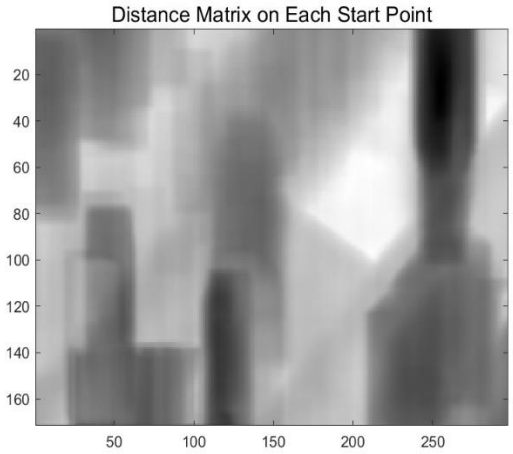
Yukun Duan
Computer Vision for HCI AU'22
Homework Assignment #6

1.
The covariance tracking runs pretty fast and the result is focusing on this guy below. The location of the best match is at the position: [255,26], with covariance of 0.7061.
The distance matrix indicates that the right top corner has a dark region, which has the lowest distance. The index of the darkest point is around [20, 200] which matches our result.
When we compare the distance matrix with the original image, there are several dark regions on the distance matrix. When we find these corresponding dark regions on the origin image, we found similar color composition to our result.



5.
The model with circle on location [150.0, 175.0] is shown below:

Model Image w/ Circle at the Specified Location.

The final location is: [137.650896, 175.281092], with distance: 0.011138. Image is below:

Result Image w/ Circle at the Calculated Location.

The red circle is focused on the body of the red car, which was shifted from x=150 to x=137. Zoom in the image and we can see the element in both circles are almost identical. The red

circle crosses the front tire, covering the garbage can and the edge of the roads, which quite makes sense. The final distance is 0.011138 which indicates the convergence of iteration. I also did experiments with iterations = 100, and the final result is around [137.6374, 175.2953] with distance = 1.39*e-12, which is almost zero. I rounded the result to [138, 175] and drew the circle with radius = 25, as figure above.

Attached Code:

```
% Yukun Duan
% CSE5524 - HW6
% 10/02/2022


%% Problem 1
% import raw image in double
raw = double(imread("target.jpg"));
xind = double(repmat(1:size(raw,2),size(raw,1),1)); % col => x
yind = double(repmat((1:size(raw,1))', 1, size(raw,2))); % row => y
target = cat(3,xind,yind,raw); % concate x index and y index to image

% import model cov matrix
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];

% iteratively test all 1-pixel overlapping windows (70*24)
res = [1,1,realmax('single')];
match_dist = zeros([raw(1) raw(2)]);
for r = 1:(240-70+1)
    for c = 1:(320-24+1)
        cm = covMatrix(target, r, c, 70, 24);
        dm = distMatrix(cm, modelCovMatrix);
        match_dist(r,c) = dm;
        if dm <= res(3)
            res = [r, c, dm];
        end
    end
end

x = res(1);
y = res(2);
disp(res)
imagesc(match_dist)
```

```
colormap('gray')
title('Distance Matrix on Each Start Point','FontSize',14)
saveas(gcf,'Part1_Dist.jpg')
pause;
resim = double(imread("target.jpg"));
resim = resim(res(1):res(1)+69, res(2):res(2)+23,:);
imshow(resim/255, 'InitialMagnification','fit')
title('Q1: Tracked Person in Covariance Tracking')
saveas(gcf,'Part1.jpg')
pause;



%% Problem 2
% Functions are below

%% Problem 3
% Functions are below

%% Problem 4
% Functions are below

%% Problem 5
im1 = double(imread("img1.jpg"));
im2 = double(imread("img2.jpg"));
location = [150.0 175.0]; % (x, y)
previous_loc = location;
X_model = circularNeighbors(im1, 175.0, 150.0, 25.0);
q_model = colorHistogram(X_model, 16, 175.0, 150.0, 25.0);
for i = 1:25
    X_test = circularNeighbors(im2, location(2), location(1), 25.0);
    p_test = colorHistogram(X_test, 16, location(2), location(1), 25.0);
    % update parameter
    weight = meanshiftWeights(X_test, q_model, p_test, 16);
    location = weightedAverage(X_test, weight);
    % display distance and location
    dist = pdist([previous_loc; location], 'euclidean');
    fprintf(sprintf('Location: %f, %f. Dist: %f \n', location, dist))
    previous_loc = location;
end

imshow(im1/255, 'InitialMagnification','fit')
viscircles([150 175], 25)
title('Model Image w/ Circle at the Specified Location.', 'FontSize',14)
```

```
saveas(gcf,'Model.jpg')
pause;
imshow(im2/255, 'InitialMagnification','fit')
viscircles(round(location), 25)
title('Result Image w/ Circle at the Calculated Location.', 'FontSize',14)
saveas(gcf,'Test.jpg')
```

```
%% Problem 1 function
% Return distance matrix of model and test window
function dm = distMatrix(m1, m2)
    e = eig(m1, m2);
    e(e <= 0) = 1; % convert log(0 or neg) => log(1) = 0
    dm = sqrt(sum(log(e).^2, 'all'));
end

% Return the covariance matrix of fk, with start point (row,col) and
size(rl, cl)
function cm = covMatrix(fk,row,col,rl,cl)
    window = fk(row:(row+rl-1), col:(col+cl-1), :);
    window = reshape(window, [], size(fk, 3));
    cm = cov(window, 1);
end

%% Problem 2 function
% Return the feature vector of all pixels in circular (dim = ?*5)
% Rows in X: [x(col), y(row), r, g, b]
function X = circularNeighbors(img, r, c, radius)
    X = [];
    for i = 1:size(img,2) % x, column
        for j = 1:size(img,1) % y, row
            if pdist([i j; c r], 'euclidean') <= radius
                line = double([i, j, reshape(img(j, i, :), 1, 3)]);
                X = cat(1, line, X);
            end
        end
    end
end

%% Problem 3 function
% Return a cube (dim = bins*bins*bins) histogram of X.
```

```
function hist = colorHistogram(X, bins, x, y, h)
    hist = zeros([bins bins bins]);
    interval = 256.0/bins; % [0*gap - 1*gap), [1*gap - 2*gap), ...
    for i = 1:size(X, 1)
        dist = pdist([X(i, 2) X(i, 1); x y], 'euclidean');
        k = max(0, 1-(dist/h)^2); % weight
        r = floor(X(i, 3)/interval)+1;
        g = floor(X(i, 4)/interval)+1;
        b = floor(X(i, 5)/interval)+1;
        hist(r,g,b) = hist(r,g,b) + k;
    end
    hist = hist / sum(hist, 'all');
end


%% Problem 4 function
% Return the weights vector.
function weight = meanshiftWeights(X, q_model, p_test, bins)
    weight = zeros([1, size(X,1)]);
    interval = 256/bins; % [0*gap - 1*gap), [1*gap - 2*gap), ...
    p_test(p_test<0) = 1.0; % prevent zero dividing
    q_model(p_test<0);
    qube_weighted = sqrt(q_model./p_test);
    for i = 1:size(X, 1)
        r = floor(X(i, 3)/interval)+1;
        g = floor(X(i, 4)/interval)+1;
        b = floor(X(i, 5)/interval)+1;
        weight(i) = weight(i) + qube_weighted(r,g,b);
    end
end

%% Problem 5 function
% Return the next best location [x(col), y(row)].
function location = weightedAverage(X, weight)
    ind = X(:, 1:2).* weight';
    location = sum(ind) /sum(weight, 'all');
end
```