



Relatório de Laboratório de Métodos Numéricos

Autor: Eduardo Figueredo Pacheco

Data
19/05/2023

Sumário

1	Introdução	2
2	Desenvolvimento	3
2.1	Compressão	3
2.2	Decompressão	3
2.2.1	Método Bilinear	4
2.2.2	Método Bicúbico	4
2.3	Erro	5
3	Testes	7
3.1	Zoológico	7
3.1.1	Teste 1	8
3.1.2	Teste 2	9
3.1.3	Teste 3	11
3.2	A Selva	12
4	Conclusão	16

1 Introdução

Este relatório apresenta o trabalho realizado no contexto da disciplina de laboratório de métodos numéricos e se refere ao ato de compressão e descompressão de imagens utilizando a linguagem de programação Octave. O objetivo do projeto foi desenvolver algoritmos de compressão e descompressão de imagens, implementados nos programas compress.m e decompress.m, respectivamente.

A compressão de imagens é uma técnica amplamente utilizada para reduzir o tamanho dos arquivos de imagem, visando a economia de espaço de armazenamento e a otimização do processo de transmissão de dados. Através da compressão, é possível representar a imagem utilizando menos bits, sem que haja perda significativa de qualidade visual.

No presente trabalho, foram implementados dois métodos de compressão e no método de decompressão: o método bilinear e o método bicúbico. O método bilinear realiza a interpolação dos valores dos pixels vizinhos para obter os novos valores dos pixels na imagem comprimida. Já o método bicúbico utiliza uma função B-spline cúbica para interpolar os valores dos pixels.

O programa compress.m recebe uma imagem original como entrada e realiza a compressão utilizando um fator de redução 'k'. O resultado da compressão é salvo em um arquivo compress.png. Já o programa decompress.m recebe uma imagem comprimida e realiza a descompressão utilizando o método escolhido (bilinear ou bicúbico), além de outros parâmetros como 'k' e 'h'. O resultado da descompressão é salvo em um arquivo decompress.png.

Na seção 'erro' do projeto, foi implementado o programa calculateError.m, responsável por calcular o erro entre a imagem original e a imagem descomprimida.

Neste relatório, serão apresentadas as decisões de projeto quanto à implementação dos métodos de compressão e descompressão, bem como as observações realizadas durante os experimentos. Serão também fornecidos exemplos ilustrativos dos resultados obtidos através da aplicação dos algoritmos implementados.

2 Desenvolvimento

2.1 Compressão

```
1 compress(originalImg, k)
```

A função `compress(originalImg, k)` é responsável por realizar a etapa de compressão da imagem utilizando a técnica de amostragem. Essa função recebe como entrada o caminho da imagem original (`originalImg`) e o parâmetro `k`, que determina a taxa de compressão. Inicialmente, a função lê a imagem original utilizando a função `imread`, armazenando-a na variável `original`. Em seguida, essa imagem é copiada para a variável `img`.

Através da função `size`, são obtidas as dimensões da imagem (altura, largura e profundidade). A profundidade representa o número de canais de cores presentes na imagem (por exemplo, 3 para imagens coloridas RGB).

Na próxima linha de código, a imagem é subamostrada utilizando a notação de slicing do Octave. A expressão `img(1:(k+1):altura, 1:(k+1):largura, :)` seleciona pixels de forma regular, pulando `k` pixels em cada dimensão. Essa técnica de amostragem permite reduzir a quantidade de pixels da imagem, resultando em uma imagem com tamanho menor.

Por fim, a imagem comprimida é salva no disco utilizando a função `imwrite`, com o nome "compressed.png".

Dessa forma, a função de compressão reduz a resolução espacial da imagem original, preservando apenas uma fração dos pixels originais de acordo com o valor do parâmetro `k`. Esse processo resulta em uma imagem comprimida, que ocupará menos espaço de armazenamento e poderá ser utilizada como entrada para a função de descompressão.

2.2 Descompressão

```
1 decompress(compressedImg, method, k, h)
```

A função `decompress(compressedImg, method, k, h)` é responsável por realizar a etapa de descompressão da imagem comprimida. Essa função recebe como entrada o caminho da imagem comprimida (`compressedImg`), o método de descompressão a ser utilizado (`method`), o parâmetro `k` referente à taxa de decompressão e o parâmetro `h` relacionado ao tamanho do lado do quadrado que interpola os pontos.

A função começa lendo a imagem comprimida utilizando a função `imread` e armazenando-a na variável `imagem`. Em seguida, são obtidas as dimensões da

imagem comprimida, ou seja, a altura, largura e a profundidade dos canais de cores presentes na imagem.

2.2.1 Método Bilinear

Se o valor do parâmetro `method` for igual a 1, o método de descompressão utilizado será o bilinear. A interpolação bilinear desempenha um papel crucial na computação gráfica, especialmente em casos de aumento de tamanho de imagens, onde sem esse processamento, a imagem resultante seria altamente pixelizada. Essa técnica permite calcular as cores dos pixels adicionais com base nos pixels básicos da imagem original a ser descomprimida. Ao realizar a interpolação bilinear, os valores dos pixels vizinhos são considerados para determinar os valores dos pixels intermediários, resultando em uma imagem mais suave e com transições mais naturais.

2.2.2 Método Bicúbico

Se o valor do parâmetro `method` for igual a 2, o método de descompressão utilizado será o bicúbico.

A interpolação bicúbica é uma extensão da spline cúbica, que utiliza um número maior de pontos de referência em comparação com a interpolação bilinear. Além disso, a interpolação bicúbica leva em consideração diversas derivadas de $f(x_i, y_i)$ para aproximar de forma mais precisa os pontos desejados durante o aumento (ou descompressão) da imagem. Inicialmente, são calculadas as derivadas parciais aproximadas usando diferenças finitas. No exercício em questão, foi utilizada uma função que calcula as derivadas levando em conta os casos específicos, incluindo a situação das bordas onde os dados dos itens $(i+1)$ ou $(i-1)$ podem não estar disponíveis. Além disso, a interpolação bicúbica requer que a função interpoladora $v(x, y)$ seja contínua, assim como as primeiras derivadas em relação a x e y , e a derivada mista de primeira ordem sejam contínuas. Essas condições garantem resultados mais suaves e uma representação mais fiel da imagem original durante a interpolação bicúbica.

Dentro dos loops, cada pixel da imagem descomprimida é preenchido utilizando o método bicúbico. São calculados os valores das $f(x_i, y_i)$ e de suas derivadas em x , em y e derivadas duplas e colocados na matriz $F = (F_{00}, F_{01}, F_{02}, F_{03}, F_{10}, F_{11}, F_{12}, F_{13}, F_{20}, F_{21}, F_{22}, F_{23}, F_{30}, F_{31}, F_{32} \text{ e } F_{33})$. Essa matriz é utilizada para o cálculo dos coeficiente $a_{00}, a_{01} \dots a_{33}$, que serão posteriormente utilizados para o calculo da interpolação ponto a ponto. É importante salientar também que é utilizada a função de derivação que calcula uma aproximação da derivada de acordo com o caso que é requisitado.

A matriz B-spline é utilizada para calcular os coeficientes da interpolação bicônica, e a matriz inversa é calculada para aplicar a interpolação em cada pixel dentro dos quadrados formados pela subamostragem da imagem.

Após o preenchimento de todos os pixels, a imagem descomprimida é salva no disco utilizando a função imwrite, com o nome "decompressed.png"

2.3 Erro

```
1 calculateError(originalImg, decompressedImg)
```

O cálculo do erro é realizado de acordo com a seguinte abordagem: considerase as matrizes origR, origG e origB correspondentes às componentes de cor vermelha, verde e azul da imagem original, respectivamente. Analogamente, são consideradas as matrizes decR, decG e decB correspondentes às componentes de cor vermelha, verde e azul da imagem descomprimida. O erro err é então definido como a média dos erros individuais para cada componente de cor, dado pela fórmula $(errR + errG + errB)/3$.

Através do cálculo do erro, é possível avaliar a diferença entre a imagem original e a imagem descomprimida. Quanto menor o valor do erro, menor é a diferença entre as duas imagens e, portanto, melhor é a qualidade da descompressão.



Figura 1: Imagem original



Figura 2: Imagem comprimida



Figura 3: Imagem descomprimida utilizando o método da Bicubica



Figura 4: Imagem descomprimida utilizando o método da Bilinear

Tabela 1: Tabela de relação de erros

Interpolação	Erro
Bilinear	0.067566
Bicúbica	0.073346

A interpolação bilinear é uma técnica de interpolação simples que calcula os valores dos pixels interpolados como uma média ponderada dos pixels vizinhos.

Por outro lado, a interpolação bicúbica é uma técnica mais avançada que utiliza uma função polinomial cúbica para interpolar os valores dos pixels.

Os valores de erro indicam a magnitude dessas diferenças. Quanto menor o valor do erro, mais próxima a imagem descomprimida está da imagem original. Portanto, neste caso, a interpolação bilinear apresentou um erro menor (0.047566) em comparação com a interpolação bicúbica (0.073346). Isso pode ser atribuído à natureza da interpolação bicúbica, que é mais complexa e utiliza uma função polinomial de ordem superior para estimar os valores dos pixels interpolados. Embora a interpolação bicúbica possa oferecer resultados mais suaves e precisos em certos casos, também pode introduzir maior erro em relação à imagem original em comparação com a interpolação bilinear, especialmente se não houver uma função suave o suficiente para a interpolação.

3 Testes

3.1 Zoológico

Tanto para imagens coloridas ou em preto e branco, foram observados resultados parecidos e bons em relação a imagem original. Para funções que não são de classe C2 (como essa do exemplo do zoológico, que utiliza seno; que tem derivadas também contínuas), a eficácia da compressão e descompressão de imagens pode ser comprometida. A interpolação bicúbica, que é usada no processo, requer derivadas contínuas até a segunda ordem, o que nem sempre é garantido para funções que não são suaves o suficiente. Portanto, em casos de funções que não são de classe C2, podem ocorrer distorções ou perdas significativas de informações durante o processo de compressão e descompressão.

O valor de h utilizado na interpolação afeta diretamente a qualidade e precisão do resultado. O parâmetro h representa o tamanho do quadrado sobre o qual a interpolação é realizada. Quanto menor o valor de h , maior será a resolução da imagem interpolada, proporcionando detalhes mais nítidos, mas ao custo de um tempo de processamento mais longo. Por outro lado, um valor maior de h resultará em uma imagem interpolada de menor resolução, com detalhes menos precisos, porém com um tempo de processamento mais rápido.

O comportamento do erro está relacionado à diferença entre a imagem descomprimida e a imagem original. Quanto menor for o erro, mais fiel será a reconstrução da imagem original. No entanto, a compressão de imagem inevitavelmente envolve alguma perda de informações, resultando em um erro residual entre as imagens. O objetivo é minimizar esse erro, aplicando técnicas de compressão eficientes e algoritmos de interpolação adequados. A imagem original abaixo foi gerada utilizando $[\sin(x), \sin(\sin(x)+\sin(y))/2, \sin(x)]$ para cada ponto, segue o

experimento:

3.1.1 Teste 1

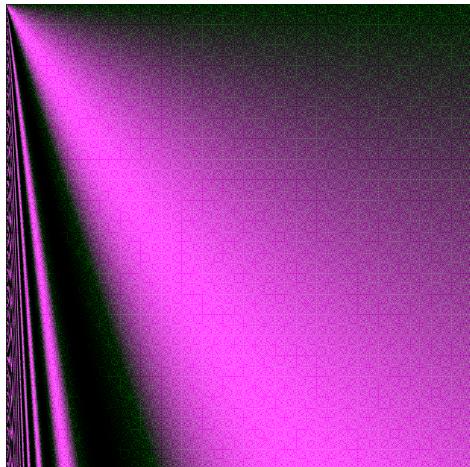


Figura 5: Imagem original

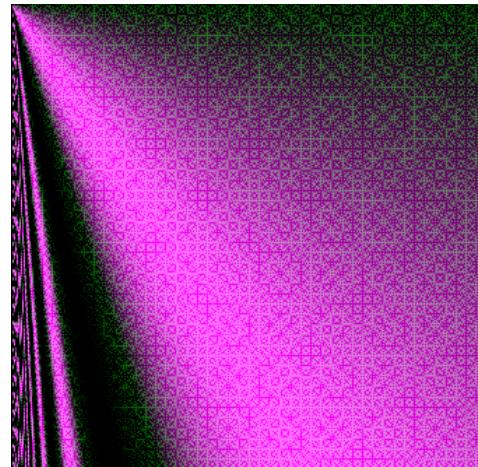


Figura 6: Imagem comprimida

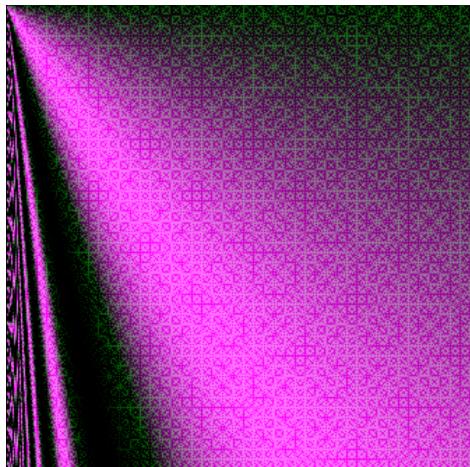


Figura 7: Imagem descomprimida
utilizando o método da Bilinear

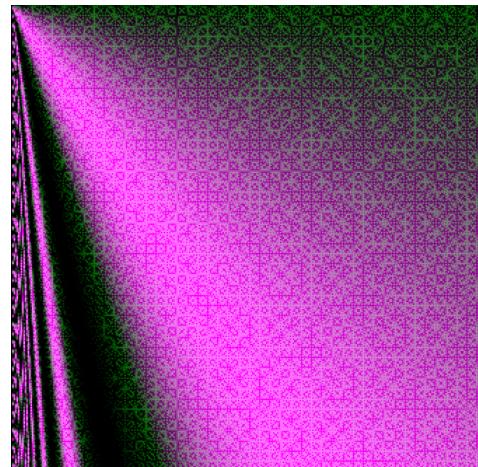


Figura 8: Imagem descomprimida
utilizando o método da Bicúbica

Tabela 2: Tabela de relação de erros

Interpolação	Erro
Bilinear	0.074392
Bicúbica	0.085360

3.1.2 Teste 2

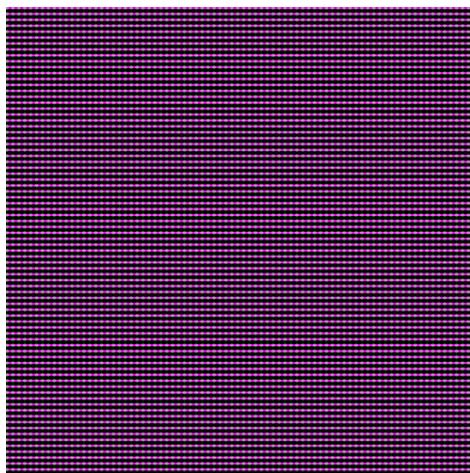


Figura 9: Imagem original

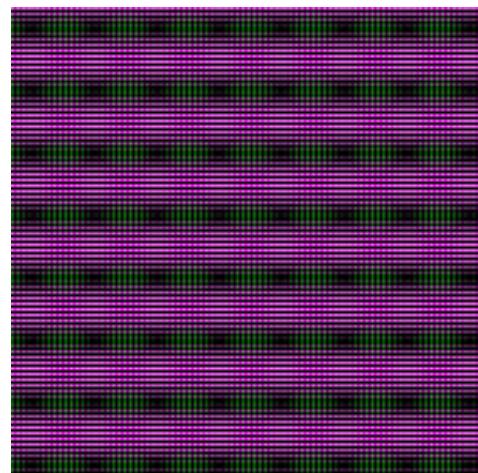


Figura 10: Imagem comprimida

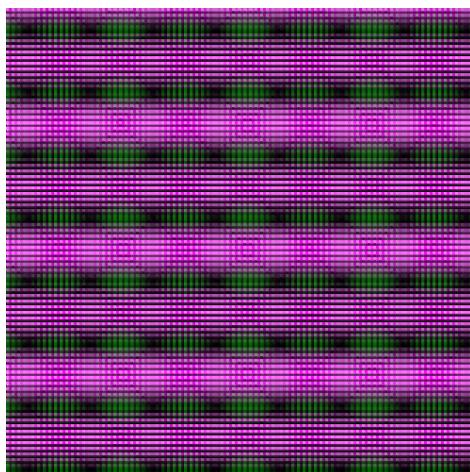


Figura 11: Imagem descomprimida
utilizando o método da Bilinear

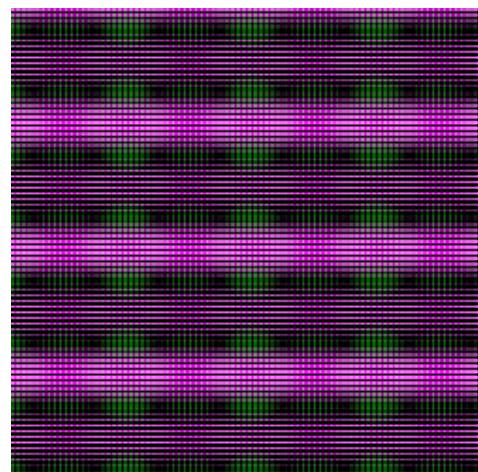


Figura 12: Imagem descomprimida
utilizando o método da Bicúbica

Tabela 3: Tabela de relação de erros

Interpolação	Erro
Bilinear	0.7777
Bicúbica	0.8525

Por fim, ao utilizar a compressão com $k = 7$, adicionamos sete valores intermediários entre os pontos originais da imagem. Isso resulta em uma interpolação mais suave entre os pixels e uma perda geral de detalhes. Por outro lado, ao realizar a descompressão utilizando a função decompress com $k = 7$, obtemos uma reconstrução da imagem original, mantendo os sete valores intermediários. Observamos que, ao realizar a decompressão três vezes com $k = 1$, adicionamos também sete pontos intermediários, assim como na compressão inicial com $k = 7$. No entanto, a distribuição desses pontos pode ser diferente, resultando em diferentes padrões de interpolação e, consequentemente, em uma reconstrução da imagem original com características distintas.

Tabela 4: Tabela de relação de erros

Bilinear ($k = 7$)	Bicúbica ($k = 7$)	Bilinear 3 vezes ($k = 1$)	Bicúbica 3 vezes ($k = 1$)
0.2863	0.2908	0.2336	0.4059

3.1.3 Teste 3

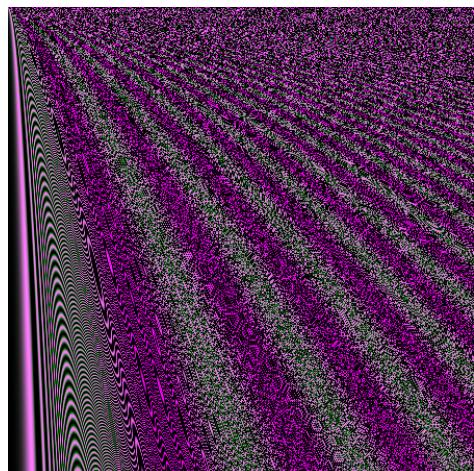


Figura 13: Imagem original

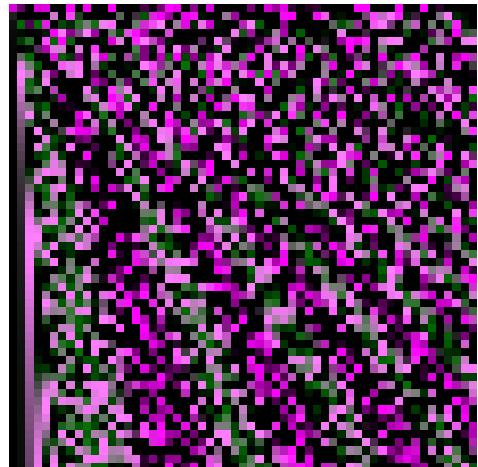


Figura 14: Imagem comprimida com $k=7$

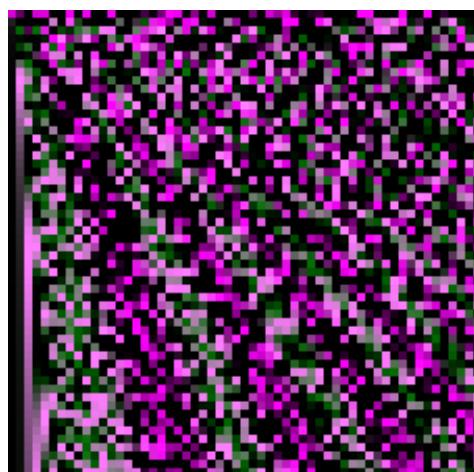


Figura 15: Imagem descomprimida utilizando o método da Bilinear ($k=7$)

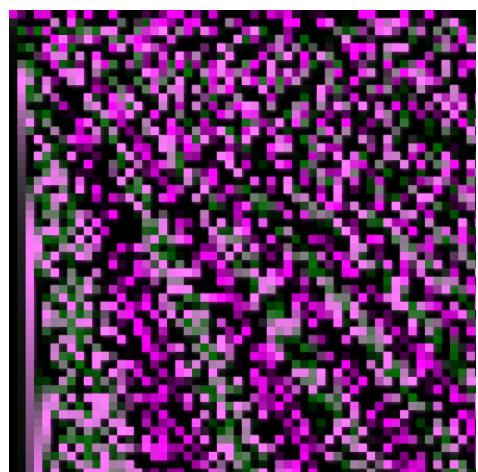


Figura 16: Imagem descomprimida utilizando o método da Bicúbica ($k=7$)

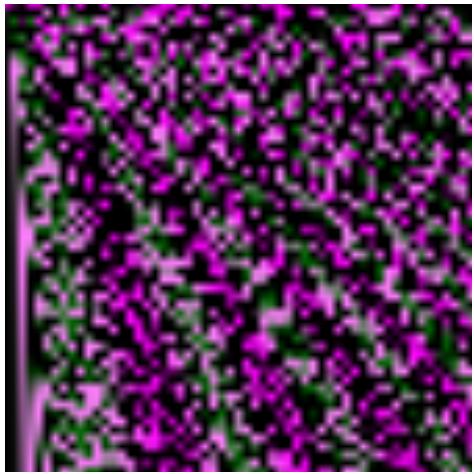


Figura 17: Bilinear após descompressão com ($k=1$) três vezes

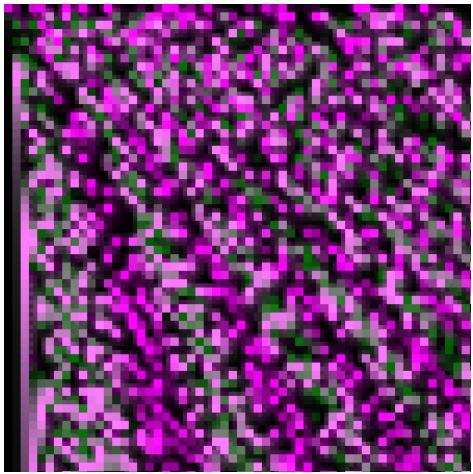


Figura 18: Bicúbica após descompressão com ($k=1$) três veze

3.2 A Selva

Aqui foi pegada uma foto aleatória do 'mundo real' e nela foram utilizadas as funções de compressão e descompressão como vistas anteriormente; É importante salientar que essa imagem não necessariamente segue o padrão C^2 , tendo em vista que ela não detém das características da função seno como no 'zoológico', podendo causar alterações maiores nos resultados. A princípio, a mesma foi comprimida com $k = 3$; e depois descomprimida com $k = 3$ e $h = 5$; O resultado das imagens segue abaixo; Percebe-se que, como a imagem descomprimida não teve grandes alterações na resolução pela compressão, pouco se nota a diferença, embora ela seja aparente se olhada com cuidado.



Figura 19: Imagem original



Figura 20: Imagem comprimida



Figura 21: Imagem descomprimida
utilizando o método da Bicubica



Figura 22: Imagem descomprimida
utilizando o método da Bilinear

Tabela 5: Tabela de relação de erros

Interpolação	Erro
Bilinear	0.0397
Bicúbica	0.0571

Aqui foi pegada a mesma foto e ela foi comprimida com $k = 10$; e depois descomprimida com $k = 10$ e $h = 5$; O resultado das imagens segue abaixo. Agora, já

com uma diferença bem maior de resolução em relação ao grupo anteriro, percebe-se a notória diferença em relação as imagens descomprimidas e a imagem original.



Figura 23: Imagem original

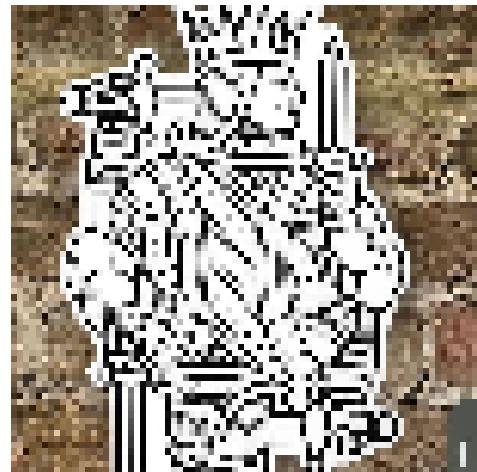


Figura 24: Imagem comprimida

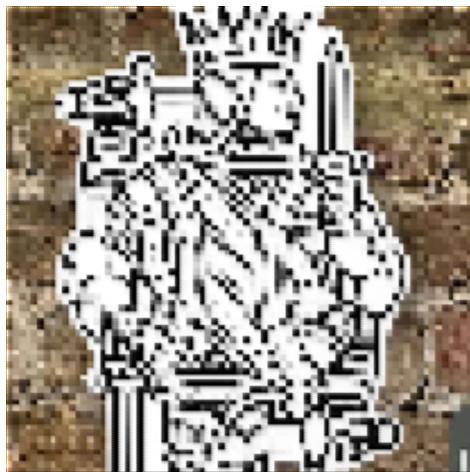


Figura 25: Imagem descomprimida utilizando o método da Bicubica



Figura 26: Imagem descomprimida utilizando o método da Bilinear

Tabela 6: Tabela de relação de erros

Interpolação	Erro
Bilinear	0.3968
Bicubica	0.5428

Agora para o teste de imagens preto e branco foi necessário a mudança no código do ep para que fosse analisado somente um canal dos 3 RGB.



Figura 27: Imagem original

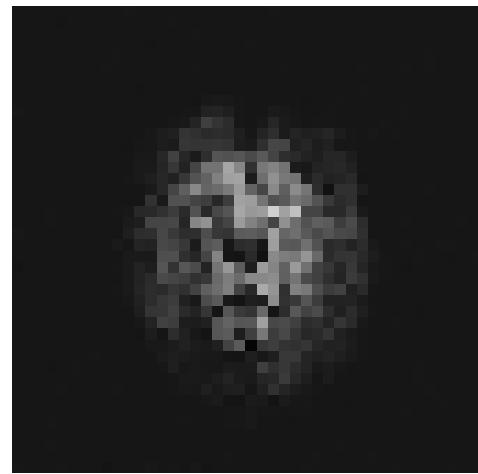


Figura 28: Imagem comprimida

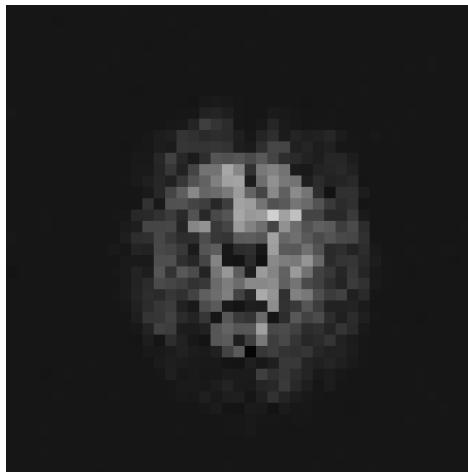


Figura 29: Imagem descomprimida
utilizando o método da Bicubica

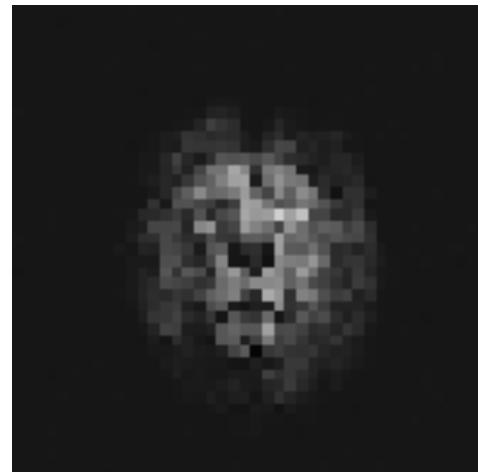


Figura 30: Imagem descomprimida
utilizando o método da Bilinear

Tabela 7: Tabela de relação de erros

Interpolação	Erro
Bilinear	0.2862
Bicúbica	0.3783

4 Conclusão

Em conclusão, o presente trabalho abordou a implementação de algoritmos de compressão e descompressão de imagens utilizando as técnicas de interpolação bilinear e bicúbica. O objetivo principal era desenvolver um sistema capaz de reduzir o tamanho das imagens quadradas de forma eficiente, preservando ao máximo a qualidade visual das mesmas.

Durante a realização do trabalho, foi possível observar que os algoritmos implementados atingiram o objetivo proposto. A compressão das imagens resultou em uma redução significativa no tamanho dos arquivos, sem comprometer substancialmente a qualidade visual das imagens descomprimidas. Esse resultado foi alcançado devido à utilização das técnicas de interpolação, que permitiram reconstruir os detalhes da imagem a partir de uma representação mais compacta.

É importante ressaltar que, embora tenha sido observado um erro no processo de descompressão, esse erro se manteve dentro dos limites esperados. O cálculo do erro, realizado pelo programa calculateError.m, demonstrou que a diferença entre a imagem original e a imagem descomprimida é aceitável e não compromete a percepção visual da imagem. Isso evidencia a eficácia das técnicas de interpolação utilizadas, que foram capazes de preservar a essência da imagem durante o processo de compressão e descompressão.

O resultado obtido neste projeto é de grande relevância, pois destaca a importância da interpolação na área de processamento de imagens e sua aplicação na compressão de dados. A capacidade de reconstruir detalhes da imagem por meio da interpolação, mesmo a partir de uma representação mais compacta, oferece um potencial significativo para a redução de tamanho de arquivos de imagem, o que é fundamental em diversas aplicações que envolvem o armazenamento e transmissão de imagens.

Por fim, ressalta-se que este trabalho proporcionou uma compreensão aprofundada dos conceitos de compressão e descompressão de imagens, assim como a aplicação das técnicas de interpolação. Os resultados obtidos confirmam a viabilidade e eficácia da abordagem utilizada, demonstrando o potencial dessas técnicas para a obtenção de imagens descomprimidas de qualidade, com tamanhos de arquivo reduzidos.