# 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 label distribution. We validate our approach over three different scene analysis techniques on three different datasets that contain different semantic object categories. Our experiments demonstrate our approach is very efficient in practice and substantially improves the quality of 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. Many works have investigated this problem using different techniques such as graphical models [1], [2], deep learning [3], [4], exemplar-based [5], [6] and iterative decoding techniques [7], [8]. Typically, they address the problem of scene analysis from a single image. Extending these techniques to 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 labels over time. In practice, the temporally inconsistent predictions result in "flickering" classifications. This effect is not just due to the motion of the camera through the 3D scene: we often observe this behavior even on images of 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.

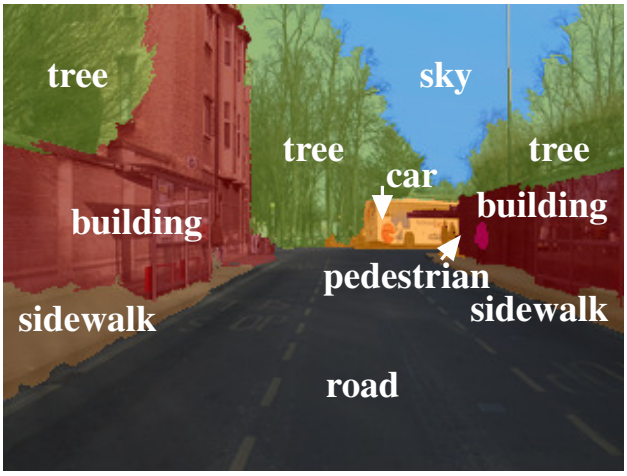



Fig. 1: Predicted semantic categories from our approach.

In this work, inspired by standard linear causal filters, we consider the most natural technique for maintaining temporally consistent predictions: a temporal filter based on exponential smoothing over past 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 is 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 label probability distribution 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. Instead, we use optical flow [11] 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 correspond to each other (and low weight for those that do not) in order to select correct correspondences and accurately propagate predictions over time. In summary, our algorithm consists of two main components:

1) Using optical flow to initialize a local neighborhood.


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

Fig. 2: Overview of our approach. The distribution of labels at a pixel in frame $I_t$ (b) is combined with a weighted average of the distributions of labels in a neighborhood of pixels in the previous frame $I_{t-1}$ (a). This neighborhood is estimated 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).

2) Selectively combining the past predictions $\hat{\mathbf{y}}^{(t-1)}$ in the neighborhood with the current prediction $\mathbf{y}_i^{(t)}$ using learned weights.

In Section III, we present an efficient algorithm to learn this similarity function, and then in Section IV, we show how to transfer predictions to obtain a temporally smoothed prediction $\hat{\mathbf{y}}_i^{(t)}$. We then validate our temporal smoothing over three distinct semantic labeling algorithms on three different datasets in Section V. Our experiments confirm that this natural approach yields drastically improved 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 [12]–[14]. One drawback of this approach is that accurate SfM computation may be slow and/or may require a large buffer of previous frames to process. Alternatively, and/or in addition, a large graphical model can be defined among multiple frames, where edges between frames propagate predictions over time [15]–[19]. 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 [10]. 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 function 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. Directly using the pixels' RGB values for the feature representation is not discriminative enough to correctly match correspondences. We instead augment the color with local texture and local binary pattern features, resulting in $\mathbf{f}_i \in \mathbb{R}^{141}$. 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-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 [20]–[23], 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

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

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

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

and can be efficiently optimized using the projected subgradient method [24]. We define $\boldsymbol{\Upsilon}_{ij}$ and $\boldsymbol{\Psi}_{ij}$ to be subgradients of the respective summands:

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

The subgradient update, with small step-size $\eta_t$, is then

$$\mathbf{M}_{t+1} \leftarrow \mathcal{P}_{\mathcal{M}} \left( \mathbf{M}_t - \eta_t (\mathbf{M}_t + \alpha \sum_{ij} \boldsymbol{\Upsilon}_{ij} + \beta \sum_{ij} \boldsymbol{\Psi}_{ik}) \right), \tag{7}$$

where $\mathcal{P}_{\mathcal{M}}$ projects to the closest matrix on the convex cone $\mathcal{M}$. Since $\boldsymbol{\Psi}_{ij}$ and $\boldsymbol{\Upsilon}_{ij}$ 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 0 [25]. To further improve run-time efficiency, we also constrain $\mathbf{M}$ to be a diagonal matrix.

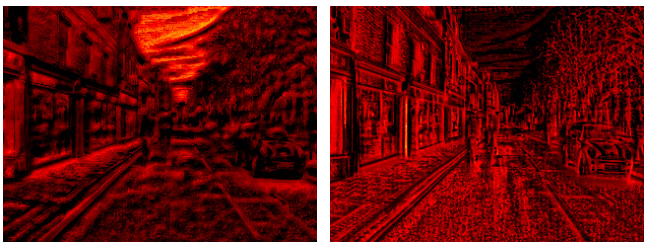
As illustrated in Fig. 3-c, our learned metric greatly improves at discriminating pixels by visual similarity than using Euclidean distance and is often small among pixels with the same semantic category. Although the computed distances may not be optimal across the entire image, we observe correct behavior over a local area. Thus, we do not have to be globally correct and can use optical flow to initialize the area in which to compute distances.

### B. Obtaining Training Data

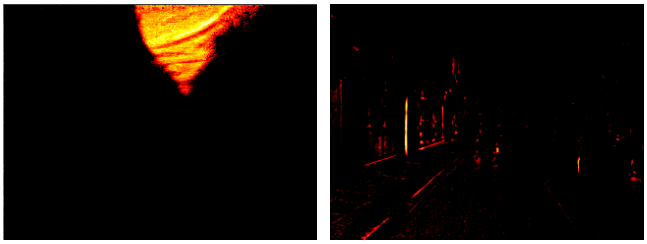
As we propagate predicted probabilities between frames, one immediate idea is to sample, from a labeled training set, pairs of pixels that belong to the same semantic category to create $\mathcal{E}_p$ and use pairs between the different categories to create $\mathcal{E}_n$. The result will be a general metric between categories rather than correspondences between pixels and is a much harder problem. Furthermore, as we expect motion in the data sequence, objects will be observed under different viewpoints, and we want our similarity metric to handle this.

We instead generate our pairs of *training* data using pixel correspondences across multiple frames and viewpoints.

(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.

These correspondences can be easily obtained through a variety of different keypoint-based algorithms. Specifically, we use the publicly available SfM package, Bundler [26]. Given a collection of images, Bundler will produce 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 each other to construct $\mathcal{E}_n$ when we learn our metric *offline*. Actual correspondences between 5 pairs of pixels across frames are shown in Fig. 5.

## IV. Temporal Consistency

With the ability to measure similarities between two pixels, we now describe how we combine predictions over time.

### A. Optical Flow

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

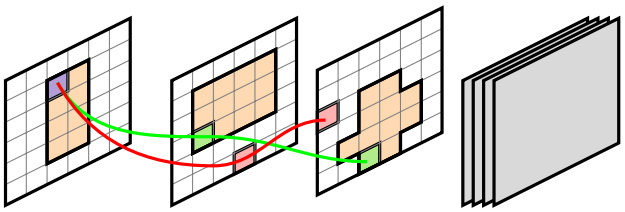


Fig. 4: Generating pairs of training data. The blue and the green pixels are correspondences. The positive examples are pairs of blue and green pixels. Negative examples are pairs between the blue and red pixels. The orange areas represent the same object under multiple viewpoints.

possible so it can potentially be used onboard of a mobile robot. To generate initial hypothesis between frames, we use the efficient dense Anisotropic Huber-L$_1$ optical flow algorithm from Werlberger *et al.* [11].

*1) Data Warping:* Given an image $I_t$ at time $t$, instead of finding each pixels neighborhood in the previous frame $I_{t-1}$, we warp the previous frame into the current frame using the estimated optical flow field. Let $(x_{t-1}, y_{t-1})$ be a pixel coordinate in $I_{t-1}$ and $V_x$ and $V_y$ be the estimated optical flow fields. The coordinates of each pixel with known velocity vector are projected from $I_{t-1}$ to $I_t$ via

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \end{bmatrix} + \begin{bmatrix} V_x[x_{t-1}, y_{t-1}] \\ V_y[x_{t-1}, y_{t-1}] \end{bmatrix}. \quad (8)$$

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 the image data and previous predictions $\hat{\mathbf{y}}_{t-1}$ with small patches, instead of one pixel, in the following way. For each pixel $(x_{t-1}, y_{t-1})$ in frame $I_{t-1}$, we use $V_x[x_{t-1}, y_{t-1}]$ and $V_y[x_{t-1}, y_{t-1}]$ to project all pixels in a small $5 \times 5$ pixel patch centered at $(x_{t-1}, y_{t-1})$ into a warped image $\tilde{I}_t$. For each warped pixel, we accumulate its RGB value and previous temporal prediction $\hat{\mathbf{y}}_i^{(t-1)}$ into its projected coordinate. 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 temporal predictions $\tilde{\mathbf{y}}^{(t)}$

*2) Forward-Backward Error:* The most typical situations when the optical flow estimation fails are occlusions or incorrect estimations. If such a situation occurs, a reasonable behavior is to stop smoothing and to use only the current measurement. Otherwise, we propagate pixels and predictions that might belong to different parts of an image.

A common method to detect optical flow failures is the *forward-backward error*. This technique has demonstrated to be effective for object [27] and point [28] tracking. We detect failures by ensuring that the optical flow forward in time from $I_{t-1} \rightarrow I_t$ is consistent with the flow backwards in time from $I_t \rightarrow I_{t-1}$. Letting $\overrightarrow{V}_x, \overrightarrow{V}_y$ denote the forward flow fields and $\overleftarrow{V}_x, \overleftarrow{V}_y$ denote the backward flow fields, we

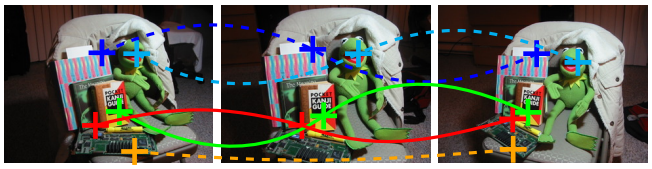


Fig. 5: Example correspondences. Same colored correspondences are used in $\mathcal{E}_p$ and mixed colored correspondences are used in $\mathcal{E}_n$.

consider pixel coordinates $(x_t, y_t)$ in $\tilde{I}_t$ to be invalid if

$$\left\| \begin{bmatrix} \overrightarrow{V}_x[x_{t-1}, y_{t-1}] \\ \overrightarrow{V}_y[x_{t-1}, y_{t-1}] \end{bmatrix} + \begin{bmatrix} \overleftarrow{V}_x[x_t, y_t] \\ \overleftarrow{V}_y[x_t, y_t] \end{bmatrix} \right\|_2 \geq \kappa, \quad (9)$$

where $\kappa = 13$ is related to the size of the averaging neighborhood, since this neighborhood is designed to allow some inaccuracies. Invalid pixels do not contribute to the exponential smoothing algorithm described in the following subsection.

### B. Temporal Smoothing

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

$$\hat{\mathbf{y}}_i^{(t)} = \frac{1}{Z_i} \sum_{j \in N_i^{(t)}} w_{ij} \tilde{\mathbf{y}}_j^{(t)} + c\mathbf{y}_i^{(t)}, \quad (10)$$

where $N_i^{(t)}$ is a $5 \times 5$ spatial neighborhood in $\tilde{I}_t$ centered at pixel $i$ excluding *invalid* pixels as defined by Eq. 9, $c = 0.25$ is our prior belief on the independent prediction from the scene analysis algorithm, and $Z_i = \sum_{j \in N_i} w_{ij} + c$ ensures that $\hat{\mathbf{y}}_i^{(t)}$ sums to one. The procedure then recurses.

## 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 [12] 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

TABLE I: $F_1$ scores and overall accuracies on CamVid

| | 05VD | | 16E5 | |
|---|---|---|---|---|
| **Class** | **Independent** | **Smoothed** | **Independent** | **Smoothed** |
| 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 II: $F_1$ scores and overall accuracies on NYU

| **Class** | **Independent** | **Smoothed** |
|---|---|---|
| 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 III: $F_1$ scores and overall accuracies on MPI

| **Class** | **Independent** | **Smoothed** |
|---|---|---|
| background | .422 | **.730** |
| road | **.497** | .340 |
| lane-marking | **.763** | .330 |
| vehicle | .076 | **.280** |
| sky | .209 | **.560** |
| **Accuracy** | .405 | **.537** |

from [8]. Without using any temporal information, we obtain state-of-the-art overall pixel and average per-class accuracies of $84.2\%$ and $59.5\%$, respectively, and is comparable [13] or exceeds [12] 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 [12], and we evaluate 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 on an urban street with a hand-held camera. In addition to the objects moving, the video has a lot of camera motion due to the person walking. 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 [29] 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 [15], provided by the authors, which is based on the JointBoost algorithm [30].

### 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). Between two frames, the average timing of the first component is $0.02$ s using a GPU implementation on a GeForce GTX 590 graphics card, and the average timing of the second component is $0.75$ s using a CPU implementation on an Intel X5670 processor. 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.

### C. Analysis

Videos comparing per-frame and the temporally smoothed classifications for each sequence are available at [31]–[34], and we show qualitative examples for each sequence in Figs. 6, 7, and 8. 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 I, II, III, 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 displacements 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

TABLE IV: 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 | 93.27 | **93.76** |

sequence are shown in Table IV. 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] W. Brendel and S. Todorovic, "Learning spatiotemporal graphs of human activities," in *ICCV*, 2011.

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

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

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

[13] 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.

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

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

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

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

[18] 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.

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

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

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

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

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

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

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

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

[27] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-Backward Error : Automatic Detection of Tracking Failures," in *ICPR*, 2010.

[28] N. Sundaram, T. Brox, and K. Krutzer, "Dense point trajectories by gpu-accelerated large displacement optical flow," in *ECCV*, 2010.

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

[30] 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.

[31] D. Munoz, "Temporal consistency on CamVid 05VD," http://www.youtube.com/watch?v=q071f75nWNk.

[32] ——, "Temporal consistency on CamVid 16E5," http://www.youtube.com/watch?v=iEgljSK8Oxc.

[33] ——, "Temporal consistency on NYU Scenes," http://www.youtube.com/watch?v=4pVzNaUd2yw.

[34] ——, "Temporal consistency on MPI-StreetScenes," http://www.youtube.com/watch?v=7MeZLVFxRC4.

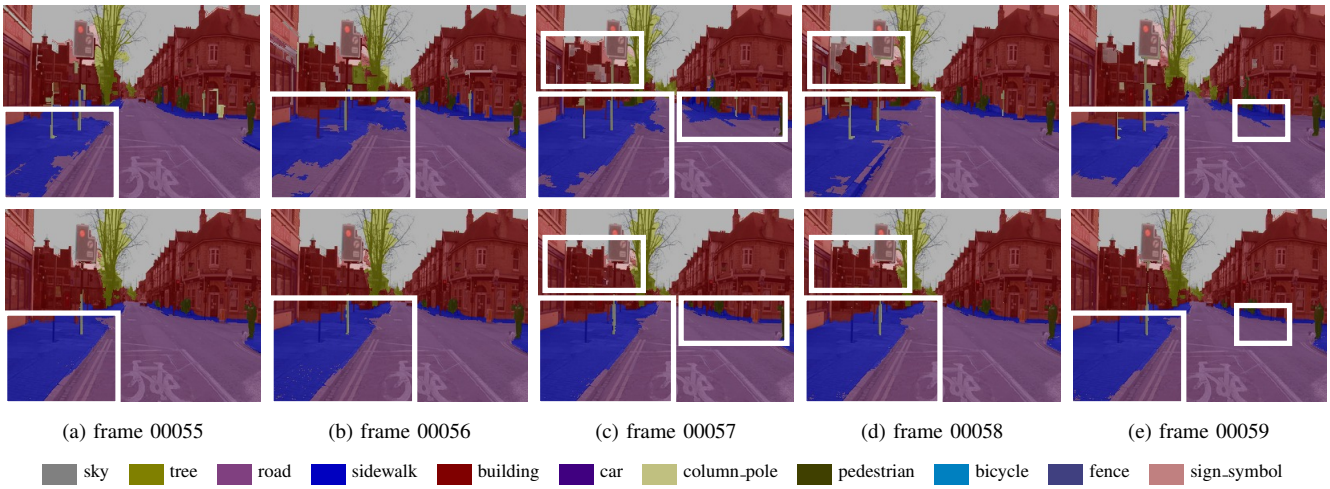



Fig. 6: `CamVid` classifications. Top: per-frame. Bottom: temporally smoothed. Inconsistent predictions are highlighted in white boxes. The full videos are available at [31], [32].

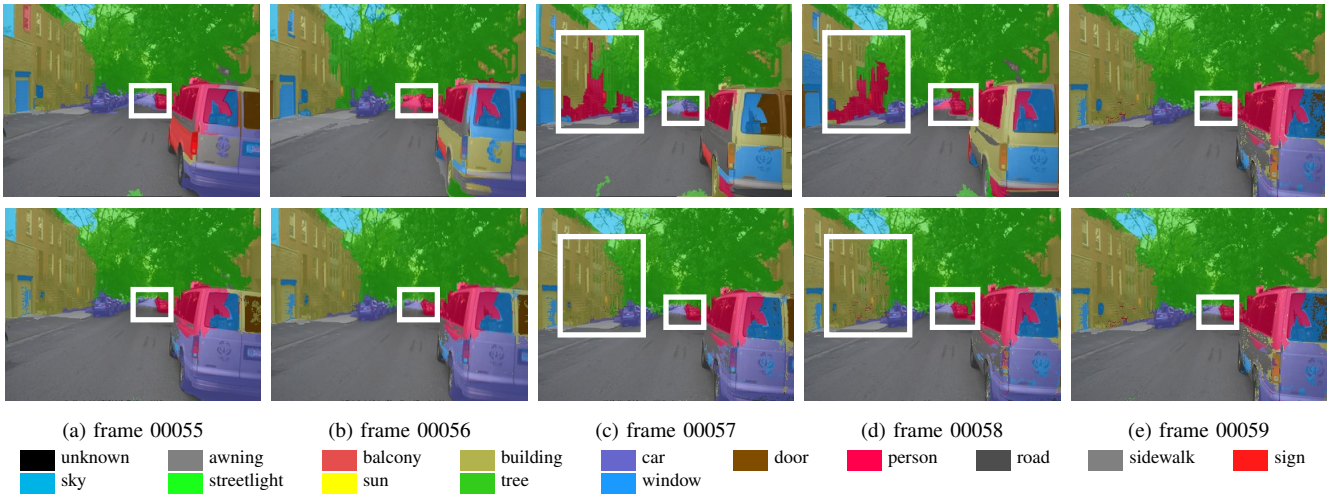



Fig. 7: `NYUScenes` classifications. Top: per-frame. Bottom: temporally smoothed. Inconsistent predictions are highlighted in white boxes. The full video is available at [33].

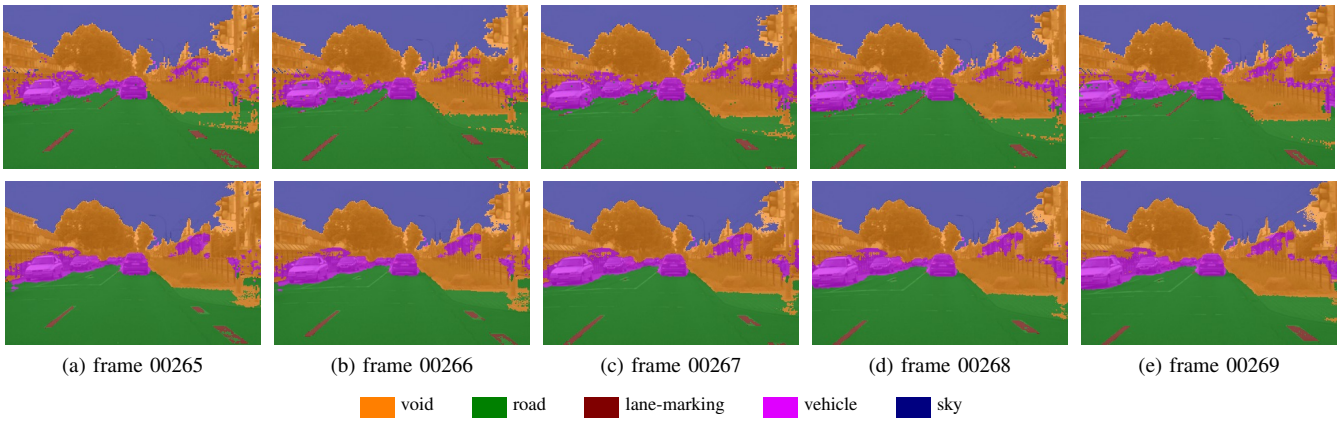



Fig. 8: `MPI-VehicleScenes` classifications. Top: per-frame. Bottom: temporally smoothed. The misclassifications between objects are not as drastic due to the small set of labels, however, there is a lot of noise in the predictions that are corrected. The full video is available at [34].