Trabalho 5

MO443/MC920 - Introdução ao Processamento de Imagem Digital

Lucas de Oliveira Silva - 220715

1 Introdução

O objetivo deste trabalho foi executar e comparar diferentes métodos de interpolação aplicados a transformações geométricas de escala e rotação. A solução foi feita com Python 3.8.5 em um único arquivo .py. Foram utilizadas as seguintes bibliotecas: numpy (1.19.1) e scikit-image (0.16.2).

2 Como executar

A implementação está no *script* trabalho5.py, as imagens e arquivos de teste na raiz e as saídas obtidas na pasta output. O programa precisa necessariamente de uma imagem de entrada (-i img) e uma de saída (-i img), ambos em formato png. O resto dos parâmetros são opcionais e são definidos como aqueles do enunciado. O método de interpolação utilizado deve ser informado por meio de um inteiro entre 0 e 3, cuja correspondência segue a ordem do enunciado.

É necessário que ao menos um parâmetro dentre -a, -e e -m seja definido, e caso não seja o programa é abortado e uma mensagem de erro é exibida.

Ex: python3 trabalho5.py -i baboon.png -o out.png -d 1024 256 -m 0

Para mais detalhes é possível executar o *script* com o parâmetro -h.

A imagem é lida usando a função *io.imread* do skimage em escala de cinzas, e logo depois os valores são convertidos para uint8 usando a função *img_as_ubyte* . O processamento feito não altera o tipo dos pixeis da imagem de saída, portanto o resultado é salvo com uma simples chamada ao método *io.imsave* do skimage.

Ao realizar o mapeamento, pontos da imagem de saída podem corresponder a pontos fora da imagem de entrada e portanto, nesses casos, um pixel padrão preto é colocado no lugar (resultado perceptível nas rotações da figura 1).

3 Avaliação dos resultados

As transformações são feitas de forma indireta para eliminar a possibilidade de que pixeis no meio da imagem de saída possuam valores indefinidos. Para poder rotacionar a imagem a partir de seu centro foi utilizado coordenadas homogêneas para representar transformações de translação, ou seja, a matriz de rotação usada consiste em uma translação seguida de uma rotação e uma translação inversa.

Os métodos de interpolação de vizinho mais próximo e bilinear tiveram tempos de execução parecidos, mas o resultado foi pouco satisfatório nos dois tipos de transformação. Na rotação retas e curvas perderam a continuidade (figura 1 a/b) e ao mudar a escala artefatos ficaram bastante evidentes (figura 4 a/b). Já que a diference de tempo foi miníma o método bilinear é mais recomendável, pois ainda sim teve um resultado levemente superior.

Já os outros métodos demoraram ao menos 3 vezes mais para executar, com um aumento ainda maior nas transformações em que a imagem de saída possui mais pixeis que a de entrada. Mas, por outro lado, o resultado foi bastante superior tanto em imagens com pouca textura e linhas marcadas (figura 3 c/d) quanto nas com texturas (figura 4 c/d).

O método de polinômios de Lagrange produziu artefatos proeminentes e indesejáveis ao ser aplicado em rotações de imagem com linhas marcadas (figura 2/3 d), mas em transformações de escala foi o que produziu o melhor resultado (figura 4 d). A Interpolação bicúbica produz imagens suavizadas ao ser aplicada em aumentos de escala, portanto ela é mais recomendada apenas para rotações.

Em resumo, a interpolação bilinear é a melhor opção para aplicações em tempo real, a por polinômios de Lagrange para obter melhores resultados em transformações de escala e a interpolação bicúbica para rotações de imagens com com linhas marcadas.

O mapeamento indireto foi crucial para melhorar a qualidade dos resultados, mas ainda sim as bordas foram ignoradas e ficaram em "branco". Uma interpolação restrita a menos pontos poderia ser utilizada para resolver esse problema.

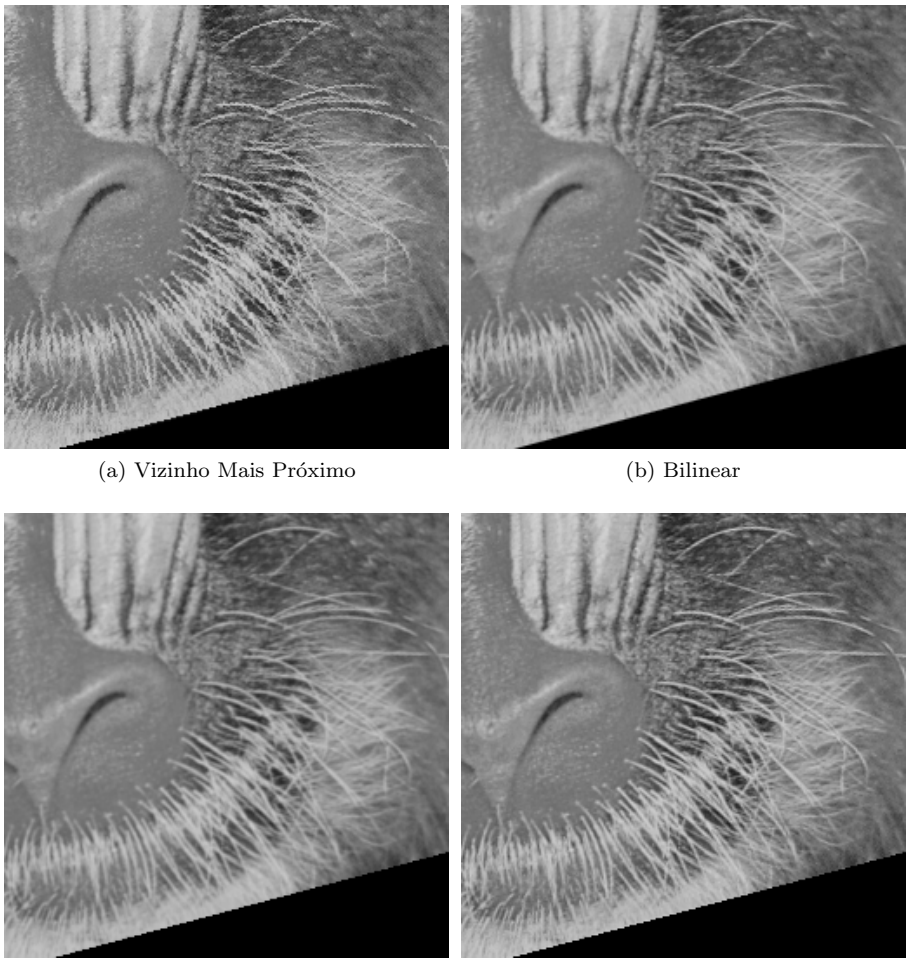
4 Conclusão

Ao testar os vários métodos ficou claro que não há nenhum motivo para utilizar a interpolação por vizinho mais próximo, pois a bilinear produz resultados superiores em praticamente o mesmo tempo.

Já com relação aos métodos mais elaborados há espaço para divergências, apesar do método de Lagrange produzir resultados superiores na maioria dos casos ele gera artefatos em rotações de tipos especiais de imagens. Mas nas transformações de escala o método de Lagrange é superior em todos os aspectos, pois em menos tempo produz um resultado melhor que a interpolação bicúbica.

Por ser implementado usando coordenadas homogêneas o programa pode ser facilmente expandido para adicionar outras transformações geométricas como translações e cisalhamentos, além disso é possível compor várias transformações numa única execução. Também é simples alterar o centro da transformação de rotação, e um parâmetro poderia ser adicionado ao *script*.

A implementação foi bem direta e com pouca preocupação com performance, portanto pequenas melhorias como vetorização poderiam diminuir bastante o tempo de execução. No geral os resultados foram bastante satisfatórios podendo ser utilizados em aplicações reais mais elaboradas.



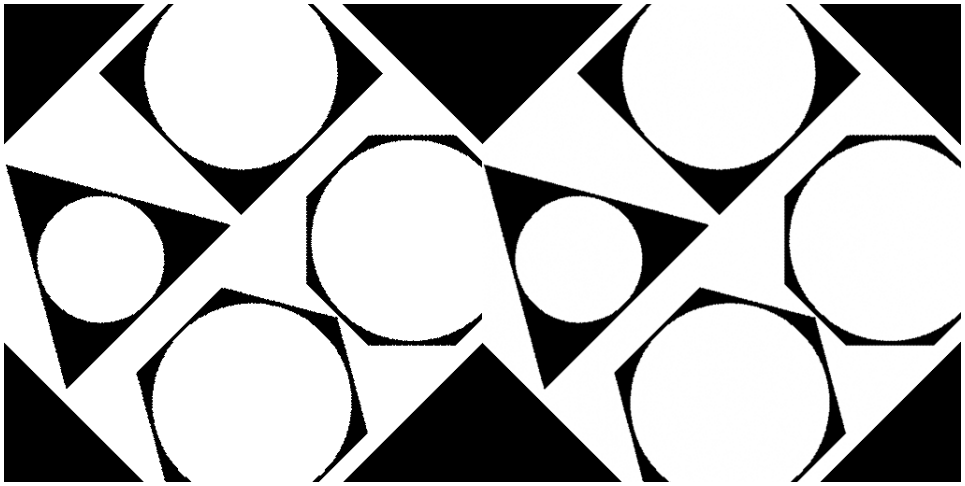
(a) Vizinho Mais Próximo

(b) Bilinear

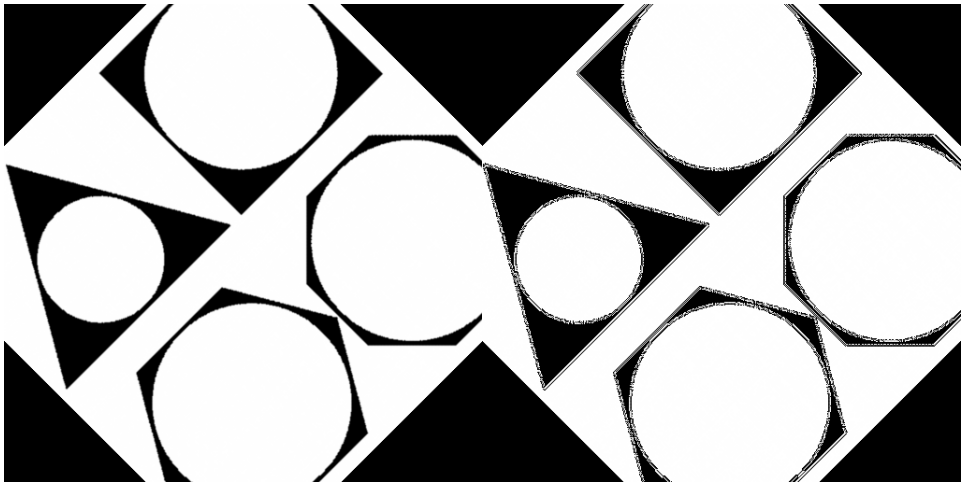
(c) Bicúbica

(d) Polinômios de Lagrange

Figura 1: Zoom na rotação da imagem baboon em $15^{\underline{o}}$



(a) Vizinho Mais Próximo (b) Bilinear



(c) Bicúbica (d) Polinômios de Lagrange

Figura 2: Rotação da imagem poly em 45º

(a) Vizinho Mais Próximo

(b) Bilinear

(c) Bicúbica

(d) Polinômios de Lagrange

Figura 3: Zoom na rotação da imagem watch em 15^{o}

(a) Vizinho Mais Próximo (b) Bilinear

(c) Bicúbica (d) Polinômios de Lagrange

Figura 4: Zoom na ampliação da imagem baboon em 5x