Automatizar la creación de un mosaico de fotos (entrega por parejas para el 7 junio)

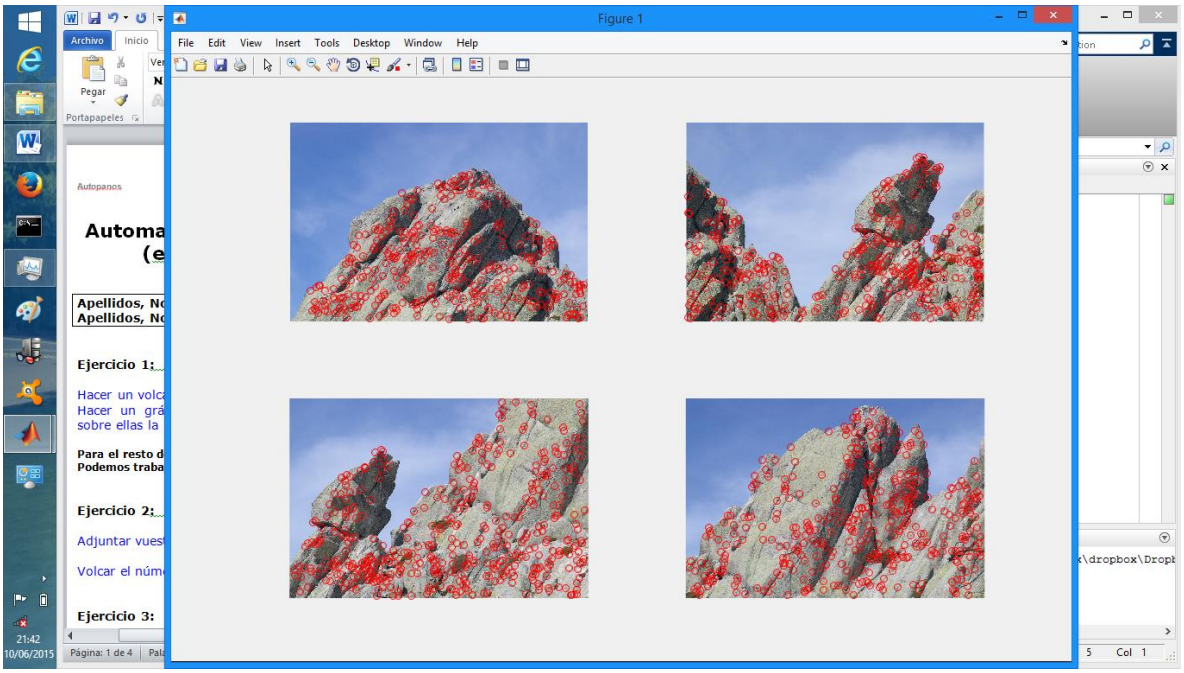
Apellidos, Nombre: Bernal España, Marcos
Apellidos, Nombre: De la Hoz Simon, Yolanda

Ejercicio 1: (Obtención de los keypoints usando algoritmo SIFT)

Hacer un volcado con el número de los "keypoints" guardados para cada imagen.

```
La imagen 1 tiene 390 keypoints
La imagen 2 tiene 407 keypoints
La imagen 3 tiene 486 keypoints
La imagen 4 tiene 431 keypoints
La imagen 5 tiene 514 keypoints
La imagen 6 tiene 906 keypoints
La imagen 7 tiene 725 keypoints
La imagen 8 tiene 760 keypoints
La imagen 9 tiene 712 keypoints
La imagen 10 tiene 456 keypoints
```

Hacer un gráfico de las 4 primeras imágenes (usar subplot(22x)) superponiendo sobre ellas la posición de sus keypoints como círculos rojos ('ro'). Adjuntar imagen.



Para el resto de la práctica ya no es necesario usar SiftWin32.exe ni trabajar en Windows. Podemos trabajar a partir de la información guardada en s{} haciendo load keypoints

Ejercicio 2: (hallar emparejamientos entre dos imágenes)

Adjuntar vuestro código de find_matches.m

```
function [xy1, xy2] = find_matches(s1,s2)
xy1 = [];
xy2 = [];

tamano_s1 = size(s1.desc);
tamano_s2 = size(s2.desc);

for num_desc_s1 = 1:tamano_s1(1)
  desc_s1 = s1.desc(num_desc_s1,:);
  %distancias = zeros(tamano_s2(1),1);
  Dmin = norm(desc_s1 - s2.desc(1,:));
  PosMin = s2.xy(1,:);
  Dmin2 = norm(desc_s1 - s2.desc(1,:))*5;

  for num_desc_s2 = 1:tamano_s2(1)
      distancia = norm(desc_s1 - s2.desc(num_desc_s2,:));
      if Dmin > distancia
          Dmin2 = Dmin;
          Dmin = distancia;
          PosMin = s2.xy(num_desc_s2,:);
      else
          if Dmin2 > distancia
              Dmin2 = distancia;
          end
      end

  end

  if (Dmin2/Dmin) >= 3
      xy1(end + 1,:) =  s1.xy(num_desc_s1,:);
      xy2(end + 1,:) =  PosMin;
  end
end
```

Volcar el número emparejamientos encontrados entre la imagen 1 y el resto.

Código

```
for k = 2:10
[xy1, xy2] = find_matches(s{1},s{k});
fprintf('Hemos encontrado %d parejas de puntos entre la imagen 1 y la imagen
%d\n',length(xy1),k);
end
```

Salida

```
Hemos encontrado 36 parejas de puntos entre la imagen 1 y la imagen 2
Hemos encontrado 0 parejas de puntos entre la imagen 1 y la imagen 3
Hemos encontrado 0 parejas de puntos entre la imagen 1 y la imagen 4
Hemos encontrado 0 parejas de puntos entre la imagen 1 y la imagen 5
Hemos encontrado 0 parejas de puntos entre la imagen 1 y la imagen 6
```

```
Hemos encontrado 0 parejas de puntos entre la imagen 1 y la imagen 7
Hemos encontrado 38 parejas de puntos entre la imagen 1 y la imagen 8
Hemos encontrado 96 parejas de puntos entre la imagen 1 y la imagen 9
Hemos encontrado 80 parejas de puntos entre la imagen 1 y la imagen 10
```

Ejercicio 3:

Hacer las modificaciones oportunas para que no get_P3 funcione correctamente para N>=3 pares de datos. Adjuntar código de vuestra función.

```
function P=get_P3(xy,xyp)
x = xy(:,1); y =xy(:,2);
xp = xyp(:,1); yp=xyp(:,2); % Extraigo datos origen (x,y) y destino (xp,yp)
tamano = size(x);
H=[x, y, ones(1,tamano(1))'];
 % matriz H cogemos de 3x3 en vez de 6x6

c1 = H\xp;
c2 = H\yp;

P = [c1';
     c2';
     0 0 1]; % Obtenemos matriz de coeficientes

return
```

Comprobad que en este caso al aplicar P a xy1 no obtenemos exactamente xy2.

```
P = get_P3(xy1,xy2);
xy2_prima = P*[xy1'; ones(1,length(xy1))];
xy2_prima = xy2_prima(1:2,:);
xy2_prima == xy2'
```

Cuando sale el resultado por pantalla obtenemos que todos son falsos (0's).

Adjuntar código de vuestra función error_ajuste

```
function err=error_ajuste(xy1,xy2,P)
% Tumbamos la matriz y añadimos una columna de unos para poder
% multiplicarla con la matriz P
longitud = size(xy1);
err_2 = ([xy2'; ones(1,longitud(1))] - P*[xy1'; ones(1,longitud(1))])';
err = err_2(:,1:2);
end
```

Volcar el resultado de aplicarla a los datos del ejemplo.
¿Cuál es el error máximo cometido para los puntos anteriores? ¿En qué punto?

```
P = get_P3(xy1,xy2);
error = error_ajuste(xy1,xy2,P);
[Valor, posicion] = max(error)
```

Tenemos el mayor error del eje x en el punto 37, para el eje y se sitúa en el punto 70. Tiene un valor de 1.7895 y 0.7663 respectivamente.

Ejercicio 4 (algoritmo RANSAC):

Adjuntar código de función.

```
function [Nmax,T]=ransac_fun(xy1,xy2)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

num_parejas = size(xy1);
Nmax = 0; select = [];

for k = 1:500
    % Hallamos un numero aleatorio entre 1 y num_parejas
    indice1 = 1 + fix(rand(1)*num_parejas(1)-1);
    indice2 = 1 + fix(rand(1)*num_parejas(1)-1);
    indice3 = 1 + fix(rand(1)*num_parejas(1)-1);
    if (indice1 == indice2 || indice2 == indice3 || indice1 == indice3)
       continue;
    end

    %Buscamos aleatoriamente tres puntos
    coordenada_xy1 = [xy1(indice1,:); xy1(indice2,:); xy1(indice3,:)];
    coordenada_xy2 = [xy2(indice1,:); xy2(indice2,:); xy2(indice3,:)];

    % Buscamos matriz de transformacion
    T = get_P3(coordenada_xy1, coordenada_xy2);
    err=error_ajuste(xy1,xy2,T);

    error_normalizado = zeros(1,num_parejas(1));
    for j = 1:num_parejas(1)-1
      error_normalizado(j) = norm(err(j,:));
    end
    % Si encontramos
    if (length(find(error_normalizado < 0.5)) > Nmax)
        Nmax = length(find(error_normalizado < 0.5));
        select = [indice1, indice2, indice3];
    end
end

coordenada_xy1 = [xy1(select(1),:); xy1(select(2),:); xy1(select(3),:)];
coordenada_xy2 = [xy2(select(1),:); xy2(select(2),:); xy2(select(3),:)];
T = get_P3(coordenada_xy1, coordenada_xy2);
%fprintf('Numero de parejas aceptadas: %d\n', Nmax);
%fprintf('Puntos escogidos:');
%display(coordenada_xy1);
%display(coordenada_xy2);




end
```

Adjuntar matriz obtenida + N "inliers" tras aplicar el algoritmo a las 45 potenciales parejas entre imagen 1 e imagen 2 obtenidas con la función find_matches() en el ejercicio 2.

Código:

```
load('keypoints.mat');
[xy1, xy2] = find_matches(s{1},s{2});
[Nmax,T]=ransac_fun(xy1,xy2)
```

Salida:

```
Nmax =

    30


T =

    0.9930   -0.0375 -615.5796
    0.0227    1.0003  -83.0164
         0         0    1.0000
```

Ejercicio 5: (determinar conectividad entre las imágenes).

Adjuntar código de find_QP()

```
function [Q P]=find_QP(s)

  Nfotos_matrix = size(s);
  Nfotos = Nfotos_matrix(2);
  Q = zeros(Nfotos);
  P = cell(Nfotos);

  for i=1:Nfotos
     for j=(i+1):Nfotos
       [xy1, xy2] = find_matches(s{i},s{j});
       tamano = size(xy1);
       if tamano(1) > 5
          [Nmax,T]=ransac_fun(xy1,xy2);
          if Nmax > 5
              Q(i,j) = Nmax;
              Q(j,i) = Nmax;
              P{i,j} = T;
              P{j,i} = inv(T);
              fprintf('Tenemos %d puntos entre las imagenes %d y
%d\n',tamano(1),i,j);
              fprintf('Entre los cuales estan aceptados %d', Nmax);
          end
       end
     end
  end
end
```

Volcar la matriz Q de "conectividad" obtenida para las 10 imágenes.

```
Q =

     0    29     0     0     0     0     0    33    46    49
    29     0    56     0     0     0    19    97    35     0
     0    56     0    27     0     0    71    56     0     0
     0     0    27     0    32    18    31     0     0     0
     0     0     0    32     0    50     0     0     0     0
     0     0     0    18    50     0    24     0     0     0
     0    19    71    31     0    24     0    54     0     0
    33    97    56     0     0     0    54     0    79     0
    46    35     0     0     0     0     0    79     0    52
    49     0     0     0     0     0     0     0    52     0
```

¿Cómo podríamos detectar en MATLAB que una de las imágenes de partida no está conectada a las demás.

En la matriz Q se indica el número de parejas de puntos que comparte con el resto de imágenes. La fila o columna n, simboliza la relación de la imagen n con el resto de imágenes, por lo que si esta línea/columna sólo tiene ceros podríamos determinar que la imagen no tienen ningún punto de conexión con el resto de imágenes.

Ejercicio 6:

Usando el criterio de conectividad al mayor número de imágenes ¿cómo podemos determinar la imagen ancla a partir de la matriz Q? Adjuntar vuestro código y el índice (1-10) de la imagen para usar como ancla.

Se puede realizar de varias maneras, aunque en el código se toma como criterio la imagen que más esté conectada a otras.

Adjuntar código

```
function T = ordena(Q,P)

Nimag_matrix = size(Q);
Nimag=Nimag_matrix(1);

% Buscamos el ancla, imagen que mas conexiones tenga con el resto
A = Q>0;
[M,ancla]=max(sum(A));
fprintf('Imagen %d es el ancla.\n',ancla)

% Vector de booleanos para llevar la cuenta
usadas = false(1,Nimag);
usadas(ancla)=true;

% La transformacion para la matriz ancla sera la matriz identidad
T{ancla}= eye(3);
fotos = 1:Nimag;

while not(all(usadas)),

    A = zeros(Nimag);
    for fila=1:Nimag,
        for colum=1:Nimag,
            if(usadas(fila) & ~usadas(colum))
                A(fila,colum)=Q(fila,colum);
            end
        end
    end

    [M,index]=max(A(:)); [fila,colum]=ind2sub(size(A),index);
    if usadas(fila(1))
        nueva_imagen=colum(1);
        ref_lista=fila(1);
    else
        nueva_imagen=fila(1);
        ref_lista=colum(1);
    end
    T{nueva_imagen}=P{nueva_imagen,ref_lista}*T{ref_lista};
    usadas(nueva_imagen)=true;

    colocadas = sprintf('%d',fotos(usadas));
    fprintf('Imagen %d enlaza con %d. COLOCADAS:
%s.\n',nueva_imagen,ref_lista,colocadas);
end
```

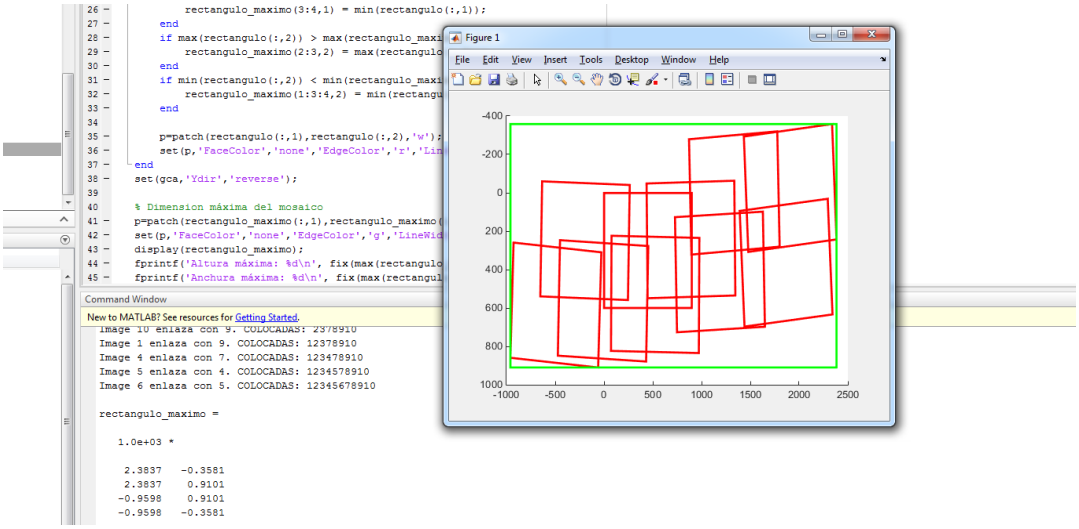
```
Q(:,ancla) =

    30
     0
    58
     0
     0
     0
    18
    97
    34
     0
```

Adjuntar volcado del orden en el vuestras imágenes van "colocándose".

```
Imagen 2 es el ancla.
Imagen 8 enlaza con 2. COLOCADAS: 28.
Imagen 9 enlaza con 8. COLOCADAS: 289.
Imagen 3 enlaza con 2. COLOCADAS: 2389.
Imagen 7 enlaza con 3. COLOCADAS: 23789.
Imagen 10 enlaza con 9. COLOCADAS: 2378910.
Imagen 1 enlaza con 9. COLOCADAS: 12378910.
Imagen 4 enlaza con 7. COLOCADAS: 123478910.
Imagen 5 enlaza con 4. COLOCADAS: 1234578910.
Imagen 6 enlaza con 5. COLOCADAS: 12345678910.
```

Ejercicio 7:

Adjuntar vuestra propia imagen obtenida mostrando el solapamiento y la posición de las imágenes en el mosaico.



Dar los valores Xmin,Xmax,Ymin,Ymax obtenidos para vuestro mosaico y las correspondientes dimensiones (ancho/alto) y desplazamientos (DX,DY) a usar. Indicad si habéis usado margen y cuál.

```
Altura minima: -358 Altura maxima: 910
Anchura minima: -959 Anchura maxima: 2383

Altura total: 1268
Anchura total: 3343

Dx: 959.00000
Dy: 358.00000

No se ha usado margen.
```

Ejercicio 8:

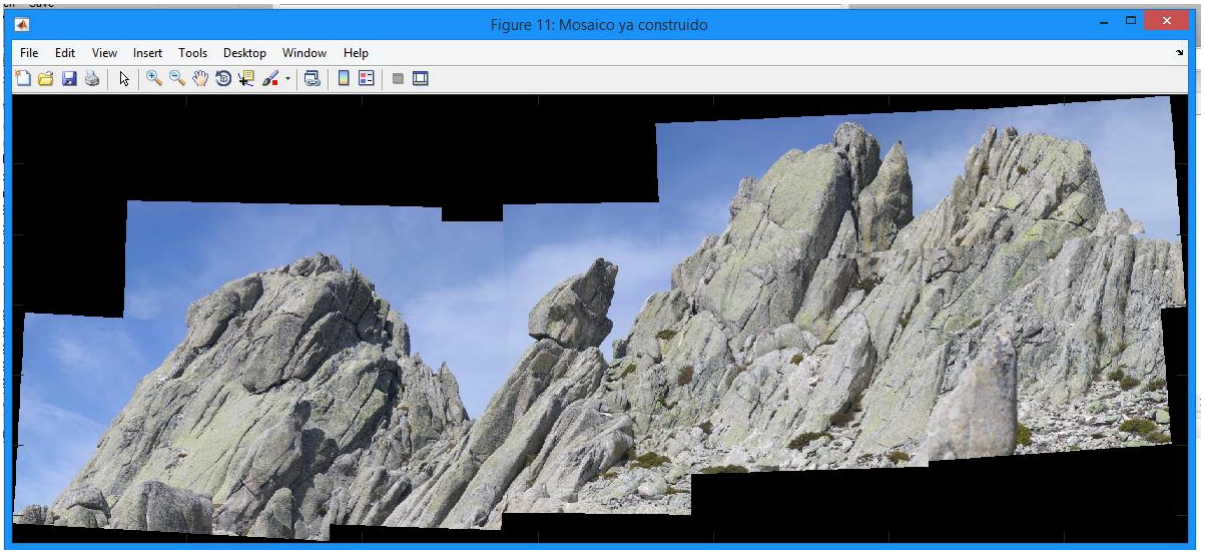
Adjuntar código de vuestra función interpola + código del bucle de "montaje" de las imágenes.

```
function im = interpola(im,P,rx,ry)
tamano_matrix_y = size(ry);
tamano_y = tamano_matrix_y(2);
tamano_matrix_x = size(rx);
tamano_x = tamano_matrix_x(2);

im=double(im); % Ahora deja las cosas como están

Xi = zeros(tamano_y,tamano_x);
Yi = zeros(tamano_y,tamano_x);

for pos_y = 1:tamano_y,
    for pos_x = 1:tamano_x,
      vec = [rx(pos_x) ry(pos_y) 1]'; % hallamos vector de coordenadas homogéneas.
      vec = P*vec; % aplicamos P al vector
      Xi(pos_y,pos_x)= vec(1,1);
      Yi(pos_y,pos_x) = vec(2,1);
    end
end

im2(:,:,1) = interp2(im(:,:,1),Xi,Yi,'bilinear');
im2(:,:,2) = interp2(im(:,:,2),Xi,Yi,'bilinear');
im2(:,:,3) = interp2(im(:,:,3),Xi,Yi,'bilinear');

im = uint8(im2);

return
```

```
load('zeldas.mat');
load('rectangulo_maximo_negativo.mat');
rectangulo_maximo_negativo = rectangulo_maximo;
load('rectangulo_maximo_justado.mat');
load('orden_del_mosaico');
T = ordena(Q,P);

tamano_matrix = size(T);
NImag = tamano_matrix(2);

dx = -fix(min(rectangulo_maximo_negativo(:,1)));
dy = -fix(min(rectangulo_maximo_negativo(:,2)));
txy = [ 1 0 dx+5;
        0 1 dy+5;
        0 0 1];

for k=1:NImag
     T{k} = txy*T{k};
end
```

```
mosaico = zeros(10+fix(max(rectangulo_maximo(:,2))-min(rectangulo_maximo(:,2))),
10+fix(max(rectangulo_maximo(:,1))-min(rectangulo_maximo(:,1))),3);


Rx = [1 900 900 1]; Ry = [1 1 600 600];


for i=1:NImag,

    [N,M]=size(Rx);

    % Iniciamos el retangulo y lo rellenamos
    rectangulo = zeros(2*N,M)';
    for k=1:M,
            vertice = T{i}*[Rx(1,k); Ry(1,k);1];
            rectangulo(k,:)= [vertice(1),vertice(2)];
    end

    rx = fix(min(rectangulo(:,1))):fix(max(rectangulo(:,1)));
    ry = fix(min(rectangulo(:,2))):fix(max(rectangulo(:,2)));
    if i ~= 10
        nombre_imagen = sprintf('image0%d.jpg',i);
    else
        nombre_imagen = sprintf('image%d.jpg',i);
    end
    im = imread(nombre_imagen);
    im2 = interpola(im,inv(T{i}),rx,ry);
    %figure(1);image(im);
    %set(gcf, 'name', 'Imagen original');
    %figure(2);image(im2);
    %set(gcf, 'name', 'Imagen interpolada');

    %cogemos solo los datos que necesitamos y lo ponemos en una parte que
    % que "recortamos" previamente del mosaico
    mosaico_copia = mosaico(ry,rx,:);
    mosaico_copia(im2 > 0) = im2(im2 > 0);
    mosaico(ry,rx,:) = mosaico_copia;

    % Pasamos la imagen a 8bits para visualizarla
    mosaico = uint8(mosaico);
    figure(i);image(mosaico);
    set(gcf, 'name', 'Mosaico');
    set(gca,'Ydir','reverse');

    % Pintamos el borde de la nueva imagen
    p=patch(rectangulo(:,1),rectangulo(:,2),'w');
    set(p,'FaceColor','none','EdgeColor','r','LineWidth',2);
end
set(gca,'Ydir','reverse');

figure(11);image(mosaico);
set(gcf, 'name', 'Mosaico ya construido');
fc_truesize();
```

Adjuntar vuestro mosaico final. Usar fc_truesize para respetar sus proporciones.



Además de incluir los trozos de código pedidos en el fichero de respuestas, en la entrega final **incluir todo el código necesario para que funcione el programa.**

Entregadlo en un .rar o similar junto con la hoja de respuestas.

Para visualizarlo ejecutar el script *lab6_script8.m* o directamente el código que hay arriba en la carpeta proporcionada. Además se muestran 10 figuras más donde se van añadiendo las imágenes una a una.

EXTRAS:

Si lo habéis hecho todo correctamente debería funcionar automáticamente para cualquier otro conjunto de imágenes con mínimos cambios (cambiando el nombre y número de imágenes, su tamaño, etc.). Procurar que todos estos parámetros estén bien etiquetados en vuestro programa para que sea sencillo cambiar su valor.

Hacer un mosaico de la facultad (perdón, escuela) o de alguno de los edificios del campus. Algunas recomendaciones:

- Tomar un mínimo de 6 fotos, usando la focal lo más larga (tele) posible. No las hagáis desde muy cerca.

- Reducirlas a tamaño 600 x 900 (formato 3/2) o 600 x 800 (formato 4/3) para que sea todo más rápido (usar IrfanView o similar). La función imresize de MATLAB también puede valer. Acordaros de que hay partes de vuestro código que dependen del tamaño de las imágenes empleadas.

- Usar SiftWin32 para obtener lista de keypoints (ejercicio 3). Recordad:

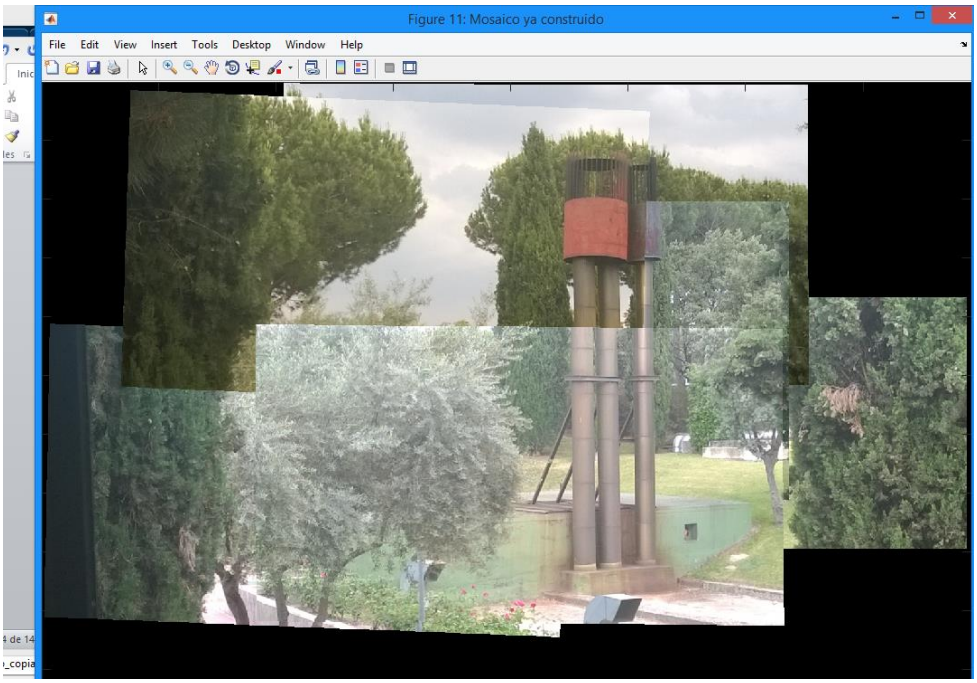
  Pasar la imagen a niveles de gris, guardar como pgm, aplicar Sift32Win.
  Parsear fichero de keypoints para leer datos.
  Reducir el número de puntos seleccionando un rango de escalas dado.

- Guardar los datos en un array de celdas s{1}, s{2}, etc.

- Aplicar el programa tal como lo tenéis.

Adjuntar FOTOS + programa completo + Fichero .mat que guarde los keypoints para poder obviar la fase de la extracción de parámetros.

Si tenéis los datos en una array s{} basta hacer >> save datos_puntos s



El código se encuentra en el fichero "ejercicio1_extra.m" más abajo se encuentra un extracto del mismo sin comentarios.

Las fotos se encuentran en la carpeta fotos_propias del zip. Si bien podríamos habernos fijado y realizar mejor las fotos, tampoco disponíamos de los recursos óptimos para realizarlo.

Código

```
name='fotos_propias/propia_0';

load('keypoints_propios.mat');

% [Q P]=find_QP(s);

% save zeldas_propio Q P;

load('zeldas_propio.mat');

T = ordena(Q,P);

Rx = [1 900 900 1]; Ry = [1 1 600 600];
tamano_matrix = size(Q);
tamano = tamano_matrix(2);
rectangulo_maximo = zeros(4,2);


for i=1:tamano,

    [N,M]=size(Rx);
```

```
    % Iniciamos el retangulo y lo rellenamos
    rectangulo = zeros(2*N,M)';
    for k=1:M,
            vertice = T{i}*[Rx(1,k); Ry(1,k);1];
            rectangulo(k,:)= [vertice(1),vertice(2)];
    end

    if max(rectangulo(:,1)) > max(rectangulo_maximo(:,1))
        rectangulo_maximo(1:2,1) = max(rectangulo(:,1));
    end
    if min(rectangulo(:,1)) < min(rectangulo_maximo(:,1))
        rectangulo_maximo(3:4,1) = min(rectangulo(:,1));
    end
    if max(rectangulo(:,2)) > max(rectangulo_maximo(:,2))
        rectangulo_maximo(2:3,2) = max(rectangulo(:,2));
    end
    if min(rectangulo(:,2)) < min(rectangulo_maximo(:,2))
        rectangulo_maximo(1:3:4,2) = min(rectangulo(:,2));
    end

    p=patch(rectangulo(:,1),rectangulo(:,2),'w');
    set(p,'FaceColor','none','EdgeColor','r','LineWidth',2);
end
set(gca,'Ydir','reverse');

% % Dimensiones
 p=patch(rectangulo_maximo(:,1),rectangulo_maximo(:,2),'w');
 set(p,'FaceColor','none','EdgeColor','g','LineWidth',2);
save rectangulo_maximo_negativo_propio rectangulo_maximo;

load('rectangulo_maximo_negativo_propio.mat');

tamano_matrix = size(T);
NImag = tamano_matrix(2);

dx = -fix(min(rectangulo_maximo(:,1)));
dy = -fix(min(rectangulo_maximo(:,2)));
txy = [ 1 0 dx+3;
        0 1 dy+3;
        0 0 1];

for k=1:NImag
     T{k} = txy*T{k};
end

mosaico = zeros(10+fix(max(rectangulo_maximo(:,2))-min(rectangulo_maximo(:,2))),
10+fix(max(rectangulo_maximo(:,1))-min(rectangulo_maximo(:,1))),3);


Rx = [1 900 900 1]; Ry = [1 1 600 600];


for i=1:NImag,

    [N,M]=size(Rx);

    % Iniciamos el retangulo y lo rellenamos
    rectangulo = zeros(2*N,M)';
```

```
    for k=1:M,
            vertice = T{i}*[Rx(1,k); Ry(1,k);1];
            rectangulo(k,:)= [vertice(1),vertice(2)];
    end

    rx = fix(min(rectangulo(:,1))):fix(max(rectangulo(:,1)));
    ry = fix(min(rectangulo(:,2))):fix(max(rectangulo(:,2)));

    fich=sprintf('%s%02d.jpg',name,i);

    im = imread(fich);
    im2 = interpola(im,inv(T{i}),rx,ry);

    %cogemos solo los datos que necesitamos y lo ponemos en una parte que
    % que "recortamos" previamente del mosaico
    mosaico_copia = mosaico(ry,rx,:);
    mosaico_copia(im2 > 0) = im2(im2 > 0);
    mosaico(ry,rx,:) = mosaico_copia;

    % Pasamos la imagen a 8bits para visualizarla
    mosaico = uint8(mosaico);
    figure(i);image(mosaico);
    set(gcf, 'name', 'Mosaico');
    set(gca,'Ydir','reverse');

    % Pintamos el borde de la nueva imagen
    p=patch(rectangulo(:,1),rectangulo(:,2),'w');
    set(p,'FaceColor','none','EdgeColor','r','LineWidth',2);
end

figure(11);image(mosaico);
set(gcf, 'name', 'Mosaico ya construido');
fc_truesize();
```

Otras posibles mejoras:

- Prever el caso de que 1 o más imágenes no pertenezcan al panorama. Tras construir la matriz Q el programa debería avisarnos de que imágenes se quedan fuera. Podríamos marcarlas con un NaN en la T{k} y usar isnan() para detectarlas luego e ignorarlas en las fases posteriores.

- Usar transformaciones proyectivas en vez de afines. En las transparencias del tema de transformadas de dominio tenéis indicaciones sobre como determinar los parámetros de una transformación proyectiva.