2013-06-24

PatchMatch-Based Content Completion of 3D Images

Joel Arthur Howard
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd
Part of the Computer Sciences Commons

BYU ScholarsArchive Citation
Howard, Joel Arthur, "PatchMatch-Based Content Completion of 3D Images" (2013). All Theses and Dissertations. 3989.
https://scholarsarchive.byu.edu/etd/3989

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
PatchMatch-Based Content Completion of 3D Images

Joel A. Howard

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Bryan S. Morse, Chair
William A. Barrett
Sean Warnick

Department of Computer Science
Brigham Young University
June 2013

Copyright © 2013 Joel A. Howard
All Rights Reserved
This thesis presents a method for completing target regions ("hole filling") in RGB stereo pairs. It builds upon the state of the art for completing single images by matching to and then blending source patches drawn from the rest of the image. A method is introduced for first completing the respective disparity maps using a coupled partial differential equation based on that of Bertalmio, et al. extended to create mutual disparity consistency. Estimated disparities are then used to guide completion of the missing color image texture. An extension to the coherence-based objective function introduced by Wexler, et al. is then introduced, which not only encourages coherence of the respective images with respect to source images but also stereoscopic consistency between the two. The PatchMatch algorithm of Barnes, et al. is extended to cross-image searching and matching. This matching is capable of automatically copying from corresponding unoccluded portions of the other image without requiring an explicit preliminary warping step. Stereoscopic consistency is produced by giving preference to matches with cross-image consistency when blending source patches. Additionally, the PatchMatch algorithm is extended to draw from scaled texture in a directed fashion based on the 3D structure of the scene estimated from the stereo image pairs. Results demonstrate that this method produces better completion than either single-image completion or previous methods for stereo completion.

Keywords: stereo, vision, completion, texture
ACKNOWLEDGMENTS

I would like to thank my committee, especially Dr. Morse, for their support, work, and advice in pursuing this research. I would also like to thank our collaborators at Adobe Research, Scott Cohen and Brian Price, for helping fund and collaborate on the research that went into this thesis. Their time and ideas helped drive this research and the results. Last but not least, I would like to thank my wife Melissa for her patience and support. I couldn’t have done this without her.
Table of Contents

List of Figures vi

1 Introduction 1

2 PatchMatch-Based Content Completion of Stereo Image Pairs 8
   2.1 Introduction ......................................................... 8
   2.2 Related Work ..................................................... 10
   2.3 Single-Image Completion as a Foundation ....................... 13
   2.4 Stereo Image Completion ......................................... 15
       2.4.1 Depth Completion ........................................... 16
       2.4.2 Texture Matching and Synthesis ............................ 18
   2.5 Results ............................................................ 23
   2.6 Conclusion ......................................................... 30

3 Extending PatchMatch-Based Content Completion of Stereo Image Pairs to include Scaled Patches 32
   3.1 Introduction ........................................................ 32
   3.2 Methods ............................................................ 34
       3.2.1 Estimating Uniform Scale Transformation ............... 34
       3.2.2 Limiting the Random Search Step ....................... 36
       3.2.3 Propagation Step ............................................. 37
       3.2.4 Nonuniform Scale Transformations ...................... 38
       3.2.5 Normalized and Scaled Disparities ..................... 39
List of Figures

1.1 Example of Adobe’s Content Aware Fill ..................................................... 2
1.2 Example of a state of the art image completion algorithm failing to maintain the stereoscopic effect. .............................................................. 3
1.3 Example and explanation of certain properties of stereo image pairs. .......... 6

2.1 Comparison of single image completion to our method and Wang et. al. [1] ...... 9
2.2 An example of our preliminary disparity completion ................................. 18
2.3 Source and target region of a stereo image completion. ............................ 19
2.4 An illustration of our extensions to PatchMatch [2] .................................. 21
2.5 Blending phase of the algorithm. .............................................................. 22
2.6 Results of our methods when applied to “graffiti removal” ....................... 25
2.7 Foreground Object removal (pig and spray bottle) .................................. 26
2.8 Intermediate Object removal (teddy bear) ............................................. 27
2.9 Results of our algorithm run on examples from Wang et al. [1] ................. 28
2.10 Comparison to Wang et. al. [1] .............................................................. 29

3.1 Example of the shortcomings of Adobe’s Content Aware Fill and our methods from Chapter 2. ............................................................... 33
3.2 Example scaled patches determined by the disparity computed from the stereo image pair ................................................................. 35
3.3 Example of the propagation phase with scaled patches ............................ 37
3.4 Flat, washed-out regions introduced by unbounded scaling of source patches . . . 41
3.5 Results of Book image. ................................................................. 45
3.6 Comparison of our results against PatchMatch [2] and the method from Chapter 2. 46
3.7 Results of the Window image with comparison. 47
3.8 Results of the Brick Wall image with comparison. 49
3.9 Results of Tsukuba Image. 50
3.10 Results of Grass Hill image. 51
3.11 Our extensions run on the Plant images and comparison with our results from Chapter 2. 52
3.12 Our extensions run on the Basket Ball images and comparison with our results from Chapter 2. 53
3.13 Our extensions run on the Teddy images and comparison with our results from Chapter 2. 53
Chapter 1

Introduction

The technology behind stereoscopic (3D or stereo for short) images has drastically improved since their invention in 1838 by Sir Charles Wheatstone. There have been several surges of popular use of the technology with varying success. The latest surge, begun with the reintroduction of 3D movies, has spurred the development of consumer grade 3D digital displays and cameras.

Consumers have long had the ability to produce digital media, whether directly through digital cameras or indirectly by scanning physical media. Coupled with the rise of digital media, products have surfaced providing consumers with tools to manipulate their own personal digital media. These tools range from simple blur filters and color adjustments to more sophisticated tools such as blending and automatic damage repair. What’s more, consumers have become used to, and even expect to have, the ability to use these tools on their personal digital media. Thus, with the increasing availability of consumer-grade stereoscopic cameras and camcorders, there has arisen a need to extend currently available tools to be able to work with stereoscopic media.

Recent forays into this field have included tools for assisted object selection [3], content cut and paste [4], and applying various filters and artistic effects [5]. There have also been some attempts to extend older algorithms for image completion (or inpainting) to stereo images [6]. Inpainting is an algorithm for automatically and intelligently replacing a selected portion, or target region, of an image with content pulled from outside that selected region, or source region. Over the years, there have been many algorithms proposed, which in many instances, perform this operation well for single images. Examples of these algorithms range from PDE-based approaches like
Figure 1.1: Example of Adobe’s Content Aware Fill
Figure 1.2: An example of a state of the art image completion algorithm (Adobe’s Content Aware Fill) failing to complete a pair of stereo images with stereoscopic consistency. (b) shows the regions (outlined in green) to be filled in the original stereo pair (a). (c) shows the result of using Adobe’s Content Aware Fill. Notice that the algorithm not only fills in both the left and right images differently, but fails to complete the structure correctly (i.e., copies in part of the Doc figurine).
Bertalmio’s “Image Inpainting” [7] to patch-based approaches like Adobe’s Content Aware Fill (Figure 1.1) which is based on [8] and [2]. The latter class of methods all follow the same basic two steps: (1) decide where to draw content from and (2) fill the target region with (a possibly blended version of) the selected source content. These algorithms, however, were designed for single images and fail to produce consistent 3D results, or take advantage of the inherent depth information, when applied to stereoscopic media (as shown in Figure 1.2).

When viewing a 3D movie or image, the viewer is actually seeing two images of the same scene from slightly different perspectives. Through the use of special glasses or screen filters the left and the right eye see only the left and the right image respectively. The different perspective seen by each eye causes the brain to perceive 3D structure. This is called the stereoscopic effect. This effect is what makes editing stereo images so difficult. When editing stereo images, both images must be edited in such a way to ensure the left and right images agree, i.e. there are no stereoscopic inconsistencies. If they do not agree, artifacts are introduced into the set, confusing the brain and breaking the 3D effect causing discomfort to the viewer.

The stereoscopic effect, however, also provides opportunities to take advantage of the additional depth information provided by the stereoscopic media. Because of the difference in perspective (or more specifically the difference in camera position called the baseline), points captured in the left image of the stereo pair are offset in the right (see Figure 1.3). This difference in position (measured in pixels) is called disparity. An interesting property of disparity is that it is inversely proportional to the actual depth of the point. This is due to the geometry between the cameras and the point in 3D space (see Figure 1.3). Although reconstructing 3D structure from stereo is still not a completely solved problem, there is a large body of work in this area [9], with a number of good algorithms that create reasonable results under most conditions. It should be noted that this technique is dependent on the cameras being in, or very near, the canonical configuration (i.e., the cameras are offset only in the X or horizontal direction and both are pointing in the same direction perpendicular to the base line). It should also be noted that estimating exact 3D coordinates
and structure is only possible with a known baseline and known camera parameters such as focal length, sensor size, etc.

Stereo images also provide another advantage over single images, particularly when applied to the problem of image completion. Because of the difference in perspective between the two images, there are portions of the scene that one camera captures that the other cannot and vice versa (see Figure 1.3). An easy way to view this phenomenon is to examine a finger with one closed eye. Alternating which eye is closed reveals different portions of the background while blocking others. These half-seen portions of the background are known as half (or partially) occluded regions. Taking advantage of half-occluded regions, especially when removing an object in the scene, is important when attempting to produce stereoscopically consistent results. This is due to the fact that it is extremely difficult to reproduce texture when there is little or no exactly matching texture elsewhere in the scene, while it is fairly trivial to draw directly from the half-occluded regions.

In this work, we extend Adobe’s Content Aware Fill (CAF) and the methods behind it to be able to complete stereo image pairs with stereo consistency. We accomplish this by altering the objective function at the heart of CAF’s algorithm to consider stereo consistency as well as image completeness. This alteration translates into several algorithmic changes that incorporate the new stereo consistency terms. These include:

1. altering the matching metric when building the nearest-neighbor field to consider depth as well as texture,

2. expanding the set of candidate patches when building the nearest-neighbor field to include the stereo corresponding patch as well as its nearest-neighbor,

3. expanding the set of candidate patches further to include scaled patches by using the depth of each target and source patch to estimate actual scale, and

4. weighting patches in the blending step according to their stereo consistency.
Figure 1.3: An example of a pair of stereo images with a diagram of an approximate configuration of the cameras and highlighted portions of the objects in the scene. The colors of the highlighted regions correspond to the colors of the projections of the points in the diagram. Note that the light blue region and its corresponding projection, while in the right image, are not in or projected onto the left image due to the occluding object. This is an example of a half-occluded region.
This extended algorithm is not only able to draw from good patches found in both images, match 3D structure as well as texture, more correctly draw texture from varying depths, but also will drive stereo consistent results by preferring patches that are stereo consistent and blend out the areas that are already inconsistent.

The following chapters outline the methods and the results of our contributions. Chapter 2 is a self-contained research paper describing our methods for disparity map and stereo image completion. It addresses items 1, 2, and 4 enumerated above. Chapter 3 presents additional unpublished work that extends the methods in Chapter 2 to include source patches scaled by the relative distances of the source and target patches (item 3 as just enumerated).
Chapter 2

PatchMatch-Based Content Completion of Stereo Image Pairs

This chapter is a paper presented at the conference of Three-Dimensional Image Modeling, Processing, Visualization, and Transmission (3DIMPVT) held in Zurich, Switzerland on October 13-15, 2012. This paper was co-authored with Bryan Price and Scott Cohen, our collaborators at Adobe Research.

2.1 Introduction

As consumer-grade 3D stereo cameras become increasingly common, users want to be able to edit stereo images in ways they are used to doing for individual images (e.g., [1, 3, 4, 10]). This introduces the challenge of maintaining stereoscopic fidelity between the edited images in order to maintain the correct 3D effect when viewed. It also introduces a new set of opportunities to take advantage of the additional depth information provided by the stereo pair.

A common editing operation is to replace selected regions of the image with other content by intelligently drawing from the rest of the image. This may be done to repair damage to the images, remove unwanted objects or markings on objects, etc. Many methods for doing such image completion in single images have been proposed, and in many situations these work quite well (e.g., [2, 8, 11, 12]). Stereo image completion can build upon these methods for single-image completion, with the additional requirement to maintain proper stereoscopic consistency in the resulting color images. An example of such stereo completion is shown in Fig. 2.1b.

While stereoscopic consistency is a challenge, the availability of additional depth information from stereo pairs allows for better image completion than might be achieved for single-image
Figure 2.1: PatchMatch-based stereo completion. When the original images (a) are completed independently (b), there are no guarantees of stereo consistency, and the lack of depth information causes inclusion of content that, while reasonable from a texture standpoint alone, may not make sense geometrically. Using depth information, our result (c,e) is both stereoscopically consistent and produces a better completion using depth-appropriate content. Half-occluded areas are automatically reproduced faithfully in the other image without an explicit warping step, and the completion of regions that are occluded in both images is superior to that reported in Wang, et al. [1] (d).
completion alone. That is, the available depth information can be used to provide an additional
dimension of information for creating a correct completion [13].

This paper presents a method for extending the current state-of-the-art for single-image
completion [2, 8, 14], to stereo image pairs by both leveraging the depth information extracted
from a pair and filling paired regions such that the result is consistent when viewed stereoscopically.
In particular, it makes the following contributions: 1) a coupled-PDE approach to pre-filling the
disparity maps in a way that maintains mutual consistency, allowing these disparity estimates to
aid further matching and blending, 2) an extension to the PatchMatch-based approach of Barnes,
et al. [2] that provides cross-image search and depth-sensitive comparison to both a target patch
and its stereo-corresponding patch in the other image, and 3) an extension to the weighted blending
of matched source patches that gives preference to strong stereo correspondence at the desired
disparities.

The result is a system that simultaneously tries to maximize stereoscopic consistency as well
as coherence of the respective target regions with respect to the rest of the source images. It can
match patches in a way that allows for cross-image copying in the case of regions that are originally
half-occluded without requiring an explicit pre-copying step. This allows it to gracefully handle
loosely marked masks without requiring that they correspond in the two images. This approach can
handle removal and replacement of texture on a 3D object in one or both images, removal of 3D
structural detail on an object (e.g., a wall or other surface), or entire removal of an object. The use
of depth information allows for better completion than single-image completion alone.

2.2 Related Work

Stereo correspondence and computation is a well-studied problem and continues to be a topic of
ongoing research [9]. This paper does not attempt to address those problems, though we do have
to work within the limitations of these methods. Of particular interest are methods proposed for
detecting and handling half-occluded regions [15, 16], those seen in one image but not the other.
The body of literature related to image completion (also known as inpainting, image repair, or texture synthesis) is also extensive. Two general approaches have been most prevalent: partial differential equations (PDEs) and neighborhood-matching.

PDE-based inpainting approaches, first proposed by [7], work by trying to smoothly diffuse content from outside the target region into it in such a way as to preserve geometry and structure. These methods do well for images with smooth regions, but they do not reproduce texture well. One can reproduce structure and texture separately by first creating a smooth approximation of the region, separating it from the texture, then filling the missing structure using a PDE and the missing texture using separate texture synthesis [17]. Our method draws from this idea by using PDE-based reconstruction of missing depth and texture-synthesis reconstruction of the missing texture.

Texture synthesis by matching patches surrounding missing pixels to other (complete) patches elsewhere in the image was first proposed by [11]. While texture synthesis is useful for filling regions with a single missing texture, it often struggles to correctly reproduce the structure of region boundaries within the target region, due in part to its greedy-algorithm “onion skin” approach of filling concentrically from the boundary of the target region. Although [18] and similar approaches try to address this by prioritizing filling along the directions of structure outside the region, these greedy methods can still struggle to produce coherent texture throughout the target region.

Kwatra, et al. [12] first proposed framing texture synthesis as an optimization solved through iterative improvement, which leads to greater consistency in the synthesized texture and generally more coherent completion. By iteratively refining the texture rather than filling it in in a single greedy pass, this also allows for a multiscale approach that can provide more robust filling overall. This idea was developed further by Wexler, et al. [8], who proposed an E-M framework for iteratively matching target patches to their best matches in the rest of the image and then blending these patches in a weighted fashion to maximize the coherence of the target region with respect to the rest of the image. This idea was extended by [14], which applied these methods to not only image completion but other applications such as image retargeting or montage creation. With all of these approaches,
though, the matching of target patches to potential source patches remains the computational bottleneck. Barnes, et al. [2] addressed this specific problem using their “PatchMatch” algorithm, which leverages spatial propagation and random search to create an efficient method for finding good matches. The core ideas of the PatchMatch algorithm have since also been used for noise reduction [19] and stereo correspondence [6]. The work described in this paper specifically builds upon the efficiency of the PatchMatch approach while extending it to stereo images.

The idea of using texture synthesis for image completion in stereo images was explored by Wang, et al. in [1], which to date is the only other method proposed for producing stereo-consistent filling. This approach relies heavily on filling half-occluded regions by directly copying and warping their counterparts in the other image. It then uses the prioritized but greedy texture-synthesis approach of [18], extended to RGBD (“D” for disparity) matching, to simultaneously fill in missing disparities along with missing color texture. Because of this, it suffers from the limitations of other greedy-filling approaches. They also add an additional constraint to the matching, requiring that the target region be filled in only with more distant content. Their approach fills each side of the stereo pair independently with a single pass of pixel-based texture synthesis. Because this is not guaranteed to result in stereo consistency, they find all patches for which consistency was achieved (using the weak-consistency constraint of [20]), then remove them from the mask and iteratively repeat the process until all of the target region has been completed consistently.

The recent work of He, et al. [13] extended the PatchMatch approach to include the additional depth information from stereo images using an RGBD approach similar to [1]. However, this work used the disparity information from a stereo pair to assist the completion of a single image from the pair. It did not attempt to produce a stereo result by completing both sides of the pair in a stereo-consistent fashion, nor did it draw information from the other image to fill half-occluded regions as did [1].

Our approach uses the RGBD-matching approach from [1] but extends that work as follows: 1) we first use a coupled PDE-based approach to estimate the missing disparities in the target region using a method similar to [7] but extended to produce stereoscopically consistent disparity...
maps, thus allowing these filled disparities to assist the synthesis of stereoscopically consistent color texture in these regions; 2) we do not explicitly copy pixels into half-occluded regions but rather let this happen naturally as part of a cross-image searching, matching, and blending process, thus creating smoother blends near the boundaries of these regions; 3) we iteratively optimize the synthesized texture in a manner similar to [2] and [8] (using an extended variation of the PatchMatch search in [2]) rather than using a single greedy filling pass; and 4) we do not greedily remove stereoscopically consistent patches from the mask but rather give priority to them during both the matching and blending steps of the E-M iteration.

Other methods have been proposed for removing objects or otherwise completing missing data in stereo video sequences, but these rely primarily on exploiting redundancy in the video to find the unoccluded content in other frames (e.g., [21]). For a single stereo pair, these other unoccluded views are not available.

It is also worth noting significant recent interest in related stereo-image editing problems such as selection [3], copying and pasting [4], and retargeting [10], which provide useful and complementary tools for stereo-image editing but do not address this particular need.

2.3 Single-Image Completion as a Foundation

As noted, our approach builds directly on the single-image, PatchMatch-based completion approach of Barnes, et al. [2] in a fashion similar to [13]. Using notation similar to that of [2], we seek to minimize the following measure of image coherence [8, 14]:

\[
d_{\text{total}}(S, T) = \sum_{t \in T} \min_{s \in S} d(s, t)
\]

where \(T\) is the target region (the region to be filled), \(S\) is the source region (rest of the image), \(t \in T\) and \(s \in S\) are patches within the target and source regions respectively, and \(d(s, t) = \|s - t\|_2^2\) is a measure of the difference between patches \(s\) and \(t\). Intuitively, this tries to ensure that every patch
within the filled region is similar to a corresponding patch in the rest of the image—any artifacts introduced would not match patches in the rest of the image and would thus be penalized.

Wexler, et al. [8] point out that although this does not imply an optimization strategy, one can observe that it will be satisfied if two conditions are met at every point \( p \):

1. All of the patches \( t \in T \) that overlap point \( p \) have an exact match \( s \in S \) (and hence \( d(s, t) = 0 \)), and

2. All of the patches \( t \in T \) overlapping \( p \) agree on the value at \( p \) (so that the blended results of the patches introduce no additional errors).

They (and subsequent approaches) thus take an E-M-style approach, iteratively alternating between matching each target patch \( t \in T \) to its best match \( s \in S \), then blending the resulting patches to synthesize content in the target region.

The PatchMatch algorithm introduced in [2] avoids exhaustive search by leveraging spatial propagation of matches and random search to create an efficient method for finding good matches. The result is a “nearest neighbor field” (NNF), which provides a mapping from all patches in the image to their best (so far) match outside the target region, and which we denote as \( s = \text{NNF}(t) \).\(^1\)

In a manner similar to [2] we use the PatchMatch algorithm to update best matches and then blend them in the target region, weighting each blended patch by a monotonically decreasing function of the distance from the patch to the boundary of the target region, which as noted by [8] helps drive content into the target region from outside. As with [2], we use a gradual-resizing approach [14] to create a multiscale pyramid. At the coarsest scale of the pyramid, we use a simple diffusion filling to initialize the PatchMatch-based E-M iteration. For subsequent scales, we upsample the NNF from the previous scale as in [2].

\(^1\)Rather than using offsets as in [2], we use absolute positions as those authors do in later descriptions of that work.
2.4 Stereo Image Completion

In a manner similar to [1, 13], we treat the stereo pair as four-valued RGBD images. We assume that the images have already been rectified and that stereo correspondence has already been computed. The methods proposed here do not assume any particular correspondence algorithm, though obviously the more accurate the disparities the better. We also assume that the user has provided masks (e.g., Figure 1.1b) specifying the respective target regions. This could be done manually or by using a stereo-based selection tool [3]. Our method does not assume or require that the masks be placed consistently between the two images, so even simple freehand marking is sufficient.

When working with stereo images, it is important to recognize that the disparity maps have very different characteristics than those of their respective color RGB images. Unlike color images with rich texture, disparity maps generally involve smooth regions with strong spatial structure, qualities commonly exploited in algorithms for computing stereo disparity [9]. Unless the user chooses to retain the existing disparities, we then first fill the disparity maps in the respective target regions. Accurate disparity maps (or at least reasonable estimates for such in the target region) are then used to guide the selection of correct source patches when completing the target region’s color texture.

Image completion may be used to remove entire foreground objects (the case most often focused on in the literature, e.g., [1, 13]), but there are two other potential uses:

- To remove 3D structural detail on an object, wall, etc.

- To retain structure while replacing object color defects (e.g., removing graffiti, cast shadows, etc.).

To accommodate these cases, we allow the user to specify whether to fill the disparities in the target region or to retain the original disparity maps. We then synthesize stereoscopically consistent color texture based on these disparities.
2.4.1 Depth Completion

We use two disparity maps, $D_L$ and $D_R$, which is important for handling depth in half-occluded regions. Prior to use, we first fill any holes from the stereo correspondence using a “smaller hole” variation of the method presented in this section.

We use the PDE-based inpainting method of [7] to recover smooth spatial structure in the disparity maps. Inpainting a single disparity map $D$ using the original method of [7] involves the following iteratively solved PDE:

$$\frac{\partial D}{\partial t} = \nabla L \cdot \nabla_D D$$

(2.2)

where $L = \nabla^2 D$ denotes the Laplacian of the disparity map. Intuitively, this PDE tries to propagate image curvature along image level curves, thus filling regions and preserving edge structure. To reduce the number of iterations required for the numerical implementation of Eq. 2.2, we first initialize the target region using a simpler diffusion-based fill [22], allowing diffusion into the target region only from disparities smaller (farther from the cameras) than the original content in the region. (Such a depth-ordering restriction was also used in the approach in [1].) We have found this particularly useful when removing the visible portion of objects that are themselves partially occluded. (Fig. 2.8).

Inpainting the two disparity maps separately does not guarantee mutual consistency, though. To enforce this, we adopt the weak consistency constraint of [20], similar to how it is used in [1]. The values in two disparity maps $D_L$ and $D_R$ may be characterized as follows:

- Consistent (seen in both images)

$$D_L(x, y) = D_R(x - D_L(x, y), y)$$
$$D_R(x, y) = D_L(x + D_R(x, y), y)$$

(2.3)
• Half-occluded (seen in one image, occluded in other)

\[ D_L(x, y) < D_R(x - D_L(x, y), y) \text{ or } D_R(x, y) < D_L(x + D_R(x, y), y) \]  

(2.4)

• Inconsistent (physically impossible)

\[ D_L(x, y) > D_R(x - D_L(x, y), y) \text{ or } D_R(x, y) > D_L(x + D_R(x, y), y) \]  

(2.5)

We modify the PDE in Eq. 2.2 to create a pair of coupled PDEs that include inpainting of the respective disparity maps and additional terms to create mutual consistency:

\[ \partial D_L/\partial t = \nabla L_L \cdot \nabla_L D_L + \lambda \rho_L \]  

(2.6)

\[ \partial D_R/\partial t = \nabla L_R \cdot \nabla_R D_R + \lambda \rho_R \]  

(2.7)

where \( L_L = \nabla^2 D_L, L_R = \nabla^2 D_R, \) and

\[ \rho_L(x, y) = \begin{cases} 
D_R(x - D_L(x, y), y) - D_L(x, y) \\
0 
\end{cases} \]

if \( D_R(x - D_L(x, y), y) - D_L(x, y) < \epsilon \)  

(2.8)

\[ \rho_R(x, y) = \begin{cases} 
D_L(x + D_R(x, y), y) - D_R(x, y) \\
0 
\end{cases} \]

if \( D_L(x + D_R(x, y), y) + D_R(x, y) < \epsilon \)  

(2.9)

are consistency terms (\( \epsilon \) controlling the tolerance). If Eq. 2.3 applies at a given pixel to within \( \epsilon \) tolerance, we assume the disparities should be consistent and adjust them to be more similar as required. If Eq. 2.4 applies by more than an \( \epsilon \) difference and the matching pixel in the other image has consistent disparities, we assume half-occluded pixels (for example, the blue highlighted region
in Figure 1.3 where part of the building is visible in the right but occluded by the statute in the left) and allow them to retain their differing disparities. If Eq. 2.5 applies, the maps are adjusted accordingly to correct this inconsistency. For the results shown here we use $\epsilon = 1$.

### 2.4.2 Texture Matching and Synthesis

To synthesize texture over the respective disparity maps we extend the objective function in Eq. 2.1 to allow for the drawing of source textures from either image and to penalize stereo mismatches between the two images.

As illustrated in Figure 2.3, Let $S_L$ and $S_R$ denote source regions in the left and right images respectively, and similarly let $T_L$ and $T_R$ denote respective target regions. Also let $C_{LR}(t)$ denote the mapping from patch $t_L \in T_L$ centered at $(x, y)$ to the corresponding patch $t_R \in T_R$ centered at $(x - D_L(x, y), y)$. Similarly, let $C_{RL}(t)$ denote the mapping from patch $t_R \in T_R$ centered at $(x, y)$ to the corresponding patch $t_L \in T_L$ centered at $(x + D_R(x, y), y)$. To simplify further notation, we let $C(t)$ denote the stereo-corresponding patch in the other image such that $C(t) = C_{LR}(t)$ for patches in the left image and $C(t) = C_{RL}(t)$ for patches in the right image.
We thus define optimization of stereo-filling coherence as minimization of the following objective function:

\[
d_{\text{total}}(S_L, S_R, T_L, T_R) = \sum_{t \in T_L \cup T_R} \min_{s \in S_L \cup S_R} d(s, t) + \sum_{t \in T_L} d(t, C_{LR}(t)) + \sum_{t \in T_R} d(t, C_{RL}(t))
\]  

(2.10)

Here, we redefine the patch-difference measure \(d(s, t)\) to be a mean squared difference between the RGBD values of the patches unless otherwise noted.

The first term is similar to Eq. 2.1 and encourages filling the target regions in the respective images coherently. Note that we explicitly allow here the matching of patches across the two images in order to provide a richer set of source patches. The additional two terms encourage stereo consistency by penalizing patches that exhibit visual dissimilarity at the relevant disparity.

We extend the E-M approach of [8] by further observing that Eq 2.10 is minimized if both of the conditions identified in Sect. 2.3 for minimizing Eq. 2.1 are met and if all pixels in the target regions are filled with content exactly matching their corresponding patch in the other image at the relevant disparity. To encourage this we modify the patch-blending step of the E-M process to give increased weight to patches that are stereo-consistent (unless occluded in the other image). We also expand the patch-matching search to include patches from both images, including a propagation step designed to facilitate stereo consistency.
**Stereo Patch Matching**

Because the two source images provide a larger set of source patches than either image alone, and because some useful patches may be visible in one image but not in the other, we extend the PatchMatch algorithm [2] to include cross-image searching.

The original PatchMatch algorithm uses two parts to search for better patches than are currently found:

- A propagation step, in which the neighbors of patches matched to those neighboring the current patch are considered (i.e., to update $\text{NNF}(t)$ we consider the current NNF matches for the neighbors of $t$), and
- A random search step.

We extend this as illustrated in Fig. 2.4 to include

- A stereo-correspondence step, in which we consider the stereo-corresponding patch in the other image, i.e., for patch $C(t)$, and
- A stereo propagation step, in which we consider matches found for the neighbors of the corresponding patch $C(t)$.

A similar form of cross-image propagation was also used by Bleyer, et al. [6], though that work focused on stereo correspondence and propagated parameters for slanted support windows rather than constructing an NNF.

Including the current values for the stereo-corresponding patch in the other image $C(t)$ is the only time we allow matching to a patch that is inside or overlaps either target region. This inclusion in the expanded search allows copying (and subsequent blending) of good patches found in the other image, leading to minimization of the latter two terms of Eq. 2.10. However, the stereo-corresponding patch is not just copied over blindly—it must still be selected as the best-corresponding patch during the patch-matching process, which ultimately allows the image for which the best completion is found to dominate the other.
Figure 2.4: Cross-image expanded patch matching as described in Section 2.4.2. In addition to the spatial propagation (green) and random search (blue) components of PatchMatch [2], we include a stereo propagation component (red, brown) in the search. This considers the corresponding patches in the other image and their current best match.

During this search the corresponding patch may be in the source, not target, region for the other image. This happens commonly when removal of a foreground object disoccludes part of one image that is visible in the other. The approach of [1] relies heavily on explicitly warping such originally half-occluded data, but the proposed method does not require such an explicit copying pre-step. Cross-image copying happens automatically as part of the searching and synthesis process. This allows trading off between matching the surrounding area and matching across images, making the method more robust to differences in imaging parameters and to minor errors in the disparity maps.

In addition to the spatial propagation step of the original PatchMatch algorithm, we also include a “stereo propagation” step, which expands the pool of candidate source patches further to include not only the corresponding patch $C(t)$ in the other image, but the current best matches to $C(t)$ according to the other image’s NNF. Due to subpixel disparities, which are present in the multiscale hierarchy even if the original disparity maps use only integer disparities, this means searching two possible candidates using the floor and ceiling of the $x$ coordinate of $C(t)$. 
Figure 2.5: Blending phase of the algorithm. In this example, though $t_1$ and $t_2$ match quite closely to their respective source patches $s_1$ and $s_2$, $t_2$ will receive a higher weight to its vote for the color of pixel $p$ during the blending step because it is stereo consistent with its stereo corresponding patch in the other image.

**Stereo-Consistent Patch Blending**

Once the nearest-neighbor field is updated using this extended PatchMatch algorithm, a “patch voting” step is performed to blend the source patches and fill the target region.

To promote stereo consistency we give increased blending weight to those patches $t$ that are consistent with their stereoscopic counterparts $C(t)$ in the other image, as illustrated in Figure 2.5.

The color $c$ of a target pixel $p$ is calculated using a weighted blending of the values of the source patches $s$ matched to each target patch $t$ that overlaps pixel $p$, in a manner similar to [2] or [8]. Let \{\(t^1, t^2, \ldots, t^k\)\} denote the set of patches overlapping pixel $p$, whether entirely inside the target region $T$ or not, and \{\(s^1, s^2, \ldots, s^k\)\} their respective best matches. If we let $c^i$ denote the color for pixel $p$ suggested by the source patch $s^i$ and weight $w^i$ denote the weight given to patch $t^i$, the color $c$ for pixel $p$ is given by the weighted blending

$$c = \frac{\sum_i w^i c^i}{\sum_i w^i} \quad (2.11)$$
The weights $w^i$ are a combination of two factors, the one used for single-image filling as described at the end of Sect. 2.3 and another that penalizes stereoscopic mismatches:

$$w^i = w^i_d w^i_s$$  \hfill (2.12)

The distance-based weight $w^i_d$ is given as in [8] by

$$w^i_d = \gamma^{-\text{dist}(p^i, T)}$$  \hfill (2.13)

where \(\text{dist}(p^i, T)\) is the distance from $p_i$ (the center of patch $t^i$) to the boundary of the target region $T$, or 0 if $p^i$ lies outside of $T$. (We use a value of $\gamma = 1.3$ as in [8].)

The stereoscopic-consistency weight $w^i_s$ is given by comparing the unoccluded parts of patch $t^i$ to its (possibly subpixel) counterpart in the other image:

$$w^i_s = e^{-\frac{\bar{d}_s(t^i, C(t^i))}{2\sigma^2}}$$  \hfill (2.14)

where the occlusion-respecting patch squared difference $\bar{d}_s(t^i, C(t^i))$ is calculated as the mean squared difference between the mutually unoccluded portions of the patches $t^i$ and $C(t^i)$, again using a variation on [23] to handle subpixel comparison. If the entire patch $t^i$ is occluded from view in the other image, we set $\bar{d}_s(t^i, C(t^i))$ to the maximum $3 \cdot 255^2$ to give it minimal (but non-zero!) weight in the blending. Within the half-occluded region these patches all have the same (low) weight, which effectively removes the effect of this weighting factor from Eq. 2.12 through the normalization in Eq. 2.11. This has the effect of causing half-occluded regions to be filled from the unoccluded side.

### 2.5 Results

Fig. 2.1 compares our method (2.1b,d) to both independent PatchMatch-based single-image completion as implemented in Photoshop’s Content-Aware Fill tool (2.1c) and the results published in [1]
As expected, the results of single-image completion lack stereo consistency, but the lack of depth information also causes them to draw textural content that is not appropriate geometrically (duplication of the foreground “Doc” figurine).

The result for this image published in [1]—which included the result for the left image only—does not suffer from these geometric errors, but the quality of the texture completion is not as good as that of more recent texture synthesis methods. Specifically, compare their result in Fig. 2.1e to our result in Fig. 2.1d (left).

Fig 2.6 demonstrates our method used for “graffiti removal”, where the original disparity maps were retained and unwanted markings on the objects are removed. For the first pair we remove the handwritten “50” on the rock in the lower right (marking loosely around it) as well as the handwritten name on the upper right rock in the right image (outside the field of view of the left). We also remove the vein in the rock just above the center of the images. For the second pair we removed the partially occluded “Warning” sticker as well as the handwritten “X” marked on the wood above it. For both images the target areas are filled with realistic texture (rock or wood), and the completed regions in the left and right images are consistent. (For easier viewing we show the resulting target areas magnified and cropped since the rest of the image is unchanged.)

Figs 2.7 and 2.8 demonstrates our method used to remove objects. In the first pair in Figure 2.7 we remove the small pig figurine along with its shadow. In the right image there is little visible area between the pig and the edge of the colored block in the fabric behind, but our method is able to recover the texture correctly by drawing across images. The occluded edge between the cardboard base and the background cloth is correctly reproduced. In the second pair we remove the spray bottle. The reconstructed regions include the bottom of the white hamper, the wooden slats (including grain), and the pink object behind these slats—all stereoscopically consistent. The method struggled to reproduce the occluded T-junction in the lower right between the cloth, pillow, and wood—as all methods do with such high-curvature junctions—but the result is still
Figure 2.6: “Graffiti removal” mode—replacing texture while retaining original disparities. In the Rock images (a), handwritten marks on the rocks in the lower right and upper right (in the right image only) are removed, as well as the vein in the rock just above the center of the image. A warning sign as well as a penciled “X” is removed from the Wood images (b).

stereoscopically consistent. Fig 2.8 shows removal of an intermediate object, the teddy bear, which both occludes the background behind it and is itself partially occluded. Here the initial estimation
of the disparity maps while preserving depth ordering allows for correct completion of the missing background while preserving the top edge of the occluding birdhouse. Previous methods that are not aware of this this depth discontinuity do not preserve this edge well.

Fig. 2.9 shows our results for examples provided in the work of Wang et al. [1]. In these examples we remove the statue in the first set (a), and the basketball hoop in the second set (b).
Figure 2.8: Intermediate Object removal (teddy bear): Note that the bear occludes the background chart while being itself occluded by the foreground bird house, showing that the method works well for removing not only foreground objects but intermediate ones as well.

Notice particularly in the statue example that we effectively duplicate the half-occluded regions seen in one image into the other. This is particularly evident in the left image where the statue occludes a large portion of the white marble building. Despite this, the resulting fill correctly reproduces the marble texture as well as the set of windows on the far left of the marble building. In the basketball example (Figure 2.9b), we do a good job reproducing the texture of the roof and completing the bush occluded by the basketball hoop’s pole. It should be noted that we once again benefit from being able to draw from half-occluded regions. In particular, this is evident when examining the right image, which copies the chimney and the line of the roof from the left.

Fig. 2.10 provides side-by-side comparison of our results to [1], whose authors could only provide these three images (Figs. 2.1 and 2.10). Without manual constraints such as in [2] the method struggles to reproduce accurately the boundary between the two background buildings in the “statue” image, though this is a difficult task for any completion method. For the statue image our hierarchical, iterative approach better reconstructs the general shape of the boundary between the two buildings than does the single-resolution greedy approach (see Figure 2.10a). [1] completes
Figure 2.9: Results of our algorithm run on examples from Wang et al. [1]
Figure 2.10: Additional comparison to [1], who include only one result each for the statue (left only) and basketball (right only) images. (a), (b), (c), and (d) are highlighted differences/comparisons between our results (left) and the results of Wang et. al. [1] (right).
the red brick of the left building vertically, while ours completes the adjacent window horizontally, including the lower ledge separating it from the brick below, both reasonable results. They also construct an extra window above the pedestal, which we do not (see Figure 2.10b). Otherwise the result is as good as or better than that reported in [1], including drawing source patches across the images in half-occluded areas when possible. For the basketball image, our method again produces a stereo-consistent result with texture filling superior to that of [1]. In particular, the tiles of the completed roof are more regular in our result due to iterative refinement than obtained with the older greedy approach (see Figure 2.10c). Additionally, we better reproduce the texture of the bush just below the AC unit (see Figure 2.10d). The result for the left image was not published in [1] and is not available for comparison.

Figure 2.9b, however, shows one weakness of our algorithm. The extraneous fence pole in the left image is caused by an error in the original disparity map. Our algorithm fills the target region of the disparity map with plausible disparities by diffusing the disparities around the edge of the hole into the hole. Thus, as long as the disparities around the hole are correct, the resulting fill is reasonable and can produce reasonable results. However, when there are inconsistencies in the disparity map (particularly around the hole) our resulting fill still produces stereo consistent disparity maps, but introduces artifacts in the resulting texture fill that are less plausible.

2.6 Conclusion

This paper has presented a method that extends recent efficient, optimization-based methods for single-image completion to produce stereo-consistent results for stereo images. We have extended the objective function presented in [8] and used in [2] to include stereo-consistency terms, and we have introduced a new stereo-consistency weight during the patch-blending step that causes reduction of these terms. Cross-image searching and matching allows for greater selection of source patches and for copying half-occluded areas. This iterative optimization approach improves upon the greedy approach used in the only other prior work on this problem [1]. Results demonstrate the
stereo-consistency of the completed regions as well as their improved coherence with respect to the source images.

Acknowledgment

We would like to thank Dan Goldman and Eli Shechtman of Adobe Research for their helpful discussions and suggestions.
This chapter presents an extension of our Stereo PatchMatch algorithm by extending the algorithm to search for and find scaled versions of source patches. Though this is not an entirely novel extension, no current patch-based inpainting method takes advantage of the inherent depth information found in stereo images to intelligently search the range of possible scales for source patches.

3.1 Introduction

In the algorithm of Chapter 2 we not only compare the texture of each target patch with its prospective source patch but also their 3D structure. This comparison, achieved by comparing absolute disparities, effectively limits the potential source patches the algorithm can draw from to those of similar depth. This produces desirable results when applied to stereo images of frontoparallel surfaces, in many cases avoiding drawing texture from unsuitable areas. It can also work well when applied to surfaces that are not frontoparallel (have varying distances) so long as there is sufficient suitable source texture for each distance. Similarly, it may not work well even for frontoparallel surfaces when the only areas of suitable source texture are on other surfaces at distances other than that of the target. To address these two issues, we extend our previous algorithm to incorporate source patches scaled appropriately according the relative distances between the source and target regions.
Scaling source patches, however, is not a new concept. Since the work of Barnes et al. [2] in 2009, there have been several efforts to extend the PatchMatch algorithm beyond the scope of simple untransformed patches. Among these efforts are the extensions presented in the works of Barnes et al. [19], HaCohen et al. [24], Mansfield et al. [25], and Darabi et al. [26]. These extensions add, among several other enhancements, multiple geometric and photometric transformations including rotation, reflection, brightness, gain and bias, and both uniform and non-uniform scale, allowing the algorithm to better inpaint a larger set of completion tasks.

These extensions, however, are designed for single image completion and cannot take advantage of the additional 3D information available in 3D images. As such, one potential source patch (obtained with a given transformation) is as good as the next, so they employ a random search algorithm similar to that of the original PatchMatch algorithm. Such an approach 1) randomly chooses multiple values for the parameters of each transformation to test and 2) propagates any good matches using these random parameters to neighboring target patches. Thus, with each new transformation (and subsequently each new degree of freedom), the search space grows exponentially, requiring arbitrary limitations on the transformations to avoid unreasonable execution times.

With the use of stereo images, and their inherent 3D information, estimating the correct scaling between two patches can be done directly. Applying an estimation method can mitigate most of the necessity to select random scales when searching for potential source patches, allowing the algorithm to converge faster. This is the basis for our method.
3.2 Methods

This section presents our methods for estimating patch-to-patch scaling to extend our Stereo PatchMatch algorithm to be able to intelligently draw from scaled textures by using the inherent 3D information to estimate the scale instead of performing an entirely random search. We also extend this idea (as in [26]) to include nonuniform scale transformations (i.e., scales in the $x$ and $y$ directions differently) to address image foreshortening along planes.

3.2.1 Estimating Uniform Scale Transformation

In the absence of any additional information, other than RGB, it is reasonable to follow PatchMatch’s random approach to find (possibly) scaled source patches from the rest of the image. However, stereo images provide by their very nature the 3D structure contained in the scene. Taking advantage of this information provides a better way to direct the search towards improved possible matches, removing the necessity to randomly search the entire space.

Given real-world coordinates (or at least real-world depth) of two objects in a scene, it is fairly trivial to estimate the relative projected scaling $\phi_{t,s}$ between them using a simple ratio of their depths:

$$\phi_{t,s} = \frac{z_s}{z_t} \quad (3.1)$$

where $z_t$ is the depth of the target patch and $z_s$ is the depth of the source patch. Stereo Disparities are simply inversely proportional to depth as follows:

$$z_p = \frac{f B}{d_p} \quad (3.2)$$

where $z_p$ is the real world depth (or distance from the camera) at pixel $p$, $d_p$ is the disparity at pixel $p$, $f$ is the focal length of the camera, and $B$ is the baseline (or horizontal distance) between the two cameras (assuming a canonical configuration). It should be noted that although disparity can be estimated, using one of any number of algorithms, the focal length and baseline cannot be estimated...
Figure 3.2: Example scaled patches determined by the disparity computed from the stereo image pair. Illustrated here is a disparity map and example scaled source patches for a given target patch. Here $s_2$ is upsampled because it is farther from the camera than $t$ while $s_1$ is downsampled because it is closer. The scale factors $\phi$ between the source and target patches are determined by the disparities of the anchor positions (top left hand pixel of the patch). $\phi$ may be different in the $x$ direction than in the $y$ direction (i.e., $\phi_x$ and $\phi_y$ might not be equal).

without prior knowledge or camera calibration. for this reason, we can estimate the depth of each pixel only up to some unknown scale factor.

Now, using the estimated depth, we can estimate the scaling factor between two patches. Substituting $fB/d_p$ for our $z$ values in Eq. 3.1 and simplifying:

$$\phi_{t,s} = \frac{d_t}{d_s}$$

(3.3)

Notice that the unknown scale factors $f$ and $B$ cancel out. Thus we can estimate the scale between any two patches by simply using the ratio of their disparities. As illustrated in Figure 3.2, we use an anchor point, or reference point, to estimate the parameters for the whole patch. In other words, we assume one pixel of the patch, for our implementation the top left, is reminiscent of the entire patch.
3.2.2 Limiting the Random Search Step

Although using the disparities computed from the stereo images gives us an initial estimate of the scale between two patches, there is still some need to search around the estimated scale. Computed disparities are rarely accurate due to the nature of how they are computed. Compounding this is the fact that most algorithms do not calculate sub-pixel disparities, introducing slight errors in even the disparities it correctly computes. To address these issues, we alter the algorithm to randomly search in a window of scales around the originally estimated scale, thus allowing the algorithm to refine the estimated scales.

Instead of arbitrarily picking a fixed range of scales around the estimate, we use the disparities of the target and source patches to intelligently determine a reasonable range of scales to search. We do so as follows:

\[
S_{\text{window}} = \left[\frac{d_t + \delta}{d_t - \delta}, \frac{d_s - \delta}{d_s + \delta}\right]
\]  

(3.4)

where \(\delta\) is simply the range of expected error values in the disparity map. For example, assuming correct disparity maps up to whole integer values, \(\delta\) would be 1 to account for sub-pixel disparities. Thus, the range of scales is greater for patches that are farther away (where even slight errors in the disparities can greatly affect the computed scale transformation) and smaller for closer patches (where slight errors matter much less).

This windowed search is accomplished by limiting the range of scales the algorithm can randomly choose when considering a specific target and source patch pair. Barnes et al. [19] and subsequent papers simply choose each parameter in isolation bounded by a exponentially decreasing window around the current best patch. We, on the other hand, follow a similar method when choosing a random translation but limit the scale parameter according to the target and (potential) source anchor points’ disparity (as described above). Thus, we have a search similar to that of previous methods but use the estimated scale to focus the algorithm on more plausible, and in all likelihood better, scales.
3.2.3 Propagation Step

One of the essential steps of the original PatchMatch [2] algorithm is the propagation of good matches found in the random search phase to neighboring pixels, taking advantage of the inherent structure of images. As noted in [19], introducing new transformations into the algorithm complicates this phase slightly as neighboring patches are no longer related by a simple translation, making it necessary to transform the relative offsets between neighboring patches. To put it another way, the original PatchMatch algorithm propagated good matches by following the assumption that a good candidate for a given target patch $t$ is an offset of $t$’s neighbor’s current best nearest neighbor. With uniform frontoparallel transformations (i.e., translation only), the offset is simply the difference between $t$ and its neighbor. When you add other types of transformations, however, the offset must reflect this (i.e., a rotated source patch will rotate its neighbor with it).

We address this issue by following the methodology of [19]. Let $T(\text{NNF}(x))$ be the full transformation defined by $(x; y; \phi)$ where $x$ and $y$ are the translation in the $x$ and $y$ direction
respectively and \( \phi \) is a uniform scale factor. (We will relax this uniformity later.) The propagated candidate is then as follows:

\[
\text{NNF}(x - \Delta p) + T'(\text{NNF}(x - \Delta p)) \Delta p
\]

(3.5)

where \( \Delta p \) is simply the difference in position of the target patch and its neighbor. In simpler terms, and illustrated in Figure 3.3, we apply the target’s transformation to the neighbor’s position, giving a potential patch that is correctly offset and oriented from the source patch (i.e. producing the source’s neighbor according to the specified transformation). It should be noted that the source’s neighbor inherits all the parameters (aside from translation) from the source patch. In our case, this simply means that the scale is also propagated. We also use the computed scale at the propagated translation to simply add another potential source patch to the candidate pool.

### 3.2.4 Nonuniform Scale Transformations

The assumption of uniform scale is only valid when the surface being drawn from is frontoparallel to the camera. As soon as the plane is slanted, the farther portions of the plane are foreshortened due to the perspective warp which occurs during image acquisition. The more slanted the plane, the worse the foreshortening. This foreshortening cannot be modeled by uniform scale transformations, particularly when target and source patches lie on differently-slanted planes. It can, however, be approximated by non-uniform scale transformations. As such, we extend our random search phase to include non-uniform scale transformations.

We accomplish this by decomposing the uniform scale factor \( \phi \) into two separate directional scale factors, \( \phi_x \) and \( \phi_y \), and expanding the transformation space to \( (x, y, \phi_x, \phi_y) \). Though this does add another dimension to the search space, we can still apply our previous optimizations from Section 3.2.2. We do so by estimating the horizontal scale \( \phi_x \) as we estimated the uniform scale factor; that is, we use the disparities from horizontally offset cameras to 1) estimate the scale between the source and target patches and 2) determine a reasonable search window around that
scale. We then determine a vertical scale $\phi_y$ for each candidate $\phi_x$ by randomly choosing an aspect ratio $\theta_{xy}$ and applying that to $\phi_x$:

$$\phi_y = \theta_{xy} \phi_x \quad (3.6)$$

Thus, we still follow the methodology of previous extensions but, once again, focus the algorithm to more plausible, and in all likelihood better, scale factors. It should be noted that we limit the range of possible aspect ratios to reasonable values (between half and double size). We do this to avoid extreme foreshortening of the source patches.

### 3.2.5 Normalized and Scaled Disparities

An essential aspect of any inpainting algorithm is deciding which source material to blend into the hole. For patch-based methods, this involves choosing source patches that match the current target patches in the hole based on a selected similarity metric. The original PatchMatch paper [2] used a simple sum of the squared difference (SSD) of the RGB channels. Though this works well when matching texture, it fails to consider structure in the image.

In Chapter 2, we introduced the image disparity (i.e., a new channel) into the similarity metric to allow the algorithm to not only compare the potential source patch’s texture but its 3D structure as well. However, simply matching disparities, as we did in Chapter 2, folds depth into the metric, limiting the depths which the algorithm can consider.

To address this, we remove absolute depth from the similarity metric when considering scaled patches and compare only local relative-depth structure. This is accomplished by normalizing the disparities according to the disparity at the given patch’s anchor point as follows:

$$d'_n[i] = d_n[i] - d_n[n_{anchor}] \quad (3.7)$$

where $d_n[i]$ is the disparity of patch $n$ at position $i$ and $n_{anchor}$ is $n$’s anchor position. This transforms the disparity to be relative to its current patch anchor, essentially removing the inherent depth information while preserving the encoded 3D structure. Folding this idea into the disparity
distance metric, we get:

\[ \overline{D}_d = \sum_i (d'_t[i] - \phi_x d'_s[i])^2 \]  

(3.8)

which compares the relative structure of the target and source patches without preferring similar depths. Note that we scale \( d'_s \) by \( \phi_x \) or the horizontal scale factor between \( t \) and \( s \). Because of the inverse proportionality of disparity and depth, a difference of one at lower values of disparity is greater than a difference of one at higher values when converted to depth. To emulate this behavior, we scale the target relative disparity by the horizontal scale factor which encodes the relative scaling between the two patches.

To better understand the intuition of this scale factor (and to reduce computation) we substitute Eqs. 3.3 and 3.7 into Eq. 3.8 and simplify as follows:

\[
\begin{align*}
\overline{D}_d &= \sum_i ((d_t[i] - d_t[t_{anchor}]) - \frac{d_t[t_{anchor}]}{d_s[s_{anchor}]}(d_s[i] - d_s[s_{anchor}])(d_s[i] - d_s[s_{anchor}]))^2 \\
&= \sum_i (d_t[i] - d_t[t_{anchor}] - \frac{d_t[t_{anchor}]}{d_s[s_{anchor}]})d_s[i] + d_t[t_{anchor})^2 \\
&= \sum_i (d_t[i] - \frac{d_t[t_{anchor}]}{d_s[s_{anchor}]})d_s[i])^2 \\
&= \sum_i (d_t[i] - \phi_x d_s[i])^2
\end{align*}
\]  

(3.9)

Note that the anchor disparities fall out and we are left comparing the target disparity with a relatively scaled source disparity. This better shows what we are trying to accomplish. Essentially, we are moving the source patch to the same depth as the target patch in the scene and adjusting the disparities accordingly, i.e. by their relative scale. Thus, we can compare their relative 3D structure instead of actual (up to some scale factor) 3D location.

3.2.6 Building in Preferences for Downsampling Source Patches

We have found that allowing the PatchMatch algorithm to perform either directed or undirected searches for source patches at any scale can produce large flat or washed out regions in the resulting
Figure 3.4: Flat, washed-out regions introduced by unbounded scaling of source patches. Notice in (a) there are not only washed out regions along the left side of the fill, but there are blurred lines along the top of the fill. These both are caused by extreme upsampling. These artifacts are not present when using a penalty for such extreme upsampling (b).

When considering the algorithm, this makes sense. PatchMatch uses a simple sum-of-the-squared-distance metric to measure the similarity between a given source and target patch. This metric, though in general a good measure, has no real concept of texture. Thus, a flat source patch whose color is close to the average color of a less textured target patch can and will be selected over a similarly textured source patch that has more color variation, driving the solution to have (possibly less appealing) large flat regions.
These flat source regions, though they may be present in the image, can be artificially introduced by picking scaled source patches that must be upsampled, as demonstrated in Figure 3.4. When upsampling, or enlarging, discrete digital signals (such as digital photographs), high-frequency information cannot be reproduced (i.e. is lost). This technique results in blurred texture when applied to digital images. This is evident when viewing a small resolution photo in any photo viewing application with the zoom turned all the way up. If the source is upsampled enough, patches of even the most textured regions will become flat and featureless. This is what happens when PatchMatch attempts to draw from very small scales.

To address this problem, we encourage the algorithm to prefer to draw from non-scaled and downsampled patches. We accomplish this by including an additional cost term in the distance metric as follows:

\[
Dist(s, t) = Dist_{rgb}(s, t) + \lambda_d Dist_d(s, t) + \lambda_c cost(\phi_x, \phi_y)
\]

(3.10)

where \(Dist(s, t)\) is the sum squared difference between the specified channel(s), \(\lambda\) is a scale factor for the specified term, and \(\phi_x\) and \(\phi_y\) are the directional scale factors between patches \(s\) and \(t\) in the \(x\) and \(y\) directions respectively. We define \(cost(\phi_x, \phi_y)\) as follows:

\[
cost(\phi_x, \phi_y) = \begin{cases} 
\infty & \text{if } \phi_x > \phi_{max} \text{ or } \phi_y > \phi_{max} \\
\left(e^{\frac{[\phi_x] - 1 - [\phi_y] - 1}{e^{\phi_{max} - 1} - 1}} - 1\right) & \text{if } \phi_{max} >= \phi_x > 1 \text{ or } \phi_{max} >= \phi_y > 1 \\
0 & \text{otherwise}
\end{cases}
\]

(3.11)

where \(\phi_{max}\) is the maximum allowed upsample factor. Thus we disallow any patches with upsample factors greater than the maximum allowed scale factor and penalize on an exponential scale any upsample factor within the accepted range. This allows the algorithm to still draw from upsampled patches, but builds in a preference for non-scaled, downsampled, and only slightly upsampled patches.
3.2.7 Additional Gradient Term

As demonstrated by Darabi, et al. [26], slight color variations in the resulting fill are less noticeable than breaks in continuing edges. To address this, we also compare gradients within the potential source patches to those within their respective target patches. This is implemented by extending our distance metric to include comparison of the gradients in a manner similar to [26]:

\[
\text{Dist}(s, t) = \text{Dist}_{rgb}(s, t) + \lambda_d \text{Dist}_d(s, t) + \lambda_c \text{cost}(\phi_x, \phi_y) + \lambda_g \text{Dist}_g(\nabla s, \nabla t) \tag{3.12}
\]

where \( \text{Dist}_g(\nabla s, \nabla t) \) is the sum of the squared difference of the gradients of patches \( s \) and \( t \), and \( \lambda_g \) controls the weight given to this gradient term.

As noted by Darabi et al. [26], adding this gradient term to our distance metric boosts the high frequencies of local descriptors. In other words, instead of simply matching color when measuring texture similarity, we are also matching edges in the texture as well as the variation of the texture. Thus, we can better complete edges in the target region and match high-frequency content.

3.3 Results

Examples of the results of our extensions can be found in Figures 3.5 to 3.10. In Fig. 3.5 we remove the book and reproduce the wood grain underneath. Notice that we follow the direction of the wood grain while appropriately scaling the texture, providing a convincing result. As shown in Fig. 3.6, both the original PatchMatch algorithm [2][1] and our algorithm from Chapter 2 perform poorly in comparison.

The original PatchMatch algorithm, though not completing the two images in a stereoscopically consistent fashion, does correctly follow the direction of the texture. This is because it can draw liberally from source patches outside the mask. However, it has no concept of depth or scaling and thus copies, for example, smaller-scale texture from more distant patches, which without rescaling looks out of place in the target region. At the opposite extreme, our algorithm

\[1\) As implemented in Adobe Photoshop CS6\]
from Chapter 2, which uses only unscaled patches from the same depth as the respective target patches, produces texture that is appropriately scaled. However, preferring source patches at the same depth as the respective target patches severely diminishes the possible source patches for the algorithm to draw from and reduces the plausibility of the result. In this case, because the plane to be filled recedes vertically, the algorithm can only draw same-depth patches from the source regions to the left and right of the target. Because these areas are limited, this leads to undesirable strong repetition of these limited textures in the filled area. Since we have extended our algorithm to search for and blend source patches, scaled appropriately to match the depth of the target region, we overcome the limitations of both PatchMatch and the same-depth approach in Chapter 2.

Please note the visualization in Fig. 3.5c, which shows the relative scales of the source patches blended into the target region. Here, darker gray indicates upsampling, lighter gray indicates downsampling, and mid-level gray (a value of 128) indicates no scaling. There are several things to note about the scales presented in the visualization. First, the algorithm draws from the same scale where it can, such as on the left side of the target region. Second, moving from the edge of the target region to its center, in general, increases the amount of up- or down-sampling. Since the image is planar and on a slant, target patches towards the center of the target region have depths that are increasingly divergent from the target patches at the edges, particularly parallel to the direction of slant. Since there is little to no texture at the same depth for the algorithm to draw from (i.e., to the left and right of the target region) it is forced to draw from increasingly scaled textures from different depths. Also notice that the algorithm downsamples more than upsamples, particularly in the center of the target region. This is due to our built-in preference to avoid blurred or washed out results. Thus, the algorithm only upsamples texture along the top edge of the target region where the relative scale is close to one, incurring only a slight penalty. Finally, the patchiness, i.e., lack of smooth transitions of scales, is simply caused by the nature of the PatchMatch algorithm. Both translation and scale are passed when good matches are propagated to neighbors in the search phase,
Figure 3.5: Results of Book image. Here we remove the book and complete the texture underneath. The bottom row is a visualization of the relative scale between the target patches and their corresponding source patches. Darker gray indicates down sampling, lighter gray indicates upsampling, and mid-level gray (a value of 128) indicates no scaling. This image was taken from [27]
Figure 3.6: Comparison of our results against PatchMatch [2] and the method from Chapter 2. This image was taken from [27]
Figure 3.7: Results of the Window image. Here we remove the window on the wall. Also included is a comparison to PatchMatch and our algorithm from Chapter 2. This image was taken from the BoLD database [28].

and as such the algorithm tends to draw relatively large contiguous regions of scaled source material where possible.
In Fig. 3.7 we remove the window along the large brick wall. Notice that the texture of the wall has both a strong direction and strong pattern. More importantly, due to the perspective of the images, the horizontal lines of the bricks are divergent from left to right, so there is limited texture of the correct orientation for the algorithm to draw from. Regardless, our algorithm completes the texture in a pleasing way with only slight warping of the straight lines of the bricks.

Once again, both PatchMatch (3.7d) and our depth limited algorithm from Chapter 2 (3.7e) do not perform as well. PatchMatch (as before) follows the direction of the texture, but suffers once again from not being able to scale the texture to better fit the target region. As such, it attempts to merge large regions of source material, each with slightly different orientations and spacing of the lines of the texture. This, at best, causes moderate warping and, at worst, discontinuities of the lines. (See the top and the middle of the target region for examples of this.)

Our depth limited algorithm from Chapter 2 (3.7e) fairs better as it has ample source material to draw from immediately around the target region. However, due to the divergent nature of the texture, it has little to no source material of the correct orientation. As such, there are several discontinuities and slight warping artifacts introduced on borders where it blends larger regions of source material, (see the right side of the target region). Also of note, is the top left of the target region. Because the algorithm is depth-limited, it has no source material of similar color to draw from that is not better than the source material just above the target region. This drives the algorithm to continue the vertical lines down into the filled region leading to undesirable strong repetition of the limited texture in to filled area. Once again, since our algorithm is extended to search for and blend source patches that are scaled appropriately to match the depth of the target region, we overcome the limitations of both PatchMatch and the depth-limited approach in Chapter 2.

In Fig. 3.8 we remove the shadow on the brick wall using the graffiti removal mode discussed in Chapter 2. Notice that this set of images has more severe divergent lines in the region to be completed than the previous example. Also of note is the high frequency of the texture. These two aspects make this a hard image to complete. In particular, the latter aspect makes any inconsistency in the fill immediately apparent to the viewer. That being said, our extensions
Figure 3.8: Results of the Brick Wall image. Here we remove the shadow on the wall using the "Graffiti Removal" mode. Also included is a comparison of our results to PatchMatch and our depth limited algorithm from Chapter 2. This image was taken from the BoLD database [28].
Figure 3.9: Results of Tsukuba Image. This image was taken from the new Tsukuba Stereo Dataset. Here we remove the stapler and attempt to reproduce the texture underneath.

produce a nice result with only slight warping towards the top of the target region. For this example, both PatchMatch (3.8d) and our depth-limited algorithm from Chapter 2 (3.8e) have difficulty addressing the strong directional divergent nature of the brick texture on the wall. In particular, both draw texture into the target region that is not the correct orientation causing visible warping and inconsistencies in the brick pattern (i.e., breaks in the lines). For PatchMatch, this is because it does not scale the source patches appropriately. For our depth-limited algorithm, this is because it does not have texture with the correct orientation that has similar depth to the target region. Our extensions are better able to handle the strong directional divergent nature of the texture by drawing from and appropriately scaling texture from different areas of the image.

Figs. 3.9 and 3.10 show additional results of the extensions of our algorithm. In Fig. 3.9 we remove the stapler and complete the texture underneath. In Fig. 3.10 we remove the diffuse and metallic balls from the forefront of the image. Both of these results are pleasing with only slight
inconsistencies or warping in the target region. However, these results are only slight improvements over our previous methods as there is plenty of content to draw from at or around the same depth.

Figs. 3.11 to 3.13 show examples drawn from Chapter 2 run with our extensions. In these examples we remove the plant, basketball hoop, and teddy bear respectively. As is shown in the comparison of each figure, our algorithm produces results (c) at least as good, if not better, than our depth-limited algorithm from Chapter 2. This is simply due to the fact that introducing depth appropriate scaled source patches does not limit our extensions from drawing non-scaled source patches. As such, we can only improve our results from Chapter 2.

It should be noted that in the teddy bear example (Figure 3.13) we reproduce the structure of the roof on the birdhouse (Figure 3.13c) much better than in our results from Chapter 2 (Fig-
Figure 3.11: Our extensions run on the Plant images and comparison with our results from Chapter 2.

Figure 3.13b). The initial poor completion is driven by slight blurring in the disparity map which allows the depth limited methods of Chapter 2 to continue the roof of the birdhouse. Since our extensions are not depth limited, we are not as fragile to blurring in the disparity map, and as such do not continue the roof of the birdhouse beyond the limits of the disparity maps.

It should also be noted that adding the gradient term in our distance metric has driven some of the other additional improvements. In particular, this is evident in Figures 3.11 and 3.12. In Figure 3.11, where we remove the plant, we better complete the line of the roof along the front face of the barn. In Figure 3.12, where we remove the basketball hoop, we complete the texture of roof
Figure 3.12: Our extensions run on the Basket Ball images and comparison with our results from Chapter 2.

Figure 3.13: Our extensions run on the Teddy images and comparison with our results from Chapter 2.
continuing the line between the light and dark portions of the roof, where as the results from our previous chapter do not.

3.4 Conclusion

This chapter has presented a method that extends our depth-limited, frontoparallel, stereo-aware, PatchMatch algorithm to be able to search for and blend source patches scaled appropriately to match the depth of the target region. We have introduced a cost term to our distance metric to prefer non-scaled and downsampled source patches, as well as incorporate a gradient based distance term, similar to that of Darabi et al. [26], to better complete edges in the target region. The results of our extensions still demonstrate stereo-consistency of the completed regions as well as extend the class of images our algorithm can complete in a pleasing fashion.
Chapter 4

Conclusion

In this work, we have extended a state-of-the-art image completion algorithm to fill stereoscopic or 3D images by taking advantage of their inherent depth information and filling the target regions such that the result is consistent when viewed stereoscopically. In particular, we make the following contributions:

1. a coupled-PDE approach to pre-filling the disparity maps (computed or otherwise) in such a way that maintains mutual consistency, allowing these disparity estimates to aid further matching and blending,

2. an extension of PatchMatch [2] that provides cross image searching (driven by the estimated disparities) and a depth-sensitive comparison of target patches with potential source patches,

3. an extension to the weighted blending of matched source patches that gives preference to strong stereo correspondence, and

4. a directed approach to find geometrically scaled source patches by using the disparities to estimate the patch to patch scale transformation.

The result is a system that simultaneously attempts to maximize stereoscopic consistency as well as coherence of the target regions with respect to the rest of the source image.

4.1 Future Work

As stated in Chapter 3, there have already been several extensions [19, 24–26] to the original work of Barnes et al. [2] that introduce additional geometric and photometric transformations that the
algorithm can apply to find potential source patches. These additional transformations allow the algorithm to better complete a larger class of images with more visually compelling results. We did not include most of these transformations and enhancements in our algorithm as it was not the focus of our research. However, neither our enhancements nor the extensions of these other works preclude the integration of one with the other. In other words, a simple extension and improvement of our algorithm would include the new geometric and photometric transformations described in the aforementioned extensions.

It should also be noted that, like previous work, we assume all patches are front facing. This limits the ability of our algorithm to accurately reproduce texture on planes when drawing from other non-parallel planes in the scene. Though our work and that of Darabi et al. [26] attempt to alleviate this problem with the introduction of non-uniform scale patches, it often does not always produce pleasing results, especially when the planes do not slant in the same direction (for example the corner of a building). However, since we can determine the relative 3D structure (up to some unknown scale factor) of our scene in stereo images, we can approximate normals of the surfaces in the scene. These normals lend themselves very well to computing patch-to-patch homography transformations and could as future work be used to further extend the potential source patches our extended PatchMatch algorithm can draw from. An example of this extension would be transforming the texture on one side of a building (according to the 3D structure of the scene) to fit a target region on another (i.e., the corner of a building).
References


