博士論文

Neighbor-Aware Approaches for Pixel Labeling
(近傍を考慮した画素ラベリング)

48-167405
古田　諒佑

指導教員　山﨑　俊彦　准教授
東京大学大学院　情報理工学系研究科　電子情報学専攻

This dissertation is submitted for the degree of
Doctor of Philosophy

December 2018

Acknowledgements

First of all, I thank my supervisor, Prof. Toshihiko Yamasaki. He provided me kind and excellent guidance not only about the research but also about the other various things. I also thank Prof. Kiyoharu Aizawa for advising me about the research direction.

I would like to thank my advising professors, Prof. Shin'ichi Satoh and Prof. Takeshi Oishi for the discussions and their useful advises. I would also like to thank the referees, Prof. Yoichi Sato and Prof. Takeshi Naemura for their useful comments.

I am grateful to Prof. Satoshi Ikehata of National Institute of Informatics. I learned how to do research from him. I would like to thank Prof. Ikuko Tsubaki of Tokyo University of Technology for the collaborative research. I would like to thank Dr. Tatsunori Taniai of RIKEN AIP Center for his help with PatchMatch Filter code.

I also thank all of our lab members for discussing and chatting me. Finally, I would like to express the deepest appreciation to my parents for supporting me.

Abstract

Pixel labeling is one of the most classical and important problems in the field of computer vision because it has a variety of applications. In this thesis, we tackle two major challenges of pixel labeling: (i) **how to deal with the large solution space,** and (ii) **how to learn the relationships between neighbor labels effectively.**

For the first challenge, we propose two neighbor-aware fast optimization methods.

- **Neighbor-aware fast optimization for general MRF**: One is the fast optimization method for general pixel-labeling problems based on Markov random field (MRF) models where the smoothness between the neighbor labels is forced. We focus on an optimization method called cost-volume filtering (CVF) and propose a coarse-to-fine strategy for CVF that efficiently and accurately addresses pixel-labeling problems with a large label space size. Experimental results show that our algorithm achieves much higher efficiency than the original CVF method while maintaining a comparable level of accuracy on stereo matching and optical flow estimation.

- **Neighbor-aware fast optimization for special MRF**: The other is the fast optimization method for special case of pixel-labeling problems where the neighbor labels are forced to be connected. We propose a fast optimization method named "multi-pass dynamic programming" for this optimization problem, which is approximately 90 times faster and consumes 8 times less memory than conventional graph cuts methods. The main application of this optimization problem is volume seam carving (seam carving for 3D cost volume), which is applied to various of image processing tasks such as video retargeting, tone mapping, and contrast enhancement.

For the second challenge, we propose two novel neighbor-aware learning methods that boost the performance of pixel labeling.

- **Learning neighbors with convolutional neural network**: We reveal the mathematical relationship between the fixed point iteration of dense CRF and recurrent convolution. Based on this interpretation, we propose a new model based on dense CRF, which automatically learns the relationships between neighbor labels from training data and enables jointly train with deep neural networks. The proposed dense CRF can be incorporated into fully convolutional network (FCN) as a module and trained end-to-end. Experimental results show that our method obtains better results on semantic segmentation, compared with the existing methods based on hand-crafted CRF.

- **Learning neighbors with deep reinforcement learning**: We propose a completely novel problem setting (pixelRL) and an effective neighbor-aware learning method for pixelRL named reward map convolution. PixelRL is a novel pixel-labeling problem combined with reinforcement learning, where the label is a sequence of actions at each pixel, and its objective is to maximize the accumulated total rewards at all pixels. We apply the proposed method to three image processing tasks: image denoising, image restoration, and local color enhancement. Our experimental results demonstrate that the proposed method achieves comparable or better performance, compared with the state-of-the-art methods based on supervised learning at each task.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Pixel Labeling

Pixel labeling is one of the most classical and important problems in the field of computer vision because it has a variety of applications such as stereo matching [7], optical flow estimation [8], image segmentation [9] and so on. The objective of pixel labeling is to assign a label l_i to each pixel $i \in \{1, \cdots, N\}$ appropriately. Formally, it is defined as an energy minimization problem to obtain an optimal labeling $\boldsymbol{l} = (l_1, \cdots, l_N)$:

$$\boldsymbol{l}^* = \arg\min_{\boldsymbol{l}} E(\boldsymbol{x}, \boldsymbol{l}), \tag{1.1}$$

where $\boldsymbol{x} = (x_1, \cdots, x_n)$ is the input image, and $E(\boldsymbol{x}, \boldsymbol{l})$ is the energy function (also called cost function).

Let's start with the simplest case of pixel labeling, where the energy function is defined independently at each pixel:

$$E(\boldsymbol{x}, \boldsymbol{l}) = \sum_i \phi_i(\boldsymbol{x}, l_i). \tag{1.2}$$

In this case, it is a trivial problem because we can divide it into N independent subproblems and can obtain optimal label at each pixel:

$$l_i^* = \arg\min_{l_i} \phi_i(\boldsymbol{x}, l_i). \tag{1.3}$$

However, it is generally said that this simple modeling causes noisy and poor results because the relationships (smoothness, co-occurrence, exclusiveness and so on) between labels at neighbor pixels are not taken into account.

Therefore, in this thesis, we tackle **neighbor-aware** pixel labeling, which is formulated as follows:

$$E(\boldsymbol{x},\boldsymbol{l}) = \sum_i \phi_i(\boldsymbol{x},l_i) + \sum_i \psi_{\mathcal{N}_i}(\boldsymbol{l}_{\mathcal{N}_i}), \tag{1.4}$$

where $\mathcal{N}_i$ is the set of neighbor pixels around i, and $\boldsymbol{l}_{\mathcal{N}_i}$ is the labeling assigned to $\mathcal{N}_i$. This model is called Markov/conditional random field (MRF/CRF). $\phi_i(\boldsymbol{x},l_i)$ is the unary term, and $\psi_{\mathcal{N}_i}(\boldsymbol{l}_{\mathcal{N}_i})$ represents the relationships between neighbor labels. In neighbor-aware pixel labeling, the optimization problem in Eq. (1.4) is no longer trivial, and there are two major challenges: (i) **how to deal with the large solution space,** and (ii) **how to learn the relationships between neighbor labels effectively.**

1.2 Neighbor-Aware Fast Optimizations

Here, we discuss the first challenge. When the label candidates (pre-defined label set) is $l_i \in \{1,...,L\}$, the entire solution space is $O(L^N)$, where the brute force search is computationally impractical. If we don't consider the relationships between neighbor labels (*i.e.*, Eq. (1.2)), we can obtain the optimal labeling by Eq. (1.3) with the computational cost of $O(LN)$. In contrast, in the case of neighbor-aware pixel labeling in Eq. (1.4), efficient optimization methods are required. To this end, we propose neighbor-aware fast optimization methods in chapter 2 and 3, respectively.

In chapter 2, we tackle the pixel labeling with general MRF model, where the smoothness between neighbor labels are forced. We focus on an optimization method called cost-volume filtering (CVF), which is one of the most widely used techniques for solving general pixel-labeling problems based on a Markov random field (MRF). Although CVF is easy to implement and provides high-quality results, it is inefficient when the label space size (*i.e.*, the number of labels) is large. Therefore, we presents a coarse-to-fine strategy for cost-volume filtering that efficiently and accurately addresses pixel-labeling problems with a large label space size. Based on the observation that true labels at the same coordinates in images of different scales are highly correlated, we truncate unimportant labels in each local region by leveraging the labeling output of lower scales. Experimental results show that our algorithm achieves much higher efficiency than the original CVF method while maintaining a comparable level of accuracy. Although we performed experiments that deal with only stereo matching and optical flow estimation, the proposed method can be employed in many other applications because of the applicability of CVF to general discrete pixel-labeling problems based on an MRF.

In chapter 3, we focus on the special case of MRF optimization in Eqs. (1.5) and (1.6).

$$\arg\min_{\boldsymbol{l}} E(\boldsymbol{l}) = \arg\min_{\boldsymbol{l}} \sum_i \phi_i(l_i) + \sum_{(i,j)\in\mathcal{N}} \psi_{i,j}(l_i,l_j) \tag{1.5}$$

$$\psi_{i,j}(l_i,l_j) = \begin{cases} 0 & \text{if } |l_i - l_j| \leq 1 \\ \infty & \text{otherwise,} \end{cases} \tag{1.6}$$

where $\mathcal{N}$ is a set of all neighbor pixels. By the constraint in Eq. (1.6), the neighbor labels are forced to be connected, in other words, the differences between two labels assigned to neighbor pixels are forced to be less than one. To date, the graph cuts algorithm [10] has been the only choice in this special case of MRF optimization. However, graph cuts algorithm requires a tremendous amount of computational time and memory when the number of nodes and edges increases. Therefore, we propose a fast optimization method named "multi-pass dynamic programming" for this optimization problem, which is approximately 90 times faster and consumes 8 times less memory than conventional graph cuts methods. The main application of this optimization problem is volume seam carving (seam carving for 3D cost volume), which is applied to various of image processing tasks such as video retargeting [10], video summarization [11, 12], tone mapping [13], contrast enhancement, and depth remapping [14]. The volume seam carving procedure is as follows: A cost volume $C(x,y,z)$ is first created, then a seam surface $\boldsymbol{z} = z(x,y)$ is determined that affects the cost less if it is removed. That seam surface is then removed, and the procedure is repeated until the targeted 3D volume is obtained. A seam surface is a two-dimensional (2D) manifold in the cost volume. In the procedure, to determine a seam surface is an optimization problem, which is described as

$$\arg\min_{\boldsymbol{z}} \sum_{x,y} C(x,y,z(x,y)), \tag{1.7}$$

$$s.t.\ |z(x,y) - z(x+1,y)| \leq 1, \tag{1.8}$$

$$|z(x,y) - z(x,y+1)| \leq 1. \tag{1.9}$$

We can regard this optimization problem as one of the MRF optimization in Eqs. (1.5) and (1.6) by setting the energy function as

$$\arg\min_{\boldsymbol{z}} E(\boldsymbol{z}) = \arg\min_{\boldsymbol{z}} \sum_i \phi_i(z_i) + \sum_{(i,j)\in\mathcal{N}} \psi_{i,j}(z_i, z_j), \tag{1.10}$$

$$\phi_i(z_i) = C(x, y, z(x,y)), \tag{1.11}$$

$$\psi_{i,j}(z_i, z_j) = \begin{cases} 0 & \text{if } |z_i - z_j| \le 1 \\ \infty & \text{otherwise,} \end{cases} \tag{1.12}$$

where $i = (x, y)$ is a pixel position and the label $z_i = z(x, y)$ is the seam surface at the position. Therefore, our fast optimization method for the MRF in Eqs. (1.5) and (1.6) makes various of image processing techniques based on volume seam carving more practical.

1.3 Learning Neighbors

As discussed in Sec. 1.1, the relationships between neighbor labels can be taken into account with MRF/CRF models. The remaining question is how to learn the relationships between the neighbor labels effectively. To answer the question, in chapter 4 and 5, we propose novel neighbor-aware learning methods that boost the performance of pixel labeling.

Most of existing works use hand-crafted CRFs to represent the relationships between neighbor labels, which require domain knowledge by humans. To solve the problem, we propose a new model based on dense CRF, which automatically learns the relationships between neighbor labels from training data and enables jointly train with deep neural networks in chapter 4. We reveal the mathematical relationship between the fixed point iteration of dense CRF and recurrent convolution, and by this interpretation, the dense CRF can be incorporated into fully convolutional network (FCN) as a module and trained end-to-end. Experimental results show that our method obtains better results on PASCAL VOC semantic segmentation benchmark, compared with the existing methods based on hand-crafted CRF.

In chapter 5, we propose a completely novel problem setting (pixelRL) and an effective neighbor-aware learning method for pixelRL named reward map convolution. PixelRL is a novel pixel-labeling problem combined with reinforcement learning, where the label is a sequence of actions $\boldsymbol{a}_i = (a_i^{(1)}, \cdots, a_i^{(T)})$ at each pixel, and its objective is to maximize the accumulated total rewards at all pixels. In pixelRL, each pixel has an agent, and the agent changes the pixel value by taking an action. The proposed reward map convolution significantly improves the performance by considering not only the future states of the own pixel but also those of the neighbor pixels. The proposed method can be applied to some image processing tasks that require pixel-wise manipulations, where deep RL has never been applied. We apply the proposed method to three image processing tasks: image denoising, image restoration, and local color enhancement. Our experimental results demonstrate that the proposed method achieves comparable or better performance, compared with the state-of-the-art methods based on supervised learning at each task.

Chapter 2

Neighbor-Aware Fast Optimization for General MRF

2.1 Introduction

Many low-level computer-vision problems (*e.g.*, stereo matching and optical flow estimation) are formulated as multi-labeling problems, where discrete labels (*e.g.*, disparity and motion vector) are assigned to pixels. In general, there are two approaches to solve these problems: global and local. The former models a labeling problem as a Markov random field (MRF), where global optimization techniques [15–22] are used to minimize the energy function. Although such an approach is effective, using it to solve a large optimization problem makes the inference intractable when the image size or label space is large. Rhemann *et al*. [1] presented a local approach called *cost-volume filtering* (CVF), which efficiently solves general multi-labeling problems by performing MRF optimization via fast local filtering of label costs instead of global smoothing. CVF is easy to implement and provides high-quality results; therefore, it has been widely used to solve various multi-labeling problems [23–27]. However, a limitation of CVF is that it does not scale to extremely large label sets (*e.g.*, sub-pixel stereo matching and up-sampling of 16-bit depth maps captured by a Kinect sensor).

To overcome this limitation, Lu *et al*. [28] proposed the PatchMatch filter (PMF), which performs CVF iteratively on local superpixels with compact label subsets instead of performing it on the entire image coordinate space. In general, the average size of local label subsets is much smaller than the size of the entire label space; therefore, although PMF and CVF provide similar levels of accuracy, the efficiency of

PMF is considerably higher. Nevertheless, PMF relies on global optimization based on the complex PatchMatch approach [29, 30] to estimate a label subset for each superpixel. Thus, the computational complexity of PMF increases with the number of superpixels, and therefore, PMF becomes less effective when an image is divided into many superpixels.

This chapter presents an alternative coarse-to-fine strategy for efficiently estimating compact label subsets to solve the label space problem in cost-volume filtering. Based on the observation that true labels at the same coordinates in an image of different scales are highly correlated, we propose that lower-scale labeling outputs be leveraged for estimating higher-scale local label subsets. Starting with an image of very low-resolution, we iteratively truncate unimportant labels at each higher scale, and finally, we assign compact and approximately optimal label subsets to local regions of the original scale. The advantage of the proposed framework is a simple and efficient coarse-to-fine strategy, which does not require any global optimization as in [28]; moreover, its computational complexity is not significantly affected by the number of local regions. Extensive experiments described in Sec. 2.4 show that our algorithm achieves higher efficiency than PMF and CVF while providing a comparable or often superior level of accuracy.

Note that we are not proposing a better algorithm for stereo matching and optical flow estimation, but proposing a coarse-to-fine method to drastically reduce the computational time of CVF while preserving its accuracy. As presented in [1, 24–27], the CVF can be used for wide range of applications and the stereo matching and the optical flow estimation presented in this chapter is just an example.

Our proposed algorithm can be directly applied to not only original CVF [1] but also several of its variants picked up in Sec. 2.2. In addition, our proposed algorithm can be implemented on GPU similar to the original CVF. However, in this chapter, we did not perform those implementations, and compared with only the original CVF because we focus on "how to deal with the large label space efficiently", not to improve the accuracy and not the real-time application.

The reminder of this chapter is organized as follows. Section 2.2 reviews related studies. Section 2.3 briefly reviews CVF [1] and describes the details of the proposed coarse-to-fine strategy. Section 2.4 presents the experimental results and describes their evaluation using the Middlebury benchmark [31, 32]. Finally, Section 2.5 summarizes our findings and concludes the chapter.

2.2 Related Works

In this section, we mainly focus on related works about stereo matching and optical flow estimation, because they are main problems among multi-labeling problems and a lot of methods using cost-volume filtering techniques have been proposed in stereo matching and optical flow estimation. However, as mentioned in Sec. 2.1, the cost-volume filtering technique is not only used for them but also applied to wide range of multi-labeling problems such as image segmentation [26], and depth-map up-sampling [27].

2.2.1 Cost Aggregation Methods for Labeling Problems

First, we review cost aggregation methods for correspondence field estimation. Yoon and Kweon [33, 34] proposed a cost aggregation method using an adaptive weighted window such as an edge-preserving bilateral filter [35]. This method is slow because it needs to perform naive bilateral filtering iteratively, where the number of iterations is equal to the number of disparity candidates. To address this problem, Richard *et al.* [36] proposed an approximate bilateral filtering technique that reduces the computational complexity of adaptive support weight calculation. However, this approach provides low-quality results , as compared to state-of-the-art stereo matching methods. On the other hand, Yang [23, 37] proposed a tree-based non-local cost aggregation method using a minimum spanning tree. This method aggregates the cost values based on a tree structure constructed using input images, and the final disparity refinement process is also performed on the basis of the tree structure. Bai *et al.* [38] proposed an algorithm based on loop-erased random walk to improve the support weighted window of [23] near depth discontinuities. As stated in Sec. 2.1, Rhemann *et al.* [1] proposed CVF for general multi-labeling problems. By using an $O(1)$ edge-preserving filter called a guided filter (GF) [39] for cost aggregation, CVF can efficiently solve general multi-labeling problems and achieve high-quality results. Lu *et al.* [40] proposed a new edge-preserving filter called a cross-based local multipoint filter (CLMF), which is an extension of the GF. Although the shape of the local support region of the GF is a square, that of the CLMF can be an adaptively derived from a reference image. Further, Lu *et al.* [40] showed that higher-quality stereo matching results can be achieved by applying the CLMF instead of the GF for cost aggregation. Zhang *et al.* [41] proposed a cross-scale cost aggregation algorithm based on CVF [1] for stereo matching. They showed that higher-quality disparity maps can be obtained by adding a regularization term between the cost values of different scales, and that the computational time of cross-scale aggregation is not significantly greater than that of the original CVF [1]. This method [41] is similar to ours in terms of multi-scale cost-volume utilization, but its purpose is to improve the quality of the disparity maps, not to reduce the computational complexity. Recently, Zhan *et al*. [42] proposed some techniques for local stereo matching methods to improve the accuracy: mask filtering as a pre-processing, an improved matching cost function, and multi-step disparity refinement as a post-processing. Inspired by the great success of convolutional neural networks (CNNs) in image recognition task, CNNs are recently used for computing the label costs (matching costs in stereo matching and optical flow estimation) instead of hand-crafted cost functions [43–47], which has led to significant improvement in terms of accuracy. In MC-CNN [44, 46], the CNN directly outputs the matching cost of two input patches. Cross-based cost aggregation and semi-global matching are preformed for the obtained cost-volume to produce accurate disparity map. To speed up computing the matching cost, Chen *et al*. [45] and Luo *et al*. [47] proposed similar ideas, where the matching cost is defined as the inner product of two features from CNN. In FlowNet [43], the matching costs are defined as the correlation between two patches of feature maps, and the final flow map is obtained by upconvolution operation. The computation of the correlations is implemented as correlation layer, which is incorporated into CNN.

Most of local methods perform cost aggregation for all the labels (disparities) at every pixel. Therefore, those methods are limited in that they do not scale to extremely large label sets. To overcome this problem, with regard to stereo matching, Min *et al*. [48, 49] proposed a technique to estimate a compact disparity subset for every pixel by considering disparities with the local minima of the pre-filtered cost values. Although this method efficiently achieves high-quality results with the Middlebury stereo benchmark [31], it cannot be applied to general multi-labeling problems directly. Wang *et al*. [50] adapted the sequential probability ratio test to reduce the disparity search range with the sufficient confidence in stereo matching problem. Helala and Qureshi [51] proposed the Accelerated CVF using an occulusion handling technique for stereo matching problem. For general multi-labeling problems, Lu *et al*. [28] proposed PMF, which is based on CVF [1]. As mentioned in Sec. 2.1, PMF estimates a compact label subset for every superpixel using the PatchMatch [29, 30] strategy; therefore, it is usually much more efficient than CVF while maintaining a similar level of accuracy. However, because PMF relies on complex PatchMatch-based global optimization to estimate a label subset for each superpixel, it becomes less effective when an image is divided into many superpixels.

2.2.2 Coarse-to-Fine Strategy

Coarse-to-fine strategies have been employed in a variety of methods for labeling problems such as stereo matching and optical flow estimation. We can classify them into two types: the coarse-to-fine strategies where the cost aggregation results from all resolution are merged in order to obtain more accurate results such as [52, 53, 41], and ones where the results of lower resolution are propagated to higher resolution in order to reduce the search range of labels such as [54, 55]. We focus on the latter because our method is classified into latter group.

Brox *et al*. [56] employed a coarse-to-fine strategy in their global optimization framework to estimate a optical flow field. They obtain an output flow field as the solution of their energy minimization formulation by solving Euler-Lagrange equations. They supplied a theoretical explanation that justifies their coarse-to-fine strategy by regarding it as a part of the two nested iterations for non-convex optimization, and argued that their coarse-to-fine strategy helps the convergence to the global minimum by setting the solution of coarser scale to the initialization of the next finer scale. Similar to [56], Wedal *et al*. [57] employed a coarse-to-fine strategy in their optical flow estimation framework, where the flow field is obtained by solving the total variation (L1 norm) minimization problem using linear approximation and alternating optimization scheme. They argued that their coarse-to-fine strategy has the advantage of avoiding poor local minimum by propagating the solution of coarser scale to the finer scale. Those coarse-to-fine strategies such as [56, 57] are tailored for global optimization techniques. These method iteratively update one solution for the entire image and propagate it to the next scale after the predetermined number of iterations. Therefore, their coarse-to-fine approaches cannot be used for CVF which needs pixel-wise cost computation for all possible labels and obtains pixel-wise solutions by winner-take-all strategy. Yang *et al*. [55] proposed a coarse-to-fine technique for belief propagation (BP), which reduces the computational complexity in both spatial and depth domain. This method is tailored for BP and cannot be directly applied to CVF. Different from these approaches, we propose a coarse-to-fine strategy for the cost-volume filtering technique that is categorized in local methods.

Next, we discuss the coarse-to-fine strategies employed in local cost aggregation methods which are close to our method. Zhao *et al*. [54] employed a coarse-to-fine strategy in their elegant implementation on GPGPU for real-time stereo. They limit the search range within ± 2 pixels of the disparity value obtained in lower resolution. The main difference between their method and ours is that the reduction of the search range is performed per pixel in their method, while it is done in each local region in our method. In addition, the comparison with their method has little meaning because their objective is the efficient disparity estimation in only foreground region and their algorithm is optimized for it. Their experimental results on Middlebury stereo datasets with the assumption that whole image area is foreground show the poor accuracy especially around the object boundaries (Disc. in Table 1 [54]). Tao *et al.* [58] proposed a multiscale local cost aggregation method for optical flow estimation called SimpleFlow. They upsampled the flow field obtained at the coarser scale and skipped the cost computation by interpolating the flow using simple bilinear interpolation in the regions where the flow was smooth. Therefore, their method can obtain a flow field with sublinear time with respect to the size of input images. Thier coarse-to-fine strategy is different from ours because our method estimates a compact label set in each local region to handle the large label space. In addition, without the refinement using the global optimization [59], the accuracy of the flow fields obtained by SimpleFlow [58] is much lower than that of CVF [1]. Although the SimpleFlow with the refinement can obtain the comparable accuracy to the CVF, the running time drastically increases because the global optimization in the refinement process is computationally expensive (Table 4 in [60]). On the other hand, our method can obtain comparable accuracy to the CVF [1] and is several times faster than CVF. Bao *et al.* [60, 61] proposed a fast edge-preserving PatchMatch for optical flow estimation. Their method estimates an approximate nearest neighbor field (NNF) using PatchMatch search at the coarsest scale, and repeats upsampling the NNF and the refinement of it within a small search range (3×3 pixels) until the original resolution. Their method is very fast and can achieve high-quality results for large displacement optical flow. However, for the datasets with small displacement optical flow, their coarse-to-fine strategy obtains the less accurate results than when without it (Table 4 [60]) because their method is tailored for large displacement optical flow. In contrast, our coarse-to-fine strategy for general multi-labeling problems obtains the comparable or more accurate results than the original CVF both when the label space is small and large.

2.3 Coarse-to-Fine Strategy for Efficient CVF

In this section, we present a coarse-to-fine strategy for CVF [1] in order to address multi-labeling problems with a large label space. Given a label set $\mathcal{L} = \{l_0, \cdots, l_{L-1}\}$, the objective of a multi-labeling problem is to assign a label $l_i \in \mathcal{L}$ to each pixel $i \triangleq [x_i,\ y_i]$ $(i = 0, \ldots, M-1)$ in the image coordinate space I such that it minimizes the

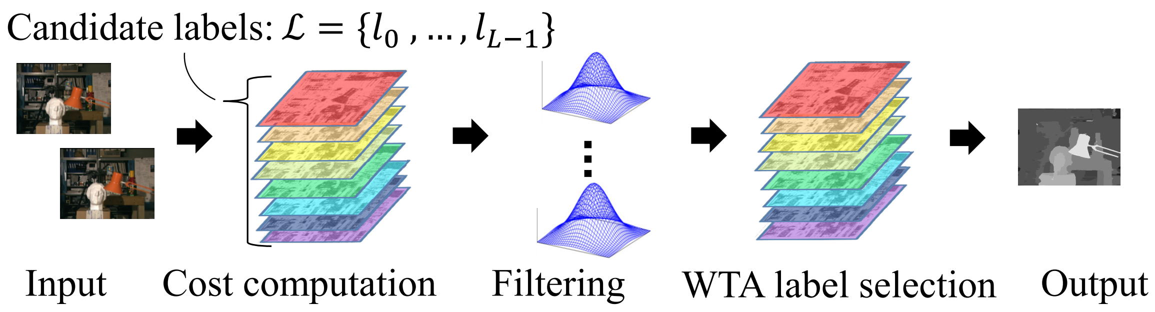


Figure 2.1: Framework of CVF [1].

label costs encoded in the energy function [1]. Here, L and M denote the number of labels and the number of pixels, respectively.

2.3.1 CVF

The outline of CVF [1] is shown in Fig. 2.1. CVF solves a multi-labeling problem in three steps. First, a 3-D cost volume C is constructed as a collection of costs $C(i,l)$ for selecting label l at each pixel i on the basis of the data term in the energy function. Then, each slice of the cost volume is independently filtered by an edge-preserving filter [39, 40], which is substituted for the smoothness term in the energy function:

$$C(i,l) \leftarrow \sum_{i' \in \omega_i} W_{ii'} C(i',l), \tag{2.1}$$

where ω_i is the squared window centered at the pixel i. Finally, the label at pixel i is simply selected by the winner-takes-all (WTA) strategy:

$$l_i = \arg\min_{l \in \mathcal{L}} C(i,l). \tag{2.2}$$

When an $O(1)$ edge-preserving filter (*e.g.*, guided filter [39]) is used, the computational complexity of filtering an entire cost volume is $O(ML)$; thus, it is difficult to handle an extremely large label space.

One possible strategy for handling a large label space is to locally change the label space in order to reduce its size. Because the true label configuration is generally smooth in space (*e.g.*, disparities are smooth except for object boundaries), the label space required for performing CVF on a local region should be smaller than the entire label space. As an example, we present a colored true disparity map of *cones* (see Fig. 2.2) that is divided into local regions by regular rectangular grids. In

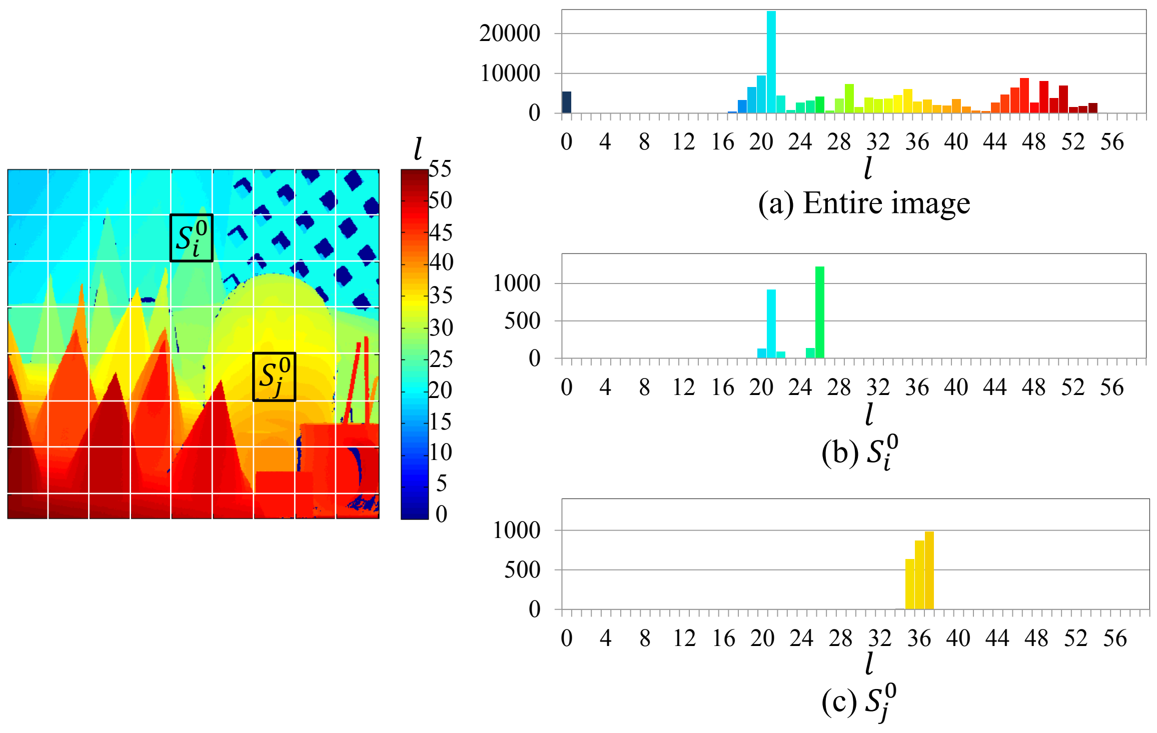


Figure 2.2: Colored true disparity map of *cones*, and a histogram of the true disparities l in the entire image and the ones in local regions S_i^0 and S_j^0. The disparities l are rounded off to integer values.

addition, we show a histogram of the true disparities l in the entire image and the ones in the local regions S_i^0 and S_j^0. We observe that the types of true labels in a local region are fewer than those in the entire label space.

However, the problem is of course that we do not know *a priori* which labels are important for each local region, and thus, the estimation of local label subsets is required [28].

2.3.2 Problem Statement

Here, we present a simple but efficient label subset estimation algorithm. Unlike Lu *et al.* [28], we do not rely on global optimization for estimating local label subsets; instead, we leverage the coarse-to-fine framework. An overview of the proposed method is shown in Fig. 2.3. Our algorithm mainly involves two steps (i) in-scale cost-volume filtering and (ii) across-scale label propagation. The latter is an essential feature of our approach, whereby a local label subset is estimated from the CVF output at a low-resolution. Because the computational cost of CVF for a

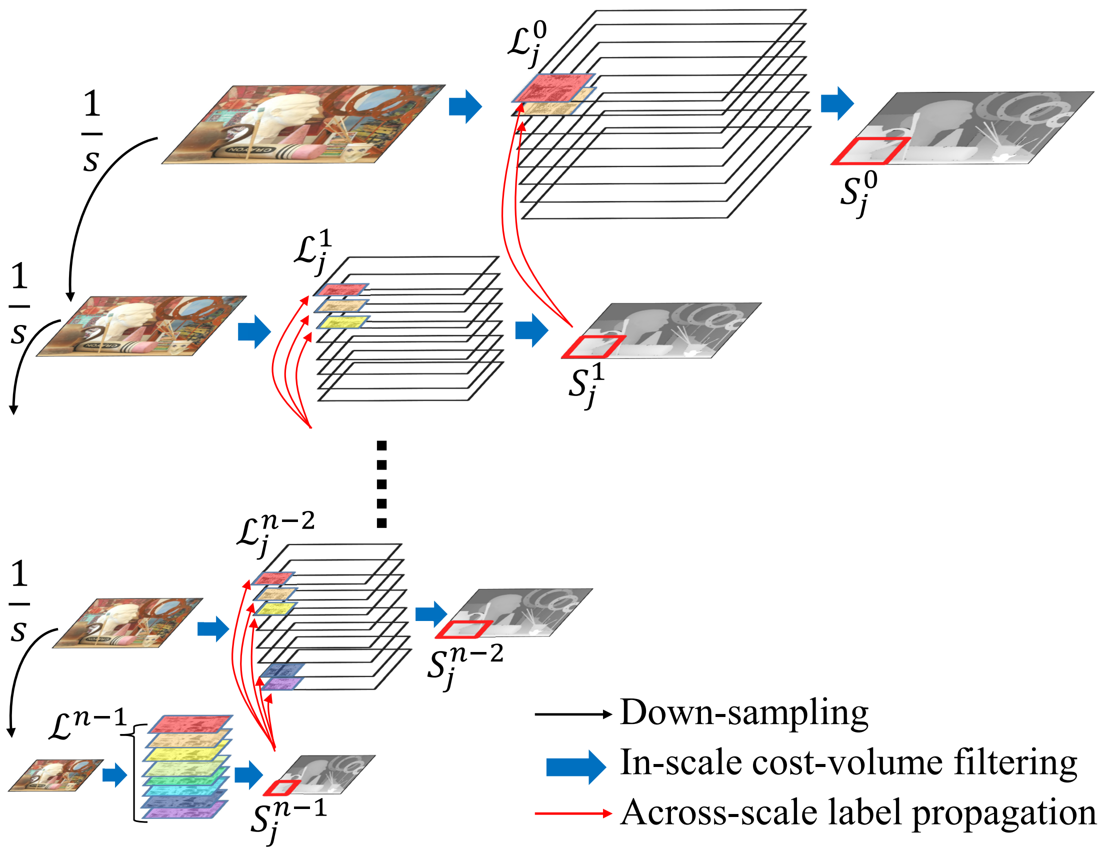


Figure 2.3: Framework of proposed method.

low-resolution image is negligibly small, we perform CVF using a large label space with a low-resolution and truncate unimportant labels using the output.

Let $I^k (k = 0, \ldots, n-1)$ denote a cascade of images of decreasing resolution ranging from the original scale (*i.e.*, $I^{k+1} = I^k_{\downarrow s}$, where $\downarrow$ is a down-scaling operator with a scale factor $s \in (1, \infty)$)[1], and let $\mathcal{L}^k$ denote the set of all possible labels at the k-th scale. Then, we divide I^0 $(= I)$ into m non-overlapping local regions S_j^0 and partition $I^k (k \geq 1)$ into local regions $S_j^k (j = 0, \ldots, m-1)$ such that $S_j^{k+1} = S^k_{j\downarrow s}$. In addition, we represent a label subset for S_j^k as $\mathcal{L}_j^k$ and its size as L_j^k. The total computational complexity of CVF from the lowest scale ($k = n-1$) to the original scale ($k = 0$) is expressed as

$$O(\sum_{k=0}^{n-1} \sum_{j=0}^{m-1} M_j^k L_j^k), \tag{2.3}$$

[1]We used the "buildPyramid" function in OpenCV to down-sample images.

where M_j^k is the number of pixels in S_j^k (*i.e.*, $M_j^k = s^{-2k}M_j^0$). Therefore, our objective is to estimate compact label subsets $\mathcal{L}_j^k$ such that $\sum_{k=0}^{n-1}\sum_{j=0}^{m-1} M_j^k L_j^k \ll ML$ while maintaining the accuracy of CVF. The optimal m and n values will be discussed in Sec. 2.4.

2.3.3 Across-Scale Label Propagation

In this section, we present an algorithm for estimating compact label subsets ($\mathcal{L}_j^k$) that sufficiently reduce the computational cost in Eq. (2.3) without truncating important labels. Our algorithm begins with the coarsest scale (*i.e.*, $k = n-1$). At this scale, we set $\forall j\ \mathcal{L}_j^{n-1} \leftarrow \mathcal{L}^{n-1}$ and simply perform CVF [1] to acquire the filtered cost volume C^{n-1} at the $(n-1)$-th scale. Note that although we use a complete label set, the computational complexity of CVF at this scale is $O(s^{-2(n-1)}ML)$, which is generally negligible (*e.g.*, if we set s to 2 and n to 4, $O(s^{-2(n-1)}ML) \approx O(10^{-2} \times ML)$). Then, we initialize the label subset at the higher resolution ($\tilde{\mathcal{L}}_j^{n-2}$) by merging labels having the smallest cost values in C^{n-1} at the corresponding local regions S_j^{n-1}. Strictly speaking, the initialization is expressed as

$$\tilde{\mathcal{L}}_j^{n-2} = \bigcup_{i \in S_j^{n-1}} f(l_i), \ \ l_i = \arg\min_l C^{n-1}(i,l), \tag{2.4}$$

where $C^{n-1}(p,q)$ is the value of the cost volume at the $(n-1)$-th scale with regard to the position p and the label q, and f is a *projection function* that normalizes the label space if required. In general, the projection function is represented as a constant scale factor giving $f = s$. For instance, a disparity l at the k-th scale corresponds to sl at the $(k-1)$-th scale in the stereo matching problem[2]. The initialization method based on across-scale label propagation is motivated by a reasonable observation that true labels at the same coordinates in images of different scales are highly correlated; in particular, they are very close when the difference in scales is small.

Although the initial estimation $\tilde{\mathcal{L}}_j^{n-2}$ is a good approximation of the optimal label subset $\mathcal{L}_j^{n-2}$, the problem is that $\tilde{\mathcal{L}}_j^{n-2}$ does not consist of labels that are not included in $f(\mathcal{L}_j^{n-1})$, which results in aliasing artifacts when the intermediate labels of $\mathcal{L}_j^{n-1}$ should be included in $\mathcal{L}_j^{n-2}$ (artifacts become more problematic as the scale difference increases). In addition, the filtered cost volume C^{n-1} often contains numerical errors due to occlusion boundaries or insufficient energy modeling. We

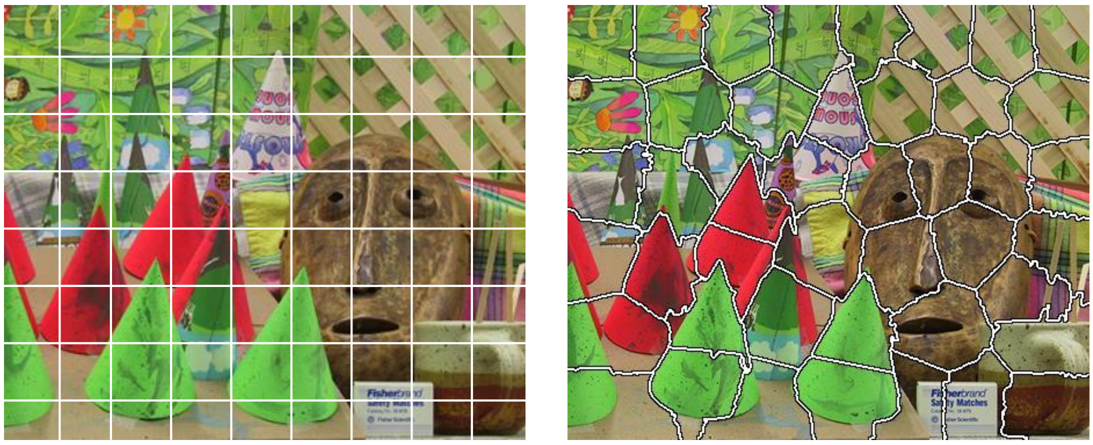
[2]In some cases, the label space does not need to be normalized because the scale of a label does not depend on the image coordinate space. Examples include depth-map up-sampling [27] and image segmentation [26].

adopt two strategies to overcome these difficulties. First, we down-sample images with a relatively small scale factor (*e.g.*, $s \leq 2$), such that the scale difference between two layers becomes sufficiently small. Second, we complete the initial label subset by adding the supporting labels within $\pm s/2$. Note that our algorithm supports floating labels (*e.g.*, sub-pixel disparity values). For instance, if the scale factor is 2 and the disparity unit is 0.5, the initial estimation $\tilde{\mathcal{L}}_j^{n-2} = \{2,5\}$ is extended as $\mathcal{L}_j^{n-2} = \{1,1.5,2,2.5,3,4,4.5,5,5.5,6\}$. Once a compact label subset $\mathcal{L}_j^{n-2}$ has been constructed, the target layer is shifted to the higher scale (*i.e.*, $k \leftarrow n-2$). Similarly to the case of the coarsest scale, CVF is performed on S_j^{n-2} with regard to $\mathcal{L}_j^{n-2}$. Cost-volume filtering with respect to $\mathcal{L}_j^k$ and the estimation of $\mathcal{L}_j^{k-1}$ from C^k are iterated $n-1$ times until $\mathcal{L}_j^0$ is obtained. Then, the final label at each pixel in S_j^0 is selected by a simple WTA strategy, as in the case of CVF [1].

For the entirety of the coarse-to-fine process, we fix the radius of the edge-preserving filter to smooth the cost-volumes; in other words, the radius is not changed when the target scale is shifted to a higher scale. Therefore, the lower the scale, the more strongly is the cost-volume smoothed. Thus, incorrect labels that accidentally have low costs are truncated during our coarse-to-fine process. In the original CVF [1], especially near object boundaries, the low costs of such incorrect labels are sometimes not sufficiently smoothed, and these incorrect labels are selected by the WTA strategy. Therefore, in such cases, our coarse-to-fine strategy sometimes increases the accuracy of the output at the finest scale, as compared to the original CVF. The results will be presented in Sec. 2.4.1.

It is possible to generate S_j^0 in various ways, *e.g.*, using regular rectangular grids or superpixels [62], as shown in Fig. 2.4. The former is simple and suitable for edge-preserving filters using integral images, *e.g.*, a guided filter [39]. In contrast, when S_j^0 are generated by superpixels, some additional computational time is required because we need to apply the edge-preserving filter to the bounding-box containing each region, as in the case of [28]. However, in such cases, it is easier to estimate the local label subsets because the local regions based on the superpixels are less likely to cross object boundaries than regular grids. For these reasons, we use both regular rectangular grids and superpixels for generating local regions S_j^0, as described in Sec. 2.4.1.

The proposed algorithm is summarized as Algorithm 1.



(a) Rectangular regular grids (b) SLIC super-pixels [62]

Figure 2.4: Examples of local regions S_j^0.

2.4 Results

In this chapter, we demonstrate the validity of our coarse-to-fine approach for CVF by applying it to stereo matching and optical flow estimation. Important to note that our technical contribution is the computational efficiency as compared to the original CVF algorithm, not the accuracy improvement. Besides, the application of CVF is not limited to stereo matching and optical flow estimation.

2.4.1 Middlebury Stereo

Experiments were conducted to evaluate the performance of our proposed method using the Middlebury stereo matching benchmark [31]. In stereo matching, the label l corresponds to the integer disparity between a pixel i in the target image I and its equivalent in the reference image I' shifted by the disparity. In the same manner, the cost function is selected as [1]:

$$C(i,l) = (1-\alpha)\min[\|I'_{i+l} - I_i\|, \tau_1] + \alpha \min[\|\nabla_x I'_{i+l} - \nabla_x I_i\|, \tau_2], \tag{2.5}$$

where ∇_x is the gradient in the x direction. The model parameters α, τ_1, and τ_2 are set to 0.89, 0.0027, and 0.0078, respectively[3]. We divide eight test image pairs of the Middlebury stereo datasets [31] into two categories according to their size: *small* and *large*. The *small* category includes *cones* (450×375), *teddy* (450×375),

[3]Parameters have been provided by the authors of [1]

Algorithm 1: The proposed coarse-to-fine strategy

```
1 INPUT: image pyramid $I^k$ ($k = 0, \cdots, n-1$)
2 OUTPUT: labeling at the original scale ($k = 0$).

    set $k \leftarrow n-1$ and $\mathcal{L}_*^{n-1} \leftarrow \mathcal{L}^{n-1}$  # start from the coarsest scale.
    while scales $k \geq 0$ do
      divide the image $I^k$ into $m$ local regions $S_j^k$
      for regions $j = 0$ to $m-1$ do
        for all $i \in S_j^k$ and $l \in \mathcal{L}_j^k$ do
          compute the cost value $C^k(i,l)$.
        end for
        for all $i \in S_j^k$ and $l \in \mathcal{L}_j^k$ do
          $C^k(i,l) \leftarrow \sum_{i' \in \omega_i} W_{ii'} C^k(i',l)$  # filter the cost volume.
        end for
        $\tilde{\mathcal{L}}_j^{k-1} = \bigcup_{i \in S_j^k} f(l_i),\ l_i = \arg\min_{l \in \mathcal{L}_j^k} C^k(i,l)$  # across scale label propagation.
        $\mathcal{L}_j^{k-1} = \tilde{\mathcal{L}}_j^{k-1} + \textit{supporting labels}$
      end for
      $k \leftarrow k-1$  # move to the next higher scale.
    end while

    # at the original scale ($k = 0$)
    for regions $j = 0$ to $m-1$ do
      for all $i \in S_j^k$ do
        $l_i = \arg\min_{l \in \mathcal{L}_j^0} C^0(i,l)$  # get the final labeling.
      end for
    end for
```

tsukuba (384×288), and *venus* (434×383). Further, the *large* category includes *art* (1390×1110), *books* (1390×1110), *moebius* (1390×1110), and *reindeer* (1342×1110). The label space size L is set to 60 for *small* datasets and 240 for *large* datasets. All the experiments were performed using an Intel Core i7-2600 (3.4GHz, single thread) machine with 16 GB of RAM, and they were implemented in C++. As in the original study of CVF [1], we use the guided filter [39] to smooth the cost volume (the radius of the filter is fixed at 9).

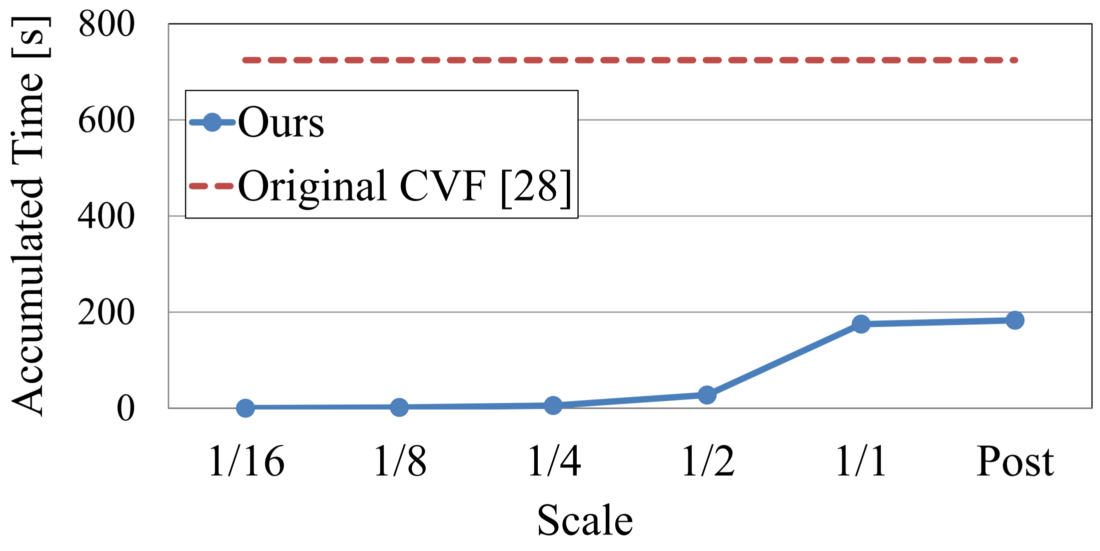


Figure 2.5: Evaluation of the computational time. The results of eight Middlebury stereo datasets are averaged. *Post* indicates the total computational time after weighted median filtering for the final disparity-map refinement.

Evaluation of Label Selection

We begin by evaluating the efficiency of our coarse-to-fine strategy, as compared to that of CVF [1]. Here, we apply our method ($n = 5, s = 2, m = 30$) and CVF [1] to both *small* and *large* datasets; the results are averaged as shown in Fig. 2.5. We observe that overall, our coarse-to-fine strategy takes much less time than CVF [1]. As expected, the computational time for small scales (*e.g.*, $1/16, 1/8, 1/4\times$) is negligible as compared to that for the original resolution ($1/1\times$).

Further, we present the average size of local label subsets estimated in our coarse-to-fine process, as compared to the size of the entire label space (see Fig. 2.6). We observe that although the latter increases exponentially with the scale, there is no significant increase in the former, which is much smaller than the latter in the original scale. As a result, our method is much more efficient than CVF [1].

However, an important question arises, which directly addresses the accuracy of the final label selection: "Are the estimated label subsets of the original scale really correct?" To answer this question, we define two metrics for measuring the correctness of the final label subset:

$$P(j) = \frac{|\mathcal{L}_j^0 \cap \mathcal{L}_j|}{|\mathcal{L}_j^0|}, \quad R(j) = \frac{|\mathcal{L}_j^0 \cap \mathcal{L}_j|}{|\mathcal{L}_j|}, \tag{2.6}$$

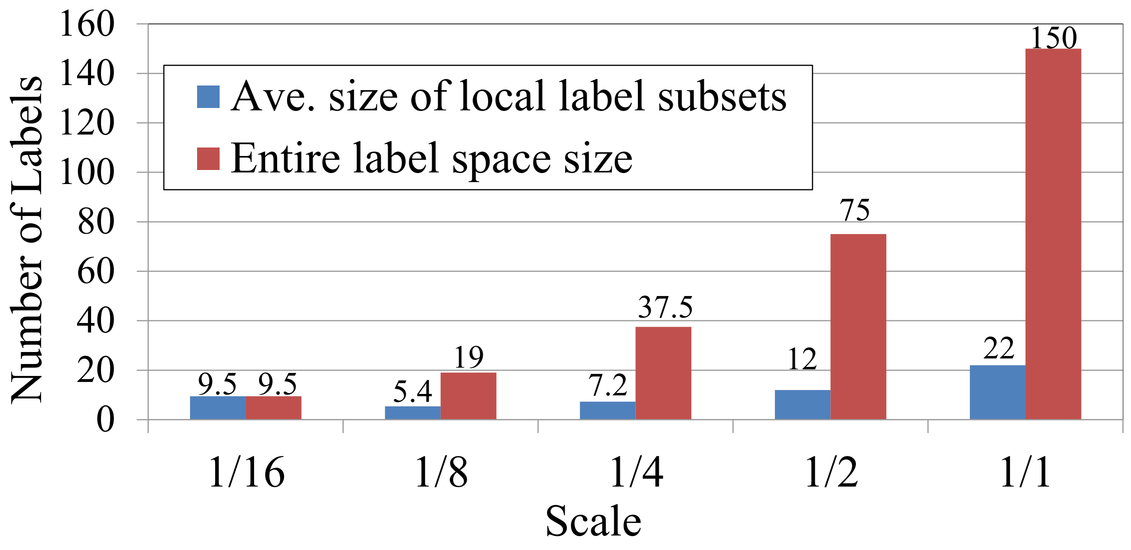


Figure 2.6: Evaluation of label set size at each scale. The results of eight Middlebury stereo datasets are averaged.

where $\mathcal{L}_j$ is the subset of ground truth labels at the original scale (*i.e.*, a collection of ground truth disparity values that emerge in the j-th region), and we recall that $\mathcal{L}_j^0$ is the subset of estimated labels at the original scale. These two metrics evaluate the estimated label subset in two different aspects: $P(j) \in [0,1]$ measures the *precision* of $\mathcal{L}_j^0$, which implies how correctly unimportant labels are removed, and $R(j) \in [0,1]$ measures the *recall* of $\mathcal{L}_j^0$, which implies how correctly important labels are maintained. Note that the ideal situation of course occurs when $\forall j\ \mathcal{L}_j^0 = \mathcal{L}_j$. For $\mathcal{L}_j$, we used the ground truth of the disparity maps precomposed in the Middlebury stereo datasets [31].

Using these metrics, we evaluate our method with a varying scale factor s and number of layers n using only *small* datasets, as shown in Table 2.1 and Table 2.2. Here, the results are averaged over all the datasets in this category.

Table 2.1 shows the evaluation of the label subset estimation with a fixed lowest scale and varying scale differences. We observe that when the scale difference between two layers is small (down-scale factor $s = 2$), our algorithm successfully maintains around 90% of ground truth labels and truncates more than 50% of unnecessary labels, on average, whereas the original label subset contains 90% of unnecessary labels. When the scale difference is large ($s = 16$), our method maintains more than 70% of unnecessary labels, on average. Therefore, we select a small down-scale factor ($s = 2$) in the following.

Table 2.1: Evaluation of label subset estimation with fixed lowest scale and varying scale differences.

Transition of scale	Ave. Precision	Ave. Recall
1/16→1/8→1/4→1/2→1/1 (s=2, n=5)	0.58	0.89
1/16→1/4→1/1 (s=4, n=3)	0.48	0.89
1/16→1/1 (s=16, n=2)	0.23	0.93
1/1 (CVF[1])	0.13	1.00

Table 2.2: Evaluation of label subset estimation with fixed scale difference and varying number of layers.

Transition of scale	Ave. Precision	Ave. Recall
1/16→1/8→1/4→1/2→1/1 (s=2, n=5)	0.58	0.89
1/8→1/4→1/2→1/1 (s=2, n=4)	0.58	0.91
1/4→1/2→1/1 (s=2, n=3)	0.57	0.90
1/2→1/1 (s=2, n=2)	0.49	0.92
1/1 (CVF[1])	0.13	1.00

Next, Table 2.2 shows the case of a fixed scale difference and varying number of layers. We observe that when the number of layers n is set to 4, the performance of our method is optimal, considering both the precision and the recall. In such cases, our algorithm maintains more than 90% of ground truth labels and truncates more than 50% of unnecessary labels, on average. Further, we observe that when the number of layers is small ($n = 2$), the precision is low (less than 50%).

In summary, our observations are in good agreement with our experiments: the improvement in precision is generally limited when the number of layers is too small or the scale difference between two layers is too large. When setting the appropriate number of layers ($n = 4$) and scale difference ($s = 2$), our method successfully maintains important labels and removes unimportant labels using the coarse-to-fine strategy. Therefore, in the experiments described below, we fix n to 4 and s to 2.

Comparison with PatchMatch Filter

Here, we evaluate the performance of our method by comparing it with PatchMatch filter (PMF) [28] using both *small* and *large* datasets of the Middlebury stereo benchmark [31]. We did not compare the performance of our method with other algorithms dedicated for stereo matching because the stereo matching is merely one of the applications of our method for general multi-labeling problems. For a fair comparison, our method and PMF are performed using the same superpixels clustered by SLIC [62], the cost function, and post-processing based on left-right cross-checking and median-filtering (for further details, see [1])[4]. Further, we evaluate the performance of our method on the basis of a regular image grid with varying block size. Note that the number of local regions is inversely proportional to the block size. The results are presented in Table 2.3 and Table 2.4. Here, the percentage disparity errors (threshold is set as one for *small* datasets, and one and four for *large* datasets) are averaged over all images within the same category. We observe that although our method, PMF [28], and CVF [1] provide nearly the same level of accuracy, our method is the most efficient method for both categories. In particular, for *large* datasets, our method achieves 6× faster performance than CVF [1], while providing a similar (or higher level) accuracy. We also observe that our method outperforms PMF when the number of local regions is large (*e.g.*, superpixels with $K = 200, 500$) or when the image is divided into local regions on the basis of a simple image grid. This is because unlike the case of PMF [28], we do not consider any spatial smoothness of label subsets within the scale; instead, we consider the cross-scale smoothness of the local label subset, which is independent of the spatial coherence.

The estimated disparity maps of the *teddy* and *art* datasets are shown in Fig. 2.7 and Fig. 2.8, respectively. These are compared with those obtained by PMF [28] and CVF [1]. We observe that our method succeeds in estimating smoother and more reasonable disparity maps than CVF and PMF, especially in the case of the *teddy* dataset. Near object boundaries, CVF and PMF assign many incorrect labels, whereas our method does not. The reason is that our coarse-to-fine strategy successfully truncates incorrect labels that accidentally have low costs, as mentioned in Sec. 2.3.3.

Finally, we present the estimated disparity maps of *small* and *large* datasets in Fig. 2.9 and Fig. 2.10, respectively.

2.4.2 KITTI Stereo 2015

We also conducted experiments on the KITTI stereo 2015 benchmark, which is more difficult than the Middlebury stereo dataset in Sec. 2.4.1 in terms of disparity range and image resolution. All the parameters and the cost function are exactly same as those in Sec. 2.4.1. We used 200 training images with ground truth disparity maps. The resolutions of all images are 1241×376. In this dataset, we did not

[4]Post-processing is performed on our method only in the original resolution.

Table 2.3: Comparison with PMF using *small* datasets.

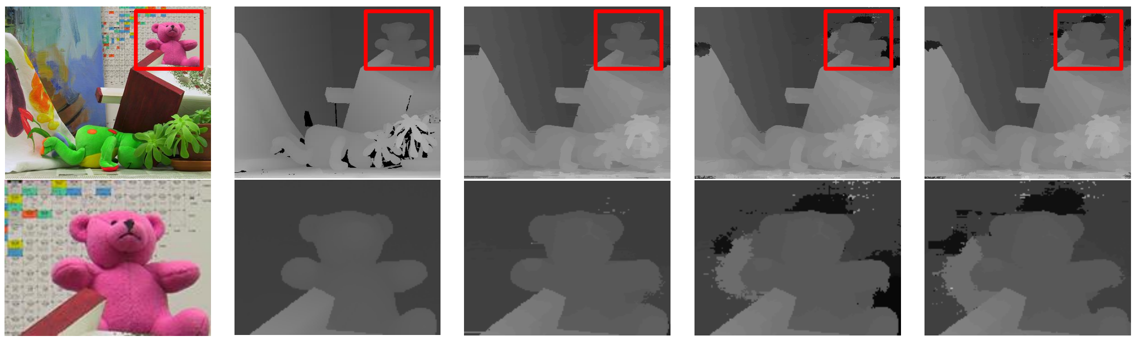
Method	Time[s]	Err. %: thre. = 1.0		
		nonocc	all	disc
CVF[1]	35.38	3.30	6.17	9.74
PMF[28] (K=50)	23.43	3.19	**5.97**	9.56
PMF[28] (K=100)	28.97	3.23	6.03	9.32
PMF[28] (K=200)	43.14	3.27	6.04	9.36
PMF[28] (K=500)	73.21	3.30	6.08	**9.31**
Ours (Superpixels, K=50)	15.98	3.51	6.31	10.8
Ours (Superpixels, K=100)	16.56	3.46	6.23	10.7
Ours (Superpixels, K=200)	18.48	3.69	6.48	11.3
Ours (Superpixels, K=500)	23.55	4.15	7.03	12.2
Ours (Grid, 150x150)	17.67	**3.11**	5.98	10.1
Ours (Grid, 75x75)	**12.47**	3.22	6.02	10.4

Table 2.4: Comparison with PMF using *large* datasets.

Method	Time[s]	Err. % (all)	
		Err. thre.=1	Err. thre.=4
CVF[1]	1413	21.5	14.8
PMF[28] (K=50)	266	22.7	15.6
PMF[28] (K=100)	322	22.5	15.5
PMF[28] (K=200)	484	22.5	15.6
PMF[28] (K=500)	802	23.3	16.2
Ours (Superpixels, K=50)	269	22.5	15.3
Ours (Superpixels, K=100)	249	23.0	15.7
Ours (Superpixels, K=200)	262	23.6	16.0
Ours (Superpixels, K=500)	304	24.5	17.2
Ours (Grid, 600x600)	1186	21.1	14.4
Ours (Grid, 300x300)	796	**20.5**	**13.4**
Ours (Grid, 150x150)	371	21.6	14.4
Ours (Grid, 75x75)	**246**	25.2	17.6

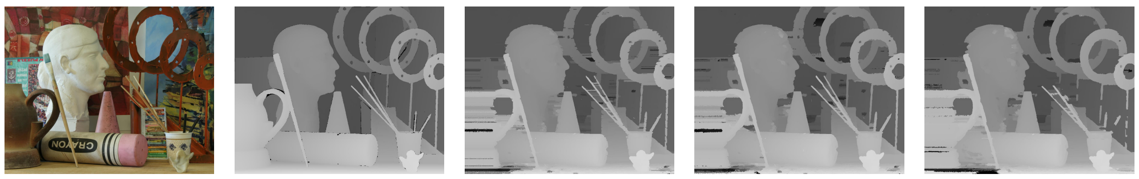
perform the post-processing (weighted median filtering) in order to compare the pure performance of each method. The search range of disparity was set to 256 in all methods.

We show the comparison of computational time and accuracy in Table 2.5. Following the official evaluation rule of KITTI stereo 2015, we computed the percentage of error pixels. We regarded the pixel to be correctly estimated if the disparity error is less than 3 pixel or 5% at each pixel. The results of 200 images



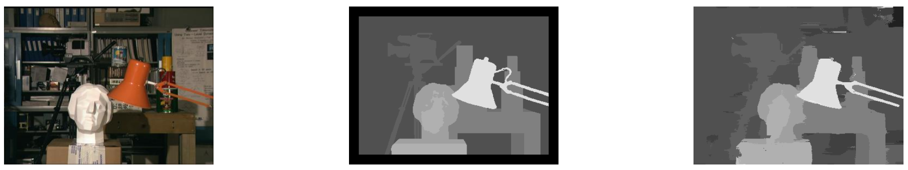
(a) Left image (b) Ground truth (c) Ours (Grid, 150x150) (d) CVF [1] (e) PMF (K=500) [28]

Figure 2.7: Qualitative comparison with regard to estimated disparity maps of *Teddy* dataset.

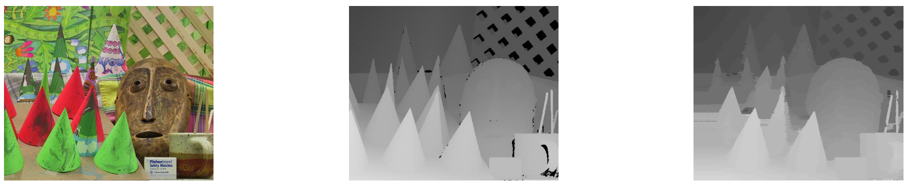


(a) Left image (b) Ground truth (c) Ours (Grid, 300x300) (d) CVF [1] (e) PMF (K=500) [28]

Figure 2.8: Qualitative comparison with regard to estimated disparity maps of *Art* dataset.

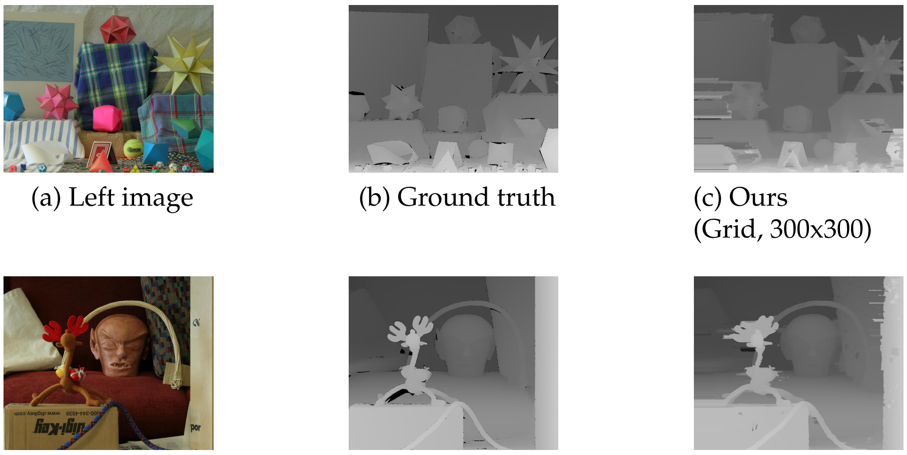


(a) Left image (b) Ground truth (c) Ours (Grid, 150x150)



(d) Left image (e) Ground truth (f) Ours (Grid, 150x150)

Figure 2.9: Qualitative results on the *small* datasets.



(a) Left image (b) Ground truth (c) Ours (Grid, 300x300)

(d) Left image (e) Ground truth (f) Ours (Grid, 300x300)

Figure 2.10: Qualitative results on the *large* datasets.

Table 2.5: Comparison of computational time and accuracy using KITTI stereo 2015 datasets.

Method	Time[s]	Err. %	
		Err. Nonocc.	Err. All
CVF[1]	244	32.0	33.2
PMF[28] (K=50)	**28.5**	35.9	36.6
PMF[28] (K=100)	30.7	35.5	36.2
PMF[28] (K=200)	37.7	35.1	35.8
PMF[28] (K=500)	53.5	35.2	35.9
Ours (Superpixels, K=50)	58.2	24.2	25.5
Ours (Superpixels, K=100)	50.9	23.5	24.8
Ours (Superpixels, K=200)	45.2	22.6	23.8
Ours (Superpixels, K=500)	44.5	**22.3**	**23.6**
Ours (Grid, 300x300)	89.4	26.3	27.5
Ours (Grid, 150x150)	67.3	27.2	28.4
Ours (Grid, 75x75)	39.3	24.3	25.6

are averaged in Table 2.5. We observe that the accuracy of PMF [28] ($K = 50$) is worse than the original CVF [1] although PMF [28] is the fastest. In this dataset, our method with superpixel division is 5 or 6 times faster than the original CVF [1], and our method ($K = 500$) is much more accurate in both non-occluded and all regions. We observe that the patchmatch search did not work effectively in this dataset as opposed to our coarse-to-fine strategy. However, our method with regular grid

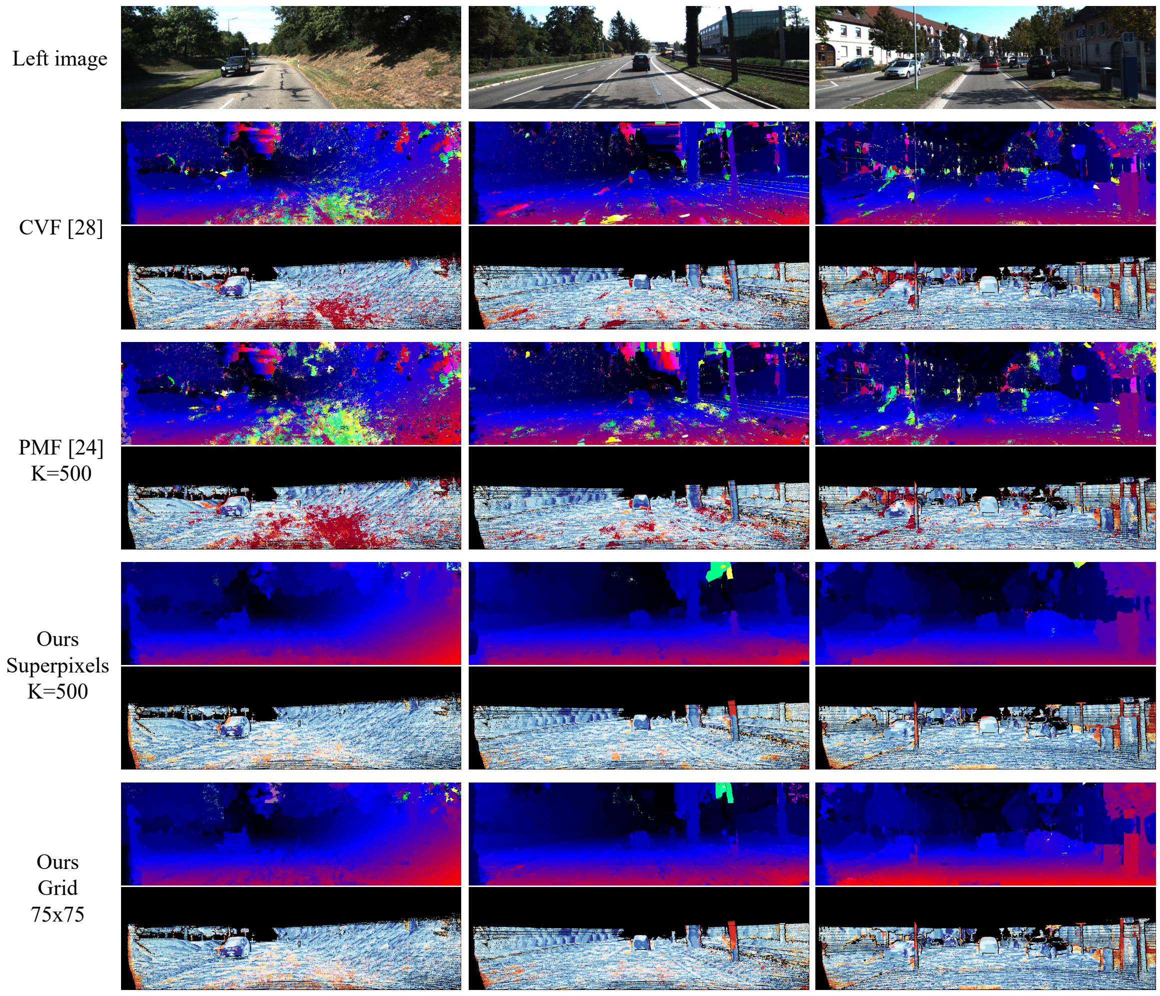


Figure 2.11: Qualitative comparison with regard to estimated disparity maps of KITTI stereo 2015 dataset. Disparity maps (upper rows) and error maps (lower rows).

division is worse than that with superpixel division in terms of both efficiency and accuracy.

The estimated disparity maps are shown in Fig. 2.11. Compared with CVF [1] and PMF [28], our method achieved smoother and more reasonable results by truncating unnecessary labels with the coarse-to-fine strategy. We observe that our method is better especially in less or repeated texture regions (*e.g.*, on the road and in the sky).

2.4.3 Middlebury Optical Flow

We also carried out experiments using the Middlebury optical flow benchmark [32]. In optical flow estimation, the label l corresponds to the 2-D motion vector (u,v)

between the target image and the reference image. Further, u and v denote the displacements along the x and y directions, respectively, and they take floating values. We use the same cost function as that in the original CVF [1]:

$$C(i,l) = (1-\alpha)\min[||I'_{i+l} - I_i||, \tau_1] + \alpha \min[||\nabla_x I'_{i+l} - \nabla_x I_i|| + ||\nabla_y I'_{i+l} - \nabla_y I_i||, \tau_2], \quad (2.7)$$

where ∇_x and ∇_y are the gradients in x and y direction, respectively. The parameters are set to the same values as those in the experiments for stereo matching; only τ_2 is changed to 0.0156 in the same manner as in [1]. In all the datasets, the search ranges of both u and v are set to the interval of -10 to 10 pixels. To achieve sub-pixel accuracy, the units are set to 0.25 pixel (*i.e.*, each space of u and v is $\{-10, -9.75, -9.5, \ldots, 0, \ldots, 9.5, 9.75, 10\}$). Therefore, the size of the entire label space is $81 \times 81 = 6561$.

The results are listed in Table 2.6. Here, the average angle error (AAE) and average end-point error (AEE) are used for evaluation. They are defined, respectively, as

$$AE = cos^{-1}\left(\frac{1.0 + u \times u_{GT} + v \times v_{GT}}{\sqrt{1.0 + u^2 + v^2}\sqrt{1.0 + u_{GT}^2 + v_{GT}^2}}\right), \quad (2.8)$$

$$EE = \sqrt{(u - u_{GT})^2 + (v - v_{GT})^2}, \quad (2.9)$$

where (u, v) is the estimated flow and (u_{GT}, v_{GT}) is the ground truth flow. We did not compare the performance of our method with other algorithms dedicated for optical flow estimation for the same reason as stereo matching. From Table 2.6, we observe that our method is superior to and much faster than the original CVF [1] in all cases. In particular, by using superpixel division ($K = 50$), our method achieves the most accurate results and much faster performance than CVF (10× or more). Further by using regular grid division, our method achieves a higher level of accuracy than CVF, and it is the most efficient.

The estimated flow maps of the Middlebury optical flow dataset are shown in Fig. 2.12. We observe that our method estimates the flow around boundaries more accurately than CVF. As in the case of our stereo matching results, this is because erroneous flow vectors, which yield minimum costs even though they are the wrong choices, are efficiently removed by our hierarchical approach.

Table 2.6: Comparison in optical flow estimation.

Method	RubberWhale			Grove2			Venus		
	time[s]	AAE	AEE	time[s]	AAE	AEE	time[s]	AAE	AEE
CVF[1]	6978	5.20	0.165	10792	3.65	0.258	5353	6.60	0.432
Ours (Superpixels, K=50)	537	**4.26**	**0.139**	768	**2.59**	**0.191**	493	4.21	0.318
Ours (Superpixels, K=100)	523	4.36	0.143	748	2.63	0.194	485	4.13	0.315
Ours (Superpixels, K=200)	588	4.45	0.145	786	2.63	0.194	524	4.24	0.317
Ours (Superpixels, K=500)	769	4.44	0.145	1009	2.67	0.197	752	**4.07**	**0.306**
Ours (Grid, 150x150)	574	4.32	0.140	739	2.62	0.193	902	4.18	0.312
Ours (Grid, 75x75)	**376**	4.29	**0.1399**	**555**	2.61	0.193	**448**	4.10	0.308

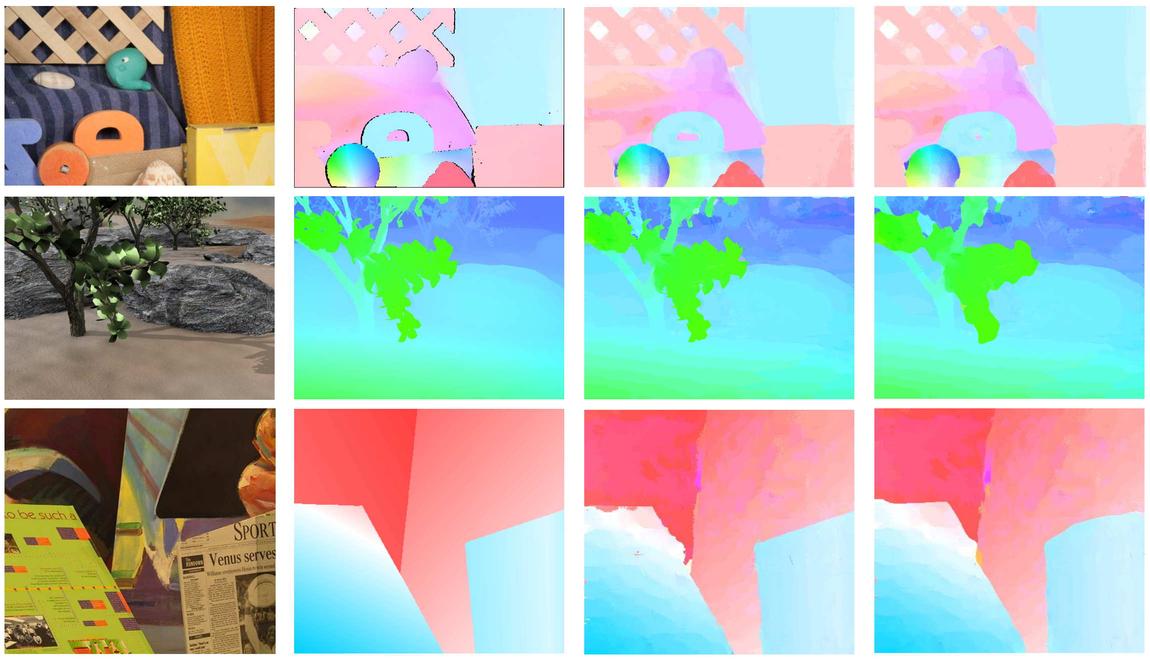

(a) First frame (b) Ground truth (c) Ours (Grid, 150x150) (d) CVF [1]

Figure 2.12: Qualitative comparison with regard to estimated flow maps of Middlebury optical flow dataset.

2.5 Conclusions

In this chapter, we proposed a coarse-to-fine strategy to reduce the large label space for efficient cost-volume filtering. The proposed method truncates redundant labels in each local region by using the labeling output of lower scales. Our method demonstrated higher efficiency than CVF while maintaining a comparable level of accuracy in stereo matching and optical flow estimation. Compared with PMF, our method showed comparable performance. Although PMF estimates compact label sets to reduce the computational cost by complex patchmatch search, our method does by simple coarse-to-fine strategy. Therefore, our method is yet another approach to optimize the label sets for efficient cost-volume filtering, which is much easier to implement than PMF.

In future work, as the performance of our method depends on the shape and number of local regions, we intend to explore the optimal division of local regions. In addition, we plan to investigate the GPU implementation of our method for real-time applications.

Chapter 3

Neighbor-Aware Fast Optimization for Special MRF

3.1 Introduction

This chapter proposes a scheme for seam carving (finding seam surfaces) in three-dimensional (3D) cost volume that features low computation time and memory consumption (Fig. 3.1). (Hereafter, we call seam carving for 3D cost volume as "volume seam carving.") Volume seam carving has been applied to various image processing tasks, such as video retargeting [10], video summarization [11] [12], and tone mapping [13]. The volume seam carving procedure is as follows: Create a cost volume, find a seam surface that is less affected if removed, and then remove that seam surface. A seam surface is a two-dimensional (2D) manifold in the cost volume and must be monotonic and connective.

To achieve the volume seam carving outlined above, to date, users can only use the graph-cut algorithm [10] because more efficient algorithms such as the dynamic-programming (DP) algorithm used in still image seam carving [63] are not considered for use in this area. Rubinstein et al. [10] stated that volume seam carving cannot be solved with DP and deemed graph cuts the only choice. However, it is well known that the graph-cut algorithm utilizes a tremendous amount of computational time and memory when the number of nodes and edges increases. Consequently, to make problems solvable, Rubinstein et al. [10] proposed a multi-resolution method and Chen and Sen [11] employed a video-chunking method. However, the significant computation time and high memory usage of the graph-cut algorithm remain critical problems in the era of high-resolution images/videos.

In this chapter, a multi-pass DP scheme for volume seam carving that realizes volume seam carving with low computation time and memory consumption is proposed. Here, we propose two options: a continuous mode and a discontinuous mode. The continuous method, which produces a connective seam surface, is presented in subsection 3.3.1. The discontinuous method, which produces a discontinuous seam surface, is presented in subsection 3.3.2. The discontinuous seam surface is connective in one direction and discontinuous in the other direction. We describe a monotonic manifold as a seam surface even if it is unconnected. In video retargeting, seams may be amenable to the discontinuous mode between frames, as discussed by Grundmann et al. [64], Chao et al. [65], and Yan et al. [66]. In contrast, in tone mapping, the connectivity of the seam surface is important. For these reasons, two different operational modes are presented here.

The proposed method is applied to video retargeting, tone mapping, and contrast enhancement to verify its efficacy. We present the results obtained, including the computation time and memory consumption, which, on average, decreased to approximately 1/90th and 1/8th, respectively, of those of graph cut-based solutions. We also verify via large-scale subjective evaluation experiments that the suboptimal solution can produce an image quality roughly equivalent to that of graph cuts in volume seam carving applications. It is important to note that this chapter presents a new and general optimization method (multi-pass DP) that is more efficient in terms of computation time and memory usage than the conventional graph cut-based methods for various types of volume seam carving-based applications, not for a specific application using the multi-pass DP.

The reminder of this chapter is organized as follows. Section 3.2 reviews related studies. Section 3.3 describes the details of the proposed multi-pass DP. Section 3.5 presents the applications and experimental results. Finally, Section 3.6 summarizes our findings and concludes the chapter.

3.2 Related Works

3.2.1 DP for 3D Volume

DP has been utilized in various image-processing applications other than seam carving. Let us take depth estimation as an example and compare it with seam carving. In depth estimation, a cost volume $C(d, x, y)$ (hereafter, for ease of understanding, the seam-carved axis is indicated as the first element) is created and a depth value d

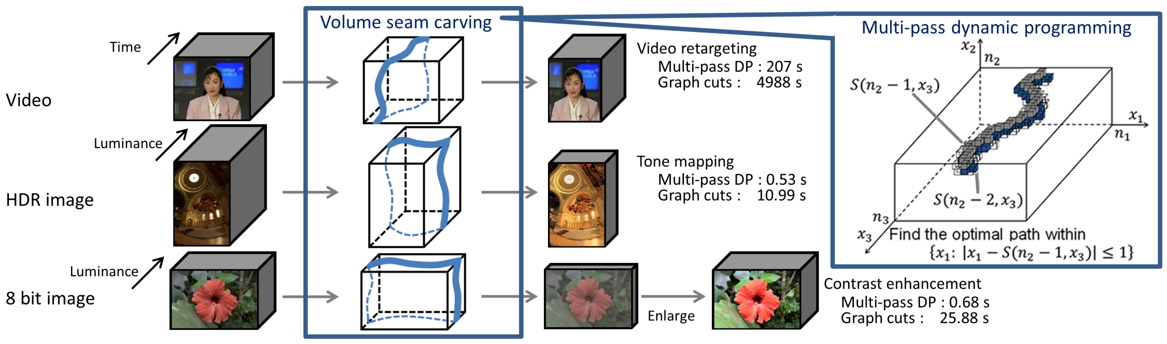


Figure 3.1: Multi-pass DP for volume seam carving proposed to reduce computation time and memory consumption. Three example applications of multi-pass dynamic programming are presented.

is estimated at each pixel so that the sum of the energy in all pixels is the minimum. This minimization is formulated in a Markov random field (MRF) framework to obtain a smooth depth map and can be solved using algorithms such as graph cuts and belief propagation [67]. DP is also used for this minimization when quick response is required. For example, scanline optimization is a well-known method in which energy minimization is solved in each line, considering the continuity only in the x direction [68]. In other words, the continuity in the y direction is not considered at all. Our multi-pass DP is different from this approach because 2D connectivity is guaranteed in its discontinuous mode and 3D connectivity is guaranteed in its continuous mode.

DP for 3D volume has been applied to depth estimation by Kim et al. [69], in which DP is performed in both the x and the y directions and discontinuous paths are penalized to maintain the smoothness of the depth map. Fukushima et al. [70] proposed a similar method to that of Kim et al. [69] for free viewpoint image rendering. Their method [70] performs DP in not only the forward x direction but also the backward direction and then sums the two accumulated costs. After that, DP is performed in the y direction similarly to [69]. Semi-global matching (SGM) [71] is a DP-based technique for 3D volume in which cost accumulation is performed in 8 or 16 directions and a final disparity map is obtained in a winner-takes-all manner. Fukushima et al. [72] applied an idea similar to SGM [71] for real-time free viewpoint image generation. Chen and Koltun [73] proposed a fast MRF optimization method based on DP for 3D volume, in which label assignments in even/odd lines in an image are optimally updated by DP fixing odd/even lines. However, unlike in our method, discontinuity is allowed in both the x and the y directions in these methods [69–73]. This is because, for example, discontinuous depth is reasonable at the boundary of two objects. On the other hand, continuity is sometimes required for seam carving, depending on the applications. The connectivity of a seam is assured by localizing the search range as done by Avidan and Shamir [63]. If a seam is discontinuous, the image quality may be deteriorated by removing the pixels on the seam.

In [12], a DP-based video seam carving technique called ribbon carving is presented. However, the seam obtained in [12] is a chunk of connected rigid rectangle running vertically or horizontally. The technique is suitable for surveillance video summarization, but not for general-purpose volume seam carving. Moreover, the seam can be found by 2D DP rather than by multi-pass DP. From this point of view, the ribbon carving technique [12] is essentially different from our multi-pass DP.

3.2.2 Video Retargeting

Video retargeting is a nonlinear image-resizing method for videos. Various methods have been presented for video retargeting, including cropping and warping-based methods [74] [75] [76] [77] [78] [79] [80]. Seam carving has also been applied to video retargeting, with each frame image retargeted sequentially by seam carving [64] [65] [66]. In contrast to still image retargeting, temporal correlation is taken into account for video retargeting.

Rubinstein et al. [10] introduced volume seam carving for video retargeting. In their scheme, the video sequence is treated as a space-time volume and seam carving is extended from a seam on a 2D image to a seam surface in a 3D volume. The cost volume $C(x, y, t)$ is created by aligning the energy functions of all frames, and a seam surface is derived by energy minimization with graph cuts. Furthermore, all frame images are reduced in size by removing the pixels on the seam surface. Forward energy has also been proposed as an alternative to conventional backward energy to maintain the image quality. The forward energy criterion looks ahead to the resulting image after removing the seam, which takes into account the inserted energy due to the new edges created by previously nonadjacent pixels that become neighbors once the seam is removed. Conversely, the backward energy criterion looks behind at the image before removing the seam. This method can reduce temporal flicker because the removed seams are continuous in the temporal direction. However, it is time- and memory-consuming because it uses slow graph cuts to obtain an optimal seam surface, as discussed in Sec. 3.1. This method can be significantly accelerated while maintaining the output quality by replacing graph cuts with our method.

Han et al. [81] proposed an algorithm that finds multiple seam surfaces simultaneously to improve the output quality. Although they use some techniques to accelerate the running time such as multi-resolution strategy and searching for a small number of seam surfaces iteratively, their method is still slow because they employ graph cuts to obtain a globally optimal solution [81].

Recently, Jain et al. [82] proposed a video retargeting method based on gaze tracking, in which a cropping window is determined by considering the gaze information. Katti et al. [83] utilized the gaze information to calculate the saliency costs for seam carving. Although these methods [82, 83] can obtain preferable results for users, they need to perform eye tracking to get the gaze information.

As discussed above, a variety of methods have been proposed for video retargeting. However, it is important to note that we are proposing a fast and memory-efficient optimization method for volume seam carving applications, not one specific to video retargeting.

3.2.3 Tone Mapping

Tone mapping is a technique for dynamic range compression that creates a low-dynamic–range (LDR) image from a high-dynamic-range (HDR) image. Many tone-mapping methods have already been introduced. They can be roughly classified into global and local methods. Global methods have a uniform conversion characteristic in the whole image. Specifically, plural pixels that have the same luminance value in an input HDR image have the same luminance value in the output LDR image. Similar luminance values in the HDR image are grouped and replaced with a luminance value in the LDR image. Therefore, textures whose variation in luminance is small easily vanish in the global methods. A typical global method is histogram equalization, which converts the luminance distribution so that the histogram of the LDR image has a high variation. A high-contrast LDR image is obtained by histogram equalization; however, the contrast is sometimes too exaggerated. Larson et al. [84] subsequently presented an improved histogram equalization method that constrains the contrast.

On the other hand, local methods do not have a uniform conversion characteristic in the entire image and volume seam carving-based tone mapping is categorized as a local method. Recently, various methods that preserve detailed local textures have been introduced. Fattal et al. [85] proposed a gradient domain method that manipulates the gradient field of the luminance image by attenuating the magnitudes of large gradients while preserving the fine details: the LDR image is obtained by solving a Poisson equation on the gradient field. Paris et al. [2] proposed local Laplacian filters (LLFs) that decompose an HDR luminance image into a Laplacian pyramid and compress the dynamic range in each level while keeping the local contrast. Aubry et al. [86] developed fast local Laplacian filters that improve LLFs in terms of computation time. Gu et al. [3] proposed local edge-preserving filters (LEPFs) that apply a multiscale decomposition to an HDR luminance image using an edge-preserving filter and compress the dynamic range in each level. He et al. [87] proposed a method based on a guided filter that decomposes an HDR image into a detail layer and a base layer, and adds an enhanced detail layer to the compressed base layer. Recently, Ma et al. [88] proposed an iterative tone-mapping method that optimizes the tone-mapped image quality index (TMQI) score. Because this method [88] requires a large number of iterations to converge, it is time consuming, compared with the filter-based approaches [2, 3].

The histogram equalization method can create higher-contrast LDR images than the above methods, but it may delete the fine details. Local methods can retain fine details, but are prone to reducing the contrast. A method that uses seam surfaces was proposed by Tsubaki and Iwauchi [13] to satisfy both high-contrast and fine-detail requirements by creating a cost volume $C(l,x,y)$ with luminance l and space axes. The cost volume is defined based on a local luminance histogram, and a graph is constructed similar to that by Rubinstein et al. [10]. Furthermore, a seam surface $S(x,y)$ is derived by graph cuts, and luminance values that are larger than the value in the seam surface are reduced by one. This process is iterated until the obtained image has the desired dynamic range. This method can obtain better or comparable results, compared with filter-based state-of-the-art methods [2, 3]. However, this method needs subsampling in 3D volume to make processing time and memory usage reasonable because it employs graph cuts to obtain an optimal seam surface, similar to [10]. This method can be significantly accelerated while maintaining the output quality by replacing graph cuts with our method.

3.2.4 Contrast Enhancement

Contrast enhancement is a classical issue for which many algorithms have already been proposed. Histogram equalization and histogram modification are widely utilized and can create a very high contrast image [89]. However, the contrast of the image obtained is sometimes too exaggerated; furthermore, fine details are not preserved. These problems for contrast enhancement are the same as for tone mapping. Enhancement methods based on the retinex theory, which estimates the reflectance and the illumination at each pixel, are also utilized widely [90] [91]. They retain the fine details better than histogram modification; however, halo artifacts sometimes arise.

Image contrast is deteriorated by various factors, such as blur and haze. Some detail enhancement and dehazing approaches can also enhance the contrast. Yun et al. [92] adopted Laplacian pyramid decomposition to enhance the contrast of a low-pass filtered image by histogram equalization. An LLF has also been applied to detail enhancements other than tone mapping [2]. A dehazing method that utilizes a local color-line model was more recently proposed by Fattal [93]. However, to the best of our knowledge, contrast enhancement using volume seam carving has not been reported to date. Furthermore, as stated above, contrast enhancement is similar to tone mapping. Thus, contrast enhancement using volume seam carving is proposed for the first time in this chapter.

3.3 Fast Volume Seam Carving with Multi-Pass DP

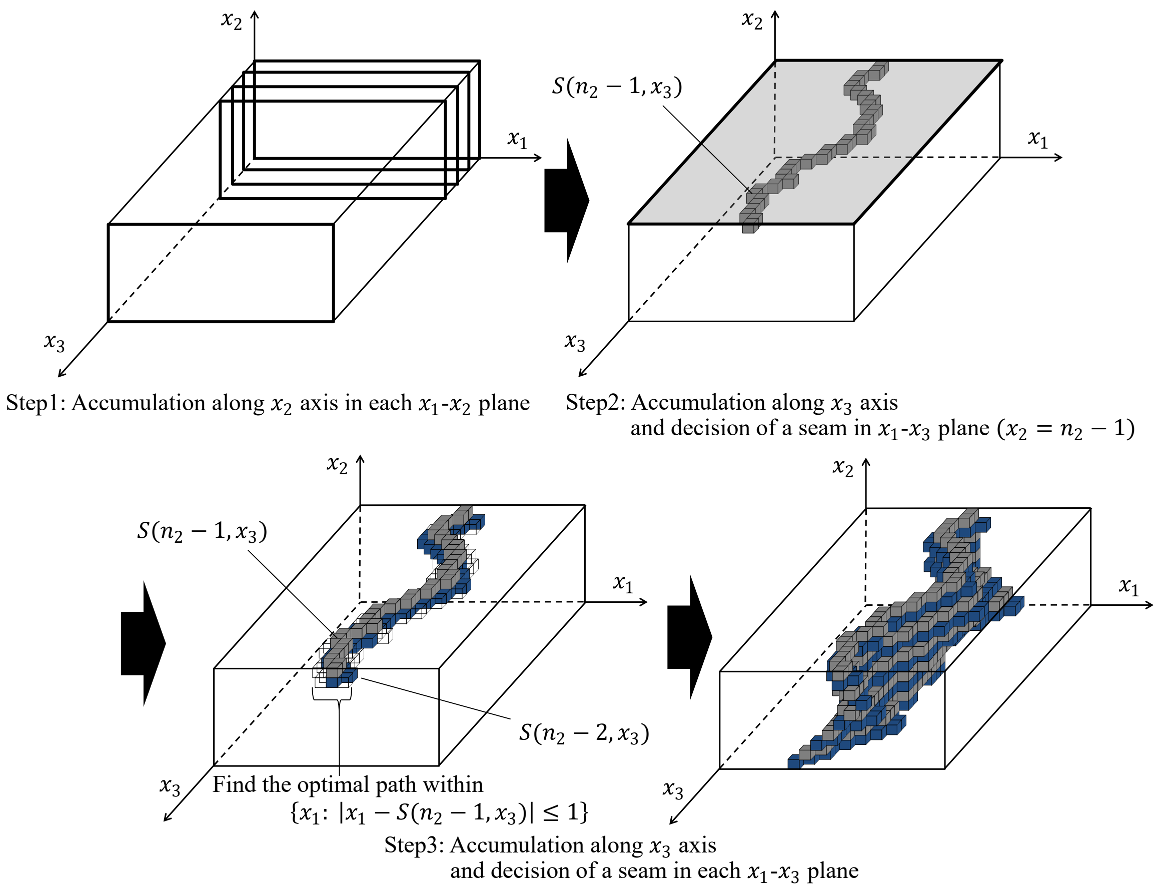
We propose a new approach based on DP for volume seam carving. A cost volume is defined as $C(x_1,x_2,x_3)$, where x_1,x_2,x_3 are integers in the ranges $0 \le x_1 < n_1$, $0 \le x_2 < n_2$, and $0 \le x_3 < n_3$, and n_1, n_2, and n_3 are the resolutions of the cost volume in the x_1,x_2,x_3 directions, respectively. A seam surface that crosses the x_1 axis is defined as $S(x_2,x_3)$. When the seam surface passes a coordinate (x_1,x_2,x_3), we describe it as $S(x_2,x_3)=x_1$. $(x_1,x_2,x_3)=(x,y,t)$ for video retargeting, which reduces the x resolution at each frame t. (n_1,n_2) is the image size and n_3 is the number of frames. The value of a cost volume $C(x_1,x_2,x_3)$ at each coordinate is obtained in the same manner as with conventional methods. For example, $C(x,y,t)=|\frac{\partial}{\partial x}I(x,y,t)|+|\frac{\partial}{\partial y}I(x,y,t)|$, which is defined as the backward energy for video retargeting [10] ($I(x,y,t)$ is the pixel value at the location (x,y) in the t-th frame).

We present two methods: a continuous method and a discontinuous method. These two methods are proposed because different applications require different characteristics in the seam surface, as discussed in Sec. 3.1.

3.3.1 Continuous Method to Obtain a Connective Seam Surface

Step 1: Accumulation along the x_2 axis

In this step, cost values are accumulated in each x_1-x_2 plane along the x_2 axis, from $x_2=0$ to n_2-1, with minimization just like in the original seam carving for image



x_2
x_1
x_3
Step1: Accumulation along x_2 axis in each x_1-x_2 plane
$S(n_2-1, x_3)$
Step2: Accumulation along x_3 axis and decision of a seam in x_1-x_3 plane $(x_2 = n_2 - 1)$
$S(n_2-1, x_3)$
$S(n_2-2, x_3)$
Find the optimal path within $\{x_1\colon |x_1 - S(n_2-1, x_3)| \leq 1\}$
Step3: Accumulation along x_3 axis and decision of a seam in each x_1-x_3 plane

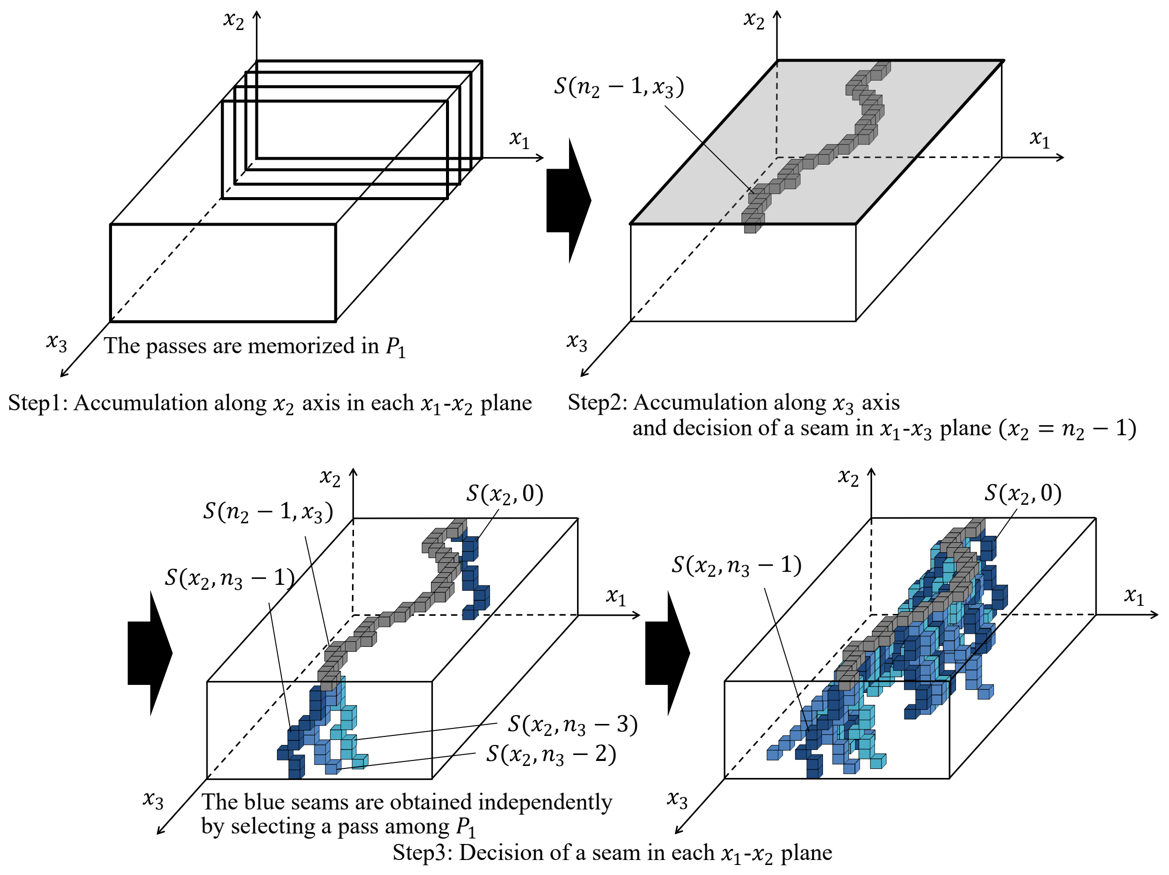
Figure 3.2: The continuous DP process. The seam surface is connected in both the x_2 and the x_3 directions.

retargeting [63]. An accumulated cost function A_1 is obtained by

$$\begin{cases} A_1(x_1,x_2,x_3) = C(x_1,x_2,x_3) + \min\limits_{j\in\{-1,0,1\}} A_1(x_1+j, x_2-1, x_3), & (x_2>0) \\ A_1(x_1,0,x_3) = C(x_1,0,x_3). \end{cases} \tag{3.1}$$

Here, C is the cost defined for the volume. Note that the minimum cost of $A_1(x_1,x_2,x_3)$ is not computed yet when arriving at end point x_2, unlike with the original seam carving.

Step 2: Accumulation along the x_3 axis and determination of a seam in the x_1-x_3 plane at $x_2=n_2-1$.
A seam is derived in the x_1-x_3 plane at $x_2=n_2-1$ in this step. The accumulated cost function $A_1(x_1,x_2,x_3)$ is further accumulated in the x_3 direction, from $x_3=0$ to n_3-1, with minimization, and an accumulated cost function A_2 is obtained. The j value



x_2
x_1
x_3
The passes are memorized in P_1
Step1: Accumulation along x_2 axis in each x_1-x_2 plane
$S(n_2-1,x_3)$
Step2: Accumulation along x_3 axis and decision of a seam in x_1-x_3 plane $(x_2=n_2-1)$
$S(x_2,0)$
$S(x_2,n_3-1)$
$S(x_2,n_3-3)$
$S(x_2,n_3-2)$
The blue seams are obtained independently by selecting a pass among P_1
Step3: Decision of a seam in each x_1-x_2 plane

Figure 3.3: The discontinuous DP process. The seam surface is connected in the x_2 direction. The seam obtained as an intersection between the seam surface and the x_1-x_3 plane at $x_2=n_2-1$ is connected in the x_3 direction. The seams in different x_1-x_3 planes are not necessarily connected.

selected at the minimization is saved in a path $P_2(x_1,n_2-1,x_3)$.

$$\begin{cases} A_2(x_1,x_2,x_3) = A_1(x_1,x_2,x_3) + \min\limits_{j\in\{-1,0,1\}} A_2(x_1+j,x_2,x_3-1), & (x_3>0) \\ P_2(x_1,x_2,x_3) = \arg\min\limits_{j\in\{-1,0,1\}} A_2(x_1+j,x_2,x_3-1)+x_1, \ (x_3>0) \\ A_2(x_1,x_2,0) = A_1(x_1,x_2,0). \end{cases} \tag{3.2}$$

Then, x_1, which minimizes $A_2(x_1,n_2-1,n_3-1)$ in the current x_2, is chosen and assigned to $S(n_2-1,n_3-1)$. A seam is obtained as the optimal path by following P_2 from

$x_3 = n_3 - 1$ to 0.

$$\begin{cases} S(x_2,n_3-1) = \underset{x_1}{\arg\min} A_2(x_1,x_2,n_3-1) \\ S(x_2,x_3) = P_2(S(x_2,x_3+1),x_2,x_3+1), \quad (x_3 < n_3-1). \end{cases} \tag{3.3}$$

Step 3: Accumulation along the x_3 axis and determination of a seam in each $x_1 - x_3$ plane.
A seam in each $x_1 - x_3$ plane is derived in this step, starting from $x_2 = n_2 - 2$ and then reducing x_2 by one. First, the accumulated cost function A_2 is updated at each $x_1 - x_3$ plane:

$$A_2(x_1,x_2,x_3) = \begin{cases} A_2(x_1,x_2,x_3), & (|x_1 - S(x_2+1,x_3)| \leq 1) \\ \infty, & (otherwise). \end{cases} \tag{3.4}$$

This update has the effect of making the seams between consecutive x_2's connected.

Second, an accumulated cost function $A_2(x_1,x_2,x_3)$ and a path $P_2(x_1,x_2,x_3)$ are obtained by accumulating $A_1(x_1,x_2,x_3)$ in the x_3 directions, from $x_3 = 0$ to $n_3 - 1$, in the same way as Eq. (3.2). Then, x_1, which minimizes $A_2(x_1,x_2,n_3-1)$, is chosen and assigned to $S(x_2, n_3-1)$. A seam is obtained as the optimal path by following P_2 from $x_3 = n_3 - 1$ to zero in the same way as in Eq. (3.3). Subsequently, x_2 is reduced by one, and the derivation of a seam in the next x_2 is started from Eq. (3.2). The complete seam surface is obtained when the process ends at $x_2 = 0$.

Fig. 3.2 shows the continuous DP process. The gray path shows the seam obtained at $x_2 = n_2 - 1$. By creating a seam in each $x_1 - x_3$ plane successively so as to be connected to the seam in the previous x_2, the continuous method enables the seam surface to become totally connected in the x_2 direction. In addition, the seam obtained in each x_2 is connected in the x_3 direction because j is selected from among $\{-1,0,1\}$ in Eq. (3.2). Hence, the obtained seam surface $S(x_2,x_3)$ is guaranteed to be connected.

3.3.2 Discontinuous Method to Obtain a Discontinuous Seam Surface

Steps 1 and 2 are virtually the same as in subsection 3.3.1. The only difference is that, unlike in the continuous method, a path P_1 in the x_2 direction is saved in Eq. (3.1) to give

$$P_1(x_1,x_2,x_3) = \underset{j\in\{-1,0,1\}}{\arg\min} A_1(x_1+j,x_2-1,x_3)+x_1, \quad (x_2 > 0). \tag{3.5}$$

Step 3: Determination of a seam in each x_1-x_2 plane
A seam in each x_1-x_2 plane is derived in this step. By selecting a path among those in P_1 that crosses seam $S(n_2-1,x_3)$ in the x_1-x_3 plane at $x_2=n_2-1$, the discontinuous method can obtain a seam independently in each x_1-x_2 plane.

$$S(x_2,x_3) = P_1(S(x_2+1,x_3),x_2+1,x_3), \qquad (x_2<n_2-1) \tag{3.6}$$

The complete seam surface is derived by obtaining a seam in every x_1-x_2 plane.

Fig. 3.3 shows the process followed by the discontinuous method. The gray path shows the seam obtained at $x_2=n_2-1$. The seam obtained in each x_3 is connected in the x_2 direction because j is selected from among $\{-1,0,1\}$ in Eq. (3.5). Although the seam in $x_2=n_2-1$ is connected in the x_3 direction because of Eq. (3.2), any seam obtained as an intersection of the seam surface and other x_1-x_3 planes is not guaranteed to be connected because Eq. (3.3) is calculated independently at each x_3. The connective seam in the x_1-x_3 plane at $x_2=n_2-1$, however, has an effect that makes the seam surface prone to connecting in other x_1-x_3 planes.

3.3.3 Concrete Example

For ease of understanding, we explain the concrete process of the proposed method in video retargeting. Let (x,y,t) be the pixel location (x,y) in the t-th frame in the video. For video retargeting to reduce the width of the image, DP is first conducted in each $x-y$ plane to the direction $y=n-1$ (where n is the image height). Specifically, the cost is accumulated at the $x-t$ plane ($y=n-1$). Then, the second DP is applied to the $x-t$ plane, where $y=n-1$. For the continuous mode, the best seam in the $x-t$ plane ($y=n-2$) is generated to ensure the connectivity with that in the $x-t$ plane ($y=n-1$), and this process is repeated until the last $x-t$ plane ($y = 0$). In this manner, connectivity is ensured even though the generated seam surface is suboptimal, not global optimal. For the discontinuous mode, the seam is searched by tracing back in each $x-y$ plane. In this mode, connectivity is not guaranteed, but a lower energy path than that in the continuous mode can be found.

3.4 Derivation of the Proposed Multi-Pass DP

In this section, we describe the derivation of the proposed multi-pass DP.

3.4.1 DP in 2D Plane

First, we consider the process of dynamic programming (DP) to obtain an optimal path in a two dimensional plane as a simple example. As shown in Fig. 3.4, the objective is to obtain a path S that crosses the x_1 axis in the $x_1 - x_2$ plane ($0 \le x_1 < n_1$, $0 \le x_2 < n_2$). When the path passes coordinate (x_1,x_2), we describe it as $x_1 = S(x_2)$. We call the path "seam" in seam carving problems, and the seam must be connected in the x_2 direction. The cost $C(x_1,x_2)$ is assigned to each coordinate (x_1,x_2), and our objective is to obtain a seam such that the sum of the cost is minimum. This optimization problem is described as

$$\arg\min_{S} \sum_{x_2} C(S(x_2),x_2), \tag{3.7}$$

$$s.t.\ |S(x_2) - S(x_2+1)| \le 1. \tag{3.8}$$

The accumulation process of DP along the x_2 axis is described as

$$\begin{cases} A(x_1,0) = C(x_1,0), \\ A(x_1,x_2) = C(x_1,x_2) + \min\limits_{j\in\{-1,0,1\}} A(x_1+j,x_2-1), \quad (x_2>0) \\ P(x_1,x_2) = \arg\min\limits_{j\in\{-1,0,1\}} A(x_1+j,x_2-1) + x_1, \ (x_2>0) \end{cases} \tag{3.9}$$

where $A(x_1,x_2)$ is the accumulated cost function, and the value of j selected during the accumulation is recorded in paths P. The optimal path is obtained by tracking back P from $x_2 = n_2 - 1$ to 0.

$$\begin{cases} S(n_2-1) = \arg\min\limits_{x_1} A(x_1,n_2-1), \\ S(x_2) = P(S(x_2+1),x_2+1), \quad (x_2<n_2-1) \end{cases} \tag{3.10}$$

Because j is selected from among $\{-1,0,1\}$, the obtained path $S(x_2)$ is guaranteed to be connected in the x_2 direction, in other words, Eq. (3.8) is satisfied.

The optimal path is obtained by the process described above. However, Fukushima et al. [70] stated that a suboptimal solution that is almost equal to the optimal one can be obtained by selecting the x_1 that has the minimum accumulated cost $A(x_1,x_2)$ at each x_2 without recording paths for tracking back (Fig. 7 in [70]). That process is described as

$$S(x_2) = \arg\min_{x_1} A(x_1,x_2). \tag{3.11}$$

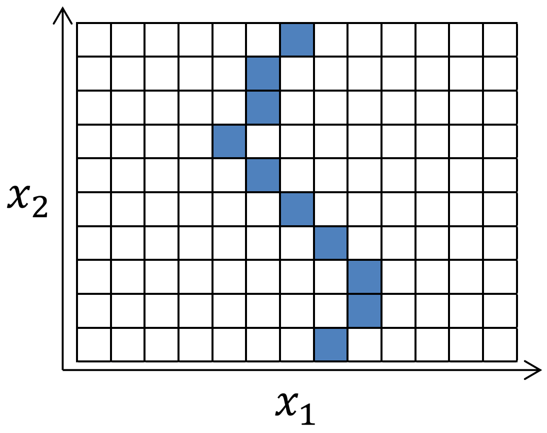


Figure 3.4: Connected pass by DP in 2D plane.

However, the path obtained by Eq. (3.11) is not guaranteed to be connected in the x_2 direction. This method is reasonable for disparity estimation, in which (x_1, x_2) corresponds to $(disparity, xpixel)$, and the smoothness of the path (disparity) is required, but the connectivity is not. Therefore, for seam carving problems in which the connectivity is required, we vary Eq. (3.11) as

$$\begin{cases} S(n_2-1) = \arg\min_{x_1} A(x_1, n_2-1), \\ S(x_2) = \arg\min_{x_1 \in \{S(x_2+1), S(x_2+1)\pm 1\}} A(x_1, x_2). \quad (x_2 < n_2-1) \end{cases} \tag{3.12}$$

First, the x_1 which has the minimum accumulated cost $A(x_1, x_2)$ is selected at $x_2 = n_2 - 1$. Subsequently, x_2 is reduced by one, and the range of x_1 is restricted among $\{S(x_2+1)-1, S(x_2+1), S(x_2+1)+1\}$ where $S(x_2+1)$ is the previously selected x_1. In other words, the candidate of the next x_1 is restricted within the range of ±1 of the previous result. The final seam that is connected in the x_2 direction is obtained by repeating the process from $x_2 = n_2 - 2$ to 0.

The seam obtained by Eq. (3.12) is equal to the one obtained by Eq. (3.13).

$$\begin{cases} S(x_2) = \arg\min_{x_1} A(x_1, x_2), \\ A(x_1, x_2) = \begin{cases} A(x_1, x_2), & (|x_1 - S(x_2+1)| \leq 1) \\ \infty, & (otherwise). \end{cases} \end{cases} \tag{3.13}$$

When x_2 is reduced by one, the costs that are out of the range of ±1 of the previous result is set to infinity. This setting makes the obtained seam satisfy Eq. (3.8).

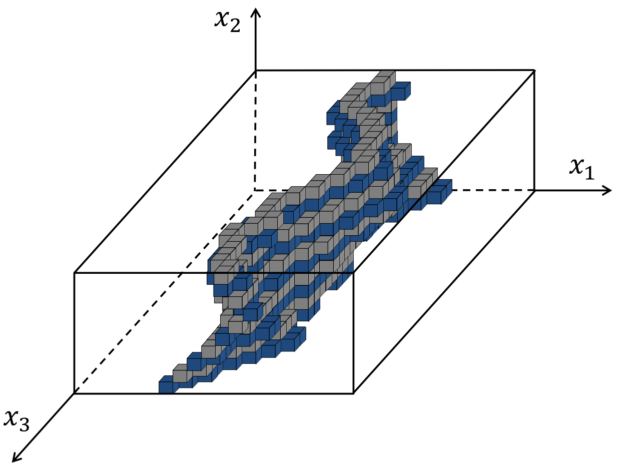


Figure 3.5: Connected seam surface in 3D space.

3.4.2 DP in 3D Volume

In this section, we consider the problem to obtain a seam surface that is connected in three dimensional space as shown in Fig. 3.5. When the seam surface passes coordinate (x_1,x_2,x_3), we describe it as $x_1 = S(x_2,x_3)$. This optimization problem is described as

$$\arg\min_{S} \sum_{x_2,x_3} C(S(x_2,x_3),x_2,x_3), \tag{3.14}$$

$$s.t.\ |S(x_2,x_3) - S(x_2+1,x_3)| \le 1, \tag{3.15}$$

$$|S(x_2,x_3) - S(x_2,x_3+1)| \le 1. \tag{3.16}$$

The globally optimal solution of this problem cannot be obtained by DP as Rubinstein et al.. [10] pointed out. Here, we consider the reason why DP cannot directly be applied to this volume seam carving problem. We first perform the accumulation process of DP along the x_2 axis in each $x_1 - x_2$ plane independently as Eq. (3.17).

$$\begin{cases} A_1(x_1,0,x_3) = C(x_1,0,x_3), \\ A_1(x_1,x_2,x_3) = C(x_1,x_2,x_3) + \min\limits_{j\in\{-1,0,1\}} A_1(x_1+j,x_2-1,x_3). \quad (x_2>0) \\ P_1(x_1,x_2,x_3) = \arg\min\limits_{j\in\{-1,0,1\}} A_1(x_1+j,x_2-1,x_3)+x_1, \ \ (x_2>0). \end{cases} \tag{3.17}$$

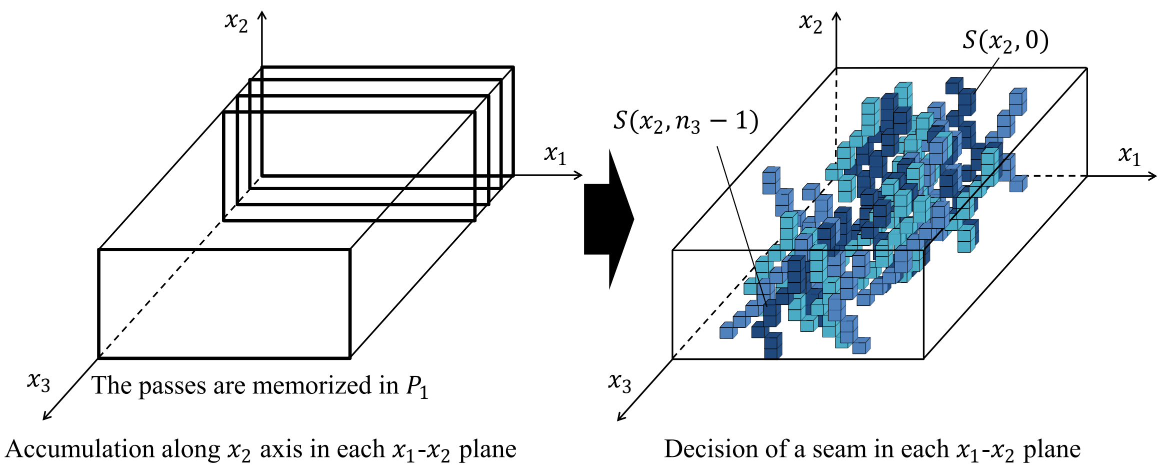

x_2
x_1
x_3
The passes are memorized in P_1
Accumulation along x_2 axis in each x_1-x_2 plane
x_2
$S(x_2, 0)$
$S(x_2, n_3 - 1)$
x_1
x_3
Decision of a seam in each x_1-x_2 plane

Figure 3.6: DP in each $x_1 - x_2$ plane independently. Although the obtained seam is connected in the x_2 direction, not connected in the x_3 direction.

We obtain the seam surface by tracking back as Eq. (3.18).

$$\begin{cases} S(n_2-1,x_3) = \arg\min\limits_{x_1} A_1(x_1,n_2-1,x_3) \\ S(x_2,x_3) = P_1(S(x_2+1,x_3),x_2+1,x_3), \qquad (x_2<n_2-1) \end{cases} \tag{3.18}$$

The obtained seam surface is connected in the x_2 direction because j is selected from among $\{-1,0,1\}$. However, it is not connected in the x_3 direction as shown in Fig. 3.6 because the DP is performed in each $x_1 - x_2$ plane independently. In other words, although Eq. (3.15) is satisfied, Eq. (3.16) is not.

3.4.3 Multi-Pass DP in 3D Volume

Continuous Method

The continuous method of the proposed multi-pass DP can obtain a suboptimal solution that is connected in both the x_2 and x_3 directions in 3D space as shown in Fig. 3.5. The flow chart is shown in Fig. 3.2. We derivate our continuous method by extending the suboptimal solution of DP in 2D plane to 3D space.

Definition

We define a vector whose elements are $x_1 = S(x_2,x_3)$ $(x_3 = 0,\cdots,n_3-1)$ as $\boldsymbol{X}_1$ when x_2 is fixed.

$$\boldsymbol{X}_1 := \boldsymbol{S}(x_2) = [S(x_2,0),\cdots,S(x_2,n_3-1)]. \tag{3.19}$$

A $\boldsymbol{X}_1$ corresponds to a gray (or blue) seam in Fig. 3.5. We denote by $C(\boldsymbol{X}_1, x_2)$ the cost of the seam $\boldsymbol{X}_1$, in other words, $C(\boldsymbol{X}_1, x_2)$ is the sum of the costs at the coordinates where the seam $\boldsymbol{X}_1$ passes.

$$C(\boldsymbol{X}_1, x_2) := \sum_{x_3} C(S(x_2, x_3), x_2, x_3). \tag{3.20}$$

Extension of the 2D suboptimal to 3D

First, we independently accumulate the costs along the x_2 axis in each $x_1 - x_2$ plane, similarly to Eq. (3.17). This process is the step 1 in subsection 3.3.1.

If simply tracking back P_1, the obtained seam surface is not guaranteed to be connected in the x_3 direction as described in Section 3.4.2. As Rubinstein et al. [10] pointed out, we cannot obtain an optimal solution by DP in volume seam carving problems. In other words, we cannot extend the optimal solution obtained by Eq. (3.10) to 3D space. Therefore, we instead extend the suboptimal solution in Eq. (3.11) to 3D space in Eq. (3.22).

$$\boldsymbol{S}(x_2) = \arg\min_{\mathbf{X}_1} \sum_{x_3} A_1(S(x_2, x_3), x_2, x_3) \tag{3.21}$$

$$= \arg\min_{\mathbf{X}_1} A_1(\mathbf{X}_1, x_2). \tag{3.22}$$

In Eq. (3.11), we obtain a suboptimal seam by selecting the x_1 whose cost is the minimum at each x_2. Similarly, in Eq. (3.22), we obtain a suboptimal seam surface by selecting the seam $\boldsymbol{X}_1$ whose cost is minimum in each $x_1 - x_3$ plane. The solution $\boldsymbol{X}_1 = [S(x_2, 0), \cdots, S(x_2, n_3 - 1)]$ in Eq. (3.22) can be obtained by performing 2D DP in the $x_1 - x_3$ plane.

$$\begin{cases} A_2(x_1,x_2,0) = A_1(x_1,x_2,0). \\ A_2(x_1,x_2,x_3) = A_1(x_1,x_2,x_3) + \min\limits_{j\in\{-1,0,1\}} A_2(x_1+j,x_2,x_3-1), \quad (x_3>0) \\ P_2(x_1,x_2,x_3) = \arg\min\limits_{j\in\{-1,0,1\}} A_2(x_1+j,x_2,x_3-1)+x_1, \ (x_3>0) \\ S(x_2,n_3-1) = \arg\min\limits_{x_1} A_2(x_1,x_2,n_3-1) \\ S(x_2,x_3) = P_2(S(x_2,x_3+1),x_2,x_3+1), \quad (x_3<n_3-1). \end{cases} \tag{3.23}$$

This is the step 2 in subsection 3.3.1. Because j is selected from among $\{-1, 0, 1\}$, the obtained seam $\boldsymbol{X}_1$ is guaranteed to be connected in the x_3 direction, in other words, it satisfies Eq. (3.16).

However, the obtained seam $\boldsymbol{X}_1$ is not guaranteed to be connected in the x_2 direction ($\boldsymbol{S}(x_2)$ and $\boldsymbol{S}(x_2+1)$ are not connected), in other words, it does not satisfy Eq. (3.15). Therefore, similarly to Eq. (3.12), we restrict the range of candidate seam $\boldsymbol{X}_1$ within ±1 of the previous seam $\boldsymbol{S}(x_2+1)$ when reducing x_2 by one. Subsequently, we obtain the optimal seam in Eq. (3.22). Similarly to Eq. (3.13), this process is redescribed as

$$A_2(x_1,x_2,x_3)=\begin{cases} A_2(x_1,x_2,x_3), & (|x_1-S(x_2+1,x_3)|\leq 1) \\ \infty. & (otherwise) \end{cases} \tag{3.24}$$

This is the step 3 in the continuous method in subsection 3.3.1. From $x_2 = n_2 - 1$ to 0, by repeating the acquisition of seam in Eq. (3.23) and the restriction of the range in Eq. (3.24) alternately, we can obtain a suboptimal seam surface in both the x_2 and x_3 directions.

3.4.4 Discontinuous Method

As shown in Fig. 3.3, the discontinuous method can obtain a seam surface that is connected in the $x_1 - x_3$ plane at $x_2 = n_2 - 1$ by simply tracking back P_1 after first step 2 in Eq. (3.23). This connectivity has the effect of making the seam surface prone to connecting in other $x_1 - x_3$ planes, but not guaranteed to be connected in other $x_1 - x_3$ planes. It is important because disconnectivity is sometimes required for seam carving depending on the applications (*e.g.*, retargeting for videos with extreme motion) as described in the next section.

3.5 Applications and Results

To show that the proposed method is general, we applied our proposed multi-pass DP to three different applications. All experiments were performed on a machine with an Intel Core i7-2600 3.4-GHz CPU and 16 GB of RAM, with the methods implemented in C++. In our experiments, we also tried to use belief propagation [94], but it did not work well for our applications.

3.5.1 Video Retargeting

Method

Video retargeting is achieved by applying the proposed method to the energy functions presented by Rubinstein et al. [10]. When the backward energy is utilized, the procedure is as follows: First, the cost volume is calculated. Then, a seam surface is derived using the proposed method. Finally, the image size is reduced by one at each frame by removing the pixel on the seam surface. This process is iterated until the image size has reached the target size.

When the forward energy is utilized, the methods described in subsections 3.3.1 and 3.3.2 should be modified as follows. For the continuous method, Eq. (3.1) is replaced by

$$\begin{cases} A_1(x_1,x_2,x_3) = \min[A_1(x_1-1,x_2-1,x_3)+C_L(x_1,x_2,x_3), \\ \qquad A_1(x_1,x_2-1,x_3)+C_U(x_1,x_2,x_3), \\ \qquad A_1(x_1+1,x_2-1,x_3)+C_R(x_1,x_2,x_3)] \quad (x_2>0) \\ A_1(x_1,0,x_3) = \min[C_L(x_1,x_2,x_3), C_U(x_1,x_2,x_3), C_R(x_1,x_2,x_3)], \end{cases} \tag{3.25}$$

$$\begin{cases} C_L(x_1,x_2,x_3) = |I^t(x+1,y)-I^t(x-1,y)|+|I^t(x,y-1)-I^t(x-1,y)| \\ C_U(x_1,x_2,x_3) = |I^t(x+1,y)-I^t(x-1,y)| \\ C_R(x_1,x_2,x_3) = |I^t(x+1,y)-I^t(x-1,y)|+|I^t(x,y-1)-I^t(x+1,y)|, \end{cases} \tag{3.26}$$

where $(x,y,t)=(x_1,x_2,x_3)$ and $I^t(x,y)$ is a luminance image of the t_{th} frame. For the discontinuous method, Eq. (3.5) is replaced in the same manner as in the continuous method. We adopted the forward energy for video retargeting in all the experiments.

Experimental Results

We tested the proposed method with five videos. The results for the five videos are shown in Fig. 3.7. The resolution (width, height, and number of frames) of the original videos 1 - 5 in Fig. 3.7 is (352, 288, 300), (148, 144, 131), (282, 288, 91), (540, 280, 99), and (540, 280, 97), respectively. For comparison, the results for the seam surfaces derived by graph cuts are shown in Fig. 3.7(d). A simple multi-resolution method was adopted for Fig. 3.7(d). More specifically, first, the images were downsampled by four, without any prefiltering, and a preliminary seam surface was derived by graph cuts. Then, the maximum and minimum value of the preliminary seam surface $x_{1\text{max}}, x_{1\text{min}}$ were calculated. Finally, a final seam surface was derived from

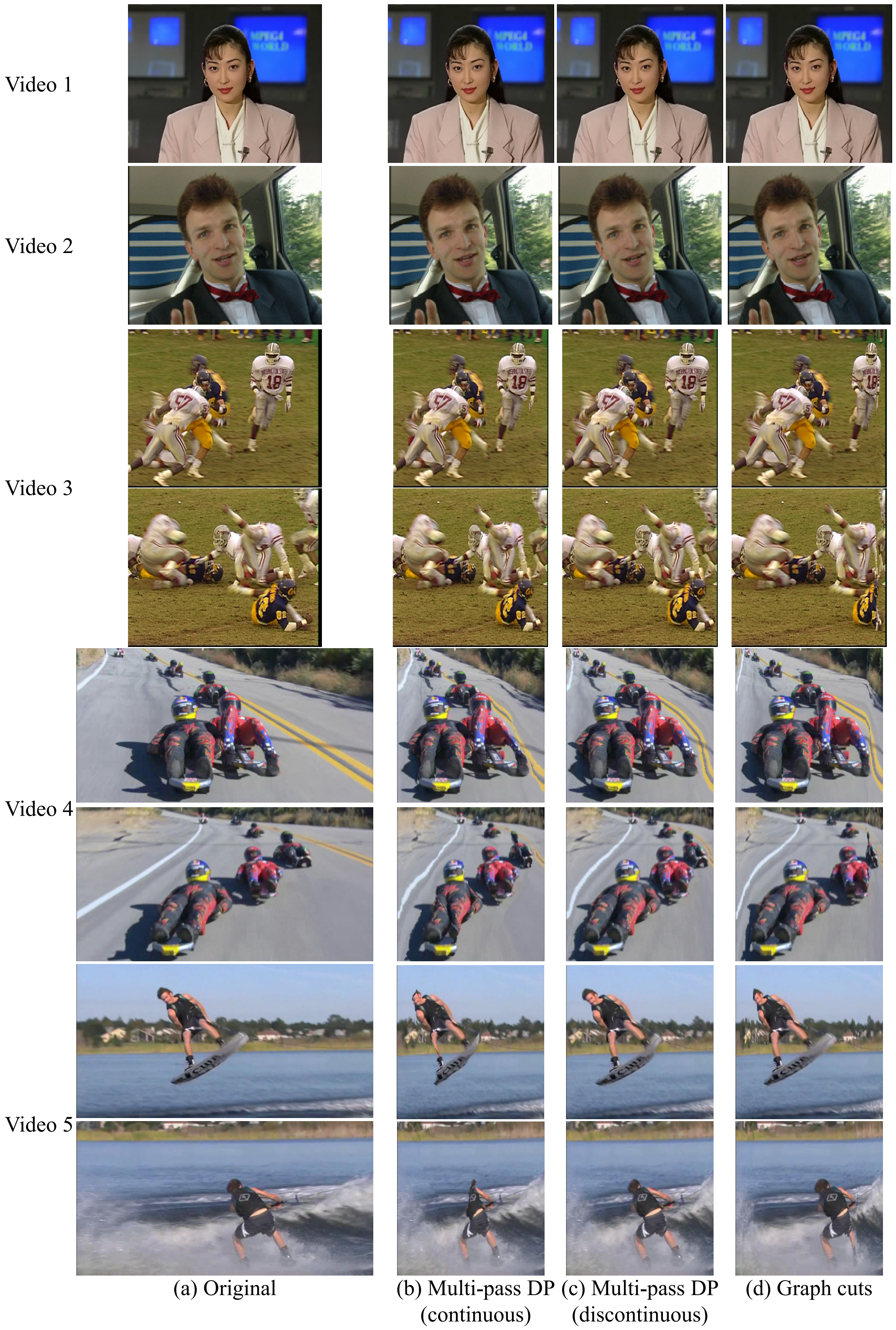

Video 1
Video 2
Video 3
Video 4
Video 5
(a) Original
(b) Multi-pass DP (continuous)
(c) Multi-pass DP (discontinuous)
(d) Graph cuts

Figure 3.7: Results of the video retargeting. The upper and lower rows are different frames in videos 3, 4, and 5.

the full-resolution images by graph cuts within the range of x_1 from $(4x_{1\mathrm{min}}-4)$ to $(4x_{1\mathrm{max}}+4)$. No multi-resolution or downsampling was adopted for Fig. 3.7(b) and (c) because it is not necessary.

Videos 1 and 2 are scenes in which foreground objects are moving and the background is stable. The other videos are scenes in which both the foreground and the background are moving. In videos 1 and 2, noticeable deterioration is not created in Fig. 3.7(b) and (d). In contrast, the foreground or background is shaking between frames in Fig. 3.7(c) because connectivity is not guaranteed in (c). In video 3, visible deterioration is not created in Fig. 3.7(b) and (c). In videos 4 and 5, some players are distorted, as shown in Fig. 3.7(b) and (d). It is clear that the continuous method is suited for videos that have little motion and that the discontinuous method is suited for videos whose motion is strenuous. The discontinuous method may be suited for high-motion scenes compared to graph cuts. If a salient object is moving, a seam near the object should leap the object between adjacent frames to avoid distorting it. When graph cuts or the continuous DP is utilized, the seam cannot leap the object and then may cross it. On the other hand, our discontinuous multi-pass DP can avoid this problem by its discontinuous nature.

Fig. 3.8 shows the seam surfaces obtained for videos 1 and 5 in Fig. 3.7. Fig. 3.8(a) and (c) verifies that the seam surfaces are connective because the values change smoothly over the entire seam surface. The seam surfaces obtained by all methods for video 1 have horizontal stripes. This occurs because the object has moved slightly; therefore, the seam surface crosses at virtually the same position in every frame. In Fig. 3.8(b), video 1, some vertical stripes are shown, which cause background shaking. In Fig. 3.8(b), video 5, many vertical stripes are shown; however, the stripes vanish at the bottom of the seam surface because the seam surface is connective in the $x_2 = n_2 - 1$ plane.

Fig. 3.9 shows histograms of the discontinuous level in seam surfaces using the discontinuous method. Discontinuous level k is defined as $k = |S(y,t+1) - S(y,t)|$. The vertical axis shows the frequency of k averaging in all seam surfaces. $k = 0$ and 1 indicate that the seam surface is connective at a specific point. Other values of k indicate that the seam surface is discontinuous at those points. In Fig. 3.9(a), because the sum of the frequency at $k = 0$ and 1 is approximately 60%, the seam surfaces are connective over approximately 60% of the area. In Fig. 3.9(b), because the sum of the frequency at $k = 0$ and 1 is approximately 30%, the seam surfaces are connective in only approximately 30% of the area. This figure clarifies why the seam surfaces for video 5 are more discontinuous than those for video 1.

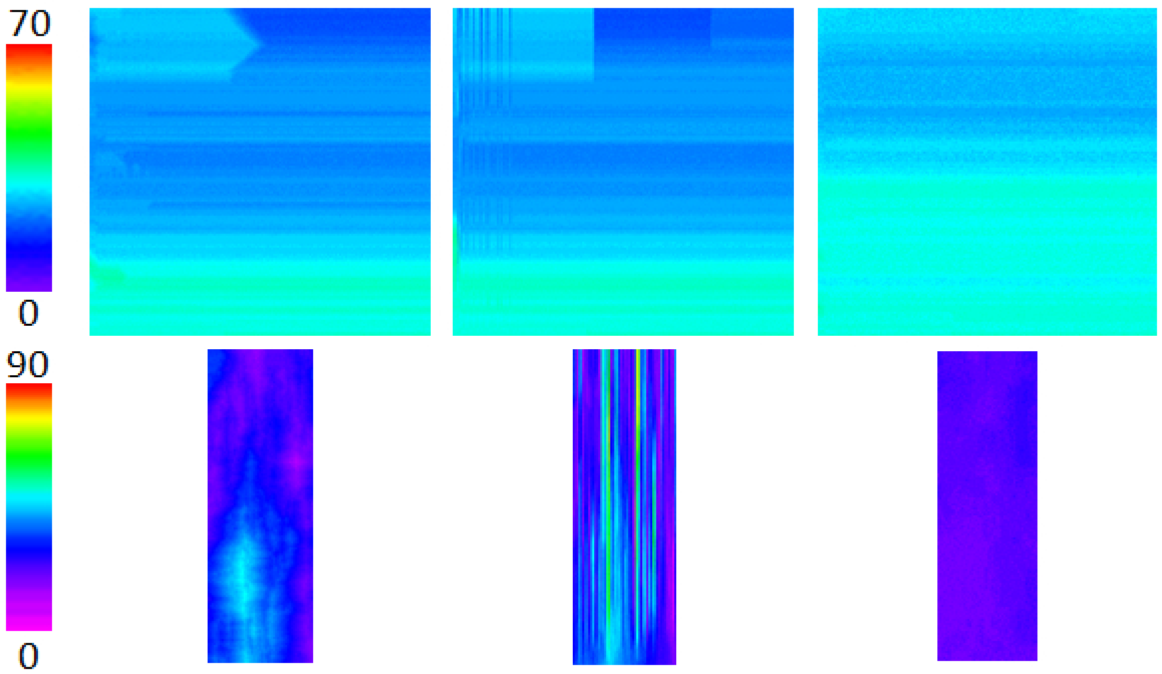
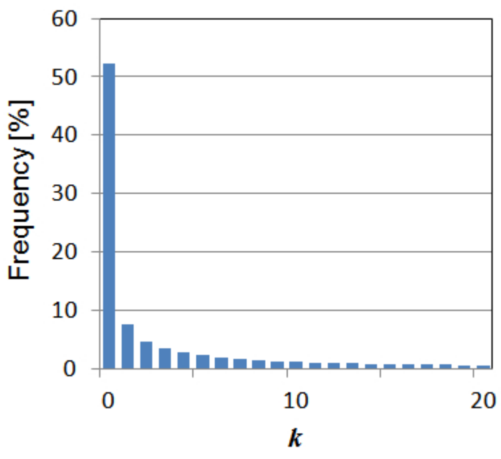


Figure 3.8: Seam surfaces obtained in Fig. 3.7 in the first iteration. The upper and lower lines are for videos 1 and 5, respectively. The horizontal axis shows the time.

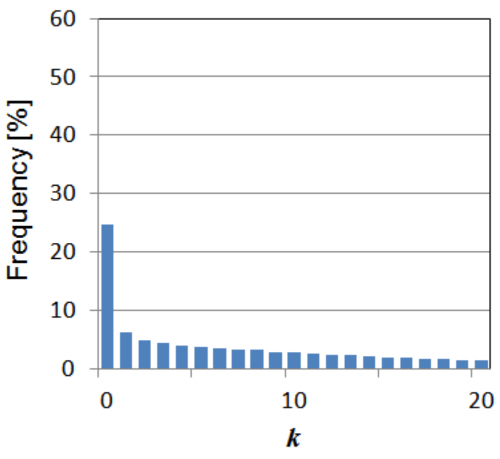
The computation time and maximum memory consumption are shown in Table 3.1(a) and (b), respectively. Continuous and discontinuous DP require virtually the same processing time in each video. By replacing graph cuts with multi-pass DP, we can reduce the computation time to 1.5%. The maximum memory consumption is reduced to 5.6% by replacing graph cuts with continuous DP and to 11% with discontinuous DP, even without the multi-resolution or downsampling. Note that the computation time and memory usage of our multi-pass DP would increase by the order of $O(n)$, where n is the number of voxels in the cost volume. On the other hand, those of graph cut-based methods grow more rapidly with n.

Subjective Analysis

We conducted subjective evaluation experiments to verify that the quality of the videos retargeted by our multi-pass DP is almost equal to that of graph cuts with respect to human perception. We used a crowdsourcing service, and 163 participants took part in the test. The flow of the test was as follows: First, subjects were asked to watch four videos (the original and three videos retargeted by our continuous



(a) Video 1



(b) Video 5

Figure 3.9: Histograms of the discontinuous level in seam surfaces using the discontinuous method.

method, our discontinuous method, and graph cuts) on our web page, in which the original video was displayed at the center of the top row and the three retargeted videos were arranged horizontally on the row beneath. After that, subjects were asked to "choose the best video in terms of quality." They could replay the videos any number of times and were able to choose one of four choices: "Video A is the best," "Video B is the best," "Video C is the best," or "Cannot notice the difference." The three retargeted videos and four choices were randomly ordered to remove any bias. Each video was scaled by a factor of $\min(\frac{320}{w}, \frac{240}{h})$, where w and h are the video width and height, respectively, to fit it in a 320×240 window. The aspect ratio was not changed. We used the eight videos mentioned in subsection 3.5.1 along with two other videos for dummy questions, in which three identical retargeted videos were displayed on the lower row. Therefore, the subjects were asked to answer 10 questions including two dummy questions. The order of the 10 questions was also random.

The result of the five videos in Fig. 3.7 is shown in Table 3.2(a). We observed that, on average, approximately half of the 163 subjects answered that there was no noticeable difference between the retargeted videos. For video 1, which contains little movement in the scene, the graph–cut method was the best of the three methods. However, our continuous method was the best for videos 2 and 3, which contain moderate motion. In addition, our discontinuous method was the best for videos 4 and 5, which contain extreme motion. On average, both of our methods obtained

Table 3.1: Computation time and memory consumption for Fig. 3.7.

(a) Computation time [s]

	Multi-pass DP (continuous)	Multi-pass DP (discontinuous)	Graph–cuts
Video 1	207	213	4988
Video 2	13	13	347
Video 3	83	85	4645
Video 4	445	449	43,419
Video 5	436	436	24,115
Average	237	239	15,503

(b) Maximum memory consumption [MB]

	Multi-pass DP (continuous)	Multi-pass DP (discontinuous)	Graph cuts
Video 1	124	241	2.4×10^3
Video 2	15	27	358
Video 3	41	77	721
Video 4	65	123	1.2×10^3
Video 5	64	121	866
Average	62	118	1.1×10^3

a score that was near that of graph cuts, and the sum of the two methods was significantly more than that of graph cuts. Therefore, in practical use, users can obtain an output that is equivalent to that of graph cuts by selecting the best result after applying both of our methods to the input video because our methods are very fast to calculate.

Table 3.2(b) shows the results of the 72 subjects who correctly selected the choice “Cannot notice the difference" in the dummy questions, *i.e.*, the subjects who were not fooled by the dummy questions. We can regard the result in Table 3.2(b) as a more reliable result than that of Table 3.2(a) because the noisy results of dishonest subjects were filtered out. We observed that, on average, 67.8% of these 72 subjects answered that there were no noticeable differences between the retargeted videos.

As discussed above, the majority of subjects could perceive no differences. In addition, here, we excluded the subjects who selected the choice “Cannot notice the difference” and show that there was no significant difference between the number of subjects who preferred graph cuts and multi-pass DP. Following [95] and [96],

Table 3.2: Number of subjects who preferred the retargeted video of each method.

(a) All subjects

	Multi-pass DP (continuous /discontinuous)	Graph cuts	Cannot notice the difference	Total
Video 1	24 / 8	39	**92**	163
Video 2	42 / 13	28	**80**	163
Video 3	40 / 17	22	**84**	163
Video 4	14 / 50	25	**74**	163
Video 5	19 / 39	37	**68**	163
Average	27.8 / 25.4	30.2	**79.6**	163

(b) Subjects who were not fooled by the dummy questions

	Multi-pass DP (continuous /discontinuous)	Graph cuts	Cannot notice the difference	Total
Video 1	6 / 2	11	**53**	72
Video 2	14 / 2	10	**46**	72
Video 3	11 / 2	5	**54**	72
Video 4	3 / 15	8	**46**	72
Video 5	4 / 12	11	**45**	72
Average	7.6 / 6.6	9.0	**48.8**	72

we conducted a significance test of score differences. We regarded the score of each method as the sum of the numbers in each column in Table 3.2(a) (*e.g.*, the score of the continuous DP is $24+42+40+14+19=139$). The scores of the continuous DP, discontinuous DP, and graph cuts were 139, 127 and 151, respectively, and total score was 417. Unlike in [95] and [96], our subjective evaluation test was not a paired comparison because we asked the subjects to compare three videos at one time. Therefore, we regarded these scores as the result of a paired comparison where the number of subjects was $n = 417/{}_3C_2 = 139$ and the number of objects (methods) was $t = 3$. We needed to find a value R so that there was no significant difference between the two methods with the confidence level α if the score difference was less than or equal to R. It can be found from Eq. (3.27):

$$P(W_{t,\alpha} \geq \frac{2R - 0.5}{\sqrt{nt}}). \tag{3.27}$$

We set the confidence level $\alpha = 0.05$ and obtained $W_{3,0.05} = 3.31$ because the values of $W_{t,\alpha}$ were tabulated in [97]. We obtained $R = 34.05$ from Eq. (3.27) and observed that the results of the three methods (continuous DP, discontinuous DP, and graph cuts) were not significantly different because the score differences between any pairs of the three methods were less than R.

3.5.2 Tone Mapping

Method

Most tone-mapping methods compress the Y (luminance) values of the YCbCr color space. We also compress the Y values, but we convert them using the logarithmic function in advance. Luminance image is defined by $L^0(x,y) = \log Y(x,y)$, where $Y(x,y)$ is the Y value of the input HDR image $I(x,y;c)$. First, a guided filter [87] is performed on the luminance image $L^0(x,y)$ and a high-frequency component $H(x,y)$ is calculated by

$$H(x,y) = \frac{L^0(x,y) - \bar{L}(x,y)}{L_{\max} - L_{\min}}, \tag{3.28}$$

where $\bar{L}(x,y)$ is the image obtained by the guided filtering. $L_{\min}$ and $L_{\max}$ are the minimum and maximum values of $L^0(x,y)$, respectively. The initial luminance image $L(x,y)$ is defined as

$$L(x,y) = (r_0 - 1)\frac{\bar{L}(x,y) - L_{\min}}{L_{\max} - L_{\min}}, \tag{3.29}$$

where r_0 is a constant that determines an initial luminance range.

Next, the cost volume is defined based on a local luminance histogram at each block. To reduce the memory consumption, we subsample the cost volume in both the space and the luminance directions in advance. A subsampled coordinate and luminance value is expressed as $\tilde{x} = x/(2p+1)$, $\tilde{y} = y/(2p+1)$, $\tilde{l} = l/q$ where $2p+1, q$ is the subsampling parameter of the space and luminance, respectively. The cost volume is defined as

$$C(\tilde{l},\tilde{x},\tilde{y}) = \sum_{i=-w}^{w}\sum_{j=-w}^{w}\sum_{k=0}^{q-1} f(\tilde{l}+i, \tilde{x}+j, \tilde{y}+k) \tag{3.30}$$

$$f(l,x,y) = \begin{cases} 1, & l = \mathrm{floor}(L(x,y)) \\ 0, & l \neq \mathrm{floor}(L(x,y)), \end{cases}$$

where w is a window size and floor() indicates rounding down to an integer.

Then, a subsampled seam surface $\tilde{S}(\tilde{x},\tilde{y})$ is derived from the cost volume $C^t(\tilde{l},\tilde{x},\tilde{y})$ using the proposed continuous method. A seam surface in full resolution $S(x,y)$ is obtained from $\tilde{S}(\tilde{x},\tilde{y})$ by simple bilinear interpolation. Converting the luminance image using the obtained seam surface is expressed simply as

$$L'(x,y) = \begin{cases} L(x,y) - q, & L(x,y) > qS(x,y) \\ L(x,y), & L(x,y) \leq qS(x,y), \end{cases} \tag{3.31}$$

where $L'(x,y)$ is a converted luminance image. This equation has the effect of reducing the maximum luminance value by q. The meaning of this process is that the cuboid is divided into two parts by the seam surface: one part is pruned on the surface, and both parts are again bonded to each other. As a result, the length of the luminance direction of the cuboid becomes shorter by q. The process from Eqs. (3.30) to (3.31) is iterated until the dynamic range of $L'(x,y)$ becomes the target range r_T. The parameter q is calculated at every iteration using the current range r of $L(x,y)$ by $q = \text{ceil}(r/\rho)$, where ceil() indicates rounding up to an integer. The parameter ρ controls the maximum size of the cost volume in the luminance direction.

Finally, the high-frequency component ($H(x,y)$ in Eq. (3.28)) is added to the obtained luminance image $L'(x,y)$ with enhancement and the LDR luminance image $L^L(x,y)$ is obtained. The final dynamic range is controlled to be r_L by linear transformation. r_L is the desired range of the LDR image, typically $r_L = 256$:

$$L^L(x,y) = \frac{r_L}{r_T} L'(x,y) + \lambda(r_L - 1)H(x,y), \tag{3.32}$$

where λ is a constant that controls the level of enhancement. The LDR color image $I^L(x,y;c)$ is created from $L^L(x,y)$ by a method similar to that used by Fattal et al. [85]:

$$I^L(x,y;c) = L^L(x,y)\left(\frac{I(x,y;c)}{Y(x,y)}\right)^{s_c}, \tag{3.33}$$

where $c = \{R,G,B\}$ and the exponent s_c controls the color saturation of the resulting image. The method described above is almost the same as that of Tsubaki and Iwauchi [13]. The only difference is that graph cuts is replaced by the proposed multi-pass DP and the guided filter is utilized.

Experimental Eesults

We tested the continuous method with six HDR images obtained from [98], [99], and [100]. The results for the six images are shown in Fig. 3.10. The resolution (width, height) of images 1 - 6 in Fig. 3.10 is (1000, 664), (760, 1016), (512, 381), (401, 535), (512, 768), and (1000, 563), respectively. The parameters $r_0 = 1024, r_T = 512, r_L = 256, p = 12, w = 12, \rho = 128, \lambda = 4, s_c = 0.5$ were used. In the guided filtering, the window size was five and $\varepsilon = 0.01$. For comparison, the results for the seam surfaces derived by graph cuts are shown in Fig. 3.10 (b). The images in Fig. 3.10 (a) and (b) look very similar. Fig. 3.11 shows the difference images for Fig. 3.10(a) and (b), where the contrast was exaggerated by eight. The mean square errors between Fig. 3.10(a) and (b) were 9.0 and 20.5 in images 1 and 6, respectively. Because the peak signal-to-noise ratio (PSNR) was approximately 40, there were no visible differences between these images. Fig. 3.12 shows the seam surfaces obtained for Fig. 3.10 in the last iteration. The seam surface appears to be smooth and keeps the connectivity with the adjacent blocks.

Fig. 3.10(c) and (d) shows the results for LLF [2] and LEPF [3], respectively, for comparison. The parameters $\sigma_r = \log(2.5), \alpha = 0.5, \beta = 0$ were used for LLF. On the whole, the detail is clear in (c) and (d); however, the contrast in (a) and (b) is higher than that in (c) and (d). We evaluated these images using Q_{MOS}, sharpness, and variance. Q_{MOS}, the mean-opinion-score prediction from 0 to 100, where 100 is the highest quality, was proposed by Mantiuk et al. [101] for evaluating image quality. Sharpness is defined as $S = \frac{1}{N}\sum|I|$, where N is the number of pixels in image I [3]. Furthermore, we calculated the variance of the luminance image to express the image contrast. Fig. 3.13 shows the Q_{MOS}, sharpness, and variance of the images in Fig. 3.10. Multi-pass DP and graph cuts had similar values in all the indexes. LLF had the highest Q_{MOS}, and LEPF had a high sharpness. Multi-pass DP and graph cuts had high variance but had moderate values for Q_{MOS} and sharpness.

The computation time and maximum memory consumption are shown in Table 3.3(a) and (b), respectively. By replacing graph cuts with continuous DP, we reduced the computation time to 2.2% and the maximum memory consumption to 51%. The number of iterations was 81 in (a) and (b) for each image.

Fig. 3.14 shows the computation time and maximum memory consumption when the subsampling parameter p was changed. The ratio of the time for multi-pass DP to the time for graph cuts was smaller when p was smaller. The memory consumption for multi-pass DP did not vary by p.

Table 3.3: Computation time and memory consumption for Fig. 3.10.

	(a) Computation time [s]		(b) Max memory consumption [MB]	
	Multi-pass DP (continuous)	Graph cuts	Multi-pass DP (continuous)	Graph cuts
Image 1	0.88	18.38	40	70
Image 2	4.17	245.29	46	82
Image 3	0.28	24.82	12	22
Image 4	0.28	3.69	14	24
Image 5	0.53	10.99	24	42
Image 6	0.76	26.09	34	90
Average	1.15	54.88	28	55

Subjective Analysis

We conducted subjective evaluation tests for the tone-mapping results, similar to those for video retargeting in subsection 3.5.1, to verify that the image quality of the results of our proposed method was almost equal to that of graph cuts with respect to human perception. The number of subjects was 198, and only two tone-mapped images were displayed side by side on the web page. Subjects were asked to select one of three choices: "Image A is better," "Image B is better," or "Cannot notice the difference." We used the 10 HDR images mentioned in subsection 3.5.2 and three other images for the dummy questions, in which two identical images were displayed.

The results for the six images in Fig. 3.10 are shown in Table 3.4(a). We did not compare our method with the results of LLF [2] in Fig. 3.10(c) and LEPF [3] in Fig. 3.10(d). These methods are not volume seam carving-based methods, and the purpose of this experiment was to verify that the same image quality can be obtained by our multi-pass DP and graph cuts. We observed that, on average, 60.5% of the 198 subjects noticed no difference between the two tone-mapped images.

Table 3.4(b) shows the results of the 101 subjects who were not fooled by the three dummy questions. We observed that, on average, 80.4% of the 101 subjects noticed no difference between the two tone-mapped images.

We conducted a significance test of score difference for Table 3.4(a), similar to that in video retargeting. The score of continuous DP and graph cuts was 216 and 253, respectively, and the total score was 469. We regarded these scores as the results of a paired comparison where the number of participants was $n = 469$. We obtained $W_{2,0.05} = 3.64$ from [97] and calculated $R = 42.67$ from Eqs. (3.27). Because the score

Table 3.4: Number of subjects who preferred the tone-mapped image of each method.

(a) All subjects

	Multi-pass DP (continuous)	Graph cuts	Cannot notice the difference	Total
Image 1	23	46	**129**	198
Image 2	**86**	48	64	198
Image 3	28	58	**112**	198
Image 4	30	28	**140**	198
Image 5	36	35	**127**	198
Image 6	13	38	**147**	198
Average	36.0	42.2	**119.8**	198

(b) Subjects who were not fooled by the dummy questions

	Multi-pass DP (continuous)	Graph cuts	Cannot notice the difference	Total
Image 1	3	11	**87**	101
Image 2	34	14	**53**	101
Image 3	7	17	**77**	101
Image 4	5	6	**90**	101
Image 5	7	9	**85**	101
Image 6	1	5	**95**	101
Average	9.5	10.3	**81.2**	101

difference was less than R, we observed that there was no significant difference between the two methods.

3.5.3 Contrast Enhancement

Method

We propose a contrast enhancement method based on luminance range compression. First, the dynamic luminance range is reduced via a process similar to tone mapping in subsection 3.5.2. The luminance values that have low frequency in the local histogram are then shrunk in this process. Next, the dynamic range is reverted to the original range using the linear enlargement method. The local contrast is enhanced by this process because the spatial gradient of the luminance image is relatively enlarged.

The compression process is the same as in Eqs. (3.28)-(3.31), with the exception that luminance value $Y(x,y)$ in the YCbCr color space is used for $L^0(x,y)$ directly ($L^0(x,y) = Y(x,y)$) instead of using a logarithmic function, and parameter q is fixed at 1. The linear enlargement is performed using

$$L''(x,y) = \frac{L_{\max}-L_{\min}}{r_T}L'(x,y) + L_{\min}. \tag{3.34}$$

The high-frequency component is added by

$$L^L(x,y) = L''(x,y) + \lambda(L_{\max}-L_{\min}+1)H(x,y). \tag{3.35}$$

Finally, a color image $I^L(x,y;c)$ is created from $L^L(x,y)$ using Eq. (3.33).

Experimental Results

We tested the continuous method with six images obtained from [2] and [102]. The results for the six images are shown in Fig. 3.15. The resolution (width, height) of images 1 - 6 in Fig. 3.15 is (800, 533), (800, 533), (512, 512), (876, 584), (795, 532), and (876, 584), respectively. The parameters $r_0 = L_{\max} - L_{\min} + 1, r_T = 128, p = 12, w = 12, \lambda = 2, s_c = 1.4$ were used. An 8-bit luminance value was reduced to 7 bits and then reverted to 8 bits. In the guided filtering, the window size was three and $\varepsilon = 0.01$. For comparison, the results for the seam surfaces derived by graph cuts are shown in Fig. 3.15(c). The images in Fig. 3.15(b) and (c) look very similar, with the exception that the blue sky is brighter in (b) than in (c). Fig. 3.16 shows the difference images for Fig. 3.15(b) and (c), where the contrast was exaggerated by eight. The mean square errors between Fig. 3.15(b) and (c) were 0.2 and 2.6 in image 2 and image 4, respectively. Because the PSNR was larger than 40, there were no visible differences between these images. Fig. 3.17 shows the seam surfaces obtained for Fig. 3.15(b) in the final iteration. The seam surface looks smooth and keeps the connectivity with the adjacent blocks.

Fig. 3.15(d) shows the results for the LLF method [2] for comparison. The parameters $\sigma_r = 0.4, \alpha = 0.25, \beta = 1.0$ were used for LLF. On the whole, the detail was clear in Fig. 3.15(d); however, the contrast in (b) and (c) was higher than that in (d). We evaluated these images using sharpness and variance in the same way as for tone mapping. Fig. 3.18 shows the sharpness and variance of the images in Fig. 3.15. Multi-pass DP and graph cuts had similar values in both indexes. The results for LLF were very sharp. The results for multi-pass DP and graph cuts had high variance, and the sharpness was higher than that in the original images.

Table 3.5: Computation time and memory consumption for Fig. 3.15.

	(a) Computation time [s]		(b) Maximum memory consumption [MB]	
	Multi-pass DP (continuous)	Graph cuts	Multi-pass DP (continuous)	Graph cuts
Image 1	0.90	335.89	28	79
Image 2	0.68	25.88	27	68
Image 3	0.46	115.37	17	44
Image 4	0.62	24.12	30	71
Image 5	0.89	99.25	28	74
Image 6	0.92	200.16	33	86
Average	0.75	133.45	27	70

The computation time and maximum memory consumption are shown in Table 3.5(a) and (b), respectively. By replacing graph cuts with continuous DP, we reduced the computation time to 0.6% and the maximum memory consumption to 39%.

Subjective Analysis

We conducted subjective evaluation tests for the contrast enhancement results, similar to those for tone mapping in subsection 3.5.2, to verify that the image quality of the results of our proposed method is almost equal to that of graph cuts with respect to human perception. The number of subjects was 191, and only two enhanced images were displayed side by side on the web page. Subjects were asked to select one of three choices: “Image A is better," “Image B is better," or “Cannot notice the difference." We used the 10 images mentioned in subsection 3.5.3 and three other images for the dummy questions, in which two identical images were displayed.

The results for the six images in Fig. 3.15 are shown in Table 3.6(a). We did not compare our method with the results of LLF [2] in Fig. 3.15(d). This latter method is not a volume seam carving-based method, and the purpose of this experiment is to verify that the same image quality can be obtained by our multi-pass DP as that of graph cuts. We observed that, on average, 34.7% of 191 subjects noticed no difference between the two tone-mapped images.

Table 3.6(b) shows the results of the 90 subjects who were not fooled by the three dummy questions. We observed that, on average, 46.3% of the 90 subjects noticed no difference between the two enhanced images.

Table 3.6: Number of subjects who preferred the enhanced image of each method.

(a) All subjects

	Multi-pass DP (continuous)	Graph cuts	Cannot notice the difference	Total
Image 1	74	**75**	42	191
Image 2	39	36	**116**	191
Image 3	37	**102**	52	191
Image 4	31	28	**132**	191
Image 5	16	**123**	52	191
Image 6	75	**112**	4	191
Average	45.3	**79.3**	66.3	191

(b) Subjects who were not fooled by the dummy questions

	Multi-pass DP (continuous)	Graph cuts	Cannot notice the difference	Total
Image 1	30	**36**	24	90
Image 2	9	9	**72**	90
Image 3	12	**42**	36	90
Image 4	7	4	**79**	90
Image 5	1	**53**	36	90
Image 6	35	**52**	3	90
Average	15.7	32.7	**41.7**	90

We conducted a significance test of score difference for Table 3.6(a), similar to that in tone mapping. The score of continuous DP and graph cuts was 476 and 272, respectively, and the total score was 748. We regarded these scores as the results of a paired comparison where the number of participants was $n = 748$. We obtained $W_{2,0.05} = 3.64$ from [97] and calculated $R = 53.82$ from Eq. (3.27). Because the score difference was higher than R, we observed that there was a significant difference between the two methods. However, there were no visible differences between the results from continuous DP and graph cuts in some images, as discussed in subsection 3.5.3 (*e.g.*, images 2 and 4 in Fig. 3.15). The quality improvement in other images is our future work.

3.6 Conclusions

In this chapter, we have proposed a fast volume seam carving method based on multi-pass DP and applied it to video retargeting, tone mapping, and contrast enhancement. The seam surface obtained by the continuous method is monotonic and connective, as with the method by Rubinstein et al. [10]. Discontinuous DP creates a seam surface that is monotonic and connective in one direction. We verified that suboptimal seam surfaces obtained by our methods can be utilized sufficiently in those applications. We also showed that our methods create a seam surface that is approximately 90 times faster and consumes 8 times less memory than conventional methods, which makes volume seam carving more practical.

Changing the axes enables video seam carving to be applied to entirely different applications. The characteristics of the cost volume and the optimal seam surfaces differ depending on the applications. In tone mapping and contrast enhancement, because a seam surface is only utilized to determine whether the value of the seam surface at a pixel is higher than the luminance value of the image, many different seam surfaces can give the same result, in contrast to video retargeting. Furthermore, the density of the cost volume is different. Because the cost volume in tone mapping is very sparse, it is easy to find a seam surface whose energy is sufficiently low. However, because the cost volume in video retargeting is dense, the optimality of seam surfaces is more important.

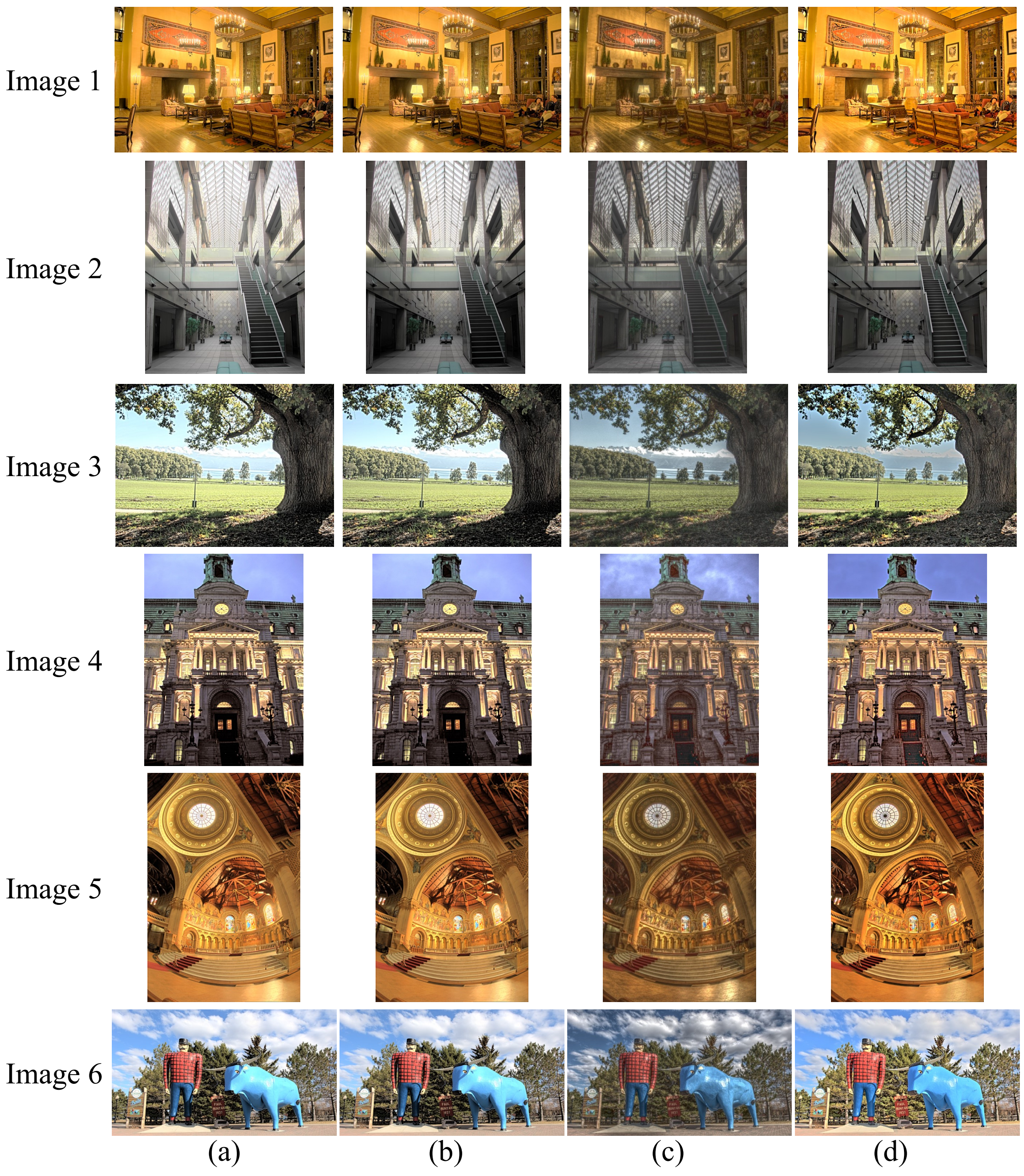


Figure 3.10: Tone mapping results. (a) Multi-pass DP (continuous), (b) graph cuts, (c) LLF [2], and (d) LEPF [3].

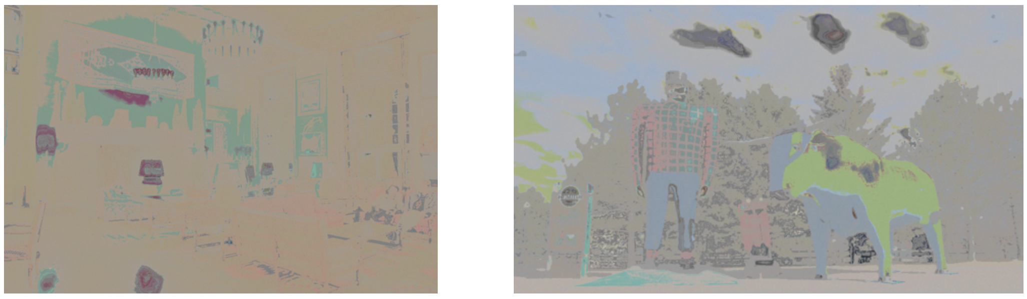


Figure 3.11: Difference images for Fig. 3.10 (a) and (b). The contrast was exaggerated by eight.

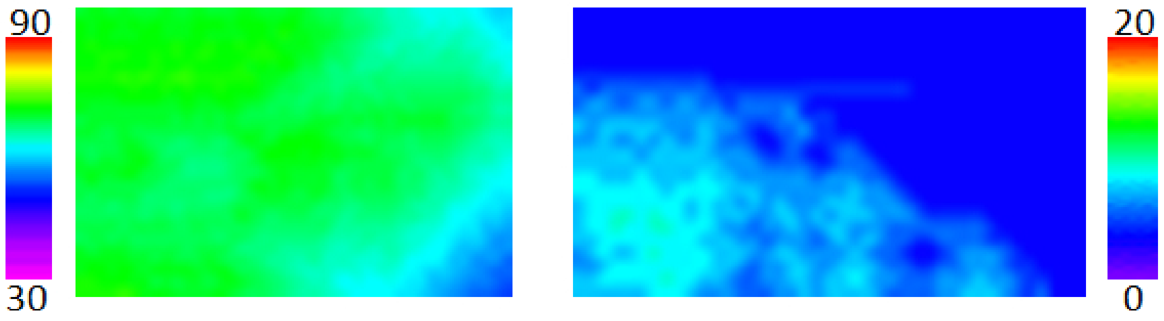
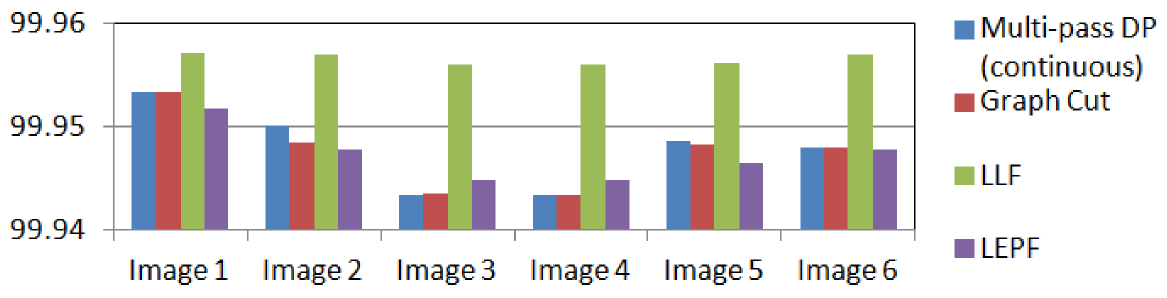
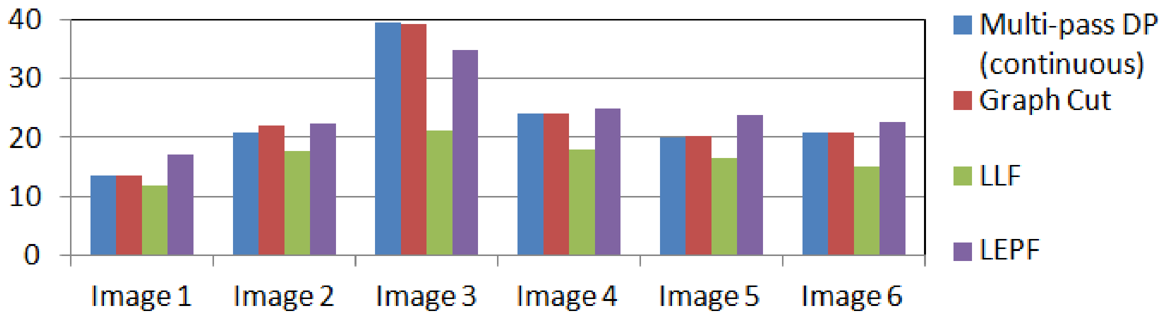


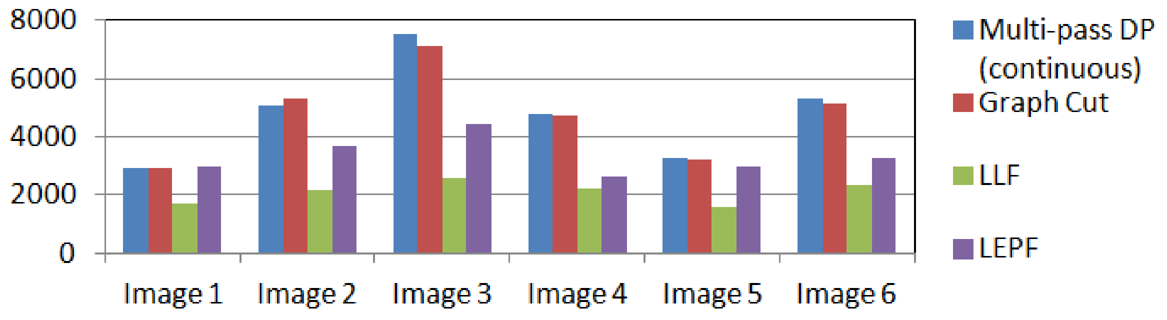
Figure 3.12: Seam surfaces obtained by continuous DP for Fig. 3.10 in the last iteration.



(a) Q_{MOS}

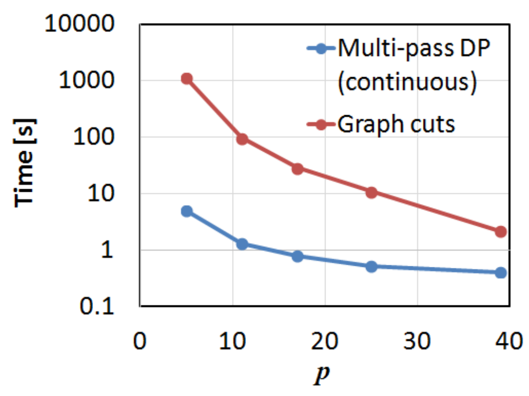


(b) Sharpness

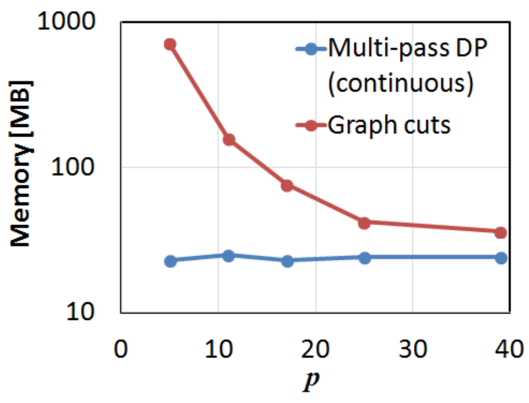


(c) Variance

Figure 3.13: Evaluation results for Fig. 3.10.



(a) Computation time



(b) Memory consumption

Figure 3.14: Computation time and memory consumption for image 5 in Fig. 3.10. The horizontal axis shows the subsampling parameter of the space.

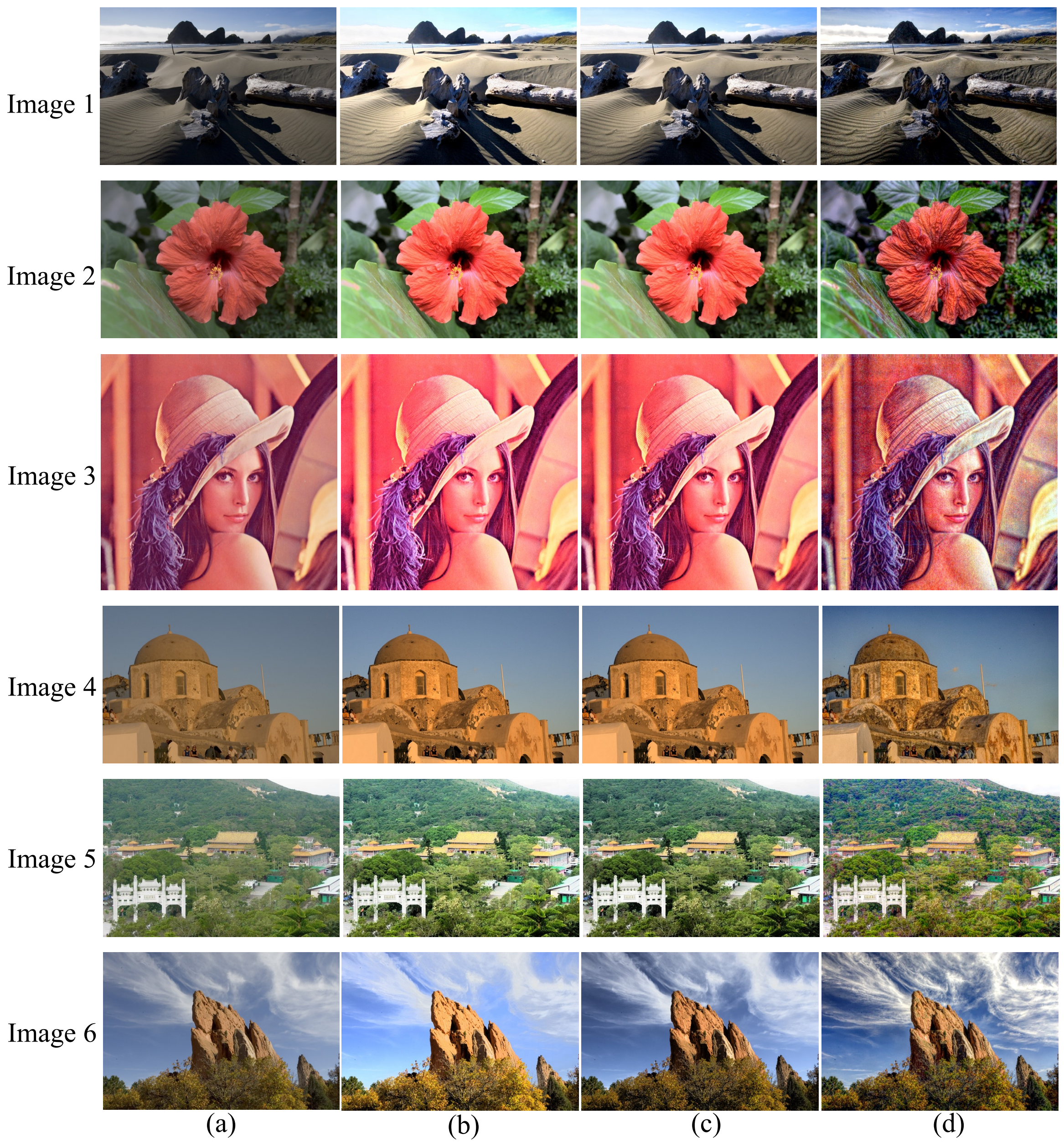


Figure 3.15: Contrast enhancement results. (a) Input, (b) multi-pass DP (continuous), (c) graph cuts, and (d) LLF [2].

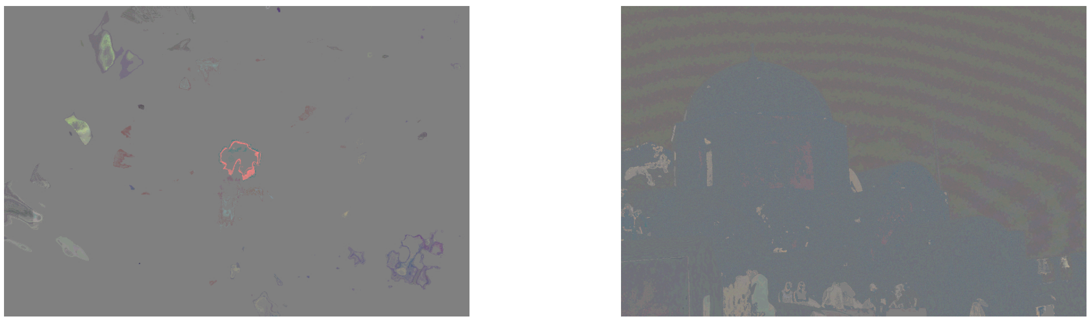


Figure 3.16: Difference images between Fig. 3.15 (b) and (c). The contrast was exaggerated by 8.

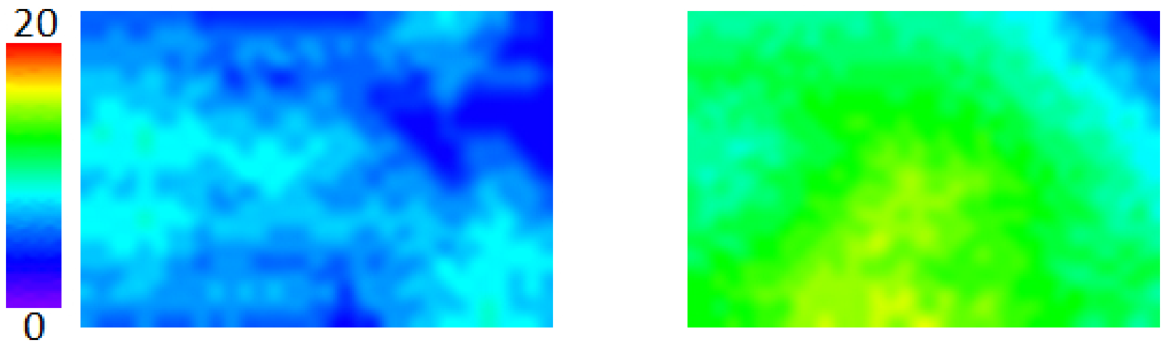
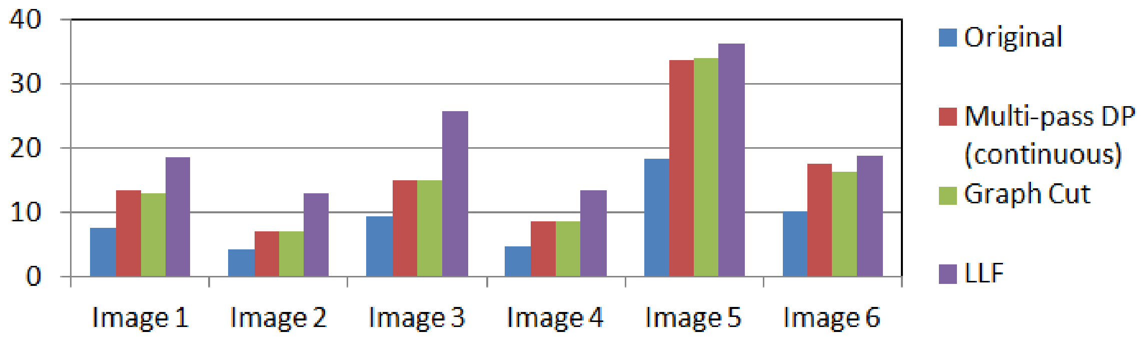
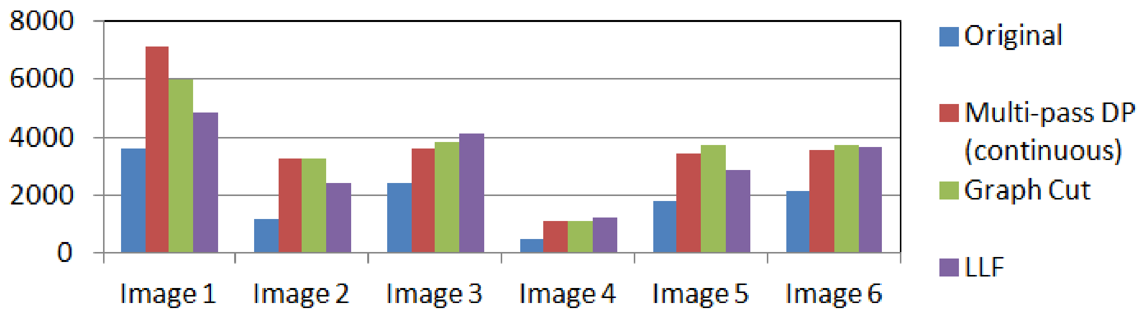


Figure 3.17: Seam surfaces obtained by continuous DP for Fig. 3.15 in the last iteration.



(a) Sharpness



(b) Variance

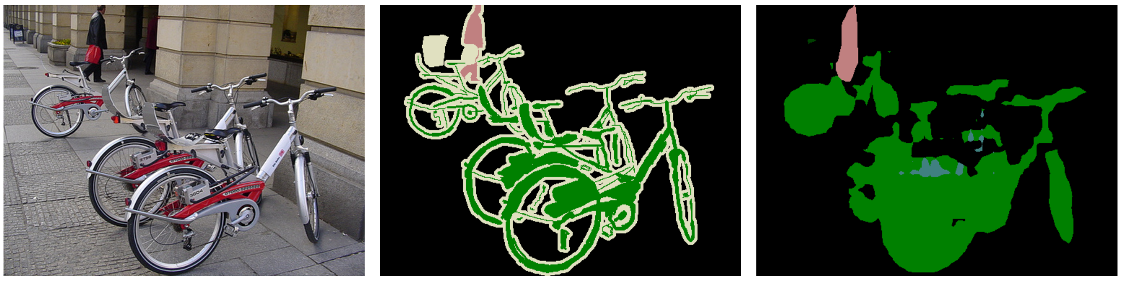
Figure 3.18: Evaluation results for Fig. 3.15.

Chapter 4

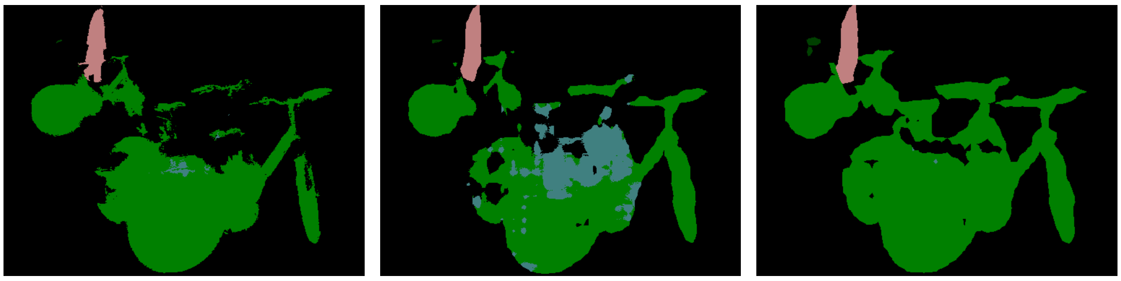
Learning Neighbors with Convolutional Neural Network

4.1 Introduction

Pixel labeling problems such as semantic segmentation and depth estimation are the tasks to assign a label to each pixel from the pre-defined label set (Fig. 4.1). The performance of pixel labeling problems has been improved significantly after the introduction of convolutional neural networks (CNN) [105–108]. Because of its simple architecture and high performance, fully convolutional network (FCN) [109, 110] that have only convolution and pooling layers (*i.e.*, no fully connected layer) has been widely used [111, 43, 112–116]. However, FCN still has two drawbacks. One is that the prediction accuracy around the object boundary is not good because the resolution of the output is much less than the input image due to the pooling layers. The other is that the relationships between each pixels are not considered because the label at each pixel is independently predicted by the last convolutional layer. To tackle these problems, Chen *et al.* [117] proposed to use densely connected conditional random field (dense CRF) [118] as a post-processing of FCN in semantic segmentation. The dense CRF takes the output from FCN as unary terms and considers every pair of pixels in a certain range. The penalty is imposed if the two different labels are assigned to the pair of pixels whose color or spatial distance is close. Because the optimization of dense CRF with mean field approximation is extremely efficient and significantly improves the accuracy, it has been used in many works [119–122]. However, FCN and the dense CRF are treated as completely independent modules. The parameters of dense CRF are manually decided by



(a) Input image (b) Ground truth (c) DeepLab-v2 [103]



(d) DeepLab-v2 w/ denseCRF [103] (e) CRFasRNN [104] (f) Proposed

back-ground	aeroplane	bicycle	bird	boat	bottle	bus
car	cat	chair	cow	table	dog	horse
motorbike	person	plant	sheep	sofa	train	tv/monitor

(g) Label set

Figure 4.1: An example of pixel labeling problems: semantic segmentation, where the labels are pre-defined object class names.

cross-validation while those of FCN are learned by back-propagation. To solve this drawback, Zheng *et al.* [104] found that the mean field algorithm for the dense CRF is composed of only filtering processes and proposed an end-to-end trainable model of FCN and dense CRF (named CRFasRNN). However, its pairwise terms can take only bilateral weights and the learnable parameters are still limited (label compatibility and weights of kernels). This drawback still remains although similar end-to-end trainable models of dense CRF and FCN are recently proposed for not only semantic segmentation but also depth estimation [123].

In this chapter, we propose a new model for pixel labeling problems, where FCN and dense CRF can be trained end-to-end. The proposed model is based on the interpretation that the fixed-point iteration for the general dense CRF mathematically equals to the recurrent convolution, which is shown in Sec. 4.3. The dense CRF in our model is more flexible than that in CRFasRNN [104] and can represent more general functions. In addition, our model can treat not only pairwise terms but also higher-order terms. Experimental results show that our method achieves comparable accuracy but much faster on the PASCAL VOC semantic segmentation benchmark.

4.2 Related Works

4.2.1 FCN and Dense CRF in Pixel Labeling Problems

Here, we focus on only the works that are relevant to dense CRF because so many pixel labeling methods have been proposed. Semantic segmentation has attracted large attention after the instruction of FCN [109, 110]. FCN is the network that does not have fully-connected layer and is composed of only convolution and pooling layers. Because of its simple structure and high performance, FCN is used as the bases of many works [111, 43, 112–116]. Chen *et al.* [117] showed that the accuracy is significantly improved by using dense CRF optimization [118] as a post-process of FCN. Desmaison *et al.* [124] proposed an efficient optimization method for the dense CRF with quadratic programming and linear programming by relaxing the discrete labels to continuous values. Ajanthan *et al.* [125] recently extend the work [124] and proposed a more accurate method by solving the dual problem of dense CRF. Although these methods can obtain more accurate results, they are slower than mean field approximation [118]. Moreover, the FCN and dense CRF are still independent modules.

Zheng *et al.* [104] regarded the mean field algorithm for dense CRF as the sequential filtering operations and proposed an end-to-end trainable model of FCN and dense CRF. However, its pairwise terms can take only bilateral weights and the learnable parameters are still limited as discussed in Sec. 4.1. Similar ideas have been also proposed by [126, 127]. Liu *et al.* [128] proposed a method that directly learns the unary and pairwise costs of dense CRF by using CNN, which is based on message passing algorithm. After that, Lin *et al.* [129] proposed an efficient piecewise training method for [128], which separately trains the unary and pairwise networks. Although these methods [128, 129] are more accurate than hand-crafted dense CRF, they do not scale to high resolution images because it is needed to concatenate every pair of intermediate features and input it to the pairwise network. Vemulapalli *et al.* [130] proposed an efficient optimization method for Gaussian dense CRF, where the unary and pairwise values are obtained as the output of CNN. Chandra *et al.* [131]

showed that the exact solution of the sparse CRF that is jointly learned with FCN is more accurate then the approximate solution of dense CRF. Different from the above methods that explicitly formulate only pairwise terms, our method can treat higher-order terms. Arnab *et al.* [132] extended CRFasRNN [104] to the model that can treat higher-order terms. However, their higher-order terms are hand-crafted, which imposes the penalty when the labeling result is inconsistent with the detection results of Faster-RCNN [133]. Therefore, it is different direction from our method that learns general higher-order terms.

The dense CRF is used in not only semantic segmentation but also continuous labeling problems such as denoising and depth estimation. Ristovski *et al.* [134] proposed an efficient optimization method for continuous dense CRF. Xu *et al.* [123] proposed a joint model of FCN and the continuous dense CRF for depth estimation. Vemulapalli *et al.* [135] proposed an end-to-end trainable model of Gaussian dense CRF and CNN for denoising. Knöbelreiter *et al.* [136] recently proposed a hybrid CNN-CRF model for stereo matching. These methods are based on only pairwise formulations. Wang *et al.* [137] showed that the proximal algorithm for structured labeling problems can be represented by recurrent convolution blocks and applied the end-to-end trainable model for denoising, depth refinement and optical flow estimation. Recently, the joint trainable model of dense CRF and FCN is applied to instance-aware semantic segmentation [138].

4.2.2 CNN and CRF in Other Applications

The joint trainable models of CRF and CNN have also been proposed in other applications (not pixel labeling problems): *e.g.*, human pose estimation [139–141], object detection [142, 143], image tag prediction [144], character recognition [145, 144], and multi-label classification [146, 147]. These methods do not scale to the problems where the numbers of nodes and edges are extremely large (*i.e.*, pixel labeling problems).

4.3 Proposed Method

4.3.1 Basic Model

First, we derive the mathematical relationship between the dense CRF and recurrent convolution. Our formulation is inspired by DeepMRF model [148], in which that between grid CRF and 2D RNN was derived. We start from the simple dense CRF model. Let $\boldsymbol{x}_u$ denote the RGB vector of the u-th pixel in the input image $\boldsymbol{x}$. The objective of pixel labeling problems is to assign a label $\boldsymbol{y}_u \in \mathcal{L}$ to each pixel $u(=1,\cdots,n)$. For notational simplicity, we introduce the vector $\boldsymbol{h}_u$. $\mathcal{L}$ is a continuous space (*e.g.*, $\mathcal{L} = [0,L]^m$ or $\mathbb{R}^m$), and $\boldsymbol{h}_u$ directly represents the continuous label (*i.e.*, $\boldsymbol{y}_u = \boldsymbol{h}_u$) when we consider continuous labeling problems such as depth or optical flow estimation. On the other hand, we have a pre-defined label set (*i.e.*, $\mathcal{L} = \{0,\cdots,L\}$), and $\boldsymbol{h}_u \in \{0,1\}^L$ is the indicator vector of the one-dimensional label $y_u \in \mathcal{L}$ when considering discrete labeling problems such as semantic segmentation: in other words, the l-th element of $\boldsymbol{h}_u$ is 1 when the label $y_u = l$ is assigned.

We define the dense CRF model as following:

$$p(\boldsymbol{x},\boldsymbol{h})=\frac{1}{Z}\prod_u\prod_{v\in\mathcal{N}_u}\phi(\boldsymbol{h}_u,\boldsymbol{x}_v)\prod_u\prod_{v\in\mathcal{N}_u}\psi(\boldsymbol{h}_u,\boldsymbol{h}_v)\prod_u\lambda(\boldsymbol{h}_u), \tag{4.1}$$

$$\phi(\boldsymbol{h}_u,\boldsymbol{x}_v)=\exp(\boldsymbol{h}_u^\top\boldsymbol{R}_{uv}\boldsymbol{x}_v), \tag{4.2}$$

$$\psi(\boldsymbol{h}_u,\boldsymbol{h}_v)=\exp(\boldsymbol{h}_u^\top\boldsymbol{W}_{uv}\boldsymbol{h}_v), \tag{4.3}$$

$$\lambda(\boldsymbol{h}_u)=\exp(-\boldsymbol{1}^\top\eta(\boldsymbol{h}_u)), \tag{4.4}$$

where $\phi(\boldsymbol{h}_u,\boldsymbol{x}_v)$ is the unary factor that represents the relationship between the label $\boldsymbol{h}_u$ and the neighbor pixel value $\boldsymbol{x}_v$. $\psi(\boldsymbol{h}_u,\boldsymbol{h}_v)$ is the pairwise factor that represents the relationship between the label $\boldsymbol{h}_u$ and the neighbor label $\boldsymbol{h}_v$. This dense CRF model considers the relationships between the pixel u and every neighbor pixel v in the certain range $\mathcal{N}_u$ centered at u. $\lambda(\boldsymbol{h}_u)$ is the regularization factor, and $\boldsymbol{1}$ is the vector with all elements being 1. η is the element-wise nonlinear function (see [148] for details). $\boldsymbol{R}_{uv}$ and $\boldsymbol{W}_{uv}$ are the learnable parameters, and Z is the partition function to normalize the sum to 1. This model is same as DeepMRF [148] except for the range of pairwise connection: DeepMRF [148] is the grid CRF that has connections with only 4 neighbors.

We obtain the optimal labeling $\hat{\boldsymbol{h}}$ by maximizing the probability in Eq. (4.1):

$$\hat{\boldsymbol{h}} = \arg\max_{\boldsymbol{h}} p(\boldsymbol{x},\boldsymbol{h}). \tag{4.5}$$

As described above, $\boldsymbol{h}_u$ is a discrete vector and defined as Eq. (4.6) in discrete labeling problems:

$$\boldsymbol{h}_u \in \{0,1\}^L,\ \sum_l h_u(l) = 1. \tag{4.6}$$

We relax the $\boldsymbol{h}$ to continuous vector as Eq. (4.7) in order to make the inference and training tractable,

$$\boldsymbol{h}_u \in [0,1]^L, \sum_l h_u(l) = 1, \tag{4.7}$$

and the final labeling is obtained as Eq. (4.8):

$$y_u = \arg\max_l \hat{h}_u(l). \tag{4.8}$$

We solve the maximization problem in Eq. (4.5) with the similar way to [148]. First, we focus on the pixel u and marginalize Eq. (4.1) as following:

$$p(\boldsymbol{h}_u | \boldsymbol{x}_{\mathcal{N}_u}, \boldsymbol{h}_{\mathcal{N}_u}) \propto \lambda(\boldsymbol{h}_u) \prod_{v \in \mathcal{N}_u} \phi(\boldsymbol{h}_u, \boldsymbol{x}_v) \prod_{v \in \mathcal{N}_u} \psi(\boldsymbol{h}_u, \boldsymbol{h}_v) \tag{4.9}$$

$$= \exp\left(-\mathbf{1}^\top \eta(\boldsymbol{h}_u) + \sum_{v \in \mathcal{N}_u} \boldsymbol{h}_u^\top \boldsymbol{R}_{uv} \boldsymbol{x}_v + \sum_{v \in \mathcal{N}_u} \boldsymbol{h}_u^\top \boldsymbol{W}_{uv} \boldsymbol{h}_v \right). \tag{4.10}$$

Next, we differentiate Eq. (4.10) by $\boldsymbol{h}_u$,

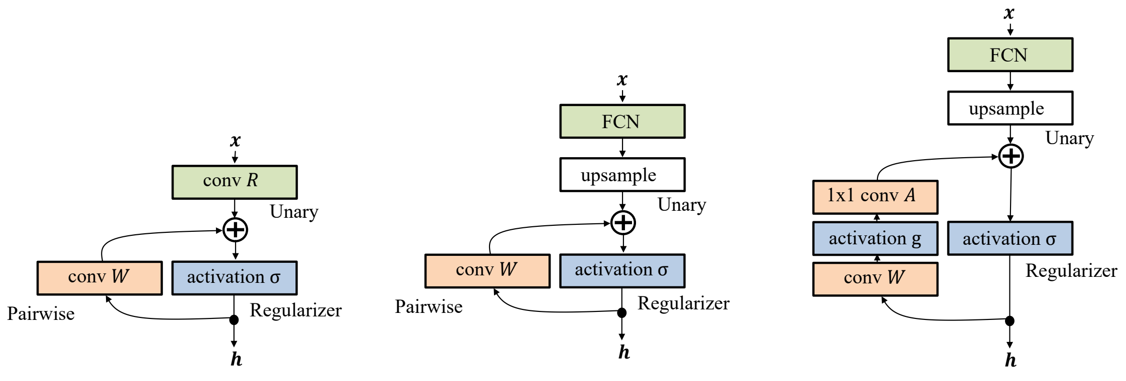
$$\frac{\partial p}{\partial \boldsymbol{h}_u} = \exp(\sim) \left(-\mathbf{1}^\top \eta'(\boldsymbol{h}_u) + \sum_{v \in \mathcal{N}_u} \boldsymbol{R}_{uv} \boldsymbol{x}_v + \sum_{v \in \mathcal{N}_u} \boldsymbol{W}_{uv} \boldsymbol{h}_v \right), \tag{4.11}$$

due to limitations of space, we omitted the content in the blanket of exp function. Eq. (4.12) is derived by making the right hand of Eq. (4.11) equal to 0,

$$\boldsymbol{h}_u = \sigma\left(\sum_{v \in \mathcal{N}_u} \boldsymbol{R}_{uv} \boldsymbol{x}_v + \sum_{v \in \mathcal{N}_u} \boldsymbol{W}_{uv} \boldsymbol{h}_v \right), \tag{4.12}$$

where we put $\sigma^{-1}(z) = \eta'(z)$. Wu *et al.* [148] proposed a way to update the $\boldsymbol{h}_u$ by running 2D RNN on image grid because their model is based on the grid CRF. However, that way cannot be applied to our model because our model is the dense CRF where every pair of pixels has a connection in a certain range. Therefore, we obtain the optimal $\boldsymbol{h}$ by fixed-point iteration as Eq. (4.13):

$$\boldsymbol{h}_u^{(t+1)} = \sigma\left(\sum_{v \in \mathcal{N}_u} \boldsymbol{R}_{uv} \boldsymbol{x}_v + \sum_{v \in \mathcal{N}_u} \boldsymbol{W}_{uv} \boldsymbol{h}_v^{(t)} \right), \tag{4.13}$$



(a) Basic model (b) unary FCN (c) Higher-order model

Figure 4.2: Our model can be interpreted as recurrent convolution. $\boldsymbol{x}$ is the input image and $\boldsymbol{h}$ is the output labeling.

where $\boldsymbol{h}_u^{(t)}$ is the probability vector $\boldsymbol{h}_u$ at t-th iteration. Different from the DeepMRF [148] where $\boldsymbol{h}_u (u = 1, \cdots, n)$ is updated sequentially, we update $\boldsymbol{h}_u (u = 1, \cdots, n)$ simultaneously at all pixels. The second term of the right hand in Eq. (4.13) can be regarded as a convolution because $\mathcal{N}_u$ is the certain range centered at u. σ can be regarded as an activation function. Consequently, our model can be interpreted as recurrent convolution as shown in Fig. 4.2a. The first convolution with $\boldsymbol{R}$ works to predict unary scores, and the second one with $\boldsymbol{W}$ refines $\boldsymbol{h}$ by considering pairwise relations in the convolution range $\mathcal{N}$.

4.3.2 FCN as Unary Network

Our basic model in Fig. 4.2a has one convolution with $\boldsymbol{R}$ as unary part because we started from simple dense CRF in Sec. 4.3.1. However, one convolution layer is generally not enough to predict the labels. Therefore, we replace the convolution layer $\boldsymbol{R}$ with the powerful FCN for more accurate prediction as shown in Fig. 4.2b. In that case, the unary term in Eq. (4.2) is replaced with Eq. (4.14), and the update function is represented as Eq. (4.15):

$$\phi(\boldsymbol{h}_u, \boldsymbol{x}) = \exp(\boldsymbol{h}_u^\top \boldsymbol{f}_u^{FCN}(\boldsymbol{x})), \tag{4.14}$$

$$\boldsymbol{h}_u^{(t+1)} = \sigma\left(\boldsymbol{f}_u^{FCN}(\boldsymbol{x}) + \sum_{v \in \mathcal{N}_u} \boldsymbol{W}_{uv} \boldsymbol{h}_v^{(t)}\right), \tag{4.15}$$

where $\boldsymbol{f}_u^{FCN}(\boldsymbol{x})$ is the unnormalized unary score predicted by FCN. Because the resolutions of the output maps from FCN are small, we upsample them to the original resolution by simple linear interpolation. In our experiments (Sec. 4.4), we use DeepLab-v2 [103], which is a kind of FCNs and more accurate than the original FCN [109, 110] on semantic segmentation.

4.3.3 Extension to Higher-Order Term

Although our formulation starts from the pairwise dense CRF, we can extend the proposed model to higher-order CRF. As shown in Fig. 4.2c, we simply add the element-wise nonlinear function g and 1x1 convolution. By this addition, the pairwise term in Eq. (4.3) and the update function in Eq. (4.15) are extended as Eq. (4.16) and Eq. (4.17), respectively:

$$\psi(\boldsymbol{h}_{\mathcal{N}_u}) = \exp\left(\boldsymbol{h}_u^\top \boldsymbol{A}\, g\left(\sum_{v\in\mathcal{N}_u} \boldsymbol{W}_{uv}\boldsymbol{h}_v\right)\right), \tag{4.16}$$

$$\boldsymbol{h}_u^{(t+1)} = \sigma\left(\boldsymbol{f}_u^{FCN}(\boldsymbol{x}) + \boldsymbol{A}\, g\left(\sum_{v\in\mathcal{N}_u} \boldsymbol{W}_{uv}\boldsymbol{h}_v^{(t)}\right)\right), \tag{4.17}$$

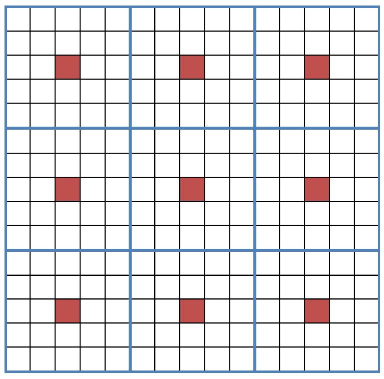
where $\boldsymbol{A}$ is the learnable parameter of 1x1 convolution. Eq. (4.16) is a higher-order term because it has the nonlinear function g and cannot be decomposed to the sum of pairwise terms. Eq. (4.16) has more expressive power than the pairwise term in Eq. (4.3) because the $\boldsymbol{h}_u$ is predicted from the higher-order interaction around u.

4.3.4 Large Field of View (FoV)

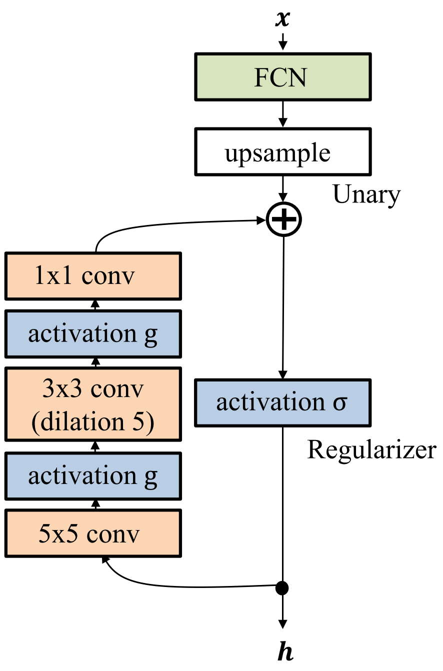
In order to make the higher-order term in Eq. (4.16) more powerful, we employ the dilated convolution [149] as shown in Fig. 4.3a. Our large FoV model is as shown in Fig. 4.3b, which has a 5x5 conv layer followed by a 3x3 conv layer with dilation 5. The higher-order term in our model has large field of view (15x15) by these two convolution layers with less number of parameters than that of a 15x15 convolution layer.

4.3.5 Coarse-to-Fine Strategy

In our model (Fig. 4.3b), we upsample the output map from DeepLab by the factor 8 and input it to the recurrent convolution part. For more accurate prediction, we take



(a) Dilated convolution



(b) Large FoV model

Figure 4.3: To make the field of view large, we employ the dilated convolution as (a). The size of first convolution layer is 5x5 (blue area), and the second convolution layer is 3x3 with dilation 5 (red pixels). The large area (15x15) is covered by the two convolutions with less number of parameters than that of one 15x15 convolution layer. Our large FoV model is as (b).

the coarse-to-fine strategy as shown in Fig. 4.4. The output map from DeepLab is 2x upsampled and refined by the recurrent convolution part. This process is repeated until the original resolution. The recurrent convolution part at higher resolution can capture fine local patterns while it has small FoV. At lower resolution it has larger FoV while it can capture only rough local patterns. We show this coarse-to-fine strategy leads to more accurate results in Sec. 4.4.

4.4 Experimental Results

4.4.1 Dataset

We conducted experiments on the PASCAL VOC 2012 segmentation benchmark, which has 20 object classes and background (total 21 classes). The original dataset has only 1,464 train images. Therefore, similar to [103], we used 10,582 annotated images for training, which was provided by [150]. We used 1,449 validation images for test. For evaluation metric, we used intersection-over-union (IoU) that are official metric on the dataset and class accuracy that are shown in recent papers [109, 124, 125, 129, 110].

4.4.2 Implementation Details

We implemented the proposed model based on DeepLab-v2 [103] using the Caffe library [151] on a Tesla K80 GPU with 12GB memory. The number of output channels of 5x5 convolution is 128, and that of 3x3 convolution is 256. We used the batch normalization just after each of these two convolution layers. We adopted softmax as the activation function σ and ReLU as the nonlinear function g in Eq. (4.16). Although the original DeepLab-v2 [103] uses three ResNets [152] for multi-scale training and testing, we used only one ResNet due to the GPU memory limitation. Similar to [103], we employed the poly learning, where the learning rate started from 2.5×10^{-4} and multiplied by $(1 - (\frac{iter}{max\_iter})^{power})$ at each iteration. We set the *max_iter* to 20,000, *power* to 0.9, momentum to 0.9, and weight decay to 5.0×10^{-4}. When training our model, we adopted the piecewise training: the unary network (DeepLab-v2) was first trained, and then the unary network and the recurrent convolution part were trained end-to-end. We used the model pre-trained on ImageNet as the initial weights of the unary network. As a reference, we also report the results when the unary network was frozen and only the recurrent convolution part was trained. We set the number of iteration of the recurrent convolution part to one. Although we tried more iterations, it little improved the performance. We used the pixel-wise cross-entropy between the groundtruth label $\bar{y}_u$ and $\boldsymbol{h}_u$ defined as following:

$$loss(\bar{\boldsymbol{y}}, \boldsymbol{h}) = \frac{1}{n} \sum_u \sum_j [\bar{y}_u = j] \log h_u(j), \tag{4.18}$$

where $[\cdot]$ is the indicator function that is 1 if the statement in the blanket is true and 0 otherwise.

4.4.3 Results

We show the quantitative comparison of the unary network, unary network with CRFasRNN [104], and unary network with the proposed model in Table 4.1. We observe that the performance of proposed model is a little better than the only unary network in terms of both mean accuracy and IoU. When the proposed model is

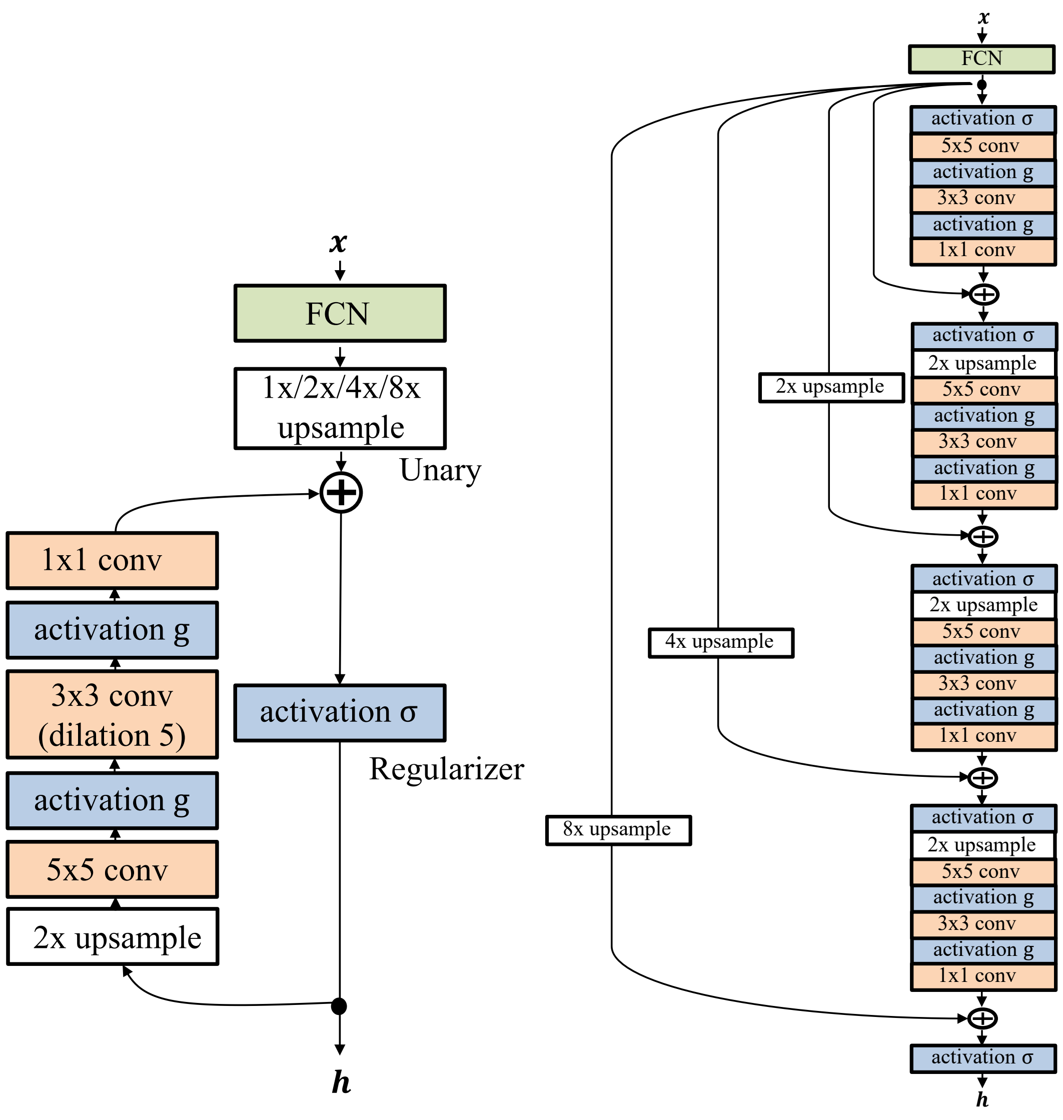
Table 4.1: Quantitative comparison on the PASCAL VOC 2012 segmentation benchmark. The unary network is ResNet-based DeepLab-v2.

method	end-to-end	coarse-to-fine	mean acc.	mean IoU	feedforward time [ms]
unary (ResNet)			81.8	74.6	296
unary + denseCRF [103]			82.3	75.6	3710
unary + CRFasRNN [104]	✓		**83.7**	**76.1**	1144
unary + proposed			82.4	74.9	414
unary + proposed	✓		82.6	75.5	
unary + proposed	✓	✓	**83.7**	**76.1**	454

trained end-to-end, the performance is further improved. The coarse-to-fine strategy further improves the performance, and the coarse-to-fine model is comparable or a little better than the hand-crafted dense CRF [103]. The proposed models are comparable with the CRFasRNN [104] in terms of both mean accuracy and IoU. However, in terms of the computational time for feedforward, the proposed model is much faster than CRFasRNN: $(1144-296)/(454-296) = 848/158 \approx 5.4$x faster[1]. The reason is because the proposed model can benefit from the parallelization by GPU while the CRFasRNN cannot.

We show the IoU of each class in Table 4.2 and the qualitative comparison in Fig. 4.5. We observe that the proposed method successfully labels to the small objects such as the plant, the legs of the bird, and the cat in the cat's house.

[1]The run time was measured for 500x450 pixel images due to the GPU memory limitation.



(a) Coarse-to-fine (b) Unfolded

Figure 4.4: Coarse-to-fine strategy: the output map from DeepLab is 2x upsampled and refined by the recurrent convolution part. This process is repeated until the original resolution. The recurrent part in (a) is unfolded as (b).

Table 4.2: IoU of each class. The unary network is ResNet-based DeepLab-v2.

method	end-to-end	coarse-to-fine	mean	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	dinigtable	dog	horse	motorbike	person	potted-plant	sheep	sofa	train	tv/monitor
unary			74.6	88.2	39.6	85.4	64.4	80.2	93.6	85.5	89.9	32.3	84.9	49.3	84.4	85.6	81.8	84.6	57.8	84.1	42.8	86.9	72.8
unary + denseCRF [103]			75.6	**90.1**	40.1	**86.7**	65.6	81.1	94.0	86.0	91.0	33.0	85.7	50.0	85.5	86.7	82.7	**85.9**	58.8	85.4	43.8	87.1	73.9
unary + CRFasRNN [104]	✓		**76.1**	89.0	41.1	85.3	65.4	81.1	94.2	86.1	**91.5**	34.6	**87.4**	**55.3**	**86.7**	86.0	82.9	85.8	58.5	85.2	**46.2**	**87.7**	**74.5**
unary + proposed			74.9	88.1	39.7	85.4	64.3	80.4	93.6	85.4	90.1	32.6	85.2	50.4	84.5	85.8	82.1	84.9	58.3	84.5	43.9	87.0	73.2
unary + proposed	✓		75.5	89.9	**41.8**	86.3	65.6	**81.3**	**94.5**	86.3	91.1	33.6	85.7	50.6	85.6	86.1	**83.0**	85.2	58.9	84.3	43.4	87.4	70.5
unary + proposed	✓	✓	**76.1**	87.4	41.4	85.3	**70.0**	80.0	94.4	**86.5**	91.2	**35.7**	86.6	54.8	85.3	**86.8**	82.9	85.7	**59.0**	**86.0**	43.4	87.0	74.0

Table 4.3: Quantitative comparison on the PASCAL VOC 2012 segmentation benchmark. The unary network is VGG-based DeepLab-v2.

method	end-to-end	coarse-to-fine	mean acc.	mean IoU	feedforward time [ms]
unary (VGG)			81.0	68.7	168
unary + denseCRF [103]			**81.3**	**71.2**	3582
unary + proposed			81.1	68.7	289
unary + proposed	✓		80.3	69.3	
unary + proposed	✓	✓	80.1	69.3	332

Fig. 4.6 shows the failure cases of the proposed method. The proposed method predicts the legs of the horse by considering the spatial relationships with the body although they are occluded. In the second case, the proposed method labels the person in the tv monitor because the situation where a person appears in a tv monitor is not unnatural.

4.4.4 Discussion

Table 4.3 shows the results when the unary network is the VGG-based DeepLab-v2. Similar to [103], we set the initial learning rate to 1.0×10^{-3} and the batch size to 10 when training the VGG-based Deeplab. We set the batch size to 5 when training the single-scale proposed model, and set it to 4 when training the coarse-to-fine proposed model due to the GPU memory limitation. We observe that the proposed model cannot improve the performance, compared with the only unary network. That is consistent with [153], where the authors reported that their model could not be successfully trained from the output of the unary network (VGG-based FCN [109]) because it was too noisy. When the unary network is highly accurate (ResNet-based), the proposed model can be trained well as shown in Table 4.1. Therefore, we believe that the proposed model can possibly improve the performance further when the unary network is more accurate (*e.g.*, [115]).

4.5 Conclusions

In this chapter, we proposed a joint model of FCN and dense CRF, which is derived from the fixed-point iteration of the dense CRF. We employed some techniques such as enlarging the field of view with dilated convolution and coarse-to-fine strategy in order to improve the performance. Experimental results on semantic segmentation benchmark show that the performance of the proposed method is comparable with the CRFasRNN. However, the proposed method is yet another approach of joint model of FCN and dense CRF, which is much faster and simple. Compared with the CRFasRNN, the proposed method is easy to implement because it is composed of only fundamental layers in deep learning libraries. Applying the proposed model to other applications such as optical flow or depth estimation is a future work.

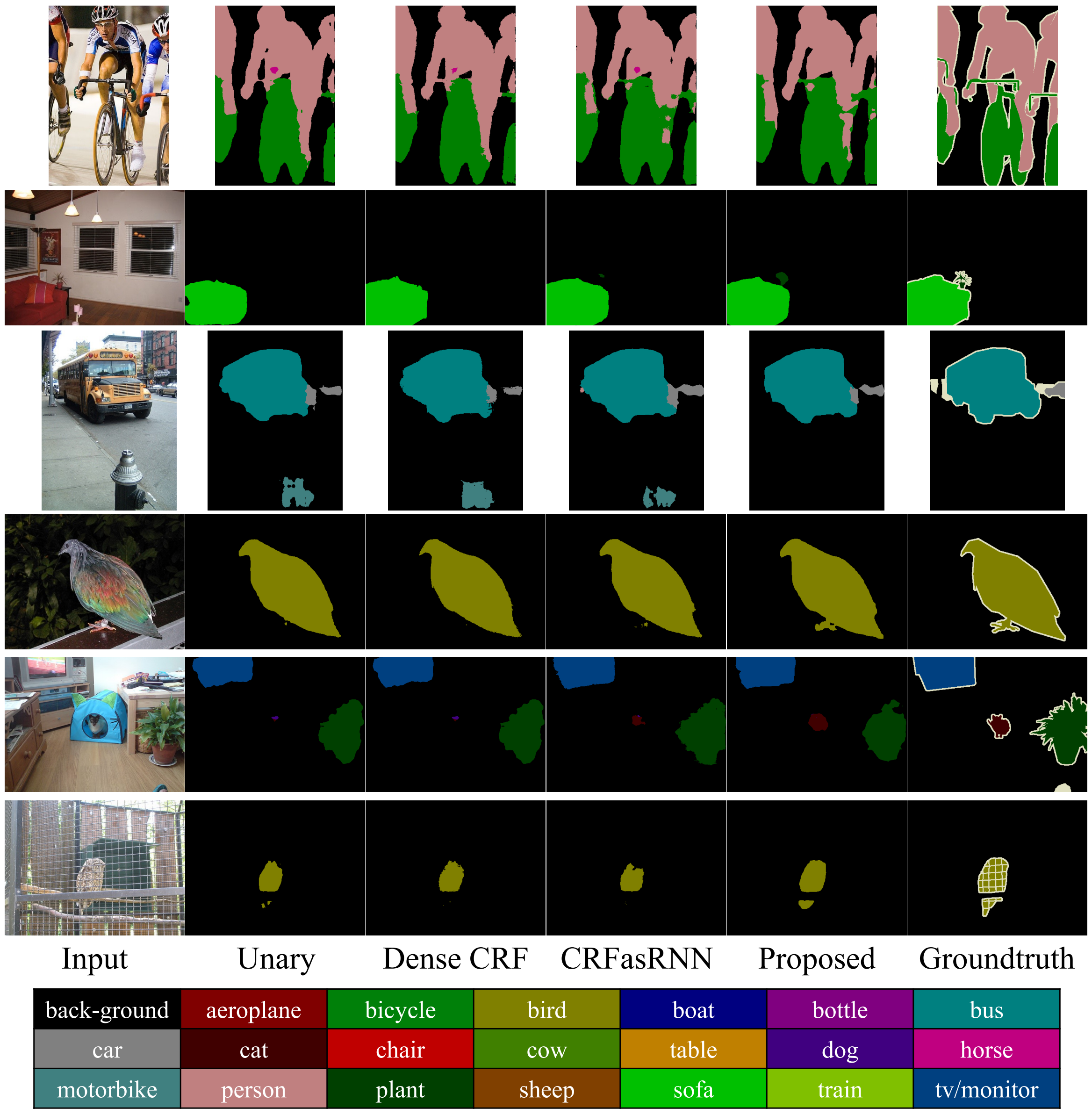
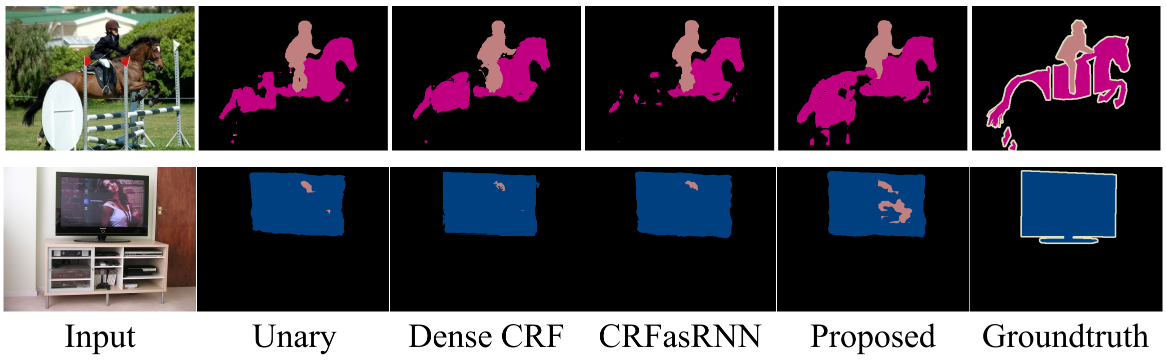


Figure 4.5: Successful cases of the proposed method on the PASCAL VOC 2012 segmentation benchmark. The unary network is ResNet-based DeepLab-v2.



back-ground	aeroplane	bicycle	bird	boat	bottle	bus
car	cat	chair	cow	table	dog	horse
motorbike	person	plant	sheep	sofa	train	tv/monitor

Figure 4.6: Failure cases of the proposed method.

Chapter 5

Learning Neighbors with Deep Reinforcement Learning

5.1 Introduction

After the introduction of the deep Q-network (DQN) [154], which can play Atari games on the human level, much attention has been focused on deep reinforcement learning (RL). Recently, deep RL is also applied to a variety of image processing tasks [155–157]. However, these methods can execute only global actions for the entire image and are limited to simple applications, *e.g.*, image cropping [155] and global color enhancement [157, 158]. Therefore, these methods cannot be applied to applications that require pixel-wise manipulations such as image denoising.

To overcome this drawback, we propose a new problem setting: pixelRL for image processing. PixelRL is a multi-agent RL problem, where the number of agents is equal to that of pixels. The agents learn the optimal behavior to maximize the mean of the expected total rewards at all pixels. Each pixel value is regarded as the current state and is iteratively updated by the agent's action. Applying the existing techniques of the multi-agent RL to pixelRL is impractical in terms of computational cost because the number of agents is extremely large. Therefore, we solve the problem by employing the fully convolutional network (FCN). The merit of using FCN is that all the agents can share the parameters and learn efficiently. Herein, we also propose *reward map convolution*, which is an effective learning method for pixelRL. By the proposed reward map convolution, each agent considers not only the future states of its own pixel but also those of the neighbor pixels.

The proposed pixelRL is applied to image denoising, image restoration, and local color enhancement. To the best of our knowledge, this is the first work to apply RL to such low-level image processing for each pixel or each local region. Our experimental results show that the agents trained with the pixelRL and the proposed reward map convolution achieve comparable or better performance, compared with state-of-the-art methods based on supervised learning. Although the actions must be pre-defined for each application, the proposed method is interpretable by observing the actions executed by the agents, which is a novel and different point from the existing deep learning-based image processing methods for such applications. The interpretability is important for some applications such as medical image processing as discussed in [159].

Our contributions are summarized as follows:

- We propose a novel problem setting: pixelRL for image processing, where the existing techniques for multi-agents RL cannot be applied.
- We propose *reward map convolution*, which is an effective learning method for pixelRL and boosts the performance.
- We apply the pixelRL to image denoising, image restoration, and local color enhancement. The proposed method is a completely novel approach for these tasks, and shows better or comparable performance, compared with state-of-the-art methods.
- The actions executed by the agents are interpretable to humans, which is of great difference from conventional CNNs.

5.2 Related Works

5.2.1 Deep RL for Image Processing

Very recently, deep RL has been used for some image processing applications. Cao et al. [160] proposed a super-resolution method for face images. The agent first chooses a local region and inputs it to the local enhancement network. The enhancement network converts the local patch to a high-resolution one, and the agents chooses the next local patch that should be enhanced. This process is repeated until the maximum time step; consequently, the entire image is enhanced. Li et al. [155] used deep RL for image cropping. The agent iteratively reshapes the cropping window to maximize the aesthetics score of the cropped image. Yu et al. [161] proposed the RL-restore method, where the agent selects a toolchain from a toolbox (a set of light-weight CNNs) to restore a corrupted image. Park et al. [157] proposed a color enhancement method using DQN. The agent iteratively chooses the image manipulation action (*e.g.*, increase brightness) and retouches the input image. The reward is defined as the negative distance between the retouched image by the agent and the one by an expert. A similar idea is proposed by Hu et al. [158], where the agent retouches from RAW images. As discussed in the introduction, all the above methods execute global actions for entire images. In contrast, we tackle pixelRL, where pixel-wise actions can be executed.

Wulfmeier et al. [162] used the FCN to solve the inverse reinforcement learning problem. This problem setting is different from ours because one pixel corresponds to one state, and the number of agents is one in their setting. In contrast, our pixelRL has one agent at each pixel.

5.2.2 Image Denoising

Image denoising methods are classified into two categories: non-learning and learning based. Many classical methods are categorized into the former class (*e.g.*, BM3D [163]). Although learning-based methods include dictionary-based methods such as [164], the recent trends in image denoising is neural network-based methods [4, 165]. Generally, neural-network-based methods have shown better performances, compared with non-leaning-based methods.

Our denoising method based on pixelRL is a completely different approach from other neural network-based methods. While most of neural-network-based methods learn to regress noise or true pixel values from a noisy input, our method iteratively removes noise with the sequence of simple pixel-wise actions (basic filters).

5.2.3 Image Restoration

Similar to image denoising, image restoration (also called image inpainting) methods are divided into non-learning and learning-based methods. In the former methods such as [166], the target blank regions are filled by propagating the pixel values or gradient information around the regions. The filling process is highly sophisticated, but they are based on a handcrafted-algorithm. Roth and Black [167] proposed a Markov random field-based model to learn the image prior to the neighbor pixels. Mairal et al. [168] proposed a learning-based method that creates a dictionary from an image database using K-SVD, and applied it to image denoising and inpainting. Recently, deep-neural-network-based methods were proposed [169, 5], and the U-Net-based inpainting method [5] showed much better performance than other methods.

Our method is categorized into the learning-based method because we used training images to optimize the policies. However, similar to the classical inpainting methods, our method successfully propagates the neighbor pixel values with the sequence of basic filters.

5.2.4 Color Enhancement

One of the classical methods is color transfer proposed by Reinhard et al. [170], where the global color distribution of the reference image is transfered to the target image. Hwang et al. [171] proposed an automatic local color enhancement method based on image retrieval. This method enhances the color of each pixel based on the retrieved images with smoothness regularization, which is formulated as a Gaussian MRF optimization problem.

Yan et al. [6] proposed the first color enhancement method based on deep learning. They used a DNN to learn a mapping function from the carefully designed pixel-wise features to the desired pixel values. Gharbi et al. [172] used a CNN as a trainable bilateral filter for high-resolution images and applied it to some image processing tasks. Similarly, for fast image processing, Chen et al. [173] adopted an FCN to learn an approximate mapping from the input to the desired images. Unlike deep learning-based methods that learn the input for an output mapping, our color enhancement method is interpretable because our method enhances each pixel value iteratively with actions such as [157, 158].

5.3 Background Knowledge

Herein, we extend the asynchronous advantage actor-critic (A3C) [174] for the pixelRL problem because A3C showed good performance with efficient training in the original paper[1]. In this section, we briefly review the training algorithm of A3C. A3C is one of the actor-critic methods, which has two networks: policy network and value network. We denote the parameters of each network as θ_p and θ_v, respectively. Both networks use the current state $s^{(t)}$ as the input, where $s^{(t)}$ is the state at time step

[1]Note that we can employ any deep RL methods such as DQN instead of A3C.

t. The value network outputs the value $V(s^{(t)})$: the expected total rewards from state $s^{(t)}$, which shows how good the current state is. The gradient for θ_v is computed as follows:

$$R^{(t)} = r^{(t)} + \gamma r^{(t+1)} + \gamma^2 r^{(t+2)} + \cdots + \gamma^{n-1} r^{(t+n-1)} + \gamma^n V(s^{(t+n)}), \tag{5.1}$$

$$d\theta_v = \nabla_{\theta_v} \left(R^{(t)} - V(s^{(t)}) \right)^2, \tag{5.2}$$

where γ^i is the i-th power of the discount factor γ.

The policy network outputs the policy $\pi(a^{(t)}|s^{(t)})$ (probability through softmax) of taking action $a^{(t)} \in \mathcal{A}$. Therefore, the output dimension of the policy network is $|\mathcal{A}|$. The gradient for θ_p is computed as follows:

$$A(a^{(t)}, s^{(t)}) = R^{(t)} - V(s^{(t)}), \tag{5.3}$$

$$d\theta_p = -\nabla_{\theta_p} \log \pi(a^{(t)}|s^{(t)}) A(a^{(t)}, s^{(t)}). \tag{5.4}$$

$A(a^{(t)}, s^{(t)})$ is called the advantage, and $V(s^{(t)})$ is subtracted in Eq. (5.3) to reduce the variance of the gradient. For more details, see [174].

5.4 Reinforcement Learning with Pixel-wise Rewards (PixelRL)

Here, we describe the proposed pixelRL problem setting. Let I_i be the i-th pixel in the input image $\boldsymbol{I}$ that has N pixels ($i = 1, \cdots, N$). Each pixel has an agent, and its policy is denoted as $\pi_i(a_i^{(t)}|s_i^{(t)})$, where $a_i^{(t)} (\in \mathcal{A})$ and $s_i^{(t)}$ are the action and the state of the i-th agent at time step t, respectively. $\mathcal{A}$ is the pre-defined action set, and $s_i^{(0)} = I_i$. The agents obtain the next states $\boldsymbol{s}^{(t+1)} = (s_1^{(t+1)}, \cdots, s_N^{(t+1)})$ and rewards $\boldsymbol{r}^{(t)} = (r_1^{(t)}, \cdots, r_N^{(t)})$ from the environment by taking the actions $\boldsymbol{a}^{(t)} = (a_1^{(t)}, \cdots, a_N^{(t)})$. The objective of the pixelRL problem is to learn the optimal policies $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_N)$ that maximize the mean of the total expected rewards at all pixels:

$$\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{\pi}} \mathbb{E}_{\boldsymbol{\pi}} \left(\sum_{t=0}^{\infty} \gamma^t \bar{r}^{(t)} \right), \tag{5.5}$$

$$\bar{r}^{(t)} = \frac{1}{N} \sum_{i=1}^{N} r_i^{(t)}, \tag{5.6}$$

where $\bar{r}^{(t)}$ is the mean of the rewards $r_i^{(t)}$ at all pixels.

A naive solution for this problem is to train a network that output Q-values or policies for all possible set of actions $\boldsymbol{a}^{(t)}$. However, it is computationally impractical because the dimension of the last fully connected layer must be $|\mathcal{A}|^N$, which is too large.

Another solution is to divide this problem into N independent subproblems and train N networks, where we train the i-th agent to maximize the expected total reward at the i-th pixel:

$$\pi_i^* = \arg\max_{\pi_i} \mathbb{E}_{\pi_i}\left(\sum_{t=0}^{\infty} \gamma^t r_i^{(t)}\right). \tag{5.7}$$

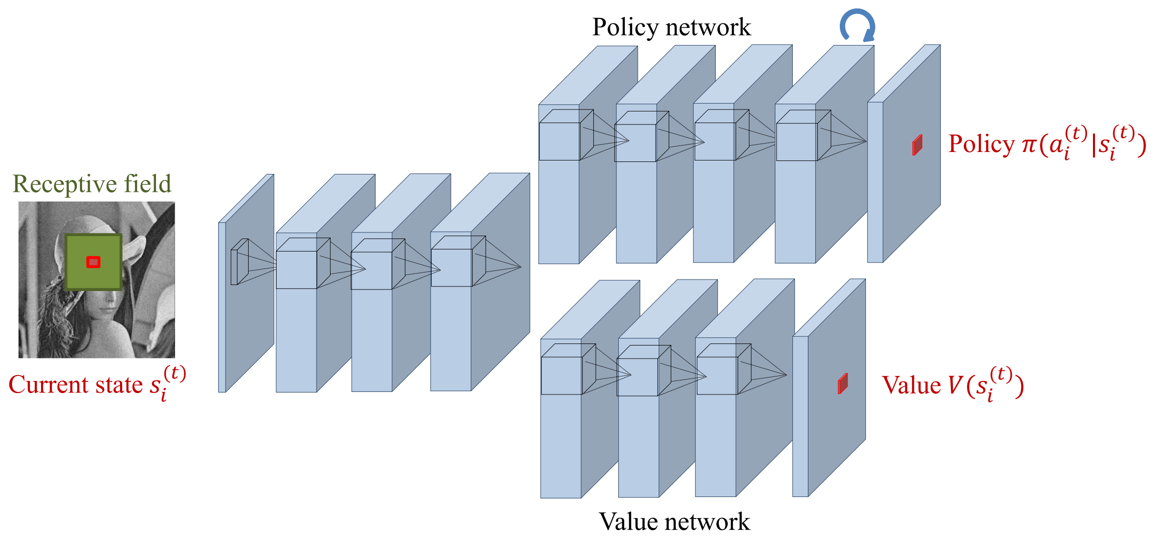
However, training N networks is also computationally impractical when the number of pixels is large. In addition, it treats only the fixed size of images. To solve the problems, we employ a FCN instead of N networks. By using the FCN, all the N agents can share the parameters, and we can parallelize the computation of N agents on a GPU, which renders the training efficient. Herein, we employ A3C and extend it to the fully convolutional form. Our architecture is illustrated in Fig. 5.1.

The pixelRL setting is different from typical multi-agent RL problems in terms of two points. The first point is that the number of agents N is extremely large ($> 10^5$). Therefore, typical multi-agent learning techniques such as [175] cannot be directly applied to the pixelRL. Next, the agents are arrayed in a 2D image plane. In the next section, we propose an effective learning method that boosts the performance of the pixelRL agents by leveraging this property, named *reward map convolution*.

5.5 Reward Map Convolution

Here, for the ease of understanding, we first consider the one-step learning case (*i.e.*, $n = 1$ in Eq. (5.1)).

When the receptive fields of the FCNs are 1x1 (*i.e.*, all the convolution filters in the policy and value network are 1x1), the N subproblems are completely independent. In that case, similar to the original A3C, the gradient of the two networks are



Shared network				Policy network			
Conv+ReLU	Conv+ReLU	Conv+ReLU	Conv+ReLU	Conv+ReLU	Conv+ReLU	ConvGRU	Conv +Softmax
3x3, 1, 64	3x3, 2, 64	3x3, 3, 64	3x3, 4, 64	3x3, 3, 64	3x3, 2, 64	3x3, 1, 64	3x3, 1, $\|A\|$

Value network		
Conv+ReLU	Conv+ReLU	Conv
3x3, 3, 64	3x3, 2, 64	3x3, 1, 1

Figure 5.1: Network architecture of the fully convolutional A3C. The numbers in the table denote the filter size, dilation factor, and output channels, respectively.

computed as follows:

$$R_i^{(t)} = r_i^{(t)} + \gamma V(s_i^{(t+1)}), \tag{5.8}$$

$$d\theta_v = \nabla_{\theta_v} \frac{1}{N} \sum_{i=1}^{N} \left(R_i^{(t)} - V(s_i^{(t)})\right)^2, \tag{5.9}$$

$$A(a_i^{(t)}, s_i^{(t)}) = R_i^{(t)} - V(s_i^{(t)}), \tag{5.10}$$

$$d\theta_p = -\nabla_{\theta_p} \frac{1}{N} \sum_{i=1}^{N} \log \pi(a_i^{(t)}|s_i^{(t)}) A(a_i^{(t)}, s_i^{(t)}). \tag{5.11}$$

As shown in Eqs. (5.9) and (5.11), the gradient for each network parameter is the average of the gradients at all pixels.

However, one of the recent trends in CNNs is to enlarge the receptive field to boost the network performance [176, 4]. Our network architecture, which was inspired by [4] in Fig. 5.1, has a large receptive field. In this case, the policy and value networks observe not only the i-th pixel $s_i^{(t)}$ but also the neighbor pixels to output the policy π and value V at the i-th pixel. In other words, the action $a_i^{(t)}$ affects not only the $s_i^{(t+1)}$ but also the policies and values in $\mathcal{N}(i)$ at the next time step, where $\mathcal{N}(i)$ is the local window centered at the i-th pixel. Therefore, to consider it, we replace R_i in Eq. (5.8) as follows:

$$R_i^{(t)} = r_i^{(t)} + \gamma \sum_{j \in \mathcal{N}(i)} w_{i-j} V(s_j^{(t+1)}), \tag{5.12}$$

where w_{i-j} is the weight that means how much we consider the values V of the neighbor pixels at the next time step $(t+1)$. $\boldsymbol{w}$ can be regarded as a convolution filter weight and can be learned simultaneously with the network parameters θ_p and θ_v. It is noteworthy that the second term in Eq. (5.12) is a 2D convolution because each pixel i has a 2D coordinate (i_x, i_y).

Using the matrix form, we can define the $\boldsymbol{R}^{(t)}$ in the n-step case.

$$\begin{aligned}\boldsymbol{R}^{(t)} = \boldsymbol{r}^{(t)} + \gamma \boldsymbol{w} * \boldsymbol{r}^{(t+1)} + \gamma^2 \boldsymbol{w}^2 * \boldsymbol{r}^{(t+2)} + \cdots \\ + \gamma^{n-1} \boldsymbol{w}^{n-1} * \boldsymbol{r}^{(t+n-1)} + \gamma^n \boldsymbol{w}^n * V(\boldsymbol{s}^{(t+n)}),\end{aligned} \tag{5.13}$$

where $*$ is the convolution operator, and $\boldsymbol{w}^n * \boldsymbol{r}$ denotes the n-times convolution on $\boldsymbol{r}$ with the filter $\boldsymbol{w}$. Similar to θ_p and θ_v in Eqs. (5.9) and (5.11), the gradient for $\boldsymbol{w}$ is computed as follows:

$$d\boldsymbol{w} = -\nabla_{\boldsymbol{w}} \frac{1}{N} \sum_{i=1}^{N} \log \pi(a_i^{(t)} | s_i^{(t)})(R_i^{(t)} - V(s_i^{(t)})) + \nabla_{\boldsymbol{w}} \frac{1}{N} \sum_{i=1}^{N} (R_i^{(t)} - V(s_i^{(t)}))^2. \tag{5.14}$$

Similar to typical policy gradient algorithms, the first term in Eq. (5.14) encourages a higher expected total reward. The second term operates as a regularizer such that R_i is not deviated from the prediction $V(s_i^{(t)})$ by the convolution.

We summarize the training algorithm of the fully convolutional A3C with the proposed reward map convolution in Algorithm 2. The differences from the original A3C are highlighted in red.

5.6 Applications and Results

We implemented the proposed method on Python with Chainer [177] and ChainerRL [2] libraries, and applied it to three different applications.

[2]https://github.com/chainer/chainerrl

Algorithm 2: Training pseudo-code of fully convolutional A3C with the proposed reward map convolution

```
// Assume global shared parameter vectors $\theta_p$, $\theta_v$, and $\boldsymbol{w}$ and global counter $T = 0$.
// Assume thread-specific parameter vectors $\theta'_p$, $\theta'_v$, and $\boldsymbol{w}'$.
Initialize thread step counter $t \leftarrow 1$.
repeat
    Reset gradients: $d\theta_p \leftarrow 0$, $d\theta_v \leftarrow 0$, and $d\boldsymbol{w} \leftarrow 0$.
    Synchronize thread-specific parameters $\theta'_p = \theta_p$, $\theta'_v = \theta_v$, and $\boldsymbol{w}' = \boldsymbol{w}$
    $t_{start} = t$
    Obtain state $s_i^{(t)}$ for $\forall i$
    repeat
        Perform $a_i^{(t)}$ according to policy $\pi(a_i^{(t)}|s_i^{(t)})$ for $\forall i$
        Receive reward $r_i^{(t)}$ and new state $s_i^{(t+1)}$ for $\forall i$
        $t \leftarrow t+1$
        $T \leftarrow T+1$
    until terminal $s_i^{(t)}$ or $t - t_{start} == t_{max}$
    for $\forall i$ $R_i = \begin{cases} 0 & \text{for terminal } s_i^{(t)} \\ V(s_i^{(t)}) & \text{for non-terminal } s_i^{(t)} \end{cases}$
    for $k \in \{t-1, \cdots, t_{start}\}$ do
        $R_i \leftarrow \gamma R_i$
        Convolve $\boldsymbol{R}$ with $\boldsymbol{w}$: $R_i \leftarrow \sum_{j \in \mathcal{N}(i)} w_{i-j} R_j$ for $\forall i$
        $R_i \leftarrow r_i^{(k)} + R_i$
        Accumulate gradients w.r.t. $\theta'_p$:
        $d\theta_p \leftarrow d\theta_p - \nabla_{\theta'_p} \frac{1}{N} \sum_{i=1}^{N} \log \pi(a_i^{(k)}|s_i^{(k)})(R_i - V(s_i^{(k)}))$
        Accumulate gradients w.r.t. $\theta'_v$: $d\theta_v \leftarrow d\theta_v + \nabla_{\theta'_v} \frac{1}{N} \sum_{i=1}^{N} (R_i - V(s_i^{(k)}))^2$
        Accumulate gradients w.r.t. $\boldsymbol{w}'$:
        $d\boldsymbol{w} \leftarrow d\boldsymbol{w} - \nabla_{\boldsymbol{w}} \frac{1}{N} \sum_{i=1}^{N} \log \pi(a_i^{(k)}|s_i^{(k)})(R_i - V(s_i^{(k)})) + \nabla_{\boldsymbol{w}} \frac{1}{N} \sum_{i=1}^{N} (R_i - V(s_i^{(k)}))^2$
    end for
    Update $\theta_p$, $\theta_v$, and $\boldsymbol{w}$ using $d\theta_p$, $d\theta_v$, and $d\boldsymbol{w}$, respectively.
until $T > T_{max}$
```

5.6.1 Image Denoising

Method

The input image $\boldsymbol{I}(= \boldsymbol{s}^{(0)})$ is a noisy gray scale image, and the agents iteratively remove the noises by executing actions. It is noteworthy that the proposed method can also be applied to color images by independently manipulating on the three channels. Table 5.1 shows the list of actions that the agents can execute, which were

Table 5.1: Actions for image denoising and restoration.

	action	filter size	parameter
1	box filter	5x5	-
2	bilateral filter	5x5	$\sigma_c = 1.0, \sigma_S = 5.0$
3	bilateral filter	5x5	$\sigma_c = 0.1, \sigma_S = 5.0$
4	median filter	5x5	-
5	Gaussian filter	5x5	$\sigma = 1.5$
6	Gaussian filter	5x5	$\sigma = 0.5$
7	pixel value += 1	-	-
8	pixel value -= 1	-	-
9	do nothing	-	-

empirically decided. We defined the reward $r_i^{(t)}$ as follows:

$$r_i^{(t)} = (I_i^{target} - s_i^{(t)})^2 - (I_i^{target} - s_i^{(t+1)})^2, \tag{5.15}$$

where I_i^{target} is the i-th pixel value of the original clean image. Intuitively, Eq. (5.15) means how much the squared error on the i-th pixel was decreased by the action $a_i^{(t)}$. As shown in [178], maximizing the total reward in Eq. (5.15) is equivalent to minimizing the squared error between the final state $\boldsymbol{s}^{(t_{max})}$ and the original clean image $\boldsymbol{I}^{target}$.

Implementation Details

We used BSD68 dataset [167], which has 428 train images and 68 test images. Similar to [4], we added 4,774 images from Waterloo exploration database [179] to the training set. We set the minibatch size to 64, and the training images were augmented with 70×70 random cropping, left-right flipping, and random rotation. To train the fully convolutional A3C, we used ADAM optimizer [180] and the poly learning, where the learning rate started from 1.0×10^{-3} and multiplied by $(1 - \frac{episode}{max\_episode})^{0.9})$ at each episode. We set the *max_episode* to 30,000 and the length of each episode t_{max} to 5. Therefore, the maximum global counter T_{max} in Algorithm 2 was $30,000 \times 5 = 150,000$. To reduce the training time, we initialize the weights of the fully convolutional A3C with the publicly available weights of [4], except for the convGRU and the last layers. We adopted the stepwise training: the fully convolutional A3C was trained first, subsequently it was trained again with the reward map convolution. We set the filter size of $\boldsymbol{w}$ to 33×33, which is equal to the receptive field size of the networks

Table 5.2: PSNR [dB] on BSD68 test set with Gaussian noise.

Method			std. σ 15	25	50
BM3D [163]			31.07	28.57	25.62
WNNM [181]			31.37	28.83	25.87
TNRD [182]			31.42	28.92	25.97
MLP [183]			-	28.96	26.03
CNN [4]			31.63	29.15	26.19
CNN [4] +aug.			**31.66**	**29.18**	**26.20**
Proposed					
+convGRU	+RMC	+aug.			
			31.17	28.75	25.78
✓			31.26	28.83	25.87
✓	✓		31.40	28.85	25.88
✓	✓	✓	31.49	28.94	25.95

in Fig. 5.1. The number of asynchronous threads was one (*i.e.*, equivalent to A2C: advantage actor-critic). $\boldsymbol{w}$ was initialized as the identity mapping (*i.e.*, only the center of $\boldsymbol{w}$ was one, and zero otherwise). It required approximately 15.5 hours for the 30,000 episode training, and 0.44 sec on average for a test image whose size is 481×321 on a single Tesla V100 GPU.

Results

Table 5.2 shows the comparison of Gaussian denoising with other methods. RMC is the abbreviation for reward map convolution. Aug. means the data augmentation for test images, where a single test image was augmented to eight images by a left-right flip and 90°, 180°, and 270° rotations, similar to [184]. We observed that CNN [4] is the best. However, the proposed method achieved the comparable results with other state-of-the-art methods. Adding the convGRU to the policy network improved the PSNR by approximately +0.1dB. The RMC significantly improved the PSNR when $\sigma = 15$, but improved little when $\sigma = 25$ and 50. That is because the agents can obtain much reward by removing the noises at their own pixels rather than considering the neighbor pixels when the noises are strong. The augmentation for test images further boosted the performance. We report the CNN [4] with the same augmentation for a fair comparison.

Table 5.3 shows the comparison of Poisson denoising. Similar to [185], we simulated the Poisson noise with different peak intensities. The lower the peak intensity,

Table 5.3: PSNR [dB] on BSD68 test set with Poisson noise.

Method			Peak Intensity 120	30	10
CNN [4]			31.62	28.20	25.93
CNN [4] +aug.			**31.66**	**28.26**	**25.96**
Proposed +convGRU	+RMC	+aug.			
			31.17	27.84	25.55
✓			31.28	27.94	25.64
✓	✓		31.37	27.95	25.70
✓	✓	✓	31.47	28.03	25.77

Table 5.4: PSNR [dB] on BSD68 test set with Salt&Pepper noise.

Method			Noise density 0.1	0.5	0.9
CNN [4]			40.16	29.19	23.58
CNN [4] +aug.			**40.40**	29.40	23.76
Proposed +convGRU	+RMC	+aug.			
			36.51	27.91	22.73
✓			37.86	29.26	23.54
✓	✓		38.46	29.78	23.78
✓	✓	✓	38.82	**29.92**	**23.81**

the higher is the noise generated. An almost similar tendency to Gaussian denoising was observed. The proposed method achieved a slightly lower performance, compared with CNN [4].

Fig. 5.2 shows the number of actions executed by the proposed method at each time step for Gaussian denoising ($\sigma = 50$) on the BSD68 test set. We observed that the agents successfully obtained a strategy in which they first removed the noises using strong filters (box filter, bilateral filter $\sigma_c = 1.0$, and Gaussian filter $\sigma = 1.5$); subsequently they adjusted the pixel values by the other actions (pixel values +=1 and -=1).

Table 5.4 shows the comparison of salt and pepper denoising. We observed that the RMC significantly improved the performance. In addition, the proposed method outperformed the CNN [4] when the noise density is 0.5 and 0.9. Unlike Gaussian and Poisson noises, it is difficult to regress the noise with CNN when the noise

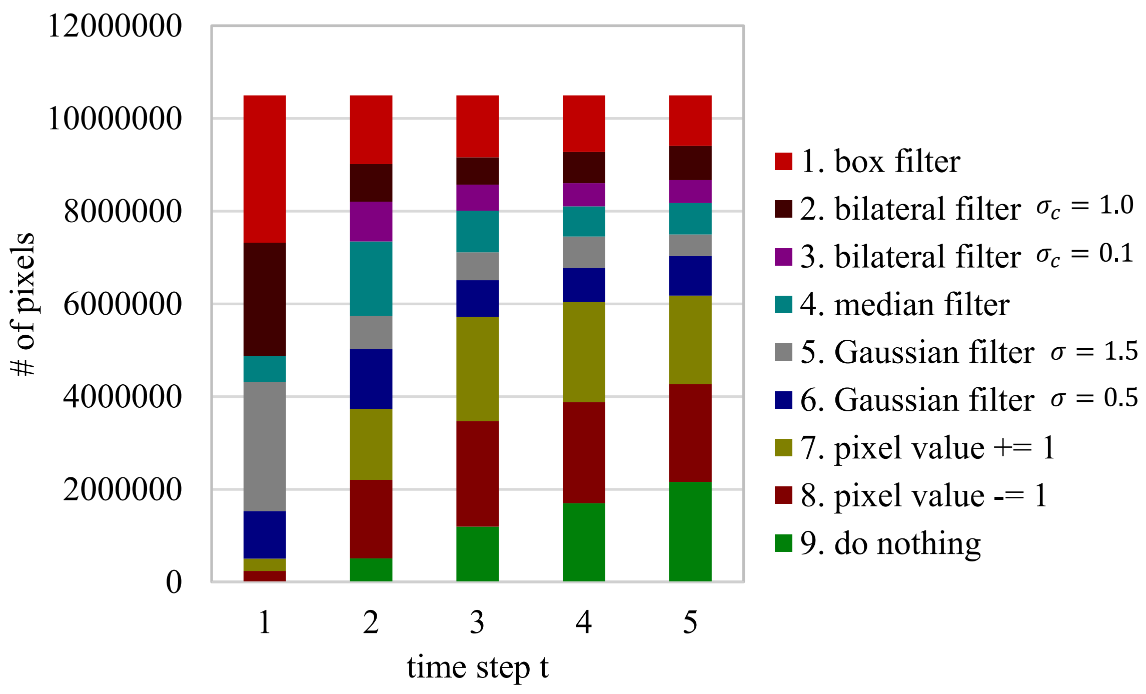


Figure 5.2: Number of actions executed at each time step for Gaussian denoising ($\sigma = 50$) on the BSD68 test set.

density is high because the information of the original pixel value is lost (*i.e.*, the pixel value was changed to 0 or 255 by the noise). In contrast, the proposed method can predict the true pixel values from the neighbor pixels with the iterative filtering actions.

We visualize the denoising process of the proposed method, and the action map at each time step in Fig. 5.3. We observed that the noises are iteratively removed by the chosen actions.

Fig. 5.4 shows the qualitative comparison with CNN [4]. The proposed method achieved both quantitative and visually better results for salt and pepper denoising.

Ablation Study with Different Action Sets

We conducted the ablation studies with different set of actions on Gaussian denoising. Table 5.5 shows the results. When the actions were only basic filters ([1] box filters, [4] median filter, [5] Gaussian filter with $\sigma = 1.5$, and [9] do nothing in Table 5.1), its PSNRs were 29.82, 27.60, and 25.20 for noise std. 15, 25, and 50, respectively. When we added [2] bilateral filter with $\sigma_c = 1.0$, the PSNRs increased to 29.93, 27.81, and 25.30. In addition, when we added the two actions ([7] pixel value += 1 and [8] pixel

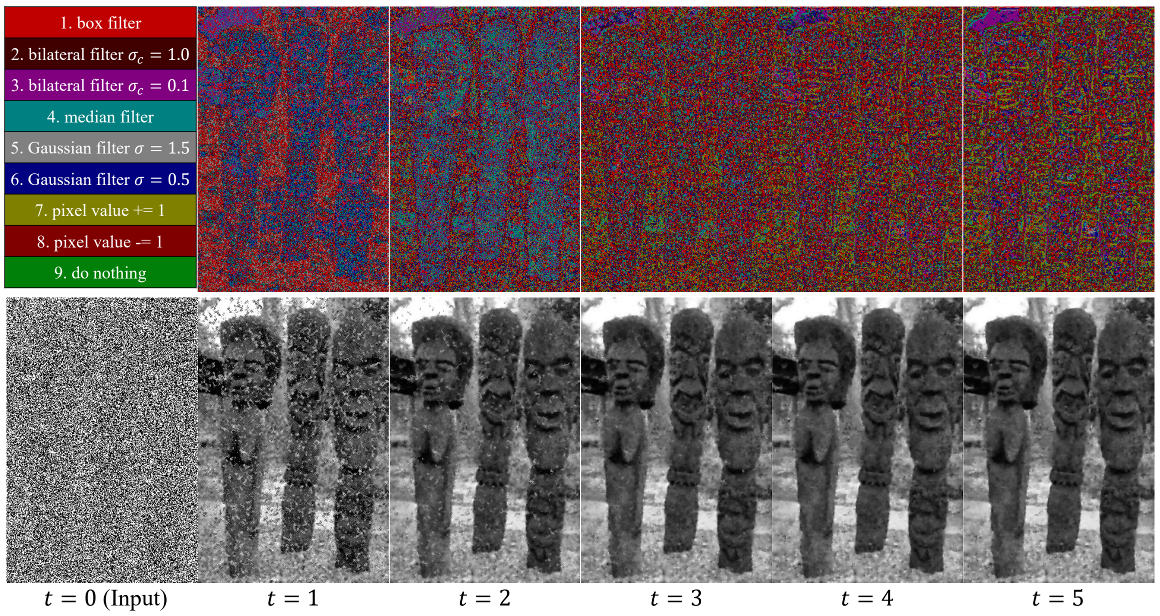

1. box filter
2. bilateral filter $\sigma_c = 1.0$
3. bilateral filter $\sigma_c = 0.1$
4. median filter
5. Gaussian filter $\sigma = 1.5$
6. Gaussian filter $\sigma = 0.5$
7. pixel value += 1
8. pixel value -= 1
9. do nothing
$t = 0$ (Input)
$t = 1$
$t = 2$
$t = 3$
$t = 4$
$t = 5$

Figure 5.3: Denoising process of the proposed method and the action map at each time step for salt and pepper denoising (density=0.9).

Table 5.5: PSNR [dB] on Gaussian denoising with different action sets.

Actions	std. σ 15	25	50
only basic filters	29.82	27.60	25.20
+ bilateral filter $\sigma_c = 1.0$	29.93	27.81	25.30
+ pixel value $\pm = 1$	30.72	28.25	25.59
+ different filter parameters.	31.26	28.83	25.87

value -= 1), the PSNRs further increased to 30.72, 28.25, and 25.59. Finally, when we added the two filters with different parameters ([3] bilateral filter with $\sigma_c = 0.1$ and [6] Gaussian filter with $\sigma = 0.5$), the PSNRs were 31.26, 28.83, and 25.87. Therefore, all the actions are important for the high performance although it may be further increased if we can find more proper action set. Although we tried adding some advanced filters for image denoising such as guided filter [186] and non-local means filter [187], the performance was not improved any more.

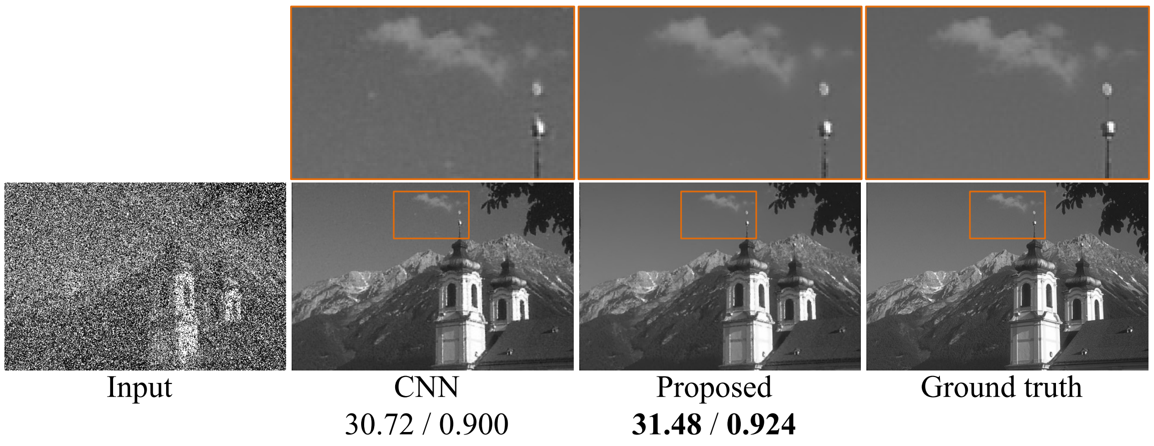


Figure 5.4: Qualitative comparison of the proposed method and CNN [4] for salt and pepper noise (density=0.5). PSNR / SSIM are reported.

5.6.2 Image Restoration

Method

We applied the proposed method to "blind" image restoration, where no mask of blank regions is provided. The proposed method iteratively inpaints the blank regions by executing actions. We used the same actions and reward function as those of image denoising, which are shown in Table 5.1 and Eq. (5.15), respectively.

For training, we used 428 training images from the BSD68 train set, 4,774 images from the Waterloo exploration database, and 20,122 images from the ILSVRC2015 val set [188] (a total of 25,295 images). We also used 11,343 documents from the Newsgroups 20 train set [189]. During the training, we created each training image by randomly choosing an image from the 25,295 images and a document from 11,343 documents, and overlaid it on the image. The font size was randomly decided from the range [10,30]. The font type was randomly chosen between *Arial* and *Times New Roman*, where the bold and Italic options were randomly added. The intensity of the text region was randomly chosen from 0 or 255. We created the test set that has 68 images by overlaying the randomly chosen 68 documents from the Newsgroup 20 test set on the BSD68 test images. The settings of the font size and type were the same as those of the training. The random seed for the test set was fixed between the different methods. All the hyperparameters were same as those in image denoising, except for the length of the episodes, *i.e.*, $t_{max} = 15$.

Table 5.6: Comparison on image restoration.

Method		PSNR [dB]	SSIM
Net-D and Net-E [5]		29.53	0.846
CNN [4]		29.75	0.858
Proposed			
+convGRU	+RMC		
✓		29.50	0.858
✓	✓	**29.97**	**0.868**

Results

Table 5.6 shows the comparison of the averaged PSNR between the output and ground-truth images. We saved the models of the compared methods at every one epoch, and reported the best results. For the proposed method, we saved the model at every 300 episodes ($\simeq$ 0.76 epoch) and reported the best results. Here, we compared the proposed method with the two methods (Net-E and Net-D [5] and CNN [4]) because the Net-E and Net-D achieved much better results than the other restoration methods in the original paper. We found that the RMC significantly improved the performance, and the proposed method obtained the best result. This is the similar reason to the case of the salt and pepper noise. Because the information of the original pixel value is lost by the overlaid texts, its regression is difficult. In contrast, the proposed method predicts the true pixel value by iteratively propagating the neighbor pixel values with the filtering actions.

Fig. 5.5 is the visualization of restoration process of the proposed method, and the action map at each time step. Fig. 5.6 shows the qualitative comparison with Net-E and Net-D [5] and CNN [4]. We observed that there are visually large differences between the results from the proposed method and those from the compared methods.

5.6.3 Local Color Enhancement

Method

We also applied the proposed method to the local color enhancement. We used the dataset created by [6], which has 70 train images and 45 test images downloaded from Flicker. Using Photoshop, all the images were enhanced by a professional photographer for three different stylistic local effects: Foreground Pop-Out, Local

Table 5.7: Thirteen actions for local color enhancement.

Action		
1	contrast	×0.95
2	contrast	×1.05
3	color saturation	×0.95
4	color saturation	×1.05
5	brightness	×0.95
6	brightness	×1.05
7	red and green	×0.95
8	red and green	×1.05
9	green and blue	×0.95
10	green and blue	×1.05
11	red and blue	×0.95
12	red and blue	×1.05
13	do nothing	

Xpro, and Watercolor. Inspired by [157], we decided the action set as shown in Table 5.7. Given an input image $\boldsymbol{I}$, the proposed method changes the three channel pixel value at each pixel by executing an action. When inputting $\boldsymbol{I}$ to the network, the RGB color values were converted to CIELab color values. We defined the reward function as the decrease of L2 distance in the CIELab color space as follows:

$$r_i^{(t)} = |I_i^{target} - s_i^{(t)}|_2 - |I_i^{target} - s_i^{(t+1)}|_2. \tag{5.16}$$

All the hyperparameters and settings were same as those in image restoration, except for the length of episodes, *i.e.*, $t_{max} = 10$.

Results

Table 5.8 shows the comparison of mean L2 errors on 45 test images. The proposed method achieved better results than DNN [6] on all three enhancement styles, and comparable or slightly better results than pix2pix. We observed that the RMC improved the performance although their degrees of improvement depended on the styles. It is noteworthy that the existing color enhancement method using deep RL [157, 158] cannot be applied to this local enhancement application because they can execute only global actions.

Fig. 5.7 is the visualization of the color enhancement process of the proposed method, and the action map at each time step. Similar to [157], the proposed method

Table 5.8: Comparison of mean L2 testing errors on local color enhancement. The errors except for the proposed method and pix2pix are from [6].

Method		Foreground Pop-Out	Local Xpro	Watercolor
Original		13.86	19.71	15.30
Lasso		11.44	12.01	9.34
Random Forest		9.05	7.51	11.41
DNN [6]		7.08	7.43	7.20
Pix2pix [190]		**5.85**	6.56	8.84
Proposed				
+convGRU	+RMC			
✓		6.75	6.17	6.44
✓	✓	6.69	**5.67**	**6.41**

is interpretable while the DNN-based color mapping method [6] is not. We can see that the brightness and saturation were mainly increased to convert the input image to watercolor style.

Fig. 5.8 shows the qualitative comparison between the proposed method and DNN [6]. The proposed method achieved both quantitatively and qualitatively better results.

5.7 Conclusions

We proposed a novel pixelRL problem setting and applied it to three different applications: image denoising, image restoration, and local color enhancement. We also proposed an effective learning method for the pixelRL problem, which boosts the performance of the pixelRL agents. Our experimental results demonstrated that the proposed method achieved comparable or better results than state-of-the-art methods on each application. Different from the existing deep learning-based methods for such applications, the proposed method is interpretable. The interpretability of deep learning has been attracting much attention [191], and it is especially important for some applications such as medical image processing [159].

The proposed method can maximize the pixel-wise reward; in other words, it can minimize the pixel-wise non-differentiable objective function. Therefore, we believe that the proposed method can be potentially used for more image processing applications where supervised learning cannot be applied.

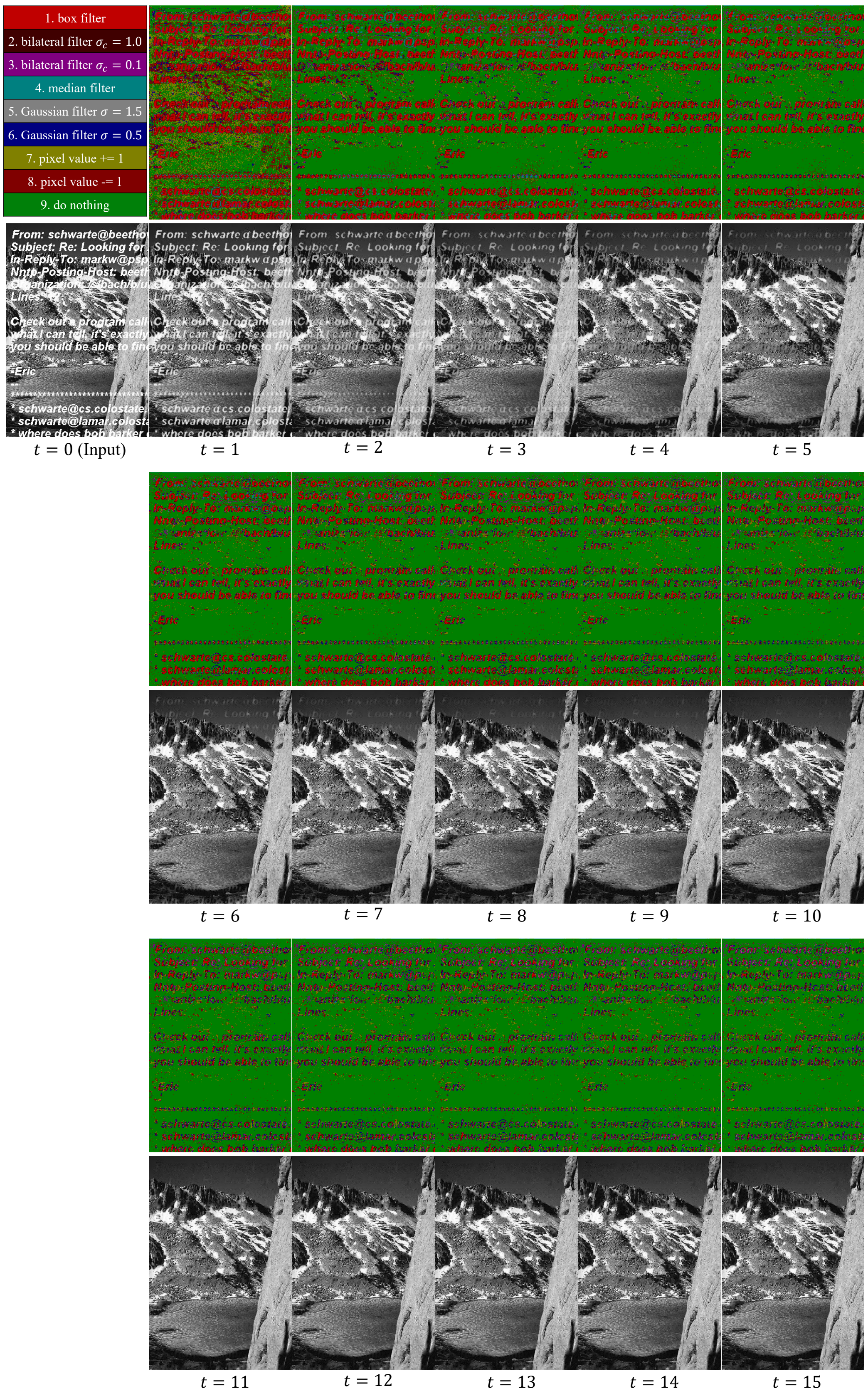


Figure 5.5: Restoration process of the proposed method and the action map at each time step.

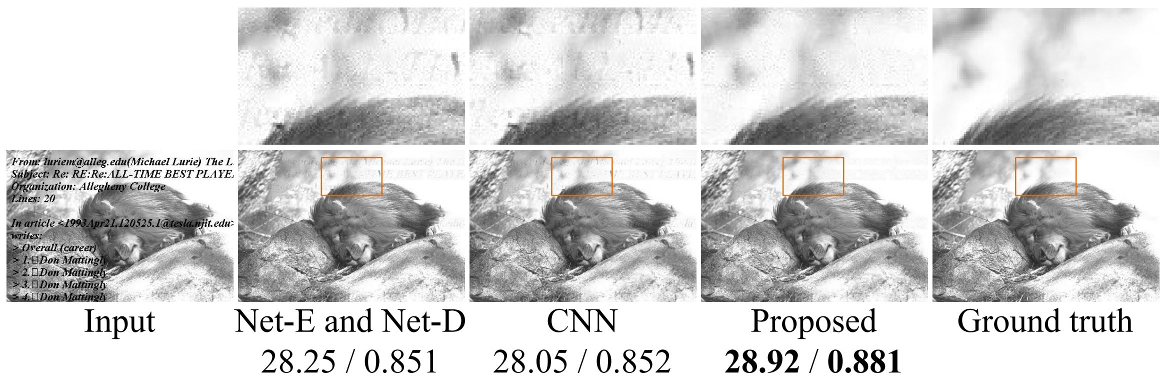


Figure 5.6: Qualitative comparison of the proposed method with Net-E and Net-D [5] and CNN [4] on image restoration. PSNR / SSIM are reported.

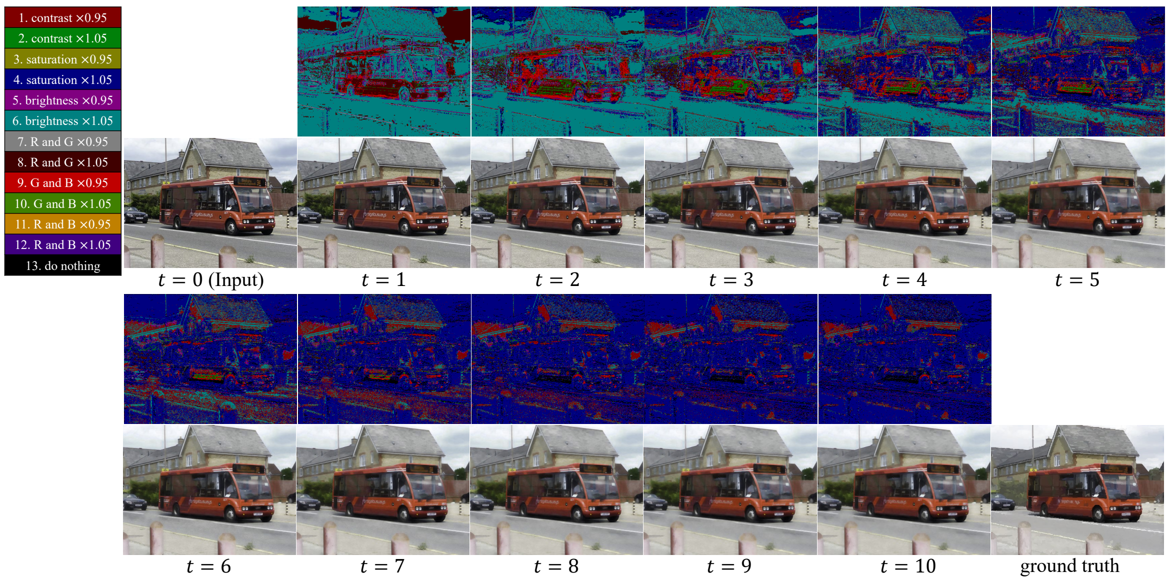


Figure 5.7: Color enhancement process of the proposed method for watercolor, and the action map at each time step.

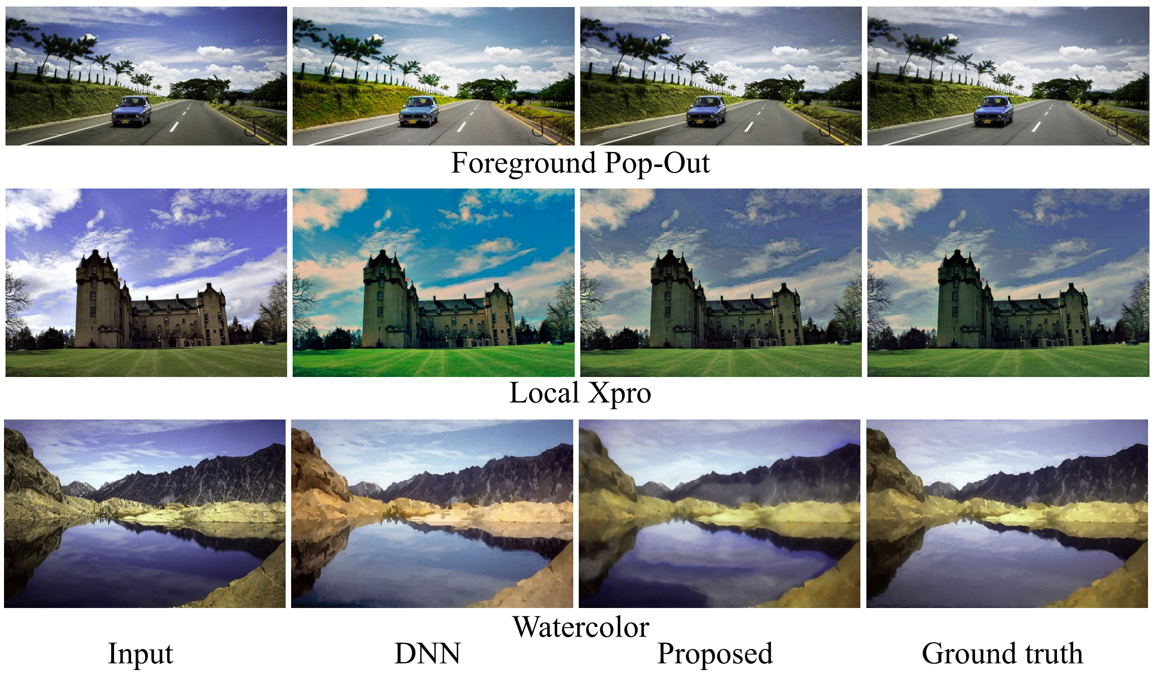


Figure 5.8: Qualitative comparison of the proposed method and DNN [6]. The saturation of the images from DNN appear higher owning to the color correction for the sRGB space (for details, see https://github.com/stephenyan1231/dl-image-enhance).

Chapter 6

Conclusions

In this thesis, we tackled two challenges of pixel labeling: (i) **how to deal with the large solution space,** and (ii) **how to learn the relationships between neighbor labels effectively.** For the first challenge, we proposed two neighbor-aware fast optimization methods in chapter 2 and 3. For the second challenge, we proposed two effective learning methods that boost the performance of pixel labeling by considering neighbor labels in chapter 4 and 5.

In chapter 2, we proposed a fast optimization method based on CVF for general pixel-labeling problems, where smoothness of labels between neighbor pixels are forced. We applied the proposed method to stereo matching and optical flow estimation. It is important to note that the applications of the proposed method are not limited to these two. Experimental results showed that the proposed method was much more efficient when the label space is large while keeping the accuracy, compared with the original CVF. In addition, our method was more accurate than CVF in some cases by successfully truncating noisy labels with the proposed coarse-to-fine strategy.

In chapter 3, we proposed a fast optimization method named "multi-pass dynamic programming" for the MRF optimization with the constraint that the neighbor labels must be connected. Volume seam carving (seam carving for 3D cost volume) is its main application, which is applied to various image processing tasks. The proposed method was applied to three different image processing tasks: video retargeting, tone mapping, and contrast enhancement. Experimental results showed that the proposed method is approximately 90 times faster and consumes 8 times less memory than graph cuts, which has been the only choice for the volume seam carving so far. Our large scale subjective analysis by more than 198 crowd workers confirmed that a suboptimal solution obtained by the proposed method creates almost equivalent image quality to that of graph cuts. The proposed method makes various of image processing techniques based on volume seam carving more practical.

In chapter 4, we revealed that the fixed point iteration of a dense CRF is mathematically equivalent to recurrent convolution. Based on this observation, we proposed a new dense CRF model that can automatically learn the relationships between neighbor labels from training data in contrast to conventional hand-crafted CRF. In addition, the proposed model can be trained end-to-end with the unary part (FCN) by backpropagation. We applied the proposed model to semantic segmentation and confirmed that the proposed model achieved superior performance on PASCAL VOC benchmark.

In chapter 5, we proposed a novel problem setting *pixelRL* by extending deep reinforcement learning to pixel labeling. We also proposed reward map convolution, which is a neighbor-aware learning method in the pixelRL framework. We applied the proposed method to three image processing applications: image denoising, image restoration, and local color enhancement. Experimental results demonstrated that the proposed method achieved comparable or better results than state-of-the-art methods. It was also confirmed that the proposed reward map convolution boosted the performance by considering the future states of neighbor pixels.

Future Directions

We presented two neighbor-aware fast optimization methods and two neighbor-aware learning methods for pixel labeling. We believe that combining the proposed methods is a promising future direction and can possibly solve more difficult pixel-labeling problems that have never been tackled. For example, by combining the proposed coarse-to-fine strategy in chapter 2 and multi-pass DP in chapter 3, we can tackle the pixel-labeling problems where the label space is extremely large and neighbor labels must be connected. Another example is that the proposed CRF model in chapter 4 can be incorporated into pixelRL in chapter 5 to consider neighbor actions at current time step. These combinations can potentially broaden the range of applications of pixel labeling.

References

[1] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," in *Proc. CVPR*, 2011.

[2] S. Paris, S. Hasinoff, and J. Kautz, "Local Laplacian filters: edge-aware image processing with a Laplacian pyramid," *ACM TOG*, vol. 30, no. 4, 2011.

[3] B. Gu, W. Li, M. Zhu, and M. Wang, "Local edge-preserving multiscale decomposition for high dynamic range image tone mapping," *IEEE TIP*, vol. 22, no. 1, pp. 70–79, 2013.

[4] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep cnn denoiser prior for image restoration," in *Proc. CVPR*, 2017.

[5] Y. Liu, J. Pan, and Z. Su, "Deep blind image inpainting," *arXiv preprint arXiv:1712.09078*, 2017.

[6] Z. Yan, H. Zhang, B. Wang, S. Paris, and Y. Yu, "Automatic photo adjustment using deep neural networks," *ACM TOG*, vol. 35, no. 2, p. 11, 2016.

[7] J. Sun, N.-N. Zheng, and H.-Y. Shum, "Stereo matching using belief propagation," *IEEE TPAMI*, vol. 25, no. 7, pp. 787–800, 2003.

[8] B. Glocker, N. Paragios, N. Komodakis, G. Tziritas, and N. Navab, "Optical flow estimation with uncertainties through dynamic mrfs," in *Proc. CVPR*, 2008.

[9] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in nd images," in *Proc. ICCV*, 2001.

[10] M. Rubinstein, A. Shamir, and S. Avidan, "Improved seam carving for video retargeting," *ACM TOG*, vol. 27, no. 3, 2008.

[11] B. Chen and P. Sen, "Video carving," in *Proc. Eurographics, Short Papers*, 2008.

[12] Z. Li, P. Ishwar, and J. Konrad, "Video condensation by ribbon carving," *IEEE TIP*, vol. 18, no. 11, pp. 2572–2583, 2009.

[13] I. Tsubaki and K. Iwauchi, "Tone mapping based on luminance compression using seam surfaces," *IEICE Trans. Inf. Syst. (Japanese Edition)*, vol. 98-D, no. 4, pp. 606–615, 2015.

[14] I. Tsubaki and K. Iwauchi, "Depth remapping using seam carving for depth image based rendering," in *Proc. SPIE 9399, Image Processing: Algorithms and Systems XIII*, 2015.

[15] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE TPAMI*, vol. 23, no. 11, pp. 1222–1239, 2001.

[16] J. Sun, N. N. Zheng, and H. Y. Shum, "Stereo matching using belief propagation," *IEEE TPAMI*, vol. 25, no. 7, pp. 787–800, 2003.

[17] M. F. Tappen and W. T. Freeman, "Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters," in *Proc. ICCV*, 2003.

[18] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithm for energy minimization in vision," *IEEE TPAMI*, vol. 26, no. 9, pp. 1124–1137, 2004.

[19] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "Map estimation via agreement on trees: Message-passing and linear-programming approaches," *IEEE Trans. Information Theory*, vol. 51, no. 11, pp. 3697–3717, 2005.

[20] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *IEEE TPAMI*, vol. 28, no. 10, pp. 1568–1583, 2006.

[21] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *IJCV*, vol. 70, no. 1, pp. 41–54, 2006.

[22] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for markov random fields with smoothnes-based priors," *IEEE TPAMI*, vol. 30, no. 6, pp. 1068–1080, 2008.

[23] Q. Yang, "A non-local cost aggregation method for stereo matching," in *Proc. CVPR*, 2012.

[24] A. Hosni, C. Rhemann, M. Bleyer, and M. Gelautz, "Temporally consistent disparity and optical flow via efficient spatio-temporal filtering," in *Proc. PSIVT*, 2012.

[25] A. Brunton, J. Lang, and E. Dubois, "Efficient multi-scale stereo of high-resolution planar and spherical images," in *Proc. 3DIMPVT*, 2012.

[26] V. Kramarev, O. Demetz, C. Schroers, and J. Weickert, "Cross anisotropic cost volume filtering for segmentation," in *Proc. ACCV*, 2013.

[27] J. Cho, S. Ikehata, H. Yoo, M. Gelautz, and K. Aizawa, "Depth map up-sampling using cost-volume filtering," in *Proc. IVMSP Workshop*, 2013.

[28] J. Lu, H. Yang, and N. D. Min, D. Minh, "PatchMatch filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation," in *Proc. CVPR*, 2013.

[29] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, “PatchMatch: A randomized correpondence algorithm for structual image editing,” in *Proc. SIGGRAGH*, 2009.

[30] C. Bames, E. Shechtman, D. B. Goldman, and A. Finkelstein, “The generalized patchmatch correspondence algorithm,” in *Proc. ECCV*, 2010.

[31] “Middlebury stereo database,” http://vision.middlebury.edu/stereo/.

[32] “Middlebury optical flow database,” http://vision.middlebury.edu/flow/.

[33] K. J. Yoon and I. S. Kweon, “Locally adaptive support-weight approach for visual correspondence search,” in *Proc. CVPR*, 2005.

[34] K. J. Yoon and I. S. Kweon, “Adaptive support-weight approach for correspondence search,” *IEEE TPAMI*, vol. 28, no. 4, pp. 650–656, 2006.

[35] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in *Proc. ICCV*, 1998.

[36] C. Richardt, D. Orr, I. Davies, A. Criminisi, and N. A. Dodgson, “Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid,” in *Proc. ECCV*, 2010.

[37] Q. Yang, “Stereo matching using tree filtering,” *IEEE TPAMI*, vol. 37, no. 4, pp. 834–846, 2015.

[38] X. Bai, X. Luo, S. Li, and H. Lu, “Adaptive stereo matching via loop-erased random walk,” in *Proc. ICIP*, 2014.

[39] K. He, J. Sun, and X. Tang, “Guided image filtering,” in *Proc. ECCV*, 2010.

[40] J. Lu, K. Shi, D. Min, L. Lin, and M. N. Do, “Cross-based local multipoint filtering,” in *Proc. CVPR*, 2012.

[41] K. Zhang, Y. Fang, D. Min, L. Sun, S. Yang, S. Yan, and Q. Tian, “Cross-scale cost aggregation for stereo matching,” in *Proc. CVPR*, 2014.

[42] Y. Zhan, Y. Gu, K. Huang, C. Zhang, and K. Hu, “Accurate image-guided stereo matching with efficient matching cost and disparity refinement,” *IEEE TCSVT*, vol. 26, no. 9, pp. 1632–1645, 2016.

[43] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, “Flownet: Learning optical flow with convolutional networks,” in *Proc. ICCV*, 2015.

[44] J. Zbontar and Y. LeCun, “Computing the stereo matching cost with a convolutional neural network,” in *Proc. CVPR*, 2015.

[45] Z. Chen, X. Sun, L. Wang, Y. Yu, and C. Huang, “A deep visual correspondence embedding model for stereo matching costs,” in *Proc. ICCV*, 2015.

[46] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *JMLR*, vol. 17, no. 1-32, p. 2, 2016.

[47] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proc. CVPR*, 2016.

[48] D. Min, J. Lu, and M. N. Do, "A revisit to cost aggregation in stereo matching: How far can we reduce its computational redundancy?" in *Proc. ICCV*, 2011.

[49] D. Min, J. Lu, and M. N. Do, "Joint histogram-based cost aggregation for stereo matching," *IEEE TPAMI*, vol. 35, no. 10, pp. 2539–2545, 2013.

[50] Y. Wang, K. Wang, E. Dunn, and J.-M. Frahm, "Stereo under sequential optimal sampling: A statistical analysis framework for search space reduction," in *Proc. CVPR*, 2014.

[51] M. A. Helala and F. Z. Qureshi, "Accelerating cost volume filtering using salient subvolumes and robust occlusion handling," in *Proc. ACCV*, 2014.

[52] R. Yang and M. Pollefeys, "Multi-resolution real-time stereo on commodity graphics hardware," in *Proc. CVPR*, 2003.

[53] X. Tan, C. Sun, D. Wang, Y. Guo, and T. D. Pham, "Soft cost aggregation with multi-resolution fusion," in *Proc. ECCV*, 2014.

[54] Y. Zhao and G. Taubin, "Real-time stereo on gpgpu using progressive multi-resolution adaptive windows," *Image and Vision Computing*, vol. 29, no. 6, pp. 420–432, 2011.

[55] Q. Yang, L. Wang, and N. Ahuja, "A constant-space belief propagation algorithm for stereo matching," in *Proc. CVPR*, 2010.

[56] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Proc. ECCV*, 2004.

[57] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers, "An improved algorithm for tv-l 1 optical flow," in *Statistical and Geometrical Approaches to Visual Motion Analysis*, 2009, pp. 23–45.

[58] M. Tao, J. Bai, P. Kohli, and S. Paris, "Simpleflow: A non-iterative, sublinear optical flow algorithm," *Computer Graphics Forum*, vol. 31, no. 2pt1, pp. 345–353, 2012.

[59] D. Sun, S. Roth, and M. J. Black, "Secrets of optical flow estimation and their principles," in *Proc. CVPR*, 2010.

[60] L. Bao, Q. Yang, and H. Jin, "Fast edge-preserving PatchMatch for large displacement optical flow," in *Proc. CVPR*, 2014.

[61] L. Bao, Q. Yang, and H. Jin, "Fast edge-preserving patchmatch for large displacement optical flow," *IEEE TIPs*, vol. 23, no. 12, pp. 4996–5006, 2014.

[62] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE TPAMI*, vol. 34, no. 11, pp. 2274–2281, 2012.

[63] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," *ACM TOG*, vol. 26, no. 3, 2007.

[64] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Discontinuous seam-carving for video retargeting," in *Proc. CVPR*, 2010, pp. 569–576.

[65] W.-L. Chao, H.-H. Su, S.-Y. Chien, W. Hsu, and J.-J. Ding, "Coarse-to-fine temporal optimization for video retargeting based on seam carving," in *Proc. ICME*, 2011.

[66] B. Yan, K. Sun, and L. Liu, "Matching-area-based seam carving for video retargeting," *IEEE TCSVT*, vol. 23, no. 2, pp. 302–310, 2013.

[67] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for Markov random fields with smoothness-based priors," *IEEE TPAMI*, vol. 30, no. 6, pp. 1068–1080, 2008.

[68] S. Birchfield and C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," *IJCV*, vol. 35, no. 3, pp. 269–293, 1999.

[69] J. Kim, K. Lee, B. Choi, and S. Lee, "A dense stereo matching using two-pass dynamic programming with generalized ground control points," in *Proc. CVPR*, 2005.

[70] N. Fukushima, T. Yendo, T. Fujii, and M. Tanimoto, "Free viewpoint image generation using multi-pass dynamic programming," in *SPIE Stereoscopic Displays and Virtual Reality Systems XIV*, vol. 6490, 2007, pp. 460–470.

[71] H. Hirschmüller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *Proc. CVPR*, 2005.

[72] N. Fukushima, T. Fujii, Y. Ishibashi, T. Yendo, and M. Tanimoto, "Real-time free viewpoint image rendering by using fast multi-pass dynamic programming," in *Proc. 3DTV-Conf.*, 2010.

[73] Q. Chen and V. Koltun, "Fast mrf optimization with application to depth reconstruction," in *Proc. CVPR*, 2014.

[74] F. Liu and M. Gleicher, "Video retargeting: Automating pan and scan," in *Proc. ACMMM*, 2006.

[75] L. Wolf, M. Guttmann, and D. Cohen-Or, "Non-homogeneous content-driven video-retargeting," in *Proc. ICCV*, 2007.

[76] B. Li, L.-Y. Duan, J. Wang, R. Ji, C.-W. Lin, and W. Gao, "Spatiotemporal grid flow for video retargeting," *IEEE TIP*, vol. 23, no. 4, pp. 1615 – 1628, 2014.

[77] Y.-S. Wang, J.-H. Hsiao, O. Sorkine, and T.-Y. Lee, "Scalable and coherent video resizing with per-frame optimization," *ACM TOG*, vol. 30, no. 4, 2011.

[78] Y.-S. Wang, H.-C. Lin, O. Sorkine, and T.-Y. Lee, "Motion-based video retargeting with optimized crop-and-warp," *ACM TOG*, vol. 29, no. 4, 2010.

[79] Y. Niu, F. Liu, X. Li, and M. Gleicher, "Warp propagation for video resizing," in *Proc. CVPR*, 2010.

[80] Y.-S. Wang, H. Fu, O. Sorkine, T.-Y. Lee, and H.-P. Seidel, "Motion-aware temporal coherence for video resizing," *ACM TOG*, vol. 28, no. 5, 2009.

[81] D. Han, X. Wu, and M. Sonka, "Optimal multiple surfaces searching for video/image resizing-a graph-theoretic approach," in *Proc. ICCV*, 2009.

[82] E. Jain, Y. Sheikh, A. Shamir, and J. Hodgins, "Gaze-driven video re-editing," *ACM TOG*, vol. 34, no. 2, 2015.

[83] H. Katti, A. K. Rajagopal, M. Kankanhalli, and R. Kalpathi, "Online estimation of evolving human visual interest," *ACM TOMCCAP*, vol. 11, no. 1, 2014.

[84] G. Larson, H. Rushmeier, and C. Piatko, "A visibility matching tone reproduction operator for high dynamic range scenes," *IEEE TVCG*, vol. 3, no. 4, pp. 291–306, 1997.

[85] R. Fattal, D. Lischinski, and M. Werman, "Gradient domain high dynamic range compression," *ACM TOG*, vol. 21, no. 3, pp. 249–256, 2002.

[86] M. Aubry, S. Paris, S. Hasinoff, J. Kautz, and F. Durand, "Fast local Laplacian filters: Theory and applications," *ACM TOG*, vol. 33, no. 5, 2014.

[87] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE TPAMI*, vol. 35, no. 6, pp. 1397–1409, 2013.

[88] K. Ma, H. Yeganeh, K. Zeng, and Z. Wang, "High dynamic range image compression by optimizing tone mapped image quality index," *IEEE TIP*, vol. 24, no. 10, pp. 3086–3097, 2015.

[89] T. Arici, S. Dikbas, and Y. Altunbasak, "A histogram modification framework and its application for image contrast enhancement," *IEEE TIP*, vol. 18, no. 9, pp. 1921–1935, 2009.

[90] D. Jobson, Z. Rahman, and G. Woodell, "Properties and performance of a center/surround retinex," *IEEE TIP*, vol. 6, no. 3, pp. 451–462, 1997.

[91] M. Bertalmio, V. Caselles, and E. Provenzi, "Issues about Retinex theory and contrast enhancement," *IJCV*, vol. 83, no. 1, pp. 101–119, 2009.

[92] S.-H. Yun, J. H. Kim, and S. Kim, "Image enhancement using a fusion framework of histogram equalization and Laplacian pyramid," *IEEE Trans. Consum. Electron.*, vol. 56, no. 4, pp. 2763–2771, 2010.

[93] R. Fattal, “Dehazing using color-lines,” *ACM TOG*, vol. 34, no. 1, 2014.

[94] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient belief propagation for early vision,” *IJCV*, vol. 70, no. 1, pp. 41–54, 2006.

[95] M. Rubinstein, D. Gutierrez, O. Sorkine, and A. Shamir, “A comparative study of image retargeting,” *ACM TOG*, vol. 29, no. 6, 2010.

[96] I. Setyawan and R. L. Lagendijk, “Human perception of geometric distortions in images,” in *SPIE, Security, Steganography and Watermarking of Multimedia Contents VI*, 2004, pp. 256–267.

[97] E. S. Pearson and H. O. Hartley, “Biometrika tables for statisticians,” *Cambridge University Press*, vol. 1, 1966.

[98] F. Drago, K. Myszkowski, T. Annen, and N. Chiba, “Adaptive logarithmic mapping for displaying high contrast scenes,” *Computer Graphics Forum*, vol. 22, no. 3, pp. 419–426, 2003.

[99] M. D. Fairchild, *The HDR Photographic Survey*. MDF Publications, 2008, http://www.rit-mcsl.org/fairchild/HDRPS/HDRthumbs.html.

[100] H. Yeganeh and Z. Wang, “Objective quality assessment of tone mapped images,” *IEEE TIP*, vol. 22, no. 2, pp. 657–667, 2013.

[101] R. Mantiuk, K. J. Kim, A. G. Rempel, and W. Heidrich, “HDR-VDP-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions,” *ACM TOG*, vol. 30, no. 4, 2011.

[102] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, “Edge-preserving decompositions for multi-scale tone and detail manipulation,” *ACM TOG*, vol. 27, no. 3, 2008.

[103] L.-C. 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,” *IEEE TPAMI*, 2017.

[104] 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 *Proc. ICCV*, 2015.

[105] F. Liu, C. Shen, and G. Lin, “Deep convolutional neural fields for depth estimation from a single image,” in *Proc. CVPR*, 2015.

[106] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. Yuille, “Towards unified depth and semantic prediction from a single image,” in *Proc. CVPR*, 2015.

[107] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich, “Feedforward semantic segmentation with zoom-out features,” in *Proc. CVPR*, 2015.

[108] R. Gadde, V. Jampani, M. Kiefel, D. Kappler, and P. V. Gehler, “Superpixel convolutional networks using bilateral inceptions,” in *Proc. ECCV*, 2016.

[109] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, 2015.

[110] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE TPAMI*, vol. 39, no. 4, pp. 640–651, 2017.

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

[112] L.-C. 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," in *Proc. CVPR*, 2016.

[113] G. Ghiasi and C. C. Fowlkes, "Laplacian pyramid reconstruction and refinement for semantic segmentation," in *Proc. ECCV*, 2016.

[114] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation," in *Proc. CVPR*, 2017.

[115] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. CVPR*, 2017.

[116] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE TPAMI*, 2017.

[117] 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 *Proc. ICLR*, 2015.

[118] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *Proc. NIPS*, 2011.

[119] J. Dai, K. He, and J. Sun, "Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation," in *Proc. ICCV*, 2015.

[120] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. ICCV*, 2015.

[121] D. Lin, J. Dai, J. Jia, K. He, and J. Sun, "Scribblesup: Scribble-supervised convolutional networks for semantic segmentation," in *Proc. CVPR*, 2016.

[122] A. Arnab and P. H. Torr, "Bottom-up instance segmentation using deep higher-order crfs," in *Proc. BMVC*, 2016.

[123] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe, "Multi-scale continuous crfs as sequential deep networks for monocular depth estimation," in *Proc. CVPR*, 2017.

[124] A. Desmaison, R. Bunel, P. Kohli, P. H. Torr, and M. P. Kumar, "Efficient continuous relaxations for dense crf," in *Proc. ECCV*, 2016.

[125] T. Ajanthan, A. Desmaison, R. Bunel, M. Salzmann, P. H. Torr, and M. P. Kumar, "Efficient linear programming for dense crf," in *Proc. CVPR*, 2017.

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

[127] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang, "Semantic image segmentation via deep parsing network," in *Proc. ICCV*, 2015.

[128] G. Lin, C. Shen, I. Reid, and A. van den Hengel, "Deeply learning the messages in message passing inference," in *Proc. NIPS*, 2015.

[129] G. Lin, C. Shen, I. Reid *et al.*, "Efficient piecewise training of deep structured models for semantic segmentation," in *Proc. CVPR*, 2016.

[130] R. Vemulapalli, O. Tuzel, M.-Y. Liu, and R. Chellappa, "Gaussian conditional random field network for semantic segmentation," in *Proc. CVPR*, 2016.

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

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

[133] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Proc. NIPS*, 2015.

[134] K. Ristovski, V. Radosavljevic, S. Vucetic, and Z. Obradovic, "Continuous conditional random fields for efficient regression in large fully connected graphs." in *Proc. AAAI*, 2013.

[135] R. Vemulapalli, O. Tuzel, and M.-Y. Liu, "Deep gaussian conditional random field network: A model-based deep network for discriminative denoising," in *Proc. CVPR*, 2016.

[136] P. Knöbelreiter, C. Reinbacher, A. Shekhovtsov, and T. Pock, "End-to-end training of hybrid cnn-crf models for stereo," in *Proc. CVPR*, 2017.

[137] S. Wang, S. Fidler, and R. Urtasun, "Proximal deep structured models," in *Proc. NIPS*, 2016.

[138] A. Arnab and P. H. Torr, "Pixelwise instance segmentation with a dynamically instantiated network," in *Proc. CVPR*, 2017.

[139] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Proc. NIPS*, 2014.

[140] A. Kirillov, D. Schlesinger, S. Zheng, B. Savchynskyy, P. H. Torr, and C. Rother, "Joint training of generic cnn-crf models with stochastic optimization," in *Proc. ACCV*, 2016.

[141] X. Chu, W. Ouyang, X. Wang *et al.*, "Crf-cnn: Modeling structured information in human pose estimation," in *Proc. NIPS*, 2016.

[142] T.-H. Vu, A. Osokin, and I. Laptev, "Context-aware cnns for person head detection," in *Proc. ICCV*, 2015.

[143] Z. Hayder, X. He, and M. Salzmann, "Learning to co-generate object proposals with a deep structured network," in *Proc. CVPR*, 2016.

[144] L. C. Chen, A. Schwing, A. Yuille, and R. Urtasun, "Learning deep structured models," in *Proc. ICML*, 2015.

[145] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep structured output learning for unconstrained text recognition," in *Proc. ICLR*, 2015.

[146] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam, "Large-scale object classification using label relation graphs," in *Proc. ECCV*, 2014.

[147] N. Ding, J. Deng, K. Murphy, and H. Neven, "Probabilistic label relation graphs with ising models," in *Proc. ICCV*, 2015.

[148] Z. Wu, D. Lin, and X. Tang, "Deep markov random field for image modeling," in *Proc. ECCV*, 2016.

[149] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proc. ICLR*, 2016.

[150] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *Proc. ICCV*, 2011.

[151] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACMMM*, 2014.

[152] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016.

[153] S. Xie, X. Huang, and Z. Tu, "Top-down learning for structured labeling with convolutional pseudoprior," in *Proc. ECCV*, 2016.

[154] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *Proc. NIPS Deep Learning Workshop*, 2013.

[155] D. Li, H. Wu, J. Zhang, and K. Huang, "A2-rl: Aesthetics aware reinforcement learning for automatic image cropping," in *Proc. CVPR*, 2018.

[156] S. Lan, R. Panda, Q. Zhu, and A. K. Roy-Chowdhury, "Ffnet: Video fast-forwarding via reinforcement learning," in *Proc. CVPR*, 2018.

[157] J. Park, J.-Y. Lee, D. Yoo, and I. S. Kweon, "Distort-and-recover: Color enhancement using deep reinforcement learning," in *Proc. CVPR*, 2018.

[158] Y. Hu, H. He, C. Xu, B. Wang, and S. Lin, "Exposure: A white-box photo post-processing framework," *ACM TOG*, vol. 37, no. 2, p. 26, 2018.

[159] M. I. Razzak, S. Naz, and A. Zaib, "Deep learning for medical image processing: Overview, challenges and the future," in *Classification in BioApps*, 2018, pp. 323–350.

[160] Q. Cao, L. Lin, Y. Shi, X. Liang, and G. Li, "Attention-aware face hallucination via deep reinforcement learning," in *Proc. CVPR*, 2017.

[161] K. Yu, C. Dong, L. Lin, and C. C. Loy, "Crafting a toolchain for image restoration by deep reinforcement learning," in *Proc. CVPR*, 2018.

[162] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," in *Proc. NIPS Deep Reinforcement Learning Workshop*, 2015.

[163] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE TIP*, vol. 16, no. 8, pp. 2080–2095, 2007.

[164] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proc. ICCV*, 2009.

[165] S. Lefkimmiatis, "Non-local color image denoising with convolutional neural networks," in *Proc. CVPR*, 2017.

[166] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proc. SIGGRAPH*, 2000.

[167] S. Roth and M. J. Black, "Fields of experts: A framework for learning image priors," in *Proc. CVPR*, 2005.

[168] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE TIP*, vol. 17, no. 1, pp. 53–69, 2008.

[169] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Proc. NIPS*, 2012.

[170] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE CG&A*, vol. 21, no. 5, pp. 34–41, 2001.

[171] S. J. Hwang, A. Kapoor, and S. B. Kang, "Context-based automatic local image enhancement," in *Proc. ECCV*, 2012.

[172] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, and F. Durand, "Deep bilateral learning for real-time image enhancement," *ACM TOG*, vol. 36, no. 4, p. 118, 2017.

[173] Q. Chen, J. Xu, and V. Koltun, "Fast image processing with fully-convolutional networks," in *Proc. ICCV*, 2017.

[174] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. ICML*, 2016.

[175] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. NIPS*, 2017.

[176] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Proc. CVPR*, 2017.

[177] S. Tokui, K. Oono, and S. Hido, "Chainer: a next-generation open source framework for deep learning," in *Proc. NIPS Workshop on Machine Learning Systems*, 2015.

[178] F. Maes, L. Denoyer, and P. Gallinari, "Structured prediction with reinforcement learning," *Machine learning*, vol. 77, no. 2-3, p. 271, 2009.

[179] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang, "Waterloo exploration database: New challenges for image quality assessment models," *IEEE TIP*, vol. 26, no. 2, pp. 1004–1016, 2017.

[180] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[181] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *Proc. CVPR*, 2014.

[182] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE TPAMI*, vol. 39, no. 6, pp. 1256–1272, 2017.

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

[184] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, L. Zhang, B. Lim, S. Son, H. Kim, S. Nah, K. M. Lee *et al.*, "Ntire 2017 challenge on single image super-resolution: Methods and results," in *CVPR Workshops*, 2017, pp. 1110–1121.

[185] F. Luisier, T. Blu, and M. Unser, "Image denoising in mixed poisson–gaussian noise," *IEEE TIP*, vol. 20, no. 3, pp. 696–708, 2011.

[186] K. He, J. Sun, and X. Tang, "Guided image filtering," in *Proc. ECCV*, 2010.

[187] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Proc. CVPR*, 2005.

[188] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[189] K. Lang, “Newsweeder: Learning to filter netnews,” in *Machine Learning Proceedings*, 1995, pp. 331–339.

[190] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *Proc. CVPR*, 2017.

[191] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proc. ICCV*, 2017.

Publications

Publications related to the thesis

International Journal

[1] R. Furuta, S. Ikehata, T. Yamasaki, and K. Aizawa, "Efficiency-enhanced cost-volume filtering featuring coarse-to-fine strategy," *MTAP*, vol. 77, no. 10, pp. 12469–12491, 2018.

[2] R. Furuta, I. Tsubaki, and T. Yamasaki, "Fast volume seam carving with multi-pass dynamic programming," *IEEE TCSVT*, vol. 28, no. 5, pp. 1087-1101, 2018.

International Conference

[3] R. Furuta, N. Inoue, and T. Yamasaki, "Fully convolutional network with multi-step reinforcement learning for image processing," in *Proc. AAAI*, 2019.

[4] R. Furuta, I. Tsubaki, and T. Yamasaki, "Fast volume seam carving with multi-pass dynamic programming," in *Proc. ICIP*, 2016.

[5] R. Furuta, S. Ikehata, T. Yamasaki, and K. Aizawa, "Coarse-to-fine strategy for efficient cost-volume filtering," in *Proc. ICIP*, 2014.

Domestic Conference

[6] 古田諒佑, 椿郁子, 山崎俊彦, "Multi-pass dynamic programming for fast volume seam carving," MIRU, poster, 2016.

[7] 古田諒佑, 池畑諭, 山崎俊彦, 相澤清晴, "多段階コストボリュームフィルタリングによる高速オプティカルフロー推定," ITE年次大会, 2014.

[8] 古田諒佑, 池畑諭, 山崎俊彦, 相澤清晴, “階層的処理によるコストボリュームフィルタリングの高速化とステレオマッチング応用,” 3次元画像コンファレンス, 2014.

[9] 古田諒佑, 池畑諭, 山崎俊彦, 相澤清晴, “多段階処理によるコストボリュームフィルタリングの高速化,” IEICE総合大会, 2014.

[10] 古田諒佑, 池畑諭, 山崎俊彦, 相澤清晴, “領域分割・投票によるコストボリュームフィルタリングの高速化,” ITE冬季大会, 2013.

Publications non-related to the thesis

International Journal

[11] R. Furuta, N. Inoue, and T. Yamasaki, “Efficient and interactive spatial-semantic image retrieval,” *MTAP*, 2019. (major revision)

International Conference

[12] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa, “Cross-domain weakly-supervised object detection through progressive domain adaptation,” in *Proc. CVPR*, 2018.

[13] R. Furuta, N. Inoue, and T. Yamasaki, “Efficient and interactive spatial-semantic image retrieval,” in Proc. *MMM*, 2018.

[14] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa, “Object detection refinement using Markov random field based pruning and learning based rescoring,” in *Proc. ICASSP*, 2017.

[15] S. Shen, R. Furuta, T. Yamasaki, and K. Aizawa, “Fooling neural networks in face attractiveness evaluation: adversarial examples with high attractiveness score but low subjective score,” in *Proc. BigMM*, short paper, 2017.

[16] R. Furuta, Y. Fukushima, T. Yamasaki, and K. Aizawa, “Multi-label classification using class relations based on higher-order MRF optimization,” in *CVPRW-BigVision*, 2016.

[17] T. Yamasaki, Y. Fukushima, R. Furuta, and K. Aizawa, “Towards online prediction of oral presentation,,” in *Proc. BigMMW-ACM*, 2016.

[18] T. Yamasaki, Y. Fukushima, R. Furuta, L. Sun, K. Aizawa, and D. Bollegala, “Prediction of user ratings of oral presentations using label relations,” in *Proc. ACMMMW-ASM*, 2015.

Domestic Conference

[19] 古田諒佑, 井上直人, 山崎俊彦, 相澤清晴, “意味領域分割に基づく効率的かつインタラクティブな画像検索,” MVE, 2018.

[20] 井上直人, 古田諒佑, 山崎俊彦, 相澤清晴, “スタイル転移を利用した物体検出におけるドメイン適応,” IE研究会, 2018.

[21] 井上直人, 古田諒佑, 山崎俊彦, 相澤清晴, “疑似バウンディングボックス生成によるドメインを跨いだ物体検出,” FIT, 2017.

[22] 井上直人, 古田諒佑, 山崎俊彦, 相澤清晴, “文脈・状況を考慮した物体検出の高精度化,” PRMU, 2017.

[23] 井上直人, 古田諒佑, 山崎俊彦, 相澤清晴, “Markov random field based pruning and learning based rescoring for object detection,” MIRU, poster, 2016.

[24] 井上直人, 古田諒佑, 山崎俊彦, 相澤清晴, “類似シーン画像を用いた物体検出のフィルタリング,” IPSJ, 2016.

[25] 古田諒佑, 福島悠介, 山崎俊彦, 相澤清晴, “高階MRFによるクラス間の共起情報を用いたマルチラベル分類の精度改善,” IE研究会, 2016.

[26] 井上直人, 古田諒佑, 山崎俊彦, 相澤清晴, “クラス間の共起情報を用いた物体検出結果のフィルタリング,” ITE年次大会, 2015.

[27] 古田諒佑, 山崎俊彦, “粒子群確率伝搬法を用いたステレオマッチング,” PRMU, 2015.

[28] 山崎俊彦, 福島悠介, 古田諒佑, 相澤清晴, “文書・音声特徴によるリアルタイム・プレゼンテーション解析にむけた検討,” IMPS, 2015.

[29] 古田諒佑, 福島悠介, 山崎俊彦, 相澤清晴, “高階エネルギーのMRF最適化によるラベル共起を考慮したマルチラベル分類,” IMPS, 2015.

[30] 古田諒佑, 福島悠介, 山崎俊彦, 相澤清晴, “マルコフ確率場に基づくラベル関係性を考慮したマルチラベル分類,” CVIM研究会, 2015.

Technical descriptions

[31] 松井勇佑，佐野峻平，古田諒佑. “ICIP 2014 参加報告”. 映像情報メディア学会誌，2月号，vol. 69，pp. 136-140，2015.

Awards

[32] 古田諒佑, 福島悠介, 山崎俊彦, 相澤清晴, 2015年度 画像工学研究会 IE賞.

[33] 山崎俊彦, 福島悠介, 古田諒佑, 相澤清晴, 2015年度 IMPS優秀論文フロンティア賞.

[34] R. Furuta, S. Ikehata, T. Yamasaki, and K. Aizawa, IEEE SPS Japan Student Paper Award, 2015.

[35] 古田諒佑, 2015年度 電気・電子情報学術振興財団 原島博学術奨励賞.

[36] R. Furuta, S. Ikehata, T. Yamasaki, and K. Aizawa, Top 10% Paper Award in IEEE International Conference on Image Processing (ICIP) 2014.

[37] 古田諒佑, 2015年度 映像情報メディア学会 学生優秀発表賞.

[38] 古田諒佑, 2015年度 映像情報メディア学会 鈴木記念奨励賞.