ASSIGNMENT 3_Q4

```
function main()
    % Read the video file
    vidFile = 'CV Vamsi Ass Video.mp4';
    vidReader = VideoReader(vidFile);

    % Parameters for reference frames
    refFrames = [1, 11, 31];

    % Optical flow parameters
    opticFlow = opticalFlowFarneback('NumPyramidLevels',3, 'PyramidScale',0.5,
'NumIterations',15, 'NeighborhoodSize',7, 'FilterSize',5);

    % Define the output video
    outVideo = VideoWriter('Vamsi_Output.mp4', 'MPEG-4');
    outVideo.FrameRate = vidReader.FrameRate;
    open(outVideo);

    % Read and process each frame
    processFrames(vidReader, refFrames, opticFlow, outVideo);

    % Close the video writer
    close(outVideo);
end

function processFrames(vidReader, refFrames, opticFlow, outVideo)
    % Read the first frame
    prevFrame = readFrame(vidReader);
    prevGray = rgb2gray(prevFrame);

    % Process each frame
    while hasFrame(vidReader)
        frame = readFrame(vidReader);
        gray = rgb2gray(frame);

        % Loop over the reference frames
        for i = 1:length(refFrames)
            if refFrames(i) == 1 || mod(vidReader.CurrentTime*vidReader.FrameRate,
refFrames(i)) == 0
                % Calculate optical flow
                flow = estimateFlow(opticFlow, prevGray);

                % Plot optical flow vectors
                frameWithFlow = plotOpticalFlow(frame, flow);

                % Write frame with optical flow to video
```

```
                writeVideo(outVideo, frameWithFlow);
            end
        end

        % Update previous frame
        prevGray = gray;
    end
end

function frameWithFlow = plotOpticalFlow(frame, flow)
    % Plot optical flow vectors
    imshow(frame);
    hold on;
    plot(flow, 'DecimationFactor', [10 10], 'ScaleFactor', 2);
    hold off;

    % Convert figure to frame
    drawnow;
    frameWithFlow = getframe;

    % Resize frame to match original frame size
    frameWithFlow = imresize(frameWithFlow.cdata, [size(frame, 1), size(frame, 2)]);
end
```

ASSIGNMENT 3_Q6

```
% Load image sets
setDir = fullfile(toolboxdir('vision'), 'visiondata', 'imageSets');
imgSets = imageSet(setDir, 'recursive');

% Partition the dataset
trainingSets = partition(imgSets, 0.8, 'randomize');

% Create the bag of features
bag = bagOfFeatures(trainingSets, 'Verbose', false);

% Extract features and labels from the training set
numImages = sum([trainingSets.Count]);
features = zeros(numImages, bag.VocabularySize);
labels = cell(numImages, 1);
counter = 1;
for i = 1:numel(trainingSets)
    for j = 1:trainingSets(i).Count
        img = read(trainingSets(i), j);
        features(counter, :) = encode(bag, img);
```

```
        labels{counter} = trainingSets(i).Description;
        counter = counter + 1;
    end
end

% Train a classifier (e.g., SVM)
classifier = fitcecoc(features, labels);

% Initialize variables for testing
testingFeatures = [];
testingLabels = {};

% Extract features and labels from the testing set (first two images from each image
set)
for i = 1:numel(imgSets)
    for j = 1:2 % Use only the first two images for testing
        img = read(imgSets(i), j);
        testingFeatures = [testingFeatures; encode(bag, img)];
        testingLabels = [testingLabels; imgSets(i).Description];
    end
end

% Predict labels for testing features
predictedLabels = predict(classifier, testingFeatures);

% Calculate accuracy
accuracy = sum(strcmp(predictedLabels, testingLabels)) / numel(testingLabels);
disp(['Accuracy: ', num2str(accuracy)]);
```

ASSIGNMENT 3_Q7

```
%% Step 1: Read Stereo Image Pair
I1 = imread("v-mark-1.jpeg");
I2 = imread("v-mark-2.jpeg");

% Convert to grayscale.
I1gray = im2gray(I1);
I2gray = im2gray(I2);

figure
imshowpair(I1,I2,"montage")
title("I1 (left); I2 (right)")

figure
imshow(stereoAnaglyph(I1,I2))
```

```
title("Composite Image (Red - Left Image, Cyan - Right Image)")

%%Step 2: Collect Interest Points from Each Image
blobs1 = detectSURFFeatures(I1gray,MetricThreshold= 2000);
blobs2 = detectSURFFeatures(I2gray,MetricThreshold=2000);

figure
imshow(I1)
hold on
plot(selectStrongest(blobs1,30))
title("Thirty Strongest SURF Features In I1")

figure
imshow(I2)
hold on
plot(selectStrongest(blobs2,30))
title("Thirty Strongest SURF Features In I2")

%% Step 3: Find Putative Point Correspondences
[features1,validBlobs1] = extractFeatures(I1gray,blobs1);
[features2,validBlobs2] = extractFeatures(I2gray,blobs2);

indexPairs = matchFeatures(features1,features2,Metric="SAD", ...
  MatchThreshold=5);

matchedPoints1 = validBlobs1(indexPairs(:,1),:);
matchedPoints2 = validBlobs2(indexPairs(:,2),:);

figure
showMatchedFeatures(I1, I2, matchedPoints1, matchedPoints2)
legend("Putatively Matched Points In I1","Putatively Matched Points In I2")

%% Step 4: Remove Outliers Using Epipolar Constraint
[fMatrix, epipolarInliers, status] = estimateFundamentalMatrix(...
  matchedPoints1,matchedPoints2,Method="RANSAC", ...
  NumTrials=10000,DistanceThreshold=0.1,Confidence=99.99);

if status ~= 0 || isEpipoleInImage(fMatrix,size(I1)) ...
  || isEpipoleInImage(fMatrix',size(I2))
  error(["Not enough matching points were found or the epipoles are inside the images.
Inspect and improve the quality of detected features and images."]);
end

inlierPoints1 = matchedPoints1(epipolarInliers, :);
inlierPoints2 = matchedPoints2(epipolarInliers, :);

figure
showMatchedFeatures(I1, I2, inlierPoints1, inlierPoints2)
```

```
legend("Inlier Points In I1","Inlier Points In I2")

%% Step 5: Rectify Images

[tform1, tform2] = estimateStereoRectification(fMatrix, ...
  inlierPoints1.Location,inlierPoints2.Location,size(I2));

[I1Rect, I2Rect] = rectifyStereoImages(I1,I2,tform1,tform2);
figure
imshow(stereoAnaglyph(I1Rect,I2Rect))
title("Rectified Stereo Images (Red - Left Image, Cyan - Right Image)")
```