Online, Real-Time Tracking Using a Category-to-Individual Detector*

David Hall and Pietro Perona

California Institute of Technology, USA
{dhall,perona}@vision.caltech.edu

Abstract. A method for online, real-time tracking of objects is presented. Tracking is treated as a repeated detection problem where potential target objects are identified with a pre-trained category detector and object identity across frames is established by individual-specific detectors. The individual detectors are (re-)trained online from a single positive example whenever there is a coincident category detection. This ensures that the tracker is robust to drift. Real-time operation is possible since an individual-object detector is obtained through elementary manipulations of the thresholds of the category detector and therefore only minimal additional computations are required. Our tracking algorithm is benchmarked against nine state-of-the-art trackers on two large, publicly available and challenging video datasets. We find that our algorithm is 10% more accurate and nearly as fast as the fastest of the competing algorithms, and it is as accurate but 20 times faster than the most accurate of the competing algorithms.

1 Introduction

The objective of tracking is to determine the size and location of a target object in a sequence of video frames, given the initial state of the target. It is important for a variety of applications, including the tracking of pedestrians in railway stations and airports; vehicles on the road for traffic monitoring and faces for interfacing people with computers. It is a well studied problem and although many offline tracking methods exist [2,8,9,30,33,34,36,37], the focus of this work will be on online trackers.

We present a novel method for real-time, online, appearance-based tracking. Tracking is treated as a detection problem where an individual-object detector is trained online to distinguish the target-to-be-tracked from background. Objects to track are first identified with a pre-trained category detector (a face, pedestrian or vehicle detector for example); each of the detections made by the category detector are now identified as individual targets to be tracked; for each of these targets an individual detector is learnt on-the-fly using IDBoost, a category-to-individual learning algorithm [27]. During tracking, the individual detector is updated using the currently detected sample of the target but only if

* Project Website: http://vision.caltech.edu/~dhall/projects/CIT/

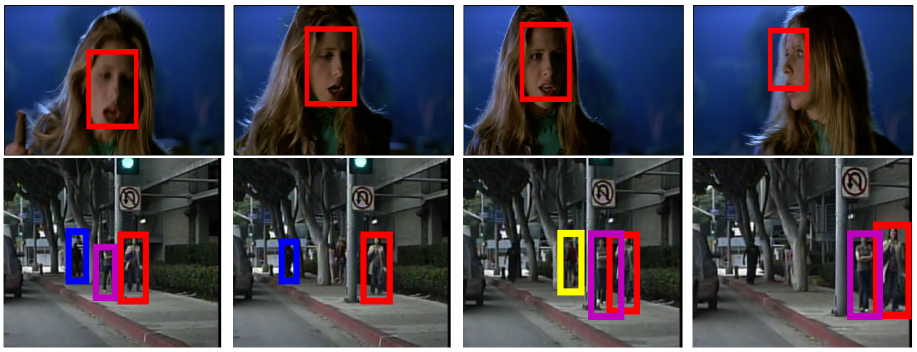

Fig. 1. Tracking individuals across a video sequence using our proposed tracking algorithm. (Top) a face from the Buffy dataset and (bottom) pedestrians from Caltech Pedestrians. The appearance of the face changes over time with a full frontal face example at initialisation (far left) to a quasi-frontal face in the final frame (far right). Despite the change in appearance, our tracker is able to track the target since the model for the target is updated over time (see Sec. 3). Pedestrians are also tracked successfully even when they are subject to occlusion. The magenta target is initialised in the first frame; is occluded by the light pole in the second and is successfully reacquired in the third. Individual tracker outputs are colour-coded. Each image is 30 frames apart (model update is still occurring every frame).

there is a coincident category detection. This ensures that the tracker is robust to drift. Crucially, the algorithm runs in real-time since the individual-object detector is obtained through elementary manipulations of the thresholds of the category detector. We then benchmark our tracking algorithm against 9 other, publicly available, state-of-the-art trackers on two challenging video datasets. We make two main contributions:

1. A fast, accurate, tracking algorithm that is robust to drift.
2. A careful and reliable benchmark of state-of-the-art trackers.

2 Related Work

Appearance-based tracking algorithms typically contain the following elements: 1) an appearance model for the target object to be tracked; 2) a search strategy to find potential candidates in subsequent frames that match the target; and 3) a mechanism to dynamically update the appearance model of the target so that changes in pose and illumination over time can be modelled.

A pitfall of dynamically updating the appearance model is that when trackers make mistakes this incorrect information is then incorporated into the model. The result is that the tracker no longer tracks the original target. This is known as drift. Designing algorithms that are robust to noisy updates is thus essential for good tracking performance.

Many trackers use generative appearance models. One of the representations used for target objects within this class of trackers are subspaces. Black and Jepson [11] learn offline an eigenbasis to model the target object along with particle filtering as a search mechanism. The IVT method of Ross et al. [38] proposes an online update of the target subspace over time using incremental PCA. Wu's ORIA [45] tracker also includes online updates but updates only occur if the new target is significantly different from the existing basis set.

Kernel based methods are also used to represent target objects. The influential work of Comaniciu et al. with their Kernel-Based Object Tracker (KMS) [13] represent the target object with a histogram in some feature space; a metric based on the Bhattacharyya coefficient is used to match the target to potential candidates; while the mean-shift algorithm [23] is used to efficiently search for these candidates. In traditional histogram-based algorithms information about the spatial distribution of features is lost; the FRAG [1] tracker of Adam et al. represents the target using multiple histograms obtained from many different patches in the target thus preserving some of the spatial information. Neither of these methods update the model online.

Alternative representations include probability distribution fields (DFT) [39]; sparse linear combinations of target and trivial templates (L1AP) [5] and the superpixels of (LOT) [35].

Discriminative appearance models are also widely used. Avidan's ensemble tracker [3] constructs a feature vector for every pixel. A classifier to separate pixels belonging to the target from those in the background is then trained by using an adaptive ensemble of weak classifiers. The compressive tracking (CT) algorithm of Zhang et al. [46] generates multi-scale features for the positive and negative samples and applies a sparse sampling matrix to reduce dimensionality. A naive Bayes classifier with online update is then used to classify windows as target or background. Grabner et al.'s online boosting method (OAB) [24] adaptively selects features to discriminate the object from the background.

To avoid drift, Grabner et al. [25] propose a semi-supervised, online boosting algorithm (SBT) where only the initial samples of the target are labelled while all of the self-learnt samples are unlabelled. The MIL tracker of Babenko et al. [4] uses multiple instance learning where a bag of positive samples are used to update the model. Kalal's [31] TLD tracker also approaches tracking as a semi-supervised learning problem with positive and negative examples being selected by an online classifier that has structural constraints. The BSBT tracker of Stalder et al. [41] combine the supervised and semi-supervised approaches into a single implementation.

For all of the approaches mentioned so far a sparse sampling strategy is used. This means that in each frame, positive samples are collected close to the predicted target while the negative samples are further from the target's centre. The CSK algorithm of Henriques et al. [28] proposes a different approach where a classifier is trained using all possible samples in a dense sampling strategy. The circulant structure of the problem is exploited allowing for not only efficient training but fast detection since all responses can be computed simultaneously rather than using a sliding-window scheme.

There are many benchmark datasets available for a number of vision problems. For pedestrians there is INRIA [14] and Caltech Pedestrians [17]; for unconstrained face recognition there is LFW [29]; and for person reidentification there is VIPeR [26]. Tracking, however, still lacks a decent benchmarking dataset although progress has been made recently to fill this gap. Wu et al. [44] have collected a benchmark that contains 50 sequences commonly used in the literature to evaluate tracking algorithms. While most algorithms have been evaluated on a subset of these sequences by their original authors, Wu et al. provide a far more comprehensive analysis by evaluating 29 tracking algorithms on all 50 sequences given the initial bounding box of the target object.

While the progress that has been made by Wu et al. is appreciable, we feel that the dataset is lacking for the following reasons: 1) The size of the dataset is too small; 2) The difficulty of most sequences is low with a focus on tracking only single objects; 3) About half of the sequences are unrealistic; they are in controlled environments and 4) trackers are perfectly initialised by a ground truth bounding box. This gives little insight into how robust trackers are to poor initialisation. Wu et al. address this by jittering the ground truth bounding box.

3 Approach

In this section we present the details of our tracking algorithm. We treat the tracking problem as a detection task and train an individual-object detector, online, to distinguish the target from background. We break down the algorithm into 5 major components: 1) identify target objects to track; 2) initialise the tracker; 3) evaluate the tracker on a new frame; 4) update the tracker; 5) stop the tracker. There is no assumption made about the number of objects being tracked and our algorithm is able to handle tracking multiple objects that enter and exit a scene. An outline of our approach is depicted in Figure 2.

3.1 Identify Target Objects to Track

Identifying new targets to track is a problem that most of the tracking literature avoids. It is usually assumed that an initial tracking window has already been provided either manually or by a ‘perfect’ category detector [38]. In this work, instead of providing initial locations by hand, a category detector is used. This is a more realistic setting under which trackers would operate since for most online applications, having a human operator identify potential targets would not be feasible; particularly if there are many targets to identify. This setting also allows us to evaluate how robust tracking algorithms are to poor initialisation.

A new target is identified if the category detector makes a detection and there is no coincident individual detection as shown in Figure 2.

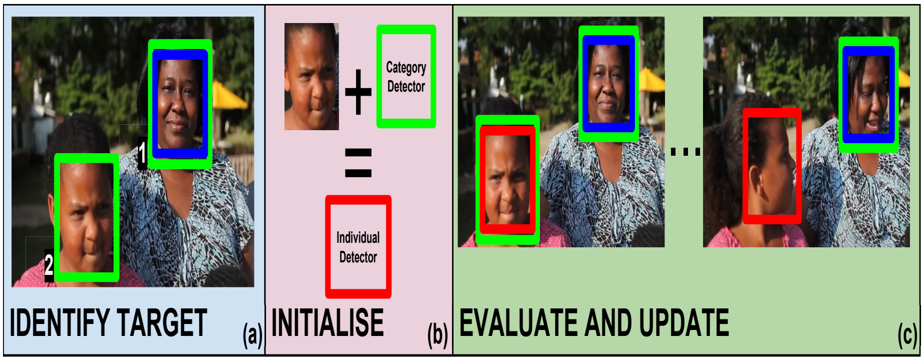

Fig. 2. Tracking Algorithm Outline. (a) A new target (2) is identified by a non-coincident category detection (green). There is also a category detection that coincides with an individual detection (blue) which indicates that this target (1) is already being tracked. (b) An individual detector (red) for target 2 is initialised using the category-to-individual learning algorithm in [27]. The only two pieces of information required to initialise the individual detector is a single positive example of the target and the category detector itself. (c) The individual detector is then evaluated on subsequent frames using a sliding window scheme. The location with the maximal score is identified as the new location of the target object. If a detection made by the individual detector is coincident with a detection made by the category detector then the individual detector is re-initialised with the current example of the target. If there is no coincident category detection the individual detector is not updated. If this occurs for more than a fixed number of frames tracking of that target stops.

Category detectors are trained offline using AdaBoost [22]. A boosted classifier takes feature vector $\mathbf{x} \in \mathbb{R}^D$ as input and outputs a binary decision:

$$H(\mathbf{x}) = \operatorname{sign}\left(\sum_{m=1}^{M} \alpha_m h_m(\mathbf{x}) - \tau\right) \tag{1}$$

where $h_m(\mathbf{x})$ is a weak classifier; α_m its weight and the threshold τ is chosen to produce the desired trade-off between false reject rate and false alarm rate.

The family of weak classifiers used are stumps. This means that given an input $\mathbf{x} \in \mathbb{R}^D$, the decision only depends on the j-th dimension of $\mathbf{x}$, a threshold $\theta \in \mathbb{R}$ and a polarity $p \in \{\pm 1\}$

$$h_m(\mathbf{x}) = \begin{cases} 1, & p_m x_{j_m} > p_m \theta_m \\ -1, & \text{otherwise} \end{cases} \tag{2}$$

Note that any boosting method and decision trees of any depth could be used to train the category detector; our proposed method is agnostic to these choices.

3.2 Initialise Tracker

Once a new target has been identified; a tracker can now be initialised to track the object. In this section we briefly outline IDBoost, the category-to-individual learning algorithm (see [27] for details) which allows us to efficiently train the individual detector that only detects the target object.

The approach for learning an individual detector from a category detector has four elements:

The individual detector is a boosted cascade of classifiers
Cascades are fast and their performance is state-of-the-art [7,16,18,42,43], making the individual detector suitable for real-time operation.

The individual detector is learnt from a single sample
The object identified by the category detector, which we will denote by $\mathbf{u}^0 \in \mathbb{R}^D$, is used as the single positive training example to train the individual detector. Costly computations are avoided by using a single positive sample and by not mining negative samples.

The individual detector uses the same M features that were selected by AdaBoost for the category detector
We will denote this set of features by $\mathbf{J} = (j_1, \ldots, j_M)$ where $j_m \in \{1, \ldots, D\}$ and the importance of each feature through the weights $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)$. Computational cost is reduced since there is no need to compute extra features (this has already been done for the category detector). There is also no cost for selecting which features to use since they are fixed.

Only the thresholds for a single weak classifier in the boosted cascade of the individual detector are modified
Selecting the thresholds for a single weak classifier $h'(x')$, which depends on a single feature $x' \in \mathbb{R}$, can be achieved at almost zero computational cost.

Consider the target that has been detected by the category detector and call u' the value of feature x'. An interval, centred at u', can now be defined. The width of this interval is determined offline, from a small validation set that contains tracks of a few individuals. The standard deviation of feature x' for a single individual across its track is calculated; the median standard deviation or spread σ' is then computed across the set of individuals. The spread σ' gives an estimate of the width of the interval.

Formally, the interval is $(u' - \beta\sigma', u' + \beta\sigma')$ where β is a free parameter that can be tuned experimentally. If the appearance of the individual is changing slowly over time, which is a reasonable assumption to make for video (since the difference from frame-to-frame is small), this interval represents the most likely values that the feature x' will take for that individual.

The weak classifier $h'(x')$ can thus be obtained from one training example and by only setting two thresholds (making the computational cost near zero):

$$h'(x';u',\sigma') = \begin{cases} 1 & u' - \beta\sigma' < x' < u' + \beta\sigma' \\ -1 & \text{otherwise.} \end{cases} \tag{3}$$

This weak classifier provides evidence for an individual being present (absent) if the feature x' lies inside (outside) the interval $(u' - \beta\sigma', u' + \beta\sigma')$.

An individual detector in the form of a cascaded boosted classifier can now be constructed. Given the set of features $\mathbf{J}$ and weights $\boldsymbol{\alpha}$ that were selected for the category detector, the single positive example $\mathbf{u}^0$ of the target, and an estimate of the spread $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_M)$ of the features $\mathbf{J}$, a classifier $F(\mathbf{x})$ for the target can be defined by:

$$F(\mathbf{x};\mathbf{u}^0,\boldsymbol{\sigma}) = \sum_{m=1}^{M} \alpha_m h'(x_{j_m};u_{j_m},\sigma_m) \tag{4}$$

3.3 Evaluate Tracker

Given the next frame in the video the individual detector $F(\mathbf{x};\mathbf{u}^0,\boldsymbol{\sigma})$ is evaluated using a sliding-window scheme. The location with the maximal classification score is identified as the new location of the target object. Since the individual detector is a cascade of boosted classifiers, sliding window detection is very efficient. It would also be possible to use a motion model to reduce the number of sub-windows evaluated (we have not done this).

3.4 Update Tracker

If at the new location of the target object (the one identified by the individual detector) there is also a coincident detection made by the category detector, then the individual detector is updated with the new sample, $\mathbf{u}^1$, of the target. The update procedure is then as simple as reinitialising the individual detector with the new sample which results in $F(\mathbf{x};\mathbf{u}^1,\boldsymbol{\sigma})$. The updated individual detector is then applied to the next frame and so on. If there fails to be a coincident category detection then the individual detector is not updated at all and is simply applied to the next frame. Figure 2 outlines this update procedure. Drift is avoided by only updating the model when the individual detector and category detector are coincident. This strategy ensures that only "good" positive samples are used to update the model. A category detection and individual detection are coincident if their overlap is greater than 50%.

3.5 Stop Tracker

There are two conditions, either of which can be met, for tracking of the target object to cease. The first is that there are no detections made by the individual detector for T_1 consecutive frames. This condition is usually met when the target object leaves the frame. The second is that the detections made by the individual detector fail to coincide with those made by the category detector for

T_2 consecutive frames. This condition is usually met when a tracker is initialised by a false detection made by the category detector. In this work we set both of these quantities to five frames.

4 Datasets

To benchmark the performance of our tracking algorithm we use two publicly available video datasets.

The first dataset is the Caltech Pedestrians [17,19] dataset. It contains 250,000 frames of video, at a resolution of 640x480, taken from a vehicle driving through regular traffic in an urban environment. The dataset is labelled with a total of 350,000 bounding boxes and around 1900 unique individual pedestrians. Around 30% of the frames have two or more pedestrians. Pedestrians are visible for 150 frames on average. The dataset is divided into 11 sets; 6 are used for training (S0-S5) and 5 are used for testing (S6-S10). This division is provided by the authors. The training set contains 192,000 bounding boxes and the test set contains 155,000. The authors also refer to experiments conducted on pedestrians over 50 pixels tall, with no or partial occlusion as the *reasonable evaluation setting.* There are 73,256 labelled bounding boxes and 769 unique individuals that meet these requirements in the test set. All of our experiments are conducted using the reasonable evaluation setting.

The second dataset used is the Buffy dataset [40]. It has been regularly used for the automatic labelling of faces of TV characters using subtitle and script text [40,21]. In this work we use the publicly available, ground truth labels of [6]. The dataset contains episodes 1–6 from season 5 of Buffy the Vampire Slayer with around 64,000 frames per episode. The faces are labelled using an automatic algorithm rather than by a human operator. This means that the ground truth labels are noisy. In total there are 317,831 labelled bounding boxes and 5513 unique face tracks across the six episodes. There is also a wide variety in the appearance of individuals in this dataset with many shots set outdoors and at night time; there are also a number of close-up shots.

Examples of the different individuals and how their appearance changes over time for both datasets are displayed in Figure 3.

The reason we benchmark tracking performance on these datasets is 1) because of their size and 2) because of their complexity. The benchmarking procedure of Wu et al. [44] only evaluates tracking performance on 50 different individuals. In this work we make this evaluation on around 6000 individuals; a 120-fold increase. The datasets here are also more complex. Having to track multiple objects of a similar appearance is a far more difficult task than the tracking of a single object, particularly when the target objects interact with each other, which may lead to trackers swapping identities. These datasets are also less biased since they were collected independently of the authors of any the trackers mentioned in section 1. They are also more realistic in the sense that the sequences haven't been generated in a lab as around half of the sequences in [44] are.

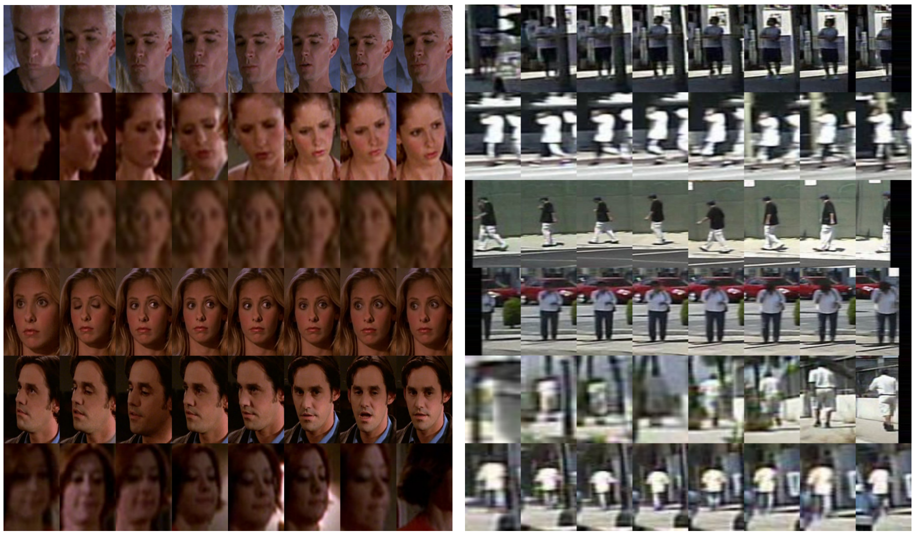

Fig. 3. Dataset Examples. (Left) Face tracks from the Buffy dataset. (Right) pedestrian tracks from Caltech Pedestrians. The ground-truth trajectories were sampled randomly from a video in each of the datasets. There are frontal and profile faces; the lighting can change considerably across a face track; facial expressions are varied. The pedestrians are low resolution; they change scale as the car mounted camera approaches and there is occlusion. The datasets are large with a combined total of 7500 individual tracks and 800,000 bounding boxes. They are also difficult due to the variety in pose, lighting, background and scale across a track as well as having multiple objects present at any one time. To the best of our knowledge this is one of the largest datasets that appearance-based tracking algorithms have been evaluated on.

5 Performance Metrics

There are a variety of methods [44,32,10] to measure the performance of a tracking algorithm. In this work we use three simple and intuitive measures to capture tracking performance (refer to Figure 4).

In a multiple-object tracking scenario, given a single frame, there are a number of options for determining whether a tracked target and ground-truth location match. The centre location error which is the Euclidean distance between the centre's of the two locations is one option [4,44,10]. This measure is not very robust to labelling error and relies heavily on the fact that the ground-truth is perfect. A more robust method to use is the bounding box overlap. Using the PASCAL criteria [20] a tracked object matches a ground-truth object if the area of overlap between their respective bounding boxes exceeds 50%.

In addition to bounding box overlap we also include another criterion for matching to occur. A tracked object matches a ground-truth object if the tracker was initialised by a target that has the same identity as the ground-truth object.

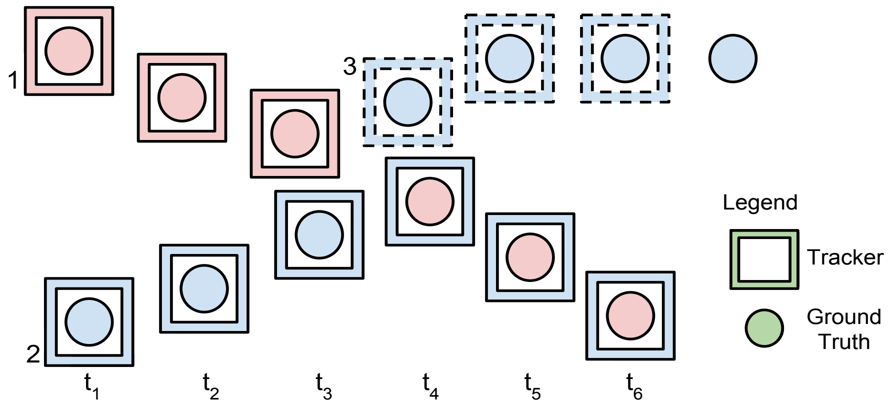

Fig. 4. A graphical example of the performance measures. The performance measures used are precision, recall and continuity/fragmentation of a trajectory. Tracker 1 (pink squares) is initialised to track the ground-truth target A (pink circles). Tracker 1 has 3 matches since it is only ever matched with target A and has a track length of 3. Tracker 2 (solid blue squares) only has 3 matches with a length of 6. This is because it is initialised to track the ground-truth target B (blue circles), however, it's last three detections coincide with ground-truth target A which is incorrect. Tracker 3 (dotted blue squares) is also initialised to track ground-truth target B and has 3 matches with a length of 3. The precision is then 9/12. Ground-truth target A has 3 matches with a length of 6. The last three ground-truth targets are unmatched due to tracker 2 being initialised to track target B. Ground-truth target B has 6 matches and a length of 7 since all of the corresponding trackers were initialised to track target B. The recall is 9/13. Target A is only covered by tracker 1 (tracker 2 is not included since it has not been initialised to track A) while target B is covered by trackers 2 and 3. The fragmentation is thus 3/2 and so the continuity is 2/3.

This is a reasonable condition to enforce since a tracker initialised by target A should not be tracking target B.

Now that single frame matching has been defined, performance across trajectories can be measured. If there are a total of N tracking trajectories and M ground truth trajectories then three quantities can be defined: precision, recall and continuity.

If $l^n_{tracker}$ is the length of tracking trajectory n and $d^n_{tracker}$ is the number of matches in trajectory n then the precision P across all tracking trajectories is defined as:

$$P = \frac{\sum_{n=1}^{N} d^n_{tracker}}{\sum_{n=1}^{N} l^n_{tracker}} \tag{5}$$

This measure is similar to the MOTP metric proposed by Bernardin and Stiefelhagen [10]. Precision gives a measure of how well a tracker initialised on target A tracks target A. If the tracker drifts or there is an identity swap precision will be low.

If l_{gt}^m is the length of ground truth trajectory m and d_{gt}^m is the number of matches in ground truth trajectory m then the recall R across all ground-truth trajectories is defined as:

$$R = \frac{\sum_{m=1}^{M} d_{gt}^m}{\sum_{m=1}^{M} l_{gt}^m} \tag{6}$$

Let f^m be the number of trackers that are required to cover ground truth trajectory m. The continuity C is then defined as:

$$C = \frac{\sum_{m=1}^{M} \mathbb{1}(f^m > 0)}{\sum_{m=1}^{M} f^m} \tag{7}$$

where $\mathbb{1}$ is the indicator function. Continuity is the inverse of the mean number of trackers needed to cover a ground truth trajectory. This is described as fragmentation in [32]. Trajectories that have no trackers covering them are not included in this calculation. The reason continuity is used instead of fragmentation is that continuity is easier to compare to precision and recall since a value of 1 for all three of these measures indicates perfect performance.

Recall and continuity give a measure of how well targets are tracked. If the tracker does not adapt to the appearance of the target, continuity will be low.

The overall performance of a tracking algorithm is given as the mean of the precision, recall and continuity. We feel that this is a reasonable metric to compare algorithms on since all three quantities are equally as important.

6 Experiments

For the remainder of this paper we will refer to our proposed tracking algorithm as the Category-to-Individual Tracker (CIT). To assess the performance of CIT we conduct experiments using the Buffy and Caltech Pedestrian datasets (refer to Sec 4). To the best of our knowledge, this is one of the largest performance evaluations for appearance-based tracking. The most important findings are reported here in this manuscript; the remainder is included as supplementary material.

The results of CIT are compared to 9 other, publicly available, appearance-based tracking algorithms. The source code for each of the algorithms was downloaded from the original author's website. Minor modifications were then made so that each tracker would operate in our multi-target, automatic initialisation framework (as opposed to single target; human initialised frameworks that these algorithms were originally tested on). If an algorithm had parameters to set then those that were recommended by the original authors were used.

For all of the experiments we use the multi-scale ACF detector of Dollar et al. [18,15] for the category detector. The code was downloaded from the website[1].

The category detector used for the Buffy dataset was trained using 882 faces from a separate dataset [12] which was also downloaded from the web. Due to

[1] `http://vision.ucsd.edu/~pdollar/toolbox/doc/`

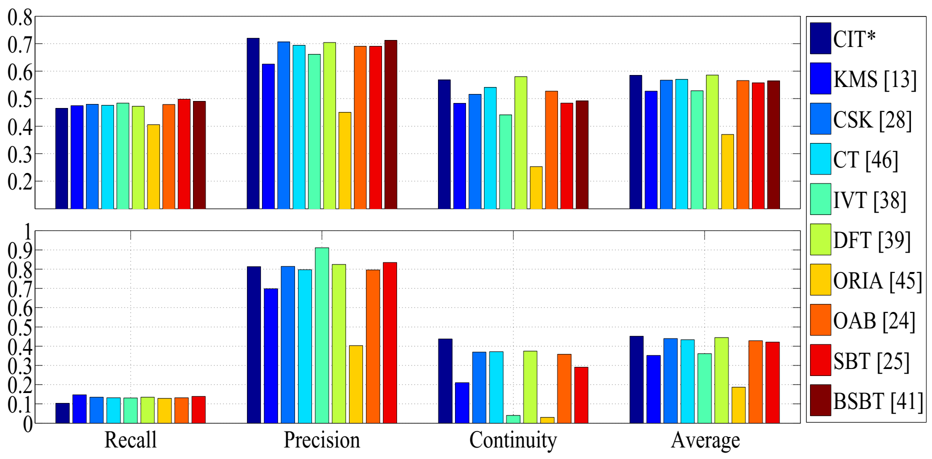

Fig. 5. Tracking Performance on Buffy (top) and Caltech Pedestrians (bottom) for a category detector operating at a recall of 0.5. Refer to Section 5 for definitions of recall, precision and continuity. The average performance is the mean of the precision, recall and continuity. The precision calculation ignores trajectories that have been initialised by false category detections. A tracker with perfect performance would have a value of 1 for each of the measurements. The results indicate that our method CIT is one of the best performers, however, most of the methods have similar performance results. The relative performance of the trackers is roughly equivalent across the two very different, very distinct datasets. The BSBT algorithm ran too slowly and so its performance on Caltech Pedestrians is omitted.

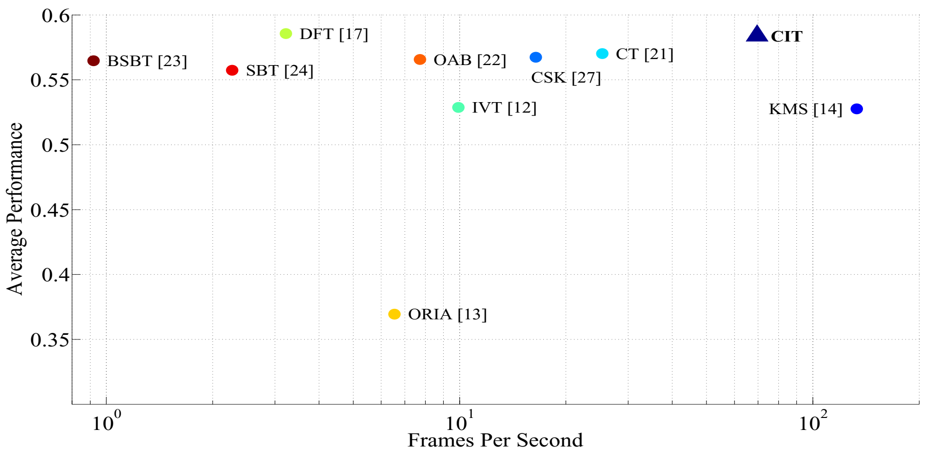

Fig. 6. The average performance and frame rate for each tracking algorithm using the Buffy dataset. The average performance is the same as the value in the top plot of Fig 5. The results indicate that our method, CIT is 10% more accurate and nearly as fast as the fastest of the competing algorithms. It is also as accurate but 20x faster than the most accurate of the competing algorithms.

the ground-truth for Buffy being noisy, an external dataset was used to train the face detector to ensure detection results were reasonable. The category detector used for the Caltech Pedestrians dataset was trained using the recommended training sets S0-S4 (refer to Sec.4).

Each category detector was then calibrated to operate at a specific recall by using a validation set. Episode 1 was used for Buffy and set S5 for Caltech Pedestrians. By operating the category detectors at different recall rates different targets are identified thus having an impact on the initialisation and update stages of tracking. We only include the results for a category detector recall of 0.5. Refer to the supplementary material for other values.

The validation sets were also used to calibrate each of the tracking methods. Each of the tracking methods have a confidence score associated with their predictions; in our regime, individuals enter and exit the scene so it is important to stop tracking a target when the confidence of the tracker is below a certain threshold otherwise the trackers will update their target model with background. To choose this threshold we evaluated each algorithm on the validation set for each dataset and selected the threshold that gave the best average performance. For the CIT tracker we selected the value of β (refer to Eqn. 3) during validation rather than the confidence threshold which was set to zero.

Each tracking algorithm was then evaluated on the test set for Buffy (episodes 2-5) and for Caltech Pedestrians (sets S6-S10) with trackers being initialised by non-coincident category detections (refer to Fig. 2). The resulting trajectories were used to compute precision, recall and continuity for each algorithm. The results for both datasets are in Figure 5 which includes the average performance which is the mean of the precision, recall and continuity. The precision calculation ignores trajectories that have been initialised by false category detections.

The speed at which each of these algorithms operate at is also important. To compute the average frame rate of a tracking algorithm we need to decouple the computation time due to the tracker and the computation time due to the category detector. The time it takes for the category detector to run without tracking is subtracted from the time it takes a tracker to run. The results are in Figure 6. We only include the results for Buffy since the results on Caltech Pedestrians (in the supplementary material) are similar. The average frame rates were computed using the first 10,000 frames of episode 3 of Buffy on an Intel i5, 3.20 GHz machine.

Qualitative examples of how our tracking method works on both Buffy and Caltech Pedestrians can be found in Figure 1. The sequences give an indication of how tracking of a target is successful despite occlusion and changes in pose.

7 Discussion and Conclusions

We have presented a novel tracking method which is designed to track objects belonging to a specific category. The method makes use of a category detector to identify target objects to track and of an individual-specific detector to track the target in subsequent video frames. The individual-specific detector is trained on-the-fly at almost no extra cost, making it possible for the tracker to operate in real-time. The well-known problem of drift is addressed by updating the individual-specific detector only when there are coincident category detections.

We compare the performance of our scheme to 9 state-of-the-art trackers and find that it is as accurate as the most accurate competitor, but 20x faster. It is only slightly slower than the fastest competitor, but 10% more accurate.

In order to carry out our benchmark comparison we developed a methodology based on considering four metrics: precision, recall, fragmentation and computational cost. Our experiments were carried out on two large (hundreds of thousands of detections), challenging and heterogeneous datasets of faces and pedestrians; we observe identical rankings of the various algorithms on the two datasets, which gives us the confidence that our findings are general and may be expected to carry over to a variety of datasets and tasks. We believe that our benchmark surpasses, both in method and set size, any such comparative evaluation in the literature.

Acknowledgements. This work is funded by the ARO-JPL NASA Stennis NAS7.03001 grant and the MURI ONR N00014-10-1-0933 grant.

References

1. Adam, A., Rivlin, E., Shimshoni, I.: Robust Fragments-Based Tracking Using the Integral Histogram. In: CVPR (2006)
2. Andriyenko, A., Schindler, K.: Globally Optimal Multi-target Tracking on a Hexagonal Lattice. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 466–479. Springer, Heidelberg (2010)
3. Avidan, S.: Ensemble Tracking. PAMI 29(2), 431–435 (2007)
4. Babenko, B., Belongie, S., Yang, M.H.: Visual Tracking with Online Multiple Instance Learning. In: CVPR (2009)
5. Bao, C., Wu, Y., Ling, H., Ji, H.: Real Time Robust L1 Tracker using Accelerated Proximal Gradient Approach. In: CVPR (2012)
6. Bauml, M., Tapaswi, M., Stiefelhagen, R.: Semi-Supervised Learning with Constraints for Person Identification in Multimedia Data. In: CVPR (2013)
7. Benenson, R., Mathias, M., Timofte, R., Van Gool, L.: Pedestrian Detection at 100 Frames per Second. In: CVPR (2013)
8. Berclaz, J., Fleuret, F., Fua, P.: Multiple Object Tracking using Flow Linear Programming. In: PETS (2009)
9. Berclaz, J., Türetken, E., Fleuret, F., Fua, P.: Multiple Object Tracking using K-Shortest Paths Optimization. PAMI 33(9), 1806–1819 (2011)
10. Bernardin, K., Stiefelhagen, R.: Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. EURASIP JIVP (1), 1–10 (2008)
11. Black, M.J., Jepson, A.D.: EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation. IJCV 26(1), 63–84 (1996)
12. Burgos-Artizzu, X., Hall, D., Perona, P., Dollár, P.: Merging Pose Estimates across Space and Time. In: BMVC (2013)
13. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-Based Object Tracking. PAMI 25(5), 564–577 (2003)

14. Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: CVPR (2005)
15. Dollár, P., Appel, R., Belongie, S., Perona, P.: Fast Feature Pyramids for Object Detection. PAMI (2014)
16. Dollár, P., Tu, Z., Perona, P., Belongie, S.: Integral Channel Features. In: BMVC (2009)
17. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian Detection: A Benchmark. In: CVPR
18. Dollár, P., Belongie, S., Perona, P.: The Fastest Pedestrian Detector in the West. In: BMVC (2010)
19. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian Detection: an Evaluation of the State of the Art. PAMI 34(4), 743–761 (2012)
20. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2009 (VOC) Results (2009)
21. Everingham, M.R., Sivic, J., Zisserman, A.: Hello! My Name is.... Buffy – Automatic Naming of Characters in TV Video. In: BMVC (2006)
22. Freund, Y., Schapire, R.E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
23. Fukunaga, K., Hostetler, L.: The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. IEEE Transactions on Information Theory 21 (1975)
24. Grabner, H., Grabner, M., Bischof, H.: Real-Time Tracking Via On-line Boosting. In: BMVC (2006)
25. Grabner, H., Leistner, C., Bischof, H.: Semi-supervised On-Line Boosting for Robust Tracking. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 234–247. Springer, Heidelberg (2008)
26. Gray, D., Tao, H.: Viewpoint Invariant Pedestrian Recognition with an Ensemble of Localized Features. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 262–275. Springer, Heidelberg (2008)
27. Hall, D., Perona, P.: From Categories to Individuals in Real Time - A Unified Boosting Approach. In: CVPR (2014)
28. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: Exploiting the Circulant Structure of Tracking-by-Detection with Kernels. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part IV. LNCS, vol. 7575, pp. 702–715. Springer, Heidelberg (2012)
29. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. Tech. Rep. 07-49, University of Massachusetts, Amherst (October 2007)
30. Jiang, H., Fels, S., Little, J.J.: A Linear Programming Approach for Multiple Object Tracking. In: CVPR (2007)
31. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-Learning-Detection. PAMI 6(1), 1–14 (2011)
32. Li, Y., Huang, C., Nevatia, R.: Learning to Associate: HybridBoosted Multi-Target Tracker for Crowded Scene. In: CVPR (2009)
33. Ma, Y., Yu, Q., Cohen, I.: Target Tracking with Incomplete Detection. Computer Vision and Image Understanding 113(4), 580–587 (2009)
34. Nevatia, R.: Global Data Association for Multi-Object Tracking using Network Flows. In: CVPR (2008)
35. Oron, S., Bar-Hillel, A., Levi, D., Avidan, S.: Locally Orderless Tracking. In: CVPR (2012)

36. Pellegrini, S., Ess, A., Van Gool, L.: Improving Data Association by Joint Modeling of Pedestrian Trajectories and Groupings. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 452–465. Springer, Heidelberg (2010)
37. Pirsiavash, H., Ramanan, D., Fowlkes, C.C.: Globally-Optimal Greedy Algorithms for Tracking a Variable Number of Objects. In: CVPR (2011)
38. Ross, D.A., Lim, J., Lin, R.S., Yang, M.H.: Incremental Learning for Robust Visual Tracking. IJCV 77, 125–141 (2007)
39. Sevilla-Lara, L., Learned-Miller, E.: Distribution Fields for Tracking. In: CVPR (2012)
40. Sivic, J., Everingham, M., Zisserman, A.: Who are You? - Learning Person Specific Classifiers from Video. In: CVPR (2009)
41. Stalder, S., Grabner, H., Van Gool, L.: Beyond Semi-Supervised Tracking: Tracking Should be as Simple as Detection, but not Simpler than Recognition. In: ICCV Workshops (2009)
42. Viola, P., Jones, M.: Rapid Object Detection Using a Boosted Cascade of Simple Features. In: CVPR (2001)
43. Viola, P., Jones, M.J.: Robust Real-Time Face Detection. IJCV 57(2), 137–154 (2004)
44. Wu, Y., Lim, J., Yang, M.H.: Online Object Tracking: A Benchmark. In: CVPR (2013)
45. Wu, Y., Shen, B.: Online Robust Image Alignment Via Iterative Convex Optimization. In: CVPR (2012)
46. Zhang, K., Zhang, L., Yang, M.-H.: Real-Time Compressive Tracking. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part III. LNCS, vol. 7574, pp. 864–877. Springer, Heidelberg (2012)