



**CENTRO UNIVERSITÁRIO IESB
ENGENHARIA DE COMPUTAÇÃO**

Diego Vieira Santos - 2012082035

Thiago Monteiro Nogueira - 2222082018

Rafael da Rocha Henrard - 1812082028

Laboratório 3

**BRASÍLIA, DF
2023**

0.1 Redimensionamento de imagens

O redimensionamento de imagens é uma técnica essencial em processamento de imagens e visão computacional, usada para alterar as dimensões físicas de uma imagem. Essa operação permite ajustar o tamanho ou proporções de uma imagem de acordo com as necessidades específicas de uma aplicação ou tarefa. Seja para reduzir o tamanho de uma imagem para economizar espaço de armazenamento, ajustar uma imagem para encaixar em uma interface de usuário, ou ampliar uma imagem para destacar detalhes, o redimensionamento desempenha um papel fundamental.

No processo de redimensionamento, a imagem original é submetida a uma transformação que altera o número de pixels ou o espaçamento entre eles. A operação pode ser realizada de forma uniforme, preservando a relação entre altura e largura, ou de forma não uniforme, o que pode distorcer a imagem. Existem várias técnicas de redimensionamento disponíveis, cada uma com seus próprios algoritmos e características. As técnicas mais comuns incluem a interpolação, que envolve a estimativa de novos valores de pixel com base nos pixels originais, e a reamostragem, que envolve a seleção de um subconjunto dos pixels originais para criar a nova imagem.

O redimensionamento de imagens é amplamente utilizado em uma variedade de aplicações, como visualização de imagens, processamento de vídeo, design gráfico, processamento de fotos e em muitos campos de pesquisa, incluindo visão computacional e aprendizado de máquina. A escolha da técnica de redimensionamento apropriada depende dos objetivos específicos da aplicação e das características da imagem, e é uma consideração fundamental para garantir a qualidade e a eficácia do processamento de imagens.

0.1.1 Redimensionando imagens com auxilio do MatLab

Carrega e mostra a imagem

```
x = imread("Lenna.jpg");
imshow(x, 'InitialMagnification',100);
```

Imagen Original



Converte a imagem para escala de cinza normalizada

```
x_cinza = double(rgb2gray(x))/255;  
imshow(x_cinza,'InitialMagnification',100);
```

Imagen original em escala de cinza



0.1.2 Compressão

Mostra a imagem com tamanho comprimido

```
y_comprimido = dil_or_com(x_cinza,"c");  
imshow(y_comprimido,'InitialMagnification',100);
```

Imagen comprimida por fator N = 2



0.1.3 Dilatação

Mostra a imagem com tamanho dilatado

```
y_dilatado = dil_or_com(x_cinza,"d");
imshow(y_dilatado, 'InitialMagnification',100);
```

Imagen dilatada por fator N = 2



0.2 Quantização de Imagens

A quantização de imagens é um processo no qual reduzimos o número de cores ou tons de cinza em uma imagem. Esse processo é comumente utilizado para comprimir imagens e reduzir o espaço de armazenamento necessário. Aqui estão os principais aspectos relacionados ao método de quantização de imagens:

- **Redução da Paleta de Cores:** Em imagens coloridas, a quantização envolve a redução do número de cores disponíveis na paleta de cores. Por exemplo, uma imagem colorida de 24 bits (16,7 milhões de cores) pode ser quantizada para uma paleta de 256 cores.
- **Tons de Cinza:** Em imagens em tons de cinza, a quantização envolve a redução do número de tons de cinza disponíveis. Por exemplo, uma imagem em tons de cinza de 256 níveis pode ser quantizada para 64 níveis de cinza.
- **Mapeamento de Cores:** O processo de quantização envolve o mapeamento de cores ou tons de cinza originais para as cores ou tons de cinza disponíveis na paleta reduzida. Isso geralmente é feito com base na correspondência de cores mais próximas ou valores de tons de cinza mais próximos na nova paleta.
- **Perda de Informação:** A quantização geralmente leva à perda de informação. Com menos cores ou tons de cinza disponíveis, a imagem resultante pode parecer menos detalhada e mais "achatada". A qualidade da imagem pode ser afetada, e a perda de detalhes sutis é inevitável.
- **Compactação de Dados:** Um dos principais motivos para a quantização de imagens é a redução no tamanho do arquivo. Isso é particularmente útil em aplicações onde o espaço de armazenamento é limitado, como na web, em dispositivos móveis e na transmissão de imagens.
- **Algoritmos de Quantização:** Existem vários algoritmos de quantização disponíveis, incluindo o algoritmo de Paleta Uniforme, onde as cores ou tons de cinza são distribuídos uniformemente na nova paleta, e o algoritmo de Difusão de Erro, que tenta distribuir o erro de quantização ao longo da imagem para reduzir o impacto visual da perda de detalhes.

Primeiro se define um valor de N e depois se utiliza o algoritmo de quantização para esse valor, exemplificaremos com valores de N iguais a 2,3,4,5,10 e 25.

0.2.1 Quantização para imagem sem excesso ou escassez de detalhes

```
x = imread("Lenna.jpg");
imshow(x);
```

Imagen Original



```
x = double(rgb2gray(x))/255;  
imshow(x);
```

Imagen em escala de cinza



Utilizando o algoritmo de quantização para gerar imagens com vários níveis de cinzas diferentes

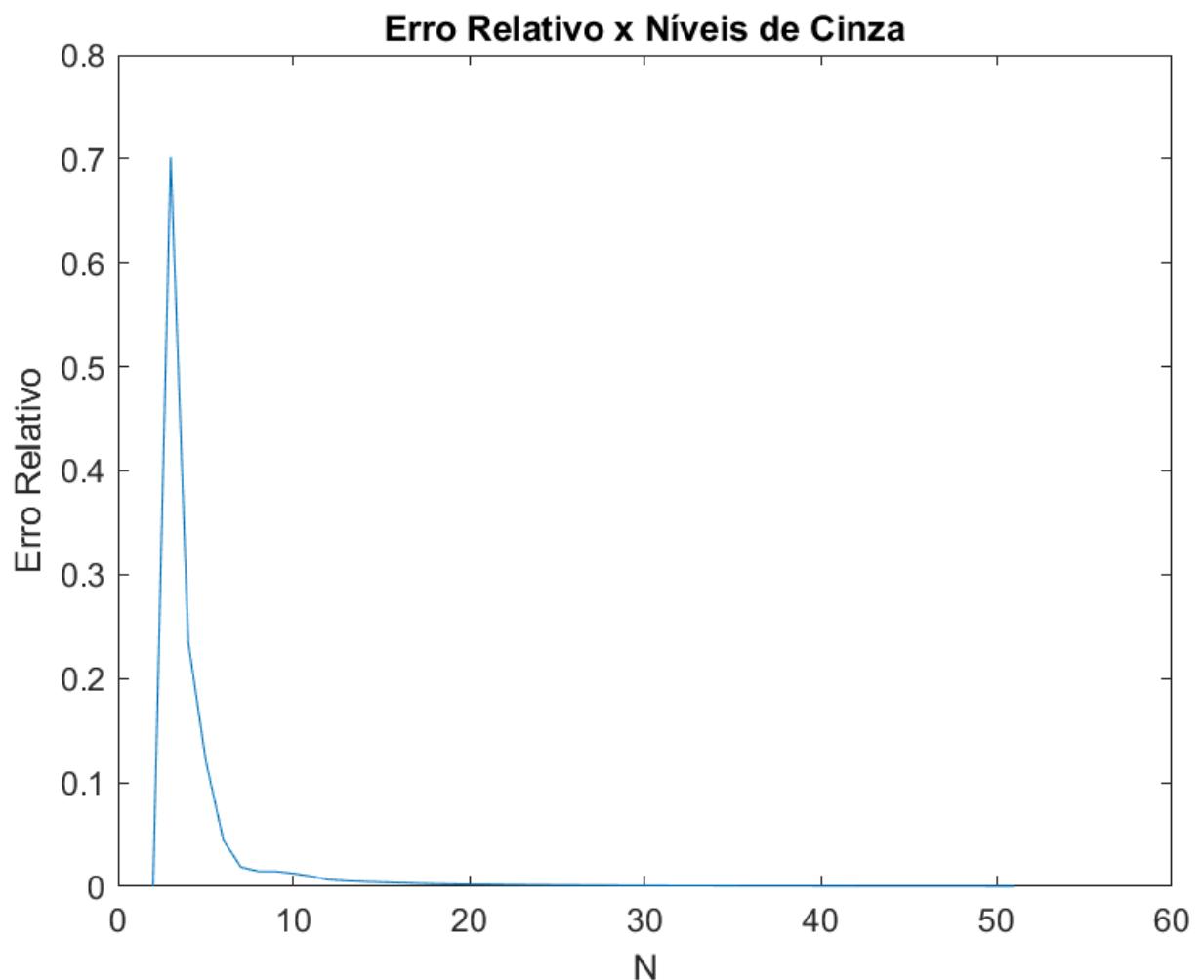
```
plot_quantizacao(x);
```



É possível perceber que a imagem vai perdendo "detalhe" sempre que diminui a quantidade de níveis de cinza, por exemplo com 2 níveis de cinza a imagem está em preto e branco, já com 25 ela está com uma quantidade considerada de níveis de cinza de forma que não se perde tanto detalhe e isso se aplica também aos outros 2 tipos de imagens testadas

Agora mostraremos o gráfico de erro relativo que relaciona a diferença entre a imagem de saída com a da entrada de acordo com o valor de N.

```
plot_erro_relativo(x);
```



0.2.2 Quantização para imagem com excesso de detalhes

```
x = imread("galaxy.jpeg");
imshow(x);
```

Imagen Original

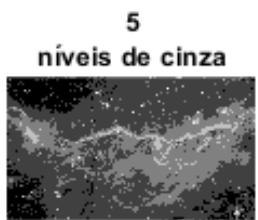


```
x = double(rgb2gray(x))/255;  
imshow(x);
```

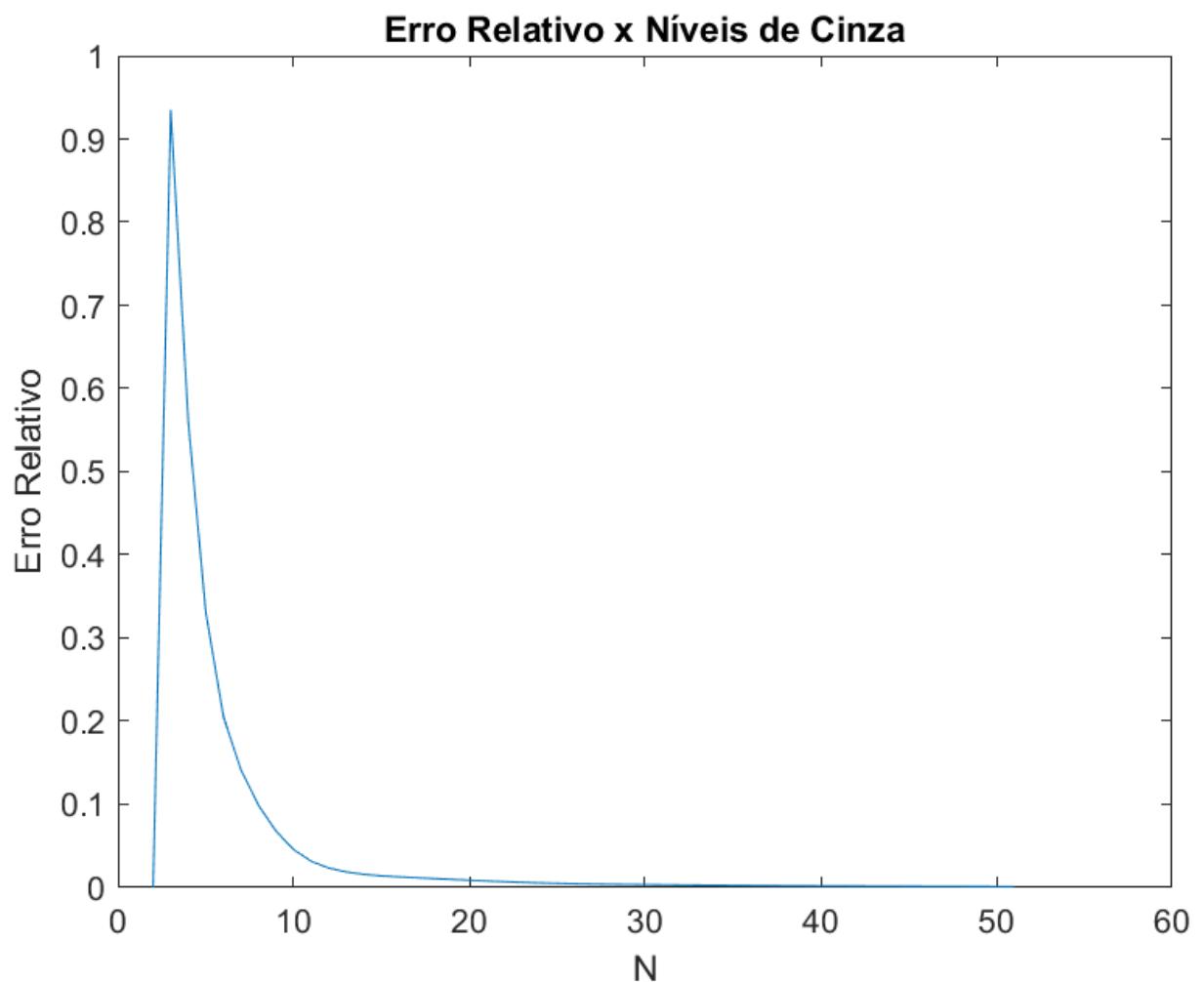
Imagen em escala de cinza



```
plot_quantizacao(x);
```



```
plot_erro_relativo(x);
```



0.2.3 Quantização para imagem com escassez de detalhes

```
x = imread("olho.jpg");
```

Imagen Original



```
x = double(rgb2gray(x))/255;  
imshow(x);
```

Imagen em escala de cinza



```
plot_quantizacao(x);
```

2
níveis de cinza



3
níveis de cinza



4
níveis de cinza



5
níveis de cinza



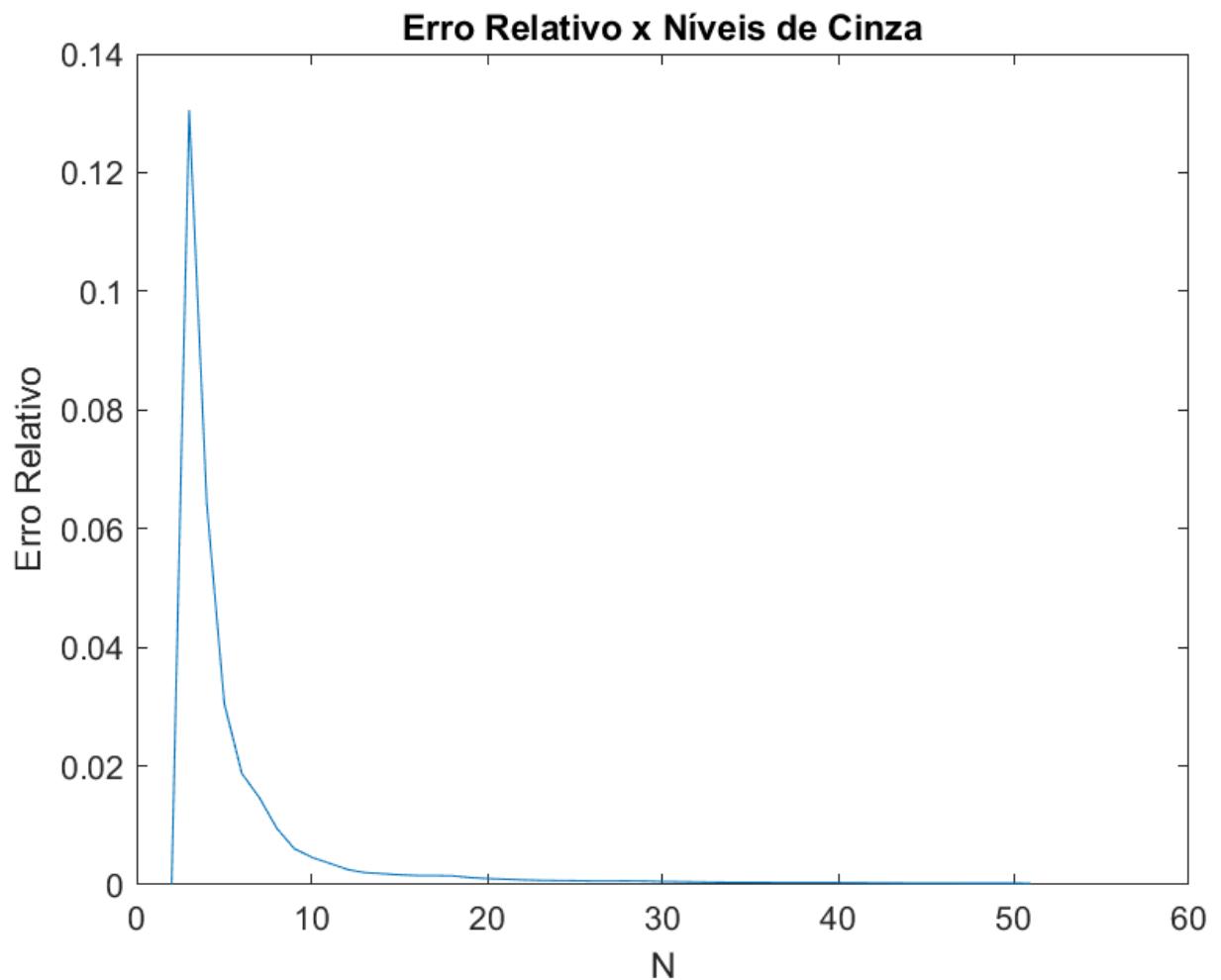
10
níveis de cinza



25
níveis de cinza



```
plot_erro_relativo(x);
```



Através dos testes de quantização utilizando varios tipos de niveis de cinza foi possivel notar que quanto maior a quantidade de niveis de cinza, mais nítida é a imagem e isso também implica que o erro relativo é menor para niveis de cinza suficientemente grandes, por exemplo: com N igual a 25 ja foi possivel observar que a imagem estava bem nítida, porém não tanto quanto a original pois ao fundo é possível perceber pequenos 'borrões'.

0.3 Funções Utilizadas

0.3.1 Função para dilatar ou comprimir uma imagem

Essa função recebe como entrada um arquivo de imagem em escala de cinza e um modo, sendo que o modo pode ser c ou d, representando compressão e dilatação respectivamente. Através de operações com matrizes a função dilata ou comprime a imagem e retorna a mesma como saída.

```
function y = dil_or_com (x,modo)
    [L,C] = size(x);
    if(modo == 'd')
        x_dil = zeros(L*2,C*2);
        for i = 1 : L
            for j = 1 : C
                atual = x(i,j);
                x_dil(i+i-1,j+j-1) = atual;
                x_dil(i+i-1+1,j+j-1) = atual;
                x_dil(i+i-1,j+j-1+1) = atual;
                x_dil(i+i-1+1,j+j-1+1) = atual;
            end
        end
        y = x_dil;
    else
        if (modo == 'c')
            [L,C] = size(x);
            x_com = zeros(round(L/2),round(C/2));
            [L,C] = size(x_com);
            for i = 1 : L
                for j = 1 : C
                    atual = x(i+i-1,j+j-1);
                    x_com(i,j) = atual; x_com(i+1,j) = atual;
                    x_com(i,j+1) = atual; x_com(i+1,j+1) = atual;
                end
            end
            x_com = x_com(1:end-1, 1:end-1);
            y = x_com;
        end
    end
end
```

0.3.2 Função para fazer quantização

Essa função recebe uma imagem em escala de cinza e um valor de N que é referente a quantidade de níveis de quantização e retorna como saída a imagem com os níveis desejados e seu erro relativo em relação a original. Ela utiliza o linspace do matlab para gerar a forma correta de valores para cada nível de N e também usa outra função que faz a interpolação da matriz com o valor mais próximo dos valores disponíveis de N, fazendo um "arredondamento" do valor do pixel.

```
function [y, erro_relativo] = quantizacao (x,N)
    niveis = linspace(0,1,N);
    y = interp1(niveis, niveis, x, 'nearest');
    [L,C] = size(x);
    num = zeros(L,C);
    den = num;
    for i = 1 : L
        for j = 1 : C
            num(i,j) = (abs(x(i,j) - y(i,j)))^2;
            den(i,j) = (abs(x(i,j)))^2;
        end
    end
    erro_relativo = num./(den+eps);
    erro_relativo = mean(erro_relativo(:));
end
```

0.3.3 Função para gerar o gráfico da relação de erro relativo da imagem para diferentes níveis de cinza

Essa função utiliza a função anterior com todos os valores possíveis de N para que seja possível gerar um gráfico os relacionando, o gráfico gerado só mostra parte dos valores testados pois a partir de um certo ponto fica imperceptível a diferença entre a imagem original e a quantizada.

```
function plot_erro_relativo(x)
    vet_N = 2:256;
    vet_erros = zeros(1,255);
    for N = 2 : length(vet_N)
        [y, vet_erros(N)] = quantizacao(x,N);
    end
    figure;
    plot(vet_N(1:50),vet_erros(1:50));
    xlabel('N');
    ylabel('Erro Relativo');
    title('Erro Relativo x Níveis de Cinza');
end
```

0.3.4 Função para gerar imagem com os 6 níveis de cinza

Essa função utiliza da função de quantização para gerar uma imagem unindo o resultado de quantização para 6 níveis de cinza diferentes, tornando assim mais fácil a visualização.

```
function plot_quantizacao(x)
    vet_N = [2 3 4 5 10 25];
    figure;
    for i = 1 : 6
        [y,z] = quantizacao(x,vet_N(i));
        subplot(2,3,i);
        imshow(y);
        title([int2str(vet_N(i)), "níveis de cinza"]);
    end
    figure;
end
```