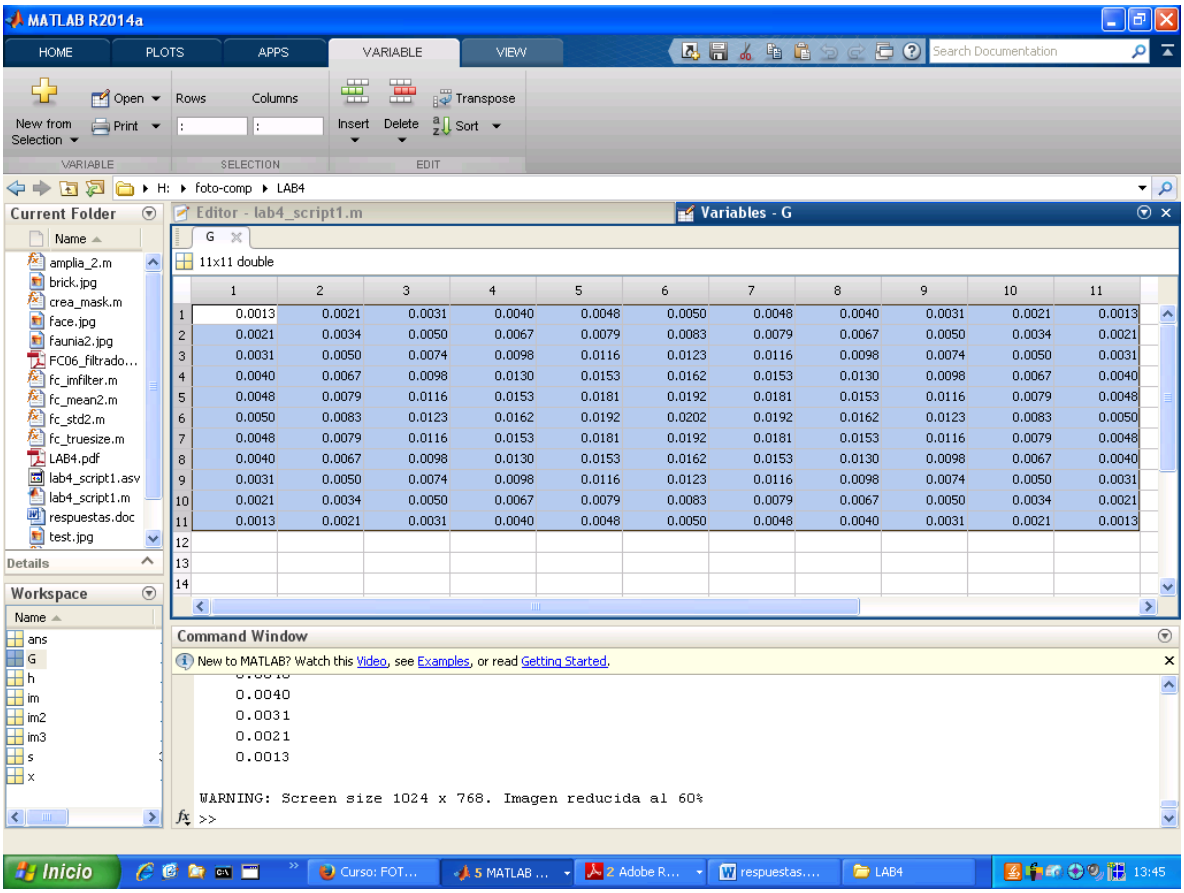
Apellidos, Nombre: Marcos Bernal España
Apellidos, Nombre: Yolanda de la Hoz Simón

Ejercicio 1:

Adjuntar los coeficientes (G) del filtro resultante.

Los mostramos en una matriz de 11x11 a través de una captura de pantalla de Matlab.



Adjuntar ambas imágenes submuestreadas (con y sin filtro previo). ¿Se han reducido los efectos de "Moire"?

Se reducen los efectos de "Moire" tal y como podemos apreciar con en las dos siguientes imágenes con las figuras 2 y 4.

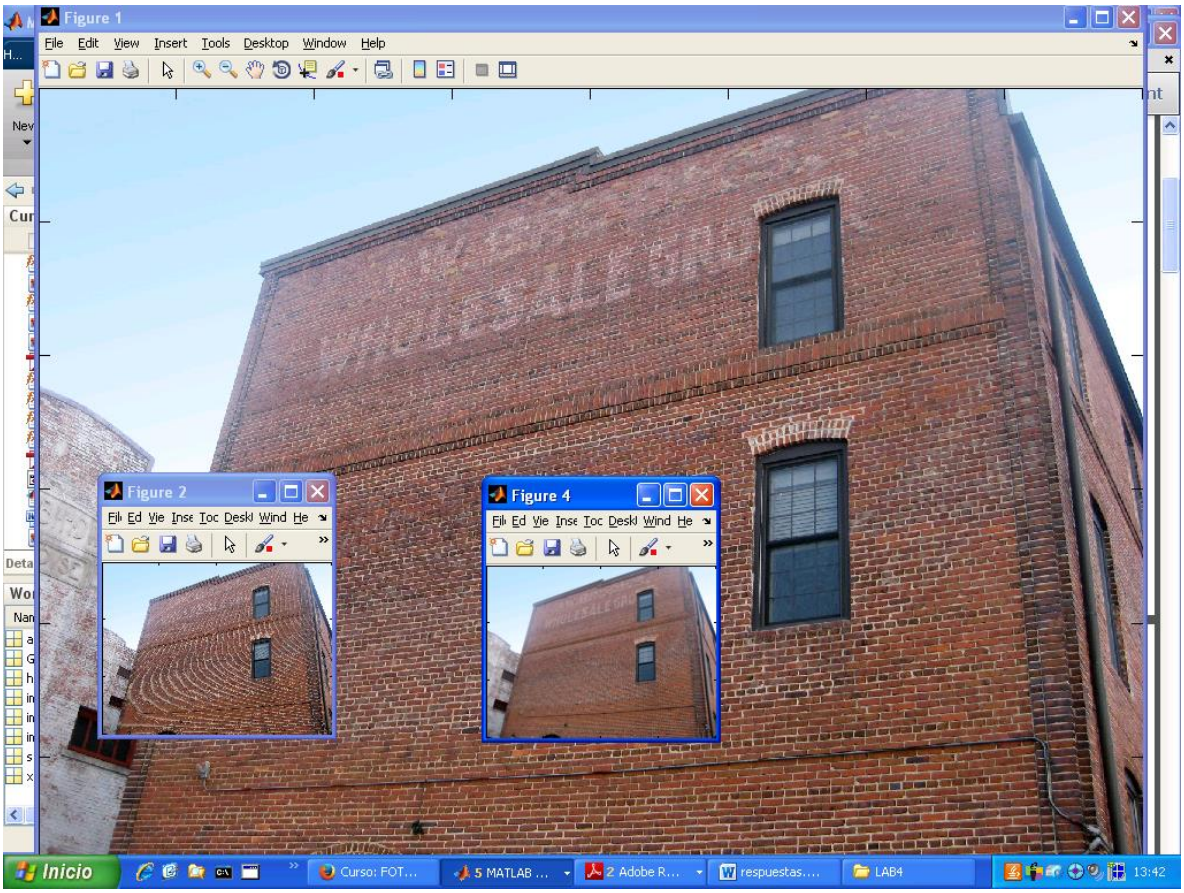


Figure 1
File Edit View Insert Tools Desktop Window Help
Figure 2
Figure 4
Inicio
Curso: FOT...
5 MATLAB ...
2 Adobe R...
respuestas....
LAB4
13:42

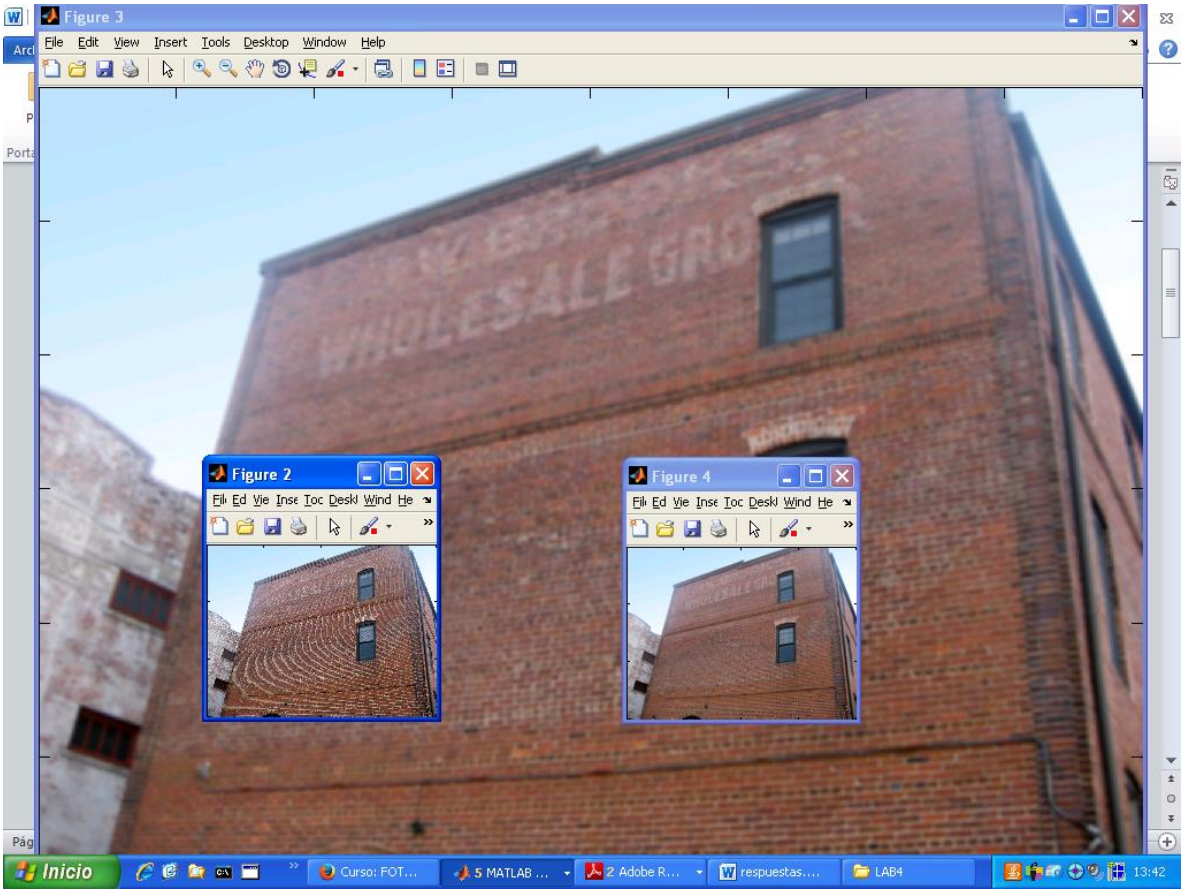


Figure 3
File Edit View Insert Tools Desktop Window Help
Figure 2
Figure 4
Inicio
Curso: FOT...
5 MATLAB ...
2 Adobe R...
respuestas....
LAB4
13:42

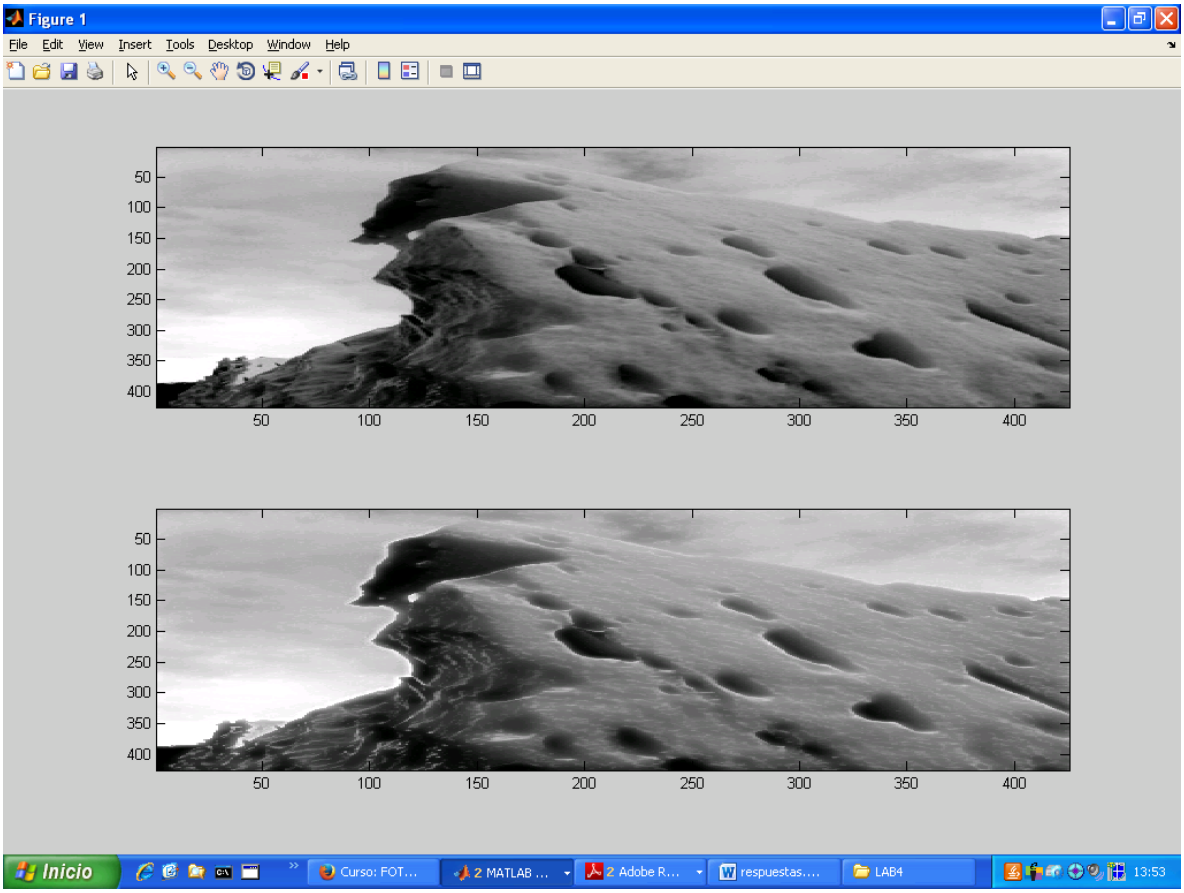
Código

```
im = imread('brick.jpg');
figure(1);image(im);
fc_truesize();

im2 = im(1:8:end,1:8:end,:);
figure(2);image(im2);
fc_truesize();


%% Filtro gausiano

s=3; x = [-5:5]; % sigma = 1, soporte = ?2:2 (N=5)
h = exp(-(x.^2)/(2*s.^2)); % Expresion gauss 1D
h = h/sum(h); % Normaliza para que suma=1
G = h' * h; % Crea mascara 2D


im3=fc_imfilter(im,G,'symmetric');
figure(3);image(im3);
fc_truesize();

im3 = im3(1:8:end,1:8:end,:);
figure(4);image(im3);
fc_truesize();
```

Ejercicio 2:

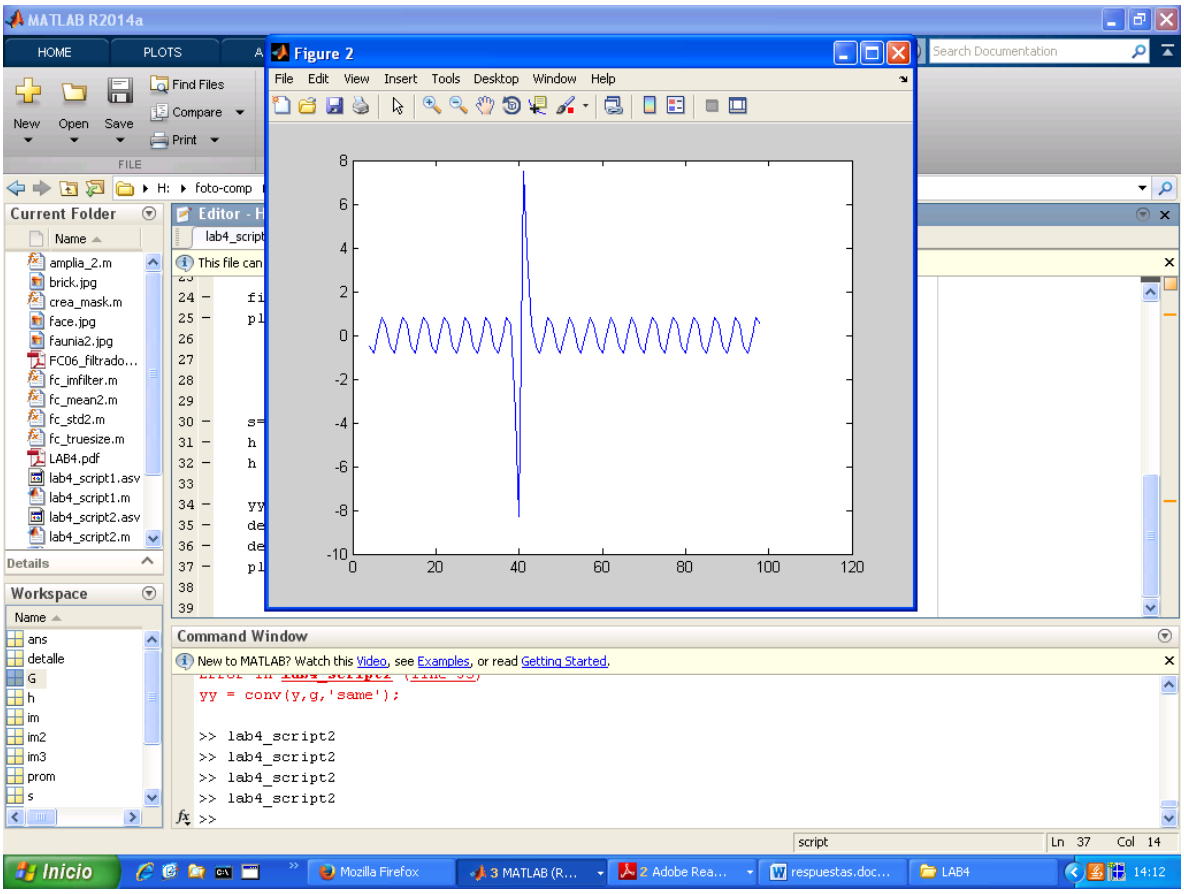
Adjuntar la figura resultante. Notad como la textura del hielo se ve más marcada en la segunda imagen. ¿Notáis algún efecto indeseable en la imagen procesada?

Si, el contorno de la textura de hielo tiene una degradación muy marcada.



Hacer un plot del "detalle" extraído y adjuntar imagen resultante.
¿Hemos capturado las pequeñas oscilaciones que simulaban el detalle de la imagen? ¿Qué más aparece en la gráfica?

Si hemos capturado las pequeñas oscilaciones de la imagen además del salto entre los bordes de la misma.



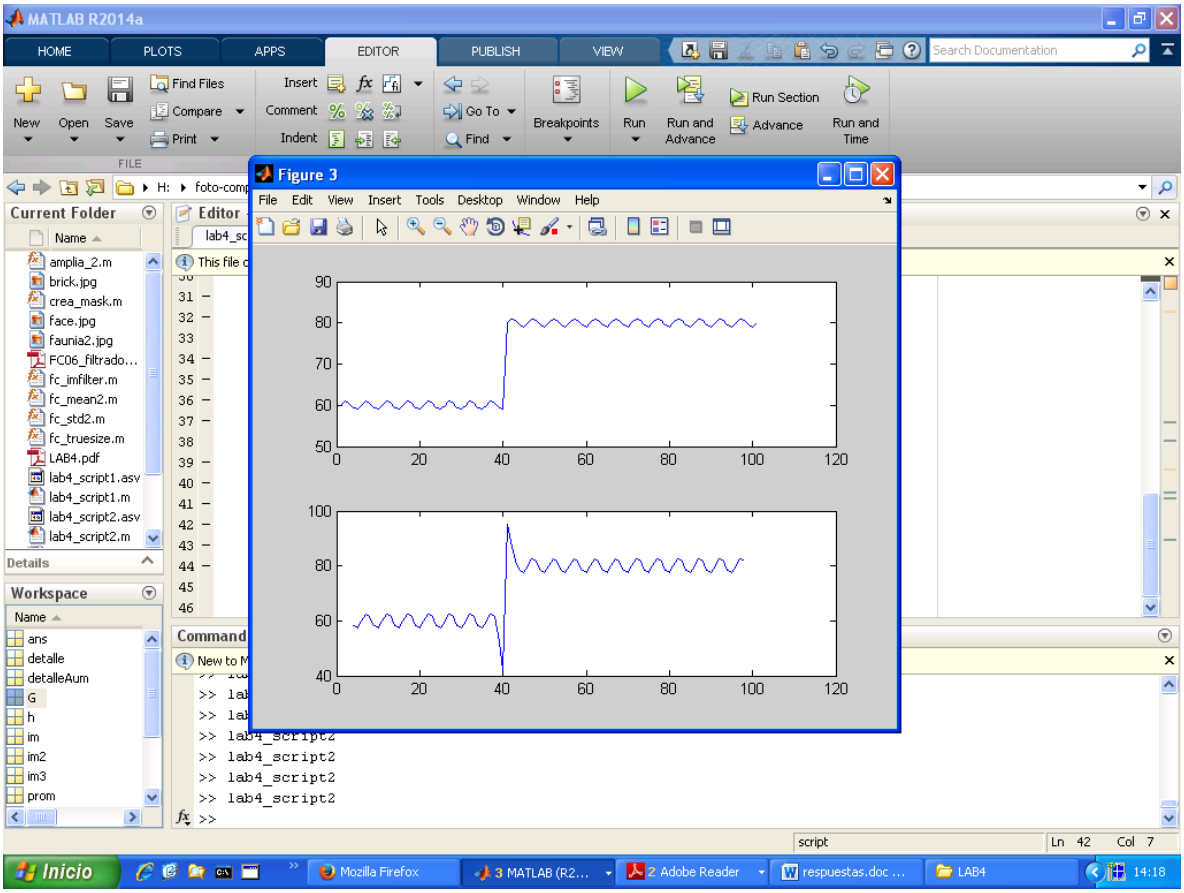
Superponer una "imagen" de la señal original (y) y de una señal con el detalle resaltado (yy + 3*detalle). Adjuntar la gráfica resultante. ¿Se ha magnificado el detalle? ¿Qué ocurre en el borde?

Si, se puede apreciar cómo se realzan los picos y los valles en la gráfica así como el "salto" que tiene de una forma mucho más marcada.

Código

```
% Filtro gausiano

s=3; x = [-5:5]; % sigma = 1, soporte = ?2:2 (N=5)
h = exp(-(x.^2)/(2*s.^2)); % Expresion gauss 1D
h = h/sum(h); % Normaliza para que suma=1
G = h' * h; % Crea mascara 2D

im = imread('test.jpg');

prom = fc_imfilter(im,G,'symmetric');

detalle = im - prom;

im2 = prom+2*detalle;

subplot(211);
image(im);
subplot(212);
```

```
image(im2);
colormap(gray(256));

t = (0:0.01:1); y(t<0.4)=60; y(t>=0.4)=80; y = y+sin(2*pi*t*20);

figure(2);
plot(t,y);

s=2; x = [-2:2]; % sigma = 1, soporte = ?2:2 (N=5)
h = exp(-(x.^2)/(2*s.^2)); % Expresion gauss 1D
h = h/sum(h); % Normaliza para que suma=1

yy = conv(y,h,'same');
detalle = y-yy;
detalle(1:3)=NaN; detalle(end-2:end)=NaN;
plot(detalle)

detalleAum=yy + 3*detalle;
figure(3);
subplot(211);
plot(y);
subplot(212);
plot(detalleAum);
```



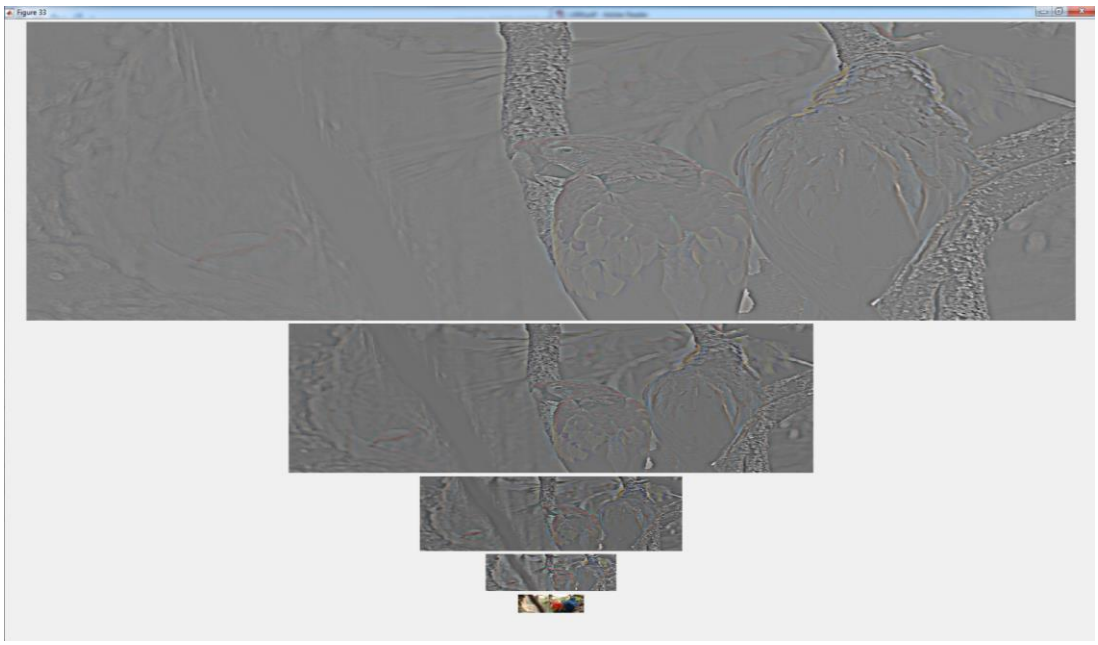
Proyecto: representaciones en pirámide. (a entregar, por parejas, dentro de dos semanas)

1) Implementación de la pirámide Laplaciana

Adjuntar código de vuestra función lap.m

```
function p=lap(im_original,N)
if nargin==1, N=5; end % Si numero de argumentos igual a uno asignamos a N 5
p=cell(1,N);

% Crear filtro gaussiano 2D a usar
s=3; x = -5:5; % sigma = 1, soporte = ?2:2 (N=5)
h = exp(-(x.^2)/(2*s.^2)); % Expresion gauss 1D
h = h/sum(h); % Normaliza para que suma=1
G = h' * h; % Crea mascara 2D

% Calcular los niveles p{1}, p{2}, ..., p{N} de la piramide laplaciana
% Recordad que p{N} es la versión reducida de la imagen
im_original=double(im_original);
for k=1:N-1,
    im_suavizada=fc_imfilter(im_original,G,'symmetric'); % Obtenemos la imagen
suavizada
    p{k} = im_original - im_suavizada; %obtenemos los detalles
    im_original = im_suavizada(1:2:end,1:2:end,:);
end

p{N} = im_original;

return
```

Adjuntar vuestra imagen de la pirámide resultante usando visualiza_lap



2) Inversa de la pirámide Laplaciana

Adjuntar código de vuestra función invlap.m

```
function im=invlap(p)
N = length(p); % Número de níveles de la piramide
im=p{N}; % Inicializamos con version tamaño sello.

% Ir sumando los detalles de los niveles p{k-1}, p{k-2}, ..., p{1} de la piramide
laplaciana
% Recordad que p{N} es la versión reducida de la imagen

for k=N-1:-1:1,
    im = amplia_2(im);
    im = im + p{k};
end

im = uint8(im);

end
```

Comando usado para calcular la discrepancia entre las imágenes original y recuperada. ¿De que orden es dicha diferencia?

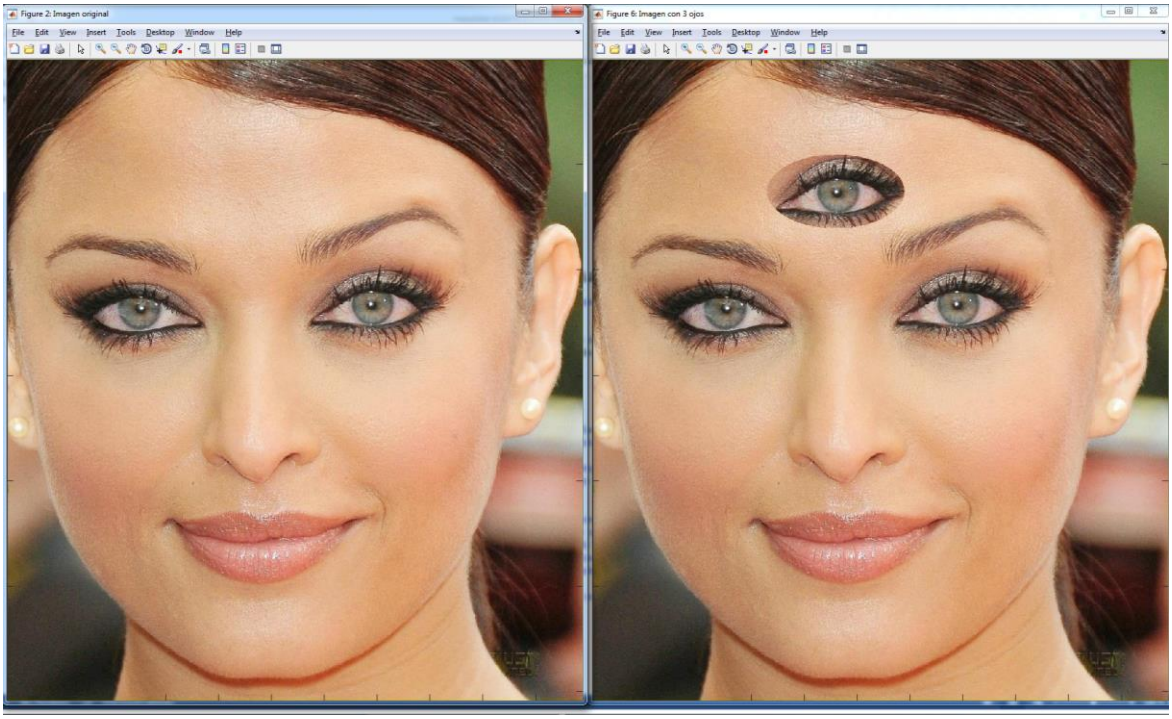
La diferencia es de 0.0015, tan sólo representa un 0.001% de la media de cada imagen.

Código empleado

```
im = imread('faunia2.jpg');
p = lap(im,5);
visualiza_lap(p);

im_invlap = invlap(p);
figure(2); image(im_invlap);
set(gcf, 'name', 'Imagen con invlap');

figure(3); image(im);
set(gcf, 'name', 'Imagen original');

media_original = mean2(im);
media_invlap = mean2(im_invlap);

diferencia = abs(media_original-media_invlap);
display(media_original);
display(media_invlap);
display(diferencia);
```

Aplicación: Fusión de imágenes con pirámide Laplaciana.

Adjuntar vuestra imagen im1. Hacer una fusión directa con la máscara binaria. Adjuntar código e imagen final.

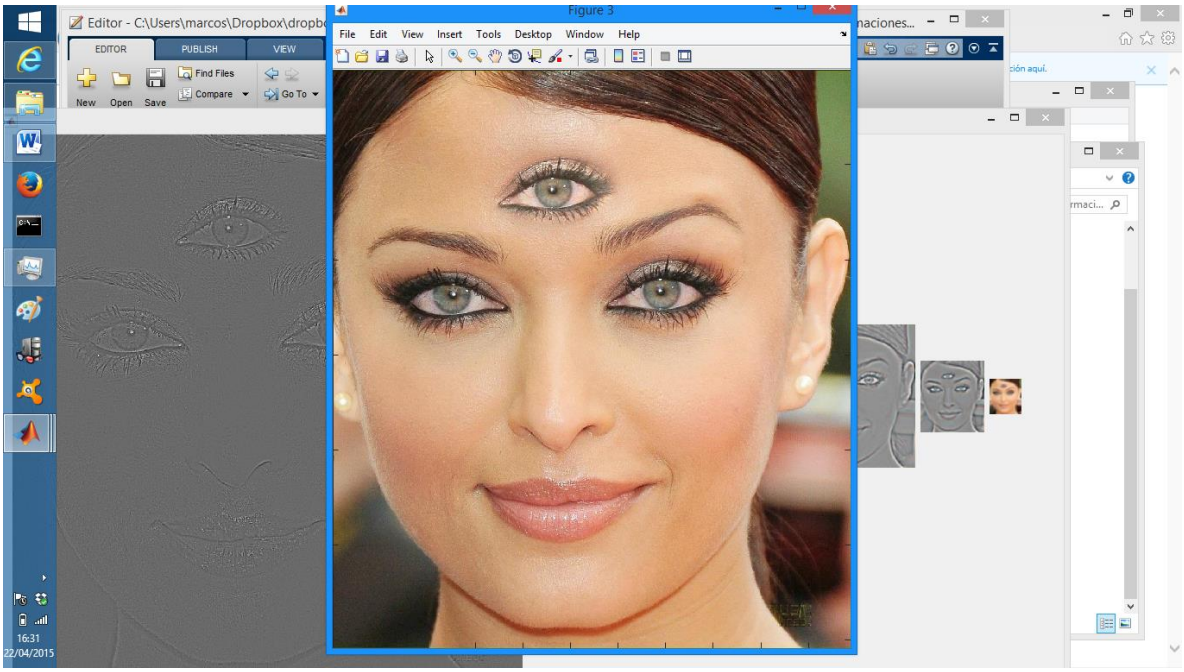


Código

```
im0 = imread('face.jpg');
figure(2); image(im0);
fc_truesize;
set(gcf, 'name', 'Imagen original');

X_tam = 300; Y_tam =210;
X0 = 537; Y0 = 351;
im2 = im0(Y0:Y0+Y_tam,X0:X0+X_tam,:); %imagen del ojo
%figure(3); image(im2);
%fc_truesize;

X_fr= 310; Y_fr=138;
im1 = im0; im1(Y_fr:Y_fr+Y_tam,X_fr:X_fr+X_tam,:) = im2(:,:,:);
%figure(4); image(im1);
%fc_truesize;

mask = crea_mask(im0,[X_fr+X_tam/2 Y_fr+Y_tam/2+8]);
%figure(5); image(mask);
%fc_truesize;

im = (1-mask).*double(im0) + mask.*double(im1);
im = uint8(im);
figure(6);image(im);
fc_truesize;
set(gcf, 'name', 'Imagen con 3 ojos');
```

Hacer fusión usando pirámides tal como se ha descrito anteriormente, usando 5 o 6 niveles en las pirámides (tanto en las pirámides laplacianas de las imágenes como en la pirámide gaussiana de la máscara).

Adjuntar vuestro código y la imagen final.



Código

```
function p=piramideGau(im_original,N)
if nargin==1, N=5; end % Si numero de argumentos igual a uno asignamos a N 5
p=cell(1,N);

% Crear filtro gaussiano 2D a usar
s=3; x = -5:5; % sigma = 1, soporte = ?2:2 (N=5)
h = exp(-(x.^2)/(2*s.^2)); % Expresion gauss 1D
h = h/sum(h); % Normaliza para que suma=1
G = h' * h; % Crea mascara 2D

% Calcular los niveles p{1}, p{2}, ..., p{N} de la piramide laplaciana
% Recordad que p{N} es la versión reducida de la imagen
im_original=double(im_original);
for k=1:N,
    im_suavizada=fc_imfilter(im_original,G,'symmetric'); % Obtenemos la imagen
suavizada
    p{k} = im_suavizada; %obtenemos los detalles
    im_original = im_suavizada(1:2:end,1:2:end,:);
end

return
```

ScriptAplicacion2

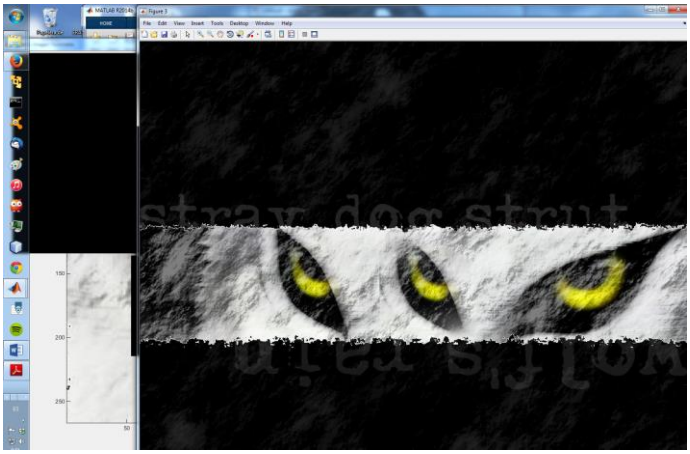
```
clear;
im0 = imread('face.jpg');
X_tam = 300; Y_tam =210;
X0 = 537; Y0 = 351;
im2 = im0(Y0:Y0+Y_tam,X0:X0+X_tam,:); %imagen del ojo
X_fr= 310; Y_fr=138;
im1 = im0; im1(Y_fr:Y_fr+Y_tam,X_fr:X_fr+X_tam,:) = im2(:,:,:);
mask = crea_mask(im0,[X_fr+X_tam/2 Y_fr+Y_tam/2+8]);
%im = (1-mask).*double(im0) + mask.*double(im1);
im1 = uint8(im1);

%figure(1); image(im0);
%figure(2); image(im1);

%Fusion por bandas

p0 = lap(im0,5); %piramide lapciana de im0
p1 = lap(im1,5); %piramide lapciana de im0
mask_cell = piramideGau(mask,5); %piramide lapciana de mascara gaussiana

new=cell(1,5);

for k = 1:5,
 % Piramide laplaciana compuesta
 new{k} = (1- mask_cell{k}).*p0{k} + mask_cell{k}.*p1{k};
end

figure(1); visualiza_lap(mask_cell);
figure(2); visualiza_lap(new);

imagen_fusionada = invlap(new);


set(gcf, 'name', 'Imagen fusionada');
figure(3);image(imagen_fusionada);
fc_truesize();
```

EXTRAS: Cuando hayáis replicado estos resultados, usar vuestro programa con una foto o fotos de vuestra elección. Tened en cuenta que:

- Dependiendo del objeto a "fusionar" seguramente tendréis que cambiar el tamaño de la máscara usando el tercer argumento opcional de la función crea_mascara explicado antes.

- Vuestra función lap (e inv_lap) solo va a funcionar correctamente si las dimensiones (alto/ancho) de la imagen son múltiplos de una potencia de 2 suficientemente alta para seguir siendo enteros después de las sucesivas divisiones por 2 en cada nivel de la pirámide. Recortar las imágenes a usar para que tengan las dimensiones adecuadas.

¿Cómo podríais modificar lap (e invlap) para que funcionaran con imágenes de un tamaño arbitrario?



Para realizar la tercera parte hemos cogido una imagen distinta y hemos cambiado las medidas de im2 y el vector w además de las referencias X0, Y0, X_tam, Y_tam, X_fr y Y_fr. Al ser cogido de un fondo de pantalla ha sido fácil encontrar una foto con un múltiplo de 2 bastante alto.

Sin embargo para modificar lap e invlap de manera que funcione con cualquier imagen habría que tener en cuenta la resolución. Una forma fácil de hacerlo sería guardar en un array, de tamaño igual al número de niveles de la pirámide, si se cambia la resolución o no en cada nivel. Para obtener la resolución correcta basta con restar una fila o columna a una dimensión que sea impar.

Código cambiado

```
clear;
im0 = imread('wolf_rain3.jpg');
X_tam = 265; Y_tam =315;
X0 = 355; Y0 = 550;
im2 = im0(Y0:Y0+Y_tam,X0:X0+X_tam,:); %imagen del ojo
X_fr= 701; Y_fr= 560;
im1 = im0; im1(Y_fr:Y_fr+Y_tam,X_fr:X_fr+X_tam,:) = im2(:,:,:);
w=[130 170];%w=[130 70]
mask = crea_mask(im0,[X_fr+X_tam/2 Y_fr+Y_tam/2+8],w);
```