Tracking Objects in YouTube Videos

Author: Aparajith Sairam
Guide: David Held
Instructor: Prof. Silvio Savarese

Abstract:

Object tracking is an area of active research in Computer Vision. Tracking is a challenging problem, fundamentally due to the loss of information caused by the projection of a 3D scene on a 2D image. In this project, we are interested in tracking objects in YouTube videos, which are very diverse and have high variation in terms of changes in illumination, scale of objects, viewpoint, intra class variablity, etc. making it even more complicated. In this project, we briefly explore different techniques for object tracking and focus on one particular technique, "Tracking-Learning-Detection" (TLD) [1]. An open source implementation of TLD, OpenTLD [2], was tested and some of its capabilities which are useful for tracking objects in YouTube videos have been noted, along with the changes needed to make it capable of being used effectively for this project. Some of these vital improvements were made to OpenTLD, like making it view point invariant, and the resulting performance has been compared to the original version.

1. Introduction

The aim of this project is to detect and track an object present in YouTube videos, given a single image or multiple images of that same object as input. This is a part of a larger project, which aims at construction of a database of 3D objects by crawling many YouTube videos. Since, for 3D reconstruction, tracking the object of interest is a key step, it is crucial step of the larger project. Consider Fig. 1(a), where an image of a soccer ball has been given with a bounding box for the ball. Our goal is to detect all the occurrences of the ball in a video and track its location in every frame where it appears as depicted in Fig. 2.

Figure 1. Soccer ball with bounding box defined

Figure 2. Tracking the soccer ball after defining its bounding box in the first frame

Tracking objects in a video is a difficult problem due to loss of information caused by the projection of a 3D world on a 2D image. Some of the ways to simplify this problem are by having some prior knowledge of the video, like the motion of the object, or by imposing constraints on the video such as the object moves smoothly without abrupt changes. It is common for tracking algorithms to assume that the object to track is always in view and is never occluded. However, these constraints cannot be imposed on YouTube videos, making the problem much more complicated. A tracking technique for YouTube videos should factor in changes in object scale, orientation, illumination, view point, occlusions and intra class variability.

A technique for long term tracking of unknown objects in a video stream, "Tracking-Learning-Detection" (TLD), has been proposed by Z. Kalal et al. [1], and an open source implementation of their technique, OpenTLD [2] has been made available by the authors. This technique is the most promising of all the techniques for object tracking in YouTube videos. So, we investigated both the method of tracking and the implementation. OpenTLD was configured to work in a Windows machine and was tested with many different videos as input. The videos used for testing were selected such that, different aspects of the system like orientation invariance, scale invariance, etc. could be evaluated. The results of the tests revealed aspects of the system which could be improved for tracking in YouTube videos. OpenTLD only needs the bounding box of the object to be tracked be provided in the start frame of the video. We modified OpenTLD to accept multiple images as input instead of a single image (first frame). This gave significant benefits, which makes it better suited for tracking in YouTube videos. Some examples comparing the modified version of OpenTLD with the original version have been enlisted in this paper.

2.1 Review of Previous Work

The aim of an object tracker is to generate the trajectory of an object over time by locating its position in every frame of the video where it is visible. A detailed survey of techniques for object tracking has been done by Yilmaz, A. et al. [3]. Here we will enlist some of the categories of object tracking techniques. One way of categorizing them is in the following manner: Point tracking, Kernel tracking and Silhouette tracking.

2.1.1 Point Tracking

Point tracking can be formulated as the correspondence of detected objects represented by points across frames. Point correspondence is a complicated problem, especially in the presence of occlusions, misdetections, entries, and exits of objects. This approach requires an external mechanism for detecting objects in every frame. Point tracking approaches can further be classified as deterministic and probabilistic. The deterministic methods use qualitative motion heuristics to constrain the correspondence problem. One such approach for tracking is MGE tracker [4]. On the other hand, probabilistic methods explicitly take the object measurement and take uncertainties into account to establish correspondence. Kalman Filter [5] is a well-known example of these kinds of methods.

2.1.2 Kernel Tracking

Objects are tracked by computing the motion of the kernel in consecutive frames. Kernel refers to object shape and appearance. The motion is usually in the form of a parametric transformation such as translation, rotation, and affine. Kernel tracking can be divided into two major categories, Template and density based appearance models and Multi-view appearance models. Mean Shift [6] is an example of Template based tracking. Eigentracking [7] is a representative work which falls under the category of Multi-view appearance models.

2.1.3 Silhouette Tracking

Tracking is performed by estimating the object region in each frame. Silhouette tracking methods use the information encoded inside the object region. This information can be in the form of appearance density and shape models which are usually in the form of edge maps. Silhouette trackers can be divided into two categories, one using shape matching approach and the other using contour tracking approach. Shape matching approaches search for the object silhouette in the current frame. Contour tracking approaches, on the other hand, evolve an initial contour to its new position in the current frame by either using the state space models or direct minimization of some energy functional. State space models [8] use a shape matching approach, while Hough transform [9] uses contour tracking approach.

2.1.4 Tracking-Learning-Detection

This is a novel approach to tracking that explicitly decomposes the long-term tracking task into three distinct tasks, tracking, learning and detection. The tracker follows the object from frame to frame. The detector localizes all appearances that have been observed so far and corrects the tracker if necessary. The learning estimates detector's errors and updates it to avoid these errors in the future.
The chief goal of this approach of tracking is long-term tracking of unknown objects in a video stream. The authors of TLD have an open source implementation of their technique, OpenTLD. Given, the bounding box of the object to be tracked in the first frame of the video, tracking of the object takes place throughout the video, as long as there are no drastic view point changes in the video. The different components of the TLD framework are diagrammed in Fig. 3, taken from the original work [1].

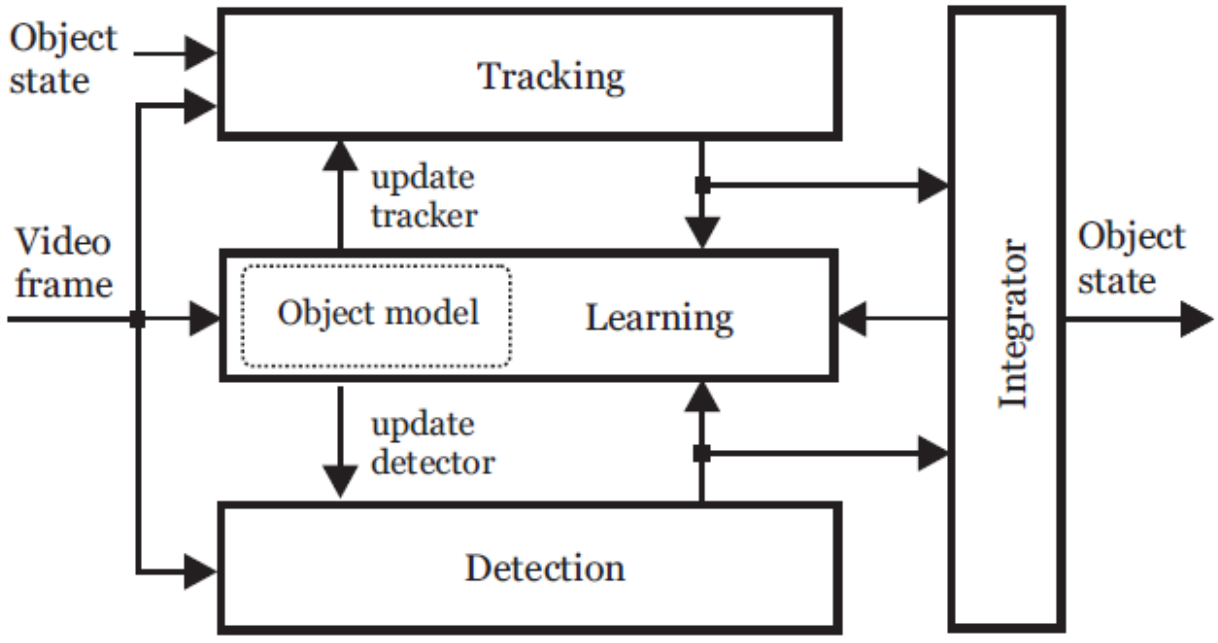

Figure 3. Block diagram of TLD framework

2.1.4.1 Object Tracker

TLD uses Median Flow tracker with failure detection [10]. Median Flow tracker estimates the motion of the object represented by a bounding box between consecutive frames. Internally the

tracker uses Lucas-Kanade tracker [11]. Lucas-Kanade uses 2 levels of pyramid and represents the points by 10x10 patches. The tracker estimates displacements of a number of points within the object's bounding box, estimates their reliability, and votes with 50% of the most reliable displacements for the motion of the bounding box using median.

2.1.4.2 Object Detector

The detector used in TLD scans the input image by a scanning-window and for each patch decides about presence or absence of the object. All possible scales and shifts of an initial bounding box are generated for matching each patch to a target patch which contains the object to track. As the number of bounding boxes to be evaluated is large, the classification of every single patch has to be very efficient. TLD breaks down this task into three stages: patch variance, ensemble classifier and nearest neighbor.
The first stage of detection is the computation of patch variance. This stage rejects all patches for which gray-value variance is smaller than 50% of variance of the patch that was selected for tracking. This stage typically rejects more than 50% of non-object patches. The variance threshold restricts the maximal appearance change of the object.
The second stage of detection is ensemble classifier. The input to the ensemble is an image patch that was not rejected by the variance filter. The ensemble consists of n base classifiers. Each base classifier i performs a number of pixel comparisons on the patch resulting in a binary code x, which indexes to an array of posteriors Pi(y|x), where y can take a value 0 or 1. The posteriors of individual base classifiers are averaged and the ensemble classifies the patch as the object if the average posterior is larger than 50%.
The third stage of detection is nearest neighbor classifier. After filtering the patches by the variance filter and the ensemble classifier, typically, several of the bounding boxes are not decided yet. These patches are compared to all the patches that have been classified in the past by TLD as target patches. If the relative similarity between the patch in question and any of the target patches is greater than a threshold value, then it is classified as a target patch. Exact details on measurement of relative similarity can be found in the original work [1].

2.1.4.3 Integrator

Integrator combines the bounding box of the tracker and the bounding boxes of the detector into a single bounding box. If both the tracker and detector do not output any bounding box, then the object is declared as not visible. Otherwise the integrator outputs the maximally confident bounding box, measured using conservative similarity (again, refer [1] for details on this measurement). The tracker and detector represent fundamentally different estimates of the object state. The detector identifies the location of the object using previously identified templates, whereas the tracker extracts the location of the object using motion of the object from frame to frame. The learning component adds the newly identified template, given by the tracker, to the detector's training set, if the detector does not identify a target patch using previous training data.

2.1.4.4 Learning Component

The task of the learning component is to initialize the object detector in the first frame and update the detector in run-time using two types of experts called, the P-expert and the N-expert.
P-expert exploits the temporal structure in the video and assumes that the object moves along a trajectory. The P-expert remembers the location of the object in the previous frame and estimates the object location in current frame using a frame-to-frame tracker. If the detector labeled the current location as negative (i.e. made false negative error), the P-expert generates a positive example and adds it to the detector's training set.
N-expert exploits the spatial structure in the video and assumes that the object can appear at a single

location only. The N-expert analyzes all responses of the detector in the current frame and the response produced by the tracker and selects the one that is the most confident. Patches that are not overlapping with the maximally confident patch are labeled as negative. The maximally confident patch re-initializes the location of the tracker.

2.2 Key Contributions

For implementing tracking of objects in YouTube videos, OpenTLD has been used as the starting point. The key contributions of our work are:

i. Understanding TLD and Reverse Engineering its implementation, OpenTLD
ii. Modifying OpenTLD such that multiple images can be provided as input as opposed to only the bounding box of the object in the first frame of the video
iii. Identifying the area in TLD algorithm, where multiple images can be made useful for improved tracking
iv. OpenTLD has translation and scale invariance; it becomes rotation invariant over time. However, it is quite limited in terms of viewpoint invariance. It does not have any support for intra class variations. Both of these issues, which are major problems for the goal of tracking of objects in YouTube videos, have been mitigated through the use of multiple images as training examples
v. Tests were conducted on OpenTLD and the modified version of OpenTLD, the results of which have been documented in this paper
vi. Originally, OpenTLD accepted a sequence of images as input, instead of a video. So, initially a MATLAB script was written for converting any video into a sequence of images, with the names of images in a specific format (which was not mentioned in the documentation of OpenTLD). We have now modified OpenTLD so that a video can be given as input directly, without having to first convert it into an image sequence
vii. A prototype UI has been built which allows the user to provide the input video and the training images

3.1 Summary of Technical Solution

TLD accepts only the bounding box of the object in the first frame of the video as input for tracking. This input is sufficient for tracking objects only if the video follows certain constraints. The video must not have drastic changes in view point, otherwise TLD fails in tracking. A majority of YouTube videos have changes in view point, which makes TLD unsuitable for tracking in YouTube videos, without any modifications. TLD was only designed to follow an object which can be represented in the first frame of the video. Naturally, it does not support intra class variations. For tracking objects in many YouTube videos, it is not possible to manually assign bounding box for the object of interest in each video's first frame. A plausible method for doing this is supplying a set of training images with bounding boxes around the object of interest in each image. Since, TLD accepts only one image as input, this is not possible.

Both these problems are solved by modifying TLD such that it accepts multiple images as input. We found a way to add training images to the inbuilt detector of TLD through the learning component of TLD. The learning component of TLD has two types of experts, P-expert and N-expert. We were able to change the implementation of OpenTLD, such that the P-expert interprets the many images that are provided as input to be positive examples. The portion of every input image within the bounding box is converted to a 15x15 patch which acts as a positive template. P-expert adds this template to the training set of the detector, enhancing its generality. Thus, whenever there is a drastic change in view, the tracker of TLD fails; however, the detector succeeds in locating the object if the appropriate template is added by any of the input images. Similarly, if different classes of an object are identified and the corresponding images are given as input, then intra-class variability will not pose a problem to this tracker.

3.2 Detailed Explanation of Technique

For using multiple images as input, there were two possible approaches: using a separate detector which accepts multiple images as input and does detection whenever TLD's tracker and detector both fail and then re-initialize the tracker. The second approach was to modify the inbuilt detector of TLD itself by providing it with additional training images. In this project, the second approach has been used, thus no external component had to be added to the TLD framework. To understand how this approach works, we have to look at how the TLD framework's learning component works thoroughly.

In the first frame of the video, the user provides the bounding box of the object to track as input. The first bounding box initializes both the tracker and detector. The portion of the first frame within the bounding box is converted to a 15x15 patch and given by the learning component to the detector as the first positive training template. The detector has two sets of training data, one for positive examples and one for negative examples. The detector compares every patch of every frame after the first to the training data templates. If the patch of image matches a positive template, then it is categorized as a potential location of the object. If the patch matches a negative template, then it is rejected straight away, as it is categorized as belonging to the background.

As long as the object's motion in video is contiguous, without any abrupt changes, the tracker provides the location of the object. The detector validates the location identified by the tracker in each frame. Ideally, both the tracker and detector should give the same locations in all frames. However, when the pose of the object changes; the detector fails in locating the object. This is where the learning component comes into the picture. Whenever, a tracker locates an object, and the detector fails in doing so, the learning component treats it as a false negative by the detector. The P-expert converts the portion of image within the bounding box given by the tracker into a 15x15 patch and adds it as a positive template to the detector's training set. In this manner different poses of the object can be captured by the detector. In this manner the size of positive training data grows. Suppose, the object goes out of view, and when it returns in the video, it comes in a different pose, then it can still be located by the detector, owing to the availability of training data in that pose.

Now let us consider how the negative training data grows. The detector identifies a set of object locations in every frame. The tracker identifies a unique location in every frame. All the patches that are identified as potential locations of the object by the detector, and are not by the tracker, get treated as false positives. The N-expert converts all such patches into 15x15 templates and adds it to the negative training data of the detector.

Now that we understand how the learning component works thoroughly, we can modify it for adding multiple images as input. Every training image that is given to the system in the beginning (with the bounding box) is given to the P-expert. The P-expert converts all the relevant portions of the images into 15x15 templates and gives it to the detector as positive training data. We need not (cannot) use the N-expert, as the background of the training images has nothing to do with the background of the video in which we want to perform tracking.

We can give multiple views of the object to track as input and they get stored as positive training data for the detector. Thus we achieve viewpoint invariance. Similarly, we can give different classes of an object as input and obtain tracking of the object in videos having different versions of the object. Thus we achieve a solution to intra class variability.

4. Experiments

Different videos were downloaded from YouTube for testing OpenTLD and the modified version of OpenTLD. The videos were chosen so as to test different aspects of the tracker, like rotation invariance, scale invariance, etc. I enlist some of these findings below, along with links to the videos.

4.1 Results

OpenTLD has translation and scale invariance. A video which shows this is given in [12]. The image given as input has the face of a person at a higher scale and at a different location from the first frame of the video.
OpenTLD is not inherently rotation invariant. If only one image is given as input and the pose of the object in the first frame of the video is different from the training image, then the tracker fails, up to the point in the video where the pose of the object matches that of the sample. It becomes rotation invariant after learning positive examples over time. A video which shows this is [13]. The car changes its orientation during the course of the video, but the tracking still works, although, there are some points when the tracking fails. This video also demonstrates that even though the tracker fails, TLD is able to recover at a further frame.
OpenTLD is quite limited in terms of viewpoint invariance. The modified version of OpenTLD is successfully at mitigating this problem. An example which shows the improvement is given by [14], which shows the original OpenTLD tracking, and [15], which shows the tracking performed by the modified version. Fig. 4 shows some snapshots from the video [15].

Figure 4. Snapshots from Soccer Video showing tracking under multiple view points

OpenTLD has no support for intra class variability. The modified version is capable of handling intra-class variability. The videos [16] and [17] show an example of intra class variability handled by the new version, which is not possible in the original version. The videos show two runs of the modified version of OpenTLD, with nearly identical training images (comprising of two types of chair) tracking both types of chair in the videos.

4.2 Future Work

The main limitation of OpenTLD (which persist in the modified version) which was found in the experiments is that the object in the video has to closely resemble at least one training image. The detector in TLD does not accept objects which have even small variations from the object in the training samples. The variations that we are referring to are in relation to intra-class variability, not to variations in scale or position. So, this problem can be mitigated by using a training data with large number of objects of varying classes.
There are some possible improvements that can be made to OpenTLD which will improve its abilities further for tracking in videos in offline mode. OpenTLD plays video in only the forward direction, since it is tracking in real time. Our final goal is 3D reconstruction, not real time tracking. So we can play in the backward direction too. Object tracking fails only when detector and tracker both fail. So, when both detector and tracker fail between frame A and frame B in forward direction, we can play the video in reverse direction, i.e. from B to A, and TLD's tracker may succeed in a frame going in backward direction from B, thus getting more data to reconstruct.

5. Conclusion

In this paper we studied the challenges involved in tracking objects in YouTube videos, proposed a method for achieving it and showed the results of implementing it. We briefly studied the TLD framework, found some important ways in which it can be modified to obtain better tracking through the use of multiple images of the object to track as input. We analyzed the TLD algorithm and found out where additional images can be used to enhance tracking. We showed some problems of using TLD for tracking in YouTube videos directly, through experimental results. We also demonstrated how some of the major problems in using TLD for tracking in YouTube videos have been solved by the modifications made to the tracking framework.

6. References

[1] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," *Pattern Analysis and Machine Intelligence* 2011.

[2] https://github.com/zk00006/OpenTLD

[3] Yilmaz, A., Javed, O., and Shah, M. 2006. Object tracking: A survey. *ACM Comput. Surv. 38*, 4, Article 13 (Dec. 2006), 45 pages. DOI = 10.1145/1177352.1177355

[4] SALARI, V. AND SETHI, I. K. 1990. Feature point correspondence in the presence of occlusion. *IEEE Trans. Patt. Analy. Mach. Intell. 12*, 1, 87–91.

[5] BROIDA, T. AND CHELLAPPA, R. 1986. Estimation of object motion parameters from noisy images. *IEEE Trans.Patt. Analy. Mach. Intell. 8*, 1, 90–99.

[6] COMANICIU, D., RAMESH, V., AND MEER, P. 2003. Kernel-based object tracking. *IEEE Trans. Patt. Analy. Mach. Intell. 25*, 564–575.

[7] BLACK, M. AND JEPSON, A. 1998. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *Int. J. Comput. Vision 26*, 1, 63–84.

[8] BERTALMIO, M., SAPIRO, G., AND RANDALL, G. 2000. Morphing active contours. *IEEE Trans. Patt. Analy. Mach. Intell. 22*, 7, 733–737.

[9] SATO, K. AND AGGARWAL, J. 2004. Temporal spatio-velocity transform and its application to tracking and interaction. *Comput. Vision Image Understand. 96*, 2, 100–128.

[10] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-Backward Error: Automatic Detection of Tracking Failures," International Conference on Pattern Recognition, pp. 23–26, 2010.

[11] J. Y. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm," Technical Report, Intel Microprocessor Research Labs, 1999.

[12] https://www.youtube.com/watch?v=PBjwyrvrHfw

[13] https://www.youtube.com/watch?v=tS6KdK5qCjo

[14] http://www.youtube.com/watch?v=JEV_OcY4hAE&list=UU6oIgYZPp8iVtMDPHg-ajuw

[15] http://www.youtube.com/watch?v=u5qOdJzohac&list=UU6oIgYZPp8iVtMDPHg-ajuw

[16] http://www.youtube.com/watch?v=pMXJKTDJvJE&list=UU6oIgYZPp8iVtMDPHg-ajuw

[17] http://www.youtube.com/watch?v=RQcnoXeWy1g&list=UU6oIgYZPp8iVtMDPHg-ajuw