Pacific Graphics 2014
J. Keyser, Y. J. Kim, and P. Wonka
(Guest Editors)

Volume 33 (2014), Number 7

Efficient Depth Propagation for Constructing a Layered Depth Image from a Single Image

S. Iizuka Y. Endo Y. Kanamori J. Mitani Y. Fukui

University of Tsukuba

Abstract

In this paper, we propose an interactive technique for constructing a 3D scene via sparse user inputs. We represent a 3D scene in the form of a Layered Depth Image (LDI) which is composed of a foreground layer and a background layer, and each layer has a corresponding texture and depth map. Given user-specified sparse depth inputs, depth maps are computed based on superpixels using interpolation with geodesic-distance weighting and an optimization framework. This computation is done immediately, which allows the user to edit the LDI interactively. Additionally, our technique automatically estimates depth and texture in occluded regions using the depth discontinuity. In our interface, the user paints strokes on the 3D model directly. The drawn strokes serve as 3D handles with which the user can pull out or push the 3D surface easily and intuitively with real-time feedback. We show our technique enables efficient modeling of LDI that produce sufficient 3D effects.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—

1. Introduction

3D scene reconstruction from a single image has been a challenging problem in computer graphics and computer vision because of the ambiguities related to depth and occlusions behind foreground objects. To that end, a number of single-view modeling methods that automatically or semi-automatically create simple scene models composed of a few polygons have been discussed [DTM96, HAA97, CRZ00, KPiAS01,HEH05a,SSN09,IKMF11]. Other techniques construct free-form scene models that have smoothly varying surfaces [OCDD01, ZDPSS02]. However, the former methods are only applicable to a limited number of scene types, such as outdoor settings that have extensive flat ground surfaces, while the latter methods require complicated user inputs, and thus can take inordinate amounts of time.

Our 3D scene representation is in the form of a Layered Depth Image (LDI) [SGHS98] composed of a foreground layer and a background layer, and each layer has a corresponding texture and depth map. Thanks to the background layer, LDI representation prevents large "holes" from appearing behind foreground objects when the viewpoint is moved.

Our method is designed to fulfill the following three requirements for constructing a LDI. First, we require users' depth inputs to exploit human perception to estimate scene depth, but the inputs should be as sparse as possible to reduce users' burden. We adopt sparse depth strokes as inputs. Although existing optimization-based approaches [WLF*11] also use depth strokes, they suffer from slow convergence and insufficient propagation of user-specified depth to distant pixels, which makes many scribbles needed. Second, the resultant depth map should be smooth even if the input image contains textures or noise while preserving sharp depth discontinuities. The depth details often cause 3D surfaces bumpy, as demonstrated in Figure 9, and therefore should be avoided in most cases. Third, the system should provide the user sufficiently fast and intuitive feedback for interactive editing.

Our coarse-to-fine approach satisfies these requirements as follows. The user can draw depth strokes on the image, and then intuitively manipulate already-drawn stroke depth by pulling out or pushing the stroke as a 3D handle. The stroke depth is then propagated to the rest of the regions instantly based on interpolation with geodesic-distance weighting and optimization-based smoothing. This process is computed using small homogeneous regions called superpixels to avoid creating noisy surfaces and to re-

Figure 1: *3D construction results. For each image pair, the left is the input and the right is the generated 3D model.*

duce computational costs. Our method also detects the depth discontinuities automatically, which are used for improving the edge-preserving depth propagation and the resultant model geometry, i.e., removing polygons at depth discontinuities. After depth assignment for the foreground layer, a background layer is constructed automatically based on the discontinuities. 3D coordinates are computed by assigning a depth value to each pixel along a viewing ray. Finally, we create a 3D scene by simply connecting adjacent pixels as polygons and then mapping the foreground and background textures onto the corresponding polygonal meshes.

Our main contributions are summarized as follows:

- The overall design of an interactive technique for creating a LDI from a single image via simple and sparse user inputs, and
- The efficient propagation of depth values using superpixel-based interpolation with geodesic-distance weighting followed by optimization.

We demonstrate satisfactory 3D models as well as interesting applications of depth maps obtained from various images using our technique.

2. Related work

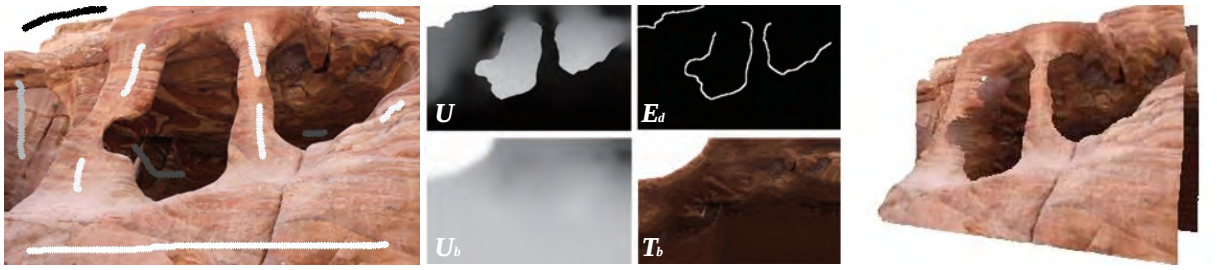
There have been numerous research efforts aimed at creating 3D scenes from single images. Debevec et al. [DTM96] reported a method for constructing a building model by applying primitives that are determined by the user based on the building outlines in an input image. Several methods compute simple models that are composed of a few polygons based on parallel lines or a vanishing point specified by the user [HAA97, CRZ00]. Kang et al. [KPiAS01] described a semi-automatic method for creating 3D models using a vanishing line, such as a horizon line, which can be applied to panoramic pictures. Iizuka et al. [IKMF11] proposed an interactive system that constructs models using a user-drawn boundary line between a ground region and other regions (e.g., buildings or sky) in order to compute the 3D coordinates.

Hoiem et al. [HEH05a] proposed a fully-automatic single-view modeling method under an assumption that the input image is composed of three main regions, specifically, "ground", "vertical", and "sky". This method is improved for estimating rough geometry of 3D scenes [HEH05b, HEH07] and occlusion boundaries [HEH11]. They construct a 3D scene model by labeling superpixels using machine learning. Saxena et al. [SSN09] created 3D scenes automatically by estimating a set of plane parameters such as 3D orientation using Markov Random Field via supervised learning. However, even though these methods do not require user inputs, they cannot create smooth surfaces due to the simplicity of 3D models.

In contrast to the automatic approaches above, Oh et al. [OCDD01] proposed a system that allows the user to interactively assign depth values to each region. This method can assign a depth value to each pixel and extract layers using a brush interface. Additionally, there are several methods [ZDPSS02, LGG14] for reconstructing free-form models from a single image. They use a number of shape constraints specified by the user for generating smooth 3D surfaces. Although these methods can create smooth detailed surfaces, they require many complicated user inputs and thus take significant amounts of editing time. Assa and Wolf [AW07] presented a semi-automatic diorama construction method that focuses on intensification of the depth perception using large-scale depth cues such as atmospheric scattering and the amount of defocus blur on the objects. However, their method does not aim at 3D reconstruction.

Various techniques for modeling an object observed in an image are also proposed. They focus on particular shapes such as symmetric architecture [JTC09], curved surfaces [OTC12, TNC13], and sweep objects [CZS*13], and then reconstruct the 3D models using several constraints (e.g., object's silhouette, relative volume constraints, or primitive assignments). Although these methods can reconstruct plausible 3D models of target objects, they do not consider a background region including occluded regions behind the objects in an image.

Wang et al. [WLF*11] proposed an interactive technique for creating stereoscopic contents from a single image by assigning depth values to image regions with scribbles via optimization. In video editing, such scribble-based depth propagation is used for creating a depth map [RCK*12]. Yücer et al. [YSHSH13] proposed an interactive depth assignment



(a) User inputs (b) Results (c) 3D scene model

Figure 2: *Overview of our system. (a) The user first specifies depth values, after which (b) our system automatically generates a depth map U, discontinuities E_d along with a depth map and texture for the background that includes the occluded region U_b and T_b and in real time. (c) Using these elements, a 3D model can be constructed. The input image is obtained from [WLF*11].*

using transfusive content-aware weight functions with inequality constraints. Such scribble-based selection is simple and adopted by other applications such as a painting tool for cartoon drawings [SDC09]. Our work is inspired by these methods in terms of the use of simple and sparse inputs. Although they require only simple operations, their optimization-based propagation often suffers from spreading input depth to the rest of the image in case of few scribbles. Considering the interface, depth assignment using scribbles on 2D images with 2D feedback of the depth map is not particularly intuitive because the relative depth of each scribble is difficult to recognize using grayscale values or pseudo colors that correspond to the depth values.

In our system, we improve both the quality and speed of depth map generation based on geodesic distance-based propagation and optimization-based edge-aware smoothing in terms of superpixels. Unlike the existing methods that compute only a single depth map, we further generate both the depth and texture of the occluded regions behind foreground objects for constructing more plausible 3D scene models. With our interface, the user can draw depth scribbles directly on the 3D surface and manipulate already-drawn scribbles as 3D handles to modify the 3D surface intuitively while receiving real-time 3D feedback.

Edit propagation methods based on all-pair constraints [AP08, XLJ*09] are also powerful tools to propagate sparse user edits to the entire image. Although they are often effective for propagating color or tone adjustments to pixels with similar appearances, they are unsuited for depth propagation because depth values often vary between distant pixels even though their appearances are quite similar, as demonstrated in Section 6.2.

3. Overview

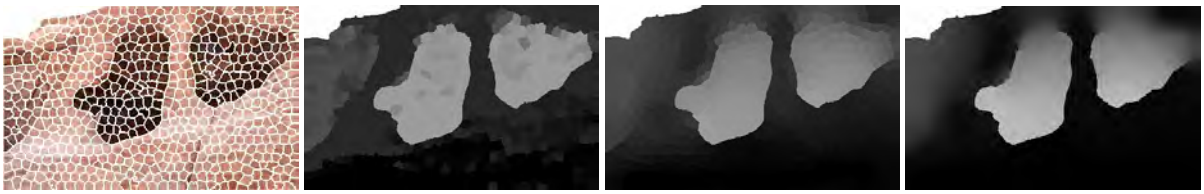
Figure 2 shows the overview of our system. The user first specifies sparse depth scribbles on the input image, and then our system automatically computes the following information for constructing a LDI:

- The depth map of the entire image,
- Discontinuities of image regions, and
- The depth map and texture of background regions, including the occluded regions.

During the preprocessing phase, we segment an input image to superpixels, which are used as the basis to compute geodesic distances to the depth strokes (Section 4.1) and edge-aware optimization (Section 4.2). The superpixel-wise depth map is then converted to a pixel-accurate depth map by removing gaps between neighboring superpixels by solving Laplace's equation (Section 4.3). This smoothing is avoided around depth discontinuities, which are detected from the color edges of the input image and the depth map before smoothing (Section 5.1). These depth propagation and discontinuity detection processes are performed in real time, which allows the user to edit the resultant model interactively. Finally, background regions behind the foreground objects are estimated automatically from the discontinuities (Section 5.2), after which the depth and texture maps are generated (Section 5.3). We describe these processes in detail in the following sections.

4. Scene depth construction

To compute the depth map, our system first computes superpixels (Figure 3(a)) of the input image. Using superpixels instead of pixels, we can not only reduce computational cost but also avoid undesired propagation caused by noises in natural images or paintings. We use the simple linear iterative clustering (SLIC) [ASS*12] to generate more compact and regularly-shaped superpixels than those of previous methods (e.g., graph-based algorithm [FH04]). This is important when computing the superpixel-based distances to alleviate distance errors related to superpixel size. In our results, we set the number of superpixels as one percent of that of pixels.



(a) Superpixels (b) Geodesic distance (c) Optimization (d) Resultant depth map

Figure 3: *Computing a depth map. The user inputs are shown in Figure 2(a). (a) An input image is segmented into superpixels. (b) Next, depth values are computed based on superpixel-based geodesic distances. (c) Then it is smoothed by optimization. (d) Finally, the gaps of the depth values between superpixels are removed by solving Laplace's equation.*

Our system computes a superpixel-wise depth map in two steps. First, a rough depth map is computed based on geodesic distance-based propagation (Figure 3(b)) that can efficiently propagate depth values while considering color edges in the image. Geodesic distances can be computed in linear time. Then, the rough depth map is used as the initial solution and the data term in optimization (Figure 3(c)) where the data term defines the overall shape whereas the smoothness term ensures local smoothness of the depth map. After the optimization, we remove depth gaps between neighboring superpixels to obtain a pixel-wise depth map by solving Laplace's equation (Figure 3(d)).

4.1. Computing depth based on geodesic distances

To compute the initial depth value of each superpixel, we compute geodesic distances from each superpixel to depth scribbles, and then blend the scribbles' depth values according to their geodesic distances (Figure 3). Geodesic distance has been used for image editing techniques such as image colorization [YYSS04] and foreground object extraction [CSB08, BS09].

Chaurasia et al. [CDSHD13] used superpixel-based geodesic distance for depth synthesis in poorly reconstructed regions of multiview stereo. The crucial difference between ours is that they only use it for finding superpixels that would belong to the same object and they blend depth inputs based on Euclid distance, completely ignoring color variations that reflect surface details. Consequently, their method yields piecewise flat depth maps for sparse depth inputs, as demonstrated in Figure 9.

Here we introduce our depth interpolation technique based on geodesic distance. We measure the distance between two adjacent superpixels as the sum of squared differences (SSD) of pixel values that have been randomly sampled from each superpixel. Random sampling is used here to consider color variations within each superpixel.

Suppose that the user inputs L strokes with different depth values, where stroke $l(=1,2,...,L)$ has depth value $d_l \in [0,1]$. The larger depth values are assigned to pixels far from the camera. Note that grayscale values of depth inputs in our figures in this paper are reversed just for the display purpose to prevent white strokes from being hidden in white backgrounds. Let Ω_l be a set of superpixels that include stroke l. Superpixels in Ω_l have the same depth d_l. The geodesic distance D_l from stroke l to every superpixel S_k is computed using the following equation:

$$D_l(S_k) := \min_{S_j \in \Omega_l} dist(S_j, S_k), \tag{1}$$

$$dist(S_j, S_k) := \min_{C(S_j,S_k)} \sum_{S_x,S_y} W_{S_xS_y}, \tag{2}$$

where $C(S_j,S_k)$ is a path connecting the superpixels S_j and S_k, and S_x and S_y represent adjacent superpixels on the path $C(S_j,S_k)$. The weight $W_{S_xS_y}$ is the SSD of pixel values in *Lab* color space that are randomly sampled from the superpixels S_x and S_y.

$$W_{S_xS_y} = \sum_{s \in S_x, t \in S_y} \|\mathbf{c}_s - \mathbf{c}_t\|^2, \tag{3}$$

where s and t are pixels sampled from superpixels S_x and S_y, $\mathbf{c}_s$ and $\mathbf{c}_t$ are the pixel values in *Lab* color space. Following the algorithm [YBS05], the geodesic distances can be computed in optimal linear time. Because unlike previous methods [YYSS04, CSB08, BS09] we use superpixels instead of pixels and the number of superpixels is much smaller than that of pixels, we can significantly accelerate the computation of geodesic distances. We then compute the depth value $G(S_k)$ of the superpixel S_k by blending scribbles' depth values according to geodesic distances:

$$G(S_k) = \frac{\sum_l D_l(S_k)^{-b} d_l}{\sum_l D_l(S_k)^{-b}}, \tag{4}$$

where b is the constant value to determine the degrees of the impact of the geodesic distances. If the value is greater, the input depth values of the smaller geodesic distances are emphasized. We use $b = 2$ in our system.

Figure 3(b) shows that this geodesic distance-based approach efficiently propagates depth inputs to the rest of the image even with fewer scribbles than [WLF*11]. However,

the resultant depth map is locally coarse, and thus we apply optimization-based smoothing in the next step.

4.2. Optimization

We apply edge-preserving smoothing to the superpixel-wise rough depth map G via optimization as follows. We set up a cost function $E(U)$ regarding unknown depth map U with data term $E_{data}(U)$ and smoothness term $E_{smooth}(U)$. The data term $E_{data}(U)$ makes depth map U closer to the rough depth map G whereas the smoothness term $E_{smooth}(U)$ tries to equalize the depth of S_i with the weighted average depth of S_i's neighbors $N(S_i)$.

$$E(U) = E_{data}(U) + \gamma E_{smooth}(U), \quad (5)$$

$$E_{data}(U) = \sum_{S_i} (U(S_i) - G(S_i))^2, \quad (6)$$

$$E_{smooth}(U) = \sum_{S_i} \left(U(S_i) - \sum_{S_j \in N(S_i)} w_{S_i S_j} U(S_j) \right)^2, \quad (7)$$

where γ is a weight used to adjust the degree of smoothing. $w_{S_i S_j} \propto \exp(-W_{S_i S_j}/2\sigma^2)$ is a weight and $\sum_{S_j \in N(S_i)} w_{S_i S_j} = 1$. We use $\gamma = 30$ and $\sigma = 0.5$. By initializing the depth map U with G, simple Gauss-Seidel iteration converges very quickly.

4.3. Per-pixel depth assignment

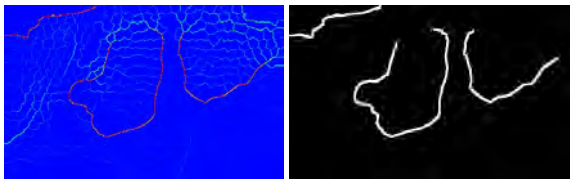
The depth map we obtained so far is superpixel-wise and thus the depth within each superpixel is constant. For eliminating the depth gaps between neighboring superpixels, the standard edge-preserving smoothing filters such as bilateral filter are unsuitable because they do not strictly preserve particular values, e.g., depth along discontinuities. Now we obtain a pixel-wise smooth depth map by solving Laplace's equation with appropriate boundary conditions, regarding depth value d_p of pixel p over the gap region Ω:

$$\Delta d_p = 0 \ over \ \Omega, \ with \ d_p|_{\partial\Omega} = d_p^0, \quad (8)$$

where d_p^0 are fixed values used for Dirichlet boundary conditions, and are calculated as follows. We search for junctions of three or more adjacent superpixels and fix the depth values of pixels around each junction as their average depth. We also fix the depth values along depth discontinuities to preserve depth edges. The computation of depth discontinuities is described in Section 5.1.

5. LDI construction

Now we describe our LDI construction process, including the discontinuity detection, background detection, as well as the background depth and texture construction steps. We first assume that, if the depths of two adjacent superpixels are sufficiently different, then the two superpixels should be disconnected at their common edge. We perform such discontinuity detection using the depth and image edges. Note that depth discontinuities are used for computing a pixel-accurate depth map (Section 4.3) and are therefore computed beforehand. These discontinuities are also used to detect background regions behind foreground objects. Both the depth and texture maps of the detected background region are then computed automatically. Adding the background layer to the foreground layer, which is directly computed from the input image and the depth map, a complete LDI is constructed.



(a) Edge strength (b) Discontinuities

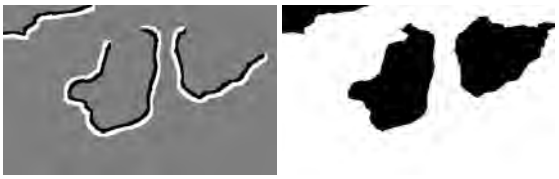
Figure 4: *Discontinuity detection. (a) We first define edge strength using edges of a depth map and edges of an image. (b) Then, discontinuities are extracted by thresholding.*

5.1. Discontinuity detection

To construct a plausible LDI model, we find depth discontinuities and then cut the surface along them. To accomplish this, we define an edge strength E_s using an image edge E_i and a depth edge E_d with the weights w_i and w_d (i.e., $E_s = w_i E_i + w_d E_d$). We use $w_i = 0.2$ and $w_d = 0.8$. These edges are detected using a standard Laplacian filtering. Discontinuities are then detected by applying a threshold to E_s (Figure 4(b)). The threshold value is 0.7 in our implementation. In our model, we apply a fast image matting [BS09] to the boundaries of the foreground regions around the discontinuities to improve the appearance.

5.2. Occluded region detection

To construct the background layer including occluded regions, we need to extract the occluded regions first. Such occluded regions are overlapped by foreground regions, and thus we can specify occluded regions by extracting foreground regions. One concern here is that object surfaces might seamlessly transit from foreground to background, as shown in the cave in Figure 2(a), and thus boundaries between foreground and background might be ambiguous. We determine such potentially-ambiguous boundaries by a segmentation technique based on depth boundaries; we assume that at least one side of each discontinuity with larger depth is in background and the other side with smaller depth belongs to foreground. We thus construct a trimap where either side of each discontinuity is labeled as absolutely "foreground" or "background" and the rest is labeled as "unknown" (Figure 5(a)), and then solve a binary labeling problem using the geodesic segmentation [BS09] (Figure 5(b)).



(a) Trimap (b) Foreground region

Figure 5: *Occluded region detection. (a) We create a trimap composed of foreground (white), background (black), and unknown (gray) regions using the discontinuities. (b) The foreground region coincides with the occluded region, and is extracted in real time by the geodesic segmentation technique.*

The trimap is constructed as follows. First, we dilate each discontinuity using a morphological operation (five-pixel wide in our results). Next, we construct a histogram of depth values in each discontinuity band and separate pixels within it by thresholding. The threshold value is computed by the discriminant analysis method [Ots79]. Pixels that have smaller depth values than the threshold are labeled as "foreground" whereas pixels that have larger values are labeled as "background".

5.3. Texture and depth maps of occluded region

After detecting the occluded regions, we consider the regions as "holes" to be inpainted and compute texture and depth maps for them. We first generate a full background texture including the occluded regions (Figure 6(a)) using the fast inpainting method [BSFG09]. Next, the depth of the occluded region is computed. We initialize a background depth map by copying the depth map of the image except for the occluded region, and then solve Equation (5) to fill the occluded region (Figure 6(b)) with referring to depth values of non-foreground scribbles.

5.4. 3D visual feedback

In our system, the user can paint depth values onto a 3D surface using a brush directly with 3D visual feedback. Already-drawn brush strokes stay on the 3D surface, and can be used as 3D control handles to further manipulate the 3D surface intuitively. During this editing session, the user-specified depth values are propagated instantly together with automatic discontinuity detection. The user can verify the resultant 3D shape in real time. The accompanying video shows a real-time demonstration.

The idea to use already-drawn strokes as 3D control handles is similar to that of the sketch-based 3D modeling technique called *FiberMesh* [NISA07]. Integrating the advanced mesh editing operations into our system would be an interesting direction of future work.

(a) Background texture (b) Background depth

Figure 6: *Texture and depth of a background with occluded regions. Propagating depth values on (a) the background image, and (b) a depth map, allows the background to be created.*

6. Results

We implemented our prototype system with C++, and ran the program on a PC equipped with a 3.4 GHz CPU and 8 GB of memory. The sizes of input images are all in the range of 0.3 to 1 megapixels. In our experiments, the average computation time for geodesic distances was approximately 0.01 seconds, optimization was approximately 0.01 seconds, gap removal was approximately 0.1 seconds, and the total time required for depth propagation and a simple 3D model construction (including discontinuity extraction) was within 0.2 seconds.

Thanks to the fast computation and simple interface, the user can edit the 3D model interactively. After the simple model construction, the background layer is generated automatically. The average time required for creating the background layer, including the texture and depth computation, was approximately four seconds. The total time, including manual operations, was within three minutes for all the results shown in this paper. In our current system, these computations are all processed in a single CPU core.

3D construction. Figures 1, 2 and 7 show the results of 3D construction using our system. We can see that 3D models with smoothed surfaces are created successfully, taking into account the object shapes in the input images. Additionally, our system separates the discontinuities and generates background regions behind the objects in the scene. When combined with the background, these texture-mapped smooth models provide satisfactory 3D effects to the viewer. In our experience, we obtain plausible results by drawing depth scribbles along ambiguous boundaries and across a color-varying region on the same object. Note that our method can represent curved objects as demonstrated in the candy example in Figure 7 and the portrait example in Figure 9(e). Adding more scribbles allows more detailed curved surfaces.

6.1. Applications

The resultant depth map obtained using our system is useful for various image editing processes that require scene depth. Here we show just a few examples.

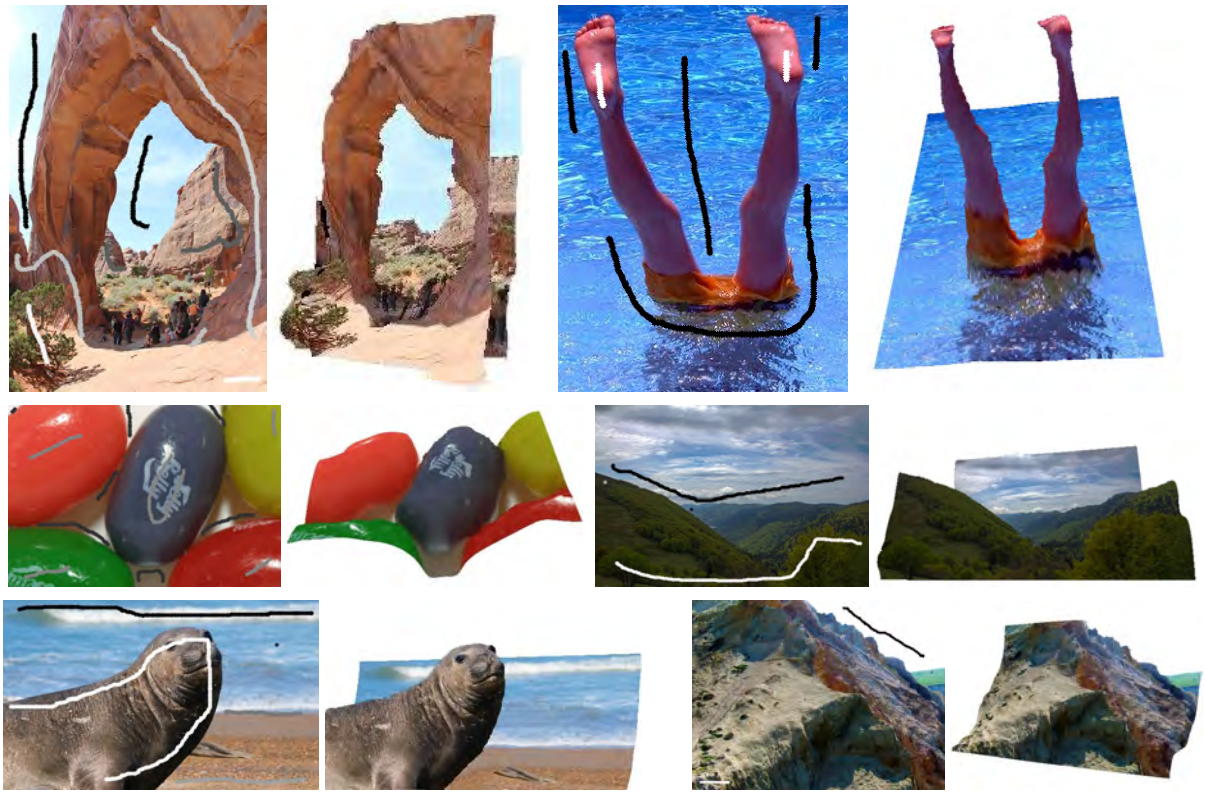


Figure 7: *3D construction results. For each image pair, the left is the input and the right is the generated 3D model.*

(a) User inputs (b) Depth map (c) Anaglyph (d) Depth-of-field (e) Fog synthesis

Figure 8: *Applications with the resultant depth map.*

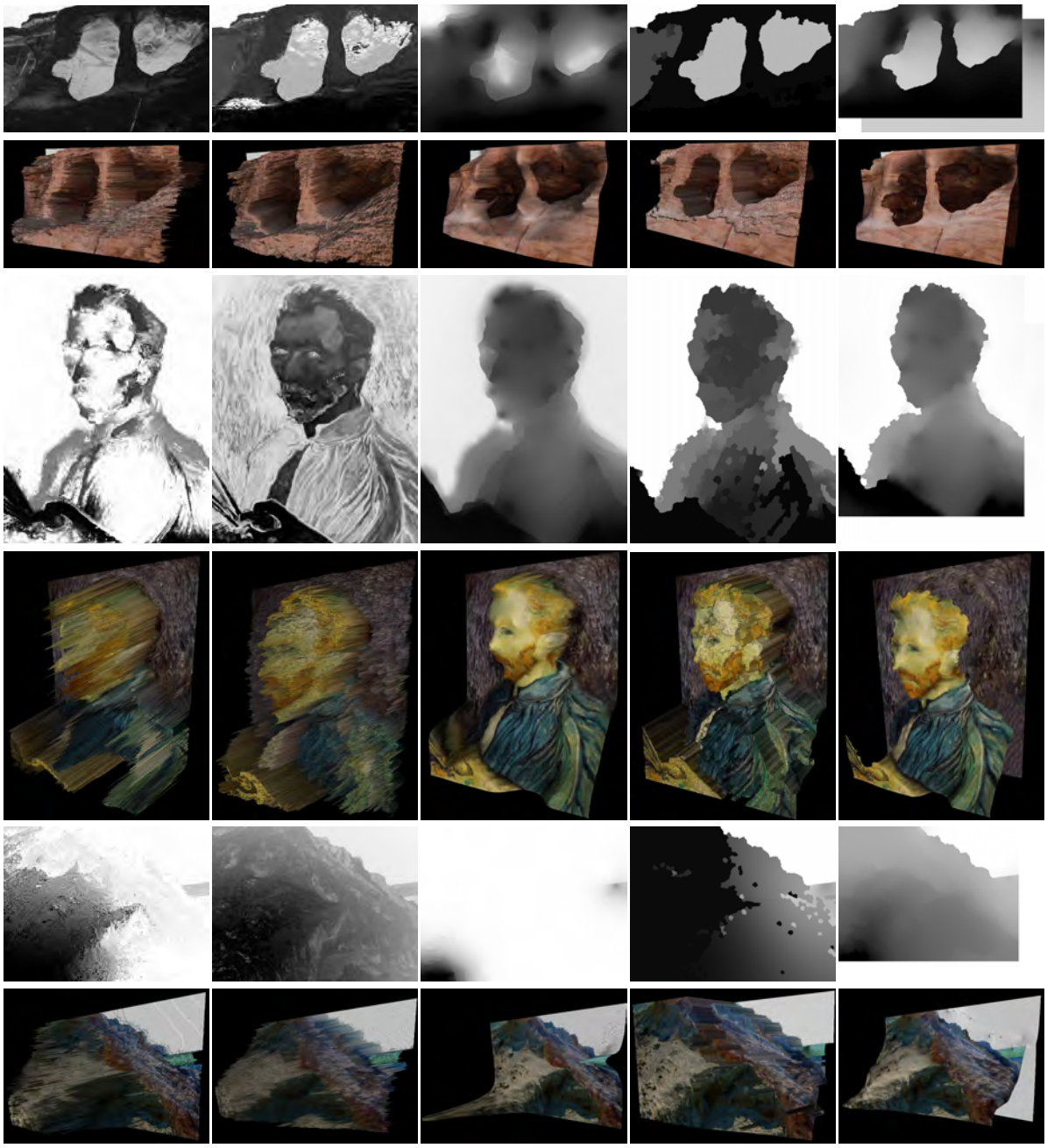
Stereo imagery is one of popular applications that take advantage of scene depth. Our depth map can be used to create stereoscopic images using Wang et al.'s method [WLF*11]. Figure 8(c) shows the result of stereo as an anaglyph image.

The depth map can be used for a depth-of-field effect to enhance a target region in the image. The user specifies the target region, and then our system automatically blurs other regions based on the depth map. Figure 8(d) shows the result of this effect, where we focus on the right moai statue.

Our method can also simulate aerial perspective to enhance the perceived depth by synthesizing fog or haze according to the scene depth. We can produce a haze image as shown in Figure 8(e) using the standard model, i.e., $\mathbf{I}(\mathbf{x}) = t(\mathbf{x})\mathbf{J}(\mathbf{x}) + (1 - t(\mathbf{x}))\mathbf{A}$, where $\mathbf{I}$ is the output image, $\mathbf{J}$ is the input image, $\mathbf{A}$ is the fog color, $t(\mathbf{x}) = e^{-\eta d(\mathbf{x})}$ is an interpolation parameter, η is the scattering coefficient, and d is the depth value of pixel $\mathbf{x}$. We set $\mathbf{A} = (0.9, 0.9, 0.9)$ and $\eta = 0.06$ in the result.

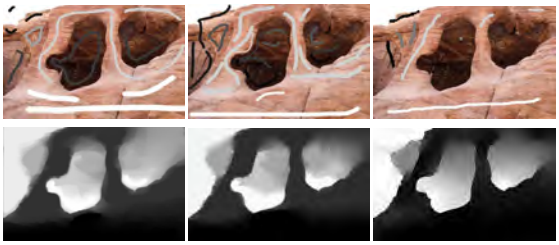
6.2. Comparisons

Figure 9 compares the depth maps and 3D models created by our method and the previous methods for edit propagation. The resultant depth maps of Chen et al. [CZZT12] and Li et al. [LJH10] are locally coarse because it propagates the input depth to all regions with similar appearance globally without considering the local smoothness spa-



(a) [CZZT12] (b) [LJH10] (c) [WLF*11] (d) [CDSHD13] (e) Ours

Figure 9: *Comparisons of depth maps and the 3D models. The user inputs are shown in Figures 1, 2 and 7. Whereas (a)(b)(c)(d) the previous methods suffer from propagating depth, (e) our method propagates depth along the image content plausibly.*



(a) [WLF*11] (b) [YSHSH13] (c) Ours

Figure 10: *Comparisons of user inputs (top row) for creating similar depth maps (bottom row). Compared to the previous methods (a)(b), (c) our method can generate a similar but smoother depth map with sharp depth discontinuities from much less depth scribbles. The images of the previous methods are taken from their papers.*

(a) User Inputs (b) Pixel-based (c) Ours

Figure 11: *Comparison of pixel-based propagation and our coarse-to-fine propagation. (a) Given sparse depth inputs, (b) compared to the pixel-basis propagation that is sensitive to textures, (c) our coarse-to-fine propagation can generate a smoothly varying depth map instantly.*

tially, which is common to the all-pair constraint methods [AP08, XLJ*09]. Their results are jaggy because they are too sensitive to underlying textures. The method of Wang et al. [WLF*11] propagates depth by optimization based on the assumption that neighboring pixels have similar colors. Our method propagates depth more efficiently compared to their method that suffers from spreading depth values to far pixels, as discussed in Section 2. The results of Chaurasia et al. [CDSHD13] are coarse and piecewise flat, as mentioned in Section 4.1. Figure 10 compares the amount of user inputs for creating similar depth maps. Compared to the previous methods [WLF*11, YSHSH13], our method requires much less inputs and generate smoother surfaces even at textured regions, while preserving the depth discontinuities.

In terms of computational time, our method requires approximately 0.2 seconds to process a 1000×620 image on a single CPU core, the methods of Chen et al. and Li et al. require more than 1 minute and 2.5 seconds respectively in our implementation. The optimization-based method of Wang et al. requires approximately 10 seconds with Matlab implementation. Chaurasia et al.'s method takes about 5 seconds depending on the number of superpixels with depth inputs. In the method of Yücer et al. [YSHSH13], their paper states that the depth computation requires 5-15 seconds in a 250×250 image depending on the number of scribbles.

Figure 11 shows a comparison of pixel-based propagation and superpixel-based propagation in our framework. Our coarse-to-fine scheme using superpixels can significantly reduce the computational time (about 10 times) and generates a smoother depth map compared to pixel-basis computation.

6.3. Limitations

Our method cannot reconstruct analytic shapes such as spheres or cones. Additionally, images with complicated scenes may increase required user inputs. For example, in a cluttered scene with numerous objects and occluded regions, our technique may encounter problems with occluded region detection and texture completion. In such cases, it will be necessary to specify additional inputs to produce more accurate region detection and texture completion.

7. Conclusion

In this paper, we have presented an interactive system for constructing a LDI, including the foreground layer and background layer, via sparse user edits. Our system computes scene depth using superpixel-based geodesic distances and optimization, and then produces a pixel-accurate depth map by smoothing the gaps between superpixels, which results in efficient depth propagation compared to previous methods. Additionally, our method allows automatic synthesis of depth and texture maps of the background regions behind foreground objects in the scene, which enhances the reality of the resultant 3D models. The resultant depth maps as well as 3D models using our method can be used to a number of applications such as stereo content, depth-of-field effects, and fog synthesis.

In future work, we would like to extend our technique to other applications which require depth information such as 3D object insertion and the animation. We would also like to apply our technique in ways that facilitate the propagation of user edits such as color or tone, and generate not only foreground and background layers but also multiple layers for more complicated scenes.

Acknowledgements

This work was supported in part by a Grant-in-Aid for JSPS Fellows. The authors would like to thank the following Flickr users for Creative Commons imagery: longhorndave, isawnyu, yeowatzup, LennartTange, daverynin, and Bob Detent.

References

[AP08] AN X., PELLACINI F.: AppProp: All-pairs appearance-space edit propagation. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 40:1–40:9. 3, 9

[ASS*12] ACHANTA R., SHAJI A., SMITH K., LUCCHI A., FUA P., SU S.: SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 11 (2012), 2274–2282. 3

[AW07] ASSA J., WOLF L.: Diorama construction from a single image. *Comput. Graph. Forum 26*, 3 (2007), 599–608. 2

[BS09] BAI X., SAPIRO G.: Geodesic matting: A framework for fast interactive image and video segmentation and matting. *Int. J. Comput. Vision 82*, 2 (2009), 113–132. 4, 5

[BSFG09] BARNES C., SHECHTMAN E., FINKELSTEIN A., GOLDMAN D. B.: PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph. (Proc. of SIGGRAPH) 28*, 3 (aug 2009), 24:1–24:11. 6

[CDSHD13] CHAURASIA G., DUCHENE S., SORKINE-HORNUNG O., DRETTAKIS G.: Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph. 32*, 3 (July 2013), 30:1–30:12. 4, 8, 9

[CRZ00] CRIMINISI A., REID I., ZISSERMAN A.: Single view metrology. *Int. J. Comput. Vision 40*, 2 (2000), 123–148. 1, 2

[CSB08] CRIMINISI A., SHARP T., BLAKE A.: Geos: Geodesic image segmentation. In *ECCV '08* (2008), Springer-Verlag, pp. 99–112. 4

[CZS*13] CHEN T., ZHU Z., SHAMIR A., HU S.-M., COHEN-OR D.: 3-sweep: Extracting editable objects from a single photo. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2013) 32*, 6 (2013), Article 195. 2

[CZZT12] CHEN X., ZOU D., ZHAO Q., TAN P.: Manifold preserving edit propagation. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 132:1–132:7. 7, 8

[DTM96] DEBEVEC P. E., TAYLOR C. J., MALIK J.: Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Proceedings of SIGGRAPH '96* (1996), ACM, pp. 11–20. 1, 2

[FH04] FELZENSZWALB P. F., HUTTENLOCHER D. P.: Efficient graph-based image segmentation. *Int. J. Comput. Vision 59*, 2 (Sept. 2004), 167–181. 3

[HAA97] HORRY Y., ANJYO K.-I., ARAI K.: Tour into the picture: using a spidery mesh interface to make animation from a single image. In *Proceedings of SIGGRAPH '97* (1997), pp. 225–232. 1, 2

[HEH05a] HOIEM D., EFROS A. A., HEBERT M.: Automatic photo pop-up. *ACM Trans. Graph. 24*, 3 (2005), 577–584. 1, 2

[HEH05b] HOIEM D., EFROS A. A., HEBERT M.: Geometric context from a single image. In *In ICCV* (2005), pp. 654–661. 2

[HEH07] HOIEM D., EFROS A. A., HEBERT M.: Recovering surface layout from an image. *Int. J. Comput. Vision 75*, 1 (Oct. 2007), 151–172. 2

[HEH11] HOIEM D., EFROS A. A., HEBERT M.: Recovering occlusion boundaries from an image. *Int. J. Comput. Vision 91*, 3 (Feb. 2011), 328–346. 2

[IKMF11] IIZUKA S., KANAMORI Y., MITANI J., FUKUI Y.: Efficiently modeling 3D scenes form a single image. *IEEE Computer Graphics and Applications 32*, 6 (2011), 18–25. 1, 2

[JTC09] JIANG N., TAN P., CHEONG L.-F.: Symmetric architecture modeling with a single image. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 113:1–113:8. 2

[KPiAS01] KANG H. W., PYO S. H., ICHI ANJYO K., SHIN S. Y.: Tour into the picture using a vanishing line and its extension to panoramic images. *Computer Graphics Forum 20*, 3 (2001). ISSN 1067-7055. 1, 2

[LGG14] LOPEZ A., GARCES E., GUTIERREZ D.: Depth from a single image through user interaction. In *Proceedings of CEIG* (2014), pp. 1–10. 2

[LJH10] LI Y., JU T., HU S.-M.: Instant propagation of sparse edits on images and videos. *Comput. Graph. Forum 29*, 7 (2010), 2049–2054. 7, 8

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: FiberMesh: Designing freeform surfaces with 3D curves. *ACM Trans. Graph. 26*, 3 (July 2007). 6

[OCDD01] OH B. M., CHEN M., DORSEY J., DURAND F.: Image-based modeling and photo editing. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), SIGGRAPH '01, ACM, pp. 433–442. 1, 2

[OTC12] OSWALD M. R., TOEPPE E., CREMERS D.: Fast and globally optimal single view reconstruction of curved objects. In *cvpr* (Providence, Rhode Island, June 2012), pp. 534–541. 2

[Ots79] OTSU N.: A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics 9* (1979), 62–66. 6

[RCK*12] RIBERA R. B. I., CHOI S., KIM Y., LEE J., NOH J.: Video panorama for 2D to 3D conversion. *Comp. Graph. Forum 31*, 7pt2 (Sept. 2012), 2213–2222. 2

[SDC09] SÝKORA D., DINGLIANA J., COLLINS S.: LazyBrush: Flexible painting tool for hand-drawn cartoons. *Computer Graphics Forum 28*, 2 (2009), 599–608. 3

[SGHS98] SHADE J., GORTLER S., HE L.-W., SZELISKI R.: Layered depth images. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (1998), SIGGRAPH '98, ACM, pp. 231–242. 1

[SSN09] SAXENA A., SUN M., NG A. Y.: Make3D: Learning 3D scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell. 31*, 5 (May 2009), 824–840. 1, 2

[TNC13] TÖPPE E., NIEUWENHUIS C., CREMERS D.: Relative volume constraints for single view 3D reconstruction. In *CVPR* (2013), pp. 177–184. 2

[WLF*11] WANG O., LANG M., FREI M., HORNUNG A., SMOLIC A., GROSS M.: StereoBrush: interactive 2D to 3D conversion using discontinuous warps. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling* (2011), SBIM '11, ACM, pp. 47–54. 1, 2, 3, 4, 7, 8, 9

[XLJ*09] XU K., LI Y., JU T., HU S.-M., LIU T.-Q.: Efficient affinity-based edit propagation using k-d tree. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 118:1–118:6. 3, 9

[YBS05] YATZIV L., BARTESAGHI A., SAPIRO G.: $O(N)$ implementation of the fast marching algorithm. *Journal of Computational Physics 212* (2005), 393–399. 4

[YSHSH13] YÜCER K., SORKINE-HORNUNG A., SORKINE-HORNUNG O.: Transfusive weights for content-aware image manipulation. In *Proceedings of the Vision, Modeling and Visualization Workshop (VMV)* (2013), Eurographics Association, pp. 57–64. 2, 9

[YYSS04] YATZIV L., YATZIV L., SAPIRO G., SAPIRO G.: Fast image and video colorization using chrominance blending. *IEEE Transctions on Image Processing 15* (2004), 2006. 4

[ZDPSS02] ZHANG L., DUGAS-PHOCION G., SAMSON J.-S., SEITZ S. M.: Single view modeling of free-form scenes. In *Proc. of CVPR* (2002), pp. 990–997. 1, 2