Improved Object Tracking in Videos using Re-initialization

Sudipto Ghosh
Indian Institute of Engineering Science and Technology, Shibpur
sudiptog.ug2018@cs.iiests.ac.in

Research Internship
School of Electrical Sciences
IIT Bhubaneswar

June 7, 2021

Overview

The Task

Task:

- Understanding what *Object Tracking* in Computer Vision is and how it is different from *Object Detection*
- The library *OpenCV* offers several *Object Tracking* APIs, choose any three and understand their mechanism and difference. (Briefly)
- However the tracking failures are irreconcilable in case of Traffic Videos as cars move in and out of the frame rapidly. Methods to improve tracking.

Introduction

- Videos/Motion Images are a series of picture frames, projected at high speed (30fps)
- Based on persistence of vision, that is, the human brain perceives more than two images that are formed on the retina within a time period of $\frac{1}{16}$ of a second.
- Hence, Object Tracking involves capturing frames from videos in some intervals, and then following the object(to track) in all frames.
- The Tracking was accomplished using three different algorithms - namely the Boosting, MIL, KCF Algorithms.

Introduction (cont.)

Object Tracking: Locating an object in successive frames of a video. Various object tracking algorithms track the displacement of one of several objects appearing in a scene, captured using cameras.

Problems:

- The use of a particular view entails the possibility of partial occlusion of the targeted object.
  The problem can be solved by the use of more cameras in the same scenario, however such requirements can become expensive and computationally heavy for the algorithm's performance.
- Most scenarios contain multiple objects (some even possibly similar to the object being tracked), with the possibility of a confusion or occlusion among them.
  To adapt to this fact, tracking techniques (Optical Flow, Kalman filters, etc) may be used.

Object Tracking v/s Object Detection

In Computer Vision, **Object Detection** is scanning and searching for an object in an image.
If we detect cars from the image of a street, it is often referred to as object detection.

Object Tracking involves spying on something i.e. following it through different frames of a motion image.
In motion images like in animated GIFs or videos, a particular object is tracked i.e. details of how an object is moving, where is it going, or its speed is recorded.

Object Tracking v/s Object Detection (cont.)

There are two sets of ways to “track” an object.

- Tracking as **Series of Detections**:
  - In a video of moving traffic, different snapshots of the video are taken, (images/frames at different times)
  - The object(to track) is then detected in all the frames, like a car or a person.
  - By checking how the object has moved in different frames of the video, the object can be easily tracked (its trajectory can be traced,its velocity can be calculated, etc.)
- Since all the pixels of the video snap are processed many times, these kind of algorithms come under **dense method** of tracking and are usually slow.
- Also, this kind of method is not purely *tracking*, as the object is not really *tracked* but is *detected* at different points in time.

Object Tracking v/s Object Detection (cont.)

- **Detection with dynamics**:
  - Considering the same scenario, when we want to track a car/person on road, its position is checked at some time t
  - Next, the position of the object is estimated at some other time, (say $t+5$), i.e. the car's trajectory is estimated
  - Then, by comparing the estimated position and the actual position of the object at $t+5$, the difference between both is minimised, and the information is used to train the tracking algorithm.
- Here, since the pixels which are near the area containing the object to be tracked are being processed, it is referred to as the **sparse method** of tracking. Also, it is faster (processing of less pixels).

Object Tracking v/s Object Detection (cont.)

Hence, using *Detection with Dynamics*, **a smoother curve** is obtained, which indicates where the object is moving, since *estimates* are applied into tracking.

However, when we do tracking as only *Series of Detections*, we tend to get a *kind of trajectory*, as to where the object actually was at different points of time and joining these points to know its trajectory, which results in a **noisier curve**.

The Boosting Algorithm

This tracker is based on an online version of the AdaBoost Algorithm.

- It is an *Online Algorithm* i.e. data available in a sequential order is used to update the best predictor for future data at each step, as opposed to batch learning techniques. This is since frames are captured from the video in a sequential manner.
- It takes after the *Adaptive Boosting* ML Algorithm i.e. the output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier.
- The classifier needs to be trained at runtime with positive and negative examples of the object.
    - The initial bounding box supplied by the user is taken as a positive example for the object
    - The image patches outside the bounding box are treated as the background

The Boosting Algorithm (cont.)

- Given a new frame, the classifier is run on every pixel in the neighborhood of the previous location and the score of the classifier is recorded.
- The new location of the object is the one where the score is maximum.
- The location of the object detected in the previous step is then used as a positive example for the classifier.
- As more frames come in, the classifier is updated with the additional data.

The Boosting Algorithm (cont.)

1. **Pros:** The classifier can be adapted while tracking the object. Therefore appearance changes of the object (e.g. out of plane rotations, illumination changes) are handled quite naturally.
Source: Real-Time Tracking via On-line Boosting, January 2006, DOI: 10.5244/C.20.6

2. **Cons:** Tracking performance is mediocre. It does not reliably know when tracking has failed.

The Multiple Instance Learning (MIL) Tracker

The algorithm looks in a small neighborhood around the current location of the object to generate several potential positive examples. The method trains a discriminative classifier in an online manner to separate the object from the background. The classifier bootstraps itself by using the current tracker state to extract positive and negative examples from the current frame.

- In MIL, regions in images are not specified as positive and negative examples, but rather, as positive and negative "bags".
- The collection of images in the positive bag are not all positive examples. Instead, only one image in the positive bag is a positive example.
- A positive bag contains the patch centered on the current location of the object and also patches in a small neighborhood around it.

The Multiple Instance Learning (MIL) Tracker (cont.)

- Hence, even if the current location of the tracked object is not accurate, when samples from the neighborhood of the current location are put in the positive bag, there is a fair chance that the bag contains at least one image in which the object is nicely centered

Pros: The performance is pretty good. It does not drift as much as the Boosting Tracker and it does a reasonable job under partial occlusion.

Cons: Tracking failure is not reported reliably. Does not recover from full occlusion.

Kernelized Correlation Filters

- This tracker builds on the ideas presented in the previous two trackers. This tracker utilizes the fact that the multiple positive samples used in the MIL tracker have large overlapping regions.
- The basic idea of the correlation filter tracking is estimating an optimal image filter such that the filtration with the input image produces a desired response. The desired response is typically of a Gaussian shape centered at the target location, so the score decreases with the distance.
- The filter is trained from translated (shifted) instances of the target patch. When testing, the response of the filter is evaluated and the maximum gives the new position of the target. The filter is trained on-line and updated successively with every frame in order the tracker adapts to moderate target changes.

Kernelized Correlation Filters (cont.)

- Major advantage of the correlation filter tracker is the computation efficiency. The reason is that the computation can be performed efficiently in the Fourier domain. Thus the tracker runs in super real time (several hundreds FPS)
- A drawback, however is that it doesn't handle Occlusion Well.

Improving Tracking

- The Functional APIs of OpenCV Object Trackers require users to select an initial ROI which is then tracked through the remaining frames
- However, when used on real time video streams (eg. Traffic Movement data), annotating the initial ROIs is not possible, owing to the large number of cars in the frame.
- Further, monitoring Traffic Videos using the naive trackers does not yield satisfactory results. This is since cars move in and out of frames pretty fast, thereby causing too many tracking failures.
- Thus, to improve the performance, a re-initialisation of the tracking must be performed.

Initial ROI Selection

- **In real-time video streams (eg. Traffic Movement data), annotating the initial ROIs is not possible, owing to the large number of cars in the frame.**
  Hence, the requirement is of an object detection algorithm that can perform in Real Time.
- The YOLOv3 Algorithm is used to perform the above task as it provides reasonably accurate results, in real time.
- Thus the bounding boxes around the objects detected by YOLO are used as initial ROIs and an attempt is made to track them through the remaining frames.

Re-Initialising Tracking

- **Monitoring Traffic Videos using the naive trackers does not yield satisfactory results. This is since cars move in and out of frames pretty fast, thereby causing too many tracking failures.**
  As a remedy to the above problem, the method above uses a re-initialisation methodology.
- Tracking failures are counted. Whenever the number of failures exceed a user supplied threshold, the tracking is re-initialised.
- That is, YOLOv3 is run on the current frame and the objects detected are used to further perform the tracking. This increases performance capabilities several notches.

Conclusion

- As an extension to the work, the team hopes to develop a strategy to intelligently infer a failure threshold, rather than relying on a user supplied threshold value exceeding which tracking shall be reinitialised.
- Also, some improvisations in the code structure shall be implemented, to make it easily integrable with the video stream sourced from Closed Circuit Cameras, used to monitor traffic.

Thank You

Thank You