Edge-preserving Image Upsampling and Image Warping

by

Jiayu Li, **B.Sc., B.Eng.**

A thesis submitted to the
Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Computer Science

Ottawa-Carleton Institute for Computer Science
The School of Computer Science
Carleton University
Ottawa, Ontario
August, 2015

The undersigned hereby recommends to the
Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Edge-preserving Image Upsampling and Image Warping

submitted by **Jiayu Li**, **B.Sc., B.Eng.**

in partial fulfillment of the requirements for the degree of

Master of Computer Science

Professor Davie Mould, Thesis Supervisor

Professor Oliver van Kaick, School of Computer Science

Professor Robert Laganiere,
School of Electrical Engineering and Computer Science

Professor Michel Barbeau, Chair,
School of Computer Science

Ottawa-Carleton Institute for Computer Science
The School of Computer Science
Carleton University
August, 2015

Abstract

The goal of this thesis is to produce sharp upsampled images with some fine details. We reviewed the previous literature in image upsampling, artistic stylization, and painterly rendering. We present cumulative range geodesic filter and bilateral roundup filter, with the former excels at edge sharpening and the latter finds a better balance between edge sharpening and fine-detail preservation. Moreover, we propose a pixel migration idea that relocates pixels in the image plane. We implemented pixel migration using a mass-spring physics simulation, where the pixel movement is enforced by varying spring rest lengths and spring coefficients. In the context of image upsampling, we assign short rest lengths to pixels along edges, thus the gaps between those pixels are closed, and edge sharpness is recovered. Applying pixel migration on the original image plane, we propose a two-stage approach to painterly rendering of photographs, where the image plane is first warped to produce a distorted or caricatured effect and then the resulting image is rendered with a painterly effect. This work on image warping was published in Expressive, 2015.

Acknowledgments

I would like to show my greatest gratitude to my mentor, supervisor and friend, Dr. David Mould. I have been extremely blessed to have your guidance, your encouragement, your advice and your patience. You are so intelligent, resourceful, knowledgeable and kind. I cannot hope to have anyone better to be my supervisor. God bless you!

I would like to thank the thesis committee for reviewing my work and giving me those valuable suggestions. I must also express my appreciation to my colleagues and friends in GIGL lab and the School of Computer Science. Their brilliance has encouraged me to aim higher and work harder.

I would like to thank GRAND, NSERC, Carleton University and GIGL lab for their persistent financial support.

We used many images from Flickr under a Creative Commons license. Thanks to the numerous photographers who provided the material: spablab (hood), Doug Kerr (cathedral), Hafiz Issadeen (garlic), Pedro Ribeiro Simes (old lady), Brian Gratwicke(penguin), Anne Worner (old man), Wayne Dumbleton (secretarybird), Yiannis Chatzitheodorou (street), Stefan Schmidt (books), InAweofGodsCreation (lighthouse), MythicSeabass (boxer), Ana Raquel S. Hernandes (towers), sean dreilinger (crying), paVan(flamingo). Thanks to Gail Carmichael from GIGL lab for providing various other photographs.

Table of Contents

Abstract iii

Acknowledgments iv

Table of Contents v

List of Tables vii

List of Figures viii

1 Introduction 1

2 Previous Work 4

2.1 Introduction 4

2.2 Image Upsampling 5

2.3 Artistic Stylization 9

2.4 Image Warping for Painterly Effect 13

3 Filtering-based Image Upsampling 18

3.1 Introduction 18

3.2 Algorithms 21

3.2.1 Cumulative Range Geodesic Filtering 21

3.2.2 Bilateral Roundup Filtering 23

3.2.3 Variations of Bilateral Roundup Filtering 25

3.3 Notes towards Mask Data Processing 32

3.4 Results 38

3.5 Conclusion 44

4 Pixel Migration **49**
4.1 Introduction 49
4.2 Algorithm 52
4.2.1 Pipeline 52
4.2.2 Edge Detection 53
4.2.3 Mass-spring System 54
4.3 Results 59
4.3.1 Pixel Migration results 59
4.3.2 Comparison between filtering-based approach and pixel migration 61
4.4 Conclusion 70

5 Image Warping **72**
5.1 Introduction 72
5.2 Algorithm 74
5.3 Results 79
5.4 Conclusion 88

6 Conclusion **92**

List of References **95**

List of Tables

4.1 Spring rest length and strength coefficient assignment. 56
4.2 Comparison between filtering-based approach and pixel migration. . . 69

List of Figures

3.1 Input images used in this chapter: barktree, picnic table, stranger, roughwater, sedona, and dragonfly. 18

3.2 Three basic interpolation methods. Left: Nearest neighbour interpolation, which introduces clustering pixel blocks. Middle: Bilinear interpolation. Right: bicubic interpolation. Both bilinear and bicubic interpolation smooth out the sharp edges. 19

3.3 general pipeline of our approaches. 21

3.4 Demonstration of Cumulative range geodesic filter. Left: input. Middle: tradition filters. Right: proposed filter. Note that, instead of having fixed shaped masks, cumulative range geodesic filter's mask only includes neighboring pixels that are similar to mask center. . . . 22

3.5 One example of bilateral roundup filter with mask size of 9, sample region size 81. Left: input. Middle: sampling region indicated by the purple boundary. Right: The output, indicated by the blue disconnected squares. Note that unlike cumulative range geodesic filter, here the output is not a connected set of pixels. 24

3.6 Left: Cumulative range geodesic. Right: Bilateral roundup. Bilateral roundup filtering preserves high-frequency, fine details better. 25

3.7 Pipeline of bilateral round up filtering with original image. 26

3.8 The improvement from cross-filtering with original image. Left: bilateral roundup on the bicubic upsampled image. Right: bilateral roundup on the original image. Note the inconsistent sharpness along the silhouette. Circled regions are less sharp. 27

3.9 Pipeline of bilateral roundup filtering multi-passes. 27

3.10 Multiple-passes results. From left to right: 0-pass, 1st-pass, 2nd-pass, 3rd-pass. More passes of this filter will keep sharpening the edges and eliminating small scale details. 28

3.11 A 2nd-pass result. Edges are well sharpened while the fine details are not overly removed. 29
3.12 Pre-filtering pipeline. 29
3.13 Pre-filtering original image improves poor edges. 30
3.14 Post-filtering texture enrichment pipeline. 30
3.15 The texture distribution scheme. Note that the order of textures appearing on this illustration is for demonstration. The actual textures are ordered randomly. 32
3.16 Adding multiple textures to upsampled image. 33
3.17 Problematic masks along interpolated edges. Left: Cumulative range geodesic. Right: Bilateral roundup. Note that the cluster of disconnected pixels must have very similar color as the mask center. 33
3.18 By removing pixels fall in the detected edge region, we are able sample mask data better. 34
3.19 The effect of edge penalizing. Left: bilateral roundup filter. Right: bilateral roundup filter with edge penalizing term. The marked region is slightly sharper. 35
3.20 With masks that covers both sides of an edge, penalizing just the edge pixels would result in a mask that still does not have a dominant color. 36
3.21 If the mask covers both sides and a thin feature itself, penalizing the edge means penalize the thin feature, which will result in losing the thin feature. 37
3.22 Edge penalize result. Dragonfly, 3 times upsampled. Top left: input at reduced resolution. Top right: full result image at reduced resolution. Bottom left: silhouette of the wing is well preserved. Bottom right: Thin features such as the legs are destroyed. 38
3.23 Left: input at reduced resolution. Top row: Bicubic upsampled. Bottom row: Cumulative Ranged Geodesic. Sharp edges are better constructed. 39
3.24 Left: input at reduced resolution. Top: Bicubic upsampled. Bottom: Cumulative Ranged Geodesic. Middle-scale irregular structures are preserved. 40
3.25 Left: input at reduced resolution. Top: Bicubic upsampled. Bottom: Cumulative Ranged Geodesic. Middle-scale structures are preserved. . 40

3.26 Left: input at reduced resolution. Top: Bicubic upsampled. Bottom: Cumulative Ranged Geodesic. Fine textures of the bricks are lost. . . 41
3.27 Dragonfly: 3-times upsampled. Top: Bicubic upsampled. Bottom: proposed multi-pass approach. 42
3.28 Texture Enrichment result. Dragonfly: 3-times upsampled. Top: Bicubic upsampled. Bottom: proposed multi-pass upsampled. Same amount of texture added to both images. 43
3.29 The result of varying texture input. Top: cement, plate, grass, gravel, towel. Bottom: hair, hay, fur, canvas, paint. 45
3.30 Extremely strong texture added. Top: proposed texture enrichment approach. Bottom: 10 times the same texture applied. 46
3.31 Comparison between bilateral roundup filter and median filter. Top: bilateral roundup. Bottom: median filter with a 5x5 mask. 47
4.1 6-times bicubic upsampled. Red: Sharp edges are terribly destroyed. Yellow: Textured areas are somewhat acceptable. 50
4.2 The difference between common interpolation and the proposed method. Top row: common interpolation. Bottow row: proposed method. 51
4.3 Pipeline. Left: The proposed pipeline. Right: A simple example that helps demonstrate the processes of each stage. 52
4.4 Mass-spring system. Each pixel is connected to its neighbours with a spring. 55
4.5 Angular Force. Left: two adjacent pixels along an edge. Middle: Their current orientation. Right: The local gradient, which signals the target orientation. Red arrows show the force directions. 57
4.6 Duck: 5-times upsampled. Top: input at reduced resolution. 2nd Top: showing how pixels are clustered along object bounadries. 3rd Top: bicubic upsampled. Bottom: Our result. Note that object boundaries (the eye and white stripes) are well preserved. 62
4.7 Harvest: 6-times upsampled. Top: input at reduced resolution. 2nd Top: Object boundary greatly sharpened. 3rd Top: Preservation of middle-scale details such as the highlights on corn. Bottom: Linear thin features (like text) are distorted. 63

4.8 Baby: 6-times upsampled. Top: input at reduced resolution. 2nd Top: Highlight on the eye is sharpened. Pupil geometry is slightly changed. 3rd Top: Nose boundary sharpened. Bottom: Mild Voronoi-shaped textures are introduced to smooth areas. 64
4.9 Cityscape: 4-times upsampled. Top: input at reduced resolution. Mid-left: Building boundary is sharpened with slight distortion. Mid-right: Similar shapes are distorted inconsistently. Bottom: Texture resembles paint strokes. Vehicles are interestingly distorted. 65
4.10 Plant: 4-times upsampled. Top: input at reduced resolution. Mid-left: Uneven distortions made the spikes look more ferocious. Mid-right: The tip of the leaf is preserved even when its surrounded by high frequency textures. Bottom: Texture introduced by our approach is similar to the texture of gravel and rocks. 66
4.11 Dragonfly: 6-times upsampled. Top: input at reduced resolution. Left column: bicubic upsampled. Right column: proposed approach. Note that the straight tail is slightly crooked; extra edges introduced around eye; the leaf texture is changed. 67
4.12 Comparison between our and R. Fattal's result [17]. Can: 8-times upsampled. Top row: input at reduced resolution. Left Column: our proposed approach. Right Column: R. Fattal's approach. Overall, we reconstruct sharp edges better. However, migrating pixels may change texture and geometric shapes. 68
5.1 Historical paintings illustrating our intention. Above: paintings by Oudry and El Greco demonstrating distortions and problematic perspectives. Below: paintings by Caravaggio and Waterhouse in a photorealistic style. 73
5.2 Schematic of our processing pipeline. Dashed boxes represent optional steps. 75
5.3 Effect of the full process. Upper left: original image. Upper right: warping, no painterly effect. Lower left: painterly effect, no warping. Lower right: both warping and painterly effect. 76
5.4 Progression of an image through the pipeline. Top row left to right: original image; SLIC segments; warped SLIC segments; Bottom row left to right: warped image; warping plus painterly effect. 78

5.5 Generic result - Cathedral. 80
5.6 Generic result - Secretary Bird. 80
5.7 Generic result - Garlic. 81
5.8 Generic result - Hood. 81
5.9 Generic result - Old man. 82
5.10 Generic result - Boxer. 82
5.11 Generic result - Street. 83
5.12 Generic result - Light house. 83
5.13 Generic result - Books. 84
5.14 original images (hood, cathedral, garlic, old man, boxer, lighthouse, street, books, secretarybird). 84
5.15 Spring strength increases from top to bottom: spring parameters are $\gamma = 1$, $\gamma = 1.5$, $\gamma = 2$, $\gamma = 3$. 85
5.16 From top to bottom: SLIC regions of diameter 20, 40, 60, 80 pixels. . 86
5.17 Comparison of our approach with abstract rendering by Sisley. Left: Sisley. Right: ours. 88
5.18 Two failure cases. Above: crying. Below: flamingo. 89

Chapter 1

Introduction

Image upsampling is the operation that constructs a higher-resolution image from lower-resolution input [41]. Common upsampling schemes smooth sharp edges and introduce blurry or blocky texture. Smoothing along an edge is caused by interpolating the gaps between two sides of the edge on the upsampled image plane. Interpolation also smooths out high-frequency textures, thus causing the loss of fine details.

Image stylization refers to the techniques capable of transforming 2D image into synthetic artwork [40]. Image stylization and image upsampling are quite distinct topics in image processing. In this thesis, we combine the two topics together. We hope that the stylization process can alleviate the problems from traditional upsampling. By sharpening, we hope to remove blocky or blurry artifacts and exhibit an abstract look; by introducing arbitrary high-frequency textures, we hope to bring back some fine details that are lost in the interpolation processes. Specifically, we try to simulate the 19th-centry European romanticism paintings, which often contain a large amount of fine detail with little or no visible paint strokes.

We have two main objectives in this thesis:

• We want the upsampled image to be sharp. Specifically, we want to focus on edge (e.g. object boundaries, feature silhouettes) preservation. We want to reconstruct the same level of edge sharpness that appeared on the input image.

• We want to transform a photorealistic image into upsampled stylized artwork. We want to produce high resolution images that give an abstract impression globally and contains high-frequency details locally.

We present two directions to achieve our goals. First we explore two filters: cumulative range geodesic filter and bilateral filter. The former excels at edge sharpening

and the latter finds a better balance between edge sharpening and fine-detail preservation. Second, we propose a pixel migration idea that relocates pixels in the image plane. We implemented pixel migration using a mass-spring physics simulation, where the pixel movement is enforced by varying spring rest lengths and spring coefficients. Applying pixel migration on the upsampled plane, we present a fast and effective edge-sharpening process.

Our first direction intends to create upsampled and stylized images using filtering-based techniques. We investigated cumulative range geodesic filter, which cumulatively grows continuous irregular-shaped masks that respect object boundaries. This filter greatly sharpens the image with the sacrifice of fine details. We also introduced the bilateral roundup filter, which generates irregular-shaped masks that are not necessarily continuous. We achieved various levels of sharpness through variations of bilateral roundup filter, such as cross-filtering with the original image or multi-pass filtering. We also introduced a simple texture enrichment process that borrows texture from other images and introduces fine details into the filtered results. Overall, the filtering process can help us better reconstruct object silhouettes and sharp edges; and the texture enrichment process can introduce new arbitrary details to compensate the fine details removed by the filters, thus increasing the aesthetic value of upsampled images.

Our second direction is to relocate the original pixel samples on the upsampled image plane. The upsampled blurry edge is caused by interpolating the gaps between the pixels located on either side of an edge. The pixel migration principle is to move the pixels on either side of an edge as close as possible to each other, such that there is no gap between the two sides. No gaps mean there will not be any blurry interpolated pixels along the edge, thus the edge sharpness would remain. We present a mass-spring system that implements the pixel migration idea. We connect each pixel to its 4 adjacent neighbours by virtual springs whose rest lengths and spring constants are functions of the pixels they connect. Thus, we are able to reposition pixels and shrink the gaps along sharp edges by varying rest lengths and spring constants. This approach generates sharp high-resolution images with minor distortion artifacts on long thin features.

As we were investigating the mass-spring system, we were fascinated by the distortion effect exhibited along object boundaries. The effect resembles the deliberate distortions observed in caricature or other exaggerated semi-representational depictions

of the subject matter. Thus as a side project, we further explored the mass-spring implementation on the original image plane and developed a warping technique that simulates painterly effects. The warping is achieved by assigning random rest lengths to the mass-spring system. We then achieve a painterly effect by applying a watercolor filter [6] introduced by Bousseau et al. The results were published in Expressive, 2015 [29]. We also encourage users to adopt their preferred stylization filters after the warping mechanism, which will produce extra aesthetic value of the user's preference. Though this image warping research digressed from our image upsampling goal, it demonstrates an application of the pixel migration idea. The caricature effect from image warping can also be applied to upsampled images, which then becomes one application of our stylized upsampling methodology.

Our research has three major contributions:

• We shift our upsampling objective from photorealistic to artistic. We suggest constructing stylized super-resolution images by sacrificing photorealism and the faithfulness to the input. This approach grants us the freedom of modifying fine details without being faithful to the input. We can thus produce high-resolution images with high aesthetic value. We proposed filtering-based techniques to implement this idea.

• We proposed the idea of pixel migration and a mass-spring simulation implementation of the idea. Applying pixel migration on the upsampled image plane yields a fast and effective algorithm for edge sharpening.

• As a side project, we investigated applying pixel migration on the original image plane, producing a warping mechanism that simulates painterly effects.

The thesis is organized as follows. Chapter 2 reviews previous literature in image upsampling, artistic stylization and painterly rendering. Chapter 3 presents cumulative range geodesic filter and bilateral roundup filter, combined with some research notes towards mask data analysis. Chapter 4 proposes the pixel migration idea, with the details of a mass-spring implementation that relocates pixels in the upsampled image plane. Chapter 5 presents the warping effect for painterly rendering. The last chapter concludes our approaches, evaluates our results, and provides comments toward future research.

Chapter 2

Previous Work

2.1 Introduction

This chapter describes previous research trends and solutions related to the three topics we discuss in this thesis: Artistic Stylization, Image Upsampling and Image Warping. We review these previous publications here as it will helps us to better recognize and understand the problems, identify and avoid known pitfalls, design and improve our algorithms, and compare and evaluate our results.

Image upsampling is the operation that constructs a higher-resolution image from lower-resolution input [41]. Section 2.2 lays out previous efforts on polynomial interpolation-based upsampling, example-based upsampling, and the techniques that use variations of low-resolution input to facilitate upsampling. We also discussed processes we adopted to gather edge or texture information.

Artistic stylization refers to the techniques capable of transforming 2D footage into synthetic artwork [40]. Section 2.3 presents previous approaches that produces various artistic styles. We adopted the watercolor filter by Bousseau et al. [6]. in our image warping research.

Section 2.4 describes previous literature that is relevant to our painterly effect and image warping research. We discussed painterly effects with a special emphasis on extreme effects that are similar to our proposed effect. We also included literature about image segmentation, since it is one important component in our proposed pipeline.

2.2 Image Upsampling

Image upsampling is the operation which estimates a fine, high-resolution image from a coarse, low-resolution input [41]. High-resolution means high pixel density within an image. Higher-resolution images offer finer details, which is of great importance in applications such as medical imaging, satellite imaging, military imaging, underwater imaging, and HDTV [16].

The direct solution to achieve high resolution is to reduce pixel size in sensor manufacturing, that is, to increase the number of pixels per unit area. However, the drawback is that the amount of available light for each pixel is decreased. Raising sensor density also greatly increases the hardware expense. Therefore, to achieve high-resolution image, the most feasible solution is to integrate hardware capabilities with software support, i.e., image upsampling techniques [16]. This section discusses trends and processes in image upsampling techniques.

Polynomial image interpolation

Image upsampling increases pixel count, which means the pixels from low-resolution input are only sparsely placed on the upsampled image plane. The value of intermediate pixels are unknown. The process of estimating the intermediate pixels between known pixels is called image interpolation.

The traditional view of interpolation is that the interpolation can be represented as an arbitrary continuous function, which is a discrete sum of weighted or shifted synthesis functions. We use the terminology of polynomial, because the interpolation can be performed using polynomials of a spacing parameter [16]. Given the spacings between unknown pixels and known, regularly sampled pixel data, we can use polynomial functions to estimate unknown pixel values. Nearest-neighbour interpolation, bilinear interpolation [46], and bicubic interpolation [24] are popular polynomial interpolations. They are popular due to their simplicity. However, they usually introduce severe blocking, aliasing, or blurring artifacts.

There exist many adaptive, improved polynomial interpolation approaches [22, 39, 43]. They focus on either improving visual quality (alleviating blurring artifacts or preserving edges) or reducing computational cost of the interpolation algorithm. They produce acceptable results when the upsampling scale is small. However, as the upsampling scale increases, they are unlikely to be capable of constructing sharp

results, due to the sparser known-pixel distribution.

Example-based upsampling

Example-based upsampling creates or adds high-resolution details that are not present in the original image, with the help or guidance from other images. The "other images" here refer images found in image databases, as well as downsampled or processed original image and original image segments. Example-based upsampling has been a popular trend over the past decade. It is powerful because the known pixel data is no longer limited to the pixel data of the input. We also have access to pixels from image databases. We can use them to help estimate the unknown intermediate pixels.

Freeman et al. developed example-based super resolution [18], a fast and simple, one-pass, training-based algorithm for creating plausible high-frequency details in enlarged images. They divide blurry polynomial interpolated images into patches (small windows centered at each pixel). They find high-resolution details in image database that matches the patches. Then they replace the patches with the matching high-resolution details. Kim and Kwon proposed an improved example-based super-resolution framework [27]. Instead of finding the best match in the image database, they find multiple candidates. Then they combine the candidates based on their estimated confidence. Glasner et al. proposed an example-based approach that uses the original image segments as image database [19]. They observed that patches in a natural image tend to redundantly recur many times. Their approach identifies the recurrence of patches in a single image, then uses these patches as database to achieve example-based super-resolution. Based on the patch-based model of Freeman et.al. [18], Hacohen et al. [21] proposed an improved method that consider textured areas as large segments instead of patches. They discovered that Freeman's patch-based model could only find locally best match, but lacked global consistency. By segmenting input image into large, similar-textured regions, they are able to find globally consistent matches in the image database. Thus, their algorithm introduces fine-scale, consistent details to large textured regions.

Overall, example-based approaches have the advantage of recreating fine details. They learn information from segments of the original image, and use this information to find higher-resolution textures or image segments in image database. However, these matching image segments, no matter how closely they resemble the example, are not the same as the example. Thus, they achieve higher resolution with fine

details by sacrificing the integrity of the original input. Moreover, to achieve close resemblance, example-based approaches usually involve managing, categorizing and searching through a vast collection of images. These tasks add difficulty to the image upsampling problem. The research in this thesis avoids using example-based approaches. The hope is that we can achieve high-resolution images without or with very limited help from other images.

Use lower-resolution as guidance

There are several approaches that use processed or lower-resolution input to guide the upsampling. Fattal proposed an edge statistics approach to achieve sharp upsampled images [17]. He points out that there exists a unique dependency between image derivatives at different resolutions. Pixel differences at higher resolutions depend on their distance from an edge, the spatial distribution of that edge and the total intensity jump across it. Such information are called edge statistics, and can be collected and estimated in low-resolution. Fattal proposed methods to predict intensity differences in the upsampled image given the edge parameters observed at the low-resolution input. The edges retrieved are sharp. The blocky artifacts coming from upsampling are minimal.

Shan et al. [41] proposed a fast upsampling method that uses deconvolution [42] result of upsampled image as feedback to enhance upsampling process. By using a Gaussian kernel, they introduced an iterative de-convolution and re-convolution scheme that is called feedback control loop. They gather information of the pixels discarded by the feedback control loop, and introduce these pixels to upsampled image plane to enhance details. Their approach simulates an inversion of the image formation process and produces a clear, finer-resolution image based on a coarse-level image. Essentially, their approach is observing the difference between the input image and processed input image, and using the difference to help achieve super-resolution.

Kopf et al. proposed a joint bilateral upsampling [28] that improves a high-resolution input image by cross-filtering with its filtered low-resolution solution. The idea of joint bilateral filter is that range filter is applied to a second guidance image. In the context of image upsampling, this guidance image is usually the downsampled (or the original) image. We adopted the cross filtering idea in our bilateral roundup filter and cumulative range geodesic filter. Applying these filters directly on the upsampled image will result a thick band-shaped mask along sharp edges. It is not ideal because

these pixels are interpolated and smooth: they destroyed the gradient intensity along sharp edges. However, if we cross-filter with the original image, we will not include these interpolated pixels in the mask. Thus, the upsampled edges are expected to be sharper.

Texture measurement and enrichment

Both image upsampling and image abstraction processes can destroy fine details or introduce smooth, dull regions. Though we do not focus on maintaining the photo-realistic look, we want the upsampled images to include small-scale details, by either replicating the texture from the input image, or borrowing textures from other images [32]. This requires us to distinguish highly-textured areas (e.g. grass, sand) from smooth areas (e.g. sky).

We define textureness as how frequently the color intensity changes around the pixel neighbourhood. Bae et al. [2] proposed a textureness measurement that we find quite useful. Their approach starts by high-pass filtering the frequency domain of the input. Then they build power maps by finding local average obtained from low-pass filtering of the frequency magnitude. Bae's approach enable us to identify textured regions of the original image automatically. These textured regions are likely to be destroyed by the upsampling and interpolation. Having Bae's textureness map, we can add another term to our bilateral filter or cumulative range geodesic filter, which helps us to sample pixels differently on highly textured areas. Alternatively, we can also introduce textures specifically for those areas after upsampling.

Xu et al. proposed an image smoothing approach using L0 gradient minimization [48]. Their work sharpens major edges and eliminates some degree of low-amplitude structures. They globally control how many non-zero gradients are permitted in order to approximate prominent structure in a sparsity-control manner. In contrast to Bae's approach, Xu's L0 gradient minimization discards local high-frequency information, but preserves globally significant structures. Thus, L0 gradient minimization helps us distinguish globally important edges. Also, combined with an ordinary gradient magnitude map, L0 gradient minimization lets us identify discarded textures. Similar to Bae's approach, we can potentially use this textureness information to guide the sampling process of various filters we applied.

Overall, previous image upsampling literature pursues a photographic style that precisely respects the low-resolution input. They want to preserve salient features

as well as estimating or reconstructing fine details. Our research shares the goal of preserving sharp, salient features. However, we also allow the result images to be stylized. We hope that by introducing a stylized look, losing or changing fine details would become tolerable; and eventually we can produce upsampled images that are aesthetically pleasing.

2.3 Artistic Stylization

Artistic stylization refers to the techniques capable of transforming 2D footage into synthetic artwork [40]. There has been a vast amount of publications in this field, and they diversify for different styles and aesthetic values. This section lays out the literature that aims for styles or effects that are comparable to our approach. We also describe operations, algorithms, and processes that we adopted or found inspiring.

Stroke-Based rendering

Stroke-based rending is the process of synthesizing artwork by compositing rendering marks (such as lines, brush strokes, or even larger primitives such as tiles) upon a digital canvas [40].

The stroke-based rendering paradigm was proposed in Haeberli's seminal work on user-assisted image stylization [23]. Haeberli's approach creates abstract images using an ordered collection of paint strokes. Given photographic images as input, by controlling the color, shape, size, and orientation of individual brush strokes, Haeberli's approach can create various impressionistic paintings. Inspired by his work, the general pipeline of our research is similar. We take photographic images as input; apply various filtering, color manipulation, pixel migration techniques; and create various types of stylized images as output. Moreover, many operations and processes adopted by our research are inspired by the techniques mentioned in Haeberli's paper. For instance, his pushing edges process is one of the earlier edge exaggeration processes we have investigated; our experiments on residual (difference between before and after filtering) calculation are similar to Haeberli's color richness enhancement technique.

Following Haeberli's work, stroke-based rendering became a popular trend for painterly style rendering. Litwinowicz proposed a stroke-based technique for impressionist effect [30]. His approach respects object boundaries by clipping the paint strokes. He also orients, moves, adds, or deletes paint strokes to match the image

content. Hertzman proposed another method that creates impressionist images with a series of spline brush strokes [25]. Brush strokes match the colors of the image source, and several layers of progressively smaller brush strokes are applied so that the visual emphasis in the painting corresponds roughly to the spatial energy present in the source image. The technique for painting long, curved brush strokes aligned with normals of image gradients is inspiring. We, too, want to recognize object boundaries or edge orientations and use that information to guide our filtering processes.

Filtering-based Rendering

Filtering is a common yet essential part of many signal processing systems, including image processing. Various filtering techniques serve quite different purposes, such as image blurring or sharpening, edge detection and exaggeration, noise introduction or removal, etc. By combining and varying these filtering techniques, we can achieve a wide range of artistic styles.

Winnemoller proposed XDoG, an extended difference-of-Gaussians filtering process that can generate various stylized images [47]. The filter can control edge-detection sensitivity, thus change the tone-mapping response of the operator. It can also adjust the thresholding so that a soft ramp between the edge and non-edge values can be created. Overall, by controlling the parameters, XDoG simulates styles like natural-media style, ghosting, speed-lines, negative edges, thresholding, hatching, etc. Our research does not directly use XDoG filter, but we find the idea of thresholding and using edge-detection sensitivity to guide mask generation inspiring.

Kass and Solomon [26] proposed a series of filters based on a smoothed histogram analysis of pixel neighbourhood. The traditional histogram sorts data in to a collection of bins. Kass and Solomon proposed a smoothed histogram by applying a Gaussian filter to the traditional histogram. Based on this smoothed histogram and its derivative, they proposed methods to identify the number of modes, the mode values, the mode widths and the percentage of the population contained within each mode. With this information, they present a series of filters and operations: a closest-mode filter for edge-preserving noise reduction; a median filter for extreme-noise reduction; a percentile filter that resembles morphological operators; a dominant-mode filter for edge sharpening; and several detail enhancement techniques based on dominant-mode filter. Kass and Solomon's work shares some goals with ours: edge sharpening, detail enhancing. However they focus on the direction of mask data processing rather than

our direction of mask data gathering. Compared to their work, the way we process our mask data is rather primitive: we simply just use the average color as output. They show us the histogram based mask data analysis, and how to produce different results purely from mask data processing. We would like to explore more variations of mask data analysis in future work.

Mould introduced cumulative range geodesic filtering, a variant of geodesic filtering [34, 35]. It preserves locally strongest edges, which lead to the preservation of not only large scale edges, but also middle to small scale structures depending on the surrounding context. The filtering process involves growing masks in a priority-based cumulative way, such that the distance in the image plane is lengthened proportional to the color distance. Therefore, irregular silhouettes can be preserved by irregular shaped masks. Though the main purpose of Mould's paper is image abstraction, this research shares similar goal, which is edge sharpening. Image upsampling and interpolation processes naturally introduce blur edges, which can be sharpened by cumulative range geodesic filtering. More importantly, irregular-shaped masks enable us to gather pixel neighbourhood information in a unique, local-structure-respectful way.

Bilateral filtering, introduced by Tomasi et al. [9], is an edge-preserving, image-smoothing filtering technique. It combines colors based on both their geometric closeness and the photometric similarity. The filter weighs mask samples by combining their color-space distances and Euclidean distances. Thus, the pixels that are close in both color and image plane distance will contribute more to the output of the mask. The technique of bilateral filtering is simple, but the idea of combining multiple meaningful terms into one filter is powerful. Bilateral filtering is one of the methods that we use to collect pixel neighbourhood data.

Based on traditional bilateral filtering, we evolved many variations to fit our upsampling purposes. In order to avoid using interpolated colors (which cause blurriness), we take our mask data directly from the original image. To avoid outliers and better preserve texture, we sort the mask pixels by their bilateral distance, and only take a portion of the best samples to calculate our result. To further sharpen edges, we cross-filter with the original image for several passes. All these variations and investigations are enabled by bilateral filtering. It is one of the most meaningful and powerful ways of gathering mask data.

Watercolor Effect

The physical processes behind watercolor painting and oil painting are very different. For watercolor painting, water-soluble pigments of various degrees of dryness and colors are applied onto the paper through hairy brushes. As the water flows on the surface of the paper, the pigment is transported, diluted and eventually deposited on the surface of the paper [40]. There are many stroke-based techniques to simulate oil paintings, whereas state-of-the-art watercolor effects are achieved by filtering or physics simulation. In order to distinguish ourselves from stroke-based rendering effects, we adopted variations of filtering-based watercolor effects.

Bousseau et al. [6] used morphological operators to achieve watercolor style. Basic morphological operators are dilation and erosion, which involve changing pixel color to the maximum intensity or minimum intensity color of its neighbourhood respectively. The dilation spreads the light features of the image whereas the erosion spreads the dark features. The region where morphological operators sample pixel neighbourhood data is called structuring element. The shape of the structuring element defines the shape of the filtered objects. Morphological opening is defined as a sequence of erosion followed by one dilation, whereas morphological closing is a sequence of dilation followed by one erosion. Opening operator removes light features of the image, whereas closing removes dark features. Bousseau et al. introduced a filter with the sequence of one closing operation followed by one opening operation. This filter removes both dark and light features and achieves an effect that resembles watercolor painting.

In addition to morphological operations, Bousseau et al. proposed a refined pipeline to achieve better watercolor effect [5]. The pipeline separates the processes into two stages: abstraction stage and effect stage. To achieve uniform color regions, the abstraction stage segments the image and applies morphological operators. Then the effect stage adds dry-brush or paper texture, then applies edge darkening filter. Our image warping technique has a similar pipeline. We also separate our processes to two stages. The warping stage involves segmenting the image, migrating the pixels and interpolating colors. The output of this stage is a warped image. The second stage is applying selected filters. Specifically for our experiments, a morphological watercolor filter by Bousseau et al. [6] is applied.

Curtis et al. [13] proposed another approach to achieve various artistic effects of watercolor. What is most interesting and inspired is they use a shallow-water fluid

simulation to calculate how color pigments or pigments streams would travel through the paper. Their work is one of the rare cases where physics-based simulations are used for non-photorealistic rendering. Our research uses mass-spring simulation to migrate pixels in original image plane or upsampled image plane. Though mass-spring system is not as complex as fluid simulation, they both fall into the category of physics.

2.4 Image Warping for Painterly Effect

Overview of painterly rendering

Painterly rendering has been a major topic in the non-photorealistic rendering literature. Broadly speaking, synthetic painted images can be created in three ways: through digital artists exercising synthetic painting tools [3, 10]; through paint primitives being distributed according to the details of a geometric scene [31]; or through image-space operations over an input image, usually a photograph, e.g., distributing strokes that match local image properties [25]. We are mainly interested in the last of these approaches.

Among the most-studied forms of painterly rendering is portraiture, with specialized techniques to draw human faces. Perhaps the most effective method is that of Zhou and Zhu [49], who use active shape models to obtain an estimate of facial feature locations, and then learn a mapping from user-drawn strokes to the face geometry. By using a large database of hand-painted portaits, their mapping is made to be fairly reliable; novel portraits can then be made from photos by estimating the face structure and drawing from the stroke placement database. This example-based system produces reasonably realistic drawings, albeit abstracted owing to the small number of strokes. An earlier method that is somewhat closer to our intent is provided by Gooch et al. [20], who automatically estimate facial feature locations in structured photographs, then distort the face shapes to create caricatures. This work is effective but by its nature restricted to portraits with empty backgrounds. Also, the proposed rendering is a monochrome pen-and-ink style, not a painterly rendering.

In general, semantically meaningful image manipulations are difficult to achieve automatically. Two standard workarounds are either to employ user-provided semantics [36] or to restrict the subject matter and layout of the input images, as in the case of portraiture [20]. In the image warping section of our thesis, we hope to

create painterly versions of arbitrary photographs, with no limitations on the layout or content. While we do not achieve the quality of results that can be attained by specialized methods applied within their target domains, this is a necessary trade-off for the robustness we gain instead.

Painterly rendering from photographs suffers from the fixed perspective of the lens. In the case of object-space rendering, the perspective can be altered as the scene is projected onto the image plane, either through interpolating multiple linear perspectives [44] or potentially by performing an entirely nonlinear projection [7, 11]. However, such techniques depend on having a full scene description; image-space techniques are more widely applicable.

Abstraction in non-photorealistic rendering falls generally into two categories. First, a basic abstraction can be achieved using a filter to remove details: various linear and nonlinear filters have been proposed for this purpose, and we consider stroke-based rendering a special case of such filters, where an image's pixel colors are replaced by the stroke colors. Second, more deliberate shape abstraction is possible using a higher-level representation of the image; the most common approach here is to create an initial segmentation of the image and then simplify the resulting segments [14, 33, 37]. We are less interested in purely abstracting the image by removing detail than in reducing its faithfulness to the original photograph; we prefer our final image to be quite detailed in spite of the abstraction process.

Extreme stylizations

Quite extreme stylizations are possible, ranging from somewhat representational forms as in cubism [8], to quite abstract forms as seen in the arty shapes of Song et al. [45]. Our image warping technique can often produce quite extreme results, especially when the input has strong salient features, like human portraits, still lives, or cityscapes.

The cubist rendering system proposed by Collomosse and Hall [8] has some stages in common with our approach. Taking multiple images as input, this system detects salient image elements, distorts them through transforming a super-quadric fit to the element perimeter to a different super-quadric, and then fuses multiple elements into a single image to produce a multi-perspective cubist style output. A customized painterly filter helps with the final fusion. Our pipeline uses a single image and attempts to produce a more conventionally representational output image rather than

the cubist-style renderings sought by Collomosse and Hall.

Arty shapes by Song et al. [45] produces extremely abstract images. They fit each region within a segmented photograph with a variety of simple shapes, such as circles, triangles, squares, superellipses. The system automatically chooses the shape that best represents the segmented region, and several layers of shapes are oriented and scaled to fit the region. Their results exhibit very high level of abstraction. With all the small-scale or mid-scale details removed, the objects in their results look like different colored and shaped cobblestones placed together. Though high level of abstraction is rather interesting, we want to find a better balance between abstraction and photorealism. We plan to exaggerate the warping along object boundaries as much as possible, while being faithful to the input on textured areas.

Overall, the method that is closest to ours is "Sisley" the abstract painter, which is a semi-automatic approach for highly stylized painterly filtering [36]. The Sisley method involves segmenting the image and optionally assigning semantic labels and abstraction levels to nodes in a segmentation hierarchy; strokes are then distributed over the image plane so as to convey the image content with the appropriate abstraction level. Stroke colors and stroke geometry are perturbed, potentially dramatically, in order to achieve a high level of abstraction. This method produces quite convincing painterly images in an expressionist style. We seek a more representational style, possibly with exaggerated colors as Sisley used; our underlying process is also quite different. The Sisley method uses shape abstraction to simplify the segments, and then relies on independent perturbations of the stroke geometries to produce further distortion. Even at low levels of abstraction, details of the image are lost. Our approach separates the image warping from the painterly rendering; while we recommend applying a rendering technique such that the resulting image has a painterly appearance, most of the effect is due to the initial automatic warping and not to the painterly filter, which in any case need not be stroke-based.

Image segmentation

As mentioned above, many image stylization effects rely on segmenting the input image into regions. Segmenting an image means dividing the image into many continuous areas. The segments are distinct from each other in some way, often based on color, texture content, object boundaries, or otherwise. Individual segments are often meaningless, but meaning emerges from the overall arrangement of segments [40].

Comaniciu and Meer [12] proposed a mean-shift approach for image segmentation. Mean-shift refers to a nonparametric procedure that identifies arbitrary shaped clusters in complex multimodal feature space. In the context of image processing, this multimodal feature space is the image plane, and the arbitrary shaped clusters are the resulting segments. Mean-shift can recover significant image features and be adapted for image segmentation. Within Comaniciu and Meer's algorithm, colors are represented in the perceptually uniform color space "Luv", which produces region boundaries that are more meaningful for human observers. This segmentation approach is adopted by DeCarlo and Santella's seminal work on segmentation driven image stylization [14].

DeCarlo and Santella's system automatically computes segmentations at different scales. A hierarchy is inferred by computing the overlap between regions at different levels, and each region is assigned a single parent, up to the top level where the entire image is a single region. This hierarchy, plus a set of image edges, is precomputed for a given image. After the hierarchy is computed, the segmentations are smoothed by low-pass filtering the segmentation boundary curves [40]. Their results exhibit a line-drawing style using bold edges and large regions of constant color.

Achanta et al. [1] proposed simple linear iterative clustering (SLIC), which outputs nearly uniform sized, high quality, compact superpixels (or segments) with a low computational expense. The algorithm performs a local clustering of pixels in the 5-D space defined by L, a, b values of the CIELAB color space and the x, y pixel coordinates. Similar to bilateral filtering, SLICs distance measure takes both color distance term and image plane distance term into consideration. The color distance term helps the superpixels to respect object boundaries. The image plane distance term enforces compactness and uniformity of superpixels.

Like many other techniques that perform image-space stylization, we depend on an initial segmentation of the image in order to preserve image features. We use the oversegmentation provided by SLIC. While the resulting segments have no semantic meaning, they tend to respect image edges and they tend to have similar sizes to nearby segments, a property that helps us distribute our distortion throughout the image. Controlling the diameter of superpixels enables us to match SLIC regions with the semantic knowledge of the image. For instance, for human portraits, we may want the SLIC regions to be fairly large, so that significant facial features (eyes, eyebrows, ears, mouth, and nose) would fit in one SLIC region; whereas for nature

scenes or cityscapes, we want to have smaller SLIC regions so that the high-frequency details are better preserved. Though observing and understanding the semantics of an image is not a necessary step for SLIC, we find it beneficial for our image warping processes.

We accomplish image warping using a mass-spring system, akin to the "pelting" procedure of Piponi and Borshukov [38], who proposed mass-spring systems for texture coordinate assignment. While Piponi and Borshukov sought a smooth stretching of the textured surface, and hence used constant spring lengths, we deliberately stretch some portions more than others to obtain warping, using spatially varying spring lengths and spring constants to do so. The classic technique for image warping involves thin-plate splines [4] but the smoothness of the splines creates a coherent distortion. We want our distortion to be coherent only insofar as it modifies a single object; the warp can be discontinuous across image edges. The mass-spring system with spring parameters associated with SLIC regions accomplishes this aim.

Chapter 3

Filtering-based Image Upsampling

3.1 Introduction

Figure 3.1: Input images used in this chapter: barktree, picnic table, stranger, roughwater, sedona, and dragonfly.

Figure 3.2 shows three basic upsampling interpolation approaches: nearest neighbor interpolation, bilinear interpolation [46], and bicubic interpolation [24]. If you

examine the boundary and the internal texture of the tree, you will find the interpolation processes have destroyed sharp edges and introduced clustering, smooth, or blurry textures. To make things worse, if we increase the upsampling scale to 10 times or higher, the interpolated pixels' colors will significantly drift away from the original input, which will make the upsampled image blurry, dull, and aesthetically unpleasant.

Figure 3.2: Three basic interpolation methods. Left: Nearest neighbour interpolation, which introduces clustering pixel blocks. Middle: Bilinear interpolation. Right: bicubic interpolation. Both bilinear and bicubic interpolation smooth out the sharp edges.

The quality of upsampled super-resolution images has risen over the years. Either by using adaptive polynomial interpolations [22, 24, 39, 43, 46] or example-based methods [17–19, 21, 27, 28], the main focus of the field is to produce photorealistic super-resolution images that closely resemble the input and have the same level of detail. This is a difficult problem. Therefore, we want to approach it differently. The plan is to simplify the problem by allowing the upsampled images to be abstract or stylized. That is, we detect, sharpen, and preserve only the salient features of the input image, such as object boundaries. Plus, we are free to reduce, modify, or introduce medium or fine-scale details, since the stylized results no longer need to be photorealistic or faithful to the input image. We hope that we can produce super-resolution images that are sharp, stylized, interesting, and aesthetically pleasing.

We attempt to achieve stylized super-resolution images with various filtering techniques. Filtering is a common yet essential part of many signal processing systems, including image processing. Image-space filtering usually involves collecting a pixel's

neighbourhood data, and process the data in some reasonable ways, e.g. taking the average as the output. Often, various filters are applied to blur or sharpen an image, detect or exaggerate object edges, introduce or remove noise, or achieve non-photorealistic stylization. Due to the difference of purpose, various filters may have very different ways of collecting and processing mask data. When we put image filtering in the context of image upsampling, the filtering process can be quite different. Because we have both original image plane and upsampled image plane to gather our mask data, we often need much larger masks compare to common filters; and due to the increased mask size, we can design algorithms that uses a small portion of our mask samples to calculate the output.

In this chapter, we propose variations of two filters: cumulative range geodesic filter [34, 35] and bilateral filter [9]. Cumulative range geodesic filter is used for image abstraction and stylization; bilateral filter is used for edge-preserving image smoothing. Both filters respect sharp edges, which is a great advantage in image upsampling. By adopting these filters, we intend to inherit their advantage in edge preserving; by varying these filters, we intend to introduce a stylized look to the resulting super-resolution images. We also explore other post-processing techniques that add fine details.

We make two contributions in this chapter:

• We suggest constructing stylized super-resolution images by sacrificing photo-realism and the faithfulness to the input. This will simplify the image upsampling problem and produce aesthetically appealing images. We have not found precedent for combining stylization with image upsampling or super-resolution.

• We propose a series of operations that achieve sharp super-resolution images. Based on cumulative range geodesic filter and bilateral filter, these operations gather pixel-neighbourhood data in a versatile, edge-sensitive way. By controlling parameters like the mask size or the number of cross-filtering passes, we are able to achieve different levels of edge sharpness. The filtering process also abstract the upsampled images, which gives the results a painterly appearance.

This chapter is organized as follows. Section 3.2 describes two filters with their variations and respective results. Section 3.3 describes our efforts, successes and failures towards mask data analysis and processing. Section 3.4 demonstrate and discuss the results of proposed methods. Section 3.5 concludes the chapter by summarizing the successes and difficulties of filter-based upsampling. We also suggest possible

future directions.

3.2 Algorithms

Figure 3.3 shows the general pipeline of our approaches. First we upsample the input image using bicubic interpolation. Then we filter or cross-filter the upsampled image using proposed filters. Following this general pipeline, we explore several variations: pre-filtering the input image, cross-filtering the upsampled image with input image, re-filtering the cross-filtered image, and adding textures to filtered image.

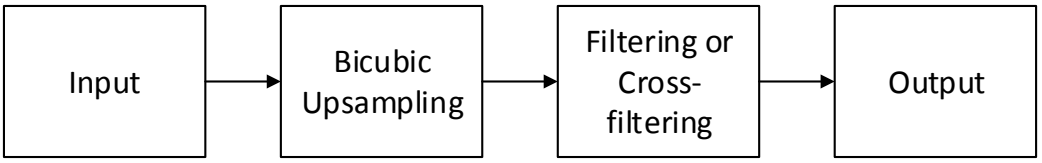

Figure 3.3: general pipeline of our approaches.

3.2.1 Cumulative Range Geodesic Filtering

Many common filtering techniques find masks centered at each input pixel, and use a function to decide how much each pixel in this mask contributes to the output. Since the mask shape is fixed, often it will include some or many pixels that are not similar to the mask center. Due to the contribution of these pixels, sharp edges from the original image can be damaged or destroyed.

Cumulative range geodesic filtering [34,35] finds arbitrary-shaped masks that can guarantee that all the pixels in this mask are relatively similar to the center. For each input pixel, it keeps a sorted list of neighboring pixels as mask candidates. Equation (3.1) is how the incremental cost is calculated from one mask pixel to an adjacent pixel.

$$C_{inc} = |I(h) - I(0)| + \gamma \times |I(h) - I(g)|, \tag{3.1}$$

where $|I(h) - I(g)|$ represents the incremental color space distance from top pixel g on the sorted candidate list to its neighbor h; $|I(h) - I(0)|$ represents the colorspace

distance between the mask center 0 and h. The constant γ represents the relative importance of the two terms.

We grow the masks in paths that have the lowest incremental costs. Every iteration, the top pixel of the list is added to the mask, and the neighbors of that pixel are added to the sorted list. The top pixels chosen this way are the most similar to mask center. Therefore, instead of including all neighboring pixels, the mask will only contain neighboring pixels that are similar to mask center. Assuming the pixels along one side of an edge are similar, then these pixels'masks will contain these pixels themselves, resulting the output of these pixels to be similar. Thus, by excluding dissimilar neighbouring pixels, instead of being smoothed out, pixels on sharp edges will remain their intensity.

We continue the mask growth iteratively. When the filter reaches the target mask size, it calculates the average color of the mask as the output. Figure 3.4 shows the mask-shape difference between traditional filtering and cumulative range geodesic filtering. The complexity of geodesic filtering is $kn\log(n)$ for a k-pixel image with mask size n. The results of cumulative range geodesic filtering is discussed in Section 3.4.

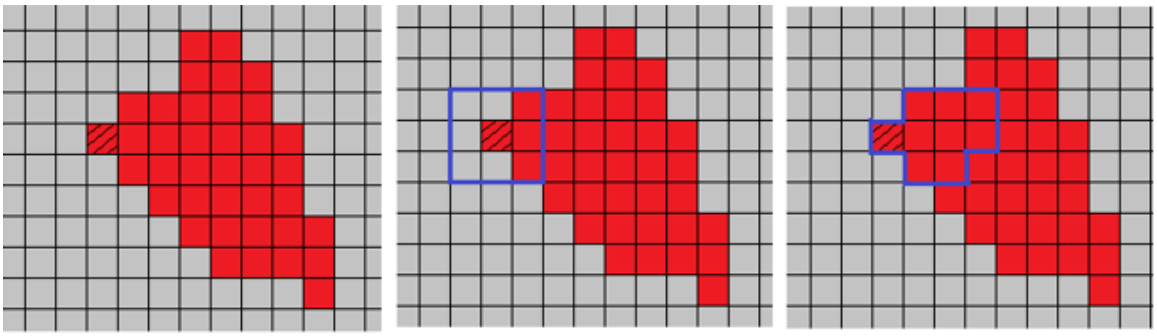

Figure 3.4: Demonstration of Cumulative range geodesic filter. Left: input. Middle: tradition filters. Right: proposed filter. Note that, instead of having fixed shaped masks, cumulative range geodesic filter's mask only includes neighboring pixels that are similar to mask center.

In the context of large-scale upsampling, cumulative range geodesic filtering itself may not be sufficient for edge preservation. After upsampling, one row of pixels from an input sharp edge will be interpolated to n rows of smoothly transitioned pixels. When these pixels grow masks, they will tend to traverse along the smoothed edge region rather than landing on or off the sharp edge. Thus, they will keep their initial

color, which is interpolated and blurry. Section 3.3 demonstrates this problem and describes our attempts of solving it.

3.2.2 Bilateral Roundup Filtering

Bilateral filtering is widely used for edge-preserving image noise reduction [9]. The idea is weighing each mask pixel by taking both color-space distance and Euclidian distance into account, as shown in Equation (3.2)

$$W = \alpha \times |I(p) - I(c)| + |X(p) - X(c)| \ , \tag{3.2}$$

where $|I(p) - I(c)|$ represents the color space distance between one pixel p in the sampling region and the mask center c. The value $|X(p) - X(c)|$ represents their Euclidian distance.

Our research introduces a variation of bilateral filtering: *Bilateral Roundup*. Roundup means we gather and select only the good ones. In other words, instead of taking all the pixels in a sample region into account, we sort the pixels by their bilateral weight and take only the top n (mask size) pixels as our mask. The process is as follows.

For each pixel c from the input image, for mask size n, we take a square-shaped sampling region M of that is much larger than n, (e.g. $8n$ to $10n$), centered at c. For each pixel p in M, calculate its weight using the bilateral weight function, Equation (3.2). Finally, we sort the pixels in M, and take the top n pixels as the actual mask. Their average color is the output for pixel c. Figure 3.5 demonstrates the idea of bilateral roundup filtering.

In Equation (3.2), the constant α defines the relative importance of the color-space distance and image-plane distance. It influences the bilateral roundup filter significantly. When α gets close to zero, this filter will become a smoothing filter. The masks will become circles centered at mask centers. When α is large (e.g. $\alpha = 10$), the resulting mask will contain only neighbouring pixels that are similar in color, which yields clustering blocky artifacts. We empirically choose α from the range 0.1 to 0.2. The complexity of bilateral roundup filtering is $km \log(m)$ for a k-pixel input image with sampling region size of m. Note that bilateral roundup filter is about 5 to 10 times slower than cumulative range geodesic filter, which is caused by the increased sampling region size.

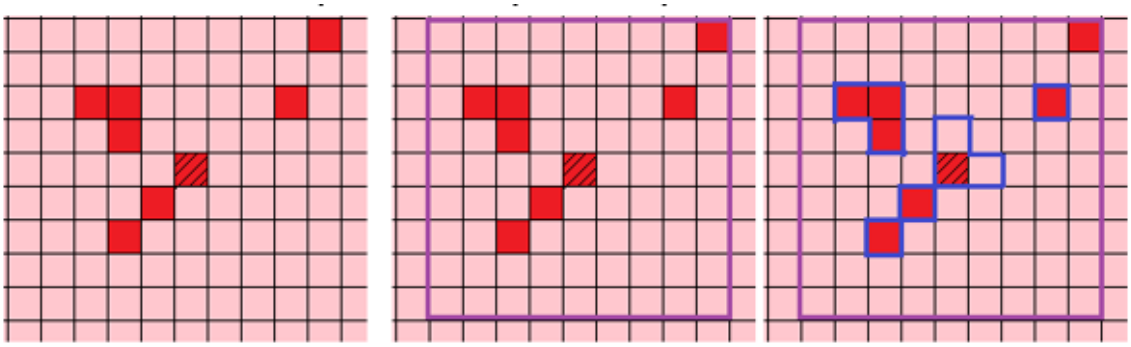

Figure 3.5: One example of bilateral roundup filter with mask size of 9, sample region size 81. Left: input. Middle: sampling region indicated by the purple boundary. Right: The output, indicated by the blue disconnected squares. Note that unlike cumulative range geodesic filter, here the output is not a connected set of pixels.

Bilateral roundup filtering inherits the edge-preserving feature from bilateral filtering. The color space difference term will enforce the similarity of mask pixels. Given a relatively large sampling region, the pixels on a sharp edge will get almost the exact same masks as each other. Thus, they will have the same output color, and the edge is preserved.

On sharp edges and large smooth regions, bilateral roundup filtering is equivalent to cumulative range geodesic filtering. They both tend to produce irregular-shaped, but continuous masks. However, comparing to cumulative range geodesic filter, bilateral roundup has the advantage of preserving fine details better, as shown in Figure 3.6. The reason is as follows. Cumulative range geodesic filtering relies on growing mask, so the mask must be a continuous set of pixels. When the mask size exceeds the number of neighboring similar pixels, the mask must cross an edge to keep growing, which will introduce pixels that are not necessarily similar to the starting pixel. However, bilateral roundup filtering samples a large region, and each pixel is weighed individually. Therefore for high-frequency regions, bilateral roundup filtering can extract more similar pixels, which grants better small-scale texture preservation. Figure 3.6 shows the difference between two filters on preserving fine details. On the left panel, cumulative range geodesic filter heavily reduced high frequency details, which leads to a clustering, quantized representation of the hair texture. Whereas for the bilateral roundup filter on the right panel, the fine details are better preserved.

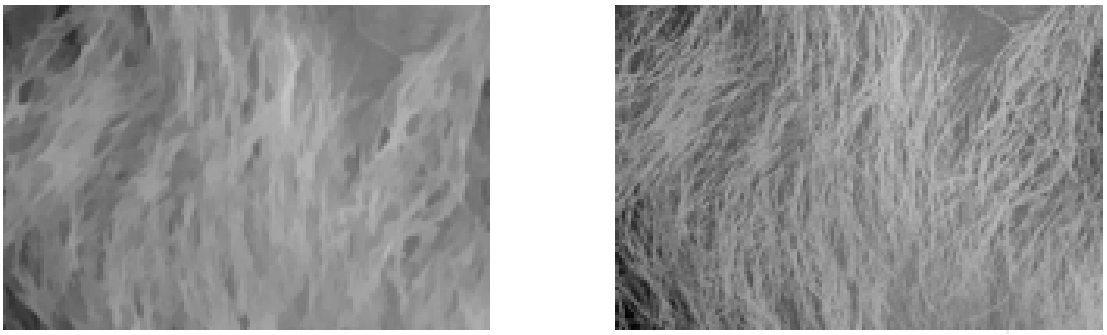

Figure 3.6: Left: Cumulative range geodesic. Right: Bilateral roundup. Bilateral roundup filtering preserves high-frequency, fine details better.

In the context of large-scale upsampling, bilateral roundup filtering has one limitation. The upsampling and interpolation process introduces a band of smooth and blurry pixels. If a bilateral roundup filter's mask center is one of these blurry pixels, the mask will include all other blurry pixels, which result in a blurry edge. In the next section, we present some variations of bilateral roundup filtering to fix this problem.

3.2.3 Variations of Bilateral Roundup Filtering

In order to overcome the blurry edge sampling problem, provide a better edge-sharpening effect, and introduce some high-frequency detail, in this section, we explore the following variations of bilateral roundup filter:

- Cross-filtering with original image to avoid sampling the interpolated blurry pixels.
- Multi-passes of bilateral roundup filtering to further sharpen the results.
- Pre-filtering the original image to increase the input image quality and alleviate jpeg artifacts.
- Post-filtering texture enrichment to introduce new arbitrary fine details.

Cross-filtering with original image

Large-scale upsampling will introduce a thick band of blurry pixels along sharp edges. Sampling mask data from these pixels will reduce the edge-sharpening effect of bilateral roundup filtering. Therefore, to avoid these blurry interpolated pixels, we explored the approach of sampling mask data on the original image rather than the

upsampled image. Figure 3.7 shows this pipeline.

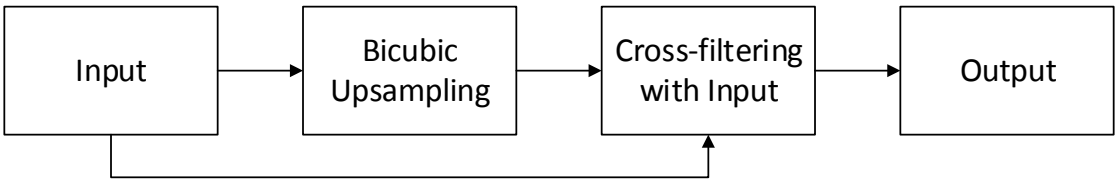

Figure 3.7: Pipeline of bilateral round up filtering with original image.

The cross-filtering process is as follows. In order to calculate the mask on original image, for each upsampled pixel, we use the interpolated color as mask center's color. We then use the interpolated pixel's fractional position on the original image plane as mask center coordinates. Finally, we apply bilateral roundup filter on the original image and use that output as upsampled pixel's color.

This approach has two advantages. First, the mask only samples pixels from the original image. The smooth, interpolated neighboring pixels do not exist on the original image, thus they will not be included in the mask, which helps the filter preserve sharp edges. Furthermore, compared to the upsampled image plane, the original image is much smaller. Therefore, growing masks on the original image usually requires a significantly smaller mask size, which greatly reduced the calculation size.

Figure 3.8 demonstrates the improvement of cross-filtering with original image. Note that the silhouette of the tree trunk is better preserved. However, the edge sharpness is not consistent along the whole silhouette. Certain segments of the edge are still blurry.

Multi-pass bilateral roundup filtering

As observed in Figure 3.8, the edge sharpness is not consistent along the whole silhouette. Certain segments of the edge can be further sharpened. Therefore, we recommend running several passes of bilateral roundup filtering on the previous results. The approach is simple: first use interpolated color and fractional position to filter the upsampled image (as described at Section 3.2.3); then iteratively apply bilateral roundup filtering (as described at Section 3.2.2) on previous passes' results. Figure 3.9 shows the pipeline of this process.

Figure 3.8: The improvement from cross-filtering with original image. Left: bilateral roundup on the bicubic upsampled image. Right: bilateral roundup on the original image. Note the inconsistent sharpness along the silhouette. Circled regions are less sharp.

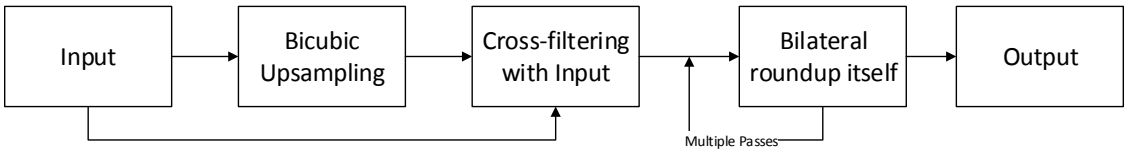

Figure 3.9: Pipeline of bilateral roundup filtering multi-passes.

More passes generate images that are more abstract: sharper edges, and less small-scale details. Figure 3.10 shows results ranging from 0 passes to 3 passes. We think the 2nd-pass result is the best among them. Its edges are significantly sharper than 1st pass, and the background (forest) is not as destroyed as the 3rd pass. It has a good balance between sharpening edges and preserving texture. Figure 3.11 shows another 2nd-pass result. We observe strong edges on boundaries of the picnic table, and the medium-scale textures on grass are preserved. Also, note that in the 3rd-pass result in Figure 3.10, some edges are a bit jagged from being over-sharpened. Therefore, we do not recommend running too many passes of bilateral roundup filter, unless the requirement of abstraction or sharpness is paramount.

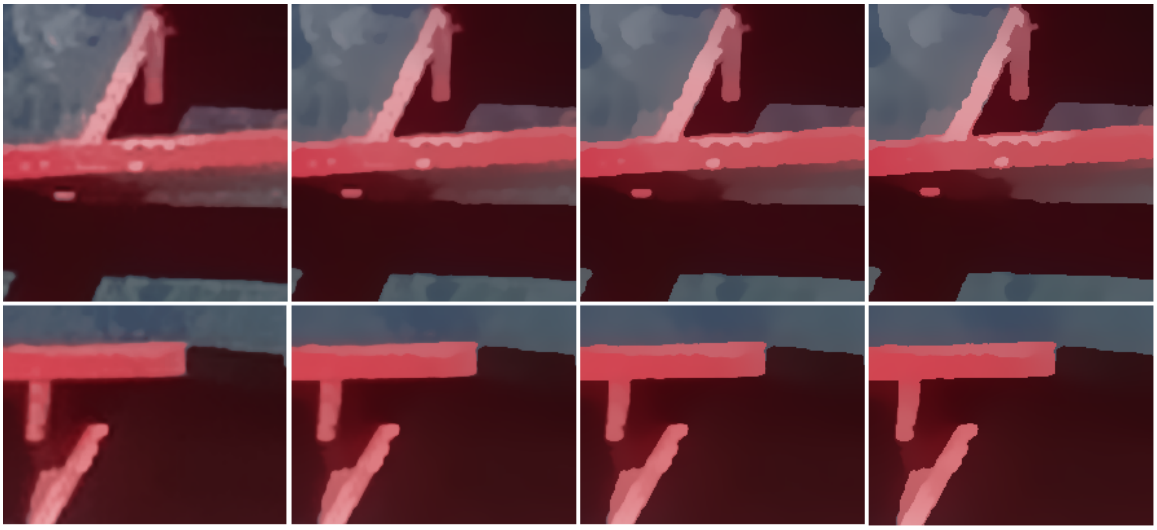

Figure 3.10: Multiple-passes results. From left to right: 0-pass, 1st-pass, 2nd-pass, 3rd-pass. More passes of this filter will keep sharpening the edges and eliminating small scale details.

Pre-filtering the original image

The process of cross-filtering with the original image suffers from low quality input. If the original image is smooth, has jagged edges, or suffers from jpeg artifacts, we cannot expect the upsampled images to be sharp and appealing. Thus, we recommend pre-sharpening the original image. We apply bilateral roundup filtering on the original image using a relatively small mask size (in the range of 5-7 pixels). This process will sharpen the original image a bit before upsampling. Though not obvious on most

Figure 3.11: A 2nd-pass result. Edges are well sharpened while the fine details are not overly removed.

images, this extra step improves the results of low-quality input images, by removing jpeg artifacts, and pre-sharpen blurry edges. Figure 3.12 shows the pipeline of this process. As shown in Figure 3.13, certain edges (marked by the red box) are slightly better constructed by pre-filtering the original image.

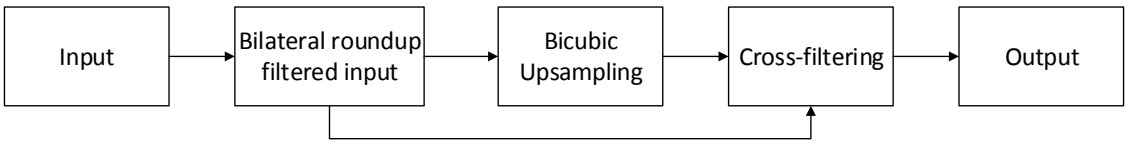

Figure 3.12: Pre-filtering pipeline.

Post-filtering texture enrichment

The upsampling process, cumulative range geodesic filtering and multiple passes of bilateral roundup filtering all reduce the high-frequency, fine details. Even though the results look clean and sharp, we often find the large smooth areas (like the sky) boring, and the areas that are expected to be textured (like grass or sand) are not textured enough. Since our goal is also to produce upsampled images with some high-frequency details, here we present a process for introducing arbitrary fine detail to the sharpened images. Note that we do not intend to introduce textures that match the textures from the input. To find matching textures we would need the following: first, understand the texture from the input, which is often difficult without

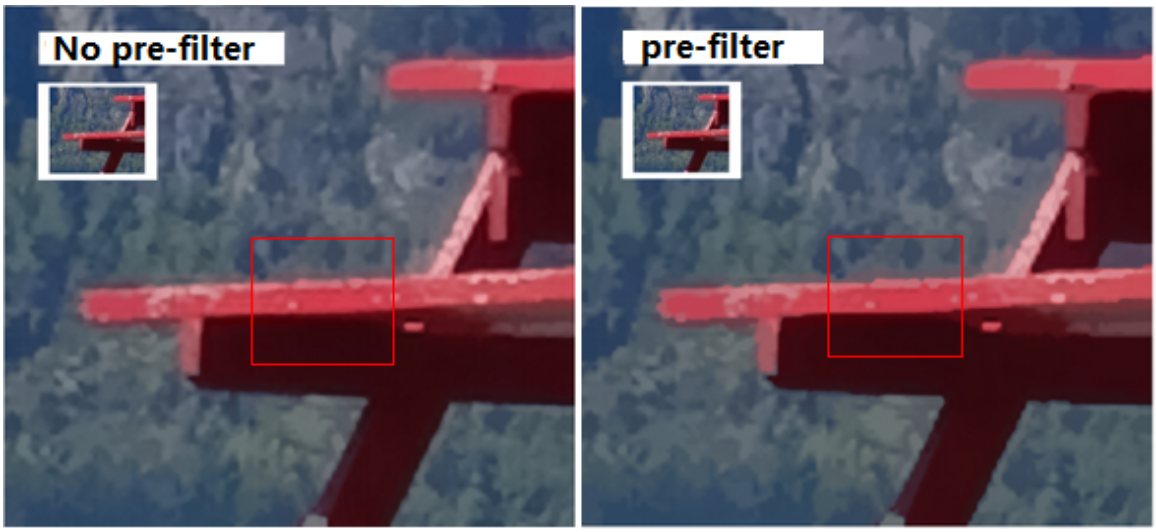

Figure 3.13: Pre-filtering original image improves poor edges.

human involvement; second, maintaining a huge texture database to satisfy different input images. In the future, we may explore the possibility of using example-based techniques to recover matching textures from the input. However, at the moment, we only propose this naive approach that adds some arbitrary high-frequency detail.

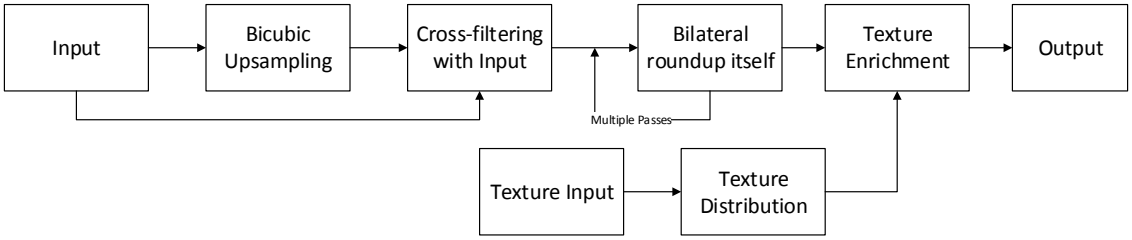

Figure 3.14: Post-filtering texture enrichment pipeline.

Figure 3.14 shows the pipeline of this post-filtering texture enrichment process. First we gather several high-frequency texture images from online image databases, using keywords like grass, gravel, hair, hay, fur, paper, or cement. Then based on the selected textures and the input pixel intensity, we calculate the resulting intensity change as follows.

We use $\Delta I(\vec{x})$ to denote the resulting intensity change of a pixel located at $\vec{x}$ in the image plane. It is a weighted sum of the m input textures, calculated using Equation (3.3)

$$\Delta I(\vec{x}) = \sum_{i=1}^{m} (T_i(\vec{x}) \times \frac{\varphi_i(\vec{x})}{\sum_{i=1}^{m} \varphi_i(\vec{x})}) \ , \tag{3.3}$$

where i denotes the label of a texture, ranging from 1 to m. In our experimental results, we use 5 input textures, i.e., $m = 5$. The value $T_i(\vec{x})$ denotes the ith texture's texture value at location $\vec{x}$. For the pixel at $\vec{x}$, the contribution of each input texture, $\varphi_i(\vec{x})$, is seperately calculated using Equation (3.4), then normalized by the sum of contributions from all m textures.

$$\varphi_i(\vec{x}) = \begin{cases} 1 - \frac{|L_i - I(\vec{x})|}{R(m)} & \text{if } |L_i - I(\vec{x})| < R(m) \\ 0 & \text{if } |L_i - I(\vec{x})| > R(m) \end{cases} \ , \tag{3.4}$$

where $I(\vec{x})$ denotes the intensity value of the pixel at $\vec{x}$. The value L_i denotes the pre-defined intensity lookup value for the input textures, (in our example, i.e., 0, 64, 128, 192, and 255). The denominator, $R(m)$, represents the maximum *range of effectiveness* of each texture in intensity domain. It is calculated using Equation (3.5):

$$R(m) = \frac{256}{(m-1)} \times k \ , \tag{3.5}$$

The idea is we divide the intensity domain (i.e., from 0 to 255) into $(m - 1)$ segments, and each input texture would influence a range with a radius of k segments. The following experimental results are obtained using 5 input textures with $k = 2$.

Overall, the closer a pixel's intensity is to a texture's lookup intensity, L_i, the higher this ith texture contributes to $\Delta I(\vec{x})$. The weighted combination of all the textures, $\Delta I(\vec{x})$, signals how much the intensity of a pixel should increase or decrease. This increase or decrease is applied to all three channels of the pixel color. Figure 3.15 demonstrates an example texture distribution scheme with 5 input textures and each texture covers an intensity range of ± 128.

There are two reasons why we distribute the textures this way. First, we want different-colored regions to adopt different textures. For instance, the added texture on the background sky should be different from the foreground table. The varied texture will help exaggerate edges, which makes the results look sharper. Second, we want same-colored regions to adopt similar textures. Therefore, within one object, there will exist a consistency of texture. For instance, there should not exist huge

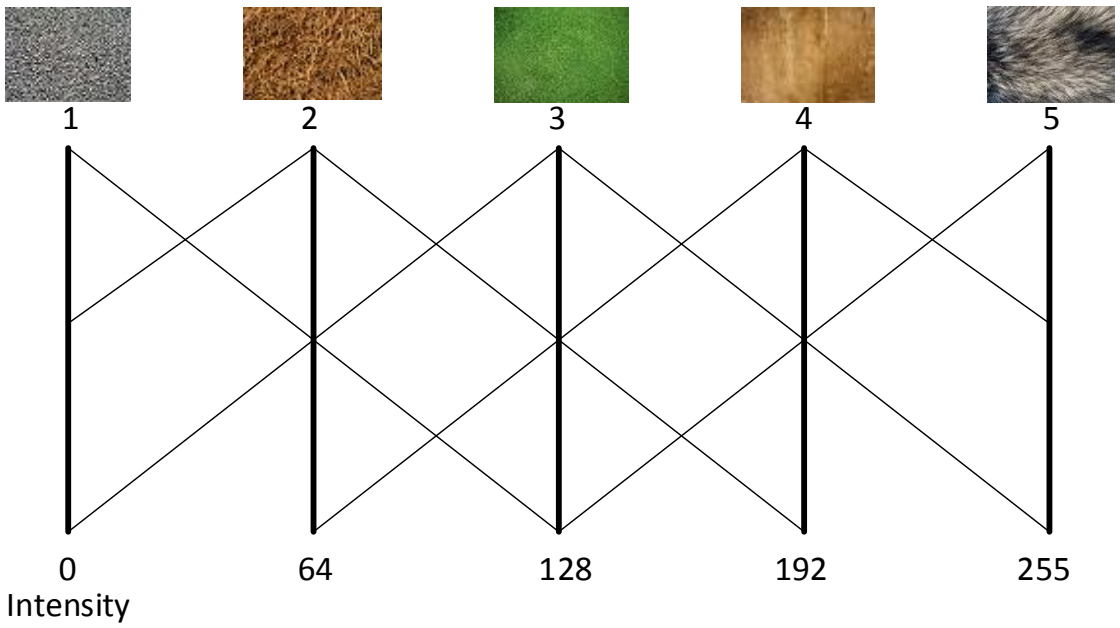

Figure 3.15: The texture distribution scheme. Note that the order of textures appearing on this illustration is for demonstration. The actual textures are ordered randomly.

variation of textures on a blue sky background.

Figure 3.16 shows one result stylized by texture enrichment. The table boundary remains sharp, while some textures are added to the table sides. The texture resembles paint brushes or the typical drying marks when paintings age. The texture of the grass ground and the forest-covered mountain are also improved. Overall, this texture enrichment process adds flavor and a stylized look to the upsampled image.

3.3 Notes towards Mask Data Processing

As discussed in Section 3.1, image filtering usually involves both gathering and processing pixel neighbourhood data. The filters described in previous sections focus heavily on data gathering, that is, we collect pixel neighbourhood information via irregular-shaped masks. Unfortunately, the way we process the mask data (averaging pixel colors) has been rather primitive.

In this section, we discuss our attempts towards mask data analysis and processing. We present our idea of penalizing the smoothly interpolated samples along edges

Figure 3.16: Adding multiple textures to upsampled image.

during mask growth or mask output calculation. Having the smoothly interpolated pixels removed or penalized, we hope that the edges can be better reconstructed. Even though the idea has several limitations, we believe it is worthwhile to note down our discoveries. We hope it will be a starting point for future research on this topic.

Edge Penalization

As we upsample an image by a large scale (e.g. 10 times), a sharp edge will become a band of smoothly interpolated pixels. When we sample these pixels, the mask is likely to grow along this band, due to their similarity in color and the closeness in Euclidean distance. If we average the pixel colors of this mask, the result would be as blurry as bicubic upsampling. Figure 3.17 demonstrates this problem.

Figure 3.17: Problematic masks along interpolated edges. Left: Cumulative range geodesic. Right: Bilateral roundup. Note that the cluster of disconnected pixels must have very similar color as the mask center.

In previous sections, we tried to address this problem by sampling pixels on the

original image, which avoids using the interpolated colors totally. Here we approach the problem differently. Since the problematic region lies along object boundaries, we can use edge detection filters to identify pixels along object boundaries, then penalize or remove these pixels either during mask growth or mask data processing. Figure 3.18 demonstrates our edge penalizing idea. Ideally, we would be able to identify the edges, avoid using the pixels along the edge, and most importantly, the remainder of the mask data would all fall into one side of the edge (either all in black or all in white in Figure 3.18).

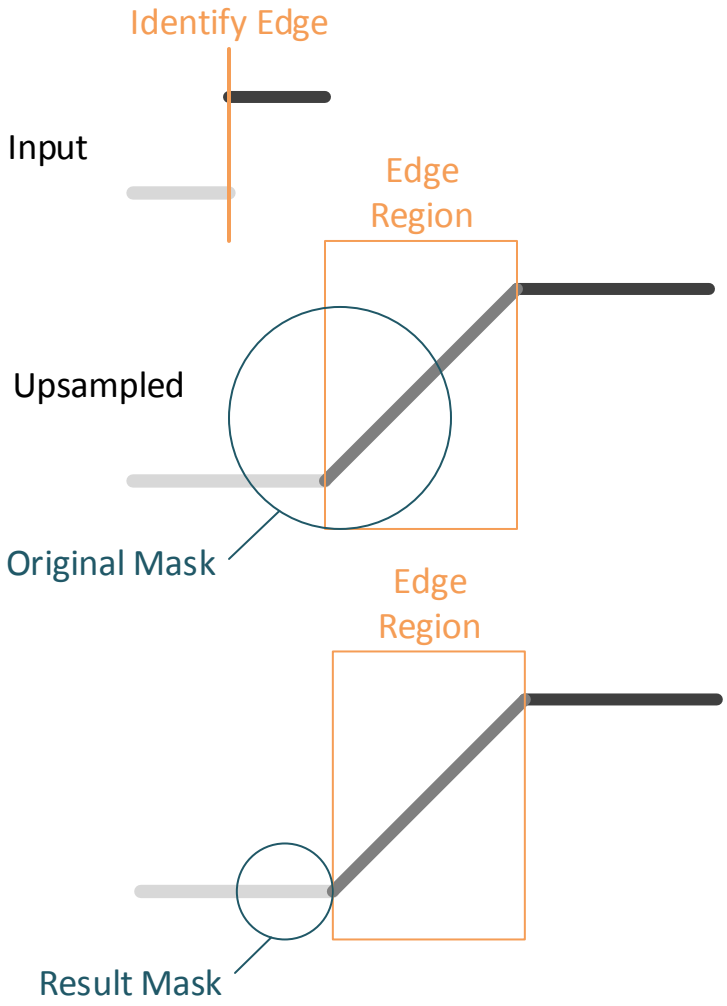

Figure 3.18: By removing pixels fall in the detected edge region, we are able sample mask data better.

We modify bilateral roundup filter's weight function, Equation (3.2), to Equation (3.6).

$$W = \alpha \times |I(p) - I(c)| + |X(p) - X(c)| + \gamma \times G(p) \ , \tag{3.6}$$

We add an extra edge penalizing term: $\gamma \times G(p)$, where $G(p)$ denotes the gradient magnitude of pixel p, which is calculated using a Sobel operator [15, p. 271-273] (Further described in Section 4.2.2). The edge penalizing term greatly increases the weight of blurry interpolated pixels on sharp edges, so they are less likely to be included in the output mask, thus the edge sharpness would remain. The parameter γ scales the edge penalizing term. Figure 3.19 shows the effect of this edge penalizing variation. As we may notice, with the edge-penalizing term, the tree boundary is slightly sharper.

Figure 3.19: The effect of edge penalizing. Left: bilateral roundup filter. Right: bilateral roundup filter with edge penalizing term. The marked region is slightly sharper.

However, this edge-penalizing approach sometimes fails. For instance, as shown in Figure 3.20, if the mask is large enough to cover both sides of the edge, despite penalizing the interpolated edge pixels, the remainder does not have one dominant color. Also, as shown in Figure 3.21, for long thin features, such as a hair, penalizing the edge may result in excluding all the pixels on the hair, which eventually will eliminate the hair feature.

Overall, in most cases, edge penalizing would locally solve the problem. But we

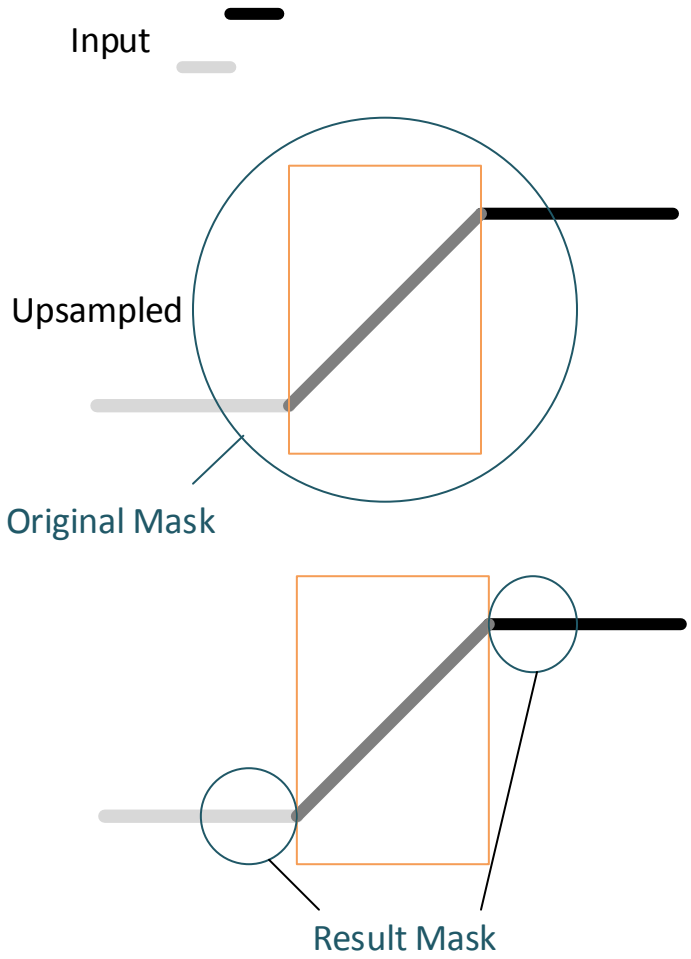

Figure 3.20: With masks that covers both sides of an edge, penalizing just the edge pixels would result in a mask that still does not have a dominant color.

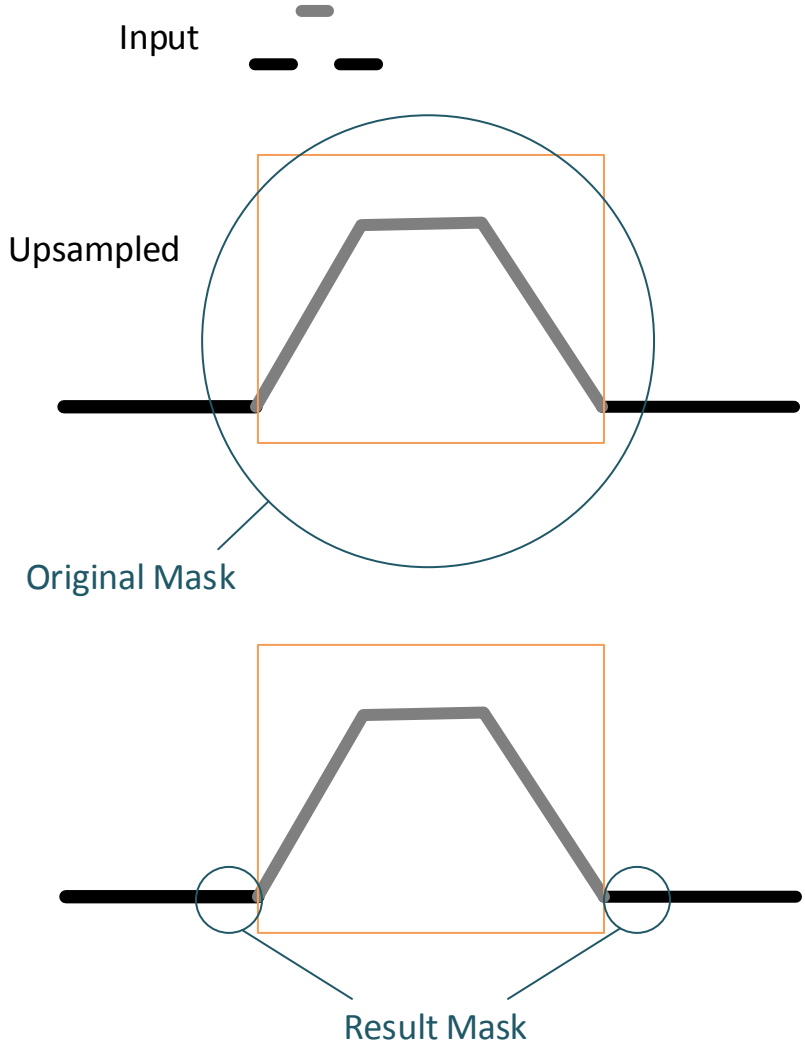

Figure 3.21: If the mask covers both sides and a thin feature itself, penalizing the edge means penalize the thin feature, which will result in losing the thin feature.

were unable to create a scheme or find out a set of parameters that universally applies. Figure 3.22 shows one result of the edge penalizing approach. Certain edges, such as the silhouette of the wing, are well preserved. However, some long, thin features, such as the front legs, disappeared. Nevertheless, we think the idea of identifying edges and treat pixels along the edge differently is worth investigating. In a more general sense, the idea of identifying special cases (such as edges, textures, and smooth areas) and treating those areas differently is worth investigating.

Figure 3.22: Edge penalize result. Dragonfly, 3 times upsampled. Top left: input at reduced resolution. Top right: full result image at reduced resolution. Bottom left: silhouette of the wing is well preserved. Bottom right: Thin features such as the legs are destroyed.

3.4 Results

Figure 3.23 to 3.26 show results of cumulative range geodesic filtering. The filter excels at edge sharpening. Figure 3.23 shows how well the edges are preserved. The edge of the tree trunk and leaves are as clear-cut as the original input. Moreover, middle-scale, irregular structures and shapes (e.g. the rock surface in Figure 3.24, the water surface in Figure 3.25) are also reconstructed. Overall the filter brings an abstract look to the upsample images. This effect resembles hand-drawn paintings. However,

this filter has the significant drawback of being unable to preserve fine-scale details. Figure 3.26 shows that the high-frequency textures on the brick surface are removed. We can argue that overall this approach generates abstract upsampled images, which naturally look non-photorealistic and somewhat arty. But over-reducing details makes certain areas of the image look boring, dull or counter-intuitive (such as the smooth surface of rocks or bricks).

Figure 3.23: Left: input at reduced resolution. Top row: Bicubic upsampled. Bottom row: Cumulative Ranged Geodesic. Sharp edges are better constructed.

Figure 3.27 shows a 2nd-pass, 3-times upsampled result for the dragonfly image. It is achieved by following the pipeline described in Section 3.2.3. Overall the image looks sharp and abstract. The silhouette of the dragonfly is well preserved. But there are many defects in the image. Thin features such as the front legs are quite damaged. Certain regions with smooth intensity transitions, such as the head, are quantized to a few hasty transitions. Also, there are aliasing effects along some edges. Though we are not satisfied with these artifacts, we think our texture enrichment process can

Figure 3.24: Left: input at reduced resolution. Top: Bicubic upsampled. Bottom: Cumulative Ranged Geodesic. Middle-scale irregular structures are preserved.

Figure 3.25: Left: input at reduced resolution. Top: Bicubic upsampled. Bottom: Cumulative Ranged Geodesic. Middle-scale structures are preserved.

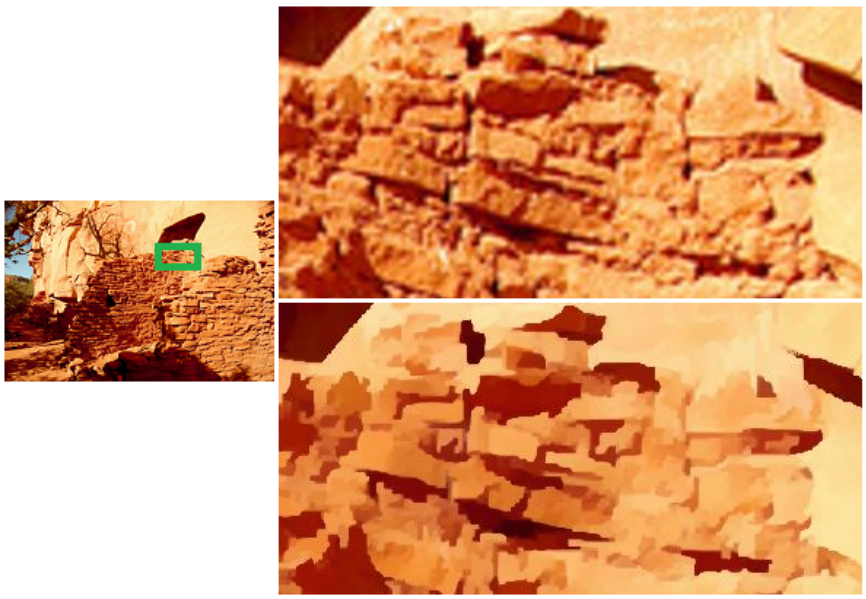

Figure 3.26: Left: input at reduced resolution. Top: Bicubic upsampled. Bottom: Cumulative Ranged Geodesic. Fine textures of the bricks are lost.

alleviate some of these problems.

Figure 3.28 shows our texture enrichment result, accompanied by the same texture applied to a bicubic upsampled image. We observe that the enrichment process significantly increased the amount of fine details. The quantization defect from over-sharpening is almost cured. The aliasing effect is also largely alleviated. The problem on thin features still exists, but that is destroyed by filtering process not the texture enrichment. Comparing with the result of adding textures to bicubic upsampled image, we have a significant advantage on edge sharpening. It is especially obvious along object silhouettes, such as the dragonfly wings. Also, the bicubic upsampled result looks very noisy. Our result is less noisy because the multiple filtering passes already removed all fine details. Thus, the amount of extra fine details that is ideal for us, would be excessive for bicubic upsampling.

Figure 3.29 shows the result of varying the textures input and their respective strength. Different combinations and choices of textures may bring very different looks to the results, which demonstrates the versatility of our approach, even though hand-picking and tuning the input textures is a bit laborious. Figure 3.30 demonstrates one result that scaled up all the added textures to 10 times as shown in other results.

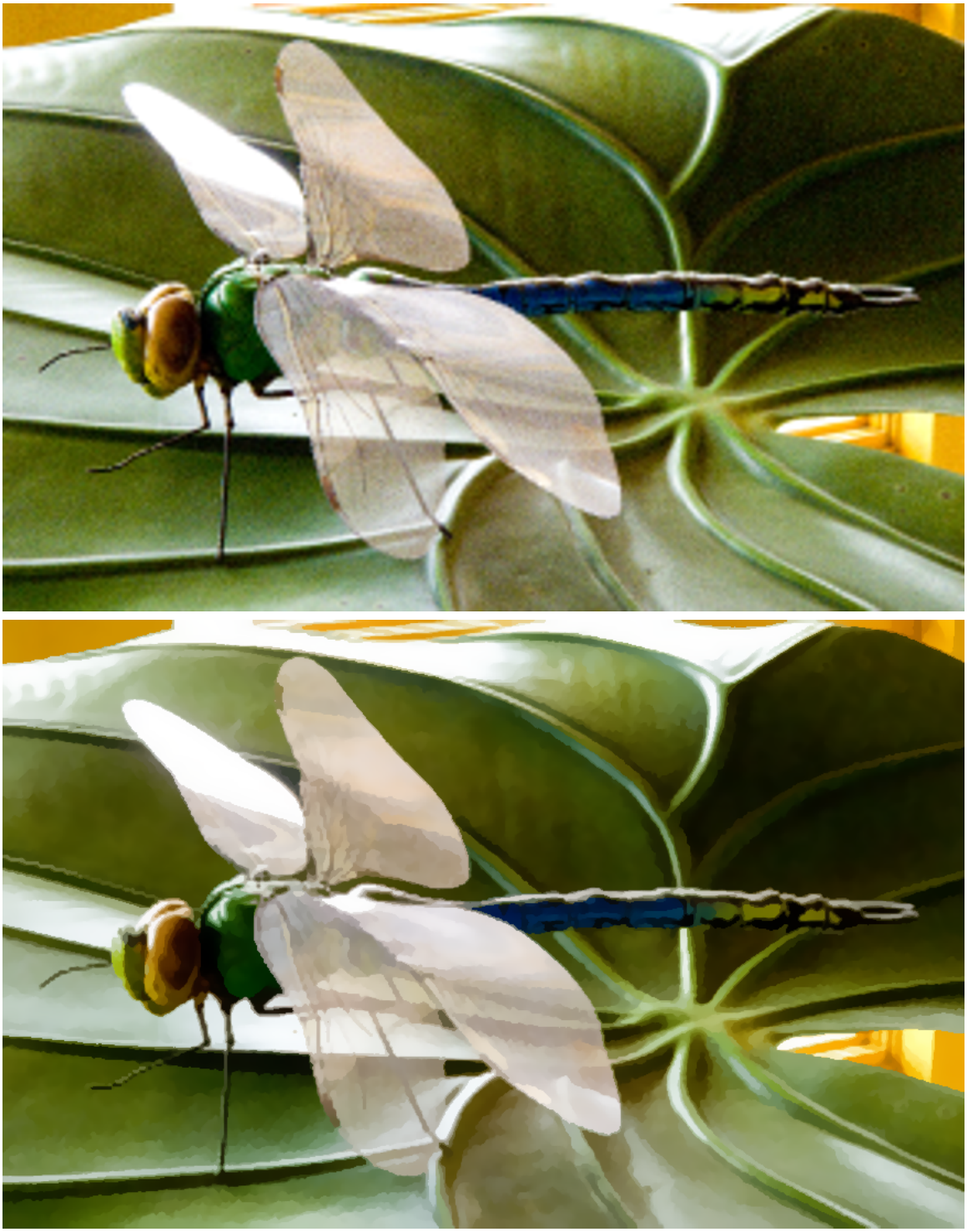

Figure 3.27: Dragonfly: 3-times upsampled. Top: Bicubic upsampled. Bottom: proposed multi-pass approach.

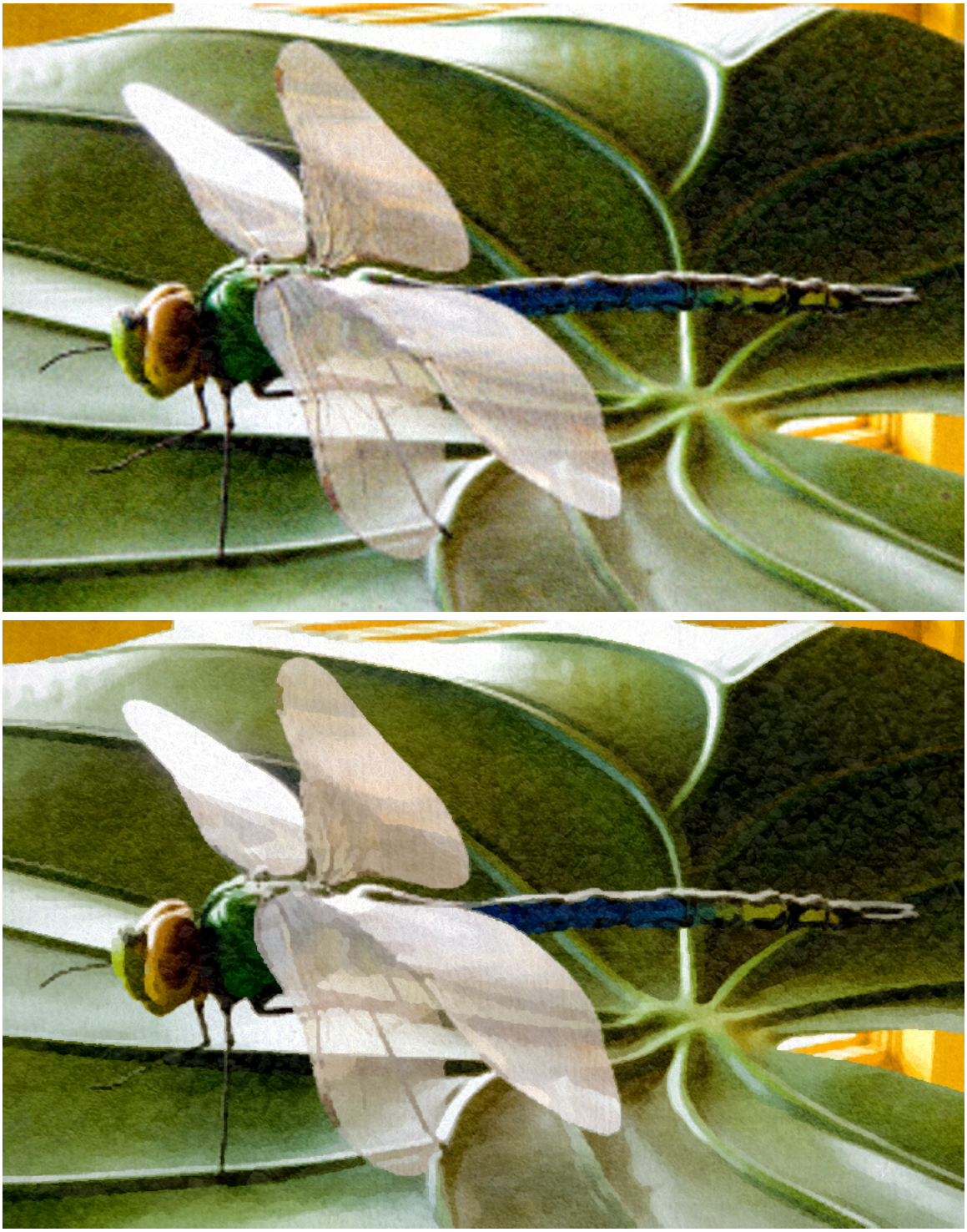

Figure 3.28: Texture Enrichment result. Dragonfly: 3-times upsampled. Top: Bicubic upsampled. Bottom: proposed multi-pass upsampled. Same amount of texture added to both images.

It is not appealing because the added detail is so strong that it destroyed the shapes and intensities of the input image.

We observed some similarities between our bilateral roundup filter and median filter. Figure 3.31 shows a comparison between the two filters. Both filters, to some degree, sharpen edges and reduce noises, with resulting images both exhibit an abstract look. However, even though both filters remove fine details, bilateral roundup damages fine details less. The white paint along the rim of the table, and the varied colors of the foreground grass are much better preserved by bilateral roundup filter. In the context of upsampling, we find our bilateral roundup solution already destroys a lot of fine details. Median filtering, which removes even more fine details, would definitely be excessive.

3.5 Conclusion

In this chapter, we presented a new perspective towards image upsampling. We believe by relaxing the faithfulness to input and aiming for stylized looks, we can simplify the problem yet still produce aesthetically pleasing results.

We proposed filtering-based processes to produce sharp, stylized super-resolution images. Both cumulative range geodesic filtering, bilateral roundup filtering and their variations have edge-preserving features. Cumulative range geodesic filtering gives strong edge-preserving effect, while sacrificing small-scale details. Applying multiple passes of bilateral roundup filtering has a good balance between preserving large-scale and small-scale features, but the processes are about 8 to 10 times slower. We also recommended a final touch to the upsampled images: texture enrichment. With the help from other texture images, this last step adds fine details that are removed by previous filtering processes. It makes the results look more stylized and interesting.

Both cumulative range geodesic filter and bilateral filter take the average of mask pixels as output. We find that rather naive and primitive. We hope that a better analysis of mask data will help us design more sophisticated schemes on mask data processing. We investigated the idea of differentiating special regions (such as the edge) from other regions. We hope that with the information of these regions, we can design smarter ways of sampling mask data, or calculating mask output.

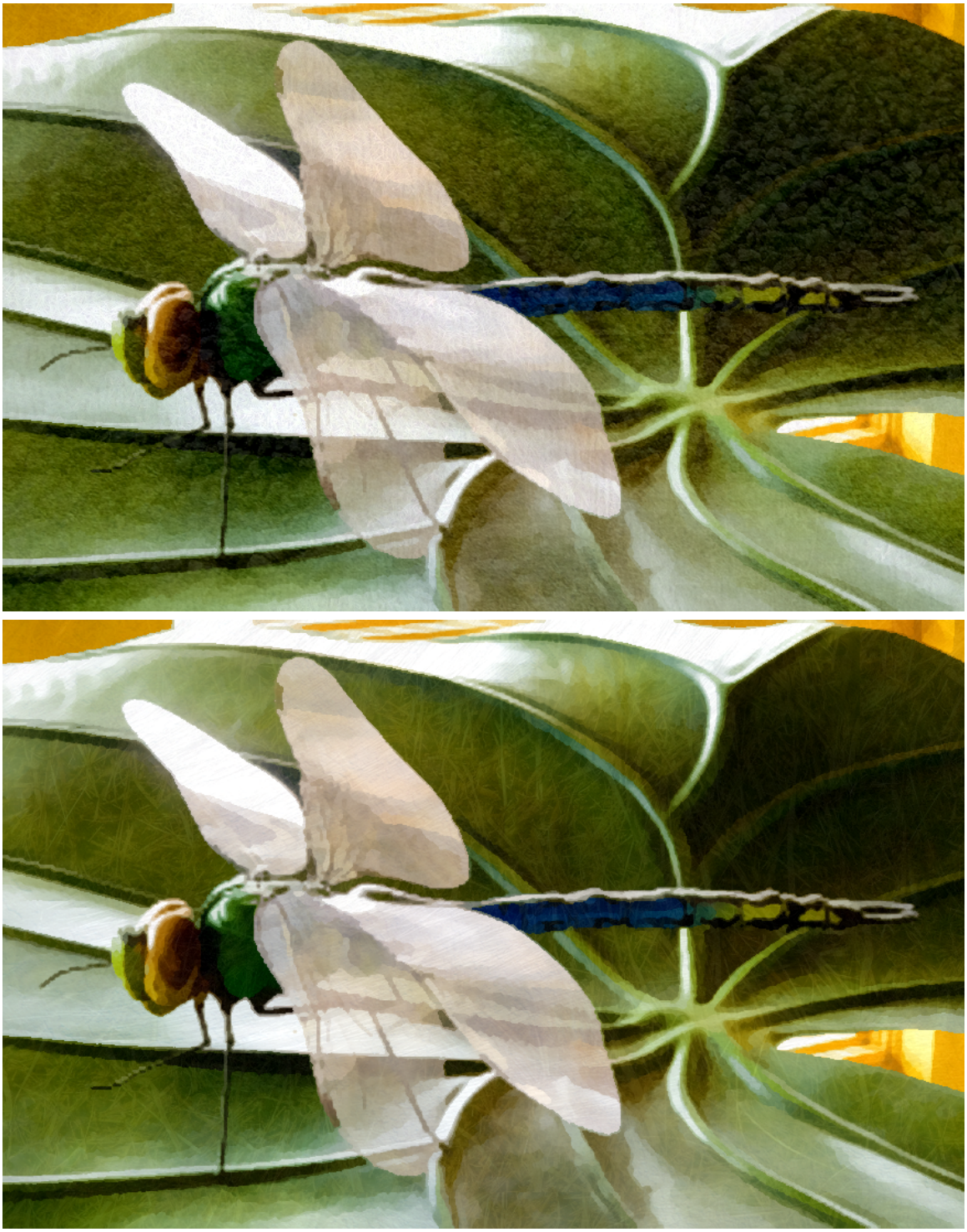

Figure 3.29: The result of varying texture input. Top: cement, plate, grass, gravel, towel. Bottom: hair, hay, fur, canvas, paint.

Figure 3.30: Extremely strong texture added. Top: proposed texture enrichment approach. Bottom: 10 times the same texture applied.

Figure 3.31: Comparison between bilateral roundup filter and median filter. Top: bilateral roundup. Bottom: median filter with a 5x5 mask.

Our current experimental results are based on upsampling scales from 3 to 6. As the scale of upsampling increases, the memory requirement and computation time will become very demanding. One focus in the future would be optimizing the implementation and accelerating the computations. Thus we can carry out the experiments more efficiently. We should also explore the possibility of bringing in other styles to upsampled images. It will be nice to have a variety of styles for the users to choose.

Chapter 4

Pixel Migration

4.1 Introduction

The last chapter described filtering-based solutions to image upsampling, which aim to produce stylized high-resolution images. In this chapter, we simplify the image upsampling problem from a different angle: we focus more on sharp edge preservation and relax our expectation on the quality of textured areas. We present one fast and effective approach to produce sharp upsampled images.

Figure 4.1 shows the treebark image that is 6-times upsampled using bicubic upsampling. Overall the image looks blurry. Thus, we proposed filtering-based approaches that apply to *all* the pixels, such that all the areas of an image is equally filtered and sharpened.

This time, we want to examine different areas of the image and approach them differently. First we look at the silhouette of the tree trunk. It is the sharpest edge of the input image, and it exhibits a large intensity jump (from white to dark gray). This jump is destroyed by the interpolation and has become a series of slow, smooth intensity changes (from white to light gray, to gray, then to dark gray). Losing the intensity jumps means losing salient features (i.e. sharp edges and object boundaries), which contributes a huge image quality loss. On the other hand, when we observe the textured areas of the input, such as the tree bark, we no longer see huge intensity jumps. Instead, we find many small but frequent intensity changes. These intensity changes may be strong locally, but for the whole textured area, they are smaller and less noticeable than the silhouette. Obviously these small intensity changes are smoothed out by upsampling as well. However, compared to losing sharp edges, the quality loss on textured areas are less severe, and we want to argue that this type of

quality loss is somewhat acceptable.

Figure 4.1: 6-times bicubic upsampled. Red: Sharp edges are terribly destroyed. Yellow: Textured areas are somewhat acceptable.

Silhouettes or sharp edges are usually formed by two rows of differently colored pixels, with one row on one side of the edge, and the other row on the other side. We can detect the location and orientation of an edge using edge-detection operators, such as Sobel operator [15, p. 271-273]. We can also quantify the edge-crossing intensity change using the gradient magnitude value. Thus, we think it is relatively easy to understand and describe sharp edges of the input image. On the other hand, textures are more complicated. Texture emerges from the behavior of an area of pixels. The frequency, magnitude, orientation and pattern of how pixels change in an area contribute to a texture. Though we can measure the texture intensity using textureness map proposed by Bae et al. [2], the pattern changes are difficult to understand and describe. Therefore, compared to textured areas, sharp edges are easier to detect and quantify. Thus, from a reconstruction standpoint, because we understand sharp

edges better, we believe we can recover sharp edges better.

This chapter presents our *Pixel Migration* solution to image upsampling, which is briefly demonstrated in Figure 4.2. Pixel migration refers to moving sample pixels on the image plane, so that the distribution of pixels is no longer even. From the input image, we want to identify pixels along sharp edges; and on the upsampled image plane, we want to move these pixels close to each other, such that the gaps along sharp edges are closed. Sharp edges are identified and quantified by applying various edge-detection filters; pixel relocation is achieved by a mass-spring physics simulation.

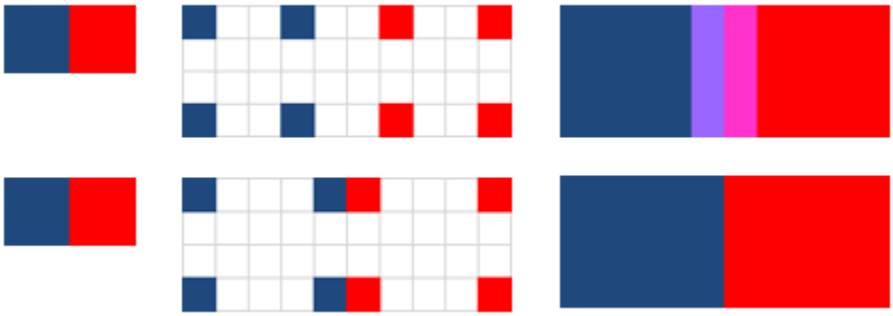

Figure 4.2: The difference between common interpolation and the proposed method. Top row: common interpolation. Bottow row: proposed method.

We make three major contributions in this chapter:

• We propose another perspective to image upsampling problem. We encourage designing algorithms that emphasis on edge sharpness preservation (which is easier but more important), while relaxing the expectation on textured areas (which is difficult but less important).

• We propose the idea of pixel migration. In the context of image upsampling, we want pixels along sharp edges to migrate closer to each other, while other regions remain unchanged.

• We present a mass-spring implementation of pixel migration. The mass-spring implementation produces large-scale upsampled images with sharp edges that are comparable to other state-of-the-art approaches.

The following sections are organized as follows. Section 4.2 describes in detail how we implemented the proposed approach. Section 4.3 demonstrates and discusses various results achieved by the mass-spring implementation. We conclude the chapter

in the last section with comments and directions for future research.

4.2 Algorithm

This section describes the implementation details of the proposed algorithm, including the pipeline of our processes, the edge-detection operators, the implementation of mass-spring system, and how we vary the spring parameters to achieve pixel migration.

4.2.1 Pipeline

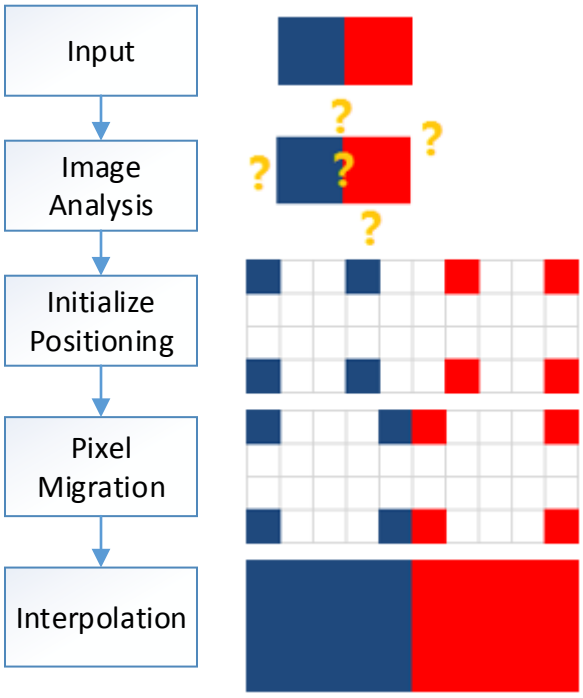

Figure 4.3: Pipeline. Left: The proposed pipeline. Right: A simple example that helps demonstrate the processes of each stage.

Figure 4.3 demonstrates the proposed pipeline. Given an input image, we first use edge-detection filters to extract edge information. We then create our upsampled image plane and evenly distribute the input pixel samples. Next, based on the edge information, we migrate the pixel samples in the sense that neighbouring pixels with

higher gradient tend to move closer to each other; and the orientations of neighbouring pixels tend to follow the edge orientations. We iterate this process until the forces within the mass-spring system are stabilized, then we barycentrically interpolate the deformed pixel lattice.

4.2.2 Edge Detection

Edges are locations in an image at which the color intensity changes sharply. In the following discussion, we use the notion of *standalone edge* to describe large-scale continuous edges, which are observed along object silhouettes. We focus less on high-frequency intensity changes, which are often observed on highly-textured areas. In the context of image upsampling, we believe it is easier and more important to identify and reconstruct standalone edges. We relax our expectation on recovering edges on high-frequency textures, because it is difficult and often the input does not have enough information for perfect reconstruction.

At the early stage of this research, we assumed the color intensity difference between two neighbouring pixels is enough information for standalone edge detection. The rationale was simple: if there exist a large intensity jump between two neighbouring pixels, there must exist a standalone edge. It seemed reasonable, especially considering that we use a mass-spring system that every pair of neighbouring pixels are connected by a spring. Unfortunately, we met two significant drawbacks. First, noisy areas, such as high-frequency textures, may contain strong local intensity changes too. Pixel pairs located in these areas may all have quite drastic intensity jumps, thus they will tend to move closer to each other. This will cause higher-frequency textured areas to shrink, and nearby lower-frequency textured areas to expand to compensate. Thus, a visible difference of texture will appear, or worse, an edge that did not exist in the input will appear. Another pitfall is that we did not consider the orientation of edges. The color intensity difference between a pair of pixels can only help us decide how much spring force we apply to the pixels, but it will not tell us how to orient the relative position between the two, so that they would align with the edge orientation.

Our goal is to shrink the springs along standalone edges and let other springs expand to compensate. Our observation is: if a spring crosses a standalone edge, then it will probably have one of the top gradient magnitudes around its neighbourhood. Thus, by identifying the locally highest gradient magnitudes, we can estimate the

location of sharp edges.

For an input image A, its gradient magnitude is calculated using the Sobel operator [15, p. 271-273] following Equation (4.1) to (4.3).

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \ , \tag{4.1}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A \ , \tag{4.2}$$

where * denotes the 2D convolution operation. The gradient magnitude M is then calculated using Equation (4.3):

$$M = \sqrt{|G_x|^2 + |G_y|^2} \tag{4.3}$$

We consider M to be a gradient magnitude field, from where we can query gradient magnitude of individual pixel or spring located at $\vec{x}$.

The local edge orientation is perpendicular to the local gradient direction. Therefore, to address the edge orientation problem, we want to measure the local gradient directions and orient the springs accordingly. A unit vector $\hat{G}(\vec{x})$ denotes the gradient direction of a spring located at $\vec{x}$, which is calculated as the normalized sum of $G_x(\vec{x})$ and $G_y(\vec{x})$. We introduced an angular force that orients a spring, so that it is perpendicular to the gradient direction $\hat{G}(\vec{x})$. Section 4.2.3 describes how we use $\hat{G}(\vec{x})$ to adjust spring orientations.

4.2.3 Mass-spring System

Initialization

We assume on the original image plane with n input pixels, the default spacing between adjacent pixels is 1 unit. Thus, for an upsampling scale of U, the upsampled image plane should have the size of U^2n pixels, with default pixel spacing of U units. We initialize the pixel locations with this same spacing and connect each pixel and

their four adjacent neighbours with springs. Figure 4.4 demonstrates how pixels are connected in the mass-spring system.

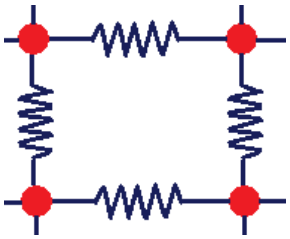

Figure 4.4: Mass-spring system. Each pixel is connected to its neighbours with a spring.

Spring variable assignment

Traditionally, spring force is calculated using Hooke's law (4.4):

$$F = k \times \Delta x \ , \tag{4.4}$$

where Δx denotes how much the ends of the spring are displaced from the relaxed position (also known as rest length); and k denotes the strength coefficient of the spring.

For an arbitrary spring s, the value $L(s)$ denotes the current spring length. We use $T(s)$ to represent the rest length (or the target length) of spring s. Therefore, the value $\Delta x = L(s) - T(s)$. We control the spring forces by varying their rest lengths $T(s)$ and their spring coefficient k. The mass-spring similation iteratively re-calculates and updates the current spring length $L(s)$.

Our goal is to shrink the springs along standalone edges and let other springs expand to compensate. Thus, we need to distinguish the springs appear on standalone edges or textured areas. Our observation is as follows. If a spring crosses a standalone edge, then it will probably have one of the top gradient magnitudes around its neighbourhood. We should give such springs the minimum rest length, and the strongest strength coefficient, so that they can shrink drastically. If a spring resides in a smooth area, it will probably have a gradient magnitude that is below average of the neighbourhood. We should assign such springs default rest length, and a much weaker spring coefficient, so that they can expand freely. For springs that are on neither standalone edges nor smooth areas, they may be on textured areas or on

locally less-significant edges. We want to sharpen these features, but not as much as the standalone edges. Therefore, we linearly interpolate their rest lengths based on their ranking, and use the same spring coefficient as the springs on sharp edges.

We introduce a ranking system that compares the gradient magnitude $M(s)$ of spring s with the gradient magnitudes of other springs in a 5×5 neighbourhood. Based on the ranking of s, denoted as $R(s)$, we assign its rest length and spring coefficient. The assignment scheme is displayed in Table 4.1.

Ranking: R(s)	Rest length: T(s)	Strength Coefficient: k
$R(s) < R_{\text{top}}$	T_{min}	K
$R_{\text{top}} < R(s) < R_{\text{mid}}$	$T_{\text{min}} + (U - T_{\text{min}}) \times (\frac{R(s)-R_{\text{top}}}{R_{\text{mid}}-R_{\text{top}}})$	K
$R_{\text{mid}} < R(s)$	U	K_{weak}

Table 4.1: Spring rest length and strength coefficient assignment.

The constant R_{top} and R_{mid} denote the ranking thresholds. We consider springs with higher ranking than R_{top} the standalone edge-crossing springs; and springs with lower ranking than R_{mid} the smooth-area springs. The shortest rest length, denoted as T_{min} is assigned to the standalone edge-crossing springs; whereas the default spacing U is assigned to smooth-area springs, accompanied by weak spring coefficients, denoted as K_{weak}. For springs with ranking between R_{top} and R_{mid}, we linearly interpolate their rest lengths between T_{min} and U, based on their ranking.

For our implementation, we empirically choose our parameters as follows: $R_{\text{top}} = 0.2$, $R_{\text{mid}} = 0.5$, $T_{\text{min}} = 0.1$, $K = 1$, $K_{\text{weak}} = 0.3$. We observe that increasing R_{top} will produce sharper but more distorted results. A reasonable range for R_{top} would be between 0.2 to 0.3. We recommend choosing T_{min} to be much smaller than the upsampling scale U, so that the two end pixels of an edge-crossing spring will move close to each other. Ideally, at the end of the simulation, they would be so close that they overlap. We empirically set $T_{\text{min}} = 0.1$. Depends on the metrics and implementation of forces and distances, default spring constant K may vary. However, the relationship between K and K_{weak} should remain similar to $K_{\text{weak}} = 0.3K$.

By combining Equation (4.4) and the spring parameter assignment table: Table 4.1, we calculate spring force using Equation (4.5):

$$F(s) = \begin{cases} (L(s) - T_{\text{min}}) \times K & \text{if } R(s) < R_{\text{top}} \\ (L(s) - (T_{\text{min}} + (U - T_{\text{min}}) \times (\frac{R(s) - R_{\text{top}}}{R_{\text{mid}} - R_{\text{top}}}))) \times K & \text{if } R_{\text{top}} < R(s) < R_{\text{mid}} \\ (L(s) - U) \times K_{\text{weak}} & \text{if } R_{\text{mid}} < R(s) \end{cases} \tag{4.5}$$

Angular springs

In order to reduce distortion along long, thin edges, we also want to orient the springs so that they are perpendicular to local gradient direction. Therefore, on top of the forces exerted by the connecting springs, we want to add angular forces to orient the pair of pixels. We use $\theta(s)$ to represent the current orientation of a spring s. We use $\theta(g)$ to represent the target edge orientation, which is perpendicular to $\hat{G}(\vec{x})$. The goal here is to adjust the current orientation $\theta(s)$ to match the target edge orientation $\theta(g)$. Figure 4.5 shows how we apply the force.

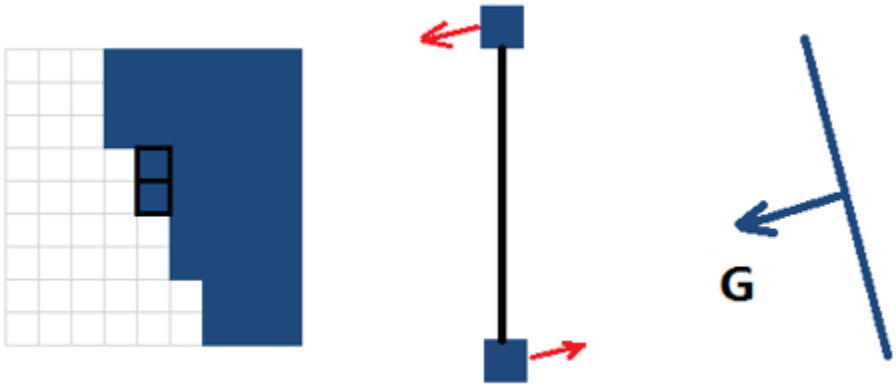

Figure 4.5: Angular Force. Left: two adjacent pixels along an edge. Middle: Their current orientation. Right: The local gradient, which signals the target orientation. Red arrows show the force directions.

We apply angular forces to both ends of a spring, using Equation (4.6):

$$\hat{F_a}(s) = \hat{G}(\vec{x}) \times (\theta(g) - \theta(s)) \ , \tag{4.6}$$

where unit vector $\hat{G}(\vec{x})$ gives the direction of the force. The angular force is scaled by the difference between current orientation $\theta(s)$ and the target orientation $\theta(g)$. Therefore, if there exists a huge difference between the two, the angular force will soon orient the springs to align with the edge direction. As shown in Figure 4.5, angular forces are applied to both end pixels of a spring. The sign of the force is the sign of $\theta(g) - \theta(s)$.

Finally we combine the connecting-spring's force (Equation 4.5) and angular force (Equation 4.6) together. We calculate one end pixel's force using Equation (4.7)

$$F = F(s) + \alpha \times \hat{F_a}(s) \ , \tag{4.7}$$

where α scales the angular force down. Note that angular forces need to be scaled to much lower than the connecting spring forces. The purpose of adding this angular spring term is to mildly adjust the orientation. Since it is also re-calculated and re-applied every iteration, frequently over-orient the springs will incorrectly jitter the pixel locations, which reduces the overall mass-spring system's stability. In our implementation with a connecting-spring coefficient $K = 1$, we empirically choose $\alpha = 0.1$.

Simulation and interpolation

We iteratively re-calculate and re-apply the forces within the system. For each pixel, the connecting-springs' forces and angular forces are combined. Acceleration is then calculated and pixel velocity is changed. We use the standard forward Euler integrator to calculate pixel's displacement and location. To enforce stability, we introduce damping force using Equation(4.8).

$$F_{damp} = -\mu \times v \tag{4.8}$$

where v denotes the velocity, and μ is the damping coefficient. In our implementation, we empirically choose $\mu = 1$ for strong springs and $\mu = 0.3$ for weak springs.

We terminate the system after 50 iterations, which is when the forces within the system are mostly balanced. Finally, given the deformed pixel lattice, we interpolate the unknown pixel values. We first triangulate each quadrilateral formed by 4 neighbouring pixels, then barycentrically interpolate each triangle.

The complexity of our proposed method is qU^2n for n-pixels input with upsampling scale U and q iterations of mass-spring simulation. Its extremely fast, because q and U^2 are small constant values. Our non-optimized CPU implementation takes about 5 seconds for 8-times upsampling a quarter-megapixel image.

4.3 Results

4.3.1 Pixel Migration results

Figure 4.6 to 4.12 demonstrate our results. We investigated the effect on images with varied subject matter and backgrounds, including portraits, still lifes, animals, plants, and cityscapes.

We achieved the goal of edge preservation. All of the results show strong sharp-edge preserving effect. Figure 4.6 and Figure 4.11 show the comparison between bicubic upsampling and the proposed method. In the duck image, the silhouette of the eye and the white stripes are perfectly reconstructed. The boundary between differently textured regions is clear. Overall the sharpened image looks more colorful and vivid. In the dragonfly image, in addition to edge sharpening, the proposed process also removed the blocky texture exhibited along the lower rim of the tail and the chin. Figure 4.12 shows how much better we preserve sharp edges than another state-of-the-art upsampling approach by Fattal [17]. Notice that for medium-sharp edges, such as the silhouette of the finger, we preserved the same level of sharpness as the input; Fattal's approach cannot identify the silhouette, thus failed in its recovery.

One advantage of our approach is that we can raise the upsampling scale and achieve the same degree of edge sharpness. This is achieved by setting the minimum spring rest length to 0.1 units. No matter the upsampling scale, the pixels along sharp edges are forced to remain about 0.1 units distance, which will cause them to cluster or overlap, thus reconstruct the same sharp edge as the input. Figure 4.6 shows how pixels are clustered along sharp edges. Note that other pixels from smooth regions are spread out to compensate the clustering.

Our approach is also effective in preserving middle-scale details. In Figure 4.7, we observe that middle-scale details, such as the surface and highlights on the corn, are very well preserved. In Figure 4.10, the spikes look sharp and dangerous; the varied texture on the leaf surface is visible. Note that the tip of the leaf shown in the mid-right panel is distinguishable from the gravel background. This is achieved by

retaining the same edge orientation as the input, which is enforced by including the angular spring forces.

The major disadvantage of our approach is the distortion. The distortion is especially noticeable along long, thin, or straight features, as shown in Figure 4.7, 4.11, 4.12. Because our rest length assignment is based on gradient, minor gradient changes along long features will disturb the local rest length ranking process. Also, a long feature may stretch across several regions of different textures. Distortions will appear on the segments where severe texture changes happen. The distortion is also very noticeable when there exist small to middle scale features that have unique geometric shapes. For instance, upsampled circle-like features may not look as round as the input; sharp corners of polygons may become smooth and roundish; text symbols and letters may be heavily warped to a degree that they are no longer recognizable. That said, we are not so concerned about the warped text. Text is a difficult problem to start with, and we do not usually find a lot of fine-scale text on photographs anyway. We are capable of preserving middle-scale text. They will be distorted, but still recognizable, as shown in 4.7.

Another disadvantage is the change in texture. We observe some texture change on low- to mid-frequency areas, as shown in Figure 4.8, 4.11, 4.12. In Figure 4.8, the pupil geometry is not as round as the input. Some extra textures are introduced to the baby's forehead. A slight quantization effect appears on the tip of the baby's nose. In Figure 4.9, there exist an inconsistency in the shapes and sizes of the windows. A strong quantization and distortion effect covered the vehicles. We find some of these changes add flavor to the upsampled image, such as the distorted vehicles that exhibit an impressionism style. However, some of the changes are less appealing, such as the introduced texture to the baby's forehead, since we expect the baby skin to be very smooth. We discovered that the change of texture is also caused by the gradient ranking system. For low-frequency areas, there usually do not exist any dominant gradients. Thus, additional edges may appear on the pixels that are incorrectly chosen to be the local maximum gradient. However, as described early in this chapter, our focus is on sharp edges rather than textured areas. We think the change of texture is somewhat tolerable. As shown in Figure 4.10, the typical texture caused by our approach resembles the texture of gravel and rock surface, which made our approach a great choice for upsampling images containing a lot of rocks.

Figure 4.12 shows a comparison between Fattal's and our work. At large-scale

upsampling (e.g., 8 times), Fattal's approach reconstructs relatively sharp edges, as shown on the rim of the can. However, our approach sharpens the feature better. Also, for edges that do not exhibit high sharpness from the input, such as the finger, Fattal's approach smoothed out the object boundary, whereas we reconstructed the same level of edge sharpness. One significant disadvantage of ours is the distortion, which is quite obvious on the area with text. Though the edges of the text are sharp, the distortion almost rendered the text unreadable.

Overall, our proposed method succeeds in sharp edge and medium-scale detail preservation. The results look sharp, colorful, and vivid. However, uneven pixel migration may cause the change of medium or fine scale geometry, which leads to significant change of textures. We hope that we can solve this problem by better understanding the nature of texture, and improve our current rest length distribution system; or come up with other methods to implement the pixel migration idea.

4.3.2 Comparison between filtering-based approach and pixel migration

Table 4.2 shows a comparison between filtering-based techniques from the last chapter and pixel migration from this chapter.

Overall, both approaches generate sharp upsampled results. Filtering-based techniques do not purposely distinguish object boundaries from other image content, whereas pixel migration focuses only on detected edges. Therefore, filtering-based results exhibit an overall clean look (before texture enrichment), whereas pixel migration generates extremely sharp edges along object boundaries with the rest of the image content less sharpened.

The filtering processes preserve large to middle-scale details, while removing most of the fine-scale details. We proposed a naive texture enrichment process that introduces fine-scale details, but the details introduced are arbitrary. The pixel migration approach does not remove fine-scale details. However, it often introduces some

Figure 4.6: Duck: 5-times upsampled. Top: input at reduced resolution. 2nd Top: showing how pixels are clustered along object bounadries. 3rd Top: bicubic upsampled. Bottom: Our result. Note that object boundaries (the eye and white stripes) are well preserved.

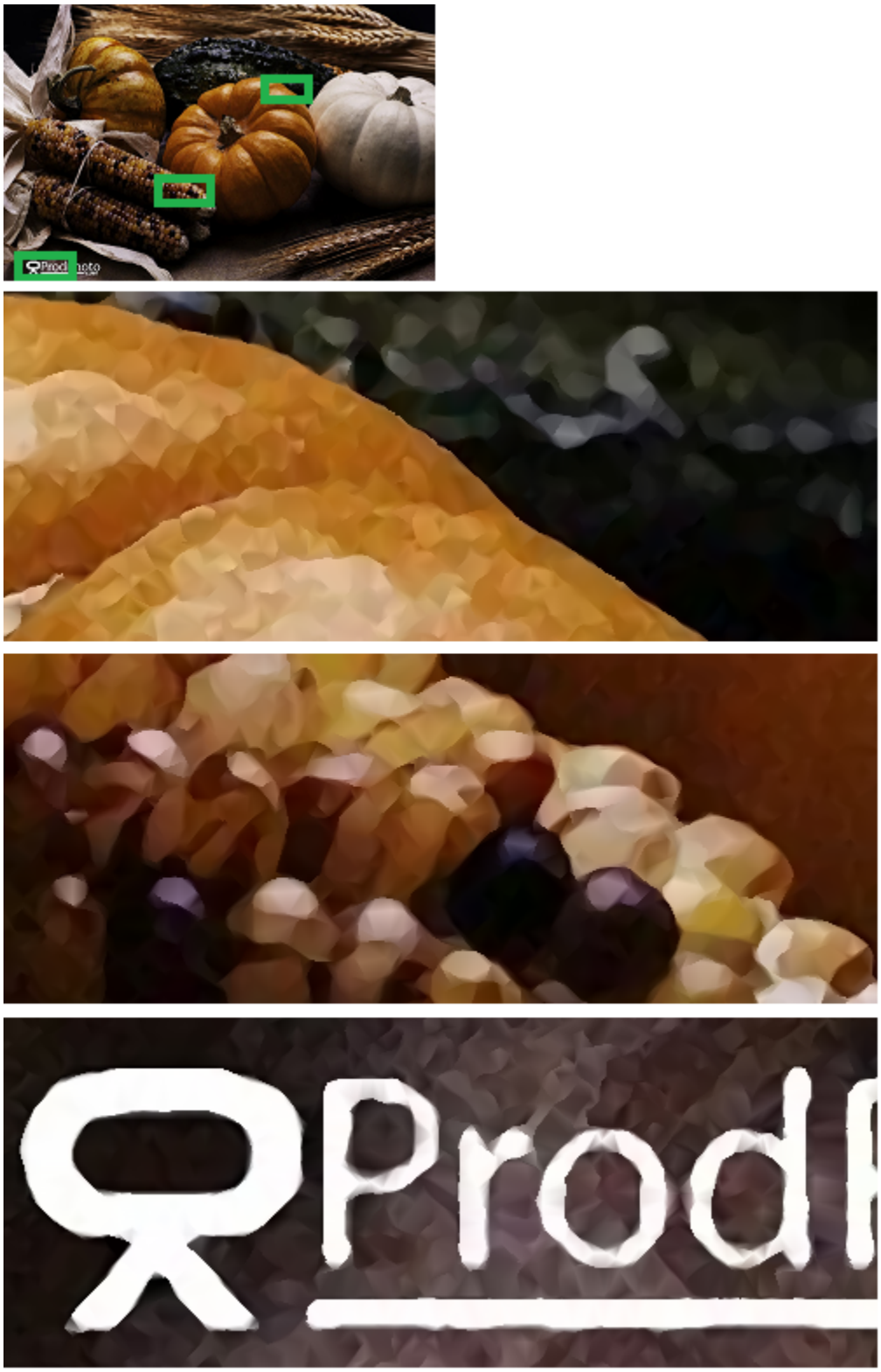

Figure 4.7: Harvest: 6-times upsampled. Top: input at reduced resolution. 2nd Top: Object boundary greatly sharpened. 3rd Top: Preservation of middle-scale details such as the highlights on corn. Bottom: Linear thin features (like text) are distorted.

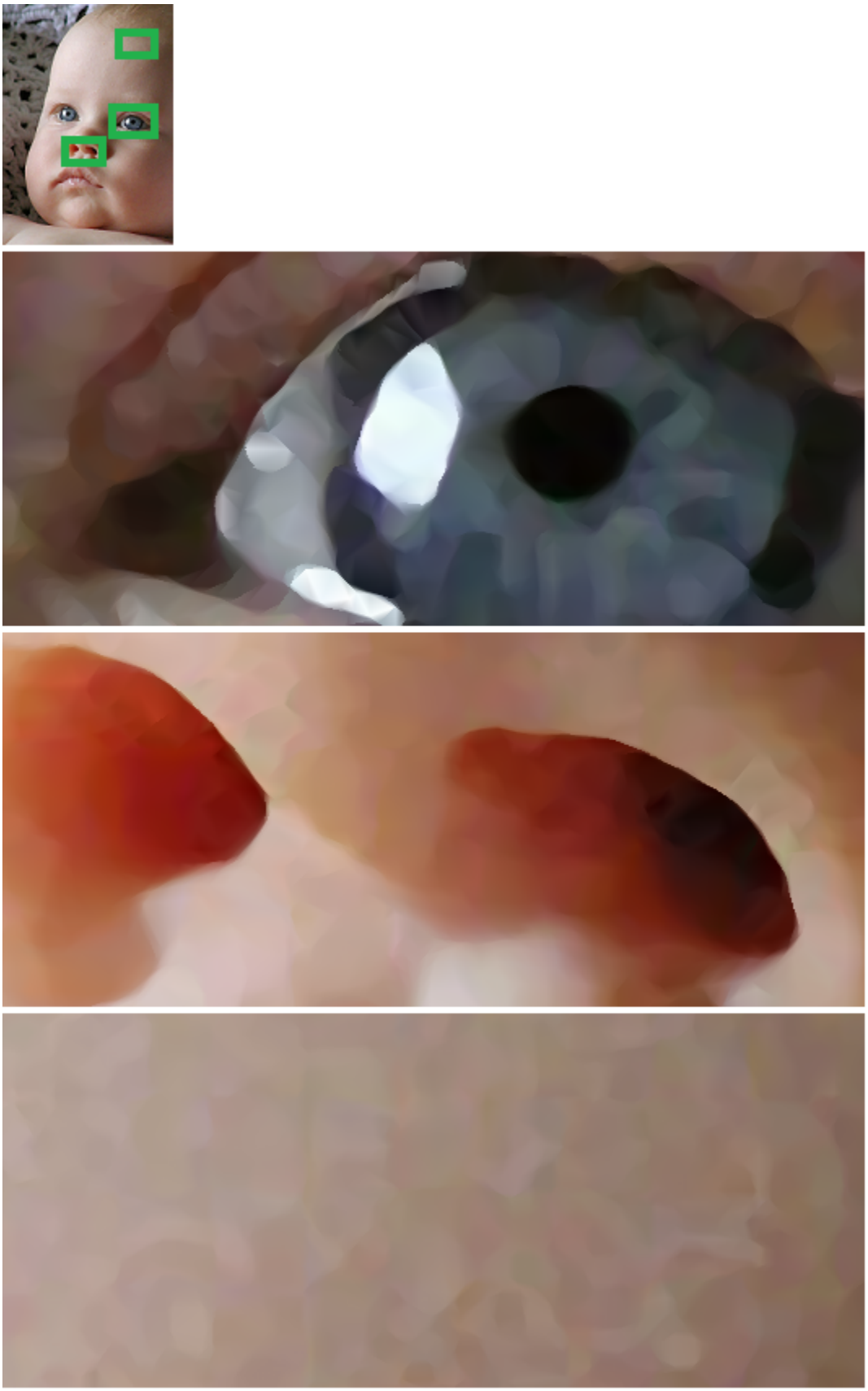

Figure 4.8: Baby: 6-times upsampled. Top: input at reduced resolution. 2nd Top: Highlight on the eye is sharpened. Pupil geometry is slightly changed. 3rd Top: Nose boundary sharpened. Bottom: Mild Voronoi-shaped textures are introduced to smooth areas.

Figure 4.9: Cityscape: 4-times upsampled. Top: input at reduced resolution. Mid-left: Building boundary is sharpened with slight distortion. Mid-right: Similar shapes are distorted inconsistently. Bottom: Texture resembles paint strokes. Vehicles are interestingly distorted.

Figure 4.10: Plant: 4-times upsampled. Top: input at reduced resolution. Mid-left: Uneven distortions made the spikes look more ferocious. Mid-right: The tip of the leaf is preserved even when its surrounded by high frequency textures. Bottom: Texture introduced by our approach is similar to the texture of gravel and rocks.

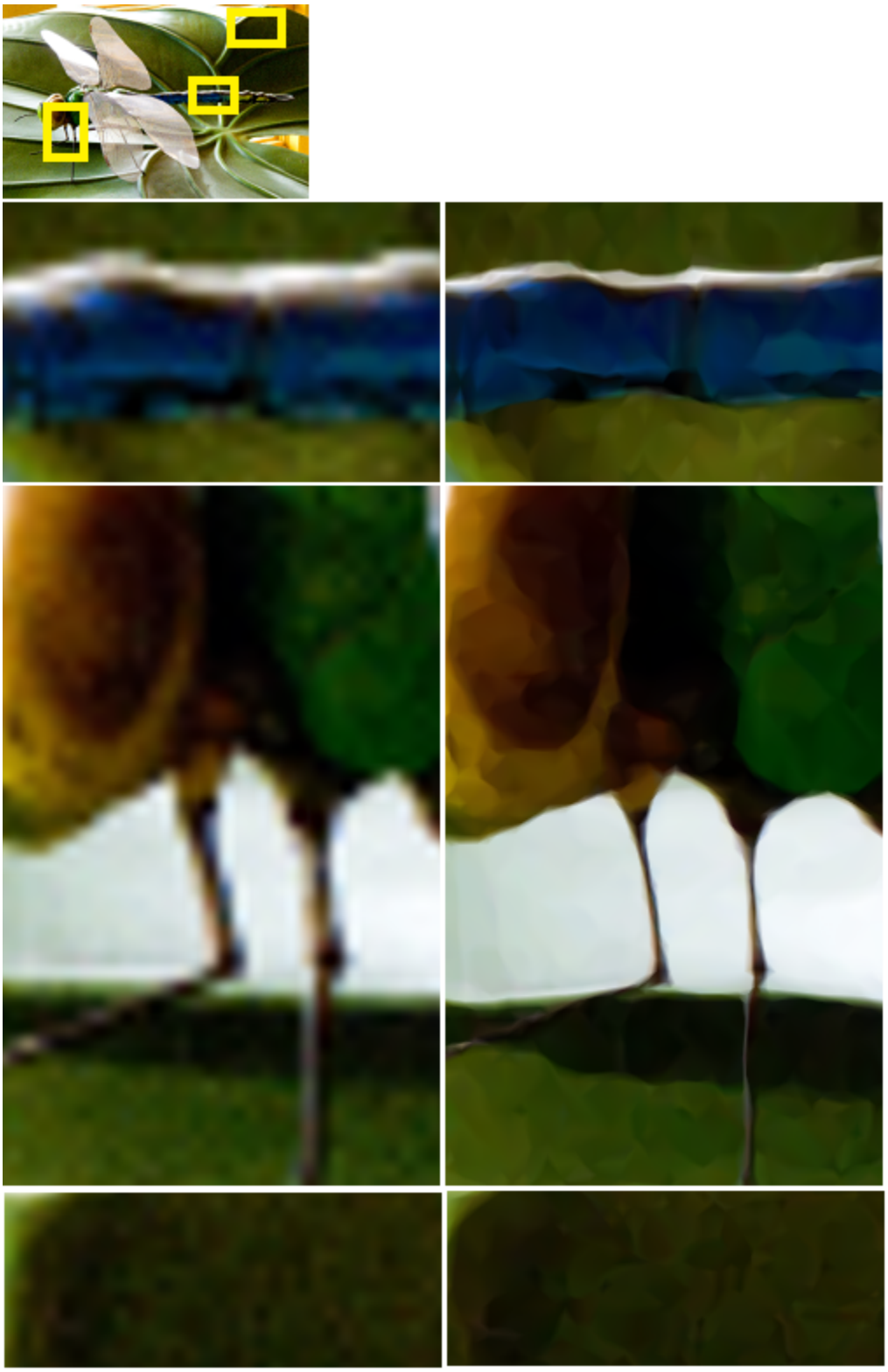

Figure 4.11: Dragonfly: 6-times upsampled. Top: input at reduced resolution. Left column: bicubic upsampled. Right column: proposed approach. Note that the straight tail is slightly crooked; extra edges introduced around eye; the leaf texture is changed.

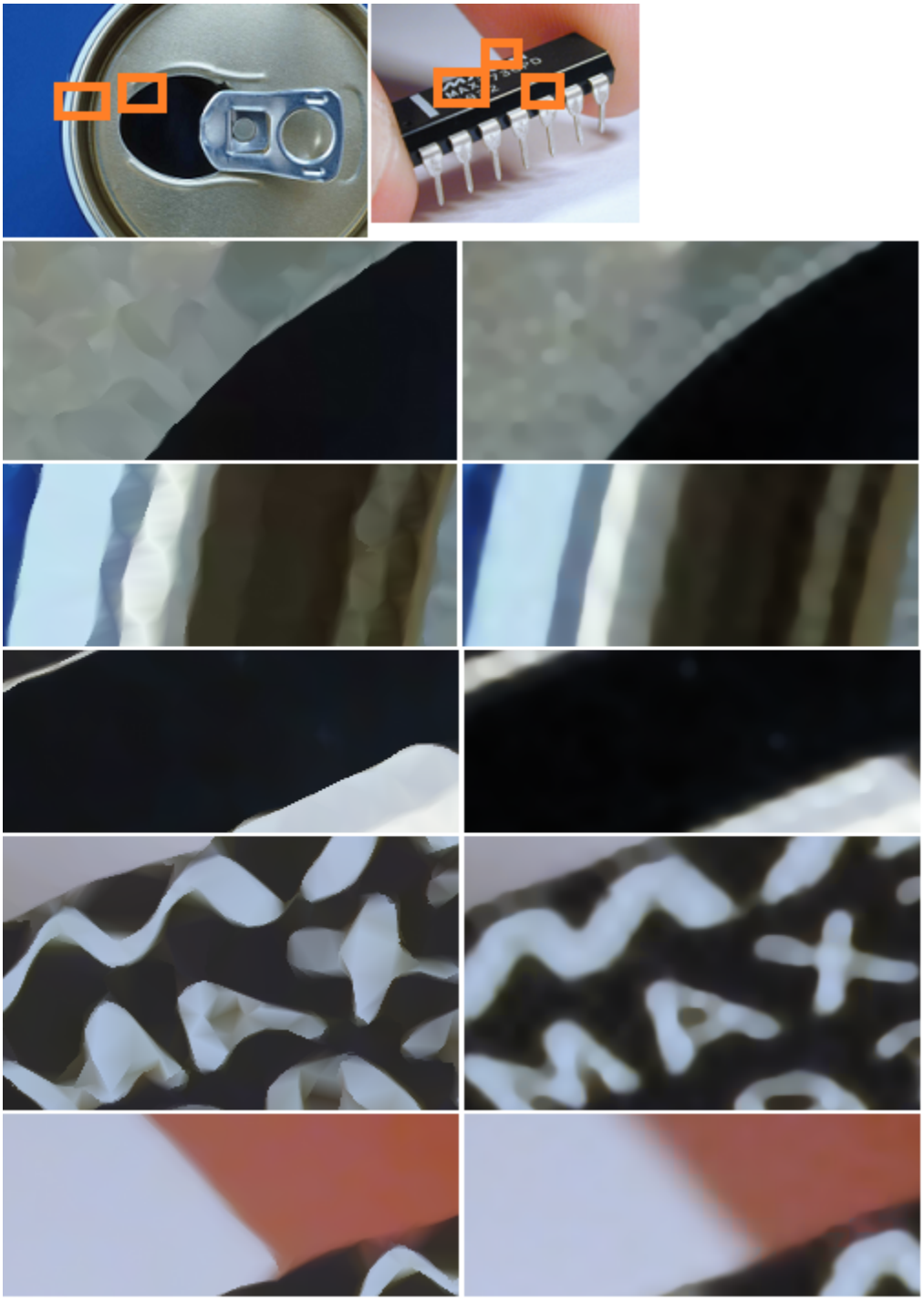

Figure 4.12: Comparison between our and R. Fattal's result [17]. Can: 8-times upsampled. Top row: input at reduced resolution. Left Column: our proposed approach. Right Column: R. Fattal's approach. Overall, we reconstruct sharp edges better. However, migrating pixels may change texture and geometric shapes.

	Proposed Filters	Pixel Migration
Sharpening location	Global	Mostly along edges
Fine-scale details	Lost or arbitrary	Voronoi-like
Recommended Scaling factor	2–4 times	2–10 times
Distortion	None	Along long thin features
Timing (8-times upsampled, quarter-megapixel)	About 300 seconds	5 seconds
Style	More abstract	Realistic

Table 4.2: Comparison between filtering-based approach and pixel migration.

Voronoi-like small-scale details on smooth areas, and it often introduces distortion along long thin features.

Pixel migration is well suited for the task of large-scale upsampling (e.g., with a scaling factor of 8 or higher). Theoretically speaking, no matter the scaling factor, it will reconstruct the same sharp edges because it is the expansion of neighbouring loose springs that is compensating the high scaling factor. Therefore, we recommend using pixel migration for large-scale upsampling. On the other hand, for filtering-based techniques, large-scale upsampling will often produce blocky or smooth artifacts. Moreover, since filtering-based techniques are much slower, if we greatly increase the scaling factor, the time consumed will shift from barely tolerable to totally unacceptable.

As for the styles that exhibit from the two approaches, filtering-based techniques often produce mildly abstract results, due to the sharpening and loss of fine details; whereas pixel migration does not naturally produce any styles. The texture enrichment process adds arbitrary fine details to filtered abstract images. Since the fine details are arbitrary, the resulting style may vary significantly. For pixel migration, occasionally it will produce some Voronoi-like textures that resembles paint strokes. But in general, we consider this process does not produce stylized results. Note that the texture enrichment process could also be applied as a post-processing step to pixel migration results, which may introduce varied styles to pixel migration as well. In future work, we would like to extend our texture enrichment process and try to restrain our stylization to a more specific style, such as oil painting or watercolor.

4.4 Conclusion

In this chapter, we presented pixel migration for edge-preserving image upsampling. We suggest breaking the sample-spacing consistency in the upsampled plane, and instead, migrate pixel samples so that pixels along sharp edges will remain close together on the output image. Thus, the interpolation process will not introduce smoothly interpolated pixels along sharp edges, and the same degree of edge-crossing gradient would be preserved in the upsampled images.

Our pixel migration was achieved using a mass-spring system, with spring coefficients and rest lengths chosen based on the local gradient magnitude distribution. We observed that local maximum gradient usually appears on sharp edges. Thus, by comparing one spring's gradient with its neighbourhood, we can determine whether the spring is an edge-crossing spring, and assign its rest length and spring coefficient accordingly. We also introduced a supplementary angular force that mildly adjusts spring orientations, such that the springs will tend to have the same orientations as the edges.

Based on the combination of forces, pixel positions are re-calculated and updated iteratively. After the migration is terminated, we triangulate the deformed pixel lattice and barycentrically interpolate the unknown pixel values in the upsampled plane.

Our process is fast and effective. Compared to filtering-based approaches which would take minutes to process one image, it only takes seconds to complete our mass-spring simulation. The edge-preserving effect is strong, even if we increase the upsampling scale to 8 times or greater. The edges reconstructed are sharper than those from the state-of-the-art upsampling approach by Fattal [17].

The major limitation of our approach is distortion, which is visible on long thin linear features or textured areas. The distortion is caused by inconsistent or incorrect local gradient ranking. We consider the distortion on textured areas to be somewhat tolerable, while the distortion on linear features need further investigation.

In future work, we are interested in more sophisticated understanding of the input image, so that we can identify linear features and high-frequency textures. Following this train of thought, we are interested in image segmentation, so that different features are enveloped by separate regions. Thus, we can apply different pixel migration schemes to differently-featured segments accordingly.

We also want to investigate other ways of migrating pixels. We find our current

mass-spring implementation not versatile enough. There are too many differently-featured areas that could appear in an image. The problem is so challenging that it is difficult to model and control with only rest lengths and spring coefficients.

Moreover, we are interested in combining pixel migration with stylized image upsampling – the approach from the last chapter. We hope that we can find a pixel migration solution that automatically introduces a stylized look to upsampled images. Currently, our mass-spring implementation results exhibit mild abstract looks, with certain regions showing textures resemble paint strokes. We want to explore the possibility of producing other styles, probably with the help of filtering-based techniques.

Pixel migration methodology is powerful and not limited to image upsampling. It motivated us to open up another line of research: applying pixel migration on the original image plane. Our preliminary work along that direction was published in Expressive, 2015. The next chapter presents how we achieved warping effect using pixel migration. We hope that in the future, we can continue explore other applications of pixel migration.

Chapter 5

Image Warping

5.1 Introduction

Painterly rendering is among the oldest styles in non-photorealistic rendering, dating back to Haeberli's seminal work on user-assisted image stylization [23]. Many others have sought to create synthetic painted images since, whether using dedicated simulations of paint [3], image processing [6, 25], or particle systems over geometric models [31].

The quality of synthetic painted images has risen steadily over the years. Nonetheless, painterly rendering of geometry and painterly effects based on input photographs usually adhere closely to the structure of the input, and thus captures only a limited range of the possible painterly images; historically, paintings have spanned a wide range from meticulously detailed representations to wholly abstract images.

In this chapter, we propose to use image warping to modify an input image before conducting a painterly stylization, thus producing a lively outcome which resembles caricature or other fanciful, exaggerated semi-representational depictions of the subject matter. By manipulating the image before applying the painterly effect, we can convey the impression of a painted image with a painterly filter that is less aggressive, hence more able to preserve the details in the initial photograph.

Figure 5.1 helps to illustrate our objective. Above, we see two paintings from Jean-Baptiste Oudry and El Greco that show deliberate distortions and non-photorealistic perspectives. Below, oil paintings from Caravaggio and John William Waterhouse show a high level of detail and fidelity to the original subject matter. Non-photorealistic painterly stylizations of images have concentrated on abstracted images, often using visible strokes in an expressionist style to emphasize the painterly

effect. Here, we attempt to create the impression of a painted image without visible brushstrokes.

Figure 5.1: Historical paintings illustrating our intention. Above: paintings by Oudry and El Greco demonstrating distortions and problematic perspectives. Below: paintings by Caravaggio and Waterhouse in a photorealistic style.

Our approach involves dividing the image processing into two phases. First, we distort the image by warping the pixel locations in the image plane. Second, we apply a painterly effect to the warped image; we suggest using a relatively non-intrusive painterly filter so that the details of the input image remain visible. The postprocessing has the benefit of reunifying the warped image, concealing defects that might otherwise be visible after the warping phase. The degree of warping is controllable: we can obtain a delicate effect with little warping, a fairly plausible

painterly effect using a medium degree of warping, or a more extreme caricature by warping the image even more heavily. We prefer the results in the middle of this range, but the extremes are available to those who might want to use them to obtain a particular effect.

We make two contributions in this chapter:

• We suggest constructing lively painterly images by a two-stage process of first warping the image plane and then applying an image-space painterly filter to the warped image. There is limited precedent for this approach in general painterly rendering.

• We provide a specific mechanism for accomplishing the painterly rendering process just described. In our approach, the warping is accomplished by performing a mass-spring simulation over the image lattice. The painterly effect can be produced by any of several existing image-space painterly rendering methods; for this thesis, we primarily rely on a variation of the morphological watercolor effect of Bousseau et al.

This chapter is organized as follows. Section 5.2 describes our approach in detail, with Section 5.3 giving results of applying the method to sample images and showing comparisons to other methods. The chapter's final section concludes and provides suggestions about possible future directions.

5.2 Algorithm

Figure 5.2 shows the pipeline of our method. First, we segment the input image into clusters using SLIC. The resulting clusters tend to be compact, uniform in size, and edge-sensitive. Second, we construct the mass-spring system by connecting springs between each pixel and its four-connected neighbours. All springs within a SLIC cluster receive a rest length according to a single randomly chosen value assigned to that cluster; the rest lengths range from approximately 0.1 to 2. Since the default spacing between adjacent pixels is 1, the clusters with x_0 smaller than 1 tend to shrink; and the clusters with x_0 greater than 1 tend to expand.

Our mass-spring simulation then iteratively computes the forces and updates the pixel locations. We halt the simulation after a fixed number of iterations, and create a warped image by triangulating the new pixel locations and interpolating the color. Finally, we apply a painterly filter to the warped image; we used a morphological

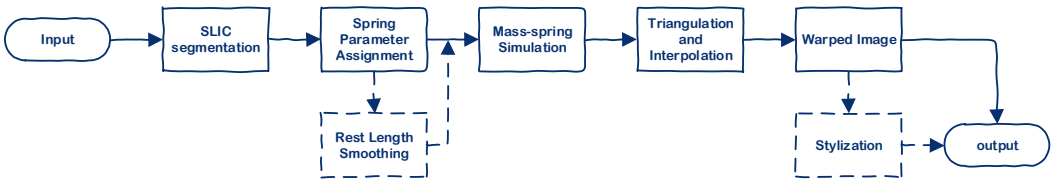

Figure 5.2: Schematic of our processing pipeline. Dashed boxes represent optional steps.

filter to achieve a watercolor appearance [6] , but any desired filter could be used instead. The process is summarized in Figure 5.4.

Given an input image and the approximate superpixel diameter S, we measure the distance D between pixels and superpixel centers using equation (5.1):

$$D = d_{rgb} + (m/S)d_{xy} \ , \tag{5.1}$$

where d_{rgb} denotes the Euclidean distance the RGB color cube, and d_{xy} is the Euclidean distance in the image plane. The parameter m lets us control the compactness of the segmentation. In this thesis, we empirically use $m = 150$; note that RGB values lie in the range 0 to 255. Using this distance computation, we iteratively move all superpixel centers to the centroids of their regions until convergence.

We attach each pixel with springs to its four-connected neighbours. Assuming the spacing between adjacent pixels is 1 unit, we randomize the springs' rest lengths roughly in the range of 0.1 to 2 units. We exaggerate the warping effect by exaggerating the difference between rest lengths, we want a distribution biased towards the minimum and maximum, with less likelihood of values between. All springs within the same superpixel get the same rest lengths, so that the whole region can expand or shrink. With all these considerations, we randomize the rest lengths x_0 using Equation (5.2):

For all the springs s in one SLIC segment,

$$x_0(s) = x_{\min} + x_d + x_d \times (|r|^{\alpha}) \times \text{sign}(r) \tag{5.2}$$

where x_{min} denotes the minimum rest length. Parameter x_d is the difference between the minimum rest length and the average rest length: i.e., the average length is

Figure 5.3: Effect of the full process. Upper left: original image. Upper right: warping, no painterly effect. Lower left: painterly effect, no warping. Lower right: both warping and painterly effect.

$x_{\min} + x_d$. The maximum permitted rest length is $x_{\min} + 2 \times x_d$. Parameter r is a random number drawn from a uniform distribution from -1 to 1, and α is a real number in the range 0 to 1, controlling the distribution of spring lengths. Lower values of α make the spring lengths more likely to be close to minimum or maximum, with the extreme case of a binary distribution at $\alpha = 0$. A choice of $\alpha = 1$ gives a uniform distribution.

We experimented with different settings for α, but the difference is quite subtle, especially compared to other influences such as superpixel size and spring-strength constant. The results shown in this chapter were obtained using $\alpha = 0.5$. We also explored different combinations for the minimum and maximum rest length; we settled on a range of $(0.1, 1.9)$ for the results that we show.

For an arbitrary spring s, its constant k is scaled according to equation (5.3):

$$k(s) = \gamma \times |(x_0(s) - (x_{min} + x_d))| \ , \tag{5.3}$$

where $|(x_0 - (x_{\min} + x_d))|$ denotes the deviation of the rest length of this spring from the average. Very long springs and very short springs have greater deviation. Thus, they are stronger, with higher $k(s)$. The γ value scales the overall strength of all springs. Greater γ values produce a stronger warping effect. A γ of around 3 or 4 gives a reasonable amount of distortion; we show the effect of varying γ in the next section.

Optionally, we can smooth the spring parameters across regions: for applications where quality is paramount and extra processing time is less of a concern, we suggest smoothing the spring parameters by applying a cross-bilateral filter against the original image. This process causes similar regions to blend spring parameters together, yielding a less noticeable transition. Dissimilar neighboring regions do not change much. The effect is minor, but in our judgement produces a slight improvement. We applied the smoothing process to all the results shown in this thesis.

Once spring parameters have been finalized, our mass-spring simulator iteratively calculates the forces exerted by all springs and moves the pixels accordingly. The physics simulation is the same as described in Chapter 4, Section 4.2.3. We also halt the simulation at 50 iterations, a figure found to produce adequate convergence. Having completed the mass-spring simulation, we triangulate the new pixel grid and apply barycentric interpolation to determine pixel color, which gives us the warped image.

Even standing alone, many warped images are already quite interesting. Nonetheless, we suggest post-processing the warped images with painterly filters to accentuate the artistic appearance. Users can apply different filters, if desired; with our objective of maintaining some detail, we relied on a morphological watercolor style inspired by Bousseau et al. [6]. A visual summary of the approach is presented in Figure 5.4.

Figure 5.4: Progression of an image through the pipeline. Top row left to right: original image; SLIC segments; warped SLIC segments; Bottom row left to right: warped image; warping plus painterly effect.

5.3 Results

Figure 5.5 to 5.13 show some results. We demonstrate the effect on images with varied subject matter and backgrounds, including portraits, still lifes, animals, landscapes, and cityscapes. We spent some, but minimal, effort selecting good parameters for specific images; a wide range of parameters give substantially similar overall results for a particular photograph. Figure 5.14 shows the source images.

These result images contain many elements of interest. Textures in the image are largely retained (e.g., books, secretarybird feathers), albeit somewhat abstracted by the painterly filter; in some cases, such as the concrete in the street image, the texture has been augmented by the warping. The distortion of facial features gives new expressions and provides a new interpretation of the image: we find the processed old man image particularly intriguing. The secretarybird now looks monstrous, with distortions in the beak now resembling teeth, and the distorted eye shape adding a further sense of menace. More subtle effects are also possible: the eyes of the boxer are slightly enlarged, giving the dog an even more plaintive expression. The abstracted and distorted face of the man in the hood gives the image a wild look. The warped, fanciful architecture of the cathedral might serve to illustrate a book of fairy tales. The "garlic" still life has a scratchy look that looks somewhat careless and handmade. In general, there is a feeling of a hand-drawn style, with exaggerated shapes (e.g., lighthouse) and wavering lines (e.g., books), combined with a high degree of detail.

Our non-optimized CPU implementation takes about 50 seconds for a half-megapixel image. This time is broken down into approximately 85% to compute the SLIC segmentation, 15% to do the mass-spring simulation, and negligible time to do the painterly filter (less than one second). We are confident that this time could be improved considerably with some effort; the SLIC calculations, in particular, can be sped up a great deal. These timing figures scale linearly with the number of pixels.

Figure 5.5: Generic result - Cathedral.

Figure 5.6: Generic result - Secretary Bird.

Figure 5.7: Generic result - Garlic.

Figure 5.8: Generic result - Hood.

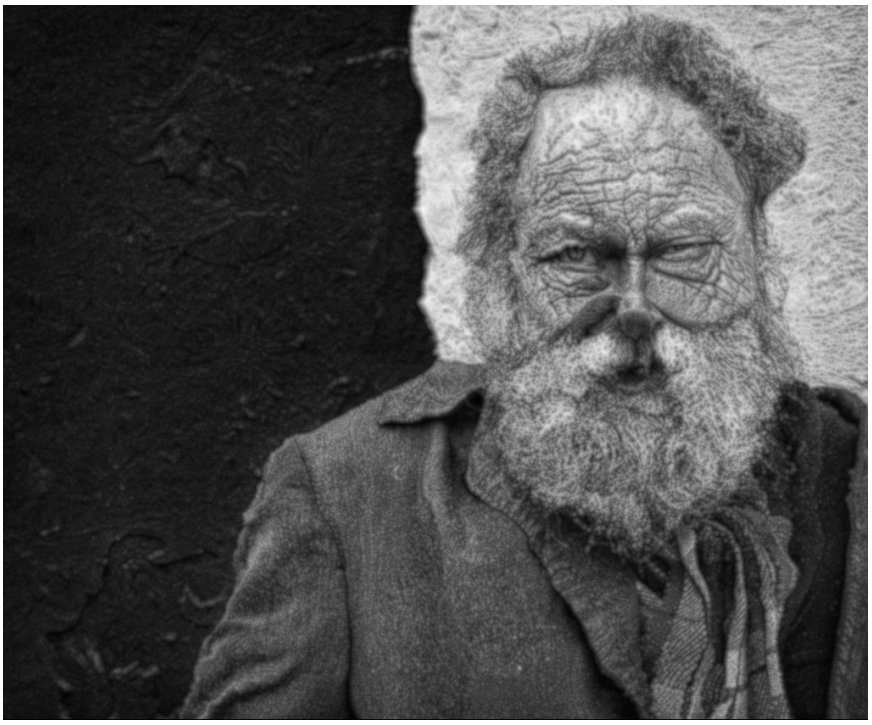

Figure 5.9: Generic result - Old man.

Figure 5.10: Generic result - Boxer.

Figure 5.11: Generic result - Street.

Figure 5.12: Generic result - Light house.

Figure 5.13: Generic result - Books.

Figure 5.14: original images (hood, cathedral, garlic, old man, boxer, lighthouse, street, books, secretarybird).

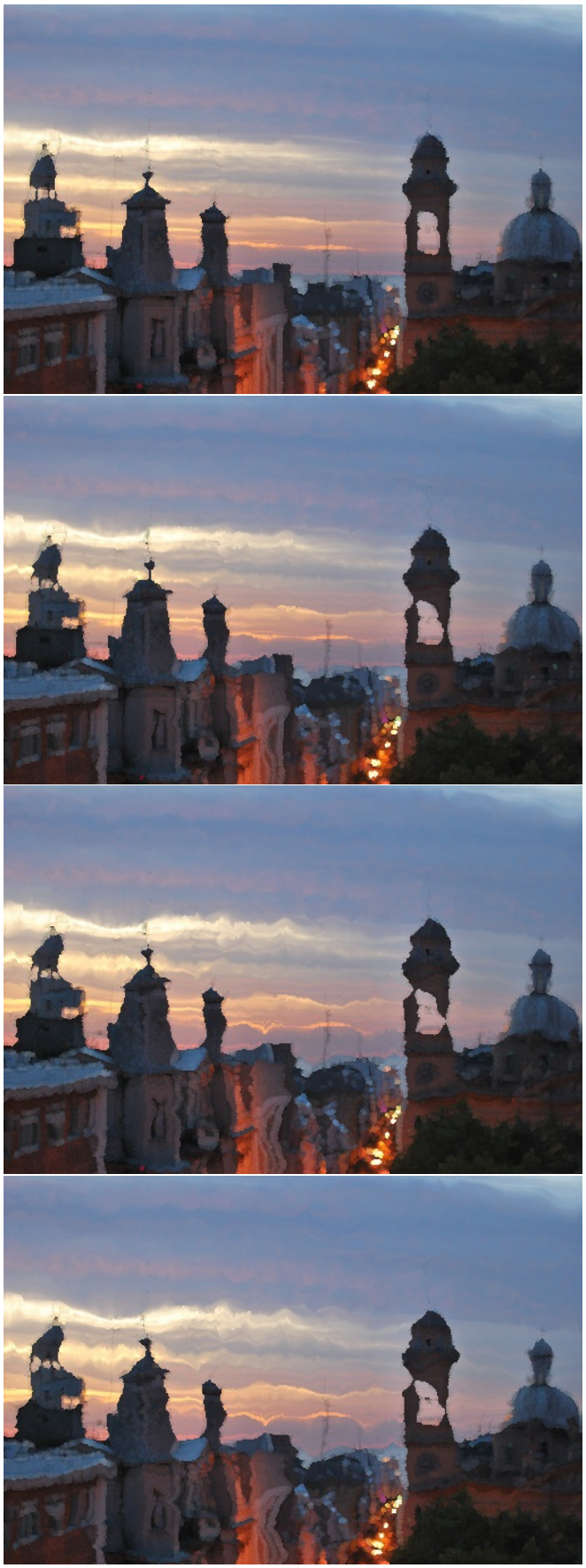

Figure 5.15: Spring strength increases from top to bottom: spring parameters are $\gamma = 1$, $\gamma = 1.5$, $\gamma = 2$, $\gamma = 3$.

Figure 5.16: From top to bottom: SLIC regions of diameter 20, 40, 60, 80 pixels.

Figures 5.15 and 5.16 show the results of changing the algorithm's parameters. In Figure 5.15, the images with higher γ have stronger springs with more polarized rest lengths, producing more exaggerated distortion. The weak springs with $\gamma = 1$ produce an image with scarcely noticeable distortion. As γ increases, the distortion becomes more apparent. We prefer the milder distortions of approximately $\gamma = 2$, but even more extreme distortions are possible. Note that the exact nature of the distortion depends on the specific parameters assigned to specific regions, and because we assign parameters randomly, different outcomes are available by redoing an image.

In the examples in Figure 5.16, the SLIC region size ranges from 20 to 80. The smaller regions produce more high-frequency structure; with larger, fewer regions, the distortion occurs at a larger scale and larger objects can be coherently distorted. The "street" input image has structure on multiple scales, so which SLIC size is appropriate depends on the user's intent: all of these results are plausible. In images with a single important scale, such as portraits, it is more crucial to obtain the right size – see our failure examples below.

Figure 5.17 shows a comparison between our results with Sisley [36] under a low-abstraction configuration. As befitting its efforts to make a fairly abstract image, Sisley removes both shape and color detail; small-scale details are blurred out. With these low-abstraction settings, the colors are fairly close to those of the original photograph. There is a strong paint texture and visible paint strokes. In our result, the details are somewhat preserved, but are still somewhat modified. most notably on the face of the square building just above the sailboat. There is some large-scale distortion: the shoreline is altered, and the right-hand sail of the sailboat is pushed out as if billowing in the wind. The aims of the two methods are quite different, but somewhat complementary, with Sisley aiming at greater degrees of abstraction and ours aiming for an impression of painterliness but retaining significant amounts of detail.

Figure 5.18 shows two failure cases. These images show the effects of the warping only, without painterly postprocessing. In the "crying" image, there is a mismatch between the size of the facial features and the relatively small SLIC regions. This mismatch, plus the use of very strong springs, induces a high-frequency texture over the face; the facial features become jagged as neighboring regions pull them in different directions. The "flamingo" image has quite strong springs as well, but a better match

Figure 5.17: Comparison of our approach with abstract rendering by Sisley. Left: Sisley. Right: ours.

of SLIC region size to feature size. However, the blurred but varied background now has a cobblestone-like texture from the SLIC regions. Also, the flamingos' legs are particularly distorted: in general, when the image contains extended linear features, the distortion is particularly noticeable, and the effect may or may not be acceptable depending on the user's intention. We have used this image to illustrate specific problems, but it has some visual interest anyway, and the birds' feathers are nicely conveyed. Overall, though, our process is not well suited to this type of image.

5.4 Conclusion

In this chapter, we presented a two-stage methodology for creating painterly images from photographs: first, the image is distorted using a mass-spring system, and then a painterly filter is applied to the warped image. We recommend using a painterly filter that does not change the image very much. The warping process only changes the size and relative position of details, but does not by itself remove detail; by using a painterly filter only lightly, we can unify the image into a painterly style but still produce a detailed image. We hope that this chapter spurs others to consider more representational automatic painterly rendering, as opposed to the more abstract and expressionist examples of the medium that have been popular in NPR.

Our warping was done using a mass-spring system, with spring coefficients and rest lengths chosen randomly for each region of a SLIC oversegmentation of the input photograph. The mass-spring distortion is fast and flexible, properties noted by

Figure 5.18: Two failure cases. Above: crying. Below: flamingo.

Piponi and Borshukov [38]. Using SLIC gives us a general and robust segmentation, where the segments are all approximately the same size. Because the segment boundaries tend to lie on image edges, we preserve some structure of the input image through this process.

After warping, the image can be processed by any conventional image-space painterly rendering system. We used a morphological filtering method to produce a watercolor effect, following Bousseau et al. [6].

Our process is effective at generating painted-looking images. The two elements of the process cooperate to produce the illusion: the warping ensures that the image is not excessively faithful to the underlying photograph, while the painterly post-processing makes the image look painted and allows the naive viewer to attribute the distortion to the painting process itself. Sisley [36] is the process most similar to ours, doing both distortion and painterly rendering in a unified stroke-based rendering environment; its aim was to create highly abstracted images, whereas ours is to produce representational images that nonetheless are painterly and not excessively photographic. Collomosse and Hall [8] used separate warping and painterly filtering stages in their pipeline, with the aim of creating cubist paintings rather than more conventionally representational images.

Our method has some limitations. Its effectiveness depends somewhat on matching the scale of the segmentation to the scale of the objects within the image: the spatially-varying distortion of textures and very small objects may look incoherent. Because of the random assignment of spring parameters, executing the process multiple times over the same image produces different results, not all of which are equally appealing.

In future work, we are interested in further exploring a detailed painterly style that lacks visible brushstrokes. Replacing our content-agnostic distortions with shape simplifications and using more spatially coherent parameters might be helpful. Although in this work we concentrated on fully automatic processing, the system could benefit from a lightweight user-assisted labeling interface.

We want to continue the pursuit of realism in painting, as practiced by Romantic painters, for example. One aspect of this is to include paint texture without necessarily indicating individual brush strokes. Alternatively, producing less-realistic paintings could be accomplished by fusing the results of multiple different distortions of a single input image. Lastly, adapting the technique to video could be interesting:

in this case, we would want to attach the spring parameters to persistent objects in the video, possibly accomplished by performing a segmentation over the video cube.

Chapter 6

Conclusion

In this thesis, we intend to achieve stylized upsampled images which contain sharp edges as well as exhibit some artistic value. To achieve this goal, two ideas are explored: filtering-based techniques and pixel migration. Through various implementations of the two ideas, we sharpened edges, reconstructed fine details and achieved various styles.

Stylized upsampling refers to a change of upsampling goals. Unlike other existing methods that aim to produce photographic looks, we want the upsampled images to exhibit a stylized look. We achieved this goal through preserving the sharp edges of the input, while reducing, or replacing fine details. We believe that the freedom of changing fine details makes image upsampling easier, and stylized high-resolution images may express great aesthetic value. We implemented this stylized upsampling idea using filtering-based techniques.

We presented variations of two existing filters: cumulative ranged geodesic filter and bilateral filter. Cumulative ranged geodesic filter grows continuous irregular-shaped masks, whereas bilateral roundup, an innovative variation of bilateral filter, grows discrete irregular-shaped masks. Both filters' masks respect sharp edges, preserve medium scale details, and remove fine details. Their filtering results preserve sharp edges of the input and exhibit a mild abstract, painterly looking. We also introduced a simple texture-enrichment approach that further enhances the stylization by introducing fine details.

We proposed a second idea, pixel migration, for solving the image upsampling problem. Pixel migration means we relocate pixels on the the image plane. In the context of image upsampling, the relocations happen mostly along sharp edges, in the sense that pixels from either side of an edge would move close to each other.

The relocation closes the gaps along upsampled edges; thus, removes the smoothly interpolated pixels and preserves edge sharpness.

We implemented pixel migration using a mass-spring system, where each pixel is connected to its four adjacent neighbours by springs. We vary the rest lengths and spring coefficients in the system, such that the pixels have to move to compensate the force differences. The local highest gradient magnitudes usually exist along sharp edges. Thus, we identify the edge-crossing springs by comparing their gradient magnitudes with those of their neighbours. We assign the shortest rest length and strongest spring coefficient to these edge-crossing springs, so that the two end pixels of the spring would be brought together. Thus, the gap between two sides of an edge is removed. The mass-spring implementation is fast and effective. It generates results that has extremely sharp edges, with some distortion artifacts along thin linear features.

Fascinated by the distortion effect along object silhouettes, we extended our pixel migration methodology to the original image plane. We were able to achieve an effect that resembles caricature or other exaggerated semi-representational depictions of the subject matter. We presented a two-stage process for creating such effect: first the image is distorted using a mass-spring system, and then a painterly filter is applied to the warped image.

Similar to the mass-spring system we used for upsampling, we enforce pixel movement by varying the spring rest lengths and spring coefficients. First, we oversegment the image using SLIC [1]. This segmentation respects edges, in the sense that edges only exist on segment boundaries where multiple segments meet. We choose spring parameters randomly for each SLIC region. The force inconsistency between SLIC regions will cause them to shrink or expand, which warps the underlying edges reside along the deformed region boundaries. After warping, the image can be processed by any conventional image-space painterly rendering system. We used a morphological filtering method to produce a watercolor effect, following Bousseau et al. [6].

This process effectively generates painted-looking images. The warping ensures that the image is not excessively faithful to the underlying photograph, while the painterly post-processing makes the image look painted and allows the naive viewer to attribute the distortion to the painting process itself. This result is published in Expressive, 2015 [29].

In future work, we are interested in further combining the operations we mentioned

above. Individually, they are specialized in edge sharpening, fine detail preservation, or generating painterly effects. Combined, they give us a comprehensive and sophisticated tool set. We hope we can apply this tool set to design image upsampling algorithms that inherit our successes and overcome the limitations.

We would like to further explore other painterly styles. Our filtering-based upsampling results exhibit abstract looks with elements resembles brush strokes; whereas our image warping results pursued a balance between abstract and realism. We believe there must exist many other styles that can be put in the context of image upsampling, and their respective stylized upsampled results may express higher aesthetic value.

We are interested in larger-scale upsampling. Most of our current results are generated from 4 to 8-times upsampling. We find it difficult to increase upsampling scale due to the increased memory requirement. Also, we want to improve the performance of our filtering-based approaches by refining and accelerating the implementation. Finally, we would like to put all of the proposed processes together and create an image upsampling and stylization tool. It will help us better demonstrate our findings, make our work more accessible to the public, and hopefully, motivate and inspire future research in this field.

List of References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2012.120

[2] S. Bae, S. Paris, and F. Durand, "Two-scale tone management for photographic look," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 637–645, Jul. 2006. [Online]. Available: http://doi.acm.org/10.1145/1141911.1141935

[3] W. Baxter, J. Wendt, and M. C. Lin, "Impasto: A realistic, interactive model for paint," in *Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering*, ser. NPAR '04. New York, NY, USA: ACM, 2004, pp. 45–148. [Online]. Available: http://doi.acm.org/10.1145/987657.987665

[4] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 6, pp. 567–585, Jun. 1989. [Online]. Available: http://dx.doi.org/10.1109/34.24792

[5] A. Bousseau, M. Kaplan, J. Thollot, and F. X. Sillion, "Interactive watercolor rendering with temporal coherence and abstraction," in *Proceedings of the 4th International Symposium on Non-photorealistic Animation and Rendering*, ser. NPAR '06. New York, NY, USA: ACM, 2006, pp. 141–149. [Online]. Available: http://doi.acm.org/10.1145/1124728.1124751

[6] A. Bousseau, F. Neyret, J. Thollot, and D. Salesin, "Video watercolorization using bidirectional texture advection," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007. [Online]. Available: http://doi.acm.org/10.1145/1276377.1276507

[7] J. Brosz, F. F. Samavati, M. S. T. Carpendale, and M. C. Sousa, "Single camera flexible projection," in *Proceedings of the 5th International Symposium on Non-photorealistic Animation and Rendering*, ser. NPAR '07. New York, NY, USA: ACM, 2007, pp. 33–42. [Online]. Available: http://doi.acm.org/10.1145/1274871.1274876

[8] J. P. C. and P. M. H., "Cubist style rendering from photographs," in *IEEE transactions on visualization and computer graphics*, 2003, pp. 443–453.

[9] T. C. and M. R., "Bilateral filtering for gray and color images," in *Computer Vision*. IEEE, 1998, pp. 839–846.

[10] N. S.-H. Chu and C.-L. Tai, "Moxi: Real-time ink dispersion in absorbent paper," in *ACM SIGGRAPH 2005*, ser. SIGGRAPH '05. New York, NY, USA: ACM, 2005. [Online]. Available: http://doi.acm.org/10.1145/1187112.1187186

[11] P. Coleman and K. Singh, "Ryan: Rendering your animation nonlinearly projected," in *Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering*, ser. NPAR '04. New York, NY, USA: ACM, 2004, pp. 129–156. [Online]. Available: http://doi.acm.org/10.1145/987657.987678

[12] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002. [Online]. Available: http://dx.doi.org/10.1109/34.1000236

[13] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin, "Computer-generated watercolor," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '97. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 421–430. [Online]. Available: http://dx.doi.org/10.1145/258734.258896

[14] D. DeCarlo and A. Santella, "Stylization and abstraction of photographs," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 769–776, Jul. 2002. [Online]. Available: http://doi.acm.org/10.1145/566654.566650

[15] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[16] F. E. A. El-Samie, M. M. Hadhoud, and S. E. El-Khamy, *Image Super-Resolution and Applications*. 711 Third Avenue, New York: CRC Press, 2012.

[17] R. Fattal, "Image upsampling via imposed edge statistics." *ACM Trans. Graphics*, vol. 26, no. 3, pp. 95–102, Jul 2007.

[18] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *IEEE Computer Graphics and Applications*, vol. 22, no. 2, pp. 56–65, 2002.

[19] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image." Computer Vision, Oct. 2009, pp. 349–356.

[20] B. Gooch, E. Reinhard, and A. Gooch, "Human facial illustrations: Creation and psychophysical evaluation," *ACM Trans. Graph.*, vol. 23, no. 1, pp. 27–44, Jan. 2004. [Online]. Available: http://doi.acm.org/10.1145/966131.966133

[21] Y. HaCohen, "Image upsampling via texture hallucination," *Computational Photography (ICCP)*, pp. 1–8, Mar 2010.

[22] M. M. Hadhoud, M. I. Dessouky, F. E. A. El-Samie, and S. E. El-Khamy, "Adaptive image interpolation based on local activity levels," *Radio Science Conference*, vol. 1, no. 4, 2003.

[23] P. Haeberli, "Paint by numbers: Abstract image representations," *Computer Graphics*, vol. 24, no. 4, pp. 207–214, 1990.

[24] J.-K. Han and H.-M. Kim, "Modified cubic convolution scaler with minimum loss of information," *Optical Engineering*, vol. 40, no. 4, pp. 540–546, 2001. [Online]. Available: http://dx.doi.org/10.1117/1.1355250

[25] A. Hertzmann, "Painterly rendering with curved brush strokes of multiple sizes," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '98. New York, NY, USA: ACM, 1998, pp. 453–460. [Online]. Available: http://doi.acm.org/10.1145/280814.280951

[26] M. Kass and J. Solomon, "Smoothed local histogram filters," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 100:1–100:10, Jul. 2010. [Online]. Available: http://doi.acm.org/10.1145/1778765.1778837

[27] K. I. Kim and Y. Kwon, "Example-based learning for single-image super-resolution and jpeg artifact removal," *Technical Report*, vol. 173, no. 5, pp. 745–770, August 2008.

[28] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, "Joint bilateral upsampling," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007. [Online]. Available: http://doi.acm.org/10.1145/1276377.1276497

[29] J. Li and D. Mould, "Image warping for a painterly effect," *Computational Aesthetics*, May, 2015.

[30] P. Litwinowicz, "Processing images and video for an impressionist effect," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '97. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 407–414. [Online]. Available: http://dx.doi.org/10.1145/258734.258893

[31] B. J. Meier, "Painterly rendering for animation," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 477–484. [Online]. Available: http://doi.acm.org/10.1145/237170.237288

[32] J. Miller and D. Mould, "Accurate and discernible photocollages," in *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, ser. CAe '12. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2012, pp. 115–124. [Online]. Available: http://dl.acm.org/citation.cfm?id=2328888.2328908

[33] D. Mould, "A stained glass image filter," in *Proceedings of the 14th Eurographics Workshop on Rendering*, ser. EGRW '03. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 20–25. [Online]. Available: http://dl.acm.org/citation.cfm?id=882404.882407

[34] ——, "Texture-preserving abstraction," in *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, ser. NPAR '12. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2012, pp. 75–82. [Online]. Available: http://dl.acm.org/citation.cfm?id=2330147.2330162

[35] ——, "Image and video abstraction using cumulative range geodesic filtering," *Computers and Graphics*, vol. 37, no. 5, pp. 413–430, Aug 2013.

[36] D. Piponi and G. Borshukov, "Seamless texture mapping of subdivision surfaces by model pelting and texture blending," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 471–478. [Online]. Available: http://dx.doi.org/10.1145/344779.344990

[37] ——, "Seamless texture mapping of subdivision surfaces by model pelting and texture blending," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 471–478. [Online]. Available: http://dx.doi.org/10.1145/344779.344990

[38] ——, "Seamless texture mapping of subdivision surfaces by model pelting and texture blending," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 471–478. [Online]. Available: http://dx.doi.org/10.1145/344779.344990

[39] G. Ramponi, "Warped distance for space-variant linear image interpolation," *Image Proc.*, vol. 8, no. 5, pp. 629–639, nov 1999.

[40] P. Rosin and J. Collomosse, *Image and Video-Based Artistic Stylization*. London, New York: Springer, 2013.

[41] Q. Shan, Z. Li, J. Jia, and C. K. Tang, "Fast image video upsampling," *ACM Trans. Graphics.*, vol. 27, no. 5, pp. 1–7, 2008.

[42] Q. Shan, J. Jia, and A. Agarwala, "High-quality motion deblurring from a single image," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 73:1–73:10, Aug. 2008. [Online]. Available: http://doi.acm.org/10.1145/1360612.1360672

[43] J. H. Shin, J. H. Jung, and J. K. Paik, "Regularized iterative image interpolation and its application to spatially scalable coding," *Consumer Electronics*, vol. 44, no. 3, pp. 1042–1047, Aug 1998.

[44] K. Singh, "A Fresh Perspective," in *Proceedings of the Graphics Interface 2002 Conference*, May 2002, pp. 17–24.

[45] Y.-Z. Song, P. L. Rosin, P. M. Hall, and J. Collomosse, "Arty shapes," in *Proceedings of the Fourth Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging*, ser. Computational Aesthetics'08. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2008, pp. 65–72. [Online]. Available: http://dx.doi.org/10.2312/COMPAESTH/COMPAESTH08/065-072

[46] M. Unser, "Splines: a perfect fit for signal and image processing," *Signal Processing Magazine, IEEE.*, vol. 16, no. 6, pp. 22–38, 1999.

[47] H. Winnemöller, "XDoG: Advanced image stylization with extended difference-of-gaussians," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*, ser. NPAR '11. New York, NY, USA: ACM, 2011, pp. 147–156. [Online]. Available: http://doi.acm.org/10.1145/2024676.2024700

[48] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via L0 gradient minimization," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 174:1–174:12, Dec. 2011. [Online]. Available: http://doi.acm.org/10.1145/2070781.2024208

[49] M. Zhao and S.-C. Zhu, "Portrait painting using active templates," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*, ser. NPAR '11. New York, NY, USA: ACM, 2011, pp. 117–124. [Online]. Available: http://doi.acm.org/10.1145/2024676.2024696