Towards Better Methods of Stereoscopic 3D Media Adjustment and Stylization

by

Lesley Istead

A thesis
presented to the University of
Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2018

© Lesley Istead 2018

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner	NAME	Robert Allison
	Title	Associate Professor
Supervisor(s)	NAME	Craig S. Kaplan
	Title	Associate Professor
Internal Member	NAME	Daniel Vogel
	Title	Associate Professor
Internal-external Member	NAME	Ben Thompson
	Title	Associate Professor
Other Member(s)	NAME	Jeff Orchard
	Title	Associate Professor

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis was built from previously published material which I authored, but contains feedback from my supervisor, Dr. Craig S. Kaplan, and co-author Dr. Paul Asente.

Work on disparity map correction was conducted with Dr. Paul Asente, Dr. Scott D. Cohen, and Dr. Brian L. Price at Adobe Research Labs and is discussed in detail in our patent " Methods and apparatus for correcting disparity maps using statistical analysis on local neighborhoods" [2]. Chapter 3 was taken from "Layer-based disparity adjustment in stereoscopic 3D media" [38]. Chapter 5 was taken from "Stereoscopic 3D image stylization" and "Consistent stylization and painterly rendering of stereoscopic 3D images" [61, 62].

Abstract

Stereoscopic 3D (S3D) media is pervasive in film, photography and art. However, working with S3D media poses a number of interesting challenges arising from capture and editing. In this thesis we address several of these challenges. In particular, we address disparity adjustment and present a layer-based method that can reduce disparity without distorting the scene. Our method was successfully used to repair several images for the 2014 documentary "Soldiers' Stories" directed by Jonathan Kitzen. We then explore consistent and comfortable methods for stylizing stereo images. Our approach uses a modified version of the layer-based technique used for disparity adjustment and can be used with a variety of stylization filters, including those in Adobe Photoshop. We also present a disparity-aware painterly rendering algorithm. A user study concluded that our layer-based stylization method produced S3D images that were more comfortable than previous methods. Finally, we address S3D line drawing from S3D photographs. Line drawing is a common art style that our layer-based method is not able to reproduce. To improve the depth perception of our line drawings we optionally add stylized shading. An expert survey concluded that our results were comfortable and reproduced a sense of depth.

Acknowledgements

I'd like to thank the following people and organizations for their advice and contributions to my research.

- Sean Arden, Research Technician, Emily Carr University. Thank you for providing feedback on the quality of our disparity adjustment method results.
- Paul Asente, Senior Principal Scientist, Creative Intelligence Lab, Adobe Research. Thank you for the wonderful research internship opportunities, for helping to bring our S3D work to life, and providing feedback.
- Don Carmody, Producer. Thank you for letting me be your guest on the set of Pompeii. It was a wonderful and valuable experience.
- Scott Cohen, Principle Scientist, Creative Intelligence Lab, Adobe Research. Thank you for helping with our disparity map cleanup algorithm, and for providing feedback on our S3D stylization work.
- Christine Demore, Cristin Fidler, Pam Fidler, Victoria Graham, Karla Istead, Laurie Istead, Sierra Scanlon and Don Smith. Thank you for providing thousands of your personal photographs to use in our inpainting experiments.
- Vic DiCiccio, Research Professor, University of Waterloo. Thank you for everything—introducing me to so many individuals and companies that have taught me about S3D and film, provided feedback and helped drive my research. Thank you for giving honest advice.
- Everyone at Gener8. Thank you for the wonderful research collaboration and giving me an opportunity to work with S3D conversion of several S3D films.
- John Helliker, Dean of Innovation and Engagement, Sheridan College. Thank you for the wonderful research collaborations over the years at SIRT.
- Nick Iannelli, Senior Vice President, Postproduction and Ad Services at Deluxe Toronto. Thank you for arranging a mini-internship with Deluxe so that I could learn the ropes of post production, and for arranging on-set visits to Pompeii.
- Joe Istead. Thank you for everything, providing advice, feedback, time.
- Craig S. Kaplan, Associate Professor, University of Waterloo. Thank you for being my supervisor and supporting me through this wild, completely un-conventional PhD.
- Ali Kazimi, Professor and Cinematograper, York University. Thanks for teaching me how to use my Fuji W3 S3D camera properly.

- Jonathan Kitzen, Director, Writer and Producer. Thank you for the wonderful opportunity to work with you in repairing WW1 S3D photos for your film Soldiers' Stories. I can't think of a better way to demonstrate that our methods work, than to use it in a real production!
- Glen MacPherson, Cinematograper, CSC/ASC. Thank you for taking time on the set of Pompeii to discuss your S3D cinematography, capture methods and technology.
- Brian Price, Senior Research Scientist, Creative Intellgence Lab, Adobe Research. Thank you for providing feedback on our disparity map cleanup algorithm and S3D stylization work.
- Denise Quesnel, PhD Student, iSpace Lab, School of Interactive Arts + Technology at Simon Fraser University. Thank you for providing critical feedback on the quality of our disparity adjustment and stylization method results.
- Dr. Christian Richardt, Assistant Professor, University of Bath. Thank you for your insight into our algorithms, for rectifying our 100yr old WWI S3D photos, and providing valuable feedback on our results.
- Study Participants. Thank you for participating in our various studies, giving feedback and helping to produce inputs.
- Oliver Wang, Senior Research Scientist, Creative Intelligence Lab, Adobe Research. Thank you for providing comparative results for our disparity adjustment method and providing feedback on our S3D line drawing results.

Dedication

For Morwenna, Baby Neville, and Joe.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Stereoscopic 3D (S3D) media is widespread. From films and photos to games, technical diagrams, art and cereal box prizes, S3D is used to add depth and create more immersive media. But how does it work? What complications arise from this form of media?

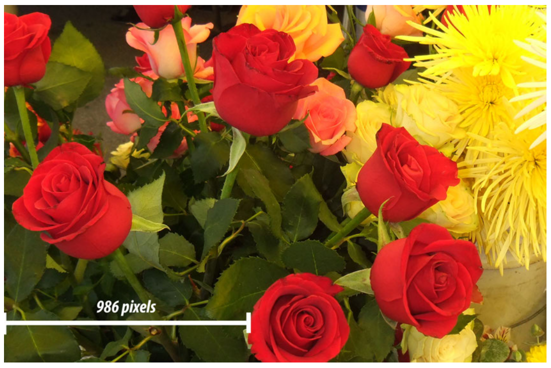
In order to fully infer depth from S3D media, binocular vision is required. 95% of the human population has fully functional binocular vision and can perceive depth [21]. Binocular vision requires two eyes working together. Each eye observes a slightly different image due to the separation between the eyes. This separation is called the *interocular distance*. The difference between the observed images presents itself as a horizontal shift in object positions. The distance of the horizontal shift is called *horizontal parallax* or just *parallax* [88]. Figure 1.1 demonstrates parallax. Note how the red roses are horizontally shifted between the left and right images.

Stereopsis is the ability to infer depth from parallax. It is the primary cue required for depth perception. The *interaxial* distance is the distance between two cameras used to capture an S3D image. The interaxial distance impacts parallax and our perception of depth. Increasing the interaxial distance increases parallax and the apparent depth or roundness of objects. Decreasing the interaxial distance decreases parallax causing objects to appear more flat [58]. However, parallax is not the only cue that helps us perceive depth and object shape. Monocular depth cues such as occlusion, motion, relative size and lighting add to our understanding [15, 58].

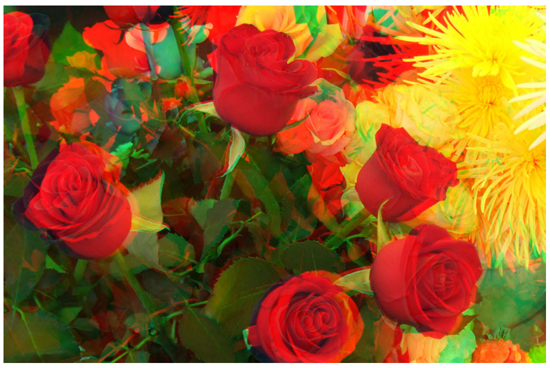
It is difficult to create effective S3D photographs and films. Instead of a single image or film, two are needed, one for each eye to produce stereopsis. Typically an S3D camera is used to capture S3D media. S3D cameras such as the Fuji FinePix REAL 3D W3 have two horizontally separated lenses corresponding to the left and right eyes. However, S3D images can be captured in a number of ways. For example, a pair of cameras can be used in a side-by-side arrangement to produce the left and right views. A single camera may also be used by taking consecutive, horizontally separated images to produce left and right views. Or, a single image can be converted to S3D by producing an artificial second view.

Regardless of how S3D media is captured many new parameters must be considered to produce S3D media that is comfortable to view and produces believable depth. Once captured, the S3D image often contains flaws that cause discomfort and hence may require clean-up or correction. For example, if discrepancies exist in colour or focus between left and right images viewers may experience discomfort. A more serious issue occurs when parallax is too large, causing viewer discomfort due to eye strain [50, 58]. Conversely, when parallax is too small, objects will appear flatter than desired. It is tempting to use the interaxial distance to separate lenses/cameras to

(a) left view



(b) right view



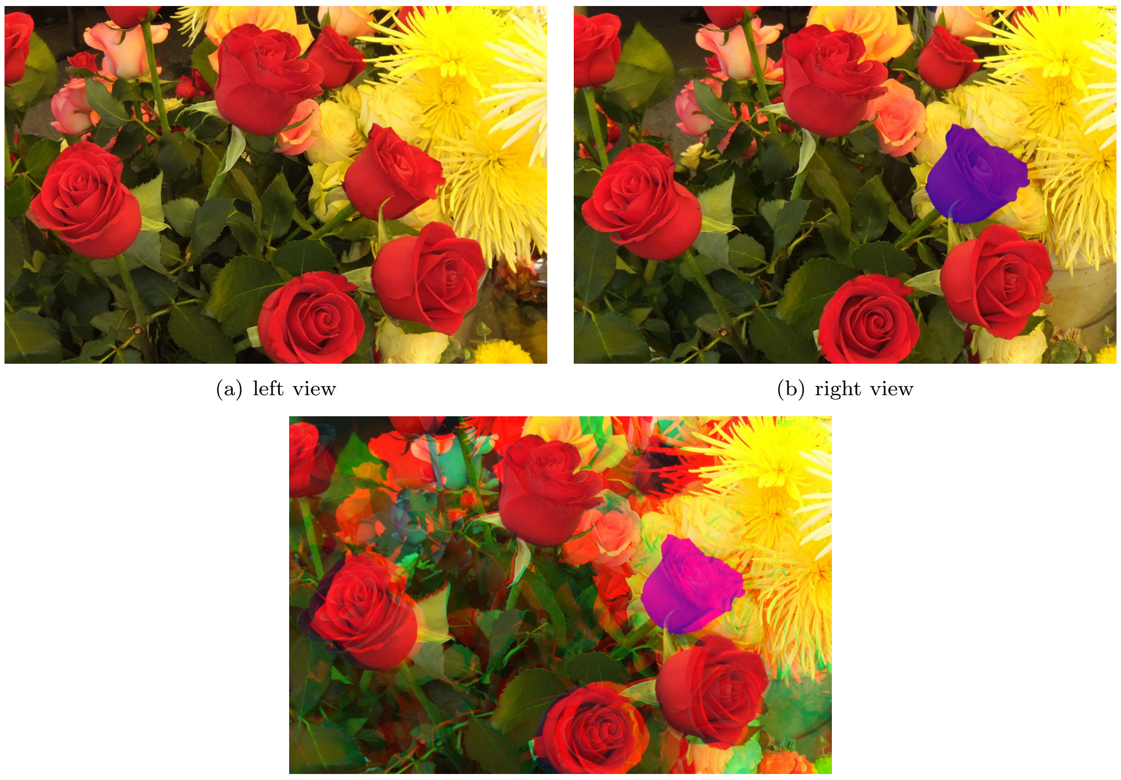
(c) S3D image, red/cyan anaglyph

Figure 1.1: Left and right views as seen by the left and right eyes. Note that all anaglyph images in this thesis require red/cyan glasses for optimal viewing.

eliminate this issue. While this would simulate the human visual system it restricts artistic freedom with regards to depth. It may also still produce uncomfortable or unexpected results when viewed on larger screens or combined with longer lenses. Instead, clean-up and corrections are handled by post-production artists [72]. The process is costly and requires significant manual labour as there are few automated tools to perform these tasks. Some corrections, such as parallax adjustment, are particularly challenging and typically involve re-capturing the image.

Captured and corrected S3D images may be further manipulated for artistic purposes. For example, artists may add depth-of-field to emphasize the subject. Or, artists may wish to give their piece a painted look by applying a painterly rendering algorithm such as Hertzmann's [31]. However, applying stylization filters from Instagram, Snapchat or Adobe Photoshop to S3D images can produce viewer discomfort because they are not depth-aware and often introduce inconsistencies between the left and right views.

While there exist algorithms specifically designed to apply stylization to S3D media, the greatest challenge is to maintain *consistency*. S3D images are consistent when objects appearing in both views match visually. If the colour, texture, rotation or vertical position of an object differs between views, then the pair is *inconsistent* and the image is less comfortable to view since the brain cannot match the objects to perform stereopsis [9, 47]. Figure 1.2 demonstrates inconsistent views—note how a rose that is red in one view, is purple in the other. Note the discomfort that arises when the left and right images differ in colour—you may even experience binocular rivalry, where the brain rapidly switches focus between the red and purple roses. While inconsistencies of this magnitude do occur, subtle inconsistencies are more common. Many existing S3D algorithms fail to guarantee


(a) left view (b) right view

(c) stereo anaglyph

Figure 1.2: Inconsistent left and right views.

consistency.

In this thesis we address several issues in manipulating S3D media. First, we address parallax adjustment with a layer-based method that guarantees consistency regardless of the S3D input. Unlike previous methods, ours does not distort objects. The method was used to adjust the parallax of a dozen images in the 2014 S3D documentary "Soldiers' Stories" directed by Jonathan Kitzen. These images could not successfully be adjusted using commerical tools, would be impossible to re-capture, and would be too costly to convert.

Our layer-based parallax adjustment method introduces holes in the image where regions are exposed that were not previously visible. We present a simple, depth-aware approach to filling these holes. We note, however, that it would be more efficient to fill small holes with interpolation algorithms instead of costlier inpainting algorithms such as PatchMatch, used in Adobe Photoshop's "Content-Aware Fill" feature [6]. We then conducted an experiment to determine if it was possible to use machine learning to select a convincing inpainting algorithm for a hole based on the properties of the hole and the surrounding image. Our experiment indicates that it is possible, and our prototype was able to select an algorithm that would produce a convincing result 82% of the time.

Next, we address stylization of S3D media. We modify our layer-based approach so that instead of adjusting parallax, we use it apply stylization filters to S3D images with guaranteed consistency. We also present an S3D painterly rendering algorithm, which utilizes our layered approach to produce painterly S3D images with long, curved strokes that follow object contours. A user study indicates that our stylized images were more comfortable to view than those produced by previous methods.

Finally, while our layered approach to stylization guarantees consistent images, some stylizations produce unpleasant, shattered results, while others like line drawings simply cannot be produced. While algorithms to produce S3D line drawings exist for 3D models, we present a method to produce S3D line drawings directly from S3D images. We add shading to improve depth perception and evaluate our results via a panel of experts.

The intended application of this work is S3D film, however, we chose to use S3D images to develop and assess our methods. This enabled us to focus on producing consistent, convincing results without worrying about the added complexities of film, such as temporal coherence. We note that our methods can be applied to film on a frame-by-frame basis, however, we leave optimization of the presented algorithms for film as future work.

This thesis is organized as follows. Chapter 2 discusses relevant background information regarding stereopsis, stereo production and relevant issues. Chapter 3 presents our parallax adjustment algorithm. Chapter 4 discusses our automatic inpainting algorithm selection experiment. Chapter 5 presents our S3D stylization and painterly rendering algorithms. Chapter 6 presents our S3D line drawing method. Chapter 7 and 8 present concluding remarks and future work.

Chapter 2

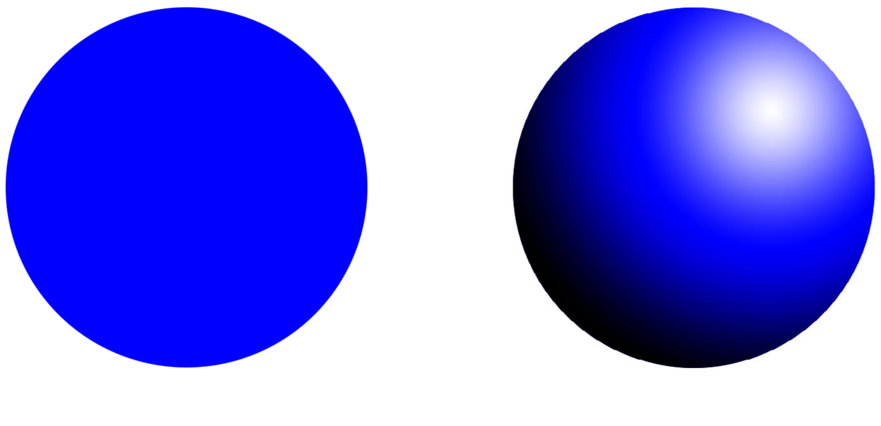
Background

2.1 Stereo Vision and Comfort

This section offers an informal overview of the visual cues that contribute to depth perception and discomfort. A full, rigorous treatment of the subject can be found in a recent publication by Brenner and Smeets [14].

Our sense of depth perception and 3D shape is created from both 2D and 3D cues. 2D or *monoscopic* depth cues can be observed even by those lacking binocular vision. These cues include [15, 58, 66]:

1. **Lighting/shading:** shadows and highlights can depict object roundness and surface texture, as illustrated in Figure 2.1;



(a) flat shading; is it a circle or a sphere? (b) shadows and highlights allude to a sphere

Figure 2.1: Monoscopic depth cues: lighting and shading

2. **Linear perspective, relative size and familiarity:** linear perspective is an important depth cue, as are relative size and familiarity. For example, suppose an image contains a pea and an apple, but the pea is the same size as the apple, viewers would expect the pea to be closer than the apple. Similarly, in Figure 2.2 the woman appears to be as tall as the Eiffel tower, yet we know this is not true because we are familiar with the relative size of a human

to the Eiffel tower. We also know from linear perspective that objects appear smaller as they approach the horizon. Hence, we know the tower must be far behind the woman.

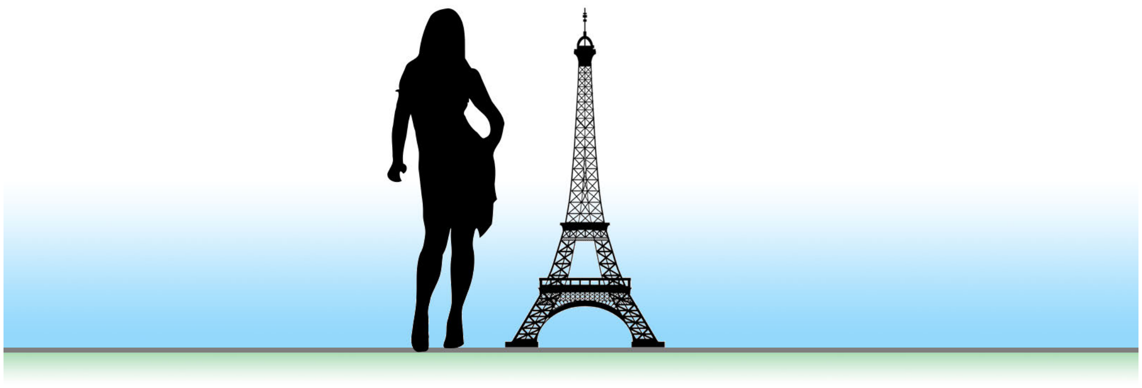


Figure 2.2: Monoscopic depth cues: relative size and understanding

3. **Occlusion:** objects that obstruct the view of others are closer to the viewer, as illustrated in Figure 2.3—the seagull occludes the railroad tracks, indicating it is closer to the viewer;

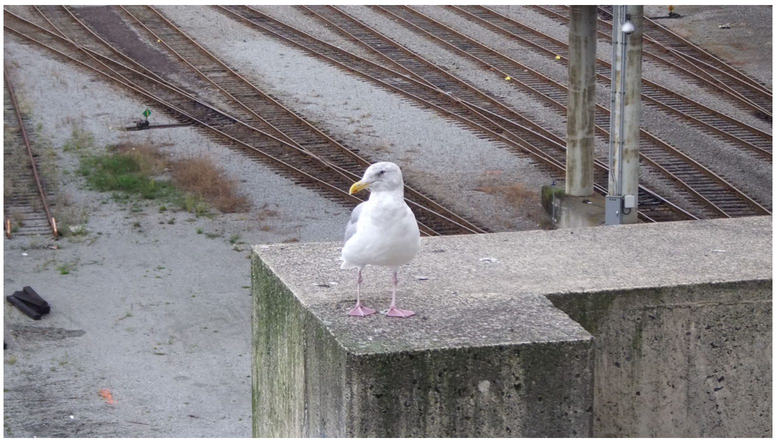


Figure 2.3: Monoscopic depth cues: occlusion

4. **Motion:** an object's movement can indicate its depth; for example, if an object is moving towards the viewer, it will appear to become larger and move faster than an object moving away from the viewer.

Individuals lacking binocular vision or those with visual impairments such as strabismus or amblyopia rely on monoscopic depth cues to perceive depth [24]. However, these clues alone may be insufficient to discern object shape, position and depth, as illustrated in Figure 2.4. Note that with white walls, ceiling and floor and uniform lighting it is not possible to determine if we are observing the outside of a convex 3D shape, or the interior corner of a room.

The primary stereoscopic 3D depth cue is parallax—the apparent separation between a point visible to both eyes. Parallax can be observed with a simple experiment. Close your right eye and hold your thumb out at arms length in front of it. Note the position of your thumb with respect to the surroundings, then close your left eye and open your right. Notice how your thumb

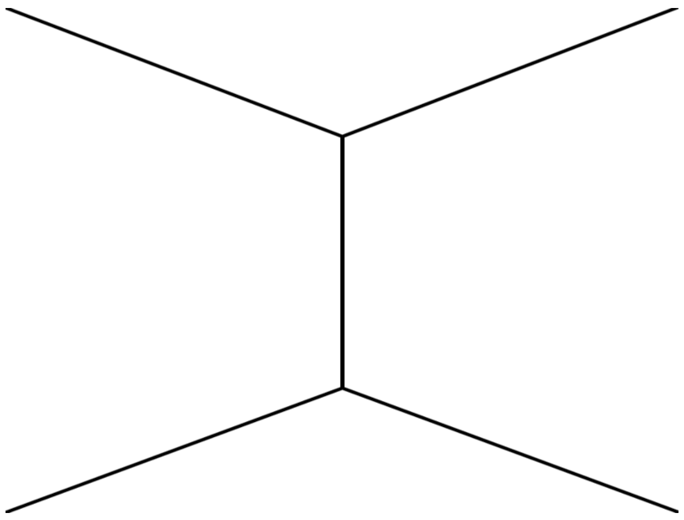


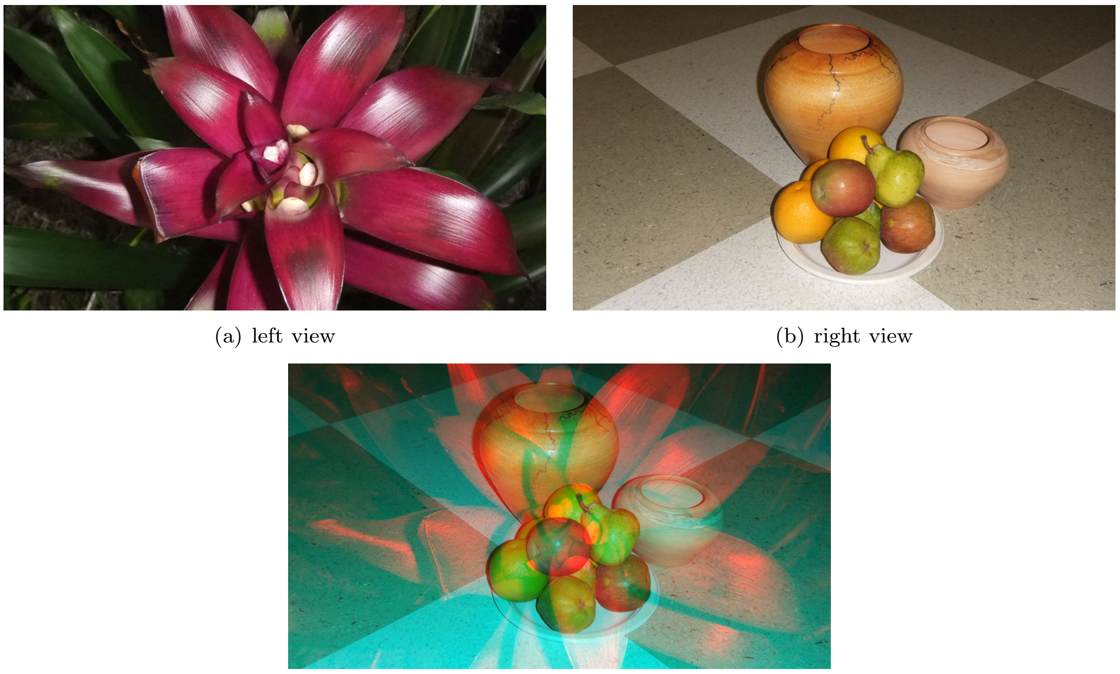
Figure 2.4: When monoscopic depth cues are insufficient.

appears to have moved? The horizontal distance your thumb appears to move is parallax. The brain interprets these small shifts as depth [88]. However, thus far the human visual system is only known to be able to perceive the depth of objects that are less than 248m away [58, 63]. The maximum possible depth is determined by the interocular distance.

In addition to parallax, *vergence* and *accommodation* assist with depth perception at distances less than 10m [4, 63, 68]. *Convergence* refers to an inward rotation of the eyes to focus on an object (Figure 2.17), while *divergence* is the outward rotation of the eyes to focus on an object. *Accommodation* is the eye's ability to bring an object into focus and works together with convergence to estimate depth. For these short distances, if convergence does not match the expected accommodation, the resulting mismatch is referred to as the *vergence-accommodation conflict.* This conflict can cause viewer eye strain as they try to refocus on the object at an unexpected position [58]. However, it is an issue only when viewing S3D media on a flat surface because objects are depicted at a specific depth when in fact they live on the surface.

Hence, when producing S3D media it is critically important to consider the capabilities of the human visual system. Objects with large parallax, requiring large convergence or divergence motions, can produce eye strain. Additionally, interfering with the vergence-accommodation expectations of the human visual system can cause eye strain. To reduce the likelihood of viewers experiencing these issues, a general rule is used for capturing S3D media—the parallax should not exceed 2.5% – 3% of the length of the diagonal of the screen [58].

Another type of discomfort observed in S3D media is called binocular rivalry. Binocular rivalry occurs when the left and right views differ significantly, or at contours or in luminance or contrast. Instead of fusing the images, the left and right views are seen alternately, which can be disorienting and painful [13]. Figure 2.5 gives an artificial example of binocular rivalry in which the left and right eyes see completely different images. This technique of inconsistent views has been used for artistic effect at the expense of audience comfort, for example in the 2011 short film "Chiral Eyes" by Willy Le Maitre. A shot from "Chiral Eyes" is shown in Figure 2.6.



(a) left view (b) right view

(c) stereo anaglyph

Figure 2.5: Inconsistent left and right views.

2.2 A Brief History of Stereoscopic 3D

The ancient Greeks had a basic understanding of vision and the perception of 3D but were not able to create stereo images or viewing devices. It was not until the 1800s that stereo images and viewing devices became reality. Sir Charles Wheatstone, an English scientist, began experimenting with the creation of stereoscopic 3D drawings in 1830. By 1833, he had developed the first *stereoscope*, a device that used mirrors to deliver separate images to each eye in order to re-create the stereoscopic 3D effect [89]. In 1838, he released a paper that provided detailed ink drawings of his stereoscope along with a number of stereoscopic 3D drawings, as seen in Figure 2.7 [88]. Soon afterward others began working on their own stereoscopes as Wheatstone's design was expensive and not practical.

Stereoscopic 3D photography was introduced in the 1840s. The first S3D photos were taken using a single-lens camera, capturing one view, shifting the camera horizontally, then capturing the second view. However, two-lens S3D cameras were quick to market [89] and stereo photography became vastly popular (Figure 2.8).

At the Great Exhibition in London in 1851, Sir David Brewster and Jules Dubosq introduced an improved stereoscope for displaying 3D photographs (Figure 2.9). It won the favour of Queen Victoria, and the public followed suit. Within five years the stereoscope was popular as far away as America, where Oliver Wendell Holmes created a cheaper, more effective stereoscope that could be found in nearly every household by the 1940s.

The first era of S3D came to an end with WWII and the popularization of film, but it returned in what is called the "golden era" of 3D, the mid-1950s. In the 1950s, film theatres were losing audiences quickly as televisions were appearing in many homes. To lure audiences back into theatres, nearly everything was tried, including S3D. Many films were captured and shown in S3D, including: Alfred Hitchcock's 1954 film "Dial M for Murder" and Andre DeToth's 1953 film "House of Wax". However, the era was short-lived. Other technologies, such as widescreen, were not only

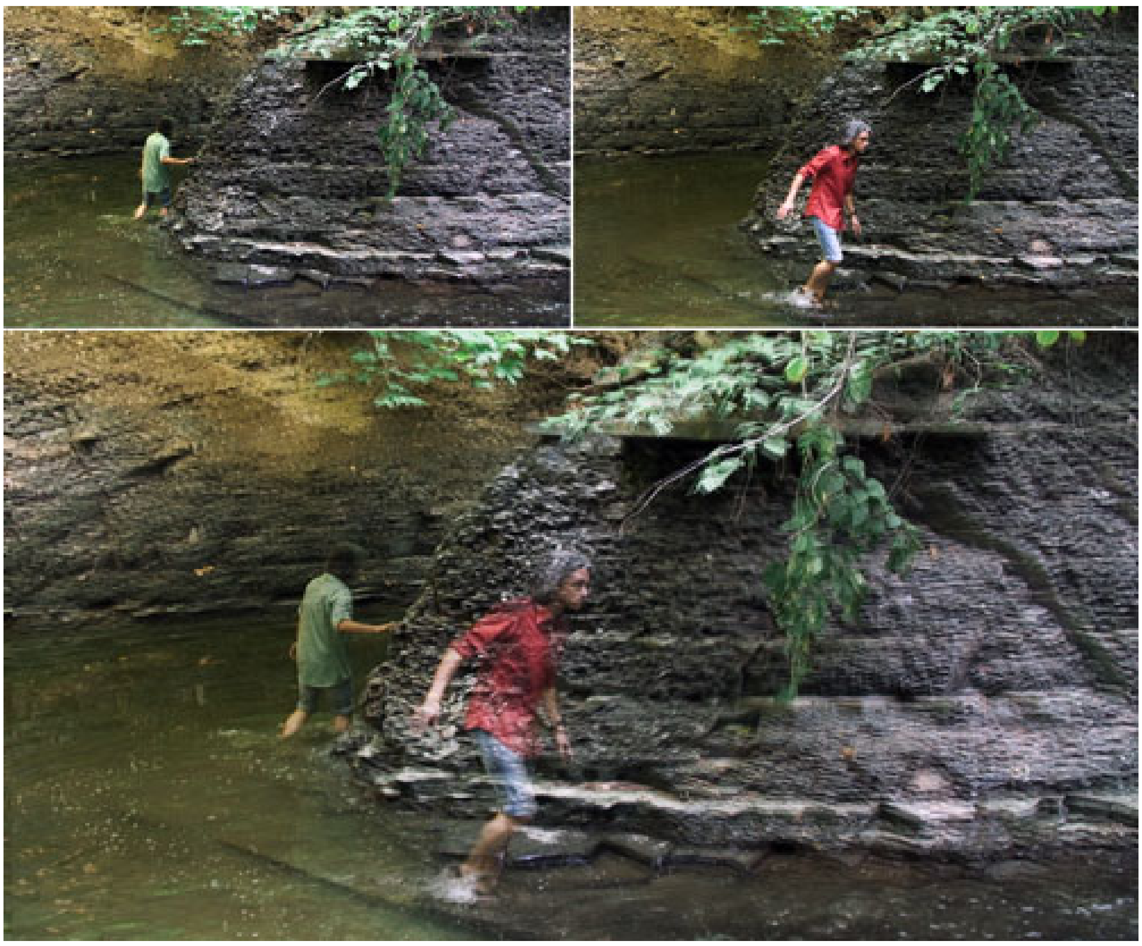
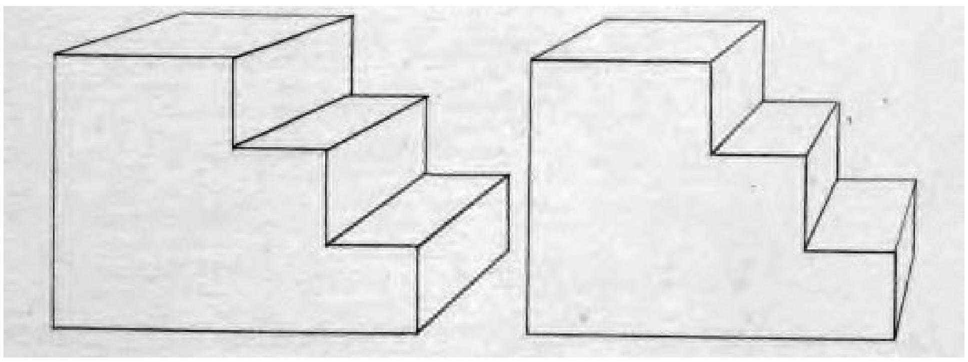


Figure 2.6: Inconsistent views in the 2011 short film “Chiral Eyes” by Willy Le Maitre.

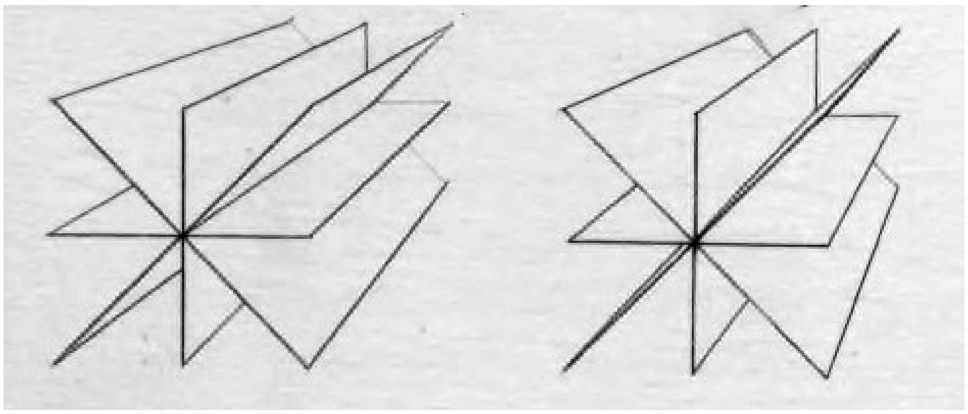
cheaper to produce and project but also did not introduce viewer discomfort.

A short revival of S3D was experienced in the 1970s when a new technology was introduced that allowed both views of an S3D film to be stored in a single reel instead of two. Each 35mm frame now stored left and right views, horizontally scaled and side-by-side. Anamorphic lenses were then used to stretch the views back out for viewing. This simplification to the projection process eliminated many projection issues that caused audience’s discomfort. However, the S3D films of this era were primarily horror and adult films. The most notable films of this era include Morrissey and Warhol’s 1974 film “Flesh for Frankenstein” and the most successful 3D film of all time (in terms of box office sales per budget dollar spent), Silliman’s 1969 adult film “The Stewardesses”.

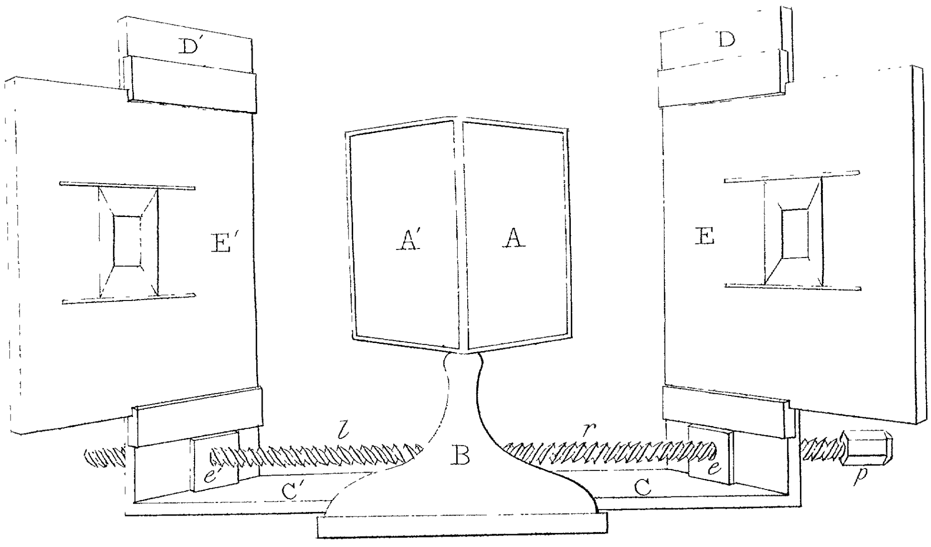
S3D did not return to mainstream film until the early 2000s, when children’s films such as the 2005 film “Chicken Little” were released in theatres in S3D. But it was the huge success of James Cameron’s 2009 film “Avatar” that brought S3D back into popularity. In the early 2010s television, camera, cell phone, and video game manufacturers jumped on the S3D-at-home bandwagon. However, today it is nearly impossible to find S3D TVs, cell phones, or cameras in retail stores. The most recent era of S3D has faded with the rise of technologies such as High-Frame-Rate (HFR), High-Dynamic-Range (HDR) and 4K. These technologies improve the visual quality of the video by increasing the temporal resolution, contrast, and image resolution. They tend to be less expensive, and do not require viewers to wear uncomfortable glasses.



(a) S3D drawing of stairs



(b) S3D drawing of extruded starburst



(c) Diagram of the The Wheatstone Stereoscope

Figure 2.7: Stereoscopic 3D drawings and a diagram of the Wheatstone Stereoscope from Wheatstone's paper [88].

2.3 Stereoscopic 3D Media Production

To capture an S3D photo or video, two views are needed, one for each of the left and right eyes. There exist a number of methods for capturing or creating these views. The first S3D photos

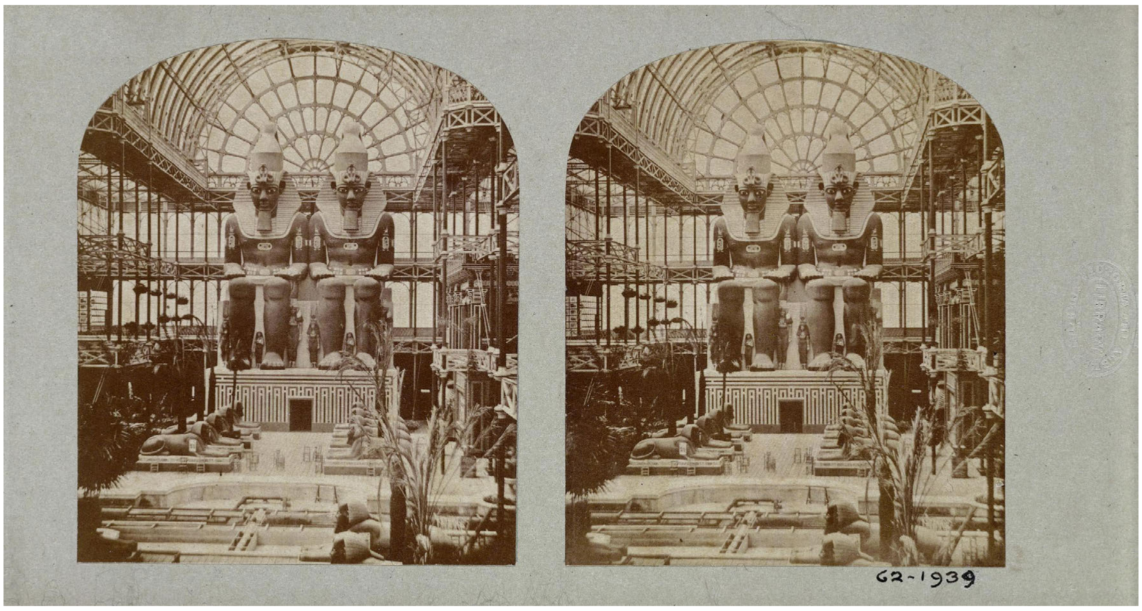


Figure 2.8: The Two Colossal Statues at Crystal Palace. Thomas R. Williams, 1854. Photo courtesy of the Victoria and Albert Museum, London.

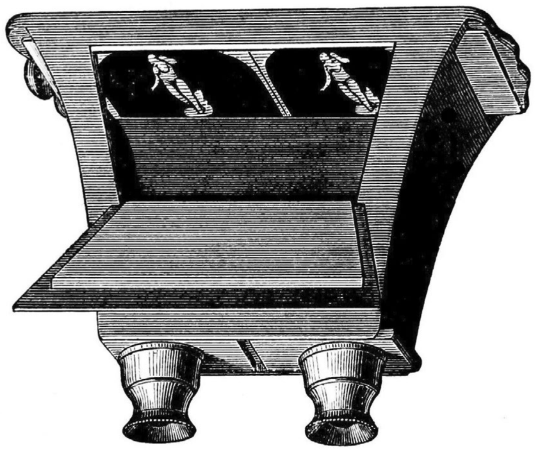
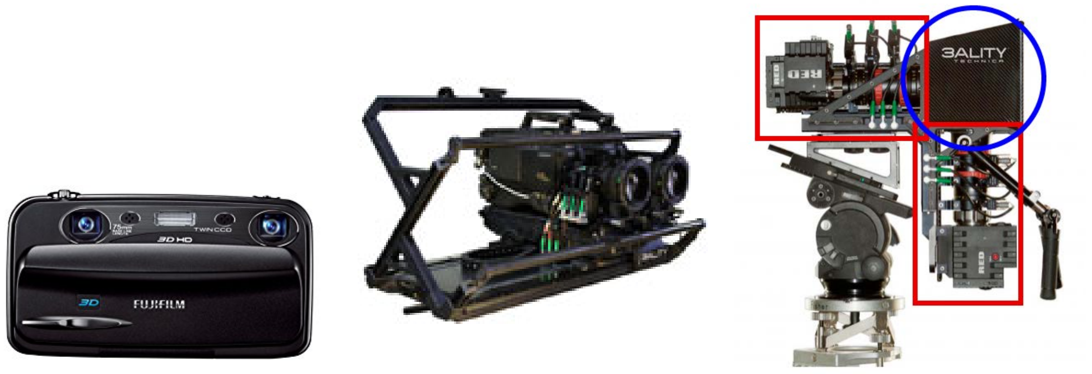


Figure 2.9: The Brewster Stereoscope, by Sir David Brewster.

were taken with a single-lens camera. First the left view was captured, the camera was shifted horizontally, and the right view was captured. This approach can be used with any camera. However, it works only for still life photographs. This method cannot capture action or be used to capture film because the subject(s) may move in between capturing left and right views.

S3D cameras were released shortly after the introduction of S3D photography. They have two lenses, each connected to a capture plate/surface. The lenses are separated horizontally by the *interaxial distance*, which is similar to the interocular distance. Alternatively, an S3D camera can be created with a pair of cameras arranged and held in a multi-camera tripod or rig. In a *side-by-side* rig the two cameras sit parallel and can be moved horizontally to vary the interaxial distance.

Another rig, called a *beamsplitter*, arranges cameras perpendicular to each other while a 50/50[1] mirror is used to deliver the correct image to each camera [58]. Figure 2.10 illustrates the common types of S3D cameras.



(a) Fuji FinePix REAL 3D W3: two lenses, two capture surfaces, fixed interaxial.

(b) 3ality Technica TS-4: side-by-side rig, two cameras, adjustable interaxial.

(c) 3ality Technica TS-5: beamsplitter rig, two perpendicular cameras, adjustable interaxial.

Figure 2.10: Common types of S3D cameras. Note that the two cameras used in (c) are outlined with red and the position of the 50/50 mirror is outlined in blue.

Unfortunately these S3D cameras do not provide perfect solutions to S3D capture. S3D cameras with fixed lenses do not offer artists the ability to adjust interaxial distance and parallax. Side-by-side rigs allow the interaxial distance to be adjusted, but the minimum interaxial is restricted by the camera size. Beamsplitter rigs allow for free manipulation of interaxial distance, but the 50/50 mirror is costly and introduces polarization artifacts that must be removed later. Additionally, issues of desaturation, inconsistent colour timing, and other imaging artifacts resulting from capture must be cleaned up later [58, 66].

Another common approach for creating S3D media is called *conversion*. Conversion requires only a single view, which can be acquired by any means. This single view is typically considered to be the left view. It is processed to create a second, artificial right view to produce the stereo effect. First, the image is *rotoscoped*, a process where artists trace each object, contour, and silhouette in the scene. Next, artists create an artificial disparity (parallax) map using the traced scene as a guide [76]. Figure 2.11 depicts a scene from the 1993 film "Jurassic Park" that has been rotoscoped [3]. The generated disparity map is used to horizontally shift (or *stereo warp*) the provided left view to a new, artificial right view. Specifically, each pixel (x, y) in the left view has a corresponding pixel in the right view found at $(x-d, y)$, where d is the disparity of pixel (x, y). However, warping creates holes in the right view that correspond to regions of the left view that were occluded. By "moving" the camera, some of these regions are now visible. Artists typically hand-paint, or use inpainting algorithms to fill these holes [3, 76]. Inpainting these holes is discussed in greater detail in Chapters 3 and 4.

2.3.1 Stereoscopic 3D Viewing

There are a number of methods for viewing S3D media. The simplest method, called *parallel viewing*, does not require any special viewing device or screen to achieve stereopsis. Instead, the

[1]50/50 mirrors reflect 50% of light and transmit 50% of light, splitting a single "beam" of light into two

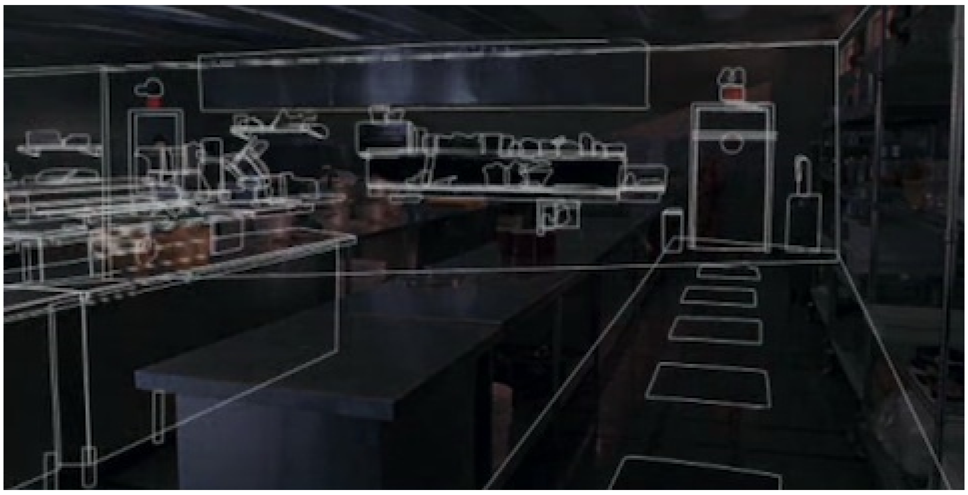
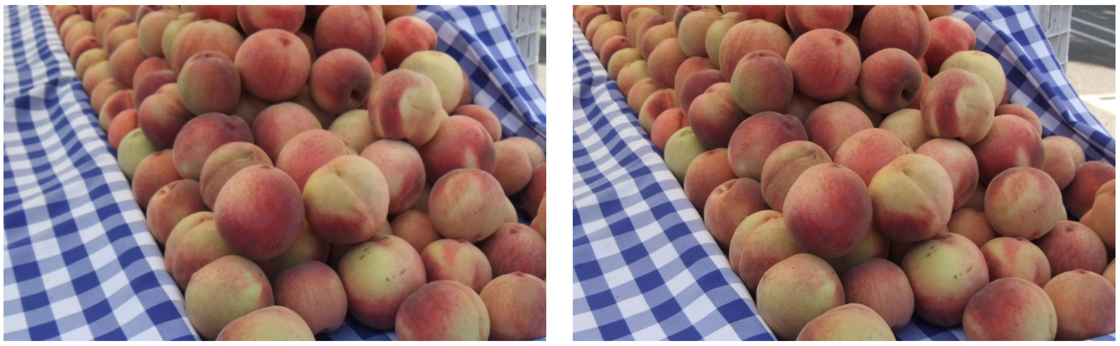


Figure 2.11: Rotoscoped scene from “Jurassic Park” [3].

left and right views are placed side-by-side and the viewer focuses their eyes such that each eye focuses on the intended image. An easier method is called *cross-eyed viewing*, where the left and right views are swapped and viewers fuse by crossing their eyes and focusing on the pair. Figure 2.12 gives an example S3D image arranged for cross-eyed viewing—try to fuse this image by crossing your eyes. Both of these methods are called *free-view*, since they require no apparatus.



(a) right view (b) left view

Figure 2.12: A stereo pair in arranged for cross-eyed viewing.

However, many individuals are unable to achieve stereopsis in this way.

There are many methods to view S3D media using various devices.

1. **Stereoscopes.** A stereoscope, introduced earlier in this chapter, separates left and right eyes via lenses/barriers to display a different image to each eye. Figure 2.13 depicts a View-Master, a variation of a stereoscope introduced in 1939. Unlike previous stereoscopes the View-Master used a wheel of transparent slides to store S3D photos. Each wheel holds seven stereo pairs that the viewer could see in turn by flicking a small switch on the side of the device. The concept of stereoscopes persist today. Simple S3D viewers such as Google Cardboard and Daydream are stereoscopes, while personal augmented/virtual reality systems such as HTC Vive and Oculus Rift operate similarly to stereoscopes.

2. **Anaglyphs.** An anaglyph combines both views into a single image. The left and right

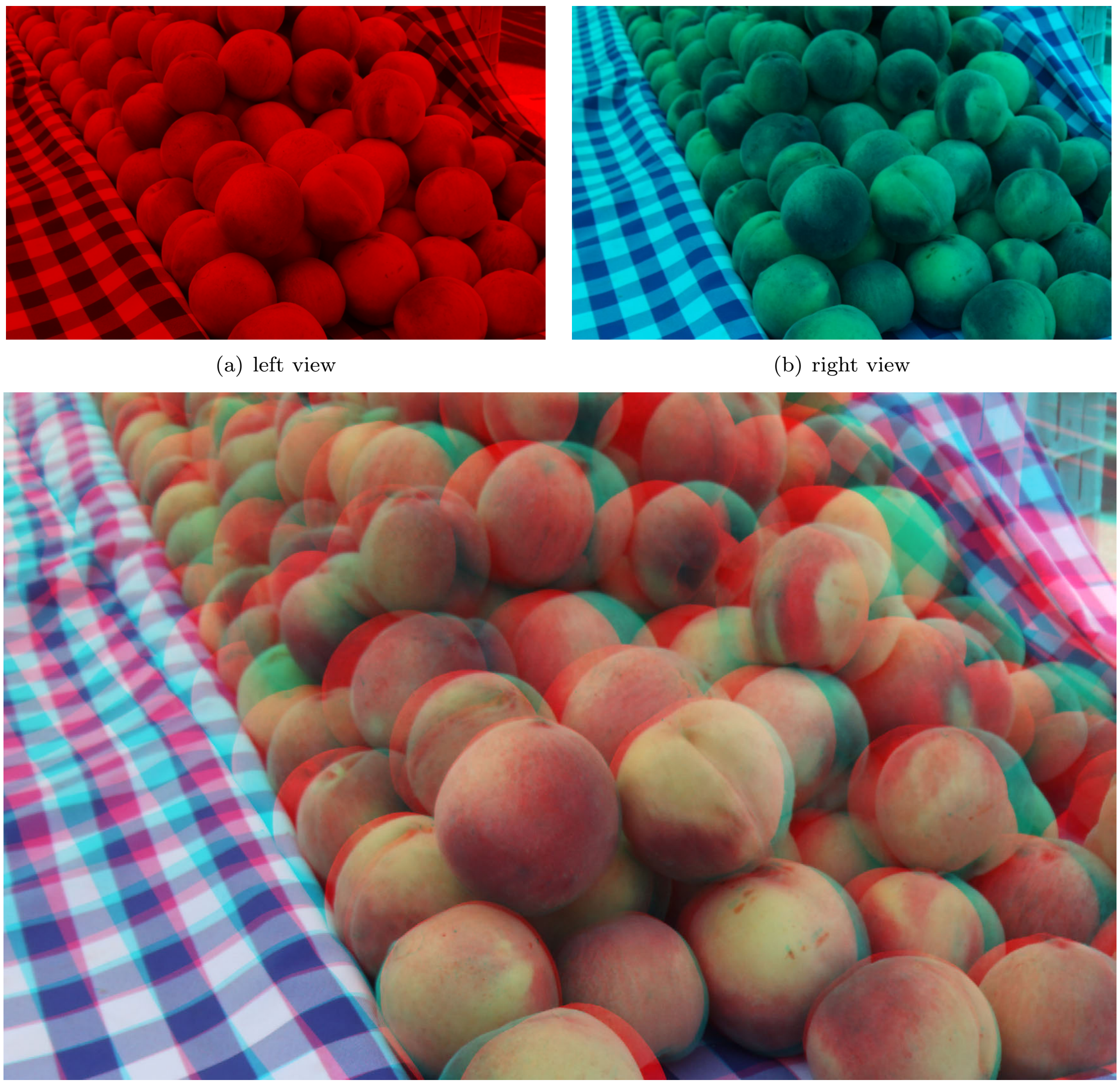
Figure 2.13: The View-Master Stereoscope, a popular childrens' toy until the 1990s.

images are filtered by colour, such that the left view is visible under a lens of one colour (typically red), and the right view is visible under a filter of a different colour (typically blue or cyan). Figure 2.14 provides a sample red/cyan anaglyph of a colour image.

While anaglyphs are easy to produce, they negatively affect colour reproduction due to the colour filtering. They often suffer from *crosstalk*, where the opposing view is faintly visible with the correct view, which can cause discomfort. Crosstalk arises when the colour filter applied to the anaglyph image does not match the colour of the filters in the glasses used for viewing, which is a common issue.

3. **Polarizaton.** Polarization is a form of anaglyph image where left and right views are combined into a single image. However, instead of filtering by colour, different polarization filters, typically circular, are applied to the left and right eyes [58]. Special polarized glasses are required to separate the projected image correctly for each eye. While polarization does not negatively impact image colour, the images appear dimmer than when viewed separately without polarization. This dimness results from each eye receiving only 50% of the projected light and from the darkness of the polarization filters through which the media is viewed. Polarization also suffers from crosstalk, which can worsen considerably if the viewer's head is not level. This technique for viewing S3D content has been in use for over 70 years and is the most common way S3D content is shown in theatres today.

4. **Active View Systems.** Instead of projecting overlapping left and right views and using special filtering techniques to deliver the correct image to each eye, active view systems rapidly alternate between projecting the left and right views. A matching pair of powered LCD glasses black out the view for the eye that is not currently displayed so that the viewer receives both views alternately. The speed of the alternation is fast enough that the brain perceives the left and right views arriving at the same time, permitting stereopsis. However, these systems are expensive, including the glasses, which are very fragile. Additionally, the


(a) left view (b) right view

(c) red/cyan anaglyph

Figure 2.14: Red/cyan colour anaglyph.

speed of the black out alternation tends to darken images.

5. **Autostereoscopic Displays.** Autostereoscopic displays permit 3D viewing without the use of special glasses. There are two types of displays that do not require glasses for viewing: lenticular displays and parallax barrier displays. For both displays, the stereo image is rendered as a single image with alternating columns of image data from the left and right views, illustrated in Figure 2.20.

   Lenticular displays have been around for over 100 years and consist of an array of tiny cylindrical lenses placed over the alternating stereo image. These cylindrical lenses project the left view to the left eye, and the right view to the right eye. Lenticular displays can also be used to project more than two views, to create a smooth animation as one moves.

   Parallax barrier displays use an array of narrow slits instead of cylindrical lenses placed over the alternating stereo image. The slits are positioned such that the left eye will see only the pixels for the left image, and the right eye will see only the pixels for the right image.

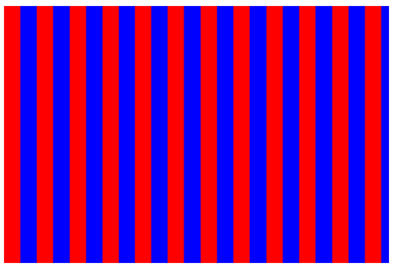


Figure 2.15: Alternating columns of left and right image data for lenticular or parallax barrier displays. Note that red columns are for the left view, and blue columns are for the right view.

These display techniques have been used in cereal box prizes, S3D printed images, autostereoscopic displays on cameras, mobile phones and hand-held video game consoles such as the Nintendo 3DS. However, due to the display technology, the horizontal resolution of the image is reduced.

2.3.2 Media Production Pipelines

There are three phases to the traditional production pipeline for film or photography [1]:

1. **Preproduction.** During preproduction, the piece is planned in great detail with sketches, animations, etc.

2. **Production.** During production, the piece is captured.

3. **Post-production.** During post-production, the captured materials are corrected and arranged into the final product.

Errors can arise during any of these three phases and can have costly impacts on a project. For example, poor planning during preproduction can lead to mistakes during production that are detected and repaired, often at great cost, during post-production.

The S3D media production pipeline is the same as the traditional one. However, the risk of error is significantly higher and each error can be costlier to correct. Next we discuss some of the issues that arise during the S3D production process and how they are handled presently.

2.3.3 Corrections

A number of errors can arise during S3D production due to poor planning and capture or equipment errors.

1. **Lens differences.** Subtle differences between the left and right lenses, such as physical differences in focal length, distortions, polish, inclusions of small particulate matter, scratches, or tinting, can cause inconsistencies between the left and right views. These issues arise during production but are typically not noticed until later on, when the content is viewed. To reduce the probability of differences between lenses, consecutively numbered lenses from the same batch are preferred. However, any remaining artifacts are removed during post-production with tools such as Adobe Photoshop.

2. **Focal differences.** Inconsistencies between left and right views can be caused when the focal lengths of the left and right lenses differ. This problem was more noticeable with older S3D cameras because it was difficult to set the focal lengths identically. This problem is less common today due to mechanisms that can precisely set the focal length. Unfortunately, when this issue occurs it is not easily repaired.

3. **Vertical parallax.** Horizontal parallax creates depth. Vertical parallax causes objects in one view to be higher than in the other, which inhibits stereopsis. This issue arises when one lens or camera is higher than the other and is often corrected by applying a vertical shift and cropping both views. A stereo pair exhibiting vertical parallax is given in Figure 2.16. Note how the vermillion flower is higher in the right view than in the left.

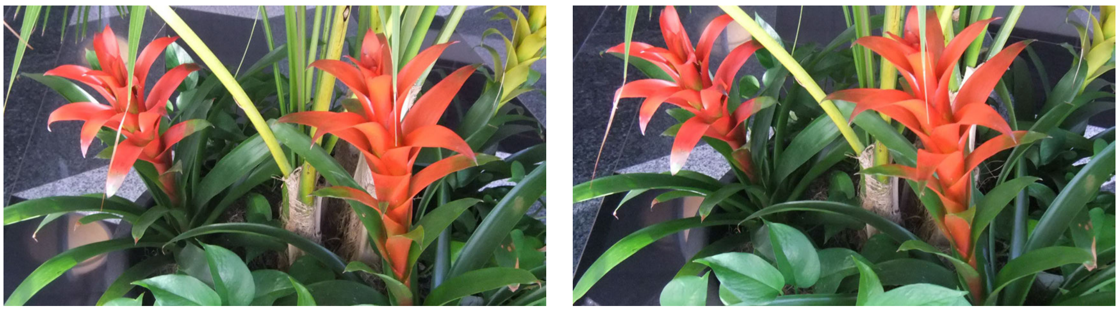

(a) left view (b) right view

Figure 2.16: A stereo image exhibiting vertical parallax.

4. **Keystoning.** Keystoning is a common artifact seen with projected images. It presents itself as a trapezoidal distortion of the image. Keystoning is a common problem for S3D media. If one angles the cameras inwards such that their optical axes overlap, a convergence point is created as demonstrated in Figure 2.17. The convergence point is where parallax is zero and it represents the object the viewer should focus on. However, since the cameras are no longer parallel, the images exhibit keystoning. Keystoning introduces vertical parallax at the top and bottom of the images. An example of keystoning is given in Figure 2.18. Note how objects appear to have different scales between the left and right views. Also note the vertical parallax near the edges. Keystoning can be repaired via "spatially-varying vertical warping", applying varying vertical scales based on horizontal position [53].

5. **Rotational differences.** It is possible for the left and right views to be at an angle to each other, as demonstrated in Figure 2.19. These rotational differences between left and right views arise when the capture or display surfaces are not level. Regardless of the source of rotational differences, the S3D image may cause discomfort. This can be corrected by rotating one or both views until they are level with each other and cropping. Vertical and rotational parallax are typically removed during a process called *rectification.*

6. **Too small or too large interaxial distance.** If the interaxial distance of the camera is too large then the parallax may also be too large for human eyes to accommodate. This causes eye strain and difficulty fusing. If the interaxial distance is too small, the parallax will likely be comfortable. However, the image may not produce the desired depth effect—i.e., objects will appear more flat than round. This issue typically arises due to poor planning or inexperience. This issue is not easily repaired and is discussed in greater detail in Chapter 3.

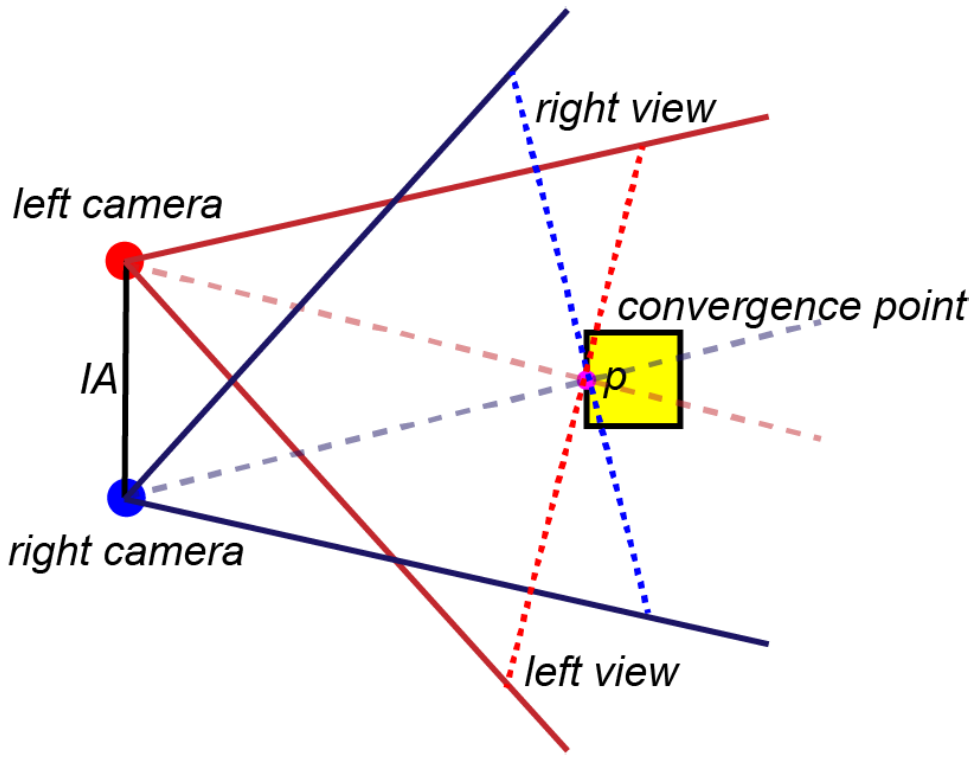


Figure 2.17: Stereo convergence. The left and right views passing through the point of convergence exhibit zero parallax. IA is the interaxial distance.

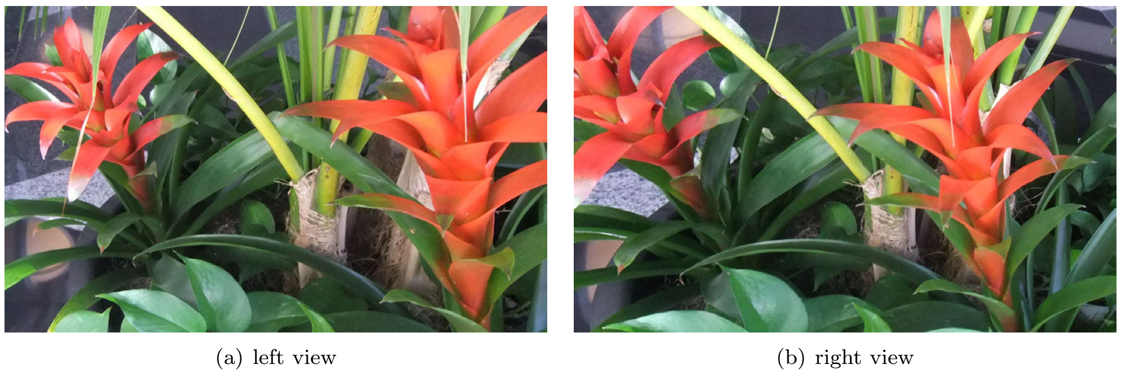


Figure 2.18: A stereo pair exhibiting keystoning.

7. **Polarization.** When using a beamsplitter rig with a 50/50 mirror, light is polarized differently for each view. This causes one view to see reflections, such as those on the surface of water. The other view will not see reflections, instead seeing through the water. This inconsistency causes binocular rivalry because the scene is not rendered identically in both eyes, leading to viewer discomfort. To remove these polarization artifacts, special lens filters known as quarter-wave retarders are used during capture to alter the polarization such that both views will match [7].

8. **Exposure differences.** When using a dual capture system it is possible that the exposures may differ, causing one image to be brighter than the other as demonstrated in Figure 2.19. Exposure can be corrected or equalized using tools such as Adobe Photoshop.

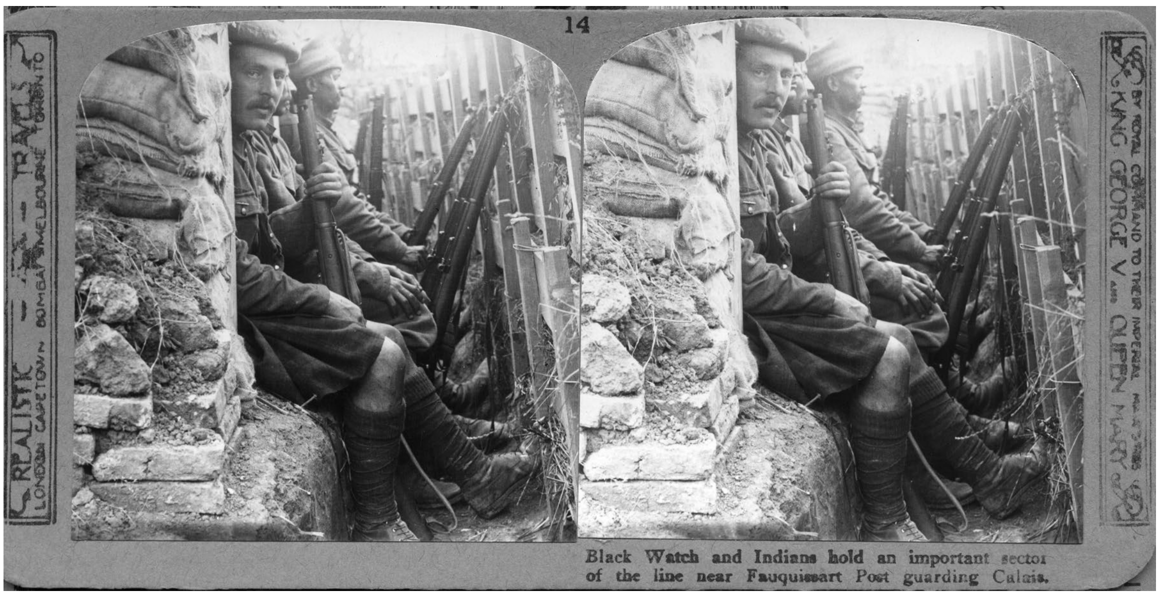

14
REALISTIC TRAVELS
LONDON CAPETOWN BOMBAY MELBOURNE TORONTO
BY ROYAL COMMAND TO THEIR IMPERIAL MAJESTIES KING GEORGE V AND QUEEN MARY
Black Watch and Indians hold an important sector of the line near Fauquissart Post guarding Calais.

Figure 2.19: A stereo image exhibiting different exposures. Note that this stereo card also exhibits minor rotational differences since the printing of the views on the card is not parallel. Image courtesy of Jonathan Kitzen.

9. **Scratches and damage** Scratches or damage to lenses, capture surfaces, or media introduce inconsistencies that may cause discomfort. These can be removed with tools such as Adobe Photoshop.

10. **Proscenium arch violations** The *Proscenium Arch* is the stereoscopic border of the screen through which S3D content is viewed [58, 60]. It is also called the *stereo window.* If objects sit in front of this window at the left or right border, they will be visible only to one eye. This is called a *window violation* and it can cause binocular rivalry. To "solve" this issue, the window can be moved forward such that offending objects are now behind the window. Although this does not change the object geometry, by moving the stereo window we change the apparent depths of objects, which changes the visual story. Alternatively and more commonly, the offending objects are cropped to the window so that they do not exceed it and the object is visible to both eyes. This has the unfortunate side effect of reducing the horizontal resolution of the media.

Stylization

Once the S3D image has been rectified and corrected artists may apply stylization filters to create ambiance. For example, artists might add depth-of-field, film grain, or apply a painterly style as seen in the 1998 monoscopic film "What Dreams May Come". These effects are typically added using standard industry tools such as Nuke, Adobe Photoshop, Adobe Premier, and Adobe After Effects. However, filters in these tools are not typically designed with S3D in mind. Therefore, to apply a filter to an S3D image, one must apply the filter separately to the left and right views. Hence, the left and right views often contain mismatched and inconsistent stylization that causes discomfort [70]. This thesis explores the consistent application of stylization filters to S3D media as described in Chapters 4 and 5.

2.4 Working with Stereoscopic 3D Media

S3D rectification and cleanup of lens dependent side effects is for the most part a fairly straightforward process that can be performed with many existing, even automatic tools. However, other corrections such as adjustments to the interaxial/parallax and stylization are not easily perfomed. These operations require stereo awareness via a parallax or *disparity map*.

2.4.1 Disparity

A stereoscopic 3D image consists of two images called the *left view* and the *right view*, corresponding to the left and right eyes. For each of these views there is an associated *disparity map*.

The value of each pixel in a disparity map is the distance one must move left or right to find the matching pixel in the other image. This distance is called the *disparity*. Let z represent the depth of a pixel p, and IA be the baseline or interaxial distance. Let f be the focal length, and d, the disparity. Then, disparity d is inversely proportional to depth z, as shown in Equation 2.1. Disparity is also known as parallax.

$$z = (\mathrm{IA}f)/d \tag{2.1}$$

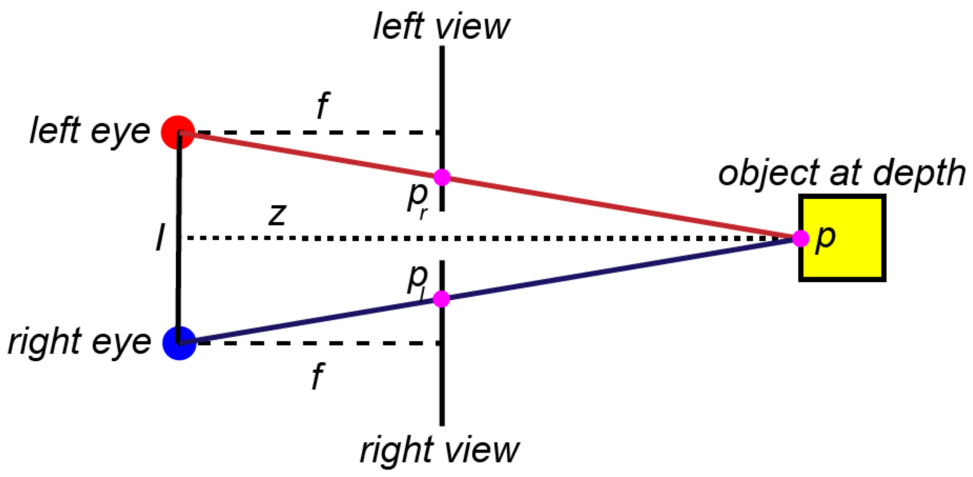
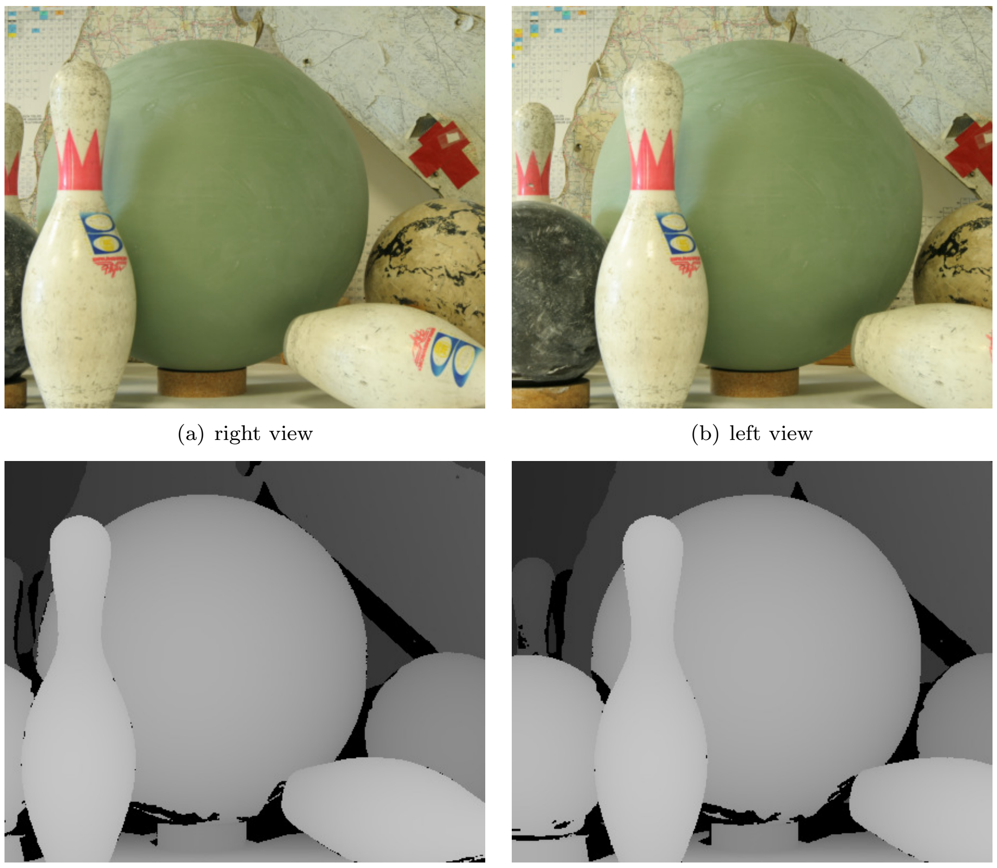


Figure 2.20: The relationship between disparity and depth. I is the interocular distance and can be used interchangeably with IA. The disparity of point p is $p_l - p_r$.

If a point in the scene is visible in both views, d_l is the disparity of pixel (x_l, y) in the left view, and d_r is the disparity of the pixel (x_r, y) in the right view, then: $(x_l - d_l, y) = (x_r, y)$ and $(x_r + d_r, y) = (x_l, y)$, that is, $d_l = d_r$. However, if this equation does not hold or the disparity is undefined, then: the pixel may be occluded or not visible in the other view, or, the disparity of the pixel may be incorrect.

Disparity maps are usually provided as grayscale images (Figure 2.21). The intensity of the pixel is the disparity, which is typically scaled so that it fits in a particular range. If the image is converged, or has been adjusted so that objects at a particular depth have zero disparity, then the corresponding disparity map may contain both positive and negative values. In these cases, an offset is often required to be added to get the true disparity of any pixel. Scale factors and offsets must be provided with the disparity map in order to use it. The majority of disparity maps


(a) right view (b) left view

(c) right disparity map (d) left disparity map

Figure 2.21: A stereoscopic 3D image pair with its right and left disparity maps. In these disparity maps, objects closer to the viewer are brighter than objects that are farther away. Black regions correspond to undefined disparity. Image courtesy of The Middlebury Dataset [73].

use eight bits to encode each pixel, limiting the number of unique disparities to 256. This can be limiting for large images or where very accurate, smooth disparity is desired.

Disparity maps typically come from one of several sources:

1. The underlying model, if the image is of a synthetic scene. For example in an S3D raytracer the interaxial distance, focal length and depth are known so the disparity can be calculated.

2. Structured light, where a known pattern is projected onto the scene and captured. The differences between the pattern and its projection over the scene are used to calculate disparity [73]. The disparity maps shown in Figure 2.21 were produced using structured light.

3. Depth-capturing cameras, such as the Lytro light field camera use an array of lenses and a depth sensor to capture the intensity of light, and its direction in a scene. Such cameras not only produce a depth map, but also provide adjustable focus after capture.

4. Correspondence-based computer vision algorithms such as GraphCut, which attempt to find matching regions of pixels between left and right views [37, 69].

Models and structured light can produce accurate, high-resolution disparity maps. Depth-capturing cameras produce maps that are typically accurate but noisy and low resolution. Vision algorithms vary greatly in how well they do on a particular scene, but results are often noisy, particularly around surface boundaries as demonstrated in Figure 5.4.

Disparity maps derived from vision algorithms often contain undefined values that correspond to areas in one image that are not visible in the other because of occlusion or being near the left or right edges of the image. Because these occluded objects are visible only in one view, there is often no way to determine their disparities and therefore depths. This also occurs in disparity maps derived from structured light in regions where the light did not reach, as seen in Figure 2.21.

The capture and computation of high quality disparity maps is an active area of research in computer graphics and computer vision. These methods focus on finding matching pixels or regions from the left image in the right one [37, 69]. However, a match is not always possible, even when human eye can find it. For example, Figure 2.22 has a large, over-exposed snowbank that confuses stereo matching algorithms. Similarly, a bright sky, or more generally, a large, solid-coloured region can confuse stereo matching algorithms because it is difficult to determine which pixel or block of pixels matches when there are multiple, equally good choices. This low-texture issue has been the focus of many recent papers in stereo matching [37].

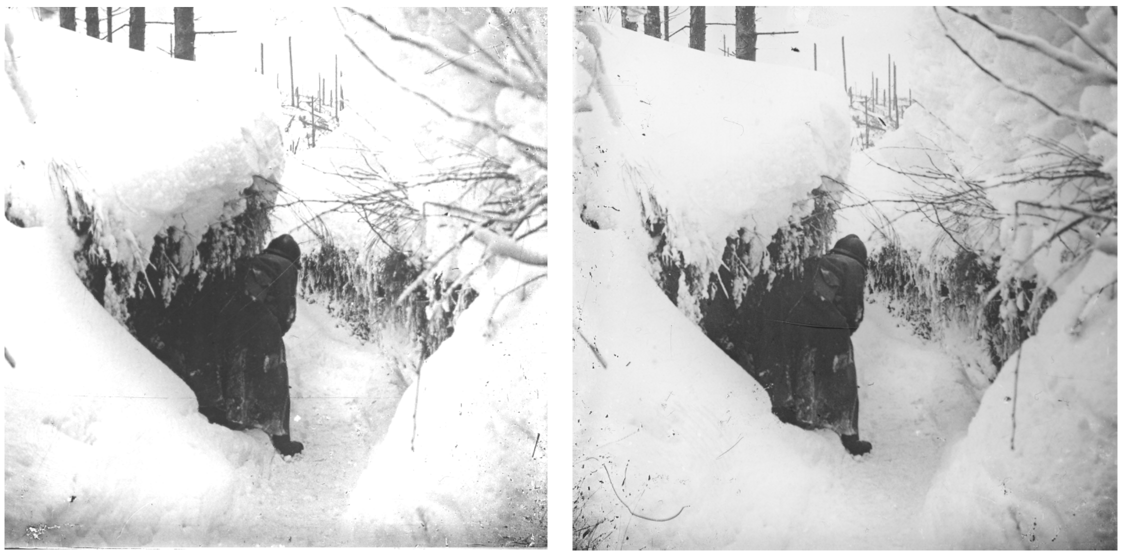

(a) left view (b) right view

Figure 2.22: S3D pair exhibiting exposure differences and damage. Note how the left view has higher contrast and is brighter than the right view. The blown-out snowbank is void of texture, making it difficult for vision algorithms to determine disparity. Also note the view-dependent damage seen in the lower left corner of the right view. Image courtesy of Jonathan Kitzen [45].

2.4.2 Hybrid Approaches to Acquiring Disparity Maps

When structured light or other depth capturing techniques are not available, disparity maps are produced by computer vision algorithms. However, as shown in Figure 2.23, these methods can produce unsatisfactory results. Specifically, note how the disparity regions do not correspond to the regions of the rose. Instead, the smooth surfaces of the petals are found to be very noisy. In these cases, a multi-phase method may be used to produce a usable disparity map:

1. **S3D image cleanup and correction.** Computer vision algorithms to generate disparity maps work best when the S3D media is rectified and does not exhibit keystoning or differences

in colour, exposure, or focal length. Additionally, the S3D media should not contain view-dependent artifacts such as scratches.

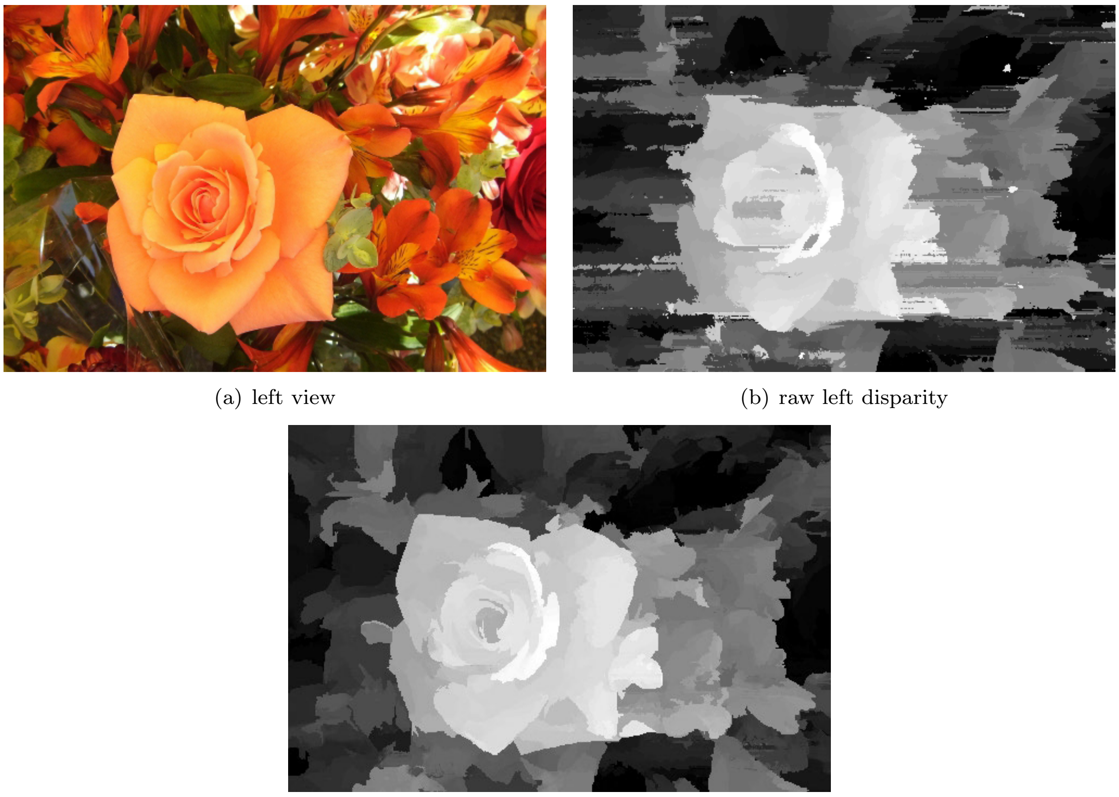
Removing keystoning, colour, exposure and focal differences along with view-dependent artifacts is done first. This enables automatic rectification algorithms such as those found in StereoPhoto Maker and OpenCV to produce more accurate results.

2. **Generate disparity map.** Once the S3D image is rectified and cleaned up, computer vision algorithms can be applied to produce the initial disparity map.

3. **Automatic correction of the disparity map.** There exist automated methods to correct disparity maps. Specifically, these methods aim to preserve object edges and remove noise. In work peripherally related to the content of this dissertation, I developed an algorithm for processing disparity maps to preserve edges and reduce noise [2]. This patented work corrects disparity maps using segmentation and statistical analysis. The method is based on the assumption that a region of the image, where colour does not vary greatly would correspond to a region where depth also does not vary greatly, because lighting would cause strong shadows or highlights if it did. Using this observation, we segmented the images by colour and observed the disparities in the matching regions of the disparity map. Statistical methods and histograms were then used to identify any pixels that were unlikely to be correct and new disparity values were interpolated from nearby, correct pixels to repair them. This method significantly reduced noisy object edges and outlier disparities, as shown in Figure 2.23.

4. Manual correction of the disparity map. Even after applying automated methods for disparity map correction, especially poor maps may still require manual correction. Manual correction involves hand-painting and applying inpainting algorithms to carefully repair or fill regions of the map and is typically performed in an image processing tools such as Photoshop.

Each of the methods described in this thesis requires the use of disparity maps and assume they are available or can be computed for a stereo pair. In Chapter 3, we work with hundred-year-old S3D photos that are damaged and our cleanup process for the images and disparity maps is described in detail.


(a) left view (b) raw left disparity

(c) corrected left disparity

Figure 2.23: Automatic correction of disparity using our algorithm [2]. The raw disparity map was produced using belief propagation [80].

Chapter 3

Adjusting Disparity

3.1 Introduction

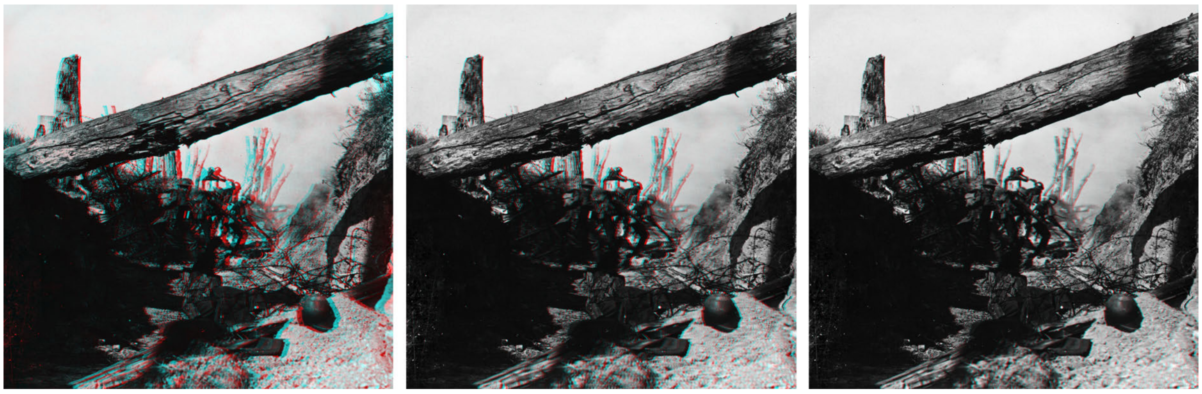


Figure 3.1: World War I stereoscopic 3D photo from the 2014 documentary "Soldiers' Stories" directed by Jonathan Kitzen [45]. Left: original photo; middle and right: disparity reduced by 60% and 80% respectively using our method. Red-cyan anaglyphs.

Stereoscopic 3D (S3D) images have left and right views, one for each eye. Objects in each view appear offset in the other view, and these fixed disparities (horizontal parallax) create the perception of depth. If the disparity at any point in the image is too large, users may experience discomfort when viewing it on certain screen sizes [40, 49]. For example, the S3D photo taken during WWI and presented in Figure 3.1 causes viewers discomfort when viewed on an IMAX screen [45]. To make this image comfortable for viewers, the disparity had to be reduced. The reduced-disparity version of this S3D photo was included in the 2014 documentary "Soldiers' Stories", directed by Jonathan Kitzen [45].

In many cases creators can address visual discomfort by re-shooting with different S3D parameters [40]. However, as our WWI photos are 100 years old, re-shooting was not an option. In general, it may be impossible or infeasible to re-shoot some subjects, like sporting events or movies with expensive or unavailable performers. Another common approach used to address visual discomfort is to use one of the original S3D views as the basis for a 2D-to-3D conversion [58]. However, this conversion process is time-consuming and expensive.

The creator may also wish to alter the disparity of S3D content for artistic reasons. For example, animated films may use a multi-rig technique, where each object in the scene is rendered

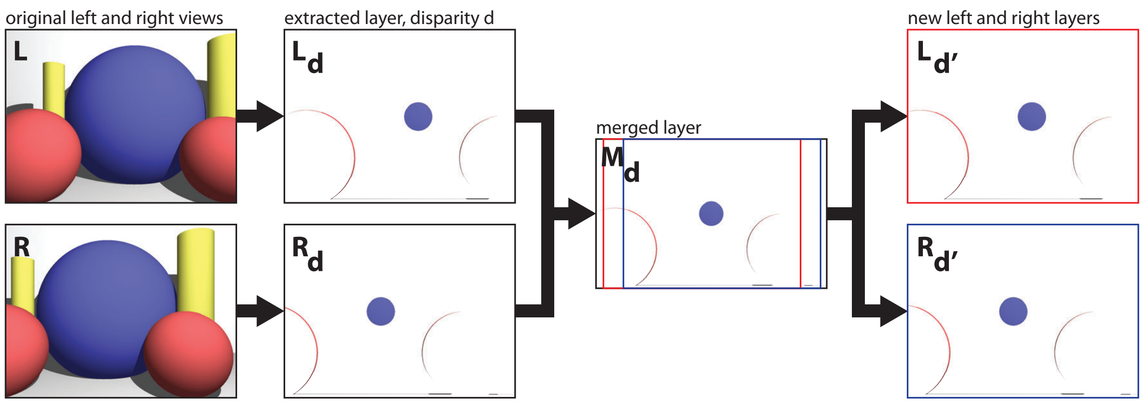

original left and right views
L
R
extracted layer, disparity d
L d
R d
merged layer
M d
new left and right layers
L d′
R d′

Figure 3.2: The process of extracting a layer with a given disparity d from left and right views, merging that data into a single layer, then re-creating new left and right views with a new disparity. Note how the left arc in $R_{d'}$ incorporates pixels from L_d.

with different S3D camera parameters, allowing the filmmaker freedom to flatten or deepen certain objects in the scene to elicit emotional responses from the viewer [72].

In this chapter we present a method to adjust the disparity of an S3D image non-uniformly using *disparity layers* instead of re-shooting, warping or stereo-converting the image. Since our method may introduce holes by exposing previously occluded regions in the image, we also present a disparity-aware inpainting method to fill these holes. We test our method by comparing our adjusted results to raytraced S3D ground truth images. Finally, we discuss the adjustment process of several images in “Soldiers’ Stories”.

3.2 Previous Work

Recent algorithms for adjusting disparity involve warping functions and algorithms for inpainting the holes that appear when previously occluded regions of the image become visible. Lang et al. [50] presented a method to address S3D content viewing discomfort and adjust disparity non-uniformly with a stereoscopic warping function. Wang et al. [85] built upon this method to produce a 2D-to-3D conversion warping function that preserves object edges cleanly. Although the warping functions used in both methods prevent holes from opening, they do so by distorting regions of the image—squashing and stretching features to fill those regions where a hole would appear. These distortions are noticeable in regions where the disparity changes rapidly or where it has been adjusted significantly.

Furihata et al. [25] also use warping functions, this time to generate synthetic views between left and right views for Free-viewpoint Television (FTV). Their warping functions introduce holes in the synthetic view that correspond to occluded regions in the original view. This method is also limited to reducing disparity uniformly across the image, so it cannot be used for artistic multi-rig disparity adjustments. Pan et al. [64] use a similar approach to improve viewing comfort. These approaches focus on adjusting disparity uniformly and filling holes with the lowest disparity adjacent the hole on the same scanline.

Manap et al. [55, 56] describe a method to synthesize new views that decomposes S3D images into layers by disparity range (disparity layers) and re-composites the layers at new positions to produce the new view. Their method aggregates multiple disparities into compound layers, which may flatten objects and produce a cardboard-cutout appearance. Furthermore, they do not consider non-uniform disparity adjustment or methods for filling holes opened by the adjustment process.

Most disparity adjustment methods require a secondary algorithm to fill regions that were previously occluded and are now visible. Notable S3D inpainting algorithms were presented by Wang et al. [84] and Morse et al. [59]. Both methods focus on filling holes resulting from object removal. Wang et al. use RANSAC plane fitting and texture synthesis to fill the hole. Morse et al. use a partial differential equation and a modified PatchMatch algorithm [6] to fill holes. Both approaches use the disparity of the removed object as a reference for which regions of the image can be sampled.

However, holes opened by adjusting disparity are not generated by the removal of a specific object but the movement of several objects, so the appropriate reference disparities are not known. Additionally, while both methods attempt to minimize inconsistencies introduced by the inpainting algorithm, neither guarantees consistency.

3.3 Approach

Our S3D disparity adjustment method has three phases:

1. **Decomposition.** The left and right views of the S3D image are decomposed and merged into layers using the disparity maps, which we assume have been provided and are free from noise or inconsistencies. Unlike previous work by Northam et al and Manap et al, each layer only contains pixels of a single disparity [56, 61, 62].
2. **Adjustment.** Adjust the disparity by using user-specified formulas or values to separate the merged layers into new left and right layers.
3. **Assemble.** Composite the new adjusted layers for left and right views into their final images.

3.3.1 Adjusting the Disparity with Disparity Layers

We start by creating a set of disparity layers. Specifically, let S be a stereoscopic 3D image composed of left and right views (L, R) and let the corresponding disparity maps be D_L and D_R. Let (w, h) be the width and height of L and R. Let $d_{x,y}$ be the disparity of a pixel $p(x, y)$ in L or R, and D be the set of distinct disparities that occur in L and R. We will change the disparity for some members of D. For each such member, call the original disparity d and the new disparity d'. Let D' be the set of adjusted disparities d'. Note that the mapping from D to D' need not be monotonic, i.e., if $d_i > d_j$, then d'_i may be greater than, equal to, or less than d'_j. This may cause a re-ordering of layers. Additionally, for a uniform reduction in disparity, the original disparities d can be multiplied by a scalar factor, α, where $0 \leq \alpha \leq 1$. To adjust the disparity of the S3D image (Figure 3.2):

1. For each d, construct two layers L_d and R_d, consisting of the pixels in L and R that have disparity d. Then combine L_d and R_d into a merged image M_d with dimensions $(w + d, h)$. Composite L_d into M_d at the left edge, and R_d into M_d at the left edge plus d. For pixels that overlap, combine the values by linear interpolation, so that the leftmost pixels are primarily from L_d and the rightmost primarily from R_d. This blending of L_d and R_d smooths any inconsistencies in colour, lighting and grain between L_d and R_d.

2. For each d, create new layers $L_{d'}$ and $R_{d'}$ as follows:
   Let:
   $$\alpha = (d - d')/2. \tag{3.1}$$
   Then the new left layer $L_{d'}$ and right layer $R_{d'}$ are
   $$L_{d'} = M_d[\alpha, w + \alpha] \tag{3.2}$$
   $$R_{d'} = M_d[\alpha + d', w + \alpha + d']. \tag{3.3}$$
   Where $M_d[x_0, x_1]$ represents the region of image M_d from column x_0 to x_1. The creation of a new left and right layer is illustrated in Figure 3.4.

3. After constructing all the new layers, composite the $L_{d'}$ layers together to create the adjusted left view L'. Composite the $R_{d'}$ layers together to create the adjusted right view R'. Composite in order, from back to front.

Since the new left and right views L' and R' are constructed from the same layers, this approach forces consistency in the adjusted image pair even if the original L and R were not consistent.

Note that if a pixel p is in both L_d and R_d, the colour of the combined pixel in merged layer M_d is calculated by linear interpolation using the pixel's distance from the left and right edges of M_d. Thus the colours of pixels closer to the left edge of M_d are more heavily influenced by colours in L_d than those in R_d, and vice-versa. It is common for left and right views of an S3D image to differ slightly in exposure and lighting, and our blending method preserves some of these qualities nearest the left and right edges while guaranteeing consistency between L and R. Additionally, this blending technique ensures that there are no sudden changes in colour in the merged layers due to a pixel existing only in the left or right image. Previous disparity adjustment techniques using layers choose the average of $L_d(p)$ and $R_d(p)$, which guarantees consistency, but may lose some of the desired lighting/exposure traits and produces sudden changes in colour at the boundary where L_d and R_d overlap [56]. A visualization of our technique compared to Manap et al. is given in Figure 3.3 [56].

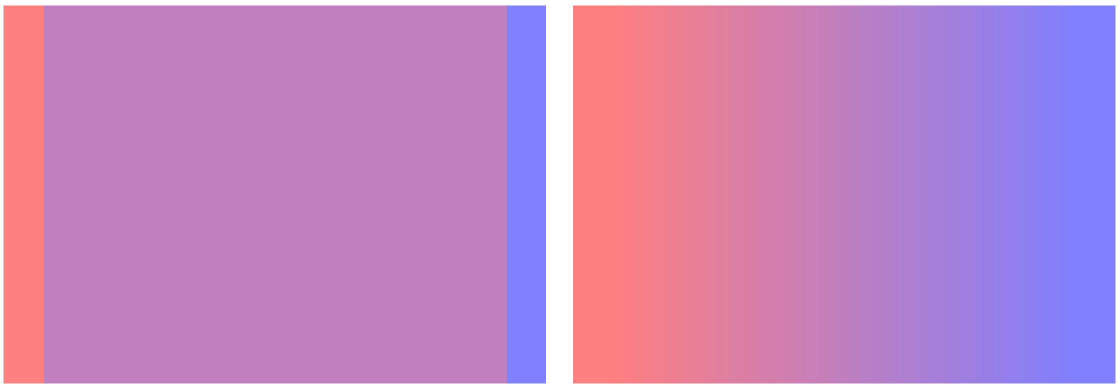

(a) Layer blending approach of Manap et al. [56]. (b) Our approach.

Figure 3.3: Comparison of layer blending approaches. Pixels from the left layer are red, and the right layer are blue. The discrete boundaries produced by Manap et al.'s method in (a) are not present in our approach (b).

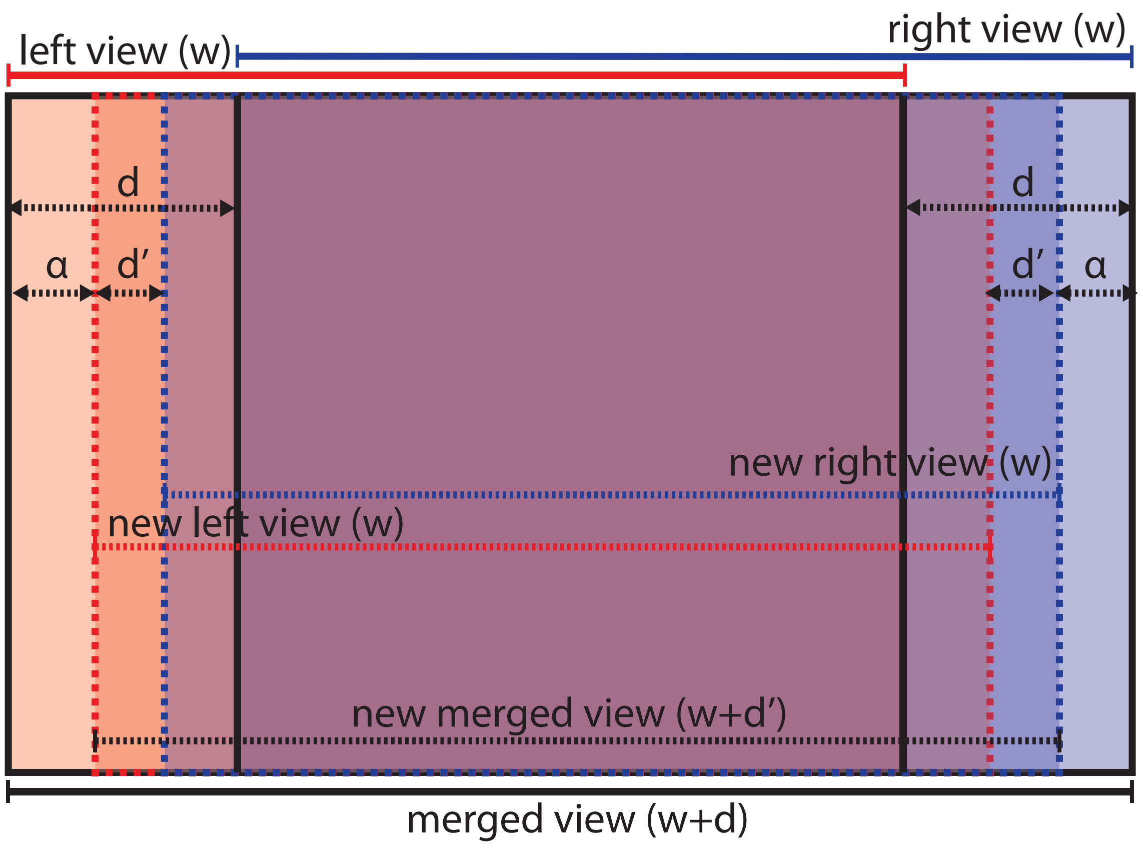


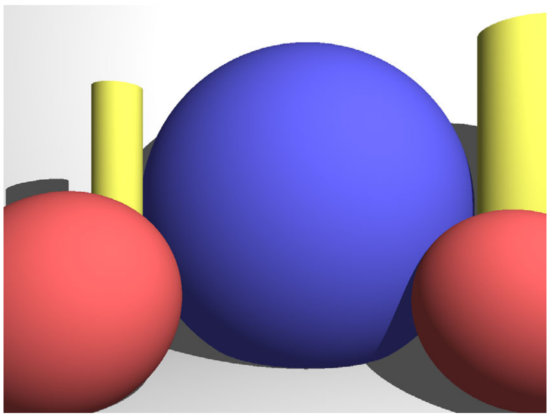
Figure 3.4: Merged left and right layers (M_d, L_d, R_d) with new left and right layers $(L_{d'}, R_{d'})$. The left layer is red, and the right is blue. Note how the new left and right layers have the same width as the original left and right views; only the disparity changes.

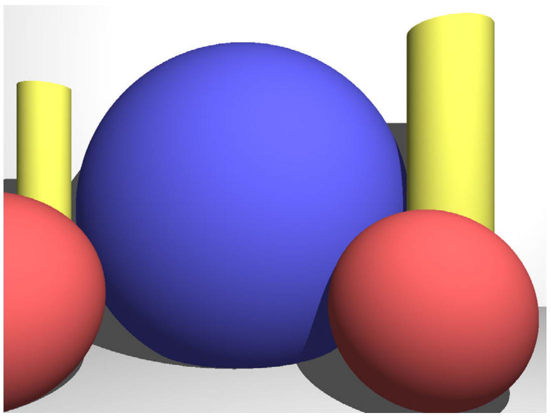
3.3.2 Holes

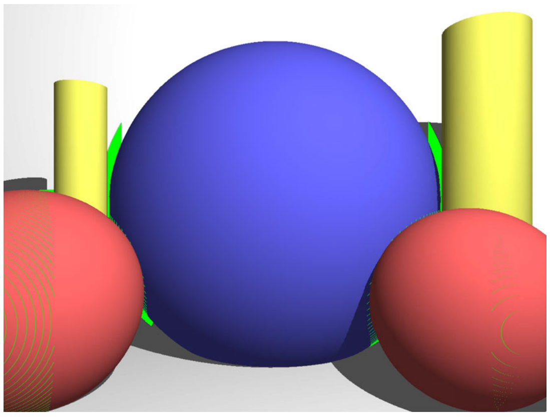
The disparity-adjusted L' and R' views contain unfilled regions (holes) that correspond to areas of the 3D scene that are no longer occluded (Figure 3.5(c)). These holes could be filled by hand with a digital painting tool, but this is a time-consuming and costly process. Instead, holes in L' and R' could be filled independently with an automatic inpainting algorithm such as PatchMatch [6], or a fluid-dynamics based algorithm [10, 81]. However, manipulating stereo views independently usually introduces inconsistencies that can cause viewers discomfort [70]. Additionally, general inpainting algorithms are not disparity-aware and may fill holes with colours and patterns belonging to objects at vastly different depths, as demonstrated in Figure 3.5(d). Note how the hole opened between the yellow cylinder and the blue sphere has been filled with yellow, blue and even red from the left red sphere, but the region should be filled with white from the background.

Another approach to filling holes in L' and R' is to fill the corresponding holes in the disparity map, then use a disparity-aware inpainting algorithm to complete the images. This approach is taken by Wang et al. [84] and Morse et al. [59]. However, unlike these approaches, we do not know the disparity of the occluding object.

If the holes were filled in the merged layers, M_d, prior to constructing the final disparity-adjusted left and right views L' and R', then the fills would be consistent in L' and R' [61, 62]. Additionally, applying an inpainting algorithm to holes in the individual merged layers ensures


(a) Original left view.


(b) Original right view.


(c) Synthesized view with holes shown in green.

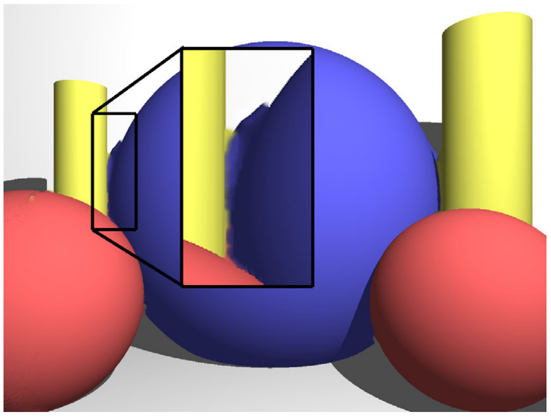

(d) Filling without knowledge of disparity.

Figure 3.5: While adjusting disparity, regions of the image that were previously occluded become visible. These holes need to be filled to complete the image. Note that (d) illustrates the filled hole in a magnified inset.

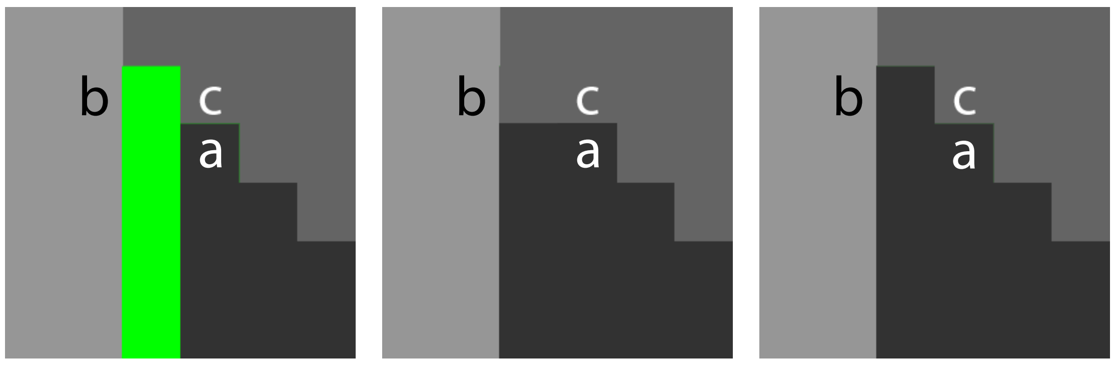
that reference data for the inpainting algorithm is relevant to the current disparity. However, in order to fill the holes in M_d, the disparities of the pixels in the holes must be known to determine which M_d layers require filling.

3.3.3 Inpainting the Disparity Map

Holes occur where previously occluded areas become visible. Therefore, the disparities of pixels in a hole correspond to neighbouring background disparities [25, 55, 56].

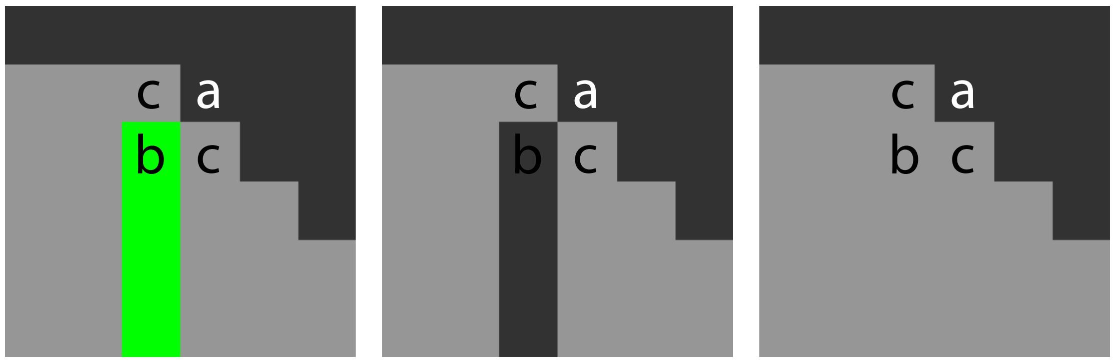
A common approach to filling these holes is to choose the smallest scanline-adjacent disparity [25, 64]. However, this method horizontally smears incorrect disparities when multiple objects are adjacent to the hole, as demonstrated in Figure 3.6. In these situations, the correct disparity value for a hole pixel may be in the scanline above or below it.

Choosing the lowest 8-neighbour adjacent disparity works in most situations, except when filling thin holes in foreground objects where one pixel of the hole is adjacent to the background. In these cases the background disparity, which is adjacent to a single pixel in the hole, is propagated into the object (Figure 3.7). To prevent this, we count the number of times the smallest adjacent disparity $d_{\min}$ occurs within a square neighbourhood with radius r of the current pixel p. If the number of smallest adjacent disparities is less than a threshold τ (percent of known neighbours),



(a) Disparity map with a hole indicated in green.
(b) Filling the hole with the lowest scanline disparity.
(c) Correct fill.

Figure 3.6: Filling a hole in the disparity map with the lowest scanline disparity. Note the scanline containing disparities **b** and **c**; because the true lowest disparity (**a**) is not located on the same scanline, the disparity **c** is chosen instead, creating a smeared appearance. For this image and all others, darker shades represent lower disparities (farther away from the viewer).



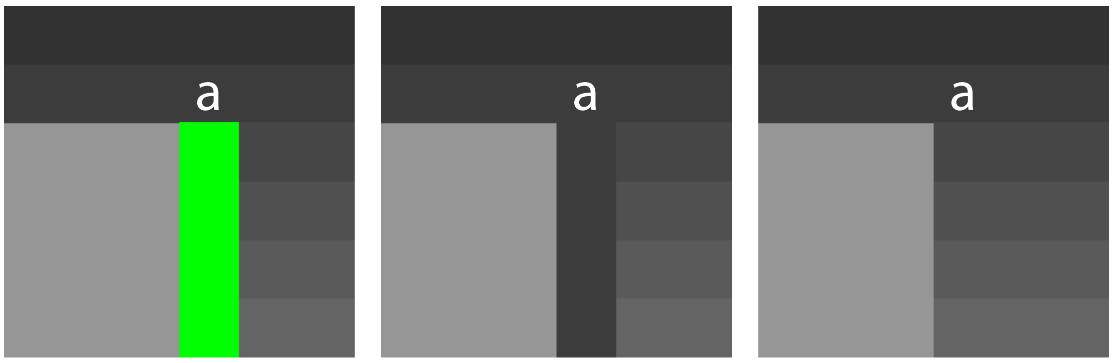
(a) Disparity map with a hole indicated in green.
(b) Hole filled with lowest-adjacent disparity.
(c) Hole filled with correct value.

Figure 3.7: Choosing the lowest adjacent disparity with $r = 1$ and $\tau = 12.5\%$. The pixel with disparity **b** is located inside a foreground object with disparity **c**, but the lowest adjacent disparity is **a**. If only one of the adjacent disparities corresponds to the minimum, choose the next smallest value, in this case **c**.

then we set d_{min} to be the next smallest disparity. In our experiments we used a radius $r = 2$ (a total of 25 pixels), and a threshold $\tau = 12.5\%$, eliminating cases where the lowest disparity occurs in only one of the eight immediately adjacent pixels.

However, filling each pixel in the hole with the selected minimum disparity d_{min} works only when the surface is squarely facing the camera. If the hole is adjacent to an angled surface, once inpainted it will be flat, filled with a single value and facing the camera (Figure 3.8). Ideally, the inpainted hole would blend smoothly into the surface. To fit the surface we use linear least squares plane fitting to calculate an appropriate disparity for each pixel in the hole, using the pre-calculated d_{min} values to select appropriate reference points. Specifically, to fill pixel $p(x, y)$ whose minimum adjacent disparity is d_{min}:

1. Let S be all pixels within Δ of $p(x, y)$ with known disparities.
2. Let T be all pixels of S whose disparities are within a threshold k of d_{min}. This step ensures no pixels from the foreground are selected as references for fitting.



(a) Disparity map with a hole indicated in green. (b) Hole filled with d_{min}, in this case **a**. (c) Hole filled by fitting a plane to adjacent disparities.

Figure 3.8: Filling holes in disparity map using linear least squares plane fitting.

3. If $|T| < 3$ then the disparity of $p(x, y)$ is d_{min} since there is insufficient data to fit a plane.
4. If $|T| \geq 3$, then fit a plane to the pixels in T. Let A be a matrix formed by the coordinates of the pixels in T, and b be a vector of disparities of the pixels in T:

$$A = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \qquad b = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix}$$

5. Solve the system of equations to find $\vec{m}$, the coefficients of the plane:

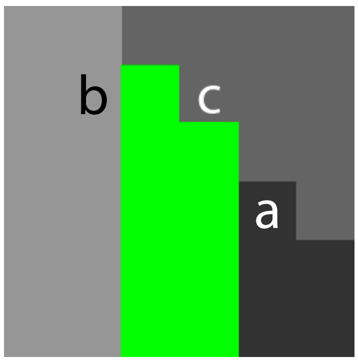
$$A^T A \vec{m} = A^T b$$

6. Then the disparity of $p(x, y)$ is: $(x, y, 1) \cdot \vec{m}$.
7. Repeat for each pixel in the hole until completed.

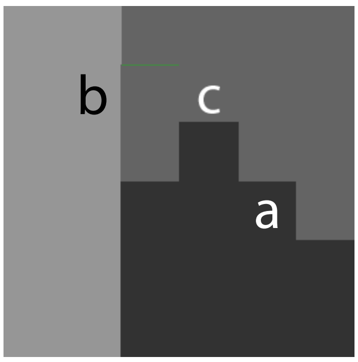
We found that $\Delta = 15$ worked well for our experiments with images at a resolution of 1200×900. We also found that a threshold $k = 5$ worked well with our test images that had a maximum disparity range of 256. Note that selecting smaller values for k reduces the number of pixels usable by the plane fitting algorithm, and that choosing large values for k may cause foreground or extreme background pixels to be used by the plane fitting algorithm.

Filling the disparity of each pixel using a localized linear least squares plane fit preserves surface shape, as demonstrated in Figure 3.8(c). However, consider the case of Figure 3.9, where two objects expose the background. If the pixels are filled top-to-bottom, then when trying to fill the region underneath the overlapping objects there are no background pixels adjacent to the first pixel in the hole (Figure 3.9(a)). Therefore, d_{min} will be the disparity of the mid-ground object instead of the background. The same situation can occur with a differently shaped hole if pixels are filled bottom-to-top (Figure 3.9(d)). This problem can be corrected by filling the holes top-to-bottom (D_{top}) and again bottom-to-top (D_{bottom}) then choosing $\min(D_{\text{top}}, D_{\text{bottom}})$ for each pixel.

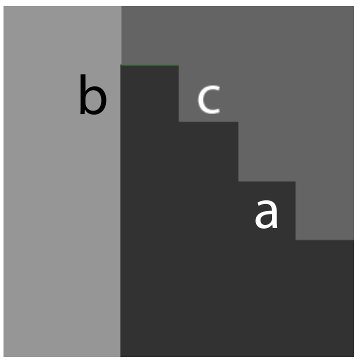
One final improvement can be made. When filling a pixel in a large hole, there may be few pixels within Δ of p to use for plane fitting, as seen in Figure 3.10(a). Smaller holes do not have this



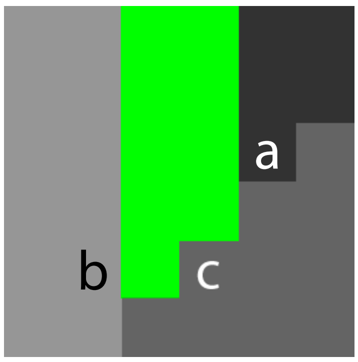
(a) Disparity map with a hole indicated in green.



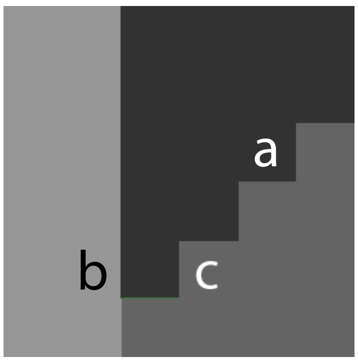
(b) Hole filled top-to-bottom.



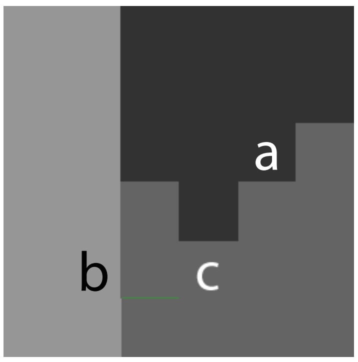
(c) Hole filled bottom-to-top.



(d) Disparity map with a hole indicated in green.



(e) Hole filled top-to-bottom.



(f) Hole filled bottom-to-top.

Figure 3.9: Filling a hole between two surfaces (one vertical object and one angled object). The desired disparity for the entire hole is **a**, but **c** may be incorrectly selected when filling top-to-bottom or bottom-to-top. By filling the hole in both directions and choosing the minimum per-pixel disparity, the correct value will be selected. Note that each scanline is filled left-to-right.

issue because there are more known pixels within Δ of the hole that could be used for plane fitting (Figure 3.10(b)). To reduce the size of the hole to be filled we *slowly* open the hole: we adjust the disparity in small increments, filling holes and repeating until the final disparity is reached. If d and d' are the original and adjusted disparities, then:

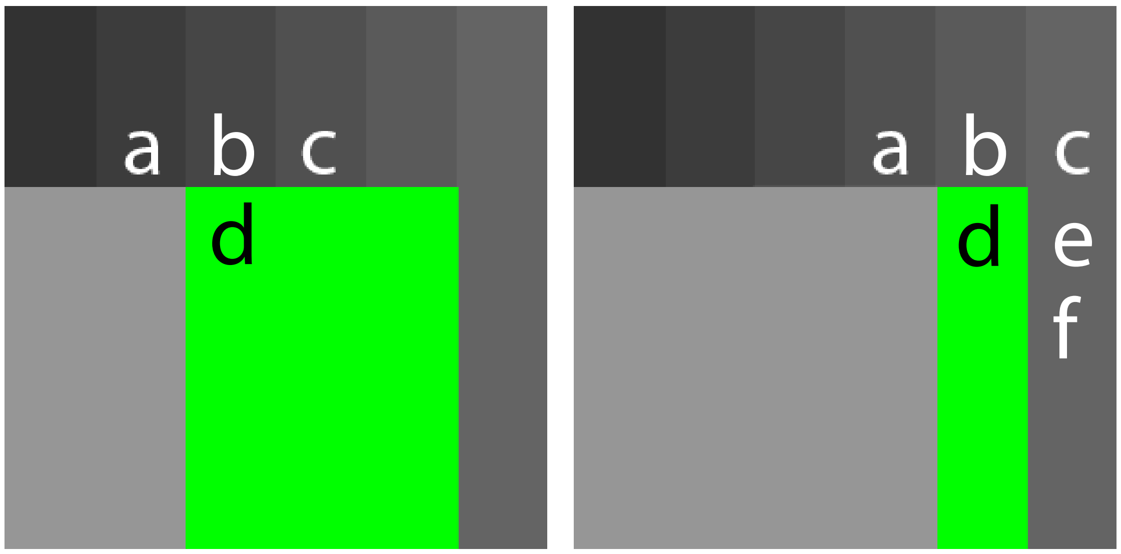
1. Let I be the number of iterations.
2. For each iteration $i \in I$, adjust the disparity by $|d - d'|/I$, filling holes in only the disparity map with each iteration.
3. After all iterations are complete, fill holes in the image layers, and re-create the adjusted image.

In practice, we use 10 to 20 iterations for images with widths between 1200 and 2000 pixels.

This disparity inpainting method can also be used to fill holes in adjusted disparity maps generated by stereo warping, such as those by Furihata et al. [25].

3.3.4 Filling Holes in the Image

Once the holes are filled in the disparity map, the corresponding holes can be inpainted in M_d, the merged image layers, using a stereo-aware inpainting algorithm. We observed two types of holes:



(a) Final adjusted disparity map; pixel **d** has three known neighbours with lower disparities.

(b) Partially adjusted disparity map; pixel **d** has five known neighbours with lower disparities.

Figure 3.10: Filling a hole in the disparity map. Note the number of known neighbours that can be used as references for plane fitting with $r = 1$.

thin cracks or speckles, and large contiguous regions. Thin cracks, typically vertical due to the nature of disparity adjustment, are holes that are only one or two pixels wide, and speckles are small holes up to two pixels in diameter. Cracks and speckles can be filled quickly by interpolating their colours from nearby known pixels. Then, larger, contiguous holes can be filled with a more robust inpainting algorithm, such as PatchMatch [6]. After the holes are inpainted in M_d, the final disparity-adjusted images can be created.

We observed that some M_d layers have small holes and few known values, not enough for the inpainting method to function correctly. Therefore, we include the immediately adjacent layers M_{d-1} and M_{d+1} with M_d to provide additional reference data for the inpainting algorithm.

3.4 Results

To test our method we produced a set of synthetic S3D images with ground-truth disparity maps, as seen in Figure 3.11. We tested the accuracy of inpainting the disparity map by comparing our adjusted disparity maps to the raytraced result using the root mean squared error (RMSE) and average absolute error (AAE) over all pixels (to compare filled holes and adjusted disparity accuracy). However, RMSE and AAE treat all errors the same, which is not the case with S3D image. If an object in the background moves farther back, it is an error, but it is less likely to be noticed by the viewer since it is in the background and not a focus of the image. If an object in the background moves forward, towards the middle or foreground, it is an error, but may be noticed by the viewer since it is closer to the focal objects in the image. To reflect this difference, we introduce a penalized RMSE, PRMSE, which doubles the error for any pixel incorrectly brought forward.

We tested our method by reducing the disparity of each raytraced S3D image uniformly by half. We found that discrepancies between ground truth and our adjusted disparity maps occur in two places: edges and holes. Single-pixel-wide errors occur along the edges of foreground objects.

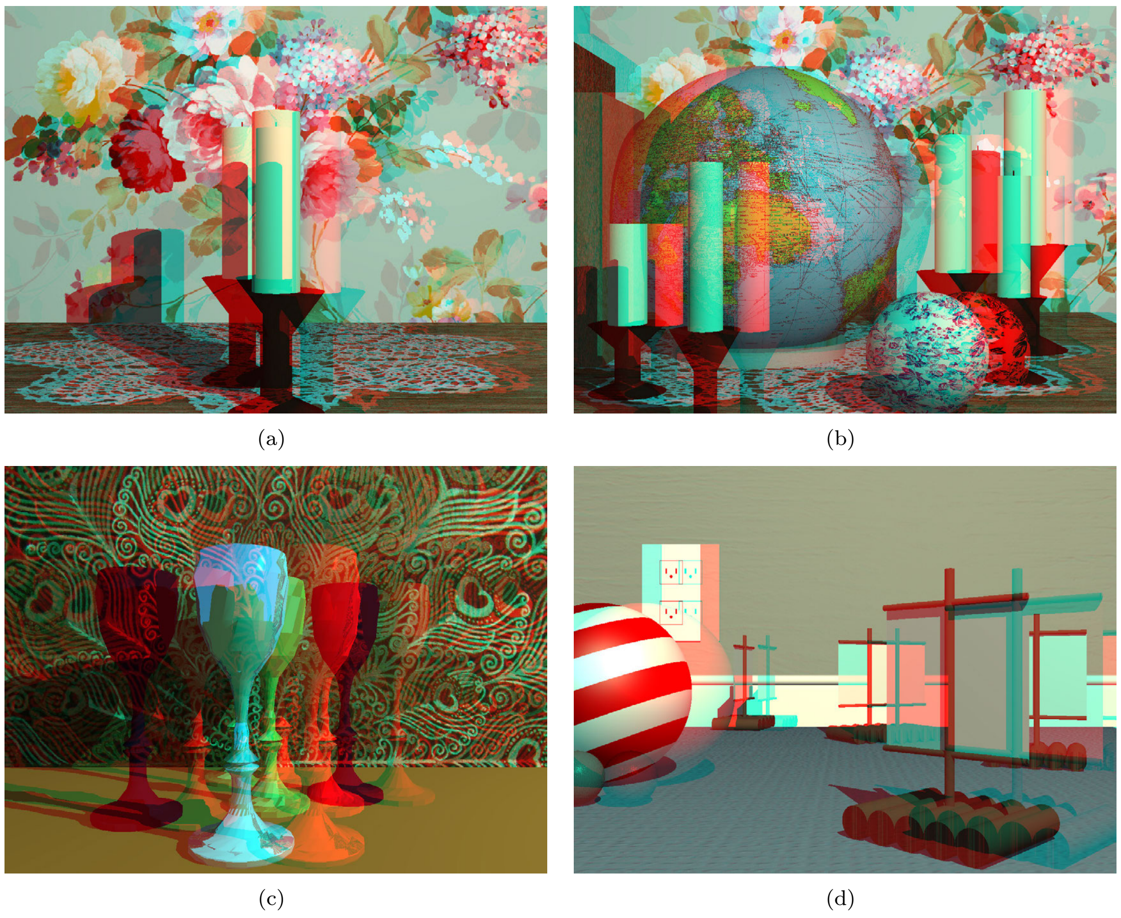

(a) (b) (c) (d)

Figure 3.11: Ground-truth raytraced test images used in our experiments. Red/cyan anaglyph images.

These errors occur because we crop and composite images at integer positions even if the adjusted disparity is not an integer. Errors in holes are expected as this is the region being filled with our “best guess”. Overall, since our adjusted disparity maps are nearly identical to ground truth, except in regions where inpainting took place, we compare only these inpainted regions.

We compared our inpainted disparity maps to those produced by other proposed techniques. First, we compared our results to the lowest-scanline approaches of Furihata et al., Manap et al., and Pan et al. Then, we compared our results against PatchMatch applied directly to the disparity map. Finally, we compared our results to the S3D inpainting approach of Wang et al., which restricts the PatchMatch algorithm to choosing patches with disparity less than some target. The results are presented in Tables 3.1, 3.2, and 3.3. Note that all disparity maps used in our experiments are 8-bit.

For all measures of error, our iterative method produces a more accurate disparity map than other algorithms on three out of four of the test images. For those measures of error and images where we do not produce the most accurate disparity map, our method had the second-lowest error, and that error differs by less than 1.

We also compared our disparity adjustment method to that of Lang et al. [50], which does not use an inpainting strategy to fill holes, but warps the image instead. This comparison is illustrated in Figure 3.12. Note how our approach preserves object shape, while Lang et al.’s approach

Image	lowest scanline	Patch Match	S3D PatchMatch	Our Method, i iterations $i = 1$	$i = 4$	$i = 8$
a	2.33333	2.99145	15.2194	2.01993	1.98291	**1.96581**
b	**2.30414**	11.7236	4.88262	3.36719	2.95991	2.88789
c	1.8974	6.33242	1.23871	1.16963	**1.08618**	1.09166
d	1.07788	4.50451	2.79063	0.892212	**0.804176**	0.808691

Table 3.1: Average absolute error (AAE) between ground truth and synthetic disparity maps in inpainted region. Our method has the lowest AAE for three out of four test images.

Image	lowest scanline	Patch Match	S3D PatchMatch	Our Method, i iterations $i = 1$	$i = 4$	$i = 8$
a	4.9045	**4.19673**	15.7843	4.95421	4.63051	4.60521
b	**5.84128**	16.0786	7.75662	7.58208	6.81753	6.79496
c	4.0653	7.66171	3.74376	3.84551	**3.37286**	3.37773
d	5.28136	10.3468	7.69036	5.03268	**4.96551**	4.97755

Table 3.2: RMSE between ground truth and synthetic disparity maps in inpainted regions. Our method has the lowest RMSE for three out of four test images.

Image	lowest scanline	Patch Match	S3D PatchMatch	Our Method, i iterations $i = 1$	$i = 4$	$i = 8$
a	7.44113	7.73364	31.5685	4.95421	4.63051	**4.60521**
b	9.73508	32.1356	14.6165	8.0241	6.83174	**6.80148**
c	6.61751	15.2096	**3.74376**	4.14594	3.77315	3.7997
d	7.42255	19.1259	12.9376	**5.07839**	5.19268	5.20922

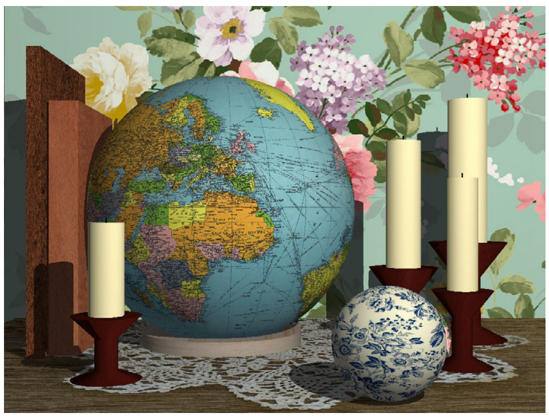
Table 3.3: Penalized RMSE (PRMSE) between ground truth and synthetic disparity maps in inpainted region. Our method has the lowest PRMSE for three out of four test images.

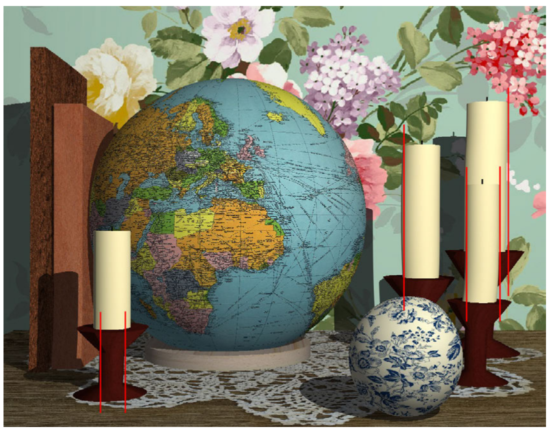
struggles with the large adjustment and introduces significant distortions to the candlesticks. This distortion would not be visible for S3D media requiring only small adjustments to disparity, but media requiring a large adjustment, such as the application discussed in Section 3.5, would have large distortions where objects are stretched to fill holes.

Finally, our disparity adjustment method can be used for both uniform and non-uniform changes, as demonstrated in Figure 3.13. Note that the non-uniform adjustment in Figure 3.13 is non-realistic to demonstrate the potential for artistic expression.

We did not assess the viewing comfort of our adjusted S3D images, although methods to predict viewing comfort exist in literature. Richardt et al.'s model computes a coherence score based on the correlation between computed left and right disparity maps [70]. While their method was shown to predict the comfort of images exhibiting binocular rivalry, randomness, and the shower door effect, it does not consider the discomfort produced by large disparities. Additionally, as our method does not introduce inconsistencies in the image, but removes them, we did not feel this metric would provide a useful assessment of our adjustments.

Jung et al.'s method computes the visually important regions of the image using both the image and disparity map [41]. They then predict viewing comfort by observing the saliency-weighted disparity magnitude and gradient. While this method was able to predict the viewing comfort of images with large or varying disparities, we chose not to assess our adjustments with this method. Our focus is purely on providing a method for consistently adjusting disparity and filling the holes created by those adjustments. We leave the discussion of "what is comfortable" to the user, noting first that the comfort of an S3D image is related to how and where it is viewed,


(a) our method


(b) Lang's method [49]

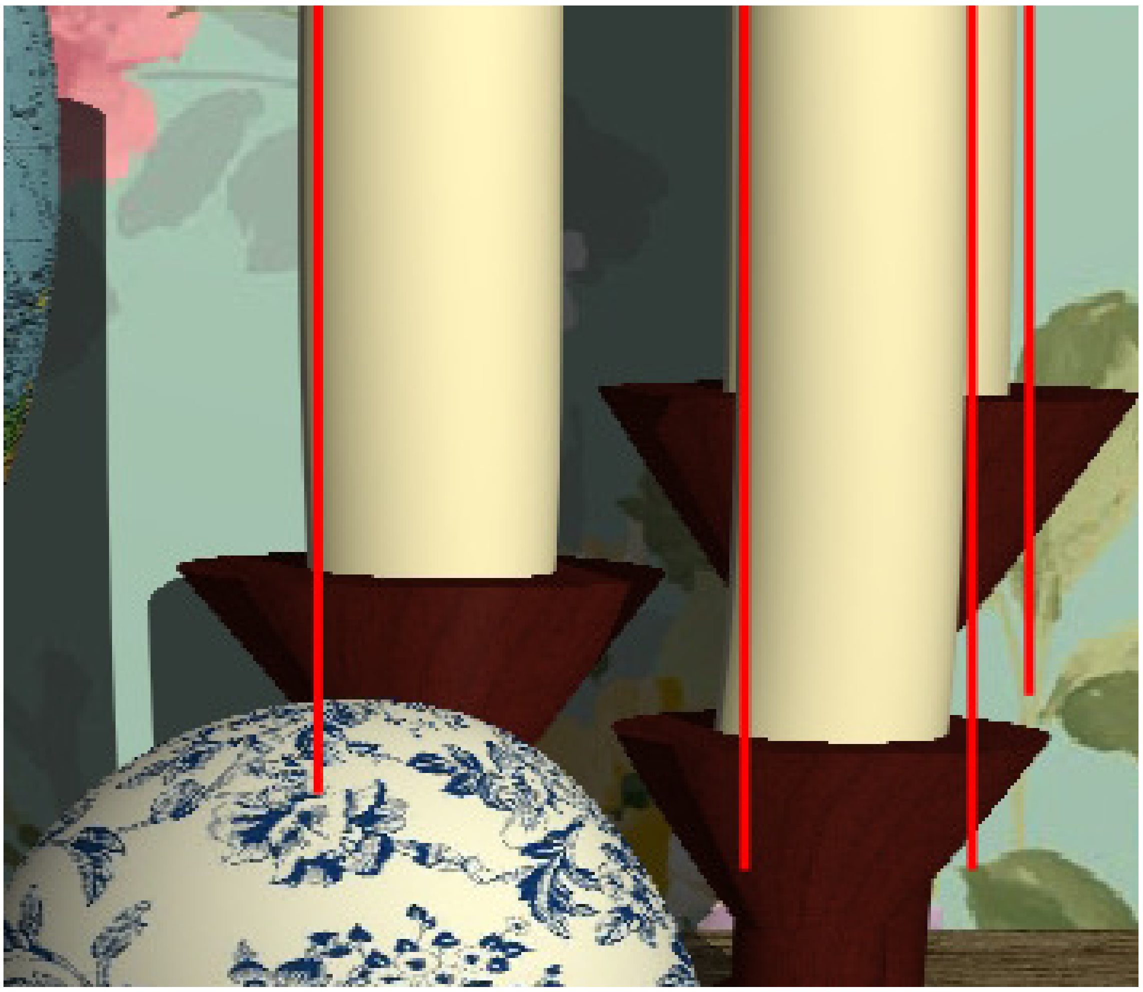
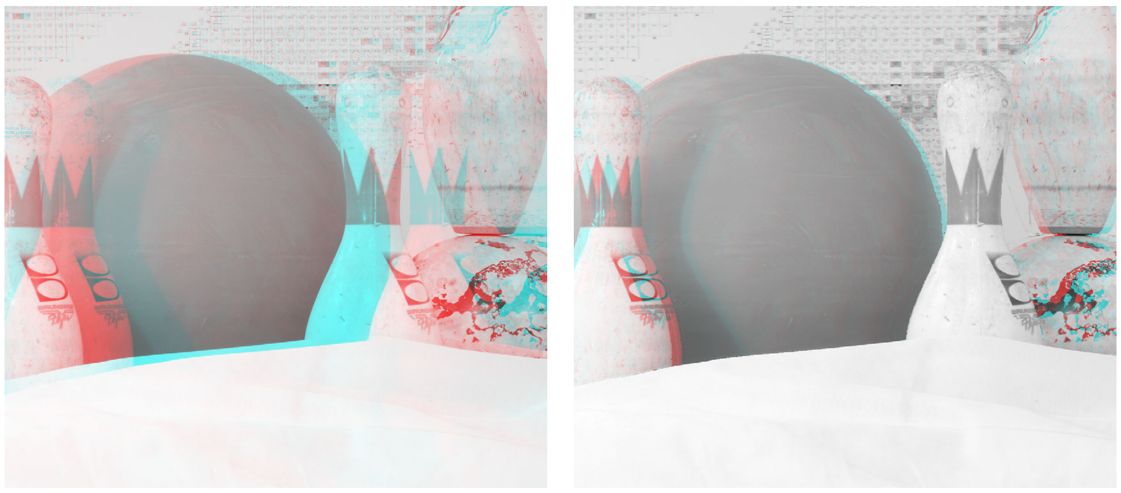

(c) close-up of warping produced by Lang's method

Figure 3.12: Reducing the disparity by 50%. Note the warping of the candlesticks in Lang's results. Red vertical lines provided for comparison.

and second, that users may purposefully wish to make or leave regions of images uncomfortable for artistic purposes.



(a) original image (b) non-uniform reduction

Figure 3.13: Adjusting disparity non-uniformly for artistic effect. The foreground of (b) is flattened and moved behind the background objects, whose disparity was reduced by 80%. Image courtesy of Scharstein and Pal [73].

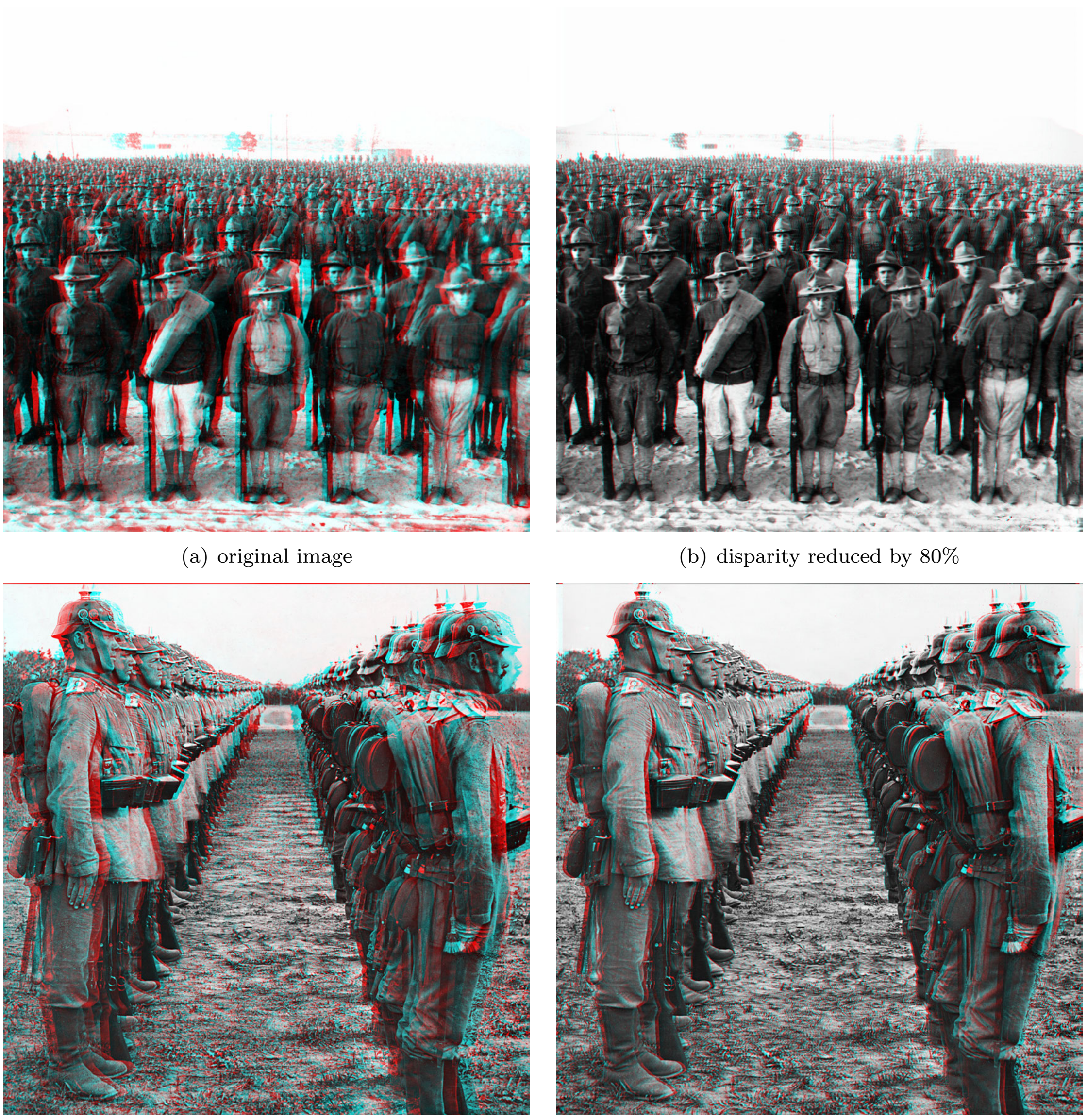
3.5 Adjusting 3D Photos

We used our method to adjust the disparities of several S3D photos for the 2014 documentary “Soldiers’ Stories”. The film tells the story of World War I from the perspective of the soldiers through a sequence of S3D images captured by both allied and central forces. As might be expected, due to both camera quality and age, these photos contain a number of flaws that make stereo viewing and manipulation difficult. Specifically, these photos exhibited vertical parallax, rotational parallax, different exposures and focus, scratches, and various forms of decay and deterioration. However, the biggest flaw in many of these photos is that the horizontal parallax exceeds the comfortable viewing range, particularly on a large IMAX screen, where the film was eventually going to be screened.

In order to adjust the disparities of these S3D photos (and reduce the horizontal parallax), the disparity maps are needed. However, the flaws present in the photos pose a problem for the stereo matching algorithms used to produce disparity maps. To account for these flaws, we used the following process to produce the disparity maps and adjust the disparity:

1. rectify images
2. remove scratches, decay and equalize images in Photoshop
3. generate disparity maps with Semi-Global Block Matching [35]
4. analyze and repair disparity map edges and segments using the method described in Section 2.4.2 and in our disparity correction patent [2]
5. correct outstanding disparity errors manually in Photoshop (typically large patches of sky or solid coloured regions)
6. adjust disparity and fill holes

For those images used in “Soldiers’ Stories”, disparity was successfully reduced by 60%–80%. Figures 3.1 and 3.14 presents three of these adjusted images.


(a) original image

(b) disparity reduced by 80%

(c) original image

(d) disparity reduced by 60%

Figure 3.14: Adjusted images from “Soldiers’ Stories” (Jonathan Kitzen, [45]).

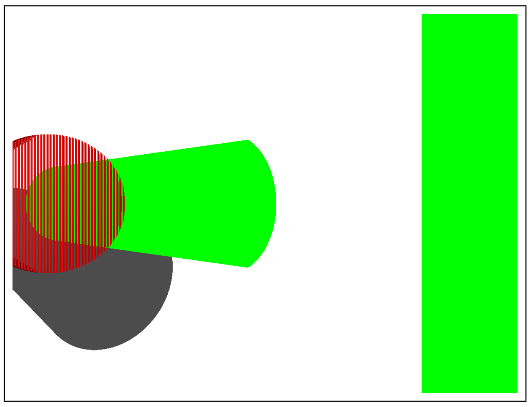
3.6 Limitations

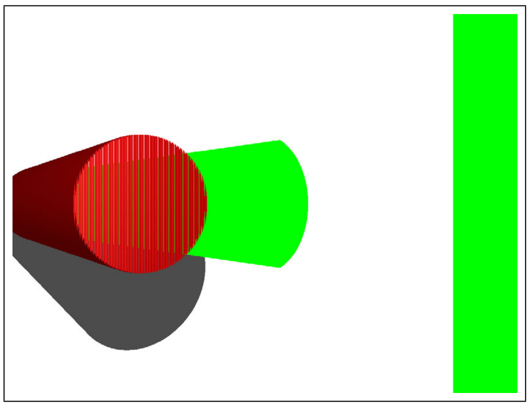
Our method has several limitations. First, the quality of the adjusted image is directly linked to the quality of the disparity map. Since it is difficult to acquire accurate disparity maps for S3D photos, this is the biggest limitation of our method. Second, our method currently supports integer disparities, which quantizes the disparity and increases planarization during adjustment. In the future we hope to expand our method to real-valued disparity maps for smoother adjustment. Finally, while our method can be used to expand disparities, doing so will produce a visible, discrete stepping on previously smooth surfaces. This stepping occurs because adjustment methods, ours and stereo warping, do not increase the number of disparity layers, they only transform or reduce them. Therefore increasing disparity by a factor of k maps a disparity d to kd, and $d+1$ to $kd+k$, but no layers between kd and $kd+k$ will be produced, creating a stepped appearance. Figure 3.15 illustrates the effects of increasing the disparity by large factors. Note that green regions are those regions that require filling. Each “ring” represents pixels with the same disparity value; when disparity is multiplied, pixels in the ring remain together, but the separation between the rings increases, creating the stepped appearance. Also note that our filling method cannot be applied here because the holes are opened in the middle of foreground objects—not from exposing the region behind them.

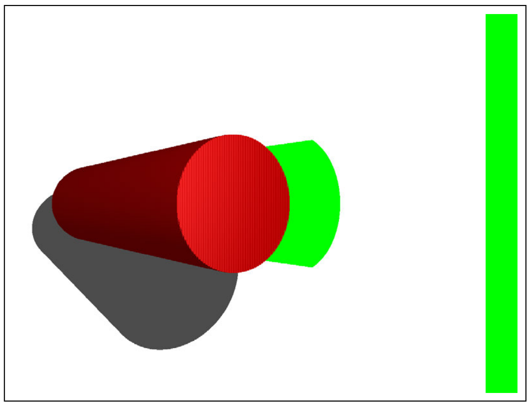
3.7 Conclusions

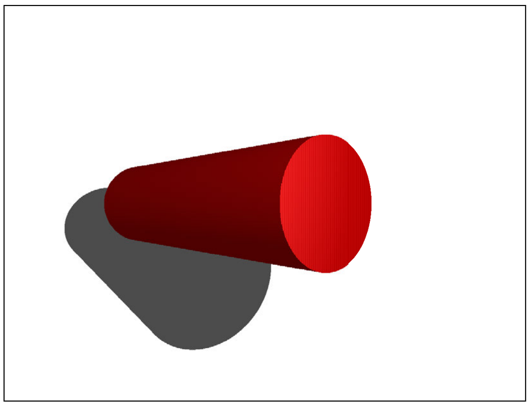
In this chapter we presented a method to adjust the disparity of stereoscopic 3D (S3D) images non-uniformly. We provided a disparity-aware method for filling holes arising from previously occluded regions becoming visible. We compared our results to several other disparity adjustment approaches and found our method produces more accurate results and preserves the integrity of scene object shapes (i.e., does not distort objects). Finally, we applied our method successfully to several hundred-year-old S3D photos for the WWI documentary “Soldiers’ Stories”.

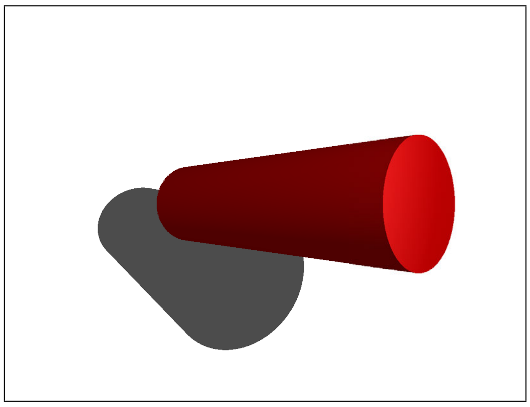
We note however that our method for inpainting the holes created by adjusting disparity could be improved. Notably we use a hard-coded rule-based algorithm to choose either interpolation or PatchMatch as the inpainter. The modern frontier of computer graphics research naturally suggests a generalization of this approach to one that uses machine learning to classify holes and choose an inpainting algorithm. In the next chapter, we explore this idea.

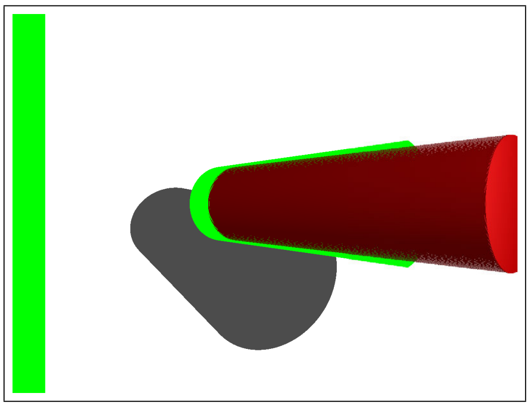

(a) new left image, $d' = 4d$

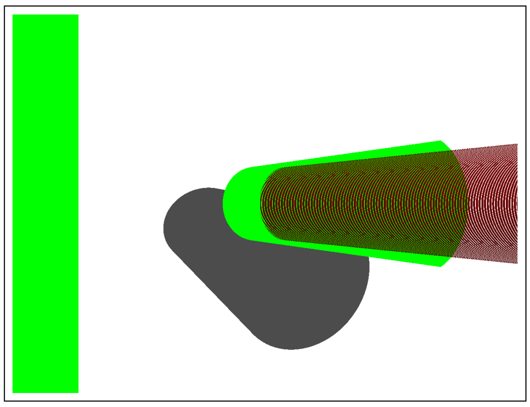

(b) new left image, $d' = 3d$


(c) new left image, $d' = 2d$


(d) original left image


(e) original right image


(f) new right image, $d' = 2d$


(g) new right image, $d' = 3d$

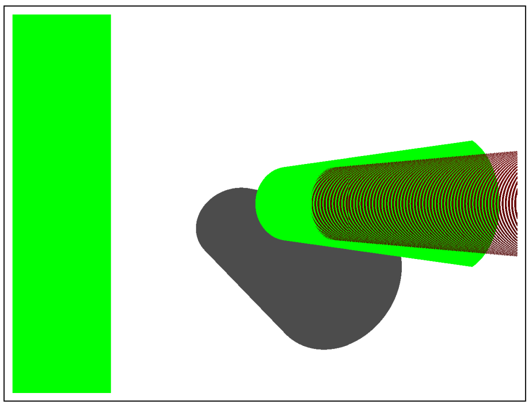

(h) new right image, $d' = 4d$

Figure 3.15: The effect of increasing the disparity by large factors. Note that green regions are those that require filling.

Chapter 4

EnsembleFill: Automatic Inpainter Selection

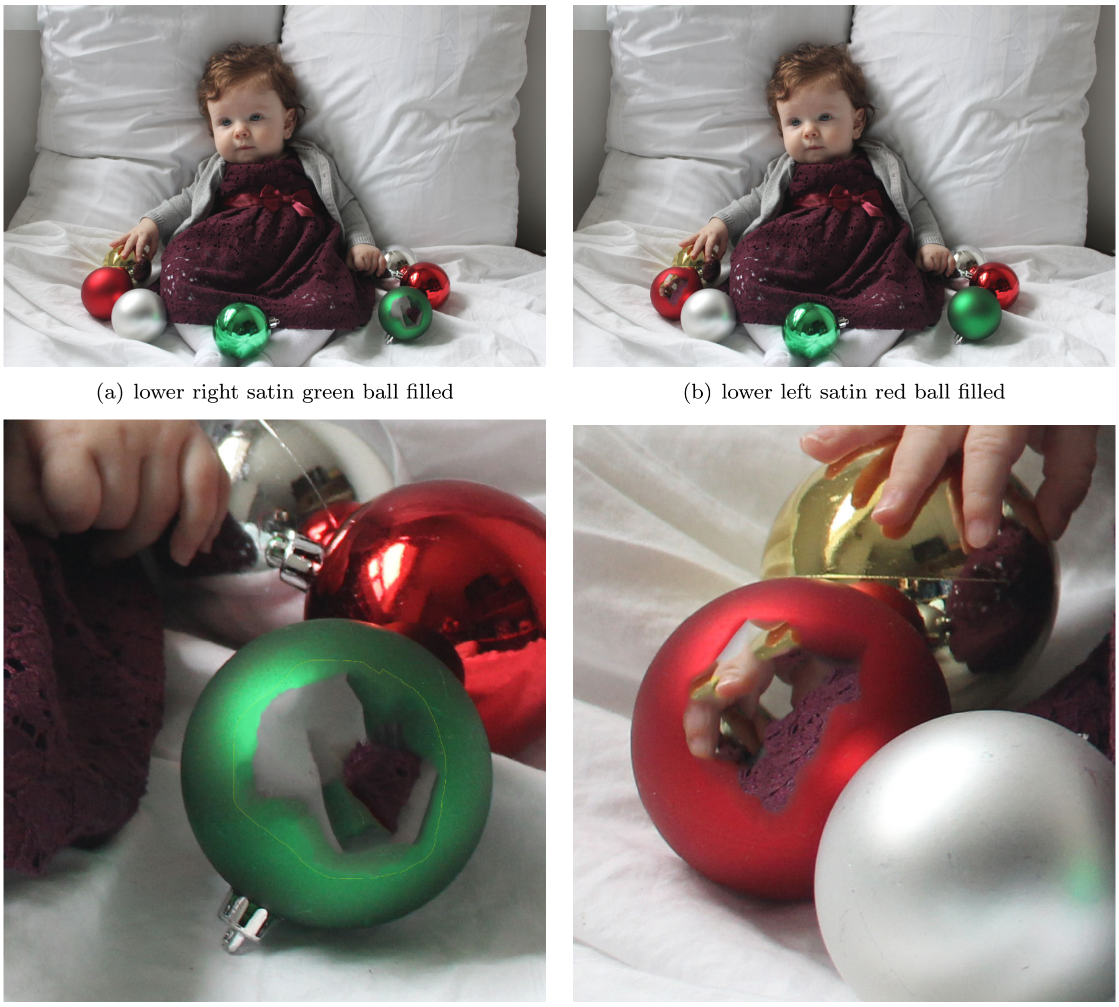
In the previous chapter we discussed a rule-based approach to selecting an inpainting algorithm for filling holes in layers of the stereo image. If the hole is only a few pixels in size, fill with nearest-neighbour interpolation. Otherwise, fill with PatchMatch, an inpainting algorithm that uses texture generation to fill empty regions [6]. For a small hole, nearest-neighbour interpolation is faster than PatchMatch, which does not provide a noticeable improvement in fill quality. However, PatchMatch does provide an improvement in fill quality for larger holes. While this approach worked well for the small holes produced by disparity adjustment, it could be generalized to fill holes from a variety of sources.

One source of holes requiring fill is stereo conversion. Stereo conversion takes 2D media and constructs an artificial second view to produce an S3D image. The conversion process begins with artists rotoscoping (tracing) the contents of each frame then segmenting each frame into objects before assigning each object a depth. To produce the artificial second view, objects in the scene are shifted horizontally according to their depths. This shift, much like with disparity adjustment, exposes the region behind the shifted object. However, since these regions were occluded in the input frame, there is no image data to fill holes. The conversion process is just one of many contexts in which an image may require inpainting.

Another common source of holes in images is the removal of undesirable artifacts. For example, it is common to remove signposts and power lines from landscape photos. Or, in older photos that have been scanned, coffee stains, cracks, tears and holes, such as the deterioration seen in Figure 2.19, may be removed. When these artifacts are removed, the hole left behind must be filled.

Holes can be filled in a variety of ways, from hand painting to automatic inpainting algorithms. The simplest inpainting algorithm is interpolation, which blends or averages the colours of nearby pixels to produce a fill. More complex inpainting algorithms such as structural inpainting attempt to preserve large, structural elements such as tree trunks or the sky [11]. Textural inpainting algorithms aim to preserve the fine textures and patterns, i.e., random noise, grass or brick, of the surrounding region [11]. Given a hole in an image, which of these methods will produce the highest quality, most convincing fill? One solution is to try every method, look at each result and choose the best one, but this would be costly.

How do artists or home users choose an inpainting algorithm when presented with multiple options? We observed professional S3D conversion artists filling holes created by exposing previously occluded regions. The artists were presented with a variety of fully-parameterizable inpainting algorithms. Their goal was to select the algorithm and parameters that produced the best fill for each hole. However, artists had thousands of holes to fill in a very short period of time because this process was repeated for every frame of the film. We noted that most often an algorithm was randomly selected and the parameters were ignored. In many cases the fill chosen was not the best fill that could be chosen. Poor quality fills then need to be hand-painted by another artist for the final version—a very costly task. The cost could be reduced if the initially selected inpainting algorithm was selected more carefully and more intelligently.


(a) lower right satin green ball filled

(b) lower left satin red ball filled

(c) close-up of the satin green ball's fill

(d) close-up of the satin red ball's fill

Figure 4.1: Unexpected results from content-aware fill.

When inpainting holes created by removing undesirable artifacts from photographs, how does one choose an inpainting algorithm? In Adobe Photoshop, the most commonly selected method for all holes is "Content Aware Fill" — in fact, it is the default. While "Content Aware Fill" fills most holes reasonably well, occasionally the algorithm produces unexpected and undesirable results as demonstrated in Figure 4.1. Note how the satin green ball in the bottom-right is filled with unexpected colours (a); similarily, the lower-left red ball is also filled with unexpected colours (b).

Since it is too costly to try every possible fill, but applying the same fill to every hole may produce undesirable results, is there a better way to choose an inpainting algorithm that will produce the best fill for a hole? Consider the hole itself; holes that are few pixels in size in a large image are barely noticeable. Therefore, it is likely they could be filled with any algorithm without affecting visual quality, in which case we might simply choose the fastest one. However, large holes—relative to image size—cover large, varying regions of the image and are very noticeable. Greater care must be taken in choosing an inpainting algorithm for such holes. We could measure the hole and select an inpainting algorithm based on that measurement. Or, we could gather information about the hole and the image data in its vicinity and use machine learning to choose the inpainting algorithm. The goal of this chapter is to demonstrate that it is possible to use machine learning to intelligently select the most appropriate inpainting algorithm for a hole.

The rest of this chapter is organized as follows. First, we discuss the strengths and weaknesses of different types of inpainting algorithms. Then we discuss our experiment, starting with data collection and setup. Finally we present our results.

Note that this chapter, unlike the rest of this thesis, does not work with S3D images or disparity maps. This chapter represents a preliminary exploration of a new idea. As such, we wanted to use the simpler 2D image domain to uncover a methodology that we could apply to S3D media later.

4.1 Inpainting Algorithms

The simplest inpainting algorithms use pixels adjacent to the hole to calculate an average fill colour from. However, averaging does not preserve image structures or textures. Therefore these methods tend to produce convincing fills only for small holes, or holes in solid colour regions. The majority of inpainting algorithms aim to restore one or both of these image features and are classified as either structural or textural. This section discusses the strengths and weaknesses of these two approaches.

4.1.1 Structural Inpainting

Structural inpainting algorithms search the vicinity of the hole for strong edges, using them as a fill direction to propagate these structures into the hole.

Bertalmio et al. identify isophotes and use Navier-Stokes based fluid dynamics to propagate colour through the hole along isophote directions [10]. The method produces smooth continuations of structures if the diameter of the hole along the isophote direction is small. However, for large diameter holes extrapolated isophotes may “kink”—be discontinuous within the hole—producing less convincing fills.

Hocket et al. precompute a vector field to guide fill direction from vicinity edges, automatically detected splines and optional user-provided splines [36]. The algorithm eliminates kinking and produces a smooth continuation of image structures.

Regardless of the specific algorithm used to propagate isophote and edge colour through the hole, structural inpainting algorithms do not propagate textures into the hole, as demonstrated in Figure 4.2. Therefore, structural inpainting algorithms generally produce the most convincing fills for very thin holes or holes in regions lacking texture.

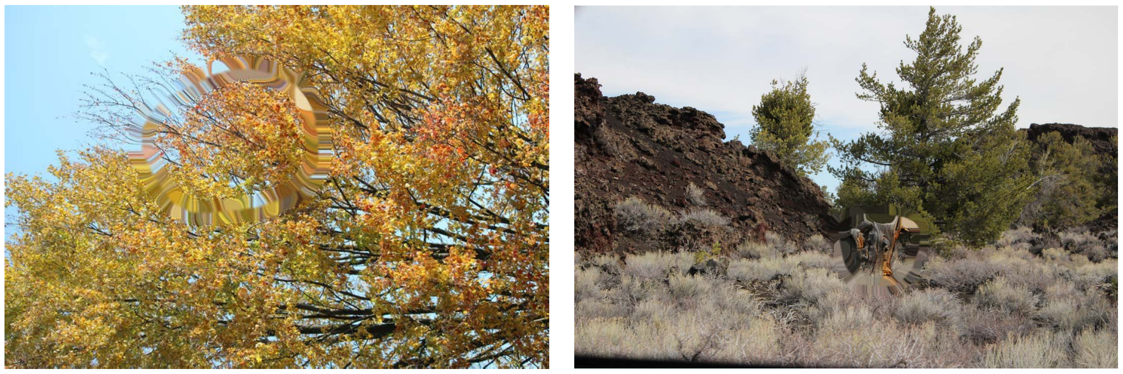

(a) missing leaf texture (b) missing grass, stone and wood textures

Figure 4.2: Inpainting highly textured images with a Navier-Stokes based structural inpainting algorithm [10].

4.1.2 Textural Inpainting

Texture synthesis has a long history in computer graphics. Texture synthesis algorithms take a small sample of a texture, such as brick or grass, and expand it to produce a larger image filled with that same texture [22]. These algorithms have given rise to hole filling algorithms. Instead of expanding a single texture, textural inpainting algorithms search the image space for exemplar textures in order to synthesize new texture within the hole.

Criminisi et al. search along isophotes for small image patches that best match each patch along the edge of the hole (the “front”) [20]. Filling in patches along the front prioritizes regions along isophotes/strong edges to preserve structure while synthesizing believable texture. However, the limited search space can produce inconsistent or repetitive fills and the search is costly.

Barnes et al. use image patch correspondences found by nearest-neighbour search to find appropriate patches and fill the hole [6]. Initial patch correspondences are randomly selected and then improved by nearest-neighbours search. However, instead of evaluating every neighbour patch to find the best match—which is costly—this modified nearest-neighbour search evaluates a small set of randomly selected patches within a radius of the current match. This randomization improves performance and reduces the probability of falling into a local minimum for matches.

However, a single poorly chosen match can produce a cascade of nonsensical fills during hole filling, as each patch is used as a basis for choosing the next one. This can result in a variety of incorrect fills, from obviously incorrect colour, to adding extra facial features, as demonstrated in Figure 4.1. The algorithm presented by Barnes et al. is called PatchMatch, and it is also the basis of Adobe Photoshop’s Content Aware Fill tool. Textural inpainting tends to produce the most convincing fills for small-to-medium sized holes in densely textured regions that exist throughout the image.

4.1.3 Combining Strategies and Alternate Approaches

Another approach to inpainting is to decompose an image into structural and textural components and apply the corresponding inpainting algorithms to each before recomposing the image. This is the approach taken by Hesabi et al. [34]. However, the results can be negatively affected by both structural inpainting issues like kinking and flattening of curved isophotes in a large hole, as well as textural inpainting shortcomings arising from poor patch selection.

Frantc et al. presented a method that uses machine learning to assess the quality of inpainting algorithms [23]. Using a number of saliency and image statistics to model the human visual system as features, a regression system is trained to predict fill quality. This system can be used to fill a hole with all possible inpainting algorithms and choose, afterwards, which has the highest quality. However, it is costly to apply a number of inpainting algorithms to the hole and then choose the most convincing.

Convolutional neural networks (CNNs) have also been used for inpainting, superresolution and denoising [16, 48, 83]. However, for large holes that cover a number of salient features or fine textures these methods produce blurry fills or artifacts similar to PatchMatch [6].

4.2 Experiment

Our goal is to demonstrate that through analysis of the hole and its vicinity we can build a classifier to automatically suggest the inpainting algorithm that would produce the most convincing fill. The experiment has the following steps:

1. Collect a set of images and holes to use for training and testing.
2. Fill each image-hole pair with a selection of inpainting algorithms.
3. Conduct a user survey to determine which inpainting algorithm produced the most convincing fill for each image-hole pair.
4. Choose and extract features from the hole, vicinity and image.
5. Build and tune a classifier with minimum accuracy of 75%.
6. Evaluate predictions.

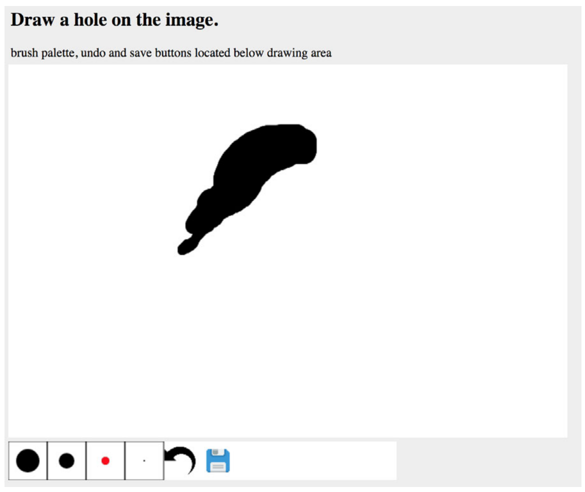
4.2.1 Data Collection

We collected a set of 2D photographs from amateur photographers, enthusiasts, and family members so that our dataset had a wide range of image qualities and subjects. The photos were captured with mid-range DSLR, point-and-shoot, and mobile phone cameras, yielding a variety of resolutions and aspect ratios. In total, 1500 images were collected.

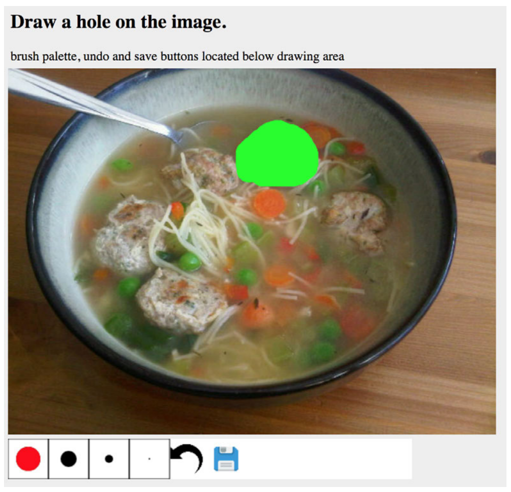
To produce a set of holes for our collected images we conducted a brief user experiment. The experiment was conducted using a simple web application and was open to everyone regardless of age, gender, or experience. Additionally, we permitted participants to repeat the experiment. No information was collected regarding the actual number of unique participants or the number of times each participant repeated the experiment. This may introduce a small bias in the collected data.

The experiment asked participants to perform one of two tasks. The first task asked participants to draw a "blind" hole on a rectangular white canvas of a random size. A blind hole is a hole drawn without knowing what image it would be applied to. The second task asked participants to draw a hole on a specific, randomly selected image. Figure 4.3 illustrates the two tasks from the perspective of the participant. No information was collected apart from the participant's drawing.

Blind holes produced from the first task were randomly assigned to images of the same resolution after collection. Holes produced from the second task are already paired with an image. Although


Draw a hole on the image.
brush palette, undo and save buttons located below drawing area

(a) Task 1: blind hole drawing


Draw a hole on the image.
brush palette, undo and save buttons located below drawing area

(b) Task 2: drawing hole in an image

Figure 4.3: Data collection tasks to produce image holes. The hole is drawn in black for Task 1, and in green for Task 2 to ensure maximum visibility for participants.

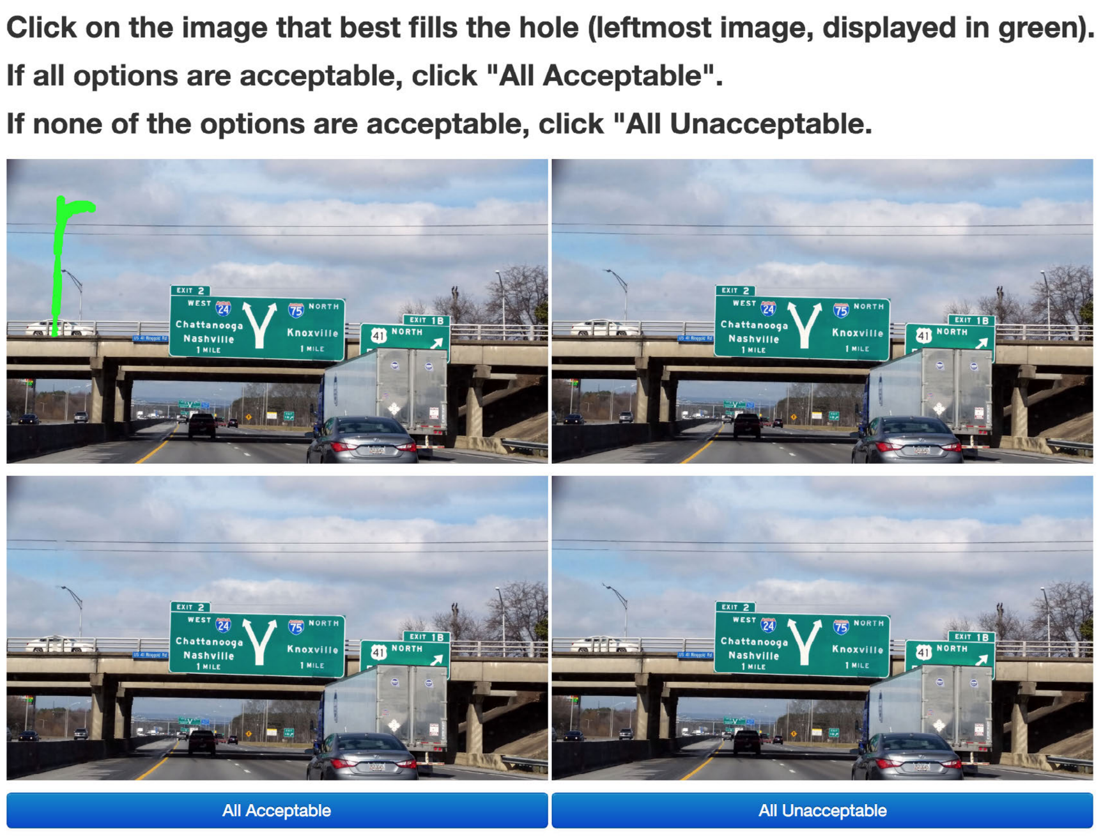
there were 1500 source images, images were selected randomly for participants resulting in just over 1200 hole-image pairs. However, we found a number, around 100, of offensive or obscene drawings in the collected data. While these were valid holes, we removed them because these pairs would be used in a second user experiment at a later date. The final dataset contained 1097 pairs.

We filled each hole-image pair using basic interpolation (herein referred to as Basic), the fluid-based method from Bertalmio et al. for structural inpainting [10], and the patch based method from Barnes et al. for textural inpainting [6].

Once hole-image pairs were filled with each of the inpainting algorithms we conducted a second user experiment to determine which method produced the most convincing fill. Similar to the previous experiment a web application was used to collect data and could be completed by anyone. The experiment provided participants with the four different versions of an image. The first image depicts the hole in bright green for reference purposes. The other three images correspond to each of the three fill methods. Participants were then asked to select which of the three filled images had the most convincing fill. If none of the fills were convincing, participants could select “All Unacceptable”. If all of the fills were equally convincing, participants could select “All Acceptable". Participants could choose to assess ten, twenty or forty images but could repeat the experiment. We randomized the ordering of the filled images in order to encourage participants to answer honestly instead of clicking the same screen position each time. A sample from our second experiment is given in Figure 4.4.

Data Composition

Participants classified the collected data into five categories as shown in Table 4.1. On average, seven votes for each hole-image pair were received and the final category was decided by popular vote. This table also shows the breakdown of our training (822 images) and test sets (275 images) (randomly partitioned as 75% and 25% of the raw data correspondingly). Note that our collected dataset is heavily unbalanced. More than 50% of the samples are classified as Textural, while less than 5% are classified as Basic.



Click on the image that best fills the hole (leftmost image, displayed in green).
If all options are acceptable, click "All Acceptable".
If none of the options are acceptable, click "All Unacceptable.

Figure 4.4: Example image from our data collection experiment to select the best fill for each image.

Table 4.1: Breakdown of collected data classes.

Set	Basic	Structural	Textural	All Acceptable	All Unacceptable
raw	35	68	609	252	133
training	29	53	430	199	111
test	6	15	179	53	22

Data Comparison

Fills produced by inpainting algorithms are informed guesses. Our second experiment collected participant responses regarding which inpainting algorithm would produce the best fill for a given hole-image pair. However, requiring participants to select the ground truth for training and testing is costly. Could we use a simple metric to select the correct fill for an image so that we can produce more input data? How do participant responses compare to mathematical evaluation of fill quality?

To calculate how accurate fills are, we computed the average RMSE error for each hole between ground truth and each fill. The most accurate inpainting algorithm was then selected according to which produced the lowest RMSE score. If the RMSE scores for all algorithms were within a small threshold of each other then the sample was labeled All Acceptable. The classification breakdown, of all samples, using this metric is shown in Table 4.2.

Participant classifications matched only 30.90% of RMSE classifications. However, note that

Table 4.2: Breakdown of collected data classes according to RMSE.

Basic	Structural	Textural	All Acceptable	All Unacceptable
23.79% (261)	27.35% (300)	33.45% (367)	15.40% (169)	N/A

the RMSE score only provides insight into the per-pixel accuracy of an inpainting algorithm. RMSE does not measure higher-level structural or textural accuracy, which explains the difference between participant classification and RMSEs. For this reason, RMSE should not be used as a metric to assess the accuracy of a given inpainting algorithm for the purposes of producing new training or test inputs.

The Structural Similarity Index (SSIM) is popularly used for assessing the quality of a noisy or distorted image. SSIM evaluates the similarity of structural elements, not just per-pixel differences [86]. We measured the accuracy of each fill using SSIM for all hole-image pairs, selecting the fill with the highest index as the winner. Table 4.3 shows the breakdown of samples as classified by SSIM. Note that participant responses matched only 19.42% of SSIM classifications. This also indicates that SSIM should not be used as a metric to assess the accuracy of a given inpainting algorithm. Hence, we are unable to produce additional ground truth results to use as training or testing data by using RMSE or SSIM.

Table 4.3: Breakdown of collected data classes according to SSIM.

Basic	Structural	Textural	All Acceptable	All Unacceptable
31.81% (349)	21.70% (238)	46.49% (510)	0.0% (0)	N/A

4.2.2 Features

We observed that basic interpolation produces a convincing fill when the region near the hole is a solid colour or simple gradient or when the hole is unnoticeable without significant inspection. Structural inpainting produces convincing fills for simple images and holes with small diameters, while textural inpainting produces high quality fills when the image contains high-frequency noise, patterns and textures. Using these observations as a guide, the following features were extracted from the hole and nearby pixels (the *vicinity*):

1. Average width and height of hole.
2. Area of hole.
3. Density of SIFT and SURF keypoints in vicinity. SIFT, or the Scale-Invariant Feature Transform, attempts to identify salient features of an image such as sharp corners and high contrast regions [54]. Features are represented by keypoints with a feature scale and direction. SURF, or Speed Up Robust Features, also attempts to identify the features of an image. SURF has faster runtime than SIFT, but it finds fewer features [8].
4. Average SIFT keypoint scale and direction within vicinity.
5. Variance in luminosity within vicinity.

6. Number of unique colours within vicinity.

7. Contrast ratio within vicinity.

We analyzed several features according to the participant's selected class to search for interesting data trends. Figure 4.5 plots hole area, Figure 4.6 plots average hole width and Figure 4.7 plots average hole height by class. Note that the All Unacceptable class has on average the largest hole area, average width and height, while All Acceptable consistently has the smallest. Figure 4.8 plots the variance in luminosity in the hole vicinity by class, and Figure 4.9 plots the number of unique colours in the vicinity of the hole. Figure 4.2.2 plots the contrast in the vicinity of the hole. Figure 4.10 shows the density of SIFT keypoints in the hole vicinity, Figure 4.11 plots the average scale of SIFT keypoints in the hole vicinity, and Figure 4.12 plots the average direction of SIFT keypoints in the hole vicinity. Note that for all box-whisker plots the whiskers represent the 25th to 75th percentiles.

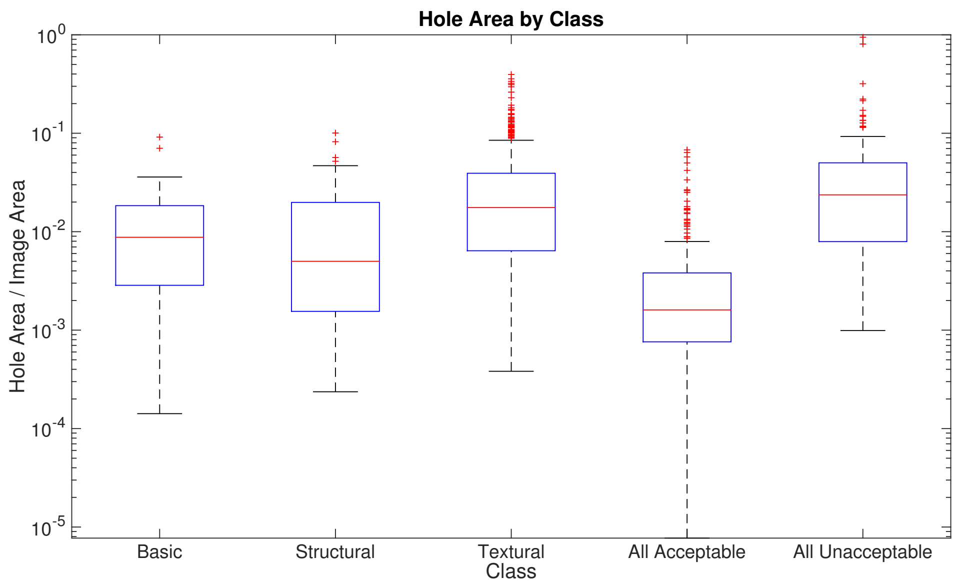


Figure 4.5: Average area of hole by class.

4.2.3 Training

We constructed several classifiers with scikit-learn, a popular Python library for machine learning [65]. In particular, as we have a small dataset, we were able to experiment with k-NN, SVM, and Adaboost classifiers with cross validation for hyper-parameter tuning [27, 28]. 75% (822 images) of the sample set was randomly selected and reserved for training and the remaining 25% (275 images) for testing.

In addition to each learner's standard hyper-parameters, we added a new hyper-parameter: the radius of the hole vicinity. We used five radii: 0.01%, 0.015%, 0.02%, 0.025% and 0.03% of the image diagonal with a maximum radius of 150 pixels.

Each feature in the feature vector was normalized to the range $[-1, 1]$ prior to training or testing. For each classifier, we tried a variety of combinations of features from the feature vector including

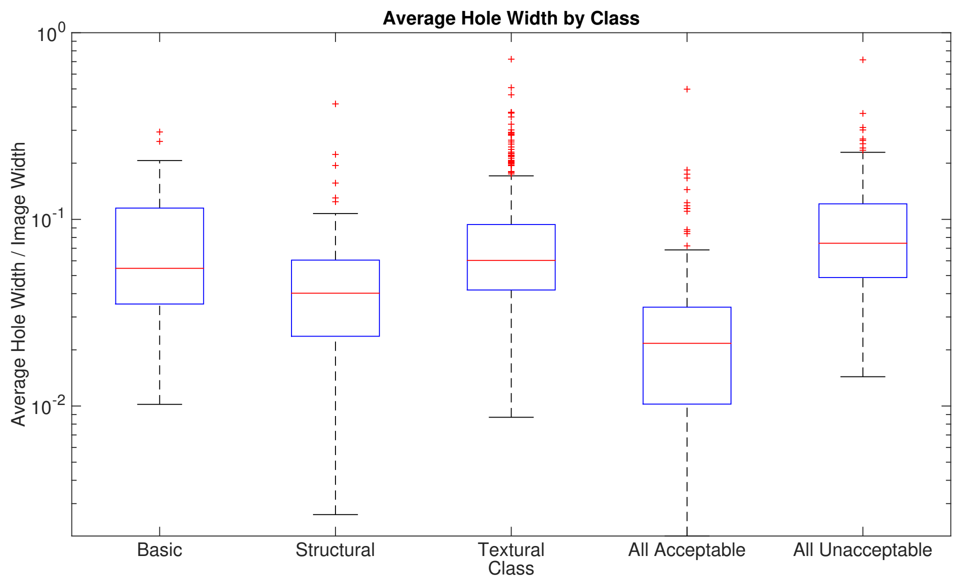


Figure 4.6: Average hole width by class.

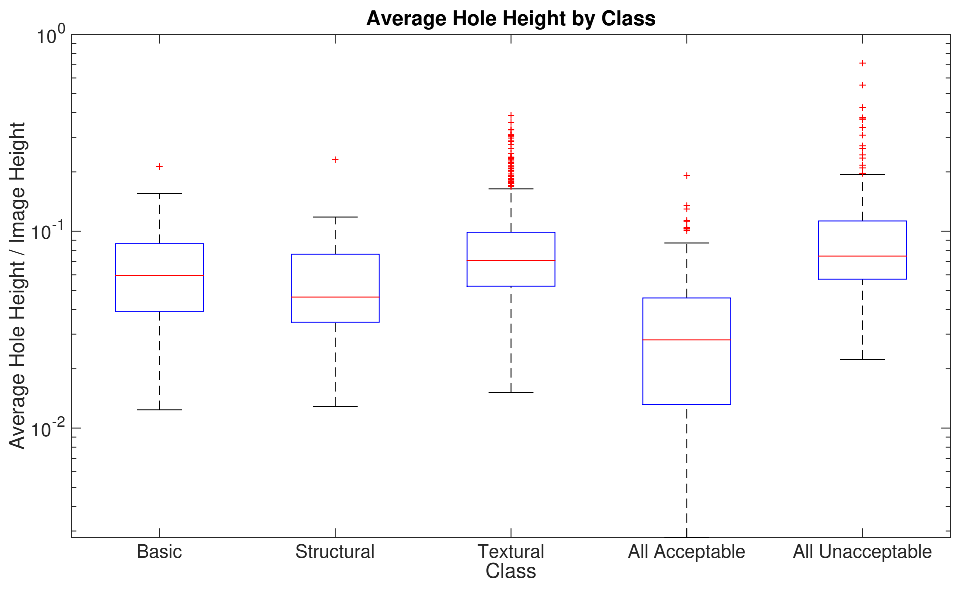


Figure 4.7: Average hole height by class.

applying Principle Component Analysis to perform feature reduction. We also noted that samples in the class All Acceptable were very similar to those in Basic. Because All Acceptable implies that all of the inpainting algorithms produce convincing fills, we chose to combine All Acceptable with Basic, thereby preferring basic interpolation over the more computational intensive PatchMatch.

We measure two types of prediction accuracy. The first, is accuracy, the percentage of samples correctly predicted. However, we found that any sample that was originally labeled All Acceptable could be filled convincingly with any algorithm. Therefore any predicted class, other than All Un-

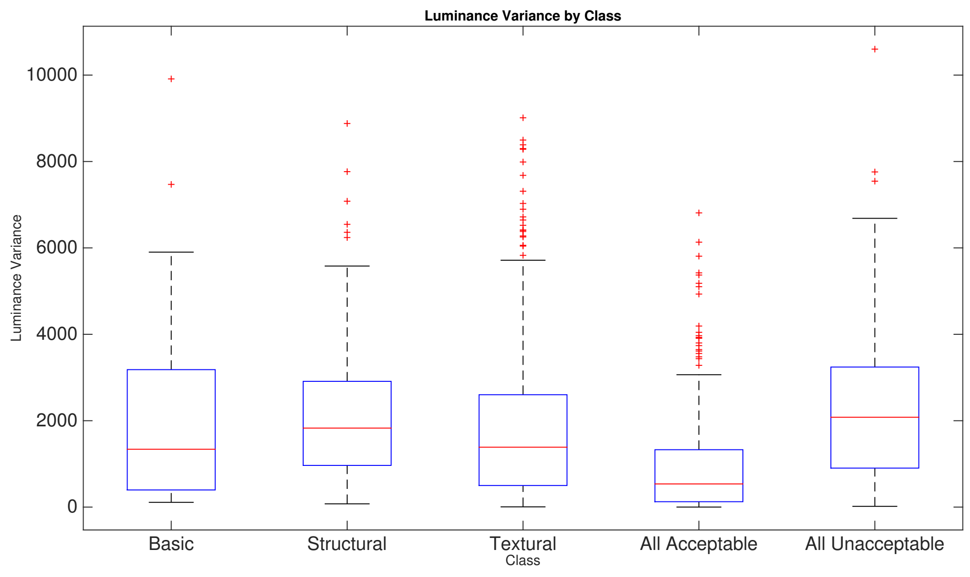


Figure 4.8: Luminosity variance in the hole vicinity.

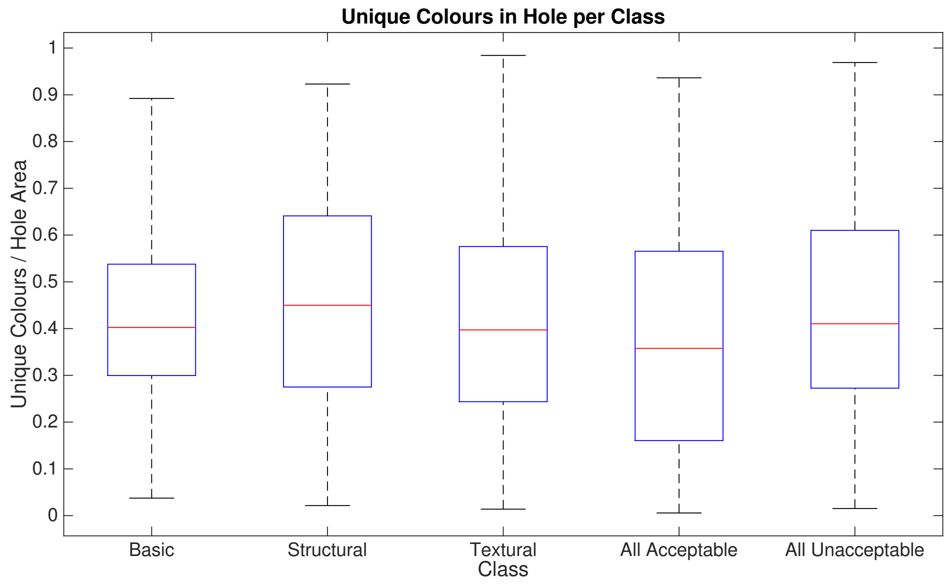


Figure 4.9: Unique colours in the hole vicinity.

in the hole vicinity.

acceptable, would be acceptable fill choice for an All Acceptable sample. Hence, Basic, Structural and Textural should also be considered a correct prediction for any sample in the All Acceptable class. We use "adjusted accuracy" to be the percentage of samples predicted correctly such that any sample originally labeled All Acceptable is considered to be correct if is is predicted to be Basic, Structural or Textural.

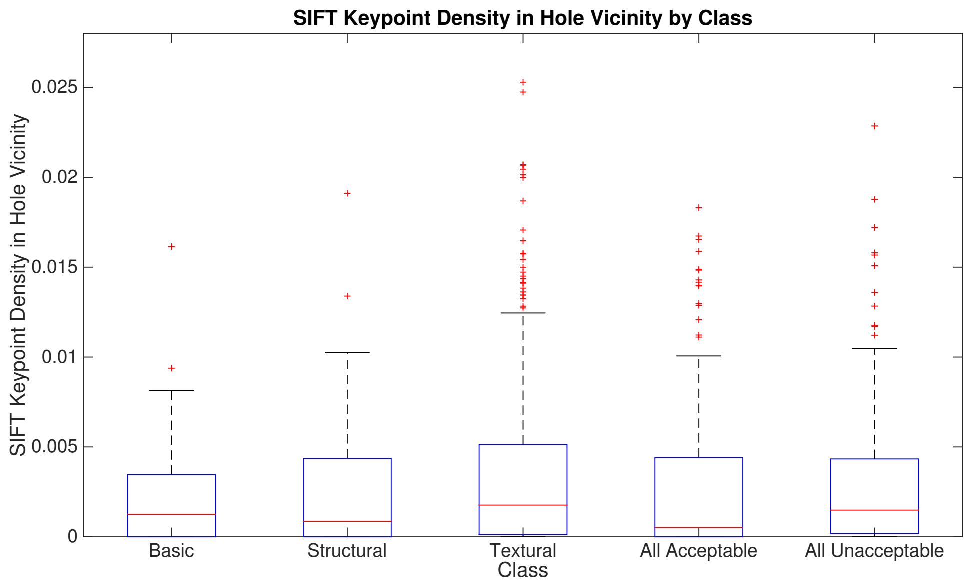


Figure 4.10: Density of SIFT keypoints in the hole vicinity.

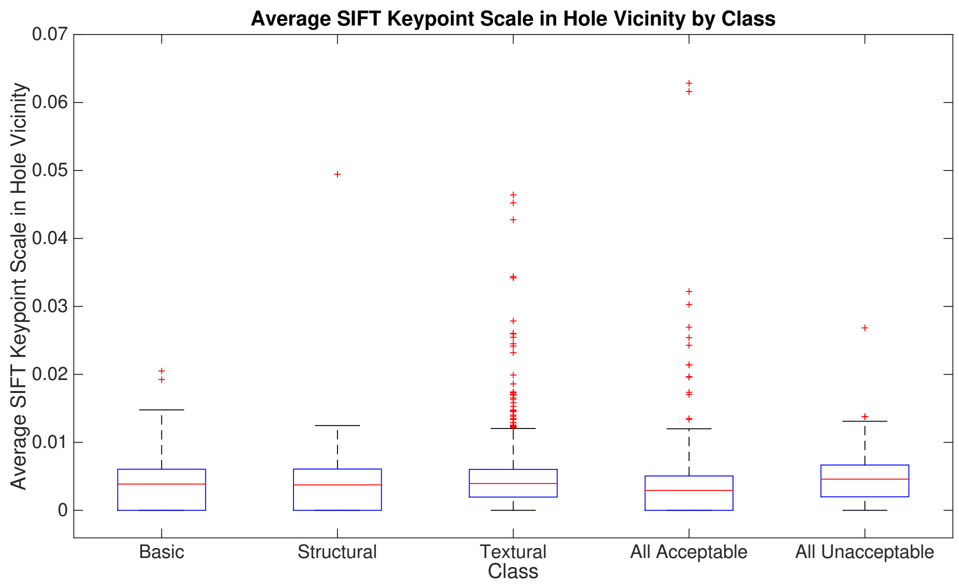


Figure 4.11: Average scale of SIFT keypoints in the hole vicinity.

4.3 Results and Discussion

Our original goal was to demonstrate that a classifier can be constructed that predicts the correct fill for a hole 75% of the time (accuracy). We had no goal for the adjusted accuracy in our proof-of-concept experiment. Note that simply selecting textural inpainting for all samples results in an accuracy of 65.09% and an adjusted accuracy of 84.36%.

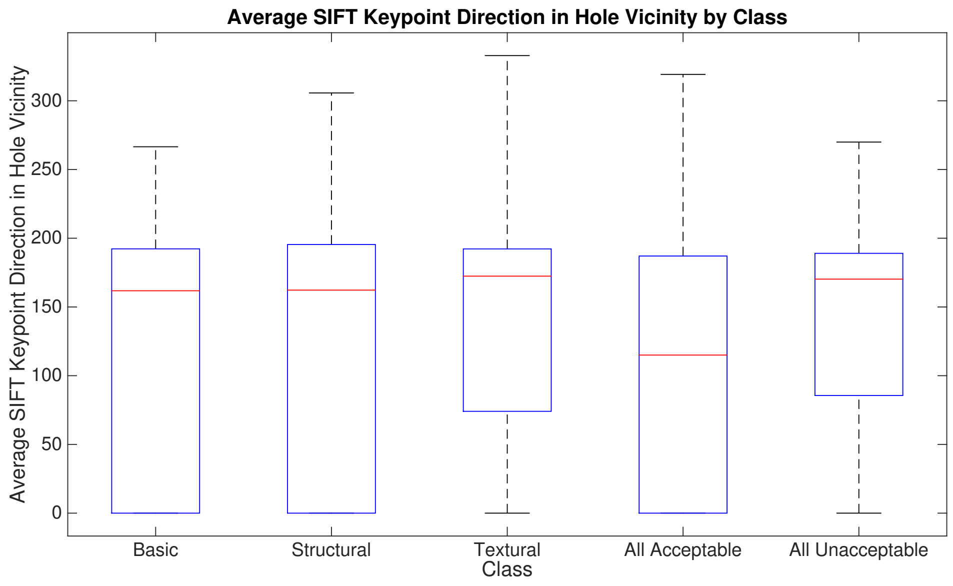


Figure 4.12: Average direction of SIFT keypoints in the hole vicinity.

4.3.1 Adaboost with Decision Trees

Our Adaboost classifiers were constructed with decision trees and cross-validation was used to set the hyperparameters controlling learning rate and number of estimators. Each of the classifiers was trained with a different set of features ranging from all, feature-reduction by PCA, and hand-picked. None of the classifiers produced a sufficient accuracy. Since our dataset contains unbalanced classes, we believe that our dataset may be ill-suited to Adaboost.

4.3.2 k-NN

The accuracy of our k-NN classifier was 75.27% and the adjusted accuracy was 82.54%. This corresponds to 207 and 227 of the 275 test images being classified correctly or acceptably. The best performing k-NN classifier used Manhattan distances, 15 neighbours, and uniform weights [27, 28]. The hole vicinity radius was 0.02%. The feature vector consisted of six features: average hole width, height and area, hole width and height variance, and luminosity variance in the hole vicinity.

No samples from the test set were predicted to be in the Structural class, and only a single sample was placed in the All Unacceptable class. Our k-NN learner yielded a higher accuracy than always filling with Textural, but the adjusted accuracy was 1.82% lower. However, since fewer holes would be inpainted using the more costly Textural method, using this k-NN learner would reduce inpainting costs.

4.3.3 SVM

Overall, we found that our SVM classifiers produced the highest raw accuracy. The best classifier used a hole vicinity radius of 0.015%, and the penalty parameter $C = 19.5$, with a polynomial kernel of degree three [27, 28]. The feature vector consisted of five features: average hole width and height, luminance variance, unique colours, and contrast of the hole vicinity. This classifier had an accuracy of 77.09% and an adjusted accuracy of 81.82%. This corresponds to 212 and 225 of the 275 test images being classified correctly or acceptably. The Textural class was predicted correctly 95% of the time.

We noted that no samples were predicted to be Structural or All Unacceptable but most samples in the All Unacceptable set were predicted to be textural. Again, the accuracy of this learner is higher than the accuracy of applying Textural inpainting to everything. However, the adjusted accuracy is lower. But, since fewer images will be filled using costly Textural methods, this learner would reduce overall inpainting costs.

4.3.4 Discussion

We noted that our classifiers generally did not predict samples to be in the Structural or All Poor categories. In particular, SVM did not predict any samples in either class. Why could this be the case?

Recall the hole size and saliency metrics used for feature vectors. Samples in the Structural category have significant feature similarities to those in the Basic category. Also, samples in the All Unacceptable category have features with values very similar to the Textural category. These similarities along multiple dimensions make it difficult to separate these classes.

Second, our dataset is unbalanced—over 50% of samples fall into the Textural category and the Basic/All Acceptable combined category makes up 26.26% of the dataset. There are few samples in the Structural (6.20%) and All Unacceptable (12.12%) categories. Many classifiers do not handle unbalanced data well, treating classes of few data points as outliers, or placing heavy bias on the largest class.

In order to improve our prediction accuracy, we need to find new features to separate the classes. It would also help to find more samples in small classes, such as All Unacceptable, to balance the dataset.

We also note that our participants were gathered from a wide pool of willing individuals across the internet. They do not represent professional image editors or anyone experienced in image repair. Therefore, it is possible that some of our data is poorly classified into categories like All Acceptable due to participant inexperience. Were we to repeat these experiments with professionals, it is possible that a different and more balanced breakdown of classes would be seen.

Reducing Imbalance

Unbalanced data can be addressed in several ways. In SVMs, unbalanced data can be addressed by adding a "class weight" parameter. For each class, the penalty parameter C is multiplied by that class's weight. Specifically, by increasing the weight of the Structural and All Unacceptable classes and reducing the weight of the Textural class. However, while this increased the number of samples correctly predicted as Structural and All Unacceptable, it also significantly reduced the number of samples predicted as Textural, thereby reducing both accuracy scores.

Another method for handling unbalanced data is to add a random super-sampling of data points in the training set that fall in the smaller categories. Applying this method to k-NN increased the number of All Unacceptable samples correctly predicted, but no Structural samples were correctly predicted and the number of correctly predicted Textural samples decreased as super-sampling increased [27, 28].

Alternatively, instead of super-sampling data in small categories, it is possible to produce artificial, but statistically believable data points for training. To produce an artificial training sample we calculate the mean and standard deviation of each feature for each class requiring extra samples. Then, create a new random feature vector for the class using a normal distribution with the feature mean and standard deviation. These artificial samples are added to the training set in an attempt to balance the classes. Applying this method to k-NN improved raw accuracy to 76% and accuracy to 82.91%. Applying this method to SVM did not improve classification accuracy and fewer samples were correctly predicted as Textural. While this method can balance a dataset, the artificial samples are heavily biased towards the data already used in the training set. These artificial samples may not represent reality.

4.4 Future Work

Our experiment could be improved in a number of ways. First, collecting more data for Structural and All Unacceptable classes could help to reduce class imbalance and improve learning and prediction accuracy. Second, we could try other learners more suited to unbalanced datasets.

We chose to use the image diagonal as a measurement for how much data adjacent to the hole should be used for classification. Alternatively, we could have used the size of the image hole to determine the amount of hole adjacent data to use. This would give larger holes more known data to select an inpainting algorithm, which may improve results.

Another approach to inpainting using our proposed method is to subdivide the hole according to saliency of the vicinity. Then, fill each sub-hole using the inpainting algorithm predicted by the classifier. The advantage of this strategy is that holes covering a highly-textured region and a smooth region would be divided on that line and each portion filled with the most appropriate algorithm creating a more convincing fill.

Data in the All Unacceptable class could be used as an interesting benchmark for future inpainting research. It may also be used to learn why inpainting algorithms perform poorly on these images and discover ways to improve the state of the art.

The methodology discovered in this chapter shows that it is possible to use machine learning to select an appropriate inpainting algorithm for filling holes in 2D images. However, the end-goal is to apply this to S3D images. A few changes would be needed to apply this method to S3D images. First, it is important that pixels used for the hole vicinity belong to the current and adjacent disparities to the hole disparities. If pixels from distant disparities are included in the vicinity, then information regarding distant objects would be used in the decision process. These tainting pixels could negatively impact the classification since they belong to regions that are not relevant. Secondly, there are different types of holes in S3D images that should be considered, such as holes created from object removal to holes created by damage, and holes created by disparity adjustment.

4.5 Conclusion

In this chapter we illustrated that interpolative, structural and textural inpainting algorithms produce convincing fills for certain types of holes. Our initial proof-of-concept demonstrated that it was possible to produce a classifier that selects a convincing fill 83% of the time using only hole measurements and measures of saliency in the surrounding region. With further study, we believe these results could be significantly improved.

Now that we have corrected the disparity and filled any holes in the S3D image, stylization can be applied. The next chapter discusses how the layer-based method used for disparity adjustment can be used to apply stylization to S3D media consistently.

Chapter 5

Stereo Consistent Stylization

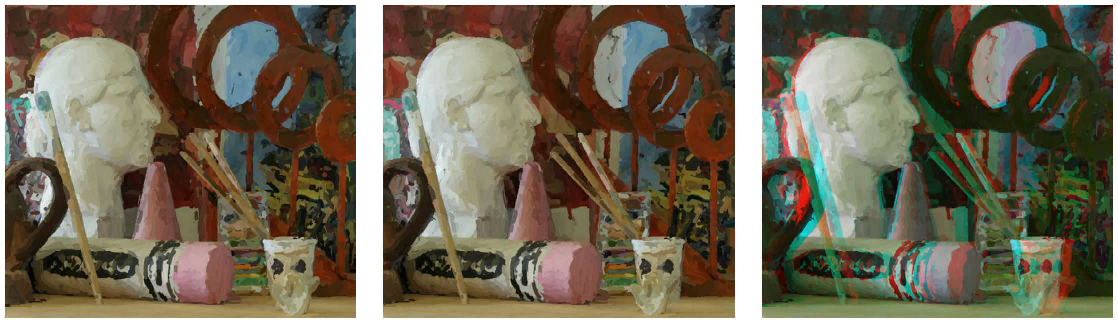

(a) left view (b) right view (c) anaglyph

Figure 5.1: A stereoscopic image stylized with painterly rendering. All anaglyph images were prepared for red/cyan glasses. Image courtesy of the Middlebury Stereo Dataset [73].

5.1 Introduction and previous work

In stereoscopic 3D images, consistency between left and right views is key to visual comfort and stereopsis. An image is consistent when objects appearing in both views match visually. If the colour, texture, or rendering of an object differs between views, then the pair is inconsistent and the image is less comfortable to view [9, 47].

There are many image stylization algorithms that produce artistic effects when applied to a single image, ranging from simple image filters to complex painterly rendering algorithms. As stereoscopic 3D smart phones and cameras become available, users will expect to be able to apply the same stylizations to stereoscopic photos that they currently do to single images, and they will expect the stereo results to be similar to the single image results. However, the straightforward approach of applying stylization algorithms to the views in a stereo pair can introduce artifacts that lead to viewer discomfort. Richardt et al. [70] classify stylization artifacts as binocular rivalry, which presents itself as strong differences between views, the shower door effect, where it appears as though we are viewing the scene through a dirty glass panel, and randomness, which appears as random noise between the views. In their paper, they proposed a method for estimating consistency between stereo pairs, which they tested using pairs where the left and right views were stylized independently. They found that viewer discomfort arises primarily when areas of the views that correspond to the same location in a scene are not stylized in the same way, creating binocular rivalry (Figure 5.2(e)–5.2(f)). A user study verified that mismatched stylization artifacts do indeed cause viewer discomfort, and, further, that the degree of discomfort is proportional to how substantial the differences are between the two views.

Standard stroke-based rendering algorithms [30, 31, 52], which produce a hand-painted appearance, introduce many artifacts when applied independently to the left and right views of a stereoscopic 3D image. Artifacts arise from paint strokes spanning large changes in depth, particularly object boundaries, and from the overall uncorrelated placement and ordering of strokes (Figure 5.2(g)–5.2(i)).

Stavrakis and Gelautz presented several stereoscopic adaptations [26, 78, 79] of Hertzmann's original painterly rendering algorithm [31]. In their work, they generate strokes in one view, then use the disparity map (see Chapter 2) to warp the strokes into the other view. They also generate additional strokes in the other view to fill in regions that were occluded in the first view. However, these new strokes can extend into regions that are visible in both views. Since they exist in only one view, they can cause inconsistencies, as shown in Figure 5.3. These inconsistencies exist in all versions of their work, but are most visible in their 2005 paper because the other versions used smaller stroke diameters. Their methods also prevent paint from spanning large depth ranges by terminating any stroke that would cross a depth discontinuity before it can do so, similar to Litwinowicz's painterly rendering algorithm that trims strokes to image edges [52]. However, this method leaves unpainted regions in the results, which are filled by compositing the original input views underneath the painted strokes. An area of an image with fine details and large changes of depth will have many strokes that terminate early, leading to many gaps between strokes. In extreme cases, their algorithm might not be able to place any strokes at all.

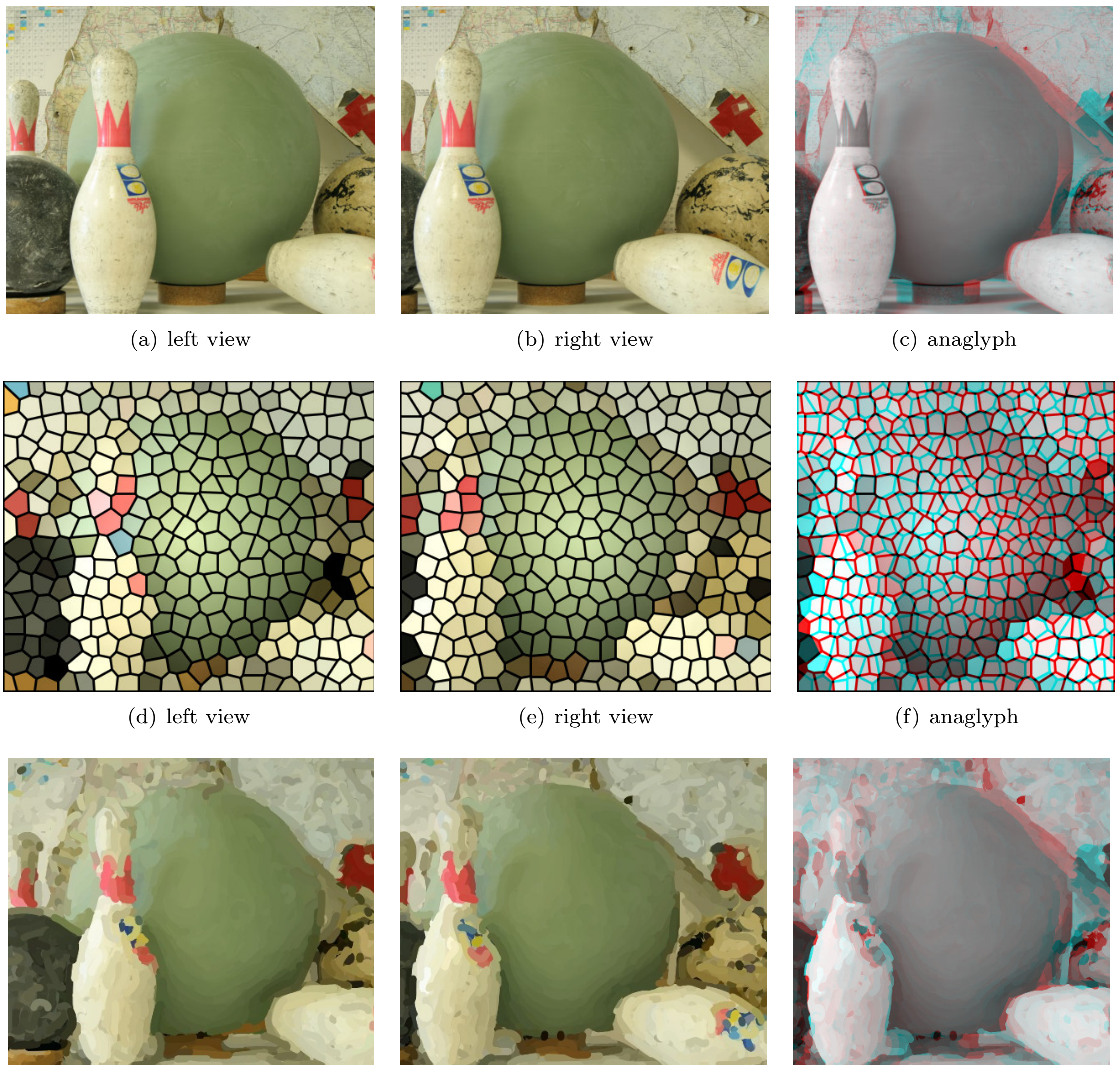
Snavely et al. presented a method for stylizing 2D video that uses depth information to direct hatches or painted strokes along surface geometry [77]. However, this method ignores painted surface details, such as polka dots on a flat surface. More recently, Richardt et al. investigated the application of a variety of stylization techniques to RGBZ video [71].

Painterly rendering in three dimensions has an active history, starting with the seminal work of Meier [57]. These techniques use 3D models as input rather than stereo photographs, so they are not directly applicable to our problem, but they share the goal of trying to create a stylized version of a 3D scene. Even the earliest work stressed that working in view space, not in model space, was an important part of generating results that look like paintings. Every frame of Meier's video looks like a painting of haystacks in a field, and not like a painted field full of painted haystacks.

More recently, the OverCoat system [74] greatly advanced three-dimensional interactive painting and stroke rendering. It does not, however, address the issues of generating the strokes automatically. It does address the thorny problem of deciding projected depth ordering of an ordered collection of 3D strokes as seen from different viewpoints. However, their solution can generate different stroke orders for slightly different viewpoints, which could cause inconsistencies if used to generate stereo pairs.

There has been extensive work (for example, by Kolmogorov and Zabin [46]) on reconstructing geometry from stereo photographs. While it would be possible to use this geometry as input to 3D stylization techniques, doing so would generate a scene of stylized objects, and not a stylized scene. An additional problem is that stylization algorithms designed to work on images might require substantial changes before they could work seamlessly on 3D-modeled objects.

In this chapter we present a general approach to stylizing stereoscopic 3D images that generates



(a) left view (b) right view (c) anaglyph

(d) left view (e) right view (f) anaglyph

(g) left view (h) right view (i) anaglyph

Figure 5.2: Inconsistency artifacts that arise when stylization is applied independently to left and right views. (a)–(c) are the original image, (d)–(f) shows the Adobe Photoshop Stained Glass filter, and (g)–(i) shows Hertzmann's painterly rendering algorithm. These pairs exhibit binocular rivalry and randomness caused by applying the stylization filter to the views independently. Image courtesy of the Middlebury Stereo Dataset [73].

results that resemble stylized single images. This approach guarantees consistency between left and right views, producing stylized images that are comfortable to view. The input to our algorithm is a stereoscopic pair of views and their corresponding disparity maps, and the output is a corresponding stereoscopic pair with stylization applied. To ensure consistent stylization, we split the input views into discretized disparity layers, as introduced in Chapter 3, and merge corresponding left and right layers into combined layers where stylization takes place. We then reassemble the stylized layers into a final image. Our method works with many off-the-shelf 2D non-photorealistic filters such as those found in Adobe Photoshop, and the stylized results can be further improved by tailoring these rendering algorithms to stereoscopic 3D. As a proof of concept, we present a consistent stereoscopic adaptation of Hertzmann's painterly rendering algorithm that uses disparity information to assist in stroke placement. Our method does not wrap stylization filters around the 3D surfaces, but instead

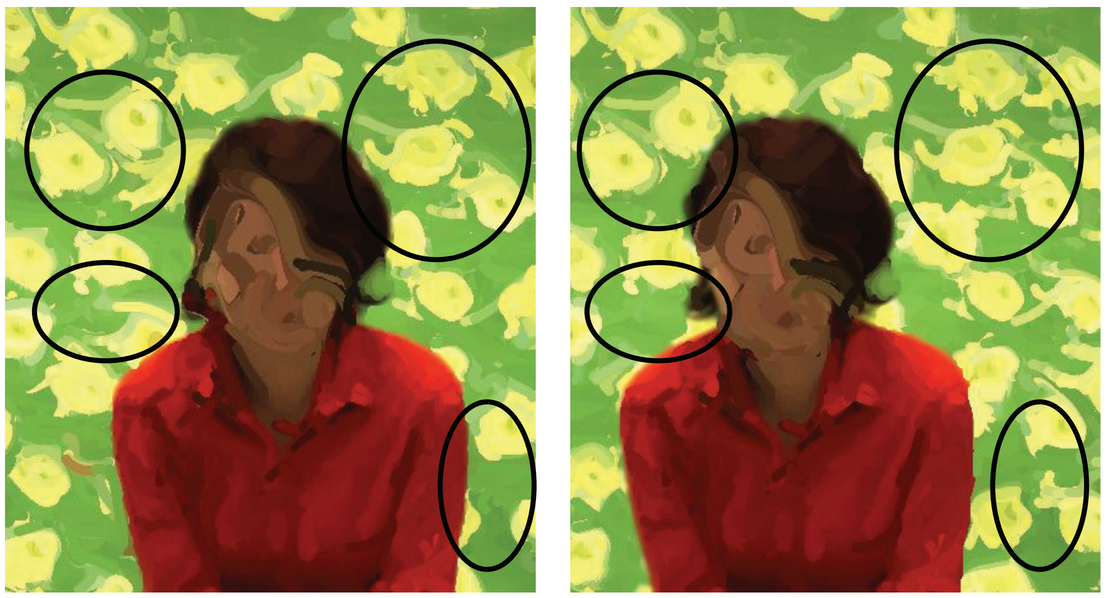


Figure 5.3: Comparing left and right views of a stereoscopic 3D image stylized with the technique of Stavrakis and Gelautz [79]; the images were taken from that paper. The circled regions identify inconsistencies between views.

places stylization in depth similar to the painted strokes of the stereoscopic works of Salvador Dalí, for example, his 1972 S3D piece “The Sleeping Smoker” .

First we detail our general approach to stylizing stereoscopic 3D images (Section 5.2). We then move on to the special case of stroke-based painterly rendering (Section 5.3). We close by discussing the results of a user study to evaluate our algorithm (Section 5.4). In Sections 5.2.5–5.2.7 we discuss methods for eliminating or reducing visible layer discretization for stylization filters that rely on global image properties or introduce structures such as stipples. We also discuss how our method can be used to achieve several different effects including partially stylized images and customized depth of field.

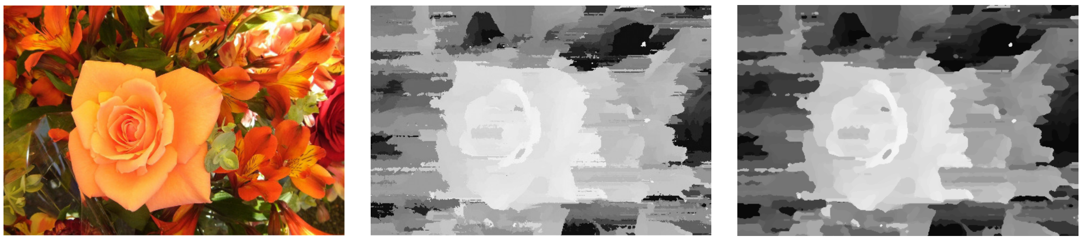
5.2 Approach

Our method relies on disparity maps, which we assume have been provided and are of reasonable quality. However, we note that many disparity maps contain noise, which will negatively impact our algorithm. To reduce noise in computed disparity maps we apply a median filter [87] before using them (Figure 5.4(c)). When necessary, we apply the disparity correction algorithm presented in Chapter 2.

Our approach to stereoscopic image stylization uses the observation that for a given disparity d, some areas occluded in the right view are visible in the left view, and vice versa. Merging the pixels of left and right views at disparity d produces a *merged view* which contains all available information at d (Figure 5.6). This is the same layer-based approach that we use for adjusting the disparity, however, it differs in that we will construct masks to clip the stylizable material and merge adjacent layers to provide more material for stylization algorithms to work with.

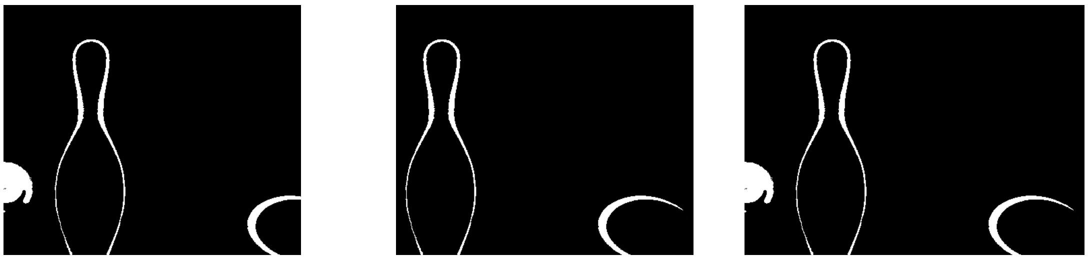
The algorithm is made up of the following three phases, similar to disparity adjustment:

1. **Construct a region mask and merged view** for each disparity d, combining nearby

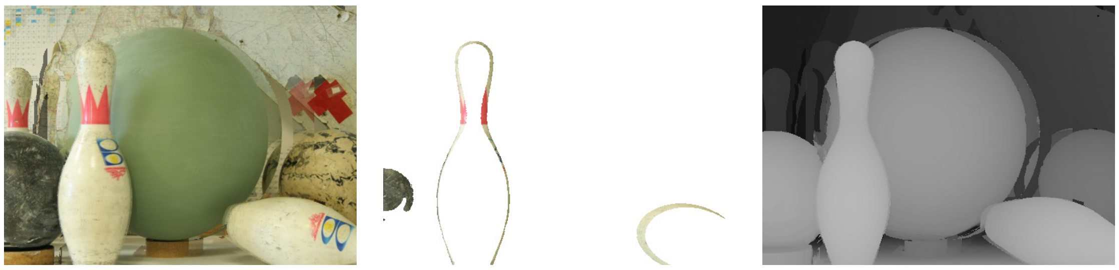


(a) one view of a stereo image (b) raw computed disparity map (c) median filtered disparity map

Figure 5.4: A view from a stereo image, the disparity map computed automatically using computer vision algorithms discussed in Chapter 2, and the median-filtered map. Features of the image are distorted in the raw disparity map, making boundaries and details difficult to identify.

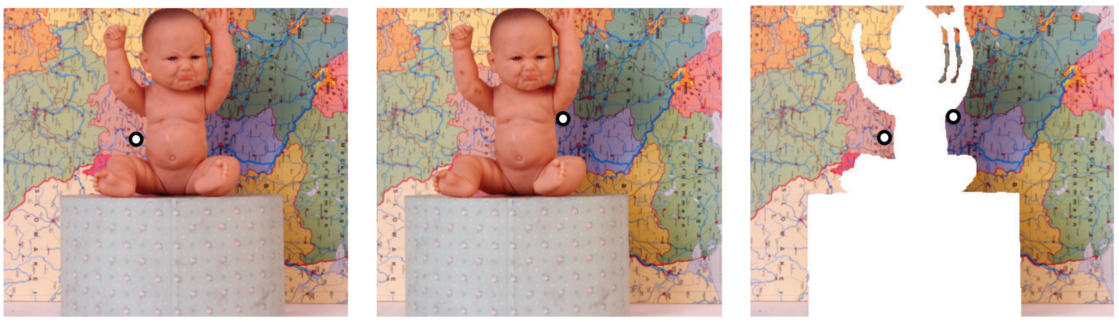


(a) left mask (b) right mask (c) mask



(d) merged view (e) masked merged view (f) merged disparity map

Figure 5.5: The mask, merged view and merged disparity map for a given disparity. Note that only the region corresponding to the mask is valid (correctly aligned) in the merged view and disparity map. Image courtesy of the Middlebury Stereo Dataset [73].



(a) left view (b) right view (c) merged view

Figure 5.6: The region around the dot in the right image is obstructed by the foreground in the left image and vice versa. By removing the foreground and aligning the background using the disparity map, the merged view contains a complete non-obstructed view of the background regions visible in the stereoscopic pair. Image courtesy of the Middlebury Stereo Dataset [73].

disparities if they contain an insignificant number of pixels.

2. **Stylize** each merged view, using the corresponding region mask to mask out pixels at other disparities.

3. **Assemble** the stylized layers into left and right views.

We now describe each of these steps in greater detail.

5.2.1 Construct a region mask and merged view

The input to our algorithm consists of the left view L and right view R in a stereoscopic image pair, together with their associated disparity maps D_L and D_R. All four images must have the same width w and height h. We construct a mask M_d for disparity d as follows:

1. Construct left and right masks M_L and M_R with dimension (w, h), where a mask pixel is 1 if the corresponding image pixel has disparity d, and 0 otherwise.

2. Construct M_d with dimensions $(w + d) \times h$ that is M_L or'ed with the result of shifting M_R d pixels to the right:

$$\begin{aligned} &M_d(0 \le x \le d, y) = M_L(x, y) \\ &M_d(d < x < w, y) = M_L(x, y) \vee M_R(x - d, y) \\ &M_d(w \le x \le w + d) = M_R(x - d, y) \end{aligned} \tag{5.1}$$

To construct the merged view V_d for disparity d, we first construct occlusion masks O_L and O_R, which are 0 for pixels that are visible in both views and 1 for pixels that are occluded in the other view. A pixel is visible in both views if the disparity information maps its location in one view to a location in the other view that maps back to the original location. It is occluded in the other view if it maps to a location that does not map back to the original location.

$$\begin{aligned} &O_L(x, y) = 1 && \text{if } x \text{ - } D_L(x, y) < 0 \\ &O_L(x, y) = 1 && \text{if } D_L(x, y) \ne D_R(x - D_L(x, y), y) \\ &O_L(x, y) = 0 && \text{if } D_L(x, y) = D_R(x - D_L(x, y), y) \\ &O_R(x, y) = 1 && \text{if } x + D_R(x, y) > w \\ &O_R(x, y) = 1 && \text{if } D_R(x, y) \ne D_L(x + D_R(x, y), y) \\ &O_R(x, y) = 0 && \text{if } D_R(x, y) = D_L(x + D_R(x, y), y) \end{aligned} \tag{5.2}$$

Construct V_d, the merged view with dimensions $(w + d, h)$:

$$\begin{aligned} &V_d(0 \le x \le d, y) = L(x, y), \\ &V_d(d < x < w, y) = \begin{cases} L(x, y) & \text{if } O_R(x - d, y) = 1 \\ R(x - d, y) & \text{if } O_L(x, y) = 1 \\ \frac{L(x,y)+R(x-d,y)}{2} & \text{otherwise.} \end{cases} \\ &V_d(w \le x \le w + d, y) = R(x - d, y), \end{aligned} \tag{5.3}$$

Effectively we shift the right image to the right by d pixels, and construct a merged image by taking pixels from one image when they are not available in the other, and use weighted interpolation when pixels are available in both images, as in Section 3.3.1.

For some stylization algorithms, such as the painterly rendering algorithm we present in Section 5.3, a merged version of the disparity map is required. We construct VD_d, the merged disparity image for disparity d with dimensions $(w + d, h)$ from the left and right disparity maps (D_L and D_R) in exactly the same way that we construct V_d.

Figure 5.5 provides a sample mask, merged view, and merged disparity map.

Combining Disparity Levels

For some images, the number of pixels at a given disparity d may be very small, producing noisy, disconnected regions in the mask. Many stylization methods do not work well with such small input areas. Therefore, it may be desirable to combine the pixels at these disparities with the pixels of neighbouring disparities to reduce noise and create larger regions.

On the other hand, if the range of disparities in a combined level is too large, merging the left and right views may produce undesirable "double vision", since some pixels will not correctly align. We therefore provide a user-tunable parameter that limits the range of disparities that can be combined. We find that combining up to 5 adjacent disparity levels works well for most images, striking a balance between having levels with too few pixels and levels that combine too great a depth range. Note that this parameter is an upper limit on combination; if a given disparity level has enough pixels filled, there is no need to combine it with others.

A disparity level that merges disparities d_0 through d_1 is treated as a level with disparity $(d_0 + d_1)/2$ and has width $w + (d_0 + d_1)/2$.

5.2.2 Stylize

Stylization is applied only to the region of a merged view V_d corresponding to the current disparity layer. This region is found by applying the mask M_d as an alpha channel to the merged view V_d, creating an image that has pixels from the merged view where the disparity matches d, and is transparent elsewhere. We call this image the *mask view*. The desired stylization is applied to each mask view separately, giving a stylized view S_d.

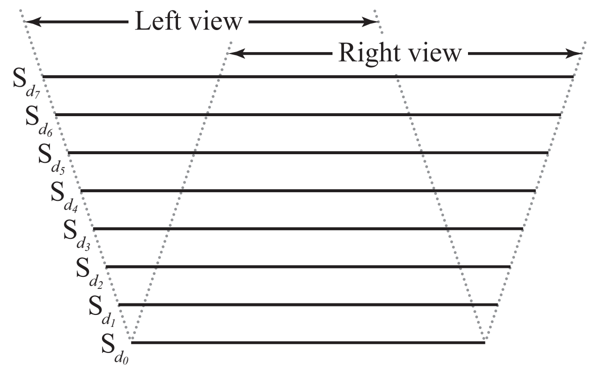
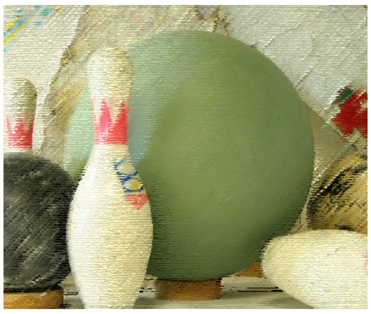
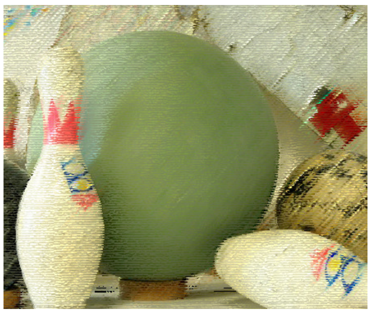


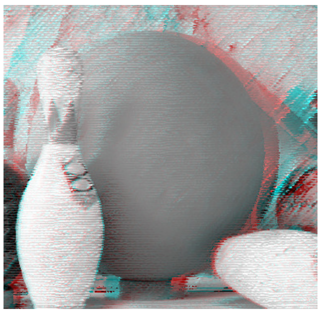
Figure 5.7: A stack of stylized layers, viewed from their top edges. We composite the left- and rightmost portions of each stylized layer to create the stylized left and right views.

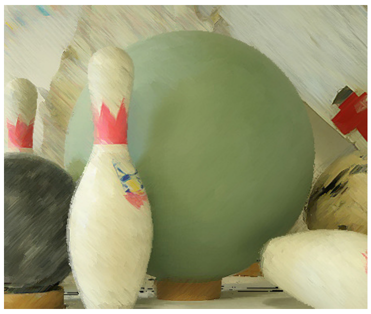
5.2.3 Assemble

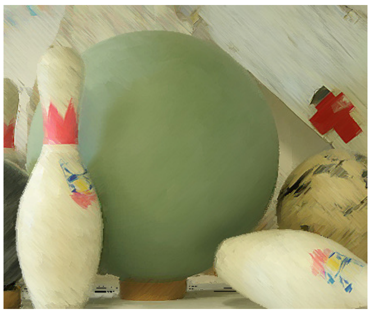
After applying a stylization filter to the mask views we have a set of stylized views $(S_{d_0}, S_{d_1}, \ldots S_{d_n})$. Because of combined disparity levels, and because there may be disparities that do not occur in

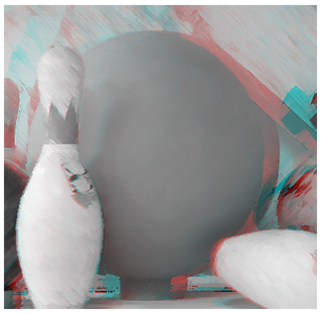

(a) Rough Pastels left view

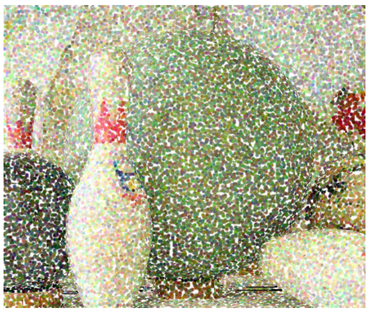

(b) Rough Pastels right view

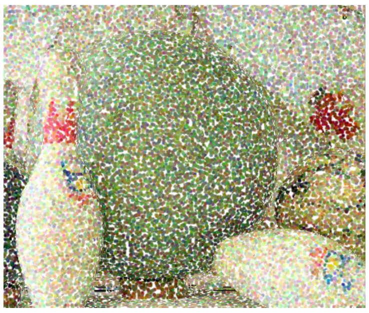

(c) Rough Pastels anaglyph

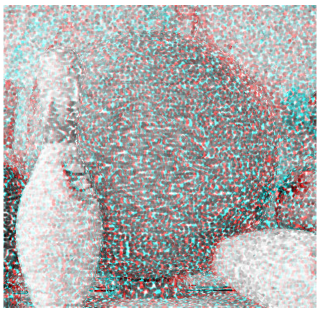

(d) Angled Strokes left view

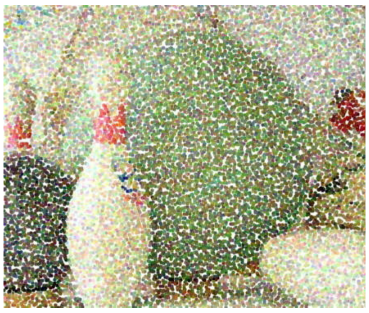

(e) Angled Strokes right view

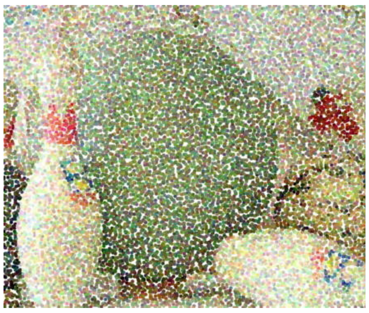

(f) Angled Strokes anaglyph


(g) Pointillize left view


(h) Pointillize right view


(i) Pointillize anaglyph


(j) Pointillize independent left view


(k) Pointillize independent right view

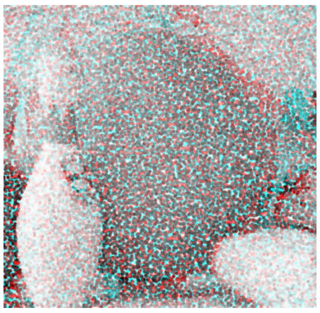
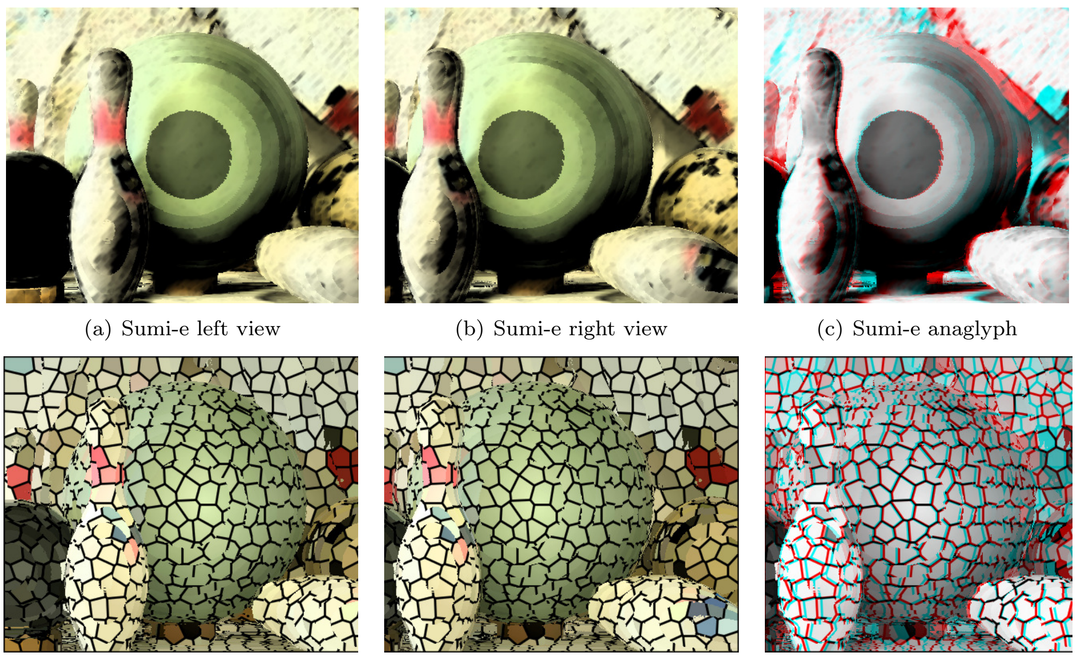

(l) Pointillize independent anaglyph

Figure 5.8: Stereoscopic 3D image stylization using our method with Photoshop filters. In these examples the discretized layers are not obvious. For comparison, (j)–(k) shows the result of applying the Pointillize filter independently to the left and right views. Image courtesy of the Middlebury Stereo Dataset [73].

the source image, the disparities associated with the views may not be consecutive—the set might contain S_0, S_3, S_8, and so forth, without intermediate disparities. Each image S_{d_i} has dimensions $(w + d_i) \times h$. The final left and right views are constructed from these images as follows:


(a) Sumi-e left view (b) Sumi-e right view (c) Sumi-e anaglyph

(d) Stained Glass left view (e) Stained Glass right view (f) Stained Glass anaglyph

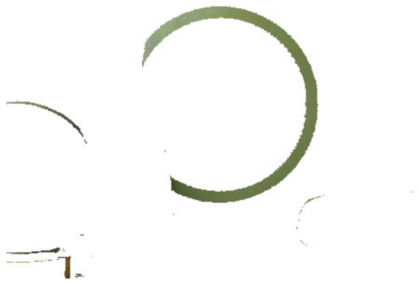
Figure 5.9: Stereoscopic 3D image stylization with Photoshop filters. In these examples the discretized layers are strongly visible, but the pair is consistent. Image courtesy of the Middlebury Stereo Dataset [73].

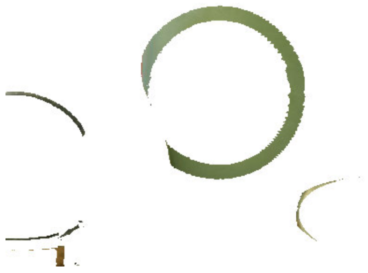
1. For each image S_{d_i}:
   (a) Let L_{d_i} be the left region of S_{d_i}, taking the pixels from columns 0 through w.
   (b) Let R_{d_i} be the right region of S_{d_i}, taking the pixels from columns d_i through $w + d_i$.
2. Composite all the the L_{d_i} images, starting with the farthest and ending with the closest, into the stylized left view.
3. Composite all the the R_{d_i} images, starting with the farthest and ending with the closest, into the stylized right view.

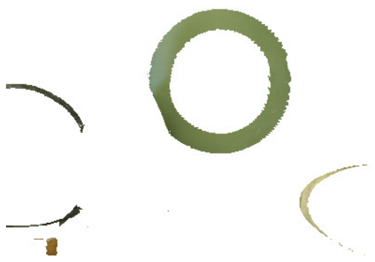
After re-constructing the left and right views, they can be converted to an anaglyph or other 3D image format. Effectively we are constructing a stack of matted images, and using isometric projections along the left and right borders of the stack as the left and right views; see Figure 5.7. Because these views are of the same 3D scene, they are guaranteed to be consistent.

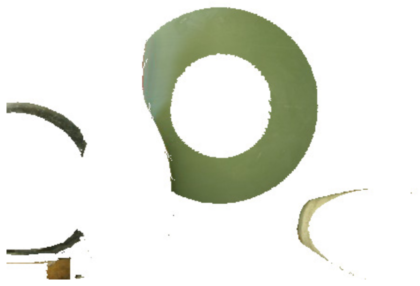
5.2.4 Results

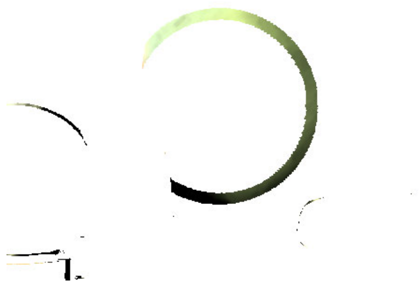
We tested a variety of non-photorealistic Photoshop filters. While all stylized stereoscopic 3D images produced with these filters are consistent, some filters produced more natural images than others. Filters that stylize based only on the local neighbourhood of a point, such as Rough Pastels, Angled Strokes, and Pointillize, worked well with our layered stylization algorithm, as demonstrated in Figure 5.8. Our discretized layers are hard or impossible to detect in the final image. For comparison, Figure 5.8(j)–5.8(l) shows the results of applying the Pointillize filter to the left and right views independently in the style of Stavrakis and Gelautz [79]. The randomness

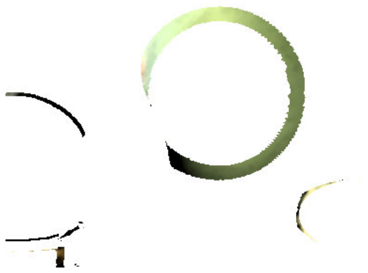

(a) original masked view A

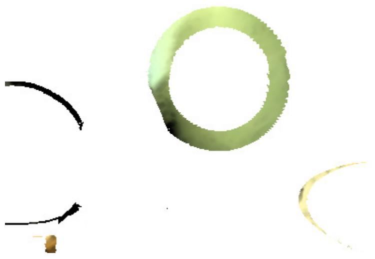

(b) original masked view B


(c) original masked view C


(d) original views stacked


(e) Sumi-e stylized view A


(f) Sumi-e stylized view B


(g) Sumi-e stylized view C

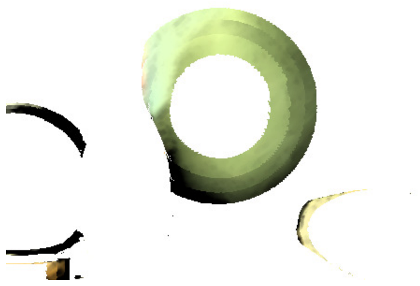

(h) Sumi-e stylized views stacked

Figure 5.10: A close-up view of layer discretization caused by filters that use global image properties. Notice that the rings' brightnesses differ in each of the three views, leading to visible banding when the layers are assembled. Image courtesy of the Middlebury Stereo Dataset [73].

introduced by the filter makes it difficult to fuse the stylized views and greatly reduces the sense of depth present in the stylized image.

Other filters, including Sumi-e and Stained Glass, worked poorly, despite producing consistent images, as shown in Figure 5.9. Sumi-e and related filters use all available input data to compute the stylization at a point, rather than stylizing based only on the point's immediate neighbourhood. Since each masked view contains only a portion of the image, the filters adjust properties like colour and contrast inconsistently between layers.

Filters such as Pointillize and Stained Glass introduce overall structure that was not present in the original image. When applied to a stereoscopic 3D image, the result is a consistent image; however, the structures get shattered by the layering, producing an effect that is different from the intended stylization. In some of these filters, such as Pointillize, the shattered effect is barely noticable because the structure elements are small. In others, like Stained Glass, the effect is pronounced, as demonstrated in Figures 5.9(d)–5.9(f).

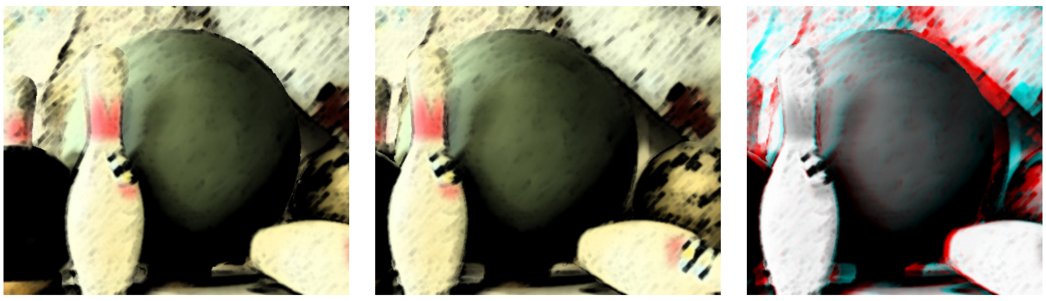

(a) Sumi-e left view (b) Sumi-e right view (c) Sumi-e anaglyph

Figure 5.11: The Adobe Photoshop Sumi-e filter applied to a stereoscopic 3D image using our modified method. Note that previously visible layering is no longer present. Image courtesy of the Middlebury Stereo Dataset [73].

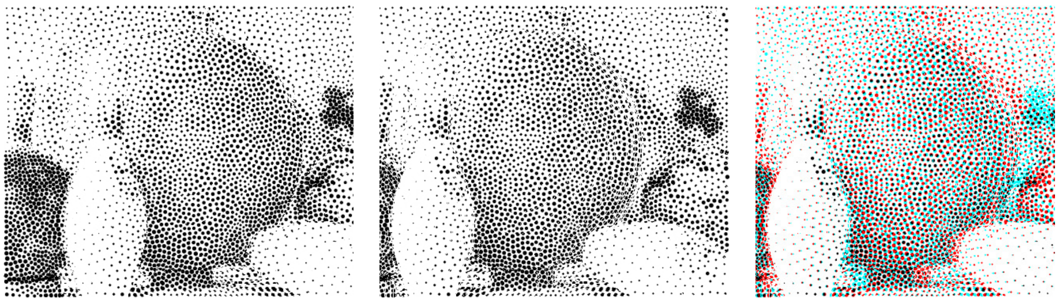
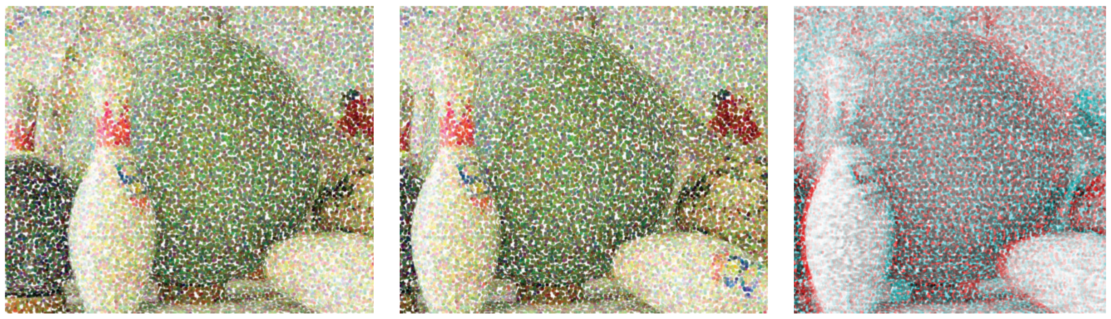

(a) Stipple left view (b) Stipple right view (c) Stipple anaglyph

Figure 5.12: Voronoi stippling algorithm applied to a stereoscopic 3D image using our modified method. Note that while stipple density is consistent across regions, incomplete stipples cropped by mask application cause layering to remain visible. Also note that some objects appear indistinct in the individual left and right views, but the stereoscopic 3D image is both consistent and has strongly visible objects. Image courtesy of the Middlebury Stereo Dataset [73].

5.2.5 Reducing Visible Layer Discretization

Our method for stereoscopic 3D stylization separates image pixels of a particular disparity from the rest of the image prior to stylization. However, as discussed previously, this can produce visible



(a) Pointillize left view (b) Pointillize right view (c) Pointillize anaglyph

Figure 5.13: Photoshop Pointillize filter with mask modification. Note the reduction in fractured points. Image courtesy of the Middlebury Stereo Dataset [73].

layers in the final image for filters that require global image information because individual layers may contain only partial information. Figure 5.10 shows the Sumi-e filter's discretized regions for a few layers. Note how the layers of the original image are not visible when assembled, but, when the Sumi-e filter is applied, those layers are now clearly visible due to discretized regions of colour.

To mitigate this issue, we stylize the merged images prior to separating out the pixels of a particular disparity. This modification folds more global image information into each stylization step, reducing or eliminating banding artifacts.

More precisely, this small modification to our original method works as follows:

1. Compute the region mask M_d and merged view V_d.
2. Apply the stylization filter to V_d, to create a stylized merged view V_d'.
3. Apply the region mask M_d to the stylized merged view V_d' to create the stylized view S_d.
4. Assemble the stylized views as with the original method.

Essentially Steps 2 and 3 are swapped relative the original algorithm.

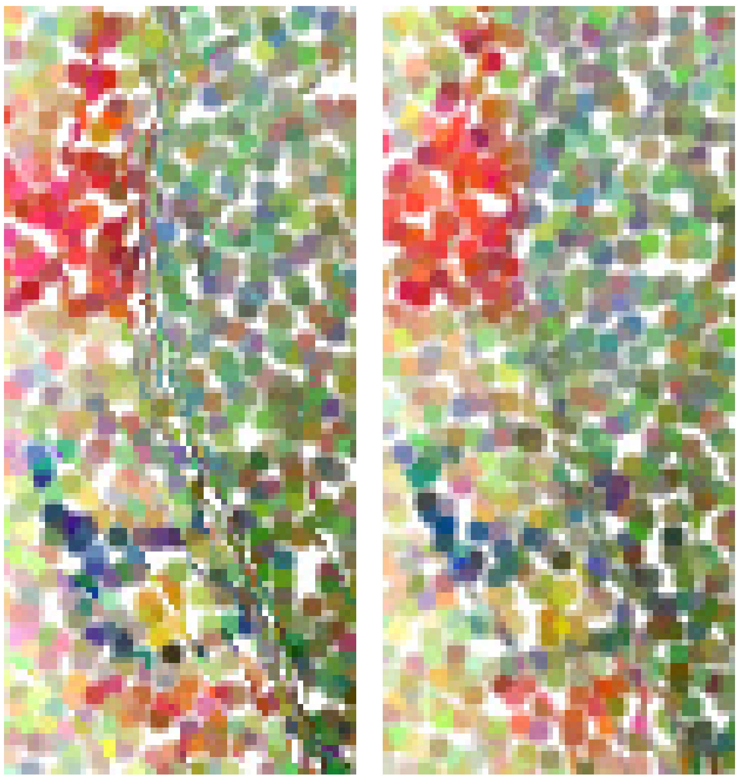
Figure 5.11 illustrates how this small modification to our method eliminates the visible banding with the Sumi-e filter. We found experimentally that this modified method works for other filters that use global image information, such as Watercolour, Dark Strokes and Chalk and Charcoal. However, the modified method does not work for all filters.

Specifically, we find that filters can be divided into three groups:

1. Filters such as Gaussian Blur that blend data from neighbouring regions, possibly placing some pixels outside the masked area, work best with the original method presented in Section 3.3. It prevents image fragments from bleeding into regions whose disparity differs.
2. Filters that use global image information work best with the modified filter presented above.
3. Filters that do not use global information and do not cause bleeding work well with either method.

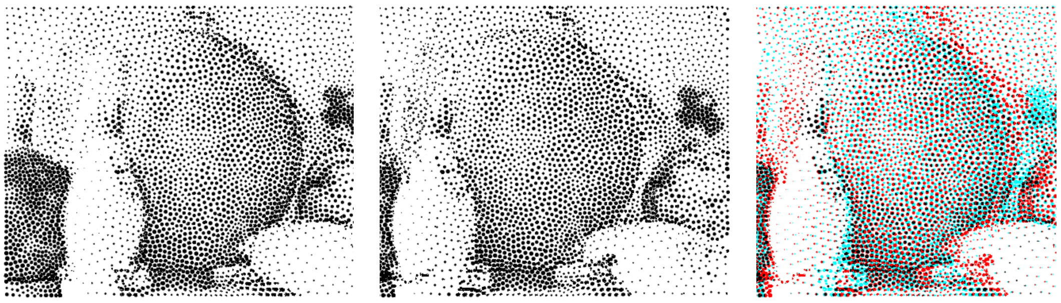
Structured Stylization Filters

There are two main types of stylization filters that introduce structures:



(a) Pointillize filter before correction (b) Pointillize filter after correction

Figure 5.14: Reducing the appearance of fractured points. Note how the edges of the bowling pin in the left image appear sharp and fractured, but the edge of the bowling pin in the right is softer and less fractured. Image courtesy of the Middlebury Stereo Dataset [73].



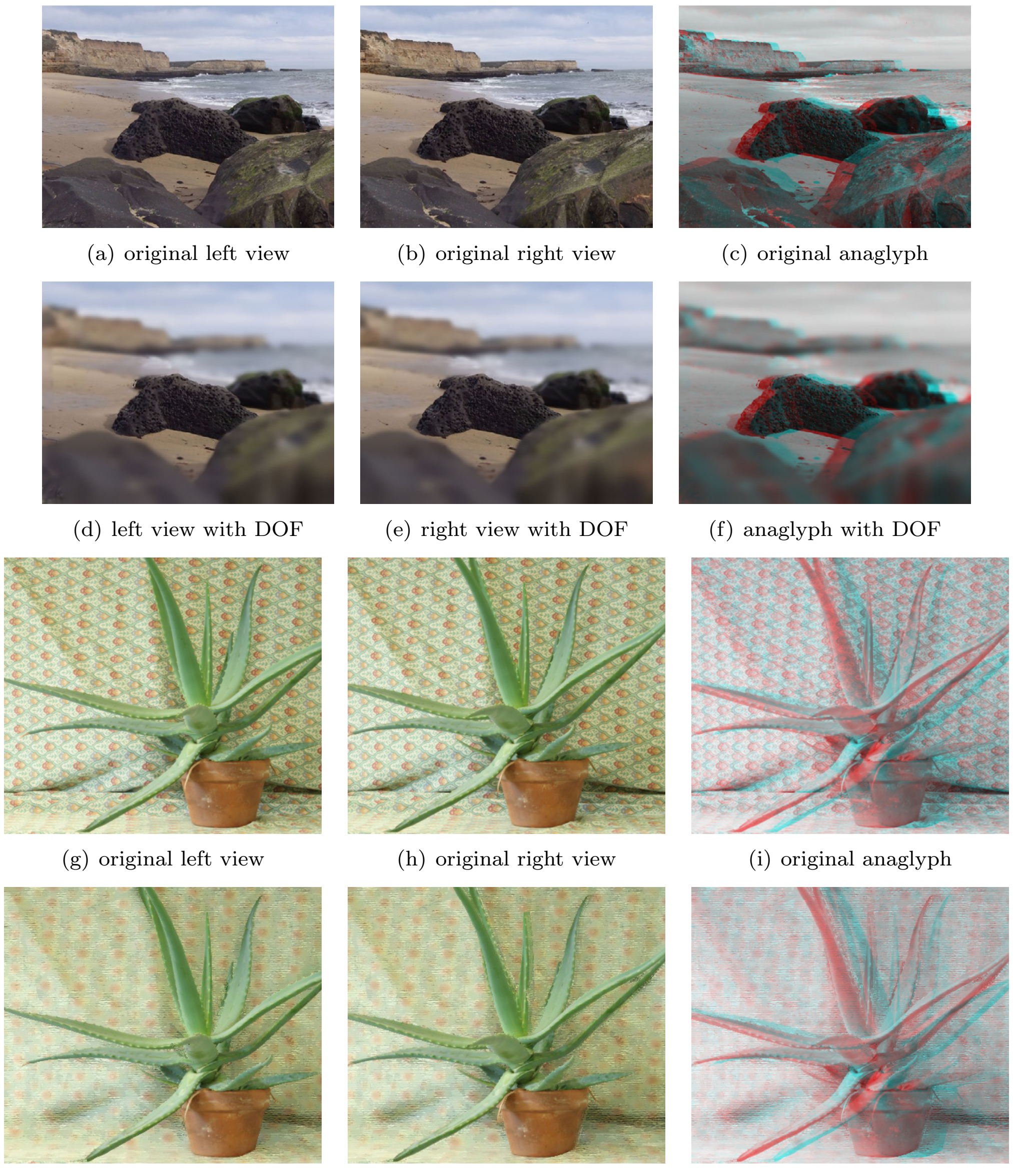
(a) Stipple left view (b) Stipple right view (c) Stipple anaglyph

Figure 5.15: Voronoi stippling algorithm applied to a stereoscopic 3D image using our modified method with masks extended to reduce fractured stipples. Furthermore, the non-stippled regions were transparent in the stylized views so that no fractured stipples remain. Image courtesy of the Middlebury Stereo Dataset [73].

1. Those that distribute structures uniformly over the image, such as Pointillize and Stained Glass, and

2. Those that distribute structures non-uniformly over the image, such as Weighted Voronoi Stippling [75].

Filters in Group 1 should follow the rules outlined previously when selecting a method for stylization. Filters in Group 2 should use our modified approach because structure distribution is



(a) original left view (b) original right view (c) original anaglyph

(d) left view with DOF (e) right view with DOF (f) anaglyph with DOF

(g) original left view (h) original right view (i) original anaglyph

(j) stylized left view (k) stylized right view (l) stylized anaglyph

Figure 5.16: Adding depth of field (a–f), and applying Photoshops' Rough Pastels filter to background regions (g–l), to stereoscopic 3D images. Aloe plant image courtesy of the Middlebury Stereo Dataset [73].

determined by global image properties, such as luminance. For example, in Weighted Voronoi Stippling, black stipples are distributed according to tone: darker regions have more stipples and lighter regions have fewer. The method places a user specified number of stipples randomly onto the image. It then iterates using a weighted version of Lloyd's method to shift the stipples towards a distribution that reflects the tone of the region [75]. If our original method is used to produce a stippled stereoscopic 3D image, the distribution of stipples in each layer would differ because each masked region contains only a subset of the image and the regions differ in size. Thus, Weighted Voronoi Stippling and other non-uniform distribution filters should use our modified method to ensure consistent structure distribution across layers. Figure 5.12 illustrates how our modified method works with the Weighted Voronoi Stippling style.

Even after selecting the appropriate method to apply stylization to stereoscopic 3D images, the fractured appearance of structures can remain for some filters. We found that simple modifications to the algorithms, or the masks and/or regions surrounding the mask can reduce fracturing for some filters. For example, using Photoshop, we reduced the fractured appearance of the Pointillize filter by extending the mask area of the effect by half the point size and then softening the edges prior to applying the Pointillize filter. The results are shown in Figure 5.13. Figure 5.14 shows a close-up of the improvement. Alternatively, one could modify the Pointillize algorithm to not cut points off at mask edges.

In Section 5.3, we discuss how to modify Hertzmann's painterly rendering algorithm to work within our framework and produce a painterly rendered stereoscopic 3D image.

5.2.6 Nonuniform stylization

A single stylization need not be applied uniformly across all mask view layers. Combining multiple filters or using different parameters can produce some desirable results. For example, one can create a depth of field (DOF) effect by applying a Gaussian blur to each layer with a different kernel size. The focal point and depth can be adjusted by changing the blurring parameters applied to each layer. There exist methods to simulate depth of field effects, however many of these methods are simple approximations, such as variable-diameter blur, and require workarounds to avoid colour contamination across depth discontinuities. Our proposed method does not have this problem since blur is applied to depths in isolation. We can also apply completely different stylization techniques to different depth ranges in the image. For example, a stylization could be applied to the background while leaving the foreground unchanged. Figure 5.16 demonstrates these effects.

It should be noted that the left and right views of these images are interesting by themselves; they do not need to be viewed in stereo to see the effect. Using a stereo image as input enables new types of stylization that would be difficult or impossible with just a single input image, even if the end result is a single image.

5.2.7 Synthesizing intermediate viewpoints

Our basic method uses the leftmost and rightmost w pixels of each layer to reconstruct the left and right views. However, other sets of w pixels can be used to synthesize other viewpoints. If we take the middle w pixels, for example, we can synthesize a view that is halfway between the left and right views; this would correspond to taking a vertical slice through the center of the stack shown in Figure 5.7. We can do this with or without stylizing the layers. Figure 5.17 shows a synthesized view constructed in this way. Note that some information is missing because there are some areas of the scene that would be visible from the center viewpoint but which are not visible in either the left or right views. We could use some of the stereo consistent fill techniques from Chapter 3 to fill these regions.

Some glasses-free stereoscopic displays require some number of intermediate views between the leftmost and rightmost viewpoints, and our method could be used to provide those views. It also could be used to reduce the interocular separation for a stereo photograph. Close-up photographs taken with a stereo camera can be difficult to view, since fusing the image is equivalent to focusing on an object that is right in front of one's eyes. Reducing the separation can greatly improve the viewing comfort for such images with the trade-off that the perceived depth range shrinks.

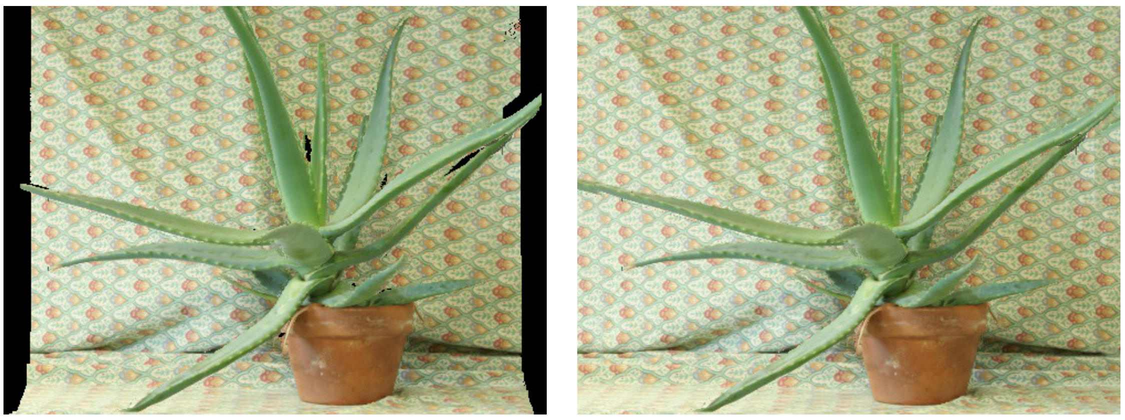


Figure 5.17: The left image shows a synthetic viewpoint halfway between the left and right views of the aloe image in Figure 5.16. The right image uses inpainting to fill areas not visible in the original views. Image courtesy of the Middlebury Stereo Dataset [73].

5.3 Stereoscopic 3D painterly rendering

Our basic method does not exploit the depth information from the scene other than to create the disparity levels. It also can produce narrow regions, which can be problematic for some advanced stylization algorithms such as painterly rendering. In this section, we provide an example of how to adapt a painterly rendering algorithm to use depth and paint into areas that will be later covered with layers closer to the eye, giving better results.

We choose Hertzmann's painterly rendering algorithm as a stable, canonical representative of stroke-based rendering [31]. It is also the basis of Stavrakis's work, making comparisons more fair. When applied to 2D images, it produces a hand-painted effect with long curved strokes and multiple brush sizes. It begins with an input image V and a blank canvas C. At each step, it computes an error image $E = (C - V)$ and looks for points where the error is largest. If this error at a point P is above some tolerance, it begins a stroke there, using the colour of V at P. The stroke extends from P by adding new control points, traveling perpendicular to the image gradient of V so that the stroke follows the direction of least colour change. A stroke terminates if it exceeds a maximum length, or if it has achieved some minimum length and extending it would take it into an area that does not need paint (the value of E there is already small), or if the colour at the next control point differs from the stroke colour by more than a tolerance. After filling the canvas this way, it then repeats the process, using successively smaller brushes to refine the painting.

As with most stylization techniques, Hertzmann's algorithm produces an inconsistent and hard-to-fuse stereo pair when it is applied directly to left and right views of a stereoscopic 3D image, as shown in Figure 5.2(g)–5.2(i). We could apply Hertzmann's algorithm as-is to the individual masked views in our basic layered approach, giving the results in Figure 5.18(a)–5.18(c), but in regions where the mask is narrow, strokes have very limited space in which to travel. The image gradients may encourage them to travel across narrow regions instead of along them, so the strokes are often very short. Further, there is nothing to prevent larger diameter strokes from partially extending outside of their masked areas, which leads to an overall bloated appearance, and scalloping when strokes start or end near depth discontinuities. Some modifications to the algorithm address these issues:

1. We add extra area for strokes to travel into, so they do not end quickly.

2. We encourage strokes to follow paths of constant depth, especially along depth discontinuities, so they tend to follow narrow disparity layers rather than go across them.

3. We do not let large-diameter strokes extend outside their masks by more than a very small amount.

Observe that it is acceptable to paint in areas that are in front of the current layer, as those areas will be covered up when the closer painted layers are composited to create the final images. Permitting strokes to travel into these regions will encourage the creation of longer strokes. Therefore, we first modify each merged disparity mask M_i to include the three layers in front of the current one, as shown in Figure 5.19. These forward layers are not set to 1 like the pixels of the current mask region, but to intermediate values between 0 and 1, indicating how far in front of current layer they are.

The error images E_i are multiplied by the masks M_i so that regions outside the mask have zero error, preventing strokes from starting or travelling into these regions. Because M_i includes the three closer disparity levels, these levels are included in the error image, but because their values in M_i are less than 1, the values in the error image are attenuated. This allows the strokes to extend into these areas, but usually they will not start there, since these are unlikely to contain the point of greatest error.

To ensure strokes are not prematurely terminated, they should follow surface geometry more strongly than in Hertzmann's algorithm, which does not distinguish between surfaces and patterns. However, stroke direction should not ignore surface patterns either. Therefore, we enhance surface boundaries by creating an image that highlights them, and add it to the input image when computing the image gradient. Surface boundaries are found by computing the edges of the merged disparity map VD_i, and the enhanced gradient for a given layer i is computed using Algorithm 1.

Algorithm 1 Constructing Surface-Boundary Enhanced and Un-enhanced Image Gradients

```
Compute image $L_i$, the input image $V_i$ converted to luminosity values
Compute $VD_e$, the edges of the merged disparity map $VD_i$ using a Sobel operator
Compute $VD_m$ by median filtering $VD_e$
$L_i \leftarrow L_i + VD_m$
$B_i \leftarrow$gaussianBlur$(L_i)$
Compute $D_{xi}, D_{yi}$, the derivatives of $B_i$, using the Sobel filter
Compute $VD_{xi}, VD_{yi}$, the derivatives of $VD_i$, the merged disparity image, using the Sobel filter
```

Algorithm 2 Stereoscopic Painterly Rendering

```
for $i \in D$ do
  $C_i$ is the canvas, equal to image $V_i$
  $G_I \leftarrow$image gradient$(C_i)$ using Algorithm 1
  for each brush diameter $b$ do
    error image $E_i \leftarrow (C_i - V_i) \times M_i$
    for each region $r \in E_i$ do
      if average error of $r > \tau$ some tolerance then
        find $P$, the point in region $r$ of greatest error
        create a stroke starting at $P$ using Algorithm 3
      end if
    end for
  end for
end for
```

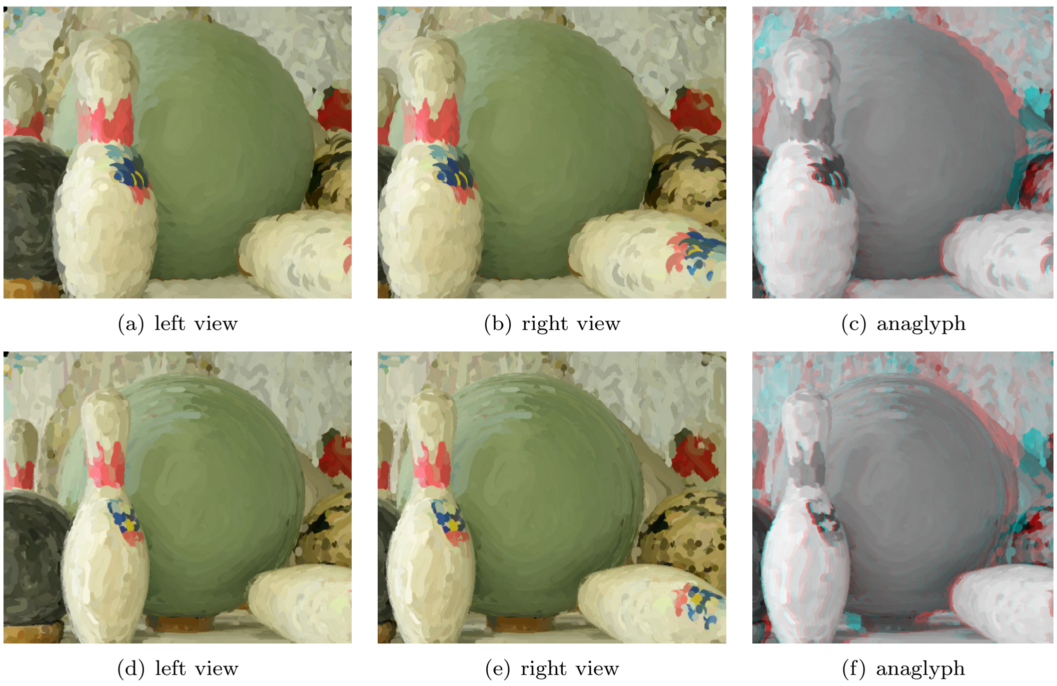


Figure 5.18: Two approaches to combining Hertzmann's painterly rendering algorithm with our stylization method. In (a)–(c), each layer is painted naively, producing short strokes and a scalloped appearance. In (d)–(f) we take direct advantage of disparity information, leading to longer strokes that follow surfaces.

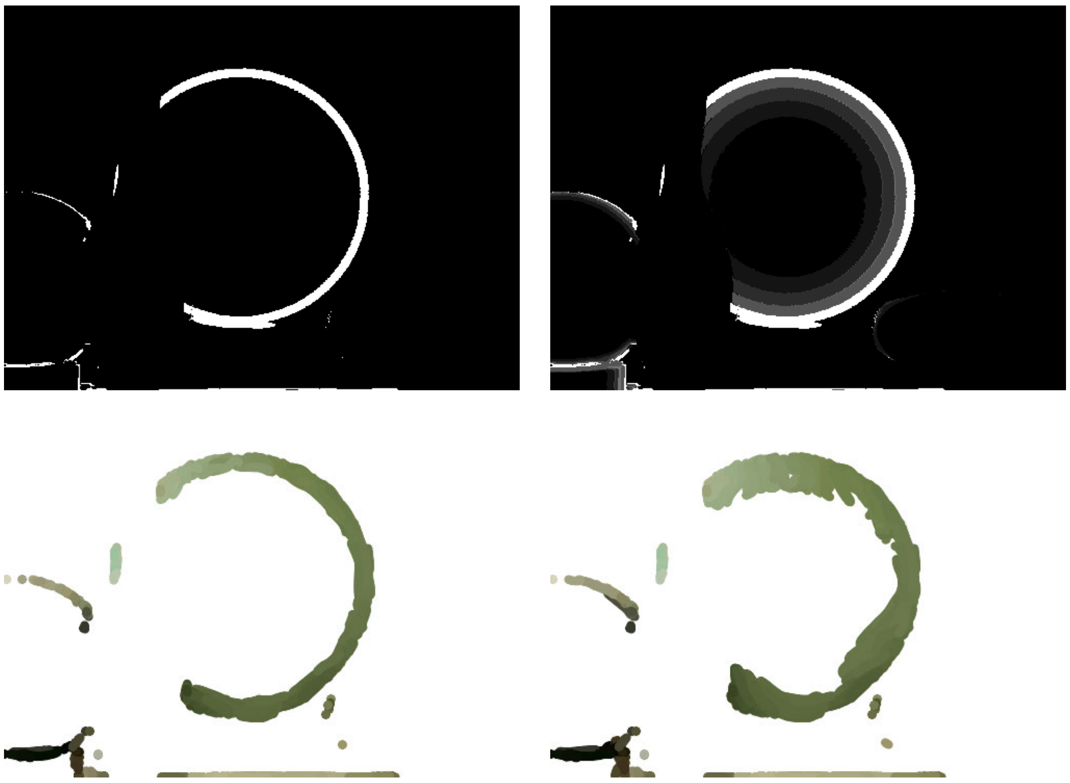


Figure 5.19: A single layer mask and one combining multiple layers, and the results when painting them.

Median filtering the edges of the disparity map prior to adding them to the input removes edge noise and increases the influence of strong edges in nearby regions. Figure 5.20 shows an image in which gradients have been enhanced to emphasize edges.

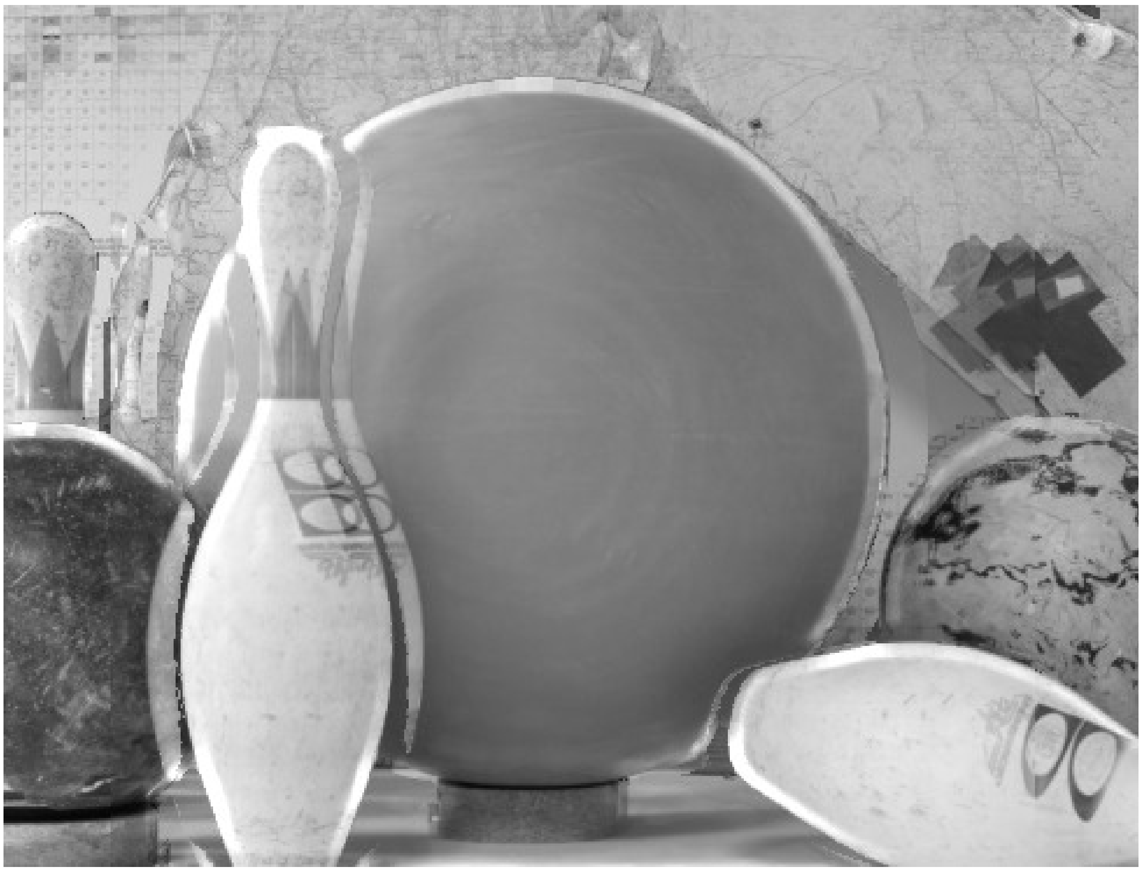


Figure 5.20: The effect of enhancing the edges on the image gradient (L_i). The enhancement encourages strokes to follow edges in the image.

The input to our painterly algorithm is the set of merged views V_d, masks M_d, and merged disparity maps VD_d that correspond to disparities $D = \{d_0, \ldots, d_k\}$. We proceed as described in Algorithm 2.

To further encourage the construction of longer strokes, we modify how strokes are created.

Let i be the current disparity layer, b_j the current brush diameter and P the stroke starting point. A stroke is created using Algorithm 3, as described below.

For every step of stroke generation, we construct two potential next control points. One control point is produced by following the surface boundary enhanced image gradient, and the other, by surface curvature (disparity) gradient alone. The first would continue in the direction of least colour change, the second in the direction of least depth change. If one of these control points is invalid, then the other is used as the stroke's next control point. If both are invalid, the stroke is terminated, and if both are valid, the control point corresponding to the least magnitude gradient is selected. Providing strokes multiple choices for their next control point reduces the probability of early termination.

For our stereoscopic 3D painterly algorithm, a control point P is declared invalid if any of the following conditions are true:

1. More than p pixels within brush diameter b of control point P are outside the masked region and b is not the smallest brush diameter. p is a tunable parameter of the system, with values around 5 giving good results.
2. b is the smallest brush diameter and there are no pixels within b of control point P in the masked region.
3. The difference between the disparities at the stroke's first control point and at P is greater than some user-specified tolerance.
4. The error at P is less than a user-specified tolerance τ and the stroke has reached the user-specified minimum length.
5. The difference between the stroke's colour and the colour at P is greater than user-specified tolerance τ.
6. The magnitude of the gradient at P is close to zero.

Algorithm 3 Create Stroke

```
if more than some tolerance $p$ pixels within $b_j$ of $P$ are not in the mask specified by $M_i$, and $b_j$ is not the smallest brush diameter then
  delete stroke and return
end if
$previous\_point \leftarrow P$
while true do
  $\vec{v_0} \leftarrow (D_{yi}(P_x, P_y), -D_{xi}(P_x, P_y))$ is the perpendicular of the gradient at point $P$ in the enhanced image
  $\vec{v_1} \leftarrow (VD_{yi}(P_x, P_y), -VD_{xi}(P_x, P_y))$ is the perpendicular of the gradient at point $P$ in the merged disparity image
  $P_0 \leftarrow previous\_point + \vec{v_0}$ is a new control point computed using the enhanced image
  $P_1 \leftarrow previous\_point + \vec{v_1}$ is a new control point computed using the merged disparity image
  if $P_0$ and $P_1$ are invalid then
    terminate stroke and return
  else if $P_0$ is invalid then
    $n \leftarrow 1$
  else if $P_1$ is invalid then
    $n \leftarrow 0$
  else
    if $|\vec{v_0}| \geq |\vec{v_1}|$ then
      $n \leftarrow 0$
    else
      $n \leftarrow 1$
    end if
  end if
  add $P_n$ to list of stroke control points
  $previous\_point \leftarrow P_n$
end while
```

Condition 1 keeps larger strokes from extending over depth discontinuities by ensuring that strokes do not travel outside the masked region. This eliminates the bloated, scalloped appearance of Figure 5.18(a)–5.18(c). Condition 2 ensures canvas coverage by permitting the smallest diameter strokes to travel a short distance away from the masked region. Because the stroke diameter is small, the effects of extending over the discontinuity are minimized. Condition 3 prevents strokes from traveling into disparity regions that the user indicates are “too far away”. This particular condition affects layers that represent a range of disparities. Conditions 4, 5, and 6 are part of Hertzmann’s original algorithm and prevent strokes from traveling into regions that do not need refinement, or that differ greatly from the current colour.

Figure 5.18(d)–5.18(f) demonstrates our modified algorithm using the enhanced image gradient. Notice the presence of longer strokes that tightly follow the surface of the bowling ball and pins.

Our method takes approximately five times as long to paint a stereo image as to paint a non-stereo image of the same size. Because each area of the image belongs to four disparity levels—its own level, and up to three closer levels—the painting takes about four times as long. The remaining time is spent in creating the levels. However, our algorithm is readily parallelized since each disparity layer can be processed independently.

5.4 Evaluation

We conducted a user study to evaluate our stereoscopic 3D stylization algorithm. For our study, we produced a set of stylized stereoscopic 3D images using a variety of non-photorealistic filters and asked participants to view each image and rate the viewing comfort, quality of depth reproduction, and overall aesthetic quality of the image on a scale from one to five (poor to aesthetically pleasing). Our test image set included

1. The original stereoscopic 3D photographs;
2. Images where the left and right views have been stylized separately with Photoshop filters or Hertzmann’s painterly rendering algorithm,
3. Images rendered with Stavrakis and Gelautz’s painterly algorithm;
4. Images rendered with our painterly rendering algorithm; and
5. Images where Photoshop filters were applied to our layered stereo approach.

Additionally, for each image, we presented three levels of interocular separation to account for variations in participant viewing comfort.

We recruited 24 participants between the ages of 18 and 40 with stereo vision for our experiment. We relied on participants assessment of their own stereo acuity instead of testing participants ourselves. Since we had a large number images, we divided participants and images into two groups to reduce visual fatigue. These two groups, A and B, and each group viewed a set of 78 stereoscopic images. Each group’s set contained 42 images that were not in the other set and 36 images that were in the other set. Participants in each group viewed their images in the same order, which randomly shuffled different scenes, stylization methods, and separations. They were not informed which algorithm was used to produce each image. For Photoshop filters, each group contained both independent and consistent layered filter application for each source image. For painterly rendering, each group contained independent, Stavrakis, and consistent layered renderings for each source image. Participants viewed images full screen on a stereoscopic display with active shutter glasses, and provided verbal responses to reduce potential eye strain caused by switching between paper and screen.

Figure 5.21 shows the results, comparing our results with other results for the same image. The first set of columns compares our results to stereoscopic 3D images stylized with any other

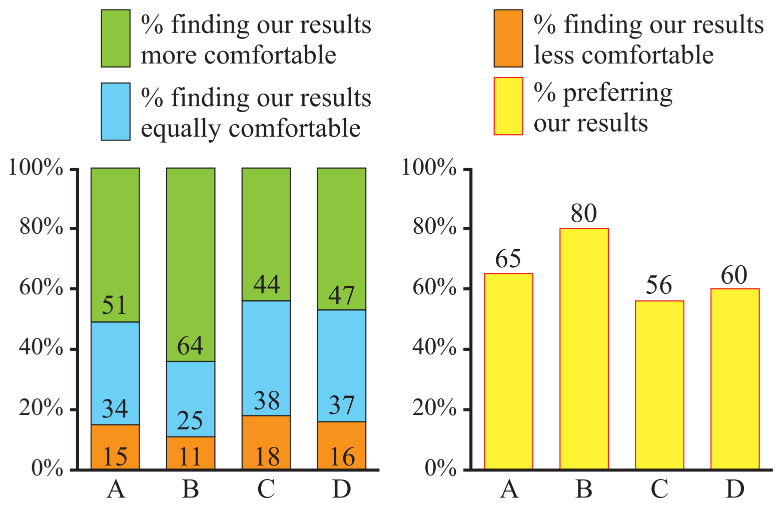


Figure 5.21: Results of a user study comparing our results with other stylization approaches. (A) Our results compared to all other methods. (B) compared to Hertzmann's algorithm [31]. (C) compared to Stavrakis's algorithm [79]. (D) compared to independent Photoshop filter application.

method. Overall, 51% found our results to be strictly more comfortable, and an additional 34% found our results to be equally comfortable. Additionally, 65% of participants preferred our results (as measured by adding together the comfort, depth and aesthetic scores) to those of any other stylization method.

The remaining sets of columns compare our painterly results to Hertzmann's algorithm, our painterly results to Stavrakis's algorithm, and our Photoshop results to applying the filters to the left and right views independently. In all cases our results were judged as comfortable or more comfortable by a large majority of participants, and our results were always preferred. The preference for our results was strongest when compared to Hertzmann's algorithm. This is not a criticism of Hertzmann's algorithm, simply an observation regarding the dangers of stylizing the views separately.[1]

5.4.1 Poor disparity maps

We used a variety of stereoscopic 3D images as input to our algorithm including those with ground truth disparity maps from the Middlebury dataset [73]. We also used stereoscopic 3D photographs taken with a Fuji W3 camera, and for these images we computed the disparity maps using a variety of computer vision algorithms, including that of Rhemann et al. [67]. However, even with the most state-of-the-art algorithms, computed disparity maps usually contain errors—regions where the disparity is obviously wrong. Median filtering can help mitigate these errors, and our stylization algorithm can sometimes produce good results even with faulty disparity maps, as shown in Figure 5.22. Applying our automatic disparity adjustment method discussed in Chapter 2 can also improve the quality of results produced by our stylization algorithm. Many of our user study participants rated such images highly, even though they were generated from disparity maps that had many errors. However, in general, the quality of our results is only as good as the quality of the disparity map. It is worth noting that applying our method has the effect of making the disparity map into the truth, whether it is correct or not. If an area of the image has the

[1]Our test images and results may be downloaded from www.cgl.uwaterloo.ca/lanortha/papers/stereo.html.

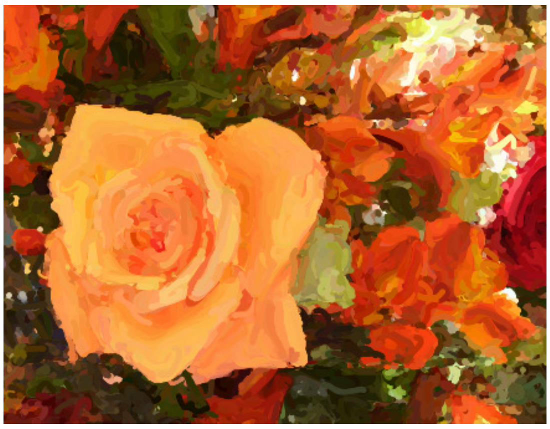
incorrect disparity, the stylization of that area will appear to be at the depth that corresponds to the incorrect disparity, and not the depth of that area in the original stereo image. In some cases the perceived depth of the stylized image may not make sense, and the individual stylized views may contain visual artifacts of disparity errors.

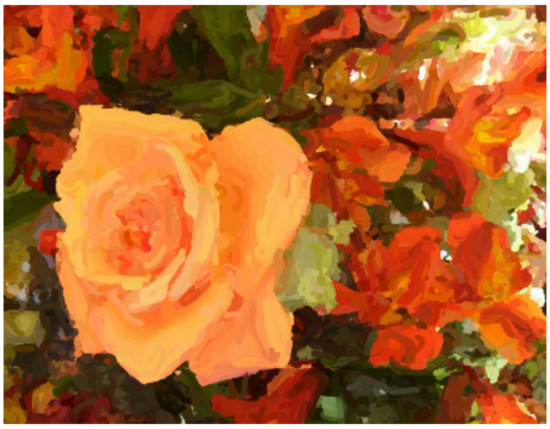
Disparity map computation is an area of active research in the vision community, with results getting dramatically better each year. As their results improve, ours will too.

5.5 Conclusion and future work

In this chapter we presented a method based on our layered approach to disparity adjustment for stylizing stereoscopic 3D images that guarantees consistency between left and right views. We also presented a stereoscopic 3D painterly rendering algorithm. We conducted a user study that demonstrated that our method produces stereoscopic 3D images that viewers find more comfortable than other approaches to stereo stylization.

We found simple modifications to our method to remove the presence of visible layering in stylized results for filters that do not introduce structures. In the future we would like to explore structure-introducing filters such as stippling and stained glass and discover how to modify them to make them more stereo-friendly. In the next chapter, we explore one such problem: rendering S3D line drawings from stereo pairs.


(a) left view


(b) right view

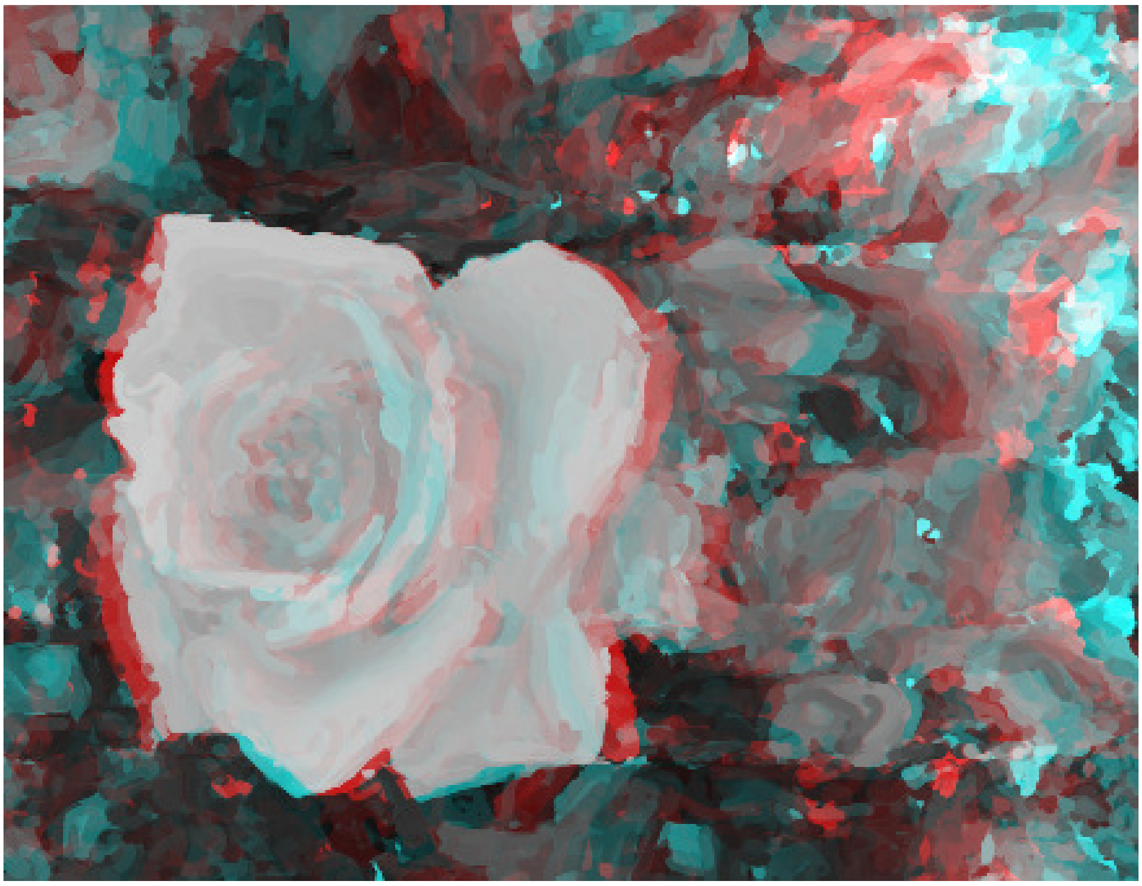

(c) anaglyph

Figure 5.22: A painted image based upon poor underlying disparity maps (shown in Figure 5.4).

Chapter 6

Stylized Stereoscopic 3D Line Drawing

Stereoscopic 3D drawing and painting are over 180 years old, as discussed in Chapter 2. Before Sir Charles Wheatstone introduced the tabletop stereoscope in 1838, he experimented with stereoscopic 3D drawings [88]. During these early days of stereoscopes people often viewed drawings and paintings, such as *The Wheatstone Arch* depicted in Figure 6.1, until stereo photography became commonplace in the 1850s. Although stereoscopes have fallen from popularity, stereoscopic 3D

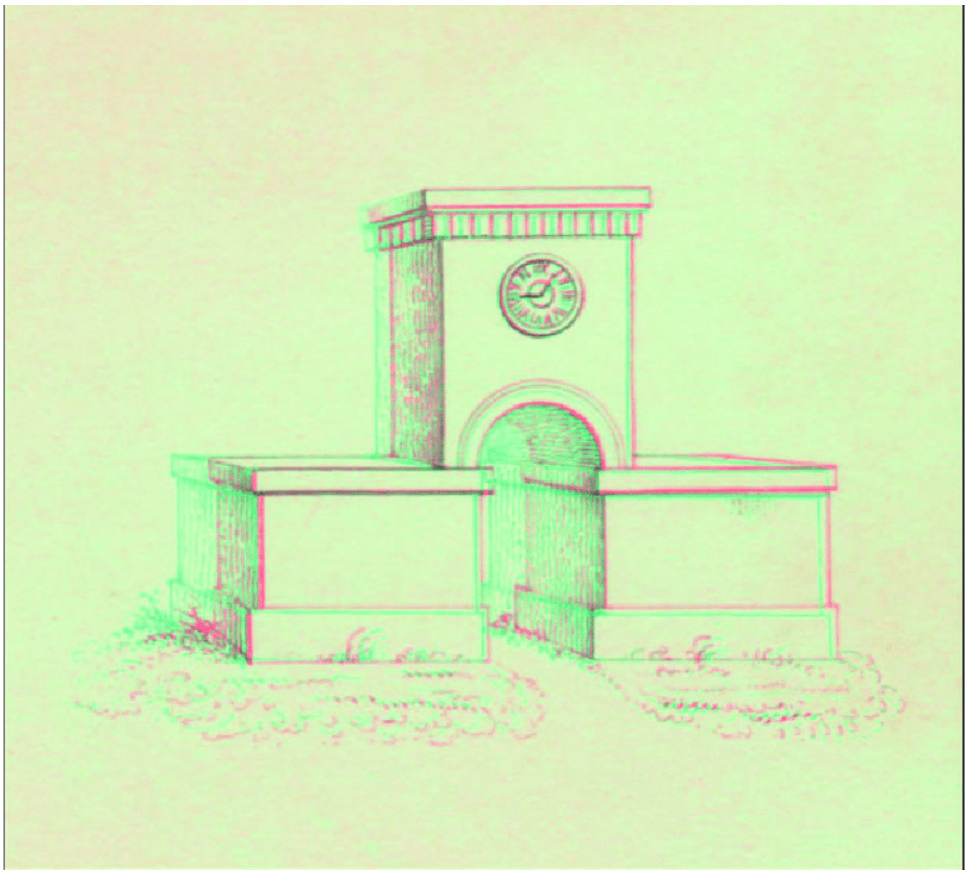
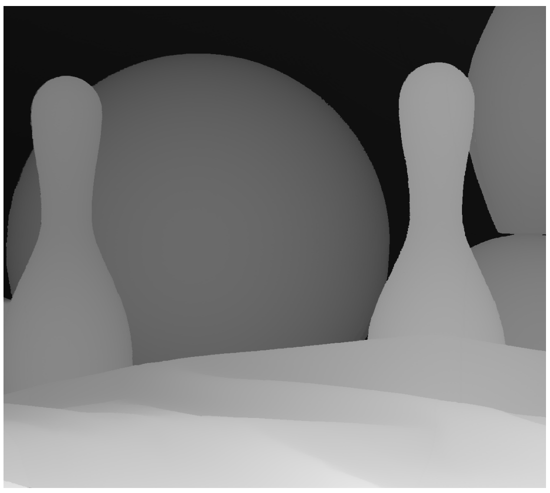
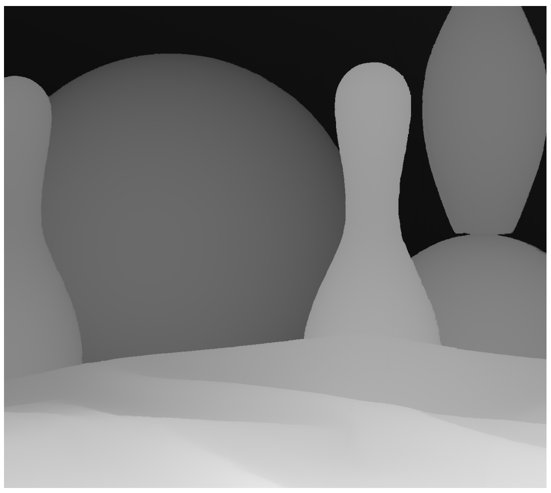
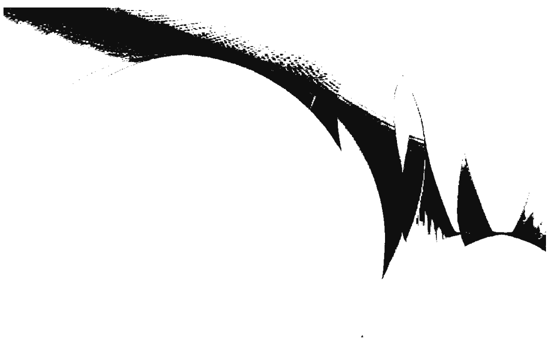


Figure 6.1: *The Wheatstone Arch*, Charles Wheatstone, 1838 [15]


(a) left disparity


(b) right disparity


(c) merged disparity layer for $d = 15$

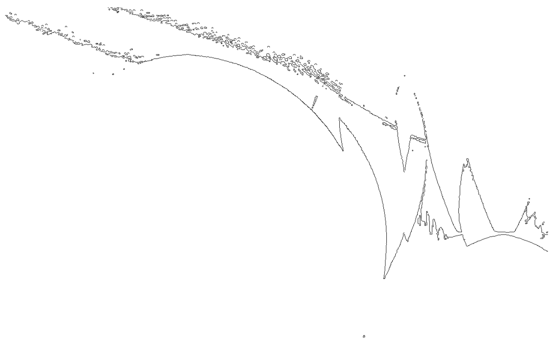

(d) edges of merged disparity layer for $d = 15$

Figure 6.2: Line drawing from disparity layers. Note how each merged disparity layer only contains pixels of one value. Hence, extracting lines from such layers produces the edges of the layer pixels instead of the desired object contours.

line drawings persist today—in children's cereal boxes, as diagrams, and as art.

Stereoscopic 3D line drawings can be drawn by hand or generated from 3D meshes using a variety of algorithms. When creating these drawings, emphasis is placed on consistency: ensuring that the object/scene visible in both views matches exactly for a comfortable viewing experience and correct depiction of depth [44, 62]. However, little work has examined how to create S3D line drawings from S3D photos [43].

While it would seem natural to apply our layer-based approach to S3D stylization, it is not appropriate for this task. Layer-based stylization divides the S3D image and disparity maps into layers by disparity such that each layer only contains pixels from a single disparity. However, unlike previous stylization algorithms, our line drawing method uses the disparity map itself for stylization, not the image. Contours, the primary type of line rendered in line drawings, may be found in the colour image. However, a myriad of other lines such as light/shadow boundaries, which are not desirable, are also present and difficult to separate from contours. Hence, we use the disparity map to isolate object contours, as it does not contain extraneous information regarding shading or texture. If we attempt to extract object contours from the disparity layers, only the edges of the layer will be found, because each disparity layer contains only pixels of the given disparity and empty pixels. Figure 6.2 illustrates this issue. Note how the edges found for the merged disparity layer do not correspond to object contours. Hence, our layer method cannot be used to produce a line drawing. We also note that applying edge detection algorithms to the left and right disparity maps, while simple, would produce inconsistent and therefore uncomfortable left and right drawings.

Our goal is tp produce stylized stereoscopic 3D line drawings from 3D photos. We also provide the option of shading our drawings to enhance the perception of depth. Shading can help resolve ambiguities in drawings where lines alone are insufficient to communicate 3D shape, as in the contour of a sphere (Figure 2.1). Our line drawings are produced with consistency in mind to ensure viewing comfort and high quality depth reproduction. The algorithm is presented in Section 6.2. Additionally, as lighting is not view-consistent in nature, we will not enforce consistency in our shading, as discussed in Section 6.3. Finally, we asked a number of stereoscopic 3D experts to evaluate the 3D comfort and depth quality of our generated images to verify their quality. A discussion of their evaluation is found in Section 6.5.

6.1 Background

A line drawing study by Cole et al. observed artists and noted where they drew lines for a variety of objects [19]. They observed that contours, creases and folds, lines that describe the shape of the object, were drawn. However, lines depicting shadows or highlights were not. This was also observed by Hertzmann et al. while rendering line drawings for smooth meshes [33].

In a stereoscopic 3D line drawing these lines give the primary sense of an object's 3D shape and depth. Without other S3D cues from which to infer depth, it is important that these lines are as consistent as possible between left and right views [5, 44]. Inconsistencies can cause viewing discomfort from binocular rivalry and double-vision, detracting from the viewer's perception of object depth. Studies have shown that these S3D lines alone sufficiently convey object shape or depth [5, 44, 51].

A number of algorithms have been proposed to produce stereoscopic 3D line drawings from meshes. Most notably, Kim et al. presented a method that produces 3D line drawings by generating contours for left and right eyes separately [44]. Contours are then pruned for view consistency by checking the visibility of points along the curve formed by creating an epipolar plane between a pair of points on the left and right contours. Kim et al. also describe a method for consistent stylization of lines by linking control points between matching contours and applying stylization to the linked pairs. However, this method can only be used with full 3D models.

Another paper by Kim et al. describes a method for producing stylized stereoscopic 3D line drawings from S3D photographs [43]. Their paper applies Canny edge detector [17] to the edge tangent field [42] of the left stereo image and warps the discovered edges to the right image using the disparity map. However, the rendered lines are from all edges that can be found in the actual image, including object contours as well as texture or pattern contours. By contrast, a hand-drawn stereoscopic 3D line drawing would be likely to include only object contours and creases. As their method is based on edge detection purely in the colour domain, they cannot differentiate between geometric discontinuities and colour discontinuities. The method could be modified to be applied to disparity maps, however, care must be taken when warping curves that overlap multiple objects.

Our goal is to construct S3D line drawings directly from S3D image pairs by harnessing the information found in the disparity map.

6.1.1 Perception and Monoscopic Depth Cues

Our perception of depth arises from both monoscopic (2D) and 3D depth cues. Monoscopic depth cues include: shading, relative size, occlusion, and motion [5, 15]. Adding shading to an S3D line drawing can improve depth perception, but the amount of improvement is limited for images with rich detail [51]. However, for S3D line drawings of simple objects with few internal lines, shading provides the necessary information about object shape. Consider Figure 6.3, is this a line drawing of a circle or a sphere?

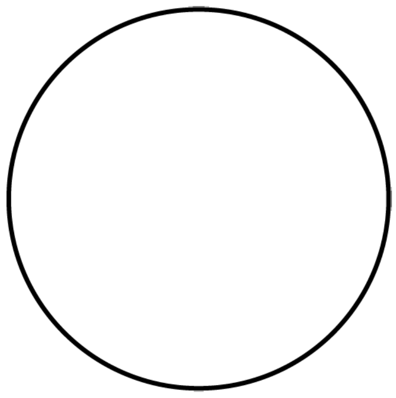


Figure 6.3: An ambiguous line drawing. Is this a drawing of a circle or a sphere? In the absence of shading, how can we tell?

Stereo-consistent shading is complicated by the fact that shading can be view-dependent. Apart from purely Lambertian surfaces, shading features such as specular highlights may appear in only one eye due to the position of the eyes with respect to the light source and object. Due to the position of the eyes with respect to the light source and object, specular highlights may be visible only in one eye [12]. This phenomenon can also be observed in S3D photographs, as demonstrated in Figure 6.4. Note that both specular highlights and reflections differ between left and right views and are circled in cyan for visibility. While these specular highlights are natural to the human visual system, they can be problematic for computer vision algorithms commonly used with stereo [12]. Additionally, it is believed in film that specular highlights can cause binocular rivalry if they are rendered inconsistently between views. Therefore, they are often removed or redrawn to be consistent [58]. We provide users the option of adding shading to our S3D line drawings. However, our shading will remain true-to-nature. That is, we will not remove or adjust the natural lighting of the S3D images.

6.1.2 Stylization

Stylization can be applied to both the S3D lines and shaded regions. Surprisingly, simple line styles such as overdrawn lines, varying thickness, and jitter do not negatively impact depth perception and comfort in S3D images [51]. However, when applying stylization to S3D lines, consistency is key. One approach would be to render stylized lines in the left view, then use the disparity map and warp them to the right. As the disparities underlying the stylized lines may overlap many

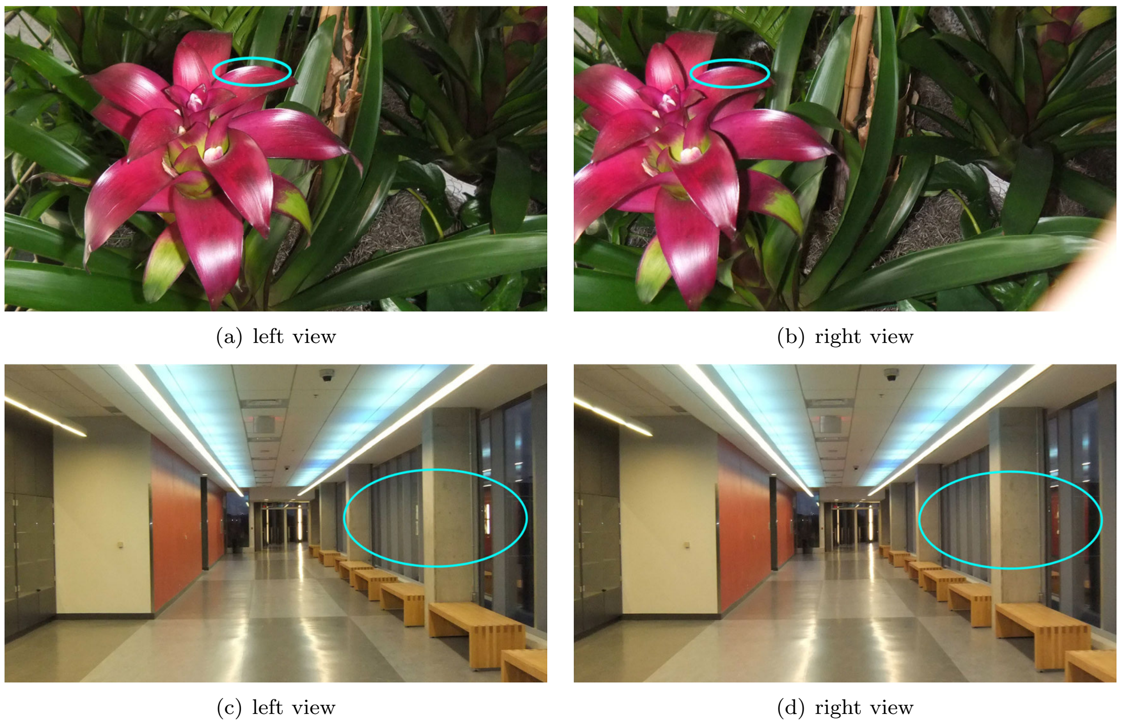

(a) left view (b) right view (c) left view (d) right view

Figure 6.4: Inconsistent specular highlights and reflections.

objects, warping individual pixels would not produce smooth lines. Alternatively, the control points of curves or the endpoints of line segments could be warped. But if any of these points from the stylized lines lie on other objects, the lines rendered in the right view may be discontinuous or distorted. Instead, we will match the control points to their underlying disparities prior to stylization or rendering, similar to the approach used by Kim et al. [44]. Although there are many ways to stylize lines, we focus on overdrawn and jittered styles.

In addition to stylized contours, we will also stylize shaded regions. Stereoscopic 3D image stylization has been well studied, though these methods focus on stylizing the whole image consistently instead of a smaller region. Stavrakis et al. applied stylization to the left image and use the disparity map to warp it to the right, then do the reverse for occluded regions [78, 79]. In the previous chapter, we described our own technique for stylization based on combinations of disparity layers [61, 62]. However, since lighting and specular highlights are view dependent these methods would enforce consistency where none exists. Hence, we will apply stylization algorithms to shaded regions in a view-dependent manner to preserve these inconsistencies. By preserving inconsistencies, we contradict Richardt et al. [70] and our previous work, which focus on establishing or maintaining consistency at all cost. However, we believe that because shading will not be used to render contours, which are essential for depth perception, binocular rivalry and randomness will have minimal effects on viewer perception.

6.2 Producing a Stereoscopic 3D Line Drawing

Our method has five stages:

1. Extract contours that depict object shape;

2. Split lines into segments by view visibility: left-only, right-only, and shared;
3. Apply line segment stylization, e.g., jitter, to control points;
4. Warp and render line segments;
5. Optional stylization or shading.

6.2.1 Extracting Contours

Contours depict the shape of an object. Interior lines, such as creases and folds, depict the interior shape of the object [32, 33]. Both contours and interior lines are needed to give a clear sense of shape [19]. These lines may be found in the disparity map using edge detection algorithms.

Extracting Lines

Contours and interior lines can be found in the disparity map using edge detection algorithms. In particular, we used the Canny edge detector as suggested by Gelautz and Markovic [26]. Strong edges, such as those depicted in Figure 6.5, are easily identified. However, certain contours and

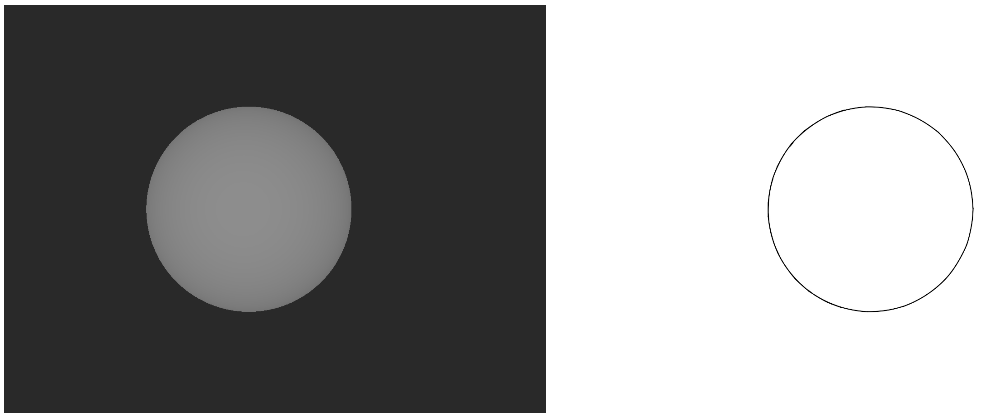

(a) left disparity map (b) edges of disparity map

Figure 6.5: The strong edges of a disparity map.

interior lines are not easily found by standard edge detection algorithms. For example, the intersection of two planar surfaces should result in an edge. Consider Figure 6.6, a raytraced image of a golden cat. The intersection of the wall and floor is distinctly visible in the image, but not in the disparity map. We attempted to isolate this edge using a variety of edge detectors and parameters. However, we found that none of a Canny edge detector, a Sobel operator, or a Laplacian were able to identify this edge without identifying other superfluous edges. Figure 6.7 illustrates several applications of the Canny edge detector using various parameters. Note that when the wall or floor edge is found, so are other "edges" that do not exist in the image. A similar problem can be seen in the crease between the cat's paws and the floor. Likewise, Figure 6.8 illustrates several applications of the Laplacian operator. Note that as kernel size increases, more edges are detected. However, when the desired edges are found, so are many superfluous edges, making it difficult to identify the desired edges. Additionally, stronger edges thicken, making them harder to place upon the foreground surface.

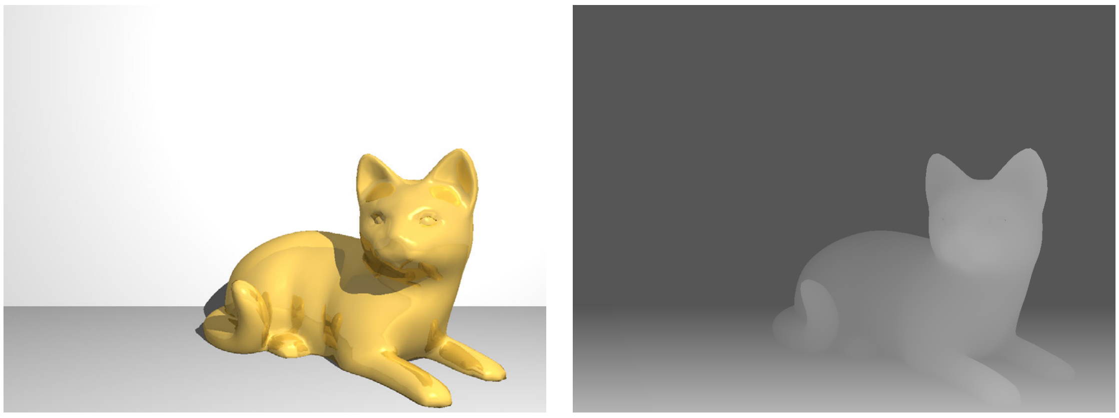

(a) left view (b) left disparity map

Figure 6.6: Left view and disparity map of a raytraced cat model.

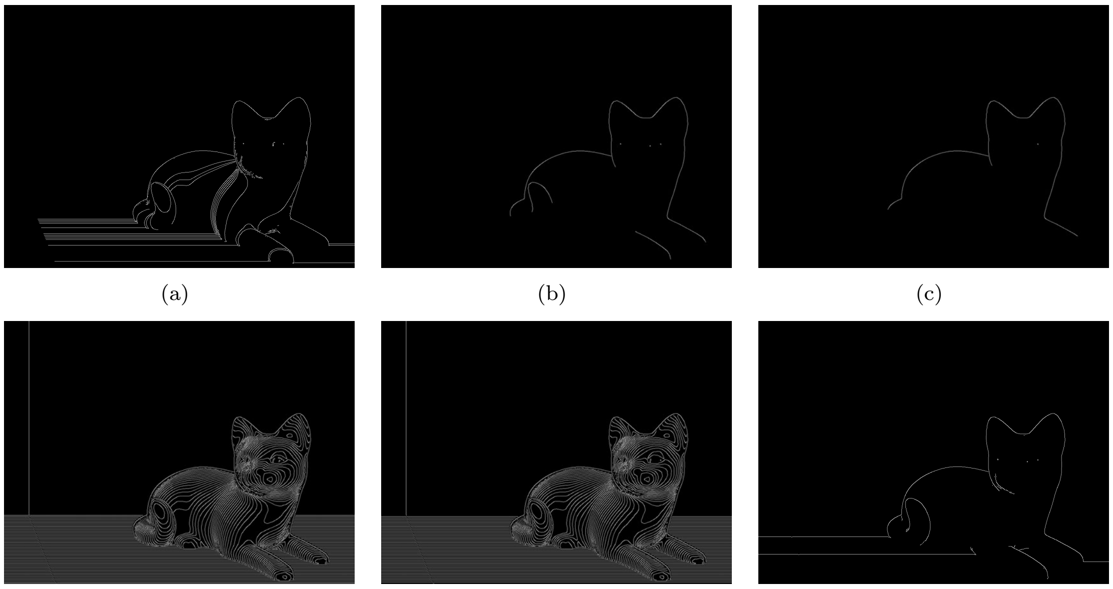

(a) (b) (c) (d) (e) (f)

Figure 6.7: Edges of the disparity map found with Canny edge detection and various parameters.

Hertzmann suggested a different method to identify these edges [33]. By casting different coloured directional lights along each of the positive and negative axes onto a 3D model, a normal map is produced. An edge operator is then applied to this shaded image. However, given that we have only a disparity pair, a 3D model must be produced first. Surface triangulation is used to produce a mesh and vertex normals from the disparity map. Assume that each position (x, y) in the disparity map has depth z equal to the disparity at that position. Triangular "faces" are formed by a point p in the disparity map and two of its immediate neighbours, q and r. A normal can then be calculated for each of these faces. To find the vertex normal of point p, compute the average of the eight adjacent triangular face normals as depicted in Figure 6.9.

Once the normal vector for each pixel is calculated, directional lights are cast, as illustrated in Figure 6.10. However, casting directional lights onto the disparity map illustrates an interesting issue. Note that dark red tones correspond to front-facing surfaces. However, instead of observing a smoothly shaded cat, the normal map appears stepped—with rings of front-facing planes. This

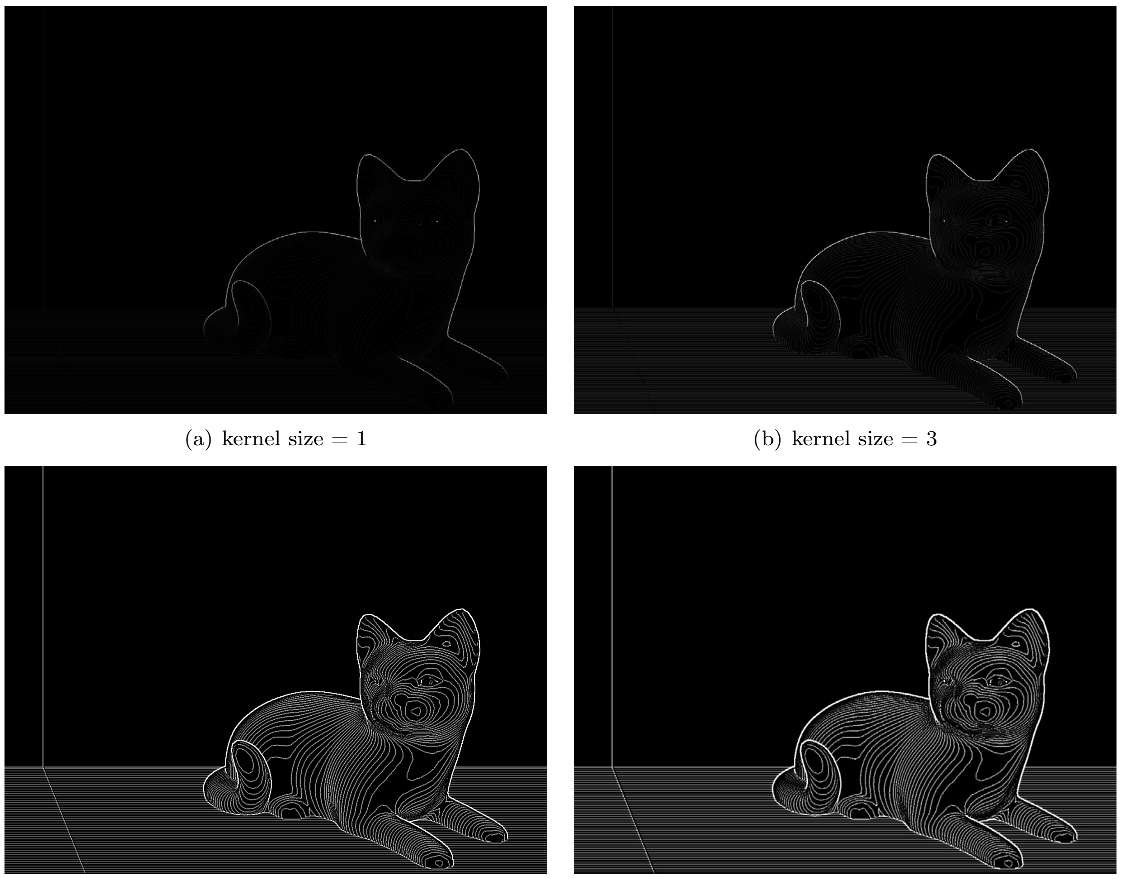

(a) kernel size = 1 (b) kernel size = 3
(c) kernel size = 5 (d) kernel size = 7

Figure 6.8: Edges of the disparity map found with the Laplacian operator and various parameters.

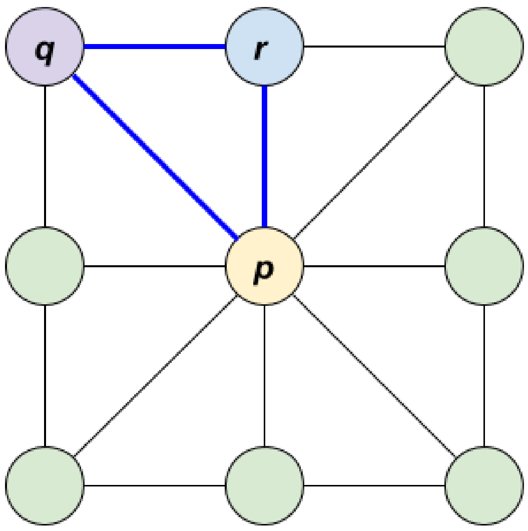
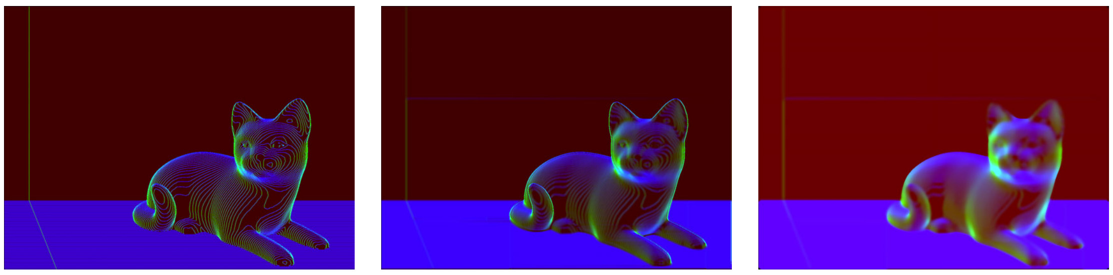


Figure 6.9: The eight immediate neighbours of point p. Note how p, q and r form a triangular face. Calculating the normal of each such adjacent face and averaging them produces the vertex normal for point p.



(a) raw disparity map directionally lit (b) smoothed disparity map directionally lit (c) applying median and bilateral filters to (b)

Figure 6.10: Directionally lit disparity map. Note that brightness is increased for (c) to improve visibility.

stepped appearance is a consequence of the limited dynamic range of disparity maps. A perfectly smooth surface cannot be depicted in this discrete space, resulting in many pixels being assigned the same integer disparity instead of their actual values. The edges of these plateaus are the same as those found by the Canny edge operator and the Laplacian in Figures 6.7 and 6.8. These artificial edges make discovery of actual edges, such as the interface between the wall and floor, difficult to achieve. For many of our test cases involving synthetic scenes a ground truth normal map is available, however, this is an exceptional case and the majority of S3D images will require one be created.

Ideally, converting the disparity map from 8-bit to floating point and applying an out-of-the-box smoothing operation would smooth these plateaus out. However, a bilateral filter may preserve or even enhance these edges and a Gaussian or box filter would soften all edges indiscriminately, effectively blending objects together.

We observed that plateaus have a small width and the plateau's disparity and the adjacent pixel's disparities differ by no more than one. That is, if the plateau has disparity d, then the left edge of the plateau has disparity d_l, and the right edge has disparity d_r, then $|d_l - d| \leq 1$ and $|d_r - d| \leq 1$. The same is true for the top and bottom edges of the plateau. To smooth these plateaus and generate a smooth lit disparity:

1. Linearly interpolate from plateau edges horizontally (H) and vertically (V).
2. Apply a "selective box" filter to smooth and combine H and V. For each pixel p with original disparity d and a box radius of b_r, the new disparity d' is the average of all pixels within b_r of p from both H and V for which the disparity is within some tolerance τ of d.
3. Calculate the vertex normals.
4. Cast directional lights to colourize and produce the normal map. Note the directionally lit disparity map is significantly smoothed, as seen in Figure 6.10(b).
5. Apply a median filter followed by a strong bilateral filter to smooth remaining noise without destroying object edges.
6. Optionally increase the contrast of the smoothed image to place emphasis on object boundaries. The final result can be seen in Figure 6.10(c).

After smoothing the disparity map and casting directional lights on it the final edge pass can be performed. Applying the Canny edge operator to this image now finds the missing object edges, as seen in Figure 6.11. Since many S3D images do not have these types of edges, this second phase of edge detection is optional.

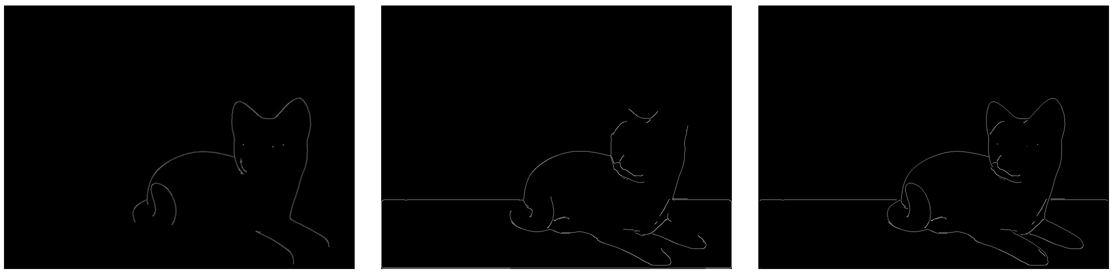

(a) edges from raw disparity map (b) edges from smoothed disparity map (c) combined edges

Figure 6.11: Edges from the raw and smoothed disparity map.

Splitting Lines for Consistency

After extracting edges from the left and right disparity maps, the edges are rendered as smooth splines for the corresponding view. However, this view dependent rendering can produce inconsistencies, as seen in Figure 6.12. These inconsistencies appear in the regions of the S3D image visible in both views. Inconsistencies arise when edge detection algorithms find an edge in one disparity map, but not the other due to the slight variation in the viewpoint. Note how the cat's face in Figure 6.12 is drawn with more lines in the right view. Additionally, the cat's snout is rendered with a single line in the left view, but two in the right view.

To prevent these inconsistencies, we select the left view to be the "true" edges. Then, any line in the left view that is visible in the right is warped using the disparity map to the right view. Lines that are occluded in one view, but not the other, are rendered only in the non-occluded view. Specifically, lines only visible in the left are rendered only in the left, likewise, lines only visible in the right are rendered only in the right.

There are two approaches to warping lines. First, render the lines in the left view then use the disparity map to warp the left view to the right. However, as demonstrated in Figure 6.13, warping after rendering produces a fragmented appearance. This is because pixels of the rendered line may lie on other surfaces with different disparities. Instead, we warp the control points of the lines, and render them in each view separately. However, this may introduce subtle inconsistencies, so we use a large number of control points to reduce them.

Finally, we note that long edges extracted from the disparity map may span multiple objects and both occluded and visible regions. Warping the entire stroke can result in partially occluded edges being visible in the wrong view. To prevent this we split strokes whenever the visibility of adjacent control points change, for example, from visible to occluded. We also split strokes when the disparity of adjacent control points differs by more than some threshold τ_d. Strokes that cannot be warped, because they are only visible in one view, are only rendered in one view.

Consistent Control Point Stylization

Monoscopic and S3D line drawings are often stylized and represent objects using rough, overdrawn and jittery lines, as seen in Figure 6.14. We also offer users the option of stylizing S3D lines with overdrawn or jittered styles. Stylizing lines has an advantage, besides producing a more

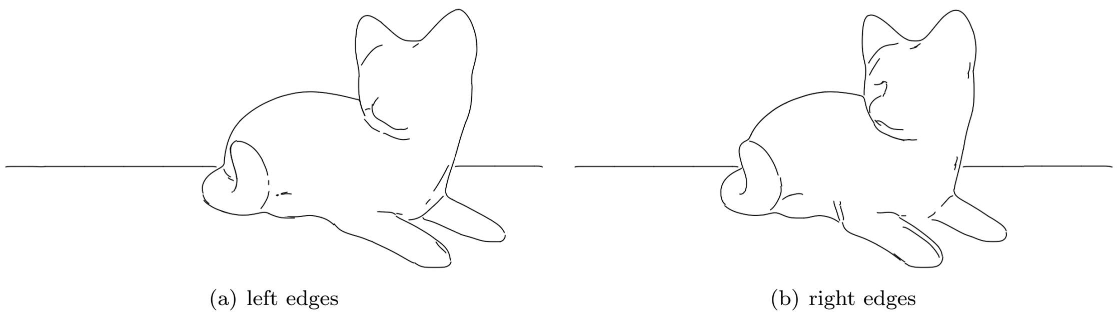

(a) left edges (b) right edges

Figure 6.12: Edges of the left and right views rendered independently.

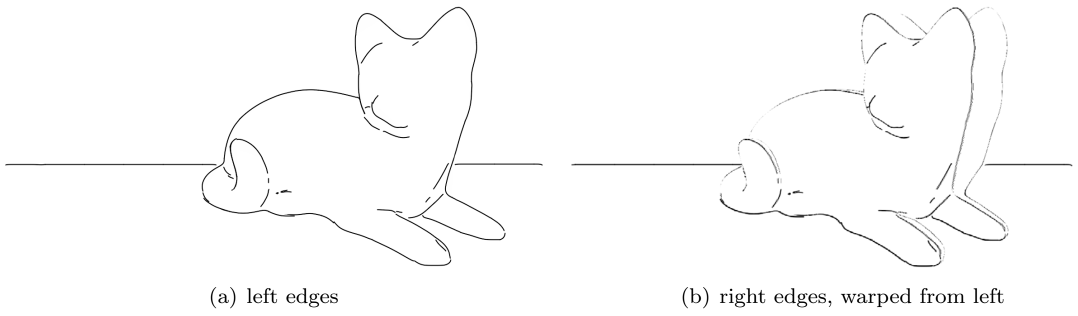

(a) left edges (b) right edges, warped from left

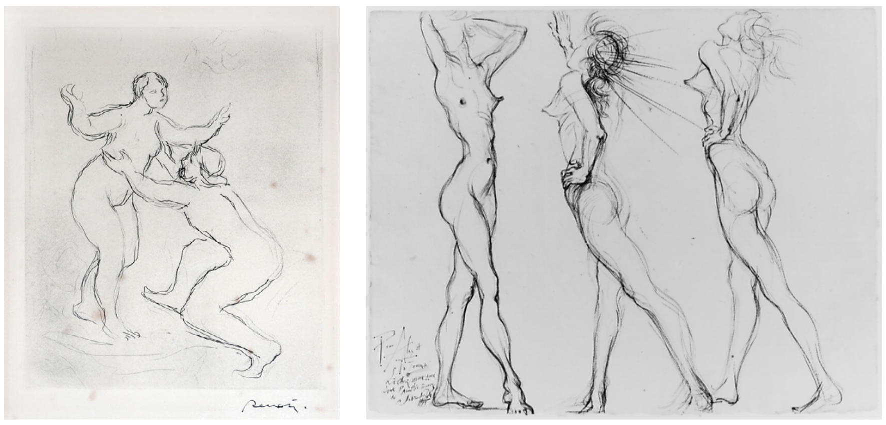
Figure 6.13: Warping strokes from left-to-right after rendering strokes, similar to the stylization approach used by Stavrakis and Gelautz [79].

aesthetically pleasing drawing. Disparity maps produced by stereo matching algorithms often contain noise and inaccuracies, as seen in Figure 2.23. Note that the edges of the rose are distorted and that there are streaks of noise on the left side. These artifacts negatively affect our ability to produce a recognizable S3D line drawing of a scene by introducing distortions to the warped splines. By applying stylization, subtle errors in the disparity map will be less noticeable.

Kim et al. discussed a method for stylizing stereoscopic 3D lines. Their method performs stylization after lines have been discovered for both left and right images. Specifically, it links line segments in the left view to the matching segments in the right and consistently renders texture to these linked and parameterized curves [44].

Our approach stylizes lines prior to warping by replicating and transforming control points. To stylize lines we assume that each line is stored as a list of control points, then:

1. Duplicate each line a fixed number of times, to create overdrawn lines;
2. Transform the control points of each line by:
    (a) Adding a small random translation vector to each point for a jittered appearance;



(a) Le Fleuve Scamandre. Pierre-August Renoir (b) Three Nudes. Salvadore Dali

Figure 6.14: Rough ink drawings and etchings.

(b) Scaling each control point by a small factor with respect to the center of the image, to produce overdrawn lines;

Note that prior to altering the control points of a line it is important to store the original, pre-transformed disparities of those control points, so that they can be correctly warped after stylization.

6.3 Monocular Depth Cues

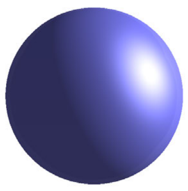
S3D line drawings depict the shapes of objects. However, these lines do not convey information such as surface texture or roundness—but shading and highlights do. Object shading and shadows are monocular depth cues [29]. Shading, in particular, specular highlights are view dependent [82]. Adding monocular depth cues to S3D line drawings can improve understanding of surface shape, improve depth perception. Also, because lighting is view dependent, the left and right views will be stylized independently to preserve their separate lighting characteristics.

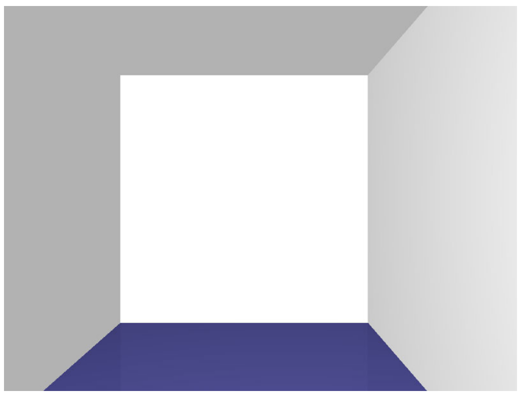
To produce the stylized lighting, the left and right input images are converted to grayscale and stylized using a variety of algorithms. While any stylization algorithm or filter could be used, we chose those that did not explicitly render contours, as our method will produce those separately. Finally, the stylized shading and S3D lines are combined to produce the final image.

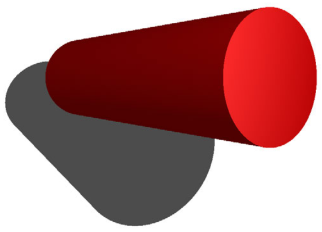
6.4 Results

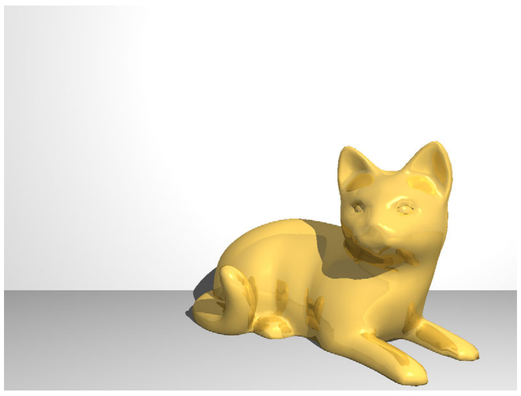
We tested our method on a number of stereoscopic 3D images ranging from a single object to 3D photographs; several of these inputs are shown in Figure 6.15. All of the images used as input to our method have near-perfect disparity maps, originating from structured light or raytracing. This helps avoid disparity inconsistencies and errors impacting our evaluation of the line drawing algorithm.

Figure 6.16 illustrates the 2D line drawings produced by our method using the left views. Note that since lines are generated from the disparity map, contours and interior lines are the only lines


(a) raytraced sphere


(b) raytraced box interior


(c) raytraced cylinder in front of wall


(d) raytraced cat model

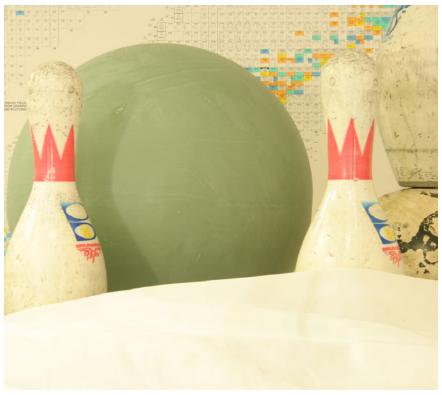

(e) stereoscopic 3D photo courtesy of [73]

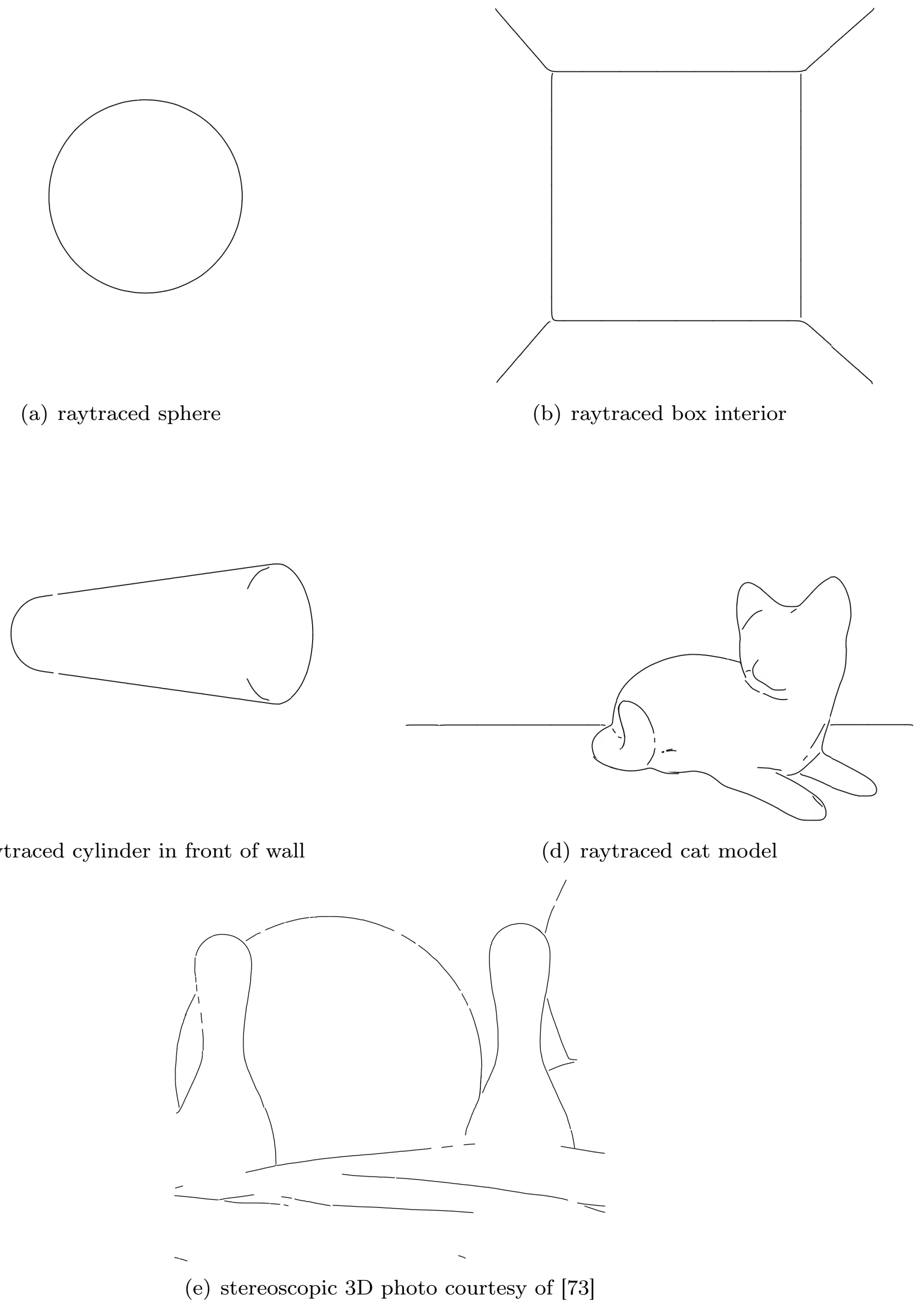
Figure 6.15: Sample inputs to S3D line drawing algorithm.

visible. Also, while many lines are broken, our goal is to produce stylized line drawings instead of accurate ones.

Extending these line drawings to 3D produced the anaglyphs displayed in Figure 6.17. Note that left and right views are mostly consistent, with small noted exceptions. For example, the sphere has a slight inconsistent distortion on the left side.

Stylizing the S3D lines yielded the images depicted in Figure 6.18. Note that even with jittered and overdrawn lines, the left and right views remain consistent.

We used three types of stylization for shading: toon shading, fine dots/lines, and halftoning with large particles. None of these stylizations render contours, so there is no overlap between


(a) raytraced sphere (b) raytraced box interior

(c) raytraced cylinder in front of wall (d) raytraced cat model

(e) stereoscopic 3D photo courtesy of [73]

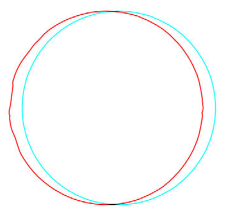
Figure 6.16: Line drawings.

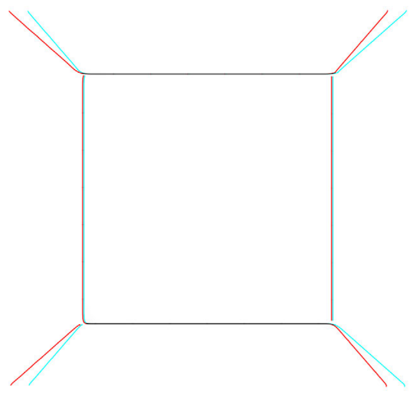
shading and the line drawing. Figure 6.19 illustrates each of these styles.

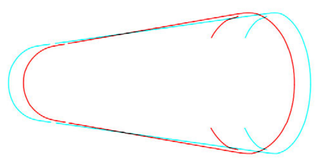
We combined the stylized S3D line drawings with stylized shading to produce our final images, shown in Figure 6.20. To ensure the visibility of the S3D lines, the intensity of stylized shading was decreased by 50%.

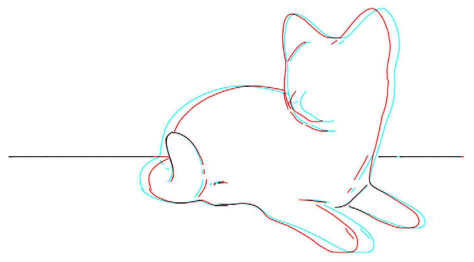
6.5 Evaluation

Instead of conducting a general user study to evaluate the quality of our results, we asked S3D experts with a mix of industry and academic experience to perform a blind evaluation of our results.


(a) raytraced sphere


(b) raytraced box interior


(c) raytraced cylinder in front of wall


(d) raytraced cat model

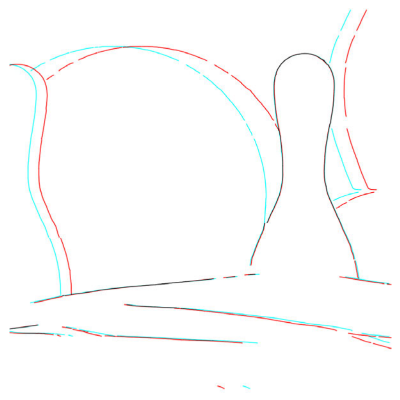
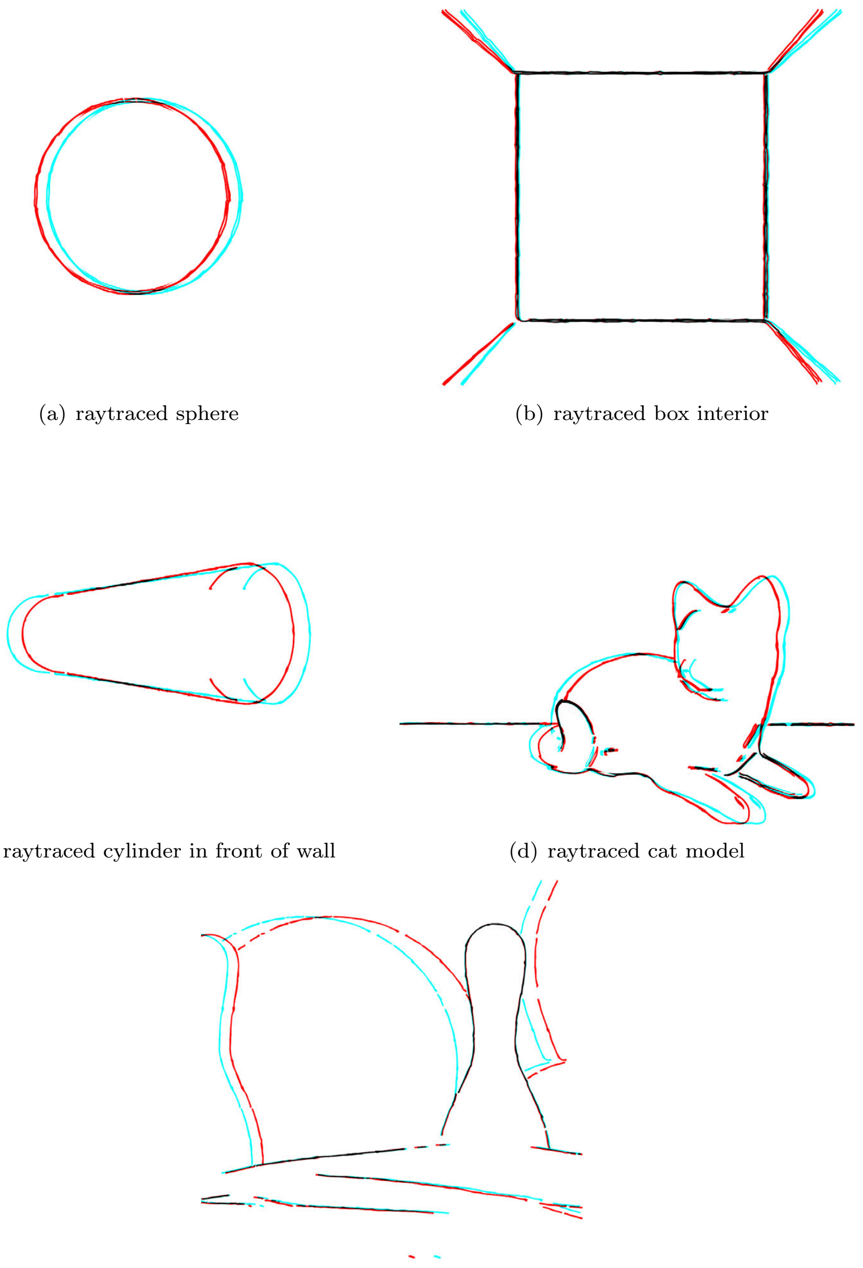

(e) stereoscopic 3D photo courtesy of The Middlebury Dataset [73]

Figure 6.17: S3D line drawings.

Each of our four experts were each given a unique package of results, varying in complexity and stylization and asked questions about viewing comfort and depth quality. They were not told anything about how each of the images was produced.

The expert evaluators agreed that our method produces line drawings that are more comfortable than the view dependent line drawing approach. They also agreed that the line drawings reproduced a sense of depth and that adding 2D depth cues (shading) improved the perception of depth. The majority felt that overdrawn lines were unnatural because these lines formed a thick, ribbon-like outline resulting in perceived blurred or thick edges. However, they noted that it was not uncomfortable. Several experts found the large-particle shading (halftoning) difficult to view,


(a) raytraced sphere (b) raytraced box interior

(c) raytraced cylinder in front of wall (d) raytraced cat model

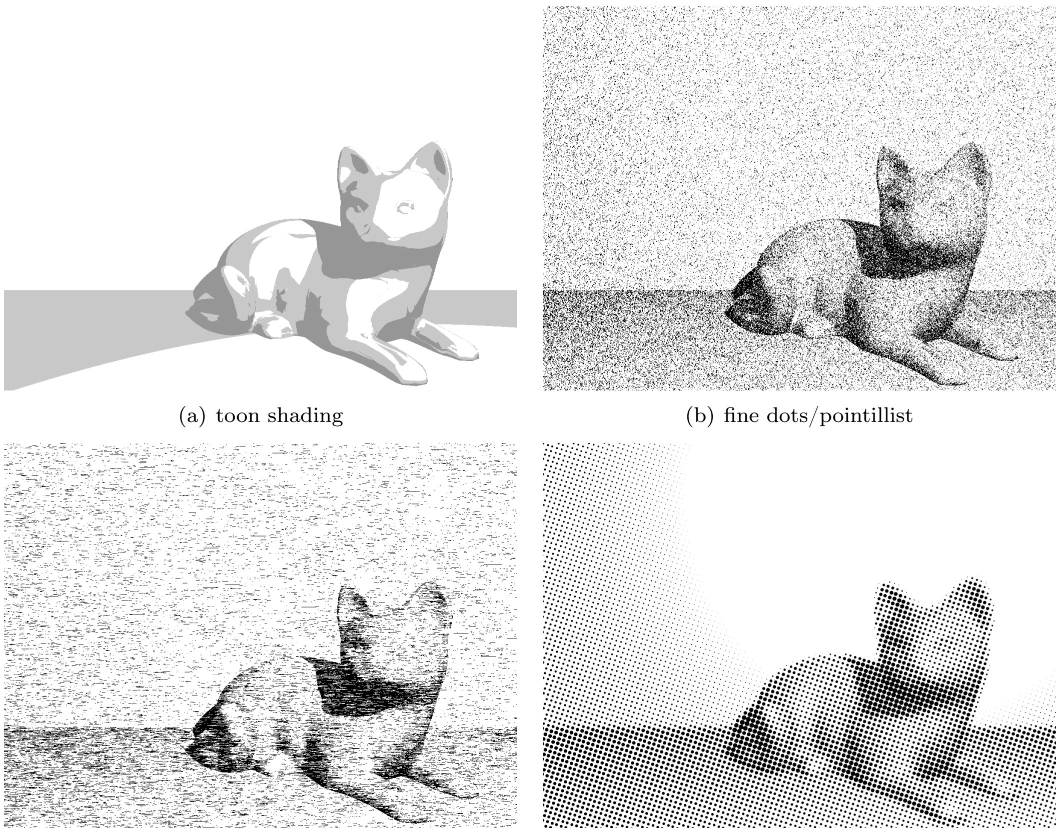
(e) stereoscopic 3D photo courtesy of [73]

Figure 6.18: S3D line drawings.

but were not able to articulate the reason for the discomfort. We anticipated that our halftoning results would be difficult to view because the view dependent rendering of larger particles would be more noticeable. Experts did not comment on stylization filters applied with smaller particles.

6.6 Conclusion

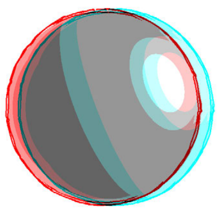
Our algorithm successfully produces stylized stereoscopic 3D line drawings from photographs. These line drawings reproduce 3D shape, especially when combined with monoscopic shading. Futhermore, for fine-grained stylizations, inconsistent shading did not have a negative impact on

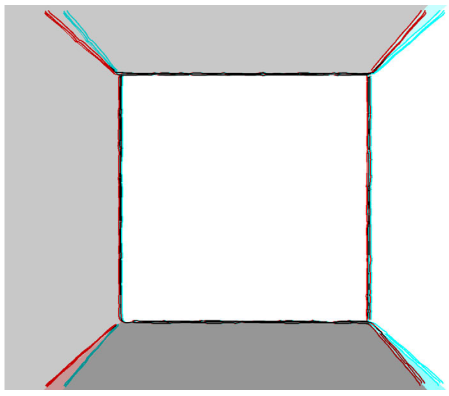

(a) toon shading (b) fine dots/pointillist

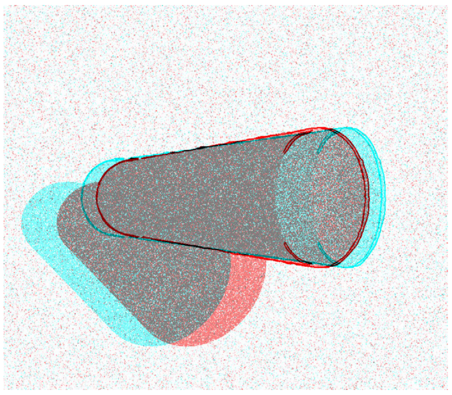
(c) fine, short lines (d) halftone

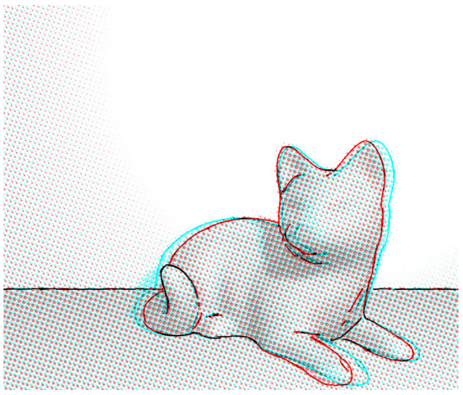
Figure 6.19: Types of shading stylization.

the perception of depth.


(a) raytraced sphere


(b) raytraced box interior


(c) raytraced cylinder in front of wall


(d) raytraced cat model

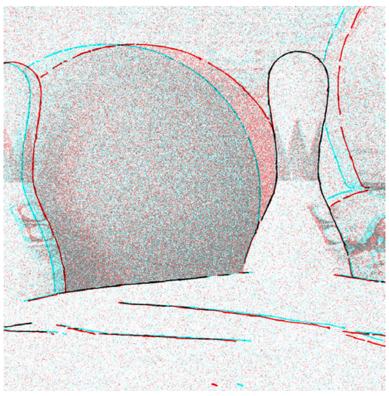

(e) stereoscopic 3D photo courtesy of [73]

Figure 6.20: Final stylized S3D line drawings.

Chapter 7

Conclusions

Working with S3D media poses a number of challenges. Capturing S3D introduces errors, from uncomfortable parallax to inconsistencies arising from misaligned lenses/capture surfaces or subtle lens differences. While some of these issues can be handled in simple image processing tools, others, such as uncomfortable parallax, cannot. Once images have been corrected, additional stylization can be applied, which poses further challenges since maintaining consistency between the left and right views is key for visual comfort.

Presently, much of these S3D correction and stylization processes are conducted manually, making them costly. Additionally, in order to perform many of these tasks a disparity map is required. Disparity map generation is an active area of research in S3D, and a notoriously difficult task. Even when a disparity map is provided, it often requires manual correction.

We considered several of these issues—adjusting disparity, filling previously occluded regions, applying stylization consistently, and S3D line drawing. Our disparity adjustment method decomposed the S3D image into layers by disparity, adjusted parallax, then composited the layers back together into the adjusted views. One side effect of our approach is that regardless of any inconsistencies in the input left and right views, our disparity-adjusted output is consistent.

Our layered approach can also be used to apply stylization to S3D images consistently, with a few minor modifications. In particular, for disparity adjustment, we produced one layer for each unique disparity value. For stylization, on the other hand, such layers often contain too few pixels to apply stylization filters. We therefore combine adjacent layers to produce a sufficient surfaces to stylize. The stylized layers are composited to produce the final, consistent, stylized image.

Finally we address the production of stylized S3D line drawings from photos, a style that our layered approach cannot reproduce. Our method identifies the object boundaries from a modified disparity map, applies stylization and warps the control points, rendering the lines in each view. To improve our perception of depth and object shape, stylized shading is added; but unlike our previous work, this is done in a view independent way to preserve the inconsistent nature of shading.

While our methods are not fully automated, they give the user the ability to specify how the S3D media should be manipulated—for example, what stylization to apply, or by how much disparity should be reduced. Our methods were verified by mathematical evaluations, user studies, experts, and by real-world use in an S3D documentary.

Chapter 8

Future Work

The research presented in this thesis suggests a large number of potential future projects.

8.1 Increasing Disparity

While our disparity adjustment process can easily decrease disparity, increasing it poses a unique challenge. In particular, if we want to double the disparity, sending each disparity d to $2d$, then the final disparity map will contain no pixels with odd disparities. If we quadruple the disparity, sending each disparity d to $4d$, the final map would contain only disparities divisible by four. This creates a stepped appearance where the larger the multiplier, the greater the distance, along z, between originally adjacent disparity layers. Smooth surfaces such as a sphere will lose their smoothness and appear as flat layers. One way to reduce this stepped appearance is to subdivide each disparity layer into several through erosion. Each sub-layer is then assigned a new, real-valued disparity that is an interpolated value between the neighbouring low and high disparities. However, care must be taken when handling disparity layers that have single-pixel diameter.

This problem is somewhat similar to the problem of super-resolution in regular 2D image processing. However, an additional challenge arises producing new pixel values due to the low dynamic range of the disparity map. In particular, unlike with colour images, simple interpolation is insufficient to produce intermediate values.

8.2 Film Grain

One of the interesting visual properties of the traditional film process is its grain. Grain is a texture apparent in the developed image resulting from silver grains or colour dyes deposited on the media surface [1]. Many directors and photographers consider grain to be desirable and part of the artistic expression. Newer digital capture technologies also exhibit grain, but it is significantly less noticeable. Hence, some directors choose to add artificial film grain or enhance existing grain in media captured by digital cameras, as seen in Figure 8.1. However, it is far from clear how to interpret the aesthetic of film grain in the context of S3D media. If the film grain in left and right views differs, then it introduces inconsistencies that may add discomfort. If film grain is consistent, then it has a specific depth. Grain may be in the absolute foreground, where it appears as though we are looking through a dirty shower door, or placed on the screen plane (i.e., the plane of zero disparity) and grain appears as a dirty window at an unexpected place. If film grain is rendered on

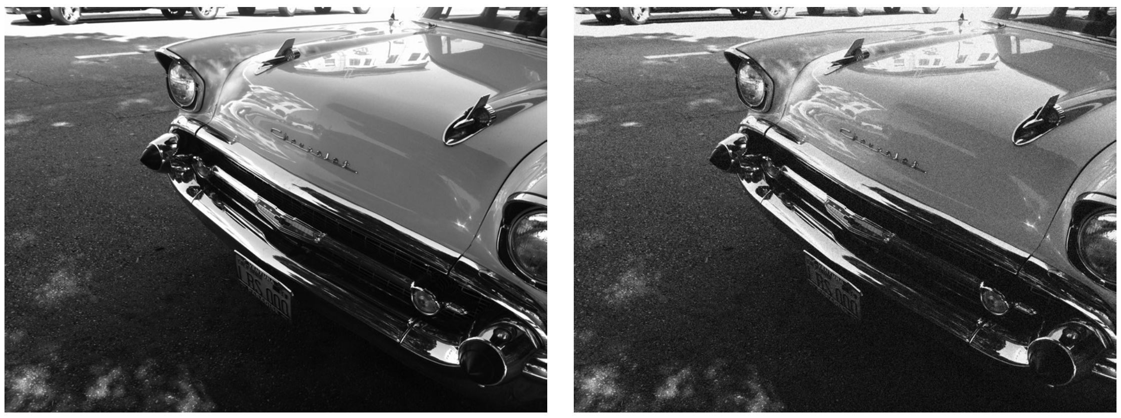

(a) original image (b) original image with added grain

Figure 8.1: Adding grain for aesthetic appeal.

object surfaces, however, this gives the scene objects the appearance of being dusty. Yet another approach is to disperse grain randomly through 3D space; however, this gives the appearance of the air itself being dusty. The question is, does film grain belong in S3D? If so, what is the most logical and comfortable way to introduce it? If film grain does not belong in S3D, i.e., there is no logical or comfortable way to add it, how can we convince directors to abandon it?

8.3 Lens Flare

A lens flare is an artifact of the optical elements of the camera lens, and has no implied location in the 3D scene. However, similar to film grain, many directors like to add lens flare as part of their artistic vision. Similar to film grain, it is not clear what role a lens flare plays in S3D. How should the flare be rendered? At a particular depth? Through a range of depths?

8.4 Stylization with Structures

Many non-photorealistic filters stylize images using structures like large dots or tiles. Our layer-based approach can produce consistent stylization for these styles. However, by using either of the methods described in Chapter 5, structures will either be trimmed or overlapping, destroying any aesthetic created by the spacing between them. However, stereo warping, as used by others, does not solve this problem either — the warping process can distort the structures and the spacing between them. Is there a way to apply these stylizations to S3D without distorting/obstructing the structures or spacing? Or, are there certain stylizations that simply do not translate to S3D?

8.5 Disparity Adjustment of Video

Our current method for disparity adjustment has been applied to stills, but not video. The layer-based adjustment process of Chapter 3 translates to video easily, simply by applying the method to each frame separately. However, the disparity map inpainting and filling of the previously occluded regions could be improved for use with video. In particular, with a sequence of video frames it is possible that nearby frames contain some of the data that is missing, for either the image or the disparity map, and this can be used to produce more accurate fills [18, 39]. When adjusting video we must also consider that our fills, at least the disparity fills, need to be temporally coherent because the depth of a still object should not change significantly from frame to frame.

8.6 Stylizing Video

Our S3D stylization algorithms focus on applying filters to still images instead of video. Like our disparity adjustment method, the layer based approach translates naturally to video, but the stylizations themselves generally do not. In particular, when each frame is stylized independently of the others, consecutive frames may exhibit flashes of colour or shapes. While this has been used for artistic effect in many films, including the popular Christmas short “The Snowman”, not every type of stylization produces a pleasing effect when used in this manner. Most videos aim for some form of temporal coherence. It would be interesting to adapt our layered approach to support temporally coherent stylization.

8.7 Inpainting

Our proof-of-concept approach to the general problem of inpainting uses machine learning to attempt to identify from the hole and neighbourhood which inpainting algorithm will produce the most convincing results. However, our best experiment only produced a convincing fill 82.4% of the time, which is about 1.5% less than applying textural inpainting algorithm PatchMatch for every case. We believe our results can be improved by trying to understand why participants marked all inpainting algorithms as producing unconvincing fills for certain images and by adding feature recognition for humans and animals. Additionally, we believe these results can be significantly improved by collecting more sample inputs and participant responses.

8.8 Edge Consistency in S3D Line Drawing

Our existing S3D line drawing algorithm does not render perfectly consistent lines because errors in the disparity map may produce unexpected warps of control points, and because we warp the control points of the splines instead of the rendered splines themselves. Since the splines are drawn independently in each view, they are not guaranteed to be consistent. Using the disparities of the control points, we would like to investigate adjusting the underlying disparity map so that the rendered splines can be correctly warped to solve this issue.

8.9 Disparity Map Cleanup

Our patented disparity map cleanup tool uses statistics and segmentation to find pixels of the disparity map that are likely incorrect, then interpolates new disparities for those pixels from known correct values. However, this method fails when there are few known correct pixels in a segment. In such cases, we could consider using plane or shape fitting approaches to replace the incorrect disparities. Another approach may be to do an image-wide search for similarly coloured regions—these regions may have similar disparities to the currently-under-repair one, and might be useful in providing additional “correct” values to interpolate from, or just to identify which pixels are likely incorrect. It is possible that deep learning methods may be applied to identify likely correct and incorrect pixels in the disparity map.

8.10 Consistent Stylization User Study

We conducted a user study to assess the comfort of our consistent stylization method in Chapter 5. The results are presented in Figure 5.21. This experiment was conducted in 2011, and over the course of the work presented in this thesis we have realized our original methodology was lacking. In particular, while we required that participants had stereo vision, we did not test the stereo acuity of our participants. We noted that participant responses varied greatly for a few images—a couple of participants noted that these images were very comfortable while the majority found them very uncomfortable. This variance in response may have been due to the presumed stereo acuity of participants. We also did not conduct a rigorous statistical analysis of the collected data. In the future, it would be interesting to repeat this experiment and test stereo acuity prior to accepting participants into the study. Additionally, there may be value in exploring the use of quantitative metrics, such as those presented by Richardt et al., for viewer comfort as part of our methodology for evaluation [70].

Bibliography

1. Ascher, S. & Pincus, E. *The Filmmaker's Handbook: A Comprehensive Guide for the Digital Age* (Plume, 2007).
2. Asente, P., Cohen, S. D., Price, B. L. & Northam, L. *Methods and apparatus for correcting disparity maps using statistical analysis on local neighborhoods* 2014.
3. Bailey, J. *"Jurassic Park": How a 2D movie becomes a 3D move* `http://flavorwire.com/382554/jurassic-park-how-a-2d-movie-becomes-a-3d-movie`.
4. Banks, M. S., Read, J. R., Allison, R. S. & Watt, S. J. *Stereoscopy and the Human Visual System* in *SMPTE 2nd Annual International Conference on Stereoscopic 3D for Media and Entertainment* (2011), 2–31. doi:`10.5594/M001418`.
5. Banks, M. S., Read, J. C. A., Allison, R. S. & Watt, S. J. Stereoscopy and the Human Visual System. *SMPTE Motion Imaging Journal* **121,** 24–43 (2012).
6. Barnes, C., Shechtman, E., Finkelstein, A. & Goldman, D. B. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Transactions on Graphics* **28,** 24:1–24:11 (July 2009).
7. Baumgartner, R. *3D Postproduction: Stereoscopic Workflows and Techniques* (Taylor & Francis, 2014).
8. Bay, H., Ess, A., Tuytelaars, T. & Van Gool, L. Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst.* **110,** 346–359. ISSN: 1077-3142 (June 2008).
9. Benoit, A., Le Callet, P., Campisi, P. & Cousseau, R. *Quality Assesment of Stereoscopic Images* in *EURASIP Journal on Image and Video Processing* (2008), 1–13.
10. Bertalmio, M., Bertozzi, A. & Sapiro, G. *Navier-stokes, fluid dynamics, and image and video inpainting* in *Computer Vision and Pattern Recognition* **1** (2001).
11. Bertalmio, M., Vese, L., Sapiro, G. & Osher, S. Simultaneous Structure and Texture Image Inpainting. *Trans. Img. Proc.* **12,** 882–889. ISSN: 1057-7149 (Aug. 2003).
12. Bhat, D. N. & Nayar, S. K. Stereo and Specular Reflection. *Int. J. Comput. Vision* **26,** 91–106. ISSN: 0920-5691 (Feb. 1998).
13. Breese, B. B. Binocular rivalry. *Psychological Review* **16,** 410–415.
14. Brenner, E. & Smeets, J. B. J. *Depth Perception: Sensation, Perception, and Attention* ISBN: 9781119170174 (Wiley, 2018).
15. Brooks, K. R. Depth Perception and the History of Three-Dimensional Art: Who Produced the First Stereoscopic Images? *i-Perception* **8,** 2041669516680114 (2017).

16. Cai, N. *et al.* Blind inpainting using the fully convolutional neural network. *The Visual Computer* **33,** 249–261 (Feb. 2017).

17. Canny, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-8,** 679–698 (Nov. 1986).

18. Chang, R.-C., Lin, L. H., Tian, C.-T. & Shih, T. K. *Video Inpainting and Restoration Techniques* in *Proceedings of the 13th Annual ACM International Conference on Multimedia* (ACM, New York, NY, USA, 2005), 804–805.

19. Cole, F. *et al. Where Do People Draw Lines?* in *ACM SIGGRAPH 2008 Papers* (2008).

20. Criminisi, A., Perez, P. & Toyama, K. Region Filling and Object Removal by Exemplar-based Image Inpainting. *Trans. Img. Proc.* **13** (Sept. 2004).

21. Ding, J. & Levi, D. M. *Recovery of stereopsis through perceptual learning in human adults with abnormal binocular vision* in *Proceedings of the National Academy of Sciences* **108** (2011), E733–E741.

22. Efros, A. A. & Leung, T. K. *Texture Synthesis by Non-Parametric Sampling* in *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2* (1999).

23. Frantc, V. A. *et al. Machine learning approach for objective inpainting quality assessment* in *Proc. SPIE* **9120** (2014), 9120 –9120 –9.

24. Freeland, C. On Being Stereoblind in an Era of 3D Movies. *Essays in Philosophy* **13** (2 2012).

25. Furihata, H., Yendo, T., Panahpour Tehrani, M., Fujii, T. & Tanimoto, M. *Novel view synthesis with residual error feedback for FTV* in *Proc. SPIE* **7524** (2010), 75240K–75240K–12.

26. Gelautz, M. & Markovic, D. *Recognition of Object Contours from Stereo Images: an Edge Combination Approach* in *3D Data Processing, Visualization and Transmission* (2004).

27. Glassner, A. *Deep Learning, Vol. 1: From Basics to Practice* (Amazon Digital Services, 2018).

28. Glassner, A. *Deep Learning, Vol. 2: From Basics to Practice* (Amazon Digital Services, 2018).

29. Goldstein, E. *Sensation and Perception* (Cengage Learning, 2009).

30. Hays, J. & Essa, I. A. *Image and video based painterly animation.* in *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering* (2004), 113–120.

31. Hertzmann, A. *Painterly rendering with curved brush strokes of multiple sizes* in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998).

32. Hertzmann, A. Introduction to 3D Non-Photorealistic Rendering: Silhouettes and Outlines. *SIGGRAPH Course Notes* (Jan. 1999).

33. Hertzmann, A. & Zorin, D. *Illustrating Smooth Surfaces* in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (2000).

34. Hesabi, S., Jamzad, M. & Mahdavi-Amiri, N. *Structure and texture image inpainting* in *2010 International Conference on Signal and Image Processing* (Dec. 2010), 119–124.

35. Hirschmuller, H. Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **30,** 328–341 (Feb. 2008).

36. Hocking, R., MacKenzie, R. & Bibiane Schönlieb, C. Guidefill: GPU Accelerated, Artist Guided Geometric Inpainting for 3D Conversion. *CoRR* **abs/1611.05319** (2016).

37. Huang, X., Yuan, C. & Zhang, J. *Graph Cuts Stereo Matching Based on Patch-Match and Ground Control Points Constraint* in *Advances in Multimedia Information Processing – PCM 2015* (eds Ho, Y.-S., Sang, J., Ro, Y. M., Kim, J. & Wu, F.) (Springer International Publishing, Cham, 2015), 14–23.

38. Istead, L., Asente, P. & Kaplan, C. S. *Layer-based Disparity Adjustment in Stereoscopic 3D Media* in *Proceedings of the 13th European Conference on Visual Media Production (CVMP 2016)* (2016).

39. Jia, C. *Video Inpainting Algorithm Based on Texture Synthesis* in *2016 International Conference on Smart Grid and Electrical Automation (ICSGEA)* (Aug. 2016), 201–204.

40. Jones, G. R., Lee, D., Holliman, N. S. & Ezra, D. *Controlling perceived depth in stereoscopic images* in. **4297** (2001), 42–53.

41. Jung, Y. J., Sohn, H., Lee, S.-I., Park, H. W. & Ro, Y. M. Predicting Visual Discomfort of Stereoscopic Images Using Human Attention Model. *IEEE Trans. Cir. and Sys. for Video Technol.* **23,** 2077–2082. ISSN: 1051-8215 (Dec. 2013).

42. Kang, H., Lee, S. & Chui, C. K. *Coherent Line Drawing* in *Proceedings of the 5th International Symposium on Non-photorealistic Animation and Rendering* (2007).

43. Kim, Y.-S., Kwon, J.-Y. & Lee, I.-K. *Stereoscopic Line Drawing Using Depth Maps* in *ACM SIGGRAPH 2012 Posters* (2012).

44. Kim, Y., Lee, Y., Kang, H. & Lee, S. Stereoscopic 3D Line Drawing. *ACM Transactions on Graphics* **32** (July 2013).

45. Kitzen, Dir. J. *Soldiers' Stories* `http://www.kallistimedia.com/2013/10/05/soldiers-stories/` (Kallisti Media, 2014).

46. Kolmogorov, V. & Zabih, R. *Multi-camera Scene Reconstruction via Graph Cuts* in *in European Conference on Computer Vision* (2002), 82–96.

47. Kooi, F. L. & Toet, A. Visual comfort of binocular and 3D displays. *Displays* **25,** 99 –108 (2004).

48. Koziarski, M. & Cyganek, B. in *Computer Vision and Graphics: International Conference, ICCVG 2016, Warsaw, Poland, September 19-21, 2016, Proceedings* (eds Chmielewski, L. J., Datta, A., Kozera, R. & Wojciechowski, K.) 163–173 (Springer International Publishing, Cham, 2016).

49. Lambooij, M. T. M., Jsselsteijn, W. A., Fortuin, M. & Keynderickx, I. Visual Discomfort and Visual Fatigue of Stereoscopic Displays: A Review. *Journal of Imaging Science and Technology* **53** (2009).

50. Lang, M. *et al. Nonlinear Disparity Mapping for Stereoscopic 3D* in *ACM SIGGRAPH 2010 Papers* (ACM, New York, NY, USA, 2010), 75:1–75:10.

51. Lee, Y., Kim, Y., Kang, H. & Lee, S. *Binocular Depth Perception of Stereoscopic 3D Line Drawings* in *Proceedings of the ACM Symposium on Applied Perception* (2013).

52. Litwinowicz, P. *Processing images and video for an impressionist effect* in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997).

53. Liu, F., Niu, Y. & Jin, H. *Keystone correction for stereoscopic cinematography* in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Providence, RI, USA, June 16-21, 2012* (2012), 1–7.

54. Lowe, D. G. *Object Recognition from Local Scale-Invariant Features* in *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2* (1999).

55. Manap, N. & Soraghan, J. *Novel view synthesis based on depth map layers representation* in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video* (2011), 1–4.

56. Manap, N., Soraghan, J. & Petropoulakis, L. *Depth Image Layers Separation (DILS) algorithm of image view synthesis based on stereo vision* in *Signal and Image Processing Applications (ICSIPA)* (2013), 61–66.

57. Meier, B. J. *Painterly rendering for animation* in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996).

58. Mendiburu, B. *3D Movie Making: Stereoscopic Digital Cinema from Script to Screen* (Focal Press, May 2009).

59. Morse, B., Howard, J., Cohen, S. & Price, B. *PatchMatch-Based Content Completion of Stereo Image Pairs* in *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on* (Oct. 2012), 555–562.

60. Nazzar, Y., Bouchard, J. & Clark, J. J. *Detection of Stereo Window Violation in 3D Movies* in *ACM SIGGRAPH 2014 Posters* (2014).

61. Northam, L., Asente, P. & Kaplan, C. S. *Consistent Stylization and Painterly Rendering of Stereoscopic 3D Images* in *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering* (2012).

62. Northam, L., Asente, P. & S. Kaplan, C. Stereoscopic 3D image stylization. *Computers and Graphics* **37,** 389–402 (Aug. 2013).

63. Palmisano, S., Gillam, B., Govan, D. G., Allison, R. S. & Harris, J. M. Stereoscopic perception of real depths at large distances. *Journal of Vision* **10,** 19 (2010).

64. Pan, H., Yuan, C. & Daly, S. *3D video disparity scaling for preference and prevention of discomfort* in *Proc. SPIE* **7863** (2011), 786306–786306–8.

65. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **12,** 2825–2830. ISSN: 1532-4435 (Nov. 2011).

66. Pennington, A. & Giardina, C. *Exploring 3D: the new grammar of stereoscopic filmmaking* (Burlington, Mass. Focal Press, 2013).

67. Rhemann, C., Hosni, A., Bleyer, M., Rother, C. & Gelautz, M. *Fast Cost-Volume Filtering for Visual Correspondence and Beyond* in *IEEE Computer Vision and Pattern Recognition (CVPR)* (Colorado Springs, USA, 2011).

68. Richards, W. & Miller, J. F. Convergence as a cue to depth. *Perception & Psychophysics* **5,** 317–320 (1969).

69. Richardt, C., Orr, D., Davies, I., Criminisi, A. & Dodgson, N. A. *Real-Time Spatiotemporal Stereo Matching Using the Dual-Cross-Bilateral Grid* in *Computer Vision – ECCV 2010* (eds Daniilidis, K., Maragos, P. & Paragios, N.) (Springer Berlin Heidelberg, Berlin, Heidelberg, 2010), 510–523.

70. Richardt, C., Świrski, L., Davies, I. P. & Dodgson, N. A. *Predicting Stereoscopic Viewing Comfort Using a Coherence-based Computational Model* in *Proceedings of the International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging* (ACM, New York, NY, USA, 2011), 97–104.

71. Richardt, C., Stoll, C., Dodgson, N., Siedel, H.-P. & Theobalt, C. *Coherent Spatiotermporal Filtering, Upsampling, and Rendering of RGBZ Videos* in *Computer Graphics Forum (Proceedings of Eurographics 2012)* (2012).

72. Sarto, D. *Stereoscopic Supervisor Robert Neuman Talks Disney 3-D* VFXWorld Magazine Online. Apr. 2012.

73. Scharstein, D. & Pal, C. *Learning Conditional Random Fields for Stereo* in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on* (June 2007), 1–8.

74. Schmid, J., Senn, M. S., Gross, M. & Sumner, R. W. *OverCoat: an implicit canvas for 3D painting* in *ACM SIGGRAPH 2011 papers* (2011).

75. Secord, A. *Weighted Voronoi Stippling* in *Proceedings of the 2Nd International Symposium on Non-photorealistic Animation and Rendering* (2002).

76. Seymour, M. *Art of stereo conversion: 2D to 3D* `https://www.fxguide.com/featured/art-of-stereo-conversion-2d-to-3d/`.

77. Snavely, N., Zitnick, C. L., Kang, S. B. & Cohen, M. *Stylizing 2.5-D video* in *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering* (2006).

78. Stavrakis, E. & Gelautz, M. *Image-based Stereoscopic Painterly Rendering* in *Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques* (Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2004), 53–60.

79. Stavrakis, E. & Gelautz, M. *Stereoscopic Painting with Varying Levels of Detail* in *In Proceedings of SPIE - Stereoscopic Displays and Virtual Reality Systems XII* (2005), 55–64.

80. Sun, J., Zheng, N.-N. & Shum, H.-Y. Stereo Matching Using Belief Propagation. *IEEE Trans. Pattern Anal. Mach. Intell.* **25,** 787–800. ISSN: 0162-8828 (July 2003).

81. Telea, A. An Image Inpainting Technique Based on the Fast Marching Method. *J. Graphics, GPU, and Game Tools* **9,** 23–34 (2004).

82. Toth, R., Hasselgren, J. & Akenine-Möller, T. *Perception of Highlight Disparity at a Distance in Consumer Head-mounted Displays* in *Proceedings of the 7th Conference on High-Performance Graphics* (Los Angeles, California, 2015), 61–66. ISBN: 978-1-4503-3707-6.

83. Ulyanov, D., Vedaldi, A. & Lempitsky, V. Deep Image Prior. `https://dmitryulyanov.github.io/deep_image_prior` (2017).

84. Wang, L., Jin, H., Yang, R. & Gong, M. *Stereoscopic inpainting: Joint color and depth completion from stereo images* in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (June 2008), 1–8.

85. Wang, O. *et al. StereoBrush: Interactive 2D to 3D Conversion Using Discontinuous Warps* in *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling* (Vancouver, British Columbia, Canada, 2011), 47–54.

86. Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* **13,** 600–612 (2004).

87. Weiss, B. *Fast median and bilateral filtering* in *ACM SIGGRAPH 2006 Papers* (2006).

88. Wheatstone, C. *Contributions to the Physiology of Vision: Part the First: On Some Remarkable and Hitherto Unobserved Phenomena of Binocular Vision* (1838).

89. Zone, R. *Stereoscopic Cinema and the Origins of 3-D Film, 1838-1952* (University Press of Kentucky, 2014).