CMPE 685 Project 2: Color and Texture Features

Cody Smith
Rochester Institute of Technology
cds7494@rit.edu

1. Introduction

In this project, various aspects of image analysis and retrieval were examined including histograms and histogram analysis, segmentation, and class based image retrieval based on histogram matching as well as texture feature matching. The algorithms and equations described were implemented in Mathwork's Matlab.

2. Color Image Segmentation via k-means

2.1. Methods

In this exercise color images were analyzed and segmented. This task was done by using the k-means algorithm. The first segmentation was done using just the intensity, which was defined as such:

$$I = (R+G+B)/3$$

Equation 1: Intensity Definition

This intensity calculation is done for every pixel, and then a histogram of the intensities is computed using Matlab's built-in function 'histcounts', in this case 256 bins were used. Then, k-means clustering was done on the histogram.

This was done again using the full color of the image. Histograms with 256 bins were calculated for the 3 color channels, and then 3-dimensional k-means clustering was completed.

Finally, for both the intensity and color segmentation to visualize the clusters every pixel in the original image is gone over, and replaced with it's group.

For this exercise, Mathwork's built-in 'kmeans' function was used as a template and a custom k-means function was developed to mirror it's functionality. It can be viewed in the appendix as the file 'kmeans_cluster.m'. It take X, and a k and returns idx, a column vector that contains the group assignments for every point in X. This implementation of k-means works simply by looping until either the solution is reached because the centroids positions do not change or the maximum number of iterations are reached. Inside of this loop, every point is assigned to it's closest centroid, and then the centroid positions are updated based on what points are in it's group. The centroids are initialized to the first k points in X where k is the number of centroids.

2.1. Results

The results of taking an image and splitting the color channels (RGB) into separate 384-bin histograms and then being summed can be seen as Figure 1. This shows how for a given image the different channels have different levels of intensity, and these histograms are what was used by the k-means algorithm to segment the images.

The results of k-means segmentation for k values of 2 and 10 are shown as Figure 2. Of particular interest with this test image is how as the value of k is increased, the sky which is a gradient starts to become more and more segmented.

The results of k-means segmentation based on color with k values of 2, 5, and 10 is shown as Figure 3. Notice again how the sky gradient becomes segmented as the value of k increases, however with color it is less segmented with a k value of 5 then the intensity was even at a k value of 2. In general comparing the intensity segmentation vs. the color segmentation the color segmentation seems to perform a lot better, especially as the values of k go up.

3. Image Retrieval Using Color Histogram Features

3.1. Methods

In this exercise, similar images to a query image in one of 7 classes were attempted to be retrieved by using histogram intersection distance. First, the entire data set was stored in a 'map' object for easy retrieval, and the color histograms for each image were calculated. In this case, 96 bin color histograms were used.

The design of the program was to take a query image, and find the 4 closest images in the dataset based on histogram intersection distance. To do this, the function seen in the appendix as 'get_closest_hist_images.m' was created, which takes the dataset, and a query image name and returns 4 the four closest matches.

It does this by looping over every image in the dataset map scoring them, and then returning the four images with the lowest four scores indicating that they are the closest matches. The histogram intersection distance is calculated using the function seen in the appendix as 'histogram_diff.m', which loops over every bin in the histograms and adds to a result variable the square of the difference between the two bins.

Finally, an experiment was constructed which takes one image from each of the classes and finds the four closest matches, and this experiment was run many times to get a good idea of how the histogram matching was performing.

3.2. Results

The results of three runs of the experiment can be seen as Figures 4, 5, and 6. Each row represents one trial, which the query image is on the left (as is labeled), and the resulting 4 matches next. Notice how in most cases the color histogram matching does a very good job identifying similar images

This was used to observe some images where the matching performed well, and where it did not perform well. These examples are shown as Figure 7.

The advantages of color histogram matching is perhaps that it is pretty easy to perform, and in many cases does well identifying similar images. However, it starts to have trouble when images of the same object are in different lighting situations or with non-similar background. For example, it would not be able to detect the same animal in an image in a bright orange desert as well as a dark green forest.

4. Image Retrieval Using Texture Features

4.1. Methods

In this exercise, similar images to a query image in 6 classes split by texture were attempted to be retrieved by using texture features. A 9-dimensional texture feature was computed by running a series of masks to get a filter response. These responses were averaged across the whole image to after taking the absolute value so that a single value could be obtained from the response.

The design of the program was to take a query image, and find the 4 closest images in the dataset by using the chi-squared distance between their feature vectors.

The function to do this is in the appendix as 'get_closest_texture_images.m'. A loop was constructed to go over every image in the dataset and compute the chi-squared distance between the computed feature vectors of the query image and the other images in the dataset. The closest matches were the ones with the lowest distance, and the top 4 are returned from the function.

The chi-square distance between the vectors used for the purposes of this exercise is defined in equation 2.

$$d(x,y)=\sum[(x_i-y_i)^2/(x_i+y_i)]*1/2$$

Equation 2: Chi-Square Distance

This equation was implemented in Matlab as the function, seen in the appendix as 'pdist2_chisq.m'. This function was designed to interface the same way the Matlab 'pdist' function does so that different distance function could be tested easily.

Finally, an experiment was constructed in which one image in each class is tested. This was run multiple times to see how well the texture features performed on multiple images.

4.2. Results

The results of three runs of the experiment can been seen as Figures 8, 9, and 10. Each row represents one trial, with the query image on the left, and the resulting 4 matches to the right of it. Notice

how the performance of the retrieval is highly dependent on the pattern in the image. Certain patterns are easily retrieved, and others seem to be difficult for the algorithm to find.

These trials and others were used to observe some images that the matching performed well, and others that did not match well. These examples are shown as Figure 11.

The effectiveness of this approach is clearly very good. For textures where the patterns matched up well with texture feature, the results are consistently good. Where the approach seems to fall short is when the texture does not line up well with the texture feature, and perhaps the performance of the image retrieval could be tuned to work better with a certain pattern by adjusting the filters used to generate the feature.

5. References

1. http://www.mathworks.com/help/matlab/
2. http://www.cs.columbia.edu/~mmerler/project/code/pdist2.m

6. Appendix

1. part1.m
2. kmeans_cluster.m
3. part2.m
4. get_closest_hist_images.m
5. histogram_diff.m
6. part3.m
7. get_closest_texture_image.m
8. pdist2_chisq.m
9. Figures

```
% Part 1
% By: Cody Smith

%% Part A: Color Histogram
dir_path = '/Volumes/Macintosh HD/Google Drive/School/Spring 2016/Computer Vision/Project
2/Color_Images/';
img_path = strcat(dir_path, '248.jpg');

% Read in image, split into RGB
I = imread(img_path);
R = I(:,:,1);
G = I(:,:,2);
B = I(:,:,3);
figure('Name', 'Part A: Color Histogram');

R_h = histcounts(R(:), 384);
G_h = histcounts(G(:), 384);
B_h = histcounts(B(:), 384);

subplot(3,3, 1);
imshow(I);
title('Image');

subplot(3,3, 4);
bar(R_h, 'r');
title('Red');

subplot(3,3, 5);
bar(G_h, 'g');
title('Green');

subplot(3,3, 6);
bar(B_h, 'b');
title('Blue');

subplot(3,3, 7);
bar(sum([R_h; G_h; B_h]), 'black');
title('Sum');

%% Part C: K-means on Intensity

figure('Name', 'Part C: K-means Segmentation based on Intensity');

subplot(2,2, 1);
imshow(I);
title('Original');

intensity = uint8(round((sum(I, 3))/3));
subplot(2,2, 2);
imshow(intensity);
title('Intensity');

intensity_hist = histcounts(intensity(:), 256).';

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

% Compute kmeans (k = 2)
k_result = kmeans_cluster(intensity_hist, 2);

% Create result image from result
[size_x, size_y] = size(intensity);
kmean_image = zeros(size_x, size_y);

for i = 1:size_x
```

```
    for j = 1:size_y
        kmean_image(i,j) = k_result(intensity(i,j)+1);
    end
end

subplot(2,2, 3);
imshow(kmean_image, [min(min(kmean_image)) max(max(kmean_image))]);
title('K-mean with k = 2');

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

% Compute kmeans (k = 10)
k_result = kmeans_cluster(intensity_hist, 10);

% Create result image from result
[size_x, size_y] = size(intensity);
kmean_image = zeros(size_x, size_y);

for i = 1:size_x
    for j = 1:size_y
        kmean_image(i,j) = k_result(intensity(i,j)+1);
    end
end

subplot(2,2, 4);
imshow(kmean_image, [min(min(kmean_image)) max(max(kmean_image))]);
title('K-mean with k = 10');

%% Part E: K-means Segmentation based on Color

figure('Name', 'Part E: K-means Segmentation based on Color');

subplot(2,2, 1);
imshow(I);
title('Original');

R_h = histcounts(R(:), 256).';
G_h = histcounts(G(:), 256).';
B_h = histcounts(B(:), 256).';

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

% Compute kmeans
k = 2;
k_result = kmeans_cluster([R_h, G_h, B_h], k);

% Create result image from result
[size_x, size_y] = size(intensity);
kmean_image = zeros(size_x, size_y);

for i = 1:size_x
    for j = 1:size_y
        kmean_image(i,j) = k_result(intensity(i,j)+1);
    end
end

subplot(2,2, 2);
imshow(kmean_image, [min(min(kmean_image)) max(max(kmean_image))]);
title('K-mean with k = 2');

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

% Compute kmeans
k = 5;
```

```
k_result = kmeans_cluster([R_h, G_h, B_h], k);

% Create result image from result
[size_x, size_y] = size(intensity);
kmean_image = zeros(size_x, size_y);

for i = 1:size_x
    for j = 1:size_y
        kmean_image(i,j) = k_result(intensity(i,j)+1);
    end
end

subplot(2,2, 3);
imshow(kmean_image, [min(min(kmean_image)) max(max(kmean_image))]);
title('K-mean with k = 5');

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

% Compute kmeans
k = 10;
k_result = kmeans_cluster([R_h, G_h, B_h], k);

% Create result image from result
[size_x, size_y] = size(intensity);
kmean_image = zeros(size_x, size_y);

for i = 1:size_x
    for j = 1:size_y
        kmean_image(i,j) = k_result(intensity(i,j)+1);
    end
end

subplot(2,2, 4);
imshow(kmean_image, [min(min(kmean_image)) max(max(kmean_image))]);
title('K-mean with k = 10');
```

```
% Part 2
% By: Cody Smith

%% Part A: Color Histogram

% map of strucutures
data = containers.Map();

dir_path = '/Volumes/Macintosh HD/Google Drive/School/Spring 2016/Computer Vision/Project
2/Color_Images/';

% Loop over the files and add to the map of histograms
files = dir(strcat(dir_path, '*.jpg'));
for file = files'
    I = imread(strcat(dir_path, file.name));
    R = I(:,:,1);
    G = I(:,:,2);
    B = I(:,:,3);

    s = struct();
    s.I = I;
    s.R_h = histcounts(R(:), 96);
    s.G_h = histcounts(G(:), 96);
    s.B_h = histcounts(B(:), 96);
    s.filename = file.name;
    data(file.name) = s;
end

% 200 class
query = '205.jpg';
[match1, match2, match3, match4] = get_closest_hist_images(data, query);

figure('Name', 'Top 4 matches for classes');

subplot(7,5, 1);
imshow(data(query).I);
title(strcat('Query Image: ', query));

subplot(7,5, 2);
imshow(match1.I);
title(strcat('Match 1: ', match1.filename));

subplot(7,5, 3);
imshow(match2.I);
title(strcat('Match 2: ', match2.filename));

subplot(7,5, 4);
imshow(match3.I);
title(strcat('Match 3: ', match3.filename));

subplot(7,5, 5);
imshow(match4.I);
title(strcat('Match 4: ', match4.filename));

% 300 class
query = '302.jpg';
[match1, match2, match3, match4] = get_closest_hist_images(data, query);

subplot(7,5, 6);
imshow(data(query).I);
title(strcat('Query Image: ', query));

subplot(7,5, 7);
imshow(match1.I);
```

```
title(strcat('Match 1: ', match1.filename));

subplot(7,5, 8);
imshow(match2.I);
title(strcat('Match 2: ', match2.filename));

subplot(7,5, 9);
imshow(match3.I);
title(strcat('Match 3: ', match3.filename));

subplot(7,5, 10);
imshow(match4.I);
title(strcat('Match 4: ', match4.filename));

% 500 class
query = '502.jpg';
[match1, match2, match3, match4] = get_closest_hist_images(data, query);

subplot(7,5, 11);
imshow(data(query).I);
title(strcat('Query Image: ', query));

subplot(7,5, 12);
imshow(match1.I);
title(strcat('Match 1: ', match1.filename));

subplot(7,5, 13);
imshow(match2.I);
title(strcat('Match 2: ', match2.filename));

subplot(7,5, 14);
imshow(match3.I);
title(strcat('Match 3: ', match3.filename));

subplot(7,5, 15);
imshow(match4.I);
title(strcat('Match 4: ', match4.filename));

% 600 class
query = '602.jpg';
[match1, match2, match3, match4] = get_closest_hist_images(data, query);

subplot(7,5, 16);
imshow(data(query).I);
title(strcat('Query Image: ', query));

subplot(7,5, 17);
imshow(match1.I);
title(strcat('Match 1: ', match1.filename));

subplot(7,5, 18);
imshow(match2.I);
title(strcat('Match 2: ', match2.filename));

subplot(7,5, 19);
imshow(match3.I);
title(strcat('Match 3: ', match3.filename));

subplot(7,5, 20);
imshow(match4.I);
title(strcat('Match 4: ', match4.filename));

% 700 class
query = '702.jpg';
```

```
[match1, match2, match3, match4] = get_closest_hist_images(data, query);

subplot(7,5, 21);
imshow(data(query).I);
title(strcat('Query Image: ', query));

subplot(7,5, 22);
imshow(match1.I);
title(strcat('Match 1: ', match1.filename));

subplot(7,5, 23);
imshow(match2.I);
title(strcat('Match 2: ', match2.filename));

subplot(7,5, 24);
imshow(match3.I);
title(strcat('Match 3: ', match3.filename));

subplot(7,5, 25);
imshow(match4.I);
title(strcat('Match 4: ', match4.filename));

% 800 class
query = '802.jpg';
[match1, match2, match3, match4] = get_closest_hist_images(data, query);

subplot(7,5, 26);
imshow(data(query).I);
title(strcat('Query Image: ', query));

subplot(7,5, 27);
imshow(match1.I);
title(strcat('Match 1: ', match1.filename));

subplot(7,5, 28);
imshow(match2.I);
title(strcat('Match 2: ', match2.filename));

subplot(7,5, 29);
imshow(match3.I);
title(strcat('Match 3: ', match3.filename));

subplot(7,5, 30);
imshow(match4.I);
title(strcat('Match 4: ', match4.filename));

% 900 class
query = '902.jpg';
[match1, match2, match3, match4] = get_closest_hist_images(data, query);

subplot(7,5, 31);
imshow(data(query).I);
title(strcat('Query Image: ', query));

subplot(7,5, 32);
imshow(match1.I);
title(strcat('Match 1: ', match1.filename));

subplot(7,5, 33);
imshow(match2.I);
title(strcat('Match 2: ', match2.filename));

subplot(7,5, 34);
imshow(match3.I);
```

```
title(strcat('Match 3: ', match3.filename));

subplot(7,5, 35);
imshow(match4.I);
title(strcat('Match 4: ', match4.filename));
```

```
function [ match1, match2, match3, match4 ] = get_closest_hist_images( data, query )
%GET_CLOSEST_HIST_IMAGES Given data and an input image name finds histogram
%matches

query_image = data(query);

match1_score = Inf;
match2_score = Inf;
match3_score = Inf;
match4_score = Inf;

keys = data.keys;

for i = 1:length(keys)

    key = keys{i};

    if ~isequal(query, key)

        test_image = data(key);

        score = histogram_diff(query_image.R_h, test_image.R_h) +
histogram_diff(query_image.G_h, test_image.G_h) + histogram_diff(query_image.B_h,
test_image.B_h);

        if score < match1_score
            match1 = test_image;
            match1_score = score;
        elseif score < match2_score
            match2 = test_image;
            match2_score = score;
        elseif score < match3_score
            match3 = test_image;
            match3_score = score;
        elseif score < match4_score
            match4 = test_image;
            match4_score = score;
        end
    end

end

end
```

```
function [ sum ] = histogram_diff( histogram1, histogram2 )
%HISTOGRAM_DIFF Calculates the diff between two histograms

    sum = 0;

    [size_y, size_x] = size(histogram1);
    for x=1:size_x
        sum = sum + (histogram1(x) - histogram2(x))^2;
    end

end
```

```
% Part 3: Texture Features
% By: Cody Smith

dir_path = '/Volumes/Macintosh HD/Google Drive/School/Spring 2016/Computer Vision/Project
2/Texture_Images/';

% Filter definitions %

% 1D filter components
L5 = [1 4 6 4 1];
E5 = [-1 -2 0 2 1];
S5 = [-1 0 2 0 -1];
R5 = [1 -4 6 -4 1];

% 2D filter combinations
L5E5 = L5' * E5;
L5R5 = L5' * R5;
L5S5 = L5' * S5;
E5L5 = E5' * L5;
E5R5 = E5' * R5;
E5S5 = E5' * S5;
E5E5 = E5' * E5;
S5L5 = S5' * L5;
S5E5 = S5' * E5;
S5R5 = S5' * R5;
S5S5 = S5' * S5;
R5L5 = R5' * L5;
R5E5 = R5' * E5;
R5S5 = R5' * S5;
R5R5 = R5' * S5;

% Loop over the files %

data = containers.Map();

subdirs = dir(dir_path);
for subdir = subdirs'

    full_path = strcat(dir_path,subdir.name,'/');
    files = dir(strcat(full_path, '*.jpg'));

    for file = files'

        % Store each texture image in the data map

        s = struct();
        s.I = imread(strcat(full_path, file.name));
        s.filename = file.name;

        % Compute 9 dimensional texture feature
        s.features = [];

        % Feature 1
        f1 = imfilter(s.I, L5E5);
        f2 = imfilter(s.I, E5L5);
        feature_val = mean(mean(abs([f1;f2])));
        s.features = [s.features feature_val];

        % Feature 2
        f1 = imfilter(s.I, L5R5);
        f2 = imfilter(s.I, R5L5);
        feature_val = mean(mean(abs([f1;f2])));
        s.features = [s.features feature_val];
```

```
        % Feature 3
        f1 = imfilter(s.I, L5S5);
        f2 = imfilter(s.I, S5L5);
        feature_val = mean(mean(abs([f1;f2])));
        s.features = [s.features feature_val];

        % Feature 4
        f1 = imfilter(s.I, E5R5);
        f2 = imfilter(s.I, R5E5);
        feature_val = mean(mean(abs([f1;f2])));
        s.features = [s.features feature_val];

        % Feature 5
        f1 = imfilter(s.I, E5S5);
        f2 = imfilter(s.I, S5E5);
        feature_val = mean(mean(abs([f1;f2])));
        s.features = [s.features feature_val];

        % Feature 6
        f1 = imfilter(s.I, S5R5);
        f2 = imfilter(s.I, R5S5);
        feature_val = mean(mean(abs([f1;f2])));
        s.features = [s.features feature_val];

        % Feature 7
        f1 = imfilter(s.I, E5E5);
        feature_val = mean(mean(abs(f1)));
        s.features = [s.features feature_val];

        % Feature 8
        f1 = imfilter(s.I, R5R5);
        feature_val = mean(mean(abs(f1)));
        s.features = [s.features feature_val];

        % Feature 9
        f1 = imfilter(s.I, S5S5);
        feature_val = mean(mean(abs(f1)));
        s.features = [s.features feature_val];

        % Add struct to data
        data(file.name) = s;

    end
end

figure('Name', 'Top 4 matches for classes');

% T01 class
query = 'T01_03.jpg';
[match1, match2, match3, match4] = get_closest_texture_images(data, query);

subplot(6,5, 1);
imshow(data(query).I);
title(strcat('Query Image: ', query), 'interpreter', 'none');

subplot(6,5, 2);
imshow(match1.I);
title(strcat('Match 1: ', match1.filename), 'interpreter', 'none');

subplot(6,5, 3);
imshow(match2.I);
title(strcat('Match 2: ', match2.filename), 'interpreter', 'none');

subplot(6,5, 4);
```

```
imshow(match3.I);
title(strcat('Match 3: ', match3.filename), 'interpreter', 'none');

subplot(6,5, 5);
imshow(match4.I);
title(strcat('Match 4: ', match4.filename), 'interpreter', 'none');

% T05 class
query = 'T05_03.jpg';
[match1, match2, match3, match4] = get_closest_texture_images(data, query);

subplot(6,5, 6);
imshow(data(query).I);
title(strcat('Query Image: ', query), 'interpreter', 'none');

subplot(6,5, 7);
imshow(match1.I);
title(strcat('Match 1: ', match1.filename), 'interpreter', 'none');

subplot(6,5, 8);
imshow(match2.I);
title(strcat('Match 2: ', match2.filename), 'interpreter', 'none');

subplot(6,5, 9);
imshow(match3.I);
title(strcat('Match 3: ', match3.filename), 'interpreter', 'none');

subplot(6,5, 10);
imshow(match4.I);
title(strcat('Match 4: ', match4.filename), 'interpreter', 'none');

% T12 class
query = 'T12_03.jpg';
[match1, match2, match3, match4] = get_closest_texture_images(data, query);

subplot(6,5, 11);
imshow(data(query).I);
title(strcat('Query Image: ', query), 'interpreter', 'none');

subplot(6,5, 12);
imshow(match1.I);
title(strcat('Match 1: ', match1.filename), 'interpreter', 'none');

subplot(6,5, 13);
imshow(match2.I);
title(strcat('Match 2: ', match2.filename), 'interpreter', 'none');

subplot(6,5, 14);
imshow(match3.I);
title(strcat('Match 3: ', match3.filename), 'interpreter', 'none');

subplot(6,5, 15);
imshow(match4.I);
title(strcat('Match 4: ', match4.filename), 'interpreter', 'none');

% T13 class
query = 'T13_03.jpg';
[match1, match2, match3, match4] = get_closest_texture_images(data, query);

subplot(6,5, 16);
imshow(data(query).I);
title(strcat('Query Image: ', query), 'interpreter', 'none');

subplot(6,5, 17);
```

```
imshow(match1.I);
title(strcat('Match 1: ', match1.filename), 'interpreter', 'none');

subplot(6,5, 18);
imshow(match2.I);
title(strcat('Match 2: ', match2.filename), 'interpreter', 'none');

subplot(6,5, 19);
imshow(match3.I);
title(strcat('Match 3: ', match3.filename), 'interpreter', 'none');

subplot(6,5, 20);
imshow(match4.I);
title(strcat('Match 4: ', match4.filename), 'interpreter', 'none');

% T18 class
query = 'T18_03.jpg';
[match1, match2, match3, match4] = get_closest_texture_images(data, query);

subplot(6,5, 21);
imshow(data(query).I);
title(strcat('Query Image: ', query), 'interpreter', 'none');

subplot(6,5, 22);
imshow(match1.I);
title(strcat('Match 1: ', match1.filename), 'interpreter', 'none');

subplot(6,5, 23);
imshow(match2.I);
title(strcat('Match 2: ', match2.filename), 'interpreter', 'none');

subplot(6,5, 24);
imshow(match3.I);
title(strcat('Match 3: ', match3.filename), 'interpreter', 'none');

subplot(6,5, 25);
imshow(match4.I);
title(strcat('Match 4: ', match4.filename), 'interpreter', 'none');

% T25 class
query = 'T25_03.jpg';
[match1, match2, match3, match4] = get_closest_texture_images(data, query);

subplot(6,5, 26);
imshow(data(query).I);
title(strcat('Query Image: ', query), 'interpreter', 'none');

subplot(6,5, 27);
imshow(match1.I);
title(strcat('Match 1: ', match1.filename), 'interpreter', 'none');

subplot(6,5, 28);
imshow(match2.I);
title(strcat('Match 2: ', match2.filename), 'interpreter', 'none');

subplot(6,5, 29);
imshow(match3.I);
title(strcat('Match 3: ', match3.filename), 'interpreter', 'none');

subplot(6,5, 30);
imshow(match4.I);
title(strcat('Match 4: ', match4.filename), 'interpreter', 'none');
```

```
function [ match1, match2, match3, match4 ] = get_closest_texture_images( data, query )
%GET_CLOSEST_TEXTURE_IMAGE Gets the closest matches for a texture

query_image = data(query);

match1_score = Inf;
match2_score = Inf;
match3_score = Inf;
match4_score = Inf;

keys = data.keys;

for i = 1:length(keys)

    key = keys{i};

    if ~isequal(query, key)

        test_image = data(key);

        %TODO chi-square
        score = pdist2_chisq(test_image.features, query_image.features);

        if score < match1_score
            match1 = test_image;
            match1_score = score;
        elseif score < match2_score
            match2 = test_image;
            match2_score = score;
        elseif score < match3_score
            match3 = test_image;
            match3_score = score;
        elseif score < match4_score
            match4 = test_image;
            match4_score = score;
        end
    end

end

end
```

```
function [ d ] = pdist2_chisq( x, y )
%PDIST2_CHISQ Compute chi-square distance

    [size_y, size_x] = size(x);

    di = zeros(size_x,1);

    for i=1:size_x
        di(i) = ((x(i)-y(i))^2) / (x(i) + y(i));
    end

    d = sum(di)/2;

end
```

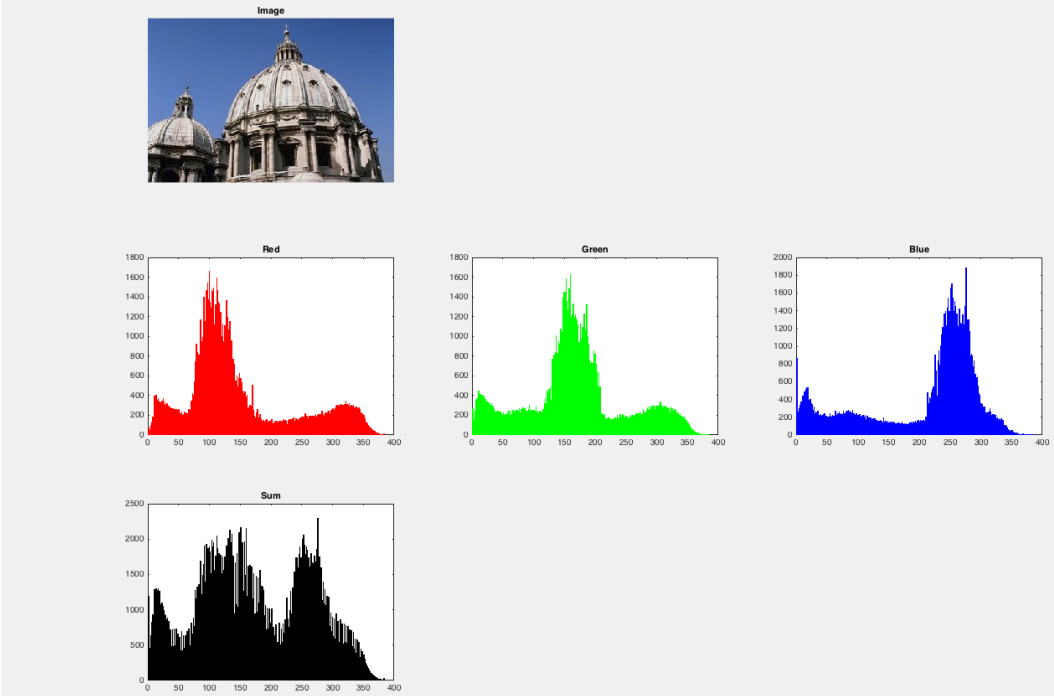

Figure 1: Result of Taking Color Histograms of Image

Figure 2: K-means Segmentation Based on Intensity

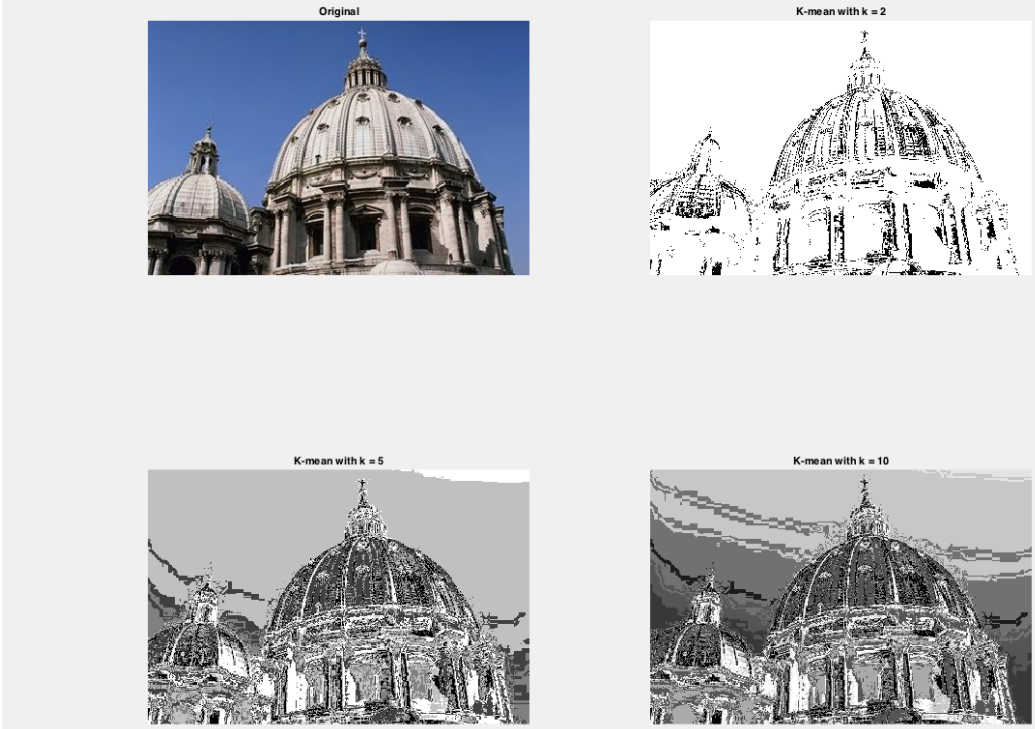

Figure 3: K-means Segmentation Based on Color

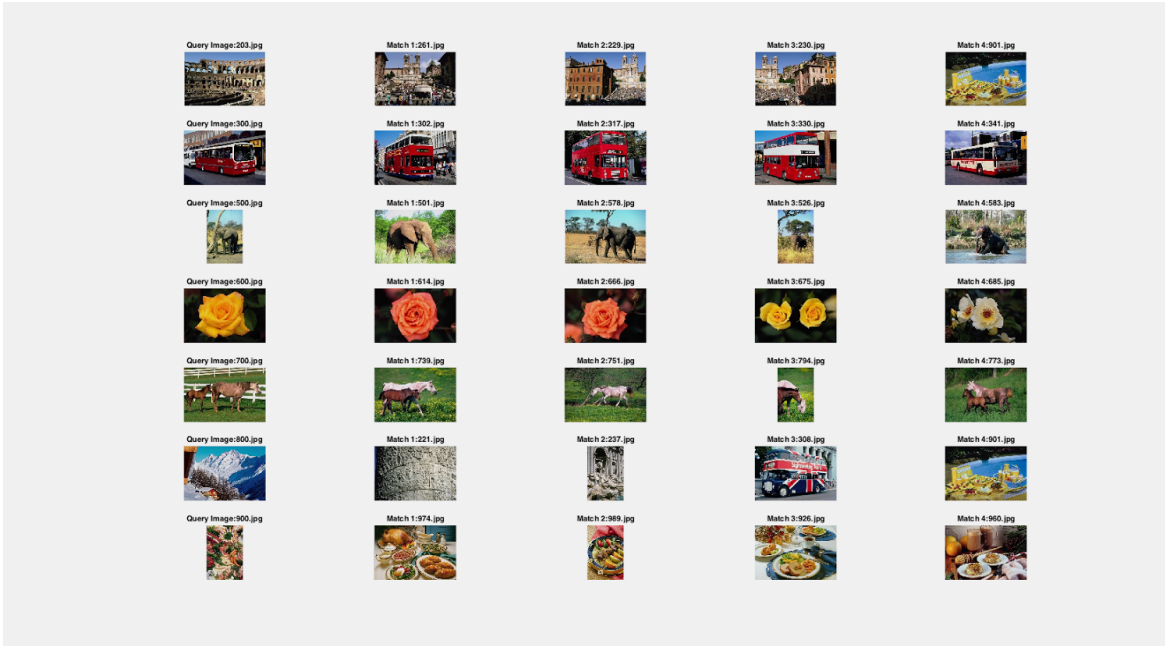

Figure 4: Result of Color Histogram Matching Trial 1

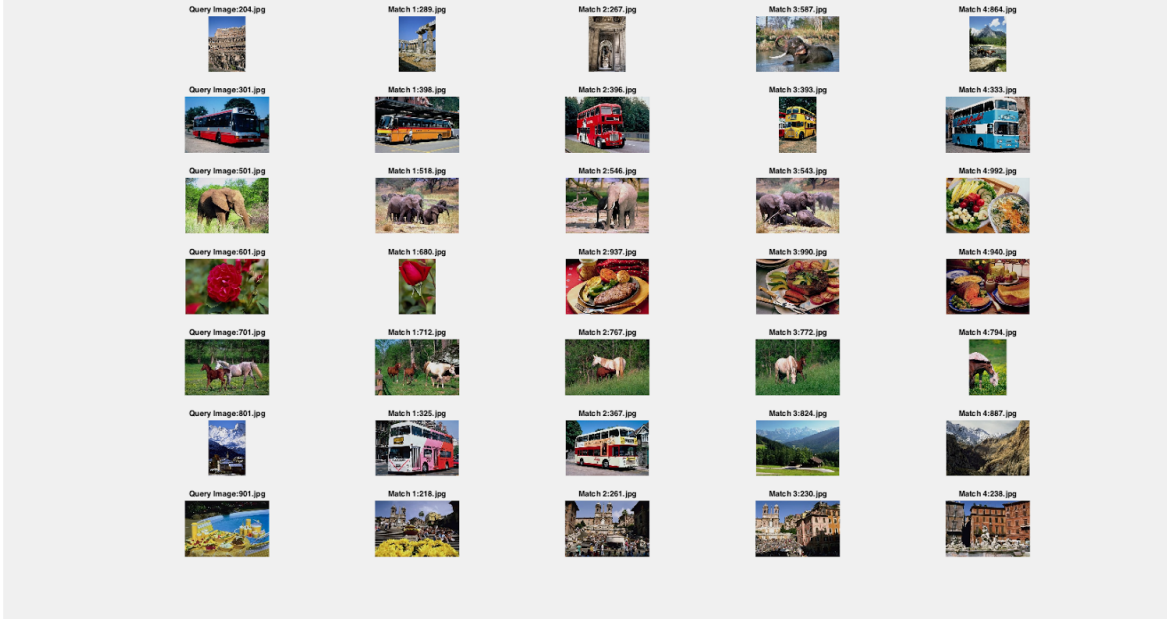

Figure 5: Result of Color Histogram Matching Trial 2

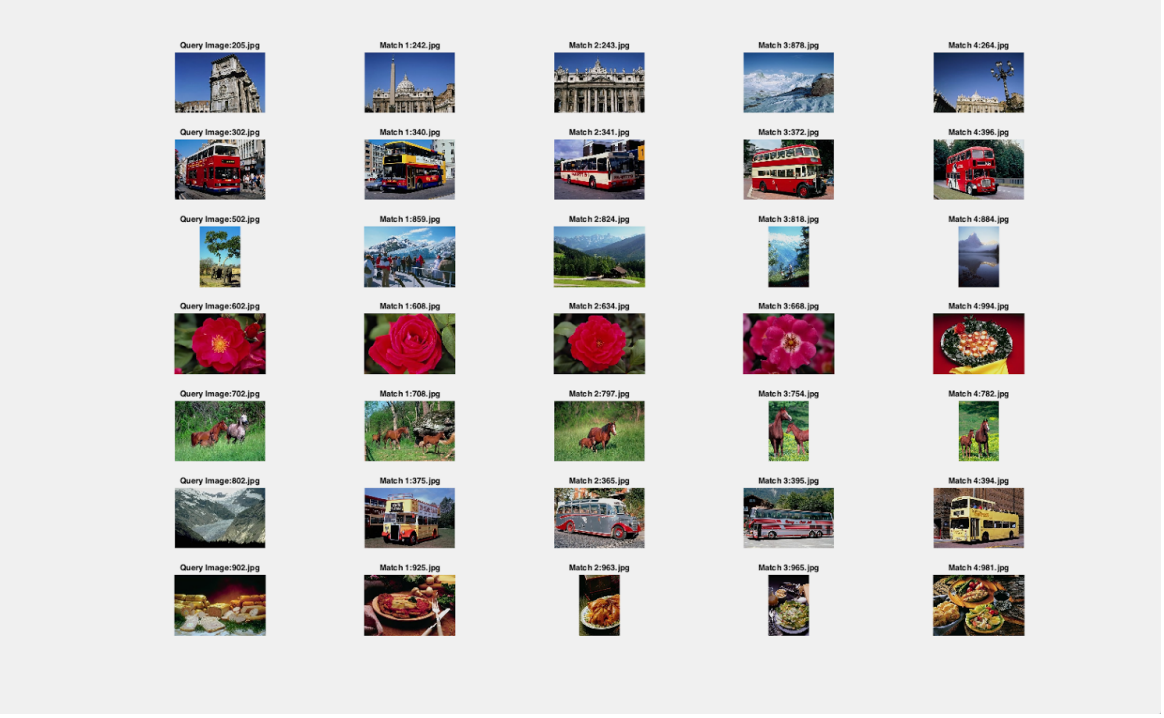

Figure 6: Result of Color Histogram Matching Trial 3

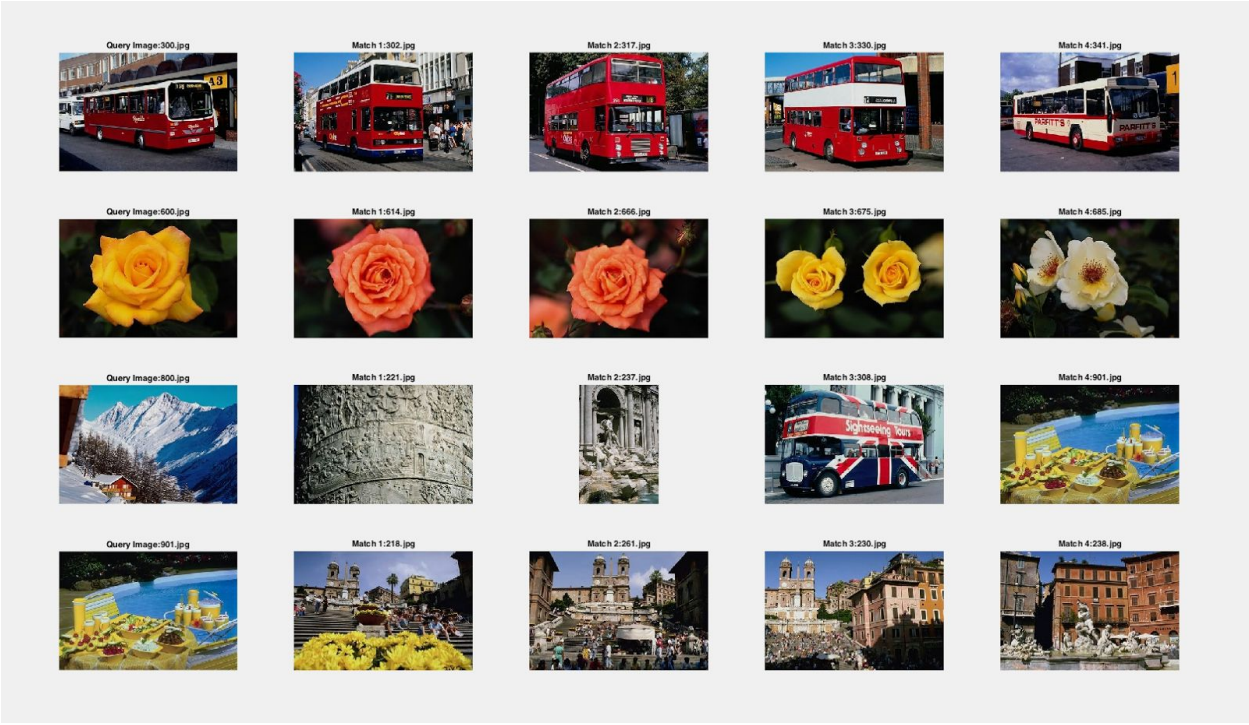

Figure 7: Good and Bad Examples of Histogram Color Matching

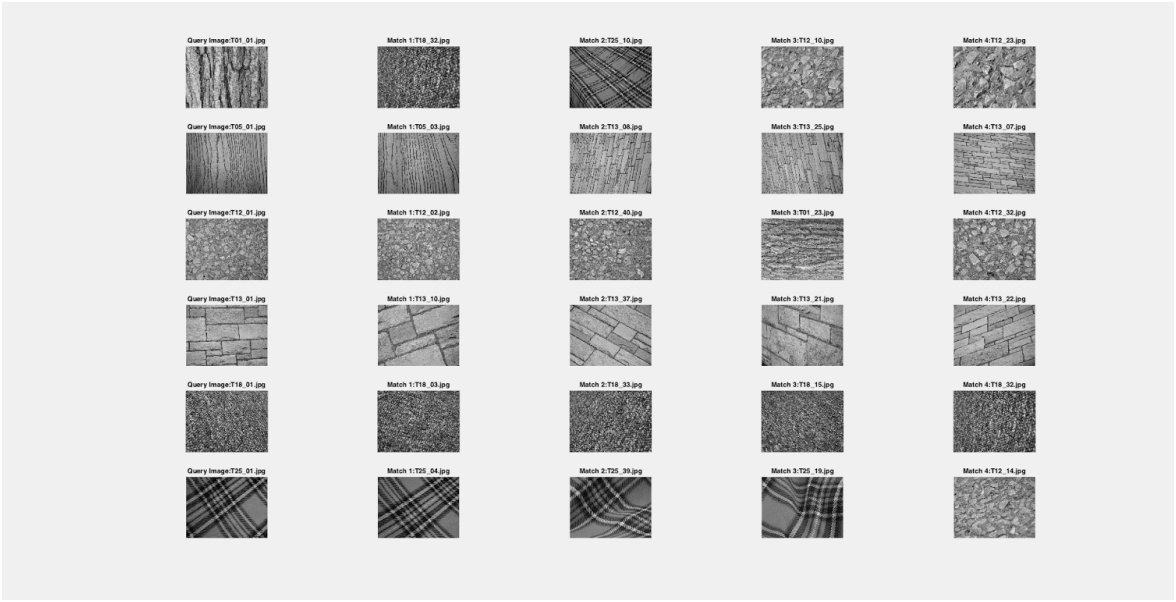

Figure 8: Result of Texture Matching Trial 1

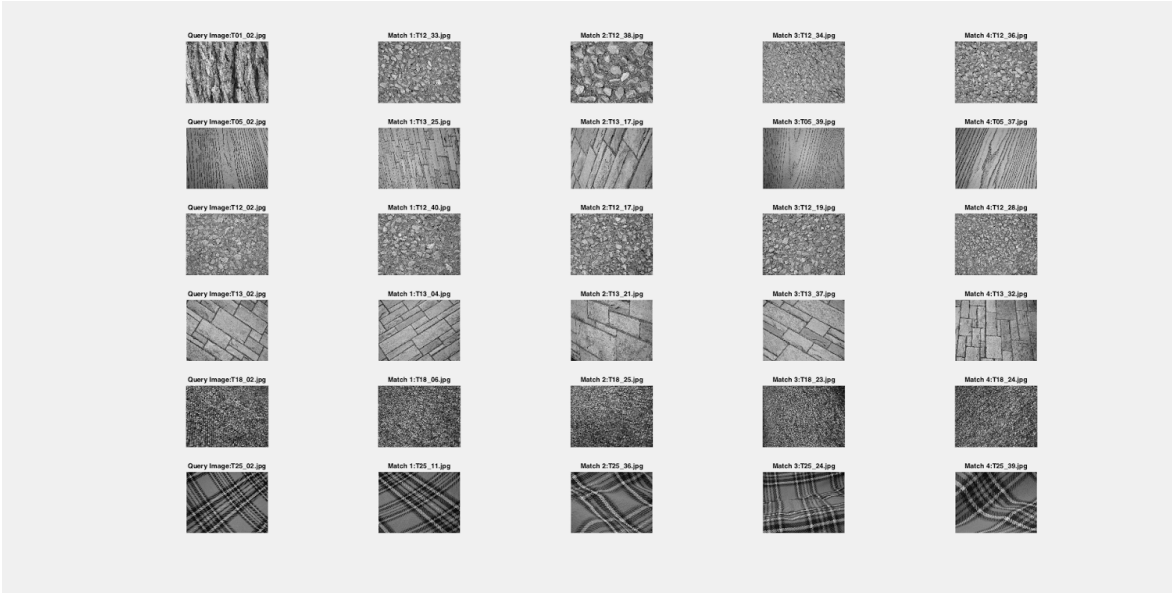

Figure 9: Result of Texture Matching Trial 2

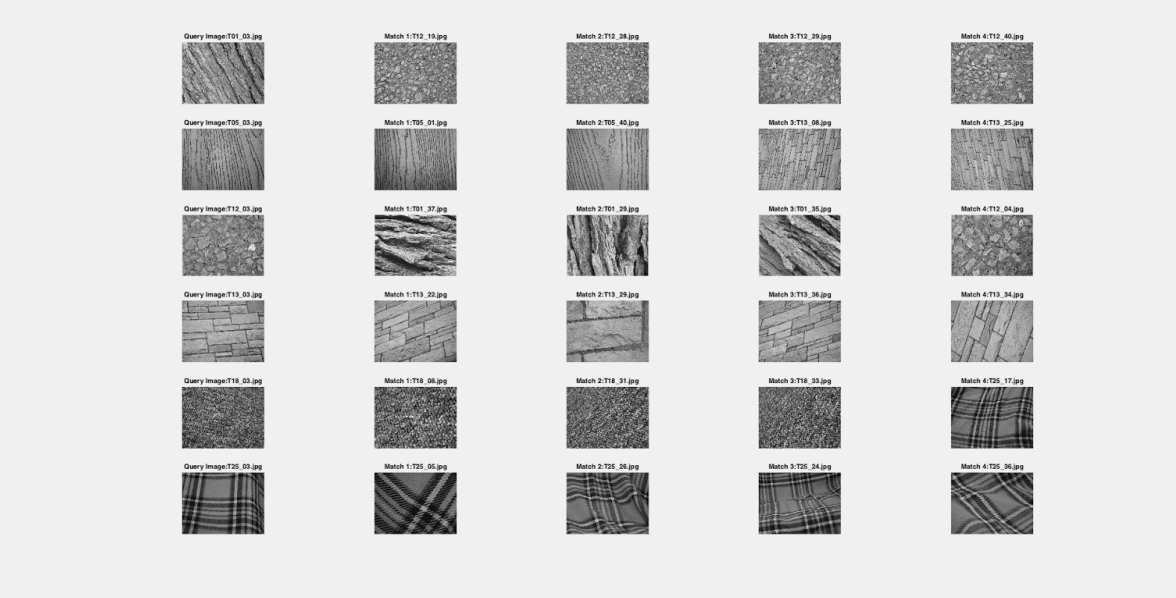

Figure 10: Result of Texture Matching Trial 3

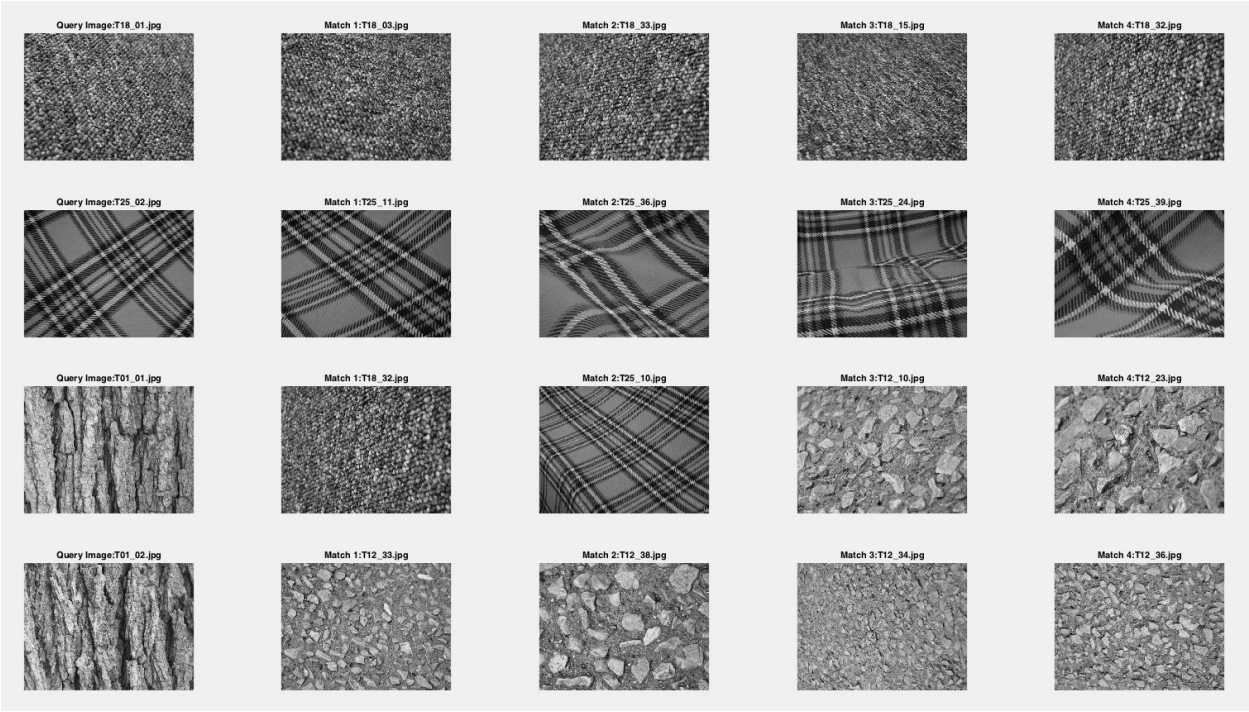

Figure 11: Good and Bad Examples of Texture Matching