# Assignment 3

Question 4: For the video (problem 1) you have taken, plot the optical flow vectors on each frame using MATLAB's optical flow codes. (i) treating every previous frame as a reference frame (ii) treating every 11th frame as a reference frame (iii) treating every 31st frame as a reference frame

```
% Read the video file
videoFile = 'cv-10s_video_ass2.mp4';
videoReader = VideoReader(videoFile);

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

% Optical flow parameters
opticFlow = opticalFlowFarneback('NumPyramidLevels',3, 'PyramidScale',0.5, 'NumIterations',15, 'NeighborhoodSize',7, 'Filt

% Define the output video
outputVideo = VideoWriter('matlab_q4_optical_flow_output.mp4', 'MPEG-4');
outputVideo.FrameRate = videoReader.FrameRate;
open(outputVideo);

% Read the first frame
prevFrame = readFrame(videoReader);
prevGray = rgb2gray(prevFrame);

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

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

            % 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)]);

            % Write frame with optical flow to video
            writeVideo(outputVideo, frameWithFlow);
        end
    end

    % Update previous frame
    prevGray = gray;
end

% Close the video writer
close(outputVideo);
```

Question 6: Refer to the Bag of Features example MATLAB source code provided in the classroom's classwork page. In your homework, pick an object category that would be commonly seen in any household (e.g. cutlery) and pick 5 object types (e.g. for cutlery pick spoon, fork, butter knife, cutting knife, ladle). Present your performance evaluation.

```
% 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)]);
```

Question 7: Repeat the image capture experiment from problem (3), however, now also rotate (along the ground plane) the camera 2 (right camera) towards camera 1 position, after translation by T. Make sure the marker is within view. Note down the rotation angle. Run the tutorial provided for uncalibrated stereo rectification in here: https://www.mathworks.com/help/vision/ug/uncalibrated-stereo-image-rectification.html (MATLAB is mandatory for this exercise). Exercise this tutorial for the image pairs you have captured. You can make assumptions as necessary, however, justify them in your answers/description. *(Note: you can print out protractors from any online source and place your cameras on that when running experiments: http://www.ossmann.com/protractor/conventional-protractor.pdf).*

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