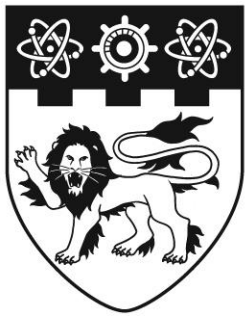
Title	Exploiting visual context and consistency for semantic segmentation(Thesis)
Author(s)	Kang, Dang
Citation	Kang, D. (2018). Exploiting visual context and consistency for semantic segmentation. Doctoral thesis, Nanyang Technological University, Singapore.
Date	2018-11-07
URL	http://hdl.handle.net/10220/46581
Rights	



NANYANG TECHNOLOGICAL UNIVERSITY

EXPLOITING VISUAL CONTEXT AND CONSISTENCY FOR SEMANTIC SEGMENTATION

DANG KANG

SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING

2018

EXPLOITING VISUAL CONTEXT AND CONSISTENCY FOR SEMANTIC SEGMENTATION

DANG KANG

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University in partial fulfillment of the requirement for the degree of Doctor of Philosophy

2018

Acknowledgment

I wish to express my sincere gratitude towards my supervisor Prof Yuan Junsong and Prof Ma Kai Kuang. Prof Yuan has been a great mentor and has provided substantial support, guidance, and encouragements on both Ph.D. projects and career developments. He is a role model not only as a good researcher but also as a person who always treats others with respect and kindness. I appreciate his patience and understanding. Prof Ma has provided useful instructions on my presentation skills and has shared many great advice for personal developments.

I wish to thank all my teammates and friends for their help, especially Gang Yu, Ganqiang Zhao, Chunluan Zhou, Jiong Yang, Suchen Wang, Chaoqun Weng, Junwu Weng, Hongxin Wang, Yuwei Wu, Bo Hu, Weixiang Hong, Jianfeng Ren, Hui Liang, Mengjia Yan, Ankang Liu, Wei Ye, Liyong Lin among others, who have helped me a lot in study and daily life.

Finally, my gratitude goes to my family for their love and care.

Abstract

This thesis studies the problem of semantic segmentation in both images and videos. It targets at addressing two challenges of semantic segmentation: robustness and consistency. First, assigning pixels to the corresponding semantic categories may be unreliable especially under adverse imaging conditions. Second, existing semantic segmentation algorithms often produce inconsistent labeling at region boundaries or across neighboring frames. Visual context and consistency cues are highly useful to address the two challenges, hence the objective of this thesis is to develop techniques that exploit these cues to enhance the performance of semantic segmentation. The thesis consists of the following three technical works.

The first work is to exploit spatial layout context for semantic segmentation in images. When parsing images with regular spatial layout, the pixel location provides important prior for its semantic label and can be highly complementary to appearance cues. Therefore this thesis proposes a novel way to leverage both location and appearance information for pixel labeling. The proposed method utilizes the spatial layout by building a field of local pixel classifiers that are location-constrained, *i.e.*, trained with pixels from a local neighborhood region only. Our proposed local learning works well in challenging image parsing problems, such as pedestrian parsing, street-view scene parsing, and object segmentation, and outperforms existing methods that rely on one unified pixel classifier. To better understand the behavior of our local classifier, we perform theoretical analysis to explain why the local classifier is more discriminative and can handle misalignment.

Abstract

The second work is to exploit spatio-temporal consistency for semantic segmentation in video streams. We propose an efficient online video smoothing method, called adaptive exponential smoothing (AES), to refine pixel classification maps in a video stream. We first trace each pixel in the past frames by finding an optimal spatio-temporal path; then temporal smoothing is performed over the found path with exponentially decreasing weights over time. Thanks to the pixel tracing, AES is adaptive and non-linear; thus it can better improve the "flickering" maps while avoiding over-smoothing. To enable real-time smoothing, a linear-complexity dynamic programming scheme is designed to trace all pixels simultaneously in the video stream. We apply the proposed smoothing method to improve both saliency detection maps and scene parsing maps. The comparisons with average and exponential filters, as well as more sophisticated approaches that rely on optical flow computations, validate that our AES can effectively refine the pixel classification maps in real-time.

The third work is to exploit region consistency for actor-action semantic segmentation, *i.e.*, joint labeling of actor and action categories for frames in videos. One major challenge is that different body parts provide different levels of action cues and may have inconsistent labeling when they are labeled independently. To address this issue, we utilize high-quality region masks from instance segmentation, and enforce pixels inside the region mask to take the same action label to achieve consistent labeling. Our approach uses a two-stream network which captures both appearance and motion information, followed by region-based actor-action segmentation networks which take the fused features from the two-stream network as the input. Our experiments on the A2D dataset demonstrate that both the region-based segmentation strategy and fused features from the two-stream network contribute to the performance improvements, which lead to significantly better results when compared with state-of-the-art methods.

Contents

List of Figures

List of Tables

List of Symbols

(x, y)	spatial coordinate of a pixel in an image
(x, y, t)	spatio-temporal coordinate of a pixel in a video
$\mathbf{f}$	feature vector of a pixel
l	semantic label of a pixel
$\mathcal{N}$	spatial neighborhood of a pixel
$\mathcal{I}$	an image
W	height of an image
H	width of an image
T	video length
p	probability value
$loss$	loss function
$\mathcal{P}$	a spatio-temporal path

Chapter 1

Introduction

1.1 Motivation and Objectives

This thesis focuses on the problem of semantic segmentation(used interchangeably with semantic labeling, pixel labeling, image parsing and video parsing in the rest of this document) in both images and videos. Semantic segmentation aims to assign each pixel one of the predefined categories such as cars, people, and road. It is a fundamental problem in computer vision with many applications including autonomous driving, robotics, computer-aided diagnosis, surveillance video analysis, image and video editing.

There are two main challenges in semantic segmentation: robustness and consistency (see Fig. 1.1 for some examples). The first challenge is that associating pixels with the corresponding semantic categories can be unreliable due to adverse lighting conditions, poor resolutions, large intra-class variations, etc. The second challenge is that existing algorithms often produce inconsistent labeling at region boundaries or across neighboring frames because of camera motion, object motion, illumination changes, adverse imaging conditions, sensor noises, etc. The inconsistent predictions can significantly affect certain applications, such as robotics tasks where obstacles

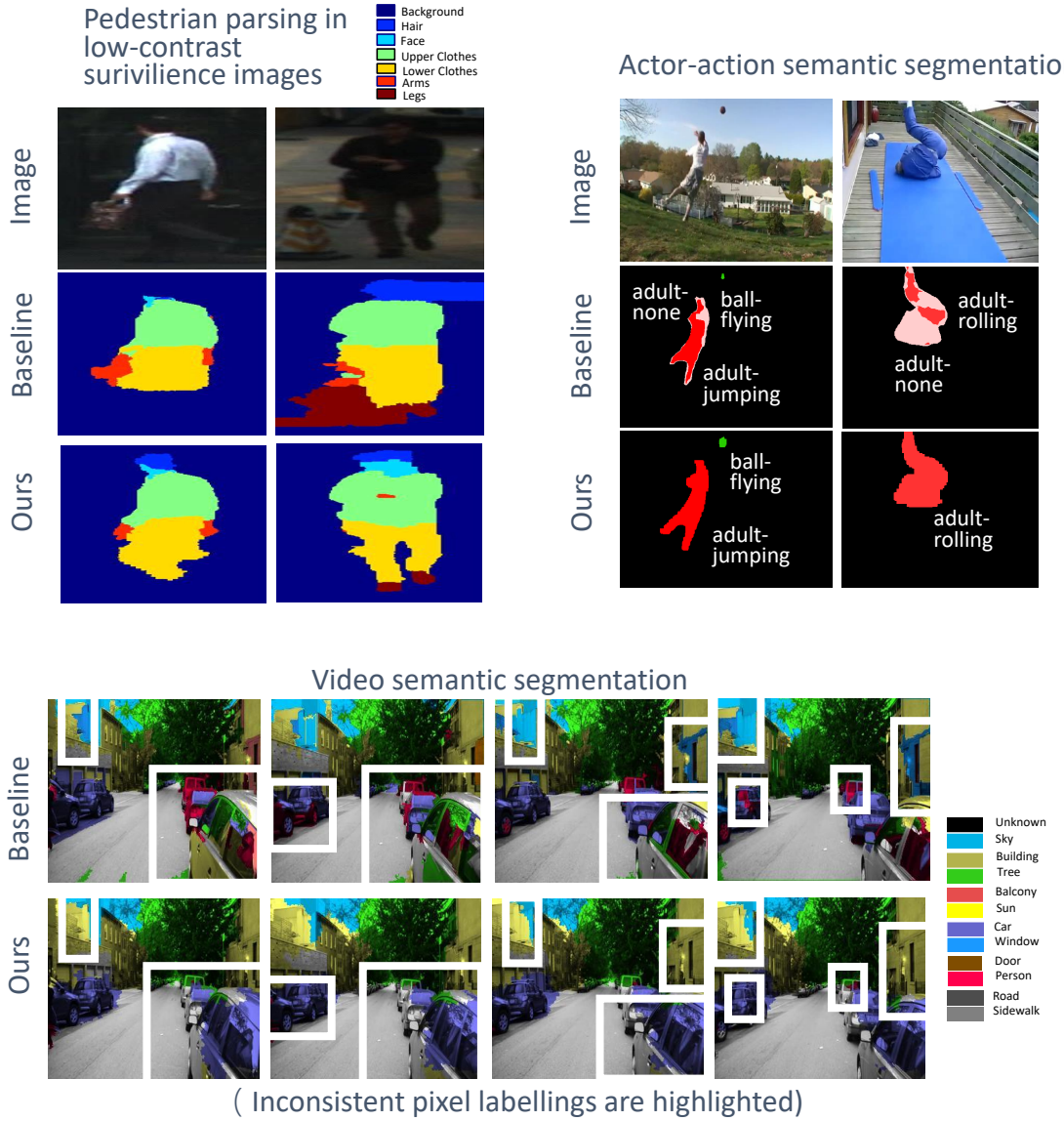


Figure 1.1: Examples of challenging semantic segmentation problems studied by this thesis. The first problem is to improve the robustness of pedestrian parsing for low-contrast surveillance images. The second problem is to improve the spatial-temporal consistency in video semantic segmentation, so that the pixel labeling is consistent across neighboring frames. The third problem is to improve the region consistency for actor-action semantic segmentation, such that the labeling of actor-action categories is consistent with region boundaries. The figure shows some results of our methods compared with the previous approaches for these problems.

appear and disappear suddenly in neighboring frames. A well-designed semantic segmentation algorithm should handle both simultaneously.

To improve semantic segmentation robustness, existing works [2, 7, 11–15] often exploit *visual context* cues. Visual context cues [16] can be roughly defined as any visual information outside the local region. They may be represented by relative layout feature [11,12], global layout feature [7,13], hierarchical layout feature [14] in traditional semantic segmentation pipelines, and dilation convolution operator [2], pyramid pooling operator [15] and locally connected layer [17] in deep learning frameworks. Visual context cues are complementary to local cues such as color, gradient, and texture, hence can improve the pixel classification robustness especially under adverse conditions.

To improve semantic segmentation consistency, previous works [2, 10, 18–26] frequently exploit *consistency* cues. In these approaches, the output pixel labels are forced to depend on each other, so the joint labeling is consistent with some predefined criteria. For example, in a video sequence, a pixel is more likely to belong to a certain category if nearby pixels in previous frames also belong to the same category. Another example is that the same label should be assigned to all the pixels in an image region with the similar appearance. Consistency cues are commonly incorporated in probabilistic graphical models [18–21], image and video filters [10, 22], iterative refinement procedures [2,23,24], and region-based approaches [25,26] either as separate post-processing steps or simultaneously enforced with the pixel classification operation. The success of the previous consistency enforcement approaches drives us to further explore in this direction.

Thanks to the rapid growth of data, continuous evolution of algorithms and the resultant expansion of applications, it is of great interests to further develop techniques of the above two major directions in semantic segmentation. Therefore the objective of this thesis is to develop principled approaches in exploiting visual con-

text and consistency cues for semantic segmentation. The following problems are studied in the three technical chapters of the thesis.

- The first problem is to exploit global layout context or absolute location for image parsing. Previous methods [1, 9, 11, 27–31] incorporate location information by fusing pixel absolution location (x, y) and appearance feature $\mathbf{f}$ together. They train one unified pixel classifier using different fusion strategies, *e.g.*, early fusion with feature concatenation, or probabilistic late fusion with a Markov Random Field (MRF). However, these strategies have limitations, and the theoretical understanding towards the problem is still lacking. Hence it is important to further research alternative approaches of utilizing global layout context with better performance and strong theoretical justifications.

- The second problem is to exploit spatio-temporal consistency for pixel classification map smoothing. Different from the first topic where we process each image independently, this problem is about online smoothing of the pixel classification maps based on previous frames. It can significantly reduce the "flickering" classifications over time due to the spatio-temporal inconsistencies and noisy classifications. Different from existing approaches [10] that rely on computationally expensive optical flow calculation, our goal is to propose methods that are efficient, effective, and easy to implement for real-time applications.

- The third problem is to exploit region consistency for actor-action semantic segmentation in videos. Actor-action semantic segmentation is a challenging problem requiring the joint labeling of both action categories, *e.g.*, running and jumping, and actor categories, *e.g.*, adult and cat. One main issue for this task is that when an actor performs an action, different body parts of

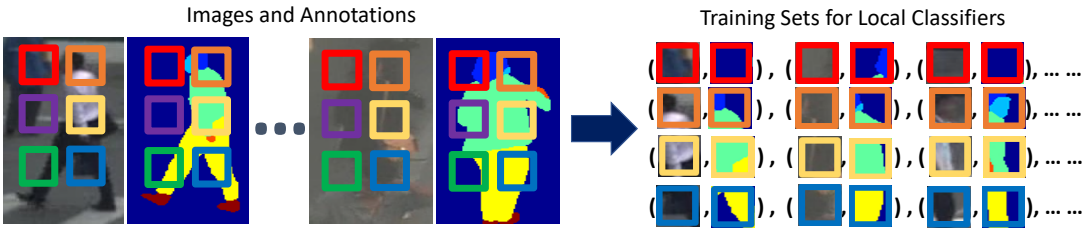


Figure 1.2: In Chapter 3 we study how to fuse location and appearance information to parse images with regular spatial layout. Instead of training a single global pixel classifier for the entire image, we train a field of local pixel classifiers where each local classifier is location constrained. In this way, the learning of each local classifier becomes easier, as within a local neighborhood both the input image appearance and output labelling distribution become simpler.

the actor provide different levels of cues for the action category and may have inconsistent action labeling when they are labeled independently. Different from previous works [32–35], our key assumption is that incorporating high-quality region masks can enforce consistent predictions over actor regions and partially resolve such inconsistent action labeling. Hence our goal is to design and experiment with an end-to-end region-based actor-action segmentation network to confirm the usefulness of incorporating high-quality region masks for this task.

1.2 Contributions of the Thesis

This section summarizes the three contributions corresponding to the research problems defined in Sec. 1.1. The thesis is developed during a period of dramatic changes in the state-of-the-art techniques. As a consequence, the first two works mostly use traditional frameworks with hand-craft features, while the last work takes advantages of the recent deep learning techniques.

1.2.1 Exploiting Layout Context

When parsing images with regular spatial layout, the location of a pixel (x, y) can provide important prior for its semantic label. In Chapter 3, we propose a technique to leverage both location and appearance information for pixel labeling. The main novelty of the proposed method lies in the adoption of local learning paradigm. Instead of relying on one unified pixel classifier, the proposed method utilizes the spatial layout of the image by building a field of local pixel classifiers that are location-constrained, *i.e.*, trained with pixels from a local neighborhood region only. Fig. 1.2 illustrates the motivation behind location-constraint pixel classifiers for layout modeling.

Experimentally we demonstrate the proposed local learning works well in challenging image parsing problems, such as pedestrian parsing, street-view scene parsing, and object segmentation. To better understand the behavior of our local classifier, we perform bias-variance analysis, and demonstrate that the proposed local classifier essentially performs spatial smoothness over the target estimator that uses appearance information and location, which explains why the local classifier is more discriminative but can still handle misalignment. Our theoretical and experimental studies provide useful insights on selecting a critical parameter, the neighborhood size of location-constrained learning which can significantly influence the parsing results.

1.2.2 Exploiting Spatio-Temporal Consistency

In Chapter 4, we propose an efficient online video smoothing method, called adaptive exponential smoothing (AES) to refine pixel classification maps in a video stream. Motivated by the classic exponential smoothing, we apply exponentially decreasing weights over time to smooth the classification score of each pixel. However, instead of fixing the pixel location to perform temporal smoothing, we trace each pixel in

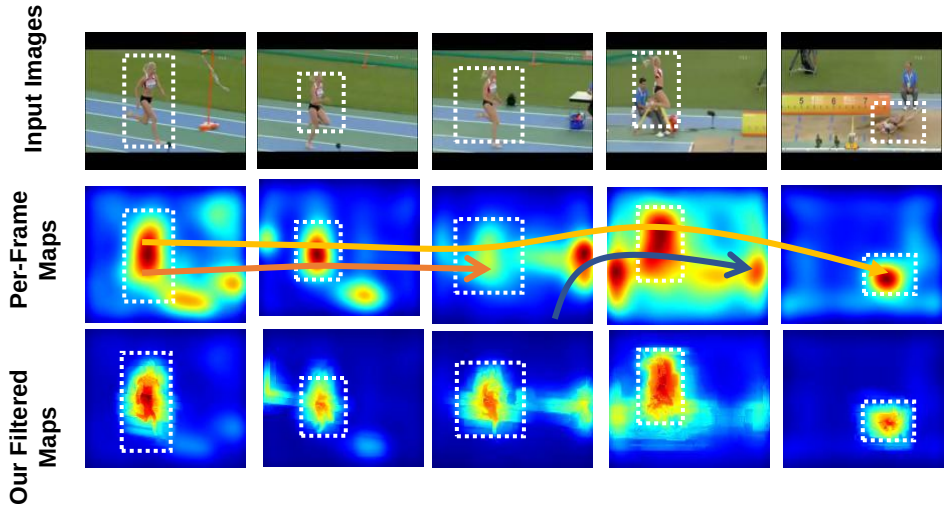


Figure 1.3: In Chapter 4 we study online efficient smoothing of pixel prediction maps. For each pixel we discovery a flexible spatio-temporal filtering path, and perform the exponential smoothing along this found path to remove false alarms or missing detections.

the past frames by finding the spatio-temporal path that can bring the maximum exponential smoothing score, and perform temporal smoothing over the found path (see Fig. 1.3 for an illustration).

Thanks to the pixel tracing, AES is adaptive and non-linear; thus it can better improve the "flickering" maps while avoiding over-smoothing. To enable real-time smoothing, a linear-complexity dynamic programming scheme is designed to trace all pixels simultaneously in the video stream. We apply the proposed smoothing method to improve both saliency detection maps and scene parsing maps. The comparisons with average and exponential filters, as well as more sophisticated approaches that rely on optical flow computations, validate that our AES can effectively refine the pixel classification maps in real-time. To better understand behavior of the proposed method, we also perform extensive analytical experiments to establish the causes that our AES performs well for certain scene parsing videos.

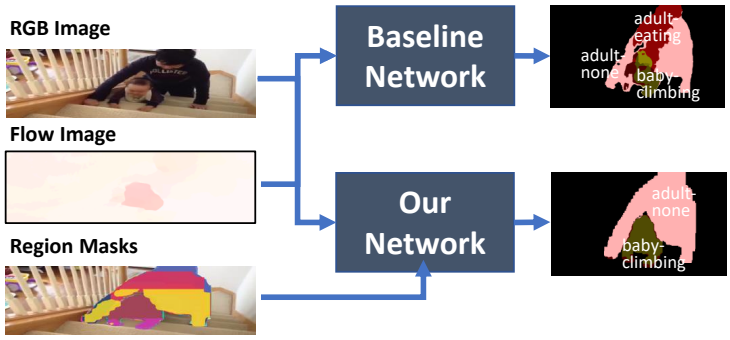


Figure 1.4: In Chapter 5 we study actor-action semantic segmentation. Because many actions are reflected more prominently by movements of certain body parts instead of the whole body, different body parts of the actors may have inconsistent action labelling if pixels are labelled independently. Our main novelty is to exploit region mask as a constraint to avoid labeling pixels independently - instead we assign a single action label to these pixels inside a region mask to achieve consistent action labelling.

1.2.3 Exploiting Region Consistency

In Chapter 5, we study actor-action semantic segmentation, *i.e.*, joint labeling of actor and action categories for all the pixels in videos frames. The main contribution is an end-to-end region-based actor-action segmentation approach which relies on region masks from an instance segmentation algorithm. Instead of labeling pixels in a region mask separately, we assign a single action label to these pixels to achieve consistent action labeling. When a pixel belongs to multiple region masks, max pooling is applied to resolve labeling conflicts. Our approach uses a two-stream network as the frontend which learns two types of features capturing appearance and motion information respectively, and uses two region-based actor-action segmentation networks as the backend which takes the fused features from the two-stream network as the input. Fig. 1.4 provides a high-level overview of the proposed approach.

Our experiments on the A2D dataset demonstrate that both the region-based segmentation strategy and fused features from the two-stream network contribute to the performance improvements. The proposed approach significantly outperforms the state-of-the-art methods in several evaluation settings, which validates the ef-

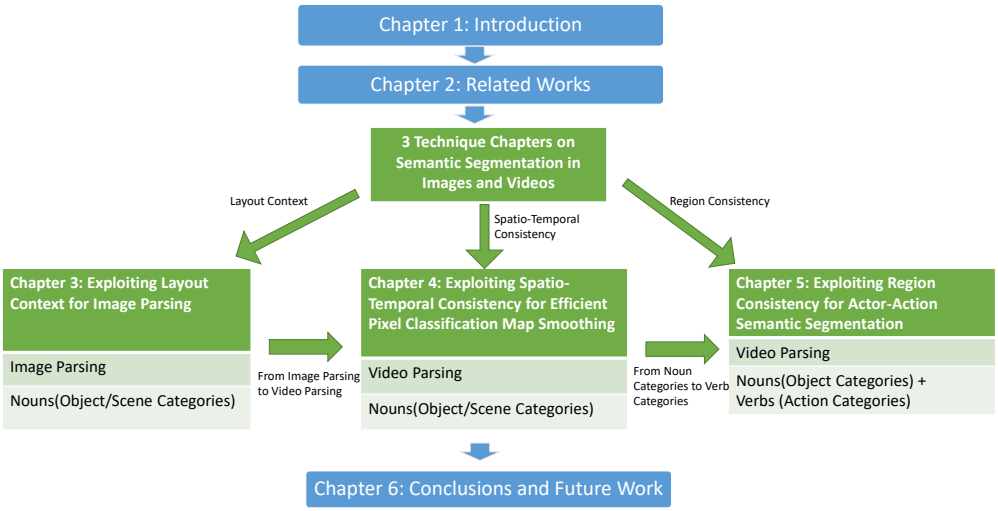


Figure 1.5: Organization of the thesis. We also highlight the connections between three technique chapters.

fectiveness of the proposed method.

1.3 Organization of the Thesis

As illustrated in Fig. 1.5, the thesis is composed of three technical chapters with topics evolve from image parsing to video parsing, and from labeling of actor-only categories to labeling of joint actor-action categories. In Chapter 2, we review important works on image and video semantic segmentation that are mostly related to the thesis. In Chapter 3, we discuss how to exploit spatial layout context for image parsing. This is followed by Chapter 4, where we exploit spatio-temporal consistency in videos to perform online smoothing of image parsing maps. In Chapter 5 we discuss how to exploit region consistency for actor-action semantic segmentation, *i.e.*, joint labeling of both actor and action categories for all the pixels in video frames. It is a more general problem compared with previous chapter topics that only involve actor categories. In Chapter 6 we summarize the whole thesis and list several directions for future studies.

Chapter 2

Literature Review

We review important works on image semantic segmentation and video semantic segmentation in Section 2.1 and Section 2.2, respectively. Considering the scope of this topic, the review is not meant to be exhaustive. Instead, it focuses on certain aspects of visual context and consistency cues that are mostly related to the thesis. In Section 2.3 we highlight the thesis contributions, and summarize the applications and benchmark datasets tested in later chapters.

2.1 Image Semantic Segmentation

2.1.1 Visual Primitives

While many approaches adopt image pixels as the visual primitive and process each pixel individually, it is often beneficial to treat a larger group of pixels together as the whole unit for more complex downstream tasks. We review such visual primitives, *i.e.*, superpixels and region masks, in below.

Image superpixel algorithms partition an image into a set of non-overlapping superpixels. Each superpixel is a small group of pixels with similar low-level properties and can be extracted with a variety of low-level segmentation algorithms [36–40].

Among them, SLIC [40] can produce superpixels of consistent size and shape, is linear complexity in terms of image size, and can run in real-time. There are three benefits of adopting such superpixels as the visual primitives. The first benefit is improved computational efficiency, as the number of extracted superpixel usually only needs to be hundreds to thousands, which is several orders smaller than number of image pixels, so it significantly reduces the number of variables to be considered in later stages. The second benefit is to provide better spatial support for calculating context features, as the superpixel itself can span a larger spatial neighborhood. The third benefit is that as superpixels usually conform to significant image contours, we implicitly enforce the labeling results to be consistent with object boundaries by treating each superpixel as a whole unit. For some of these reasons, we adopt superpixels as the visual primitive in certain experiments of Chapter 3 and 4.

One drawback of superpixel algorithms is that they only provide an over-segmentation of the image, *i.e.*, one object can be partitioned into many superpixels, hence some information is lost. An alternative type of visual primitive is object region masks that can be extracted by generic region proposal approaches [41–44] or instance segmentation algorithms [4, 45], such that each region mask will roughly correspond to one object. Different from superpixel algorithms which divide the image into non-overlapping partitions, region masks are allowed to overlap with each other. As region masks' boundaries roughly match with the actual object boundaries, compared to superpixels, they provide better visual context cues, *e.g.*, object shape or size for their recognition. In addition, they provide better region consistency cues, as we can enforce the pixels within entire mask region to have the identical category label. Considering these advantages, in Chapter 5 we adopt instance segmentation region masks as the visual primitive to perform actor-action semantic segmentation.

2.1.2 Traditional Approaches

Traditional image semantic segmentation pipeline consists of the following stages. For the sake of either computational efficiency or segmentation accuracy, instead of using image pixels, it is common to first generate a set of larger-sized visual primitives, *i.e.*, superpixels or object masks, as the fundamental processing elements for later stages. For each visual primitive, a set of visual features is then extracted within a local region or more distant image areas, and each visual primitive is assigned into one of predefined semantic labels independently. Finally some post-processing steps are applied to ensure the output labelling results of individual visual primitives are consistent with each other, so their joint labeling satisfies certain predefined consistency criteria. Visual primitive extraction has been reviewed in Section 2.1.1, and in below we discuss how to incorporate visual context and consistency cues during the later steps.

2.1.2.1 Visual Context Cues

Visual context cues are commonly encoded during the visual feature extraction stage. In the super-parsing system [13], multiple types of visual features have been extracted for each superpixel, including local color and texture descriptors, as well as visual context features such as location and shape descriptors. In the texton-boost approach [11], the authors extract texton-layout feature for each pixel, which simultaneous encode the dependency between appearance and relative layout of different image regions. Such relatively layout features have also been used in other applications such as medical image analysis [46,47]. Besides relatively layout, another type of spatial context cue—absolute layout—models the dependencies on global image coordinate in an explicit manner. Chapter 3 of the thesis is devoted to the study of absolute layout modeling and we detailedly review absolute layout techniques in Section 2.1.4.

There are two ways of extracting region-based context features from region masks. The first way is to calculate a set of carefully-designed empirical descriptors [41] such as region properties and gestalt properties for each region mask. The second approach is to use the region mask as a spatial support for feature pooling [48], so a single descriptor that summarizes the local features inside the region is generated. The extracted region-based context features are then used to train a regressor to rank the region mask proposals on how likely it covers a foreground object; the top-ranked proposals are used to do recognition and semantic segmentation. Among region-based approaches, second-order pooling [48] was one of the top-performing traditional semantic segmentation methods. Inspired by these and other region-based semantic segmentation approaches [25, 26, 41, 48], we exploit region-based context cue for the actor-action segmentation problem in Chapter 5.

2.1.2.2 Consistency Cues

The most common way to perform consistency enforcement is applying a probabilistic graphical model such as Markov random field (MRF) or conditional random field (CRF). For example, in the super-parsing system [13], a MRF with contrast-enhanced potties model is used to ensure pairwise superpixel labels are consistent to image edges in a local neighborhood. To incorporate region-based constraints, Robust-Pn CRF model [19] is proposed to simultaneously enforcing all the pixel labels inside a region to be consistent with each other. To model long-range interactions, fully connected CRF [18] is proposed to connect all pairs of individual pixels in images, which can produce refined segmentation with a higher amount of details.

Besides probabilistic graphical model, another type of consistency enforcement approaches is iterative refinement procedure, *i.e.*, performing another round of pixel classification by taking the labelling results of previous rounds. For example: the stacked hierarchical labelling approach [14] takes pixel labelling results of large image

regions as inputs to generate the refined pixel labelling over the child subregions with another pixel classifier. In the auto-context algorithm [49], the pixel labelling results of the previous round is used as long-range semantic context information to generate refined segmentation in another round of pixel classification. Compared to probabilistic graphical models, iterative refinement approaches are usually more computationally efficient, and more flexible as their formulations are not restricted to a particular parametric form. However, their performance could suffer in case of insufficient training data.

2.1.3 Deep-Learning Methods

Deep-learning has revolutionized the performance of a variety of computer vision tasks including semantic segmentation. In particular, fully convolution network (FCN) from Long*et al.* [50] has popularized deep-learning architectures for dense prediction tasks, and most current state-of-the-art approaches [2, 3, 15, 20, 26, 51–53] on semantic segmentation are based on this landmark work. It transforms a classification network for dense prediction tasks by modifying the final fully-connected layers to convolution layers. There are two main factors behind its success. First, as a deep-learning approach, it learns powerful and discriminative features specific to a particular task or dataset, and hence the performance is much better than traditional approaches that use hand-crafted features. Second, it allows the adoption of powerful image classification network models, such as VGG [54], GoogleNet [55], ResNet [56] and DenseNet [57]. As the classification network is often pretrained from large-scale datasets such as ImageNet [58], the performance could be impressive even if it is fine-tuned on a small-scale dataset for semantic segmentation.

One major difficulty in applying deep-learning network to semantic segmentation is the goals of reliability and consistency being somewhat contradictory to each other. For a pixel to be classified reliably, the network has to be invariant to

transformations and variations. However, for the pixel labelling to be consistent to object boundaries, the network should be sensitive to spatial transformations and deformations. To deal with two conflicting goals simultaneously, many works have been proposed to improve the original FCN network with a two-stage pipeline: independent classification of each pixel followed by some consistency enforcement procedures. We will review some important approaches that study how to explicitly incorporate visual context cues during pixel classification stage, and different ways of enforcing consistency to refine the initial pixel labelling.

2.1.3.1 Visual Context Cues

Visual context cues are most frequently encoded as a new type of computational layers, which are trained together end-to-end with a pretrained classification network to perform semantic segmentation. For example, to encode global scene context, in the ParseNet work [59], feature maps average pooled from the entire image are concatenated with the local feature maps before the final classification layer. Zhao *et al.* [15] propose a pyramid scene parsing network to integrate global context information pooled from feature maps at different levels of a spatial pyramid. To encode region context, Holger *et al.* [26] propose an end-to-end image semantic segmentation system. In their approach, each region mask is first classified using a free-form ROI pooling layer, then the mask classification scores are converted into pixel classification scores with a differentiable region-to-pixel layer. Their region-to-pixel layer is also adopted in Chapter 5 of this thesis, where we exploit region context to develop an actor-action semantic segmentation system.

Context cues can also be modeled with a new type of network operator. For example, dilated convolution operator [2, 3] is proposed to rapidly increase the filter receptive field size (*i.e.*, to incorporate long-range context information) while preserving the same feature map resolution. It is one of key elements in many

state-of-the-art semantic segmentation networks [2, 3, 15, 60] and has been successfully applied to other dense prediction tasks [61] as well. By adjusting the dilation rate, dilation convolution operator can also be used to model the multi-scale context information as was done in the DeepLab approach [3]. We adopt the DeepLab network [3] as the basis for developing our approach in Chapter 5 due to its good performance.

2.1.3.2 Consistency Cues

Similar to traditional image semantic segmentation approaches, probability graphical models are commonly used in deep-learning approaches for further refining the segmentation results. For example, in the DeepLab approach [3], a fully-connected CRF [18] is used as a separate component to refine the network outputs. Other works [20, 62, 63] combine deep-learning network and graphical model together for end-to-end learning and inference. In the deep parsing network [62], MRF pairwise smoothing terms are represented using local convolution operators and learned together with the unary feature network. In the CRF-RNN system [20], the iterative mean-field inference procedure of the fully-connected CRF [18] is implemented as a recurrent neural network. Lin *et al.* propose an approach [63] to learn both unary and pairwise terms of a CRF using separate feature networks with the piecewise training procedure. As an alternative to probability graphical model, certain approaches [2, 53] apply a refinement network to enforce the consistency of pixel labelling.

Superpixels or region masks also provide valuable consistency cues, as their boundaries often roughly match with image edges or actual object boundaries. Caesar *et al.* [26] propose a differentiable region-to-pixel layer and free-form region-of-interest pooling layer to improve the object boundary accuracies for image semantic segmentation. Arnab *et al.* [64] exploit object detection box constraints and superpixel

region constraints with an end-to-end trainable higher-order CRF for image semantic segmentation. He *et al.* [25] incorporate superpixel region constraints into a convolutional network to address the multi-view semantic segmentation application. In Chapter 5, we extend this direction of study to another application, where we exploit region mask constraints to improve the consistency of actor-action labeling.

2.1.4 Absolute Layout Context

In this section, we review some most relevant approaches to Chapter 3 of the thesis which utilizes the absolute layout context.

Some of previous works use early fusion to fuse appearance and absolute layout information [1, 13, 27, 30], *i.e.*, a unified discriminative classifier is trained by the concatenation of appearance feature and position coordinates $(\mathbf{f}, x, y)$. For example, Xiao and Quan [27] concatenate the vertical position of a pixel with its appearance descriptor for street-scene parsing. Tighe and Lazebnik [13] concatenate the position and appearance descriptor for general scene parsing. Akselrod-Ballin *et al.* [29] and Tu and Toga [30] aggregate 3-d location coordinates and appearance feature for brain structure segmentation. Despite of being a popular approach, the performance of this approach can be affected by the unbalanced concatenated feature vector, as appearance feature $\mathbf{f}$ is often much longer than (x, y).

Besides the early fusion, appearance and absolute location information can be fused via late fusion as well, *i.e.*, separately training $p(l \mid x, y)$ and $p(l \mid \mathbf{f})$ where l stands for the pixel label, and then combining them at a later stage [9, 11, 28, 31]. For example, in Bo and Fowlkes's work for pedestrian parsing [9], location prior is modeled using average shape mask, and combined with other terms by multiplication. In the work of Yang *et al.* for clothing parsing [31], location context is modeled using 2-D Gaussian distributions, and incorporated into the singleton potential and exterior affinity terms for clothes co-labeling. Late fusion provides the

flexibility that appearance and location information can be utilized with different types of models, however it also ignores the correlations between them.

Both the above early and late fusion methods train unified pixel classifiers to label the entire image. In contrast, some other methods [17, 65–67] and our method described in Chapter 3 of this thesis [8, 68] train local pixel classifiers to label local image neighborhoods. For example, Cuingnet *et al.* [65] perform kidney segmentation with a two-stage process: the local image neighborhood containing the kidney is first detected, and then a local pixel classifier is used within the detected neighborhood for subsequent segmentation. Such a two-stage process is also used by Štern *et al.* [69] for anatomical landmark localization. Different from their works, our method does not rely on the previous detection result. Bai *et al.* [66] and Gong and Cheng [67] use local pixel classifiers for interactive segmentation in videos. Their approach is effective due to the strong spatial alignment between consecutive frames. Different from their work, we focus on image parsing problem which has a higher degree of spatial misalignments, so we study bias-variance trade off to better understand and handle such misalignments.

More recent works by Hariharan *et al.* [17], Liu *et al.* [70] and Li *et al.* [4] apply local pixel classifiers into a deep network for a variety of dense labelling tasks, such as simultaneous detection and segmentation, and object part labelling. Different from these works, the focus of Chapter 3 is a theoretical understanding of the technique. We apply local pixel classifiers to different problems, *i.e.*, horse parsing, pedestrian parsing and scene parsing, and study how to handle misalignments via a properly selected neighborhood size.

Approaches		Descriptions
Traditional Approaches	General Scene Layout [11, 13, 46, 47]	• Generic image parsing applications
	Absolute Layout with Feature Fusion [1, 9, 11, 13, 27, 28, 30, 31]	• Applications to street-view scene parsing, medical segmentation, pedestrian parsing and clothes parsing • Performance may be worse than local learning approaches
	Absolute Layout with Local Learning (Existing Approaches) [65–67]	• Applications to medical segmentation and video interactive segmentation
	Absolute Layout with Local Learning **(Our Work [7, 8], Chapter 3)**	• Theoretical understanding of a field of local pixel classifiers for image parsing • Applications to pedestrian parsing and street-view scene parsing, and focusing on how to handle misalignments via a properly selected neighborhood size.
Deep-Learning Approaches	General Scene Layout	• Generic image parsing applications • State-of-the-art performance on benchmark datasets
	Absolute Layout with Locally Connected Layers	• Applications to simultaneous detection and segmentation (instance segmentation) and object part labelling

Table 2.1: Position of Chapter 3 of this thesis with respect to existing layout modeling techniques for image semantic segmentation. We highlight the contributions of various approaches.

2.2 Video Semantic Segmentation

We cover aspects of video semantic segmentation most related to this thesis *i.e.*, utilizing spatio-temporal context and consistency information for video semantic segmentation. Other important but less relevant topics like a generic review of video object segmentation will be excluded.

2.2.1 Spatio-Temporal Context Cues

In deep-learning area, there are two main types of approaches to exploit spatio-temporal context cues for video semantic segmentation: two-stream network and 3d-conv network. The first type of approaches utilize both RGB images and motion (optical-flow) images from the video, and two subnetworks are used to process them separately before fusion at a later stage. The advantage is that each subnetwork usually has similar architecture as existing image classification network, so powerful pretrained model can be adopted. The second type of approaches directly apply 3d convolution operation to the video itself to extract appearance and motion information simultaneously. Their advantage is that the correlations among multiple video frames can be learned using the 3d convolution operator. However as the operator requires more parameters, less number of network layers have to be used to avoid the over-fitting problem. This may negatively affect the results. To achieve better performance, some works combine both two-stream network and 3d-conv network together. Incorporating motion information improves the performance of some interesting applications, such as actor-action semantic segmentation where both the actor categories (*e.g.*, person, bird, and etc.) and the action categories (*e.g.*, running, flying, and etc.) are labelled for each pixel. In Chapter 5 we adopt two-stream network to study the actor-action semantic segmentation problem.

Xu *et al.* [32] introduce the actor-action semantic segmentation application with a

	Descriptions
[32, 35]	• Modeling the interactions between actor and action categories
[34]	• A weakly supervised approach for this application
[33, 71]	• Spatio-temporal consistency enforcement with probabilistic graphical models or LSTM models • Does not explicitly incorporate region mask as consistency constraints
Our Work [72], Chapter 5)	• Region consistency enforcement with high-quality region masks from an instance segmentation algorithm • State-of-the-art performance on the benchmark dataset

Table 2.2: Position of Chapter 5 of this thesis with respect to existing approaches for actor-action semantic segmentation. We highlight the differences in the contributions of various approaches.

large-scale benchmark dataset. They further develop this application by addressing the inconsistent labeling issue with a graphical model [33], and by proposing a weakly supervised method [34] to reduce the labeling efforts. Kalogeiton *et al.* [35] first perform object detection and then refine the bounding-box detections to generate segmentation results; the novelty is to study the joint modeling of actor and action categories with different loss functions. Qiu *et al.* [71] propose to use 3D convolution network and convolutional LSTM for modeling the spatio-temporal dependency. Different from these previous works, in Chapter 5 we directly incorporate high-quality region masks into a deep network for consistent actor-action labeling.

2.2.2 Spatio-Temporal Consistency Enforcement

In this section, we review some most relevant approaches to Chapter 4 of the thesis which performs spatio-temporal consistency enforcement.

Many video analysis methods are built on image-based algorithms which process each video frame independently. For example, the methods in [73–75] and [76] use per-frame visual and motion saliency maps for video object segmentation. Tran *et al.* [77] perform abnormal event detection in videos by generating an abnormality

confidence map per frame. Tighe *et al.* [13] and Miksik *et al.* [10] perform pixel-wise video parsing based on per-frame image parsing results. However, by analyzing each frame independently, the image parsing results can be "flickering" due to the image analysis algorithm's sensitivity to camera and object movements or illumination changes [10]. To reduce the flickering effects and boost the performance, spatio-temporal smoothing is usually needed to enforce the consistency across different frames.

Classic linear causal filters, *e.g.*, the exponential filter [78] has been widely incorporated into different vision applications to model spatio-temporal consistency, such as face tracking [79], gesture recognition [80], pose reconstruction [81], and pixel classification map smoothing [10, 73]. It can well suppress additive noises in static backgrounds but usually tends to overly smooth moving objects. Because it filters each pixel along a straight temporal path, severe tailing artifacts may be introduced on fast moving pixels.

To better deal with moving objects, adaptive smoothing is preferred. Instead of using fixed pixel locations, previous methods [10, 73, 82, 83] connect each pixel adaptively to the previous frame by optical flow or feature flow connection. Miksik *et al.* [10] further apply metric learning so that smoothing is performed adaptively according to the learned appearance similarities. These methods require the original video to be available, and the optical flow or feature flow to be recoverable. In addition, they are relatively computationally intensive. To address these limitations, Yan *et al.* [84] propose a smoothing method that does not rely on the original video, and the method is accelerated by dynamic programming. However, it overly emphasizes the past frames due to the lack of an adaptive temporal weighting scheme.

Probabilistic graphical models [21, 73–75, 85–91] are also used to perform spatio-temporal smoothing. Although most of them run offline, on-line variations [21, 85–88, 90] have also been proposed. In these online models, labeling consistency among

different frames is enforced via pairwise edge terms. To meet the online requirement, some of them restrict the message to be passed only from past frames to the current frame [86–88, 90], and others [21, 85] limit the MRF optimization to be performed within a previous temporal sliding window. While the probabilistic graphical models yield good performance, efficient inference over large graphical model is still a challenging problem. For example, the feature space optimization approach [21] extends the fully connected CRF work [18] to video semantic segmentation, and achieves an impressive performance, but it takes more than 10 seconds to process a single frame. In addition, some of them provide discretized labeling results without retaining the classification scores. However, as argued in Miksik *et al.* [10], classification scores are useful for late decisions, and thus it is preferable that the smoothing can directly refine the classification scores.

While methods like feature space optimization [21] are used as a separate post-processing step to refine deep-learning network outputs, other approaches [92–95] directly incorporate spatio-temporal smoothing techniques into a deep neural network. In this way, the network feature learning and spatio-temporal smoothing processes can be optimized together in an end-to-end manner, and better features that are suitable to the smoothing techniques may be learned. For example, David and Cristian [92] propose an approach that combines a spatial transformation network for optical-flow wrappings with a gated recurrent unit to adaptively smooth semantic information over time. Raghudeep *et al.* [93] wrap intermediate network feature representation via optical flow with end-to-end training using neighboring frames. Besides such filtering based methods, deep probabilistic graphical models such as deep Gaussian random field [95] has been proposed for video semantic segmentation. These methods contain a larger number of parameters and can learn to better cope with the particular type of spatio-temporal inconsistencies or irregularities associated with an image segmentation network. However, they may perform

worse when training data is insufficient, and the trained model may not be able to handle noises different from those presented in training data.

Instead of performing smoothing based on existing classification maps, spatio-temporal consistency may also be enforced during generation of classification maps. Supervoxel methods [96–102] are capable of aggregating the score of each pixel to generate the classification score, and each supervoxel can be considered as a whole for further processing. Wang *et al.* [103] and Sturgess *et al.* [104] integrate spatio-temporal-based features calculated from motion or sift-flow [105] into the classification step. However, even with the prior steps, inconsistent classification may still occur. Thus, a post-smoothing step is still needed.

Regarding the underlying formulation, our work described in Chapter 4 is most closely related to the max-path formation used in Tran *et al.* [77], Zhang *et al.* [75] and Yan *et al.* [84]. Instead of performing a pixel-wise smoothing, the methods in Tran *et al.* [77] and Zhang *et al.* [75] enforce the spatio-temporal constraint by directly finding a single, most confident spatio-temporal smoothing trajectory. Moreover, Zhang *et al.* [75] is designed to filter sparse region proposals instead of dense pixels. Our work is significantly different from Yan *et al.* [84] for the following reasons:

- Our work generalizes classic exponential smoothing to denoise dense pixel confidence maps. Instead, the formulation in [84] is more relevant to moving average. As commonly known that exponential smoothing is significantly different and sometimes more effective than moving average for temporal filtering, it justifies that our method is non-trivially different from [84]. In addition, we prove that dynamic programing can still work in the form of exponential filtering.

- Yan *et al.* [84] applies heuristic procedures such as multiplication of accumulative confidence and original score to pull up the performance. In contrast,

our method is a principled solution and does not rely on such heuristics.

- Our experimental comparisons show that, Yan *et al.* [84] performs much poorer than ours especially in scene parsing filtering. This validates the advantage of our exponential filtering compared with [84]. The later overly emphasizes the past frame thus has less capability in filtering current frame accurately.

- Yan *et al.* [84] is mainly designed for online detection as shown in its experiments. However, our evaluations also include online scene parsing experiments.

Chapter 4 is also related to the well-studied video denoising techniques that are used to reduce the artifacts or noises in video content [83, 106–108]. However, in this work, we propose an online smoothing method to denoise the pixel classification maps produced by the detection or classification models. Some commonly used assumptions in video denoising techniques may not be the best choice to classification map denoising, because the noises mainly originate from the imperfection of the classification models instead of the video content itself.

2.3 Summary

In this chapter, we have detailedly reviewed some important approaches that exploit visual context and consistency cues, for both image and video semantic segmentation. We have provided background information to the following thesis chapters, and has compared and contrasted our work against the existing literature. A summary of position of each thesis chapter with respect to the current literature is shown in Table 2.1, 2.3 and 2.2. In Table 2.4 we provide a list of benchmark datasets used in later experiments.

Approaches		Descriptions
Traditional Approaches	Supervoxel methods [96–104]	• Treat each supervoxel as a whole for classification to enforce consistent pixel labelling within the supervoxel region • Cannot resolve inconsistent labelling results between supervoxel regions
	Adaptive Filters (Other Approaches) [10, 73, 82, 83]	• Based on classic filters [78, 109] • Rely on optical flow or appearance modeling which may be relatively computationally intensive • May not work well when the optical flow is not recoverable
	Adaptive Filters **(Our Work** [110], **Chapter 4)**	• Based on classic filters [78, 109] • Inspired by related approaches using the max-path formulation [75, 77, 84] • Suitable for removing isolated spatio-temporal inconsistencies • May not work well for noises that span a larger area or near boundaries
	Probabilistic Graphical Models [85–88], [21, 73–75]	• Performance is good • Inference over large graph may be inefficient • Certain graphical models can only produce discrete labels without confidence scores
Deep-Learning Approaches	Adaptive Filters and Probabilistic Graphical Models [92, 93, 95]	• State-of-the-art performance on benchmark datasets • A sufficiently large training dataset may be required for good performance.

Table 2.3: Position of Chapter 4 of this thesis with respect to spatio-temporal consistency enforcement techniques for video semantic segmentation. We highlight the strengths and weaknesses of various approaches.

Application	Dataset Name	Dataset Split	Semantic Classes
Chapter 3: Exploiting Layout Context for Image Parsing			
Horse Segmentation	Weizmann horse [111]	328 horses and randomly split into half for training and half for testing	horse and background
Pedestrian Parsing	Penn-Fudan [9]	169 labeled pedestrian images. 68 images for training and 101 images for testing. 700 additional labeled images from HumanEva dataset [112] is also available for training.	hair, face, upper-clothes, lower-clothes, arm, leg and background
	PPSS [1]	2069 images for training and 1892 images for testing.	
Street-View Scene Parsing	CamVid [113]	468 images for training and 232 images for testing.	building, tree, sky, car, sign, road, pedestrian, fence, pole, sidewalk and bicyclist
Chapter 4: Exploiting Spatio-Temporal Consistency for Efficient Pixel Classification Maps Smoothing Note: this chapter aims to improve the quality of pixel classification maps in a video stream, where initial pixel classification maps are generated using an image processing algorithm frame by frame. We specify both the benchmark datasets and the image processing algorithms used for generating the initial classification maps. Our approach does not require training so only number of evaluation images is mentioned.			
Saliency Map Filtering	UCF101 [114]	1599 videos for evaluation	The objective is to improve the saliency map quality so it matches with foreground region inside the ground-truth bounding box. We use the phase discrepancy method [115] to obtain the initial dense saliency map estimations.

Scene Parsing Map Filtering	NYU	74 annotated frames for testing	building, car, door, person, pole, road, sidewalk, sign, sky, tree and window. The initial scene parsing maps are generated from a deep-learning architecture [116].
	MPI [117]	156 annotated frames	background, road, lane, vehicle, sky The initial scene parsing maps are obtained from a boosted classifier [88].
	05VD and 01TP [113]	05VD and 01TP videos contain 5100 and 1831 frames taken at 30 Hz during daylight and dusk, respectively. The sequences are sparsely labeled at 1 Hz with 11 semantic labels.	building, tree, sky, car, sign, road, pedestrian, fence, pole, sidewalk and bicyclist We use hierarchical inference machine [14] for 05VD and our location constraint SVM classifier [7] for 01TP.
Chapter 5: Exploiting Region Consistency for Actor-Action Semantic Segmentation			
Actor-Action Semantic Segmentation	A2D Dataset [32]	9561 frames for training and 2365 for testing.	The dataset consists of 3782 videos which are labeled with both pixel-level actors (adult, baby, ball, bird, car, cat, and dog) and actions (climb, crawl, eat, fly, jump, roll, run, walk) for sparsely sampled frames.

Table 2.4: List of benchmark datasets used in this thesis.

Chapter 3

Exploiting Layout Context for Image Parsing

3.1 Introduction

Spatial layout of an image conveys significant information for labeling its pixels. For example, in street view images, the sky pixels are more likely to appear in the top of the image, and road pixels are more likely to appear at the bottom. As illustrated in Figure 3.1, absolute location is useful for a variety of image parsing problems, such as pedestrian parsing [1, 9], scene parsing [27, 28, 118], and medical image parsing [30, 119]. Previous work [11, 48] demonstrated the significance of pixel classification to achieve good image parsing performance, and it motivates us to study different ways that can leverage both appearance information and absolute location to build better pixel classifiers.

Many approaches have been proposed to fuse absolute position (x, y) and feature vector $\mathbf{f}$ for image parsing [1, 9, 11, 27–31]. Some of them use early fusion, *e.g.*, concatenating feature and position to form $(\mathbf{f}, x, y)$, then training one unified classifier $p(l \mid \mathbf{f}, x, y)$ where l is the pixel label [27, 29, 30]. Although it has been a common

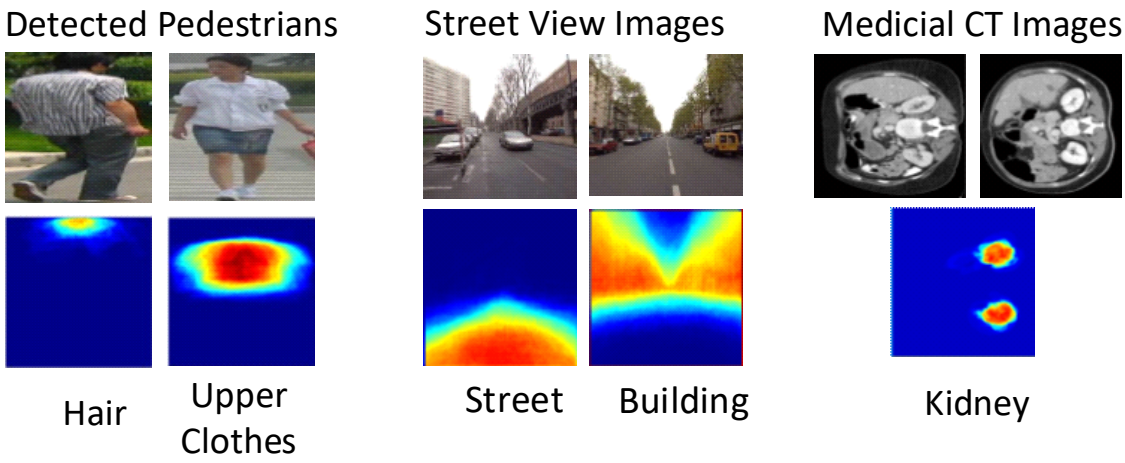


Figure 3.1: Examples of location priors that help pixel labeling. The 1^{st} Row: original images. The 2^{nd} Row: the location priors for specific semantic categories.

practice to use a direct concatenation of appearance feature $\mathbf{f}$ and location prior (x, y) for pixel classification, we argue that such a simple early fusion strategy may not provide satisfactory result as the feature dimension $\mathbf{f}$ is usually much higher than that of (x, y), thus the concatenated feature vector $(\mathbf{f}, x, y)$ can be highly unbalanced and affect the fusion performance. In general, it is not a trivial issue to determine the optimal weight of $\mathbf{f}$ and (x, y) during the early fusion.

Some other methods use late fusion to combine absolute position (x, y) and feature vector $\mathbf{f}$ for image parsing, *e.g.*, they model $p(l \mid \mathbf{f})$ and $p(l \mid x, y)$ separately where l stands for the pixel label, and combine them at a later stage, *e.g.*, via weighted multiplication or Markov Random Field (MRF). For example in scene and clothing parsing [28, 31] the location prior and appearance potential are modeled separately as different unary potentials in the MRF [28] or generative models [31]. However, in these cases, as the location prior and appearance feature are modeled separately, they do not fully explore the correlation information between (x, y) and $\mathbf{f}$.

In both the early fusion and late fusion strategies mentioned above, most of previous works attempt to solve the pixel classification problem with one single model. In other words, they learn one unified pixel classifier to parse the whole image, and

all pixels of the image are used to train the pixel classifier. In contrast, our proposed method is based on local learning. Given location (x, y) and feature $\mathbf{f}$, instead of learning one unified pixel classifier for the entire image, at each image location we learn a separate classifier, *i.e.*, local classifier, and generate a field of local classifiers to classify pixels at different spatial locations. Moreover, each local classifier $p_{\mathcal{N}}(l \mid \mathbf{f})$ is trained only using pixel samples from the neighborhood region $\mathcal{N}(x, y, s)$ centered at (x, y) and of region size s. Since each local pixel classifier is learned by only using the pixels in a local neighborhood, it is expected to better fit the local pixel distribution and capture local discriminative information. Compared with building a unified classifier, such a local learning task is usually easier, as it avoids dealing with confusing pixels outside the local region, which have similar appearance feature but belong to different categories, thus easily confuse the unified classifier. Meanwhile, as the number of classes that can be observed in a local neighborhood is usually not large, learning the local classifier is less challenging too.

The good performance of learning a field of local classifiers to parse an image, however, relies on the good alignment among the images, *e.g.*, training and testing images share similar spatial layout. Such an assumption usually does not strictly hold in real applications, thus we require that the learned local classifier field can handle the misalignments or spatial layout variations. To prevent local classifier overly depending on the image location and to improve its generalization ability, the selection of appropriate neighborhood size s is important. On one hand, if s is too large, each local classifier behaves more like a global one that does not rely on location. On the other hand, if s is too small, local classifiers strongly depend on location and will be sensitive to the image misalignments. To better understand the trade-off, we perform bias-variance analysis on the proposed local classifiers, and establish the relationship between probabilistic distribution of local classifier and target distribution. Our theoretical and experimental results both validate that a

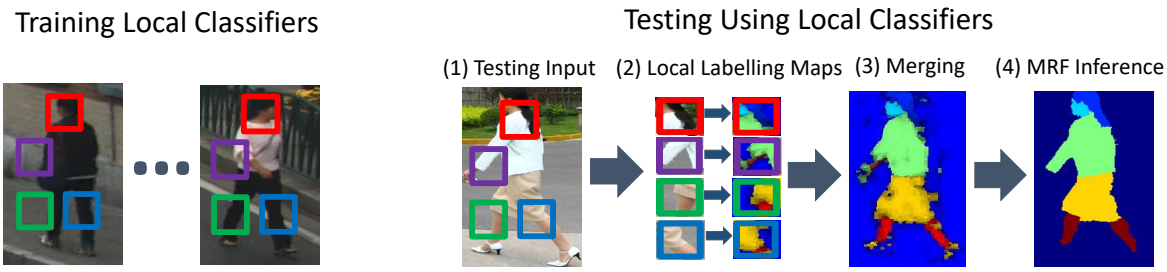


Figure 3.2: The pipeline of our proposed method. At every image location we train a position dependent local pixel classifier with training samples from its neighborhood region, *i.e.*, an image patch. The trained classifier is then used at the same location of the testing image (step 1). Each classifier outputs a local labeling map of its neighborhood region (step 2). Subsequently these local maps are merged to build the overall labeling map (step 3). Finally we refine the overall labeling map with a MRF (step 4).

proper selection of neighborhood size s is critical to obtain good performance.

The main steps of our proposed algorithm are illustrated in Figure 3.2. At each location we learn a location-constrained local classifier with training pixel samples only from a neighborhood. Then the learned classifier will be used to classify pixels of the same region in a testing image. As local classifiers overlap with each other, each pixel will be covered by a few local classifiers. We can perform model averaging to obtain the final classification of the pixel using all engaged local classifiers. As a result, after each local classifier outputs a local labeling map, we merge all local maps to build the overall labeling map. The final result is obtained after a spatial smoothing of the labeling map with a MRF. We verify the performance of our proposed algorithm on two pedestrian parsing datasets [1,9], CamVid scene parsing dataset [113] and Weizmann Horse dataset [111]. In experiments we demonstrate that for these datasets with roughly aligned images, the proposed location constraint pixel classifier achieves significant improvements compared to unified pixel classifier and previous results.

We summarize the contributions of this chapter below:

- To parse images of regular spatial layout, we propose a technique of learning a

field of local pixel classifiers. Compared with learning a unified pixel classifier, our local classifiers can better adapt to the local pixel distribution and perform better in different image parsing experiments.

- To understand the performance of our local classifiers, we establish the connection between probabilistic distribution of local classifier and target distribution, and analyze the property of the local learning neighborhood size via bias-variance analysis.

3.2 Proposed Method

3.2.1 Feature Extraction

Given an image, we first over-segment it using SLIC superpixel [40]. All the subsequent operations are then performed on superpixels instead of pixels to reduce computational cost. We notice that in most cases an image of size 300×300 can be represented with around 5000 superpixels without sacrificing important details such as object boundaries. In the training stage, the label of each superpixel can be set to the majority label of its constituted pixels. However, in the situation that a superpixel contains mixed pixel labels, if the pixels of majority labeling are less than 70%, we consider this superpixel unreliable and do not use it for training.

To describe each superpixel, we use classic hand-crafted features: RFS filter banks [120], dense SIFT [121] and LBP [122]. For each feature type, we construct the dictionary via K-means, and then assign the raw feature of every pixel to its nearest cluster center. With every pixel assigned to a cluster center, we have obtained the texton map of the image and the cluster center index is called texton index [123]. To describe the superpixel, we extract a patch around the superpixel center and then compute the histogram of texton indexes within the patch. After performing

this calculation for all three types of raw features, the final feature is obtained by the concatenation of the three histograms.

Besides the above texton-based feature, our method can also accommodate deep-learning-based feature from convolutional neural networks [2, 51]. We first extract the final feature maps before the network softmax classifier layer. These maps are then upsampled to the input image size to be used as pixel features. For each superpixel, the pixel features within the same superpixel are average pooled to retrieve the corresponding superpixel descriptor.

3.2.2 Training Local Classifier

We have a collection of training images $\mathcal{I} \in \mathbb{R}^{W \times H}$, where W and H stand for width and height of the images respectively. Given a superpixel's central position (x, y) and its associated feature vector $\mathbf{f}$, our goal is to predict the class label l of that superpixel. Instead of simply concatenating $\mathbf{f}$ and (x, y) and training a unified classifier to estimate the target $p(l \mid \mathbf{f}, x, y)$ for superpixels from the whole image $\mathcal{I}$, we are interested in learning a number of local classifiers $p_{\mathcal{N}}(l \mid \mathbf{f})$ at different spatial locations. $\mathcal{N}(x, y, s)$ or its abbreviation $\mathcal{N}$ stands for a local image neighborhood, which is a patch centered at (x, y) and of width $s \times W$ and height $s \times H$. $s \in [0, 1]$ stands for the neighborhood size of local learning. The training set for each local classifier $p_{\mathcal{N}}(l \mid \mathbf{f})$ is $\{(l_i, \mathbf{f}_i) \mid \forall (x_i, y_i) \in \mathcal{N}(x, y, s)\}$. In other words, the i^{th} superpixel's data $(l_i, \mathbf{f}_i)$ will be used for training the local classifier $p_{\mathcal{N}}(l \mid \mathbf{f})$, if the superpixel's central location (x_i, y_i) belongs to the patch $\mathcal{N}(x, y, s)$. We use linear SVM [124] or joint-boost [125] to train the local classifier while other more advanced learning methods can be used as well. In case the training set contains superpixels of the same label k, we just fix the prediction so $p_{\mathcal{N}}(l = k \mid \mathbf{f}) = 1$ and $p_{\mathcal{N}}(l \neq k \mid \mathbf{f}) = 0$.

The trained local classifier approximates the following conditional distribution:

$$p_{\mathcal{N}}(l \mid \mathbf{f}) = \frac{\sum_{(x,y)\in\mathcal{N}} p(l, \mathbf{f}, x, y)}{\sum_{(x,y)\in\mathcal{N}} p(\mathbf{f}, x, y)} \tag{3.1}$$

$$= \frac{\sum_{(x,y)\in\mathcal{N}} p(l \mid \mathbf{f}, x, y) p(\mathbf{f}|x, y)}{\sum_{(x,y)\in\mathcal{N}} p(\mathbf{f} \mid x, y)}$$

$$\propto \sum_{(x,y)\in\mathcal{N}} p(l \mid \mathbf{f}, x, y) p(\mathbf{f} \mid x, y), \tag{3.2}$$

As these are local distributions, we perform summation only for locations belonging to the patch $\mathcal{N}$.

Equation 3.2 explains the relationship between probabilistic distribution of our local classifier $p_{\mathcal{N}}(l \mid \mathbf{f})$ and the target distribution $p(l \mid \mathbf{f}, x, y)$. The proposed local classifier $p_N(l \mid \mathbf{f})$ is a spatially smoothed version of the target $p(l \mid \mathbf{f}, x, y)$ in a local neighborhood, where the weight $p(\mathbf{f} \mid x, y)$ characterizes the dependency of the observed feature $\mathbf{f}$ at the superpixel location (x, y). As a result, $p(\mathbf{f} \mid x, y)$ actually serves as a smoothing kernel over the domain $\mathcal{N}$ and s determines the smoothing bandwidth. The neighborhood size s plays an important role in building the local classifier. On one hand, when the local neighborhood contains only a single spatial location, we abuse the notation to denote it with $s = 0$. Our local classifier degenerates into the target:

$$p_{\mathcal{N}(x,y,0)}(l \mid \mathbf{f}) = p(l \mid \mathbf{f}, x, y). \tag{3.3}$$

On the other hand, when the local neighborhood $\mathcal{N}(x, y, s)$ expands to the entire image, it becomes a global classifier $p_{\mathcal{N}(x,y,1)}(l \mid \mathbf{f}) = p(l \mid \mathbf{f})$, which indicates position information (x, y) is not utilized at all. In Section 3.3, we will further discuss the significance of selecting an appropriate neighborhood size s for local learning based on bias-variance analysis.

3.2.3 Spatial Interpolation and MRF Inference

Spatial interpolation. As it is computationally intensive to train local classifiers to score all image locations, they are trained coarsely on a uniform grid. We set the grid spacing proportional to the neighborhood size s; so for classifiers of larger s their locations are sampled more coarsely. As local classifiers with larger s have larger spatial support, the nearby classifiers become more similar to each other so we purposely enlarge the sampling distance to reduce the correlation between nearby classifiers. In our implementation the grid spacing is set to $0.2 \times s \times (W, H)$, where s ranges from 0 to 1.

As our local classifiers are coarsely trained on the uniform grid, we use the following methods to interpolate the classifier outputs to every location in image. A simple method is the nearest neighbor:

$$\phi_{nearest}(l \mid \mathbf{f}, x, y) = p_{\mathcal{N}_{(x_k, y_k, s)}}(l \mid \mathbf{f}), \tag{3.4}$$

where (x_k, y_k) is the location of the nearest trained classifier to (x, y). Such a nearest neighbor interpolation may not obtain satisfactory result if the interpolated location is not close to the nearest classifier. Alternatively, we can also use the model averaging to perform interpolation.

After training the local superpixel classifier $p_{\mathcal{N}}(l \mid \mathbf{f})$, it is used to label all superpixels belonging to the same patch $(x, y) \in N$. So at each position (x, y) a superpixel will receive multiple votes from nearby local classifiers that cover the superpixel, *i.e.*, if $(x, y) \in \mathcal{N}$ the local classifier $p_{\mathcal{N}}(l \mid \mathbf{f})$ will vote for (x, y). We collect all local classifiers that can influence (x, y): $\Omega = \{k : (x, y) \in \mathcal{N}_{(x_k, y_k, s)}\}$, where k is the classifier's index. The voting scores of nearby classifiers can be fused

via model averaging:

$$\phi_{ave}(l \mid \mathbf{f}, x, y) = \frac{1}{|\Omega|} \sum_{k \in \Omega} p_{\mathcal{N}_{(x_k, y_k, s)}}(l \mid \mathbf{f}), \tag{3.5}$$

where $|\Omega|$ is the total number of votes for position (x, y).

We further analyze the property of model averaging. From Equation 3.1 and 3.5 we have:

$$\begin{aligned} &\phi_{ave}(l \mid \mathbf{f}, x, y) \\ &= \frac{1}{|\Omega|} \sum_{k \in \Omega} \frac{\sum_{(x', y') \in \mathcal{N}_{(x_k, y_k, s)}} p(l, \mathbf{f}, x', y')}{\sum_{(x', y') \in \mathcal{N}_{(x_k, y_k, s)}} p(\mathbf{f}, x', y')} \\ &\approx \frac{1}{|\Omega|} \frac{\sum_{k \in \Omega} \sum_{(x', y') \in \mathcal{N}_{(x_k, y_k, s)}} p(l, \mathbf{f}, x', y')}{\sum_{(x', y') \in \mathcal{N}_{(x, y, s)}} p(\mathbf{f}, x', y')}, \end{aligned} \tag{3.6}$$

where (x', y') refers to the sample position. The approximation used in Equation 3.6 assumes local spatial smoothness, *i.e.*, for any given $\mathbf{f}$, $p(\mathbf{f}, x', y')$ is a constant for all $(x', y') \in N_{(x,y,2s)}$.

We can write Equation 3.6 in another form:

$$\begin{aligned} &\phi_{ave}(l \mid \mathbf{f}, x, y) \\ &\approx \frac{1}{|\Omega|} \frac{\sum_{(x', y') \in \mathcal{N}_{(x, y, 2s)}} w(x', y', x, y) p(l, \mathbf{f}, x', y')}{\sum_{(x', y') \in \mathcal{N}_{(x, y, s)}} p(\mathbf{f}, x', y')} \qquad (3.7) \\ &= \frac{1}{|\Omega|} \frac{\sum_{(x', y') \in \mathcal{N}_{(x, y, 2s)}} w(x', y', x, y) p(\mathbf{f} \mid x', y') p(l \mid \mathbf{f}, x', y')}{\sum_{(x', y') \in \mathcal{N}_{(x, y, s)}} p(\mathbf{f} \mid x', y')}. \qquad (3.8) \end{aligned}$$

By assuming the local classifiers votes are distributed uniformly in an image, we can obtain the analytic form for weighting factor (please refer to the appendix for the explanation),

$$w(x', y', x, y) \propto (s \times W - |x - x'|)(s \times H - |y - y'|), \tag{3.9}$$

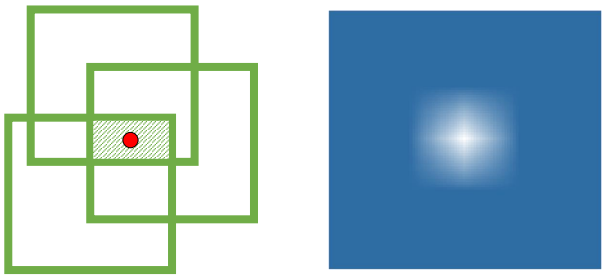


Figure 3.3: Left: each testing pixel is scored by all the nearby overlapping local classifiers represented by the patches, and these scores are merged by averaging. Right: visualization of weighting factor $w(x', y', x, y)$ for the testing pixel at the center.

where $s \times W$ and $s \times H$ are width and height of the neighborhood, respectively. From Equation 3.8 the model averaging result is a weighted summation of the target $p(l \mid \mathbf{f}, x', y')$. The weighting coefficient $p(\mathbf{f} \mid x', y')w(x', y', x, y)$ depends on both the co-occurrences of the observed feature and $w(x', y', x, y)$. From Figure 3.3, we see the weighting factor $w(x', y', x, y)$ places a higher weight for nearby data samples, which behaves similarly as a kernel smoother for density estimation.

MRF Inference. After obtaining the superpixel labeling from the local classifiers, we formulate the image parsing problem as the minimization of a standard MRF energy function over the field of superpixel labels $\mathbf{l} = \{l_i\}$ [126]:

$$E(\mathbf{l}) = \sum_{i \in \mathcal{S}} E_{unary}(l_i; \mathbf{f_i}, x_i, y_i) + k \sum_{(i,j) \in \mathcal{E}} E_{smooth}(l_i, l_j), \tag{3.10}$$

where $\mathcal{S}$ is the set of superpixels for the testing image, $\mathcal{E}$ is the set of pairs of adjacent superpixels, and l_i, $\mathbf{f_i}$, x_i, y_i refer to the label, feature and position coordinates of i-th superpixel, respectively.

The first term considers the unary potential,

$$E_{unary}(l_i; \mathbf{f_i}, x_i, y_i) = -\log(\phi(l_i \mid \mathbf{f_i}, x_i, y_i)), \tag{3.11}$$

where $\phi(l_i \mid \mathbf{f_i}, x_i, y_i)$ can be interpolation using nearest neighbor by Equation 3.4,

or model averaging by Equation 3.5. The second term models the pairwise label smoothness with respect to the appearance similarities between the two neighboring superpixels,

$$E_{smooth}(l_i, l_j) = \delta[l_i \neq l_j] e^{-\gamma ||\mathbf{I}_i - \mathbf{I}_j||^2}, \tag{3.12}$$

where δ represents the indicator function, $\gamma = (2 < ||\mathbf{I}_i - \mathbf{I}_j||^2 >)^{-1}$ [11], and $\mathbf{I}_i$ indicates normalized color histogram of the corresponding superpixel. From the pairwise term, similar neighboring superpixels are more likely to have similar labels. The parameter k is used to balance the influences of two terms, which is obtained from the validation set. We minimize the above energy function using graph-cut with GCO library [1] [127–129].

It is worth noting that [11, 28] combine position and appearance feature into a MRF with an additional location term:

$$E(\mathbf{l}) = \sum_{i \in \mathcal{S}} E_a(l_i; \mathbf{f_i}) + k_1 \sum_{i \in \mathcal{S}} E_l(l_i; x_i, y_i) + k_2 \sum_{(i,j) \in \mathcal{E}} E_{smooth}(l_i, l_j), \tag{3.13}$$

where $E_a(l_i; \mathbf{f_i})$ and $E_l(l_i; x_i, y_i)$ refer to appearance and location term, respectively. Let $E_a(l_i; \mathbf{f_i}) = -\log(\phi(l_i \mid \mathbf{f_i}))$ and $E_l(l_i; x_i, y_i) = -\log(\phi(l_i \mid x_i, y_i))$, the two energy terms can be combined into a single one: $E_{fused}(l_i; \mathbf{f_i}, x_i, y_i) = E_a(l_i; \mathbf{f_i}) + k_1 \times E_l(l_i; x_i, y_i) = -\log(\phi(l_i \mid \mathbf{f_i}) \times \phi(l_i \mid x_i, y_i)^{k_1})$. So potential function of the combined term is in principle a product of expert of the two terms [130]. We will compare the performance of our method and feature fusion using product of expert in experiments.

[1] `http://vision.csd.uwo.ca/code/`

3.3 Neighborhood Size of Local Learning via Bias-Variance Analysis

Bias-variance analysis is commonly used to understand data fitting and model selection. Using bias-variance analysis, we discuss the influence of neighborhood size s for local classifiers in this section.

To perform bias-variance analysis of the local classifiers, we assume to have a large number of training sets, following the target joint distribution $q(l, \mathbf{f}, x, y)$. Each training set can build its own local classifier model $p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})$. To evaluate the loss of a model obtained from a specific training set, we measure the KL-divergence between the model $p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})$ and the target $q(l \mid \mathbf{f}, x, y)$. The target equals to $q_{\mathcal{N}(x,y,0)}(l \mid \mathbf{f})$ according to Equation 3.3.

$$
\begin{aligned}
& loss\Big(p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})\Big) \\
=& \mathbb{E}\Big[d_{KL}\big(\underbrace{p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})}_{model} \;||\; \underbrace{q_{\mathcal{N}(x,y,0)}(l \mid \mathbf{f})}_{target}\big)\Big] && (3.14) \\
=& \mathbb{E}\Big[\sum_{l,\mathbf{f}} p_{\mathcal{N}(x,y,s)}(l, \mathbf{f}) \log \frac{p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})}{q_{\mathcal{N}(x,y,0)}(l \mid \mathbf{f})}\Big] && (3.15) \\
=& \mathbb{E}\Big[\sum_{l,\mathbf{f}} p_{\mathcal{N}(x,y,s)}(l, \mathbf{f}) \log \frac{p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})}{\mathbb{E}\big[p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})\big]}\Big] \\
&+ \sum_{l,\mathbf{f}} \mathbb{E}\big[p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})\big] \log \frac{\mathbb{E}\big[p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})\big]}{q_{\mathcal{N}(x,y,0)}(l \mid \mathbf{f})} && (3.16) \\
&= \underbrace{\mathbb{E}\Big[d_{KL}\Big(\underbrace{p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})}_{model} \;||\; \underbrace{\mathbb{E}\big[p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})\big]}_{average}\Big)\Big]}_{var(p_{\mathcal{N}(x,y,s)}(l|\mathbf{f}))} \\
&+ \underbrace{d_{KL}\Big(\underbrace{\mathbb{E}\big[p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})\big]}_{average} \;||\; \underbrace{q_{\mathcal{N}(x,y,0)}(l \mid \mathbf{f})}_{target}\Big)}_{bias(p_{\mathcal{N}(x,y,s)}(l|\mathbf{f}))}, && (3.17)
\end{aligned}
$$

where the expectation is taken over all the training sets. In Equation 3.17 we have

decomposed the loss into the bias and the variance terms via algebraic manipulations.

Let: $var(s)$ be the variance of testing error averaged over all pixel locations,

$$var(s) = \frac{1}{|\mathcal{I}|} \sum_{(x,y)\in\mathcal{I}} var\Big(p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})\Big), \tag{3.18}$$

where $var(p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f}))$ is defined in Equation 3.17. Based on the bias-variance analysis, Theorem 3.1 shows that the averaged testing error variance will reduce with a larger local learning neighborhood size s. The proof of Theorem 3.1 can be found in the appendix.

Theorem 3.1 *Assume images are aligned across any two datasets i and j for neighborhood size of s_1, such that $\forall(x, y) \in \mathcal{I}$:*

$$\sum_{(x',y')\in\mathcal{N}(x,y,s_1)} p_{i,N}(\mathbf{f}, x', y') = \sum_{(x',y')\in\mathcal{N}(x,y,s_1)} p_{j,N}(\mathbf{f}, x', y'), \tag{3.19}$$

and $s_2 = k \times s_1$ for $k \in \mathbb{N}$, and s_2 is much smaller than images size, we have:

$$var(s_2) \leq var(s_1). \tag{3.20}$$

From Theorem 3.1, using a larger neighborhood size s for local learning will result in a larger overlap among the nearby local classifiers, such that nearby local classifiers behave more similar to each other. Such a smoothness makes the classification less sensitive to the alignment variations, thus can better tolerate spatial misalignments.

For the *bias* term defined by Equation 3.17, by assuming that the ensemble is unbiased, the following approximation holds: $\mathbb{E}\big[p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})\big] \approx q_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})$. The bias term of our model can then be approximated as: $d_{KL}\Big(q_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f}) \,||\, q_{\mathcal{N}(x,y,0)}(l \mid \mathbf{f})\Big)$. From the above approximation, we see that the bias will equal zero when neighborhood size $s = 0$. As KL divergence is non-negative, the bias term will increase

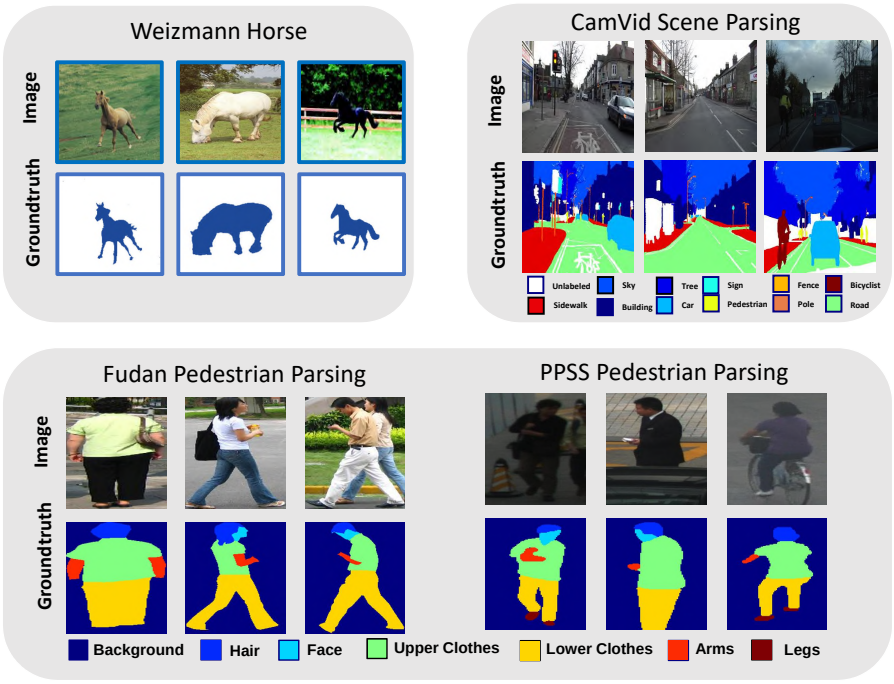


Figure 3.4: Sample images and annotations for the benchmark datasets used in this chapter.

with a larger neighborhood size s. Because the bias increases with neighborhood size s and the variance decreases, an appropriate neighborhood size is important for balancing the bias and the variance for minimizing the testing error.

3.4 Experiments

3.4.1 Experimental Protocols

To evaluate the performance of our proposed method, we perform experiments on three different tasks: (1) object segmentation on Weizmann horse dataset (Section 3.4.2.2); (2) pedestrian parsing (Section 3.4.3.1); and (3) street-view scene parsing (Section 3.4.4). Some sample images and annotations for the benchmark datasets are displayed in Fig. 3.4. For all the experiments we compare our performance with

four existing methods of building a single pixel classifier that relies on either $\mathbf{f}$, or both $\mathbf{f}$ and (x, y):

$\mathbf{f}$ + SVM. We use only appearance feature $\mathbf{f}$ and put it into a SVM classifier. For computational efficiency we use linear SVM.

$(\mathbf{f}, x, y)$ + SVM as used by [29, 131]. We concatenate feature and position information together to form $(\mathbf{f}, x, y)$, and put it into a linear SVM classifier [124].

$(\mathbf{f}, x, y)$ + Boosting as used by [27, 30]. We put the concatenated feature vector $(\mathbf{f}, x, y)$ into a joint boosting classifier [125]. The optimal number of boosting rounds is obtained via a validation set, and it ranges from 2000 to 5000.

Product of Experts as used by [11, 28]. The fusion is performed by multiplying the two posterior probability maps: $\frac{p(l|x,y)^k p(l|\mathbf{f})^{(1-k)}}{Z}$, where k is the weight parameter between 0 and 1, and Z is a normalization constant. As discussed in Section 3.2.3, this is equivalent to the summation of appearance and location energy terms in a MRF setting.

For most experiments we use texton-based descriptor [123]; for street-view scene parsing deep-learning-based feature [2] is also tested. To train each local classifier, linear SVM [124] is used for horse segmentation and pedestrian parsing experiments; for street-view scene parsing we use joint-boost [125] as local classifiers which provide better results. To interpolate the prediction map of the whole image, we use model averaging unless otherwise specified. After spatial interpolation, we perform MRF smoothing. We follow previous works to split the datasets, and reserve 20% from the training set as the validation set. To ensure fair comparisons, all the parameters have been thoroughly adjusted via the validation set, and some parameter settings are stated in Section 3.6.3. As in previous works, we use intersection-over-union

(IOU) score as the evaluation metric, which is defined between ground-truth A and output B by $\frac{|A \cap B|}{|A \cup B|}$.

It is worth noting that pedestrian parsing and street-view scene parsing are multi-class problems, thus we need to address the class imbalancing issue. To better handle the minority class, we do not sample each class in proportional to its prior frequency. Instead, we sample majority class less aggressively and minority labeling more aggressively, in a similar manner proposed by Tighe and Lazebnik [132]. We find such a scheme is critical to achieve good performance.

3.4.2 Results on Horse Segmentation

3.4.2.1 Dataset Description

We first evaluate our method on Weizmann horse dataset [111], which can be obtained from `http://www.msri.org/people/members/eranb/`. It is a binary segmentation problem and most images contain only one horse. Although the images are roughly aligned, e.g., horses are mostly located at the central portion of the images and facing left, they still exhibit large amounts of appearance and pose variations. This makes the dataset challenging if we want to accurately segment the whole horse. The dataset consists of 328 horses and we randomly split the dataset into half for training and half for testing similar to [23, 133].

3.4.2.2 Main Results

We list the performance of our method and the other alternatives in Table 3.1. We also compare the performance of our method with current state of the arts for the dataset. Although our technique [7, 8] was originally published in 2014, its performance is still comparable to or slightly worse than some of recent approaches [133–136, 138]. We also find our method performs reasonably well even for some cases that position, size, or pose of the horses do not follow position prior

Method	IOU score
f + SVM	74.1
(**f**,x,y) + SVM	74.1
(**f**,x,y) + Boosting	77.2
Product of Expert	78.0
Ours (nearest)	**81.2**
Ours (average)	**82.0**

Table 3.1: Comparisons with alternative ways of fusing **f** and (x, y) for Weizmann horse dataset.

Method	Accuracy	IOU score	F1 score
[133]	95.4	-	89.9
[134]	94.6	80.1	-
Ours	**95.6**	**82.0**	**90.1**
Comparisons with recent works			
[135]	95.7	84.0	-
[136]	95.8	84.0	-
[137]	**96.8**	**91.6**	-
[138]	94.6	80.4	-

Table 3.2: Comparisons with existing methods for Weizmann horse dataset. The performance of some recent methods published after our work [7, 8] are also displayed.

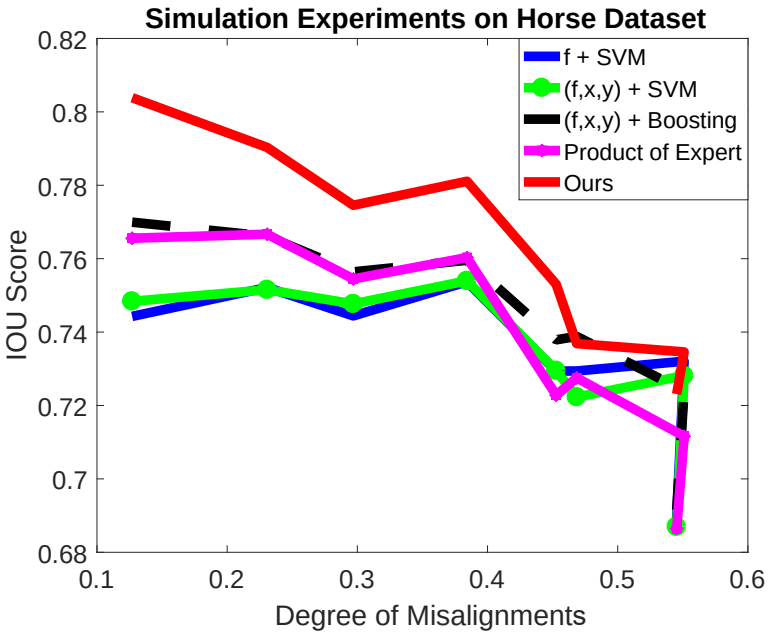


Figure 3.5: Comparisons with alternative ways of fusing $\mathbf{f}$ and (x, y) for simulation experiments on Weizmann horse dataset. Degree of misalignments refers to one minus intersection-over-union between generated and ground-truth boxes.

(see Figure 3.6). These results validate that the field of local classifiers is flexible to handle image misalignments.

3.4.2.3 Simulation Experiments

To further validate that our method can tolerant image misalignments, we perform simulation experiments by randomly generating one bounding box for each image at a random location. Then both the training and the testing are performed only using those randomly generated bounding boxes. The intersection over union (IOU) scores between generated and ground-truth boxes are varied from 0.5 to 1, so different degrees of image misalignments are tested. From Fig. 3.5 and Fig. 3.6 it is clear that when misalignments are small , our method can significantly outperform different approaches that rely on one unified pixel classifier. At the same, in the cases of large misalignments, our local classifiers simply degenerate into the global classifier $\mathbf{f}$ + SVM that does not rely on location, thus become insensitive to such misalignments.

Figure 3.6: Segmentation results from Weizmann horse. In the 1^{st} and 2^{nd} row we show examples from the original dataset, where the horses are mostly aligned to the central portion of the images. To further demonstrate that the method can tolerant misalignments, in the 3^{rd} and 4^{th} row we show some examples from the generated dataset, where the original dataset images have been cropped by the generated bounding boxes of random locations and sizes.

3.4.3 Results on Pedestrian Parsing

3.4.3.1 Dataset Description

We perform the second set of experiments on Penn-Fudan [9] and PPSS dataset [1] for pedestrian parsing. Both datasets contain 7 semantic labels of body parts, such as hair, face, upper-clothes, etc.

- Penn-Fudan contains 169 labeled pedestrian images. For Penn-Fudan, besides the 68 training images from the dataset itself, 700 additional labeled images from HumanEva dataset [112] are also provided for training [9]. However instead of using all the 700 images, for most experiments on this dataset we only use 250 HumanEva images which are randomly selected.

- PPSS contains 3961 labeled images from surveillance video, which can be downloaded from `http://mmlab.ie.cuhk.edu.hk/projects/luoWTiccv2013DDN/index.html`. We use 2069 images for training and the remaining 1892 images for testing, with the same dataset split as in [1].

3.4.3.2 Main Results

We first follow the conventional setting and evaluate the effectiveness of our pedestrian parsing by parsing the pedestrian within the ground-truth bounding box. In such a case, the pedestrian in each bounding box has been roughly aligned. Table 3.3 compares our pedestrian parsing performance with current state-of-the-art methods. It shows that our performance is better than current state of the arts [1, 139, 140] on PPSS and comparable to some of recent approaches [1, 9, 50] on Penn-Fudan. As PPSS's images are extracted from surveillance video, their appearance quality is low, and the location prior becomes more useful. Our method is competitive in such a scenario, as it can effectively leverage location prior and successfully handle challenging cases, such as low contrast images and cluttered background as shown

Penn-Fudan	Hair	Face	UC	Arms	LC	Legs	BG	Overall
[9]	44.9	**60.8**	74.8	26.2	71.2	42.0	81.0	57.3
[1]	43.2	57.1	**77.5**	27.4	**75.3**	52.3	**86.3**	59.9
Ours (protocol of [1,9])	54.0	53.8	63.5	32.3	71.5	48.1	77.7	57.4
Ours	**60.9**	54.9	69.5	**35.9**	73.7	**54.9**	81.3	**61.6**
Comparisons with recent works								
[50]	48.7	49.1	70.2	33.9	69.6	29.9	-	-
[141]	**63.2**	**56.2**	**78.1**	**40.1**	**80.0**	**45.5**	-	-

PPSS	Hair	Face	UC	Arms	LC	Legs	BG	Overall
[1]	35.5	44.1	68.4	17.0	**61.7**	23.8	80.0	47.2
Ours	**55.3**	**45.9**	**71.6**	**31.1**	58.2	**24.8**	**86.4**	**53.3**
Comparisons with recent works								
[139]	51.7	**51.0**	65.9	**29.5**	52.8	20.3	83.8	50.7
[140]	**53.1**	50.2	**69.0**	29.4	**55.9**	**21.4**	**85.7**	**52.1**

Table 3.3: Comparisons with existing methods for pedestrian parsing datasets. "UC" stands for upper-clothes, "LC" for lower-clothes and "BG" for background. "Overall" stands for the average IOU score over all labels. For Penn-Fudan dataset the evaluation results with the modified training set are shown as "Ours", while the ones with the same training set as in [1, 9] are shown as "Ours (protocol of [1, 9])". The performance of some recent methods published after our work [7, 8] are also displayed.

in Figure 3.7.

Table 3.3 also shows that compared to [1, 9], our method performs significantly better on parsing the hair, with an improvement of around 0.1 in IOU score. From Figure 3.7, we see parsing hair is difficult, as it is often confused with the dark background. As our method can model the spatial location prior, it implicitly encodes the spatial distribution information of the hair which is valuable for parsing it.

From Table 3.3, we observe that our parsing is good for upper-cloths, lower-clothes and background but worse for arms and legs. Arms and legs are inherently more challenging categories not only because of pose variations, but also their small area size in the image. As a result, the number of training pixels is less that other classes

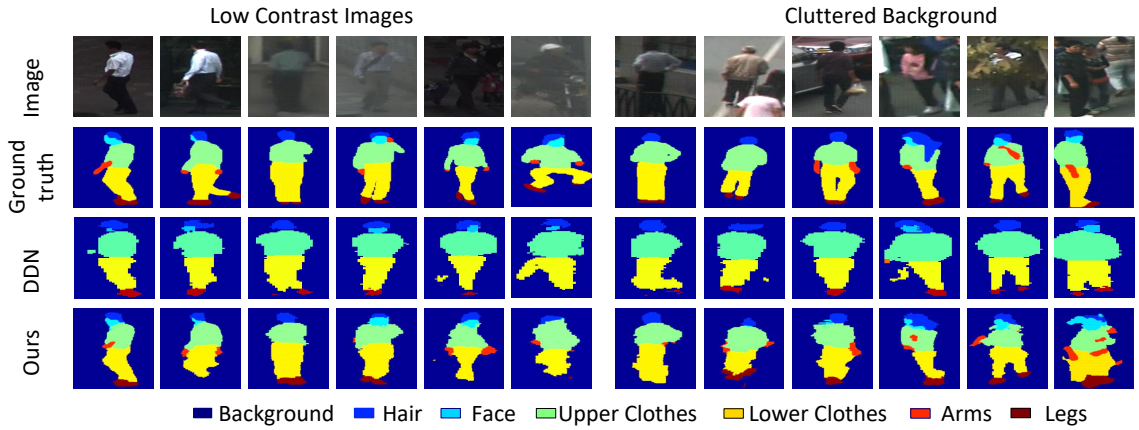


Figure 3.7: Image parsing results from PPSS dataset. Our method achieves good performance in parsing the hair, and can still work in challenging conditions such as low-contrast images and cluttered background. However it performs worse with significant pose variations, or presence of confusing background clutter (6^{th} and 12^{th} column). In the 3^{rd} row we show the comparisons with DDN [1].

and the result of pixel classification is weaker. However, our method still performs better than [1, 9] in the classification of arms and legs.

Table 3.4 shows that our performance is superior to other approaches that fuse $\mathbf{f}$ and location (x, y). In Figure 3.8, we also compare the training errors among different schemes to obtain $p(l \mid \mathbf{f}, x, y)$. It shows that our local learning method has a smaller training error compared with alternative approaches, which partially explains the better performance in the testing phase. To make a fair comparison, for all the methods, the parameters have been thoroughly adjusted from the validation set to prevent under-fitting or over-fitting. From the comparison, it validates that our learned local classifier field can adapt to the local image characteristics and better fit to the data, compared with those who use a unified model to estimate $p(l \mid \mathbf{f}, x, y)$.

3.4.3.3 Parameter Sensitive Analysis

Figure 3.9 and 3.10 illustrate the influences of the neighborhood size s to learn our local classifiers. Figure 3.9 shows that as s increases, accuracy first climbs to the

	Penn-Fudan	PPSS
$\mathbf{f}$ + SVM	35.7	31.0
$(\mathbf{f}, x, y)$ + SVM	49.7	39.1
$(\mathbf{f}, x, y)$ + Boosting	55.2	44.9
Product of Expert	51.2	44.5
Ours (nearest)	58.6	52.1
Ours (average)	**61.6**	**53.3**

Table 3.4: Comparisons with alternative ways of fusing $\mathbf{f}$ and (x, y). The performance metric is the average IOU score over all labels.

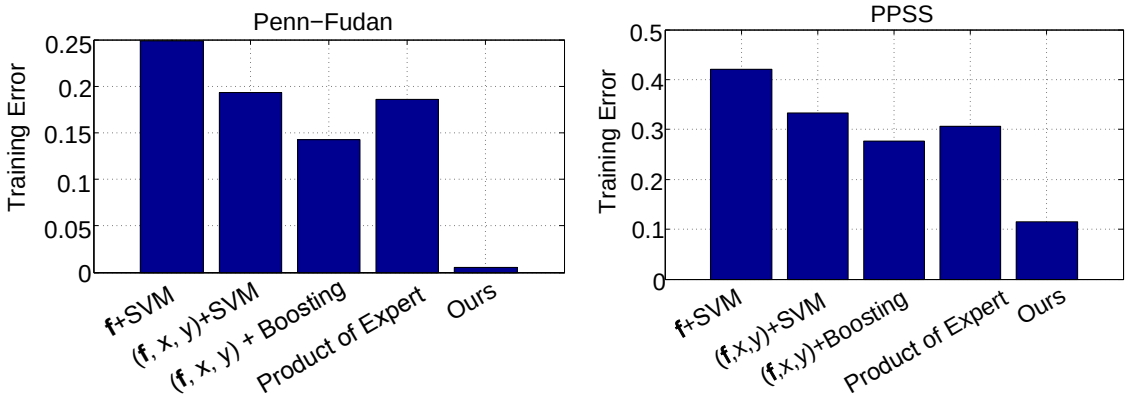


Figure 3.8: Comparisons of average classification error over the training samples for different feature fusion methods.

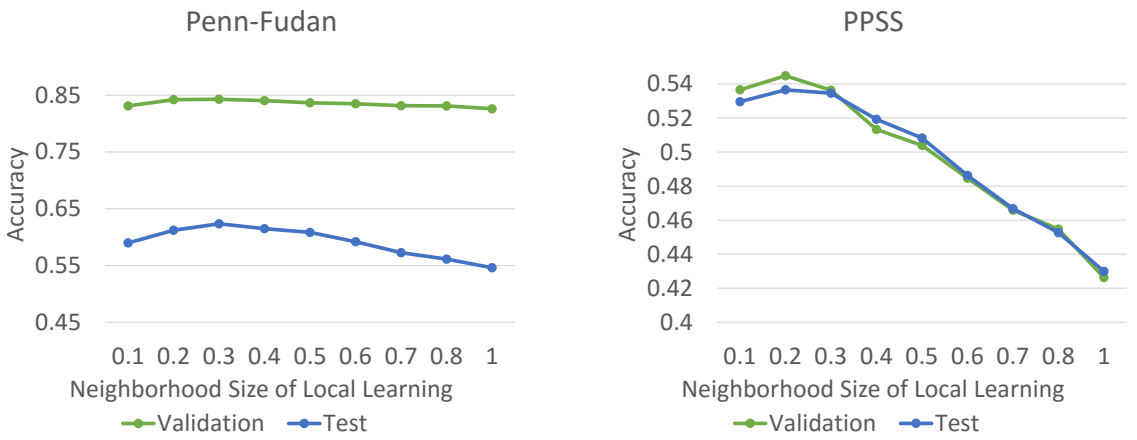


Figure 3.9: Influences of the neighborhood size of local learning on parsing performance. It can be seen that both validation and testing accuracy have similar trend, thus we can use a validation set to select a good neighborhood size

maximum value when s is optimal, and then drops quickly. A good neighborhood size can be found using the validation set. Figure 3.10 shows that when s is very small such as $s = 0.1$, location prior dominates. In such a case, it introduces significant error when the testing image is not aligned to the position prior as shown in the second and the third row. On the other hand, if the neighborhood size is too large such as $s = 0.4$, location information is "smoothed out" and consequently errors in the posterior map cannot be corrected as shown in the first and the second row. As a result, the selection of a good neighborhood size for local learning is important for obtaining good performance.

As mentioned in Section 3.2.3, we apply an interpolation scheme to avoid training local classifiers at every location. In Table 3.4, we evaluate and compare the performance of different interpolation schemes. It shows that model averaging performs slightly better than nearest neighbor interpolation.

3.4.3.4 Pedestrian parsing in detected bounding box.

To further evaluate the capability of our local learning to handle image misalignments, we choose to parse each pedestrian in the detected bounding box using

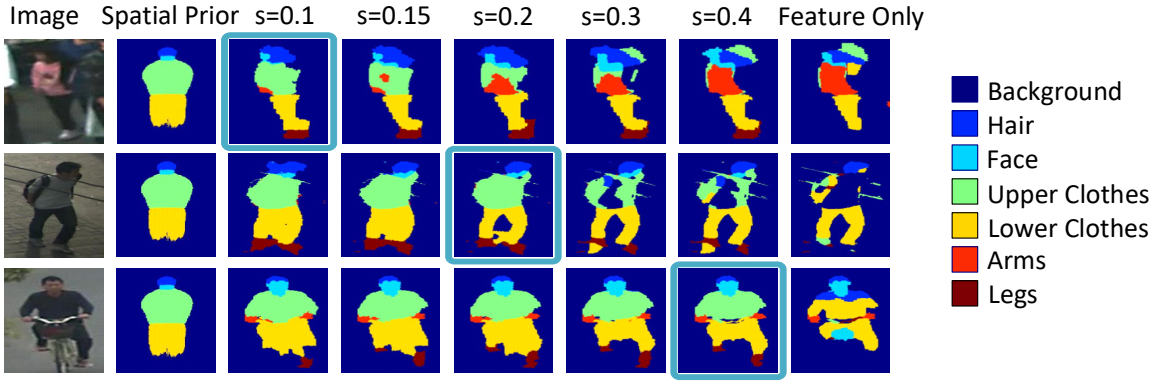


Figure 3.10: Influences of the neighborhood size of local learning s on parsing results. The "location prior" refers to $p(l|x, y)$ followed by a quantization using maximum a posterior (MAP) rule, and "feature only" refers to $p(l|\mathbf{f})$ followed by MRF smoothing, respectively.

f + SVM	(f, x, y) + SVM	(f, x, y) + Boosting	Product of Expert	Ours
34.2	45.8	50.7	46.6	**55.3**

Table 3.5: Results of Penn-Fudan dataset when parsing pedestrian bounding boxes obtained by poselet detection.

poselet detector [142], instead of using the ground-truth bounding box. Because only Penn-Fudan dataset provides full-sized images suitable for detection task, we only use this dataset for evaluation. The detected bounding boxes are used for both training and testing. After performing the detection we find that the intersection over union scores (IOU scores) between the detected and the ground-truth bounding boxes are mostly around 0.7, which shows a sufficient amount of misalignments.

Table 3.4 shows that our method still performed significantly better than the alternative approaches that rely on one unified pixel classifier. From Figure 3.11 we see that the parsing results are mostly satisfactory, except for the cases of serious misalignments and presence of background clutter (the last row images). It demonstrates that our method can still work with image misalignments.

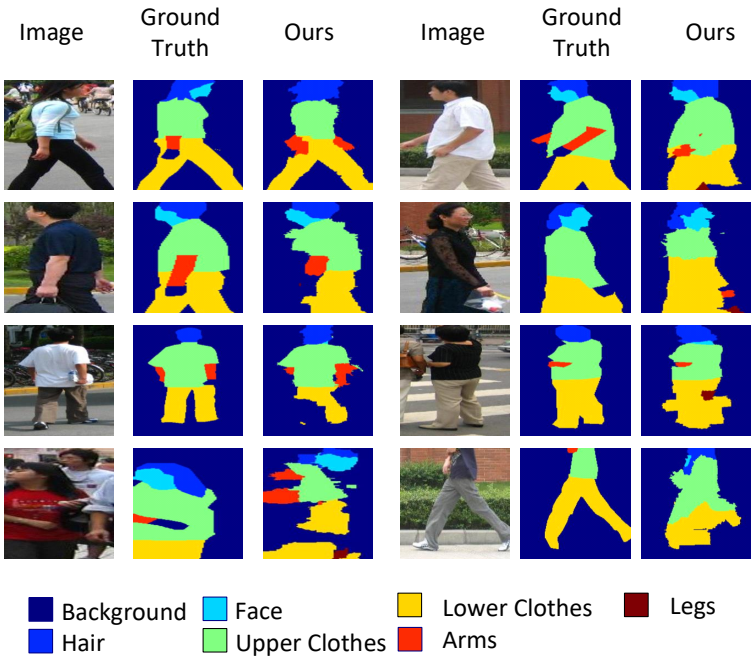


Figure 3.11: Image parsing results of Penn-Fudan dataset with the detected bounding boxes.

Our Method	Overall	Walking	Riding
Without Pose Grouping	53.3	54.8	47.8
With Pose Grouping	**54.2**	**55.0**	**50.7**

Table 3.6: Results of PPSS dataset with pose grouping. The 1^{st} row shows the results of parsing all the images together. The 2^{nd} row shows the results of parsing different groups of images individually using separate classifiers.

3.4.3.5 Handling Pose Variation.

PPSS dataset contains a large amount of pose variations such as walking and riding vehicles. To better handle such variations we manually separate the training images into two groups, such that within each group the location prior is strengthened. The walking and riding groups contain 3084 and 877 images, respectively. Then we train pixel classifiers and perform labeling on each group separately. This divide and conquer strategy has also been used by Huang et.al [143] for scene parsing.

Table 3.6 shows that overall the improvements from such a grouping process is

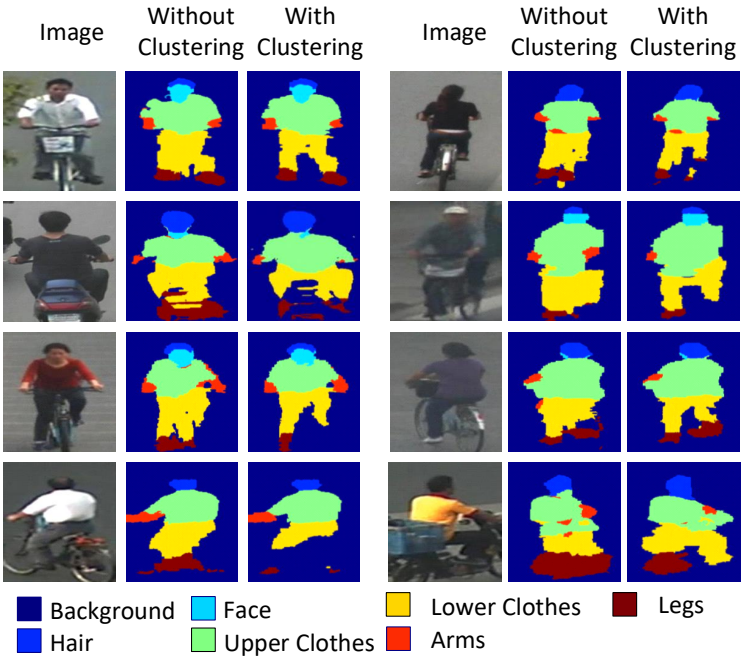


Figure 3.12: Image parsing results of people riding vehicles from the PPSS dataset. We see significantly improved visual results in the 3^{rd} and 6^{th} column especially in the lower-clothes and legs, as the classifiers are trained specifically for this group of images.

0.9% and riding has the largest improvement of 2.9%. As riding has a distinctly different prior map, and our method is dependent on the prior map, training specialized classifies for this group will benefit our method the most. Such an advantage is also illustrated in Figure 3.12. This experiment demonstrates that incorporating pose information can further enhance our performance. We will consider the extension of joint segmentation and pose estimation [31, 144] as our future work.

3.4.4 Results on Street-View Scene Parsing

3.4.4.1 Dataset Description and Experimental Protocols

Our third experiment is performed on Cam-Video dataset [113], a dataset of images from driving cars captured at both daylight and dusk. We follow the protocol of previous work [113] and use 11 semantic classes. The training set consists of 468

	Texton-based Descriptor	Deep-learning-based Feature
f + SVM	82.0 (41.0)	91.2 (65.7)
(**f**,x,y) + SVM	83.0 (43.2)	91.2 (65.7)
(**f**,x,y) + Boosting	82.2 (44.7)	90.8 (65.3)
Product of Expert	83.5 (42.7)	91.2 (65.7)
Ours (nearest)	83.8 (46.5)	91.2 (65.8)
Ours (average)	**85.9** (**48.8**)	**91.3** (**66.6**)

Table 3.7: Comparisons with alternative ways of fusing $\mathbf{f}$ and (x, y) for Cam-Video Dataset. Numbers outside and within brackets are per-pixel and per-class accuracies, respectively.

images and the testing set consists of 232 images. The dataset is available from `http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/`.

By populating the object sizes from the largest to the smallest, we note that the distribution of objects in images is heavy-tailed. In total 75% of pixels come from majority classes and the remaining 25% of pixels belong to minority classes. Based on this observation and previous works, two types of accuracy measures are used for different purposes. Per-pixel accuracy measures the pixel classification rate. Per-class accuracy measures the average IOU score over all the semantic classes. Compared with per-pixel accuracy which emphasizes on the correctness of the majority classes, per-class accuracy balances majority classes and minority classes.

We test our method with both texton-based descriptor and deep-learning-based feature. To obtain the deep-learning-based feature, we extract from "fc7" layer of dilated network [2] which contains 4096 feature maps resized to the input image size. For each superpixel, we calculate the average of pixel values within the same superpixel for every feature map, and obtain a 4096-dimensional feature vector. To further reduce computational cost of training phase, we reduce it to 1024 dimension by SVM feature ranking [145].

3.4.4.2 Main Results

Table 3.7 shows that when using the texton-based descriptor, our performance is superior to all the alternative methods by a large margin. However with deep-learning-based feature, the performance improvement is smaller. As stacked convolution layers from the dilated network [2] have a large receptive field, the extracted deep-learning-based feature has implicitly encoded the long-range contextual information and the spatial layout. Therefore, our technique is comparably less effective in further improving the performance with deep-learning-based feature.

Table 3.8 shows that our method has comparable or better performance than existing works [2, 13, 51, 146–151] for both types of features. The deep-learning approaches have significantly outperformed the traditional methods, thanks to the various techniques such as new network architecture [51,150], dilated convolution [2, 149], and refinement structure [151]. By fusing location information, our method slightly improves the performance with deep-learning based features from [2]. Our results might be further enhanced by using features from more recent deep-learning network [150], by performing post-processing with a more sophisticated graphical model [18,149], and by implementing the method in an end-to-end deep network [17].

Figure 3.13 presents examples of our parsing results with deep-learning-based feature. We see in most cases at both daylight and dusk, our parsing quality is quite good and is able to identify a number of details, such as poles, pedestrians, and cars at different distances. The errors arise mainly from the confusions between road and sidewalk, tree and building, because of their similar color and texture. They also come from our method's bias to label larger pixel areas for thin structure such as poles, potentially due to the use of superpixels. We notice that some worse-performing categories, such as sign, pole, bicyclist and pedestrian, may be further improved by using an object detector [132].

	Building	Tree	Sky	Car	Sign	Road	Pedestrian	Fence	Pole	Sidewalk	Bicyclist	Per-Pixel	Per-Class
Comparisons with traditional approaches													
[13]	70.4	54.8	83.5	43.3	**25.4**	83.4	11.6	18.3	5.2	57.4	8.9	83.3	42.0
[146]	66.8	**66.6**	**90.1**	62.9	21.4	85.8	**28.0**	17.8	8.3	63.5	8.5	82.8	47.2
[147]	-											82.1	43.3
[148]	-											84.7	**48.8**
Ours (Texton Features)	**73.9**	62.9	89.7	**66.0**	5.5	**87.7**	25.0	**28.0**	**9.7**	**70.2**	**17.7**	**85.9**	**48.8**
Comparisons with deep-learning approaches													
[51]	68.7	52.0	87.0	58.5	13.4	86.2	25.3	17.9	16.0	60.5	24.8	82.8	46.4
[149]	81.5	74.6	89.0	82.2	42.3	**92.2**	48.4	27.2	14.3	75.4	50.1	-	61.6
[2]	82.6	76.2	89.9	84.0	**46.9**	**92.2**	**56.3**	35.8	23.4	75.3	55.5	-	65.3
Ours (Features from [2])	**85.7**	**78.0**	**90.5**	**87.6**	30.0	91.8	54.3	**51.2**	**24.0**	**78.8**	**60.8**	**91.3**	**66.6**
Comparisons with recent works													
[150]	**83.0**	**77.3**	**93.0**	77.3	**43.9**	**94.5**	**59.6**	37.1	**37.8**	82.2	50.5	91.5	66.9
[151]	82.5	76.8	92.1	**81.8**	43.0	**94.5**	54.6	**47.1**	33.4	**82.3**	**59.4**	-	**68.0**

Table 3.8: Comparisons with existing methods on the Cam-Video Dataset. The bold number refers to the score of the top-performing method of each column. The performance of some recent methods published after our work [7, 8] are also displayed.

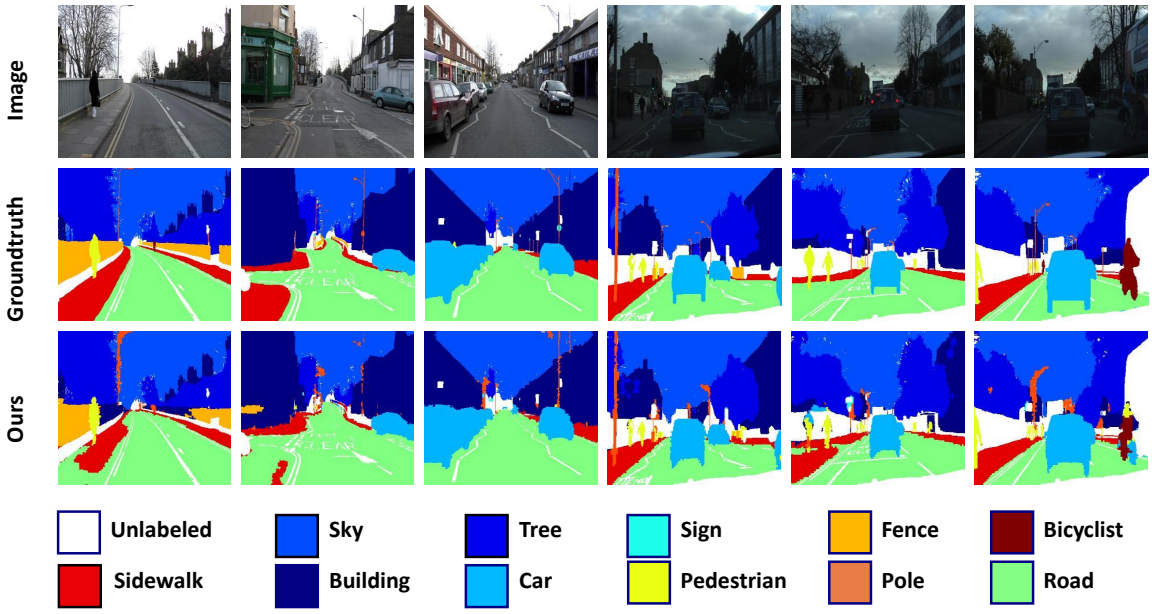


Figure 3.13: Image parsing results on Cam-Video Dataset using deep-learning-based feature.

3.4.5 Computational Complexity

To understand the computational complexity of our method, we first explain implementation details of the field of local classifiers. As discussed in Section 3.2.3 of the thesis, for the sake of computationally efficiency, we apply local classifiers coarsely on a uniform grid. The grid spacing is set to $\lambda \times s \times (W, H)$, where s is the neighborhood size ranging from 0 to 1, W and H is the image width and height. λ decides the grid density and is set to 0.2 in our experiments. We can derive that under such classifier grid settings the total testing time of all the local classifiers is of $O((\frac{1}{\lambda})^2 \times N \times T)$, where N refers to the number of testing samples (superpixels) in an image and T represents the computational time for a single sample(superpixel). In other words, the computational complexity is dependent on classifier grid density controlled by λ; and it is not directly dependent on the total number of local classifiers.

We implement the method on a quad-core i7 machine, with Matlab-C hybrid programming and parallel processing enabled. During the testing, for each image,

superpixel extraction and feature extraction take around 3.5 seconds in total. If applying linear SVM as the classification algorithm, a single global pixel classifier takes around 0.1 seconds, while a field of local classifiers takes around 1.6 seconds to process each image. If applying joint boosting, a single global pixel classifier and a field of local classifiers take around 1.6 and 2.0 seconds respectively. Post-processing steps using a MRF take another 0.5 seconds.

3.4.6 Limitations of the Method

Our method divides the problem of parsing the entire image into subproblems of parsing local image neighborhoods, thus may simplify the learning. For example, when the images are well aligned, inside a local image neighborhood there may be fewer categories to classify thus learning is easier. On the other hand, each classifier only gets a subset of data from the local neighborhood, so the distribution of training may no longer correspond to the testing if the images are poorly aligned. This may lead to worse generalization. As a result, the benefits of our method depend critically on the degree of misalignments. By reviewing previous works, we note that such local learning strategy is only used on other applications with roughly aligned images, such as in video parsing [66, 67], medical segmentation [65], face alignment [152] and object detection for static camera [153]. For unaligned scenarios, a unified pixel classifier is used instead such as in general image parsing [11, 13]. For example, in [11] appearance model $p(l \mid \mathbf{f})$ is learned using one unified pixel classifier which is insensitive to misalignments.

Based on the above discussion and experimental results, we identify the following two limitations of the proposed method: (1) Our method is less effective in fusing appearance and location information for unaligned scenarios. In those cases, to tolerant the spatial misalignments, we have to select a large neighborhood size to train the local classifiers. If the selected local learning neighborhood size be-

comes too large, each local classifier behaves more like a global one that does not utilize location information. (2) Our method is comparably less useful in further improving the performance with strong image appearance features. As demonstrated in CamVid scene parsing experiments, compared with weaker traditional features, adopting deep-learning features greatly improves the final semantic segmentation performance. However, there is less performance gap between our method and the baseline method which does not rely on location information. We note as the deep-learning network [154] has a large receptive field, the extracted deep-learning-based feature has implicitly encoded the long-range contextual information and the spatial layout. Therefore, our technique is comparably less effective to enhance the performance with such deep-learning feature.

In spite of these weaknesses, our algorithm has been demonstrated to be effective when images are well aligned and image appearance features are weak. For example, Fig 3.7 shows that the proposed algorithm is quite useful for pedestrian parsing within ground-truth bounding boxes in low-quality surveillance images. As a future work, we can implement the method in an end-to-end deep network [17] to study whether deep network can learn features more invariant to spatial misalignments when combined with our approach.

3.5 Conclusions

This chapter proposes a technique of fusing location and appearance information for pixel labeling by local learning. Each local classifier is trained with pixels from its neighborhood region only thus better fits the local distribution and can be more discriminative. We analyze the bias-variance trade-off of our proposed local classifier, and indicate the importance of selecting an appropriate neighborhood size to train the local classifier such that it can tolerate the spatial misalignments. Our local

learning scheme can accommodate any pixel classifier with arbitrary features and is also compatible to any graphic model such as MRF or CRF. In experiments we compare our method with alternative ways of fusion of feature $\mathbf{f}$ and location (x, y) to build a unified pixel classifier. The results validate that when the images are roughly aligned, our proposed local classifier can be more effective than alternative ways that rely on one unified classifier. On three different tasks of horse segmentation, pedestrian parsing, and scene parsing, our local learning can significantly improve the performance compared with existing methods.

3.6 Appendix

3.6.1 Proof of Theorem 3.1

For notation simplicity we denote $\mathcal{N}(x, y, s_1)$ as $\mathcal{N}_1(x, y)$ or $\mathcal{N}_1$, and $\mathcal{N}(x, y, s_2)$ as $\mathcal{N}_2(x, y)$ or $\mathcal{N}_2$. Because $s_2 = k \times s_1$, for $k \in \mathbb{N}$ being a natural number, $\mathcal{N}_2$ can be considered as a combination of many identical copies of $\mathcal{N}_1$ with different central locations. In other words, $\mathcal{N}_2 = \mathcal{N}_{11} \cup ... \cup \mathcal{N}_{1t}$, where $\mathcal{N}_{1i} = \mathcal{N}(x + \delta_{xi}, y + \delta_{yi}, s_1)$. We also define $\mathcal{N}_{10} = \mathcal{N}_1$.

From uniform distribution of the pixel locations, we have: $\alpha_1 = \sum_{(x',y')\in\mathcal{N}_1} p(x', y') = \frac{s_1 \times s_1}{W \times H}$, $\alpha_2 = \sum_{(x',y')\in\mathcal{N}_2} p(x', y') = \frac{s_2 \times s_2}{W \times H}$, and $\alpha_2 = t \times \alpha_1$.

In addition, we have the following definitions:

$$\begin{aligned} &\underbrace{\mathbb{E}\Big[d_{KL}\Big(p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f}) \;||\; \mathbb{E}\big[p_{\mathcal{N}(x,y,s)}(l \mid \mathbf{f})\big]\Big)\Big]}_{var(p_{\mathcal{N}(x,y,s)}(l|\mathbf{f}))} \\ =&\mathbb{E}\Big[\sum_{l,\mathbf{f}} p_{\mathcal{N}}(l, \mathbf{f}) \log \frac{p_{\mathcal{N}}(l, \mathbf{f})}{\mathbb{E}[p_{\mathcal{N}}(l \mid \mathbf{f})]}\Big]. \end{aligned} \tag{3.21}$$

$$f(x,y,i) = \sum_{(x',y')\in\mathcal{N}_{1i}(x,y)} p(l,\mathbf{f},x',y') \times \log \frac{\sum_{(x',y')\in\mathcal{N}_{1i}(x,y)} p(l,\mathbf{f},x',y')}{\sum_{(x',y')\in\mathcal{N}_{1i}(x,y)} \mathbb{E}[p(l,\mathbf{f},x',y')]}. \tag{3.22}$$

$$\begin{aligned}
&var(s_2)\\
=&\frac{1}{|\mathcal{I}|}\sum_{(x,y)\in\mathcal{I}} var\Big(p_{\mathcal{N}_2}(l\mid\mathbf{f})\Big) && (3.23)\\
=&\frac{1}{|\mathcal{I}|}\sum_{(x,y)\in\mathcal{I}} \mathbb{E}\Big[\sum_{l,\mathbf{f}} p_{\mathcal{N}_2}(l,\mathbf{f})\log\frac{p_{\mathcal{N}_2}(l\mid\mathbf{f})}{\mathbb{E}[p_{\mathcal{N}_2}(l\mid\mathbf{f})]}\Big] && (3.24)\\
=&\frac{1}{|\mathcal{I}|}\sum_{(x,y)\in\mathcal{I}} \mathbb{E}\Big[\sum_{l,\mathbf{f}}\frac{1}{\alpha_2}\sum_{(x',y')\in\mathcal{N}_2} p(l,\mathbf{f},x',y') \times\log\frac{\sum_{(x',y')\in\mathcal{N}_2} p(l,\mathbf{f},x',y')}{\sum_{(x',y')\in\mathcal{N}_2}\mathbb{E}[p(l,\mathbf{f},x',y')]}\Big] && (3.25)\\
=&\frac{1}{|\mathcal{I}|}\sum_{(x,y)\in\mathcal{I}} \mathbb{E}\Big[\sum_{l,\mathbf{f}}\frac{1}{\alpha_2}\sum_{i=1}^{t}\sum_{(x',y')\in\mathcal{N}_{1i}} p(l,\mathbf{f},x',y') \times\log\frac{\sum_{i=1}^{t}\sum_{(x',y')\in\mathcal{N}_{1i}} p(l,\mathbf{f},x',y')}{\sum_{i=1}^{t}\sum_{(x',y')\in\mathcal{N}_{1i}}\mathbb{E}[p(l,\mathbf{f},x',y')]}\Big] && (3.26)\\
\leq&\frac{1}{|\mathcal{I}|}\sum_{(x,y)\in\mathcal{I}} \mathbb{E}\Big[\sum_{l,\mathbf{f}}\frac{1}{\alpha_2}\sum_{i=1}^{t}\Big(\sum_{(x',y')\in\mathcal{N}_{1i}} p(l,\mathbf{f},x',y') \times\log\frac{\sum_{(x',y')\in\mathcal{N}_{1i}} p(l,\mathbf{f},x',y')}{\sum_{(x',y')\in\mathcal{N}_{1i}}\mathbb{E}[p(l,\mathbf{f},x',y')]}\Big)\Big] && (3.27)\\
=&\frac{1}{|\mathcal{I}|}\sum_{(x,y)\in\mathcal{I}} \mathbb{E}\Big[\sum_{l,\mathbf{f}}\frac{1}{\alpha_2}\sum_{i=1}^{t} f(x,y,i)\Big] && (3.28)\\
=&\frac{1}{|\mathcal{I}|}\sum_{i=1}^{t}\mathbb{E}\Big[\sum_{l,\mathbf{f}}\frac{1}{\alpha_2}\sum_{(x,y)\in\mathcal{I}} f(x,y,i)\Big]. && (3.29)
\end{aligned}$$

By the definition of $var(s)$ from Equation 3.18, we obtain Equation 3.23. After applying the definition of KL-divergence (Equation 3.21) to Equation 3.23, Equation

3.24 can be derived. Furthermore from the definition of our local classifier by Equation 3.1, the alignment assumption by Equation 3.19, and Equation 3.24, Equation 3.25 holds. Equation 3.26 holds because of Equation 3.25 and $N_1 = \{N_{21} \cup ... \cup N_{2t}\}$. Equation 3.27 holds because of Equation 3.26 and Log-Sum Inequality [155]. And finally, Equation 3.28 holds by substituting Equation 3.22 into Equation 3.27.

The values $f(x, y, i)$ where $i = 1, ...t$, do not equal to each other. The difference is due to different location offsets $(\delta_{xi}, \delta_{yi})$ of these patches $\mathcal{N}_{1i}(x, y) = \mathcal{N}(x + \delta_{xi}, y + \delta_{yi}, s_1)$. However by averaging over all the image locations $(x, y) \in \mathcal{I}$, such a difference disappears except at the image boundary. In other words, if we assume images size is much larger than the neighborhood size of local learning, in order to ignore the complications on the image boundary, we have:

$$\begin{aligned}
&\sum_{(x,y)\in\mathcal{I}} f(x, y, i) = \sum_{(x,y)\in\mathcal{I}} f(x, y, 0) \\
&= \sum_{(x,y)\in\mathcal{I}} \Big(\sum_{(x',y')\in\mathcal{N}_1(x,y)} p(l, \mathbf{f}, x', y') \times \log \frac{\sum_{(x',y')\in\mathcal{N}_1(x,y)} p(l, \mathbf{f}, x', y')}{\sum_{(x',y')\in\mathcal{N}_1(x,y)} \mathbb{E}[p(l, \mathbf{f}, x', y')]} \Big), \qquad (3.30)
\end{aligned}$$

for all $i = 1, ..., t$.

From Equation 3.29 and 3.30, we have:

$$\begin{aligned}
&var(s_2) \\
\leq &\frac{1}{|\mathcal{I}|} \sum_{i=1}^{t} \mathbb{E}\Big[\sum_{l,\mathbf{f}} \frac{1}{\alpha_2} \sum_{(x,y)\in\mathcal{I}} \Big(\sum_{(x',y')\in\mathcal{N}_1(x,y)} p(l, \mathbf{f}, x', y') \\
&\times \log \frac{\sum_{(x',y')\in\mathcal{N}_1(x,y)} p(l, \mathbf{f}, x', y')}{\sum_{(x',y')\in\mathcal{N}_1(x,y)} \mathbb{E}[p(l, \mathbf{f}, x', y')]} \Big) \Big] \qquad (3.31) \\
= &\frac{1}{|\mathcal{I}|} \mathbb{E}\Big[\sum_{l,\mathbf{f}} \frac{1}{\alpha_2} \times t \sum_{(x,y)\in\mathcal{I}} \Big(\sum_{(x',y')\in\mathcal{N}_1(x,y)} p(l, \mathbf{f}, x', y') \\
&\times \log \frac{\sum_{(x',y')\in\mathcal{N}_1(x,y)} p(l, \mathbf{f}, x', y')}{\sum_{(x',y')\in\mathcal{N}_1(x,y)} \mathbb{E}[p(l, \mathbf{f}, x', y')]} \Big) \Big] \qquad (3.32) \\
= &\frac{1}{|\mathcal{I}|} \sum_{(x,y)\in\mathcal{I}} \mathbb{E}\Big[\sum_{l,\mathbf{f}} \frac{1}{\alpha_1} \sum_{(x',y')\in\mathcal{N}_1(x,y)} p(l, \mathbf{f}, x', y')
\end{aligned}$$

$$\times \log \frac{\sum_{(x',y')\in\mathcal{N}_1(x,y)} p(l,\mathbf{f},x',y')}{\sum_{(x',y')\in\mathcal{N}_1(x,y)} \mathbb{E}[p(l,\mathbf{f},x',y')]}\Bigg] \tag{3.33}$$

$$= var(s_1) \tag{3.34}$$

The last equation holds by substitutions and algebraic manipulations, from previous definitions and assumptions by Equation 3.33, 3.18, 3.21, 3.1 and 3.19.

3.6.2 Explanation of the Weighting Factor in Equation 3.9

From Equation 3.6 and Equation 3.7, we see the weighting factor $w(x',y',x,y)$ is proportional to the number of local classifier patches covering both the sampling position (x',y') and the testing position (x,y). In other words, $w(x',y',x,y)$ is proportional to cardinality of the set $\Omega_0 = \{\mathcal{N}(x_k,y_k,s)|(x',y') \in \mathcal{N}(x_k,y_k,s),(x,y) \in \mathcal{N}(x_k,y_k,s)\}$. We assume that $|x'-x| \leqslant s \times W$ and $|y'-y| \leqslant s \times H$; so there will be at least one patch covering both pixels. When we shift that patch along x or y direction, the patch may still cover both pixels. By assuming the patches are sampled uniformly in image, the number of patches covering both the sampling position (x',y') and the testing position (x,y) will be proportional to the maximum distance for shifting along x or y directions. It is obvious that the maximum distance for shifting along x direction will be $s \times W - |x-x'|$ and along y direction will be $s \times H - |y-y'|$, where $s \times W$ and $s \times H$ are width and height of the patch, respectively. So the weighting factor will have the following form:

$$w(x',y',x,y) \propto (s \times W - |x-x'|)(s \times H - |y-y'|) \tag{3.35}$$

3.6.3 Parameter Settings

All the parameter settings, such as neighborhood size of local pixel classifiers, SVM regularization parameter, number of boosting stages for joint-boosting classifier,

	Type	Neighborhood Size of Local Classifiers
Table 3.1 and 3.2	linear SVM	0.2
Table 3.3 and 3.4	linear SVM	0.3 (penn-fudan dataset) or 0.2(ppss dataset)
Table 3.5	linear SVM	0.2
Table 3.6	linear SVM	0.2
Table 3.7 and 3.8	joint boosting	0.6

Table 3.9: Local pixel classifier parameters used in experiments.

as well as MRF pairwise term weighting factor are obtained on a validation set. We summarize the parameters that are most relevant to our approach, such as neighborhood size and type of local pixel classifiers in Table 3.9.

Chapter 4

Exploiting Spatio-Temporal Consistency for Efficient Pixel Classification Maps Smoothing

4.1 Introduction

For many applications of video analytics, it is of great interest to analyze the video stream and generate a sequence of per-pixel classification maps that can capture the object, *e.g.*, saliency maps, or understand the scene, *e.g.*, scene parsing maps. Despite the previous success, pixel classification in a video stream remains a challenging problem due to the spatio-temporal structure among the pixels and the real-time constraint to analyze the video stream. In practice, when each video frame is analyzed or parsed independently, the per-pixel classification maps are usually "flickering" due to the spatio-temporal inconsistencies and noisy classifications, *e.g.*, caused by object and camera movements or low quality videos. Thus an efficient online smoothing of the pixel classification maps is important to improve the video analytics performance and can benefit many streaming video analytics applications.

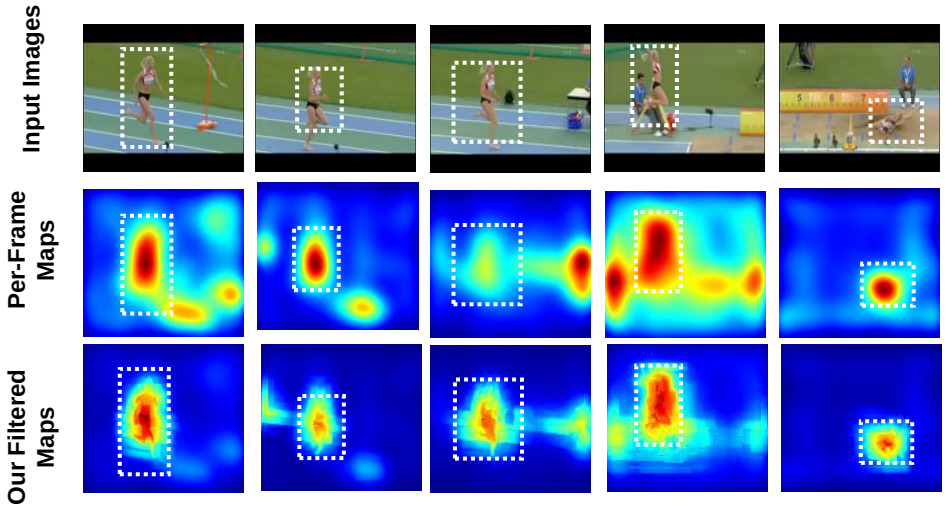


Figure 4.1: Adaptive Exponential Smoothing (AES) to reduce the "flicking" effect of pixel classification map in video streams. The 1^{st} Row: input video. The 2^{nd} Row: per-frame classification maps. The 3^{rd} Row: refined maps by our proposed AES.

Enforcing spatio-temporal smoothness constraints over the pixel classification maps is a widely used technique [10, 86, 87, 96, 110] to improve the quality of classification maps by reducing the "flickering" effect. However, existing methods still have difficulty in providing an efficient and effective solution. On the one hand, despite a lot of previous work [106, 107] on video content processing and denoising, they are originally designed to improve the video quality rather than the pixel classification maps. It is worth noting that linear spatio-temporal smoothing methods such as moving average or exponential filter can only work well for additive noises that are independent of the video signal. They may not provide satisfactory results on the pixel classification maps, which are usually affected by non-additive and signal dependent noises. Thus, special spatio-temporal smoothing methods are required to deal with such noises. On the other hand, although a few spatio-temporal smoothing methods have been proposed to refine pixel classification maps, most of them only perform in an offline or batch mode, where the whole video is required to perform the smoothing [76, 156–158]. Although a few recent works have been developed

for online smoothing of video classification maps, they usually rely on extra steps, such as producing temporally consistent superpixels from a streaming video [96], or leveraging metric learning and optical flow [10]. They are difficult to implement in real-time, and will not work well when the estimated optical flow is noisy.

To address the above limitations, in this chapter we propose an efficient online smoothing method which is able to perform online and real-time filtering of video classification maps. Given a sequence of pixel classification maps, in which each pixel is associated with a detection score or a probabilistic multi-class distribution, our goal is to provide a causal filter that can improve the spatio-temporal smoothness of the pixel classification maps by reducing "flickering" effects. Our method is motivated by the classic exponential smoothing, as we also apply exponentially decreasing weights over time to smooth the classification score of each pixel. However, instead of fixing the pixel location to perform temporal smoothing, for each pixel, we first find a spatio-temporal smoothing path along which the pixels are likely to belong to the same category. In this way, the smoothing will be less likely to blend the classification scores of different classes. Exponential smoothing is then performed along this found path, as illustrated in Fig. 4.2. To find the path for each pixel, we rely on both the pixel appearance and pixel classification scores along the path. If the path consists of pixels of the same category, it should be composed by pixels of similar appearance, and the accumulated classification score along the path should be much higher than an arbitrary path. As a result, for each pixel, the path of highest accumulated classification score is selected to perform exponential smoothing. As different pixels will have different paths, we call our smoothing method adaptive exponential smoothing (AES). Similar to exponential smoothing, our method can correct individual false alarms, *i.e.*, pixels with high classification scores but low accumulated scores along the path, as well as missing detections, *i.e.*, pixels with low classification scores but high accumulated scores along the path.

Meanwhile, thanks to the pixel tracing, our method can better address object or camera movements when performing spatio-temporal smoothing.

One limitation of AES, however, is its heavy computational cost of finding the smoothing path for each pixel. For efficient online implementation, instead of performing pixel tracing for each individual pixel separately, we propose a dynamic programming algorithm that can trace all pixels simultaneously. It guarantees to obtain the optimal smoothing path, *i.e.*, the path of the highest accumulated score, for all pixels. The proposed implementation of AES is of linear complexity and only needs to keep the most recent pixel classification map for online smoothing. As a result, it can bring real-time online smoothing of pixel classification maps. For a video stream of frame size 320×240, our implementation can achieve 50 frames per second without code optimization.

To evaluate the performance of our proposed method, we perform two different streaming video analytics tasks, *i.e.*, online saliency map filtering and online multi-class scene parsing map filtering. For saliency map filtering, we test the performance on 24 categories of the UCF 101 dataset. While for video scene parsing, we evaluate the performance on four benchmark video sequences, and different image parsing methods are applied to test the generalization ability of our smoothing method. The comparisons with average and exponential filters, as well as more sophisticated approaches that rely on optical flow computations, validate the effectiveness and efficiency of the proposed spatio-temporal smoothing method, *i.e.*, AES, for streaming video analytics.

4.2 Proposed Method

We denote a video sequence as $\mathcal{S} = \{\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_T\}$, where $\mathcal{I}_k$ is a $W \times H$ image frame. For each spatio-temporal location (x, y, t), we assume that a classification

score $U(x, y, t)$ is provided by an independent image analysis module. As the pixel classification scores are generated independently for each frame, they do not necessarily enforce the temporal consistency across frames. So smoothing is needed to refine the pixel classification maps. We firstly explain two classic linear filters below:

Moving Average (Ave) [159] :

$$M(x, y, t) = \frac{1}{\delta_T + 1} \sum_{i=t-\delta_T}^{t} U(x, y, i). \tag{4.1}$$

Exponential Smoothing (Exp) [78]:

$$\begin{aligned} & M(x, y, t) = \\ & \alpha \times M(x, y, t-1) + (1-\alpha) \times U(x, y, t) \\ =& \alpha^{(t-1)} U(x, y, 1) + (1-\alpha) \sum_{i=2}^{t} \alpha^{(t-i)} U(x, y, i) \\ \approx& (1-\alpha) \sum_{i=1}^{t} \alpha^{(t-i)} U(x, y, i). \end{aligned} \tag{4.2}$$

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. 4.2 decays exponentially with respect to t. Unlike moving average which assigns equal weights for input scores within a temporal window, exponential filter 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. 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

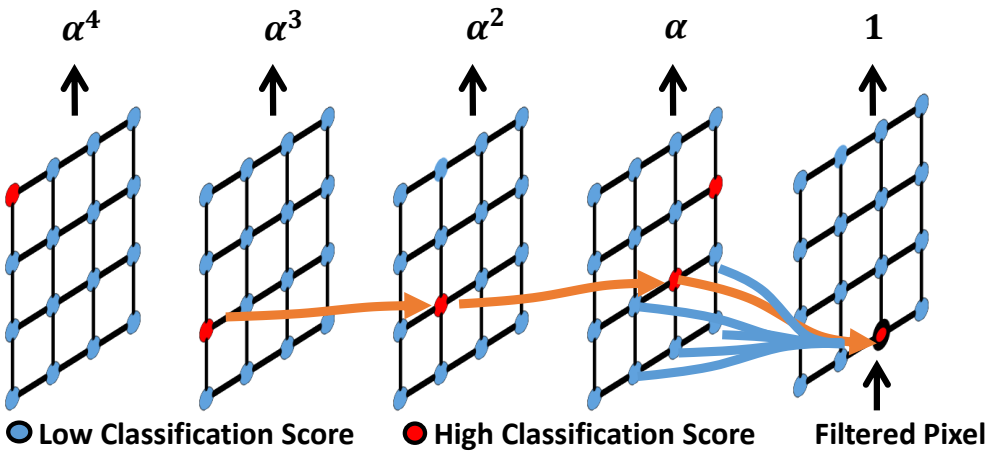


Figure 4.2: An illustration of our adaptive exponential smoothing mechanism. It shows that using weighted accumulation score, pixel tracing from previous frame can help to strengthen true positives and suppress false positives. $\alpha \in [0, 1]$ refers to the temporal weighting factor.

motivates us to propose an adaptive smoothing method that is pixel dependent.

4.2.1 Adaptive Exponential Smoothing (AES)

We assume each spatio-temporal location $v_t = (x, y, t)$ is associated with a discriminative classification 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, and a low negative score indicates a low likelihood. To better explain the proposed AES, we represent the video as a 3-dimensional $W \times H \times T$ trellis denoted by $\mathcal{G}$. For each pixel $v = v_t$, we trace it in the past frames to obtain a path $\mathcal{P}_{s\to t}(v_t) = \{v_i\}_{i=s}^{t}$ in $\mathcal{G}$. Here i is the frame index, and v_i is a pixel at frame i. The path $\mathcal{P}_{s\to t}(v_t)$ 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 classification 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 $\mathcal{P}_{s\to t}(v_t)$:

$$M(\mathcal{P}_{s\to t}(v_t)) = \sum_{i=s}^{t} \alpha^{(t-i)} U(v_i), \tag{4.3}$$

where $\alpha \in [0, 1]$ is the temporal weighting factor. 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 $M(\mathcal{P}_{s\to t}(v_t))$ 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 smoothing problem as finding the path that maximizes the exponential smoothing score:

$$\mathcal{P}^*_{s\to t}(v_t) = \underset{\mathcal{P}_{s\to t}(v_t)\in \text{path}(\mathcal{G}, v_t)}{\arg\max} M(\mathcal{P}_{s\to t}(v_t)), \tag{4.4}$$

where $\mathcal{P}^*_{s\to t}(v_t)$ 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.*, $M(x, y, t) = M(P^*_{s\to t}(v_t))$. An illustration of our method can be seen in Fig. 4.2. A pixel with high isolated positive score but low exponential smoothing score, *i.e.*, $U(v_t)$ is high but $M(P^*_{s\to t}(v_t))$ 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 $M(P^*_{s\to t}(v_t))$ 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 the longer path, the better. The length of the path, i.e., the temporal smoothing

bandwidth, is content adaptive, which may well address missing detections and false alarms.

4.2.2 Online Pixel Filtering

A brute-force way to solve the pixel-tracing problem defined in Eq. 4.4 is time consuming. Because the starting frame number s of all candidate paths range from 1 to t, the search space is large for even a single pixel location v_t, *i.e.*, $O((2R+1)^{2\times T})$, where T is the video length. To achieve better efficiency, we propose an efficient online smoothing algorithm based on dynamic programming, which can trace all pixels simultaneously in linear complexity. By Eq. 4.3 and Eq. 4.4, the pixel tracing objective can also be written as:

$$M(\mathcal{P}^*_{s\to t}(v_t)) = \max_{1\le s\le t}\left\{\max_{\substack{\forall s\le i\le t-1,\\ (x_i,y_i)\in\mathcal{N}(x_{i+1},y_{i+1})}}\sum_{i=s}^{t-1}\alpha^{(t-i)}U(v_i)\right\}+U(v_t). \tag{4.5}$$

Please note that we abuse the notation such that the inner summation $\sum_{i=s}^{t-1}\cdot$ is defined to be zero when $s=t$. 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 attempt to trace it back to its origin v_s in the past frames, such that the exponential smoothing score along the path $\mathcal{P}^*_{s\to t}(v_t)$ is maximized. Instead of performing back-tracing, our idea is to perform forward-tracing using dynamic programming, so that all pixels are traced simultaneously. It is in spirit similar to the max-path search in [77, 84]. We explain the idea of our dynamic programming using the following lemma which is proved in the appendix.

Lemma 4.1 *$\widehat{M}(v_t)$ in Eq. 4.6 is identical to $M(\mathcal{P}^*_{s\to t}(v_t))$ defined by Eq. 4.5.*

$$\widehat{M}(v_t) = \begin{cases} \max\left\{ \max_{(x_{t-1},y_{t-1})\in\mathcal{N}(x_t,y_t)} \right. \\ \qquad \left. \alpha \times \widehat{M}(v_{t-1}), 0 \right\} + U(v_t), & \text{if } t > 1, \\ U(v_1), & \text{if } t = 1. \end{cases} \tag{4.6}$$

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 propagate it to the current location v_t. Otherwise, a new path is started from v_t because connecting to any neighbor in the previous frame will decrease its score. Based on Lemma 4.1 we implement the algorithm in Algorithm 4.1.

Algorithm 4.1 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 T **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: $\widehat{M}(v_t) \leftarrow \max\left\{ \alpha \times \widehat{M}(v^*_{t-1}), 0 \right\} + U(v_t)$
6: **end for**
7: **end for**

The time complexity of our algorithm is linear in terms of the number of pixels and quadratic with respect to the spatial neighborhood radius, *i.e.*, $O(W \times H \times T \times R^2)$ for the whole video. The slowest part is the step 4 of Algorithm 4.1, i.e., $\max_{(x_{t-1},y_{t-1})\in\mathcal{N}(x_t,y_t)} \widehat{M}(v_{t-1})$, which is essential to perform max filtering on the previous frame exponential smoothing map. To reduce the max filter's quadratic complexity to linear complexity, we perform max filtering on the neighborhood of radius R by repeatedly applying R times the max filter on the neighborhood of radius 1. In this way, the computational cost of our algorithm becomes $O(W \times H \times T \times R)$, and it is identical to the classic linear filters, *e.g.*, temporal moving average or exponential

	Objective	Conditions
Moving Average Filtering [159] (Ave)	$M(x,y,t) = \frac{1}{\delta_T+1}\sum_{i=t-\delta_T}^{t} U(x,y,i)$	$\alpha = 1, R = 0,$ $s = t - \delta_T$
Exponential Filtering [78] (Exp)	$M(x,y,t) \approx (1-\alpha)\sum_{i=1}^{t} \alpha^{(t-i)} U(x,y,i)$	$R = 0, s = 1$
Adaptive Exponential Smoothing (AES)	$M(x,y,t) = M(\mathcal{P}^*_{s\to t}(v_t)) = \sum_{i=s}^{t} \alpha^{(t-i)} U(v_i)$	-

Table 4.1: Relationship between AES and classic linear filters. δ_T and α are the temporal smoothing bandwidth for moving average and the temporal weighting factor for exponential smoothing, respectively. R stands for the spatial neighborhood radius. $\mathcal{P}^*_{s\to t}(v_t)$ refers to the filtering path ending at the current location (x, y, t) which maximizes AES score. s and t stand for the starting and the ending frame of the path, respectively.

filter.

Multi-Class Classification Map Filtering. Our method can be easily extended to filter multi-class pixel classification maps. In this case, each pixel has K classification scores of K classes denoted by $U(v, c)$, where $c = 1, ..., K$ refers to class labels. Correspondingly each pixel has K paths $\mathcal{P}^*_{s\to t}(v_t, c)$ and K path scores $M(\mathcal{P}^*_{s\to t}(v_t, c), c)$, by performing pixel tracing independently for all K classes. The final classification can be determined by a "winner-taking-all" strategy.

$$c^* = \arg\max_{c\in\{1,..K\}} M(\mathcal{P}^*_{s\to t}(v_t, c), c). \tag{4.7}$$

Handling Probabilistic Input. If the image analysis module produces a probabilistic score $p(c \mid v)$ rather than a discriminative score, we convert it into the discriminative score by subtracting a small offset θ : $U(v, c) = p(c \mid v) - \theta$. In our experiments, we empirically [1] fix $\theta = 1/K$ where K is the number of classes.

[1]The offset θ represents a threshold above which the score accumulation takes place. For many applications such as scene parsing, the trained classifier is often biased so their scores may be always low for minority classes, especially when classes are imbalanced and the number of class K is large. In such a case, a small offset θ should be selected to ensure score accumulation happening for the minority classes. In this work, the offset θ is set to $1/K$ empirically. In the future, with training data available, we will consider more sophisticated, class-dependent schemes.

4.2.3 Relationship with Classic Linear Filters

Our proposed method is a generalization of classic smoothing approaches such as moving average and exponential smoothing. In Table 4.1, we show how Eq. 4.3 can be converted into the objective functions of these classic filters. For example, when setting the neighborhood radius $R = 0$, the temporal weighting factor $\alpha = 1$ and fixing the path starting location $s = t - \delta_T$, our method degenerates into the moving average filter. When setting $R = 0$ and fixing $s = 1$, our method approximately becomes the exponential filter. As moving average and exponential smoothing are the special cases, with proper parameter settings, our method is guaranteed to be the same or better than these methods in terms of performance. We will verify this claim from the experimental comparisons.

4.2.4 Appearance Modeling

In Eq. 4.3, we only consider the propagation of classification maps along the pixel dependent path with a fixed temporal weighting factor α. To further leverage image appearance information, we introduce an adaptive temporal weighting factor to selectively propagate classifications based on the appearance similarities of pixels along the path. To be specific, for each spatio-temporal path $\mathcal{P}_{s\to t}(v_t)$, the exponential smoothing score of v_i is propagated to current pixel v_t based on the appearance coherency of the sub-path $\{v_i, ..., v_t\}$. Formally we introduce a new exponential smoothing score with appearance term as follows:

$$M(\mathcal{P}_{s\to t}(v_t)) = \sum_{i=s}^{t} \alpha^{(t-i)} \Big(\prod_{j=i}^{t-1} h(j) \Big) U(v_i), \tag{4.8}$$

where $\prod_{j=i}^{t-1} h(j)$ measures the appearance coherence of the sub-path $\{v_i, ..., v_t\}$. Note that we simplify the notation such that the product $\prod_{j=i}^{t-1} \cdot$ is defined to be one when $i = t$. $h(j)$ is the adaptive temporal weighting factor and is determined by

the pairwise similarity between pixels v_j and v_{j+1}:

$$h(j) = exp(\frac{-|\mathbf{I}_j - \mathbf{I}_{j+1}|^2}{\gamma}), \tag{4.9}$$

where $\mathbf{I}_j$ stands for the LAB color vector of pixel v_j. $\gamma = (\frac{\sum_{(x_{j'},y_{j'})\in\mathcal{N}(x_{j+1},y_{j+1})} |\mathbf{I}_{j'}-\mathbf{I}_{j+1}|^2}{|\mathcal{N}(x_{j+1},y_{j+1})|})$ represents the degree of local color contrast.

Similar to Eq. 4.3, the pixel tracing problem with appearance term is formulated as Eq. 4.10, and solved using the online tracing method stated in Lemma 4.2. The Lemma 4.2 is implemented using an algorithm similar to Algorithm 4.1, except that $\alpha \times h(t-1)$ is used in place of α.

$$M(\mathcal{P}^*_{s\to t}(v_t)) = \max_{1\le s\le t} \left\{ \max_{\substack{\forall s\le i\le t-1,\\ (x_i,y_i)\in\mathcal{N}(x_{i+1},y_{i+1})}} \sum_{i=s}^{t-1} \left(\alpha^{(t-i)} \Big(\prod_{j=i}^{t-1} h(j) \Big) \times U(v_i) \right) \right\} + U(v_t). \tag{4.10}$$

Lemma 4.2 *$\widehat{M}(v_t)$ from Eq. 4.11 is identical to $M(\mathcal{P}^*_{s\to t}(v_t))$ defined by Eq. 4.10, where $h(t-1)$ is defined by Eq. 4.9.*

$$\widehat{M}(v_t) = \begin{cases} \max \left\{ \max_{(x_{t-1},y_{t-1})\in\mathcal{N}(x_t,y_t)} \right. \\ \qquad \left. \alpha \times h(t-1) \times \widehat{M}(v_{t-1}), 0 \right\} + U(v_t), & \text{if } t > 1, \\ U(v_1), & \text{if } t = 1. \end{cases} \tag{4.11}$$

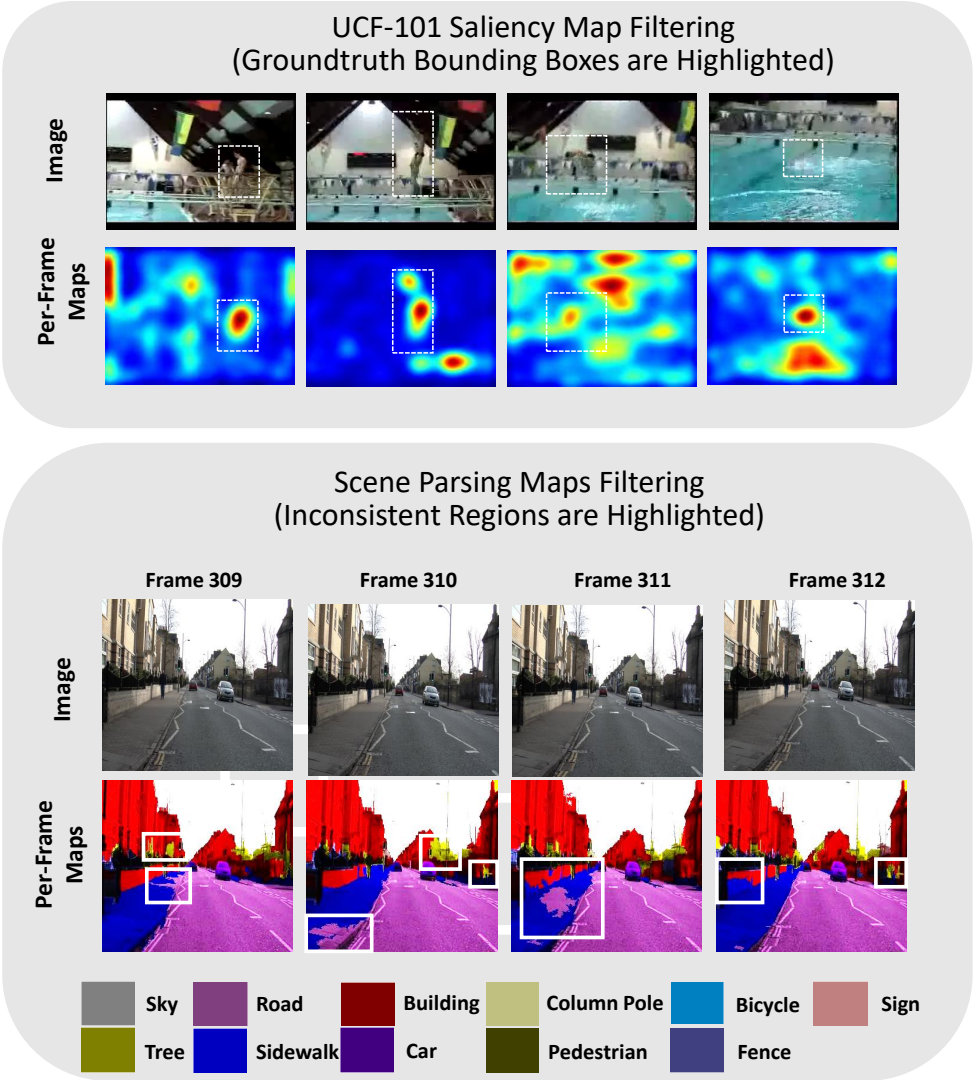


Figure 4.3: Sample images for the benchmark datasets used in this chapter.

4.3 Experiments

4.3.1 Experimental Protocols

We evaluate our approach on two applications: online saliency map filtering (Section 4.3.2) and online scene parsing map filtering (Section 4.3.3.1). Some sample images and initial classification maps for the benchmark datasets are displayed in Fig. 4.3. For all the experiments, we compare our method with classic moving average (Ave) [159] and exponential filter (Exp) [159]. Moreover, we also compare with spatio-temporal versions of these filters:

- Spatio-Temporal Moving Average (ST-Ave) [109]:

$$M(x,y,t) = \frac{1}{\delta_T + 1} \sum_{i=t-\delta_T}^{t} U'(x,y,i), \tag{4.12}$$

  where $U'(x,y,i) = \frac{1}{(2\delta+1)^2} \sum_{\delta x=-\delta}^{\delta x=+\delta} \sum_{\delta y=-\delta}^{\delta y=+\delta} U(x+\delta x, y+\delta y, i)$.

- Spatio-Temporal Exponential Filter (ST-Exp) [78]:

$$M(x,y,t) = \alpha \times M(x,y,t-1) + (1-\alpha) \times U'(x,y,t), \tag{4.13}$$

  where $U'(x,y,t) = \frac{1}{(2\delta+1)^2} \sum_{\delta x=-\delta}^{\delta x=+\delta} \sum_{\delta y=-\delta}^{\delta y=+\delta} U(x+\delta x, y+\delta y, t)$.

To implement the above spatio-temporal filters, we first perform spatial average smoothing of bandwidth δ, and then apply the temporal smoothing as before. We also compare with Yan *et al.* 's method [84] with a similar max-path formulation.

We perform an extensive analysis of scene parsing map filtering experiments, and study appearance modeling and compare with other smoothing methods that are also based on the video appearance: optical flow guided spatio-temporal exponential smoothing and Miksik *et al.* 's method [10]. The optical flow guided spatio-temporal

exponential smoothing is defined below:

$$M(x,y,t) = \alpha \times M(x+u_x, y+u_y, t-1) + (1-\alpha) \times U'(x,y,t), \quad (4.14)$$

where (u_x, u_y) is the flow vector computed using [160], and $U'(x,y,t) = \frac{1}{(2\delta+1)^2} \sum_{\delta x=-\delta}^{\delta x=+\delta} \sum_{\delta y=-\delta}^{\delta y=+\delta} U(x+\delta x, y+\delta y, t)$. Although these methods achieve good performance, they are computationally expensive. In Section 4.3.3.2 we also discuss why our method performs well for certain scene parsing videos in a controlled setting.

4.3.2 Online Filtering of Saliency Map

4.3.2.1 Dataset Description and Experimental Protocols

We use UCF 101 [114] to test the performance of our method of improving video saliency maps. UCF 101 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 25 action categories except for "basketball" as its ground-truth annotation is overly noisy. We randomly pick 50% of the videos for each category and downsample the frames to 160×120 for computational efficiency. In total 1599 video sequences are evaluated. We use the phase discrepancy method [115] to obtain the initial dense saliency map estimations.

F-measure is used to evaluate the saliency map quality. Let S_d and S_g denote the detected pixel-wise saliency map and the ground truth, respectively, the F-measure in percentage can be computed as F-measure $= \frac{2 \times \text{precision} \times \text{recall}}{\text{precision}+\text{recall}}$, where precision $= \frac{Trace(S_d^T S_g)}{\mathbf{1} S_d \mathbf{1}^T}$ and recall $= \frac{Trace(S_d^T S_g)}{\mathbf{1} S_g \mathbf{1}^T}$.

Original	Ave	ST-Ave	Exp	ST-Exp	[84]	Ours
30.6	30.6	30.7	30.6	31.5	31.0	**36.2**

Table 4.2: Saliency map filtering on UCF 101. For our method the spatial search radius is set to 6 while the temporal weighting factor is fixed to 0.8. For each of the baseline methods, the parameters are obtained similarly with a grid search.

4.3.2.2 Comparisons with Classic Linear Filters [109] and Yan *et al.* [84]

The quantitative evaluations are shown in Table 4.2. Some qualitative examples are also shown in Figure 4.4. From the results, we can see that the proposed filtering method can significantly improve the saliency map quality and can better suppress false detections compared with the baseline methods. From Figure 4.5 we observe that in contrast to the baseline methods, our method uses non-straight filtering path, hence can cope with the challenges of fast motions such as jumping and diving in UCF 101 dataset. One limitation of the proposed method is that it dilates the modes of the saliency maps as shown in the 6^{th} column of Figure 4.4. If a more compact map is preferred, we apply the proposed method with appearance modelling as described in Section 4.3.3.4. As shown in the 7^{th} column of Figure 4.4, our method with appearance term can preserve size and shape of the modes from the original saliency map.

4.3.2.3 Parameter Sensitivity Evaluation

We also evaluate the influences 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 4.6. An important 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.

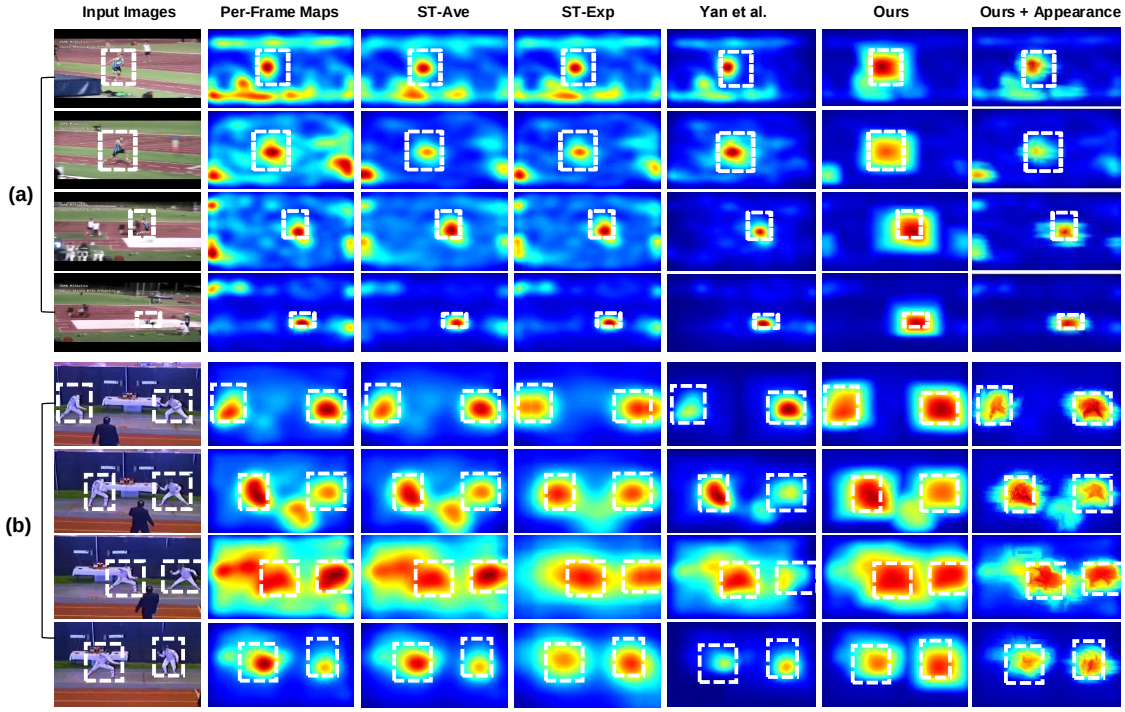


Figure 4.4: Results of saliency map filtering on UCF 101.

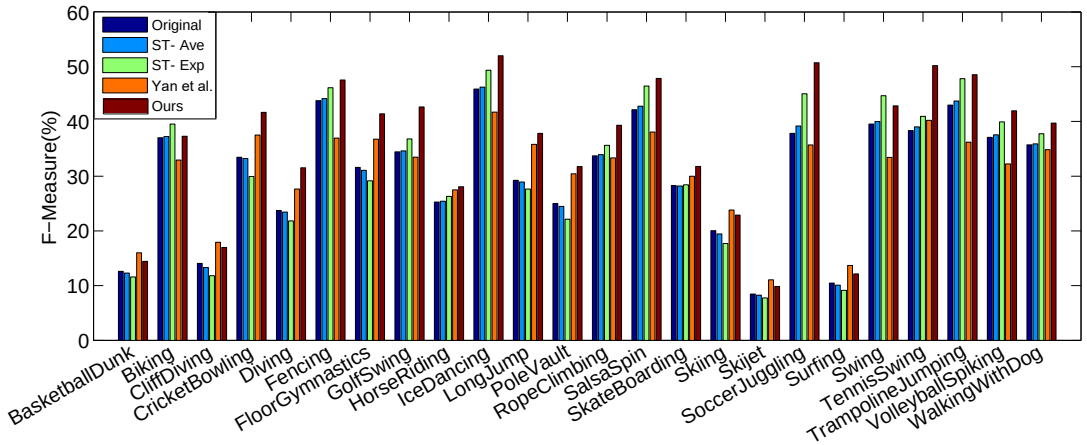


Figure 4.5: Per-category results on UCF 101.

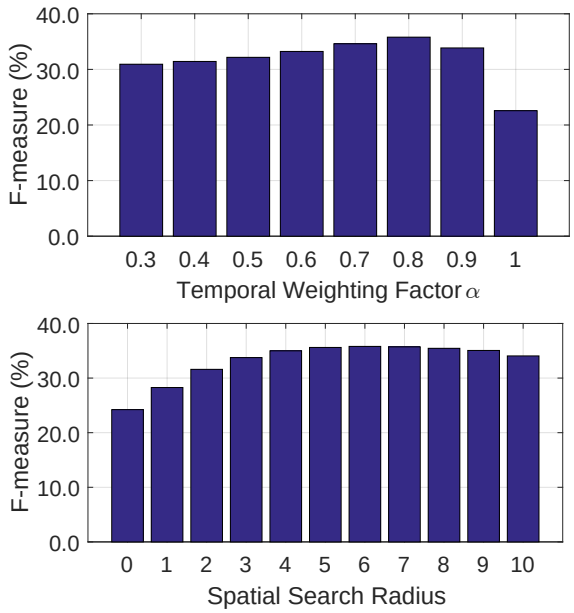


Figure 4.6: Parameter sensitivity evaluation of the proposed method on UCF 101. The vertical axis is the mean F-measure of all the videos. The temporal weighting factor is evaluated by fixing the spatial search radius to 6 and the spatial search radius is evaluated by fixing the temporal weighting factor to 0.8.

As the method in [84] did not consider the temporal weighting, it becomes similar [2] to our special case of $\alpha = 1$ which overly emphasizes the past frames. This may partially explain its weak performance as shown in Table 4.2.

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 120 frames per second when the input size is 160×120 and the spatial neighborhood radius is 6.

4.3.3 Online Filtering of Scene Parsing Map

4.3.3.1 Dataset Description and Experimental Protocols

For the main parts of this experiment, we evaluate our approach on online filtering of scene parsing maps of four sequences. The video and initial scene parsing maps of NYU, MPI and CamVid-05VD videos can be obtain from `http://www.cs.cmu.`

[2]Different from our method, the method in [84] applied certain heuristic procedures such as multiplying the pixel tracing score by the original score to pull up the performance.

edu/~dmunoz/projects/onboard.html.

- 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 [116].
- MPI [117] 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 [88].
- 05VD and 01TP videos [113] contain 5100 and 1831 frames taken at 30 Hz during daylight and dusk, respectively. The sequences are sparsely labeled at 1 Hz with 11 semantic labels. To perform initial scene parsing, hierarchical inference machine [14] is used for 05VD and our location constraint SVM classifier [7] for 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 implemented baseline methods[3], we use a different set of parameters obtained by a grid search for each video. The parameter settings are stated in Section 4.5.2. To avoid the over fitting, the following procedure is adapted for both the baselines and our method. We uniformly sample 25% of the annotated frames from each video, and only use them for parameter search. We find the performance does not differ much compared with using all the frames for parameter selection of that video.

Two measures are used for evaluation, *i.e.*, per-pixel accuracy and per-class accuracy. Per-pixel accuracy measures the pixel classification rate and per-class accuracy measures the average intersection over union (IOU) score over all the semantic labels. The IOU score between ground-truth A and output B is defined as $\frac{|A \cap B|}{|A \cup B|}$ for

[3]Note that we do not have code of Miksik *et al.* 's method [10], hence we quote their results as reported in the paper.

each semantic label. Per-pixel accuracy emphasizes the correctness of the majority classes, while per-class accuracy balances both majority and minority classes. We express both measures in percentage term.

Besides the above main parts of the experiments, we also test the algorithm on other datasets to see how it compares with some state-of-the-art methods [21,92–94], such as deep-learning based methods for video semantic segmentation.

We organize the scene parsing experiments into five parts. In Section 4.3.3.2 we firstly discuss why our method performs well for certain cases in a controlled scenario. Secondly, in Section 4.3.3.3 we compare with some classic filters that do not rely on video appearance. This is followed by a discussion on appearance modeling and the related methods in Section 4.3.3.4. To achieve real-time performance, we perform smoothing on superpixels. The benefits of superpixel-level smoothing are discussed in Section 4.3.3.5. Finally, in Section 4.3.3.6 we analyze the performance and the limitations of our methods compared with state-of-the-art smoothing methods.

4.3.3.2 Simulation Experiments

To illustrate that our method can well handle isolated noises in a controlled condition, we perform simulation experiments based on the ground-truth maps of MPI video. To simplify the analysis, we only consider a binary labeling scenario where "lane-marking" and "vehicles" are used as foreground. To simulate the noisy labeling maps, for each frame we randomly pick a few patches of size 5×5 at different locations and switch their ground-truth labeling from foreground to background and vice versa. The percentage of noisy pixels varies from 0% to 25%, and the noise behaves like salt-and-pepper noise.

We compare our method with moving average and exponential filter. The image examples and quantitative evaluations are shown in Fig. 4.7 and Fig. 4.8, respec-

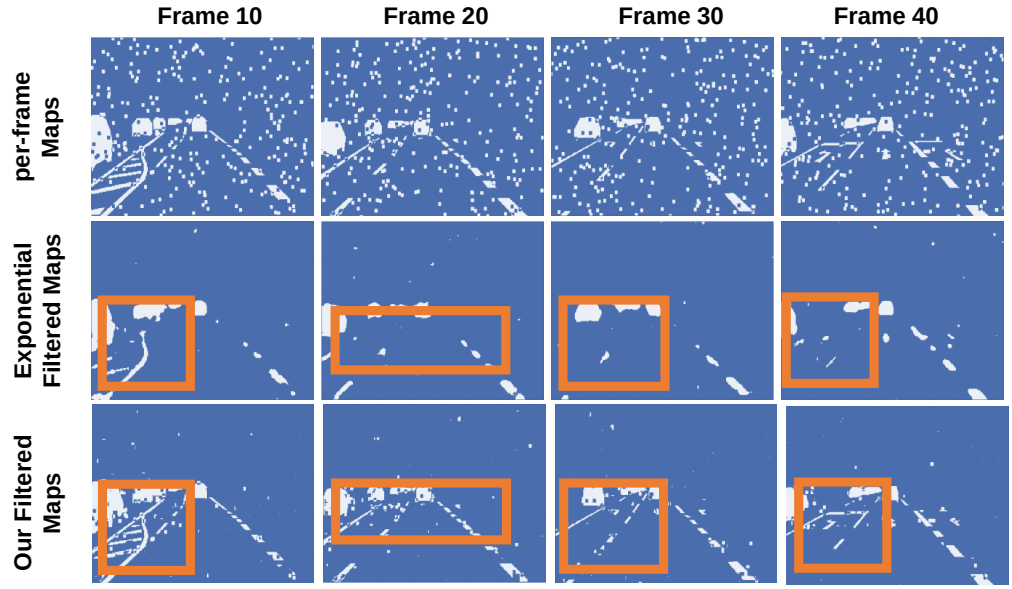


Figure 4.7: Results of our method on simulated scene parsing maps. The 1^{st} Row: per-frame classification maps. The 2^{nd} Row: refined maps by exponential filter. The 3^{rd} Row: refined maps by our filter. The highlighted regions demonstrate that our non-linear smoothing can better preserve fine details (such as "lane-marking" region) compared with spatio-temporal exponential smoothing.

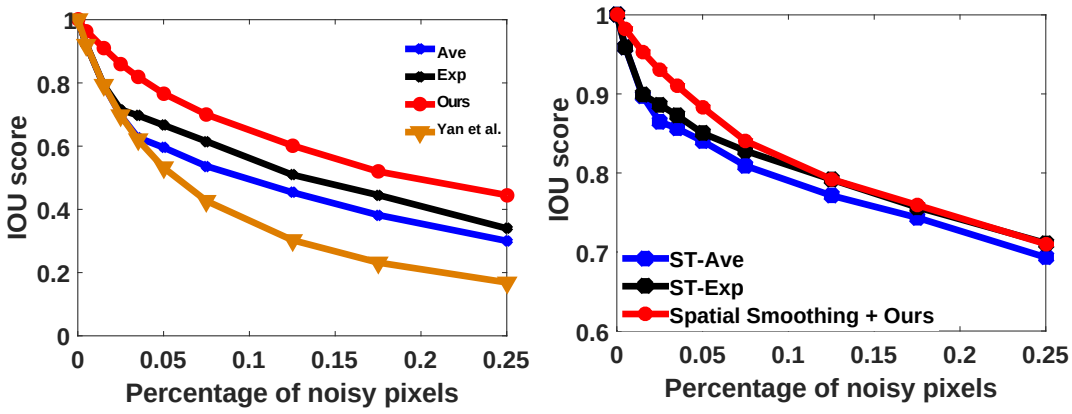


Figure 4.8: Left: Comparisons with moving average and exponential filter. Right: Comparisons among spatio-temporal moving average, exponential filter and our filter with spatial smoothing. The IOU score is calculated for the foreground.

	Original	Ave	ST-Ave	Exp	ST-Exp	[84]	Ours
NYU	71.1 (28.3)	74.3 (28.0)	74.9 (28.1)	75.0 (28.4)	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)	93.7 (76.0)	93.7 (76.0)	**94.3 (77.4)**	**94.3** (77.1)
01TP	76.7 (39.0)	78.1 (40.0)	82.4 (43.7)	78.8 (40.7)	82.4 (43.8)	76.7 (39.0)	**83.1** (**44.5**)
05VD	84.6 (47.8)	85.2 (47.8)	85.2 (47.8)	85.5 (48.0)	85.5 (48.0)	84.7 (47.5)	**85.6** (**48.1**)

Table 4.3: Comparisons with baseline smoothing methods. Numbers outside and within brackets are per-pixel accuracies, and average per-class IOU scores, respectively.

tively. Our main conclusions are: (1) our method performs significantly better than moving average and exponential filter when the noise level is small. However, with large amounts of noises, the performance gain is less significant. (2) Our non-linear smoothing can better preserve fine details in the texture-rich regions (such as "lane-marking" region) compared with linear filters such as spatio-temporal exponential smoothing. In video smoothing, it is a challenging task to keep fine image details while still removing the noises. Besides the results shown in the figures, we also experiment with BM4D video denoising method [161] and find it performed poorly due to their Gaussian assumption. This highlights the fact that some commonly used assumptions in video denoising methods are not the best choice to perform labeling map denoising, as labeling noises depend on the underlying image analysis algorithm and may have different behavior. This simulation demonstrates the advantages of our AES in denoising salt-and-pepper noises in texture-rich regions. We would expect similar behaviors in the following experiments.

4.3.3.3 Comparisons With classic linear filters [109] and Yan *et al.* [84]

We compare with some closely related filters of similar complexity. From Table 4.3, we observe that our method outperforms all these baselines by a considerable margin, except for 05VD. In Fig. 4.9 and Fig. 4.10, we also show some image

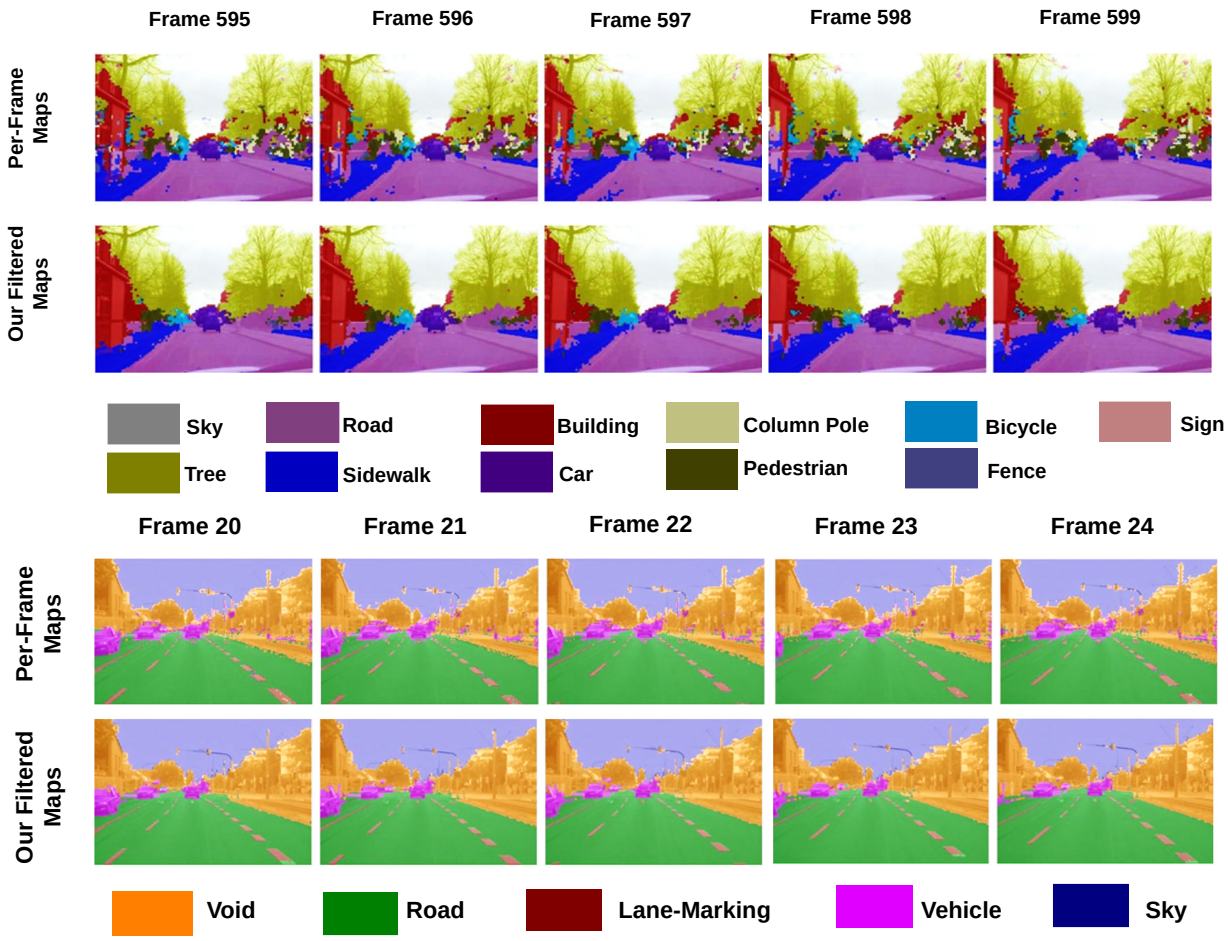


Figure 4.9: Results of our method with appearance modeling on 01TP and MPI, respectively. The 1^{st} Row: per-frame classification maps. The 2^{nd} Row: refined maps by our filter.

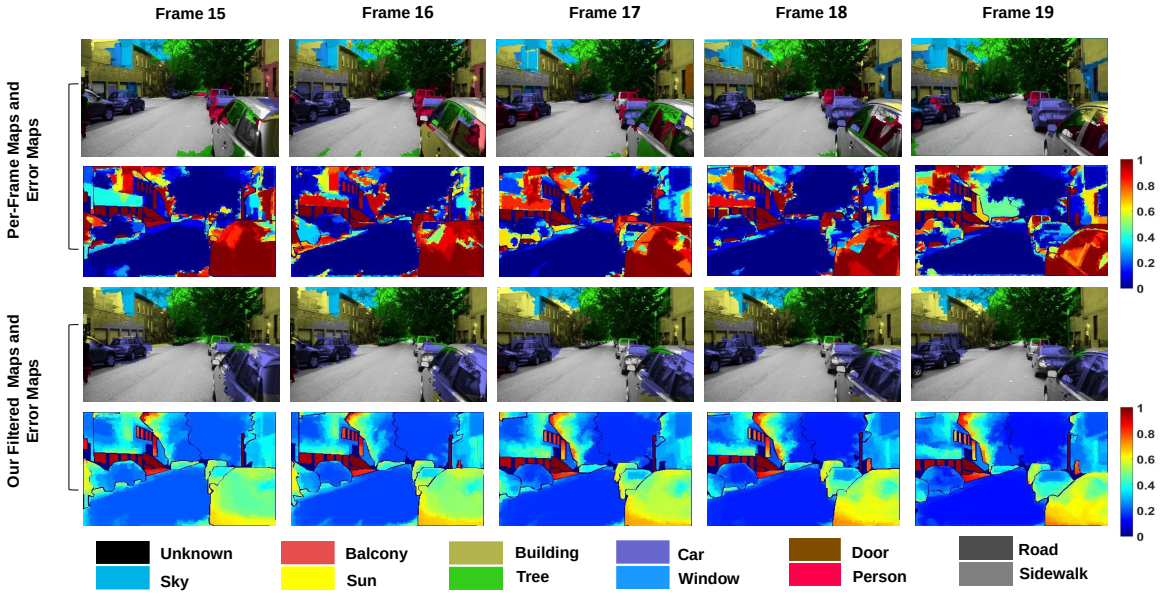


Figure 4.10: Results of our method with appearance modeling on NYU. The 1^{st} and the 2^{nd} Rows: per-frame classification maps and the corresponding error maps, respectively. The 3^{rd} and the 4^{th} Rows: refined maps by our filter and the corresponding error maps, respectively. The error maps are obtained by calculating the score differences between ground truth and per-frame classification scores.

results. The significant benefits of our spatio-temporal smoothing can be clearly observed, as we have successfully corrected a lot of "flickering" classifications in the initial maps. Our method is particularly good at handling salt-and-pepper noises, *i.e.*, filling small labeling holes and removing isolated noisy segments. Such benefits have been discussed in Section 4.3.3.2.

Analysis of the temporal noise pattern. To have a good understanding of the above experimental results, studying the noise pattern of each video as shown in Fig. 4.11 is important. We firstly compute the error map as the pixel-wise absolute differences between the ground truth and actual classification scores. For example, if a pixel is under "building" category and the classification score for "building" is 0.6, the error will be 0.4. As Miksik *et al.* [10] point out, temporal smoothing algorithm including our method cannot correct the bias of an image analysis algorithm. For example, if an image analysis algorithm consistently miss

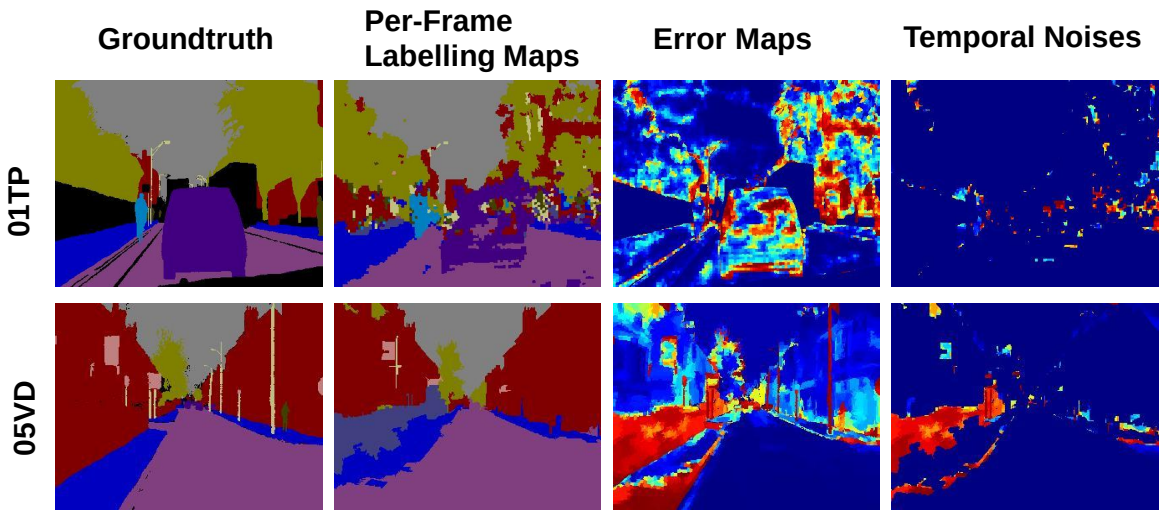


Figure 4.11: Temporal noises maps for 01TP and 05VD. The error maps show the pixel-wise absolute differences between the ground truth and actual classification scores. The temporal noise maps only include the wrongly labeled pixels having different labels in the previous frame.

the "car" category, there is little hope of recovering it by considering the temporal consistency. Thus we do not consider such pixels that are consistently and wrongly labeled across the neighboring frames. Our analysis only focuses on the temporal noise maps, highlighting the wrongly labeled pixels having different labels in the previous frame, where the neighborhood relationship between pixels are determined by optical flow.

From Fig. 4.11, the temporal noises of 01TP are often isolated noises. In contrast, 05VD contains large region noises. Our method is well suited for handling isolated temporal noises, by considering the exponential smoothing scores from previous frame's spatial neighborhoods. However as the noises of 05VD is of larger area, the noises in current and last few frames have a higher probability of being overlapping with each other, and become spatially and temporally consistent. Thus they are more difficult to correct and the performance on 05VD is expected to be worse.

	NYU	MPI	01TP	05VD
Ours	**76.6** (28.3)	94.3 (77.1)	83.1 (44.5)	**85.6** (48.1)
Ours + Appearance	76.3 (**29.5**)	**94.6** (**78.8**)	**84.1** (**46.2**)	**85.6** (**48.3**)

Table 4.4: Effects of appearance modelling. Numbers outside and within brackets are per-pixel accuracies and average per-class IOU scores, respectively.

4.3.3.4 Appearance Modeling

We demonstrate the effects of appearance modeling and compare with more sophisticated methods that also takes into account the video appearance. Table 4.4 shows that the appearance term can further boost the average per-class score, since it can avoid overly smoothing certain labels by considering the color difference. It also shows the appearance information can consistently improve the performance for videos with salt-and-pepper noises such as MPI and 01TP. From Table 4.7 to Table 4.9, we observe that it can help to preserve and recover some small but important image regions, such as "lane-marking" in MPI (the accuracy improves from 39.7% to 48.1 %), and "pedestrian" and "car" in 01TP (the accuracies improve from 25.4% to 31.4% and from 64.5% to 67.5%, respectively). Our method's ability in preserving small details and objects can also be seen from Table 4.3 and Table 4.4. They show that our average per-class IOU scores are higher than baselines, implying that our method is better at preserving small classes. In spite of that, both Miksik *et al.* 's method [10] and our method have smoothed out certain classes, i.e., "door" and "sign" in NYU, so preserving small classes is a challenging issue which requires further study.

Table 4.5 shows the performance of two appearance-based baseline methods. From the first row, we see that our method performs comparably well or better than the optical flow guided spatio-temporal exponential filter defined by Eq. 4.14, especially when the obtained optical flow is not reliable. For example, as 01TP is captured

	NYU	MPI	01TP	05VD
ST-Exp + Flow	75.8 (29.4)	93.7 (76.1)	82.8 (43.8)	86.1 (48.5)
[10]	75.3 (**29.6**)	93.6 (74.2)	-	**86.9 (50.0)**
Ours + Appearance	**76.3** (29.5)	**94.6** (**78.8**)	**84.1 (46.2)**	85.6 (48.3)

Table 4.5: Comparisons with optical flow guided spatio-temporal exponential smoothing and Miksik *et al.* [10] on scene parsing map smoothing. Numbers outside and within brackets are per-pixel accuracies and average per-class IOU scores, respectively.

at dusk, its image quality is low and the noisy optical flow contributes negatively to the performance. In addition, mainly due to the slowness of the optical flow computation [160], filtering of each frame takes more than 1.5 seconds.

Because Miksik *et al.* 's method [10] use sophisticated appearance modeling techniques such as metric learning and optical flow to do pixel tracing, it is more robust to the noises in the initial maps. Therefore from the second row of Table 4.5 our method performs worse than [10] on 05VD. However our method performs comparably for the other videos, and better for some classes such as "lane" as shown in Table 4.7. Moreover, their method uses 0.8 seconds to process each frame which is more than 10 times slower than our method.

4.3.3.5 Superpixel-Level Filtering

As scene parsing maps contain multi-class annotations, we have to run our smoothing algorithm multiple times (*e.g.*, 11 times for NYU) and use "winner-taking-all" to make the final classification. To reduce computational cost, we extract around 5000 superpixels on each frame using the SLIC algorithm [40]. The initial score of each superpixel is set to the average score of its pixels, and its smoothing score is then updated from the previous frame spatial neighborhood, i.e., using the scores of previous frame superpixels whose central position (x_{t-1}, y_{t-1}) falling into $\mathcal{N}(x_t, y_t)$.

	Original	Ours (Pixel Level)	Ours (Superpixel Level)
NYU	71.1 (28.3)	76.5 (**28.4**)	**76.6** (28.3)
MPI	93.1 (74.6)	93.9 (76.4)	**94.3** (**77.1**)

Table 4.6: Comparisons between superpixel-level and pixel-level smoothing for NYU and MPI videos. Numbers outside and within brackets are per-pixel accuracies, and average per-class IOU scores, respectively.

	Background	Road	Lane	Vehicle	Sky	Ave
Original	87.4	91.0	45.7	54.4	94.6	74.6
[10]	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
Ours + Appearance	**90.9**	**91.9**	**48.1**	68.8	94.1	**78.6**

Table 4.7: Per-class intersection-over-union (IOU) scores on MPI.

Because all the smoothing operations are performed on the superpixel level, a significant speedup is achieved. For a 320×240 video with 11 semantic labels, when the spatial neighborhood radius R is set to 10, we can accelerate the smoothing speed to 50 frames per second without appearance modelling and 20 frames per second with appearance modelling using 8-core parallel processing. So overall the real-time performance is achieved. In contrast, the code runs at 5 frames per second if the smoothing is performed on pixel level.

From Table 4.6, superpixel level smoothing not only runs faster but also produces comparable or slightly better results. As the score of each superpixel is obtained by averaging its pixels' scores, a small amount of spatial noises can be removed via the averaging.

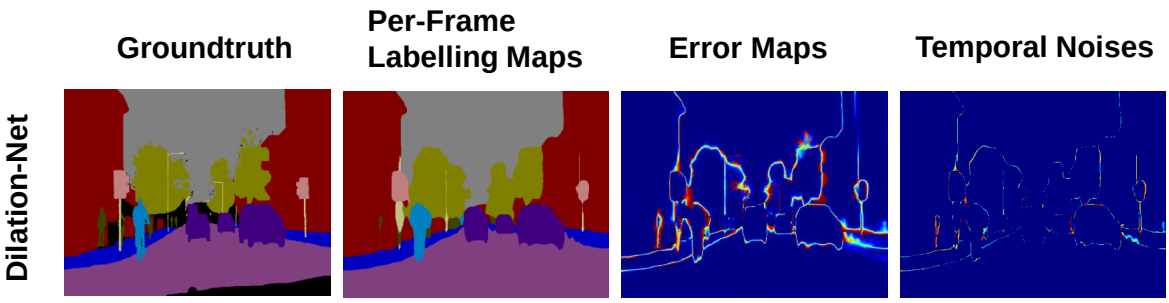


Figure 4.12: Temporal noises maps for 01TP using DilatedNet [2]. The error maps show the pixel-wise absolute differences between the ground truth and actual classification scores. The temporal noise maps only include the wrongly labeled pixels having different labels in the previous frame.

4.3.3.6 Comparisons with State-of-the-Art Smoothing Methods [21,92–94] for Video Semantic Segmentation

In this section, we discuss our method's performance and limitations compared with some state-of-the-art smoothing approaches [21, 92–94] for video semantic segmentation. The current state-of-the-art approaches use a recent deep neural network called DilatedNet [2] to generate the initial semantic segmentation scores on CamVid dataset [113], hence we re-run our algorithm on DilatedNet outputs to test how much we can further improve its performance using spatio-temporal smoothing. The existing approaches [21,92–94] can at least improve the mean IOU score by 0.8 (mean IOU improved from 65.3 to 66.1) or more. However our approach was not able to further improve DilatedNet outputs. To understand such a result, we study the noise pattern of DilatedNet's outputs as shown in Fig. 4.12.

From the above figure, we observe that the errors of DilatedNet [2] distribute mostly around boundaries and certain small objects. It is harder for our algorithm to correct such types of errors around boundaries for the following reasons. For each pixel, our algorithm will attempt to discovery a spatio-temporal path in previous frames along which the accumulated score(evidence) for the correct category

is sufficiently large, so the accumulated score(evidence) of the correct category can be used to overcome the wrong classification at current frame. This is easier to achieve if most surrounding pixels in past frames have correct classification scores; in other words, our algorithm can handle isolated noises better if in the past frames, most surrounding pixels' classifications are correct. Such an assumption is difficult to satisfy around boundary pixels, simply because accurate pixel classification around object boundaries are very challenging. For example, due to imaging defects, lighting conditions or camera motions, around the boundaries image appearance is often blurred and ambiguous, and boundary pixels' classification becomes unstable or incorrect in many frames; our algorithm may not be able to handle such type of errors. However, this does not imply our algorithm cannot preserve boundary. As shown in the simulation experiments in Section. 4.3.3.2 and some scene parsing experiments, if the initial boundary pixel classification is correct, our algorithm can preserve the boundaries or other types of details better than the baseline methods. For such type of boundary errors, a more sophisticated appearance based approach may work better.

In addition, From Fig. 4.12 we observe DilatedNet [2] output does not have a lot of temporal noises. Our algorithm is more suitable to handle "flickering" temporal noises. For the errors that are consistent across different frames (for example, traffic poles are often consistently missed by the DilatedNet [2] outputs for many frames), it is better to use a learning-based algorithm to correct the biases in the initial classification maps. End-to-end learning of image semantic segmentation deep network and spatio-temporal smoothing network are more suitable to correct such type of biases; this partially explains the good performance of these deep-learning based approaches [92–94].

4.4 Conclusions

In this chapter, we have proposed an efficient online video filtering method, named adaptive exponential smoothing (AES), to refine pixel classification maps. Compared with the traditional average and exponential filter, our AES does not fix the spatial location or temporal smoothing bandwidth while performing temporal smoothing. Instead, it performs adaptive filtering for different pixels; therefore, it is able to better address missing and false pixel classifications, and better tolerate fast object movement and camera motion. The proposed appearance-based adaptive weighting factor can also help to preserve important fine details.

The experimental evaluations on saliency map filtering and multi-class scene parsing map filtering validate the superiority of the proposed method compared with average and exponential filters, as well as alternative approaches that rely on expensive optical flow computations. In addition, our extensive analysis experiments demonstrate that the proposed AES is particularity suitable for salt-and-pepper noises which explains why it can better smooth certain scene parsing videos. Thanks to the proposed dynamic programming algorithm for pixel tracing, our filtering method has linear time complexity and runs in real time.

	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
[10]	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	1.1	0.0	8.0	81.8	8.2	28.3
Ours + Appearance	54.1	54.2	0.0	17.9	0.8	82.3	13.9	0.0	**9.9**	**83.1**	8.4	29.5

Table 4.8: 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	62.8	85.5	50.2	0.0	76.5	17.8	**5.7**	9.4	60.0	15.5	39.0
	Ours	57.7	70.4	89.6	64.5	0.0	79.4	25.4	2.6	9.0	61.8	28.1	44.5
	Ours + Appearance	**59.5**	**71.3**	**92.0**	**67.5**	0.0	**80.1**	**31.4**	2.0	**13.3**	**62.5**	**28.8**	**46.2**
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
	[10]	**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	78.2	59.4	88.6	70.1	26.3	86.8	32.8	15.2	7.2	56.3	8.1	48.1
	Ours + Appearance	78.4	59.5	88.6	70.6	26.6	86.8	34.1	15.0	7.4	56.5	7.7	48.3

Table 4.9: Per-class intersection-over-union (IOU) scores on 01TP and 05VD.

4.5 Appendix

4.5.1 Proof of Lemma 4.2

We provide the proof of Lemma 4.2. Lemma 4.1 is simply a special case of Lemma 4.2 when setting $h(j) = 1, \forall j \in \mathbb{N}$.

Proof: We prove by induction. When $t = 1$:

$$M(\mathcal{P}^*_{s\to 1}(v_1)) = \widehat{M}(v_1). \tag{4.15}$$

Suppose when $t = m$:

$$M(\mathcal{P}^*_{s\to m}(v_m)) = \widehat{M}(v_m) \tag{4.16}$$

When $t = m + 1$, we have:

$$\begin{aligned}
&M(\mathcal{P}^*_{s\to m+1}(v_{m+1})) \\
&= \max_{1\le s\le m+1}\left\{ \max_{\substack{\forall s\le i\le m,\\ (x_i,y_i)\in\mathcal{N}(x_{i+1},y_{i+1})}} \sum_{i=s}^{m}\left(\alpha^{(m+1-i)}\Big(\prod_{j=i}^{m} h(j)\Big)\right.\right. \\
&\quad \left.\left.\times U(v_i)\right)\right\} + U(v_{m+1}) \qquad (4.17)\\
&= \max\left\{ \max_{1\le s\le m}\left\{ \max_{\substack{\forall s\le i\le m,\\ (x_i,y_i)\in\mathcal{N}(x_{i+1},y_{i+1})}} \sum_{i=s}^{m}\left(\alpha^{(m+1-i)}\right.\right.\right. \\
&\quad \left.\left.\left.\times \Big(\prod_{j=i}^{m} h(j)\Big)U(v_i)\right)\right\}, 0\right\} + U(v_{m+1}). \qquad (4.18)
\end{aligned}$$

From Eq. 4.17 the starting frame of path can be varied by $1 \le s \le m + 1$, where $m + 1$ is the current frame number. $s = m + 1$ occurs in the optimal path solution if and only if previous accumulative evidence is always negative, *i.e.*, the inner summation of Eq. 4.18 $\sum_{i=s}^{m} ...$ becomes negative for any $1 \le s \le m$. In such

a case we start a new path with only current score $U(v_{m+1})$. So we obtain Eq. 4.18.

We also have:

$$\max_{1\le s\le m}\left\{\max_{\substack{\forall s\le i\le m,\\(x_i,y_i)\in\mathcal{N}(x_{i+1},y_{i+1})}}\sum_{i=s}^{m}\left(\alpha^{(m+1-i)}\Big(\prod_{j=i}^{m}h(j)\Big)\times U(v_i)\right)\right\}$$

$$=\alpha\times h(m)\times\max_{(x_m,y_m)\in\mathcal{N}(x_{m+1},y_{m+1})}\max_{1\le s\le m}$$

$$\left\{\max_{\substack{\forall s\le i\le m-1,\\(x_i,y_i)\in\mathcal{N}(x_{i+1},y_{i+1})}}\sum_{i=s}^{m-1}\left(\alpha^{(m-i)}\Big(\prod_{j=i}^{m-1}h(j)\Big)U(v_i)\right)+U(v_m)\right\} \tag{4.19}$$

$$=\alpha\times h(m)\times\max_{(x_m,y_m)\in\mathcal{N}(x_{m+1},y_{m+1})}M(\mathcal{P}^{*}_{s\to m}(v_m)) \tag{4.20}$$

From Eq. 4.18 and Eq. 4.20, we have:

$$M(\mathcal{P}^{*}_{s\to m+1}(v_{m+1}))$$

$$=\max\left\{\max_{(x_m,y_m)\in\mathcal{N}(x_{m+1},y_{m+1})}\alpha\times h(m)\times M(\mathcal{P}^{*}_{s\to m}(v_m)),0\right\}+U(v_{m+1}) \tag{4.21}$$

$$=\max\left\{\max_{(x_m,y_m)\in\mathcal{N}(x_{m+1},y_{m+1})}\alpha\times h(m)\times\widehat{M}(v_m),0\right\}+U(v_{m+1}) \tag{4.22}$$

$$=\widehat{M}(v_{m+1}) \tag{4.23}$$

From Eq. 4.21 to Eq. 4.22 we use the induction hypothesis by Eq. 4.16. We see when $t=m+1$, $M(\mathcal{P}^{*}_{s\to m+1}(v_{m+1}))=\widehat{M}(v_{m+1})$ also holds. It concludes the proof according to mathematical induction.

4.5.2 Parameter Settings

The parameter settings of UCF101 experiments have been detailedly explained in Section 4.3.2; so in this section we mainly explain the parameters used in scene parsing experiments. As motioned in Section 4.3.3.1, we use a different set of parameters obtained by a grid search for each of the four videos, as they use different

	NYU	05VD	MPI	01TP
Ours	(2, 0.9)	(0, 0.5)	(20, 0.8)	(12, 0.8)
Ours + Appearance	(5, 0.9)	(2, 0.7)	(17, 0.8)	(11, 0.9)

Table 4.10: Spatio-temporal smoothing parameters used in scene parsing experiments. The first number in each pair refers to spatial search radius, while the second number refers to temporal weighting factor.

scene parsing algorithms and the noise patterns of initial classification maps are also distinguishably different.

The selected parameters are shown in Table 4.10. We note that the initial classification maps of NYU and 05VD videos contain large-sized noises, while for MPI and 01TP videos they are mostly isolated noises. From Section 4.3.3.2, our method works well for removing isolated temporal noises, by considering the exponential smoothing scores from previous frame's spatial neighborhood. Hence the neighborhood search radius for MPI and 01TP is selected to be a larger value to enable more denoising capability.

Chapter 5

Exploiting Region Consistency for Actor-Action Semantic Segmentation

5.1 Introduction

In this chapter, we study the actor-action semantic segmentation problem. This task was recently proposed by Xu *et al.* [32], and targets joint labeling of both action categories (*e.g.*, running and jumping), and actor categories (*e.g.*, adult and cat) for every location in a scene. It has a variety of applications, ranging from robotics and autonomous driving to video captioning and video editing.

Despite much work in related research areas, such as image semantic segmentation [3, 20, 26, 50, 52, 154], video semantic segmentation [21, 162], video object segmentation [163, 164] and action detection [35, 165], relatively few works [32–35] have specifically targeted at the actor-action segmentation problem. These works study the actor-action segmentation problem though different lens; examples include a focus on the interaction between actor and action labels [32, 35], mechanisms to model

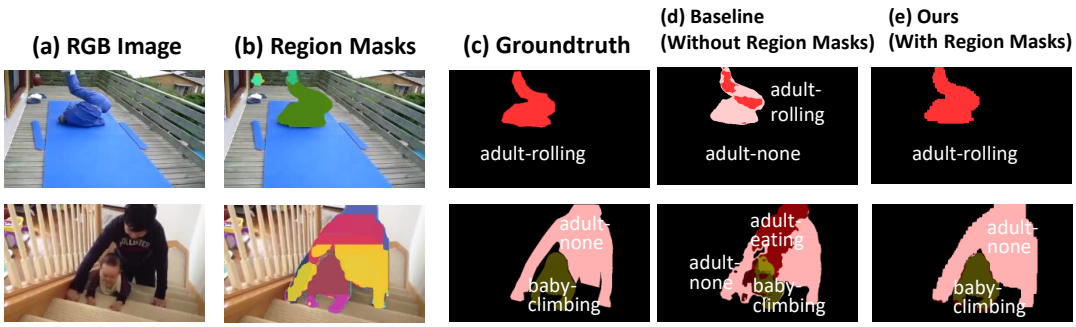


Figure 5.1: Examples that demonstrate region masks can be utilized to improve actor-action segmentation results. Baseline refers to the DeepLab-CNN [3] based method which does not utilize region masks.

long-range interaction of different actors [33], and training procedures which only require weakly-supervised actor-action labeling [34]. While the results are encouraging, we show that the performance can be amplified by harnessing region masks from instance segmentation algorithms.

Instance segmentation algorithms [4, 45] have made significant advancements recently, and is able to provide high-quality region masks. As demonstrated in previous region-based segmentation approaches [25, 26, 64], incorporating region masks into semantic segmentation networks can provide an adaptive receptive field and enforce consistent predictions over regions - such a property is also beneficial for the actor-action segmentation task. When an actor performs an action, different parts of the actor provide the different types of cues for the action. The movements of some parts better reflect the action than those of the others. When the pixels inside the actor regions are labeled independently, different parts may receive different action labels (see Fig. 5.1(d)). However, by enforcing pixels inside the actor regions to take the same action label, we can achieve consistent labeling for better actor-action segmentation performance (see Fig. 5.1(e)).

For the above reasons, we propose an end-to-end region-based actor-action segmentation approach as shown in Fig. 5.2. For each frame, we first obtain high-quality region masks using an object instance segmentation network [4] trained explicitly

for a specific target dataset. Then, a two-stream network is used to learn two types of features which capture appearance and motion information respectively. Finally, the fused features from the two streams together with the region masks are passed to two region-based segmentation networks for predicting actor-action labels. We conduct a comprehensive set of experiments on the A2D dataset [32], and demonstrate that both region-based segmentation strategy and robust fused features from the two-stream network contribute to performance improvements. The proposed approach outperforms the state-of-the-art results by more than 8% in mean class accuracy, and more than 5% in mean class IOU, which confirms the effectiveness of the proposed method.

In summary, the contributions of this chapter are as follows:

- Because many actions are reflected more prominently by movements of certain body parts instead of the whole body, different body parts of the actors may have inconsistent action labelling if pixels are labelled independently. Hence, we propose to utilize region masks to address the inconsistency issue of actor-action semantic segmentation problem. Through comprehensive experiments, we demonstrate the importance of exploiting region masks for consistent actor-action labeling.

- We design an end-to-end network specifically for actor-action semantic segmentation. In particular, 1) for each object we obtain multiple region masks and fuse their predicted categories via a region-to-pixel layer for final prediction, and 2) we use a two-stream network architecture taking an RGB image and an optical flow image as the input to further improve the semantic segmentation performance.

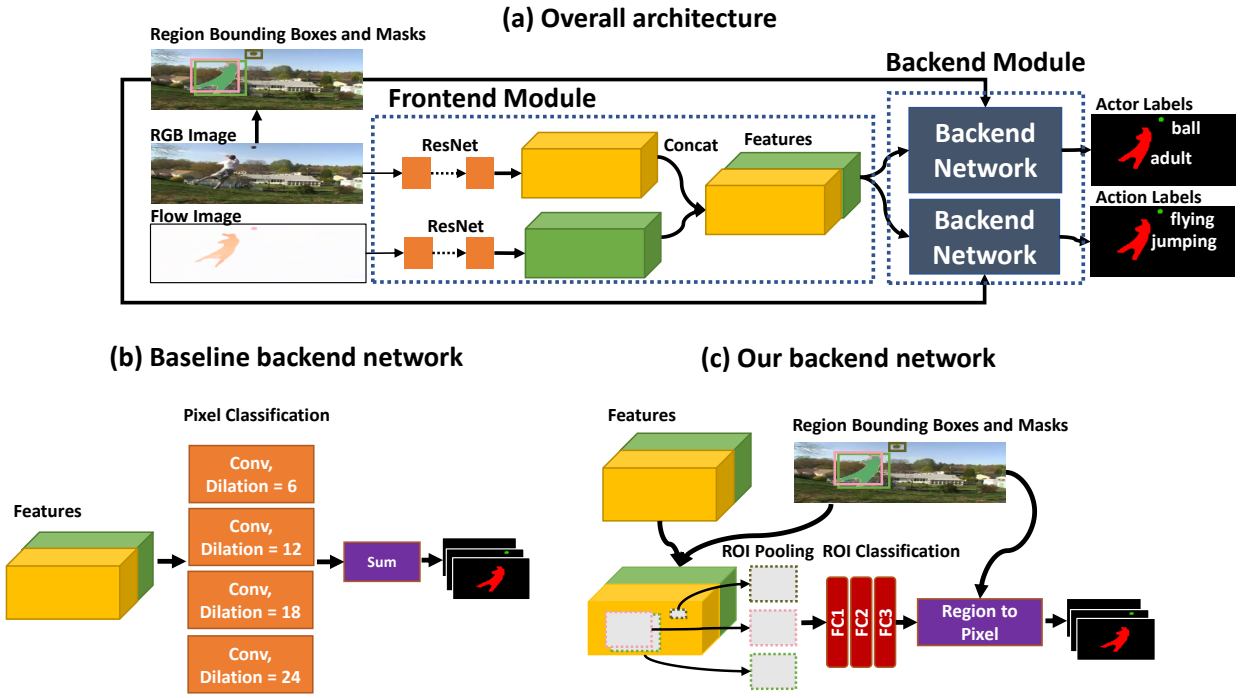


Figure 5.2: Overview of our end-to-end region-based actor-action segmentation approach. Our network consists of a front-end module and a back-end module containing two networks of identical structure. The front-end module is a two-stream network that learns both appearance and motion features. The concatenated features from both streams are passed to the back-end module which predict actor and action labels, respectively. Our main contribution is the adaption of region-based semantic segmentation networks as the back-end module.

5.2 Proposed Method

For a given video, we assign an actor-action label (*e.g.*, adult-running) to each pixel. The overview of the proposed approach is illustrated in Fig. 5.2. First, for each frame, a set of candidate region masks are generated using an instance segmentation algorithm [4] (Section 5.2.1). These region masks along with RGB and the corresponding optical flow image are taken as inputs. Then, a two-stream network (front-end module) is used to obtain both appearance and motion features (Section 5.2.2). Finally, the concatenation of both types of features is passed to two region-based semantic segmentation networks (back-end module) which predict actor and action labels, respectively (Section 5.2.3).

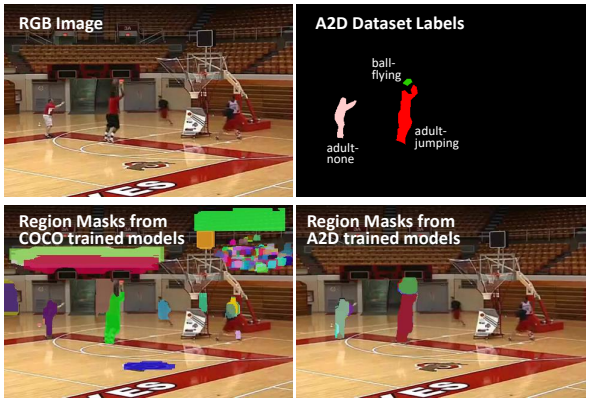


Figure 5.3: Examples of region masks generated from fully convolution instance segmentation (FCIS) algorithm [4]. To perform semantic segmentation on a particular dataset, *e.g.*, A2D dataset, FCIS should be fine-tuned on the same dataset, as region masks generated by FCIS trained on a more generic dataset, *e.g.*, COCO [5] may contain irreverent background objects.

5.2.1 Region Mask Generation

Principally, any mask proposal methods [42, 44] or instance segmentation algorithms [4, 45] can be adopted to generate a set of candidate region masks for each frame. Let $\{\mathbf{b}_i, \mathcal{M}_i\}_{i=1}^{N}$ be the set of generated candidate region masks where N is the number of region masks, $\mathbf{b}_i \in \mathcal{R}^4$ is a bounding box and $\mathcal{M}_i \in \mathcal{R}^{W_b \times H_b}$ is a probability mask indicating the likelihood of each bounding box pixel belonging to the mask.

To achieve good performance, it is important to use a set of high-quality region masks. In our experiments, we adopt a fully convolution instance segmentation (FCIS) [4] algorithm for region mask generation. For each frame, the predicted instance segmentation masks by the network are taken as candidate region masks. For actor-action segmentation on a specific dataset, we train the FCIS network on the same dataset instead of on a general large-scale dataset like COCO [5]. The FCIS network trained on the COCO dataset [5] is more generic but would produce more irrelevant background regions which could harm the actor-action segmentation performance (see Fig. 5.3 for illustration).

5.2.2 Front-end Module

We adopt a two-stream architecture similar to the one used by [35, 163] to generate robust feature representations to be used by the back-end semantic segmentation networks. The first stream models appearance information by taking as input an RGB image, and the second stream models motion information by taking as input the optical flows. We use FlowNet [166] to calculate optical flows which are converted into an RGB image with the method used in [167].

Each stream is adapted from a ResNet-101 classification network [56]. To preserve spatial details, the last two layers of the ResNet-101 are replaced by dilated convolutions, which increases the resolution of the output map from 1/32 to 1/8 of the original image size. After we obtain the feature maps from both streams, we concatenate them together as an input for the back-end module. That is, $\mathbf{f} \in R^{W \times H \times 2C}$, where $\mathbf{f}$ is the concatenated feature representation, W and H are the width and height of the feature maps and C is the number of feature channels of each individual stream. Different from [163] which uses a late fusion for the sake of reducing model complexity, we apply this early fusion strategy to combine the two streams with better flexibilities.

5.2.3 Back-end Module

Given the set of candidate region masks $\{\mathbf{b}_i, \mathcal{M}_i\}_{i=1}^{N}$ and feature representation from the front-end module $\mathbf{f} \in R^{W \times H \times 2C}$, we want to predict both actor and action categories of each pixel. In other words, we predict two label maps $\mathbf{l}_{actor} \in \mathcal{R}^{W \times H \times K_{actor}}$ and $\mathbf{l}_{action} \in \mathcal{R}^{W \times H \times K_{action}}$, where K_{actor} and K_{action} are the number of actors and actions to be predicted. We employ a multi-task formulation with two networks to achieve this goal. The first network is to predict the actor label and the second network is to predict the action label. Similar to previous works [3, 50, 163], the cross-entropy loss is used for each branch. Specifically, the training loss for each

pixel j is defined by

$$loss_j = -\log(p_{j,actor}(o_{j,actor})) - \log(p_{j,action}(o_{j,action})), \tag{5.1}$$

where $o_{j,actor}$ is the ground-truth actor label and $o_{j,action}$ is the ground-truth action label. At test time, the joint actor-action probability is the element-wise product of the two probability vectors.

In DeepLab CNN [3], multi-scale dilated convolution layers are applied to the front-end feature map to compute class probabilities directly (see Fig. 5.2(b)). However, class probabilities of each pixel are independently computed, and the scale set of receptive fields are fixed a priori. This may lead to undesirable results where the body of a single actor is segmented into multiple parts of different action labels (see examples in Fig. 5.5(a)), and small-sized objects may be missed (see examples in Fig. 5.5(b)). To address this problem, in our back-end network (see Fig. 5.2(c)), we utilize region masks to adaptively select receptive fields and enforce the pixels in each region mask to have a consistent labeling.

First, we obtain the overall region mask scores $s_{i,actor}$ and $s_{i,action}$ for each bounding box $\mathbf{b}_i$. To do this, we classify each bounding box via an ROI pooling layer and several fully connected (FC) layers, similar to the Faster R-CNN framework [168]. Secondly, after obtaining region classification scores, we use a region-to-pixel layer to transform the region scores to pixel scores following [26]:

$$s_{j,actor} = \max_{i=1,\ldots,N, j\in\mathbf{b}_i} (m_{i,j} \times s_{i,actor}), \tag{5.2}$$

$$s_{j,action} = \max_{i=1,\ldots,N, j\in\mathbf{b}_i} (m_{i,j} \times s_{i,action}), \tag{5.3}$$

where j refers to an image pixel, $m_{i,j}$ refers to the probabilities that the pixel j belonging to the mask i, $s_{i,actor}$ and $s_{i,action}$ refer to the classification scores of mask

i for actor and action categories, respectively. This region-to-pixel layer serves to resolve labeling conflicts when a pixel belongs to multiple region masks, i.e., we take the max among them to calculate the fused score. The resulting pixel scores $s_{j,actor}$ and $s_{j,action}$ are then passed to a softmax layer to output class probabilities.

5.2.4 Network Training

As the FC layers in our network contain quite a large number of parameters, instead of training the whole network end-to-end, we adopt a two-stage training procedure to accelerate the training. We implement our approach using PyTorch [169]. Some implementation details and parameter settings are stated in Section 5.5.1.

We initialize the front-end network with the pretrained model from [163]. At the first stage, we train the front-end with a light-weight back-end network with fewer parameters. We train our front-end network with the baseline back-end shown in Fig. 5.2(b) using stochastic gradient descent (SGD). The learning rates for the front-end and back-end networks are set to 1×10^{-4} and 2×10^{-3}. We use a mini-batch size of 10 and train the network for 20000 iterations.

At the second stage, we fix the parameters of the front-end network and train our back-end. Note that only the fully connected layers are learn-able. Region pooling and region-to-pixel layers do not contribute to additional parameters. We use a learning rate of 2.5×10^{-4} and a smaller mini-batch size of 1 to train the network for 80000 iterations.

5.3 Experiments

5.3.1 Dataset Description and Experimental Protocols

In our experiments, we have several objectives. Firstly, we demonstrate the effectiveness of our approach by comparing it with the DeepLab type baseline, as illustrated

	global pixel accuracy			mean class accuracy			mean class IoU		
	actor	action	actor-action	actor	action	actor-action	actor	action	actor-action
Baseline (RGB only)	**95.7**	**93.5**	**93.0**	77.0	58.5	42.7	**67.1**	46.0	32.1
Ours (RGB only)	95.0	92.9	92.5	**85.5**	**68.8**	**51.5**	67.0	**48.1**	**34.5**
Baseline (RGB + flow)	**95.8**	**93.9**	**93.5**	78.0	61.7	46.1	**68.1**	49.1	34.9
Ours (RGB + flow)	95.3	93.4	93.0	**86.0**	**70.7**	**56.4**	**68.1**	**51.1**	**38.6**

Table 5.1: Comparison with a DeepLab-CNN [3] type baseline, which does not use region masks.

in Fig. 5.2(b), which does not utilize region masks. Secondly, we analyze the various factors that affect the performance, such as the inclusion of optical flow and using different mask proposals. Lastly, we demonstrate our method's performance in comparison with the current state of the arts.

We evaluate our method on the A2D dataset [32], which is the only dataset providing pixel-level dense annotation for actor and action pairs. The dataset can be downloaded from `http://web.eecs.umich.edu/~jjcorso/r/a2d`. It consists of 3782 videos which are labeled with both pixel-level actors (*i.e.*, adult, baby, ball, bird, car, cat, and dog) and actions (*i.e.*, climb, crawl, eat, fly, jump, roll, run, walk) for sparsely sampled frames. We use identical partition scheme as previous works [32–35] , i.e., 9561 frames for training and 2365 for testing. Some example images and annotations of the dataset are provided in Fig. 5.4.

We measure the performance using three metrics: (1) global accuracy measures the overall pixel classification accuracy; (2) mean per-class accuracy is obtained by first calculating classification accuracy for each class separately followed by averaging over all the classes; (3) mean per-class IOU is similar to mean per-class accuracy but evaluates the performance at pixel level using intersection over union (IOU). The global accuracy is dominated by frequently occurring objects, while the other two metrics bias towards rarely occurring classes. We calculate these metrics in three settings: actor, action and joint actor-action. Unless otherwise specified, we report the performance using both RGB and optical-flow images, and using region masks from FCIS model trained on the A2D dataset.

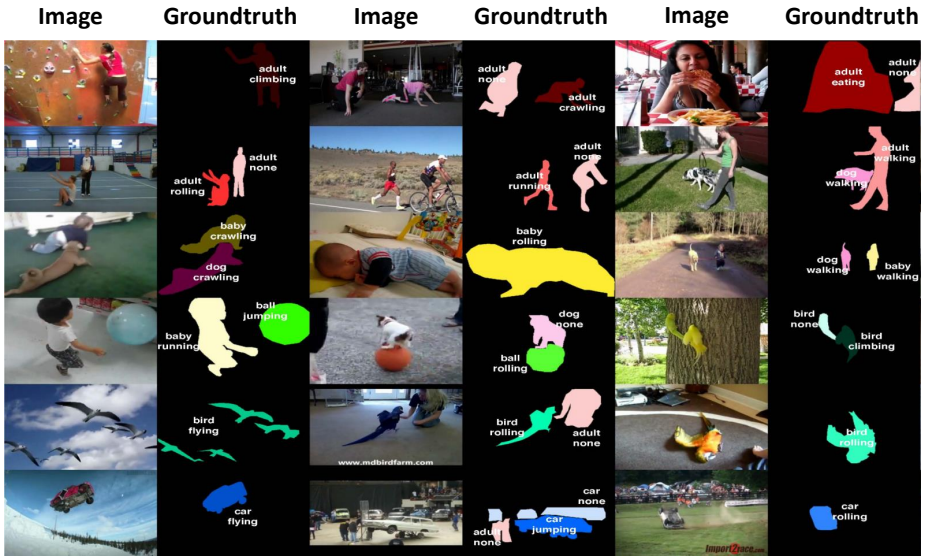


Figure 5.4: Sample images and annotations from the A2D dataset. Figure reproduced from the authors' website [6]. Reproduced with permission.

	global pixel accuracy			mean class accuracy			mean class IoU		
	actor	action	actor-action	actor	action	actor-action	actor	action	actor-action
Baseline (Non-Boundary)	**97.7**	**96.3**	**95.8**	85.1	67.8	51.6	**75.6**	53.9	39.0
Ours (Non-Boundary)	97.3	96.0	95.7	**88.9**	**73.6**	**57.8**	73.3	**56.1**	**41.8**

Table 5.2: Comparison with a DeepLab-CNN [3] type baseline for the image regions excluding object boundaries.

5.3.2 Main Results

5.3.2.1 Usefulness of actor region masks

Table 5.1 shows the results of our method compared to a DeepLab type baseline which does not use region masks. For the global pixel accuracy metric, our approach performs slightly worse than the baseline. This may be caused by the resolution of region masks generated by our FCIS model being only the half of the baseline output, which leads to boundaries inaccuracies. Our method performs around 10% better than the baseline in mean class accuracy for all the three settings, and around 3% better in mean class IOU for the action and actor-action settings. For the latter two metrics, the performance boost mainly comes from two situations. Firstly, for some actions like adult-jumping and bird-rolling which are prominently reflected by certain body parts, an actor region is enforced to have consistent labeling with region masks, while the baseline tends to segment the actor region into several parts of different action labels (see Fig. 5.5(a)). Secondly, small objects may be missed by the baseline because of its inflexible selection of receptive fields (see Fig. 5.5(b)). In contrast, the region masks used by our method can provide adaptive receptive fields according to object sizes. In Table 5.5, the performance of adult-jumping and bird-rolling and the performance of small objects such as bird and ball are significantly improved by our approach, which justifies our analysis. Fig. 5.5(c) shows examples that both the region masks and optical flow can improve the results. Fig. 5.5(d) shows a failure case of our method due to inaccurate region masks.

It is well known that incorporating region masks are particularly beneficial for segmentation of object boundaries [25,26]. To find out how much of the performance gain can be attributed to resolving inconsistent labelling results rather than better handling of boundaries, we perform an additional evaluation for the non-boundary regions only. We define the boundary region as a narrow band around the ground

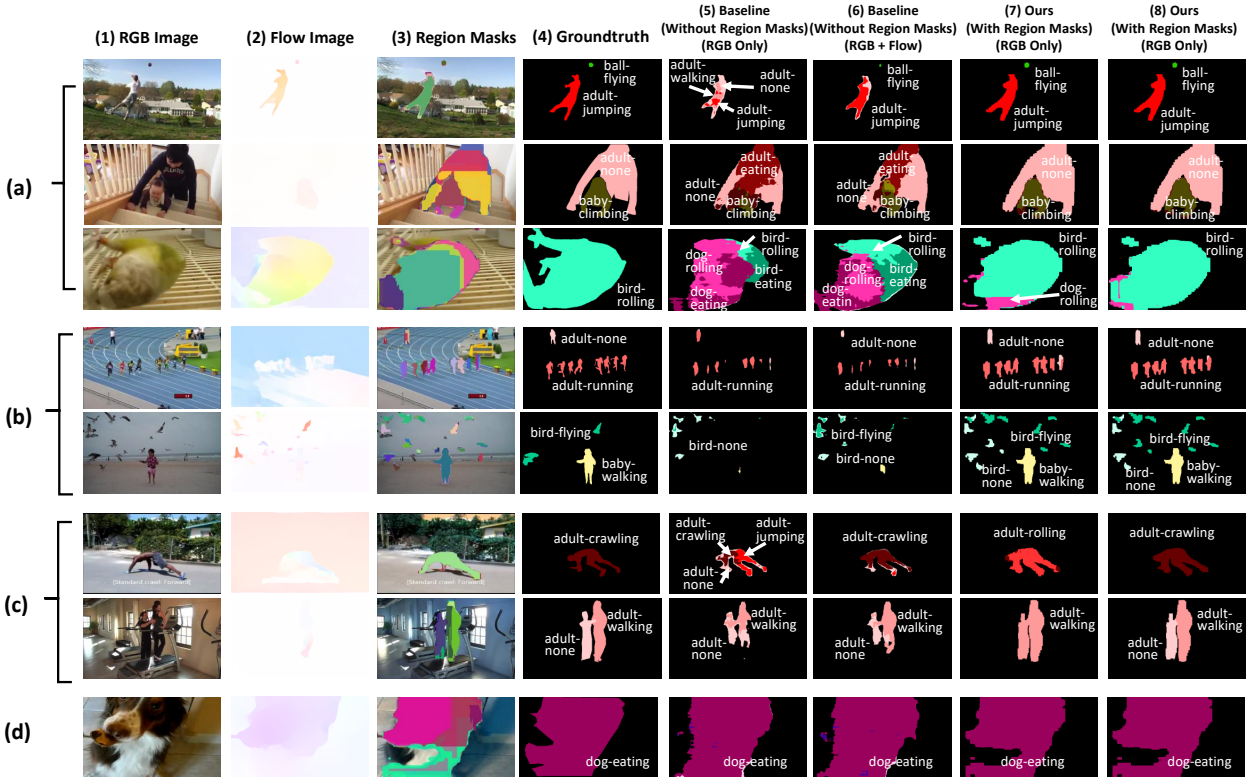


Figure 5.5: Examples of the actor-action semantic segmentation for our method contrasted against a DeepLab-CNN [3] type baseline which does not use region masks. (a) Examples that demonstrate our method yields more coherent segmentations for challenging poses or actions. (b) Examples that illustrate our method better captures small objects. (c) Examples that show both the region masks and optical flow can improve the results. (d) A failure example due to inaccurate region masks.

truth label changes, and the width of this band is set to 15 pixels. Non-boundary region refers to rest of the image. Table 5.2 shows that compared with the baseline, our method performs around 6% and 3% better in mean class accuracy and mean class IOU for the actor-action settings respectively. In addition, the improvements are more notable for action categories compared with actor categories, which is expected as our method is designed for resolving labeling inconsistency issue that is more server in action categories rather than actor categories.

region masks (training)	region masks (testing)	global pixel accuracy			mean class accuracy			mean class IoU		
		actor	action	actor-action	actor	action	actor-action	actor	action	actor-action
COCO + GT	COCO	94.7	93.0	92.6	82.4	69.4	54.1	65.1	50.9	37.9
A2D + GT	A2D	**95.3**	**93.4**	**93.0**	**86.0**	**70.7**	**56.4**	**68.1**	**51.1**	**38.6**
A2D + GT	GT	98.1	96.3	95.9	87.7	72.5	57.7	80.9	59.1	45.6

Table 5.3: Effects of actor region masks quality. Region masks generated from FCIS model trained on COCO and A2D datasets are denoted as "COCO" and "A2D" respectively. Ground truth region masks are denoted as "GT". The top two rows show the results using FCIS generated masks, while the last row shows the results when ground truth region masks are applied during the testing.

	global pixel accuracy			mean class accuracy			mean class IoU		
	actor	action	actor-action	actor	action	actor-action	actor	action	actor-action
[32]	-			47.4	49.4	30.5	-		
[33]	85.2	85.3	84.2	61.2	60.5	43.9	-		
[34]	83.8	83.1	81.7	-	-	41.7	-		
[35]	90.6	89.3	88.7	73.7	61.4	48.0	49.5	42.2	29.7
[71]	-	-	**93.0**	60.0	59.9	45.0	-	-	33.4
Ours	**95.3**	**93.4**	**93.0**	**86.0**	**70.7**	**56.4**	**68.1**	**51.1**	**38.6**

Table 5.4: Comparison with the previous works. When multiple results are given, the best one for each metric is displayed.

5.3.2.2 Influence of the quality of region masks

We train two FCIS models on A2D and COCO datasets respectively for actor region mask generation. The generated masks are fed as inputs to our segmentation network at the training or testing time. As suggested by [26], additionally we use ground truth actor region masks during the training, as they may be important for small objects that are not tightly covered by FCIS generated region masks.

The top two rows of Table 5.3 show the results of our method using FCIS generated actor region masks at the testing time. We find that our approach using region masks from the FCIS model trained on A2D consistently performs better than COCO in all the three metrics, especially in the actor setting with performance improvement of around 4% in mean class accuracy and 3% in mean class IOU. These results demonstrate that high-quality, dataset-specific candidate region masks contribute to the performance improvements.

The last row shows the results when ground truth actor region masks are used at

the testing time. It is an upper bound performance by assuming perfect instance segmentation results, and hence the errors can only be attributed to incorrect actor-action semantic segmentation. While the improvements are very significant compared with FCIS generated masks (*e.g.*, more than 7% in mean class IOU for all the three settings), the performance may still be unsatisfactory (*e.g.*, less than 50% in the mean class IOU for the actor-action setting). Therefore, it is useful to research on both actor-action semantic segmentation and instance segmentation methods to further improve the results.

5.3.2.3 Influence of incorporating optical flow

From Table 5.5, we see optical flow can significantly improve the performance of some fast-moving actor-action categories. For example, the per-category accuracies of adult-jumping, car-running, ball-flying, cat-running, and dog-jumping are increased by more than 10%. On average, our method with optical flow can improve the performance in both mean class accuracy and mean class IOU by more than 4% in the actor-action setting (see Table 5.1). These results suggest optical flow can effectively help improve our performance.

5.3.2.4 Comparison with Previous Works

Table 5.4 shows the results of our method and earlier works. Our approach outperforms the state-of-the-art techniques by more than 8% in mean class accuracy and more than 5% in mean class IOU. It should be noted that the network architecture and pretrained model used in our method is different from the ones used by [32–35, 71]. Hence, to validate the effectiveness of our method, we compare it with DeepLab baseline and conduct an ablation study (see Tables 5.1 , 5.2, 5.3 and 5.5). These are carried out in controlled settings to ensure a fair comparison. The results demonstrate that specific designs (exploiting region masks, optical flow, and

		adult								baby				
method	BG	climbing	crawling	eating	jumping	rolling	running	walking	none	climbing	crawling	rolling	walking	none
Baseline (RGB only)	98.1	49.0	71.9	91.7	22.9	34.1	31.0	61.7	51.1	68.8	59.3	74.1	55.6	16.4
Baseline (RGB + flow)	**98.2**	53.4	77.7	89.9	31.5	36.5	32.2	**64.6**	51.5	69.3	69.2	73.5	58.7	20.3
Ours (RGB only)	96.2	**76.7**	76.3	**95.5**	27.6	**59.1**	49.5	56.4	62.2	83.0	75.3	**88.6**	57.5	8.9
Ours (RGB + flow)	96.4	73.2	**79.7**	93.1	**43.0**	48.1	**53.0**	61.1	**66.1**	**83.3**	**77.3**	86.3	**61.0**	**37.0**

	ball				bird							car				
method	flying	jumping	rolling	none	climbing	eating	flying	jumping	rolling	walking	none	flying	jumping	rolling	running	none
Baseline (RGB only)	0.8	2.6	49.5	6.5	57.9	45.9	60.7	5.6	43.5	46.0	25.4	28.6	80.2	42.9	44.8	50.6
Baseline (RGB + flow)	2.0	7.6	53.9	12.8	62.4	45.5	61.1	9.9	46.6	55.0	**28.5**	29.2	84.6	45.5	56.6	46.3
Ours (RGB only)	5.4	28.6	65.8	12.6	**68.3**	48.6	**81.6**	**29.7**	57.9	54.5	3.0	**53.0**	94.3	58.9	56.2	37.5
Ours (RGB + flow)	**26.7**	**30.8**	**81.1**	**18.5**	63.2	**50.5**	80.2	22.2	**65.0**	**61.9**	3.0	34.9	**97.9**	**62.3**	**75.7**	**62.9**

	cat							dog						**Avg**
method	climbing	eating	jumping	rolling	running	walking	none	crawling	eating	jumping	rolling	running	walking	-
Baseline (RGB only)	48.8	65.5	19.2	51.1	14.5	41.5	4.4	42.2	80.2	5.5	38.9	6.8	71.8	42.7
Baseline (RGB + flow)	53.1	64.7	15.8	49.2	20.3	52.3	**8.1**	**51.9**	**82.9**	13.2	47.8	8.4	**74.8**	46.1
Ours (RGB only)	58.2	74.6	**32.8**	**82.1**	34.6	37.0	4.0	46.1	77.9	8.8	48.4	19.5	70.3	51.5
Ours (RGB + flow)	**62.8**	**75.7**	30.9	75.7	**46.1**	**49.4**	6.7	49.2	81.3	**35.7**	**61.9**	**27.0**	71.0	**56.4**

Table 5.5: Per-category accuracy result.

quality of region masks) in our method contribute to the performance gains.

5.3.3 Computational Complexity

Our testing platform is a server with Intel Xeon 2.4GHZ CPU and Nvidia K80 GPU. We limit the hardware usage to single CPU and GPU. The optical flow computation using FlowNet [166], region mask calculation using FCIS [4] and our region-based segmentation network inference take around 350ms, 300ms and 450ms for each frame, respectively. Hence the overall computational time of the proposed approach is around 1100ms per frame. In contrast, the overall testing time of the baseline DeepLab-type segmentation approach is around 395ms per frame inclusive of optical flow computation. We note that certain components of our network, *e.g.*, region-to-pixel layer, could be further optimized with a native CUDA implementation. As a future direction, we are also interested in integrating actor-action semantic segmentation with optical flow estimation and region mask generation into one unified

deep network for better efficiency and accuracy.

5.4 Conclusion

In this chapter, we propose an end-to-end region-based actor-action segmentation approach. We design a deep network consisting of a two-stream front-end and a region-based actor-action segmentation back-end The two-stream front-end learns two types of features: one for appearance and the other for motion. The fused features from the two-stream front-end together with region masks are passed to the back-end for actor-action labeling. We validate the effectiveness of our approach on the A2D dataset. Experiment results show that both the region-based segmentation strategy and fused features from the two-stream front-end contribute to performance improvements.

5.5 Appendix

5.5.1 Parameter Settings

In our backend module, we need to calculate classification scores of each region mask. To do this, we classify the bounding box of each region mask via an ROI pooling layer and several fully connected (FC) layers, similar to the Faster R-CNN framework [168]. The network settings regarding this part are as follows: the ROI pooling layer produces a set of small feature maps of size 7×7. These small feature maps are used as the inputs of three FC layers to generate bounding box classification scores. The first two FC layers have numbers of input and output channels being 50176 and 25088, 25088 and 25088, respectively, while for the last FC layer, the number of input channel is 20588, and the number of output channel equals the number of output classes.

	First Stage Training (frontend + baseline backend)	Second Stage Training (our backend)
Optimization Solver:	SGD	SGD
Learning Rate:	1e-4 (frontend) 2e-3 (backend)	2.5e-4 (backend)
Learning Rate Schedule:	polynomial decay, power = 0.9	constant
Momentum:	0.9	0.9
Weight Decay:	5e-4	5e-4
Batch Size:	10	1
Number of Iterations:	20000	80000

Table 5.6: Network training parameters used in experiments.

As mentioned in Section 5.2.4, we adopt a two-stage training procedure; the training parameter settings for both stages are shown in Table 5.6.

Chapter 6

Conclusion and Future Work

6.1 Conclusions

In summary, this thesis explores the problem of semantic segmentation in both images and videos. With the contributions listed below, it demonstrates that visual context and consistency are effective cues to improve the performance of semantic segmentation.

To validate the usefulness of spatial layout context for semantic segmentation in images, Chapter 3 proposes to fuse location and appearance information for image parsing by local learning. We train each local classifier with pixels from its neighborhood region only, so it better fits the local appearance distribution and can be more discriminative. We analyze the bias-variance trade-off of the proposed local classifier and indicate the importance of selecting an appropriate neighborhood size for handling the spatial misalignments. On three different tasks of horse segmentation, pedestrian parsing, and scene parsing, our local learning can significantly improve the performance compared with existing methods that rely on one unified classifier.

To demonstrate the utility of spatio-temporal consistency for semantic segmenta-

tion in videos, Chapter 4 proposes an efficient online video filtering method, named adaptive exponential smoothing (AES), to refine pixel classification maps. Our AES performs adaptive filtering for each pixel with a different spatio-temporal filtering path, so it can better improve the "flickering" maps while avoiding over-smoothing, and better tolerate fast object movement and camera motion. The experimental evaluations on saliency map filtering and multi-class scene parsing map filtering validate the superiority of the proposed method compared with average and exponential filters, as well as alternative approaches that rely on optical flow computations. Thanks to the proposed dynamic programming algorithm for AES, our filtering method has linear time complexity and is suitable for real-time applications.

To demonstrate the usefulness of region consistency for actor-action semantic segmentation in videos, Chapter 5 proposes an end-to-end region-based actor-action segmentation approach. We utilize high-quality region masks from an instance segmentation algorithm, and enforce pixels inside the region mask to take the same action label, so that different body parts may have consistent labeling of actor-action category. Our approach also includes a two-stream network which captures both appearance and motion information. Experiment results show that both the region-based segmentation strategy and fused features from the two-stream network contribute to performance improvements, which lead to significantly better performance compared with state-of-the-art methods.

6.2 Future Works

Some suggestions for extending the presented approaches are listed in below.

- *Exploiting Layout Constraint for 3D LiDAR PointCloud with Local Classifiers.* In Chapter 3, we have studied how to exploit spatial layout of an image with a field of local classifiers. LiDAR point cloud is commonly used in autonomous

driving system, and it is useful to extend this study to point cloud as well. One difficulty with point cloud processing is that local density of LiDAR points varies a lot depending on the distance with respect to the LiDAR sensor. For example, in the far away region, the reflected points can be very sparse, while they are more dense around the LiDAR sensor.

To perform semantic segmentation on LiDAR point cloud, *i.e.*, assign each LiDAR point a semantic label, we first extract the corresponding local LiDAR feature using information from the LiDAR point neighborhood. However, due to such LiDAR points density variations, the extracted local LiDAR feature will have quite different characteristics depending on the distance with respect to the LiDAR sensor. This might cause difficulties if we train a single unified classifier to label all the LiDAR points for the whole point cloud, hence it is useful to learn local classifiers separately for different distances.

In addition, according to our previous work [170], height of an object with respect to the ground plane can be a useful feature to do scene labelling. For example, in the indoor scene, a computer is possibly placed on a table, chairs are usually placed on the floor, while fans are commonly attached to the room ceiling. Height is quite invariant to the distance and viewpoint changes. For such a reason, it might also worth trying to train separate local classifiers for different heights with respect to the ground plane.

- *Learning-based approaches for pixel classification map smoothing in videos.* In Chapter 4, we have applied several heuristics in the proposed adaptive exponential smoothing. For example, the input probabilistic scores need to be converted to discriminative scores by subtracting a small offset; a predefined similarity kernel with fixed weights is used for appearance modeling. It would be interesting to explore learning-based algorithms to replace such steps. For

example, we can learn convolution filters to convert the input scores to discriminative scores that are more suitable to our formulation and the underlying data. A more sophisticated and yet efficient learning-based appearance modeling technique may also be applied to further enhance the filtering performance. Related works such as long short-term memory [171], boundary flow siamese network [172], and video propagation network [173] are potentially useful in this direction of research.

- *Efficient actor-action semantic segmentation with region masks.* In Chapter 5, our approach requires outputs from other computationally expensive algorithms, such as optical flow estimations from FlowNet [166] and region masks from fully convolution instance segmentation (FCIS) algorithm [4]. While such a decoupled design provides more flexibility, it also leads to potentially higher computational costs and worse performance. As a future direction, we are interested in integrating these different algorithms into one system. For example, we can modify the original FCIS network by adding another branch to perform actor-action semantic segmentation. In this way, both instance segmentation and actor-action semantic segmentation can share the same set of features for better efficiency. Also, region mask proposals can be learned together for better accuracy.

Author's Publications

1. Kang Dang, Chunluan Zhou, Zhigang Tu, Michael Hoy, Justin Dauwels, Junsong Yuan, "Actor-Action Semantic Segmentation with Region Masks", in *Proc. British Machine Vision Conference (BMVC)*, 2018

2. Michael Hoy, Zhigang Tu, Kang Dang, Justin Dauwels, "Learning to Predict Pedestrian Intention via Variational Tracking Networks", *in Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018

3. Kang Dang, Junsong Yuan, "Learning location constrained pixel classifiers for image parsing," in *Journal of Visual Communication and Image Representation (JVCI)*, 2017

4. Kang Dang, Michael Hoy, Justin Dauwels, Junsong Yuan, "Real-time hierarchical fusion system for semantic segmentation in offroad scenes," in *Proc. International Conference on Information Fusion (FUSION)*, 2017

5. Kang Dang, Jiong Yang, Junsong Yuan, "Adaptive Exponential Smoothing for Online Filtering of Pixel Prediction Maps," in *Proc. International Conference on Computer Vision (ICCV)*, 2015

6. Kang Dang, Junsong Yuan, "Location Constrained Pixel Classifiers for Image Parsing with Regular Spatial Layout," in *Proc. British Machine Vision Conference (BMVC)*, 2014

7. Gangqiang Zhao, Junsong Yuan, Kang Dang, "Height Gradient Histogram (HIGH) for 3D Scene Labeling," in *Proc. International Conference on 3D Vision (3DV)*, 2014

8. Kang Dang, Junsong Yuan, Ho Yee Tiong, "Voxel labelling in CT images with data-driven contextual features," in *Proc. International Conference on Image Processing (ICIP)*, 2013

Bibliography

[1] Ping Luo, Xiaogang Wang, and Xiaoou Tang, "Pedestrian parsing via deep decompositional network," in *ICCV*, 2013.

[2] Fisher Yu and Vladlen Koltun, "Multi-scale context aggregation by dilated convolutions," in *ICLR*, 2016.

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *arXiv:1606.00915*, 2016.

[4] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei, "Fully convolutional instance-aware semantic segmentation," *arXiv preprint arXiv:1611.07709*, 2016.

[5] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

[6] Jason J Corso, "A2d: A dataset and benchmark for action recognition and segmentation with multiple classes of actors," Oct. 2017.

[7] Kang Dang and Junsong Yuan, "Location constrained pixel classifiers for image parsing with regular spatial layout," in *BMVC*, 2014.

[8] Kang Dang and Junsong Yuan, "Learning location constrained pixel classifiers for image parsing," *Journal of Visual Communication and Image Representation*, 2017.

[9] Yihang Bo and Charless C Fowlkes, "Shape-based pedestrian parsing," in *CVPR*, 2011.

[10] Ondrej Miksik, Daniel Munoz, J Andrew Bagnell, and Martial Hebert, "Efficient temporal consistency for streaming video scene analysis," in *ICRA*, 2013.

[11] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi, "Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *ECCV*. 2006.

[12] Kang Dang, Junsong Yuan, and Ho Yee Tiong, "Voxel labelling in ct images with data-driven contextual features," in *ICIP*, 2013.

[13] Joseph Tighe and Svetlana Lazebnik, "Superparsing," *International Journal of Computer Vision*, 2013.

[14] Daniel Munoz, J Andrew Bagnell, and Martial Hebert, "Stacked hierarchical labeling," in *ECCV*, 2010.

[15] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia, "Pyramid scene parsing network," *arXiv:1612.01105*, 2016.

[16] Oge Marques, Elan Barenholtz, and Vincent Charvillat, "Context modeling in computer vision: techniques, implications, and applications," *Multimedia Tools and Applications*, 2011.

[17] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik, "Hypercolumns for object segmentation and fine-grained localization," *arXiv:1411.5752*, 2014.

[18] Vladlen Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *NIPS*, 2011.

[19] Pushmeet Kohli, Philip HS Torr, et al., "Robust higher order potentials for enforcing label consistency," *International Journal of Computer Vision*, 2009.

[20] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr, "Conditional random fields as recurrent neural networks," in *CVPR*, 2015.

[21] Abhijit Kundu, Vibhav Vineet, and Vladlen Koltun, "Feature space optimization for semantic video segmentation," in *CVPR*, 2016.

[22] Jonathan T Barron and Ben Poole, "The fast bilateral solver," in *ECCV*, 2016.

[23] Zhuowen Tu and Xiang Bai, "Auto-context and its application to high-level vision tasks and 3d brain image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[24] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid, "Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation," *arXiv:1611.06612*, 2016.

[25] Yang He, Wei-Chen Chiu, Margret Keuper, Mario Fritz, and Saarland Informatics Campus, "Std2p: Rgbd semantic segmentation using spatio-temporal data-driven pooling," in *CVPR*, 2017.

[26] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari, "Region-based semantic segmentation with end-to-end training," in *ECCV*, 2016.

[27] Jianxiong Xiao and Long Quan, "Multiple view semantic segmentation for street view images," in *ICCV*, 2009.

[28] Ce Liu, Jenny Yuen, and Antonio Torralba, "Nonparametric scene parsing via label transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[29] Ayelet Akselrod-Ballin, Meirav Galun, Moshe John Gomori, Ronen Basri, and Achi Brandt, "Atlas guided identification of brain structures by combining 3d segmentation and svm classification," in *MICCAI*. 2006.

[30] Zhuowen Tu and Arthur W Toga, "Towards whole brain segmentation by a hybrid model," in *MICCAI*. 2007.

[31] Wei Yang, Ping Luo, and Liang Lin, "Clothing co-parsing by joint image segmentation and labeling," in *CVPR*, 2014.

[32] Chenliang Xu, Shao-Hang Hsieh, Caiming Xiong, and Jason J Corso, "Can humans fly? action understanding with multiple classes of actors," in *CVPR*, 2015.

[33] Chenliang Xu and Jason J Corso, "Actor-action semantic segmentation with grouping process models," in *CVPR*, 2016.

[34] Yan Yan, Chenliang Xu, Dawen Cai, and Jason Corso, "Weakly supervised actor-action segmentation via robust multi-task ranking," in *CVPR*, 2017.

[35] Vicky Kalogeiton, Philippe Weinzaepfel, Vittorio Ferrari, and Cordelia Schmid, "Joint learning of object and action detectors," in *ICCV*, 2017.

[36] Jianbo Shi and Jitendra Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, 2000.

[37] Pedro F Felzenszwalb and Daniel P Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, 2004.

[38] Andrea Vedaldi and Stefano Soatto, "Quick shift and kernel methods for mode seeking," in *European Conference on Computer Vision*, 2008.

[39] Alex Levinshtein, Adrian Stere, Kiriakos N Kutulakos, David J Fleet, Sven J Dickinson, and Kaleem Siddiqi, "Turbopixels: Fast superpixels using geometric flows," *IEEE transactions on pattern analysis and machine intelligence*, 2009.

[40] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk, "Slic superpixels," *EPFL Technical Report*, 2010.

[41] Joao Carreira and Cristian Sminchisescu, "Constrained parametric min-cuts for automatic object segmentation," in *CVPR*, 2010.

[42] Koen EA Van de Sande, Jasper RR Uijlings, Theo Gevers, and Arnold WM Smeulders, "Segmentation as selective search for object recognition," in *ICCV*, 2011.

[43] Jordi Pont-Tuset, Pablo Arbelaez, Jonathan T Barron, Ferran Marques, and Jitendra Malik, "Multiscale combinatorial grouping for image segmentation and object proposal generation," *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[44] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollár, "Learning to segment object candidates," in *NIPS*, 2015.

[45] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, "Mask r-cnn," *arXiv:1703.06870*, 2017.

[46] Albert Montillo, Jamie Shotton, John Winn, Juan Eugenio Iglesias, Dimitri Metaxas, and Antonio Criminisi, "Entangled decision forests and their application for semantic segmentation of ct images," in *International Conference on Information Processing in Medical Imaging*, 2011.

[47] Peter Kontschieder, Pushmeet Kohli, Jamie Shotton, and Antonio Criminisi, "Geof: Geodesic forests for learning coupled predictors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 65–72.

[48] João Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu, "Semantic segmentation with second-order pooling," in *ECCV*. 2012.

[49] Zhuowen Tu, "Auto-context and its application to high-level vision tasks," in *CVPR*, 2008.

[50] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[51] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv:1511.00561*, 2015.

[52] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," *arXiv:1511.02680*, 2015.

[53] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation.," in *CVPR*, 2017.

[54] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[55] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[56] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[57] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, "Densely connected convolutional networks.," in *CVPR*, 2017.

[58] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[59] Wei Liu, Andrew Rabinovich, and Alexander C Berg, "Parsenet: Looking wider to see better," *arXiv preprint arXiv:1506.04579*, 2015.

[60] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia, "Icnet for real-time semantic segmentation on high-resolution images," *arXiv preprint arXiv:1704.08545*, 2017.

[61] Iasonas Kokkinos, "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory.," in *CVPR*, 2017.

[62] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen-Change Loy, and Xiaoou Tang, "Semantic image segmentation via deep parsing network," in *CVPR*, 2015.

[63] Guosheng Lin, Chunhua Shen, Anton Van Den Hengel, and Ian Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *CVPR*, 2016.

[64] Anurag Arnab, Sadeep Jayasumana, Shuai Zheng, and Philip HS Torr, "Higher order conditional random fields in deep neural networks," in *ECCV*, 2016.

[65] Rémi Cuingnet, Raphael Prevost, David Lesage, Laurent D Cohen, Benoît Mory, and Roberto Ardon, "Automatic detection and segmentation of kidneys in 3d ct images using random forests," in *MICCAI*. 2012.

[66] Xue Bai, Jue Wang, David Simons, and Guillermo Sapiro, "Video snapcut: robust video object cutout using localized classifiers," in *ACM Transactions on Graphics*, 2009.

[67] Minglun Gong and Li Cheng, "Foreground segmentation of live videos using locally competing 1svms," in *CVPR*, 2011.

[68] Kang Dang and Junsong Yuan, "Location constrained pixel classifiers for image parsing with regular spatial layout," in *BMVC*, 2014.

[69] Darko Štern, Thomas Ebner, and Martin Urschler, "From local to global random regression forests: Exploring anatomical landmark localization," in *MICCAI*, 2016.

[70] Shu Liu, Xiaojuan Qi, Jianping Shi, Hong Zhang, and Jiaya Jia, "Multi-scale patch aggregation (mpa) for simultaneous detection and segmentation," in *CVPR*, 2016.

[71] Zhaofan Qiu, Ting Yao, and Tao Mei, "Learning deep spatio-temporal dependence for semantic video segmentation," *IEEE Transactions on Multimedia*, 2018.

[72] Kang Dang and Junsong Yuan, "Actor-action semantic segmentation with region masks," in *BMVC*, 2018.

[73] Anestis Papazoglou and Vittorio Ferrari, "Fast object segmentation in unconstrained video," in *ICCV*, 2013.

[74] Wenguan Wang, Jianbing Shen, and Fatih Porikli, "Saliency-aware geodesic video object segmentation," in *CVPR*, 2015.

[75] Dong Zhang, Omar Javed, and Mubarak Shah, "Video object segmentation through spatially accurate and temporally dense extraction of primary object regions," in *CVPR*, 2013.

[76] Suyog Dutt Jain and Kristen Grauman, "Supervoxel-consistent foreground propagation in video," in *ECCV*. 2014.

[77] Du Tran, Junsong Yuan, and David Forsyth, "Video event detection: From subvolume localization to spatiotemporal path search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[78] Robert Goodell Brown, *Smoothing, forecasting and prediction of discrete time series*, Courier Corporation, 2004.

[79] Pei-Lun Hsieh, Chongyang Ma, Jihun Yu, and Hao Li, "Unconstrained real-time facial performance capture," in *CVPR*, 2015.

[80] Yale Song, David Demirdjian, and Randall Davis, "Continuous body and hand gesture recognition for natural human-computer interaction," *ACM Transactions on Interactive Intelligent Systems*, 2012.

[81] Andreas Baak, Meinard Müller, Gaurav Bharaj, Hans-Peter Seidel, and Christian Theobalt, "A data-driven approach for real-time full body pose reconstruction from a depth camera," in *Consumer Depth Cameras for Computer Vision*. 2013.

[82] Manuel Lang, Oliver Wang, Tunç Ozan Aydin, Aljoscha Smolic, and Markus H Gross, "Practical temporal consistency for image-based graphics applications," *ACM Transactions on Graph*, 2012.

[83] Nicolas Bonneel, James Tompkin, Kalyan Sunkavalli, Deqing Sun, Sylvain Paris, and Hanspeter Pfister, "Blind video temporal consistency," *ACM Transactions on Graphics (TOG)*, 2015.

[84] Xinchen Yan, Junsong Yuan, Hui Liang, and Liqing Zhang, "Efficient online spatio-temporal filtering for video event detection," *ECCV Workshops*, 2014.

[85] Antonio Hernández-Vela, Nadezhda Zlateva, Alexander Marinov, Miguel Reyes, Petia Radeva, Dimo Dimov, and Sergio Escalera, "Graph cuts optimization for multi-limb human segmentation in depth maps," in *CVPR*, 2012.

[86] Andreas Ess, Tobias Mueller, Helmut Grabner, and Luc J Van Gool, "Segmentation-based urban traffic scene understanding.," in *BMVC*, 2009.

[87] Albert YC Chen and Jason J Corso, "Temporally consistent multi-class video-object segmentation with the video graph-shifts algorithm," in *WACV*, 2011.

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

[89] Jiangmin Jiang and Xiaonan Song, "An optimized higher order crf for automated labeling and segmentation of video objects," *IEEE Transactions on Circuits and Systems for Video Technology*, 2016.

[90] Ozan Şener, Kemal Ugur, and A Aydın Alatan, "Efficient mrf energy propagation for video segmentation via bilateral filters," *IEEE Transactions on Multimedia*, 2014.

[91] Zhongyu Lou and Theo Gevers, "Extracting primary objects by video co-segmentation," *IEEE Transactions on Multimedia*, 2014.

[92] David Nilsson and Cristian Sminchisescu, "Semantic video segmentation by gated recurrent flow propagation," *arXiv preprint arXiv:1612.08871*, 2016.

[93] Raghudeep Gadde, Varun Jampani, and Peter V. Gehler, "Semantic video cnns through representation warping," in *ICCV*, 2017.

[94] Varun Jampani, Raghudeep Gadde, and Peter V Gehler, "Video propagation networks," in *CVPR*, 2017.

[95] Siddhartha Chandra, Camille Couprie, and Iasonas Kokkinos, "Deep spatio-temporal random fields for efficient video segmentation," in *CVPR*, 2018.

[96] Camille Couprie, Clément Farabet, Laurent Najman, and Yann Lecun, "Convolutional nets and watershed cuts for real-time semantic labeling of rgbd videos," *Journal of Machine Learning Research*, 2014.

[97] Chenliang Xu and Jason J Corso, "Evaluation of super-voxel methods for early video processing," in *CVPR*, 2012.

[98] Yiliang Xu, Dezhen Song, and Anthony Hoogs, "An efficient online hierarchical supervoxel segmentation algorithm for time-critical applications," in *BMVC*, 2014.

[99] Chenliang Xu, Caiming Xiong, and Jason J Corso, "Streaming hierarchical video segmentation," in *ECCV*. 2012.

[100] Fabio Galasso, Margret Keuper, Thomas Brox, and Bernt Schiele, "Spectral graph reduction for efficient image and streaming video segmentation," in *CVPR*, 2014.

[101] Juho Lee, Suha Kwak, Bohyung Han, and Seungjin Choi, "Online video segmentation by bayesian split-merge clustering," in *ECCV*. 2012.

[102] Sylvain Paris, "Edge-preserving smoothing and mean-shift segmentation of video streams," in *ECCV*. 2008.

[103] Tinghuai Wang and John Collomosse, "Probabilistic motion diffusion of labeling priors for coherent video segmentation," *IEEE Transactions on Multimedia*, 2012.

[104] Paul Sturgess, Karteek Alahari, Lubor Ladicky, and Philip HS Torr, "Combining appearance and structure from motion features for road scene understanding.," in *BMVC*, 2009.

[105] Ce Liu, Jenny Yuen, Antonio Torralba, Josef Sivic, and William T Freeman, "Sift flow: Dense correspondence across different scenes," in *ECCV*. 2008.

[106] Jaemin Kim and John W Woods, "Spatio-temporal adaptive 3-d kalman filter for video," *IEEE Transactions on Image Processing*, 1997.

[107] Mona Mahmoudi and Guillermo Sapiro, "Fast image and video denoising via nonlocal means of similar neighborhoods," *IEEE Signal Processing Letters*, 2005.

[108] Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang, "Learning blind video temporal consistency," *arXiv preprint arXiv:1808.00449*, 2018.

[109] James C Brailean, Richard P Kleihorst, Serafim Efstratiadis, Aggelos K Katsaggelos, and Reginald L Lagendijk, "Noise reduction filters for dynamic image sequences: A review," *Proceedings of the IEEE*, 1995.

[110] Kang Dang, Jiong Yang, and Junsong Yuan, "Adaptive exponential smoothing for online filtering of pixel prediction maps," in *ICCV*, 2015.

[111] Eran Borenstein and Shimon Ullman, "Class-specific, top-down segmentation," in *ECCV*. 2002.

[112] Leonid Sigal, Alexandru O Balan, and Michael J Black, "Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion," *International Journal of Computer Vision*, 2010.

[113] Gabriel J Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla, "Segmentation and recognition using structure from motion point clouds," in *ECCV*. 2008.

[114] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *CRCV-TR-12-01*, 2012.

[115] Bolei Zhou, Xiaodi Hou, and Liqing Zhang, "A phase discrepancy analysis of object motion," in *ACCV*. 2011.

[116] Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun, "Scene parsing with multiscale feature learning, purity trees, and optimal covers," in *ICML*, 2012.

[117] Christian Wojek, Stefan Roth, Konrad Schindler, and Bernt Schiele, "Monocular 3d scene modeling and inference: Understanding multi-object traffic scenes," in *ECCV*. 2010.

[118] Li-Li Wang and Nelson HC Yung, "Hybrid graphical model for semantic image segmentation," *Journal of Visual Communication and Image Representation*, 2015.

[119] M Freiman, A Kronman, SJ Esses, L Joskowicz, and J Sosna, "Non-parametric iterative model constraint graph min-cut for automatic kidney segmentation," in *MICCAI*. 2010.

[120] Thomas Leung and Jitendra Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *International Journal of Computer Vision*, 2001.

[121] David G Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004.

[122] Timo Ojala, Matti Pietikainen, and Topi Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[123] Song-Chun Zhu, Cheng-En Guo, Yizhou Wang, and Zijian Xu, "What are textons?," *International Journal of Computer Vision*, 2005.

[124] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, "LIBLINEAR: A library for large linear classification," *The Journal of Machine Learning Research*, 2008.

[125] Antonio Torralba, Kevin P Murphy, and William T Freeman, "Sharing features: efficient boosting procedures for multiclass object detection," in *CVPR*, 2004.

[126] Hongyuan Zhu, Fanman Meng, Jianfei Cai, and Shijian Lu, "Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation," *Journal of Visual Communication and Image Representation*, 2016.

[127] Yuri Boykov, Olga Veksler, and Ramin Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

[128] Vladimir Kolmogorov and Ramin Zabin, "What energy functions can be minimized via graph cuts?," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[129] Yuri Boykov and Vladimir Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[130] Geoffrey E Hinton, "Products of experts," in *Artificial Neural Networks, International Conference on*, 1999.

[131] Hamed Akbari and Baowei Fei, "Automatic 3d segmentation of the kidney in mr images using wavelet feature extraction and probability shape model," in *SPIE Medical Imaging*, 2012.

[132] Joseph Tighe and Svetlana Lazebnik, "Finding things: Image parsing with regions and per-exemplar detectors," in *CVPR*, 2013.

[133] Mojtaba Seyedhosseini and Tolga Tasdizen, "Scene labeling with contextual hierarchical models," *arXiv:1402.0595*, 2014.

[134] Jimei Yang, Simon Safar, and Ming-Hsuan Yang, "Max-margin boltzmann machines for object segmentation," in *CVPR*. IEEE, 2014.

[135] Fayao Liu, Guosheng Lin, and Chunhua Shen, "Crf learning with cnn features for image segmentation," *Pattern Recognition*, 2015.

[136] Jimei Yang, Brian Price, Scott Cohen, Zhe Lin, and Ming-Hsuan Yang, "Patchcut: Data-driven object segmentation via local shape transfer," in *CVPR*, 2015.

[137] Francesco Visin, Marco Ciccone, Adriana Romero, Kyle Kastner, Kyunghyun Cho, Yoshua Bengio, Matteo Matteucci, and Aaron Courville, "Reseg: A recurrent neural network-based model for semantic segmentation," in *CVPR Workshops*, 2016.

[138] Fayao Liu, Guosheng Lin, Ruizhi Qiao, and Chunhua Shen, "Structured learning of tree potentials in crf for image segmentation," *IEEE transactions on neural networks and learning systems*, 2017.

[139] Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek, "Semantic segmentation using adversarial networks," *arXiv preprint arXiv:1611.08408*, 2016.

[140] Yawei Luo, Zhedong Zheng, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang, "Macro-micro adversarial network for human parsing," *arXiv preprint arXiv:1807.08260*, 2018.

[141] Fangting Xia, Jun Zhu, Peng Wang, and Alan L Yuille, "Pose-guided human parsing by an and/or graph using pose-context features.," in *AAAI*, 2016.

[142] Lubomir Bourdev and Jitendra Malik, "Poselets: Body part detectors trained using 3d human pose annotations," in *ICCV*, 2009.

[143] Qixing Huang, Mei Han, Bo Wu, and Sergey Ioffe, "A hierarchical conditional random field model for labeling and segmenting images of street scenes," in *CVPR*, 2011.

[144] Kota Yamaguchi, M Hadi Kiapour, Luis E Ortiz, and Tamara L Berg, "Parsing clothing in fashion photographs," in *CVPR*. IEEE, 2012.

[145] Yin-Wen Chang and Chih-Jen Lin, "Feature ranking using linear svm.," in *WCCI causation and prediction challenge*, 2008.

[146] Buyu Liu and Xuming He, "Multiclass semantic video segmentation with object-level active inference," in *CVPR*, 2015.

[147] Samuel Rota Bulo and Peter Kontschieder, "Neural decision forests for semantic image labelling," in *CVPR*, 2014.

[148] Buyu Liu and Xuming He, "Learning dynamic hierarchical models for anytime scene labeling," in *ECCV*, 2016.

[149] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *ICLR*, 2015.

[150] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, 2017.

[151] Md Amirul Islam, Mrigank Rochan, Neil DB Bruce, and Yang Wang, "Gated feedback refinement network for dense image labeling," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 4877–4885.

[152] Shaoqing Ren, Xudong Cao, Yichen Wei, and Jian Sun, "Face alignment at 3000 fps via regressing local binary features," in *CVPR*. 2014.

[153] Peter M Roth, Sabine Sternig, Helmut Grabner, and Horst Bischof, "Classifier grids for robust adaptive object detection," in *CVPR*, 2009.

[154] Fisher Yu and Vladlen Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv:1511.07122*, 2015.

[155] Thomas M Cover, Joy A Thomas, and John Kieffer, "Elements of information theory," *SIAM Review*, 1994.

[156] Georgios Floros and Bastian Leibe, "Joint 2d-3d temporally consistent semantic segmentation of street scenes," in *CVPR*, 2012.

[157] Branislav Micusik, Jana Košecká, and Gautam Singh, "Semantic parsing of street scenes from video," *International Journal of Robotics Research*, 2012.

[158] Vijay Badrinarayanan, Ignas Budvytis, and Roberto Cipolla, "Semi-supervised video segmentation using tree structured graphical models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[159] John Francis Kenney and Ernest Sydney Keeping, "Mathematics of statistics-part one," 1954.

[160] Jonas Wulff and Michael J Black, "Efficient sparse-to-dense optical flow estimation using a learned basis and layers," in *CVPR*, 2015.

[161] Matteo Maggioni, Giacomo Boracchi, Alessandro Foi, and Karen Egiazarian, "Video denoising using separable 4d nonlocal spatiotemporal transforms," in *IS&T/SPIE Electronic Imaging*, 2011.

[162] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell, "Clockwork convnets for video semantic segmentation," in *ECCV Workshops*, 2016.

[163] Suyog Dutt Jain, Bo Xiong, and Kristen Grauman, "Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos," in *CVPR*, 2017.

[164] Anna Khoreva, Federico Perazzi, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung, "Learning video object segmentation from static images," *arXiv:1612.02646*, 2016.

[165] Xiaojiang Peng and Cordelia Schmid, "Multi-region two-stream r-cnn for action detection," in *ECCV*, 2016.

[166] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox, "Flownet: Learning optical flow with convolutional networks," in *ICCV*, 2015.

[167] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision*, 2011.

[168] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2015.

[169] Adam Paszke, Soumith Chintala, Ronan Collobert, Koray Kavukcuoglu, Clement Farabet, Samy Bengio, Iain Melvin, Jason Weston, and Johnny Mari-

ethoz, "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration, may 2017," .

[170] Gangqiang Zhao, Junsong Yuan, and Kang Dang, "Height gradient histogram (high) for 3d scene labeling," in *3D Vision (3DV), 2014 2nd International Conference on*, 2014.

[171] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[172] P. Lei, F. Li, and S. Todorovic, "Boundary Flow: A Siamese Network that Predicts Boundary Motion without Training on Motion," *arXiv:1702.08646*, 2017.

[173] Varun Jampani, Raghudeep Gadde, and Peter V Gehler, "Video propagation networks," *arXiv:1612.05478*, 2016.