UNIVERSITY OF CALIFORNIA, MERCED

Learning Affinity to Parse Images

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering and Computer Science

by

Sifei Liu

Committee in charge:

Professor Ming-Hsuan Yang, Chair
Professor Shawn Newsam
Doctor Chang Huang

2017

The dissertation of Sifei Liu is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Professor Shawn Newsam

Doctor Chang Huang

Professor Ming-Hsuan Yang Chair

University of California, Merced

2017

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

VITA

2008	B. S. in Control and Computer Engineering, North China Electrical Power University, Beijing, China
2012	M. S. in Electronic Science and Technology, University of Science and Technology of China, Hefei, China
2017	Ph. D. in Electrical Engineering and Computer Science, University of California, Merced

PUBLICATIONS

Sifei Liu, Shalini De Mello, Jinwei Gu, Guangyu Zhong, Ming-Hsuan Yang, Jan Kautz. *Learning Affinity via Spatial Propagation Networks*, Neural Information Processing Systems (NIPS 2017).

Sifei Liu, Jianping Shi, Ji Liang, Ming-Hsuan Yang. *Face Parsing via Recurrent Propagation*. British Machine Vision Conference (BMVC 2017, spotlight).

Jingchun Cheng, Sifei Liu, Yi-Hsuan Tsai, Wei-Chih Hung, Shalini De Mello, Jinwei Gu, Jan Kautz, Shengjin Wang, Ming-Hsuan Yang. *Learning to Segment Instances in Videos with Spatial Propagation Network*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017 workshop).

Kihyuk Sohn, Sifei Liu, Guangyu Zhong, Xiang Yu, Ming-Hsuan Yang, Manmohan Chandraker. *Unsupervised Domain Adaption for Face Recognition in Unlabeled Videos*, International Conference on Computer Vision (ICCV 2017).

Yijun Li, Sifei Liu, Jimei Yang, and Ming-Hsuan Yang. *Generative Face Completion*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017).

Chih-Yuan Yang, Sifei Liu, and Ming-Hsuan Yang, *Hallucinating Compressed Face Images*, International Journal of Computer Vision (IJCV 2017)

Sifei Liu Jinshan Pan and Ming-Hsuan Yang *Learning Recursive Filters for Low-Level Vision via a Hybrid Neural Network*, European Conference on Computer Vision (ECCV 2016, oral).

Shizhan Zhu, Sifei Liu and Chen-Change Loy*Deep Cascaded Bi-Network for Face Hallucination*, European Conference on Computer Vision (ECCV 2016).

Sifei Liu, Jimei Yang, Chang Huang, and Ming-Hsuan Yang, *Multi-Objective Convolutional Learning for Face Labeling*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015).

Sifei Liu, Chih-Yuan Yang, Ming-Hsuan Yang, *Compressed Face Hallucination*, International Conference on Image Processing (ICIP 2014, oral).

Chih-Yuan Yang, Sifei Liu, and Ming-Hsuan Yang, *Structured Face Hallucination*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2013).

Sifei Liu, Dong Yi, Zhen Lei, Stan Z. Li, *Heterogeneous Face Image Matching Using Multi-scale Features*, IAPR/IEEE conference on Biometrics (ICB 2012).

Sifei Liu, Dong Yi, Bin Li, Stan Z. Li, *Face Alignment under Partial Occlusion in Near Infrared Images*, In Proceedings of Chinese Conference on Pattern Recognition 2010 (CCPR 2010, best paper candidate).

Jianfei Zhu, Zhen Ray, Sifei Liu,Stan Z. Li, Discriminant Analysis with Gabor Phase for Robust Face Recognition, IAPR/IEEE conference on Biometrics (ICB 2012).

Zhiwei Zhang, Junjie Yan, Sifei Liu, Zhen Lei, Dong Yi, Stan Z. Li. Anti-spoofing Attacks in Face Biometrics: A Comprehensive Database and A Baseline. IAPR/IEEE conference on Biometrics (ICB 2012).

Zhangxian Wu, Guotian Yang, Xiangjie Liu, Pengyuan Yang, Sifei Liu, *A Dynamic Texture Model for Fire Recognition*, In Proceedings of Asian Conference on Computer Vision 2009: Workshop on Video Event Categorization, Tagging and Retrieval (ACCV 2009 Workshop).

ABSTRACT OF THE DISSERTATION

Learning Affinity to Parse Images

by

Sifei Liu
Doctor of Philosophy in Electrical Engineering and Computer Science

University of California Merced, 2017

Professor Ming-Hsuan Yang, Chair

Recent years have witnessed the success of deep learning models such as convolutional neural networks (ConvNets) for numerous vision tasks. However, ConvNets have a significant limitation: they do not have effective internal structures to explicitly learn image pairwise relations. This yields two fundamental bottlenecks for many vision problems of label and map regression, as well as image reconstruction: (a) pixels of an image have large amount of redundancies but cannot be efficiently utilized by ConvNets, which predict each of them independently, and (b) the convolutional operation cannot effectively solve problems that rely on similarities of pixel pairs, e.g., image pixel propagation and shape/mask refinement.

This thesis focuses on how to learn pairwise relations of image pixels under jointly, end-to-end learnable neural networks. Specifically, this is achieved by two different approaches: (a) formulating the conditional random field (CRF) objective as a non-structured objective that can be implemented via ConvNets as an additional loss, and (b) developing spatial propagation based deep-learning-friendly structures that learn the pairwise relations in an explicit manner.

In the first approach, we develop a novel multi-objective learning method that optimizes a single unified deep convolutional network with two distinct non-structured loss functions: one encoding the unary label likelihoods and the other encoding the pairwise label dependencies. We propose to apply this framework on face parsing, while experiments on both LFW and Helen datasets demonstrate the additional pairwise loss significantly improves the labeling performance compared to a single loss ConvNet with the same architecture.

In the second approach, we explore how to learn pairwise relations using spatial propagation networks, instead of using additional loss functions. Unlike ConvNets, the propagation module is a spatially recurrent network with a linear transformation between adjacent rows and columns. We propose two typical structures: a one-way connection using one-dimensional propagation, and a three-way connection using two-dimensional propagation. For both models, the linear weights are spatially variant output maps that can be learned from any ConvNet. Since such modules are fully differentiable, they are flexible enough to be inserted into any type of neural network. We prove that while both structures can formulate global affinities, the one-way connection constructs a sparse matrix, and the three-way forms a much denser one. While both structures demonstrate their effectiveness over a wide range of vision problems, the three-way connection is more powerful with challenging tasks (e.g., general object segmentation). We show that a well-learned affinity can benefit numerous computer vision applications, including but not limited to image filtering and denoising, pixel/color interpolation, face parsing, as well as general semantic segmentation. Compared to graphical model base pairwise learning, the spatial propagation network can be a good alternative in deep-learning based frameworks.

Chapter 1

Introduction

1.1 Overview

Learning affinity with deep-learning frameworks is a challenging task. On one hand, convolutional based network has a significant limitation: it does not have an effective internal structure to explicitly learn image pairwise relations. On the other hand, existing pairwise learning frameworks, e.g., graphical models, are not efficient to be formulated as a deep learning module. In this thesis, we focus on how to learn pairwise relations of pixels under jointly, end-to-end learnable neural networks. This is achieved by two different directions: First, we formulate the conditional random field (CRF) objective as two non-structured objectives that can be implemented via ConvNets with multiple losses, as introduced in Chapter 3. Second, we develop spatial propagation based structures that are capable of learning the pairwise relations explicitly. Such structures are fully differentiable and can be inserted into any type of neural network as a jointly learning module. Specifically, Chapter 4 and 5 present two types of spatial propagation networks (SPNs) while both utilize the one-way connections. Chapter 6 further extends the one-way connection to a three-way connection, which substantially improves the capability of pairwise learning in more complex vision tasks. We discuss the problem context and solutions respectively as follows.

1.2 Affinity Learning for Computer Vision Problems

An affinity matrix is a generic matrix that determines how close or similar two points are in a space. In computer vision tasks, it is a weighted graph that regards each pixel as a node, and connects each pair of pixels by an edge [88, 54, 53, 42]. The weight on that edge should reflect the pairwise similarity with respect to different tasks. For example, for low-level vision tasks such as image filtering, the affinity values should reveal the low-level coherence of color and texture [29, 42]; For mid to high-level vision tasks such as image matting and segmentation [54, 69, 41, 4], the affinity measure should reveal the semantic-level pairwise similarities.

Introducing the affinity learning into a typical ConvNet is important but challenging. Let's consider using a fully convolutional network for problems of pixel-wise regression, which typically refers to image object segmentation, filtering and denoising, etc. A well-learned ConvNet that takes an image as its input will compute for each pixel the desired output properties (e.g., semantic labels, smoothed pixel value, etc.), with the consideration of its neighboring local region with a predefined receptive field. In contrast, it is not easy to directly model the desired relation between any pair of pixel with a specific task using such network architecture, since the output dimension is comparatively much larger and redundant.

Another category of the applications for affinity matrices is image filtering. Typical filters include bilateral filter [100], guided filter [42] and weighted least square filter [29]. All of these rely on the manual design of pairwise kernels that construct all the entries of the corresponding affinity matrix. The main disadvantage is that such kernels are manually designed by low-level vision features; they cannot be generalized to high-level vision problems, such as object semantic segmentation.

In this thesis, we focus on both low-level (Chapter 4) and semantic-level (Chapter 3, 5, 6) vision tasks. We show that with the proposed general framework, a well-learned affinity matrix can accurately approximate various types of filtering effects, refine the semantic probability maps and even help to correct the semantic integrity for object segmentation. Other than this, it is also successfully applied to image denoising, inpainting and interpolation with significant quantitative and qualitative improvements over the other solutions and efficient computational speeds.

1.3 Solutions

In this thesis, we provide a convolutional network-based solution which introduces an additional loss function for learning the local pairwise relations, and two types of propagation network-based solutions, which design a novel, deep-learning friendly structure to explicitly learn the global pairwise relations.

1.3.1 Affinity Learning with Deep Convolutional Networks

We model affinity via deep ConvNets for semantic segmentation through multiple objectives. The major goal is to leverage the prediction pixel-wise labels and the regularization of labeling structures together, in order to jointly improve the performance. We develop an end-to-end image segmentation network that can incorporate the structured loss of pixel-wise labeling into the plain, no-structured ConvNet. Specifically, we formulate the labeling of pixels as a conditional random field with unary and pairwise classifiers. This is carried out by developing a multi-objective learning method that optimizes a single unified deep convolutional network with two distinct, non-structured loss functions: one encoding the unary label likelihoods and the other encoding the pairwise label dependencies. For segmenting some domain specific images (e.g., face), the network can be further regularized by using a nonparametric prior (e.g., mean face) as new input channels in addition to the RGB image, which introduces significant performance improvements through a much smaller network size. The proposed algorithm is applied to face parsing on the LFW and Helen datasets, reported in [64] and specified in Chapter 3. It demonstrates superior performance over all the existing non-deep-learning methods, with accurate labeling results on challenging images.

1.3.2 Affinity Learning with Spatial Propagation Networks

The proposed spatial propagation networks (SPNs) focus on learning the pairwise relations via recurrent based structures. Compared to the graphical model-based methods (e.g., CRF), spatial propagation networks are fully differentiable and do not need the separate design of learning and inference. In Chapter 6, we show that a general spatial propagation module can have two forms of connections: (a) one-way connection that shows great effect and efficiency for both low-level [62] (Chapter 5) and high-level [63] (Chapter 5) vision problems, and (b) three-way connection [61] (Chapter 6) that can learn dense affinity matrix for more challenging tasks.

Learning the one-way propagation for low-level vision. In Chapter 4, we use the one-way connection structure for learning numerous low-level vision problems (e.g., edge-preserving filtering and denoising). Specifically, we prove that the one-way connection is equivalent to a one-dimensional recursive filtering, where the weights of connections are equivalent to the coefficients of the recursive filter. We formulate a hybrid network that contains several one-way propagation layers as equivalents of a group of distinct recursive filters for each pixel, and a deep ConvNet that learns their weights. The deep ConvNet can learn regulations of recurrent propagation for various tasks and effectively guides spatial propagation over an entire image. The spatial propagation strategy makes sufficient use of image pixel redundancy, and greatly reduces the model size as well as computational cost required by a ConvNet-base framework on similar tasks. The proposed model does not need a large number of convolutional channels nor big kernels to learn features for low-level vision filters. We report both the theory and the experimental results in Chapter 4, which demonstrates that many low-level vision tasks can be effectively learned and carried out in real-time by the proposed algorithm.

Learning the one-way propagation for high-level vision. In Chapter 5, we further show that the one-way connection structure brings substantial improvements to face parsing, a relatively high-level vision task. We propose a face parsing framework that combines hierarchical representations learned by a ConvNet, and accurate label propagation achieved by a spatially variant recurrent neural network (RNN), which is a variant of the one-way connection. The RNN-based propagation approach enables efficient inference over a global space with the guidance of semantic edges generated by a local convolutional model. Unlike Chapter 4, this work focus on formulating an efficient and light-weight solution. Since the convolutional architecture can be shallow and the spatial RNN can have few parameters, the framework is much faster and more light-weighted than the state-of-the-art ConvNets for the same task. We apply the proposed model to coarse-grained and fine-grained face parsing. For fine-grained face parsing, we develop a two-stage approach by first identifying the main regions and then segmenting the detail components, which achieves better performance in terms of accuracy and efficiency. We demonstrate in this chapter that, with a GPU, the proposed algorithm parses face images accurately at 300 frames per second, which facilitates real-time applications.

Learning the three-way propagation: A dense affinity. We provide a three-way connection in Chapter 6, and theoretically prove that a dense affinity matrix can be formulated in such an efficient way. We show that by constructing a row/column linear propagation model, the spatially varying transformation matrix exactly constitutes an affinity matrix that models dense, global pairwise relationships of an image. Specifically, we develop a three-way connection for the linear propagation model, which (a) formulates a sparse transformation matrix, where all elements can be the output from a deep ConvNet, but (b) results in a dense affinity matrix that effectively models any task-specific pairwise similarity matrix. Instead of designing the similarity kernels according to image features of two points, we can directly output all the similarities in a purely data-driven manner. The spatial propagation network is a generic framework that can be applied to many tasks, which traditionally benefit from designed affinity, *e.g.*, image matting, colorization and guided filtering, to name a few. Essentially, the model can learn semantically-aware affinity values for high-level vision tasks due to the powerful learning capability of the deep neural network classifier. We validate the framework on the task of refinement for image segmentation boundaries. Experiments on the HELEN face parsing and PASCAL VOC-2012 semantic segmentation tasks show that the spatial propagation network provides a general, effective and efficient solution for generating high-quality segmentation results.

1.4 Organization

The organization of the thesis is as follows: In Chapter 2, we make the literature review with respect to the general pairwise learning problems and methods, and specifically introduce the related applications in this thesis. In Chapter 3, we present how to introduce a local spatial pairwise term to deep convolution networks based on multiple objectives, and validate it for the task of 3-class and 11-class face parsing. In Chapter 4, we introduce how to construct a one-way connection propagation network that is equal to one-dimensional recursive filter, in order to approximate any type of image filter using a light-weight deep model. We specifically show the results with respect to numerous low-level vision applications, including but not limited to edge-preserving smoothing and enhancement, image denoising and inpainting, image and color interpolation. In Chapter 5, we introduce a fast solution for face parsing using the combination of a recurrent-based shallow net for the main regions, and several subnets for the details facial components. In Chapter 6, we provide the general theory of the spatial propagation network (SPN), in which the Chapter 4 and 5 are special cases (a sparse affinity matrix) under this framework. We introduce a three-way connection that can provide a dense affinity matrix, which proves to have better performance for more complex vision tasks. We validate the method on both high-resolution face parsing in the HELEN dataset, and the general object semantic segmentation in the Pascal VOC dataset. We conclude the thesis in Chapter 7, and specifically discuss the potential future work with border range of pairwise learning and its applications.

Chapter 2

Literature Review

In this Chapter, we review the literature related to the research work in terms of general pairwise learning as well as its typical applications, including learning image filters, face parsing and object segmentation.

2.1 Pairwise Learning

Numerous methods explicitly compute affinity matrices for image filtering [42], colorization [53], matting [54] and image segmentation [51] based on the physical nature of the problem. Other methods, such as total variation (TV) [84] and learning to diffuse [60] improve the modeling of pairwise relationships by incorporating more priors into the framework of diffusion partial differential equations (PDEs). However, due to the lack of an effective learning strategy, it is still challenging to model complex pairwise learning problems. Recently, Maire et al. [69] trained a deep ConvNet to directly predict the entities of an affinity matrix, which demonstrated good performance on image segmentation. However, since the affinity is followed by a solver of spectral embedding as an independent part, it is not directly supervised for the classification/prediction task. Bertasius et al. [4] introduced a random walk network that optimizes the objectives of pixel-wise affinity for semantic segmentation. Differently, the affinity matrix is additionally supervised by ground-truth sparse pixel similarities, which limits the potential connections between pixels.

On the other hand, many graphical model-based techniques have successfully model pixel pairwise relations in semantic labeling spaces and improves the performance of image segmentation. An early work is proposed in [28], which combines multiscale ConvNets with a region tree structure for scene parsing. Specifically, the ConvNet is trained in an unsupervised layer-wise manner from multiple scales. The learned multiscale image features are then used to train region-wise classifiers for label prediction in a preconstructed segmentation tree. This is a typical two-step approach that sequentially trains a ConvNet and a graphical model. Other than sequential combination, joint training of ConvNets and graphical models have been reported in several algorithms [71, 81, 101]. Ranftl et al. [81] combine a variational energy model with ConvNets for foreground/background image segmentation. The variational model used in [81] can be considered as a relaxation of CRF labeling models. Three ConvNets for unary, vertical pairwise and horizontal pairwise terms are trained separately without weight sharing. The joint training approach has also been applied to human pose estimation. In [101], a ConvNet model is used to train part detectors (unary) and part likelihood maps are then combined with image input to train pairwise spatial models between parts. All the above methods are closely related to the algorithm proposed in Chapter 3, however, our multi-objective learning uses shared weights that further reduces the computational complexity, and was the first deep learning framework introducing the pairwise learning into the problem of face parsing.

More recently, CRFs by efficient mean-field inference have been frequently used as pairwise modules for deep learning frameworks [51, 125, 56, 15, 86, 3]. Some of them [125, 56, 86, 3] translate inference into a differentiable module with an end-to-end training strategy. Others apply it as a post-processing step with manually defined kernels [51, 15]. While both CRFs and the affinity matrix describe the pairwise relationships between pixels, they solve for different objective functions. To connect with convolutional networks, mean-field approximation [51, 125] is commonly used to construct a jointly learnable graphical module. However, since this module (a) still relies on a handcrafted kernel, and (b) requires iterative computation during training and inference, its effectiveness and efficiency is potentially restricted. We specifically compare the propagation network-based methods with Dense CRF [51, 15] for face parsing and general object semantic segmentation in both chapters 5 and 6. We show that the proposed algorithms are more suitable for the deep-learning framework with better performance and faster computation speed.

Our propagation network structure is also related to the recurrent networks [18], which have been shown to be effective for modeling long term dependencies in sequential data (e.g., speech). Its variants include long short-term memory [37, 23, 39, 9], gated recurrent units (GRUs) [18], and others. For image data, we can apply one-dimensional RNN to multiple dimensions in row/column-wise manner [36, 105, 49] or multi-dimensional RNN (MDRNN) [103, 9, 36] such that each neural node can receive information from multiple directions [9, 104] (as opposed to one direction in the conventional RNN). In addition, there are other variants that leverage these two models, e.g., the grid LSTM [49, 55]. In Chapter 4 and 5, we propose one-dimensional propagation modules that fall into the first category. Specifically, both models utilize linear recurrent formulation to associate the adjacent pixels in either the semantic label space or the low-level image space. A similar module is proposed in [14] with the concept of domain transform, where object edges learned on top of the intermediate layers of a fully convolutional network (FCN [66]) is used to regularize the transforms between adjacent pixels. In contrast, the module proposed in Chapter 4 extends the recurrent structure to high-order recursive filters to model more variations in the low-level space, while on the other hand, the module proposed in Chapter 5 is utilized to formulate a shallow and faster architecture as a real-time solution for face parsing. In Chapter 6, we propose a three-way, two-dimensional propagation module that is more related to the second category. However, since this module is fully linear as an affinity learning framework, it is very different with the standard RNN and LSTM as well as their variants, which contain multiple non-linear units.

2.2 Affinity Related Vision Applications

A well-learned affinity can benefit numerous vision problems in terms of performance and visual quality. In this thesis, we mainly demonstrate the effectiveness of the proposed methods for the applications of face parsing, general object semantic segmentation and low-level vision filters. Among them, face parsing is a practical application of semantic segmentation in the face domain for facial image analysis and editing, which is proposed in Chapter 3, 5 as well as 6. We also demonstrate the proposed propagation networks for general low-level vision problems in Chapter 4, and object semantic segmentation in Chapter 6. We introduce the related work with these applications in the following part.

2.2.1 Face Parsing

Face landmark vs face parsing. While both are important methods for analyzing a face image, they are different in concepts and applications. Face landmark detection [8, 10, 126] is to localize the key positions and define the basic shape of a face. Compared to face parsing that semantically labels all pixels, it has very sparse definitions on a fixed number of locations. One can use the landmarks to generate pixel-wise labels by connecting all neighboring pixels in a certain order. However, the results can be very coarse and even unavailable for some undefined regions, such as forehead and hair. However, face landmark detection can provide the correspondences of all control points among different faces, while face parsing cannot. For applications, face landmark detection is usually used for deforming/warping a face image [46], while face parsing is more frequently applied by image processing with semantic regions [80].

Face parsing. Face parsing considered in this work assigns dense semantic labels to all pixels in an image. Typically, only one face image is assumed to be detected in an input frame. Several approaches have been developed based on graphical models [48], exemplars [94] and convolution networks [67]. Warrell and Prince [106] use a family of multinomial priors to model facial structures and a CRF for labeling facial components. In [48], Kae et al. model the face shape prior with a restricted Boltzmann machine and combine it with a CRF for 3 classes labeling (background, face and hair). These two methods train classifiers based on hand-crafted image features as the unary terms of CRFs. Luo et al. [68] propose a deep learning based hierarchical face parsing method by combining several separately trained models, in which only facial components are labeled. In [93], Smith et al. develop a method to parse facial components and skin by transferring labeling masks from aligned exemplars. In [116] Yamashita *et al.* propose a weight cost function to deal with unbalanced samples for parsing small regions.

In this thesis, we provide two typical solutions for face parsing: Chapter 3 (also in [64]) develops a unified, ConvNet-based model to generate complete labels of facial regions in one single pipeline. An additional supervision of semantic edges is utilized to achieve substantial improvements for coarse-grained and fine-grained face parsing. Chapter 5 further provides a much faster solution (also in [63]) using a two-stage training method with hybrid networks based on the idea of spatial propagation. In addition, we also provide a general propagation method for parsing high resolution face images in Chapter 6 (also in [61]). All these methods demonstrate the efficiency and effectiveness of modeling the pairwise relations in the labeling space.

2.2.2 Learning Low-level Vision Filters

Low-level vision. The recent years have witnessed significant advances in numerous low-level vision problems due to the use of designed priors and propagation methods under the guidance of image structures. In edge-preserving image smoothing, the key problem is to design structural priors to preserve sharp edges. Some explicit weight-averaging filters, e.g., bilateral filters [100] and guided image filters [42] exploit internal or guided image structures to preserve the edges of filtered images. Most energy-based edge-preserving methods explicitly or implicitly design adaptive weight maps through image structures (e.g., image gradients), such as edge-preserving decompositions [29], relative total variation [115], to achieve this goal. In PDE-based image processing, the edge-preserving effect is achieved by hand-craft anisotropic diffusion operators [107]. These adaptive weight maps control whether the image regions should be smoothed or not. Similar ideas have been used in image denoising and inpainting.

Numerous recent image processing methods, e.g., colorization [53, 114] and image matting [54, 114], involve propagation that is equivalent to implicitly filtering an image according to its structure. Although significant progress has been made, solving any of these problems is not a trivial task as specific operations are required. For example, in [53, 54], the specific rules for the propagation affinity are manually designed. Such methods cannot be easily generalized as a unified framework for image filtering.

Deep learning models for low-level vision problems. Several data-driven deep learning methods for low-level vision have been explored in recent years [112, 113, 82, 83, 24]. One significant advantage is that these data-driven models are good approximations to multiple conventional filters/enhancers via one learning paradigm. The uniform edge-preserving ConvNet filter [113] is able to achieve 200 times acceleration against some conventional methods. In addition, ConvNets have been applied to image denoising [7, 110, 2], super resolution [25], and deconvolution [112], among others. However, there are two factors that limit the performance of deep ConvNet based models. First, these models are generally large due to numerous convolutional operations. Second, the redundancy between pairs of pixels is not effectively made use of. As another class of neural networks, RNNs have been recently exploited for high-level vision problems such as object recognition [105] and scene labeling [9], through applying recurrent propagations over the spatial domain. In Chapter 4, we show that the one-dimensional recurrent networks can be better exploited for effective and efficient image filtering for low-level vision problems. Different with the standard RNNs, this network utilizes a linear, spatially variant structure that can equivalently formulate a standard one-dimensional recursive filter with arbitrary orders.

2.2.3 Object Segmentation

Non-deep learning methods. Object segmentation is an important computer vision problem that has been studied for decades. In the previous decade, most methods relied on hand-crafted features combined with a unary classifiers (e.g., boosting [90], support vector machine [30], random forest [89]). Significant improvements have been achieved by introducing richer information from context [11, 117, 65] and structure prediction [28, 71, 81, 101], where several of them are introduced in Section 2.1. Typically, the performance of this system is limited by the expressive power of the handcraft features.

Deep regression of pixel-wise labels. More recently, deep ConvNet has been dominating with its significant advantage of learning powerful feature representations. One of the most typical categories of approaches is to utilize ConvNet to directly regress the dense image labels [66, 26] in a fully convolutional fashion, transforming the last fully connected layers of the ConvNet into convolutional or deconvolutional layers. Specifically, [66] produces the class probability map with the softmax layer at the 8× smaller resolution of the original input image. It then bilinearly upsamples the map to the desired resolution. [26] refines the prediction result using an additional ConvNet. Much more advanced network architectures have been proposed after these two work [17, 72, 124], which demonstrate superior performance on the VOC 2012 dataset with respect to the object segmentation task. The common disadvantage is that these work relies on pure convolutional units, which do not explicitly model the pixel pairwise relations in the labeling space.

Segmentation refinement by utilizing pixel pairwise relations. As has been introduced in Section 2.1, a well-designed/learned pairwise relations can significantly improve the performance and object details visually for semantic segmentation. For modeling pixel pairwise relations, graphical models are frequently exploited [28, 71, 81, 101, 51, 125, 56, 15, 86, 3]. We have specifically introduced the CRF-related methods [125, 3, 56, 15] in Section 2.1. Similar strategies are frequently used to refine a coarse probability map. Other than these, several affinity matrix related refinement modules [41, 4] are proposed, either using a learned guidance that is conditioned on the original input image [41], or integrated through linear matrix multiplication based on a jointly learned kernel [4]. Similar as [4], the method [13] is proposed under the framework of gaussian CRFs, with the key difference that the matrix multiplication uses a reversed formulation and thus requires solving linear equations during both the training and inference stages. In this thesis, the propagation networks, either with one-way or three-way connections (specific introductions can be found in Chapter 6), can be used as a segmentation refinement module as demonstrated in Chapter 5 (for face parsing) and Chapter 6 (for general object semantic segmentation). Our method also relies on the matrix multiplication similar with [4], but exploits the spatial propagation structure instead of the kernel methods. We provide thorough theoretical analysis and proofs for the properties of their corresponding affinity matrix in Chapter 6, as well as substantial quantitative results in comparison to a typical graphical model based pairwise learning methods [51, 16, 15] for face parsing and semantic object segmentation in Chapter 5 and 6, respectively.

Chapter 3

Multi-Objective Convolutional Learning for Face Labeling

3.1 Introduction

Deep convolutional neural networks (ConvNets) have been applied to image labeling and parsing problems [21, 28, 19, 68, 95]. As powerful end-to-end nonlinear classifiers, ConvNets generate more discriminative representations compared to traditional methods based on hand-crafted features. Conditional random fields (CRFs) are another important class of image labeling models [70, 31, 35, 5] that carry out structured prediction by considering label dependencies and allow flexible use of pre-trained image features. We are concerned with combining ConvNets and CRFs for image labeling by exploiting rich features from ConvNets and structured output from CRFs [81, 101]. Considering a typical CRF energy function with unary and edge terms, a straightforward combination is to add CRF based structured losses on top of ConvNets. Learning ConvNets with structured loss, however, requires MAP inference of all the samples during training cycles. On one hand, it significantly increases computational cost while restricting the training flexibility. On the other hand, the direct combination of ConvNets and CRFs with the structured loss may not guarantee convergence.

This chapter presents a novel learning method by decomposing the structured loss into two distinct, non-structured losses: softmax loss for the unary term and logistic loss for the edge term. The training process is carried out through a multi-objective optimization, which minimizes the losses of unary and edge terms respectively through a unified convolutional network. Weight sharing is enforced between them so that the network is strengthened by learning from both objectives. Compared to a structured loss ConvNets, our method has two advantages. First, the training process is as efficient as existing ConvNets with non-structured losses. Second, by converting the edge term into a logistic loss (edge versus non-edge), semantic image boundaries are learned for effective labeling.

Our model is trained on patches for flexibility. During the test stage, by making some simple adjustments, we can apply our patch model directly to a full image without patch cropping for efficient pixel-wise label prediction. We apply the proposed learning algorithm to a practical problem, face labeling that assigns every pixel a facial component label, e.g. skin, nose, eyes and mouth. Two examples are shown in Figure 3.1. Compared to facial landmarks, face labeling provides a better intermediate representation for many face analysis, synthesis and editing tasks.

Faces are highly structured visual patterns. For image labeling, we integrate a global facial prior into our learning model. The global facial prior is estimated by transferring labeling masks from exemplars through landmark detection [93]. Unlike existing methods [93, 118, 58] that use this nonparametric prior at the inference stage, our method uses it as additional input channels, other than raw RGB image intensities, to train a ConvNet. We show that this nonparametric prior significantly reduces the size of ConvNet in terms of both parameters and connections. In other words, it provides strong regularizations for ConvNet training to facilitate lightweight architectures.

The proposed face labeling algorithm is evaluated on two challenging benchmark datasets with 3 classes (LFW) [45] and 11 classes (Helen) [93]. Experimental results show that our algorithm performs favorably against state-of-the-art methods. We also present hair parsing results that can be generated simultaneously from the unified framework, which is more challenging and rarely addressed by existing methods.

The contributions of this chapter are summarized as follows:

- a multi-objective convolutional learning method is developed for image labeling problems by decomposing the structured loss of CRFs into two distinct, non-structured losses, and optimizing a single unified ConvNet model with weight sharing;
- a nonparametric facial prior is introduced to ConvNet training that significantly reduces the network size;
- an efficient testing method is proposed to ensure fast, full-sized labeling.

3.2 Multi-Objective Convolutional Learning

We formulate the problem of labeling a face image $\mathbf{X}$ as a CRF model $P(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z}\exp(-E(\mathbf{Y},\mathbf{X}))$ where Z is the partition function and $\mathbf{Y}$ is a set of random variables $y_i \in \mathbf{Y}$ defined on every pixel i. Each variable y_i takes a value from a set of labels $\{\ell = 1, 2, .., K\}$. To consider the label dependencies, we introduce a 4-connected graph $(\mathcal{V}, \mathcal{E})$ where each node represents one pixel $i \in \mathcal{V}$ and edges represent the connections between any two adjacent pixels $i, j \in \mathcal{E}$. Therefore, the CRF model can be expressed as a energy function $E(\mathbf{Y},\mathbf{X})$ with two data-dependent terms:

$$E(\mathbf{Y},\mathbf{X}) = \sum_{i\in\mathcal{V}} E_u(y_i, \mathbf{x}_i) + \lambda \sum_{(i,j)\in\mathcal{V}} E_b(y_i, y_j, \mathbf{x}_{ij}). \tag{3.1}$$

The unary term $E_u(y_i, \mathbf{x}_i)$ measures the assignment cost of variable y_i based on the image patch $\mathbf{x}_i$ centered at the pixel i and the pairwise term $V(y_i, y_j, \mathbf{x}_{ij})$ encodes the consistency cost of adjacent variables y_i, y_j given their overlapping patch $\mathbf{x}_{ij}$. In addition, λ is the mixing constant. We introduce a multi-class classifier $P_u(y_i = \ell | x_i, \omega_u)$ to express the label

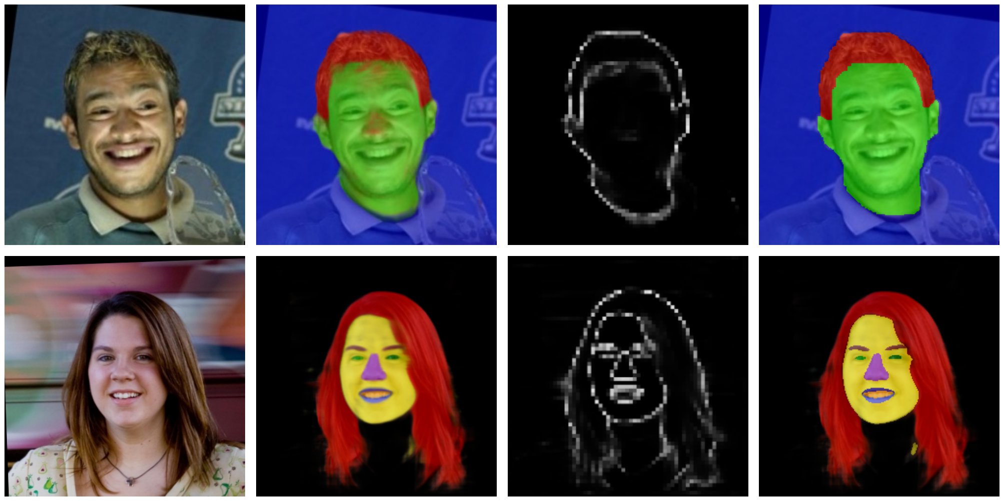


Figure 3.1: Face labeling on the LFW [48] and Helen [1] (a) input images. (b) pixel-wise label likelihoods. (c) semantic edge maps. (d) face labeling results. Our algorithm first generates pixel-wise label likelihoods and semantic edge maps, which are combined in a CRF energy function to generate face labels. The images in (b) are soft labels (probabilistic outputs) and images in (d) are hard labels (excluding hair) which are shown in different colors. While the pixel-wise maps alone are effective for labeling, the use of edge maps further facilitates delineating the details, especially near the class boundaries.

assignment cost for the unary term,

$$E_u(y_i, \mathbf{x}_i, \omega_u) = -\log P_u(y_i = \ell | x_i, \omega_u). \tag{3.2}$$

To measure the consistency of two adjacent pixels i, j in the pairwise term, we introduce a new label $z_{ij} = 1$, if $y_i = y_j$ and $z_{ij} = 0$, otherwise. Thus the pairwise term is defined by the output of a binary classifier $P_b(z_{ij} = 1 | x_i, \omega_u)$,

$$E_b(y_i, y_j, \mathbf{x}_{ij}, \omega_b) = -\log P_b(z_{ij} = 1 | x_i, \omega_b). \tag{3.3}$$

In this work, we use ConvNets with 9 layers for both unary and pairwise classifiers as they provide end-to-end predictions without using hand-crafted features.

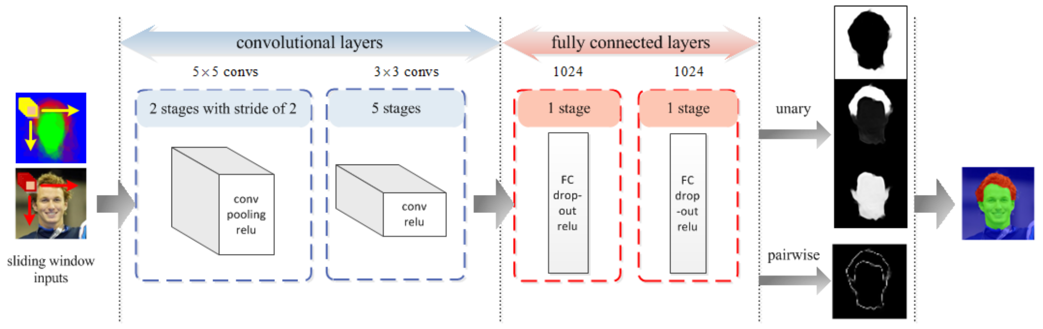


Figure 3.2: Proposed ConvNet classifier with sliding window based inputs.

Learning ConvNet parameters ω_u and ω_b jointly with the CRF model is difficult as it needs to explore not only the combinatorial labeling space but also the large parameter space. To avoid this problem, an obvious approach is to train two independent ConvNets for the unary and pairwise terms, respectively. We note that both ConvNets are based on local image patches and should share very similar features in the lower layers [91]. In addition, the potentially large set of parameters from two ConvNets may cause overfitting problems. In this chapter, we propose to learn a single unified ConvNet for both unary and pairwise classifiers. By sharing all the features within a single ConvNet, the two classifiers are able to enjoy better generalization ability and higher computational efficiency.

We define two distinct loss functions for unary and pairwise classifiers, respectively. We denote the parameters of shared ConvNet network by ω, and the feature response extracted from the topmost intermediate layer of ConvNet by $h_i = h(\mathbf{x}_i, \omega)$. Thus, the output of the unary classifier is given by a softmax function,

$$P_u(y_i = \ell | h_i, \omega_u) = \frac{\exp((\omega_u^\ell)^\top h_i)}{\sum_{\ell=1}^K \exp((\omega_u^\ell)^\top h_i)}, \tag{3.4}$$

where ω_u^ℓ represents the parameters for the ℓ-th class. Accordingly, the softmax loss for unary term is

$$L_u(y_i, \mathbf{x}_i, \omega, \omega_u) = -\log P_u(y_i = \ell | h_i, \omega, \omega_u). \tag{3.5}$$

On the other hand, the output of pairwise classifier is given by a logistic function,

$$P_b(z_{ij} = 1 | h_i, \omega_b) = \frac{1}{1 + \exp(\omega_b^\top h_i))}, \tag{3.6}$$

and accordingly, the logistic loss for pairwise term is

$$L_b(z_{ij}, \mathbf{x}_{ij}, \omega, \omega_b) = -\log P_b(z_{ij} = 1 | h_i, \omega, \omega_b). \tag{3.7}$$

Based on these two loss functions (3.4) and (3.6), we train the unified ConvNet through multi-objective optimization,

$$\begin{aligned} &\min_{\omega} \{O_u(\omega, \omega_u), O_b(\omega, \omega_b)\}, \\ &\begin{cases} O_u(\omega, \omega_u) = \mathbb{E}(\sum_{i \in \mathcal{V}} L_u(y_i, \mathbf{x}_i, \omega, \omega_u)) + \Psi(\omega, \omega_u) \\ O_b(\omega, \omega_b) = \mathbb{E}(\sum_{i,j \in \mathcal{E}} L_b(z_{ij}, \mathbf{x}_{ij}, \omega, \omega_b)) + \Phi(\omega, \omega_b) \end{cases} \end{aligned} \tag{3.8}$$

where $O_u(\omega, \omega_u)$ is the expected loss $\mathbb{E}(\cdot)$ for the unary classifier and $O_b(\omega, \omega_b)$ is the expected loss for the binary classifier over all the training samples. In addition, $\Psi(\omega, \omega_u)$ and $\Phi(\omega, \omega_b)$ are regularization terms. The network is updated through combining gradients of both the softmax and logistic loss functions for backpropagation.

This multi-objective ConvNet has two main advantages: First, the convolutional network generates expressive representations at lower levels (layers that are close to the input end) that can be utilized for both unary and edge model regressions. Second, the unified network can be learned by backpropagating errors from both outputs jointly such that the network can learn features that are highly adaptive to both objectives. The shared model also alleviates overfitting problems and reduces the overall model size such that both training and testing can be carried out efficiently.

3.2.1 ConvNet Architecture

Since ConvNets usually operate on a patch level centered at each pixel, Our labeling pipeline is based on a sliding window input [87, 101] with overlapping patches, as shown

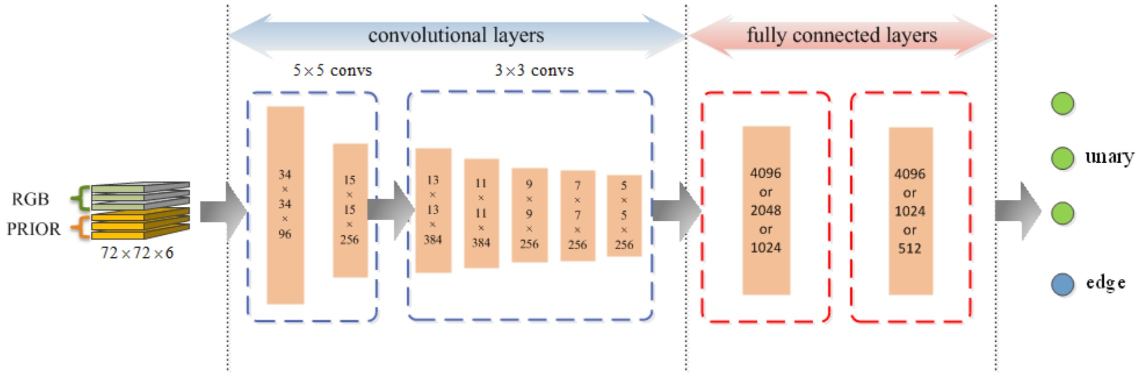


Figure 3.3: Proposed ConvNet classifier for the LFW-PL dataset (patch-based training phase) with input and activation size of each layer.

in Figure 3.2. We propose an architecture similar but deeper [91] than that of [52], with 7 convolutional and 2 fully connected layers. The inputs are 72×72 single scale patches which are passed to two top consecutive convolutional unites with a filter of 5×5, where each convolutional layer is followed by one max pooling layer with a downsampling stride of 2. Following that is another stack of small convolutional unites with a receptive field of 3×3 and no pooling layer. All the layers are equipped with a rectification (ReLU) non-linearity and a local response normalization (LRN) layer. We denote the input or activation of each layer in the proposed ConvNet as $h \times w \times d$, where h and w are spatial size, and d is the number of channels for the input, or dimensions for activation. The detailed pipeline with nonparametric prior input is illustrated as shown in Figure 3.3.

3.2.2 Nonparametric Prior

We introduce a nonparametric prior as global regularization for face labeling, which is estimated by transferring label masks from exemplars. Given a test image, 20 ($K = 20$) exemplars are selected by comparing the Euclidean distance of PCA coefficients in the keypoints subspace. The selected exemplars are then aligned to the test image through the similarity transformation. The least-squares optimization is used on the corresponding five keypoint pairs (exemplar vs. test image) to estimate their similarity transformation. The ground truth labels are then transformed using the the similarity transformation, and combined with weights to generate the prior. For any specific class, the prior is:

$$M = \sum_{k=1}^{K} \alpha_k M_K, \quad \alpha_k = \frac{\|UP - UP_k\|_2^2}{\sum_{l=1}^{K} \|UP - UP_l\|_2^2}, \tag{3.9}$$

where U is the eigenvector of keypoints on the validation set. In addition, P and P_k are respectively the detected five keypoints for test image and the k-th exemplar, and M_k is the ground truth binary label for the k-th exemplar. In (3.9), UP_k forms the projection of P_k in the subspace. The weight α is proportional to the Euclidean distance of the test image to the exemplars, where α_k are the weight for the k-th exemplar among the nearest K exemplars. The value of M ranges from 0 to 1, which is then used as an additional input channel for training the ConvNet. A typical labeling improvement using this prior is shown on the right side of Figure 3.4. The ConvNet trained on image patches incorrectly labels the face on the upper left part according to its local content while the ConvNet trained on both prior and image patches is able to reject the false label assignments. Moreover, the prior input introduces relaxation to the ConvNet so that the training could converge faster. We will also show that using the prior leads to significant reduction of the network size without degrading performance.

3.2.3 Adaptive Inference

Submodular energy function. In the testing stage, the labeling process involves evaluating the learned ConvNet for both unary and pairwise terms and CRF inference. Figure 3.2 demonstrate the inference pipeline. Given pixel-wise label likelihood maps for the unary term $E_u(y_i)$ and the edge map for the pairwise term $E_b(y_i, y_j)$, we convert the original energy function to a submodular one such that the GraphCut algorithm can be used for

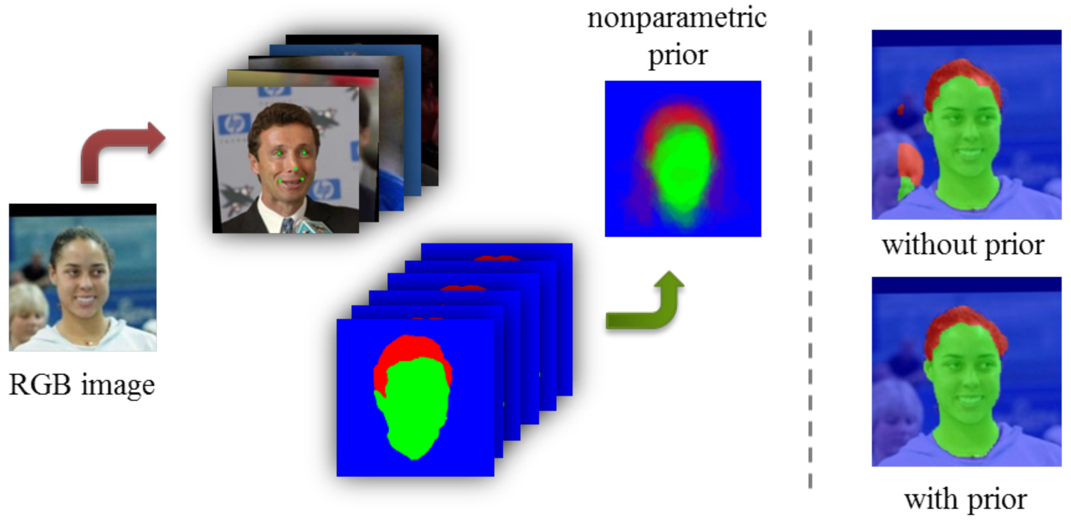


Figure 3.4: An nonparametric prior is proposed based on label transfer, as shown on the left. A typical labeling improvement is show on the right. The ConvNet trained on image patches without exemplars incorrectly labels the face on the upper left part according to its local content while the ConvNet trained on both prior and image patches is able to reject the false label assignments.

efficient inference,

$$\min E(\mathbf{Y}) = \sum_{i \in \mathcal{V}} E_u(y_i) + \sum_{i,j \in \mathcal{V}} E_b(y_i, y_j)\mathbb{I}(y_i \neq y_j), \tag{3.10}$$

where $\mathbb{I}(\cdot)$ is the indicator function.

Adapting the patch ConvNet to the full image. To generate pixel-wise label likelihood maps efficiently, we make some adjustments for the ConvNet trained on patches. Our ConvNet architecture consists of more layers but smaller filter size, compared to that in [52]. Thus, we have a considerate size of overall receptive field and more nonlinearity of the decision function, without increasing the number of parameters. Specifically, we use fewer pooling layers (only in the first two convolutional units) to preserve more spatial information from the input image. We propose to use the following efficient testing schemes to enable full-sized label likelihood maps. In the training stage, we sample a group of patches centered with randomly selected pixels for each training image. The network is supervised through the corresponding labels y and z respectively for unary and edge training. We update model by mini-batch gradient descent.

Since convolutional operations share computations between overlapped patches, computing the sliding-window pipeline for each pixel of a test image is computationally redundant. We propose to use an efficient *patch-training and image-testing* strategy introduced in [87] by replacing the fully connected (FC) layers with equivalent convolutional layers, and setting the filter size as 1×1 (See Figure 3.2). We then apply the full-convolutional model directly to a test image. Note that a test image is proper padded to ensure that every pixel corresponded "window" can be covered in order to generate the exactly equivalent result. Thus, both unary and edge probability maps can be generated by one full image forward propagation, which is much faster than applying the model many times to sliding windows.

One problem with the proposed full image testing approach is that the size of the output maps is smaller than that of the original image due to the downsampling strides in the max pooling layers. Most existing approaches upsample the low-resolution map to the image size [28]. We propose to obtain the full-sized output maps by forward propagating a group of input images, generated by shifting the original input image with one or more pixels (depending on the zooming factor) on the x and y axis, as described in [77]. The way of generating an upsampled output map with a zooming factor of 2 is illustrated in Figure 3.5. In this work (two pooling layers with downsampling stride of 2), 4×4 times of forward propagations from shifted input images generate an upsampled output map with a zooming factor of 4 in a similar way. Since the size of the maps may still be inconsistent with that of the original image due to border effect of convolutional operation, the final output map can be obtained by rescaling them to the exact input image size. Note that 16 times of forward propagation is still much faster than applying the convolutional network to patches at each location of an image.

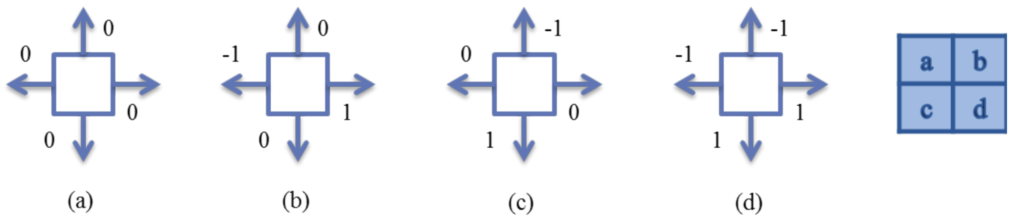


Figure 3.5: Generation of a twice upsampled output map. The original image in (a) is shifted with additional 3 versions (b-d) along the x- and y-axis, with a step of 1. The desired high-resolution one is obtained by interlacing them in the way shown on the right, with a 2×2 block. The final upsampled map layouts are shown on the right.

3.3 Experimental Results

We evaluate the proposed algorithm on two different benchmark dataset with different face labeling tasks. We show that it applies to both tasks and performs favourably against state-of-the-art methods with the same framework and experimental settings. Specifically, we demonstrate that both the multi-objective approach and the nonparametric prior improves the performance in all aspects compared to a per-pixel ConvNet classifier. We validate that the nonparametric prior introduces regularization to the network by reducing the number of network parameters and connections.

3.3.1 Datasets and Settings

Datasets. We use the LFW [48] dataset which has been used by recent methods for face labeling [106, 68]. However, the image subsets that are used for training and testing by these two methods are not available to the public. Kae *et al.* [48] report their 3-classes face labeling results on a released subset of LFW. For fair comparison, we choose to conduct our first labeling experiment on the same subset of images, named *LFW part labels database (LFW-PL)*, with the same evaluation criteria.

The LFW-PL set contains 2927 face images of 250×250 pixels acquired in unconstrained environments. All of them are manually annotated with skin, hair and background labels using superpixels. This dataset is divided into a training set with 1500 images, a testing set with 927 images, and a validation set with 500 images. The validation set is used to generate the nonparametric prior for each training and testing image, as described in Section 3.4.

We use the HELEN [1, 93] dataset with 11-classes face labels for the second set of experiments. It is composed of 2330 face images of 400×400 pixels with labeled facial components generated through manually-annotated contours along eyes, eyebrows, nose, lips and jawline. The hair region, not considered in the labeling categories in [93], is annotated through a matting algorithm. The dataset is also divided into a training set (corresponding to the exemplar set in [93]) with 2000 images, a testing set with 100 images, and a validation set (corresponding to the tuning set in [93]) with 300 images.

Network Configurations. Similar network configurations are applied to the *LFW-PL* and *Helen* datasets for face labeling. As mentioned in the Section 3.2.1, we use a single-scale patch input with size of 72×72 pixels in order to keep a proper receptive field. Compare to a multi-scale input [28, 101], the single-scale configuration make the network to be easily adapted from patch-based training to image-based testing. The released images in the *LFW-PL* dataset are coarsely aligned using the congealing alignment method [48]. We align the images of the *HELEN* dataset to a canonical position by detecting five facial keypoints using [97], and computing the similarity transformation using least squares minimization. To adapt to the receptive field to the input patch size, we further resize the images and evaluate them with size of 250×250 pixels.

The ConvNet training procedure is carried out using mini-batch gradient decent with where the momentum, weight decay, dropout ratio, and batch size are set to 0.9, 5×10^{-4}, 0.5, and 50, respectively. All are kept unchanged throughout the training procedures. The learning rate is initially set to 10^{-3} and is manually decreased by a factor of 10 when the loss on the validation set starts fluctuating.

Table 3.1: Overall per-pixel accuracy on the *LFW-PL* dataset with channel numbers of two FC layers setting as 4096 and 1024. Also the F-measure of skin (F-skin), hair (F-hair) and background (F-bg) are presented.

(%)	accuracy	F-skin	F-hair	F-bg
S-CNNs	92.92	90.07	73.73	95.18
MO-unary	93.45	91.45	78.03	95.84
MO-GC	93.77	91.95	79.06	96.03
S-CNNs with prior	94.25	92.79	77.18	96.63
MO-unary with prior	94.94	93.64	79.95	97.02
MO-GC with prior	**95.12**	**93.93**	**80.70**	**97.10**

For evaluation, we use the sliding-window ConvNets, without edges and nonparametric prior, as a baseline of our approach, denoted as **S-CNNs**. We evaluate our multi-objective approach with respect to: (a) unary term (**MO-unary**) with a softmax probabilistic output; (b) inference results from both unary and edge terms through GraphCut, denoted as **MO-GC**. We denote a suffix of **"with prior"** for the approaches with the non-parametric prior.

Sampling. In the training stage, patch sampling is generally based on a random criteria. However, the number of patches from rare classes may be insufficient for training an effective network with such sampling strategy. This is particularly obvious with semantic edges and facial components, where the edge and some facial components such as eyes and lips takes a relatively small portion of pixels. Therefore, we apply a two-stage training with different sampling approaches. We first train the convolutional network by keeping a certain ratio of patch number between one or more rare classes and the others, such that we draw a sufficient of samples for the rare classes. We then apply the globally random sampling for fine-tuning to ensure the network adapts to the natural distributions of classes so in order to further improve the performance.

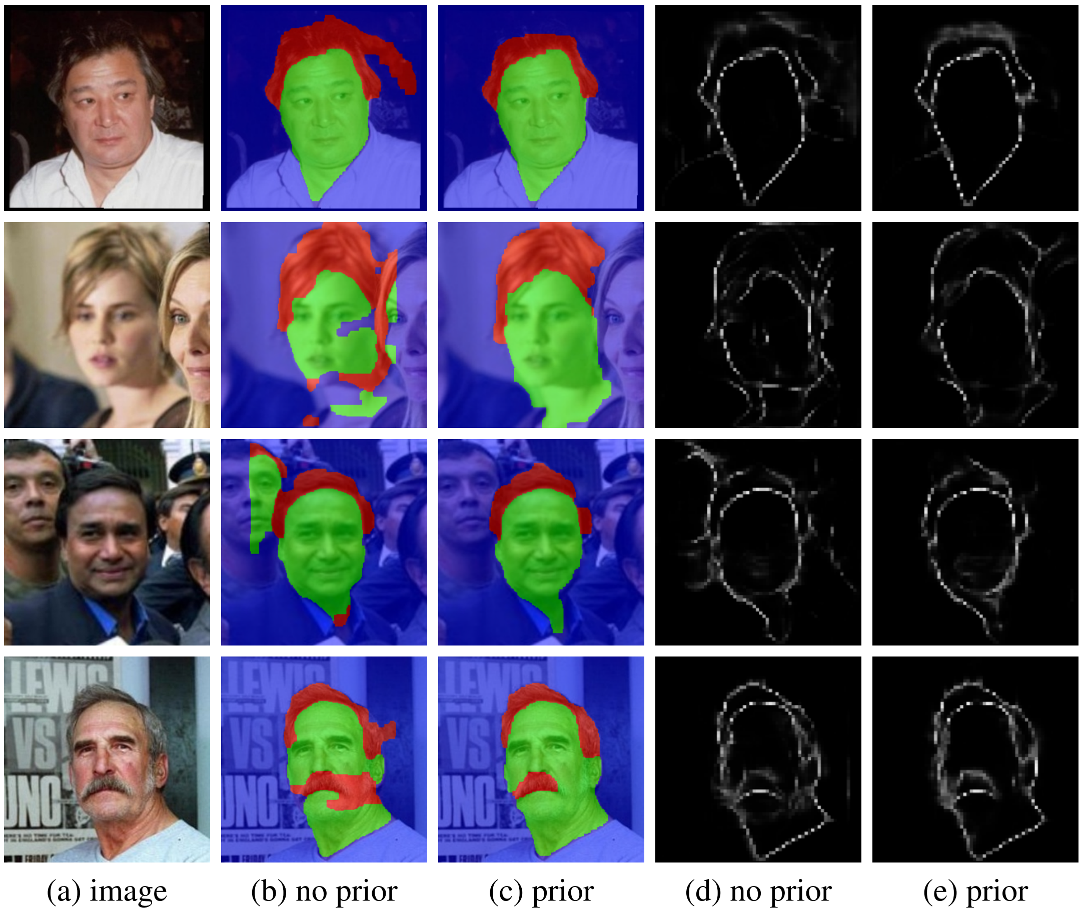
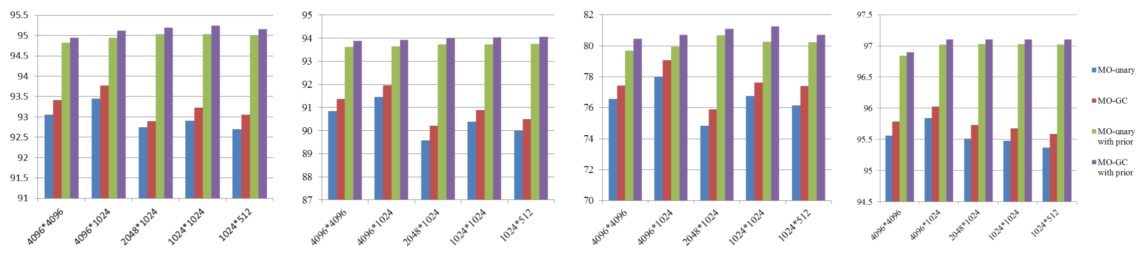


Figure 3.6: Comparison for usage of nonparametric prior. (a) test images; (b) labeling results generated by MO-GC; (c) labeling results generated by MO-GC with nonparametric prior. (d) semantic edge generated by MO-GC; (e) semantic edge generated by MO-GC with nonparametric prior. Best viewed in colors.

Table 3.2: Overall accuracy on LFW-PL with comparison to [48]. Note that the evaluation of GLOC is based on a superpixel-wise accuracy, and ours are based on a per-pixel evaluation.

(%)	GLOC (SP)	MO-unary	MO-GC
accuracy	94.95	95.03	**95.24**
error reduction	25.41	26.59	**29.69**



(a) accuracy (b) F-skin (c) F-hair (d) F-bg

Figure 3.7: We show the network regularization by introducing the proposed nonparametric prior as an additional input. Four FC settings associated with Table 3.3 are used to control the size of the network, as shown in the X-axis.

Runtime. All models are trained and tested using the Caffe package on a single NVIDIA Tesla K10c GPU. Our proposed method (MO-GC with prior) with FCs setting of $1024 * 1024$ takes approximately 120 ms to forward propagate an 250×250 6-channel input (See Figure 3.2). To generate original-sized output maps with 16 shifted versions of an input and to infer the final labeling results, it takes less than two seconds for the full pipeline. The configurations and code will be released to the public.

3.3.2 LFW-PL

We first show results on face labeling of skin, hair and background. In this task, the 3 classes are relatively balanced in the number of pixels, and we randomly sample 12 batches from each training image. We additionally sample 12 batches with the ratio of non-edge and edge setting to 1.2 (this step is removed in the fine-tuning stage). We also apply jitter that generated with random affine transformations over patches [28], to one of the batches. This is easy to be applied to a patch-based training approach, and is particularly effective for increasing the variation of training samples, especially when the number of training images is small.

In table 3.1, we test a series of approaches with the channel numbers of the two FC layers as 4096 and 1024, and evaluate the results with respect to per-pixel accuracy and F-measure of each class. The first 3 rows show the approaches without nonparametric prior input, while the lower 3 rows are those using it. Overall, the nonparametric prior significantly improves the results when compared with all corresponding approaches. We specifically compare the labeling and semantic edge results in terms of the nonparametric prior in Figue 3.6. By comparison between network with (shown in (c) and (e)), and without (shown in (c) and (e)) the nonparametric prior, we observe that the labeling is improved in terms of blurry hair region (Figure 3.6(a)), blurry face (Figure 3.6(b)), multiple persons (Figure 3.6(c)) and moustache (Figure 3.6(c)), through introducing the prior. Moreover, the proposed multi-objective approach (begin with MO) generally outperform the ConvNet classifiers (S-CNNs and S-CNNs with prior). The inference step can further improves the performance of all tested approaches.

Another major improvement of the multi-objective approach can be observed from the comparison between S-CNNs and MO-unary, as shown in the row 1 vs.2 and 4 vs.5 in Table 3.1. Both of them are generated directly from the output probabilities. The only difference is that, the MO-unary contains an additional output that learns the semantic edges. Namely, even when two networks are trained under the same conditions (network configurations, inputs, with or without nonparametric prior, etc.), the one with a supervised semantic edge learning generates results in more expressive representations by back-propagating information of edges.

Network Regularization. By introducing a nonparametric prior as an additional input, the network size can be significantly reduced without degrading the performance. We use different settings of the FC layers as they usually take a large portion of weights and connections in a deep ConvNet architecture. The combination of channel numbers regarding to two FC layers is denoted by $a * b$, where a and b are numbers for the first and the second FC layer, respectively. Four network configurations, as listed in Table 3.3, are applied to evaluate of different model sizes, as shown in Figure 3.7 and table 3.4-3.7. Note that we

Table 3.3: Four settings of channel numbers for the two FC layers, and their corrsponding model size in MB.

FC $a * b$	4096* 4096	4096* 1024	2048* 1024	1024* 1024	1024* 512
size (MB)	163	119	65	38	36

only test and compare at the setting of $4096 * 1024$ (119MB) for **S-CNNs** and **S-CNNs with prior**.

With a nonparametric prior input (colored with green and purple), the performance of the small size networks (e.g. $1024 * 1024$ and $1024 * 512$) are comparatively similar to that of the large-size networks (e.g. $4096 * 4096$ and $4096 * 1024$). With the inference step (MO-GC with prior), the network of $1024 * 1024$ achieves the highest overall accuracy, while the network of $1024 * 512$ achieves the highest F-score for the class of skin. On the contrary, with no prior input (colored with blue and dark red), the network generally has a worse performance when decreasing its size. For the networks less than 119 MB ($4096 * 1024$), the per-pixel accuracies are no higher than 93%.

Table 3.4: Overall per-pixel accuracy on the *LFW-PL* dataset.

accuracy (%)	4096* 4096	4096* 1024	2048* 1024	1024* 1024	1024* 512
S-CNNs	-	92.92	-	-	-
MO-unary	93.05	93.45	92.74	92.91	92.70
MO-GC	93.41	93.77	92.89	93.23	93.05
S-CNNs with prior	-	94.25	-	-	-
MO-unary with prior	94.82	94.94	95.03	95.03	94.99
MO-GC with prior	94.95	95.12	95.19	**95.24**	95.16

Comparison to GLOC. We compare the results with GLOC [48] by following their evaluation of overall accuracy and error reduction with respect to a standard CRF with features in Huang *et al.* [44], as shown in Table 3.2. We apply the setting of FCs with

Table 3.5: Overall F-measure of skin on the *LFW-PL* dataset.

F-skin	4096*	4096*	2048*	1024*	1024*
(%)	4096	1024	1024	1024	512
S-CNNs	-	90.07	-	-	-
MO-unary	90.84	91.45	89.58	90.39	89.99
MO-GC	91.37	91.95	90.21	90.88	90.49
S-CNNs with prior	-	92.79	-	-	-
MO-unary with prior	93.61	93.64	93.73	93.73	93.75
MO-GC with prior	93.89	93.93	94.00	94.03	**94.05**

Table 3.6: Overall F-measure of hair on the *LFW-PL* dataset.

F-hair	4096*	4096*	2048*	1024*	1024*
(%)	4096	1024	1024	1024	512
S-CNNs	-	73.73	-	-	-
MO-unary	76.56	78.03	74.84	76.76	76.17
MO-GC	77.45	79.06	75.90	77.64	77.42
S-CNNs with prior	-	78.03	-	-	-
MO-unary with prior	79.70	79.95	80.67	80.27	80.24
MO-GC with prior	80.47	80.70	81.09	**81.27**	80.70

Table 3.7: Overall F-measure of background on the *LFW-PL* dataset.

F-bg	4096*	4096*	2048*	1024*	1024*
(%)	4096	1024	1024	1024	512
S-CNNs	-	95.18	-	-	-
MO-unary	95.56	95.84	95.51	95.48	95.37
MO-GC	96.79	96.03	95.73	95.68	95.59
S-CNNs with prior	-	96.63	-	-	-
MO-unary with prior	96.84	97.02	97.03	97.03	97.02
MO-GC with prior	96.90	**97.10**	**97.10**	**97.10**	**97.10**

$1024 * 1024$ which achieves the best performance. Note that, a major difference of the evaluation is that [48] applies a superpixel-wise accuracy since it is a superpixel based method, while we use a per-pixel accuracy evaluation since our approach outputs a per-pixel labeling map. For GLOC, the per-pixel evaluations may be slightly different, which however is not reported in their paper.

Figure 3.8- 3.10 more experimental results using challenging images including blurry hair region with low contrast, occlusions and mustache. For unary output on the third column, we show a "soft mask" with values ranges from 0 to 1 for each class. Specifically, the hair region (red) reveals its natural properties of transparency by showing a smooth probability map. For edge output on the second column, we also illustrate a probabilistic output ranging from 0 to 1, which is directly used on the inference step. Our generated edge is clean (with much little of the background) and accurate, which further helps infer labeling results with better class boundaries as shown on the forth column. Although our approach is not specifically designed to handle occlusions, it handles such factors well as shown in Figure 3.8(row 2, 5), Figure 3.9(row 1) and Figure 3.10(row 3).

The fifth column of figure 3.8- 3.10 shows the ground truth labeling for selected examples. We notice that the superpixel labeling proposed by [48] does not generate accurate annotations. Some typical examples are shown in Figure 3.8(row 1, 3, 4, 5), Figure 3.9(row 2, 5, 6, 7) where the boundary regions are not well defined by superpixels, and inaccurate annotations are thus generated. Furthermore, humans may not be able to annotate details well, e.g., the mustache region in Figure 3.9(row 3) and the low-contrast hair region in Figure 3.10(row 6). On one hand, the inaccuracy introduces noise to the supervised ConvNet training, on the other hand, the superpixel-wise evaluation in [48] does not reveal a real accuracy. For instance, our results in the forth column contain a certain number of incorrect label assignments evaluated on the ground truth in the fifth column. However, they are visually even better than the ground truth, particularly along the class boundaries.

3.3.3 HELEN

We also show results on the labeling of 11 classes: two eyes, two eyebrows, nose, upper and lower lips, inner mouth, facial skin and hair. Unlike LFW-PL, some facial components are rare classes (e.g. eyes, lips, etc.) and therefore the two-stage sampling strategy proposed in Section 3.2.3 is applied. Instead of sampling the first 12 batches in a random way as in LFW-PL, we propose to firstly separate the labels as foreground (containing all facial components, skin and hair) and background. We then sample the first 11 batches randomly from foreground and the remaining one from background. In this way, the foreground is sufficiently trained in the first stage, and a natural foreground label distribution can be preserved. We repeat the same edge sampling and jitter generation strategy with LFW-PL. Specifically we train two models for HELEN: For the first model, we train a 11-classes unified convolutional network, with the multi-objective approach with nonparametric prior as additional input. Therefore, we show that the hair labeling can be generated along with other facial labels, which is not addressed in prior work. For the second model, we merge the ground truth hair label with the background to train a 10-classes network using the same approach. In this way, a fair comparison with the work of [93] can be obtained.

Based on the same subset of images with same evaluation criteria, we simply report the results of [93]. In Table 3.8, a large variation in F-measure with respect to each facial component can be seen between [93] and the proposed approaches. While [93] bases the work on exemplar transfer, and obtains better results on relatively rare facial classes, such as eyes, nose and mouth, we outperform it in facial skin and the overall components. Specifically, we achieve an overall F-measure of 0.854, which is a noticeable improvement over the work of [93].

Table 3.8 shows that the labeling of hair regions, which is challenging and seldom addressed in existing facial component labeling methods, can be successfully generated together with other facial components by the proposed algorithm in a unified model. With hair labeling, this proposed method still performs well in overall facial components against the state-of-the-art method on the HELEN dataset. Unlike the superpixel-based annotation in the LFW-PL dataset, the hair in the HELEN dataset is annotated by matting with a "soft mask" that ranging from 0 to 1, as shown in the fifth row in Figure 3.113.12. To be

Table 3.8: Evaluations on HELEN. We use float numbers instead of percentage to keep consistent on the numarical pericision with [93]. For comparison, eyes, brow and mouth all are computed by combining related categories, and the overall denotes all facial components excluding facial skin.

methods	eyes	brows	nose	in mouth	upper lip	lower lip	mouth all	facial skin	overall
Smith *et.al* [93]	**0.785**	0.722	**0.922**	0.713	**0.651**	**0.700**	**0.857**	0.882	0.804
Ours, 11 classes	0.768	0.713	0.909	0.808	0.623	0.694	0.841	0.910	0.847
Ours, 10 classes	0.768	**0.734**	0.912	**0.824**	0.601	0.684	0.849	**0.912**	**0.854**

consistent with the ground truth, we also visualize hair regions with "soft masks" generated by unary probabilistic output maps, while keeping the other classes with "hard masks", as shown on the fourth row of Figure 3.113.12. Our approach generates accurate labeling results on each facial component (second row) compared to the ground truth (third row). Specifically, it generates visually pleasant labeling results in some challenging cases (even for human beings), as shown in the sixth and seventh column.

3.4 Summary

We propose a deep convolutional network that jointly models pixel-wise likelihoods and label dependencies through a multi-objective learning method. We introduce a non-parametric prior, combined with the RGB image together as input to the network, and show that this prior provides a strong regularization to the network, so that we can use a much smaller model to achieve a competitive performance. Experiments on face labeling tasks show that the proposed multi-objective learning and the nonparametric prior significantly improves the labeling performance.

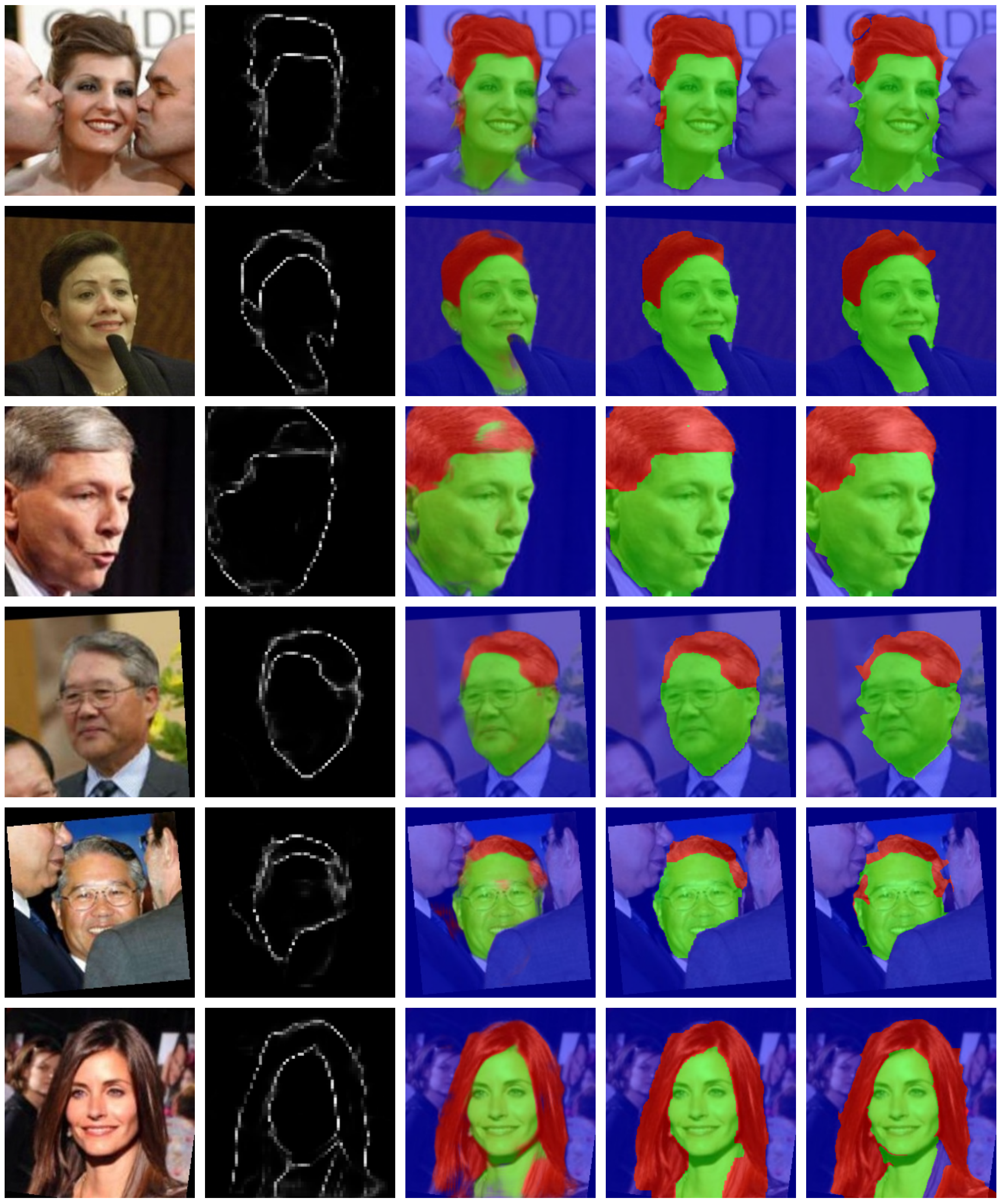


Figure 3.8: Face labeling results and semantic edge maps from LFW-PL dataset. (a) test images; (b) edge term output; (c) unary term output; (d) labeling result by GraphCut inference, denoted as GC; (e) ground truth. Best viewed in colors.

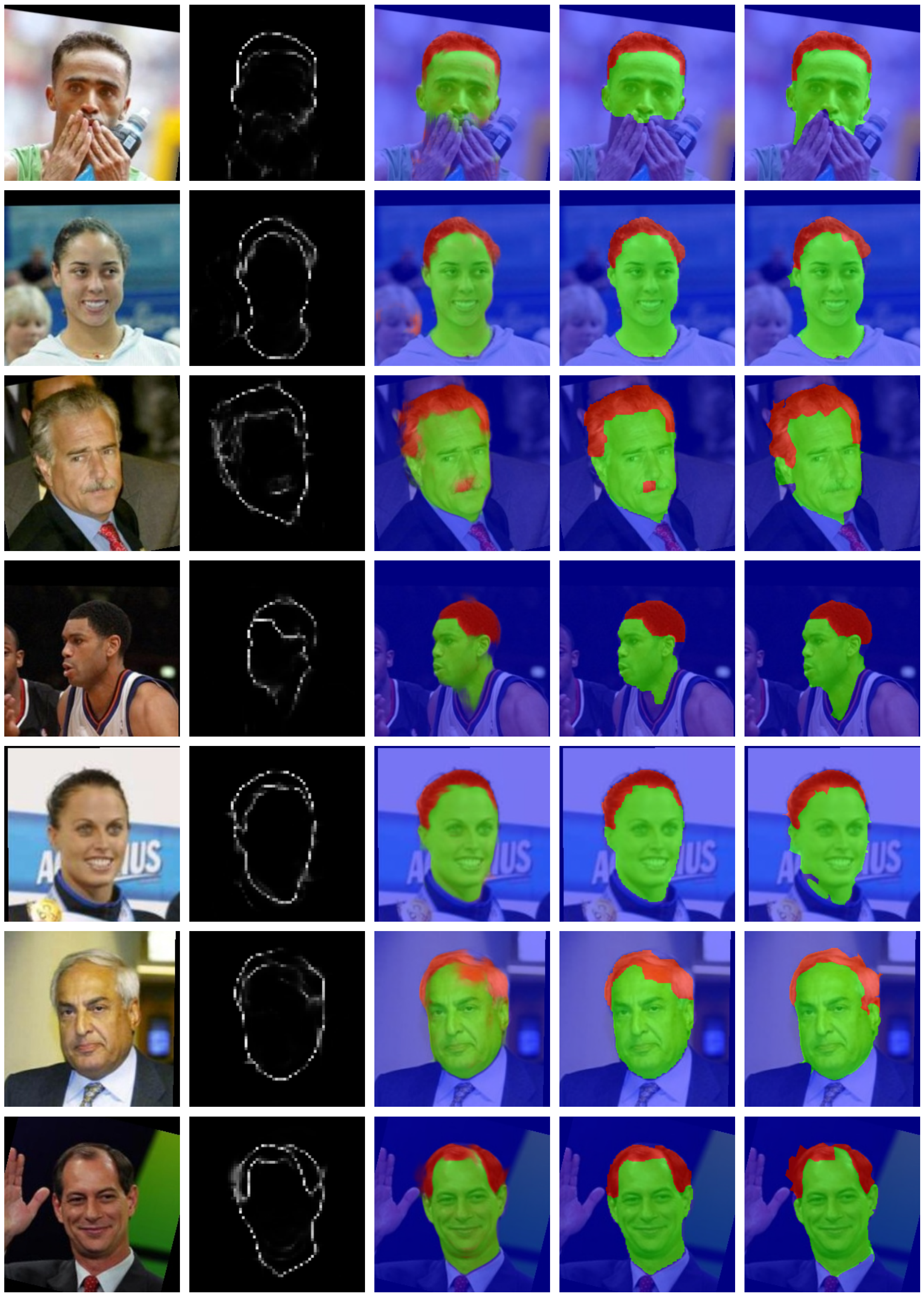


Figure 3.9: Face labeling results and semantic edge maps from LFW-PL dataset. (a) test images; (b) edge term output; (c) unary term output; (d) labeling result by GraphCut inference, denoted as GC; (e) ground truth. Best viewed in colors.

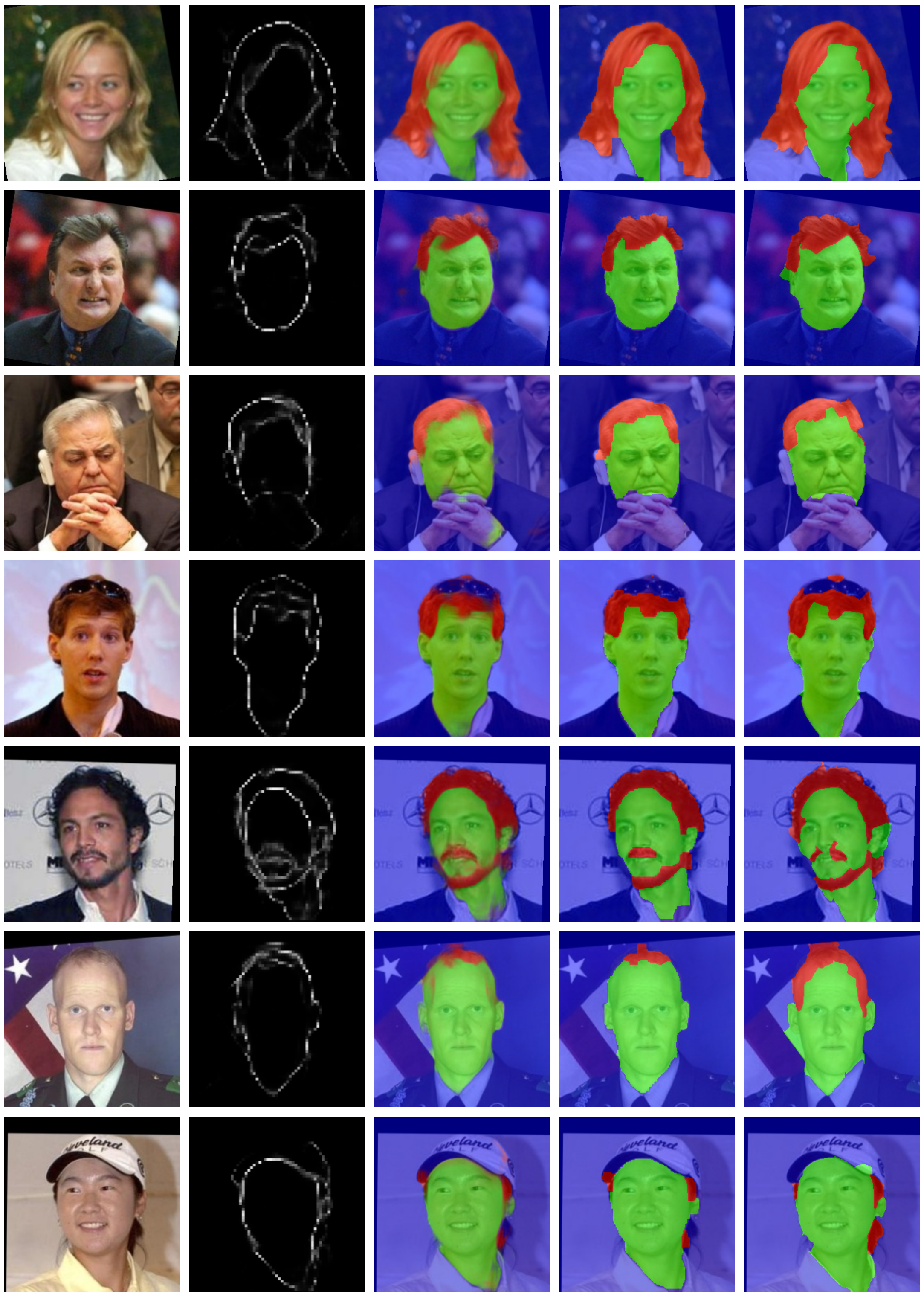


Figure 3.10: Face labeling results and semantic edge maps from LFW-PL dataset. (a) test images; (b) edge term output; (c) unary term output; (d) labeling result by GraphCut inference, denoted as GC; (e) ground truth. Best viewed in colors.

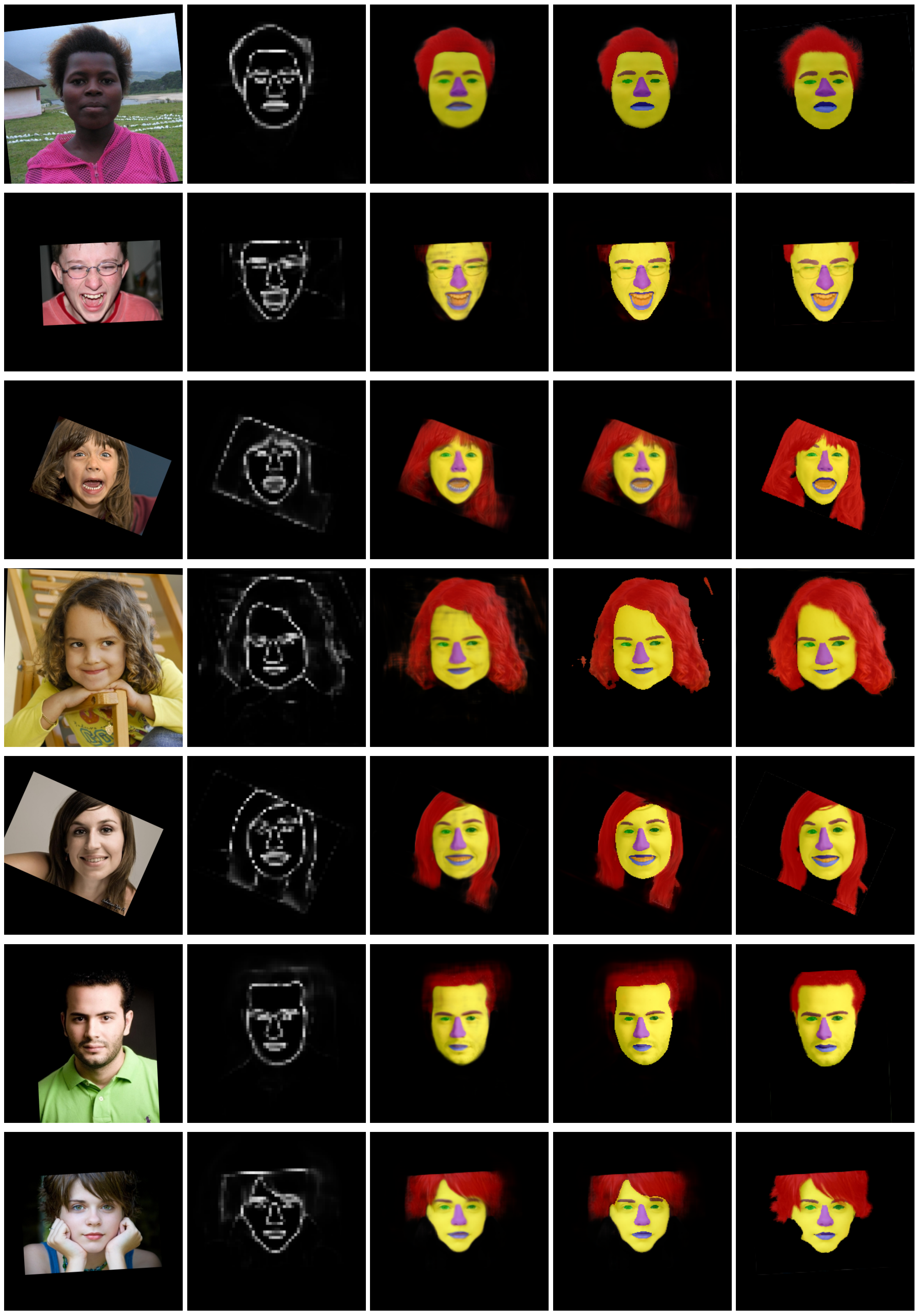


Figure 3.11: Face labeling results and semantic edge maps from the HELEN dataset. GC denotes labeling result by GraphCut inference. Best viewed in colors.

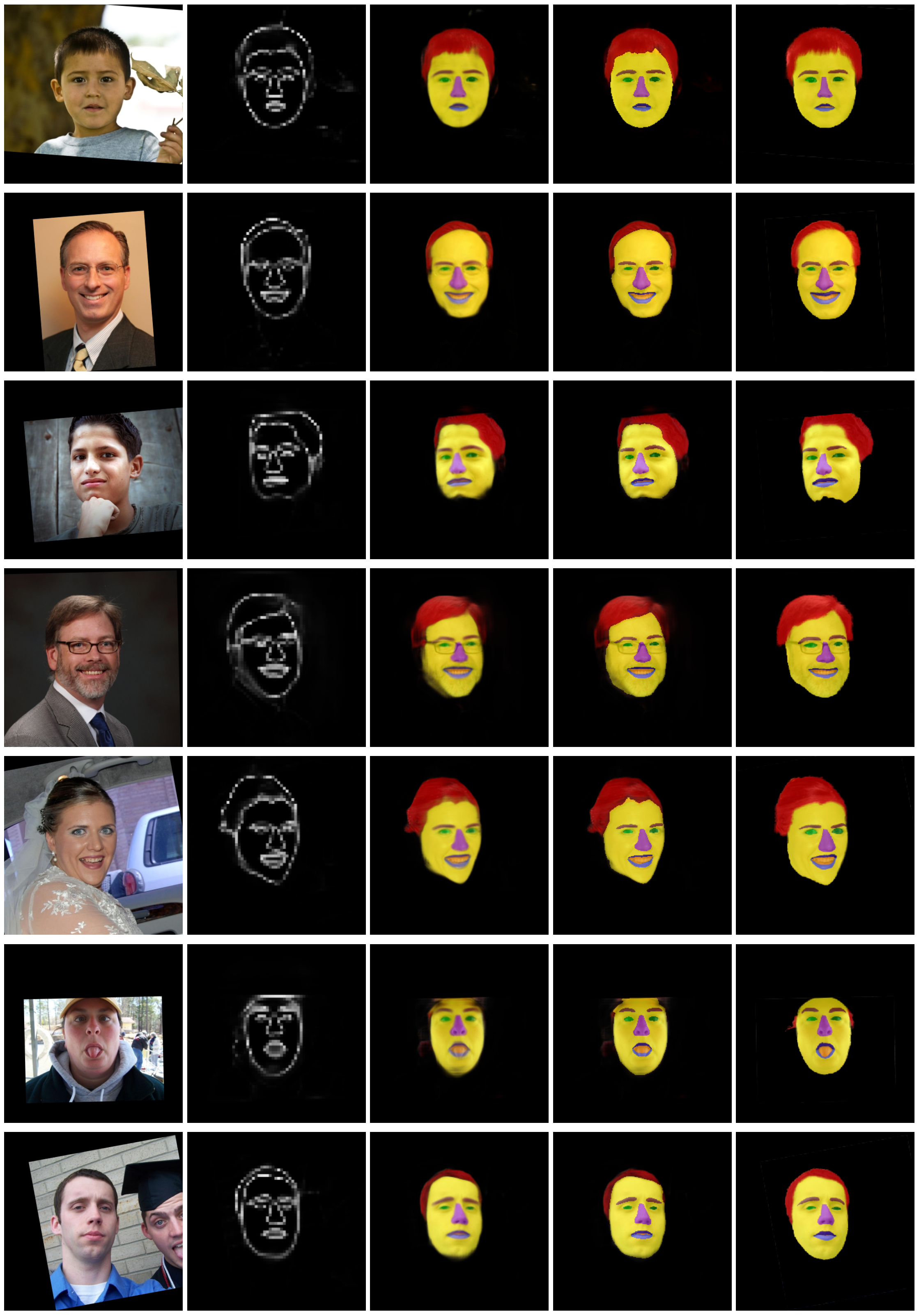


Figure 3.12: Face labeling results and semantic edge maps from the HELEN dataset. GC denotes labeling result by GraphCut inference. Best viewed in colors.

Chapter 4

Learning Recursive Filters for Low-Level Vision via a Hybrid Neural Network

4.1 Introduction

Recursive filters, also called Infinite Impulse Response (IIR) filters, are efficient algorithms that account for signals with infinite duration. As such, recursive implementations are commonly exploited to accelerate image filtering methods, such as spatially invariant/variant Gaussian filters [22, 120, 99], bilateral filters [119] and domain transforms [32]. However, few methods are developed based on recursive formulations for low-level vision tasks mainly due to the difficulty in filter design.

Recently, several deep ConvNet based methods have been proposed for low-level vision tasks [112, 113, 82, 83, 24]. A convolutional filter can be considered equivalent to a finite impulse response (FIR) filter. Unlike IIR filters, it is easier to design FIR filters at the expense of using more parameters to support non-local dependency. In deep ConvNets, Xu et al. [113] approximate a number of edge-preserving filters using a data-driven approach

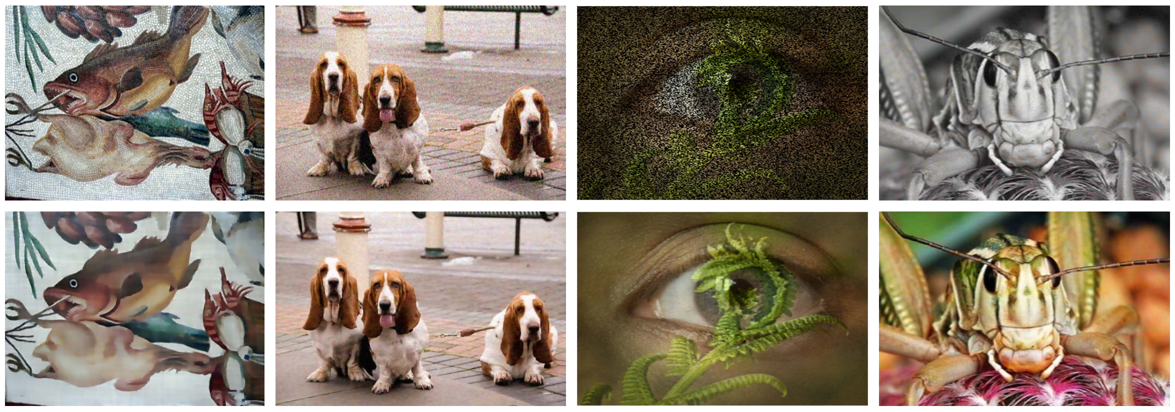


Figure 4.1: Several applications of the proposed algorithm. (a) Approximation of relative total variation (RTV) [115] for edge-preserving smoothing. (b) Denoising. (c) Restoration of an image with random 50% pixels occluded. (d) Restoration of an image with only 3% color informations retained.

which can utilize hundreds of convolutional channels to support spatially variant filtering or large (up to 16×16) kernels to support global convolution. In spite of using a large number of parameters, this model does not present local image structures well. Furthermore, it is difficult to extend the deep ConvNet model to other low-level vision problems such as colorization and image completion.

Figure 4.1 shows a number of low-level vision tasks, e.g., denoising and inpainting, which can be efficiently carried out by the proposed algorithm. In this chapter, we incorporate a group of RNNs as an equivalent of a recursive filter. As an important class of neural networks, RNNs have been used for modeling contextual information in sequential data [36, 9, 105]. The linear formulation of a RNN is equivalent to a first order recursive filter, and the weight matrix corresponds to the coefficients. In addition, higher order recursive filters can be formulated with several RNNs integrated either in cascade, or in parallel. To design a data-driven RNN filter, a straightforward approach is to take each pixel as a hidden recurrent node in a two-dimensional (2D) spatial sequence [73, 96, 49], and use the recurrent structure to learn the propagation weight matrix. However, a standard RNN uses an invariant weight matrix, which makes all pixels share one single recursive filter. Thus, this approach cannot be directly applied to filters that are conditioned on an input image with spatially variant structures, e.g. edge-preserving smoothing.

To address these issues, we propose a spatially variant RNN by introducing a weight map conditioned on the input image. The map has a set of distinct values for each node which control the node-wise recurrent propagation, or equivalently, each node has a distinct recursive filter. The weight map is associated with an image representation that reveals important structures e.g., salient edges (useful for edge-preserving smoothing and denoising). It can be jointly trained through a deep ConvNet that is combined with RNNs in an end-to-end fashion. The proposed hybrid network is shown in Figure 4.3, which exhibits significant differences from existing pure data-driven ConvNet models [112, 113, 82, 83, 24]. It is worth emphasizing that the ConvNet is not used to extract hierarchical image features, but to learn the coefficients of RNNs. We show that a variety of low-level vision tasks can be carried out as recursive image filtering by the proposed neural network.

The contributions of this chapter are summarized as: (a) A hybrid neural network is proposed to learn recursive filters for low-level vision tasks. The network contains several spatially variant RNNs as equivalents of a group of distinct recursive filters for each pixel, and a deep ConvNet that learns the weights of the RNNs. (b) The deep ConvNet effectively guides the propagation of RNNs through learned regulations in a data-driven fashion. Specifically, the weight map from the ConvNet is highly correlated to the corresponding image structures, which plays an important role in low-level vision problems. (c) The proposed model achieves promising results without any special design, regularization of the coefficients, pre-training or post-processing, and is suitable for real-time applications.

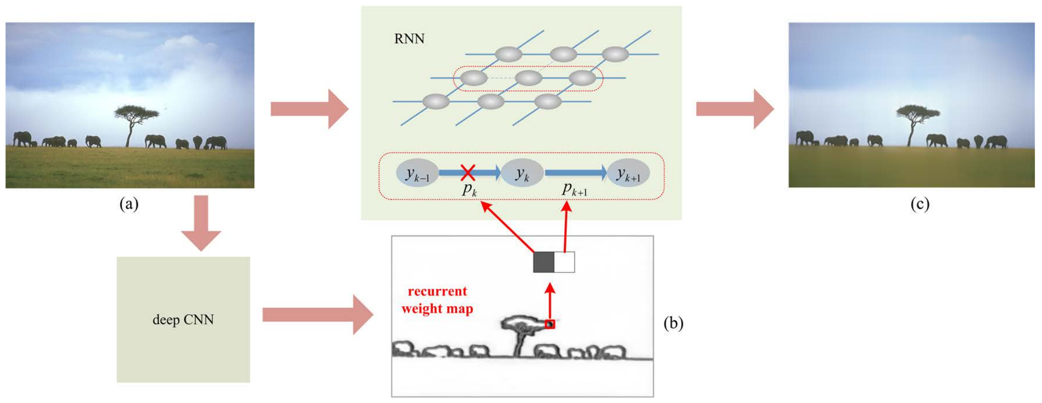


Figure 4.2: An illustrative example of the proposed model for edge-preserving image smoothing with a single RNN. The deep ConvNet generates a weight map (b) that guides the propagation of the RNN. We consider an image as a group of sequences, and take the left-to-right recurrent propagation in 1D as an example, where k denotes a spatial location. For a single RNN, the weight map corresponds to the edges of an image and can be clearly visualized. When p_k is close to zero, it cuts off the propagations from $k-1$ to k so that the edge is preserved (i.e., near boundary). On the other hand, p_{k+1} maintains the propagation from k to $k+1$ so that the image is smoothed at any non-edge location. The ConvNet and RNN are jointly trained and the proposed network can be generalized to many other applications such as colorization, inpainting and denoising (see Figure 3.1).

4.2 Recursive Filter via RNNs

The proposed model contains two parts: a deep ConvNet, and a set of RNNs that take the output of the ConvNet as their input. Different from existing ConvNet based methods [112, 113, 82], the filtered images are generated only through the set of RNNs. The deep ConvNet, on the other hand, does not contribute any features or outputs for the filtered result. Instead, it learns the internal regulations (see Figure 4.2, an example of a single RNN for edge-preserving smoothing) to guide the propagation process for each hidden node. In terms of the network structure, the deep ConvNet does not need to have a large number of channels or large kernels, since it focuses on learning the guidance for recurrent propagation instead of kernels for low-level filters. In comparison to recent deep ConvNet models for [25, 112, 113], the proposed model is much more efficient and light-weighted.

In this section, we describe the algorithmic details of the low-level part in the proposed network. We show that a recursive filter can be equally expressed by a set of RNNs, with its coefficients corresponding to the weight matrices of RNNs. We present two schemes to combine a group of RNNs for constructing a recursive filter, and show how to ensure the stability of the system.

4.2.1 Preliminaries of Recursive Filters

We first review recursive IIR filters [78] before presenting the hybrid neural network. For illustration, we use a one-dimensional (1D) convolution FIR filter, in which the output $y[k]$ is composed of a weighed sum of the input signal $x[k-i]$, expressed in the causal, discrete-time formulation:

$$y[k] = \sum_{i=0}^{M} a_i x[k-i], \quad k = 0, \dots, N, \tag{4.1}$$

where N is the range of the sequence to be filtered, k is one point in the signal which practically corresponds to a frame, character, or pixel in the sequential data. A 1D IIR filter is different in the sense that the output also contains the previously computed values:

$$y[k] = \sum_{i=0}^{P} a_i x[k-i] + \sum_{j=1}^{Q} b_j y[k-j], \; k = 0, \dots, N, \tag{4.2}$$

where $x[k-i]$ is the input and $y[k]$ is the output sequence, $\{a_i, b_i\} \in \mathbb{R}$ are filter coefficients, P and M are the order of convolutional filters, and Q is the order of the recursive filter. A 0-th order IIR filter is reduced to a FIR filter. An IIR filter (4.2) is equivalent to a FIR filter (4.1) by recursively expanding its second term. For an impulse input, the expanded terms can be infinitely long with exponentially decaying coefficients. That is, an IIR filter bypasses a long convolution, with only a few coefficients involved. The causal

IIR system from (4.2) is equivalently described in the z-domain by its transfer function $H(z)$ [78]:

$$H\left(z\right) = \frac{\sum_{i=0}^{P} a_i z^{-i}}{1 - \sum_{j=1}^{Q} b_j z^{-j}}. \tag{4.3}$$

It describes the frequency properties of IIRs independent of specific input signals. The output sequence $y\left[k\right]$ can be obtained from the z-transform of the input signal $X\left(z\right)$ and $H\left(z\right)$ by computing the inverse z-transform of $H\left(z\right)X\left(z\right)$. Note that for causal filters, we need to define the initial conditions of the input signal $x\left[-i\right]$ where $i = 1, ..., P$, and the output signal $y\left[-j\right]$ where $j = 1, ..., Q$. In this work, we set the initial conditions to zero in the training process since we only use up to the second order ($Q \leq 2$). Similarly, we obtain the testing results by padding image borders.

4.2.2 Recursive Decomposition

The Q-th order IIR filter can be decomposited into a set of first order filters in two different forms.

Cascade Decomposition. A recursive filter can be described in the z-plane with poles and zeros [78]. Denoting the poles by $\{p_j\}_{j=1}^{Q}$ and the nonzero zeros by $\{q_i\}_{i=1}^{P}$, we have

$$\begin{aligned} &H\left(z\right) = H_r\left(z\right)H_c\left(z\right), \\ &H_r = \textstyle\prod_{j=1}^{Q} \frac{g_j}{1-p_j z^{-1}},\ H_c = \prod_{i=1}^{P} h_i(1 - q_i z^{-1}), \end{aligned} \tag{4.4}$$

where H_r and H_c are recursive and convolutional parts, g_i and h_j are their coefficients respectively, $\{g, h, p, q\} \in \mathbb{C}$. While H_c is equivalent to an ordinary 0-th order FIR that can be constructed through a convolutional layer, H_r is a cascade of Q first order IIR units. The spatial domain formulation with respect to the j-th unit from sequences of input $x^r\left[k\right]$ and output $y^r\left[k\right]$ is:

$$y_j^r\left[k\right] = g_j x_j^r\left[k\right] + p_j y_j^r\left[k-1\right]. \tag{4.5}$$

We denote this formulation as a cascade decomposition.

Parallel Decomposition. In [33], it is shown that $H(z)$ can be decomposed into a sum of Q first order recursive filters:

$$\begin{aligned} &H(z) = H_r(z) + H_c(z), \\ &H_r = \textstyle\sum_{j=1}^{Q} \frac{g_j}{1-p_j z^{-1}},\ H_c = \sum_{i=0}^{P-Q} h_i z^{-i}, \end{aligned} \tag{4.6}$$

where $\{g, h, p\} \in \mathbb{C}$. Similar to the cascade formulation, the parallel decomposition also contains a FIR H_c with different kernel size $(P - Q + 1)$ of a convolutional layer, as well as Q summed first order IIR units. Each one shares the same formulation as in (4.5). We refer to this formulation as a parallel decomposition.

To simplify the framework, we do not apply H_c from (4.4) and (4.6) in this work. Therefore, the parallel way has $P = Q - 1$, which is greater than the cascade one with $P = 0$ when $Q > 1$. It is more amenable to be designed as a high-pass filter (e.g., for enhancement effect) compared to the cascade connection [78].

4.2.3 Constructing Recursive Filter via Linear RNNs

Single Linear RNN is 1st Order Filter. RNNs have been used to learn sequential data of varying length for various tasks. Let $x \in X$ be the input signal, $h \in H$ be the hidden state, and $\{W_x, W_h\}$ be the weight matrices, then the recurrent relation over space or time is modeled by

$$h[k] = f\{W_x x[k] + W_h(h[k-1] + b)\}. \tag{4.7}$$

The formulation (4.7) is slightly different from the first order recursive filter, as expressed in (4.5), where the sigmoid is often used for f to ensure the output is bounded and the recurrent system is stable in transition.

To model the recursive filter (4.5), we set f as an identity function $f(x) = x$, and $\{W_x, W_h\}$ as diagonal matrixes. We refer to this neural network as the Linear Recurrent Neural Network (LRNN) in this chapter. With this method, we ignore the bias term in (4.7) and formulate LRNN using the dot product:

$$h[k] = g \cdot x[k] + p \cdot h[k-1], \tag{4.8}$$

where $x[k] \in \mathbb{R}^{n\times 1}$. The $\{g, p\} \in \mathbb{R}^{n\times 1}$ can be regarded as the diagonal values of W_x and W_h, where $\cdot$ is a dot product operator.

We further formulate (4.8) in a normalized filter, which has unit gain at some specified frequency. For example, a low-pass filter commonly has unit gain at $z = 1$, which implies that its discrete impulse response should sum to one. Normalizing a filter is carried out by scaling its impulse response by an appropriate factor, where (4.8) is computed by setting $g = 1 - p$ such that the prediction of coefficients is reduced to estimating the parameter p only:

$$h[k] = (1 - p) \cdot x[k] + p \cdot h[k-1]\,. \tag{4.9}$$

Its backward pass can be generalized by back propagation thorough time (BPTT) used in RNNs [38]. The derivations with respect to $h[k]$, denoted as $\theta[k]$ is,

$$\theta[k] = \delta[k] + p \cdot \theta[k+1]\,. \tag{4.10}$$

The stability of LRNN (4.9) is different from the standard RNN (4.7) because the range of $h[k]$ is not controlled through some nonlinear functions (e.g., sigmoid). The output sequence is likely to go to infinity when p is greater than one. According to z-transform [78], the causal recursive system can be stabilized by regularizing p inside the unit circle $|p| < 1$, which we discuss in the next section. In addition, the propagation of (4.9) can reach to a long range when p is close to one, thereby enabling global propagation over an entire image.

Construction of High Order Filters. High order recursive filters [33] can be constructed by combining a group of LRNNs in cascade or parallel schemes as discussed in Section 4.2.2. In the cascade decomposition, LRNNs are stacked with the input signal passing through one to the next. In the parallel approach, each LRNN receives the input signal respectively, where the outputs are integrated with node-wise operations. The FIR terms (which we do not use in this work) can be implemented by convolutional layers that are integrated in the same way.

Two Dimensional Image. To filter an image we need to extend the 1D LRNN in (4.9) to 2D. We adopt a strategy similar to the 4-way directional propagation for two-dimensional data in [14]. First, the 1D LRNN is processed respectively along left-to-right, top-to-bottom and their reverse directions, as shown in Figure 4.3. In any direction, we treat each row or column as 1D sequence. Taking the left-to-right case as an example, the LRNN scans each row from left to right. As a result, four hidden activation maps are generated. We integrate the four maps through selecting the optimal direction based on the maximum response at each location. This is carried out by a node-wise max pooling, which effectively selects the maximally responded direction as the desired information to be propagated and rejects noisy information from other directions. We note that the four directions can be executed in parallel for acceleration as they are independent.

4.3 Learning Spatially Variant Recursive Filters

One problem with the standard or linear RNN in (4.7) and (4.9) is that it takes a group of fixed weights for every point k. Filtering an image in such a way means that each pixel is processed with the same recursive filter, which is not effective for many low-level tasks, e.g., edge-preserving smoothing, where the edge and texture areas need to be processed differently.

4.3.1 Spatially Variant Linear Recurrent Network

Therefore, we propose a spatially variant recurrent network by extending the fixed parameter p to $p\,[k]$, so that each pixel has a distinct recursive filter. Taking edge-preserving smoothing as an example (see Figure 4.2), and considering the first order recursive filter (a single LRNN), $\{p\,[k]\}$, namely the weight map, is supposed to be associated with an "edge map". Specifically, the weights that lie on the edge regions should be close to zero such that the input $x\,[k]$ is preserved, and one otherwise so that the other regions can be

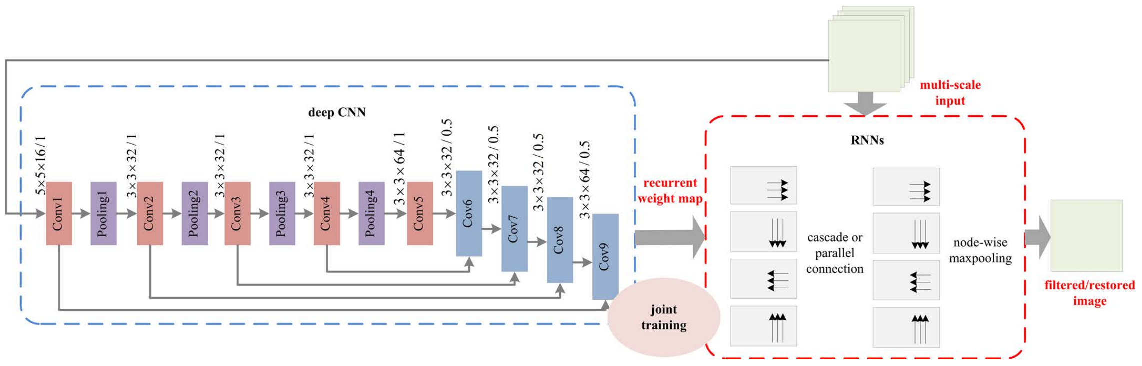


Figure 4.3: Proposed hybrid network that contains a group of RNNs to filter/restore an image and a deep ConvNet to learn to propagate the RNNs. The process of filtering/restoration is carried out through RNNs with two inputs and one output result, denoted in red. Both parts are trained jointly in an end-to-end fashion.

smoothed out via recurrent propagation (as in (4.9)). For higher order recursive filters and some other tasks, e.g., inpainting, the weight maps are more complex and do not correspond to some explicit image structures. However, they reveal the propagation regulations with respect to specific tasks, which are conditioned on the input image.

We have two types of input to a LRNN, i.e., an image X and a weight map P. Given a hidden node $h\left[k\right]$ and similar to (4.9), the spatially variant LRNN is:

$$h\left[k\right] = \left(1 - p\left[k\right]\right) \cdot x\left[k\right] + p\left[k\right] \cdot h\left[k-1\right]. \tag{4.11}$$

In the back propagation pass, the derivative $\sigma\left[k\right]$ with respect to $p\left[k\right]$ is:

$$\sigma\left[k\right] = \theta\left[k\right] \cdot \left(h\left[k-1\right] - x\left[k\right]\right), \tag{4.12}$$

such that the weight map $p\left[k\right]$ of a spatially variant recursive filter can be learned.

4.3.2 Learning Weight Maps of Recurrent Networks via ConvNets

We propose to learn the weight maps through a deep ConvNet, which takes an image to be filtered as its input. The ConvNet can be small and deep, since it learns the guidance of propagation instead of learning convolutional filters. The proposed network is equipped with 10 convolutional layers. The first five layers are followed by a max pooling, while the other five are followed by a bilinear upsampling. The RELUs are used between adjacent convolutional layers. In addition, 4 links between corresponding downsampling and upsampling units connect feature maps of the same size at different levels in order to learn better representations, where similar settings can be found in [66]. We use 3×3 kernels with the number of channels ranging from 16 to 64, as shown in Figure 4.3.

To connect with the LRNNs of different directions (4 distinct hidden layers, see Figure 4.3), the weight map can be equally split into 4 parts for the 4 directions. To simplify the network implementation, each axis is allowed to share the same part (e.g., the left-to-right and right-to-left directions share a common horizontal map). Thus, for each LRNN, we have two parts in a weight map for the x and y-axis, respectively. We find that better results can be obtained by linearly transferring the RGB input of LRNN into a feature space, e.g., through one convolutional layer, and then perform LRNN on the proposed transform space. We are then able to select a best direction at each point on the feature space using a node-wise integration strategy, which combines the four directions. The combined maps can be transferred back to a 3-channel image through another convolutional layer. We configure both of the transform convolutional layers using 3×3 kernels. We set the number of channels in each hidden layer of LRNNs to $m = 16$ in all experiments so that each $x[k]$ and $p[k]$ in (4.11) are vectors with dimension of 16. The number of output channels for ConvNet is $2 \times m \times R$, where R denotes the order of recursive filter (or equivalently the number of LRNNs), e.g., it should be set to 64 with a network configured with a $2nd$ order recursive filter. It is important that we equip a hyperbolic tangent function as the topmost layer of the ConvNet, so that the weight map is restricted to $(-1, 1)$ to stabilize the LRNN, as introduced in Section 4.2.3.

4.4 Experimental Results

We apply the proposed model to a variety of low-level vision problems including edge-preserving smoothing, enhancement, image denoising, inpainting and colorization. All the following applications share the same model size as well as the run-time. Specifically, our model reaches real-time performance on images of 320×240 pixels (QVGA) using a Nvidia Geforce GTX Ti GPU with 3 GB memory. Due to space limitations, we present some results in this section. During the training phase, the momentum, weight decay and batch size are set to be 0.9, 10^{-3}, and 20, where the initial learning rate is set as 10^{-4}. Specifically, our model takes 0.55 and 0.88 ms for an input image with $1080p$ or $2k$ resolution, respectively. The corresponding run time performance of the ConvNet filter can be found and compared in [113]. The trained models and source code is available at `www.sifeiliu.net/project`.

Experimental Settings. To obtain rich information from different scales of an image, we use multi-scale input through downsampling the color image with ratio of $\{1/2, 1/4, 1/8, 1/16\}$, resizing them to the original size, and concatenating them to be a single input. Therefore, nodes in a LRNN can reach to a more global range via processing on coarse scales, without increasing the number of coefficient maps to be learned. We use 96×96 image patches as the original inputs that are randomly cropped from training images, which are then processed as multi-scale input through average pooling and upsampling. All patches are augmented through perturbation using the similarity transform, so as to adapt to the scale-variant property for some existing filters. We use roughly $400,000$ image patches that are randomly cropped from the MS COCO dataset [57] in the training process with data augmentations. For all the following applications, the order of filter is set to 2 with specific structures shown in Figure 4.3. The only difference lies in the integration manner with respect to these 2 LRNNs, e.g., cascade or parallel, which is specified in each application.

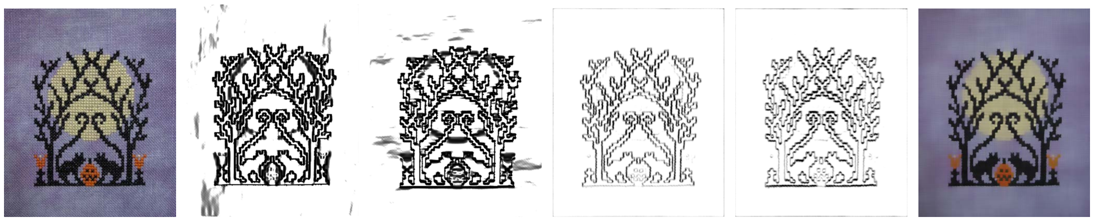

(a) original (b) RTV-x (c) RTV-y (d) weight-x (e) weight-y (f) our result

Figure 4.4: Visualization of weight maps for approximating the RTV filter using first order recursive filter. (a) original image; (b) and (c): manually designed edge prior maps in RTV for x and y axes; (d) and (e): weight maps generated from the ConvNet for x and y; (f) our filtered result.

4.4.1 Edge-Preserving Smoothing

Xu et al. in [113] propose a ConvNet model to approximate various filters such that many conventional implementations can be accelerated significantly. We show that the proposed algorithm is able to approximate various filters and performs favorably against [113] in terms of accuracy, run time, and model size. We selectively learn a group of local and global filters including bilateral filter (BLF) [100], weighted least square (WLS) [29], L0 smoothing [111], RTV texture smoothing [115], weighted median filter (WMF) [122], and rolling guidance filter (RGF) [121].

Visualization of Weight Maps. We first demonstrate through a first order recursive filter using a single scale RGB image without any linear transformation as the input to both ConvNet and LRNN, where the weight maps with respect to x and y axes accurately correspond to the edges of the image. This is carried out by setting the number of output channels of the ConvNet to 2, such that the maps for x and y axes, which are then shared by all channels of the hidden layers in the LRNN, can be obtained and visualized.

We note that some edge-preserving methods, e.g., RTV [115], focus on extracting the main structures of an image. The designed edge prior maps for RTV (Figure 4.4(b) and (c)), which reflect the main structures of an image, determine whether the image regions

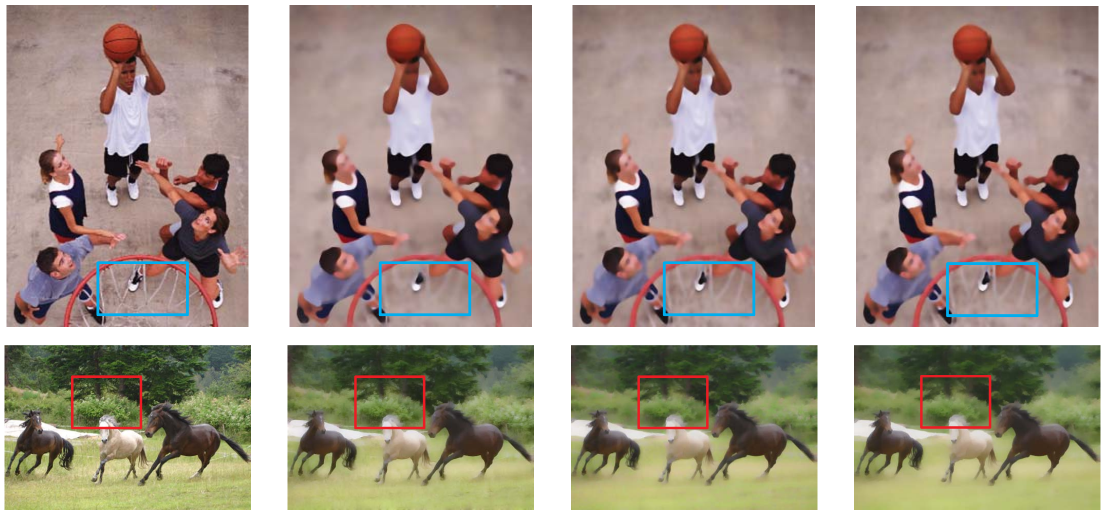


Figure 4.5: Approximation of edge-preserving filters. (a) input images. (b) results by Xu et al. [113]. (c) results of our model. (d) results from the original filters. First row: Results by approximating RGF [121]. Second row: Results by approximating WLS smoothing [29].

should be smoothed or not in the propagation step [115]. Interestingly, the learned data-driven weight maps by our model (see Figure 4.4(d) and (e)) have the similar effects to the handcrafted maps. They accurately locate the image edges with cleaner background, and effectively remove the grid-like texture in the input image, as shown in Figure 4.4(f). As our method is data-driven, different weight maps can be generated for different tasks. The data-driven approach allows the proposed algorithm to be generalized to a variety of applications without handcrafted priors. Similar weight maps can be generated through approximating other edge-preserving filters (e.g., L0 filter [111]), which is not designed based on edge prior, as shown in Figure 4.10. While one can also manually design the weight maps and feed them to the RNNs to create new type of filters, it is beyond the scope of data-driven approach and not be discussed in this chapter.

Table 4.1: Quantitative evaluations for learning various image filters.

Methods	L_0 [111]	BLF [100]	RTV [115]	RGF [121]	WLS [29]	WMF [122]	Shock filter [74]
PSNRs of [113]	32.8	38.4	32.1	35.9	36.2	31.6	30.0
Our PSNRs	30.9	**38.6**	**37.1**	**42.2**	**39.4**	**34.0**	**31.8**
SSIM of [113]	0.99	**0.99**	0.98	**0.99**	0.98	0.98	**0.97**
SSIM ours	0.97	**0.99**	**0.98**	**0.99**	**0.99**	0.97	**0.97**

Quantitative Comparisons. We show the applications that are based on a second order filter. Specifically for edge-smoothing tasks (e.g., L0, WLS and RTV, etc.), the two LRNNs are connected in cascade since it is more amicable to low-pass filtering. On the other hand, we use the parallel integration scheme for learning shock filters [74] with enhancement effects. We quantitatively evaluate the proposed algorithm against [113] on the dataset used in [113]. Table 4.1 shows that our method generates high quality filtered images with significant improvements over the state-of-the-art ConvNet based method. In addition, the proposed model is much smaller and faster due to its hybrid structure, which can be used to accelerate more conventional algorithms, e.g., region covariance filter (RegCov) [50] and local laplacian filter (LLF) [75].

Figure 4.5 shows approximations of RGF [121] and WLS smoothing [29]. The results by our model preserve more accurate structures without including details that are supposed to be removed. The filtered images are visually the same as those generated by the original implementations. We note that the ConvNet based filter [113] misses important local structures by approximating the RGF, and includes some details that should be removed by approximating the WLS, as shown in Figure 4.5(b). We also show more qualitative results for the approximation of other edge-preserving/enhancement filters to demonstrate the effectiveness of the proposed method. Specifically, we crop one patch for each image in visualizing the approximation of shock filter (see Figure 4.15), for better comparisons with respect to the region details.

Run Time and Model Size We evaluate all the following methods with the same computer

Table 4.2: Run-time (second) performance against [113] and some conventional methods at different resolutions of color images.

method	QVGA	VGA	720p
BLF [100]	0.46	1.41	3.18
WLS [29]	0.71	3.40	11.38
RTV [115]	0.81	3.51	9.94
WMF [122]	0.67	1.70	3.80
EPLL [128]	33.82	466.79	1395.61
Levin [53]	2.10	9.24	31.09
Xu et al. [113]	0.23	0.83	2.10
Ours	**0.05**	**0.16**	**0.37**

introduced in the beginning of this section. The proposed method achieves favorable speed as shown in Table 4.2, and is significantly smaller than that of [113] (0.54 vs 5.60 MB). It can speed up a variety of conventional filters for denoising, inpainting and colorization, etc.

4.4.2 Image Denoising

The proposed method can be used to learn filters for image denoising. Specifically, we train the model with thousands of patches in which white Gaussian noise with the standard deviation of 0.01 is added. At the output end, the model is supervised by the original image patches. We apply the parallel connection to the two LRNNs to preserve more details. The other settings are the same as those used in Section 4.4.1.

Figure 4.6 shows the results with two state-of-the-art algorithms including expected patch log likelihood (EPLL) [128] and a deep ConvNet based model [83]. The denoising method [128] is based on a prior of image patches, and the vectorization-based deep ConvNet [83] is based on a two-layer convolutional model. Although significant noise has been removed by both methods, some details are not preserved well and the restored results can be over-smoothed. The learned filter by the proposed model generates clear images with well preserved fine details, as shown in Figure 4.6(d). It retains important image contents such as the brushstrokes of oil painting in the first row, or pattens of the feather in the second row.

We apply the test set of Berkeley segmentation dataset 500 (BSDS500) which contains 200 natural images, and compare the proposed algorithm with the state-of-the-art methods, including EPLL [128], bm3d [20] and deep ConvNet based model [83].

In Figure 4.16, several patches are cropped for better visualization and comparisons. The EPLL algorithm over-smooths many regions (in all examples) especially on the background, and introduces color noise (being obvious on the first and third columns). The ConvNet based method preserves more details. However, it produces more texture-like noise on smooth regions. Comparatively, the results generated by the proposed algorithm (see Figure 4.16, the 4-th row) are visually pleasant on both preserving details and removing noise.

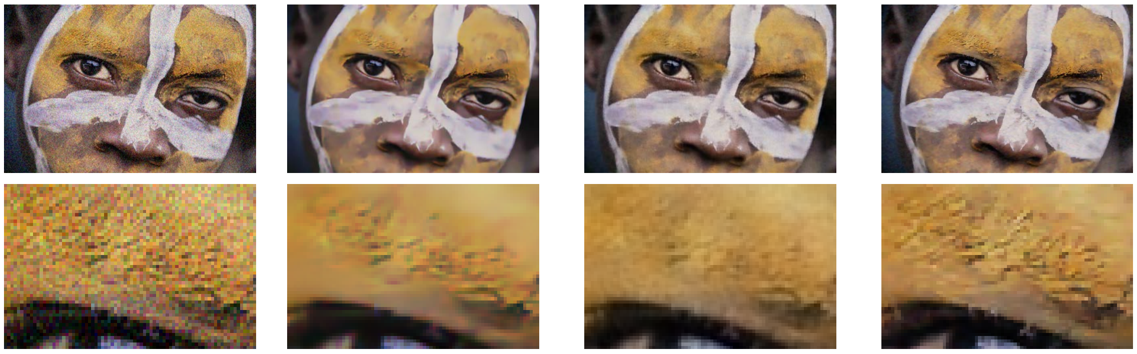
Table 4.3: Quantitative evaluations for image denoising on BSDS500-test.

Methods	EPLL [128]	deep ConvNet [83]	bm3d [20]	ours
Average PSNRs	28.38	28.82	28.38	**31.05**

The deep ConvNet method is likely to be slower in terms of run-time (was not specified in [83]) due to its large model size, while the EPLL takes more than hundreds of seconds to process one image. In contrast, the proposed method achieves several order of magnitude accelerations (see Section 4.4.1). As a result, it outperforms the state-of-the-art methods in terms of the overall performance as well as efficiency.

4.4.3 Image Propagation Examples

In this section, we validate the effectiveness of propagation-study of the network by restoring images from degraded frames with masks. The deep ConvNet here learns more complex rules than the edges that are used for smoothing. We apply the proposed model


(a) Input (b) EPLL, PSNR: 31.0 (c) ConvNet, PSNR:31.0 (d) Ours, PSNR: 32.3

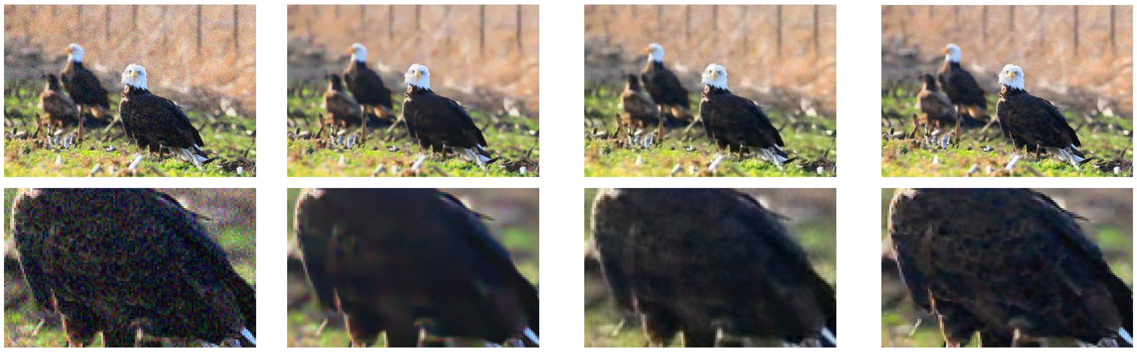

(a) Input (b) EPLL, PSNR: 31.1 (c) ConvNet, PSNR: 29.5 (d) Ours, PSNR: 31.6

Figure 4.6: Image denoising. (b) denotes the results of image patch prior based method EPLL [128]. (c) denotes the results by end-to-end trainable ConvNet method [83].

to two interesting applications for pixel and color interpolation (e.g., inpainting and colorizaiton). Specifically, we retain randomly 50% pixels for the image interpolation and 3% monochrome pixels for the color interpolation. The proposed model takes degraded images as well as masks as input channels, and learns the weight maps with the supervision of the original images. It learns complex regulations including identifying the occluded pixels and restoring them by propagating information from the other pixels, and identifying the image structures such that the restored pixels can naturally adapt to them.

Pixel Interpolation. The goal of pixel interpolation is to restore the values in missing regions according to a mask of pixels that are to be restored. In this model, the random mask is concatenated with the degraded image as the input, such that it learns the propagation rules according to all the visual information. The LRNNs filter the degraded image according to the learned rules and output an interpolated result. It does not require explicit regulations to compute the missing data, nor expensive optimizations for each test image. Therefore, it is accurate and fast to execute through forward propagation.

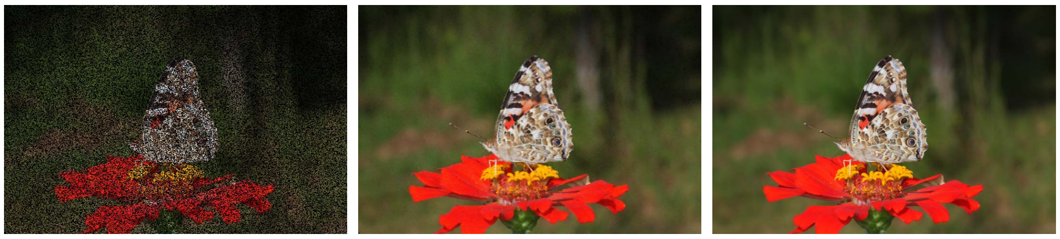

(a) occluded (b) restored (c) original

Figure 4.7: Pixel interpolation. (a) input image. (b) restored image for masking half pixels in (a).

We show that the proposed algorithm can restore fine details (e.g., pattens on a butterfly) in Figure 4.7 with randomly half pixels are masked. We discover that the proposed model trained for image interpolation can be directly applied to image inpainting with texts, as shown in the first row of Figure 4.8. Both results are visually very similar to the original images, as shown in Figure 4.7(b) and 4.8(c).

For ease of comparison, we show all results by different methods on one page, and the details can be clearly viewed at the original image resolution, or equivalently by zooming in on Figure 4.17. The EPLL algorithm can recover the edges but over-smooths many details (in all examples). The ConvNet based method, on the other hand, produces jagged boundaries (e.g., edges along houses on the hill in (a), long edges in (b)). Comparatively, the results generated by the proposed algorithm (fourth row of Figure. 4.17) are visually pleasant on both detail and edge preserving, and are visually similar to the ground truth images (fifth row of Figure. 4.17).

Color Interpolation. The proposed algorithm can be applied to color image restoration

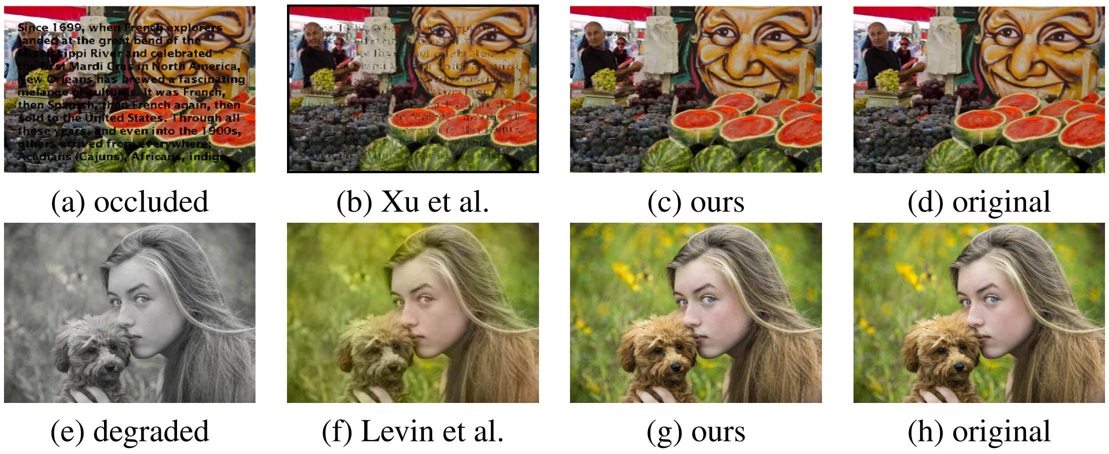


Figure 4.8: First row: image inpainting on the regions of texts with comparison to Xu et al. [82]. We directly apply the pixel interpolation model to inpainting. The model does not require any network finetuning on texts masks. Second row: color interpolation with comparison to Levin et al. [53].

and editing despite providing little color information, e.g., user inputs. Given the brightness channels (y channel in the YCbCr color space), we retain only 3% color pixels, as shown in Figure 4.8(e). Taking a degraded image and a mask as input, the proposed model learns to propagate the known colors to other regions to be restored. We compare the results of the proposed algorithm with those generated by the state-of-the-art method [53] in Figure 4.18, and show more results in Figure 4.19. The proposed model generates favorable results (visually the same with the original image) compared to the state-of-the-art method [53], which takes more than 3 seconds on a QVGA image.

The proposed model can also be generalized to image re-colorization by applying the brightness channel of an input image, and directly taking 3% color pixels from the monochrome channels in a reference image of the same size. The re-colored image has the contents of the original image, but with the color style of the reference image. Figure 4.9 shows examples of image re-colorization.

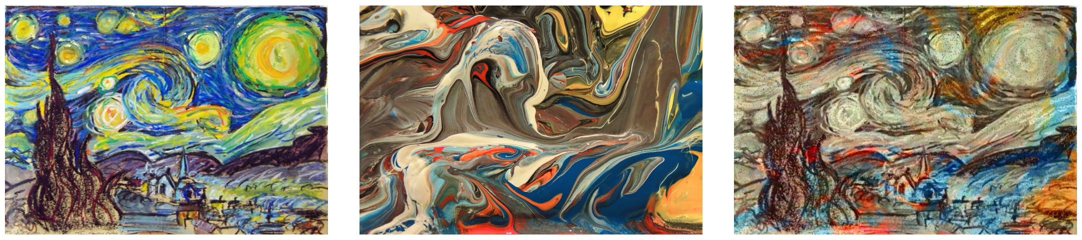

(a) origin (b) reference (c) re-colored

Figure 4.9: Re-colorization by applying the brightness channel of (a) and directly taking 3% color pixels from the monochrome channels in a reference image with the same size.

4.5 Summary

In this chapter, we propose a novel hybrid neural network for low-level vision tasks, based on the recursive filters whose coefficients can be learned by a deep ConvNet. We show that the proposed model is faster and significantly smaller than the deep ConvNet filters. It is also more generic, and can effectively and efficiently handle a variety of applications including image smoothing and enhancement, image denoising and pixel interpolation.

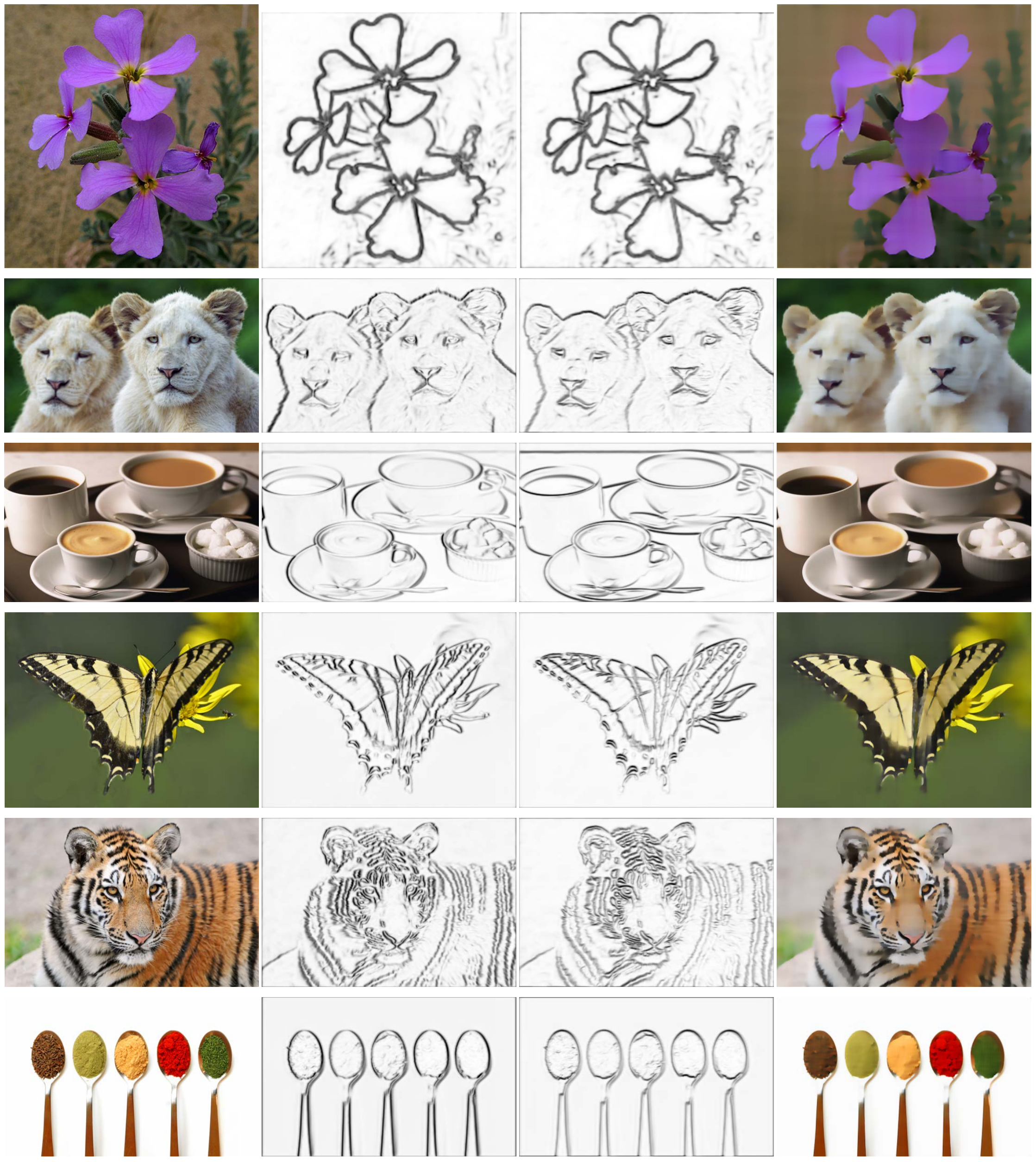
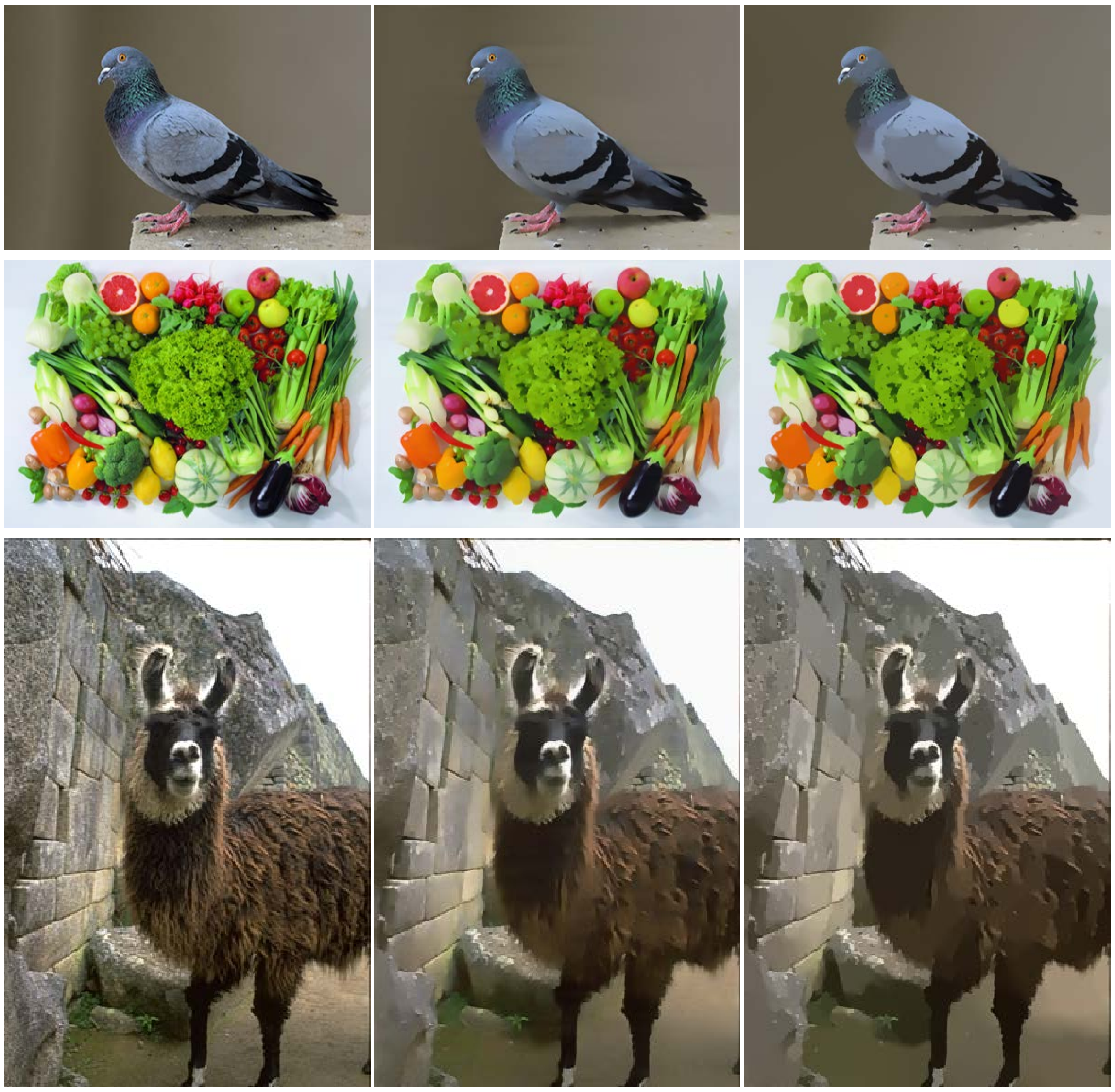
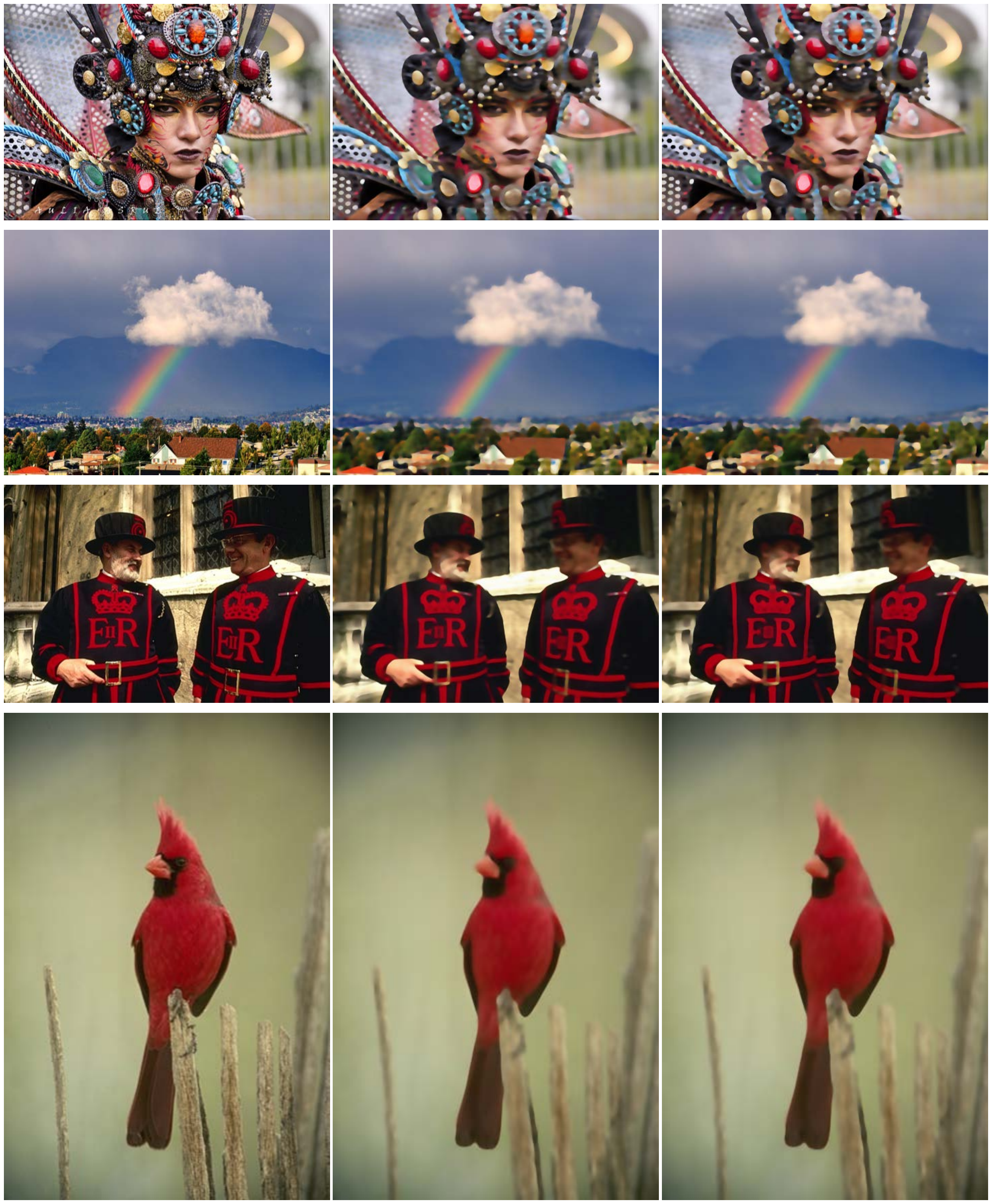

(a) input (b) x-map (c) y-map (d) smoothed

Figure 4.10: Visualization of weight maps for L0 edge-preserving smoothing filter [111].



(a) input (b) proposed (c) L0

Figure 4.11: Approximation of L0 edge-smoothing method [111]. Zooming in to see details.



(a) input (b) proposed (c) RGF

Figure 4.12: Approximation of RGF [121] edge-smoothing method. Zooming in to see details.

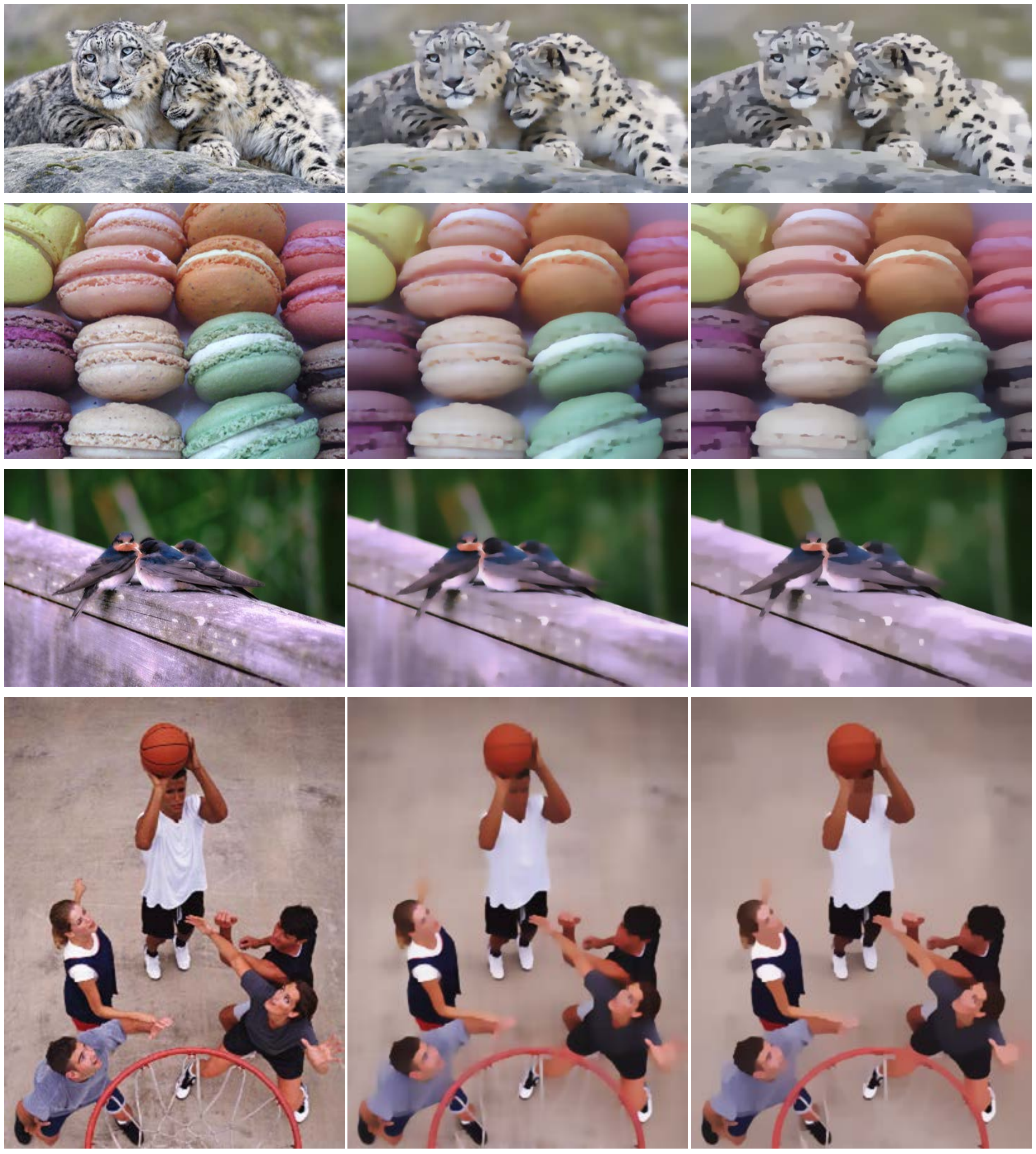

(a) input (b) proposed (c) RTV

Figure 4.13: Approximation of RTV [115] edge-smoothing method. Zooming in to see details.

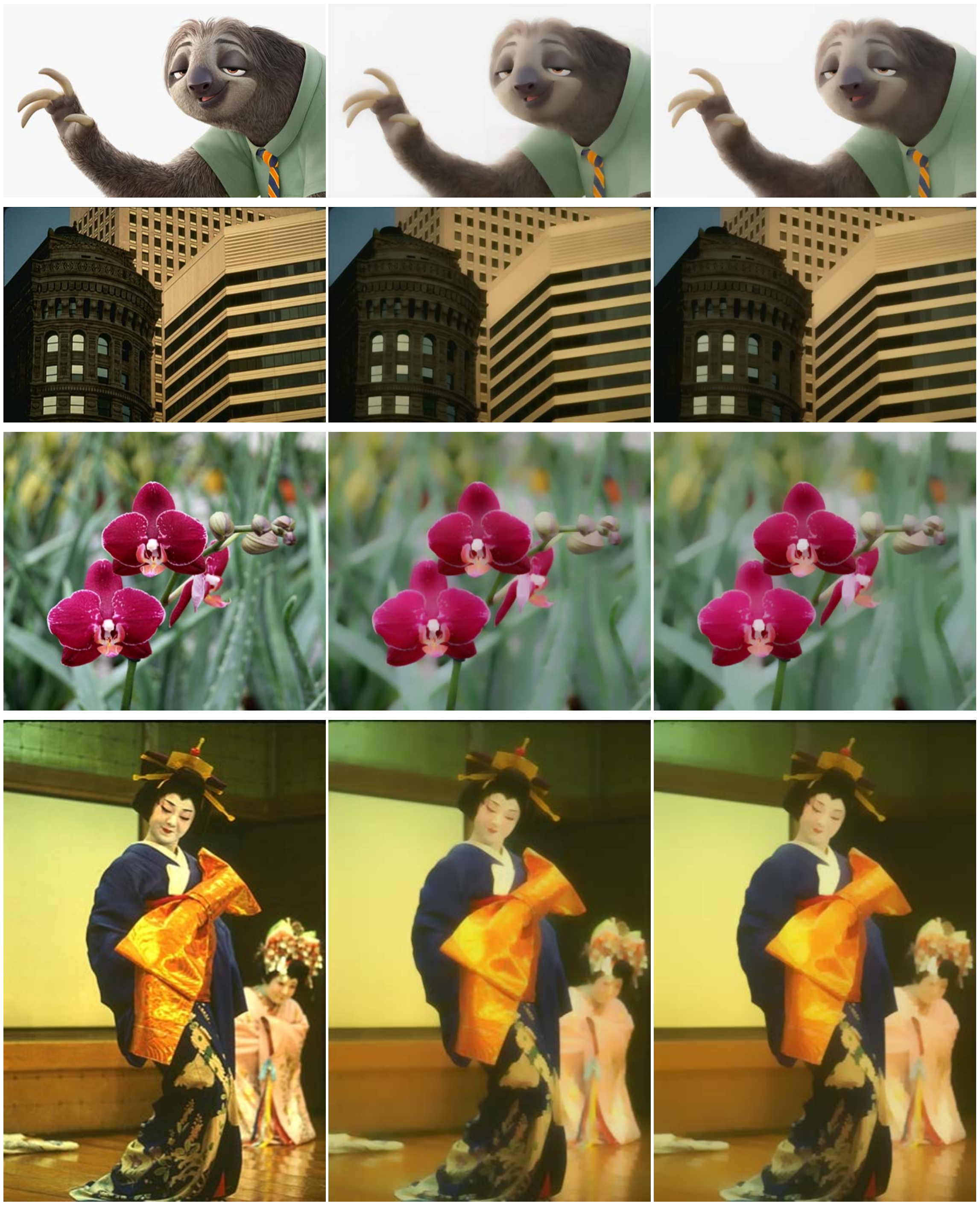
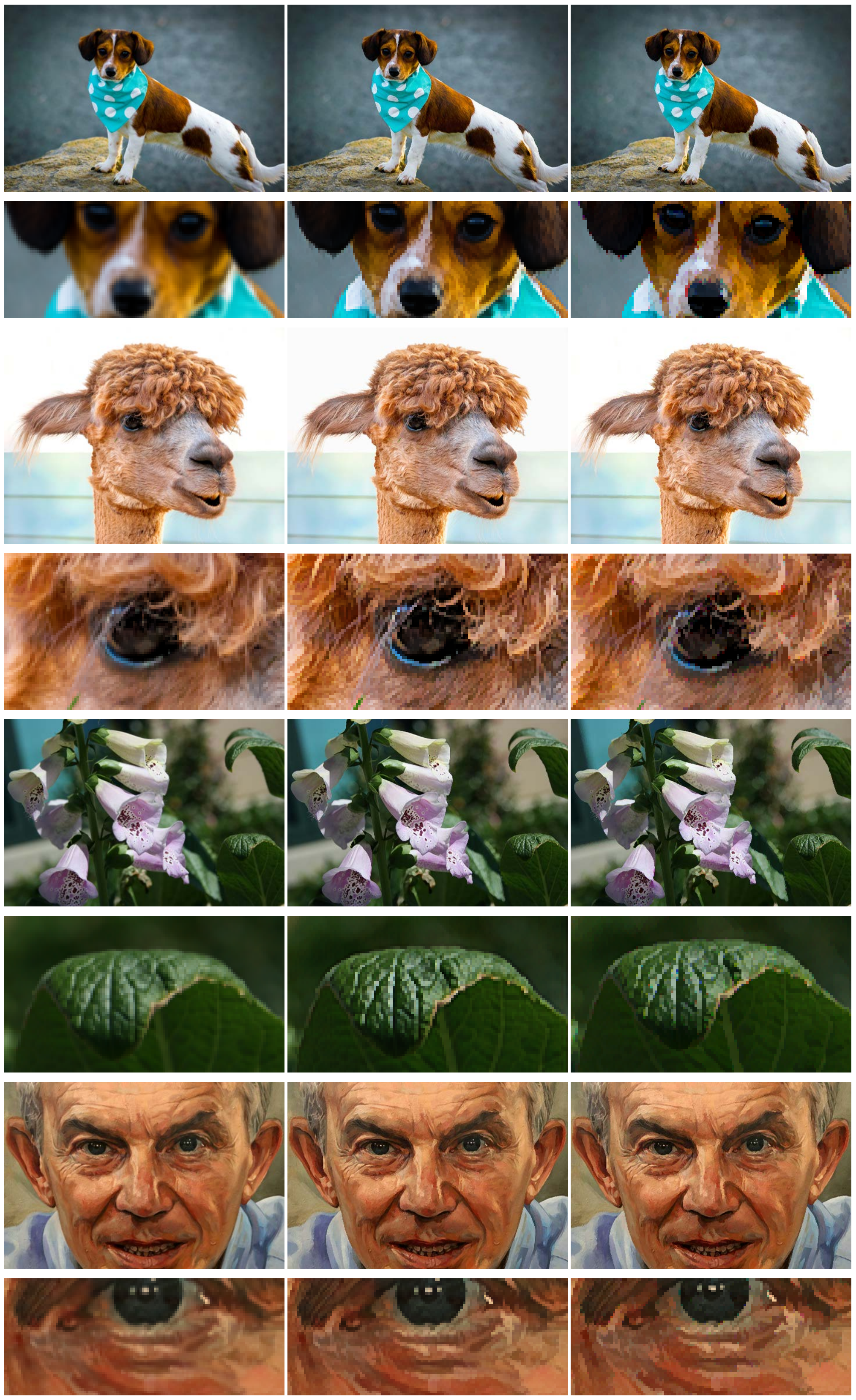
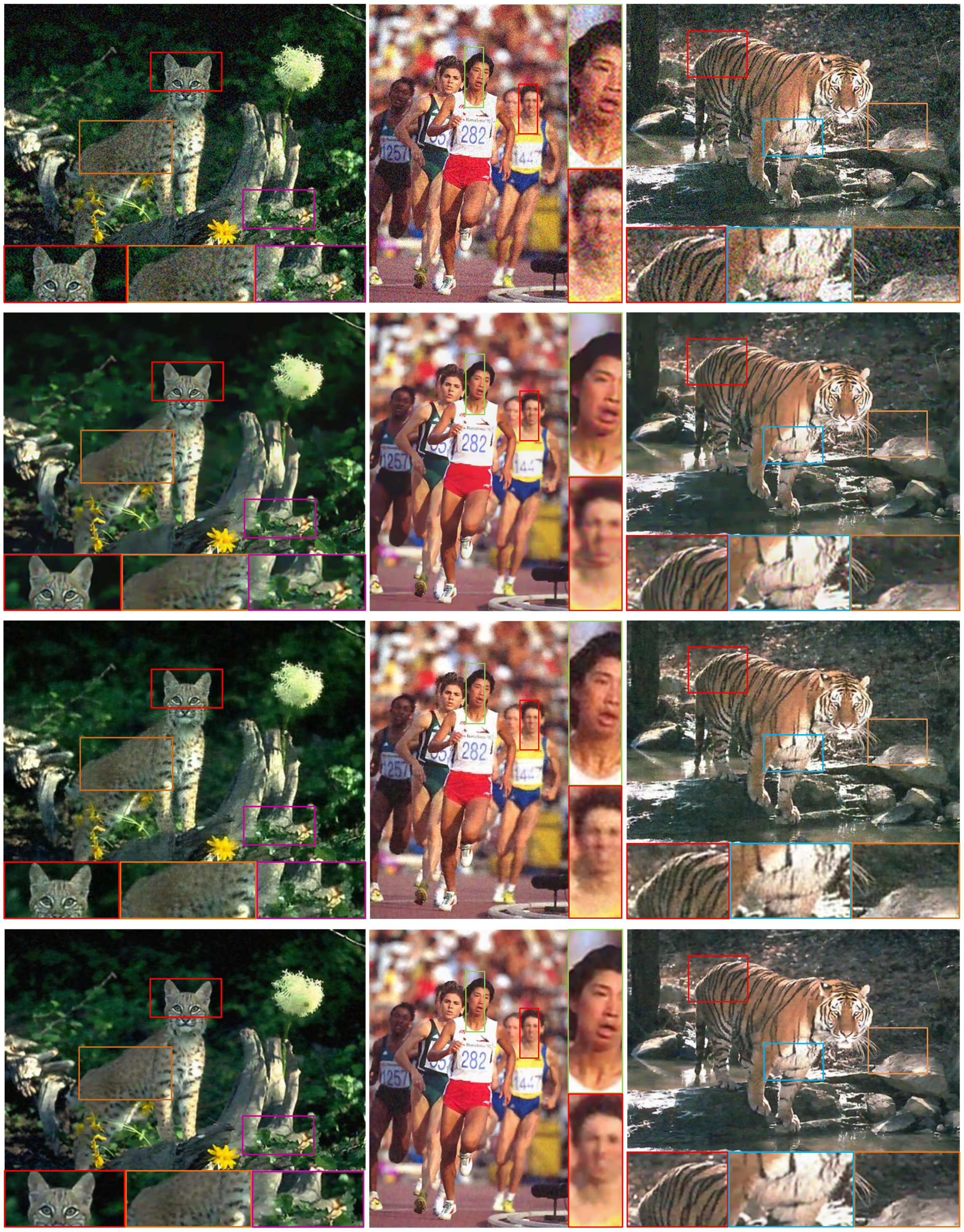


Figure 4.14: Approximation of WLS [29] edge-smoothing method. Zooming in to see details.



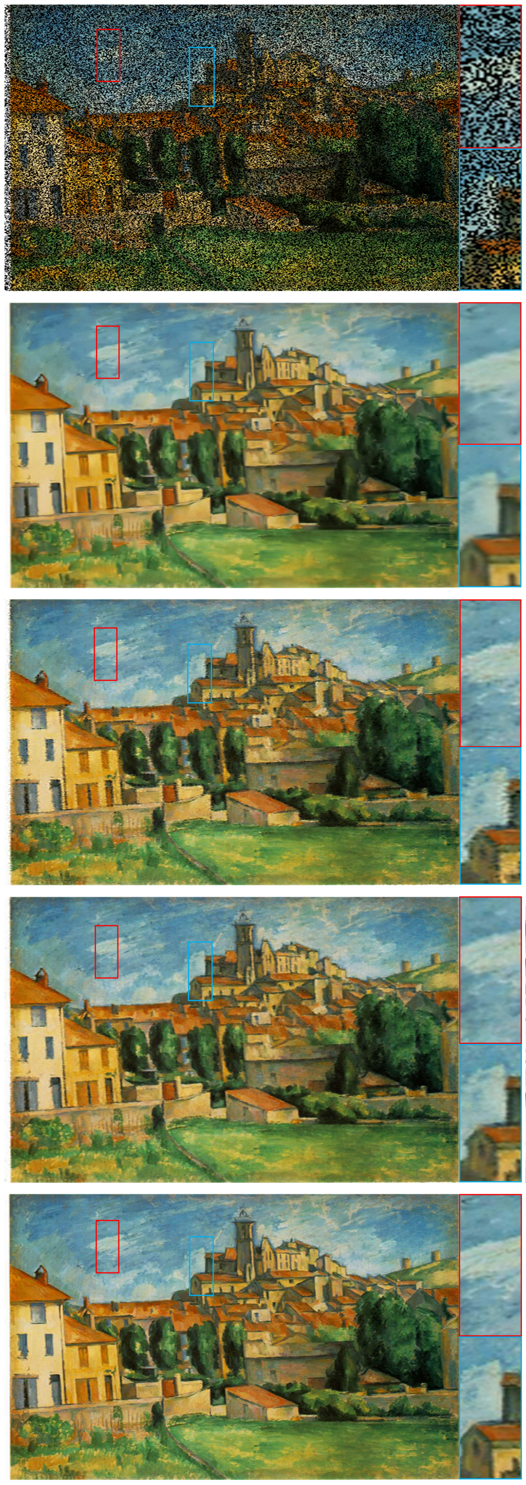
(a) input (b) proposed (c) shock

Figure 4.15: Approximation of Shock filter [74] image enhancement method. Zooming in to see details.



(a) (b) (c)

Figure 4.16: Image denoising. First row: image with white Gaussian noise; Second row: image denoised by EPLL [128]; Third row: image denoised by deep ConvNet based method [83]; Forth row: image denoised by the proposed algorithm. Zooming in to see details.


(a)

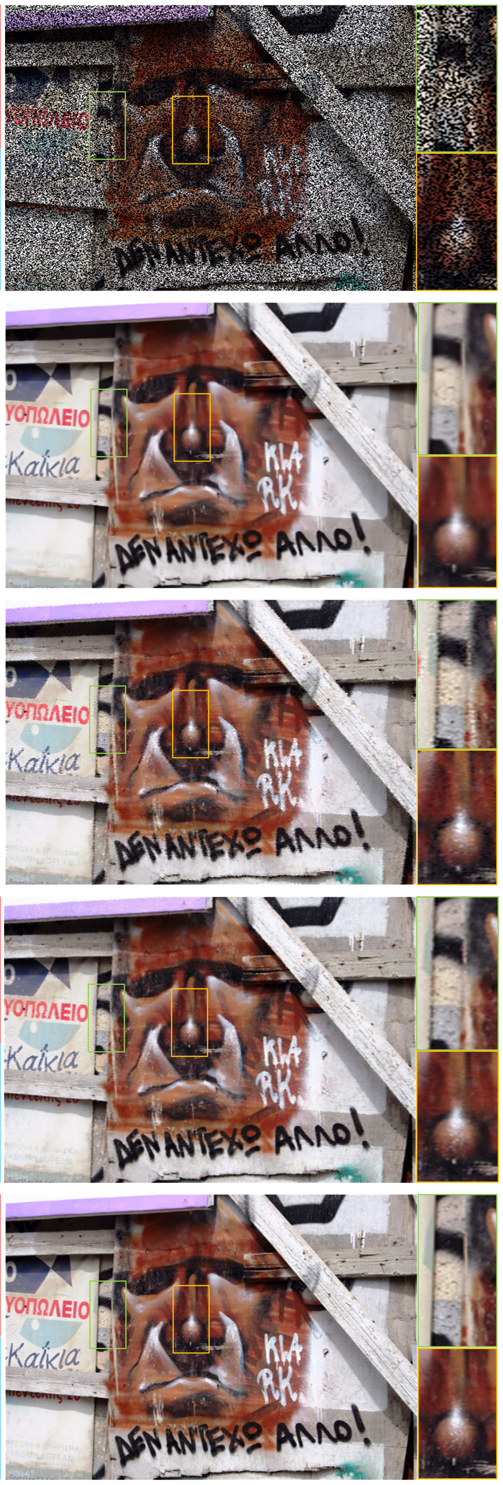

(b)

Figure 4.17: Pixel interpolation. First row: occluded image; Second row: EPLL based inpainting [128]; Third row: ConvNet based inpainting [82]; Fourth row: restored by proposed algorithm; Fifth row: the original image. Zooming in to see details.

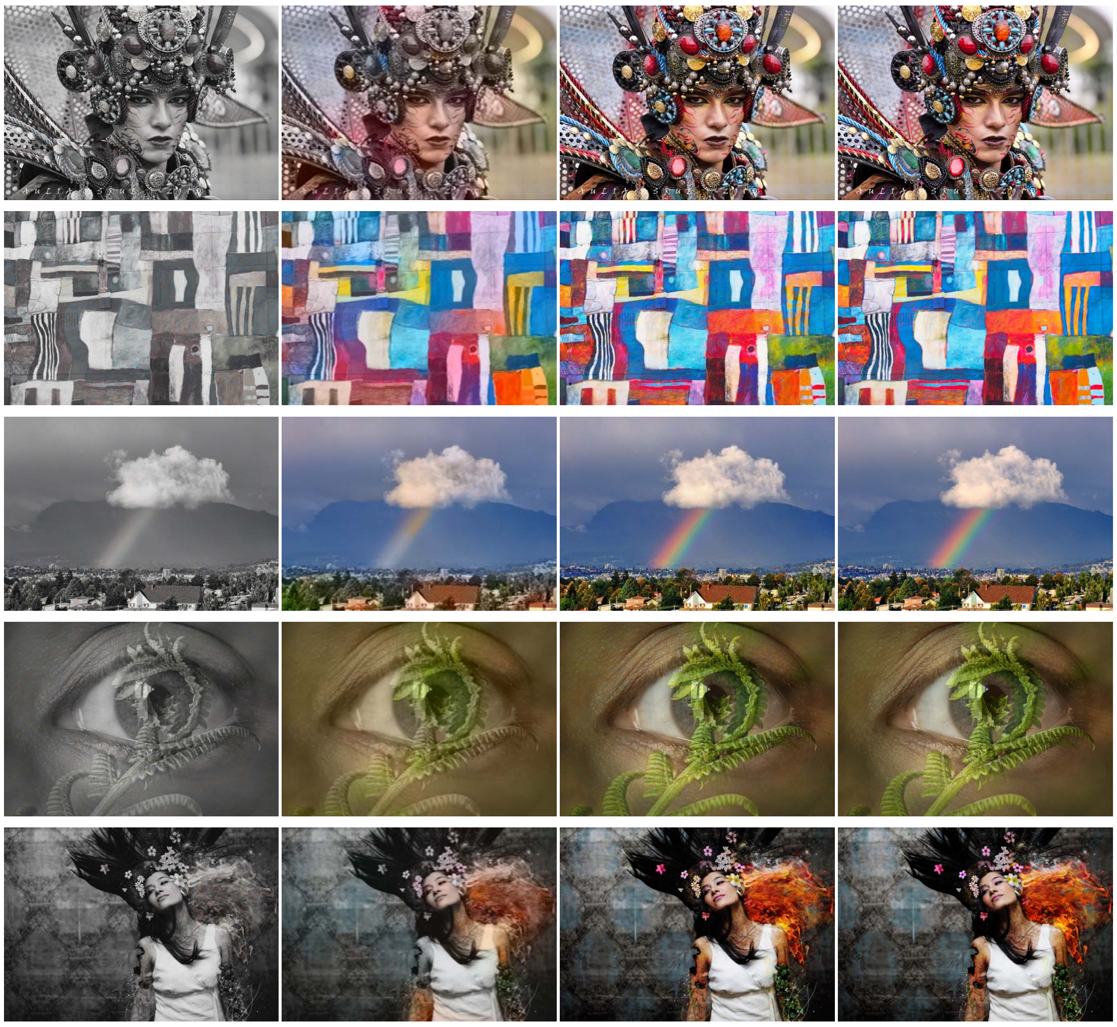
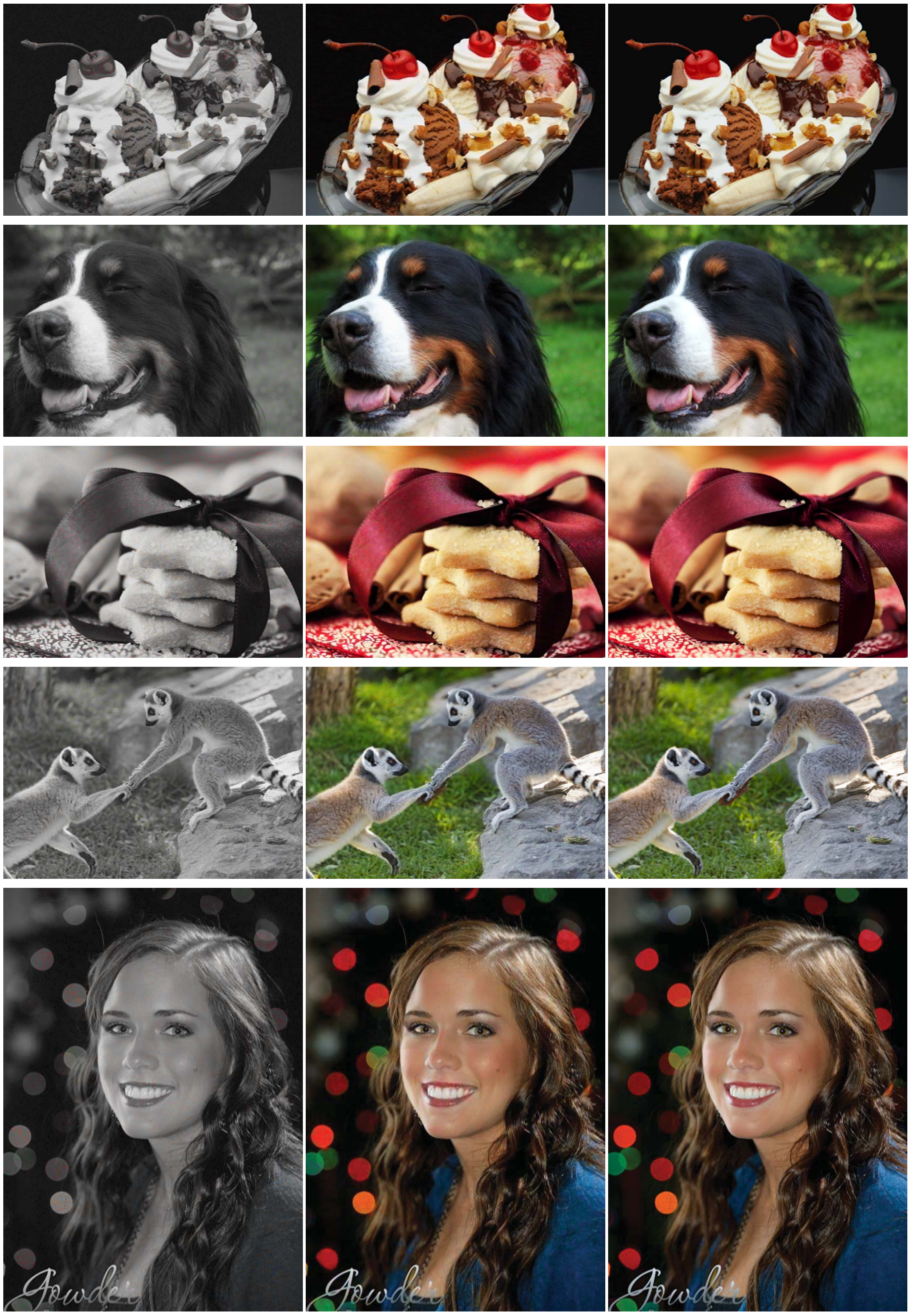

(a) degraded (b) Levin *et al.* (c) proposed (d) original

Figure 4.18: Color interpolation with comparison to Levin *et al.* [53].



(a) degraded (b) proposed (c) original

Figure 4.19: Color interpolation via proposed algorithm.

Chapter 5

Face Parsing via Recurrent Propagation

5.1 Introduction

Recent years have witnessed significant progress in object segmentation and image parsing using deep ConvNets [28, 21, 85, 16, 125, 65]. With end-to-end nonlinear classifiers and hierarchical features, ConvNet-based face parsing methods [64, 102] achieve the state-of-the-art performance than approaches based on hand-crafted features [48]. The main issues with existing ConvNet-based face parsing are the heavy computational load and large memory requirement. Both issues can be alleviated by using shallow or light-weighted convolutional structures, but at the expense of parsing accuracy.

In this work, we propose a face parsing algorithm in which a spatially variant recurrent module is incorporated for global propagation of label information. A straightforward combination of ConvNet and RNN is to take each activation in a ConvNet feature map as the input to a hidden recurrent node in a two-dimensional (2D) spatial sequence and use the recurrent structure to learn the propagation weight matrix in an end-to-end fashion [9, 55]. These models either utilize a spatial RNN [9], or a stacked long short-term memory (LSTM) [55]. In contrast, the proposed recurrent structure exploits the strength of both models in which we apply a simple structure similar to a typical RNN but maintains the capability of spatially variant propagation of an LSTM. Specifically, the proposed model uses a spatially variant gate map to control the propagation strength over different locations in the label space. For face parsing, this gate is naturally associated with the semantic boundary. A gate allows propagation between pixels in a label-consistent region or stops it otherwise. We show that this gate can be obtained via a relatively shallow ConvNet that focuses on learning low and mid-level image features. The RNN module, controlled by the gate, can utilize rich redundant information by propagating the predicted labels to their neighboring pixels in the label-consistent region. Compared to a deep ConvNet face parser with similar performance, the propagation layer requires a small amount of model parameters and significantly reduces the computational cost. As a result, we construct a model that is hundreds of times faster and smaller than deep ConvNet-based methods [64, 102] for face parsing without loss of accuracy.

We validate the proposed algorithm on both coarse-grained (parsing an image with major regions including skin, hair and background) and fine-grained (parsing an image with detailed facial components such as eyes, eyebrows, nose and mouth) face parsing. Both are of critical importance for real-world applications in face processing, e.g., coarse-grained face parsing for style transfer [12] and fine-grained face parsing for virtual makeup. Parsing only the main classes is generally easier under the same settings due to the complexity of solutions and more balanced distributions of training samples. We show that the proposed model can parse all faces of an image in one shot, and significantly outperform the state-of-the-art methods in terms of accuracy and speed.

One issue with applying a single network to fine-grained face parsing is the performance on small facial components. This is due to the extremely unbalanced sample distributions and image size of these regions. We design a two-stage method to parse these components efficiently. We train the model for the main classes in the first stage and then focus on the others with relatively simpler sub-networks. Specifically, the sub-networks in the second stage take a cropped facial region as input. In contrast to a face component may occupy a small amount of pixels from a whole image, the distributions of the pixels for a cropped region are more balanced. We show that by dividing the second face parsing problem into several sub-tasks, the overall network complexity is significantly reduced.

The contributions of this work are summarized as follows. First, a light-weighted network is proposed for pixel-wise face parsing by combining a shallow ConvNet and a spatially variant RNN, which significantly reduces the computational load of deep ConvNet. Second, we show that when parsing a face image with multiple detailed components, dividing the problem into several sub-tasks is significantly more efficient than using one single model, with even better accuracy. Experimental results on the HELEN [94], LFW-PL [48] and Multi-Face demonstrate the efficiency and effectiveness of the proposed face parsing algorithm against the state-of-the-art methods.

5.2 Proposed Algorithm

Most ConvNet-based face parsing algorithms [64, 116] apply deep networks with a large number of parameters, which entail heavy computational loads. On the other hand, shallow models can be executed efficiently but not able to model global data dependency. In this work, we use a shallow ConvNet with a combination of spatially variant recurrent propagation module to model image data effectively and efficiently.

Our model contains a shallow ConvNet and a spatial RNN, as shown in Figure 5.1. First, the ConvNet takes a color image as its input and learns a coarse pixel-wise label score map (Figure 5.1(b)). Second, the coarse label result is fed to a spatial recurrent unit for global propagation. Specifically, the spatial propagation is controlled by a gate map (Figure 5.1(c)), which is referred to as a recurrent gate in the rest of the chapter. Each pixel in the map, formulated as a scalar weight coefficient to the recurrent term, controls the connection between two adjacent hidden nodes at the corresponding location. Since a gate map can be supervised by the ground truth semantic boundaries from labeled annotations, it enables the recurrent propagation to be discriminative between semantically consistent and inconsistent regions, with respect to the specific input image.

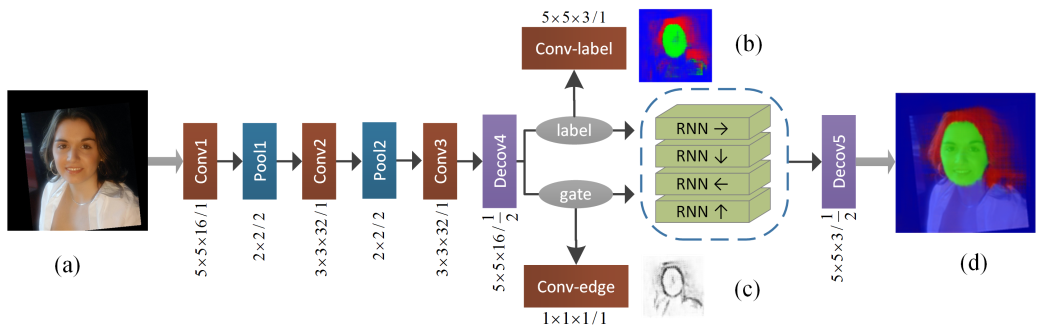


Figure 5.1: Proposed parsing network architecture by combining a ConvNet and a spatial RNN. The ConvNet generates a coarse label map (b) and a recurrent gate (c), which are fed into 4 RNNs with different directions to generate a more accurate result (d). The network structure is shown where the notation for Conv1 “5×5×16/1” means convolution layer with 5×5 kernel, 16 channels and stride 1. The face image in (d) is further segmented with detailed labels in the second stage (see text and Figure 5.2).

We first briefly review conventional RNNs and describe how we extend it to the 2D space for image data, before introducing the recurrent gates. We then discuss how to train the hybrid model in an end-to-end fashion.

5.2.1 Recurrent Neural Networks

The conventional RNN is developed to process 1D sequential data where each hidden node represents a single character, frame, pixel and is connected to its adjacent neighbor. The hidden node i, denoted as $h_i \in H$ receives two inputs: an external input $x_i \in X$ and its previous activation h_{i-1} from one step back. The summation of these two inputs is then non-linearly mapped via a function $\theta(\cdot)$ as the activation of the current step:

$$h_i = \theta(a_i), \quad a_i = \omega_x x_i + (\omega_h h_{i-1} + b). \tag{5.1}$$

In this formulation, x_i and h_i can have different dimensions, where the input transition matrix ω_x aligns x_i to have the same dimension as h_i. In addition, b is a bias or offset term to model data points centered at a point other than the origin. For simplicity, we set x_i and h_i to have the same dimension, and remove the ω_x so that only the recurrent state transition matrix needs to be learned.

To extend the 1D RNN in (5.1) to 2D images, we consider each row/column as 1D sequence, and then adopt an approach similar to the bidirectional recurrent neural network for processing temporal sequences [36]. First, the 1D sequential RNN is processed respectively along left-to-right, top-to-bottom, and their reverse ways. Taking the left-to-right direction for a 2D feature/label map as an example, the 1D sequential RNN scans each row from left to right. As a result, four hidden activation maps are generated.

The four hidden activation maps can be grouped either in parallel or cascade, as introduced in [62]. The four maps share the same input X with the parallel method, while in the cascade way, each RNN takes the output from a previous RNN as its input. We adopt the parallel method and integrate the maps by selecting the optimal direction based on the maximum response at each location. This is carried out by a node-wise max pooling operation that can effectively select the maximally responded direction as the desired information to be propagated and reject noisy information from other directions. We note that the four-directional RNNs with parallel integration can be executed simultaneously with multiple GPUs for further acceleration as they are independent.

The backward pass is also an extension of the back propagation through time (BPTT) method used in RNNs [109]. Due to space limitation, we only present the derivative with respect to a_i:

$$\delta_i = \theta'(a_i) \cdot (\xi_i + \omega_h \delta_{i+1}), \tag{5.2}$$

where ω_h is a square weight matrix and all the others are 1D vectors. We denote ξ as the influence from the output layer on top of the proposed spatial RNN, and the second term in (5.2) the influence from the next hidden node. The derivatives are passed back in reverse order against the feedforward process, with four distinct directions computed respectively [109].

5.2.2 Spatially Variant Recurrent Network

The fundamental problem of the RNN in (5.1) is that the hidden state transition matrix ω_h is spatially invariant. As such, it tends to propagate any pixel to its adjacent ones with a group of fixed weights. However, the label space is spatially variant with respect to different locations. The propagation between pixels that share the same label should be distinguished from those between pixels with different labels on the semantic boundaries.

To this end, we propose a spatially variant recurrent network with gate maps $g_i \in G$ as an additional input to the spatial RNN. Each g_i is an additional coefficient that controls the strength of connections between nodes to guide the recurrent propagations. Intuitively, strong connections (e.g., g_i is close to 1) should be enforced between nodes in the label-consistent region. On the other hand, weak connections (e.g., g_i is close to 0) should be assigned to the nodes belonging to semantically different categories, so that they can be successfully separated.

To reformulate the framework, we have two types of inputs to a RNN, i.e., an external input X, and a spatially variant gate G. Given a hidden node h_i, the spatially controllable recurrent propagation is:

$$a_i = x_i + g_i \cdot (\omega_h h_{i-1} + b). \tag{5.3}$$

The propagation of the hidden activation at $i-1$ to i is controlled by dot product with g_i. We use the identity function $\theta(x) = x$ as the activation (also used by [14, 62]), since experimentally it achieves better performance. To maintain the stability of the linearized formulation, the absolute value of g_i, and norm of ω_h are both normalized to be within one during parameter update in order to prevent the hidden activation in H to increase exponentially.

Similar to the sequential RNN, the BPTT algorithm is adopted to adjust X and G in the spatially variant RNN. The derivatives with respect to a_i and g_i, denoted as δ_i and ε_i are:

$$\delta_i = \xi_i + g_i \cdot \omega\delta_{i+1}, \qquad \varepsilon_i = \delta_i \cdot (\omega_h h_{i+1} + b). \tag{5.4}$$

In addition, the derivative from RNN with respect to x_i is equal to δ_i.

5.2.3 Hybrid Model of ConvNet and RNN

In the proposed framework, we apply a ConvNet that provides label representation X and spatially variant gate representation G to the spatial RNN (see Figure 5.1). With the effective propagation of RNN, the ConvNet can be relatively shallow as revealed in the experimental analyses. Taking the three-class face parsing as an example, the main part of ConvNet is equipped with only three convolutional layers, two max pooling (downsampling) as well as deconvolutional (upsampling) layers, as shown in Figure 5.1, and at most 32 channels for each layer. The proposed network is significantly smaller than most existing ConvNet-based face parsing models based on 6 convolutional layers with 2 fully-connected layers [64], or 16 layers [102] from VGG [92].

To connect with the spatial RNN, the feature maps with 16 channels generated from the first deconvolutional layer (Deconv6 in Figure 5.1) are equally split into two components (each with 8 channels), where one is for pixel-wise labels and the other is for the recurrent gate, with equal width and height. They are then fed to four recurrent layers with different directions as X and G respectively, where each pixel i in the hidden layers is processed by combining x_i and g_i based on (5.3).

The hybrid network contains three different loss layers. At the top of the ConvNet, both X and G are supervised with the softmax cross entropy loss. The labeling representations are transferred by a convolutional layer to be directly supervised by the ground truth labels (see Figure 5.1(b)). The gate representations are transferred by a 1×1 convolutional layer to have a single channel output, which is supervised by the boundary between different categories (see Figure 5.1(c)). Finally, the output of RNN with 8 channels are transferred to 3 channels, upsampled to the original image scale, and supervised by the ground truth labels (see Figure 5.1(d)). All the losses encourage the ConvNet to learn better label candidates as well as guidances to the propagation. Specifically, the ground truth boundaries are obtained from the annotated labels, by setting its boundary pixels to zeros and all the others to one. For example, with a pixel i that is located on a boundary of two categories, the ground truth value is set to zero, which can encourage the g_i to "cut off" the connection between different classes, and vise-versa.

5.3 Sub-networks for the Detailed Components

As discussed in Section 5.1 and revealed in the experiments, a single network does not perform well on small facial components. One problem is that some facial components amount to small percentage of the entire dataset, e.g., the eye regions in Figure 5.1(a) occupy less than 1% of the whole image. It is difficult to parse such components in one stage due to unbalanced labeled data. The work of [64] applies a simple strategy by sampling with an equal number of input patches. However, the performance on small facial components is not satisfactory compared to categories with more pixels, e.g., skin. The other problem is the limited resolution of facial components. With a larger input image, more details of the components can be learned. However, it requires deeper or larger models to adapt to the enlarged receptive fields. For a single model, it is a trade-off between effectiveness and efficiency.

We decompose a unified face segmentation network into a two-stage framework. In practice, parsing major classes with either frontal, canonical face or multiple random faces can be handled using the first stage only. For parsing 11 classes in the HELEN dataset, each component can be labeled independently first and then combined with the major ones.

First Stage Skin-Hair-Background Network. The first stage network classifies an image into skin, hair and background regions using the combination of ConvNet and RNN, as introduced in Section 5.2. Since there are only three labels with relatively equal distribution, we do not need to balance the samples. As these classes do not contain detailed structures such as facial components, the input resolution does not need to be high. Similar to [64],

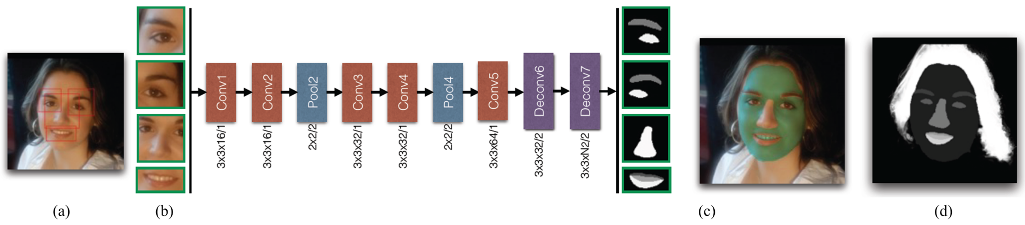


Figure 5.2: The second stage network operates on the cropped region, i.e., left and right eyes, nose, and mouth, to parse accurate facial components. The final parsing result in (d) is the combination of segments from two stages.

a face image is detected and roughly aligned to the center using [98], with a resolution of 128×128 pixels. The result of the label has the same resolution as the input image.

Second Stage Facial Component Networks. We locate the facial components for high resolution faces image through 5 detected key points (eye centers, nose tip, and mouth corners) [98], and crop the patches accordingly. We train three simple and efficient networks to segment eye and eyebrow, nose, and mouth regions, respectively. Figure 5.2(b) shows the structure of eye/eyebrow network. It contains five convolution layers, two max-pooling layers, and two deconvolution layers, with an input size of 64×64. Similar network structures are used for the nose as well as the mouth, and the input image size is 64×64 and 32×64, respectively. Since each image is cropped around each facial component, it does not include many pixels from the skin region. Therefore, the sample distribution is balanced for network training. The final parsing result is composed of the accurate facial component segments in the second stage and the coarse segments in the first stage. Since the segmentation task in the second stage is easier, we do not apply the spatial RNN for efficiency reason.

Table 5.1: Quantitative results on the LFW-PL dataset."F" denotes f-score, "bg" denotes background, "-" denotes not available. We denote the results by Chapter 3 as Liu *et al.* [64]

(%)	GLOC [48]	Liu *et al.* [64]	CNN-S	CNN-deep	CNN-CRF [16]	RNN [129]	RNN-G
F-skin	-	93.93	90.47	91.63	91.25	93.72	**97.55**
F-hair	-	80.70	76.09	78.30	75.21	81.21	**83.43**
F-bg	-	97.10	95.42	95.95	99.58	97.15	**94.37**
Accuracy	94.95	95.09	92.44	93.27	92.59	94.85	**95.46**
Time (ms)	254 (CPU)	~ 110	< 1	~ 2	~ 7	~ 2	~ 2

5.4 Experimental Results

We carry out experiments on images containing one or multiple faces. For single face parsing, we evaluate our method on the LFW-PL [48] and HELEN [94] datasets. In addition, we develop a Multi-Face dataset to evaluate parsing numerous faces in one image. All experiments are conducted on a Nvidia GeForce GTX TITAN X GPU.

5.4.1 Datasets and Settings

LFW-PL and HELEN Datasets. The LFW part label (LFW-PL) dataset contains $2,927$ face images. Each face image is annotated as skin, hair or background using superpixels, and roughly aligned to the center [48]. The HELEN dataset contains $2,330$ face images with manually labeled facial components including eyes, eyebrows, nose, lips, etc. For both datasets, the most centered face in each image is annotated. We adopt the same setting of data splits as [64] and resize each image and its corresponding label to 128×128. For the HELEN dataset, the hair region is trained as one category in the first stage of our algorithm but is not evaluated for fair comparisons with the existing method [94, 64].

Multi-Face Dataset. We collect a Multi-Face dataset where each image contains multiple faces. It contains $9,645$ images in unconstrained environments with pixel-wise labels

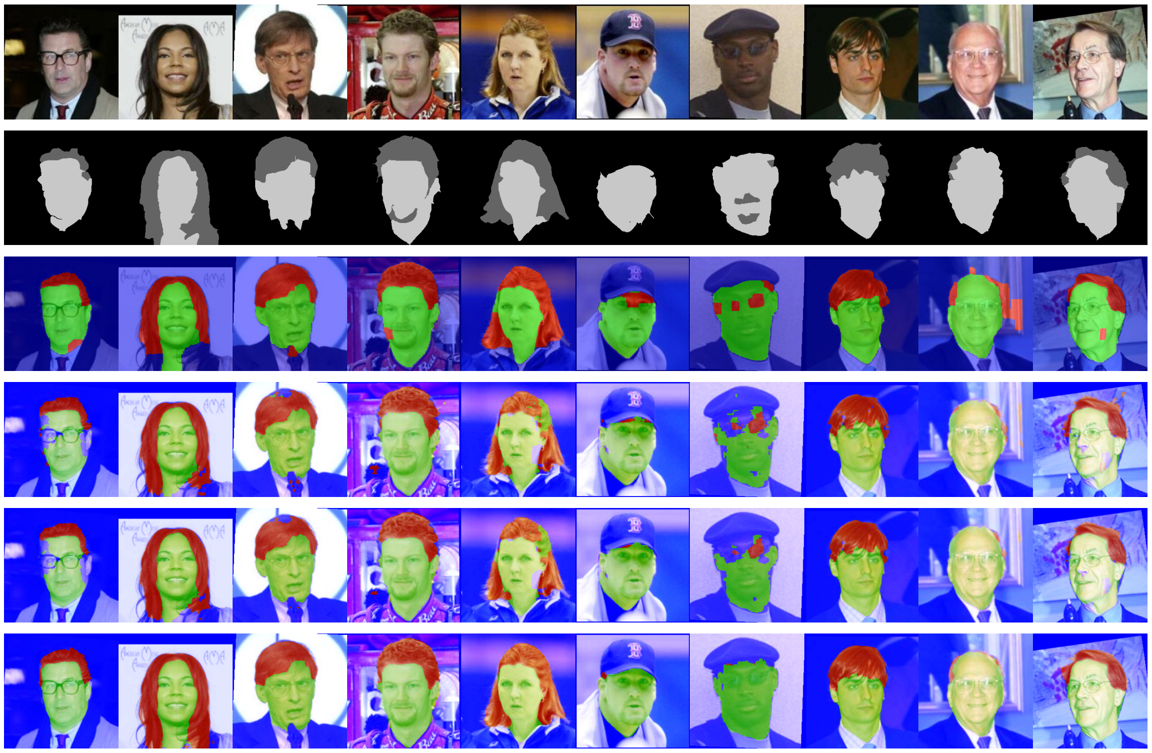


Figure 5.3: Face parsing results on the LFW-PL dataset. First row: input image. Second row: ground-truth annotations. Third row: results from [64]. Fourth row: results from CNN-S. Fifth row: results from CNN with dense CRF. Sixth row: results by RNN-G.

including skin, hair, and background. This dataset is divided into a training set of $9,045$ images, a test set of 200 images, and a validation set of 200 images. We rescale each image and its corresponding label according to the length of the long side to maintain the aspect ratio. Each one is zero padded to result in a 512×512 image where all faces appear clearly.

Network Implementation. Our network structures are described in Figure 5.1 and 5.2. We use the first stage model (see Figure 5.1) to parse images in the LFW-PL and Multi-Face datasets, and the facial skin and hair regions in the Helen dataset. In addition, we use the second stage model (see Figure 5.2) to parse facial components of images in the

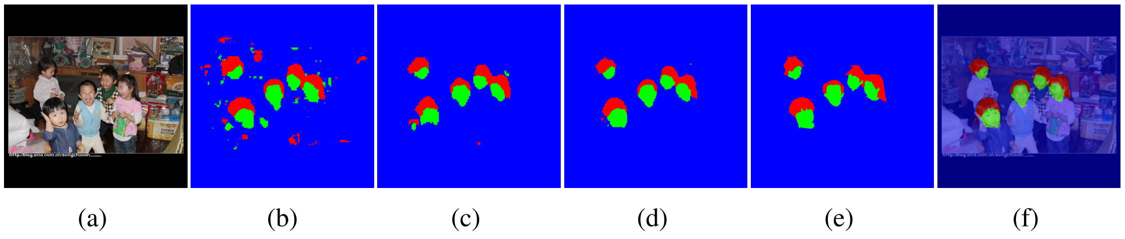


Figure 5.4: Parsing results on the Multi-Face dataset. (a) input image. (b) results by the baseline ConvNet. (c) results by the standard RNN. (d) results from RNN-G. (e) the ground truth. (f) a visualized version of RNN-G. Our method is able to effectively and efficiently parse multiple faces in the cluttered background.

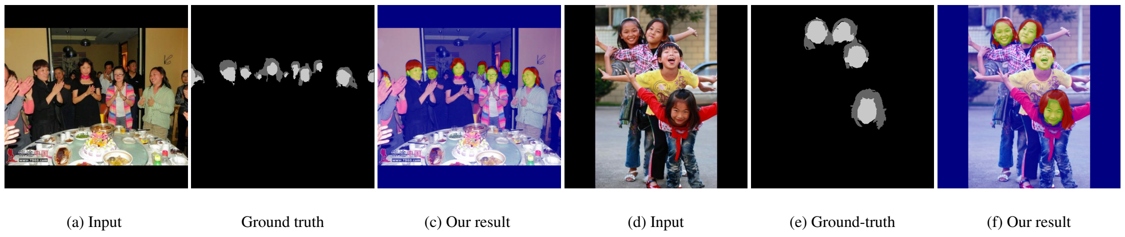


Figure 5.5: Parsing results on the Multi-Face dataset. We can successfully process multiple face with our network.

HELEN dataset.

For fair comparison with the previous work, we align the input images according to the standards in Chapter 3 in the HELEN dataset. The faces in the LFW-PL dataset do not need additional processing since the released images are already coarsely aligned. On the other hand, we directly use the 512×512 images as the network inputs, and do not preprocess any face for the Multi-Face dataset. We quantitatively evaluate and compare our model using per-pixel accuracy and F-measure for each class in all experiments.

In the first stage, the boundaries in Figure 5.1(c) are balanced with the ratio of positive/negative number of pixels set to $1 : 5$ such that a sufficient number of boundary samples can be drawn. The training images are augmented by random affine and mir-

Table 5.2: Quantitative results on the Multi-Face dataset.

(%)	CNN-deep	CNN-CRF	RNN	Det+RNN-G single	RNN-G
F-skin	75.56	77.84	73.33	81.02	**87.36**
F-hair	64.62	61.53	62.85	55.35	**73.09**
F-background	96.5	97.08	96.18	97.10	**98.19**
AC	93.39	94.5	92.78	94.42	**96.35**

ror transformations for increasing the variation of training samples. The network for the Multi-Face dataset has two more $3 \times 3 \times 16$ convolutional units (with max-pooling) and one more deconvolutional layer to adapt to the input size. The results are evaluated with the resolution of 256×256. The boundary loss sampling and training image augmentation strategies are uniformly applied to all experiments. For the second stage model, we crop the facial components based on the 5 facial key points from [98] for training and tests. We include at least additional 20% height/width of the total foreground height/width in the cropped images during training and maintain the aspect ratio.

5.4.2 Coarse-grained Face Parsing

Face parsing with 3 classes are carried out using the first stage model on the LFW-PL and the Multi-Face datasets, respectively. We compare the proposed method, denoted as RNN-G with: (a) shallow ConvNet part only (CNN-S). (b) shallow ConvNet with the RNN module replaced by two 3×3 convolutional layers with 32 channels as a baseline network, denoted as CNN-Deep. We increase the number of the output channels of Deconv6 (Figure 5.1) from 8 to 16 to ensure that the shallow model can converge. (c) a combination of the shallow ConvNet and the post processing with a dense CRF, denoted as CNN-CRF, which is commonly used in recent semantic segmentation tasks [16]. (d) a standard RNN in (5.1) (similar to [129]) with the same ConvNet. We note that both [16, 129] do not have experiments on shallow networks.

We show two more baseline methods [48, 64] (also Chapter 3) evaluated on the LFW-

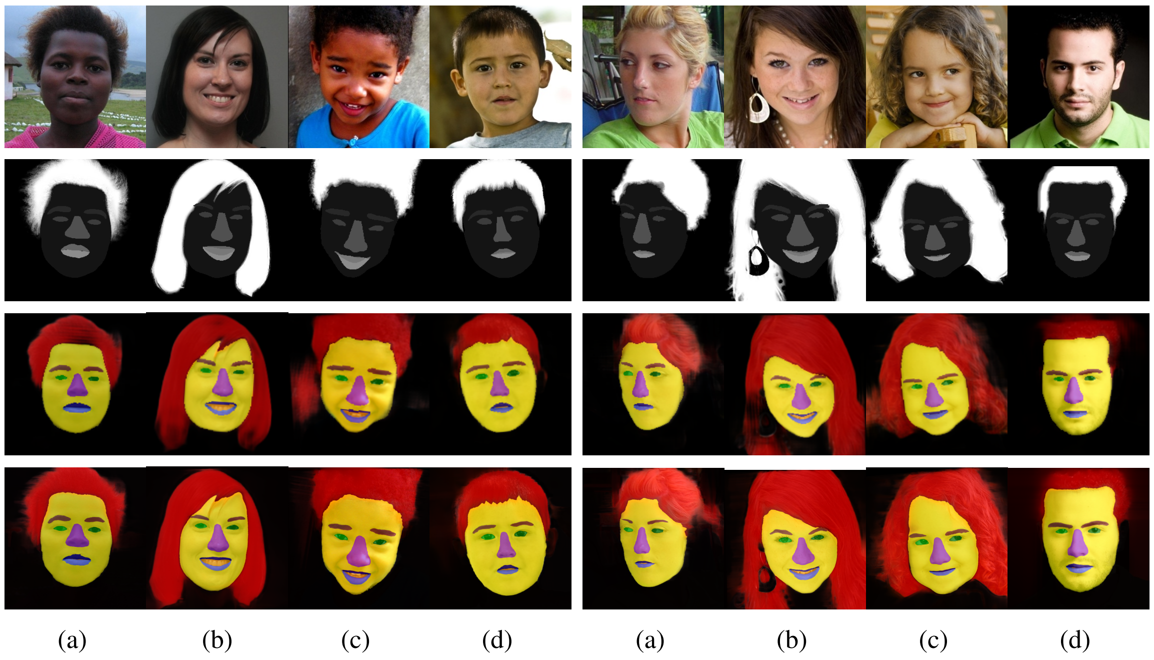


Figure 5.6: Face parsing results on the HELEN [94] dataset. (a) input image. (b) ground-truth annotations. (c) results from [64]. (We roughly crop the results for better visual comparisons.) (d) our results with 11-class pixel-wise parsing.

PL dataset. Specifically, we adjust the results in Chapter 3 by using only one-time feedforward with 2× bilinear upsampling layer for fair comparisons in speed and accuracy. For the Multi-Face dataset, we use the models to parse all faces in images without using any detector. This is computationally efficient and useful for numerous applications without the need of instance-level information. We note the label distribution of the Multi-Face dataset with respect to different categories are significantly unbalanced since the vast majority of pixels belong to the background regions. Thus, we apply a data sampling strategy at each loss layer by maintaining the number of sampled background pixels as 5 times of the total number of pixels for skin and hair regions.

Table 5.1 and 5.2 show the results with similar trends on these two datasets. Overall, the shallow ConvNet, i.e., CNN-S, has limited performance. There is no significant im-

Table 5.3: Quantitative evaluation results on the HELEN dataset. We denote the upper and lower lips as "U-lip" and "L-lip", and overall mouth part as "mouth", respectively. See Chapter 3 (denoted as Liu *et al.* [64] in the table) and [94] for more details.

Methods	eyes	brows	nose	in mouth	U-lip	L-lip	mouth	skin	overall
Liu *et al.* [59]	77.0	64.0	84.3	60.1	65.0	61.8	74.2	88.6	73.8
Smith *et al.* [94]	78.5	72.2	92.2	71.3	65.1	70.0	85.7	88.2	80.4
Liu *et al.* [64]	76.8	71.3	90.9	80.8	62.3	69.4	84.1	91.0	84.7
Ours 1-stage	63.3	53.7	87.5	65.7	54.0	72.6	80.6	91.1	78.8
Ours 2-stage	**86.8**	**77.0**	**93.0**	79.2	**74.3**	**81.7**	**89.1**	**92.1**	**88.6**

provement gain by simply adding more layers (CNN-Deep) or adding an additional dense CRF module (CNN-CRF). The standard RNN without the spatially variant gate performs better, but still worse than the proposed method. With the spatially variant gate, the RNN-G model performs significantly better than the baseline CNN-S, CNN-Deep and RNN models. The results demonstrate the effectiveness of the proposed spatially variant RNN structure. The proposed models operate at 500 fps for a 128×128 single face image and 200 fps for a 512×512 image with multiple faces.

Figure 5.3 and 5.4 show some parsing results on the two datasets. The proposed RNN-G model performs favorably against the CNN-S, CNN-CRF, standard RNN, and the method using nonparametric prior and graph cut inference in Chapter 3. For Multi-Face dataset, we evaluate the alternative method using a face detector [79] and the single face parser trained on the LFW-PL dataset, which operates at 37 fps on average (depending on the number of detected faces). The RNN-G model performs favorably in the cluttered background against all alternative methods in terms of accuracy and efficiency.

5.4.3 Fine-grained Face Parsing

In the HELEN dataset, we evaluate the parsing results following the settings in Chapter 3, where the second stage network is utilized to improve parsing results. Since the second stage takes less than 1 ms, the overall run-time for parsing a face with 11 classes can operate at 300 fps on a single GPU.

Table 5.3 and Figure 5.6 show the quantitative and qualitative parsing results. We first show that by using a single stage, the unified model cannot handle detailed facial parts even with the spatially variant RNN module. Our two-stage network performs favorably against the state-of-the-art methods, and the one stage network model, on all categories. It is worth noting that the overall F-measure achieved by the RNN-G model is 0.886, which amounts to 20% reduction in error rate from the state-of-the-art method [59]. These experimental results demonstrate that the two-stage network structure with the spatially variant gate is effective for accurate and efficient face parsing.

5.5 Applications

Based on the proposed fast face parsing algorithm, automatic facial editing applications can be constructed. We take (a) eyebrow type switching, (b) eyelash editing, (c) lip color adjustment, (d) facial skin beautification, and (e) facial makeup transferring as examples to demonstrate the applications of the proposed algorithm.

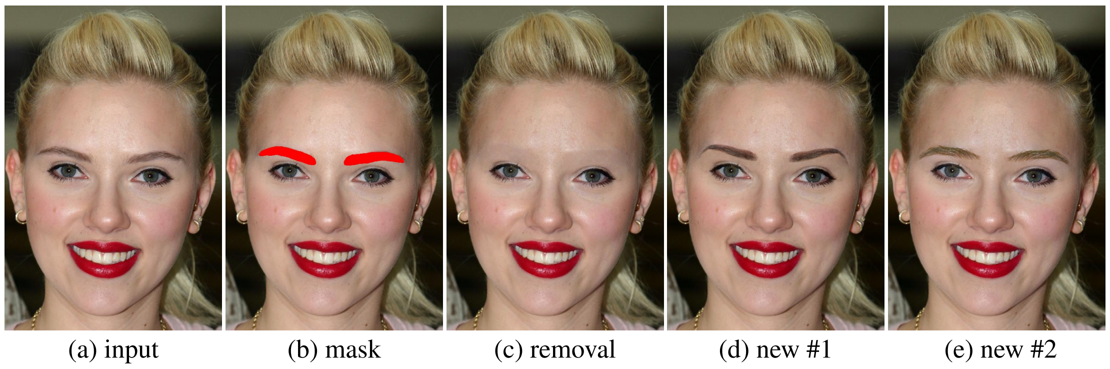


Figure 5.7: Swithcing of eyebrow types given the parsed facial components.

5.5.1 Eyebrow Editing

The application of eyebrow editing can be carried out by removing the existing eyebrow and appending a new type based on the accurate boundaries generated by face parsing. Given the input image in Fig. 5.7 (a), and eyebrow parsing results in (b), we apply Poisson image editing [76] to remove the existing eyebrow, shown in (c). In particular, we slightly dilate the mask to accommodate for any error caused by an uncovered boundary. New types of eyebrows can be easily generated through alpha blending. Fig. 5.7 (d) and (e) show two different eyebrow types enabled by our mask.

5.5.2 Eyelash Editing

We can append eyelashes on the eye regions based on the accurate eye boundaries generated from parsing results. Given a face image in Fig. 5.8 as input, we adopt the thin plate splines (TPS) algorithm [6] to map the eyelash template to the eye boundary. The results are obtained through alpha blending.

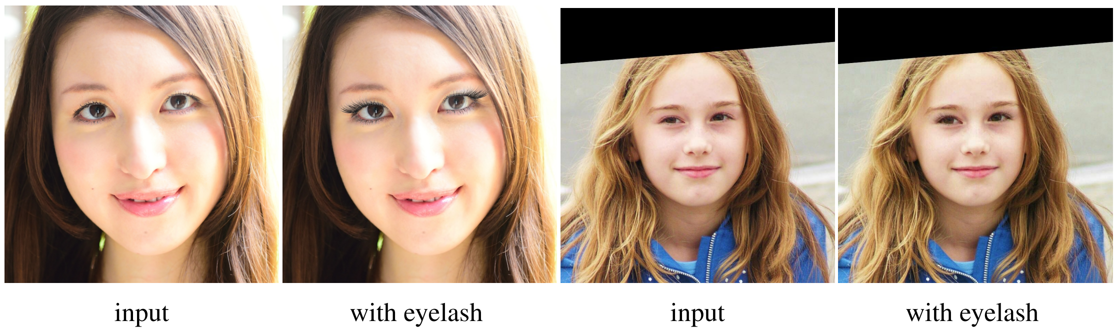


Figure 5.8: Eyelash editing. Best viewed by zooming in.

5.5.3 Lip Color Adjustments

With the accurate mouth region in Fig. 5.9, we can directly change the color tone of the lips. Two examples are shown in Fig. 5.9 (b) and (c).

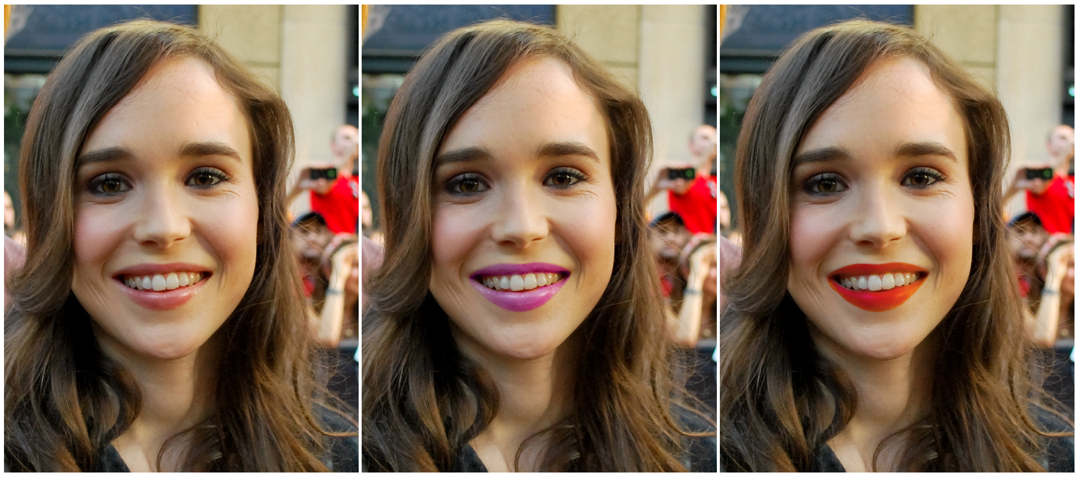


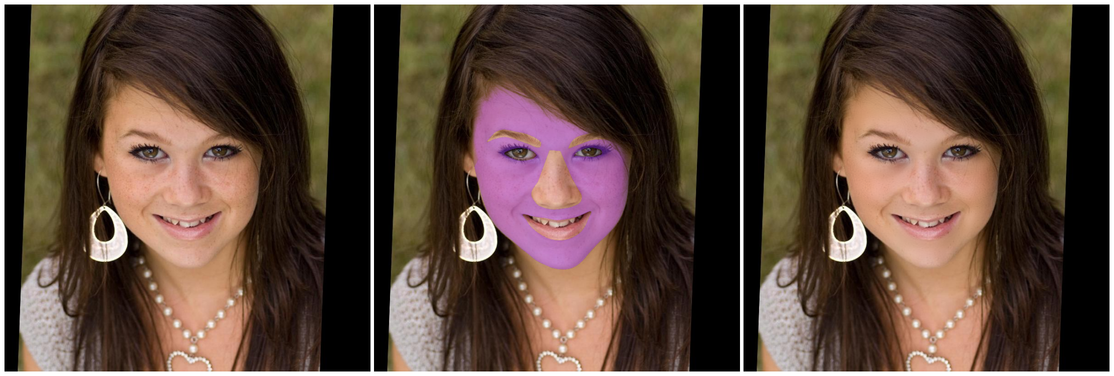
Figure 5.9: Lip Color Adjustments. Best viewed in color.

5.5.4 Skin Smoothing

Equipped with the facial skin mask, we can smooth the facial area without affecting the details in other components. We first apply the rolling guidance filter (RGF) [121] for edge-preserving smoothing. However, the filtered image does not contain fine details and is visually inauthentic. We denote the edges located by the RFG as edge region, and the other part as the smooth region. To preserve the fine details (e.g., skin poles), we preserve the high-frequency image texture on the corresponding smooth region, which can be obtained through the residue of a standard Gaussian filter ($\delta = 2$). The final result is illustrated in Fig. 5.10 (c).

5.5.5 Makeup Transfer

We combine the above-mentioned applications for full makeup editing. Specifically, we can transfer the makeup from a set of reference images with distinct makeup styles (see first row of Fig. 5.11) to a test image (see second row of Fig. 5.11). This is carried



(a) input (b) parsing skin area (c) smoothed

Figure 5.10: Smoothing the skin region. Best viewed by zooming in.

out by first accurately aligning the face on the reference image to the one on the test image through TPS with the semantic boundaries extracted from the parsing result, and then applying the facial fine texture and color from the corresponding components on the reference face. Given the fine segments, the proposed processing is simpler compare to the work by Guo and Sim [40].

For makeup transfer, we first apply the facial texture and color by linearly blending the gradients of facial skin of two images in the brightness channel, where the weights with respect to each image can be adjusted by user according to the final visual effect. Second, we reconstruct the brightness channel with the new facial gradient map through Poisson image editing [76]. Third, we apply the chromatic channels of the reference image with respect to facial skin and lips. The final virtual makeups, as shown in the second row of Fig. 5.11, are realistic and accurate on the facial component boundaries due to the fine precision of the face parsing results.

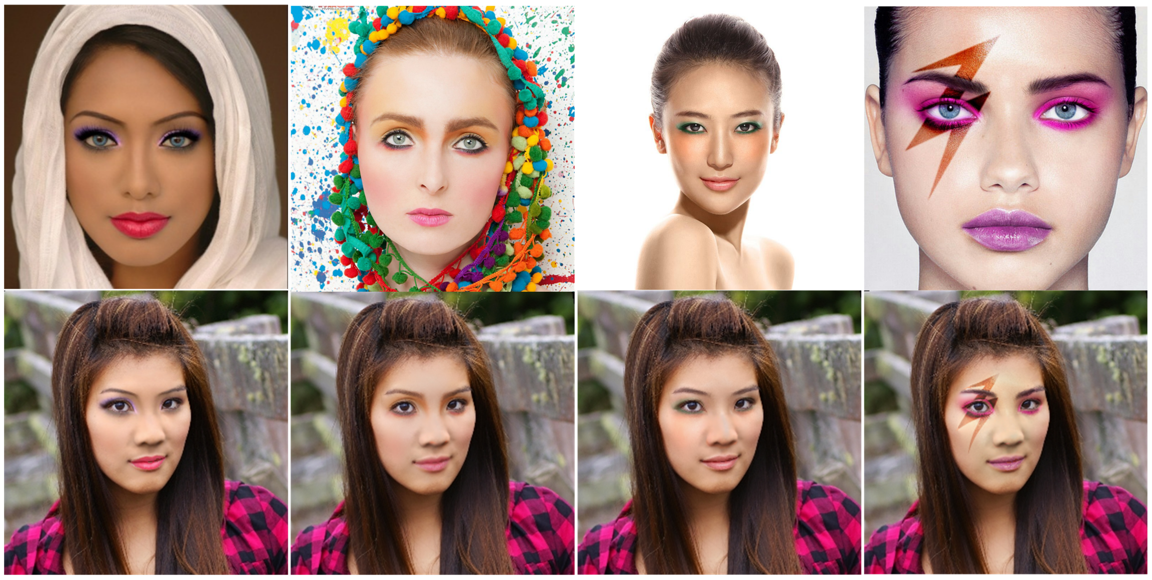


Figure 5.11: Facial makeup transfer. First row: reference model images with specific-stylized makeup. Second row: virtual makeup by applying facial detail and color from the models in the first row. Best viewed in color through zooming in.

5.6 Summary

In this chapter, we propose a pixel-level face parsing network by combining a shallow ConvNet and a spatially variant RNN. The recurrent propagation infers globally over the entire image with the guidance of a local model, which reduces the computational load of deep ConvNets. We develop a two-stage approach for accurate parsing of the detailed facial component. Experimental results on the HELEN [94], LFW-PL [48] and the proposed Multi-Face datasets demonstrate the efficiency and effectiveness of the proposed face parsing algorithm against the state-of-the-art methods.

Chapter 6

Learning Affinity via Spatial Propagation Networks

6.1 Introduction

An affinity matrix is a generic matrix that determines how close or similar two points are in a space. In computer vision tasks, it is a weighted graph that regards each pixel as a node, and connects each pair of pixels by an edge [88, 54, 53, 42]. The weight on that edge should reflect the pairwise similarity with respect to different tasks. For example, for low-level vision tasks such as image filtering, the affinity values should reveal the low-level coherence of color and texture [29, 42]; for mid to high-level vision tasks such as image matting and segmentation [54, 69, 41, 4], the affinity measure should reveal the semantic-level pairwise similarities. Most techniques explicitly or implicitly assume a measurement or a similarity structure over the space of configurations. The success of such algorithms depends heavily on the assumptions made to construct these affinity matrices, which are generally not treated as part of the learning problem.

In this chapter, we show that the problem of learning the affinity matrix can be equivalently expressed as learning a group of small row/column-wise, spatially varying linear transformation matrices. Since a linear transformation can be easily implemented as a differentiable module in a deep neural network, the transformation matrix can be learned in a purely data-driven manner as opposed to being constructed by hand. Specifically, we adopt an external deep ConvNet to output all entities of the matrix with the input of the original RGB images, such that the affinity is learned from a deep model conditioned on the specific inputs. We show that using a three-way connection, instead of the fully connection, is sufficient for learning a dense affinity matrix and requires much fewer output channels of a deep ConvNet. Therefore, instead of using designed features and kernel tricks, our network outputs all entities of the affinity matrix in a data-driven manner.

The advantages of learning affinity matrix in a data-driven manner are multifold. First,with SPN, high-level affinity measures (e.g, object segmentation requires semantic-level similarity) that are not easy to design by hand can be learned in an end-to-end fashion. Since the proposed method learns and outputs all entities of an affinity matrix under direct supervision of ultimate loss functions, it learns an effective metric to measure similarities, without considering how the pairs of pixels are compared. Second, it does not need iterative processing during learning and inference. Thus, it is much more efficient with a deep learning framework than the existing solutions for performing similar tasks, such as dense conditional random fields (CRFs) [51, 15, 125].

The proposed spatial propagation network (SPN) contains a deep ConvNet that learns the affinity entities and a spatial linear propagation module. Images or general 2D matrix are fed into the module, and propagated under the guidance of the learned affinity for specific tasks. All modules are differentiable and jointly trained using stochastic gradient descent (SGD) method. The spatial linear propagation module is computationally efficient for inference due to the linear time complexity of the propagation architecture.

6.2 Proposed Approach

In this chapter, we construct a spatial propagation network (SPN) that can transform a two-dimensional (2D) map (*e.g.*, the result of coarse image segmentation) to a new one with desired properties (*e.g.*, a new segmentation map with significantly refined details). With spatially varying parameters that supports the propagation process, we show theoretically in Section 6.2.1 that this module is equivalent to the standard anisotropic diffusion process [108, 60]. As proved, the transformation of maps is controlled by a Laplacian matrix that is constituted by the parameters of the spatial propagation module. Since the propagation module is differentiable, its parameters can be learned by any type of neural network (*e.g.*, a typical deep ConvNet) that is connected to this module through joint training. We introduce the propagation network in Section 6.2.2, and specifically analyze the properties of different types of connections within the framework for learning the affinity matrix.

6.2.1 Linear Propagation as Spatial Diffusion

We apply a linear transformation by means of the spatial propagation network, where a matrix is scanned row/column-wise in four fixed directions: left-to-right, top-to-bottom, and verse-vise. This strategy is used widely in [36, 103, 62, 14] (see Chapter 4 and 5). We take the left-to-right direction as an example for the following discussion. Other directions are processed independently in the same manner.

We denote X and H as two 2D maps of size $n \times n$, with exactly the same dimensions as the matrix before and after spatial propagation, where x_t and h_t, respectively, represent their t^{th} columns with $n \times 1$ elements each. We linearly propagate information from left-to-right between adjacent columns using an $n \times n$ linear transform matrix w_t as:

$$h_t = (I - d_t)\, x_t + w_t h_{t-1}, \quad t \in [2, n] \tag{6.1}$$

where I is the $n \times n$ identity matrix, the initial condition $h_1 = x_1$, and $d_t(i, i)$ is a diagonal matrix, where the i^{th} element is the sum of all the off-diagonal elements of the i^{th} row of w_t:

$$d_t(i,i) = \sum_{j=1, j\neq i}^{n} w_t(i,j). \tag{6.2}$$

As shown, the matrix H, where $\{h_t \in H, t \in [1,n]\}$, is updated in a column-wise manner recursively. For each column, h_t is a linear, weighted combination of the previous column h_{t-1}, and the corresponding column x_t in X.

When the recursive scanning is finished, the updated 2D matrix H can be expressed with an expanded formulation of Eq. (6.1):

$$\begin{bmatrix} I & 0 & \cdots & \cdots & 0 \\ w_2 & \lambda_2 & 0 & \cdots & \cdots \\ w_3 w_2 & w_3\lambda_2 & \lambda_3 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \lambda_n \end{bmatrix} X_v = G X_v, \tag{6.3}$$

where G is a lower triangular, $N \times N (N = n^2)$ transformation matrix, which relates X and H. H_v and X_v are vectorized versions of X and H, respectively, with the dimension of $N \times 1$. Specifically, they are created by concatenating h_t and x_t along the same, single dimension, *i.e.*, $H_v = \left[h_1^T, ..., h_n^T\right]^T$ and $X_v = \left[x_1^T, ..., x_n^T\right]^T$. All the parameters $\{\lambda_t, w_t, d_t, I\}, t \in [2,n]$ are $n \times n$ sub-matrices, where $\lambda_t = I - d_t$.

In the following section, we validate that Eq. (6.3) can be expressed as a spatial anisotropic diffusion process, with the corresponding propagation affinity matrix constituted by all $w_t, t \in [2,n]$.

Theorem 1. *The summation of elements in each row of G equals to one.*

Since G contains $n \times n$ sub-matrices, each representing the transformation between the corresponding columns of H and X, we denote all the weights used to compute h_t as the

t^{th} block-row G_t. On setting $\lambda_1 = I$, the k^{th} constituent $n \times n$ sub-matrix of G_t is:

$$G_{tk} = \begin{cases} \prod_{\tau=k+1}^{t} w_\tau \lambda_k, & k \in [1, t-1] \\ \lambda_k, & k = t \end{cases} \tag{6.4}$$

To prove that the summation of any row in G equals to one, we instead prove that for $\forall t \in [1, n]$, each row of G_t has the summation of one.

Proof. Denoting $E = [1, 1, ..., 1]^T$ as an $n \times 1$ vector, we need to prove that $G_t\,[1, ..., 1]^T_{N \times 1} = E$. Equivalently $\sum_{k=1}^{t} G_{tk} E = E$, because G is a lower triangular matrix. In the following part, we first prove that when $m \in [1, t-1]$, we have $\sum_{k=1}^{m} G_{tk} E = \prod_{\tau=m+1}^{t} w_t E$ by mathematical induction .

Initial step. When $m = 1$, $\sum_{k=1}^{m} G_{tk} E = G_{t1} E = \prod_{\tau=2}^{t} w_\tau E$, which satisfies the assertion.

Inductive step. Assume there is a $n \in [1, t-1]$, such that $\sum_{k=1}^{n} G_{tk} E = \prod_{\tau=n+1}^{t} w_t E$, we must prove the formula is true for $n + 1 \in [1, t-1]$.

$$\begin{aligned} \sum_{k=1}^{n+1} G_{tk} E = \sum_{k=1}^{n} G_{tk} E + G_{t(n+1)} E &= \prod_{\tau=n+1}^{t} w_\tau E + \prod_{\tau=n+2}^{t} w_\tau \\ &= \prod_{\tau=n+2}^{t} w_\tau \left[\left(w_{n+1} + I - d_{n+1} \right) E \right]. \end{aligned} \tag{6.5}$$

According to the formulation of the diagonal matrix in Eq. (6.2) we have $\sum_{k=1}^{n+1} G_{tk} E = \prod_{\tau=n+2}^{t} w_\tau E$. Therefore, the assertion is satisfied. When $m = t$, we have:

$$\begin{aligned} \sum_{k=1}^{t} G_{tk} E = \sum_{k=1}^{t-1} G_{tk} E + G_{tt} E &= \prod_{\tau=t}^{t} w_\tau E + \lambda_t E \\ &= w_\tau E + (I - d_t) E = E, \end{aligned} \tag{6.6}$$

which yields the equivalence of Theorem 1. □

Theorem 2. *We define the evolution of a 2D matrix as a time sequence* $\{U\}_T$, *where* $U(T = 1) = U_1$ *is the initial state. When the transformation between any two adjacent states* *follows Eq.* (6.3)*, the sequence is a diffusion process expressed with a partial differential equation (PDE):*

$$\partial_T U = -LU \tag{6.7}$$

where $L = D - A$ is the Laplacian matrix, D is the degree matrix composed of d_t in Eq. (6.2)*, and A is the affinity matrix composed by the off-diagonal elements of G.*

Proof. We substitute the X and H as two consecutive matrices U_{T+1} and U_T in (6.3). According to Theorem 1, we ensure that the sum of each row $I - G$ is 0 that can formulate a standard Laplacian matrix. Since G has the diagonal sub-matrix $I - d_t$, we can rewrite (6.3) as:

$$U_{T+1} = (I - D + A)\, U_T = (I - L)\, U_T \tag{6.8}$$

where $G = (I - D + A)$, D is an $N \times N$ diagonal matrix containing all the d_t and A is the off-diagonal part of G. It then yields $U_{T+1} - U_T = -LU_T$, a discrete formulation of (6.7) with the time discretization interval as one. □

Theorem 2 shows the essential property of the row/column-wise linear propagation in Eq. (6.1): it is a standard diffusion process where L defines the spatial propagation and A, the affinity matrix, describes the similarities between any two points. Therefore, learning the image affinity matrix A in Eq. (6.8) is **equivalent** to learning a group of transformation matrices w_t in Eq. (6.1).

In the following section, we show how to build the spatial propagation (6.1) as a differentiable module that can be inserted into a standard feed-forward neural network, so that the affinity matrix A can be learned in a data-driven manner.

6.2.2 Learning Data-Driven Affinity

Since the spatial propagation in Eq.(6.1) is differentiable, the transformation matrix can be easily configured as a row/column-wise fully-connected layer. However, we note that since the affinity matrix indicates the pairwise similarities of a specific input, it should also

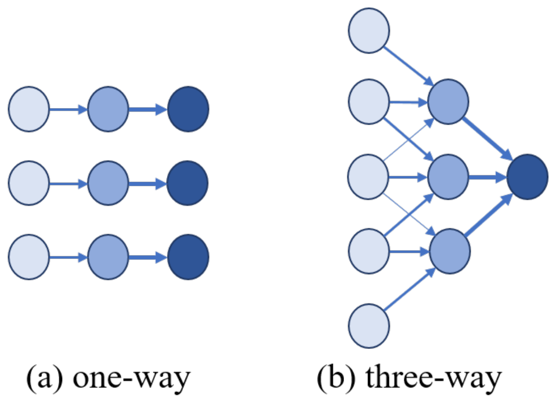


Figure 6.1: Different propagation ranges for (a) one-way connections; and (b) three-way connections. Each pixel (node) receives information from a single line with one-way connection, and from a 2 dimensional plane with three-way connection.

be conditioned on the content of this input (*i.e.*, different input images should have different affinity matrices). Instead of setting the w_t matrices as fixed parameters of the module, we design them as the outputs of a deep ConvNet, which can be directly conditioned on an input image.

One simple way is to set the output of the deep ConvNet to use the same size as the input matrix. When the input has c channels (*e.g.*, an RGB image has $c = 3$), the output needs $n \times c \times 4$ channels (there are n connections from the previous row/column per pixel per channel, and with four different directions). Obviously, this is too many (*e.g.*, an $128 \times 128 \times 16$ feature map needs an output of $128 \times 128 \times 8192$) to be implemented in a real-world system. Instead of using full connections between the adjacent rows/columns, we show that certain sparse connections can also achieve good results for affinity learning. Specifically, we introduce the (a) one-way connection and the (b) three-way connection as two different ways to implement Eq. (6.1).

One-way connection. The one-way connection enables every pixel to connect to only one pixel from the previous row/column (see Figure 6.1(a)). It is equivalent to an one dimensional (1D), linear recurrent propagation that scans each row/column independently as an 1D sequence. Following Eq. (6.1), we denote $x_{k,t}$ and $h_{k,t}$ as the k^{th} pixels in the t^{th} column, where the left-to-right propagation for one-way connection is:

$$h_{k,t} = (1 - p_{k,t}) \cdot x_{k,t} + p_{k,t} \cdot h_{k,t-1}, \tag{6.9}$$

where p is a scaler weight indicating the propagation strength between the pixels at $\{k, t-1\}$ and $\{k, t\}$. Equivalently, w_t in Eq. (6.1) is a diagonal matrix, with the elements constituted by $p_{k,t}, k \in [1, n]$.

The one-way connection is a direct extension of sequential recurrent propagation [36, 105, 49]. The exact formulation of Eq. (6.9) has been used previously for semantic segmentation [14] and for learning low-level vision filters [62] (also in Chapter 4). In [14], Chen *et al.*explain it by domain transform, where in semantic segmentation, p corresponds to the object edges. Chapter 4 explain it by arbitrary-order recursive filters, where p corresponds to more general image properties (*e.g.*, low-level image/color edges, missing pixels, etc.). Both of these can be explained as the same linear propagation framework of Eq. (6.1) with one-way connection.

Three-way connection. We propose a novel three-way connection in this chapter. It enables each pixel to connect to three pixels from the previous row/column, *i.e.*, the left-top, middle and bottom pixels from the previous column for the left-to-right propagation direction (see Figure. 6.2(b)). With the same notations, we denote $\mathbb{N}$ as the set of these three pixels. Then the propagation for the three-way connection is:

$$h_{k,t} = \left(1 - \sum_{k \in \mathbb{N}} p_{k,t}\right) x_{k,t} + \sum_{k \in \mathbb{N}} p_{k,t} h_{k,t-1} \tag{6.10}$$

Equivalently, w_t forms a tridiagonal matrix, with $p_{:,k}, k \in \mathbb{N}$ constitute the three non-zero elements of each row/column.

Relations to the affinity matrix. As introduced in Theorem 2, the affinity matrix A with linear propagation is composed of the off-diagonal elements of G in Eq. (6.3). The one-way connection formulates a spares affinity matrix, where each sub-matrix of A has nonzero elements only along its diagonal. On the other hand, the three-way connection, also with a sparse w_t, can form a relatively dense A with the multiplication of several different tridiagonal matrices. It means pixels can be densely and globally associated, by simply increasing the number of connections of each pixel during spatial propagation from one to three. As shown in Figures 6.2(a) and 6.2(b), the propagation of one-way connections is restricted to a single row, while the three-way connections can expand the region to a triangular 2D plane (blue in Figure. 6.2(b)). The summarization of the four directions result in dense connections of all pixels to each other.

Stability of linear propagation. Model stability is of critical importance for designing linear systems. In the context of spatial propagation (Eq. 6.1), it refers to restricting the responses or errors that flow in the module from going to infinity, and preventing the network from encountering the vanishing of gradients in the backpropagation process [127]. Specifically, the norm of the temporal Jacobian $\partial h_t \setminus \partial h_{t-1}$ should be equal to or less than one. In our case, it is equivalent to regularizing each transformation matrix w_t with its norm satisfying

$$||\partial h_t \setminus \partial h_{t-1}|| = ||w_t|| \leq \lambda_{max}, \tag{6.11}$$

where λ_{max} denotes the largest singularity value of w_t. This condition, $\lambda_{max} \leq 1$ provides a sufficient condition for stability.

Theorem 3. *Let $\left\{p_{t,k}^{K}\right\}_{k\in\mathbb{N}}$ be the weight in w_t, the model can be stabilized if $\sum_{k\in\mathbb{N}} \left|p_{t,k}^{K}\right| \leq 1$.*

Proof. Let λ be the eigenvalue of matrix w_t and λ_{max} be the largest one. According to Gershgorin′s Theorem [34], where every eigenvalue of a square matrix w_t satisfies:

$$\left|\lambda - p_{t,t}\right| \leq \sum_{k=1,k\neq t}^{n} \left|p_{k,t}\right|, \quad t \in [1, n] \tag{6.12}$$

then $\left|\lambda - p_{t,t}\right| + \left|p_{t,t}\right| \leq \sum_{k=1}^{n} \left|p_{k,t}\right|$. According to the triangle inequality, and since $\sum_{k=1,t\neq k}^{n} \left|p_{k,t}\right| \leq 1$, we have

$$\lambda_{max} \leq \left|\lambda - p_{t,t}\right| + \left|p_{t,t}\right| \leq \sum_{k=1}^{n} \left|p_{k,t}\right| \leq 1 \tag{6.13}$$

which satisfies the model stability condition. □

Theorem 3 shows that the stability of a linear propagation model can be maintained by regularizing the all weights of each pixel in the hidden layer H, with the summation of their absolute values less than one. For the one-way connection, Chen *et al.* [14] limited each scalar output p to be within $(0, 1)$. We extended the range to $(-1, 1)$ in Chapter 4, where the negative weights showed preferable effects for learning image enhancers. It indicates that the affinity matrix is not necessarily restricted to be positive/semi-positive definite (*e.g.*, the setting is also applied in [54].) For the three-way connection, we simply regularize the three weights (the output of a deep ConvNet) according to Theorem 3 without restriction to be any positive/semi-positive definite.

6.3 Implementation

We describe the implementation of the three-way connection-based network. We specify two separate branches: (a) a deep ConvNet, namely the guidance network that outputs all elements of the transformation matrix, and (b) a linear propagation module that outputs the propagation result (see Figure 6.2). The structure of a guidance network can be any typical deep ConvNet, which is designed for the task at hand. Examples of this network are described in Section 6.4. The propagation module receives an input map and outputs its refined or transformed version. It also takes the matrix learned by the deep ConvNet guidance network as the second input.

The guidance network takes, as input, any 2D matrix that can help with learning the affinity matrix (*e.g.*, typically an RGB image). It outputs all the weights that constitute the transformation matrix w_t. The linear propagation module takes, as inputs, a 2D map

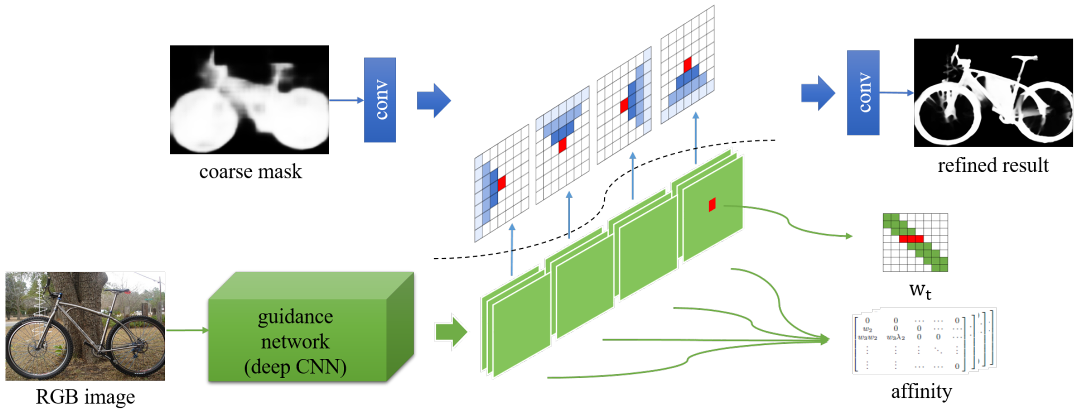


Figure 6.2: We illustrate the general architecture of the SPN using a three-way connection for segmentation refinement. The network, divided by the black dash line, contains a propagation module (upper) and a guidance network (lower). The guidance network outputs all entities that can constitute four affinity matrices, where each sub-matrix w_t is a tridiagonal matrix. The propagation module, being guided by the affinity matrices, deforms the input mask to a desired shape. All modules are differentiable and jointly learned via SGD.

that needs to be propagated (*e.g.*, a coarse segmentation mask), and the weights generated by the guidance network. Suppose that we have a map of size $n \times n \times c$ that is input into the propagation module, the guidance network needs to output a weight map with the dimensions of $n \times n \times c \times (3 \times 4)$, *i.e.*, each pixel in the input map is paired with 3 scalar weights per direction, and 4 directions in total. The propagation module contains 4 independent hidden layers for the different directions, where each layer combines the input map with its respective weight map using Eq. (6.10). All submodules are differentiable and jointly trained using stochastic gradient descent (SGD). We use node-wise max-pooling, similarly to Chapter 4, to integrate the hidden layers and to obtain the final propagation result. See the project page for the information of development details.

6.4 Experimental Results

We validate the SPN on the task of refinement of image segmentation masks. Given a coarse image segmentation mask as the input to the spatial propagation module, we show that the SPN can produce higher-quality masks with significantly refined details at object boundaries (see Figure 6.2). Many models [66, 15] generate low-resolution segmentation masks with coarse boundary shapes, to seek a balance between resolution and semantic accuracy. On the one hand, producing an original high-resolution segmentation mask usually requires the network to neither reduce the size of the input nor that of the output. In such settings, configuring a network with both sufficient capacity and a global receptive field is usually impractical due to the huge model size. On the other hand, keeping a reasonable model capacity while maintaining semantic accuracy, which requires a large receptive field usually results in some sacrifice of the output resolution. The majority of the work [66, 15, 125] chooses the later option and tries to refine the results using either post-processing [15] or jointly trained refinement modules [125]. It is a non-trivial task for producing high-quality segmentation results.

We carry out the refinement of segmentation on two tasks: (a) generating high-resolution segmentation results on the HELEN face parsing dataset [94]; and (b) refining generic object segmentation on top of a pretrained model (e.g., VGG and ResNet based models [66, 15]. For the HELEN dataset, we directly use low-resolution RGB face images to train a baseline parser, which successfully catches the global semantic information. The SPN is then trained on top of the coarse segmentation to generate high-resolution output. For the Pascal VOC dataset, we train the SPN on top of the coarse segmentation results generated by the FCN-8s [66], and directly generalize it to any other pretrained model. We implement the network with a modified CAFFE [47]. The SPN is parallelized during propagating each row-column to the next one with CUDA. We used SGD optimizer, and set the base learning rate to 0.0001. In general, we train 10 epochs, The inference time of SPN on HELEN and Pascal VOC is about 7ms and 60ms for an image of 512×512

resolution, respectively. The source code will be made available to the public.

General network settings. For the HELEN dataset, we train the SPN with smaller patches cropped from the original high-resolution images, their corresponding coarse segmentation maps produced by our baseline parser, and with the corresponding high-resolution ground-truth segmentation masks for supervision. All coarse segmentation maps are obtained by applying the baseline (for HELEN) or pre-trained (for Pascal VOC) image segmentation ConvNets on their standard training splits [27, 15]. Since the baseline HELEN parser produces low-resolution segmentation results, we upsample them using a bi-linear filter to be of the same size as the desired higher output resolution. For the Pascal VOC dataset, we use the original output image segmentation masks produced by the pre-trained ConvNet models as is. These ConvNet models contain upsampling layers, that typically upsample the internal feature representations by 8× (*e.g.*, in [66, 15]) and produce output segmentation masks that are of the same size as that of the input images. We set the SPN as a patch refinement model on top of the coarse map with basic semantic information. We fix the size of our input patches to 128×128, use the *softmax* loss, and use the SGD solver for all the experiments. During training, the patches are sampled from image regions that contain more than one ground-truth segmentation label (*e.g.*, a patch with all pixels labeled as "background" will not be sampled).

We combine the guidance network and the spatial propagation module similarly to [62]. We use two propagation units (*e.g.*, the bottom dashed block in Figure. 6.2 is one propagation unit) with cascaded connections to achieve better results. Differently, we feed in the integrated hidden map of the first unit to the second unit, instead of cascading each direction separately and integrate them at the end of the second unit. We use two more convolutional layers with 32 channels before and after the propagation units to transfer the input map to an intermediate feature map, to make it compatible with the node-wise max-pooling. In addition, we maintain a smaller size of the propagation layer to make the model more efficient w.r.t computational speed and memory. This is carried out by bilinearly downsampling/upsampling after the two convolutional layers, so that the hidden maps of propagation module is with a smaller dimension of 64×64. Note that to compare the one-way with the three-way connection, we use exactly the same settings except the propagation units. We do not apply any configuration used by [14] or [62] (Chapter 4).

HELEN Dataset. The HELEN dataset provides high-resolution photography-style face images (2330 in total), with high-quality manually labeled facial components including eyes, eyebrows, nose, lips, and jawline, which makes the high-resolution segmentation tasks applicable. All previous work utilize low-resolution parsing output as their final results for evaluation. Although many [94, 116, 64] achieve preferable performance, their results cannot be directly adopted by high-quality facial image editing applications. We use the setting that splits 100 samples for test following [116, 64]. We still take the hair region as one category, but do not evaluate it for fair comparisons with the state-of-the work [64] (also in Chapter 3). We use similarity transformation according to the results of 5-keypoint detection [123] to align all face images to the center. Keeping the original resolution, we then crop or pad them to the size of 1024×1024.

We first train a baseline ConvNet with a symmetric downsample/upsample structure. The input image is $8\times$ downsampled from the original version. The downsampling part of the network is equipped with five consecutive conv+relu+max-pooling (with stride of 2) layers. Starting from 32, each one has double the number of channels, resulting in a $4 \times 4 \times 512$ feature maps at the bottleneck. In order to use the information at different levels of image resolution, we add skipped-links by summing features maps of the same dimensions from the corresponding upsample and dowsample layers. The upsample part has symmetric configurations, except that the max-pooling is replaced with bilinear upsampling, and the last sub-module has 11 channels for the 11 classes. We apply the multi-objective loss as introduced in Chapter 3 to improve the accuracy along the boundaries. We note that the symmetric structure is powerful, since the results we obtained for the baseline ConvNet are comparable (see Table. 6.1) to that of Chapter 3, who apply a much larger model (38 MB vs. 12 MB) in comparison. We then train a SPN on top of the baseline ConvNet results, with patches of input RGB image and coarse segmentations

Table 6.1: Quantitative evaluation results on the HELEN dataset. We denote the upper and lower lips as "U-lip" and "L-lip", and overall mouth part as "mouth", respectively. The label definitions follow that in Chapter 3, and denoted as Liu *et al.* [64] in the table.

method	skin	brows	eyes	nose	mouth	U-lip	L-lip	in-mouth	overall
Liu *et al.* [64]	90.87	69.89	74.74	90.23	82.07	59.22	66.30	81.70	83.68
baseline-CNN	90.53	70.09	74.86	89.16	83.83	55.61	64.88	71.72	82.89
Highres-CNN	91.78	71.84	74.46	89.42	81.83	68.15	72.00	71.95	83.21
SPN (one-way)	92.26	75.05	85.44	91.51	88.13	77.61	70.81	79.95	87.09
SPN (three-way)	**93.10**	**78.53**	**87.71**	**92.62**	**91.08**	**80.17**	**71.63**	**83.13**	**89.30**

masks sampled from the preprocessed high-resolution image. For the guidance network, we use the same structure as that of the baseline segmentation network, except that its upsampling part ends at a resolution of 64×64, and its last layer has $32 \times 12 = 384$ channels. In addition, we train another face parsing ConvNet with 1024×1024 sized inputs and outputs (CNN-Highres) for better comparison. It has three more sub-modules at each end of the baseline network, where all are configured with 16 channels to process higher resolution images.

We show quantitative and qualitative results in Table. 6.1 and 6.3 respectively. We compared the one/three way connection SPNs with the baseline, the CNN-Highres and the most relevant state-of-the-art technique for face parsing [64]. Note that the results of baseline and [64][1] are bi-linearly upsampled to 1024×1024 before evaluation. Overall, both SPNs outperform the other techniques with a significant margin of over 6 intersection-over-union (IoU) points, especially for the smaller facial components (*e.g.*, eyes and lips) where with smaller resolution images, the segmentation network performs poorly. We note that the one-way connection-based SPN is quite successful on relatively simple tasks such as the HELEN dataset, but fails for more complex tasks, as revealed by the results of Pascal VOC dataset in the following section.

Pascal VOC Dataset. The PASCAL VOC 2012 segmentation benchmark [27] involves

[1]The original output (also for evaluation) size it $250 * 250$.

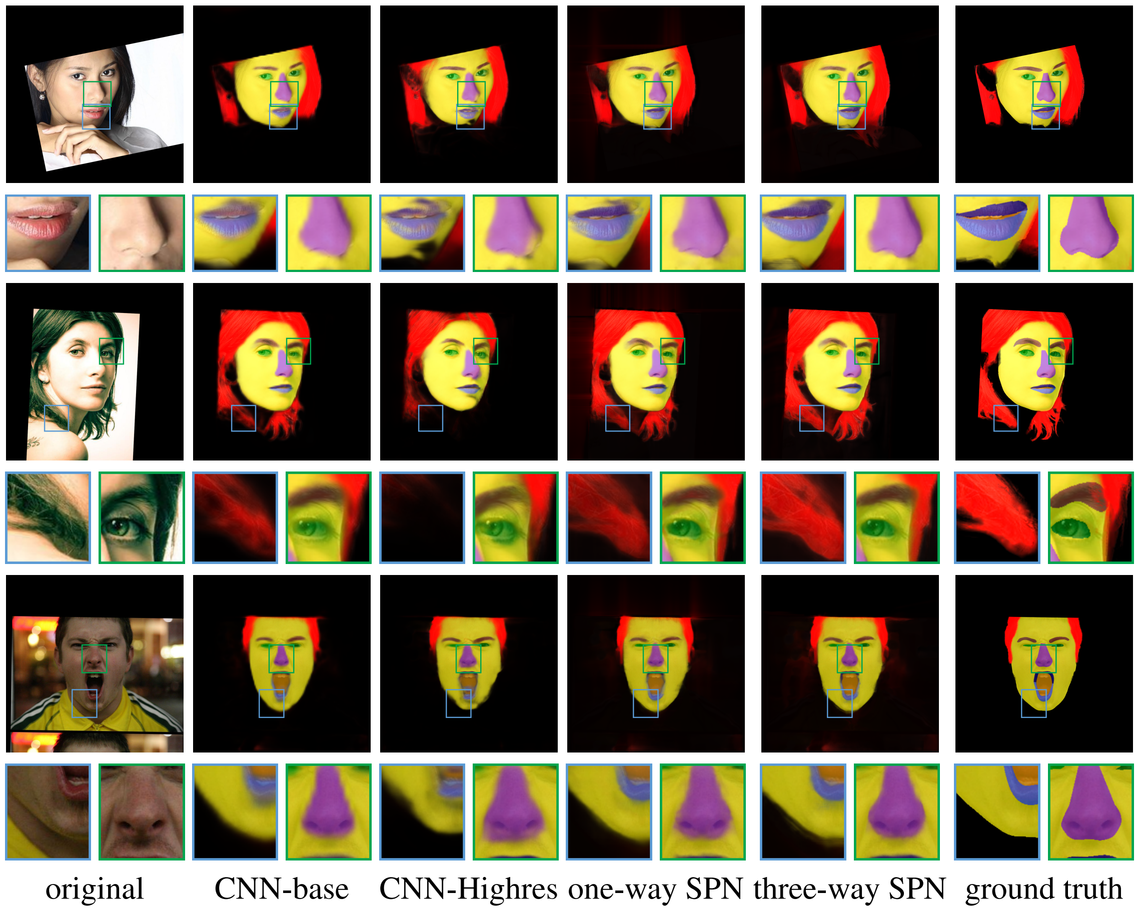


Figure 6.3: Results of face parsing on the HELEN dataset with detailed regions cropped from the high resolution images. (Images are all with high resolution and can be viewed by zoom-in.)

20 foreground object classes and one background class. The original dataset contains 1464 training, 1499 validation and 1456 testing images, with pixel-level annotations. The performance is mainly measured in terms of pixel IoU averaged across the 21 classes. We train our SPNs on the train split with the coarse segmentation results produced by the FCN-8s model [66]. The model is fine-tuned on a pre-trained VGG-16 network, where different levels of features are upsampled and concatenated to obtain the final, low-resolution

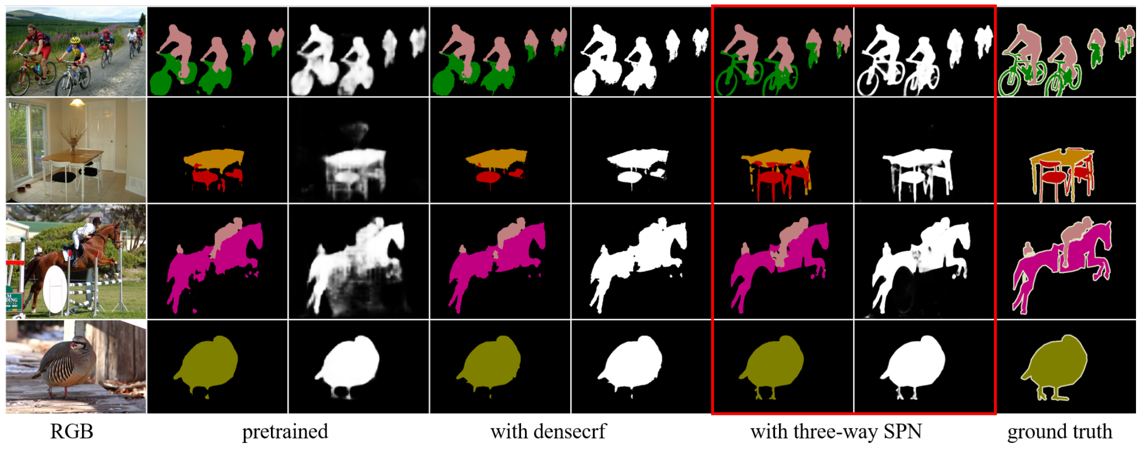


Figure 6.4: Visualization of Pascal VOC segmentation results (left) and object probability (by $1 - P_b$, P_b is the probability of background). The "pretrained" denotes the base Deeplab ResNet-101 model, while the rest 4 columns show the base model combined with the dense CRF [15] and the proposed SPN, respectively.

segmentation results (8× smaller than the original image size). The guidance network of the SPN also fine-tunes the VGG-16 structure from the beginning till the *pool5* layer as the downsampling part. Similar to the settings for the HELEN dataset, the upsampling part has a symmetric structure with skipped links until the feature dimensions of 64×64. The spatial propagation module has the same configuration as that of the SPN that we employed for the HELEN dataset. The model is applied on the coarse segmentation maps of the validation and test splits generated by any image segmentation algorithm without fine-tuning. We test the refinement SPN on three base models: (a) FCN-8s [66], (b) the atrous spatial pyramid pooling (ASPP-L) network fine-tuned with VGG-16, denoted as Deeplab VGG, and (c) the ASPP-L: a multi-scale network fine-tuned with ResNet-101 [43] (pre-trained on the COCO dataset), denoted as Deeplab ResNet-101. Among them, (b) and (c) are the two basic models from [15], which are then refined with dense CRF [51] conditioned on the original image.

Table 6.2 shows that through the three-way SPN, the accuarcy of segmentation is sig-

Table 6.2: Quantitative evaluation results on the Pascal VOC dataset. We compare the two connections of SPN with the corresponding pre-trained models, including: (a) FCN-8s (F), (b) Deeplab VGG (V) and (c) Deeplab ResNet-101 (R). AC denotes accuracy.

model	F	F+1 way	F+3 way	V	V+1 way	V+3 way	R	R+1 way	R+3 way
overall AC	91.22	90.64	**92.90**	92.61	92.16	**93.83**	94.63	94.12	**95.49**
mean AC	77.61	70.64	**79.49**	80.97	73.53	**83.15**	84.16	77.46	**86.09**
mean IoU	65.51	60.95	**69.86**	68.97	64.42	**73.12**	76.46	72.02	**79.76**

Table 6.3: Quantitative comparison (mean IoU) with dense CRF-based refinement [15] on Deeplab pre-trained models.

mean IoU	ConvNet	+dense CRF	+SPN
VGG-16	68.97	71.57	**73.12**
ResNet-101	76.40	77.69	**79.76**

nificantly improved over the coarse segmentation results for all the three base image segmentation models. It has strong capability of generalization, and can successfully refine any coarse maps from different pre-trained models by a large margin. Different with the Helen dataset, the one-way SPN fails to refine the segmentation, which is probably due to its limited capability of learning preferable affinity with a sparse form, especially when the data distribution gets more complex. Table 6.3 shows that by replacing the dense CRF module with the same refinement model, the performance is boosted by a large margin, without fine-tuning. One the test split, the DeepNet ResNet-101 based SPN achieves the mean IoU of **80.22**, while the dense CRF gets 79.7. The three-way SPN produces fine visual results, as shown in the red bounding box of Figure 6.4. By comparing the probability maps (column 3 versus 7), SPN exhibits fundamental improvement in object details, boundaries, and semantic integrity.

6.5 Summary

In this chapter, we propose spatial propagation networks for learning the affinity matrix for vision tasks. The spatial propagation network is a generic framework that can be applied to numerous tasks, and in this work we demonstrate the effectiveness in object segmentation. Experiments on the HELEN face parsing and PASCAL VOC semantic segmentation tasks show that the spatial propagation network is general, effective and efficient for generating high-quality segmentation results.

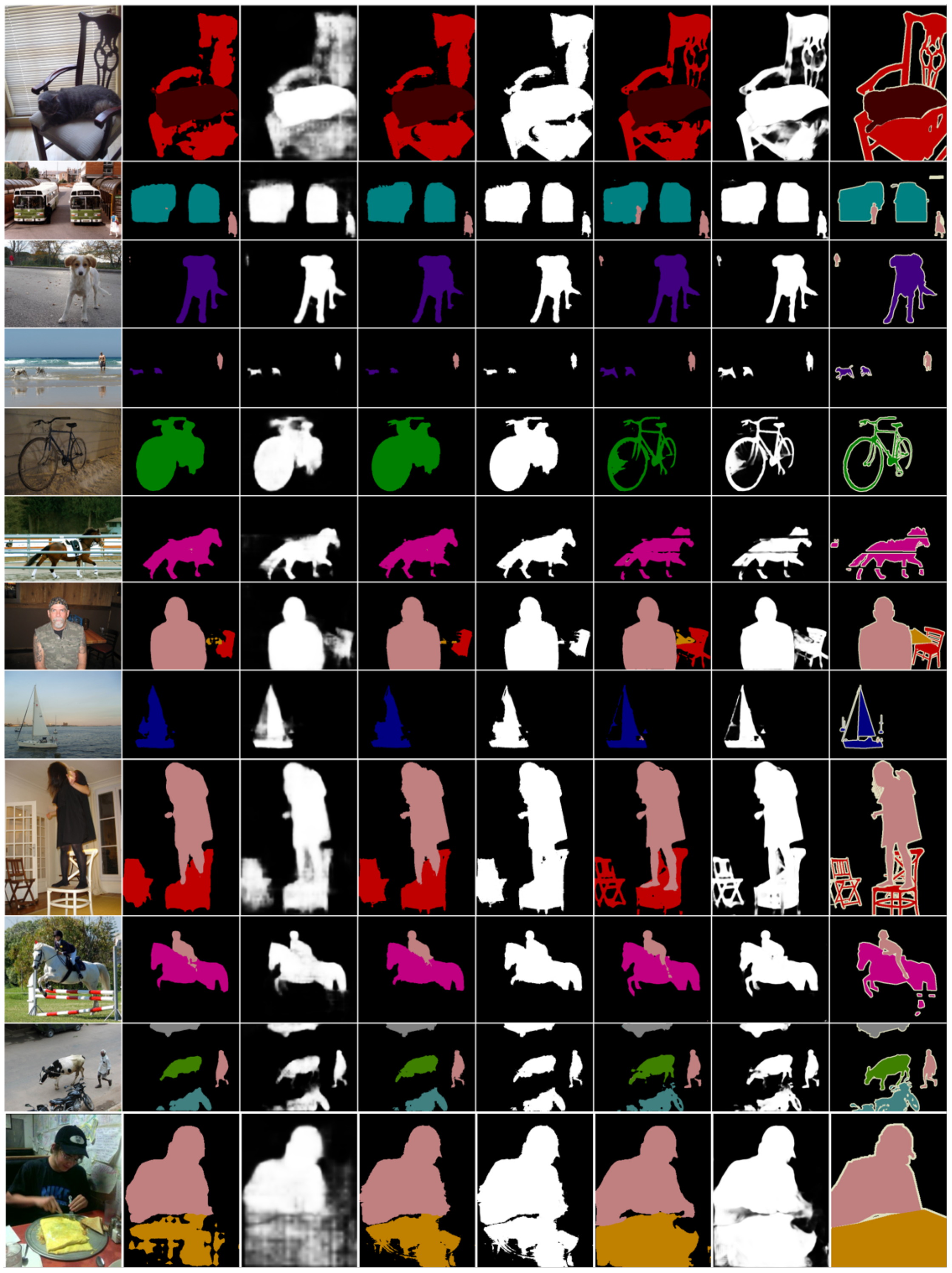


Figure 6.5: Visualization of Pascal VOC segmentation results (left) and object probability (by $1 - P_b$, where P_b denotes the probability of the background region).

Chapter 7

Conclusion and Future Work

7.1 Summary

This thesis investigates problems of how to efficiently and effectively model pixel pairwise relationships in general deep learning framework. We explore a pure ConvNets based approach through introducing multiple objectives in Chapter 3, and new spatial propagation structures that model the pairwise connections explicitly in Chpater 4 5 and 6. We experimentally demonstrate the effectiveness of all the related theories and methods with numerous applications in computer vision. In this Chapter, we specifically discuss the relations among all these approaches according to the properties of their individual affinity, and their performance with respect to the common applications.

Low-level vs high-level for recurrent propagation. In this part, we exploit the relations between Chapter 4 and Chapter 5 that deal with low-level image filtering and high-level semantic segmentation, respectively. Both work utilize very similar, one-dimensional propagation module that either filters the original image, or refines an input coarse segmentation label map. While they are targeting on diverse applications, both models are exploiting to learn their desired affinity matrix on different levels in a data-driven manner. On the other hand, their architectures are different: the low-level vision network takes the propagation model as an image filter which takes in an original image and outputs the filtered result, therefore, the filtering part is very shallow (with one propagation layer, two feature transform layers). The high-level segmentation network, in contrast, takes the module as a label refiner which requires the input to be a coarse semantic label probability map. Therefore, at least one label regression network (e.g., FCN [66]) is needed before the propagation module. Since the propagation guidance corresponding to the semantic boundary can share the network weights with the label regression network, the whole network contains only one sequentially connected ConvNet and a propagation layer. In addition, the motivations of using propagation network are also different: The model in Chapter 4 demonstrates that the affinity for low-level vision problems, or the guidance of propagation, can be learned via an independent ConvNet as deep representations. In contrast, since the one in Chapter 5 is targeting at a fast solution, the affinity is utilized to introduce reflexions to fully convolutional structure in the front end.

One-way connection for Chapter 4 and 5. While we refer the propagation structures proposed in Chapter 4 and 5 as spatially variant recurrent networks (see Section 4.3.1 and 5.2.2), they are substantially reviewed and formulated as "one-way connection" under the general framework of spatial propagation network introduced in Chpater 6. Similarly, the "one-dimensional propagation" refers to the same structure. The reason for using different naming is that Chapter 4 (also mentioned in 5) analyzes the structure with a standard, arbitrary order recursive filter (usually for one-dimensional signal). While we prove that they are equivalent, the name is more suitable for such concept. On the other hand, the "one/three-way connection" describes the spatial propagation with a much border range: the concept is no longer restricted to one-dimensional signals, and can be fitted into the general framework of image diffusion. The name of "one/three-way connection" also helps to distinguish different ways of connections, with respect to their respective properties of the affinity matrix. In addition, as demonstrated in the experimental analysis in Chapter 6.4, the one-way connection is not powerful enough to handle complex semantic segmentation tasks very likely due to the sparsity of its affinity.

Learning affinity for face parsing. The application of face parsing is validated for the majority of methods proposed in the thesis, including Chapter 3, 5 and 6. Compared to the task of semantic segmentation in general or other domains, the domain of faces is more constrained by introducing the face alignment as a preprocessing step, making the problem easier to solve with simpler and lighter network and fewer training data. However, the faster solution proposed in Chpater 5 can be also generalized to more general cases other than facial images. Moreover, the Helen face parsing dataset [1] is a good example for investigating the high-resolution image segmentation problem, such as the relative experiments proposed in Chapter 6.

7.2 Future work

In the near future, we will continue to focus on how to apply the spatial propagation network to a border range of applications, in order to facilitate better models and reduce the data redundancy. For example, a similar framework as Chapter 6 can be utilized to refine the images of optical flow, depth map and surface normal under the guidance of the corresponding RGB images, or applied to more interesting applications, including but not limited to affinity-based image editing and colorization from user-defined scribbles, to name a few. In this section, we are more interested in analyzing the algorithm itself: what is learned from SPN and which applications can be significantly accelerated through it.

7.2.1 What is Learned from Spatial Propagation Network?

A typical way of investigating what is learned from SPN is to visualizing the output feature maps from the guidance network. This is relatively easy for the one-dimensional propagation networks. For example, in Chapter 4, we show the affinity, namely “weight map” by training a single-scale image without any linear transform layers before or after the propagation module, and illustrating the output maps from the guidance network. We specifically use two output maps for the x and y-axis, respectively, as explained in Section 4.4.1 and shown in Figure 4.4 and 4.10. In Chapter 5, the visualization is even easier (see Figure 5.1) since the affinity, namely the "recurrent gates" in this work is directly supervised by the semantic boundaries.

In contrast, visualizing the three-way connection SPN is not easy and direct. A possible solution is as follows:

- Reformulating the output maps into a full matrix according to (6.3). In this way, each direction can obtain an upper/lower triangular matrix.
- We can analyze each axis (e.g., x-axis includes left-to-right and its reverse direction) respectively, by averaging the non-zero part of the upper and lower triangular metrics (e.g., by transposing the upper triangular matrix to a lower one, and averaging it with the lower triangular matrix). We then duplicate this part by summarizing it to its transposed copy, so that to formulate a symmetrical matrix.
- For each axis, we can visualize the spectral gap [53] of the corresponding symmetrical Laplacian matrix. The spectral gap can show rich pixel-level distance or class information that is learned through SPN.

7.2.2 Acceleration for Real-time Rendering

Other than restoring or refining images as mentioned above, SPN also has a great potential in computer graphics, e.g., for acceleration of ray tracing. Recently, although ray tracing generates highly realistic images, simulating millions of virtual light rays for each image still carries a large computational cost. Partially computed images appear noisy, like a photograph taken in extremely low light. The proposed SPN can used to predict final, rendered images from partly finished results with even better computational efficiency than refining a feature map. The main reason is that, the guidance information, which is commonly the original RGB image in computer vision tasks, can be the predefined material and surface normal in the tasks of ray tracing. Once the SPN is trained, the affinity can be preprocessed and stored. During denoising, the propagation network is the only module that is involved in the inference step. The computation can be further accelerated by transferring the recurrent process to direct matrix multiplication, since the multiplication of each individual transform matrix can also be calculated beforehand. Therefore, we can expect high resolution, high quality results generated by the extension method of SPN.

Bibliography

[1] http://www.ifp.illinois.edu/ vuongle2/helen/.

[2] F. Agostinelli, M. R. Anderson, and H. Lee. Adaptive multi-column deep neural networks with application to robust image denoising. In *NIPS*, 2013.

[3] A. Arnab, S. Jayasumana, S. Zheng, and P. H. Torr. Higher order conditional random fields in deep neural networks. In *ECCV*. Springer, 2016.

[4] G. Bertasius, L. Torresani, S. X. Yu, and J. Shi. Convolutional random walk networks for semantic image segmentation. *arXiv preprint arXiv:1605.07681*, 2016.

[5] L. Bertelli, T. Yu, D. Vu, and B. Gokturk. Kernelized structural svm learning for supervised object segmentation. In *CVPR*, 2011.

[6] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence*, 11(6):567–585, 1989.

[7] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *CVPR*, 2012.

[8] X. Burgos-Artizzu, P. Perona, and P. Dollár. Robust face landmark estimation under occlusion. In *CVPR*, 2013.

[9] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. Scene labeling with lstm recurrent neural networks. In *CVPR*, 2015.

[10] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. *International Journal of Computer Vision*, 107(2):177–190, 2014.

[11] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. *ECCV*, 2012.

[12] A. J. Champandard. Semantic style transfer and turning two-bit doodles into fine artworks. *arXiv preprint arXiv:1603.01768*, 2016.

[13] S. Chandra and I. Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs. In *ECCV*, 2016.

[14] L. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. *arXiv preprint arXiv:1511.03328*, 2015.

[15] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

[16] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[17] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016.

[18] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[19] D. Ciresan, A. Giusti, L. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *NIPS*, 2012.

[20] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *Image Processing, IEEE Transactions on*, 16(8):2080–2095, 2007.

[21] G. David, B. Léon, and C. Ronan. Deep convolutional networks for scene parsing. In *ICML Deep Learning Workshop*, 2009.

[22] R. Deriche. *Recursively implementating the Gaussian and its derivatives*. PhD thesis, INRIA, 1993.

[23] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[24] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014.

[25] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*. 2014.

[26] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

[27] M. Everingham, S. A. Eslami, L. V. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.

[28] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE PAMI*, 35(8):1915–1929, 2013.

[29] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. In *ACM TOG*, volume 27, page 67, 2008.

[30] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *CVPR*, 2009.

[31] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *ICCV*, 2009.

[32] E. S. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. In *ACM TOG*, volume 30, page 69, 2011.

[33] E. S. Gastal and M. M. Oliveira. High-order recursive filtering of non-uniformly sampled signals for image and video processing. In *Computer Graphics Forum*, volume 34, pages 81–93, 2015.

[34] S. Geršgorin. Uber die abgrenzung der eigenwerte einer matrix. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, 1931.

[35] J. M. Gonfaus, X. Boix, J. V. D. Weijer, A. D. Bagdanov, J. Serrat, and J. Gonzalez. Harmony potentials for joint classification and segmentation. In *CVPR*, 2010.

[36] A. Graves, S. Fernndez, and J. Schmidhuber. Multi-dimensional recurrent neural networks. In *ICANN*, 2007.

[37] A. Graves, A. rahman Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, pages 6645–6649. IEEE, 2013.

[38] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *NIPS*, 2009.

[39] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. DRAW: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[40] D. Guo and T. Sim. Digital face makeup by example. In *CVPR*, 2009.

[41] A. W. Harley, K. G. Derpanis, and I. Kokkinos. Learning dense convolutional embeddings for semantic segmentation. *arXiv preprint arXiv:1511.04377*, 2015.

[42] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE transactions on pattern analysis and machine intelligence*, 35(6):1397–1409, 2013.

[43] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[44] G. Huang, M. Narayana, and E. Learned-Miller. Towards unconstrained face recognition. In *CVPR Workshop*, 2008.

[45] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[46] K. Iryna, S. Wenzhe, D. Joni, and T. Lucas. Fast face-swap using convolutional neural networks. *arXiv preprint arXiv:1611.09577*, 2016.

[47] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[48] A. Kae, K. Sohn, H. Lee, and E. Learned-Miller. Augmenting CRFs with Boltzmann machine shape priors for image labeling. In *CVPR*, 2013.

[49] N. Kalchbrenner, I. Danihelka, and A. Graves. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*, 2015.

[50] L. Karacan, E. Erdem, and A. Erdem. Structure-preserving image smoothing via region covariances. *ACM TOG*, 32:176, 2013.

[51] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[53] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM TOG*, 23(3):689–694, 2004.

[54] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *IEEE PAMI*, 30(2):228–242, 2008.

[55] X. Liang, X. Shen, D. Xiang, J. Feng, L. Lin, and S. Yan. Semantic object parsing with local-global long short-term memory. *arXiv preprint arXiv:1511.04510*, 2015.

[56] G. Lin, C. Shen, I. D. Reid, and A. van den Hengel. Deeply learning the messages in message passing inference. *arXiv preprint arXiv:1506.02108*, 2015.

[57] T. Lin, M. Maire, S. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. 2014.

[58] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing via label transfer. *IEEE PAMI*, 33(12):2368–2382, 2011.

[59] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing via label transfer. *PAMI*, 2011.

[60] R. Liu, G. Zhong, J. Cao, Z. Lin, S. Shan, and Z. Luo. Learning to diffuse: A new perspective to design pdes for visual analysis. *IEEE transactions on pattern analysis and machine intelligence*, 38(12):2457–2471, 2016.

[61] S. Liu, S. D. Mello, J. Gu, G. Zhong, M.-H. Yang, and J. Kautz. Learning affinity via spatial propagation networks. In *NIPS*, 2017.

[62] S. Liu, J. Pan, and M.-H. Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *ECCV*, 2016.

[63] S. Liu, J. Shi, J. Liang, and M.-H. Yang. Face parsing via recurrent propagation. In *BMVC*, 2017.

[64] S. Liu, J. Yang, C. Huang, and M.-H. Yang. Multi-objective convolutional learning for face labeling. In *CVPR*, 2015.

[65] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *ICCV*, 2015.

[66] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[67] P. Luo, X. Wang, and X. Tang. Hierarchical face parsing via deep learning. In *CVPR*. IEEE, 2012.

[68] P. Luo, X. Wang, and X. Tang. Hierarchical face parsing via deep learning. In *CVPR*, 2012.

[69] M. Maire, T. Narihira, and S. X. Yu. Affinity CNN: learning pixel-centric pairwise relations for figure/ground embedding. *CoRR*, abs/1512.02767, 2015.

[70] S. Martin, K. Pushmeet, and H. Derek. Learning CRFs using graph cuts. In *ECCV*, 2008.

[71] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. E. Barbano. Toward automatic phenotyping of developing embryos from videos. *TIP*, 14(9):1360–1371, 2005.

[72] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

[73] A. V. D. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[74] S. Osher and L. I. Rudin. Feature-oriented image enhancement using shock filters. *SIAM Journal on Numerical Analysis*, 27(4):919–940, 1990.

[75] S. Paris, S. W. Hasinoff, and J. Kautz. Local laplacian filters: edge-aware image processing with a laplacian pyramid. *ACM Trans. Graph.*, 30(4):68, 2011.

[76] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *ACM TOG*, pages 313–318, 2003.

[77] P. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene parsing. *arXiv preprint arXiv:1306.2795*, 2013.

[78] G. Proakis John and G. Manolakis Dimitris. Digital signal processing, principles, algorithms, and applications. *Pentice Hall*, 1996.

[79] H. Qin, J. Yan, X. Li, and X. Hu. Joint training of cascaded cnn for face detection. In *CVPR*, 2016.

[80] S. Qin, S. Kim, and R. Manduchi. Automatic skin and hair masking using fully convolutional networks. In *ICME*, 2017.

[81] R. Ranftl and T. Pock. A deep variational model for image segmentation. In X. Jiang, J. Hornegger, and R. Koch, editors, *Pattern Recognition*, Lecture Notes in Computer Science, pages 107–118. Springer International Publishing, 2014.

[82] J. S. Ren, L. Xu, Q. Yan, and W. Sun. Shepard convolutional neural networks. In *NIPS*. 2015.

[83] J. S. J. Ren and L. Xu. On vectorization of deep convolutional neural networks for vision tasks. In *AAAI*, 2015.

[84] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

[85] H. Schulz and S. Behnke. Learning object-class segmentation with convolutional neural networks. In *ESANN*, volume 3, page 1, 2012.

[86] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015.

[87] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[88] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[89] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008.

[90] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *Int. Journal of Computer Vision (IJCV)*, 2009.

[91] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[92] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[93] B. Smith, L. Zhang, J. Brandt, Z. Lin, and J. Yang. Exemplar-based face parsing. In *CVPR*, 2013.

[94] B. M. Smith, L. Zhang, J. Brandt, Z. Lin, and J. Yang. Exemplar-based face parsing. In *CVPR*, 2013.

[95] R. Socher, C. Lin, C. Manning, and A. Y. Ng. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, 2011.

[96] M. F. Stollenga, W. Byeon, M. Liwicki, and J. Schmidhuber. Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. *arXiv preprint arXiv:1506.07452*, 2015.

[97] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *CVPR*, 2013.

[98] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *CVPR*, 2013.

[99] S. Tan, J. L. Dale, and A. Johnston. Performance of three recursive algorithms for fast space-variant gaussian filtering. *Real-Time Imaging*, 9(3):215–228, 2003.

[100] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998.

[101] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. *arXiv preprint arXiv:1406.2984*, 2014.

[102] S. Tsogkas, I. Kokkinos, G. Papandreou, and A. Vedaldi. Semantic part segmentation with deep learning. *arXiv preprint arXiv:1505.02438*, 2015.

[103] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[104] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[105] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. Courville, and Y. Bengio. Renet: A recurrent neural network based alternative to convolutional networks. *arXiv preprint arXiv:1505.00393*, 2015.

[106] J. Warrell and S. J. Prince. Labelfaces: Parsing facial features by multiclass labeling with an epitome prior. In *ICIP*, 2009.

[107] J. Weickert. *Anisotropic Diffusion in Image Processing*. B.G. Teubner Stuttgart, 1998.

[108] J. Weickert. *Anisotropic diffusion in image processing*, volume 1. Teubner Stuttgart, 1998.

[109] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. *Back-propagation: Theory, architectures and applications*, pages 433–486, 1995.

[110] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *NIPS*, 2012.

[111] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via l-0 gradient minimization. In *ACM TOG*, page 174, 2011.

[112] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *NIPS*, 2014.

[113] L. Xu, J. S. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *ICML*, 2015.

[114] L. Xu, Q. Yan, and J. Jia. A sparse control model for image and video editing. *ACM TOG*, 32(6):197:1–197:10, 2013.

[115] L. Xu, Q. Yan, Y. Xia, and J. Jia. Structure extraction from texture via relative total variation. *ACM TOG*, 31(6):139, 2012.

[116] T. Yamashita, T. Nakamura, H. Fukui, Y. Yamauchi, and H. Fujiyoshi. Cost-alleviative learning for deep convolutional neural network-based facial part labeling. *IPSJ Transactions on Computer Vision and Applications*, 7:99–103, 2015.

[117] J. Yang, B. Price, S. Cohen, and M.-H. Yang. Context driven scene parsing with attention to rare classes. In *CVPR*, 2014.

[118] J. Yang, Y.-H. Tsai, and M.-H. Yang. Exemplar cut. In *CVPR*, 2013.

[119] Q. Yang. Recursive bilateral filtering. In *ECCV*. 2012.

[120] I. T. Young and L. J. V. Vliet. Recursive implementation of the gaussian filter. *Signal processing*, 44(2):139–151, 1995.

[121] Q. Zhang, X. Shen, L. Xu, and J. Jia. Rolling guidance filter. In *ECCV*. 2014.

[122] Q. Zhang, L. Xu, and J. Jia. 100+ times faster weighted median filter (wmf). In *CVPR*, 2014.

[123] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *ECCV*, 2014.

[124] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[125] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015.

[126] S. Zhu, C. Li, C. C. Loy, and X. Tang. Face alignment by coarse-to-fine shape searching. In *CVPR*, 2015.

[127] J. G. Zilly, R. K. Srivastava, J. Koutník, and J. Schmidhuber. Recurrent highway networks. *arXiv preprint arXiv:1607.03474*, 2016.

[128] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *ICCV*, 2011.

[129] Z. Zuo, B. Shuai, G. Wang, X. Liu, X. Wang, B. Wang, and Y. Chen. Convolutional recurrent neural networks: Learning spatial dependencies for image representation. In *CVPR Workshops*, 2015.