Asian Journal of Convergence in Technology *Volume III, Issue III*
ISSN No.:2350-1146, I.F-2.71

TRACKING, LEARNING AND DETECTION OF AN OBJECT IN VIDEO

Miss Afrin R Pathan (*Author*)
Electronics Engineering
R.A.I.T.
Nerul, India
afrin.betti@gmail.com

Mrs. Poornima Talwai
Electronics Engineering
R.A.I.T.
Nerul, India
poornima.talwai@gmail.com

Abstract—**This paper investigate long-term tracking of unknown objects in a video stream. The object is defined by its location and extent in a single frame. In every frame that follows, the task is to determine the objects location and extent or indicate that the object is not present. A novel approach called Tracking-Learning-Detection (TLD) framework that explicitly decomposes the long term tracking task into tracking, learning and detection. The tracker follows the object from frame to frame. The locator localizes all appearances that have been observed so far and corrects the tracker if necessary. The learning estimates the detectors errors and updates it to avoid these errors in the future.**

I. INTRODUCTION

Consider a video stream taken by a hand-held camera depicting various objects moving in and out of the cameras field of view. Given a bounding box defining the object of interest in a single frame, our goal is to automatically determine the objects bounding box or indicate that the object is not visible in every frame that follows. The video stream is to be processed at frame-rate and the process should run indefinitely long. We refer to this task as long-term tracking.

To enable the long-term tracking, there are a number of problems which need to be addressed. The key problem is the detection of the object when it reappears in the cameras field of view. This problem is aggravated by the fact that the object may change its appearance thus making the appearance from the initial frame irrelevant. Next, a successful long-term tracker should handle scale and illumination changes, background clutter, partial occlusions and operate in real-time. The long-term tracking can be approached either from tracking or from detection perspectives. Tracking algorithms estimate the object motion. Trackers require only initialization, are fast and produce smooth trajectories. On the other hand, they accumulate error during run-time (drift) and typically fail if the object disappears from the camera view. Research in tracking aims at developing increasingly robust trackers that track longer. The post-failure behaviour is not directly addressed. Detection-based algorithms estimate the object location in every frame independently. Detectors do not drift and do not fail if the object disappears from the camera view. However, they require an offline training stage and therefore cannot be applied to unknown objects.

We introduce the design of a novel framework (TLD) that decomposes the long-term tracking task into three sub-tasks: tracking, learning and detection. Each sub-task is addressed by a single component and the components operate simultaneously. 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 detectors errors and updates it to avoid these errors in the future. While a wide range of trackers and detectors exist, we are not aware of any learning method that would be suitable for the TLD framework. Such a learning method should:

- deal with arbitrarily complex video streams where the tracking failures are frequent
- never degrade the detector if the video does not contain relevant information
- operate in real-time

Figure 1.1: The block diagram of the TLD framework.

For all these challenges, we rely on the various information sources contained in the video. Consider, for instance, a single patch denoting the object location in a single frame. This patch

defines not only the appearance of the object, but also determines the surrounding patches, which define the appearance of the background. When tracking the patch, one can discover different appearances of the same object as well as more appearances of the background. This is in contrast to standard machine learning approaches, where a single example is considered independent from other examples. This opens interesting questions how to effectively exploit the information in the video during learning.

A. Problem Formulation

Manual tracking requires the interaction with the user in every frame. Automated tracking methods use a priori information in order to initialize the tracking process automatically. In semi-automated tracking, user input is required in order to initialize the tracking process. Another challenge is introduced by appearance variations of the target itself. Intrinsic appearance variability includes pose variation and shape deformation, whereas extrinsic appearance variability includes illumination change, camera motion and different camera viewpoints. Approaches that maintain a template of the object of interest typically face the template update problem that relates to the question of how to update an existing template so that it remains a representative model. If the original template is never changed, it will eventually no longer be an accurate representation of the model. When the template is adapted to every change in appearance, errors will accumulate and the template will steadily drift away from the object. This problem is closely related to the stability-plasticity dilemma, which relates to the trade-off between the stability required to retain information and the plasticity required for new learning. This dilemma is faced by all learning systems [1]. Objects undergo occlusions when covered by other object or when they leave the field of view of the camera. In order to handle such cases, a mechanism is necessary that re-detects the object independently of its last position in the image. Requirements on the execution time pose another difficulty. Object tracking methods do not take the environment factors into consideration, and are therefore not efficient and effective. In Object tracking methods the environment constraints often results in high computational overhead and possibly low tracking accuracy. In the existing works of video object tracking, the state vector only includes the dynamics characteristics of the object, e.g., location, orientation, scale, etc. To tackle all these challenges, introduce the design of a novel framework (TLD) that decomposes the long-term tracking task into three sub-tasks: Detection, tracking and learning. Each sub-task is addressed by a single component and the components operate simultaneously. 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 detectors errors and updates it to avoid these errors in the future.

II. Literature Survey

Video tracking is the process of locating a moving object (or multiple objects) over time using a camera. It has a variety of uses, some of which are: human-computer interaction, security and surveillance, video communication and compression, augmented reality, traffic control, medical imaging [1] video editing Video tracking can be a time consuming process due to the amount of data that is contained in video. Adding further to the complexity is the possible need to use object recognition techniques for tracking [5].

P. Sand et al.[4] tells When tracking planar objects, the motion model is a 2D transformation (affine transformation or homography) of an image of the object (e.g. the initial frame) . Ramanan et al. [6] proposed When the target is a rigid 3D object, the motion model defines its aspect depending on its 3D position and orientation. Lowe et al. [1] tells for video compression, key frames are divided into macroblocks. The motion model is a disruption of a key frame, where each macroblock is translated by a motion vector given by the motion parameters. The image of deformable objects can be covered with a mesh; the motion of the object is defined by the position of the nodes of the mesh.

A. Match Moving

In cinematography, match moving is a cinematic technique that allows the insertion of computer graphics into live-action footage with correct position, scale, orientation, and motion relative to the photographed objects in the shot. The term is used loosely to describe several different methods of extracting camera motion information from a motion picture. Sometimes referred to as motion tracking, match moving is related to photoscoping and photogrammetry.

Match moving is sometimes confused with motion capture, which records the motion of objects, often human actors, rather than the camera. Typically, motion capture requires special cameras and sensors and a controlled environment (although recent developments such as the Kinetic camera have begun to change this). Match moving is also distinct from motion control photography, which uses mechanical hardware to execute multiple identical camera moves. Match moving, by contrast, is typically a software-based technology, applied after the fact to normal footage recorded in uncontrolled environments with an ordinary camera.

III. System Design

Object tracking methods do not take the environment factors into consideration, and are therefore not efficient and effective. In Object tracking methods the environment constraints often results in high computational overhead and possibly low tracking accuracy. In the existing works of video object tracking, the state vector only includes the dynamics characteristics of the object, e.g., location, orientation, scale, etc. A live video is captured from a web camera of 720p format which has a resolution of 1280 x 720(16: 9) having frame rate of 20fps. An object of interest is selected within the

frame of video by the user to initialize the process which is known as semi-automated system where no further user input is required. The object is the detected in each frame of the video by template matching method. A bounding box is formed at the detected object in the live video. The object is then tracked by checking each frame of the live video. But as the time goes on the dimension and shape of the object is altered. So initial image of an object is irrelevant at the point of time. So the learning processes changes the image of an object time to time when is shape or dimension is altered. And again the changed object is detected in the video by template matching method. This loop goes on till the object of interest is within the frame. Fig. 3.1 depicts the workflow of the approach.

- We explicitly decompose the long-term tracking task into tracking, learning and detection.
- Tracker estimates the object motion under the assumption that the object is visible and its motion is limited
- Detector performs full scanning of the image to localize all appearances that have been observed in the past.
- Learning observes performance of both, the tracker and the detector, identifies errors of the detector and generates training examples to avoid these errors in the future.

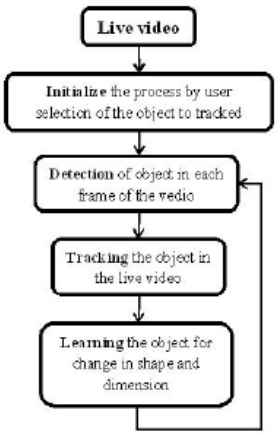

Fig. 1. The process is initialized by manually selecting the object of interest. No further user interaction is required

A. Object Tracking

Object tracking is the tasks of estimation of the motion trackers. Typically assumes that the object is visible throughout the sequence. Various representations of the object are used in practice, for example points articulated models or grid models [1]. Here we focus on the methods that represent the objects by geometric shapes and their motion is estimated between consecutive frames, i.e. the so-called frame-to-frame tracking [7]. Template tracking is the most straightforward approach in that case. The object is described by a target template and the motion is defined as a transformation that minimizes mismatch between the target template and the candidate patch. Template tracking can be either realized as static or adaptive [1].

Developed a novel learning method (P-N learning) which estimates the errors by a pair of "experts":

1. P-expert estimates missed detections, and
2. N-expert estimates false alarms.

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 labelled the current location as negative (i.e. made false negative error), the P-expert generates a positive example.

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 labelled as negative. The maximally confident patch re-initializes the location of the tracker.

The P-N learning is initialized by supervised training of so-called initial detector. In every frame, the P-N learning performs the following steps: (*i*) evaluation of the detector on the current frame, (*ii*) estimation of the detector errors using the P-N experts, (*iii*) update of the detector by labelled examples output by the experts. The detector obtained at the end of the learning is called the final detector.

B. Design Model

- DFD LEVEL- 0

At this Level-0 Camera capture the Live Video. And store this data into the template buffer as a template.

Fig. 2. DFD LEVEL- 0

- DFD LEVEL- 1

At this Level-1 Learn the object search and match object in the buffer and Detect in the frame.

Fig. 3. DFD LEVEL- 0

- DFD LEVEL- 0

At this level-2 we take the templates and match the objects in the frame and track the live position.

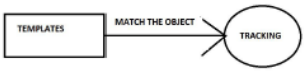

Fig. 4. DFD LEVEL- 0

C. Library used

OpenCV is a library of programming functions mainly aimed at real-time computer vision, developed by Intel, and now supported by Willow Garage and Itseez. It is free for use under the open source BSD license [1]. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate it [9].

D. Template Matching

Object detector is based on a sliding-window approach, which is illustrated in Fig.4.1. The image at the top is presented to the object detector, which then evaluates a classification function at certain predefined subwindows within each input image. Depending on the size of the initial object, we typically employ 50,000 to 200,000 subwindows for an image of VGA (640 x 480) resolution. Each subwindow is tested independently whether it contains the object of interest. Only if a subwindow is accepted by one stage in the cascade, the next stage is evaluated. Cascaded object detectors aim at rejecting as many non-relevant subwindows with a minimal amount of computation. The four stages that we use for image classification are shown below the input image. First, we use a background subtraction method in order to restrict the search space to foreground regions only. This stage requires a background model and is skipped if it is not available. In the second stage all subwindows are rejected that exhibit a variance lower than a certain threshold. The third stage comprises an ensemble classifier based on random ferns. The fourth stage consists of a template matching method that is based on the normalised correlation coefficient as a similarity measure. We handle overlapping accepted subwindows by employing a non-maximal suppression strategy.Each subwindow is tested independently whether it contains the object of interest. Only if a subwindow is accepted by one stage in the cascade, the next stage is evaluated. Cascaded object detectors aim at rejecting as many non-relevant subwindows with a minimal amount of computation [1]. The four stages that we use for image classi_cation are shown below the input image. First, we use a background subtraction method in order to restrict the search space to foreground regions only. This stage requires a background model and is skipped if it is not available. In the second stage all subwindows are rejected that exhibit a variance lower than a certain threshold. The third stage comprises an ensemble classifier based on random ferns. The fourth stage consists of a template matching method that is based on the normalised correlation coefficient as a similarity measure. Handle overlapping accepted subwindows by employing a non-maximal suppression strategy.[1]

1. Sliding-Window Approach

In sliding-window-based approaches for object detection, subimages of an input image are tested whether they contain the object of interest. Potentially, every possible subwindow in an input image might contain the object of interest.

We need two primary components:

a. Source image (I): The image in which we expect to find a match to the template image
b. Template image (T): The patch image which will be compared to the template image.

Our goal is to detect the highest matching area:

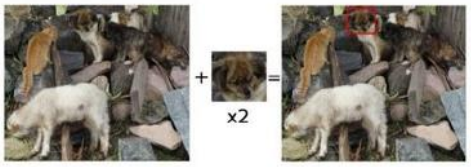

Fig. 5. Source image + template image

- To identify the matching area, we have to compare the template image against the source image by sliding it:

Fig. 6. Sliding-window-based approaches for object detection

- By sliding, we mean moving the patch one pixel at a time (left to right, up to down). At each location, a metric is calculated so it represents how good or bad the match at that location is (or how similar the patch is to that particular area of the source image).

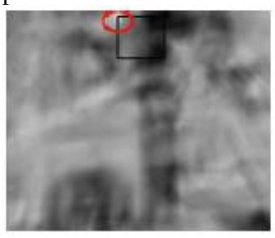

Fig. 7. Best matched result.

For each location of T over I, you store the metric in the result matrix (R). Each location (x,y) in R contains the match metric: the image above is the result R of sliding the patch with a metric TM_CCOEFF_NORMED. The brightest locations indicate the highest matches. As you can see, the location marked by the red circle is probably the one with the highest value, so that location (the rectangle formed by that point as a corner and width and height equal to the patch image) is considered the match.

2. Matching methods available in OpenCV

OpenCV implements Template matching in the function matchTemplate( ). The available methods are 6:

Square difference matching method (*method = CV_TM_SQDIFF*): These methods match the squared difference, so a perfect match will be 0 and bad matches will be large:

$$\boldsymbol{R}sq\_diff\,(x;\,y) = \Sigma_{x',y'}[T(x',\,y') - I(x + x',\,y + y')]^2$$

Correlation matching methods (*method = CV_TM_CCORR*) : These methods multiplicatively match the template against the image, so a perfect match will be large and bad matches will be small or 0.

$$\boldsymbol{R}ccorr(x;\,y) = \Sigma_{x',y'}[T(x',y') \,.\, I(x + x',y + y')]^2$$

Correlation coefficient matching methods (*method = CV_TM_CCOEFF*) : These methods match a template relative to its mean against the image relative to its mean, so a perfect match will be 1 and a perfect mismatch will be 1; a value of 0 simply means that there is no correlation (random alignments).

$$\boldsymbol{R}ccoeff\,(x;\,y) = \Sigma x',y'[T'(x',\,y') \,.\, I'(x + x',\,y + y')]^2$$

Where:

$$T'(x';\,y') = T(x';\,y') - 1/[(w{:}h)\Sigma_{x'',y''}\,T'(x'';y'')]$$

$$I'(x + x';\,y + y') = I(x + x';\,y + y') - 1/[(w{:}h)\Sigma_{x'';y''}\,I(x+x'';y+y'')]$$

Normalized methods: For each of the three methods just described, there are also normalized versions first developed by Galton [Galton] as described by Rodgers [Rodgers88]. The normalized methods are useful because, as mentioned previously, they can help reduce the effects of lighting differences between the template and the image. In each case, the normalization coefficient is the same:

$$Z(x;\,y) = \sqrt{\Sigma_{x';y'}\,T(x';\,y')^2} \,.\, \sqrt{\Sigma_{x';y'}\,I(x + x';\,y + y')^2}$$

The values for method that give the normalized computations are listed:

- *CV_TM_SQDIFF_NORMED*

$$\boldsymbol{R}sq\_diff\_normed(x;\,y) = \boldsymbol{R}sq\_diff\,(x;y) \,/\, Z(x;y)$$

- *CV_TM_CCORR_NORMED*

$$\boldsymbol{R}ccorr\_normed(x;\,y) = \boldsymbol{R}ccorr(x;y) \,/\, Z(x;y)$$

- *CV_TM_CCOEFF_NORMED*

$$\boldsymbol{R}ccoeff\_normed(x;\,y) = \boldsymbol{R}ccoeff\,(x;y) \,/\, Z(x;y)$$

As usual, we obtain more accurate matches (at the cost of more computations) as we move from simpler measures (square difference) to the more sophisticated ones (correlation coefficient)*:* In outdoor imagery especially, it's almost always better to use one of the normalized methods. Among those, correlation coefficient gives the most clearly delineated match but, as expected, at a greater computational cost. For a specific application, such as automatic parts inspection or tracking features in a video, you should try all the methods and find the speed and accuracy trade-off that best serves your needs.

IV. System Implemantation

Its best to do some test trials of all these settings and then choose the one that best trades off accuracy for speed in your application. We use the *CV_MINMAX* flag when normalizing; this tells the function to shift and scale the floating-point images so that all returned values are between 0 and 1. Shows the results of sweeping the face template over the source image using each of *cvMatchTemplate*()*s* available matching methods.

Fig. 8. Source image

and a template image:

Fig. 9. Template image

Generate the following result matrices (First row are the standard methods SQDIFF, CCORR and CCOEFF, second row are the same methods in its normalized version). In the first column, the darkest is the better match, for the other two columns, the brighter a location, the higher the match.

Fig. 10. SQDIFF, CCORR and CCOEFF

Fig. 11. Normalized version of SQDIFF, CCORR and CCOEFF

The right match is shown below (black rectangle around the face of the guy at the right). Notice that CCOEFF gave erroneous best matches, however its normalized version did it right, this may be due to the fact that only considering the "highest match" and not the other possible high matches.

Fig. 12. Best matched result.

V. Conclusion

In sequences containing occlusions, approaches based on Tracking-Learning-Detection outperform adaptive tracking-by-detection methods. Attribute this to the following reasons. Adaptive tracking-by-detection methods typically perform a form of self-learning, meaning that the output of a classifier is used for labelling unlabelled data. In Tracking-Learning-Detection, unlabelled data is explored by a tracking mechanism that is not dependent on the detector but bases its decision on a different measure, which in our case is the optical own. The performance of approaches based on Tracking-Learning-Detection is further improved by the automatic detection of tracking failures and by introducing criteria for validity that have to be met when learning is performed. Clearly, our approach heavily depends on the quality of the results delivered by the recursive tracker. Principally, the quality of the results can be improved in two ways. First, the timespan during which the tracker is following the object of interest correctly could be increased. This would present the object detector with more true positive examples. Second, The automatic detection of tracking failures could be improved, which would further prevent the object detector from drifting. A real-time implementation of the framework has been described in detail. And an extensive set of experiments was performed.

Acknowledgment

With great pleasure, I avail this opportunity to express my deep sense of gratitude to my guide, Prof. Poornima Talwai, for her spirited guidance and inspiration. I have deep sense of admiration for her innate goodness and inexhaustible enthusiasm. It helped me to work in right direction to attain desired objective. I am also thankful to our Project Coordinator, Prof. Trupti Agarkar and our Head of Department Dr. Vishwesh A. Vyawahare who devoted their valuable time and helped me in all possible ways towards partial completion of this work. I thank all those who have contributed directly or indirectly to this work. I am thankful to our Principal Dr. Ramesh Vasappanavara for his support and encouragement. I extend thanks to my friends who have done lots of nice things for me. I cannot end without thanking my lovely family for their encouragement.

References

[1] , Georg Nebehay, "Robust Object Tracking Based on Tracking-Learning-Detection",May 2012.

[2] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *International Joint Conference on Arti_cial Intelligence, vol.81, pp. 674-679*, 1981.

[3] J. Shi and C. Tomasi, "Good features to track," *Conference on Computer Vision and Pattern Recognition*, 1994.

[4] P. Sand and S. Teller, "Particle video: Long-range motion estimation using point trajectories," *International Journal of Computer Vision, vol. 80, no. 1, pp. 72-91*, 2008.

[5] L. Wang, W. Hu, and T. Tan, "Recent developments in human motion analysis," *Pattern Recognition, vol. 36, no. 3, pp. 585-601*, 2003.

[6] D. Ramanan, D. A. Forsyth, and A. Zisserman, "Tracking people by learning their appearance," *IEEE Transactions on Pattern Analysis and Machine Intelligence, pp.65-81*, 2007.

[7] P. Buehler, M. Everingham, D. P. Huttenlocher, and A. Zisserman, "Long term arm and hand tracking for continuous sign language TV broadcasts," *British Machine Vision Conference*, 2008.

[8] M. Isard and A. Blake, "CONDENSATION - Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision, vol. 29, no. 1, pp. 5-28*, 1998.