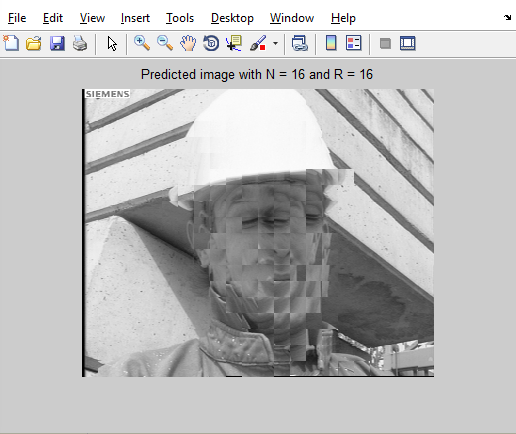
**ECE508 Project 2**

**Video Communications**

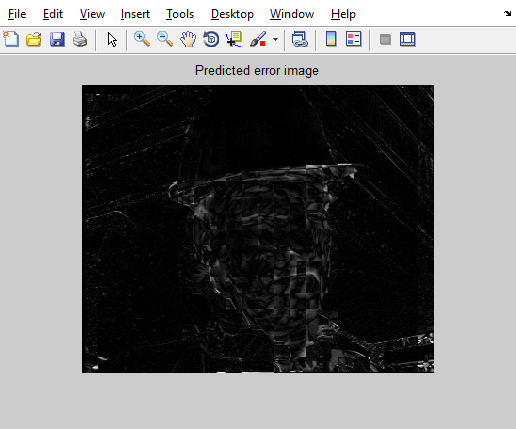
Name: **Krishna Pramod Kanakapura Umesh**

Student ID: **A20337195**

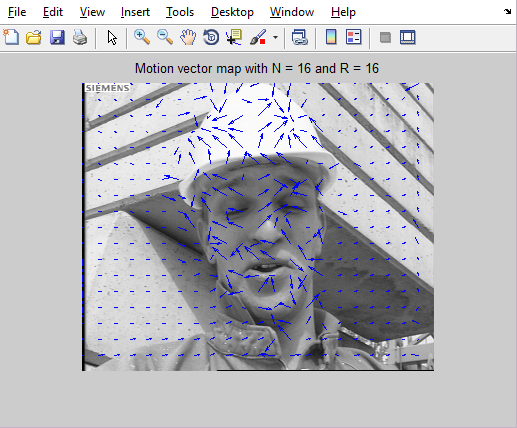
1. For Integer-Pel EBMA:
2. Predicted Image:

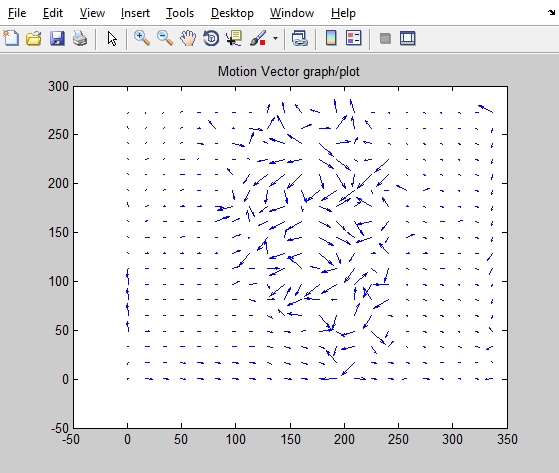


Prediction Error Image:

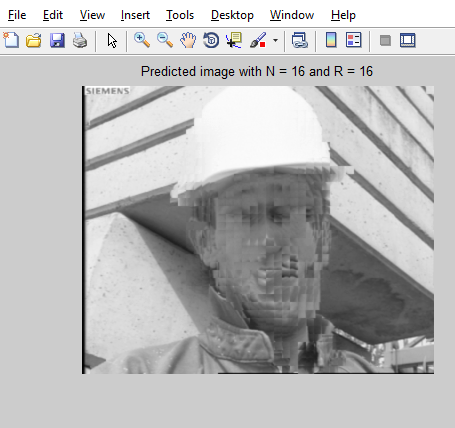


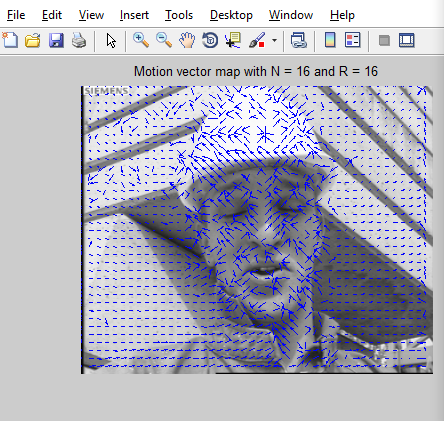
1. The estimated motion field is as shown in figure below:

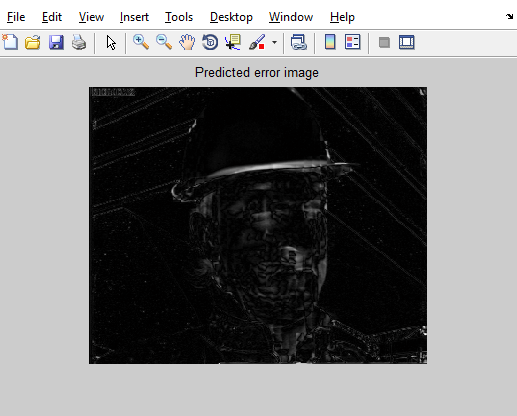




1. PSNR of the Predicted Frame = 27.5221 dB.
2. The PSNR of the predicted frame in Half-pel is 27.8215 dB.







Half-pel EBMA yields more accurate motion field and prediction, but it takes more computation time compared to Integer-Pel. Hence, in applications where time is a constraint, Integer-Pel can be used and Half-pel method can be used for EBMA where accuracy is important.

1. In case of dense motion field, the prediction accuracy reduces since the PSNR= 24.7609dB, but the computation time is faster when compared with both the Integer-Pel and Half-Pel.

Source Code:

function IntegerPel\_EBMA(N, R, Fract, pix\_flag)

% For Integer Pel Fract = 1

% N - Block size of the image

% R - Search Range from which the user can select a particular range

% Fract - Fractional Pel size

% pix\_flag - Interpolated Pixel based dense motion estimation flag.

clc;

%close all;

%clear all;

if isequal(Fract,1)

pix = pix\_flag;

else

pix = 0;

if pix\_flag

warning('Pixel-based dense motion estimation omitted')

end

end

W=352; % Width of the given frame

H=288; % Height of the given frame

f2\_org=int16(fread(fopen('foreman66.Y','r'),[W,H])');

f1\_org=int16(fread(fopen('foreman72.Y','r'),[W,H])');

figure(1);

imshow((f2\_org),[]); % Display the original image

title('Anchor Image')

figure(2);

imshow((f1\_org),[]); % Display the target image

title('Target Image')

t = 0;

if Fract >1

tic

f1 = imresize(f1\_org, Fract,'bilinear');

f2 = imresize(f2\_org, Fract,'bilinear');

t = toc;

else

f1 = f1\_org;

f2 = f2\_org;

end

f2\_extnd = wextend(2,'zpd',f2,R);

F1 = nonoverlap(f1, N);

F2 = overlap(f2\_extnd, N);

dx = struct('val',[],'idx',[]);

dy = struct('val',[],'idx',[]);

%% EBMA Algorithm given in class lecture

d = -R:R;

tic

for m = 1:size(f1,1)/N

for n = 1:size(f1,2)/N

MAD\_min = 256\*N\*N;

search\_lmt = 2\*R ;

for k = 1:search\_lmt

for l = 1:search\_lmt

MAD = sum(sum(abs(F1{m,n} - F2{(m-1)\*N +k +1, (n-1)\*N +l +1})));

if MAD<MAD\_min

MAD\_min = MAD;

dx.val = d(l);

dy.val = d(k);

dx.idx = l;

dy.idx = k;

end

end

end

FP{m,n} = F2{(m-1)\*N +1 +dy.idx, (n-1)\*N +1 +dx.idx};

mvx{m,n} = int16(dx.val);

mvy{m,n} = dy.val;

end

end

t = toc +t;

fp = cell2mat(FP);

if Fract > 1

disp(['Time taken by fractional-pel motion estimation = ' num2str(t)])

fp = imresize(fp,1/Fract,'bilinear');

else

disp(['Time taken by integer-pel motion estimation = ' num2str(t)])

end

figure(3);

imshow(f2,[]);

hold

[x1,x2] = meshgrid(1:N:size(f2,2), 1:N:size(f2,1));

quiver(x1,x2, cell2mat(mvx),cell2mat(mvy));

title(['Motion vector map with N = ' num2str(N) ' and R = ' num2str(R)])

% Predicted Image

figure(4);

imshow((fp),[]);

title(['Predicted image with N = ' num2str(N) ' and R = ' num2str(R)])

% Motion Vector Graph/Plot

figure(5);

quiver(x1,x2,flipud(cell2mat(mvx)),flipud(cell2mat(mvy)));

title('Motion Vector graph/plot')

% Predicted Error Image

figure(6);

imshow((abs(f1\_org-fp)),[]);

title('Predicted error image')

% PSNR calculation

PSNR = 10\*log10(255\*255/mean(mean((abs(f1\_org-fp)).^2)));

disp(['The PSNR of the predicted frame = ' num2str(PSNR)])

if pix && (N >1)

tic

MVX = bilinear\_func(int16(cell2mat(mvx)), N);

MVY = bilinear\_func(int16(cell2mat(mvy)), N);

for m = 1:size(f1,1)

for n = 1:size(f1,2)

dx.val = MVX(m,n);

dy.val = MVY(m,n);

dx.idx = find(dx.val==d);

dy.idx = find(dy.val==d);

FP\_pix{m,n} = f2\_extnd((m-1) +1 +dy.idx, (n-1) +1 +dx.idx);

end

end

fp\_pix = cell2mat(FP\_pix);

t\_p = toc;

disp(['Time taken by interpolated pixel-based dense motion estimation = ' num2str(t\_p)])

figure(7);

imshow(f2,[]);

hold

[x1,x2] = meshgrid(1:size(f2,2), 1:size(f2,1));

quiver(x1,x2, (MVX),(MVY));

title('Dense Pixel-based motion vector map')

figure(8);

imshow((fp\_pix),[]);

title('Dense Pixel-based predicted image')

figure(9);

imshow((abs(f1\_org-fp\_pix)),[]);

title('Predicted error image')

% PSNR calculation

PSNR = 10\*log10(255\*255/mean(mean((abs(f1\_org-fp\_pix)).^2)));

disp(['The PSNR of the pixel based predicted frame = ' num2str(PSNR)])

end

end

function F = nonoverlap(f, N)

%% Creating non overlaping NxN blocks

row\_s = 1;

row\_e = N;

blk\_r = 1;

while (row\_e <= size(f,1))

col\_s = 1;

col\_e = N;

blk\_c = 1;

while (col\_e <= size(f,2))

F{blk\_r, blk\_c} = f(row\_s:row\_e, col\_s:col\_e);

col\_s = col\_s +N;

col\_e = col\_e +N;

blk\_c = blk\_c +1;

end

row\_s = row\_s +N;

row\_e = row\_e +N;

blk\_r = blk\_r +1;

end

end

function F = overlap(f, N)

%% Creating overlaping RxR search region blocks

pel\_inc = 1;

row\_s = 1;

row\_e = N;

blk\_r = 1;

while (row\_e <= size(f,1))

col\_s = 1;

col\_e = N;

blk\_c = 1;

while (col\_e <= size(f,2))

F{blk\_r, blk\_c} = f(row\_s:row\_e, col\_s:col\_e);

col\_s = col\_s +pel\_inc;

col\_e = col\_e +pel\_inc;

blk\_c = blk\_c +1;

end

row\_s = row\_s +pel\_inc;

row\_e = row\_e +pel\_inc;

blk\_r = blk\_r +1;

end

end

function Y = bilinear\_func(X, N)

%% Bilinear Interpolation function

X = wextend('2','sym',X,1);

Y\_temp = interp(X', N);

Y\_temp = interp(Y\_temp', N);

Y = Y\_temp(N/2 +1:size(Y\_temp,1) -N/2, N/2 +1:size(Y\_temp,2) -N/2);

end

function Y = interp(X, N)

for m = 1:size(X,1)

for n = 1:N:(size(X,2)\*N -N)

idx = floor((n-1)/N)+1;

Y(m,n:(n+N -1)) = int16(double(X(m, idx +1) - X(m, idx))\*((0:N -1)/(N))) + X(m, idx);

end

end

end

For Integer- Pel without dense motion:

IntegerPel\_EBMA(16,16,1,0)

With Dense motion:

Pix\_flag = 1

For Half-Pel without dense:

function HalfPel\_EBMA(N, R, Fract, pix\_flag)

% For Half Pel Fract = 2

% N - Block size of the image

% R - Search Range from which the user can select a particular range

% Fract - Fractional Pel size

% pix\_flag - Interpolated Pixel based dense motion estimation flag.

clc;

%close all;

%clear all;

if isequal(Fract,2)

pix = pix\_flag;

else

pix = 0;

if pix\_flag

warning('Pixel-based dense motion estimation omitted')

end

end

W=352; % Width of the given frame

H=288; % Height of the given frame

f2\_org=int16(fread(fopen('foreman66.Y','r'),[W,H])');

f1\_org=int16(fread(fopen('foreman72.Y','r'),[W,H])');

figure(1);

imshow((f2\_org),[]); % Display the original image

title('Anchor Image')

figure(2);

imshow((f1\_org),[]); % Display the target image

title('Target Image')

t = 0;

if Fract >1

tic

f1 = imresize(f1\_org, Fract,'bilinear');

f2 = imresize(f2\_org, Fract,'bilinear');

t = toc;

else

f1 = f1\_org;

f2 = f2\_org;

end

f2\_extnd = wextend(2,'zpd',f2,R);

F1 = nonoverlap(f1, N);

F2 = overlap(f2\_extnd, N);

dx = struct('val',[],'idx',[]);

dy = struct('val',[],'idx',[]);

%% EBMA Algorithm given in class lecture

d = -R:R;

tic

for m = 1:size(f1,1)/N

for n = 1:size(f1,2)/N

MAD\_min = 256\*N\*N;

search\_lmt = 2\*R ;

for k = 1:search\_lmt

for l = 1:search\_lmt

MAD = sum(sum(abs(F1{m,n} - F2{(m-1)\*N +k +1, (n-1)\*N +l +1})));

if MAD<MAD\_min

MAD\_min = MAD;

dx.val = d(l);

dy.val = d(k);

dx.idx = l;

dy.idx = k;

end

end

end

FP{m,n} = F2{(m-1)\*N +1 +dy.idx, (n-1)\*N +1 +dx.idx};

mvx{m,n} = int16(dx.val);

mvy{m,n} = dy.val;

end

end

t = toc +t;

fp = cell2mat(FP);

if Fract > 1

disp(['Time taken by fractional-pel motion estimation = ' num2str(t)])

fp = imresize(fp,1/Fract,'bilinear');

else

disp(['Time taken by integer-pel motion estimation = ' num2str(t)])

end

figure(3);

imshow(f2,[]);

hold

[x1,x2] = meshgrid(1:N:size(f2,2), 1:N:size(f2,1));

quiver(x1,x2, cell2mat(mvx),cell2mat(mvy));

title(['Motion vector map with N = ' num2str(N) ' and R = ' num2str(R)])

% Predicted Image

figure(4);

imshow((fp),[]);

title(['Predicted image with N = ' num2str(N) ' and R = ' num2str(R)])

% Motion Vector Graph/Plot

figure(5);

quiver(x1,x2,flipud(cell2mat(mvx)),flipud(cell2mat(mvy)));

title('Motion Vector graph/plot')

% Predicted Error Image

figure(6);

imshow((abs(f1\_org-fp)),[]);

title('Predicted error image')

% PSNR calculation

PSNR = 10\*log10(255\*255/mean(mean((abs(f1\_org-fp)).^2)));

disp(['The PSNR of the predicted frame = ' num2str(PSNR)])

if pix && (N >1)

tic

MVX = bilinear\_func(int16(cell2mat(mvx)), N);

MVY = bilinear\_func(int16(cell2mat(mvy)), N);

for m = 1:size(f1,1)

for n = 1:size(f1,2)

dx.val = MVX(m,n);

dy.val = MVY(m,n);

dx.idx = find(dx.val==d);

dy.idx = find(dy.val==d);

FP\_pix{m,n} = f2\_extnd((m-1) +1 +dy.idx, (n-1) +1 +dx.idx);

end

end

fp\_pix = cell2mat(FP\_pix);

t\_p = toc;

disp(['Time taken by interpolated pixel-based dense motion estimation = ' num2str(t\_p)])

figure(7);

imshow(f2,[]);

hold

[x1,x2] = meshgrid(1:size(f2,2), 1:size(f2,1));

quiver(x1,x2, (MVX),(MVY));

title('Dense Pixel-based motion vector map')

figure(8);

imshow((fp\_pix),[]);

title('Dense Pixel-based predicted image')

figure(9);

%imshow(abs(f1\_org-fp\_pix),[]);

title('Predicted error image')

% PSNR calculation

PSNR = 10\*log10(255\*255/mean(mean((abs(f1\_org-fp\_pix)).^2)));

disp(['The PSNR of the pixel based predicted frame = ' num2str(PSNR)])

end

end

function F = nonoverlap(f, N)

%% Creating non overlaping NxN blocks

row\_s = 1;

row\_e = N;

blk\_r = 1;

while (row\_e <= size(f,1))

col\_s = 1;

col\_e = N;

blk\_c = 1;

while (col\_e <= size(f,2))

F{blk\_r, blk\_c} = f(row\_s:row\_e, col\_s:col\_e);

col\_s = col\_s +N;

col\_e = col\_e +N;

blk\_c = blk\_c +1;

end

row\_s = row\_s +N;

row\_e = row\_e +N;

blk\_r = blk\_r +1;

end

end

function F = overlap(f, N)

%% Creating overlaping RxR search region blocks

pel\_inc = 1;

row\_s = 1;

row\_e = N;

blk\_r = 1;

while (row\_e <= size(f,1))

col\_s = 1;

col\_e = N;

blk\_c = 1;

while (col\_e <= size(f,2))

F{blk\_r, blk\_c} = f(row\_s:row\_e, col\_s:col\_e);

col\_s = col\_s +pel\_inc;

col\_e = col\_e +pel\_inc;

blk\_c = blk\_c +1;

end

row\_s = row\_s +pel\_inc;

row\_e = row\_e +pel\_inc;

blk\_r = blk\_r +1;

end

end

function Y = bilinear\_func(X, N)

%% Bilinear Interpolation function

X = wextend('2','sym',X,1);

Y\_temp = interp(X', N);

Y\_temp = interp(Y\_temp', N);

Y = Y\_temp(N/2 +1:size(Y\_temp,1) -N/2, N/2 +1:size(Y\_temp,2) -N/2);

end

function Y = interp(X, N)

for m = 1:size(X,1)

for n = 1:N:(size(X,2)\*N -N)

idx = floor((n-1)/N)+1;

Y(m,n:(n+N -1)) = int16(double(X(m, idx +1) - X(m, idx))\*((0:N -1)/(N))) + X(m, idx);

end

end

end

Without dense motion

HalfPel\_EBMA(16,16,2,0)

With dense motion

HalfPel\_EBMA(16,16,2,1)