Proyecto de Transformaciones Geométricas

Apellidos, Nombre: Fernández Cabrita, Alejandro
Apellidos, Nombre: Evangelista Sarabia, Carlos
Apellidos, Nombre: De Almeida Pissarra, Inês

1) Deformaciones lineales (30%)

Obtención de la matriz P de una transformación proyectiva

Volcad la matriz P obtenida (usad la función dump_mat(P) que os doy)

```
  0.6975   -0.7094  275.0000
  0.4166    0.0266   30.0000
  0.0001   -0.0011    1.0000
```

Adjuntad código de get_proy.m.

```
function P=get_proy(xy,uv)

    Z=zeros(4,3);
    A=[xy(1,:)' xy(2,:)' ones(4,1)];
    b=[uv(1,:)' uv(1,:)' uv(1,:)'];
    B=-A.*b;
    c=[uv(2,:)' uv(2,:)' uv(2,:)'];
    C=-A.*c;

    M = [A Z B; Z A C];

    [U,S,V]=svd(M);
    h = V(:,9);
    P = reshape(h,3,3)';
    P = P/P(3,3);

end
```

Adjuntad los valores de las diferencias entre uv y uv2.

```
0              2.855e-08      -3.568e-08      1.635e-09
4.158e-09      1.141e-08      -6.700e-08      -1.073e-08
```

¿A qué coordenadas u,v iría a parar un punto con coordenadas (x=200,y=100) al aplicarle la transformación dada por P?

u = 376.4211
v = 127.0767

¿Son iguales? Justificad este resultado con las propiedades que vimos de las matrices proyectivas.

P_inversa (iP) y con inv(P)
```
  0.2769    1.9681 -135.1835
 -1.9548    3.1710  442.4478
 -0.0021    0.0032    1.4843
```
P_inversa (iP_2) con get_proy(uv, xy);
```
  0.1865    1.3260  -91.0776
 -1.3170    2.1364  298.0917
 -0.0014    0.0021    1.0000
```

Las matrices obtenidas son distintas, pero representan la misma transformación.

iP_2 se puede transformar en iP multiplicándose por una constante

iP = 1.4843*iP_2

El efecto de multiplicar iP_2 por esta constante es nulo. Dadas unas coordenadas (x,y) de partida, y la matriz proyectiva iP_2=[a b c; d e f; g h 1], las coordenadas (u,v) que resultan de la transformación se obtienen de la siguiente forma.

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\longrightarrow \begin{cases} u = \dfrac{U}{W} \\ v = \dfrac{V}{W} \end{cases}$$

(u,v) con iP_2

u=U/W
v=V/W

(u,v) con iP=1.48*iP_2

u=(1.48*U1)/(1.48*W) = U1/W
v=(1.48*V1)/(1.48*W) = V1/W

Podemos observar que el efecto de multiplicar una matriz proyectiva por una constante sigue generando la misma transformación, ya que esta constante se cancela. Esta es la propiedad que permite que dos matrices proyectivas generen la misma transformación.

Adjuntad código de convierte()

```
function uv=convierte(xy,P)
 size_xy = size(xy);
 xy = [xy; ones(1, size_xy(2))];
 mult = P*xy;
 uv = mult(1:2,:)./mult(3, :);
end
```

Deformación de imagen usando la transformación dada por una matriz P

Adjuntad código de warp_img().

```
function im2=warp_img(im,iP)
      size_im = size(im);
      im2 = im;
      X = zeros(size_im(1), size_im(2));
      Y = zeros(size_im(1), size_im(2));
      uv = zeros(2, size_im(2));
      uv(1, :)= 1:size_im(2);

      for k=1:size_im(1)
          uv(2, :)= k;
          xy = convierte(uv, iP);
          X(k, :)=xy(1,:);
          Y(k, :)=xy(2,:);

      end

      for i=1:size_im(3)
          im2(:,:,i)=interp2(im(:,:,i),X,Y,'bilinear');
      end
end
```

Adjuntad la imagen resultante.

¿Qué valor devuelve MATLAB en esos puntos?
En esos puntos el Matlab devuelve [NaN, NaN, NaN]

Adjuntad captura de demoP4 donde se muestre la imagen original y la deformada.

Imagen original

Imagen deformada

Volcad también las coordenadas de los vértices de ambos cuadriláteros para vuestro resultado y la matriz P correspondiente a la deformación usada (dump_mat).
Cuadrilátero imagen original (xy)

X1=(160.187, 197.851)
X2=(446.279, 194.618)
X3=(600,900)
X4=(1,900)

Cuadrilátero imagen deformada (uv)

Y1=(1.785,168.757)
Y2=(600, 171.989)
Y3=(488.304, 900)
Y4=(119.778, 900)

Matriz P

6.8179	3.3973	-1758.5712
0.2035	13.2482	-2111.5564
0.0002	0.0110	1.0000

Transformaciones recursivas

Adjuntad la matriz P de la transformación a usar.

P =	0.0974	-0.0108	580.7560
	-0.1084	0.3456	264.6910
	-0.0003	-0.0000	1.0000

Adjuntad la imagen resultante de aplicar la transformación.

Adjuntad una captura de la imagen final resultante.

Indicad cómo podríais repetir el proceso para insertar copias adicionales de la foto dentro del nuevo cartel SIN TENER QUE VOLVER A MEDIR más coordenadas.

Para ello, se podría crear una función recursiva f que reciba un número n (el número de copias deseadas), una matriz inversa proyectiva iP, y una imagen im como parámetro. Si el valor de n es mayor o igual a 1

1. Filtrar la imagen correspondiente usando filtro gaussiano
2. Deformar la imagen im llamando a warp_img(im, iP), dando lugar a una imagen im_2
3. Llamar de forma recursiva a la función f(n-1,iP, im_2), y guardar este valor en im_3
4. Sustituir los valores NaN de im_3 por los correspondientes de im.

```
function im_res = img_recursiva(n,im,iP)
    if n==0
        im_res=im;
        return;
    end
    S = 1;
    L = round(2*S);
    filtro = fspecial('gauss', 2*L+1, S);
    im_2 = imfilter(im, filtro, 'sym');
    im_2 = warp_img(im_2,iP);
    im_rec=img_recursiva(n-1, im_2, iP);
    idxMat=isnan(im_rec);
    im_rec(idxMat)=im(idxMat);
    im_res=im_rec;
end
```

Adjuntad código usado en este apartado.

```
im = im2double(imread('billboard.jpg'));
figure()
imshow(im); % para medir las coordenadas
xy = [1 1500 1500 1;1 1 840 840];
uv = [583 1195 1195 580;266 174 649 556];
P = get_proy(xy, uv);

S = 1;
L = round(2*S);
filtro = fspecial('gauss', 2*L+1, S);

im_s = imfilter(im, filtro, 'sym');

im2 = warp_img(im_s, inv(P));

im2(isnan(im2)) = im(isnan(im2));
figure();
imshow(im2);
```

2) Ejemplo de transformación no lineal (20%)

Adjuntad código de vuestra función.

```
function P=get_nolineal(xy,uv)

  H = [ones(4, 1) xy' (xy(1,:).*xy(2,:))'];
  coefs = H\uv(1,:)';
  coefs2 = H\uv(2,:)';
  P = [coefs' ; coefs2'];

end
```

Usando dump_mat() volcar la matriz P con los coeficientes de la transformación directa (de xy a uv).

```
 P =  83.9027   0.2436  -0.5489   0.0025
     -65.1918   0.7290   1.8063  -0.0030
```

Adjuntad volcado de matriz de coeficientes de la transformación inversa (uv → xy)

```
iP =  904.8393  -1.4605  -2.0292   0.0064
     -846.2042   2.3800   2.7031  -0.0057
```

Adjuntad código de vuestra función convierte() modificada.

```
function uv=convierte(xy,P)
    size_xy = size(xy);
    if(numel(P)==9)
        xy = [xy; ones(1, size_xy(2))];
        mult = P*xy;
        uv = mult(1:2, :)./mult(3, :);
    else
        xy = [ones(1, size_xy(2)); xy; (xy(1,:).*xy(2,:))];
        uv = P*xy;
    end

end
```

Adjuntad matriz de las diferencias entre uv y las coordenadas obtenidas.

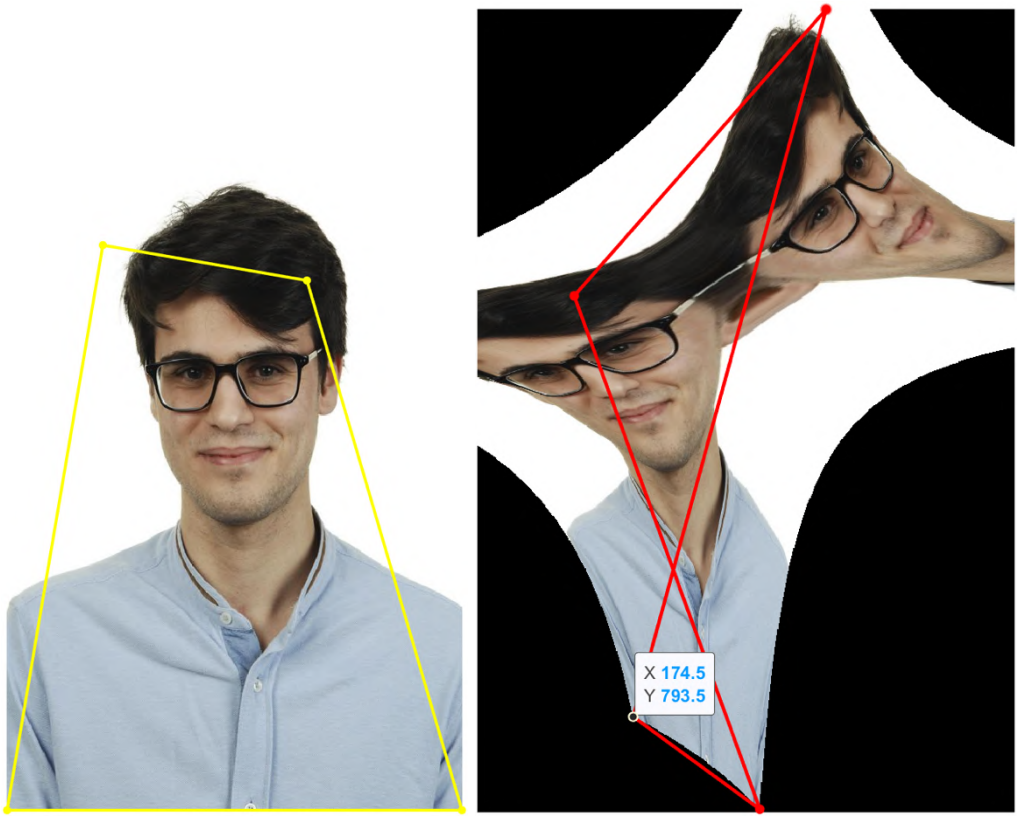
0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000

Deformación de una imagen con esta transformación no lineal

Adjuntad la imagen resultante.

¿Qué imagen obtendríamos si usamos la transformación P directa (de xy a uv) al llamar a warp_img? Adjuntad captura del nuevo resultado.

Adjuntad captura de demoP4 con la imagen original y la deformada. Volcad como en el caso lineal las coordenadas de los vértices de ambos cuadriláteros y la matriz P de coeficientes de la transformación usada.



Coordenadas xy
X1=(127.1,153.9)
X2=(395.9,200.5)
X3=(600,900)
X4=(1,900)

Coordenadas uv
Y1=(389.9,1)
Y2=(108.6,321.9)
Y3=(315.9,897.1)
Y4=(174.5,793.5)

P = 600.8730 -1.3586 -0.4740 0.0018
-324.9542 1.2430 1.2425 -0.0012

3) Seam Carving (50 %)

Indicad sus dimensiones y su relación de aspecto.
Dimensiones: 648x1152
Relación de aspecto: 1152/648 = 1.7778

Adjuntad código de vuestra función.

```
function E=energia(im)
    soporte = [7 7];
    S = 1.5;
    filtro = fspecial('gauss', soporte, S);

    im2 = imfilter(im, filtro, 'sym');
    det = abs(im - im2);
    detalle = det(:,:,1)+det(:,:,2)+det(:,:,3);
    E = 10*log2(1+detalle);

End
```

Valor de E en el píxel (10,10) y el valor medio (mean2) de E.
Valor de E en el píxel (10, 10) = 0.1372
Valor medio = 0.5287

Visualizar el resultado de E con **`imagesc();`** usando la paleta de colores predefinida "jet" con **`colormap("jet")`**. Poned una barra vertical al lado de la imagen mostrando la relación entre lvalores de E y los colores usados. Adjuntad la imagen resultante.

Eliminación de píxeles con menor energía

¿Cuál debería ser ahora su ancho para tener una relación de aspecto 3:2? ¿Número n de columnas a eliminar de la foto original?
Su ancho debería ser 972. n = 180 columnas a eliminar.

Al terminar calcular la media de la energía de la imagen final.
Media = 0.8160

Adjuntad código usado y la imagen resultante.

```
im = im2double(imread('img.jpg'));
size_im = size(im);
altura = size_im(1);
ancho_actual = size_im(2);

ancho_meta = altura * 3/2;
n = ancho_actual - ancho_meta;
im_actual = im;

for i=1:n
    E = energia(im_actual);
    [~, m] = min(E);
    m = mod(m, ancho_actual) + 1;
    new_im = zeros(altura, ancho_actual-1, 3);
    for j=1:altura
        new_im(j, 1:m(j)-1,:) = im_actual(j,1:m(j)-1,:);
        new_im(j,m(j):ancho_actual-1,:) = im_actual(j,m(j)+1:ancho_actual,:);
    end
    ancho_actual = ancho_actual-1;
    im_actual = new_im;
end
```

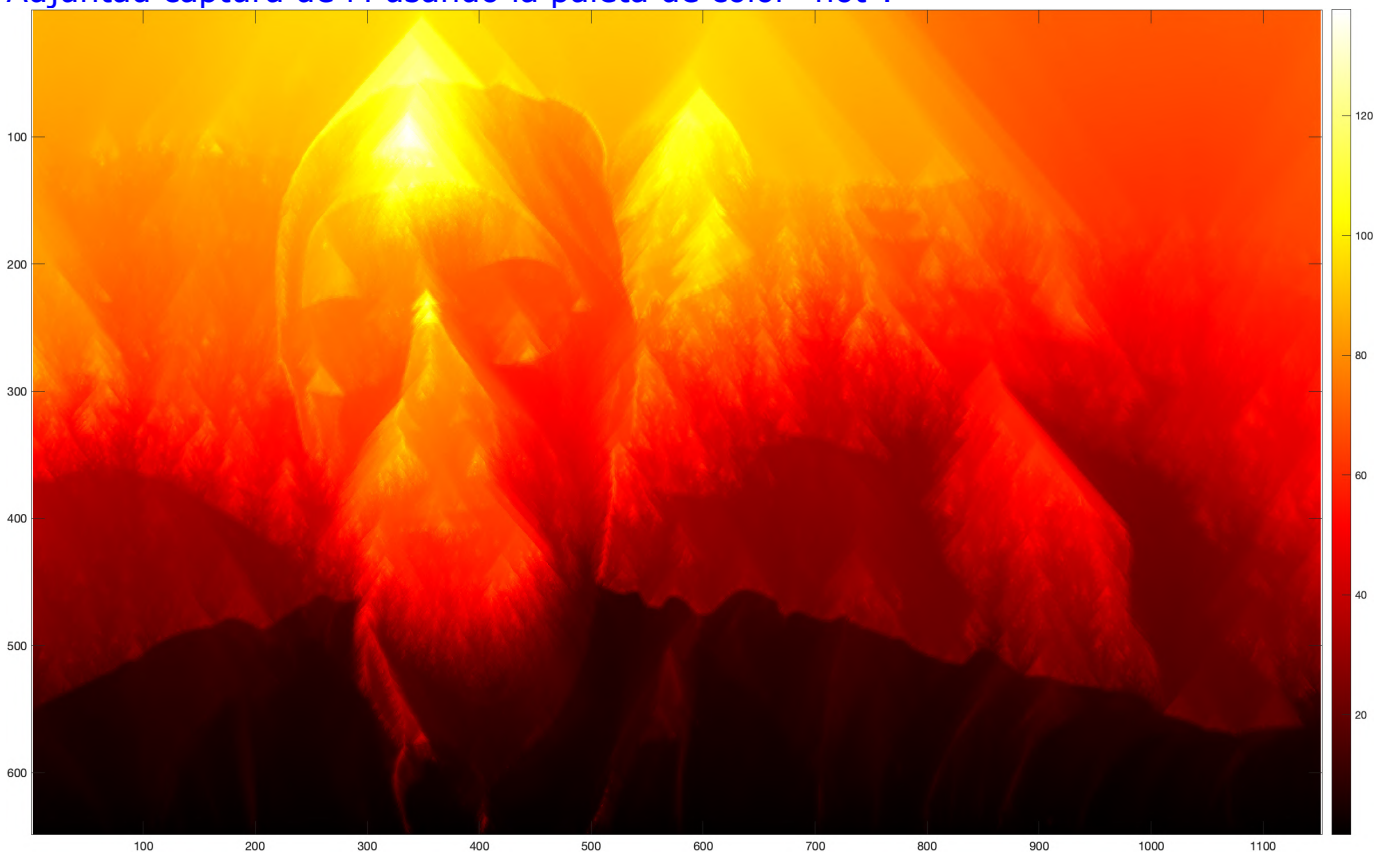
Eliminación de costuras verticales

Adjuntad código de vuestra función y los 3x4 valores de M en la esquina superior derecha (las 4 últimas columnas de las 3 primeras filas de M).

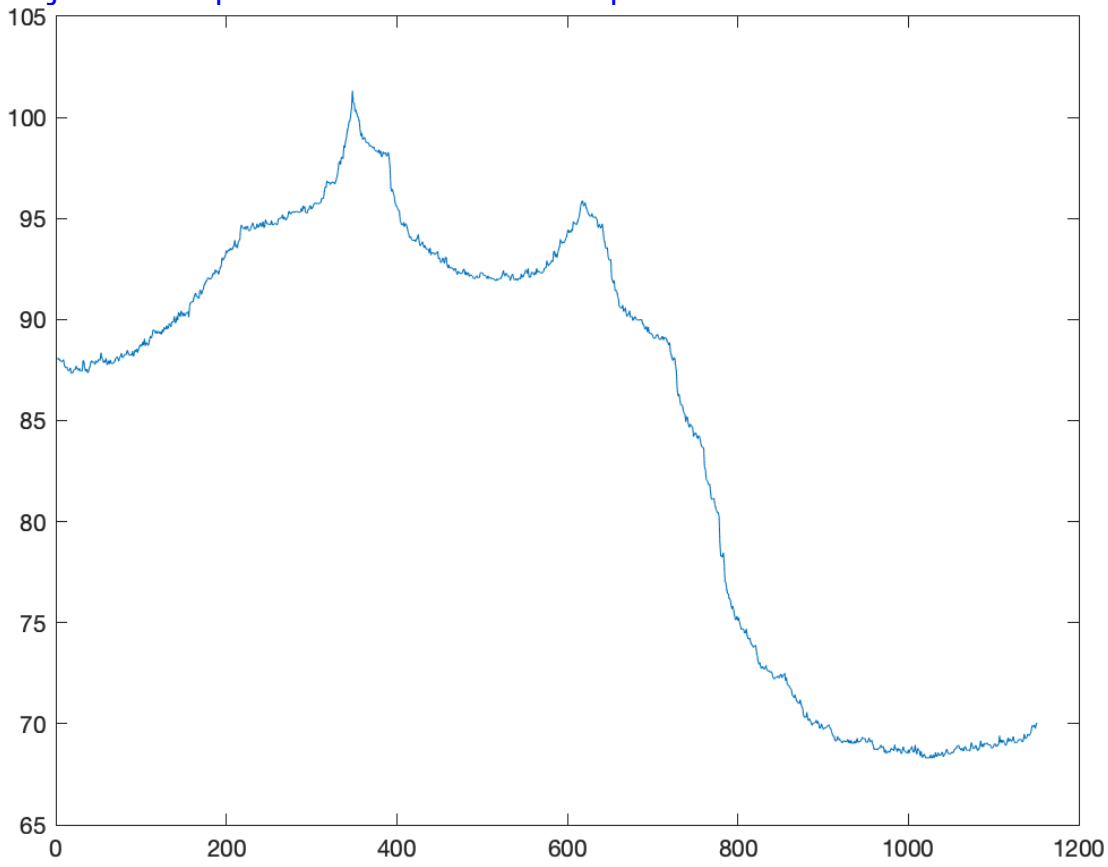
```
function M=calcula_M(E)
     M=Inf(size(E));
     [alto, ancho] = size(E);
     for j=2:ancho-1
        M(end,j) = E(end,j);
     end
     for i=alto-1:-1:1
        for j=2:ancho-1
            M(i,j) = E(i,j) + min(M(i+1,j-1:j+1));
        end
     end
end
```

69.9467	69.7636	69.9588	70.0499	Inf
69.7974	69.7609	69.9192	69.7876	Inf
69.7787	69.7625	69.7442	69.7537	Inf

Adjuntad captura de M usando la paleta de color “hot”.



Adjuntad un plot de los valores de la primera fila de la matriz M.



¿Valor y posición de dicho mínimo?
valor: 68.2987, posición: 1028

Adjuntad código de la función y mostrar la costura mínima encontrada superpuesta (en verde) sobre la imagen original. Volcad las coordenadas de los 3 últimos puntos de la costura.

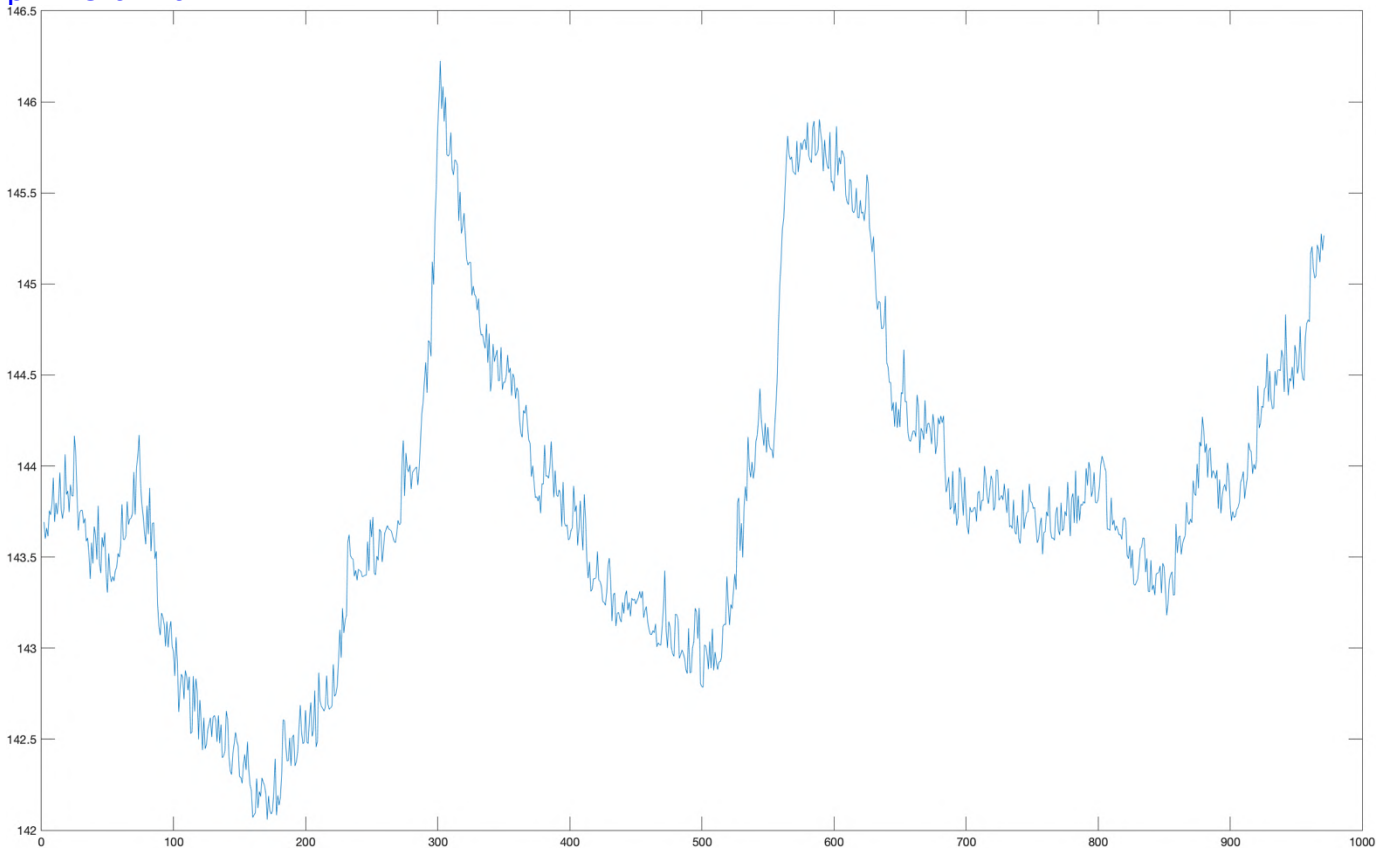
```
function s=find_seam(M)
    [alto,~] = size(M);
    s = zeros(alto, 1);

    [~, s(1)]=min(M(1,:));

    for k=2:alto
        [~,s(k)] = min(M(k, (s(k-1)-1):(s(k-1)+1)));
        s(k)=s(k)+s(k-1)-2;
    end
end
```

Coords: (646;954)
(647;953)
(648;953)

Al terminar el proceso re-calcular la M de la imagen final y adjuntad el plot de su primera fila.



Adjuntad el código usado para eliminar columnas en la imagen.

```
im = im2double(imread('img.jpg'));
im_actual = im;

size_im = size(im);
altura = size_im(1);
ancho_actual = size_im(2);

ancho_meta = altura * 3/2;
n = ancho_actual - ancho_meta;

for n=1:n
    E = energia(im_actual);
    M = calcula_M(E);
    s=find_seam(M);
    new_im = zeros(altura, ancho_actual-1, 3);
    for j=1:altura
        new_im(j, 1:s(j)-1,:) = im_actual(j,1:s(j)-1,:);
        new_im(j,s(j):ancho_actual-1,:) = im_actual(j,s(j)+1:ancho_actual,:);
    end
    ancho_actual = ancho_actual-1;
    im_actual = new_im;
end
```

Presentad una encima de la otra las siguientes imágenes (mismas dimensiones)

- La imagen original reescalada a las nuevas dimensiones usando imresize().
- La imagen original eliminando las primeras n columnas.
- La imagen que habéis obtenido con el algoritmo de seam-carving.

Adjuntad captura del resultado.

Calculad como antes la energía media (mean2) de la imagen final. ¿Es mayor o menor que la obtenida en la imagen final del método anterior? ¿Por qué?
Energia media: 0.6223. Es menor que la energía media obtenida en la imagen final anterior (0.8160) porque, con la condición de conectividad, no necesariamente se eliminan todos los píxeles con menos energía, sino el camino con menos energía. Eliminar todos los píxeles con la energía más baja hace que la energía sea lo más alta posible (en el método anterior).

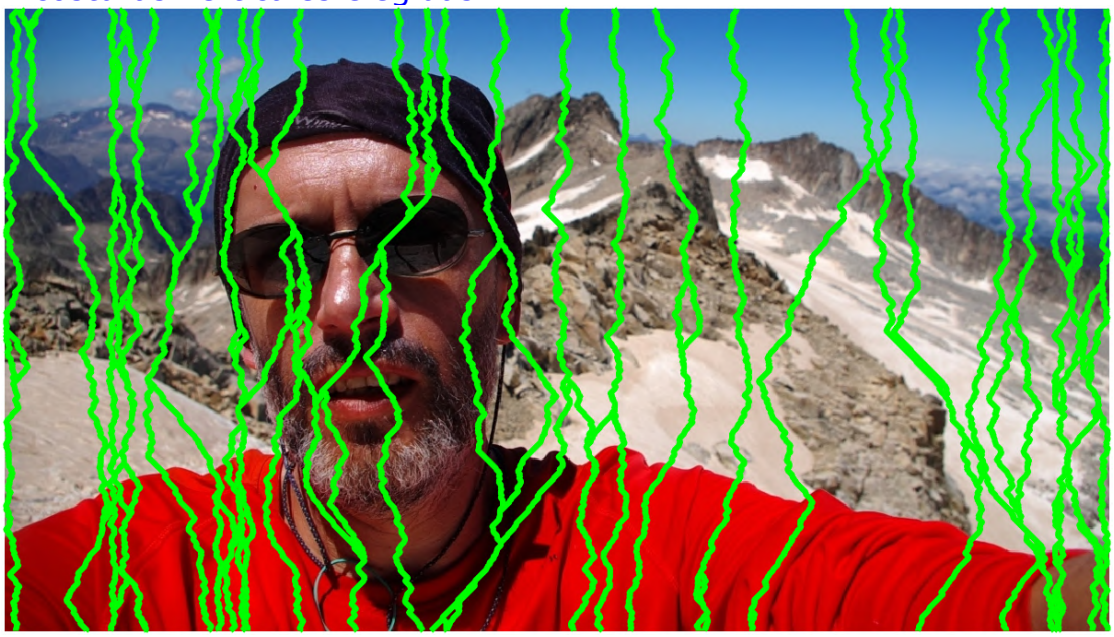
Inserción de costuras verticales

Imaginad que ahora queremos añadir columnas a la imagen original hasta conseguir una relación de aspecto de 2:1. ¿Cuántas columnas habría que añadir?

Habría que añadir 144 columnas

Adjuntad imagen resultante.

Mostrar la imagen original y superponer sobre ella gráficamente (en color verde) las n costuras verticales elegidas.



Adjuntad el código usado para ampliar la imagen.

```
im = im2double(imread('img.jpg'));

size_im = size(im);
altura = size_im(1);
ancho_actual = size_im(2);

ancho_meta = altura * 2;
n = ancho_meta-ancho_actual;

E = energia(im);
S = zeros(altura, n);

for k=1:n
    M = calcula_M(E);
    S(:,k)=find_seam(M);
    E(:, S(:,k)') = E(:, S(:,k)')*1.5;
end


im2 = zeros(altura, ancho_meta, 3);
for k=1:altura
    idx = sort([1:ancho_actual S(k, :)]);
    im2(k,:,:)=im(k, idx,:);
end
```

Presentad juntas (una encima de la otra) las tres imágenes siguientes (todas deben tener las mismas dimensiones):

- La imagen original reescalada a las nuevas dimensiones usando imresize().
- La imagen original añadiendo dos tiras negras de n/2 columnas a cada lado.
- La imagen obtenida con este método.

Adjuntad captura del resultado. Comentad las diferencias.

En la primera solución se "estira" la foto para obtener el tamaño deseado. Como podemos ver, la cara se ve más afectada que en la solución de seam carving, haciéndose más ancha (aunque se ven bastante similares). En la segunda foto, se colocan tiras negras a cada lado de la foto. Esto provoca que no se altere la foto en sí, pero desaprovechamos "pantalla". Observamos entonces que la tercera imagen (seam carving) es la opción que aprovecha todo el espacio requerido, deformando lo menos posible la imagen.

Partid de una foto de uno de vosotros con la misma relación de aspecto que la imagen original. Procurad que tenga un fondo con bastante detalle para ver las diferencias con respecto la foto del ejemplo. Procesadla como hemos visto para:

- Quitar columnas para quedaros con un formato 3:2.
- Añadir columnas para "estirarla" a un formato 2:1.

Adjuntad la foto original y las dos imágenes resultado, indicando las dimensiones de las tres imágenes.

648x1152

648x972

648x1296