Project 3: Car Tracking

ENPM673 Perception for Autonomous Robots - Spring 2017

University Of Maryland

Project Report *by*

Neel Parikh
Ravi Bhadeshiya
Kavit Patel

Contents

List of Figures

Listings

Chapter 1

Brief Pipeline

1.1 Car detection

To detect the cars in a given frame, Cascade classifiers are used in this project. For training Cascade classifiers, HOG features are being used. For the training purpose, the provided dataset was augmented by flipping and rotating positive as well as negative images. Approximately, 150,000 images were used as negative samples and 60,000 were used as positive samples.

```
clear all; close all; clc;

cd ../input;
videoFReader = vision.VideoFileReader('simple.avi');
carDetector=vision.CascadeObjectDetector('cars.xml');
cd ../code;
videoFWriter = vision.VideoFileWriter('../Output/output.mp4',
    'FileFormat','MPEG4');

carDetector.MergeThreshold=1;
carDetector.MinSize=[50 50];
videoFrame=videoFReader();
videoFrame(1:240,:,:)=0;
boundingBox=step(carDetector, videoFrame);
carAnotate=insertObjectAnnotation(videoFrame, 'rectangle',
    boundingBox, 'Car');
imshow(carAnotate);
```

Listing 1.1: Starter code

1.1.1 Masking and feature detection

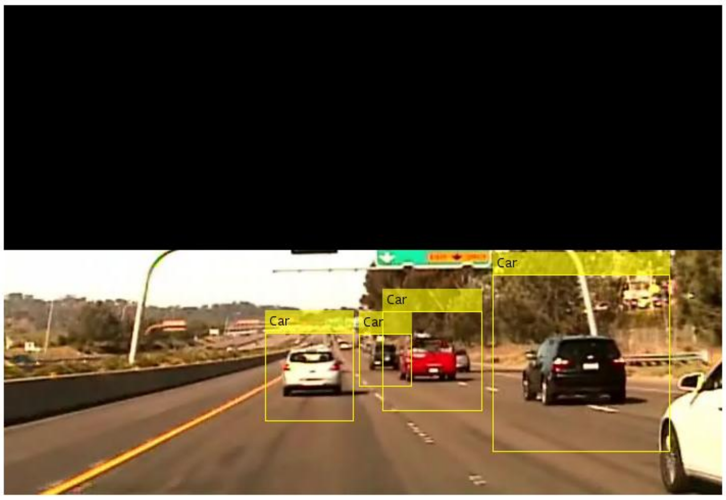

Figure 1.1: Car detection and filtering

To improve the detection in a given frame, the first frame in which car detection takes place that image is truncated as shown in figure 1.2. This step is justified since above horizon there won't be any object of interest to detect. To further improve, the minimum size of the object to be detected is also set which reduces amount of false positives. Apart from that, *Merge-threshold* property of classifier was also set to *1*. This property groups the multiple detections in multi-scale space into one object if it is within the threshold value.

```
%%
videoPlayer  = vision.VideoPlayer('Position',...
    [100 100 [size(videoFrame, 2), size(videoFrame, 1)]+30]);
%%
for i=1:size(boundingBox,1)
    pointTracker{i} = vision.PointTracker('MaxBidirectionalError',2);

    points = detectSURFFeatures(rgb2gray(videoFrame),'ROI',boundingBox(i,:), 'MetricThreshold', 300, 'NumOctaves', 20);

    initialize(pointTracker{i},points.Location,videoFrame);
    bboxPoints{i} = bbox2points(boundingBox(i,:));

    pointsStruct{i} = points.Location;
    oldPoints{i} = pointsStruct{i};
end
```

Listing 1.2: Feature Extraction and Tracker initialization

1.2 Car Tracking

After detecting the object of interest in the previous step, the next step is to extract the features in the region of interest(ROI) obtained. In this project, Speeded-Up Robust Features are used to identify the keypoints in ROI. These features are then tracked in the subsequent frames using Kanade-Lucas-Tomasi(KLT) tracking algorithm.

1.2.1 Point Tracking

Before tracking the points of interest in the frames, the main step is to identify them. For this as mentioned above, SURF features are used which are scale invariant like SIFT but faster than it and more robust than it. These extracted features, are then used as points of interest in KLT feature tracker. KLT tracker tracks those points in all the subsequent frames without needing to detect the object again.

```
%%
while ~isDone(videoFReader)
    % get the next frame
    videoFrame = step(videoFReader);
    visiblePointsList=[];
    oldInliers=[];
    % Track the points. Note that some points may be lost.
    for j=1:length(pointTracker)

        [points, isFound] = step(pointTracker{j}, videoFrame)
   ;
        visiblePoints{j} = points(isFound, :);
        visiblePointsList = [visiblePointsList; visiblePoints
   {j}];
        oldInliers = [oldInliers; oldPoints{j}(isFound, :)];

    end
    % Estimate the geometric transformation between the old
   points
```

```
    % and the new points and eliminate outliers
     [xform, oldInliers, visiblePointsList] = estimateGeometricTransform(...
         oldInliers, visiblePointsList, 'similarity', 'MaxDistance', 1);
     for j=1:length(pointTracker)
         % Apply the transformation to the bounding box points
         bboxPoints{j} = transformPointsForward(xform, bboxPoints{j});

         % Insert a bounding box around the object being tracked
         bboxPolygon = reshape(bboxPoints{j}', 1, []);
         videoFrame = insertShape(videoFrame, 'Polygon', bboxPolygon, ...
             'LineWidth', 2, 'Color', [255*(j-1) 255*(j-2) 255*(mod(j,2))]);

         % Display tracked points
         videoFrame = insertMarker(videoFrame, visiblePoints{j}, '+', ...
             'Color', 'white');
         % Reset the points
         oldPoints{j} = visiblePoints{j};
         setPoints(pointTracker{j}, oldPoints{j});
     end
    % Display the annotated video frame using the video player object
     step(videoPlayer, videoFrame);
     step(videoFWriter,videoFrame);
end
```

Listing 1.3: Main Loop for tracking

1.2.2 Estimate Geometric Transform

As the car moves, feature points may get transformed and thus it becomes necessary to compute the transform between two consecutive frames. In this step, the transform is estimated between all the detected features between previous and current frame. To generate the bounding box in the current frame, the obtained transformed is multiplied with the coordinates of the bounding box in the previous frame which gives the new coordinates of the box.

1.3 Output

The figure below shows the final output image after doing all the above mentioned steps. The white points shown in the figure are the tracked features across the frames. The algorithm was able to detect and track fours cars across the whole video without leaving any feature at any point of time.

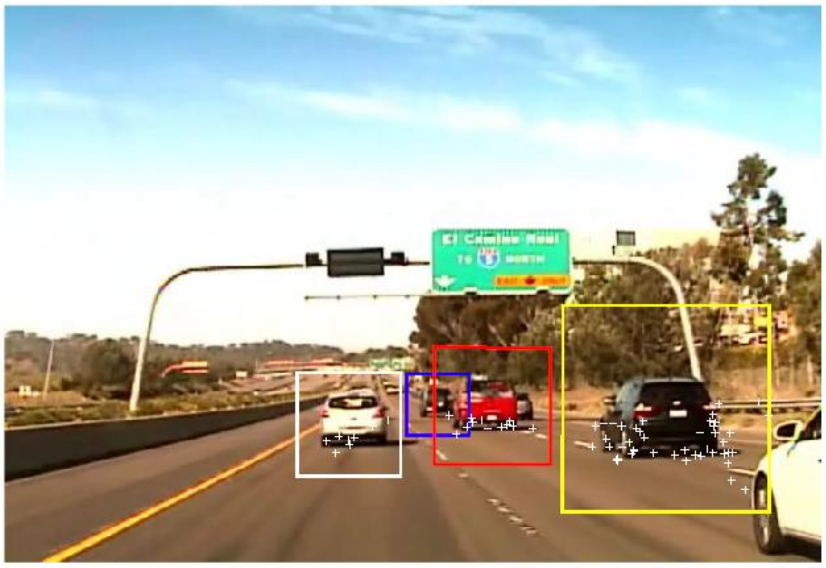

Figure 1.2: Output Image