A Study on Multi-Object Motion Tracking Computer Vision Project Report

Thomas Binu

November 24, 2016

1 Introduction

The goal of this project is to track and analyze moving traffic from a video footage. We will examine some computer vision based techniques for multi-object motion tracking and simulate a working sample using MATLAB. Motion tracking and speed estimation through computer vision have numerous application in the industry in the areas of Autonomous Vehicle Systems, Traffic Control, Geographical Information Systems, Motion Blur Reduction etc.

1.1 Defining the problem

The project can be broken down into the two parts

1. Identifying moving vehicles from the background *i.e* the foreground object detection.
2. Tracking and predicting the motion of vehicles *i.e* object tracking

1.2 Foreground object detection

Since the objective is to track moving vehicles from a video,we can rule out approaches based on segmenting foreground objects from still images, and turn our focus to detecting relative motion of objects between different images. This way, our solution become applicable to moving objects of any shape or size or color.

The first step is to identify foreground objects in the given video. This technique is called *background subtraction* and is commonly used in traffic monitoring, objecting tracking and advanced human interfaces. A flexible and reliable method of background subtraction should handle

1. Subtle illumination changes to the background because of lighting conditions.
2. Noise or minute high-frequency repetitive motions such as movement of leaves, grass, dust etc.

1.2.1 Naive Approach

A rough estimation of the foreground objects can be made by using simple *frame differencing* ie taking the difference between two consecutive frames with a threshold. Depending on the value of the threshold used and frame differences, the quality of the results will differ but in general, this approach is erratic and unreliable.

This approach can be further refined by using a mean or median of the previous n frames to estimate the background and construct a foreground mask by taking the difference of the background and the current frame.

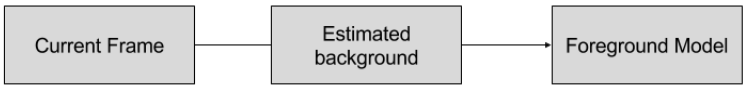

Figure 1: Foreground estimation using frame differencing

1.2.2 Background Subtraction using Gaussian Mixture Model

This approach based on a paper by Chris Stauffer and W.E.L Grimson discusses using a mixture of adaptive *Gaussian for background subtraction* By using a set of training frames, a multivariate Gaussian Mixture probabilistic model is then created for the background pixels. That pixels which do not match with the model is classified as a foreground pixel.

Advantages of Background subtraction using GMM

1. There is a different threshold for each pixel based based on its location and training set of frames
2. The pixel wise threshold is adapted over time.
3. This model can handle gradual variations in lighting conditions and high-frequency noise.

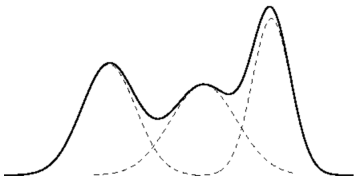

Figure 2: Gaussian mixture models (GMM)

1.3 Motion Tracking

1.3.1 Noise removal

After you perform background subtraction on the input video, the foreground binary image is run through a preprocessing step called *morphological opening* to remove the excess salt and pepper noise. This process involves sliding a small shape or template called the structuring element over the image. The structuring element is placed at all possible active pixels over the image and compared with the neighboring pixels, and the center pixel is either set to 0 or 1 depending on the operation. Morphological opening is erosion followed by dilation, and is denoted by

$$A \circ B = (A \ominus B) \oplus B$$
$$\text{where} \ominus \text{ is erosion}, \oplus \text{ is dilation}$$

1.3.2 Blob Analysis

The foreground detector step will yield a binary image with multiple foreground objects. The next step will be to identify and track the objects between frames. Separating the foreground objects from the image and classifying and computing its various properties is called *Blob Analysis*. Various algorithms exist for Blob Analysis. Given below are two common approaches

1. *Grass-Fire Algorithm*: The algorithm scans the image from top right to bottom left. If it finds a foreground pixel(with a value of 1) it will be marked as 'visited' and added as a part of a blob object. Next, taking this pixel as the center, the algorithm checks the neighboring pixels to see if any of them

are also a foreground pixel. If yes, those pixels are added to the current blob object and marked as 'visited'. Then this visited pixel is taken as the center, and its neighbors are checked as described earlier. After all the pixels in the foreground object are added the the blob object, the algorithm moves through the image until the next foreground pixel is identified, which is taken as the second blob object. This process continues until all blobs are segmented.

2. *Laplacian of Gaussian*: One of the most common approaches for blob detection is based on the Laplacian of Gaussian(LoG). The basic algorithm is as follows

   (a) Filter the image with Gaussian at different scales

   (b) Find the difference in the images from above to approximate the Laplacian

   (c) Thus, it will yield maximal responses if applied to an image neighborhood that contains a similar blob structure at a corresponding scale. By searching for scale-space extrema of the LoG, we can therefore detect circular blob structures.

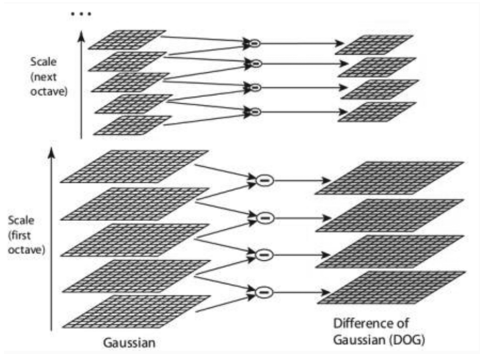

Figure 3: Difference of Gaussian

1.4 Kalman Filter

Kalman Filter, based of the seminal paper published in the 1960s, is an algorithm that uses a series of measurements over time to calculate the next measurement. Kalman Filter has a variety of application in the industry, especially in the field of navigation control and guidance. The algorithm can run in real time, with the only input being the current measurement and previously calculated state and uncertainty matrix, with no additional data required.

When applied to motion estimation, Kalman filter will try to calculate the position of the moving object based on measurements of a set of previous positions. If the position measurements of the moving vehicle stop getting updated, for example if the vehicle is obstructed by another large vehicle the Kalman filter will try to predict the current position of the vehicle based on the variations and patterns calculated from the previous measurements.

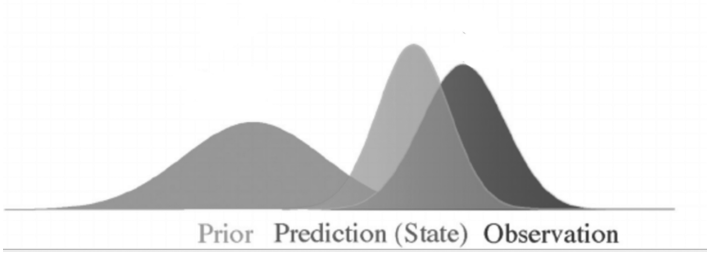

Figure 4: Kalman filter basic workflow

2 Implementation

2.1 Key Matlab Functions

Given below is a list of Matlab functions that was used in this project.

1. System Objects: Special MATLAB objects designed to handle streaming data efficiently. They have a step function to process streaming data and run the system object.
2. Foreground Detector: Computer a foreground mask based on a series of training frames.
3. imopen: To perform morphological opening. This method is used to remove salt and pepper noise from the output of Foreground detector.
4. Blob Analysis: To perform connected component analysis in a binary image. This method will be used in the implementation to compute the bounding box of foreground objects.

2.2 Algorithm

1. Read the video file using vision.VideoReader System Object.
2. Instantiate the foreground detector system object. A value of 50-100 for training frames works well for the example
3. Instantiate the blob analysis detector. A value of 150 for minimum blob area works well for removing noise.
4. Train the foreground detector by running through the first 100 frames in the video.
5. While loop through the video frame-by-frame
   (a) Calculate the foreground by using foreground detector.
   (b) Clean up the foreground image using morphological opening.
   (c) Calculate the bounding box of the blobs in the image using blob analysis
   (d) Using the bounding box to draw a rectangle over the foreground images.

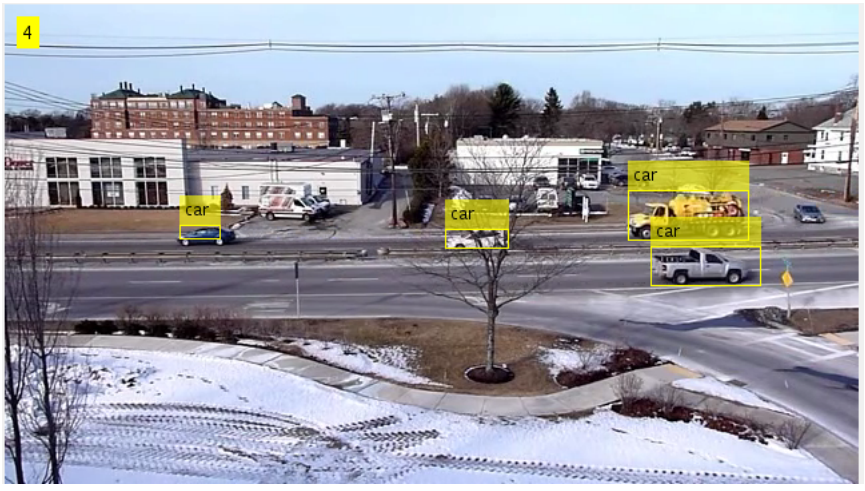

Figure 5: A single frame from the output video

References

[1] https://www.cs.unc.edu/ welch/kalman/media/pdf/Kalman1960.pdf

[2] https://www.ncbi.nlm.nih.gov/pubmed/23757570

[3] http://www.isprs.org/proceedings/XXXVIII/part3/a/pdf/61_XXXVIII-part3A.pdf

[4] https://www.mathworks.com/help/vision/examples/motion-based-multiple-object-tracking.html

[5] http://www.cs.unc.edu/ tracker/media/pdf/SIGGRAPH2001_CoursePack_08.pdf