Interpolation Based Tracking for Fast Object Detection In Videos

Rahul Jain, Pramod Sankar K.*, C. V. Jawahar
Center for Visual Information Technology
IIIT-Hyderabad, INDIA
{*rahuljain@research., pramod_sankar@research., jawahar@*}*iiit.ac.in*

***Abstract*—Detecting objects in images and videos is very challenging due to i) large intra-class variety and ii) pose/scale variations. It is hard to build strong recognition engines for generic object categories, while applying them to large video collections is computationally infeasible (due to the explosion of frames to test). In this paper, we present a detection-by-interpolation framework, where object-tracking is achieved by interpolating between candidate object detections in a subset of the video frames. Given the location of an object in two frames of a video-shot, our algorithm tries to identify the locations of the object in the intermediate frames. We evaluate two tracking solutions based on greedy and dynamic programming approaches, and observe that a hybrid method gives significant performance boost as well as speedup in detection. On 6 hours of HD quality video, we were able to cut-down the detection time from 10000 hours to 1500 hours, while simultaneously improving the detection accuracy from 54% (of [1]) to 68%. As a result of this work, we build a dataset of 100,000 car images, spanning a wide range of viewpoints, scale and make; about 100 times larger than existing collections [2], [3].**

***Index Terms*—Object Detection, Object Tracking, Video.**

I. Introduction

One of the first steps towards understanding videos is to recognize the objects present in them. Accurate object recognition in videos would have significant impact in applications such as human-computer interaction, robotics, surveillance, computer-assisted driving, etc. However, there are many challenges toward localizing objects in videos: i) the relative motion of object(s) and camera results in a large variety of viewpoint and scale changes with-in a given video, ii) the motion blur across frames of a video is more pronounced, than in photographs, iii) the intra-class variety of most categories is much larger than those trained for, and iv) large quantities of data need to be processed for each video, since each second of a video consists of 25 frames/images.

In recent years, there has been significant progress in recognising generic objects such as *chair, bus, potted plant, etc.* from *images* [2]. Object recognition is typically posed as a classification problem. Features and classifiers are carefully selected and trained for this purpose. Use of dense SIFT-like descriptors and SVM-like discriminative classifiers have become the widely accepted candidates for visual recognition [4], [5], [6]. The localization of objects is typically performed using the sliding-window technique [7]. A window is moved across the image, each window is classified against the recognition engine. Multiple scales and aspect-ratios are used for the sliding window to cover different viewpoints.

Fig. 1. Car detections from our approach in frames where [1] could not localize the same object with such accuracy. We use the localization in neighboring frames to interpret the location in any given frame, resulting in more accurate and faster detection.

However, there are two issues with applying standard object localization approaches to video data. Firstly, due to the motion of the object, a wide variety of *novel* viewpoints and aspect ratios are encountered for a given object. It is almost impossible to train classifiers to handle such large variations. Hence traditional classifiers typically fail in many instances where the object is in an untrained orientation. Secondly, traditional recognition schemes are computationally intensive. A typical detection algorithm [1] runs at 1 frame/minute, which is not suitable to handling large volumes of data. Due to the slow nature of object detectors, researchers refrain from applying them on every frame, instead applying them only on every 10th or 25th frames.

In this paper, we shall propose an algorithm that addresses both these issues for object localization in large collection of videos. The premise of our work is that detectors need not be agnostic within a given video. An object found in one of the frames of the video, is very likely to appear in other frames of the same video. In such cases, the appearance of the object could be learnt and used to perform a more accurate classification. Further, we use KD-Trees to speedup matching of detections and candidate windows, which would otherwise be computationally expensive. Some example results are shown in Figure 1, where a typical object detector performs poorly or misses the object altogether, we are able to accurately localize the object. The contributions of our paper are:

1) Augmenting state-of-the-art object detectors with tracking mechanisms, thus improving detection performance by 14% over videos.
2) Speeding up object detection in videos; about twice as fast as state-of-art.
3) An approach to build a large dataset of 100K cars, consisting of various types-of-cars, viewpoints and scales.

*Pramod Sankar K. is now with Xerox Research, Webster, USA.

978-0-7695-4599-8/11 $26.00 © 2011 IEEE
DOI 10.1109/NCVPRIPG.2011.31

II. Previous Work

In the literature, object detection modules are typically designed for specific classes of objects, such as faces and person/pedestrians. Face detection is performed using simple features based on Haar-wavelets [8]. Histogram of Oriented Gradients (HoG) descriptors were designed and effectively used for the task of pedestrian detection [9], which was later applied to generic object detection [10], [11].

In case of videos, the detection time is an important criterion. There are two common approaches to speeding up object detection: the first is a cascade based approach [8], while the other is by using random forests [12]. Zhu *et al.* [13] integrated classifier-cascades with HoG [9], while Vedaldi *et al.* [7] uses a cascade of classifiers with progressively increasing complexity, for generic object detection. The major drawback of cascade-based approaches is that one cannot recover from the mistakes committed early in the cascade. The second approach of random forests [12], [14], uses multiple decision trees which typically use simple decision functions at each node. However, they have the disadvantages of over-fitting and lack of a principled method to tune and improve their classification performance.

In another direction of work, detection is used as a step in tracking objects. Kernel tracking [15] is typically performed by computing the motion of the object from one frame to the next. Such approaches are popular in the surveillance industry, to track objects, typically from a fixed camera. Apart from pure motion models, appearance of the object can be very useful in tracking objects. For example, SIFT features on facial regions was used to build 'tubes' of face detections within a shot [16], while clothing and hair information was used to track at longer intervals in [17]. In cases where object motion is hard to model, especially of non-rigid objects, the problem of tracking is re-posed as one of successive detection [18]. The tasks of detection and tracking are combined into a unifying framework in [19], and applied on tracking people in cluttered scenes.

III. Detection as Interpolation

Typical object detection considers each video as a bag-of-frames, thereby ignoring the temporal ordering and visual overlap present in videos. In a sliding window approach, each window of the video frames is considered as an isolated test case, and the classifier is applied on them separately. This results in a linear time complexity for these techniques, resulting in detection times of the order 1 frame/minute [1]. It is commonly observed that the detector accurately localizes an object in one frame, while it fails to locate the object in the very next frame, even in the presence of little visual variation. In this paper, we shall propose a joint detection-tracking approach, that effectively incorporates the temporal overlap in a video, to speedup as well as improve and object localization.

Let us assume that we are given the detection of an object in the first and last frames of a video-shot. Such a *seed* can be obtained by using manual annotation or by using a strong (albeit expensive) object detector. Manual annotation was traditionally considered expensive, but in recent times, crowd-sourced image labeling using online games [20] or with Mechanical Turk have become very common [21].

Given such seed detections, the problem is to infer the position of the object in the intermediate frames. We observe that the object traces a smooth but unknown path in the 3D volume of space-time. Example paths of cars in the space-time volume are shown in Figure 2 (left). Thus, the problem of object detection at every frame, can be reposed as one of identifying the path of the given object through the space-time volume. Such a search would benefit from the fact that the *appearance* of the object remains relatively unchanged through the video-shot. Since the object to be detected in each frame is known from the seeds, the search can focus on identifying exactly those windows that belong to the selected object; instead of comparing them with the variety of objects present in the training set of a typical classifier. For example, given the detections for a *red Ferrari*, the detector can focus only on finding the same red Ferrari in intermediate frames. Thus, it can avoid unnecessary comparisons of the candidate windows with a *black car* or a *blue bus*, vis-a-vis the support vectors of an SVM classifier.

Suppose, the object in the first frame has a location and scale of $L_1 = (x_1, y_1, w_1, h_1)$, while the object in last frame k of the video-shot is at $L_k = (x_k, y_k, w_k, h_k)$. If the object and camera are both stationary, the intermediate locations of the object in the video-shot would remain exactly the same as the first and last coordinates. If relative motion between the object and camera is a straight line, the object locations can be interpolated in the 4D coordinate space defined by the location and scale parameters. However, if the relative motion of the object with the camera is arbitrary, then the interpolation needs to take into account the appearance of the object. If the appearance of a window W_i is represented by any reasonable feature named F_{W_i}, then the interpolation uses the distance between two windows W_i and W_j (from two different frames in the videos) is defined by:

$$d_{W_i,W_j} = \alpha \cdot ||F_{W_i}, F_{W_j}|| + (1-\alpha) \cdot ||L_i, L_j||.$$

Here, the first component $||F_{W_i}, F_{W_j}||$ is a distance computed over the features of the windows W_i and W_j, and the second component $||L_i, L_j||$ measures the distance between the windows in the position/dimension space. We now present two techniques that can be used to obtain object detections without explicit recognition.

Greedy Extrapolation The greedy extrapolation method performs a one-directional search in the temporal dimension. Given a detection L_i in frame i, the localization L_{i+1} of the object in the next frame is assumed to be very close, $||L_{i+1}, L_i|| < \epsilon$. Using the statistics of the distribution of location and scale, the search is limited to within 100 and 50 pixels along the horizontal and vertical directions respectively, of the given seed detection. Six different aspect ratios are created for each given seed detection, and the pruned search window is scanned with a sliding window. Each window

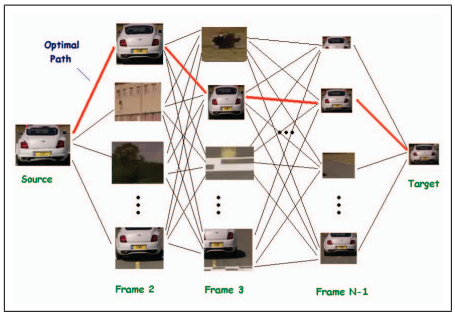

Fig. 2. Depiction of the optimal interpolation method using a multi-stage graph, solved with Dynamic Programming.

obtained is matched with the seed detection and the closest window is declared as the detection in the current frame. The newly obtained detection is used as the seed for localizing in the next consecutive frame. Typically, the greedy extrapolation needs to be restarted, if the detection strays off-the-course from the actual object path, which is very common in real datasets.

Optimal Interpolation The interpolation method does a bi-directional search of the sliding window space, in the temporal direction. The path of the object is constrained to begin and end at the given detection seeds at the first and last (or any two) frames. Let us call the detection in the first frame, the *source*, and the last frame as the *target*. The video-shot can now be modeled as a directed multi-stage graph, such as the one depicted in Figure 2 (right). Each stage of the graph, consists of nodes representing each window of the frame. Edges are present only across the different stages, and absent within each stage. The edge weight is given by the distance $w_{j,k} = d_{W_j,W_k}$ defined above, where $W_j \in Frame_i$ and $W_k \in Frame_{i+1}$. The cost of a given path $w_{source}, w_{source+1}, ..., w_{target-1}, w_{target}$, is given by $\sum_{i \in [source,target]} w_{i,i+1}$.

The optimal path from source to the target should minimize this objective function. While a naive exhaustive search on all paths is expensive, this function can be solved efficiently using Dynamic Programming (DP). At each frame $Frame_i$, the optimal path from the source to each window $W_j \in Frame_i$ is stored along with the accumulated distance. Thus at each stage of the multi-stage graph, only the optimal path to each window needs to be stored along with the cost associated with each path. When the target is reached in the graph, the optimal path is simply the minimum cost path from the source. The corresponding path from the source to the target, is output as the interpolated track of the given object. This process is depicted in Figure 2 (right).

A. Speeding up with Indexing Schemes

A single frame typically generates 50-60K sliding windows. Matching candidate windows across frames is thus an expensive task of order $O(N^2)$. A brute force matching would require about 4.5 hours for each pair of frames. However, the need for exact matching can be speeded-up using approximate nearest neighbor search. With approximate-NN, the data is indexed and nearest neighbors are obtained from those data points that fall in the same index.

The approximate NN approach of ours uses KD-Trees [22]. A KD-Tree partitions the feature space with axis-parallel hyperplanes. The algorithm splits the data in half at each level of the tree on the dimension for which the data exhibits the greatest variance. When multiple randomized trees are built, the split dimension is chosen randomly from the first D dimensions on which data has the greatest variance. Multiple trees define an overlapping split of the feature-space. These trees are looked up for NNs, by comparing the query with the bin-boundary at each level of the tree(s). The stopping criterion for the search is determined by parameters set by the user. Building a KD-tree from n points takes $O(n\ log\ n)$ time, by using a linear median finding algorithm. Inserting a new point takes $O(log\ n)$ and querying for nearest neighbors takes $O(n^{1-1/k}+m)$, where k is the dimension of the KD-Tree and m is the number of nearest neighbors

For KDTrees, we use the FLANN software provided by [22]. In the implementation, the algorithm first traverses the KD-tree and adds the unexplored branches in each node along the path to a priority queue. It then finds the closest center in the priority queue to the given query, and uses this node to restart the traversal. The process is stopped when a predetermined number of nodes are visited.

B. Hybrid Approach

From our experiments (see Table I), we observed that the greedy approach is fast but inaccurate. We observe that greedy approach works well on short sequences of frames, but fails over longer ranges. This is mostly due to the influence of background features in the windows, that confuse the matching scheme. It was observed that the features on the background are more stable than those on the car, especially when the pose of the car changes. Thus the extrapolation slowly drifts to the background. In the DP based approach, this drifting of the detection window is checked by the constraint that the final detection has to coincide with the target, which is known to be a location of the car. However, the DP based algorithm is slower than the greedy approach.

It is thus natural to combine both the techniques to create a fast and accurate detector. In the hybrid approach, the DP is not applied on every frame but only on equally spaced frames (say every K-th frame). The DP returns a set of most plausible detections at each K-th frame. The intermediate detections between these pairs of every K-th frame, is performed using the greedy approach. The results of the hybrid approach at various K values is given in Table I (below). As is expected, the hybrid method works best at a $K = 5$, as against larger steps, while sampling at one frame per second (K=25), results in a loss of performance in comparison with a full-fledged detection.

IV. Experiments

The dataset we consider for this paper comes from the popular BBC TV show *Top Gear* [23]. We evaluate the performance of our algorithms over 200 randomly selected shots from three episodes of Season 15 of Top Gear. Groundtruth

		PHOW		PHOG	
	Baseline [1]	Greedy	DP	Greedy	DP
Accuracy	0.54	0.44	0.66	0.43	0.62
Time/frame	67	17	42	8	36
		Hybrid Algorithm (PHOG)			
	Baseline [1]	K = 5	K = 10	K=25	K = 50
Accuracy	0.54	0.59	0.53	0.47	0.39
Time/frame	67	16	12	10	9

TABLE I
(ABOVE)DETECTION PERFORMANCE AND DETECTION TIME (IN SECONDS), FOR THE VARIOUS PROPOSED ALGORITHMS. (BELOW) DETECTION PERFORMANCE FOR THE HYBRID ALGORITHM; WITH A K OF 5 WE OBTAIN ACCEPTABLE PERFORMANCE WITH 4 TIMES SPEEDUP.

is created on two random frames extracted from each of these shots, for which the car in each frame is manually outlined. The performance of the car detectors is evaluated by the overlap measure [2] and defined as $Overlap = \frac{Intersection}{Union}$. $Intersection$ is defined as the region that is common to both the inferred detections and the groundtruth, while $Union$ is their combined area.

Pre-processing The video is first segmented to shots, using color histograms computed on multiple spatial blocks of each frame. The distance between two adjacent frames is computed, and a distance threshold is learnt from a small training dataset. We choose a tight distance threshold to ensure high precision of shot detection.

Seeds The initial *seed detection* that drive the extrapolation/interpolation algorithms could be obtained either manually or using an off-the-shelf object detector. In our experiments, we use the detector provided by [1] to obtain seed detections. The detector was applied on the first and last frames of every shot in the given video. Only those shots which have cars detected in them are fed to the next phase of interpolation across the shot. On average only 20% of the shots are found to contain cars in them, thus the detection at every frame can focus only on these shots.

Features We extract two different sets of features for each window, namely PHOG [10], [11] and PHOW [7]. The comparison of the two feature sets is shown in Table I. It can be seen that the PHOW feature is a better descriptor than PHOG, but the advantage of PHOG is that it is quicker to compute. It was observed that a large percentage of the actual time for detection is spent in feature extraction, in both PHOG and PHOW. For example, in the greedy method with PHOG, the feature computation alone was 6 seconds per frame. Thus, further speedup could be achieved using more simpler features which are quicker to compute; with a certain compromise on localization performance.

Building the Car Dataset From our experiments, we identify the best setting for large-scale car detection as using PHOW features with chi-square distance with a hybrid algorithm of $K = 5$. These settings were used to detect cars in all the six episodes of Season 15, amounting to 6 hours of HD quality video, i.e 1280x720 resolution, 25 fps (9 GB of compressed data). The resultant car dataset consists of more than 100K cars, in a variety of orientations, scale and make of the car.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we outlined a procedure to build large datasets for specific object categories. In doing so, we were able to improve detector performance as well as speed it up by replacing explicit recognition with simpler feature matching. In future work, we would address limitations such as tracking cars that could go out-of-frame within a shot. Also, we shall look to apply the techniques on larger datasets, as well as to other object categories. The datasets built from these videos would be used to re-train object detectors. It would be interesting to apply similar techniques to track non-rigid objects such as people.

REFERENCES

[1] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE PAMI*, vol. 32, no. 9, pp. 1627–1645, 2010.

[2] M. Everingham, L. V. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *IJCV*, pp. 303–338, 2010.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[4] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories." in *Workshop on Generative-Model Based Vision*, 2004.

[5] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. CVPR*, 2006, pp. 2169–2178.

[6] A. Vedaldi and S. Soatto, "Relaxed matching kernels for object recognition," in *Proc. CVPR*, 2008.

[7] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *Proc. ICCV*, 2009.

[8] P. A. Viola and M. J. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. CVPR*, 2001, pp. 511–518.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. CVPR*, 2005, pp. 886–893.

[10] A. Bosch, A. Zisserman, and X. Munoz, "Image classification using random forests and ferns," in *Proc. ICCV*, 2007.

[11] O. Chum and A. Zisserman, "An exemplar model for learning object classes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[12] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Computation*, vol. 9, no. 7, pp. 1545–1588, 1997.

[13] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *Proc. CVPR*, 2006, pp. 1491–1498.

[14] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *IEEE PAMI*, vol. 28, no. 9, pp. 1465–1479, 2006.

[15] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, December 2006.

[16] J. Sivic, M. Everingham, and A. Zisserman, "Person spotting: Video shot retrieval for face sets," in *ACM International Conference on Image and Video Retrieval*, 2005.

[17] D. Ramanan and S. B. S. Kakade, "Leveraging archival video for building face datasets," in *Proc. ICCV*, 2007.

[18] D. Ramanan, D. A. Forsyth, and A. Zisserman, "Tracking people by learning their appearance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 65–81, 2007.

[19] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *Proc. CVPR*, 2008, pp. 1–8.

[20] L. von Ahn, R. Liu, and M. Blum, "Peekaboom: A game for locating objects in images," in *Proc. ACM CHI*, 2006, pp. 55–64.

[21] A. Sorokin and D. Forsyth, "Utility data annotation with amazon mechanical turk," in *First IEEE Workshop on Internet Vision at CVPR*, 2008.

[22] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. VISAPP*, 2009.

[23] BBC TopGear at: http://www.bbc.co.uk/topgear.