Mingxuan Wang

University of California, San Diego

December 4, 2014

ASSIGNMENT 2

Problem 1. **Epipolar Geometry Theory [2 pts]**

Answers: Assume P1 is right side points

1.

$$R_1 P^W + T_1 = P_1^{C1}$$
$$R_2 P^W + T_2 = P_2^{C2}$$

So we have:

$$R_2 R_1^{-1} P_1^{C1} + T_2 - R_2 R_1^{-1} T_1 = P_2^{C2}$$

2. The length of the baseline of the stereo pair is :

$$b = \|T_2 - R_2 R_1^{-1} T_1\|$$

3.The Essential Matrix is :

$$E = [(T_2 - R_2 R_1^{-1} T_1)_x] R_2 R_1^{-1}$$

Problem 2. **Epipolar Geometry [3 pts]**

Answers:

a).We can easily know that two image points is world coordinate is (-9,6,1) and (16,6,1). So according to similar triangular:

$$\frac{x+15}{6} = \frac{z}{1}$$
$$\frac{x-15}{1} = \frac{z}{1}$$

So we have: $x = 21$ and $z = 6$.

Similarly, we can get $y = 36$.

So the 3D location of this point is (21,36,6).

b).Assume the world coordinate of the point is (xc,0,-xc).

According to similar triangular:

$$\frac{xc+15}{x} = \frac{-xc}{1}$$
$$\frac{-xc+15}{-u} = \frac{-xc}{1}$$

So we have $d = x - u = -2(1+u)$.

Problem 3. **NCC [2pts]**

Answers:

Assume $\widetilde{\omega}_1 = W_1(:)$ and $\widetilde{\omega}_2 = W_2(:)$.

We have:

$$\widetilde{\omega} = \frac{\omega - \overline{\omega}}{\sqrt{(\omega - \overline{\omega})^T(\omega - \overline{\omega})}}$$

With this, we can easily know that:

$$\widetilde{\omega}_1^T \widetilde{\omega}_1 = \widetilde{\omega}_2^T \widetilde{\omega}_2 = 1$$

$$C_{NSSD} = \sum_{i,j} |\widetilde{W}_1(i,j) - \widetilde{W}_2(i,j)| = (\widetilde{\omega}_1 - \widetilde{\omega}_2)^T(\widetilde{\omega}_1 - \widetilde{\omega}_2) = \widetilde{\omega}_1^T\widetilde{\omega}_1 - 2\widetilde{\omega}_1^T\widetilde{\omega}_2 + \widetilde{\omega}_2^T\widetilde{\omega}_2 = 2 - 2\widetilde{\omega}_1^T\widetilde{\omega}_2$$

$$C_{NCC} = \sum_{i,j} \widetilde{W}_1(i,j)\widetilde{W}_2(i,j) = \widetilde{\omega}_1^T\widetilde{\omega}_2 \tag{1}$$

Since the derivation of CNSSD is equal to negative of derivation of CNCC. So that maximizing the Normalized Cross Correlation(NCC) is is equivalent to minimizing the Normalized Sum Squared Distance(NSSD).

Problem 4. **Where's Waldo? [10 points]**

Answers:

(a) **4.1 Warmup**

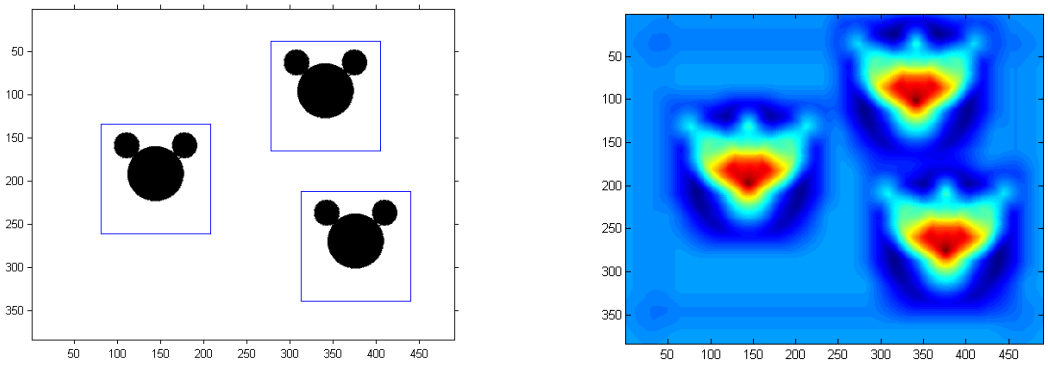
I think this algorithm may work for some realistic images for some situation. But there is bigger possibility that it will fail. Here's what i am considering:

(1) First, the result depends on the filter we choose. The filter can be judged good or not by the

scale, features and many other facts. The filter may be totally different from the realistic images we want to apply this algorithm to. In this case, we may fail.

(2) Second, the result depends on the preprocess we apply to the realistic images. There may be lots of noise in the image. We must come up with some great methods to minimize the influence of the noise otherwise we may fail.

The output is figure 1.



a b

Figure 1: (a) bounding box, (b) heat map

(b) **4.2 Detection Error**

Note: I choose 3 different ground true bounding box—1 exactly the true fit. 1 too big. and 1 too small. The result is figure 2-4.

We can see that some of the image's overlap rate is over 50% but some not.

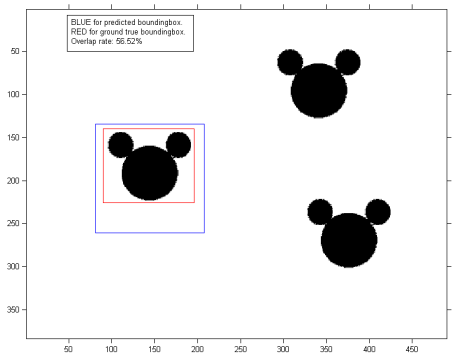


Figure 2: bounding box 1, overlap rate:0.5625

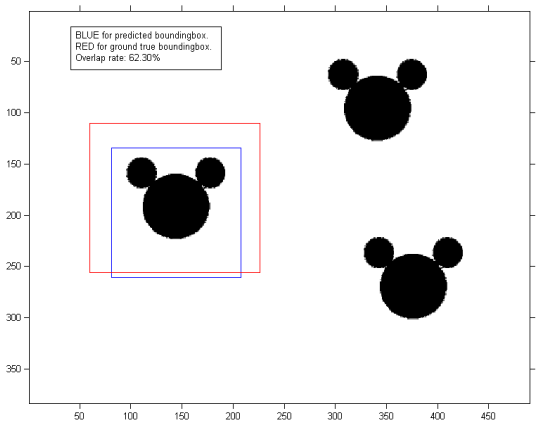


Figure 3: bounding box 2, overlap rate:0.6230

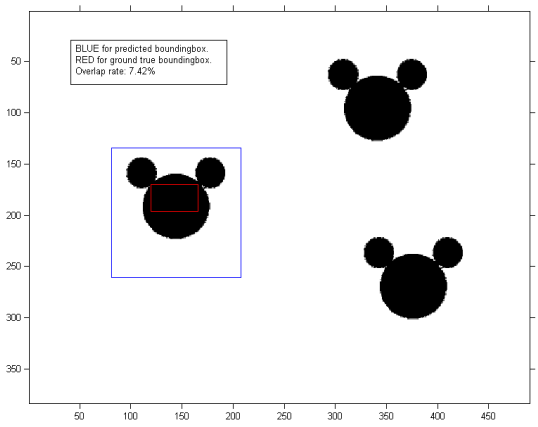


Figure 4: bounding box 3, overlap rate:0.0741

(c) **4.3 More Realistic Images**

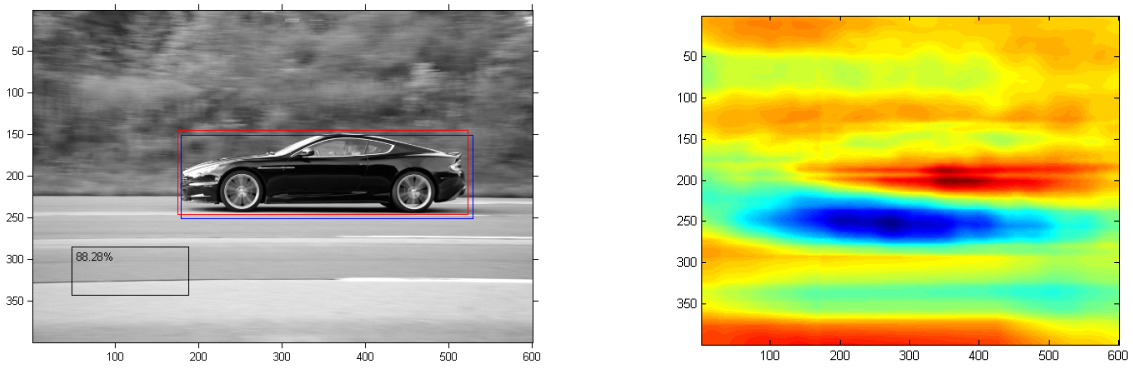
(1)For car1 image:

a). Pre-processing step:

Crop the filter to keep the middle part of the filter and discard the blank around the real car. Resize the filter to the size near the bounding-box. This two steps are aimed to make the size of the filter more close to the real size of the car in the image. Then I flip the filter upsidedown bacause of the convolution. Then substract the mean from the filter and the car image to normalize. At last I do median filtering to both to make the it blur while not to blur the edge. In this case I hope to blur the background and highlight the edge.

The result is 88.28% overlaprate. It turns out to be good job.

b). Result:



a b

Figure 5: (a) car1 bounding box(overlaprate:88.28%), (b) car1 heat map

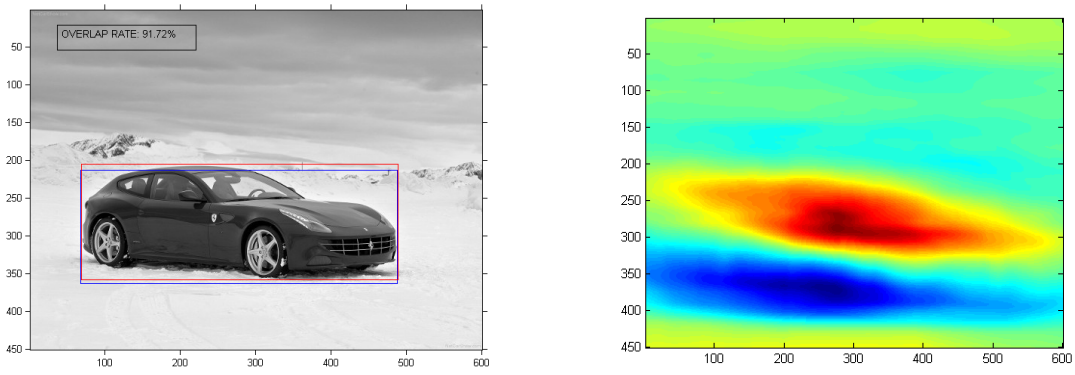
(2)For car2 image:

a). Pre-processing step:

Crop the filter to keep the middle part of the filter and discard the blank around the real car. Resize the filter to the size near the bounding-box. This two steps are aimed to make the size of the filter more close to the real size of the car in the image. Then transfer gray scale image to black and white, which will highlight the black feature of the car from the white background. Then flip the filter because the car in the car image is more like to the flip version of the filter. Then I flip the filter upsidedown bacause of the convolution. At last substract the mean from the filter and the car image to normalize.

The result is 91.72% overlaprate. It turns out to be good job.

b). Result:



a b

Figure 6: (a) car2 bounding box(overlaprate:91.72%), (b) car2 heat map

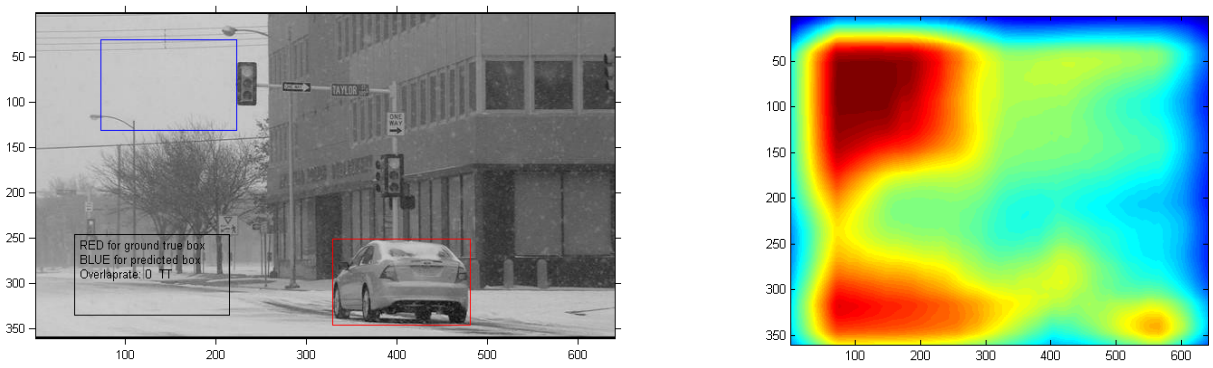
(3)For car3 image:

a). Pre-processing step:

Crop the filter to keep the middle part of the filter and discard the blank around the real car. Resize the filter to the size near the bounding-box. This two steps are aimed to make the size of the filter more close to the real size of the car in the image.Then substract the mean from the filter and the car image to normalize. Then I do gaussian filter to blur the car image and filter. Then I flip the filter upsidedown bacause of the convolution. At last, I use 0.45-filter to compute the final filter. I do this because I notice that the real car in the car image is silver gray while the car in the filter is black. So I transfer the color of the filter to nearly silver gray to match the car in the car image.

The result is not good because somewhere in the sky the feature(i.e. color or brightness) is more like the filter than the real car in the car image. This turns out that color or brightness is not a good feature to do matching in this situation.

b). Result:



a b

Figure 7: (a) car3 bounding box(overlaprate:0%), (b) car3 heat map

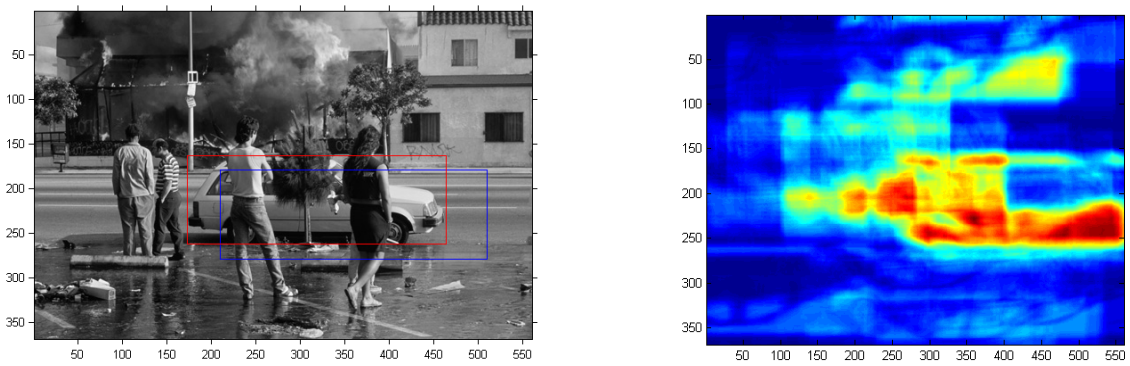
(4)For car4 image:

a). Pre-processing step:

Crop the filter to keep the middle part of the filter and discard the blank around the real car. Resize the filter to the size near the bounding-box. This two steps are aimed to make the size of the filter more close to the real size of the car in the image.Then substract the mean from the filter and the car image to normalize. Then transfer gray scale image to black and white, which will be helpful to easily distinguish the feature of the car. Note that threshhold of im2bw function is tested and set. Then flip the filter because the car in the car image is more like to the flip version of the filter. Then I flip the filter upsidedown bacause of the convolution.

The result is not good enough (56.48%)because the real car in the car image is blocked by some people and other things so that convolution cannot find the features that the filter matches the car image. Also, there are many similar block of pixels on the background which matches better than the car.

b). Result:



a b

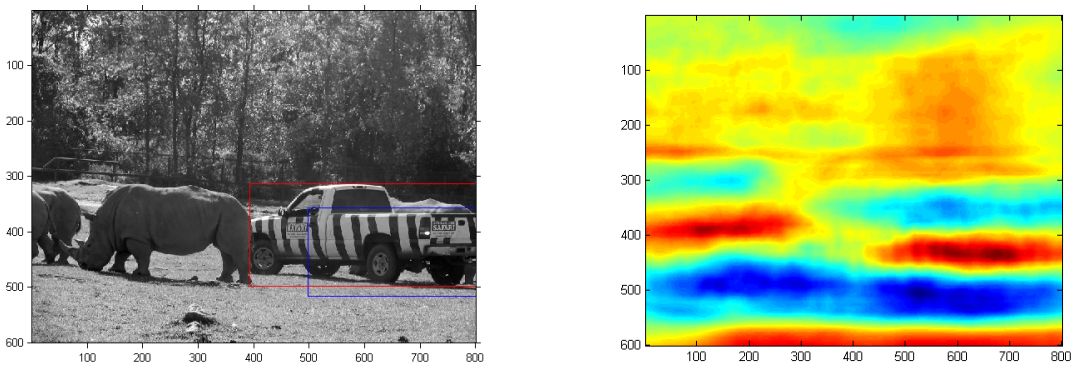
Figure 8: (a) car4 bounding box(overlaprate:56.48%), (b) car4 heat map

(5)For car5 image:

a). Pre-processing step:

Crop the filter to keep the middle part of the filter and discard the blank around the real car. Resize the filter to the size near the bounding-box. This two steps are aimed to make the size of the filter more close to the real size of the car in the image.Then substract the mean from the filter and the car image to normalize. Then flip the filter because the car in the car image is more like to the flip version of the filter. Then I flip the filter upsidedown bacause of the convolution. The result is not good enough (47.56%)because the real car in the car image is quite different from that of filter including the shape, color and other features. It is hard to distinguish a feature of the car from background while this feature is held by the filter at the same time.

b). Result:



a b

Figure 9: (a) car5 bounding box(overlaprate:47.56%), (b) car5 heat map

(d) **4.4 Invariance**

The algorithm used in this problem is not good enough in many cases. It is not invariant to color(in RGB images) or brightness(in gray scale images), which means one filter images with fixed brightness in every pixel will not work on everything. Similarly, this algorithm is not invariant to the direction of the filter, which means we may want to flip or rotate the filter in some cases.

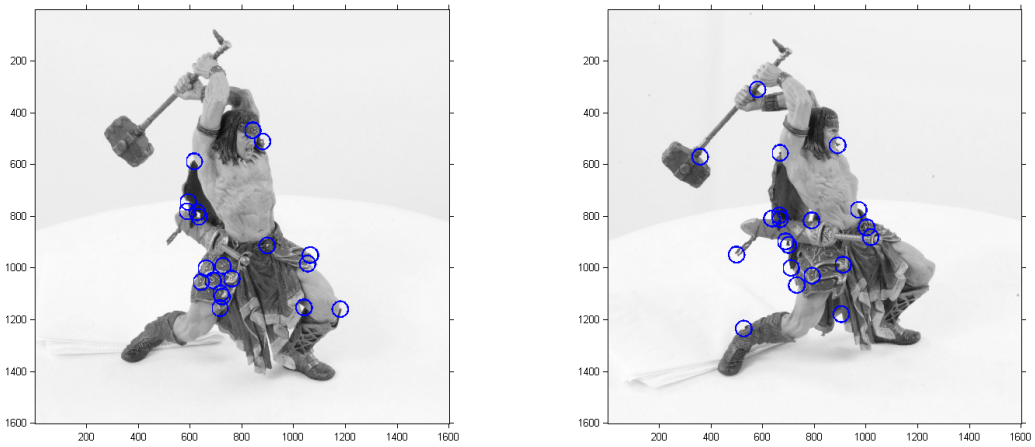
Problem 5. **Sparse Stereo Matching [10pts]**

For image warrior:

(a) **5.1 Corner Detection [2pts]**

Different Non-maximum suppression window sizes may lead to different results. Bigger size of window may lead to non-enough corners detected, so I discard these situations and only include 2 situation here:

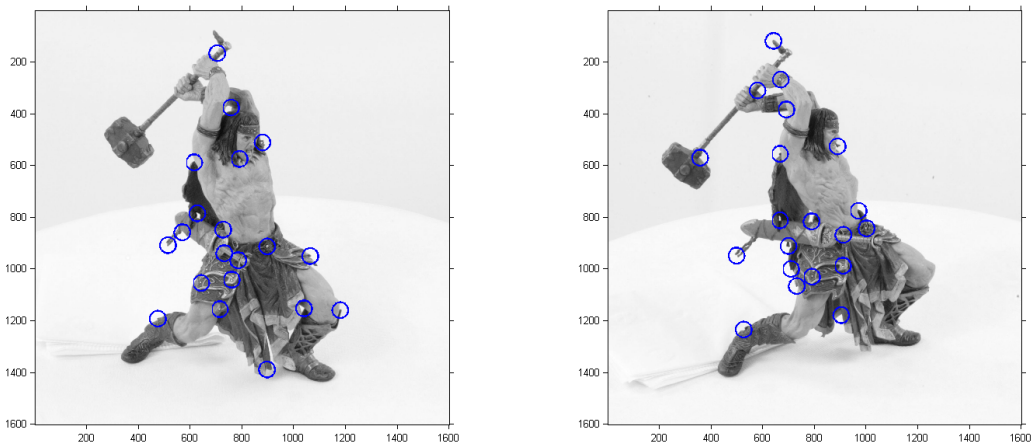
Non-maximum suppression window size is equal to 10:



a b

Figure 10: (a)Corner detection I1, (b)Corner detection I2

Non-maximum suppression window size is equal to 50:



a b

Figure 11: (a)Corner detection I1, (b)Corner detection I2

(b) **5.2 SSD matching[1pts]**

See the code SSDmatch.m in appendix.

(c) **5.3 Naive matching [2pts]**

The result is as below(Non-maximum suppression window:25, matching window R:5, SSDth:5):

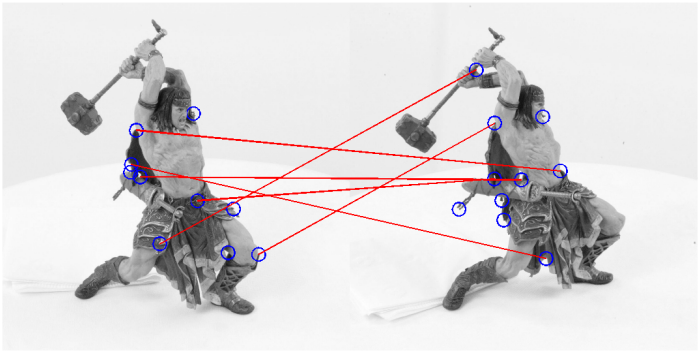
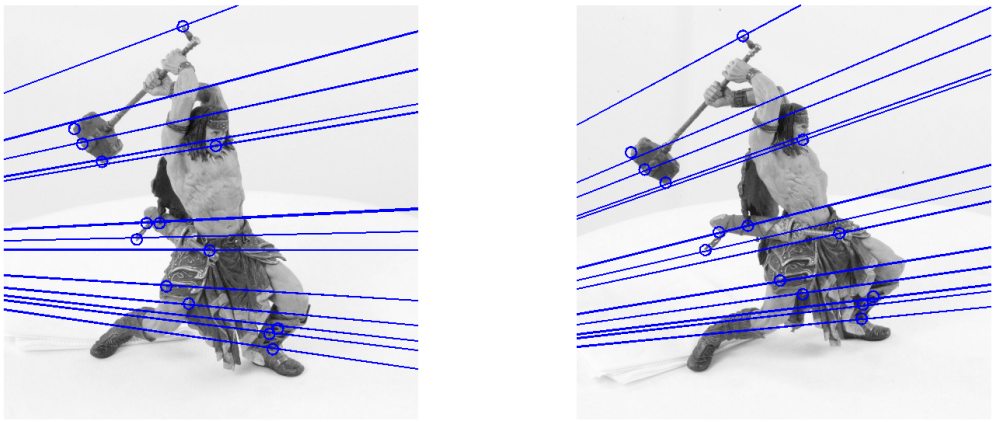


Figure 12: Naive matching result

(d) **5.4 Epipolar geometry [1pts]**

The result is as below:



a b

Figure 13: (a)epipolar line in I1 from corners of I2, (b)epipolar line in I2 from corners of I1

Note that there are few lines hasn't passed through exact middle of the circle. I think it is because the little error of cor1 and cor2 correspondance.

(e) **5.5 Epipolar geometry based matching [2pts]**

The result is as below(Non-maximum suppression window:25, matching window R:3, SSDth:1):

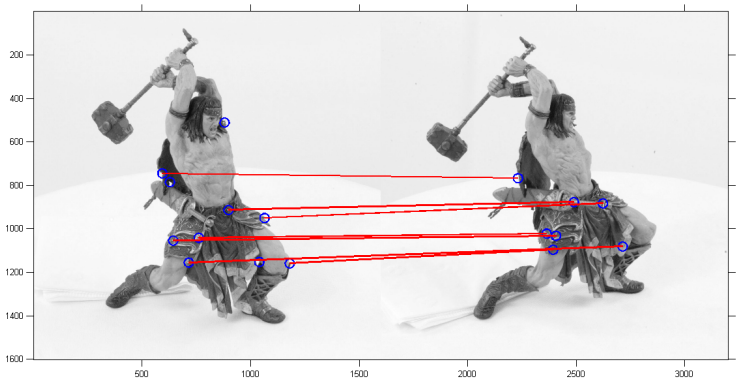


Figure 14: Epipolar geometry based matching result

(f) **5.6 Triangulation [2pts]**

The result is as below(Non-maximum suppression window:25, matching window R:5, SSDth:5):

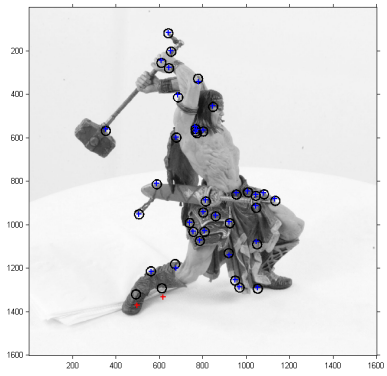


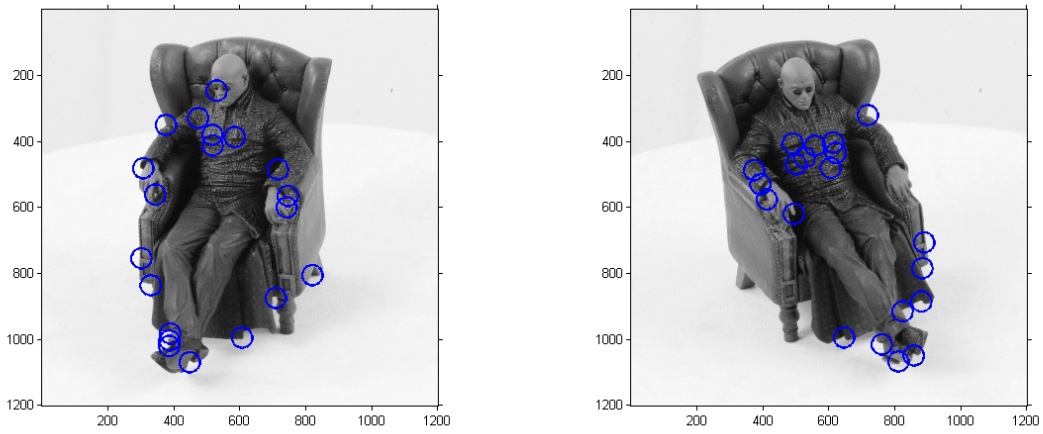
Figure 15: Epipolar geometry based matching result

For image matrix:

(a) **5.1 Corner Detection [2pts]**

Different Non-maximum suppression window sizes may lead to different results. Bigger size of window may lead to non-enough corners detected, so I discard these situations and only include 2 situation here:

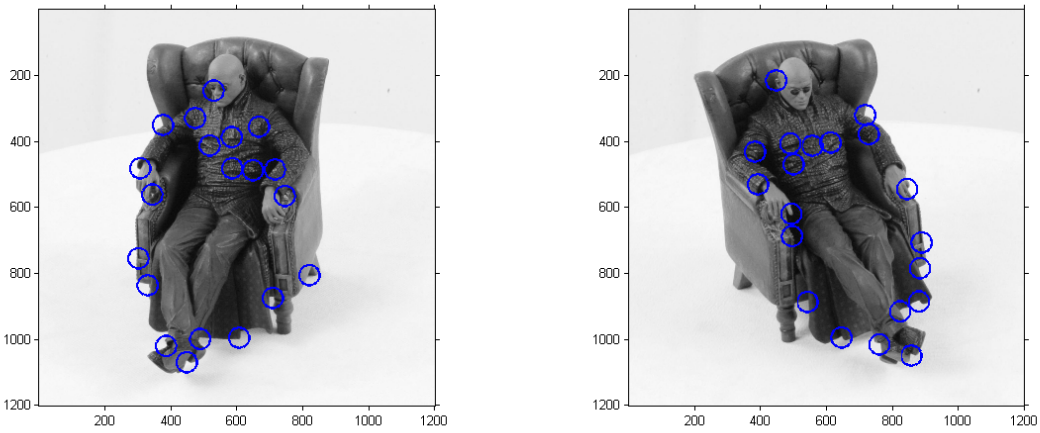
Non-maximum suppression window size is equal to 10:



a b

Figure 16: (a)Corner detection I1, (b)Corner detection I2

Non-maximum suppression window size is equal to 50:



a b

Figure 17: (a)Corner detection I1, (b)Corner detection I2

(b) **5.2 SSD matching[1pts]**

See the code SSDmatch.m in appendix.

(c) **5.3 Naive matching [2pts]**

The result is as below(Non-maximum suppression window:25, matching window R:5, SSDth:5):

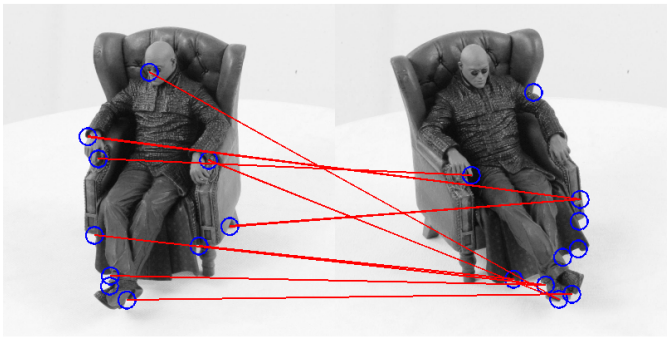
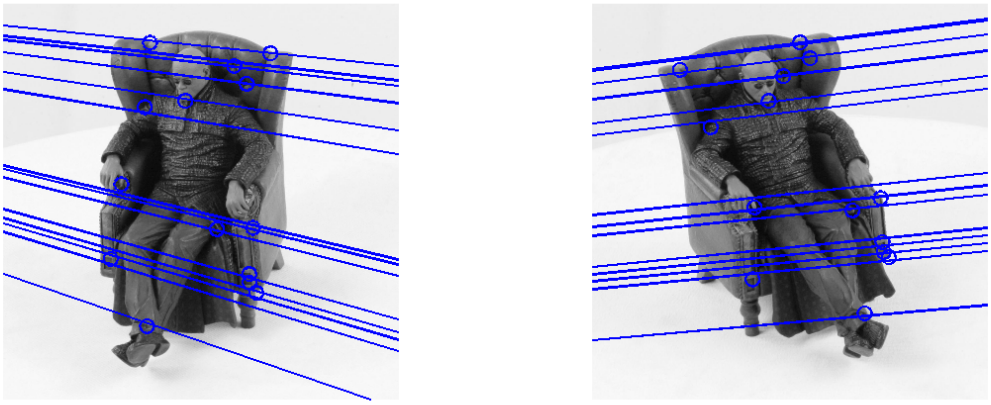


Figure 18: Naive matching result

(d) **5.4 Epipolar geometry [1pts]**

The result is as below:



a b

Figure 19: (a)epipolar line in I1 from corners of I2, (b)epipolar line in I2 from corners of I1

Note that there are few lines hasn't passed through exact middle of the circle. I think it is because the little error of cor1 and cor2 correspondance.

(e) **5.5 Epipolar geometry based matching [2pts]**

The result is as below(Non-maximum suppression window:25, matching window R:5, SSDth:2.5):

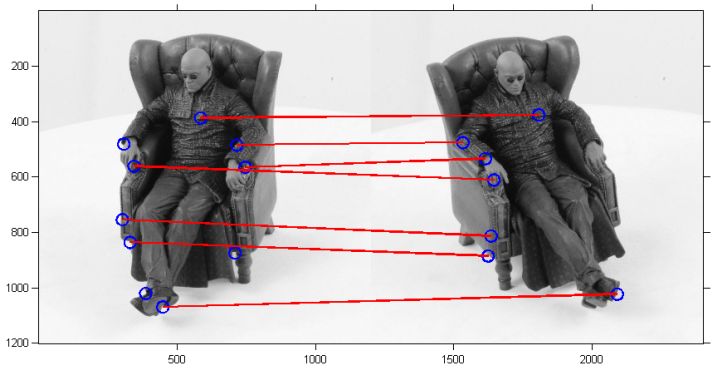


Figure 20: Epipolar geometry based matching result

(f) **5.6 Triangulation [2pts]**

The result is as below(Non-maximum suppression window:10, matching window R:5, SSDth:5):

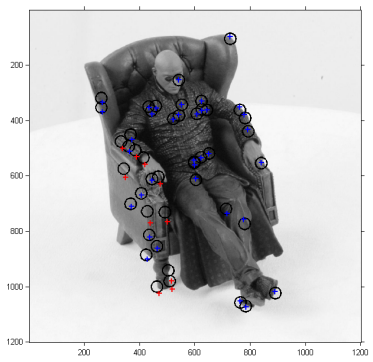


Figure 21: Epipolar geometry based matching result

Listing 1: Codes of hw3 part4.1-script

```
clear;clc;
toy_img = im2double(imread('F:\252\example.png'));
filter_img = im2double(imread('F:\252\filter.jpg'));
[ltoy,wtoy] = size(toy_img);[lfil,wfil] = size(filter_img);

filter_img_star = filter_img - mean(filter_img(:));
toy_img_star = toy_img - mean(toy_img(:));
C = conv2(toy_img_star, filter_img_star, 'same');

colormap('default');
figure(1);
imagesc(C);

max1=max(C);
max2=max(max1);
[row, col] = find(C==max2);
x1=col(1)-(wfil-1)/2; y1=row(1)-(lfil-1)/2;
x2=col(2)-(wfil-1)/2; y2=row(2)-(lfil-1)/2;
x3=col(3)-(wfil-1)/2; y3=row(3)-(lfil-1)/2;

x1min=x1;x1max=x1+wfil-1;y1min=y1;y1max=y1+lfil-1;
x2min=120;x2max=165;y2min=170;y2max=195;

figure(2);
imshow(toy_img);
axis on;
hold on;
rectangle('Position',[x1,y1,wfil,lfil],'edgecolor','b');
rectangle('Position',[x2,y2,wfil,lfil],'edgecolor','b');
rectangle('Position',[x3,y3,wfil,lfil],'edgecolor','b');
rectangle('Position',[x2min,y2min,x2max-x2min+1,y2max-y2min+1],'edgecolor','r');

overlaprate = OverlapFunc(x1min,y1min,x1max,y1max,x2min,y2min,x2max,y2max);
```

Listing 2: Codes of hw3 part4.2-function-OverlapFunc

```
function [ overlap ] = OverlapFunc(x1min,y1min,x1max,y1max,x2min,y2min,x2max,y2max)
w1 = (x1max-x1min+1); h1 = (y1max-y1min+1);
w2 = (x2max-x2min+1); h2 = (y2max-y2min+1);
x1c = (x1min+x1max)/2; y1c = (y1min+y1max)/2;
x2c = (x2min+x2max)/2; y2c = (y2min+y2max)/2;
xv1 = [x1min x1max x1max x1min]; yv1 = [y1min y1min y1max y1max];
xv2 = [x2min x2max x2max x2min]; yv2 = [y2min y2min y2max y2max];

xmin = round(min(x1min,x2min)); xmax = round(max(x1max,x2max));
ymin = round(min(y1min,y2min)); ymax = round(max(y1max,y2max));

x = repmat([xmin:xmax], ymax-ymin+1, 1);
x = x(:);
y = repmat([ymin:ymax]', 1, xmax-xmin+1);
y = y(:);


in1 = inpolygon(x, y, xv1, yv1);
in2 = inpolygon(x, y, xv2, yv2);
in = in1 + in2;
op = sum(in==2); tp = sum(in>0);
overlap = (op)/(tp);
end
```

Listing 3: Codes of hw3 part4.3-script

```
clear;clc;
%% preprocessing---car1
a=175;b=145;c=522; d=245;
car_img = im2double(imread('F:\252\car1.jpg'));
filter_img_ori = im2double(imread('F:\252\cartemplate.jpg'));
filter_img = imresize(filter_img_ori,[d-b+30 c-a+30]);
filter_img_star1 = imresize(imcrop(filter_img,[25,55,c-a-10,d-b-40]),[100 350]);
[ltoy,wtoy] = size(car_img);[lfil,wfil] = size(filter_img_star1);
filter_img_star1 = filter_img_star1 - mean(filter_img_star1(:));
car_img_star1 = car_img - mean(car_img(:));
%median filter
filter_img_star1 = medfilt2(filter_img_star1,[5,5]);
car_img_star1 = medfilt2(car_img_star1,[3,3]);
filter_img_star1 = flipud(filter_img_star1);

%% preprocessing---car2
a=69;b=205;c=488; d=357;
car_img = im2double(imread('F:\252\car2.jpg'));
filter_img_ori = im2double(imread('F:\252\cartemplate.jpg'));
%rescale the image
filter_img = imresize(filter_img_ori,[d-b+30 c-a+30]);
filter_img_star1 = imresize(imcrop(filter_img,[25,55,c-a-10,d-b-40]),[150 420]);
[ltoy,wtoy] = size(car_img);[lfil,wfil] = size(filter_img_star1);
%transfer to bw
filter_img_star1 = double(im2bw(filter_img_star1,0.8));
car_img_star1 = double(im2bw(car_img,0.05));
%substract mean
filter_img_star1 = filter_img_star1 - mean(filter_img_star1(:));
car_img_star1 = car_img - mean(car_img(:));
%flip
filter_img_star1 = flipud(filter_img_star1);

%% preprocessing---car3
```

```
a=329;b=251;c=480; d=345;
car_img = im2double(imread('F:\252\car3.jpg'));
filter_img_ori = im2double(imread('F:\252\cartemplate.jpg'));
filter_img = imresize(filter_img_ori,[d-b+30 c-a+30]);
filter_img_star1 = imresize(imcrop(filter_img,[10,40,c-a+15,d-b-20]),[100 150]);
[ltoy,wtoy] = size(car_img);[lfil,wfil] = size(filter_img_star1);
filter_img_star1 = filter_img_star1 - mean(filter_img_star1(:));
car_img_star1 = car_img - mean(car_img(:));
% gaussian filter
GaussFil = fspecial('gaussian',[5 5],0.6);
filter_img_star1 = imfilter(filter_img_star1,GaussFil,'same');
car_img_star1 = imfilter(car_img_star1,GaussFil,'same');
% inverse color
filter_img_star1 = 0.45-filter_img_star1;
filter_img_star1 = flipud(filter_img_star1);

%% preprocessing---car4
a=173;b=163;c=463; d=261;
car_img = im2double(imread('F:\252\car4.jpg'));
filter_img_ori = im2double(imread('F:\252\cartemplate.jpg'));
filter_img = imresize(filter_img_ori,[d-b+30 c-a+30]);
filter_img_star1 = imresize(imcrop(filter_img,[20,40,c-a-15,d-b-20]),[100 300]);
[ltoy,wtoy] = size(car_img);[lfil,wfil] = size(filter_img_star1);
filter_img_star1 = filter_img_star1 - mean(filter_img_star1(:));
car_img_star1 = car_img - mean(car_img(:));
filter_img_star1 = double(im2bw(filter_img_star1,0.1));
car_img_star1 = double(im2bw(car_img_star1,0.4));
filter_img_star1 = flipud(filter_img_star1);
filter_img_star1 = fliplr(filter_img_star1);

%% preprocessing---car5
a=393;b=313;c=800; d=497;
car_img = im2double(imread('F:\252\car5.jpg'));
filter_img_ori = im2double(imread('F:\252\cartemplate.jpg'));
```

```
68 filter_img = imresize(filter_img_ori,[d-b+30 c-a+30]);
69 filter_img_star1 = imresize(imcrop(filter_img,[25,65,c-a-15,d-b-50]),[160 360]);
70 [ltoy,wtoy] = size(car_img);[lfil,wfil] = size(filter_img_star1);
71 filter_img_star1 = filter_img_star1 - mean(filter_img_star1(:));
72 car_img_star1 = car_img - mean(car_img(:));
73 filter_img_star1 = flipud(filter_img_star1);
74 filter_img_star1 = fliplr(filter_img_star1);
75
76 %% convolution
77 C = conv2(car_img_star1, filter_img_star1, 'same');
78 colormap('default');
79 figure(1);
80 imagesc(C);
81
82 %% find max and output the box
83 max1=max(C);
84 max2=max(max1);
85 [row, col] = find(C==max2);
86 x1=col(1)-(wfil-1)/2; y1=row(1)-(lfil-1)/2;
87 x1min=x1;x1max=x1+wfil-1;y1min=y1;y1max=y1+lfil-1;
88 x2min=a;x2max=c;y2min=b;y2max=d;
89
90 figure(2);
91 imshow(car_img);
92 axis on;
93 hold on;
94 rectangle('Position',[x1,y1,wfil,lfil],'edgecolor','b');
95 rectangle('Position',[x2min,y2min,x2max-x2min+1,y2max-y2min+1],'edgecolor','r');
96
97 %% calculate overlap rate
98 overlaprate = OverlapFunc(x1min,y1min,x1max,y1max,x2min,y2min,x2max,y2max);
```

Listing 4: Codes of hw3 part5.1-script-corner detect

```
%% load data
clear;clc;
load('F:\252\matrix2');
%load('F:\252\warrior2');
%load('F:\252\dino2');
%I1 = rgb2gray(warrior01);
%I2 = rgb2gray(warrior02);
%I1 = rgb2gray(dino01);
%I2 = rgb2gray(dino02);
I1 = rgb2gray(matrix01);
I2 = rgb2gray(matrix02);

height1 = size(I1, 1);width1 = size(I1, 2);
height2 = size(I2, 1);width2 = size(I2, 2);


%% find 10 corners for the image
nCorners = 20;
windowSize = 11;
smoothSTD = 0.5;
[corners1y corners1x] = CornerDetect(I1, nCorners, smoothSTD, windowSize);
corners1 = [corners1x corners1y];
[corners2y corners2x] = CornerDetect(I2, nCorners, smoothSTD, windowSize);
corners2 = [corners2x corners2y];

figure(1);
imshow(I1);
hold on;
axis on;
 for i=1:nCorners
    r = 30;
    sita=0:pi/20:2*pi;
    plot(corners1y(i)+r*cos(sita),corners1x(i)+r*sin(sita),'LineWidth',2);
```

```
34   end
35  figure(2);
36  imshow(I2);
37  hold on;
38  axis on;
39   for i=1:nCorners
40       r = 30;
41       sita=0:pi/20:2*pi;
42       plot(corners2y(i)+r*cos(sita),corners2x(i)+r*sin(sita),'LineWidth',2);
43   end
```

Listing 5: Codes of hw3 part5.1-function-CornerDetect

```
function [cornery,cornerx] = CornerDetect( Image, nCorners, smoothSTD, windowSize)
%% load data
image = Image;
imgHeight = size(image, 1);imgWidth = size(image, 2);
halfWsize = floor(windowSize/2);

%% apply gaussian filter
GaussFil = fspecial('gaussian',[5 5],smoothSTD);
Ig_Filted = imfilter(image,GaussFil,'same');

%% compute the gradient everywhere.
[Ix,Iy] = gradient(Ig_Filted);
for i=1:(halfWsize+1)
    Ix(:,i)=0;
end
for i=(imgWidth-halfWsize):imgWidth
    Ix(:,i)=0;
end
for i=1:(halfWsize+1)
    Iy(i,:)=0;
end
for i=(imgHeight-halfWsize):imgHeight
    Iy(i,:)=0;
end

%% move window over image and construct C over the window.
lamda = zeros(imgWidth,imgHeight);
for x=1:imgWidth
    for y=1:imgHeight
        Ixsub = Window(Ix, x, y, halfWsize);
        Iysub = Window(Iy, x, y, halfWsize);
        Ixx=Ixsub.*Ixsub;
        Ixy=Ixsub.*Iysub;
```

```
34              Iyy=Iysub.*Iysub;
35              C=[sum(sum(Ixx)),sum(sum(Ixy));sum(sum(Ixy)),sum(sum(Iyy))];
36              [V,D]=eig(C);
37              lamda(x,y) = min(D(1,1),D(2,2));
38          end
39      end
40
41      [cornery cornerx] = NMS2(lamda,nCorners);
42
43      end
```

Listing 6: Codes of hw3 part5.1-function-NMS2

```
function [y, x] = NMS2(cimg, max_pts)
%% initialization
[m,n] = size(cimg);
mean_cimg = mean(mean(cimg));
boarder = 1;
%% local non-maximum suppression
local_max = zeros(m,n);
x_local = zeros(1000,1);
y_local = zeros(1000,1);
value_local = zeros(1000,1);
k = 1;
nmssize = 25;
% Search for the pixels that have a corner strength larger than the pixels
% around it and larger than the average strength of the image.
for i = (boarder+nmssize):(m-boarder-nmssize)
    for j = (boarder+nmssize):(n-boarder-nmssize)
        a = cimg(i,j);
        b = max(max(cimg((i-nmssize):(i+nmssize),(j-nmssize):(j+nmssize))));
        if ((a>mean_cimg)&&(a>=b))
            local_max(i,j)=1;
            value_local(k) = a;
            y_local(k) = i;
            x_local(k) = j;
            k = k+1;
        end
    end
end
y_local(k:end) = [];
x_local(k:end) = [];
value_local(k:end) = [];

%% sort the local maximum value and output the most NUM value and coordinate
rij = [value_local, y_local, x_local];
```

```
34  rij_sorted = sortrows(rij,-1);
35
36  rij_cut = rij_sorted(1:max_pts,:);
37  y = rij_cut(:, 2);
38  x = rij_cut(:, 3);
39
40  end
```

Listing 7: Codes of hw3 part5.1-function-Window

```
function [ I ] = Window( Img, centerX,centerY,WSize )

ImgX=size(Img,2);
ImgY=size(Img,1);

xs=round(centerX-WSize);
if xs<1
    xs=1;
end
xe=round(centerX+WSize);
if xe>ImgX
    xe=ImgX;
end

ys=round(centerY-WSize);
if ys<1
    ys=1;
end
ye=round(centerY+WSize);
if ye>ImgY
    ye=ImgY;
end

I=Img(ys:ye, xs:xe);
end
```

Listing 8: Codes of hw3 part5.2-function-SSDmatch

```
function [ SSD ] = SSDmatch( I1 , I2 )
if(size(I1 ,1)~=size(I2 ,1) || size(I1 ,2)~=size(I2 ,2))
    I2=zeros(size(I2 ,1) , size(I2 ,2));
end
height = min(size(I1 , 1),size(I2 , 1));width = min(size(I1 , 2),size(I2 , 2));
SSD=0;
m=size(I1 ,1);
n=size(I2 ,2);
for i = 1:height
    for j= 1:width
        if pdist([ceil(m/2),ceil(n/2);i , j])<=ceil(m/2)
        diff = I1(i , j)-I2(i , j);
        square = diff^2;
        SSD = SSD + square;
        end
    end
end
end
```

Listing 9: Codes of hw3 part5.3-script-Naivematching

```
%% load data
clear;clc;
load('F:\252\matrix2');
load('F:\252\warrior2');
load('F:\252\dino2');
warrior01_gray = rgb2gray(warrior01);
warrior02_gray = rgb2gray(warrior02);
dino01_gray = rgb2gray(dino01);
dino02_gray = rgb2gray(dino02);
matrix01_gray = rgb2gray(matrix01);
matrix02_gray = rgb2gray(matrix02);

%% compute w/ two images
nCorners = 10;
windowSize = 11;
smoothSTD = 5;
SSDth = 5;
I1 = matrix01_gray;
I2 = matrix02_gray;
R = 5;
[corners1y corners1x] = CornerDetect(I1, nCorners, smoothSTD, windowSize);
[corners2y corners2x] = CornerDetect(I2, nCorners, smoothSTD, windowSize);
corners1 = [corners1x corners1y];
corners2 = [corners2x corners2y];
[I, corsSSD] = naiveCorrespondanceMatching(I1, I2, corners1, corners2, R, SSDth);
corners2yp = size(I1, 2) + corners2y;

%% output images

figure(1);
imshow(I);
hold on;
 for i=1:nCorners
```

```
34        r = 30;
35        sita=0:pi/20:2*pi;
36        plot(corners1y(i)+r*cos(sita),corners1x(i)+r*sin(sita),'LineWidth',2);
37        plot(corners2yp(i)+r*cos(sita),corners2x(i)+r*sin(sita),'LineWidth',2);
38     end
39     hold on;
40    for i=1:nCorners
41        if (corsSSD(i)~=0)
42        j = corsSSD(i);
43        line([corners1y(i) corners2yp(j)],[corners1x(i) corners2x(j)],'LineWidth',2,'Color','r
44        end
45    end
```

Listing 10: Codes of hw3 part5.3-function-naiveCorrespondanceMatching

```
function [ I, corsSSD ] = naiveCorrespondanceMatching( I1, I2, corners1, corners2, R, SSDth
height1 = size(I1, 1);width1 = size(I1, 2);
height2 = size(I2, 1);width2 = size(I2, 2);
height = max(height1,height2);
width = width1 + width2;
I = zeros(height, width);
I(1:height1,1:width1)=I1;
I(1:height2,width1+1:width)=I2;
corners1x = corners1(:,1);
corners1y = corners1(:,2);
corners2x = corners2(:,1);
corners2y = corners2(:,2);

for i=1:length(corners1x)
    matchimg1 = Window(I1,corners1y(i),corners1x(i),R);
    for j=1:length(corners2x)
        matchimg2 = Window(I2,corners2y(j),corners2x(j),R);
        tmdSSD(1,j) = SSDmatch( matchimg1,matchimg2 );
    end
    minSSD = min(tmdSSD(tmdSSD>0));
    if (minSSD>SSDth)
        corsSSD(i,1) = 0;
    else
    corsSSD(i,1) = find(tmdSSD==minSSD);
    end
end
end
```

Listing 11: Codes of hw3 part5.4-script-epipolar geometry

```
%% load data
clear;clc;
load('F:\252\matrix2');
%load('F:\252\warrior2');
%load('F:\252\dino2');
%I1 = rgb2gray(warrior01);
%I2 = rgb2gray(warrior02);
%I1 = rgb2gray(dino01);
%I2 = rgb2gray(dino02);
I1 = rgb2gray(matrix01);
I2 = rgb2gray(matrix02);
height1 = size(I1, 1);width1 = size(I1, 2);
height2 = size(I2, 1);width2 = size(I2, 2);
nCorners = 14;

%% calculate fundamental matrix
F = fund(cor1,cor2);
for i=1:nCorners
    line2(i,:,:) = F*[cor1(i,:),1]';
    line1(i,:,:) = F'*[cor2(i,:),1]';
end
for i=1:nCorners
    pts1(:,:,i) = linePts(line1(i,:,:), [1;width1], [1;height1]);
end
for i=1:nCorners
    pts2(:,:,i) = linePts(line2(i,:,:), [1;width2], [1;height2]);
end
%% output images
figure(1);
imshow(I1);
hold on;
 for i=1:nCorners
    r = 30;
```

```
34        sita=0:pi/20:2*pi;
35        plot(cor1(i,1)+r*cos(sita),cor1(i,2)+r*sin(sita),'LineWidth',2);
36        line([pts1(1,1,i) pts1(2,1,i)],[pts1(1,2,i) pts1(2,2,i)],'LineWidth',2,'Color','b');
37    end
38
39    figure(2)
40    imshow(I2);
41    hold on;
42  for i=1:nCorners
43        r = 30;
44        sita=0:pi/20:2*pi;
45        plot(cor2(i,1)+r*cos(sita),cor2(i,2)+r*sin(sita),'LineWidth',2);
46        line([pts2(1,1,i) pts2(2,1,i)],[pts2(1,2,i) pts2(2,2,i)],'LineWidth',2,'Color','b');
47  end
```

Listing 12: Codes of hw3 part5.5-script-epipolarline matching

```
%% load data
clear;clc;
load('F:\252\matrix2');
%load('F:\252\warrior2');
%load('F:\252\dino2');
%I1 = rgb2gray(warrior01);
%I2 = rgb2gray(warrior02);
%I1 = rgb2gray(dino01);
%I2 = rgb2gray(dino02);
I1 = rgb2gray(matrix01);
I2 = rgb2gray(matrix02);

height1 = size(I1, 1);width1 = size(I1, 2);
height2 = size(I2, 1);width2 = size(I2, 2);


%% find 10 corners for each picture
nCorners = 10;
windowSize = 11;
smoothSTD = 0.5;
SSDth = 2.5;
R = 5;
[corners1y corners1x] = CornerDetect(I1, nCorners, smoothSTD, windowSize);
corners1 = [corners1x corners1y];

%% calculate fundamental matrix
F = fund(cor1,cor2);

% search along the epipolar line in I2


[I,corsSSD] = correspondanceMatchingLine( I1, I2, corners1, F, R, SSDth);

```

```
34
35
36 %% output images
37
38 figure(1);
39 imshow(I);
40 hold on;
41 axis on;
42  for i=1:nCorners
43      r = 20;
44      sita=0:pi/20:2*pi;
45      plot(corners1y(i)+r*cos(sita),corners1x(i)+r*sin(sita),'LineWidth',2);
46  end
47  hold on;
48 for i=1:nCorners
49      if (corsSSD(i,1)~=0)
50          xm = corsSSD(i,1);
51          ym = corsSSD(i,2);
52          plot(xm+size(I1, 2)+r*cos(sita),ym+r*sin(sita),'LineWidth',2);
53          line([corners1y(i) xm+size(I1, 2)],[corners1x(i) ym],'LineWidth',2,'Color','r');
54      end
55 end
```

Listing 13: Codes of hw3 part5.5-function-correspondanceMatchingLine

```
function [ I,corsSSD ] = correspondanceMatchingLine(I1, I2, corners1, F, R, SSDth )

height1 = size(I1, 1);width1 = size(I1, 2);
height2 = size(I2, 1);width2 = size(I2, 2);
nCorners = 50;
height = max(height1,height2);
width = width1 + width2;
I = zeros(height, width);
I(1:height1,1:width1)=I1;
I(1:height2,width1+1:width)=I2;
corners1x = corners1(:,1);
corners1y = corners1(:,2);
line2 = epipolarLine(F,[corners1y corners1x]);
line2_u1(1:nCorners,1) = line2(:,1)./line2(:,3);
line2_u2(1:nCorners,1) = line2(:,2)./line2(:,3);
line2_u3(1:nCorners,1) = 1;
line2_homo = [line2_u1 line2_u2 line2_u3];
for i=1:nCorners
    pts2(:,:,i) = linePts(line2_homo(i,:,:), [1;width2], [1;height2]);
end
yest = zeros(width2,nCorners);
tmdSSD = zeros(width2,nCorners);
minSSD = zeros(nCorners);
corsSSD = zeros(nCorners,4);
for i=1:nCorners
    matchimg1 = Window(I1,corners1y(i),corners1x(i),R);
    for j=1:width2
        slope = (pts2(1,2,i)-pts2(2,2,i))/(pts2(1,1,i)-pts2(2,1,i));
        if (round(pts2(1,2,i)-slope*(pts2(1,1,i)-j))<height2
            && round(pts2(1,2,i)-slope*(pts2(1,1,i)-j))>0)
            yest(j,i) = round(pts2(1,2,i)-slope*(pts2(1,1,i)-j));
        elseif(round(pts2(1,2,i)-slope*(pts2(1,1,i)-j))>=height2)
            yest(j,i) = height2;
```

```
34            elseif(round(pts2(1,2,i)-slope*(pts2(1,1,i)-j))<=0)
35                yest(j,i) = 0;
36            end
37            matchimg2 = Window(I2,j,yest(j,i),R);
38            tmdSSD(j,i) = SSDmatch( matchimg1,matchimg2 );
39        end
40            a=tmdSSD(:,i);
41            minSSD(i) = min(a(a>0));
42        if (minSSD(i)>SSDth)
43            corsSSD(i,1) = 0;
44            corsSSD(i,2) = 0;
45        else
46            corsSSD(i,1) = find(a==minSSD(i));
47            corsSSD(i,2) = yest(corsSSD(i,1),i);
48        end
49        corsSSD(i,3) = corners1y(i);
50        corsSSD(i,4) = corners1x(i);
51    end
52    end
```

Listing 14: Codes of hw3 part5.6-script-triangulation

```
%% load data
clear;clc;
%load('F:\252\matrix2');
load('F:\252\warrior2');
%load('F:\252\dino2');
I1 = rgb2gray(warrior01);
I2 = rgb2gray(warrior02);
%I1 = rgb2gray(dino01);
%I2 = rgb2gray(dino02);
%I1 = rgb2gray(matrix01);
%I2 = rgb2gray(matrix02);
%P1 = proj_dino01; P2 = proj_dino02;
P1 = proj_warrior01; P2 = proj_warrior02;
%P1 = proj_matrix01; P2 = proj_matrix02;
height1 = size(I1, 1);width1 = size(I1, 2);
height2 = size(I2, 1);width2 = size(I2, 2);


%% find 50 corners for each picture
nCorners = 50;
windowSize = 11;
smoothSTD = 0.5;
SSDth = 2.5;
R = 5;
[corners1y corners1x] = CornerDetect(I1, nCorners, smoothSTD, windowSize);
corners1 = [corners1x corners1y];

%% calculate fundamental matrix
F = fund(cor1,cor2);

% search along the epipolar line in I2


```

```
34 [I,corsSSD] = correspondanceMatchingLine( I1, I2, corners1, F, R, SSDth);
35
36
37 %% triangulation and find outliers
38 outlierTH = 20;
39 points3D = triangulate(corsSSD, P1, P2);
40 [ inlier, outlier ] = findOutliers(points3D, P2, outlierTH, corsSSD);
41
42 %% output images
43
44 figure(1);
45 imshow(I2);
46 hold on;
47 axis on;
48  for i=1:nCorners
49       r = 20;
50       sita=0:pi/20:2*pi;
51      if (corsSSD(i,1)~=0)
52          xm = corsSSD(i,1);
53          ym = corsSSD(i,2);
54          plot(xm+r*cos(sita),ym+r*sin(sita),'k','LineWidth',2);
55      end
56 end
57  hold on;
58 for i=1:size(inlier,1)
59       plot(inlier(i,1),inlier(i,2),'b+','LineWidth',2);
60 end
61 for i=1:size(outlier,1)
62       plot(outlier(i,1),outlier(i,2),'r+','LineWidth',2);
63 end
```

Listing 15: Codes of hw3 part5.6-function-triangulate

```
function [points3D] = triangulate(corsSSD, P1,P2)
%% make matrix
%P1 = proj_dino01; P2 = proj_dino02;
length = size(corsSSD,1);
for i=1:length
    x1(i) = corsSSD(i,3);
    y1(i) = corsSSD(i,4);
    x2(i) = corsSSD(i,1);
    y2(i) = corsSSD(i,2);
    A1 = x1(i)*P1(3,:)-P1(1,:);
    A2 = y1(i)*P1(3,:)-P1(2,:);
    A3 = x2(i)*P2(3,:)-P2(1,:);
    A4 = y2(i)*P2(3,:)-P2(2,:);
    A = [A1;A2;A3;A4];
    [U,S,V] = svd(A);
    points3D(i,1)=V(1,4); points3D(i,2)=V(2,4);
    points3D(i,3)=V(3,4);  points3D(i,4)=V(4,4);
end
end
```

Listing 16: Codes of hw3 part5.6-function-findOutliers

```
function [ inlier, outlier ] = findOutliers(points3D, P2, outlierTH, corsSSD);
length = size(corsSSD,1);
for i=1:length
    truepts2homo(i,:) = P2*points3D(i,:)';
    truepts2(i,1)=truepts2homo(i,1)/truepts2homo(i,3);
    truepts2(i,2)=truepts2homo(i,2)/truepts2homo(i,3);
end
a=1;b=1;
for i=1:length
    if (corsSSD(i,1)~=0)
        xo(i) = corsSSD(i,1);
        yo(i) = corsSSD(i,2);
```

```
13          dist = (xo(i)-truepts2(i,1))^2+(yo(i)-truepts2(i,2))^2;
14          if (dist>outlierTH^2)
15              outlier(a,1)= truepts2(i,1);
16              outlier(a,2)= truepts2(i,2);
17              a=a+1;
18          else
19              inlier(b,1)=truepts2(i,1);
20              inlier(b,2)=truepts2(i,2);
21              b=b+1;
22          end
23      end
24  end
25
26  end
```

Submitted by Mingxuan Wang on December 4, 2014.