Efficient Temporal Consistency for Streaming Video Scene Analysis

Ondrej Miksik Daniel Munoz J. Andrew Bagnell Martial Hebert

***Abstract*— We address the problem of image-based scene analysis from streaming video, as would be seen from a moving platform, in order to efficiently generate spatially and temporally consistent predictions of semantic categories over time. In contrast to previous techniques which typically address this problem in batch and/or through graphical models, we demonstrate that by learning visual similarities between pixels across frames, a simple filtering algorithm is able to achieve high performance predictions in an efficient and online/causal manner. Our technique is a meta-algorithm that can be efficiently wrapped around any scene analysis technique that produces a per-pixel semantic category distribution. We validate our approach over three different scene analysis techniques on three different datasets that contain different semantic object categories. Our experiments demonstrate that our approach is very efficient in practice and substantially improves the consistency of the predictions over time.**

I. Introduction

A semantic understanding of the environment from images, as illustrated in Fig. 1, plays an important role for a variety of robotic tasks, such as autonomous navigation. This problem has been investigated extensively in prior work using different techniques such as graphical models [1], [2], deep learning [3], [4], exemplar-based [5], [6], and iterative decoding techniques [7], [8]. Typically, these techniques address the problem of scene analysis from a single image, and extending them to handle temporal sequences of images, as would be seen from a mobile platform, is very challenging. Simply applying the scene analysis algorithm to each image independently is not sufficient because it does not properly enforce consistent predictions over time. In practice, the temporally inconsistent predictions result in "flickering" classifications. This effect is not solely due to the motion of the camera through the 3D scene: we often observe this behavior even on a sequence of images from a *static* scene due to subtle illumination changes. These inconsistencies in predictions can have a major impact on robotic tasks in practice, *e.g.*, predicted obstacles may suddenly appear in front of the robot in one frame and then vanish in the next. The situation is further complicated by the need for online/causal algorithms in robotics applications, in which the system does not have access to future frames, unlike video interpretation systems which can proceed in batch mode by using all the available frames [9], [10].

O. M. is with the Center for Machine Perception, Czech Technical University in Prague. ondra.miksik@gmail.com

D. M., J. A. B. and M. H. are with The Robotics Institute, Carnegie Mellon University. {dmunoz, dbagnell, hebert}@ri.cmu.edu

This work was conducted through collaborative participation in the Robotics Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016 and by the Czech Science Foundation grant GACR P103/12/G084.

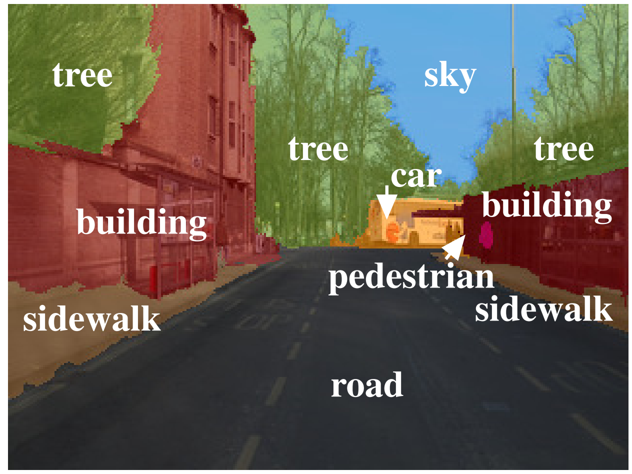
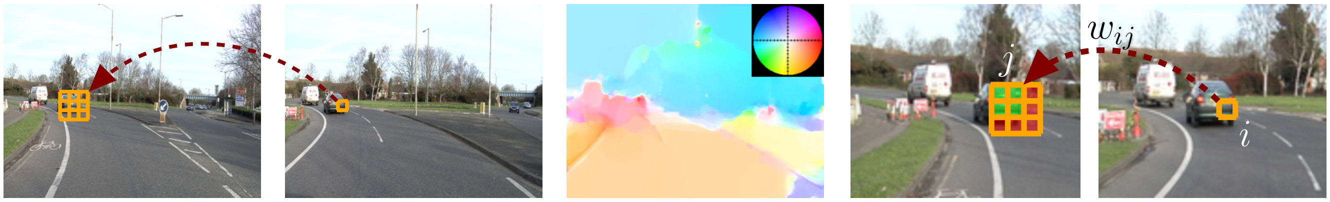


Fig. 1: Predicted semantic categories from our approach.

Inspired by early work in robotics using linear filters [11], we consider a simple, causal filtering technique for maintaining temporally consistent predictions. Our approach is a meta-algorithm in the sense that it is agnostic to the specific way in which predictions are generated, so that it can be used with any per-frame scene analysis technique. Our only requirement is that the per-frame scene analysis technique predicts a per-pixel probability distribution over semantic labels, instead of a single label.

Our algorithm is illustrated in Fig. 2. At the current frame $I^{(t)}$, each pixel i is associated with a label probability distribution $\mathbf{y}_i^{(t)}$, which is produced by a scene analysis algorithm. Our goal is to ensure that the final label distribution that we return for pixel i is consistent with the temporal prediction $\hat{\mathbf{y}}_j^{(t-1)}$ from its corresponding pixel j in the previous frame $I^{(t-1)}$, which we do not know. Hence, we use optical flow [12] to estimate a neighborhood of candidate correspondences in the previous frame. Giving all neighbors equal weight and defining the smoothed prediction based on the average of the neighborhood's predictions is unwise because the neighborhood could include pixels of completely different objects. Therefore, between pixels $i \in I^{(t)}$ and $j \in I^{(t-1)}$, we propose to *learn* a data-driven, visual similarity function to assign a high weight w_{ij} between pixels that are likely to correspond to each other (and low weight for those that are not) in order to select correct correspondences and accurately propagate predictions over time.

Before discussing how predictions are combined between



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

Fig. 2: Overview of our approach. (a) Frame $I^{(t-1)}$. (b) Frame $I^{(t)}$. The distribution of labels at a pixel in frame $I^{(t)}$ is combined with a weighted average of the distributions of labels in a neighborhood of pixels (the 3×3 orange grid) in the previous frame $I^{(t-1)}$. This neighborhood is initialized using optical flow techniques (c). We propagate predictions across time by *learning* a similarity function between pixels $i \in I^{(t)}$ (e) and $j \in I^{(t-1)}$ (d). This similarity assigns high values w_{ij} between visually similar pixels (green cells) and low values over visually different pixels (red cells).

two frames, in Sec. III, we first demonstrate the importance of using a data-driven function for measuring visual similarity between candidate pixels and present an efficient algorithm to learn this similarity function. In Sec. IV, we discuss how candidate pixels between frames are generated and how to combine the previous frame's predictions using the learned similarity function. In Sec. V, we validate our proposed method over three distinct semantic labeling algorithms on three different datasets. Our experiments confirm that this natural approach yields substantial improvements in the consistency of the predictions over time, with the additional important benefits of being very efficient and simple to implement.

II. Related Work

One popular way to incorporate temporal information is to compute Structure from Motion (SfM) between consecutive frames in order to compute geometric/motion-based features [13]–[15]. One drawback of this approach is that accurate SfM computation may be slow or may require a large buffer of previous frames to process. Alternatively, or in addition, a large graphical model can be defined among multiple frames, where edges between frames propagate predictions over time [16]–[21]. Performing bounded, approximate inference over such large models remains a challenging problem. Furthermore, in order to efficiently compute approximate solutions, only an estimate of the MAP distribution is returned, *i.e.*, there is no uncertainty in the labeling or marginal distributions. To further improve efficiency in practice, techniques make further approximations at the cost of loss of guarantees on the solution quality. By returning label probabilities, our approach may be more useful as input for subsequent robotic algorithms, such as reasoning about multiple interpretation hypotheses. Another technique for maintaining temporal consistency, which is similar to defining a spatio-temporal graphical model, is to analyze volumes from a spatio-temporal segmentation [9]. This batch approach is omniscient in the sense that it requires processing the entire video sequence, which is typically not suitable for most robotic applications.

III. Learning Similarity

A. Metric Learning

In order to selectively propagate predictions from the previous frame, we assign high weight between pixels that are visually similar. One standard way to measure similarity w_{ij} between two pixels is through a radial basis kernel

$$w_{ij} = \exp\left(-\frac{d(\mathbf{f}_i, \mathbf{f}_j)}{\sigma^2}\right), \tag{1}$$

where $\mathbf{f}_i \in \mathbb{R}^d$ is the vector of visual features of pixel i, $\sigma = 0.4$ controls the bandwidth of the kernel, and $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ is a distance function. Only using the pixels' RGB values for the feature representation is not discriminative enough to correctly match correspondences. We instead augment RGB values with the responses from 17 gradient filters and a 11×11 local binary pattern window, resulting in a feature descriptor $\mathbf{f}_i \in \mathbb{R}^{140}$. Because of the increase in dimensionality, the standard squared Euclidean distance

$$d(\mathbf{f}_i, \mathbf{f}_j) = (\mathbf{f}_i - \mathbf{f}_j)^T(\mathbf{f}_i - \mathbf{f}_j) = \mathrm{Tr}(\Delta_{ij}), \tag{2}$$

where $\Delta_{ij} = (\mathbf{f}_i - \mathbf{f}_j)(\mathbf{f}_i - \mathbf{f}_j)^T$, is typically large between most feature vectors, as illustrated in Fig. 3-a,b. Alternatively, we can learn a distance that has the desirable properties for our application. That is, for a pair of pixels (i, j) from the set of pairs of pixels that truly correspond to each other, $\mathcal{E}_p$, we want $d(\mathbf{f}_i, \mathbf{f}_j)$ to be small, and for a pair of pixels (i, k) from the set that do not correspond, $\mathcal{E}_n$, we want $d(\mathbf{f}_i, \mathbf{f}_k)$ to be large. Learning a distance, or metric, also remains an active area of research [22]–[27], and a variety of techniques could be used to learn d. As we are concerned with efficiency in the predictions, we use a simple Mahalanobis distance

$$d_{\mathbf{M}}(\mathbf{f}_i, \mathbf{f}_j) = (\mathbf{f}_i - \mathbf{f}_j)^T\mathbf{M}(\mathbf{f}_i - \mathbf{f}_j) = \mathrm{Tr}(\mathbf{M}^T\Delta_{ij}), \tag{3}$$

and propose an efficient method to learn the parameters $\mathbf{M}$ offline from training data.

We follow the max-margin learning strategy and learn a metric such that the distances between pixels that do *not* correspond $(i, k) \in \mathcal{E}_n$ are larger by a margin than the distances between pixels that do correspond $(i,j) \in \mathcal{E}_p$. This can be formulated as the convex, semidefinite program

$$\begin{aligned} \min_{\mathbf{M},\xi,\zeta} \quad & \|\mathbf{M}\|_F^2 + \alpha \sum_{(i,j)\in\mathcal{E}_p} \xi_{ij} + \beta \sum_{(i,k)\in\mathcal{E}_n} \zeta_{ik} \\ \text{s.t.} \quad & d_{\mathbf{M}}(\mathbf{f}_i, \mathbf{f}_j) \leq 1 + \xi_{ij}, \qquad \forall (i,j) \in \mathcal{E}_p \\ & d_{\mathbf{M}}(\mathbf{f}_i, \mathbf{f}_k) \geq 2 + \zeta_{ik}, \qquad \forall (i,k) \in \mathcal{E}_n \\ & M \in \mathcal{M}, \end{aligned} \tag{4}$$

where $\mathcal{M} = \{\mathbf{M} | \mathbf{M} \succeq 0, \mathbf{M} = \mathbf{M}^T\}$ is the convex cone of symmetric positive-semidefinite matrices, and α and β penalize violating the margins. This program can be equivalently rewritten as the unconstrained, convex minimization

$$\begin{aligned} \min_{\mathbf{M}\in\mathcal{M}} \operatorname{Tr}(\mathbf{M}^T\mathbf{M}) &+ \alpha \sum_{(i,j)\in\mathcal{E}_p} \max(0, \operatorname{Tr}(\mathbf{M}^T\Delta_{ij}) - 1) \\ &+ \beta \sum_{(i,k)\in\mathcal{E}_n} \max(0, 2 - \operatorname{Tr}(\mathbf{M}^T\Delta_{ik})), \end{aligned} \tag{5}$$

and can be efficiently optimized using the projected subgradient method [28]. We define $\boldsymbol{\Upsilon}_{ij}$ and $\boldsymbol{\Psi}_{ik}$ to be subgradients of the respective summands:

$$\begin{aligned} \boldsymbol{\Upsilon}_{ij} &= \begin{cases} \Delta_{ij}, & \operatorname{Tr}(\mathbf{M}^T\Delta_{ij}) - 1 > 0 \\ 0, & \text{otherwise} \end{cases}, \\ \boldsymbol{\Psi}_{ik} &= \begin{cases} -\Delta_{ik}, & 2 - \operatorname{Tr}(\mathbf{M}^T\Delta_{ik}) > 0 \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{6}$$

The subgradient update, with small step-size η_t, is then

$$\mathbf{M}_{t+1} \leftarrow \mathcal{P}_{\mathcal{M}}(\mathbf{M}_t - \eta_t(\mathbf{M}_t + \alpha \sum_{(i,j)\in\mathcal{E}_p} \boldsymbol{\Upsilon}_{ij} + \beta \sum_{(i,k)\in\mathcal{E}_n} \boldsymbol{\Psi}_{ik})), \tag{7}$$

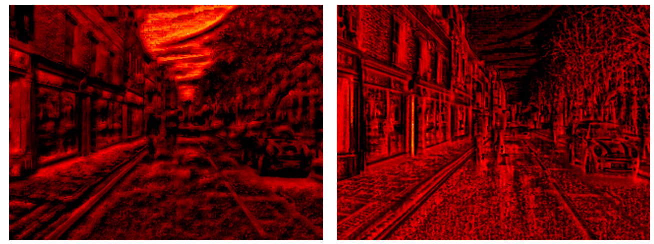
where $\mathcal{P}_{\mathcal{M}}$ projects to the closest matrix on the convex cone $\mathcal{M}$. Since $\boldsymbol{\Upsilon}_{ij}$ and $\boldsymbol{\Psi}_{ik}$ are symmetric by construction, the closest projection, with respect to the Frobenius norm, is computed by reconstructing the matrix with its negative eigenvalues set to zero [29]. To further improve run-time efficiency, we also constrain $\mathbf{M}$ to be a diagonal matrix.

As illustrated in Fig. 3-c, our learned metric measures visual similarity much better than Euclidean distance. Although the computed distances may not be optimal across the entire image, we observe correct behavior over a local area. Thus, we use optical flow to initialize the area in which to compute distances over.

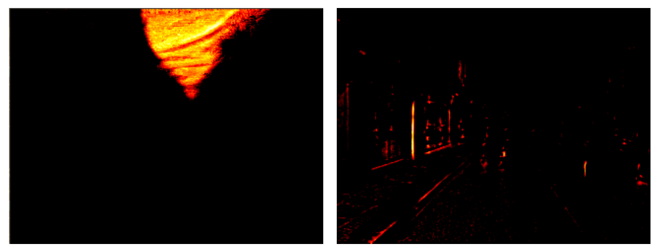
B. Obtaining Training Data

Learning the metric requires annotated examples of pairs of pixels that do and do not correspond. One way to generate these examples is to use the annotated images and sample pairs of pixels that belong to the same semantic category to create $\mathcal{E}_p$ and use pairs between the different categories to create $\mathcal{E}_n$. The result will be a general metric between categories rather than correspondences between pixels and is a much harder problem. Furthermore, our similarity metric should work well between correspondences under different viewpoints, as this is the mode of operation while the robot is moving.

(a)



(b)



(c)

Fig. 3: Comparing similarity metrics. (a) A scene with two pixels of interest selected. (b) The inverted heatmaps of Euclidean distances of the respective pixel of interest to every other pixel in the image (bright means small distance). (c) The inverted heatmaps from our learned Mahalanobis metric.

We instead generate our pairs of *training* data using pixel correspondences across multiple frames and viewpoints. These correspondences can be easily obtained through a variety of different keypoint-based algorithms. Specifically, we use the publicly available SfM package, Bundler [30]. Given a collection of images, Bundler produces a 3-D reconstruction of the scene (which we ignore) as well as the corresponding pixels across multiple frames. As illustrated in Fig. 4, we use pairs of pixels from the same correspondence to construct $\mathcal{E}_p$ and use pairs that do not correspond to construct $\mathcal{E}_n$ when learning the metric *offline*. In addition to hard margin constraints, it may be beneficial to also incorporate (convex) *relative* margin constraints over triplets of pixels i, j, k

$$d_{\mathbf{M}}(\mathbf{f}_i, \mathbf{f}_j) \leq d_M(\mathbf{f}_i, \mathbf{f}_k) + \delta_{ijk}, \tag{8}$$

where $\delta_{ijk} \in \mathbb{R}^+$ encodes the relative benefit (*e.g.*, distance) of true correspondences (i,j) over incorrect correspondences (i,k); we leave this for future work. Now that we can accurately measure visual similarity, in the following section we describe how to combine predictions over time.

Fig. 4: Generating pairs of training data. Pairs of pixels of true correspondences (along the same colored line) form $\mathcal{E}_p$ and pairs of pixels that do not correspond (between different colors lines) form $\mathcal{E}_n$.

IV. Temporal Consistency

A. Optical Flow for Data Propagation

We are interested in general scene analysis without making strong assumptions on the movement/frame-rate of the camera and/or the type of object motions in the scene. Furthermore, we require the procedure to be as efficient as possible so it can potentially be used onboard a mobile robot. To generate initial hypothesis between frames, we use the efficient Anisotropic Huber-L$_1$ dense optical flow algorithm from Werlberger *et al.* [12] to obtain flow fields[1] $\overleftarrow{\mathbf{V}}_x, \overleftarrow{\mathbf{V}}_y$ that project pixel coordinates from $(u^{(t)}, v^{(t)}) \in I^{(t)}$ to $(u^{(t-1)}, v^{(t-1)}) \in I^{(t-1)}$ via

$$\begin{bmatrix} u^{(t-1)} \\ v^{(t-1)} \end{bmatrix} = \begin{bmatrix} u^{(t)} \\ v^{(t)} \end{bmatrix} + \begin{bmatrix} \overleftarrow{\mathbf{V}}_x(u^{(t)}, v^{(t)}) \\ \overleftarrow{\mathbf{V}}_y(u^{(t)}, v^{(t)}) \end{bmatrix}. \tag{9}$$

Although we use a state-of-the-art optical flow algorithm, it is not perfect and some velocity vectors may not match exactly and/or some correspondences between frames might be missing. To help recover from missing flow information, we warp/transfer *small patches* of data at a time, instead of a single pixel. That is, for each pixel $(u^{(t)}, v^{(t)}) \in I^{(t)}$ and its correspondence $(u^{(t-1)}, v^{(t-1)}) \in I^{(t-1)}$, we use the forward-in-time flow vector $[-\overleftarrow{\mathbf{V}}_x(u^{(t)}, v^{(t)}), -\overleftarrow{\mathbf{V}}_y(u^{(t)}, v^{(t)})]^T$ to transfer *both* the RGB values and temporally smoothed predictions, $\hat{\mathbf{y}}^{(t-1)}$, of each pixel in a 5×5 patch centered at $(u^{(t-1)}, v^{(t-1)})$. For each warped pixel, we accumulate RGB values and previous temporal predictions into the respective coordinates. After all pixels from the previous frame have been warped, the RGB values and predictions are appropriately averaged by the number of projections that fell into each coordinate, resulting in a warped RGB image $\tilde{I}^{(t)}$ and warped predictions $\tilde{\mathbf{y}}^{(t)}$ into the current time t.

Estimated optical flows are imperfect, and in these scenarios we do not want to propagate label predictions over time. A standard method to detect optical flow failures is to ensure that the flows forward and backward in time are consistent. We consider the flow at pixel $(u^{(t)}, v^{(t)})$ to be invalid if

$$\left\| \begin{bmatrix} \overrightarrow{\mathbf{V}}_x(u^{(t-1)}, v^{(t-1)}) \\ \overrightarrow{\mathbf{V}}_y(u^{(t-1)}, v^{(t-1)}) \end{bmatrix} + \begin{bmatrix} \overleftarrow{\mathbf{V}}_x(u^{(t)}, v^{(t)}) \\ \overleftarrow{\mathbf{V}}_y(u^{(t)}, v^{(t)}) \end{bmatrix} \right\|_2 \geq \kappa, \tag{10}$$

[1]Here, the subscripts x, y are overloaded to indicate direction and are not related to features nor labels.

Algorithm 1 Causal Temporal Smoothing

Inputs: Metric $\mathbf{M}$, previous frame $I^{(t-1)}$ with its temporally smoothed predictions $\hat{\mathbf{y}}^{(t-1)}$, and current frame $I^{(t)}$ with its independent predictions $\mathbf{y}^{(t)}$.
Compute $\overrightarrow{\mathbf{V}}$ and $\overleftarrow{\mathbf{V}}$ between $I^{(t-1)}$ and $I^{(t)}$ using [12]
Use $\overleftarrow{\mathbf{V}}$ to warp $I^{(t-1)} \to \tilde{I}^{(t)}$ and $\hat{\mathbf{y}}^{(t-1)} \to \tilde{\mathbf{y}}^{(t)}$, except for invalid locations as defined by Eq. 10
Compute per-pixel features for $\tilde{I}^{(t)}$ and $I^{(t)}$
for $i \in I^{(t)}$ **do**
    Define $\hat{\mathbf{y}}_i^{(t)}$ using Eqs. 11, 1, 3
end for
return Temporally smoothed predictions $\hat{\mathbf{y}}^{(t)}$

where $\overrightarrow{\mathbf{V}}$ is the flow from $I^{(t-1)}$ to $I^{(t)}$ and $\kappa = 13$ is a small threshold which is related to the size of the neighborhood used in the update described in the following subsection.

B. Causal Temporal Smoothing

Now we have all the necessary components for recursive temporal smoothing: a metric for measuring visual similarity between two pixels, w_{ij}, and a method to warp pixels between frames. At the recursive step of our procedure, we have the warped RGB image, $\tilde{I}^{(t)}$, its warped temporally smoothed predictions, $\tilde{\mathbf{y}}^{(t)}$, and the predicted per-pixel probabilities from the scene analysis algorithm $\hat{\mathbf{y}}^{(t)}$ for current frame $I^{(t)}$. For each pixel $i \in I^{(t)}$ and $j \in \tilde{I}^{(t)}$, we compute the pixel features (RGB, texture gradients, local binary pattern) $\mathbf{f}_i, \mathbf{f}_j$ that were used to learn our metric. Using an exponential smoothing update, we define a pixel's new temporally smoothed predictions using the update rule

$$\hat{\mathbf{y}}_i^{(t)} = \frac{1}{Z_i}\left(\sum_{j \in N_i^{(t)}} w_{ij}\tilde{\mathbf{y}}_j^{(t)} + c\mathbf{y}_i^{(t)}\right), \tag{11}$$

where $N_i^{(t)}$ is a 5×5 spatial neighborhood in $I^{(t)}$ centered at pixel i, $c = 0.25$ is our prior belief on the independent prediction from the scene analysis algorithm, and $Z_i = \sum_{j \in N_i^{(t)}} w_{ij} + c$ ensures that $\hat{\mathbf{y}}_i^{(t)}$ sums to one. The procedure then repeats to smooth predictions $\mathbf{y}^{(t+1)}$ using $\hat{\mathbf{y}}^{(t)}$. The entire procedure is summarized in Algorithm 1.

V. Experimental Analysis

We evaluate our approach over three sophisticated scene analysis algorithms over three different datasets. All results were obtained using the same smoothing parameters across the different algorithms/datasets.

A. Algorithms and Datasets

1) `CamVid`*:* This dataset [13] consists of over 10 minutes of 30 Hz video captured in an urban environment during daylight and dusk containing 11 semantic categories. The sequences are sparsely annotated in time at 1 Hz. For this dataset, we use the hierarchical inference machine algorithm from [8]. Without using any temporal information, we obtain state-of-the-art overall pixel and average per-class accuracies

TABLE I: Average timings between two frames

Computation	Time (s)
Optical flows (GPU)	0.02
Smoothing (CPU)	0.75
Total	0.77

of 84.2% and 59.5%, respectively, and is comparable [14] or exceeds [13] other techniques which use temporal features. For evaluating temporal consistency, we trained two separate models. The first is trained on the standard training fold from [13], and evaluated the test sequence `05VD`. The second model is evaluated on the `16E5` sequence and trained on the remaining images not from this sequence.

2) `NYUScenes`: This dataset consists of 74 annotated frames and is captured by a person walking in an urban street with a hand-held camera. In addition to the moving objects, the video has a lot of camera motion due to the motion of the person. For this dataset, we used the outputs from the deep learning architecture [4], provided by the authors. This model was trained on the `SIFTFlow` dataset [6], which contains 33 semantic categories.

3) `MPI-VehicleScenes`: This dataset[2] [31] consists of 156 annotated frames captured from a dashboard-mounted camera while driving in a city and consists of only 5 semantic classes. For this dataset, we used the outputs from the per-pixel, boosted classifier [16], provided by the authors, which is based on the JointBoost algorithm [32].

B. Efficiency

There are two main components of our approach: 1) forward and backward, dense optical flow computation and 2) temporal smoothing (which includes warping, pixel features, and the weighted averaging). The average computation times between two frames are shown in Table I. The experiments were performed using a GeForce GTX 590 GPU and an Intel X5670 CPU. We observe that dense optical flow computation time can be brought down from the order of seconds with using a CPU to 10s of milliseconds with using a GPU. As our approach relies on many, simple numeric computations, we would expect a similar efficiency improvement with a GPU implementation.

In practice, the computational bottleneck is often from the scene analysis algorithm, depending on how expressive the features and/or model are. For example, both of the structured prediction algorithms [4], [8] we use take between 1 and 2 seconds to process a frame, which also includes feature computation. However, we demonstrate with the third experiment on `MPI-VehicleScenes` that our approach can yield spatially and temporally consistent classifications with a simple per-pixel classifier.

C. Analysis

Videos comparing per-frame and the temporally smoothed classifications for the sequences are available in the supplementary multimedia attachment; qualitative examples from

[2]We use video sequence `Continuous_2008.07.30_at_13.10.53`

TABLE II: Per-class F_1 scores and accuracy on `CamVid`

	05VD		16E5	
Class	**Per-frame**	**Temporal**	**Per-frame**	**Temporal**
sky	.303	**.682**	.237	**.639**
tree	.352	**.563**	.336	**.518**
road	.197	**.546**	.150	**.529**
sidewalk	.277	**.512**	.188	**.357**
building	.165	**.232**	.275	**.395**
car	.127	**.456**	.304	**.597**
pole	.201	**.386**	.252	**.285**
person	.138	**.218**	**.324**	.193
bicycle	**.323**	.165	**.348**	.043
fence	.325	**.712**	.335	**.668**
sign	**.367**	.029	**.378**	.039
Accuracy	.261	**.591**	.286	**.530**

TABLE III: Per-class F_1 scores and accuracy on `NYU`

Class	Per-frame	Temporal
building	.231	**.547**
car	.207	**.630**
door	**.046**	.000
person	.094	.080
pole	.169	.139
road	.152	**.575**
sidewalk	**.373**	.274
sign	**.127**	.000
sky	.002	.019
tree	.353	**.630**
window	.102	.101
Accuracy	.209	**.500**

TABLE IV: Per-class F_1 scores and accuracy on `MPI`

Class	Per-frame	Temporal
background	.420	**.730**
road	**.503**	.321
lane-marking	**.779**	.319
vehicle	.075	**.276**
sky	.206	**.571**
Accuracy	.407	**.533**

each sequence are shown in Figs. 5, 6, and 7. The videos demonstrate the substantial benefit of using temporal smoothing, especially on the `CamVid` sequences which are captured at a much higher frame rate.

It is important to remember that our approach relies on the output of the inner scene analysis algorithm and cannot fix misclassifications due to the biases of the base algorithm. Hence, for quantitative evaluation we first only consider pixels from which the prediction obtained by the scene analysis algorithm differs with our temporal smoothing. We compute a confusion matrix over these differing pixels and report the per-class F_1 scores in Tables II, III, IV, as well as the per-pixel accuracy on the differing pixels; improvements greater than 0.03 are bolded. This evaluation measures whether we are worse or better off with using temporal smoothing.

The behavior across datasets is consistent: categories which occupy large areas of the image (*e.g.*, sky, trees, cars, buildings) are significantly improved upon and predictions on some of the smaller objects (*e.g.*, signs, people, lane-markings) are sometimes oversmoothed. There are various reasons as to why performance may decrease. 1) Optical flow estimation on small objects may fail due to large displace-

TABLE V: Overall pixel accuracies (%)

Dataset	Independent	Smoothed
CamVid-05VD	84.60	**86.85**
CamVid-16E5	87.37	**88.84**
NYUScenes	71.11	**75.31**
MPI-VehicleScenes	93.10	**93.55**

ments and/or excessive blurring, resulting in neighborhoods that are not adequately initialized. 2) As these objects occupy a small number of pixels, over-smoothing from spatially adjacent objects will cause a large drop in performance. Similarly, it is challenging to accurately annotated these intricate objects, and imperfections in the ground truth can further skew evaluation. 3) These small object categories are typically challenging to classify. When the predicted label distributions from the scene analysis algorithm are less confident, *i.e.*, have high entropy, the resulting weighted combination may be incorrect. Nonetheless, the overall improvement in accuracy shows a clear benefit of using temporal smoothing rather than per-frame classification.

The comparisons of overall per-pixel accuracy for each sequence are shown in Table V. Due to the sparseness of the CamVid annotations, the quantitative improvements are not as drastic as we would expect, however, there is a noticeable gain. We also observe a large quantitative improvement in the NYUScenes sequence, even in the presence of large camera motion. The improvement in the MPI-VehicleScenes dataset is modest, however, this can be attributed to a small label set of 5 categories (vs. 11 and 33 from the other two) which often have little confusion. Furthermore, we note the predictions are qualitatively much smoother in appearance, even from using a per-pixel classifier.

VI. Conclusion

We propose an efficient meta-algorithm for the problem of spatio-temporal consistent 2-D scene analysis from streaming video. Our approach is based on recursive weighted filtering in a small neighborhood, where large displacements are captured by dense optical flow, and we propose an efficient algorithm to learn image-based similarities between pixels. As we do not require information about future frames, our causal algorithm can handle streaming images in a very efficient manner. Furthermore, we demonstrate that our approach can be wrapped around various structured prediction algorithms to improve predictions without a difficult redefinition of an inference process.

Acknowledgments

We thank A. Wendel for helping with the optical flow computation, C. Fabaret for providing the NYUScenes dataset and his classifications, C. Wojek for providing his classifications on the MPI-VechicleScenes dataset, and J. Tighe for discussions about the CamVid dataset.

References

[1] L. Ladicky, P. Sturgess, K. Alahari, C. Russell, and P. H. S. Torr, “What, where & how many? combining object detectors and crfs,” in *ECCV*, 2010.

[2] D. Munoz, J. A. Bagnell, N. Vandapel, and M. Hebert, “Contextual classification with functional max-margin markov networks,” in *CVPR*, 2009.

[3] R. Socher, C. C.-Y. Lin, A. Y. Ng, and C. D. Manning, “Parsing natural scenes and natural language with recursive neural networks,” in *ICML*, 2011.

[4] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, “Scene parsing with multiscale feature learning, purity trees and optimal covers,” in *ICML*, 2012.

[5] J. Tighe and S. Lazebnik, “Superparsing: Scalable nonparametric image parsing with superpixels,” in *ECCV*, 2010.

[6] C. Liu, J. Yuen, and A. Torralba, “SIFT flow: Nonparametric scene parsing via label transfer,” *IEEE T-PAMI*, vol. 33, no. 12, 2011.

[7] Z. Tu and X. Bai, “Auto-context and its application to high-level vision tasks and 3d brain image segmentation,” *IEEE T-PAMI*, vol. 32, no. 10, 2010.

[8] D. Munoz, J. A. Bagnell, and M. Hebert, “Stacked hierarchical labeling,” in *ECCV*, 2010.

[9] M. Grundmann, V. Kwatra, M. Han, and I. Essa, “Efficient hierarchical graph-based video segmentation,” in *CVPR*, 2010.

[10] W. Brendel and S. Todorovic, “Learning spatiotemporal graphs of human activities,” in *ICCV*, 2011.

[11] A. Giachetti, M. Campani, and V. Torre, “The use of optical flow for road navigation,” in *ICRA*, 1998.

[12] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof, “Anisotropic huber-L1 optical flow,” in *BMVC*, 2009.

[13] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, “Segmentation and recognition using structure from motion point clouds,” in *ECCV*, 2008.

[14] P. Sturgess, K. Alahari, L. Ladicky, and P. H. S. Torr, “Combining appearance and structure from motion features for road scene understanding,” in *BMVC*, 2009.

[15] B. Micusik, J. Kosecka, and G. Singh, “Semantic parsing of street scenes from video,” *IJRR*, vol. 31, no. 4, 2012.

[16] C. Wojek and B. Schiele, “A dynamic conditional random field model for joint labeling of object and scene classes,” in *ECCV*, 2008.

[17] A. Ess, T. Müller, H. Grabner, and L. Van Gool, “Segmentation-based urban traffic scene understanding,” in *BMVC*, 2009.

[18] J. Xiao and L. Quan, “Multiple view semantic segmentation for street view images,” in *ICCV*, 2009.

[19] R. de Nijs, J. S. Ramos, G. Roig, X. Boix, L. V. Gool, and K. Kühnlenz., “On-line semantic perception using uncertainty,” in *IROS*, 2012.

[20] V. Badrinarayanan, F. Galasso, and R. Cipolla, “Label propagation in video sequences,” in *CVPR*, 2010.

[21] A. Y. C. Chen and J. J. Corso, “Temporally consistent multi-class video-object segmentation with the video graph-shifts algorithm,” in *WACV*, 2011.

[22] E. Xing, A. Y. Ng, M. Jordan, and S. Russell, “Distance metric learning, with application to clustering with side-information,” in *NIPS*, 2003.

[23] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon, “Information-theoretic metric learning,” in *ICML*, 2007.

[24] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, “Neighbourhood components analysis,” in *NIPS*, 2004.

[25] R. Hadsell, S. Chopra, and Y. Lecun, “Dimensionality reduction by learning an invariant mapping,” in *CVPR*, 2006.

[26] K. Q. Weinberger and L. K. Saul, “Distance metric learning for large margin nearest neighbor classification,” *JMLR*, vol. 10, 2009.

[27] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, “Large scale online learning of image similarity through ranking,” *JMLR*, vol. 11, 2010.

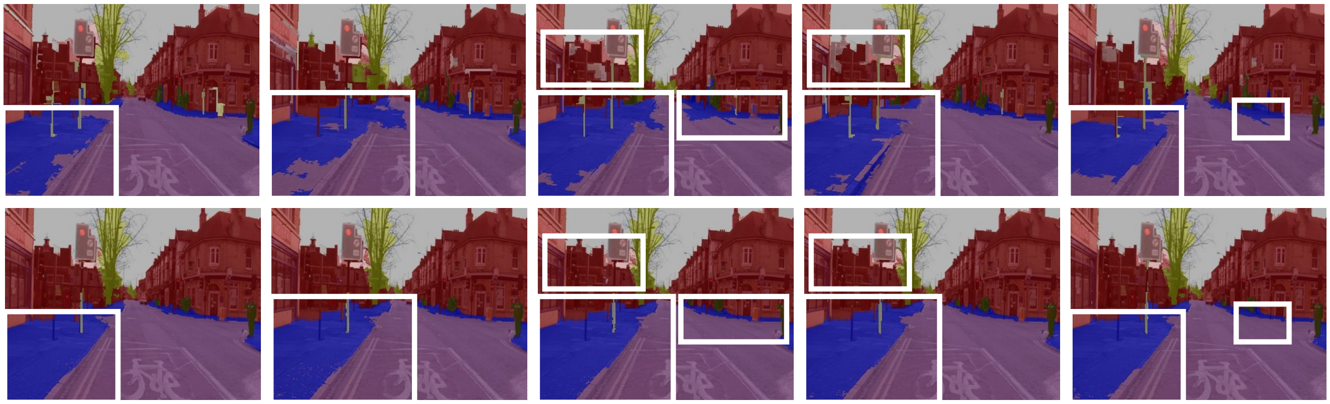
[28] N. Ratliff, J. Bagnell, and M. Zinkevich, “Online subgradient methods for structured prediction,” in *AISTATS*, 2007.

[29] G. Golub and C. F. Van Loan, *Matrix Computations*. John Hopkins University Press, 1996.

[30] N. Snavely, S. M. Seitz, and R. Szeliski, “Photo tourism: Exploring photo collections in 3d,” *ACM SIGGRAPH*, 2006.

[31] C. Wojek, S. Roth, K. Schindler, and B. Schiele, “Monocular 3d scene modeling and inference: Understanding multi-object traffic scenes,” in *ECCV*, 2010.

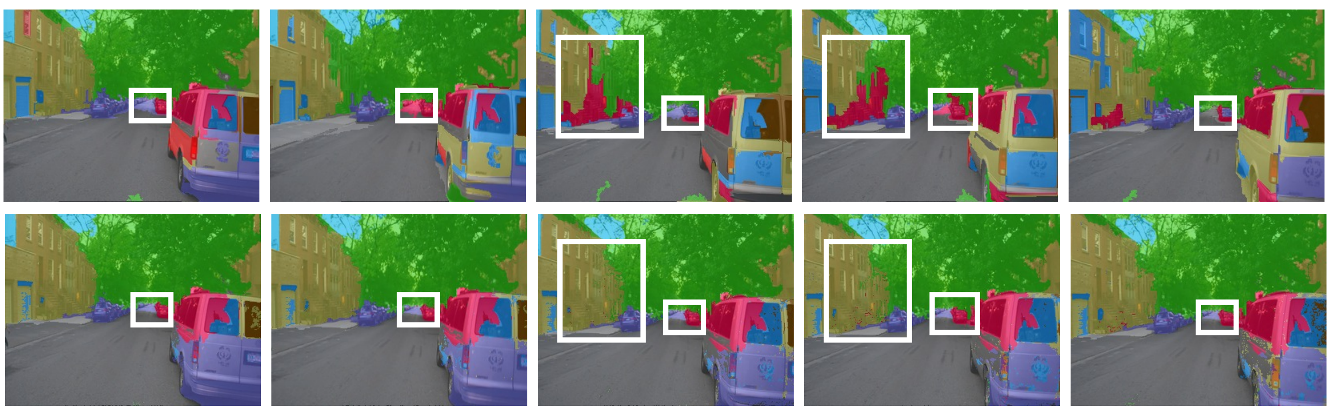
[32] A. Torralba, K. P. Murphy, and W. T. Freeman, “Sharing visual features for multiclass and multiview object detection,” *IEEE T-PAMI*, vol. 29, no. 5, 2007.



(a) frame 00055 (b) frame 00056 (c) frame 00057 (d) frame 00058 (e) frame 00059

sky tree road sidewalk building car column_pole pedestrian bicycle fence sign_symbol

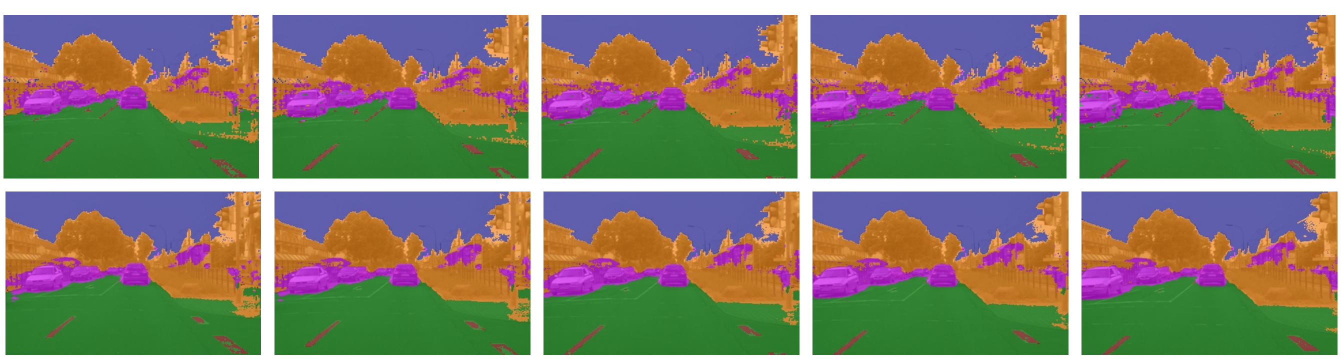
Fig. 5: `CamVid` classifications. Top: per-frame. Bottom: temporally smoothed. Inconsistent predictions are highlighted.



(a) frame 00055 (b) frame 00056 (c) frame 00057 (d) frame 00058 (e) frame 00059

unknown awning balcony building car door person road sidewalk sign
sky streetlight sun tree window

Fig. 6: `NYUScenes` classifications. Top: per-frame. Bottom: temporally smoothed. Inconsistent predictions are highlighted.



(a) frame 00265 (b) frame 00266 (c) frame 00267 (d) frame 00268 (e) frame 00269

void road lane-marking vehicle sky

Fig. 7: `MPI-VehicleScenes` classifications. Top: per-frame. Bottom: temporally smoothed.