

Adaptive Exponential Smoothing for Online Filtering of Pixel Prediction Maps

Kang Dang, Jiong Yang, Junsong Yuan
 School of Electrical and Electronic Engineering,
 Nanyang Technological University, Singapore, 639798
 {dang0025, yang0374}@e.ntu.edu.sg, jsyuan@ntu.edu.sg

Abstract

We propose an efficient online video filtering method, called adaptive exponential filtering (AES) to refine pixel prediction maps. Assuming each pixel is associated with a discriminative prediction score, the proposed AES applies exponentially decreasing weights over time to smooth the prediction score of each pixel, similar to classic exponential smoothing. However, instead of fixing the spatial pixel location to perform temporal filtering, we trace each pixel in the past frames by finding the optimal path that can bring the maximum exponential smoothing score, thus performing adaptive and non-linear filtering. Thanks to the pixel tracing, AES can better address object movements and avoid over-smoothing. To enable real-time filtering, we propose a linear-complexity dynamic programming scheme that can trace all pixels simultaneously. We apply the proposed filtering method to improve both saliency detection maps and scene parsing maps. The comparisons with average and exponential filtering, as well as state-of-the-art methods, validate that our AES can effectively refine the pixel prediction maps, without using the original video again.

1. Introduction

Despite the success of pixel prediction, *e.g.*, saliency detection and parsing in individual images, its extension to video pixel prediction remains a challenging problem due to the spatio-temporal structure among the pixels and the huge computations to analyze the video data. For example, when each video frame is parsed independently, the per-pixel prediction maps are usually “flickering” due to the spatio-temporal inconsistencies and noisy predictions, *e.g.*, caused by object and camera movements or low quality videos. Thus an efficient online filtering of the pixel prediction maps is important for many streaming video analytics applications.

To address the “flickering” effects, enforcing spatio-temporal smoothness constraints over the pixel predictions can improve the quality of the prediction maps [25, 8, 12, 9]. However, existing methods still have difficulty in providing a solution that is both efficient and effective. On the one hand, despite a lot of previous works [19, 22] on real-time video denoising, they are designed to improve the video quality rather than its pixel prediction maps. It is worth

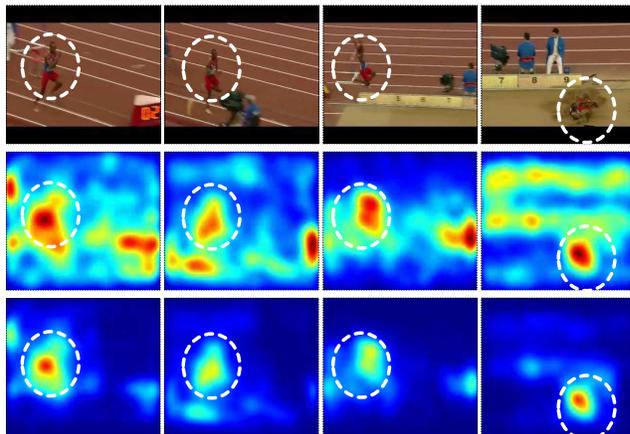


Figure 1: We propose a spatio-temporal filtering framework, to refine the per-frame prediction maps from an image analysis module. Top Row: input video. Middle Row: per-frame prediction maps. Bottom Row: refined maps by our filter.

noting that linear spatio-temporal filtering methods such as moving average or exponential smoothing that works well for independent additive video noises may not provide satisfactory results on the pixel prediction maps, which are usually affected by non-additive and signal dependent noises. Thus special spatio-temporal filtering methods are required to deal with them. On the other hand, although a few spatio-temporal filtering methods have been proposed to refine pixel prediction maps, most of them only perform in an offline or batch mode where the whole video is required to perform the smoothing [14, 17, 24, 2]. Although a few recent works have been developed for online video filtering, they usually rely on extra steps, such as producing temporally consistent superpixels from a streaming video [9], or leveraging metric learning and optical flow [25], thus are difficult to be implemented in real-time.

To address the above limitations, in this paper we propose an efficient online video filtering method which is able to perform online and real-time filtering. Given a sequence of pixel prediction maps, where each pixel is associated with a detection score or a probabilistic multi-class distribution, our goal is to provide a causal filtering that can

improve the spatio-temporal smoothness of the pixel prediction maps thus to reduce the “flickering” effects. Our method is motivated by the classic exponential smoothing, as we also apply exponentially decreasing weights over time to smooth the prediction score of each pixel. However, instead of fixing the pixel location to perform temporal filtering, for each pixel, we firstly search for a smoothing path of maximum score that traces this pixel over past frames, and then perform temporal smoothing over the found path. To find the path for each pixel, we rely on the pixel prediction maps. For example, if a pixel truly belongs to the car category, it should be easily traced back to the pixels in previous frames that also belong to the car category.

For efficient online implementation, instead of performing pixel tracing for each individual pixel, we propose a dynamic programming algorithm that can trace all pixels simultaneously with only linear complexity in terms of the total number of pixels. It guarantees to obtain the optimal paths for all pixels, and only needs to keep the most recent pixel prediction values for online filtering. Thanks to the pixel tracing, our method can better address object or camera movements when performing spatio-temporal filtering. Moreover, similar to exponential smoothing, our method can well address false alarms, *i.e.*, pixels with high prediction score but low exponential smoothing score, as well as missing detections, *i.e.*, pixels with low prediction score but high exponential smoothing score. We also discuss the relationship between the proposed filtering method and the existing filtering methods and show that they are actually the special cases of the proposed method.

We perform two different streaming video analytics tasks, *i.e.*, online saliency map filtering and online multi-class scene parsing, to evaluate the performance. We achieve more than 55 frames per second for a video of size 320×240 . The excellent performance compared with the state-of-the-art methods validates the effectiveness and efficiency of the proposed spatio-temporal filtering for video analytics applications.

2. Related Work

Our work is inspired by classical linear casual filters, *e.g.*, spatio-temporal exponential filter. While these filters can well suppress additive noises in static backgrounds, they usually tend to overly smooth moving objects. To better deal with moving objects, previous methods [25, 27] restrict support of the filters by optical flow connection. Also, [25] applies metric learning so that filtering is performed adaptively according to the learned appearance similarities. While effective, the methods are computationally intensive. In contrast, our method is also an extension of exponential filter but with two important differences: (1) no appearance or motion information is needed by our method, and (2) the computational cost is much smaller.

Probabilistic graphical models [16, 12, 8, 35] are also used to perform on-line spatio-temporal smoothing. In these models, labeling consistency among the frames is en-

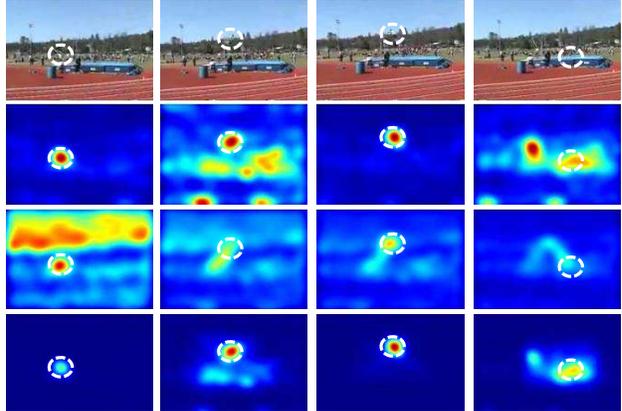


Figure 2: Exponential filter (Eq. 2) can introduce significant tailing artifacts when filtering fast-moving pixels. First row: input video. 2nd Row: per-frame prediction maps. 3rd Row: refined maps by exponential filter. Bottom Row: refined maps by our filter.

forced via pairwise edge terms. To satisfy the online requirement, some of them restrict the message passing from past to current frame [12, 8, 35]. While they yield good performance, efficient inference over large graphical model is still a challenging problem. In addition, they only provide discretized labeling results without retaining the confidence scores. However, as argued in [25], confidence scores are useful for certain applications thus it is preferable that the filtering can directly refine the prediction scores.

Different from the above, which perform online filtering based on existing per-frame prediction maps, online super-voxels methods [9, 36, 39, 38, 15, 21, 28] can be used to enforce spatio-temporal consistency during prediction step. However, even with the spatio-temporal consistent super-voxels, inconsistent predictions may still occur, thus filtering may be needed.

Our work is also related to max-path formulation for video event detection [32, 40]. However [32] only needs to find the max path among all paths while our target is to denoise the dense maps thus we need to trace each individual pixel. The formulation in [40] is more relevant to moving average, in contrast our work generalizes classic exponential smoothing. Furthermore, our work is related to the offline techniques which model spatio-temporal structure among pixels [14, 17, 24, 41, 27, 37, 2] and video denoising [19, 22]. It should be noted most video denoising methods are mainly designed for appearance denoising, rather than noises introduced from classifier output, *i.e.*, prediction maps denoising.

3. Proposed Method

We denote a video sequence as $\mathcal{S} = \{I_1, I_2, \dots, I_T\}$, where I_k is a $W \times H$ image frame. For each spatio-temporal location (x, y, t) , we assume that a prediction score $U(x, y, t)$ is provided by an independent image analy-

sis module. As the pixel scores are generated independently per frame, they do not necessarily enforce the temporal consistency across frames. So filtering is needed to refine the pixel prediction maps. We first explain two classical linear filters below:

Moving Average (Ave) [18] :

$$M(x, y, t) = \frac{1}{\delta_T} \sum_{i=t-\delta_T}^t U(x, y, i). \quad (1)$$

Exponential Smoothing (Exp) [7]:

$$\begin{aligned} M(x, y, t) &= \\ &\alpha \times M(x, y, t-1) + (1-\alpha) \times U(x, y, t) \\ &= \alpha^{(t-1)} U(x, y, 1) + (1-\alpha) \sum_{i=2}^t \alpha^{(t-i)} U(x, y, i) \\ &\approx (1-\alpha) \sum_{i=1}^t \alpha^{(t-i)} U(x, y, i). \end{aligned} \quad (2)$$

Here $M(x, y, t)$ is the filtered response, δ_T and α are temporal smoothing bandwidth for moving average and temporal weighting factor for exponential smoothing, respectively. The approximation error in Eq. 2 decays exponentially with respect to t . Unlike moving average which assigns the equal weight for input scores within a temporal window, exponential filtering weights input scores in an exponentially decreasing manner.

When applying to videos, these filters operate along a fixed pixel location (x, y) to perform temporal smoothing. As a result, they can easily overly smooth fast-moving pixels and cause tailing artifacts as shown in Fig. 2. To better handle moving pixels, a good spatio-temporal filter should be able to adapt to different pixels, so the temporal smoothing will be less likely to overly smooth moving pixels. The above observation motivates us to propose an adaptive smoothing that is pixel dependent.

3.1. Adaptive Exponential Smoothing (AES)

We assume each spatio-temporal location $v_t = (x, y, t)$ is associated with a discriminative prediction score $U(v_t)$. For example, a high positive score $U(v_t)$ implies a high likelihood that the current pixel belongs to the target class, while a high negative score indicates a low likelihood. To better explain the proposed AES, we represent the video as a 3-dimensional trellis $W \times H \times T$ denoted by \mathcal{G} . For each pixel $v = v_t$, we trace it in the past frames to obtain a path $\mathcal{P}_{s \rightarrow t}(v_t) = \{v_i\}_{i=s}^t$ in \mathcal{G} . Here i is the frame index, and v_i is a pixel at frame i . The path $\mathcal{P}_{s \rightarrow t}(v_t)$ satisfies the spatio-temporal smoothness constraints: $x_i - R \leq x_{i+1} \leq x_i + R$, $y_i - R \leq y_{i+1} \leq y_i + R$ and $t_{i+1} = t_i + 1$ where R represents the spatial neighborhood radius, *i.e.*, $(x_{i+1}, y_{i+1}) \in \mathcal{N}(x_i, y_i) = [x_i - R, x_i + R] \times [y_i - R, y_i + R]$.

Instead of performing temporal exponential smoothing at fixed spatial location, for each pixel, we propose to trace

it back in the past frames by finding its origin, such that the pixels of the found path are more likely to belong to the same label category. So the temporal smoothing will be less likely to blend the prediction scores of different classes. To perform pixel tracing, the exponential smoothing score of pixel v_t is used as the pixel's tracing score, which is defined as the weighted accumulation score of all the pixels along the path $\mathcal{P}_{s \rightarrow t}(v_t)$:

$$M(\mathcal{P}_{s \rightarrow t}(v_t)) = \sum_{i=s}^t \alpha^{(t-i)} U(v_i), \quad (3)$$

where α is the temporal weighting factor ranging from 0 to 1. Similar to exponential smoothing, to filter the current location v_t , we assign the previous score $U(v_i)$ a smaller weight according to its "age" in the path, *i.e.*, $t - i$.

As the exponential smoothing score $M(\mathcal{P}_{s \rightarrow t}(v_t))$ stands for the accumulative evidence to current label and we want to find a path whose pixels are more likely to have the same label, we formulate the filtering problem as finding the path that maximizes the exponential smoothing score:

$$\mathcal{P}_{s \rightarrow t}^*(v_t) = \arg \max_{\mathcal{P}_{s \rightarrow t}(v_t) \in \text{path}(\mathcal{G}, v_t)} M(\mathcal{P}_{s \rightarrow t}(v_t)), \quad (4)$$

where $\mathcal{P}_{s \rightarrow t}^*(v_t)$ is a pixel dependent path to perform exponential smoothing, and $\text{path}(\mathcal{G}, v_t)$ refers to the set of all the candidate paths that end at v_t . Based on the formulation, the maximum exponential smoothing score is used as the pixel's filtered score, *i.e.*, $M(x, y, t) = M(\mathcal{P}_{s \rightarrow t}^*(v_t))$. A pixel with high isolated positive score but low exponential smoothing score, *i.e.*, $U(v_t)$ is high but $M(\mathcal{P}_{s \rightarrow t}^*(v_t))$ is low, will be treated as false positive, and a pixel with low isolated negative score but high exponential smoothing score, *i.e.*, $U(v_t)$ is low but $M(\mathcal{P}_{s \rightarrow t}^*(v_t))$ is high, will be treated as false negative. In addition, as the individual score $U(v_i)$ can be either positive or negative, it is not necessary that a longer path is better. The length of the path actually adaptively determines the temporal smoothing bandwidth, which may well address missing detections and false alarms.

3.2. Online Pixel Filtering

A brute-force way to solve the pixel-tracing problem defined in Eq. 4 is time consuming, because the starting frame number s of all candidate paths range from 1 to t , so the search space is large for even a single pixel location v_t , *i.e.*, $O((2R+1)^T)$, where T is the video length. To achieve better efficiency, we propose an efficient online filtering algorithm based on dynamic programming, that can trace all pixels simultaneously with a linear complexity. By Eq. 3

	Objective	Conditions
Moving Average Filtering (Ave)	$M(x, y, t) = \frac{1}{\delta_T} \sum_{i=t-\delta_T}^t U(x, y, i)$	$\alpha = 1, R = 0,$ $s = t - \delta_T$
Spatio-Temporal Moving Average Filtering (ST-Ave)	$M(x, y, t) = \frac{1}{\delta_T} \sum_{i=t-\delta_T}^t U'(x, y, i)$	$\alpha = 1, R = 0,$ $s = t - \delta_T$
Exponential Filtering (Exp)	$M(x, y, t) \approx (1 - \alpha) \sum_{i=1}^t \alpha^{(t-i)} U(x, y, i)$	$R = 0, s = 1$
Spatio-Temporal Exponential Filtering (ST-Exp) [25] (online)	$M(x, y, t) \approx (1 - \alpha) \sum_{i=1}^t \alpha^{(t-i)} U'(x, y, i)$	$R = 0, s = 1$
Adaptive Exponential Smoothing (AES)	$M(\mathcal{P}_{s \rightarrow t}^*(v_t)) = \sum_{i=s}^t \alpha^{(t-i)} U(v_i)$	-

Table 1: Relationship between our AES and other online filtering methods. Parameters α and R stand for the temporal weighting factor and the spatial neighborhood radius, respectively. s stands for starting location of the path. $U'(x, y, t) = \frac{1}{K} \sum_{\delta x=-\delta}^{\delta x=+\delta} \sum_{\delta y=-\delta}^{\delta y=+\delta} U(x + \delta x, y + \delta y, t)$, where δ is the spatial smoothing bandwidth. $K = (2 \times \delta + 1)^2$ is a normalization factor.

and Eq. 4, the pixel tracing objective can also be written as:

$$M(\mathcal{P}_{s \rightarrow t}^*(v_t)) = \max_{1 \leq s \leq t} \left\{ \max_{\substack{\forall s < i \leq t-1, \\ (x_i, y_i) \in \mathcal{N}(x_{i+1}, y_{i+1})}} \sum_{i=s}^{t-1} \alpha^{(t-i)} U(v_i) \right\} + U(v_t). \quad (5)$$

The outer maximization searches for the optimal starting frame and the inner maximization searches for the optimal path from the starting frame s to the current frame.

As explained before, for each pixel v_t , we try to trace it back to its origin v_s in the past frames, such that the exponential smoothing score along the path $\mathcal{P}_{s \rightarrow t}^*(v_t)$ is maximized. Instead of performing back-tracing, our idea is to perform forward-tracing using dynamic programming, such that all pixels are traced simultaneously. It is in spirit similar to the max-path search in [32, 40]. We explain the idea of our dynamic programming using the following lemma¹:

Lemma 3.1. $\widehat{M}(v_t)$ in Eq. 6 is identical to $M(\mathcal{P}_{s \rightarrow t}^*(v_t))$ defined by Eq. 5.

$$\widehat{M}(v_t) = \max \left\{ \max_{(x_{t-1}, y_{t-1}) \in \mathcal{N}(x_t, y_t)} \alpha \times \widehat{M}(v_{t-1}), 0 \right\} + U(v_t). \quad (6)$$

Let $\widehat{M}(v_{t-1}^*)$ indicates the maximized score from v_t 's neighbors in the previous frame, *i.e.*, $\widehat{M}(v_{t-1}^*) = \max_{(x_{t-1}, y_{t-1}) \in \mathcal{N}(x_t, y_t)} \widehat{M}(v_{t-1})$. If it is positive, we

¹The proof can be obtained from: <https://sites.google.com/site/kangdang/>.

propagate it to the current location v_t . Otherwise a new path is started from v_t , because connecting to any neighbor in previous frame will decrease its score. Based on the lemma we implement the algorithm in Alg. 1.

Algorithm 1 Adaptive Exponential Filtering

- 1: $\widehat{M}(v_1) \leftarrow U(v_1), \forall (x_1, y_1) \in [1, W] \times [1, H]$
 - 2: **for** $t \leftarrow 2$ to n **do**
 - 3: **for all** $(x_t, y_t) \in [1, W] \times [1, H]$ **do**
 - 4: $\widehat{M}(v_{t-1}^*) \leftarrow \max_{(x_{t-1}, y_{t-1}) \in \mathcal{N}(x_t, y_t)} \widehat{M}(v_{t-1})$
 - 5: $\widehat{M}(v_t) \leftarrow \max \left\{ \alpha \times \widehat{M}(v_{t-1}^*), 0 \right\} + U(v_t)$
-

The time complexity of our algorithm is linear in terms of the number of pixels, *i.e.*, $O(W \times H)$ for one frame and $O(W \times H \times T)$ for the whole video. Due to its simplicity, the computational cost of our algorithm is identical to the classical linear filters, *e.g.*, temporal moving average or exponential filter.

3.3. Filtering Multi-Class Pixel Prediction Maps

Our method can be easily extended to filter multi-class pixel prediction maps. In this case each pixel has K prediction scores of K classes denoted by $U(v, c)$, where $c = 1, \dots, K$ refers to class labels. Correspondingly there are K paths $\mathcal{P}_{s \rightarrow t}(v_t, c)$ and path scores $M(\mathcal{P}_{s \rightarrow t}(v_t, c), c)$, by performing pixel tracing independently for all K classes. The final classification can be determined by “winner-taking-all” strategy below:

$$c^* = \arg \max_{c \in \{1, \dots, K\}} \max_{\mathcal{P}_{s \rightarrow t}(v_t, c) \in \text{path}(\mathcal{G}, v_t)} M(\mathcal{P}_{s \rightarrow t}(v_t, c), c). \quad (7)$$

However, as the filtering scores of different classes are independently calculated, the obtained filtering scores may not be directly comparable. We can train a classifier such as linear SVM or logistic regression to further perform score calibration [23, 31].

If the image analysis module produces a probabilistic score $Pr(c | v)$ rather than a discriminative score, where $\sum_{k=1}^K Pr(c = k | v) = 1$, we convert it into the discriminative score by subtracting a small offset: $U(v, c) = Pr(c | v) - 1/K$, where K is the number of classes.

3.4. Relationship with Other Filtering Methods

The proposed AES is a generalization of existing online filtering approaches such as classic exponential smoothing and moving average. In Table 1, we show how Eq. 3 can be converted into the objective functions of other filtering methods under different settings. For example: when setting neighborhood radius $R = 0$, temporal weighting factor $\alpha = 1$ and forcing path starting location $s = t - \delta_T$, our method can become a moving average. When setting $R = 0$ and forcing $s = 1$, our method can approximately become an exponential filtering. As moving average or exponential filtering become the special cases of our framework, with proper parameter settings, our method is guaranteed to be the same or better than these methods in performance. We will verify this claim from experimental comparisons.

4. Experiments

4.1. Online Filtering of Saliency Map

In this experiment, we evaluate our method on saliency map filtering using UCF 101 and SegTrack datasets.

UCF 101 [30]. UCF101 is an action recognition dataset and the per-frame bounding-box annotations are provided for 25 out of 101 categories. To obtain the pixel-wise annotations for evaluation, we label the pixels inside the annotated bounding boxes as ground truth. We use all the action categories except for “Basketball”, because its ground-truth annotations are excessively noisy. We randomly pick 50% of the videos for each category and down-sample the frames to 160×120 for computational efficiency. In total we have evaluated 1599 video sequences.

SegTrack [33]. SegTrack dataset is popular in object tracking [33] and detection [27]. It contains 5 short video sequences² and the per-frame pixel-wise annotations are provided to denote the primary object in each video.

To obtain the initial dense saliency map estimations, we use the phase discrepancy method [42] for UCF 101 dataset and the inside-outside map [27] for SegTrack dataset. We use F-measure to evaluate the saliency map quality on both datasets. Let S_d and S_g denote the detected pixel-wise saliency map and the ground truth, respectively, the F-measure can be computed as F-measure =

²We exclude the penguin sequence by following the setup of [27]. Penguin contains multiple foreground objects but it does not label every one. It is mainly for object tracking and not suitable for saliency map evaluation.

Original	Ave	ST-Ave	Exp	ST-Exp	[40]	Ours
30.6	30.6	30.7	30.6	31.6	31.0	36.2

Table 2: Saliency map filtering on UCF 101.

	Birdfall	Cheetah	Girl	Monkey	Parachute	Mean
Original	62.6	36.1	37.1	17.0	78.4	46.2
Ave	64.9	35.5	37.1	20.2	77.8	47.1
ST-Ave	59.7	40.5	42.1	24.4	76.1	48.5
Exp	67.1	36.3	36.8	21.1	78.6	48.0
ST-Exp	60.7	40.1	42.0	23.7	76.2	48.5
[40]	62.3	38.0	37.7	17.2	78.3	46.7
Ours	65.3	40.1	44.5	22.9	78.8	50.3

Table 3: Saliency map filtering on SegTrack.

$$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \text{ where precision} = \frac{\text{Trace}(S_d^T S_g)}{1 S_d 1^T} \text{ and recall} = \frac{\text{Trace}(S_d^T S_g)}{1 S_g 1^T}.$$

We compare our method with all the filters mentioned in Table 1 as well as [40]. It is interesting to note that our method dilates the modes of the saliency maps as shown in the 6th column of Figure 6. If a more compact map is preferred, we can take a further step to normalize the filtered saliency map to $[0, 1]$ and multiply it with the original saliency map. As shown in the 7th column of Figure 6, it can select the modes from the original saliency maps.

All the parameters of the proposed and the baseline methods are selected by grid search. We fix the parameters within each dataset for all the experiments. The quantitative evaluations are shown in Table 2 and Table 3. Some qualitative examples are also shown in Figure 4 and Figure 6. From the results, we can see that the proposed filtering method can significantly improve the saliency map quality and is better than the compared baseline methods especially for UCF 101 dataset. Compared with SegTrack dataset, UCF 101 dataset contains fast motions such as jumping and diving, thus our method can work more effectively. In contrast to the baseline approaches, the proposed method uses non-straight filtering path, hence can cope with these challenges. We also find that the original saliency maps of UCF 101 are usually disrupted by false alarms while SegTrack maps contain a lot of miss detections. The better performance on both datasets shows that our method can deal with both types of imperfections.

We have also evaluated the effects of the parameter variations of our method, *i.e.*, the temporal weighting factor α and the spatial neighborhood radius R , and the results are shown in Figure 5. An interesting observation is the steep performance dropping when the temporal weighting factor α is increased from 0.9 to 1.0. This is due to the amplifying effect of the exponential function, *i.e.*, $0.9^{15} \approx 0.2$ while $1^{15} = 1$, and it further validates the importance of the temporal weighting factor α . As [40] does not consider the temporal weighting, it becomes similar³ to our special

³Different from our method, [40] applies certain heuristic procedures such as multiplication of the pixel tracing score and the original score to pull up the performance.

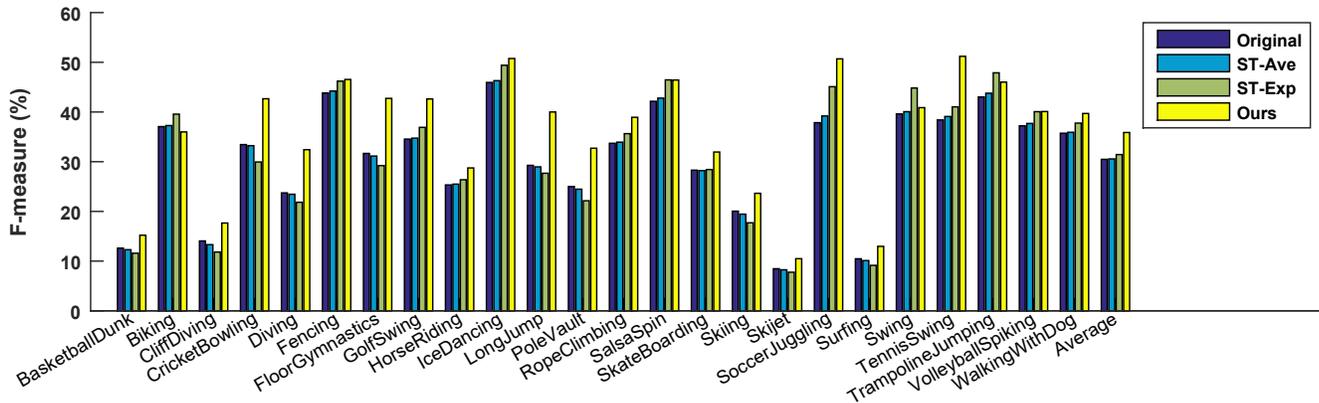


Figure 3: Per-category results on UCF101.

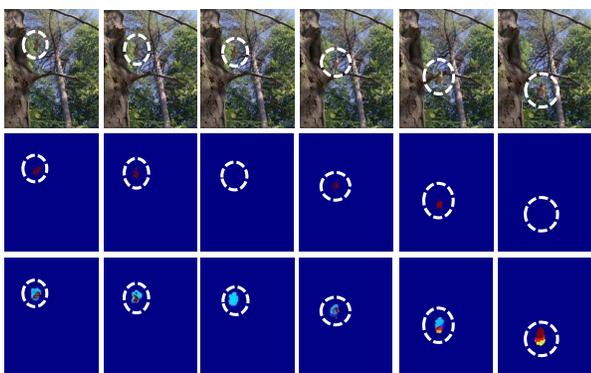


Figure 4: Qualitative results of saliency map filtering on SegTrack. Top Row: input video. Middle Row: per-frame saliency maps. Bottom Row: filtered saliency maps by our filter.

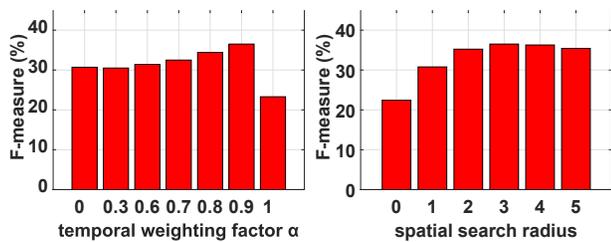


Figure 5: Parameter sensitivity evaluation of the proposed method on UCF 101. The vertical axis is the mean F-measure of all the selected categories. The temporal weighting factor is evaluated by fixing the spatial search radius to 3 and the spatial search radius is evaluated by fixing the temporal weighting factor to 0.9.

case of $\alpha = 1$ which overly emphasizes the past frames. This may partially explain its weak performance as shown in Table 2 and Table 3.

The filtering algorithm is implemented in C++ and the experiments are conducted on a laptop with Intel Core i7 processor. Our code runs at around 450 frames per second when the input size is 160×120 and the spatial neighborhood radius is 3.

4.2. Online Filtering of the Scene Parsing Map

	Original	Ave	ST-Ave	Exp	ST-Exp	[40]	Ours
NYU	71.1 (28.3)	74.5 (28.1)	75.4 (28.0)	74.8 (28.8)	75.8 (28.3)	72.1 (29.2)	76.6 (28.3)
MPI	93.1 (74.6)	93.6 (76.0)	93.6 (76.0)	93.6 (76.0)	93.6 (76.0)	94.3 (77.4)	94.3 (77.1)
01TP	77.8 (40.6)	78.8 (41.4)	81.0 (43.5)	79.2 (41.9)	80.8 (43.1)	78.3 (40.9)	81.7 (44.5)
05VD	84.6 (47.8)	85.2 (47.8)	85.2 (47.8)	85.4 (48.1)	85.4 (48.1)	84.7 (47.5)	85.4 (48.2)

Table 4: Comparisons with baseline filtering methods. Numbers outside and within brackets are per-pixel accuracies, and average per-class IOU scores, respectively.

	NYU	MPI	01TP	05VD
ST-Exp + Flow	75.9 (29.2)	93.7 (76.1)	78.9 (42.2)	86.1 (48.5)
[25]	75.3 (29.6)	93.6 (74.2)	-	86.9 (50.0)
Offline MRF	75.3 (28.1)	94.8 (79.7)	80.2 (42.5)	85.3 (47.6)
Ours	76.6 (28.3)	94.3 (77.1)	81.7 (44.5)	85.4 (48.2)

Table 5: Comparisons with optical flow guided spatio-temporal exponential filtering, Miksik et al. [25] and offline MRF on scene parsing map filtering. Numbers outside and within brackets are per-pixel and average per-class IOU scores, respectively.

In the second experiment, we evaluate our approach on online filtering of scene parsing maps of four videos⁴. NYU is a video of 74 annotated frames with 11 semantic labels captured from a hand-held camera. The initial scene parsing maps are generated from a deep-learning architecture [13]. MPI [34] consists of 156 annotated frames with 5 semantic labels captured from a dashboard-mounted camera. The initial scene parsing maps are obtained from a boosted classifier [35]. CamVid-05VD and CamVid-01TP videos [6] contain 5100 and 1831 frames taken at 30 Hz during day-

⁴The initial scene parsing maps of NYU, MPI and CamVid-05VD videos can be obtained from <http://www.cs.cmu.edu/~dmunoz/projects/onboard.html>.

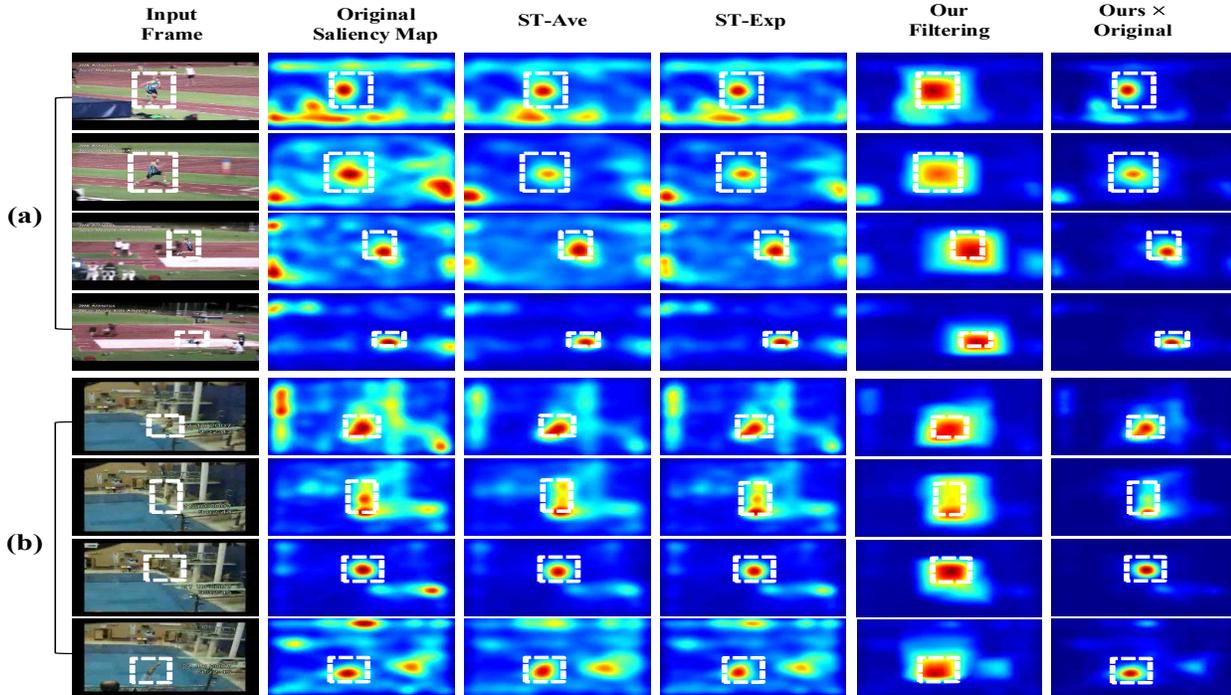


Figure 6: Qualitative results of saliency map filtering on UCF 101.



Figure 7: Image results of NYU dataset. Top: original prediction. Bottom: temporally smoothed using our method. Inconsistent regions are highlighted.

light and dusk, respectively. The sequences are sparsely labeled at 1 Hz with 11 semantic labels. To perform initial scene parsing, we use hierarchical inference machine [26] for CamVid-05VD and location constraint SVM classifier [11] for CamVid-01TP. Because the four videos use different scene parsing algorithms, the noise patterns of their initial maps are also distinguishably different. Therefore for both the proposed method and all the baseline methods, for each video we use a different set of parameters obtained by grid search.

As the scene parsing maps contain multi-class annotations, we have to run our filtering algorithm multiple times (e.g., 11 times for NYU) and use the “winner-taking-all” strategy to determine the filtered label. To reduce computational cost, we extract around 5000 superpixels on each frame using SLIC algorithm [1]. All the following operations are then performed on the superpixel level, which provides a significant speedup. On average the superpixel extraction runs at 50 frames per second, and the filtering runs at 15 frames per second. We can further accelerate the fil-

	Building	Car	Door	Person	Pole	Road	Sidewalk	Sign	Sky	Tree	Window	Ave
Per-Frame	45.2	42.4	1.1	18.4	6.8	81.0	16.9	3.2	9.8	78.8	8.0	28.3
[25]	52.0	52.5	0.0	16.9	5.8	83.0	12.8	0.0	9.8	83.0	9.4	29.6
Ours	54.6	55.3	0.0	19.7	0.8	82.1	1.2	0.0	8.0	81.8	8.2	28.3

Table 6: Per-class intersection-over-union (IOU) scores on NYU.

		Building	Tree	Sky	Car	Sign	Road	Pedestrian	Fence	Column	Sidewalk	Bicyclist	Ave
01TP	Original	46.0	61.7	86.2	55.8	0.0	78.7	21.9	4.7	9.5	61.6	20.4	40.6
	Ours	53.9	66.0	88.5	64.8	0.0	80.5	30.1	3.4	11.3	62.4	28.8	44.5
05VD	Original	76.2	56.3	88.9	69.1	23.8	86.2	31.0	14.9	11.6	55.2	12.5	47.8
	[25]	79.7	60.1	89.7	73.6	27.5	88.6	37.8	16.5	8.7	62.2	5.0	50.0
	Ours	77.9	58.8	88.6	70.6	26.0	86.8	32.7	15.0	7.8	56.5	9.7	48.2

Table 7: Per-class intersection-over-union (IOU) scores on CamVid-01TP and CamVid-05VD.

	Background	Road	Lane	Vehicle	Sky	Ave
Original	87.4	91.0	45.7	54.4	94.6	74.6
[25]	89.7	91.0	39.3	55.8	95.2	74.2
Ours	90.7	91.2	39.7	69.5	94.3	77.1

Table 8: Per-class intersection-over-union (IOU) scores on MPI.

tering speed to 50 frames per second with quad-core parallel processing using OpenMP [10]. So overall the real-time performance can be achieved. In contrast, the code runs at 5 frames per second if the filtering is performed on the pixel level.

Comparisons with filters from Table 1 and Yan et al. [40]. From Table 4, we observe that our method outperforms all these baselines by a considerable margin, except for CamVid-05VD. The initial scene parsing maps of CamVid-05VD contain heavy amounts of noises varying differently for different semantic labels. Such noises negatively affect our method’s pixel tracing thus the performance gain is smaller. In Figure 7 we also show some image results of NYU. The significant benefits of our spatio-temporal filtering can be clearly observed, as we have successfully corrected a lot of “flickering” classifications in its initial maps.

Comparisons with optical flow guided spatio-temporal exponential filtering. To perform optical flow wrapping, spatio-temporal exponential filter in Table 1 is modified to $M(x, y, t) = \alpha \times M(x + u_x, y + u_y, t - 1) + (1 - \alpha)U'(x, y, t)$, where (u_x, u_y) is the flow vector computed using [3]. From Table 5, we see that our method performs comparable with or better than the optical flow guided spatio-temporal exponential filtering. This implies that pixel tracing from our method is sometimes more effective than optical flow for prediction maps filtering. For example, as CamVid-01TP is captured at dusk, its image quality is low and the optical flow computation becomes less reliable.

Comparisons with Miksik et al. [25]. Because [25] uses sophisticated appearance modeling techniques such as metric learning and optical flow to do pixel tracing, it is more robust to the noises of initial maps. Therefore from Table 5 our method performs worse than [25] on CamVid-05VD. However our method performs compar-

bly for the other videos, and better for some fast-moving categories such as “vehicle” as shown in Table 8. Moreover ours runs 20 times faster than [25].

Comparisons with offline Markov Random Field. We construct an MRF for the entire video where nodes represent superpixels and edges connect pairs of superpixels that are spatially adjacent in the same frame or between neighboring frames. The unary and edge energy terms are defined similarly to [29], and the constructed MRF is then inferred using GCMex package [5, 20, 4]. For the long video sequences, i.e., CamVid-05VD and CamVid-01TP, the MRF is constructed only on the annotated key frames for computational efficiency. Table 5 shows that our method performs better than the MRF in all the videos except for MPI. This again demonstrates that our performance is quite promising in spite of the method’s simplicity.

5. Conclusions

In this work, we propose an efficient online video filtering method, named adaptive exponential smoothing (AES), to refine pixel prediction maps. Compared with the traditional average and exponential filtering, our AES does not fix the spatial location or temporal smoothing bandwidth while performing temporal smoothing. Instead, it performs adaptive filtering for different pixels, thus can better address missing and false pixel predictions, and better tolerate fast object movements and camera motion. The experimental evaluations on saliency map filtering and multi-class scene parsing validate the superiority of the proposed method compared with the state of the art. Thanks to the proposed dynamic programming algorithm for pixel tracing, our filtering method has linear time complexity and runs in real time.

6. Acknowledgements

The authors are thankful for Mr Xincheng Yan and Mr Hui Liang for providing the code of [40], and Dr Daniel Munoz and Mr Ondrej Miksik for providing the initial scene parsing maps of [25]. This work is supported in part by Singapore Ministry of Education Tier-1 Grant M4011272.

References

- [1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels. *EPFL*, 2010.
- [2] V. Badrinarayanan, I. Budvytis, and R. Cipolla. Semi-supervised video segmentation using tree structured graphical models. *TPAMI*, 2013.
- [3] L. Bao, Q. Yang, and H. Jin. Fast edge-preserving patch-match for large displacement optical flow. *TIP*, 2014.
- [4] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *TPAMI*, 2004.
- [5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *TPAMI*, 2001.
- [6] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*. 2008.
- [7] R. G. Brown. *Smoothing, forecasting and prediction of discrete time series*. Courier Corporation, 2004.
- [8] A. Y. Chen and J. J. Corso. Temporally consistent multi-class video-object segmentation with the video graph-shifts algorithm. In *WACV*, 2011.
- [9] C. Couprie, C. Farabet, L. Najman, and Y. Lecun. Convolutional nets and watershed cuts for real-time semantic labeling of rgb-d videos. *JMLR*, 2014.
- [10] L. Dagum and R. Enon. Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering*, 1998.
- [11] K. Dang and J. Yuan. Location constrained pixel classifiers for image parsing with regular spatial layout. In *BMVC*, 2014.
- [12] A. Ess, T. Mueller, H. Grabner, and L. J. Van Gool. Segmentation-based urban traffic scene understanding. In *BMVC*, 2009.
- [13] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Scene parsing with multiscale feature learning, purity trees, and optimal covers. *ICML*, 2012.
- [14] G. Floros and B. Leibe. Joint 2d-3d temporally consistent semantic segmentation of street scenes. In *CVPR*, 2012.
- [15] F. Galasso, M. Keuper, T. Brox, and B. Schiele. Spectral graph reduction for efficient image and streaming video segmentation. In *CVPR*, 2014.
- [16] A. Hernández-Vela, N. Zlateva, A. Marinov, M. Reyes, P. Radeva, D. Dimov, and S. Escalera. Graph cuts optimization for multi-limb human segmentation in depth maps. In *CVPR*, 2012.
- [17] S. D. Jain and K. Grauman. Supervoxel-consistent foreground propagation in video. In *ECCV*. 2014.
- [18] J. F. Kenney and E. S. Keeping. Mathematics of statistics-part one. 1954.
- [19] J. Kim and J. W. Woods. Spatio-temporal adaptive 3-d kalman filter for video. *TIP*, 1997.
- [20] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *TPAMI*, 2004.
- [21] J. Lee, S. Kwak, B. Han, and S. Choi. Online video segmentation by bayesian split-merge clustering. In *ECCV*. 2012.
- [22] M. Mahmoudi and G. Sapiro. Fast image and video denoising via nonlocal means of similar neighborhoods. *SPL*, 2005.
- [23] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.
- [24] B. Micusik, J. Košecká, and G. Singh. Semantic parsing of street scenes from video. *IJRR*, 2012.
- [25] O. Miksik, D. Munoz, J. A. Bagnell, and M. Hebert. Efficient temporal consistency for streaming video scene analysis. In *ICRA*, 2013.
- [26] D. Munoz, J. A. Bagnell, and M. Hebert. Stacked hierarchical labeling. In *ECCV*. 2010.
- [27] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, 2013.
- [28] S. Paris. Edge-preserving smoothing and mean-shift segmentation of video streams. In *ECCV*. 2008.
- [29] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 2009.
- [30] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *CRCV-TR*, 2012.
- [31] J. Tighe and S. Lazebnik. Finding things: Image parsing with regions and per-exemplar detectors. In *CVPR*, 2013.
- [32] D. Tran, J. Yuan, and D. Forsyth. Video event detection: From subvolume localization to spatiotemporal path search. *TPAMI*, 2014.
- [33] D. Tsai, M. Flagg, A. Nakazawa, and J. M. Rehg. Motion coherent tracking using multi-label mrf optimization. *IJCV*, 2012.
- [34] C. Wojek, S. Roth, K. Schindler, and B. Schiele. Monocular 3d scene modeling and inference: Understanding multi-object traffic scenes. In *ECCV*. 2010.
- [35] C. Wojek and B. Schiele. A dynamic conditional random field model for joint labeling of object and scene classes. In *ECCV*. 2008.
- [36] C. Xu and J. J. Corso. Evaluation of super-voxel methods for early video processing. In *CVPR*, 2012.
- [37] C. Xu, S. Whitt, and J. J. Corso. Flattening supervoxel hierarchies by the uniform entropy slice. In *ICCV*, 2013.
- [38] C. Xu, C. Xiong, and J. J. Corso. Streaming hierarchical video segmentation. In *ECCV*. 2012.
- [39] Y. Xu, D. Song, and A. Hoogs. An efficient online hierarchical supervoxel segmentation algorithm for time-critical applications. In *BMVC*, 2014.
- [40] X. Yan, J. Yuan, H. Liang, and L. Zhang. Efficient online spatio-temporal filtering for video event detection. *ECCVW*, 2014.
- [41] D. Zhang, O. Javed, and M. Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *CVPR*, 2013.
- [42] B. Zhou, X. Hou, and L. Zhang. A phase discrepancy analysis of object motion. In *ACCV*. 2011.