**ECE 442**

**Lab #4**

**Video Processing**

**Linjian Xiang**

**1373037**

Q1.

Please see Appendix for plots and code.

Q2.

Please see Appendix for plots and code.

The MAD is 5.9591.

Q3.

Please see Appendix for plots and code.

The MAD is 6.0939.

Q4.

Exhaustive search:

Advantage: If the frames in the video are not moved suddenly the exhaustive search could found the best matched motion vector. Which could create the best compressed image for video transmission.

Disadvantage: The exhaustive search consume the great amount of calculation. It has a very large complexity. Especially, will the searching block is large.

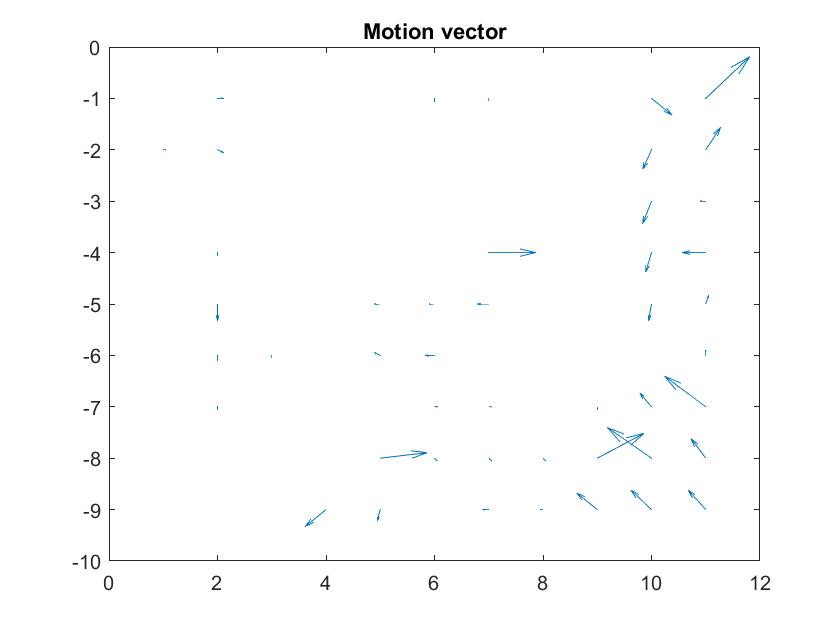
Three step search:

Advantage: Comparing with Exhaustive search, the complexity of three step search is much smaller, which provides much faster coder processor before video transmission. Also it has very close performance compare with Exhaustive search.

Disadvantage: For this lab, Comparing with the Exhaustive search, the three step search has a large MAD, even it is very small. Furthermore, when the step size gets larger the complexity of the calculation doesn’t change, however, the performance will be decrease. Especially when the step size is too large, the three step search cannot get right motion vector.

**Appendix**

Motion vector plot



%--------------------------------------------------------------------------

% EE442 Lab4 Video Processing

% Motion Estimation using Exhaustive Search Function Template

% Author : Adam Harrison

% If it suits you, you may keep this function's layout, which accepts block

% size, N1,N2,and the search range, R, as parameters.

%--------------------------------------------------------------------------

function [u,v]=Exhaustive\_Search(Im\_ref,Im\_cur,N1,N2,Range)

Im\_ref= imread(Im\_ref);

Im\_cur= imread(Im\_cur);

close all

figure(1);

subplot(1,2,1),imshow(Im\_ref);

subplot(1,2,2),imshow(Im\_cur);

Im\_ref=double(Im\_ref);

Im\_cur=double(Im\_cur);

%Find the motion vector using exhaustive search. You may find it beneficial

%to use sub-functions for this portion. This is the biggest task of your

%function.

[row col] = size(Im\_cur);

u = zeros(row/N1,col/N2);

v = zeros(row/N1,col/N2);

counter1 = 0;

counter2 = 0;

min\_value = 999999; % Give a very large initial value

% The searching starts at the top left corner of the image

for i = 1 : N1 : row-N1+1

counter1 = counter1 +1;

for j = 1 : N2 : col-N2+1

counter2 = counter2 +1;

for k = -Range : Range

for t = -Range : Range

Blk\_ref\_Ver = i + k; % row/Vert co-ordinate for ref block

Blk\_ref\_Hor = j + t; % col/Horizontal co-ordinate

if ( Blk\_ref\_Ver < 1 || Blk\_ref\_Ver+N1-1 > row ...

|| Blk\_ref\_Hor < 1 || Blk\_ref\_Hor+N2-1 > col)

continue;

end

temp = costFuncMAD(Im\_cur(i:i+N1-1,j:j+N2-1), ...

Im\_ref(Blk\_ref\_Ver:Blk\_ref\_Ver+N1-1, Blk\_ref\_Hor:Blk\_ref\_Hor+N2-1), Range);

if(temp<min\_value)

min\_value = temp;

temp\_k = k;

temp\_t = t;

end

end

end

%Store the motion vectors in u and v.

%for each sub-block, ie, if you have (rows/N1, columns/N2) sub-blocks, you

%will have the same number of u and v values

u(counter1, counter2) = temp\_k;

v(counter1, counter2) = temp\_t;

min\_value = 100000; % Give a very large initial value

end

counter2 = 0;

end

figure(2);

%show the result using quiver

%call quiver with the very block

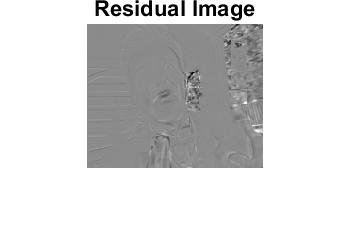
[x,y]=meshgrid([1:11],[-1:-1:-9]);

quiver(x,y,u,-v);

title('Motion vector');

end

Q2



%--------------------------------------------------------------------------

% EE442 Lab4 Video Processing

% Motion Compensation Function Template

% Author : Adam Harrison

% If it suits you, you may keep this function's layout, which accepts block

% size, N1,N2 as parameters.

%--------------------------------------------------------------------------

function [Im\_est Residual total\_MAD]=Motion\_Compensation(u,v,Im\_ref1,Im\_cur1,N1,N2)

Im\_ref=imread(Im\_ref1);

Im\_cur=imread(Im\_cur1);

counter1 = 0; %Use for indexing the row value

counter2 = 0; %Use for indexing the column value

%Compute estimated image based on motion vectors u and v. Store result in

%Im\_est. This is the biggest task of your function.

[row col] = size(Im\_ref);

for i = 1:N1:row-N1+1

counter1 = counter1 +1;

for j = 1:N2:col-N2+1

counter2 = counter2 +1;

dy = u(counter1,counter2);

dx = v(counter1,counter2);

BlkVertical = i + dy;

BlkHorizontal = j + dx;

Im\_est(i:i+N1-1,j:j+N2-1) = Im\_ref(BlkVertical:BlkVertical+N1-1, BlkHorizontal:BlkHorizontal+N2-1);

end

counter2 = 0;

end

%Compute the residual, cast operands to double to allow negative numbers

Residual=double(Im\_cur)-double(Im\_est);

%Cast estimated image to uint8 for display purposes

Im\_est=uint8(Im\_est);

figure(3);

imshow(Im\_est);title('Reconstructed Image');

imwrite(Im\_est,'Reconstructed\_Image.jpg');

figure(4);

imshow(Residual,[]);title('Residual Image');

%reformate the Residual matrix inorder to store a more identifiable

%residual image

Residual\_reformat = Residual - min(min(Residual));

Residual\_reformat = uint8(Residual\_reformat);

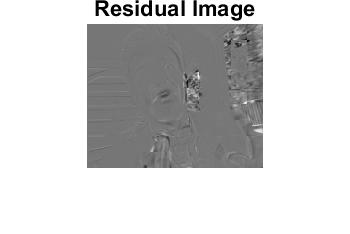
imwrite(uint8(Residual\_reformat),'Residual.jpg');

%Compute the MAD between Im\_est and Im\_cur

%total\_MAD= %(use the appropriate equation here)

total\_MAD = sum(abs(Residual(:)))/(row\*col)

Q3.



%--------------------------------------------------------------------------

% EE442 Lab4 Video Processing

% Motion Estimation using Exhaustive Search Function Template

% Author : Adam Harrison

%--------------------------------------------------------------------------

function [p,q]=Log\_Search(Im\_ref,Im\_cur,N1,N2,Range)

Im\_ref = imread(Im\_ref); %load reference image

Im\_cur = imread(Im\_cur); %load current image

figure(1);

subplot(1,2,1),imshow(Im\_ref);

subplot(1,2,2),imshow(Im\_cur);

Im\_ref=double(Im\_ref);

Im\_cur=double(Im\_cur);

%Find the motion vector using logarithmic search. You may find it beneficial

%to use sub-functions for this portion. This is the biggest task of your

%function.

[row col] = size(Im\_cur);

p = zeros(row/N1,col/N2); %Used to store horizantal moving vectors

q = zeros(row/N1,col/N2); %Used to store vertical moving vectors

counter1 = 0;

counter2 = 0;

% Start off from the top left of the image

% For every block, find a minimum MAD value

% store the motion vector into u and v

for m = 1 : N1 : row-N1+1

counter1 = counter1 +1;

for n = 1 : N2 : col-N2+1

counter2 = counter2 +1;

% three-step search starts

step = Range;

u = 0;

v = 0;

while( step>= 2 )

step = round(step/2);

MAD = inf;

for i=-1:1

for j=-1:1

test\_x = j\*step+u+n;

test\_y = i\*step+v+m;

if(test\_y<1 || test\_y+N1-1>row ...

|| test\_x<1 || test\_x+N2-1>col)

continue;

end

%compute the MAD for two blocks

MAD\_temp = costFuncMAD(Im\_cur(m:m+N1-1,n:n+N2-1), ...

Im\_ref(test\_y:test\_y+N1-1, test\_x:test\_x+N2-1), Range);

if MAD\_temp<MAD

MAD = MAD\_temp;

Temp\_v = i\*step+v;

Temp\_u = j\*step+u;

end

end

end

u = Temp\_u;

v = Temp\_v;

end

%Store the motion vectors in u and v. Note you should have a u and v value

p(counter1, counter2) = v;

q(counter1, counter2) = u;

end

counter2 = 0; %Reset counter2

end

figure(5);

%show the result using quiver

%call quiver with the correct parameters

[x,y]=meshgrid([1:11],[-1:-1:-9]);

quiver(x,y,p,-q);

title('Motion vector');

end