




ELIMINACIÓN DE DISTORSIÓN EN IMAGEN

PEDRO JOSÉ DÍAZ GARCÍA

SISTEMAS DE PERCEPCIÓN, 4º GIERM
Ejercicio Práctico 2



El objetivo de este ejercicio es la realización de un programa en MATLAB capaz de corregir el efecto de distorsión de lente en una imagen con distorsión conocida; esto se hará utilizando dos métodos de interpolación diferentes.

El programa que se ha desarrollado está estructurado de la siguiente manera: en primer lugar, se le pide al usuario que seleccione qué imagen utilizar en función del tipo de distorsión que se quiere corregir, también se le pide que indique el método de interpolación a utilizar; tras cargar la imagen y los parámetros necesarios, se procede al cálculo de la imagen corregida, la cual finalmente será representada por pantalla junto a su correspondiente forma distorsionada.

El procedimiento seguido para calcular la imagen corregida consta de dos pasos. En el primero de ellos se calcula la correspondencia entre píxeles de ambas imágenes, es decir, para cada píxel de la imagen corregida se calculan las coordenadas del píxel resultado de aplicarle la distorsión; esta tarea debe realizarse en este sentido, para evitar que en la imagen corregida queden píxeles vacíos o sobrescritos. Una vez obtenida la correspondencia entre dos píxeles, también se tendrá la correspondencia entre sus intensidades. Este proceso viene implementado en las siguientes líneas de código:

```
xn=(u-u0)/fx;  
yn=(v-v0)/fy;  
  
xnd=xn*(1+kr1*(xn^2+yn^2));  
ynd=yn*(1+kr1*(xn^2+yn^2));  
  
ud=xnd*fx+u0;  
vd=ynd*fy+v0;
```

En definitiva, para cada píxel de la imagen corregida se obtienen sus coordenadas expresadas en m, se les aplica la distorsión correspondiente, y se devuelven las coordenadas en píxeles para su correspondiente distorsionado. Es importante tener en cuenta la posibilidad de que el resultado exceda las limitaciones de la imagen, de manera que, si esto ocurre, no existirá correspondencia real entre intensidades y se deberá asignar un valor de intensidad arbitrario:

```
if(vd<1 || vd>M || ud<1 || ud>N)  
  
    imagen_corregida(v,u)=127;
```

Como podrá comprobarse, el resultado de estos cálculos no tiene por qué ser entero, lo que nos lleva al segundo paso del procedimiento: aplicar una interpolación para aproximar dicho resultado a un entero, de manera que se corresponda con un píxel. En función del tipo de interpolación elegida por el usuario al comienzo de la ejecución del programa, se empleará el siguiente código:

- Aproximación al vecino más cercano.

La opción más simple para obtener coordenadas enteras a partir del resultado anterior es redondear dicho resultado, de manera que la intensidad que se le aplique al píxel de la imagen corregida será la intensidad del píxel de la imagen distorsionada obtenido mediante este redondeo.

```
vd=round(vd);  
ud=round(ud);  
  
imagen_corregida(v,u)=dimagen_distorsion(vd,ud);
```

- Interpolación bilineal.

En este caso, en lugar de utilizar un redondeo, se calculará la intensidad a asignar como una ponderación por distancia entre las coordenadas objetivo (resultado del primer paso del proceso) y sus cuatro píxeles más cercanos. Esto viene implementado en las siguientes líneas de código:

```
vd1=floor(vd);
vd2=ceil(vd);
ud1=floor(ud);
ud2=ceil(ud);

imagen_corregida(v,u)=(vd2-vd)*(ud2-ud)*dimagen_distorsion(vd1,ud1)+...
                      (vd2-vd)*(ud-ud1)*dimagen_distorsion(vd1,ud2)+...
                      (vd-vd1)*(ud2-ud)*dimagen_distorsion(vd2,ud1)+...
                      (vd-vd1)*(ud-ud1)*dimagen_distorsion(vd2,ud2);
```

A continuación, se presentan los resultados obtenidos para cada tipo de distorsión y cada tipo de interpolación.

Imagen distorsionada

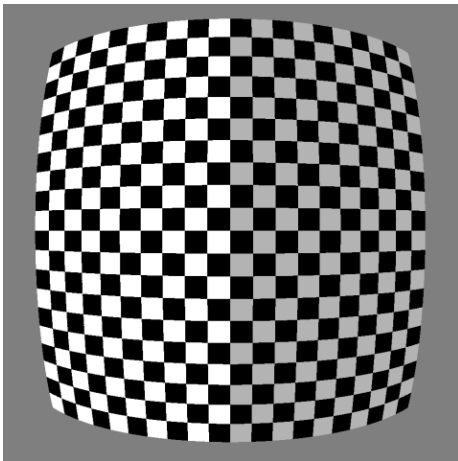


Imagen corregida

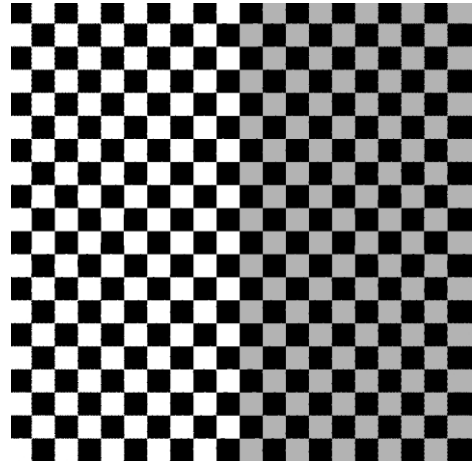


Ilustración 1. Corrección de distorsión tipo barrel con aproximación al vecino más próximo.

Imagen distorsionada

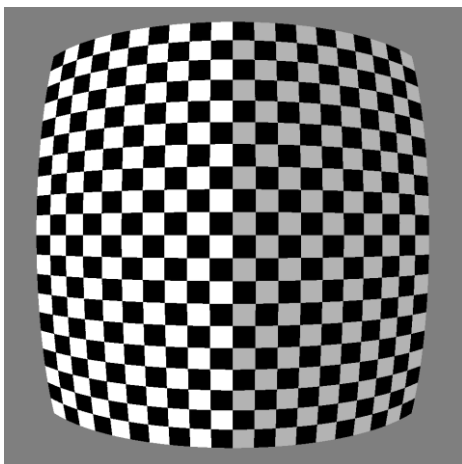


Imagen corregida

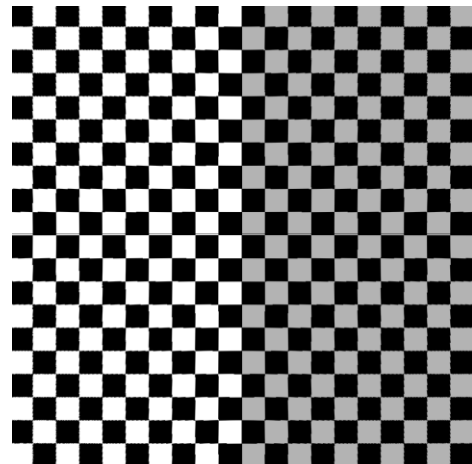


Ilustración 2. Corrección de distorsión tipo barrel con interpolación bilineal.

Imagen distorsionada

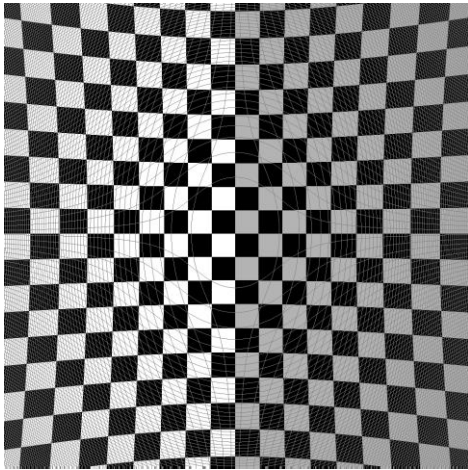


Imagen corregida

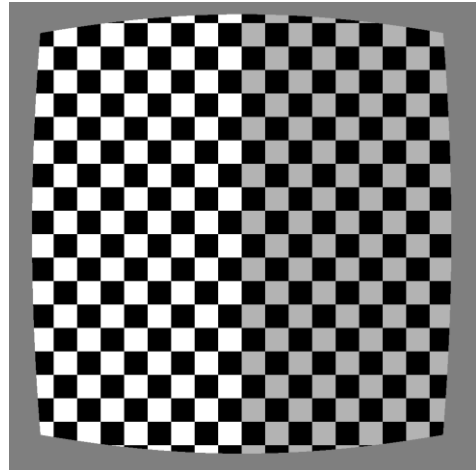


Ilustración 3. Corrección de distorsión tipo pincushion con aproximación al vecino más próximo.

Imagen distorsionada

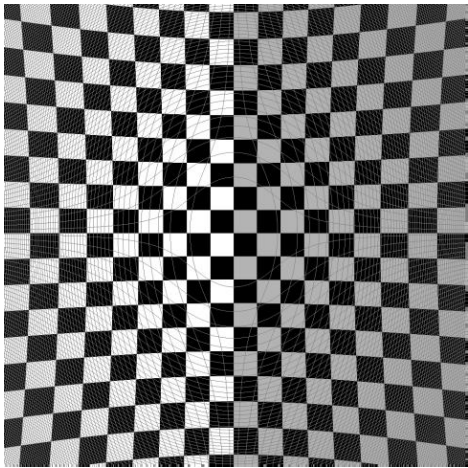


Imagen corregida

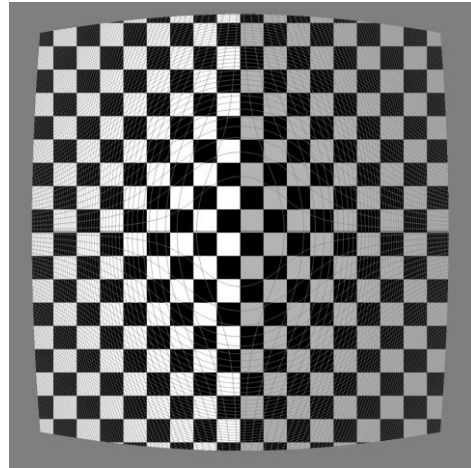


Ilustración 4. Corrección de distorsión tipo pincushion con interpolación bilineal.

Código completo

```
%Trabajo 2. Eliminación de distorsión en imagen. (Por Pedro José Díaz García)
clear all; close all;

%Al ejecutar el programa, se le pide al usuario que decida qué tipo de
%imagen y qué tipo de interpolación utilizar. En caso de que el usuario
%seleccione un valor no válido, se mostrará un mensaje de error y se
%volverá a ejecutar el programa.
tipo_imagen=input('Seleccione la imagen a usar: 1=barrel 2=pincushion \n');
tipo_interpolacion=input('Seleccione el tipo de interpolación que utilizará: 0=vecino
más próximo 1=bilineal \n');

%%DEFINICIÓN DE DATOS Y PARÁMETROS

%Se carga la imagen seleccionada por el usuario, junto a su correspondiente
%factor de distorsión. Se muestra el mensaje de error en caso de que sea
%necesario.
if (tipo_imagen==1)
    imagen_distorsion=imread('chessBoardDistorted1.jpg');
    kr1=-0.4320; %Coeficiente de distorsión radial
elseif (tipo_imagen==2)
    imagen_distorsion=imread('chessBoardDistorted2.jpg');
    kr1=0.4320; %Coeficiente de distorsión radial
else
    disp('ERROR: No se ha seleccionado una imagen disponible'); %Mensaje de error.
    run('Codigo_trabajo_practico_2');
end

f=0.0042; %Distancia focal (m)
N=1000; M=1000; %Resolución de la imagen (N:ancho, M:alto) (pix)
w=0.00496; h=0.00352; %Tamaño del sensor (w:ancho, h:alto) (m)
u0=N/2+1; v0=M/2-2; %Punto principal de la imagen (pix)
fx=f*N/w; fy=f*M/h; %Longitudes focales efectivas

%%CORRECCIÓN DE LA DISTORSIÓN
imagen_corregida=[];
dimagen_distorsion=double(imagen_distorsion); %Para operar, pasar la imagen a tratar a
double.

%Se recorre cada píxel de la imagen corregida...
for u=1:N
    for v=1:M
        %Cálculo del píxel distorsionado correspondiente al píxel de la
        %imagen corregida.
        xn=(u-u0)/fx;
        yn=(v-v0)/fy;

        xnd=xn*(1+kr1*(xn^2+yn^2));
        ynd=yn*(1+kr1*(xn^2+yn^2));

        ud=xnd*fx+u0;
        vd=ynd*fy+v0;

        %Saturación en caso de que el cálculo del píxel distorsionado resulte fuera de
        los límites de la imagen.
        %En este caso, se asigna un valor de intensidad arbitrario.
        if(vd<1 || vd>M || ud<1 || ud>N)
            imagen_corregida(v,u)=127;
        else
```

%Aplicación del tipo de interpolación elegida por el usuario. En caso de que el usuario seleccione un valor no válido, se mostrará un mensaje de error y se volverá a ejecutar el programa.

```
%%INTERPOLACIÓN BILINEAL
if(tipo_interpolacion==1)
    %Cálculo de las coordenadas enteras de los píxeles más
    %próximos al píxel distorsionado calculado.
    vd1=floor(vd);
    vd2=ceil(vd);
    ud1=floor(ud);
    ud2=ceil(ud);

    %Se determina la intensidad del píxel de la imagen
    %corregida como una ponderación por proximidad de las
    %intensidades de los cuatro píxeles más cercanos al píxel distorsionado
calculado.
    imagen_corregida(v,u)=(vd2-vd)*(ud2-ud)*dimagen_distorsion(vd1,ud1)+...
        (vd2-vd)*(ud-ud1)*dimagen_distorsion(vd1,ud2)+...
        (vd-vd1)*(ud2-ud)*dimagen_distorsion(vd2,ud1)+...
        (vd-vd1)*(ud-ud1)*dimagen_distorsion(vd2,ud2);

%%INTERPOLACIÓN POR EL VECINO MÁS PRÓXIMO
elseif (tipo_interpolacion==0)
    %Cálculo de las coordenadas enteras del píxel más
    %próximo al píxel distorsionado calculado.
    vd=round(vd);
    ud=round(ud);

    %La intensidad de este último píxel calculado será la
    %intensidad del píxel correspondiente en la imagen
    %corregida.
    imagen_corregida(v,u)=dimagen_distorsion(vd,ud);

else
    disp('ERROR: No se ha seleccionado un tipo de interpolación correcto');
%Mensaje de error.
    run('Codigo_trabajo_practico_2');
end
end
end
end

imagen_corregida=uint8(imagen_corregida); %Para mostrar la imagen, pasarla a uint8.

%Representación conjunta de la imagen distorsionada y su corrección.
figure(1);
subplot(1,2,1);imshow(imagen_distorsion);title('Imagen distorsionada');
subplot(1,2,2);imshow(imagen_corregida);title('Imagen corregida');
```