EP2 - Laboratório de Métodos Númericos

Gabriel H. P. Rodrigues
Rafael Vieira De Carvalho

Junho 2020

Métodos implementados

- **decompress(compressedImg, method, k, h)** - Função descompressora, usando o método imread a imagem é transformada em matriz pelo Octave, a decisão de projeto foi transformar toda matriz lida para o tipo double, sendo assim as representações de cores ficam no intervalo de 0 a 1.
- **compress(originalImg)** - Função compressora.
- **calculateError(originalImg, decompressedImg)** - Função que calcula o erro entre a imagem original e a imagem interpolada, a matriz lida é transformada em double, para imagens coloridas o erro é calculado como especificado no enunciado do EP2, para imagens não coloridas o erro é calculado de maneira semelhante, só que ao final não há a divisão por 3, mas sim por 1.
- **funcao.m** - Arquivo que contém um conjunto de funções que são utilizadas no Zoológico.

1 Zoológico

Vamos testar a nossa interpolação utilizando algumas funçoes do tipo $f : R^2 \to R^3$ para imagens coloridas, para imagens em preto e branco usaremos funçoes do tipo $f : R^2 \to R$.

Para $f(x, y) = (sen(x), \frac{sen(x)+sen(y)}{2}, sen(x))$ as seguinte imagens foram geradas:

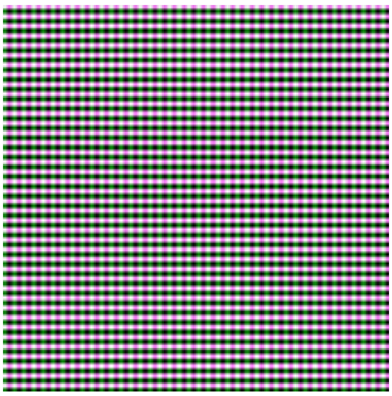

Figura 1: Imagem original

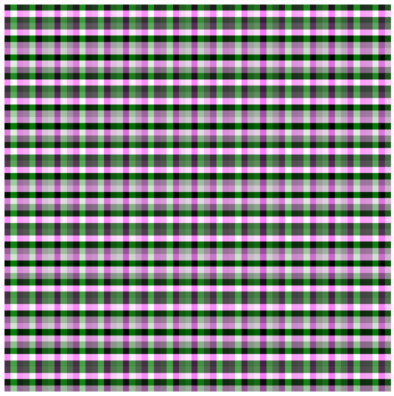

Figura 2: Imagem comprimida $k = 3$

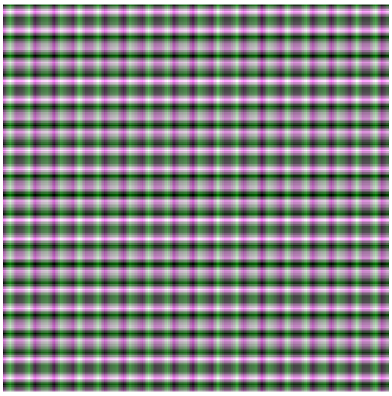

Figura 3: Descompressão bilinear
$k = 3$ e $h = 0.01$

Erro bilinear $= 0.67107$

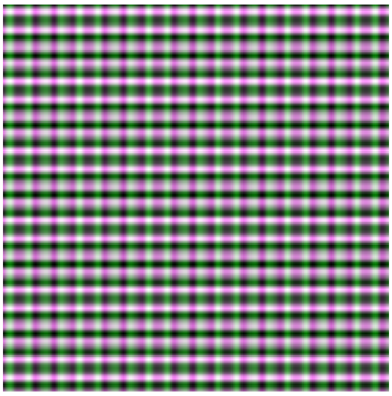

Figura 4: Descompressão bicúbica
$k = 3$ e $h = 0.01$

Erro bicúbico $= 0.69503$

Como estamos representado a imagem como uma matriz de double com os elementos variando no intervalo $[0, 1]$ os valores negativos da função estão sendo mapeados para o intervalo $[0, 0.5]$, isto é, o valor -1 equivale a 0, o valor de -0.5 equivale a -0.25 e assim por diante. Os valores positivos estão sendo mapeados para o intervalo $]0.5, 1]$

Para a função $f(x, y) = (xy,\ x^2 + y^2,\ y^3)$ os seguintes resultados foram obtidos:

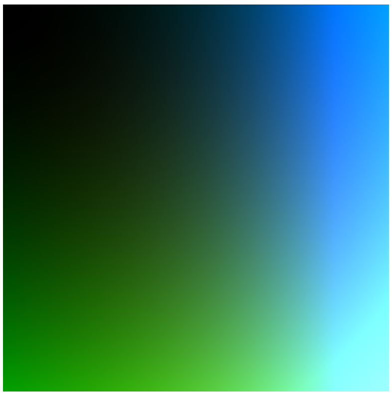

Figura 5: Imagem original

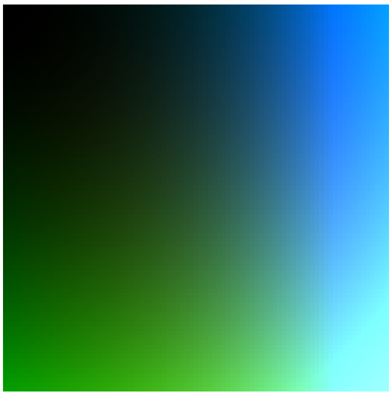

Figura 6: Imagem comprimida $k = 2$

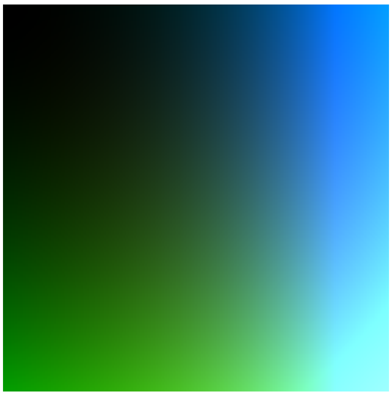

Figura 7: Descompressão bilinear
$k = 2$ e $h = 1$

Erro bilinear $= 0.022766$

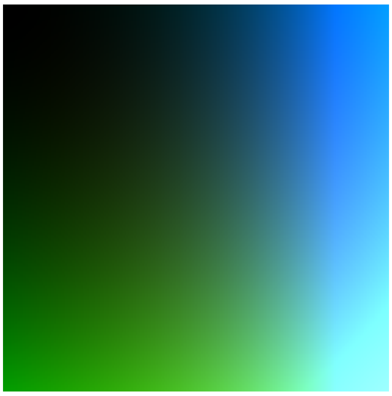

Figura 8: Descompressão bilinear
$k = 2$ e $h = 0.1$

Erro bilinear $= 0.022766$

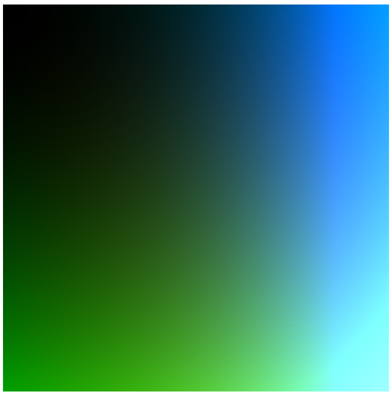

Figura 9: Descompressão bicúbico
$k = 2$ e $h = 1$
Erro bicúbico $= 0.022756$

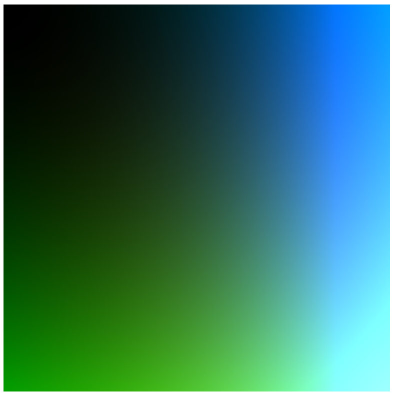

Figura 10: Descompressão bicúbico
$k = 2$ e $h = 0.1$
Erro bicubico $= 0.022756$

Tabela de erros da descompressão $k = 3$

Interpolação Bilinear

	$h = 10$	$h = 1$	$h = 0.1$	$h = 0.01$
k = 3	0.0467 **6052197926**	0.0467 **6019065049**	0.0467 **5434146961**	0.0467 **5777526625**

Interpolação Bicúbica

	$h = 10$	$h = 1$	$h = 0.1$	$h = 0.01$
k = 3	0.04662 **172859901**	0.04662 **174917182**	0.04662 **077724798**	0.04662 **138277218**

Como no exemplo anterior, tivemos que mapear essa função no intervalo $[0, 1]$. Para fazer isso dividimos o valor de cada coordenada, do vetor de 3 coordenadas, pela ordem de grandeza do máximo global que ele pode assumir, por exemplo, para a primeira coordenada (xy) dividimos pela ordem de grandeza do máximo global que é n^2 (n é o tamanho da imagem).

Agora vamos testar para imagens em preto e branco $f(x, y) = sin(x + y)$

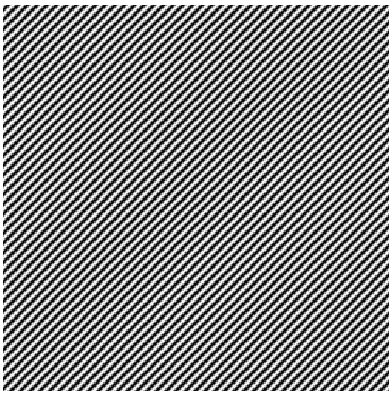

Figura 11: Imagem Original

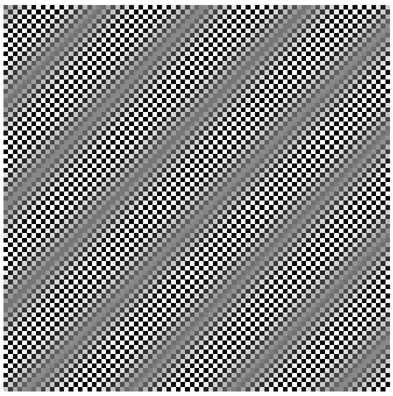

Figura 12: Imagem comprimida $k = 2$

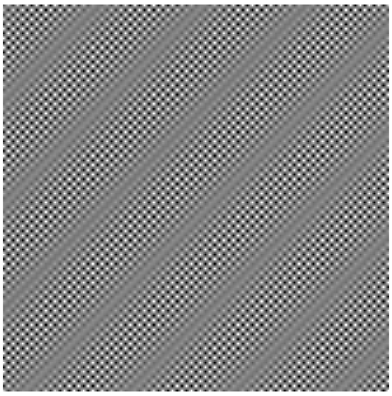

Figura 13: Descompressão bilinear $k = 2$ e $h = 0.1$

Erro bilinear $= 0.60337$

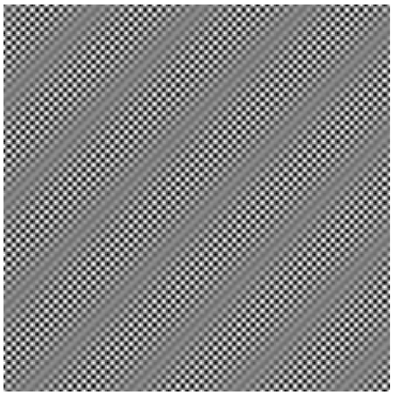

Figura 14: Descompressão bicúbica $k = 2$ e $h = 0.1$

Erro bicúbico $= 0.63747$

$$f(x, y) = \begin{cases} x^2 sin(\frac{1}{x-1}) & \text{se } x \neq 0 \\ 0 & \text{se } x = 0 \end{cases}$$

Figura 15: Imagem original

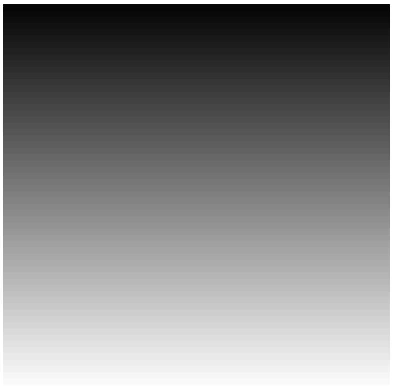

Figura 16: Imagem comprimida $k = 3$

Figura 17: Descompressão bilinear $k = 3$ e $h = 0.1$

Erro bilinear $= 0.020248$

Figura 18: Descompressão bicúbica $k = 3$ e $h = 0.1$

Erro bicúbico $= 0.020248$

Esse ultimo experimento foi pra mostrar que mesmo a função não sendo de classe C^2 ainda conseguimos interpolar a imagem.

Teste das descompressões sucessivas

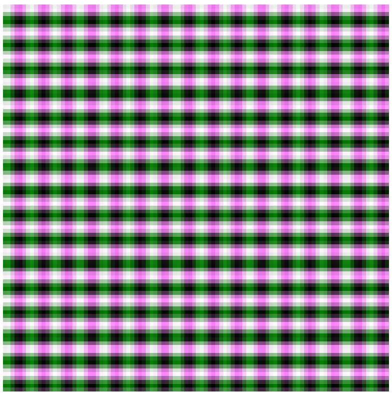

Figura 19: Imagem original

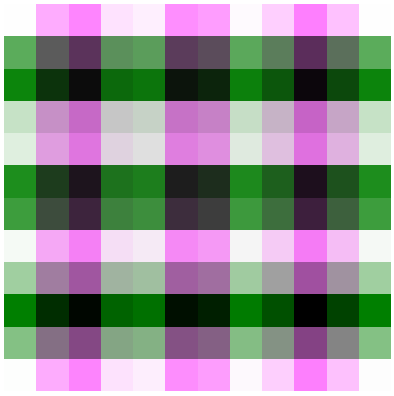

Figura 20: Imagem comprimida $k = 7$

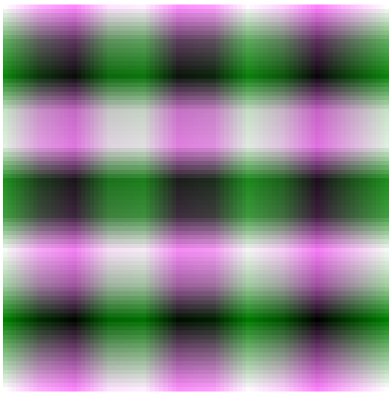

Figura 21: Descompressão bilinear
$k = 7$ e $h = 1$

Erro bilinear = 0.54390 **5218800**

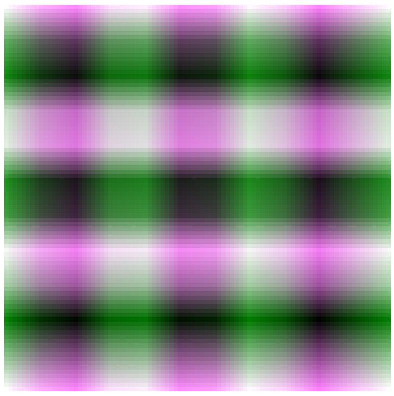

Figura 22: 3 Descompressões bilineares sucessivas
$k = 1$ e $h = 1$

Erro bicúbico = 0.54390 **6687462**

É verificável que a imagem com descompressões sucessivas apresenta um maior erro, visto que a cada descompressão a interpolação utiliza dados não originais (dados interpolados) da imagem comprimida.

Conclusões

Após realizar esses experimentos percebemos que a interpolação funciona tanto para imagens coloridas quanto para imagens em preto e branco. Percebe-se também que funciona tanto para funçoes de classe C^2 quanto para funções que não são de classe C^2 (isso ficará mais claro na Selva). Notou-se que, embora diminuindo o valor de h, o erro não diminuiu (diferente do que é proposto na teoria), isto pode estar atrelado ao fato da quantidade de bits do computador ser limitada e, portanto, houveram erros de arredondamento. Notou-se também que a compressão realizada não é uma das melhores, poís ela altera a imagem, como pode-se perceber nas imagens 2 e 12.

2 A Selva

Teste 1 - Imagem Axolote

O mesmo K utilizado na compressão também foi utilizado na descompressão.

TESTE: K = 1

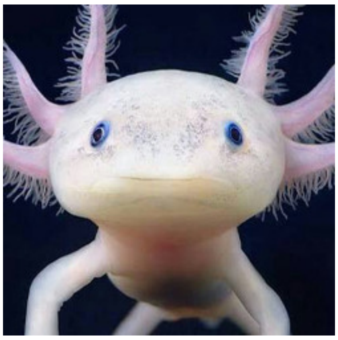

Figura 23: Imagem Original - (250x250)

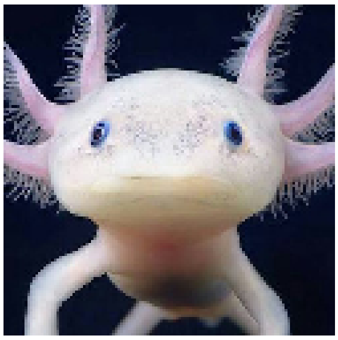

Figura 24: Compressão
Tamanho da imagem (125x125)

Figura 25: Descompressão bilinear
$h = 1$

Figura 26: Descompressão bicúbica
$h = 1$

TESTE: K = 3

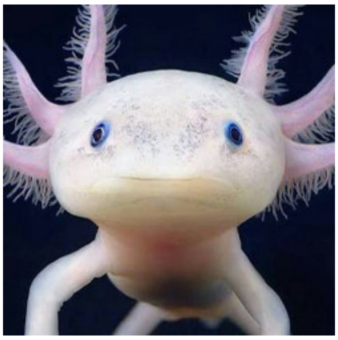

Figura 27: Imagem Original - 250x250

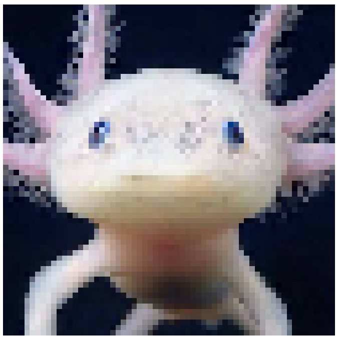

Figura 28: Compressão - 62x62

Figura 29: Descompressão bilinear
$h = 1$

Figura 30: Descompressão bicúbica
$h = 1$

TESTE: K = 5

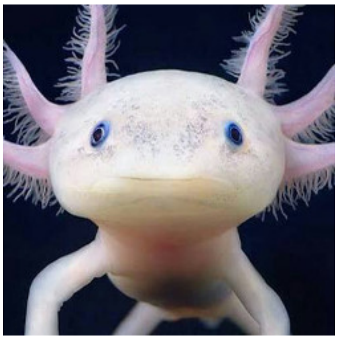

Figura 31: Imagem Original - 250x250

Figura 32: Compressão - 41x41

Figura 33: Descompressão bilinear
$h = 1$

Figura 34: Descompressão bicúbica
$h = 1$

Tabela de erros da descompressão

O k utilizado na compressão foi o mesmo utilizado na descompressão. O erro foi calculado usando a função **calculateError**.

A parte em negrito representa os valores que não são iguais para todos os valores de h analisados, o espaço antes da parte em negrito foi colocado para facilitar a leitura.

Interpolação Bilinear

	$h = 10$	$h = 1$	$h = 0.1$	$h = 0.01$
k = 1	0.035546 **96167**	0.035546 **93196**	0.035546 **66104**	0.035546 **42183**
k = 3	0.090053 **97170**	0.090053 **93155**	0.090053 **78828**	0.090053 **95422**
k = 5	0.13975 **06949**	0.13975 **06451**	0.13975 **05881**	0.13975 **05108**

Interpolação Bicúbica

	$h = 10$	$h = 1$	$h = 0.1$	$h = 0.01$
k = 1	0.03569 **513877**	0.03569 **513238**	0.03569 **511207**	0.03569 **509249**
k = 3	0.0916814 **2065**	0.0916814 **1715**	0.0916814 **1202**	0.0916814 **1756**
k = 5	0.14294285 **9286**	0.14294285 **7066**	0.14294285 **7138**	0.14294285 **4665**

Teste 2 - Imagem Taj Mahal

O mesmo K utilizado na compressão também foi utilizado na descompressão.

TESTE: K = 1

Figura 35: Imagem Original - 250x250

Figura 36: Compressão - 125x125

Figura 37: Descompressão bilinear
$h = 1$

Figura 38: Descompressão bicúbica
$h = 1$

TESTE: K = 3

Figura 39: Imagem Original - 250x250

Figura 40: Compressão - 62x62

Figura 41: Descompressão bilinear
$h = 1$

Figura 42: Descompressão bicúbica
$h = 1$

TESTE: K = 7

Figura 43: Imagem Original - 250x250

Figura 44: Compressão - 31x31

Figura 45: Descompressão bilinear
$h = 1$

Figura 46: Descompressão bicúbica
$h = 1$

Tabela de erros da descompressão

O k utilizado na compressão foi o mesmo utilizado na descompressão. O erro foi calculado usando a função **calculateError**.

Interpolação Bilinear

	$h = 10$	$h = 1$	$h = 0.1$	$h = 0.01$
k = 1	0.05358**201046**	0.05358**196424**	0.05358**195072**	0.05358**199677**
k = 3	0.11616**45655**	0.11616**45265**	0.11616**46859**	0.11616**47765**
k = 7	0.1490**999667**	0.1490**999418**	0.1491**000129**	0.1491**000167**

Interpolação Bicúbica

	$h = 10$	$h = 1$	$h = 0.1$	$h = 0.01$
k = 1	0.05393**708385**	0.05393**707952**	0.05393**708415**	0.05393**709194**
k = 3	0.12117**48954**	0.12117**48936**	0.12117**49052**	0.12117**49045**
k = 7	0.1578942**048**	0.1578942**055**	0.1578942**061**	0.1578942**115**

Conclusões

É perceptível que a decompressão também funciona bem para funções que não são de classe C^2, um dos problemas que ocorre é que a diferença entre a interpolação bilinear e bicúbica torna-se menor a maneira que a suposição de classe C^2 não é satisfeita, pois as derivadas parciais que antes ajudavam a 'suavizar' a imagem interpolada, não existem mais. Esse último fato foi percebido pelo erro da interpolação, no qual a interpolação bicúbica apresentou um erro semelhante e até maior em alguns casos, com relação a interpolação bilinear.

Teoricamente quanto menor o valor de h, menor deverá ser o erro da interpolação, no entanto nos testes foi percebido que algumas valores muito pequenos de h possuiam um erro semelhante a outros valores maiores de h. Uma das explicações para essa ocorrência é a limitação da variável double do Octave, pelo fato de sua capacidade de armazenamento ser de 16 bits, para valores muito pequenos de h, o tipo de dado utilizado torna-se incapaz de armazenar com precisão o valor do polinômio interpolador no ponto calculado, dessa forma, ao invés do erro diminuir, ele aumenta.

Excluindo os valores muito pequenos de h, foi notado que quando h diminui, o erro da interpolação também diminui, apresentando portanto uma relação direta.