LiDAR Inpainting from a Single Image

Jacob Becker[1], Charles Stewart[1] and Richard J. Radke[2*]
[1]Department of Computer Science
[2]Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute, Troy, NY USA 12180
{beckej,stewart}@cs.rpi.edu, rjradke@ecse.rpi.edu

Figure 1. High-detail 3D structure is incrementally generated by our automatic image-guided LiDAR hole-filling algorithm.

Abstract

Range scans produced by LiDAR (Light Detection and Ranging) intrinsically suffer from "shadows" of missing data cast on surfaces by occluding objects. In this paper, we show how a single additional image of the scene from a different perspective can be used to automatically fill in high-detail structure in these shadow regions. The technique is inspired by inpainting algorithms from the computer vision literature, intelligently filling in missing information by exploiting the observation that similar image regions often correspond to similar 3D geometry. We first create an example database of image patch/3D geometry pairs from the non-occluded parts of the LiDAR scan, describing each uniform-scale region in 3D with a rotationally invariant image descriptor. We then iteratively select the best location on the current shadow boundary based on the amount of known supporting geometry, filling in blocks of 3D geometry using the best match from the example database and a local 3D registration. We demonstrate that our algorithm can generate realistic, high-detail new geometry in several synthetic and real-world examples.

1. Introduction

Time-of-flight LiDAR (Light Detection and Ranging) scanners are capable of capturing highly accurate and detailed geometry of real-world objects. As a result, LiDAR is increasingly used in surveying, architecture, cultural resource management, archaeology, and other fields where accurate 3D models are useful. However, missing data in the form of holes or "shadows" produced by occluding foreground surfaces are a common artifact of the scanning process.

In this paper, we show how a single additional image of the scene from a different perspective can be used to automatically fill in high-detail structure in these shadow regions. We first create an example database of image patch/3D geometry pairs from the non-occluded parts of the LiDAR scan, describing each uniform-scale region in 3D with a rotationally invariant image descriptor. We then iteratively select the best location on the current shadow boundary based on the amount of known supporting geometry, filling in blocks of 3D geometry using the best match from the example database and a local 3D registration. Figure 1 illustrates an example of our LiDAR inpainting algorithm, showing successive steps of high-detail hole filling.

The rest of the paper is organized as follows. In Section 2 we review related work on filling holes in range data, with or without additional images. Section 3 defines the input and output of the problem and describes data preprocessing steps. Section 4 provides a taxonomy of holes that typically arise in LiDAR data. Section 5 describes our overall inpainting algorithm, including the generation of the example database, the definition of the patch fill order, and the fill-in algorithm itself. Section 6 presents experimental results on both simulated and real datasets to demonstrate the technique. Section 7 concludes the paper with discussion and ideas for future work.

*This work was supported in part by the DARPA Computer Science Study Group under the award HR0011-07-1-0016.

2. Related Work

There has been a substantial amount of work on repairing small to medium-sized holes in range data. For example, Stavrou et al. [16] adapted 2D image repair algorithms to 3D data by treating the depth map of returns from the scanner as an image. Sharf et al. [15] proposed a hole-filing algorithm inspired by example-based image completion methods. Using a volumetric descriptor, they searched an example dataset for the closest match, and then warped the resulting geometry into the hole by solving a partial differential equation. Park et al. [13] extended the idea to work with textured scans and solved the warping partial differential equation in 2D.

Xu et al. [17] used a single image to guide hole-filling in range data, by estimating normal vectors for points inside the hole based on training data from image patches. They used the estimated normal vectors to integrate over the hole, producing 3D surfaces that were more physically accurate than methods based purely on geometry, but tended to be smoother than the ground truth.

Techniques based on stereo [7, 6] and structure-from-motion [5, 1, 3] use multiple images to estimate 3D geometry. This geometry can be quite accurate, but is generally much sparser and more irregular than is typically acquired with a range scanner. Our interest in this paper is in learning 3D structure from a single image/scan pair.

Hoiem et al. [11] used a supervised learning approach to classify image superpixels into different geometric types, which were used to make realistic 3D "pop-ups" from a single photograph. Saxena et al. [14] used a multi-scale Markov Random Field over superpixels in a supervised learning approach to generate 3D data from image patches. Neither approach can generate fine 3D geometric detail, which is our interest here.

Finally, Hessner and Basri [9] used a large database of image/depth pairs to estimate convincing depth information for images of people, hands, and fish. Our approach is related, but learns 3D structure in much larger-scale real-world architectural scans.

Our overall technique is inspired by recent work on exemplar-based image inpainting [4] and image analogies [10]. Criminisi et al. [4] presented an algorithm for automatically filling in a desired target area (e.g. a region to be replaced in a digital photograph) with patches from a source region so that the resulting image seems unmanipulated. We adopt a similar idea of filling in LiDAR holes in a prioritized order depending on the amount of known geometry near a given pixel on the hole boundary. Hertzmann et al. [10] automatically created image analogy filters by transferring a learned relationship between a (original image, filtered image) pair to a new original image. Our algorithm uses a similar idea: we create a database of (image patch, 3D geometry) pairs from the non-hole regions of a scan to estimate the 3D geometry that corresponds to a new image patch.

3. Input, Output, and Preprocessing

Our algorithm begins with a LiDAR scan S (Figure 2(a)), an image of the same scene from a different perspective I (Figure 2(b)), and a camera model P that specifies the position, orientation, and internal parameters of the camera that produced I in the coordinate system of S. In all the real-world experiments reported here, we used a Leica HDS3000 range scanner to obtain the scan S, which consists of a set of 3D points acquired using a 2D angular grid of rays from the scanner's origin.

The camera model P could be found by manually selecting correspondences between S and I and applying a resectioning algorithm [8], or automatically using a direct 2D-3D registration algorithm. For this paper, we used the direct algorithm proposed by Yang et al. [18], which is based on alternating between estimating the camera model from SIFT correspondences [12] and growing the region over which the estimated model is reliable. Figure 2(c) shows the result of applying this algorithm to the scan/image pair in Figures 2(a) and 2(b), re-rendering the LiDAR scan from the estimated perspective of the camera that acquired the image.

We run two preprocessing steps on the scan S. First, we robustly estimate the normal at each scan point by fitting a local tangent plane, visiting relatively flat and noise-free regions first, since they are likely to produce stable estimates. We also create a simple quadrilateral mesh on S based on the 2D grid of ray directions cast by the scanner, and identify each edge in the grid as a surface edge or a boundary discontinuity based on the depth difference between adjacent faces.

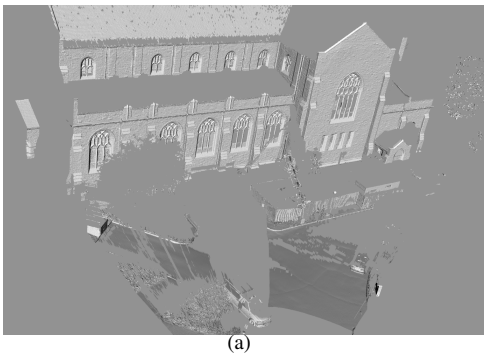
4. Types of Holes

Three main types of holes generally exist in LiDAR scans, as illustrated in Figure 3.

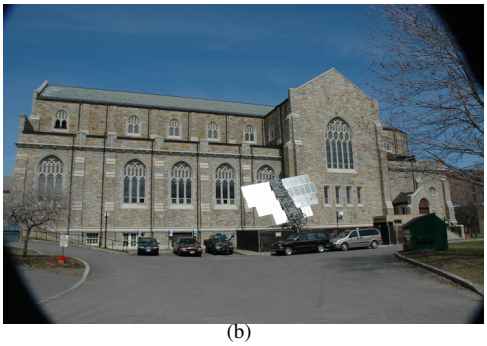
1. Occlusions (Figure 3 light gray lines). Since the closest surface to the scanner along each ray is what produces the depth measurement, further scene points along this ray on more distant surfaces appear as "shadows". This is the most common type of hole.

2. No Returns (Figure 3 thick line). The laser pulse may never return to the sensor for several reasons. For example, there may be no surface to reflect off of (i.e., sky), the closest surface may be beyond the scanner's range, or the surface material may be unconducive to scanning (e.g., windows and specular surfaces).

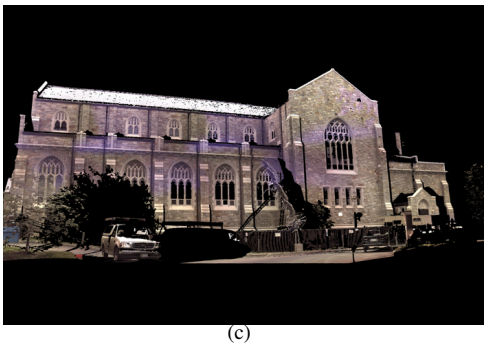
3. Out of View (Figure 3 medium gray lines). Scans with a finite angular extent have boundaries outside which no geometry is acquired.



(a)



(b)



(c)

(d)

Figure 2. (a) A LiDAR scan S of an example environment. (b) An image I of the same environment from a different perspective. (c) A re-rendering of the LiDAR scan from the estimated perspective of the camera that acquired the image in Figure 2(b). (d) The inpainting mask M. Black corresponds to the source region and white to the target region to be inpainted.

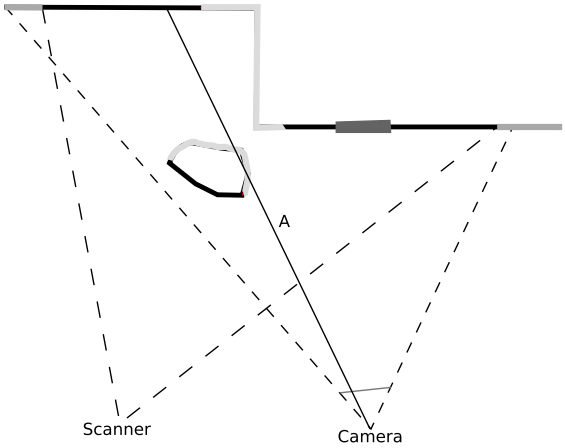
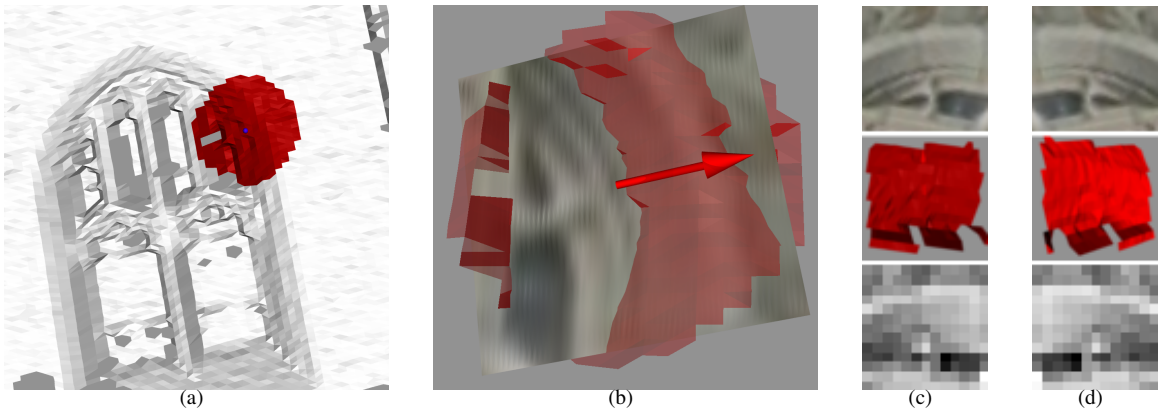


Figure 3. Types of holes: occlusions (light gray), no returns (thick), and out of view (medium gray). Ray A is an example of a false intersection. The solid black line is the scanned geometry.

For inpainting, we require a binary mask M that specifies which image pixels correspond to LiDAR holes. Generally, these can be automatically identified by analyzing the no-geometry regions in the LiDAR scan re-rendered using the estimated camera model (i.e., black regions in Figure 2(c)). Reasoning about which hole type corresponds to each pixel is straightforward based on the view frustum of the camera model and the boundary discontinuity information generated at the preprocessing step; however, this classification is not strictly necessary for the inpainting algorithm proposed in the next section.

The non-hole regions of the image will be used to form the source data for generating the example dataset of (image patch, 3D geometry) pairs. However, fully automatic generation of the source region from M is challenging due to *false intersections*. That is, an optical ray in the image may intersect a surface close to the camera that did not produce a LiDAR return, while the LiDAR scanner acquired 3D geometry from a surface further along the optical ray (e.g., the ray labelled A in Figure 2(c)). While these pixels are not LiDAR holes, they cannot be used for the inpainting source region since the image texture and 3D geometry are inconsistent. While we explored several approaches to automatically detect such regions (e.g., using image change detection algorithms), we currently defer to the user to edit the mask M to ensure that false intersections are not included in the source region. Figure 2(d) illustrates the mask M for the scan/image pair in Figures 2(a)-2(c). We denote pixels where M=0 to be the source region (i.e., trusted image/geometry locations) and pixels where M=1 to be the target region to be filled by inpainting.



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

Figure 4. (a) 3D point and surrounding area, (b) fitted planar patch with image samples and gradient, (c) head-on view of patch oriented with dominant gradient direction up (top: high-resolution image used to estimate gradient, middle: corresponding 3D geometry, bottom: 13×13 descriptor), (d) the mirrored patch that is also added to the database.

5. The Inpainting Algorithm

5.1. Generating the Example Database

Our first step is to generate an example database D of image patches and their corresponding 3D geometry. In this paper, we use the source region defined by the mask M for a given image to generate this database, although information from other scan/image pairs (or multiple such pairs) could also be used. Since a fixed-size image region does not correspond to a fixed-size region in 3D space, we generate the database using fixed-sized regions in 3D space, determined by a user-defined radius r, and estimate rotation and scale-invariant image patch descriptors corresponding to a given 3D region.

We iterate over each pixel i in I that is considered to be trustworthy geometry in M (i.e., $M(i) = 0$). From the camera model P, we know the corresponding intersection with the 3D scan S; denote this location $S(i)$ (Figure 4(a)). We next fit a 3D planar patch Π to the set of LiDAR returns within a sphere of radius r centered at $S(i)$, as illustrated in Figure 4(b). We impose a uniform grid on Π and estimate a color at each grid point by projecting the planar patch into the image using P and applying bilinear interpolation. We reject patches that contain less than 90% trustworthy geometry (i.e., more than 90% of the patch must be outside a hole region).

We now treat this colored planar patch as a small image to be added to the example database. The descriptor we use is simply a w^2-length vector of pixel intensities corresponding to a subsampled $w \times w$ grid on Π (Figure 4(c)). The subsampled grid is oriented so that the x-axis is aligned with the dominant gradient of the patch, estimated using the algorithm proposed by Lowe [12]. If multiple dominant gradients are detected (e.g., for a strong corner), we generate multiple patch orientations and corresponding descriptors. We also add a "mirror" of each patch to the database by reflecting it across its y-axis (Figure 4(d)), to make it easier to find symmetries in the matching phase (see Section 5.3). Finally, we create a query kd-tree K from the descriptors of all accepted patches, to ease the search problem in the matching phase. Throughout this paper, we used the descriptor dimension $w^2 = 169$, which we found to produce good descriptive power while keeping the kd-tree search fast.

5.2. Fill Order

The next step is to determine the fill order of pixels in the target region. For image inpainting, Criminisi et al. [4] demonstrated that the order of filling in image patches has a significant effect on the quality of the final result. Their approach defined the fill order based on a priority score that was the product of (1) a confidence term that measured the amount of known image texture surrounding a given pixel on the fill boundary and (2) a data term that was related to the dot product between the normal vector to the fill region boundary and the local image gradient. In our case, we only use a confidence term since the auxiliary image provides good visual evidence about how to fill the LiDAR hole (unlike image inpainting, where no side information about the hole interior is available).

Let b be a pixel on the boundary of the target region in M. As in Section 5.1, we fit a 3D planar patch Π_b around the 3D structure corresponding to pixel b, and backproject the values of M onto this plane to estimate the fraction of the patch that is part of the source region. If the patch is quantized with a grid of $u \times u$ bins, the confidence term is thus

$$C(b) = \frac{1}{u^2} \sum_{q \in \Pi_b} \hat{M}(q). \quad (1)$$

where $\hat{M}(q)$ is the backprojected mask value corresponding to pixel q. Figure 5 illustrates the fill order for a real example, demonstrating that target regions that protrude into source regions will be filled in first.

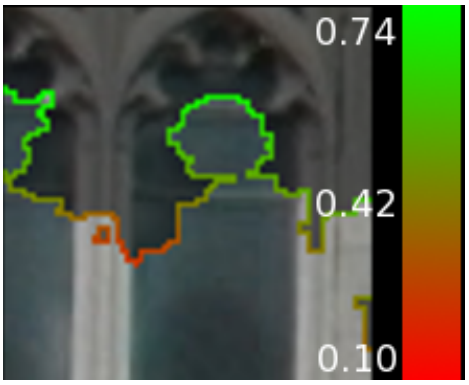


Figure 5. Fill-in order, expressed on a scale from red (low priority) to green (high priority).

5.3. The Fill-in Algorithm

Finally, we specify how 3D geometry from the source region is copied into the target region using the example database. We iteratively choose the highest-priority boundary pixel b based on (1) and perform the following steps inspired by image analogies [10]. The overall process is illustrated in Figure 6.

We compute the image patch I_b corresponding to b using the method described previously. That is, I_b is computed from a square patch Π_b in 3D, colored with pixel values from I, and oriented in the direction of the dominant gradient. In 3D, this corresponds to a set of points S_b which we determine using the estimated camera model P. We know that there are regions of the image near b that have no corresponding 3D geometry in S_b, since by definition the pixel b is adjacent to a LiDAR hole in the re-rendered image of S.

We next compare I_b to the example database of image patches. We use the optimized kd-tree K from Section 5.1 to find the top 10 matches, which is fast since the patches in the database are coarsely sampled. Then we compute the normalized cross-correlation between the image patch and the planar patch around each of the 10 candidates at a higher resolution (in this paper, we used 61×61 patches). Finally, we choose the patch with the largest correlation as the best match. One example of this process is illustrated in Figure 7.

We copy the corresponding geometry S_{d_b} into the hole region of S_b, after 3D alignment of the two regions. To align S_{d_b} to S_b, we robustly compute an initial transformation (rotation plus translation) $\hat{T}_b$ by aligning the planar patch Π_{d_b} to Π_b. We determine the final transformation T_b by running the Iterative Closest Point algorithm [2] on S_{d_b} and S_b, using $\hat{T}_b$ as the initialization. We then mark the newly filled pixels in M as no longer part of the target, and update the confidence term C for all the new pixels by setting their values to 1.

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

Figure 6. The overall fill-in algorithm. (a) A patch is selected on the current target boundary in the mask. (b) The corresponding image patch is identified. (c) The corresponding partial 3D structure is identified. (d) The best match to the image patch is found in the example database. (e) The structure from the best match is registered to the hole boundary and copied into place.

	Query	1	2	3	4	5	6	7	8	9	10
Dist		58.5	58.8	59.1	59.4	59.5	59.7	59.8	61	61.4	62.1
Corr		0.749	**0.795**	0.784	0.741	0.753	0.785	0.795	0.733	0.738	0.736

Figure 7. The query patch (I_b) and its top 10 results. Underneath each result is its kd-tree descriptor distance (Dist) and its normalized cross-correlation (Corr). The bold number indicates the selected patch (D_{d_b}). The result patches that are very similar to each other were sampled from neighboring pixels.

6. Experiments

We tested our algorithm on synthetic and real-world examples, with both artificially introduced and natural holes. All experiments were run on an 8-core 2.66GHz Xeon computer. For experiments where we have ground truth (Section 6.1 and Section 6.2), we provide two quantitative measurements of the quality of the result. First, we compute the mean Euclidean distance of from each filled in point p_f to the closest ground truth point p_g, i.e., $\|p_f - p_g\|$. The second measure is the mean of the ratio

$$\frac{\|p_f - P_c\|}{\|p_g - P_c\|} \tag{2}$$

where P_c is the location of the camera in the coordinate system of S. This measure is useful since it explicitly takes into account the distance of from the sensors to the acquired/generated points.

6.1. 3D Letters

Figure 8 illustrates an inpainting test using a synthetic model of raised letters on a plane with the phrase "TEST FOR 3DIM". We removed the word "FOR" from the model to generate a hole (Figure 8(a)). Using Figure 8(b) as the input image and Figure 8(a) as the scan, we automatically generated new geometry to fill in the artificial hole in Figure 8(c). The scan was 4.3×4.3m and the fill radius was $r = 0.20$m. The mean distance of from the camera to the missing points was 12.5m.

Our algorithm generated a convincing 3D "FOR" even though none of the letters existed in the rest of the 3D scan. The result has a median distance from the ground truth of 0.33cm and a median ratio of 1.00. The rotationally invariant patch description and the addition of mirrored patches into the database are critical to the algorithm's success, as illustrated in Figure 9. The third and fifth columns are particularly interesting, showing patches that straddle two characters and have excellent matches to mirrored parts of entirely different character pairs.

(a) The manually introduced hole (b) The image (c) Word filled

Figure 8. Letters experiment. (a) Scan of letters with middle "FOR" removed to create a hole. (b) Corresponding image. (c) Inpainted geometry.

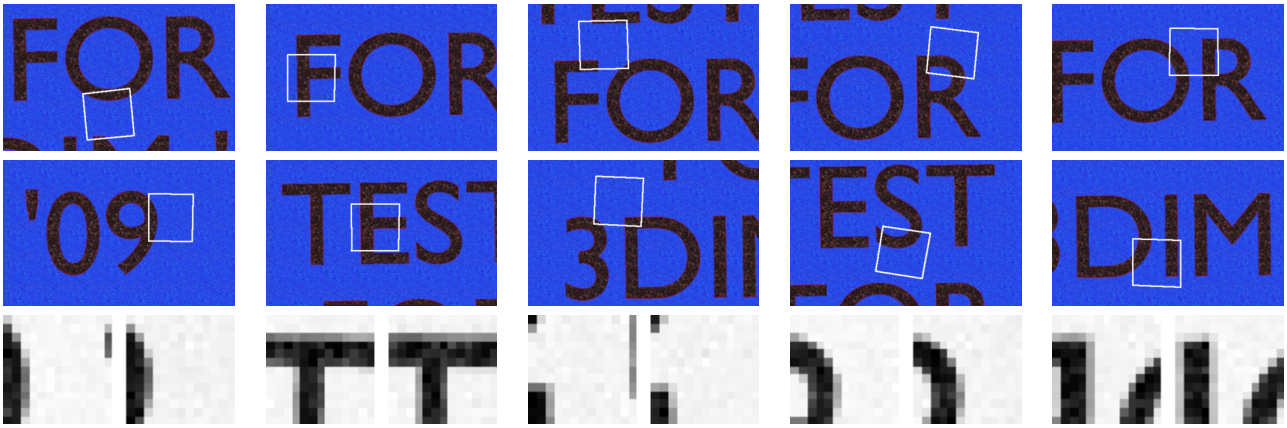
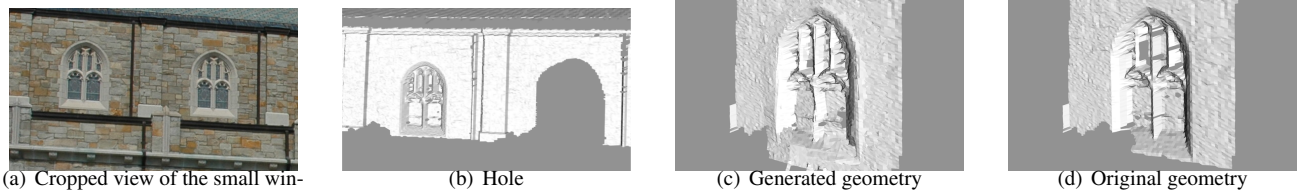
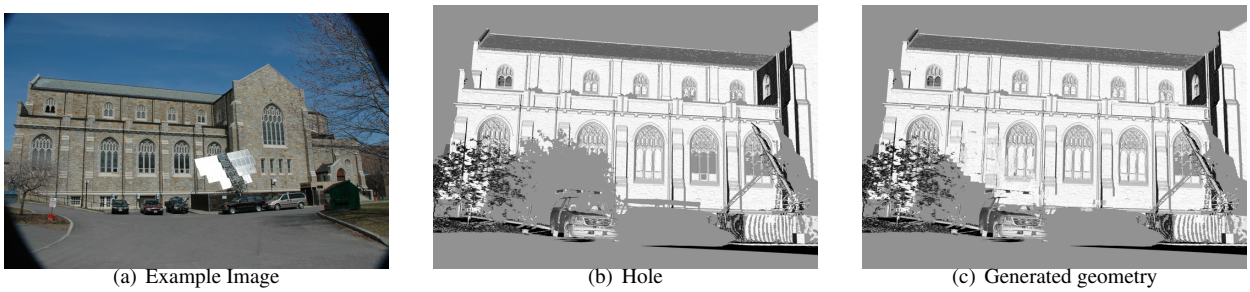


Figure 9. The first row shows several target locations around the hole boundary. The second row shows the corresponding automatically estimated image correspondences based on which the inpainted 3D geometry was generated. The third row shows the oriented descriptor for each query (left) and best matching patch (right). We can see the effect of rotationally invariant patch descriptors (columns 1 and 4) and mirroring (columns 3 and 5).



(a) Cropped view of the small window (b) Hole (c) Generated geometry (d) Original geometry

Figure 10. (a) Cropped view of the repetitive small windows. (b) One of the five windows is manually removed from the LiDAR scan. (c) Using the visual evidence from the image, new 3D structure is generated. (d) The original 3D structure for comparison.



(a) Example Image (b) Hole (c) Generated geometry

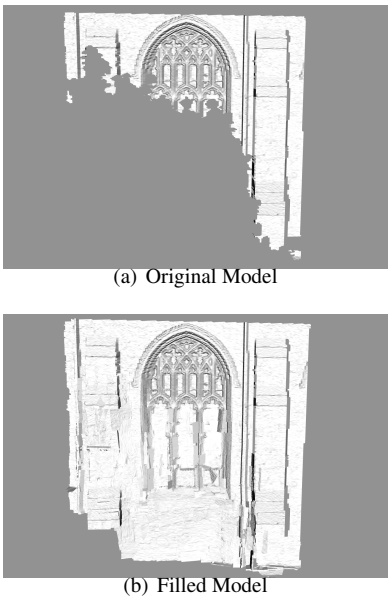
Figure 11. (a) Source image. (b) LiDAR scan. (c) Using the visual evidence from the image, new 3D structure is generated.

6.2. Small Window

We acquired a real scan of a large building with five similar windows near its roof (Figure 10(a)). We manually removed one of these small windows, so that no geometric evidence of the window remained (Figure 10(b)), and created the example database with with samples from the other small windows. For this experiment we set $r = 0.35m$. The mean distance from the camera to the scene points was 72.4m. The filling algorithm took 52 iterations to complete and about 15min wall clock time; several snapshops of the iterations were shown in Figure 1. The final result is rendered in Figure 10(c), which compares favorably with the original 3D geometry (Figure 10(d)) for the window. The median distance from the ground truth is 7.1cm, and the median ratio is 1.00.

6.3. Large-scale LiDAR shadows

Finally, we ran our algorithm on several natural holes formed from occlusion in a real-world LiDAR scan, using the large church-like building in Figure 2. We note that not all of the holes have enough supporting evidence in the patch/geometry database to be filled. Figure 11 illustrates a result showing large swaths of filled-in texture. A closer view of one window is illustrated in Figure 12, showing convincing high-detail 3D geometry.



(a) Original Model

(b) Filled Model

Figure 12. Detailed view of Figure 11 around the large window.

7. Conclusions and Future Work

We presented an algorithm for automatically filling in convincing high-detail 3D structure in missing-data regions of a LiDAR scan using a single additional image of the scene from a different perspective. The approach can mitigate the "shadows" characteristic of range scanning, improving the utility of LiDAR scans for planning and situational awareness. Another application is the construction of realistic architectural models; our algorithm works best when repetitive structure is present, and this is quite often the case with architecture. We note that while the generated

geometry is quite plausible, it should not be treated with the same level of confidence as the real geometry.

The algorithm proposed here is only a first step to completely solving the problem. We believe that a "data term" that uses partial 3D geometry to influence the fill order and database matching is critical to improve the perceptual and actual quality of the results. We also plan to design a method to determine when the inpainting is no longer reliable or accurate, to prevent aggregating errors (e.g., when extrapolating beyond the scan boundary). In this case, the hole type (occlusion, no return, out of view) should also play a role in the fill priority and fill-in algorithms. We are also investigating how to adaptively change the 3D inpainting radius based on estimated 3D structure (e.g., large, flat regions could be filled in more aggressively than small, complex regions). We also should consider how the descriptor dimension should vary with different scenes.

Finally, while the focus of this paper was on inpainting for a given target region, we also plan to automatically generate the entire source and target regions, correctly detecting and discarding false intersections and reducing the amount of user input required to define these regions.

References

[1] A. Abdelhafiz, B. Riedel, and W. Niemeier. Towards a 3d true colored space by the fusion of laser scanner point cloud and digital photos. In *Proc. of the ISPRS Working Group V/4 Workshop (3D-ARCH)*, 2005.

[2] P. Besl and H. McKay. A method for registration of 3-d shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb. 1992.

[3] A. Brunton, S. Wuhrer, and C. Shu. Image-based model completion. In *Proc. of the 6th Int. Conf. on 3DIM*, pages 305–311, 2007.

[4] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. Image Processing*, 13:1200–1212, 2004.

[5] P. Dias, V. Sequeira, F. Vaz, and J. Goncalves. Registration and fusion of intensity and range data for 3d modelling of real world scenes. In *Proc. 4th Int. Conf. on 3DIM*, pages 418–425, Oct. 2003.

[6] C. Frueh, S. Jain, and A. Zakhor. Data processing algorithms for generating textured 3d building facade meshes from laser scans and camera images. *Int. J. Comp. Vis.*, 61(2):159–184, 2005.

[7] C. Frueh, R. Sammon, and A. Zakhor. Automated texture mapping of 3d city models with oblique aerial imagery. In *Proc. 2nd Int. Symp. on 3DPVT*, pages 396–403, Sept. 2004.

[8] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[9] T. Hassner and R. Basri. Example based 3d reconstruction from single 2d images. In *Proc. Beyond Patches Workshop, CVPR*, pages 15–23, June 2006.

[10] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proc. SIGGRAPH*, pages 327–340, 2001.

[11] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. In *ACM Trans. Graphics (SIGGRAPH '05)*, August 2005.

[12] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comp. Vis.*, 60(2):91–110, Nov. 2004.

[13] S. Park, X. Guo, H. Shin, and H. Qin. Shape and appearance repair for incomplete point surfaces. In *Proc. 10th ICCV*, volume 2, pages 1260–1267, Oct. 2005.

[14] A. Saxena, S. H. Chung, and A. Y. Ng. 3-d depth reconstruction from a single still image. *Int. J. Comp. Vis.*, 76(1):53–69, Jan. 2008.

[15] A. Sharf, M. Alexa, and D. Cohen-Or. Context-based surface completion. *Proc. SIGGRAPH*, 23(3):878–887, 2004.

[16] P. Stavrou, P. Mavridis, G. Papaioannou, G. Passalis, and T. Theoharis. 3d object repair using 2d algorithms. In *Proc. International Conference on Computational Science (2)*, pages 271–278, 2006.

[17] S. Xu, A. Georghiades, H. Rushmeier, J. Dorsey, and L. McMillan. Image guided geometry inference. In *Proc. 3rd Int. Symp. on 3DPVT*, pages 310–317, 2006.

[18] G. Yang, J. Becker, and C. V. Stewart. Estimating the location of a camera with respect to a 3d model. In *Proc. 6th Int. Conf. on 3DIM*, pages 159–166, 2007.