

Proyecto 1. Procesado de imágenes con MATLAB
Máster Universitario en Ingeniería de Telecomunicación
Procesado de señales acústicas e imágenes

Enrique Fernández Sánchez

Link al código: Github: Raniita/image-processing-matlab

20 de noviembre de 2021

Índice

1	Anonimizado	2
2	Contraste	4
3	Iluminación	5
4	Suavizado	6
5	Realzado	7
6	Ruido	10
7	Patrones	12
8	Pseudocoloración	13

Listado de códigos

1	Implementación anonimizado con MATLAB	2
2	Implementación de filtros para el realzado en MATLAB	7
3	Implementación filtro mediana para reducir ruido en MATLAB	10
4	Implementación de pseudocoloración en MATLAB	13

1 Anonimizado

En este ejercicio tenemos que realizar técnicas de anonimizado para recortar la información a procesar de la imagen 1, para ello podemos aplicar una técnica de máscara binaria o bien seleccionar manualmente la parte de la imagen a procesar.

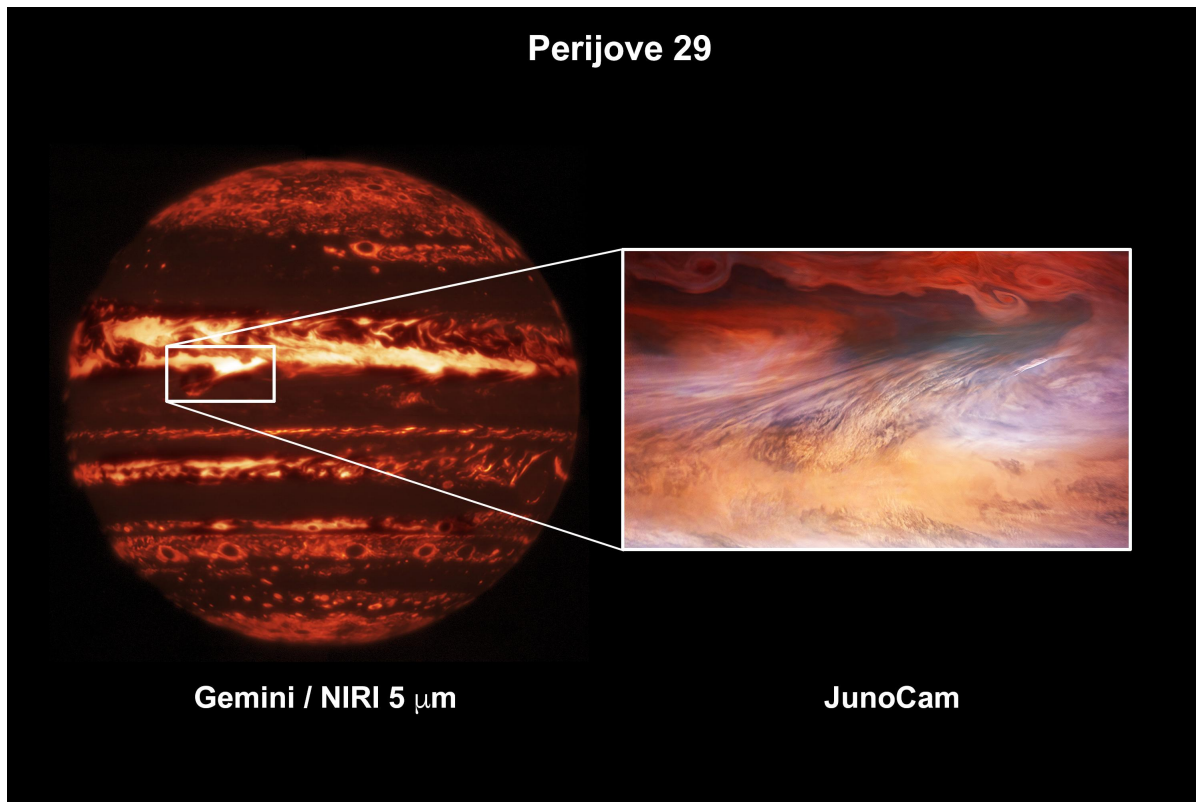


Figura 1: Imagen inicial anonimizado

En mi implementación, he optado por realizar una selección manual de la zona a recortar. Sin embargo, si tuviéramos que procesar muchas imágenes diferentes, la mejor opción sería realizar una máscara binaria, ya que así podríamos aplicarla de manera automática (sin intervención manual del usuario).

El código implementado sería el siguiente:

Código 1: Implementación anonimizado con MATLAB

```
1 % 1 - Anonimizado
2 % Enrique
3 % Ref: https://es.mathworks.com/help/images/ref/imcrop.html
4 clear;
5
6 % Cargamos la imagen
7 img = imread('anonimizado.jpg');
8
9
```

```

10 % Representamos la imagen original
11 figure
12 imshow(img);
13 title('Imagen inicial (sin recortar)')
14
15 % Recortamos la imagen con un rectangulo
16 [crop_img, rect_crop] = imcrop(img);
17
18 % Manera automatica
19 %rect_crop = [120 300 2750 1400]; % [xmin ymin width height]
20 %crop_img = imcrop(img, rect_crop);
21
22 % Comparativa imagen original vs imagen recortada
23 figure
24 %suptitle('Imagen Anonimizada')
25
26 subplot(1,2,1)
27 imshow(img)
28 title('Imagen inicial')
29
30 subplot(1,2,2)
31 imshow(crop_img)
32 title('Imagen recortada utilizando un rectangulo')
33

```

Una vez ejecutado el código, tendremos como salida una comparativa entre la imagen inicial (ver figura 1) y la imagen recortada:

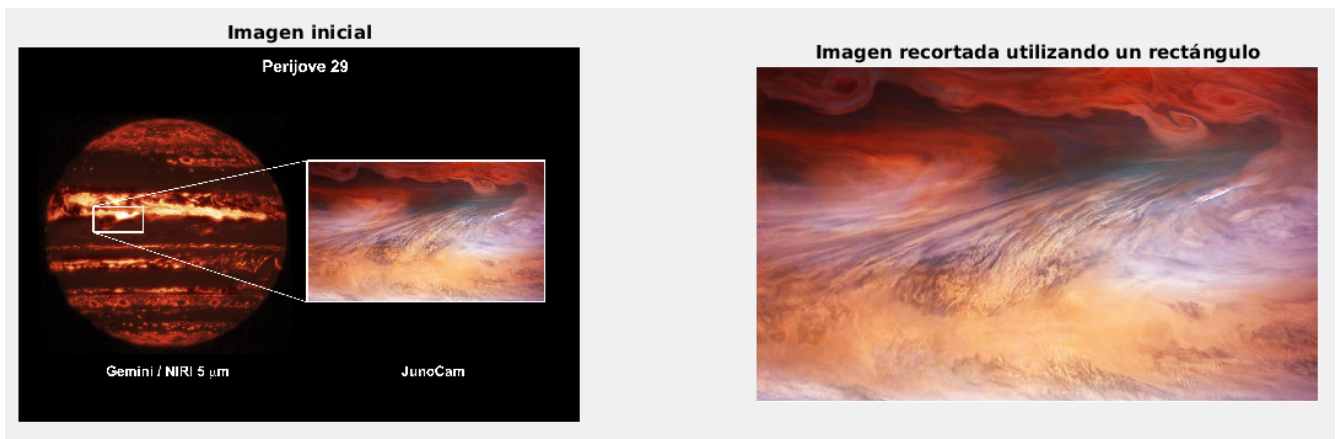


Figura 2: Figura de salida al ejecutar script de MATLAB anonimizado.m

2 Contraste

3 Iluminación

4 Suavizado

5 Realzado

En este ejercicio tenemos que aplicar técnicas de realzado para obtener más detalles en la imagen X. Para ello, aplicaremos diferentes filtros, con el objetivo de resaltar los cambios entre las diferentes zonas de color de la imagen.



Figura 3: Imagen inicial realzado

Para realzar la imagen podemos aplicar filtros directamente sobre la imagen original. En este caso, hemos aplicado un preajuste del histograma, para hacer que el margen dinámico ocupe todo el histograma. Una vez tenemos la imagen preajustada, la filtramos con `prewitt` o `sobel`.

Código 2: Implementación de filtros para el realzado en MATLAB

```
1 % 5 - Realzado
2 % Enrique
3 clear;
4
5 img_rgb = imread('realzado.jpg');
6
7 % Preajuste expansion margen dinamico
8 img_adj = imadjust(img_rgb, stretchlim(img_rgb), [ ]);
9
10 % Caso uno, prewitt
11 filter = fspecial('prewitt');
12 img_filtered = imfilter(img_adj, filter, 'replicate');
13 %img_rgb_1 = imadd(img_adj, img_filtered);
14 img_rgb_1 = img_adj - img_filtered;
15
```

```

16 % Caso dos, sobel
17 filter = fspecial('sobel');
18 img_filtered = imfilter(img_adj, filter, 'replicate');
19 %img_rgb_2 = imadd(img_adj, img_filtered);
20 img_rgb_2 = img_adj - img_filtered;
21
22 figure
23 subplot(2,2,1)
24 imshow(img_rgb, [])
25 title('Imagen original')
26
27 subplot(2,2,2)
28 imshow(img_adj, [])
29 title('Imagen preadjustada')
30
31 subplot(2,2,3)
32 imshow(img_rgb_1, [])
33 title('Imagen realzada (prewitt)')
34
35 subplot(2,2,4)
36 imshow(img_rgb_2, [])
37 title('Imagen realzada (sobel)')
38
39

```


Una vez ejecutado el código, obtenemos como salida una figura comparativa entre la imagen original, la imagen preajustada y las opciones de filtrado (`prewitt` y `sobel`).

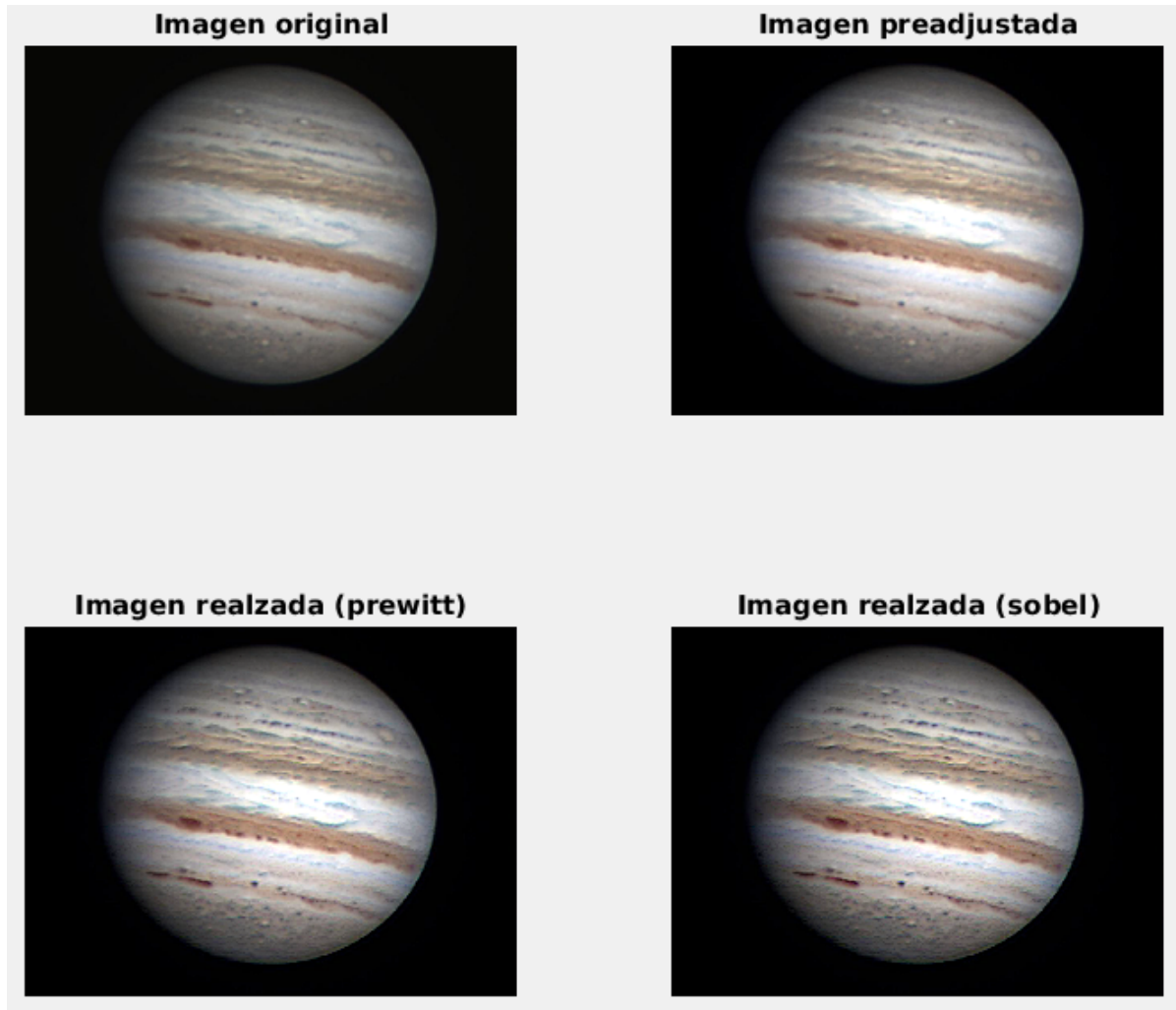


Figura 4: Imagen inicial ruido

Si tuviéramos que quedarnos con uno de los filtrados, el filtro `sobel` parece que realza mucho más las transiciones de color, cosa que en el caso de `prewitt` también distinguimos, pero en esta imagen en concreto el realzado es menor que respecto a `sobel`.

6 Ruido

En este ejercicio tenemos que aplicar técnicas de reducción de ruido a la imagen 5. Lo primero que tendríamos que intentar hacer es ver que tipo de ruido es, en este caso creemos que es ruido tipo “sal y pimienta”.



Figura 5: Imagen inicial ruido

Puesto que el ruido de la imagen se puede entender como un ruido de tipo “sal y pimienta”, una de las técnicas que podemos utilizar es ejecutar de manera recursiva un filtro de mediana sobre la imagen.

Código 3: Implementación filtro mediana para reducir ruido en MATLAB

```
1 % 6 - Ruido Sal y Pimienta
2 % Enrique
3 clear;
4
5 img_rgb = 'ruido.jpg';
6 neighborn = 4;
7 times = 5;
8
```

```

9 img = imread(img_rgb);
10 img = mat2gray(img, [0 255]);
11 img = rgb2gray(img);
12
13 figure
14 subplot(1,2,1)
15 imshow(img, [])
16 title('Imagen original [RGB]')
17
18 % Aplicamos filtro mediana
19 img_median = img;
20 for n=1:times,
21 img_median = medfilt2(img_median, [1 1]*neighborn);
22 end
23
24 % Representamos resultado
25 subplot(1,2,2)
26 imshow(img_median, []),
27 axis off image,
28 title(['Imagen eliminado el ruido usando filtro mediana con N=' num2str(neighborn)
        ' T=' num2str(times) ])
29

```

Una vez ejecutado el código, tendremos como salida una comparativa entre la imagen inicial (ver figura 5) y imagen resultante tras eliminar el ruido “sal y pimienta”:

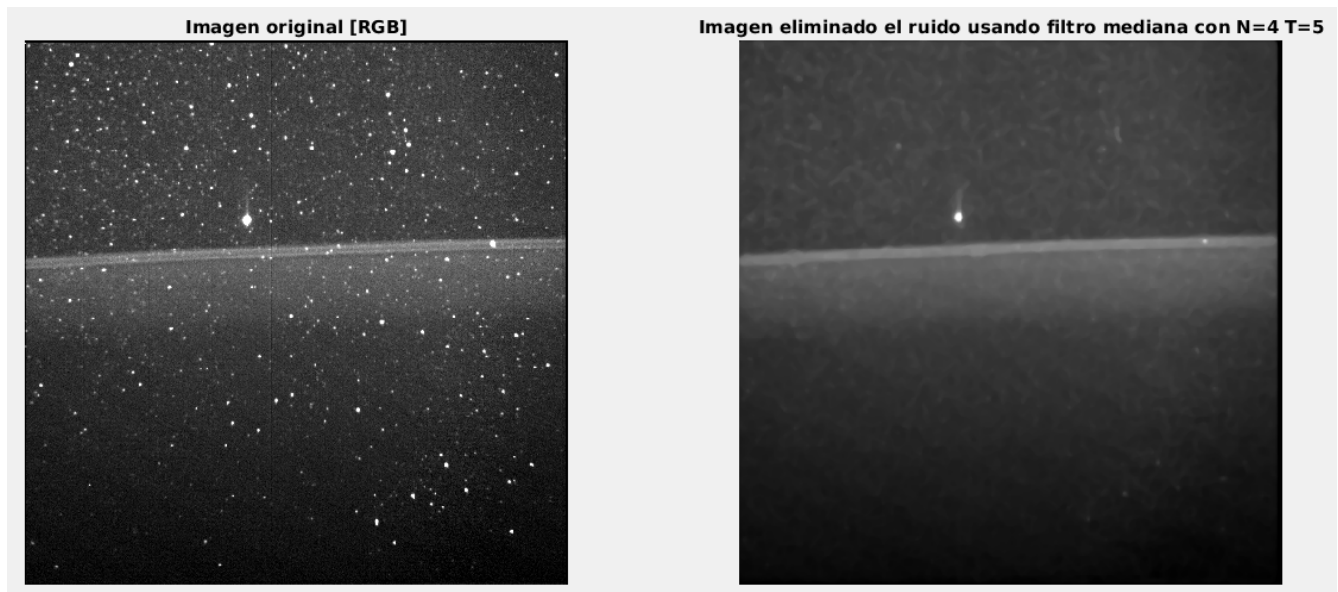


Figura 6: Figura de salida al ejecutar script de MATLAB ruido.m

7 Patrones

8 Pseudocoloración

En este ejercicio tenemos que aplicar técnicas de pseudocoloración a la imagen de un planeta (ver Figura X), con esto pretendemos ver detalles, que en el caso de ver la imagen en niveles de gris no veríamos.



Figura 7: Imagen inicial pseudocoloración

Para la resolución de este ejercicio hemos usado la técnica de coloración de una imagen dependiendo de su nivel de gris. Para ello, hemos probado todos los mapas de colores que MATLAB nos ofrece, sin embargo solo copper, pink, bone, hot y parula nos aportaban resultados útiles.

Código 4: Implementación de pseudocoloración en MATLAB

```
1 % 8 - Pseudocoloracion por nivel de gris
2 % Enrique
3
4 img_rgb = 'pseudocoloracion.jpg';
5 gray_lvl = 64;
6
7 img = imread(img_rgb);
8 img = mat2gray(img, [0 255]);
9 img = rgb2gray(img);
10
11 % Utilizamos 64 niveles de gris
12 img_colored = grayslice(img, gray_lvl);
13
14
```

```

15     % Representamos cada una de las coloraciones
16     figure
17     subplot(2,3,1),
18     subimage(img);
19     axis off image,
20     title('Imagen original en escala de gris')
21
22     subplot(2,3,2),
23     subimage(img_colored, copper(gray_lvl));
24     axis off image,
25     title('Pseudocoloracion (copper)')
26
27     subplot(2,3,3),
28     subimage(img_colored, pink(gray_lvl));
29     axis off image,
30     title('Pseudocoloracion (pink)')
31
32     subplot(2,3,4),
33     subimage(img_colored, bone(gray_lvl));
34     axis off image,
35     title('Pseudocoloracion (bone)')
36
37     subplot(2,3,5),
38     subimage(img_colored, hot(gray_lvl));
39     axis off image,
40     title('Pseudocoloracion (hot)')
41
42     subplot(2,3,6),
43     subimage(img_colored, parula(gray_lvl));
44     axis off image,
45     title('Pseudocoloracion (parula)')
46

```

Una vez ejecutado el código, tendremos como salida una comparativa entre la imagen inicial (ver figura 7) y las diferentes imágenes resultantes tras aplicar pseudocoloración:

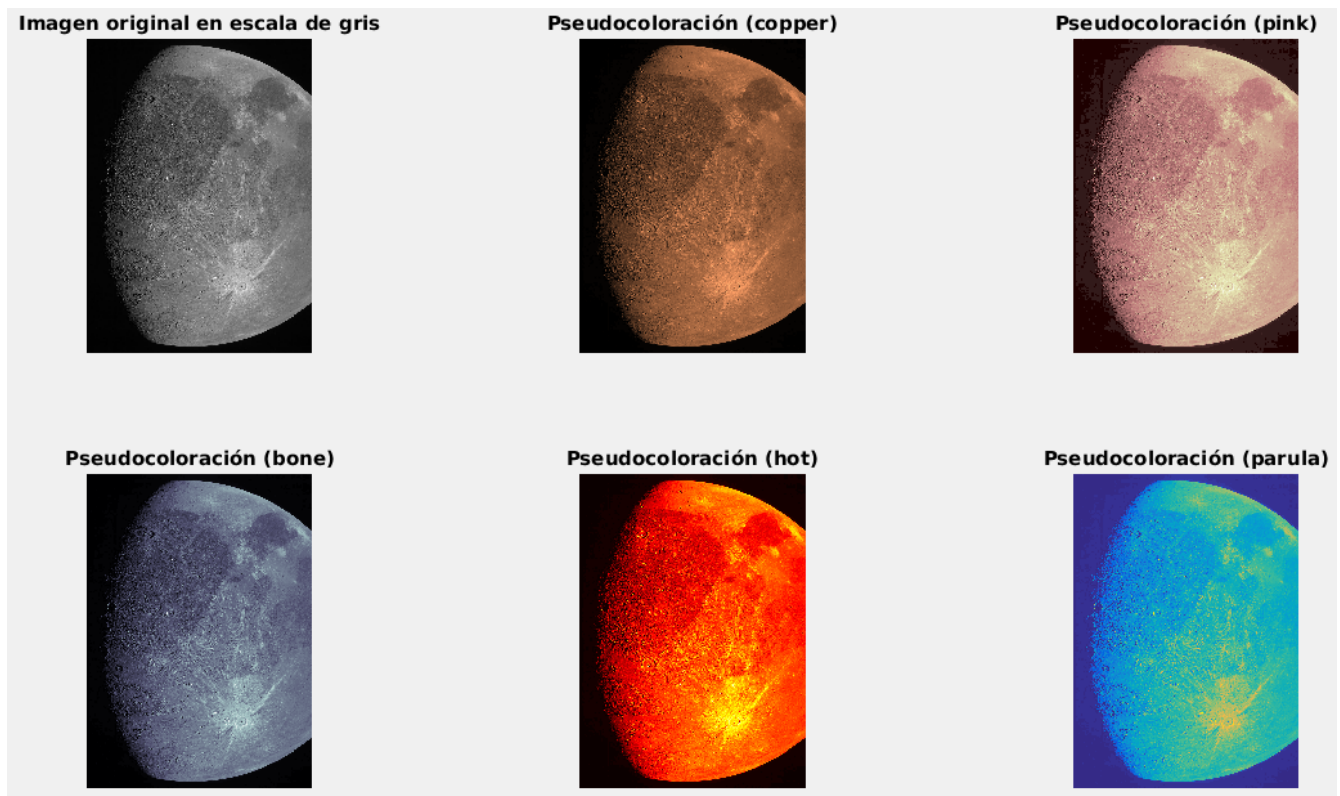


Figura 8: Figura de salida al ejecutar script de MATLAB `pseudocoloracion.m`

Si tuviera que elegir entre alguna de las imágenes obtenidas, considero que con `hot` y `parula` podemos distinguir más detalles, sin embargo el resto de coloraciones también aportan más detalle que la original.