UNIVERSITY OF CALIFORNIA, MERCED

# Learning Spatial and Temporal Visual Enhancement

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

in

Electrical Engineering and Computer Science

by

Wei-Sheng Lai

Committee in charge:

Professor Ming-Hsuan Yang, Chair
Professor Shawn Newsam
Professor Sungjin Im
Professor Jia-Bin Huang

2019

The dissertation of Wei-Sheng Lai is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Professor Shawn Newsam

Professor Sungjin Im

Professor Jia-Bin Huang

Professor Ming-Hsuan Yang Chair

University of California, Merced

2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

## VITA

| | |
|---|---|
| 2012 | B. S. in Electronics Engineering, National Taiwan University, Taipei, Taiwan |
| 2014 | M. S. in Electronics Engineering, National Taiwan University, Taipei, Taiwan |
| 2019 | Ph. D. in Electrical Engineering and Computer Science, University of California, Merced |

## PUBLICATIONS

Wei-Sheng Lai, Deqing Sun, Jinwei Gu, Orazio Gallo, Ming-Hsuan Yang, and Jan Kautz, *Learning to Stitch Videos*, in preparation for British Machine Vision Conference (BMVC), 2019.

Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang, *Depth-Aware Video Frame Interpolation*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

Lerenhan Li, Jinshan Pan, Wei-Sheng Lai, Changxin Gao, Nong Sang, and Ming-Hsuan Yang, *Blind Image Deblurring vis Deep Discriminative Priors*, International Journal of Computer Vision (IJCV), 2019.

Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang, *Learning Blind Video Temporal Consistency*, European Conference on Computer Vision (ECCV), 2018.

Xinyi Zhang, Hang Dong, Zhe Hu, Wei-Sheng Lai, Fei Wang, and Ming-Hsuan Yang, *Gated Fusion Network for Joint Image Deblurring and Super-Resolution*, British Machine Vision Conference (BMVC), 2018.

Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang, *Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2018.

Ziyi Shen, Wei-Sheng Lai, Tingfa Xu, Jan Kautz, and Ming-Hsuan Yang, *Deep Semantic Face Deblurring*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

Lerenhan Li, Jinshan Pan, Wei-Sheng Lai, Changxin Gao, Nong Sang, and Ming-Hsuan Yang, *Learning a Discriminative Prior for Blind Image Deblurring*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

Wei-Sheng Lai, Jia-Bin Huang, and Ming-Hsuan Yang, *Semi-Supervised Learning for Optical Flow with Generative Adversarial Networks*, Neural Information Processing Systems (NIPS), 2017.

Wei-Sheng Lai, Yujia Huang, Neel Joshi, Chris Buehler, Ming-Hsuan Yang and Sing Bing Kang, *Semantic-driven Generation of Hyperlapse from 360-Degree Video*, IEEE Transactions on Visualization and Computer Graphics (TVCG), 2017.

Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang, *Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

Jiawei Zhang, Jinshan Pan, Wei-Sheng Lai, Rynson Lau, Ming-Hsuan Yang, *Learning Fully Convolutional Networks for Iterative Non-blind Deconvolution*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

Wei-Sheng Lai, Jia-Bin Huang, Zhe Hu, and Ming-Hsuan Yang, *A Comparative Study for Single-Image Blind Deblurring*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

ABSTRACT OF THE DISSERTATION

# Learning Spatial and Temporal Visual Enhancement

by

Wei-Sheng Lai

Doctor of Philosophy in Electrical Engineering and Computer Science

University of California Merced, 2019

Professor Ming-Hsuan Yang, Chair

Visual enhancement is concerned with problems to improve the visual quality and viewing experience for images and videos. Researchers have been actively working on this area due to its theoretical and practical interest. However, obtaining high visual quality often comes with a cost of computational efficiency. With the growth of mobile applications and cloud services, it is crucial to develop effective and efficient algorithms for generating visually attractive images and videos. In this thesis, we address the visual enhancement problems in three aspects, including the spatial, temporal, and the joint spatial-temporal domains. We propose efficient algorithms based on deep convolutional neural networks for solving various visual enhancement problems.

First, we address the problem of **spatial** enhancement for single-image super-resolution. We propose a deep Laplacian Pyramid Network to reconstruct a high-resolution image from an input low-resolution input in a coarse-to-fine manner. Our model directly extracts features from input LR images and progressively reconstructs the sub-band residuals. We train the proposed model with a multi-scale training, deep supervision, and robust loss functions to achieve the state-of-the-art performance. Furthermore, we exploit the recursive learning technique to share parameters across and within pyramid levels to significantly reduce the model parameters. As most of the operations are performed on a low-resolution space, our model requires less memory and runs faster than state-of-the-art methods.

Second, we address the **temporal** enhancement problem by learning the temporal consistency in videos. Given an input video and a per-frame processed video (processed by an existing image-based algorithm), we learn a recurrent network to reduce the temporal flickering and generate a temporally consistent video. We train the proposed network by minimizing both short-term and long-term temporal losses as well as a perceptual loss to strike a balance between temporal coherence and perceptual similarity with the processed frames. At test time, our model does not require computing optical flow and thus runs at 400+ FPS on GPU for high-resolution videos. Our model is task independent, where a single model can handle multiple and unseen tasks, including but not limited to artistic style transfer, enhancement, colorization, image-to-image translation and intrinsic image decomposition.

Third, we address the **spatial-temporal** enhancement problem for video stitching. Inspired by the pushbroom cameras, we cast the stitching as a spatial interpolation problem. We propose a pushbroom stitching network to learn dense flow fields to smoothly align the input videos. The stitched videos can be generated from an efficient pushbroom interpolation layer. Our approach generates more temporally stable and visually pleasing results than existing video stitching approaches and commercial software. Furthermore, our algorithm has immediate applications in many areas such as virtual reality, immersive telepresence, autonomous driving, and video surveillance.

# Chapter 1

# Introduction

## 1.1 Overview

With the growth of mobile cameras (e.g., smart phones, GoPro, and $360°$ camera) and the wide spread of social media (e.g., Facebook, Instagram, and YouTube), millions of photos and videos are captured and uploaded to the Internet every day. However, many photos may suffer from artifacts (i.e., blurriness, noise, low spatial resolutions, or temporal flickering) or visual obstructions (i.e., limited field-of-view, reflection or occlusion). To address these issues, several problems have been studied in the field of computer vision, such as image super-resolution, motion deblurring, inpainting, video frame interpolation and video stitching and stabilization.

Conventional algorithms typically rely on a variety of priors or assumptions, e.g., total variation, sparse representation, self-similarity, brightness constancy and spatial smoothness, and develop complex optimization frameworks to solve the visual enhancement problems. In recent years, data-driven approaches have been shown more effective to learn priors from a large image or video datasets. In particular, the deep convolutional neural networks (CNNs) have demonstrated great performance in many high-level as well as low-level vision problems due to its strong learning capacity. In this thesis, we propose efficient algorithms based on deep CNNs for solving visual enhancement problems in the following three aspects.

**Spatial enhancement.** Existing CNN-based single image super-resolution algorithms typically require a large number of network parameters and entail heavy computational loads for generating high-accuracy super-resolution results. Therefore, we propose a deep Laplacian Pyramid Network which performs fast and accurate single-image super-resolution.

**Temporal enhancement.** Applying image-based algorithms independently to each frame of a video often leads to temporally inconsistent results. On the other hand, task-specific video-based algorithms usually cannot be applied or generalized to different applications. Therefore, we propose a learning-based task-independent approach with a deep recurrent network for enforcing temporal consistency in a video.

**Spatial-temporal enhancement.** Despite the long history of image and video stitching research, existing academic and commercial solutions still produce strong artifacts due to the challenges of handling parallax. To address these issues, we propose a pushbroom stitching network by casting the stitching as a spatial interpolation problem.

## 1.2 Organization

In Chapter 2, we review and discuss the pros and cons of existing methods in the above three aspects.

In Chapter 3, we propose a deep Laplacian Pyramid Super-Resolution Network (LapSRN) that has both high reconstruction accuracy and fast execution speed to perform single-image super-resolution. Our model directly extracts features from the input LR images and progressively reconstructs the sub-band residuals of high-resolution images in a coarse-to-fine manner. Furthermore, we adopt a multi-scale deep supervision and a robust loss function to improve the reconstruction accuracy. As most of the operations are performed on a low-resolution space, our LapSRN requires less memory and runs faster than state-of-the-art methods. We further extend our LapSRN to incorporate recursive layers, local skip connections and multi-scale training to significantly improve the performance. By sharing parameters across and within pyramid levels, we reduce $73\%$ of the network parameters while achieving better reconstruction accuracy. Extensive quantitative and qualitative evaluations on benchmark datasets demonstrate that the proposed algorithm performs favorably against the state-of-the-art methods in terms of speed and accuracy.

In Chapter 4, we propose a generic approach with a deep recurrent network to reduce the temporal flickering in a per-frame processed video. Our model takes as inputs the original and per-frame processed videos (processed by an existing image-based algorithm) and generate a temporally consistent video. We train the proposed network by minimizing both short-term and long-term temporal losses as well as a perceptual loss to strike a balance between temporal coherence and perceptual similarity with the processed frames. At test time, our model does not require computing optical flow and thus runs at 400+ FPS on GPU for high-resolution videos. The proposed method is agnostic to specific image processing algorithms applied to the original video. Therefore, a single model can handle multiple and unseen tasks, including but not limited to artistic style transfer, enhancement, colorization, image-to-image translation and intrinsic image decomposition.

In Chapter 5, we propose a video stitching algorithm that is temporally stable and tolerant to strong parallax. Our key insight is that stitching can be cast as a problem of learning a smooth spatial interpolation between the input videos, inspired by the pushbroom cameras. We introduce a fast pushbroom interpolation layer and propose a novel pushbroom stitching network, which learns a dense flow field to smoothly align the input videos. Our approach generates more visually pleasing results than existing approaches and has immediate applications in many areas such as virtual reality, immersive telepresence, autonomous driving, and video surveillance.

In Chapter 6, we conclude the contributions in this thesis and discuss several future research directions, including the practical applications of low-level vision, semi-supervised learning and domain adaptation, and learning-based computational photography.

# Chapter 2

# Literature Review

In this chapter, we review the literature related to the research work presented in the following chapters.

## 2.1 Single Image Super-Resolution

Single-image super-resolution has been extensively studied in the literature [120]. Here we focus our discussion on recent example-based and CNN-based approaches.

### 2.1.1 SR Using Internal Databases

Several methods [31, 119, 35] exploit the self-similarity property in natural images and construct LR-HR patch pairs based on the scale-space pyramid of the LR input image. While internal databases contain more relevant training patches than external image datasets, the number of LR-HR patch pairs may not be sufficient to cover large textural appearance variations in an image. Singh et al. [100] decompose patches into directional frequency sub-bands and determine better matches in each sub-band pyramid independently. Huang et al. [47] extend the patch search space to accommodate the affine and perspective deformation. The SR methods based on internal databases are typically slow due to the heavy computational cost of patch searches in the scale-space pyramid. Such drawbacks make these approaches less feasible for applications that require computational efficiency.

### 2.1.2 SR Using External Databases

Numerous SR methods learn the LR-HR mapping with image pairs collected from external databases using supervised learning algorithms, such as nearest neighbor [32], manifold embedding [8, 18], kernel ridge regression [59], and sparse representation [122, 123, 128]. Instead of directly modeling the complex patch space over the entire database, recent methods partition the image set by K-means [121], sparse dictionary [106, 105] or random forest [95], and learn locally linear regressors for each cluster. While these approaches are effective and efficient, the extracted features and mapping functions are hand-designed, which may not be optimal for generating high-quality SR images.

### 2.1.3 CNN-based SR

CNN-based SR methods have demonstrated state-of-the-art results by jointly optimizing the feature extraction, non-linear mapping, and image reconstruction stages in an end-to-end manner. The VDSR network [57] shows significant improvement over the SRCNN method [24] by increasing the network depth from 3 to 20 convolutional layers. To facilitate training a deeper model with a fast convergence speed, the VDSR method adopts the global residual learning paradigm to predict the differences between the ground truth HR image and the bicubic upsampled LR image instead of the actual pixel values. Wang et al. [113] combine the domain knowledge of sparse coding with a deep CNN and train a cascade network (SCN) to upsample images progressively. In [58], Kim et al. propose a network with multiple recursive layers (DRCN) with up to 16 recursions. The DRRN approach [103] further trains a 52-layer network by extending the local residual learning approach of the ResNet [41] with deep recursion. We note that the above methods use bicubic interpolation to pre-upsample input LR images *before* feeding into the deep networks, which increases the computational cost and requires a large amount of memory.

To achieve real-time speed, the ESPCN method [98] extracts feature maps in the LR space and replaces the bicubic upsampling operation with an efficient sub-pixel convolution (i.e., pixel shuffling). The FSRCNN method [25] adopts a similar idea and uses a hourglass-shaped CNN with transposed convolutional layers for upsampling. As a trade-off of speed, both ESPCN [98] and FSRCNN [25] have limited network capacities for learning complex mappings. Furthermore, these methods upsample images or features in *one upsampling* *step* and use only one supervisory signal from the target upsampling scale. Such a design often causes difficulties in training models for large upsampling scales (e.g., $4\times$ or $8\times$). In contrast, our model progressively upsamples input images on *multiple* pyramid levels and use *multiple* losses to guide the prediction of sub-band residuals at each level, which leads to accurate reconstruction, particularly for large upsampling scales.

All the above CNN-based SR methods optimize networks with the $\mathcal{L}_2$ loss function, which often leads to over-smooth results that do not correlate well with human perception. We demonstrate that the proposed deep network with the robust Charbonnier loss function better handles outliers and improves the SR performance over the $\mathcal{L}_2$ loss function. Most recently, Lim et al. [75] propose a multi-scale deep SR model (MDSR) by extending ESPCN [98] with three branches for scale-specific upsampling but sharing most of the parameters across different scales. The MDSR method is trained on a high-resolution DIV2K [104] dataset (800 training images of 2k resolution), and achieves the state-of-the-art performance. Table 2.1 shows the main components of the existing CNN-based SR methods. The proposed LapSRN and MS-LapSRN are listed on the last two rows.

### 2.1.4 Laplacian Pyramid

The Laplacian pyramid has been widely used in several vision tasks, including image blending [15], texture synthesis [43], edge-aware filtering [84] and semantic segmentation [34]. Denton et al. [23] propose a generative adversarial network based on a Laplacian pyramid framework (LAPGAN) to generate realistic images, which is the most related to our work. However, the proposed LapSRN differs from LAPGAN in two aspects.

First, the *objectives* of the two models are different. The LAPGAN is a generative model which is designed to synthesize diverse natural images from random noise and sample inputs. On the contrary, the proposed LapSRN is a super-resolution model that predicts a particular HR image based on the given LR image and upsampling scale factor. The LAPGAN uses a cross-entropy loss function to encourage the output images to respect the data distribution of the training datasets. In contrast, we use the Charbonnier penalty function to penalize the deviation of the SR prediction from the ground truth HR images.

Second, the differences in *architecture designs* result in disparate inference speed and network capacities. The LAPGAN upsamples input images *before* applying convolution

Table 2.1: **Feature-by-feature comparisons of CNN-based SR algorithms.** Methods with direct reconstruction performs one-step upsampling from the LR to HR space, while progressive reconstruction predicts HR images in multiple upsampling steps. Depth represents the number of convolutional and transposed convolutional layers in the longest path from input to output for $4\times$ SR. Global residual learning (GRL) indicates that the network learns the difference between the ground truth HR image and the upsampled (i.e., using bicubic interpolation or learned filters) LR images. Local residual learning (LRL) stands for the local skip connections between intermediate convolutional layers.

| Method | Input | Depth | Filters | Parameters | GRL | LRL | Multi-scale training | Loss function |
|---:|:---:|:---:|:---:|:---:|:---:|:---:|:---:|:---:|
| SRCNN [24] | LR + bicubic | 3 | 64 | 57k | | | | $\mathcal{L}_2$ |
| FSRCNN [25] | LR | 8 | 56 | 12k | | | | $\mathcal{L}_2$ |
| ESPCN [98] | LR | 3 | 64 | 20k | | | | $\mathcal{L}_2$ |
| SCN [113] | LR + bicubic | 10 | 128 | 42k | | | | $\mathcal{L}_2$ |
| VDSR [57] | LR + bicubic | 20 | 64 | 665k | ✓ | | ✓ | $\mathcal{L}_2$ |
| DRCN [58] | LR + bicubic | 20 | 256 | 1775k | ✓ | | | $\mathcal{L}_2$ |
| DRRN [103] | LR + bicubic | 52 | 128 | 297k | ✓ | ✓ | ✓ | $\mathcal{L}_2$ |
| MDSR [75] | LR | 162 | 64 | 8000k | | ✓ | ✓ | Charbonnier |
| LapSRN (ours) | LR | 24 | 64 | 812k | ✓ | | | Charbonnier |
| MS-LapSRN (ours) | LR | 84 | 64 | 222k | ✓ | ✓ | ✓ | Charbonnier |

at each level, while our LapSRN extracts features directly from the LR space and upscales images at the end of each level. Our network design effectively alleviates the computational cost and increases the size of receptive fields. In addition, the convolutional layers at each level in our LapSRN are *connected* through multi-channel transposed convolutional layers. The residual images at a higher level are therefore predicted by a deeper network with shared feature representations at lower levels. The shared features at lower levels increase the non-linearity at finer convolutional layers to learn complex mappings.

### 2.1.5 Adversarial Training

The Generative Adversarial Networks (GANs) [36] have been applied to several image reconstruction and synthesis problems, including image inpainting [86], face completion [74], and face super-resolution [126]. Ledig et al. [66] adopt the GAN framework for learning natural image super-resolution. The ResNet [41] architecture is used as the generative network and train the network using the combination of the $\mathcal{L}_2$ loss, perceptual loss [56], and adversarial loss. The SR results may have lower PSNR but are visually plausible. Note that our LapSRN can be easily extended to incorporate adversarial training.

## 2.2 Video Temporal Consistency

We address the temporal consistency problem on a wide range of applications, including automatic white balancing [44], harmonization [9], dehazing [39], image enhancement [33], style transfer [48, 56, 72], colorization [50, 130], image-to-image translation [52, 132], and intrinsic image decomposition [7]. In the following, we discuss task-specific and task-independent approaches that enforce temporal consistency on videos.

### 2.2.1 Task-Specific Approaches

A common solution to embed the temporal consistency constraint is to use optical flow to propagate information between frames, e.g., colorization [71] and intrinsic decomposition [125]. However, estimating optical flow is computationally expensive and thus is impractical to apply on high-resolution and long sequences. Temporal filtering is an efficient approach to extend image-based algorithms to videos, e.g., tone-mapping [5], color

Table 2.2: **Comparison of blind temporal consistency methods**. Both the methods of Bonneel et al. [11] and Yao et al. [124] require dense correspondences from optical flow or PatchMatch [6], while the proposed method does not explicitly rely on these correspondences at test time. The algorithm of Yao et al. [124] involves a key-frame selection from the entire video and thus cannot generate output in an online manner.

| | Bonneel et al. [11] | Yao et al. [124] | Ours |
|---|---|---|---|
| Content constraint | gradient | local affine | perceptual loss |
| Short-term temporal constraint | ✓ | - | ✓ |
| Long-term temporal constraint | - | ✓ | ✓ |
| Require dense correspondences at test time | ✓ | ✓ | - |
| Online processing | ✓ | - | ✓ |

transfer [10], and visual saliency [65] to generate temporally consistent results. Nevertheless, these approaches assume a specific filter formulation and cannot be generalized to other applications.

Recently, several approaches have been proposed to improve the temporal stability of CNN-based image style transfer. Huang et al. [45] and Gupta et al. [37] train feed-forward networks by jointly minimizing content, style and temporal warping losses. These methods, however, are limited to the specific styles used during training. Chen et al. [19] learn flow and mask networks to adaptively blend the intermediate features of the pre-trained style network. While the architecture design is independent of the style network, it requires the access to intermediate features and cannot be applied to non-differentiable tasks. In contrast, the proposed model is entirely blind to specific algorithms applied to the input frames and thus is applicable to optimization-based techniques, CNN-based algorithms, and combinations of Photoshop filters.

### 2.2.2 Task-Independent Approaches

Several methods have been proposed to improve temporal consistency for multiple tasks. Lang et al. [65] approximate global optimization of a class of energy formulation (e.g., colorization, optical flow estimation) via temporal edge-aware filtering. In [26], Dong et al. propose a segmentation-based algorithm and assume that the image transformation is spatially and temporally consistent. More general approaches assume gradient similarity [11] or local affine transformation [124] between the input and the processed frames. These methods, however, cannot handle more complicated tasks (e.g., artistic style transfer). In contrast, we use the VGG perceptual loss [56] to impose high-level perceptual similarity between the output and processed frames. We list the feature-by-feature comparisons between Bonneel et al. [11], Yao et al. [124] and the proposed method in Table 2.2.

## 2.3 Video Stitching

We first review the most relevant work on image and video stitching and then discuss the conventional pushbroom camera, which inspires the proposed algorithm.

### 2.3.1 Image Stitching

Existing image stitching methods often build on the conventional pipeline of Brown and Lowe et al. [13], which first estimates a 2D transformation (e.g., homography) for alignment and then stitches images using the seam-cutting [28] or multi-band blending [16]. However, ghosting artifacts and mis-alignment still exist, especially when input images have large parallax. To account for parallax, several methods adopt spatially varying local warping based on the affine [78] or projective [127] transformations. Zhang et al. [129] integrate the content-preserving warping and seam-cutting algorithms to handle parallax while avoiding local distortions. More recent methods combine the homography and similarity transforms [17, 76] to reduce the projective distortion (i.e., stretched shapes) or adopt a global similarity prior [22] to preserve the global shape of the whole stitched images.

While the above techniques are effective at creating a panorama from still images, applying these algorithms to stitch a video frame-by-frame results in a significant amount of temporal instability. In contrast, the proposed algorithm stitches each frame individually but is able to generate spatio-temporally coherent stitched videos due to the design of the pushbroom interpolation layer.

### 2.3.2 Video Stitching

Due to computational efficiency, it is not straightforward to enforce spatio-temporal consistency in existing image stitching algorithms. Commercial software, e.g., VideoStitch Studio [110] or AutoPano Video [4], often finds a fixed transformation (with camera calibration) to align all the frames, but cannot align local content well. Recent methods integrate local warping and optical flow [89] or find a spatio-temporal content-preserving warping [55] to stitch videos, which are computationally expensive. Lin et al. [77] stitch videos captured from hand-held cameras based on 3D scene reconstruction, which is also time-consuming. On the other hand, several approaches, e.g., Rich360 [69] and Google Jump [2], create $360°$ videos from multiple videos captured on a structured rig. Recently, NVIDIA provides a toolkit, VRWorks [83], to efficiently stitch videos based on depth and motion estimation. Still several artifacts, e.g., broken objects and ghosting, are visible on the stitched video.

Different from existing methods, the proposed algorithm learns locally adaptive warping based on a deep CNN to effectively and efficiently align the input views. The warping is learned to optimize the quality of the stitched video in an end-to-end fashion.

### 2.3.3 Pushbroom Panorama

The linear pushbroom camera [38] is mounted on a moving platform, e.g., satellite, and moves along a straight line. At each time stamp, the sensor captures 1D images of the viewing plane, e.g., surface of the earth. The stack of these 1D images constitutes the 2D panoramic image. The pushbroom camera has been widely used to create satellite images or panorama for street scenes [96]. It works well when the scene is far from the sensor or has nearly uniform depth. However, distortions (e.g., stretched or squashed objects) appear when the captured scene has large depth variations or dynamic objects. Several methods handle this issue by estimating scene depth [92], finding a cutting-seam on the space-time volume [114], or optimizing the viewpoint for each pixel [1].

The proposed method is a software simulation of the pushbroom camera to create panoramas and synthesizes the scan of a scene through spatial interpolation. We further use a refinement network to reduce artifacts created by the interpolation process and learn the entire model end-to-end to optimize the stitched view from a realistic synthetic dataset.

# Chapter 3

# Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks

## 3.1 Introduction

Single image super-resolution (SR) aims to reconstruct a high-resolution (HR) image from one single low-resolution (LR) input image. Example-based SR methods have demonstrated the state-of-the-art performance by learning a mapping from LR to HR image patches using large image datasets. Numerous learning algorithms have been applied to learn such a mapping function, including dictionary learning [122, 123], local linear regression [105, 121], and random forest [95], to name a few.

Convolutional Neural Networks (CNNs) have been widely used in vision tasks ranging from object recognition [41], segmentation [79], optical flow [30], to super-resolution. In [24], Dong et al. propose a Super-Resolution Convolutional Neural Network (SRCNN) to learn a nonlinear LR-to-HR mapping function. This network architecture has been extended to embed a sparse coding model [113], increase network depth [57], or apply recursive layers [58, 103]. While these models are able to generate high-quality SR images, there remain three issues to be addressed. First, these methods use a pre-defined upsampling operator, e.g.bicubic interpolation, to upscale an input LR image to the desired spatial resolution *before* applying a network for predicting the details (Figure 3.1(a)). This pre-upsampling step increases unnecessary computational cost and does not provide additional high-frequency information for reconstructing HR images. Several algorithms accelerate the SRCNN by extracting features directly from the input LR images (Figure 3.1(b)) and replacing the pre-defined upsampling operator with sub-pixel convolution [98] or transposed convolution [25] (also named as deconvolution in some literature). These methods, however, use relatively small networks and cannot learn complicated mappings well due to the limited model capacity. Second, existing methods optimize the networks with an $\mathcal{L}_2$ loss (i.e., mean squared error loss). Since the same LR patch may have multiple corresponding HR patches and the $\mathcal{L}_2$ loss fails to capture the underlying multi-modal distributions of HR patches, the reconstructed HR images are often over-smoothed and inconsistent to human visual perception on natural images. Third, existing methods mainly reconstruct HR images in *one upsampling step*, which makes learning mapping functions for large scaling factors (e.g., $8\times$) more difficult.

To address these issues, we propose the deep Laplacian Pyramid Super-Resolution Network (LapSRN) to progressively reconstruct HR images in a coarse-to-fine fashion. As shown in Figure 3.1(c), our model consists of a feature extraction branch and an image reconstruction branch. The feature extraction branch uses a cascade of convolutional layers to extract non-linear feature maps from LR input images. We then apply a transposed convolutional layer for upsampling the feature maps to a finer level and use a convolutional layer to predict the sub-band residuals (i.e., the differences between the upsampled image and the ground truth HR image at the respective pyramid level). The image reconstruction branch upsamples the LR images and takes the sub-band residuals from the feature extraction branch to efficiently reconstruct HR images through element-wise addition. Our network architecture naturally accommodates deep supervision (i.e., supervisory signals can be applied simultaneously at each level of the pyramid) to guide the reconstruction of HR images. Instead of using the $\mathcal{L}_2$ loss function, we propose to train the network with the robust Charbonnier loss functions to better handle outliers and improve the performance. While both feature extraction and image reconstruction branches have multiple levels, we train the network in an end-to-end fashion without stage-wise optimization.

Our algorithm differs from existing CNN-based methods in the following three aspects:

1. **Accuracy**. Instead of using a pre-defined upsampling operation, our network jointly optimizes the deep convolutional layers and upsampling filters for both images and

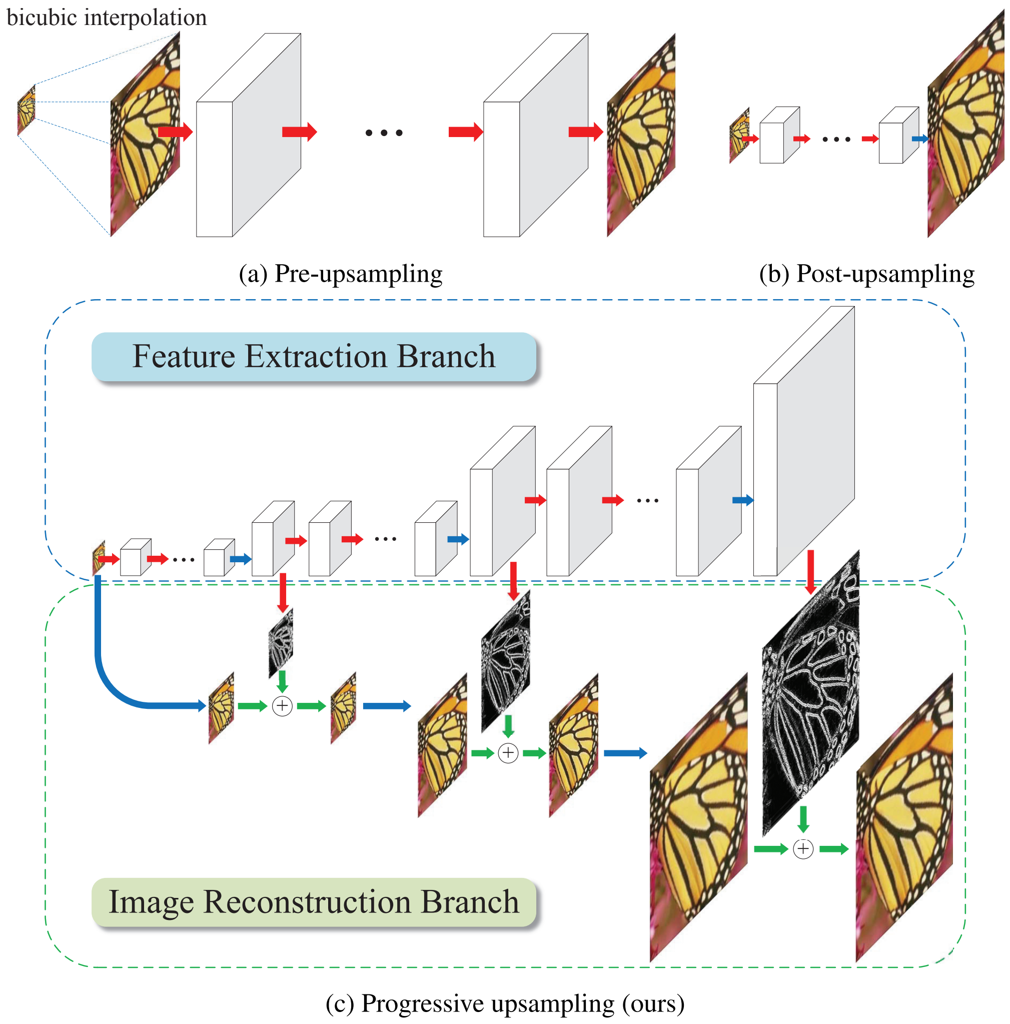

bicubic interpolation
(a) Pre-upsampling
(b) Post-upsampling
Feature Extraction Branch
Image Reconstruction Branch
(c) Progressive upsampling (ours)

Figure 3.1: **Comparisons of upsampling strategies in CNN-based SR algorithms.** Red arrows indicate convolutional layers. Blue arrows indicate transposed convolutions (upsampling), and green arrows denote element-wise addition operators. (a) Pre-upsampling based approaches (e.g., SRCNN [24], VDSR [57], DRCN [58], DRRN [103]) typically use the bicubic interpolation to upscale LR input images to the target spatial resolution before applying deep networks for prediction and reconstruction. (b) Post-upsampling based methods directly extract features from LR input images and use sub-pixel convolution [98] or transposed convolution [25] for upsampling. (c) Progressive upsampling approach using the proposed Laplacian pyramid network reconstructs HR images in a coarse-to-fine manner.

feature maps by minimizing the Charbonnier loss function. As a result, our model has a large capacity to learn complicated mappings and effectively reduces the undesired artifacts caused by spatial aliasing.

2. **Speed**. Our LapSRN accommodates both fast processing speed and high capacity of deep networks. Experimental results demonstrate that our method is faster than several CNN-based super-resolution models, e.g., VDSR [57], DRCN [58], and DRRN [103]. The proposed model achieves real-time performance as FSRCNN [25] while generating significantly better reconstruction accuracy.

3. **Progressive reconstruction**. Our model generates multiple intermediate SR predictions in *one* feed-forward pass through progressive reconstruction. This characteristic renders our method applicable to a wide range of tasks that require resource-aware adaptability. For example, the same network can be used to enhance the spatial resolution of videos depending on the available computational resources. For scenarios with limited computing resources, our $8\times$ model can still perform $2\times$ or $4\times$ SR by simply bypassing the computation of residuals at finer levels. Existing CNN-based methods, however, do not offer such flexibility.

In addition, we exploit the following techniques to substantially improve our LapSRN:

1. **Parameter sharing**. We re-design our network architecture to share parameters *across* pyramid levels and *within* the feature extraction sub-network via recursion. Through parameter sharing, we reduce $73\%$ of the network parameters while achieving better reconstruction accuracy on benchmark datasets.

2. **Local skip connections**. We systematically analyze three different approaches for applying local skip connections in the proposed model. By leveraging proper skip connections to alleviate the gradient vanishing and explosion problems, we are able to train an 84-layer network to achieve the state-of-the-art performance.

3. **Multi-scale training**. Unlike in the preliminary work where we train three different models for handling $2\times$, $4\times$ and $8\times$ SR, respectively, we train one *single* model to handle *multiple* upsampling scales. The multi-scale model learns the inter-scale correlation and improves the reconstruction accuracy against single-scale models.

We refer to our multi-scale model as MS-LapSRN.

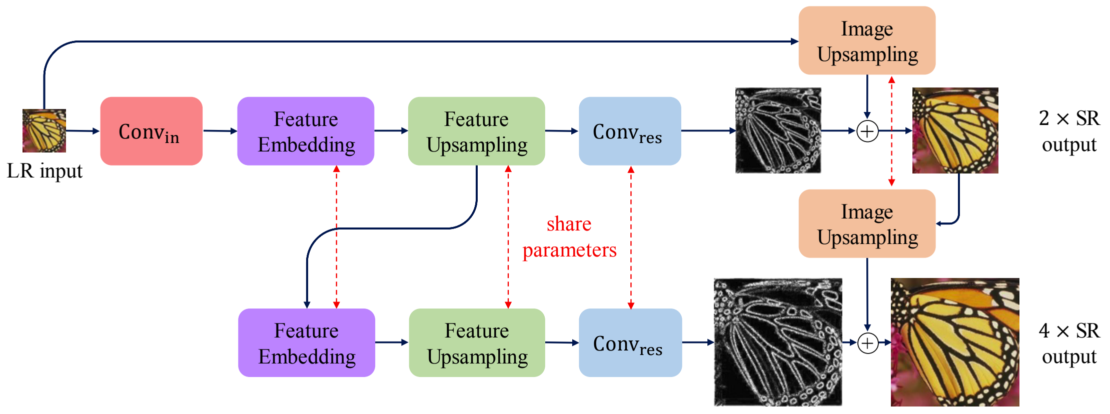


Figure 3.2: **Detailed network architecture of the proposed LapSRN.** At each pyramid level, our model consists of a feature embedding sub-network for extracting non-linear features, transposed convolutional layers for upsampling feature maps and images, and a convolutional layer for predicting the sub-band residuals. As the network structure at each level is highly similar, we share the weights of those components across pyramid levels to reduce the number of network parameters.

## 3.2 Deep Laplacian Pyramid Network for SR

In this section, we describe the design methodology of the proposed LapSRN, including the network architecture, parameter sharing, loss functions, multi-scale training strategy, and details of implementation as well as network training.

### 3.2.1 Network Architecture

We construct our network based on the Laplacian pyramid framework. Our model takes an LR image as input (rather than an upscaled version of the LR image) and progressively predicts residual images on the $\log_2 S$ pyramid levels, where $S$ is the upsampling scale factor. For example, our network consists of $3$ pyramid levels for super-resolving an LR image at a scale factor of $8$. Our model consists of two branches: (1) feature extraction and (2) image reconstruction.

**Feature extraction branch.** As illustrated in Figure 3.1(c) and Figure 3.2, the feature extraction branch consists of (1) a feature embedding sub-network for transforming high-dimensional non-linear feature maps, (2) a transposed convolutional layer for upsampling the extracted features by a scale of 2, and (3) a convolutional layer ($\text{Conv}_{\text{res}}$) for predicting the sub-band residual image. The first pyramid level has an additional convolutional layer ($\text{Conv}_{\text{in}}$) to extract high-dimensional feature maps from the input LR image. At other levels, the feature embedding sub-network directly transforms features from the upscaled feature maps at the previous pyramid level. Unlike the design of the LAPGAN, we do not *collapse* the feature maps into an image before feeding into the next level. Therefore, the feature representations at lower levels are connected to higher levels and thus can increase the non-linearity of the network to learn complex mappings at the finer levels. Note that we perform the feature extraction at the *coarse* resolution and generate feature maps at the *finer* resolution with only one transposed convolutional layer. In contrast to existing networks (e.g., [57, 103]) that perform all feature extraction and reconstruction at the finest resolution, our network design significantly reduces the computational complexity.

**Image reconstruction branch.** At level $s$, the input image is upsampled by a scale of 2 with a transposed convolutional layer, which is initialized with a $4 \times 4$ bilinear kernel. We then combine the upsampled image (using element-wise summation) with the predicted residual image to generate a high-resolution output image. The reconstructed HR image at level $s$ is then used as an input for the image reconstruction branch at level $s+1$. The entire network is a cascade of CNNs with the same structure at each level. We jointly optimize the upsampling layer with all other layers to learn better a upsampling function.

### 3.2.2 Feature Embedding Sub-network

In our preliminary work [61], we use a stack of multiple convolutional layers as our feature embedding sub-network. In addition, we learn distinct sets of convolutional filters for feature transforming and upsampling at different pyramid levels. Consequently, the number of network parameters increases with the depth of the feature embedding sub-network and the upsampling scales, e.g., the $4\times$ SR model has about twice number of parameters than the $2\times$ SR model. We explore two directions to reduce the network parameters of LapSRN.

**Parameter sharing across pyramid levels.** Our first strategy is to share the network parameters *across* pyramid levels as the network at each level shares the same structure and the task (i.e., predicting the residual images at $2\times$ resolution). As shown in Figure 3.2, we share the parameters of the feature embedding sub-network, upsampling layers, and the residual prediction layers across all the pyramid levels. As a result, the number of network parameters is independent of the upsampling scales. We can use a single set of parameters to construct multi-level LapSRN models to handle different upsampling scales.

**Parameter sharing within pyramid level.** Our second strategy is to share the network parameters *within* each pyramid level. Specifically, we extend the feature embedding sub-network using deeply recursive layers to effectively increase the network depth without increasing the number of parameters. The design of recursive layers has been adopted by several recent CNN-based SR approaches. The DRCN method [58] applies a *single* convolutional layer repeatedly up to 16 times. However, with a large number of filters (i.e., 256 filters), the DRCN is memory-demanding and slow at runtime. Instead of reusing the weights of a single convolutional layer, the DRRN [103] method shares the weights of a *block* (2 convolutional layers with 128 filters). In addition, the DRRN introduces a variant of local residual learning from the ResNet [41]. Specifically, the identity branch of the ResNet comes from the output of the *previous* block, while the identity branch of the DRRN comes from the input of the *first* block. Such a local skip connection in the DRRN creates multiple short paths from input to output and thereby effectively alleviates the gradient vanishing and exploding problems. Therefore, DRRN has 52 convolutional layers with only 297k parameters.

In the proposed LapSRN, the feature embedding sub-network has $R$ recursive blocks. Each recursive block has $D$ distinct convolutional layers, which controls the number of parameters in the entire model. The weights of the $D$ convolutional layers are shared among the recursive blocks. Given an upsampling scale factor $S$, the depth of the LapSRN can be computed by:

$$\text{depth} = (D \times R + 1) \times L + 2, \tag{3.1}$$

where $L = \log_2 S$. The $1$ within the parentheses represents the transposed convolutional layers, and the 2 at the end of (3.1) represents the first convolutional layer applied on input images and the last convolutional layer for predicting residuals. Here we define the depth of a network as the longest path from input to output.

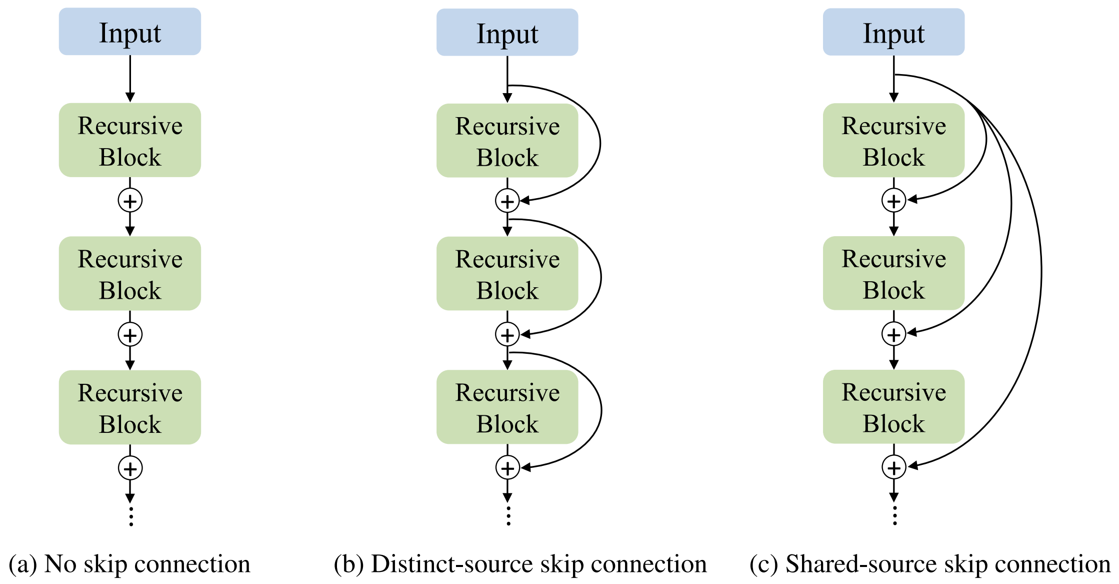


Figure 3.3: **Local residual learning.** We explore three different ways of local skip connection in the feature embedding sub-network for training deeper models.

**Local residual learning.** As the gradient vanishing and exploding problem are common issues when training deep models, we explore three different methods of local residual learning in our feature embedding sub-network to stabilize our training process:

1. **No skip connection**: A plain network without any local skip connection. We denote our LapSRN without skip connections as LapSRN$_{\textbf{NS}}$.

2. **Distinct-source skip connection**: The ResNet-style local skip connection. We denote our LapSRN with such skip connections as LapSRN$_{\textbf{DS}}$.

3. **Shared-source skip connection**: The local skip connection introduced by DRRN [103]. We denote our LapSRN with such skip connections as LapSRN$_{\textbf{SS}}$.

We illustrate the three local residual learning methods in Figure 3.3 and the detailed structure of our recursive block in Figure 3.4. We use the pre-activation structure [42] without the batch normalization layer in our recursive block.

### 3.2.3 Loss Function

Let $x$ be the input LR image and $\theta$ be the set of network parameters to be optimized. Our goal is to learn a mapping function $f$ for generating an HR image $\hat{y} = f(x;\theta)$ that is as

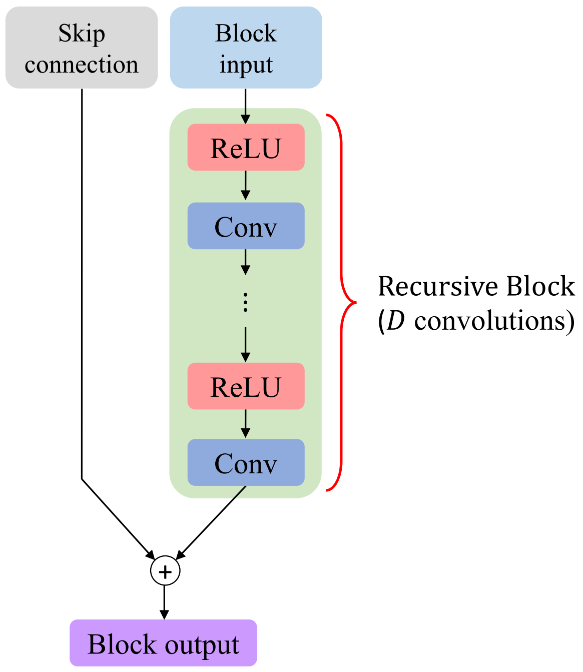


Figure 3.4: **Structure of our recursive block.** There are $D$ convolutional layers in a recursive block. The weights of convolutional layers are distinct within the block but shared among all recursive blocks.

similar to the ground truth HR image $y$ as possible. We denote the residual image at level $l$ by $\hat{r}_l$, the upscaled LR image by $x_l$ and the corresponding HR images by $\hat{y}_l$. The desired output HR images at level $l$ is modeled by $\hat{y}_l = x_l + \hat{r}_l$. We use the bicubic downsampling to resize the ground truth HR image $y$ to $y_l$ at each level. Instead of minimizing the mean square errors between $\hat{y}_l$ and $y_l$, we use a robust loss function to handle outliers. The overall loss function is defined as:

$$\begin{aligned}\mathcal{L}_S(y,\hat{y};\theta) &= \frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{L}\rho\left(y_l^{(i)} - \hat{y}_l^{(i)}\right)\\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{L}\rho\left((y_l^{(i)} - x_l^{(i)}) - \hat{r}_l^{(i)}\right),\end{aligned} \tag{3.2}$$

where $\rho(x) = \sqrt{x^2+\epsilon^2}$ is the Charbonnier penalty function (a differentiable variant of $\mathcal{L}_1$ norm) [14], $N$ is the number of training samples in each batch, $S$ is the target upsampling scale factor, and $L = \log_2 S$ is the number of pyramid levels in our model. We empirically set $\epsilon$ to $1e-3$.

In the proposed LapSRN, each level $s$ has its own loss function and the corresponding ground truth HR image $y_s$. This multi-loss structure resembles the deeply-supervised networks for classification [67] and edge detection [115]. The deep multi-scale supervision guides the network to reconstruct HR images in a coarse-to-fine fashion and reduce spatial aliasing artifacts.

### 3.2.4 Multi-Scale Training

The multi-scales SR models (i.e., trained with samples from multiple upsampling scales simultaneously) have been shown more effective than single-scale models as SR tasks have inter-scale correlations. For pre-upsampling based SR methods (e.g., VDSR [57] and DRRN [103]), the input and output of the network have the same spatial resolution, and the outputs of different upsampling scales are generated from the *same* layer of the network. In the proposed LapSRN, samples of different upsampling scales are generated from *different* layers and have different spatial resolutions. We use $2\times$, $4\times$, and $8\times$ SR samples to train a multi-scale LapSRN model. We construct a 3-level LapSRN model and minimize the combination of loss functions from three different scales:

$$\mathcal{L}(y, \hat{y}; \theta) = \sum_{S \in \{2,4,8\}} \mathcal{L}_S(y, \hat{y}; \theta). \tag{3.3}$$

We note that the pre-upsampling based SR methods could apply scale augmentation for arbitrary upsampling scales, while in our LapSRN, the upsampling scales for training are limited to $2^n\times$ SR where $n$ is an integer.

### 3.2.5 Implementation and Training Details

In the proposed LapSRN, we use 64 filters in all convolutional layers except the first layer applied on the input LR image, the layers for predicting residuals, and the image upsampling layer. The filter size of the convolutional and transposed convolutional layers are $3 \times 3$ and $4 \times 4$, respectively. We pad zeros around the boundaries before applying convolution to keep the size of all feature maps the same as the input of each level. We initialize the convolutional filters using the method of He et al. [40] and use the leaky rectified linear units (LReLUs) [80] with a negative slope of 0.2 as the non-linear activation function.

We use 91 images from Yang et al. [123] and 200 images from the training set of the Berkeley Segmentation Dataset [3] as our training data. The training dataset of 291 images is commonly used in the state-of-the-art SR methods [57, 95, 61, 103]. We use a batch size of 64 and crop the size of HR patches to $128 \times 128$. An epoch has $1,000$ iterations of back-propagation. We augment the training data in three ways: (1) *Scaling*: randomly downscale images between $[0.5, 1.0]$; (2) *Rotation*: randomly rotate image by $90°$, $180°$, or $270°$; (3) *Flipping*: flip images horizontally with a probability of $0.5$. Following the training protocol of existing methods [24, 57, 103], we generate the LR training patches using the bicubic downsampling. We use the MatConvNet toolbox [109] and train our model using the Stochastic Gradient Descent (SGD) solver. In addition, we set the momentum to $0.9$ and the weight decay to $1e-4$. The learning rate is initialized to $1e-5$ for all layers and decreased by a factor of 2 for every 100 epochs.

## 3.3 Discussions and Analysis

In this section, we first validate the contributions of different components in the proposed network. We then discuss the effect of local residual learning and parameter sharing in our feature embedding sub-network. Finally, we analyze the performance of multi-scale training strategy.

### 3.3.1 Model Design

We train a LapSRN model with 5 convolutional layers (without parameters sharing and the recursive layers) at each pyramid level to analyze the performance of pyramid network structure, global residual learning, robust loss functions, and multi-scale supervision.

**Pyramid structure.** By removing the pyramid structure, our model falls back to a network similar to the FSRCNN but with the global residual learning. We train this network using 10 convolutional layers in order to have the same depth as our LapSRN. Figure 3.5 shows the convergence curves in terms of PSNR on the SET14 for $4\times$ SR. The quantitative results in Table 3.1 and Figure 3.6 show that the pyramid structure leads to considerable performance improvement (e.g., 0.7 dB on SET5 and 0.4 dB on SET14), which validates

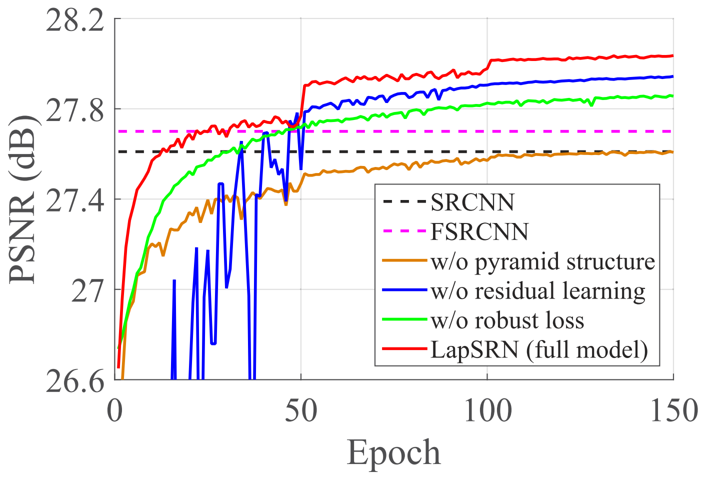


Figure 3.5: **Convergence analysis**. We analyze the contributions of the pyramid structures, loss functions, and global residual learning by replacing each component with the one used in existing methods. Our full model converges faster and achieves better performance.

Table 3.1: **Ablation study of LapSRN.** Our full model performs favorably against several variants of the LapSRN on both SET5 and SET14 for $4\times$ SR.

| GRL | Pyramid | Loss | SET5 | SET14 |
|---|---|---|---|---|
| ✓ | | Charbonnier | 30.58 | 27.61 |
| | ✓ | Charbonnier | 31.10 | 27.94 |
| ✓ | ✓ | $\mathcal{L}_2$ | 30.93 | 27.86 |
| ✓ | ✓ | Charbonnier | **31.28** | **28.04** |

the effectiveness of our Laplacian pyramid network design.

**Global residual learning.** To demonstrate the effectiveness of global residual learning, we remove the image reconstruction branch and directly predict the HR images at each level. In Figure 3.5, the performance of the non-residual network (blue curve) converges slowly and fluctuates significantly during training. Our full LapSRN model (red curve), on the other hand, outperforms SRCNN within 10 epochs.

**Loss function.** To validate the effectiveness of the Charbonnier loss function, we train the proposed network with conventional $\mathcal{L}_2$ loss function. We use a larger learning rate ($1e-4$) since the gradient magnitude of the $\mathcal{L}_2$ loss is smaller. As illustrated in Figure 3.5,

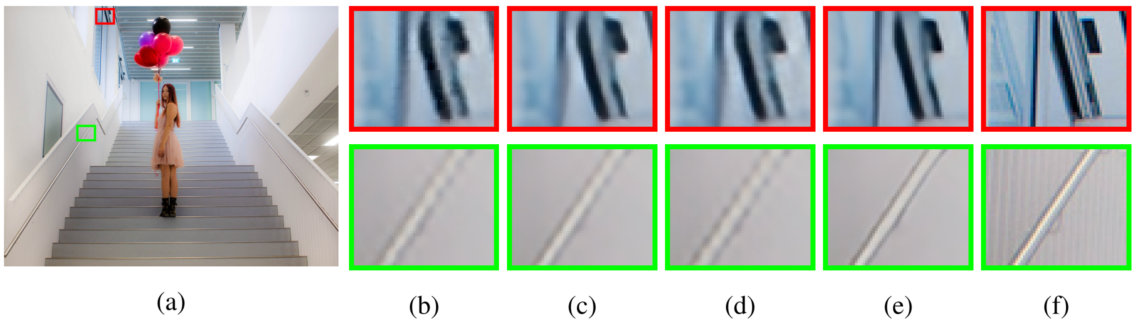


Figure 3.6: **Contribution of different components in LapSRN**. (a) Ground truth HR image (b) without pyramid structure (c) without global residual learning (d) without robust loss (e) full model (f) HR patch.

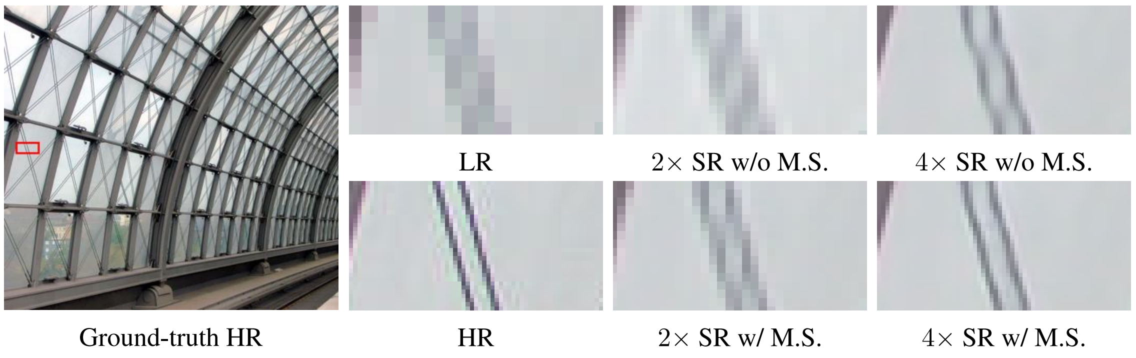


Figure 3.7: **Contribution of multi-scale supervision (M.S.)**. The multi-scale supervision guides the network training to progressively reconstruct the HR images and help reduce the spatial aliasing artifacts.

the network optimized with the $\mathcal{L}_2$ loss (green curve) requires more iterations to achieve comparable performance with SRCNN. In Figure 3.6, we show that the SR images reconstruct by our full model contain relatively clean and sharp details.

**Multi-scale supervision.** As described in Section 3.2.3, we use the multiple loss functions to supervise the intermediate output at each pyramid level. We show the intermediate output images in Figure 3.7. The model without the multi-scale supervision (i.e., only applying supervision at the finest scale) cannot reduce the spatial aliasing artifacts well, while our LapSRN progressively reconstructs clear and sharp straight lines.

### 3.3.2 Parameter Sharing

In this section, we reduce the network parameters in our LapSRN by sharing weights *across* and *within* pyramid levels and discuss the performance contribution.

**Parameter sharing across pyramid levels.** Our LapSRN $4\times$ model [61] has 812k parameters as each pyramid level has distinct convolutional and transposed convolutional layers. By sharing the weights across pyramid levels as shown in Figure 3.2, we reduce the number of parameters to 407k. Such model has 10 convolutional layers, 1 recursive block, and does not use any local residual learning strategies. We denote this model by LapSRN$_{\textbf{NS}}$-D10R1. We compare the above models on the BSDS100 and URBAN100 datasets for $4\times$ SR. Table 3.2 shows that the LapSRN$_{\textbf{NS}}$-D10R1 achieves comparable performance with the LapSRN [61] while using only half of the network parameters.

**Parameter sharing within pyramid levels.** We further reduce the network parameters by decreasing the number of convolutional layers (D) and increasing the number of recursive blocks (R). We train another two models: LapSRN$_{\textbf{NS}}$-D5R2 and LapSRN$_{\textbf{NS}}$-D2R5, which have 222k and 112k parameters, respectively. As shown in Table 3.2, while the LapSRN$_{\textbf{NS}}$-D5R2 and LapSRN$_{\textbf{NS}}$-D2R5 have fewer parameters, we observe the performance drop, particularly on the challenging URBAN100 dataset.

Table 3.2: **Parameter sharing in LapSRN.** We reduce the number of network parameters by sharing the weights between pyramid levels and applying recursive layers in the feature embedding sub-network.

| Model | #Parameters | BSDS100 | URBAN100 |
|---|---|---|---|
| LapSRN [61] | 812k | **27.32** | **25.21** |
| LapSRN$_{NS}$-D10R1 | 407k | **27.32** | 25.20 |
| LapSRN$_{NS}$-D5R2 | 222k | 27.30 | 25.16 |
| LapSRN$_{NS}$-D2R5 | 112k | 27.26 | 25.10 |

### 3.3.3 Training Deeper Models

In Section 3.3.2, we show that we can achieve comparable performance to the preliminary LapSRN by using only half or $27\%$ of parameters. Next, we train deeper models to improve the performance without increasing the number of the network parameters.

**Local residual learning.** We increase the number of recursive blocks in our feature embedding sub-network to increase the depth of network but keep the number of parameters the same. We test three LapSRN models: D5R2, D5R5, and D5R8, which have 5 distinct convolutional layers with 2, 5 and 8 recursive blocks, respectively. We train the models with three different local residual learning methods as described in Section 3.2.2. We plot the convergence curves of the LapSRN-D5R5 in Figure 3.8 and present the quantitative evaluation in Table 3.3. Overall, the shared-source local skip connection method (LapSRN$_{SS}$) performs favorably against other alternatives, particularly for deeper models (i.e., more recursive blocks).

**Study of D and R.** Our feature embedding sub-network consists of $R$ recursive blocks, and each recursive block has $D$ distinct convolutional layers which are shared among all the recursive blocks. Here we extensively evaluate the contributions of R and D to the reconstruction accuracy. We use D = $2, 4, 5, 10$ to construct models with different network depth. We use the shared-source local skip connection for all the evaluated models. We show the quantitative evaluation in Table 3.4 and visualize the performance over the network depth in Figure 3.9. While the D2R5, D5R2, and D10R1 models perform comparably, the D5R8

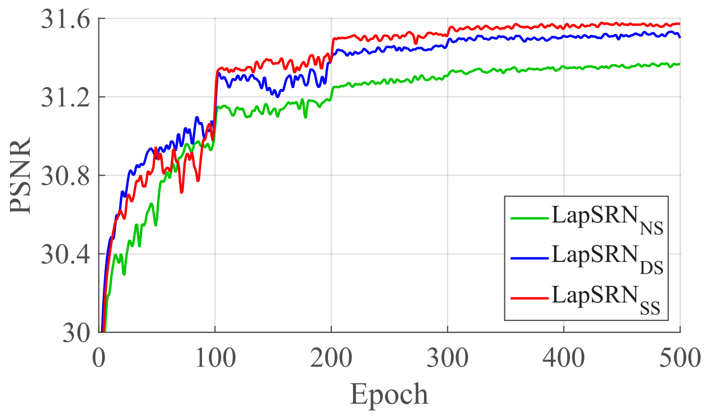


Figure 3.8: **Comparisons of local residual learning**. We train our LapSRN-D5R5 model with three different local residual learning methods as described in Section 3.2.2 and evaluate on the SET5 for $4\times$ SR.

Table 3.3: **Quantitative evaluation of local residual learning**. We compare three different local residual learning methods on the URBAN100 dataset for $4\times$ SR. Overall, the shared local skip connection method (LapSRN$_{SS}$) achieves superior performance for deeper models.

| Model | Depth | LapSRN$_{NS}$ | LapSRN$_{DS}$ | LapSRN$_{SS}$ |
|---|---|---|---|---|
| D5R2 | 24 | 25.16 | 25.22 | **25.23** |
| D5R5 | 54 | 25.18 | 25.33 | **25.34** |
| D5R8 | 84 | 25.26 | 25.33 | **25.38** |

Table 3.4: **Quantitative evaluation of the number of recursive blocks $\mathrm{R}$ and the number of convolutional layers $\mathrm{D}$ in our feature embedding sub-network**. We build LapSRN with different network depth by varying the values of D and R and evaluate on the BSDS100 and URBAN100 datasets for $4\times$ SR.

| Model | #Parameters | Depth | BSDS100 | URBAN100 |
|---|---|---|---|---|
| D2R5 | 112k | 24 | 27.33 | 25.24 |
| D2R12 | 112k | 52 | 27.35 | **25.31** |
| D2R20 | 112k | 84 | **27.37** | **25.31** |
| D4R3 | 185k | 28 | 27.33 | 25.25 |
| D4R6 | 185k | 52 | **27.37** | 25.34 |
| D4R10 | 185k | 84 | **27.37** | **25.35** |
| D5R2 | 222k | 24 | 27.32 | 25.23 |
| D5R5 | 222k | 54 | 27.38 | 25.34 |
| D5R8 | 222k | 84 | **27.39** | **25.38** |
| D10R1 | 407k | 24 | 27.33 | 25.23 |
| D10R2 | 407k | 44 | 27.36 | 25.27 |
| D10R4 | 407k | 84 | **27.38** | **25.36** |

method achieves the best reconstruction accuracy when the network depth is more than 80.

### 3.3.4 Multi-Scale Training

We train our LapSRN using the multi-scale training strategy. As our pyramid network design only accounts for training with $2^n\times$ samples, we train our LapSRN$_{\text{SS}}$-D5R8 model with the following scale combinations: $\{2\times\}$, $\{4\times\}$, $\{8\times\}$, $\{2\times, 4\times\}$, $\{2\times, 8\times\}$, $\{4\times, 8\times\}$ and $\{2\times, 4\times, 8\times\}$. During training, we equally split a batch of samples for every upsampling scale. Note that all these models have the same numbers of parameters due to parameter sharing. We evaluate the above models for $2\times$, $4\times$ and $8\times$ SR by constructing LapSRN with the corresponding pyramid levels. We also evaluate $3\times$ SR using our 2-level LapSRN and resizing the network output to the desired spatial resolution. We present the quantitative evaluation on the SET14, BSDS100 and URBAN100 datasets in Table 3.5 and and visual comparisons in Figure 3.10. From our experimental results, the model trained

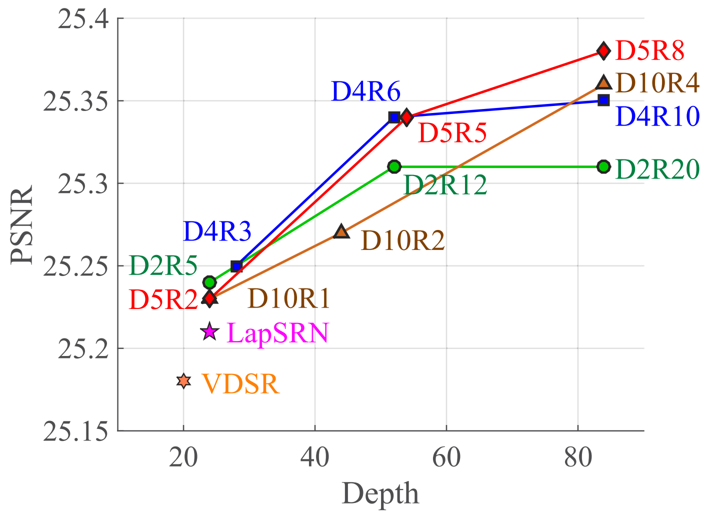
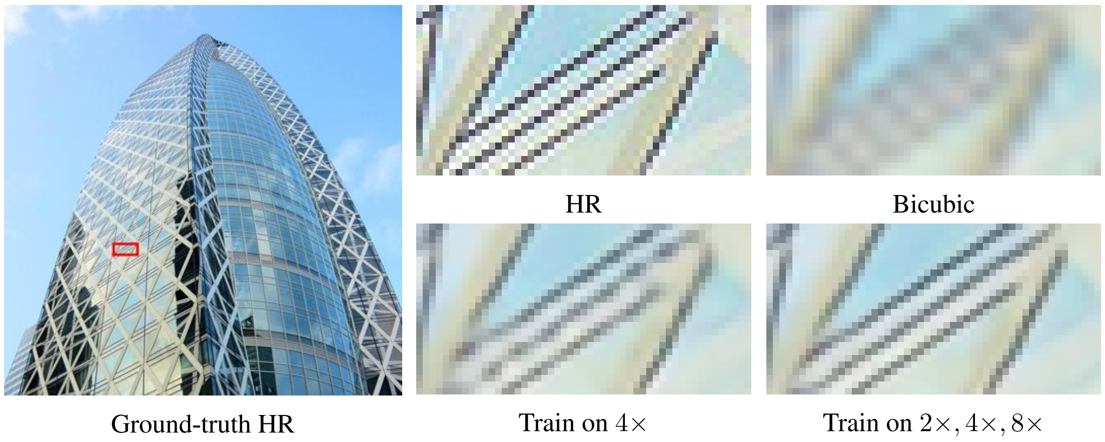


Figure 3.9: **PSNR versus network depth**. We test the proposed model with different $D$ and $R$ on the URBAN100 dataset for $4\times$ SR.

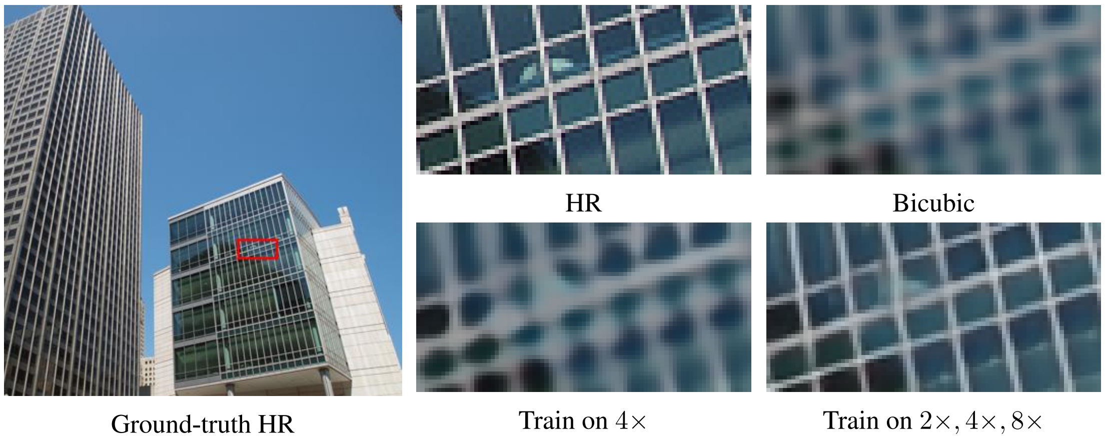
with $2\times, 4\times$ and $8\times$ samples has the capacity to handle multiple upsampling scales and generalizes well to the unseen $3\times$ SR examples. Furthermore, the multi-scale models perform favorably against the single-scale models, particularly on the URBAN100 dataset. Note that our models do not use any $3\times$ SR samples for training. Although our $4\times$ model is able to generate decent results for $3\times$ SR, the multi-scale models, especially the models trained on $\{2\times, 4\times\}$ and $\{2\times, 4\times, 8\times\}$ SR, further improve the performance by 0.14 and 0.17 dB on the URBAN100 dataset, respectively.

## 3.4 Experimental Results

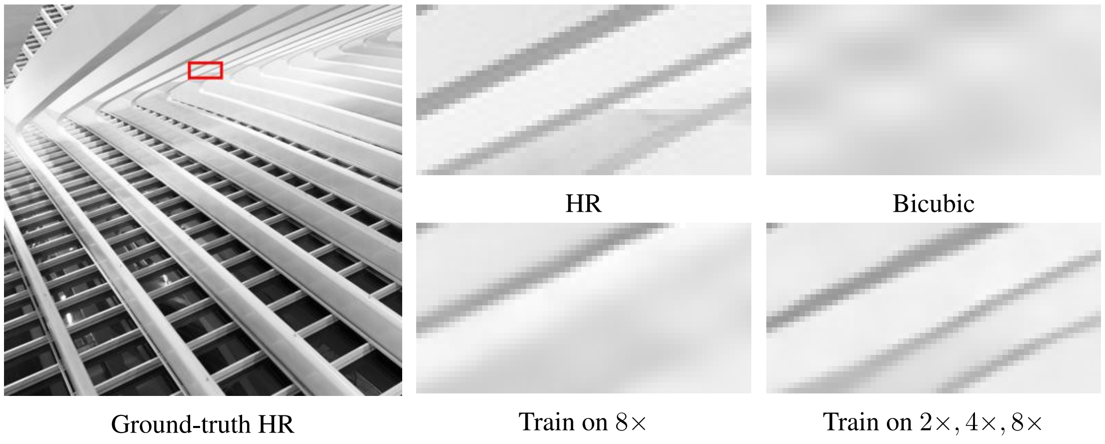
In this section, we compare the proposed LapSRN with several state-of-the-art SR methods on benchmark datasets. We present the quantitative evaluation, qualitative comparison, runtime, and parameters comparisons. We then evaluate our method on real-world photos, compare with the LAPGAN [23], and incorporate the adversarial training. In addition, we conduct a human subject study using the pairwise comparison to evaluate the subjective preference on SR results. Finally, we discuss the limitation of the proposed method. We provide our source code and SR results generated by all the evaluated methods on our project website at `http://vllab.ucmerced.edu/wlai24/LapSRN`.



(a) $3\times$ SR



(b) $4\times$ SR



(c) $8\times$ SR

Figure 3.10: **Visual comparison of multi-scale training.**

Table 3.5: **Quantitative evaluation of multi-scale training.** We train the proposed model with combinations of $2\times, 4\times$ and $8\times$ SR samples and evaluate on the Set14, BSDS100 and Urban100 datasets for $2\times, 3\times, 4\times$ and $8\times$ SR. The model trained with $2\times, 4\times$ and $8\times$ SR samples together achieves better performance on all upsampling scales and can also generalize to unseen $3\times$ SR examples.

| Train \ Test | Set14 | | | | BSDS100 | | | | Urban100 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $2\times$ | $3\times$ | $4\times$ | $8\times$ | $2\times$ | $3\times$ | $4\times$ | $8\times$ | $2\times$ | $3\times$ | $4\times$ | $8\times$ |
| $2\times$ | 33.24 | 27.31 | 25.21 | 21.81 | 32.01 | 26.93 | 25.13 | 22.25 | 31.01 | 24.93 | 22.80 | 19.89 |
| $4\times$ | 32.96 | 29.90 | 28.21 | 23.97 | 31.80 | 28.89 | 27.39 | 24.26 | 30.53 | 27.30 | 25.38 | 21.61 |
| $8\times$ | 32.36 | 29.89 | 28.14 | 24.52 | 31.28 | 28.84 | 27.34 | 24.63 | 29.16 | 27.15 | 25.23 | 21.95 |
| $2\times, 4\times$ | 33.25 | 29.96 | **28.27** | 24.15 | 32.04 | **28.93** | 27.42 | 24.32 | **31.17** | 27.44 | 25.49 | 21.81 |
| $2\times, 8\times$ | 33.22 | 29.93 | 28.17 | 24.52 | 32.01 | 28.89 | 27.38 | 24.63 | 31.05 | 27.32 | 25.38 | 22.00 |
| $4\times, 8\times$ | 32.88 | 29.90 | 28.20 | 24.48 | 31.76 | 28.89 | 27.40 | 24.62 | 30.42 | 27.37 | 25.43 | 22.01 |
| $2\times, 4\times, 8\times$ | **33.28** | **29.97** | 28.26 | **24.57** | **32.05** | **28.93** | **27.43** | **24.65** | 31.15 | **27.47** | **25.51** | **22.06** |

### 3.4.1 Comparisons with State-of-the-arts

We compare the proposed method with 10 state-of-the-art SR algorithms, including dictionary-based methods (A+ [105] and RFL [95]), self-similarity based method (SelfExSR [47]), and CNN-based methods (SRCNN [24], FSRCNN [25], SCN [113], VDSR [57], DRCN [58], DRRN [103] and our preliminary approach [61]). We carry out extensive experiments on five public benchmark datasets: Set5 [8], Set14 [128], BSDS100 [3], Urban100 [47] and Manga109 [82]. The Set5, Set14 and BSDS100 datasets consist of natural scenes; the Urban100 set contains challenging urban scenes images with details in different frequency bands; and the Manga109 is a dataset of Japanese manga.

We evaluate the SR results with three widely used image quality metrics: PSNR, SSIM [112], and IFC [97] and compare performance on $2\times$, $3\times$, $4\times$ and $8\times$ SR. We retrain existing methods for $8\times$ SR using the source code (A+ [105], RFL [95], SRCNN [24], FSRCNN [25], VDSR [57], and DRRN [103]) or our own implementation (DRCN). Both the SelfExSR and SCN methods can naturally handle different scale factors using progressive reconstruction. We use $2\times, 4\times$ and $8\times$ SR samples for training VDSR [57] and DRRN [103] while use only $8\times$ SR samples for other algorithms to follow the training strategies of individual methods.

Table 3.6: **Quantitative evaluation of state-of-the-art SR algorithms**. We report the average PSNR/SSIM/IFC for $2\times$, $3\times$, and $4\times$ SR. **Red** and blue indicate the best and the second best performance, respectively. Both LapSRN [61] and the proposed MS-LapSRN do not use any $3\times$ SR images for training. To generate the results of $3\times$ SR, we first perform $4\times$ SR on input LR images and then downsample the output to the target resolution.

| Algorithm | Scale | SET5 | SET14 | BSDS100 | URBAN100 | MANGA109 |
|---|---|---|---|---|---|---|
| | | PSNR / SSIM / IFC | PSNR / SSIM / IFC | PSNR / SSIM / IFC | PSNR / SSIM / IFC | PSNR / SSIM / IFC |
| Bicubic | | 33.69 / 0.931 / 6.166 | 30.25 / 0.870 / 6.126 | 29.57 / 0.844 / 5.695 | 26.89 / 0.841 / 6.319 | 30.86 / 0.936 / 6.214 |
| A+ [105] | | 36.60 / 0.955 / 8.715 | 32.32 / 0.906 / 8.200 | 31.24 / 0.887 / 7.464 | 29.25 / 0.895 / 8.440 | 35.37 / 0.968 / 8.906 |
| RFL [95] | | 36.59 / 0.954 / 8.741 | 32.29 / 0.905 / 8.224 | 31.18 / 0.885 / 7.473 | 29.14 / 0.891 / 8.439 | 35.12 / 0.966 / 8.921 |
| SelfExSR [47] | | 36.60 / 0.955 / 8.404 | 32.24 / 0.904 / 8.018 | 31.20 / 0.887 / 7.239 | 29.55 / 0.898 / 8.414 | 35.82 / 0.969 / 8.721 |
| SRCNN [24] | | 36.72 / 0.955 / 8.166 | 32.51 / 0.908 / 7.867 | 31.38 / 0.889 / 7.242 | 29.53 / 0.896 / 8.092 | 35.76 / 0.968 / 8.471 |
| FSRCNN [25] | $2\times$ | 37.05 / 0.956 / 8.199 | 32.66 / 0.909 / 7.841 | 31.53 / 0.892 / 7.180 | 29.88 / 0.902 / 8.131 | 36.67 / 0.971 / 8.587 |
| SCN [113] | | 36.58 / 0.954 / 7.358 | 32.35 / 0.905 / 7.085 | 31.26 / 0.885 / 6.500 | 29.52 / 0.897 / 7.324 | 35.51 / 0.967 / 7.601 |
| VDSR [57] | | 37.53 / 0.959 / 8.190 | 33.05 / 0.913 / 7.878 | 31.90 / 0.896 / 7.169 | 30.77 / 0.914 / 8.270 | 37.22 / 0.975 / 9.120 |
| DRCN [58] | | 37.63 / 0.959 / 8.326 | 33.06 / 0.912 / 8.025 | 31.85 / 0.895 / 7.220 | 30.76 / 0.914 / 8.527 | 37.63 / 0.974 / 9.541 |
| LapSRN [61] | | 37.52 / 0.959 / 9.010 | 33.08 / 0.913 / 8.501 | 31.80 / 0.895 / 7.715 | 30.41 / 0.910 / 8.907 | 37.27 / 0.974 / 9.481 |
| DRRN [103] | | 37.74 / 0.959 / 8.671 | 33.23 / 0.914 / 8.320 | **32.05** / 0.897 / 7.613 | **31.23** / **0.919** / 8.917 | **37.92** / **0.976** / 9.268 |
| MS-LapSRN-D5R2 (ours) | | 37.62 / **0.960** / 9.038 | 33.13 / 0.913 / 8.539 | 31.93 / 0.897 / 7.776 | 30.82 / 0.915 / 9.081 | 37.38 / 0.975 / 9.434 |
| MS-LapSRN-D5R5 (ours) | | 37.72 / **0.960** / 9.265 | 33.24 / 0.914 / 8.726 | 32.00 / **0.898** / 7.906 | 31.01 / 0.917 / 9.334 | 37.71 / 0.975 / 9.710 |
| MS-LapSRN-D5R8 (ours) | | **37.78** / **0.960** / **9.305** | **33.28** / **0.915** / **8.748** | **32.05** / **0.898** / **7.927** | 31.15 / **0.919** / **9.406** | 37.78 / **0.976** / **9.765** |
| Bicubic | | 30.41 / 0.869 / 3.596 | 27.55 / 0.775 / 3.491 | 27.22 / 0.741 / 3.168 | 24.47 / 0.737 / 3.661 | 26.99 / 0.859 / 3.521 |
| A+ [105] | | 32.62 / 0.909 / 4.979 | 29.15 / 0.820 / 4.545 | 28.31 / 0.785 / 4.028 | 26.05 / 0.799 / 4.883 | 29.93 / 0.912 / 4.880 |
| RFL [95] | | 32.47 / 0.906 / 4.956 | 29.07 / 0.818 / 4.533 | 28.23 / 0.782 / 4.023 | 25.88 / 0.792 / 4.781 | 29.61 / 0.905 / 4.758 |
| SelfExSR [47] | | 32.66 / 0.910 / 4.911 | 29.18 / 0.821 / 4.505 | 28.30 / 0.786 / 3.923 | 26.45 / 0.810 / 4.988 | 27.57 / 0.821 / 2.193 |
| SRCNN [24] | | 32.78 / 0.909 / 4.682 | 29.32 / 0.823 / 4.372 | 28.42 / 0.788 / 3.879 | 26.25 / 0.801 / 4.630 | 30.59 / 0.914 / 4.698 |
| FSRCNN [25] | $3\times$ | 33.18 / 0.914 / 4.970 | 29.37 / 0.824 / 4.569 | 28.53 / 0.791 / 4.061 | 26.43 / 0.808 / 4.878 | 31.10 / 0.921 / 4.912 |
| SCN [113] | | 32.62 / 0.908 / 4.321 | 29.16 / 0.818 / 4.006 | 28.33 / 0.783 / 3.553 | 26.21 / 0.801 / 4.253 | 30.22 / 0.914 / 4.302 |
| VDSR [57] | | 33.67 / 0.921 / 5.088 | 29.78 / 0.832 / 4.606 | 28.83 / 0.799 / 4.043 | 27.14 / 0.829 / 5.045 | 32.01 / 0.934 / 5.389 |
| DRCN [58] | | 33.83 / 0.922 / 5.202 | 29.77 / 0.832 / 4.686 | 28.80 / 0.797 / 4.070 | 27.15 / 0.828 / 5.187 | 32.31 / 0.936 / 5.564 |
| LapSRN [61] | | 33.82 / 0.922 / 5.194 | 29.87 / 0.832 / 4.662 | 28.82 / 0.798 / 4.057 | 27.07 / 0.828 / 5.168 | 32.21 / 0.935 / 5.406 |
| DRRN [103] | | 34.03 / **0.924** / **5.397** | 29.96 / 0.835 / **4.878** | **28.95** / 0.800 / **4.269** | **27.53** / 0.764 / **5.456** | **32.74** / **0.939** / **5.659** |
| MS-LapSRN-D5R2 (ours) | | 33.88 / 0.923 / 5.165 | 29.89 / 0.834 / 4.637 | 28.87 / 0.800 / 4.040 | 27.23 / 0.831 / 5.142 | 32.28 / 0.936 / 5.384 |
| MS-LapSRN-D5R5 (ours) | | 34.01 / **0.924** / 5.307 | 29.96 / **0.836** / 4.758 | 28.92 / 0.801 / 4.127 | 27.39 / 0.835 / 5.333 | 32.60 / 0.938 / 5.559 |
| MS-LapSRN-D5R8 (ours) | | **34.06** / **0.924** / 5.390 | **29.97** / **0.836** / 4.806 | 28.93 / **0.802** / 4.154 | 27.47 / **0.837** / 5.409 | 32.68 / **0.939** / 5.621 |
| Bicubic | | 28.43 / 0.811 / 2.337 | 26.01 / 0.704 / 2.246 | 25.97 / 0.670 / 1.993 | 23.15 / 0.660 / 2.386 | 24.93 / 0.790 / 2.289 |
| A+ [105] | | 30.32 / 0.860 / 3.260 | 27.34 / 0.751 / 2.961 | 26.83 / 0.711 / 2.565 | 24.34 / 0.721 / 3.218 | 27.03 / 0.851 / 3.177 |
| RFL [95] | | 30.17 / 0.855 / 3.205 | 27.24 / 0.747 / 2.924 | 26.76 / 0.708 / 2.538 | 24.20 / 0.712 / 3.101 | 26.80 / 0.841 / 3.055 |
| SelfExSR [47] | | 30.34 / 0.862 / 3.249 | 27.41 / 0.753 / 2.952 | 26.84 / 0.713 / 2.512 | 24.83 / 0.740 / 3.381 | 27.83 / 0.866 / 3.358 |
| SRCNN [24] | | 30.50 / 0.863 / 2.997 | 27.52 / 0.753 / 2.766 | 26.91 / 0.712 / 2.412 | 24.53 / 0.725 / 2.992 | 27.66 / 0.859 / 3.045 |
| FSRCNN [25] | $4\times$ | 30.72 / 0.866 / 2.994 | 27.61 / 0.755 / 2.722 | 26.98 / 0.715 / 2.370 | 24.62 / 0.728 / 2.916 | 27.90 / 0.861 / 2.950 |
| SCN [113] | | 30.41 / 0.863 / 2.911 | 27.39 / 0.751 / 2.651 | 26.88 / 0.711 / 2.309 | 24.52 / 0.726 / 2.860 | 27.39 / 0.857 / 2.889 |
| VDSR [57] | | 31.35 / 0.883 / 3.496 | 28.02 / 0.768 / 3.071 | 27.29 / 0.726 / 2.627 | 25.18 / 0.754 / 3.405 | 28.83 / 0.887 / 3.664 |
| DRCN [58] | | 31.54 / 0.884 / 3.502 | 28.03 / 0.768 / 3.066 | 27.24 / 0.725 / 2.587 | 25.14 / 0.752 / 3.412 | 28.98 / 0.887 / 3.674 |
| LapSRN [61] | | 31.54 / 0.885 / 3.559 | 28.19 / 0.772 / 3.147 | 27.32 / 0.727 / 2.677 | 25.21 / 0.756 / 3.530 | 29.09 / 0.890 / 3.729 |
| DRRN [103] | | 31.68 / 0.888 / 3.703 | 28.21 / 0.772 / 3.252 | 27.38 / 0.728 / **2.760** | 25.44 / 0.764 / 3.700 | 29.46 / 0.896 / 3.878 |
| MS-LapSRN-D5R2 (ours) | | 31.62 / 0.887 / 3.585 | 28.16 / 0.772 / 3.151 | 27.36 / 0.729 / 2.684 | 25.32 / 0.760 / 3.537 | 29.18 / 0.892 / 3.750 |
| MS-LapSRN-D5R5 (ours) | | **31.74** / 0.888 / 3.705 | 28.25 / 0.773 / 3.238 | 27.42 / **0.731** / 2.737 | 25.45 / 0.765 / 3.674 | 29.48 / 0.896 / 3.888 |
| MS-LapSRN-D5R8 (ours) | | **31.74** / **0.889** / **3.749** | **28.26** / **0.774** / **3.261** | **27.43** / **0.731** / 2.755 | **25.51** / **0.768** / **3.727** | **29.54** / **0.897** / **3.928** |

Table 3.7: **Quantitative evaluation of state-of-the-art SR algorithms**. We report the average PSNR/SSIM/IFC for $8\times$ SR. **Red** and blue indicate the best and the second best performance, respectively.

| Algorithm | Scale | SET5 | SET14 | BSDS100 | URBAN100 | MANGA109 |
|---|---|---|---|---|---|---|
| | | PSNR / SSIM / IFC | PSNR / SSIM / IFC | PSNR / SSIM / IFC | PSNR / SSIM / IFC | PSNR / SSIM / IFC |
| Bicubic | $8\times$ | 24.40 / 0.658 / 0.836 | 23.10 / 0.566 / 0.784 | 23.67 / 0.548 / 0.646 | 20.74 / 0.516 / 0.858 | 21.47 / 0.650 / 0.810 |
| A+ [105] | | 25.53 / 0.693 / 1.077 | 23.89 / 0.595 / 0.983 | 24.21 / 0.569 / 0.797 | 21.37 / 0.546 / 1.092 | 22.39 / 0.681 / 1.056 |
| RFL [95] | | 25.38 / 0.679 / 0.991 | 23.79 / 0.587 / 0.916 | 24.13 / 0.563 / 0.749 | 21.27 / 0.536 / 0.992 | 22.28 / 0.669 / 0.968 |
| SelfExSR [47] | | 25.49 / 0.703 / 1.121 | 23.92 / 0.601 / 1.005 | 24.19 / 0.568 / 0.773 | 21.81 / 0.577 / 1.283 | 22.99 / 0.719 / 1.244 |
| SRCNN [24] | | 25.33 / 0.690 / 0.938 | 23.76 / 0.591 / 0.865 | 24.13 / 0.566 / 0.705 | 21.29 / 0.544 / 0.947 | 22.46 / 0.695 / 1.013 |
| FSRCNN [25] | | 25.60 / 0.697 / 1.016 | 24.00 / 0.599 / 0.942 | 24.31 / 0.572 / 0.767 | 21.45 / 0.550 / 0.995 | 22.72 / 0.692 / 1.009 |
| SCN [113] | | 25.59 / 0.706 / 1.063 | 24.02 / 0.603 / 0.967 | 24.30 / 0.573 / 0.777 | 21.52 / 0.560 / 1.074 | 22.68 / 0.701 / 1.073 |
| VDSR [57] | | 25.93 / 0.724 / 1.199 | 24.26 / 0.614 / 1.067 | 24.49 / 0.583 / 0.859 | 21.70 / 0.571 / 1.199 | 23.16 / 0.725 / 1.263 |
| DRCN [58] | | 25.93 / 0.723 / 1.192 | 24.25 / 0.614 / 1.057 | 24.49 / 0.582 / 0.854 | 21.71 / 0.571 / 1.197 | 23.20 / 0.724 / 1.257 |
| LapSRN [61] | | 26.15 / 0.738 / 1.302 | 24.35 / 0.620 / 1.133 | 24.54 / 0.586 / 0.893 | 21.81 / 0.581 / 1.288 | 23.39 / 0.735 / 1.352 |
| DRRN [103] | | 26.18 / 0.738 / 1.307 | 24.42 / 0.622 / 1.127 | 24.59 / 0.587 / 0.891 | 21.88 / 0.583 / 1.299 | 23.60 / 0.742 / 1.406 |
| MS-LapSRN-D5R2 (ours) | | 26.20 / 0.747 / 1.366 | 24.45 / 0.626 / 1.170 | 24.61 / 0.590 / 0.920 | 21.95 / 0.592 / 1.364 | 23.70 / 0.751 / 1.470 |
| MS-LapSRN-D5R5 (ours) | | **26.34** / 0.752 / 1.414 | **24.57** / **0.629** / 1.200 | **24.65** / 0.591 / 0.938 | **22.06** / 0.597 / 1.426 | 23.85 / 0.756 / 1.538 |
| MS-LapSRN-D5R8 (ours) | | **26.34** / **0.753** / **1.435** | **24.57** / **0.629** / **1.209** | **24.65** / **0.592** / **0.943** | **22.06** / **0.598** / **1.446** | **23.90** / **0.759** / **1.564** |

We compare three variations of the proposed method: (1) LapSRN$_{SS}$-D5R2, which has similar depth as the VDSR [57], DRCN [58] and LapSRN [61], (2) LapSRN$_{SS}$-D5R5, which has the same depth as in the DRRN [103], and (3) LapSRN$_{SS}$-D5R8, which has 84 layers for $4\times$ SR. We train the above three models using the multi-scale training strategy with $2\times, 4\times$ and $8\times$ SR samples and denote our multi-scale models as MS-LapSRN.

We show the quantitative results in Table 3.6 and 3.7. Our LapSRN performs favorably against existing methods especially on $4\times$ and $8\times$ SR. In particular, our algorithm achieves higher IFC values, which has been shown to be correlated well with human perception of image super-resolution [120]. We note that our method does not use any $3\times$ SR samples for training but still generates comparable results as the DRRN.

We show visual comparisons on the BSDS100, URBAN100 and MANGA109 datasets for $4\times$ and $8\times$ SR in Figure 3.11. Our method accurately reconstructs parallel straight lines, grid patterns, and texts. We observe that the results generated from pre-upsampling based methods [24, 57, 103] still contain noticeable artifacts caused by spatial aliasing. In contrast, our approach effectively suppresses such artifacts through progressive reconstruction and the robust loss function. For $8\times$ SR, it is challenging to predict HR images from bicubic-upsampled input [24, 57, 105] or using one-step upsampling [25]. The state-of-the-art methods do not super-resolve the fine structures well. In contrast, our MS-LapSRN reconstructs high-quality HR images at a relatively fast speed.

### 3.4.2 Execution Time

We use the source codes of state-of-the-art methods to evaluate the runtime on the same machine with 3.4 GHz Intel i7 CPU (32G RAM) and NVIDIA Titan Xp GPU (12G Memory). Since the testing code of the SRCNN [24] and FSRCNN [25] is based on CPU implementation, we rebuild these models in MatConvNet to measure the runtime on GPU. Figure 3.12 shows the trade-offs between the runtime and performance (in terms of PSNR) on the URBAN100 dataset for $4\times$ SR. The speed of our MS-LapSRN-D5R2 is faster than all the existing methods except the FSRCNN [25]. Our MS-LapSRN-D5R8 model outperforms the state-of-the-art DRRN [103] method and is an order of magnitude faster.

Next, we focus on comparisons between fast CNN-based methods: SRCNN [24], FSRCNN [25], VDSR [57], and LapSRN [61]. We take an LR image with a spatial resolution of $128 \times 128$, and perform $2\times$, $4\times$ and $8\times$ SR, respectively. We evaluate each method for 10 times and report the averaged runtime in Figure 3.13. The FSRCNN is the fastest algorithm since it applies convolution on LR images and has less number of convolutional layers and filters. The runtime of the SRCNN and VDSR depends on the size of *output* images, while the speed of the FSRCNN and LapSRN is mainly determined by the size of *input* images. As the proposed LapSRN progressively upscales images and applies more convolutional layers for larger upsampling scales (i.e., require more pyramid levels), the time complexity slightly increases with respect to the desired upsampling scales. However, the speed of our LapSRN still performs favorably against the SRCNN, VDSR, and other existing methods.

### 3.4.3 Model Parameters

We show the reconstruction performance versus the number of network parameters of CNN-based SR methods in Figure 3.14. By sharing parameters and using recursive layers, our MS-LapSRN has parameters about $73\%$ less than the LapSRN [61], $66\%$ less than the VDSR [57], $87\%$ less than the DRCN [58], and $25\%$ less than the DRRN [103]. While our model has a smaller footprint, we achieve the state-of-the-art performance among these CNN-based methods. Comparing to the SRCNN [24] and FSRCNN [25], our MS-

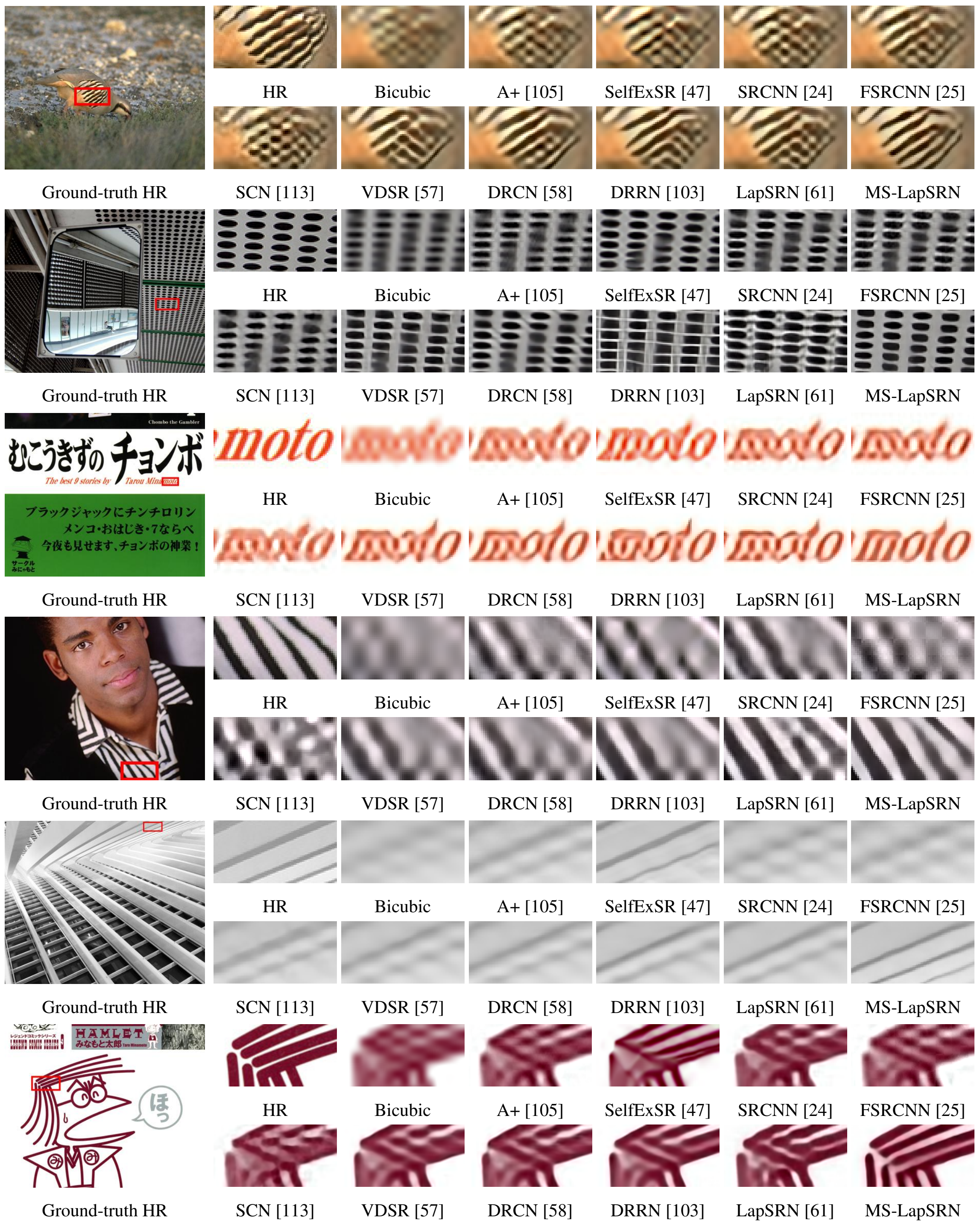


Figure 3.11: **Visual comparison for $4\times$ SR (top-3) and $8\times$ SR (bottom-3) on the BSDS100, URBAN100 and MANGA109 datasets.**

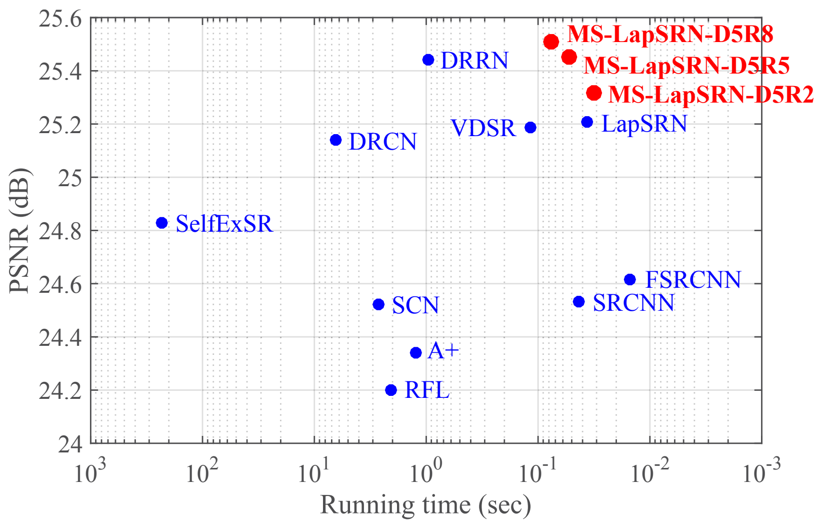


Figure 3.12: **Runtime versus performance**. The results are evaluated on the URBAN100 dataset for $4\times$ SR. The proposed MS-LapSRN strides a balance between reconstruction accuracy and execution time.

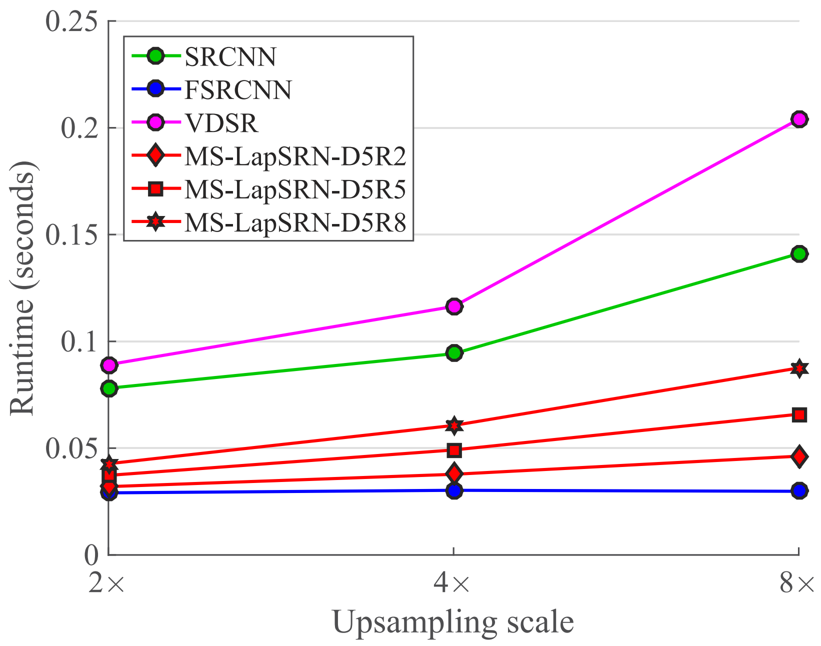


Figure 3.13: **Trade-off between runtime and upsampling scales**. We fix the size of input images to $128 \times 128$ and perform $2\times$, $4\times$ and $8\times$ SR with the SRCNN [24], FSRCNN [25], VDSR [57] and three variations of MS-LapSRN, respectively.

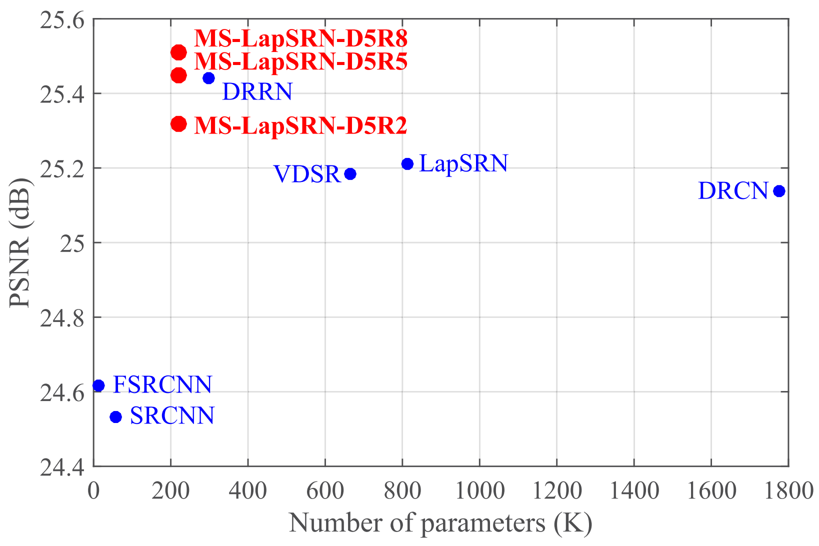


Figure 3.14: **Number of network parameters versus performance**. The results are evaluated on the URBAN100 dataset for $4\times$ SR. The proposed MS-LapSRN strides a balance between reconstruction accuracy and execution time.

LapSRN-D5R8 has about 0.9 to 1 dB improvement on the challenging URBAN100 dataset for $4\times$ SR.

### 3.4.4 Super-Resolving Real-World Photos

We demonstrate an application of super-resolving historical photographs with JPEG compression artifacts. In these cases, neither the ground-truth images nor the downsampling kernels are available. As shown in Figure 3.15, our method can reconstruct sharper and more accurate images than the state-of-the-art approaches.

### 3.4.5 Comparison to LAPGAN

As described in Section 2.1.4, the target applications of the LAPGAN [23] and LapSRN are different. Therefore, we focus on comparing the *network architectures* for image super-resolution. We train the LAPGAN and LapSRN for $4\times$ and $8\times$ SR with the same training data and settings. We use 5 convolutional layers at each level and optimize both networks with the Charbonnier loss function. We note that in [23] the sub-networks are independently trained. For fair comparisons, we jointly train the entire network for both

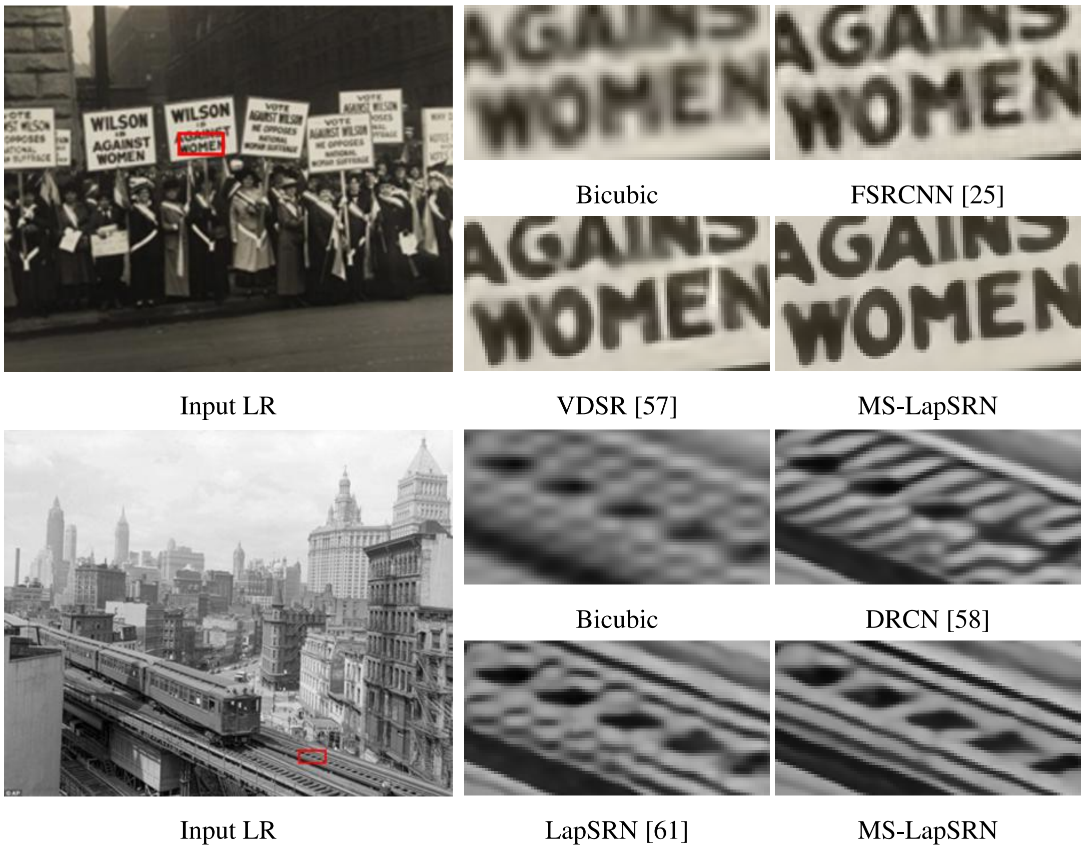


Figure 3.15: **Comparison of real-world photos for** $4\times$ **SR.** The ground truth HR images and the blur kernels are not available in these cases. On the top image, our method super-resolves the letter “W” accurately while VDSR incorrectly connects the stroke with the letter “O”. On the bottom image, our method reconstructs the rails without the artifacts.

Table 3.8: **Quantitative comparisons between the generative network of the LAPGAN [23] and our LapSRN.** Our LapSRN achieves better reconstruction quality and faster processing speed than the LAPGAN.

| Method | Scale | SET14 | | BSDS100 | |
|---|---|---|---|---|---|
| | | PSNR | Seconds | PSNR | Seconds |
| LAPGAN | 4 | 27.89 | 0.0446 | 27.09 | 0.0135 |
| LapSRN | 4 | 28.04 | 0.0395 | 27.22 | 0.0078 |
| LAPGAN | 8 | 24.30 | 0.0518 | 24.46 | 0.0110 |
| LapSRN | 8 | 24.42 | 0.0427 | 24.53 | 0.0107 |

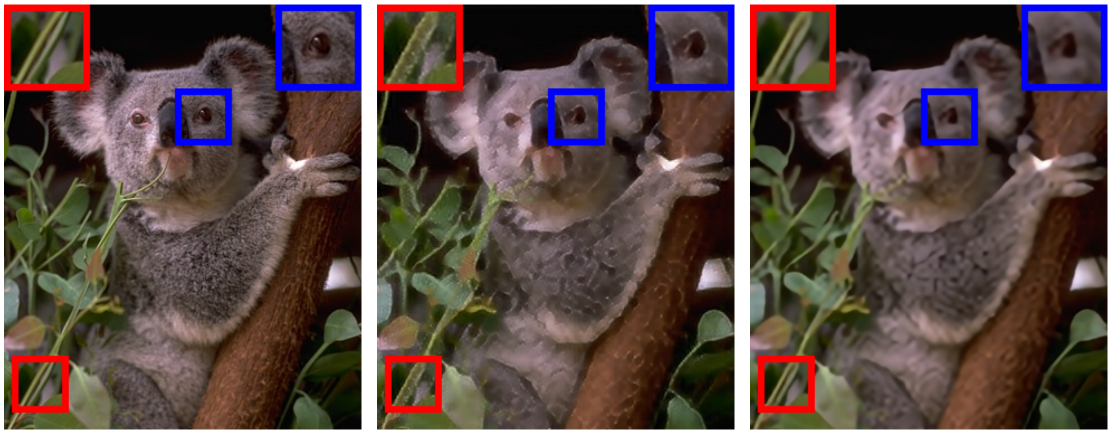

(a) Ground Truth HR (b) LapSRN + adv. (c) LapSRN

Figure 3.16: **Visual comparison for adversarial training**. We compare the results trained with and without the adversarial training on $4\times$ SR.

LAPGAN and LapSRN. We present quantitative comparisons and runtime on the SET14 and BSDS100 datasets in Table 3.8. Under the same training setting, our method achieves more accurate reconstruction and faster execution speed than that of the LAPGAN.

### 3.4.6 Adversarial Training

We demonstrate that our LapSRN can be extended to incorporate the adversarial training [36]. We treat our LapSRN as a generative network and build a discriminative network using the discriminator of the DCGAN [90]. The discriminative network consists of four convolutional layers with a stride of 2, two fully connected layers and one sigmoid layer to generate a scalar probability for distinguishing between real images and generated images from the generative network. We find that it is difficult to obtain accurate SR images by solely minimizing the cross-entropy loss. Therefore, we include the pixel-wise reconstruction loss (i.e., Charbonnier loss) to enforce the similarity between the input LR images and the corresponding ground truth HR images.

We show a visual result in Figure 3.16 for $4\times$ SR. The network with the adversarial training generates more plausible details on regions of irregular structures, e.g., grass, and feathers. However, the predicted results may not be faithfully reconstructed with respect to the ground truth high-resolution images. As a result, the accuracy is not as good as the model trained with the Charbonnier loss.

### 3.4.7 Human Subject Study

To measure the human perceptual preferences on super-resolved images, we conduct a human subject study to evaluate several state-of-the-art SR algorithms. A straightforward strategy is to ask users to give an *absolute* score (e.g., 1 to 5) or provide ranking on all SR results for each test image. One can then compute the average score for each method. However, such scores might not be sufficiently reliable when there are a large number of images to be compared. Furthermore, as super-resolved images from different algorithms often have subtle differences, it is difficult for users to make comparisons on multiple images simultaneously.

In light of this, we choose to conduct the *paired* comparison for our subject study. Paired comparison is also adopted to evaluate the perceptual quality of image retargeting [93] and image deblurring [62]. For each test, each user is asked to select the preferred one from a pair of images. We design a web-based interface (Figure 3.17) for users to switch back and forth between two given images. Users can easily see the differences between the two images and make selections. Through the study, we obtain the *relative* scores between every pair of evaluated methods. We conduct such paired comparison for FSRCNN [25], VDSR [57], LapSRN [61], DRRN [103] and the proposed MS-LapSRN on the BSDS100 [3] and Urban100 [47] datasets for $4\times$ SR.

We ask each participant to compare 30 pairs of images. To detect casual or careless users, we include 5 pairs of images as the sanity check. In these image pairs, we show the

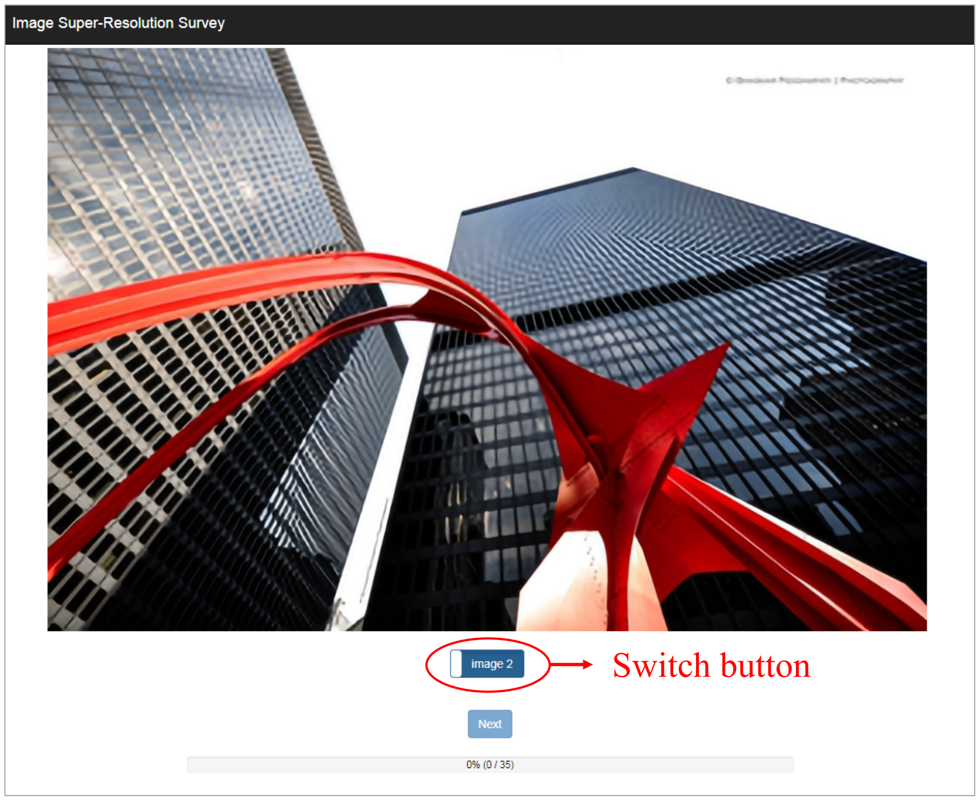
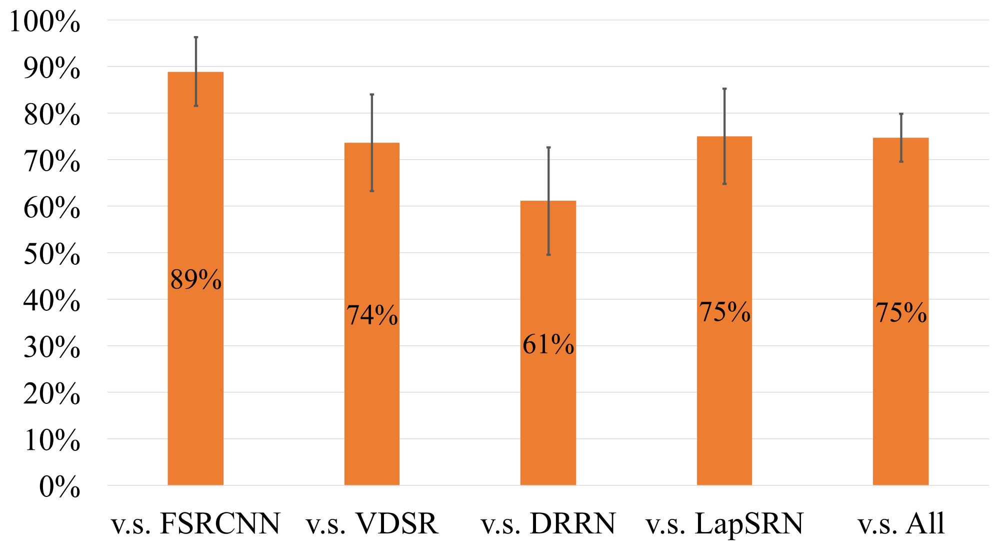


Figure 3.17: **Interface for our human subject study.** Human subjects can switch back and forth between two given images (results from two different super-resolution algorithms) to see the differences.

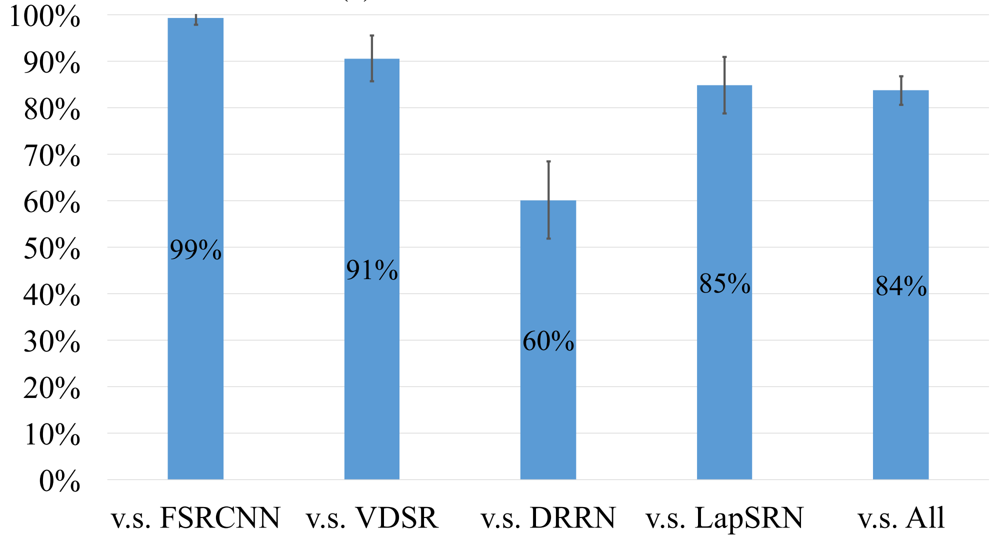
ground truth HR and the bicubic upsampled images. The users must select the ground truth HR image to pass the sanity check. We discard the results by a subject if the subject fails the sanity check more than twice. Finally, we collect the results from 71 participants.

To obtain a global score for each method, we fit the results of paired comparisons to the Bradley-Terry (BT) model [12]. We refer readers to [93, 62] for details of the BT model. We normalize the BT scores to zero means and show the results in Table 3.9. The proposed MS-LapSRN performs favorably against other approaches on both the BSDS100 and Urban100 datasets.

We further analyze the comparisons between our MS-LapSRN and other methods. We compute the percentage that users choose our MS-LapSRN over FSRCNN, VDSR, DRRN, and LapSRN and plot the results in Figure 3.18 (a) and (b). On average, our MS-LapSRN is preferred by 75% of users on the BSDS100 dataset and 84% of users on the URBAN100



(a) Results on BSDS100



(b) Results on URBAN100

Figure 3.18: **Analysis on human subject study.** Our MS-LapSRN is preferred by $75\%$ and $80\%$ of users on the BSDS100 and URBAN100 datasets, respectively. The error bars show the $95\%$ confidence interval.

Table 3.9: **BT scores of SR algorithms in human subject study.** Our MS-LapSRN performs favorably against other compared methods.

| Method | BSDS100 | URBAN100 |
|---|---|---|
| FSRCNN [25] | -1.1291 | -1.8005 |
| VDSR [57] | 0.0357 | 0.0981 |
| LapSRN [61] | 0.1910 | 0.2415 |
| DRRN [103] | 0.3721 | 0.6521 |
| MS-LapSRN | 0.5304 | 0.8087 |

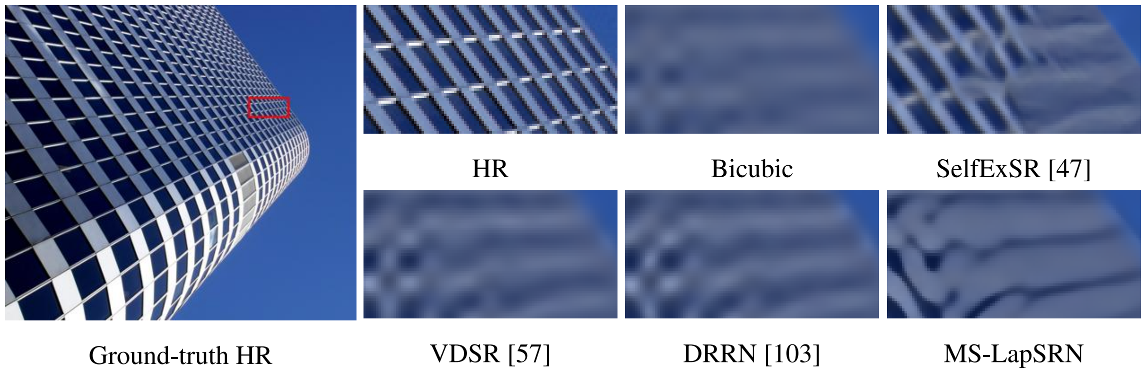


Figure 3.19: **Limitation.** A failure case for $8\times$ SR. Our method is not able to hallucinate details if the LR input image does not consist of sufficient amount of structure.

dataset. When comparing with DRRN, our MS-LapSRN obtains around $60\%$ of votes as the performance is close to each other. The human subject study also shows that the results of our MS-LapSRN have higher perceptual quality than existing methods.

### 3.4.8 Limitations

While our model is capable of generating clean and sharp HR images for large upsampling scales, e.g., $8\times$, it does not "hallucinate" fine details. As shown in Figure 3.19, the top of the building is significantly blurred in the $8\times$ downscaled LR image. All SR algorithms fail to recover the fine structure except the SelfExSR [47] method which explicitly detects the 3D scene geometry and uses self-similarity to hallucinate the regular structure. This is a common limitation shared by parametric SR methods [105, 24, 25, 57, 58, 103].

## 3.5 Conclusions

In this work, we propose a deep convolutional network within a Laplacian pyramid framework for fast and accurate image super-resolution. Our model progressively predicts high-frequency residuals in a coarse-to-fine manner with deeply supervision from the robust Charbonnier loss functions. By sharing parameters *across* as well as *within* pyramid levels, we use $73\%$ fewer parameters than our preliminary method [61] and achieve improved performance. We incorporate local skip connections in our network for training deeper models. Furthermore, we adopt the multi-scale training strategy to train a *single* model for handling *multiple* upsampling scales. We present a comprehensive evaluation on various design choices and believe that the thorough analysis benefits our community. We have shown the promise of the proposed LapSRN in the context of image super-resolution. Yet, our network design is general and can potentially be applied to other image transformation and synthesis problems.

# Chapter 4

# Learning Blind Video Temporal Consistency

## 4.1 Introduction

Recent advances of deep convolutional neural networks (CNNs) have led to the development of many powerful image processing techniques including, image filtering [73, 117], enhancement [33, 61, 118], style transfer [48, 56, 72], colorization [50, 130], and general image-to-image translation tasks [52, 68, 132]. However, extending these CNN-based methods to video is non-trivial due to memory and computational constraints, and the availability of training datasets. Applying image-based algorithms independently to each video frame typically leads to temporal flickering due to the instability of global optimization algorithms or highly non-linear deep networks. One approach for achieving temporally coherent results is to explicitly embed flow-based temporal consistency loss in the design and training of the networks. However, such an approach suffers from two drawbacks. First, it requires domain knowledge to re-design the algorithm [5, 46], re-train a deep model [37, 45], and video datasets for training. Second, due to the dependency of flow computation at test time, these approaches tend to be slow.

Bonneel et al. [11] propose a general approach to achieve temporal coherent results that is *blind* to specific image processing algorithms. The method takes the original video and the per-frame processed video as inputs and solves a gradient-domain optimization problem to minimize the temporal warping error between consecutive frames. Although the

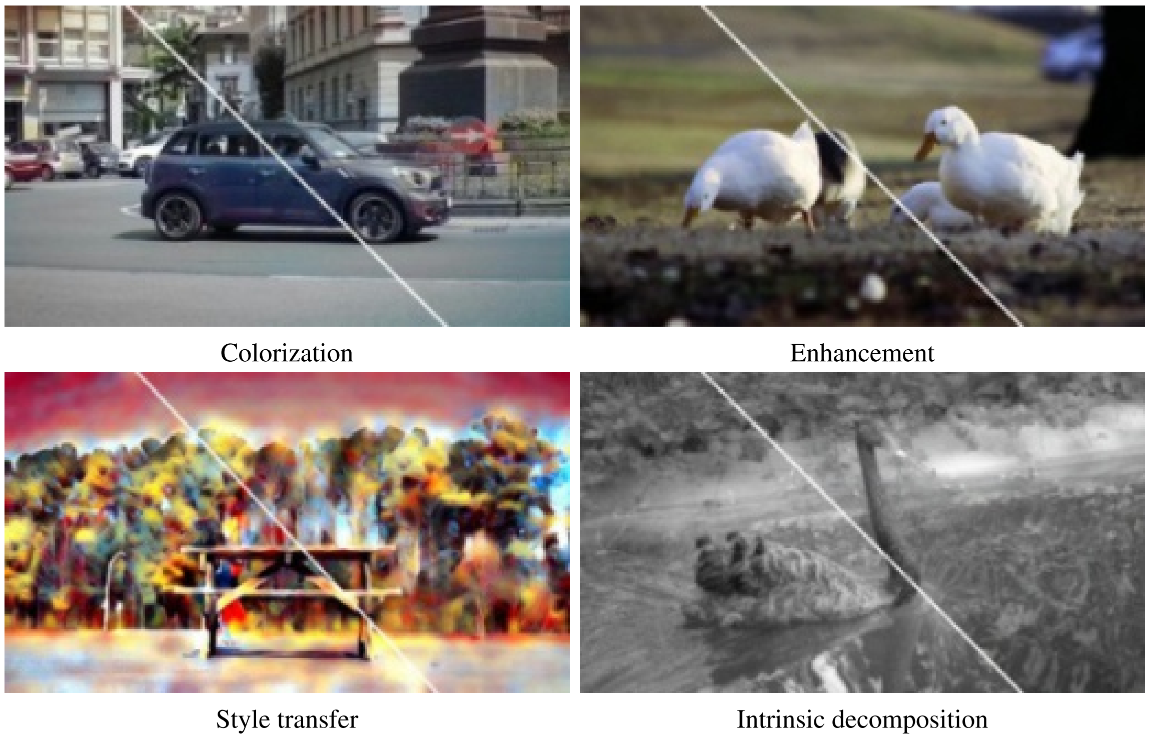


Figure 4.1: **Applications of the proposed method.** Our algorithm takes per-frame processed videos with serious temporal flickering as inputs (lower-left) and generates temporally stable videos (upper-right) while maintaining perceptual similarity to the processed frames. Our method is blind to the specific image processing algorithm applied to input videos and runs a high frame-rates. This figure contains *animated videos*, which are best viewed using Adobe Acrobat.

results of Bonneel et al. [11] are temporally stable, their algorithm highly depends on the quality of dense correspondence (e.g., optical flow or PatchMatch [6]) and may fail when a severe occlusion occurs. Yao et al. [124] further extend the method of Bonneel et al. [11] to account for occlusion by selecting a set of key-frames. However, the computational cost increases linearly with the number of key-frames, and thus their approach cannot be efficiently applied to long video sequences. Furthermore, both approaches assume that the gradients of the original video are similar to the gradients of the processed video, which restricts them from handling tasks that may generate new contents (e.g., stylization).

In this work, we formulate the problem of video temporal consistency as a learning task. We propose to learn a deep recurrent network that takes the input and processed videos and generates temporally stable output videos. We minimize the short-term and long-term temporal losses between output frames and impose a perceptual loss from the pre-trained VGG network [99] to maintain the perceptual similarity between the output and processed frames. In addition, we embed a convolutional LSTM (ConvLSTM) [116] layer to capture the spatial-temporal correlation of natural videos. Our network processes video frames sequentially and can be applied to videos with arbitrary lengths. Furthermore, our model does not require computing optical flow at *test* time and thus can process videos at real-time rates ($400+$ FPS on $1280 \times 720$ videos).

As existing video datasets typically contain low-quality frames, we collect a high-quality video dataset with 80 videos for training and 20 videos for evaluation. We train our model on a wide range of applications, including colorization, image enhancement, and artistic style transfer, and demonstrate that a *single* trained model generalizes well to *unseen* applications (e.g., intrinsic image decomposition, image-to-image translation), as shown in Figure 4.1. We evaluate the quality of the output videos using temporal warping error and a learned perceptual metric [131]. We show that the proposed method strikes a good balance between maintaining the temporal stability and perceptual similarity. Furthermore, we conduct a user study to evaluate the subjective preference between the proposed method and state-of-the-art approaches.

We make the following contributions in this work:

1. We present an efficient solution to remove temporal flickering in videos via learning a deep network with a ConvLSTM module. Our method does not require pre-computed optical flow or frame correspondences at *test* time and thus can process videos in real-time.

2. We propose to minimize the short-term and long-term temporal loss for improving the temporal stability and adopt a perceptual loss to maintain the perceptual similarity.

3. We provide a *single* model for handling multiple applications, including but not limited to colorization, enhancement, artistic style transfer, image-to-image translation and intrinsic image decomposition. Extensive subject and objective evaluations demonstrate that the proposed algorithm performs favorably against existing approaches on various types of videos.

## 4.2 Learning Temporal Consistency

In this section, we describe the proposed recurrent network and the design of the loss functions for enforcing temporal consistency on videos.

### 4.2.1 Recurrent Network

Figure 4.2 shows an overview of the proposed recurrent network. Our model takes as input the original (unprocessed) video $\{I_t | t = 1 \cdots T\}$ and per-frame processed videos $\{P_t | t = 1 \cdots T\}$, and produces temporally consistent output videos $\{O_t | t = 1 \cdots T\}$. In order to efficiently process videos with arbitrary length, we develop an image transformation network as a *recurrent convolutional network* to generate output frames in an online manner (i.e., sequentially from $t = 1$ to $T$). Specifically, we set the first output frame $O_1 = P_1$. In each time step, the network learns to generate an output frame $O_t$ that is temporally consistent with respect to $O_{t-1}$. The current output frame is then fed as the input at the next time step. To capture the spatial-temporal correlation of videos, we integrate a ConvLSTM layer [116] into our image transformation network. We discuss the detailed design of our image transformation network in Section 4.2.3.

### 4.2.2 Loss Functions

Our goal is to reduce the temporal inconsistency in the output video while maintaining the perceptual similarity with the processed frames. Therefore, we propose to train our model with (1) a perceptual content loss between the output frame and the processed frame and (2) short-term and long-term temporal losses between output frames.

**Content perceptual loss.** We compute the similarity between $O_t$ and $P_t$ using the perceptual loss from a pre-trained VGG classification network [99], which is commonly adopted in several applications (e.g., style transfer [56], super-resolution [66], and image inpainting [86]) and has been shown to correspond well to human perception [131]. The perceptual loss is defined as:

$$\mathcal{L}_p = \sum_{t=2}^{T} \sum_{i=1}^{N} \sum_{l} \left\| \phi_l(O_t^{(i)}) - \phi_l(P_t^{(i)}) \right\|_1, \tag{4.1}$$

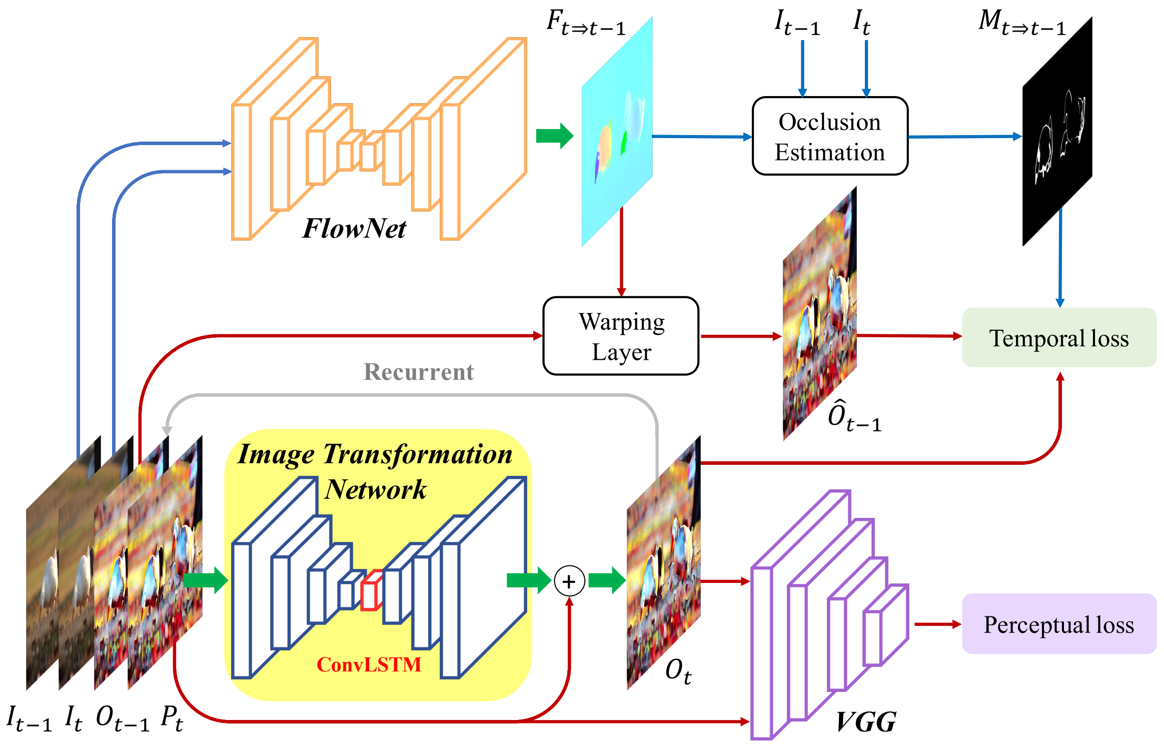


Figure 4.2: **Overview of the proposed method.** We train an image transformation network that takes $I_{t-1}, I_t, O_{t-1}$ and processed frame $P_t$ as inputs and generates the output frame $O_t$ which is temporally consistent with the output frame at the previous time step $O_{t-1}$. The output $O_t$ at the current time step then becomes the input at the next time step. We train the image transformation network with the VGG perceptual loss and the short-term and long-term temporal losses.

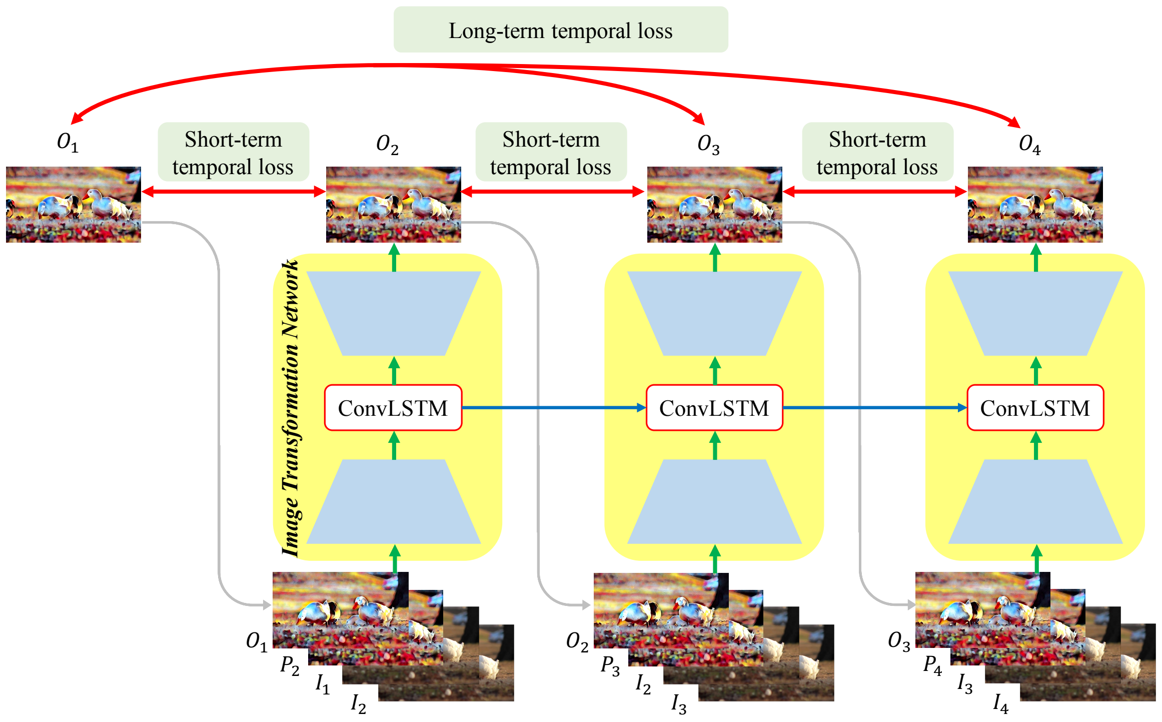


Figure 4.3: **Temporal losses.** We adopt the short-term temporal loss on neighbor frames and long-term temporal loss between the first and all the output frames.

where $O_t^{(i)}$ represents a vector $\in R^3$ with RGB pixel values of the output $O$ at time $t$, $N$ is the total number of pixels in a frame, and $\phi_l(\cdot)$ denotes the feature activation at the $l$-th layer of the VGG-19 network $\phi$. We choose the 4-th layer (i.e., `relu4-3`) to compute the perceptual loss.

**Short-term temporal loss.** We formulate the temporal loss as the warping error between the output frames:

$$\mathcal{L}_{st} = \sum_{t=2}^{T} \sum_{i=1}^{N} M_{t \Rightarrow t-1}^{(i)} \left\| O_t^{(i)} - \hat{O}_{t-1}^{(i)} \right\|_1 , \tag{4.2}$$

where $\hat{O}_{t-1}$ is the frame $O_{t-1}$ warped by the optical flow $F_{t \Rightarrow t-1}$, and $M_{t \Rightarrow t-1} = \exp(-\alpha \| I_t - \hat{I}_{t-1} \|_2^2)$ is the visibility mask calculated from the warping error between input frames $I_t$ and warped input frame $\hat{I}_{t-1}$. The optical flow $F_{t \Rightarrow t-1}$ is the backward flow between $I_{t-1}$ and $I_t$. We use the FlowNet2 [51] to efficiently compute flow on-the-fly during training. We use the bilinear sampling layer [53] to warp frames and empirically set $\alpha = 50$ (with pixel range between $[0, 1]$).

**Long-term temporal loss.** While the short-term temporal loss (4.2) enforces the temporal consistency between consecutive frames, there is no guarantee for long-term (e.g., more than 5 frames) coherence. A straightforward method to enforce long-term temporal consistency is to apply the temporal loss on *all* pairs of output frames. However, such a strategy requires significant computational costs (e.g., optical flow estimation) during training. Furthermore, computing temporal loss between two intermediate outputs is not meaningful before the network converges.

Instead, we propose to impose long-term temporal losses between the *first* output frame and all of the output frames:

$$\mathcal{L}_{lt} = \sum_{t=2}^{T} \sum_{i=1}^{N} M_{t \Rightarrow 1}^{(i)} \left\| O_t^{(i)} - \hat{O}_1^{(i)} \right\|_1 . \tag{4.3}$$

We illustrate an unrolled version of our recurrent network as well as the short-term and long-term losses in Figure 4.3. During the training, we enforce the long-term temporal coherence over a maximum of 10 frames ($T = 10$).

**Overall loss.** The overall loss function for training our image transformation network is defined as:

$$\mathcal{L} = \lambda_p \mathcal{L}_p + \lambda_{st} \mathcal{L}_{st} + \lambda_{lt} \mathcal{L}_{lt}, \tag{4.4}$$

where $\lambda_p$, $\lambda_{st}$ and $\lambda_{lt}$ are the weights for the content perceptual loss, short-term and long-term losses, respectively.

### 4.2.3 Image Transformation Network

The input of our image transformation network is the concatenation of the currently processed frame $P_t$, previous output frame $O_{t-1}$ as well as the current and previous unprocessed frames $I_t, I_{t-1}$. As the output frame typically looks similar to the currently processed frame, we train the network to predict the *residuals* instead of actual pixel values, i.e., $O_t = P_t + \mathcal{F}(P_t)$, where $\mathcal{F}$ denotes the image transformation network. Our image transformation network consists of two strided convolutional layers, $B$ residual blocks, one ConvLSTM layer, and two transposed convolutional layers.

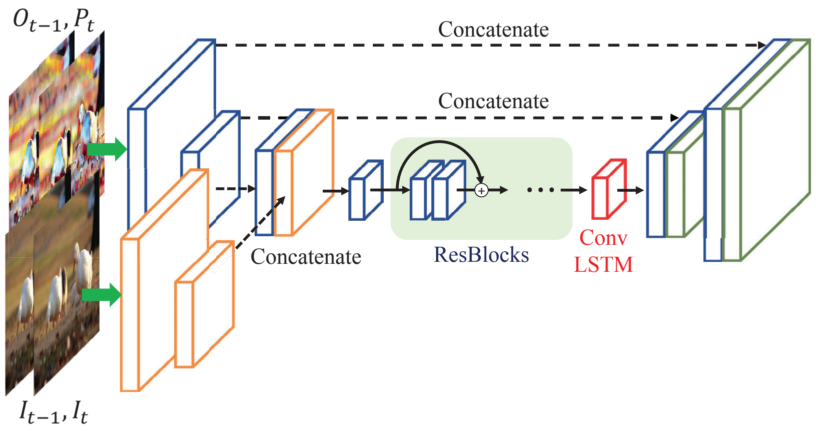


Figure 4.4: **Architecture of our image transformation network.** We split the input into two streams to avoid transferring low-level information from the input frames to output.

We add skip connections from the encoder to the decoder to improve the reconstruction quality. However, for some applications, the processed frames may have a dramatically different appearance than the input frames (e.g., style transfer or intrinsic image decomposition). We observe that the skip connections may transfer low-level information (e.g., color) to the output frames and produce visual artifacts. Therefore, we divide the input into two streams: one for the processed frames $P_t$ and $O_{t-1}$, and the other stream for input frames $I_t$ and $I_{t-1}$. As illustrated in Figure 4.4, the skip connections only add skip connections from the processed frames to avoid transferring the low-level information from the input frames.

## 4.2.4 Implementation Details

We implement our model using PyTorch [85]. We use a kernel of size $7 \times 7$ for the first and the last convolutional layers and $3 \times 3$ for all other convolutional and transposed convolutional layers. The number of filters is 32 and is multiplied by 2 when the feature maps are downsampled. All the convolutional and transposed convolutional layers (except the last layers) are followed by the instance normalization [108] and leaky ReLUs (LReLU) [80] with a negative slope of 0.2. There are 5 residual blocks between the encoder and decoder. At the end of the decoder, we use a Tanh layer to constrain the range of the output into $[-1, 1]$.

During training, we use a batch size of 4 (i.e., 4 sequences). For each sequence, we sample 10 consecutive frames, which means that the long-term temporal coherence is enforced over a maximum of 10 frames. We run the forward pass of all 10 frames before updating the network parameters. We randomly crop video frames to $192 \times 192$ and apply the data augmentation of random scaling between $[1, 2]\times$, random rotation for $90°$, $180°$ or $270°$, and horizontal flipping. The same geometric transform is applied to all the frames in the same video. We also adopt a temporal augmentation by reversing the order of sequences. The initial learning rate is set to $1e - 4$ and decreased by a factor of 2 for every 20,000 iterations. We train our model with the ADAM solver [60] for 100,000 iterations. During the training phase, only the image transformation network is updated while the FlowNet and VGG are fixed.

## 4.3 Experimental Results

In this section, we first describe the employed datasets for training and testing, followed by the applications of the proposed method and the metrics for evaluating the temporal stability and perceptual similarity. We then analyze the effect of each loss term in balancing the temporal coherence and perceptual similarity, conduct quantitative and subjective comparisons with existing approaches, and finally discuss the limitations of our method. The source code and datasets are publicly available at `http://vllab.ucmerced.edu/wlai24/video_consistency`.

### 4.3.1 Datasets

We use the DAVIS-2017 dataset [88], which is designed for video segmentation and contains a variety of moving objects and motion types. The DAVIS dataset has 60 videos for training and 30 videos for validation. However, the lengths of the videos in the DAVIS dataset are usually short (less than 3 seconds) with 4,209 training frames in total. Therefore, we collect additional 100 high-quality videos from Videvo.net [111], where 80 videos are used for training and 20 videos for testing. We scale the height of video frames to 480 and keep the aspect ratio. We use both the DAVIS and VIDEVO training sets, which contains a total of 25,735 frames, to train our network.

### 4.3.2 Applications

As we do not make any assumptions on the underlying image-based algorithms, our method is applicable for handling a wide variety of applications.

**Artistic style transfer.** The tasks of image style transfer have been shown to be sensitive to minor changes in content images due to the non-convexity of the Gram matrix matching objective [37]. We apply our method to the results from the state-of-the-art style transfer approaches [56, 72].

**Colorization.** Single image colorization aims to hallucinate plausible colors from a given grayscale input image. Recent algorithms [50, 130] learn deep CNNs from millions of natural images. When applying colorization methods to a video frame-by-frame, those approaches typically produce low-frequency flickering.

**Image enhancement.** Gharbi et al. [33] train deep networks to learn the user-created action scripts of Adobe Photoshop for enhancing images. Their models produce high-frequency flickering on most of the videos.

**Intrinsic image decomposition.** Intrinsic image decomposition aims to decompose an image into a reflectance and a shading layer. The problem is highly ill-posed due to the scale ambiguity. We apply the approach of Bell et al. [7] to our test videos. As expected, the image-based algorithm produces serious temporal flickering artifacts when applied to each frame in the video independently.

**Image-to-image translation.** In recent years, the image-to-image translation tasks attract considerable attention due to the success of the Generative Adversarial Networks (GAN) [36]. The CycleGAN model [132] aims to learn mappings from one image domain to another domain without using paired training data. When the transformations generate a new texture on images (e.g., photo $\rightarrow$ painting, horse$\rightarrow$ zebra) or the mapping contains multiple plausible solutions (e.g., gray $\rightarrow$ RGB), the resulting videos inevitably suffer from temporal flickering artifacts.

The above algorithms are general and can be applied to any type of videos. When applied, they produce temporal flickering artifacts on most videos in our test sets. We use the WCT [72] style transfer algorithm with three style images, one of the enhancement models of Gharbi et al. [33], the colorization method of Zhang et al. [130] and the shading layer of Bell et al. [7] as our training tasks, with the rest of the tasks being used for testing purposes. We demonstrate that the proposed method learns a *single* model for multiple applications and also generalizes to *unseen* tasks.

### 4.3.3 Evaluation Metrics

Our goal is to generate a temporally smooth video while maintaining the perceptual similarity with the per-frame processed video. We use the following metrics to measure the temporal stability and perceptual similarity on the output videos.

**Temporal stability.** We measure the temporal stability of a video based on the flow warping error between two frames:

$$E_{\text{warp}}(V_t, V_{t+1}) = \frac{1}{\sum_{i=1}^{N} M_t^{(i)}} \sum_{i=1}^{N} M_t^{(i)} \| V_t^{(i)} - \hat{V}_{t+1}^{(i)} \|_2^2, \tag{4.5}$$

where $\hat{V}_{t+1}$ is the warped frame $V_{t+1}$ and $M_t \in \{0, 1\}$ is a non-occlusion mask indicating non-occluded regions. We use the occlusion detection method in [94] to estimate the mask $M_t$. The warping error of a video is calculated as:

$$E_{\text{warp}}(V) = \frac{1}{T-1} \sum_{t=1}^{T-1} E_{\text{warp}}(V_t, V_{t+1}), \tag{4.6}$$

which is the average warping error over the entire sequence.

**Perceptual similarity.** Recently, the features of the pre-trained VGG network [99] have been shown effective as a training loss to generate realistic images in several vision tasks [21, 66, 86]. Zhang et al. [131] further propose a perceptual metric by calibrating the deep features of ImageNet classification networks. We adopt the calibrated model of the SqueezeNet [49] (denote as $\mathcal{G}$) to measure the perceptual distance of the processed video $P$ and output video $O$:

$$D_{\text{perceptual}}(P, O) = \frac{1}{T-1} \sum_{t=2}^{T} \mathcal{G}(O_t, P_t). \tag{4.7}$$

We note that the first frame is fixed as a reference in both Bonneel [11] and our algorithm. Therefore, we exclude the first frame from computing the perceptual distance in (4.7).

## 4.3.4 Analysis and Discussions

In this section, we first analyze the balance between the temporal stability and perceptual similarity. We then conduct experiments to understand the effect of the temporal loss, perceptual loss, and the ConvLSTM layer. We also analyze the effect of using $L_1$ and $L_2$ norm and compare the results of multi-task and single-task training.

**Temporal stability and perceptual similarity.** An extremely blurred video may have high temporal stability but with low perceptual similarity; in contrast, the processed video itself has perfect perceptual similarity but is temporally unstable. Due to the trade-off between the temporal stability and perceptual similarity, it is important to balance these two properties and produce visually pleasing results.

To understand the relationship between the temporal and content losses, we train models with the several combinations of $\lambda_p$ and $\lambda_t$ $(= \lambda_{st} = \lambda_{lt})$. We use one of the styles (i.e., udnie) from the fast neural style transfer method [56] for evaluation. We show the quantitative evaluation on the DAVIS test set in Figure 4.5. We observe that the ratio $r = \lambda_t/\lambda_p$ plays an important role in balancing the temporal stability and perceptual similarity. When the ratio $r < 10$, the perceptual loss dominates the optimization of the network, and the temporal flickering remains in the output videos. When the ratio $r > 10$, the output videos become overly blurred and therefore have a large perceptual distance to the processed videos. When $\lambda_t$ is sufficiently large (i.e., $\lambda_t \geq 100$), the setting $r = 10$ strikes a good balance to reduce temporal flickering while maintaining small perceptual distance. Our results find similar observation on other applications as well.

**Effect of temporal and perceptual losses.** Our training objective function is a combination of the content perceptual loss $L_p$, short-term temporal $L_{st}$, and long-term temporal losses $L_{lt}$. To further analyze the effect of each loss function, we train three models by setting the weights of each loss term, $\lambda_p$, $\lambda_{st}$, and $\lambda_{lt}$, to 0, respectively. We evaluate the performance of the variants using the WCT method [72] on the DAVIS test set [88] and provide quantitative comparisons in Figure 4.6.

| $\lambda_t$ | $\lambda_p$ | $r = \frac{\lambda_t}{\lambda_p}$ | $E_{\text{warp}}$ | $D_{\text{perceptual}}$ |
|---|---|---|---|---|
| 10 | 0.01 | 1000 | 0.0279 | 0.1744 |
| 10 | 0.1 | 100 | 0.0265 | 0.1354 |
| 10 | 1 | 10 | 0.0615 | 0.0071 |
| 10 | 10 | 1 | 0.0621 | 0.0072 |
| 100 | 1 | 100 | 0.0277 | 0.1324 |
| 100 | 10 | 10 | 0.0442 | 0.0170 |
| 100 | 100 | 1 | 0.0621 | 0.0072 |
| 1000 | 1 | 1000 | 0.0262 | 0.1848 |
| 1000 | 10 | 100 | 0.0275 | 0.1341 |
| 1000 | 100 | 10 | 0.0453 | 0.0158 |
| 1000 | 1000 | 1 | 0.0621 | 0.0072 |

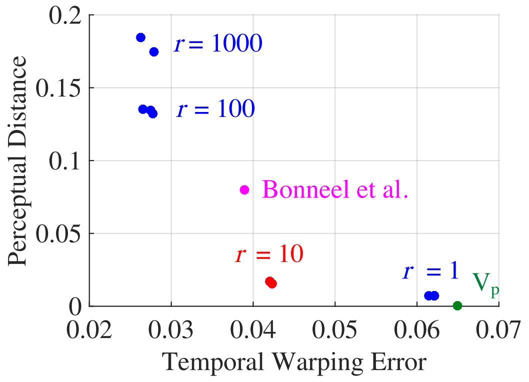


Figure 4.5: **Analysis of parameters**. (Left) When $\lambda_t$ is large enough, choosing $r = 10$ (shown in red) achieves a good balance between reducing temporal warping error as well as perceptual distance. (Right) The trade off between perceptual similarity and temporal warping with different ratios $r$, as compared to Bonneel et al. [11], and the original processed video, $V_p$.

Without the perceptual loss, the model generates blurry results. While a blurry video tends to have a low temporal warping error, the perceptual distance is large, indicating that the model cannot preserve the content of the processed video well.

Without the short-term temporal loss, the model cannot reduce the temporal flickering well. The temporal warping error is close to that of the processed video in Figure 4.6.

When training without the long-term temporal loss, the model does not capture the long-term temporal coherence well and thus is prone to error propagation and occlusion. As shown in Figure 4.7(e), the blue regions on the ground suddenly change into different colors after a man passing by. On the contrary, the model trained with all the losses produces stable results without temporal flickering.

**Effect of LSTM.** To analyze the effect of the ConvLSTM layer, we train an image transformation network without the ConvLSTM layer. To use the same amount of network parameters, we increase the number of residual blocks from 5 to 9 in this model. We show an example of stabilizing the results of a colorization method [50] on the VIDEVO dataset in Figure 4.8. The model without the ConvLSTM layer produces propagation errors (as

| Method | $E_{\text{warp}}$ | $D_{\text{perceptual}}$ |
|---|---|---|
| $V_p$ | 0.054 | 0 |
| Bonneel et al. [11] | 0.0312 | 0.0977 |
| Ours w/o $L_p$ | 0.0222 | 0.1850 |
| Ours w/o $L_{st}$ | 0.0518 | 0.0063 |
| Ours w/o $L_{lt}$ | 0.0427 | 0.0132 |
| Our full model | 0.0348 | 0.0194 |

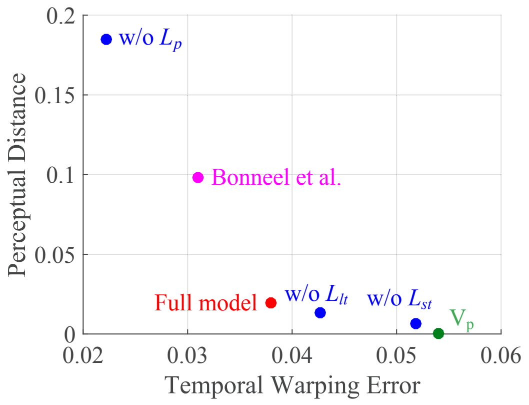


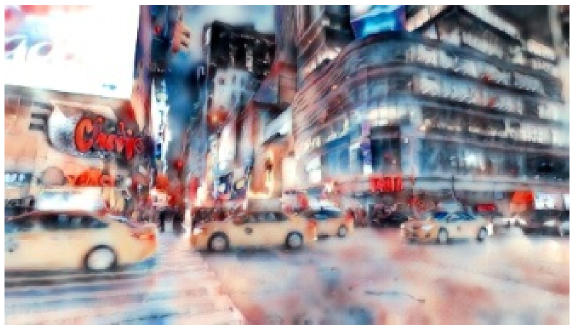
Figure 4.6: **Analysis on loss functions**. (Left) We analyze the contribution of each loss by setting the weight of each term to 0, respectively. (Right) The trade off between perceptual similarity and temporal warping with different loss functions, as compared to Bonneel et al. [11], and the original processed video, $V_p$.

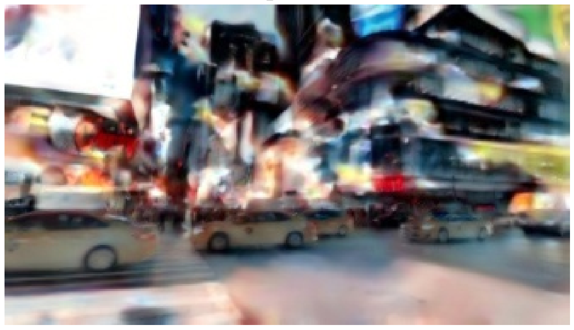
shown on the ground of Figure 4.8(c)). Our model with the ConvLSTM layer successfully captures the spatio-temporal correlation of the original input video and produces more visually pleasing videos.

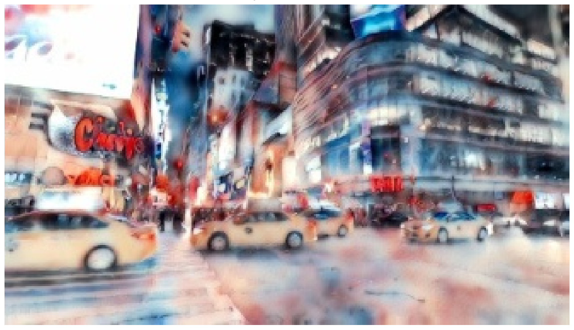
**$L_2$ norm v.s. $L_1$ norm.** We choose to use the $L_1$ loss as it is a robust loss function commonly used in several vision tasks, e.g., super-resolution [61] and inpainting [**?**]. However, we find that the choice of the loss function is not crucial in the proposed model. Here we train our model using the $L_2$ loss for computing the content and temporal losses and show the trade-off curve in Figure 4.9(a). When setting $r = 100$, the model using the $L_2$ loss performs similarly to that using the $L_1$ loss function. The model optimized with the $L_2$ loss can achieve comparable performance as our current model with a proper weights setting, i.e., adjusting $r = \lambda_t/\lambda_p$.

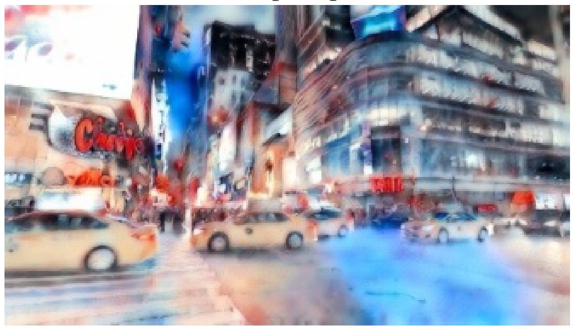
**Multi-task vs. single-task training.** We train three single-task models using one style image for the WCT [72] (denoted by $M_{\text{WCT}}$), one enhancement model of the DBL [2] (denoted by $M_{\text{DBL}}$), and the shading layer of the intrinsic decomposition algorithm [7] (denoted by $M_{\text{I}}$), respectively. We evaluate the temporal warping error and perceptual distance on the DAVIS test set and plot the trade-off curve between the average warping error and perceptual distance in Figure 4.9(b). It is interesting that the single-task models

(a) Input video


(b) Stylized video


(c) Without perceptual loss


(d) Without short-term temporal loss


(e) Without long-term temporal loss

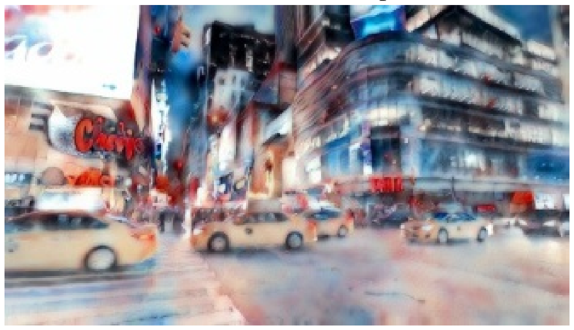
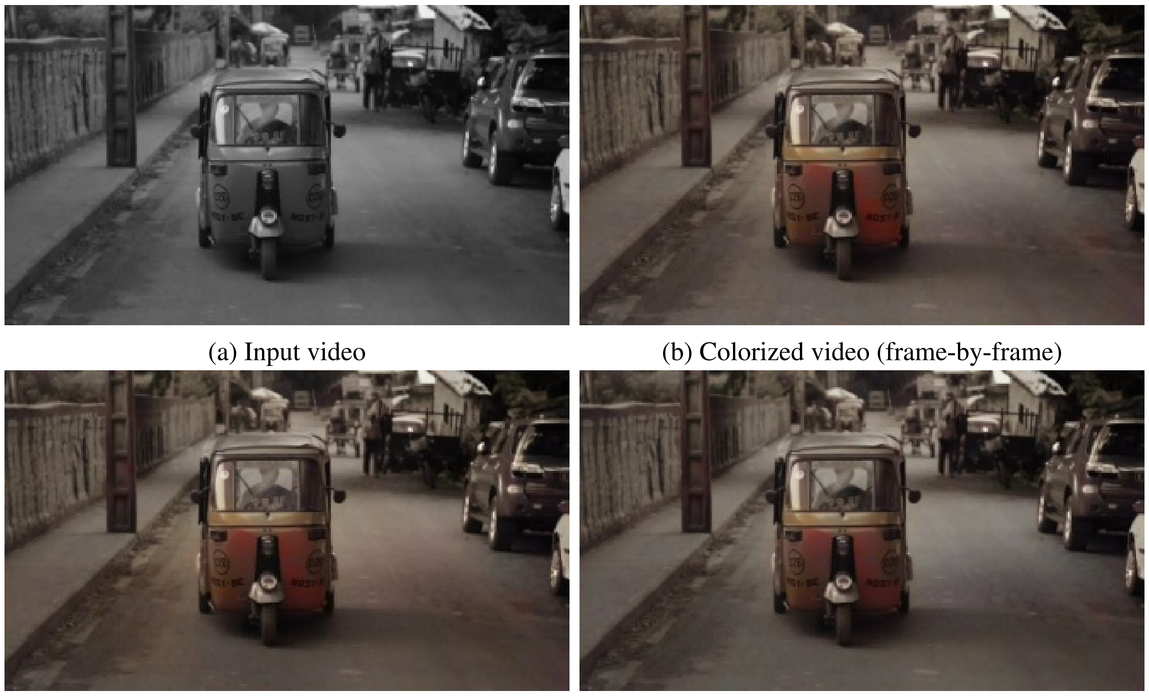

(f) Ours

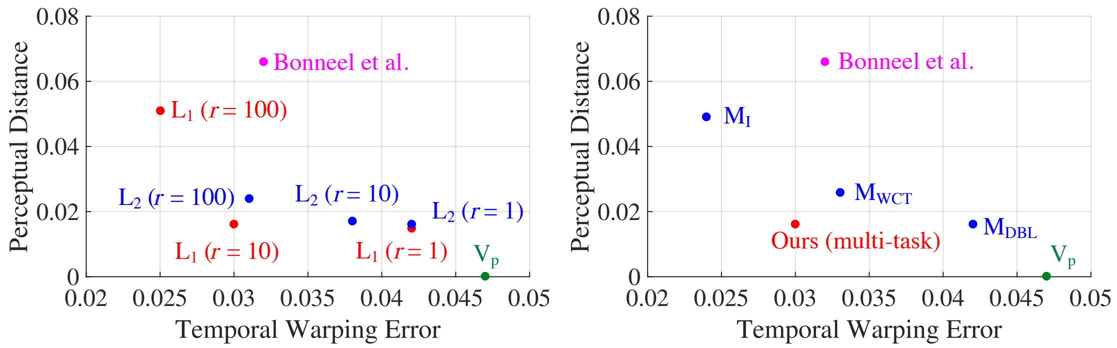
Figure 4.7: **Effect of loss functions.** Without the perceptual content loss, the results are overly smooth and have a low perceptual similarity with the processed video. While the short-term temporal loss is crucial to remove the high-frequency flickering, the long-term temporal loss further reduces low-frequency jitter and avoids error propagation (e.g., the lower-right corner in (e)). This figure contains *animated videos*, which are best viewed using Adobe Acrobat.



(a) Input video (b) Colorized video (frame-by-frame)

(c) Without ConvLSTM (d) With ConvLSTM

Figure 4.8: **Effect of ConvLSTM layer.** The model trained without the ConvLSTM layer produces propagation errors, while our full model generates more visually pleasing videos. This figure contains *animated videos*, which are best viewed using Adobe Acrobat.



(a) $L_1$ vs. $L_2$ loss (b) Single-task vs. multi-task training

Figure 4.9: **Analysis on loss function and multi-task training.**

do not always achieve the lowest temporal warping error and perceptual distance on the same task used in training. As the single-task training is susceptible to overfitting for the specific task, the single-task models may generate more artifacts and do not generalize well to multiple tasks. In contrast, the multi-task model maintains small temporal warping error and has the lowest perceptual distance.

### 4.3.5 Comparison with State-of-the-arts

We evaluate the temporal warping error (4.6) and perceptual distance (4.7) on the two video test sets. We compare the proposed method with Bonneel et al. [11] on 16 applications: 2 styles of Johnson et al. [56], 6 styles of WCT [72], 2 enhancement models of Gharbi et al. [33], reflectance and shading layers of Bell et al. [7], 2 photo-to-painting models of CycleGAN [132] and 2 colorization algorithms [50, 130]. We provide the average temporal warping error and perceptual distance in Table 4.1 and Table 4.2, respectively. In general, our results achieves lower perceptual distance while maintains comparable temporal warping error with the results of Bonneel et al. [11].

We show visual comparisons with Bonneel et al. [11] in Figure 4.10 and 4.11. Although the method of Bonneel et al. [11] produces temporally stable results, the assumption of identical gradients in the processed and original video leads to overly smoothed contents, for example from stylization effects. Furthermore, when the occlusion occurs in a large region, their method fails due to the lack of a long-term temporal constraint. In contrast, the proposed method dramatically reduces the temporal flickering while maintaining the perceptual similarity with the processed videos. We note that our approach is not limited to the above applications but can also be applied to tasks such as automatic white balancing [44], image harmonization [9] and image dehazing [39]. Due to the space limit, we provide more results and videos on our project website.

### 4.3.6 Subjective Evaluation

We conduct a user study to measure user preference on the quality of videos. We adopt the pairwise comparison, i.e., we ask participants to choose from a pair of videos. In each test, we provide the original and processed videos as references and show two results (Bonneel et al. [11] and ours) for comparisons. We randomize the presenting order of the

Table 4.1: **Quantitative evaluation on temporal warping error**. The "Trained" column indicates the applications used for training our model. Our method achieves a similarly reduced temporal warping error as Bonneel et al. [11], which is significantly less than the original processed video ($V_p$).

| Task | Trained | DAVIS | | | VIDEVO | | |
|---|---|---|---|---|---|---|---|
| | | $V_p$ | [11] | Ours | $V_p$ | [11] | Ours |
| WCT [72]/antimono | ✓ | 0.054 | **0.031** | 0.035 | 0.025 | 0.014 | **0.013** |
| WCT [72]/asheville | | 0.088 | **0.047** | 0.055 | 0.045 | 0.025 | **0.023** |
| WCT [72]/candy | ✓ | 0.069 | **0.037** | 0.045 | 0.034 | **0.018** | **0.018** |
| WCT [72]/feathers | | 0.052 | **0.029** | **0.029** | 0.027 | 0.016 | **0.012** |
| WCT [72]/sketch | ✓ | 0.046 | 0.028 | **0.023** | 0.022 | 0.015 | **0.009** |
| WCT [72]/wave | | 0.049 | 0.030 | **0.027** | 0.026 | 0.015 | **0.011** |
| Fast-neural-style [56]/princess | | 0.073 | 0.048 | **0.047** | 0.039 | 0.023 | **0.021** |
| Fast-neural-style [56]/udnie | | 0.065 | **0.039** | 0.042 | 0.028 | 0.017 | **0.015** |
| DBL [33]/expertA | ✓ | 0.039 | 0.035 | **0.028** | 0.018 | 0.016 | **0.010** |
| DBL [33]/expertB | | 0.034 | 0.031 | **0.025** | 0.015 | 0.014 | **0.008** |
| Intrinsic [7]/reflectance | | 0.024 | 0.020 | **0.015** | 0.012 | 0.008 | **0.005** |
| Intrinsic [7]/shading | ✓ | 0.016 | 0.012 | **0.009** | 0.008 | 0.006 | **0.003** |
| CycleGAN [132]/photo2ukiyoe | | 0.037 | 0.030 | **0.026** | 0.019 | 0.016 | **0.010** |
| CycleGAN [132]/photo2vangogh | | 0.040 | 0.032 | **0.029** | 0.021 | 0.017 | **0.013** |
| Colorization [130] | ✓ | 0.030 | 0.028 | **0.024** | 0.012 | 0.011 | **0.008** |
| Colorization [50] | | 0.030 | 0.028 | **0.023** | 0.012 | 0.011 | **0.008** |
| Average | | 0.047 | 0.032 | **0.030** | 0.023 | 0.015 | **0.012** |

Table 4.2: **Quantitative evaluation on perceptual distance**. Our method has lower perceptual distance than Bonneel et al. [11].

| Task | Trained | DAVIS | | VIDEVO | |
|---|---|---|---|---|---|
| | | [11] | Ours | [11] | Ours |
| WCT [72]/antimono | ✓ | 0.098 | **0.019** | 0.106 | **0.016** |
| WCT [72]/asheville | | 0.090 | **0.019** | 0.098 | **0.015** |
| WCT [72]/candy | ✓ | 0.133 | **0.023** | 0.139 | **0.018** |
| WCT [72]/feathers | | 0.093 | **0.016** | 0.100 | **0.011** |
| WCT [72]/sketch | ✓ | 0.042 | **0.021** | 0.046 | **0.014** |
| WCT [72]/wave | | 0.065 | **0.015** | 0.072 | **0.013** |
| Fast-neural-style [56]/princess | | 0.143 | **0.029** | 0.165 | **0.018** |
| Fast-neural-style [56]/udnie | | 0.070 | **0.017** | 0.076 | **0.014** |
| DBL [33]/expertA | ✓ | 0.026 | **0.011** | 0.033 | **0.007** |
| DBL [33]/expertB | | 0.023 | **0.011** | 0.030 | **0.007** |
| Intrinsic [7]/reflectance | | 0.044 | **0.013** | 0.056 | **0.008** |
| Intrinsic [7]/shading | ✓ | 0.029 | **0.017** | 0.032 | **0.009** |
| CycleGAN [132]/photo2ukiyoe | | 0.042 | **0.012** | 0.054 | **0.007** |
| CycleGAN [132]/photo2vangogh | | 0.067 | **0.016** | 0.079 | **0.011** |
| Colorization [130] | ✓ | 0.062 | **0.013** | 0.055 | **0.009** |
| Colorization [50] | | 0.033 | **0.011** | 0.034 | **0.008** |
| Average | | 0.088 | **0.017** | 0.073 | **0.012** |

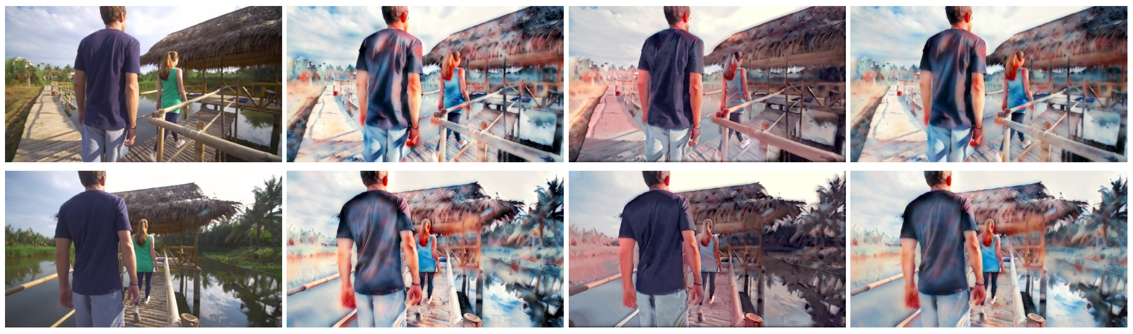

(a) Original frames (b) Processed frames (c) Bonneel et al. [11] (d) Ours

Figure 4.10: **Visual comparisons on style transfer.** We compare the proposed method with Bonneel et al. [11] on smoothing the results of WCT [72]. Our approach maintains the stylized effect of processed video and reduce the temporal flickering.

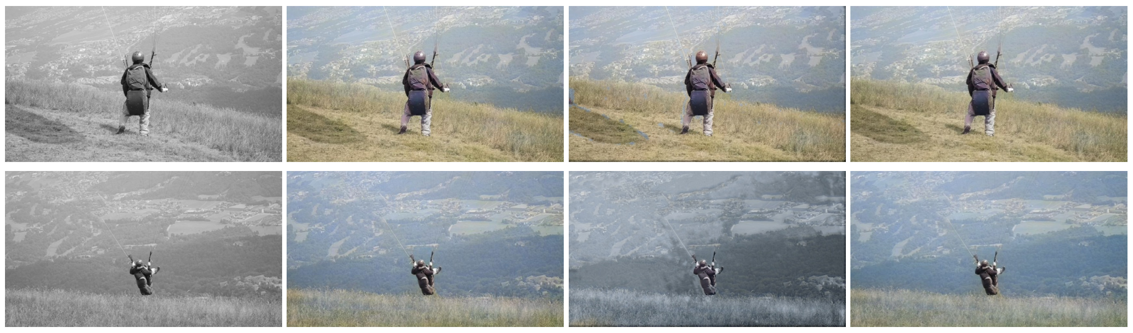


Figure 4.11: **Visual comparisons on colorization.** We compare the proposed method with Bonneel et al. [11] on smoothing the results of image colorization [50]. The method of Bonneel et al. [11] cannot preserve the colorized effect when occlusion occurs.

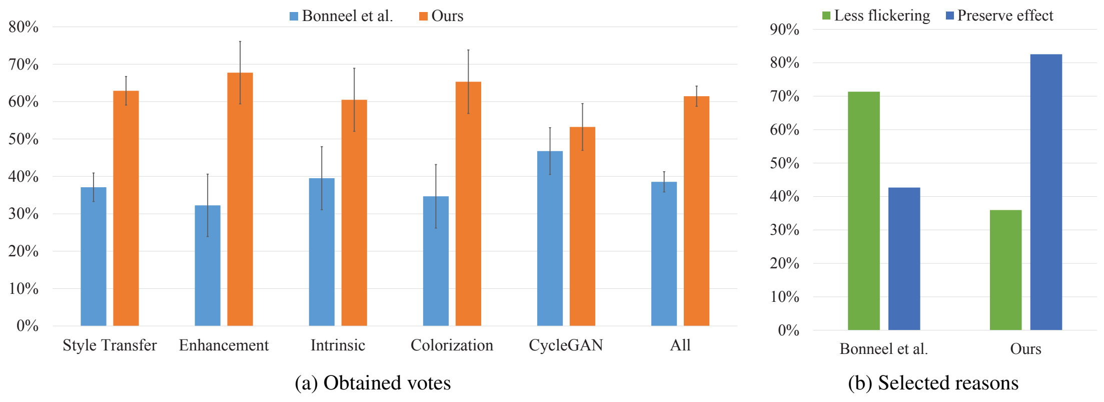


Figure 4.12: **Subjective evaluation.** On average, our method is preferred by $62\%$ users. The error bars show the $95\%$ confidence interval.

result videos in each test. In addition, we ask participants to provide the reasons that they prefer the selected video from the following options: (1) The video is less flickering. (2) The video preserves the effect of the processed video well.

We evaluate all 50 test videos with the 10 test applications that were held out during training. We ask each user to compare 20 video pairs and obtain results from a total of 60 subjects. Figure 4.12(a) shows the percentage of obtained votes, where our approach is preferred on all 5 applications. In Figure 3.18(b), we show the reasons when a method is selected. The results of Bonneel et al. [11] are selected due to temporal stability, while users prefer our results as we preserve the effect of the processed video well. The observation in the user study basically follows the quantitative evaluation in Section 4.3.5.

### 4.3.7 Execution Time

We evaluate the execution time of the proposed method and Bonneel et al. [11] on a machine with a 3.4 GHz Intel i7 CPU (64G RAM) and an Nvidia Titan X GPU. As the proposed method does not require computing optical flow at test time, the execution speed achieves 418 FPS on GPU for videos with a resolution of $1280 \times 720$. In contrast, the speed of Bonneel et al. [11] is 0.25 FPS on CPU.

### 4.3.8 Limitations and Discussion

While our experimental results show that the proposed recurrent network performs well on a variety of videos and generalizes well to multiple tasks, we do observe some failure cases as shown in Figure 4.13, where the brown color in the mountain region is wrongly propagated to the sky.

Our approach is not able to handle applications that generate entirely different image content on each frame, e.g., image completion [**?**] or synthesis [21]. Extending those methods to videos would require incorporating strong video priors or temporal constraints, most likely into the design of the specific algorithms themselves.

In addition, in the way the task is formulated there is always a trade-off between being temporally coherent or perceptually similar to the processed video. Depending on the specific effect applied, there will be cases where flicker (temporal instability) is preferable to blur, and vice versa. In our current method, the user can choose a model based on their preference for flicker or blur, but an interesting area for future work would be to investigate perceptual models for what is considered acceptable flicker and acceptable blur. Nonetheless, we use the same trained model (same parameters) for all our results and showed clear viewer preference over prior methods for blind temporal stability.

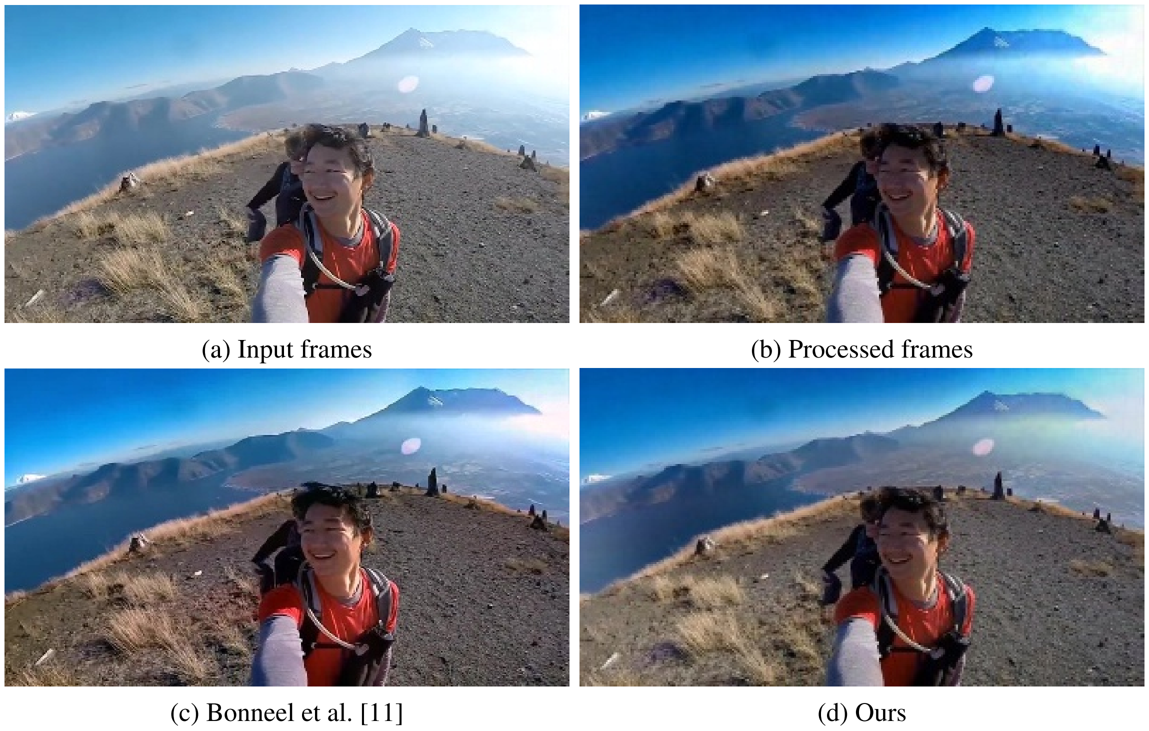


Figure 4.13: **Failure case.** The brown color in the mountain region is wrongly propagated to the sky. This figure contains animated videos, which are best viewed using Adobe Acrobat.

## 4.4 Conclusions

In this work, we propose a deep recurrent neural network to reduce the temporal flickering problem in per-frame processed videos. We optimize both short-term and long-term temporal loss as well as a perceptual loss to reduce temporal instability while preserving the perceptual similarity to the processed videos. Our approach is agnostic to the underlying image-based algorithms applied to the video and generalize to a wide range of unseen applications. We demonstrate that the proposed algorithm performs favorably against existing blind temporal consistency method on a diverse set of applications and various types of videos.

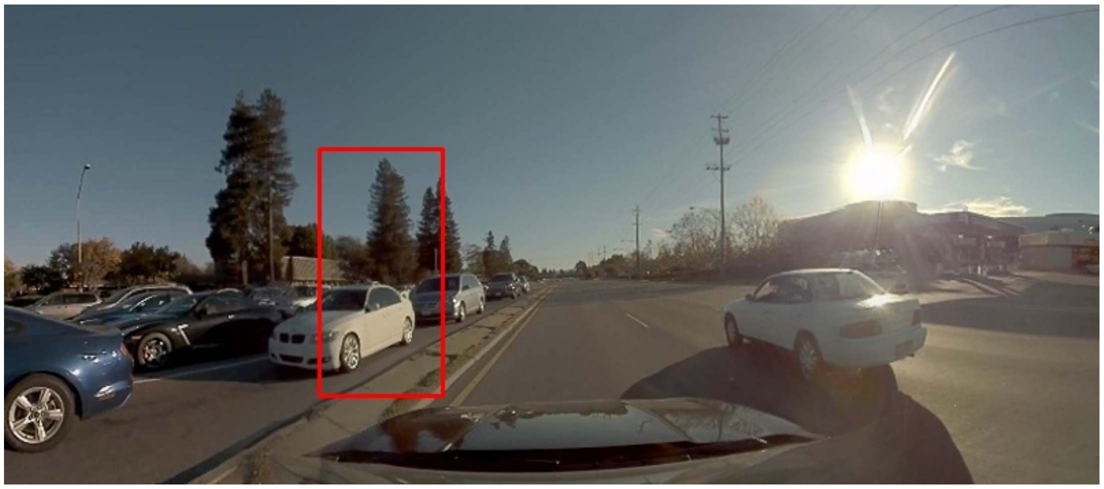
# Chapter 5

# Learning to Stitch Videos for Structured Camera Arrays

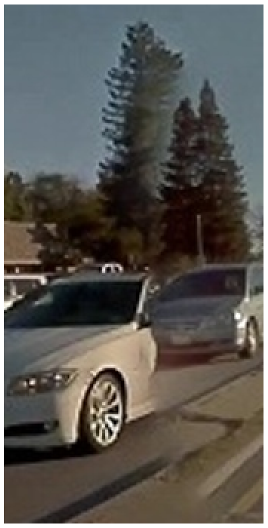
## 5.1 Introduction

Due to sensor resolution and optics limitations, the field of view (FOV) of most cameras is too narrow for applications such as autonomous driving and virtual reality. A common solution is to stitch the outputs of multiple cameras into a panoramic video, effectively extending the FOV. When the optical centers of these cameras are nearly co-located, stitching can be solved with a simple homography transformation. However, in many applications, such as autonomous driving, remote video conference, and video surveillance, multiple cameras have to be placed with *wide baselines*, either to increase view coverage or due to some physical constraints. In these cases, even state-of-the-art methods [55, 89] and current commercial solutions (e.g., VideoStitch Studio [110], AutoPano Video [4], and NVIDIA VRWorks [83]) struggle to produce artifact-free videos, as shown in Figure 5.1.

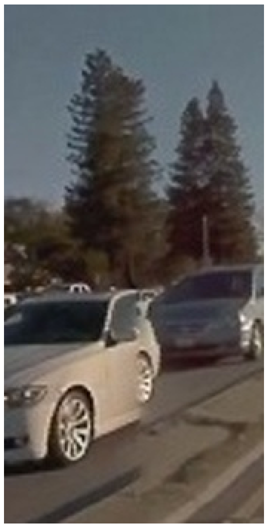
One main challenge for video stitching with wide baselines is *parallax*, i.e., the apparent displacement of an object in multiple input videos due to camera translation. Parallax varies with object depth, which makes it impossible to properly align objects without knowing dense 3D information of the scene. In addition, occlusions, dis-occlusions, and small FOV overlaps also cause a significant amount of stitching artifacts. To obtain better alignment, existing image stitching algorithms perform content-aware local warping [17, 127] or find optimal seams around objects to mitigate artifacts at the transition from one view



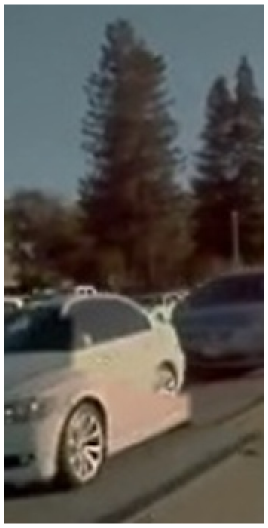
(a) Stitched video by the proposed method



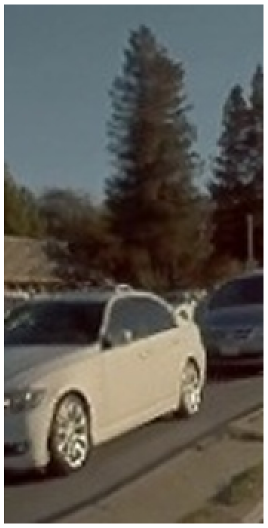
(b) VRWorks [83]



(c) AutoPano [4]



(d) STCPW [55]



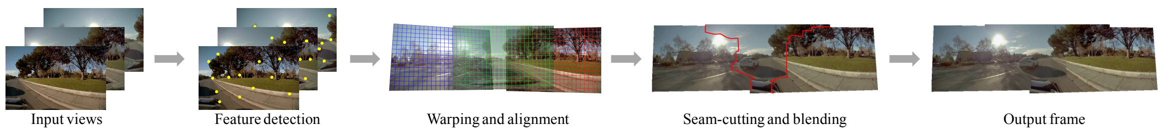
(e) Ours

Figure 5.1: **Examples of video stitching.** Inspired by pushbroom cameras, we propose a deep pushbroom stitching network to stitch multiple wide-baseline videos of dynamic scenes into a single panoramic video. The proposed learning-based algorithm compares favorably against prior work with minimal mis-alignment artifacts (e.g., ghosting and broken objects). More video results are presented in the supplementary material.

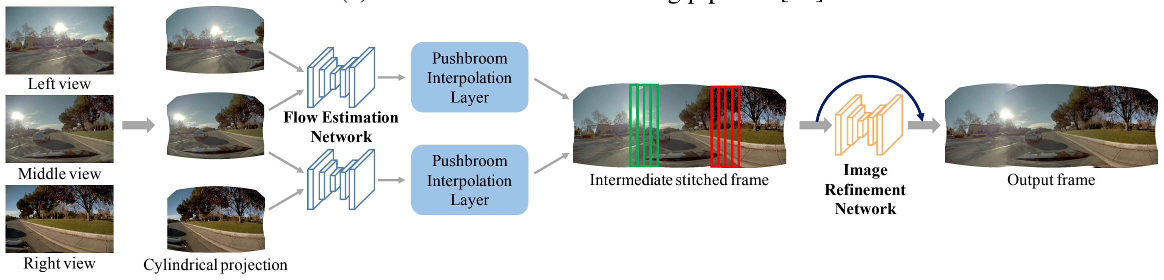
to the other [28, 129]. Applying these strategies to process a video frame-by-frame inevitably produces noticeable jittering or wobbling artifacts. On the other hand, algorithms that explicitly enforce temporal consistency, such as spatio-temporal mesh warping with a large-scale optimization [55], are computationally expensive. In fact, commercial video stitching software often adopts a simple seam cutting and multi-band blending [16] to account for efficiency. These methods, however, often cause severe artifacts, such as ghosting or misalignment, as shown in Figure 5.1. Moreover, seams can cause objects to be cut off or completely disappear from stitched images—particularly dangerous for use cases such as autonomous driving.

We identify three desirable properties for a stitched panoramic video: (1) Artifacts, such as ghosting, should not appear. (2) Objects may be distorted, but should not be cut off or disappear in any frame. (3) The stitched video needs to be temporally stable. With the three desiderata in mind, we formulate the video stitching as a spatial view interpolation problem, inspired by the multi-perspective projection for panoramas [38, 87]. To this end, we propose a pushbroom stitching network based on deep convolutional neural networks (CNNs). Specifically, we first project the input views onto a common cylindrical surface. We then estimate bi-directional optical flow, with which we *simulate* a pushbroom camera by interpolating all the intermediate views between the input views. Instead of generating all the intermediate views (which requires multiple bilinear warping steps on the entire image), we develop a *pushbroom interpolation layer* to generate the interpolated view in a single feed-forward pass. Finally, we adopt a refinement network to refine the interpolated view with the residual learning. Figure 5.2 shows an overview of the conventional video stitching pipeline and our proposed method.

Despite recent progress in optical flow estimation, the estimated flow by state-of-the-art methods are of insufficient quality for stitching. We show that it is critical to learn optical flow for stitching end-to-end, so that the flow network learns to reduce artifacts for optimal stitching results. One particular difficulty is the lack of training data as it is challenging to capture ground-truth pushbroom videos from real-world cameras. To address this issue, we render a synthetic dataset from a driving simulator for training and evaluation. This synthetic dataset enables us to develop the first learning-based approach for stitching videos. The end-to-end trained model learns locally adaptive optical flow to stitch the input videos, which significantly reduces visual artifacts and improves the



(a) Conventional video stitching pipeline [55]



(b) Proposed pushbroom stitching network

Figure 5.2: **Algorithm overview.** (a) Existing video stitching algorithm [55] solves spatio-temporal local mesh warping and 3D graph cut to align the entire video, which are often sensitive to scene content and computationally expensive. On the other hand, commercial software often adopts a simple 2D seam cutting and multi-band blending, which lead to ghosting and alignment artifacts. (b) The proposed pushbroom stitching network adopts a pushbroom interpolation layer to gradually align the input views and obtain temporally stable and artifact-free results.

temporal stability.

Finally, we conduct a human subject study for evaluation, which demonstrates that the proposed model performs favorably against existing video stitching algorithms and commercial software.

## 5.2 Stitching as Spatial Interpolation

In this section, we first illustrate the problem setup using an example of automotive. We then describe the formulation of the proposed pushbroom interpolation and the implementation details of our pushbroom stitching network.

### 5.2.1 Problem Setup

Our method produces a temporally stable stitched video from wide-baseline inputs of dynamic scenes. While the proposed approach is suitable for a generic camera configuration, here we describe it with reference to the automotive use case. Unlike other applications of structured camera arrays, in the automotive case, objects can come arbitrarily close to the cameras, thus requiring the stitching algorithm to tolerate large parallax.

For the purpose of describing the method, we define the camera setup as shown in Figure 5.3, which consists of three fisheye cameras whose baseline spans the entire car's width. Figure 5.4(a)-(c) show the typical images captured under this configuration, and underscore some of the challenges we face: strong parallax, large exposure differences, as well as geometric distortion.

To minimize the appearance change between the three views and to represent the wide FOV of the stitched frames, we first adopt a camera pose transformation to warp $C_L^i$ and $C_R^i$ to the position of $C_L^o$ and $C_R^o$, respectively. Therefore, the new origin is set at the center camera $C_M$. Then, we apply a cylindrical projection (by approximating the scene to be at infinity) to warp all the views onto a common viewing cylinder, as shown in Figure 5.3. However, even after camera calibration, exposure compensation, fisheye distortion correction, and cylindrical projection, parallax still causes significant misalignment, which results in severe ghosting, as shown in Figure 5.4(d).

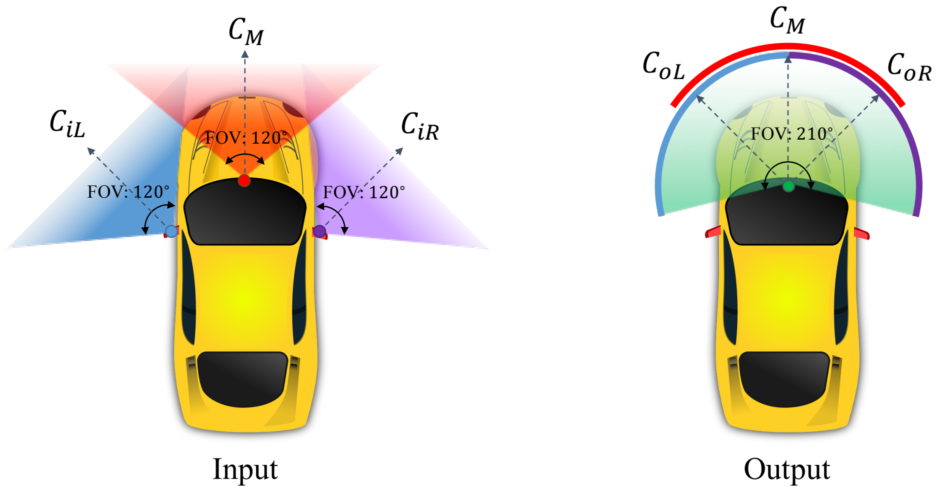


Figure 5.3: **Camera setup.** The input videos are captured from three fisheye cameras, $C_L^i$, $C_M$, and $C_R^i$. We align all the input views on a viewing cylinder centered at the same position as $C_M$.

## 5.2.2 Formulation

In this work, we cast the video stitching as a problem of spatial interpolation between the side views and the center view. We denote the output view by $\mathcal{O}$, and the input views (projected onto the output cylinder) by $I_L$, $I_M$, and $I_R$, respectively. Note that $I_L$, $I_M$, and $I_R$ are in the same coordinate system and have the same resolution. We define a *transition region* as part of the overlapping region between a pair of inputs (see the yellow regions in Figure 5.5). Within the transition region, we progressively warp $K$ vertical slices from both images to create a smooth transition from one camera to another. Outside the transition region, we directly take the pixel values from the input images without modifying them.

More formally, we take the stitching process between $I_L$ and $I_M$ as an example. We first generate $K$ intermediate frames, $\hat{I}_L^{(k)}$, which smoothly transition between $I_L$ and $I_M$. Given the flow $F_{L\to M}$ and $F_{M\to L}$, we can compute

$$\hat{I}_L^{(k)} = \mathcal{W}(I_L, \alpha_k \cdot F_{L\to M}) \tag{5.1}$$

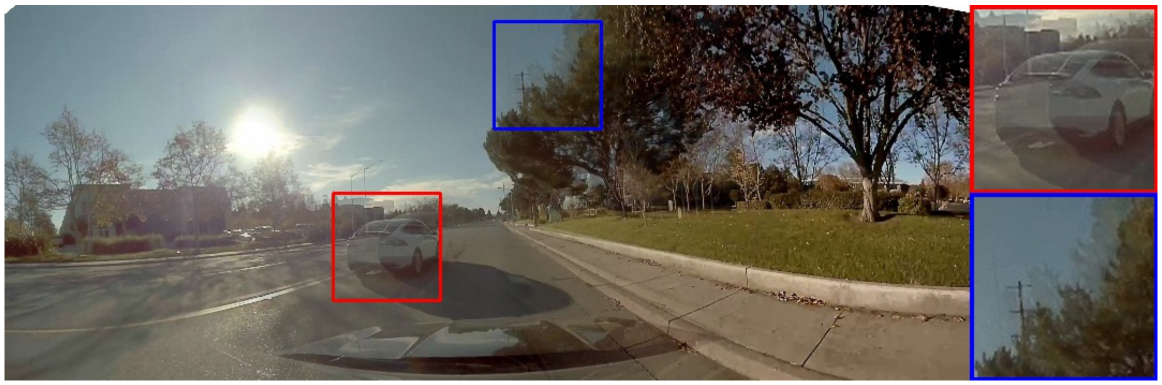
$$\hat{I}_M^{(k)} = \mathcal{W}(I_M, (1-\alpha_k) \cdot F_{M\to L}), \tag{5.2}$$

where $\mathcal{W}(I, F)$ is a function that warps image $I$ based on flow $F$, and $\alpha_k = \{k/K\}_{k=1,..,K}$ scales the flow to create the smooth transition. We define the left stitching boundary $b_L$ as the column of the leftmost valid pixel for $I_M$ on the output cylinder. Given the interpolation

(a) $I_L$

(b) $I_M$

(c) $I_R$


(d) Overlay of input views on the output cylinder

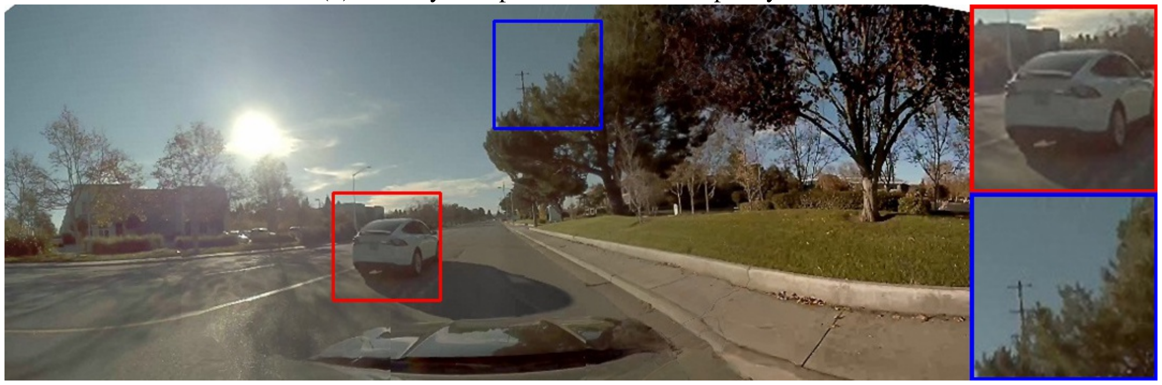

(e) Result of the pushbroom interpolation

Figure 5.4: **Example of input and stitched views.** We first project the input views $I_L$, $I_M$, and $I_R$ onto the output cylinder. The projected views on the output cylinder do not align well due to the parallax and scene depth variation. Our pushbroom interpolation method effectively stitches the views and does not produce ghosting artifacts.

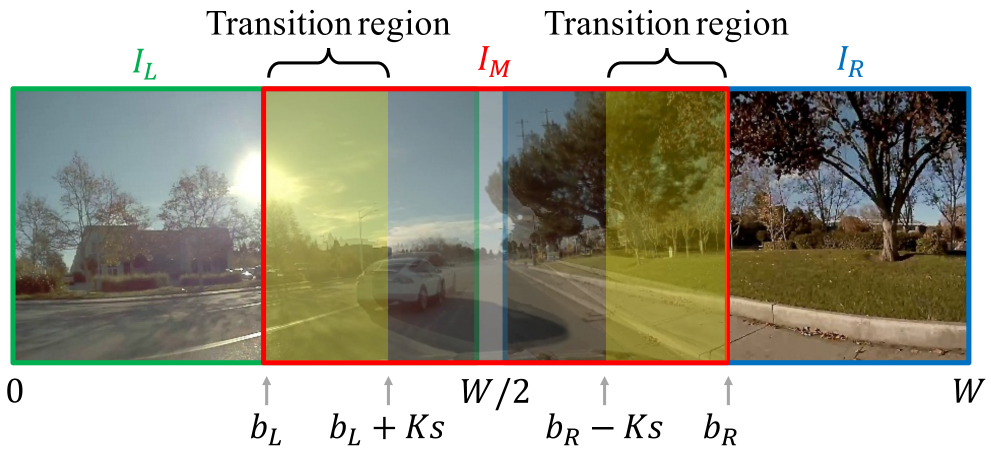


Figure 5.5: **Transition regions for stitching.** Within the transition regions, our pushbroom interpolation method progressively warps and blends $K$ vertical slices from the input views to create a smooth transition. Outside the transition regions, we do not modify the content from the inputs.

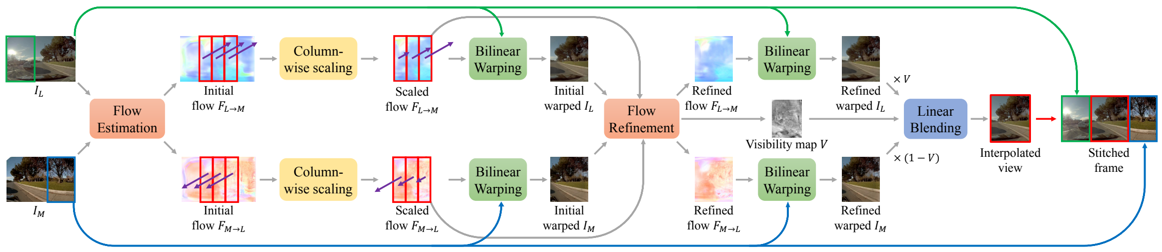


Figure 5.6: **Pushbroom interpolation layer.** A straightforward implementation of the pushbroom interpolation layer requires to generate all the intermediate flows and the intermediate views, which is time-consuming when the number of interpolated views $K$ is large. Therefore, we develop a fast pushbroom interpolation layer by a column-wise scaling on optical flows, which only requires to generate one interpolated image for any given $K$.

step size $s$, the left half part of the output view, $\mathcal{O}_L$, is constructed by

$$\mathcal{O}_L(x) = \begin{cases} I_L(x), & \text{if } 0 \leq x < b_L \\ \hat{I}_{LM}^{(k)}(x), & \text{if } b_L + (k-1) \cdot s \leq x < b_L + k \cdot s \\ I_M(x), & \text{if } b_L + K \cdot s \leq x < \frac{W}{2}, \end{cases} \tag{5.3}$$

where $\hat{I}_{LM}^{(k)}$ is obtained by appropriately fusing $\hat{I}_L^{(k)}$ and $\hat{I}_M^{(k)}$, $k = 1, \cdots, K$ (see Section 5.2.3). By construction, the output image is aligned with $I_L$ at $b_L$, and aligned with $I_M$ at $b_L + K \cdot s$. Within the transition region, the output view gradually changes from $I_L$ to $I_M$ by taking the corresponding columns from the intermediate views. The right half part of the output, $\mathcal{O}_R$, is defined similarly to $\mathcal{O}_L$. We show a stitched view of the proposed pushbroom interpolation approach in Figure 5.4(e).

We note that the finer the interpolation steps, the higher the quality of the stitched results. We find $K = 100$ and $s = 2$, i.e., $100$ slices with 2-pixel pushbroom columns, to offer sufficient good quality. However, creating $K$ images per side to generate a single frame is both computationally expensive and memory intensive. To address this issue, we propose an efficient and lightweight solution to generate $\mathcal{O}_L$ and $\mathcal{O}_R$, which we implement as a network layer, as detailed in Section 5.2.3.

### 5.2.3 Fast Pushbroom Interpolation Layer

Synthesizing the transition regions exactly as described in the previous section is computationally expensive. For each side, it requires scaling the forward and backward optical flow fields $K$ times, and using them to warp the full-resolution images just as many times. This results in $2 \times H \times W \times K$ pixels to warp for each side. However, we only need a slice of $s = 2$ pixels from each of them.

Instead of scaling each flow field in its entirety, we propose to generate a single flow field in which entries corresponding to different slices are scaled differently. For instance, from the flow field from $I_L$ to $I_M$, we generate a new field

$$\hat{F}_{L \to M}(x) = \begin{cases} 0, & \text{if } 0 \leq x < b_L \\ \alpha_k F_{L \to M}(x), & \text{if } b^{(k)} \leq x < b^{(k+1)} \\ F_{L \to M}(x), & \text{if } b_L + K \cdot s \leq x < \frac{W}{2}, \end{cases} \tag{5.4}$$

where $b^{(k)} = b_L + (k-1) \cdot s$ are the boundaries of each slice. We can then warp both images by:

$$\begin{aligned} \hat{I}_L &= \mathcal{W}(I_L, \hat{F}_{L\to M}) && (5.5) \\ \hat{I}_M &= \mathcal{W}(I_M, \hat{F}_{M\to L}), && (5.6) \end{aligned}$$

where $\hat{F}_{M\to L}$ is computed with (5.4) but with $(1-\alpha_k)$ in place of $\alpha_k$. Note that this approach only warps each pixel in the input images *once*.

To deal with the unavoidable artifacts of optical flow estimation, we use a flow refinement network to refine the scaled flows and predict a visibility map for blending. As shown in Figure 5.6, the flow refinement network takes as input the scaled optical flows, $\hat{F}_{L\to M}$ and $\hat{F}_{M\to L}$, and the initial warped images, $\hat{I}_L$, and $\hat{I}_M$, and generates the refined flows, $\tilde{F}_{L\to M}$ and $\tilde{F}_{M\to L}$, and a visibility map $V$. The visibility map can be considered as a quality measure of the flow, which prevents any potential ghosting artifacts due to occlusions. With the refined flows, we warp the input images again to obtain $\tilde{I}_L$ and $\tilde{I}_M$. The final interpolated image is then generated by a linear blending:

$$\tilde{I}_{LM} = V \cdot \tilde{I}_L + (1-V) \cdot \tilde{I}_M. \qquad (5.7)$$

Finally, the output view, $\mathcal{O}_L$, is constructed by replacing all the $\hat{I}_{LM}^{(k)}$ in (5.3) with $\tilde{I}_{LM}$. We generate $\tilde{I}_{RM}$ and construct $\mathcal{O}_R$ by mirroring the process above.

Our fast pushbroom interpolation layer generates the results with similar quality but is about $40\times$ faster than the direct implementation for an output image with a resolution of $1000 \times 600$ pixels.

### 5.2.4 Training Pushbroom Stitching Network

To further refine the interpolated frame from the pushbroom interpolation layer, we introduce an image refinement network as shown in Figure 5.2(b). We name the whole model as the pushbroom stitching network, which is end-to-end trainable. Here we provide details about the training and implementation.

**Training dataset.** One particular challenge for learning to stitch is the lack of training data. To capture the pushbroom panoramic videos, one needs to synchronize and calibrate hundred of cameras and move the cameras simultaneously. In practice, it is impossible to capture the ground-truth videos with real cameras. Therefore, we render realistic synthetic data using the urban driving simulator CARLA [27], which allows us to specify the location and rotation of cameras. We follow the setting of our camera system to specify the input cameras, $C_{iL}$, $C_M$, and $C_{iR}$, in the simulator. In addition, we linearly set 100 cameras between $C_{iL}$ and $C_M$ (also between $C_{iR}$ and $C_M$) to model the pushbroom camera. Therefore, we are able to render "ground-truth" pushbroom interpolated videos by replacing the $I_{LM}^{(k)}$ in (5.3) with the views captured from the intermediate cameras in the simulator. Finally, we synthesize 152 videos with different weathers and routes for training.

We note that, due to the cylindrical projection, some pixels do not contain any information on the ground-truth image. Thus, we also create a mask $M$ to indicate valid pixels for every ground truth image.

**Training loss.** To train our pushbroom interpolation network, we optimize the following loss functions: (1) content loss, (2) perceptual loss, and (3) temporal warping loss.

We define the content loss as the $L_1$ difference between the pixel values of the stitched image $\mathcal{O} \in \mathbb{R}^{H\times W\times 3}$ and the ground-truth stitched image $S \in \mathbb{R}^{H\times W\times 3}$:

$$\mathcal{L}_C = \sum_{x,y} M_{x,y} \cdot \|\mathcal{O}_{x,y} - S_{x,y}\|_1, \tag{5.8}$$

where $\mathcal{O}_{x,y} \in \mathbb{R}^3$ is a vector with RGB pixel values, and $|M|$ is the number of valid pixels.

We measure the perceptual loss from the features of the pre-trained VGG-19 network [99]:

$$\mathcal{L}_P = \sum_{i\in\mathcal{I}} \sum_{x,y} M_{x,y}^{(i)} \cdot \|\phi_{x,y}^{(i)}(\mathcal{O}) - \phi_{x,y}^{(i)}(S)\|_1, \tag{5.9}$$

where $\phi^{(i)}(\cdot)$ denotes the feature activation at the $i$-th layer of the VGG-19 network and $M^{(i)}$ is the valid mask downscaled to the size of the corresponding features. The set $\mathcal{I}$ consists of the `relu1-2`, `relu2-2`, `relu3-3`, and `relu4-3` layers.

To improve the temporal stability, we also optimize the temporal warping loss [63] between frame $t$ and $t'$:

$$\mathcal{L}_T = \sum_{t'\in\Omega_t} \sum_{x,y} M_{x,y} \cdot C_{x,y}^{t\Rightarrow t'} \cdot \|O_{x,y}^{(t)} - \hat{O}_{x,y}^{(t')}\|_1, \tag{5.10}$$

where $\Omega_t$ is the set of neighbor frames at time $t$, $C$ is a confidence map, and $\hat{O}^{(t')} = \mathcal{W}(O^{(t')}, F^{t\Rightarrow t'})$ is the warped frame by the optical flow $F^{t\Rightarrow t'}$. We use the pre-trained PWC-Net [101] to compute the optical flow. Note that the optical flow $F^{t\Rightarrow t'}$ is only used for training, and our model does not require the optical flow at the test phase. The confidence map $C \in [0, 1]$ is computed from the ground-truth frame $S^{(t)}$ and $S^{(t')}$:

$$C^{t\Rightarrow t'} = \exp\left(-\alpha \|S^{(t)} - \hat{S}^{(t')}\|_2^2\right). \tag{5.11}$$

A smaller value of $C$ indicates that the pixel is more likely to be occluded. We note that (5.10) is a weighted $\mathcal{L}_1$ norm, where we set a larger weight for non-occluded pixels and a smaller weight for occluded pixels.

The overall loss function is defined as:

$$\mathcal{L} = \lambda_C \mathcal{L}_C + \lambda_P \mathcal{L}_P + \lambda_T \mathcal{L}_T, \tag{5.12}$$

where $\lambda_C$, $\lambda_P$, and $\lambda_T$ are the weights to balance the content loss, perceptual loss, and temporal loss, respectively. We empirically set $\lambda_C = 1$, $\lambda_P = 0.001$, and $\lambda_T = 1$.

**Implementation details.** We implement our model using PyTorch [85]. As spatial and temporal interpolation are highly-related, we use the SuperSloMo pre-trained for temporal interpolation [54] to initialize the pushbroom interpolation layer. The image refinement network is an encoder-decoder architecture with two strided convolutional layers, five residual blocks [41], and two transposed convolutional layers. All the convolutional and transposed convolutional layers (except the last layers) are followed by the leaky ReLUs [80] with a negative slope of 0.1. As the initial stitched image from the pushbroom interpolation layer is highly similar to the target output image, we add a skip connection between the input and the output of the image refinement network (see Figure 5.2(b)). Therefore, the image refinement network learns the residuals to refine the details of the stitched image.

During the training stage, we sample three consecutive frames from the same video in each forward pass. We only compute the temporal loss for the center frame while computing the content and perceptual losses for all the sampled frames. We set the initial learning rate to $10^{-4}$ and decrease by a factor of 2 for every 20,000 iterations. In total, we optimize our network using the ADAM solver [60] for 100,000 iterations We train our model on an NVIDIA Tesla P100 GPU, which takes 1 day to converge. At the test stage, it takes 0.12 second for our network to stitch a video frame with a resolution of $1000 \times 600$.

## 5.3 Experimental Results

In this section, we first analyze the contribution of each loss function and visualize the intermediate results in the pushbroom interpolation layer. We then compare the proposed model with existing methods and commercial software through a human subject study. Finally, we discuss the limitations and future work of our method.

### 5.3.1 Model Analysis

To quantitatively evaluate the performance of the stitching quality, we use the CARLA simulator to render a test set using a different town map from the training data. We render 10 test videos, where each video has 300 frames.

We measure the PSNR and SSIM between the stitched frames and the ground-truth images for evaluating the image quality. In addition, we measure the temporal stability by computing the temporal warping error:

$$E_{\text{warp}} = \sum_{t=1}^{T-1} \frac{1}{|\tilde{M}^{(t)}|} \sum_{x,y \in \tilde{M}^{(t)}} \|O_{x,y}^{(t)} - \hat{O}_{x,y}^{(t+1)}\|_2^2, \tag{5.13}$$

where $\hat{O}^{(t+1)}$ is the flow-warped frame $O^{(t+1)}$, $\tilde{M}^{(t)}$ is a mask indicating the non-occluded pixels, and $|\tilde{M}^{(t)}|$ is the number of valid pixels in the mask. We use the occlusion detection method in [94] to estimate the mask $\tilde{M}^{(t)}$.

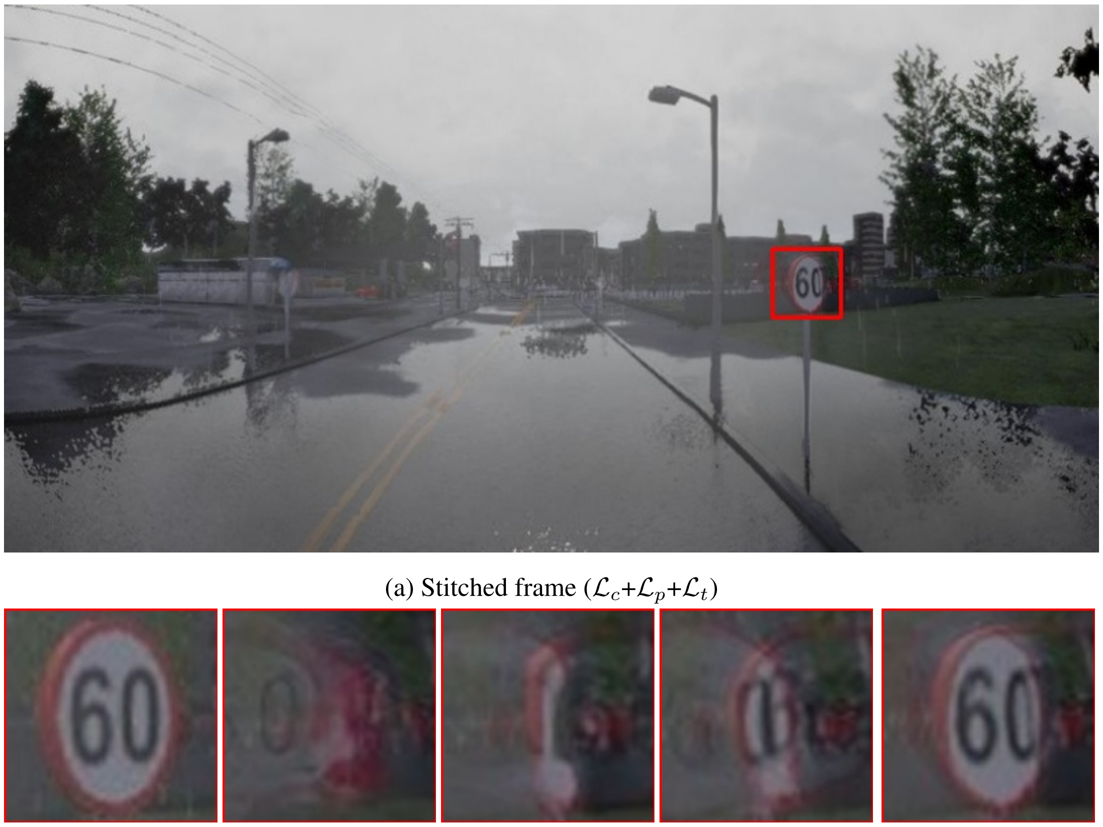
We first evaluate the baseline model, where the pushbroom interpolation layer is initialized from the pre-trained SuperSloMo [54]. The baseline model does not have the image refinement network and is not trained on our synthetic dataset. The baseline model provides a visually plausible stitching result but causes object distortion and temporal flickering due to inaccurate flow estimation. After finetuning the whole model, both the visual quality and temporal stability are significantly improved. As shown in Table 5.1, all the loss functions, $\mathcal{L}_C$, $\mathcal{L}_P$, and $\mathcal{L}_T$, improve the PSNR and SSIM and also reduce the temporal warping error. In Figure 5.7, we show an example where our full model aligns the speed sign well and avoids the ghosting artifacts.

In Figure 5.8 top, we show a stitched frame from the baseline model, where the pole on the right is distorted and almost disappeared. After training, the pole remains intact (Figure 5.8 bottom). We also visualize the optical flows before and after training the model.

Table 5.1: **Ablation study.** The baseline model is initialized from the pre-trained Super-SloMo [54]. After training the model with the content loss $\mathcal{L}_C$, perceptual loss $\mathcal{L}_P$, and the temporal loss $\mathcal{L}_T$, the image quality and temporal stability are significantly improved.

| Training loss | PSNR ↑ | SSIM ↑ | $E_{\text{warp}}$ ↓ |
|---|---|---|---|
| N.A. (baseline) | 27.69 | 0.908 | 13.89 $\times 10^{-4}$ |
| $\mathcal{L}_C$ | 30.95 | 0.925 | 11.57 $\times 10^{-4}$ |
| $\mathcal{L}_C + \mathcal{L}_P$ | 31.09 | 0.926 | 11.49 $\times 10^{-4}$ |
| $\mathcal{L}_C + \mathcal{L}_P + \mathcal{L}_T$ | 31.22 | 0.928 | 11.34 $\times 10^{-4}$ |

After training end-to-end, the flows are more smooth and warp the pole as a whole to avoid distortion.



(a) Stitched frame ($\mathcal{L}_c$+$\mathcal{L}_p$+$\mathcal{L}_t$)

(b) GT (c) Baseline (d) $\mathcal{L}_c$ (e) $\mathcal{L}_c$+$\mathcal{L}_p$ (f) $\mathcal{L}_c$+$\mathcal{L}_p$+$\mathcal{L}_t$

Figure 5.7: **Example of the synthetic video.** After training on the synthetic data, our model aligns the content well and reduce the ghosting artifacts.

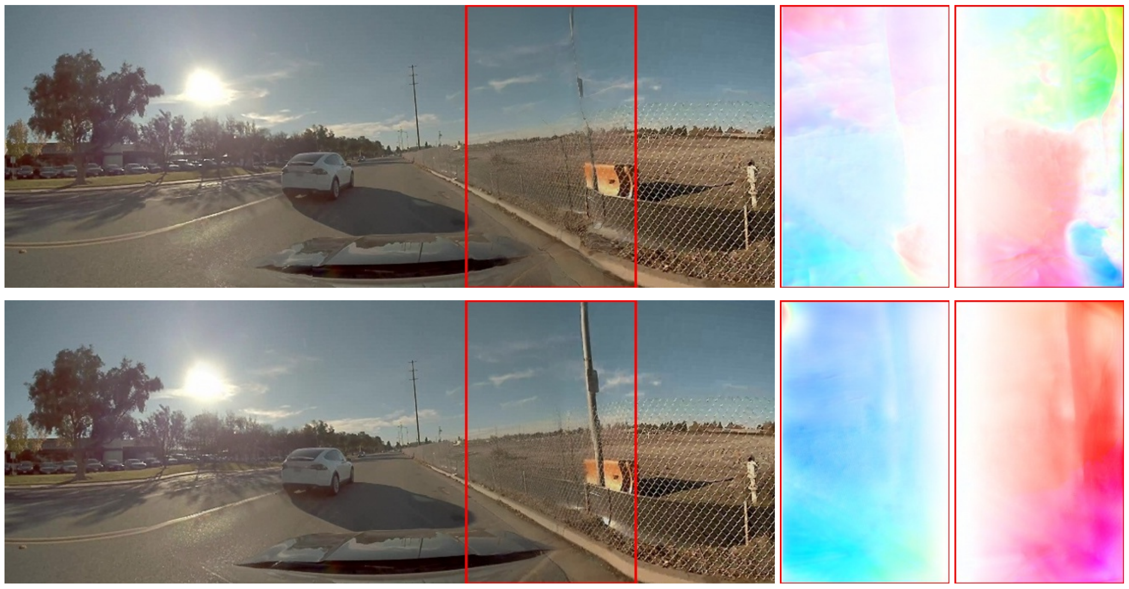


Figure 5.8: **Visualization of the pushbroom interpolation layer.** We show (a) the stitched frame, (b) forward flow, and (c) backward flows from the pushbroom interpolation layer before (top) and after (bottom) training the proposed model. The finetuned model generates smooth flow fields to warp the input views while preserving the content (e.g., the pole on the right) well.

Table 5.2: **Human subject study.** We conduct pairwise comparisons on 20 real video. We show the percentage that users prefer our method against other approaches and the percentage of reasons when users select our method.

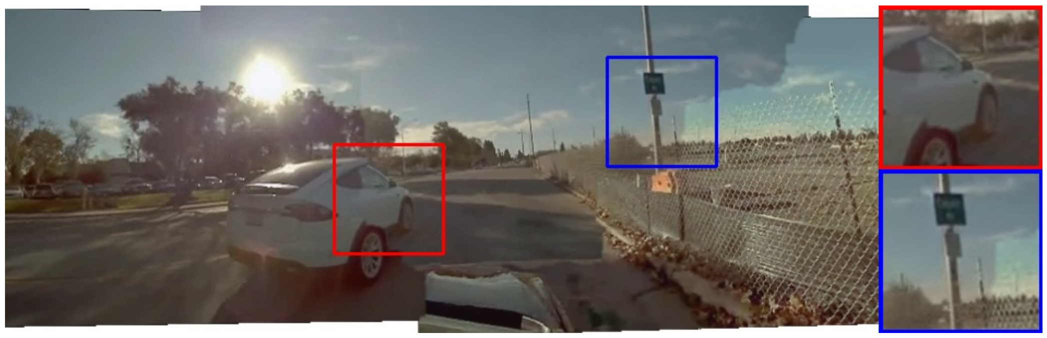
| Ours vs. | Preference | Fewer broken objects | Less ghosting | Similar results |
|---|---|---|---|---|
| AutoPano [4] | 90.74% | 85.71% | 20.41% | 10.20% |
| VRWorks [83] | 97.22% | 80.00% | 49.52% | 1.90% |
| STCPW [55] | 98.15% | 87.74% | 38.68% | 0% |
| Overall | 95.37% | 84.74% | 36.57% | 3.88% |

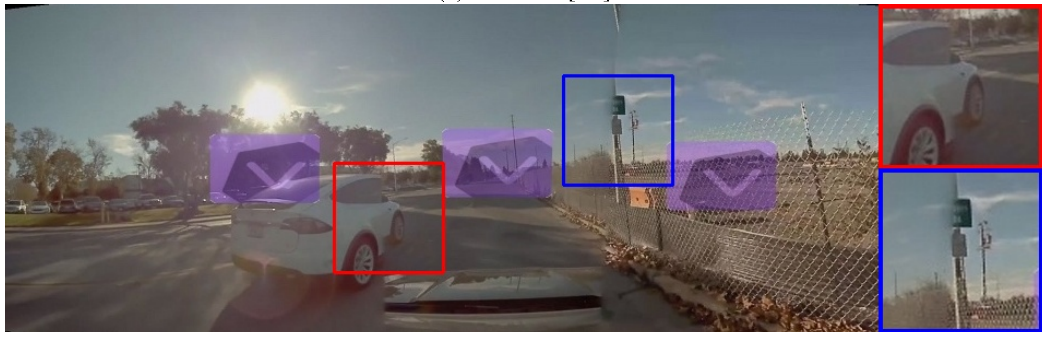
### 5.3.2 Comparisons with Existing Methods

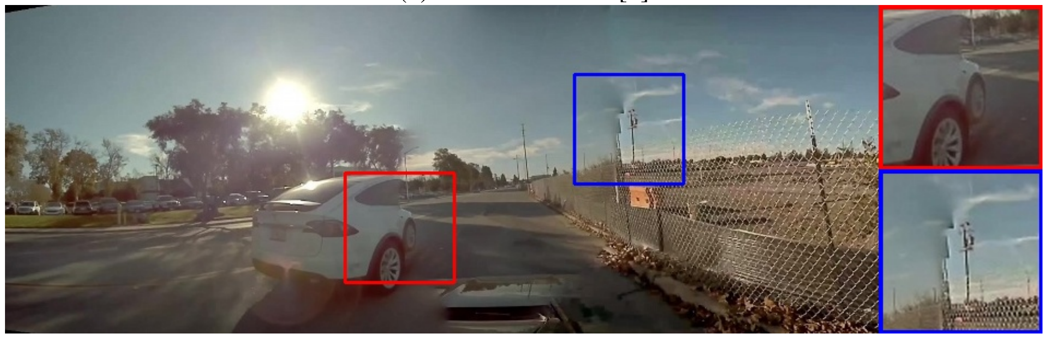
We compare the proposed method with a commercial software, AutoPanoVideo [4], and existing video stitching algorithms, STCPW [55] and NVIDIA VRWorks [83]. We show two stitched frames from real videos in Figure 5.9 and 5.10, where the proposed approach generally achieves better alignment quality with fewer broken objects and ghosting artifacts. A video comparison is available in https://www.dropbox.com/s/fa8ty6txplut45u/video_stitching_demo.mp4?dl=0.

As different methods may use different projection models, it is difficult to quantitatively evaluate the performance of video stitching algorithms. Instead, we conduct a human subject study through pairwise comparisons. Specifically, we ask the participants to choose a stitched video with fewer artifacts from a paired of videos. We evaluate a total of 20 real videos. We ask each participant to compare 12 pairs of videos. In total, we collect the results from 54 participants.

In Table 5.2, we show that our results are preferred by about $95\%$ of users, which demonstrates the effectiveness of the proposed method on generating high-quality stitching results. In addition, we ask participants to provide the reasons that they prefer the selected video from the following options: (1) the video has fewer broken lines or objects, (2) the video has less ghosting artifacts, and (3) the two videos are similar. Overall, our results are preferred due to a better alignment and fewer broken objects. Moreover, only $4\%$ of users feel that our result is comparable to others, which indicates that users generally have a clear judgment when comparing our method with other approaches.


(a) STCPW [55]


(b) AutoPano Video [4]


(c) NVIDIA VRWorks [83]

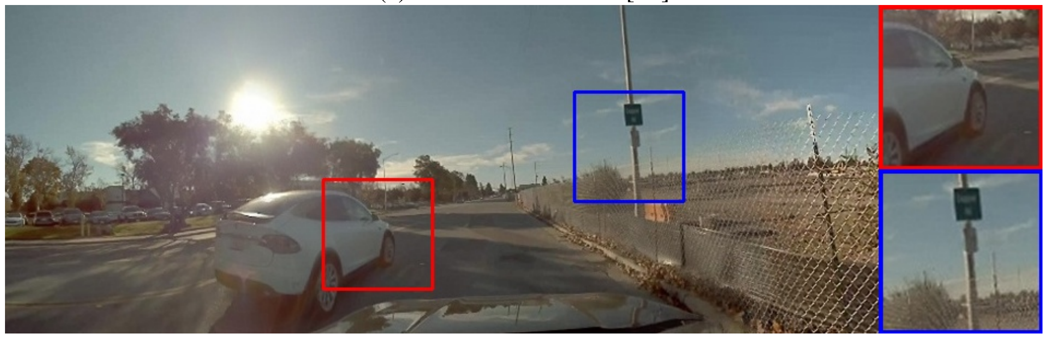
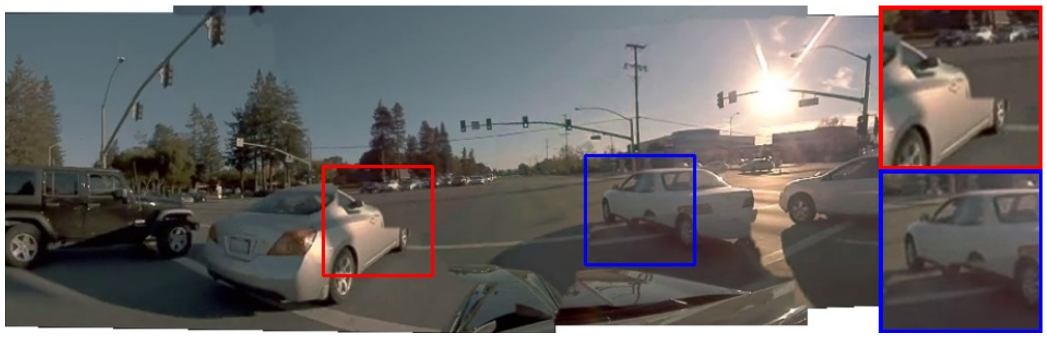
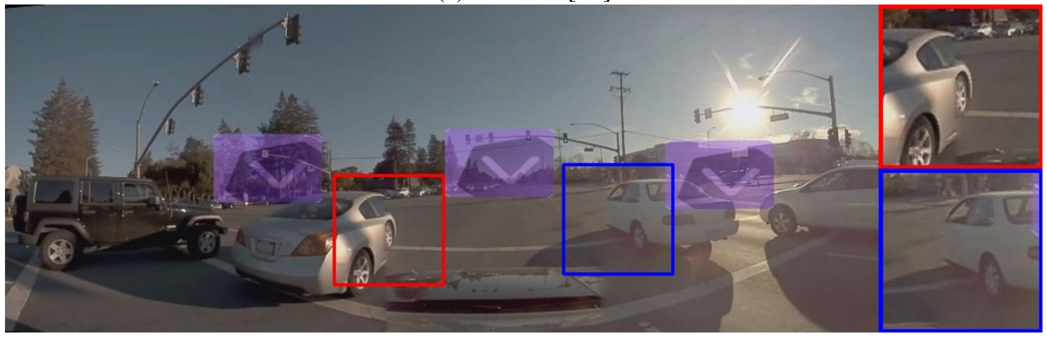
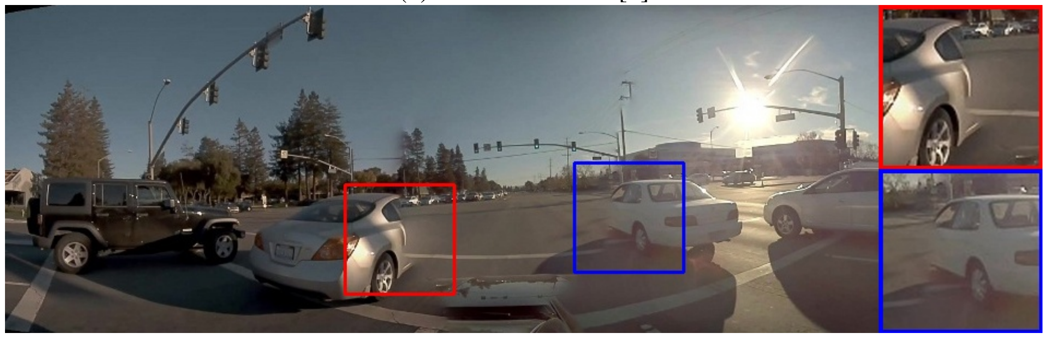

(d) Ours

Figure 5.9: **Comparison with existing video stitching methods.** The proposed method achieves better alignment quality and thus preserves the shape of objects well and avoids ghosting artifacts.


(a) STCPW [55]


(b) AutoPano Video [4]


(c) NVIDIA VRWorks [83]

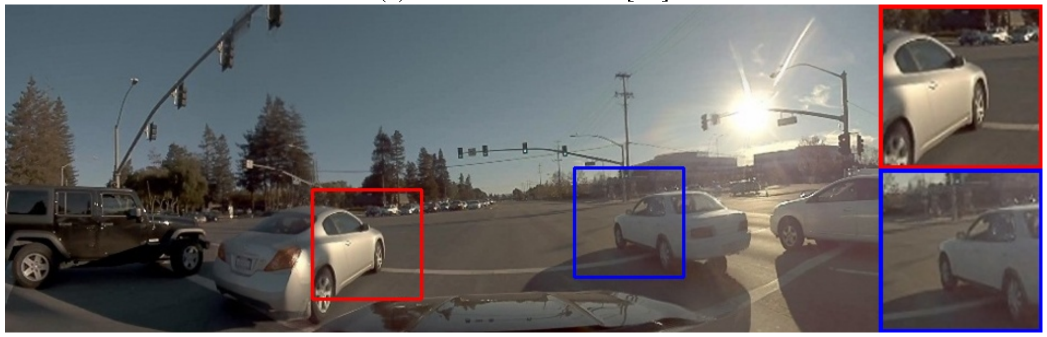

(d) Ours

Figure 5.10: **Comparison with existing video stitching methods.** The proposed method achieves better alignment quality and thus preserves the shape of objects well and avoids ghosting artifacts.

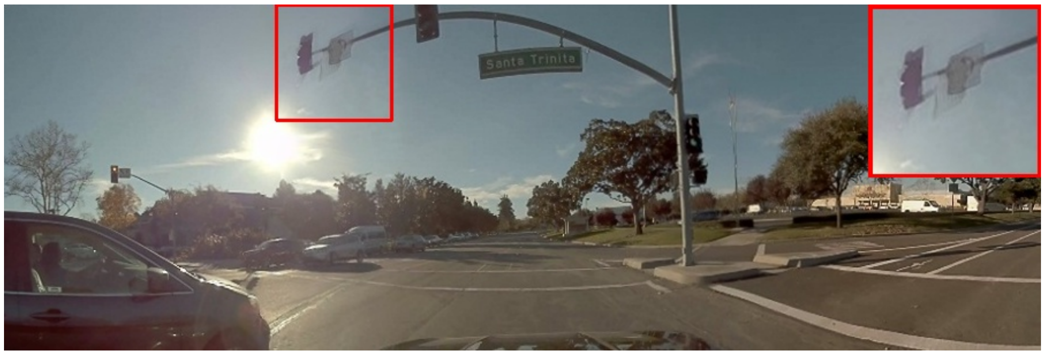


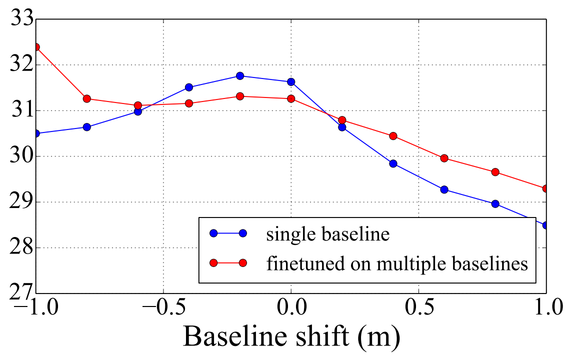
Figure 5.11: **Failure case.** Our approach might produce mis-alignment or ghosting artifacts when the flow estimation is not accurate enough.

### 5.3.3 Limitations and Discussion

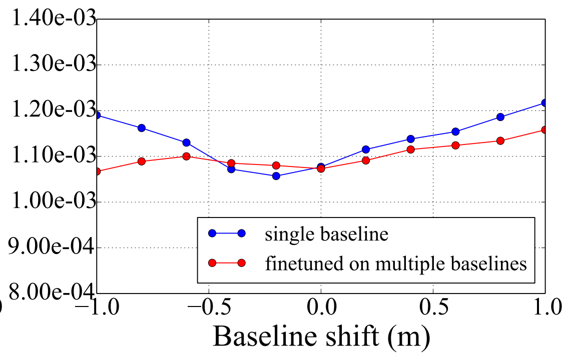
While the proposed method performs favorably against existing video stitching approaches, there are still a few limitations.

First, the proposed pushbroom interpolation relies on optical flow for warping, which inherits the common challenges of optical flow estimation. As shown in Figure 5.11, when the flow is not estimated well, e.g., thin structure or objects, our method may generate ghosting or temporal jittering.

Second, our method requires the camera calibration to obtain camera parameters for pre-warping the input views. Directly applying the pushbroom interpolation on the input videos may not produce acceptable results due to the huge perspective difference between the input views. Further, although trained for a specific configuration, our model is robust to a certain amount of deviations from the expected settings. To demonstrate the robustness of our model, we render the same test video by changing the camera baseline, i.e., horizontally shifting the side cameras inward or outward from the position used for training. The blue curves in Figure 5.12(a) and (b) offer an insight on the analysis of the impact of different baselines. Even when moving the side cameras inwards by up to 0.8m ($62.5\%$ of the original baseline), the PSNR drops by less than 1dB. While moving the side cameras outwards decreases the size of overlapped region, the PSNR drops less than 2dB when the deviation is up to 0.4m. On the other hand, the temporal warping error remains small and the video is still stable during playback. We show visual results from the default baseline and an extreme baseline shift (+1.0m) in Figure 5.13(a) and (b). Note that despite the very



(a) PSNR

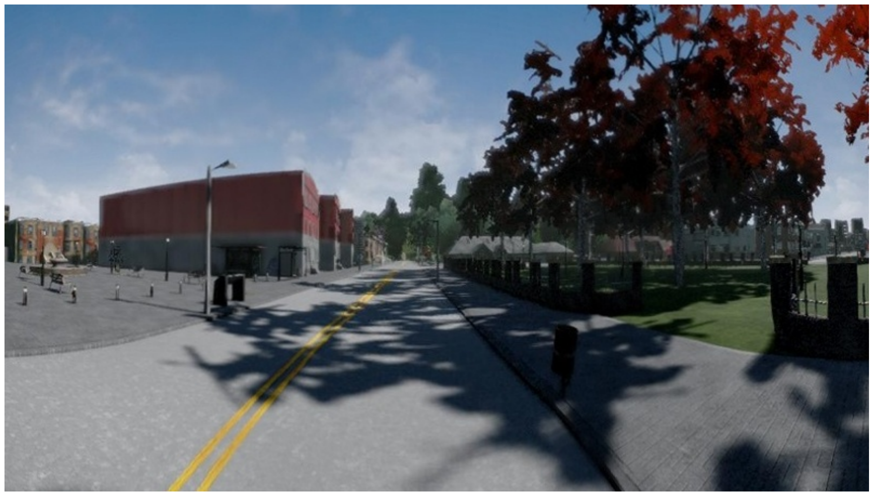


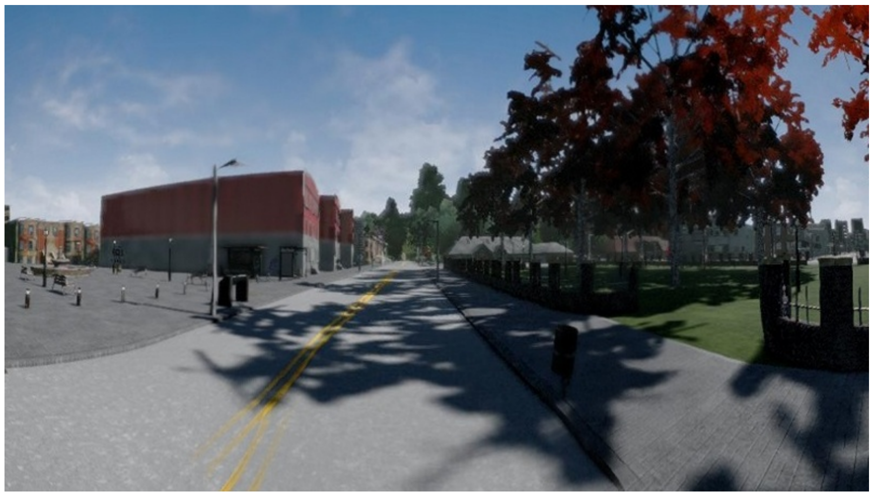
(b) Temporal warping error

Figure 5.12: **Analysis on different camera baselines.**

large change in baseline, and even when the overlap is reduced by moving the cameras outwards, the quality remains similar—see the lamp post, which is in a transition region. However, minor artifacts can be seen in close-by regions, e.g., double yellow lines.

We further fine-tune our model (trained on a single baseline) with data that mixes multiple camera baselines. The fine-tuned model further improves the performance (red curves in Figure 5.12) for a range of baseline shifts and improves the visual quality as shown in Figure 5.13(c).


(a) 1.28m (default)


(b) 2.28m (+1.0m)

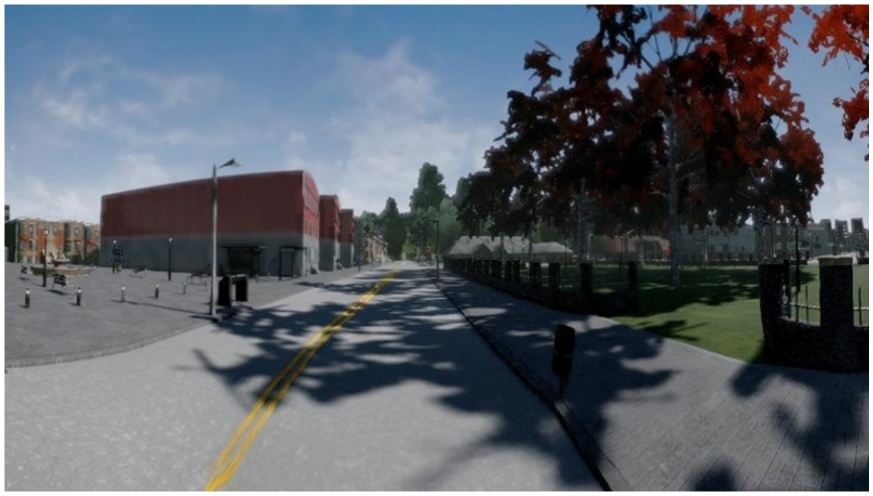

(c) 2.28m (+1.0m) fine-tuned

Figure 5.13: **Results on different camera baselines.**

## 5.4 Conclusions

In this work, we present an efficient algorithm to stitch videos with deep CNNs. We propose a pushbroom interpolation layer to gradually align input views through the spatial interpolation. Our model effectively aligns and stitches the content from different views while preserving the object shape and avoiding ghosting artifacts. By training the proposed model on a synthetic dataset, we significantly improve the stitching quality and temporal stability. A human subject study demonstrates that the proposed method performs favorably against existing video stitching algorithms and commercial software. To the best of our knowledge, our approach is the first learning-based algorithm, which sheds light on future research for developing CNN-based solutions for image and video stitching.

# Chapter 6

# Conclusion and Future Work

## 6.1 Summary

This thesis studies the problems of visual enhancement. We have made several key contributions toward effective and efficient visual enhancement in three aspects.

**Spatial enhancement.** In Chapter 3, we propose the deep Laplacian Pyramid Super-Resolution Network for fast and accurate image super-resolution. The proposed network progressively reconstructs the sub-band residuals of high-resolution images at multiple pyramid levels. In contrast to existing methods that involve the bicubic interpolation for pre-processing (which results in large feature maps), the proposed method directly extracts features from the low-resolution input space and thereby entails low computational loads. We train the proposed network with the deep supervision, multi-scale training, and robust Charbonnier loss functions and achieve high-quality image reconstruction. Furthermore, we utilize the recursive layers to share parameters across as well as within pyramid levels, and thus drastically reduce the number of parameters. The proposed model is more effective, efficient, and compact than the state-of-the-art CNN-based super-resolution methods.

**Temporal enhancement.** In Chapter 4, we present an efficient approach based on a deep recurrent network for enforcing temporal consistency in a video. We train the proposed network by minimizing both short-term and long-term temporal losses as well as a perceptual loss to strike a balance between temporal coherence and perceptual similarity with the processed frames. At test time, our model does not require computing optical flow and thus achieves real-time speed even for high-resolution videos. Our model is agnostic to specific image-based algorithms applied on the original videos. Furthermore, a *single* model can handle *multiple* and *unseen* tasks, including but not limited to artistic style transfer, enhancement, colorization, image-to-image translation and intrinsic image decomposition.

**Spatial-temporal enhancement.** In Chapter 5, we propose a pushbroom stitching network to stitch multiple wide-baseline videos into a single panoramic video. To tolerant to strong parallax and achieve temporally stable results, we cast the video stitching as a spatial interpolation problem, inspired by the pushbroom cameras. We introduce a fast pushbroom interpolation layer to learn a dense flow field to smoothly align the input videos. Our approach generates more visually pleasing results than existing approaches and has immediate applications in many areas such as virtual reality, immersive telepresence, autonomous driving, and video surveillance.

## 6.2 Future Work

Along the topic of visual enhancement, there are three interesting directions for future investigation.

### 6.2.1 Practical Applications of Low-Level Vision

Most existing CNN-based SR models focus on simulated data (e.g., bicubic downsampled images without noise), which is relatively simple compared to real-world data. It is of great importance to improve existing methods to handle real-world low-resolution data, which may be generated from unknown downsampling kernels or contain compression artifacts and unknown sensor noise. On the other hand, with the advancement of camera sensors, cameras on mobile phones can already capture photos with a 2K or 4K pixel resolution. Applying existing CNN-based algorithms on such an ultra-resolution input inevitably raises the problem of memory usage and efficiency. The same issue is not only applied to image super-resolution but also other low-level vision problems such as image deblurring and video frame interpolation. Instead of learning a deep and complex model, one may need to adopt the model compression or knowledge distillation strategies to learn compact models.

In addition, image super-resolution is highly related to the image compression technique. Some existing approaches [20, 70] focus on reducing the compression artifacts while performing image super-resolution. It will be interesting to explore a joint optimization of image downsampling, compression, and super-resolution to improve the compression rate and reduce the loss of quality.

### 6.2.2 Semi-Supervised Learning and Domain Adaptation

Some fundamental computer vision problems, such as depth and optical flow estimation, typically lack ground-truth data. Existing methods rely on synthetic data with dense ground-truth for training. However, the models still need to be fine-tuned on different datasets to achieve better performance. One of my previous work addresses the semi-supervised learning of optical flow using both the synthetic (with ground-truth) and real (without ground-truth) data [64]. Nevertheless, the synthetic and real data contain a domain gap, which may impede the model to learn from both two domains. Therefore, it is important to explore the domain adaptation strategy [107] to effectively utilize data from different domains and learn a more generalized model.

On the other hand, the flow and depth have a geometric relationship and consistency that could be exploited for joint learning. Existing methods [91, 133] mainly focus on unsupervised learning with unlabeled videos. It will also be an interesting direction to explore the semi-supervised learning for such a joint learning problem.

### 6.2.3 Learning-Based Computational Photography

Several computational photography problems (e.g., image demosaicing [102] and HDR reconstruction [81, 29]) could be cast as a learning problem if the input and output can be well defined. However, learning a single end-to-end model does not always lead to better performance. For instance, in the context of single-image HDR reconstruction, one needs to restore 32-bit pixel information from the input 8-bit image, which is extremely challenging. Therefore, it is important to exploit domain knowledge to mitigate the training difficulty We can decompose the problem into several sub-tasks by modeling the LDR image formation pipeline as dynamic range clipping, sensor noise generation, non-linear mapping, and quantization. We then learn CNNs to explicitly model the *inverse* function of each step in the camera pipeline. The whole model can be further fine-tuned end-to-end to reduce the error accumulation. By integrating the conventional domain knowledge into deep CNNs, we can design better architectures and loss functions to regularize the network training and achieve better performance.

# Bibliography

[1] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, and R. Szeliski. Photographing long scenes with multi-viewpoint panoramas. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 25(3):853–861, 2006.

[2] R. Anderson, D. Gallup, J. T. Barron, J. Kontkanen, N. Snavely, C. Hernández, S. Agarwal, and S. M. Seitz. Jump: virtual reality video. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 35(6):198:1–198:13, 2016.

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[4] AutoPano Video. `http://www.kolor.com/`.

[5] T. O. Aydin, N. Stefanoski, S. Croci, M. Gross, and A. Smolic. Temporally coherent local tone mapping of HDR video. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 33(6):196:1–196:13, 2014.

[6] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 28(3):24:1–24:11, 2009.

[7] S. Bell, K. Bala, and N. Snavely. Intrinsic images in the wild. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 33(4):159:1–159:12, 2014.

[8] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *British Machine Vision Conference*, 2012.

[9] N. Bonneel, J. Rabin, G. Peyré, and H. Pfister. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015.

[10] N. Bonneel, K. Sunkavalli, S. Paris, and H. Pfister. Example-based video color grading. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 32(4):39:1–39:12, 2013.

[11] N. Bonneel, J. Tompkin, K. Sunkavalli, D. Sun, S. Paris, and H. Pfister. Blind video temporal consistency. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 34(6):196:1–196:9, 2015.

[12] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs the method of paired comparisons. *Biometrika*, 39(3-4):324–345, 1952.

[13] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007.

[14] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[15] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540, 1983.

[16] P. J. Burt and E. H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 2(4):217–236, 1983.

[17] C.-H. Chang, Y. Sato, and Y.-Y. Chuang. Shape-preserving half-projective warps for image stitching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[18] H. Chang, D.-Y. Yeung, and Y. Xiong. Super-resolution through neighbor embedding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[19] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua. Coherent online video style transfer. In *IEEE International Conference on Computer Vision*, 2017.

[20] H. Chen, X. He, C. Ren, L. Qing, and Q. Teng. CISRDCNN: super-resolution of compressed images using deep convolutional neural networks. *Arxiv*, 2017.

[21] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. In *IEEE International Conference on Computer Vision*, 2017.

[22] Y.-S. Chen and Y.-Y. Chuang. Natural image stitching with the global similarity prior. In *European Conference on Computer Vision*, 2016.

[23] E. L. Denton, S. Chintala, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *Neural Information Processing Systems*, 2015.

[24] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2015.

[25] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *European Conference on Computer Vision*, 2016.

[26] X. Dong, B. Bonev, Y. Zhu, and A. L. Yuille. Region-based temporally consistent video post-processing. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[27] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Conference on Robot Learning*, 2017.

[28] A. Eden, M. Uyttendaele, and R. Szeliski. Seamless image stitching of scenes with large motions and exposure differences. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[29] G. Eilertsen, J. Kronander, G. Denes, R. K. Mantiuk, and J. Unger. HDR image reconstruction from a single exposure using deep CNNs. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 36(6):178:1–178:15, 2017.

[30] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision*, 2015.

[31] G. Freedman and R. Fattal. Image and video upscaling from local self-examples. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 30(2):12, 2011.

[32] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, 2002.

[33] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, and F. Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 36(4):118:1–118:12, 2017.

[34] G. Ghiasi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *European Conference on Computer Vision*, 2016.

[35] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *IEEE International Conference on Computer Vision*, 2009.

[36] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Neural Information Processing Systems*, 2014.

[37] A. Gupta, J. Johnson, A. Alahi, and L. Fei-Fei. Characterizing and improving stability in neural style transfer. In *IEEE International Conference on Computer Vision*, 2017.

[38] R. Gupta and R. I. Hartley. Linear pushbroom cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):963–975, 1997.

[39] K. He, J. Sun, and X. Tang. Single image haze removal using dark channel prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[40] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision*, 2015.

[41] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[42] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 2016.

[43] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *22nd Annual Conference on Computer Graphics and Interactive Techniques*, 1995.

[44] E. Hsu, T. Mertens, S. Paris, S. Avidan, and F. Durand. Light mixture estimation for spatially varying white balance. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 27(3):70:1–70:7, 2008.

[45] H. Huang, H. Wang, W. Luo, L. Ma, W. Jiang, X. Zhu, Z. Li, and W. Liu. Real-time neural style transfer for videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[46] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf. Temporally coherent completion of dynamic video. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 35(6), 2016.

[47] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[48] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *IEEE International Conference on Computer Vision*, 2017.

[49] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $< 0.5$ mb model size. *Arxiv*, 2016.

[50] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 35(4):110:1–110:11, 2016.

[51] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[52] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[53] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Neural Information Processing Systems*, 2015.

[54] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz. Super SloMo: High quality estimation of multiple intermediate frames for video interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[55] W. Jiang and J. Gu. Video stitching with spatial-temporal content-preserving warping. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015.

[56] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016.

[57] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[58] J. Kim, J. K. Lee, and K. M. Lee. Deeply-recursive convolutional network for image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[59] K. I. Kim and Y. Kwon. Single-image super-resolution using sparse regression and natural image prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1127–1133, 2010.

[60] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[61] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[62] W.-S. Lai, J.-B. Huang, Z. Hu, N. Ahuja, and M.-H. Yang. A comparative study for single image blind deblurring. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[63] W.-S. Lai, J.-B. Huang, O. Wang, E. Shechtman, E. Yumer, and M.-H. Yang. Learning blind video temporal consistency. In *European Conference on Computer Vision*, 2018.

[64] W.-S. Lai, J.-B. Huang, and M.-H. Yang. Semi-supervised learning for optical flow with generative adversarial networks. In *Neural Information Processing Systems*, 2017.

[65] M. Lang, O. Wang, T. O. Aydin, A. Smolic, and M. H. Gross. Practical temporal consistency for image-based graphics applications. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 31(4):34:1–34:8, 2012.

[66] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[67] C. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets, 2015. In *International Conference on Artificial Intelligence and Statistics*, 2015.

[68] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. K. S. Singh, and M.-H. Yang. Diverse image-to-image translation via disentangled representation. In *European Conference on Computer Vision*, 2018.

[69] J. Lee, B. Kim, K. Kim, Y. Kim, and J. Noh. Rich360: Optimized spherical representation from structured panoramic camera arrays. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 35(4):63:1–63:11, 2016.

[70] O. Lee, J. Lee, D. Lee, and J. Kim. Joint super-resolution and compression artifact reduction based on dual-learning. In *VCIP*, 2016.

[71] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 23(3):689–694, 2004.

[72] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. In *Neural Information Processing Systems*, 2017.

[73] Y. Li, J.-B. Huang, A. Narendra, and M.-H. Yang. Deep joint image filtering. In *European Conference on Computer Vision*, 2016.

[74] Y. Li, S. Liu, J. Yang, and M.-H. Yang. Generative face completion. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[75] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017.

[76] C.-C. Lin, S. U. Pankanti, K. Natesan Ramamurthy, and A. Y. Aravkin. Adaptive as-natural-as-possible image stitching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[77] K. Lin, S. Liu, L.-F. Cheong, and B. Zeng. Seamless video stitching from hand-held camera inputs. *Computer Graphics Forum. (Proceedings of EuroGraphics)*, 35(2):479–487, 2016.

[78] W.-Y. Lin, S. Liu, Y. Matsushita, T.-T. Ng, and L.-F. Cheong. Smoothly varying affine stitching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[79] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[80] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, 2013.

[81] D. Marnerides, T. Bashford-Rogers, J. Hatchett, and K. Debattista. ExpandNet: A deep convolutional neural network for high dynamic range expansion from low dynamic range content. *Computer Graphics Forum. (Proceedings of EuroGraphics)*, 37(2):37–49, 2018.

[82] Y. Matsui, K. Ito, Y. Aramaki, T. Yamasaki, and K. Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multimedia Tools and Applications*, 76(20):21811–21838, 2017.

[83] Nvidia VRWorks. `https://developer.nvidia.com/vrworks/vrworks-360video`.

[84] S. Paris, S. W. Hasinoff, and J. Kautz. Local laplacian filters: Edge-aware image processing with a laplacian pyramid. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 30(4):68, 2011.

[85] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[86] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[87] S. Peleg and J. Herman. Panoramic mosaics with videobrush. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[88] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[89] F. Perazzi, A. Sorkine-Hornung, H. Zimmer, P. Kaufmann, O. Wang, S. Watson, and M. Gross. Panoramic video from unstructured camera arrays. In *Computer Graphics Forum*, 2015.

[90] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.

[91] A. Ranjan, V. Jampani, K. Kim, D. Sun, J. Wulff, and M. J. Black. Adversarial collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. *Arxiv*, 2018.

[92] A. Rav-Acha, Y. Shor, and S. Peleg. Mosaicing with parallax using time warping. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2004.

[93] M. Rubinstein, D. Gutierrez, O. Sorkine, and A. Shamir. A comparative study of image retargeting. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 29(6):160:1–160:10, 2010.

[94] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. In *German Conference on Pattern Recognition*, 2016.

[95] S. Schulter, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[96] S. M. Seitz and J. Kim. Multiperspective imaging. *IEEE Computer Graphics and Applications*, 2003.

[97] H. R. Sheikh, A. C. Bovik, and G. De Veciana. An information fidelity criterion for image quality assessment using natural scene statistics. *IEEE Transactions on Image Processing*, 14(12):2117–2128, 2005.

[98] W. Shi, J. Caballero, F. Huszar, J. Totz, A. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[99] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[100] A. Singh and N. Ahuja. Super-resolution using sub-band self-similarity. In *Asia Conference on Computer Vision*, 2014.

[101] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[102] N. Syu, Y. Chen, and Y. Chuang. Learning deep convolutional networks for demosaicing. *Arxiv*, 2018.

[103] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[104] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, L. Zhang, B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. NTIRE 2017 challenge on single image super-resolution: Methods and results. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017.

[105] R. Timofte, V. De Smet, and L. Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *Asia Conference on Computer Vision*, 2014.

[106] R. Timofte, V. Smet, and L. Gool. Anchored neighborhood regression for fast example-based super-resolution. In *IEEE International Conference on Computer Vision*, 2013.

[107] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker. Learning to adapt structured output space for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[108] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[109] A. Vedaldi and K. Lenc. MatConvNet: Convolutional neural networks for matlab. In *ACM International conference on Multimedia*, 2015.

[110] VideoStitch Studio. `https://www.orah.co/news/videostitch-studio`.

[111] Videvo. `https://www.videvo.net/`.

[112] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[113] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *IEEE International Conference on Computer Vision*, 2015.

[114] Y. Wexler and D. Simakov. Space-time scene manifolds. In *IEEE International Conference on Computer Vision*, 2005.

[115] S. Xie and Z. Tu. Holistically-nested edge detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[116] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Neural Information Processing Systems*, 2015.

[117] L. Xu, J. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *International Conference on Machine Learning*, 2015.

[118] Z. Yan, H. Zhang, B. Wang, S. Paris, and Y. Yu. Automatic photo adjustment using deep neural networks. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 35(2):11:1–11:15, 2016.

[119] C.-Y. Yang, J.-B. Huang, and M.-H. Yang. Exploiting self-similarities for single frame super-resolution. In *Asia Conference on Computer Vision*, 2010.

[120] C.-Y. Yang, C. Ma, and M.-H. Yang. Single-image super-resolution: A benchmark. In *European Conference on Computer Vision*, 2014.

[121] C.-Y. Yang and M.-H. Yang. Fast direct super-resolution by simple functions. In *IEEE International Conference on Computer Vision*, 2013.

[122] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution as sparse representation of raw image patches. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[123] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010.

[124] C.-H. Yao, C.-Y. Chang, and S.-Y. Chien. Occlusion-aware video temporal consistency. In *ACM International conference on Multimedia*, 2017.

[125] G. Ye, E. Garces, Y. Liu, Q. Dai, and D. Gutierrez. Intrinsic video and applications. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 33(4):80:1–80:11, 2014.

[126] X. Yu and F. Porikli. Ultra-resolving face images by discriminative generative networks. In *European Conference on Computer Vision*, 2016.

[127] J. Zaragoza, T.-J. Chin, M. S. Brown, and D. Suter. As-projective-as-possible image stitching with moving dlt. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[128] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *International Conference on Curves and Surfaces*, 2010.

[129] F. Zhang and F. Liu. Parallax-tolerant image stitching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[130] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision*, 2016.

[131] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[132] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision*, 2017.

[133] Y. Zou, Z. Luo, and J.-B. Huang. Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *European Conference on Computer Vision*, 2018.