Zhu, Zhongjian
University of California, San Diego

March 30, 2018

HOMEWORK 2

Problem 1. **Programming: Estimation of the camera projection matrix**

(a) **Linear estimation**
Estimate the camera projection matrix $\mathrm{P}_{\mathrm{DLT}}$ using the direct linear transformation (DLT) algorithm (with data normalization).

Solution
To get the projection matrix, we need to first vectorize $\mathbf{P}$ and then solve the equation below.

$$\begin{bmatrix} [\mathbf{x_1}]^{\perp} \otimes \mathbf{X_1}^{\top} \\ [\mathbf{x_2}]^{\perp} \otimes \mathbf{X_2}^{\top} \\ \vdots \\ [\mathbf{x_n}]^{\perp} \otimes \mathbf{X_n}^{\top} \end{bmatrix} \mathbf{p} = \mathbf{0}$$

where $[\mathbf{x_1}]^{\perp}$ is the left null space of $\mathbf{x_i}$. To get the left null space of $\mathbf{x_i}$, we need the Householder Matrix.

$$\mathbf{H_V} = \mathbf{I} - 2\frac{\mathbf{V}\mathbf{V}^{\top}}{\mathbf{V}^{\top}\mathbf{V}}$$

$$\mathbf{V} = \mathbf{x} + sign(x_1)\|\mathbf{x}\|\mathbf{e_1}$$

The 2 to 3 rows of $\mathbf{H_V}$ is the left null space of $\mathbf{x_i}$.
Since the DoF of $\mathbf{p}$ is 11, we need at least 6 points to solve the equation. Suppose the left null space is $\mathbf{A}$, using Singular Value Decomposition we can get

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}$$

and $\mathbf{p}$ is the last column of $\mathbf{V}$.

Result

$$\mathbf{P_{DLT}} = \begin{bmatrix} -0.0060446 & 0.0048386 & -0.0088225 & -0.8405 \\ -0.0090945 & 0.0023023 & 0.0061782 & -0.54156 \\ -5.0076 \times 10^{-6} & -4.4768 \times 10^{-6} & -2.5529 \times 10^{-6} & -0.0012515 \end{bmatrix}.$$

(b) **Nonlinear estimation**
Use $\mathbf{P_{DLT}}$ as an initial estimate to an iterative estimation method, specifically the Levenberg-Marquardt algorithm, to determine the Maximum Likelihood estimate of the camera projection

matrix that minimizes the projection error.

Solution
The algorithm works as follow:

I $\lambda = 0.001$;
$\epsilon = \widetilde{\mathbf{x}} - \hat{\widetilde{\mathbf{x}}}$

II $\mathbf{J} = \frac{\partial \hat{\widetilde{\mathbf{x}}}}{\partial \hat{\mathbf{p}}}$

III $\mathbf{J}^\top \mathbf{\Sigma^{-1}} \mathbf{J} \delta = \mathbf{J}^\top \mathbf{\Sigma^{-1}} \epsilon$

IV $(\mathbf{J}^\top \mathbf{\Sigma^{-1}} \mathbf{J} + \lambda \mathbf{I}) \delta = \mathbf{J}^\top \mathbf{\Sigma^{-1}} \epsilon$, solve for δ.

V $\hat{\mathbf{p}}_\mathbf{0} = \hat{\mathbf{p}} + \delta$, candidate parameter vector.

VI $\hat{\mathbf{p}}_\mathbf{0} \mapsto \hat{\widetilde{\mathbf{x}}}_\mathbf{0}$;
$\epsilon_0 = \widetilde{\mathbf{x}} - \hat{\widetilde{\mathbf{x}}}_\mathbf{0}$

VII If $\epsilon_\mathbf{0}^\top \mathbf{\Sigma}_{\widetilde{\mathbf{x}}}^{\mathbf{-1}} \epsilon_\mathbf{0}$ cost less than $\epsilon^\top \mathbf{\Sigma}_{\widetilde{\mathbf{x}}}^{\mathbf{-1}} \epsilon$,
$\hat{\mathbf{p}} = \hat{\mathbf{p}}_\mathbf{0}$, $\epsilon = \epsilon_0$, $\lambda = 0.1\lambda$, go to step II or terminate.
Else,
$\lambda = 10\lambda$, go to step IV

Here, the termination condition is $0.0001 \geqslant \mathit{previous\_cost} - \mathit{current\_cost}$.

Result
The costs of every step are

Iteration	Cost
0	84.0826
1	82.7913
2	82.7902
3	82.7902

The final estimate of the camera projection matrix is

$$\mathbf{P_{LM}} = \begin{bmatrix} 0.0060943 & -0.0047265 & 0.0087902 & 0.84364 \\ 0.0090202 & -0.0022929 & -0.0061333 & 0.53666 \\ 4.9909 \times 10^{-6} & 4.4521 \times 10^{-6} & 2.5371 \times 10^{-6} & 0.0012435 \end{bmatrix}.$$

Appendix

```
close all;
clear;
clc;
```

```
%% Load Data
x = load('hw2_points2D.txt');
X = load('hw2_points3D.txt');
n = numel(x);
A = zeros(n,12);
%% Data Normalization
mu_x = mean(x);
sigma_x = sum(var(x));
s_x = sqrt(2/sigma_x);
T = [s_x      0      -mu_x(1)*s_x
     0        s_x    -mu_x(2)*s_x
     0        0      1            ];

mu_X = mean(X);
sigma_X = sum(var(X));
s_X = sqrt(3/sigma_X);
U = [s_X      0      0      -mu_X(1)*s_X
     0        s_X    0      -mu_X(2)*s_X
     0        0      s_X    -mu_X(3)*s_X
     0        0      0      1            ];
x_bar = padarray(x,[0 1],1,'post');
X_bar = padarray(X,[0 1],1,'post');
x_bar_norm = T*x_bar';
X_bar_norm = U*X_bar';
%% P
for i = 1:2:n
    xi = x_bar_norm(:,(i+1)/2);
    Xi = X_bar_norm(:,(i+1)/2);
    v = xi + sign(xi(1))*norm(xi,2)*[1;0;0];
    Hv = eye(3) - 2 * (v*v')/(v'*v);
    m = Hv(2:3,:)';
    A(i:i+1,:) = kron(m',Xi');
end
[~,~,V] = svd(A);
p = V(:,end);
format shortg
P_DLT = T\reshape(p,4,3)'*U;
P_DLT = P_DLT/norm(P_DLT,'fro');
disp('P_DLT = ')
disp(P_DLT)
%% Vecterization
x_norm = reshape(x_bar_norm(1:2,:),[],1);
%% Initialization
% P_init = reshape(p,4,3)';
% p_hat
p_bar = p;
```

```
p_hat = parameterization(p_bar);
% x_hat
x_hat = reshape(p,4,3)'*X_bar_norm;
x_hat = x_hat(1:2,:) ./ x_hat(3,:);
x_hat = reshape(x_hat,[],1);
% sigma
sigma = T(1,1)^2 * eye(n);
% cost
previous_cost = inf;
tolerance = 0.0001;
%% LM
i = 0;
disp('LM:')
fprintf('itr\tcost\n')
fprintf('--------------------\n')
% step_1
lambda = 0.001;
epsilon = x_norm - x_hat;
format short
current_cost = epsilon'*(sigma\epsilon);
fprintf('%d\t%.4f\n', i, current_cost)
% step_2
J = jcb(x_hat,X_bar_norm,p_hat);
while tolerance < previous_cost - current_cost
    i = i + 1;
    % step_3_4
    delta = (J'*(sigma\J)+lambda*eye(11))\(J'*(sigma\epsilon));
    % step_5
    p_hat0 = p_hat + delta;
    % step_6
    x_hat0 = estimate(p_hat0,X_bar_norm);
    epsilon0 = x_norm - x_hat0;
    % step_7
    format short
    previous_cost = epsilon'*(sigma\epsilon);
    current_cost = epsilon0'*(sigma\epsilon0);
    fprintf('%d\t%.4f\n', i, current_cost)
    if current_cost < previous_cost
        p_hat = p_hat0;
        epsilon = epsilon0;
        lambda = 0.1*lambda;
        J = jcb(x_hat0,X_bar_norm,p_hat);
    else
        lambda = 10*lambda;
    end
end
```

```
p_bar_final = deparameterization(p_hat0);
format shortg
P_LM = T\reshape(p_bar_final,4,3)'*U;
P_LM = P_LM/norm(P_LM, 'fro');
fprintf('\nP_LM = \n')
disp(P_LM)
```

```
function J = jcb(x_hat,X_bar_norm,p_hat)
% inputs:
% 1. inhomogeneous 2D points x vector;
% 2. homogeneous 3D points X;
% 3. p_bar(12*1)
p_bar = deparameterization(p_hat);
p_hat_norm = norm(p_hat,2);
r = size(x_hat,1);
c = numel(p_hat);
J = zeros(r,c);
for i = 1:2:r
    w = p_bar(end-3:end)'*X_bar_norm(:,(i+1)/2);
    dx_dpbar = [X_bar_norm(:,(i+1)/2)' zeros(1,4) -x_hat(i)*X_bar_norm(:,(i+1)/2)' % 2*12
                zeros(1,4) X_bar_norm(:,(i+1)/2)' -x_hat(i+1)*X_bar_norm(:,(i+1)/2)']/w;
    da_dphat = -p_bar(2:end)'/2;
    if p_hat_norm == 0
        db_dphat = eye(c)/2;
    else
        db_dphat = sin(p_hat_norm/2)/p_hat_norm*eye(c) + ...
            (p_hat_norm/2*cos(p_hat_norm/2)-sin(p_hat_norm/2))/p_hat_norm^3*(p_hat*p_hat')
    end
    dpbar_dphat = [da_dphat;db_dphat];
    J(i:i+1,:) = dx_dpbar*dpbar_dphat;
end
```

```
function x_hat0 = estimate(p_hat0,X_bar_norm)
p_bar0 = deparameterization(p_hat0);
P_hat0 = reshape(p_bar0,4,3)';

x_bar0 = P_hat0*X_bar_norm;
x_hat0 = x_bar0(1:2,:)./x_bar0(3,:);
x_hat0 = reshape(x_hat0,[],1);
end
```

```
function v = parameterization(v_bar)
a = v_bar(1);
b = v_bar(2:end);
v = 2*acos(a)/sin(acos(a))*b;
```

```
if norm(v,2)>pi
    v = (norm(v,2)-2*pi)*v/norm(v,2);
end
end
```

```
function v_bar = deparameterization(v)
v_norm = norm(v,2);
a = cos(v_norm/2);
b = sin(v_norm/2)/v_norm*v;
v_bar = [a;b];
end
```