Discovery of Camera Network Topology

Jason Lin, Xuanming Lu

Dec 10, 2008

Boston University

Department of Electrical and Computer Engineering

Technical report No. ECE-2008-06

BOSTON UNIVERSITY

Discovery of Camera Network Topology

Jason Lin, Xuanming Lu

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

Dec 10, 2008

Technical Report No. ECE-2008-06

Summary

Our task is to develop an algorithm to decide if several cameras are looking at the same scene or not. In other words, we determine if the two images taken from camera(s) either fully share a view, have disjoint fields of view, or partially share a view (top, bottom, left, and right periphery). Our solution involves choosing one of two algorithms (see full report for more details), the Mean Squared Error technique and Covariance Matrix technique; the first technique performs much faster when it comes to run-time, while the second is more robust.

Contents

List of Figures

List of Tables

1 Introduction

Our goal is to determine if two images, A and B, (taken from one or a few cameras) belong to one of three categories: they share the same view, their fields of views are disjoint, or they partially share a view (which we further divide into whether A is the left, right, above, or below B).

To better understand the meaning of this, below is an explanation of two basic cases.

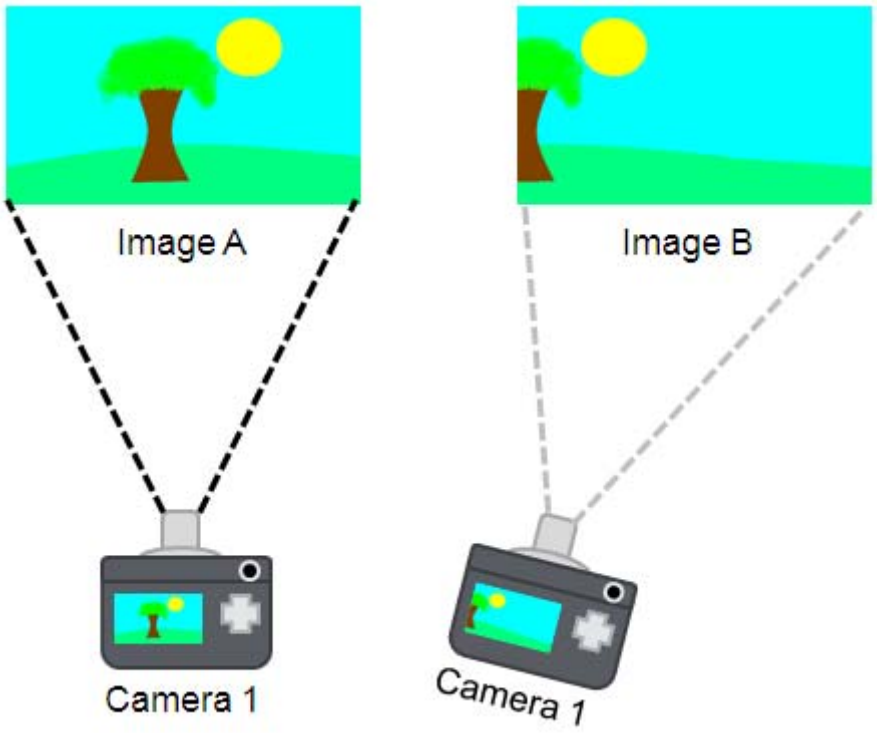


Figure 1: Basic Case 1

Here, we have an example where one camera is looking at two different scenes (one is to the left of the other). Our goal is to accurately detect this topology by stating that Image A is to the left of B.

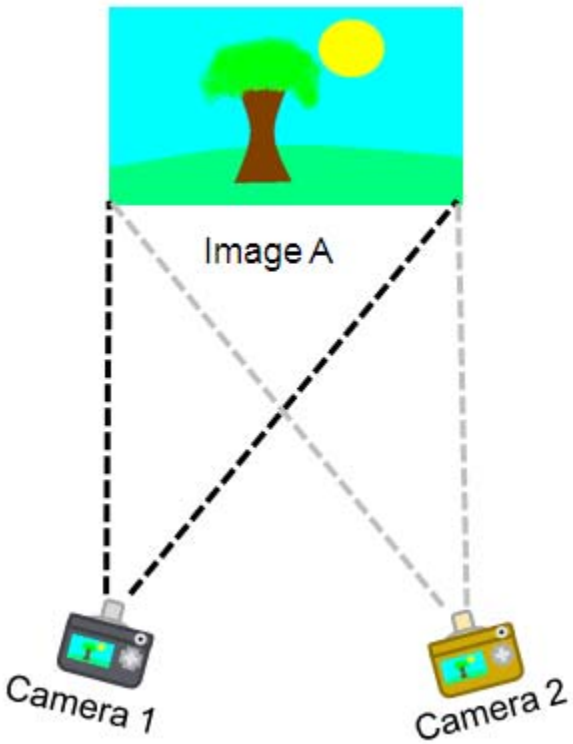


Figure 2: Basic Case 2

This case describes what happens when two cameras are looking at one particular scene. Here we assume that the cameras are looking a distant scene so that we can consider the cameras are parallel. Our goal is to detect that their field of views are the same.

Now that the purpose has been explained briefly, succeeding sections will be much easier to comprehend.

2 Literature Review

Since this topic is fairly new, it has not received much attention in literature; however, the notion of looking for "similarities" or "dissimilarities" between images/frames has been discussed in Porikli, Tuzel, and Meer's paper [1]. Basically, these authors were able to detect objects using a covariance based object description by capturing both spatial and statistical properties. The covariance matrix effectively combines multiple features while maintaining a small dimensionality. As a result, this matrix will be sufficient in matching regions of different views and poses, different image gradients, and different orientations. In addition to the high detection rate that this method yields, the authors explained each step of this process quite clearly, making it easier for a reader to understand it and implement it.

Therefore, this approach will be beneficial in solving a problem like ours. Details of this method will be explained in the "Implementation" section.

3 Problem Statement

Here are a total of three steps to accomplish the purpose we mentioned above:

Step 1: Divide up each image into N x N blocks

Figure 3: N = 128

Figure 4: N = 64 (Default)

As shown in the figures above, the first step is to divide up each image into N x N blocks, so we can compare one block in Image A with those in Image B. The default number of N is 64. *Note: if two images are of different size (i.e. they come different cameras) there exists a preliminary step before step 1 to expand the size of smaller image (via interpolation) so that they are the same size before applying it to our algorithm.*

Step 2: Find Evidence

Figure 5: Image A

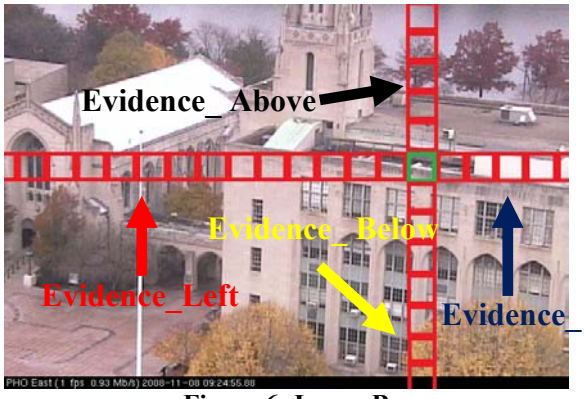


Figure 6: Image B

For each block in Image A, we compare it with blocks directly above, below, to the left, and right of that corresponding block in Image B for dissimilarity. If the blocks have a small enough dissimilarity value, we can consider this as evidence in that direction. Thus, when we go through all the blocks in image A, all the evidence will be accumulated in every direction. See the Implementation section for the methods we used to define the dissimilarity criterion (formulas and algorithms will be described there as well).

Step 3: Decision Making

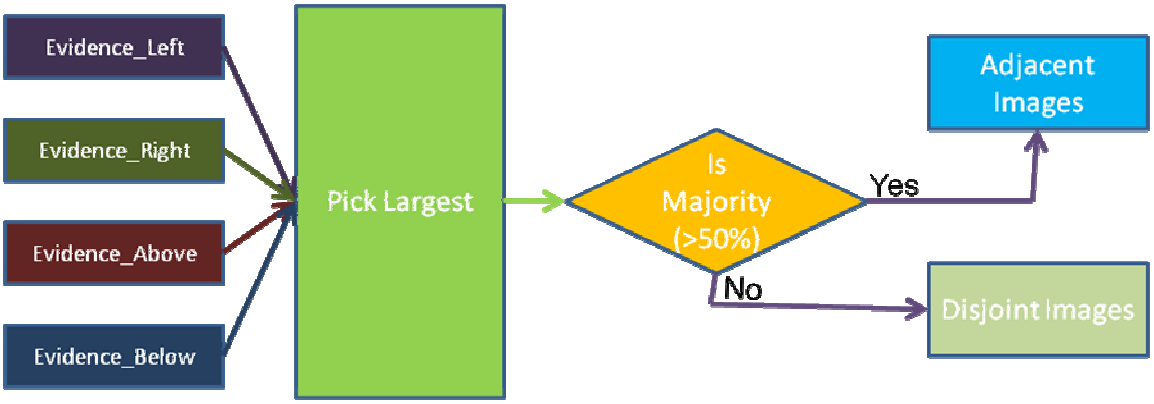


Figure 7: Block Diagram of Decision Maker

Once we accumulate all the evidence for all the blocks, we choose the largest of them and see if the majority of the evidence lies within in (>50% of the total evidence). If this is true, we have found adjacent images and their relative position; otherwise, we have found two very disjoint images. In the special case where all four evidences are above >20% (very uniformly distributed), we judge that the two images share the same field of view.

4 Implementation

Designing the method to compute the dissimilarity criterion for two blocks is the most crucial part of our project. As explained below, we have used two methods to accomplish this task.

4.1 Method1: Mean Squared Error Technique

The method here is rather typical and commonly used for basic comparison of blocks.

Equation 1: MSE Technique

$$MSE = \frac{1}{MN} \sum_{y=1}^{M} \sum_{x=1}^{N} [I(x,y) - I'(x,y)]^2$$

In the MSE Technique, we simply take the average squared difference between each pixel in the blocks. If the value is sufficiently small, we consider the images as dissimilar. Here, we found the term, small, to be less than an error of 1000, which was optimal for the various cases of images we used.

Although this method is fast and simple, it may run into problems if the two images are taken from different cameras (meaning that they are slightly rotated or angled). Such rotations will generate a large MSE value and may lead to false detections. As a result, we found a much more robust technique from [1], as explained in the following section.

4.2 Method 2: Covariance Matrix Technique

Basically, what this particular method does is to try to extract all the important features of each particular pixel in a block before we compare it with other blocks.

For example, let us say we have an N x N block. For each pixel inside of it, there are actually a few statistics that we may find useful. Some things that probably come to mind immediately are the X and Y coordinates of the pixel, and the R/G/B values. Also, we apply the edge detection (Canny) to the image so that we can distinguish the boundaries of objects more clearly; we can see either a value or 0 or 1 in the pixel which

signifies that the edge is either not present or present respectively. We place the features of pixel into a vector (as seen below):

Equation 2: Feature Vector for Each Pixel

$$\mathbf{f}_k = [\, x \quad y \quad I(x,y) \quad I_x(x,y) \quad \dots \,]$$

Now we have 6 features (x, y, R, G, B, and edge detection). Then, we simply apply the covariance matrix formula (an average of the difference between feature vector and the mean of the feature vectors) to obtain a 6 x 6 matrix to represent all the important features of our block.

Equation 3: Covariance Matrix of Features

$$\mathbf{C}_R = \frac{1}{MN}\sum_{k=1}^{MN}(\mathbf{f}_k - \boldsymbol{\mu}_R)(\mathbf{f}_k - \boldsymbol{\mu}_R)^T$$

To compute the dissimilarity between two blocks now, we find the sum of the squared logarithms of generalized eigenvalues between their covariance matrices (similar to that of the MSE approach)

Equation 4: Dissimilarity Criterion

$$\rho(\mathbf{C}_i, \mathbf{C}_j) = \sqrt{\sum_{k=1}^{d} ln^2 \lambda_k(\mathbf{C}_i, \mathbf{C}_j)}$$

If the dissimilarity is sufficiently small, this is our method of accumulating evidence. Here, we define the term, small, to be less than a dissimilarity value of 1, which was also the optimal value for the cases of images we used. *Note: all MATLAB code for these algorithms can be found in the Appendix.*

5 Experimental Results

We can separate results into two categories: those that are taken from the same camera and those that are taken from two cameras. Each category will cover a part of the decisions that we must make.

5.1 Case 1: Single Camera

Image A **Image B** **Image C** **Image D**

Figure 8: Detecting Left and Right

MSE

	A	B	C	D
A	*S*	*L*	*L*	*L*
B	*R*	*S*	*L*	*L*
C	*R*	*R*	*S*	*L*
D	*R*	*R*	*R*	*S*

Covariance Matrix

	A	B	C	D
A	*S*	*L*	*L*	*L*
B	*R*	*S*	*L*	*L*
C	*R*	*R*	*S*	*L*
D	*R*	*R*	*R*	*S*

Table 1: Detecting Left and Right

Note: S – same field of view L – row is to the left of column

Here, what you see are four separate images, A-D, showing the CAS building and Marsh Plaza of BU. Below it are the results we obtained from using both of our techniques, along with a legend to describe what each symbol means. If you look along the

diagonal of each of these images, we are obviously detecting the same field of view (since we're comparing an image with itself). What is important is that when we compare for example Image A with B, C, or D, we correctly detect that A is the left of them. This works in the opposite direction when we compare B, C, and D with Image A to find that it is to the right.

Image 1

Image 2

Image 3

Figure 9: Detecting Above and Below

MSE

	1	2	3
1	*S*	*B*	*B*
2	*A*	*S*	*B*
3	*A*	*A*	*S*

Covariance Matrix

	1	2	3
1	*S*	*B*	*B*
2	*A*	*S*	*B*
3	*A*	*A*	*S*

Table 2: Detecting Above and Below

Note: A – row is located above the column B– row is located below the column
S – same field of view

Here, we have three adjacent images 1, 2, and 3 which were taken using the PTZ2 camera. Again, we've found that both techniques have correctly found that Image 1 and

below Image 2 and 3, while Image 2 and 3 are above Image 1. These examples show that both the MSE technique and Covariance work well under these circumstances.

MSE Technique

Covariance Matrix Technique

Figure 10: Detecting Above and Below (with Luminance Differences)

MSE

	A1	B1
A1	***S***	***D***
B1	***D***	***S***

Covariance Matrix

	A1	B1
A1	***S***	***B***
B1	***A***	***S***

Table 3: Detecting Above and Below (with luminance Differences)

Note: S – same field of view D – field of view is disjoint

A – row is located above the column B– row is located below the column

Here is an example where the MSE technique fails as opposed to the covariance matrix technique. The reason why MSE believes that the two images are disjoint is because of the large luminance differences between the images. On the other hand, the covariance matrix can still correctly make the decision because the features are still well preserved in each block. Therefore, we can already see that the covariance matrix is more robust, given this example.

5.1 Case 2: Two Cameras

PTZ1

PTZ2

Figure 11: Field of View is the Same

MSE

	PTZ1	PTZ2
PTZ 1	***S***	***D***
PTZ 2	***D***	***S***

Covariance Matrix

	PTZ1	PTZ2
PTZ 1	***S***	***S***
PTZ 2	***S***	***S***

Table 4: Field of View is the Same

Note: S – same field of view D – field of view is disjoint

Here are two images taken from PTZ1 and 2 respectively, both of which are looking across the river at MIT. To no surprise, the MSE approach again failed to detect that the two cameras are looking at the same scene (since the MSE is too large if the objects are just slightly rotated within the image). On the other hand, the covariance matrix is more robust because we have again found that the features will hold in the two images, and we've correctly detected all that all the cases share the same field of view.

PTZ1

PTZ2

Figure 12: Adjacent Images (Left and Right)

MSE

	PTZ1	PTZ2
PTZ 1	*S*	*D*
PTZ 2	*D*	*S*

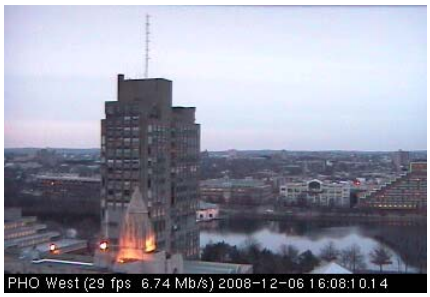
Covariance Matrix

	PTZ1	PTZ2
PTZ 1	*S*	*L*
PTZ 2	*R*	*S*

Table 5: Adjacent Images (Left and Right)

Note: S – same field of view D – field of view is disjoint
L – row is to the left of column R – row is to the right of column

As a more complicated case, we have included one there two cameras are looking at adjacent scenes (PTZ1's image is to the left of PTZ2's image). Again, we see the same problem in the MSE technique; however the covariance matrix still correctly detects it.



PTZ1 PTZ2

Figure 13: Field of View is Disjoint

MSE

	PTZ1	PTZ2
PTZ 1	*S*	*D*
PTZ 2	*D*	*S*

Covariance Matrix

	PTZ1	PTZ2
PTZ 1	*S*	*D*
PTZ 2	*D*	*S*

Table 6: Field of View is Disjoint

Note: S – same field of view D – field of view is disjoint
L – row is to the left of column R – row is to the right of column

Here is the last of our results, showing an example when the two images are completely disjoint. To no surprise, both technique work correctly. This happens primarily because the blocks between the images have significant differences.

Therefore, we can see that from all of our results, we have accomplished the task that was introduced. The next section will provide the conclusions that we have made.

6 Conclusions

We find the following if we look at the advantages of the two above techniques:

In terms of run-time of the actual algorithm, we see that the MSE technique completes in about 0.6 seconds and that the Covariance Matrix technique takes about 2.5 seconds per image. Because of its simplicity, the MSE approach runs more than four times faster than the Covariance Matrix approach. Although they are currently on the order of seconds, applying this for many images will result in much larger time differences.

In terms of accuracy (correct decision making), we find that Covariance Matrix technique is better because it is more robust to rotations, illumination changes, and translations (as we have seen in the results).

Overall, we've seen that the MSE approach works comparably well if a single camera looks at images, but fails miserably under multiple cameras. Although it is considerably faster, we have seen that it is much too sensitive to points like small rotations. Therefore, it appears that the Covariance Matrix approach is our favorable method. Robustness comes before run-time.

We believe that there still exists further room for improvements. Rather than using fixed thresholds for make the decision for dissimilarity, it could be possible to make this value related to the entropy of the images (1000 may work only for our particular cases). Also, rather than just separating the images into N x N blocks with no overlap, we can introduce overlap between the blocks (to gain a better probability of correct detection).

7 References

[1] F. Porikli, O. Tuzel, and P. Meer, "Covariance Tracking using Model Update Based on Lie Algebra", *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 1, pp. 728-735, 2006.

8 Appendix

Below is listed Matlab source code developed for this project. This first section is the code for the Covariance Matrix Approach.

```
%Xuanming Lu, Jason Lin
%EC520
%Final Project: Discovery of camera network topology

close all
clc
clear all

tic
%load images
%L_jpg = imread('Marsh_Plaza_Left.jpg');
%R_jpg = imread('Marsh_Plaza_Right.jpg');
image1 = '1';
image2 = '2';
L_jpg = imread([image1 '.jpg']);
R_jpg = imread([image2 '.jpg']);
L = double(L_jpg);
R = double(R_jpg);

%resize image if they are not already the same size
ratio = prod(size(L))/prod(size(R));
if(ratio ~=1)
    if(ratio>1)
        R2 = interp(R(:),ratio);
        R = reshape(R2,size(L));
    else
        L2 = interp(L(:),1/ratio);
        L = reshape(L2,size(R));
    end
end

%defines all parameters
%N is the length and width of a block
N = 64;
%dissimilarity_threshold is the maximum dissimilarity you will tolerate to
%define two blocks are similar
dissimilarity_threshold = 1;
%evidence_threshold is the minimum percentage of evidence you will tolerate
%to come with a conclusion for left, right, above, or below images
evidence_threshold = 0.50;
%initialize all evidence
evidence_right = 0;
evidence_left = 0;
evidence_above = 0;
evidence_below = 0;
%percentage of tolerable evidence from all directions to consider same
%field of view
same_fov_threshold = 0.20;
```

```
%divide each matrix up into N x N images
dim_L =1;
[m_L,n_L,o_L] = size(L);
%number of blocks in the horizontal and vertical direction of our original
%image
horizontal_blocks_L = floor(n_L/N);
vertical_blocks_L = floor(m_L/N);
A = zeros(N,N,o_L,horizontal_blocks_L*vertical_blocks_L);
for i = 1:N:m_L-N+1,
    for j = 1:N:n_L-N+1,
        A(:,:,:,dim_L) = L(i:i+N-1,j:j+N-1,1:o_L);
        %calculates the covariance matrix of A
        C_A(:,:,dim_L) = covariance_matrix(A(:,:,:,dim_L));
        dim_L = dim_L + 1;
    end
end

dim_R =1;
[m_R,n_R,o_R] = size(R);
%number of blocks in the horizontal and vertical direction of our original
%image
horizontal_blocks_R = floor(n_R/N);
vertical_blocks_R = floor(m_R/N);
B = zeros(N,N,o_R,horizontal_blocks_R*vertical_blocks_R);
for i = 1:N:m_R-N+1,
    for j = 1:N:n_R-N+1,
        B(:,:,:,dim_R) = R(i:i+N-1,j:j+N-1,1:o_R);
        %calculates the covariance matrix of B
        C_B(:,:,dim_R) = covariance_matrix(B(:,:,:,dim_R));
        dim_R = dim_R + 1;
    end
end


%loop through each block A in L and compare it with the respective
%blocks to the left, right, top, and bottom of it
for loop = 1:dim_L-1,
    %compare blocks to the left of current position in A
    for left = loop-mod(loop-1,horizontal_blocks_R):loop,
        D = dissimilarity(C_A(:,:,loop),C_B(:,:,left));
        if(D<dissimilarity_threshold)
            evidence_left = evidence_left+1;
        end
    end
    %compare blocks to the right of current position in A
    for right =
loop:loop+(horizontal_blocks_R-1-mod(loop-1,horizontal_blocks_R)),
        D = dissimilarity(C_A(:,:,loop),C_B(:,:,right));
        if(D<dissimilarity_threshold)
            evidence_right = evidence_right+1;
        end
    end
    %compare blocks above current position in A
    for above = loop:-horizontal_blocks_R:1,
        D = dissimilarity(C_A(:,:,loop),C_B(:,:,above));
```

```
        if(D<dissimilarity_threshold)
            evidence_above = evidence_above+1;
        end
    end
    %compare blocks below current position in A
    for below =
loop:horizontal_blocks_R:horizontal_blocks_R*vertical_blocks_R,
        D = dissimilarity(C_A(:,:,loop),C_B(:,:,below));
        if(D<dissimilarity_threshold)
            evidence_below = evidence_below+1;
        end
    end
end

%obtain conclusion
conclusion = getconclusion(image1, image2, evidence_right, evidence_left,
evidence_above, evidence_below, evidence_threshold, same_fov_threshold);
%plots the results
figure(1);
subplot(1,2,1),
imshow(L_jpg);
title(['Image ' image1 ': ' conclusion])
subplot(1,2,2),
imshow(R_jpg);
title(['Image ' image2 ': ' conclusion])
toc
```

```
%calculate the covariance matrix of each matrix
function C = covariance_matrix(A)
%takes features like x,y,R,G,B, and edge detection
[m_A,n_A,o_A] = size(A);
i = 1:n_A;
j = 1:m_A;
%gets the x and y components
i_component = repmat(j,n_A,1)';
j_component = repmat(i,m_A,1);
%gets the rgb components
A_RGB = reshape(A,[],3);
%gets the edge
A_Edge = edge(rgb2gray(uint8(A)),'canny');
%puts all the components into a set of feature vectors
features = cat(2, j_component(:), A_RGB);
features = cat(2, i_component(:), features);
features = cat(2, features, A_Edge(:));
%obtains the covariance matrix based on those features
C = cov(double(features));
```

```
%calculates the dissimilarity between two blocks
function D = dissimilarity(A,B)
%generalized eigenvalues of A and B
E = eig(A,B);
%compute the dissimilarity between covariance matrices
D = sqrt(sum(log(E).^2));
```

```
%obtain the conclusion (relationship between images) based on our evidence
function conclusion = getconclusion(image1, image2, evidence_right,
evidence_left, evidence_above, evidence_below, evidence_threshold,
same_fov_threshold)
total_evidence = evidence_right + evidence_left + evidence_above +
evidence_below;
%error check for divide by zero(in case the total is equal to 0)
if(total_evidence==0)
    conclusion = ['Field of Views (' image1  ' and ' image2 ') are disjoint'];
elseif (evidence_right/total_evidence > evidence_threshold)
    conclusion = ['Field of Views are Adjacent - ' image1  ' is to the right
of ' image2];
elseif (evidence_left/total_evidence > evidence_threshold)
    conclusion = ['Field of Views are Adjacent - ' image1  ' is to the left
of ' image2];
elseif (evidence_above/total_evidence > evidence_threshold)
    conclusion = ['Field of Views are Adjacent - ' image1  ' is above ' image2];
elseif (evidence_below/total_evidence > evidence_threshold)
    conclusion = ['Field of Views are Adjacent - ' image1  ' is below ' image2];
elseif
((evidence_right>same_fov_threshold*total_evidence)&&(evidence_left>sam
e_fov_threshold*total_evidence)&&(evidence_above>same_fov_threshold*tot
al_evidence)&&(evidence_below>same_fov_threshold*total_evidence))
    conclusion = ['Field of Views (' image1  ' and ' image2 ') are the same'];
else
    conclusion = ['Field of Views (' image1  ' and ' image2 ') are disjoint'];
end
```

This second section is the code for the Mean Squared Error Approach.

```
%Xuanming Lu, Jason Lin
%EC520
%Final Project: Discovery of camera network topology

clc
clear all

tic
%load images
%L_jpg = imread('Marsh_Plaza_Left.jpg');
%R_jpg = imread('Marsh_Plaza_Right.jpg');
image1 = 'A';
image2 = 'B';
L_jpg = imread([image1 '.jpg']);
R_jpg = imread([image2 '.jpg']);
L = double(L_jpg);
R = double(R_jpg);

%resize image if they are not already the same size
ratio = prod(size(L))/prod(size(R));
if(ratio ~=1)
    if(ratio>1)
        R2 = interp(R(:),ratio);
        R = reshape(R2,size(L));
    else
        L2 = interp(L(:),1/ratio);
        L = reshape(L2,size(R));
    end
end

%defines all parameters
%N is the length and width of a block
N = 64;
%dissimilarity_threshold is the maximum dissimilarity you will tolerate to
%define two blocks are similar
dissimilarity_threshold = 1;
%evidence_threshold is the minimum percentage of evidence you will tolerate
%to come with a conclusion for left, right, above, or below images
evidence_threshold = 0.50;
%initialize all evidence
evidence_right = 0;
evidence_left = 0;
evidence_above = 0;
evidence_below = 0;
%percentage of tolerable evidence from all directions to consider same
%field of view
same_fov_threshold = 0.20;

%divide each matrix up into N x N images
dim_L =1;
[m_L,n_L,o_L] = size(L);
%number of blocks in the horizontal and vertical direction of our original
%image
```

```
horizontal_blocks_L = floor(n_L/N);
vertical_blocks_L = floor(m_L/N);
A = zeros(N,N,o_L,horizontal_blocks_L*vertical_blocks_L);
for i = 1:N:m_L-N+1,
    for j = 1:N:n_L-N+1,
        A(:,:,:,dim_L) = L(i:i+N-1,j:j+N-1,1:o_L);
        %calculates the covariance matrix of A
        C_A(:,:,dim_L) = covariance_matrix(A(:,:,:,dim_L));
        dim_L = dim_L + 1;
    end
end

dim_R =1;
[m_R,n_R,o_R] = size(R);
%number of blocks in the horizontal and vertical direction of our original
%image
horizontal_blocks_R = floor(n_R/N);
vertical_blocks_R = floor(m_R/N);
B = zeros(N,N,o_R,horizontal_blocks_R*vertical_blocks_R);
for i = 1:N:m_R-N+1,
    for j = 1:N:n_R-N+1,
        B(:,:,:,dim_R) = R(i:i+N-1,j:j+N-1,1:o_R);
        %calculates the covariance matrix of B
        C_B(:,:,dim_R) = covariance_matrix(B(:,:,:,dim_R));
        dim_R = dim_R + 1;
    end
end



%loop through each block A in L and compare it with the respective
%blocks to the left, right, top, and bottom of it
for loop = 1:dim_L-1,
    %compare blocks to the left of current position in A
    for left = loop-mod(loop-1,horizontal_blocks_R):loop,
        D = dissimilarity(C_A(:,:,loop),C_B(:,:,left));
        if(D<dissimilarity_threshold)
            evidence_left = evidence_left+1;
        end
    end
    %compare blocks to the right of current position in A
    for right =
loop:loop+(horizontal_blocks_R-1-mod(loop-1,horizontal_blocks_R)),
        D = dissimilarity(C_A(:,:,loop),C_B(:,:,right));
        if(D<dissimilarity_threshold)
            evidence_right = evidence_right+1;
        end
    end
    %compare blocks above current position in A
    for above = loop:-horizontal_blocks_R:1,
        D = dissimilarity(C_A(:,:,loop),C_B(:,:,above));
        if(D<dissimilarity_threshold)
            evidence_above = evidence_above+1;
        end
    end
    %compare blocks below current position in A
```

```
    for below =
loop:horizontal_blocks_R:horizontal_blocks_R*vertical_blocks_R,
        D = dissimilarity(C_A(:,:,loop),C_B(:,:,below));
        if(D<dissimilarity_threshold)
            evidence_below = evidence_below+1;
        end
    end
end

%obtain conclusion
conclusion = getconclusion(image1, image2, evidence_right, evidence_left,
evidence_above, evidence_below, evidence_threshold, same_fov_threshold);
%plots the results
figure(1);
subplot(1,2,1),
imshow(L_jpg);
title(['Image ' image1 ': ' conclusion])
subplot(1,2,2),
imshow(R_jpg);
title(['Image ' image2 ': ' conclusion])
toc
```

```
%calculates the dissimilarity between two blocks
function D = dissimilarity(A,B)
%generalized eigenvalues of A and B
E = eig(A,B);
%compute the dissimilarity between covariance matrices
D = sqrt(sum(log(E).^2));
```

```
%obtain the conclusion (relationship between images) based on our evidence
function conclusion = getconclusion(image1, image2, evidence_right,
evidence_left, evidence_above, evidence_below, evidence_threshold,
same_fov_threshold)
total_evidence = evidence_right + evidence_left + evidence_above +
evidence_below;
%error check for divide by zero(in case the total is equal to 0)
if(total_evidence==0)
    conclusion = ['Field of Views (' image1  ' and ' image2 ') are disjoint'];
elseif (evidence_right/total_evidence > evidence_threshold)
    conclusion = ['Field of Views are Adjacent - ' image1  ' is to the right
of ' image2];
elseif (evidence_left/total_evidence > evidence_threshold)
    conclusion = ['Field of Views are Adjacent - ' image1  ' is to the left
of ' image2];
elseif (evidence_above/total_evidence > evidence_threshold)
    conclusion = ['Field of Views are Adjacent - ' image1  ' is above ' image2];
elseif (evidence_below/total_evidence > evidence_threshold)
    conclusion = ['Field of Views are Adjacent - ' image1  ' is below ' image2];
elseif
((evidence_right>same_fov_threshold*total_evidence)&&(evidence_left>sam
e_fov_threshold*total_evidence)&&(evidence_above>same_fov_threshold*tot
al_evidence)&&(evidence_below>same_fov_threshold*total_evidence))
    conclusion = ['Field of Views (' image1  ' and ' image2 ') are the same'];
else
```

```
    conclusion = ['Field of Views (' image1  ' and ' image2 ') are disjoint'];
end
```