Proyecto proyección 3D/2D (entrega por parejas)

Apellidos, Nombre: Ortiz Pasamontes, Enrique

Ejercicio 1:

Adjuntar código de vuestra rutina de proyección

```
function [ uv Zc ] = Tproj( X,datos3D )
global u0 v0 f

%traslacion
datos3Dcamara = [datos3D(1,:,:)-X(1);X(3)-datos3D(3,:,:);datos3D(2,:,:)-X(2)];
%rotacion
    %conversión a radianes
    az=X(4)* pi /180;
    elev=X(5)* pi /180;
    inc=X(6)* pi /180;
    %definicion de matrices
    matAzimut = [cos(az),0,-sin(az);0,1,0;sin(az),0,cos(az)];
    matElevacion = [1,0,0;0,cos(elev),sin(elev);0,-sin(elev),cos(elev)];
    matInclinacion = [cos(inc),sin(inc),0;-sin(inc),cos(inc),0;0,0,1];
    %rotacion
datos3Dcamara = matInclinacion*matElevacion*matAzimut*datos3Dcamara;

Xc = datos3Dcamara(1,:);
Yc = datos3Dcamara(2,:);
Zc = datos3Dcamara(3,:); %distacias de la camara a los puntos.

%Obtencion de coordenadas normalizadas
xx = Xc./Zc;
yy = Yc./Zc;

%Paso a pixeles
 u = (xx*f) + u0;
 v = (yy*f) + v0;

%posiciones de los puntos en la foto
uv = [u;v];

end
```

Ejercicio 2:

¿Cuántos puntos de los 30 serían visibles en la foto?
Son visibles 11 de los 30 puntos.
Adjuntar vuestro código y la imagen resultante.

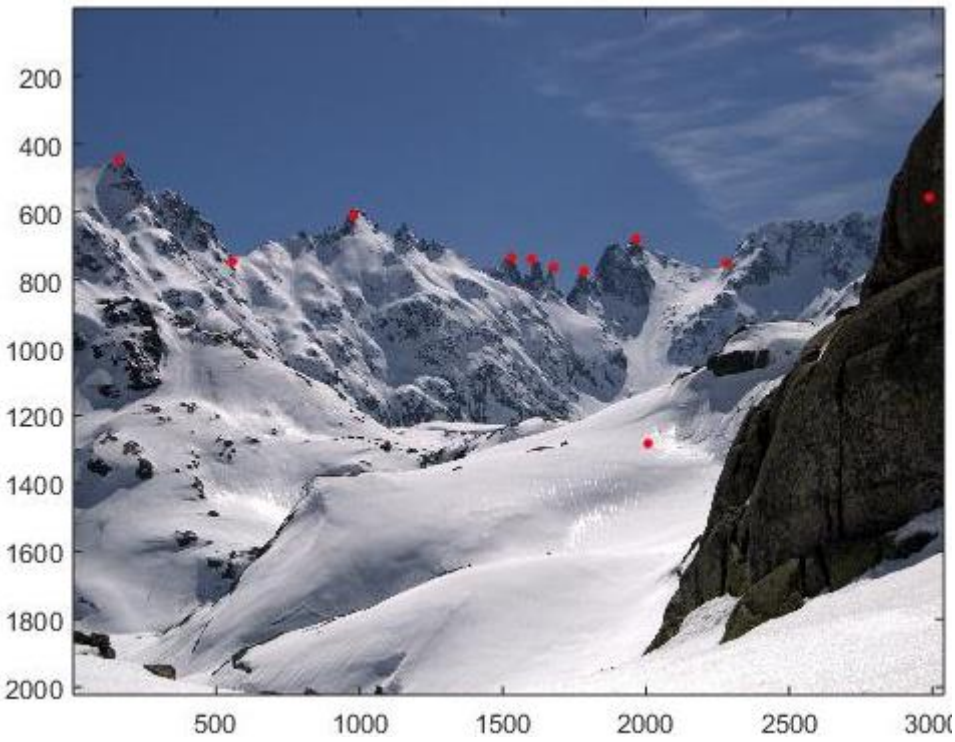
```
clear all;
close all;
global u0 v0 f;

im=imread('IMGP3047.jpg');
% im=imread('IMGP3029.jpg');
Xsize = size(im,2);
Ysize = size(im,1);

u0 = Xsize/2;
v0 = Ysize/2;
% f = 129*10;
f = 129 * 30;
X = [306607;4459599;1766;190.9;11.5;3.8];
% X = [306533;4459599;1766;190.9;11.5;3.8];
load datos_gredos.mat;

[uv,Zc]=Tproj(X,datos3D);
Points = [uv;Zc];
visibles = (Points(3,:) > 0) & (Points(1,:) > 0) & (Points(1,:) < Xsize) &
(Points(2,:) > 0) & (Points(2,:) < Ysize);
nombres = nombres(visibles)
Points = Points(:,visibles);
Points(3,:)

image(im)
% fc_truesize
hold on
plot(Points(1,:),Points(2,:),'ro','MarkerFaceColor','r','MarkerSize',3);
```



¿Hay algún punto que debería verse pero está tapado por el terreno?

El primer pico desde la derecha no es directamente visible, al igual que el refugio situado en la parte inferior de la fotografía.

Con la información disponible ¿podríamos detectar esa situación?

Sí que podríamos conocer la situación de estos puntos, ya que disponemos de la posición de la cámara en el mundo, sus parámetros extrínsecos y la posición del punto respecto a la cámara.

¿Cómo podríais calcular las distancias (en metros) desde la cámara a los puntos que habéis superpuesto sobre la foto?

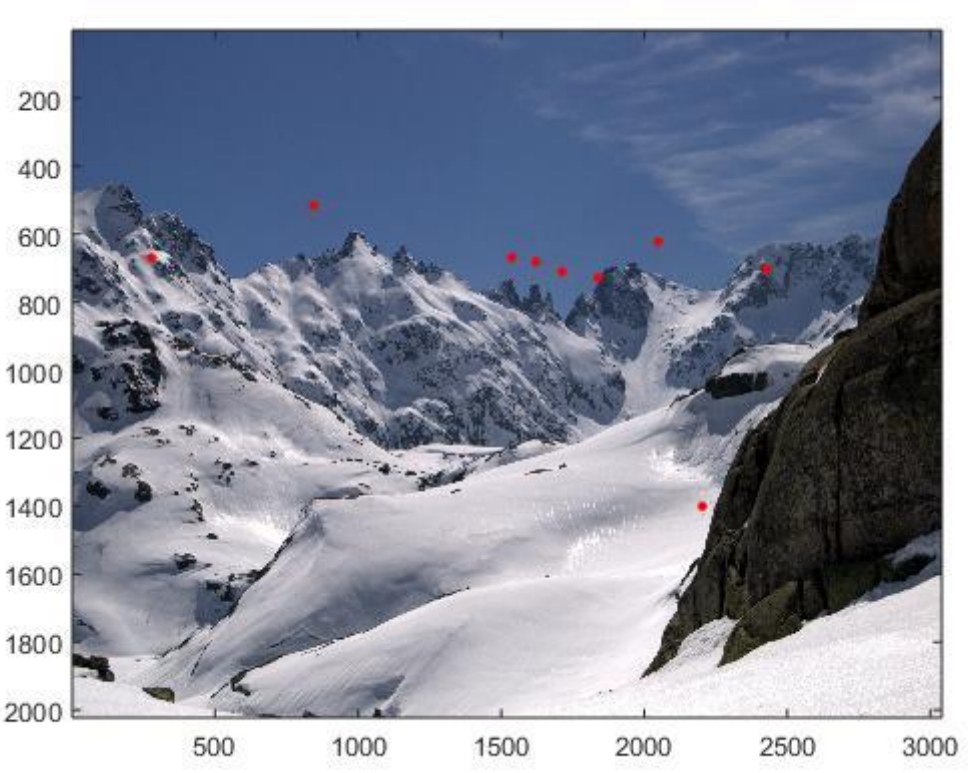
Gracias a la función Tproj que hemos hecho en el ejercicio anterior obtenemos en Zc la distancia de cada punto respecto a la cámara en metros.

Hacer un volcado de los puntos visibles dando su nombre y la distancia a la cámara.

Risco Hoyuelas	Port. Hoyuelas	Cuchillar Hermanitos	Hermanito 3	Hermanito 2	Hermanito 1	Perro	Casquerazo	Port. Machos	Cuchillar Navajas	Ref. Elola
1626	1805	2028	2207	2228	2242	2332	2538	2503	2516	1384

Usar como nueva posición (equivocada) de cámara: E0=306533 N0=4459214 h0=1846

Adjuntar la imagen resultante (observar como los puntos ya no caen donde deben).



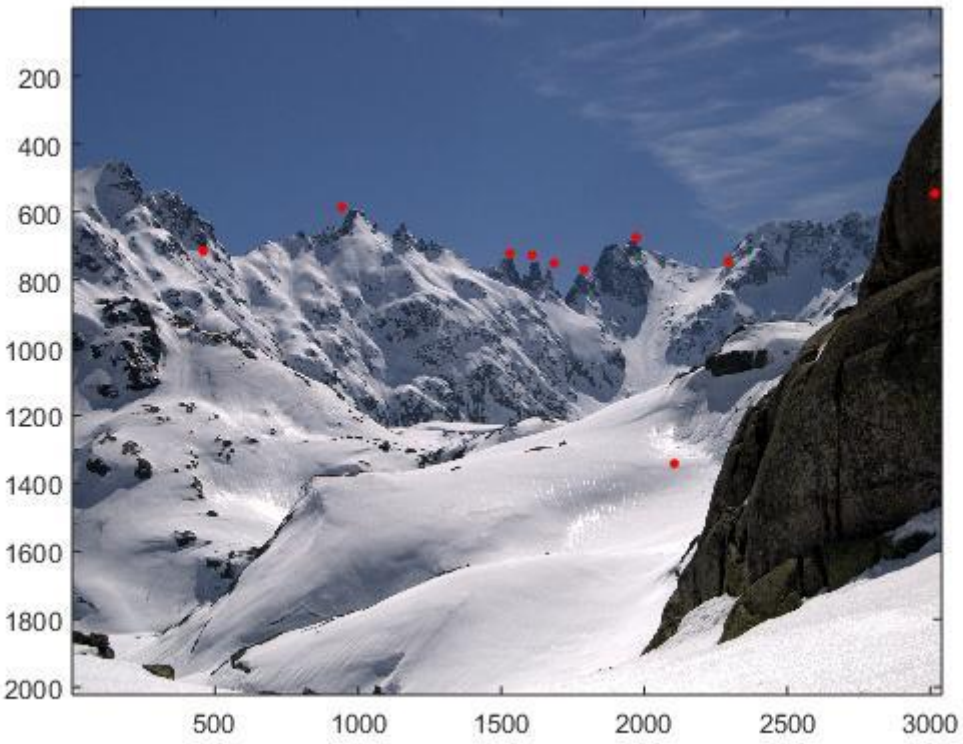
¿En qué sentido creéis que debéis alterar la focal f? ¿Por qué?

Adjuntar la imagen final y el valor de f usado.

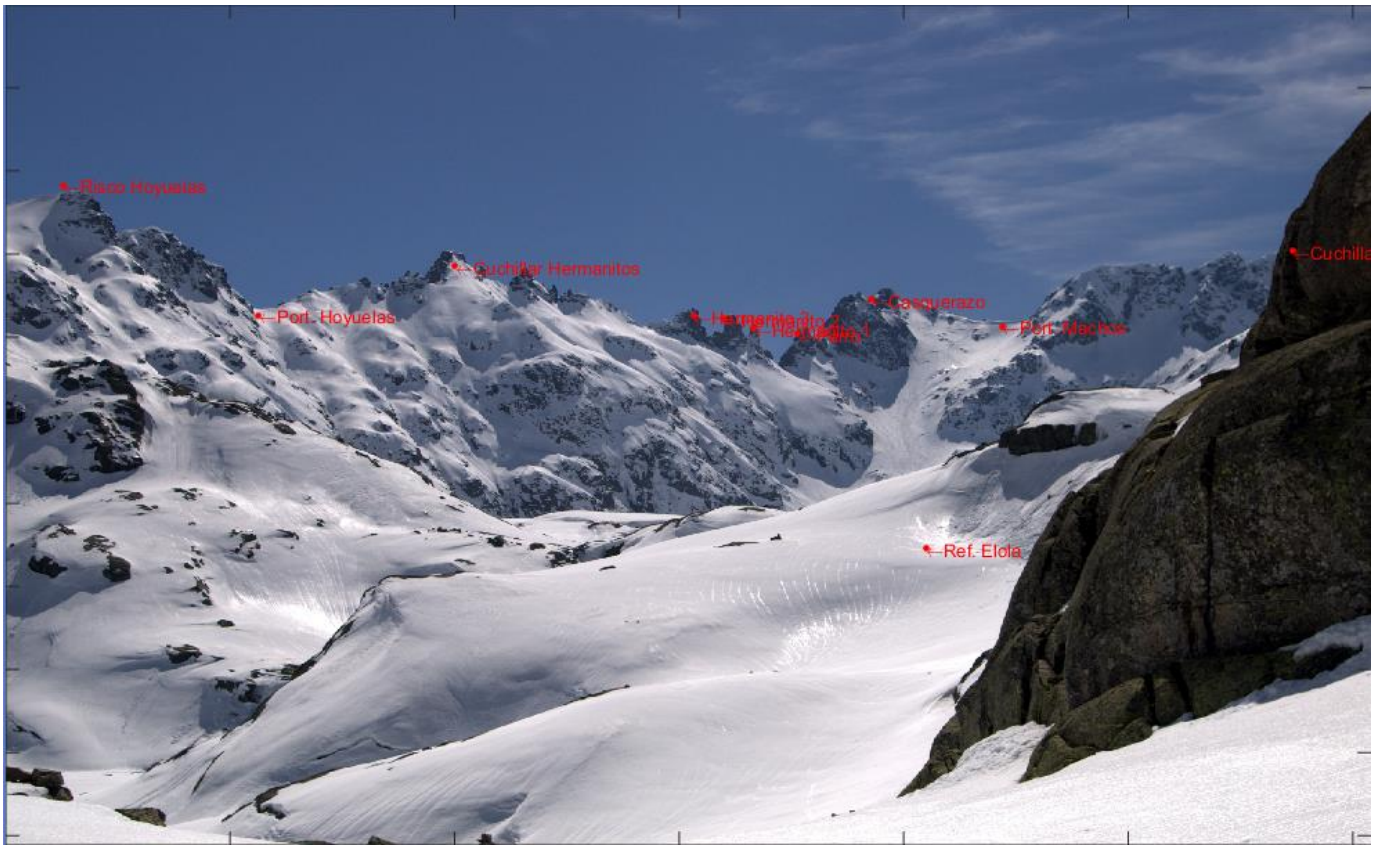
Debemos de disminuir la focal utilizada para que el ángulo de visión de la cámara sea mayor. De esta manera la imagen se reduce (mayor cantidad de cosas se proyectan en el área del sensor) y los puntos, que aparecen más distanciados entre sí que antes volverán a coincidir, e incluso podremos volver a ver el punto situado

más a la derecha que ahora dese la nueva posición no entra en el campo de visión de la cámara.

Si usamos una focal de 30mm nos acercamos bastante a la solución. Obtendríamos la fotografía más parecida posible a la tomada desde el primer lugar. Aunque para ver en la posición correcta el punto Risco Hoyuelas necesitaremos reducir la focal a unos 27 mm para que más o menos concuerde en su punto.



Adjuntar la imagen final etiquetada. Para no solapar la etiqueta con el punto que ya teníais escribir la etiqueta 15 o 20 píxeles a la derecha de la posición del punto. Añadiendo text(…,'Color','r') a la orden podéis indicar el color a usar en etiquetas.



Ejercicio 3:

Volcar vuestra matriz de datos 2D (2x4) con los píxeles (u,v) de los puntos de control (redondeados como enteros). Para asegurar que no habéis metido la pata superponer los valores anteriores sobre la imagen con un plot usando círculos rojos como hicimos en el ejercicio anterior. Adjuntar imagen resultante.

Adjuntar código de vuestra función discrepancias.m

```
function dif = discrepancias( X,datos3D,datos2D )
[ uv Zc ] = Tproj( X,datos3D );
dif=datos2D-uv;
dif = [dif(1,:),dif(2,:)]';
end
```

Dar la estimación inicial para las coordenadas E,N,h y ángulos de giro de la cámara obtenida de acuerdo a las indicaciones anteriores.

Empleando el siguiente código obtenemos la siguiente posición:

```
mediaDatos3D=sum(datos3D,2)/size(datos3D,2);
X = [mediaDatos3D(1);mediaDatos3D(2)+4000;mediaDatos3D(3);180;0;0]
```

```
X =

   1.0e+06 *

    0.3058
    4.4623
    0.0024
    0.0002
         0
         0
```

Adjuntar vuestro código de la iteración.

```
% Limpieza
clear all;
close all;
global u0 v0 f;

% Carga de datos
im=imread('IMGP3029.jpg');
Xsize = size(im,2);
Ysize = size(im,1);
u0 = Xsize/2;
v0 = Ysize/2;
load datos_gredos.mat;
datos3D = [datos3D(:,6),datos3D(:,10),datos3D(:,24),datos3D(:,26)];
datos2D = [1044,1206,2472,3002;727,667,343,374];
datosMedidos = datos2D;
f = 129 * 23;   %Focal de la cámara


% Hipótesis de posición inicial
mediaDatos3D=sum(datos3D,2)/size(datos3D,2);
X0 = [mediaDatos3D(1);mediaDatos3D(2)+4000;mediaDatos3D(3);180;0;0];

% Algoritmo de optimización
f_ptr=@(X)discrepancias(X,datos3D,datos2D);
dx = ones(2*size(datos3D,2),1);
while ~isempty(dx(dx > 0.1 | dx < -0.1))
    [X fval]=opt_gauss_newton(f_ptr,X0);
    fval; %mostramos erroes de esta iteración
    dx = X-X0;
    X0 = X;
end

% Dibujado
image(im)
fc_truesize
hold on
plot(datosMedidos(1,:),datos2D(2,:),'ro','MarkerFaceColor','r','MarkerSize',5);
% plot(datosMedidos(1,:),datos2D(2,:),'ro','MarkerFaceColor','b','MarkerSize',3);
[uv,Zc]=Tproj(X,datos3D);
Points = [uv;Zc];
plot(Points(1,:),Points(2,:),'bo','MarkerFaceColor','b','MarkerSize',3);
```

Volcar vuestra mejor hipótesis final de posición/pose y los resultados pedidos al final de la iteración.

```
X =

   1.0e+06 *

    0.3072
    4.4606
    0.0020
    0.0002
    0.0000
    0.0000
```

Adjuntar la imagen final y algún zoom que muestre gráficamente los errores en algunos de los puntos de control.

EXTRAS:

Podemos seguir la trayectoria de los puntos durante las iteraciones modificando un poco el algoritmo:

```
% Limpieza
clear all;
close all;
global u0 v0 f;

% Carga de datos
im=imread('IMGP3029.jpg');
Xsize = size(im,2);
Ysize = size(im,1);
u0 = Xsize/2;
v0 = Ysize/2;
load datos_gredos.mat;
datos3D = [datos3D(:,6),datos3D(:,10),datos3D(:,24),datos3D(:,26)];
datos2D = [1044,1206,2472,3002;727,667,343,374];
datosMedidos = datos2D;
f = 129 * 23;   %Focal de la cámara


% Hipótesis de posición inicial
mediaDatos3D=sum(datos3D,2)/size(datos3D,2);
X0 = [mediaDatos3D(1);mediaDatos3D(2)+4000;mediaDatos3D(3);180;0;0];

% Algoritmo de optimización
f_ptr=@(X)discrepancias(X,datos3D,datos2D);
dx = ones(2*size(datos3D,2),1);

% Dibujado
image(im)
fc_truesize
hold on
plot(datosMedidos(1,:),datosMedidos(2,:),'ro','MarkerFaceColor','r','MarkerSize',5);
[uv,Zc]=Tproj(X0,datos3D);  %puntos iniciales
LastPoints = [uv;Zc];
plot(LastPoints(1,:),LastPoints(2,:),'bo','MarkerFaceColor','b','MarkerSize',3);

while ~isempty(dx(dx > 0.1 | dx < -0.1))
    [X fval]=opt_gauss_newton(f_ptr,X0);
    fval; %mostramos erroes de esta iteración
    dx = X-X0;
    X0 = X;

    %Dibujado
    [uv,Zc]=Tproj(X,datos3D);
    Points = [uv;Zc];
    plot(Points(1,:),Points(2,:),'bo','MarkerFaceColor','b','MarkerSize',3);
    for i=1:size(Points,1)+1
        plot([LastPoints(1,i),Points(1,i)],[LastPoints(2,i),Points(2,i)],'b');
    end
    LastPoints=Points;
end

% Mostrar datos por terminal
X % Posición y pose
fval % Discrepancias
mediaErrorAbs = sum(abs(fval))/size(fval,1)     % Media valor absoluto de los errores
```

Obtenemos como resultado esto: