15 Febbraio 2012

VISIONE COMPUTAZIONALE
-LABORATORIO-

Cristina Segalin
ID353053

Esercizio 1.
Generare 100 punti casuali nello spazio tridimensionale e una telecamera. Proiettare i punti 3d sul piano immagine della telecamera. Effettuare la calibrazione della telecamere dai punti 3d e le loro proiezioni. Dare una misura dell'errore nella calibrazione così ottenuta rispetto alla telecamera reale.

Le funzioni di supporto usate per questo esercizio sono:
-camera.m
-proj.m
-resect.m

CAMERA.M

```
%camera(pos,lookpoint, up, k)
%pos position
%lookpoint point on z axe
%up for vertical direction
%k intrinsic parameters
%r1 is y axe on camera reference
%world reference r2=y e r3=z
%centre expression C=-Q^-1*q
%[kR|kt] --> C=(KR)^-1*kt =R^tk^-1kt
function [P,R]=camera(pos,lookpoint, up, K)
    z=lookpoint-pos; %look direction vector
    z=z/norm(z); %normalize to get versor
    x=cross(up, z); %get a vector orthogonal to both z and up
    x=x/norm(x,2); %normalize to get versor
    y=cross(z,x); %get a vector orthogonal to both z and x axis
    y=y/norm(z,2); %normalize to get versor
    R=[x'; y'; z']; %Rotation matrix
    assert(all(all(abs(R'*R)-eye(3))<10^(-10)))
    t=(-R)*pos;
    P=K*[R t]; %camera Projection Matrix

    % camera([1 1 1]',[0 0 5]',[0 1 0]',[1 0 1; 0 1 1; 0 0 1]
    R=P/norm(P) %normalize
end
```

PROJ.M

```
%Project 3d points 3d (M) to 2d (m) using projection matrix P

 function m=proj(P,M)
    M_om=[M; ones(1, size(M,2))];%put 3d points in omogeneous coordinates
    m_om=P*M_om; %apply projection
    m=[m_om(1,:)./m_om(3,:);m_om(2,:)./m_om(3,:)]; %cartesian coords
end
```

RESECT.M

```
% Returns the projection matrix P that gives the transformation between M and m
using DLT and kronecher product

function P=resect(M, m)
%transform into omogeneous coordinates
M=[M; ones(1,size(M, 2))];
m=[m; ones(1,size(m, 2))];
%build an L matrix for the linear system L*vec(P')=0
L=[];
for i=1:size(M, 2)
    L=[L; kron(M(:, i)', star(m(:, i)))];  %cross product is to avoid scaling
problems
end
```

```
%the solution of the system is the nullspace of L,
%given by the last column of the V matrix in the SVD decomposition of L
[U D V]=svd(L);
P=reshape(V(:,end), 3, 4); %reshape to get 3x4 matrix

%for testing i can normalize or make vec mtrix of both P e get the rank
P1=P./norm(P);
end
```

ES1.M

```
close all;
clear all;

%Generate 100 3d points, and camera points;

M=rand(3,100); %100 random 3d points
%generate the camera
pos=[1 1 1]'; %camera Center of Projection
lookpoint=[0 0 5]';%camera lookpoint
up=[0 1 0]';%camera up vector (doesn't need to be orthogonal to lookppoint)
k=eye(3); %intrinsic parameters of the camera
[P R]=camera(pos,lookpoint,up,k); %geerate the camera

%project point on camera image plane

m=proj(P,M);%project 3d points in the camera view

figure; %draw camera view
scatter(m(1,:), m(2,:));

%make camera resect using 3d fixpoints and 2d projected points
Pn=resect(M,m)

%misure the errore between the result camera and the real camera
%error is given by the norm of the difference between P and Pn
sf=P(1,1)/Pn(1,1); %scaling factor
Pn=Pn*sf;
norm(P-Pn, 2)

% check that 2 camera are the same
%assert(all(all(abs(P1-P)<10^-10)));
%vec P' and Pn
Pt=P';
vecP=Pt(:);
Pnt=Pn';
vecPn=Pnt(:);
%unico le due matrice di priezione
Punione=[vecP';vecPn'];
rank(Punione)
```

ESERCIZIO 2

Generare 100 punti casuali nello spazio tridimensionale e n telecamere. Proiettare i punti sui piani immagine delle telecamere. Effettuare la triangolazione. Dare una misura dell'errore. Mostrare i punti nello spazio 3D insieme alla posizione e orientazione delle telecamere.

Le funzioni di supporto che serviranno per questo esercizio sono:
-camera.m
-proj.m
-resect.m
-intersect.m

INTERSECT.M

```
function M= intersect(m,P)
%takes 2 set of corresponding points, camera and returns 3d point
%P=3x4xn P(:,:,j)
%m=2x1xn

for j=1:size(m,2)
    L=[];
    for i=1:size(P,3)
        l1=P(1,:,i);
        l2=P(2,:,i);
        l3=P(3,:,i);
        L=[L;l1-m(1,j,i).*l3;l2-m(2,j,i).*l3];
    end
     [U(:,:,j) D(:,:,j) V(:,:,j)]=svd(L);
end
    M_om=squeeze(V(:,end,:));
    M=[M_om(1,:)./M_om(4,:); M_om(2,:)./M_om(4,:);M_om(3,:)./M_om(4,:)]
end
```

ES2.M

```
clear all
close all

%% Generate 100 3d points and a set of cameras that look at them.
% 3d points generation
M=randn(3, 100);

% 1st cam
K=eye(3);
lookp1=[0;0;0];
up1=[0, 1, 0]';
pos1=[-3 0 -5]';
[P1 R1]=camera(pos1,lookp1, up1,K);

% 2nd cam
lookp2=[0;0;0];
up2=[0, 1, 0]';
pos2=[-1 0 -5]';
[P2 R2]=camera(pos2, lookp2, up2 ,K);

% 3rd cam
lookp3=[0;0;0];
up3=[0, 1, 0]';
pos3=[1 0 -5]';
[P3 R3]=camera(pos3, lookp3, up3, K);


%%Projections of the 3d M on the image planes of the cameras
```

```
m(:,:,1)=proj(P1, M);
m(:,:,2)=proj(P2, M);
m(:,:,3)=proj(P3, M);

%Plotting the views
figure;
subplot(2, 2, 1);scatter(m(1,:,1), m(2,:,1));
subplot(2, 2, 2);scatter(m(1,:,2), m(2,:,2));
subplot(2, 2, 3);scatter(m(1,:,3), m(2,:,3));

%%make triangulation with cameras and fixpoints

%Generating a Matrix containing all the cameras
PPM(:,:,1)=P1;
PPM(:,:,2)=P2;
PPM(:,:,3)=P3;

%Triangulation
M1=intersect(m,PPM);

%% give a misure of error
norm(M1-M,'fro')

%%chek
M1t=M1';
vecM1=M1t(:);
Mt=M';
vecM=Mt(:);

Munion=[vecM';vecM1'];
rank(Munion)
```

ESERCIZIO 3.

Generare 100 punti nello spazio tridimensionale. Proiettare i punti sui piani immagine di 2 diverse telecamere. Applicare l'algoritmo Structure And Motion ai punti così ottenuti per ricostruire la posizione e l'orientazione delle telecamere.

Le funzioni di supporto che serviranno per questo esercizio sono:
-camera.m
-proj.m
-intersect.m
-fund.m
-sr.m

FUND.M

```
%% Generate the essential matrix E from the 2 given sets of 2d points in
%% normalized coordinates

function F = fund(m1,m2)
%% m1: first set of points, m2: second set of points
m1_omo=[m1(:,:);ones(1,size(m1,2))];
m2_omo=[m2(:,:);ones(1,size(m2,2))];
T=[];
for i=1:size(m1,2)
    m1t=m1_omo(:,i);
    m2t=m2_omo(:,i);
    T=[T;  kron(m1t',m2t')];
end
    [U D V]=svd(T);
    F = reshape(V(:,end),3,3);
    %E1 is now probably affected by errors, sig1!= sig2, sig3!=0

    %% generating the Essential matrix E most similar in Frobenius norm to E1
    [U D1 V]=svd(E1);
    D=diag([(D1(1,1)+D1(2,2))/2, (D1(1,1)+D1(2,2))/2, 0]);
    E=U*D*V';
end
```

SR.M

```
%% Factorize an essential matrix into an Antisimmetric matrix S and a rotation
%% matrix R

function [R t]=sr(E,m1,m2)
    [U V D]=svd(E);
    S1=[0 -1 0; 1 0 0; 0 0 0];
    R1=[0 1 0; -1 0 0; 0 0 1];

    t_mat=U*S1*U'
    R=det(U*V')*U*R1'*V';
    t=[t_mat(2,3);t_mat(1,3);-t_mat(1,2)];
end
```

ES3.M

```
% E = fund(m1,m2); 8 points algorihtm
%triangolulation
M=rand(3,100);

% camera parameters
c=[0 0 0]';
lookup=[5 3 2]';
up = [6 1 2]';
```

```
K = [3 0 0; 0 2 0; 0 0 1];
% generate camera matrix
P1 = camera(c,lookup,up,K);
m1=proj(P1,M);

c1=[0 0 4]';
lookup1=[0 4 2]';
up1 = [0 0 4]';
K1 = [1 0 0; 0 1 0; 0 0 1];
% generate camera matrix
P2 = camera(c1,lookup1,up1,K1);
m2=proj(P2,M);

%get essential matrix
F1=fund(m1,m2);
%get essential matrix for check
F2=fm(m1,m2);

F1t=F1';
vecF1=F1t(:);
F2t=F2';
vecF2=F2t(:);

Funion=[vecF1';vecF2'];
%check the correctness
rank(Funion)

%get
E=K*F1*K1';
%get sr factorization
[R t]=sr(E,m1,m2);
```

ESERCIZIO4.
Utilizzare l'algoritmo Structure and Motion date 5 immagini dello stesso soggetto prese da 5 telecamere diverse e ad intrinseci noti, di cui sono fornite le corrispondenze di alcuni punti notevoli.

Le funzioni di supporto che serviranno per questo esercizio sono:
-camera.m
-fund.m
-sr.m
-intersect.m
-proj.m

ES4.M

```
clear all;
close all;

%% Usare l'algoritmo Structure And Motion per ricavare la posizione e
%% l'orientazione delle telecamere e la struttura 3d della scena date le
%% viste, le corrispondenze e gli intrinseci della telecamera

%caricamento dati e immagini

%matrice degli intrinseci
K= [1.3415194e+03   0.0000000e+00   3.8274742e+02;...
   0.0000000e+00   1.3431237e+03   5.2147988e+02;...
   0.0000000e+00   0.0000000e+00   1.0000000e+00];

%Viste del Chiostro
im(:,:,1)=rgb2gray(imread('chiostro1.jpg'));
im(:,:,2)=rgb2gray(imread('chiostro2.jpg'));
im(:,:,3)=rgb2gray(imread('chiostro3.jpg'));
im(:,:,4)=rgb2gray(imread('chiostro4.jpg'));
im(:,:,5)=rgb2gray(imread('chiostro5.jpg'));

%punti e corrispondenze estratti dalle viste
load chiostro3D

%Sostituisco i NaN con degli 0 (SVD non li accetta)
m1(isnan(m1))=0;
m2(isnan(m2))=0;

%Mostro le 5 viste e i punti estratti per ognuna di esse
figure;
for i=1:5
    subplot(1, 5, i);
    imshow(im(:,:,i));
    hold on;
    scatter(m1(:,i), m2(:,i));
    hold off;
end

%Assumo il SdR mondo con l'origine sulla prima telecamera
PPM(:,:,1)=K*[eye(3),[0,0,0]'];
m=[m1(:,1)';m2(:,1)'; ones(1, size(m1, 1))];
tmp=K\m;
m_norm(:,:,1)=[tmp(1,:)./tmp(3,:);tmp(2,:)./tmp(3,:)];
p2d(:,:,1)=m;
CoP(:,1)=[0,0,0]';

%Calcolo la seconda telecamera e normalizzo le traslazioni imponendo
%||t12||=1
```

```
m=[m1(:,2)';m2(:,2)'; ones(1, size(m1, 1))];
tmp=K\m;
m_norm(:,:,2)=[tmp(1,:)./tmp(3,:);tmp(2,:)./tmp(3,:)];
E12=fund(m_norm(:,:,1), m_norm(:,:,2));
[S12 R12]=sr(E12);
test_point(:,:,1)=m_norm(:,1);
test_point(:,:,2)=m_norm(:,2);
[S12 R12]=check_SR(S12, R12, test_point);
t12=rev_star(S12);
t12=t12./norm(t12);
PPM(:,:,2)=K*[R12,t12];
CoP(:,2)=t12;

%Calcolo le altre 3 telecamere...
for i=3:5
    m=[m1(:,i)';m2(:,i)'; ones(1, size(m1, 1))];
    tmp=K\m;
    m_norm(:,:,i)=[tmp(1,:)./tmp(3,:);tmp(2,:)./tmp(3,:)];

    E1i=fund(m_norm(:,:,1), m_norm(:,:,i));
    [S1i R1i]=sr(E1i);
    test_point(:,:,1)=m_norm(:,1);
    test_point(:,:,2)=m_norm(:,i);
    [S1i R1i]=check_SR(S1i, R1i, test_point);
    t1i=rev_star(S1i);
    t1i=t1i./norm(t1i);

    E2i=fund(m_norm(:,:,2), m_norm(:,:,i));
    [S2i R2i]=sr(E2i);
    test_point(:,:,1)=m_norm(:,2);
    test_point(:,:,2)=m_norm(:,i);
    [S2i R2i]=check_SR(S2i, R2i, test_point);
    t2i=rev_star(S2i);
    t2i=t2i./norm(t2i);

    %...mantenendo la scala delle traslazioni
    t1i=t1i./(((cross(t1i, t2i))'*(cross(R2i*t12,
t2i)))/norm(cross(R2i*t12,t2i)));

    PPM(:,:,i)=K*[R1i,t1i];
    CoP(:,i)=t1i;
    p2d(:,:,i)=m;
end
%Effettuo la triangolazione
p3d=intersect(PPM, p2d);
```