

T-Boletin-de-laboratorio.pdf



Ruben_Bueno_Menendez



Teledetección



4º Grado en Ingeniería Informática - Ingeniería del Software

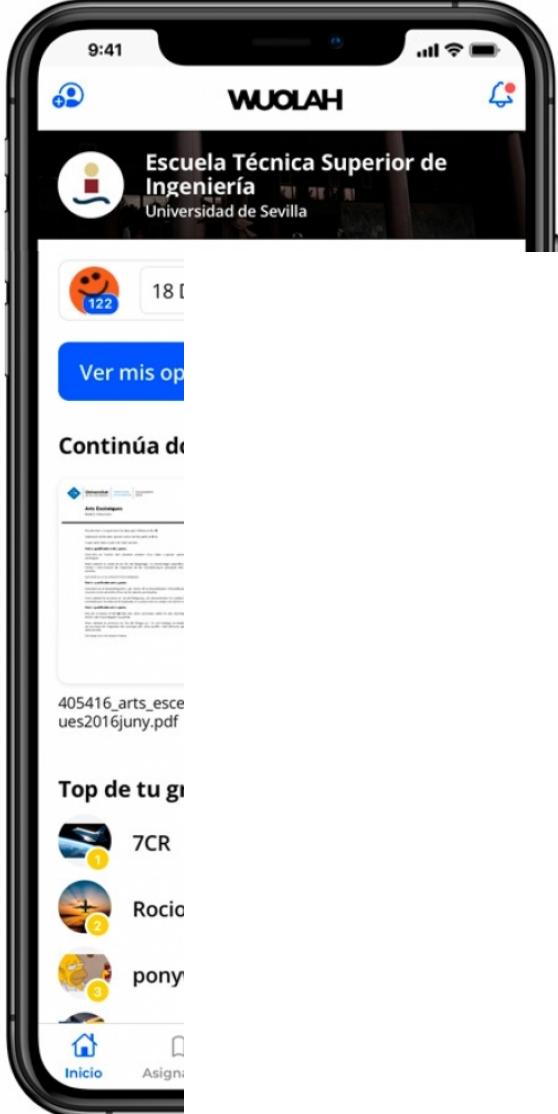


Escuela Técnica Superior de Ingeniería Informática
Universidad de Sevilla



Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.





Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the
App Store

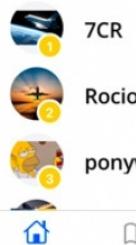
GET IT ON
Google Play

Continúa d



405416_arts_esce
ues2016juniy.pdf

Top de tu gr



Prácticas de Teledetección

~ Curso 2019-2020 ~

Prof. Alejandro Millán Calderón

Índice general

Laboratorio	2
Memoria de laboratorio	3
LAB1 - MATLAB	4
LAB2 - Histograma	17
LAB3 - Filtrado	26
LAB4 - Escalado	38
LAB5 - Color	49
LAB6 - Clasificación	67
LAB7 - Geocorrección	74



**KEEP
CALM
AND
ESTUDIA
UN POQUITO**

Laboratorio

El programa de prácticas propuesto tiene como objetivo la realización en el ordenador de una serie de ejercicios que permitan al alumnado realizar las diferentes técnicas expuestas a lo largo de la asignatura. Para ello, es fundamental contar con una herramienta que permita manipular imágenes, así como realizar cálculos numéricos complejos de manera simple y eficiente. Así, la herramienta elegida es MATLAB. Se trata de un lenguaje de alto nivel orientado fundamentalmente al cálculo numérico y que puede ser utilizado tanto a nivel de línea de comandos como a nivel de *script* (un *script* es un archivo de texto que contiene una serie de comandos). Presenta las siguientes ventajas fundamentales:

- Incluye todas las funciones necesarias para realizar las técnicas propuestas.
- Permite trabajar con archivos de imagen de manera transparente al usuario en cuanto al formato de almacenamiento.
- Está disponible para los sistemas operativos más habituales en ordenadores personales (Linux, Mac OS y Windows).

Este boletín de ejercicios ha sido preparado utilizando concretamente la versión de MATLAB R2017a.

Memoria de laboratorio

De cara a la evaluación de las prácticas, cada alumno debe elaborar una memoria de laboratorio y entregarla en formato PDF a final de curso. Para ello se deben tener en cuenta las siguientes consideraciones:

- Al final de cada sección de laboratorio se indica que material debe incluirse en la memoria de prácticas.
- De cara a la entrega de imágenes propias, el alumno debe descargar algunas a su elección para realizar los tratamientos.
- Algunos sitios de dónde se pueden descargar este tipo de imágenes:
 - <https://eos.com/landviewer/>
 - <https://gisgeography.com/free-satellite-imagery-data-list/>
- Software de tratamiento de imágenes de satélite que puede utilizar para convertir las imágenes a otros formatos:
 - <https://step.esa.int/main/download/snap-download/>



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

LAB1 - MATLAB

El objetivo de esta primera sesión de laboratorio es suponer una primera toma de contacto con el entorno de desarrollo. Al final de esta práctica, el alumno debería saber cómo realizar las siguientes tareas:

- Ejecutar comandos en la consola de MATLAB.
- Solicitar ayuda sobre la sintaxis de un determinado comando.
- Trabajar con vectores y matrices.
- Crear una función como archivo independiente y utilizarla.
- Trabajar con imágenes externas (carga y almacenamiento).

Ejercicio 1.1

En la línea de comandos, ejecute las siguientes instrucciones:

```
a = 5
b = a ^ 2
x = 0:1:9
y = x .^ 2
z = x * 2
c = 10:2:20
d = c - 1
clear
sin(pi / 4)
x = 0:0.01:2 * pi
y = sin ( x )
plot(x)
plot ( y )
plot(x, y)
z = cos(x)
plot(x, y, 'r', x, z, 'g')
```

a = 5

>> a = 5

a =

5

b = a ^ 2

>> b = a ^ 2

b =

25

x = 0:1:9

x =

0 1 2 3 4 5 6 7 8 9

y = x .^ 2

5

```
>> y = x .^ 2

y =
0     1     4     9    16    25    36    49    64    81

z = x * 2

>> z = x * 2

z =
0     2     4     6     8    10    12    14    16    18

c = 10:2:20

>> c = 10:2:20

c =
10    12    14    16    18    20

d = c - 1

>> d = c - 1

d =
9     11    13    15    17    19
```

Tras la ejecución de todo lo anterior tenemos las siguientes variables:

Workspace	
Name	Value
a	5
b	25
c	[10,12,14,16,18,20]
d	[9,11,13,15,17,19]
x	[0,1,2,3,4,5,6,7,8,9]
y	[0,1,4,9,16,25,36,49,64,81]
z	[0,2,4,6,8,10,12,14,16,18]

clear

Tras la ejecución de este comando perdemos todos los valores:

Workspace	
Name ▾	Value

$\sin(\pi / 4)$

```
>> sin(pi / 4)
```

ans =

0.7071

x = 0:0.01:2 * pi

Se inserta en la variable “x” 629 valores que van desde 0 hasta $2 * \pi = 6,28$ incrementándose en 0,01

```
>> x = 0:0.01:2 * pi

x =

Columns 1 through 9

    0    0.0100    0.0200    0.0300    0.0400    0.0500    0.0600    0.0700    0.0800

Columns 10 through 18

    0.0900    0.1000    0.1100    0.1200    0.1300    0.1400    0.1500    0.1600    0.1700

Columns 19 through 27

    0.1800    0.1900    0.2000    0.2100    0.2200    0.2300    0.2400    0.2500    0.2600

Columns 28 through 36

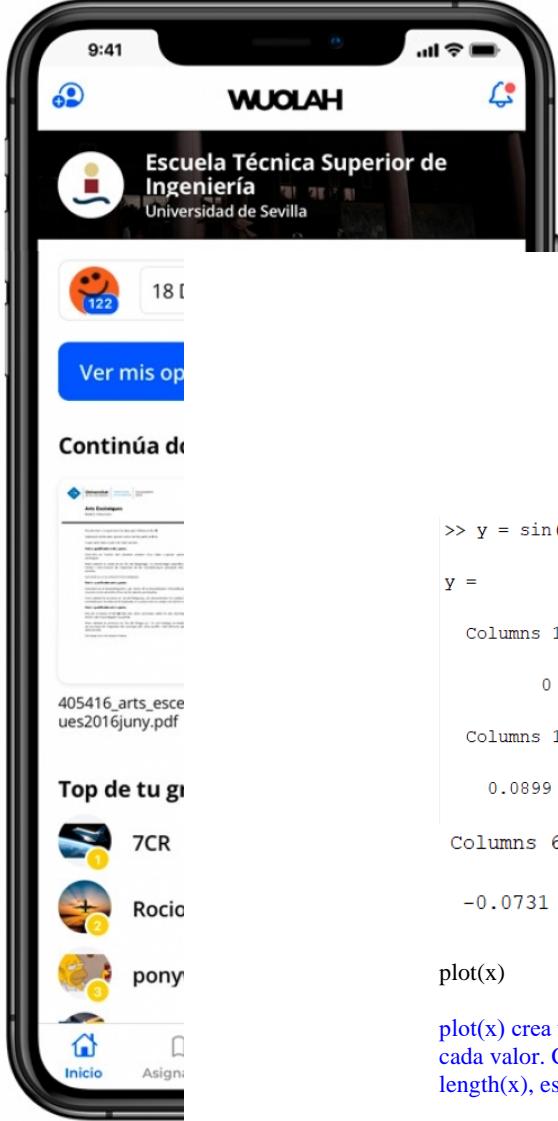
    0.2700    0.2800    0.2900    0.3000    0.3100    0.3200    0.3300    0.3400    0.3500

Columns 622 through 629

    6.2100    6.2200    6.2300    6.2400    6.2500    6.2600    6.2700    6.2800
```

y = sin(x)

Se inserta en la variable “y” los 629 valores de x aplicándole antes la función seno



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the App Store

GET IT ON Google Play

```
>> y = sin(x)

y =

Columns 1 through 9

    0     0.0100    0.0200    0.0300    0.0400    0.0500    0.0600    0.0699    0.0799

Columns 10 through 18

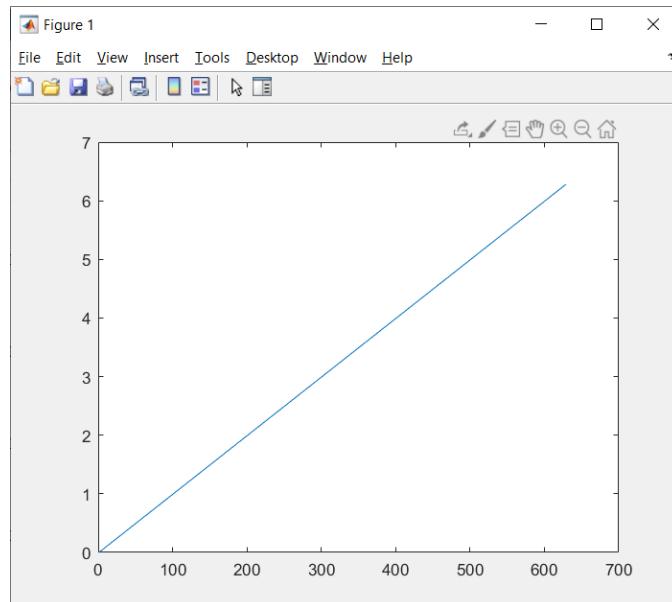
    0.0899    0.0998    0.1098    0.1197    0.1296    0.1395    0.1494    0.1593    0.1692

Columns 622 through 629

   -0.0731   -0.0631   -0.0532   -0.0432   -0.0332   -0.0232   -0.0132   -0.0032

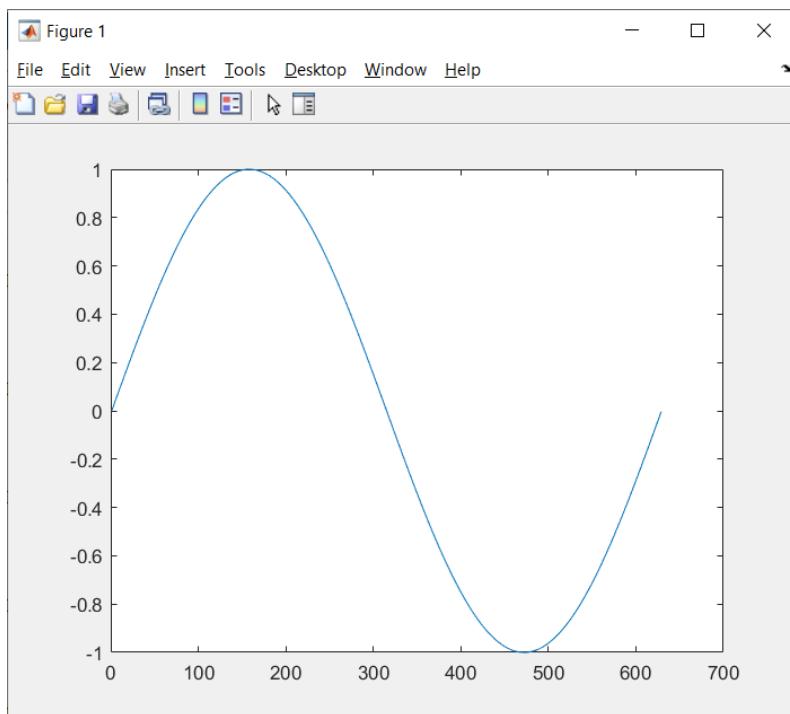
plot(x)
```

plot(x) crea un gráfico de líneas 2D de los datos de “x” en comparación con el índice de cada valor. Como en este caso “x” es un vector, la escala del eje x oscila entre 1 y length(x), es decir, 629.



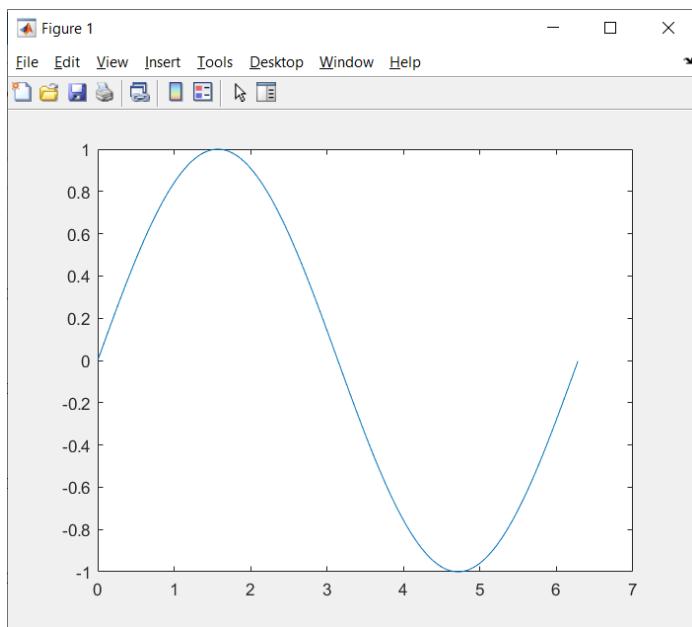
plot(y)

plot(y) crea un gráfico de líneas 2D de los datos de “y” en comparación con el índice de cada valor. Como en este caso “y” es un vector, la escala del eje x oscila entre 1 y length(y), es decir, 629.



`plot(x, y)`

`plot(x,y)` crea un gráfico de líneas 2D de los datos de “y” frente a los valores correspondientes de “x”. Como “x” e “y” son ambos vectores, deben tener la misma longitud y la función plot traza y frente a “x”.



`z = cos(x)`

```

>> z = cos(x)

z =

Columns 1 through 9

1.0000    1.0000    0.9998    0.9996    0.9992    0.9988    0.9982    0.9976    0.9968

Columns 10 through 18

0.9960    0.9950    0.9940    0.9928    0.9916    0.9902    0.9888    0.9872    0.9856

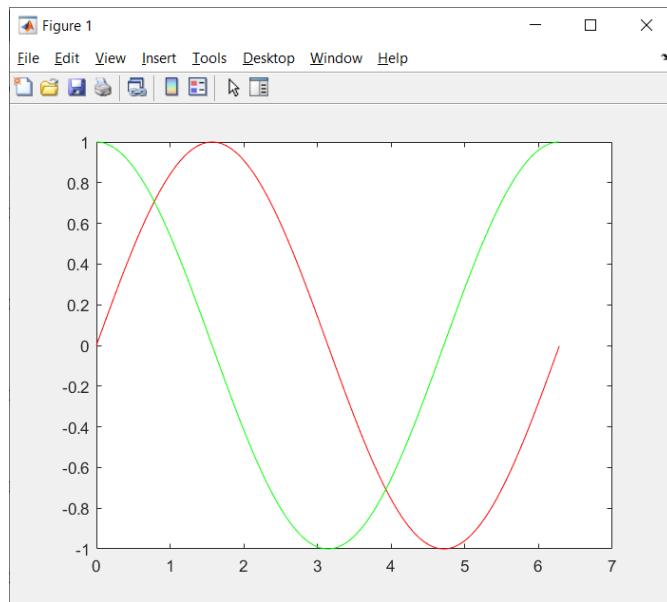
Columns 622 through 629

0.9973    0.9980    0.9986    0.9991    0.9994    0.9997    0.9999    1.0000

```

`plot(x, y, 'r', x, z, 'g')`

En este caso, se crea un gráfico de líneas 2D de los datos de “y” frente a los valores correspondientes de “x” en rojo (‘r’) y otro de “z” frente a los valores correspondientes de “x” en verde (‘g’)



Ejercicio 1.2

Cree un archivo con el nombre *facto1.m* que contenga el siguiente código:

```
function f = facto1 (N)
f = 1;
for n = 2: N
    f = f * n;
end
```

Ejecute la función anterior escribiendo su nombre en la línea de comandos. Si no se encuentra la función, utilice el comando *cd* para llegar al directorio dónde se halle el archivo creado.

Podemos observar que *f* será el valor final que devuelve la función y que será necesario pasar un valor numérico por parámetro, por ejemplo, 10:

```
>> facto1(10)

ans =

3628800
```



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the App Store

GET IT ON Google Play

Ejercicio 1.3

Cree un archivo con el nombre *facto2.m* que contenga el siguiente código:

```
function f = facto2 (N)
if N == 0
    f = 1;
else
    f = N * facto2 (N - 1);
end
```

Ejecute la función anterior.

Podemos observar que *f* será el valor final que devuelve la función y en comparación al anterior ejercicio se utiliza recursividad. Será necesario pasar un valor numérico por parámetro, por ejemplo, 10:

```
>> facto2(10)

ans =
```

3628800

Ejercicio 1.4

Ejecute los siguientes comandos:

```
clear  
a = [1 2 3 4 5]  
b = [1 2; 3 4; 5 6]  
c = 1:20  
length(a)  
size(b)  
size(b, 1)  
size(b, 2)  
rem(c, 4)
```

¿Qué cree usted que devuelven las funciones *length* y *size*? ¿Y la función *rem*? Si no lo sabe, ejecute los siguientes comandos:

```
help length  
help size  
help rem
```

length devuelve la longitud de la dimensión de matriz más grande. En este caso, para a, al tratarse de un vector, la longitud es simplemente el número de elementos.

size devuelve un vector de fila cuyos elementos contienen la longitud de la dimensión correspondiente. En este caso, para b, es una matriz de 3 x 2, *size(a)* devuelve el vector [3 2].

Si se especifica en *size* una dimensión, entonces devuelve la longitud de la dimensión especificada. En este caso, para (b, 1) devuelve 3 y para (b, 2) devuelve 2.

rem devuelve el resto después de la división. En este caso, la división de los valores de c entre 4, donde los valores de c es el dividendo y 4 es el divisor.

Ejercicio 1.5

Existen una serie de funciones nativas que permiten trabajar con imágenes digitales de forma sencilla. Estas son:

- *imread*: para leer una imagen desde un archivo.
- *imshow*: para mostrar una imagen previamente leída en pantalla.
- *imwrite*: para escribir una imagen a un archivo.

Utilice estas funciones para visualizar en pantalla algunas de las imágenes suministradas en el paquete de laboratorio (cada banda está separada en un archivo PNG monocromo independiente).

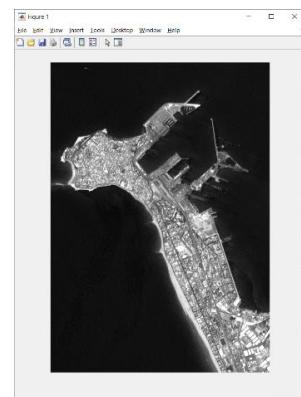
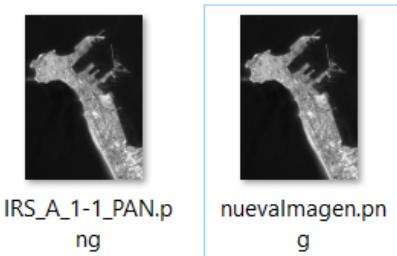
He usado de ejemplo la imagen IRS_A_1-1_PAN.png contenida dentro de la carpeta "Imágenes". Usando el comando `a = imread('imágenes/IRS_A_1-1_PAN.png')` se obtiene lo siguiente:

Command Window		Workspace	
		Name	Value
235	221	a	117x832 uint8
235	253		
240	234		
208	185		
174	173		
192	214		
214	216		
200	182		
182	174		
126	140		
107	108		
232	255		
255	247		
247	211		
178	183		
183	196		
196	213		
213	236		
236	227		
227	201		
201	165		
165	141		
141	109		
109	135		
228	240		
240	255		
255	252		
252	223		
223	197		
197	209		
209	220		
220	226		
226	234		
234	206		
206	169		
169	136		
136	116		
116	116		
116	150		

Cada pixel corresponde a una posición de la matriz.

Usando el comando `imshow(a)` se muestra la imagen como en la figura de la derecha

Usando el comando `imwrite` escribimos una imagen en un archivo, en este caso por ejemplo vamos a escribir la imagen leída en otro nuevo archivo con `imwrite(a, 'Imágenes/nuevaImagen.png')`



Ejercicio 1.6

Cree un archivo con el nombre *qsort.m* que ordene ascendenteamente un vector utilizando el algoritmo Quicksort. Para ordenar ascendenteamente una secuencia x utilizando dicho algoritmo, es necesario seguir un proceso recursivo:

- El caso base de la recursión es aquel en el que la longitud de x es cero, por lo que no hay nada que ordenar. Para comprobar esto puede utilizar la función *isempty* o la función *length*.
- En el caso recursivo, se elige un elemento cualquiera de x (denominado pivote, p) y se divide la secuencia x en dos subsecuencias: a , que contiene al resto de elementos menores o iguales que p , y b , que contiene al resto de elementos mayores que p . A continuación se ordenan esta dos subsecuencias de manera recursiva (supongamos que el resultado de esta ordenación es *aord* y *bord*). Por último, se obtiene la secuencia ordenada mediante la concatenación de las dos subsecuencias ordenadas y el pivote:

```
xord = [aord, p, bord]
```

El comando nativo *sort* también ordena vectores. Compruebe si su función es correcta ejecutando los siguientes comandos:

```
x = rand(1, 100);  
y = sort(x);  
z = qsort(x);  
b = (y == z);  
min(b)
```

Compruebe también que no hay problema si el vector inicial contiene elementos repetidos:

```
a = rand(1, 30);  
x = [a a a];  
y = sort(x);  
z = qsort(x);  
b = (y == z);  
min(b)
```

¿Qué debe devolver *min(b)* para saber que ambos resultados son idénticos?

MEMORIA:

- Código 1.6



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the App Store GET IT ON Google Play

```
function [xord] = qsort(X)
if isempty(X)
    xord = X;
else
    x = randi(length(X)); %Generamos un número aleatorio entre 1 y el tamaño del vector
    p = X(x); %Extraemos como pivote el valor de la posición x
    X(x) = [];
    a = X(X <= p); %Subtraemos un vector para todos los elementos menor o igual al pivote
    b = X(X > p); %Subtraemos un vector para todos los elementos mayor al pivote
    aord = qsort(a);
    bord = qsort(b);
    xord = [aord, p, bord];
end
```

Como min lo que devuelve es el valor mínimo del vector b, se espera que devuelva 1. Esto es porque b contiene un vector de 100 elementos, los cuales deben ser 1 (haciendo referencia a true), ya que deben ser iguales los elementos del vector "y" y del vector "z" para cada posición. Si estuviera mal la ordenación entonces habría algún elemento como mínimo que no correspondería y entonces el min devolvería 0.

Podemos observar que el valor resultante ha sido 1 como era lo esperado

min(b)	<input checked="" type="checkbox"/> ans <input checked="" type="checkbox"/> b <input type="checkbox"/> x <input type="checkbox"/> y <input type="checkbox"/> z	1 1x100 logical 1x100 double 1x100 double 1x100 double
ans =		
<u>logical</u>		
1	<input type="checkbox"/> a <input checked="" type="checkbox"/> ans <input checked="" type="checkbox"/> b <input type="checkbox"/> x <input type="checkbox"/> y <input type="checkbox"/> z	1x30 double 1 1x90 logical 1x90 double 1x90 double 1x90 double

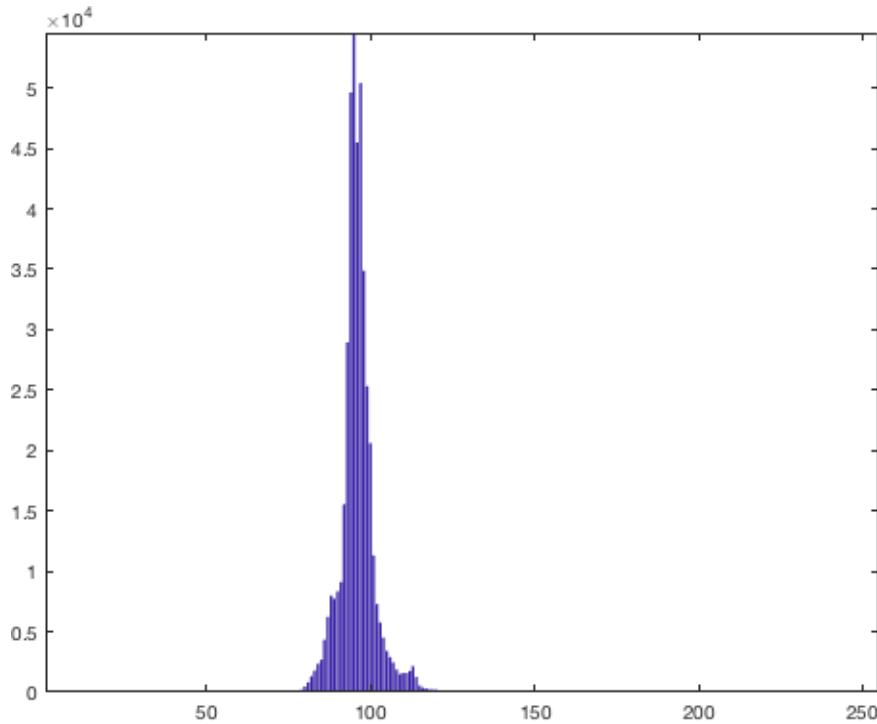
LAB2 - Histograma

Esta sesión se centra en los tratamientos de imágenes digitales basados en el histograma. Al realizar los ejercicios tenga en cuenta los siguientes aspectos:

- Considere que todas las imágenes son monobanda. Para comprobar sus resultados utilice alguna de las suministradas en el paquete de laboratorio. Una apropiada puede ser la NOAA (banda 5).
- Trabaje únicamente en el rango [1, 255] (el valor 0 se reserva para *ND* inválido).
- Todas las funciones de tratamiento deben visualizar en pantalla la imagen resultante al finalizar.

Ejercicio 2.1

Escriba una función llamada *histo* que calcule el histograma de una imagen monobanda y lo visualice en pantalla. Represente únicamente los *ND* entre 1 y 255. Para una presentación mejorada, utilice las funciones *bar* y *axis*.



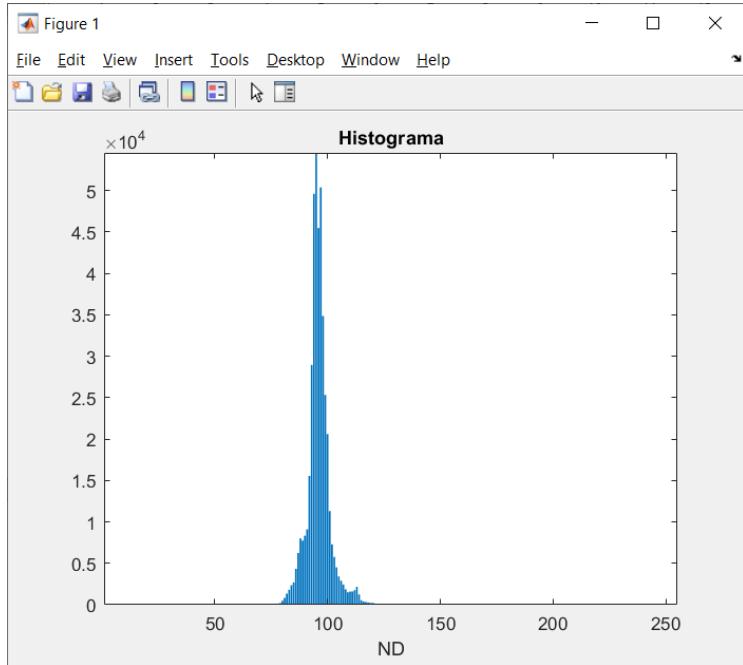
Histograma de la imagen NOAA (banda 5).

El código de la función es el siguiente:

```
function h = histo(im, b)
    im = im(:,:,b); %Se indica la banda a extraer para que sea monobanda
    [F, C] = size(im);
    h = zeros(1,255);
    for f = 1:F
        for c = 1:C
            nd = im(f,c);
            if nd > 0
                h(nd) = h(nd) + 1;
            end
        end
    end
    bar(h);
    axis([1, 255, 0, max(h)]);
    xlabel('ND');
    title('Histograma')
```

Tras ejecutar los siguientes comandos conseguimos el histograma:

```
im = imread('ímágenes/NOAA_5-5_LWIR.png');  
histo(im, 1);
```





Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the App Store

GET IT ON Google Play

Ejercicio 2.2

Escriba una función llamada *expan* que realice la expansión lineal de una imagen monobanda a partir de dos niveles de corte indicados como parámetros de entrada. La función debe devolver la imagen resultante así como visualizarla en pantalla. Recuerde trabajar exclusivamente con *ND* entre 1 y 255 (el 0 se reserva para *ND* inválidos).

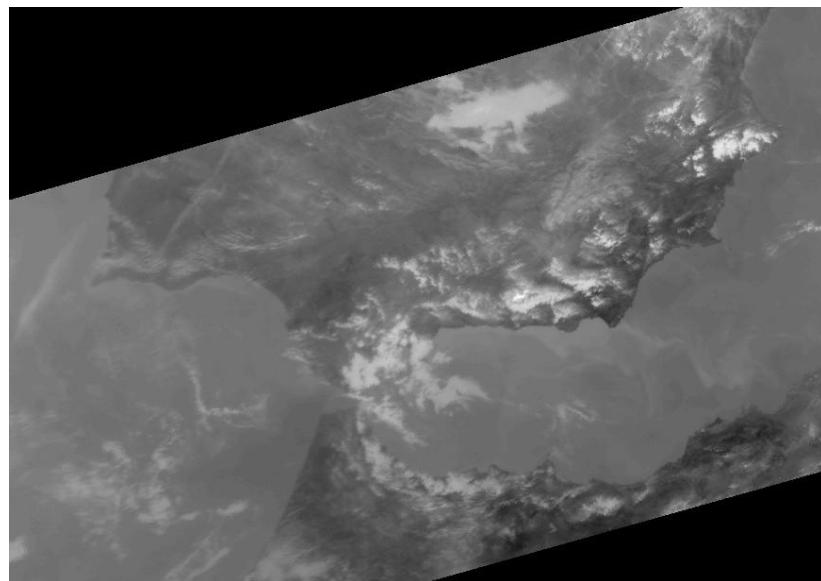


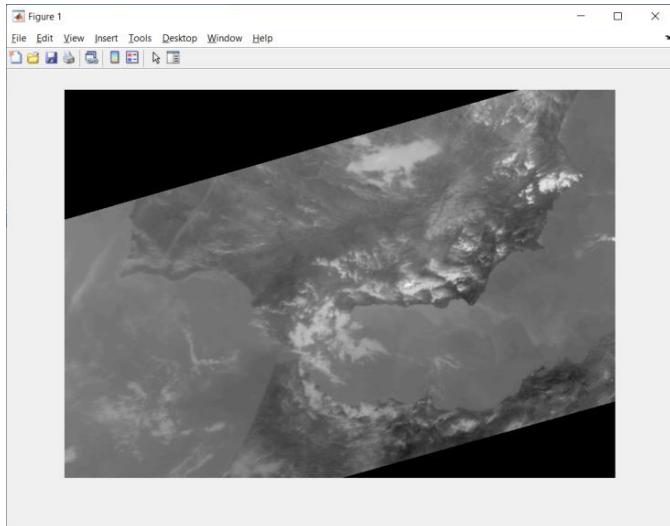
Imagen NOAA (banda 5) tras expansión lineal con cortes a 70 y 130.

El código de la función es el siguiente:

```
function im2 = expan(im1, m, M, b)
    im1 = im1(:,:,b); %Se indica la banda a extraer para que sea monobanda
    [F, C] = size(im1);
    im2 = uint8(zeros(F, C));
    for f = 1:F
        for c = 1:C
            nd = im1(f,c);
            if nd > 0
                ndp = (nd - m) * (254 / (M - m)) + 1;
                im2(f, c) = ndp;
            end
        end
    end
    imshow(im2);
```

Tras ejecutar los siguientes comandos conseguimos la imagen resultante:

```
im = imread('imágenes/NOAA_5-5_LWIR.png');  
expan(im, 70, 130, 1);
```



Ejercicio 2.3

Escriba una función llamada *corte* que realice un corte de colas a una imagen monobanda a partir de un determinado porcentaje de corte indicado como parámetro de entrada (para ello puede llamar a la función *expan* del ejercicio anterior).

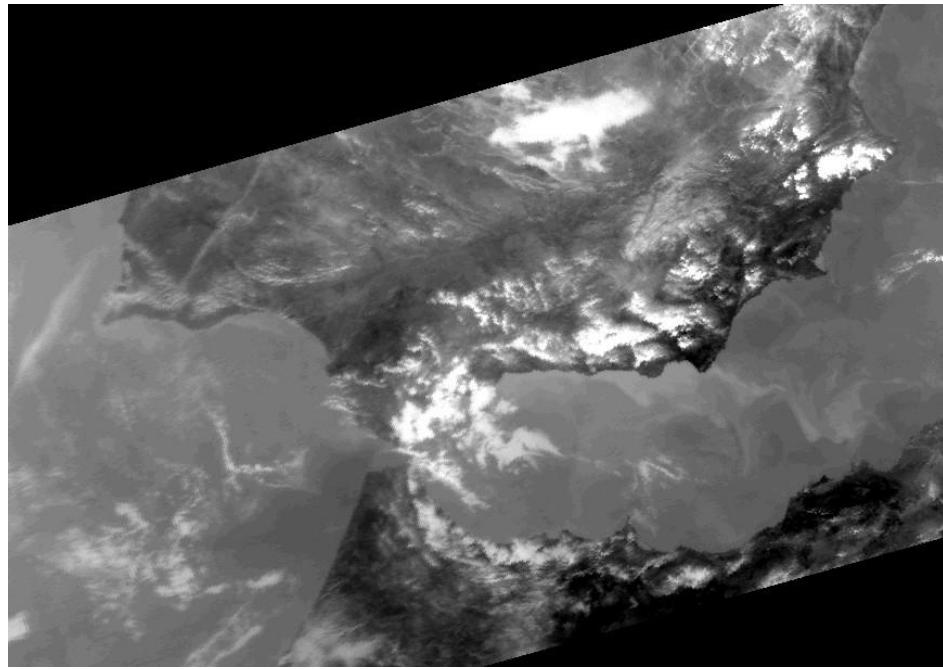


Imagen NOAA (banda 5) tras corte de colas al 1 %.

El código de la función es el siguiente:

```

function im2 = corte(im1, p, b)
    im1 = im1(:, :, b); %Se indica la banda a extraer para que sea monobanda
    numPixels = sum(im1(:) > 0);
    pixelsPercentage = numPixels * (p / 100);
    h = histo(im1, 1);
    ac = 0;
    for v = 1:length(h)
        ac = ac + h(v);
        if ac >= pixelsPercentage
            m = v;
            break
        end
    end
    ac = 0;
    for v = length(h):-1:1
        ac = ac + h(v);
        if ac >= pixelsPercentage
            M = v;
            break
        end
    end
    im2 = expand(im1, m, M, 1);

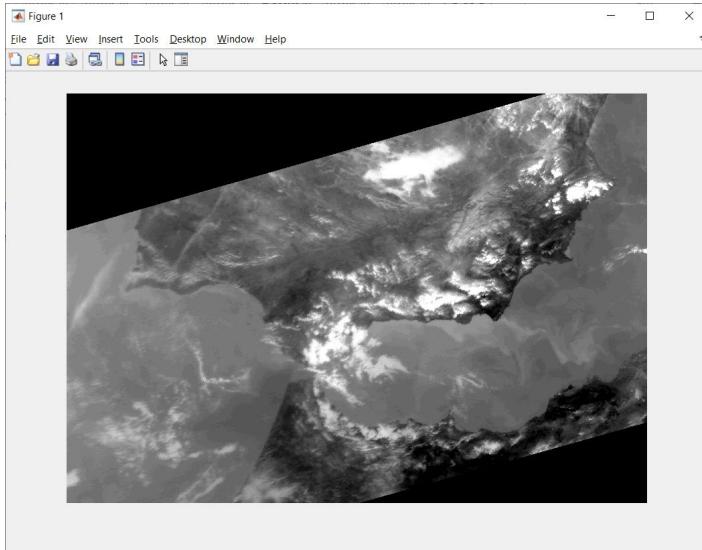
```

Tras ejecutar los siguientes comandos conseguimos la imagen resultante:

```

im = imread('imágenes/NOAA_5-5_LWIR.png');
corte(im, 1, 1);

```





Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

Ejercicio 2.4

Escriba una función llamada *ecual* que realice la ecualización de una imagen monobanda.

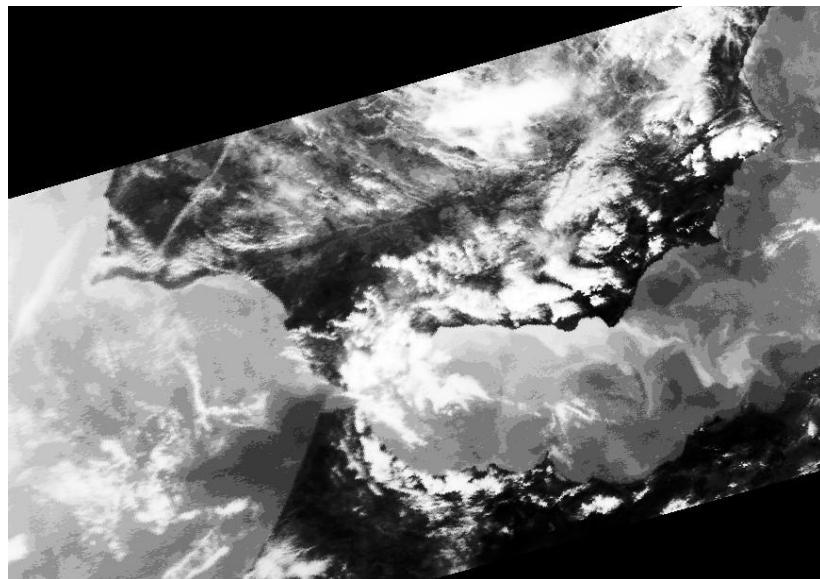


Imagen NOAA (banda 5) tras ecualización.

MEMORIA:

- Código 2.3 + Imagen propia tratada (antes/después)
- Código 2.4 + Imagen propia tratada (antes/después)

El código de la función es el siguiente:

```

function im2 = ecual(im1, b)
    im1 = im1(:, :, b); %Se indica la banda a extraer para que sea monobanda
    [F, C] = size(im1);
    im2 = uint8(zeros(F, C));
    h = histo(im1, 1);
    ac = 0;
    for v = 1:length(h)
        ac = ac + h(v);
        h(v) = ac;
    end
    P = sum(im1(:) > 0);
    FE = 255 / P;
    for f = 1:F
        for c = 1:C
            nd = im1(f, c);
            if nd > 0
                ndp = round(h(nd) * FE);
                im2(f, c) = ndp;
            end
        end
    end
    imshow(im2);

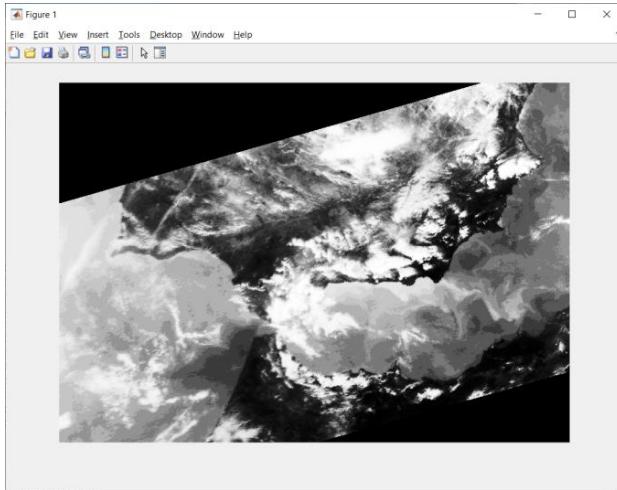
```

Tras ejecutar los siguientes comandos conseguimos la imagen resultante:

```

im = imread('imágenes/NOAA_5-5_LWIR.png');
ecual(im, 1);

```



LAB3 - Filtrado

Esta sesión se centra en los tratamientos de imágenes digitales basados en filtros. Al realizar los ejercicios tenga en cuenta los siguientes aspectos:

- Considere que todas las imágenes son monobanda.
- Para comprobar sus resultados utilice alguna de las suministradas en el paquete de laboratorio.
- Salvo en el ejercicio 3.4, corrección del error de moteado, trabaje únicamente en el rango [1, 255] (el valor 0 se reserva para *ND* inválido).
- Todas las funciones de tratamiento deben visualizar en pantalla la imagen resultante al finalizar.

Ejercicio 3.1

Escriba una función llamada *filtro* que realice el filtrado lineal de una imagen monobanda a partir de una matriz 3×3 de coeficientes de filtrado indicada como parámetro de entrada. La función debe devolver la imagen resultante así como visualizarla en pantalla.

El código de la función es el siguiente:

```
function im2 = filtro(im1, cf, b)
im1 = im1(:,:,b); %Se indica la banda a extraer para que sea monobanda
[F, C] = size(im1);
im2 = uint8(zeros(F, C));
for f = 2:F-1 %Los bordes de la imagen no se tienen en cuenta, así que se excluyen
    for c = 2:C-1
        nd = im1(f,c);
        if nd > 0
            en = double(im1(f-1:f+1, c-1:c+1));
            ndp = sum(sum(en .* cf));
            im2(f, c) = ndp;
        end
    end
end
imshow(im2);
```



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

Ejercicio 3.2

Utilice la función *filtro* del ejercicio anterior para aplicar los siguientes filtros a algunas de las imágenes disponibles:

1	1	1
1	1	1
1	1	1

Media

1	1	1
1	2	1
1	1	1

Media + Imagen

1	2	1
2	4	2
1	2	1

Gaussiano

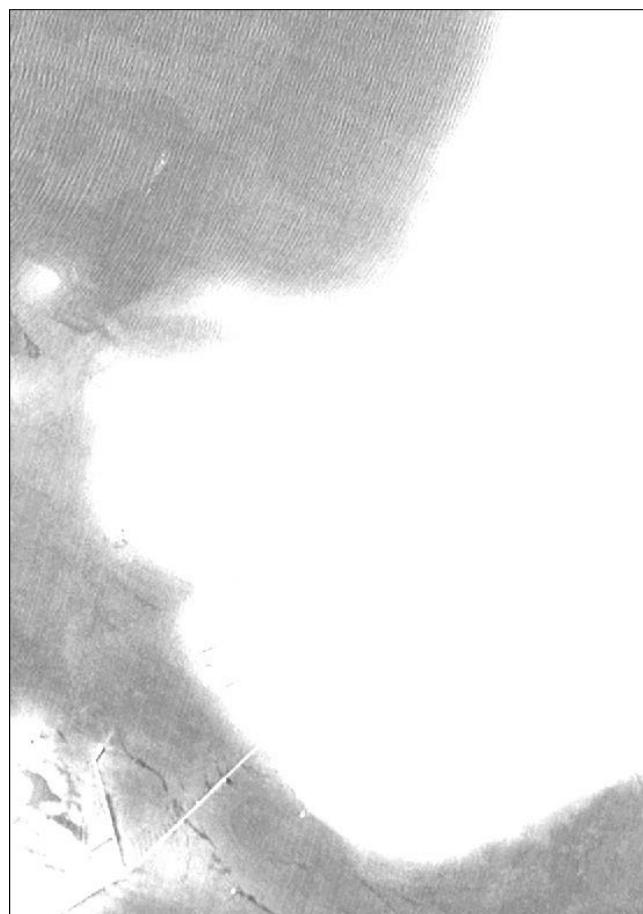
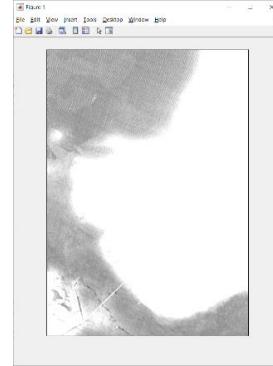


Imagen IRS B tras aplicar filtro de media sin normalización de coeficientes.

Tras ejecutar los siguientes comandos conseguimos la imagen resultante:

Filtro media:

```
im = imread('imágenes/IRS_B_1-1_PAN.png');
cf = ones(3,3);
filtro(im, cf, 1);
```



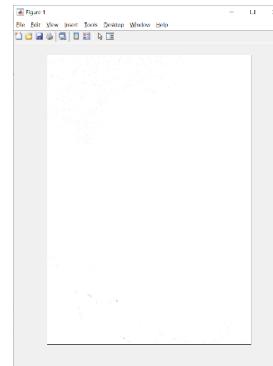
Filtro media + imagen:

```
im = imread('imágenes/IRS_B_1-1_PAN.png');
cf = ones(3,3);
cf(2,2) = 2;
filtro(im, cf, 1);
```



Filtro gaussiano:

```
im = imread('imágenes/IRS_B_1-1_PAN.png');
cf = [1,2,1;2,4,2;1,2,1];
filtro(im, cf, 1);
```



Ejercicio 3.3

Modifique la función *filtro* anterior para que, antes de realizar el proceso, normalice la matriz de coeficientes (su suma debe ser la unidad). Repita el proceso con las mismas imágenes.



Imagen IRS B tras aplicar filtro de media con normalización de coeficientes.

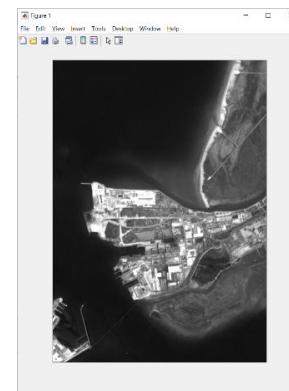
El código de la función es el siguiente:

```
function im2 = filtro(im1, cf, b)
im1 = im1(:,:,b); %Se indica la banda a extraer para que sea monobanda
[F, C] = size(im1);
im2 = uint8(zeros(F, C));
cf = cf ./ sum(sum(cf)); %Se normaliza para que la suma de los coeficientes sea 1
for f = 2:F-1 %Los bordes de la imagen no se tienen en cuenta, así que se excluyen
    for c = 2:C-1
        nd = im1(f,c);
        if nd > 0
            en = double(im1(f-1:f+1, c-1:c+1));
            ndp = sum(sum(en .* cf));
            im2(f, c) = ndp;
        end
    end
end
imshow(im2);
```

Tras ejecutar los siguientes comandos conseguimos la imagen resultante:

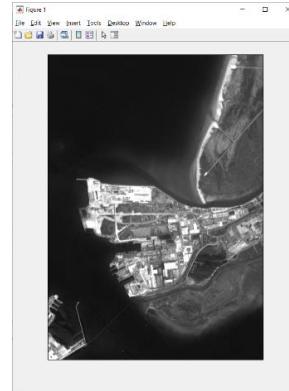
Filtro media:

```
im = imread('imágenes/IRS_B_1-1_PAN.png');
cf = ones(3,3);
filtro(im, cf, 1);
```



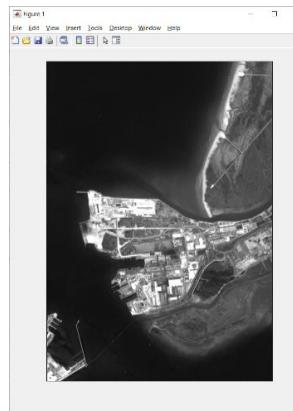
Filtro media + imagen:

```
im = imread('imágenes/IRS_B_1-1_PAN.png');
cf = ones(3,3);
cf(2,2) = 2;
filtro(im, cf, 1);
```



Filtro gaussiano:

```
im = imread('imágenes/IRS_B_1-1_PAN.png');
cf = [1,2,1;2,4,2;1,2,1];
filtro(im, cf, 1);
```





Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

Ejercicio 3.4

Escriba una función que implemente un filtro de mediana. En este caso filtre también los píxeles con $ND = 0$. Compruebe su funcionamiento sobre la imagen WorldView. Repita el ejercicio mejorando el proceso para que sólo se filtren aquellos píxeles cuyo ND sea muy diferente del de su entorno (establezca un criterio adecuado para ello).

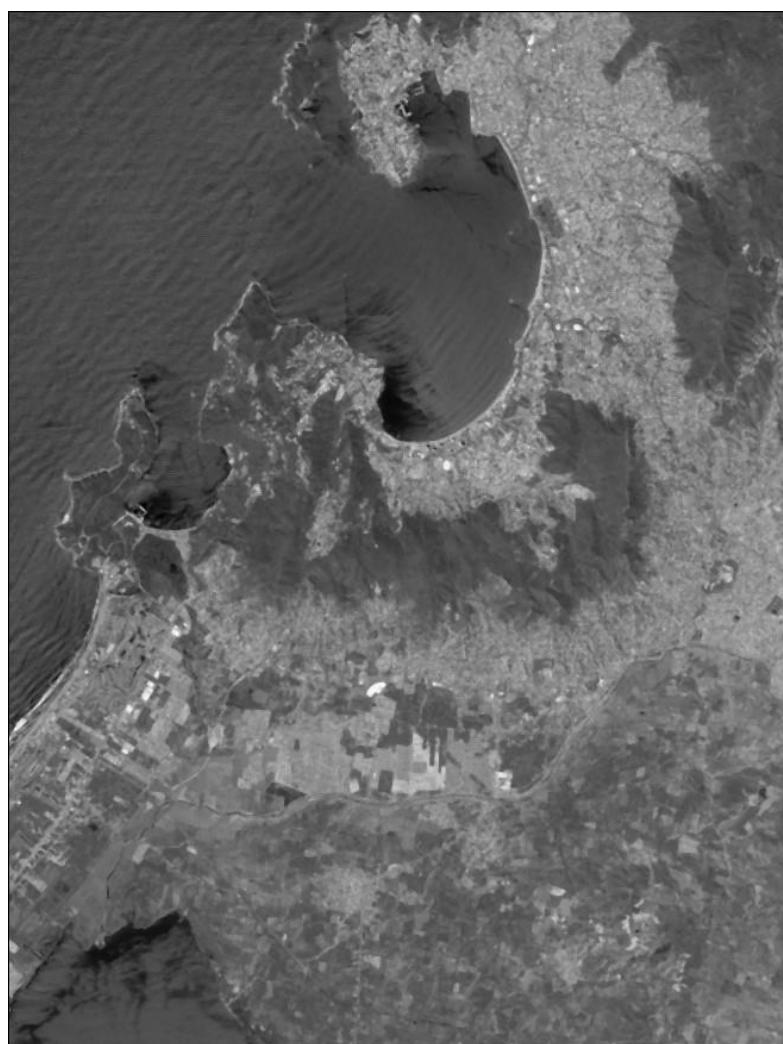


Imagen WorldView tras aplicar filtro de mediana a nivel global.



Imagen WorldView tras aplicar filtro de mediana sólo a píxeles con ND muy diferente a los de su propio entorno.

El código de la función es el siguiente:

```

function im2 = mediana(im1, b)
im1 = im1(:,:,b); %Se indica la banda a extraer para que sea monobanda
[F, C] = size(im1);
im2 = uint8(zeros(F, C));
for f = 2:F-1 %Los bordes de la imagen no se tienen en cuenta, así que se excluyen
    for c = 2:C-1
        en = double(im1(f-1:f+1, c-1:c+1));
        ndp = median(reshape(en, 1, 9));
        im2(f, c) = ndp;
    end
end
imshow(im2);

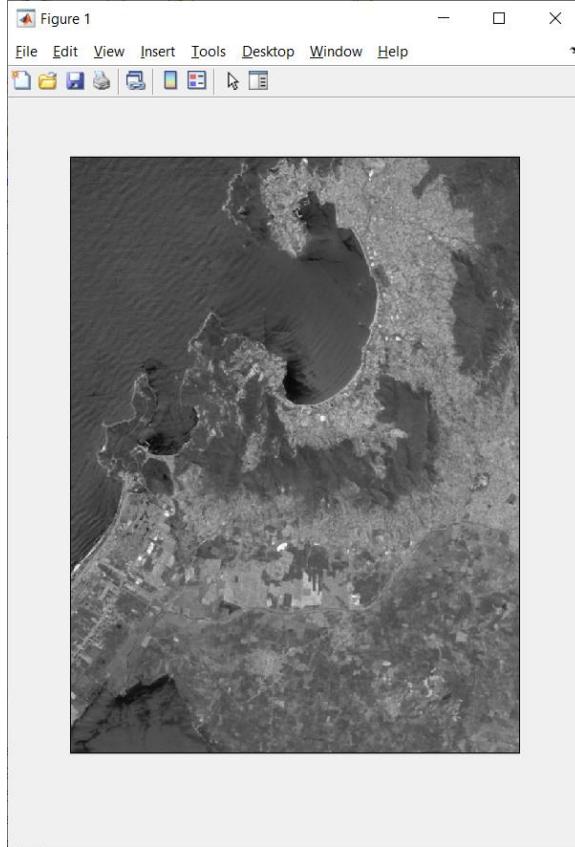
```

Tras ejecutar los siguientes comandos conseguimos la imagen resultante:

```

im = imread('imágenes/WorldView_1-1_PAN.png');
mediana(im, 1);

```

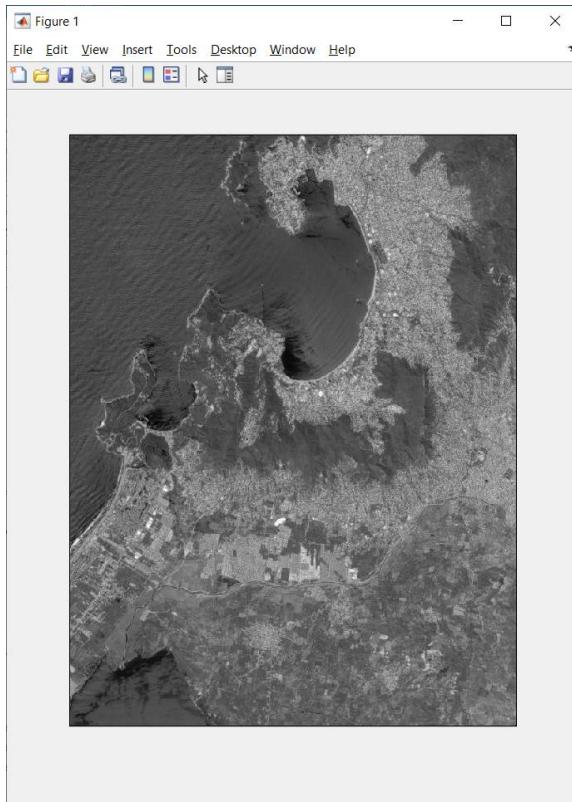


Modificamos el código de la función para que se aplique el filtro de mediana sólo a píxeles con ND muy diferente a los de su propio entorno. El criterio que seguiré es que haya una diferencia de al menos un valor de 50 entre la mediana y su valor del entorno máximo o mínimo:

```

function im2 = mediana(im1, b)
im1 = im1(:,:,b); %Se indica la banda a extraer para que sea monobanda
[F, C] = size(im1);
im2 = uint8(zeros(F, C));
for f = 2:F-1 %Los bordes de la imagen no se tienen en cuenta, así que se excluyen
    for c = 2:C-1
        en = double(im1(f-1:f+1, c-1:c+1));
        maximo = max(en(:));
        minimo = min(en(:));
        ndp = median(reshape(en, 1, 9));
        if ndp - minimo >= 50 || maximo - ndp >= 50
            im2(f, c) = ndp;
        else
            im2(f, c) = im1(f, c);
        end
    end
end
imshow(im2);

```





Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

Ejercicio 3.5

Escriba una función que implemente un filtro de detección de bordes de Sobel.
Comprueba su funcionamiento sobre la imagen IRS B.



Imagen IRS B tras aplicar filtro de Sobel.

MEMORIA:

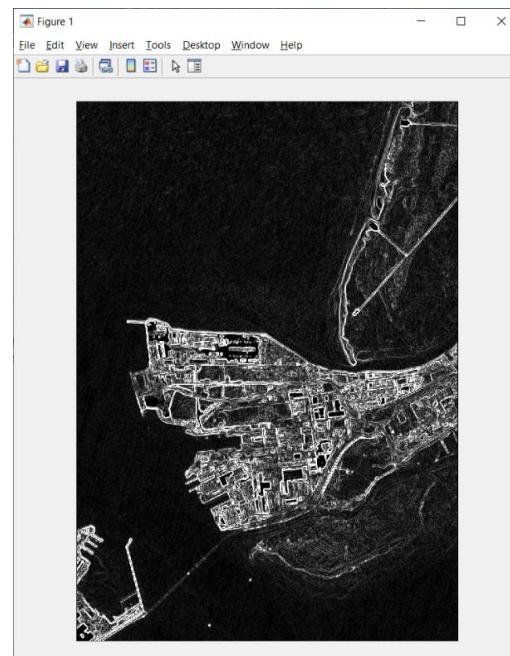
- Código 3.3 + Imagen propia tratada con filtro de media (antes/después)
- Código 3.4 mejorado + Imagen propia tratada (antes/después)

El código de la función es el siguiente:

```
function im2 = sobel(im1, b)
im1 = im1(:,:,b); %Se indica la banda a extraer para que sea monobanda
[F, C] = size(im1);
im2 = uint8(zeros(F, C));
for f = 2:F-1 %Los bordes de la imagen no se tienen en cuenta, así que se excluyen
    for c = 2:C-1
        nd = im1(f,c);
        if nd > 0
            en = double(im1(f-1:f+1, c-1:c+1));
            z0 = en(1,1);
            z1 = en(1,2);
            z2 = en(1,3);
            z3 = en(2,1);
            z5 = en(2,3);
            z6 = en(3,1);
            z7 = en(3,2);
            z8 = en(3,3);
            SC = (z2 + 2*z5 + z8) - (z0 + 2*z3 + z6);
            SF = (z0 + 2*z1 + z2) - (z6 + 2*z7 + z8);
            ndp = sqrt(SC^2 + SF^2);
            im2(f, c) = ndp;
        end
    end
end
imshow(im2);
```

Tras ejecutar los siguientes comandos conseguimos la imagen resultante:

```
im = imread('imágenes/IRS_B_1-1_PAN.png');
sobel(im, 1);
```



LAB4 - Escalado

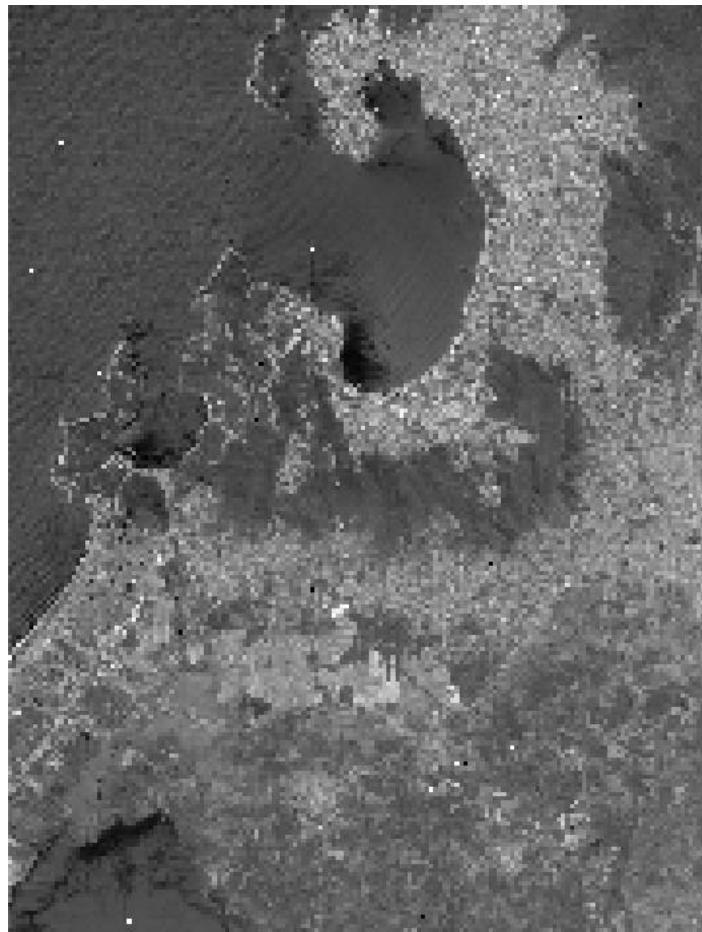
Esta sesión se centra en el escalado de imágenes abarcando tanto procedimientos de reducción como de ampliación. Al realizar los ejercicios tenga en cuenta los siguientes aspectos:

- Todas las funciones de escalado deben basar su comportamiento en un factor de escalado R que establece la relación entre el tamaño original y el nuevo. Es decir, para $R = 2,0$ la nueva imagen debe tener el doble de filas y el doble de columnas mientras que para $R = 0,50$ tendría la mitad de cada una de ellas.
- Todas las funciones de tratamiento deben visualizar en pantalla la imagen resultante al finalizar.

Ejercicio 4.1

Escriba una función llamada *diezmado* que realice la reducción de una imagen mediante dicha técnica. La función deberá elegir el nivel de diezmado más apropiado de acuerdo al factor de escalado R recibido. Los valores válidos para R son:

R	Consevar (diezmar el resto)
0,50	1 de cada 2 filas/columnas
0,67	1 de cada 3 filas/columnas
0,75	1 de cada 4 filas/columnas



Reducción de imagen WorldView mediante diezmado ($R = 0,75$).



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the
App Store

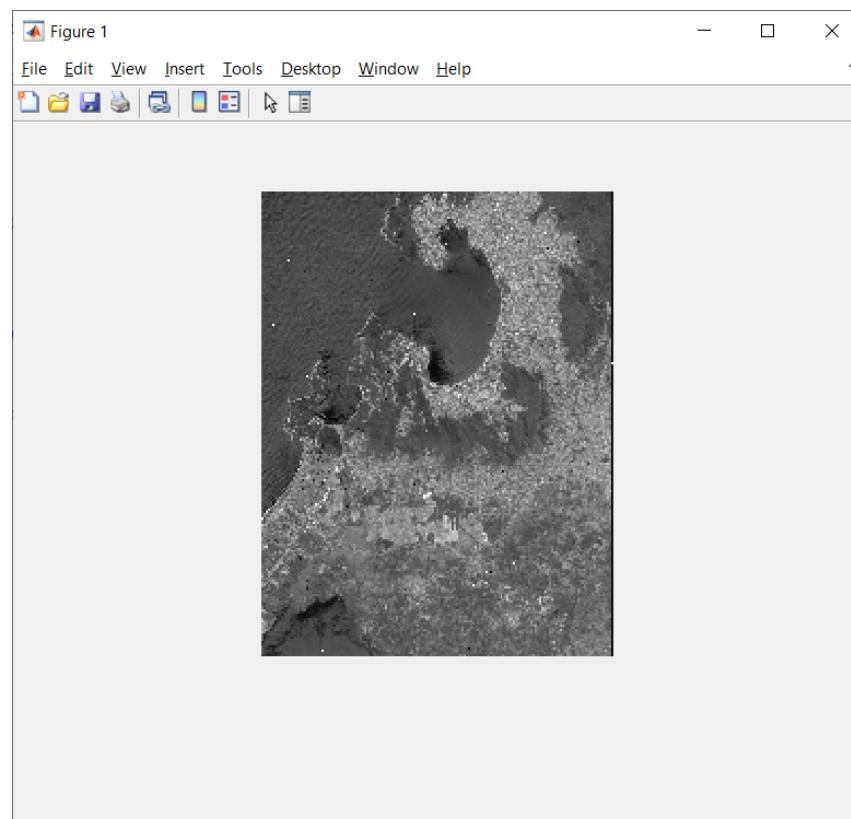
GET IT ON
Google Play

El código de la función es el siguiente:

```
function im2 = diezmado(im1, R, b)
im1 = im1(:, :, b); %Se indica la banda a extraer para que sea monobanda
[F, C] = size(im1);
if R > 0 && R <= 1
    if R == 1
        r = 1;
    else
        r = round(1 / (1 - R));
    end
    im2 = im1(1:r:F, 1:r:C);
    imshow(im2);
else
    error('El valor insertado de R no es válido, inserte uno mayor a 0 hasta 1');
end
```

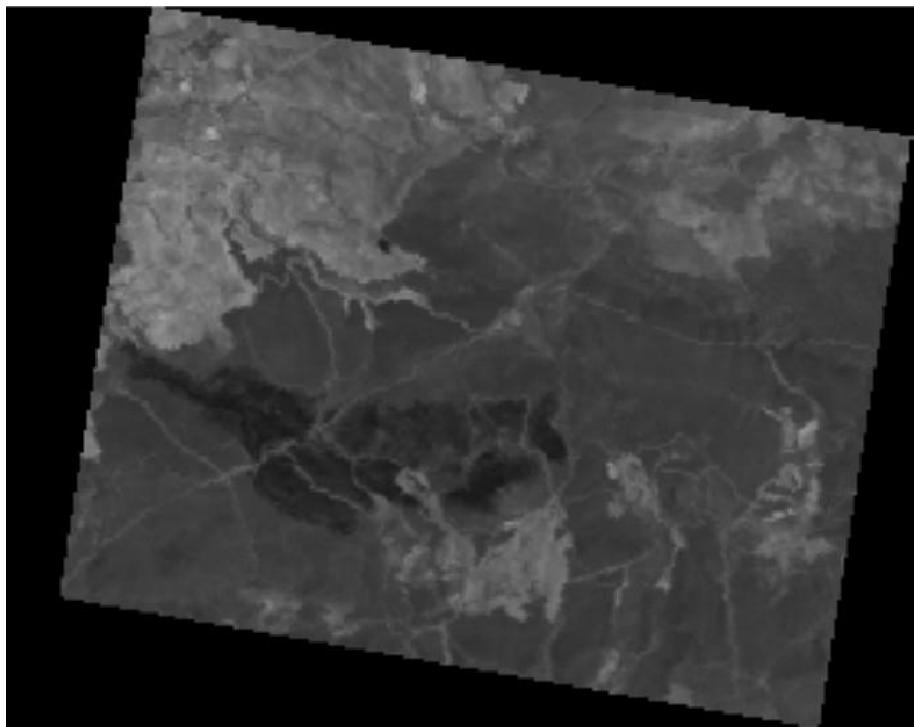
Tras ejecutar los siguientes comandos conseguimos la imagen resultante:

```
im = imread('imágenes/WorldView_1-1_PAN.png');
diezmado(im, 0.75, 1);
```



Ejercicio 4.2

Escriba una función llamada *amplia* que realice la ampliación de una imagen mediante la técnica de interpolación bilineal. Suponga un factor de escalado $R = 2$ constante. Es decir, la imagen resultante debe tener el doble de filas y el doble de columnas.



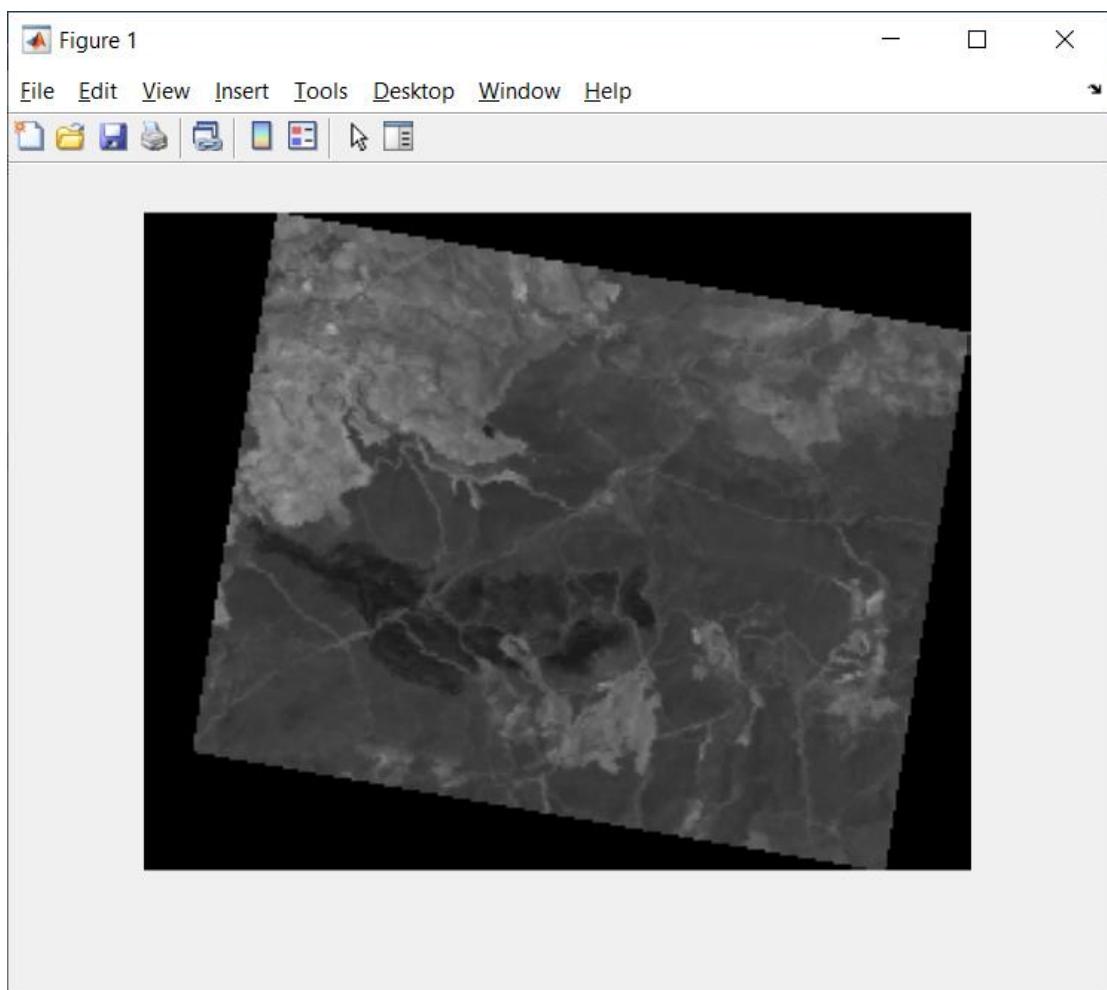
Ampliación de imagen Landsat B (banda 4) mediante interpolación bilineal.

El código de la función es el siguiente:

```
function im2 = amplia(im1, R, b)
im1 = im1(:,:,b); %Se indica la banda a extraer para que sea monobanda
[F, C] = size(im1);
if R >= 1
    Fr = F * R;
    Cr = C * R;
else
    error('El valor insertado de R no es válido, inserte uno mayor a 1');
end
im2 = zeros(Fr, Cr);
im2(1:R:Fr, 1:R:Cr) = im1;
for f = 1:R:Fr
    for c = 2:(Cr-(R-1))
        if mod(c, R) == 0
            nd1 = im2(f, c-(R-1)); %Pixel de la izquierda con valor
            nd2 = im2(f, c+1);      %Pixel de la derecha con valor
            ndp = (nd1 + nd2) / 2; %Media de ambos valores para rellenar
            for e = c-(R-2):c
                im2(f, e) = ndp;
            end
        end
    end
end
for f = 2:(Fr-(R-1))
    if mod(f, R) == 0
        f1 = im2(f -(R-1), :);
        f2 = im2(f + 1, :);
        fnd = (f1 + f2) / 2;
        for e = f-(R-2):f
            im2(e, :) = fnd;
        end
    end
end
%Finalmente rellenamos las líneas negras finales con la última línea buena:
for e = Fr:-1:Fr-(R-1)
    im2(:, e) = im2(:, Fr-R, :);
end
for e = Cr:-1:Cr-(R-1)
    im2(:, e) = im2(:, Cr-R);
end
im2 = uint8(im2);
imshow(im2);
```

Tras ejecutar los siguientes comandos conseguimos la imagen resultante:

```
im = imread('imágenes/Landsat_B_4-6_NIR.png');
R = 2;
amplia(im, R, 1);
```



Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.



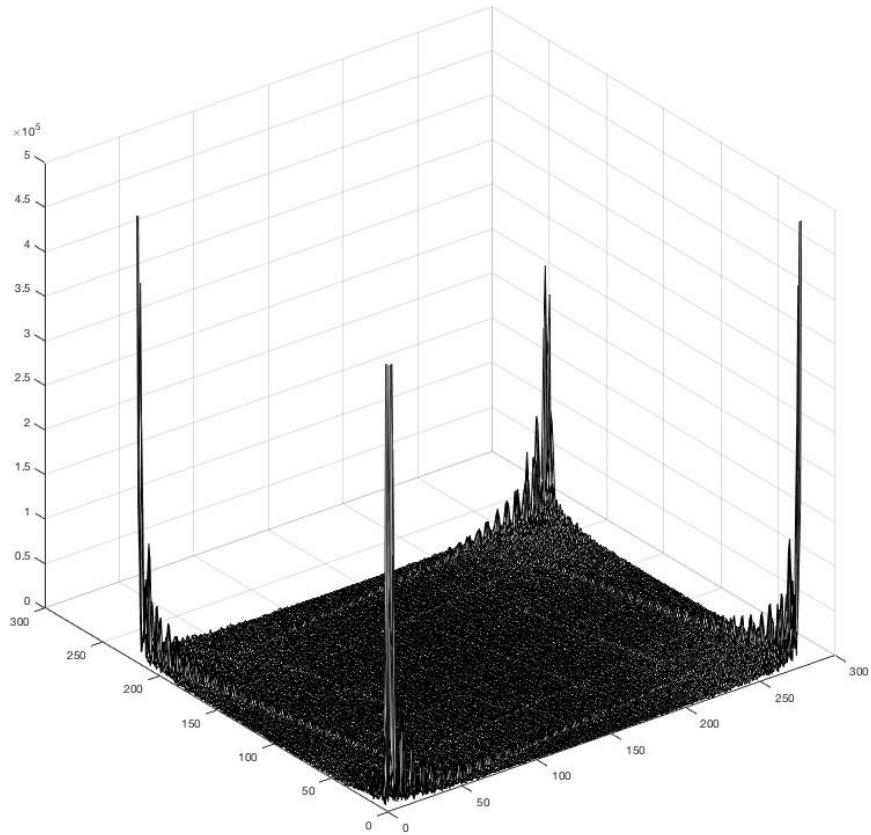
Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.

Available on the
App Store

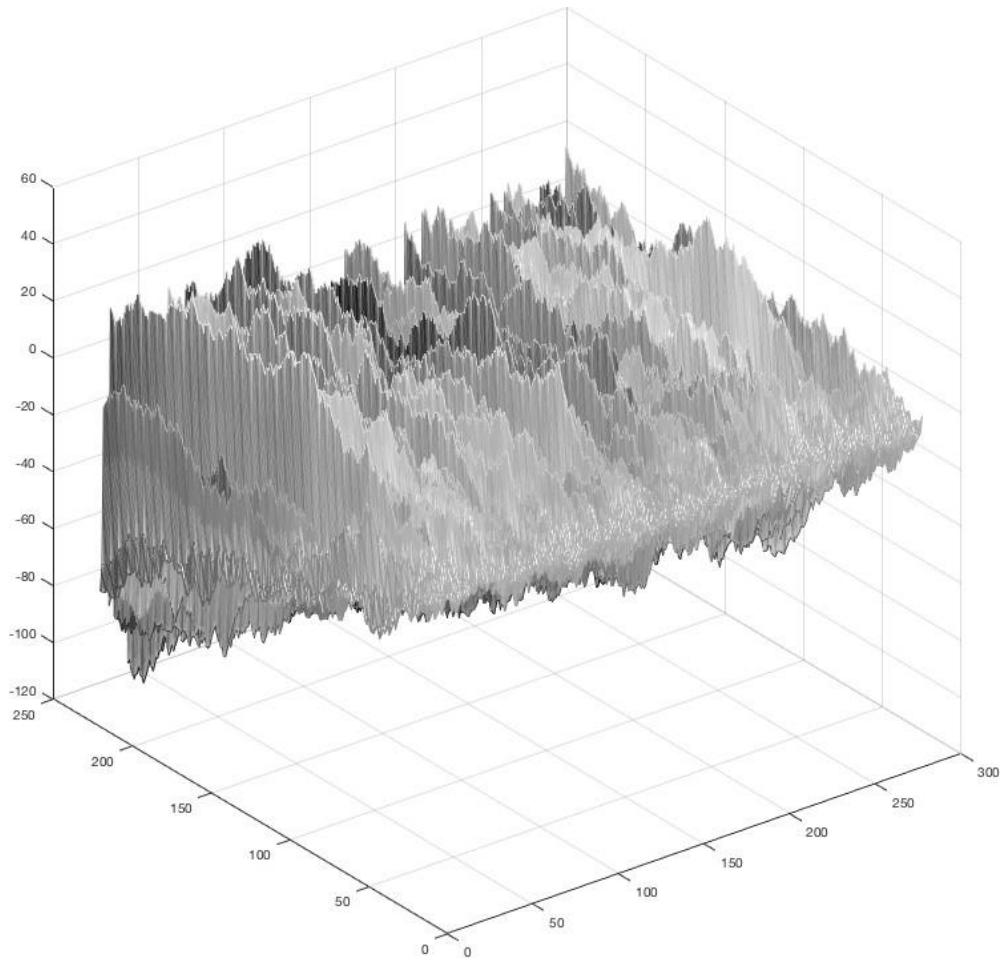
GET IT ON
Google Play

Ejercicio 4.3

Utilizando la función `fft2` visualice la FFT (*Fast Fourier Transform*) bidimensional de la imagen Landsat B (banda 4). Genere dos figuras diferentes mediante la función `mesh`: una para el módulo de la transformada (extráigalo mediante la función `abs`) y otra para la fase (extráigala mediante la función `angle`). Es aconsejable utilizar la función `axis` para ajustar los ejes de las figuras así como la función `unwrap` para desenvolver los datos de fase.



Módulo de la FFT bidimensional de la imagen Landsat B (banda 4).



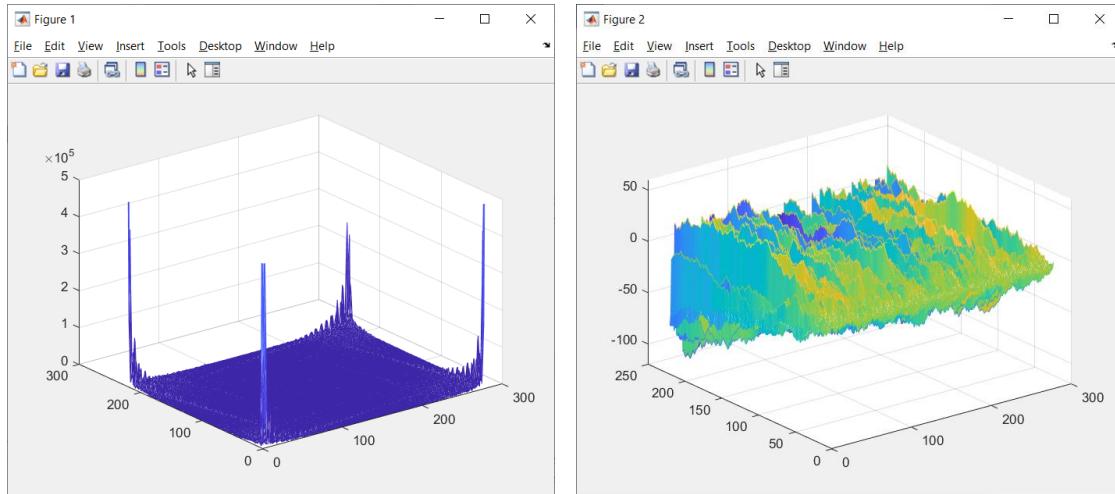
Fase de la FFT bidimensional de la imagen Landsat B (banda 4).

El código de la función es el siguiente:

```
function IM = transFourierBidim(im1, b)
im1 = im1(:,:,b); %Se indica la banda a extraer para que sea monobanda
IM = fft2(im1);
figure(1);
mesh(abs(IM));
axis([0 300 0 300 0 5*10^5]);
figure(2);
mesh(unwrap(angle(IM)));
axis([0 300 0 250 -120 60]);
%La función unwrap analiza hacia donde van creciendo
%los valores del ángulo. Si va creciendo o decreciendo se va sumando o
%restando 2pi respectivamente
```

Tras ejecutar los siguientes comandos conseguimos las figuras resultantes:

```
im = imread('imágenes/Landsat_B_4-6_NIR.png');  
transFourierBidim(im, 1);
```



Ejercicio 4.4

Repite el ejercicio 4.2 (ampliación de imágenes) pero utilizando la técnica de superresolución basada en FFT y para un factor de escalado R indeterminado, aunque superior a la unidad, y que se establece como parámetro de entrada. Llame a la función *ampliafft*. Compare los resultados para $R = 8$ con los obtenidos mediante la función *amplia* (aplíquela repetidamente).

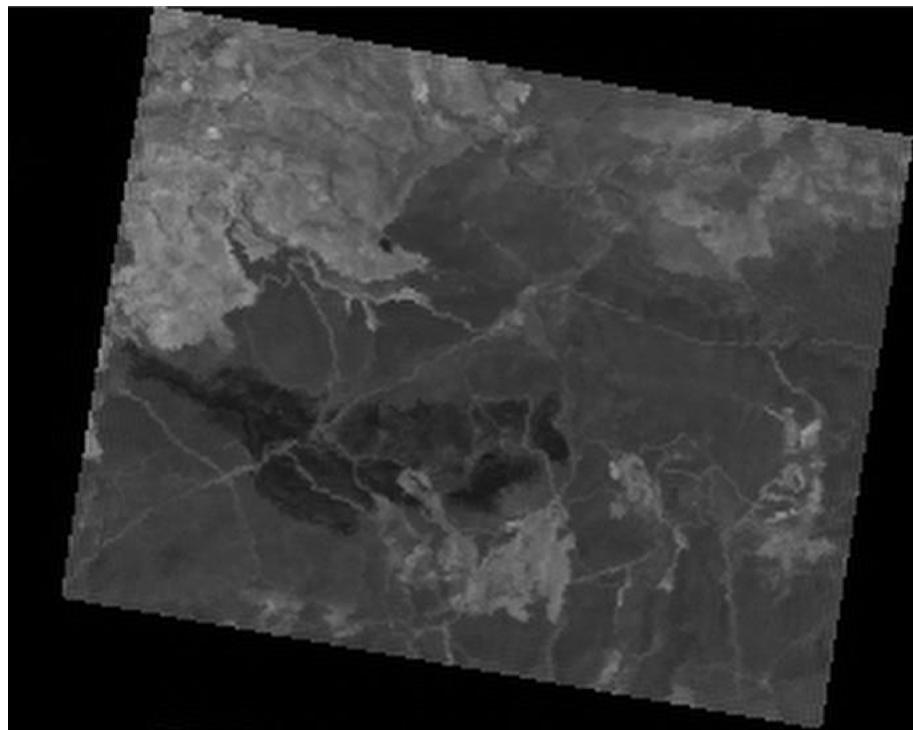


Imagen Landsat B (banda 4) ampliada mediante FFT ($R = 8$).

MEMORIA:

- Código 4.2 + Imagen propia tratada (antes/después)
- Código 4.4 + Imagen propia tratada (antes/después)



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the
App Store

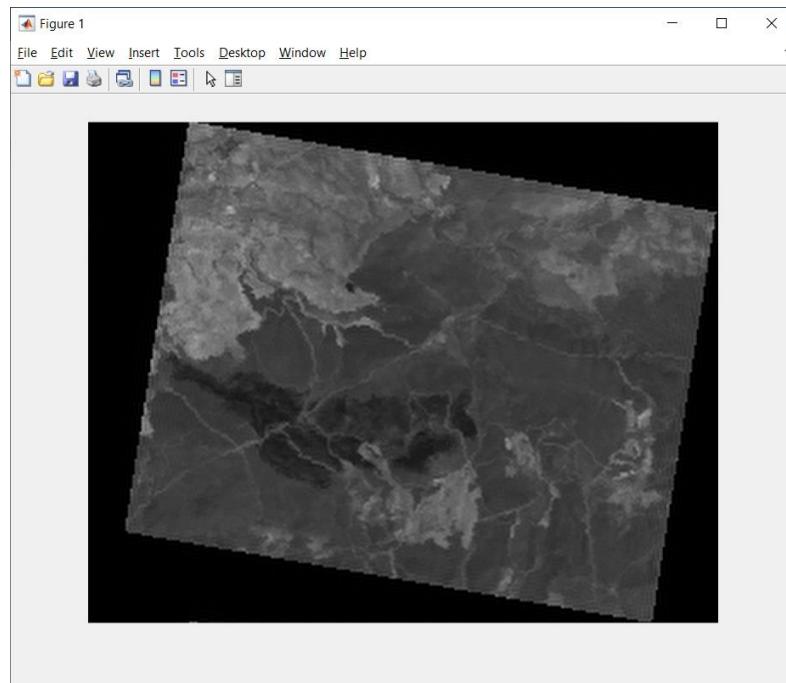
GET IT ON
Google Play

El código de la función es el siguiente:

```
function im2 = ampliafft(im1, R, b)
im1 = im1(:, :, b); %Se indica la banda a extraer para que sea monobanda
[F, C] = size(im1);
if R > 1
    Fr = F * R;
    Cr = C * R;
else
    error('El valor insertado de R no es válido, inserte uno mayor a 1');
end
IM1 = fft2(im1);
IM2 = zeros(Fr, Cr);
IM2(1:F/2, 1:C/2) = IM1(1:F/2, 1:C/2); %Primer cuadrante
IM2(Fr-(F/2)+1:Fr, 1:C/2) = IM1((F/2)+1:F, 1:C/2); %Segundo cuadrante
IM2(1:F/2, Cr-(C/2)+1:Cr) = IM1(1:(F/2), (C/2)+1:C); %Tercer cuadrante
IM2(Fr-(F/2)+1:Fr, Cr-(C/2)+1:Cr) = IM1((F/2)+1:F, (C/2)+1:C); %Cuarto cuadrante
im2 = real(ifft2(IM2)) * ((Fr*Cr)/(F*C)); %Recuperamos la imagen adecuando su amplitud
im2 = uint8(im2);
imshow(im2);
```

Tras ejecutar los siguientes comandos conseguimos la imagen resultante:

```
im = imread('imágenes/Landsat_B_4-6_NIR.png');
R = 8;
ampliafft(im, R, 1);
```



Si comparamos la imagen resultante con la de la función amplia, observamos que la de ampliafft tiene mejores resultados, la otra se ve más borrosa.

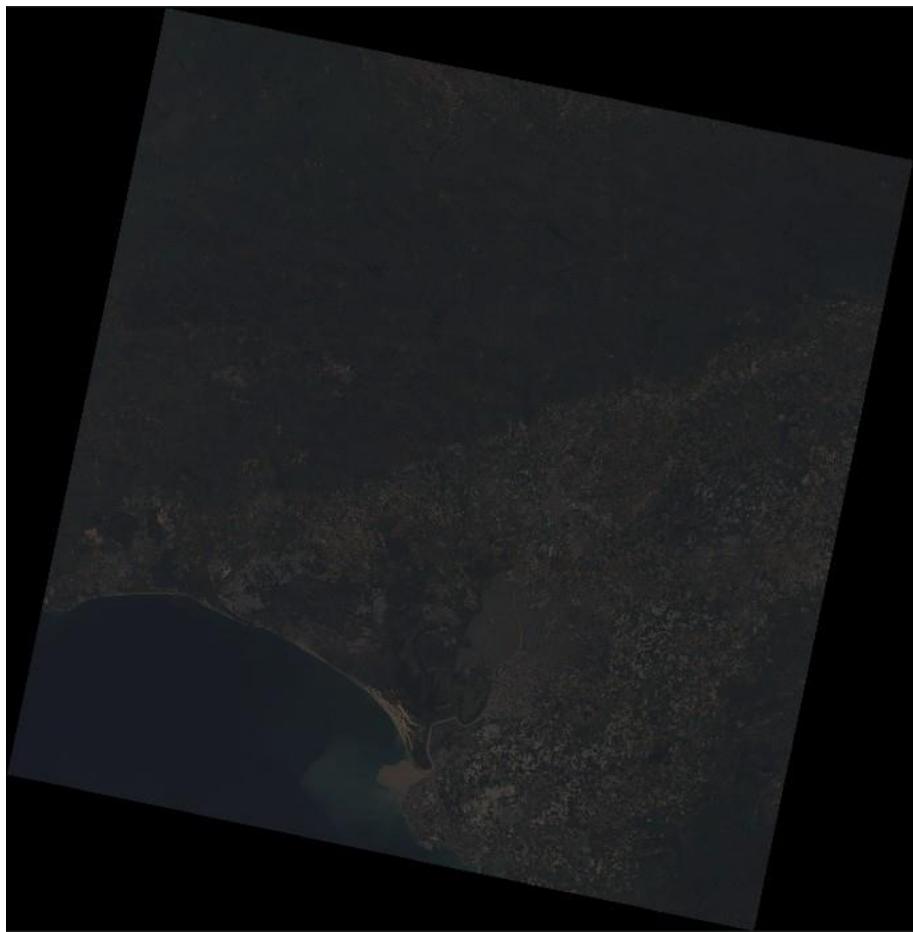
LAB5 - Color

Esta sesión se centra en las composiciones y transformaciones a color. Al realizar los ejercicios tenga en cuenta los siguientes aspectos:

- Trabaje únicamente en el rango [1, 255] (el valor 0 se reserva para *ND* inválido).
- Todas las funciones de tratamiento deben visualizar en pantalla la imagen resultante al finalizar.

Ejercicio 5.1

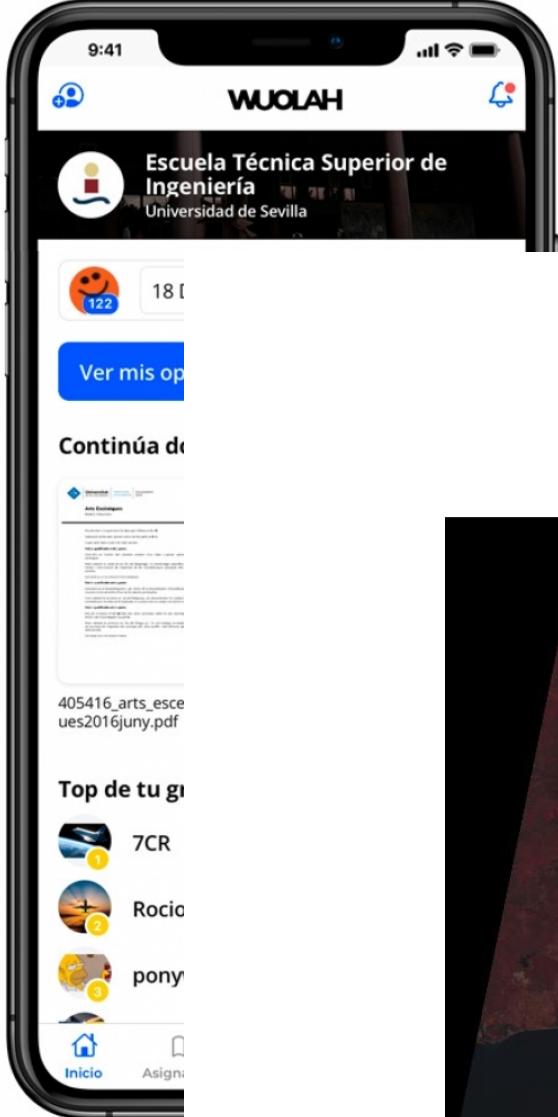
Escriba una función llamada *falso* que, a partir de 3 imágenes monobanda recibidas como entrada, las visualice en pantalla como una combinación en color (cada una de ellas debe considerarlas como uno de los colores primarios RGB). Compruebe los resultados sobre la imagen Landsat C generando, en concreto, una composición en color verdadero y otra en falso color típico NIR-G-R. A continuación, repita el ejercicio realizando un corte de colas al 1 % a cada banda antes de llevar a cabo la composición.



Composición en color verdadero de imagen Landsat C sin corte de colas previo.



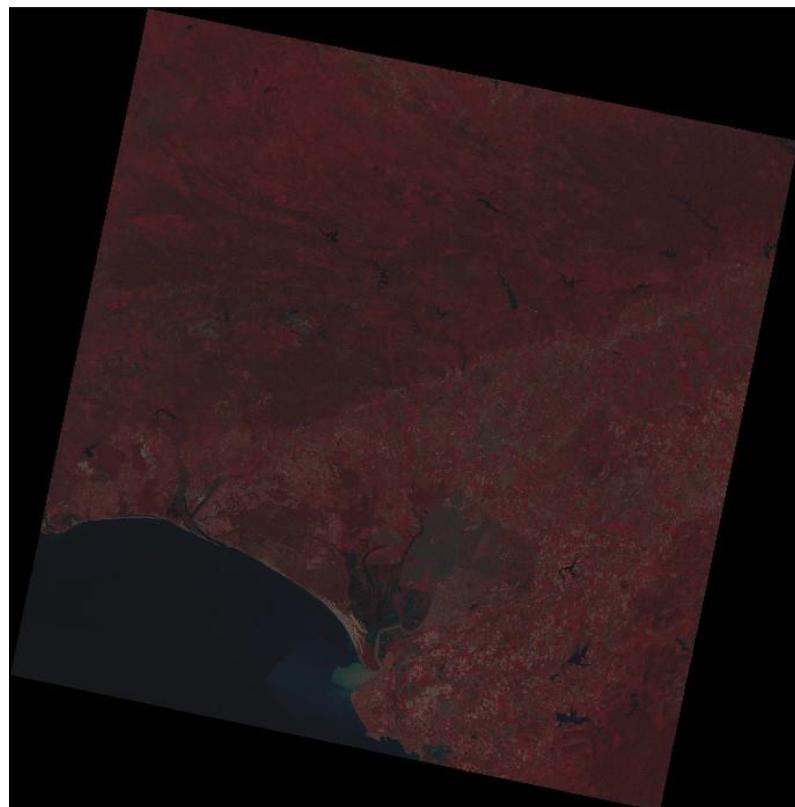
Composición en color verdadero de imagen Landsat C con corte de colas previo.



Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play



Composición en falso color (NIR-G-R) de imagen Landsat C sin corte de colas previo.



Composición en falso color (NIR-G-R) de imagen Landsat C con corte de colas previo.

El código de la función sin corte de colas es el siguiente:

```
function im2 = falso(imR, imG, imB, imNIR)
figure(1);
im2(:,:,1) = imR;
im2(:,:,2) = imG;
im2(:,:,3) = imB;
imshow(im2);
figure(2);
im2(:,:,1) = imNIR;
im2(:,:,2) = imR;
im2(:,:,3) = imG;
imshow(im2);
```

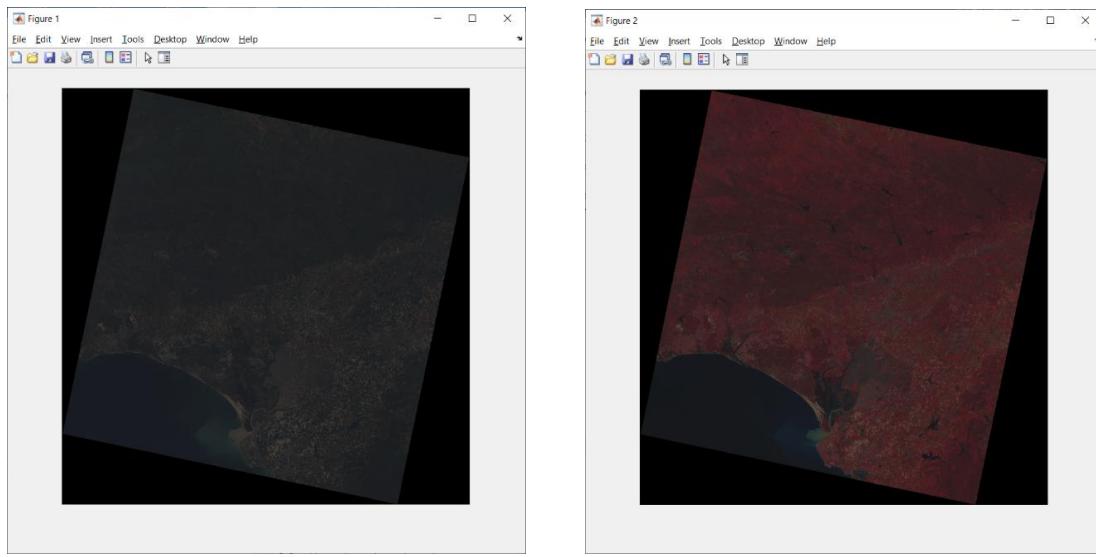
Tras ejecutar los siguientes comandos conseguimos las imágenes resultantes:

```
imR = imread('imágenes/Landsat_C_04-11_R.png');
imG = imread('imágenes/Landsat_C_03-11_G.png');
```

```

imB = imread('imágenes/Landsat_C_02-11_B.png');
imNIR = imread('imágenes/Landsat_C_05-11_NIR.png');
falso(imR, imG, imB, imNIR);

```



El código de la función con corte de colas es el siguiente:

```

function im2 = falso(imR, imG, imB, imNIR)
figure(1);
im2(:,:,:,1) = corte(imR, 1, 1); %corte(imagen, porcentaje, banda)
im2(:,:,:,2) = corte(imG, 1, 1);
im2(:,:,:,3) = corte(imB, 1, 1);
imshow(im2);
figure(2);
im2(:,:,:,1) = corte(imNIR, 1, 1);
im2(:,:,:,2) = corte(imR, 1, 1);
im2(:,:,:,3) = corte(imG, 1, 1);
imshow(im2);

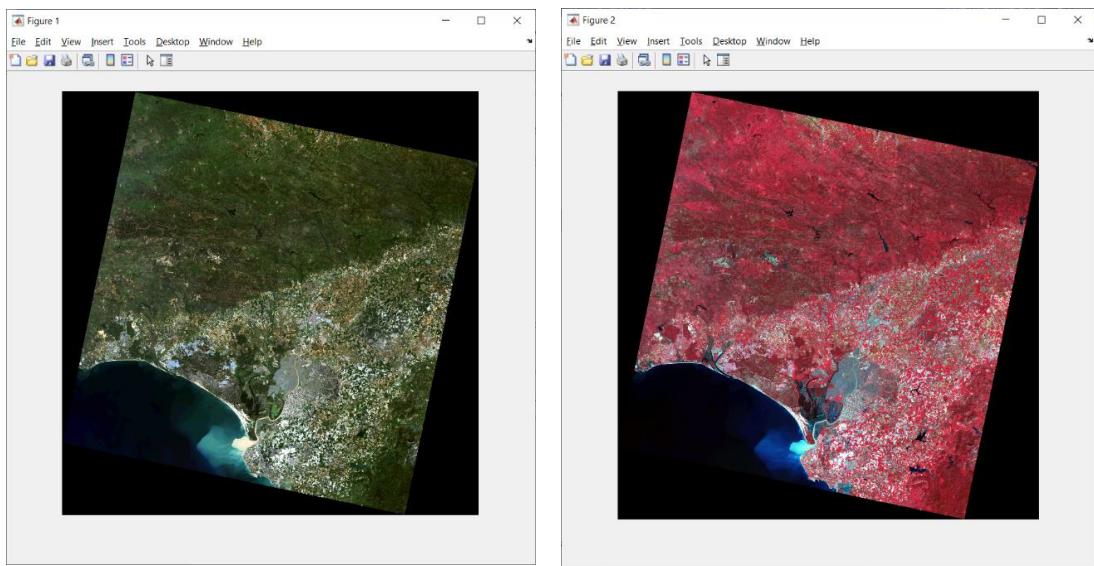
```

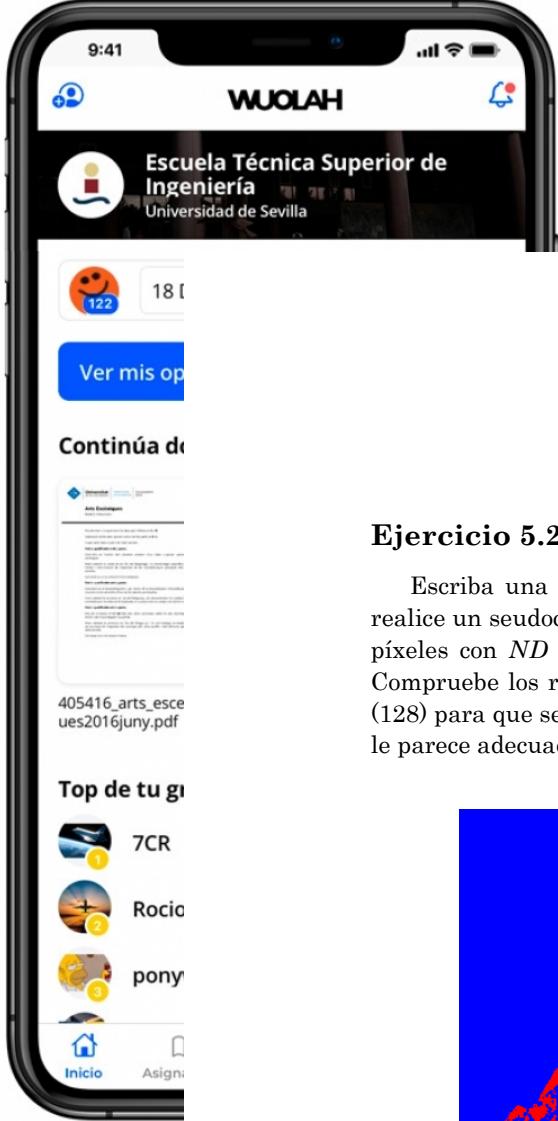
Tras ejecutar los siguientes comandos para un 1% conseguimos las imágenes resultantes:

```

imR = imread('imágenes/Landsat_C_04-11_R.png');
imG = imread('imágenes/Landsat_C_03-11_G.png');
imB = imread('imágenes/Landsat_C_02-11_B.png');
imNIR = imread('imágenes/Landsat_C_05-11_NIR.png');
falso(imR, imG, imB, imNIR);

```





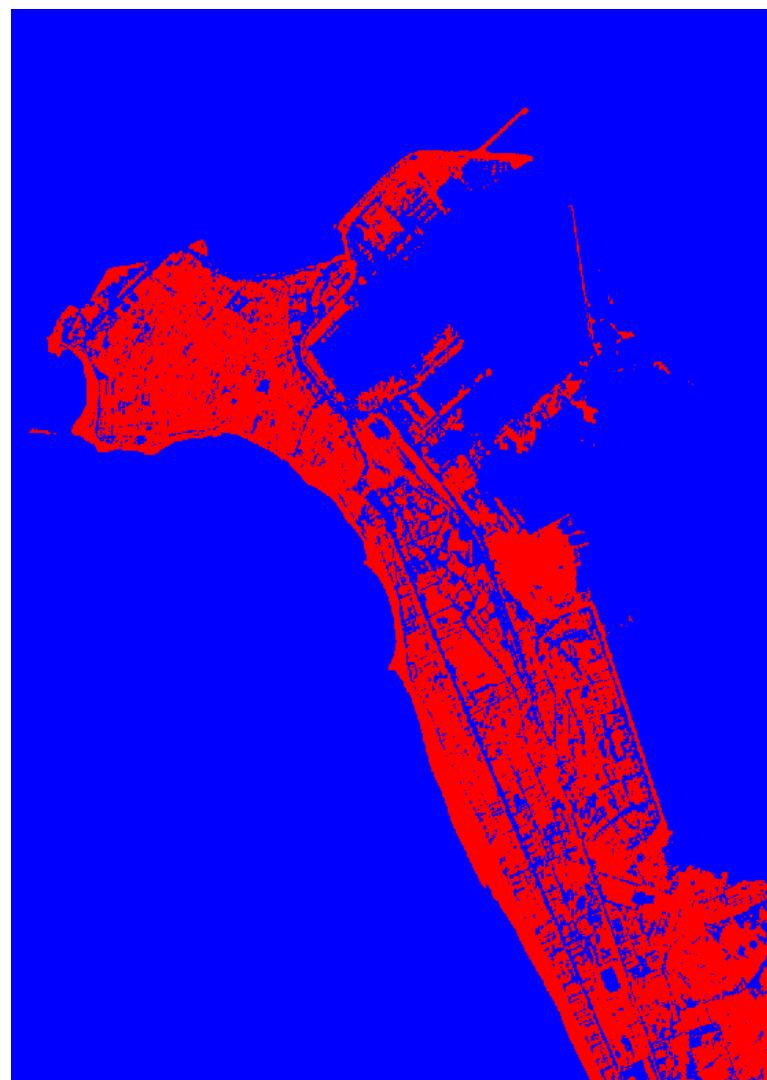
Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

Ejercicio 5.2

Escriba una función llamada *seudo* que, a partir de una imagen monobanda, realice un seudocolor basado en el *ND*: aplique un color (de su elección) a todos los píxeles con $ND < 128$ y otro color (de su elección) a aquellos con $ND \geq 128$. Compruebe los resultados sobre la imagen IRS A. Ajuste ese nivel de corte inicial (128) para que se discrimine lo mejor posible el mar de la tierra. ¿Qué valor de corte le parece adecuado?



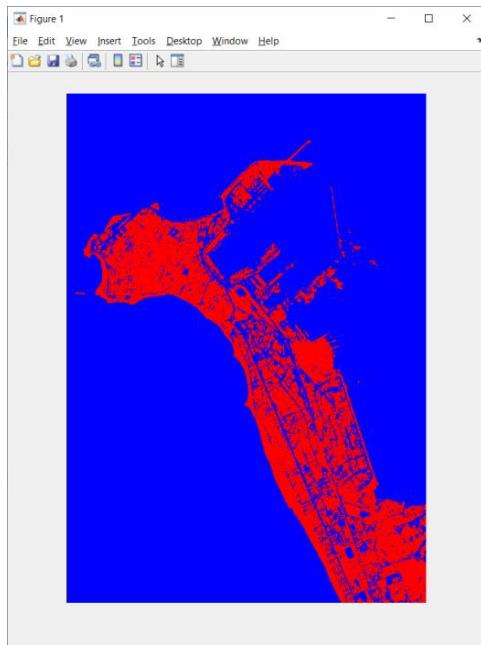
Seudocolor de imagen IRS A (Azul: < 128 , Rojo: ≥ 128).

El código de la función es el siguiente:

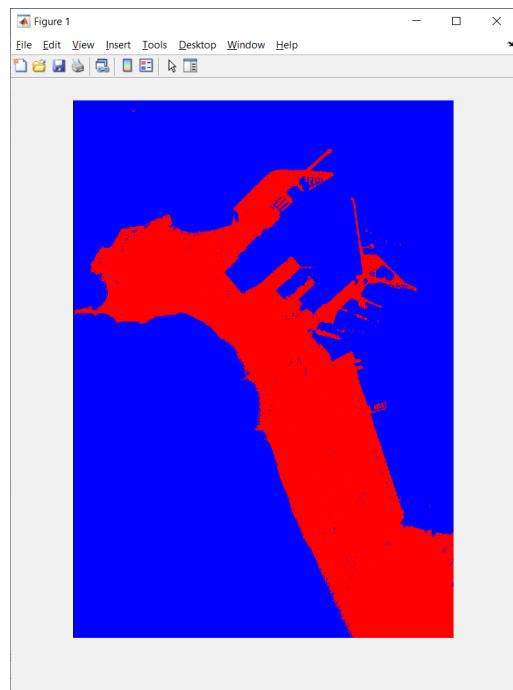
```
function im2 = seudo(im1, b)
im1 = im1(:,:,b); %Se indica la banda a extraer para que sea monobanda
[F, C] = size(im1);
COLOR1 = [0,0,255]; %Se ha elegido el azul
COLOR2 = [255,0,0]; %Se ha elegido el rojo
im2 = uint8(zeros(F, C, 3));
for f = 1:F
    for c = 1:C
        nd = im1(f,c);
        if nd > 0
            if nd < 128
                ndp = COLOR1;
            else
                ndp = COLOR2;
            end
            im2(f, c, :) = ndp;
        end
    end
end
imshow(im2);
```

Tras ejecutar los siguientes comandos conseguimos la imagen resultante:

```
im = imread('imágenes/IRS_A_1-1_PAN.png');
seudo(im, 1);
```

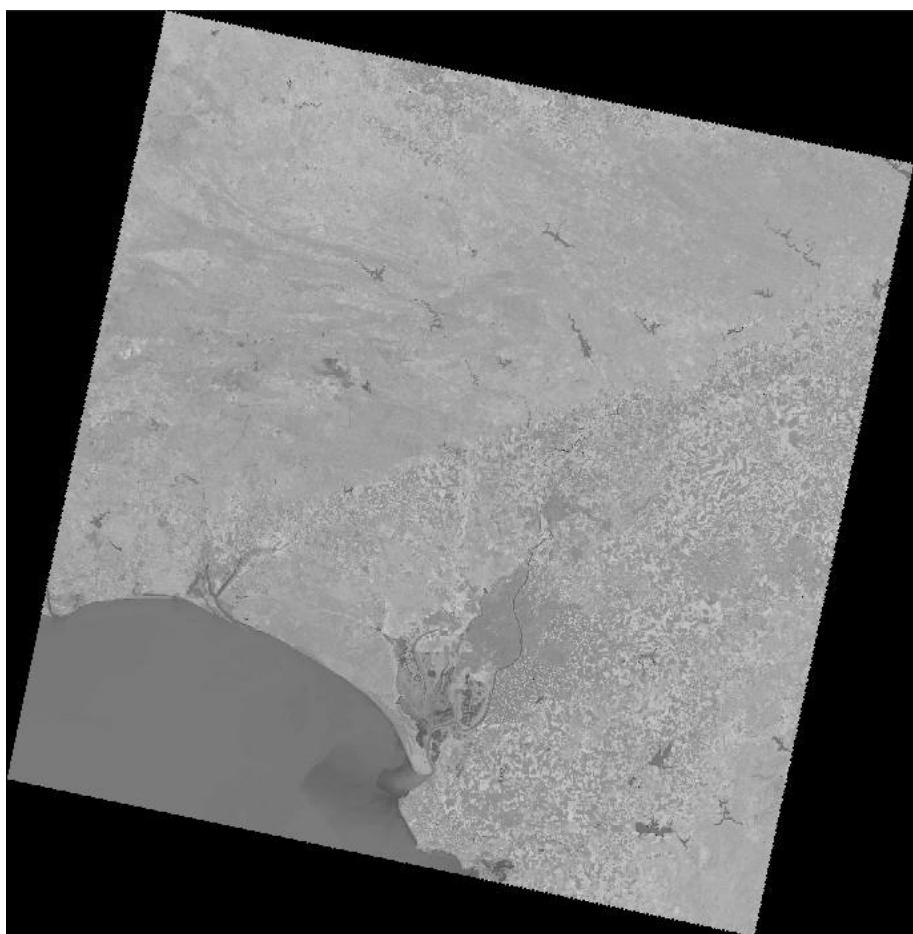


Ajustando el nivel de corte inicial a 60, se discrimina mejor el mar de la tierra:



Ejercicio 5.3

Escriba una función llamada *ndvi* que, a partir de 2 imágenes monobanda recibidas como entrada (correspondientes a R y NIR), calcule el NDVI correspondiente a cada píxel. Lleve los resultados al rango [1, 255] antes de visualizarlos y devolverlos. Compruebe los resultados sobre la imagen Landsat C.



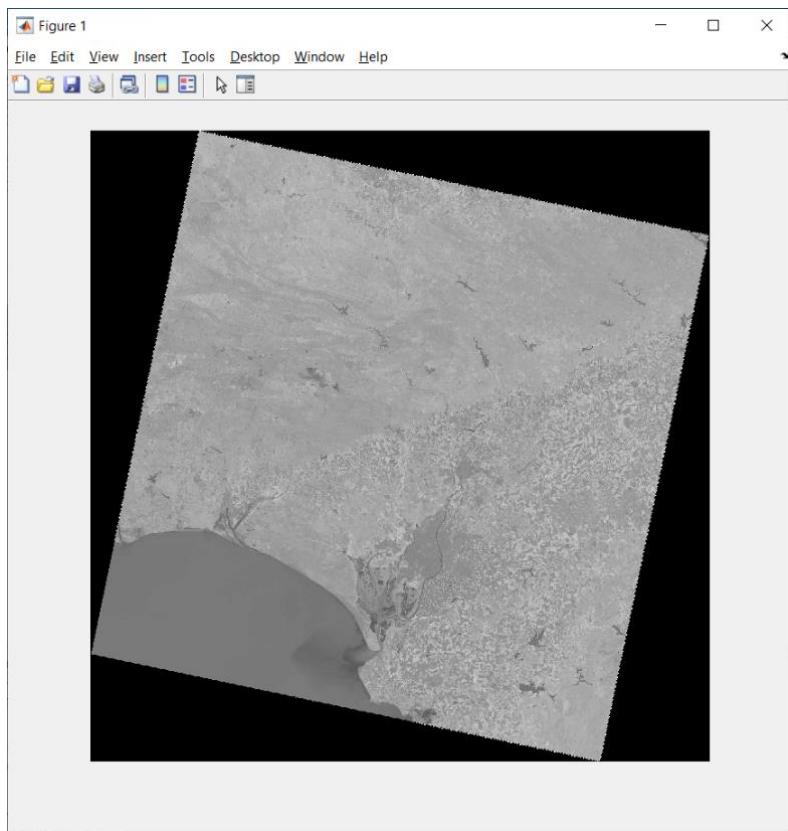
NDVI calculado a partir de imagen Landsat C y llevado a rango [1, 255].

El código de la función es el siguiente:

```
function im2 = ndvi(imR, imNIR)
imR = im2double(imR);
imNIR = im2double(imNIR);
[F, C] = size(imR);
im2 = uint8(zeros(F, C));
for f = 1:F
    for c = 1:C
        ndR = imR(f,c);
        ndNIR = imNIR(f,c);
        if ndR > 0 && ndNIR > 0
            ndp = (ndNIR - ndR) / (ndNIR + ndR);
            ndp = round(((ndp + 1) / 2) * 254 + 1);
            im2(f, c) = ndp;
        end
    end
end
imshow(im2);
```

Tras ejecutar los siguientes comandos conseguimos la imagen resultante:

```
imR = imread('imágenes/Landsat_C_04-11_R.png');
imNIR = imread('imágenes/Landsat_C_05-11_NIR.png');
ndvi(imR, imNIR);
```





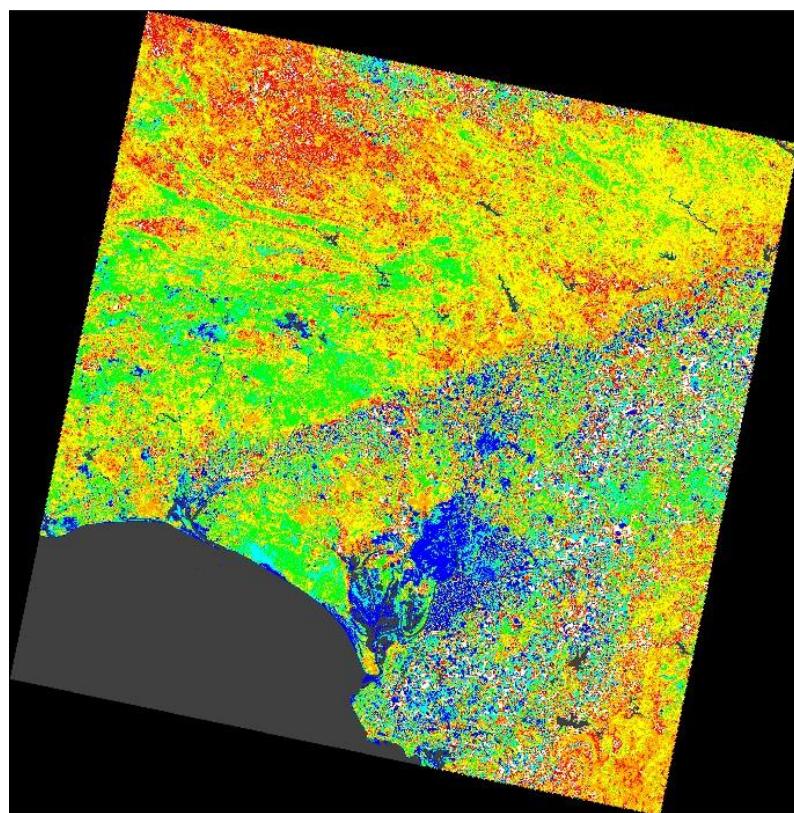
Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

Ejercicio 5.4

Escriba una función llamada *densi* que realice un *density slicing* sobre una imagen monobanda. Elija al menos 4 rangos de *ND* y aplique colores de su elección. Compruebe sus resultados sobre la imagen NDVI obtenida en el ejercicio anterior.



Density slicing sobre NDVI a partir de imagen Landsat C en rango [128, 192].

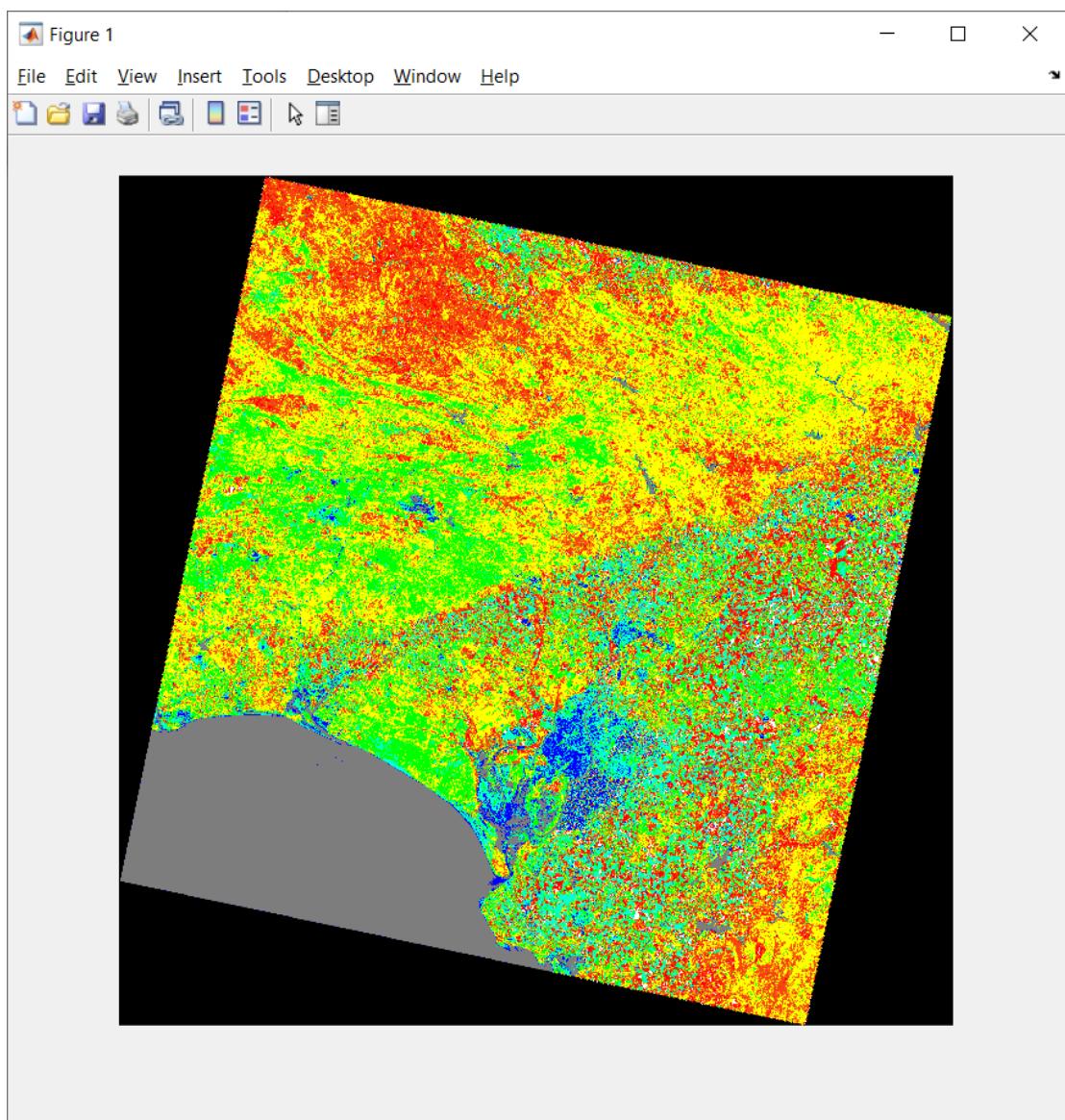
El código de la función es el siguiente:

```
function im2 = densi(im1, b)
im1 = im1(:,:,b); %Se indica la banda a extraer para que sea monobanda
maximoValor = find(histo(im1,1) > 1000, 1, 'last'); %He elegido 1000 como mínimo de valor destacable
[F, C] = size(im1);
COLOR0 = [125,125,125]; %Gris
COLOR1 = [0,0,255]; %Azul
COLOR2 = [7,255,209]; %Celeste
COLOR3 = [0,255,0]; %Verde
COLOR4 = [255,255,0]; %Amarillo
COLOR5 = [244,70,17]; %Naranja
COLOR6 = [255,0,0]; %Rojo
COLOR7 = [255,255,255]; %Blanco
numColores = 7;

im2 = uint8(zeros(F, C, 3));
for f = 1:F
    for c = 1:C
        nd = im1(f,c);
        if nd > 0
            if nd < 128
                ndp = COLOR0;
            elseif nd < ((maximoValor-128)/numColores + 128)
                ndp = COLOR1;
            elseif nd < ((maximoValor-128)/numColores*(numColores - 5) + 128)
                ndp = COLOR2;
            elseif nd < ((maximoValor-128)/numColores*(numColores - 4) + 128)
                ndp = COLOR3;
            elseif nd < ((maximoValor-128)/numColores*(numColores - 3) + 128)
                ndp = COLOR4;
            elseif nd < ((maximoValor-128)/numColores*(numColores - 2) + 128)
                ndp = COLOR5;
            elseif nd < ((maximoValor-128)/numColores*(numColores - 1) + 128)
                ndp = COLOR6;
            else
                ndp = COLOR7;
            end
            im2(f, c, :) = ndp;
        end
    end
end
imshow(im2);
```

Tras ejecutar los siguientes comandos conseguimos la imagen resultante:

```
imNIR = imread('imágenes/Landsat_C_05-11_NIR.png');
imR = imread('imágenes/Landsat_C_04-11_R.png');
densi(ndvi(imR, imNIR), 1);
```



Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Ejercicio 5.5

Escriba una función que realice un corte de colas en HSV a una determinada composición en color. Para ello, elija alguna combinación a color a partir de las imágenes disponibles y:

- Transforme la combinación (RGB) al dominio HSV.
- Realice un corte de colas de la componente S o V.
- Transforme la imagen resultante (HSV) de nuevo al dominio RGB.

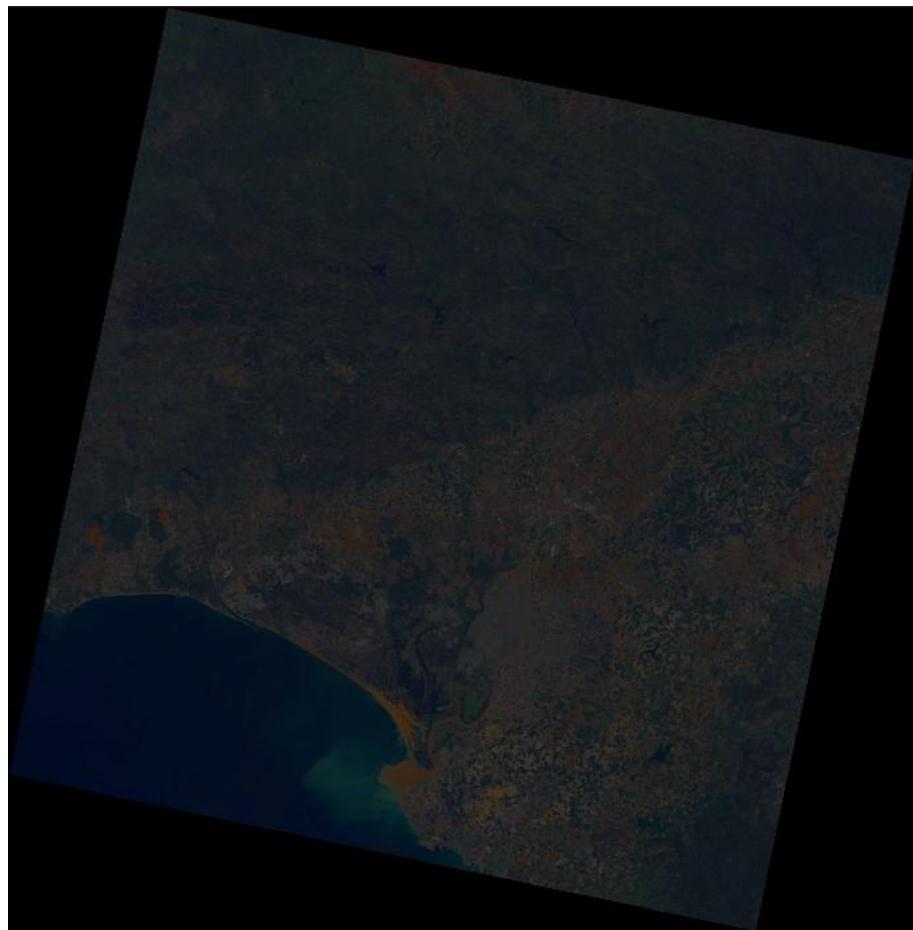
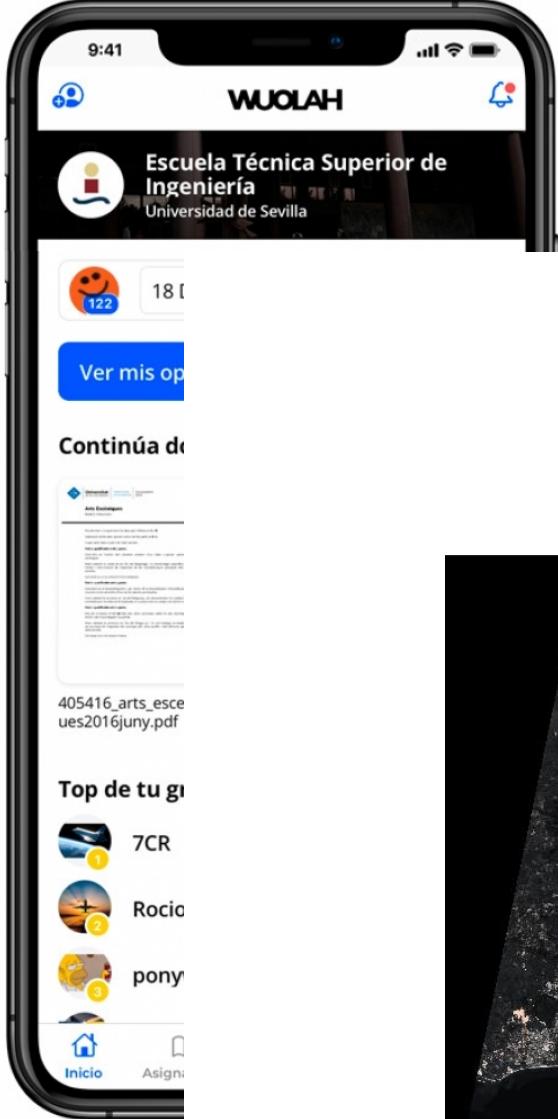


Imagen Landsat C tras corte de colas de la componente S en dominio HSV.



Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

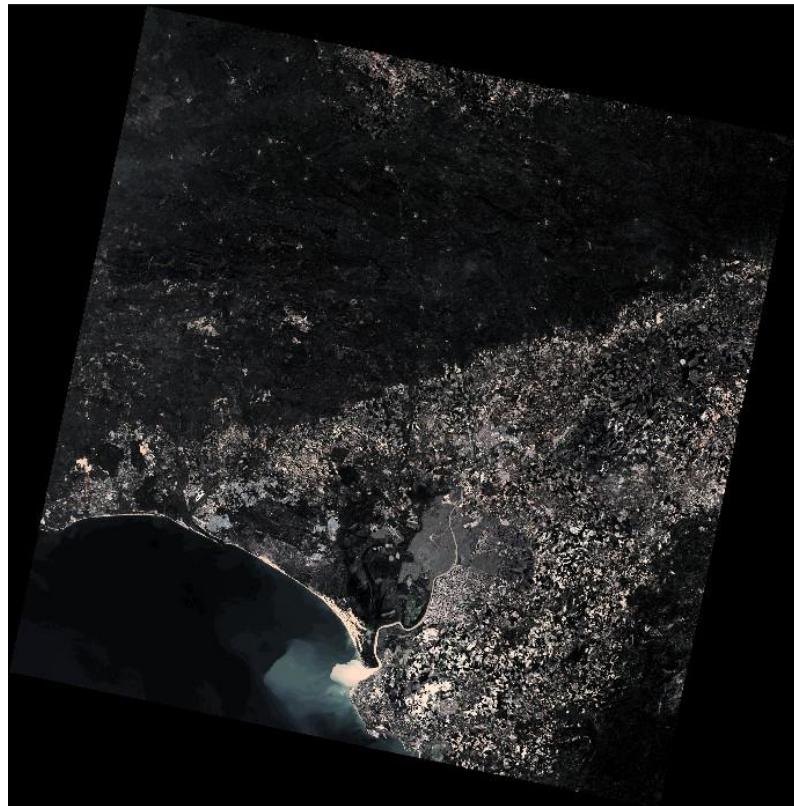


Imagen Landsat C tras corte de colas de la componente V en dominio HSV.

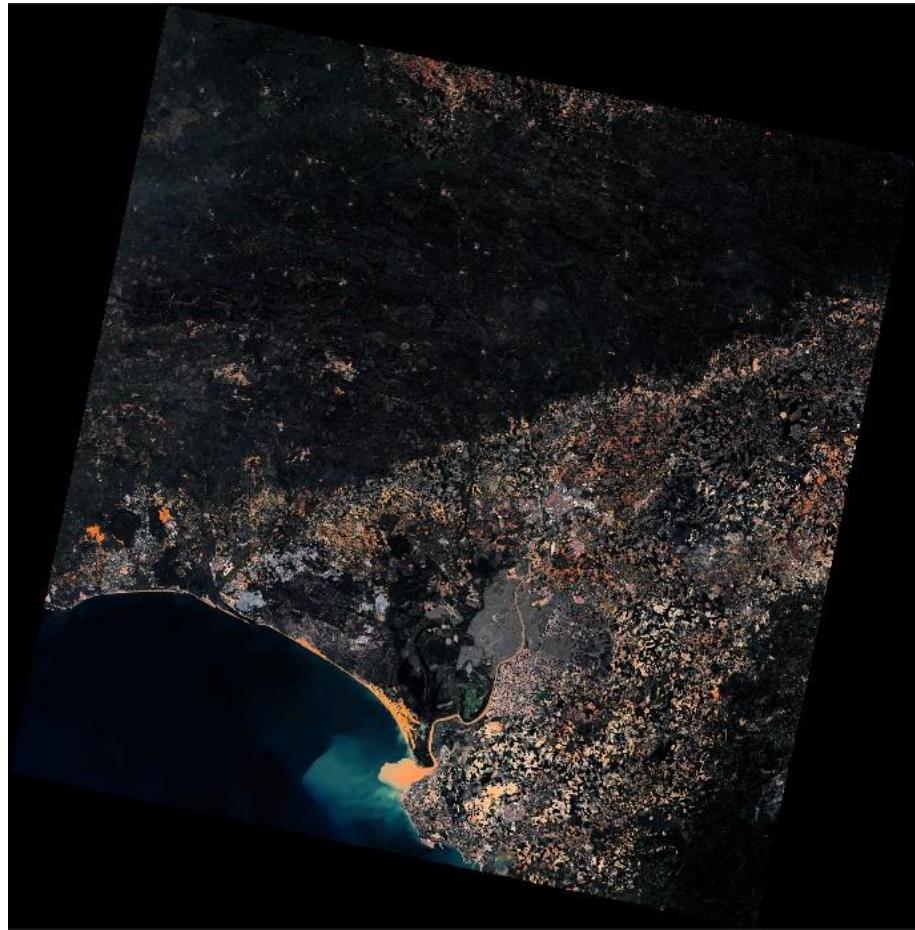


Imagen Landsat C tras corte de colas de ambas componentes S y V en dominio HSV.

MEMORIA:

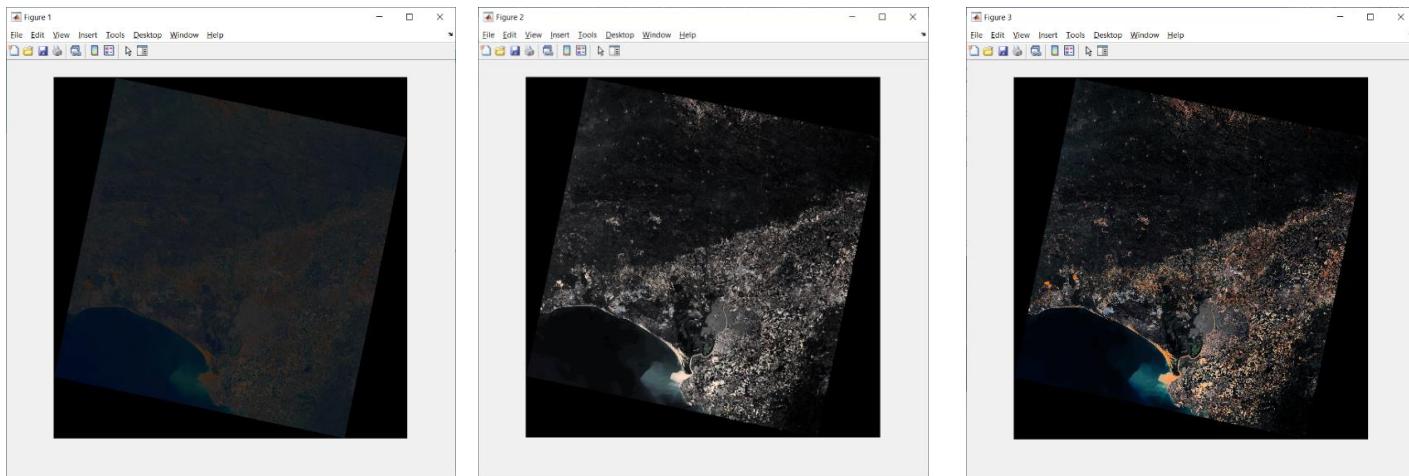
- Código 5.3 + Imagen propia tratada (antes/después)
- Código 5.4 + Imagen propia tratada (antes/después)
- Código 5.5 + Imagen propia tratada (antes/después)

El código de la función es el siguiente:

```
function im2 = corteHSV(im1)
for v = 1:3
    figure(v)
    im2 = rgb2HSV(im1);
    im2 = uint8(im2 .* 255);
    if v == 1
        imTemp = im2(:,:,2); %Para S
        im2(:,:,2) = corte(imTemp, 1, 1); %Corte de cola al 1%
    elseif v == 2
        imTemp = im2(:,:,3); %Para S
        im2(:,:,3) = corte(imTemp, 1, 1); %Corte de cola al 1%
    else
        imTemp1 = im2(:,:,2); %Para S
        imTemp2 = im2(:,:,3); %Para V
        im2(:,:,2) = corte(imTemp1, 1, 1); %Corte de cola al 1%
        im2(:,:,3) = corte(imTemp2, 1, 1); %Corte de cola al 1%
    end
    im2 = double(im2) ./ 255;
    im2 = HSV2RGB(im2);
    im2 = uint8(im2 .* 255);
    imshow(im2);
end
```

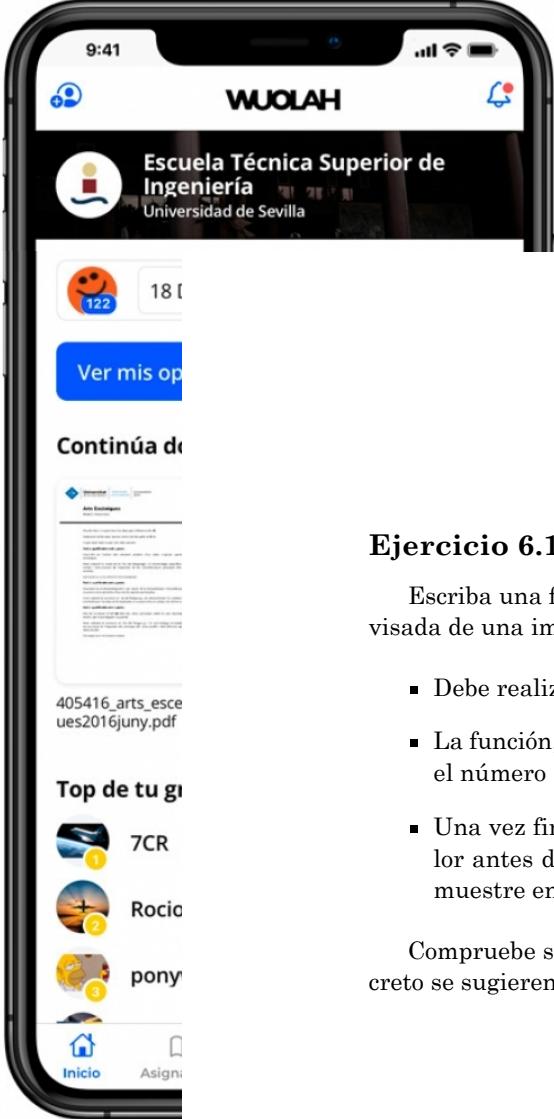
Tras ejecutar los siguientes comandos conseguimos las imágenes resultantes:

```
imR = imread('imágenes/Landsat_C_04-11_R.png');
imG = imread('imágenes/Landsat_C_03-11_G.png');
imB = imread('imágenes/Landsat_C_02-11_B.png');
im = imR;
im(:,:,2) = imG;
im(:,:,3) = imB;
corteHSV(im);
```



LAB6 - Clasificación

Esta sesión se centra en la clasificación de imágenes y, en concreto, en el método de entrenamiento no supervisado. Al realizarla tenga en cuenta que debe visualizar en pantalla la imagen resultante al finalizar.



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

Ejercicio 6.1

Escriba una función llamada *isodata* que lleve a cabo la clasificación no supervisada de una imagen. Tenga en cuenta los siguientes aspectos:

- Debe realizar la clasificación mediante el algoritmo ISODATA.
- La función, además de la imagen original, recibirá como parámetro de entrada el número *C* de clases a utilizar en el proceso.
- Una vez finalizada la clasificación, debe aplicar un procedimiento de seudocolor antes de la visualización de los resultados con objeto de que cada clase se muestre en diferente color.

Compruebe su funcionamiento aplicándola a las imágenes del paquete. En concreto se sugieren las siguientes pruebas:

Imagen	C
IRS A	2
NOAA	3
Landsat A	5

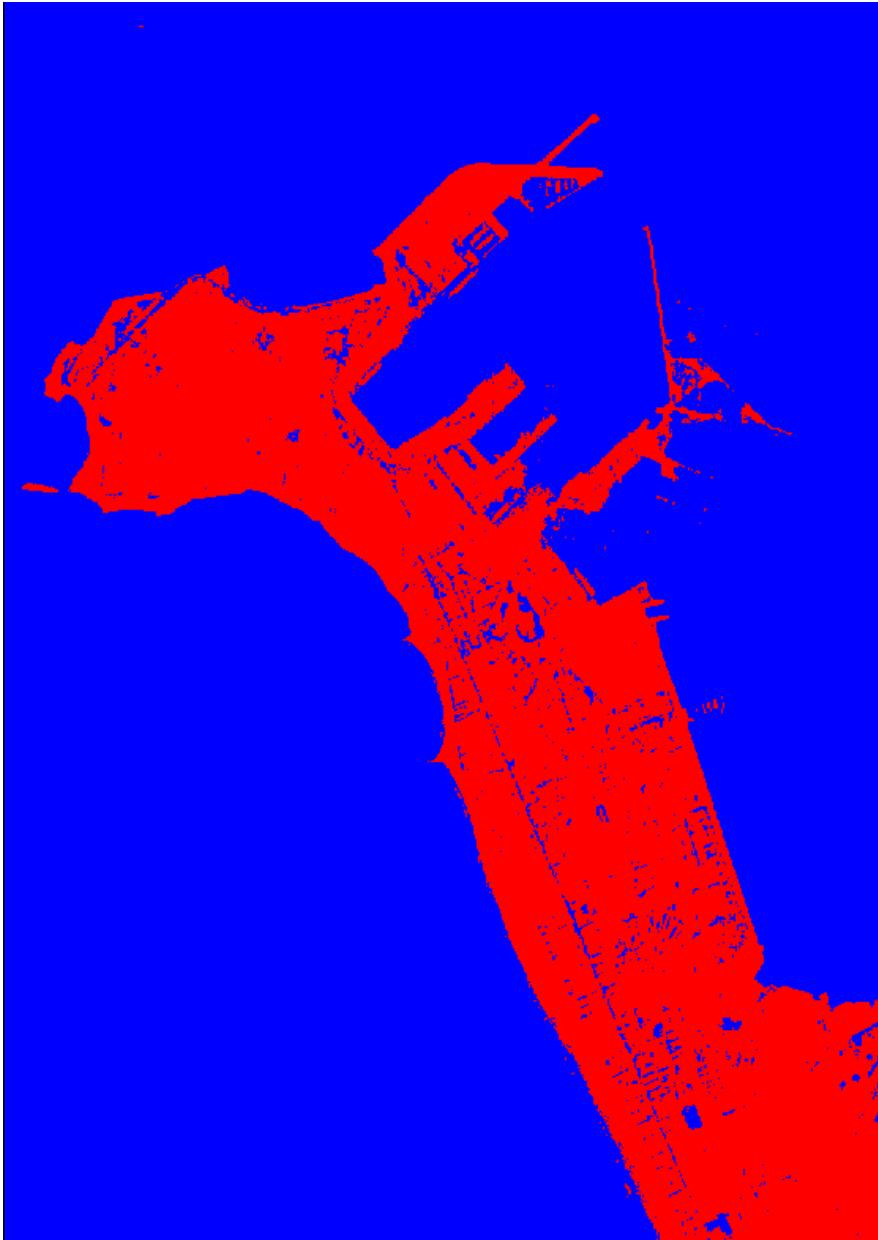


Imagen IRS A clasificada en 2 categorías mediante ISODATA.

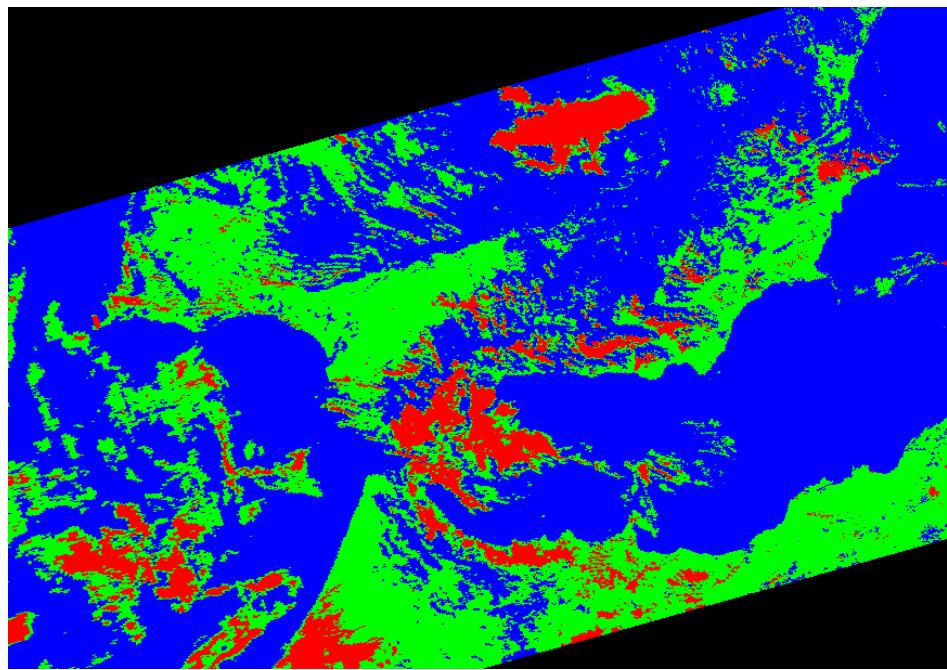


Imagen NOAA clasificada en 3 categorías mediante ISODATA.

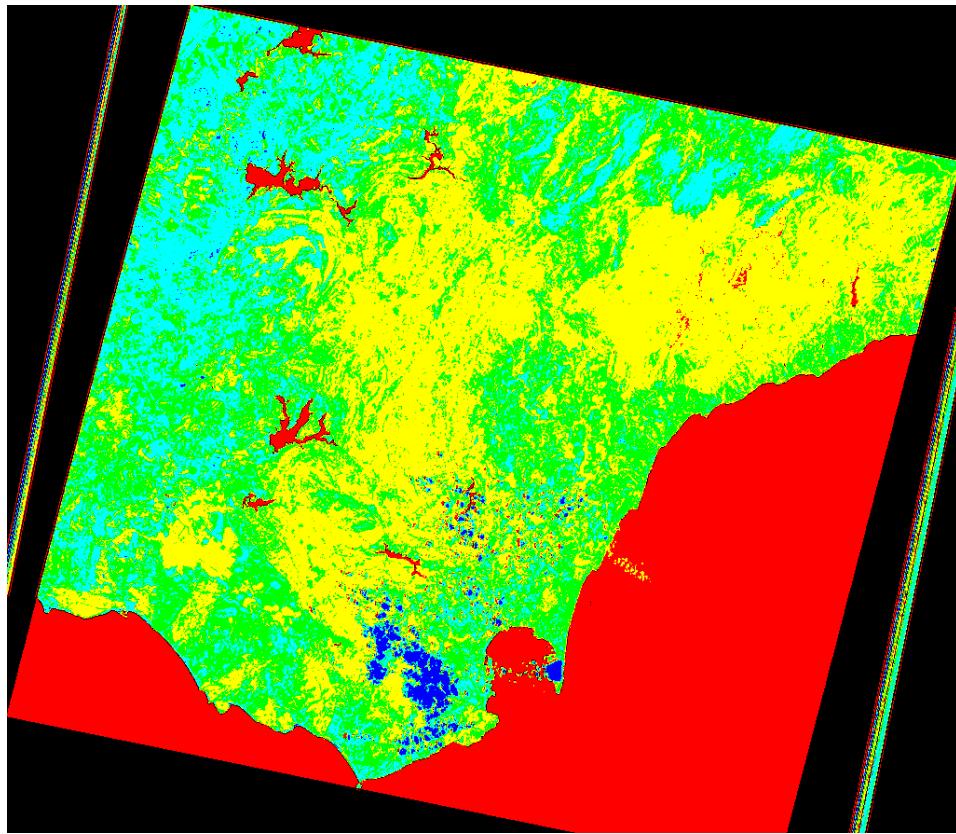


Imagen Landsat AR clasificada en 5 categorías mediante ISODATA.

MEMORIA:

- Código 6.1
- Imagen propia clasificada en 3 categorías (antes/después)
- Imagen propia clasificada en 5 categorías (antes/después)

El código de la función es el siguiente:



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

```

function im2 = isodata(im1, clases)
[F, C, B] = size(im1);
im2 = uint8(zeros(F, C, 3));
%Generar un color aleatorio para cada clase
cellColores = cell(1, clases);
for c = 1:clases
    cellColores{c} = [randi(255), randi(255), randi(255)];
end
%Elegir al azar los centros de las clases en el hiperespacio de bandas
b = 0;
while b == 0
    cellVectores = cell(1, clases);
    for cl = 1:clases
        y = round(rand * F);
        x = round(rand * C);
        listaBandas = zeros(1, B);
        for b = 1:B
            listaBandas(b) = im1(y, x, b);
        end
        cellVectores(cl) = double(listaBandas);
    end
    for v = 1:length(cellVectores)
        if prod(cellVectores(v)) == 0
            b = 0;
            break;
        else
            b = 1;
        end
    end
    cellGruposTemp = cell(1, clases);
    %Mientras haya algún punto que haya cambiado de grupo en una iteración
    b = 0;
    while b == 0
        cellGrupos = cell(1, clases);
        for f = 1:F
            for c = 1:C
                listaBandas = zeros(1, B);
                for b = 1:B
                    listaBandas(b) = im1(f, c, b);
                end
                nd = double(listaBandas);
                if prod(nd) > 0
                    %Calcular la distancia entre todos los píxeles y los centros elegidos
                    listaDistancias = zeros(1, clases);
                    for v = 1:length(cellVectores)
                        listaDistancias(v) = norm(nd - cellVectores(v));
                    end
                    %Asignar cada pixel al centro más próximo formando grupos
                    distMenor = min(listaDistancias);
                    clase = find(listaDistancias == distMenor, 1, 'last');
                    im2(f, c, :) = cellColores{clase};
                    cellGrupos{clase} = [cellGrupos{clase} ; nd];
                end
            end
        end
        %Recalcular el centro de cada grupo como el punto medio del mismo
        for v = 1:length(cellVectores)
            cellVectores{v} = mean(cellGrupos{v});
        end
        if cellfun(@isequal, cellGrupos, cellGruposTemp)
            b = 1;
        end
        cellGruposTemp = cellGrupos;
    end
    imshow(im2);

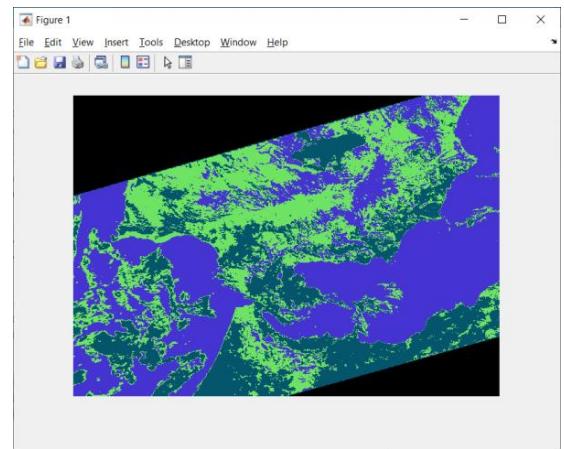
```

Tras ejecutar los siguientes comandos para cada imagen de ejemplo, las imágenes resultantes son (los colores son aleatorios):

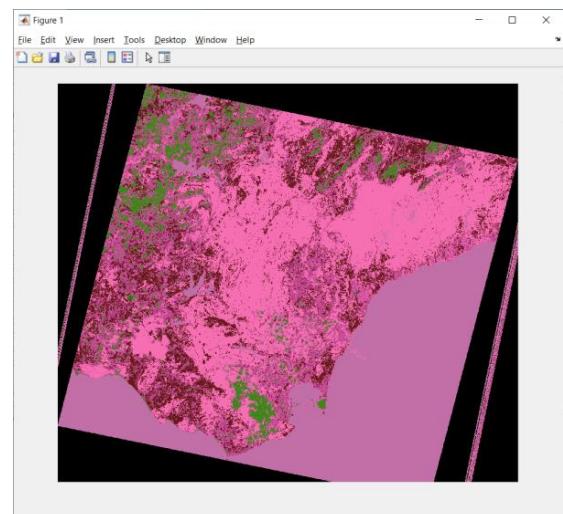
```
im = imread('imágenes/IRS_A_1-1_PAN.png');
im = imresize(im, 0.75);
isodata(im,2);
```



```
im = imread('imágenes/NOAA_1-5_R.png');
im(:,:,2) = imread('imágenes/NOAA_2-5_NIR.png');
im(:,:,3) = imread('imágenes/NOAA_3-5_SWIR.png');
im = imresize(im, 0.75);
isodata(im,3);
```



```
im = imread('imágenes/Landsat_A_3-3_R.png');
im(:,:,2) = imread('imágenes/Landsat_A_2-3_G.png');
im(:,:,3) = imread('imágenes/Landsat_A_1-3_B.png');
im = imresize(im, 0.5);
isodata(im,5);
```



LAB7 - Geocorrección

Esta sesión se centra en la corrección geométrica de una imagen NOAA de la península ibérica mediante puntos de control (*Ground Control Points*, GCP). Al realizarla tenga en cuenta las siguientes consideraciones:

- Las coordenadas geográficas (latitud y longitud) de los puntos de control puede obtenerlas de la web: <https://www.geoplaner.com/>
- La conversión de dichas coordenadas geográficas a coordenadas UTM puede realizarla en la web: <http://www.zonums.com/online/coords/cotrans.php?module=13>. Es recomendable obtenerlas para una zona UTM en torno a la 30.
- Unas coordenadas UTM adecuadas como esquina superior izquierda de la imagen de salida podrían ser: UTM 30 N –750000 5400000. Tenga en cuenta también, que para cubrir toda la península ibérica (partiendo de dichas coordenadas) debe generar una imagen de aproximadamente 2500 km de anchura.
- Para obtener la fila y la columna de cada punto de control en la imagen original es recomendable utilizar algún editor gráfico que permita visualizar dicha información como por ejemplo: <https://www.piskelapp.com/>
- Para agilizar el proceso de desarrollo puede considerar una resolución inferior y subirla cuando haya comprobado que todo funciona correctamente. Por ejemplo, para imágenes NOAA puede trabajar con una resolución de 4000 m y subir finalmente a los 1000 m originales.

Ejercicio 7.1

A partir de la imagen NOAA-ORBIT, elija una banda y recorte una zona que cubra la península ibérica completa. Sobre la imagen recortada, aplique una corrección geométrica basada en puntos de control.

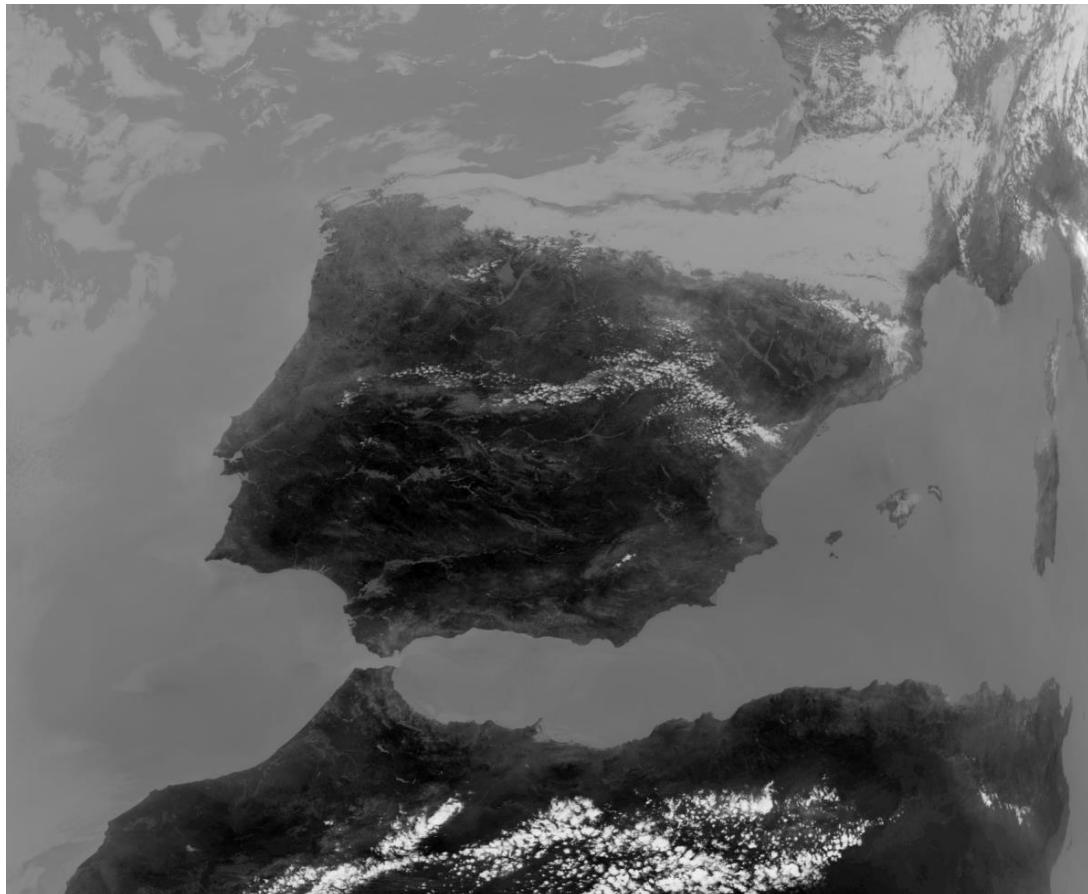


Imagen NOAA-ORBIT original recortada (banda 4).

9:41



WUOLAH



Escuela Técnica Superior de
Ingeniería
Universidad de Sevilla



122

18

Ver mis op

Continúa d



405416_arts_esce_ues2016juniy.pdf

Top de tu gr



7CR



Rocio

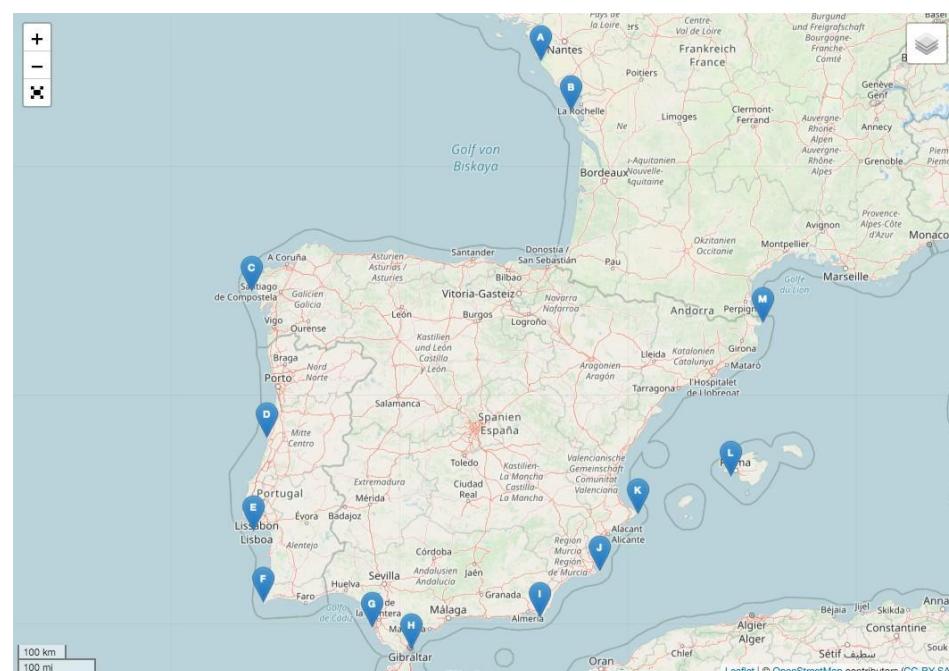


pony



Inicio

Asigni



Selección de puntos de control de la zona mediante Geoplanner.

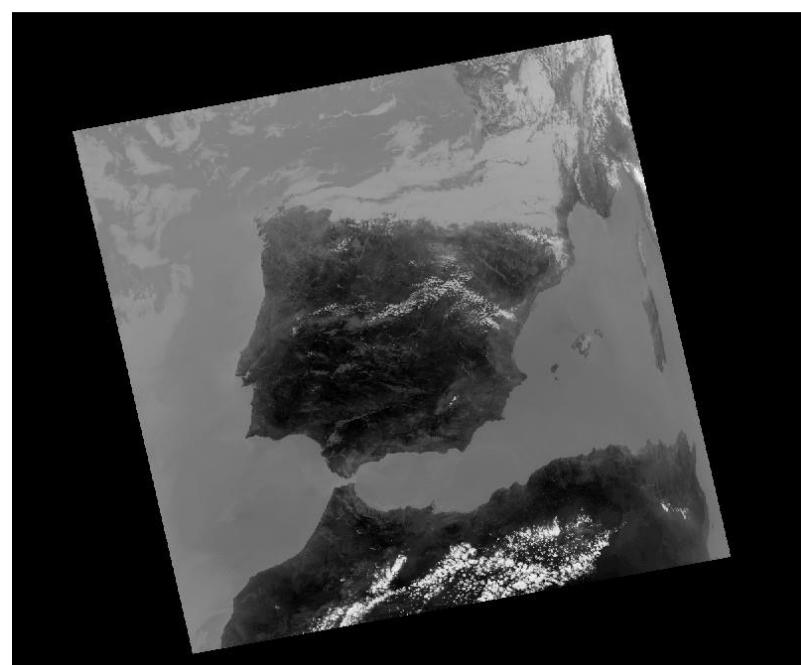


Imagen NOAA-ORBIT corregida geométricamente mediante GCP.

MEMORIA:

- Código 7.1
- Imagen recortada previa a la corrección
- Captura de Geoplanner con puntos de control
- Imagen corregida geométricamente

El código de la función para esa imagen en concreto es el siguiente:

```
function im2 = gcpnoaa(im1, b)
im1 = im1(:,:,b); %Se indica la banda a extraer para que sea monobanda
xGCP = [564182.660; 619428.166; -8699.905; -1211.293; -47835.184;...
-31344.329; 201608.988; 289282.102; 576581.823; 702631.129;...
777819.735; 971756.716; 1018974.579]; %Puntos elegidos de la coordenada x
yGCP = [5189100.568; 5105847.492; 4772143.250; 4459524.870; 4276722.581;...
4112783.164; 4048541.418; 4001266.879; 4065272.753; 4169356.976;...
4296246.791; 4383322.393; 4703410.408]; %Puntos elegidos de la coordenada y
cGCP = [1259; 1294; 511; 441; 352; 327; 560; 645; 1015; 1166; 1266; 1435; 1512]; %Columnas aproximadas a las coordenadas x
fGCP = [50; 136; 333; 608; 772; 912; 1011; 1067; 1053; 984; 880; 836; 558]; %Filas aproximadas a las coordenadas y
o = ones(13,1);
a = regress(fGCP, [o, xGCP, yGCP]);
b = regress(cGCP, [o, xGCP, yGCP]);

[F, C] = size(im1);
im2 = uint8(zeros(F+500, C+500)); %Incrementamos el tamaño de la imagen
%destino para que quepa la imagen original corregida
x = -750000; %Coordenada x de la esquina superior izquierda
y = 5400000; %Coordenada y de la esquina superior izquierda
xTemp = x;
for f = 1:F+500
    for c = 1:C+500
        fn = a(1) + a(2) * x + a(3) * y;
        cn = b(1) + b(2) * x + b(3) * y;
        x = x + 1000;
        if fn >= 1 && fn <= F && cn >= 1 && cn <= C
            im2(f,c) = im1(round(fn), round(cn));
        end
    end
    x = xTemp;
    y = y - 1000;
end
imshow(im2);
```

La imagen recortada previa es la siguiente:



Tras ejecutar los siguientes comandos para la imagen del ejemplo, la imagen resultante es:

```
im = imread('imágenes/NOAA-ORBIT_4-5_LWIR.png');  
gcpnoaa(im, 1);
```

