CLab-2 Report

Jeff Yuanbo Han u6617017

April 3, 2018

Contents

1 Harris Corner Detector

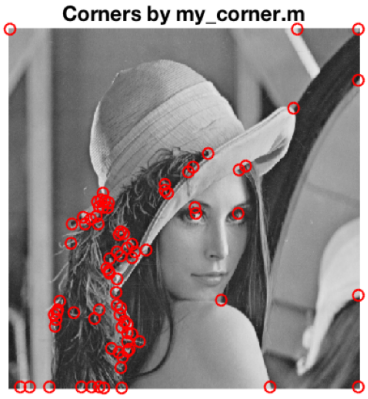
Five images are tested on both my detector and the built-in *corner*() function. Results are displayed in pairs below (Figure 1). Performances for *my_corner*() are in the **left** column. As we can see, the results from different functions are quite similar.

During experiment, my setting of threshold for each picture is:

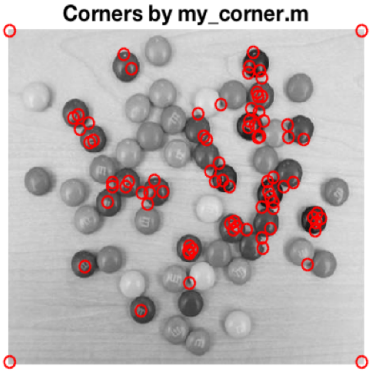
Image	thresh
Lenna.png	0.08
mandm.png	0.01
peppers.png	0.01
Right.jpg	0.03
scene.pgm	0.1

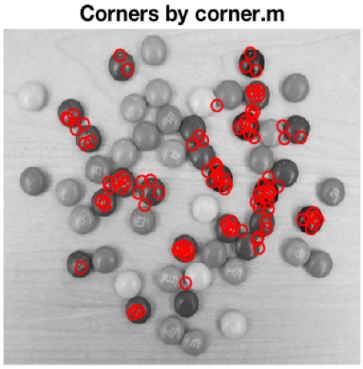
2 K-Means Clustering

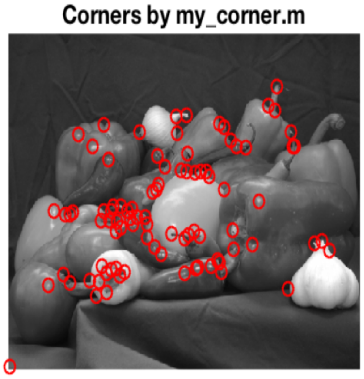
Two images are implemented k-means clustering in this section. The number of clusters is set as 5 from beginning to end. The result from my function is the **top** one of the two (see Figure 2). Note that when implementing k-means algorithm, results are not deterministic due to the random initialization. However, results with *kmeans*() do not differ very much from those with built-in *kmeans*().

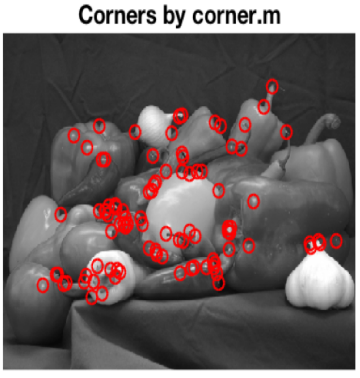

Corners by my_corner.m


Corners by corner.m


Corners by my_corner.m


Corners by corner.m


Corners by my_corner.m


Corners by corner.m

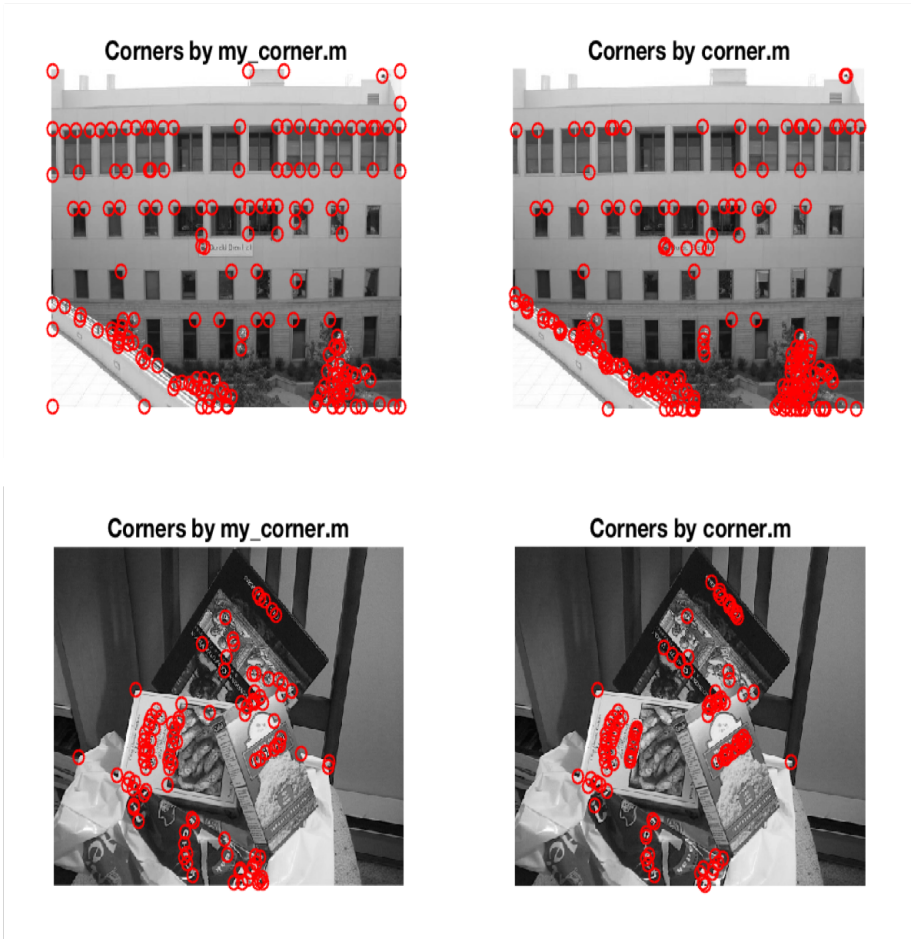
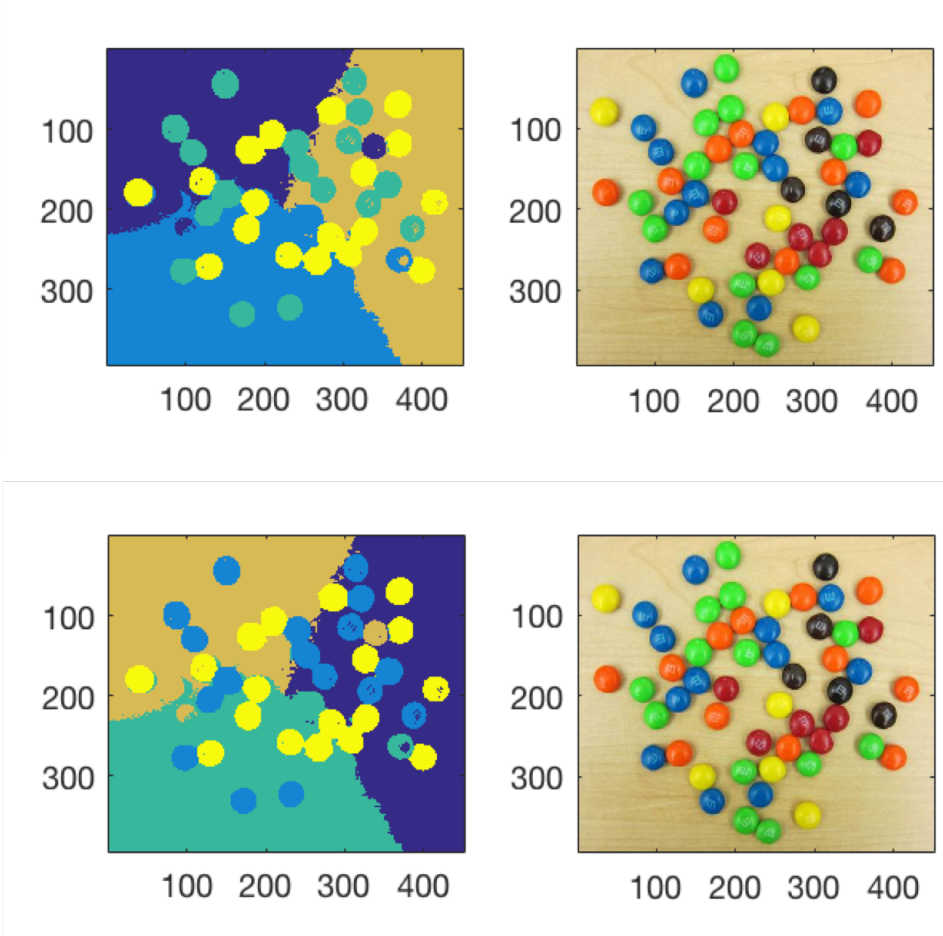


Figure 1: Harris Corner


100
200
300
100 200 300 400
100
200
300
100 200 300 400
100
200
300
100 200 300 400
100
200
300
100 200 300 400

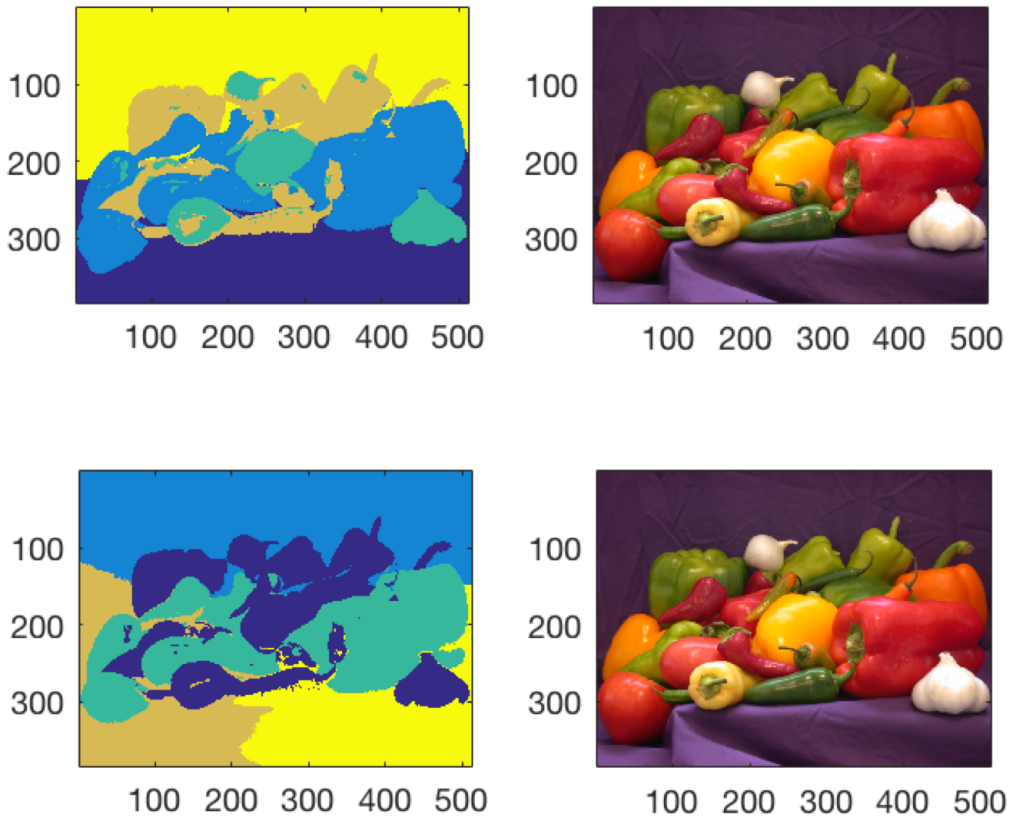


Figure 2: K-Means Clustering

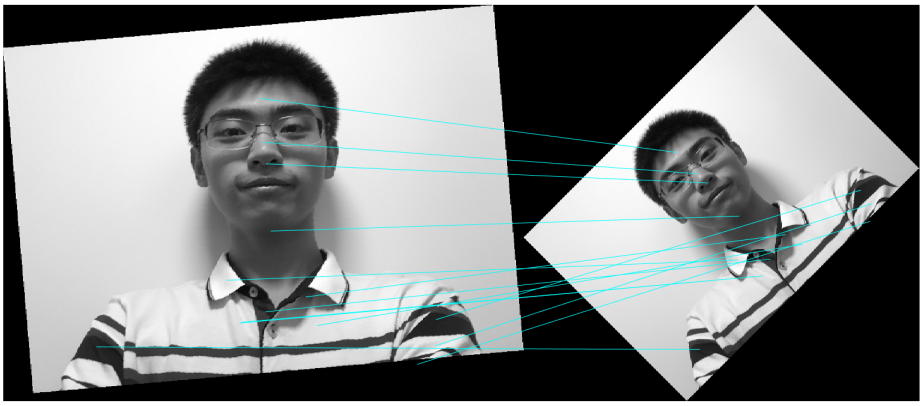


Figure 3: SIFT Feature Match

3 SIFT Feature Match

I write the script *siftMatch.m* to perform this task. 280 matches in total between these two pictures are found. The result is shown in Figure 3, where only 15 matching points are connected in order to avoid cluttered visualization.

A MATLAB Codes

A.1 my_corner.m

```
function [corners, count] = my_corner(bw, thresh)
% MY_CORNER computes Harris Corners of an image.
% Input:
%   bw: a grayscale image matrix
%   thresh: threshold
%
% Output:
%   corners: a matrix of the same size as bw, which records the Harris
%            Corners (corner's value == 1, otherwise == 0)
%   count: The number of corners obtained
%
% By Jeff Yuanbo Han (u6617017), 2018-04-02.

sigma = 2;  % the sigma for Gaussian filter
count = 0;  % the number of detected corners

```

```
% Gaussian first partial derivatives
dy = [-1 0 1;-1 0 1;-1 0 1];
dx = dy';

% Gaussian first gradient of the intensity
Ix = conv2(bw,dx,'same');
Iy = conv2(bw,dy,'same');

% Generate a Gaussian kernel.
g = fspecial('gaussian',max(1,fix(6*sigma)),sigma);

% Gaussian second gradient of the intensity
Ix2 = conv2(Ix.^2,g,'same');
Iy2 = conv2(Iy.^2,g,'same');
Ixy = conv2(Ix.*Iy,g,'same');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Task: Compute the Harris Cornerness R
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[height, width] = size(bw);  % Get the size of bw.
R = zeros(height, width);  % Harris Cornerness
Rmax = 0;  % The maximum value of R

for i = 1:height
for j = 1:width
M = [Ix2(i,j) Ixy(i,j); Ixy(i,j) Iy2(i,j)];  % M matrix
R(i,j) = det(M) - 0.04*trace(M)^2;  % Compute R value.
if R(i,j)> Rmax
Rmax = R(i,j);  % Update Rmax.
end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Task: Perform non-maximum suppression and threshold here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
corners = zeros(height, width);  % Record the corners (value will be 1).
for i = 2:height-1
for j = 2:width-1
% threshold and non-maximum
if R(i,j) > thresh*Rmax ...
&& R(i,j) > R(i-1,j-1) && R(i,j) > R(i-1,j) ...
&& R(i,j) > R(i-1,j+1) && R(i,j) > R(i,j-1) ...
&& R(i,j) > R(i,j+1) && R(i,j) > R(i+1,j-1) ...
&& R(i,j) > R(i+1,j) && R(i,j) > R(i+1,j+1)
corners(i,j) = 1;
```

```
count = count + 1;
end
end
end

end
```

A.2 my_kmeans.m

```
function [data_clusters, cluster_stats] = my_kmeans(data, nc)
% Performs k-means clustering on data, given (nc) = the number of clusters.
% Euclidean distance is used in this function.

% Random Initialization
[ndata, ndims] = size(data);

random_labels = floor(rand(ndata,1) * nc) + 1;

data_clusters = random_labels;

cluster_stats = zeros(nc,ndims+1);

distances = zeros(ndata,1);  % Distance to the assigned cluster center.

while(1)
pause(0.03);

% Make a copy of cluster statistics for comparison purposes.
% If the difference is very small, the while loop will exit.
last_clusters = cluster_stats;

% For each cluster,
for c = 1:nc
% Find all data points assigned to this cluster.
[ind] = find(data_clusters == c);
num_assigned = size(ind,1);

% Some heuristic codes for exception handling.
if( num_assigned < 1 )
disp(['No points were assigned to this cluster,'...
' some special processing is given below.']);

% Calculate the maximum distances from each cluster.
max_distances = max(distances);

[~, cluster_num] = max(max_distances);
```

```
38  [~, data_point] = max(distances(:,cluster_num));
39
40  data_clusters(data_point) = cluster_num;
41
42  ind = data_point;
43  num_assigned = 1;
44  end   %% End of exception handling.
45
46  % Save number of points per cluster, plus the mean vectors.
47  cluster_stats(c,1) = num_assigned;
48  if( num_assigned > 1 )
49 summ = sum(data(ind,:));
50  cluster_stats(c,2:ndims+1) = summ / num_assigned;
51  else
52  cluster_stats(c,2:ndims+1) = data(ind,:);
53  end
54
55  end
56
57  % Exit criteria
58  diff = sum(abs(cluster_stats(:) - last_clusters(:)));
59  if( diff < 0.00001 )
60  break;
61  end
62
63  % Assign each point to the nearest cluster center,
64  % and thus update distances and data_clusters.
65  [distances, data_clusters] = ...
66  min(dist(data, cluster_stats(:, 2:ndims+1)'), [], 2);
67
68  % Display clusters for the purpose of debugging.
69  cluster_stats
70  %pause;
71  end
```

A.3 swiftMatch.m

```
1  % By Jeff Yuanbo Han (u6617017), 2018-04-03.
2
3  [img1, dcp1, loc1] = sift('photo1.png');
4  [img2, dcp2, loc2] = sift('photo3.png');
5
6  % Find matched feature points.
7  r = 0.6;
8  distance = dist(dcp1, dcp2');
9  [sorted_dist, idx] = sort(distance, 2);
```

```
i = find(sorted_dist(:,1) < r*sorted_dist(:,2));
j = idx(i,1);
match1 = [loc1(i,1), loc1(i,2)];
match2 = [loc2(j,1), loc2(j,2)+size(img1,2)];

% Concatenate img1 and img2 as common_img.
common_img = appendimages(img1, img2);

% Display common_img, and draw lines connecting the matched SIFT feature
% points.
figure;
imshow(common_img);
hold on;
for m = 1:min(size(i),15)
line([match1(m,2),match2(m,2)], [match1(m,1),match2(m,1)]);
end
```