MULTIMEDIA COMMUNICATION SYSTEMS

ECE 434

Computer Project 2

Compare the motion analysis of a digital video stream using the following methods:

(a) Optical Flow.
(b) Displaced Frame Difference.

Perform each method for the following applications:

(1) Pixel-based motion analysis.
(2) Block-based motion analysis.
(3) Region-based motion analysis.
(4) Global motion analysis.

Kai Zhao, 670720413

Shao-Lun Chien, 653414377

ECE 434 Multimedia Systems

Project 02 Motion Analysis

2013 November 10

Compare the motion analysis of a digital video stream using the following methods:

(a) Optical Flow.
(b) Displaced Frame Difference.

Perform each method for the following applications:

(1) Pixel-based motion analysis.
(2) Block-based motion analysis.
(3) Region-based motion analysis.
(4) Global motion analysis.

Motion Estimation Criterion

- To minimize the displaced frame difference (DFD)

$$E_{\mathrm{DFD}}(\mathbf{a}) = \sum_{x \in \Lambda} \left| \psi_2(\mathbf{x} + \mathbf{d}(\mathbf{x};\mathbf{a})) - \psi_1(\mathbf{x}) \right|^p \rightarrow \min$$

$$p = 1 : \mathrm{MAD}; \quad P = 2 : \mathrm{MSE}$$

- To satisfy the optical flow equation

$$E_{\mathrm{OF}}(\mathbf{a}) = \sum_{x \in \Lambda} \left| \left(\nabla \psi_1(\mathbf{x})\right)^T \mathbf{d}(\mathbf{x};\mathbf{a}) + \psi_2(\mathbf{x}) - \psi_1(\mathbf{x}) \right|^p \rightarrow \min$$

Optical Flow on Pixel-based motion analysis

The motion vectors will depends the change in intensity of the placement, the old pixel, and the new pixel. For Pixel-based motion analysis, the change in intensity from motion is irrelevant because motion cannot be detected in any single pixel because it is not sure which pixel corresponds to which for most multimedia. Therefore, optical flow on pixel-based motion analysis will be the same as displaced frame difference on pixel-based motion (see next section).

Displaced Frame Difference on Pixel-based motion analysis

DFD motion analysis is done on atrium.avi, a video in the MATLAB library. It shows people walking around an atrium.

```
clear all;
% select the video to do motion analysis
video = VideoReader('atrium.avi');
% define video properties
vF = video.NumberOfFrames;
vH = video.Height;
vW = video.Width;
video = read(video);
numberOfFrames = 31;            % do motion analysis on last 30 frames
% extract RGB componenet to convert to luminance
videoR = video(:, :, 1, vF-numberOfFrames+1:vF);
videoG = video(:, :, 2, vF-numberOfFrames+1:vF);
videoB = video(:, :, 3, vF-numberOfFrames+1:vF);
video = 0.299*videoR + 0.587*videoG + 0.114*videoB;
video = reshape(video, vH, vW, numberOfFrames);
% DEFINE BLOCK HEIGHT. Set to 1, 1 for pixel-based; vH, vW for global; and
% anything between the two for block-based
blockHeight = 1;
blockWidth = 1;
searchRadius = 1;               % set higher to search more globally
numberOfBlocksAlongY = floor(vH/blockHeight);
numberOfBlocksAlongX = floor(vW/blockWidth);
numberOfBlocks = numberOfBlocksAlongY*numberOfBlocksAlongX;
blocks = uint8(zeros([blockHeight blockWidth numberOfBlocks]));
objectLocation = zeros([2 numberOfBlocks]);
encodedVideo = uint8(zeros([vH vW numberOfFrames-1]));
for t=1:numberOfFrames-1
    t                           % print out the frame that it is working on
    nextFrame = video(:, :, t);
    for i = 1:numberOfBlocksAlongY
        for j = 1:numberOfBlocksAlongX
            blocks(:, :, (i-1)*numberOfBlocksAlongX+mod(j-1,
numberOfBlocksAlongX+1)+1) = video((i-1)*blockHeight+1:i*blockHeight, (j-
1)*blockWidth+1:j*blockWidth, t);
        end
    end
    for b = 1:numberOfBlocks
        newStartingY = floor((b-1)/numberOfBlocksAlongX)*blockHeight+1;
        newStartingX = mod(b-1, numberOfBlocksAlongX)*blockWidth+1;
        minMSE = inf;
        % create a backup in case a better location is found for the block
        nextFrameBackup =  nextFrame;
        for i = newStartingY-searchRadius:newStartingY+searchRadius
            for j = newStartingX-searchRadius:newStartingX+searchRadius
                % search within the specified search region for comparison
                if (i < 1)
                    beginingI = 1;
                    beginingBlockI = 1 - i + 1;
                else
                    beginingI = i;
                    beginingBlockI = 1;
                end
```

```
                if (i+blockHeight-1 > vH)
                    endingI = vH;
                    endingBlockI = (vH - i) + 1;
                else
                    endingI = i+blockHeight-1;
                    endingBlockI = blockHeight;
                end
                if (j < 1)
                    beginingJ = 1;
                    beginingBlockJ = 1 - j + 1;
                else
                    beginingJ = j;
                    beginingBlockJ = 1;
                end
                if (j+blockWidth-1 > vW)
                    endingJ = vW;
                    endingBlockJ = (vW - j) + 1;
                else
                    endingJ = j+blockWidth-1;
                    endingBlockJ = blockWidth;
                end
                pixelCount = (endingI-beginingI+1)*(endingJ-beginingJ+1);
                MSE = sum(sum(abs(blocks(beginingBlockI:endingBlockI,
beginingBlockJ:endingBlockJ, b) - video(beginingI:endingI, beginingJ:endingJ,
t+1)).^2));
                MSE = MSE/pixelCount;
                if (pixelCount > 0 && MSE < minMSE)
                    minMSE = MSE;
                    nextFrame = nextFrameBackup;
                    nextFrame(beginingI:endingI, beginingJ:endingJ) =
blocks(beginingBlockI:endingBlockI, beginingBlockJ:endingBlockJ, b);
                    objectLocation(1, b) = i;
                    objectLocation(2, b) = j;
                end
            end
        end
        encodedVideo(:, :, t) = nextFrame;
    end
end
implay(encodedVideo);
disp('done');
```

(Shown below) original frame from atrium.avi

(Shown below) DFD frame from atrium.avi

The frame from doing DFD on pixel-based motion analysis appears to be blurrier because it only uses pixels from the previous frame to approximate each pixel among their neighboring pixels. Looking closely at the man on the right or the pillars will reveal that some pixels are a little out of place, which makes the man look blurry.

Optical Flow on Block-based motion analysis

```
% This optical flow motion analysis uses sample code found online and
% MATLAB's built-in function
videoReader =
vision.VideoFileReader('atrium.avi','ImageColorSpace','Intensity','VideoOutpu
tDataType','uint8');
converter = vision.ImageDataTypeConverter;
opticalFlow = vision.OpticalFlow('ReferenceFrameDelay', 1);
opticalFlow.OutputValue = 'Horizontal and vertical components in complex
form';
shapeInserter = vision.ShapeInserter('Shape','Lines','BorderColor','Custom',
'CustomBorderColor', 1);
videoPlayer = vision.VideoPlayer('Name','Motion Vector');
while ~isDone(videoReader)
    frame = step(videoReader);
    im = step(converter, frame);
    of = step(opticalFlow, im);
    lines = videooptflowlines(of, 20);
    if ~isempty(lines)
      out =  step(shapeInserter, im, lines);
      step(videoPlayer, out);
    end
end
release(videoPlayer);
release(videoReader);
```

(Shown below) OF frame from atrium.avi

The frame from doing OF on block-based motion analysis shows the direction of where each block is moving. Knowing where each block moves will help approximate its location in the new frame. The video can be encoded by keeping track of the blocks and its motion vector, then moving the block along its motion vector to generate the next frame.

Displaced Frame Difference on Block-based motion analysis

```
clear all;
% select the video to do motion analysis
video = VideoReader('atrium.avi');
% define video properties
vF = video.NumberOfFrames;
vH = video.Height;
vW = video.Width;
video = read(video);
numberOfFrames = 31;            % do motion analysis on last 30 frames
% extract RGB componenet to convert to luminance
videoR = video(:, :, 1, vF-numberOfFrames+1:vF);
videoG = video(:, :, 2, vF-numberOfFrames+1:vF);
videoB = video(:, :, 3, vF-numberOfFrames+1:vF);
video = 0.299*videoR + 0.587*videoG + 0.114*videoB;
video = reshape(video, vH, vW, numberOfFrames);
% DEFINE BLOCK HEIGHT. Set to 1, 1 for pixel-based; vH, vW for global; and
% anything between the two for block-based
blockHeight = 50;
blockWidth = 50;
searchRadius = 3;               % set higher to search more globally
numberOfBlocksAlongY = floor(vH/blockHeight);
numberOfBlocksAlongX = floor(vW/blockWidth);
numberOfBlocks = numberOfBlocksAlongY*numberOfBlocksAlongX;
blocks = uint8(zeros([blockHeight blockWidth numberOfBlocks]));
objectLocation = zeros([2 numberOfBlocks]);
encodedVideo = uint8(zeros([vH vW numberOfFrames-1]));
for t=1:numberOfFrames-1
    t                           % print out the frame that it is working on
    nextFrame = video(:, :, t);
    for i = 1:numberOfBlocksAlongY
        for j = 1:numberOfBlocksAlongX
            blocks(:, :, (i-1)*numberOfBlocksAlongX+mod(j-1,
numberOfBlocksAlongX+1)+1) = video((i-1)*blockHeight+1:i*blockHeight, (j-
1)*blockWidth+1:j*blockWidth, t);
        end
    end
    for b = 1:numberOfBlocks
        newStartingY = floor((b-1)/numberOfBlocksAlongX)*blockHeight+1;
        newStartingX = mod(b-1, numberOfBlocksAlongX)*blockWidth+1;
        minMSE = inf;
        % create a backup in case a better location is found for the block
        nextFrameBackup =  nextFrame;
        for i = newStartingY-searchRadius:newStartingY+searchRadius
            for j = newStartingX-searchRadius:newStartingX+searchRadius
                % search within the specified search region for comparison
                if (i < 1)
                    beginingI = 1;
                    beginingBlockI = 1 - i + 1;
                else
                    beginingI = i;
                    beginingBlockI = 1;
                end
                if (i+blockHeight-1 > vH)
                    endingI = vH;
                    endingBlockI = (vH - i) + 1;
```

```
                else
                    endingI = i+blockHeight-1;
                    endingBlockI = blockHeight;
                end
                if (j < 1)
                    beginingJ = 1;
                    beginingBlockJ = 1 - j + 1;
                else
                    beginingJ = j;
                    beginingBlockJ = 1;
                end
                if (j+blockWidth-1 > vW)
                    endingJ = vW;
                    endingBlockJ = (vW - j) + 1;
                else
                    endingJ = j+blockWidth-1;
                    endingBlockJ = blockWidth;
                end
                pixelCount = (endingI-beginingI+1)*(endingJ-beginingJ+1);
                MSE = sum(sum(abs(blocks(beginingBlockI:endingBlockI,
beginingBlockJ:endingBlockJ, b) - video(beginingI:endingI, beginingJ:endingJ,
t+1)).^2));
                MSE = MSE/pixelCount;
                if (pixelCount > 0 && MSE < minMSE)
                    minMSE = MSE;
                    nextFrame = nextFrameBackup;
                    nextFrame(beginingI:endingI, beginingJ:endingJ) =
blocks(beginingBlockI:endingBlockI, beginingBlockJ:endingBlockJ, b);
                    objectLocation(1, b) = i;
                    objectLocation(2, b) = j;
                end
            end
        end
        encodedVideo(:, :, t) = nextFrame;
    end
end
implay(encodedVideo);
disp('done');
```

(Shown below) Zoomed in block-based DFD frame from atrium.avi, which shows duplicate copies of a man's head (left 1) and misaligned bodies (right 2).

(Shown below) DFD frame from atrium.avi

The frame from doing DFD on block-based motion analysis shows where each block from the previous frame are best matched in the current frame. The DFD frame shows displaced and multiple locations of a man's head because no blocks replaced the current location of the man's head and because the man's head is better placed in the adjacent blocks. There are misaligned bodies when the top of the body is moving forward while the person's leg is in place. As the block size increases, then the objects are less likely to move if there is are only a few objects moving in the frame because having everything else stay where they are in the background is optimal in that scenario.

Optical Flow on Region-based motion analysis

As shown in the comments of the code, OF motion analysis is done on a self-made video of a square that starts on the top left, then moves to the top right, then moves to the bottom right, and finally moves back to the top left (initial position).

```
%First create a moving object. Our video will have a square that starts on
%the top left, then moves to the top right, then moves to the bottom right,
%and finally moves back to the top left (initial position).
clear all;
numberOfLines = 3;
framesPerLine = 16;
objectWidth = 25;
objectHeight = 25;
frameWidth = 100;
frameHeight = 100;
video = ones([frameWidth frameHeight 16*numberOfLines]);
for i = 1:16
    video(1:25, (i-1)*5+1:(i-1)*5+25, i) = 0;
end
```

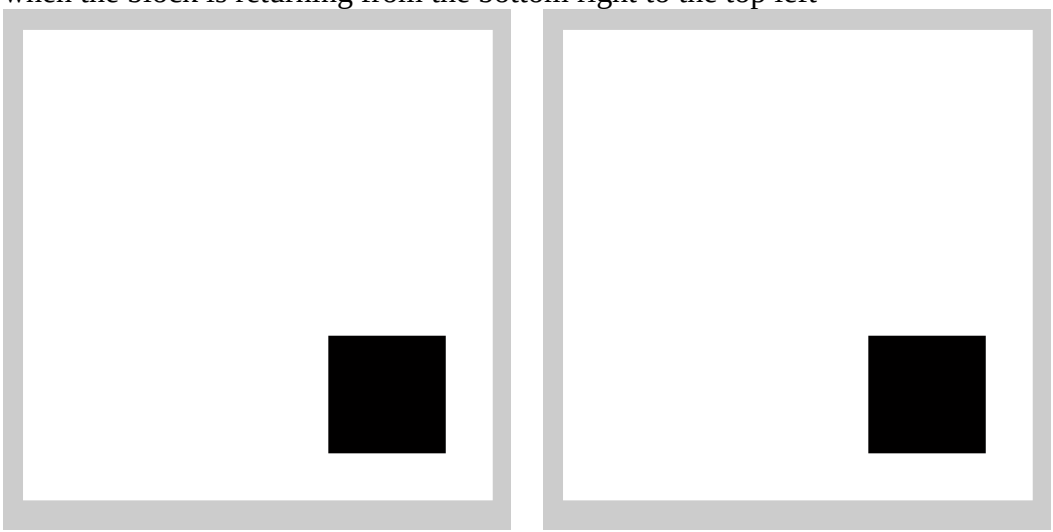
```
for i = 17:32
    video((i-17)*5+1:(i-17)*5+25, 76:100, i) = 0;
end
for i = 33:48
   video(76-(i-33)*5:100-(i-33)*5, 76-(i-33)*5:100-(i-33)*5, i) = 0;
end
for i = 1:numberOfLines*framesPerLine
    imshow(video(:, :, i))
    pause(.03);
end
%The original video contains 100*100*48 doubles. The first 100 is width.
%The second 100 is the height. And 48 is number of frames. Essentially, the
%video grows by H*W per a frame.

%First, attempt the problem using optical flow. This involes computing the
%speed of of the object by subtracting the object's current location from
%the object's previous location. Only the object's speed is stored to
%compute each new frame.
disp('started optical flow region-based motion analysis');
videoSize = size(video);
objectSpeed = zeros([2 videoSize(3)]);
newStartingX = 1;
newStartingY = 1;
for t = 1:videoSize(3)
    MinMSE = frameWidth*frameHeight;
%The next 2 lines of code limits the expected displacement vector to a
%region based on the object on the previous frame
    for i = newStartingX-10:newStartingX+10
        for j = newStartingY-10:newStartingY+10
            if (i < 1 || i+objectWidth-1 > videoSize(1) || j < 1 ||
j+objectHeight-1> videoSize(2))
                continue;
            end
            MSE = sum(sum(abs(video(i:i+objectWidth-1, j:j+objectHeight-1,
t).^2)));
            if (MSE < MinMSE)
                MinMSE = MSE;
                % fprintf('object found on %2d, %2d in time %2d\n', i, j, t);
                objectSpeed(1, t) = i;
                objectSpeed(2, t) = j;
            end
        end
    end
    objectSpeed(1, t) = objectSpeed(1, t) - newStartingX;
    objectSpeed(2, t) = objectSpeed(2, t) - newStartingY;
    newStartingX = objectSpeed(1, t) + newStartingX;
    newStartingY = objectSpeed(2, t) + newStartingY;
end
objectSpeed % is now computed to display the video
frame = ones([frameWidth frameHeight]);
frame(1:objectHeight, 1:objectWidth) = 0;
startingY = 1;
startingX = 1;
imshow(frame);
for t = 1:videoSize(3)
    frame(startingY:startingY+objectHeight-1,
startingX:startingX+objectWidth-1) = 1;
```

```
    startingY = startingY + objectSpeed(1, t);
    startingX = startingX + objectSpeed(2, t);
    frame(startingY:startingY+objectHeight-1,
startingX:startingX+objectWidth-1) = 0;
    imshow(frame);
    pause(.03);
end
disp('done with optical flow region-based motion analysis');
%The encoded video contains 100*100+2*48 doubles. 100*100 doubles for the
%first frame and 2 doubles for every additional frame to store the x and y
%speed. Essentially, the video grows by 2 per an object per a frame.
```

(Show below) Output of region-based OF frame from self-made video, which shows the motion vector. Columns 1 through 15 of the motion vector is filled with [0 5] so that the object moves toward the right. Columns 16 through 30 of the motion vector is filled with [5 0] so that the object moves toward the bottom. Finally, columns 31 through 45 of the motion vector is filled with [-5 -5] so that the object moves back to the original location.

```
started optical flow region-based motion analysis
objectSpeed =
  Columns 1 through 20
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    5    5    5
    0    5    5    5    5    5    5    5    5    5    5    5    5    5    5    5    0    0    0    0
  Columns 21 through 40
    5    5    5    5    5    5    5    5    5    5    5    5    0   -5   -5   -5   -5   -5   -5   -5
    0    0    0    0    0    0    0    0    0    0    0    0    0   -5   -5   -5   -5   -5   -5   -5
  Columns 41 through 48
   -5   -5   -5   -5   -5   -5   -5   -5
   -5   -5   -5   -5   -5   -5   -5   -5
done with optical flow region-based motion analysis
```

(Show below) The original frame (left) and the encoded recovered frame based on OF (right) when the block is returning from the bottom right to the top left



Displaced Frame Difference on Region-based motion analysis

DFD motion analysis is done on the same self-made video of a square that starts on the top left, then moves to the top right, then moves to the bottom right, and finally moves back to the top left (initial position).

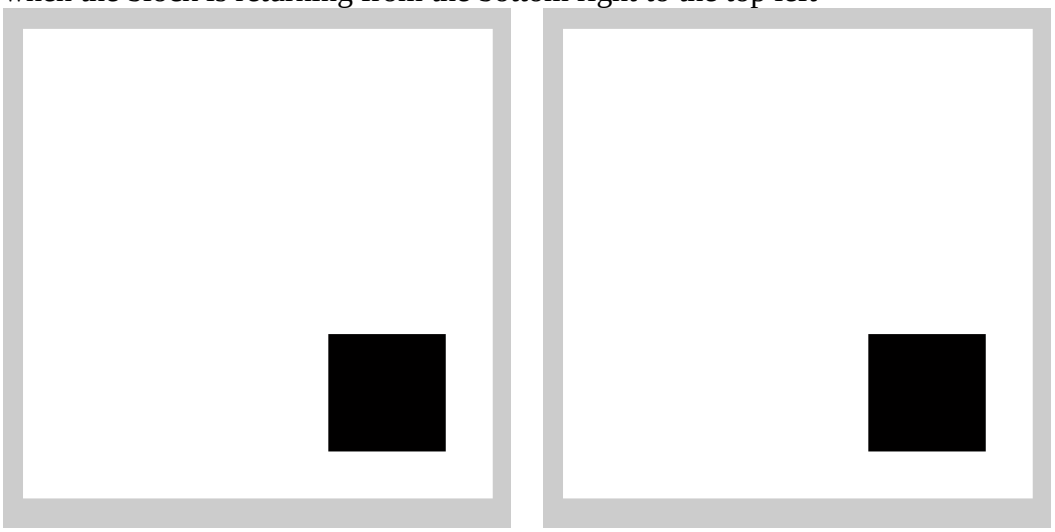
```
%Next, attempt the problem using displaced frame difference. This involes
%attempting to match the object everywhere and attempting to minimize the
%difference the object's difference between its current location and its
%new location. Only the location of the object is stored to compute each
%frame.
disp('started with displaced frame difference region-based motion analysis');
videoSize = size(video);
objectLocation = zeros([2 videoSize(3)]);
objectLocation(1, 1) = 1;
objectLocation(2, 1) = 1;
for t = 2:videoSize(3)
    MinMAD = frameWidth*frameHeight;
    for i = objectLocation(1, t-1)-10:objectLocation(1, t-1)+10
        for j = objectLocation(2, t-1)-10:objectLocation(2, t-1)+10
            if (i < 1 || i > videoSize(1) || j < 1 || j > videoSize(2))
                continue;
            end
            MAD = 0;
            for k = 1:objectWidth
                for l = 1:objectHeight
                    if (i + k > frameWidth || j + l > frameHeight)
                        MAD = MAD + 1;
                    else
                        MAD = MAD + video(i, j, t);
                    end
                end
            end
            if MAD < MinMAD
                MinMAD = MAD;
                objectLocation(1, t) = i;
                objectLocation(2, t) = j;
            end
        end
    end
end
objectLocation % is now computed to display the video
frame = ones([frameWidth frameHeight]);
frame(1:objectHeight, 1:objectWidth) = 0;
startingY = 1;
startingX = 1;
imshow(frame);
for t = 1:videoSize(3)-1
    frame(objectLocation(1,t):objectLocation(1,t)+objectHeight-1,
objectLocation(2,t):objectLocation(2,t)+objectWidth-1) = 1;
    frame(objectLocation(1,t+1):objectLocation(1,t+1)+objectHeight-1,
objectLocation(2,t+1):objectLocation(2,t+1)+objectWidth-1) = 0;
    pause(1);
    imshow(frame);
end
disp('done with displaced frame difference region-based motion analysis');
```

```
%The encoded video contains 100*100+2*48 doubles. 100*100 doubles for the
%first frame and 2 doubles for every additional frame to store the x and y
%speed. Essentially, the video grows by 2 per an object per a frame.
```

(Show below) Output of region-based DFD frame from self-made video, which shows the location vector. Columns 1 through 15 of the location vector has increasing y so that the object moves toward the right. Columns 16 through 30 of the location vector increasing x so that the object moves toward the bottom. Finally, columns 31 through 45 of the location vector has decreasing x and decreasing y so that the object moves back to the original location.

started with displaced frame difference region-based motion analysis
objectLocation =
Columns 1 through 20
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 6 11 16
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 76 76 76 76
Columns 21 through 40
21 26 31 36 41 46 51 56 61 66 71 76 76 71 66 61 56 51 46 41
76 76 76 76 76 76 76 76 76 76 76 76 76 71 66 61 56 51 46 41
Columns 41 through 48
36 31 26 21 16 11 6 1
36 31 26 21 16 11 6 1
done with displaced frame difference region-based motion analysis

(Show below) The original frame (left) and the encoded recovered frame based on DFD (right) when the block is returning from the bottom right to the top left



Optical Flow on Global motion analysis

OF motion analysis is done on shaky_car.avi, a video in the MATLAB library. It shows the recording of a shaky camera from a car's dashboard.

```
% This optical flow motion analysis uses sample code found online and
% MATLAB's built-in function
videoReader =
vision.VideoFileReader('shaky_car.avi','ImageColorSpace','Intensity','VideoOu
tputDataType','uint8');
converter = vision.ImageDataTypeConverter;
opticalFlow = vision.OpticalFlow('ReferenceFrameDelay', 1);
opticalFlow.OutputValue = 'Horizontal and vertical components in complex
form';
shapeInserter = vision.ShapeInserter('Shape','Lines','BorderColor','Custom',
'CustomBorderColor', 1);
videoPlayer = vision.VideoPlayer('Name','Motion Vector');
while ~isDone(videoReader)
    frame = step(videoReader);
    im = step(converter, frame);
    of = step(opticalFlow, im);
    lines = videooptflowlines(of, 20);
    if ~isempty(lines)
      out =  step(shapeInserter, im, lines);
      step(videoPlayer, out);
    end
end
```

(Shown below) original frame from shaky_car.avi

(Shown below) OF frame from shaky_car.avi

The frame from doing OF on global motion analysis shows where each object is moving. OF shows that the movement vector of each object in the frame is pointing upward means that the camera is moving to face downward.

Displaced Frame Difference on Global motion analysis

DFD motion analysis is done on shaky_car.avi, a video in the MATLAB library. It shows the recording of a shaky camera from a car's dashboard.

```
clear all;
% select the video to do motion analysis
video = VideoReader('shaky_car.avi');
% define video properties
vF = video.NumberOfFrames;
vH = video.Height;
vW = video.Width;
video = read(video);
numberOfFrames = 31;            % do motion analysis on last 30 frames
% extract RGB componenet to convert to luminance
videoR = video(:, :, 1, vF-numberOfFrames+1:vF);
videoG = video(:, :, 2, vF-numberOfFrames+1:vF);
videoB = video(:, :, 3, vF-numberOfFrames+1:vF);
video = 0.299*videoR + 0.587*videoG + 0.114*videoB;
video = reshape(video, vH, vW, numberOfFrames);
% DEFINE BLOCK HEIGHT. Set to 1, 1 for pixel-based; vH, vW for global; and
% anything between the two for block-based
blockHeight = vH;
blockWidth = vW;
```

```
searchRadius = 15;              % set higher to search more globally
numberOfBlocksAlongY = floor(vH/blockHeight);
numberOfBlocksAlongX = floor(vW/blockWidth);
numberOfBlocks = numberOfBlocksAlongY*numberOfBlocksAlongX;
blocks = uint8(zeros([blockHeight blockWidth numberOfBlocks]));
objectLocation = zeros([2 numberOfBlocks]);
encodedVideo = uint8(zeros([vH vW numberOfFrames-1]));
for t=1:numberOfFrames-1
    t                           % print out the frame that it is working on
    nextFrame = video(:, :, t);
    for i = 1:numberOfBlocksAlongY
        for j = 1:numberOfBlocksAlongX
            blocks(:, :, (i-1)*numberOfBlocksAlongX+mod(j-1,
numberOfBlocksAlongX+1)+1) = video((i-1)*blockHeight+1:i*blockHeight, (j-
1)*blockWidth+1:j*blockWidth, t);
        end
    end
    for b = 1:numberOfBlocks
        newStartingY = floor((b-1)/numberOfBlocksAlongX)*blockHeight+1;
        newStartingX = mod(b-1, numberOfBlocksAlongX)*blockWidth+1;
        minMSE = inf;
        % create a backup in case a better location is found for the block
        nextFrameBackup =  nextFrame;
        for i = newStartingY-searchRadius:newStartingY+searchRadius
            for j = newStartingX-searchRadius:newStartingX+searchRadius
                % search within the specified search region for comparison
                if (i < 1)
                    beginingI = 1;
                    beginingBlockI = 1 - i + 1;
                else
                    beginingI = i;
                    beginingBlockI = 1;
                end
                if (i+blockHeight-1 > vH)
                    endingI = vH;
                    endingBlockI = (vH - i) + 1;
                else
                    endingI = i+blockHeight-1;
                    endingBlockI = blockHeight;
                end
                if (j < 1)
                    beginingJ = 1;
                    beginingBlockJ = 1 - j + 1;
                else
                    beginingJ = j;
                    beginingBlockJ = 1;
                end
                if (j+blockWidth-1 > vW)
                    endingJ = vW;
                    endingBlockJ = (vW - j) + 1;
                else
                    endingJ = j+blockWidth-1;
                    endingBlockJ = blockWidth;
                end
                pixelCount = (endingI-beginingI+1)*(endingJ-beginingJ+1);
```

```
                MSE = sum(sum(abs(blocks(beginingBlockI:endingBlockI,
beginingBlockJ:endingBlockJ, b) - video(beginingI:endingI, beginingJ:endingJ,
t+1)).^2));
                MSE = MSE/pixelCount;
                if (pixelCount > 0 && MSE < minMSE)
                    minMSE = MSE;
                    nextFrame = nextFrameBackup;
                    nextFrame(beginingI:endingI, beginingJ:endingJ) =
blocks(beginingBlockI:endingBlockI, beginingBlockJ:endingBlockJ, b);
                    objectLocation(1, b) = i;
                    objectLocation(2, b) = j;
                end
            end
        end
        encodedVideo(:, :, t) = nextFrame;
    end
end
implay(encodedVideo);
disp('done');
```

(Shown below) Cropped global DFD frame from shaky_car.avi, which shows 2 copies of the top.

(Shown below) DFD frame from shaky_car.avi

The frame from doing DFD on global motion analysis shows 2 copies of the top because the previous frame is best matched to the current frame when it is moved down.

Summary and Conclusion

Doing pixel-based, block-based, region-based, and global based motion analysis each has their own advantages as highlighted in this report. Pixel-based motion analysis is universally applicable for any video because it does not require locating any object, but requires a lot of computation. Block-based motion analysis uses a intermediate number of computation for a decent accuracy. Region-based motion is good for 1 object or else there will be a high number of computations for segmenting each object. Global motion analysis is good for estimating camera motion.

Using optical flow and displaced frame difference for motion analysis each have their own advantages as highlighted in this report as well. For example, optical flow is good for block-based motion analysis to keep track of the motion of every object. However, optical flow is not good for global motion analysis because it will show that every object is moving. On the other hand, displaced frame difference is excellent for global motion analysis because tracking camera motion is very similar to displacing the frame.