Projeto I de Introdução ao Processamento de Imagens

Maria Claudia Campos Martins
170109968@aluno.unb.br
Universidade de Brasília

***Resumo*—Este trabalho busca estudar processamento de sinais digitais, mais especificadamente de imagens, manipulando imagens no formato YUV com amostragem de 4:2:0 e comparando a imagem resultante ao alterar a amostragem para 4:4:4. Também são realizadas duas formas de redimensionar a imagem YUV. Para a segunda parte do estudo, são aplicados filtros espaciais, mais especificadamente, filtro gaussiano e filtro laplaciano, notando que ao unir o uso dos dois, com parâmetros adequados, o resultado mostra-se ser, subjetivamente, melhor. Por fim, para a terceira parte, aplica-se um filtro no domínio da frequência para retirar o padrão moiré de uma imagem. O filtro é um rejeita-notch com Butterworth passa-altas e é possível visualizar sua ação em determinadas regiões da magnitude da frequência, mostrando as regiões onde o padrão moiré está no domínio da frequência.**

I. Introdução

Processamento de imagens é uma subdisciplina de Processamento Digital de Sinais, em que se tem como objetivo manipular ou melhorar um sinal digital por meio de manipulações matemáticas [2]. Diante disso, este trabalho tem como objetivo a familiarização com o processamento de imagens, propondo desafios para lidar com a manipulação do tamanho da imagem vindo de um arquivo binário no formato *.yuv* além de propor a análise de diferentes resultados quando aplicados alguns dos filtros desenvolvidos no domínio espacial e no domínio da frequência, mais especificadamente, filtro laplaciano, filtro gaussiano e filtro passa-altas *Butterworth* como um rejeita *notch*, respectivamente.

Para lidar com a manipulação de imagens de um arquivo *.yuv*, em especifico arquivos yuv no formato 4:2:0, temos que para cada canal Y no arquivo, temos que a metade dos bytes deste canal corresponde a U e a outra metade corresponde a V. Assim, o tamanho total da imagem representada nesse formato é dada pelo tamanho em bytes de Y multiplicado pelo tamanho de U e pelo tamanho de V. [1]

A aplicação de filtros em imagens se dá pela convolução de uma matriz pré-definida, representando o *kernel* do filtro, sua característica básica, pelos *pixels* da imagem. Assim, convoluindo a matriz pela imagem, vamos obter uma imagem diferente a depender da configuração do *kernel*.

Além disso, filtros no domínio espacial podem ser usados para a detecção de bordas, por realçar o gradiente nas imagens, ou para suavizar as bordas, funcionando como um diluidor das bordas na imagem. O filtro laplaciano realça as bordas da imagem, possuindo um kernel igual a

Já o filtro gaussiano suaviza as bordas. Para o filtro gaussiano, definimos o tamanho de seu *kernel* e o desvio padrão, σ^2, que vamos aplicar. [2]

No domínio da frequência, conseguimos representar ruídos e detalhes das imagens. Com os filtros nesse domínio, podemos eliminar esses ruídos ou permitir que somente alguns detalhes apareçam na imagem final. O filtro a ser utilizado neste trabalho será o rejeita *notch* utilizando *Butterworth* como um passa-altas. Esse filtro rejeita uma faixa de frequências escolhidas, criando círculos em pontos específicos que irão excluir as frequências existentes naquela região. [4]

Por fim, este trabalho se organiza da seguinte forma: na seção II temos a metodologia utilizada, em que são apresentados os experimentos realizados além de detalhes sobre a implementação em código dos experimentos, na seção III são apresentados os resultados dos experimentos feitos a partir da metodologia com as imagens produzidas e comentários sobre o que foi obtido. Já na seção IV temos a conclusão, em que é apresentado um breve relato do que foi observado durante o projeto.

II. Metodologia

A metodologia aplicada consiste em propor resultados às questões levantadas pelas especificações dadas no roteiro para este projeto [3]. Dessa forma, cada experimento realizado para este trabalho será chamado de questão, seguido da numeração correspondente encontrada na documentação do roteiro.

Vale ainda notar que todos os códigos foram desenvolvidos em *python3.10* utilizando as bibliotecas *opencv*, *numpy* e *matplotlib* em um ambiente virtual criado dentro de uma distribuição Linux.

A. Questão 1

A primeira parte do primeiro experimento consiste em desenvolver um programa que lê um arquivo binário no formato *.yuv* e retorna para o usuário as componentes Y, U e V. O programa deve ser capaz de ler um determinado quadro, com tamanho definido pelo usuário, em uma determinada posição em bytes do arquivo, indicando que o arquivo possui vários quadros com imagens *.yuv*.

Dessa forma, o código do programa implementado recebe o nome do arquivo como uma *string*, a largura e altura da imagem além da posição escolhida como um inteiro. O programa lê o arquivo e salta para a posição escolhida, sendo

que ela na verdade é igual a posição *escolhida x tamanho do quadro*. O tamanho do quadro é dado pela multiplicação dos tamanhos das componentes Y, U e V. Para o formato 4:2:0, esse tamanho é de $3/2 * tamanhodoquadro$. Depois da leitura, o material lido do arquivo é salvo em um vetor, ou *array*. Sabendo que nesse *array* contém os elementos Y, U e V, bastou apenas separar os componentes dos *arrays* a partir do tamanho dos elementos e retorná-los no retorno da função.

Para a segunda parte do experimento, a imagem *.yuv* deve ser redimensionada em uma escala múltipla de 2. Para isso, cada pixel da imagem original foi copiado para uma imagem com *pixels* todos pretos, cujo tamanho é múltiplo de 2. A cópia foi feita de tal forma que para cada pixel da imagem original copiado, tivesse um pixel preto na sua coluna à direita e na linha abaixo. Depois, a imagem resultante é passada em outra função que preenche os *pixels* pretos com os imediatos à sua esquerda ou superior, caso eles não fossem pretos. Para casos onde tanto o imediato esquerdo quanto o superior não fosse preto, o superior é escolhido.

A terceira parte propõe outra forma de preenchimento dos *pixels* pretos. A proposta parte da própria autora e conta com calcular a média aritmética dos *pixels* vizinhos não pretos, que não são necessariamente imediatos. Essa proposta parte do principio que os *pixels* pretos estão sempre em posições fixas da imagem, assim seu preenchimento com outras cores se dá pela média aritmética dos vizinhos provenientes da imagem original, como pode ser visto na imagem 1. Assim, para *pixels* pretos nas colunas, sabemos com certeza que os *pixels* superior e inferior são da imagem original. Para *pixels* pretos nas linhas, os vizinhos na sua diagonal são da imagem original. Já para *pixels* com 2 vizinhos da imagem original, a média é feita pela horizontal. Essa relação de vizinhança pode ser vista também na imagem 1.

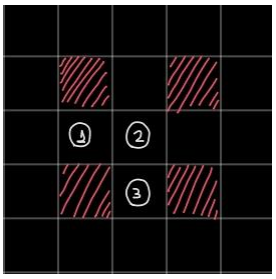

Figura 1: As hachuras indicam *pixels* da imagem original. Os *pixels* pretos indicados pelos números 1,2 e 3 indicam respectivamente, a situação dos vizinhos nas colunas, a situação dos pixels vizinhos nas diagonais e a situação dos vizinhos na horizontal.

Sabendo quais vizinhos são da imagem original, podemos fazer a média dos valores de seus *pixels* e preencher o pixel preto. Vale notar que na borda esquerda da imagem, houve a escolha de preencher os *pixels* pretos com os seus vizinhos na diagonal superior esquerda ou o imediato à esquerda a fim de suavizar uma descontinuidade encontrada durante o desenvolvimento do algoritmo.

A quarta parte busca analisar o resultado ao colocar U e V da imagem da primeira parte do mesmo tamanho que Y, tornando o formato YUV para 4:4:4. Coletou-se a imagem resultante de cada componente e a imagem resultante da interpolação de Y, U e V. A interpolação foi feita utilizando o método *stack* do *numpy*. Vale notar que os resultados foram coletados tanto para a função desenvolvida na segunda parte quanto para a desenvolvida na terceira. Estes resultados exigem a conversão das cores em YUV para RGB.

Por fim, para a última parte do experimento, busca-se analisar a interpolação e as imagens individuais quando Y, U e V são dobrados sem alterar o formato 4:2:0. Para realizar a interpolação, foi feito o processo de *upsampling* de U e V para que ficassem do mesmo tamanho que Y. Esse processo repete x e y de U e V até ficar no mesmo tamanho de Y, não alterando os dados de U e V. Depois de colocar no mesmo formato, a interpolação é feita do mesmo modo que na quarta parte. Assim como na quarta parte, os resultados foram coletados tanto para a função desenvolvida na segunda parte quanto para a desenvolvida na terceira.

Os resultados obtidos em cada parte do experimento podem ser encontrados na seção III. Nessa seção é possível visualizar as imagens resultantes da primeira parte, passando como argumentos para a função os seguintes parâmetros: **foreman.yuv 352 288 15**

Para o restante do experimento, foi utilizada a imagem resultante da primeira parte.

B. Questão 2

O segundo experimento busca avaliar diferentes resultados para diferentes parâmetros passados para o filtro Laplaciano, além de verificar se esses resultados ficam melhores ao processar a imagem em um filtro Gaussiano antes de aplicar o filtro Laplaciano.

Para a primeira parte, aplicou-se um filtro Laplaciano de tamanho 3x3, em todas as direções com centro em $+/-8$. Na segunda parte, primeiro foi aplicado um filtro gaussiano de tamanho 3x3, com $\sigma^2 = 0,5$ para então aplicar o filtro Laplaciano de tamanho 3x3, em todas as direções, com centro em $+/-4$. Já na última parte, foi aplicado um filtro gaussiano de tamanho 3x3,com $\sigma^2 = 1$ para então aplicar o filtro Laplaciano de tamanho 3x3, em todas as direções, com centro em $+/-4$. Os resultados foram obtidos para os dois casos de filtro Laplaciano para cada parte do experimento.

Para aplicar o filtro, foi criada uma função que recebe como parâmetros o nome da imagem e o tipo de filtro laplaciano a ser aplicado. O nome do arquivo que foi utilizado foi "Image1.pgm"e o tipo de filtro varia dentro dos seguintes valores, a depender da parte do experimento a ser realizada, são eles: $-8, 8, -4, 4$.

C. Questão 3

O terceiro experimento tem como objetivo avaliar o resultado da aplicação de um filtro rejeita *notch* utilizando *Butterworth* com n=4 em uma imagem com padrão móire.

Figura 2: Imagem em RGB lida a partir do arquivo *.yuv*

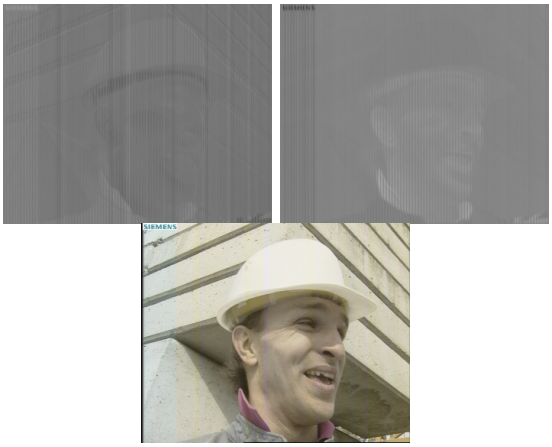

Figura 3: U e V redimensionados e preenchidos com *pixels* imediatos superior ou à esquerda.

Para isso, implementou-se um programa que recebe o caminho da imagem a ser processada, nesse caso foi "moire.tif", transforma a imagem para o domínio da frequência utilizando métodos do *numpy*, calcula o filtro de *Butterworth* considerando as seguintes fórmulas

$$d_1 = \sqrt{(u - \frac{P}{2} + u_k)^2 + (v - \frac{Q}{2} + v_k)^2} \tag{1}$$

$$d_2 = \sqrt{(u - \frac{P}{2} - u_k)^2 + (v - \frac{Q}{2} - v_k)^2} \tag{2}$$

onde (u, v) são as coordenadas da imagem no domínio da frequência, (u_k, v_k) é o centro e (P, Q) é o tamanho da imagem. Assim, se o diâmetro D_0 escolhido for menor ou igual a d_1 ou d_2, o filtro tem magnitude igual a 0. Caso contrário, o filtro tem magnitude igual a 1.

III. Resultados

Essa seção contém os resultados gráficos dos processamentos realizados em código de acordo com a metodologia explicada na seção II. Para a questão 1 e 2, os resultados foram obtidos com o método *imshow()* do *openCV*, já para a questão 3, utilizou-se os métodos do *pyplot* contido na biblioteca *matplotlib*.

A. Questão 1

A imagem YUV lida com os parâmetros escolhidos pode ser vista na figura 2. Os canais Y, U e V podem ser vistos, respectivamente, à direita, na figura inferior esquerda e na figura inferior direita na mesma figura 2.

A seguir, na figura 3 temos o resultado de redimensionar U e V, respectivamente, da esquerda para direita, utilizando o preenchimento com os *pixels* imediatos à esquerda ou superiores. Na última imagem, temos a imagem resultante da interpolação com o Y da figura 2 e U e V redimensionados.

Já na figura 3, temos o redimensionamento de U e V, respectivamente, da esquerda para direita, com o preenchimento a partir da média dos *pixels* da imagem original, além do resultado da interpolação com o Y da figura 2. Já na figura 4, a organização das imagens se dá de forma semelhante a figura 3, mas nesse caso, o preenchimento foi feito utilizando a média dos *pixels* vizinhos.

Figura 4: U e V redimensionados e preenchidos com *pixels* a partir da média dos vizinhos.

Na figura 5 temos Y redimensionado utilizando o preenchimento por imediato superior ou à esquerda na linha superior e por média, na linha inferior. À direita, temos a imagem YUV resultante da interpolação do novo Y com U e V respectivos de cada método.

Dos resultados, nota-se que aumentar o tamanho de U e V utilizando o redimensionamento com preenchimento de imediatos à esquerda ou superiores gera imagem RGB com cores menos vibrantes que a imagem resultante do preenchimento com a média. É possível ver que o canal U possui mais ruídos, com linhas muito escuras ou até mesmo pretas na imagem 3, o que não é visto na imagem 4. Quando Y também é redimensionado, vemos sua grande influência na percepção da imagem pois o resultado do preenchimento dos *pixels* com imediatos possui os mesmos riscos verticais que existem em Y, como pode ser visto na imagem 5. Como o método pela

Figura 5: Canal Y redimensionado para o dobro da escala original e imagem YUV resultante de cada método de interpolação de YUV implementado.

Figura 6: Imagem utilizada para o processamento do experimento 2.

média, Y não possui os ruídos, a imagem resultante também não possui.

B. Questão 2

Os resultados obtidos para a questão 2 podem ser vistos na figura 7, em que podemos visualizar o resultado do filtro laplaciano com *kernel* e centro em 8 e -8, respectivamente; na imagem 8, vemos o resultado de aplicar o filtro gaussiano com *kernel* de tamanho $3x3$ e desvio padrão igual a $0,5$, para depois aplicar o filtro laplaciano com centro em 4 e -4, e na imagem 9, temos o resultado do filtro gaussiano com *kernel* de tamanho $3x3$ e desvio padrão igual a 1, para aplicar o mesmo filtro laplaciano da imagem 8. Na "Fig. 6" temos a imagem original a ser processada pelos filtros.

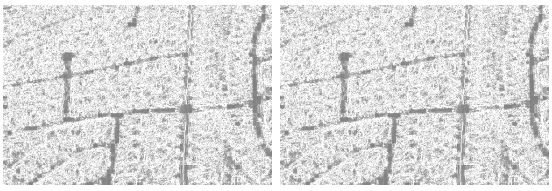

Figura 7: Filtro laplaciano com centro do *kernel* em 8 à esquerda e -8 à direita.

Das imagens obtidas, quando o filtro gaussiano é aplicado para depois aplicar o filtro laplaciano é possível ver com mais nitidez as bordas das cadas e das ruas observadas na figura original em 6. É possível ver que na imagem com apenas o

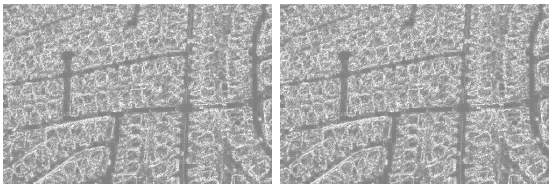

Figura 8: Filtro gaussiano aplicado com desvio padrão igual a 0.5. À esquerda temos o resultado do laplaciano com centro em 4, à direita, com centro em -4.

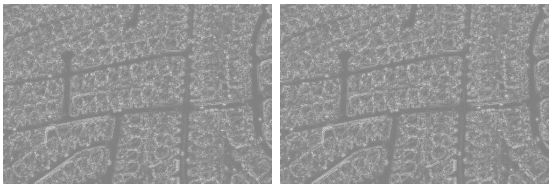

Figura 9: Filtro gaussiano aplicado com desvio padrão igual a 1. À esquerda temos o resultado do laplaciano com centro em 4, à direita, com centro em -4.

filtro laplaciano aplicado, Fig. 7, apesar de mudar o centro, é possível ver apenas um pouco das ruas. Quando o filtro gaussiano com $\sigma^2 = 0,5$ é aplicado com o laplaciano, é possível distinguir as ruas e as casas. Quando $\sigma^2 = 1$ já não é possível distinguir as casas mas é possível visualizar melhor as ruas. Independente do centro do filtro laplaciano, não se vê diferença nas imagens com gaussiano aplicado de mesmo desvio padrão.

C. Questão 3

O resultado obtido para a aplicação do filtro rejeita *notch* utilizando filtro passa-altas *Butterworth* pode ser visto na imagem 10.

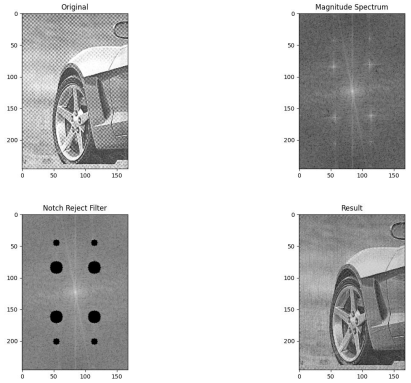

Figura 10: Imagem no domínio espacial e da frequência da esquerda superior para direita. Abaixo à esquerda temos o filtro rejeita *Notch* aplicado no domínio da frequência e à direita temos seu resultado plotado no domínio espacial.

Na figura 10 é possível visualizar que a imagem original possui o padrão moiré [CITE]. Ao aplicar o filtro *Butterworth* nas regiões definidas pela metodologia, vemos que esse filtro

rejeita as frequências dessas regiões, removendo o padrão moiré da imagem dando um aspecto mais definido das formas na imagem.

IV. Conclusões

No processamento de imagens a partir de arquivos binários no formato *yuv*, primeiro, precisamos tratar o sinal e deixá-lo adequado para o processamento de fato, mostrando a importância do tratamento adequado antes do processamento. Caso o tratamento tivesse algum erro, o resultado não era satisfatório e muitas vezes era confuso. O cuidado com a amostragem do sinal armazenado no *yuv* foi fundamental para que a imagem resultante fosse legível de fato. O redimensionamento é um processo relativamente simples, mas a interpolação dos valores dos *pixels* vazios do processo influencia muito na percepção da imagem ou na presença de até mesmo ruídos, que podem atrapalhar a visualização da imagem. Apesar da mudança nos parâmetros de amostragem, foi percebido que não houve diferença subjetiva em relação a nitidez e percepção da imagem.

Na aplicação de filtros espaciais, notou-se que muitas vezes faz-se necessário aplicar diferentes filtros com diferentes propósitos para obter o resultado que se deseja. Julgando que o objetivo seja observar as bordas das casas e das ruas, o melhor resultado foi a imagem em que se utilizou o filtro gaussiano com desvio padrão igual a 0,5, para depois aplicar o filtro laplaciano.

Por fim, o filtro rejeita *notch* se mostrou ideal para minimizar ou deletar o ruído no padrão moire, deixando a imagem mais nítida. Apesar disso, notou-se que a imagem resultante pareceu mais escura, sugerindo que alguns dos parâmetros do filtro podem estar muito grandes, como o D_0 dos círculos mais próximos da origem.

Referências

[1] Wikipedia, “Chroma Subsampling”.
[2] Bruno Macchiavello, “Slides da disciplina”, Acessado em 16/10/2023.
[3] Bruno Macchiavello, “Projeto 1”, Acessado em 16/10/2023.
[4] Rafael C. Gonzalez Richard E. Woods, ‘Digital Image Processing‘”