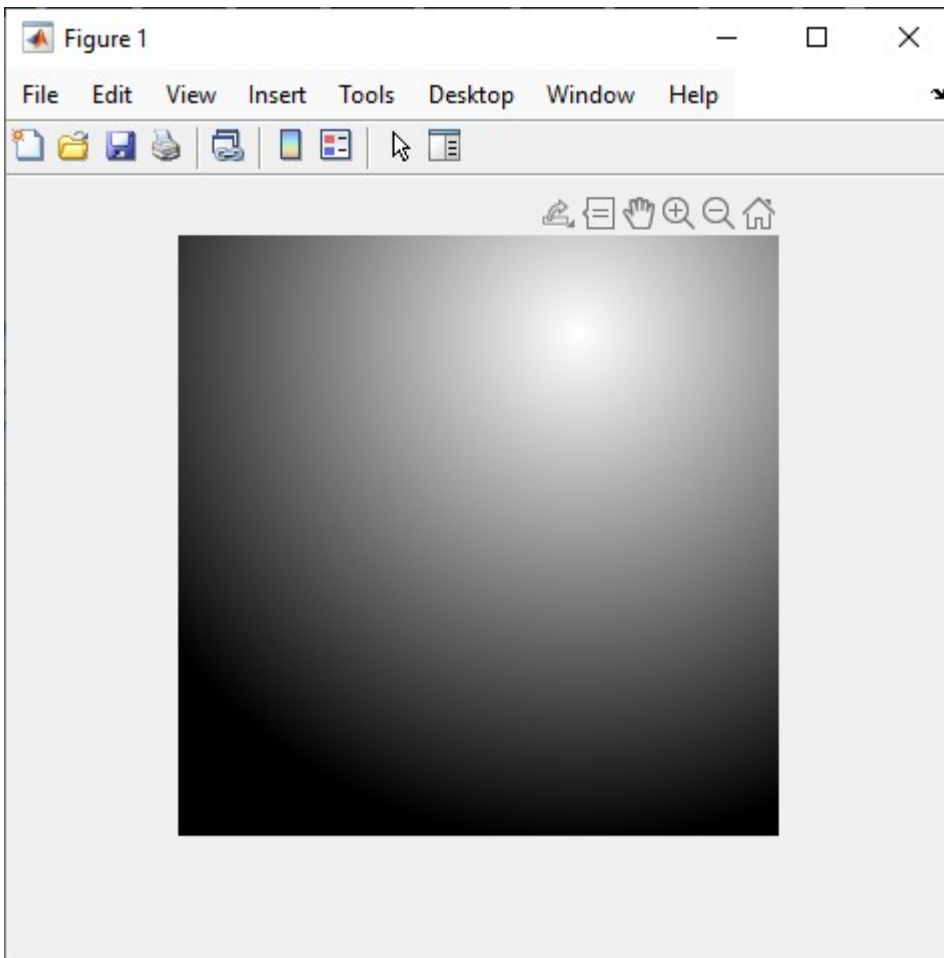


Practica 1

Ejercicio 1.A:

```
%Apartado 1
%Coordenadas del foco (a,b)
a=50;
b=200;
%Construcción imagen MxN
M=300; N=300;
for x=1:M
    for y=1:N
        %El 255 lo usamos para indicar que valor en la escala de grises vamos a
        %usar. Siendo 255 blanco y 0 negro.
        %Para indicar esto usamos un valor decimal entre 0 y 1, ambos
        %incluidos, por lo que usando la primera parte del calculo conseguimos que
        %cuanto mas nos acercamos al foco, mas blanco sea en la escala de grises.
        %Y dividiendo entre 255 conseguimos que el valor sera decimal entre 0 y 1.
        I(x,y)=(255-sqrt((x-a)^2+(y-b)^2))/255;
    end
end
imshow(I)
```

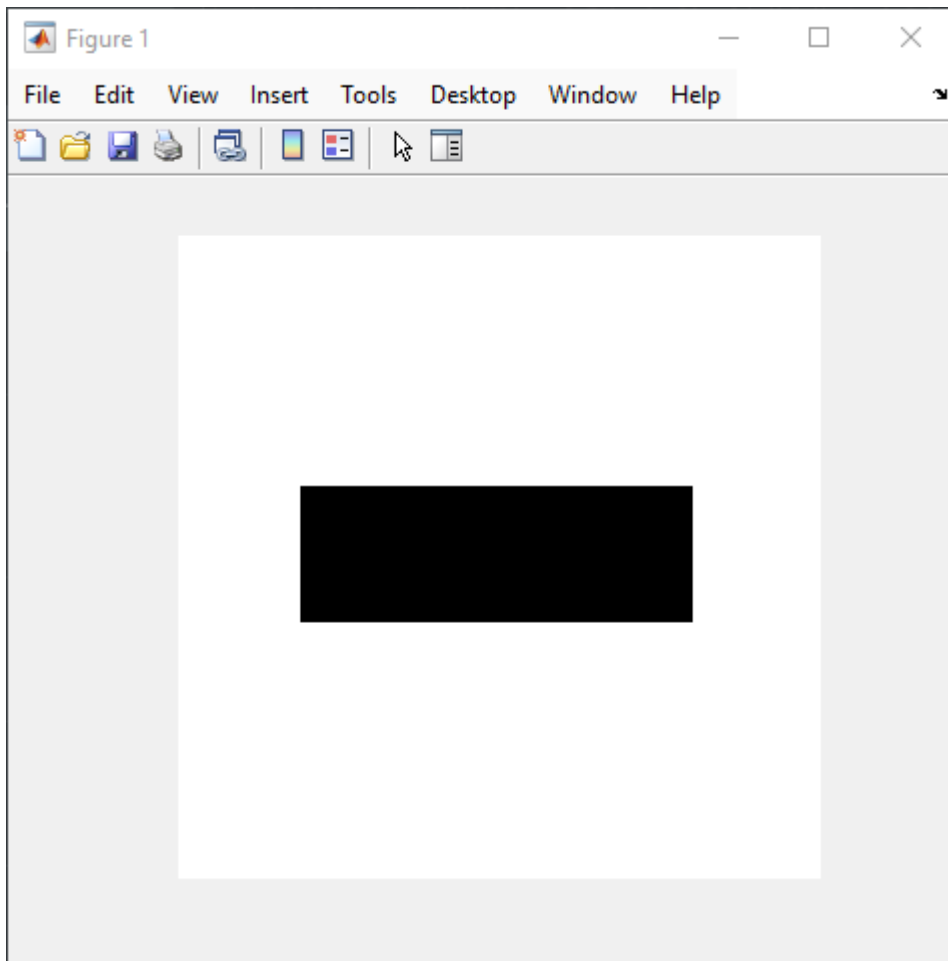
Nos da como resultado:



Ejercicio 1.B:

```
%imagen de tamaño 30x30
%Creamos una matriz de solo unos del tamaño especificado(Imagen en blanco)
I=ones(100,100);
%Añadimos 0 a una region para mostrar una figura
%En este caso, para las filas entre 40 y 60, y las columnas 20 y 80
I(40:60,20:80)=0;
%Con fit hacemos que la imagen se adapte a tamaño de la ventana
imshow(I,'InitialMagnification','fit')
```

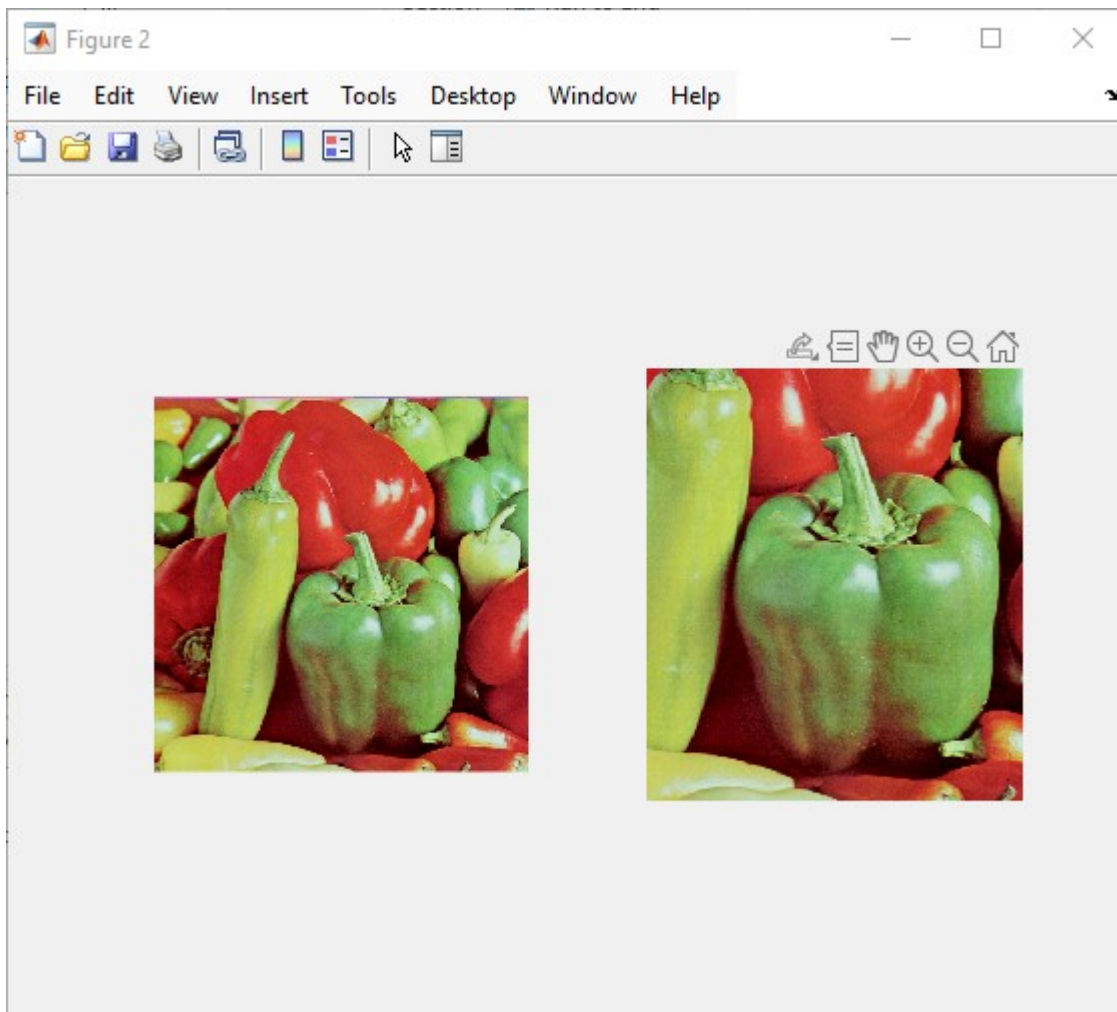
Nos da como resultado:



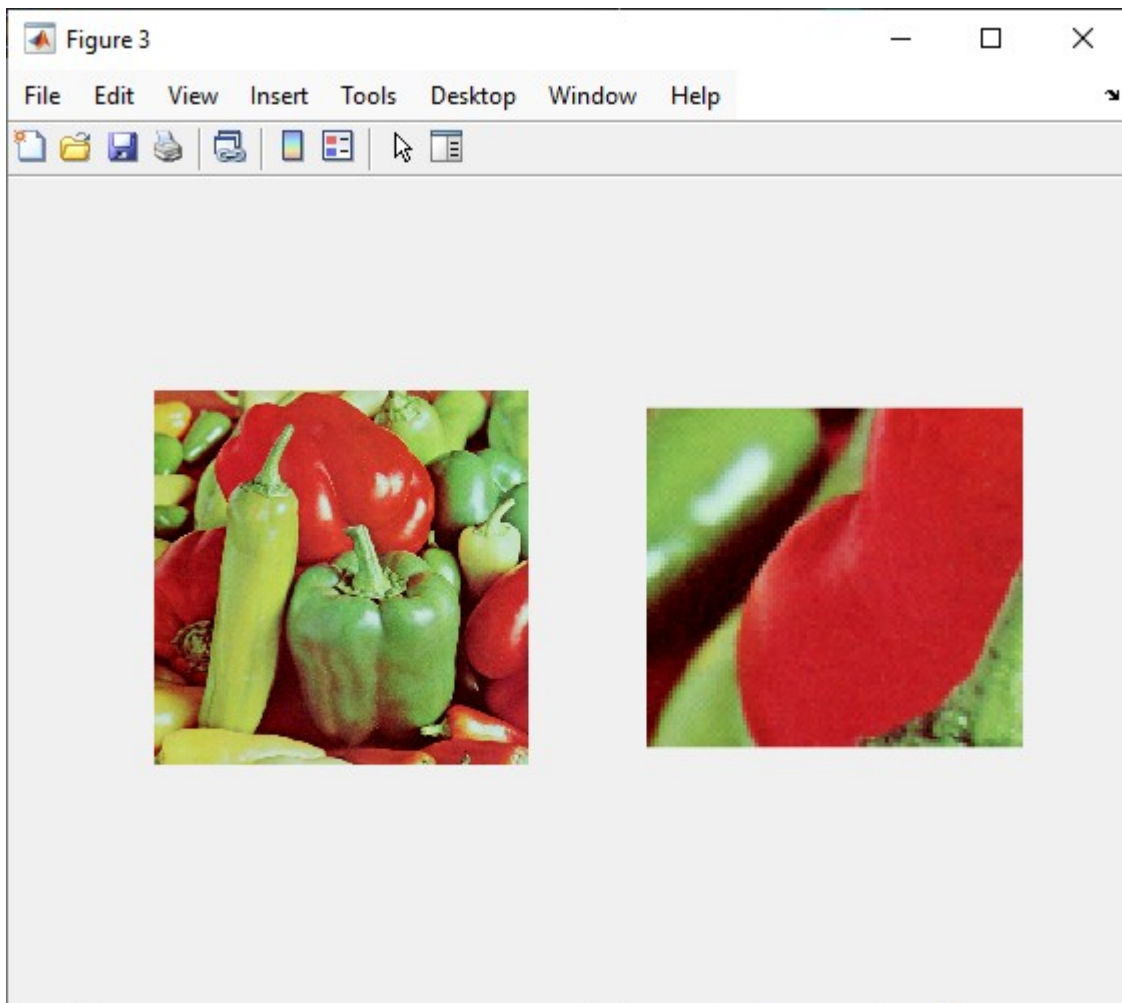
Ejercicio 1.C:

```
figure;
subplot(1,2,1)
I=imread(' ../Imagenes/4.2.07.tiff');
imshow(I)
%Nos permite seleccionar una zona con el raton
J=imcrop;
%seleccionar con el ratón la región de
%interés
subplot(1,2,2)
imshow(J)
%También se puede hacer fijando
%[xmin ymin ancho y alto]
D=imread(' ../Imagenes/4.2.07.tiff');
%A imcrop le pasamos la imagen que queremos recortar
%Despues una lista con los valores x e y donde queremos empezar a recortar
%Seguidos del ancho y alto que queremos que sea el recorte
D1=imcrop(D,[60 40 100 90]);
figure;
subplot(1,2,1);
imshow(D);
subplot(1,2,2);
imshow(D1);
```

Para la primera parte seleccionamos una parte con el ratón y nos podría como resultado algo así:



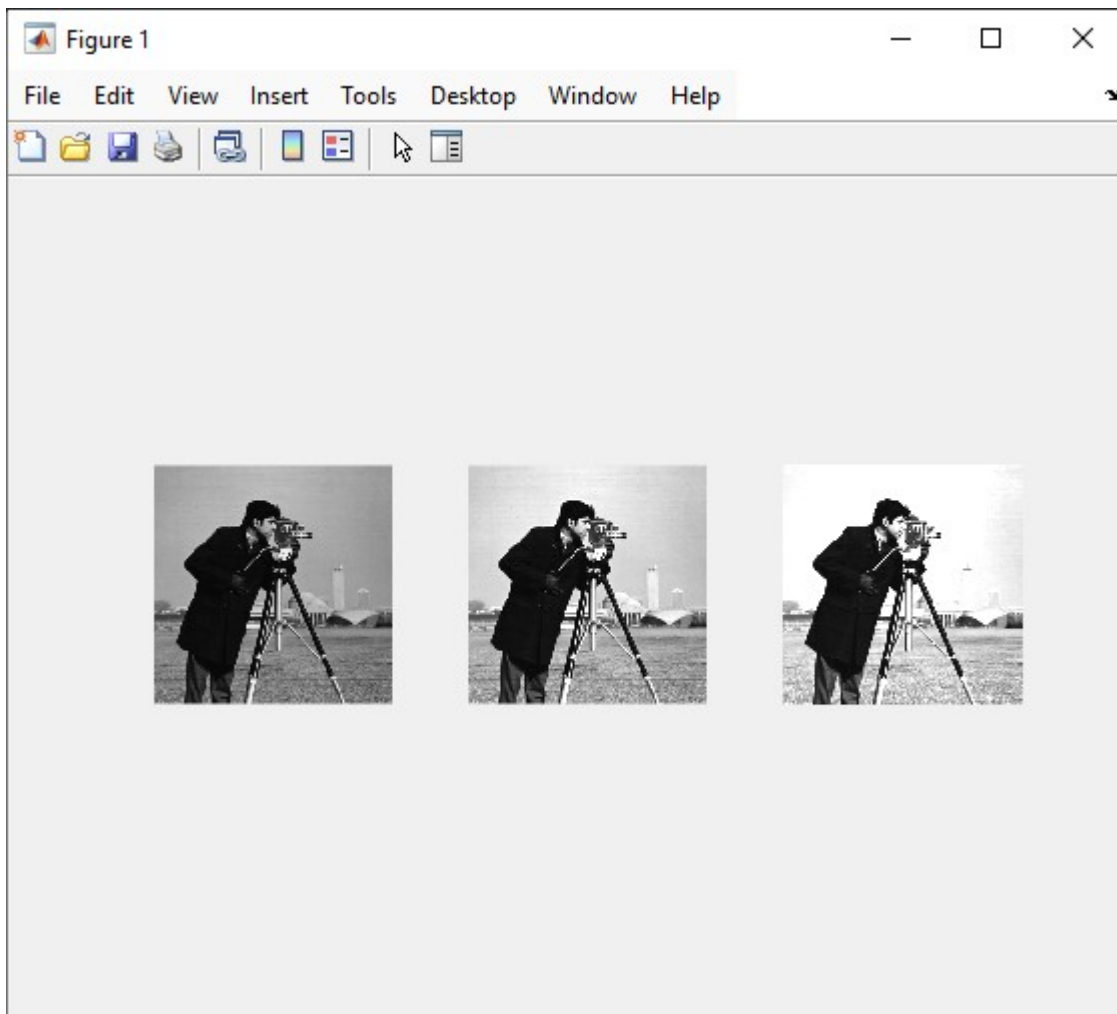
Y para la segunda parte, desde el código seleccionamos una parte concreta dándonos siempre el mismo resultado. Como por ejemplo:



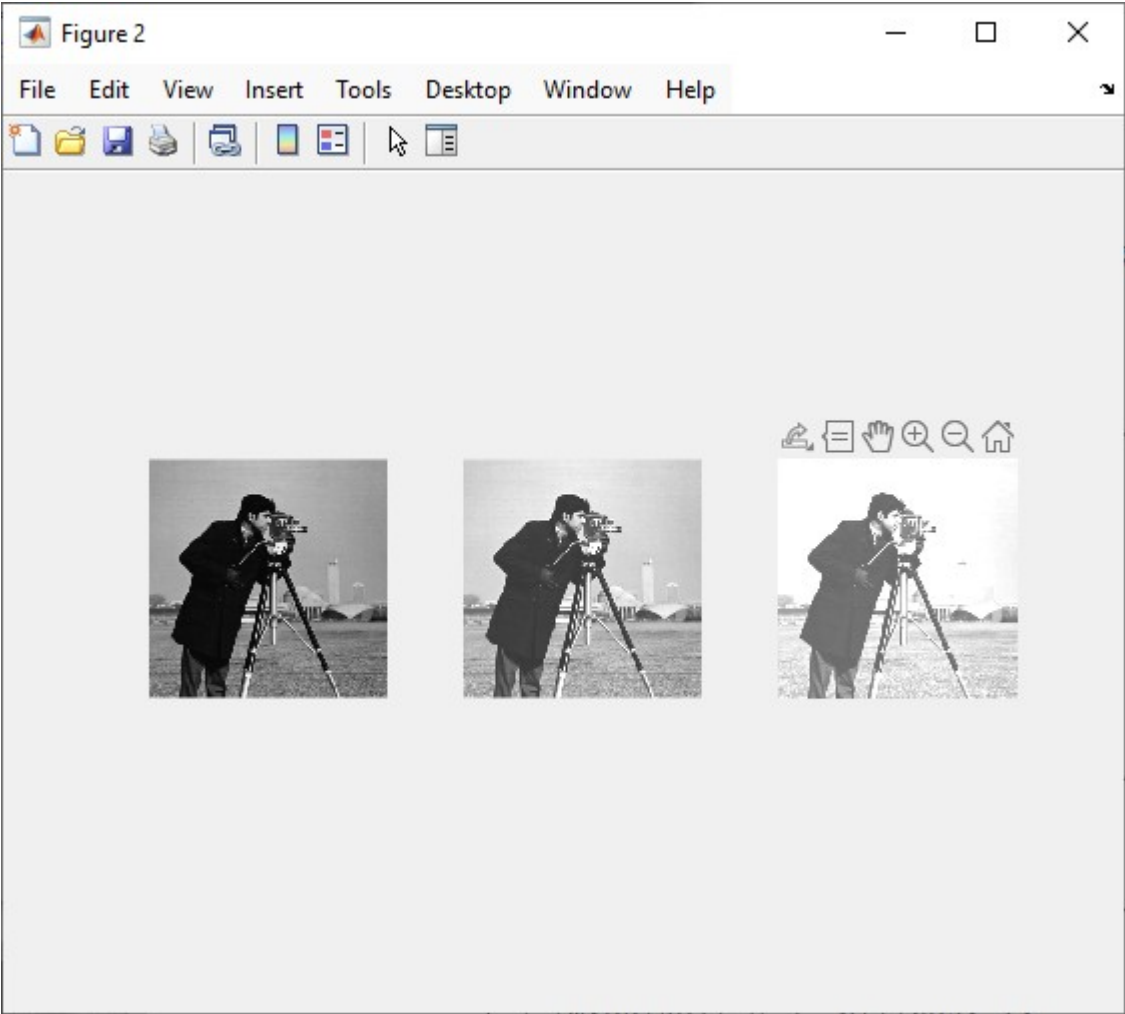
Ejercicio 2.A:

```
I=imread(' ../Imagenes/cameraman.tif');  
%Cuanto mas se multiplica la imagen, cuando mayor sea el valor original,  
%mayor sera el crecimiento, aumentando el contraste  
figure  
subplot(1,3,1)  
imshow(I, 'InitialMagnification','fit');  
subplot(1,3,2)  
imshow(I*1.3, 'InitialMagnification','fit');  
subplot(1,3,3)  
imshow(I*1.6, 'InitialMagnification','fit');  
  
%Aumenta todos los valores de la imagen de manera igual, haciendo que en  
%general sea mas blanca  
figure  
subplot(1,3,1)  
imshow(I, 'InitialMagnification','fit');  
subplot(1,3,2)  
imshow(I+50, 'InitialMagnification','fit');  
subplot(1,3,3)  
imshow(I+100, 'InitialMagnification','fit');
```

Si multiplicamos la imagen obtenemos lo siguiente:



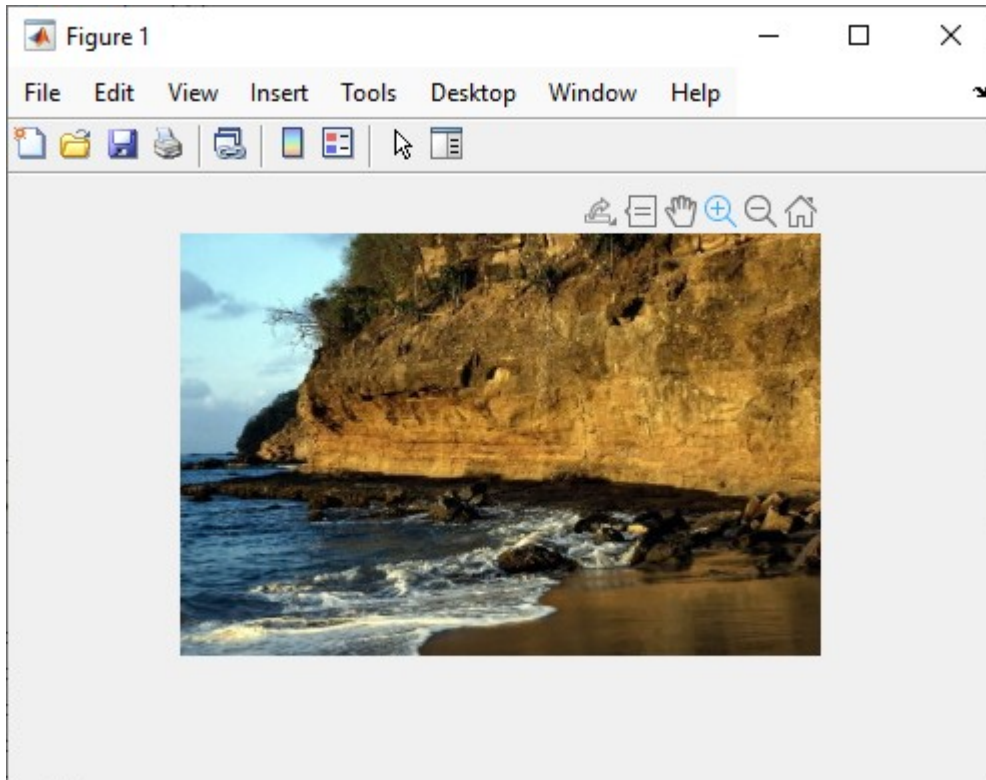
Si sumamos algún valor a toda la imagen obtenemos:



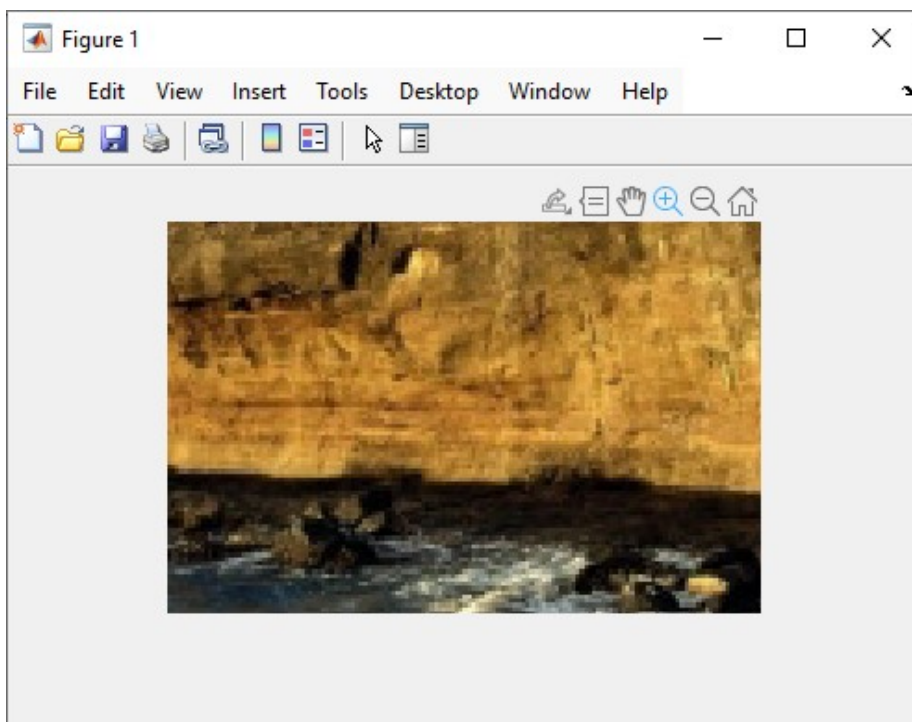
Ejercicio 2.B:

```
I=imread(' ../Imagenes/acantilado.png');  
imshow(I)  
%Zoom on nos permite hacer que al hacer click en cualquier parte de la  
%imagen, se le haga zoom ahí  
zoom on
```

De primeras vemos la imagen normal:



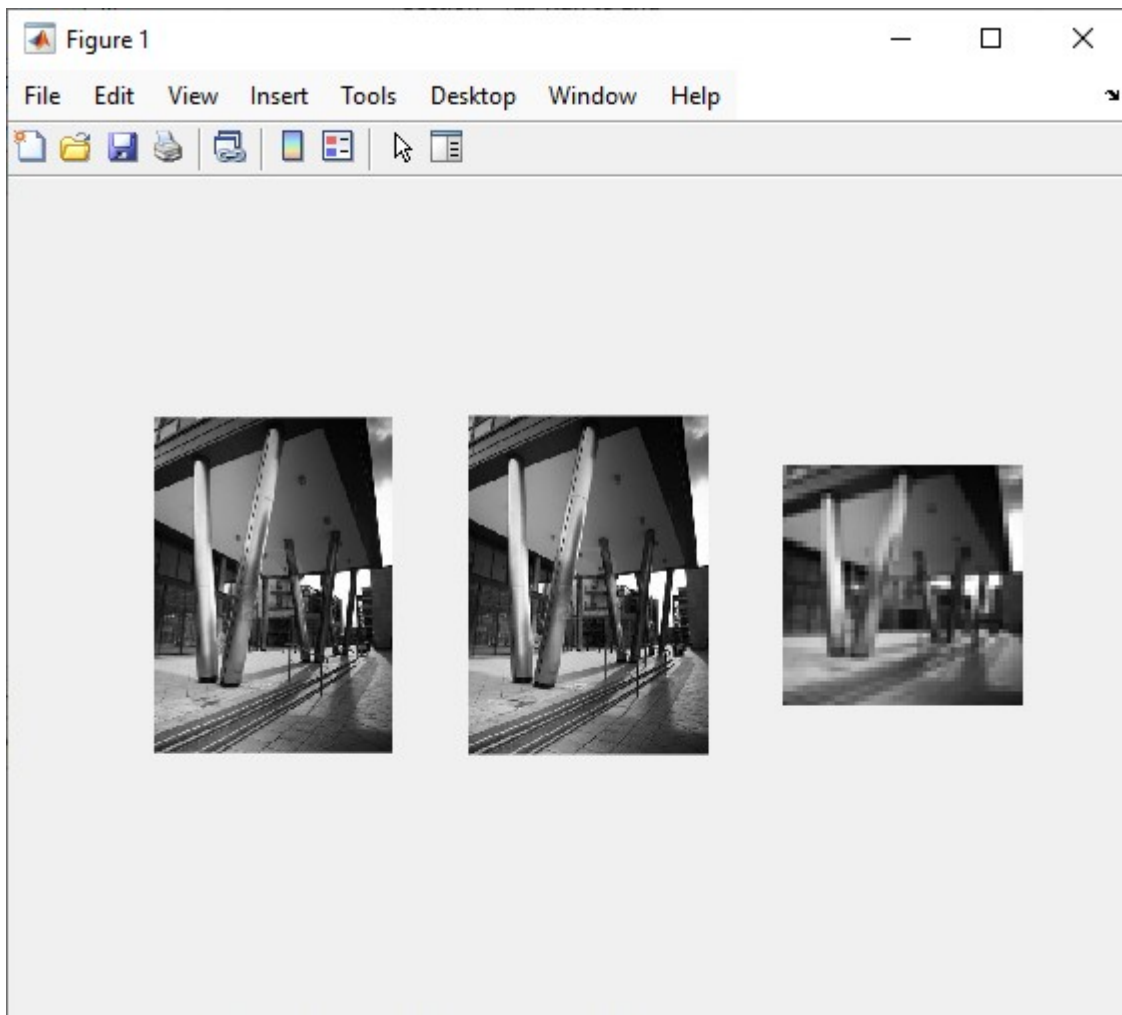
Pero al usar zoom on, al hacer click hacemos zoom:



Ejercicio 2.C:

```
I=imread(' ../Imagenes/arquitectura-en-leeds-uk.jpg');  
%imresize nos permite reducir la imagen aplicandole una escala y un  
%algoritmo  
J=imresize(I,0.5,'nearest');  
%Asi podemos reducir la imagen que queramos al numero de filas y columnas  
%que queramos, incluso cambiando el aspect ratio  
J1=imresize(I,[50 50]) %reduce la  
%imagen al tamaño 40x30; 30 filas y  
%40 columnas  
figure  
subplot(1,3,1)  
imshow(I)  
subplot(1,3,2)  
imshow(J)  
subplot(1,3,3)  
imshow(J1)
```

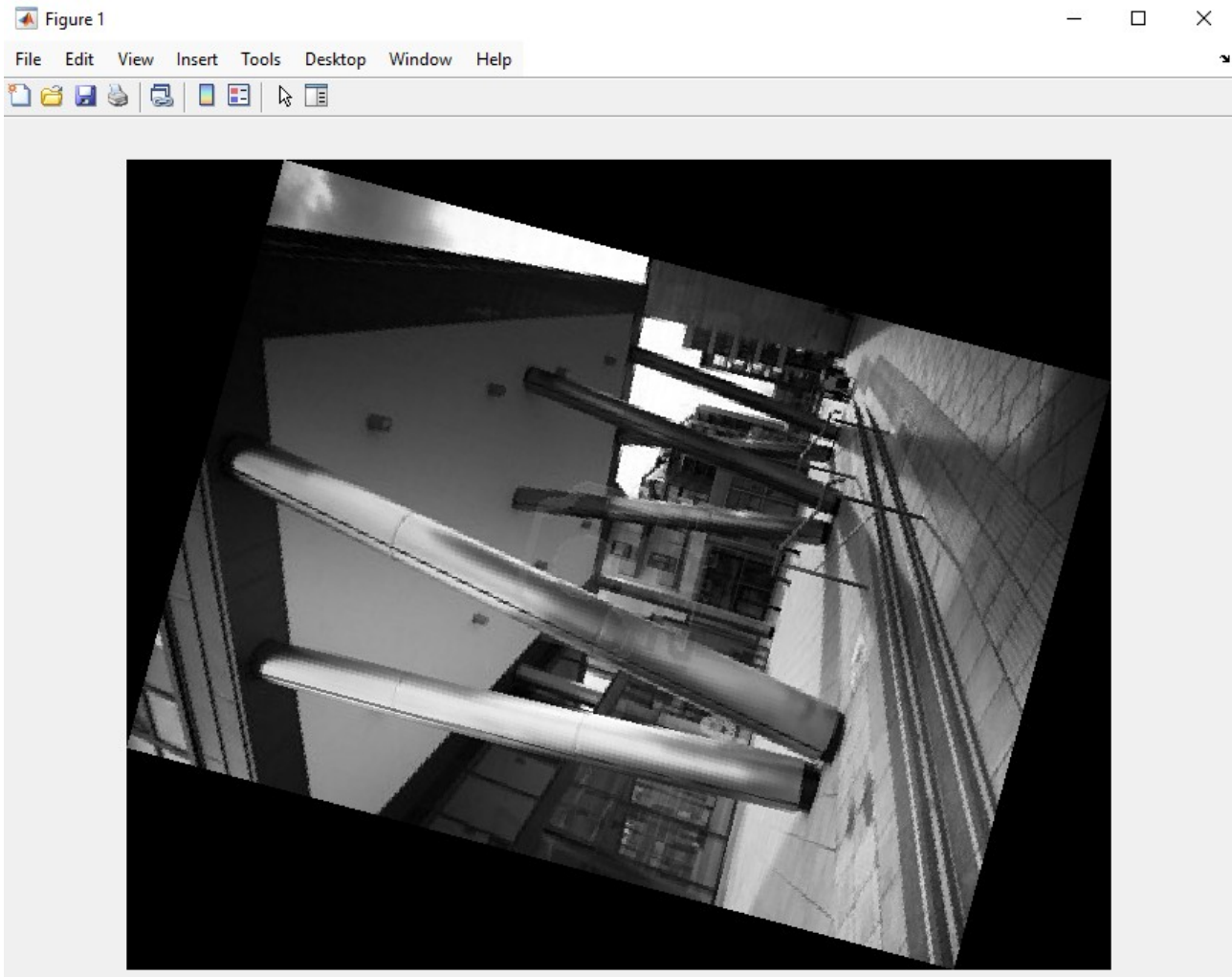
Nos da como resultado:



Ejercicio 2.D:

```
IR=imread(' ../Imagenes/arquitectura-en-leeds-uk.jpg');  
%A imrotate Le pasamos una imagen y los grados por los que queremos rotar  
%la imagen. Nos rellena los bordes con negro si no la rotamos por algun  
%multiplo de 90  
IR=imrotate(J,75);  
imshow(IR)
```

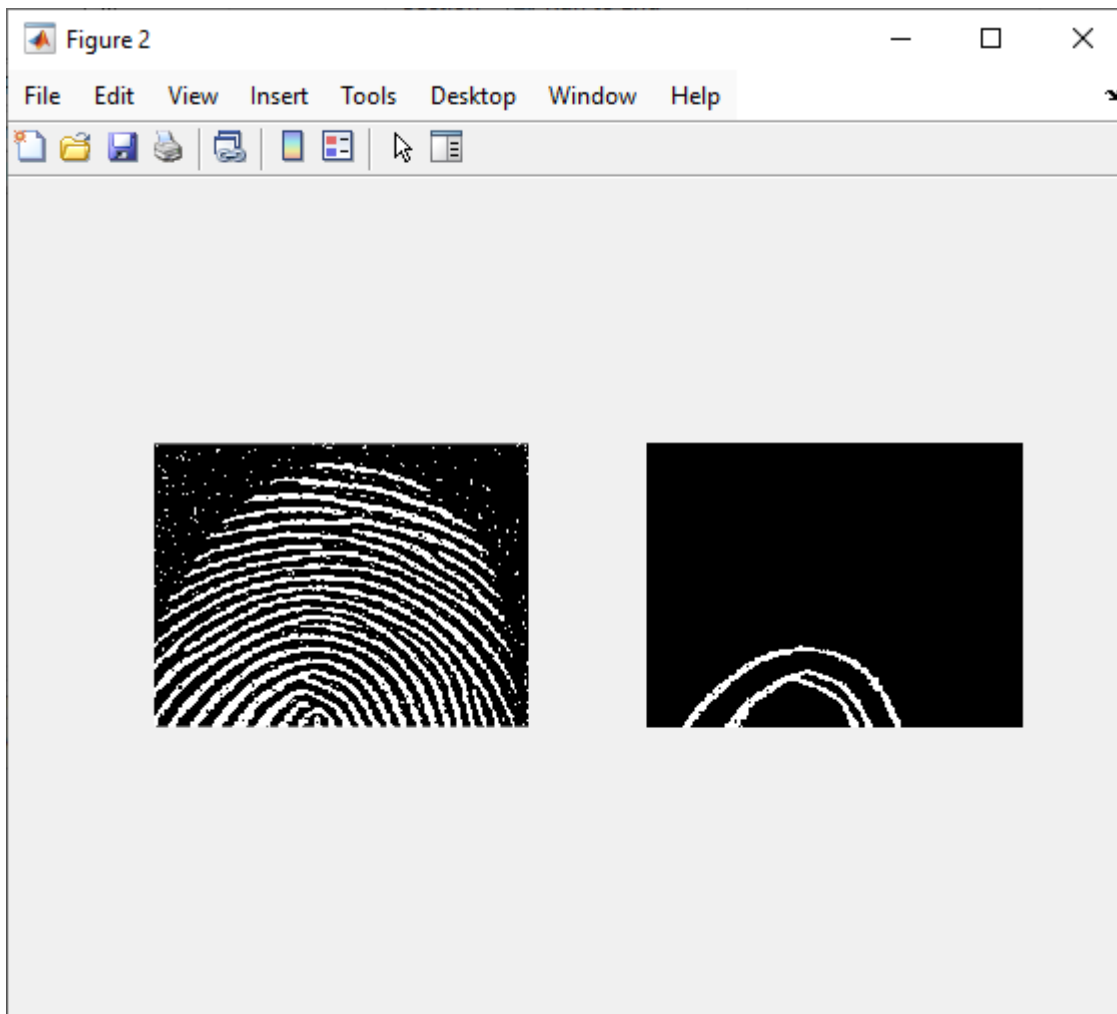
Nos da:



Ejercicio 3.A:

```
I=imread(' ../Imagenes/Fig9.11(a).jpg');
imshow(I)
%Con ginput puedo seleccionar hacer click en una imagen y me devuelve la
%posicion x e y
c = ginput(1);
r = ginput(1);
%bwselect nos permite quedarnos con la parte de la imagen que se solape con
%los pixeles localizados en c y r. Le pasamos primero la imagen, luego la
%posicion x del pixel y ultimo la posicion Y. Estos pueden ser tambien
%arrays si queremos seleccionar varias partes. El ultimo valor opcional le
%podemos decir que conectividad queremos usar, 4 u 8
I0=bwselect(I,[c(1), r(1)],[c(2), r(2)],8);
figure
subplot(1,2,1)
imshow(I)
subplot(1,2,2)
imshow(I0)
```

Si hago click en un par de sitios, obtengo la imagen:



Preguntas:

1.a) ¿Qué ocurre si no dividimos por 255?

Pasara que los valores no se quedaran en el rango de 0 a 1, de manera que no obtendremos el resultado deseado.


1.b) ¿Qué significan cada uno de los parámetros de imshow? ¿Van juntos?

Para este caso, el primero parámetro es I que es la imagen. De segundo tenemos "InitialMagnification" que sirve para indicar como queremos que se vea la imagen al abrirla. Por ultimo tenemos "fit" que nos dice que la imagen se ajustara a la venta. Estos dos últimos van juntos y podrías pasar "fit" como parámetro posicional o escribiendo "InitialMagnification='fit'".

1.c) La imagen 'ic.tif' no está en el conjunto de imágenes proporcionado, selecciona cualquier otra imagen. imcrop, ¿Funciona igual cuando la imagen es de color? ¿Recorta igual?

En mi propio código he usado una imagen a color y recorta igual ya que aplica la función a todas las matrices de colores a la vez.

2. c) ¿Qué algoritmos de redimensionamiento se pueden aplicar aparte de 'bilinear'?

Method	Description
"nearest"	Nearest-neighbor interpolation; the output pixel is assigned the value of the pixel that the point falls within. No other pixels are considered.
"bilinear"	Bilinear interpolation; the output pixel value is a weighted average of pixels in the nearest 2-by-2 neighborhood.
"bicubic"	Bicubic interpolation; the output pixel value is a weighted average of pixels in the nearest 4-by-4 neighborhood.
<div> Note Bicubic interpolation can produce pixel values outside the original range.</div>	
Interpolation Kernel	Description
"box"	Box-shaped kernel
"triangle"	Triangular kernel (equivalent to "bilinear")
"cubic"	Cubic kernel (equivalent to "bicubic")
"lanczos2"	Lanczos-2 kernel
"lanczos3"	Lanczos-3 kernel

2. d) Gira 90° la imagen sin utilizar el comando imrotate, es decir, implementa una función para devolver la imagen girada.

```
function imgRotada = girar90(img)
    [x, y] = size(img);
    %Al girar la imagen 90 grados, dimensiones de la imagen se invierten
    %Por lo que creamos una matriz de ceros del tamaño inverso de la matriz
    %original
    imgRotada = uint8(zeros(y, x));

    for i = 1:x
        for j = 1:y
            %Recorremos la imagen original colocando en la nueva matriz pero
            %en la nueva matriz invertimos los indices, de esta manera
            %vamos colocando los pixeles de la imagen original que irian,
            %por ejemplo, en la primera fila en la ultima columna de la
            %imagen rotada
            imgRotada(j, x-i+1) = img(i, j);
        end
    end
end
```

3. ¿En qué píxeles se encuentran los objetos seleccionados? Ayuda: usar `ind2sub` para convertir índices lineales devueltos por la función a coordenadas matriciales. Indica las cuatro esquinas que conforma la selección. Esto forma lo que se llama 'Bounding Box'.

```
%Usamos find para encontrar todos los 1 en I0
[rows, cols] = find(I0);
%Las esquinas seran los minimos y maximos de filas y columnas
xmin = min(cols);
xmax = max(cols);
ymin = min(rows);
ymax = max(rows);
esquinas = [[xmin, ymin], [xmax, ymin], [xmax,ymax], [xmin, ymax]]
```

Con eso podemos mostrar las coordenadas pero teniéndolas se puede dibujar un rectángulo sobre la imagen:

```
subplot(1,3,3)
imshow(I0)
rectangle('Position', [xmin, ymin, xmax - xmin, ymax - ymin], 'EdgeColor', 'r', 'LineWidth', 1);
```

