

Relatório - Trabalho 4

1. Introdução

O trabalho proposto consiste em implementar quatro métodos de interpolação (pelo Vizinho Mais Próximo, Bilinear, Bicúbica e por Polinômios de Lagrange) e uma operação de rotação em imagens digitais. O programa codificado deve realizar transformações geométricas de escala e de rotação.

2. Execução, entradas e saídas

Há cinco possíveis fluxos de execução do programa, cada uma representa uma operação em imagem digital:

- A. Rotação
- B. Escala utilizando Interpolação pelo Vizinho Mais Próximo
- C. Escala utilizando Interpolação Bilinear
- D. Escala utilizando Interpolação Bicúbica
- E. Escala utilizando Interpolação por Polinômios de Lagrange

Todas os cinco fluxos de execução possíveis, em resumo, recebem como entrada uma imagem a ser usada para realizar a operação desejada e possuem como saída a imagem resultante após a operação.

Para executar o programa, deve-se utilizar:

```
python t4.py [caminho_imagem_entrada] [caminho_imagem_saída] [operação_desejada]  
[parâmetros da operação]
```

sendo:

- *caminho_imagem_entrada* o caminho para a imagem a ser utilizada como entrada;
- *caminho_imagem_saída* o caminho para o programa salvar a imagem resultante
- *operação_desejada* o número que indica a operação a ser executada
 - 0 - rotação (A)
 - 1 - escala utilizando Interpolação pelo Vizinho Mais Próximo (B)
 - 2 - escala utilizando Interpolação Bilinear (C)
 - 3 - escala utilizando Interpolação Bicúbica (D)
 - 4 - escala utilizando Interpolação por Polinômios de Lagrange (E)
- *parâmetros da operação* os parâmetros específicos da operação
 - para operação de rotação (A):
 - ângulo de rotação em graus (em sentido anti-horário)

- para operações de interpolação (B, C, D e E)
 - fator de escala (sendo 1 para manter as dimensões atuais da imagem, portanto, sem efeito)

O fator de escala pode ser substituído por dois valores: nova dimensão x em pixels e nova dimensão y em pixels.

3. Descrição da solução

Para a implementação do programa foi utilizada a linguagem Python (versão 3.6.5) e as bibliotecas *imageio*, *math*, *matplotlib*, *NumPy* e *sys*. Nas subseções seguintes serão explicadas as soluções adotadas para cada uma das transformações.

Todos os fluxos de execução possuem em comum o carregamento da imagem de entrada e a escrita em disco da imagem resultante. O primeiro utiliza a biblioteca *imageio* e o segundo utiliza a biblioteca *matplotlib*.

3.1. Rotação

Para realizar rotacionamento na imagem, inicialmente o ângulo em graus passado como parâmetro de execução é convertido para radianos utilizando a biblioteca *math*. Dessa forma, essa mesma biblioteca poderá ser usada posteriormente para obter seno e cosseno do ângulo em radianos.

Os cálculos de rotacionamento seguem a seguinte lógica:

- sejam $f'(x',y')$ a imagem resultante, $f(x,y)$ a imagem original e θ o ângulo de rotação
- $x' = x * \cos(\theta) - y * \sin(\theta)$
- $y' = x * \sin(\theta) + y * \cos(\theta)$
- portanto, $f'(x',y') = f(x * \cos(\theta) - y * \sin(\theta), x * \sin(\theta) + y * \cos(\theta))$

3.2. Interpolações

As implementações das interpolações foram feitas basicamente utilizando funções padrão de Python e funções da biblioteca *math*. Além disso, ocorre o emprego da biblioteca *NumPy* para criar um *ndarray* zerado, que é preenchido em cada iteração, resultando ao final na imagem resultante.

4. Testes e resultados obtidos

Foram realizados testes com quatro imagens digitais e os seguintes cenários de teste:

- “*baboon.png*” (vide *Figura 1*): imagem fornecida previamente; aplicando:
 - rotação de 12,75°,
 - fator de escala igual a 0,76;
- “*butterfly.png*” (vide *Figura 2*): imagem fornecida previamente; aplicando:
 - rotação de 42,1°,
 - fator de escala de 2,25;
- “*city.png*” (vide *Figura 3*): imagem fornecida previamente; aplicando:
 - rotação de 25,75°,
 - fator de escala de 2,25;
- “*house.png*” (vide *Figura 4*): imagem fornecida previamente; aplicando:
 - rotação de 17,647°,
 - fator de escala igual a 3;
- “*seagull.png*” (vide *Figura 5*): imagem fornecida previamente; aplicando:
 - rotação de 15,75°,
 - fator de escala igual a 3,25;
- “*DSC00114.png*” (vide *Figura 6*): imagem de autoria do autor deste relatório, convertida para escala de cinza, com dimensões 4608 x 3456 pixels
 - rotação de 25,15°,
 - fator de escala de 0,35,
 - é um caso especial utilizando interpolação pelo vizinho mais próximo para obter uma imagem de 1024 x 1024 pixels.

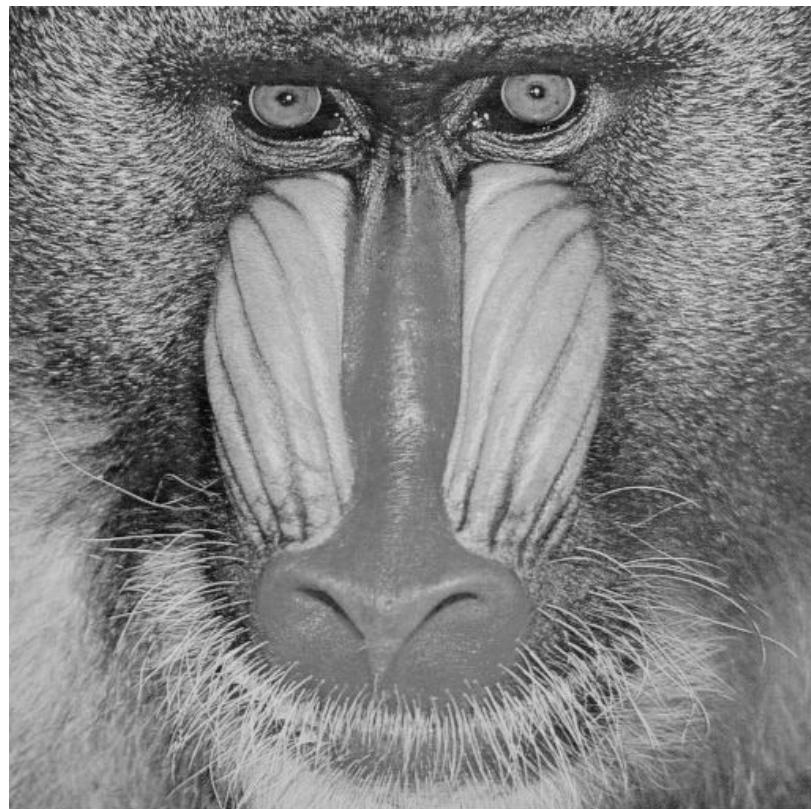


Figura 1 - “baboon.png”



Figura 2 - “butterfly.png”



Figura 3 - “city.png”

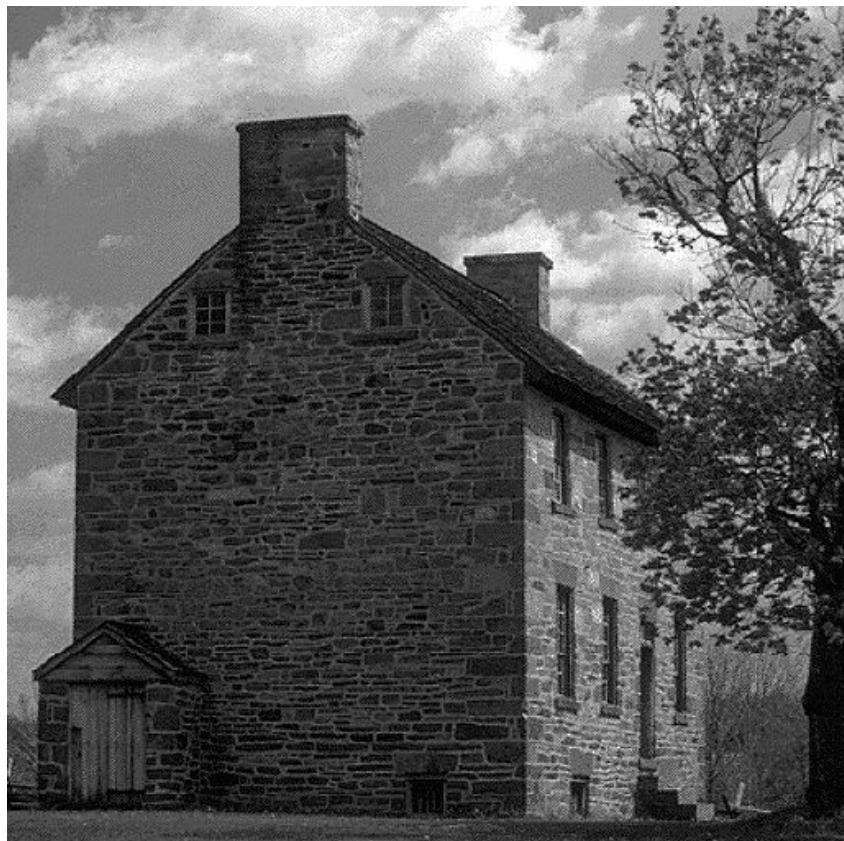


Figura 4 - “house.png”



Figura 5 - “seagull.png”



Figura 6 - "DSC00114.png"

4.1. Imagem “baboon.png”

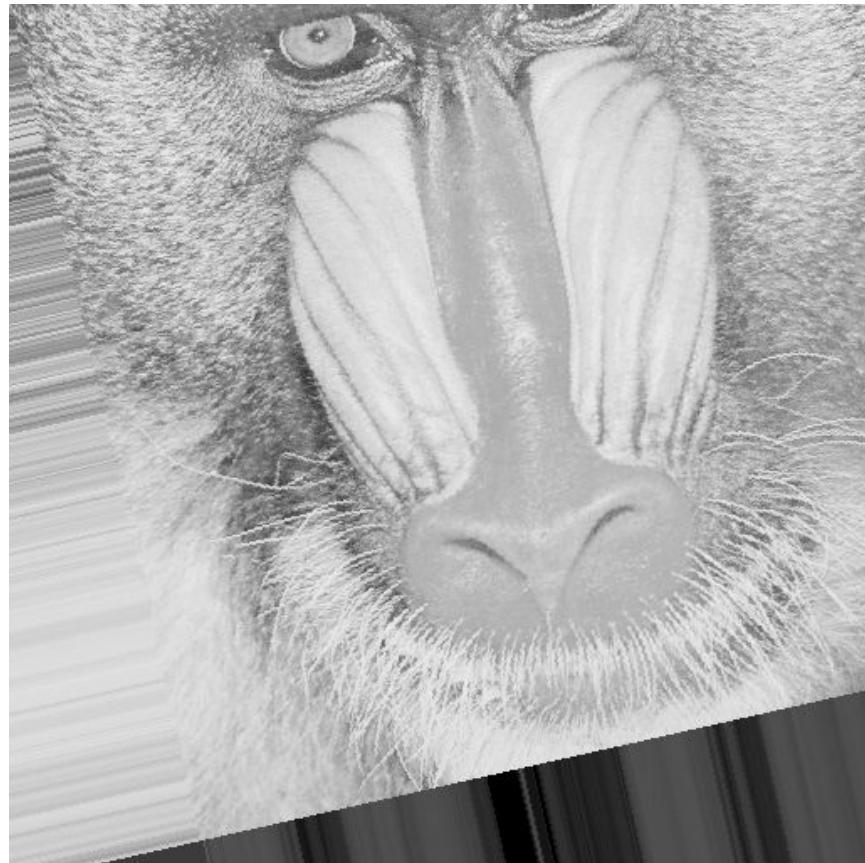


Figura 7 - Imagem “baboon.png” após rotação

A imagem “baboon.png” foi rotacionada em 12,75º, resultando na *Figura 7*.

Como se pode perceber, o rotacionamento ocorre em relação à origem, resultando em um deslocamento que passa das dimensões da imagem original. A implementação utilizada não realiza nenhum tratamento especial (de modo a re-centralizar a imagem, por exemplo) em relação aos pixels deslocados que ficam além das dimensões originais da imagem.

As *figuras 8, 9, 10 e 11* mostram a mesma imagem original após as interpolações pelo Vizinho Mais Próximo, Bilinear, Bicúbica e por Polinômios de Lagrange, respectivamente. Foi aplicado um fator de escala de 0,76.

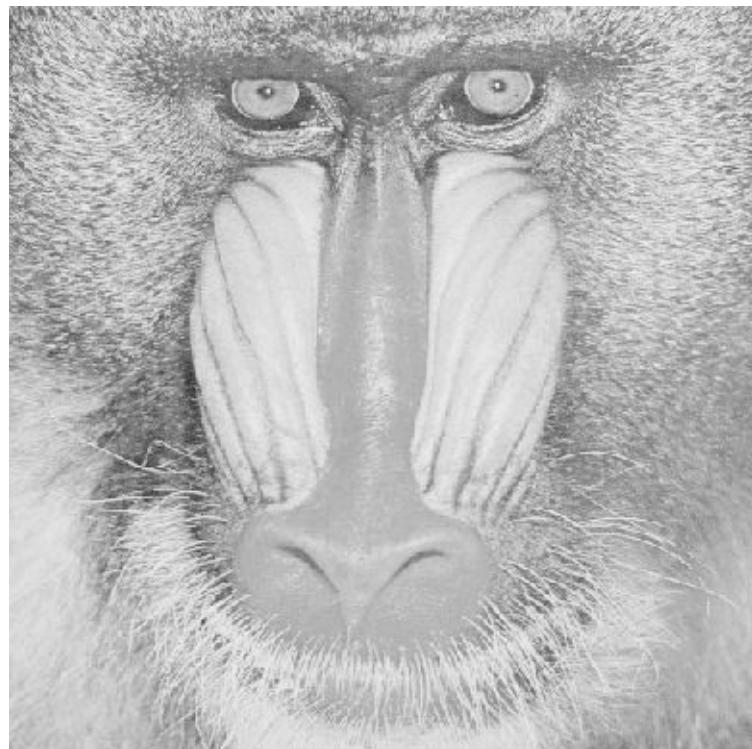


Figura 8 - “baboon.png” após Interpolação pelo Vizinho Mais Próximo

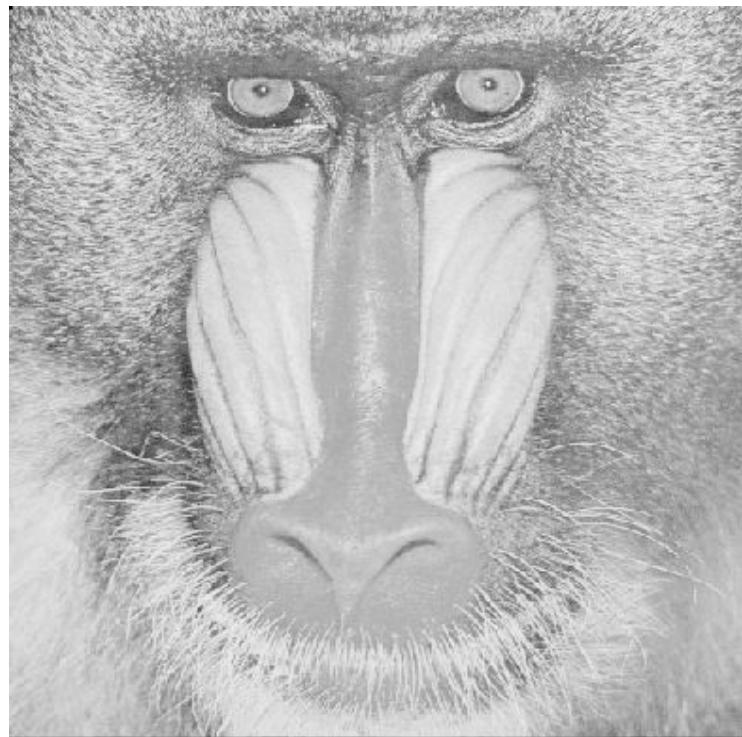


Figura 9 - “baboon.png” após Interpolação Bilinear

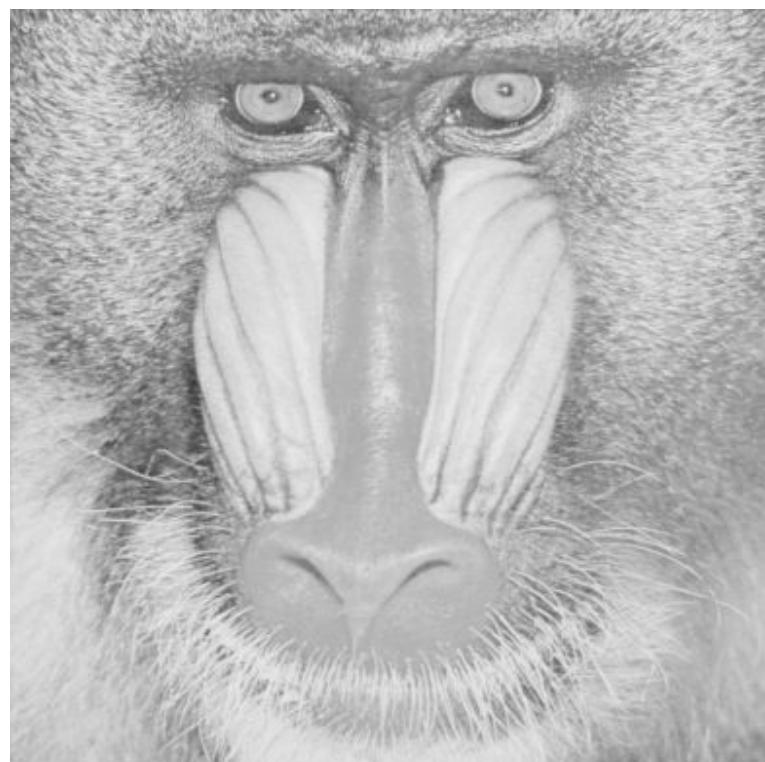


Figura 10 - “baboon.png” após Interpolação Bicúbica

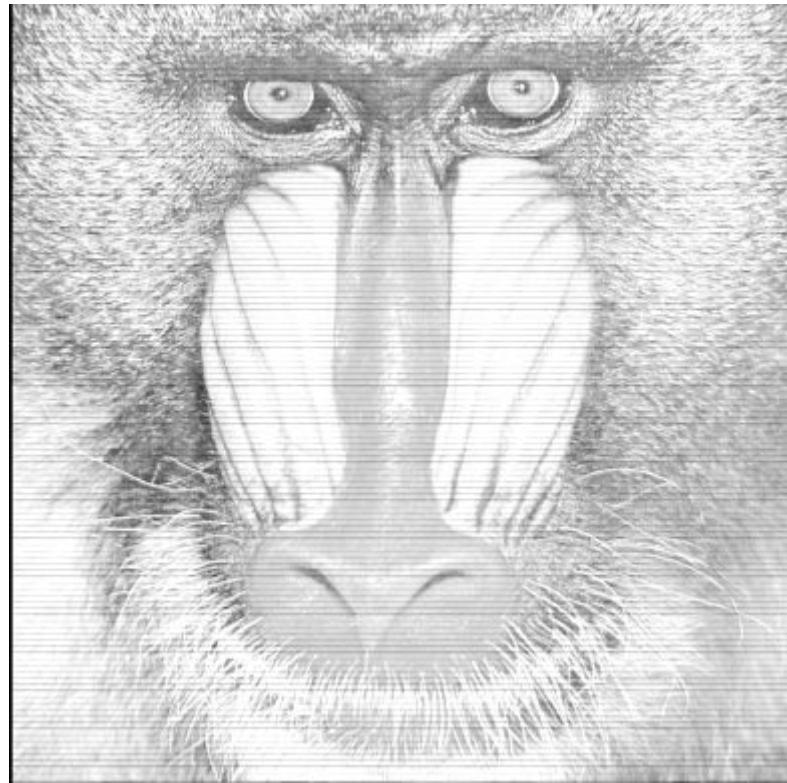


Figura 11 - “baboon.png” após Interpolação por Polinômios de Lagrange

4.2. Imagem “butterfly.png”

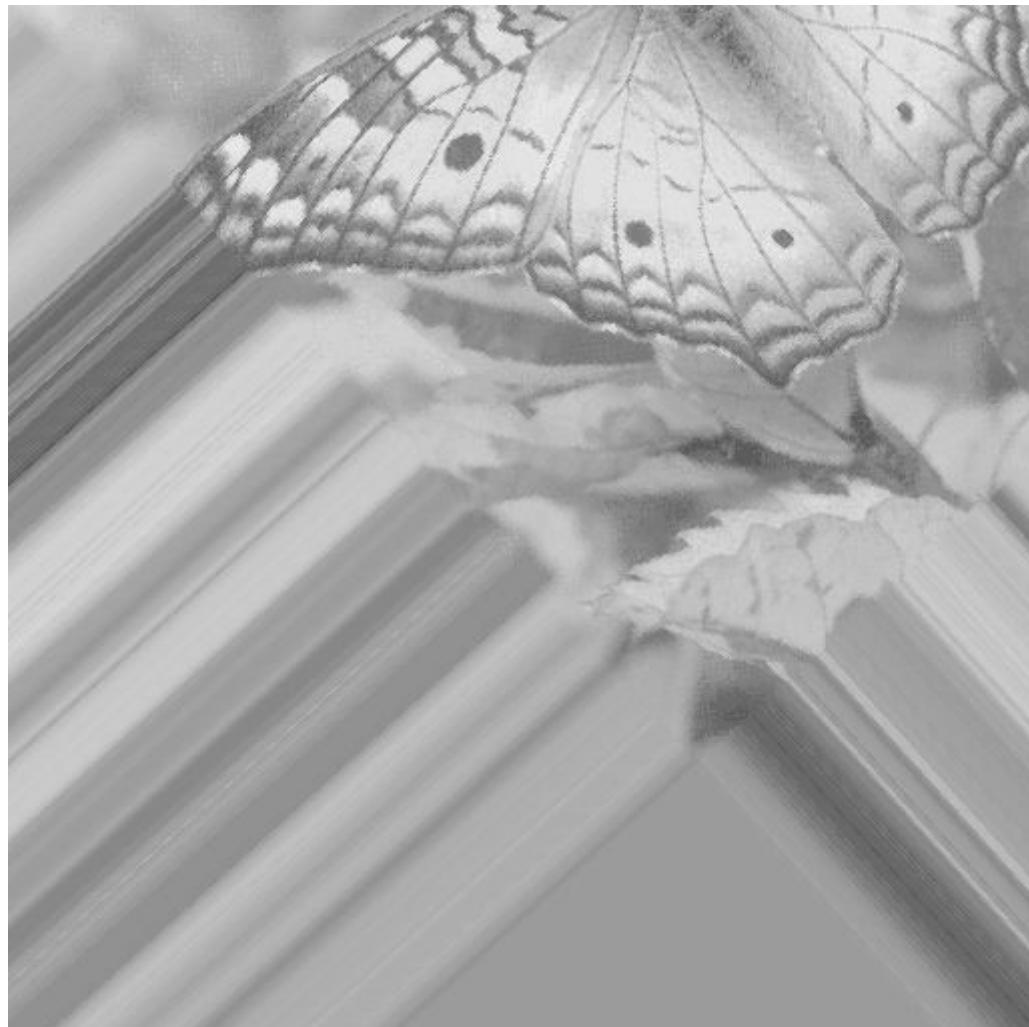


Figura 12 - “butterfly.png” após rotação

A imagem “butterfly.png” foi rotacionada em 42,1°, resultando na *Figura 12*.

As *figuras 13, 14, 15 e 16* mostram a mesma imagem original após as interpolações pelo Vizinho Mais Próximo, Bilinear, Bicúbica e por Polinômios de Lagrange, respectivamente. Foi aplicado um fator de escala de 2,25.

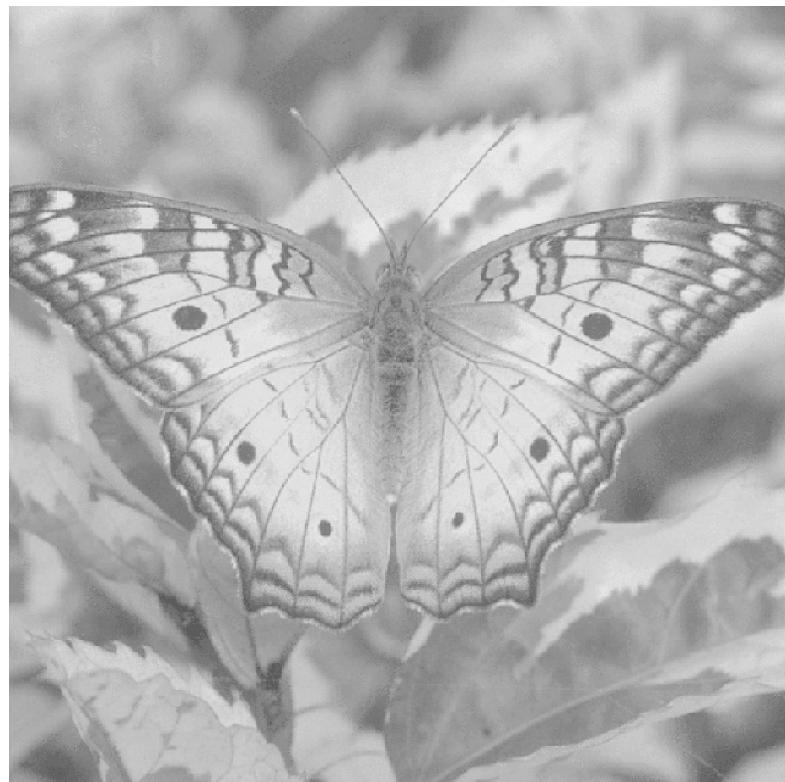


Figura 13 - “butterfly.png” após Interpolação pelo Vizinho Mais Próximo

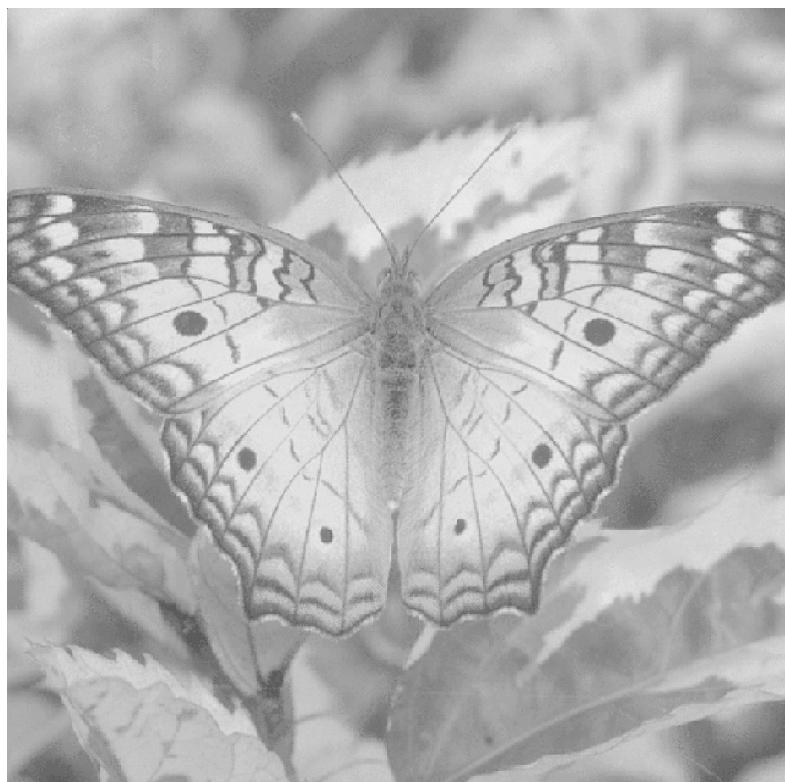


Figura 14 - “butterfly.png” após Interpolação Bilinear



Figura 15 - “butterfly.png” após Interpolação Bicúbica

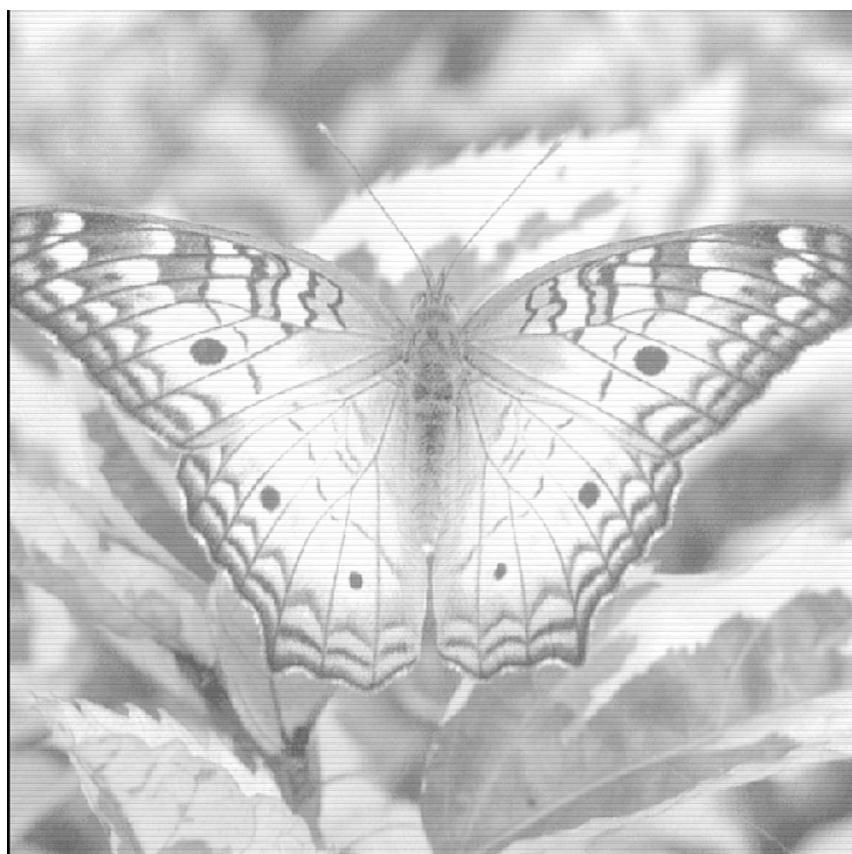


Figura 16 - “butterfly.png” após Interpolação por Polinômios de Lagrange

4.3. Imagem “city.png”



Figura 16 - “city.png” após rotação

A imagem “city.png” foi rotacionada em 25,75°, resultando na *Figura 17*.

As *figuras 18, 19, 20 e 21* mostram a mesma imagem original após as interpolações pelo Vizinho Mais Próximo, Bilinear, Bicúbica e por Polinômios de Lagrange, respectivamente. Foi aplicado um fator de escala de 2,25.



Figura 17 - “city.png” após Interpolação pelo Vizinho Mais Próximo



Figura 18 - “city.png” após Interpolação Bilinear



Figura 19 - “city.png” após Interpolação Bicúbica

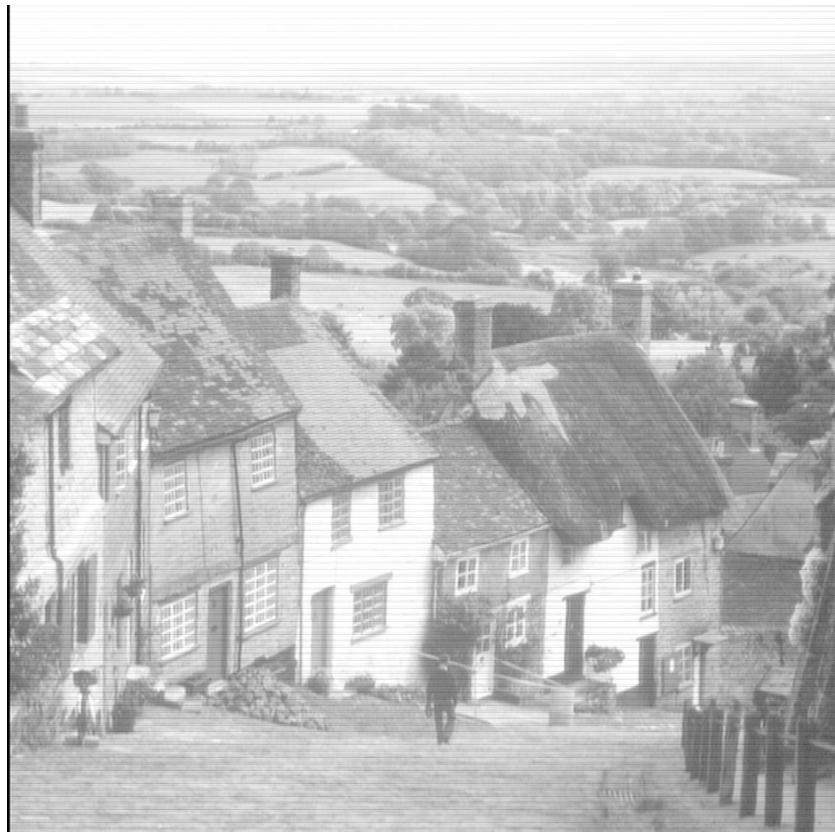


Figura 20 - “city.png” após Interpolação por Polinômios de Lagrange

4.4. Imagem “house.png”



Figura 21 - “house.png” após rotação

A imagem “house.png” foi rotacionada em $17,647^\circ$, resultando na *Figura 21*.

As *figuras 22, 23, 24 e 25* mostram a mesma imagem original após as interpolações pelo Vizinho Mais Próximo, Bilinear, Bicúbica e por Polinômios de Lagrange, respectivamente. Foi aplicado um fator de escala de 3.



Figura 22 - “house.png” após Interpolação pelo Vizinho Mais Próximo



Figura 23 - “house.png” após Interpolação Bilinear



Figura 24 - "house.png" após Interpolação Bicúbica

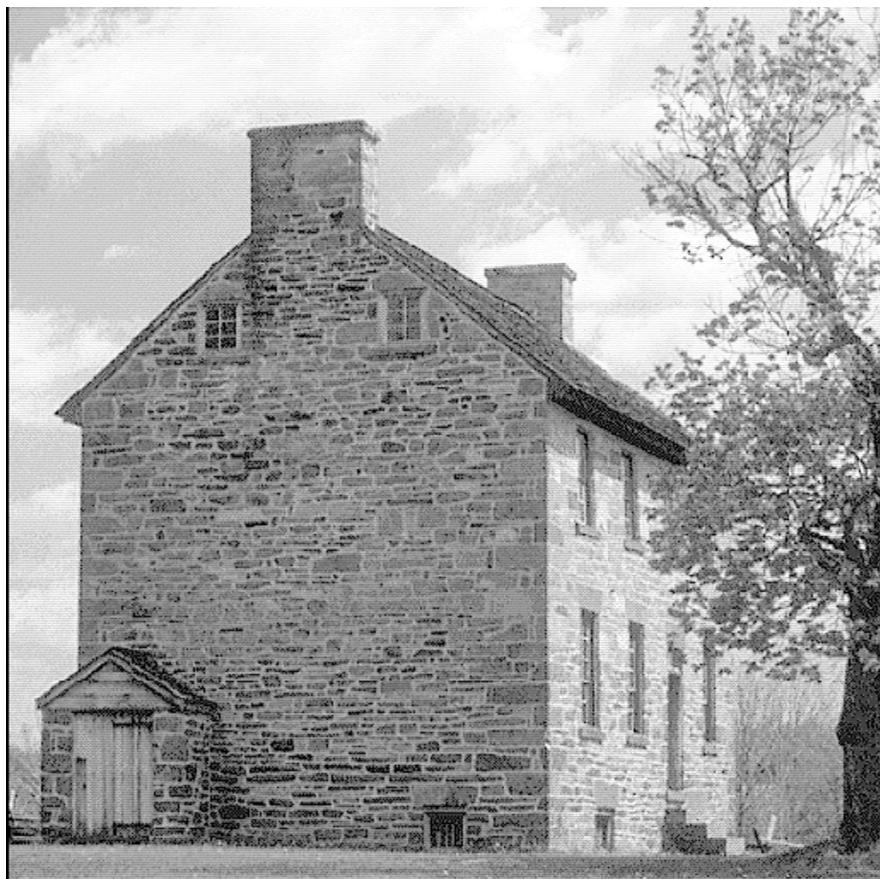


Figura 25 - "house.png" após Interpolação por Polinômios de Lagrange

4.5. Imagem “seagull.png”



Figura 26 - “seagull.png” após rotação

A imagem “seagull.png” foi rotacionada em 15,75°, resultando na *Figura 26*.

As *figuras 27, 28, 29 e 30* mostram a mesma imagem original após as interpolações pelo Vizinho Mais Próximo, Bilinear, Bicúbica e por Polinômios de Lagrange, respectivamente. Foi aplicado um fator de escala de 3,25.

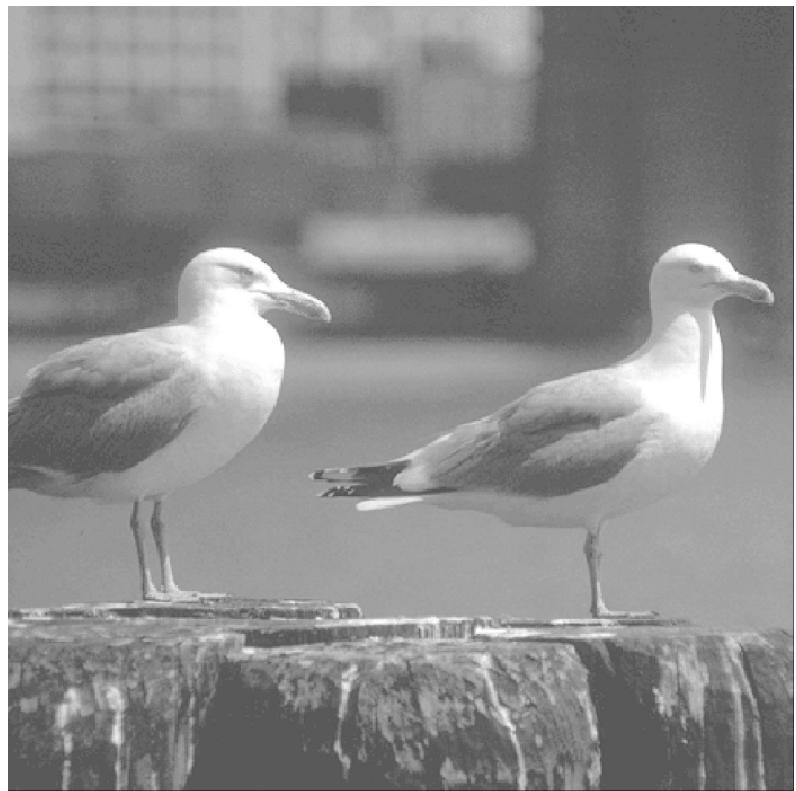


Figura 27 - “seagull.png” após Interpolação pelo Vizinho Mais Próximo



Figura 28 - “seagull.png” após Interpolação Bilinear



Figura 29 - “seagull.png” após Interpolação Bicúbica

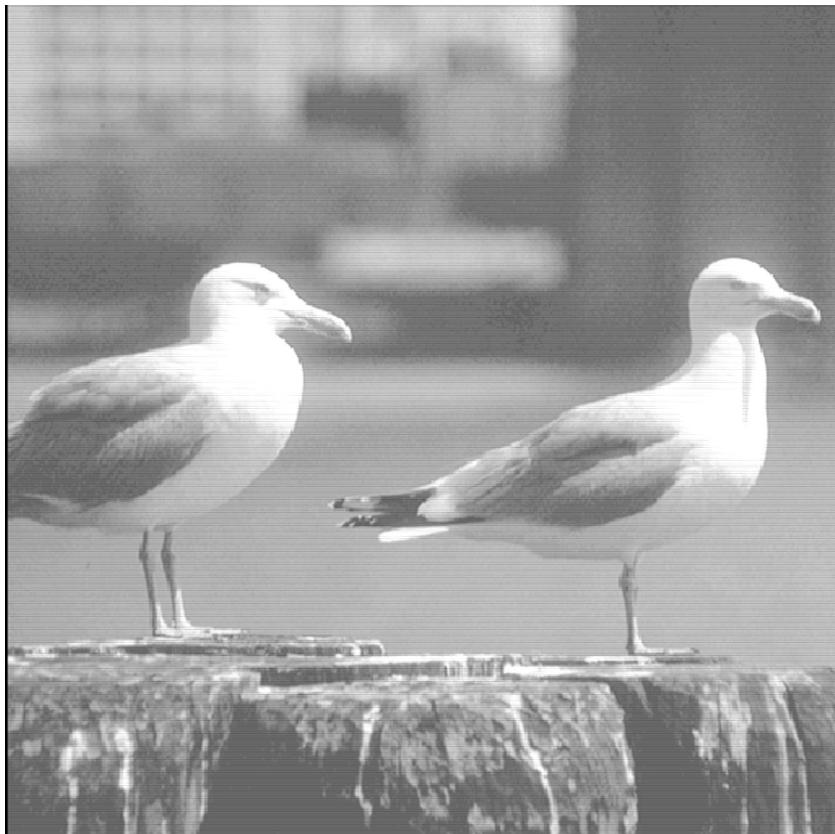


Figura 30 - “seagull.png” após Interpolação por Polinômios de Lagrange

4.6. Imagem “DSC00114.png”



Figura 31 - “DSC00114.png” após rotação

A imagem “DSC00114.png” foi rotacionada em 25,15°, resultando na *Figura 31*.

As *figuras 32, 33, 34 e 35* mostram a mesma imagem original após as interpolações pelo Vizinho Mais Próximo, Bilinear, Bicúbica e por Polinômios de Lagrange, respectivamente. Para a primeira interpolação, a fim de testar a execução do programa explicitando as dimensões da imagem resultante, foram utilizadas as dimensões 1024 x 1024 pixels. Para as demais interpolações, foi aplicado um fator de escala de 0,35.



Figura 32 - “DSC00114.png” após Interpolação pelo Vizinho Mais Próximo (com dimensões x e y da imagem resultante explicitadas nos parâmetros de execução)



Figura 33 - “DSC00114.png” após Interpolação Bilinear



Figura 34 - “DSC00114.png” após Interpolação Bicúbica



Figura 35 - “DSC00114.png” após Interpolação por Polinômios de Lagrange

5. Análise dos Resultados e Conclusões

Como já foi observado, o fato do rotacionamento ocorrer em relação à origem fez com que, dependendo do ângulo de rotação, uma parte da imagem seja “cortada”. Caso alguém tenha interesse em utilizar e aprimorar o programa escrito, sugere-se tentar realizar abordagens na rotação que mantenham pelo menos boa parte da imagem independentemente do ângulo de rotação. Uma possibilidade seria deixar que o *ndarray* da imagem “cresça” durante o preenchimento e posteriormente seja feita uma re-centralização do que efetivamente consiste na imagem rotacionada.

Quanto aos métodos de interpolação, as imagens colocadas nas seções anteriores podem não fazer diferença ao leitor em virtude do tamanho no papel ou de limitações na visualização em tela. Dessa forma, foi escolhida uma região da imagem “DSC00114.png” para ser selecionada na imagem original e nas imagens resultantes das interpolações.



Figura 36 - Região da imagem original “DSC00114.png”



Figura 37 - Região da imagem “DSC00114.png” após Interpolação pelo Vizinho Mais Próximo



Figura 38 - Região da imagem “DSC00114.png” após Interpolação Bilinear



Figura 39 - Região da imagem “DSC00114.png” após Interpolação Bicúbica



Figura 40 - Região da imagem “DSC00114.png” após Interpolação por Polinômios de Lagrange

É possível perceber pelas *Figuras 37, 38, 39 e 40*, em comparação com a *Figura 36*, que a Interpolação por Polinômios de Lagrange obtém melhor resultado quando se trata de preservação de detalhes. Uma referência que pode ser utilizada é o adesivo contendo o brasão da Cidade de São Paulo e o texto “PREFEITURA DE SÃO PAULO” na lateral do ônibus em destaque. O pior resultado notadamente foi o obtido com a Interpolação pelo Vizinho Mais Próximo, no qual o texto é praticamente ilegível.

Outra observação em relação aos resultados obtidos é de que a Interpolação por Polinômios de Lagrange, em todos os casos, resultou em imagens com listras horizontais. Provavelmente foram causados por alguma falha na implementação do método, entretanto, isso não prejudicou o nível de detalhamento da imagem resultante. Outra sugestão a quem utilizar futuramente o código escrito é analisar a implementação deste método de interpolação a fim de eliminar as listras horizontais.

A Interpolação por Polinômios de Lagrange destaca-se não apenas pelo melhor nível de detalhamento resultante mas também por ser consideravelmente menos custosa computacionalmente do que a Interpolação Bicúbica. Esta, para cada pixel, realiza dois laços do tipo *for* aninhados, enquanto Lagrange utiliza apenas um.

Evidentemente, caso a prioridade seja por um resultado obtido mais rapidamente, a Interpolação pelo Vizinho Mais Próximo é uma opção bastante viável, estando ciente da maior possibilidade perda de detalhamento.