CompE565, Semester 2024
HW3

Prepared by
Brent Son | bson3231@sdsu.edu | 827108705
Kaia Ralston | kralston1127@sdsu.edu | 825033242

Electrical and Computer Engineering
San Diego State University

Table of Contents

Introduction

Video compression is used in everyday media so ensuring efficient storage and transmission of video data is crucial to advancing technologies. Many times the similarity between successive frames causes temporal redundancy. Motion estimation is a technique utilized in this assignment that identifies and exploits this redundancy by detecting the movement of objects between frames.

We implemented a Full Search Block using a macroblock approach in this assignment. Each frame is divided into small blocks (16x16 pixels), and for each block, a search is done to find the best match in a reference frame within the defined search window (32x32 pixels). We use the Sum of Absolute Difference method to decide which frames are matching and which are not. By doing this, we can reduce the amount of data needed to represent a frame.

Since the human eye is more sensitive to brightness than color, there is a focus on the luminance component (Y) of the video in 4:2:0 format. The structure follows the intra-coded frame (I-frame) where each P-frame is predicted from the preceding I-frame.

Procedural Section

Video Processing and Frame Extraction

The provided video was accessed using MATLAB's "VideoReader" class. Once read, frames #11 through #15 were extracted from the video, converted from RGB to YCbCr color space, and the luminance component was processed. Frame #11 is treated as the I-frame and the rest are the P-frames.

Motion Estimation and Compensation

Each macroblock in the P-frames has a motion vector that we can find using the full search motion estimation algorithm. The SAD method can calculate the distortion between the macroblocks of the I-frame and the corresponding blocks in the search window of the P-frames. The two methods work together where the algorithm determines motion vectors that yield the smallest SAD value.

Reconstruction of P-Frames

To reconstruct frames, we reverse the process of predicting. For each P-frame, there is a difference frame that can be created by subtracting the motion-predicted frame from the current frame. The reconstructed frame can then be obtained by adding the difference frame to the predicted frame.

Visualization of Motion Vectors

After the motion vectors are computed for each macroblock, the vectors get plotted on the corresponding P-frame. The 'quiver' function takes the grid of starting points (x,y) and the motion vector components (u,v) to produce a field of arrows. The direction of the arrow is the direction in which the macroblock has moved from the reference frame (I-frame) to the current frame (P-frame) and a longer arrow implies a larger motion.

Results

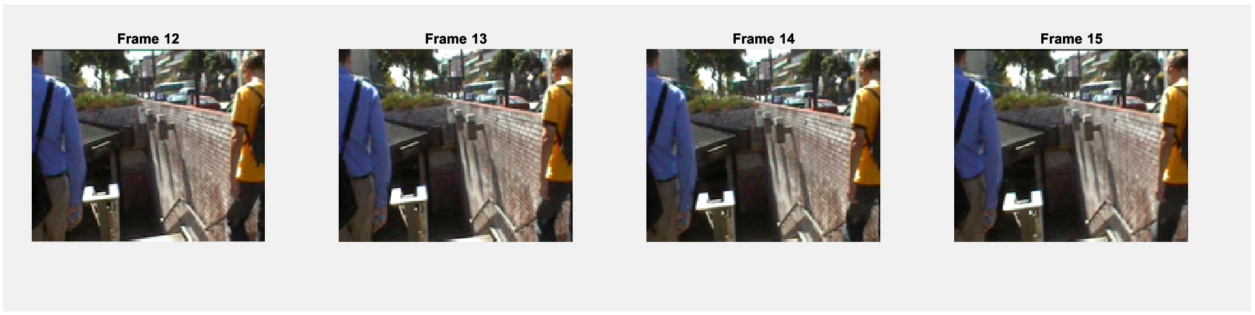


Figure 1: Target Frames for comparing the reconstructed frames with the motion estimated MBs.

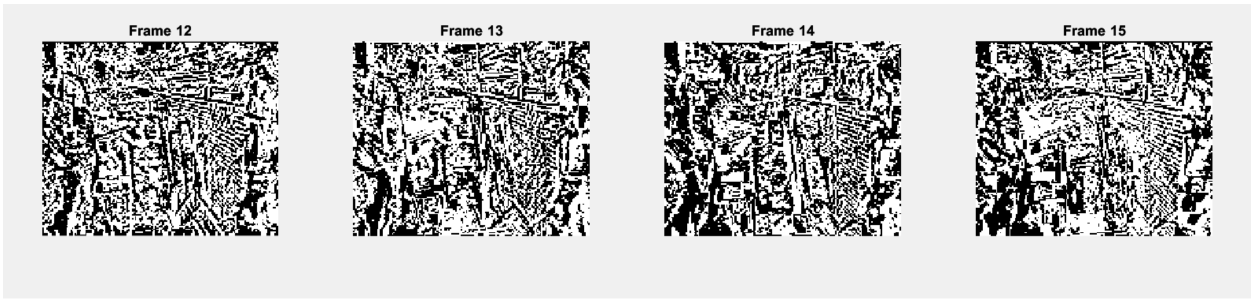


Figure 2: Visualization of the motion difference from the P frames to the I frame, or Frame 11.

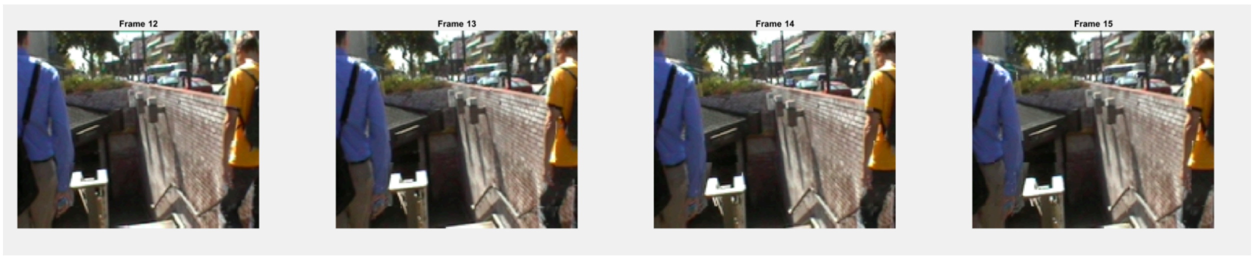
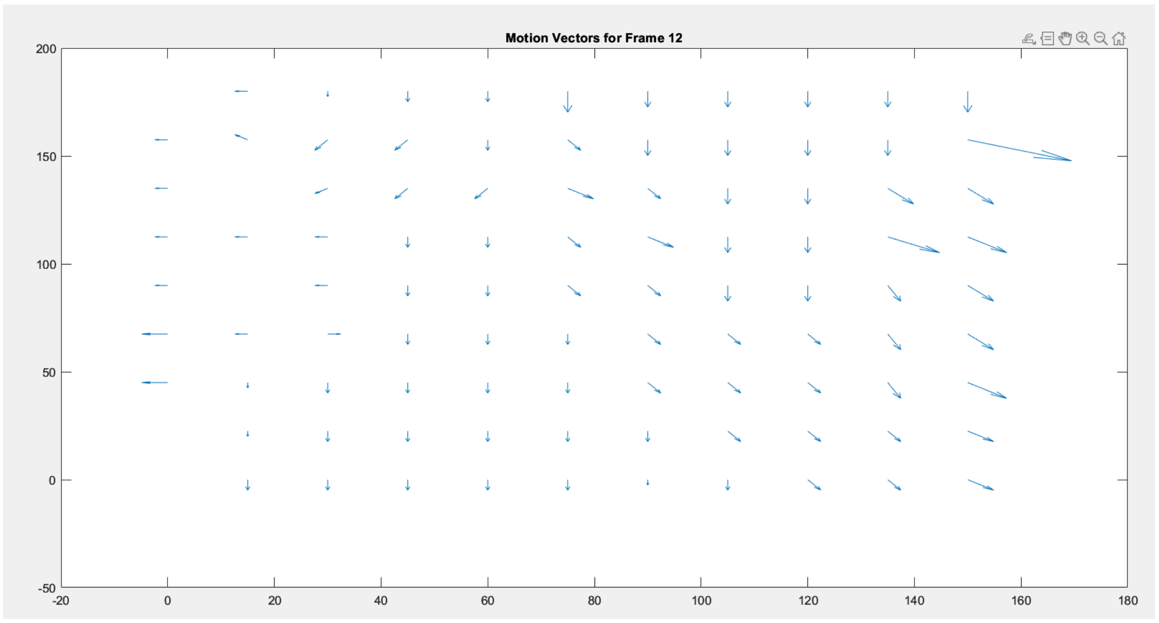
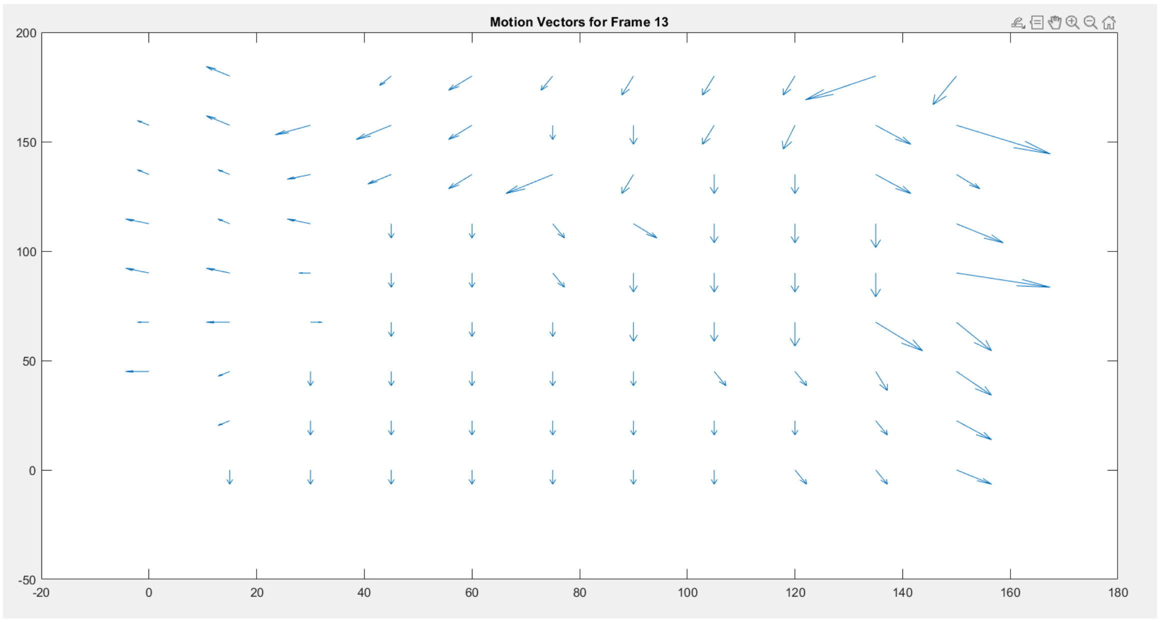
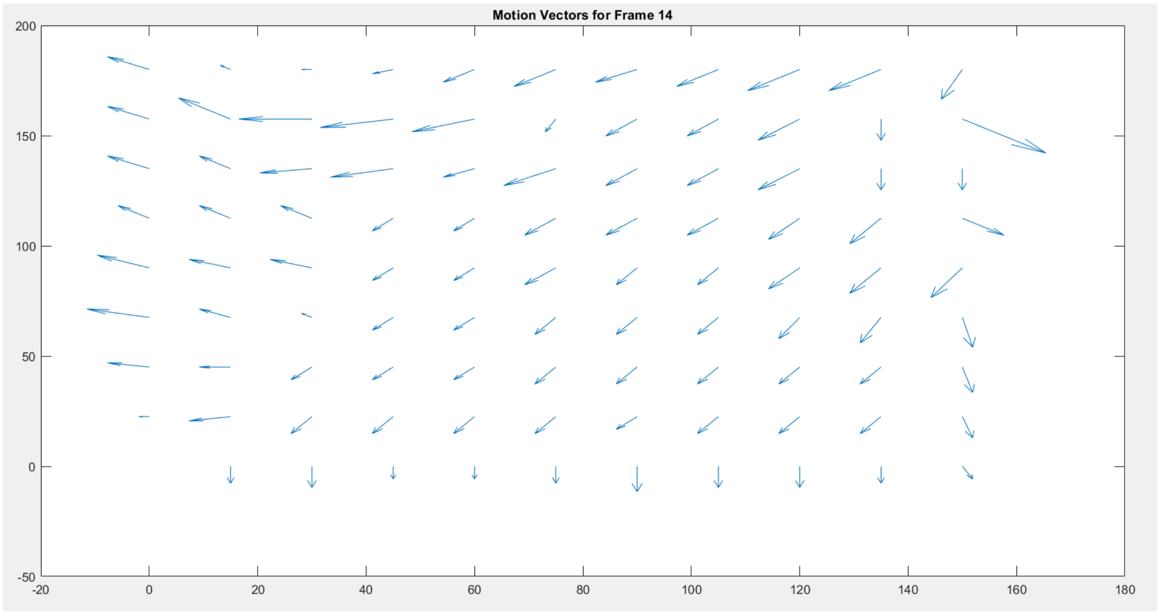


Figure 3: Reconstructed Frames using the original Cb and Cr with the Y component using a modified I frame using motion vectors calculated using SAD.







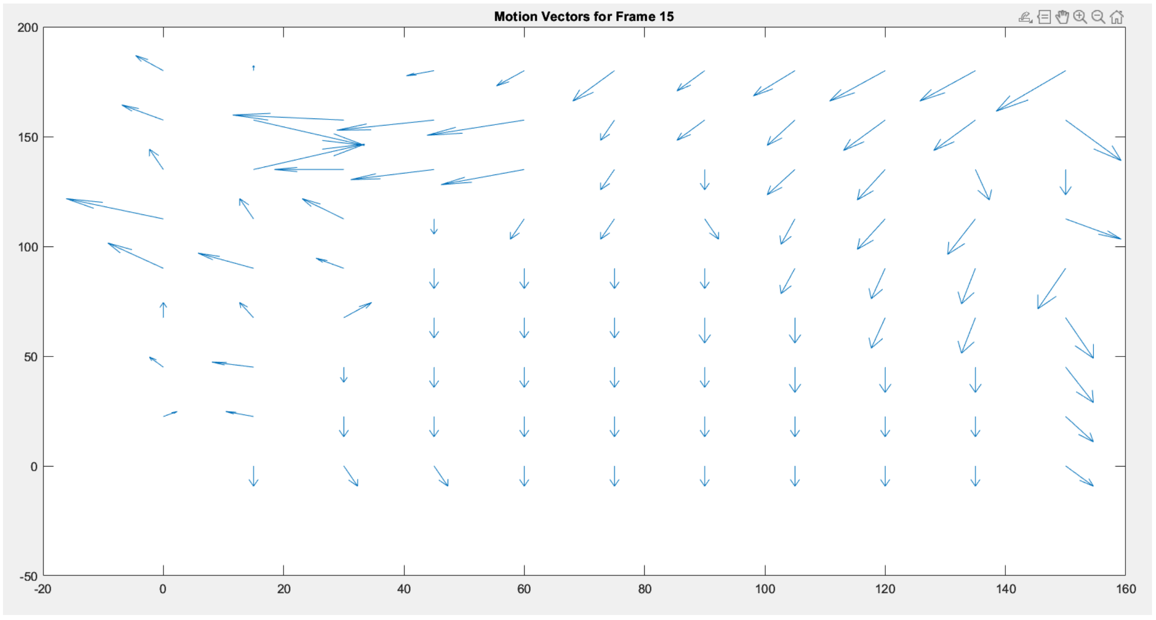


Figure 4: Motion Vector graphs to show the motion difference of the given P frame to the I frame, or Frame 11.

Conclusion

This assignment demonstrated the implementation and efficiency of full search motion estimation in reducing temporal redundancy in video media. While this technique is computationally demanding, it offers high precision in the prediction of motion, which is a vital quality to have in video compression.

In today's media, digital content creation is growing drastically and demands for efficient video compression are increasing. Services like online video streaming, video conferencing, and virtual reality applications rely on capable video compression algorithms in real time. Motion estimation serves as a tool for developers, but also acts as quality assurance for providers to maintain the integrity of video playback.

Source Code

```
%Closing and resetting for new run
close all;
% Parameters
fileName = 'walk_qcif.avi';
frameStart = 11;
frameEnd = 15;
blockSize = 16; % Macroblock size (16x16 pixels)
searchWindow = 32; % Search window size (32x32 pixels)
frameSize = [176, 144]; % QCIF frame size
% Read the video file
v = VideoReader(fileName);
% Initialization
refFrameIndex = frameStart;
motionVectors = cell(frameEnd-frameStart, 1);
reconstructedFrames = cell(frameEnd-frameStart, 1);
targetFrames = cell(frameEnd-frameStart,1);
differenceFrames = cell(frameEnd-frameStart, 1);
% Extract and store the I-frame (reference frame)
refFrame = read(v, refFrameIndex);
refFrameYCbCr = rgb2ycbcr(refFrame);
I_Y = refFrameYCbCr(:,:,1);
% Loop through the P-frames
for k = frameStart+1:frameEnd
   % Read the current frame and convert to YCbCr
   curFrame = read(v, k);
   targetFrames{k-frameStart} = curFrame;
   curFrameYCbCr = rgb2ycbcr(curFrame);
   curFrameCb = curFrameYCbCr(:,:,2);
   curFrameCr = curFrameYCbCr(:,:,3);
   P_Y = curFrameYCbCr(:,:,1);

   % Motion estimation (Full Search)
   [motionVec, diffFrame, reconFrame] = fullSearchME(I_Y, P_Y, blockSize,
searchWindow);

   % Store the results
   motionVectors{k-frameStart} = motionVec;
   differenceFrames{k-frameStart} = diffFrame;
   %Get the Cb, Cr values from the current frame and reconstruct with
   %reconstructed frame
   reconstructedFrame = cat(3, reconFrame, curFrameCb, curFrameCr);
   rgbReconstructedFrame = ycbcr2rgb(reconstructedFrame);
   reconstructedFrames{k - frameStart} = rgbReconstructedFrame;
end
%Display Target Frames
figure(1);
subplot(1,4,1), imshow(targetFrames{1});
```

```
title("Frame 12")
subplot(1,4,2), imshow(targetFrames{2});
title("Frame 13")
subplot(1,4,3), imshow(targetFrames{3});
title("Frame 14")
subplot(1,4,4), imshow(targetFrames{4});
title("Frame 15")
%Display the Difference Frames
figure(2);
subplot(1,4,1), imshow(differenceFrames{1});
title("Frame 12");
subplot(1,4,2), imshow(differenceFrames{2});
title("Frame 13");
subplot(1,4,3), imshow(differenceFrames{3});
title("Frame 14");
subplot(1,4,4), imshow(differenceFrames{4});
title("Frame 15");
%Display Reconstructed Frames
figure(3);
subplot(1,4,1), imshow(reconstructedFrames{1});
title("Frame 12");
subplot(1,4,2), imshow(reconstructedFrames{2});
title("Frame 13");
subplot(1,4,3), imshow(reconstructedFrames{3});
title("Frame 14")
subplot(1,4,4), imshow(reconstructedFrames{4});
title("Frame 15");
%Motion vector plots
U1 = motionVectors{1,1}(:,:,1);
V1 = motionVectors{1,1}(:,:,2);
U2 = motionVectors{2,1}(:,:,1);
V2 = motionVectors{2,1}(:,:,2);
U3 = motionVectors{3,1}(:,:,1);
V3 = motionVectors{3,1}(:,:,2);
U4 = motionVectors{4,1}(:,:,1);
V4 = motionVectors{4,1}(:,:,2);
%[X, Y] = meshgrid(size(motionVectors{1,1}(:,:,1)));
% Define the range for x and y coordinates
x_range = linspace(0, 150, 11);  % 11 points from 0 to 10
y_range = linspace(0, 180, 9);    % 9 points from 0 to 8
% Create the mesh grid
[X, Y] = meshgrid(x_range, y_range);
figure(4);
quiver(X, Y, U1, V1);
title("Motion Vectors for Frame 12");
figure(5);
quiver(X, Y, U2, V2);
title("Motion Vectors for Frame 13");
figure(6);
```

```
quiver(X, Y, U3, V3);
title("Motion Vectors for Frame 14");
figure(7);
quiver(X, Y, U4, V4);
title("Motion Vectors for Frame 15");
% Full Search Motion Estimation Function
function [motionVec, diffFrame, reconFrame] = fullSearchME(refY, curY,
blockSize, searchWindow)
   [rows, cols] = size(refY);
   motionVec = zeros(rows/blockSize, cols/blockSize, 2);
   diffFrame = zeros(size(curY));
   reconFrame = zeros(size(curY));

   % Define search range
   range = (searchWindow - blockSize) / 2;

   for m = 1:blockSize:rows
       for n = 1:blockSize:cols
           bestSAD = inf;
           for x = -range:range
               for y = -range:range
                   if (m+y > 0 && m+y+blockSize-1 <= rows) && (n+x > 0 &&
n+x+blockSize-1 <= cols)
                       % Current and reference blocks
                       curBlock = curY(m:m+blockSize-1, n:n+blockSize-1);
                       refBlock = refY(m+y:m+y+blockSize-1,
n+x:n+x+blockSize-1);

                       % Calculate SAD
                       SAD = sum(sum(abs(double(curBlock) -
double(refBlock))));
                       %Determine if calculated SAD is best SAD
                       if SAD < bestSAD
                           bestSAD = SAD;
                           motionVec((m-1)/blockSize+1, (n-1)/blockSize+1, :) =
[y, x];
                       end
                   end
               end
           end
           %Set the motion vectors to be applied to the reference frame
           dy = motionVec((m-1)/blockSize+1, (n-1)/blockSize+1, 1);
           dx = motionVec((m-1)/blockSize+1, (n-1)/blockSize+1, 2);

           %Construct the reconstructed block
           reconBlock = refY(m+dy:m+dy+blockSize-1, n+dx:n+dx+blockSize-1);
           %Construct the difference block
           curBlock = curY(m:m+blockSize-1, n:n+blockSize-1);
           diffBlock = curBlock - reconBlock;
           % Store the difference block and reconstructed block
           diffFrame(m:m+blockSize-1, n:n+blockSize-1) = diffBlock;
           reconFrame(m:m+blockSize-1, n:n+blockSize-1) = reconBlock +
diffBlock;
       end
   end
end
```