The DoodleVerse Lab for CS 532: Theory and Application of Pattern Recognition

Lingjiao Chen

December 12, 2015

Abstract

This paper includes work for reproduction for Lab 7 "The DoodleVerse" of CS532: Theory and Application of Pattern Recognition, prepared by Lingjiao Chen.

All the problems are prepared by LaTeX.

All related Matlab codes are included in the Appendix.

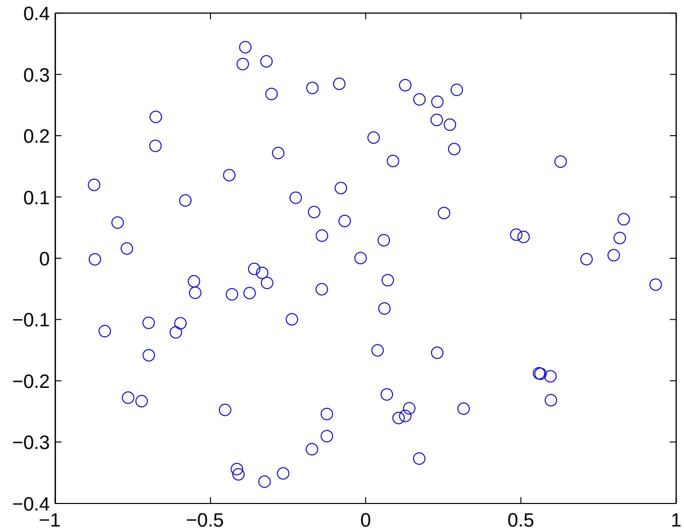

Figure 1: Star locations.

Warm-up Problem 1

Solution: One can use the G_x and G_y provided in the problem, but essentially using a threshold to select all pixels who greater than 128 is enough. The reason is that the boundary is only 1 pixel wide.

Warm-up Problem 2

Solution: Using the points on the boundary shall be a reasonable choice.

Warm-up Problem 3

Solution: The reason is that we need to reduce the number of possible matchings which should be explored later. If there are too many features, then it is very expensive to find the best matching. Another possible reason is that these starts cannot be expected to match too many feature points because they are relatively sparse compared to the diamond boundary points.

Warm-up Problem 4

Solution:

The star points are found as shown in Figure 1.

Warm-up Problem 5

Solution: In the form of clustering problem, $X = (X_1, X_2, \cdots, X_m) \in R^{n\times m}$, where $n = 3$ denotes all the samples we have, where $X_i \in R^n$is one sample representing the three angles of the triangle which it represents. $D = (D_1, D_2, \cdots, Dk)$ is the set of potential triangles we have in the dictionary, where D_i is one triangle. $W = (W_1, W_2, \cdots, W_m)$ is the indicator for each training sample, where all entries in W_i is 0 except one one whose position implies the best triangle in our dictionary that this sample matches.

n is 3 since it is the dimension of each sample, which is the number of angles in a triangle.

As k increases, we have more and more dictionary triangles and hence the error will be decreased.

Warm-up Problem 6

Solution: Clustering is preferred since the computational complexity of it is much less than that of brute-force method.

Warm-up Problem 7

Solution: Note that by the property of trace,

$$\begin{aligned}
T^* &= \arg\min_T ||TF - S||_F^2 \\
&= \arg\min_T \operatorname{tr}((TF - S)(TF - S))^T \\
&= \arg\max_T \operatorname{tr}(SF^T T^T) \\
&= \arg\max_T \operatorname{tr}(U\Sigma V^T T^T) \\
&= \arg\max_T \operatorname{tr}(U^T TV\Sigma).
\end{aligned} \tag{1}$$

$U^T TV$ is orthogonal matrix, so

$$\operatorname{tr}(U^T TV\Sigma) \leq \operatorname{tr}(\Sigma). \tag{2}$$

When $T = UV^T$, we have $U^T TV = I$ and thus $\operatorname{tr}(U^T TV\Sigma) = \operatorname{tr}(\Sigma)$. Hence, $T^* = UV^T$ is a minimizer of the original problem.

Warm-up Problem 8

Solution: We write our own matlab codes to compute the transformation T, which is $(0.5000, 0.8660; -0.8660, 0.5000)$.

Lab Problem 1

Comment: This problem is directly related to edge detector, which is essentially an image filter.
Solution:

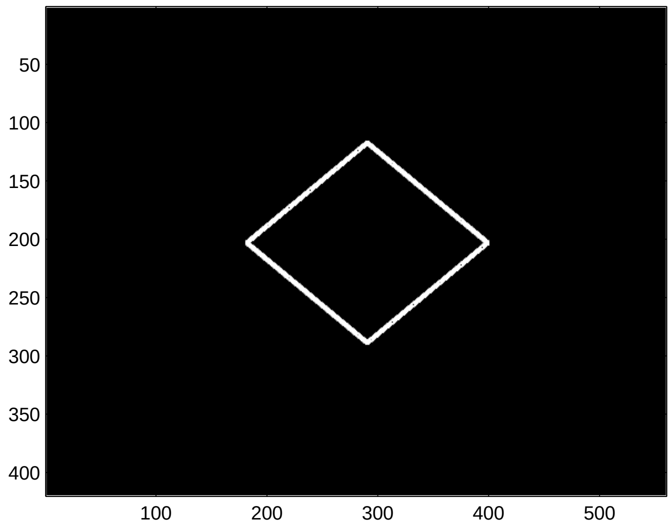

Figure 2: The edge detection result.

As shown in Figure 2, The edge detector results in a set of features which look like a diamond.

Lab Problem 2

Comment: There is no specific requirement for how to reduce it, so we do it manually.
Solution: By hand, we can find the following 4 points, $(118, 291), (203, 183), (288, 291), (203, 397)$, to be the reduced features. These points consist of the edge points of the diamond we obtain in Problem 1.

Lab Problem 3

Comment: This problem can be solved by brute-force searching.
Solution:

The results are summarized in Figure 3.

One can see that the second and third triangles are very close to each other.

Lab Problem 4

Comment: Note that the pre-transformation formulate given in the problem is not right.
Solution: The correct formulas should be

$$\hat{F} = \frac{F - \text{mean}(F)}{||F||_F} \tag{3}$$

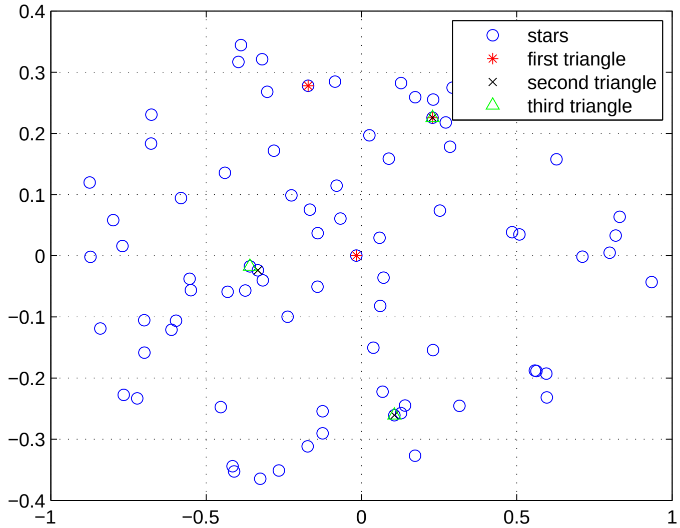

Figure 3: The best three matchings.

and

$$\hat{S} = \frac{S - \text{mean}(S)}{||F||_F}. \tag{4}$$

As Figure 4 shows, the projected diamonds do not perfectly match the original stars.

Lab Problem 5

Comment: The cost function should be in line with the intuition in terms of similarity.
Solution: The correct formulas should be

As Figure 5 shows, the first triangle does not match the diamond pattern quite well, but the second and the third one are quite close to a diamond. Based on our cost function, the second one is a little bit better. Our cost function does not depend on the star magnitude.

Appendix

All the Matlab codes are presented in this part. Warm-up Problem 4:
The main script is given as follows.

Listing 1: main for Warm-up Problem 4

```
%Warm-up Problem 4
star = load('stars.mat');
RA = star.RA/180*pi;
```

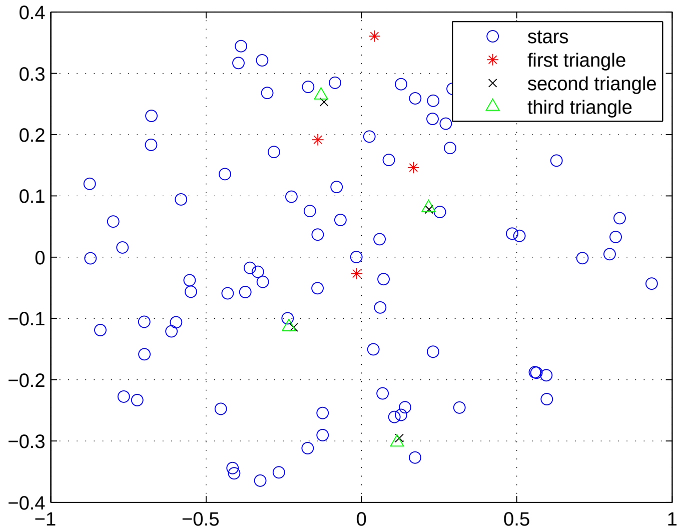

Figure 4: Projected diamond in star space.

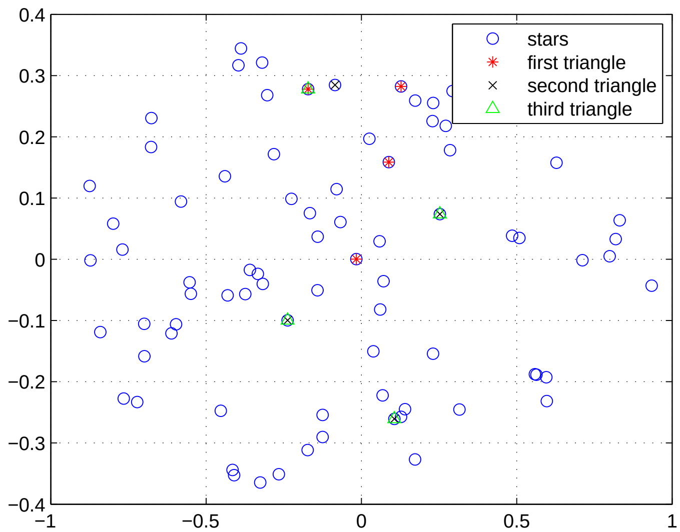

Figure 5: Best match for each projected diamond.

```
Dec = star.Dec/180*pi;
Mag = star.Mag;
n = size(RA,2);
x = [];
y = [];
lambdac = 293/180 *  pi;
phic = -2.8/180 * pi;
for i = 1:n
    theta1 = fsolve(@(x) theta_star(x,Dec(i),phic),0.5 );
    x(i) = 2 * sqrt(2) * ( RA(i) - lambdac ) * cos(theta1);
    y(i) = sqrt(2) * sin(theta1);
end
xyori = [x;y];
figure(2);
plot(x,y, 'o');
```

.

Warm-up Problem 8:
The main script is given as follows.

Listing 2: main for Warm-up Problem 8

```
%Problem 4
wdata = load('procrustes.mat')
[U,S,V] = svd(wdata.B*wdata.A');
T = U*V'
```

.

Lab Problem 1:
The main script is given as follows.

Listing 3: main for Lab Problem 1

```
%Problem 1
%
A = imread('diamond.png');
B = rgb2gray(A);
colormap(gray);
image(B);
%
Gx = [-1, 0, 1; -2, 0 , 2; -1,0,1];
Gy = [-1,-2 ,-1; 0, 0, 0; 1, 2, 1];
Cx = conv2(double(B),Gx,'same');
Cy = conv2(double(B),Gy,'same');

%G = sqrt(Gx.*Gx + Gy.* Gy);
%EdgeA = conv2(double(B),G,'same');
EdgeA = sqrt(Cx.*Cx + Cy.*Cy);
figure(1);
```

```
colormap(gray);
image(EdgeA);
```

Lab Problem 2:
The main script is given as follows.

Listing 4: main for Lab Problem 2

```
%Problem 2
%Four points
%(118,291) [362.926989902928;]
%(203, 183) [202.385770250776;]
% (288,291) [164.347193465541;]
% (203, 397) [668.766027845315;]
```

Lab Problem 3:
The main script is given as follows.

Listing 5: main for Lab Problem 3

```
%Problem 3
star = load('stars.mat');
RA = star.RA/180*pi;
Dec = star.Dec/180*pi;
Mag = star.Mag;
n = size(RA,2);
x = [];
y = [];
lambdac = 293/180 *  pi;
phic = -2.8/180 * pi;
for i = 1:n
    theta1 = fsolve(@(x) theta_star(x,Dec(i),phic),0.5 );
    x(i) = 2 * sqrt(2) * ( RA(i) - lambdac ) * cos(theta1);
    y(i) = sqrt(2) * sin(theta1);
end
xyori = [x;y];
figure(2);
plot(x,y, 'o');

feature = [118,291;203,183;288,291;203,397];% 4 point features
f3o = feature(1:3,:);
f3 = [f3o(1,:) - f3o(2,:) ; f3o(2,:) - f3o(3,:) ; f3o(3,:) - f3o(1,:)];
A3 = [acos(-f3(1,:)*f3(2,:)'/norm(f3(1,:),2)/norm(f3(2,:),2)) ; acos(-
    f3(1,:)*f3(3,:)'/norm(f3(1,:),2)/norm(f3(3,:),2)); acos(-f3(3,:)*f3
    (2,:)'/norm(f3(3,:),2)/norm(f3(2,:),2))     ];
sort(A3,'descend');
%We now can match stars with three of them
[starY,starI] = sort(Mag);
%reverse the order
```

```
xyo = [x(starI);y(starI)]';
tol = 5e-2;
MyResult =[];
counter = 0;
for i = 1:n
    if(counter == 3)
        break;
    end
    for j = (i+1):n
        for k = (j+1):n
            xytemp = xyo([i,j,k],:);
            xy = [xytemp(1,:) - xytemp(2,:) ; xytemp(2,:) - xytemp
                (3,:) ; xytemp(3,:) - xytemp(1,:)];
            Temp3 = [acos(-xy(1,:)*xy(2,:)'/norm(xy(1,:),2)/norm(xy
                (2,:),2)) ; acos(-xy(2,:)*xy(3,:)'/norm(xy(2,:),2)/norm(
                xy(3,:),2)); acos(-xy(1,:)*xy(3,:)'/norm(xy(1,:),2)/norm
                (xy(3,:),2)) ];
            sort(Temp3,'descend');
            error = norm(Temp3-A3,2);
            if(error < tol)
                tempmatch = [i,j,k];
                MyResult = [MyResult;tempmatch];
                counter = counter + 1;
            end
            if(counter == 3)
                break;
            end
        end
    if(counter == 3)
        break;
    end
    end
end

hold on;
scatter( xyo(MyResult(1,:),1), xyo(MyResult(1,:),2), 'r*');
scatter( xyo(MyResult(2,:),1), xyo(MyResult(2,:),2), 'kx');
scatter( xyo(MyResult(3,:),1), xyo(MyResult(3,:),2), 'g^');
legend('stars','first triangle','second triangle','third triangle');
hold off;
grid on;
```

Lab Problem 4:
The main script is given as follows.

Listing 6: main for Lab Problem 4

```
%Problem 4
figure(3);
plot(x,y, 'o');

feature = [118,291;203,183;288,291;203,397];% 4 point features
f3o = feature(1:3,:);
S1 = [xyo(MyResult(1,:),1)'; xyo(MyResult(1,:),2)'];
S2 = [xyo(MyResult(2,:),1)'; xyo(MyResult(2,:),2)'];
S3 = [xyo(MyResult(3,:),1)'; xyo(MyResult(3,:),2)'];
Fproj1 = projection(f3o',S1,feature');
Fproj2 = projection(f3o',S2,feature');
Fproj3 = projection(f3o',S3,feature');
hold on;
scatter( Fproj1(1,:), Fproj1(2,:), 'r*');
scatter( Fproj2(1,:), Fproj2(2,:), 'kx');
scatter( Fproj3(1,:), Fproj3(2,:), 'g^');
legend('stars','first triangle','second triangle','third triangle');
grid on;
hold off;
```

Lab Problem 5:

The main script is given as follows.

Listing 7: main for Lab Problem 5

```
%Problem 4
figure(4);
plot(x,y, 'o');

feature = [118,291;203,183;288,291;203,397];% 4 point features
f3o = feature(1:3,:);
S1 = [xyo(MyResult(1,:),1)'; xyo(MyResult(1,:),2)'];
S2 = [xyo(MyResult(2,:),1)'; xyo(MyResult(2,:),2)'];
S3 = [xyo(MyResult(3,:),1)'; xyo(MyResult(3,:),2)'];
Fproj1 = projection(f3o',S1,feature');
Fproj2 = projection(f3o',S2,feature');
Fproj3 = projection(f3o',S3,feature');
%hold off;

%n is the size of points, i.e., n = 79 in this example
[xclose1,yclose1] = findstar(Fproj1,x,y);
[xclose2,yclose2] = findstar(Fproj2,x,y);
[xclose3,yclose3] = findstar(Fproj3,x,y);
hold on;
scatter( xclose1, yclose1, 'r*');
scatter( xclose2, yclose2, 'kx');
scatter( xclose3, yclose3, 'g^');
legend('stars','first triangle','second triangle','third triangle');
```

```
grid on;
hold off;
```