Real-time Discriminative Background Subtraction

Li Cheng, *Member*, Minglun Gong *Member*, Dale Schuurmans, Terry Caelli *Fellow*

Abstract

We examine the problem of segmenting foreground objects in live video when background scene textures *change* over time. In particular, we formulate background subtraction as minimizing a penalized instantaneous risk functional—yielding a local on-line discriminative algorithm that can quickly adapt to temporal changes. We analyze the algorithm's convergence, discuss its robustness to non-stationarity, and provide an efficient non-linear extension via sparse kernels. To accommodate interactions among neighboring pixels, a global algorithm is then derived that explicitly distinguishes objects versus background using maximum a posteriori inference in a Markov random field (implemented via graph-cuts). By exploiting the parallel nature of the proposed algorithms, we develop an implementation that can run efficiently on the highly parallel Graphics Processing Unit (GPU). Empirical studies on a wide variety of datasets demonstrate that the proposed approach achieves quality that is comparable to state-of-the-art *off-line* methods, while still being suitable for real-time video analysis ($\geq$ 75 fps on a mid-range GPU).

Index Terms

Real time foreground object segmentation from video, graphics processing units (GPUs), background subtraction, large-margin methods, one class SVM, on-line learning with kernels

I. Introduction

A fundamental problem in real-time video understanding is to distinguish foreground objects from background scenes. Such an analysis provides low-level visual cues that enable further processing such as object tracking [1, 2], pose estimation [3, 4], motion analysis [5–7], activity analysis and understanding

Li Cheng is with Bioinformatics Institute, A*STAR, Singapore, Minglun Gong is with Memorial University, Canada, Dale Schuurmans is with University of Alberta, Canada, Terry Caelli is with NICTA, Australia.

[8]. In this way, background subtraction is crucial to many applications including surveillance, human computer interaction, animation and video event analysis [9].

In this paper, we focus on segmenting foreground objects from video sequences when background scene textures *change* over time. To illustrate the problem, Figure 1 presents three typical scenarios that are characterized by these non-stationary background distributions: *drifting*, *jumping* and *multi-modal switching*. Here, the three panels on the bottom show the temporal dynamics of selected pixels from the corresponding scenarios: Left presents the time series of a foreground pixel (denoted by a triangle) on the chute and a background pixel outside the chute (denoted by a circle in the image), where both exhibit a drifting temporal dynamics; Middle shows a foreground pixel on the hallway area that possesses a clearly jumping dynamics, as well as a background pixel on the way area that are relatively stationary; Right displays an on-the-road foreground pixel and an off-the-road background pixel, where both has a multi-modal switching temporal dynamics.

In general, we aim for a principled algorithm that (1) is computationally efficient for *real-time* video analysis, (2) rapidly adapts to dynamic backgrounds (due to camera motion or the background textures change with time) and (3) is capable of utilizing the spatial-temporal characteristics of video stream data.

A. Our Contribution

Our approach [1] incorporates three main contributions. First, we explicitly connect the problem of background subtraction to work in on-line learning and novelty detection, which possess a rich literature (*e.g.* [11–13]) and well-studied theoretical principles. Second, we present a series of discriminative on-line learning algorithms based on kernels (**ILK**, **SILK** and **SILK-GC**) that provide a principled method for modeling the spatial-temporal characteristics of the background subtraction problem. Third, unlike previous approaches using CPUs that often run off-line or quasi-real-time, our algorithm is designed to work with GPUs, yielding a processing speed of 170 (75) frames per second (FPS) for SILK (SILK-GC)—sufficiently fast for follow-up real-time analysis. Although a number of methods have been developed to cater to real-time processes (such as [14, 15]), they usually perform less competitively in complex scenes compared to state-of-the-art off-line methods [16–19], particularly when there are dynamic backgrounds [16, 18, 19]. By contrast, experiments on a variety of datasets used in [18, 20–22] show that our proposed approach performs comparably to the state-of-the-art algorithms while retaining real-time efficiency.

[1]The project webpage is http://ttic.uchicago.edu/~licheng/BkgSbt.htm. Part of this work appears in [10].

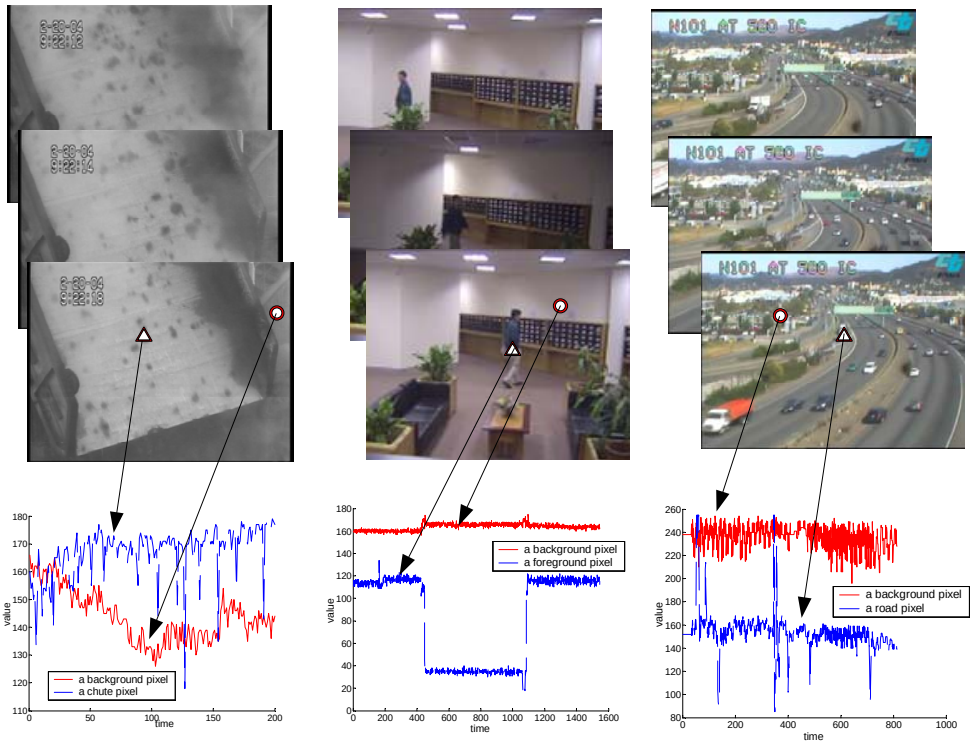


Fig. 1: Video sequences that covers three typical situations (drifting, jumping and multi-modal switching) that characterized by non-stationary, multi-modal background distributions. For each video sequence, three key frames are presented in the upper panels: Left displays the **Rock** sequence where rocks of ore are falling through a rejection chute, with drifting fog in the background; Middle displays the **Lights** sequence which is an indoor sequence where background lights are switched on and off; The right sequence is the **Traffic** sequence of a road traffic scene taking by a shaking camera. The three panels on the bottom show the temporal dynamics of selected pixels from the corresponding scenarios (see text for details). For demonstration purpose only one (the red) channel is used for color images.

B. Related Work

Although assuming various forms, current background subtraction algorithms (also referred to as foreground object segmentation algorithms) generally follow a common approach: one maintains a model of a relatively static background scene while foreground objects are detected as outliers from the background distribution. This scheme is inspired by an observation from the motion picture industry that a background scene can be recovered by exposing a film sufficiently long to wash out the moving objects. This observation, together with the normally assumed static camera constraint, motivate the *pixel process*

approach where pixels are assumed to be statistically independent and the task is to properly estimate the background distribution of each pixel.

Most existing methods for modeling pixel distributions can be categorized as either generative (parametric) or non-parametric.

The simple but often effective *generative* approaches are the single Gaussian and the mixture of Gaussians (MoG) models proposed by Wren et al. [23] and Friedman et al. [24] respectively. An important but rather difficult challenge to these techniques, *e.g.* as pointed out by [20], is to maintain robustness against changes in background scene textures both spatially and over time. A naive pixel process approach encounters difficulties in many real-life situations such as those presented in Figure 1. In cases like these, background pixels are drawn from non-stationary, multi-modal distributions. Accurate background subtraction usually requires accounting for spatial correlations among neighboring pixels in these situations, since temporally corresponding pixels in consecutive frames are not necessarily constant, *e.g.* due to a dynamic background, or a non-stationary camera. To adapt to temporal changes, recursive updating schemes [25, 26] have been considered for the MoG [27] parameters $\{w, \mu, \sigma\}$ in an attempt to maintain a currently relevant model, while Monnet et al. [28] incorporate principal component analysis (PCA) in MoG models. Alternatively [19, 21] use autoregressive models, and [19] resorts to a biologically motivated center-surround method. Attempts have also been made to incorporate the usage of spatial interactions in existing MoG schemes [16, 18], which unfortunately is computationally expensive thus not suitable for real-time video analysis.

Non-parametric methods maintain a background distribution for each pixel based on a list of past examples [20]. In particular, various kernel density estimates (KDEs) have been used for this purpose [29], including Parzen windows [30], and KDEs with variable bandwidth [31]. A major drawback of these approaches, however, is that they ignore the time-series nature of the problem. Moreover, KDE requires training data from a sequence of examples that have a relatively 'clean' background. These shortcomings can be partially remedied by using a sequential approach [32] that uses an iterative algorithm to predict the modes of the KDEs in an on-line fashion. To incorporate spatial coherency of objects, Sheikh and Shah [22] introduce a post-processing step that uses maximum a posteriori inference in a Markov random field (or MAP-MRF), where a simple Ising model is used as prior of the foreground/background label field. An off-line algorithm is considered in [17] to track the position of corresponding pixels over time using particle filters where KDE is used for the likelihood model.

An alternative approach to these aforementioned methods is based on a 'supervised learning' framework. In certain situations there might exist training images where foreground objects (or segments

thereof) have been manually labeled. This allows supervised learning-based binary scene labeling approaches to be deployed, as described in *e.g.* [14, 33–37]. Such an approach is beyond the scope of this paper, since these algorithms require off-line training on large amount of manually-labeled images. For example, one experiment in [37] is conducted in an office setting with 500 training images and 90 test images, as well as synthesized training images to increase the foreground sample size. A second drawback is that supervised learning approaches usually assume a fixed number of foreground categories (for example, [14] deals with three types of foregrounds: human, animal and vehicle). Our focus in this paper is on situations where no a priori annotations are assumed for the incoming video stream.

The remainder of this paper is organized as follows. Section II examines statistical learning techniques that are relevant to our framework: 1-SVMs, on-line learning, and kernels. In section III, the background subtraction problem is formulated as minimizing a regularized risk functional under constraints. To deal with dynamic backgrounds, we further consider an on-line learning framework based on [11, 38], which leads to a feasible closed-form solution (section III-C) as a central component of our first proposed algorithm. We subsequently analyze its convergence (III-E) and address issues such as sparse approximation (III-D) and induced truncation error (III-E). Next, to incorporate spatial correlation, the framework is extended to incorporated graph-cuts in section IV. Section V provides more details on the GPU implementation of our algorithms. This is followed by experiments in section VI and the conclusion and discussion in section VII.

II. Preparation

For completeness, we briefly examine the relevant statistical learning techniques we use in our approach: 1-SVMs, on-line learning and kernels.

A. 1-SVMs

The proposed discriminative approach is motivated by the large-margin principle [39], and in particular the 1-SVM approach [12] for detecting anomalies from stationary distributions. Here we are interested in predicting the optimal separating hyperplane $f(\cdot)$ in some high-dimensional feature space that incurs a convex loss

$$L(x_t, f) = \big(\gamma - f(x_t)\big)_+. \tag{1}$$

where $(\cdot)_+ \triangleq \max\{\cdot, 0\}$. In this case $f(x)$ can be viewed as a score function [40] that assigns larger values to values that are more confidently from the background distribution, while smaller values indicate

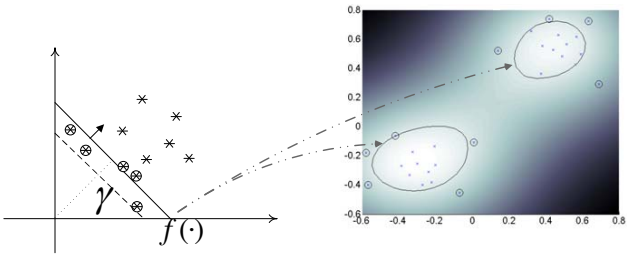


Fig. 2: An illustration of 1-SVM on a sample of examples that motivates the proposed approach. See text for details.

anomalies. The parameter γ is the 'margin value', which measures the distance of the separating hyperplane from the origin (Figure 2), and is set to one during the experiments. Let x denote the observed value of a pixel, $(x_t)_{t=1}^T$ the observation sequence of this pixel over a period of time T. Figure 2 presents a simple illustrative example, where a set of past examples are scattered in input space as a mixture of two Gaussians (right panel) and re-represented in a corresponding feature space (left panel). Regardless of the shape of the background distribution, in a proper feature space representation, a separating hyperplane f will be estimated that encloses as many background examples as possible in the input space. Examples are then partitioned into background versus outliers (*i.e.,* foreground examples) accordingly.

In this approach $f(\cdot)$ is a linear operator in a reproducing kernel Hilbert space (RKHS) corresponding to straight lines in the left panel. Due to an implicit mapping from the original representation to the feature representation, the pre-image of a high confidence region, $f(x) \geq c$, will generally be determined by a non-linear region in the input space (curved lines in the right panel). One nice aspect of a large margin approach is sparsity: the function $f(\cdot)$ is determined solely from a weighted average of so called *support vectors*, shown as circular points in Figure 2. Unfortunately, the presence of noise in practice would make it almost impossible to find such a well-separated hyperplane $f(\cdot)$ for every point.

B. Three Learning Paradigms: Batch, Incremental, and On-line

The standard risk minimization principle, suggested in the 1-SVM case above, can be deployed in different ways in practice.

To begin, assume we have access to labeled examples drawn from a distribution P. Let $L(x, y, f)$ denote the incurred loss of a prediction function f on an example (or (instance, label) pair) (x, y). We are interested in minimizing the *expected risk* functional

$$R_{\exp}(f) \triangleq E_{P(\mathcal{X}\times\mathcal{Y})}[L(x, y, f)]. \tag{2}$$

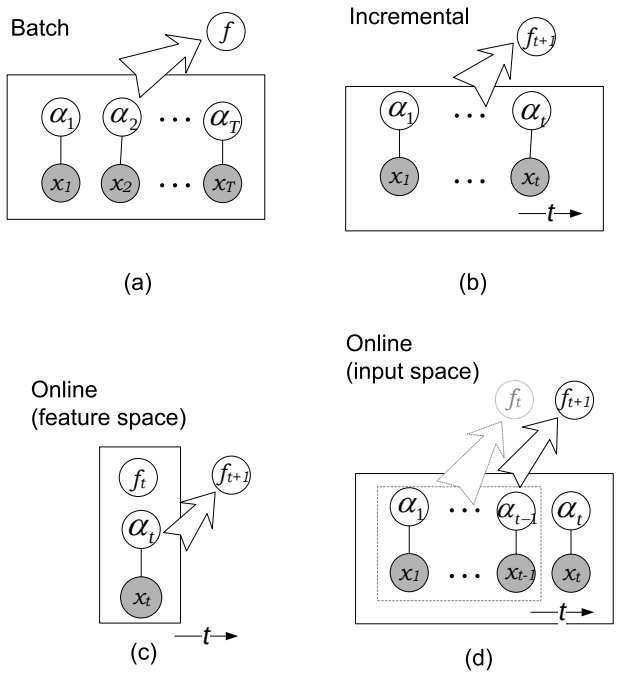


Fig. 3: Comparison of batch, incremental and on-line learning paradigms (in both feature space and input space). In each panel, shaded nodes represent observed examples, while white nodes denote the estimated weights associated with examples. The rectangular box and the arrow thats point to f together refer to the fact that f is determined by a collection of the components being held inside the box. For online learning, f_{t+1} is estimated using only the current example x_t and the previous estimate f_t. There is no need to revisit the previous examples.

as the qualitative measure of function f. In practice, since only a finite sample $S \triangleq (x_t, y_t)_{t=1}^T$ can be observed, one instead minimizes the *empirical risk* functional over S

$$R_{\text{emp}}(f) \triangleq \frac{1}{T}\sum_{t=1}^{T} L(x_t, y_t, f). \tag{3}$$

The theory of Vapnik and Chervonenkis [39] has led to the so-called structured risk minimization principle, where one minimizes a *regularized risk* that upper-bounds the *empirical risk*

$$R_{\text{bth}}(f) = R_{\text{cap}}(f) + \eta R_{\text{emp}}(f). \tag{4}$$

Here $\eta > 0$ is a tuning parameter and $R_{\text{cap}}(f)$ denotes a measure of complexity of the function f, which we will explicity measure by its 2-norm in the RKHS (feature representation) being used

$$R_{\text{cap}}(f) \triangleq \frac{1}{2}||f||_{\mathcal{H}}^2. \tag{5}$$

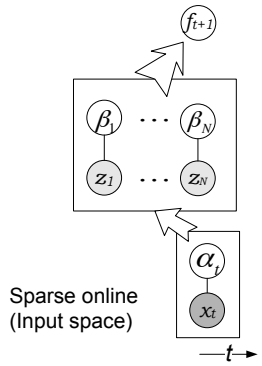


Fig. 4: The proposed sparse on-line learning diagram (SILK). See text for details.

This general approach of minimizing regularized empirical risk can be deployed in three different learning schemes—batch, incremental and on-line—as shown in Figure 3. In *batch learning*, as depicted in Figure 3(a), f is determined solely on the empirical sample $(x_t)_{t=1}^T$. In this case, the set of weighted examples with weights $\{\alpha_t\}_{t=1}^T$ determines f uniquely (referred to as the kernel expansion in Schölkopf et al. [41]). Alternatively, Figure 3(b) depicts *incremental learning*, where a learner is gradually exposed to larger set of examples as t increases, and thus the function f is determined by kernel expansions with increased number of terms. Thus, when $t \to T$, f_{t+1} approaches f of the batch learning scenario.

However, for the problem of segmenting foreground objects from a video sequence, each of these two learning paradigms has drawbacks. First, batch learning requires all previous examples to be gathered (and stored in memory) prior to training, which essentially requires buffer of unlimited size as $T \to \infty$. This is indeed not desirable for real-time analysis, since both storage and computational resources will be quickly exhausted when processing video streams. Second, both paradigms require re-training when new examples arrive. This can be implemented by off-line training (*e.g.* SVM learning) but at the expense of considerably more training time (*e.g.* re-solving a quadratic program), which makes it impractical for real time efficiency. Third, a fundamental assumption for batch (and incremental) learning is that the example distribution is *stationary*. Unfortunately, this almost never holds in video sequences, where object distributions most often change over time.

On-line learning, by constrast, is particularly well-suited to video background subtraction. First, only a current example x_t and the previous estimate f_t are required to estimate f_{t+1}, since one example (x_t, y_t) is processed at time t and f_t summarizes all previous examples $(x_i, y_i)_{i=1}^{t-1}$; as illustrated in Figure 3(c). Second, we are able to obtain an efficient closed-form update (see below) that completely avoids the issue of solving difficult optimization problems as each new example (frame) arrives. Third,

by processing examples one at a time, it is possible for the on-line learning algorithm to closely track the dynamics of the sample distributions over the time, as we will show.

In our on-line learning approach we will follow a simple training principle developed by Kivinen, Warmuth and others [11, 42]. Let $\eta_t > 0$, and let $R_{\text{inst}}(f)$ denote the *instantaneous risk* for current example (x_t, y_t). Furthermore let $R_{\text{div}}(f)$ denote a measure of the *divergence* of any new estimate f to the current estimate f_t, which we will assume is given by the squared Euclidean distance in the RKHS

$$R_{\text{div}}(f) \triangleq \frac{1}{2}||f - f_t||^2_{\mathcal{H}}. \tag{6}$$

Now, we would like to minimize the regularized risk

$$R_{\text{onl}}(f) = R_{\text{div}}(f) + \eta_t R_{\text{inst}}(x_t, y_t, f). \tag{7}$$

Henceforth we will use $R(f)$ to refer to $R_{onl}(f)$.

III. Our Approach

A. Problem Formulation

Let $\boldsymbol{x} \in \mathcal{X}$ denote an observed image consisting of n pixels, and let $\mathcal{Y}$ be the set of feasible labels. A label $\boldsymbol{y} \in \mathcal{Y}$ is defined over an image as $\{y_i\}_{i=1}^n$, where i indexes a local pixel. For the ith pixel let $y = y_i \in \{-1, +1\}$ indicate whether this pixel belongs to the foreground (-1) or background $(+1)$. When processing an input video stream $\mathcal{T} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$, the goal is to assign labels $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T)$ on the fly that correctly identify foreground versus background pixels.

B. Batch Learning

Let $L(\boldsymbol{x}, \boldsymbol{y}, f) \triangleq \sum_i L(x_i, y_i, f)$ denote a desired loss function. The problem can then be abstractly cast as learning a model that incurs least cumulative loss on $\mathcal{T}$. Let us start by considering a batch scenario where the *entire* set of images $\mathcal{T}$ is available and the learning procedure seeks to minimize a total regularized risk functional

$$\min_f \quad \frac{1}{2}\|f\|^2_{\mathcal{H}} + \frac{\eta}{T}\sum_{i,t} L(x_{i,t}, y_{i,t}, f), \tag{8}$$

Here $x_{i,t}$ and $y_{i,t}$ can also be simplified as x_t and y_t when without confusion. Generally, we will focus on using an SVM type loss function, as for example introduced in section II.A. As an immediate result of the well known Representer theorem (Theorem 4.2 of Schölkopf et al. [41]), the optimal function $f \in \mathcal{F}$ for the regularized risk of (8) in the RKHS $\mathcal{H}$ admits a representation of the form $f(\cdot) = \sum_{t=1}^T \alpha_t k(x_t, \cdot)$.

Thus the problem amounts to learning a mapping $f \in \mathcal{F}$ that minimizes $R(f)$, where the function space admits

$$\mathcal{F} = \left\{ f' \in \mathcal{H} \;\middle|\; f'(\cdot) = \sum_{i=1}^{T} \alpha_i k(x_i, \cdot), \alpha_i \in \mathbb{R} \right\}. \tag{9}$$

C. On-line Learning with Kernels

In our context, for each pixel x, a sequence of pixel observations S is collected during the time course, where x_t is observed at time t. For this pixel process at time $t+1$, we would like to predict a (possibly non-stationary) function f. This calls for the aforementioned on-line learning paradigm where, at time t, an instance x_t is presented to a learner, which uses its parameter vector f_t to predict a label. This predicted label is then compared to the true label y_t via a non-negative, convex risk function $R_{\text{inst}}(x_t, y_t, f_t)$. The learner then updates its parameter vector to minimize a convex risk function $R(f)$, and the process repeats. More precisely, we have

$$R(f) = \underbrace{\frac{1}{2}\|f - f_t\|^2_{\mathcal{H}}}_{R_{\text{div}}(f)} + \eta_t \underbrace{\left(\frac{\lambda}{2}\|f\|^2_{\mathcal{H}} + CL(x_t, f) \right)}_{R_{\text{inst}}(f)}, \tag{10}$$

where the divergence risk $R_{\text{div}}(f)$ measures the distance of predicted f from previous prediction f_t; the instantaneous risk consists of a capacity term $\|f\|_{\mathcal{H}}$ that controls the complexity of the prediction f, and a 1-SVM type loss term (1) is used to evaluate current example. At each time t, our goal is to learn a mapping $f \in \mathcal{F}$ that minimizes $R(f)$. Note that in our setting the step size η_t can be computed using sophisticated algorithms [43], but for ease of exposition we assume a fixed step size (learning rate) $\eta_t = 1$.

As $R(f)$ is convex, (10) would be solved by setting the gradient (or a subgradient when necessary) to 0. This gives

$$f_{t+1} \;=\; f_t - \partial_f R_{\text{inst}}(x_t, y_t, f_{t+1}). \tag{11}$$

Note the dependency on f_{t+1} on both the left as well as the right hand sides of the above equation. Therefore it is difficult to determine $\partial_f R_{\text{inst}}(x_t, y_t, f_{t+1})$ in closed form. An *explicit* update, as opposed to the above *implicit* update, uses the approximation $\partial_f R_{\text{inst}}(x_t, y_t, f_{t+1}) \approx \partial_f R_{\text{inst}}(x_t, y_t, f_t)$ to arrive at the more easily computable expression [44], results in the familiar stochastic gradient descent update

$$f_{t+1} \;=\; f_t - \partial_f R_{\text{inst}}(x_t, y_t, f_t). \tag{12}$$

In both cases, there must exist coefficients α_i fully specifying f_{t+1} by

$$\begin{aligned} f_{t+1} &= \sum_{i=1}^{t} \alpha_i k(x_i, \cdot) \\ &= \sum_{i=1}^{t-1} \alpha_i k(x_i, \cdot) + \alpha_t k(x_t, \cdot) \end{aligned} \tag{13}$$

Lifting this into the RKHS, the 1-SVM loss of (1) can be re-written

$$L(x_t, f) = (\gamma - f(x_t))_+ = (\gamma - \langle f, k(x_t, \cdot)\rangle_{\mathcal{H}})_+,$$

where $k(x, \cdot) \triangleq \phi(x)$. [42] shows that the subgradient of this loss can be written as $\partial_f L = \beta_t k(x_t, \cdot)$, which in our context reduces to one of three cases:

$$f(x_t) > \gamma \implies \beta_t = 0; \tag{14a}$$

$$f(x_t) = \gamma \implies \beta_t \in [-1, 0]; \tag{14b}$$

$$f(x_t) < \gamma \implies \beta_t = -1. \tag{14c}$$

As before, let $\hat{\alpha}_t$ denote the optimal estimate of α_t, which leads to $\gamma - f_{t+1}(x_t) = 0$.

We introduce an auxiliary variable $\tau = \frac{\lambda}{1+\lambda}$. Using (11) we obtain

$$\gamma - ((1-\tau) f_t(x_t) + \hat{\alpha}_t k(x_t, x_t)) = 0,$$

which yields

$$\hat{\alpha}_t = \frac{\gamma - (1-\tau) f_t(x_t)}{k(x_t, x_t)}. \tag{15}$$

On the other hand, $\hat{\alpha}_t$ has bounded influence due to the piecewise linear nature of the 1-SVM loss function (refer to (11), (13) and (14))

$$\hat{\alpha}_t \in [0, (1-\tau)C]. \tag{16}$$

Combining the two scenarios gives the update

$$\alpha_t \in \begin{cases} \hat{\alpha}_t & \text{if } \hat{\alpha}_t \in [0, (1-\tau)C]; \\ 0 & \text{if } \hat{\alpha}_t < 0; \\ (1-\tau)C & \text{if } \hat{\alpha}_t > (1-\tau)C. \end{cases} \tag{17}$$

It follows from (11), (13), (14), and (17) that

$$\alpha_i \leftarrow (1-\tau)\alpha_i \qquad \text{for } i = 1, \ldots, t-1, \tag{18}$$

$$\alpha_t \leftarrow -(1-\tau)C\beta_t. \tag{19}$$

The above closed-form expressions give rise to an on-line 1-SVM algorithm (Alg. 1) that performs *implicit* updates, and is termed as **ILK** for "implicit on-line learning with kernels". The update equations of ILK enjoy certain advantages. For example, using (18) it is easy to see that an exponential decay term can be naturally incorporated to down-weight past observations:

$$f_{t+1} = \sum_{i=1}^{t} (1-\tau)^{t-i} \alpha_i k(x_i, \cdot). \tag{20}$$

Intuitively, the decay rate $\tau \in (0, 1)$ trades off between the regularizer and the loss on the current sample. In particular, the weight $|\alpha_i|$ is always upper bounded by $(1-\tau)C$, which ensures limited influence from outliers (*cf.* (17)). In addition to dealing with stationary background distributions, when the examples are drawn from a time-varying distribution $P(t)$, the algorithm can still predict a reasonable separating hyperplane f, as long as the decay rate τ matches the drifting speed of $P(t)$.

Algorithm 1 The (**ILK**) Algorithm

Input: Margin γ, cut-off value C, threshold ϵ ($0 < \epsilon < \gamma$), decay rate τ.

Output: Weight sequence (α_t), and prediction sequence (y_t).

$f_1 \Leftarrow \underline{0}$

for $t = 1$ **to** T **do**

Receive an example x_t

Assign label as

$$y_t = \begin{cases} +1, & f_t(x_t) \geq \epsilon; \\ -1, & otherwise. \end{cases} \tag{21}$$

Update $(\alpha_i)_{i=1}^{t}$ according to (18) and (17)

end for

D. ILK and SILK Algorithms

One fact that is not obvious from the illustration of Figure 3(c) in the feature space is the complexity of kernel expansions: When lifting $\phi(\cdot)$ to map x to some infinite dimensional RKHS feature representation $k(x, \cdot)$, it has to be represented as a weighted combination of examples in the input space, a well-known issue for the kernel methods [41]. This is better revealed in Figure 3(d) where in the input space, the past estimate f_t still requires a set of kernel expansions $\{\alpha_i k(x_i, \cdot)\}_{i=1}^{t-1}$, which grows linearly as $t \to \infty$.

To address this limitation, Kivinen et al. [42] propose an algorithm that assigns equal weights to examples (*explicit* update), then truncates older examples to maintain a memory of fixed size, which is essentially a kernel perceptron algorithm with truncations in kernel expansion. We have already presented the (*implicit* update or **ILK**) algorithm that is able to assign *different* weights to examples according to

the incurred losses. This is contrast to the algorithm of [42] where all support vectors are assigned a *same* weight value. To lessen the memory and computational burdens, we further propose a sparse variant (**SILK**), where a buffer of size ω is maintained, and each new point x_t is inserted into the buffer with coefficient α_t. Once the buffer limit ω is exceeded, the point with the lowest coefficient value is discarded to maintain an upper bound on memory usage. Empirical study [38] shows that this scheme is more effective than the straightforward *least recently used* (LRU) strategy utilized in [42, 45]. In practice, the value of ω is fixed to 20 in CPU implementation and 50 in GPU implementation. Meanwhile, other buffer (or budget) maintenance schemes have also been adopted in literature: [46] instead proposes to remove the inactive (or looser) support vectors. As this is not robust against noise, Weston et al. [47] use a modified scheme that at each round remove one example that incurs the least cumulative risk on all currently seen examples. This is shown to improve the performance, at the expense of being computationally very intensive. Both schemes however rely on the assumption of a stationary target that does not hold in our context. In addition, [48] (Figure 2) proposes a randomized algorithm that removes a *random* support vector from the existing buffer at a time.

In term of complexity, SILK is both computationally more efficient and less memory demanding than ILK. Given a sequence of T samples, the space complexity of ILK grows as $O(T)$ while that of SILK is fixed at $O(\omega)$ with a constant $\omega << T$. Furthermore, the computational complexity of the algorithm is mainly dictated by the time required to compute $f(x_t)$ which in turn depends on the number of kernel expansions. Therefore, assuming constant time per kernel computation, the computational complexity per of ILK is again $O(T)$ per new example while that of SILK is $O(\omega)$.

E. Truncation Error of SILK

In addition, it is important to understand to what extent the SILK will deviate from ILK. Let $\alpha_i^{(t)}$ $(i \leq t)$ denote the weight of kernel expansion $k(x_i, \cdot)$ at time t. Let the mapping of the index of observation in buffer $\hat{S}$ of size ω to the set of observations S of size T $(\omega << T)$ be denoted as $\hat{N} : \hat{S} \rightarrow S : j \mapsto i$, where $|\hat{S}| = \omega$, $|S| = T$. The following theorem shows that the difference between the true predictor and its truncated version obtained by storing only ω expansion coefficients decreases exponentially as the buffer size ω increases.

Theorem 1: Assume $k(x_i, x_i) \leq X^2$ for any i. Let the non-truncated representation of the RKHS predictor be

$$f(\cdot) = \sum_{i=1}^{t} \alpha_i^{(t)} k(x_i, \cdot), \tag{22}$$

and the truncated approximation be

$$\tilde{f}(\cdot) = \sum_{j=1, i\in\hat{N}(j)}^{\omega} \alpha_i k(x_i, \cdot). \tag{23}$$

Then the truncation error is upper bounded by

$$\left\| f - \tilde{f} \right\|_{\mathcal{H}} \leq \frac{(1-\tau)^{\omega+1}}{\tau} C X^2, \tag{24}$$

which decreases exponentially as the size of buffer ω increases.

The detailed proof appears in the Appendix.

F. Convergence Analysis

So far we argue that our online learning algorithm (S)ILK is more efficient when comparing to the batch learning counterparts, as succinctly depicted in Figure 3. One might wonder that whether the performance of (S)ILK is still comparable to the batch learning algorithms in, for example, a stationary scenario where the batch learning methods are most suitable. We show in Theorem 2 in the Appendix that under some mild assumptions, the cumulative risk $\sum_{t=1}^{T} R(x_t, y_t, f_t)$ of the hypothesis sequence produced by (S)ILK converges to the minimum risk of the batch learning counterpart $g^* \triangleq \operatorname{argmin}_{g\in\mathcal{H}} R_{\mathrm{bth}}(g)$ at a rate of $O(T^{-1/2})$, where R_{bth} is the regularized risk of batch learning defined in (4).

IV. Incorporating Spatial Correlations

To accommodate temporal changes of each pixel, we have the proposed discriminative algorithms (S)ILK at our disposal, meanwhile it is also necessary to enforce spatial coherence between neighbor pixels. In this section, we address this issue by explicitly enforcing the smoothness constraints under a MAP-MRF framework (*e.g.* [49]), which leads to a revised algorithm (referred to as **SILK-GC**). More precisely, this MAP-MRF is defined over a 2-D lattice graph as maximizing the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ denote the observed image and the label over the image field, respectively. This can be realized as

$$-\log p(\boldsymbol{y}|\boldsymbol{x}) \propto \sum_i E_v(x_i, y_i) + \sum_{i\sim j} E_e(x_i, x_j, y_i, y_j). \tag{25}$$

Here x_i (y_i) refers to the local observation (foreground/background label) at pixel i, $i \sim j$ denotes a pair of neighbor pixels, and E_v (E_e) is the energy function of this pixel (pair of pixels). Clearly the MAP-MRF problem can be equivalently formulated as finding a globally optimal assignment $\boldsymbol{y}^*$ that minimizes the sum of energy functions in RHS.

Moreover, a graph-cuts method [50, 51] is adopted here to solved this induced inference problem of (25). Since graph-cuts is known to be able to solve this binary optimization problem *exactly*, in practice SILK-GC is able to infer a globally *optimal* (or near-optimal) solution. To utilize graph-cuts, let the source $\boldsymbol{s}$ represents the foreground object label, and the sink node $\boldsymbol{t}$ for the background label. Each pixel x_i has two edges $\in E$ connecting to $\boldsymbol{s}$ and $\boldsymbol{t}$ respectively. The energies (capacities) of these two edges are defined as

$$\begin{aligned} E_v(x_i, y_i = \boldsymbol{s}) &= (0, \epsilon_h - f_t(x_i))_+ \\ E_v(x_i, y_i = \boldsymbol{t}) &= (0, f_t(x_i) - \epsilon_l)_+ \end{aligned} \tag{26}$$

where ϵ_h and ϵ_l are the upper and lower bounds of the score $f_t(x_i)$, respectively. As a consequence, the more likely pixel x_i belongs to the foreground, the higher the value $f_t(x_i)$ is, and the lower the energy of $E_v(x_i, \boldsymbol{s})$. At the same time, the value of $E_v(x_i, \boldsymbol{t})$ will be higher since x_i is unlikely to be the background.

Meanwhile, given a pair of neighboring pixels x_i and x_j, its edge energy (or capacity), $E_e(x_i, x_j, y_i, y_j)$, follows the Ising model as:

$$E_e = \delta(y_i \neq y_j)\big[\lambda_h - \min(\lambda_h - \lambda_l, |x_i - x_j|)\big], \tag{27}$$

where $\delta(\cdot)$ is the indicator function, and $|\cdot|$ is the L1-norm. Parameters $\lambda_h = 0.2$ and $\lambda_l = 0.15$ set the upper and lower bounds of the discontinuity cost to encourage the boundary of the foreground masks along object boundaries. This allows the edge energy of the pixels across boundary to be determined adaptively by their color difference. In other words, the smaller the color difference is between neighboring pixels, the higher the cost is for neighboring pixels taking different labels, and hence the higher the capacity this corresponding edge should possess.

V. GPU IMPLEMENTATIONS

To ensure real-time analysis, both the SILK and SILK-GC approaches are designed to work with programmable graphics hardware. Graphics Processing Units (GPUs) on modern graphics cards allow developers to write their own computational kernels that are executed on multiple data chunks (vertices or pixels) in *parallel*. Traditionally GPUs are dedicated to 3D graphics applications where the computational kernels are used to calculate the transformation and lighting of each vertex (called vertex shaders), or to compute the shading of each rasterized pixel (called pixel shaders). For general purpose applications, the computation process is normally cast as a rendering process that involves one or more rendering passes. Within each rendering pass, the following operations are sequentially performed: (1) represent the input

data as 2D or 3D arrays and load them into the video memory as textures; (2) load the algorithm into the GPU as a pixel shader; (3) set either the screen or a pixel buffer in video memory as the rendering target; and (4) execute the shader by rendering a image-sized rectangle. To be able to process video sequences in real time, we carefully exploit the inherently concurrent structure of the proposed algorithms to work efficiently with GPU processors.

A. Implementation of SILK on GPU

In our GPU implementation of the SILK algorithm, the support vectors and the corresponding coefficients for different pixels are stored in a single 2D color texture, referred as the buffer texture. The buffer texture is divided into ω tiles and each tile keeps one observation-coefficient pair for each pixel in the image. To retain the most important support vectors, all observation-coefficient pairs are sorted in descending order according to the absolute values of the coefficients. When the number of support vectors is larger than ω, only the ones with higher coefficients are kept in the buffer texture.

Now, as a new frame t is presented, three major steps are preformed: score calculation, foreground object segmentation, and buffer update. In the score calculation step, a pixel shader takes the new observation and the existing buffer as two input textures, computes the function score $f_t(x_t)$ for each of the pixels, and stores the result into a new score texture. The score texture is then used as input of the second pixel shader in the object segmentation step, which segments foreground objects through local thresholding. Finally in the buffer update step, the coefficient α_t for each pixel is evaluated using (17) and is used to update the buffer texture.

B. Implementation of SILK-GC on GPU

The implementation of SILK-GC uses the same score calculation and buffer update steps as of the SILK algorithm. To incorporate spatial correlations, a graph-cuts based optimization process is performed instead during the second step. Here the graph-cuts is adapted from the parallel push-relabel algorithm [51, 52]. To summarize, it starts by initializing the excess of each node defined as the difference between the total amount flowing in and out of this node. The local flow excess is then pushed toward the sink and the residual graph is updated until all paths to the sink are saturated. Finally, excess that cannot be moved to the sink is pushed backward to the source. After all nodes have zero excess, the residual graph delivers the assignment as the minimum cuts.

In theory it takes $O(|V|)$ push-relabel steps to compute the minimum cuts [50], while we empirically find that a few push-relabel steps toward the sink together with a few backward steps toward the source

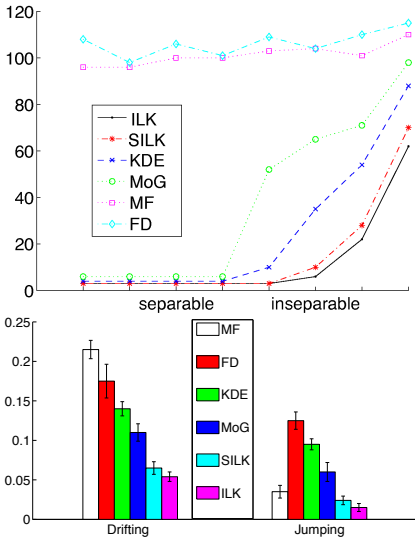


Fig. 5: Comparisons of six background subtraction algorithms (a) on sequences of examples drawn from stationary distributions of mixture of two Gaussians, where y-axis presents the cumulative mistakes, and (b) on the drifting (left) and the jumping (right) scenarios, respectively, where y-axis gives the average mistakes.

already produce near optimal assignments, by effectively removing false positives in background areas, as well as eliminating false negatives in object areas. During the experiments, both forward and backward push-relabel steps are fixed to 10.

In our graph-cuts implementation, the graph G is represented using two color textures. At pixel node x, the first texture keeps the residual capacities from x to its four neighboring nodes in its four color channels. The residual capacities from x to source $\boldsymbol{s}$ and sink $\boldsymbol{t}$ are kept in the red and green channels of the second texture, while the blue and alpha channels of the second texture store the excess and the label of node x, respectively. The graph is initialized using a pixel shader, which sets the initial residual capacities to the capacities of the corresponding edges (calculated by (26) and (27)). Then each push-relabel step is implemented using two rendering passes: the first pass computes the amount can be pushed away from the current node, whereas the second pass updates the excess and the label of each node. Once all push-relabel steps are completed, the final rendering pass yields a globally near-optimal assignment.

VI. EXPERIMENTS

We conduct experiments on a variety of video sequence datasets. The results of the proposed algorithms are compared to standard background subtraction methods including Adjacent Frame Difference (FD) [20], Mean-Filter (MF) [20], as well as to the recent variant of mixture of Gaussians (MoG) using recursive updates [25–27] [2] , KDE [29], and the method of Sheikh et al. [22] (referred to as Sheikh et al.)[3]. Throughout the experiments, Gaussian kernels are used for the proposed approach, The buffer size ω is set to 20 for the CPU implementation, and a larger buffer size of 50 is used for the GPU implementation, since it does not change much of the runtime of our GPU implementation and at the same time gives slightly better performance. The margin γ and the value of C are fixed to 1. A fixed parameter set is adopted during the experiments for both (S)ILK and SILK-GC algorithms, including $\sigma = 8$ for the RBF kernel, the decay factor $\tau = 0.95$. Meanwhile, MF uses 30 past frames, while MoG utilizes a mixture of two Gaussians ($k = 2$), and as suggested by the authors [25, 53], uses $\alpha = 1/T$ (T is the number of frames) for controlling the speed of online update. The bandwidth of KDE method is set to 20. Following the choice of Sheikh et al. [22], a fixed set of (h_r, h_d)=(16, 25) is used for its parameterized bandwidth matrix H.

A. (S)ILK on Synthetic Sequences

We conduct controlled experiments to evaluate the performance of (S)ILK in *three types* of scenarios, where two-dimensional synthetic sequences are constructed using a large quantity of background examples and a relative small amount of foreground examples. In all of the three cases, a mixture of two Gaussians is employed to generate the background examples, and similarly for foreground.

In the first scenario (presented in Figure 5 top panel), we focus on *stationary* background and foreground distributions, and evaluate the robustness of (S)ILK as the separability decreases. The vertical axis shows the number of mistakes occurred (averaged over 20 trials). Each position along the horizontal axis presents one case of stationary distributions, consisting of 2000 two-dimensional sampled instances. Along the axis a set of stationary distributions are arranged from left to right according to the decrease of separability that ranges from separable cases to inseparable cases, achieved by relocating the means of the Gaussians. Note that to maintain a separable case, all examples that would be misclassified by the Bayes optimal classifier are dropped off. Here as the stationary background model is a mixture of Gaussians, algorithms

[2]We use the implementation of [25, 53], down-loadable from http://staff.science.uva.nl/~zivkovic/Publications/CvBSLibGMM.zip.

[3]Downloaded from http://www.cs.cmu.edu/~yaser/Background.zip

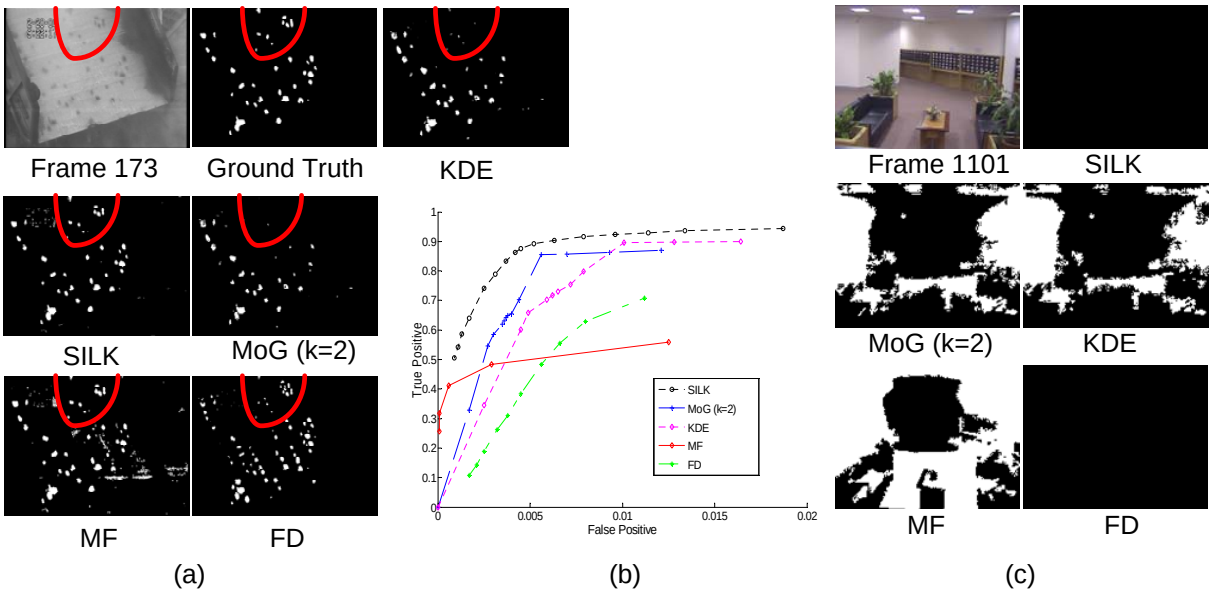


Fig. 6: (a) A comparison of five algorithms on one frame of the *Rock* sequence. FD fails for this task due to the fast sliding speed of the rocks. Notice KDE and MoG miss the rocks in the red curve while SILK still detects them. (b) ROC curve of five algorithms on rock video. (c) A comparison on one frame of the *Lights* sequence when the lights are just switched off. The result of SILK and FD are shown to adapt to this jumping situation faster than other algorithms.

such as MF and FD perform poorly by making frequent mistakes, MoG makes comparably less mistakes. On the other hand, the three non-parametric algorithms – KDE, ILK and SILK – are able to capture the mixture distribution and make least average cumulative mistakes.

Next we consider the scenarios of changing background distributions, where the examples are sampled in a similar manner, only now they are drawn sequentially from a temporally *drifting* (and *jumping*) distributions: In drifting scenario, the distribution drifts after a short time span (duration = 5 frames), while in jumping scenario, the distribution jumps after a relative longer time span (duration = 50 frames). We focus on the separable case. As clearly revealed from Figure 5 bottom panels, ILK and SILK are good at dealing with these changing distribution scenarios, where SILK makes slightly more mistakes than ILK, but both are superior to the remaining algorithms. We note that KDE empirically perform on par with MoG on both situations.

B. SILK on Real Sequences

We further evaluate SILK on the three real scenarios presented in Figure 1. We show that SILK works competitively against these widely used background subtraction methods. Here a same set of fixed parameter values from the previous experiments are also employed.

Fig. 7: This figure presents, in each row, the (image, result) pair of applying the SILK algorithm on key frames of the Traffic sequence.

The *Rock* video of Figure 1 (left) is taken from an ore mining site, where the ore rocks are falling through a rejection chute. A grey-scale surveillance camera is mounted on top of the chute to monitor the real-time processing, where statistical information is to be collected about the number and sizes of the ore rocks passed by. As presented in the bottom-left panel, both the background pixel the foreground (chute) pixel exhibit drifting behaviors, mostly due to the spread of the dust fog. Even for a human observer, some ore rocks are very difficult to be distinguished from the background scene due to similar appearance. Figure 6a displays the obtained results on the $173th$ frame, where SILK detects the ore

rocks (especially in the red curve area) as promptly as MF, with much less false positives. Obviously, both KDE and MoG miss the several rocks on the top-right corner. FD does the worst, mostly due to the fast sliding speed of the ore rocks passing through the chute. Figure 6b presents the ROC curves, where overall SILK outperforms other methods.

The middle panels of Figure 1 presents the *Lights* video, an indoor video sequence where the lights are switched off and back on, that exhibits a jumping distribution. In particular, SILK is shown to be more adaptive to this jumping situation than most of the rest algorithms, as presented in Figure 6c for frame 1101, where FPs are fixed to be close to 0.01. MF perform worst due to its slow adaption to the switch of lights. MoG is also awkward to adapt to changes when comparing to SILK. This is because the update formula of recursive MoG uses a pre-fixed weight parameter for new Gaussians, which might align well with the temporal dynamics at certain situations but fail at others. In contrast, as shown in (20), the weight parameter α_i of our algorithms is made adaptive to a new example, using (17) — a closed-form solution to the optimization problem of (11).

The right panels of Figure 1 present the *Traffic* sequence taken by a camera that shakes irregularly. This results in a switched multi-modal distributions as demonstrated in the bottom-right panel. However, since the shaking motion is neither periodical nor with a constant strength, the sequence turns out to be very challenging for any background subtraction algorithm. As visually presented in Figure 7, SILK still manages to obtain satisfactory results on these key frames.

C. SILK/SILK-GC on Real Sequences

We further evaluate the proposed SILK and SILK-GC on five standard video sequences (displayed in Figure 8) below:

Beach:[4] Multiple foreground people walk through a background beach of moving waves.

Trees:[5] This widely used sequence contains a person walking in front of a waving tree [20].

Lights:[6] This is part of the indoor video sequence of Figure 1 where the lights are turned off and back on during the capture.

Jug:[7] The background of this scene is rippling water and the foreground is a floating jug [21].

[4]Downloaded from http://www.wisdom.weizmann.ac.il/˜vision/Behavior Correlation.html

[5]Downloaded from http://research.microsoft.com/˜jckrumm/wallf lower/testimages.htm

[6]Downloaded from http://http://ttic.uchicago.edu/˜licheng/data/lights_data.avi

[7]Downloaded from http://www.cs.bu.edu/groups/ivc/data.php

Fig. 8: Five datasets: (from left to right) *Beach*, *Trees*, *Lights*, *Jug*, and *Railway*. Top row: first frame in the sequence; Middle row: one key frame; Bottom row: hand-labeled foreground mask.

Railway:[8] A strong breeze causes the camera to jitter [22], which leads to a multi-modal switching background distribution.

In addition to MoG, here the performance of the proposed algorithms is also compared to Sheikh et al. [22].

Figure 9 provides a visual comparison of recursive MoG, Sheikh et al., SILK and SILK-GC, evaluated on these datasets. Overall recursive MoG gives inferior results as it tends to produce noisy output. While SILK adapts quickly to illumination changes *e.g.* for the *Lights* dataset, it nevertheless yields label noises. On the other hand, SILK-GC produces much smoothed segmentation results while still responding quickly to illumination changes. We notice that Sheikh et al. also produce very competitive foreground segments, while in term of preserving the shape or silhouette details, it performs less successful when comparing to SILK-GC. These results empirically support that the proposed approach is capable of dealing with the challenging situations such as illumination changes, dynamic water backgrounds, and camera jitters. In addition, in Figure 11 we visually compare our result to two approaches on the *Jug* sequence, namely the autoregressive method of [21] and the method of [18] that exploits spatial neighbors, as both report only visual results. Our method is also shown to provide visually competitive results.

Both the visual comparison and the quantitative analysis (the ROC curves reported on these testbeds

[8]Downloaded from http://www.cs.cmu.edu/~yaser/new_backgroun dsubtraction.htm

Fig. 9: First row: Results of MoG. Second row: Result of Sheikh et al. [22]. Third row: Results of SILK. Fourth row: Results of SILK-GC, and the results with foreground masks in Fifth row. Compared with SILK, SILK-GC effectively removes noises, and at the same time preserves the detailed shapes of foreground objects, *e.g.* the pedestrian in the *Railway* dataset.

in Figure 10) suggest that our approach perform better or at least comparable to these state-of-the-arts. Moreover, unlike existing approaches, our result is obtained *without* resorting to any preamble pure background images for model training and initialization. We also would like to emphasize that these state-of-the-art methods including Sheikh et al. [22] (11 fps), [21] (0.125 fps), and [18] (no report on fps) are non-realtime methods.

Finally, a simple speed test is conducted to evaluate empirically how the speed of the proposed algorithm scales up with respect to image size, and the influence of different buffer sizes on the processing speed. The computer used here is a Lenovo ThinkStation W500 Laptop with Intel Core2 Duo CPU and ATI Mobility FireGL V5700 GPU. As demonstrated in Figure 12a for SILK and Figure 12b for SILK-GC, the running speed of the proposed algorithms decrease gracefully with respect to the incensement of

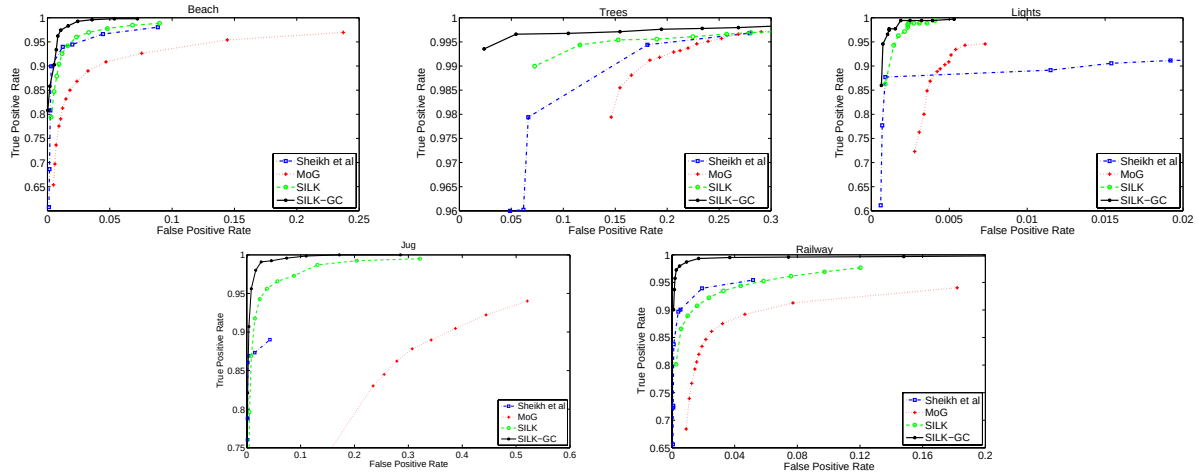


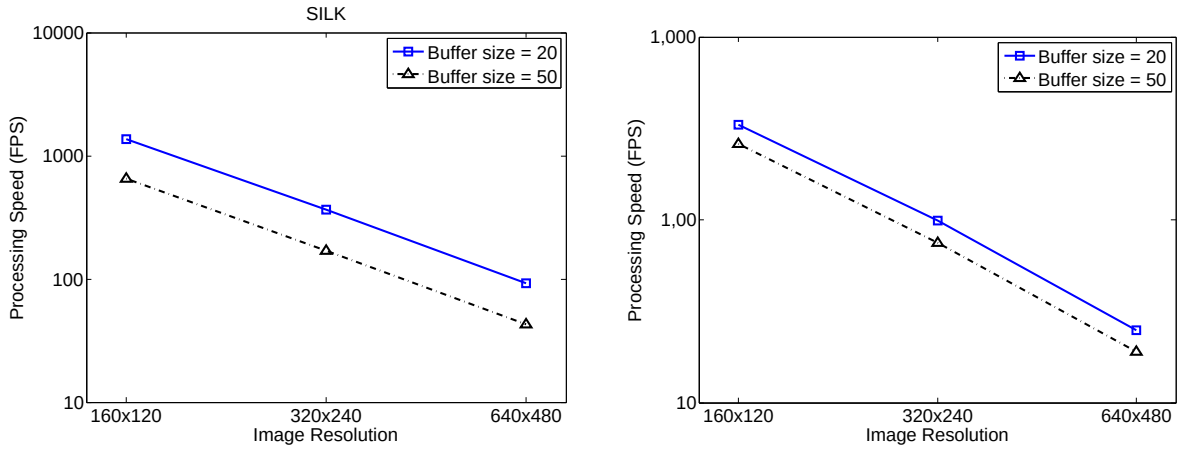
Fig. 10: ROC curves of the proposed SILK vs. SILK-GC algorithms on the *Trees*, *Lights*, *Jug*, and *Railway* datasets, respectively, from top-left to bottom-right in scanline order.

Fig. 11: Comparisons to other recent offline methods on the *Jug* dataset. From left to right: test frame, result from Figure 5 of [21], result from Figure 4 of [18], our result, and ground truth from Figure 4 of [18].

image sizes. Meanwhile, it is also shown that increasing the buffer size from 20 to 50 would result in an increase of runtime for SILK by almost 50%. Meanwhile, the runtime increment for SILK-GC is much less dramatic, since the time for graph-cut calculation is independent of the buffer size.

VII. Outlook and Discussion

In this paper, we explicitly connect the problem of background subtraction to existing work in on-line learning and novelty detection. We also propose a class of on-line discriminative algorithms using kernels to specifically address this problem. Our GPU implementation executes at 170 FPS for SILK and 75 FPS for for SILK-GC, for a typical video sequence of 320×240 resolution. Experiments on video datasets of a variety of scenarios suggest that the results are comparable to the the state-of-the-arts off-line methods [18, 20–22]. As of future work, we plan to extend our algorithm to other interesting scenarios such as segmenting foreground objects under water, in the presence of fog or during the night.



(a) SILK (b) SILK-GC

Fig. 12: This figure demonstrates the scalability of the proposed SILK and SILK-GC algorithms with respect to the varying image size as well as the buffer size.

ACKNOWLEDGMENT

The authors thank Mr. G. Dalley, Dr. J. Krumm, Dr. Y. Sheikh, Dr. S. Sclaroff and and Dr. Z. Zivkovi for sharing their datasets and (or) codes.

REFERENCES

[1] C. Bregler and J. Malik, "Tracking people with twists and exponential maps," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 1998, p. 8.

[2] M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, pp. 5–28, 1998.

[3] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter-sensitive hashing," in *IEEE International Conference on Computer Vision*. Washington, DC, USA: IEEE Computer Society, 2003, p. 750.

[4] A. Agarwal and B. Triggs, "Learning to track 3D human motion from silhouettes," in *International conference on Machine learning*. New York, NY, USA: ACM, 2004, p. 2.

[5] D. M. Gavrila, "The visual analysis of human movement: a survey," *Comput. Vis. Image Underst.*, vol. 73, no. 1, pp. 82–98, 1999.

[6] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Comput. Vis. Image Underst.*, vol. 104, no. 2, pp. 90–126, 2006.

[7] R. Poppe, "Vision-based human motion analysis: An overview," *Comput. Vis. Image Underst.*, vol. 108, no. 1-2, pp. 4–18, 2007.

[8] Q. Shi, L. Wang, L. Cheng, and A. J. Smola, "Discriminative human action segmentation and recognition using semi-markov model," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[9] L. Ren, G. Shakhnarovich, J. K. Hodgins, H. Pfister, and P. Viola, "Learning silhouette features for control of human motion," *ACM Trans. Graph.*, vol. 24, no. 4, pp. 1303–1331, 2005.

[10] L. Cheng, S. Wang, D. Schuurmans, T. Caelli, and S. V. N. Vishwanathan, "An online discriminative approach to background subtraction," in *IEEE international conference on advanced video and signal based surveillance (AVSS)*, 2006.

[11] J. Kivinen and M. K. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors," *Information and Computation*, vol. 132, no. 1, pp. 1–64, January 1997.

[12] B. Scholkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, pp. 1443–1471, 2001.

[13] N. Cesa-Bianchi, A. Conconi, and C. Gentile, "On the generalization ability of on-line learning algorithms," *IEEE Trans. Information Theory*, vol. 50, no. 9, pp. 2050–2057, September 2004.

[14] E. Rivlin, M. Rudzsky, R. Goldenberg, U. Bogomolov, and S. Lepchev, "A real-time system for classification of moving objects," in *International Conference on Pattern Recognition (ICPR)*, 2002.

[15] A. Griesser, S. D. Roeck, A. Neubeck, and L. V. Gool, "GPU-based foreground-background segmentation using an extended colinearity criterion," in *Vision, Modeling, and Visualization*, 2005.

[16] J. Migdal and W. Grimson, "Background subtraction using markov thresholds," in *IEEE Workshop on Motion and Video Computing*, 2005, pp. 58–65.

[17] W. Nam and J. Han, "Motion-based background modeling for foreground segmentation," in *ACM international workshop on Video surveillance and sensor networks*, 2006.

[18] G. Dalley, J. Migdal, and W. Grimson, "Background subtraction for temporally irregular dynamic textures," in *IEEE Workshop on Application of Computer Vision*, 2008.

[19] V. Mahadevan and N. Vasconcelos, "Background subtraction in highly dynamic scenes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[20] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *IEEE International Conference on Computer Vision*, 1999.

[21] J. Zhong and S. Sclaroff, "Segmenting foreground objects from a dynamic textured background via a robust Kalman filter," in *IEEE International Conference on Computer Vision*, 2003.

[22] Y. Sheikh and M. Shah, "Bayesian modeling of dynamic scenes for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 11, pp. 1778–1792, 2005.

[23] C. Wren, A. Azarbayejani, T. Darrell, and A. Pantland, "Pfinder:real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[24] N. Friedman and S. Russell, "Image segmentation in video sequences: A probabilistic approach," in *Proc. Thirteenth Conf. on Uncertainty in Artificial Intelligence*, 1997.

[25] Z. Zivkovic and F. Heijden, "Recursive unsupervised learning of finite mixture models," *IEEE T. Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, 2004.

[26] D. Lee, "Effective gaussian mixture learning for video background subtraction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 827–832, 2005.

[27] C. Stauffer and W. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 747–757, 2000.

[28] A. Monnet, A. Mittal, N. Paragios, and V. Ramesh, "Background modeling and subtraction of dynamic scenes," in *IEEE International Conference on Computer Vision*, 2003.

[29] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *European Conference on Computer Vision*, 2000.

[30] R. Duda and P. Hart, *Pattern classification and scene analysis*. New York: Wiley, 1973.

[31] A. Mittal and N. Paragios, "Motion-based background subtraction using adaptive kernel density estimation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.

[32] B. Han, D. Comaniciu, , Y. Zhu, and L. Davis, "Incremental density approximation and kernel-based bayesian filtering for object tracking," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.

[33] H. Lin, T. Liu, and J. Chuang, "A probabilistic svm approach for background scene initialization," in *International Conference on Image Processing*, 2002, pp. 893–896.

[34] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov, "Bilayer segmentation of live video," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[35] T. Parag, A. Elgammal, and A. Mittal, "A framework for feature selection for background subtraction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[36] Z. Hao, W. Wen, Z. Liu, and X. Yang, "Rea-ttime foreground background segmentation using adaptive support vector machine algorithm," in *International Conference on Artificial Neural Networks (ICANN)*, 2007.

[37] A. Ulges and T. Breuel, "A local discriminative model for background subtraction," in *Symposium of the German Association for Pattern Recognition (DAGM)*, 2008.

[38] L. Cheng, S. V. N. Vishwanathan, D. Schuurmans, S. Wang, and T. Caelli, "Implicit online learning with kernels," in *NIPS*. Cambridge MA: MIT Press, 2006.

[39] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[40] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, 2001.

[41] B. Scholkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge, MA: MIT Press, 2002.

[42] J. Kivinen, A. Smola, and R. Williamson, "Online learning with kernels," *IEEE Trans. on Signal Processing*, vol. 52, no. 8, pp. 2165–2176, 2004.

[43] N. Schraudolph, "Fast curvature matrix-vector products for second-order gradient descent," *Neural Comput.*, vol. 14, no. 7, pp. 1723–1738, 2002.

[44] J. Kivinen, M. Warmuth, and B. Hassibi, "The p-norm generalization of the lms algorithm for adaptive control," *IEEE Trans. Signal Processing*, vol. 54, no. 5, pp. 1782–1793, 2006.

[45] O. Dekel, S. Shalev-Shwartz, and Y. Singer, "The forgetron: A kernel-based perceptron on a budget," *SIAM J. Comput.*, vol. 37, no. 5, pp. 1342–1372, 2008.

[46] K. Crammer, J. Kandola, and Y. Singer, "Online classification on a budget," in *Neural Information Processing Systems*. MIT Press, 2003.

[47] J. Weston, A. Bordes, and L. Bottou, "Online (and offline) on an even tighter budget," in *International Workshop on Artificial Intelligence and Statistics (AISTAT)*. Society for Artificial Intelligence and Statistics, 2005.

[48] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile, "Tracking the best hyperplane with a simple budget perceptron," *Mach. Learn.*, vol. 69, no. 2-3, pp. 143–167, 2007.

[49] "Exact maximum a posteriori estimation for binary images," *Journal of the Royal Statistical Society Series B*, vol. 51, no. 2, pp. 271–279, 1989.

[50] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE T. Pattern Analysis*

and Machine Intelligence, vol. 23, no. 11, pp. 1222–1239, 2001.

[51] N. Dixit, R. Keriven, and N. Paragios, "GPU-cuts: Combinatorial optimisation, graphic processing units and adaptive object extraction," CERTIS, Ecole Nationale des Ponts et Chaussees, Tech. Rep., 2005.

[52] A. Goldberg and R. Tarjan, "A new approach to the maximum flow problem," in *ACM Symposium on Theory of Computing (STOC)*, 1986.

[53] Z.Zivkovic and F. der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, 2006.

APPENDIX

Proof of Lemma 1

Proof: (**sketch**) To prove (24), it is sufficient to show

$$\begin{aligned}
&\text{LHS}\\
&\leq \left|\sum_{i=1}^{t-1}(1-\tau)^{t-i}\alpha_i^{(i)} - \sum_{j=1,i\in\hat{N}(j)}^{\omega}(1-\tau)^{t-i}\alpha_i^{(i)}\right| X^2\\
&\leq \sum_{i=1}^{t-\omega}(1-\tau)^{t-i}(1-\tau)CX^2\\
&= (1-\tau)^{\omega+1}CX^2\\
&\quad \left((1-\tau)^{t-\omega-1} + (1-\tau)^{t-\omega-2} + \cdots + (1-\tau)^0\right)\\
&\leq \text{RHS}.
\end{aligned}$$

■

The Convergence Theorem

Theorem 2: Let $(x_t, y_t)_{t=1}^T$ be an arbitrary sequence of examples such that $k(x_t, x_t) \leq X^2$ holds for any t. Furthermore, assume that the loss function $L(x_t, y_t, f)$ is Lipschitz continuous in $f(x_t)$. Let $(f_1, \ldots, f_T)$ be the sequence of hypothesis produced by (S)ILK with learning rate $\eta_t = \eta t^{-1/2}$, $\sum_{t=1}^T R(x_t, y_t, f_t)$ the cumulative risk of this sequence, and $R_{\text{bth}}(g)$ the batch regularized risk of $(g, \ldots, g)$, for any $g \in \mathcal{H}$. Then

$$\frac{1}{T}\sum_{t=1}^{T} R(x_t, y_t, f_t) \;\leq\; R_{\text{bth}}(g) + aT^{-1/2} + bT^{-1}, \tag{28}$$

where $U = CX\lambda^{-1}$ and $b = U^2(2\eta)^{-1}$. For ILK $a = 4\eta C^2X^2 + 2U^2\eta^{-1}$, while for SILK $a = 4(1-\tau)(1+2/\tau)C^2X^2 + 2U^2(1-\tau)^{-1}$.

In particular, if

$$g^* \;=\; \arg\min_{g\in\mathcal{H}} R_{\text{bth}}(g),$$

we obtain

$$\frac{1}{T}\sum_{t=1}^{T} R(x_t, y_t, f_t) \;\leq\; R_{\mathrm{bth}}(g^*) + O(T^{-1/2}).$$

We omitted the detailed proof as it essentially follows that of Theorem 4 in [42]. To further understand this theorem, we first notice that the risk function $R_{\mathrm{bth}}(g)$ has a global minimizer as being a convex function of g. If risk function converges as $\frac{1}{T}\sum_{t=1}^{T} R(x_t, y_t, f_t) \to R_{\mathrm{bth}}(g^*)$, we would expect to have the minimizer $f \to g^*$. In addition, the minimum cumulative risk of batch learning $R_{\mathrm{bth}}(g^*)$ is itself an upper-bound of the minimum expected risk $R_{\mathrm{bth}}(f^*) \triangleq \min_f \mathbb{E}_{(x,y)\sim P(x,y)} R_{\mathrm{bth}}(x, y, f)$ [42]. As stated in [39] for the structured risk minimization framework, as the sample size T grows ($T \to \infty$), we obtain $g^* \to f^*$ in probability. This subsequently guarantees the convergence of the average regularized risk of ILK to $R(f^*)$.