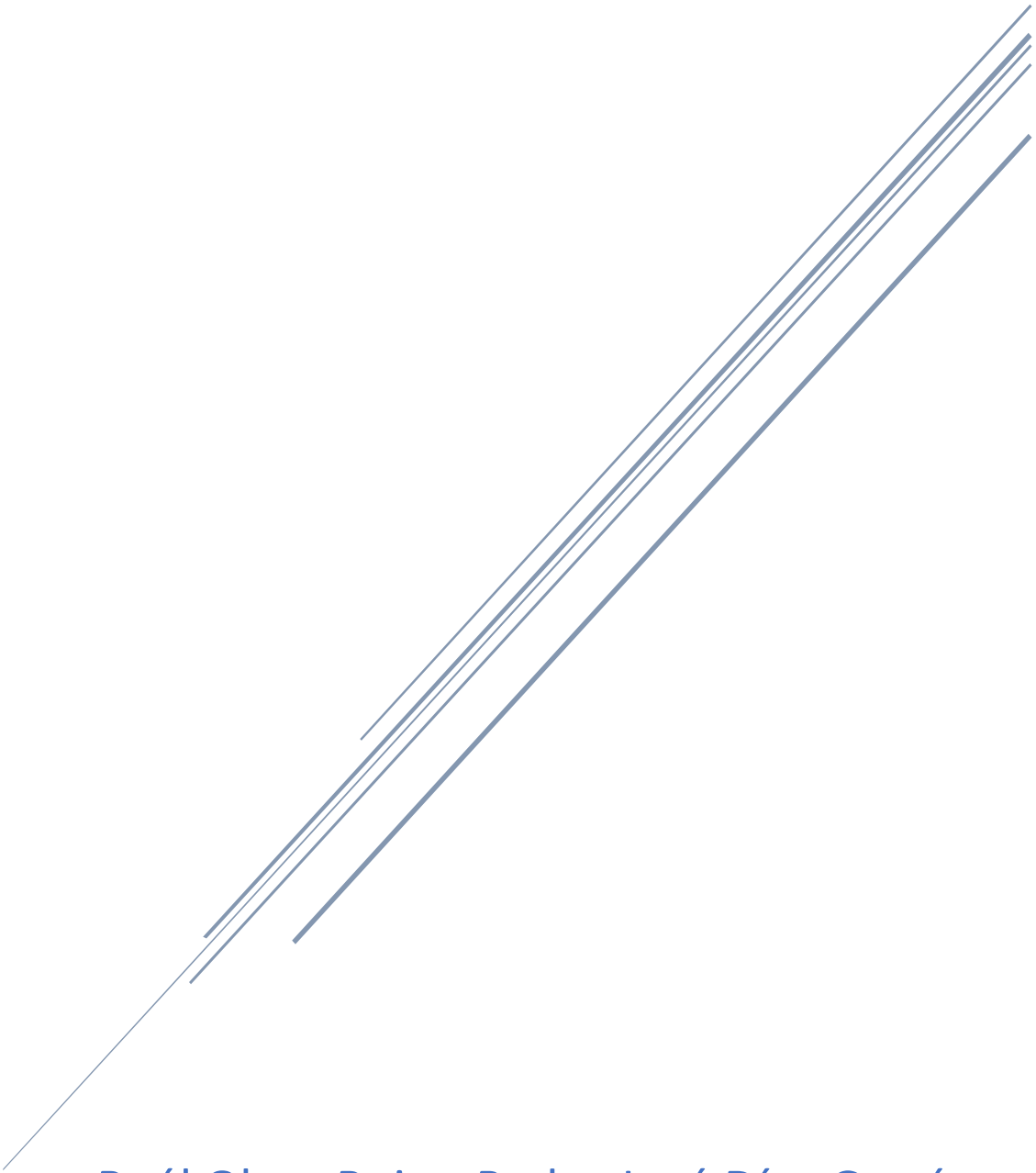
Ejemplo sencillo de realidad aumentada



Raúl Olmo Ruiz y Pedro José Díaz García

Sistemas de Percepción

Índice

1. Introducción

El objetivo de este trabajo es realizar la calibración de una cámara, y a partir de ella utilizar un objeto base de referencia sobre la que se superpondrá un objeto virtual. Dicho objeto deberá estar colocado en una posición y orientación fija sobre la base de referencia independientemente de la perspectiva desde la que haya sido tomada la fotografía del objeto de referencia.

2. Calibración de la cámara

El proceso de calibración de la cámara se realiza para obtener los parámetros intrínsecos de la cámara. Este proceso solo es necesario realizarlo una sola vez, ya que dichos parámetros son invariantes ante la posición y orientación desde la que se haya tomado cada fotografía.

El Toolbox de calibración de cámaras facilitado por el profesor de la asignatura está diseñado de forma que, si se utiliza un tablero de ajedrez como plantilla, dicha calibración se realiza de forma automática.

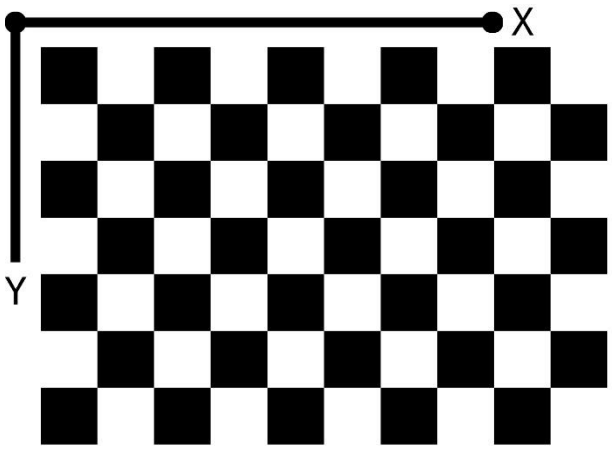


Figura 1.Plantilla de calibración

Así, se han tomado varias fotos desde diferentes perspectivas para realizar de forma más exacta la calibración. Dichas fotografías deben estar en formato JPG y deben introducirse dentro del directorio en el que se encuentre dicho Toolbox.

En primer lugar, será necesario iniciar el Toolbox en modo Standard (imágenes almacenadas en memoria) y mediante "*Image Names*" se cargarán las imágenes que se desean introducir en el Toolbox. Se tiene que introducir el nombre base de todas las imágenes sin números e indicar el formato de dichas imágenes (en nuestro caso JPG).

Una vez que el Toolbox tiene cargadas las imágenes en memoria, se ejecuta la función *"Calibration"* que dará como resultado los parámetros intrínsecos de nuestra cámara. Los resultados obtenidos son los siguientes:

$$Distancia\ focal\ (en\ píxeles): f_c = \begin{bmatrix} 1473.0 \\ 1490.3 \end{bmatrix}$$

$$Punto\ principal\ (en\ píxeles): c_c = \begin{bmatrix} 685.6 \\ 547.8 \end{bmatrix}$$

$$Coeficientes\ Distorsión\ radial: k_r = \begin{bmatrix} -0.139 \\ 0.502 \end{bmatrix}$$

$$Coeficientes\ Distorsión\ tangencial: k_t = \begin{bmatrix} -0.017 \\ -0.013 \end{bmatrix}$$

$$Skew: s = 0$$

$$Error\ de\ reproyección\ (en\ píxeles): err = \begin{bmatrix} 6.63 \\ 6.67 \end{bmatrix}$$

Una vez que se han obtenido dichos parámetros con la función *"Calibration"* será necesario guardarlos en un archivo denominado *Calib_Results.mat* para utilizarlos posteriormente para obtener los parámetros extrínsecos.

3. Objeto de referencia

Como se comentó en la introducción, el objeto virtual debe representarse sobre un objeto de referencia. Dicho objeto de referencia se trata de una plantilla formada por 4 puntos no colineales, esto ayudará posteriormente en la localización de dichos puntos para realizar el tratamiento de imágenes. Además, la plantilla elegida es asimétrica y con un punto de mayor tamaño que será usado como origen de dicho objeto.

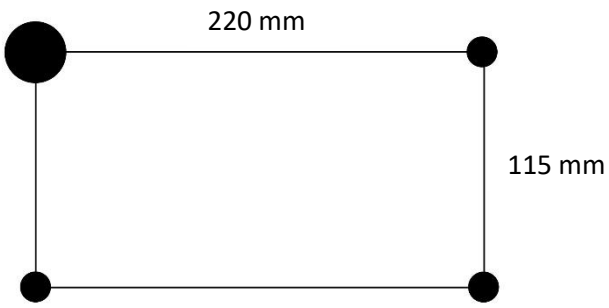


Figura 2. Plantilla de objeto de referencia

4. Obtención de parámetros extrínsecos

Una vez que han sido calculados los parámetros intrínsecos de nuestra cámara, se procede al cálculo de los parámetros extrínsecos que son dependientes de la perspectiva con la que ha sido tomada la fotografía. Para ello, en primer lugar, se determina la posición de cada uno de los puntos característicos del objeto de referencia y, posteriormente, se utilizará la función *compute_extrinsic* para determinar las matrices de parámetros extrínsecos.

4.1. Puntos característicos

Primero, se realiza un tratamiento de la imagen con el objetivo de aislar los puntos característicos del resto de la imagen. Para ello, se ha realizado el siguiente procedimiento:

1. Binarización con un determinado valor umbral.
2. Aplicación de una plantilla de corrección de bordes. Esto ha sido necesario porque las imágenes tomadas presentaban zonas oscuras en los bordes de la imagen. Dado que solo nos interesa trabajar con la plantilla de referencia, que se encuentra situada en el centro de la imagen, se pueden despreciar dichos bordes sin interferir con el objeto de referencia.
3. Aplicación de un procedimiento morfológico tipo apertura, para eliminar las líneas de referencia de la plantilla y quedarnos solo con los círculos.
4. Obtención mediante etiquetado de los centroides y las dimensiones del Bounding Box de las regiones.
5. Determinación del origen y de los ejes de referencia en función de los datos obtenidos anteriormente.
    a. El punto de origen se corresponde con el centroide de aquella región cuyo Bounding Box tenga mayores dimensiones. En un principio se utilizó el área de la región para determinar dicho punto; pero en imágenes como "*rectangulo8.jpg*", en las que la perspectiva se encuentra bastante inclinada y el punto de mayor tamaño está al fondo, este procedimiento daba lugar a errores en la determinación.

b. En base al origen determinado, se puede conocer el punto correspondiente al eje Y como aquel punto más cercano al punto de origen.
c. El punto correspondiente al eje X se determina de forma que el producto escalar entre el vector que forma dicho punto con el origen y el vector correspondiente al eje Y sea mínimo. Esto es así, ya que se considera que, para magnitudes de vectores semejantes, a menor producto escalar mayor perpendicularidad. Primeramente, se determinó el eje X como aquel punto que era el segundo más alejado del origen, pero en imágenes como *"rectangulo7.jpg"*, este procedimiento resultaba fallido.

4.2. Matrices de parámetros extrínsecos

Para obtener estas matrices se usa la función *compute_extrinsic* que recibirá como entrada los parámetros intrínsecos determinados en la calibración, además de dos matrices mp y MP.

Estas matrices contienen por columnas respectivamente las coordenadas en píxeles de los puntos característicos del objeto de referencia y las coordenadas en dimensiones reales de estos puntos expresados en los ejes de referencia. En el caso de la matriz MP, ha sido necesario invertir las coordenadas X e Y debido a que los ejes considerados por el Toolbox para la calibración son contrarios a unos ejes como los que aparecen en la figura 1, que son los que se han considerado para definir las referencias.

Una vez definidas dichas matrices, se ejecuta la función *compute_extrinsic* que nos permite conocer las matrices de parámetros extrínsecos, formada por la matriz de rotación y el vector de traslación.

5. Proyección del objeto virtual

Conocidos los parámetros intrínsecos y extrínsecos se puede proceder a proyectos los puntos que forman nuestro objeto virtual sobre la imagen del objeto de referencia. Como objeto virtual, en nuestro caso se ha elegido un monigote y una casa. Además, se dispone de dos casos diferentes, uno en el que se tratará la imagen con distorsión y otro caso sin distorsión.

En primer lugar, se definen las coordenadas de ciertos puntos de los objetos a representar respecto a los ejes de referencia. También, se ha construido la matriz de parámetros intrínsecos a partir de los resultados obtenidos del proceso de calibración. Se explicará de forma detallada, el procedimiento seguido para el caso sin distorsión y el caso con distorsión:

- **Sin distorsión:**
  1. Expresión de las coordenadas de los puntos del objeto virtual como coordenadas homogéneas.
  2. Aplicación de las matrices de parámetros intrínsecos y extrínsecos, obteniendo así las coordenadas homogéneas de las proyecciones en la imagen.
  3. Deshomogeneización para obtener la proyección final sobre la imagen.

- **Con distorsión:**
  1. Expresión de las coordenadas de los puntos del objeto virtual como coordenadas homogéneas.
  2. Aplicación de la matriz de rotación y traslación.

3. Normalización de las coordenadas obtenidas para aplicar posteriormente la distorsión.
4. Aplicación de la distorsión con los factores de distorsión radial y tangencial obtenidos en la calibración.
5. Desnormalización y expresión de las coordenadas reales en píxeles para su representación en la imagen.

Una vez se tienen las proyecciones, se grafican los puntos obtenidos sobre la imagen de referencia.

6. Ejecución y resultados

Para realizar la ejecución de la secuencia diseñada para comprobar el correcto funcionamiento del proyecto debe ejecutarse el archivo *Ejecucion_realidad_virtual.m*. Este archivo pedirá al usuario que decida si quiere tener en cuenta la distorsión o no, y cargará los datos obtenidos de la distorsión. Por último, para cada una de las imágenes que forman parte de la secuencia se ejecuta la función *Realidad_virtual*, que se encarga de realizar los cálculos descritos anteriormente y de representar los resultados de la ejecución.

El código completo se encuentra a final del documento. A continuación, se incluyen las imágenes que forman parte de la secuencia; aunque se encuentra preparado para mostrar otras imágenes adicionales descomentando solo algunas líneas.

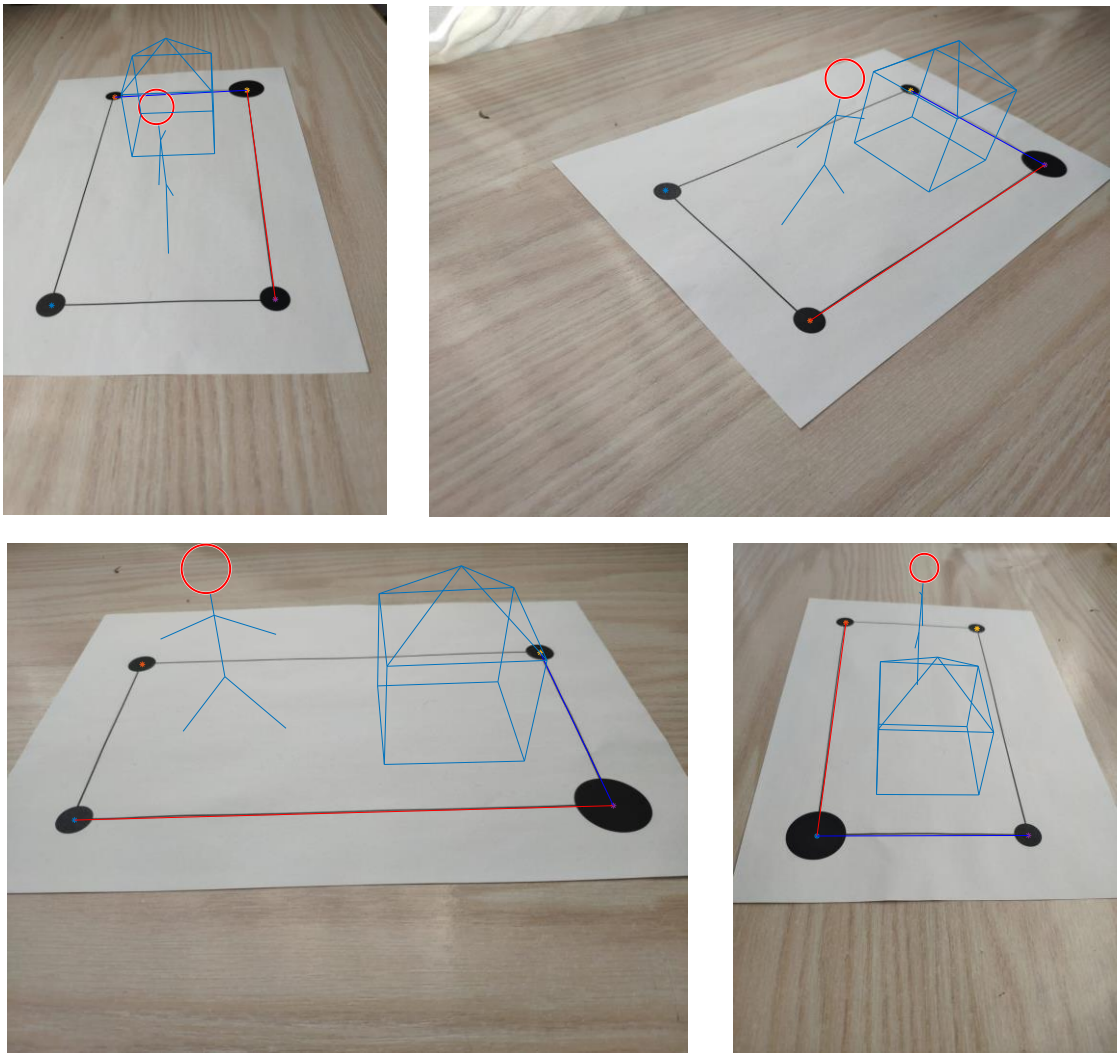


Figura 3. Secuencia de imágenes

Anexo. Código del proyecto

Ejecucion_realidad_virtual.m

```
clear all; clc; close all;
distorsion=input('Introduzca si quiere tratar sin distorsión(0) o con
distorsión (1):');
load('Calib_Results.mat');

%Para ver secuencia
imagen=imread('rectangulo6.jpg'); figure(1);
Realidad_virtual(imagen,distorsion,fc,cc,kc,alpha_c);
imagen=imread('rectangulo7.jpg'); figure(2);
Realidad_virtual(imagen,distorsion,fc,cc,kc,alpha_c);
imagen=imread('rectangulo9.jpg'); figure(3);
Realidad_virtual(imagen,distorsion,fc,cc,kc,alpha_c);
imagen=imread('rectangulo8.jpg'); figure(4);
Realidad_virtual(imagen,distorsion,fc,cc,kc,alpha_c);

%Descomentar para ver resto de imágenes
% imagen=imread('rectangulo5.jpg'); figure(5);
% Realidad_virtual(imagen,distorsion,fc,cc,kc,alpha_c);
% imagen=imread('rectangulo6.jpg'); figure(6);
% Realidad_virtual(imagen,distorsion,fc,cc,kc,alpha_c);
% imagen=imread('rectangulo7.jpg'); figure(7);
% Realidad_virtual(imagen,distorsion,fc,cc,kc,alpha_c);
% imagen=imread('rectangulo8.jpg'); figure(8);
% Realidad_virtual(imagen,distorsion,fc,cc,kc,alpha_c);
% imagen=imread('rectangulo9.jpg'); figure(9);
% Realidad_virtual(imagen,distorsion,fc,cc,kc,alpha_c);
```

Realidad_virtual.m

```
function Realidad_virtual(imagen,distorsion,fc,cc,kc,alpha_c)

f=imagen;

%DETECCIÓN DE PUNTOS DE REFERENCIA PARA OBTENER PARÁMETROS EXTRÍNSECOS

%Binarización
fgray=rgb2gray(f);
% figure(1); imshow(fgray);
fbin=fgray<60;
% figure(2); imshow(fbin);

%Eliminación de bordes mediante una plantilla de ceros y unos.
[M,N]=size(fbin);
plantilla_bordes=zeros(M,N);
w=70; %Anchura del borde a eliminar
for i=w:M-w
    for j=w:N-w
        plantilla_bordes(i,j)=1;
    end
end

fbin=fbin & plantilla_bordes;
% figure(3); imshow(fbin);


%Aplicación de apertura para eliminar líneas del objeto de referencia, ya que
solo interesan los círculos.
mask=strel('disk',4);
flimpia=imopen(fbin,mask);
% figure(4); imshow(flimpia);

%Determinación de regiones y datos característicos. Se usarán los centros
%para determinar la posición de los puntos de referencia, y el BBox para
%determinar cual de ellos es el origen.
[etiq,netiq]=bwlabel(flimpia);
datos=regionprops(etiq,'Centroid','BoundingBox');
centros=[];
LmaxBBox=[];
% figure(5);
imshow(f);
for i=1:netiq
    centros=[centros datos(i).Centroid'];
    LmaxBBox(i)=max(datos(i).BoundingBox(3:4)); %solo interesan las
dimensiones del BBox
    hold on;plot(datos(i).Centroid(1),datos(i).Centroid(2),'*');
end

%Determinación de ejes de referencia
    %El origen corresponderá a aquella región con una mayor dimensión del
    %BBox.
    [~,etiq_origen]=max(LmaxBBox);
    origen=centros(:,etiq_origen);

    %El eje Y lo determinará el punto más cercano al origen. Por otra
    %parte, el eje X lo determinará aquel punto que haga que el producto
```

```
    %escalar del vector que forma con el origen, por el vector del ejeY sea
    %mínimo.
    distancias=[];
    vectores=[];
    for i=1:size(centros,2)
        vectores(:,i)=centros(:,i)-origen;
        distancias(i)=norm(vectores(:,i));
    end
    [distancias, indices]=sort(distancias);

    %Eje X
    [~,ind]=min([abs(vectores(:,indices(2))'*vectores(:,indices(3)))
abs(vectores(:,indices(2))'*vectores(:,indices(4)))]);
    ejeX=centros(:,indices(ind+2));
    hold on; line([origen(1) ejeX(1)],[origen(2) ejeX(2)],'Color','r');

    %Eje Y
    ejeY=centros(:,indices(2));
    hold on; line([origen(1) ejeY(1)],[origen(2) ejeY(2)],'Color','b');


%Construcción de matrices MP y mp.

    %MP contiene, por columnas, las coordenadas de los puntos de
    %característicos del objeto de referencia expresados en los ejes del
objeto
    %de referencia. Se han intercambiado las coordenadas x e y porque los
    %ejes calculados están invertidos respecto a los ejes considerados por
    %la función compute_extrinsic()
    MP=[0 0 0;0 220 0;115 0 0;115 220 0]';

    %mp contiene, por columnas, las coordenadas en píxeles de de los puntos
    %característicos del objeto de referencia. La ordenación de estos
    %dentro de la matriz debe concordar con la ordenación en MP.
    mp=[origen centros(:,indices(ind+2)) centros(:,indices(2))
centros(:,indices(length(centros)-(ind-1)))];

%Uso de compute_extrinsic() para obtener los parámetros extrínsecos.
[~, cto, cRo, ~] = compute_extrinsic (mp, MP, fc, cc, kc, alpha_c);


%REPRESENTACIÓN DE LOS OBJETOS VIRTUALES
%Puntos para graficar monigote
Pmonigote=[ 57.5   57.5   57.5   57.5   57.5   57.5   57.5  57.5;
            135    185    160    160    160    160    135   185;
              0      0     30     60     70     80     50    50];

%Puntos para graficar casa
Pcasa=[ 30 30  0; 90 30 0 ; 90 90 0; 30 90 0;
        30 30 50; 90 30 50; 90 90 50;30 90 50;
        60 60 75]';

%Construcción de la matriz de parámetros intrísecos con los valores
%obtenidos en la calibración.
MIntrins=[fc(1) 0 cc(1);0 fc(2) cc(2);0 0 1];

%Cálculo de las coordenadas en píxeles de los puntos de los objetos
%virtuales
if (distorsion==0)
```

```
    %Sin distorsión
    p_monigote=MIntrins*[cRo cto]*[Pmonigote;ones(1,size(Pmonigote,2))];
    p_casa=MIntrins*[cRo cto]*[Pcasa;ones(1,size(Pcasa,2))];

    pmonigote=p_monigote(1:2,:)./p_monigote(3,:);
    pcasa=p_casa(1:2,:)./p_casa(3,:);
else

    %Con ditorsión
    p_monigote=[cRo cto]*[Pmonigote;ones(1,size(Pmonigote,2))];
    p_casa=[cRo cto]*[Pcasa;ones(1,size(Pcasa,2))];

    Pnmonigote=p_monigote(1:2,:)./p_monigote(3,:);
    Pncasa=p_casa(1:2,:)./p_casa(3,:);

    for i=1:size(Pnmonigote,2)
        r=norm(Pnmonigote(:,i));
        Pndmonigote(:,i)=Pnmonigote(:,i)*(1+kc(1)*r^2+kc(2)*r^4)+...

+[2*kc(3)*Pnmonigote(1,i)*Pnmonigote(2,i)+kc(4)*(r^2+2*Pnmonigote(1,i)^2);

2*kc(4)*Pnmonigote(1,i)*Pnmonigote(2,i)+kc(3)*(r^2+2*Pnmonigote(2,i)^2)];
    end
    for i=1:size(Pncasa,2)
        r=norm(Pncasa(:,i));
        Pndcasa(:,i)=Pncasa(:,i)*(1+kc(1)*r^2+kc(2)*r^4)+...
            +[2*kc(3)*Pncasa(1,i)*Pncasa(2,i)+kc(4)*(r^2+2*Pncasa(1,i)^2);
            2*kc(4)*Pncasa(1,i)*Pncasa(2,i)+kc(3)*(r^2+2*Pncasa(2,i)^2)];
    end

    pmonigote=[fc(1) 0;0 fc(2)]*Pndmonigote+[cc(1);cc(2)];
    pcasa=[fc(1) 0;0 fc(2)]*Pndcasa+[cc(1);cc(2)];

end

%Dibujo de los puntos resultantes sobre la imagen
hold on;
linea(pmonigote(:,1),pmonigote(:,3));
linea(pmonigote(:,2),pmonigote(:,3));
linea(pmonigote(:,3),pmonigote(:,4));
linea(pmonigote(:,4),pmonigote(:,7));
linea(pmonigote(:,4),pmonigote(:,8));
linea(pmonigote(:,4),pmonigote(:,5));
viscircles(pmonigote(:,6)',norm(pmonigote(:,5)-pmonigote(:,6)));

linea(pcasa(:,1),pcasa(:,2));
linea(pcasa(:,2),pcasa(:,3));
linea(pcasa(:,3),pcasa(:,4));
linea(pcasa(:,4),pcasa(:,1));
linea(pcasa(:,5),pcasa(:,6));
linea(pcasa(:,6),pcasa(:,7));
linea(pcasa(:,7),pcasa(:,8));
linea(pcasa(:,8),pcasa(:,5));
linea(pcasa(:,1),pcasa(:,5));
linea(pcasa(:,2),pcasa(:,6));
linea(pcasa(:,3),pcasa(:,7));
linea(pcasa(:,4),pcasa(:,8));
linea(pcasa(:,5),pcasa(:,9));
linea(pcasa(:,6),pcasa(:,9));
```

```
linea(pcasa(:,7),pcasa(:,9));
linea(pcasa(:,8),pcasa(:,9));


function linea(p1, p2)
    line([p1(1) p2(1)],[p1(2) p2(2)]);
end
end
```