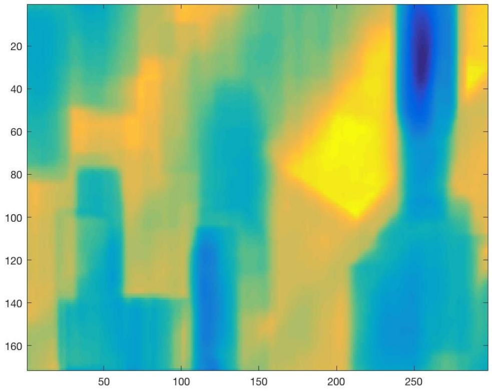
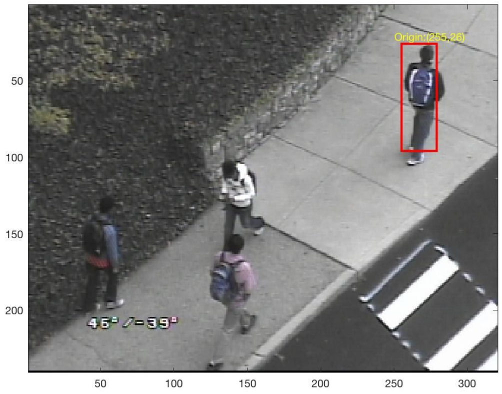
Computer Vision for HCI Abhinav Mahalingam (.15)
CSE 5524
Homework #6

1. Use the covariance matching technique to find the correct match in the color image given on the WWW site (target.jpg). The model covariance matrix (of <x,y,R,G,B> features) is given below.
   modelCovMatrix = [47.917 0 -146.636 -141.572 -123.269; 0 408.250 68.487 69.828 53.479; -146.636 68.487 2654.285 2621.672 2440.381; -141.572 69.828 2621.672 2597.818 2435.368; -123.269 53.479 2440.381 2435.368 2404.923];
   Test all possible 1-pixel overlapping windows (each of size 70 rows by 24 columns, with the upperleft-corner as the window origin) in the image with the given model. Save the match distance for each box location in the image at each pixel location (for the origin of the window). Plot/display the match-distance-image. Provide the location of the best match distance for the best candidate. Note that the above given covariance matrix is biased (normalized with 1/(M*N)), and Matlab's cov function is unbiased by default using 1/(M*N-1), so call cov(X, 1) to make it consistent. Leave the image with colors ranging 0-255 (do not scale/normalize the colors). [5 pts]



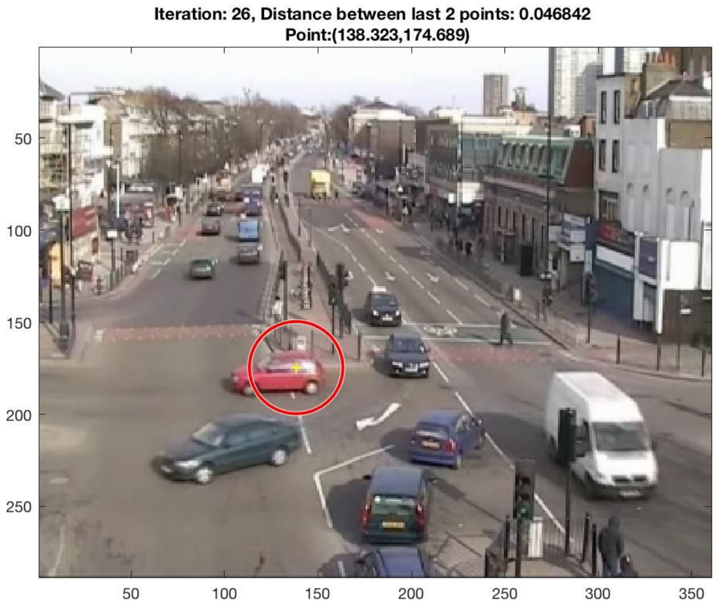


The image on the right side shows the region (origin at 255,26) that is closest to covariance matrix given in the question. The left side image is a plot of covariance distance at each pixel of the image. We can see that the closest point is in the same pixel as the origin of the window in the right image.

2. Create a function to extract a feature vector for each pixel in a circular neighborhood (< radius) around (*x*,*y*): [X]=circularNeighbors(img, x, y, radius) ;For each pixel, use the same format to return as used above (<x_i,y_i,R,G,B>). That is, *X* should be a *K*x5 matrix, where each row is for one of the pixels in the neighborhood. Assume that the (x,y) passed into the function are

real (non-integer) values, and do NOT round them in the function for computation of the neighborhood. [2 pts]

3. Create a function to build a color histogram from a neighborhood of points: [hist]=colorHistogram(X, bins, x, y, h); The histogram (*hist*) should be a *bins* x *bins* x *bins* color cube (RxGxB). The bins should be evenly spaced. For this assignment, use *bins*=16 then the pixel-value limits for each bin will be {0-15, 16-31, ..., 240-255}. Weight the construction of the histogram using an Epanechnikov kernel centered at real-valued (*x*, *y*) and with bandwidth *h*. Normalize the histogram/cube so it sums to 1. (This function will be used to make your model histogram “q_model” and to make the candidate test histogram “p_test”) [3 pts]
4. Create a function to calculate a vector of the mean-shift weights (w), a weight w_i for each pixel *i* in the neighborhood: [w]=meanshiftWeights(X, q_model, p_test, bins);[2 pts]
5. Load the images img1.jpg and img2.jpg, and use the functions above to perform mean-shift tracking. Build a model from img1 using a circular neighborhood with a radius of 25 pixels centered at (x_0,y_0) = (150.0, 175.0) and a color histogram of size 16x16x16 (cube). Build the weighted cube histogram using an Epanechnikov kernel with bandwidth *h* = 25 (same as the earlier radius). Run 25 iterations of mean-shift tracking on img2. DO NOT ROUND coordinates or values at any time! Report the final (*x*, *y*) location (DO NOT ROUND) and Euclidean distance between the last two iterations (see Step 4 on the Algorithm slide). [3 pts]



The center pixel stabilizes around (138.23, 174.69) and the euclidean distance between the last two iterations is 0.046842.

```
% Abhinav Mahalingam
% CSE5524 - HW6
% 10/1/2017
% HW6.m
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Problem 1
Im = double(imread('input/target.jpg'));
image_size = size(Im);
covariance_model = [47.917 0 -146.636 -141.572 -123.269; 0 408.250 68.487 69.828 53.479;
-146.636 68.487 2654.285 2621.672 2440.381; -141.572 69.828 2621.672 2597.818 2435.368;
-123.269 53.479 2440.381 2435.368 2404.923];
window_size = [70 24];
final_row = image_size(1) - window_size(1) + 1;
final_col = image_size(2) - window_size(2) + 1;
cov_diff_mat = zeros(final_row,final_col);
for x=1:final_col
   for y=1:final_row
      feature_matrix = generate_feature_matrix(Im,x,y,window_size(2),window_size(1));
      covariance_candidate=cov(feature_matrix,1);
      cov_diff_mat(y,x) = manifold_distance(covariance_model,covariance_candidate);
   end
end
[min_num,min_idx] = min(cov_diff_mat(:));
[min_row,min_col] = ind2sub(size(cov_diff_mat),min_idx);
imagesc(cov_diff_mat);
pause;
imagesc(Im/255);
hold on;
rectangle('Position',[min_col min_row window_size(2)
window_size(1)],'EdgeColor','Red','LineWidth',2);
text(min_col-5,min_row-5,sprintf('Origin:(%g,%g)',min_col,min_row),'Color','Yellow');
hold off;
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 2
clc;close all;
img = double(imread('input/target.jpg'));
center_x = 300;
```

```
center_y = 30;
radius = 15;
feature_matrix = circularNeighbors(img, center_x, center_y, radius);

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 3
clc;close all;
h=20;
bins = 16;
hist_cube = colorHistogram(feature_matrix, bins, center_x, center_y, h);
q_model = hist_cube;
p_test = hist_cube;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 4
weights = meanShiftWeights(feature_matrix, q_model, p_test);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 5
clc; clear; close all;
img1 = double(imread('input/img1.jpg'));
img2 = double(imread('input/img2.jpg'));
radius = 25;
h = 25;
center_x = 150.0;
center_y = 175.0;
bins = 16;
number_of_iterations = 25;
original_feature_matrix = circularNeighbors(img1, center_x, center_y, radius);
q_model = colorHistogram(original_feature_matrix,bins,center_x, center_y, h);
imagesc(img1/255);
hold on;
viscircles([center_x center_y],radius);
plot(center_x, center_y, 'y+', 'MarkerSize', 5);
hold off;
x = center_x;
y = center_y;
coordinates = [x y];
for n=1:number_of_iterations
```

```
   feature_matrix = circularNeighbors(img2, x, y, radius);
   p_test = colorHistogram(feature_matrix,bins,x,y,h);
   weights = meanShiftWeights(feature_matrix, q_model, p_test);
   sum_of_weights = sum(weights);
        new_coordinate = sum([feature_matrix(:,1).*weights',feature_matrix(:,2).*weights'],1)./
sum_of_weights;
   x = new_coordinate(1);
   y = new_coordinate(2);
   coordinates = [coordinates; x y];
   fprintf('New coordinate: (%g, %g)\n',x,y);
end
for n=1:size(coordinates,1)
   imagesc(img2/255);
   hold on;
   viscircles([coordinates(n,1) coordinates(n,2)], radius);
   plot(coordinates(n,1),coordinates(n,2),'yellow+','MarkerSize',5);
   if n>1
          dis=sqrt((coordinates(n,1) - coordinates(n-1,1)).^2 + (coordinates(n,2) -
coordinates(n-1,2) ). ^2) ;
  title(sprintf('Iteration: %d, Distance between last 2 points: %0.5g \n Point:(%g,%g)',
n,dis,coordinates(n,1),coordinates(n,2)));
   end
   hold off;
   pause();
end
```

circularNeighbors.m

```
function[feature_matrix] = circularNeighbors(Im, center_x, center_y, radius)
cols = (2*radius)+1;
rows = (2*radius)+1;
relative_center_x = radius + 1 + (center_x-floor(center_x));
relative_center_y = radius + 1 + (center_y-floor(center_y));
[X, Y] = meshgrid(1:cols,1:rows);
circle_points = ((X - relative_center_x).^2 + (Y - relative_center_y).^2) < radius^2;
box_feature_matrix = generate_feature_matrix(Im, center_x-(radius+1), center_y-(radius+1),
cols, rows);
transposed_circle_points = circle_points';
vectorized_circle_points = repmat(transposed_circle_points(:), 1, 5);
```

```
box_feature_matrix=box_feature_matrix.*vectorized_circle_points;
bounding_box_coords = box_feature_matrix(:,1:2);
feature_matrix = box_feature_matrix(all(bounding_box_coords, 2), :);
feature_matrix(:, 1) = feature_matrix(:, 1) + center_x - relative_center_x;
feature_matrix(:, 2) = feature_matrix(:, 2) + center_y - relative_center_y;
end
```

colorHistogram.m

```
function[hist] = colorHistogram(feature_matrix, bins, center_x, center_y, h)
min_x = min(feature_matrix(:,1));
min_y = min(feature_matrix(:,2));
max_x = max(feature_matrix(:,1));
max_y = max(feature_matrix(:,2));
% Epanechnikov kernel
[X, Y] = meshgrid(min_x:max_x,min_y:max_y);
kernel = 1 - (sqrt((X-center_x).^2 + (Y-center_y).^2)./h).^2;
kernel(kernel < 0) = 0;
kernel_vectors = [];
for col = 1:size(X,2)
   kernel_vectors = [kernel_vectors; X(:, col) Y(:,col) kernel(:,col)];
end
sorted_feature_matrix = sortrows(feature_matrix, [1 2]);
filtered_kernel = kernel_vectors(ismember(kernel_vectors(:, 1:2),sorted_feature_matrix(:,1:2), 'rows' ),:);
filtered_feature_matrix = sorted_feature_matrix(ismember(sorted_feature_matrix(:,1:2), filtered_kernel(:,1:2), 'rows'),:);

weighted_hist_vector(:, 1) = floor(filtered_feature_matrix(:,3)*bins/256) + 1;
weighted_hist_vector(:, 2) = floor(filtered_feature_matrix(:,4)*bins/256) + 1;
weighted_hist_vector(:, 3) = floor(filtered_feature_matrix(:,5)*bins/256) + 1;
weighted_hist_vector(:, 4) = filtered_kernel(:,3);

all_cube_indices(:,1) = repmat((1:bins)', bins*bins, 1);
all_cube_indices(:,2) = repmat(for_each(1:bins,bins)', bins, 1);
all_cube_indices(:,3) = for_each(1:bins,bins*bins)';
all_cube_indices(:,4) = zeros(bins*bins*bins, 1);

weighted_hist_vector = sortrows(weighted_hist_vector,[3 2 1]);
[unique_hist_rows,~,groupings] = unique(weighted_hist_vector(:, 1:3),'rows', 'stable');
```

```
summed_hist_vector = [unique_hist_rows, accumarray(groupings, weighted_hist_vector(:,4))];
cubes_indices_present = ismember(all_cube_indices(:,1:3), summed_hist_vector(:,1:3),'rows');
all_cube_indices(cubes_indices_present,4) = summed_hist_vector(:,4);
hist_cube = reshape(all_cube_indices(:,4),bins,bins,bins);
hist = hist_cube./sum(sum(sum(hist_cube,3),2),1);
end
```

for_each.m

```
function [repeated_vector]=for_each(row_vector,times_to_repeat)
   repeated_rows=repmat(row_vector,times_to_repeat,1);
   repeated_vector=repeated_rows(:)';
End
```

generate_feature_matrix.m

```
function[feature_matrix] = generate_feature_matrix(img,origin_x,origin_y,cols,rows)
   window=img(floor(origin_y):(floor(origin_y)+rows-1),floor(origin_x):(floor(origin_x)+cols-1),:);
   feature_matrix(:,1)=repmat(1:cols,1,rows);
   row_val=repmat(1:rows,cols,1);
   feature_matrix(:,2) = row_val(:);
   feature_matrix(:,3)=reshape(window(:,:,1)',[],1);
   feature_matrix(:,4)=reshape(window(:,:,2)',[],1);
   feature_matrix(:,5)=reshape(window(:,:,3)',[],1);
end
```

manifold_distance.m

```
function[distance] = manifold_distance(cov_a,cov_b)
   [~,eig_val]=eig(cov_a,cov_b);
   eig_val = eig_val(eig_val~=0);
   distance = sqrt(sum(log(eig_val).^2));
end
```

meanShiftWeights.m

```
function[weights] = meanShiftWeights(feature_matrix,q_model,p_test)
bins=size(q_model,1);
weights =zeros(1, size(feature_matrix,1));
for n=1:size(feature_matrix,1)
   bin_index=floor(feature_matrix(n,3:5)./bins)+1;
weights(1,n)=sqrt(q_model(bin_index(1),bin_index(2),bin_index(3))/p_test(bin_index(1),bin_index(2),bin_index(3)));
end
end
```