Topology-Constrained Layered Tracking with Latent Flow

Jason Chang*
CSAIL, MIT
jchang7@csail.mit.edu

John W. Fisher III*
CSAIL, MIT
fisher@csail.mit.edu

Abstract

We present an integrated probabilistic model for layered object tracking that combines dynamics on implicit shape representations, topological shape constraints, adaptive appearance models, and layered flow. The generative model combines the evolution of appearances and layer shapes with a Gaussian process flow and explicit layer ordering. Efficient MCMC sampling algorithms are developed to enable a particle filtering approach while reasoning about the distribution of object boundaries in video. We demonstrate the utility of the proposed tracking algorithm on a wide variety of video sources while achieving state-of-the-art results on a boundary-accurate tracking dataset.

1. Introduction

Tracking is a fundamental task in video sequence analysis. The resulting tracks can be used to analyze past behavior and predict future trajectories of objects in the scene (e.g. [6]). Segmentation and motion analysis of video provides a preprocessor for object classification. Accurate object boundaries enable methods for learning shape models (e.g. [23]). We focus on object tracking with accurate boundaries in contrast to bounding box methods. Figure 1 illustrates the difference between the two. The resulting probabilistic model for layered object tracking combines dynamic appearance and shape models, topology constraints, and Gaussian process flow. These concepts have been considered individually in a variety of contexts including tracking. Here, we consider an integrated model and develop efficient sampling-based algorithms. We obtain state-of-the-art results on the SegTrack dataset [29]. Furthermore, we also formulate a novel shape sampler that addresses a flaw in the formulation of [5] and is directly applicable to general segmentation problems.

Elements of the approach are certainly related to previous work. Layered models have been popular since the Bayesian model of [8], with the promising results of [31] motivating many similar approaches. The layered appearance model described herein is closely related to [15, 24], but as described in Section 2, differs explicitly by coupling occlusions and disocclusions with the layer supports. Similarly, Section 3 discusses how our flow model generalizes that of [32] by using Gaussian processes.

Figure 1: Bounding box tracking versus boundary accurate tracking. The object boundaries were found using the algorithm described in this paper.

Owing to the wealth of literature in tracking and segmentation, we focus on relevant prior work. Some optimization-based approaches, such as [11, 18, 21], are able to automatically *segment* objects of interest by processing the entire video offline. Others (e.g. [29]) require annotations to identify key objects or frames. Such batch processing is akin to Bayesian *smoothing*, where inference depends on both past and future observations. Alternatively, analogous to Bayesian *filtering*, one can *track* objects in an online fashion (e.g. [23, 24, 25]). The proposed method adopts a filtering approach, scaling linearly in the number of frames for computation, and having constant memory consumption.

There is also a rich literature in probabilistic tracking, most of which track bounding box trajectories instead of accurate boundaries. A notable exception is [23] which uses an unconventional particle filter that propagates particles with a number of gradient descent iterations instead of a randomized proposal. We show that this deterministic approximation is unnecessary. Additionally, the dynamics do not correspond to an underlying object motion, and therefore, do not couple the changes in appearance and shape.

To our knowledge, only [25] and [10] (optimization-based methods) consider topology constraints in video analysis. The first uses digital topology to penalize merges of two connected components (each corresponding to a sepa-

*Jason Chang was partially supported by the Defense Advanced Research Projects Agency, award FA8650-11-1-7154. John Fisher was partially supported by the Office of Naval Research Multidisciplinary Research Initiative (MURI) program, award N000141110688.

rate object). This soft constraint does not, however, necessarily preserve connected components. Similarly, the second extends binary topology to M-ary topology to restrict objects from merging. This method does not use an appearance or flow model and also does not handle occlusions that may split the visible support of objects.

Similiar to the proposed method, Sun et al. [27, 28] estimate motion and reason about depth ordering in a layered model. In fact, [27] extends [28] to infer the *number* of layers, something the proposed method does not consider. In contrast to the proposed method, these methods segment objects via batch processing of the entire sequence. Unlike our method which reasons over the shape *distribution*, [27] adopts an energy minimization approach within a graphical model formulation. While [28] achieves state-of-the-art optical flow results, it does so at considerable computational expense. Furthermore, they do not allow for temporal evolution of layer ordering. Processing times on current standard computer hardware are on the order of hours per frame as compared to approximately one minute per frame for the proposed approach.

2. Layered Model

We begin by developing the probabilistic model, depicted as a directed graph in Figure 2a, used to represent scenes. While it is important for any practical method to be able to detect new objects entering a scene, for purposes of exposition, we restrict ourselves to the case of tracking M previously detected objects. The method of initialization is discussed in Section 6.

2.1. In-Frame Appearance

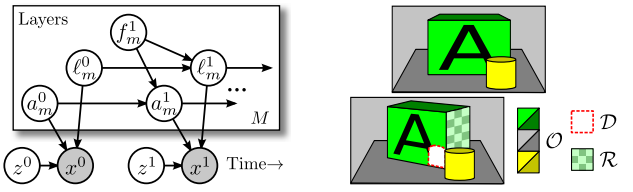
Scene models are comprised of M layers, where each object lies in one layer, and one layer is the designated background. The support of layer m at time t is denoted by $\ell_m^t \in \{0,1\}^N$, where N is the number of pixels. Specifically, $\ell_{m,i}^t = 1$ iff pixel i is in the support of layer m at time t. The layers are ordered with z^t, which contains a permutation of the integers 1 to M. The visible layer at pixel i, denoted v_i^t, can then be expressed as

$$v_i^t = \arg\min_{\{m|\ell_{m,i}^t=1\}} z_m^t. \tag{1}$$

Associated with each layer is a pixel-wise appearance model, $a_m^t \in \mathbb{R}^{N\times 3}$, where $a_{m,i}^t$ is the 3-dimensional color (we use the *Lab* colorspace) at pixel i. Assuming Gaussian observation noise, the observed image, x^t is generated by

$$x_i^t | a^t, \ell^t, z^t \sim \mathcal{N}(a_{v_i^t,i}^t, \Sigma_X). \tag{2}$$

We note that the visible pixel, v_i^t, is implicitly dependent on z^t through Equation 1. We assume the color channels are independent, *i.e.* Σ_X is diagonal. While [15] and [24] use similar models, they do not address the appearance of occluded or disoccluded pixels which we now discuss.



(a) Graphical Model (b) Types of Pixels

Figure 2: (a) Graphical model: x denotes a frame, ℓ_m, a_m, and f_m are the support, appearance, and flow for layer m, and z controls the layer order. (b) An example of the three types of pixels that can occur in a new frame.

2.2. Temporal Appearance Dynamics

Given the appearance and support of a previous frame, the dynamics of these random variables evolve jointly based on the underlying motion of the layer. We describe the motion model for layer m from frame $(t-1)$ to t, denoted f_m^t, in Section 3. The motion is related to optical flow, and we use the terms "motion" and "flow" interchangeably. While deformations of 3D objects are typically diffeomorphic, projections onto the 2D image plane are not because of occlusions and disocclusions. For example, consider the illustration in Figure 2b where a visible pixel in the new frame can belong to three possible portions of a layer: (1) observed portion, $\mathcal{O}$, which has been seen before; (2) disoccluded portion, $\mathcal{D}$, which has never been seen and was previously occluded by another layer; or (3) revealed portion, $\mathcal{R}$, which has never been seen and was previously hidden by a pixel belonging to the same layer. Consequently, we define a probability distribution over the evolving appearance model for each of these categories.

Observed portions, $\mathcal{O}_m^t$, evolve with a Gaussian distribution from the aligned appearance model

$$p(a_{m,i}^t | i \in \mathcal{O}_m^t, fa_m^{t-1}) = \mathcal{N}(a_{m,i}^t; fa_{m,i}^{t-1}, \Sigma_A), \tag{3}$$

where fa_m^{t-1} denotes the aligned appearance obtained by evolving a_m^{t-1} with the flow f_m^t, and $fa_{m,i}^{t-1}$ indexes pixel i from the image fa_m^{t-1}. Similar to the observation, we assume independent color channels with a diagonal Σ_A.

Disoccluded portions are often similar to neighboring pixels. For example, the disoccluded pixels in Figure 2b are likely to be green or black. Because no additional prior information is given, these pixels are drawn from

$$p(a_{m,i}^t | i \in \mathcal{D}_m^t, fa_m^{t-1}) \propto \sum_{j\in\mathcal{O}_m^{t-1}} \mathcal{N}(j; i, \sigma_{\mathcal{D}}^2 \mathrm{I}) \delta\left(a_{m,i}^t - fa_{m,j}^{t-1}\right), \tag{4}$$

where $\mathcal{N}(j; i, \sigma_{\mathcal{D}}^2 \mathrm{I})$ is a 2D Gaussian over pixel coordinates.

Lastly, revealed portions appear when objects turn and reveal a new side. These can look quite different from neighboring pixels. For example, the revealed side of the

box in Figure 2b can be of any color. Appearances in $\mathcal{R}_m^t$ are therefore drawn from a mixture of a uniform distribution and a kernel density estimate of the observed appearances

$$p(a_{m,i}^t | i \in \mathcal{R}_m^t, a_m^{t-1}) = (1-\alpha_R)\mathcal{U}^3(a_{m,i}^t) + \alpha_R k(a_{m,i}^t),$$
$$k(a) = \frac{1}{|\mathcal{O}_m^{t-1}|} \prod_d \sum_{j \in \mathcal{O}_m^{t-1}} \mathcal{N}(a_d; a_{m,j,d}^{t-1}, \sigma_k^2), \quad (5)$$

where $\mathcal{U}^3$ is the uniform distribution over the colorspace, d indexes a color. $k(\cdot)$ can be estimated with [33] and σ_k^2 is chosen to be the "rule-of-thumb" bandwidth. While these simple assumptions do not explain all situations, we have empirically found that they work well in most videos.

2.3. Temporal Support Dynamics

The evolution of each layer support is coupled to the evolution of its appearance. However, by introducing disoccluded and revealed pixels, we must also allow the support of each layer to deviate from the aligned support, $f\ell_m^{t-1}$. Consequently, layer supports are chosen to evolve according to a distribution proportional to exponentiated symmetric area difference (SAD), where SAD can be expressed as

$$\text{SAD}(\ell^1, \ell^2) = \sum_i \mathbb{1}\left[\ell_i^1 \neq \ell_i^2\right], \quad (6)$$

and $\mathbb{1}[\cdot]$ is one iff $[\cdot]$ is true. Additionally, we impose a curve length penalty on the shape of each layer and enforce each layer to be a single connected component. The resulting temporal dynamics on layer support are expressed as

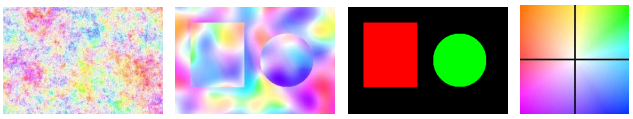
$$p(\ell_m^t | \ell_m^{t-1}, f_m^t) \propto Q_L(\ell_m^t) \prod_i Q_S(\ell_{m,i}^t | f\ell_{m,i}^{t-1}), \quad (7)$$
$$Q_L(\ell_m^t) = \mathbb{1}\left[T(\ell_m^t) = 1\right] \exp\left[-\alpha_L \mathcal{C}_m^t\right], \quad (8)$$
$$Q_S(\ell_{m,i}^t | f\ell_{m,i}^{t-1}) = \exp\left[-\alpha_S \mathbb{1}\left[\ell_{m,i}^t = f\ell_{m,i}^{t-1}\right]\right], \quad (9)$$

where $\mathcal{C}_m^t$ is the contour length of ℓ_m^t, $T(\cdot)$ counts the number of connected components, and the α_L, α_S control the relative weighting of these penalties.

The appearance and shape model we describe will not explain every situation. For example, an object that splits into two will violate the topology prior of the model. If a light is turned on, the Gaussian diffusion of appearances may be too restrictive. However, in subsequent sections we show that they yield good empirical performance across a variety of video sequences.

3. Gaussian Process Flow

Having detailed the observation model *conditioned* on the layered flow fields, we describe a flow model comprised of layered Gaussian processes (GPs). A GP can be parametrized with a mean and covariance function (c.f. [22]). We restrict the model to zero-mean GPs with stationary covariance kernels which, as shown in Section 5, enables an efficient sampling-based inference method.



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

Figure 3: Samples from different GPs: (a) sparse neighborhood precision; (b) SE kernel with layered composition using the layers of (c). (d) Flow vectors mapping to colors.

GPs have been widely used as a prior for trajectories (e.g. [2, 16, 30]). In practice, however, these formulations have been applied to object trajectories, whereas here we consider their application to dense flow between frames. GPs are not typically used to model flow for two reasons: (1) GPs are often smooth everywhere, causing errors across object boundaries; and (2) naïve inference requires inverting covariance matrices that do not scale well with the image size. We address the first issue here and the second via an approximation in Section 5.

One particular GP covariance kernel is closely related to the L_2 penalty in the Horn-Schunck optical flow formulation [13]. However, that L_2 penalty on flow *differences* results in an improper distribution with a rank deficient covariance matrix. Regardless, one can approximate this prior with a GP having a sparse precision matrix (inverse of the covariance), arising from the 4-connected neighborhood of each node. Figure 3a shows a sample from this type of GP, which fails to capture long range correlations in flow fields and, concurrently, overly penalizes discontinuities across object boundaries. Following [32], we instead compose smooth, layered flows using

$$\overline{f}_i = f_{v_i,i} \qquad \forall i \in \{1, \cdots, N\}. \quad (10)$$

Here, f_m is the flow for layer m, and $\overline{f}$ is the composite flow for visible pixels. Similar to optical flow, we assume that the x and y components of flow are independent. Figure 3b shows a sample from a GP using Equation 10 with the squared exponential (SE) kernel (c.f. [22]). Composing multiple smooth GPs in this fashion mitigates the problem of discontinuities.

Unlike [32] which uses layered flows of the form of [13], a GP flow is a valid prior distribution that can capture long-range dependency. While any 2D GP has an equivalent Gaussian Markov random field (GMRF). The associated graphical structure is fixed and depends on the covariance kernel. The resulting GMRF typically has complex connectivity impacting inference. However, by utilizing a GP formulation, we show in Section 5 that inference is easily adapted to changes in the covariance.

3.1. Smooth Deformable Flow

Tracking deformable objects encounters additional complexity when parts of the objects exhibit self-occlusions or

Figure 4: Consecutive frames of a deformable object with self occlusions and disocclusions.

disocclusions (e.g. the arms and legs of the girl in Figure 4). The SE kernel often results in overly smoothed flow estimates that do not adequately capture object deformation. We mitigate this issue by adopting a covariance kernel that is the composition of the SE kernel and a delta function. It is easily verified that the resulting GP follows the distribution

$$p(f_m) = \mathcal{N}(f_m \; ; \; 0, \Sigma_g + \sigma_f^2 \mathrm{I}), \tag{11}$$

where Σ_g is the resulting covariance from the SE kernel, and $\sigma_f^2\mathrm{I}$ is the resulting covariance from the delta kernel. Notation is slightly abused since each component of the flow is not explicitly written. We show in Section 5 how to exploit a decomposition of this flow for efficient inference. In particular, we decompose the flow into the smooth flow (denoted g_m) and the remaining independent part with

$$p(g_m) = \mathcal{N}(g_m \; ; \; 0, \Sigma_g), \tag{12}$$

$$p(f_m|g_m) = \mathcal{N}(f_m \; ; \; g_m, \sigma_f^2 \mathrm{I}). \tag{13}$$

4. Shape Sampling

Having developed a dynamic appearance and shape model, we present a Bayesian *filtering* procedure that reasons about the distribution of the hidden variables conditioned on past and current observations. Exact inference in such complex models is generally intractable. A typical approach is to use a particle filter [14], where the distribution at any time is represented by a set of weighted samples. Particles, generally propagated by the prior dynamics, often suffer from "weight decay" resulting in a poor representation. Sequential resampling techniques are often utilized to address this issue. An alternative is to propagate particles with observed data information (e.g. [7, 9, 23]). Unfortunately, many of these methods introduce additional approximation errors and/or suffer from slow convergence.

Here, we propose a more accurate particle propagation approach by incorporating both the prior and the data likelihood terms. In this case, similar to a Gibbs sampler, weight updates are unnecessary because samples are drawn from the true conditional distribution. A formal derivation is provided in the supplement. Though more accurate, this approach is typically avoided because sampling from the full conditional distribution can be computationally prohibitive. A key enabler is our extension of [5] to efficiently sample the support of shapes. The overall inference algorithm is presented in Section 5 following this discussion.

4.1. Gibbs-Inspired Metropolis Hastings

Markov chain Monte Carlo (MCMC) techniques are often used when direct sampling from a target distribution, $\pi(\ell)$, is difficult. MCMC samplers construct a transition distribution (often referred to as the proposal distribution) such that the stationary distribution of the chain is the target distribution. The Metropolis-Hastings (MH) algorithm [12] enforces the correct stationary distribution by generating $\hat{\ell}^{(t)}$ from a user-specified proposal distribution, $q(\hat{\ell}^{(t)}|\ell^{(t-1)})$, and accepting the transition with probability

$$\Pr\left[\ell^{(t)} = \hat{\ell}^{(t)}\right] = \min\left[1, \frac{\pi(\hat{\ell}^{(t)})}{\pi(\ell^{(t-1)})} \cdot \frac{q(\ell^{(t-1)}|\hat{\ell}^{(t)})}{q(\hat{\ell}^{(t)}|\ell^{(t-1)})}\right], \tag{14}$$

where (t) indexes the chain. Otherwise, the old sample is kept. Under mild technical conditions, the resulting ℓl will be a sample from $\pi(\ell)$.

The Gibbs-Inspired Metropolis Hastings shape sampler (GIMH) [5] uses the MH-MCMC algorithm. GIMH represents the support of a region using a level set function. It assumes that the target distribution only depends on the sign of the level set function and can be expressed as a product of a prior and independent likelihoods:

$$\pi(\ell) = p(\ell) \prod\nolimits_i p(x_i|\ell_i) \tag{15}$$

By using a look-up table for calculating local changes in curve length, GIMH achieves extremely fast burn-in times and can control the topology of the resulting sample. Both of these aspects are directly applicable to the prior we place on layers described in Section 2.

GIMH generates a proposal by adding a random constant to the level set function in a random subset of pixels. Because of the ordering implied by the level set function, only a linear number of possible configurations in the size of the subset is possible (as opposed to the exponential number of total configurations). A proposal distribution is chosen such that Equation 14 evaluates to 1 and every proposal is accepted. However, due to a minor flaw in the formulation GIMH does *not* preserve the correct stationary distribution. The following alternative construction, detailed in the supplement, resolves this issue while leading to computational savings.

4.2. Permutation-Based GIMH

Instead of implicitly ordering pixels with a level set function as in GIMH, we consider explicitly using a random total ordering, o, on all pixels. We now consider how to sample a binary label from this joint distribution. We provide an M-ary extension in the supplement.

We define a *consistent ordering* as an ordering that puts all pixels with $\ell_i = 0$ before pixels with $\ell_i = 1$. If the conditional distribution, $p(o|\ell)$, is chosen to be uniform over all consistent orderings, we show in the supplement that this

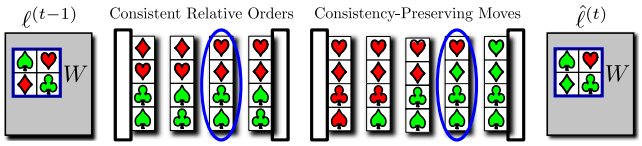


Figure 5: An example of the PGIMH proposals. A random window is chosen (left), $W = \{\spadesuit, \diamondsuit, \heartsuit, \clubsuit\}$, where suits indicate indices and colors indicate labels. A consistent relative order is selected at random. One of the $N_W + 1$ possible consistency-preserving moves is selected to produce $\ell^{(t)}$.

ordering does not need to be instantiated. In fact, the flaw of [5] is resolved via a simple sampling procedure while preserving the correct stationary distribution.

Similar to GIMH, we change a random subset of pixels, W at each iteration. Conditioned on W, a *relative ordering* is defined to be an ordering on the pixels only in W. We note that a relative ordering derived from a consistent total ordering must also be consistent. Randomly changing the labels of pixels within W will, in general, not preserve the consistency of the relative ordering. However, there are exactly $N_W + 1$ consistency-preserving changes, where $N_{(\cdot)} = |(\cdot)|$. These observations lead to the development of the Permutation-based GIMH (PGIMH) sampler, detailed in Algorithm 1. Here, W_l denotes the subset of W with label l, and $\hat{W}_l$ denotes the proposed subset with label l. Figure 5 illustrates an example proposal from PGIMH.

Algorithm 1 An iteration of sampling $\pi(\ell)$ via PGIMH

1. Randomly sample a subset of pixels, W, by selecting a circle with random location and radius
2. Sample a consistent relative ordering of pixels in W uniformly using a Knuth shuffle [17] on each W_l
3. Enumerate the $N_W + 1$ consistency-preserving moves
4. Sample $\ell^{(t)}$ from the possible moves according to

$$q(\hat{\ell} \,|\, \ell, W) \propto \pi(\hat{\ell}) \cdot \frac{1}{N_{\hat{W}_0}! N_{\hat{W}_1}!}$$

We note that the topology-controlled version of PGIMH extends straightforwardly from [5] by ignoring moves with invalid topologies. Additional details and derivations related to this section can be found in the supplement.

5. Inference

The development of the PGIMH method allows one to quickly sample layer supports from an arbitrary distribution. We now describe a sequential (in time) Gibbs sampler that utilizes PGIMH to perform tracking. As noted, this type of sampler can be viewed as a particle filter without weight updates. Joint inference of flow and layer supports often exhibits better convergence [27]. Our sampler marginalizes out part of the flow and, owing to our particular formulation, infers hidden variables on the order of one minute per frame as compared to [27] which takes hours per frame.

5.1. Single Layer Sampler

The sampler we use iterates over the following steps

$$g^t \sim p(g^t | f^t), \tag{16}$$
$$\ell^t \sim p(\ell^t | g^t, \ell^{t-1}, a^{t-1}, x^t, z^t), \tag{17}$$
$$f^t \sim p(f^t | g^t, \ell^t, \ell^{t-1}, a^{t-1}, x^t, z^t), \tag{18}$$
$$z^t \sim p(z^t | f^t, \ell^t, a^{t-1}, x^t). \tag{19}$$

Sampling from $g^t | f^t$ is equivalent to sampling a GP with observations. With some manipulation from the typical GP regression (c.f. [22]), this distribution can be expressed as

$$g^t_m | f^t_m \sim \mathcal{N}(\mu^*_g, \Sigma^*_g), \tag{20}$$
$$\mu^*_g = \Sigma[\Sigma + \sigma_f^2 \mathrm{I}]^{-1} f^t_m \;, \;\; \Sigma^*_g = \Sigma - \Sigma[\Sigma + \sigma_f^2 \mathrm{I}]^{-1}\Sigma.$$

Sampling from this expression is difficult because of the dimension of the GP. By drawing on the work of [26], we show in the supplement that a GP with a stationary covariance kernel, $k(x - x')$, can be approximately sampled with

$$g^t_m | f^t_m \sim \left[h_\mu * f^t_m\right] + \mathcal{N}(0, \mathrm{I}) * h_\Sigma, \tag{21}$$
$$h_\mu = \mathcal{F}^{-1}\left\{\tfrac{K}{K+\sigma_f^2}\right\} \;, \; h_\Sigma = \mathcal{F}^{-1}\left\{\sqrt{K - \tfrac{K^2}{K+\sigma_f^2}}\right\}.$$

Here, K denotes the Fourier transform of k. We note that this approximation degrades closer to image boundaries.

Thousands of iterations of the PGIMH sampler in Algorithm 1 are used to sample the layer supports of Equation 17 in less than a second. As each iteration perturbs only a single layer, we refer to it as the "Single Layer Sampler". We randomly choose a layer, m, and sample its support by manipulating the following proportional distribution

$$\int p(f^t_m | g^t_m) p(\ell^t_m | f \ell^{t-1}_m) p(a^t_m | f a^{t-1}_m) p(x^t | \ell^t, a^t, z^t)\, df^t_m$$
$$= Q_L(\ell^t_m) \prod\nolimits_i \int p(f^t_{m,i} | g^t_{m,i}) \mathcal{L}^t_{m,i}(f^t_{m,i} - g^t_{m,i})\, df^t_{m,i} \tag{22}$$

where the pixel-wise likelihood term, $\mathcal{L}^t_{m,i}(j)$, is

$$Q_S(\ell^t_{m,i} | g \ell^{t-1}_{m,i+j}) p(a^t_{m,i} | g a^{t-1}_{m,i+j}) p(x^t_i | \ell^t_i, a^t_i, z^t), \tag{23}$$

and we have used the fact that

$$f(\cdot)^{t-1}_{m,i} = g(\cdot)^t_{m,i+f^t_{m,i}-g^t_{m,i}}. \tag{24}$$

We can *marginalize* f^t_m by approximating $p(f^t_{m,i} | g^t_{m,i}) = \mathcal{N}(f^t_{m,i} - g^t_{m,i}, \sigma_f^2)$ with a discrete FIR filter, h_f. The distribution over ℓ^t_m in Equation 17 is then proportional to

$$Q_L(\ell^t_m) \prod\nolimits_{i=1}^N \sum\nolimits_j h_f(j) \mathcal{L}^t_{m,i}(j), \tag{25}$$

which is of the form of Equation 15 and is efficiently sampled using Algorithm 1. Equation 25 is efficiently evaluated for h_f with small support. A detailed derivation is found in the supplement. Changing the m^{th} layer accomplishes one of the following: (1) grow and occlude the currently visible layer; (2) grow behind the visible layer; (3) shrink and disocclude the next layer; (4) shrink behind the visible layer. Each situations can be expressed with Equations 2-9. Furthermore, the Gaussian appearance dynamics and observation models allow for efficient marginalization of previously observed appearances. For disoccluded and revealed appearances, we draw a sample from the distributions in Equation 4-5. Similarly, f^t_m is approximately sampled (Equation 18) by multiplying Equation 25 by $\mathbb{1}\left[f^t_{m,i} = g^t_{m,i} + j\right]$.

We conclude a full iteration by sampling the layer ordering, z^t. Because of the uniform prior on orderings, Equation 19 is proportional to the observation likelihood which can be computed using Equations 2-9. The videos we analyze typically have fewer than ten objects. As such, it is efficient to simply enumerate all possible total orders. When the number of layers is larger, a swap proposal in a Metropolis-Hastings framework can be used.

5.2. Multiple Layer Sampler

The second step of the preceding algorithm samples the support of a single layer at once which can exhibit slow convergence in certain situations. For example, if layers $m = 1, 2, 3$ all have support at pixel i, but the actual visible layer should be the background ($m = 3$), the sampler must move through an intermediate state before making the background visible. If the intermediate state is unlikely, this type of proposal can get stuck in a local extrema. Consequently, we develop a single-pixel layer support sampler that samples all layers jointly. First, a visible layer is sampled according to the following categorical distribution

$$\begin{aligned} &\Pr(v^t_i = m) \qquad (26)\\ &= p(x^t_i | v^t_i = m)\ \Pr[\ell^t_{m,i} = 1] \prod\nolimits_{\{l | z^t_l < z^t_m\}} \Pr[\ell^t_{l,i} = 0], \end{aligned}$$

where we have omitted the conditioning variables. If layer m is visible, all layers above m are explicitly precluded from having support at pixel i and all layers below are sampled from Equation 7. This process is repeated for all pixels in a random order. In practice, we sample from Equation 17 by running a full iteration of this "Multiple Layer Sampler" followed by a full iteration of the "Single Layer Sampler".

6. Experiments

Having described a model and associated inference procedure, we compare the performance of the proposed method with results reported in the literature. As noted, we assume that objects of interest have been detected. In our experiments, simple user annotations followed by Lazy Snapping [19] are used to initiate tracking. In contrast to [27] which uses hand-tuned parameters for different datasets, flow parameters are learned automatically using a procedure described in the supplement. Other parameters are fixed across all sequences and can be found in our publicly available source code (http://people.csail.mit.edu/jchang7/). We have verified that a wide range of parameters yield similar performance.

The goal of this work is to develop an efficient, integrated, probabilistic approach for tracking that incorporates flow. While the GP flow lends itself to efficient inference, we do not expect sampling-based inference in conjunction with the partial independence assumption of Equation 11 to outperform state-of-the-art optical flow algorithms, nor is that our goal. Additionally, annotating the segmentation of the first frame leads to an unfair comparison. However, for the interested reader, quantitative results on the Middlebury dataset [1] are provided in the supplement.

6.1. Implementation Details

Flow is an inferred latent variable and, theoretically, initialization does not impact convergence guarantees. However, as with any iterative procedure, convergence may be to a local mode. We have found that using a combination of the optical flow estimates of [4] and [20] as an *initialization* improves convergence empirically. For each frame, both flows are calculated, and the flow that minimizes the L_2 warped image difference is used. Additionally, while a pixelwise appearance model is used, edge effects created by the cameras are not explicitly modeled. Boundary pixels between two regions in an image are often a convex combination of the bordering regions. We find that preprocessing the frames with a simple edge-sharpening procedure improves results. Further details can be found in the supplement.

6.2. Tracking

For each video frame, we draw 100 samples and consider pixels which appear in at least T of the sampled layer supports. Of this set, we take the largest connected component. Results are shown for $T = 25\%$, but other confidence levels may be useful for other applications. Quantitative results on the SegTrack [29] dataset with the top three state-of-the-art algorithms are shown in Table 1. We note that [21] and [18] do not require the first frame to be segmented; however by depending on future data, these methods are a form of Bayesian smoothing instead of filtering. Additionally, we hand-label each video separately from the ground truth as a means to gauge human error. The proposed method achieves state-of-the-art results on most of the videos. Tracked frames and initial annotations for select SegTrack videos are shown in Figure 6. We note that there is ambiguity in the parachute sequence; our algorithm in-

Video	Human	Ours	[21]	[18]	[29]
birdfall	130	265	**189**	288	252
cheetah	308	**570**	806	905	1142
girl	762	**841**	1698	1785	1304
monkeydog	306	**289**	472	521	563
parachute	299	310	221	**201**	235
penguin	279	**456**	-	136285	1705
Mean Error	347	**455**	677*	740*	867

Table 1: Average number of incorrect pixels per frame on SegTrack [29] dataset. * indicates the exclusion *penguin*.

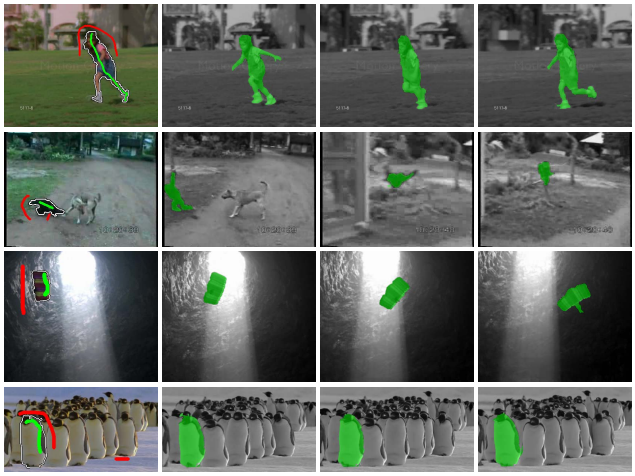


Figure 6: Four results on the SegTrack dataset [29]. First column shows user annotation and segmentation.

corporates the human (as shown in the last frame) whereas the ground truth and other algorithms do not.

Since SegTrack only contains ground truth for single object tracking, we show additional results from the datasets of [11] and [20] in Figure 7 and the supplement. While good results are achieved for most videos, the first video of Figure 7 exhibits a failure of the proposed approach when a car in the background explodes. Such a rapid appearance change is not well represented by Gaussian evolution and the flames are incorrectly labeled as foreground. On the other hand, the last video is over 500 frames long, and the proposed approach is able to track the ice skater throughout the entire sequence.

6.3. Inferring Layer Order

In this section, we show a visualization of the inferred layer order. While we do not impose temporal dependencies among layers, many videos have a static layer order for all frames. We can calculate the posterior distribution over layer orders for the entire video by treating each frame as an independent observation. We show this distribution for two videos in Figure 8. In the first video, the posterior distribution is only non-zero for the three orderings where

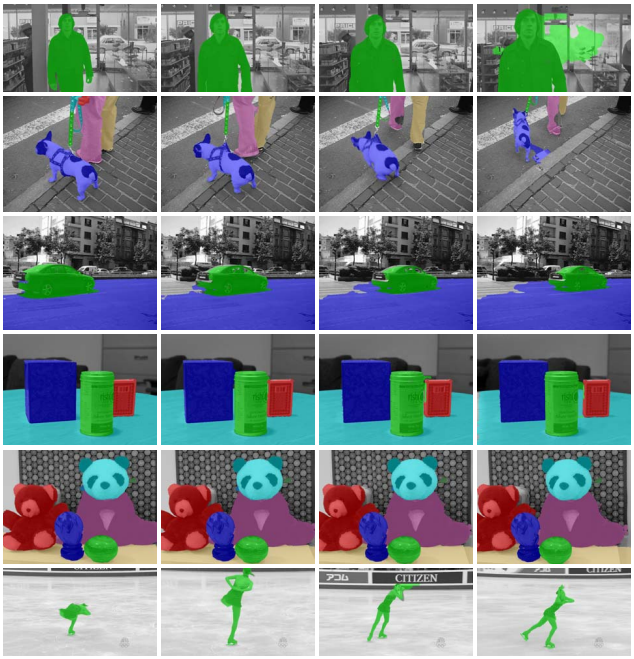


Figure 7: Results on the datasets of [11] and [20].

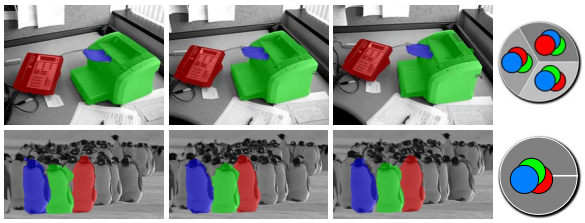


Figure 8: Video frames and pie charts showing the posterior distribution over orderings.

the printer tray is in front of the printer. The uncertainty is expected because the phone and printer do not overlap during the sequence. In the second example, we consider the penguin video of SegTrack. Although the ground truth segmentation only tracks one penguin, we track three here. Because the penguins overlap, the posterior distribution is essentially a delta function at the correct layer order.

6.4. Independent Contributions

We analyze seven variations of our algorithm in order to test different aspects of the model. We consider using only one optical flow algorithm as an initialization (Brox [3] and Liu [20]), not using edge-sharpening (NoEdge), using an optimization scheme (Max), not enforcing topology constraints (NoTop), and treating optical flow as a measurement (OF-g and OF-f). For the optimization-based inference, we change all sampling steps to maximization steps. When using optical flow as a measurement (which [11, 18, 21] do),

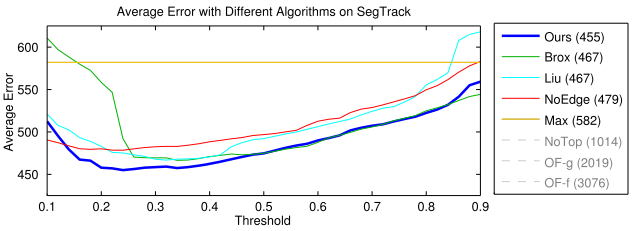


Figure 9: Average errors on SegTrack using different versions of our algorithm. Numbers in legend indicate the best achievable errors. Gray algorithms lie outside the limits.

we can choose to either equate it to g (OF-g) or f (OF-f). The resulting average errors on the SegTrack dataset are shown for varying thresholds (T) in Figure 9. Removing topology constraints or plugging in flow without reinferring it perform so poorly that we do not show the curves. Using different optical flow initializations or removing the edge-sharpening does not change results significantly. While the optimization scheme only produces one segmentation and eliminates the tradeoff with thresholds, it still performs better than the current state-of-the-art algorithms.

7. Conclusion

We presented an integrated layered approach for probabilistic tracking that combines coupled appearance and shape models, topology constraints, layered Gaussian process flow, and efficient sampling. The resulting method outperforms state-of-the-art algorithms. Additionally, we have demonstrated the ability to infer layer orders and have analyzed the impact of individual components of the model.

References

[1] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *ICCV*, 2007. 6

[2] T. J. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(1):90–99, Jan. 1986. 3

[3] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010. 7

[4] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 500–513, 2011. 6

[5] J. Chang and J. W. Fisher III. Efficient topology-controlled sampling of implicit shapes. In *ICIP*, 2012. 1, 4, 5

[6] P. Y. Choi and M. Hebert. Learning and predicting moving object trajectory: a piecewise trajectory segment approach. Technical report, Robotics Institute, 2006. 1

[7] K. Choo and D. Fleet. People tracking using hybrid monte carlo filtering. In *ICCV*, 2001. 4

[8] T. Darrell and A. Pentland. Robust estimation of a multi-layered motion representation. In *IEEE Workshop on Visual Motion*, 1991. 1

[9] D. Demirdjian, L. Taycher, G. Shakhnarovich, K. Grauman, and T. Darrell. Avoiding the "streetlight effect": tracking by exploring likelihood modes. In *ICCV*, 2005. 4

[10] K. Fragkiadaki and J. Shi. Detection free tracking: Exploiting motion and topology for segmenting and tracking under entanglement. In *CVPR*, 2011. 1

[11] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph based video segmentation. In *CVPR*, 2010. 1, 7

[12] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. 4

[13] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 1981. 3

[14] M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *ICCV*, 1998. 4

[15] N. Jojic and B. J. Frey. Learning flexible sprites in video layers. In *CVPR*, 2001. 1, 2

[16] K. Kim, D. Lee, and I. Essa. Gaussian process regression flow for analysis of motion trajectories. In *ICCV*, 2011. 3

[17] D. E. Knuth. *The Art of Computer Programming*. Addison-Wesley, Boston, 1969. 5

[18] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *ICCV*, 2011. 1, 6, 7

[19] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. In *SIGGRAPH*, 2004. 6

[20] C. Liu, W. T. Freeman, E. H. Adelson, and Y. Weiss. Human-assisted motion annotation. In *CVPR*, 2008. 6, 7

[21] T. Ma and L. Latecki. Maximum weight cliques with mutex constraints for video object segmentation. In *CVPR*, 2012. 1, 6, 7

[22] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. 3, 5

[23] Y. Rathi, N. Vaswani, and A. Tannenbaum. A generic framework for tracking using particle filter with dynamic shape prior. *IEEE Trans. on Image Proc.*, May 2007. 1, 4

[24] I. Reid and K. Connor. Multiview segmentation and tracking of dynamic occluding layers. *Image and Vision Computing*, 2010. 1, 2

[25] Y. Shi and W. Karl. Real-time tracking using level sets. In *CVPR*, 2005. 1

[26] B. W. Silverman. Spline smoothing: The equivalent variable kernel method. In *Annals of Statistics*, 1984. 5

[27] D. Sun, E. Sudderth, and M. Black. Layered segmentation and optical flow estimation over time. In *CVPR*, 2012. 2, 5, 6

[28] D. Sun, E. Sudderth, and M. J. Black. Layered image motion with explicit occlusions, temporal consistency, and depth ordering. In *NIPS*, 2010. 2

[29] D. Tsai, M. Flagg, and J. M.Rehg. Motion coherent tracking with multi-label mrf optimization. In *BMVC*, 2010. 1, 6, 7

[30] R. Urtasun, D. Fleet, and P. Fua. 3d people tracking with gaussian process dynamical models. In *CVPR*, 2006. 3

[31] J. Wang and E. Adelson. Representing moving images with layers. *IEEE Trans. on Image Proc.*, 1994. 1

[32] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *CVPR*, 1997. 1, 3

[33] C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis. Improved fast gauss transform and efficient kernel density estimation. In *ICCV*, 2003. 3