Multi-Object Tracking via Constrained Sequential Labeling

Sheng Chen, Alan Fern and Sinisa Todorovic
Oregon State University
{chenshen, afern, sinisa}@eecs.oregonstate.edu

Abstract

This paper presents a new approach to tracking people in crowded scenes, where people are subject to long-term (partial) occlusions and may assume varying postures and articulations. In such videos, detection-based trackers give poor performance since detecting people occurrences is not reliable, and common assumptions about locally smooth trajectories do not hold. Rather, we use temporal mid-level features (e.g., supervoxels or dense point trajectories) as a more coherent spatiotemporal basis for handling occlusion and pose variations.Thus, we formulate tracking as labeling mid-level features by object identifiers, and specify a new approach, called constrained sequential labeling (CSL), for performing this labeling. CSL uses a cost function to sequentially assign labels while respecting the implications of hard constraints computed via constraint propagation. A key feature of this approach is that it allows for the use of flexible cost functions and constraints that capture complex dependencies that cannot be represented in standard network-flow formulations. To exploit this flexibility we describe how to learn constraints and give a provably correct learning algorithms for cost functions that achieves finite-time convergence at a rate that improves with the strength of the constraints. Our experimental results indicate that CSL outperforms the state-of-the-art on challenging real-world videos of volleyball, basketball, and pedestrians walking.

1. Introduction

This paper presents a new approach to tracking multiple interacting people in real-world videos, where detecting people occurrences is not reliable. Examples include videos of pedestrians in crowded scenes, and team sports, like basketball and volleyball. In these videos, people are subject to long-term partial or full occlusions. Also, in sports videos, players may assume a wide range of poses and body articulations, and their motions and appearance (same team uniforms) are typically similar to those of nearby players.

These challenges make the state-of-the-art people detectors unreliable. As illustrated in Fig. 1, our experiments demonstrate that responses of the popular DPM detector [7] are very noisy, leading to poor performance of state-of-the-art trackers based on data association of detections. This raises the question of what video features could be more suitable in our setting for grounding data association.

We resort to temporal mid-level features – namely, supervoxels [20], and dense point trajectories [5] – as a more coherent spatiotemporal basis for handling occlusion, pose variations, and non-smooth trajectories. For example, boundaries of supervoxels typically align with people's contours. Therefore, tracking the right combination of mid-level features facilitates bottom-up reasoning under occlusion and varying poses. As Fig. 1 shows, this can help disambiguate two people partially occluding each other. These mid-level features, however, present fundamental challenges to the common network-flow, and related formulations of data association which use object detections (e.g., [14, 13, 2, 19, 21, 17]). In particular, a person to be tracked is typically represented by an unknown number of mid-level features, which split and merge in both space and time. Consequently, edge capacities can no longer be uniformly one and the ideal settings are not clear. Post-processing is also required to recover identities from flows.

One approach to the above problem might be to apply heuristic simplifications until the resulting problem fits into a standard network flow formulation and then solve the problem using efficient solvers. For example, we can make hypotheses about places of merging and splitting according to the size of mid-level features. However, the information loss and errors injected by applying such simplifications can often lead to an unrecoverable loss in accuracy, negating the advantage of using efficient "optimal" solvers. Further, the types of constraints and affinities among mid-level features that can be represented in such Markovian flow networks is quite limiting. Our initial efforts toward this style of approach have not yet been successful.

In this work, rather than heuristically simplify the mid-level labeling problem to an efficiently solvable framework, we instead develop a greedy labeling approach that can operate directly on the original (unsimplified) problem. In particular, our approach conducts tracking by sequentially la-

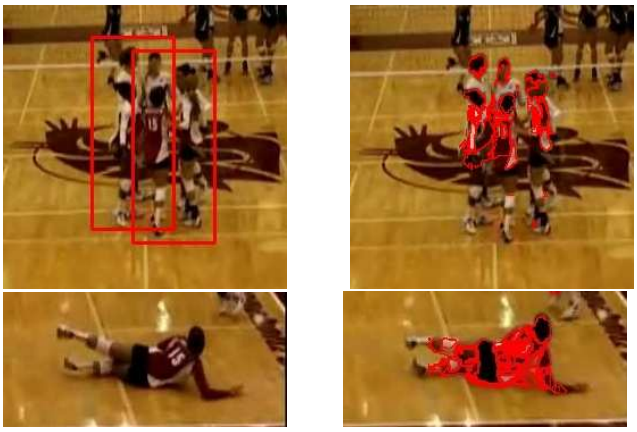

Figure 1: Bounding boxes of the DPM detector [7] (left), and supervoxels of [20] (right). DPM often misses to detect players in crowds (top) and in unexpected poses (bottom). Supervoxels are more suitable for our challenging videos, because their boundaries are usually aligned with part boundaries facilitating bottom up reasoning.

beling mid-level video features with object identifiers, under *hard* constraints. Our new algorithm, called constrained sequential labeling (CSL), uses a flexible cost function to sequentially assign labels while directly respecting the implications of hard constraints (e.g., a person cannot be at two distinct locations). A key advantage of this approach is that it places few restrictions on the form of the cost function, allowing it to capture higher-order dependencies among mid-level features. We describe supervised learning algorithms for acquiring both the constraints and cost functions.

Importantly, while our approach is inherently greedy, it comes with strong theoretical guarantees. We prove that if there is a cost function (within the considered cost function space) that supports accurate labeling given the constraints, then the learning algorithm will achieve finite-time convergence to such a cost function, where the rate of convergence improves with the "strength" of the constraints. As our experiments demonstrate these assumptions are satisfied in challenging real-world problems, where we are able to learn high-quality constraints and cost function that support accurate CSL. In particular, we present experiments using sports and pedestrian data that demonstrate significant improvements over the state-of-the-art.

2. Prior Work

Existing multi-object trackers typically use points, tracklets, or object detections [21, 17, 12, 2, 19]. The previous section highlighted the challenges of using object detections with significant occlusion and pose variation. Similarly, interest points are not suitable features for our setting, since they do not capture the spatial cohesiveness of objects, and thus require making heuristic assumptions about motions, sizes, and shapes of objects to resolve occlusions [18, 23].

For data association, dependencies between video features are typically captured by a network and are often simplified to the first or second order Markovian dependencies [14, 4]. Some approaches seek to encode hard constraints, and thus facilitate data association in ambiguous cases. However, affinities between video features are typically combined with hard constraints into an NP-hard optimization problem [3, 15]. This, in turn, requires a relaxation of the hard constraints to the continuous domain, which can often lead to non-sensical results (e.g., two distinct object detections are assigned the same object ID) and requiring heuristic post-processing. Rather, our CSL approach directly uses such hard constraints via the use of constraint propagation, which our experiments demonstrate is effective.

CSL is inspired by the recent success of sequential classification methods for solving structured prediction problems (e.g. [10]) where classifiers are used to sequentially label structured objects. However, CSL overcomes the potential shortsightedness of such approaches through the incorporation of hard constraints.

3. Constrained Sequential Labeling

Our input for tracking is a set of n mid-level, temporal features (e.g. supervoxels) $\{S_1, S_2, \ldots, S_n\}$ that capture the foreground object activity (from foreground or saliency detection) in a video. The goal is to label each S_i by a label $l_i \in \{1, \ldots, k\}$, which indicates which of k objects the feature corresponds to. For simplicity we first assume that k is known and drop this assumption later. We assume that the visible target objects in the initial frame are labeled (manually or by a detector) and also that mid-level features are at a fine enough resolution so that they typically respect the space-time extent of objects.

CSL operates by both assigning labels to features and removing labels. For this purpose, we define a *partial labeling* L as a mapping from features to label sets, such that $L(S_i) \subseteq \{1, \ldots, k\}$ is the set of labels that are not ruled out for S_i. If $L(S_i) = \{l\}$ then S_i is assigned label l. We say S_i has a *partial label* if $|L(S_j)| > 1$. CSL iteratively refines a partial labeling L, which is initialized so that $L(S_i) = \{1, \ldots, k\}$ if S_i does not start from the first frame (i.e. all labels are possible), and otherwise $L(S_i) = \{l\}$, where l is the label specified for S_i in the first frame. Each refinement iteration uses a *cost function* and *inequality constraints* in order to refine the partial labeling until reaching a fully specified labeling. We describe this process below, first ignoring constraints and then incorporating constraints. Alg. 1 gives pseudo-code for CSL. Note that lines 6-12 in Alg. 1 pertain to learning, explained in Sec. 4.

Labeling with a Cost Function. We assume a cost function $C(S_i, l \mid L)$ that assigns a cost to the decision of label-

input : A set of mid-level features $\{S_i\}$
Object labels for first frame
Weight Vector w
Set of inequality constraints Σ
Learning Option $learn$ with ground truth labeling L_{gt}
output: A labeling L or weight vector w

1 Initialize partial labeling L according to label information in first frame
2 **while** *there exists* S_i *such that* $|L(S_i)| > 1$ **do**
3 $L = \text{ConstraintPropagation}(L, \Sigma)$
4 $D = \{(S, l) : S \in \mathcal{A}(L), l \in L(S)\}$
5 $(S_i, l) = \arg\min_{(S,l)\in D} C(S, l \mid L)$
6 **if** $learn$ **and** $L_{gt}(S_i) \neq l$ **then**
7 $C^+ = \{(S, l) \in D | L_{gt}(S) = \{l\}\}$
8 $(S^+, l^+) = \arg\min_{(S,l)\in C^+} C(S, l \mid L)$
9 $C^- = \{(S, l) \in D - C^+ \mid C(S, l|L) < C(S^+, l^+|L)\}$
10 $w = w + \sum_{(S,l)\in C^-} \frac{\Psi(S,l,L)}{|C^-|} - \sum_{(S,l)\in C^+} \frac{\Psi(S,l,L)}{|C^+|}$
11 $L(S^+) = \{l^+\}$
12 **end**
13 **else**
14 $L(S_i) = \{l\}$
15 **end**
16 **end**
17 **if** $learn$ **then Return** w ;
18 **else Return** L ;

Algorithm 1: Constrained Sequential Labeling Algorithm with learning option. When the flag $learn$ is set, the algorithm performs one run of learning the weight vector w.

ing S_i as $l \in L(S_i)$, where L is the current partial labeling being refined. Given the current partial labeling L, each iteration begins by considering a set of *active* features (see below) denoted by $\mathcal{A}(L)$ as possibilities for assigning a new label. The cost function C is then used to select an active feature $S_i \in \mathcal{A}(L)$ and corresponding label $l \in L(S_i)$ to assign to it by setting $L(S_i)$ to $\{l\}$. In particular, the pair (S_i, l) is selected to be the least cost such pair considering all active features and labels for those features. These steps are depicted in lines 4,5, and 14 of the pseudo-code.

Intuitively, we want C to assign the lowest costs to correct pairs that can be most confidently labeled in the context of L. In this work, we will use linear cost functions that are expressed as $C(S_i, l \mid L) = w \cdot \Psi(S_i, l, L)$, where w is an m-dimensional weight vector and $\Psi(S_i, l, L)$ returns an m-dimensional descriptor vector that represents information about the corresponding decision. The descriptors used in this work are presented at the end of this section and Section 4 gives a learning algorithm for w.

It remains to specify the active features $\mathcal{A}(L)$. S_i is active if its label is partial and is temporally close to an already labeled feature (see Figure 2). Restricting our attention to active features reduces the number of cost function evaluations, which can result in significant speedups.

Inequality Constraints and Propagation. Using the cost function to sequentially assign all labels can be effective for simpler tracking problems, e.g. ones that do not involve significant occlusion. However, for more difficult problems, we have found it is difficult to learn or hand-code cost functions that achieve state-of-the-art performance. This is due to the short-sighted, greedy nature of the decision sequence. Fortunately we have found that this poor performance can be overcome by combining greedy selection with constraint reasoning. Section 4 provides a theoretical characterization of this observation.

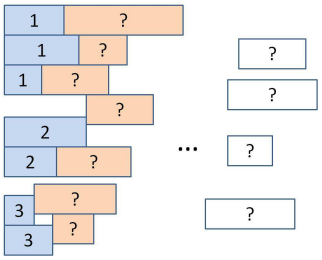

Figure 2: Example of sequential labeling of supervoxels. Blue rectangles represent already labeled supervoxels, the neighboring shaded rectangles are active supervoxels that will be considered for labeling, the remaining rectangles will not be considered until they become active.

Our CSL approach uses a set of inequality constraints Σ associated with each video. Each constraint is of the form $S_i \neq S_j$, indicating that the features cannot be assigned the same label. A partial labeling L *violates* a constraint $S_i \neq S_j$ if $L(S_i) = \{l\}$ and $l \in L(S_j)$ or vice versa. L is consistent with Σ if it is not violated by any constraint.

A basic way to combine constraints Σ and a cost function is to limit the labeling choices of the cost function to ones that will not result in constraint violations given the current partial labeling L. While using constraints in this way can improve efficiency by reducing the number of cost function evaluations, our results show that the accuracy improvement is modest. The weakness of this approach is that it only considers greedy constraint violations, rather than further reaching implications of the constraints. For example, suppose that currently $L(S_i) = \{1\}$ and $L(S_j) = \{1, 2\}$, and we have the constraint $S_i \neq S_j$. We can infer that the label for S_j is 2 since it cannot be 1 and given this new information we can eliminate label 2 as a possibility from any feature that has an inequality constraint with S_j, and so on. This process of iteratively computing the implicit consequences of constraints is known as constraint propagation [6] and is an extremely important component of state-of-the-art constraint satisfaction and optimization algorithms.

Given a current partial labeling L and constraints Σ we let ConstraintPropagate(L, Σ) denote the constraint propagation procedure, which returns a refined partial labeling L' that is consistent with Σ. Specifically, L' will be the minimal refinement of L (maximally close) that is consistent. ConstraintPropagate(L, Σ) is straightforward to implement for inequality constraints. Starting with L iteratively finds a

violated constraint $S_i \neq S_j$ such that $L(S_i) = \{l\}$ and removes label l from $L(S_j)$ to get a new labeling. The process terminates when no violation can be found.

CSL integrates constraint propagation by calling the procedure at the beginning of each iteration (line 3). Thus, in the first iteration label information will be propagated from the known labels of the first frame. In later iterations, label information will be propagated in response to the labeling decision made by the cost function in the previous iteration. This propagation can rule out many possible labels that might have otherwise been considered in current or later iterations by the cost function. In this way, the constraints have the effect of improving efficiency since the number of cost function calls can be reduced and increasing accuracy, because some of those removed cost function calls may have resulted in erroneous label assignments. Our results show the improvements are significant.

Handling Unknown Numbers of Objects. Above the number of objects k was assumed known as in many sports domains. However, CSL is easily modified to handle unknown k such as pedestrian tracking. The above CSL description is unchanged, except that k is initialized to be the number of labeled objects in the first frame. During the iteration the label set is enlarged (k is increased) whenever the results of constraint propagation remove all label possibilities for some feature S (i.e. $L(S) = \emptyset$), indicating that S should not be labeled as any of current objects. Whenever this happens, a new object is added (k is incremented) to the set of possible labels of any partially labeled features and S is assigned the new label. When the learned constraints in Σ are noisy, we may erroneously remove a correct label from $L(S)$ resulting in $L(S) = \emptyset$, which will hurt performance. Our experiments show that this is rare and that the constraints provide an overwhelming positive impact.

Cost Function Descriptors. Given a partial labeling L, the descriptor function $\Psi(S_i, l, L)$ is defined as:

$$\Psi(S_i, l, L) = [n_u, n_d, n_f, d_l, d_v, d_c]$$

The first type of descriptors n_u, n_d and n_f are the number of unlabeled features in S_i's neighborhood, the number of different labels in the neighborhood, and the number of frames S_i doesn't cover respectively. These descriptors provide measures related to how confidently a feature can be labeled, which allows a preference for labeling temporally longer features consistently with its surroundings.

The second set of descriptors d_l, d_v and d_c encodes the usual visual dissimilarity between a feature and an object template in terms of location, optical flow and color histograms. The *object template* for label l is the union of all S_i such that $L(S_i) = \{l\}$. For location and optical flow, the template is maintained every frame, and dissimilarity is calculated by averaging over the Euclidean distance between the feature and the template across each overlap frame. For color histogram we keep one model and the dissimilarity is directly the distance between the feature and the template. In order to account for the difference of color and motion among parts of an object, the template for color and optical flow is clustered into 3 parts using k-means. Given a feature and the template, the dissimilarity for color and motion is calculated according to the closest cluster. We update the corresponding object template after each CSL iteration. Note that by introducing the object template, the cost function becomes extremely high order compared to most previous work where only lower order relations are considered.

4. Cost Function Learning

We now describe a learning algorithm for tuning the weights of our linear cost function $C(S_i, l \mid L) = w \cdot \Psi(S_i, l, L)$ and give conditions for its convergence. We assume labeled training videos and associated constraints that provide the ground truth label assignments L_{gt} for features. The goal is to learn a set of weights such that CSL achieves high accuracy on the training data.

This learning problem poses at least two challenges over standard supervised learning. First, the training data is ambiguous since it does not indicate the exact sequence of labeling decisions. The second challenge is that the sequence of decisions made by the cost function are not independent since each decision depends on the current partial labeling L, which was generated based on previous decisions.

Our algorithm, CSL-Learn, addresses these challenges by directly integrating learning into CSL. The outer loop iterates through each training video, possibly performing multiple updates to the weight vector w, and terminates when either a specified number of iterations is reached or accuracy is perfect. For each training video, the learning algorithm executes a "learning enhanced" CSL procedure given by Algorithm 1 with the flag *learn* turned on.

The algorithm behaves likes CSL until the cost function using w suggests an incorrect labeling decision (line 6). At this point the weight vector is updated in a way that discourages errors at similar decision points in the future (see below). In addition, the incorrect decision is not used to update the partial labeling, but rather the least cost correct labeling decision is used to update L (line 11).

It remains to specify the weight vector update when an incorrect label l is assigned to feature S_i given partial labeling L. We define C^+ to be the set of all correct labeling decisions involving active features and (S^+, l^+) to be the least cost decision in C^+. We also define C^- to be all incorrect labeling pairs that have a smaller cost than (S^+, l^+), i.e. the incorrect pairs that are preferred over all correct pairs. The weight vector is updated using the following Perceptron-style rule:

$$w = w + \sum_{(S,l) \in C^-} \frac{\Psi(S, l, L)}{|C^-|} - \sum_{(S,l) \in C^+} \frac{\Psi(S, l, L)}{|C^+|}$$

This rule updates w so as to increase the average cost of all pairs in C^- (high-ranking incorrect pairs) and decrease the average cost of the correct pairs in C^+.

Convergence of Learning. We analyze the realizable learning setting, where it is possible to find a weight vector that correctly labels all of the training data. In particular, as for most Perceptron-style learning algorithms, we bound the number of mistakes made during the training process until the weight vector correctly labels all training data. We must assume here that the inequality constraints for each training example are consistent with the ground truth labeling, since otherwise convergence would not necessarily be possible.

As for the classic Perceptron algorithm, our convergence guarantee is in terms of a notion of margin. Given a set of training videos let $\mathcal{L}$ be the set of partial labelings that are consistent with the inequality constraints. Given a constraint set Σ, the Σ-*constrained margin* of a weight vector w for a training set is the minimum over all $L \in \mathcal{L}$ of $w \cdot \Psi(S^-, l^-, L) - w \cdot \Psi(S^+, l^+, L)$, where (S^+, l^+) is any correct candidate labeling and (S^-, l^-) is any incorrect candidate labeling. If w has a positive constrained margin then using it within CSL with the constraints will correctly label the training set. We now show that the existence of a positive margin w is sufficient for convergence. Below R is a constant such that for all possible feature vectors $\Psi(S, l, L)$ and $\Psi(S', l', L)$ we have $\|\Psi(S, l, L) - \Psi(S', l', L)\| \leq R$

Theorem 1 *Given constraints Σ and any training set such that there exists a weight vector w with Σ-constrained margin $\gamma > 0$ and $\|w\| = 1$, CSL-Learn will converge to a weight vector that correctly labels all training examples after making no more than $(R/\gamma)^2$ weight updates.*

Thus a larger margin implies a better mistake bound. The proof is presented in the supplementary material. We now relate constraints to margins.

Proposition 1 *For any constraint sets Σ and Σ', such that $\Sigma \subseteq \Sigma'$, the Σ'-constrained margin of any weight vector w is no less than the Σ-constrained margin. Furthermore there exists a training set and constraint sets $\Sigma \subseteq \Sigma'$ such that there is a weight vector w with positive Σ'-constrained margin and there is no weight vector with positive Σ-constrained margin.*

Combining the above results we can see the utility of using constraints in learning. First, adding constraints will never decrease the margin and often increase it, which means the mistake bound will never get worse and often improve, suggesting learning is made easier. Second, there are problems where constraints are necessary to obtain a positive margin and hence guarantee convergence.

5. Representing and Learning Constraints

Our goal is to learn a *constraint generator* that returns the inequality constraints over mid-level features in a video. The constraints should have high precision in the sense that if $S_i \neq S_j$ is generated then it agrees with the ground truth with high probability. Further, we would like the constraint set to be as large as possible, while maintaining high precision, in order to maximize constraint propagation.

Our constraint generation is similar in spirit to prior work that used constraints between point trajectories [9]. In that work, the connected components of each video frame were computed and an inequality constraint was included between trajectories if they belonged to different components. We found that for our features that approach often produces erroneous constraints because it is not unusual in our videos for a single object to span multiple connected components in a frame. This occurs, for example, due to imperfections in our foreground saliency mask.

Fortunately, it is often still the case that when an object does span multiple components, those components will satisfy certain spatial layout properties. For example, for people tracking, if we consider a bounding polygon for two components, certain types of bounding polygons are unlikely to correspond to a single individual. Based on this insight, we train an SVM classifier to predict whether two components in a frame correspond to different objects. To do this we use labeled training data to extract pairs of components across the video frames and label them a positive if the components correspond to different objects and otherwise assign a negative label. We then compute features of the pairs and train the SVM classifier. In our experiments we found that a simple set of four features were sufficient for achieving good results, including: the horizontal and vertical distance between component centers and the height and width of the minimal bounding box containing the components. Given the learned SVM classifier we then adjust its prediction threshold so that it achieves high precision.

Given this classifier we then generate a feature inequality constraint $S_i \neq S_j$ if in some video frame S_i and S_j are contained in components that the SVM judges are from different objects. On our test data, this approach can achieve a precision of over 97% in challenging volleyball videos while maintaining a recall of over 44%, which gives very informative constraint sets. Figure 3 illustrates an example of the constraint generation process.

6. Experimental Results

Datasets. We evaluate using benchmark datasets of volleyball and basketball videos that involve the challenges noted in the introduction. Also to compare to state-of-the-art detection-based approach, we report results on the widely used pedestrian benchmark video PETS2009-S2L1 [8] where people detectors are effective. The Volleyball dataset [1] contains 38 videos of entire collegiate volleyball plays (each 200–800 frames, 720×480). Volleyball videos are very challenging, because of inter-occlusions of many

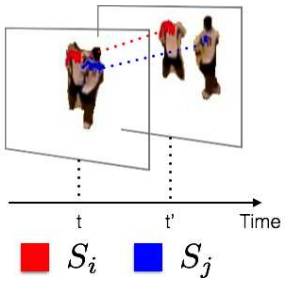

Figure 3: Illustration of determining whether two supervoxels can have the same label—inequality constraint. In frame t, S_i and S_j belong to a single foreground connected component, so the frame does not yield an inequality constraint. In frame t', the SVM classifier decides the two connected components cannot belong to a single individual, generating an inequality constraint between S_i and S_j.

players in different body articulations, and non-smooth fast motions. Ground truth bounding boxes are provided for the six near-team players, while opposing players are not annotated or used for evaluation. We extended the ground truth to include finer-level annotations every 5 frames in terms of player pixel masks which exactly delineate each player in these frames. We train and test via 3-fold cross-validation reporting averages across folds. The Figment dataset [9] contains 18 videos of basketball action (each 50–80 frames, 441×180). Ground truth player masks are provided every 7-8 frames. We used leave-one-out cross-validation for evaluation. For PETS2009-S2L1 training of the cost function and constraints was done using the PETS2009-S2L2.

Implementation Details. We tested CSL using both supervoxels (CSL-VOX) and dense point trajectories (CSL-DPT). Supervoxels were generated as the leaf-level supervoxels of the hierarchical video segmentation approach of [20]. The only input parameter for CSL-VOX is the total number of supervoxels, which is controlled by varying the video segmentation parameters. For generating dense point trajectories we followed [9]. For Volleyball we obtain foreground features via background subtraction based on a Gaussian-mixture color model. For the basketball videos, we follow [9] and extract foreground features based on on motion saliency of dense point trajectories. For PETS2009-S2L1, we use a trained background model together with motion saliency to estimate the foreground features. To fairly compare with detection-based approach on PETS2009, we use detection outputs to initialize in the first frame rather than manual initialization.

Evaluation Metrics. We use the standard CLEAR MOT evaluation metrics: miss detection (MD), false positives (FP), ID switches (ID-sw.), and accuracy (acc). Also, to compare to published results [9] on Figment we use the metrics from that work: per object clustering error (PRCE), recall, and tracking time.

Baselines. For Volleyball, we created baselines from common frameworks: 1) *NCuts* [16] with different numbers of clusters 6 (the true number of clusters) and 12, and pairwise affinities computed in terms of color, space-time location and optical flow of supervoxels. NCuts also uses the same responses of our inequality constraints classifier which are incorporated as zero-valued entries in the affinity matrix, 2) Detection-based *Network Flow* [14] using the publicly available code that applies flow-based linking to people detections. We trained a people detector in our domain and used the resulting detections as input. We also updated the code to use appearance-based affinities, which improves results, 3) *Network Flow**, the previous network flow approach, but applied to ground truth bounding boxes, rather than real detections. This is an oracle baseline (since perfect detections are used) intended to estimate idealized performance with perfect input, 4) *Superpixel* uses our CSL approach on frame-based segments (intersections of supervoxels with frames), instead of temporal supervoxels. For the other two datasets, we compare against the reported state-of-the-art results.

Quantitative Results. Tab. 1 shows the Volleyball results. First, we observe that both CSL-VOX and CSL-DPT significantly outperform all non-Oracle approaches in all metrics. Further we see that CSL-VOX outperforms CSL-DPT by a small margin, primarily because the supervoxels provide more stable affinity estimation. The significant improvement over Superpixel shows the utility of using the more coherent temporal mid-level feature rather than frame-based features within CSL. Using the temporally extended features allows for more significant constraints and less shortsighted labeling. The comparison to Network Flow shows that a state-of-the-art detection based technique faces serious challenges in our domain due to the difficulty in obtaining accurate enough detections. Rather, by using mid-level features, the CSL approaches are less vulnerable to occlusions. Surprisingly, the CSL approaches are comparable to the oracle Network Flow* approach, which is allowed to cheat and use ground truth detections. Notably, CSL is significantly better in terms of ID switches. Note that the oracle approach will necessarily achieve perfect MD and FP scores due to the use of perfect detections.

Tab. 2 shows that on Figment CSL outperforms two state-of-the-art methods [9, 5] based on graph partitioning of dense point trajectories with a variety of post-processing steps. We hypothesize that one reason for this is that the prior approaches relax hard constraints as affinity in the similarity matrix, which can result in non-sensical partitions that must be heuristically post-processed. CSL-VOX again outperforms CSL-DPT suggesting that supervoxels are a more effective mid-level feature in this domain.

Tab. 3 shows results of CSL and two top-performing detection-based methods on PETS2009S2L1. CSL achieves

Method	MD	FP	ID-sw.	Acc
CSL-VOX	2.69	2.49	0.41	94.41
CSL-DPT	5.32	4.67	0.87	89.14
NCut(6)	65.45	65.15	34.84	-65.44
NCut(12)	43.63	80.30	35.03	-58.93
Superpixel	5.15	5.15	37.27	52.43
Network Flow	56.36	3.33	0.91	39.40
Network Flow*	0	0	3.64	96.36
Uniform	2.99	2.89	0.45	93.67
SL	37.58	36.97	6.06	19.39
Uniform-SL	63.03	63.03	13.03	-39.09
CSL-noProp	20.91	20.60	16.06	42.43

Table 1: VolleyBall dataset. Network Flow* uses ground truth detections.

Method	PRCE	Recall	Tracking time
CSL-VOX	17.10	82.89	89.73
CSL-DPT	19.28	43.62	79.41
[9]	20.32	31.07	75.13
[5]	86.42	0.46	1.03

Table 2: Results for the Figment dataset. PRCE: percentage of wrongly labeled pixels per player mask; Recall: percentage of recalled pixels per player mask; Tracking time: the number of frames where recall is above 20%.

Method	Rec.	Prec.	ID-sw.	MOTA
CSL-VOX	98.28	91.07	6	89.78
CSL-DPT	97.64	90.45	8	88.13
[11]	94.03	92.40	10	84.77
[22]	96.45	93.64	8	90.3

Table 3: Results for PETS2009 S2L1

comparable performance, showing that even in domains where detections are more reliable, CSL is still competitive. The slightly worse precision performance of CSL is primarily due to the imperfect constraint classifier, which sometimes rules out all labels for certain supervoxels, which are then assigned new labels.

Sensitivity Analysis. Tab. 4 evaluates sensitivity to the only input parameter – the number of mid-level features (e.g. supervoxels in this case) – on the Volleyball dataset. As expected, increasing supervoxels improves accuracy and increases runtimes.

Tab. 1 also evaluates the importance of learning and using constraints using several CSL variants (all using supervoxels): 1) *Uniform* is CSL with no weight learning and setting all cost function weights to one, 2) *SL* is pure sequential labeling (CSL without constraints) with weight learning, 3) *Uniform-SL* is SL with no weight learning and setting all weights to one, 4) *CSL-noProp* is CSL without constraint propagation, but does use constraints to assign infinite costs to labelings that immediately violate a constraint.

Number of supervoxels	accuracy	time
500	-96.56	23s
1000	-91.20	76s
1500	84.66	124s
2000	92.28	291s

Table 4: Results of tracking accuracy and running time of our approach with different number of supervoxels as input for videos with 300 frames.

We see that constraints and constraint propagation are crucial. In particular, SL (no constraints) is significantly worse than CSL. Further, CSL-noProp is able to use constraints to improve over SL, however, it is not nearly as effective as CSL, which uses propagation. A second observation is that while weight learning is able to improve performance (CSL vs. Uniform), we see that the improvement is small in this case, which is likely due to the fact that our features are designed to be quite informative and also that the constraints play a dominating role leaving little room for improvement. However, we see that the learning algorithm is quite effective when there is room to improve. In particular, we see that learning without constraints (SL) is much better than no learning without constraints (Uniform-SL).

Runtime. The average runtime of generating supervoxels and dense point trajectories is about 28s/frame and 24s/frame respectively. Given mid-level features, Tab.4 shows that the runtime for CSL is quite fast. Further, CSL is approximately 10x faster when using constraint propagation compared to no propagation. This is because constraint propagation sets the label of approximately 9/10 of the supervoxels avoiding that many cost function evaluations.

Qualitative evaluation: Fig. 4 illustrates tracking results for CSL and the baselines on a Volleyball video where the orange player is moving left, the yellow player is moving right, and the purple player stays still. Only our method gives correct tracks in this case. NCuts is sensitive to the choice of the number of clusters. Also, NCuts may produce solutions that violate the hard constraints, since they are "softened" in the affinity matrix. Network flow* and Superpixels confused the ID's of the players on the two sides. More tracking results including some failure cases can be found in our supplementary material.

7. Conclusion

We presented a new approach for multi-object tracking under significant occlusion based on labeling mid-level, temporal features such as supervoxels. This labeling problem relaxes common assumptions of existing data-association methods. One of our main contributions is the constrained sequential labeling (CSL) approach, which leverages hard constraints and a flexible cost function to perform accurate labeling. We provided learning algorithms

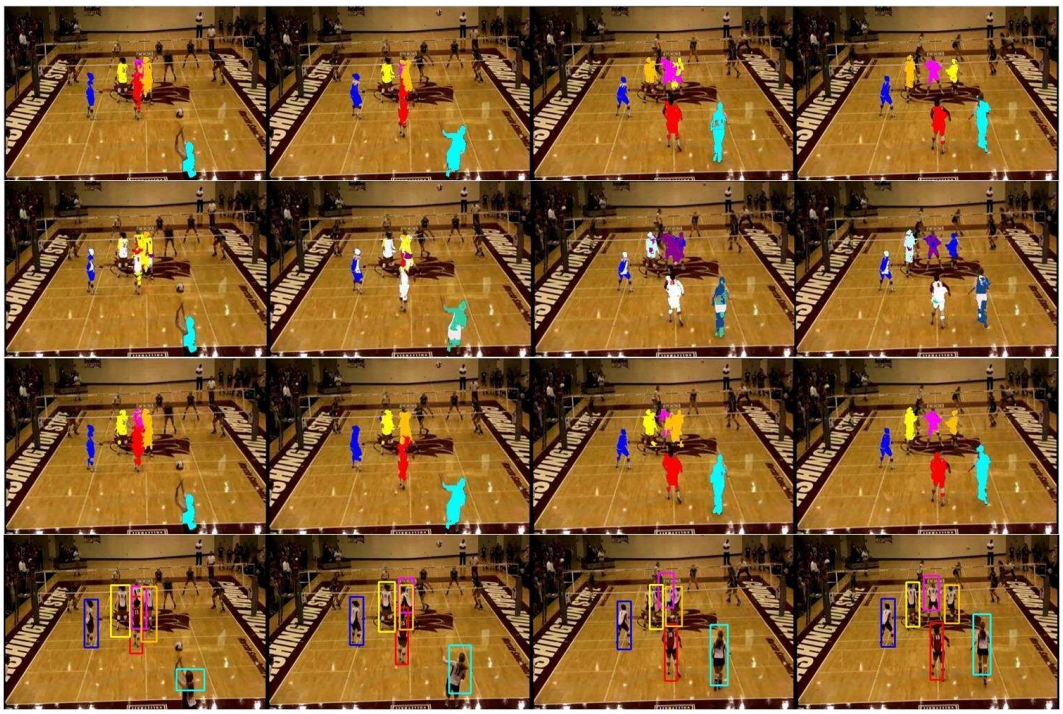

Figure 4: Volleyball dataset: first row(CSL-VOX), second row(Ncut), third row(Superpixel), forth row(Network Flow*)

for constraints and cost functions and proved that cost-function learning converges to an accurate solution in finite-time when such a solution exists. Our experimental results in volleyball, basketball, and pedestrian tracking demonstrate that the approach is superior to the state-of-the-art when people detectors are unreliable and comparable to the state-of-the-art even when detections are accurate.

Acknowledgment

This work was supported in part by NSF grants IIS 1219258, IIS 1018490 and DARPA MSEE FA 8650-11-1-7149.

References

[1] M. Amer and S. Todorovic. Sum-product networks for modeling activities with stochastic structure. In *CVPR*, 2012.

[2] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *CVPR*, 2008.

[3] A. Andriyenko, K. Schindler, and S. Roth. Discrete-continuous optimization for multi-target tracking. In *CVPR*, 2012.

[4] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. In *PAMI*, 2011.

[5] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010.

[6] R. Dechter. *Constraint processing*. Morgan Kaufmann, 2003.

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 32(9):1627–1645, 2010.

[8] J. Ferryman and A. Shahrokni. Pets2009: Dataset and challenge. In *Performance Evaluation of Tracking and Surveillance, International Workshop*, 2009.

[9] K. Fragkiadaki and J. Shi. Detection free tracking: Exploiting motion and topology for segmenting and tracking under entanglement. In *CVPR*, 2011.

[10] Hal Daumé III, J. Langford, and D. Marcu. Search-based structured prediction. *MLJ*, 75(3):297–325, 2009.

[11] J. Henriques and J. Caseiro R., Batista. Globally optimal solution to multi-object tracking with merged measurements. In *ICCV*, 2011.

[12] Y. Huang and I. Essa. Tracking multiple objects through occlusions. In *CVPR*, 2005.

[13] Y. Li, C. Huang, and R. Nevatia. Learning to associate: Hybrid-Boosted multi-target tracker for crowded scene. In *CVPR*, 2009.

[14] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, 2011.

[15] Z. Qin and C. R. Shelton. Improving multi-target tracking via social grouping. In *CVPR*, 2012.

[16] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE TPAMI*, 22(8):888–905, 2000.

[17] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah. Part-based multiple-person tracking with partial occlusion handling. In *CVPR*, 2012.

[18] B. Wu and R. Nevatia. Tracking of multiple, partially occluded humans based on static body part detection. In *CVPR*, 2006.

[19] J. Xing, H. Ai, and S. Lao. Multi-object tracking through occlusions by local tracklets filtering and global tracklets association with detection responses. In *CVPR*, 2009.

[20] C. Xu, C. Xiong, and J. J. Corso. Streaming hierarchical video segmentation. In *ECCV*, 2012.

[21] B. Yang and R. Nevatia. Online learned discriminative part-based appearance models for multi-human tracking. In *ECCV*, 2012.

[22] A. R. Zamir, A. Dehghan, and M. Shah. Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs. In *ECCV*, 2012.

[23] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *CVPR*, 2008.