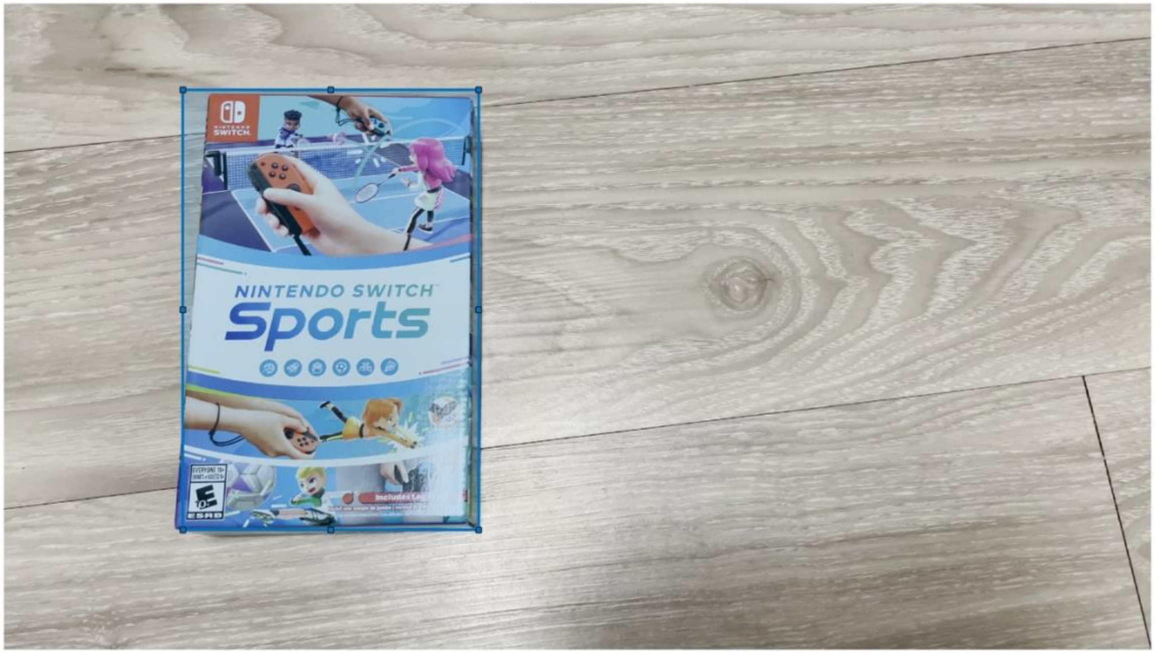
PART A: Theory

1.

From a 10 second video footage taken as required in the question, we generate a template image frame (T_M) with size (m, n) shown as:

From this generated template image, we can crop out a template patch P as our region of interest corresponding to the object in this template image. The size of such template patch P is (a, b) where $a < m, b < n$.

Then, we select another image frame (I) from the footage within the same size of (m, n) and compute a normalized correlation matrix M within the size of $(m - a + 1, n - b + 1)$. Such correlation matrix is calculated by applying a sliding window technique and comparing the patch P with the same size patches from I. In this way, we can locate the maximum normalized correlation value as M_{max} and find the corresponding patch in I.



PART B: MATLAB Prototyping
5.

```
 % If we treat every previous frame as a reference frame
 % read video file
 vidReader = VideoReader('./IMG_5905.MOV','CurrentTime',1);
 % init opticFlow object
 opticFlow = opticalFlowLK('NoiseThreshold',0.009);

 % init plot object
 h = figure;
 movegui(h);
 hViewPanel = uipanel(h,'Position',[0 0 1 1],'Title','Plot of Optical Flow
Vectors');
 hPlot = axes(hViewPanel);

 % process frames
 while hasFrame(vidReader)
     % read a frame
     frameRGB = readFrame(vidReader);
     frameGray = rgb2gray(frameRGB); % in some versions, using im2grpy(frameRGB)

     % estimate optical flow
     flow = estimateFlow(opticFlow,frameGray);
```

```
    % show optical flow
    imshow(frameRGB)
    hold on
    plot(flow,'DecimationFactor',[5 5],'ScaleFactor',10,'Parent',hPlot);
    hold off
    pause(10^-3)
end
```

```
% If we treat every 11th frame as a reference frame
% read video file
vidReader = VideoReader('./IMG_5905.MOV','CurrentTime',1);
% init opticFlow object
opticFlow = opticalFlowLK('NoiseThreshold',0.009);

h = figure;
movegui(h);
hViewPanel = uipanel(h,'Position',[0 0 1 1],'Title','Plot of Optical Flow
Vectors');
hPlot = axes(hViewPanel);

freq=11;
for v = 1:freq:vidReader.NumFrames
    frameRGB = read(vidReader, v);
    frameGray = rgb2gray(frameRGB); % in some versions, using im2grpy(frameRGB)

    flow = estimateFlow(opticFlow,frameGray);

    imshow(frameRGB)
    hold on
    plot(flow,'DecimationFactor',[5 5],'ScaleFactor',10,'Parent',hPlot);
    hold off
    pause(10^-3)
end
```

```
% If we treat every 31st frame as a reference frame
% read video file
vidReader = VideoReader('./IMG_5905.MOV','CurrentTime',1);
% init opticFlow object
opticFlow = opticalFlowLK('NoiseThreshold',0.009);

h = figure;
```

```
 movegui(h);
 hViewPanel = uipanel(h,'Position',[0 0 1 1],'Title','Plot of Optical Flow
Vectors');
 hPlot = axes(hViewPanel);

 freq=31;
 for v = 1:freq:vidReader.NumFrames
     frameRGB = read(vidReader, v);
     frameGray = rgb2gray(frameRGB); % in some versions, using im2grpy(frameRGB)

     flow = estimateFlow(opticFlow,frameGray);

     imshow(frameRGB)
     hold on
     plot(flow,'DecimationFactor',[5 5],'ScaleFactor',10,'Parent',hPlot);
     hold off
     pause(10^-3)
 end
```

6.

```
 % 6. Run the feature-based matching object detection on the images from problem
(1).
 %% read images
 % Read the target image containing a cluttered scene.
 sceneImage = imread('./example_scene.jpg'); % './q6_image/example_scene.jpg'
 sceneImage = rgb2gray(sceneImage); % in some versions, using im2grpy(frameRGB)
 % figure;
 % imshow(sceneImage);
 % title('Image of a Cluttered Scene');
 % Read the reference image containing the object of interest.
 boxImage = imread('./example_box.jpg'); % './q6_image/example_box.jpg'
 boxImage = rgb2gray(boxImage);
 % figure;
 % imshow(boxImage);
 % title('Image of a Box');
 %% Step-2: Detect feature points in both images.
 boxPoints = detectSURFFeatures(boxImage);
 scenePoints = detectSURFFeatures(sceneImage);

 % Visualize the strongest feature points found in the target image.
 figure;
 imshow(boxImage);
 title('100 Strongest Feature Points from Box Image');
 hold on;
```

```
plot(selectStrongest(boxPoints, 100));

% Visualize the strongest feature points found in the target image.
figure;
imshow(sceneImage);
title('300 Strongest Feature Points from Scene Image');
hold on;
plot(selectStrongest(scenePoints, 300));
```

```
%% Step-3: Extract feature descriptors at the interest points in both images.
[boxFeatures, boxPoints] = extractFeatures(boxImage, boxPoints);
[sceneFeatures, scenePoints] = extractFeatures(sceneImage, scenePoints);
%% Step-4: Match the features using their descriptors.
boxPairs = matchFeatures(boxFeatures, sceneFeatures);

% Display putatively matched features.
matchedBoxPoints = boxPoints(boxPairs(:, 1), :);
matchedScenePoints = scenePoints(boxPairs(:, 2), :);
figure;
showMatchedFeatures(boxImage, sceneImage, matchedBoxPoints, ...
    matchedScenePoints, 'montage');
title('Putatively Matched Points (Including Outliers)');
```

```
%% Step 5: Locate the Object in the Scene Using Putative Matches
[tform, inlierBoxPoints, inlierScenePoints] =
estimateGeometricTransform(matchedBoxPoints, matchedScenePoints, 'affine');

figure;
showMatchedFeatures(boxImage, sceneImage, inlierBoxPoints, ...
    inlierScenePoints, 'montage');
title('Matched Points (Inliers Only)');
```

```

% Get the bounding polygon of the reference image.
boxPolygon = [1, 1;...                           % top-left
        size(boxImage, 2), 1;...                 % top-right
        size(boxImage, 2), size(boxImage, 1);... % bottom-right
        1, size(boxImage, 1);...                 % bottom-left
        1, 1];                   % top-left again to close the polygon
```

```
 % Transform the polygon into the coordinate system of the target image.
 % The transformed polygon indicates the location of the object in the scene.
 newBoxPolygon = transformPointsForward(tform, boxPolygon);
 % Display the detected object.
 figure;
 imshow(sceneImage);
 hold on;
 line(newBoxPolygon(:, 1), newBoxPolygon(:, 2), 'Color', 'y');
 title('Detected Box');
```

7.

```
% 7. 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 Dataset
 imds=imageDatastore('./office_images','IncludeSubfolders',true,'LabelSource','foldernames');

 % inspect the number of images per category, as well as category labels
 tbl=countEachLabel(imds)
```

tbl = 5×2 table

	Label	Count
1	headphones	10
2	monitor	10
3	pen	10
4	scissors	10
5	stapler	10

```
 % visualization
 figure
 montage(imds.Files(1:16:end))
 %% Prepare Training and Validation Image Sets
 [trainingSet, validationSet] = splitEachLabel(imds, 0.6, 'randomize');
 %% Create a Visual Vocabulary and Train an Image Category Classifier
```

```matlab
% Creating Bag-Of-Features.
bag = bagOfFeatures(trainingSet);
```

```
Creating Bag-Of-Features.
-------------------------
* Image category 1: headphones
* Image category 2: monitor
* Image category 3: pen
* Image category 4: scissors
* Image category 5: stapler
* Selecting feature point locations using the Grid method.
* Extracting SURF features from the selected feature point locations.
** The GridStep is [8 8] and the BlockWidth is [32 64 96 128].

* Extracting features from 30 images...done. Extracted 1875000 features.

* Keeping 80 percent of the strongest features from each category.

* Creating a 500 word visual vocabulary.
* Number of levels: 1
* Branching factor: 500
* Number of clustering steps: 1

* [Step 1/1] Clustering vocabulary level 1.
* Number of features          : 1500000
* Number of clusters          : 500
* Initializing cluster centers...100.00%.
* Clustering...completed 14/100 iterations (~3.14 seconds/iteration)...converged in 14 iterations.

* Finished creating Bag-Of-Features
```

```matlab
% Encoding images using Bag-Of-Features.
img = readimage(imds, 1);
featureVector = encode(bag, img);
```

```
Encoding images using Bag-Of-Features.
--------------------------------------
* Encoding an image...done.
```

```matlab

% Plot the histogram of visual word occurrences
figure
bar(featureVector)
title('Visual word occurrences')
xlabel('Visual word index')
ylabel('Frequency of occurrence')
%% Training an image category classifier for 5 categories.
categoryClassifier = trainImageCategoryClassifier(trainingSet, bag);
```

```
Training an image category classifier for 5 categories.

--------------------------------------------------------

* Category 1: headphones

* Category 2: monitor

* Category 3: pen

* Category 4: scissors

* Category 5: stapler


* Encoding features for 30 images...done.


* Finished training the category classifier. Use evaluate to test the classifier on a test set.
```

```matlab
%% Evaluate Classifier Performance

% on training set
confMatrix = evaluate(categoryClassifier, trainingSet);
```

```
Evaluating image category classifier for 5 categories.

-------------------------------------------------------


* Category 1: headphones

* Category 2: monitor

* Category 3: pen

* Category 4: scissors

* Category 5: stapler


* Evaluating 30 images...done.


* Finished evaluating all the test sets.
* The confusion matrix for this test set is:


                              PREDICTED
KNOWN         | headphones   monitor   pen    scissors   stapler
--------------------------------------------------------------------
headphones    | 1.00         0.00      0.00   0.00       0.00
monitor       | 0.00         1.00      0.00   0.00       0.00
pen           | 0.00         0.00      1.00   0.00       0.00
scissors      | 0.00         0.00      0.33   0.67       0.00
stapler       | 0.00         0.00      0.17   0.00       0.83

* Average Accuracy is 0.90.
```

```
% on validation set
confMatrix_val = evaluate(categoryClassifier, validationSet);
```

```
Evaluating image category classifier for 5 categories.
-------------------------------------------------------


* Category 1: headphones
* Category 2: monitor
* Category 3: pen
* Category 4: scissors
* Category 5: stapler

* Evaluating 20 images...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:


                              PREDICTED
KNOWN         | headphones   monitor   pen    scissors   stapler
--------------------------------------------------------------------
headphones    | 0.75         0.25      0.00   0.00       0.00
```

```
monitor     | 0.00      1.00     0.00   0.00      0.00
pen         | 0.00      0.00     1.00   0.00      0.00
scissors    | 0.00      0.00     0.75   0.25      0.00
stapler     | 0.00      0.50     0.00   0.00      0.50

* Average Accuracy is 0.70.
```

```
% Compute average accuracy
avg_acc = mean(diag(confMatrix_val));
%% show example
% read an image
img = imread(fullfile('./office_images','headphones','frame_0001.jpg'));
figure
imshow(img)

% run classification
[labelIdx, scores] = predict(categoryClassifier, img);
```

```
Encoding images using Bag-Of-Features.
--------------------------------------
* Encoding an image...done.
```

```
labelName = categoryClassifier.Labels(labelIdx);
disp(labelName)
```

```
    {'headphones'}
```

8.

Step 1: Read Stereo Image Pair

```
I1 = imread('../HW1/images/color_marker_l.jpg');
I2 = imread('../HW1/images/color_marker_r.jpg');

% 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(['Either not enough matching points were found or '...
         'the epipoles are inside the images. You may need to '...
         'inspect and improve the quality of detected features ',...
         'and/or improve the quality of your 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

```
[t1, t2] = estimateUncalibratedRectification(fMatrix, ...
  inlierPoints1.Location, inlierPoints2.Location, size(I2));
tform1 = projective2d(t1);
tform2 = projective2d(t2);

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

PART C: Application development

Link to the github repository:
https://github.com/annieee6446/CSC-8830-Computer-Vision-HW3