

MC920 - Trabalho 1

Yuri Luiz de Oliveira
188802

22 de Outubro de 2020

1 Introdução

O trabalho consiste em filtrar imagens digitais monocromáticas aplicando a operação de convolução de algumas máscaras propostas e analisar seus resultados.

1.1 Máscaras

No total há onze máscaras propostas, representadas por matrizes de números reais, de tamanhos distintos e com efeitos diferentes na imagem filtrada.

$$h_1 = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}, \quad h_2 = \frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix},$$

$$h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad h_4 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \quad h_5 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix},$$

$$h_6 = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad h_7 = \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}, \quad h_8 = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix},$$

$$h_9 = \frac{1}{9} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$h_{10} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & 2 & 2 & 2 & -1 \\ -1 & 2 & 8 & 2 & -1 \\ -1 & 2 & 2 & 2 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix},$$

$$h_{11} = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Além de aplicar as máscaras acima, é necessário calcular o resultado da combinação das máscaras h_3 e h_4 , através da expressão

$$\sqrt{{h_3}^2 + {h_4}^2}$$

1.2 Imagens

Todas as imagens abaixo foram utilizadas para testes, entretanto, somente as imagens baboon.png e geometry.png foram utilizadas para exemplos neste relatório.

1.2.1 baboon.png

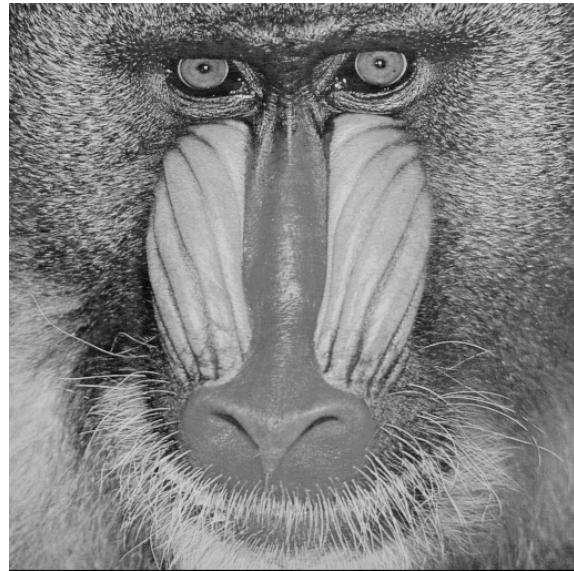


Figure 1: Imagem retirada de https://www.ic.unicamp.br/~helio/imagens_.png/baboon.png

1.2.2 butterfly.png

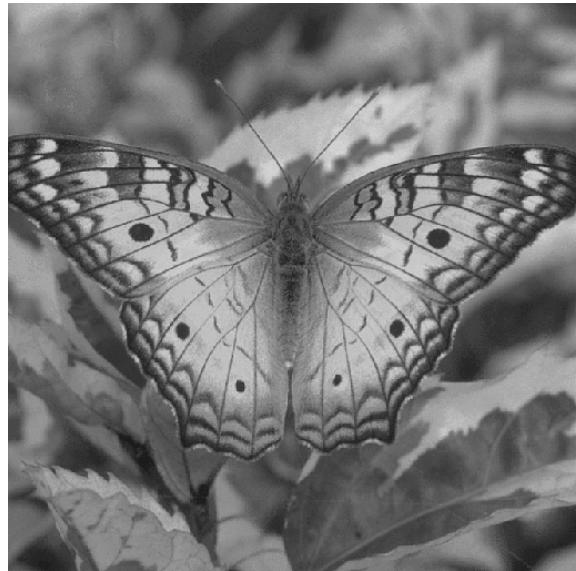


Figure 2: Imagem retirada de https://www.ic.unicamp.br/~helio/imagens_.png/butterfly.png

1.2.3 city.png



Figure 3: Imagem retirada de https://www.ic.unicamp.br/~helio/imagens_png/city.png

1.2.4 geomtry.png

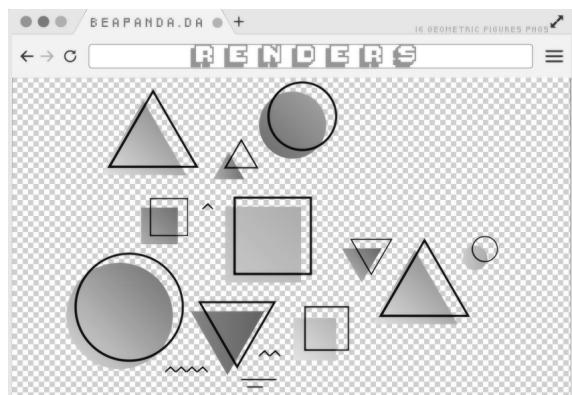


Figure 4: Imagem retirada de <https://www.deviantart.com/beapanda/art/Renders-296-Geometric-Figures-Pngs-800889683>

1.2.5 house.png

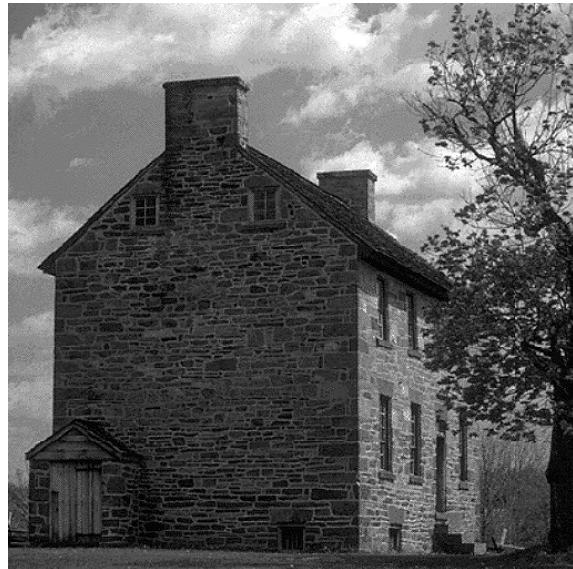


Figure 5: Imagem retirada de https://www.ic.unicamp.br/~helio/imagens_.png/house.png

1.2.6 lena.png



Figure 6: Imagem retirada de https://www.researchgate.net/figure/The-famous-Lena-image-often-used-as-an-example-in-image-processing_fig1_235673010

1.2.7 seagull.png



Figure 7: Imagem retirada de https://www.ic.unicamp.br/~helio/imagens_png/seagull.png

2 Ambiente de desenvolvimento e testes

A solução para o trabalho foi implementada e testada em Python 3.8.5, com auxílio das bibliotecas numpy, opencv, matplotlib e argparse. O sistema operacional utilizado foi Ubuntu 20.04.

2.1 Preparando o ambiente

Para instalar as bibliotecas necessárias basta entrar na pasta do trabalho e rodar o comando

```
$ pip3 install -r requirements.txt
```

3 Rodando a solução

A solução aceita argumentos de linha de comando e deve ser executada através do interpretador de python. O seguinte comando filtra a imagem no caminho *"images/baboon.png"* com a máscara *h1* e salva o resultado no arquivo de caminho *"out/baboon_h1.png"*

```
$ python3 image_filter.py -i images/baboon.png -m h3 -o out/baboon_h3.png
```

Os argumentos `-i` e `-m` são obrigatórios para o funcionamento do programa. A opção `-o` não é obrigatória e, se não for passada, o resultado não será salvo (somente mostrado na janela criada pelo programa).

3.1 Salvando a imagem combinada de h3 e h4 em um arquivo

É possível salvar a imagem resultado da combinação das imagens filtradas por h_3 e h_4 utilizando o argumento `-oc`. Veja um exemplo de como utilizar esta opção:

```
$ python3 image_filter.py -i images/baboon.png -m h3 -oc out/baboon_h3_h4.png
```

Note que o diretório `"out"` precisa existir.

3.2 Outros argumentos de execução

O programa aceita outros comandos de execução. Para ver a lista completa de argumento e seus usos, rode o comando

```
$ python3 image_filter.py --help
```

3.3 Comando para aplicar todas as máscaras a uma imagem específica

Para obter os resultados de todas as máscaras aplicadas a uma imagem específica utilize o comando abaixo. Note que **a imagem não deve ser passada com extensão** (a extensão será assumida como PNG) e a imagem deve estar contida em um diretório `"images"` no diretório atual. O exemplo abaixo executa as máscaras $h_1..h_{11}$ (e a combinação de h_3 e h_4) à imagem no caminho `"images/baboon.png"` e salva no diretório `"out"`.

```
$ ./apply_all_filters.sh baboon out/
```

3.4 Formato dos dados

A implementação e testes foram feitas com imagens PNG e não há garantia de funcionamento com outros formatos, embora não haja nenhum bloqueio em relação à extensão do arquivo. As imagens são matrizes de inteiros sem sinal de 8 bits, interpretadas como imagens em escala de cinza. Quando necessário utilizar tipos diferentes, a imagem final sofre conversão para inteiros sem sinal de 8 bits.

4 Detalhes e decisões de implementação

Os filtros foram aplicados através do método `filter2D` da biblioteca opencv. Este método requere a imagem de origem, a profundidade desejada para a imagem

de destino e a máscara. Para a profundidade desejada foi passado o valor `-1` que indica que a imagem de destino terá a mesma profundidade da imagem de origem.

4.1 Tratamento de borda

Como as máscaras tem dimensões maiores do que 1, é necessário lidar com a borda da imagem, para que não ocorra acesso a posições inválidas. É possível configurar como será tratada a borda ao utilizar a função `filter2D` através do parâmetro opcional `borderType`. O tratamento de borda utilizado nesta solução foi replicar o pixel mais próximo à borda. O resultado deste tratamento é `aaaaaa—abcdefgh—hhhhhh`, de acordo com a documentação do opencv.

4.2 Combinação dos filtros h3 e h4

A combinação dos filtros h3 e h4 foi implementada aplicando a expressão

$$\sqrt{h_3^2 + h_4^2}$$

np.uint8), ao calcular os pixels, os pixels maiores do que 255 irão sofrer overflow e seu novo valor será

$$\sqrt{p_3^2 + p_4^2} \mod 255$$

, sendo p1 o pixel da imagem filtrada por h3 e p2 o pixel da imagem filtrada por h4. Como visto na Figura 8, a imagem resultante ficou muito escura, já que o maior valor possível para a expressão é de aproximadamente 360, quando ambos os pixels originais são 255. Desta forma,

$$360 \mod 255 = 105$$

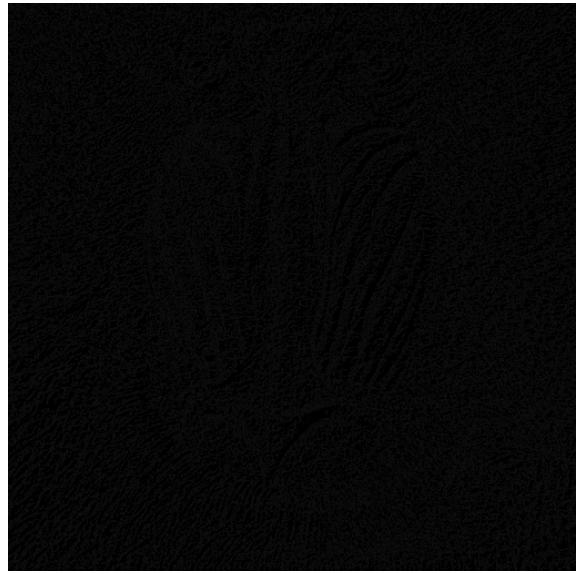


Figure 8: Imagem resultante da combinação das máscaras h_3 e h_4 utilizando inteiros sem sinal de 8 bits.

Para resolver o problema de overflow, as imagens foram convertidas para matrizes de *float*, tendo como resultado um número real, cuja parte inteira possui um tamanho suficiente para efetuar a operação sem ocorrer overflow. Obtendo como resultado a figura 2.

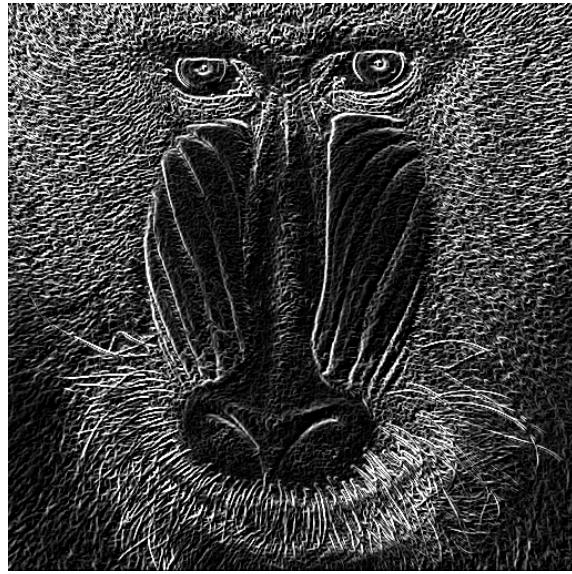


Figure 9: Imagem resultante da combinação das máscaras h3 e h4 sem normalização para o intervalo [0..255]

Por fim, dado que os pixels resultantes da combinação das imagens podem ser maiores do que 255, é necessário normalizar a imagem para o intervalo [0..255] e converter de volta para números inteiros de 8 bits, pois este é o formato adotado na implementação. Após a normalização (Figura 3) percebemos que os brancos da imagem ficam um pouco menos realçados.

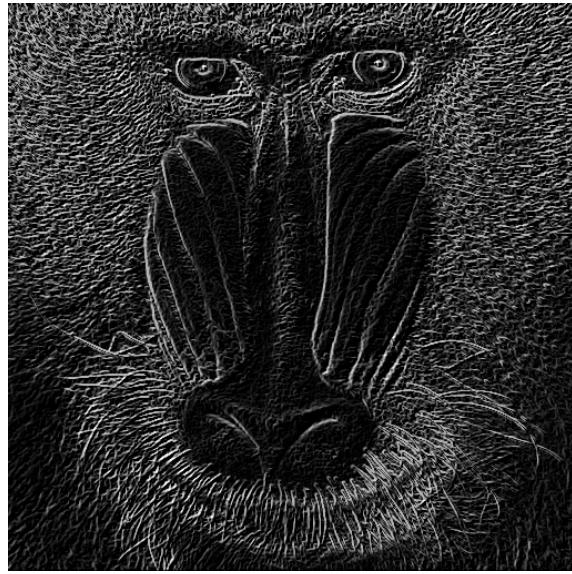


Figure 10: Imagem resultante da combinação das máscaras h3 e h4

5 Resultados obtidos

Vamos analisar o resultados das imagens filtradas por cada uma das máscaras propostas. Para cada máscara uma imagem conveniente foi selecionada. Para analisar máscaras que realçam linhas, foram utilizadas imagens de figuras geométricas, por exemplo.

5.1 h_1

A máscara h_1 realça bordas na imagem, como visto na figura 11.

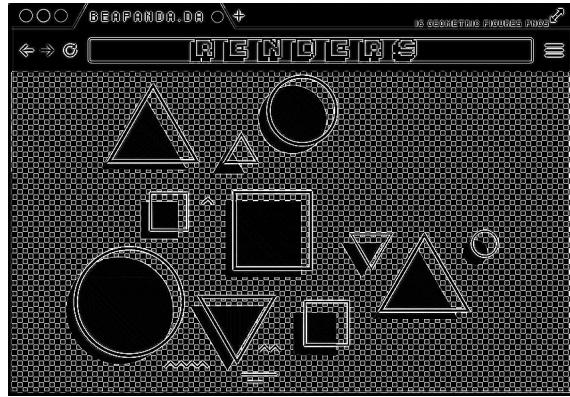


Figure 11: Imagem filtrada com a máscara h_1

5.2 h_2

A máscara h_2 borra levemente a imagem, como visto na figura 12

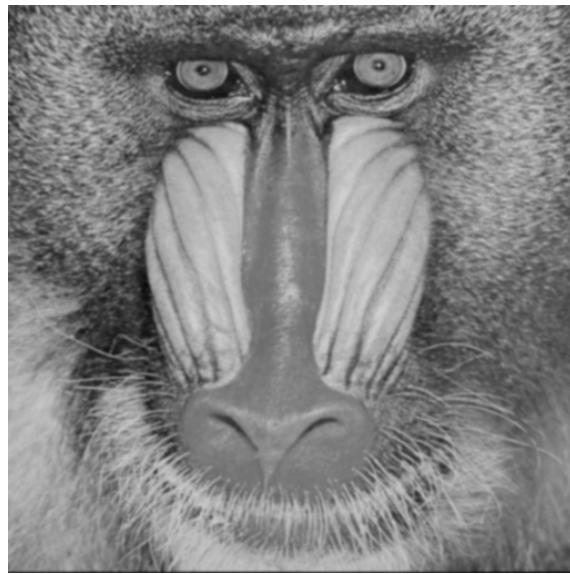


Figure 12: Imagem filtrada com a máscara h_2

5.3 h_3

A máscara h_3 realça as bordas verticais da imagem, como visto na figura 13.

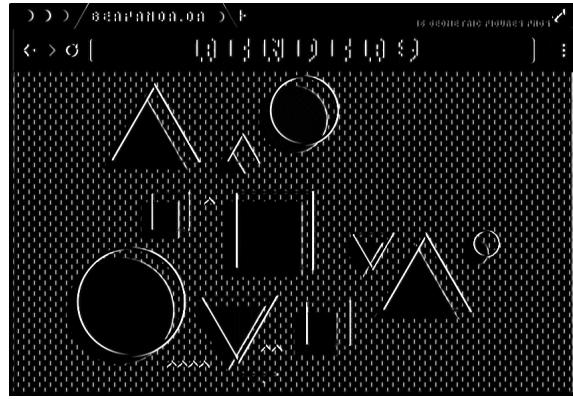


Figure 13: Imagem filtrada com a máscara h_3

5.4 h_4

A máscara h_4 realça as bordas horizontais da imagem, como visto na figura 14.

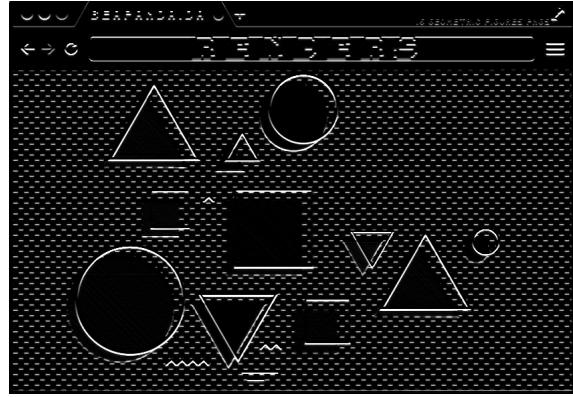


Figure 14: Imagem filtrada com a máscara h_4

5.5 h_5

A máscara h_5 realça as bordas da imagem, como visto na figura 15.

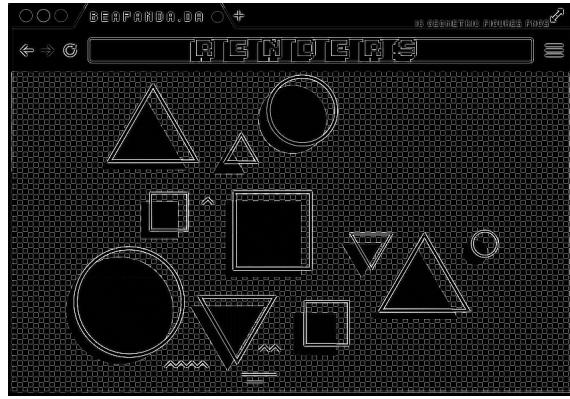


Figure 15: Imagem filtrada com a máscara h_5

5.6 h_6

A máscara h_6 suaviza as diferenças entre pixels, calculando a média entre os vizinhos, como visto na figura 16. Naturalmente, pixels próximos terão a diferença entre suas intensidades de cinzas diminuídas. Uma vez que, se os vizinhos de um pixel são mais próximos do preto, este pixel irá escurecer e o oposto para vizinhos mais próximos do branco.

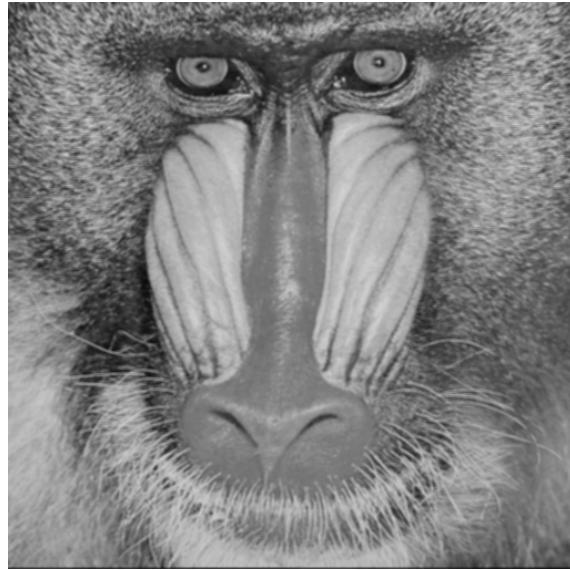


Figure 16: Imagem filtrada com a máscara h_6

5.7 h_7

A máscara h_7 realça as bordas com sentido de sudoeste a nordeste, como visto na figura 17.

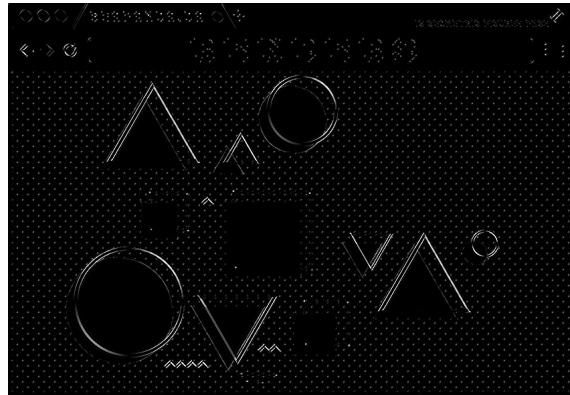


Figure 17: Imagem filtrada com a máscara h_7

5.8 h_8

A máscara h_8 realça as bordas com sentido de sudeste a noroeste, como visto na figura 18.

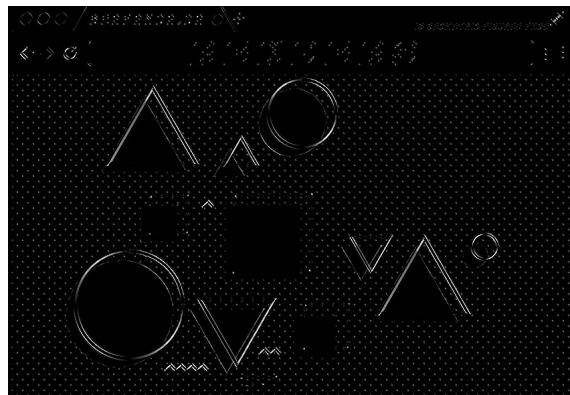


Figure 18: Imagem filtrada com a máscara h_8

5.9 h_9

A máscara h_9 deixa a imagem com efeito de movimento, como se a câmera que tirou a foto tivesse se movido enquanto tirava a foto, como visto na figura 19. É importante notar que este efeito acontece em qualquer imagem, não somente em fotos.



Figure 19: Imagem filtrada com a máscara h_9

5.10 h_{10}

A máscara h_{10} realça os detalhes das imagens, como visto na figura 20. É perceptível como a imagem os traços do rosto do babuíno ficaram mais aparentes na imagem com a máscara aplicada.

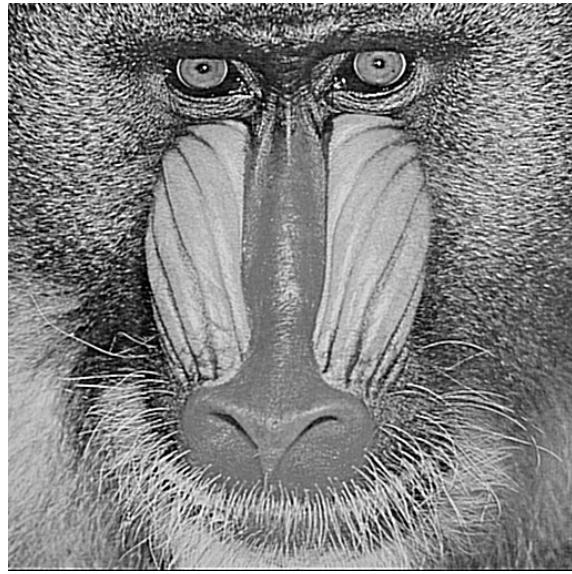


Figure 20: Imagem filtrada com a máscara h_{10}

5.11 h_{11}

A máscara h_{11} realça as bordas pela esquerda e por baixo, como visto na figura 21. Note que isso não significa que serão visíveis somente bordas esquerdas ou abaixo dos componentes da imagem. Um quadrado, por exemplo, pode ter todos seus lados realçados, entretanto, é visível que seus lados são realçados somente pela esquerda e por baixo.

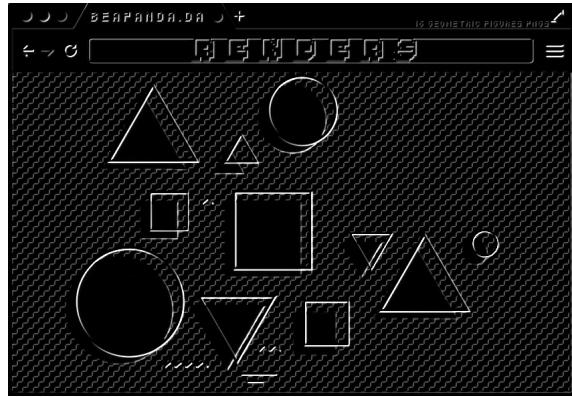


Figure 21: Imagem filtrada com a máscara h_{11}

5.12 $\sqrt{h_3^2 + h_4^2}$

Percebemos, pela Figura 22 que a combinação das máscaras, através da raiz da soma dos quadrados da imagem resultante dos filtros h_3 e h_4 tem como resultado o realce das linhas que cada uma das máscaras realça individualmente.

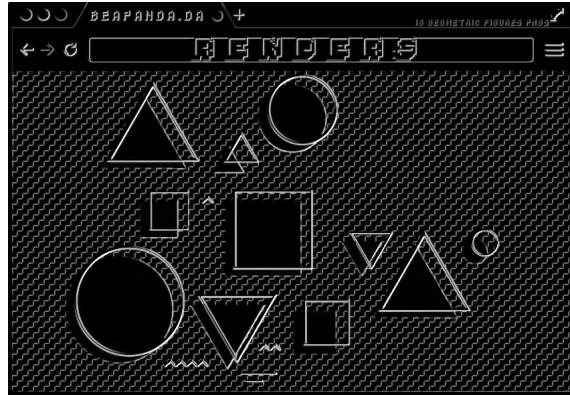


Figure 22: Imagem filtrada da combinação das máscaras h_3 e h_4

6 Limitações

Como as imagens são lidas e tratadas com escalas de cinza em que cada pixel é um inteiro sem sinal de 8 bits, nos limitamos a imagens monocromáticas.

7 Conclusões

O processo de filtragem de imagens é um processo simples, mas possui resultados muito úteis e diversos. É possível desde borrar imagens até realçar características e detalhes das imagens utilizando o mesmo método, alterando somente a máscara utilizada. Isso torna este método muito importante no processamento de imagens, o que se reflete em muito conteúdo sobre o assunto e soluções que já implementam o filtro de imagens em mais de uma biblioteca existente, como o método *gaussian_filter* (para filtros gaussianos) da biblioteca *scipy.ndimage*, além do *filter2D* utilizado neste trabalho.