A Large Margin Framework for Single Camera Offline Tracking with Hybrid Cues

Bahman Yari Saeed Khanloo[a], Ferdinand Stefanus[a], Mani Ranjbar[a], Ze-Nian Li[a], Nicolas Saunier[b], Tarek Sayed[c], Greg Mori[a]

[a] *School of Computing Science, Simon Fraser University*
[b] *Dept. of Civil, Geological and Mining Engineering, École Polytechnique de Montréal*
[c] *Dept. of Civil Engineering, University of British Columbia*

Abstract

We introduce MMTrack (max-margin tracker), a single-target tracker that linearly combines constant and adaptive appearance features. We frame offline single-camera tracking as a structured output prediction task where the goal is to find a sequence of locations of the target given a video. Following recent advances in machine learning, we discriminatively learn tracker parameters by first generating suitable bad trajectories and then employing a margin criterion to learn how to distinguish among ground truth trajectories and all other possibilities. Our framework for tracking is general, and can be used with a variety of features. We demonstrate a system combining a variety of appearance features and a motion model, with the parameters of these features learned jointly in a coherent learning framework. Further, taking advantage of a reliable human detector, we present a natural way of extending our tracker to a robust detection and tracking system. We apply our framework to pedestrian tracking and experimentally demonstrate the effectiveness of our method on two real-world data sets, achieving results comparable to state-of-the-art tracking systems.

Keywords: tracking, trajectory optimization, structured prediction, conditional random fields, discriminative learning

1. Introduction

Superior tracking performance can be obtained by fusing different cues together. The main intuition behind this observation is that better performance is achieved because the combination of different cues help the whole tracking system overcome individual failure mode of each single cue [25, 26, 9, 30, 11, 18, 31, 5]. When combining a set of cues, a principled framework for choosing the parameters for the combination is desirable. In this paper, we present MMTrack (max-margin tracker), a single-target tracker that uses the max-margin learning framework [28] to combine cues.

As an example of a tracking system with multiple cues, consider the scenario depicted in Fig. 1. The leftmost image shows one input frame with the target object indicated by a red bounding box, and three feature maps obtained from different cues are shown next to the image. The cues are used to locate the target object, and they can be based on the target's colour histogram, object class, or motion pattern, among other information. These cues are used to build feature maps, where a pixel in a feature map indicates how likely the target is to be located at that pixel location. As can be seen in Fig. 1, pixels corresponding to the target location have relatively high values.

Given this information, our aim is to formulate a principled framework to combine the cues by determining the relative importance of each cue. Note that we do not restrict our cues to a certain feature class - we only require the cues to produce a mapping from pixel locations to numeric values indicating the possibility of the target being located at some pixel location. Indeed, the cues can be any combination of appearance features, motion features, or even results of other simple trackers.

Combining the cues in this case is complicated by the fact that the cues can be a combination of different and unrelated features. For example, it is not easy to relate a cue built from the target's colour histogram to another built from the target's dynamics, as they are based on two seemingly independent models, namely the appearance model and the motion model. Further, in determining the relative contributions of the cues, ideally we should consider all the cues jointly rather than independently, as one cue's contribution should be considered with regard to all other cues.

One approach for combining different sources of information is to examine each cue separately, and weigh each cue according to a reliability score that measures how well the cue performs according to some predefined criteria. The reliability measure can be computed online, for example based on the distance between the cue's current feature map response to its average response [25], or it can be trained offline by computing an error measurement between the cue's trajectory result to some pre-labelled ground truth [26]. Combining the reliability measures of different cues in this setting is problematic because each measure is computed independently and thus they are not directly comparable across different cues.

The cues can be fused within a probabilistic frame-

work [30, 11, 18, 12, 17]. The simplest probabilistic method for fusing the cues assumes conditional independence between each cue's observation model, thus allowing the full joint observation model to be decomposed as a product of each cue's observation model [12, 17]. More sophisticated methods model the dependencies between appearance cues by decomposing the graphical models [30, 11], or by assuming sequential dependency between the appearance features [18]. Inference in these models is usually approximated with iterative sampling procedures. In contrast with this line of work, we use a discriminative margin-based learning criterion that aims for low tracker error.

However, developing vision algorithms to track objects such as pedestrians in realistic scenarios turns out to be a non-trivial task. A robust pedestrian tracking algorithm should be able to handle changes in pedestrians' appearance caused by human articulation or change in illumination. This suggests that the model should be updated to reflect changes in the pedestrian's pose. On the other hand, by continually adapting, there is a possibility that a small error in the pedestrian's hypothesized location might cause incorrect information to be absorbed by the appearance model.

Over time, the errors may accumulate, and the object model may not accurately reflect the target pedestrian's appearance anymore, causing the tracker to drift to another object. Further, because there can also be multiple pedestrians in the view at a time, the object tracking algorithm should also be able to differentiate between different instances of the pedestrians. This task is especially difficult if the pedestrians are similar in appearance. The pedestrians may also interact, introducing issues such as partial or full occlusion. Occlusions can also occur due to interaction between the target and background objects, for example when a tracked pedestrian walks behind a traffic sign. Additionally, the tracker should also be able to handle changes in scale caused by the pedestrian's relative distance to the camera. A fully automatic pedestrian tracking algorithm should also be able to automatically initialize a new track when a pedestrian enters the scene, and to terminate an existing track when a pedestrian exits the scene.

In general, tracking algorithms predict a target's location by modeling two of its characteristics: its appearance and its motion. The target's appearance is represented by an appearance model, which usually describes the target's shape or its distinctive features such as colour or texture. Some common representation of a target's shape are its silhouette [32] or simple shapes such as ellipses [7, 4] or rectangles [6, 2]. Appearance features that can be used to describe a target include colour histograms [7, 4], texture [24, 2], and edges [22].

Whereas the appearance model describes what a target looks like, the motion model, on the other hand, encodes prior knowledge or assumptions about the target's movement patterns. A motion model serves to restrict the range of possible target's movement, and is useful because a target's positional state usually does not change abruptly between consecutive frames. An example of a motion model is the Brownian motion model adopted by Ross et al. [23], which models the targets dynamics as Gaussians centered on its previous state. Babenko et al. [2] use a simpler motion model that assumes the target to be equally likely to appear at any location within a certain radius from its previous location. Another example of a motion model is the constant heading model used in [1] that assumes the target does not change its direction between a pair of consecutive frames.

Most object tracking algorithms treat appearance and motion models independently. The two models are usually integrated by using the motion model to guide the search for the location that provides the best match to the appearance model [2, 6, 7, 4].

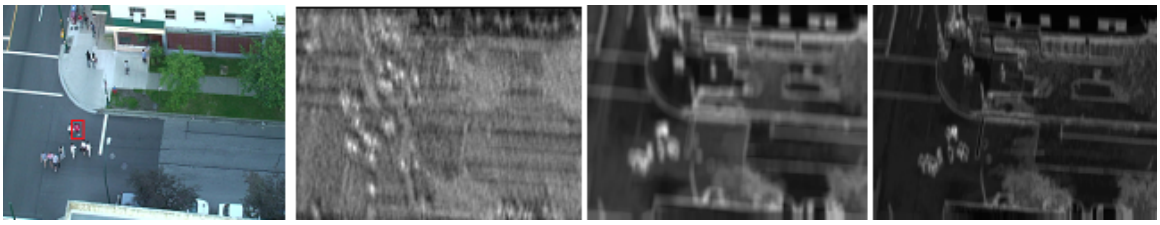
Switching methods can also be used to select the best cues out of a fixed pool of features at every frame. A common formulation of this approach estimates the foreground-background discriminability of the cues, and selects the most discriminative cues either by quantifying their discriminative power with a Fisher-like criterion [6], or through an online boosting mechanism [2]. It should be noted that the feature pools in both works are composed of a single feature class, namely linear combination of RGB channels [6] and Haar-like features [2].

In this paper we describe a method for building an automated detection and tracking system from multiple cues. The method uses max-margin learning for structured output to learn weights that combine features for tracking. This learning can be used with a variety of features, and unlike previous work jointly learns appearance and motion models in a unified framework. An initial version of this work described the MMTrack approach [15]. In this paper we include additional experiments and a fully automatic human detection and tracking system built on top of MMTrack.

The rest of this paper is organized as follows. In Section 2, we explain our choice of features that provide us with cues for tracking. In Section 3, we intorduce our tracking framework called Max-Margin Track (MMTrack) in the context of pedestrian tracking. Section 4 describes the details of our margin-based parameter estimation. In Section 5, we show how MMTrack can be extended to build an automatic detection and tracking system. Section 6 presents our qualitative and quantitative experimental results and Section 7 concludes the paper and discusses further work.

2. Features for Tracking

The main focus of this paper is on a principled combination of multiple cues for offline single target tracking. In order to ground the discussion of the learning framework (Sec. 3) we first present the set of cues we use in our system. However, the framework is general, and can be used



(a) (b) (c) (d)

Figure 1: A tracking scenario with multiple cues. (a) Input frame. (b) Person detector, (c) colour template, and (d) adaptive colour model cues for tracking. We learn the parameters to a model that combines multiple cues for tracking.

with a variety of cues. For our system we choose a representative set of cues that could be used in an offline tracker. These cues include an object detector (HOG pedestrian detector), an offline colour model (clustered colour histograms), and online appearance models (templates). In this section we motivate and describe this particular choice of features. We note that this combination is effective in practice, with experiments demonstrating that it can obtain high quality tracks (Sec. 6).

2.1. Object Detector: HOG Score

The first cue we consider contains knowledge of the object of interest, in this case pedestrians. We use information from a reliable detector that can assist in localizing a pedestrian that is being tracked. In particular, we use the Histogram of Oriented Gradients (HOG) [8] trained for human detection as we are interested in pedestrians in this work.

We expect the HOG feature to be of help to the system in discriminating between pedestrians and non-pedestrian objects (e.g. car, tree, etc.). A generic object detector such as this one can potentially reduce tracker drift, focusing the tracker on the known class of object. However, the generic detector is not tuned to a particular person, and should be used in concert with a variety of other person-specific cues. In our framework, we will learn how much to rely on each individual cue, in this case deciding on the relative importance of a generic object detector versus other tracking cues.

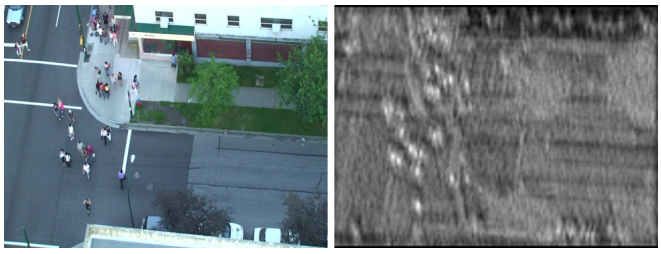
We use the output of a linear SVM classifier that operates on HOG [8] as a feature to help our system differentiate between pedestrians and other objects. The detection is performed on a pyramid built on the input image with varying scale. For each pixel, we take the maximum SVM score over all scales resulting in a score map where the peaks vote for presence of pedestrians. We then normalize these scores so they fall within the range [0,1] and use the final map as our feature. Fig. 2 illustrates an input image along with the corresponding normalized HOG score map.

2.2. Offline Colour Model: Clustered Histograms

Although HOG scores can help the system differentiate between pedestrians and other objects, they are not informative in distinguishing among different pedestrians, as many pedestrians will potentially have high HOG scores. Thus, features that convey identity, i.e. features that try to uniquely respresent the appearance of a pedestrian, are needed. We incorporate such features using a static colour histogram for each person obtained from clustering. The main idea here is that by clustering the histograms obtained from bounding boxes around the pedestrians throughout the video, we can gain a good insight into the average appearance statistics of each of the people. Thus, we will be able to model the changes instead of trying to learn the appearance and so we would be able to obtain a simple yet effective appearance model. Note that these appearance features in our tracker contribute towards gaining resistance against drift that often occurs in tracking systems which only consider dynamically-updated appearance models.

This feature is motivated by the work of Ramanan et al. [21], who learn appearance models of articulated objects (animals, people) based on detection, and then use them for tracking. Searching for targets and learning their appearance as done in [21] is not practical in our problem because we neither can assume a constant appearance nor can we get a reliable segmentation or pose like in [27] and [19] as pedestrians are far away from the camera. Instead, we build rough descriptors and try to discriminate the object of interest from surroundings. We use trackers built on top of hierarchical colour histograms that describe how the histograms of different parts of the bounding box enclosing the pedestrian deviate from their mean over time.

The generation of the colour histogram distance features



(a) (b)

Figure 2: A frame and its corresponding HOG feature map. HOG pedestrian detection responses are used as a feature in the combined tracking model.

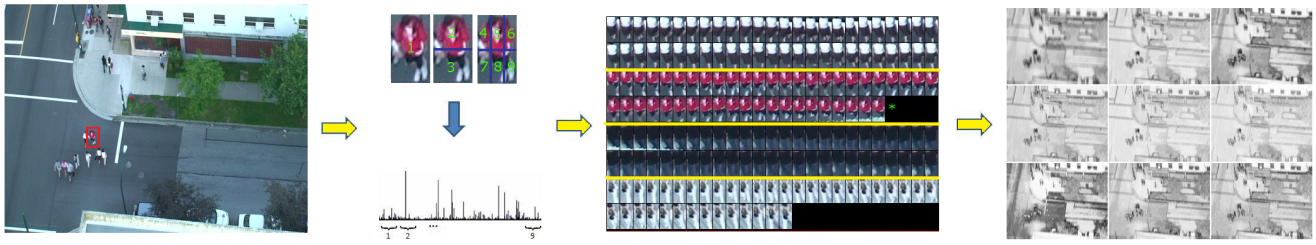


Figure 3: Colour histogram distance features generation: **a)** Nine histograms are computed over sections of the detected person's bounding box and the resulting histograms are concatenated to give a hierarchical description. **b)** all sampled histograms are clustered. **c)** Histogram distance maps are then generated for every frame by sliding a window and computing the χ^2-distance between the histogram of each of the sections of the window to the cluster mean to which the target belongs.

is as follows. We start by running a HOG person detector over the entire video, and uniformly sampling a set of detections. Based on the image evidence inside the bounding box obtained from a HOG detection, nine histograms are computed over different sections of a pedestrian's body as depicted in the second column of Fig. 3. Each histogram consists of 30 bins with 10 bins for each of the R, G and B channels. These nine histograms are then concatenated together to give one histogram characterizing the person's appearance. Next, we cluster the sampled instances of histograms of all the people in the video using the mean-shift clustering algorithm. We then represent the target pedestrian using the mean of the cluster to which it belongs. This is done using a simple search that measures the distance between the initial appearance and cluster centers since we assume that the initial location of the pedestrian is given. Finally, we compute one histogram distance map for each of the nine body sections by computing, at each pixel location, the χ^2-distance between the histogram built using the image observation within the corresponding section of the bounding box centered at that pixel and the mean of the cluster to which the target belongs. The resulting maps have low values in areas with similar colour to the target person's and high values elsewhere. We efficiently compute the histogram distance maps using the integral histogram technique [20].

In summary, these colour histogram distance features enforce the similarity of a tracked person to a global fixed, offline colour model. They measure the similarity of a target location to a colour model obtained by mapping an initial pedestrian location to a cluster. These features can be used to combat drift, and weights on these features will be learned in our algorithm. However, these features do not adapt to changes in target appearance, which will be handled by the final feature set.

2.3. Online Appearance Model: Templates

We use an online feature that models the target's appearance. This appearance template feature will give clues about a particular pedestrian at a finer level than the histogram distance features. We use two templates: a template obtained from the image patch inside the bounding box surrounding the given location of the target in the initial frame of the trajectory, and another template obtained from the previous frame of the trajectory.

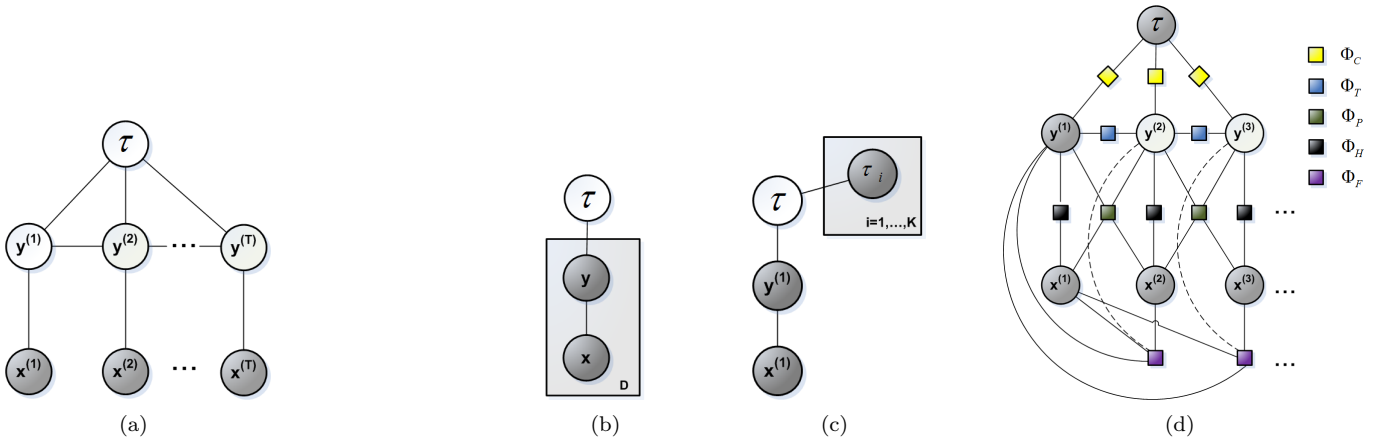
The initial template implements a constant appearance model which is used to provide a fixed reference to the person's appearance, similar to the cluster center of the colour histograms. However, the initial appearance template describes the person's appearance at a finer level of detail than the histogram distance features. This template acts as a memory template, which is stable by definition and ensures that the tracker does not completely forget about the appearance of the target when it first showed up.

On the other hand, the previous frame template incorporates the idea of adaptive appearance modeling because it mimics the appearance adaptation mechanism by encoding the expected amount of frame-to-frame change of the tamplate during the inference, which helps the system cope with some degree of object appearance changes over time.

Distance maps are computed for each template by sliding a window over the current frame and computing the sum of absolute pixel value differences in all three colour channels of each pixel belonging to that template, which is efficiently performed using a modified integral image technique [29]. These maps are normalized so the values fall in the range [0,1].

3. MMTrack: Learning to Combine Cues for Single Target Tracking

In this section, we explain the details of our tracking algorithm in the context of pedestrian tracking. Our tracker is comprised of three main components: constant appearance model, adaptive appearance model, and motion model. The constant appearance model is used to memorize the appearance of the target pedestrian in two different detail levels whereas the adaptive model is used to model the change in appearance. Finally, the motion model favors specific movement patterns from one frame to another. These components will be used to describe the



(a) (b) (c) (d)

Figure 4: **a)** Generic tracking approach where we build an appearance template τ for the target object while tracking. **b,c,d)** Our method: clustering and a simple search followed by inference in a tree-structured CRF with shared parameters where the initial position, colour (appearance) template and all the inputs given a priori. The plate notation refers to replicates of the same structure. The rectangles stand for the factors of the model. We are showing only three frames of our temporal model for simplicity of presentation.

object of interest, and the margin-based learning described in Section 4 will be used to combine them.

The rest of this section is organized as follows. Section 3.1 provides details of our model for describing trajectories. Section 3.2 outlines our trajectory representation and section 3.3 explains our inference scheme for tracking.

3.1. Trajectory Modeling

As noted earlier, we are interested in *offline* tracking where the goal is to obtain the whole trajectory in the entire sequence at once. This is in contrast to *online* tracking algorithms that greedily pick the next best location of the object at each frame. Thus, the tracking problem in this setting is formulated as one of finding the optimal trajectory $\mathbf{y} = (\mathbf{y}^{(1)}, ..., \mathbf{y}^{(T)})$ with the image sequence $\mathbf{x} = (\mathbf{x}^{(1)}, ..., \mathbf{x}^{(T)})$ given. This general setting, illustrated graphically in Fig. 4(a), requires us to build a template τ to model the appearance of the object while trying to find the best trajectory. We build a model for only a single target, and do not explicitly consider *joint* tracking of multiple targets in our model. Further, if there are multiple instances of the object of interest we identify and track them one by one. For now, we assume that the initial location of the target is provided in the first frame of the sequence. This assumption will be relaxed in Sec. 5, when we describe a fully automatic detection and tracking method.

We further assume that the appearance of the object in the initial frame is representative and reliable. Also, we further assume that an upper bound for the length T of the track is given. Moreover, in general τ is a high dimensional continuous variable and learning and inference for this loopy graph is intractable.

Instead of reasoning over the entire space of appearance models for τ, we use the clustering and discretization procedure for colour models described in Sec. 2.2. The procedure is as follows: we use the detector to find D detections (all instances) of objects in the video segment of interest and group them into K clusters. Note that variable τ is now discretized to K distinct vectors i.e. cluster centers each of which ideally represents an individual object. Given that the initial location is reliable, we find for each object the cluster $\hat{\tau} \in \tau' = \{\tau_1, ..., \tau_K\}$ which is the closest to the appearance template built in the first frame for that object. With the average appearance template $\hat{\tau}$ given, we end up with a tree-structured model for which learning and inference is practical. These simple steps, when done in sequence, aim at approximating the original problem. The steps are shown in Fig. 4(b), 4(c) and 4(d). Note that step (b) is performed once whereas steps (c) and (d) are repeated independently for each object.

Our *scoring function*, which measures the goodness of trajectories, is a mapping in the form of $F(\mathbf{x}, \mathbf{y}; \mathbf{w}) : \mathcal{X}^T \times \mathcal{Y}^T \rightarrow \mathcal{R}$ that maps a sequence of frames $\mathbf{x} = (\mathbf{x^{(1)}}, ..., \mathbf{x^{(T)}})$ and a trajectory $\mathbf{y} = (\mathbf{y^{(1)}}, ..., \mathbf{y^{(T)}})$ to a real number. Each location $\mathbf{y^{(t)}}$ is a discrete variable which is to be assigned to one of the image pixels and $\mathbf{w}$ is a set of weights that parameterize the features extracted from the frames. The scoring function is decomposed into two contributions: *transition model* and *observation model*. The transition model in our problem is summarized by the *motion model* which describes the spatial relationship between the locations of the target in two consecutive frames. The observation model is a measure of compatibility between a location and the observed features at that pixel location. We define the score of a trajectory as

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}, \tau) = \sum_{t=2}^{T} F_{\mathcal{T}}(\mathbf{y}^{(t-1)}, \mathbf{y}^{(t)}; \mathbf{w}_{\mathcal{T}}) + \sum_{t=2}^{T} F_{\mathcal{O}}(\mathbf{x}^{(1)}, \mathbf{x}^{(t-1)}, \mathbf{x}^{(t)}, \mathbf{y}^{(1)}, \mathbf{y}^{(t-1)}, \mathbf{y}^{(t)}; \mathbf{w}_{\mathcal{O}}, \tau) \quad (1)$$

where $F_{\mathcal{T}}(\cdot)$ and $F_{\mathcal{O}}(\cdot)$ are linear models describing transition and observation contributions respectively. These potential functions are parameterized by $\mathbf{w}_{\mathcal{T}}$ and $\mathbf{w}_{\mathcal{O}}$ whose concatenation we denote by $\mathbf{w}$.

3.1.1. Observation Model

The observation model includes several features whose weighted combination votes for the presence of the target pedestrian. These features include HOG score that helps with discriminating between humans and other objects, and colour histogram distance and appearance templates that describe how the pedestrian looks like and how its appearance varies over time. Thus, the observation model at time t decomposes into the following contributions

$$\begin{aligned} F_{\mathcal{O}}(\cdot;\mathbf{w}_{\mathcal{O}},\tau) &= {\mathbf{w}_{\mathcal{H}}}^T\Phi_{\mathcal{H}}(\mathbf{x}^{(t)},\mathbf{y}^{(t)}) \\ &+ {\mathbf{w}_{\mathcal{C}}}^T\Phi_{\mathcal{C}}(\mathbf{x}^{(t)},\mathbf{y}^{(t)},\tau) \\ &+ {\mathbf{w}_{\mathcal{P}}}^T\Phi_{\mathcal{P}}(\mathbf{x}^{(t-1)},\mathbf{x}^{(t)},\mathbf{y}^{(t-1)},\mathbf{y}^{(t)}) \\ &+ {\mathbf{w}_{\mathcal{F}}}^T\Phi_{\mathcal{F}}(\mathbf{x}^{(1)},\mathbf{x}^{(t)},\mathbf{y}^{(1)},\mathbf{y}^{(t)}) \end{aligned} \quad (2)$$

where we have defined

$$\Phi_{\mathcal{C}}(\mathbf{x}^{(t)},\mathbf{y}^{(t)},\tau) = d_{\chi^2}(\Psi_{\mathcal{C}}(\mathbf{x}^{(t)},\mathbf{y}^{(t)}),\tau), \quad (3)$$

and $\Phi_{\mathcal{H}}(\cdot)$, $\Phi_{\mathcal{C}}(\cdot)$, $\Phi_{\mathcal{P}}(\cdot)$ and $\Phi_{\mathcal{F}}(\cdot)$ denote the joint feature representation functions that return HOG score, χ^2 distance between different parts of the colour histogram $\Psi_{\mathcal{C}}$ and their corresponding part of mean appearance τ, and the difference between appearance templates of the previous frame and the first frame to the current frame respectively. We concatenate all the observation weights to give $\mathbf{w}_{\mathcal{O}} = [\mathbf{w}_{\mathcal{C}};\mathbf{w}_{\mathcal{H}};\mathbf{w}_{\mathcal{F}};\mathbf{w}_{\mathcal{P}}]^T$. Intuitively, $\mathbf{w}_{\mathcal{O}}$ weighs the observation features i.e. the trackers at each pixel to give a map that ideally peaks at the body center of the target pedestrian.

3.1.2. Transition Model

Similar to the observation model, we define the transition model as

$$F_{\mathcal{T}}(\mathbf{y}^{(t-1)},\mathbf{y}^{(t)};\mathbf{w}_{\mathcal{T}}) = {\mathbf{w}_{\mathcal{T}}}^T\Phi_{\mathcal{T}}(\mathbf{y}^{(t-1)},\mathbf{y}^{(t)}), \quad (4)$$

where $F_{\mathcal{T}}(\cdot)$ is a symmetric motion model. The motion model discretizes the distance travelled between two consecutive frames into a number of bins that represent concentric circles centered at the previous location. So, we have

$$\Phi_{\mathcal{T}}(\mathbf{y}^{(t-1)},\mathbf{y}^{(t)}) = \text{bin}(||\mathbf{y}^{(t-1)} - \mathbf{y}^{(t)}||_2), \quad (5)$$

$$\text{bin}_k(d') = \mathbb{1}_{[\lfloor d' \rfloor = k]}, \quad k = 0, ..., \lfloor d_{\max} \rfloor. \quad (6)$$

in which we bin the Euclidean distance between the $2d$ image locations of y and y', $\mathbb{1}_{[.]}$ is the indicator function and $\text{bin}(\cdot)$ acts as a selection operator that generates a vector of length $d_{max}+1$ with all the elements set to 0 except one being 1. The upper bound d_{max} on the travelled distance from one frame to the next one is estimated from the data. Note that the symmetric motion model results in $\mathbf{w}_{\mathcal{T}}$ being a disk-like, constant position motion prior which is learned jointly with all other parameters.

3.2. Trajectory Representation

As noted earlier, we require a combined feature representation in order to build our scoring function. We encode a trajectory-video pair $(\mathbf{x},\mathbf{y})$ using a function $\Phi(\cdot)$, whose components we introduced previously, that compactly represents their statistics. Recall that this representation is decomposed in the same way that the model parameter vector does, namely $\Phi = [\Phi_{\mathcal{H}};\Phi_{\mathcal{C}};\Phi_{\mathcal{P}};\Phi_{\mathcal{F}};\Phi_{\mathcal{T}}]$.

This final combination of all cues, aggregated over all frames of a trajectory, form the representation $\Phi(\cdot)$ for a trajectory. The weights $\mathbf{w}$ needed to score a trajectory will be set using the learning procedure. We next describe how to perform inference in this model – which is necessary to apply it to tracking as well as the aforementioned learning.

3.3. Tracking as Inference

In our setting, tracking amounts to performing inference in our conditional temporal model given the video and parameters $\mathbf{w}$, i.e. finding the highest scoring trajectory. Note that if we were to solve the general problem shown in Fig. 4(a), we would need to find

$$\hat{\mathbf{y}} = \underset{\mathbf{y},\tau}{\text{argmax}}\ F'(\mathbf{x},\mathbf{y},\tau;\mathbf{w}) \quad (7)$$

where $F'(\cdot)$ would be a scoring function that requires describing the appearance of a particular object and finding the optimal trajectory simultaneously. However, we do not do so. Instead, we perform inference using the sub-models in Fig. 4(c) and Fig. 4(d) for each object:

$$\hat{\tau} = \underset{\tau\in\tau'}{\text{argmin}}\ ||\Psi_{\mathcal{C}}(\mathbf{x}^{(1)},\mathbf{y}^{(1)}) - \tau||_2^2, \quad (8)$$

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\text{argmax}}\ F(\mathbf{x},\mathbf{y};\mathbf{w},\hat{\tau}). \quad (9)$$

Obviously, exhaustive search for the optimal trajectory is not reasonable. We efficiently solve this problem using a modified version of the Viterbi algorithm which is given by the following dynamic program

$$\begin{aligned} M^{(t)}_{(l_C)} = \max_{l_N}\Big(M^{(t-1)}_{(l_N)} + F_{\mathcal{T}}(\hat{\mathbf{y}}^{(t-1)} = l_N, \hat{\mathbf{y}}^{(t)} = l_C)\Big) \\ + F_{\mathcal{O}}(.,\hat{\mathbf{y}}^{(t)} = l_C), \quad t = 2,...,T, \quad l_N \in \mathcal{N}(l_C), \\ M^{(1)}_{(l_{\text{init}})} = 0, \quad \forall l \neq l_{\text{init}}\ , M^{(1)}_{(l)} = -\infty. \end{aligned} \quad (10)$$

In fact, back to our CRF model depicted in Fig. 4(d), we are interested in maximizing the conditional

$$p(\mathbf{y}|\mathbf{x};\mathbf{w},\tau) \propto \exp(F(\mathbf{x},\mathbf{y};\mathbf{w},\tau)) \quad (11)$$

namely finding the maximum of the log-posterior of the paths (i.e. terminating pixels marginalized over time) given the parameters with the prior for initial location $\mathbf{y}^{(1)}$ set to 1 and all other pixels set to 0. Note that we are using a *binding* prior that sets to zero the posterior over trajectories that do not start from the initial location. Each element $M^{(t)}_{(p)}$ corresponds to a pixel and indicates the score of

the highest scoring trajectory that originates at the initial location l_{init} and terminates at pixel p at time t. A traceback from the final most scoring location is done to recover the track. In our notation, l_C and l_N refer to the current hypothesized location and its neighboring location(s) respectively. We just search the neighborhood $\mathcal{N}(l_C)$ when trying to find the next possible location instead of doing a full search. Note that this local search is valid since it complies with the nature of the movements of humans as a pedestrian is not expected to jump to a pixel which is far away from the current location. Namely, we are finding an exact solution in the space of "valid" trajectories.

As we will point out later, we need to run our tracker in the original resolution since all the trackers to which we will be comparing our system are doing the same. However, performing inference in high resolutions turns out to be computationally prohibitive even with local search and integral histogram optimizations. Thus, we resort to approximate inference. So, we perform beam search and only consider the H top-scoring hypotheses and discard the rest. This allows us to produce the tracking results in the original resolution while keeping the inference feasible. Obviously, beam search will return suboptimal results because it does not explore the whole hypothesis space. However, experimental results show that our approximate inference scheme works well in practice.

4. Margin Criterion For Learning

The learning task is to find a set of parameters that can *discriminate* between a compatible video-trajectory pair and all other trajectories. Learning the model parameters in this problem setting is challenging since we do not have *negative examples*. In other words, we do not know how a "bad" trajectory looks like and more importantly, how it differs from a "good" one as this information is not included in the dataset. Moreover, as considering all possible "bad" trajectories is intractable – the number of these grows exponentially with the length of a track.

However, recent advances in structured output prediction [28] provide a principled method for choosing parameters in this setting. We can find parameters $\mathbf{w}$ that maximize the score of N given ground truth tracks while pushing down the score of potential runner-up negative examples, modulated by a measure of how "bad" the negative examples are:

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||_2^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i, \quad s.t. \quad \forall i = 1,...,N, \xi_i \geq 0, \quad (12)$$

$$\bar{F}(\mathbf{x}_i,\mathbf{y}_i) - \bar{F}(\mathbf{x}_i,\mathbf{y}) \geq \bar{\Delta}(\mathbf{y}_i,\mathbf{y}) - \xi_i. \quad \forall \mathbf{y} \in \mathcal{Y} \quad (13)$$

The constant $C > 0$ specifies the relative importance of margin maximization and error minimization which is determined by cross validation. Note that we are considering the *margin rescaling* formulation [28], i.e. requiring the score of ground truth $\mathbf{y}_i$ to be at least as far away from the score of a possibly incorrect trajectory $\mathbf{y}$ as the loss $\Delta(\mathbf{y}_i,\mathbf{y})$ incurred when predicting $\mathbf{y}$. The averaging is performed to make examples with different lengths comparable since in an unnormalized representation, the location of joint representations of both positive and negative examples with respect to the hyperplane(s) and hence the shape and location of the feasible region would also depend upon the length of the sequence.

A common loss function would measure the total squared Euclidean distance between corresponding locations in two trajectories:

$$\Delta(\mathbf{y}_i,\mathbf{y}) \quad = \quad \sum_{t=1}^{T} ||\mathbf{y}_i^{(t)} - \mathbf{y}^{(t)}||_2^2. \quad (14)$$

In tracking, a target is often considered to be "lost" if the tracker is off by more than a predefined number of pixels ρ. So, we also define a *bounded loss* function which is again additive and is expressed as:

$$\Delta_B(\mathbf{y}_i,\mathbf{y}) \quad = \quad \sum_{t=1}^{T} \min(\rho^2, ||\mathbf{y}_i^{(t)} - \mathbf{y}^{(t)}||_2^2) \quad (15)$$

As discussed earlier, we need to generate negative examples so we can try and solve for them only to make the optimization tractable. So, for each training pair example $(\mathbf{x}_i,\mathbf{y}_i)$ we seek to find

$$\hat{\mathbf{y}} = \underset{\mathbf{y}\neq\mathbf{y}_i}{\text{argmax}} \Big(F(\mathbf{x}_i,\mathbf{y}) + \Delta(\mathbf{y}_i,\mathbf{y})\Big). \quad (16)$$

This is an iterative procedure: we solve for $\mathbf{w}$ then we find $\hat{\mathbf{y}}$ (for all examples) given $\mathbf{w}$ and, having maintained a small nonredundant set of negative examples, repeat until a desired stopping criterion is met. We use the SVM^{struct} framework [28] to solve this problem.

We can imagine two main types of negative examples: 1) Tracks that start from initial location and *drift* to *background* i.e. anything other than an instance of object category of interest. 2) Tracks that start from initial location but get *hijack*ed by *distractor*(s) i.e. other instances from the same object category that are similar and close enough. Further, any mistrack would be either of the above or a combination of them.

Let us now consider the rationale behind our bounded loss function and the semantics of negative examples that we generate using Eq. 15 and 16. We consider the tracker to be lost if, at any time point, it is off by at least ρ pixels. But as long as it is lost, our loss term turns into a constant and all such trajectories are equally invalid (as far as loss is concerned) since they are false tracks anyway and we will just keep track of the amount of time they were lost. Therefore, it will be up to the scoring function to determine bad examples and this amounts to finding the most *confusing* trajectory among all wrong ones according to the current model parameters $\mathbf{w}$. In contrast to unbounded loss, we believe that a bad trajectory is useful not for being as far as possible but for being close to

(a) First iteration: The worst track possible i.e. maximizing the loss only.

(b) Drifting to background.

(c) Tracking a distractor.

(d) Convergence: tracking the right person.

Figure 5: Iterations in optimization. Green is the ground truth and red is the "worst" negative example.

the ground truth yet being wrong, i.e. belonging to background or distractor(s). Note that, a mistrack will necessarily spend some time in the background and for the case of drift this is going to be considerable which makes them less challenging. Such examples will, after a few iterations, become easy (especially if we use detector features such as HOG) and hence distractors are the most informative negative examples in this setting. We call a negative example *informative* if it contributes to discriminative power of our tracker by identifying an important and representative failure mode. Features such as transition features might be similar in both classes (e.g. distractors in this case) which would then help find suitable weights for them relative to other features. Note, however, that considering our assumption about non-overlapping objects, a distractor will spend some time in background and therefore is expected to have sufficiently distinguishable features.

Some notable stages of the optimization procedure using unbounded loss and a suitable constant C are illustrated in Fig. 5. Green indicates ground truth trajectory and red is the negative example. We can observe that, among the generated trajectories for this particular training sample, some examples can be thought of as being important failure modes. For instance, Fig. 5(b) shows a trajectory that corresponds to drift. Also, Fig. 5(c) shows a distractor negative example where the tracker is learning to avoid tracking other people whose appearance is similar to the target pedestrian. Note that examples generated using bounded loss tend to make more sense, i.e. drifting to nearby similar objects instead of going to the farthest corner as in Fig. 5(b). Obviously, the quality of the features corresponding to a trajectory does not necessarily degrade if it is shifted away from the ground truth.

5. Automatic Detection and Tracking with MMTrack

We have described MMTrack for a single target with known initial location. For most applications, simultaneous detection and tracking is of vital importance. In this section we relax the assumption of known initial target location, and show how to use MMTrack in this setting. We demonstrate that this system can be used to automatically detect and track pedestrians.

In order to extend MMTrack to a fully automatic detection and tracking system, we have to add capability to detect entering and exiting pedestrians to MMTrack. We use the HOG pedestrian detector to automatically initialize our tracker, running it over the entire video sequence. Because HOG is also used as one of our appearance features, using it to initialize MMTrack inference has an additional benefit of not incurring additional computational cost. Since a detection can occur at any arbitrary point in the pedestrian's trajectory, searching only forward in time is unreliable. Therefore, we run MMTrack forward and backward in time from these detections. Finally, since

the length of the trajectory is unknown, we need to decide when to stop the tracker and choose an upper bound based on the expected length of stay (using average speed) such that the computations are practical. We then cluster these trajectories to find person tracks.

Using a detector to initialize tracks is a common strategy, for example detectors used to initialize object tracking algorithms include hand detector [16], foreground object detector in the form of background subtraction [3], and pupil detection [10]. Note that we are dealing with a variable number of objects because we can not assume that a group of people enter and exit at the same time. Further, we cannot necessarily define any specific entrance point (e.g. frame borders), since state-of-the-art detectors may miss pedestrians.

For computational reasons and track termination, we resort to a chunking procedure to process the entire video. The procedure can be summarized as follows (see Fig. 6). We divide the whole footage into some fixed length blocks and use HOG pedestrian detector to locate the people in three consecutive blocks. We initialize one tracker per detection every L frames (we used $L = 100$) and perform tracking forward and backward in time and merge the two so we get an approximation of full trajectory optimization for all pedestrians in the *middle* block. The trajectory is terminated once all H hypotheses for the target exit a predefined region of interest (ROI) in the image. If, once out of the ROI, the target returns we will introduce a new trajectory for him. Via this process, we get potentially many trajectories belonging to the same person that differ only in their initial temporal location of detection. Finally, we cluster all the trajectories using bottom-up agglomerative clustering, prune the outliers and use the cluster centers as the final tracking results. Note that, in order to recover all trajectories, we need to consider overlapping consecutive blocks (red rectangles) as we are proceeding to generate the tracks for the next block. To summarize, for each of the D detections in a block, we run our tracker. This inference has computational complexity $O(DTH\mathcal{N})$, where T is the temporal length of a block in frames, and $\mathcal{N}$ the search neighbourhood size, as above. Feature computation near hypotheses is required, as is a mean-shift clustering on the D detections in the block. Finally, an agglomerative clustering on the D tracks with complexity $O(D^2)$ is performed.

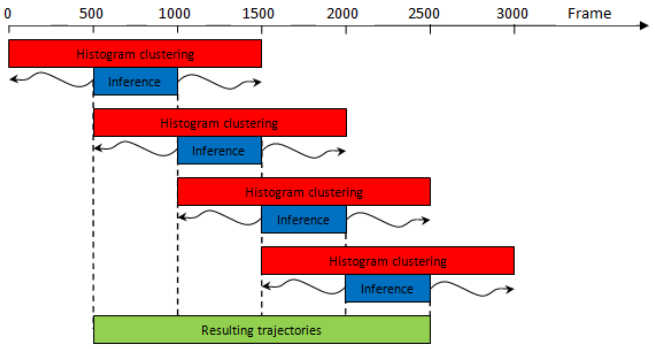


Figure 6: A chunking procedure is used for computational reasons. We divide the video into a number of overlapping sequences each of which contains 3 blocks of length T=500.

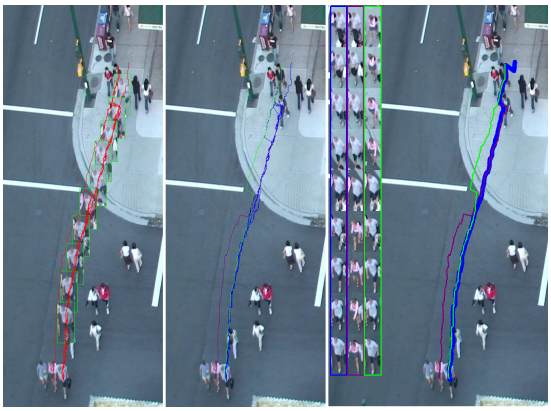
False alarms from the detector can still result in non-pedestrian objects being tracked. Further failures would be caused by people with very similar appearance who occlude each other. Also track "hijacks" that do not include a sufficient amount of stay in background to result in a low model score will cause problems. In spite of these potential issues, this procedure was successful in practice and we could track almost all the pedestrians successfully in our experiments.

This automatic detection and tracking framework is similar to that in [3]. However, Berclaz et al. use a ranking procedure which greedily picks the most promising i.e. highest quality trajectory based on their scoring function and removes those pixels from their hypothesis/search space so no other trajectory would ever be able to steal that space-time location. In contrast, our method independently clusters tracks, without a greedy, approximate consideration of space-time overlap.

Figure 7 demonstrates the result of our tracking system on one pedestrian. The image on the left shows MMTrack forward-backward inference results as red trajectories and the HOG detections as shadows with green borders. Three trajectory clusters are detected after clustering, and the membership of the clusters are shown in the middle image with trajectories coloured according to the cluster to which they belong. The means for the clusters are shown in the rightmost image with the width of a cluster trajectory being proportional to the number of trajectories belonging to that cluster. The blue cluster has the most members whose center is selected as recovered trajectory and other (singleton) clusters that belong to hijacks are discarded. Interestingly, almost all discarded trajectories in our experiments were the ones that were badly and/or untimely initialized namely observations that are not long enough to be disambiguated.

6. Experimental Results

In this section, we present our experiments on two real-world data sets. We start by introducing these data sets and the challanges in each of them along with our implementation details. We present qualitative results on long sequences and a demonstration where we provide empirical justification for our feature combination strategy. Next, we report quantitative results obtained on both datasets and show how the choices of loss function and inference scheme affect the performance.



(a) (b) (c)

Figure 7: Trajectory clustering result for one pedestrian. (a) Time lapse of all tracks containing the pedestrian. (b) Three clusters of trajectories. (c) Largest cluster, in blue, is correct cluster containing the pedestrian.

6.1. Descriptions of Datasets

UBC Fireworks Dataset: This dataset consists of clips at 1440 × 1080 resolution using a stationary camera installed on top of a building in downtown Vancouver which was initially recorded for transportation engineering data collection [13]. It contains both daytime and nighttime sequences. A top-down view of a moderately crowded scene is captured with a variety of moving objects typical to an urban setting. This includes cars, bicyclists, and pedestrians. The amount of change in illumination, scale and pose is not significant but one needs to deal with background clutter and partial occlusions. The main challenges in the dataset are the presence of crowded blobs of moving pedestrians that introduces many potential distractors and background change that occurs when people move from sidewalk to street area and vice versa (an example frame is shown in the left side of Fig. 2).

PETS09 Dataset: We use the S2.L1 dataset taken from the PETS 2009 competition. The dataset consists of a 794-frame video recorded at about 7 frames per second from a pedestrian path at a university campus. Unlike UBC Fireworks, this dataset has significant scale variations due to perspective effects. The viewpoint also introduces occlusion issues with occasions where pedestrians are completely occluded for a long time either by a background object or other pedestrians. Other challenges include the presence of many people dressing similarly in the frames and considerable pose change.

We use a portion of the UBC Fireworks as our training dataset, and the weights obtained are then used for testing of both UBC Fireworks (disjoint subset) and PETS09 datasets.

6.2. Implementation Details

The main bulk of computation time comes from the feature extraction. To reduce training time, we precompute HOG and colour histogram distance features prior to training and testing. Appearance templates, however, must be generated online since they are pairwise potentials and we compute them efficiently using integral images. We significantly reduce the space of possible trajectories in training by running the Viterbi algorithm in steps of nine pixels in both horizontal and vertical directions so the actual working resolution is 160 × 120. We define the neighborhood $\mathcal{N}_{(l_C)}$ to be the area within a radius of 2 pixels centered at the current location l_C. This choice is made based upon empirical statistics of the dataset which is in fact a function of the camera angle, average walking speed of pedestrians and frame rate. Similarly, we set the d_{max} in the motion model and ρ in the bounded loss to the same constant. Note also that because the testing processes images at different resolution from training, the motion model obtained from training must be adapted for testing. So, a simple nearest neighbor interpolation of the motion model is performed using the same number of discretization bins as in training and the weights are used directly.

For the histogram distance feature, we use the integral histogram optimization technique [20] to efficiently compute the histogram of any rectangular window in the image. With the integral histogram technique, all nine histograms representing the different body sections can be computed with a few arithmetic operations once the integral histogram is built. We also optimize computation of the appearance template features by computing the sum-of-absolute difference with a modified integral image technique [29]. However, due to the high resolution of the datasets (1440 × 1080 and 768 × 576 respectively), full inference in the original resolution turns out to be computationally prohibitive even with optimized feature generation. Therefore, we resort to pruning strategies for inference at test time by performing beam search. With beam search, we only evaluate the H highest-scoring trajectories at each time step (H is set to 3 in our experiments). Beam search allows us to perform inference in the original resolution at the expense of suboptimality as it only considers a small subset of the hypothesis space. However, applying beam search at training phase hurts the results since performance guarantees for the related learning algorithms build upon exact inference. Although both beam search and input subsampling explore only a subset of the whole space, our experimental results show that both approximations work well in practice.

The HOG detection window size for UBC Fireworks is set to 48 × 112 and the detection is performed on a pyramid whose scale varies up to 130% of the original resolution. The bounding box size that we use for computing other features is also fixed as scale and angle do not change substantially. This is not the case for PETS and we used the camera calibration information provided with the detaset to map 2D pixel coordinates to 3D world coordinates and vice versa. We assume $\rho = 1.8$m, pedestrian height of 1.6m and height:width ratio of 4 : 1 to rescale the bounding boxes and the motion model (measured in

meters) appropriately. For the colour histogram distance feature, this means computing the histograms over sections of the rescaled bounding box. For appearance template features, both the template and the hypotheses are rescaled to a fixed resolution at which the sum of absolute differences is then computed.

6.3. Qualitative Evaluation

We first describe qualitative detection and tracking results on the two datasets. We reiterate that training is done using a subset of the UBC Fireworks dataset, with testing on a disjoint subset as well as transfer of these parameters to the PETS09 dataset.

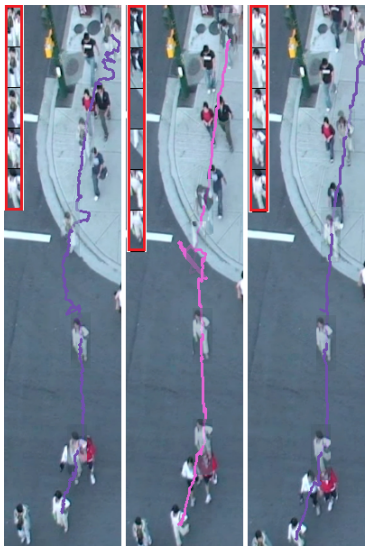
6.3.1. UBC Fireworks Qualitative Evaluation

We use three UBC Fireworks dataset clips for our qualitative evaluation. The three video clips are recorded at 25 frames per second, with durations of 6m:18s, 6m:46s, and 3m:20s. The first two clips were taken in daytime, the latter in nighttime. The same set of weights are used in all our experiments. To generate the trajectories, we run our automatic detection and tracking system as described in Sec. 5. Fig. 8 shows our performance on a daytime clip. We observe that our tracker does a decent job even in crowded conditions. We could identify two rare sources of error in our tracking system as being HOG false alarms resulting in non-pedestrian objects such as cars to be tracked, and pedestrian full occlusion that occurs when they walk behind another object such as a tree. All demo videos are available at our project website[1].

Fig. 9 is an example demo from the UBC Fireworks dataset that illustrates the importance of our cue combination strategy. As observed in the subfigures, our tracker with only HOG and histogram distance features drifts to a nearby distractor at some parts of the track as it does not know about the initial appearance of the person. The jitter in the trajectory is mostly due to lack of fine details of the appearance which roughly makes the neighbors become equally good according to the model. On the other hand, with HOG and appearance templates, the tracker gets stuck in background area at the boundary between the sidewalk and the street. The reason is that the significant change in appearance template from frame to frame which occurs at this boundary is rare and hence not supported

(a) (b)

Figure 8: Sample tracking results on crowded conditions.



(a) (b) (c)

Figure 9: An example illustrating the intuition behind our model design. The tracked "objects" throughout the trajectories are shown in red insets and also superimposed along the trajectories. (a) HOG+histograms, (b) HOG+templates, (c) HOG+histogram+templates.

by the average statistics encoded by the model. So, the tracker is not robust against sudden changes in appearance. Also, since information about average appearance and average background is lacking, drift is inevitable and the role of initial template breaks as background pixels make the stay in sidewalk more rewarding than moving toward the street. The combination of HOG, distance maps and appearance templates manages to track the person correctly. In this case, the system is stable against rapid changes while being reasonably accurate. Note that the same situation happens for PETS. However, the role of features in that data set depends on how clean the features we obtain are in practice after performing the rescaling.

6.3.2. PETS Qualitative Evaluation

We also conduct qualitative evaluation on the PETS 2009 S2.L1 dataset where we use the same weight parameters obtained from our Fireworks experiment. Generally, the tracker can track most pedestrians when they are not occluded. Two main causes of occlusions in the dataset are the signpost located in the middle of the image and close interaction with people with similar appearance.

In the PETS video, 44 trajectory clusters are displayed sequentially in descending order according to the ratio of the number of members to the length of the cluster mean. The results show promising performance, with only one mistrack in the first ten trajectories and five mistracks in the first twenty trajectories. Two full-trajectory tracking results are shown in Fig. 10. In the figure, target pedestrians are indicated by red bounding boxes, the tracker's resulting trajectories are shown in red trajectories, and snap-

shots from various points during the trajectories are shown in the blue insets. In the top image, the tracker managed to track the target when he was partially occluded by a signpost, but drifted when the target was occluded by a similar-looking pedestrian. The bottom image shows a successful long-duration tracking even on occasions of full occlusion. In this example, the target pedestrian's appearance is different from other pedestrians he interacts with and the tracker can still find him after occlusion as he never moves farther than the limit of the motion model.

6.4. Quantitative Results

We compare our single-target tracker with algorithms proposed by Collins et al. [6] and Babenko et al. [2]. We used the MIL-tracker software provided by the authors and implemented our own version of [6] which we call the *Collins-Liu* tracker. We only use the Fireworks dataset as these methods are not designed to handle scale variations. We use 10 manually-labeled trajectories from different sequences for training. Both training and test examples are of length 350-500 and contain easy, moderate and hard sequences ranging from a solitary person going through the scene to a pedestrian walking within a crowd with partial occlusions. We chose 22 other trajectories and manually labeled them for evaluating the performance. We run independent instances of the tracker forward and backward in time in order to get complete trajectories starting from the fixed set of selected detections. Trackers are terminated once they are within a certain number of pixels from image borders. We use the same procedure to extract trajectories using other methods so we can make a fair comparison.

Besides the usual average pixel error measure (Avg. Err.), we use two other performance measures proposed in [33, 14]. *Correct Detected Track* (CDT) indicates the number of correct trajectories. A track is defined as a CDT if the amount of spatial and temporal overlap with the ground truth exceed thresholds T_{ov} and TR_{ov} respectively, where T_{ov} and TR_{ov} are both set to 0.5 in our experiments. This roughly means that at least half of a CDT must temporally coincide with its ground truth, its length cannot be less than half of its ground truth, and the average spatial overlap must be at least 0.5. *Multiple Object Tracking Precision* (MOTP) [14] or *Closeness of Track* (CT) [33] is defined as the average spatial overlap between a ground truth and a system track in the temporally coincident portion of the track. Its value ranges from 0 to 1, with 1 indicating that the track is exactly the same as the ground truth in the temporally coincident section of the track. More detailed explanation of the measures are provided in [33, 14]. Note that since we are focused on single target tracking, other performance measures from [14] are not applicable.

[1] `http://www.cs.sfu.ca/research/groups/VML/MMTrack.html`.

Tracker	#CDT	MOTP	Avg. Err.
MMTrack:All	**21**	**0.67**	**7.01**
MMTrack:Hist+Templates	20	0.61	12.74
MMTrack:HOG+Templates	14	0.52	22.24
MMTrack:HOG+Hist	10	0.47	14.40
MILTrack	19	0.61	19.87
Collins-Liu	14	0.54	21.24

Table 1: Comparison of results on UBC Fireworks dataset. Higher #CDT and MOTP are better, lower Avg. Err. is better.

Tracker	# CDT	MOTP	Avg. Err.
MMTrack:All	**21**	0.61	**10.37**
MMTrack:HOG+Hist	20	**0.64**	13.48
MMTrack:HOG+Templates	15	0.56	22.16

Table 2: Quantitative results on 25 targets from PETS 2009 dataset. Higher #CDT and MOTP are better, lower Avg. Err. is better.

6.4.1. UBC Fireworks Quantitative Evaluation

To gain more insight into the importance of our feature combination, we design experiments on both datasets with groups of features turned off. For this purpose, we learn different sets of parameters for each combination of the features independently. Table 1 presents the results on Fireworks.

The second set of experiments, also shown in Table 1, compare our tracker with other trackers on the same test set. As we can see in the table, our tracker outperforms the MIL-tracker [2] and Collins-Liu tracker [6] in this dataset. These results provide a system-level comparison, since these trackers build upon different feature sets. Also, both [6] and [2] are only concerned with appearance modeling and do not include a parameterized motion model. One can explain this promising performance by reasoning about the role of different cues in our system. Specifically, the HOG feature helps the tracker eliminate areas belonging to non-pedestrian objects, the histogram distance maps provide a rough description of the pedestrian and helps alleviate drift whereas appearance templates provide finer levels of the appearance, with the previous frame appearance template allowing some degree of adaptability to appearance change over time.

Table 1 shows that removing some of the features significantly reduces the performance indicating that the combination of HOG, histogram distance and template appearance features is essential in achieving good performance. While each feature group is responsible for avoiding certain types of failures, interactions between groups accounts for difficult situations. Hence, when a feature group is discarded, not only the corresponding failure modes show up but also more complicated failure modes occur as feature groups are not independent.

Table 1 indicates that the most important components of our tracker are colour histograms and appearance templates. The second row, which corresponds to the system with the second best results, does not include HOG score which implies that our tracker does not really depend on

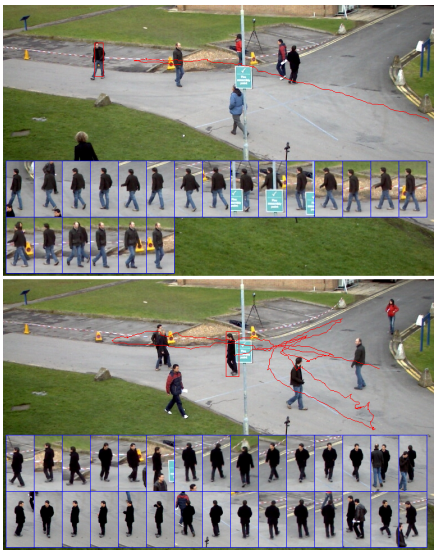


Figure 10: Sample tracking results on PETS dataset.

Learning	Loss Type	# CDT	MOTP	Avg Err
Exact	Δ	15	0.56	11.38
Exact	$\mathbf{\Delta_B}$	**21**	**0.67**	**7.01**
Approx.	Δ	17	0.61	9.90
Approx.	Δ_B	20	0.64	12.24

Table 3: Our tracking results with different learning schemes and loss functions on 22 test samples from UBC Fireworks.

a detector although a good one would slightly improve the performance.

Note that our MMTrack approach uses more features, and hence is computationally more demanding. Running on a 2.4GHz Intel Core2 Q6600 workstation with 8GB RAM, MMTrack takes on average 1253ms to process a frame, compared to 385ms for our Collins-Liu implementation and 493ms for MILTrack. These times exclude HOG detction, which takes ≈100s on an entire 1440x1080 frame using the code provided by the authors.

6.4.2. PETS Quantitative Evaluation

We also consider PETS data for our quantitative evaluations. This experiment is done with the parameters learned using the Fireworks dataset and serves as an evaluation for our method on unseen data. Reported tracking results on PETS datasets are all multi-target tracking algorithms and our method, as a single-target tracker, is not directly comparable to them. Moreover, there is no publicly available state-of-the-art single-target tracking software that handles scale change. Therefore, we do not perform comparisons with other trackers for this dataset. Because full occlusion is very common in this dataset and our single-target tracker has no inherent mechanism to perform occlusion reasoning, we split the ground truth to segments that do not contain such instances, resulting in 25 ground truth tracks.

Table 2 summarizes the results on PETS. Again, the same observations about feature combinations apply – adding features improves performance.

6.4.3. Learning Scheme and Loss Functions

We also tried exact and approximate learning schemes as well as bounded and unbounded loss while keeping the inference the same for all the experiments. Table 3 shows the performance of our tracker. As seen in the table, exact training using bounded loss achieves the best result in all measurements among all the configurations of MMTrack. Theoretical guarantees of the optimization algorithm explains the superiority of the exact training over approximate training. We believe that bounded loss better matches the nature of our measurements as it is closer to the overlap criterion in CDT and stops (over) penalizing as soon as the overlap becomes zero and so performs better compared to unbounded loss in all settings as expected.

7. Conclusion

In this paper, we introduced an offline tracker that employs a large margin learning criterion to effectively combine different trackers. Although MMTrack is used for pedestrian tracking in this work, we believe that our framework is general and can be used to track other objects provided that features can reliably describe the target object and handle situations of interest while avoiding confusions for our discriminative classifier. For instance, one could model articulated objects in the same way as we included our set of features.

The version of MMTrack we have described is a single-target tracker and thus it has no capability to reason about full occlusions. We believe that rather than adding occlusion handling capability to a single-target tracker, better results can be achieved by using a multi-target tracking framework. In contrast to single-target trackers that assign a trajectory to each target without considering other objects, multi-target trackers jointly consider the state of all targets in determining their trajectories. We believe that extending this framework to multiple target tracking would be fruitful ground for future research.

Our tracking system has limitations in handling severe occlusion and track hijacks caused by significant change in appearance or situations where the background patch is very similar to the appearance of the target. Incorporating mechanisms that would enable single target trackers to explain long-term occlusions while avoiding distractor hijacks turns out to be very challenging. Again, incorporating long term occlusion into the model and learning procedure would be interesting future research.

Another assumption in our current model is that the same linear weights are valid thoughout the tracking. One could also define variants of our model that have non-linear weights, or weights that are functions of properties of the scene or tracking situation. For example, learning different parameters for different locations in the scene may also

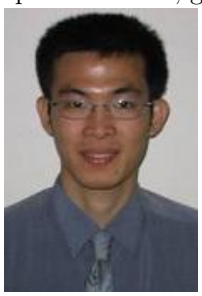
be of interest. Such a tracking system would be able to deal with location specific situations that are difficult to handle for a generic tracker. This is motivated by the intuition that the relative importance of the features is likely to be affected by the statistics of background patches and particular occlusions at different locations. On the other hand, designing trackers with more complicated statistics or background models could result in better performance. Finally, defining suitable problem-specific loss functions that directly optimize for benchmark measurements is desirable.

Acknowledgements

This work was supported by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC) and BCFRST Natural Resources and Applied Sciences (NRAS) Research Team Program.

Author Biographies


Bahman Yari Saeed Khanloo received his BSc in computer science from University of Tehran, Iran, and an MSc degree in computer science from Simon Fraser University, Canada. Currently, he is a research intern at A*STAR/NUS under SINGA international scholarship. His research interests are in convex optimization and matrix methods for machine learning, probabilistic graphical models, games and learning theory.


Ferdinand Stefanus received the BEng degree from the School of Computer Engineering, Nanyang Technological University, Singapore, in 2003, and the MSc degree in Computing Science from the School of Computing Science, Simon Fraser University, Canada, in 2011. Prior to obtaining his MSc, he worked at Institute for Infocomm Research, developing and deploying computer vision applications. He is now with MacDonald, Dettwiler, and Associates Ltd. His research interests include computer vision, software engineering, and machine learning.


Mani Ranjbar is currently a PhD candidate at the School of Computing Science, Simon Fraser University, Canada. He received his M.Sc. in Computer Architecture from Sharif University of Technology, Iran in 2007 and his B.Sc. in Computer Engineering from the same university in 2005. His research interests are in computer vision and machine learning including object detection, segmentation and tracking.


Ze-Nian Li is a Professor in the School of Computing Science at Simon Fraser University, British Columbia, Canada. Dr. Li received his undergraduate education in Electrical Engineering from the University of Science and Technology of China, and M.Sc. and Ph.D. degrees in Computer Sciences from the University of Wisconsin-Madison under the supervision of the late Professor Leonard Uhr. His current research interests include computer vision, multimedia, pattern recognition, image processing, and artificial intelligence.


Nicolas Saunier obtained his Ph.D. in Computer Science in France in April 2005 at Telecom ParisTech, using Machine Learning methods to study the influence of traffic control in a signalized intersection on the risk of road users. For the next 4 years as postdoctoral fellow and research associate at the University of British Columbia, he developed automated methods for traffic monitoring and road safety analysis, including for pedestrians. Since September 2009, he is an assistant professor in Transportation at the École Polytechnique de Montréal. His interests include intelligent transportation systems, road safety, and information technology for transportation (data collection, storage, processing, and visualization).

Tarek Sayed is a Professor and a Distinguished University Scholar at the University of British Columbia. He is the Editor of the Canadian Journal of Civil Engineering and the Director of the Bureau of Intelligent Transportation Systems and Freight Security (BITSAFS-Engineering) at UBC. Dr. Sayed has numerous awards recognizing his work to advance Transportation Engineering Research and Education.

Greg Mori received the Ph.D. degree in Computer Science from the University of California, Berkeley in 2004. He received an Hon. B.Sc. in Computer Science and Mathematics with High Distinction from the University of Toronto in 1999. He is currently an Associate Professor in the School of Computing Science at Simon Fraser University. Dr. Mori's research interests are in computer vision, and include object recognition, human activity recognition, human body pose estimation.

[1] Andriyenko, A., Schindler, K., 2010. Globally optimal multi-target tracking on a hexagonal lattice, in: European Conference on Computer Vision.

[2] Babenko, B., Yang, M.H., Belongie, S., 2009. Visual tracking with online multiple instance learning, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[3] Berclaz, J., Fleuret, F., Fua, P., 2006. Robust people tracking with global trajectory optimization, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[4] Bradski, G.R., 1998. Computer vision face tracking for use in a perceptual user interface. Intel Technology Journal.

[5] Breitenstein, M.D., Reichlin, F., Leibe, B., Koller-Meier, E., Gool, L.V., 2010. Online multi-person tracking-by-detection from a single, uncalibrated camera. IEEE Transactions on Pattern Analysis and Machine Intelligence (to appear).

[6] Collins, R., Liu, Y., Leordeanu, M., 2005. Online selection of discriminative tracking features. IEEE Transactions on Pattern Analysis and Machine Intelligence 27, 1631 –1643.

[7] Comaniciu, D., Ramesh, V., Meer, P., 2003. Kernel-based object tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 25, 564–575.

[8] Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[9] Darrell, T., Gordon, G., Harville, M., Woodfill, J., 1998. Integrated person tracking using stereo, color, and pattern detection, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[10] Davis, J.W., 2001. A perceptual user interface for recognizing head gesture acknowledgements, in: In ACM Workshop on Perceptual User Interfaces, pp. 15–16.

[11] Du, W., Piater, J., 2008. A probabilistic approach to integrating multiple cues in visual tracking, in: European Conference on Computer Vision (ECCV), Springer-Verlag.

[12] Giebel, J., Gavrila, D., Schnrr, C., 2004. A bayesian framework for multi-cue 3d object tracking, in: European Conference on Computer Vision (ECCV).

[13] Ismail, K., Sayed, T., Saunier, N., 2009. Automated collection of pedestrian data using computer vision techniques, in: Transportation Research Board Annual Meeting Compendium of Papers, Washington, D.C.

[14] Kasturi, R., Goldgof, D., Soundararajan, P., Manohar, V., Garofolo, J., Bowers, R., Boonstra, M., Korzhova, V., Zhang, J., 2009. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. IEEE Transactions on Pattern Analysis and Machine Intelligence 31, 319 –336.

[15] Khanloo, B.Y.S., Stefanus, F., Ranjbar, M., Li, Z.N., Saunier, N., Sayed, T., Mori, G., 2010. Max-margin offline pedestrian tracking with multiple cues, in: Seventh Canadian Conference on Computer and Robot Vision (CRV).

[16] Kolsch, M., Turk, M., 2004. Fast 2d hand tracking with flocks of features and multi-cue integration, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[17] Leichter, I., Lindenbaum, M., Rivlin, E., 2004. A probabilistic framework for combining tracking algorithms, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[18] Moreno-Noguer, F., Sanfeliu, A., Samaras, D., 2008. Dependent multiple cue integration for robust tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 30, 670–685.

[19] Nejhum, S.M.S., Ho, J., Yang, M.H., 2008. Visual tracking with histograms and articulating blocks .

[20] Porikli, F., 2005. Integral histogram: A fast way to extract higtograms in cartesian spaces, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[21] Ramanan, D., Forsyth, D., Barnard, K., 2006. Building models of animals from video. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 1319–1334.

[22] Roh, M.C., Kim, T.Y., Park, J., Lee, S.W., 2007. Accurate object contour tracking based on boundary edge selection. Pattern Recognition 40, 931 – 943.

[23] Ross, D.A., Lim, J., Lin, R.S., Yang, M.H., 2008. Incremental learning for robust visual tracking. International Journal of Computer Vision (IJCV) 77, 125–141.

[24] Shi, J., Tomasi, C., 1994. Good features to track, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[25] Spengler, M., Schiele, B., 2001. Towards robust multi-cue integration for visual tracking, in: Proceedings of the Second International Workshop on Computer Vision Systems.

[26] Stenger, B., Woodley, T., Cipolla, R., 2009. Learning to track with multiple observers, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[27] Taycher, L., Demirdjian, D., Darrell, T., Shakhnarovich, G., 2006. Conditional random people: Tracking humans with crfs and grid filters, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[28] Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y., 2004. Support vector machine learning for interdependent and structured output spaces, in: Proceedings of the Twenty-first International Conference on Machine Learning (ICML).

[29] Viola, P., Jones, M., 2004. Robust real-time object detection. International Journal of Computer Vision (IJCV) 57, 137–154.

[30] Wu, Y., Huang, T.S., 2004. Robust visual tracking by integrating multiple cues based on co-inference learning. International Journal of Computer Vision (IJCV) 58, 55–71.

[31] Yilmaz, A., Javed, O., Shah, M., 2006. Object tracking: A survey. ACM Comput. Surv. 38.

[32] Yilmaz, A., Li, X., Shah, M., 2004. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. IEEE Transactions on Pattern Analysis and Machine Intelligence 26.

[33] Yin, F., Makris, D., Velastin, S.A., 2007. Performance evaluation of object tracking algorithms, in: IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS2007).