

# Relatório do Trabalho Prático de PDI

Isaac Nobrega Marinho  
Engenharia da Computação  
Email: isaacmarinho@eng.ci.ufpb.br  
Matrícula: 20190021071

## Resumo

Neste relatório, serão expostos e discutidos os resultados do trabalho prático realizado para a disciplina de Processamento Digital de Imagens(PDI). O trabalho consiste em implementar em código e testar diversas funções de processamento de imagens, especificamente a conversão de uma imagem no espaço de cor RGB para o espaço YIQ, converter uma imagem em seu negativo quanto as bandas R, G e B de uma imagem no espaço de cor RGB e a banda Y do espaço de cor YIQ, correlação entre a imagem e uma máscara passada por arquivo e o filtro da mediana MxN com M e N ímpares. Para atender aos requisitos, foram utilizados os materiais disponibilizados pelo professor no SIGAA como base teórica. Para implementar as funções, foi utilizada a linguagem C++, com o auxílio da biblioteca OpenCV, que disponibiliza dentre outras utilidades, estruturas de dados próprias para o processamento de imagens e funções para ler, salvar e mostrá-las.

## I. INTRODUÇÃO

Imagens são uma parte crucial da vida dos seres humanos em geral, afinal, em pessoas saudáveis, a visão é o sentido mais preciso. O tempo todo somos bombardeados com informação visual, sejam os filmes/series que assistimos, álbuns de fotos que guardam recordações preciosas, propagandas na televisão e em outdoors entre outras coisas. Não é a toa 85% da informação presente na internet está na forma de pixels e 75% dos neurônios sensoriais são dedicados à visão.

No entanto, apesar de ser algo natural para os humanos, "ver" é um processo desafiador de se ensinar para uma máquina, afinal é dependente de diversos fatores, como o contexto da imagem, o ângulo do qual ela foi tirada, da iluminação do ambiente, das limitações e do ruído na ferramenta de captura, do que já se viu na vida entre outros fatores. Considerando o exposto, é notável que a área de visão computacional se beneficia consideravelmente da aplicação em conjunto de técnicas e expertises próprias das áreas de processamento digital e de inteligência artificial.

É notável que nas últimas décadas, a área de visão computacional vem ganhando cada vez mais espaço, especialmente com o advento das redes neurais convolucionais. Diversas aplicações vêm surgindo nesse contexto, variando desde filtros curiosos para aplicativos sociais, até aplicações mais críticas, como detecção de obstáculos para carros autônomos ou interpretação de resultados de exames médicos. Dito isto, o estudo de técnicas de processamento digital de sinais e imagens se faz crucial para a formação de um profissional completo da área, uma vez que o desenvolvimento de habilidades relacionadas a PDI é um ponto de acesso para a área de visão computacional que cresce cada vez mais nos últimos tempos.

O presente trabalho prático se insere nesse contexto a medida que objetiva solidificar os conhecimentos adquiridos e competências desenvolvidas relacionadas às técnicas mais básicas do processamento digital de imagem de maneira prática, implementando "na mão" programas capazes de realizar técnicas clássicas como conversão entre espaços de cores, negativo de uma ou mais bandas de uma imagem, filtros de correlação e expansão por histograma.

## II. MATERIAIS E MÉTODOS

Para atingir os objetivos elucidados na sessão anterior, foi desenvolvido um programa em C++ para implementar funções que realizem as operações desejadas, utilizando-se principalmente de ferramentas da biblioteca OpenCV para abrir, salvar e mostrar as imagens tanto lidas quanto processadas, e também de estruturas de dados como o Mat, que nada mais é que a representação de imagens como uma matriz. Na lista abaixo, estão destacadas as funcionalidades implementadas no programa

- Conversão de Imagem no espaço de cor RGB para o espaço de cor YIQ e vice-versa.
- Negativo de uma imagem em relação aos canais Y, R, G, B e RGB.
- Filtros correlacionais com máscaras de dimensões MxN definidas por meio de arquivos .txt.
- Filtro da Mediana de dimensões MxN definidas no código, sendo M e N ímpares, em cada um dos 3 canais R, G, B e RGB.
- Expansão por Histograma aplicada ao resultado da correlação com o filtro  $\|Sobel\|$ , assim como solicitado nas especificações do trabalho.

Expandindo um pouco mais acerca das funcionalidades supracitadas, todas as funções recebem 2 objetos do tipo Mat por referência, representando a imagem "fonte" ou de entrada, e a imagem de "destino" ou saída. Além disso, foram implementadas duas funções para conversão entre espaços de cor. Uma converte uma imagem no espaço RGB, onde cada pixel possui três canais de 8 bits cada, para uma imagem no espaço YIQ, na qual cada pixel apresenta 3 canais de pontos flutuantes de 32 bits cada, enquanto a outra faz o processo inverso, convertendo uma imagem no espaço YIQ para uma imagem análoga no espaço RGB, de acordo com as relações numéricas disponibilizadas nos materiais fornecidos pelo professor as equações 1, 2 e 3 a

seguir se referem aos cálculos para transformar uma imagem do espaço RGB para o espaço YIQ. Já as equações 4, 5 e 6 se referem aos cálculos realizados para converter uma imagem YIQ em uma imagem RGB.

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad (1)$$

$$I = 0.596 \cdot R - 0.274 \cdot G - 0.322 \cdot B \quad (2)$$

$$Q = 0.211 \cdot R - 0.523 \cdot G + 0.312 \cdot B \quad (3)$$

$$R = Y + 0.956 \cdot I + 0.621 \cdot Q \quad (4)$$

$$G = Y - 0.272 \cdot I - 0.647 \cdot Q \quad (5)$$

$$B = Y - 1.106 \cdot I + 1.703 \cdot Q \quad (6)$$

Já o negativo em Y recebe uma imagem no espaço YIQ, calcula para cada pixel o negativo do canal Y e repete o valor dos outros canais, entregando uma nova imagem no espaço YIQ. de maneira análoga, as funções de negativo em R, G e B recebem uma imagem no espaço RGB e retornam outra imagem no mesmo espaço, só que com o negativo do canal em questão. Por fim, a função de negativo em RGB também recebe uma imagem representada no espaço RBG e retorna uma imagem no mesmo espaço com o negativo em cada uma de suas bandas. O negativo em qualquer uma das bandas implementadas(Y, R, G ou B) é dado pela equação 7 abaixo, onde  $N_b$  é o valor do negativo na banda  $b$  e  $V_b$  é o valor do pixel na banda  $b$ .

$$N_b = 255 - V_b \quad (7)$$

No código, os filtros correlacionais são implementados a partir de uma função de correlação genérica, que lê os argumentos dispostos nos arquivos de especificação das máscaras, verifica se as informações obtidas a partir das especificações da máscara estão consistentes com o esperado, realiza o padding da imagem por meio da extensão por zeros e realiza a correlação por meio do calculo do produto interno entre os pixels da máscara obtida e a vizinhança do pixel alvo na imagem, de acordo com as dimensões e o pivô da máscara, e adicionando o resultado ao offset que também é uma informação presente na especificação da máscara. Vale salientar que duas funções de correlação genérica foram implementadas. Quando o resultado da correlação em um pixel é maior que 255, o valor é saturado para 255 em ambas as funções. No entanto, quando o resultado da correlação em um pixel é menor que 0, em uma versão da correlação genérica, o valor absoluto do pixel é considerado como resultado da correlação e na outra o valor é saturado para 0. Além disso, em ambas, o offset é somado apenas após verificar se o resultado do produto interno é negativo e realizar seu devido tratamento.

Por fim, a função que calcula o filtro da mediana recebe dois parâmetros adicionais além das duas matrizes para a imagem de entrada e a de saída, que são justamente as dimensões M e N do filtro. O filtro consiste em realizar o padding da imagem de acordo com as dimensões M e N, armazenar a vizinhança do pixel num vetor de acordo com as dimensões M e N, ordenar o vetor e modificar o valor do pixel para o valor do elemento do "meio"do vetor da vizinhança ordenado.

Vale salientar que o padding realizado nas operações de correlação genérica e filtro da mediana é feito por meio de uma função, que recebe uma imagem de entrada, uma matriz de destino, as dimensões da máscara aplicada e a posição do pivô. No caso dos filtros correlacionais, as dimensões e o pivô são especificados em seus argumentos, enquanto a posição do pivô do filtro da mediana é calculado a partir das dimensões do filtro, como o valor "no meio", ou seja,  $(M - 1)/2$  e  $(N - 1)/2$ .

Além disso, uma função de expansão por histograma também foi implementada, para que o filtro de  $\|Sobel\|$  possa ser testado de acordo com as especificações do projeto. ela consiste em identificar a amostra, no caso o pixel de maior e de menor valor em cada banda e aplicar a equação 8 abaixo para cada banda de cada pixel, onde  $r$  é o valor da amostra(banda do pixel) analisada,  $T(r)$  é o novo valor da amostra,  $r_{max}$  é o valor máximo que a amostra assume naquela imagem e  $r_{min}$  é o valor mínimo que a amostra assume naquela imagem.

$$T(r) = round\left(\frac{r - r_{min}}{r_{max} - r_{min}}\right) \quad (8)$$

### III. RESULTADOS E DISCUSSÃO

Tanto os resultados quanto o código fonte podem ser verificados no repositório público do projeto, hospedado no github no seguinte domínio: <https://github.com/Isaac-CI/PDI>. Nele, o código fonte está inteiramente no arquivo main.cpp, e o executável já está compilado no arquivo main.exe. As imagens utilizadas nos testes como inputs estão armazenadas na pasta inputs e as imagens resultantes dos processamentos no código estão armazenadas na pasta outputs. Os arquivos contendo as especificações das máscaras testadas estão armazenados no diretório masks.

As figuras 1, 2, 3, 4, 5 e 6 são obtidas respectivamente a imagem original, a imagem original restaurada, após convertê-la para YIQ e voltar para RGB, Negativo na banda Y(voltando para RGB), negativo na banda R, negativo na banda G e negativo na banda B.

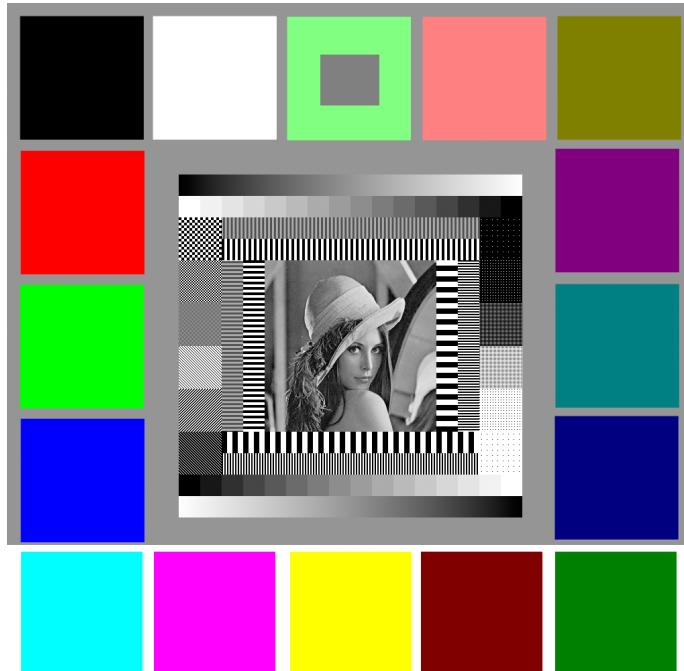


Figura 1. Imagem Original

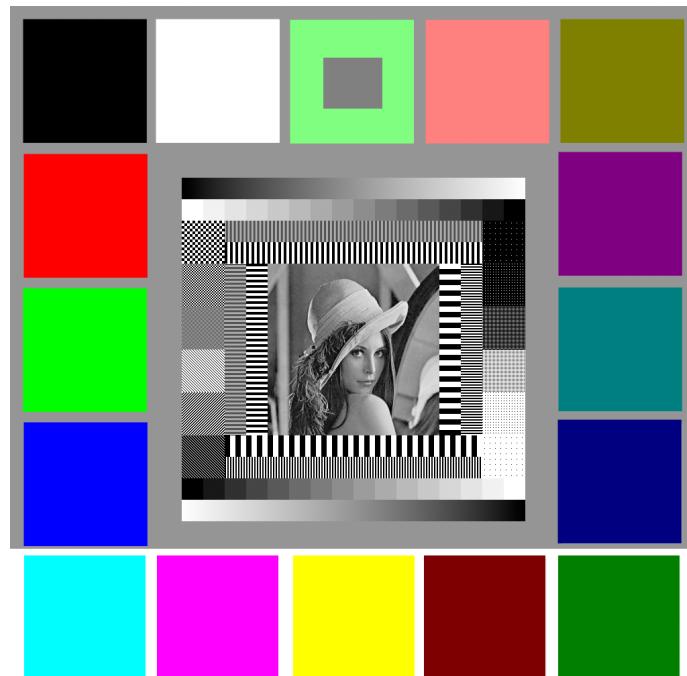


Figura 2. Imagem Restaurada

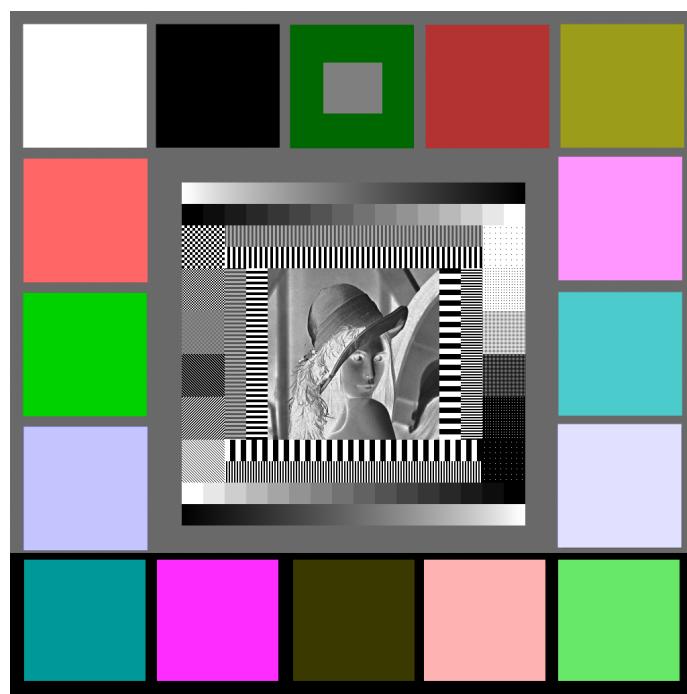


Figura 3. Negativo em Y

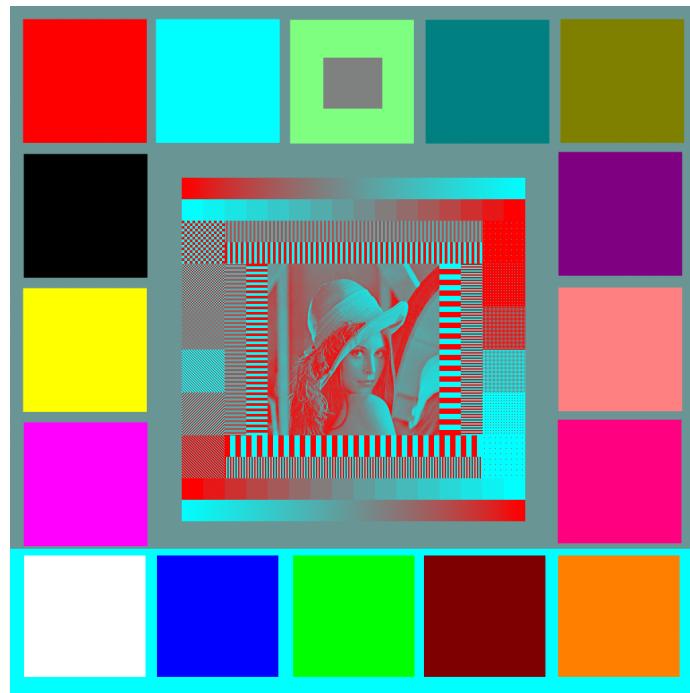


Figura 4. Negativo em R

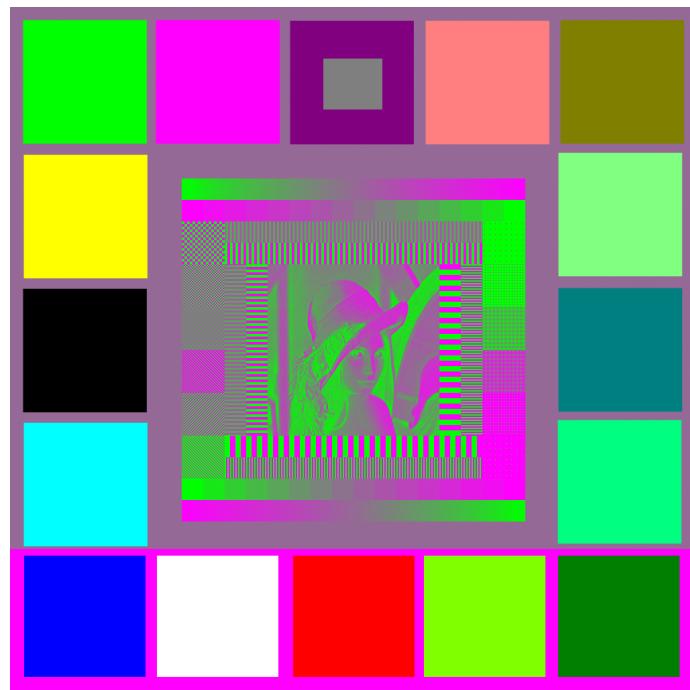


Figura 5. Negativo em G

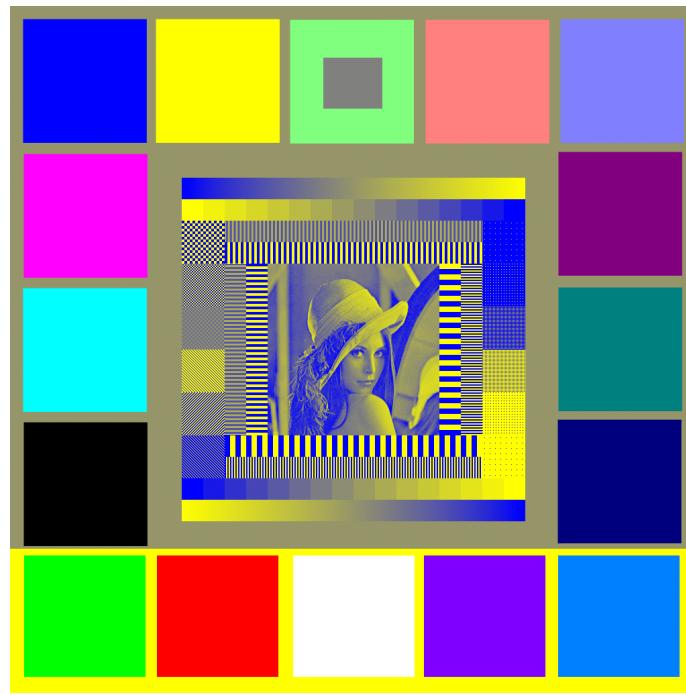


Figura 6. Negativo em B

Note que as imagens original e restaurada são idênticas como deveriam, o negativo em Y preserva a matriz da cor, enquanto inverte o brilho, tornando as cores claras em escuras e as cores escuras em claras, os negativos em R, G e B invertem seus respectivos canais, mantendo os outros inalterados, por exemplo, vermelho puro no negativo em R virou preto, amarelo puro no negativo em G virou vermelho e verde com níveis de cinza por volta de 128 permanece inalterado ne negativo em B, indicando que de fato, as transformações ocorreram sem erros.

Já as imagens 7, 8, 9, 10, 11, 12 a seguir são obtidas respectivamente da aplicação dos filtros correlacionais soma  $4 \times 2$ , Box $11 \times 1$ (Box $1 \times 11$ ), Box  $11 \times 11$ ,  $\|Sobel\|$  vertical sem e com expansão de histograma(ambos com offset de 128, para melhor visualizar o efeito da expansão) e Emboss.

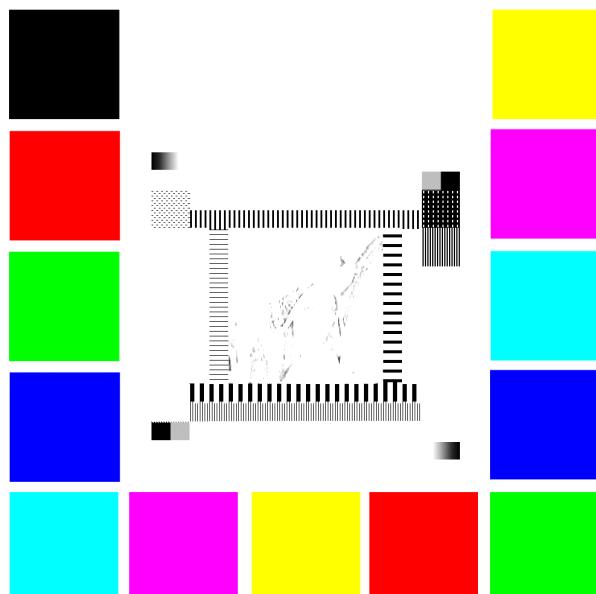


Figura 7. Soma  $4 \times 2$

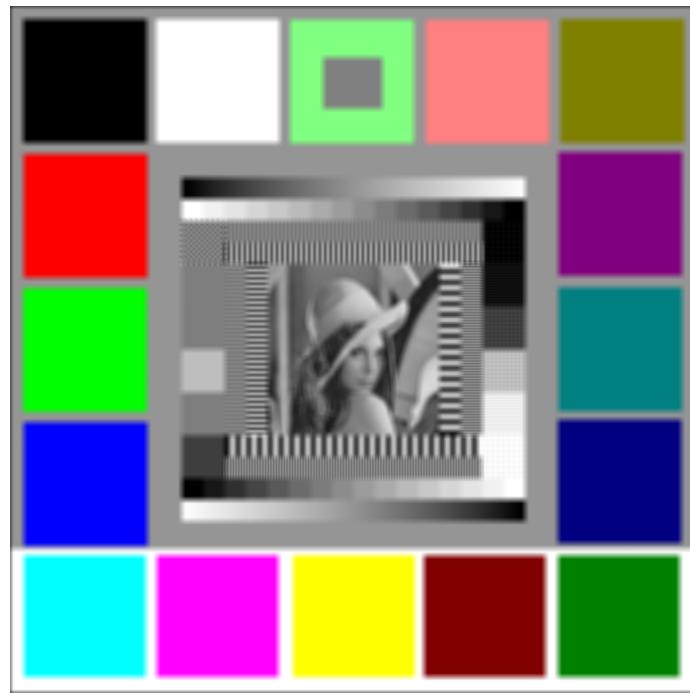


Figura 8. Box11x1(Box1x11)

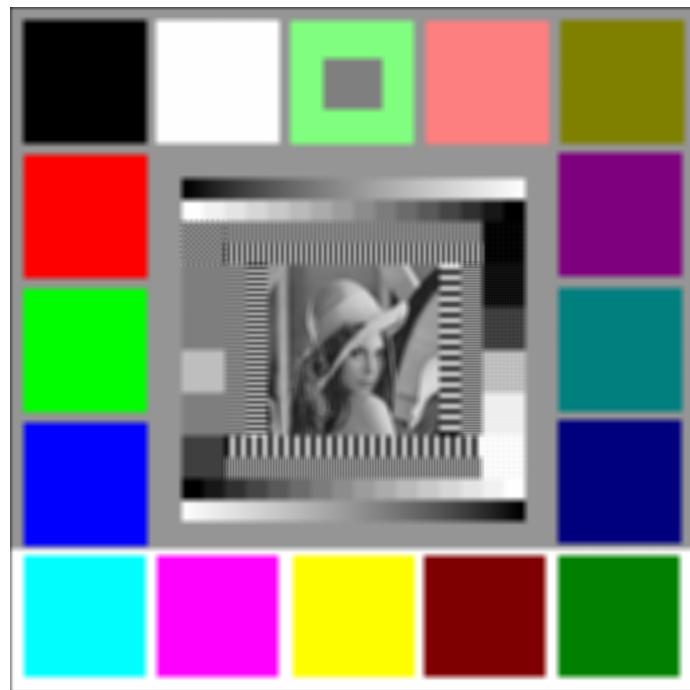


Figura 9. Box 11x11

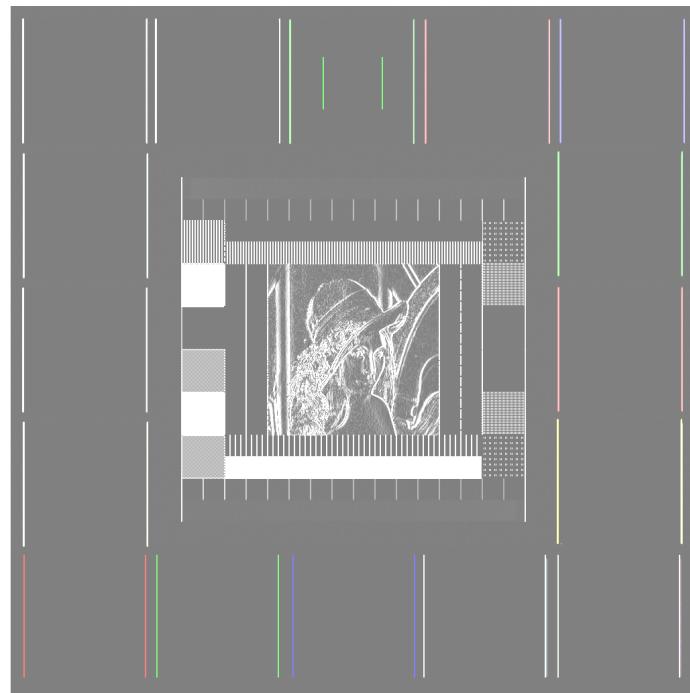


Figura 10. Sobel vertical sem expansão

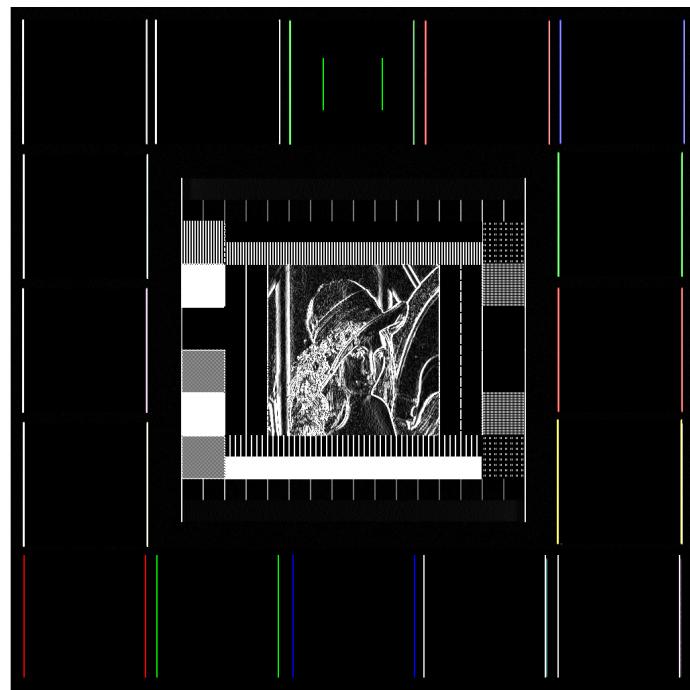


Figura 11. Sobel vertical com expansão

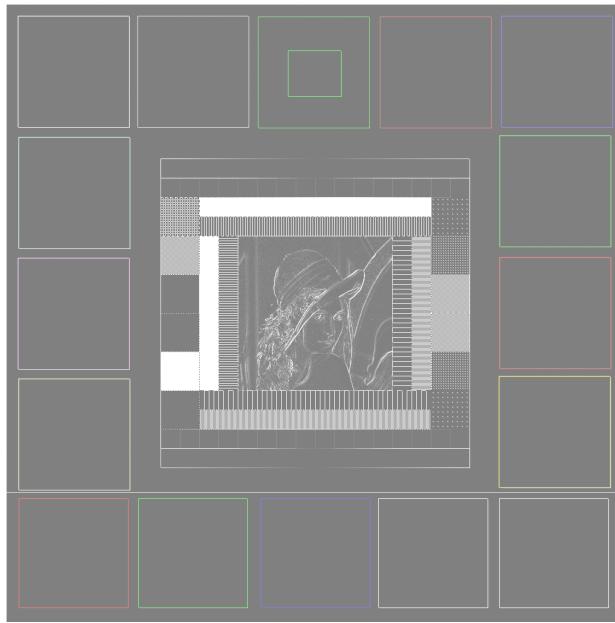


Figura 12. Emboss

Note que como o esperado, o filtro da soma satura em vários pontos, uma vez que a soma resultante é maior que 255, em especial no meio da imagem, que é povoado por tons de cinza medianos(por volta do 128). Quanto aos filtros Box, as imagens resultantes da aplicação de filtros  $1 \times 11$  e  $11 \times 1$  em cascata gera resultados indistinguíveis à aplicação do filtro box  $11 \times 11$ . Juntando esta informação ao fato de que a aplicação dos filtros menores em cascata realiza menos operações para cada canal de cada pixel(22 para os filtros em cascata e 121 para o filtro maior), é possível notar que a aplicação de máscaras menores em cascata diminui o montante de recursos gastos para realizar uma operação com resultados indistinguíveis em comparação com a opção que exige maior poder de processamento, o que é comprovado pelo fato de que em média, a aplicação dos filtros box  $1 \times 11$  e  $11 \times 1$  em cascata leva em média 2.1s, enquanto a aplicação do box  $11 \times 11$  leva em média 9.5s para ser executada, como é possível observar nos prints gerados pelo programa ao executá-lo.

Já quanto a aplicação do filtro  $\|Sobel\|$ , com offset de 128, é possível observar tanto a ação do filtro em si, quanto a da expansão por histograma. O filtro de  $\|Sobel\|$  consiste em detectar bordas a partir do cálculo do gradiente de  $\|Sobel\|$  na vizinhança do pivô na imagem. Como o modelo de correlação utilizado é o que considera os valores absolutos da correlação como o valor assumido por cada canal de cada pixel, é possível observar as bordas detectadas. Como a máscara utilizada nesse exemplo é a vertical, as bordas detectadas são as bordas verticais. Ademais, sem a expansão por histograma, é possível verificar um nível de cinza resultante da adição do offset. Esse nível de cinza é filtrado pela expansão por histograma assim como o esperado, uma vez que esta expande os níveis de cinza novamente para o intervalo total de 0 até 255.

Por fim, o filtro Emboss para detecção de bordas detecta tanto as bordas horizontais quanto as verticais, mesmo não mantendo a matriz das cores, uma vez que o modelo de correlação no qual a máscara foi aplicada é a que trata valores negativos como seu valor absoluto. Além disso, é possível notar o offset de 128 aplicado, característico do filtro, uma vez que a imagem fora das bordas não é um preto puro, mas sim um tom de cinza.

Por fim, as imagens a seguir ilustram aplicações dos 4 filtros de mediana implementados, mediana em R, mediana em G, mediana em B e mediana em RGB, sendo a máscara aplicada a cada um distinta das outras por conta de suas dimensões. A imagem 13 é o resultado da aplicação do filtro da mediana em R com máscara  $5 \times 7$ , já a imagem 14 é o resultado da aplicação do filtro da mediana em G com máscara  $11 \times 7$ , a imagem 13 por sua vez, é o resultado da aplicação do filtro da mediana em B com máscara  $25 \times 1$  e a imagem 16 é o resultado da aplicação do filtro da mediana em RGB com máscara  $1 \times 25$ .

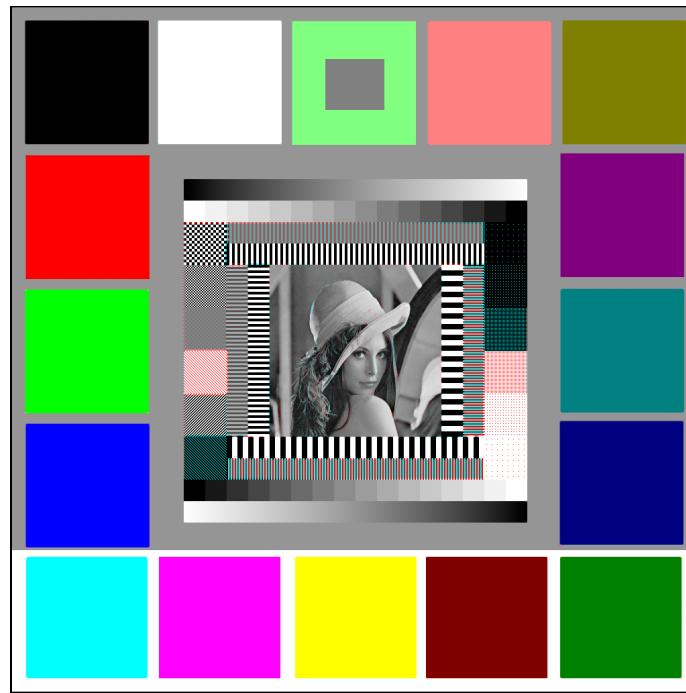


Figura 13. Mediana 5x7 em R

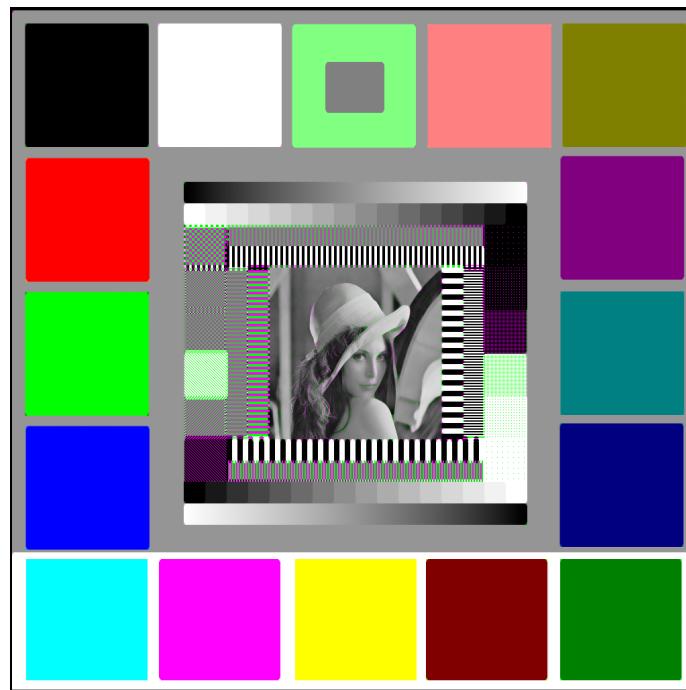


Figura 14. Mediana 11x7 em G

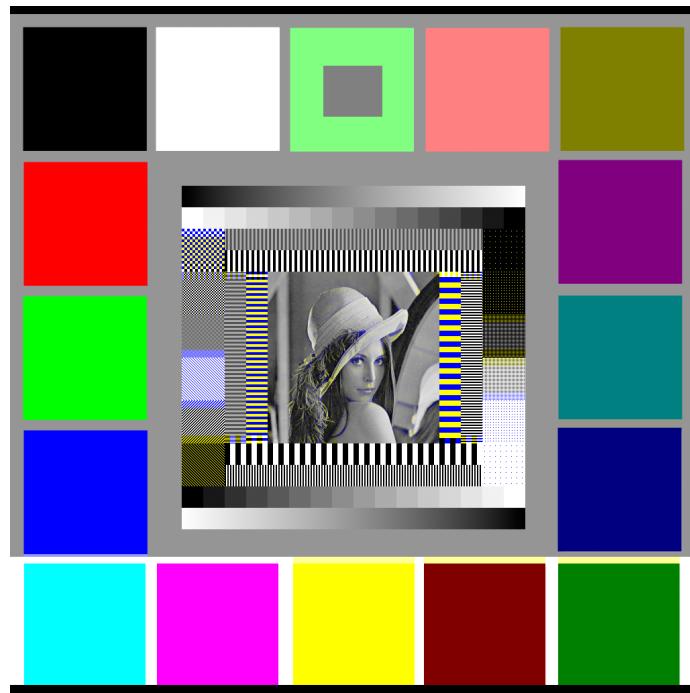


Figura 15. Mediana 25x1 em B

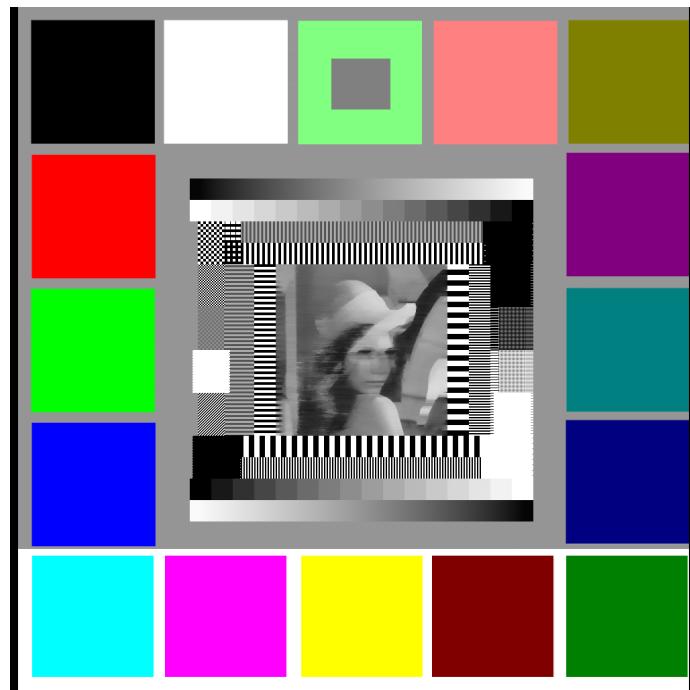


Figura 16. Mediana 1x25 em RGB

Note que nos filtros de apenas um canal, a matriz se conversa na maioria dos pontos, sofrendo alterações apenas nos níveis de cinza no centro da imagem que simulam diversos tipos de ruídos, onde esses níveis de cinza resultam em combinações da matriz da cor pura do canal aplicado e sua cor complementar, ou seja, vermelho e ciano, verde e magenta ou azul e amarelo. Já na aplicação do filtro da mediana em RGB, ao invés de observar uma degradação na matriz das cores, é possível observar um desfoco na imagem, especialmente no meio, onde se encontram os ruídos e a foto da mulher em níveis de cinza.

#### IV. CONCLUSÃO

Em conclusão, os objetivos de consolidar os conhecimentos e as habilidades desenvolvidas durante as aulas da disciplinas foram consolidados por meio do desenvolvimento do trabalho prático. Os resultados obtidos foram consistentes com os esperados em todos os testes, o que indica que as implementações estão funcionando como deveriam e que o trabalho como todo foi um sucesso.