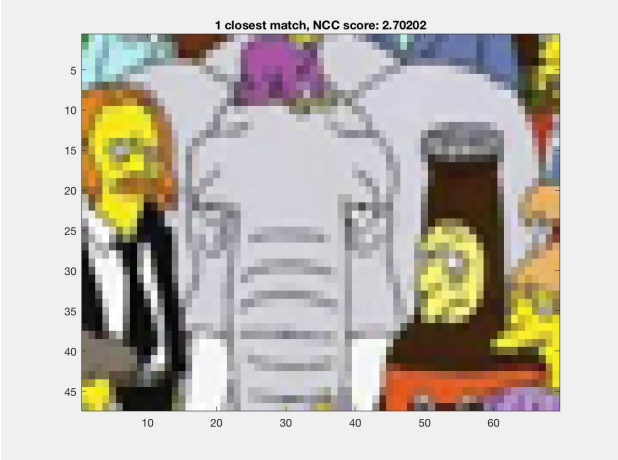
Computer Vision for HCI — Abhinav Mahalingam (.15)

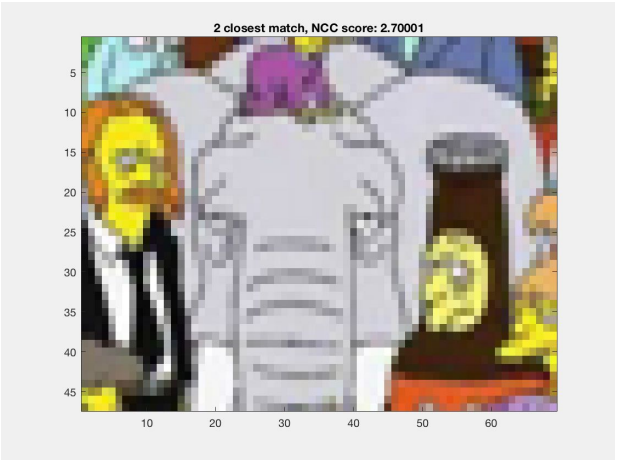
CSE 5524

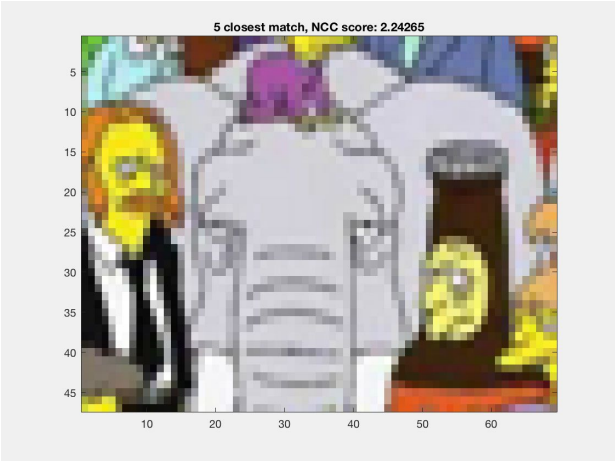
Homework #9

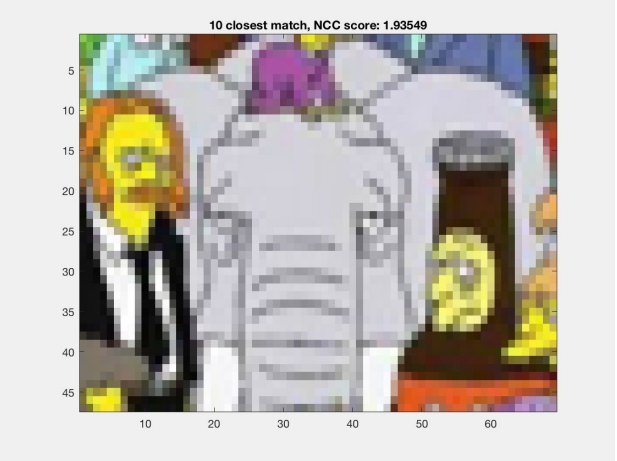
1. There's an elephant in the room. Can you find it? Search for the template template.png in the search image search.png using color-based NCC (make sure the standard deviation is "unbiased" with N-1). Assume the origin is in the center of the template image for each approach (Note: there should be a border around the search image where the metrics cannot be computed). Sort the resulting scores from best to worst. Plot the sorted scores and show the patches corresponding to the 1st, 2nd, 5th, 10th, 100th, and 500th closest matches. Compare the results. [3 pts]

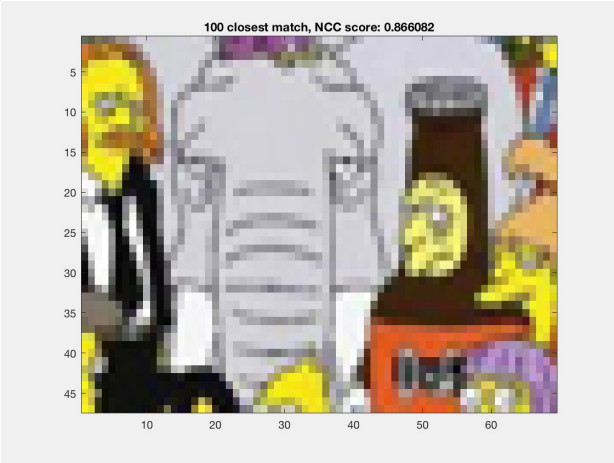
From the below images, we can observe that the top 10 closest images are very similar to the template. The 100th closest image while off by a few pixels is still relevant to the template image. The 500th closest image although is completely different from the given template, which could be because the image has similar color composition as the template.

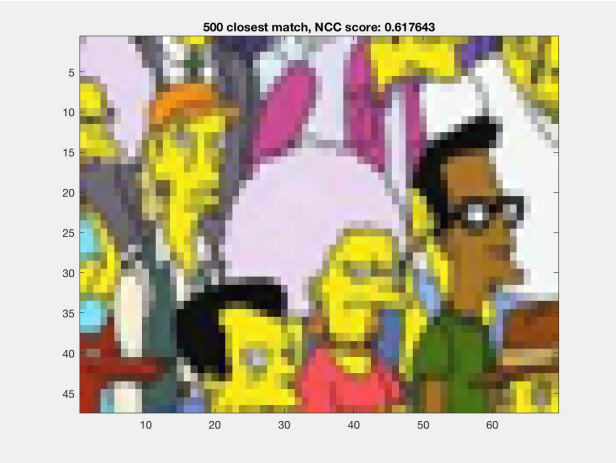


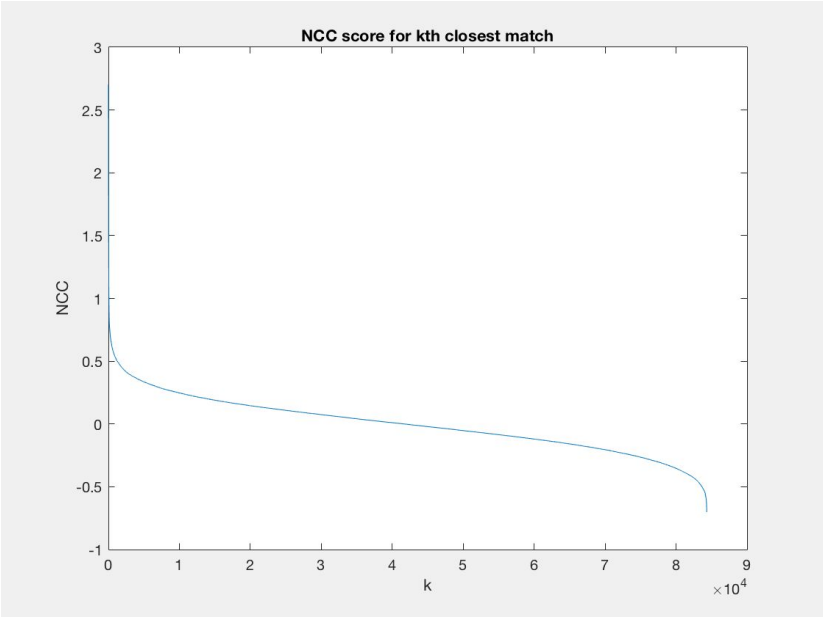






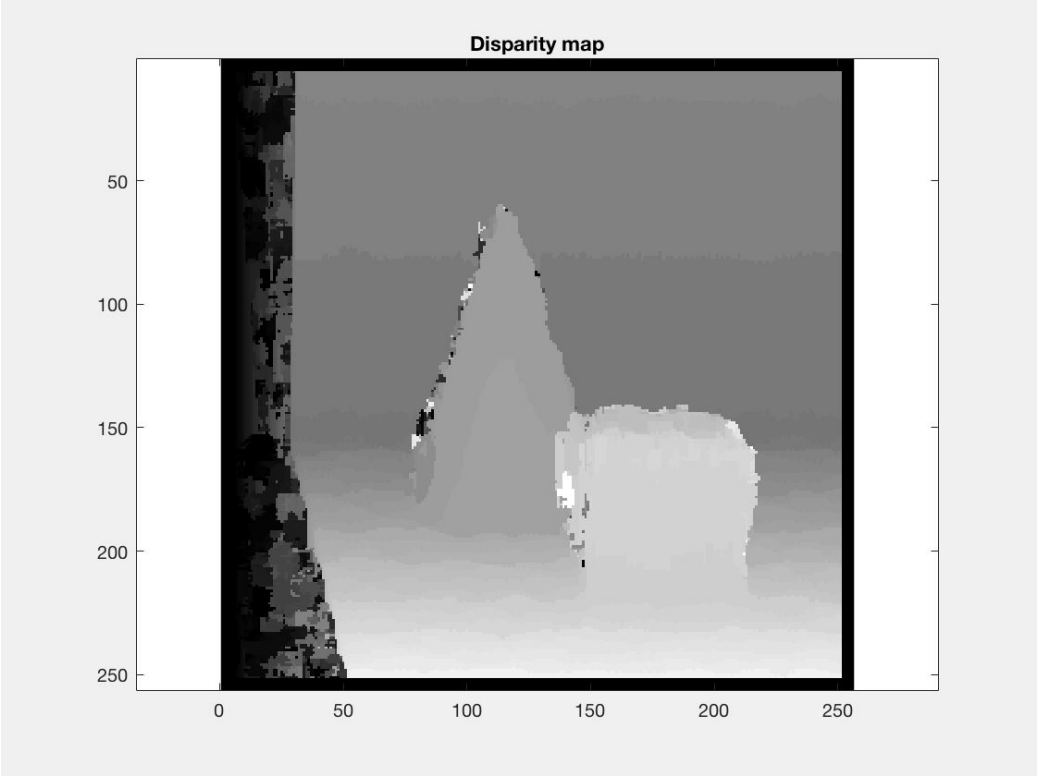




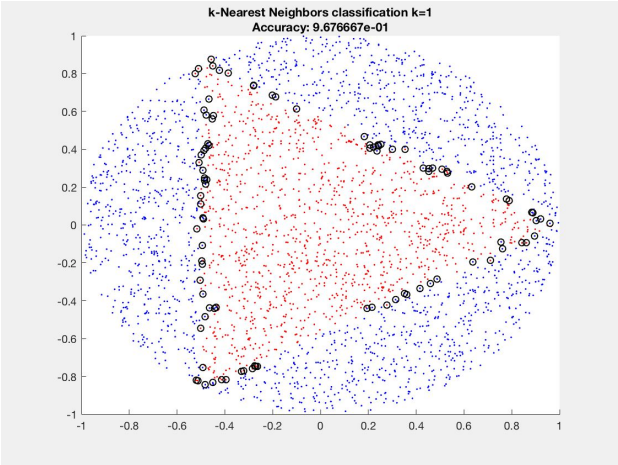


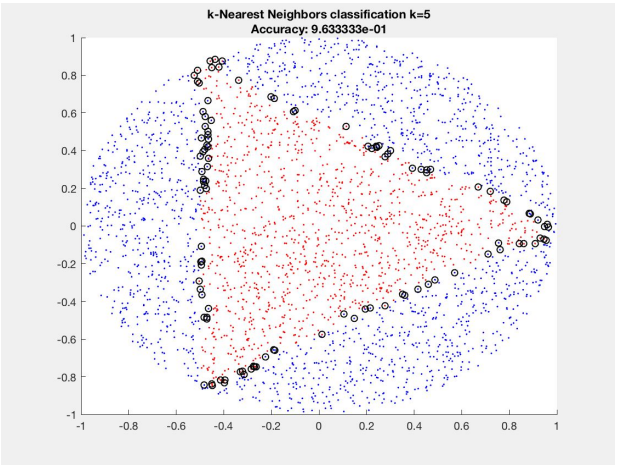
2. Compute a disparity map for the images left.png and right.png (having parallel optical axes) using the basic stereo matching algorithm. Use your NCC function to perform the template matching for each patch in the left image searching in the right image (search only leftward from the starting point along each row!), and use a window size of 11x11 pixels. To make things run a bit faster, when searching leftward, only move up to 50 pixels to the left (instead of to the edge of the image). Use the following code to display the disparity map D with a gray colormap and clip the disparity values at 50 pixels. [5 pts]

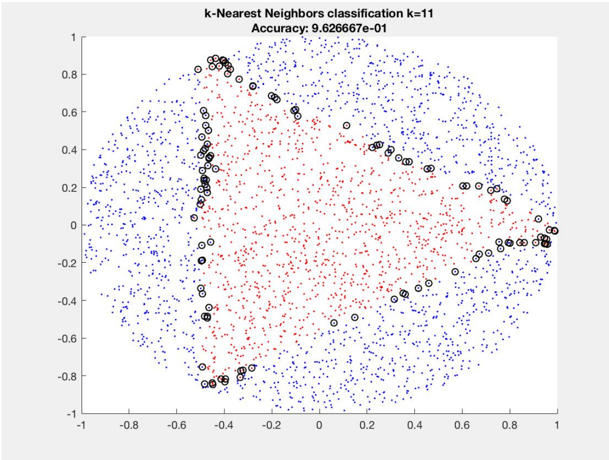
   ```
   figure;
   imagesc(D, [0 50]); axis equal;
   colormap gray;
   ```

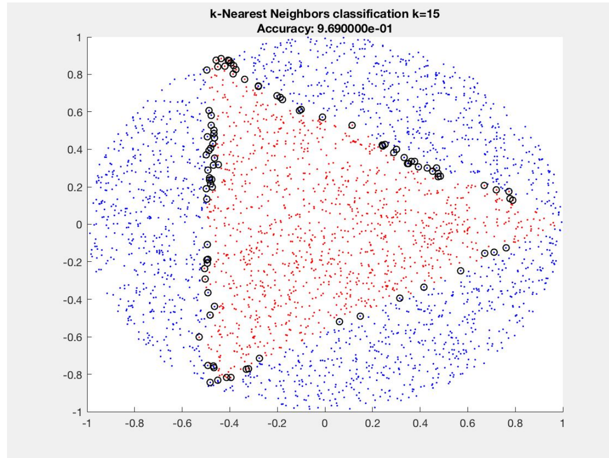


3. Write an implementation of the simple k-Nearest Neighbors (kNN) algorithm to classify data points.Use the points in file *train.txt* as training data (this file contains 1 row for each data point where the first two columns are x,y coordinates and the third column is the ground truth classification label). Classify all the test data points in the file *test.txt* (formatted in the same way) using *K*=1. Calculate and report the accuracy of your algorithm (compared to the third column ground truth of the test data). Plot the test data points, color coded by the class label your algorithm gives (use plot() options 'r.' and 'b.'). On the same figure (use hold on/off), (re)plot the points which are misclassified (use plot() option 'ko'). Repeat this for *K*=5, 11, and 15. Compare the results for different values of *K*. [4 pts]









It can be observed that only the points along the border of the triangle are misclassified, which can be attributed to the fact that they are surrounded by points of both classes. It can also be seen that as the value of k increases, there is no clear pattern in the change in accuracy.

% Abhinav Mahalingam
% CSE5524 - HW 9
% 10/28/2017

HW9.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 1
matches = [1 2 5 10 100 500];
Im = double(imread('input/search.png'));
template = double(imread('input/template.png'));
[temp_height, temp_width, ~] = size(template);
[ncc_result] = ncc_template_matching(Im, template);
for n=matches
   match = ncc_result(n, :);
   x =  match(1, 1);
   y =  match(1, 2);
   patch = Im(y:y+temp_height-1, x:x+temp_width-1, :);
   figure;
   imagesc(patch/255);
   title(sprintf('%d closest match, NCC score: %g', n, match(1, 3)));
   saveFrame(sprintf('results/ncc_%d_closest.png', n));
end
figure;
plot(ncc_result(:, 3));
```

```
xlabel('k');
ylabel('NCC');
title('NCC score for kth closest match');
saveFrame('results/ncc_plot.png');
pause;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 2
clear; close all; clc;
left_image = double(imread('input/left.png'));
right_image = double(imread('input/right.png'));
disparity_map = get_disparity_map(left_image, right_image, 11, 11);
figure;
imagesc(disparity_map, [0 50]);
axis equal;
colormap gray;
title('Disparity map');
saveFrame('results/disparity_map.png');
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 3
clear; clc; close all;
training_data = load ('input/train.txt');
test_data = load ('input/test.txt');
train_coords = training_data(:, 1:2);
test_coords = test_data(:, 1:2);
test_label = test_data(:, 3);
k_values = [1, 5, 11, 15];
for k=k_values
   [idx, D] = knnsearch(train_coords, test_coords, 'K', k);
   k_labels = zeros(size(test_coords, 1), k);
   for i=1:k
      k_labels(:, i) = training_data(idx(:,i), 3);
   end
   if k ~= 1
      classified_labels = mode(k_labels,2);
   else
```

```
    classified_labels = k_labels;
  end
  misclassification = test_data(test_label ~= classified_labels, 1:2);
  accuracy = (size(test_data,1) - size(misclassification,1))/ size(test_data,1);
  class_1_matches = test_data(classified_labels == 1, 1:2);
  class_2_matches = test_data(classified_labels == 2, 1:2);
  figure;
  hold on;
  title({sprintf('k-Nearest Neighbors classification k=%d', k), sprintf('Accuracy: %d', accuracy)});
  plot(class_1_matches(:, 1), class_1_matches(:, 2), 'r.');
  plot(class_2_matches(:, 1), class_2_matches(:, 2), 'b.');
  plot(misclassification(:, 1), misclassification(:, 2), 'ko');
  hold off;
  saveFrame(sprintf('results/k_%d_plot.png',k));
  disp(accuracy);
end
pause;
close all
```

get_disparity_map.m

```
function [disp_map] = get_disparity_map(left_image, right_image, window_width, window_height)
  [height, width] = size(left_image);
  win_half_width = floor(window_width/2);
  win_half_height = floor(window_height/2);
  disp_map = zeros(height, width);
  for x=ceil(window_width/2):(width - win_half_width)
    for y=ceil(window_height/2):(height - win_half_height)
                template = left_image ( y - win_half_height : y + win_half_height , x - win_half_width:x+win_half_width);
          search_Im = right_image ( y - win_half_height : y + win_half_height , max ( 1 , x - win_half_width -50):x+win_half_width);
        [ncc_result] = ncc_template_matching(search_Im, template);
        match_x = ncc_result(1, 1) + max(win_half_width,x-51);
        disp_map(y,x) = (x-match_x);
    end
  end
end
```

ncc_template_matching.m

```
function [result] = ncc_template_matching(Im, template)
   [temp_height, temp_width, ~] = size(template);
   [Im_height, Im_width, ~] = size(Im);
   temp_width_half_span = floor(temp_width/2);
   temp_height_half_span = floor(temp_height/2);
   temp_center_x = ceil(temp_width/2);
   temp_center_y = ceil(temp_height/2);
   ncc_values = zeros(Im_height, Im_width);

   for x=temp_center_x:(Im_width-temp_width_half_span)
      for y=temp_center_y:(Im_height-temp_height_half_span)
                  window = Im ( y - temp_height_half_span : y+temp_height_half_span , x -
temp_width_half_span : x + temp_width_half_span, :);
         for j=1:size(template, 3)
            template_color = template(:, :, j);
            template_avg = mean(template_color(:));
            window_color = window(:, :, j);
            window_avg = mean(window_color(:));
                              ncc_values(y,x) = ncc_values(y,x) + sum(sum((window_color -
window_avg).*(template_color                                                    -
template_avg)/(std(window_color(:))*std(template_color(:))))))/(temp_height*temp_width - 1);
         end
      end
   end
     ncc_wo_edges = ncc_values ( temp_center_y : ( Im_height - temp_height_half_span ) ,
temp_center_x : (Im_width-temp_width_half_span));
   [sorted_values, sorted_index] = sort(ncc_wo_edges(:), 'descend');
   [y, x] = ind2sub(size(ncc_wo_edges), sorted_index);
   result = [x y sorted_values];
end
```

saveFrame.m

```
function[] = saveFrame(fileName)
   fig = gcf;
   frame = getframe(fig);
   imwrite(frame.cdata,fileName);
end
```