Atik Faysal

1. Study the lecture note by showing the use of projection matrix that maps objects from 3-D to 2-D, the definition of intrinsic and extrinsic parameters of a camera, and the linear approach to geometric camera calibration.

The camera lenses project the 3-D world frame to a 2-D camera frame coordinate. Mathematical equations can define this perspective if all distances are measured in the camera’s reference frame. Two sets of parameters control this relation between the world and camera coordinate systems.

* Intrinsic parameters relates the camera’s coordinate system to the image pixel coordinate system.
* Extrinsic parameters relate the camera’s coordinate system to a fixed world coordinate system and specify its position and orientation in space.

The geometric camera calibration is used to process these intrinsic and extrinsic parameters.

Diagram

Description automatically generated

Figure: Capturing an image from the 3-D world coordinates to 2-D camera coordinates.

How the camera lenses work as a physical retina is described below:

Diagram

Description automatically generated

Figure: Physical and normalized coordinate system.

The 3D to 2D perspective projection on the normalized image plane is given by the equation:

Here represents a homogeneous coordinate vector in the camera coordinates.

2D coordinates represented by can be computed with for coordinates 3D points and perspective projection matrix.

Where, and

Again, . So,

Where, and

Lets consider the case where the camera frame (C) is distinct from the world frame (W).

The projection matrix is obtained as,

Where, and

is a rotation matrix

is a translation vector

denotes the homogeneous coordinate vector of ***P*** is the frame ***W***.

A projection matrix is written explicitly as a function of both intrinsic and extrinsic parameters which are:

Five intrinsic parameters, and six extrinsic parameters.

A projection matrix can be written explicitly as a function of both intrinsic and extrinsic parameters as

Follows:

And for all 3D point we will have

and *m* =

So that we may obtain the intrinsic and extrinsic parameters.

(rotation vector)

(shift vector)

1. Given an image *test\_image.bmp* captured by a camera which is posed toward a 3D chess board, we manually selected 27 points whose 2-D image coordinates are saved in the file *observe.dat*, and whose 3-D coordinates in a world coordinate are stored in the file *model.dat*. Please use the linear approach to calibrate the camera system by computing the projection matrix (M) from which you are required to compute the intrinsic and extrinsic parameters, including θ, u0, v0, α, β, and the rotation matrix (R), and the shift vector (t).

R =

Chart, scatter chart

Description automatically generated

The above figure represents our simulated and calculated project. It definitely contains some errors, which is why not not optimize to exact 0.

0.0011

0.0014

-0.0005

-0.0003

-0.0001

-0.0002

0.0003

-0.0001

-0.0011

-0.0001

-0.0006

-0.0001

0.0000

-0.0001

-0.0011

-0.0001

-0.0004

-0.0002

0.0004

-0.0003

0.0013

-0.0004

-0.0002

-0.0001

0.0004

0.0002

0.0009

0.0005

-0.0004

0.0008

0.0002

-0.0008

0.0007

-0.0005

-0.0006

-0.0002

-0.0000

0.0001

0.0005

0.0004

-0.0008

-0.0011

0.0001

0.0011

0.0001

0.0001

0.0000

-0.0007

0.0000

0.0005

-0.0000

0.0001

-0.0000

-0.0002

1. In order to verify the correctness of the camera calibration, please use the projection matrix to map the three set of 3D points, whose 3-D coordinates are {x,y,0|x,y=0,…,10}, {x,10,z|x,z=0,…,10}, {10,y,z|y,z=0,…,10}, respectively, into the 2-D image space. Discuss your results.

For all three coordinate systems in the 3D plane, we obtain the following Figure by combining the {x,y,0|x,y=0,…,10}, {x,10,z|x,z=0,…,10}, {10,y,z|y,z=0,…,10}, respectively, into the 2-D image space.

Chart

Description automatically generated

For {x,y,0|x,y=0,…,10}, the projection matrix generates the figure below:

Chart

Description automatically generated

For {x,10,z|x,z=0,…,10}, the projection matrix generates the figure below:

Chart

Description automatically generated

For {10,y,z|y,z=0,…,10}, the projection matrix generates the figure below:

A picture containing text, building material, tile, accessory

Description automatically generated

1. You are encouraged to discuss the effect of using different numbers 3D-2D point pairs on the calibration performance. Also, test this linear approach to see if it can handle the planar case when all 3D points are coplanar.
2. Create a video file that shows a 3D object moving in the 3D scene along a specific pre-defined path. For example, a 3D cube (1x1x1) can be displayed by nine lines connecting seven vertexes (see the example below). Each line can be drawn with around 100 samples. (You can see a video example from Slide 13 of the Lecture 8 handout in the slideshow mode.)

clc;clear;

%% Brief: Record video using writeVideo()

%% For the project, you can use patch() to refresh the new image

T = imread('test\_image.bmp');

v = VideoWriter('ObjectMoving.avi');

open(v);

figure

imshow(T);

for i=1:100

L=T;

% some processing in L

Frame(:,:,1)=L/i\*2; % Red channel

Frame(:,:,2)=L/i\*2; % Blue channel

Frame(:,:,3)=L/i\*2; % Green channel

Mo(i)=im2frame(Frame);

writeVideo(v, Mo)

end

% Mo = getframe;

% for tmp = 0:100

% writeVideo(v, Mo)

% end

close(v)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The Matlab source file:

clc; clear;

%%%%%%%%%%%%%%%%%% Problem 2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

data\_2d = load('observe.dat'); % Load 2D points as camera coordinates

data\_3d\_tmp = load('model.dat'); % Load 3D points as world coordinates

data\_3d = zeros(size(data\_3d\_tmp, 1), 4);

data\_3d(:, 1:3) = data\_3d\_tmp;

data\_3d(:, 4) = 1;

img = imread('test\_image.bmp'); % Import test image

[lx, ly] = size(img);

figure(1), imshow(img);

[On, Ot] = size(data\_2d);

for i=1:On

mx=data\_2d(i,1);

my=data\_2d(i,2);

for j=mx-2:mx+2

for k=my-2:my+2

img(k,j)=0; % Getting all the 2D coordinates from image

end

end

end

figure(2), imshow(img);

dots = size(data\_3d, 1);

P = zeros(2\*dots, 12);

for i = 1:dots

P((i-1)\*2+1, 1:4) = data\_3d(i, :);

P(i\*2, 5:8) = data\_3d(i, :);

P((i-1)\*2+1, 9:12) = -data\_2d(i, 1) \* data\_3d(i, :);

P(i\*2, 9:12) = -data\_2d(i, 2) \* data\_3d(i, :);

end

Q = P; % Q matrix of dimension of 2n x 12

[V,D] = eig(Q'\*Q);

for i = 1:size(D,1)

D\_all(i) = D(i,i);

end

[~, min\_index] = min(D\_all);

A = V(:, min\_index);

M = [A(1:4)';A(5:8)';A(9:12)'];

a1 = M(1, 1:3)';

a2 = M(2, 1:3)';

a3 = M(3, 1:3)';

b = M(:,4);

rho = 1 / norm(a3, 1);

r3 = rho\*a3;

r3 = r3';

u0 = (rho^2)\*dot(a1, a3);

v0 = (rho^2)\*dot(a2, a3);

cosTheta = -dot(cross(a1,a3),cross(a2,a3))/(norm(cross(a1,a3),2)\*norm(cross(a2,a3),2));

sinTheta = sqrt(1-cosTheta^2);

alpha = (rho^2)\*norm(cross(a1, a3),1)\*sinTheta;

beta = (rho^2)\*norm(cross(a2, a3), 1)\*sinTheta;

tetha = acosd(cosTheta);

r1 = cross(a2,a3)/norm(cross(a2,a3),2);

r2 = cross(r3,r1);

K = [alpha -alpha\*cot(tetha) u0;0 beta/sinTheta v0;0 0 1]; % Intrinsic Matrix

t = rho \* (b\inv(K)); % Translation Matrix

R = [r1' ;r2 ;r3]; % Rotation Matrix

% Optimization error

m\_hat = Q \* V(:, min\_index);

d2d\_output = zeros(size(data\_3d, 1), 3);

for i = 1:size(data\_3d, 1)

d2d\_output(i, :) = M\*data\_3d(i, :)'/(M(3, :) \* data\_3d(i, :)');

end

d2d = d2d\_output(:, 1:2);

d2d\_output = int16(d2d\_output); % Simulated coordinates

I = imread('test\_image.bmp');

figure(2);

imshow(I);

hold on;

for i = 1:length(d2d\_output)

plot(d2d\_output(i,1),d2d\_output(i,2),'+r', 'Linewidth', 2);

end

xlabel('Y'); ylabel('X');

title('\bf Calculated and Plotted points on the same image');

%%%%%%%%%%%%%%%%%% Problem 3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

cor = length(data\_3d\_tmp);

% Pixel cordiantesfrom the projection matrix

for loc=1:1:cor

pixel(:,loc) = transpose([data\_3d\_tmp(loc,:) 1]);

Z(loc) = dot(M(3,:),pixel(:,loc));

end

for loc=1:1:cor

u(loc,1) = dot(M(1,:),pixel(:,loc))/Z(loc);

v(loc,1) = dot(M(2,:),pixel(:,loc))/Z(loc);

end

P = horzcat(u, v);

% World cordiantes

col = 1;

for i=0:1:10

for j=0:1:10

z0(col,:) = [i j 0];

col = col+1;

end

end

col = 1;

for i=0:1:10

for j=0:1:10

y10(col,:) = [i 10 j];

col = col+1;

end

end

col = 1;

for i=0:1:10

for j=0:1:10

x10(col,:) = [10 i j];

col = col+1;

end

end

% For projection matrix

for i=1:1:length(z0)

z00(:,i) = transpose([z0(i,:) 1]);

Z(i) = dot((M(3, :)),z00(:,i));

end

for i=1:1:length(z0)

u1(i,1) = dot((M(1, :)),z00(:,i))/Z(i);

v1(i,1) = dot((M(2, :)),z00(:,i))/Z(i);

end

P1 = horzcat(u1, v1);

for i=1:1:length(y10)

y00(:,i) = transpose([y10(i,:) 1]);

Z(i) = dot((M(3, :)),y00(:,i));

end

for i=1:1:length(y10)

u2(i,1) = dot((M(1, :)),y00(:,i))/Z(i);

v2(i,1) = dot((M(2, :)),y00(:,i))/Z(i);

end

P2 = horzcat(u2, v2);

for i=1:1:length(x10)

x00(:,i) = transpose([x10(i,:) 1]);

Z(i) = dot((M(3, :)),x00(:,i));

end

for i=1:1:length(x10)

u3(i,1) = dot((M(1, :)),x00(:,i))/Z(i);

v3(i,1) = dot((M(2, :)),x00(:,i))/Z(i);

end

P3 = horzcat(u3, v3);

I=imread('test\_image.bmp'); % Load image

[Ix, Iy]=size(I); % dimension size

figure(1),

imshow(I);

for i=1:length(x10)

mx=floor(P3(i,1));

my=floor(P3(i,2));

for m1=mx-2:mx+2 % Marking the coordinate

for n1=my-2:my+2

I(n1,m1)=0;

end

end

end

for i=1:length(y10)

mx=floor(P2(i,1));

my=floor(P2(i,2));

for m1=mx-2:mx+2

for n1=my-2:my+2

I(n1,m1)=0;

end

end

end

for i=1:length(z00)

mx=floor(P1(i,1));

my=floor(P1(i,2));

for m1=mx-2:mx+2

for n1=my-2:my+2

I(n1,m1)=0;

end

end

end

figure(2),

hold on

imshow(I); % show the simulated figure