EECS 442 Computer Vision Homework 03

Fan Bu
UMID: 68836435
Unique Name: fanbu

10/24/2016

Question 1

(a)

From question, we know that a line

$$l = \left\{ r \begin{bmatrix} cos\alpha \\ cos\beta \\ cos\gamma \end{bmatrix} \middle| \quad r \in \mathbb{R} \right\}$$

can be interpreted in homogeneous coordinate system as

$$l = \left\{ \begin{bmatrix} cos\alpha \\ cos\beta \\ cos\gamma \\ \frac{1}{r} \end{bmatrix} \middle| \quad r \in \mathbb{R} \right\}$$

As the world coordinate radius of image plane goes to infinity, parallel lines will intersect at one point. When r goes to infinity, line l will reach the vanishing point. Therefore, by converting between world coordinates and image coordinates, we can write the vanishing point as

$$v = \lim_{r\to\infty} \begin{bmatrix} K & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} cos\alpha \\ cos\beta \\ cos\gamma \\ \frac{1}{r} \end{bmatrix}$$

$$\begin{aligned} \therefore v &= \begin{bmatrix} K & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} cos\alpha \\ cos\beta \\ cos\gamma \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} K & 0 \end{bmatrix} \begin{bmatrix} Rd \\ 0 \end{bmatrix} \\ &= KRd \end{aligned}$$

where K is the camera calibration matrix and R is the rotation matrix between the camera and the world coordinates.

(b)

First, show R^{-1} exists:
We know that the rotation matrix

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta_x & -sin\theta_x \\ 0 & sin\theta_x & cos\theta_x \end{bmatrix} \begin{bmatrix} cos\theta_y & 0 & -sin\theta_y \\ 0 & 1 & 0 \\ sin\theta_y & 0 & cos\theta_y \end{bmatrix} \begin{bmatrix} cos\theta_z & -sin\theta_z & 0 \\ sin\theta_z & cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

And

$$det(R) = 1 \cdot 1 \cdot 1 = 1$$

Thus, R^{-1} exists.

Then show that K^{-1} exists:
For camera matrix

$$K = \begin{bmatrix} fk_u & fk_u cot\theta & u_0 \\ 0 & \frac{fk_v}{sin\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Since $det(K) = \frac{fk_u \cdot fk_v}{sin\theta} \neq 0$, K^{-1} exists.
According to what above, $d = R^{-1}K^{-1}v$.

(c)

From section (b), we know that $d = R^{-1}K^{-1}v$, where K^{-1} and R^{-1} exists.
Therefore, $Rd = K^{-1}v$
Then, for any rotation matrix such that, $R^TR = I$, we have:

$$\begin{aligned} d_i^T d_j &= 0 \\ d_i^T (R^T R) d_j &= 0 \\ \therefore (Rd_i)^T R d_j &= 0 \\ \therefore (K^{-1}v_i)^T (K^{-1}v_j) &= 0 \end{aligned}$$

for $i \neq j$

Question 2

(a)

No. We need to calibrate the camera with at least two amages. If only one image is used, the linear system becomes rank deficient which means that we'll get something that is not invertible. Since the camera matrix is computed using inverse, we connot obtain a unique solution.

(b)

The original linear system is:

$$x = \begin{bmatrix} x_1 & ... & x_n \\ y_1 & ... & y_n \end{bmatrix}$$

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$X = \begin{bmatrix} X_1 & ... & X_n \\ Y_1 & ... & Y_n \\ Z_1 & ... & Z_n \\ 1 & ... & 1 \end{bmatrix}$$

$$x = PX$$

By expanding out the first 3D world to 2D point correspondences, we obtain:

$$Ax = b$$

where

$$A = \begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 \\ ... \end{bmatrix}, x = \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \end{bmatrix}, b = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ ... \end{bmatrix}$$

Solve by Linear least square:

$$x = (A^T A)^{-1} A^T b$$

Same reasoning, $P = xX(XX^T)^{-1}$.
Then, the Euclidean distance between the projected points and the actual point on the image is obtained using the following definition of RMS:

$$RMS = \sqrt{\sum_{i=1}^{N} \frac{1}{N}((x_i - \bar{x}_i)^2 + (y_i - \bar{y}_i)^2)}$$

where N is the number of points.

(b)

By selecting 12 points from each picture(24 points in total), the RMS error (in pixels) between the corner positions and the predicted by the camera parameters is:

$$RMS\ error = 1.3175$$

Note: codes are attached

(c)

$$Camera\ Matrix = \begin{bmatrix} 1.3543 & 0.0477 & 0.0729 & 157.5264 \\ -0.0374 & 1.3308 & 0.3560 & 59.5406 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Note: codes are attached

Matlab Codes

```
function main()
clear all;
close all;
clc;
%% Load in images
image1 = imread('1.JPG');
image2 = imread('2.JPG');
figure, imagesc(image1); colormap(gray);
title('Click a point on this image'); axis image;
%% Select corner - make sure to click in order corresponding to A
%% If you want to use my data, just comment this subblock
% num_of_points=12;
% a = [];
% for i = 1:num_of_points
%     [x y] = ginput(1);
%     a = [a [x;y]]
%     image1(round(y),round(x)) = 255;
%     imagesc(image1); colormap(gray);
%     title('Click a point on this image');
%     axis image;
% end
% figure, imagesc(image2); colormap(gray);
% title('Click a point on this image');
% axis image;
% for i = 1:num_of_points
%     [x y] = ginput(1);
%     a = [a [x;y]]
%     image1(round(y),round(x)) = 255;
%     imagesc(image2); colormap(gray);
%     title('Click a point on this image');
%     axis image;
% end
% save ground_truth
%% Load my data
load ground_truth.mat
%% 3D coordinates
```

```
%Build the reference A matrix
A = [0 0 0]';
for i=1:5
    A = [A A(:,end)];
    A(1,end)=A(1,end)+35;
end
A = [A [0 35 0]'];
for i=1:5
    A = [A A(:,end)];
    A(1,end)=A(1,end)+35;
end
A = [A [0 0 9.5]'];
for i=1:5
    A = [A A(:,end)];
    A(1,end)=A(1,end)+35;
end
A = [A [0 35 9.5]'];
for i=1:5
    A = [A A(:,end)];
    A(1,end)=A(1,end)+35;
end
A = [A;ones(1,size(A,2)) ]
A_annot = [a; ones(1, size(a,2))]
%Solve using least squares
P = Calc_P(A', a')
%RMS Calculation
A_bar = P * A
RMS_error = sqrt(mean(sum((A_bar - A_annot).^2,1)))
% RMS = 1.3175
%--------------------------------------------------------
%returns the H matrix given point (3D) and pointp(image) (transformed coordinates)
% point is N X 3, pointp is N X 2

end

function P = Calc_P (standard_point, annot_point)
A = [];
b = [];
%Build matrix A and b
for i = 1:size(standard_point,1)
    x_now = standard_point(i,1);
    x_anno_now = annot_point(i,1);
    y_now = standard_point(i,2);
    y_anno_now = annot_point(i,2);
    z_now =standard_point(i, 3);
    %calculates A matrix and B vector
    A = [A;
        x_now y_now z_now 1 0 0 0 0;
        0 0 0 0 x_now y_now z_now 1];
    b = [b; x_anno_now; y_anno_now];
end
%least squares solution
x = (A'*A)^-1 * A'*b;
P = [x(1) x(2) x(3) x(4);
    x(5) x(6) x(7) x(8);
    0 0 0 1];
end
```