Filling Large Holes in LiDAR Data By Inpainting Depth Gradients

David Doria
Rensselaer Polytechnic Institute
Troy, NY
doriad@rpi.edu

Richard J. Radke
Rensselaer Polytechnic Institute
Troy, NY
rjradke@ecse.rpi.edu

Abstract

We introduce a technique to fill large holes in LiDAR data sets. We combine concepts from patch-based image inpainting and gradient-domain image editing to simultaneously fill both texture and structure in a LiDAR scan. We discuss the problems with directly inpainting a depth image, and present a solution to this problem based on inpainting the depth gradients. Once the inpainted depth gradients are obtained, we use an image reconstruction technique to obtain the final 3D scene structure. We present several real-world examples of this technique with excellent results.

1. Introduction

Light Detection and Ranging (LiDAR) is used in surveying, architecture, and visual effects to directly gather three-dimensional information about an environment. A LiDAR scanner collects point cloud data by inferring distances to scene surfaces using the speed of light and the time taken by laser pulses to reflect off surfaces in the scene (or the phase differences of continuous-wave signals). While devices for range scanning have recently become more widely available, there is still much work to be done to improve the experience of exploring such data sets.

In this paper, we are particularly interested in filling holes in LiDAR data. Since the LiDAR laser cannot penetrate opaque objects, a "shadow" appears behind every foreground object in the scene, as illustrated in Figure 1. These shadows leave very large holes, or collections of missing points, in the background structure of nearly every real-world scan. Such holes are very distracting to the viewer when exploring LiDAR scans in 3D, and significantly reduce their ease of visualization. As a result, LiDAR data only seems "complete" when viewed from the original acquisition viewpoint.

There has recently been substantial research on the image inpainting problem in standard RGB images. That is, given an image and a specified foreground region to remove (the "hole"), we want to determine new plausible colors at the hole pixels such that the background appears to seamlessly continue through the hole without introducing visible artifacts. Image editing software like Adobe Photoshop now routinely includes tools to perform this type of inpainting operation for removing unwanted foreground objects or repairing damaged photographs. However, the same problem of hole filling in point cloud data is significantly less studied.

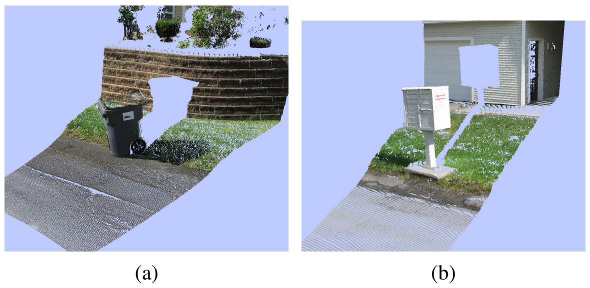


Figure 1. Examples of LiDAR shadows. We can see there are large portions of the scene background that were occluded from the scanner's viewpoint.

In this paper, we present a new algorithm to synthesize realistic information inside large holes in LiDAR data. This new information — both in terms of color and 3D structure — corresponds to a plausible explanation of what could have been present at points in the scene that were not observed during the acquisition process. We show that filling holes allows the scan to be naturally viewed from positions other than the original acquisition viewpoint. Our algorithm leverages and extends existing greedy patch-based image inpainting techniques and gradient-domain image editing techniques to create a novel 3D inpainting algorithm. In particular, we work in the depth gradient domain to intelligently copy structure from elsewhere in the scene into the hole. We then reconstruct the 3D scene by solving a variational problem resulting in a Poisson equation.

This paper is organized as follows. In Section 2 we discuss prior work on filling holes in 3D data, and introduce the key concepts from image inpainting and editing upon which our new algorithm is based. In Section 3, we describe an intuitive extension of image inpainting to depth image inpainting, and discuss why it does not produce satisfactory results. In Section 4, we discuss how to mitigate the problems demonstrated in Section 3 by working in the depth gradient domain. In Section 5, we show several real-world examples of combined texture and structure inpainting in real LiDAR datasets. These examples highlight the benefits of our proposed algorithm. Section 6 concludes the paper with discussion and ideas for future work.

2. Related Work

There have been several previous approaches to filling holes in 3D data sets. Sharf et al. [9] proposed a hole-filling algorithm for point sampled surfaces that used a coarse-to-fine approach. First, the rough geometry in the hole was estimated, then detailed structure was copied from elsewhere in the model to refine the initial estimate. Finally, a series of elastic warps was applied to ensure the copied patch matched the surrounding hole. While acceptable results were shown, the authors noted several problems with this approach. These problems included the high number of degrees of freedom in aligning two point-sampled surface patches in 3D, the difficulty of defining a coordinate system in which to work, and the boundary of a hole being ill-defined in a point-sampled surface.

Park et al. [6] extended this work to point clouds with associated colors. In the final step, rather than applying elastic warping transformations, a height field is formed and a Poisson equation is solved to join the copied patch of points smoothly to the points defining the hole. Becker et al. [1] also proposed copying 3D patches of structure directly into a 3D hole. This technique relies on solving a computationally complex 3D registration problem at each iteration of the inpainting. Additionally, there is no guarantee that the inpainted 3D points correspond to a reasonable depth map from the scanner's original perspective. In this paper, we take a similar approach to copying structure from elsewhere in the scene into the hole, but since we work in the depth image gradient domain rather than directly in 3D, the complexity of the process is greatly reduced and we avoid the problems mentioned above.

Other researchers have proposed to apply inpainting techniques directly to depth or disparity images. Salamanca et al. [8] projected a mesh with small holes into a depth image, then used inpainting techniques to repair the holes in the depth image. Weghorn et al. [13] created local height fields on a tangent plane to the points surrounding a hole, and then inpainted the resulting hole in this height field to reconstruct missing 3D geometry. Stavrou et al. [10] applied 2D image repairing algorithms to the depth image of a LiDAR scan. Their technique was demonstrated successfully, but only on scenes containing simple objects. He et al. [4] remove small objects by explicitly applying depth constraints to guide the inpainting. An example of one such constraint is that the depth values that are filled must always be greater than the depth values of the object to be inpainted. Wang et al. [12] and other researchers have studied the problem of filling holes in pairs of stereo images. The nature of the holes in these data sets is very different from the data that we are interested in in this paper — they are typically very thin halos or slivers around objects. As we will show in Section 3, techniques for filling these kinds of holes work poorly for large holes in LiDAR scans of complex real-world scenes.

Wei and Klette [14] perform an extensive analysis of the problem of reconstructing a surface from its gradients. They note several potential problems with this procedure. First, a gradient field may not correspond to a unique surface. Second, they detail the theoretical requirement for the integrability of a gradient field. Though we are aware of these problems, in practice we have found that our technique does not seem to be affected by them.

In this paper we draw from two distinct sets of prior work; patch-based image inpainting and gradient-domain image editing. In the remainder of this section, we discuss key concepts from these areas that we apply in our contributions.

2.1. Patch-Based Image Inpainting

Image inpainting is the problem of filling new colors into a specified region of an image $I(x, y)$, so that the result is visually convincing. We call this region the **hole**, and denote it by Ω. The hole is usually caused either by corruption of originally valid data or by occlusion in the scene itself. For example, a photograph may have been folded or torn, leading to missing data, or an undesirable object may occlude part of a desired background. This problem is illustrated in Figure 2. We want to determine new pixel colors in the specified region so that the object is removed but it is imperceptible to an observer that the image was modified.

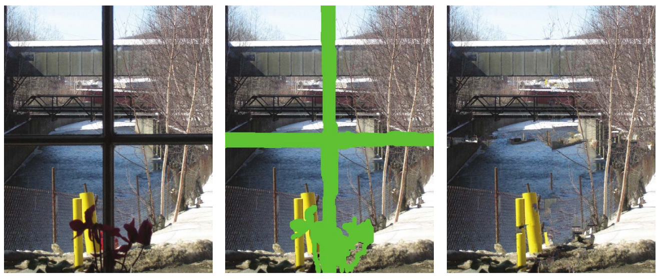


Figure 2. A demonstration of image inpainting. (a) The original image. (b) The region to inpaint is shown in bright green. The goal in this example is to remove the window frame from the image. (c) The inpainted image. If an observer was presented with only this image, they would likely not notice that it had been modified.

A well-known approach to this problem, exemplified by Criminisi's algorithm [3], is to copy patches of pixels from elsewhere in the image into the hole. These patches should continue the surrounding image colors into the hole without introducing visual artifacts. Such techniques are collectively referred to as "exemplar-based" or "patch-based" inpainting methods. The idea is shown in Figure 3.

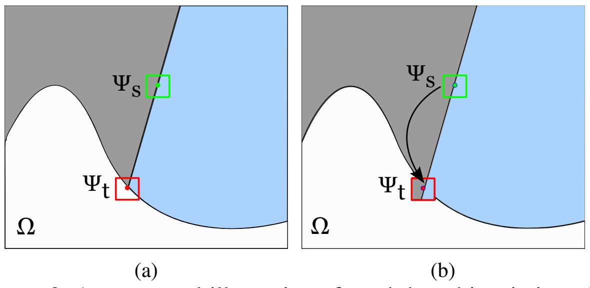


Figure 3. A conceptual illustration of patch-based inpainting. (a) The region of the image Ω to be filled is shown in white. A target patch ψ_T and a good candidate source patch ψ_S are indicated. (b) The source patch colors are copied into the target patch. The linear structure (the boundary between the blue and gray regions) has been continued appropriately.

Candidate patches that consist entirely of pixels from the known region of the image are referred to as **source patches**, Ψ_s. Likewise, a **target patch**, Ψ_t, is any patch whose center is on the current hole boundary. A typical patch-based inpainting algorithm proceeds as follows. First, we determine which target patch to fill. The hole is typically filled from the outside in, with an attempt to first fill patches that continue linear structures in the image. Next, we search for the source patch that best matches the selected target patch. We then copy colors from the source patch into corresponding pixels of the target patch that intersect the hole. Finally, we update the hole and the hole boundary and repeat the process until the target region is empty.

The problem of finding the best patch can be stated as a minimization problem as shown in Equation 1.

$$\underset{\Psi_s \in I-\Omega}{\text{argmin}}\ d(\Psi_s, \Psi_t) \tag{1}$$

That is, we search for a patch in the known region of the image with the minimum score according to some patch distance function, $d(\Psi_s, \Psi_t)$. Typically, this distance is computed as the sum of the squared differences of RGB color vectors at corresponding non-hole pixels in the two patches.

2.2. Reconstructing an Image from its Gradients

There has been much recent research interest in gradient-domain techniques for image processing. Pérez et al. [7] applied gradient-domain techniques to convincingly copy large regions of one image into an entirely different image. Bhat et al. [2] presented a generalized framework for gradient-domain image filtering that can produce several types of manipulated images. Such techniques rely on the problem of reconstructing an image from its gradients, which we summarize below.

We begin with an intensity image $I(x, y)$, a target region Ω inside the image, and the desired gradients $G_x(x, y)$ and $G_y(x, y)$ inside Ω. We wish to reconstruct new intensities $I^*(x, y)$ inside Ω subject to the constraint that I and I^* agree on the hole boundary $\partial\Omega$. That is, we want to solve

$$\begin{aligned} \min_{I^*(x,y)\in\Omega} \iint_\Omega & (I_x^*(x,y) - G_x(x,y))^2 \\ & + (I_y^*(x,y) - G_y(x,y))^2\, dx\, dy \\ s.t.\ \ & I^*(x,y)|_{\partial\Omega} = I(x,y)|_{\partial\Omega} \end{aligned} \tag{2}$$

Using the Euler-Lagrange equation from variational calculus, it can be shown that the solution to Equation 2 satisfies Equation 3 below:

$$\begin{aligned} \nabla^2 I^*(x,y) = \text{div}\ (G_x(x,y), G_y(x,y))\ \ \forall (x,y) \in \Omega \\ s.t.\ I^*(x,y)|_{\partial\Omega} = I(x,y)|_{\partial\Omega} \end{aligned} \tag{3}$$

where ∇^2 represents the image's Laplacian and div is the divergence of a 2D vector field. We discuss the solution to a discretized version of this equation in Section 4, which results in a simple system of linear equations. The approach is applied to color images by processing each channel independently.

3. A Framework for Depth Inpainting

In this section, we discuss an extension to patch-based inpainting to fill holes in depth images. We show why filling holes directly in the depth image is not appropriate, and propose a solution to these problems in Section 4.

Modern LiDAR scanners typically produce a grid of colored 3D points. That is, at each point, we know the depth (i.e., distance) from the scanner, as well as an RGB value associated with the point (usually obtained by a collocated camera). We can view the resulting dataset as a 4-channel RGBD image (Red, Green, Blue, Depth) over a 2D pixel grid. As a first attempt at inpainting holes in this type of data, we could simply extend the technique described in Section 2.1 to operate on these RGBD images. In this case, Ψ_s and Ψ_t are patches of 4D vectors defined over the 2D domain (x, y). The distance function $d(\Psi_s, \Psi_t)$ in Equation 1 is defined as the sum of squared differences of corresponding RGBD vectors at non-hole pixels. Before comparison, we normalize the channels in each RGBD image so that the standard deviations of the RGB channels sum to the standard deviation of the depth channel.

To determine the order of the patch filling, we use the priority term defined by Criminisi et al. [3]. We defer to [3] for a full explanation, but the intuition is that we prioritize patches based on two values. First, patches that are near the original hole boundary should be filled first, as we are more confident that the known data in these patches is accurate. At the same time, we attempt to continue linear structures in the image by filling patches where the image isophote directions (the vectors perpendicular to the gradient vectors) are strong and aligned with the normals of the hole boundary.

Patch-based image inpainting relies on the idea that a patch that "looks like" the one we would expect to appear in the hole exists somewhere else in the image. In many RGB images, this is indeed the case. However, in depth images, this is typically not the case. For example, consider Figure 4, in which a hole interrupts a planar surface seen from above. Though several patches with the correct structure are available, no source patch exists that has the correct depth value at the interior of the hole.

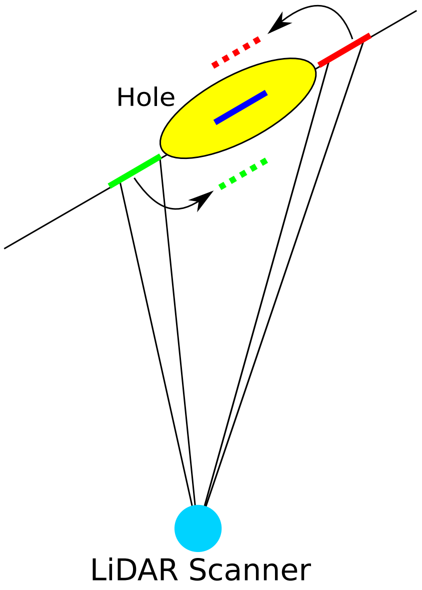


Figure 4. An illustration of why direct depth inpainting fails. We wish to fill the hole (yellow) by copying an existing depth patch to the location of the blue patch. Unfortunately, the closest patches in the depth image, though having the structure we need, do not occur at the appropriate depth. Using the green patch would result in 3D structure in front of the appropriate location (the green dashed patch), and using the red patch would result in 3D structure behind the appropriate location (the red dashed patch).

An example of this problem in a real data set is shown in Figure 5. We see that many of the patches that were copied were actually located at incorrect depths.

Another approach one might investigate is inpainting the RGB values as usual and then finding the smoothest possible depths to fill the hole. While this may work on very small holes or holes that appear in planar surfaces, in Figure 6 we show that the result is usually unacceptable for large holes with complex backgrounds. In Figure 6b the hole in the depth image has been smoothly filled, resulting in very incorrect 3D structure, shown in Figure 6d.

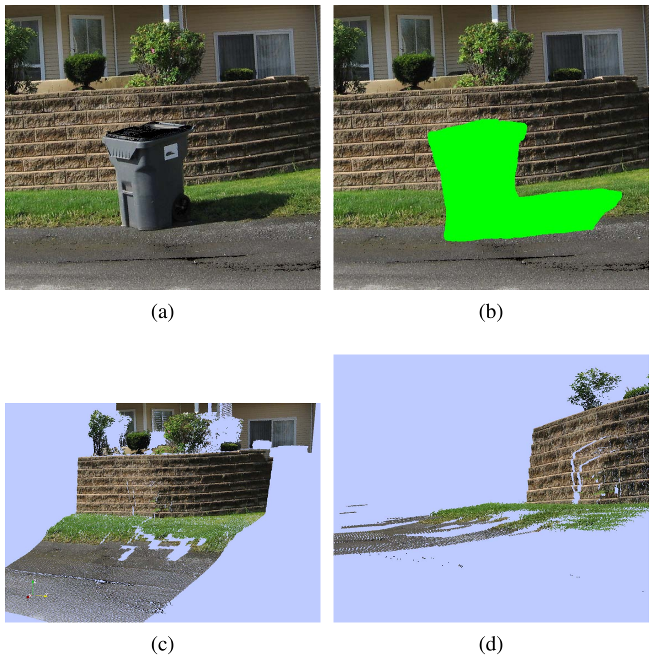


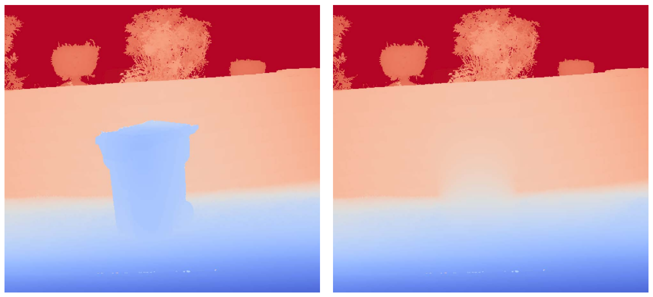
Figure 5. A demonstration of the result of directly inpainting a depth image. (a) The image associated with the original LiDAR scan. (b) The region to inpaint is indicated in bright green. (c) The structure after inpainting directly in the RGBD image. (d) A side view showing many patches at incorrect depths.

4. Inpainting 3D Structure Using Depth Gradients

To prevent the problem of copying depth patches which have very similar structure but different absolute depth values, we instead work in the depth image gradient domain. We observe that depth patches with the correct structure to complete the hole are typically available in the known region of the image.

Similar to the RGBD images discussed in Section 3, we now construct a 5-channel image consisting of the RGB values, as well as the x and y components of the depth image gradient $D_x(x, y)$ and $D_y(x, y)$. We normalize the channels of this image so that the sum of the standard deviations of the R, G, and B channels equals the sum of the standard deviations of the D_x and D_y channels.

We perform the inpainting procedure in these 5-channel $RGBD_xD_y$ images, where the patch distance function now operates on 5D vectors defined over the 2D pixel domain. After inpainting the hole by cutting and pasting patches, we can color the new scene points directly using the first three channels of the resulting patches. However, an additional


(a) (b)

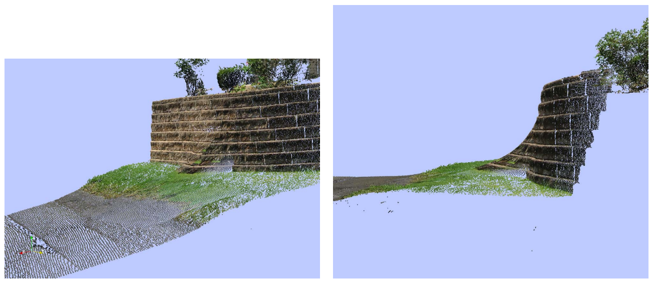

(c) (d)

Figure 6. A demonstration of smoothly filling a hole in the depth image. (a) The depth image corresponding to Figure 5a (blue = close to the scanner, red = far from the scanner). (b) The resulting depth image after removing the trashcan and filling the hole with a smooth surface. We note that the sharp edge at the boundary between the wall and the ground is not preserved. (c) The resulting 3D structure. (d) A side view of the 3D structure. It is clear that this result is unacceptable, since a surface that does not make sense in the scene has been created.

step is now required to obtain the depth values for the new points.

We have the desired depth gradient $(D_x(x,y), D_y(x,y))$ inside the hole, but what we need is the actual depth in the hole, $D(x,y)$. To perform this reconstruction of the depth image from its gradients, we apply the techniques described in Section 2.2. Here, $G_x(x,y)$ and $G_y(x,y)$ in Equation 2 are exactly our inpainted depth image gradients. We know the depth values immediately outside of the hole, so these serve as the boundary condition for the reconstruction problem. The system of equations that must be solved for $D^*(x,y)$, the reconstructed depth image, is shown in Equation 4.

$$\begin{aligned}\nabla^2 D^*(x,y) = &\frac{\partial D_x}{\partial x}(x,y) + \frac{\partial D_y}{\partial y}(x,y) \;\; \forall (x,y) \in \Omega \\ s.t. \;\; &D^*(x,y)|_{\partial\Omega} = D(x,y)|_{\partial\Omega}\end{aligned} \tag{4}$$

Explicitly, a pixel whose 4-neighbors are fully inside the hole generates Equation 5.

$$\begin{aligned}D^*(x+1,y) + D^*(x-1,y) + D^*(x,y+1)& \\ + D^*(x,y-1) - 4D^*(x,y)& \\ = \frac{\partial G_x(x,y)}{\partial x} + \frac{\partial G_y(x,y)}{\partial y}&\end{aligned} \tag{5}$$

A pixel that has at least one 4-neighbor outside the hole generates an equation similar to Equation 6, which shows the case where the pixel $(x+1,y)$ is outside the hole:

$$\begin{aligned}D^*(x-1,y) + D^*(x,y+1) + D^*(x,y-1) - 4D^*(x,y)& \\ = \frac{\partial G_x(x,y)}{\partial x} + \frac{\partial G_y(x,y)}{\partial y} - D(x+1,y)&\end{aligned} \tag{6}$$

Thus, if there are N pixels in the whole, Equations 5-6 can be written as a matrix equation $Ad = b$, where A is a known $N \times N$ matrix, b is a known $N \times 1$ vector, and d is an unknown $N \times 1$ vector containing the depths to be determined. Since A is extremely sparse, containing 5 or less non-zero entries per row, this system can be solved very efficiently, even for large N.

Once we have solved for the depths in the hole, we construct the new 3D points by placing points along the original LiDAR rays at the distances prescribed by the new depth image. We describe the entire procedure algorithmically in Algorithm 1.

Algorithm 1 FillLargeHoles(LiDAR scan)

```
Construct the $RGBD_xD_y$ image from the source data
Normalize the $RGBD_xD_y$ image
Inpaint the $RGBD_xD_y$ image:
while Hole pixels remain do
    $TargetPatch \leftarrow SelectTargetPatch()$
    $SourcePatch \leftarrow FindMatch(TargetPatch)$
    Copy SourcePatch into TargetPatch
    Update the hole and hole boundary
end while
Extract the inpainted $D_x$ and $D_y$ inside the hole
Reconstruct the hole depths $D^*$ by solving Equations 5-6
Create the new 3D points at the prescribed depths along
    the original LiDAR rays
```

5. Experiments

In this section, we demonstrate our algorithm on several real-world data sets. The results are convincing, even with complicated backgrounds. The resulting hole completions we obtain appear as if the points had been acquired by the actual LiDAR scanner, making the completions look very natural.

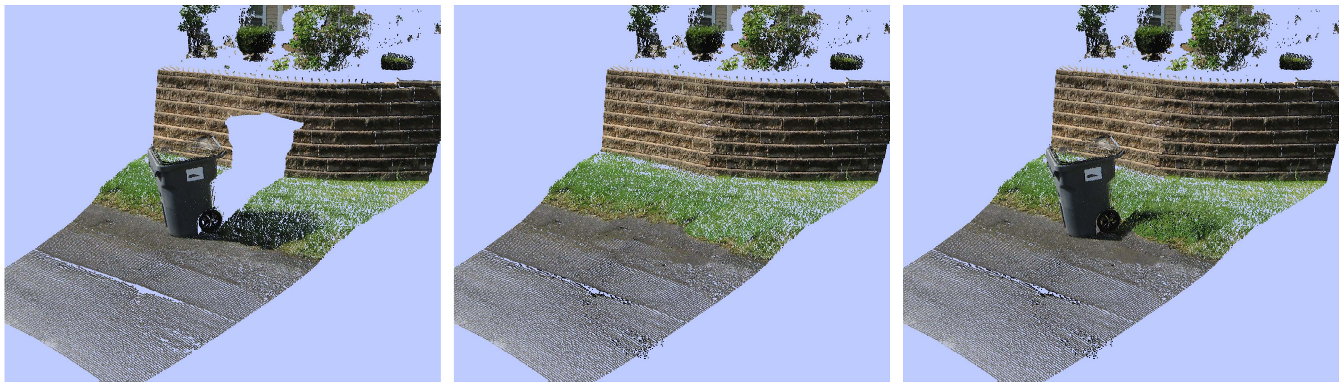


Figure 7. (a) A LiDAR scan of a trashcan in front of a background consisting of concrete, grass, and a brick wall. (b) The inpainted 3D structure behind the trashcan. (c) A composite of the trashcan with the structure behind it.

In Figure 7, we demonstrate our algorithm on a LiDAR scan of a trashcan in front of a brick wall. The hole left by the removal of the trashcan spans three textures including concrete, grass, and brick. The algorithm is able to successfully fill this large hole with convincing color and structure.

Figure 8(a)-(b) shows the depth gradient image before and after inpainting. The inpainted gradient image looks like the gradient field we would expect if the trashcan had not been present in the scene. Figure 8(c)-(d) shows the depth image before and after reconstruction from the inpainted depth gradient image, indicating that the results are realistic.

In Figure 9, we demonstrate the algorithm with a less regular background. In this LiDAR scan, several electrical boxes are present in a grassy field, with a very complicated background of bushes and trees. Again, we show that the algorithm was able to fill in convincing color and structure in both the smooth ground region as well as the noisy region of trees.

In Figure 10, we show a LiDAR scan of a mailbox with a building in the background. The mailbox occludes multiple linear structures, and a harsh shadow is present on the building. Despite these challenges, the algorithm is able to produce a satisfying result.

A summary of the data sets shown throughout this paper, including image size, hole size, and timing of the entire LiDAR inpainting process, is provided in Table 1. The experiments were performed on a computer with an Intel Core 2 Duo 3 GHz CPU.

About 75% of the time is spent in the inpainting process, while the remaining time is spent reconstructing the depth image from its gradients.

6. Discussion and Future Work

We presented an algorithm to fill large holes in LiDAR data. We inpaint the data in the depth gradient domain, then reconstruct geometry in the original scanner coordinate system. The experiments demonstrate that the method can plausibly fill large holes, making the data easily viewable from multiple viewpoints without perceptual artifacts.

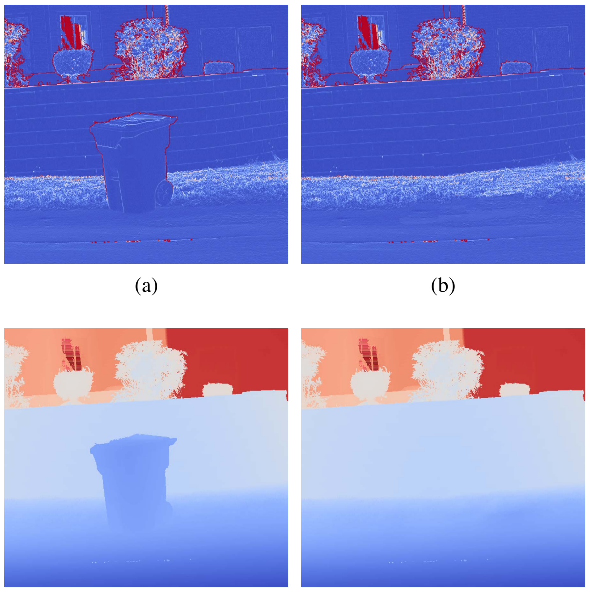


Figure 8. A demonstration of our depth gradient inpainting approach. (a) The magnitude of the original gradient (blue = low gradient magnitude, red = high gradient magnitude). (b) The magnitude of the inpainted depth gradient. (c) The original depth image (blue = close to the scanner, red = far from the scanner). (d) The depth image reconstructed from the inpainted depth gradient. We note that the structure of the corner between the wall and the ground was successfully preserved.

We note that our LiDAR inpainting technique is more sensitive to poor patch choices than a standard image inpainting problem. For example, in Figure 11 we show a

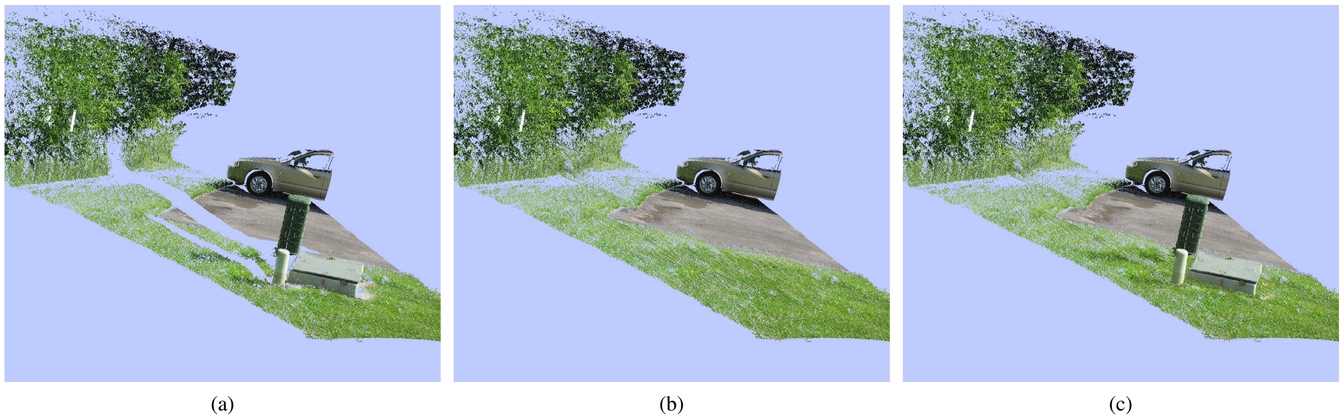


Figure 9. (a) A LiDAR scan of several electric boxes in a grassy field, with a complex background consisting of bushes and trees. (b) The inpainted scene structure behind the electric boxes. (c) A composite of the electric boxes with the inpainted scene behind them.

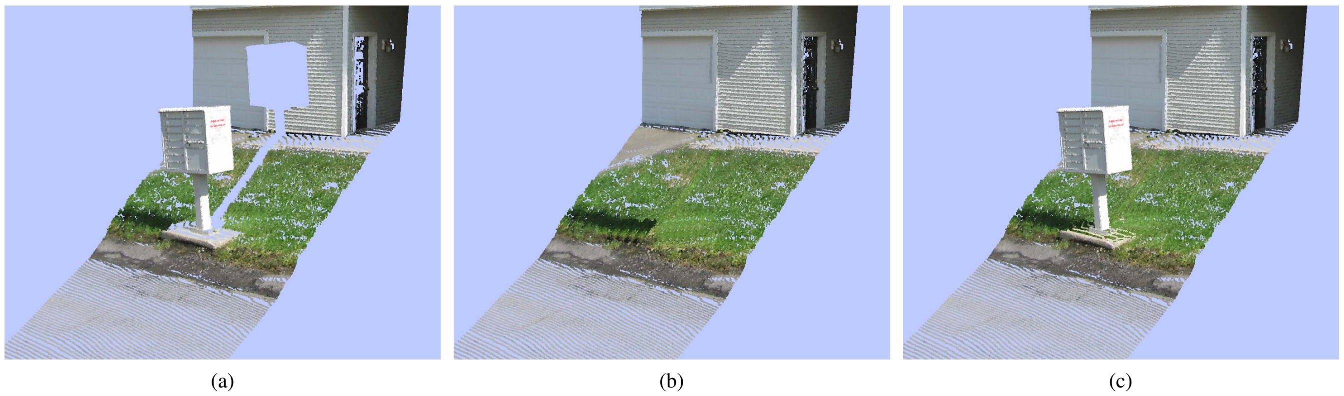


Figure 10. (a) A LiDAR scan of a mailbox with a building in the background. The LiDAR shadow interrupts multiple linear structures in the background. (b) The inpainted scene structure in the LiDAR shadow. (c) A composite of the mailbox with the background filled behind it.

Table 1. A summary of the data sets shown throughout this paper.

Data set	Image size	Hole size (pix)	Total time
Mailbox	459×489	30171	1m5s
Electric boxes	688×478	45434	2m19s
Trashcan	572×517	42734	1m59s
Air conditioners	400×496	13709	23s

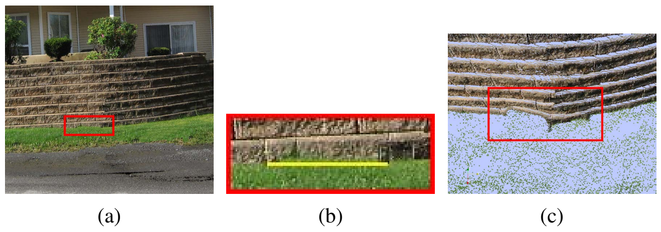


Figure 11. A demonstration of sensitivity to error in the inpainting. (a) An image of the inpainted colors of a scene. The red rectangle indicates a region in which a small error has occurred in the inpainting. (b) A zoomed-in version of the red rectangle from (a), showing that several rows of pixels were filled with grass when they should have been filled with brick to correctly continue the wall/ground boundary. (c) A 3D view of the resulting error in the reconstructed LiDAR points.

case where the image completion would have been deemed perfectly acceptable, but the resulting 3D structure exhibits strange behavior.

In this case, the pixels of grass above the yellow line in Figure 11b should have actually been filled with brick. In the image alone, the human visual system can hardly perceive this error. However, in the reconstructed 3D scene, the error manifests as a warp in the wall.

Also, as with image inpainting, there are cases where we cannot expect the algorithm to compute a reasonable completion. For example, Figure 12 shows a LiDAR scan of a corner of a building, with air conditioning units on the ground. When we attempt to inpaint these air conditioners, we have to construct the intersection of two walls and the ground that does not appear elsewhere in the scene. The way that these multiple linear structures should be joined inside the hole is ambiguous. This problem might be mitigated by allowing the user to draw guidelines inside the hole to indicate the way linear structures should be merged, as in [11]. Another potential difficulty is having very limited structure to either side of the hole, so patches are repeated multiple times. For example, in Figure 12b we can see that there is only a very small piece of the image to the left of the hole at the wall/ground boundary. The results in many of these hard cases could potentially be improved by substituting the greedy inpainting algorithm we used here with a globally optimal technique (e.g. [5]), at the cost of slower performance.

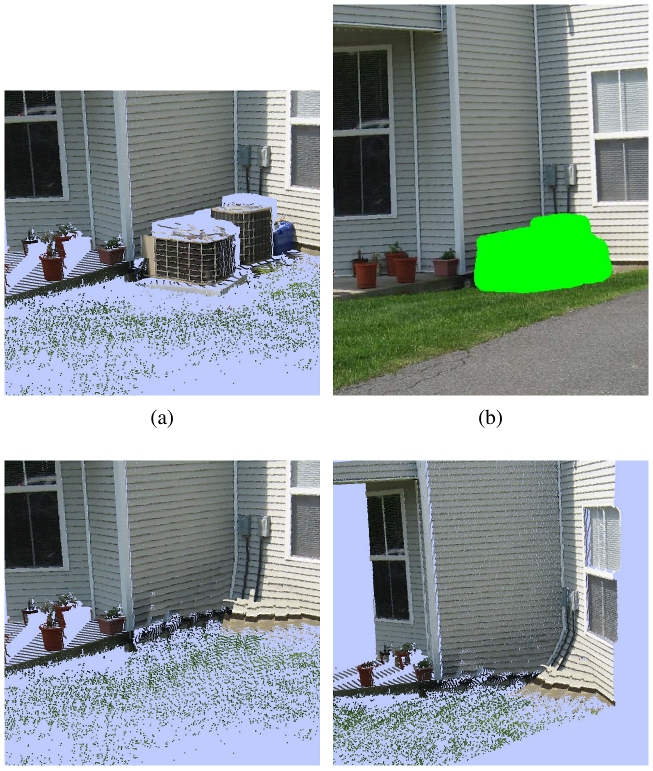

(a) (b) (c) (d)

Figure 12. A data set for which we do not expect a good result. There is no information to guide the algorithm to fill the corner that results from the intersect of the two walls and the ground. (a) An image of the LiDAR scan.(b) The region to inpaint. (c) The holes appear to be filled correctly when hidden by the objects. (d) Visible artifacts are present in the resulting scene, including a warped corner and wall.

Finally, we could apply a similar technique to achieve a different goal. Rather than filling large holes in LiDAR data by copying the gradients from elsewhere, we could correct sampling inconsistencies in LiDAR scans introduced in places where the laser is nearly parallel to scene surfaces or passes through spotty occlusion like foliage. By redistributing the depth gradient values in particular regions of the scan, we might be able to resample the 3D geometry using the same technique presented here.

References

[1] J. Becker, C. Stewart, and R. J. Radke. LiDAR inpainting from a single image. In *International Conference on 3-D Digital Imaging and Modeling (3DIM)*, 2009. 2

[2] P. Bhat, C. L. Zitnick, M. Cohen, and B. Curless. GradientShop: A Gradient-Domain Optimization Framework for Image and Video Filtering. *ACM Transactions on Graphics*, 29(2), 2010. 3

[3] A. Criminisi. Object removal by exemplar-based inpainting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003. 3

[4] L. He, M. Bleyer, and M. Gelautz. Object Removal by Depth-guided Inpainting. In *Austrian Association for Pattern Recognition*. 2

[5] N. Komodakis and G. Tziritas. Image completion using global optimization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006. 8

[6] S. Park, X. Guo, H. Shin, and H. Qin. Shape and appearance repair for incomplete point surfaces. In *IEEE International Conference on Computer Vision*, 2005. 2

[7] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318, 2003. 3

[8] S. Salamanca, P. Merch, A. Adan, C. Cerrada, and E. Perez. Filling holes in 3D meshes using image restoration algorithms. In *International Symposium on 3D Data Processing, Visualization, and Transmission*, 2008. 2

[9] A. Sharf, M. Alexa, and D. Cohen-Or. Context-based surface completion. *ACM Transactions on Graphics*, 23(3):878–887, Aug. 2004. 2

[10] P. Stavrou, P. Mavridis, G. Papaioannou, G. Passalis, and T. Theoharis. 3D object repair using 2D algorithms. 2006. 2

[11] J. Sun and J. Jia. Image completion with structure propagation. *ACM Transactions on Graphics*, 24(3):861–868, 2005. 8

[12] L. Wang, J. Hailin, R. Yang, and M. Gong. Stereoscopic Inpainting : Joint Color and Depth Completion from Stereo Images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008. 2

[13] H. Weghorn, H. W. Ed, M. Kochanowski, P. Jenke, and W. Straßer. Analysis of texture synthesis algorithms with respect to usage for hole-filling in 3D geometry. In *Annual Meeting on Information Technology and Computer Science*, 2008. 2

[14] T. Wei and R. Klette. On depth recovery from gradient vector fields. In *Algorithms, architectures and information systems security*, pages 75–95. 2009. 2