EECS 442 Midterm Exam Solution

Yi Yang

11/01/2016

Problem 1

(a)

We need to show that the result is dependent on the order of rotation. First rotate β around y axis, then rotate γ around z axis:

$$p' = R_z(\gamma)R_y(\beta)p = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} p = \begin{bmatrix} \cos\beta\cos\gamma & -\sin\gamma & \cos\gamma\sin\beta \\ \cos\beta\sin\gamma & \cos\gamma & \sin\beta\sin\gamma \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} p$$

If we first rotate γ around z axis and then rotate β around y axis, we can get:

$$p' = R_y(\beta)R_z(\gamma)p = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} p = \begin{bmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta \\ \sin\gamma & \cos\gamma & 0 \\ -\cos\gamma\sin\beta & \sin\beta\sin\gamma & \cos\beta \end{bmatrix} p$$

Hence, we know these two rotations will produce two different values of p'.

(b)

$$R_x(\alpha)R_y(0)R_z(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha+\gamma) & -\sin(\alpha+\gamma) \\ 0 & \sin(\alpha+\gamma) & \cos(\alpha+\gamma) \end{bmatrix}$$

We can easily see that the degree of freedom for this rotation is 1 since we can see parameter $(\alpha+\gamma)$ the whole as a parameter.

Problem 2

First, we need to prove the homographic transformation H defined as $p' = Hp$ has a form as KRK^{-1}. Consider two points p and p' in two images respectively that corresponds to the same 3D point with world coordinates P. Then we have:

$$p = K[I\ T]P, \quad p' = K[R\ T]P$$

$$\begin{aligned} p' &= K[R\ T]P \\ &= KR[I\ T]P \\ &= KRK^{-1}K[I\ T]P \\ &= KRK^{-1}p \end{aligned}$$

Hence, we found that the homographic transformation has a form of $H = KRK^{-1}$.

Computing H

First, let me introduce a brief derivation of DLT algorithm for homographical transformation. For point correspondences located in two images, we have:

$$x'_i = Hx_i \quad x'_i \times Hx_i = 0$$

$$\begin{aligned}
x' \times Hx &= \begin{bmatrix} 0 & -x'_3 & x'_2 \\ x'_3 & 0 & -x'_2 \\ -x'_2 & x'_1 & 0 \end{bmatrix} \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\
&= \begin{bmatrix} -x_1x'_3h_4 - x_2x'_3h_5 - x_3x'_3h_6 + x_1x'_2h_7 + x_2x'_2h_8 + x_3x'_2h_9 \\ x_1x'_3h_1 + x_2x'_3h_2 + x_3x'_3h_3 - x_1x'_1h_7 - x_2x'_1h_8 - x_3x'_1h_9 \\ -x_1x'_2h_1 - x_2x'_2h_2 - x_3x'_2h_3 - x_3x'_2h_3 + x_1x'_1h_4 + x_2x'_1h_5 + x_3x'_1h_6 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & -x_1x'_3 & -x_2x'_3 & -x_3x'_3 & x_1x'_2 & x_2x'_2 & x_3x'_2 \\ x_1x'_3 & x_2x'_3 & x_3x'_3 & 0 & 0 & 0 & -x_1x'_1 & -x_2x'_1 & -x_3x'_1 \\ -x_1x'_2 & -x_2x'_2 & -x_3x'_2 & x_1x'_1 & x_2x'_1 & x_3x'_1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} \\
&= Ah
\end{aligned}$$

Let us observe matrix A, we can see that $x'_2 \times \text{row } 2 + x'_1 \times \text{row } 1 + \text{row } 3 = 0$, we can deduce that rank(A) = 2, which means there exists only 2 independent equations for each point correspondences. In order to solve H matrix or vector h, we must provide 4 pairs of point correspondences since the DOF of H is 8. Then we can apply SVD to matrix A to solve for eigenvector matrix and h is represented as the last eigenvector which corresponds to the smallest singular value of A. The estimated matrix H is:

$$H = \begin{bmatrix} -1.8619 & -0.5207 & 591.3803 \\ -0.8226 & -2.2354 & 441.5196 \\ -0.0044 & -0.0046 & 1.0000 \end{bmatrix}$$

The Matlab codes for computing H is attached below:

```
function [H] = Homography(p1, p2)

% get number of points
n = size(p1,1);
% construct homogeneous x and x'
x = p1;
x_prime = p2;

% construct A
A = zeros(2*n,9);
for i = 1:n
    %construct Ai with xi',yi',wi' and xi
    xi_prime = x_prime(i,1);
    yi_prime = x_prime(i,2);
    wi_prime = x_prime(i,3);
    xi = x(i,:);
    Ai = [  0, 0, 0,        -wi_prime * xi,   yi_prime * xi ;
            wi_prime * xi   0, 0, 0,          -xi_prime * xi];
    A(2*i-1,:) = Ai(1,:);
```

```
    A(2*i,:) = Ai(2,:);
end
%use svd decomposition
[U,D,V] = svd(A);
%H is the last column of V
H = reshape(V(:,end),3,3)';
H = H / H(3,3);

end
```

```
%% EECS 442 - Midterm - Q2 Computing H
%  Date: 2016 / 11 / 02
%  by Yi Yang
%  DLT algorithm estimating Homographical transformation

%% Initialization

clear; close all; clc

%readPoints
[p1, p2] = readPoints('4points.txt');
p1 = [p1, ones(size(p1,1),1)];
p2 = [p2, ones(size(p2,1),1)];

H = Homography(p1, p2)
```

Convolution

The Matlab codes for Gaussian blurring is attached below:

```
%% Gaussian Blurring and Convolution
% Date: 11/01/2016
% by Yi Yang

%% Read an Image
Img = imread('garden1.jpg');
A = imnoise(Img,'Gaussian',0.04,0.003);
%Image with noise
figure,imshow(A);
I = double(A);

%Design the Gaussian Kernel
%Standard Deviation
sigma = 1.76;
%Window size
sz = 4;
[x,y]=meshgrid(-sz:sz,-sz:sz);

M = size(x,1)-1;
N = size(y,1)-1;
Exp_comp = -(x.^2+y.^2)/(2*sigma*sigma);
Kernel= exp(Exp_comp)/(2*pi*sigma*sigma);
%Initialize
Output=zeros(size(I));
%Pad the vector with zeros
I = padarray(I,[sz sz]);

%Convolution
for color = 1:3
    for i = 1:size(I,1)-M
```

```
        for j =1:size(I,2)-N
            Temp = I(i:i+M,j:j+M, color).*Kernel;
            Output(i,j, color)=sum(Temp(:));
        end
    end
end
%Image without Noise after Gaussian blur
Output = uint8(Output);
figure,imshow(Output);
```

The image that is added noise is shown below:

Figure 1: Image with Gaussian White Noise

As shown in figure 1, we add Gaussian white noise with mean 0.04 and variance 0.003 to the original image garden1.jpg. Then we design a 9×9 Gaussian kernel with Standard Deviation $=1.76$ to filter the image and reduce the effects of noise. The filtered image is shown as figure 2.

Figure 2: Gaussian Filtering Results