[MNOnL] Processamento de Imagem

Grupo 20

Rui Lopes **Francisco Ferreira** **Duarte Ribeiro**
Júlio Pinto **Daniel Pereira**

Sumário

1. Introdução 3
2. Definição do tema 3
3. Modelação do problema 3
4. Smoothing 3
   4.1. Algoritmo de Smoothing 4
   4.2. Resultado final Smoothing 9
5. Upscaling 10
   5.1. Algoritmo de Upscaling 10
   5.2. Resultado final Upscaling 14
6. ChatGPT 15
7. Conclusão 15
8. Código completo Smoothing 16
9. Código completo Upscaling 17
Bibliografia 18

1. Introdução

Este trabalho foi desenvolvido no âmbito da unidade curricular Métodos Numéricos e Otimização não Linear. Foi-nos proposta a exploração de um fenómeno do mundo real, onde se aplicassem conceitos lecionados. De facto, existem diversos fenómenos que fazem uso destes conceitos. Ainda mais foi nos incentivado o uso de ferramentas de inteligência artificial como o ChatGPT. Foi bastante interessante tomar conhecimento de alguns usos, presentes no nosso quotidiano, ainda que os mesmos não viessem a ser explorados durante este projeto.

2. Definição do tema

O nosso principal objetivo sempre foi abordar um tema que nos fosse próximo. Como estudantes de informática que somos, deparamo-nos com vários temas: computação gráfica (curvas de Bézier); machine learning (desenvolvimento de redes neuronais através de regressões) e processamento de sinais/imagens (filtragem e transformação dos mesmos). Optamos pelo último, já que um dos requisitos impostos foi o uso de pelo menos dois conceitos lecionados durante a unidade curricular. Este fenómeno faz uso de vários conceitos de métodos numéricos, tais como: integração numérica, interpolação numérica e sistemas de equações não-lineares. Para o desenvolvimento do projeto, decidimo-nos focar no uso de integração e interpolação numérica.

3. Modelação do problema

O processamento de imagem é uma área imensamente vasta - desde segmentação, passando por filtragem e indo até transformações geométricas. A verdade é que métodos numéricos constituem alta aplicabilidade aqui.

Decidimos optar por uma técnica de Smoothing, uma forma de filtragem, e pela técnica de Upscaling. O primeiro é um processo que visa reduzir o ruído e as imperfeições de uma imagem, tornando-a imagem mais suave e uniforme. Já o Upscaling consiste, como o nome indica, no aumento de resolução de uma imagem.

O nosso objetivo então passa por desenvolver dois *scripts* **MATLAB** que possam aplicar o processo de Smoothing e Upscaling a imagens. Ou seja, dada uma imagem, gerar uma nova imagem com os processamentos aplicados.

4. Smoothing

O processo de Smoothing, na prática, passa a imagem por um filtro, tal, como por exemplo, se passam sinais por filtros passa-baixo, de forma a *cortar* as frequências altas. Atualmente, existem vários filtros que são utilizados. Um dos mais utilizado é o *median filter*. Este consiste em dividir a imagem em várias porções e atribuir o resultado da mediana dos *pixels* ao *pixel* central da porção. Como é óbvio, existem algumas variações deste filtro, sendo uma delas o *mean area filter*, que consiste, de forma análoga, em atribuir o resultado da área média dos *pixels* ao *pixel* central da porção.
No desenvolvimento do nosso trabalho decidimos optar por uma implementação do segundo filtro. Este algoritmo usa como base o conceito de *patches*, pequenas áreas da imagem, para

calcular o valor de cada *pixel*. De maneira a calcular o valor de cada *pixel*, o algoritmo recorre a tópicos de integração, à regra do trapézio[1] em específico.

4.1. Algoritmo de Smoothing

Faremos agora a explicação passo a passo do algoritmo **MATLAB** de Smoothing.

Figura 1: Imagem original

Figura 2: Porção a ser analisada

Passo 1. Ler a imagem para uma matriz ($\mathcal{M}_{h,w,3}$) de tamanho h (altura da imagem) x w (comprimento da imagem) x 3 (cor RGB).

```
% Parameters
scale = 2; % Smoothing factor

% Input filename
filename = input('Enter the filename of the image: ', 's');

% Read the original image
originalImage = imread(filename);
```

Passo 2. Converter os valores h e w da matriz, que estão entre 0 e 255, para valores decimais entre 0.0 e 1.0.

```
originalImage = im2double(originalImage);
```

Passo 3. Inicializar a matriz resultado, toda a zeros, com o mesmo tamanho da matriz $\mathcal{M}$.

```
smoothedImage = zeros(size(originalImage));
```

Passo 4. Percorrer os três canais de cor (RGB).

```
for c = 1:size(originalImage, 3)
```

Passo 5. Para cada um dos canais, percorrer os valores das cores nas coordenadas i, j da imagem.

[1] A decisão do uso da regra do trapézio é explicada mais abaixo.

```
for i = 1:size(originalImage, 1)
    for j = 1:size(originalImage, 2)
```

Passo 6. Extrair o fragmento do canal de cor atual c para ser aplicado o SMOOTHING. Este fragmento é uma submatriz de $\mathcal{M}$ com as linhas do intervalo $[i - scale, i + scale]$ e com as colunas no intervalo $[j - scale, j + scale]$. São usadas as funções ***max*** e ***min*** para evitar que os fragmentos não vão para além das bordas da imagem[2].

```
patch = originalImage( ...
  max(1,i-scale):min(end,i+scale), ...
  max(1,j-scale):min(end,j+scale), ...
  c);
```

Como exemplo, faremos a aplicação do algoritmo (simplificado[3]) na porção a ser analisada mostrada na Figura 2.

0.24	0.42	0.20	0.19	0.18	0.45	0.25	0.21	0.17	0.55	0.25	0.20	0.23	0.15	0.52	0.47	0.27	0.20	0.23	0.22	0.21	0.27	0.29	0.42	0.63	0.67	0.62	0.56	0.47	0.38
0.16	0.19	0.20	0.26	0.32	0.17	0.20	0.31	0.51	0.50	0.56	0.28	0.21	0.34	0.49	0.23	0.22	0.21	0.21	0.20	0.19	0.32	0.38	0.47	0.75	0.55	0.62	0.53	0.44	0.62
0.17	0.18	0.29	0.54	0.63	0.53	0.22	0.20	0.29	0.19	0.21	0.24	0.18	0.18	0.37	0.18	0.37	0.29	0.20	0.22	0.29	0.30	0.14	0.41	0.62	0.57	0.56	0.64	0.51	0.38
0.18	0.48	0.25	0.22	0.18	0.37	0.28	0.42	0.54	0.21	0.24	0.20	0.17	0.38	0.21	0.34	0.52	0.28	0.15	0.37	0.29	0.33	0.30	0.45	0.68	0.67	0.57	0.55	0.61	0.55
0.56	0.36	0.40	0.24	0.13	0.46	0.34	0.40	0.21	0.20	0.19	0.22	0.19	0.47	0.21	0.43	0.40	0.25	0.18	0.53	0.35	0.40	0.21	0.16	0.46	0.59	0.53	0.58	0.40	0.54
0.26	0.38	0.47	0.22	0.21	0.22	0.24	0.24	0.20	0.22	0.21	0.19	0.23	0.23	0.22	0.22	0.23	0.22	0.18	0.47	0.27	0.23	0.15	0.17	0.51	0.62	0.54	0.51	0.54	0.69
0.13	0.24	0.26	0.35	0.23	0.20	0.47	0.24	0.20	0.29	0.35	0.22	0.14	0.19	0.22	0.22	0.20	0.23	0.12	0.24	0.23	0.19	0.18	0.15	0.35	0.57	0.47	0.61	0.59	0.64
0.22	0.35	0.29	0.43	0.34	0.53	0.60	0.25	0.19	0.26	0.44	0.25	0.19	0.34	0.50	0.25	0.23	0.22	0.22	0.22	0.24	0.23	0.20	0.16	0.32	0.54	0.72	0.63	0.61	0.56
0.23	0.45	0.24	0.19	0.18	0.15	0.22	0.18	0.53	0.28	0.19	0.20	0.55	0.29	0.22	0.21	0.37	0.23	0.32	0.55	0.60	0.27	0.27	0.29	0.29	0.66	0.58	0.52	0.48	0.65
0.17	0.16	0.18	0.62	0.37	0.21	0.22	0.23	0.16	0.22	0.20	0.20	0.20	0.35	0.23	0.36	0.22	0.18	0.46	0.26	0.24	0.17	0.29	0.39	0.16	0.60	0.59	0.45	0.50	0.64
0.20	0.31	0.19	0.51	0.47	0.16	0.19	0.19	0.35	0.25	0.23	0.23	0.09	0.53	0.23	0.47	0.29	0.24	0.19	0.26	0.32	0.41	0.22	0.19	0.17	0.51	0.69	0.61	0.57	0.54
0.20	0.17	0.19	0.25	0.54	0.51	0.57	0.25	0.22	0.20	0.18	0.56	0.23	0.37	0.35	0.57	0.27	0.55	0.42	0.26	0.18	0.22	0.17	0.53	0.24	0.36	0.68	0.54	0.56	0.56
0.18	0.18	0.29	0.50	0.47	0.24	0.17	0.16	0.24	0.18	0.33	0.50	0.20	0.40	0.50	0.20	0.30	0.16	0.45	0.28	0.25	0.25	0.25	0.22	0.22	0.14	0.54	0.69	0.58	0.69
0.21	0.14	0.17	0.54	0.24	0.18	0.31	0.50	0.22	0.19	0.16	0.33	0.53	0.25	0.25	0.34	0.35	0.23	0.15	0.38	0.37	0.65	0.55	0.44	0.25	0.16	0.47	0.75	0.68	0.62
0.14	0.30	0.17	0.14	0.16	0.17	0.20	0.20	0.22	0.18	0.22	0.18	0.22	0.31	0.53	0.45	0.27	0.21	0.22	0.49	0.22	0.48	0.43	0.49	0.27	0.19	0.45	0.53	0.70	0.62
0.15	0.18	0.20	0.15	0.17	0.15	0.21	0.37	0.20	0.24	0.24	0.22	0.23	0.22	0.20	0.20	0.17	0.20	0.20	0.21	0.40	0.21	0.17	0.22	0.23	0.35	0.49	0.27	0.60	0.58
0.40	0.16	0.23	0.43	0.50	0.24	0.24	0.39	0.29	0.20	0.21	0.20	0.21	0.22	0.33	0.48	0.26	0.21	0.22	0.23	0.48	0.29	0.25	0.23	0.23	0.15	0.18	0.17	0.38	0.71
0.37	0.58	0.29	0.42	0.60	0.60	0.19	0.20	0.28	0.58	0.26	0.20	0.19	0.22	0.20	0.16	0.20	0.24	0.19	0.22	0.22	0.29	0.51	0.31	0.23	0.47	0.29	0.20	0.16	0.55
0.19	0.21	0.15	0.28	0.41	0.31	0.51	0.30	0.34	0.21	0.20	0.16	0.20	0.32	0.60	0.54	0.24	0.17	0.53	0.25	0.33	0.50	0.62	0.25	0.22	0.24	0.20	0.38	0.44	0.48
0.16	0.18	0.54	0.42	0.20	0.19	0.47	0.20	0.17	0.55	0.24	0.32	0.47	0.63	0.49	0.49	0.22	0.18	0.25	0.20	0.22	0.25	0.23	0.55	0.23	0.55	0.48	0.23	0.19	0.40
0.35	0.16	0.16	0.44	0.21	0.20	0.23	0.20	0.20	0.22	0.21	0.33	0.51	0.27	0.23	0.20	0.16	0.19	0.24	0.29	0.37	0.30	0.24	0.21	0.38	0.44	0.27	0.18	0.37	0.40
0.44	0.19	0.22	0.20	0.18	0.39	0.27	0.27	0.24	0.20	0.19	0.19	0.22	0.18	0.21	0.37	0.56	0.24	0.22	0.35	0.43	0.27	0.25	0.20	0.22	0.61	0.64	0.27	0.40	0.40
0.17	0.16	0.18	0.24	0.16	0.72	0.27	0.21	0.20	0.22	0.23	0.20	0.22	0.20	0.20	0.29	0.40	0.30	0.36	0.25	0.23	0.25	0.24	0.29	0.29	0.24	0.25	0.24	0.21	0.27
0.16	0.15	0.19	0.15	0.53	0.49	0.24	0.19	0.21	0.25	0.24	0.22	0.21	0.27	0.33	0.32	0.55	0.56	0.45	0.26	0.27	0.25	0.17	0.25	0.23	0.23	0.22	0.24	0.25	0.19
0.20	0.29	0.54	0.39	0.25	0.33	0.24	0.22	0.19	0.19	0.38	0.25	0.16	0.25	0.42	0.21	0.41	0.49	0.23	0.54	0.35	0.29	0.38	0.35	0.54	0.32	0.24	0.25	0.25	0.20
0.18	0.17	0.24	0.42	0.27	0.40	0.24	0.17	0.15	0.39	0.47	0.26	0.24	0.22	0.20	0.20	0.22	0.21	0.23	0.29	0.37	0.33	0.43	0.41	0.56	0.28	0.22	0.35	0.38	0.20
0.18	0.17	0.22	0.36	0.31	0.53	0.25	0.20	0.20	0.51	0.20	0.22	0.20	0.22	0.18	0.25	0.24	0.21	0.29	0.36	0.33	0.25	0.25	0.25	0.46	0.29	0.29	0.42	0.43	0.26
0.40	0.56	0.40	0.22	0.21	0.23	0.22	0.27	0.35	0.26	0.38	0.28	0.36	0.21	0.20	0.35	0.56	0.43	0.29	0.25	0.23	0.25	0.21	0.49	0.27	0.21	0.24	0.27	0.22	0.33
0.42	0.58	0.25	0.18	0.19	0.19	0.19	0.27	0.42	0.42	0.41	0.28	0.35	0.23	0.56	0.33	0.63	0.22	0.28	0.33	0.23	0.27	0.37	0.38	0.52	0.33	0.27	0.26	0.27	0.23
0.18	0.16	0.19	0.23	0.22	0.22	0.20	0.19	0.19	0.63	0.29	0.22	0.23	0.20	0.19	0.24	0.20	0.22	0.21	0.31	0.35	0.51	0.43	0.28	0.31	0.27	0.23	0.22	0.34	0.38

Figura 3: Valores dos *pixels* da porção analisada

[2]O tamanho do *patch* será menor nessas situações.

[3]Para simplificação de demonstração, os três canais de cor foram substituídos por apenas um só, com valor brilho do *pixel* da imagem em preto e branco. Esse valor foi calculado tendo por base a norma NTSC e a respetiva fórmula da luminância, dada por $Y = 0.2126R + 0.7152G + 0.0722B$ e em que R, G e B é o valor de cada um dos canais coloridos do *pixel*.

Na execução do algoritmo simplificado, na porção, para $i = 3$ e $j = 3$, o fragmento *patch*[4] terá a seguinte matriz como valor:

$$\mathcal{P} = \begin{pmatrix} 0.24 & 0.42 & 0.20 & 0.19 & 0.18 \\ 0.16 & 0.19 & 0.20 & 0.26 & 0.32 \\ 0.17 & 0.18 & \mathbf{0.29} & 0.54 & 0.63 \\ 0.18 & 0.48 & 0.25 & 0.22 & 0.18 \\ 0.56 & 0.36 & 0.40 & 0.24 & 0.13 \end{pmatrix}$$

Passo 7. Aplicar o SMOOTHING na matriz *patch*. A função ***trapz*** é usada para integrar ao longo das linhas (1) e depois ao longo das colunas (2). O resultado é divido pelo número de elementos na matriz para calcular a média.

```
smoothedPatch = trapz(trapz(patch, 1), 2) / numel(patch);
```

Analisando a linha acima passo a passo:

```
trapz(patch, 1)
```

A chamada a esta função aplica o método do trapézio ao longo das linhas da matriz. Desta forma, a matriz resultado desta função, $\mathcal{T}_1$, será calculada da seguinte forma:

$$\mathcal{T}_1 = \begin{pmatrix} trapz\begin{pmatrix}0.24\\0.16\\0.17\\0.18\\0.56\end{pmatrix} & trapz\begin{pmatrix}0.42\\0.19\\0.18\\0.48\\0.36\end{pmatrix} & trapz\begin{pmatrix}0.20\\0.20\\0.29\\0.25\\0.40\end{pmatrix} & trapz\begin{pmatrix}0.19\\0.26\\0.54\\0.22\\0.24\end{pmatrix} & trapz\begin{pmatrix}0.18\\0.32\\0.63\\0.18\\0.13\end{pmatrix} \end{pmatrix}$$

$$= (0.9100 \;\; 1.2400 \;\; 1.0400 \;\; 1.2350 \;\; 1.2850)$$

O resultado da chamada da função acima é agora passado novamente para a função ***trapz***, mas agora operando ao longo das colunas:

```
trapz(trapz(patch, 1), 2)
```

Desta forma, a matriz resultado da chamada desta função, $\mathcal{T}_2$, será calculada assim:

$$\mathcal{T}_2 = trapz(\mathcal{T}_1) = trapz(0.9100 \;\; 1.2400 \;\; 1.0400 \;\; 1.2350 \;\; 1.2850) = 4.6125$$

Agora, para calcular a área média, dividindo pelo número de elementos da matriz:

```
trapz(trapz(patch, 1), 2) / numel(patch)
```

$$\mathcal{S} = \frac{\mathcal{T}_2}{n} = \frac{4.6125}{25} = 0.1845$$

Este resultado vai ser o valor final do canal de cor atual que o *pixel* atual irá ter.

[4] A varíavel *scale* está com valor 2, logo a nossa matriz terá de tamanho 5x5 $(2 + 1 + 2)$ para coordenadas longe das bordas da imagem.

Passo 8. Gravar o resultado $\mathcal{S}$ na matriz resultado nas posições e cor de canal atual.

```
            smoothedImage(i, j, c) = smoothedPatch;
```

Passo 9. Aplicar o mesmo algoritmo para os restantes *pixels*.

```
for c = 1:size(originalImage, 3)
    for i = 1:size(originalImage, 1)
        for j = 1:size(originalImage, 2)
            patch = originalImage( ...
              max(1,i-scale):min(end,i+scale), ...
              max(1,j-scale):min(end,j+scale), ...
              c);

            smoothedPatch = trapz(trapz(patch, 1), 2) / numel(patch);
            smoothedImage(i, j, c) = smoothedPatch;
        end
    end
end
```

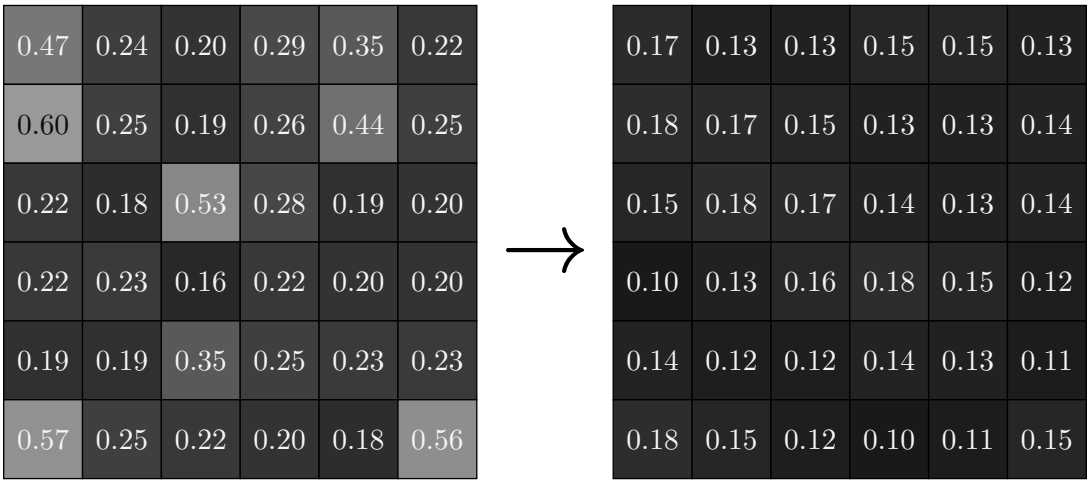

Figura 4: Execução do algoritmo numa pequena porção da imagem

Passo 10. Mostrar imagem original e resultado no MATLAB.

```
% Display the original and smoothed images
figure;
subplot(1, 4, 1);
imshow(originalImage);
title('Original Image');
subplot(1, 4, 2);
imshow(smoothedImage);
title('Smoothed Image');
```

Passo 11. Guardar a imagem num ficheiro.

```
% Save the smoothed image
smoothedFilename = 'smoothed_image.jpg';
```

A aplicação do algoritmo na porção de *pixels* analisada pode ser vista na Figura 5:

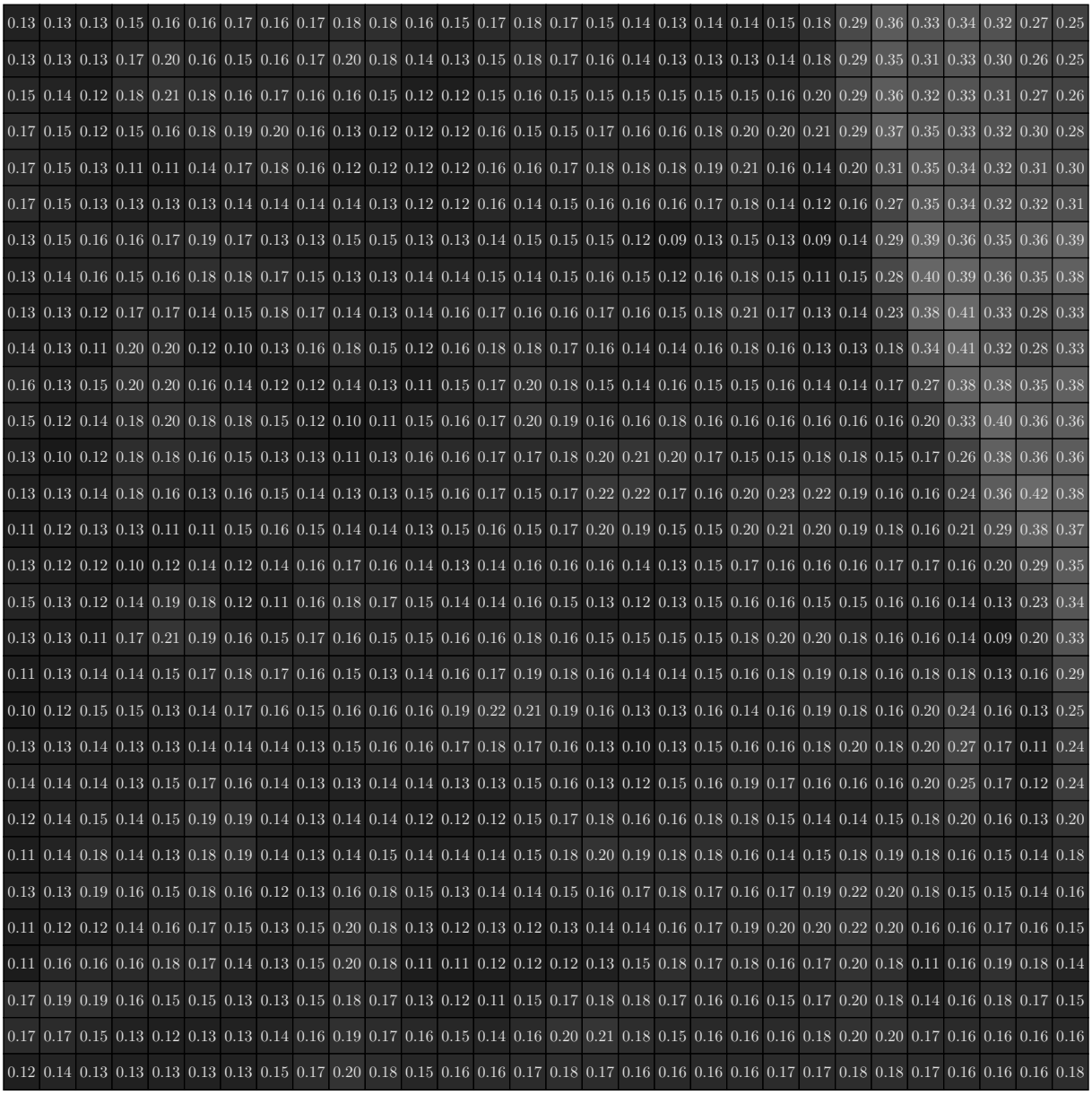

Figura 5: Execução do algoritmo na porção da imagem

Como é possível ver, a porção depois do algoritmo ser aplicado está muito mais suave.

Isto deve-se ao facto de fazermos uma integração numérica nos *patches* extraídos da imagem. Com isto, estimamos essa área média sob a curva representada pelos valores de cada *patch* usando a **regra do trapézio**.

Consideramos que este algoritmo seja o mais adequado, devido à natureza das imagens em que o estamos a aplicar. Entre pontos há muito ruído e não são aproximáveis por uma função suave em que, por exemplo, o algoritmo de Simpson seria mais apropriado.

4.2. Resultado final Smoothing

Figura 1: Imagem original

Figura 6: Imagem suavizada final

5. Upscaling

O processo de Upscaling tem por base a **interpolação numérica**, que é utilizada para estimar o valor dos *pixels* nas novas posições entre *pixels* da imagem redimensionada.

Existem vários métodos utilizados atualmente, os mais conhecidos sendo: o **método do "vizinho mais próximo"**, o **método bilinear** e o **método bicúbico**:

- O primeiro, o **método do vizinho mais próximo**, é o mais simples de todo e consiste simplesmente em atribuir o valor do *pixel* mais próximo ao *pixel* atual. É, portanto, facilmente dedutível que este método irá introduzir bastantes artefactos na imagem.

O segundo e terceiro métodos já apresentam resultados melhores em grande parte das imagens, mas requerem um poder computacional superior.

- O **método bilinear** utiliza os 4 pontos (2x2) mais próximos ao *pixel* que irá ser interpolado, define um polinómio quadrático e estima o valor final do *pixel*.
- O **método bicúbico** faz o mesmo, com a exceção de utilizar 16 *pixels* (4x4), definindo assim um polinómio cúbico, dando aso a melhores resultados.

Decidimos optar por utilizar o **método bicúbico** para o nosso algoritmo de Upscaling.

5.1. Algoritmo de Upscaling

Faremos agora a explicação passo a passo do algoritmo **MATLAB** de Upscaling.

Passo 1. Tal como no Smoothing, ler a imagem para uma matriz ($\mathcal{M}_{h,w,3}$) de tamanho h (altura da imagem) x w (comprimento da imagem) x 3 (cor RGB).

```
% Parameters
scale = 2; % Resizing factor

% Input filename
filename = input('Enter the filename of the image: ', 's');

% Read the original image
originalImage = imread(filename);
```

Passo 2. Converter os valores h e w da matriz, que estão entre 0 e 255, para valores decimais entre 0.0 e 1.0.

```
originalImage = im2double(originalImage);
```

Passo 3. Invocar a rotina ***imresize*** indicando que a interpolação desejada é `bicubic`.

```
upscaledImage = imresize(originalImage, scale, 'bicubic');
```

Esta interpolação nada mais é que a cúbica, mas aplicada a duas dimensões.

Faremos agora a explicação de como funciona a chamada à rotina **IMRESIZE**. Na demonstração do exemplo, em semelhança ao exemplo do SMOOTHING, as cores RGB também foram convertidas para um único canal de brilho.

Como exemplo, levaremos em conta a primeira submatriz 4x4 (chamemos-lhe de $\mathcal{R}$) da Figura 3:

0.24	0.42	0.20	0.19
0.16	0.19	0.20	0.26
0.17	0.18	0.29	0.54
0.18	0.48	0.25	0.22

Figura 7: Valores da matriz $\mathcal{R}$

Podemos representar esta matriz num gráfico de dispersão tridimensional, sendo que cada coluna $\boldsymbol{i}$ e cada linha $\boldsymbol{j}$ é transformado num ponto com coordenadas $(i, j, \mathcal{M}_{ij})$. O valor da cor, $\mathcal{M}_{ij}$, foi também redimensionado de $[0, 1]$ para $[0, 255]$ para melhor visualização.

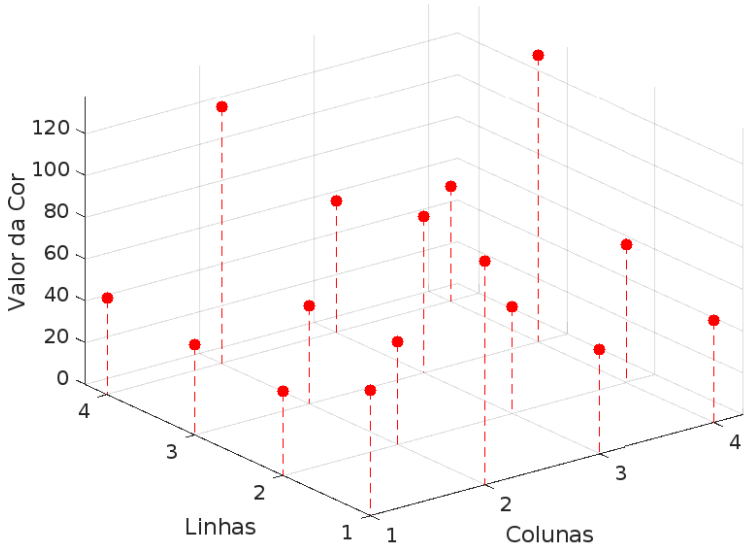

Figura 8: Matriz $\mathcal{R}$ representada num gráfico de dispersão 3D

Agora, com estes 16 pontos, é possível gerar uma superfície interpoladora desses pontos. O cálculo desta superfície pode ser feita com recurso a, por exemplo, *splines* bicúbicos.

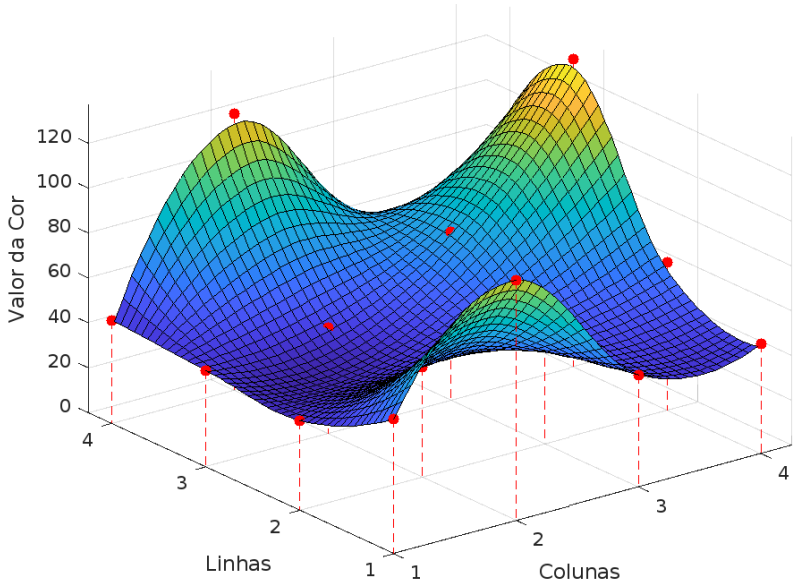

Figura 9: Representação da função interpoladora bicúbica da matriz $\mathcal{R}$

Com isto, os valores dos *pixels* da nova imagem redimensionada serão simplesmente amostras desta superfície. Como queremos redimensionar a imagem para duas vezes o seu tamanho ($scale = 2$), pegaremos em $8 * 8 = 64$ amostras, separadas uniformemente.

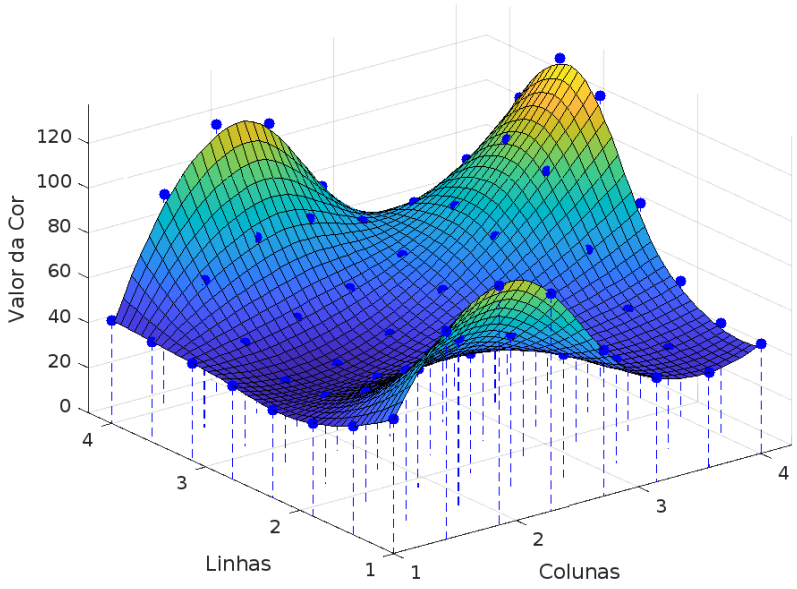

Figura 10: Representação das amostras na função interpoladora

Desta forma, temos os valores dos *pixels* da submatriz da imagem redimensionada.

0.24	0.42	0.20	0.19
0.16	0.19	0.20	0.26
0.17	0.18	0.29	0.54
0.18	0.48	0.25	0.22

Figura 7: Valores da matriz $\mathcal{R}$

0.24	0.36	0.42	0.38	0.25	0.18	0.17	0.19
0.19	0.26	0.29	0.27	0.22	0.18	0.18	0.20
0.16	0.19	0.21	0.21	0.20	0.20	0.21	0.24
0.15	0.15	0.16	0.18	0.20	0.24	0.28	0.34
0.16	0.15	0.15	0.18	0.23	0.30	0.39	0.49
0.17	0.18	0.20	0.22	0.26	0.32	0.41	0.53
0.18	0.26	0.31	0.31	0.29	0.29	0.35	0.43
0.18	0.38	0.47	0.45	0.31	0.23	0.20	0.22

Figura 11: Valores da matriz $\mathcal{R}$ após UPSCALING

Agora o mesmo procedimento é aplicado no resto da imagem.

Para aplicação do algoritmo numa imagem, a interpolação bicúbica em uma matriz de tamanho arbitrário pode ser realizada juntando as várias superfícies bicúbicas 4x4, fazendo com que as derivadas coincidam entre as bordas dessas.

Passo 4. Mostrar imagem original e resultado no MATLAB.

```
% Display the original and upscaled images
figure;
subplot(1, 4, 1);
imshow(originalImage);
title('Original Image');
subplot(1, 4, 2);
imshow(upscaledImage);
title('Upscaled Image');
```

Passo 5. Guardar a imagem num ficheiro.

```
upscaledFilename = 'upscaled_image.jpg';
imwrite(upscaledImage, upscaledFilename);
```

5.2. Resultado final Upscaling

Figura 12: Imagem original (216 x 228)

Figura 13: Imagem redimensionada com `scale = 2` (432 x 456)[5]

[5]O tamanho da imagem presente visualmente neste ficheiro é meramente representativo. O verdadeiro tamanho da imagem está entre parênteses.

6. ChatGPT

Ao longo do desenvolvimento do nosso projeto, recorremos diversas vezes ao **ChatGPT** como uma segunda ajuda. Por exemplo, para a decisão do tema.

Recorremos, também, ao **ChatGPT** como suporte no desenvolvimento do código MATLAB. Utilizámo-lo não só para escrever o código, como também para perceber como certas rotinas funcionavam e corrigir algumas imperfeições cometidas por ele mesmo. No entanto, o facto de existirem imperfeições, fez com que tivéssemos de verificar factualmente se o que o **ChatGPT** referia era totalmente correto e, para isso, utilizamos fontes seguras como a documentação do MATLAB, exemplos de aplicação e artigos científicos.

Acima de tudo, no que toca a exploração de ideias, o **ChatGPT** é extremamente útil. Este expôs situações e temas que não teríamos pensado só com uma pesquisa habitual no Google.

7. Conclusão

De um ponto de vista geral, consideramos que este projeto foi bastante gratificante. Falar sobre um tema do nosso interesse, poder abordar um projeto de uma maneira totalmente diferente e sermos incentivados a utilizar o **ChatGPT** foi muito benéfico para o desenvolvimento deste trabalho. Acreditamos que conseguimos desenvolver um projeto interessante, tendo em conta a nossa área e tendo em conta o contexto da unidade curricular.

8. Código completo Smoothing

```
% Parameters
scale = 2; % Smoothing factor

% Input filename
filename = input('Enter the filename of the image: ', 's');

% Read the original image
originalImage = imread(filename);
originalImage = im2double(originalImage);

smoothedImage = zeros(size(originalImage));

for c = 1:size(originalImage, 3)
    for i = 1:size(originalImage, 1)
        for j = 1:size(originalImage, 2)
            patch = originalImage( ...
              max(1,i-scale):min(end,i+scale), ...
              max(1,j-scale):min(end,j+scale), ...
              c);

            smoothedPatch = trapz(trapz(patch, 1), 2) / numel(patch);
            smoothedImage(i, j, c) = smoothedPatch;
        end
    end
end

% Display the original and smoothed images
figure;
subplot(1, 4, 1);
imshow(originalImage);
title('Original Image');
subplot(1, 4, 2);
imshow(smoothedImage);
title('Smoothed Image');

% Save the smoothed image
smoothedFilename = 'smoothed_image.jpg';
imwrite(smoothedImage, smoothedFilename);

% Display the sizes of the images
originalSize = size(originalImage);
smoothedSize = size(smoothedImage);

fprintf('Original Image Size: %d x %d\n', originalSize(1), originalSize(2));
fprintf('Smoothed Image Size: %d x %d\n', smoothedSize(1), smoothedSize(2));)
```

9. Código completo Upscaling

```
% Parameters
scale = 2; % Resizing factor

% Input filename
filename = input('Enter the filename of the image: ', 's');

% Read the original image
originalImage = imread(filename);
originalImage = im2double(originalImage);

% Upscaling using bicubic interpolation
upscaledImage = imresize(originalImage, scale, 'bicubic');

% Display the original and upscaled images
figure;
subplot(1, 4, 1);
imshow(originalImage);
title('Original Image');
subplot(1, 4, 2);
imshow(upscaledImage);
title('Upscaled Image');

% Save the upscaled image
upscaledFilename = 'upscaled_image.jpg';
imwrite(upscaledImage, upscaledFilename);

% Display the sizes of the images
originalSize = size(originalImage);
upscaledSize = size(upscaledImage);

fprintf('Original Image Size: %d x %d\n', originalSize(1), originalSize(2));
fprintf('Upscaled Image Size: %d x %d\n', upscaledSize(1), upscaledSize(2));)
```

Bibliografia

[**1**] Rotina max MATLAB, https://www.mathworks.com/help/matlab/ref/max.html, consultado em 24/06/2023.

[**2**] Rotina min MATLAB, https://www.mathworks.com/help/matlab/ref/min.html, consultado em 24/06/2023.

[**3**] Rotina trapz MATLAB, https://www.mathworks.com/help/matlab/ref/trapz.html, consultado em 24/06/2023.

[**4**] Rotina imresize MATLAB, https://www.mathworks.com/help/matlab/ref/imresize.html, consultado em 24/06/2023.

[**5**] Trapezoidal rule - Wikipedia, https://en.wikipedia.org/wiki/Trapezoidal_rule, consultado em 24/06/2023.

[**6**] Bicubic interpolation - Wikipedia, https://en.wikipedia.org/wiki/Bicubic_interpolation, consultado em 24/06/2023.