas abordagens diferentes. Entre as opções escolhidas para testes, estão:

- **Bordas Constantes:** Nesta técnica, ao calcular-se os valores de um pixel de borda, todos os valores indefinidos da máscara são dados por um valor i específico.
- **Bordas de Replicações:** Neste contexto, os pixels faltantes são substituídos por replicações do pixel de borda vizinho.
- **Bordas Refletidas:** Aqui, os pixel faltantes são substituídos por o reflexo dos pixel vizinhos, ou seja, caso tenhamos os pixel vizinhos como a , b , c e d os pixel faltantes são substituídos em uma mascara de tamanho 5×5 , por exemplo, por c , b , a na borda esquerda .
- **Bordas Refletidas 101:** Este caso é parecido com o caso anterior, das bordas refletidas, porém o valor do pixel da borda não é refletido para os novos valores, neste caso, para o exemplo anterior, o resultado seria d , c e b .
- **Bordas Isoladas:** Não observa os valores fora da área de interesse (ROI)

Alguns tipos de bordas não foram aplicadas neste trabalho, devido a grande quantidade já escolhidas para demonstração, mas acredita-se ser importante mencioná-las, estas bordas podem ser implementadas a partir de outras bibliotecas e seus métodos respectivos, como por exemplo o `scipy.signal.convolve2d()` ou `numpy.convolve()`:

- **Borda Wrap:** Nesta política de borda os pixels utilizados para a equação com o filtro é correspondente aos pixels da borda oposta, desta forma imagina-se a imagem como um ciclo.
- **Borda Transparente:** Significa que os pixels na imagem de destino que corresponde aos "outliers" na imagem de origem não são modificados pela função.

TABLE II
COMPORTAMENTO DOS DIFERENTES TIPOS DE FILTROS

Filtro	Comportamento de Utilização *
Bordas Constantes	$iiiiii abcdefg iiiiii$ Para um determinado i
Bordas de Replicações:	$aaaaaa abcdefg hhhhhh$
Bordas Refletidas	$fedcba abcdefg hgfedcb$
Bordas Refletidas 101	$gfedcb abcdefg gfedcb$
Bordas Isoladas	Não observa fora da ROI
Borda Wrap	$cdefgh abcdefg abcdefg$
Borda Transparente	$uvwxyz abcdefg ijklmno$

* Bordas representadas por |

IV. RESULTADOS

Visto que os resultados obtidos são muito visuais, as imagens geradas serão apresentadas para cada máscara utilizada. Todas as imagens aqui apresentadas serão imagens de saída dos algoritmos utilizados. Na segunda subseção também serão comparadas as diferentes tipos de políticas de borda.

A. Filtros

Para que estes resultados sejam comparáveis entre si, todas as imagens nesta subseção IV-A utilizaram a mesma política de borda, neste caso a Borda Refletida 101.

O primeiro filtro aplicado, $h1$, pode ser observado na Figura 2. Neste imagem pode-se perceber um grande realce nas bordas dos elementos de uma imagem digital em que a intensidade luminosa muda repentinamente. Para obter este efeito utilizou-se uma operação Laplace Gaussiana que diferente da Laplace utilizado sozinha, tem uma menor sensitividade a ruídos.

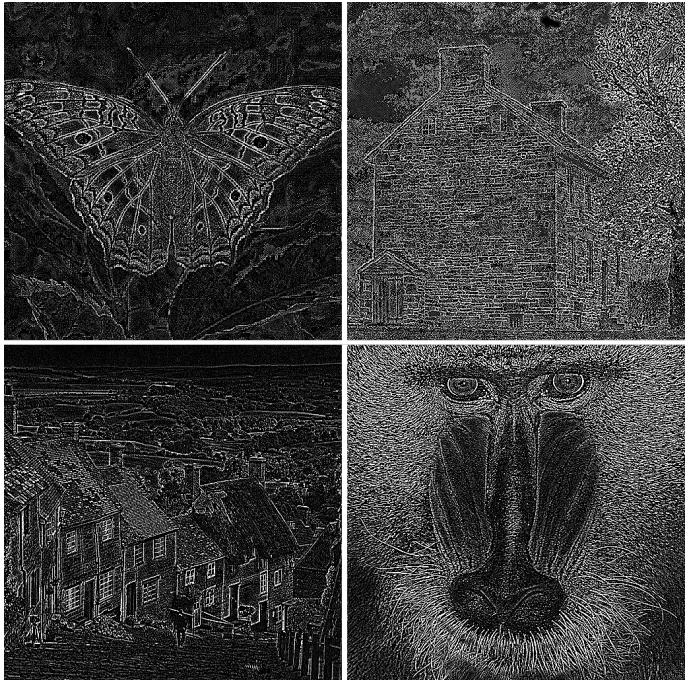


Fig. 2. Filtro $h1$. Imagens (a) butterfly.png, (b) house.png, (c)city.png e (d) baboon.png

No segundo grupo de imagens, Figura 3, utilizou-se o filtro h2, que representa uma operação de Gaussian Blur, a partir de uma máscara de operação Gaussiana, utilizada para redução de ruídos e detalhes.



Fig. 3. Filtro h2. Imagens (a) house.png, (b) baboon.png, (c)butterfly.png e (d) seagull.png

O terceiro filtro utilizado, h3, representa um operador de Sobel para direita, utilizado para detecção das bordas dos elementos de uma imagem, Figura 4. Este operador tem um efeito de suavização, portanto, são menos afetados pelo ruído. contendo um componente horizontal e um componente vertical.

Assim como apresentado anteriormente no filtro h3, o filtro h4 também representa um operador de Sobel, Figura 5, porém uma operação Sobel para baixo, tendo o mesmo intuito de utilização do filtro anterior.

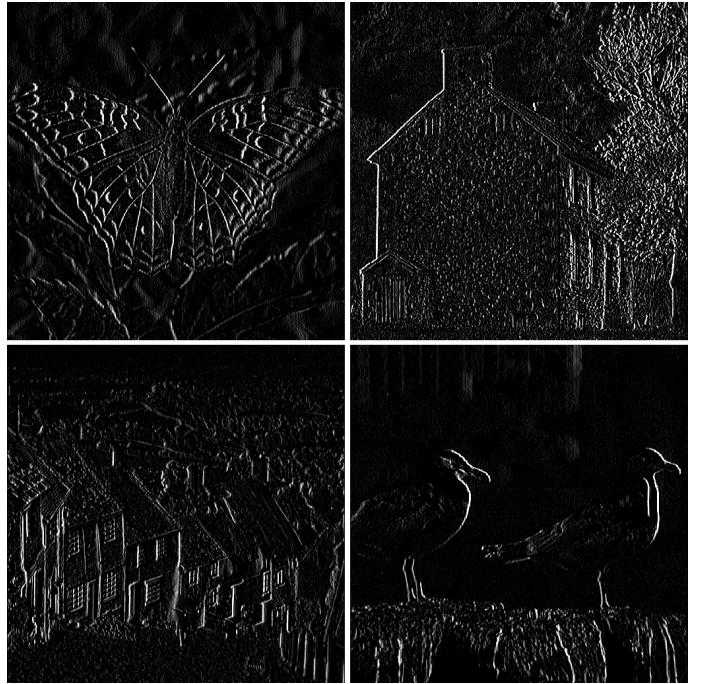


Fig. 4. Filtro h3. Imagens (a) butterfly.png, (b) house.png, (c)city.png e (d) seagull.png

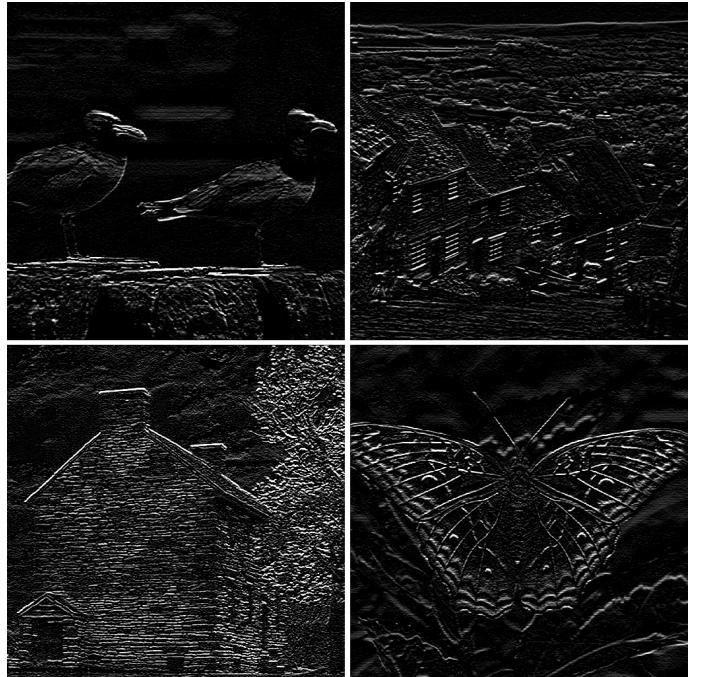


Fig. 5. Filtro h4. Imagens (a) seagull.png, (b) city.png, (c)house.png e (d) butterfly.png

O filtro apresentado pela máscara h5, Figura é uma operação de Laplace com inclusão de diagonais, este filtro é utilizado para detecção de bordas dos elementos de uma imagem, Figura 6, assim como os filtros h3 e h4, porém o h5 é mais sensível a ruídos do que os anteriormente apresentados.



Fig. 6. Filtro h5. Imagens (a) city.png, (b) butterfly.png, (c)seagull.png e (d) house.png

Este filtro h6 de convolução tem um efeito de média Figura 7. Então você acaba com um leve borrão. Observa-se que a soma de todos os elementos desta matriz é 1,0. Isso é importante. Se a soma não for exatamente um, a imagem resultante será mais clara ou mais escura.

Ao utilizar a máscara h7 pode-se detectar linhas, mais especificamente esta máscara detecta linhas em 45° .



Fig. 7. Filtro h6. Imagens (a) baboon.png, (b) seagull.png, (c)butterfly.png e (d) house.png

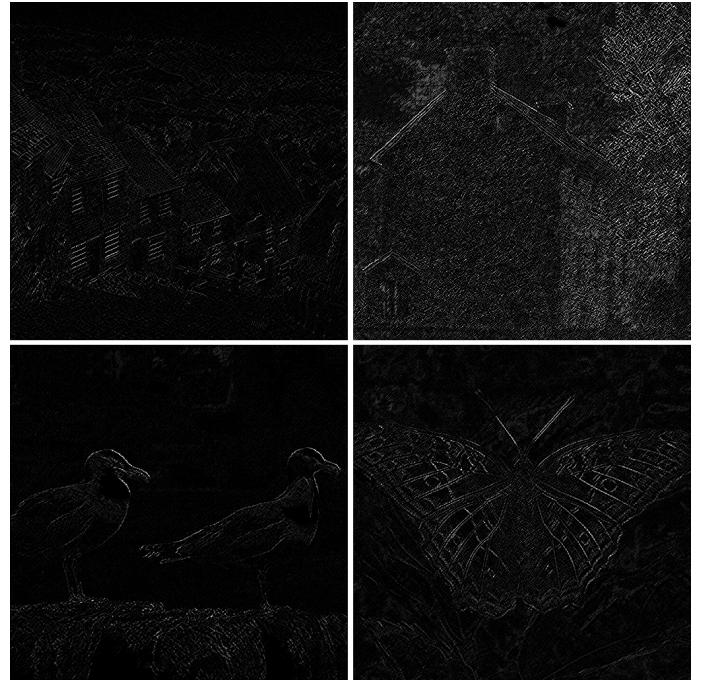


Fig. 8. Filtro h7. Imagens (a) city.png, (b) house.png, (c)seagull.png e (d) butterfly.png

O filtro aplicado com h8, assim como no h7 também é uma máscara de detecção de linhas, porém para detecção de linhas à 135° , Figura 9.



Fig. 9. Filtro h8. Imagens (a) butterfly.png, (b) city.png, (c)house.png e (d) seagull.png

Assim como a máscara h6, a máscara h9 aplica um filtro de média na imagem, causando um efeito parecido com o visto anteriormente, porém como pode ser visto na Figura 10, esta média é realizada apenas com os pixels na diagonal do pixel atual.

No conjunto de imagens da Figura 11 foram obtidos a partir do uso da máscara h10 que aplica um efeito de nitidez na imagem. Observa-se que a máscara realiza a aplicação dos pixels borrados, e alguns o valor negativo, da imagem original



Fig. 10. Filtro h9. Imagens (a) seagull.png, (b) butterfly.png, (c)baboon.png e (d) house.png

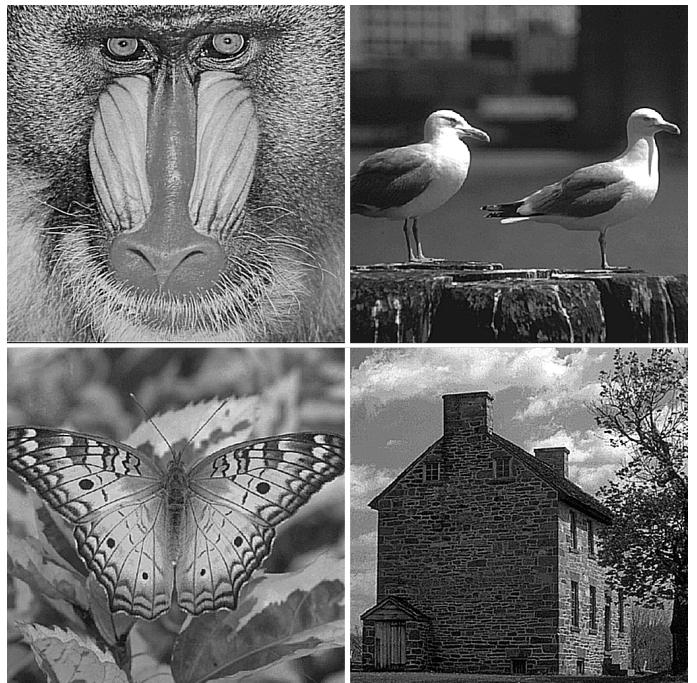


Fig. 11. Filtro h10. Imagens (a) baboon.png, (b) seagull.png, (c)butterfly.png e (d) house.png

Observa-se com a aplicação do filtro h11, a partir do conjunto de imagens da Figura 12, um realce dos objetos que estão na cena.

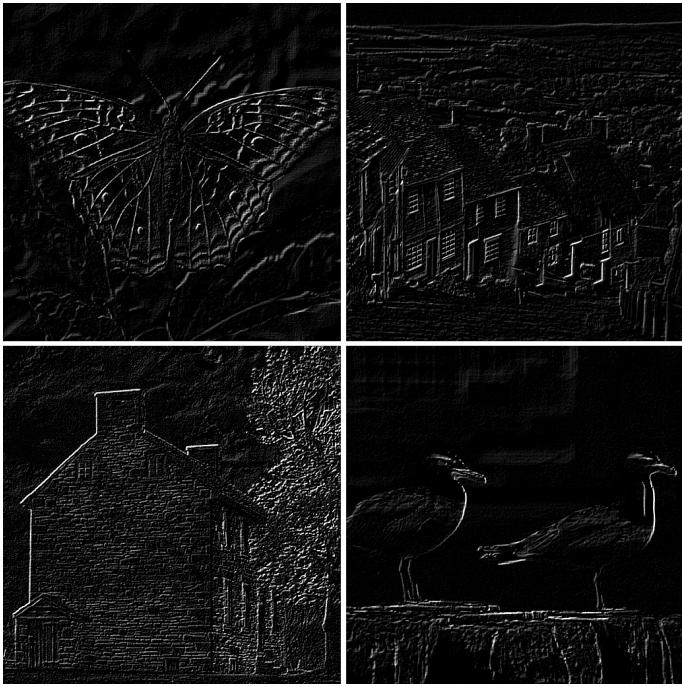


Fig. 12. Filtro h11. Imagens (a) butterfly.png, (b) city.png, (c)house.png e (d) seagull.png

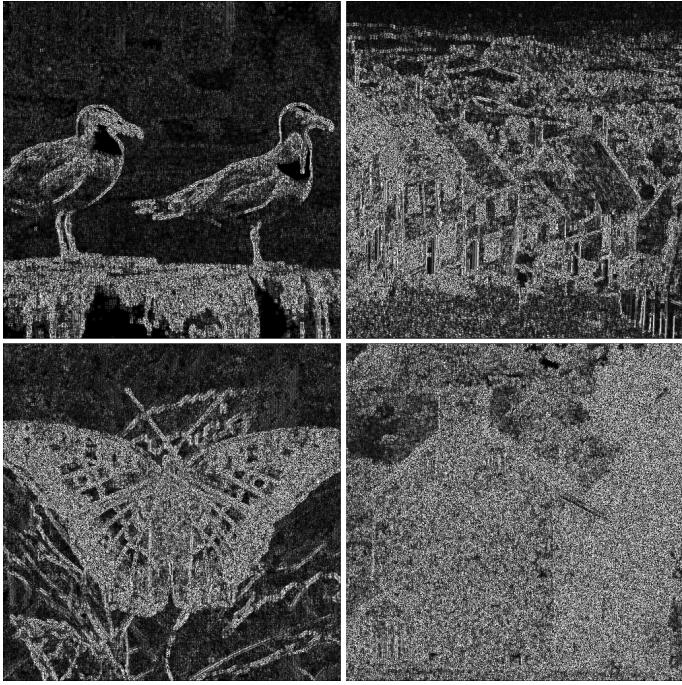


Fig. 13. Filtro h3h4. Imagens (a) butterfly.png, (b) city.png, (c)house.png e (d) seagull.png

Finalmente, ao utilizar a opção de máscaras h3 e h4 em conjunto, através da soma destas duas máscaras, utilizando a expressão: $\sqrt{(h3)^2 + (h4)^2}$. Observa-se uma combinação dos resultados obtidos através das máscaras anteriores. Pode-se

observar que as áreas com tons de cinza homogêneos não estão destacados, porém mudanças nestes tons têm um grande realce na imagem, limitando as fronteiras destes tons

B. Bordas

Nesta subseção, será apresentado as diferentes políticas de bordas que foram observadas na implementação, de acordo como descrito anteriormente, na Tabela II. Da mesma forma que na seção anterior, para facilitar a comparação, utilizaremos o mesmo filtro em todos os testes para que as bordas possam ser observadas sem outras variações.

Visto que a escolha da política de borda depende inteiramente da imagem e dos efeitos que deseja-se obter, as imagens de comparações serão somente demonstradas, sem uma descrição complementar para cada imagem. Caso desejado, a explicação do funcionamento e padrão de cada tipo de borda encontra-se na subseção II-C e na Tabela II.

Seguem as seguintes imagens:

- Figura 14: Borda Constante, na imagem City.png. Imagem (a) Filtro h1 e (b) Filtro h2.
- Figura 15: Borda com Replicações, na imagem City.png. Imagem (a) Filtro h1 e (b) Filtro h2.
- Figura 16: Bordas Refletidas, na imagem City.png. Imagem (a) Filtro h1 e (b) Filtro h2.
- Figura 17: Bordas Refletidas 101, na imagem City.png. Imagem (a) Filtro h1 e (b) Filtro h2.
- Figura 18: Borda Isoladas, na imagem City.png. Imagem (a) Filtro h1 e (b) Filtro h2.

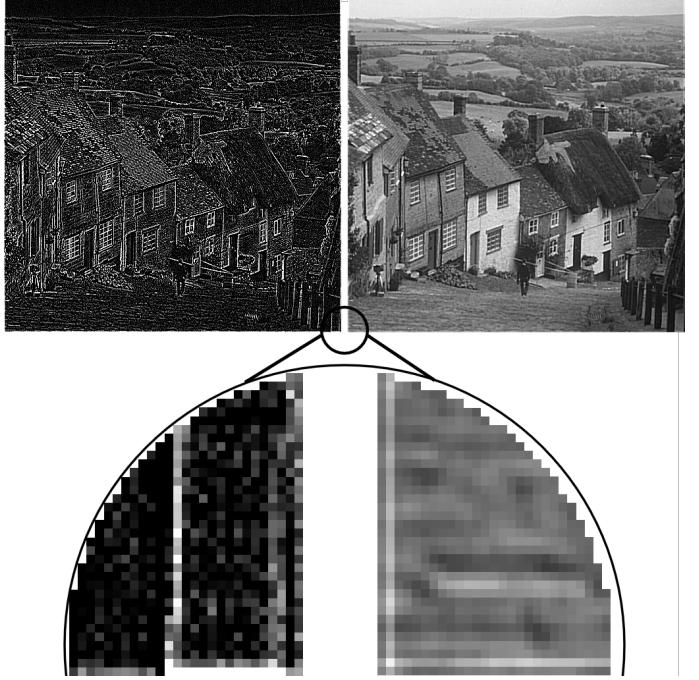


Fig. 14. Borda: Constante. (a) Filtro h1 e (b) Filtro h10

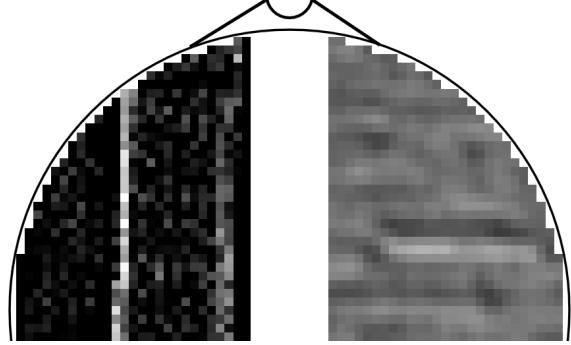


Fig. 15. Borda: Replicações. (a) Filtro h1 e (b) Filtro h10

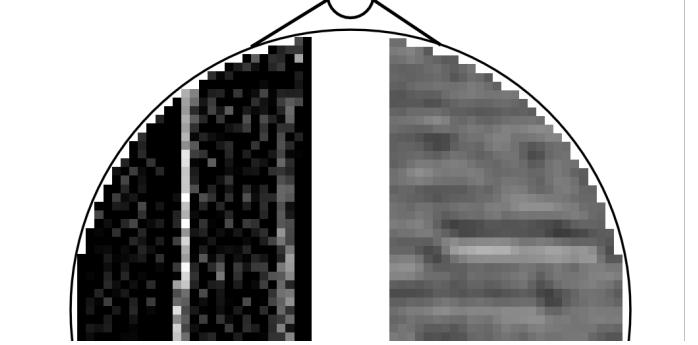


Fig. 17. Borda: Refletidas101. (a) Filtro h1 e (b) Filtro h10

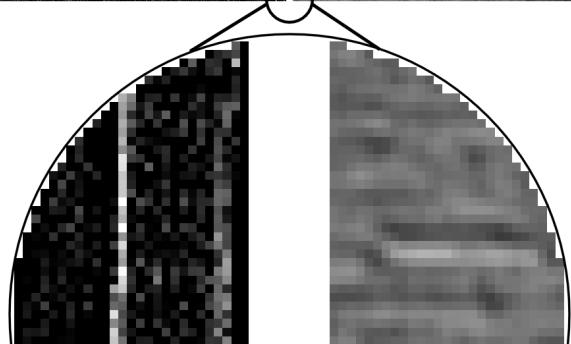


Fig. 16. Borda: Refletidas. (a) Filtro h1 e (b) Filtro h10

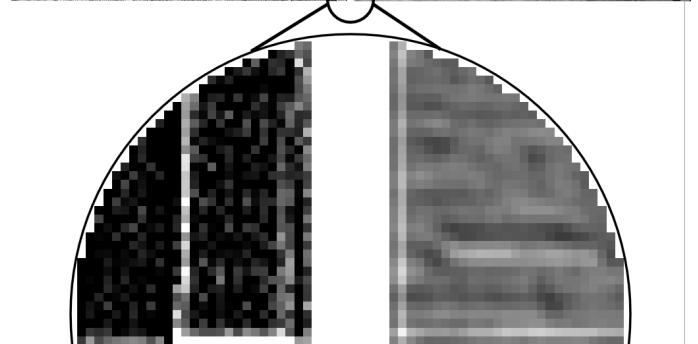


Fig. 18. Borda: Isoladas. (a) Filtro h1 e (b) Filtro h10

V. CONCLUSÕES

A partir deste trabalho, teve-se a oportunidade de trabalhar com diferentes tipos de Filtros para imagens, através da aplicação de uma máscara de convolução em diferentes imagens. Também realizou-se um estudo e comparação em relação as diferentes políticas de bordas.

Espera-se que a partir deste trabalho, o entendimento e aplicação de futuras áreas como convolução em imagens seja melhor compreendida e facilitada, além de aprofundar-se em detalhes de implementação destes modelos, como por exemplo, as dificuldades que podem ser encontradas em bordas.