Projeto 01: Ampliando e Reduzindo Imagens por Replicação de Pixels

Nome do Curso: Engenharia de Computação
Nome do Aluno: Antonio Pedro De Moura Laureno
Data de Entrega: 03/10/2025

Apresentação

Este relatório apresenta a implementação de um programa em Python capaz de ampliar e reduzir imagens digitais utilizando o algoritmo de interpolação por replicação de pixels, também conhecido como vizinho mais próximo. O trabalho segue as especificações do Projeto 01, realizando testes práticos e analisando as mudanças associadas a esses testes.

1 Discussão Técnica

A técnica central implementada neste projeto para a ampliação e redução de imagens é a **Interpolação pelo Vizinho Mais Próximo (Nearest-Neighbor Interpolation)**. Este é o método de interpolação mais fundamental e computacionalmente mais simples, sendo frequentemente descrito de forma intuitiva como Replicação de Pixels, especialmente em casos de ampliação.

Para começar, a imagem utilizada como exemplo foi a da lua de jupiter IO e ela foi carregada usando a biblioteca **OpenCV**. Para a implementação do algoritmo, foi criada uma função **Redimensionar(Imagem, Fator)**, que recebe como parâmetros a **Imagem** carregada do opencv, e um **Fator** que será utilizado para definir o quanto a imagem vai reduzir ou ampliar. Ademais, para começar, calculamos as novas dimensões da imagem, multiplicando as dimensões originais da imagem pelo **fator**. Com essas dimensões novas, criamos uma nova matriz (Imagem de destino) preenchida com zeros. Após isso, realiza-se o **mapeamento**, no qual, a coordenada do novo pixel é projetada de forma inversa sobre a grade da imagem original, com base na proporção entre os tamanhos, resultando em uma localização com valores fracionários. Depois, ocorre o **preenchimento** onde essa localização fracionária é arredondada para as coordenadas do inteiro mais próximo usando a função **round()**, identificando o "vizinho mais próximo". O valor de cor deste vizinho é então diretamente copiado (ou replicado) para o pixel da nova imagem, definindo sua cor final sem qualquer cálculo de média ou mistura com outros pixels. Outrossim, foi criado duas outras funções para executar o que foi pedido no projeto de forma eficiente: **ampliar_imagem()** e **reduzir_imagem()**, que usam como base a função **Redimensionar()**.

a lógica implementada no algoritmo se baseia na relação matemática do vizinho mais próximo definida por:

$$I'(x', y') = I\left(\left\lfloor x' \cdot \frac{1}{f} + 0.5 \right\rfloor, \left\lfloor y' \cdot \frac{1}{f} + 0.5 \right\rfloor\right) \tag{1}$$

- $I'(x', y')$: valor de um pixel na nova imagem redimensionada.
- f: fator de escala.
- função de arredondamento (vizinho mais próximo) no lado direito
- (x', y'): coordenadas na nova imagem.

Tal relação matemática usa como base as coordenadas originais de um pixel na imagem original, dada por $I(x, y)$.

2 Discussão dos Resultados e Conclusões

2.1 Resultados

Após implementar as funções de ampliar e reduzir imagens por replicação de pixels, o próximo passo foi resolver e implementar as tarefas que o projeto solicitou. Diante disso, a primeira tarefa foi aplicado uma redução na imagem original por um fator de 10 utilizando a função **reduzir_imagem()**. Uma vez que a imagem foi reduzida por um fator de 10, a segunda tarefa do projeto pede para reampliar a imagem reduzida pelo mesmo fator. Diante disso, Agora utilizando a função **ampliar_imagem()**, a tarefa foi feita. Seguem as imagens obtidas após essa sequência de processos:

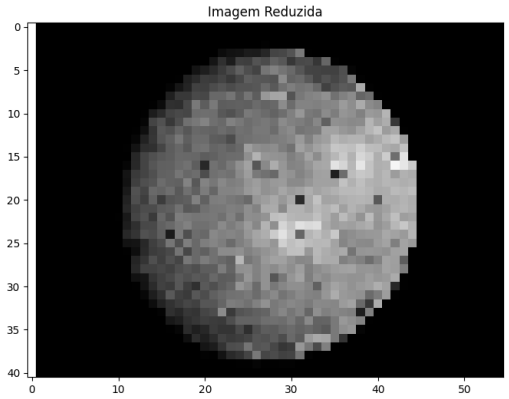

Figura 1: Imagem reduzida por um fator de 10

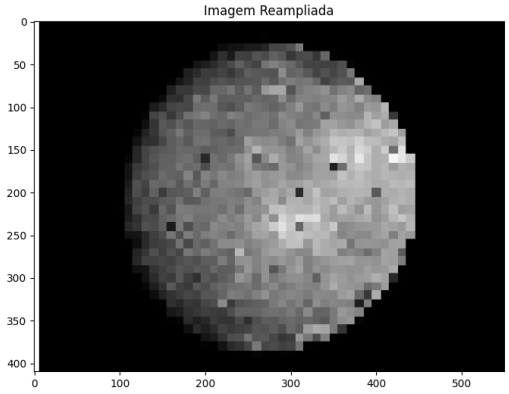

Figura 2: Imagem reampliada por um fator de 10

2.2 Conclusões

Para explicitar o aumento e a redução da imagem foi utilizado a função **.shape** para pegar o tamanho das imagens após a redução e a reampliação. A titulo de ilustração, após usar o **.shape** na imagem original, foi identificado que o tamanho dela é (416, 550). Quando foi aplicado o mesmo método para as figuras 1 e 2, encontramos os respectivos tamanhos: (41, 55), (410, 550). Com isso, a análise das dimensões feitas confirmou que a **redução por um fator de 10** diminui a altura e a largura para 1/10 das originais, resultando em uma imagem com apenas 1% do total de pixels da imagem fonte. Essa compressão drástica acarreta uma perda massiva de informação. Detalhes finos, texturas e contornos suaves presentes na imagem original são inevitavelmente descartados e substituídos por uma representação muito mais simples e quadriculada. Ademais, a conclusão mais importante do projeto é demonstrada ao **reampliar** a imagem reduzida de volta à sua dimensão original. Embora a imagem reampliada possua quase as mesmas dimensões da imagem original, sua qualidade visual é drasticamente inferior. O algoritmo de replicação de pixels, como o nome sugere, não inventa ou recupera os 99% de dados que foram perdidos. Em vez disso, ele simplesmente replica a informação limitada que restou. Logo, podemos tomar como conclusões que a redução da imagem por replicação de pixels causa perda de informação irreversível e que, ao tentarmos reampliar para o tamanho original, o algotitmo não consegue recriar os detalhes da imagem original. Por fim, segue em anexo a comparação da imagem original, reduzida e reampliada:

Figura 3: Comparação das 3 imagens.