Experiment-3.2

Aim of the Experiment : Object tracking using template matching and feature matching.

a) Object tracking using template matching.

b) Object tracking using feature matching.

1) OBJECT TRACKING USING TEMPLATE MATCHING:

Problem Description :

a) Template matching is a simple yet effective method for object tracking. It involves comparing a template image (a small patch of the object to be tracked) with different regions of the current frame.

b) The template is moved across the image, and at each position, a similarity measure is computed between the template and the image patch under it. Common similarity measures include correlation coefficient, sum of squared differences, and normalized cross-correlation.

c) The position where the similarity measure is the highest indicates the location of the object in the current frame.

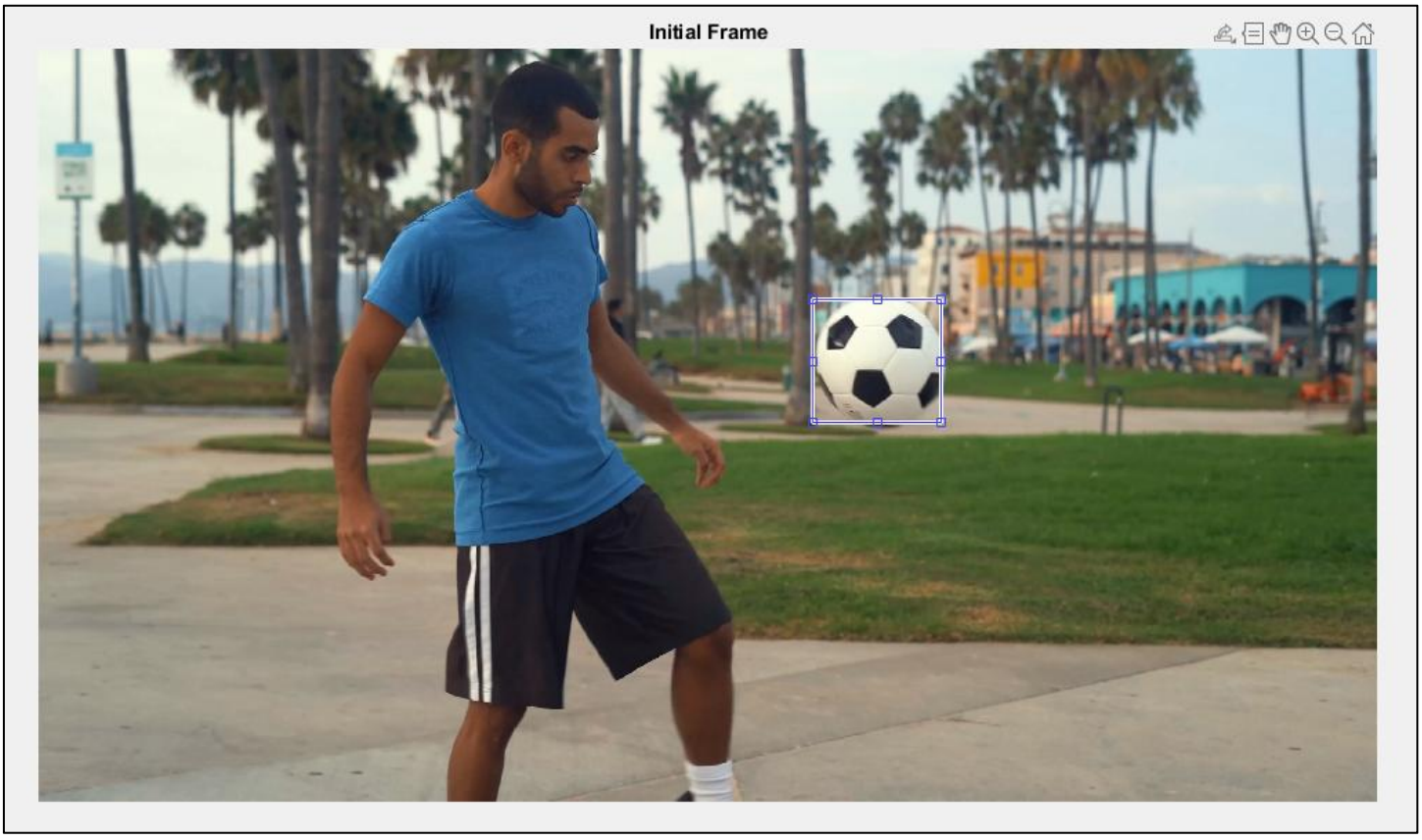
d) Template matching works well when the object being tracked maintains consistent appearance, scale, and orientation throughout the video sequence. However, it can be sensitive to changes in lighting, occlusion, and deformation.

Code for Experiment :

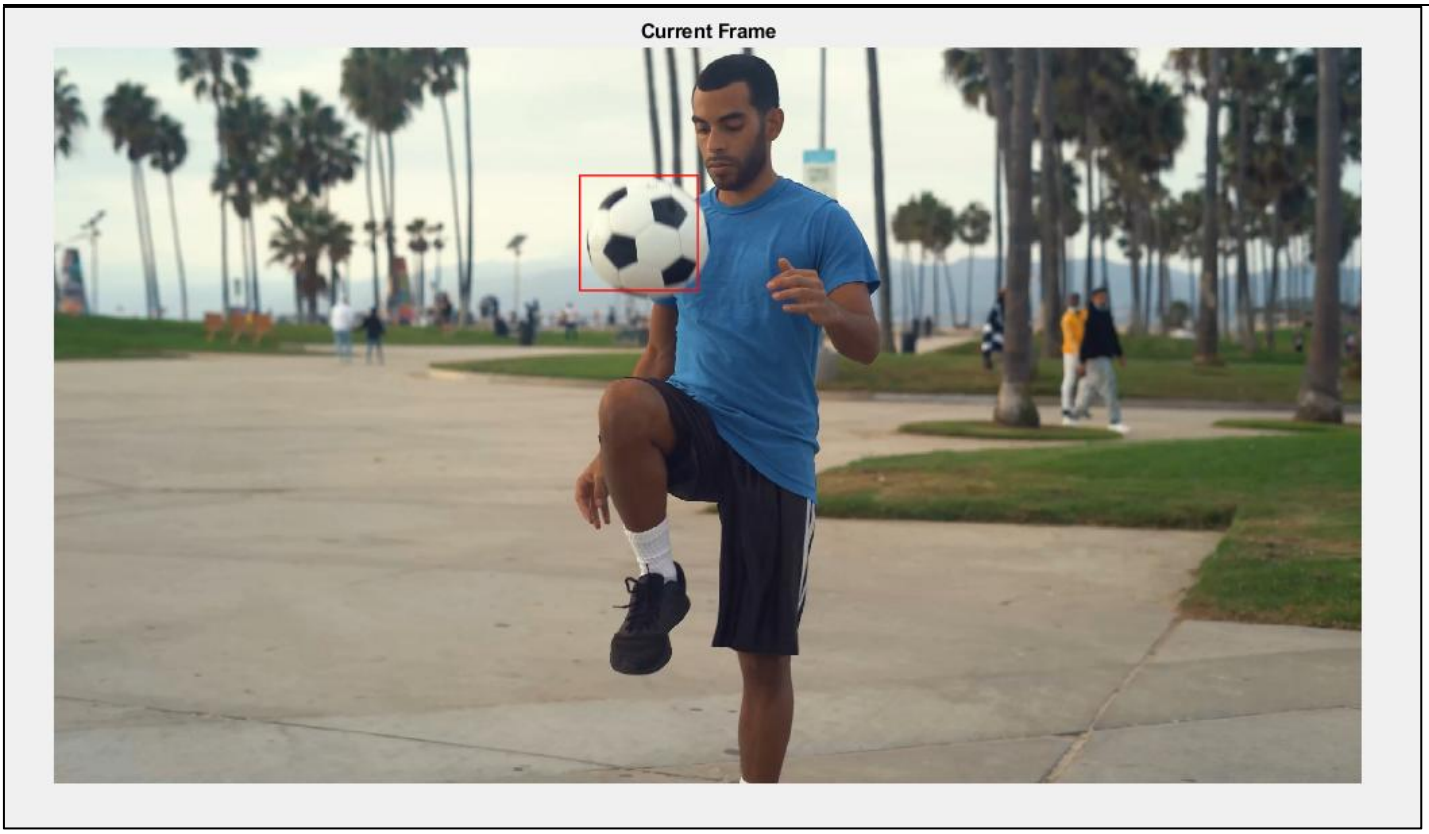
```
% Read the initial frame and select the object to track
initialFrame = imread('image.jpg');

% Display the image
imshow(initialFrame);

% Allow the user to draw a rectangle around the ROI
objectRegion = round(getPosition(imrect));

% Extract the template from the initial frame
template = imcrop(initialFrame, objectRegion);
template=rgb2gray(template);

% Create a video reader object
videoFile = 'football.mp4';
videoObject = VideoReader(videoFile);
```

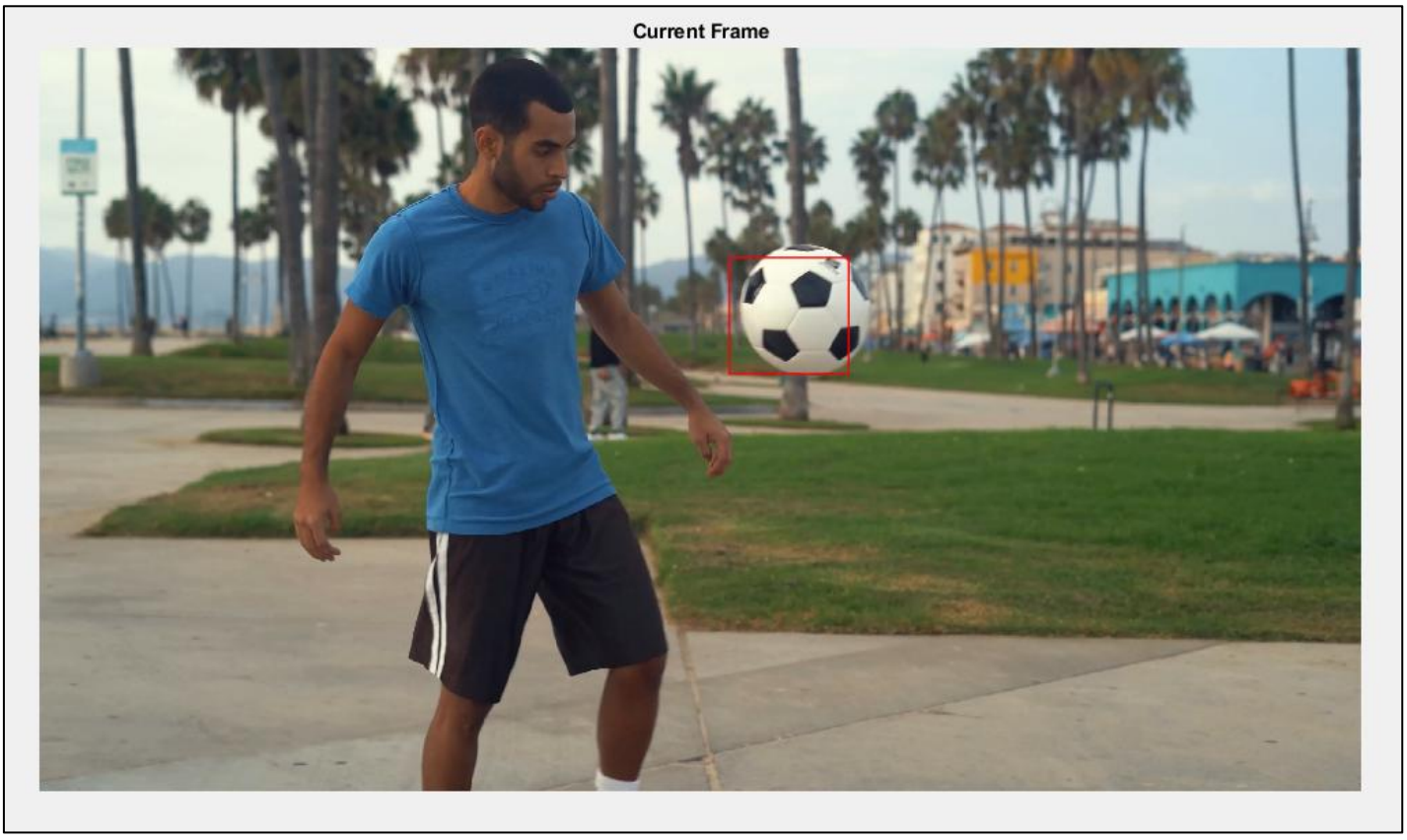
```
while hasFrame(videoObject)
        % Read the current frame
        currentFrame = readFrame(videoObject);
        % Perform template matching
        correlationMap = normxcorr2(template, rgb2gray(currentFrame));

        % Find the location of the maximum correlation
        [~, maxIndex] = max(correlationMap(:));
        [ypeak, xpeak] = ind2sub(size(correlationMap), maxIndex(1));
        % Update the object location
        objectLocation = [xpeak - size(template, 2) + 1, ypeak - size(template, 1) + 1];

        % Display the tracking result
        imshow(currentFrame);
        hold on;
        rectangle('Position', [objectLocation, objectRegion(3:4)], 'EdgeColor', 'red',
        'LineWidth', 2);
        hold off;
        drawnow;
end
```

Result/Output :



Course Name: Master of Engineering - AIML Course Code: AI-301





2) OBJECT TRACKING USING FEATURE MATCHING:

Problem Description :

a) Feature matching involves detecting distinctive features in both the template and the current frame and then matching these features to establish correspondence between the two.

b) Features can include keypoints such as corners, edges, or blobs, which are robust to changes in illumination, scale, and viewpoint.

c) Common algorithms for feature detection and matching include SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features), and ORB (Oriented FAST and Rotated BRIEF).

d) Once features are detected and described in both the template and the current frame, a matching algorithm (e.g., nearest neighbor or RANSAC) is used to find corresponding features.

e) The object's position in the current frame can be estimated by analyzing the spatial distribution of matched features.

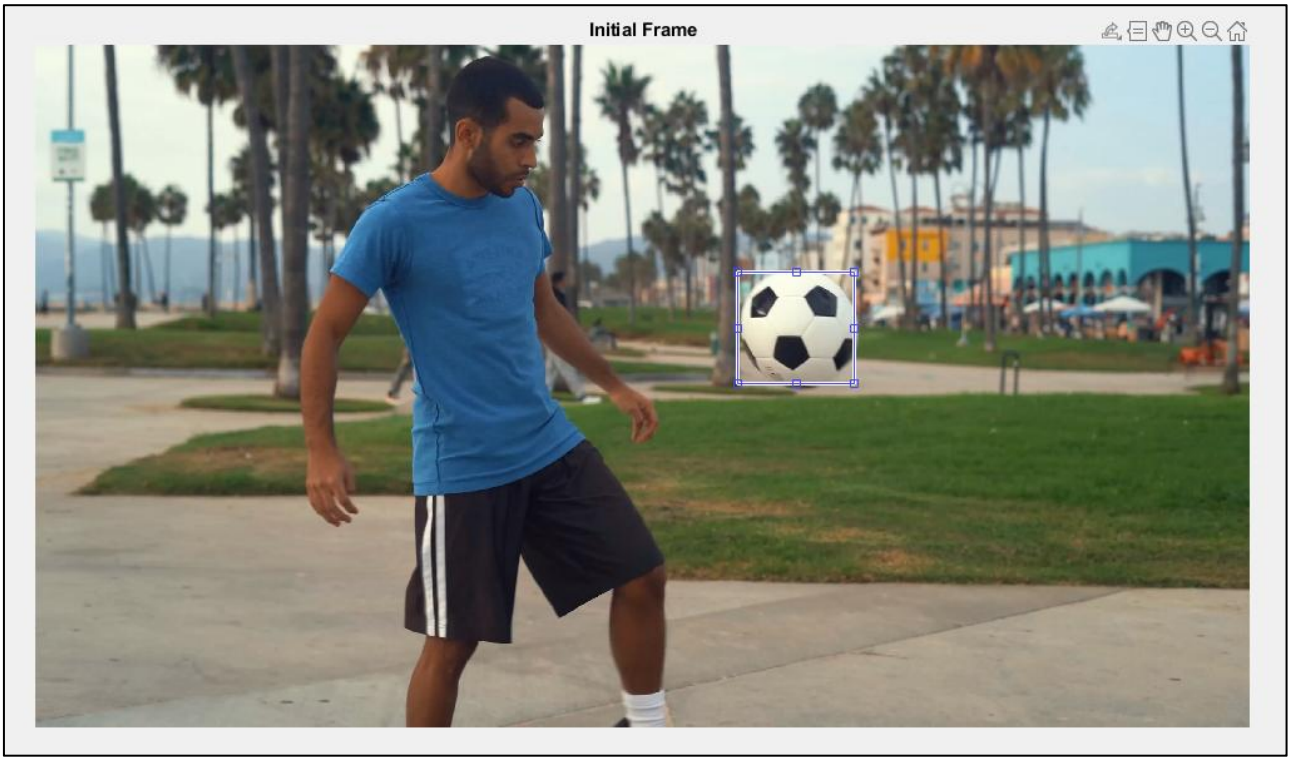
f) Feature matching is more robust than template matching in handling variations in object appearance and environmental conditions. It can handle changes in scale, rotation, and partial occlusion more effectively.

Code for Experiment :

```
% Read the initial frame and select the object to track
initialFrame = imread('image.jpg');

% Display the image
imshow(initialFrame);
title("Initial Frame");

% Allow the user to draw a rectangle around the ROI
objectRegion = round(getPosition(imrect));

% Extract the template from the initial frame
template = imcrop(initialFrame, objectRegion);
templateGray = rgb2gray(template);

% Create a video reader object
videoFile = 'football.mp4';
videoObject = VideoReader(videoFile);

% Object Tracking using Feature Matching (SURF)
% Reset the video reader object
videoObject.CurrentTime = 0;
```

```
% Detect and extract features from the template
pointsTemplate = detectSURFFeatures(templateGray);
featuresTemplate = extractFeatures(templateGray, pointsTemplate);

while hasFrame(videoObject)
    % Read the current frame
    currentFrame = readFrame(videoObject);

    % Detect and extract features from the current frame
    grayCurrentFrame = rgb2gray(currentFrame);
    pointsCurrentFrame = detectSURFFeatures(grayCurrentFrame);
    featuresCurrentFrame = extractFeatures(grayCurrentFrame, pointsCurrentFrame);

    % Match features between the template and the current frame
    indexPairs = matchFeatures(featuresTemplate, featuresCurrentFrame);
    % Display the matched features and tracking result
    matchedPointsTemplate = pointsTemplate(indexPairs(:, 1), :);
    matchedPointsCurrentFrame = pointsCurrentFrame(indexPairs(:, 2), :);

    % Display the template frame
    hold on;
    showMatchedFeatures(templateGray, grayCurrentFrame, matchedPointsTemplate,
    matchedPointsCurrentFrame, 'montage', 'Parent', gca);
    hold off;
    drawnow;
end
```

Result/Output :



Learning outcomes :

1. Learnt about the concept of Object detection.
2. Learnt about the working of template matching.
3. Learnt about the working of feature matching.
4. Learnt about various similiarity measures used in template matching.
5. Learnt about use of SIFT and SURF in feature matching.