Sistemas de percepción

Ejercicio práctico 1: Modelo de formación de la imagen

Álvaro Calvo Matos

Damián Jesús Pérez Morales

Índice

[1. Introducción 2](#_Toc21282472)

[2. Ejercicio 1 3](#_Toc21282473)

[3. Ejercicio 2 6](#_Toc21282474)

[4. Comentarios acerca de la práctica 10](#_Toc21282475)

# Introducción

En el primer ejercicio práctico, se pretende trabajar con los conocimientos adquiridos en el primer bloque de la asignatura. En este caso, representando en el sensor de la cámara un objeto plano (tablero de ajedrez), según la posición y orientación de la cámara, considerando también la existencia de distorsión en la cámara o sin ella.

Para realizar este ejercicio se ha empleado MATLAB, programando un script para el caso de distorsión y otro diferente para el caso de sin distorsión. Estos scripts hacen en resumen un movimiento de la posición y orientación de la cámara enfocando al tablero y se representa por pantalla al ejecutar el script lo que se vería en el sensor de la cámara en todo momento, haciendo una animación del movimiento.

# Planteamiento del problema

En este ejercicio, se tiene un tablero de ajedrez plano contenido en el plano XY de dimensiones arbitrarias y separación entre puntos arbitrarias, que se pueden cambiar en todo momento al inicio del script, modificando:

* ‘Naj’: Número de puntos horizontales
* ‘Maj’: Número de puntos verticales
* ‘separacion’: separación entre puntos (se tiene la misma separación entre los puntos horizontales y verticales)



Ilustración 2. Tablero inicial en MATLAB

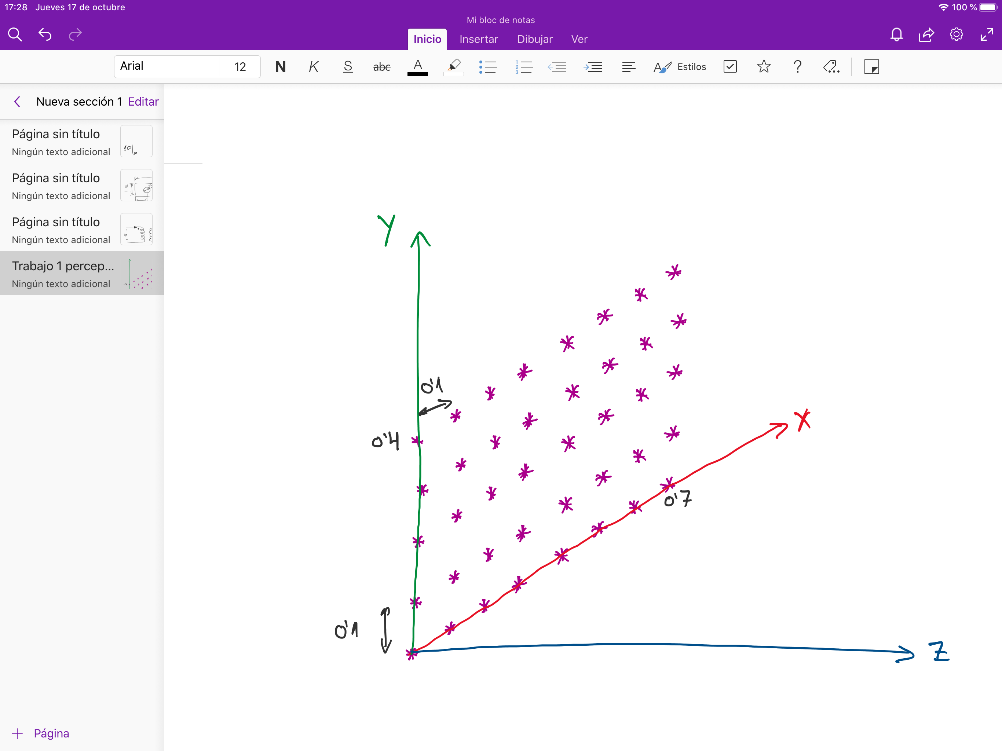


Ilustración . Tablero inicial manuscrito

Además del tablero, se tiene una cámara con los parámetros dados en el enunciado del ejercicio, donde es necesario definir una posición y una orientación inicial que se puede ajustar también con las siguientes variables:

* ‘x’, ‘y’, ‘z’: Para modificar la posición en x, y, z inicial de la cámara, respectivamente
* ‘alpha’, ‘beta’, ‘gamma’: Ángulo que se desea girar en ‘x’, ‘y’, ‘z’, respectivamente.

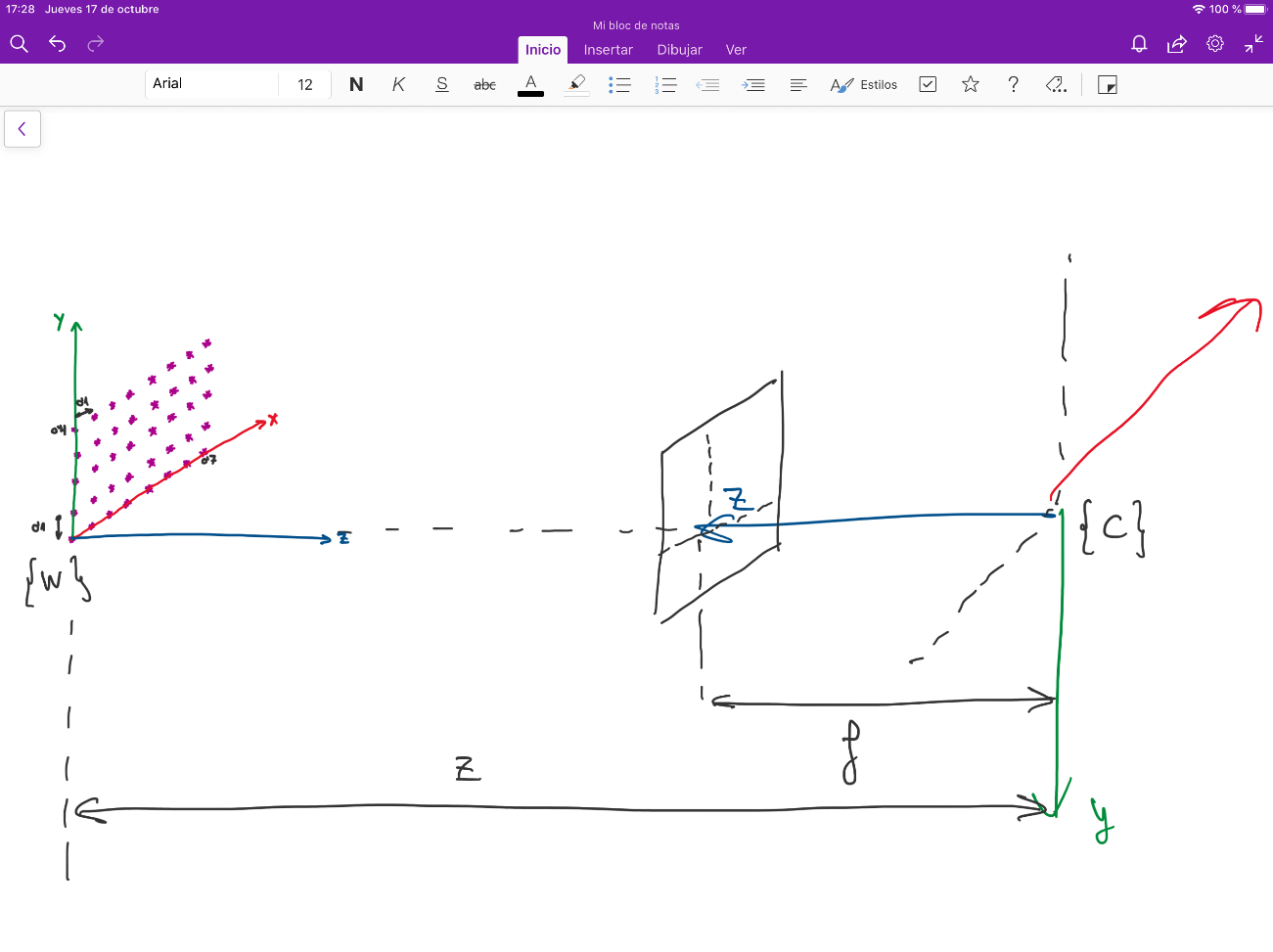


Ilustración 3. Esquema del problema planteado

Nota: cabe destacar que en el planteamiento del problema, se emplean los ejes móviles, por lo que iría posmultiplicando y el orden de las operaciones empleadas en el script son: Rx, Ry, Rz, traslación.

El resto de operaciones realizadas en el script, son las pertinentes para ajustar los ejes de la cámara e ir interpretando en un plot lo que se ve en cada momento en el sensor cámara conforme se mueve.

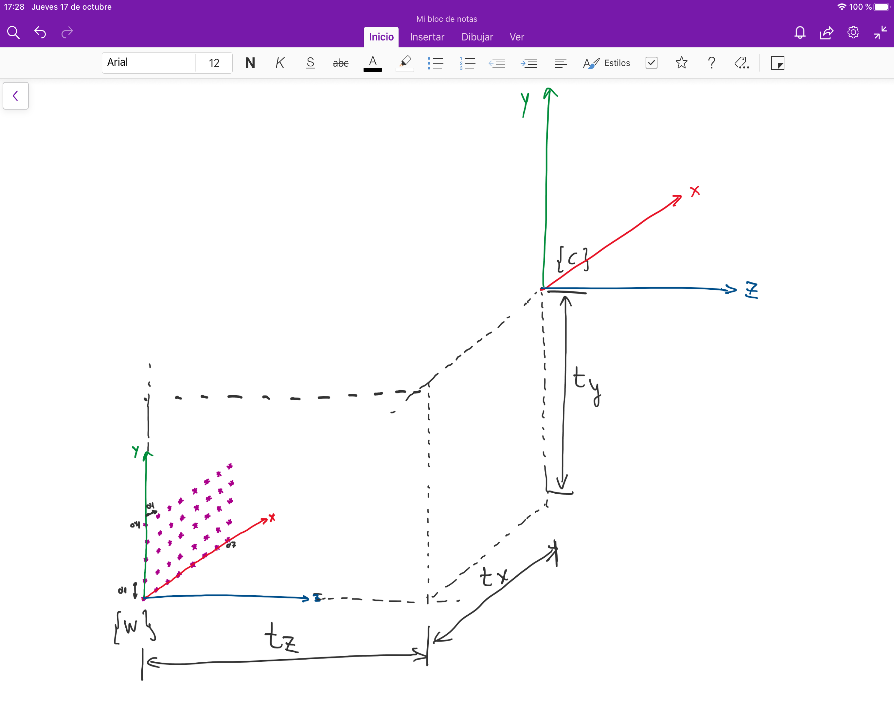


Ilustración 4. Esquema del procedimiento detallado. Traslación

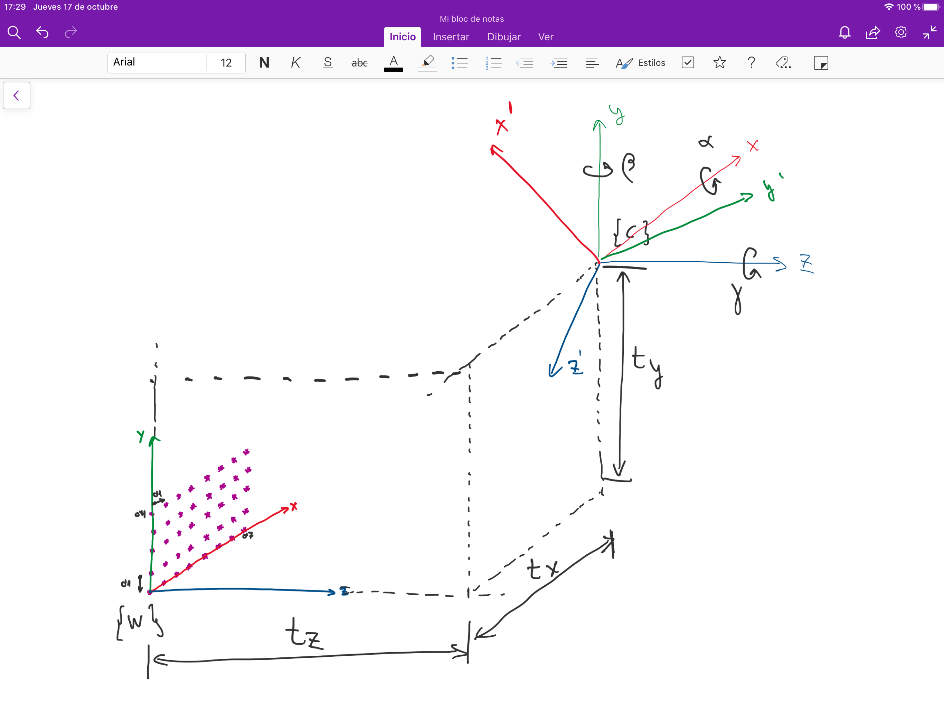


Ilustración 5. Esquema del procedimiento detallado. Traslación y rotación

El procedimiento del script es, en resumen, para el caso de con distorsión y sin distorsión:

1. Elegir dimensiones del tablero y rellenar en una matriz. Seguidamente, se representa el resultado.
2. Elegir posición y orientación inicial de la cámara.
3. Ajustar el sistema de coordenadas de la cámara respecto al tablero {W}.
4. Obtener la matriz de puntos del tablero representado en el sensor de la cámara y representar el resultado.

# Caso 1: sin distorsión

Se va a considerar, para empezar, el caso sin distorsión por ser el más sencillo. El código que hace el procedimiento del problema en este caso es el siguiente (se adjunta el fichero en la entrega del trabajo):

% En primer lugar, se va a hacer una matriz que simule el tablero de

% ajedrez que se desea proyectar en el sensor de la cámara.

clear all

close all

clc

%i --> fila; j --> columna

Naj = 8; % Número de columnas de tablero de ajedrez

Maj = 5; % Número de filas de tablero de ajedrez

M\_P = zeros(3,Naj\*Maj); % Matriz de puntos inicialmente relleno de ceros

separacion = 0.1; % Tanto entre puntos horizontales como verticales [m]

% Ángulos de giro de los ejes de la camara

alpha = 15\*(pi/180); %x

beta = 20\*(pi/180); %y

gamma = 35\*(pi/180); %z

% Posición inicial de la cámara:

x = 0.5;

y = 0.5;

z = 2;

% Generamos las coordenadas respecto al mundo de los puntos del tablero

for j = 1:Maj % Por filas del tablero

for i = 1:Naj % Por columnas del tablero

wM\_P(:,(i+(j-1)\*Naj)) = [separacion\*(i-1) separacion\*(j-1) 0];

wM\_P\_(:,(i+(j-1)\*Naj)) = [wM\_P(:,(i+(j-1)\*Naj));1]; %Matriz en homogéneas

end

end

figure(1);

plot(wM\_P(1,:),wM\_P(2,:),'\*'); grid; hold on; %Mostramos el tablero

title('Representación del tablero');

xlabel('Horizontal (m)'); ylabel('Horizontal (m)');

rectangle('Position', [0 0 separacion\*(Naj-1) separacion\*(Maj-1)]);

% Parámetros de la cámara

f = 4.2e-3; %Distancia focal (m)

N = 4128; %Número de píxeles horizontales

M = 3096; %Número de píxeles verticales

w = 4.96e-3; %Anchura de sensor de la cámara

h = 3.52e-3; %Altura de sensor de la cámara

u0 = round(N/2) + 1; %Offset de u (proyección en lente)

v0 = round(M/2) -2; %Offset de v (proyección en lente)

rho\_x = w/N; %Anchura efectiva del píxel

rho\_y = h/M; %Altura efectiva del píxel

fx = f/rho\_x; %Distancia efectiva horizontal (m)

fy = f/rho\_y; %Distancia efectiva vertical (m)

s = 0; %Skew=0

% Matriz de parámetros intrinsecos

A = [fx s\*fx u0

0 fy v0

0 0 1];

% Matrices de rotación

Rx = [1 0 0

0 cos(alpha) -sin(alpha)

0 sin(alpha) cos(alpha)];

Ry = [ cos(beta) 0 sin(beta)

0 1 0

-sin(beta) 0 cos(beta)];

Rz = [cos(gamma) -sin(gamma) 0

sin(gamma) cos(gamma) 0

0 0 1];

%Coeficientes de distorsión

%Radial

kr1 = 0.144; kr2 = -0.307;

%Tangencial

kt1 = -0.0032; kt2 = 0.0017;

El código escrito hasta ahora, solo representa lo que aparece en el sensor de la cámara en el primer momento. Para la animación:

%% Procedimiento del ejercicio sin distorsión en la lente

% Matriz de rotación de la cámara respecto al mundo (rotación x de 180º):

wRc = Rx\*Ry\*Rz;

% Matriz de traslación de la cámara respecto al mundo:

wtc = [x y z]';

% Por tanto, la matriz de transformación homogénea quedaría (PRIMERO

% TRASLADAR, LUEGO ROTAR):

wTc = [eye(3) [wtc]; [0 0 0 1]]\*[wRc [0;0;0]; [0 0 0 1]];

%Para poner la matriz de transformación homogénea respecto a la cámara:

cTw = inv(wTc);

%De donde se obtienen los parámetros extrínsecos:

cRw = cTw(1:3,1:3);

ctw = cTw(1:3,4);

%Ahora, se calculan los puntos proyectados homogéneos respecto a la

%cámara:

mp\_ = A\*[cRw ctw]\*wM\_P\_;

%La matriz de puntos proyectados queda, por tanto:

for i=1:Naj\*Maj

mp(:,i) = mp\_(1:2,i)/mp\_(3,i);

end

figure(2);

plot(mp(1,:),mp(2,:),'\*'); grid; hold on; %Mostramos el tablero

axis([-1 N+1 -1 M+1]);

title('Proyección de la imagen captada por el sensor de la cámara SIN DISTORSIÓN');

xlabel('pix'); ylabel('pix');

rectangle('Position', [0 0 N M]); hold off;

aaaa

for i = 0:9

% Ángulos de giro de los ejes de la camara

alpha = 180\*(pi/180); %x

beta = 5\*i\*(pi/180); %y

gamma = 2\*i\*(pi/180); %z

% Posición inicial de la cámara:

x = 0.5+0.05\*i;

y = 0.5-0.05\*i;

z = 2-0.15\*i;

% Matrices de rotación

Rx = [1 0 0

0 cos(alpha) -sin(alpha)

0 sin(alpha) cos(alpha)];

Ry = [ cos(beta) 0 sin(beta)

0 1 0

-sin(beta) 0 cos(beta)];

Rz = [cos(gamma) -sin(gamma) 0

sin(gamma) cos(gamma) 0

0 0 1];

% Procedimiento del ejercicio sin distorsión en la lente

wRc = Rx\*Ry\*Rz;

wtc = [x y z]';

wTc = [eye(3) [wtc]; [0 0 0 1]]\*[wRc [0;0;0]; [0 0 0 1]];

cTw = inv(wTc);

cRw = cTw(1:3,1:3);

ctw = cTw(1:3,4);

mp\_ = A\*[cRw ctw]\*wM\_P\_;

for i=1:Naj\*Maj

mp(:,i) = mp\_(1:2,i)/mp\_(3,i);

end

figure(1);

plot3(wM\_P(1,:), wM\_P(2,:), wM\_P(3,:), 'm\*');grid;hold on;

plot3([0 (Naj-1)\*separacion],[0 0],[0 0], 'r'); % eje x del tablero

plot3([0 0],[0 (Maj-1)\*separacion],[0 0], 'g'); % eje y del tablero

plot3([0 0],[0 0],[0 1], 'b'); % eje z del tablero

% Camara

plot3(x, y, z, 'm\*'); % Origen de la camara

ejex\_c = wTc\*[0.1 0 0 1]';

ejey\_c = wTc\*[0 0.1 0 1]';

ejez\_c = wTc\*[0 0 0.1 1]';

plot3([x ejex\_c(1)],[y ejex\_c(2)],[z ejex\_c(3)], 'r'); % eje x de la camara

plot3([x ejey\_c(1)],[y ejey\_c(2)],[z ejey\_c(3)], 'g'); % eje y de la camara

plot3([x ejez\_c(1)],[y ejez\_c(2)],[z ejez\_c(3)], 'b'); % eje z de la camara

title('Vista externa de la posicion relativa Camara-Tablero');

xlabel('X [m]'); ylabel('Y [m]'); zlabel('Z [m]');hold off;

% Vista desde la cámara sin distorsionar

figure(2);

plot(mp(1,:),mp(2,:),'m\*'); grid; hold on;

axis([-1 N+1 -1 M+1]);

title('Proyección de la imagen captada por el sensor de la cámara SIN DISTORSIÓN');

xlabel('pix'); ylabel('pix');

rectangle('Position', [0 0 N M]); hold off;

pause();

end

# Caso 2: con distorsión

El código empleado para resolver el problema este caso es el siguiente (está adjunto el fichero completo con el trabajo):

% En primer lugar, se va a hacer una matriz que simule el tablero de

% ajedrez que se desea proyectar en el sensor de la cámara.

clear all

close all

clc

%i --> fila; j --> columna

Naj = 8; % Número de columnas de tablero de ajedrez

Maj = 5; % Número de filas de tablero de ajedrez

M\_P = zeros(3,Naj\*Maj); % Matriz de puntos inicialmente relleno de ceros

separacion = 0.1; % Tanto entre puntos horizontales como verticales [m]

% Ángulos de giro de los ejes de la camara

alpha = 15\*(pi/180); %x

beta = 20\*(pi/180); %y

gamma = 35\*(pi/180); %z

% Posición inicial de la cámara:

x = 0.5;

y = 0.5;

z = 2;

% Generamos las coordenadas respecto al mundo de los puntos del tablero

for j = 1:Maj % Por filas del tablero

for i = 1:Naj % Por columnas del tablero

wM\_P(:,(i+(j-1)\*Naj)) = [separacion\*(i-1) separacion\*(j-1) 0];

wM\_P\_(:,(i+(j-1)\*Naj)) = [wM\_P(:,(i+(j-1)\*Naj));1]; %Matriz en homogéneas

end

end

figure(1);

plot(wM\_P(1,:),wM\_P(2,:),'\*'); grid; hold on; %Mostramos el tablero

title('Representación del tablero');

xlabel('Horizontal (m)'); ylabel('Horizontal (m)');

rectangle('Position', [0 0 separacion\*(Naj-1) separacion\*(Maj-1)]);

% Parámetros de la cámara

f = 4.2e-3; %Distancia focal (m)

N = 4128; %Número de píxeles horizontales

M = 3096; %Número de píxeles verticales

w = 4.96e-3; %Anchura de sensor de la cámara

h = 3.52e-3; %Altura de sensor de la cámara

u0 = round(N/2) + 1; %Offset de u (proyección en lente)

v0 = round(M/2) -2; %Offset de v (proyección en lente)

rho\_x = w/N; %Anchura efectiva del píxel

rho\_y = h/M; %Altura efectiva del píxel

fx = f/rho\_x; %Distancia efectiva horizontal (m)

fy = f/rho\_y; %Distancia efectiva vertical (m)

s = 0; %Skew=0

% Matriz de parámetros intrinsecos

A = [fx s\*fx u0

0 fy v0

0 0 1];

% Matrices de rotación

Rx = [1 0 0

0 cos(alpha) -sin(alpha)

0 sin(alpha) cos(alpha)];

Ry = [ cos(beta) 0 sin(beta)

0 1 0

-sin(beta) 0 cos(beta)];

Rz = [cos(gamma) -sin(gamma) 0

sin(gamma) cos(gamma) 0

0 0 1];

%Coeficientes de distorsión

%Radial

kr1 = 0.144; kr2 = -0.307;

%Tangencial

kt1 = -0.0032; kt2 = 0.0017;

%% Procedimiento del ejercicio con distorsión en la lente

% En primer lugar, es necesario poner los puntos del tablero de ajedrez

% (wM\_P) referenciados al sistema de referencia de la cámara (cM\_P):

% Matriz de rotación de la cámara respecto al mundo (rotación x 180º):

wRc = Rx\*Ry\*Rz;

% Matriz de traslación de la cámara respecto al mundo:

wtc = [x y z]';

% Por tanto, la matriz de transformación homogénea quedaría (PRIMERO

% TRASLADAR, LUEGO ROTAR):

wTc = [eye(3) [wtc]; [0 0 0 1]]\*[wRc [0;0;0]; [0 0 0 1]];

% Para poner la matriz de transformación homogénea respecto a la cámara:

cTw = inv(wTc);

% De donde se obtienen los parámetros extrínsecos:

cRw = cTw(1:3,1:3);

ctw = cTw(1:3,4);

%De este modo, se obtiene los puntos del tablero de ajedrez referenciados a

%la cámara (no se puede multiplicar directamente por A porque la distorsión

%va después de la proyección):

cM\_P = [cRw ctw] \* wM\_P\_;

% A continuación, se proyectarán los puntos NORMALIZADOS:

for i = 1:Naj\*Maj

M\_p\_norm(:,i) = [cM\_P(1,i)/cM\_P(3,i);cM\_P(2,i)/cM\_P(3,i)]; %cX/cZ;cY/cZ

end

% Una vez obtenidos los puntos proyectados normalizados, se aplican las

% distorsiones radial y tangencial a cada punto de la matriz:

% Vector de r^2 (para cada punto):

for i = 1:Naj\*Maj

r2(i) = M\_p\_norm(1,i)^2 + M\_p\_norm(2,i)^2; %ri^2 = Xni^2 + Yni^2

end

% Vector de distorsión radial:

for i = 1:Naj\*Maj

dist\_r(i) = 1 + kr1\*r2(i) + kr2\*r2(i)^2;

end

% Matriz de distorsión tangencial:

for i = 1:Naj\*Maj

dist\_t(:,i) = [ (2\*kt1\*M\_p\_norm(1,i)\*M\_p\_norm(2,i)) + kt2\*(r2(i)+2\*M\_p\_norm(1,i)^2) ;

(2\*kt2\*M\_p\_norm(1,i)\*M\_p\_norm(2,i)) + kt1\*(r2(i)+2\*M\_p\_norm(2,i)^2)];

end

% Por tanto, la matriz de puntos normalizados y distorsionados quedaría:

for i = 1:Naj\*Maj

m\_p\_norm\_dist(:,i) = [M\_p\_norm(1,i);M\_p\_norm(2,i)] \* dist\_r(i) + [dist\_t(1,i);dist\_t(2,i)];

end

% Ahora, para obtener los puntos distorsionados (sin normalizar) y

% proyectados:

for i = 1:Naj\*Maj

m\_p\_dist(:,i) = [m\_p\_norm\_dist(1,i)\*fx + u0 ; m\_p\_norm\_dist(2,i)\*fy + v0 ];

end

% Representando:

figure(3);

plot(m\_p\_dist(1,:),m\_p\_dist(2,:),'\*'); grid; hold on; %Mostramos el tablero

title('Proyección de la imagen captada por el sensor de la cámara CON DISTORSIÓN');

xlabel('pix'); ylabel('pix');

rectangle('Position', [0 0 N M]); hold off;

Para hacer la animación:

for i = 0:9

% Ángulos de giro de los ejes de la camara

alpha = 180\*(pi/180); %x

beta = 5\*i\*(pi/180); %y

gamma = 2\*i\*(pi/180); %z

% Posición inicial de la cámara:

x = 0.5+0.05\*i;

y = 0.5-0.05\*i;

z = 2-0.15\*i;

% Matrices de rotación

Rx = [1 0 0

0 cos(alpha) -sin(alpha)

0 sin(alpha) cos(alpha)];

Ry = [ cos(beta) 0 sin(beta)

0 1 0

-sin(beta) 0 cos(beta)];

Rz = [cos(gamma) -sin(gamma) 0

sin(gamma) cos(gamma) 0

0 0 1];

% Procedimiento del ejercicio sin distorsión en la lente

wRc = Rx\*Ry\*Rz;

wtc = [x y z]';

wTc = [eye(3) [wtc]; [0 0 0 1]]\*[wRc [0;0;0]; [0 0 0 1]];

cTw = inv(wTc);

cRw = cTw(1:3,1:3);

ctw = cTw(1:3,4);

mp\_ = A\*[cRw ctw]\*wM\_P\_;

for i=1:Naj\*Maj

mp(:,i) = mp\_(1:2,i)/mp\_(3,i);

end

figure(1);

plot3(wM\_P(1,:), wM\_P(2,:), wM\_P(3,:), 'm\*');grid;hold on;

plot3([0 (Naj-1)\*separacion],[0 0],[0 0], 'r'); % eje x del tablero

plot3([0 0],[0 (Maj-1)\*separacion],[0 0], 'g'); % eje y del tablero

plot3([0 0],[0 0],[0 1], 'b'); % eje z del tablero

% Camara

plot3(x, y, z, 'm\*'); % Origen de la camara

ejex\_c = wTc\*[0.1 0 0 1]';

ejey\_c = wTc\*[0 0.1 0 1]';

ejez\_c = wTc\*[0 0 0.1 1]';

plot3([x ejex\_c(1)],[y ejex\_c(2)],[z ejex\_c(3)], 'r'); % eje x de la camara

plot3([x ejey\_c(1)],[y ejey\_c(2)],[z ejey\_c(3)], 'g'); % eje y de la camara

plot3([x ejez\_c(1)],[y ejez\_c(2)],[z ejez\_c(3)], 'b'); % eje z de la camara

title('Vista externa de la posicion relativa Camara-Tablero');

xlabel('X [m]'); ylabel('Y [m]'); zlabel('Z [m]');hold off;

% Procedimiento del ejercicio con distorsión en la lente

cM\_P = [cRw ctw] \* wM\_P\_;

for i = 1:Naj\*Maj

M\_p\_norm(:,i) = [cM\_P(1,i)/cM\_P(3,i);cM\_P(2,i)/cM\_P(3,i)];

end

for i = 1:Naj\*Maj

r2(i) = M\_p\_norm(1,i)^2 + M\_p\_norm(2,i)^2; %ri^2 = Xni^2 + Yni^2

end

for i = 1:Naj\*Maj

dist\_r(i) = 1 + kr1\*r2(i) + kr2\*r2(i)^2;

end

for i = 1:Naj\*Maj

dist\_t(:,i) = [ (2\*kt1\*M\_p\_norm(1,i)\*M\_p\_norm(2,i)) + kt2\*(r2(i)+2\*M\_p\_norm(1,i)^2) ;

(2\*kt2\*M\_p\_norm(1,i)\*M\_p\_norm(2,i)) + kt1\*(r2(i)+2\*M\_p\_norm(2,i)^2)];

end

for i = 1:Naj\*Maj

m\_p\_norm\_dist(:,i) = [M\_p\_norm(1,i);M\_p\_norm(2,i)] \* dist\_r(i) + [dist\_t(1,i);dist\_t(2,i)];

end

for i = 1:Naj\*Maj

m\_p\_dist(:,i) = [m\_p\_norm\_dist(1,i)\*fx + u0 ; m\_p\_norm\_dist(2,i)\*fy + v0 ];

end

% Vista desde la cámara con distorsion

figure(3);

plot(m\_p\_dist(1,:),m\_p\_dist(2,:),'m\*'); grid; hold on; %Mostramos el tablero

axis([-1 N+1 -1 M+1]);

title('Proyección de la imagen captada por el sensor de la cámara CON DISTORSIÓN');

xlabel('pix'); ylabel('pix');

rectangle('Position', [0 0 N M]); hold off;

pause();

end

# Conclusiones

A priori, al realizar el trabajo, no se apreciaba una diferencia notoria del caso sin distorsión respecto al caso distorsionado, por lo que decidimos aumentar la distancia de separación entre los puntos del tablero significativamente, además de modificar la distancia a la que se encontraba la cámara del tablero.

La situación de la que se partía:



