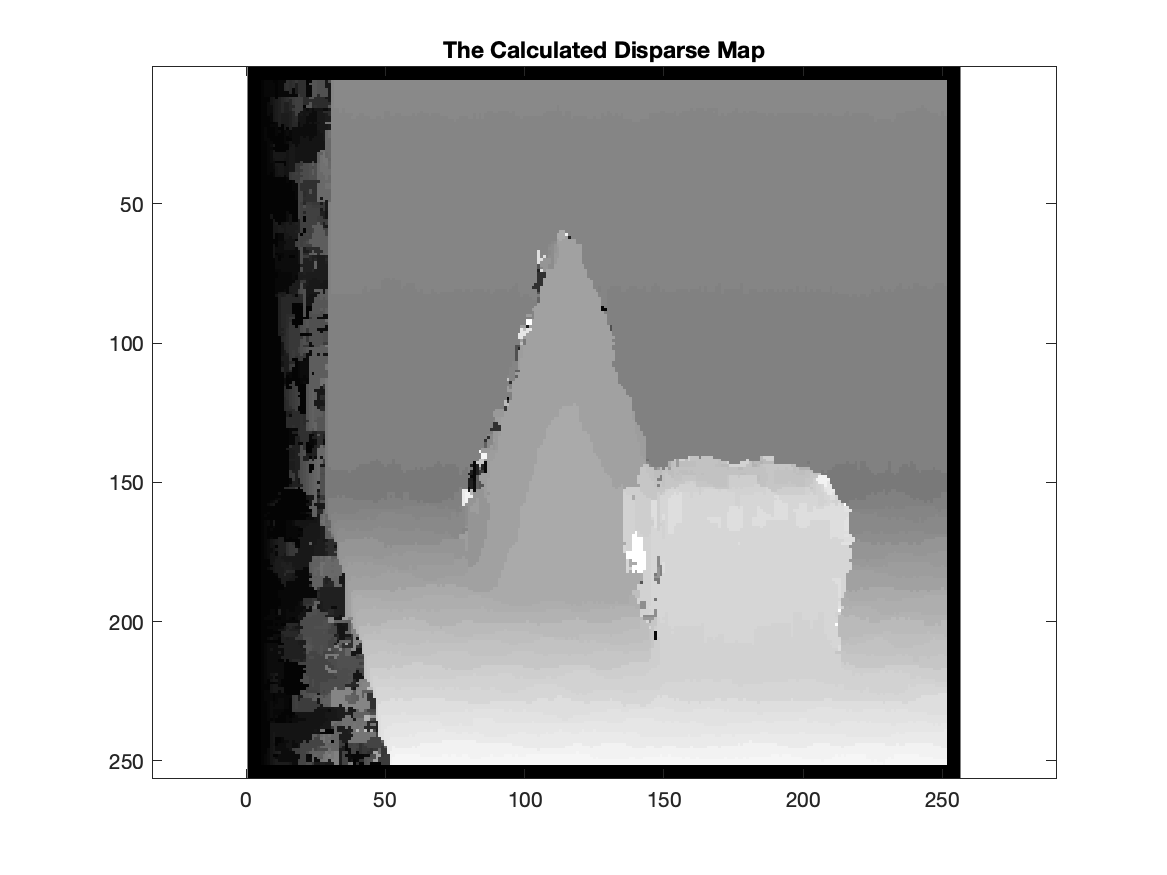
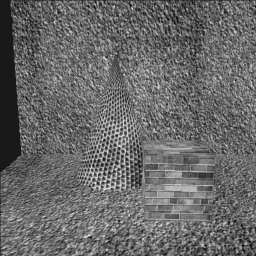
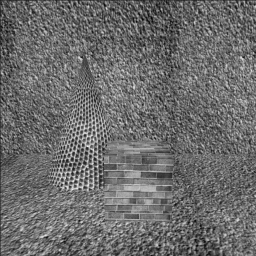
CSE 5524 HW10

Yi Zhao

1. The disparity map is printed below. The brighter area (which looks like a square) locates in the front of the room, and the cone that locates further looks darker than the square in the front. The ground that at the center of image looks darker, and the ground become brighter when it’s getting closer to the front. Overall, the brightness indicates how far the object is from the camera, and the disparity map is a good measurement of how close the object is from camera. 

1. This is the result of KNN classification with K = [1 5 11 15].

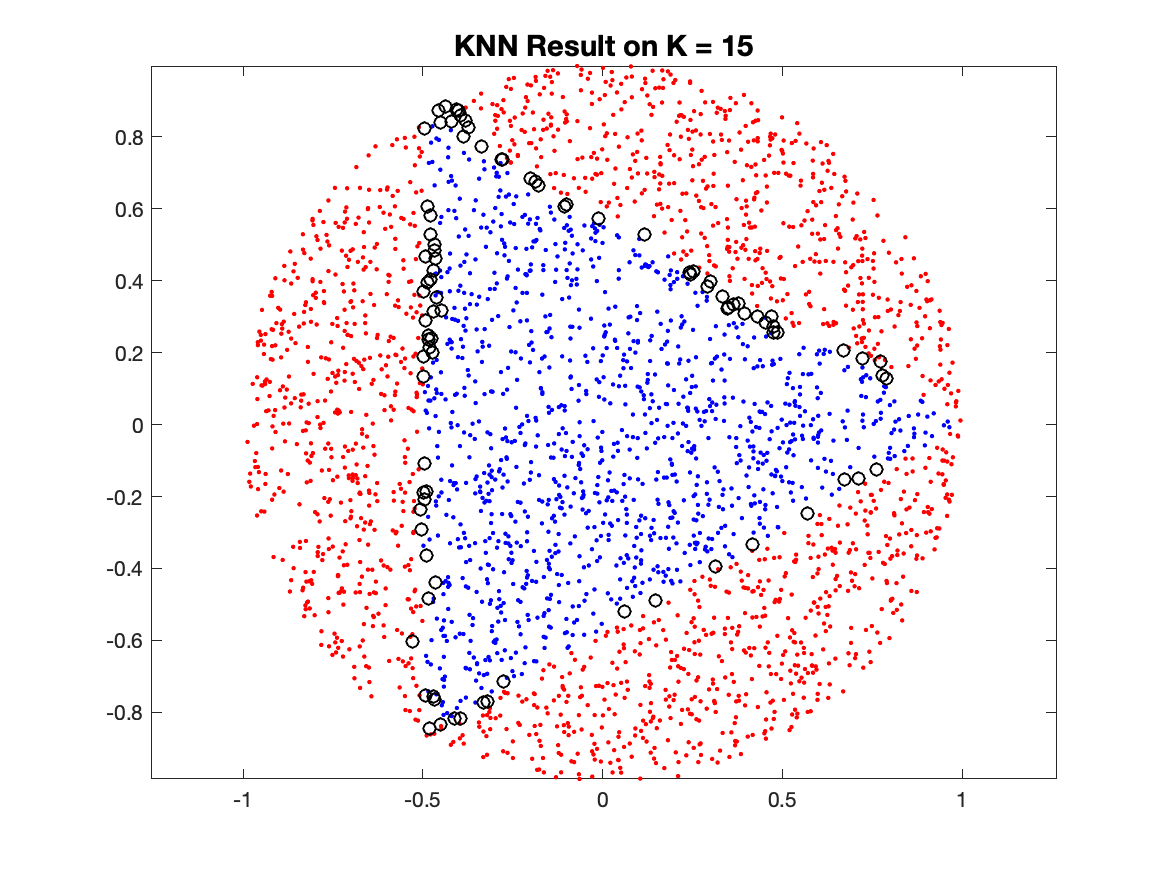
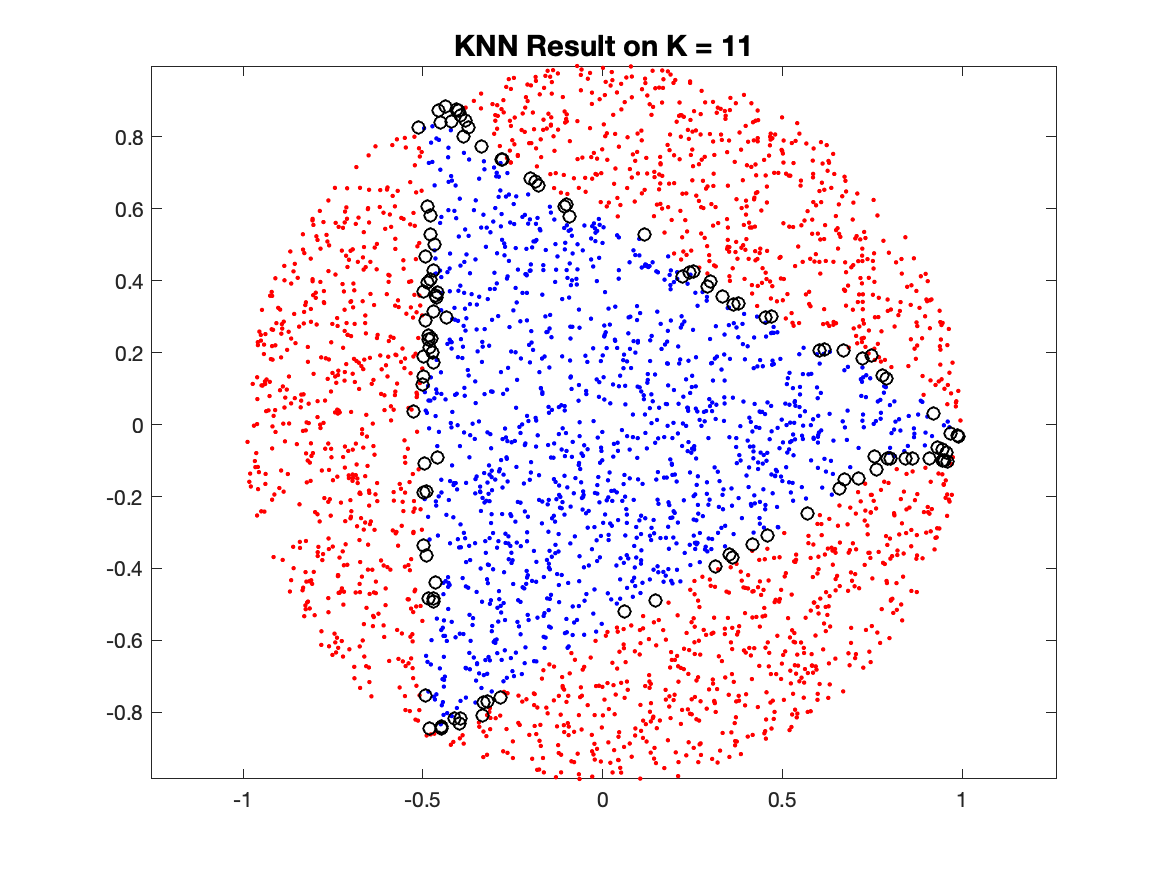
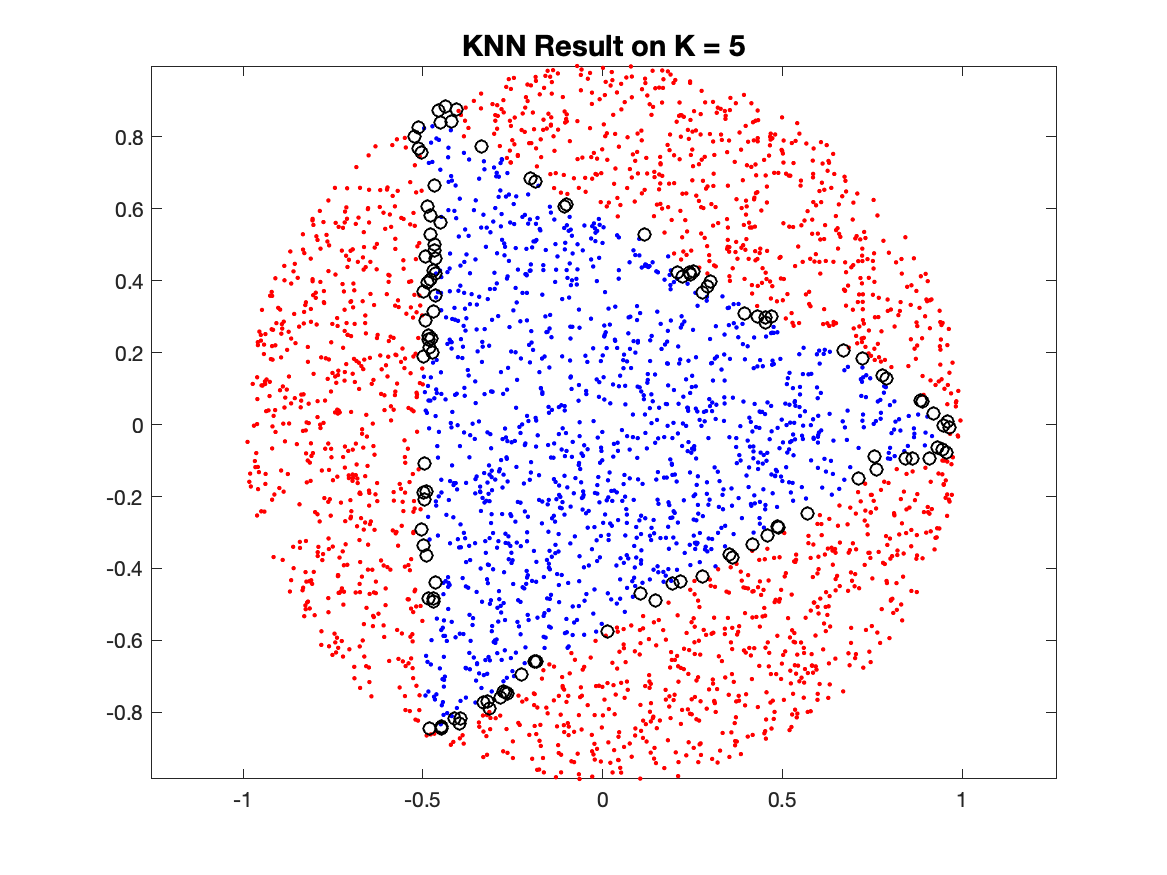
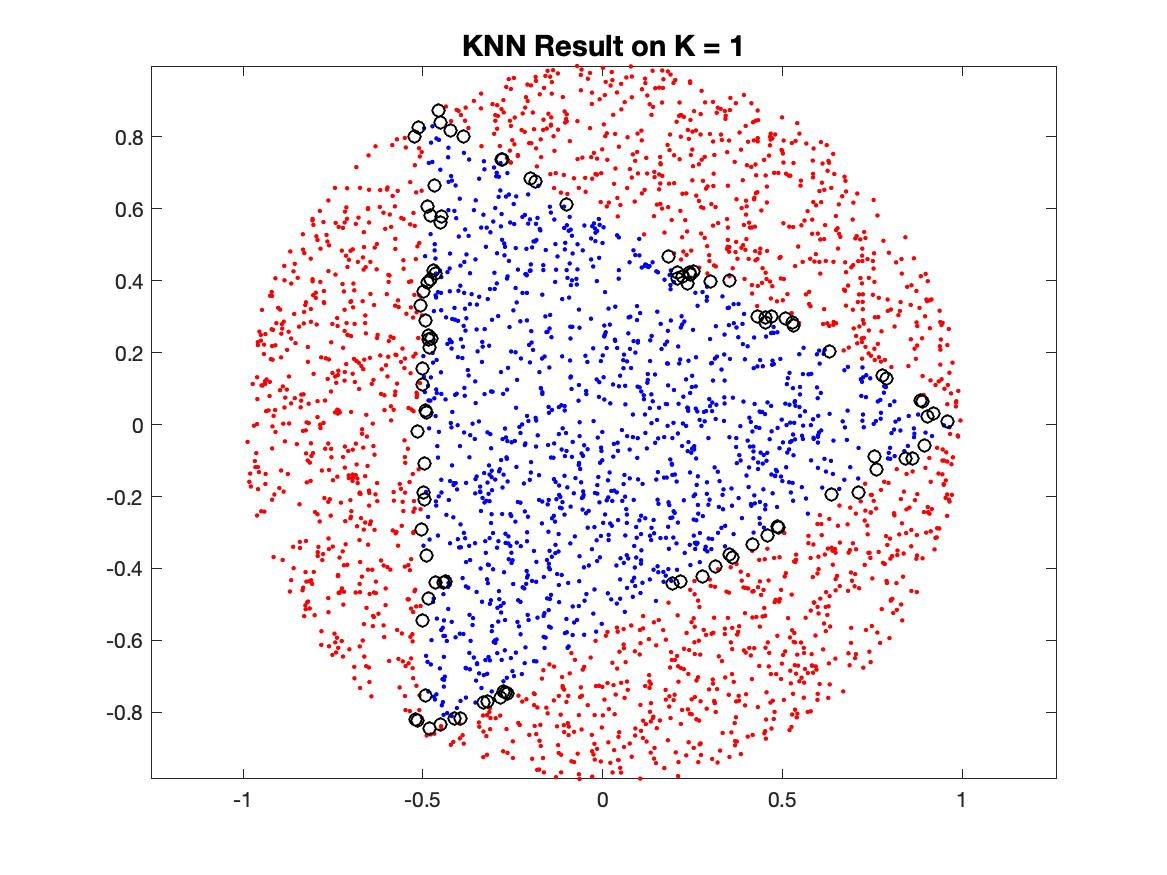
The accuracy at K = 1 is: 96.767%

The accuracy at K = 5 is: 96.333%

The accuracy at K = 11 is: 96.267%

The accuracy at K = 15 is: 96.900%

Some points at the boundary seems to hesitate. When we increase K, we hope to use more information from surrounding points to decide which class it belongs. The points that far from boundaries have accurate classification. The result at K=15 has the highest accuracy, and values between 1 and 15 seems to have less accuracy on this test set.



% Author: Yi Zhao

% CSE 5524, HW10

% 11/10/2019

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Problem 1

left = double(imread('./data/left.png'));

right = double(imread('./data/right.png'));

map = zeros(size(left));

[r,c] = size(left);

l = 11; % window size

h = floor(l/2);

for y = 1:r-l+1

for x = 1:c-l+1

leftwindow = left(y:y+11-1, x:x+11-1);

ncclist = zeros(1,x-max(1, x-50)+1);

for z = 1:x-max(1, x-50)+1

rightwindow = right(y:y+11-1, x-z+1:x-z+11);

ncclist(z) = calculateNCC(leftwindow, rightwindow);

end

[m, m\_ind] = max(ncclist);

map(y+h, x+h) = m\_ind;

end

end

figure;

imagesc(map, [0 50]);

axis equal;

colormap("gray")

title("The Calculated Disparse Map")

saveas(gcf,"./output/depth.png");

pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Problem 2 KNN

load('data/train.txt')

load('data/test.txt')

x\_train = train(:,1:2);

y\_train = train(:,3);

x\_test = test(:, 1:2);

y\_test = test(:, 3);

K = [1 5 11 15];

for k = K

predict = KNN(x\_train, y\_train, x\_test, y\_test, k);

plot(x\_test(predict==y\_test & predict == 1,1), x\_test(predict==y\_test & predict == 1,2), 'b.')

axis equal;

hold on

plot(x\_test(predict==y\_test & predict == 2,1), x\_test(predict==y\_test & predict == 2,2), 'r.')

plot(x\_test(predict~=y\_test,1 ), x\_test(predict ~= y\_test,2), 'ko')

hold off

title(sprintf('KNN Result on K = %d',k),'FontSize',14)

saveas(gcf,sprintf("./output/k%d.png",k));

pause;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Helper Function

% return the value of NCC of two eqo\_sized image.

function ncc = calculateNCC(origin, template)

p\_mean = mean(origin, [1 2]);

t\_mean = mean(template, [1 2]);

p\_sigma = std(origin, 0, [1 2]);

t\_sigma = std(template, 0, [1 2]);

arr = ((origin - p\_mean).\*(template - t\_mean))./(p\_sigma.\*t\_sigma);

% prevent inf value from dividing 0

arr(p\_sigma \* t\_sigma == 0) = min(arr, [], 'all');

ncc = sum(sum(arr, [1 2])/(size(template,1)\*size(template,2)-1), 'all');

end

% return a column of predicted labels

function predict = KNN(x\_train, y\_train, x\_test, y\_test, k)

index = knnsearch(x\_train, x\_test, 'K', k);

predict = y\_train(index);

predict = mode(predict, 2);

accuracy = sum(predict == y\_test)/size(y\_test, 1);

fprintf('The accuracy at K = %u is: %.3f%%\n', k, accuracy\*100)

end