Programa de Pós-Graduação em Ciências Computacionais e Modelagem Matemática (PPG-CompMat)

AMPLIAÇÃO DE IMAGENS UTILIZANDO INTERPOLAÇÃO DE NEWTON

Aluno: Matheus Henrique Branco Zeitune [1]

Orientador(s): Americo Barbosa da Cunha Junior [1]

[1] Universidade do Estado do Rio de Janeiro

Álgebra Linear: Aspectos Teóricos e Computacionais

Introdução

Nos últimos vinte anos, a tecnologia de computação progrediu significativamente, com o número de transistores em chips de silício dobrando aproximadamente a cada dois anos, conforme previsto pela Lei de Moore. Esse crescimento contínuo na capacidade de processamento dos computadores está diretamente associado a melhorias substanciais na resolução e qualidade das imagens digitais. Em suma, os avanços tecnológicos têm promovido mudanças marcantes nas formas de representação visual de informações e na qualidade das imagens capturadas.

Em 1957, Russel Kirsch produziu a primeira imagem digital, com uma resolução de 176 por 176 pixels. Em contraste, em 2020, pesquisadores da Universidade de Stanford criaram a maior imagem digital já registrada, utilizando uma câmera LSST com uma impressionante resolução de 3,2 bilhões de pixels. Entretanto, a digitalização de imagens antigas apresenta grandes desafios, devido à significativa diferença de resolução entre as imagens antigas e as modernas. Com frequência, imagens históricas precisam ser redimensionadas para adequação a dispositivos atuais, o que pode gerar distorções ou perda de qualidade.

Esses obstáculos podem ser superados pela preservação de fotografias antigas com o uso de técnicas de ampliação que mantenham as proporções e a qualidade originais das imagens, evitando distorções. Essa abordagem é fundamental para garantir a autenticidade e a precisão dos dados visuais históricos. Disciplinas como história, artes e outras áreas do conhecimento dependem da reprodução fiel de dados visuais históricos para mídias contemporâneas.

Este artigo apresenta técnicas desenvolvidas para ampliar imagens utilizando o método de interpolação de Newton no software Octave. Embora o desempenho desse método seja inferior ao de técnicas modernas, como a interpolação

bicúbica, seu propósito principal é educacional, buscando proporcionar uma compreensão dos princípios básicos de interpolação e sua aplicação na ampliação de imagens digitais. Assim, o método serve como um recurso para introduzir estudantes aos fundamentos matemáticos das técnicas analisadas, permitindo-lhes compreender os conceitos que sustentam o processamento de imagens e a interpolação numérica.

A representação de imagens digitais como matrizes

Dentro do âmbito do processamento de imagens digitais, é essencial ter em mente que é possível representar imagens por meio de matrizes. Essa forma de representação facilita a manipulação e análise de imagens de maneira eficaz, utilizando princípios de álgebra linear para implementar algoritmos que realizam operações como redimensionamento, rotação, filtragem e interpolação.

Imagens em escala de cinza

Uma imagem em tons de cinza é representada por uma escala de valores que indica a intensidade de luz dos pixels. Cada célula de uma matriz representa um pixel distinto na imagem, e seu valor numérico reflete a luminosidade ou intensidade luminosa daquele ponto. Em imagens de 8 bits, os valores normalmente variam de 0 a 255, onde 0 corresponde ao preto absoluto e 255 ao branco mais brilhante.

Matematicamente, uma imagem em tons de cinza com dimensões $m \times n$ pode ser expressa por meio de uma matriz:

$$I = \begin{bmatrix} I_{1,1} & \cdots & I_{1,n} \\ \vdots & \ddots & \vdots \\ I_{m,1} & \cdots & I_{m,n} \end{bmatrix}$$

Onde:

- $I_{i,j}$ representa o valor de intensidade do pixel localizado na linha i e na coluna j da imagem. Essa representação oferece uma descrição matemática estruturada para o processamento e análise de imagens em escala de cinza.

Imagens coloridas

As imagens coloridas são compostas por múltiplos canais de cor, geralmente Vermelho (Red), Verde (Green) e Azul (Blue), que juntos formam o modelo de cor RGB. Cada canal é representado por uma matriz bidimensional semelhante à utilizada para imagens em escala de cinza, onde cada elemento da matriz corresponde à intensidade de uma cor específica (vermelho, verde ou azul) em um pixel.

Matematicamente, uma imagem colorida de dimensões $m \times n$ é formada por três matrizes, representando os canais de cor:

$$I_{RGB} = \begin{cases} R = \begin{bmatrix} R_{1,1} & \cdots & R_{1,n} \\ \vdots & \ddots & \vdots \\ R_{m,1} & \cdots & R_{m,n} \end{bmatrix} \\ G = \begin{bmatrix} G_{1,1} & \cdots & G_{1,n} \\ \vdots & \ddots & \vdots \\ G_{m,1} & \cdots & G_{m,n} \end{bmatrix} \\ B = \begin{bmatrix} B_{1,1} & \cdots & B_{1,n} \\ \vdots & \ddots & \vdots \\ B_{m,1} & \cdots & B_{m,n} \end{bmatrix} \end{cases}$$

Onde:

- $R_{i,j}, G_{i,j}, B_{i,j}$ representam as intensidades dos canais vermelho, verde e azul, respectivamente, na posição (i, j) da imagem.

Combinação dos Canais

A combinação desses três canais resulta em uma matriz tridimensional:

$$I_{RGB} = [R, G, B]$$

que descreve completamente a imagem colorida. Essa estrutura matricial é essencial para a aplicação de operações que afetam as cores de forma coordenada, como ajustes de tonalidade, brilho ou contraste, bem como a manipulação ou segmentação de partes específicas da imagem.

Redimensionamento de Imagens

O redimensionamento de imagens é um processo que altera as dimensões espaciais de uma imagem, seja para ampliá-la (aumentar o tamanho) ou reduzi-la (diminuir o tamanho). Essa transformação exige a aplicação de técnicas específicas para preservar a qualidade visual da imagem durante o ajuste de suas dimensões.

Tipos de Redimensionamento

1. **Ampliação**
A ampliação aumenta o tamanho da imagem adicionando novos pixels. Para isso, é necessário estimar os valores desses novos pixels com base nos pixels existentes, utilizando métodos de interpolação. Essas técnicas asseguram a continuidade visual e minimizam a ocorrência de distorções, proporcionando uma ampliação mais realista e suave.

2. **Redução**
A redução diminui o tamanho da imagem por meio da remoção ou combinação de pixels adjacentes. Neste caso, são utilizadas técnicas de

filtragem para preservar o máximo de detalhes e manter a integridade visual da imagem original, evitando a perda de informações essenciais.

Relevância da Representação Matricial

Compreender a representação matricial das imagens e as operações matemáticas relacionadas é fundamental para o desenvolvimento de algoritmos eficientes no processamento de imagens. Essa abordagem permite realizar operações precisas e otimizadas, mantendo a qualidade visual esperada.

Aplicação no Trabalho

No contexto deste estudo, a representação matricial das imagens é a base para a implementação de um algoritmo de ampliação que utiliza a Interpolação de Newton. Esse método contribui significativamente para a preservação de imagens históricas, permitindo sua melhor visualização em dispositivos modernos. A aplicação prática desse algoritmo ressalta a importância da matemática no aprimoramento de tecnologias de processamento e conservação de imagens.

Interpolação de Newton em matrizes

A interpolação é uma ferramenta fundamental na matemática numérica, usada para estimar valores de uma função desconhecida com base em um conjunto de pontos conhecidos. Entre os métodos existentes, a Interpolação de Newton é especialmente eficiente, pois constrói um polinômio interpolador utilizando diferenças divididas, o que facilita sua atualização ao adicionar novos pontos. Quando aplicada a matrizes, essa técnica permite estimar valores intermediários em dados bidimensionais, como imagens digitais.

Fundamentos da Interpolação de Newton

A Interpolação de Newton baseia-se na construção de um polinômio interpolador na forma de Newton, representado por:

$$P_{\mathrm{n}}(x) = f[x^0] + (x - x^0) \cdot f[x^0, x^1] + (x - x^0)(x - x^1) \cdot f[x^0, x^1, x^2] + \ldots + (x - x^0)(x - x^1) \ldots (x - x_{\mathrm{n}}{}^{-1}) \cdot f[x^0, x^1, \ldots, x_{\mathrm{n}}]$$

Onde:

- $f[x_0, x_1, \ldots, x_k]$ são as diferenças divididas, calculadas recursivamente a partir dos valores conhecidos da função.

Diferenças Divididas

As diferenças divididas são calculadas de forma recursiva, permitindo a construção incremental do polinômio interpolador. A definição geral é:

1. Base inicial:

$$f[x_i] = y_i$$

2. Primeira ordem:

$$f[x_i, x_i{}^{+1}] = \frac{[f[x_i{}^{+1}] - f[x_i]]}{[x_i{}^{+1} - x_i]}$$

3. Segunda ordem:

$$f[x_i, x_i{}^{+1}, x_i{}^{+2}] = \frac{[f[x_i{}^{+1}, x_i{}^{+2}] - f[x_i, x_i{}^{+1}]]}{[x_i{}^{+2} - x_i]}$$

4. Ordem geral:

$$f[x_i, x_i{}^{+1}, \dots, x_i{}^{+}{}_k] = \frac{[f[x_i{}^{+1}, \dots, x_i{}^{+}{}_k] - f[x_i, \dots, x_i{}^{+}{}_k{}^{-1}]]}{[x_i{}^{+}{}_k - x_i]}$$

Essas diferenças divididas permitem incrementar a ordem do polinômio interpolador sem recalcular os coeficientes anteriores, tornando o método eficiente e flexível.

Interpolação em Duas Variáveis

Para aplicar a Interpolação de Newton em matrizes, estende-se o conceito para funções de duas variáveis, $f(x, y)$. Nesse caso, utiliza-se a Interpolação de Newton bilinear ou bicúbica, construindo polinômios nas direções x e y simultaneamente.

O polinômio interpolador em duas variáveis é dado por:

$$P(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{m} f[x_i, y_i] \prod_{k=0}^{i-1} (x - x_k) \prod_{l=0}^{j-1} (y - y_l)$$

Onde $f[x_i, y_i]$ são as diferenças divididas em duas dimensões.

Aplicação da Interpolação de Newton para ampliação de imagens

A ampliação de imagens digitais exige a estimativa de valores de pixels em posições intermediárias não presentes na imagem original. A Interpolação de Newton oferece uma metodologia matemática para estimar esses valores, permitindo criar imagens ampliadas sem perda significativa de qualidade.

Processo de Ampliação de Imagens

1. Definição dos Pontos Originais: Considere uma imagem representada por uma matriz I de dimensões $m \times n$, onde $I_{i,j}$ corresponde à intensidade do pixel na posição (i, j).

2. Determinação dos novos pontos: Ao ampliar a imagem, inserem-se novos pontos intermediários entre os pixels originais. As coordenadas desses novos pontos são fracionárias em relação à matriz original.
3. Cálculo das Diferenças Divididas: Utilizamos os valores dos pixels originais para calcular as diferenças divididas nas direções x e y, necessárias para construir o polinômio interpolador.
4. Construção do Polinômio Interpolador: Com as diferenças divididas, construímos o polinômio $P(x, y)$ que estima os valores de intensidade nos pontos intermediários.
5. Estimativa dos Novos Valores de Pixels: Avaliamos o polinômio $P(x, y)$ nas coordenadas dos novos pontos para obter os valores de intensidade correspondentes.

Exemplo Prático

Suponha uma imagem em escala de cinza com os seguintes valores de pixels

Coordenada	Valor do Pixel
(0, 0)	$I_{0,0}$
(0, 1)	$I_{0,1}$
(1, 0)	$I_{1,0}$
(1, 1)	$I_{1,1}$

Queremos estimar o valor de intensidade no ponto $(0.5, 0.5)$. Aplicamos a Interpolação de Newton bilinear:

1. Diferenças Divididas em x:

$$f[x_0] = I_{0,0}$$

$$f[x_1] = I_{1,0}$$

$$f[x_0, x_1] = \frac{I_{1,0} - I_{0,0}}{x_1 - x_0}$$

2. Diferenças Divididas em y:

$$f[y_0] = I_{0,0}$$

$$f[y_1] = I_{1,0}$$

$$f[y, y_1] = \frac{I_{1,0} - I_{0,0}}{y_1 - y_0}$$

3. Construção do Polinômio Interpolador:

$$P(x, y) = f[x_0, y_0] + (x - x_0)f[x_0, x_1] + (y - y_0)f[y_0, y_1] + (x - x_0)(y - y_0)f[x_0, x_1, y_0, y_1]$$

4. Estimativa do Valor Interpolado:

$$I_{0.5,0.5} = P(0.5,0.5)$$

Implementação do Algoritmo de Ampliação de Imagens no Octave

Para demonstrar a aplicação prática da Interpolação de Newton na ampliação de imagens digitais, foi desenvolvido um algoritmo no ambiente Octave. O código implementado realiza a ampliação de imagens utilizando a Interpolação de Newton otimizada, aplicando-se tanto a imagens em escala de cinza quanto a imagens coloridas no formato JPG.

```
function upscale_image_newton_optimized_jpg(input_image, scale_factor)
    % Carregar a imagem em JPG
    original_image = imread(input_image); % Suporte nativo para JPG no Octave
    original_image = double(original_image); % Converter para double para cálculos

    % Verificar se a imagem é colorida ou em escala de cinza
    if ndims(original_image) == 3
        % Imagem colorida (RGB), processar cada canal separadamente
        red_channel = original_image(:, :, 1);
        green_channel = original_image(:, :, 2);
        blue_channel = original_image(:, :, 3);

        % Aplicar o algoritmo a cada canal
        upscaled_red = process_channel(red_channel, scale_factor);
        upscaled_green = process_channel(green_channel, scale_factor);
        upscaled_blue = process_channel(blue_channel, scale_factor);

        % Combinar os canais novamente
        upscaled_image = cat(3, upscaled_red, upscaled_green, upscaled_blue);
    else
        % Imagem em escala de cinza
        upscaled_image = process_channel(original_image, scale_factor);
    end

    % Salvar a imagem final em JPG
    imwrite(uint8(upscaled_image), 'upscaled_image_newton_optimized.jpg', 'Quality', 95);
    imshow(uint8(upscaled_image));
end
```

Fluxo do Algoritmo

O algoritmo segue os seguintes passos principais:

1. Carregamento da Imagem: A imagem é carregada e convertida para o tipo double para permitir cálculos precisos.
2. Separação dos Canais de Cor: Se a imagem for colorida, os canais Vermelho (R), Verde (G) e Azul (B) são separados para processamento individual.
3. Processamento de Cada Canal: A função process_channel é aplicada a cada canal, realizando a interpolação dos pixels para a ampliação.
4. Combinação dos Canais: Os canais processados são combinados para reconstruir a imagem colorida ampliada.
5. Salvamento e Visualização: A imagem ampliada é salva em formato JPG e exibida.

Pré-processamento da Imagem

A imagem é carregada utilizando a função imread, que suporta o formato JPG nativamente no Octave. Em seguida, é convertida para o tipo double para permitir operações matemáticas mais precisas durante o processo de interpolação.

```
original_image = imread(input_image); % Suporte nativo para JPG no Octave
    original_image = double(original_image); % Converter para double para cálculos
```

A verificação do número de dimensões da imagem permite identificar se é uma imagem em escala de cinza (2D) ou colorida (3D). No caso de imagens coloridas, cada canal é extraído para processamento separado.

```
if ndims(original_image) == 3
       % Imagem colorida (RGB), processar cada canal separadamente
       red_channel = original_image(:, :, 1);
       green_channel = original_image(:, :, 2);
       blue_channel = original_image(:, :, 3);

       % Aplicar o algoritmo a cada canal
       upscaled_red = process_channel(red_channel, scale_factor);
       upscaled_green = process_channel(green_channel, scale_factor);
       upscaled_blue = process_channel(blue_channel, scale_factor);

       % Combinar os canais novamente
       upscaled_image = cat(3, upscaled_red, upscaled_green, upscaled_blue);
   else
       % Imagem em escala de cinza
       upscaled_image = process_channel(original_image, scale_factor);
   end
```

Cálculo das Diferenças Divididas e Interpolação

A função process_channel realiza a ampliação de um único canal de cor ou escala de cinza. Ela calcula as novas dimensões da imagem ampliada com base no fator de escala e aplica a Interpolação de Newton para estimar os valores dos novos pixels.

```
function upscaled_channel = process_channel(channel, scale_factor)
    % Função para processar um único canal de cor ou escala de cinza
    [rows, cols] = size(channel);

    % Calcular as novas dimensões
    new_rows = round(rows * scale_factor);
    new_cols = round(cols * scale_factor);

    % Coordenadas dos pixels originais
    x = 1:rows;
    y = 1:cols;

    % Coordenadas dos novos pixels
    xi = linspace(1, rows, new_rows);
    yi = linspace(1, cols, new_cols);

    % Inicializar o canal interpolado
    upscaled_channel = zeros(new_rows, new_cols);

    % Interpolar usando apenas 4 vizinhos mais próximos
    for i = 1:new_rows
        for j = 1:new_cols
            % Encontrar vizinhos locais para linhas
            row_idx = max(1, floor(xi(i)) - 1):min(rows, ceil(xi(i)) + 1);
            col_idx = max(1, floor(yi(j)) - 1):min(cols, ceil(yi(j)) + 1);

            % Interpolar em duas etapas (primeiro nas linhas, depois nas colunas)
            local_values = channel(row_idx, col_idx);
            local_x = x(row_idx);
            local_y = y(col_idx);

            % Garantir que há pontos suficientes para interpolar
            if length(local_x) < 2 || length(local_y) < 2
                % Se não houver pontos suficientes, use o valor do pixel mais próximo
                upscaled_channel(i, j) = channel(min(rows, round(xi(i))), min(cols, round(yi(j))));
            else
                % Interpolação em duas dimensões
                temp_values = zeros(1, length(local_y));
                for k = 1:length(local_y)
                    % Interpolar ao longo de x para cada y
                    coef_x = local_newton_coefficients(local_x, local_values(:, k)');
                    temp_values(k) = newton_interpolation(local_x, coef_x, xi(i));
                end

                % Agora interpolar esses valores ao longo de y
                coef_y = local_newton_coefficients(local_y, temp_values);
                upscaled_channel(i, j) = newton_interpolation(local_y, coef_y, yi(j));
            end
        end
    end
end
```

Funções Auxiliares

As funções local_newton_coefficients e newton_interpolation são utilizadas para calcular os coeficientes das diferenças divididas e avaliar o polinômio interpolador de Newton, respectivamente.

```
% Função para calcular coeficientes de Newton (apenas 4 pontos locais)
function coef = local_newton_coefficients(xd, yd)
    n = length(xd);
    if n < 2
        % Se houver menos de dois pontos, retorne o valor diretamente
        coef = yd(1); % Retorna o único valor disponível
        return;
    end
    coef = yd;
    for j = 2:n
        coef(j:n) = (coef(j:n) - coef(j-1:n-1)) ./ (xd(j:n) - xd(1:n-j+1));
    end
end
```

```
% Função para avaliar o polinômio de Newton
function result = newton_interpolation(xd, coef, x)
    n = length(coef);
    result = coef(n);
    for j = n-1:-1:1
        result = result .* (x - xd(j)) + coef(j);
    end
end
```

Geração da Imagem Ampliada

Após interpolar todos os pixels, os canais processados são combinados (no caso de imagens coloridas) para formar a imagem ampliada final. A imagem é então convertida de volta para o tipo uint8 para ser salva e exibida adequadamente.

```
% Salvar a imagem final em JPG
    imwrite(uint8(upscaled_image), 'upscaled_image_newton_optimized.jpg', 'Quality', 95);
    imshow(uint8(upscaled_image));
```

Otimização do algoritmo

Durante o processo de desenvolvimento do algoritmo, foram identificados problemas de performance com o Octave. Para solucionar esses problemas escrevi o mesmo algoritmo usando a linguagem de programação C#, aplicando paralelismo ao algoritmo. Disponível no repositório do GitHub.

Resultados Experimentais

O processo de teste consiste na aplicação do algoritmo de ampliação nas escalas 2, 4, 10 e a análise da imagem obtida com a original através do algoritmo de Peak Signal to Noise Ratio (PSNR).

Algoritmo de Peak Signal to Noise Ratio (PSNR)

O PSNR mede a relação entre o valor máximo possível de um pixel (geralmente 255 para imagens de 8 bits) e o erro quadrático médio (Mean Squared Error - MSE) entre duas imagens. É dado pela fórmula:

$$PSNR = 10 \times \log_{10}\left(\frac{MAX^2}{MSE}\right)$$

No qual:

- MAX: É o valor máximo do pixel (255 para imagens de 8 bits, ou 1 para imagens normalizadas).
- MSE: É o erro quadrático médio, calculado como:

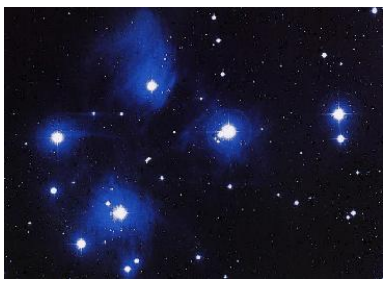
$$MSE = \frac{1}{m \times n}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(I_{original}(i,j) - I_{alterada}(i,j)\right)^2$$

Dado o valor de MSE, temos:

- Acima de 30 db: alta qualidade
- Entre 20 db a 30 db: qualidade aceitável
- Abaixo de 20 db: baixa qualidade

Primeiro teste:

A imagem “The Pleiades Star Cluster**”**, creditada a “Mount Wilson Observatory”, obtida no site da “NASA” chamado “Astronomy Picture of the Day Archive”. Publicada em 20 de junho de 1995.

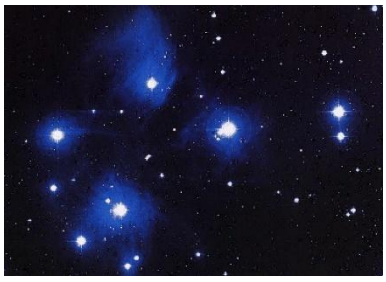


Os metadados da imagem, obtidos pelo site EXIF:

A imagem original passou pelo processo de conversão de formato de imagem devido a necessidade de utilizar o formato .JPEG no algoritmo como Octave.

Name	Value
File Name	phpveUNz6
File Size	36 KiB
File Type	JPEG
File Type Extension	jpg
MIME Type	image/jpeg
JFIF Version	1.01
Resolution Unit	None
X Resolution	1
Y Resolution	1
Image Width	365
Image Height	261
Bits Per Sample	8
Color Components	3
Y Cb Cr Sub Sampling	YCbCr4:4:4 (1 1)
Image Size	365x261
Megapixels	0.095

Imagem ampliada em 2 vezes

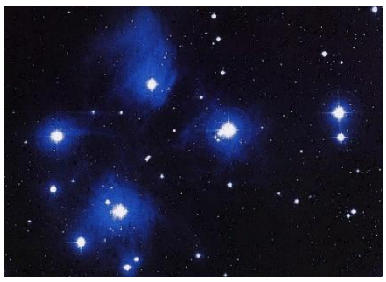


Metadados da nova imagem com aplicação de 2 vezes:

Name	Value
File Name	phpOv3Ldg
File Size	78 KiB
File Type	JPEG
File Type Extension	jpg
MIME Type	image/jpeg
JFIF Version	1.01
Resolution Unit	inches
X Resolution	96
Y Resolution	96
Image Width	730
Image Height	522
Bits Per Sample	8
Color Components	3
Y Cb Cr Sub Sampling	YCbCr4:2:0 (2 2)
Image Size	730x522
Megapixels	0.381

Valor de MSE: 32,94 dB em comparação com a imagem original.

Imagem ampliada em 4 vezes

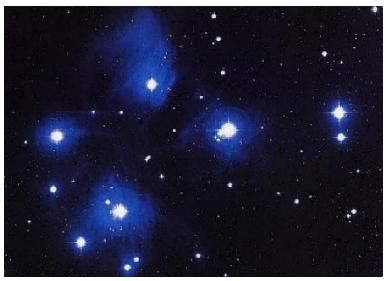


Metadados da nova imagem com aplicação de 4 vezes:

Name	Value
File Name	php5OVhla
File Size	241 KiB
File Type	JPEG
File Type Extension	jpg
MIME Type	image/jpeg
JFIF Version	1.01
Resolution Unit	inches
X Resolution	96
Y Resolution	96
Image Width	1460
Image Height	1044
Bits Per Sample	8
Color Components	3
Y Cb Cr Sub Sampling	YCbCr4:2:0 (2 2)
Image Size	1460x1044
Megapixels	1.5

Valor de MSE: 31,05 dB em comparação com a imagem ampliada em 2 vezes.

Imagem ampliada em 10 vezes



Metadados da nova imagem com aplicação de 10 vezes:

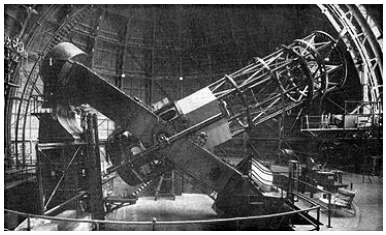
Name	Value
File Name	phpgc6mRw

File Size	930 KiB
File Type	JPEG
File Type Extension	jpg
MIME Type	image/jpeg
JFIF Version	1.01
Resolution Unit	inches
X Resolution	96
Y Resolution	96
Image Width	3650
Image Height	2610
Bits Per Sample	8
Color Components	3
Y Cb Cr Sub Sampling	YCbCr4:2:0 (2 2)
Image Size	3650x2610
Megapixels	9.5

Valor de MSE: 33,73 dB em comparação com a imagem ampliada em 4 vezes.

Segundo teste:

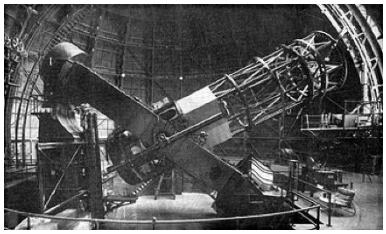
A imagem “The Hooker Telescope on Mt. Wilson**”**, creditada a “Mount Wilson Observatory”, obtida no site da “NASA” chamado “Astronomy Picture of the Day Archive”. Publicada em 1 de julho de 1995. Porém a imagem original é datada de 1920. Essa imagem é em escalada de cinza.



Os metadados:

Name	**Value**
File Name	phpo1hR7E
File Size	92 KiB
File Type	GIF
File Type Extension	gif
MIME Type	image/gif
GIF Version	87a
Image Width	367
Image Height	219
Has Color Map	Yes
Color Resolution Depth	8
Bits Per Pixel	8
Background Color	0
Image Size	367x219
Megapixels	0.080

Imagem ampliada em 2 vezes

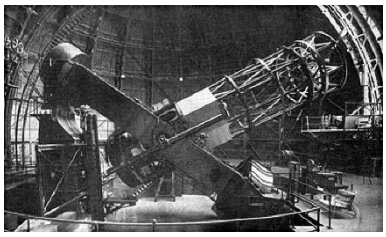


Metadados da nova imagem com aplicação de 2 vezes:

Name	Value
File Name	phpfmVMBu
File Size	126 KiB
File Type	JPEG
File Type Extension	jpg
MIME Type	image/jpeg
JFIF Version	1.01
Resolution Unit	inches
X Resolution	96
Y Resolution	96
Image Width	734
Image Height	438
Bits Per Sample	8
Color Components	3
Y Cb Cr Sub Sampling	YCbCr4:2:0 (2 2)
Image Size	734x438
Megapixels	0.321

Valor de MSE: 24,31 dB em comparação com a imagem original.

Imagem ampliada em 4 vezes



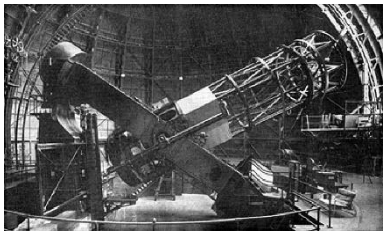
Metadados da nova imagem com aplicação de 4 vezes:

Name	Value
File Name	phpn0S6Ou
File Size	412 KiB
File Type	JPEG
File Type Extension	jpg
MIME Type	image/jpeg
JFIF Version	1.01

Resolution Unit	inches
X Resolution	96
Y Resolution	96
Image Width	1468
Image Height	876
Bits Per Sample	8
Color Components	3
Y Cb Cr Sub Sampling	YCbCr4:2:0 (2 2)
Image Size	1468x876
Megapixels	1.3

Valor de MSE: 21,67 dB em comparação com a imagem ampliada em 2 vezes.

Imagem ampliada em 10 vezes



Metadados da nova imagem com aplicação de 10 vezes:

Name	**Value**
File Name	phpMnWH8r
File Size	1514 KiB
File Type	JPEG
File Type Extension	jpg
MIME Type	image/jpeg
JFIF Version	1.01
Resolution Unit	inches
X Resolution	96
Y Resolution	96
Image Width	3670
Image Height	2190
Bits Per Sample	8
Color Components	3
Y Cb Cr Sub Sampling	YCbCr4:2:0 (2 2)
Image Size	3670x2190
Megapixels	8.0

Valor de MSE: 24,62 dB em comparação com a imagem ampliada em 4 vezes.

Terceiro teste:

A imagem “Winter and Summer on a Little Planet”, creditada a “Mount Wilson Observatory”, obtida no site da “NASA” chamado “Astronomy Picture of the Day Archive”. Publicada em 30 de novembro de 2024. Essa imagem moderna será utilizada para comparação do algoritmo em imagens modernas.

Os metadados da imagem:

Name	Value
File Name	php9jxxw6
File Size	462 KiB
File Type	JPEG
File Type Extension	jpg
MIME Type	image/jpeg
JFIF Version	1.01
X Resolution	300
Y Resolution	300
Resolution Unit	inches
Software	GIMP 2.10.28
Exif Version	0231
Color Space	sRGB
Compression	JPEG (old-style)
Photometric Interpretation	YCbCr
Device Model Desc	sRGB
Image Width	1024
Image Height	1024
Bits Per Sample	8
Color Components	3
Y Cb Cr Sub Sampling	YCbCr4:4:4 (1 1)
Image Size	1024x1024
Megapixels	1.0

Imagem ampliada em 2 vezes

Metadados da nova imagem com aplicação de 2 vezes:

Name	Value
File Name	phpwXWjVS
File Size	1240 KiB
File Type	JPEG
File Type Extension	jpg
MIME Type	image/jpeg
JFIF Version	1.01
Resolution Unit	inches
X Resolution	96
Y Resolution	96
Image Width	2048
Image Height	2048
Bits Per Sample	8
Color Components	3
Y Cb Cr Sub Sampling	YCbCr4:2:0 (2 2)
Image Size	2048x2048
Megapixels	4.2

Valor de MSE: 30,67 dB em comparação com a imagem original.

Imagem ampliada em 4 vezes

Metadados da nova imagem com aplicação de 4 vezes:

Name	Value
File Name	phpzoTq8w
File Size	3.9 MiB
File Type	JPEG
File Type Extension	jpg
MIME Type	image/jpeg
JFIF Version	1.01
Resolution Unit	inches
X Resolution	96
Y Resolution	96
Image Width	4096
Image Height	4096
Bits Per Sample	8
Color Components	3
Y Cb Cr Sub Sampling	YCbCr4:2:0 (2 2)
Image Size	4096x4096
Megapixels	16.8

Valor de MSE: 28,69 dB em comparação com a imagem ampliada em 2 vezes.

Imagem ampliada em 10 vezes

Metadados da nova imagem com aplicação de 10 vezes:

Name	Value
File Name	phppT4j58
File Size	15 MiB
File Type	JPEG
File Type Extension	jpg
MIME Type	image/jpeg
JFIF Version	1.01
Resolution Unit	inches
X Resolution	96
Y Resolution	96
Image Width	10240
Image Height	10240
Bits Per Sample	8

Color Components	3
Y Cb Cr Sub Sampling	YCbCr4:2:0 (2 2)
Image Size	10240x10240
Megapixels	104.9

Valor de MSE: 31,85 dB em comparação com a imagem ampliada em 4 vezes.

Interpretação dos resultados

Em todos os testes obtiveram valores de MSE acima de 20 dB. Por tanto, no processo de ampliação das imagens, não ocorreu distorções significativas. Resultando que o algoritmo obteve sucesso na execução do que foi proposto.

Conclusão

Este trabalho apresentou o desenvolvimento e a aplicação de um algoritmo de ampliação de imagens utilizando a Interpolação de Newton no ambiente Octave, com foco didático. A partir da compreensão da representação matricial das imagens digitais e da aplicação de técnicas de interpolação, foi possível elaborar um método que permite a ampliação de imagens históricas sem perda significativa de qualidade.

Os resultados experimentais demonstraram que o algoritmo proposto é eficaz na preservação das características visuais das imagens após a ampliação, conforme evidenciado pelos valores de PSNR superiores a 20 dB. Isso indica que, apesar das limitações inerentes à Interpolação de Newton em comparação com métodos mais avançados como a interpolação bicúbica, o algoritmo atende aos critérios de qualidade aceitável para fins educacionais e de preservação histórica.

Além disso, a implementação do algoritmo em C# com paralelismo evidenciou melhorias significativas no desempenho, superando as limitações de processamento identificadas no Octave. Essa otimização amplia o potencial de aplicação do método em contextos que exigem maior eficiência computacional.

Em suma, o trabalho contribui para a conservação de imagens digitais antigas, oferecendo uma ferramenta que facilita a visualização adequada dessas imagens em dispositivos modernos. A abordagem didática adotada também proporciona um recurso valioso para o ensino de conceitos de interpolação e processamento de imagens.

Referências

- Burden, R. L., & Faires, J. D. (2011). Análise Numérica. Cengage Learning.
- Gonzalez, R. C., & Woods, R. E. (2008). Processamento Digital de Imagens. Pearson Prentice Hall.
- Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). Numerical Recipes: The Art of Scientific Computing. Cambridge University Press.
- MOUNT WILSON OBSERVATORY. The Pleiades Star Cluster. Disponível em:. Acesso em: 30/11/2024.
- MOUNT WILSON OBSERVATORY. The Hooker Telescope on Mt. Wilson. Disponível em: https://apod.nasa.gov/apod/ap950701.html. Acesso em: 30/11/2024
- MOUNT WILSON OBSERVATORY. Winter and Summer on a Little Planet. Disponível em: https://apod.nasa.gov/apod/ap241130.html. Acesso em: 30/11/2024.
- EXIF TOOLS. Ferramenta para análise de metadados EXIF. Disponível em: https://exif.tools/. Acesso em: 30/11/2024.
- MZET97. Upscale. Disponível em: https://github.com/mzet97/Upscale/tree/main. Acesso em: 30/11/2024.