Universidade de São Paulo
Instituto de Matemática e Estatística
MAC0210

# Relatório: Exercício-Programa 2

Alunos:
Gustavo Silva No USP: 9298260
Leonardo Padilha No USP: 9298295

Professor: Ernesto G. Birgin

# Índice

# 1 Objetivo

Nosso objetivo nesse Exercício-Programa é gerar um programa que comprima e descomprima uma dada imagem.

Para comprimir, basta remover alguns pixels da imagem, mais precisamente, dado um número real $k$, e considerando que a imagem é uma matriz de pixels, então removemos as linhas e as colunas $i$ tais que $i \equiv 1 \bmod (k+1)$.

Para fazer a descompressão, assumimos que a imagem é, basicamente, uma função que vai do $\mathbb{R}^2$ para o $\mathbb{R}^3$ ($\mathbb{R}^3$ pois estamos considerando a paleta RGB, logo, uma coordenada para cada cor da paleta) e, dessa forma, cada pixel vira um ponto no plano $\mathbb{R}^2$ (devemos considerar também que o espaçamento entre dois pixels adjacentes $h$ foi definido por nós). Agora, para inserirmos os novos pixels de coordenadas $(x, y) \in \mathbb{R}^2$ basta interpolarmos a função nessa região.

Para fazer a interpolação, usamos dois métodos diferentes que são extensões para $\mathbb{R}^2$ dos métodos vistos em sala de aula. Vamos explicar e avaliar os resultados obtidos com cada um deles.

# 2 Interpolação Bilinear por partes

Esse método é uma extensão para $\mathbb{R}^2$ da interpolação linear de uma função de $\mathbb{R}$ para $\mathbb{R}$.

Vamos considerar 4 pixels conhecidos adjacentes: $Q_0$, $Q_1$, $Q_2$ e $Q_3$. Esses pixels podem ser representados como pontos do plano $\mathbb{R}^2$ de coordenadas $(x_i, y_j), (x_{i+1}, y_j), (x_i, y_{j+1}), (x_{i+1}, y_{j+1})$, respectivamente onde $x_i, x_{i+1}, y_j, y_{j+1} \in \mathbb{R}$. Supondo que a imagem é uma amostra de uma função $f$ tal que $f : \mathbb{R}^2 \to \mathbb{R}$, então, para gerar um pixel que está em uma coordenada $(x_0, y_0)$ tal que $x_0 \in [x_i, x_{i+1}]$ e $y_0 \in [y_j, y_{j+1}]$, podemos criar uma função interpoladora v, que interpola os valores no intervalo, dada por:

$$v(x, y) = c_0 + c_1(x - x_i) + c_2(y - y_j) + c_3(x - x_i)(y - y_j) \tag{1}$$

Para interpolar no ponto $(x_0, y_0)$, basta encontrarmos $c_0$, $c_1$, $c_2$ e $c_3$. Como a distância de dois pixels adjacentes é definida por um número real $h$, ou seja, $x_i - x_{i+1} = h$, então, temos o seguinte sistema:

$$\begin{aligned}
&f(x_i, y_j) = c_0 \\
&f(x_i, y_{j+1}) = c_0 + hc_2 \\
&f(x_{i+1}, y_j) = c_0 + hc_1 \\
&f(x_{i+1}, y_{j+1}) = c_0 + hc_1 + h^2c_2 + h^3c_3
\end{aligned}$$

Fazendo as manipulações necessárias, chegamos que os coeficientes da função $v$

são:

$$c_0 = f(x_i, y_j)$$
$$c_1 = \frac{f(x_{i+1}, y_j) - f(x_i, y_j)}{h}$$
$$c_2 = \frac{f(x_i, y_{j+1}) - f(x_i, y_j)}{h}$$
$$c_3 = \frac{f(x_{i+1}, y_{j+1}) - f(x_i, y_j)}{h^3} - \frac{f(x_i, y_{j+1}) - f(x_i, y_j)}{h^2}$$

Assim, conseguimos calcular facilmente os pixels que serão inseridos entre quatro pixels adjacentes da imagem original, basta que apliquemos a função intepoladora para cada pixel que queremos inserir na imagem.

### 2.1 Resultados no Zoológico

### 2.2 Resultados na Selva

## 3 Interpolação Bicúbica por partes

A ideia desse método é um pouco mais sofisticado que o anterior, pois aqui exigimos que a função seja suave e que a função interpoladora também o seja em todo o domínio da interpolação.

Para entender o funcionamento, vamos considerar novamente os quatro pixels adjacentes $Q_0, Q_1, Q_2, Q_3$ com as mesmas coordenadas vistas anteriormente. Para interpolar um pixel que possui coordenadas $(x_0, y_0)$ tal que $x_0 \in [x_i, x_{i+1}]$ e $y_0 \in [y_j, y_{j+1}]$, vamos interpolar a função $f$ geradora da imagem por uma função $v$ que terá como condições de interpolação as seguintes propriedades:

$$\frac{\partial v}{\partial x}(x_i, y_j) = \frac{\partial f}{\partial x}(x_i, y_j), \quad \frac{\partial v}{\partial y}(x_i, y_j) = \frac{\partial f}{\partial x}(x_i, y_j), \quad \frac{\partial^2 v}{\partial xy}(x_i, y_j) = \frac{\partial^2 f}{\partial xy}(x_i, y_j)$$
$$\frac{\partial v}{\partial x}(x_{i+1}, y_j) = \frac{\partial f}{\partial x}(x_{i+1}, y_j), \quad \frac{\partial v}{\partial y}(x_{i+1}, y_j) = \frac{\partial f}{\partial x}(x_{i+1}, y_j), \quad \frac{\partial^2 v}{\partial xy}(x_{i+1}, y_j) = \frac{\partial^2 f}{\partial xy}(x_{i+1}, y_j)$$
$$\frac{\partial v}{\partial x}(x_i, y_{j+1}) = \frac{\partial f}{\partial x}(x_i, y_{j+1}), \quad \frac{\partial v}{\partial y}(x_i, y_{j+1}) = \frac{\partial f}{\partial x}(x_i, y_{j+1}), \quad \frac{\partial^2 v}{\partial xy}(x_i, y_{j+1}) = \frac{\partial^2 f}{\partial xy}(x_i, y_{j+1})$$
$$\frac{\partial v}{\partial x}(x_{i+1}, y_{j+1}) = \frac{\partial f}{\partial x}(x_{i+1}, y_{j+1}), \quad \frac{\partial v}{\partial y}(x_{i+1}, y_{j+1}) = \frac{\partial f}{\partial x}(x_{i+1}, y_{j+1}),$$
$$\frac{\partial^2 v}{\partial xy}(x_{i+1}, y_{j+1}) = \frac{\partial^2 f}{\partial xy}(x_{i+1}, y_{j+1})$$

Em nossa implementação, porém, não tinhamos acesso as derivadas da função geradora da imagem $f$, por isso, para obte-la, usamos fórmulas que permitem aproximar esses valores. Nas bordas da imagem, utilizamos uma aproximação que leva em conta apenas um pixel a frente, a taxa de erro sendo $\mathcal{O}(h)$. No meio da imagem, usamos a fórmula centrada, que permite um erro de $\mathcal{O}(h^2)$, ou seja, uma aproximação melhor que nas bordas, entretanto, ficamos limitados com o erro das bordas. Vale lembrar que quanto menor for $h$, melhor será nossa aproximação e, consequentemente, nossa interpolação. Devemos apenas ficar atentos para escolha de um $h$ que seja suficientemente pequeno e que continue mantendo o erro de aproximação do método dominando o erro de arredondamento. Para o nosso algoritmo, percorremos todos os pixels da imagem e verificamos onde deverá ser introduzido um novo pixel, e lá, fazemos a interpolação usando todas as fórmulas vistas aqui e pegando como $Q_0, Q_1, Q_2, Q_3$ os pixels conhecidos que estão mais próximos dele.

## 3.1 Resultados no Zoológico

## 3.2 Resultados na Selva