Optical Engineering

OpticalEngineering.SPIEDigitalLibrary.org

Dynamic visualization of three-dimensional images from multiple texel images created from fused ladar/digital imagery

Cody C. Killpack
Scott E. Budge

SPIE.

Cody C. Killpack, Scott E. Budge, "Dynamic visualization of three-dimensional images from multiple texel images created from fused ladar/digital imagery," *Opt. Eng.* **56**(3), 031209 (2016), doi: 10.1117/1.OE.56.3.031209.

Dynamic visualization of three-dimensional images from multiple texel images created from fused ladar/digital imagery

Cody C. Killpack* **and Scott E. Budge**
Utah State University, Center for Advanced Imaging Ladar, Logan, Utah 84322-4120, United States

Abstract. The ability to create three-dimensional (3-D) image models, using registered texel images (fused ladar and digital imagery), is an important topic in remote sensing. These models are automatically generated by matching multiple texel images into a single common reference frame. However, rendering a sequence of independently registered texel images often provides challenges. Although accurately registered, the model textures are often incorrectly overlapped and interwoven when using standard rendering techniques. Consequently, corrections must be done after all the primitives have been rendered by determining the best texture for any viewable fragment in the model. This paper describes a technique to visualize a 3-D model image created from a set of registered texel images. The visualization is determined for each viewpoint. It is, therefore, necessary to determine which textures are overlapping and how to best combine them dynamically during the rendering process. The best texture for a particular pixel can be defined using 3-D geometric criteria, in conjunction with a real-time, view-dependent ranking algorithm. As a result, overlapping texture fragments can now be hidden, exposed, or blended according to their computed measure of reliability. The advantages of this technique are illustrated using artificial and real data examples.
[DOI: 10.1117/1.OE.56.3.031209]

Keywords: lidar; ladar; texel image; three-dimensional texel model; dynamic texture; view-dependent rendering; render to texture.

Paper 161021SSP received Jun. 28, 2016; accepted for publication Oct. 14, 2016; published online Oct. 26, 2016.

1 Introduction

The need for automatically generating three-dimensional (3-D) image models exists in many technical fields of study. Creating a 3-D model with visual and dimensional accuracies is an invaluable tool for disaster management, situational awareness, and even automatic target recognition. Building these models involves combining 3-D data and image textures captured from various viewpoints around an object of interest. Once the images are matched with a registration algorithm, they can be viewed and explored in a vector graphics tool. The ability to apply and overlay accurate textures within this viewing program is an important step toward achieving a usable model.

Preliminary work on the algorithms reported in this paper has been previously reported.[1]

The Center for Advanced Imaging Ladar (CAIL) at Utah State University uses texel images (fused ladar and digital imagery) to create 3-D models. Texel images are generated with a texel camera, which houses a coaxial ladar array and electro-optical (EO) camera. The data produced by a texel camera are fused at the pixel level.[2,3] Researchers at CAIL have exploited this data fusion to register multiple texel images into a single 3-D model.[4,5] The transformation parameters computed during registration are used to match individual texel images, bringing them into a single reference frame (or common world coordinate system). The result is a 3-D model composed of independent 2.5-D texel images.

Texel images are unique in that each texel image contains a lidar point cloud and higher-resolution digital imagery fused at the output of the sensor. The point cloud data in the dataset can be used to generate a 3-D triangulated interconnected network overlaid with the image data. The goal of this research was to render multiples of these textured surfaces acquired from different viewpoints jointly in a common 3-D space. The methods described here to render these images in a real-time vector graphics tool according to the current viewpoint [transforming a 3-D sorting problem into a two-dimensional (2-D) ranking solution] are unique.

Rendering these 3-D texel models continues to be an important research focus. Rendering involves loading and viewing a 3-D model. Numerous vector graphics tools provide the ability to magnify, transform, and explore a texel model in 3-D space. However, viewing a sequence of independently registered texel images often provides challenges, and the problems that arise when using standard rendering techniques need to be corrected. Even when a 3-D model is accurately registered, the individual textures assigned to it are often incorrectly overlapped and interwoven. This is caused by stretched mesh triangles, noise, and the default rendering methods used by typical vector graphics tools, as shown in Fig. 1. In Fig. 1(a), the problems with stretched triangles caused from rendering images taken from two different viewpoints are clearly shown. Figure 1(b) shows the same registered scene with the textures replaced by a color assigned to each image, indicating where the rendered textures originate.

*Address all correspondence to: Cody C. Killpack, E-mail: cody.killpack@aggiemail.usu.edu

0091-3286/2016/$25.00 © 2016 SPIE

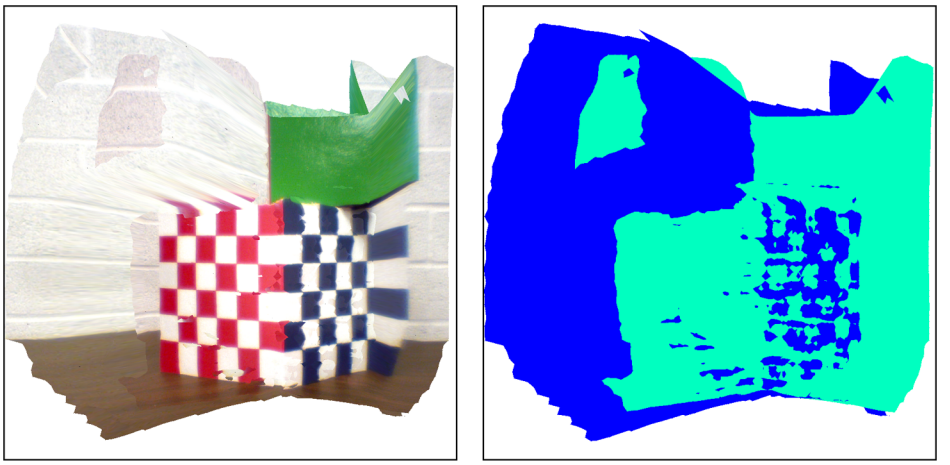

Fig. 1 A registered 3-D texel model with noise and distortion present. The individual texel images, or scans, were taken from two slightly different perspectives. (a) The combination of these two texel images in an independently registered and textured sequence. (b) The same scene found in (a) but with a unique color assigned to each image, rather than a texture.

Forming a texture atlas (stitching textures together) is impractical, due to the independent nature of the underlying datasets. Consequently, corrections must be done after all the primitives (triangles) have been rendered by determining the best texture for any given area of the model. Triangle-based multitexturing techniques have been proposed and define the best texture for a given area as a triangle labeling problem.[6]

The "accuracy" of a texture for a particular area of the model can be defined using a 3-D geometric criterion. Additionally, this criterion must be used in conjunction with a real-time, view-dependent ranking algorithm, as the virtual camera perspective also influences the suitability of a texture. The concept of view-dependent texture mapping has been proposed, along with examples of real-time applications.[7,8] Well-known 3-D geometric criteria have also been demonstrated such as the angle between a triangle normal and the viewing frustum, distance to the camera, surface area of a triangle face, and even photoconsistency metrics.[6,9–11] These methods have all been explored and ranked.[12] A combination of these metrics, referred to as the "reliability" of the texture, would allow overlapping texture fragments (pixels) to be hidden, exposed, or blended according to their computed measure of reliability. Methods for blending these textures have been proposed.[13,14] Current techniques determining the visibility and rank of a texture triangle can also be found.[14,15]

The problem addressed in this paper is unique because each texel image remains independent after registration. The depth data have not been merged, thus eliminating the need for a fused texture atlas. Therefore, the solution must determine which textures are overlapping and how to best combine them dynamically while rendering. The OpenGL Shading Language (GLSL) enables the textures to be combined on a fragment level (per-pixel) during the rendering process. Furthermore, the accuracy of a texture triangle can be interpolated, which results in greater uniform interaction among overlapping textures. As a result, the best texture assignment for a given area of the 3-D texel model can be done on a per-fragment basis. This is achievable in real-time for applications that use texel images obtained from framing ladar sensors.

The remainder of this paper is presented as follows. Details concerning triangle ranking and 3-D geometric criteria are given in Sec. 3. Examples using synthetic texel images, used to illustrate the feasibility of the rendering algorithm presented in this paper, are also given. Section 4 discusses the rendering algorithm in its entirety. Current implementation and programming techniques are covered in detail. The results can be found in Sec. 5. Section 6 concludes this paper, giving insight into the advantages and limitations of the rendering algorithm presented.

2 Transparency Sorting Limitations

When typical vector graphic blending is enabled, the renderer will combine (mix) new colors being drawn to the screen with any pixels already present in the framebuffer. As a result, all opaque objects must be drawn before attempting to render translucent ones. Successfully implementing transparency sorting involves rendering object layers in the correct order. The rendering order depends on both the transparency and the depth of the object layer, as shown in Fig. 2. In this example, Fig. 2(a) contains three colored blocks which have been rendered at different depths. Since the layers are all opaque, the rendering order is irrelevant. However, once the alpha values for these layers begin to change, the need for ordered rendering becomes apparent, as shown comparing Figs. 2(b) and 2(c). The difference between these two scenes is a direct result of the default context implementation (painter's algorithm) used by a typical vector graphics library. In this case, the renderer has painted distant objects first and closer objects second. Consequently, a higher level of algorithm complexity is needed to correctly render transparency sorted textures.

In addition to ordered rendering problems, transparency sorting techniques begin to break down with complex object interaction. At the root of these limitations lies the "cyclic" and "intersecting object" problem cases, as shown in Fig. 3. In Fig. 3(a), the unique overlapping nature of these three objects prevents transparency sorting by hindering proper

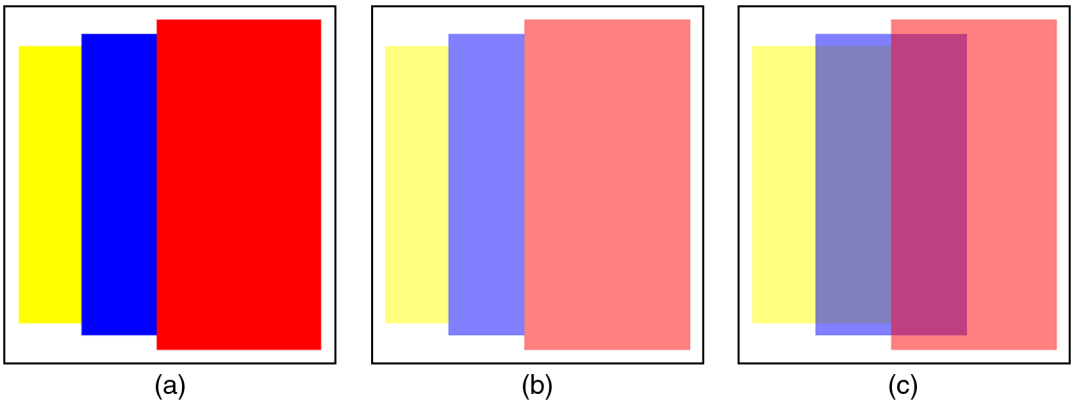

Fig. 2 Ordering principle of transparency sorting. (a) Opaque block objects rendered at different depths. (b) Transparent version of the blocks in (a) rendered "front to back." (c) Identical scene in (b) rendered "back to front."

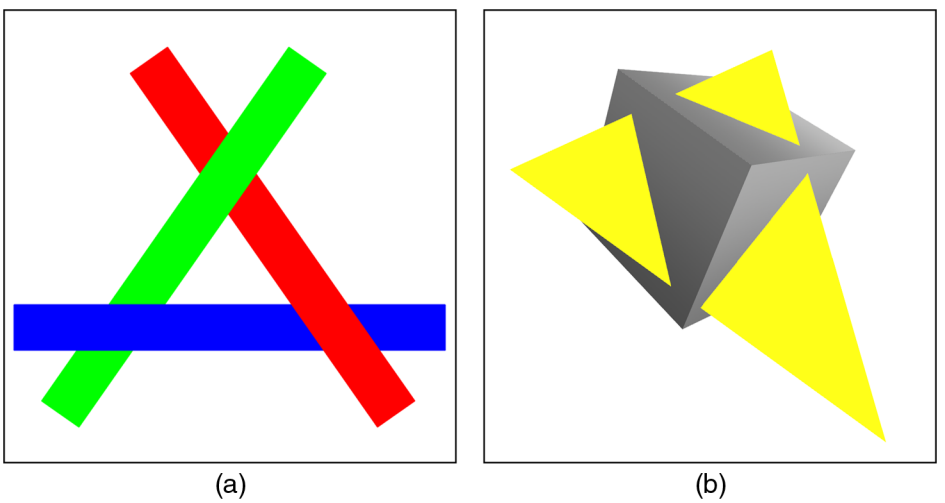

Fig. 3 Classical transparency sorting problem cases. (a) Cyclic case. (b) Intersecting case.

triangle ordering. All possible rendering orders produce conflicting results. The problem is, therefore, cyclic and can only be corrected by "splitting" intersecting areas into additional objects. A similar problem is shown in Fig. 3(b), where object intersection also prevents reliable transparency sorting and binary space partitioning[16] must be used to sort and split static transparent polygons. Unfortunately, implementing a real-time sorting and splitting algorithm for large texel images would be difficult, and the use of these traditional techniques for texel model rendering is discouraged.

Shading languages provide powerful alternatives to that of traditional transparency sorting techniques. Sorting and blending can be done per-fragment (per-pixel) using fragment shaders. Implementing multishader pipelines enables real-time processing techniques to be performed on individual texel images and allows for additional model processing on separate rendering passes. Sorting and blending fragments are faster and more efficient than attempting traditional transparency sorting methods on triangle meshes. Utilizing the graphics processing unit to perform fragment level sorting and blending is discussed in detail in Sec. 4.

3 Accumulated Texture Reliability

An estimation of accumulated texture reliability for a single texel image, or scan, is done dynamically via 3-D geometric criteria. The reliability of a texture represents an accumulated rankable value, which is computed using three specific criteria discussed in this section. The ability to determine rank among overlapping textures, based on their estimated reliability values, is a critical step for proper texel model rendering. Not only must these reliability ratings be accurate, but they must also adapt dynamically to satisfy changes in virtual camera perspective. For convenience, the values assigned by these criteria are clamped, falling between the range [0.0, 1.0]. A reliability rating of zero represents a discardable fragment (essentially unusable for rendering), while a reliability rating of one represents an ideal fragment. In general, a wide range of reliabilities will be found across the surface of a texel image mesh; however, only a limited number of these will ever be considered perfectly reliable.

The steps for computing the accumulated texture reliability of a single texel image, viewed from the current virtual camera position, are summarized in Algorithm 1. This process is repeated until all visible scans in a texel model have been rendered. Any scan located on the backside of a 3-D

Algorithm 1 Computing texture reliability for a single texel image.

1. Compute the "normal vector accuracy" for each triangle in the mesh. This is done "once" for each texel image in the scan set.
2. Render a single texel image. For each primitive (pixel) in the image:
   a. Compute the point of view (POV) confidence.
   b. Interpolate "range confidence."
   c. Compute accumulated texture reliability.
   d. Interpolate the pixel color.
   e. Store corresponding pixel color and reliability data for future pipeline processing.

texel model should not be processed. Once completed, 2-D textures (containing color and reliability information) for every scan in the model will be available for use in the second stage of the rendering pipeline. This process is known as "rendering to texture." As a result, a complicated 3-D sorting problem has been resolved with a simple 2-D texture solution. Additionally, transparency sorting and object intersection algorithms will no longer be needed for proper model rendering.

Computing accumulated texture reliability begins with "normal vector accuracy." Using the normal vector orientation of a triangle face within a texel image mesh is a suitable method for determining accuracy. Triangle accuracies are constant values and are computed using the original acquisition perspective of a texel image. Normal vectors oriented toward the acquisition center of projection (COP) usually correspond to triangles with smaller surface areas, which result in less texture interpolation (pixelation). Triangles whose normal vectors are rotated away from the frustum viewpoint are usually less accurate and often correspond to stretched and noisy areas of the mesh. Using the cosine similarity, given by

$$\cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{N}}{\|\mathbf{a}\|\|\mathbf{N}\|},$$

between a triangle normal vector $\mathbf{N}$ and the acquisition COP vector $\mathbf{a}$ provides a suitable measure of accuracy. Figure 4 helps show this principle. As the angular difference between these two vectors approaches 0 deg, the computed accuracy

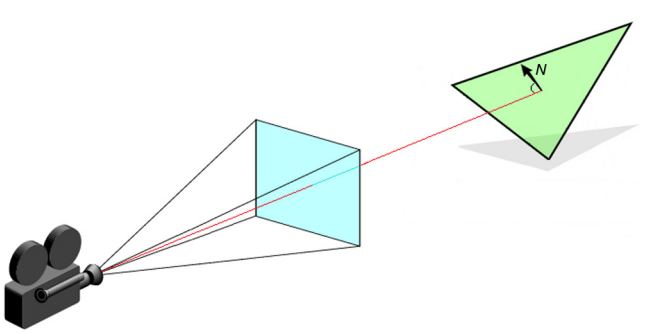

Fig. 4 Triangle normal vector accuracy.

draws closer to 1. As the angular difference increases, the accuracy level decreases, and eventually reaches 0 (90-deg difference). The acquisition COP vector is not constant, which often results in accuracy degradation for triangles located further away from the center of the texel image. Even a perfectly flat surface can only have ideal accuracies when the triangle normal vectors and the acquisition COP vector are identically oriented (the image center).

Although normal vector orientation assigns an accuracy value for every triangle in a texel image mesh, this value is constant and only represents the maximum possible accuracy for any given triangle. To account for deviations in virtual perspective, numerical values known as "confidences" must also be assigned to every texel image triangle. When used together, confidence and accuracy values define triangle reliability when the virtual camera deviates from the acquisition perspective of a texel image. Not surprisingly, as the perspective view of a triangle changes, the confidence of its texture will begin to degrade. Consequently, the most reliable view of any triangle is the view from which it was originally acquired. Any deviation from this acquisition perspective begins to decrease a triangle's confidence, thus changing its overall reliability.

"POV confidence" is the correlation between the current view (virtual camera) and the acquisition view (real texel camera). When these two perspectives align, the POV confidence of any triangle in the mesh is said to be "ideal." Not surprisingly, cosine similarity provides a reliable and inexpensive way to calculate these confidence values in real-time. A comparison is done between the acquisition COP vector and the orientation of the triangle mesh from the current POV. The measure of confidence is computed using the normalized dot product between these two vectors. Angle differences greater than 90 deg are ignored and will not be rendered.

"Range confidence" describes the correlation between the virtual and acquisition distances of a given triangle. The original acquisition distance (meters) is compared with the range, or distance, between rendered mesh triangles and the virtual camera COP. Vertex coordinates are commonly captured in metric units, therefore, the rendered scene is required to use the same coordinate system for clipping plane placement and camera movement. These confidence values change as the virtual camera zooms in and out of a rendered texel image. As the virtual camera moves closer to the 3-D location of a triangle than the range from which it was originally acquired, pixel interpolation (zooming) causes the texture to be more distorted and hence less confident.

The accumulation of triangle normal vector accuracy, POV confidence, and range confidence constitutes the "accumulated texture reliability" measure of a mesh. The three measures are combined by taking the product of the measures. Accumulated reliability values represent the overall accuracy of any triangle for a given distance and orientation in virtual 3-D space. These values are then interpolated on a per-fragment (pixel) level at render time using GLSL shaders. The accumulated reliability for a fragment is highest when its virtual and real-world acquisition frustums perfectly coincide. Overlapping fragment areas of a texel model can now be accurately ranked and weighed using their individual reliability ratings. To ease alpha blending and computation,

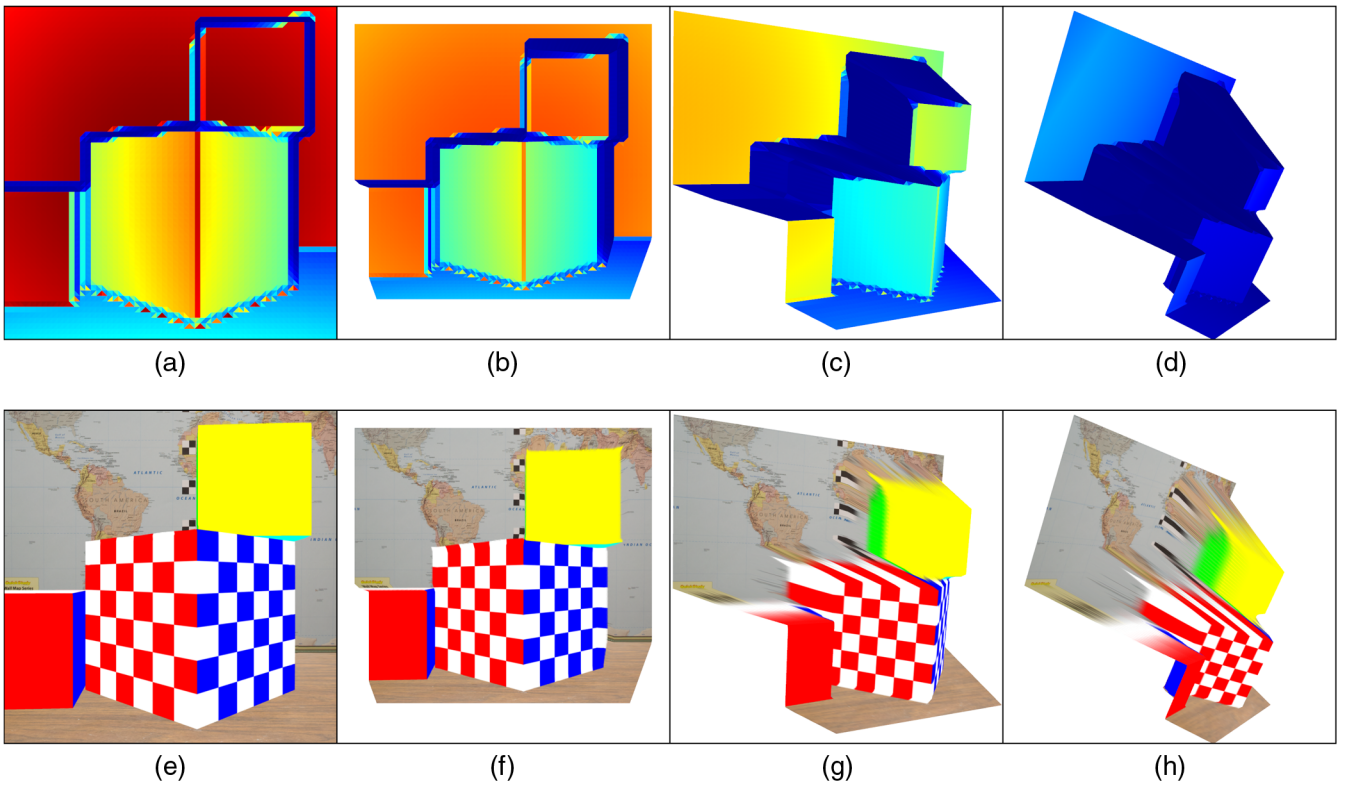

Fig. 5 An example of accumulated texture reliability is given against a single synthetic texel image, shown from a variety of virtual perspectives. (a) Acquisition alignment, which equals the maximum possible reliability for a triangle mesh. (b) Degraded accumulated reliability levels as the virtual perspective is translated back and to the left. (c) The effect on reliability due to rotation. (d) Severely degraded perspective. The perspective is so degraded that the triangles with reliability ratings of zero will be ignored by the rendering algorithm. For reference, the companion texture for (a)–(d) can be seen below each in (e)–(h), respectively.

accumulated reliability values are computed as percentages, clamped between [0.0, 1.0].

Figure 5 shows the accumulated accuracy by showing a synthetically generated texel image from various viewpoints in virtual space. Using synthetic data facilitates visualizing and evaluating imaged-based algorithms. The synthetic datasets shown in this paper mimic the real-world scenes and texel camera characteristics found in the CAIL laboratory. To aid in visualizing the reliability of the texel image mesh, a heat map has been applied. Cooler colors represent poorly rated triangles and warmer colors indicate triangles with higher reliability.

4 Texel Model Rendering

Once the individual 2.5-D scans of a texel model have been rendered and evaluated for reliability, the resulting 2-D texture outputs can be sorted and blended (multitextured) together on a per-pixel basis. Blending, or multitexturing, involves combining two overlapping textures based on an alpha value. Currently, this mask is computed using a weighted sum of reliability ratings. For a texel model composed of n scans, there may be up to n overlapping fragments for any given area. An overview of the model rendering pipeline can be seen in Fig. 6. Fragments which match the rendering background color, or those which have a reliability of zero, will be discarded. Once a list of valid fragments is obtained, they must be sorted according to their accumulated texture reliability ratings. The top ranked fragments are then extracted for processing, as shown in Algorithm 2. These fragments can be blended if they fall within a certain threshold, or simply the best fragment can be rendered. Displaying the best fragment works well with synthetic data; however, real-world data benefit from moderate blending, as a result of higher noise levels and misregistration errors.

A visual illustration of real-time, dynamic texture assignment is shown using a synthetically generated dataset in Fig. 7. In the figure, the model shown is composed of three individual 2.5-D texel images, taken from different perspectives, which have been separately rendered into 2-D textures. As a result, the final texture assignment is simplified using these 2-D images stacked on top of each other. The pixels within every column of the texture stack are sorted according to reliability. This enables the best fragments to be brought forward or possibly blended. The blue areas will be rendered from pixels in image 1, the red areas from image 2, and the yellow areas from image 3. The effect of blending is shown in Fig. 7(b), which shows how the sharp boundaries in reliability between images 1, 2, and 3 can be blended to reduce artifacts at the boundary. Figure 7(c) is the final rendered scene.

Although using a rendered stack of 2-D images simplifies model texturing, doing so eliminates any inherited visibility properties. Hidden surface removal is used to determine which surfaces should not be visible from a particular viewpoint. This process is sometimes called hiding and is often a necessary step to render images correctly. Texel model

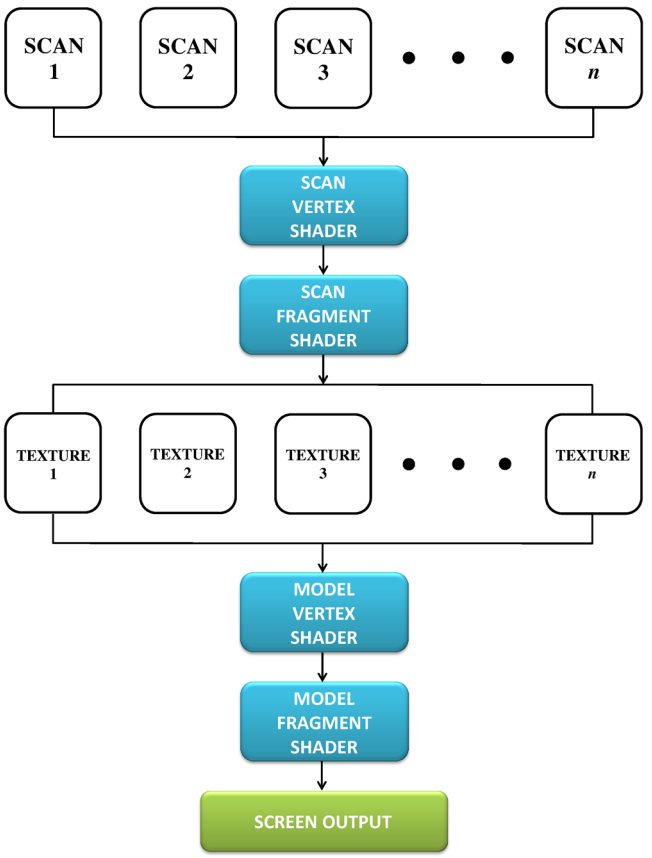

Fig. 6 Texel model rendering pipeline.

Algorithm 2 Processing overlapped textures for a 3-D texel model.

1. Extract valid pixels within a vertical column.
2. Eliminate nonrange compliant pixels.
3. Sort remaining pixels according to their accumulated reliability ratings.
4. Process the final pixel group: show best pixel or perform blending.
5. Repeat steps 1–4 for all pixels across the 2-D canvas.

rendering is no exception, as is evident in Fig. 8. In this example, portions of a texel image with higher reliability can be erroneously seen through portions of a texture closer to the virtual camera. Specifically, a small section of the world map in the background can be seen through the checkered cube. To prevent these windowing artifacts, range compliance must be used when comparing overlapping textures.

Fragments should only be processed if they are within a reasonable distance from each other (i.e., they are range compliant). Assessing range compliance is done by comparing the virtual distances between all overlapping fragments and the virtual camera COP. The distance threshold is taken from acquisition camera noise characteristics: any fragments located beyond two times the standard noise deviation should be ignored. Checking for range compliance ensures that only valid overlapping textures can be ranked, sorted, blended, and displayed.

Stretched triangles continue to present numerous challenges when rendering 3-D texel models. These areas are only valid when the virtual and acquisition camera perspectives perfectly conform, but rapidly degrade with any minor viewpoint deviation. As such, these textures have been deemed essentially "unusable" with the currently implemented rendering algorithm and are removed before overlapped texture comparisons are performed. Stretched triangles are problematic because they often erroneously pass "range compliance" checks and are, therefore, falsely considered for ranking and blending. Although removing these areas improves overall algorithm performance, it occasionally introduces unexpected discontinuities within the model. This is especially noticeable around sharp and abrupt edges, as shown in Fig. 9. These discontinuities are a direct result of inadequate sampling around abrupt edges. Techniques beyond triangles, such as "splats," may hold the key to breaking up these stretched triangles and allow certain parts of them to be hidden and others used.

Unfortunately, misregistration errors can lead to poor rendering performance, as demonstrated in Fig. 10. Although blending techniques help mitigate these problems, blurred and clouded texture areas can appear as a result. In addition, stretched and noisy triangles continue to present numerous challenges when rendering texel models. This is especially obvious when working with sharp and abrupt edges within

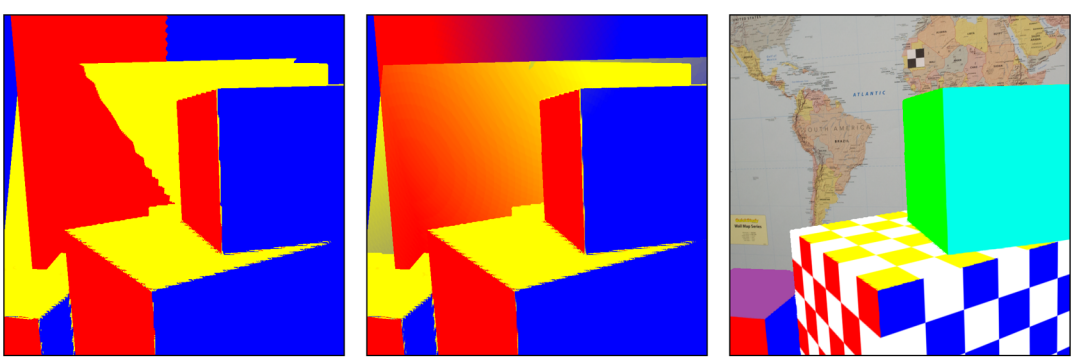

Fig. 7 A demonstration of dynamic texture assignment, computed using a real-time vector graphics renderer, using a synthetically generated 3-D model created from three texel images. (a) Sections of the image rendered from the best reliability measures in images 1 (red), 2 (blue), and 3 (yellow). (b) Blended (multitextured) reliability. (c) The final rendered image using colors from pixels selected according to the reliability map given in (a).

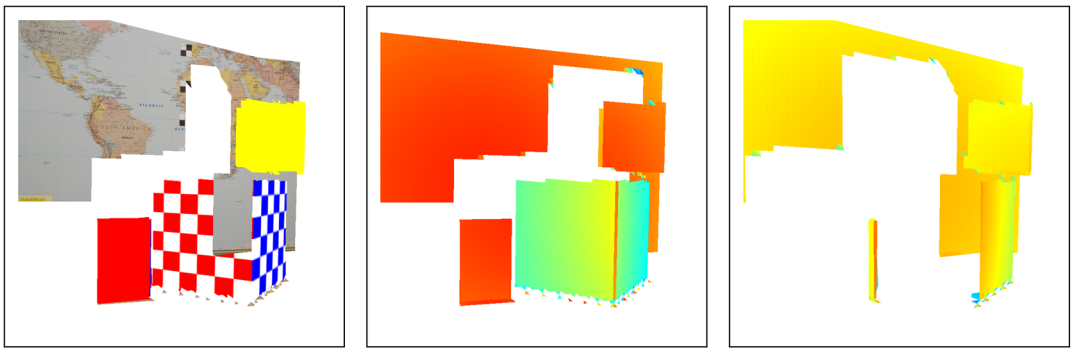

Fig. 8 The necessity of range compliance is demonstrated. Under certain circumstances, portions of imagery with higher reliability ratings can be erroneously seen through textures that are closer to the virtual camera. (b) and (c) Texture reliability heat maps for the two images in (a). Notice that image (c) has a slightly better reliability rating behind the checkered cube and begins to incorrectly overtake portions of (b).

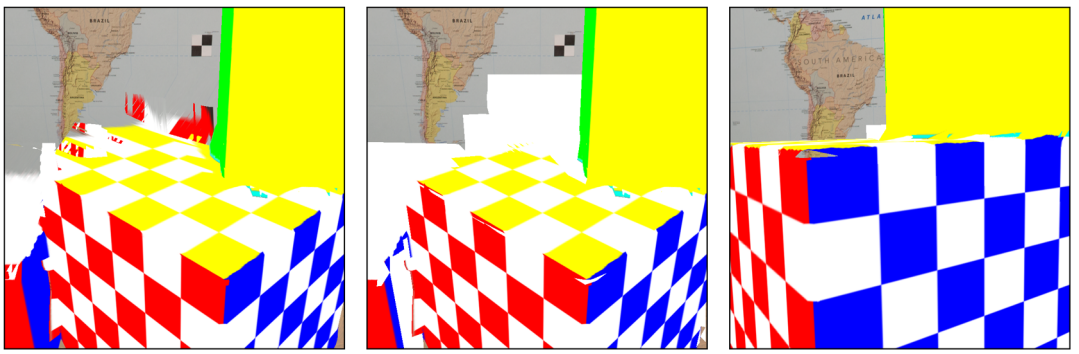

Fig. 9 Stretched triangle discontinuities, where the impact is especially noticeable around sharp and abrupt edges of a model. (a) Rendered model where the stretched triangles have not been removed. (b) Rendered model in (a), where the stretched triangles have been removed. While the overall image appearance has drastically improved, small holes have appeared along the edges of the checkered cube. (c) Alternate view of (b) with discontinuities becoming easily noticed as small portions of the background begin to show through the corner of the cube.

Fig. 10 Misregistered 3-D texel model rendering. (a) and (b) Unique texel images independently captured in the CAIL laboratory. (c) Rendered model of (a) and (b), where registration limitations become quickly apparent.

a model. Future work may investigate solutions beyond triangle-based rendering. Additionally, better blending and alternative image-based processing techniques warrant further investigation.

5 Results

Final algorithm performance is demonstrated using a synthetic dataset, shown in Fig. 11. Real-time, dynamic texturing can be seen as the model is explored from a variety of

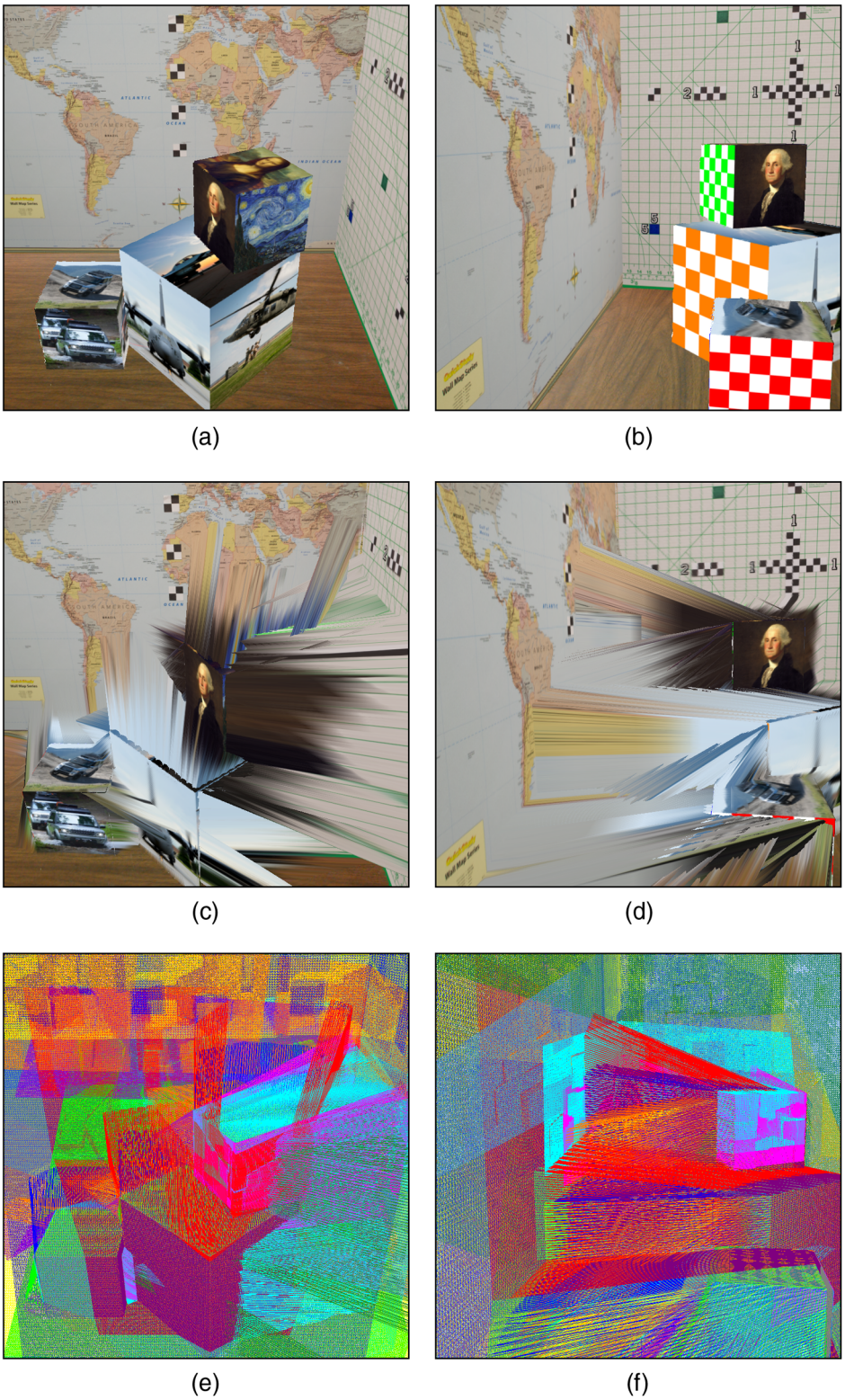

Fig. 11 Dynamic texture assignment is shown using a synthetically generated 3-D texel model from 10 texel images. (a) and (b) The same scene viewed from two different perspectives. For comparison, the original default rendering (painter's algorithm) of these two model perspectives is given in (c) and (d), respectively. The underlying triangle meshes for these two model perspectives are shown in (e) and (f).

perspectives. The model shown in Fig. 11 is composed of 10 different 2.5-D texel images which have been captured from various viewpoints around the scene using a synthetic texel image generator. The advantages of using real-time, dynamic texture assignment can be seen in Figs. 11(a) and 11(b). Although the perspectives of these two images are dramatically different, the model textures have adapted to accommodate virtual camera orientation and weighted texture

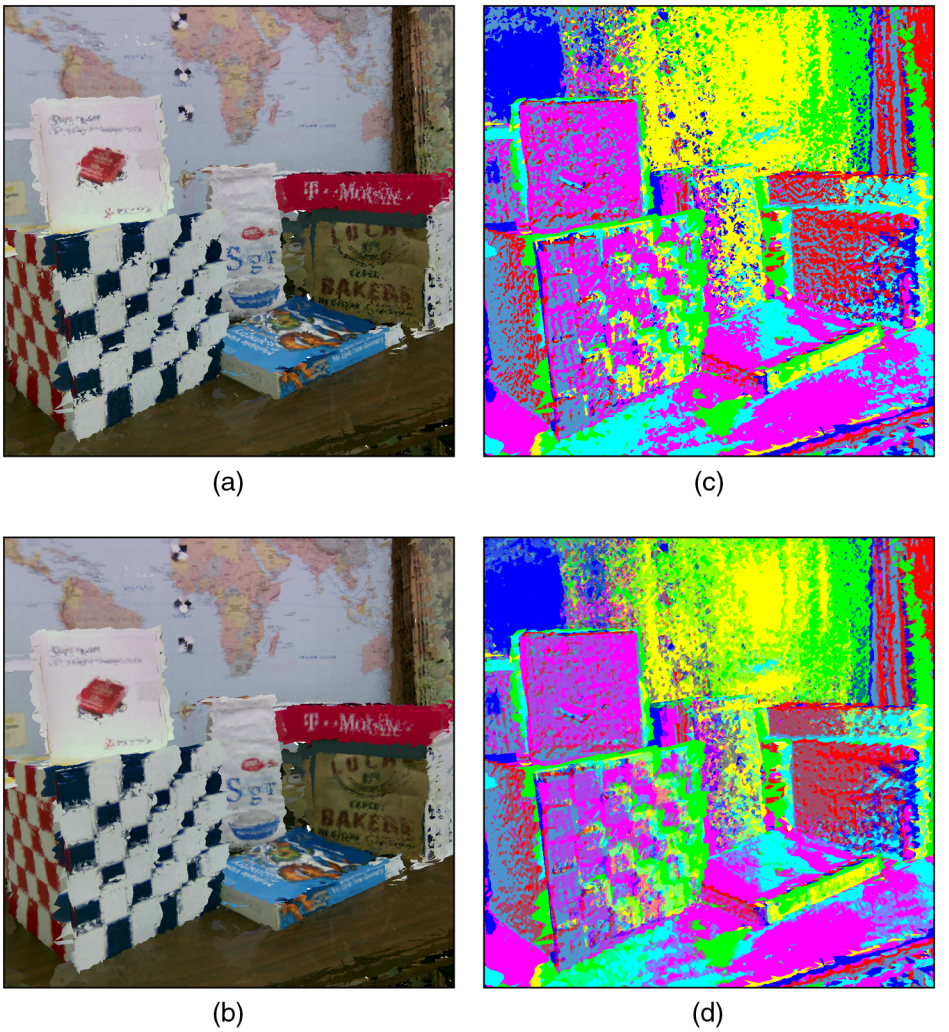

Fig. 12 Weighted texture assignment is shown using a real-world 3-D texel model from seven texel images. (a) and (c) The "best" and "blended" rendering methods. For clarity, the underlying unique color assignments for the model images are given in (b) and (d), respectively.

reliability. For comparison, the original default rendering of these two model perspectives is given in Figs. 11(c) and 11(d), respectively. The underlying triangle meshes are also given in Figs. 11(e) and 11(f). As shown from these figures, the problems with stretched triangles and overlapped textures have been visibly reduced using the method proposed in this paper.

In addition to synthetic data, rendering performance is also demonstrated against a set of real-world data, given in Fig. 12. The model shown in Fig. 12 is composed of seven different 2.5-D texel images, which were captured sequentially in the CAIL laboratory using a handheld texel camera, and registered using the method described by Khatiwada and Budge.[4] The concept of weighted reliability is showcased using the "best" and "blended" texture assignment methods. For clarity, Figs. 12(b) and 12(d) have a unique color assigned to each image, rather than a texture.

6 Conclusion

Standard rendering techniques are not suitable for texturing 3-D texel models. Due to the independent nature of the underlying datasets, corrections to model textures must be done at render time. The most reliable texture, for a given perspective, is better defined using 3-D geometric criteria in conjunction with a real-time, view-dependent ranking algorithm. The method described in this paper has shown promise in correctly rendering 3-D texel models. Improved results were observed when using higher-resolution texel models.

References

1. C. Killpack and S. E. Budge, "Visualization of 3D images from multiple texel images created from fused lidar/digital imagery," *Proc. SPIE* **9465**, 94650I (2015).
2. R. Pack, P. Israelsen, and K. Sealy, "A co-boresighted synchronized ladar/EO imager for creating 3D images of dynamic scenes," *Proc. SPIE* **5791**, 42–50 (2005).
3. S. E. Budge and N. S. Badamikar, "Calibration method for texel images created from fused lidar and digital camera images," *Opt. Eng.* **52**, 103101 (2013).
4. B. Khatiwada and S. E. Budge, "Three-dimensional image reconstruction using bundle adjustment applied to multiple texel images," *Proc. SPIE* **9832**, 98320S (2016).
5. S. E. Budge, N. S. Badamikar, and X. Xie, "Automatic registration of fused lidar/digital imagery (texel images) for three-dimensional image creation," *Opt. Eng.* **54**, 031105 (2015).
6. V. Lempitsky and D. Ivanov, "Seamless mosaicing of image-based texture maps," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '07)*, pp. 1–6, IEEE, New York (2007).
7. P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based

approach," in *Proc. of the 23rd Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pp. 11–20, ACM, New York (1996).
8. P. Debevec, Y. Yu, and G. Boshokov, "Efficient view-dependent image-based rendering with projective texture-mapping," Technical Report, EECS Department, University of California, Berkeley (1998).
9. P. Debevec et al., "Image-based modeling and rendering of architecture with interactive photogrammetry and view-dependent texture mapping," in *Proc. of the 1998 IEEE Int. Symp. on Circuits and Systems (ISCAS '98)*, Vol. **5**, pp. 514–517, IEEE, New York (1998).
10. C. Buehler et al., "Unstructured lumigraph rendering," in *Proc. of the 28th Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*, pp. 425–432, ACM, New York (2001).
11. Y. Alj et al., "Space carving MVD sequences for modeling natural 3d scenes," *Proc. SPIE* **8290**, 829005 (2012).
12. Y. Alj et al., "Multi-texturing 3D models: how to choose the best texture?" in *2012 Int. Conf. on 3D Imaging (IC3D)*, pp. 1–8, IEEE, New York (2012).
13. M. Iiyama, K. Kakusho, and M. Minoh, "Super-resolution texture mapping from multiple view images," in *2010 20th Int. Conf. on Pattern Recognition (ICPR)*, pp. 1820–1823 (2010).
14. L. Wang et al., "Optimal texture map reconstruction from multiple views," in *Proc. of the 2001 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR 2001)*, Vol. **1**, pp. I–347–I–354, IEEE, New York (2001).
15. Y. Yu, "Efficient visibility processing for projective texture-mapping," *J. Comput. Graphics* **23**(2), 245–253 (1999).
16. J. Huang and M. B. Carter, "Interactive transparency rendering for large cad models," *IEEE Trans. Visualization Comput. Graphics* **11**(5), 584–595 (2005).

Cody C. Killpack received his BS degree in computer engineering from Brigham Young University, Idaho, in 2011 and his MS degree in computer engineering from Utah State University in 2016.

Scott E. Budge received his BS, MS, and PhD degrees in electrical engineering from Brigham Young University in 1984, 1985, and 1990, respectively. He joined the faculty at Utah State University in January 1989. His general research interests include signal and image processing and sensor systems. His current research interests include the methods for exploitation of full-waveform LADAR and EO images for 3-D image applications and pattern recognition.