Dense and Occlusion-Robust Multi-View Stereo for Unstructured Videos

Jian Wei, Benjamin Resch, and Hendrik P. A. Lensch
Computer Graphics, Tübingen University
72076 Tübingen, Germany
Email: jian.wei@graphics.uni-tuebingen.de, {benjamin.resch|hendrik.lensch}@uni-tuebingen.de

***Abstract*—We present an efficient multi-view stereo system for dense reconstruction of homogeneous areas from videos with arbitrary camera trajectories. Most techniques have difficulties in recovering textureless areas, weak robustness for occlusion, and low efficiency on the massive redundant data provided by densely sampled frames. Our key idea is to use the measurable depth at object edges to recover the enclosed geometries assuming smooth surfaces, which is appropriate for most scenes. The depth values at edges are calculated by considering individual viewing rays to allow for sharp discontinuities. Then we employ perspective diffusion to create smooth surfaces between edges. The wrong interpolants between fore- and background edges are effectively invalidated by detecting large depth standard deviation which is approximated using the edge depth from other views. For reliable invalidation, the accuracy of edge depth is improved by advanced score evaluation and two-scale image sampling. Finally, we fill up holes and correct the wrongly sunken surfaces by propagating the closest valid depth across views. The pixel-level operations throughout support high parallelism, and our high-quality depth maps allow dense scene representation by point clouds.**

***Keywords*-multi-view stereo; depth map; homogeneous surface; dense reconstruction; diffusion; occlusion; GPU; video;**

I. Introduction

Multi-View Stereo (MVS) aims at 3D reconstruction from image sets. There have been considerable strides in modeling from sparse images with significantly view-dependent content [1]. As camera hardware improved, capturing a high-frame-rate video stream by arbitrarily moving a camera has received more attention since acquisition got simpler and faster. Light-field-based approaches [2] have been developed that exploit the data redundancy much better than classic MVS and are able to process the heavily increased amount of data. However, these techniques struggle on homogeneous areas which are ubiquitous in urban environments.

Because planar geometry is very common in the real world, we solve the problem by fitting the *smoothest* possible surface for textureless areas, even for curved surfaces (recovery of the correct shading is beyond the scope of this paper). We do not rely on high-level techniques like segmentation (*e.g.*, [3]) but concentrate on image operations which are most often evaluated per pixel. Hence, our approach can be efficiently implemented on GPUs.

Our pipeline is separated into two parts: First we calculate depth for the reliably detectable regions by advancing [2] towards higher depth resolution and more robustness on arbitrary camera paths. Next we diffuse edge depth to fill up homogeneous areas with perspectively correct interpolants, and afterwards we handle inconsistent surfaces.

Wei *et al.* [19] Kim *et al.* [2] Our approach

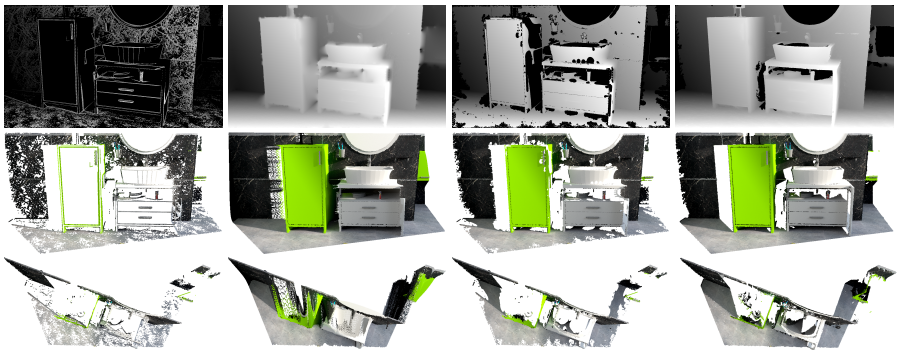
Figure 1. Points reconstructed by different methods. The proposed pipeline can recover dense and smooth surfaces in large homogeneous areas. We invalidate the unreliable surfaces by cross-view constraints, and merge the consistent subpixel-accuracy results from multiple views. Since the bottom edge on the back of the foreground box is invisible in almost all images, we fail to get correct interpolants for the table's surface behind thus getting a hole. Kim *et al.* [2] produce wrong surfaces for most unmeasurable areas.

Depth diffusion per view probably produces interpolants that connect fore- and background edges but occlude the real surfaces (see Fig. 2). As the occluded surfaces can be seen by another view, we invalidate these incorrect values by comparing their projected depth and corresponding edge depth (if available) in other views. Since the edge depths are sparse, we need to spread the reliably detected invalid pixels to the whole wrong surfaces between discontinuous objects. We achieve this using a diffusion-based approximation of the depth standard deviation. The invalidation stage creates holes where edge depths have been wrongly interpolated. However, it fails to remove the interpolants behind the real surfaces (see Fig. 8). We produce the final depth maps with both high completeness and high consistency by propagating the closest valid depth from other views.

Since our method requires reliable edge depth to avoid removing too many of the surfaces estimated by diffusion, we first focus on improving [2] (Sec. III) and then describe the individual steps of recovering homogeneous areas (Sec. IV).

II. Related Work

Fast reconstruction from videos. Recently a fast scalable Structure-from-Motion (SfM) pipeline [4] was designed that exploited the strong internal coherence in video data instead of traditionally used large image baselines. Simultaneous



Edge reconstruction | After diffusion | After invalidation | After propagation

Figure 2. Depth maps of one view (the top row) after individual steps of our reconstruction pipeline, and the corresponding point clouds viewed from the front (middle) and top (bottom) perspectives. The proposed approach successfully produces smooth and dense surfaces, with the wrong interpolants occluding the real surfaces removed.

Localization and Mapping (SLAM) algorithms are aimed at video input, but like SfM, most of them (*e.g.*, [5]) merely generated sparse detectable points, conveying little semantics. Some video-based MVS techniques [6], [7] realized fast or interactive reconstruction but they are only usable for object modeling or yield low coverage for homogeneous surfaces. Other methods [8], [9] have been introduced for fast and dense recovery of 3D scenes from video streams. Optical flow was also used to guide the reconstruction [9].

Dense 3D recovery. Advanced dense SLAM methods replaced [10], [11] or combined [3], [12] point features with planes. The segmentation effect can only be confined for initialization followed by refining the estimates using bundle optimization [13]. Some MVS pipelines [7], [14], [15] generated dense surfaces using energy minimization constrained by a smoothness term. Diffusion [16] or domain transform filter [17], [18] can be performed on the sparse, reliable estimates, like our approach. However, their depth interpolation methods are not perspective. Matching ambiguities in homogeneous areas can be solved by restricting computations in a multi-resolution manner [2], [15], [19], but might still produce depth discontinues for large homogeneous regions. Especially when occlusion occurs, the reconstructions of these techniques are not completely visibility-consistent.

Visibility consistency. Kim *et al*. [2] accounted for occlusion by propagating foreground depth to other views only if similar colors are exhibited. Consistent models can be obtained by simply removing the estimates which are inconsistent among small-baseline neighboring views or violate the occlusion consistency across large-baseline non-neighbors [7], [20]. To produce both dense and consistent reconstruction, bundle optimization [13], tetrahedra carving [21], or cross-view depth filtering [18], [19] has been adopted. In contrast, we propose a more parallelizable framework to handle visibility conflicts and global inconsistency.

III. Edge Depth Estimation

Object edges are typically aligned with color discontinuities, and thus easy to detect and to reconstruct from images by imposing the photometric constraints. Therefore, we first subsample the small-baseline image sequence for sufficient triangulation angles. Then the high-contrast edges are reconstructed for individual image samples, producing a set of edge depth maps $\boldsymbol{D}^e$.[1] For each primary image, we determine the edge depth using 100 neighboring images[2] (50 preceding and 50 following, respectively), and we also calculate a corresponding standard deviation (SD) map of the edge depth. The set of edge SD maps $\boldsymbol{S}^e$ is used for our depth invalidation (Sec. IV-B).

Sec. III-A first investigates the influence of unconstrained camera paths upon depth estimation using pixel-dependent color and score maps. We accordingly introduce a careful score aggregation method in Sec. III-B and a two-scale image sampling scheme in Sec. III-C for occlusion-robustness. Sec. III-D presents how we achieve subpixel precision and remove outliers. The SD of each edge depth estimate is calculated in Sec. III-E.

Pre-processing. We execute the SfM algorithm [4] to obtain the camera poses and feature points with inferred visibility constraints. Images are pre-smoothed by a 7×7 Gaussian filer with standard deviation of 0.5 to remove noise and aliasing. Edges are extracted if the color gradient magnitude calculated using a 3×3 Sobel operator is larger than 2.5.

A. Pixel-Wise Color and Score Maps

To attain precise depth estimates, we have to carefully deal with occlusion and out-of-image projection particularly for arbitrary camera trajectories. Our edge depth calculation is based on an analysis of color and score maps for each pixel. These maps are built in depth-view space (see Fig. 3), that enables us to easily visualize the appearance of the projected point in each secondary image with a certain depth. They help us define a depth score that distinguishes the correct depth more robustly (see the score profiles in Fig. 3).

Let us consider the i^{th} image $I_i^v \in \boldsymbol{I}^v$ as a primary image. We calculate scores for 1024 depth hypotheses[2] by selecting its 100 neighborhoods

$$\boldsymbol{N}(i, \boldsymbol{I}^v) = \{I_{i-50}^v, \ldots, I_{i+50}^v\} \backslash \{I_i^v\} \tag{1}$$

as the secondary images. Considering a pixel $\boldsymbol{p}$ in I_i^v, let $\boldsymbol{q}$ be its projection in the j^{th} secondary image N_j using the k^{th}

[1] Throughout, we use normal capitals for images and bold for image sets.
[2] See Sec. V-B for our parameter settings.

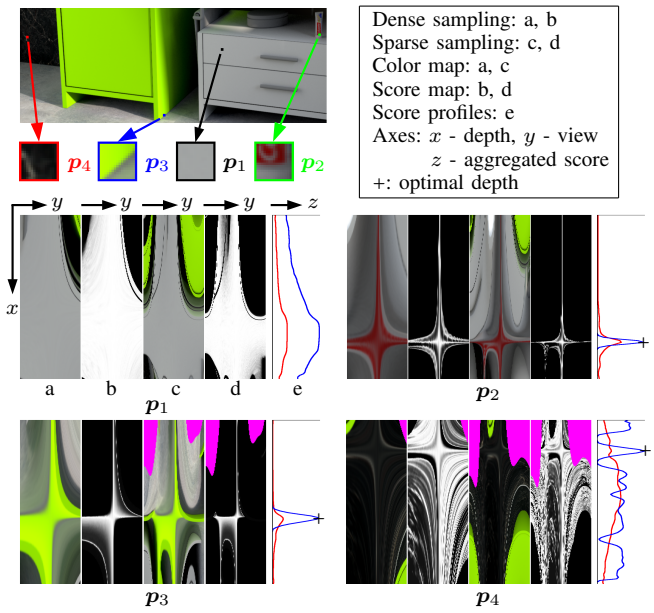


Figure 3. Depth-view-space analysis. For each marked pixel (top left, the 15×15 pixel patch is enlarged), we present its color and score maps once with densely sampled and once with sparsely distributed secondary images (see Sec. III-C for image sampling), as well as the profiles of per-depth aggregated scores by averaging (red) and our robust method (blue, Sec. III-B) on the *sparse* image samples. In cases of out-of-image projection, the corresponding areas are marked magenta. The maps are widened for better visualization.

depth d_k. To ensure that all projections have constant density along the epipolar line, the hypotheses are distributed evenly in inverse depth along the viewing ray. Using each pair of (d_k, N_j) on $\boldsymbol{p}$, we build a 1024×100 size color map with the projection color $N_j(\boldsymbol{q})$ in *depth-view space*, *i.e.*, the vertical axis represents depth values and the horizontal axis secondary views, as well as a corresponding score map with the depth score calculated using the Gaussian kernel[2]:

$$\mathrm{S}(\boldsymbol{p}, I_i^v, d_k, N_j) = \exp\left(-\frac{1}{2{\sigma_c}^2}|I_i^v(\boldsymbol{p}) - N_j(\boldsymbol{q})|^2\right). \quad (2)$$

Fig. 3 shows the generated maps for four representative pixels. Overall, there should exist a cross shape in each map: Significantly more same colors and score peaks are distributed around the center view and a certain hypothesis which is typically the real depth. The clarity of this property depends on image contrast, the image sampling scheme, and visibilities in secondary views. Pixel $\boldsymbol{p}_1$ gains a wide score peak distribution due to weak color contrast. Higher contrast (*e.g.*, $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$) or sparser image sampling (*i.e.*, the image set covers a wider range of viewing directions) results in a more recognizable distribution. The cross shape is interrupted in view direction where occlusion ($\boldsymbol{p}_3$) or out-of-image projection ($\boldsymbol{p}_4$) occurs. These desirable features are well-incorporated in our edge depth calculation.

B. Robust Score Aggregation

Edge depth is determined by aggregating for a certain hypothesis the per-secondary-image depth scores (*i.e.*, the scores in one row of the proposed score map) and finding the depth receiving the highest sum. As Fig. 3 shows, simple averaging leads to score profiles with ambiguous peaks for the pixels that are occluded (*e.g.*, $\boldsymbol{p}_3$) or out-of-image ($\boldsymbol{p}_4$) in some views. Kim *et al.* [2] handle this problem with mean-shift iterations but at the expense of computation time.

We define a robust and faster scheme by more carefully adding up the scores while moving from the nearest to outer views (preceding and following, respectively). This way, we can more reliably distinguish different cases (*e.g.*, occlusion, out-of-boundary projection, or the same surface point) and accordingly treat the remaining projections. This aggregation process equates to accumulating the scores from the center to left and right within a line of the pixel's score map.

Specifically, let the 100 depth scores of pixel $\boldsymbol{p}$ for hypothesis d_k be aggregated to $\mathrm{S}^\Sigma \leftarrow 0$, $n_m \leftarrow 0$ be the number of measurable images, and $n_h \leftarrow 0$ the number of out-of-image projections. When we *sequentially* test d_k in the preceding secondary images $\{I_{i-1}^v, \ldots, I_{i-50}^v\}$, three cases are motivated in the j^{th} image N_j:

1) If the projection moves out of the image, we hallucinate the minimum score $\mathrm{S}_{\min}$ found so far, which is initialized with $\mathrm{S}_{\min} \leftarrow 1$, for the remainder by

$$\mathrm{S}^\Sigma = \mathrm{S}^\Sigma + (50-j)\,\mathrm{S}_{\min} \quad \text{and} \quad n_h = n_h + (50-j). \quad (3)$$

This produces a lower score average than as if the projections were available but still high enough to be reasonable for the same surface point.

2) If the projection is available and the depth score $\mathrm{S} := \mathrm{S}(\boldsymbol{p}, I_i^v, d_k, \mathrm{N}_j)$ is acceptable[2], we update $\mathrm{S}_{\min}$ by

$$\mathrm{S}_{\min} = \min((1+\alpha)\,\mathrm{S}_{\min}, \mathrm{S}) \quad \text{with} \quad \alpha < 1. \quad (4)$$

We use $(1+\alpha)\,\mathrm{S}_{\min}$ to approximate a relatively bad match by allowing a score decrease of $\alpha\,\mathrm{S}_{\min}$ along a noisy score profile[2]. Then the score is aggregated by

$$\mathrm{S}^\Sigma = \mathrm{S}^\Sigma + \mathrm{S}_{\min} \quad \text{and} \quad n_m = n_m + 1. \quad (5)$$

We add up $\mathrm{S}_{\min}$ instead of S to avoid adding higher scores for the remaining images if the actual score is low. Because this low score might be introduced by a depth discontinuity and the remaining projections might be on a foreground surface with the same color.

3) If the depth score is unacceptable, it represents a discontinuity. We stop aggregating because the surfaces of the remaining projections can be arbitrary.

The aggregation is then performed on the following secondary images $\{I_{i+1}^v, \ldots, I_{i+50}^v\}$ in the same way. We let $\mathrm{S}^\Sigma = 0$ if $n_m \le 2$, and normalize it otherwise by

$$\mathrm{S}^\Sigma = f\,\mathrm{S}^\Sigma / (n_m + n_h), \quad (6)$$

where $f = \sqrt{n_m}$ is a correction factor to prefer the hypotheses generating more promising matches. Fig. 3 shows that our score aggregation creates more distinctive score profiles for pixel $\boldsymbol{p}_3$ and even more correct for $\boldsymbol{p}_4$ than averaging.

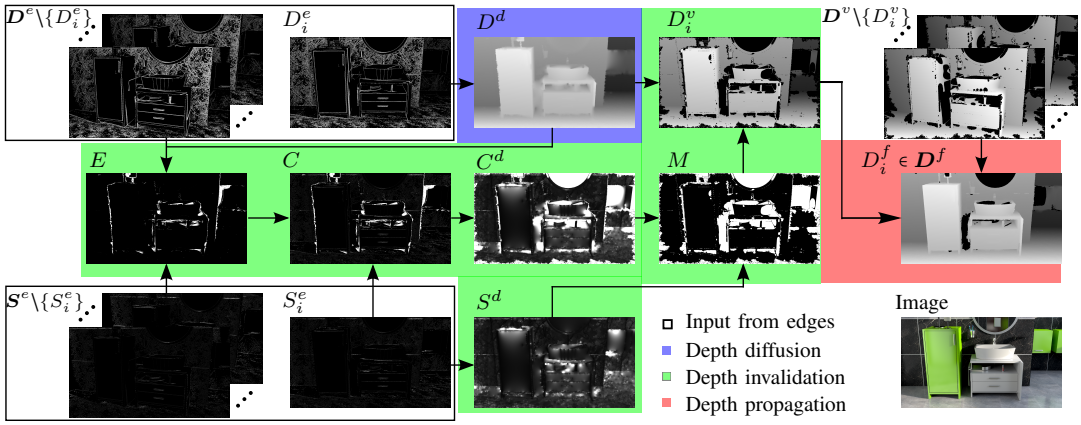


Figure 4. Homogeneous area depth estimation for the i^{th} image (bottom right). With all edge depth maps $\boldsymbol{D}^e$ and the corresponding SD maps $\boldsymbol{S}^e$, we produce the dense and consistent depth map $D_i^f \in \boldsymbol{D}^f$ via three stages: diffusion, invalidation, and propagation. See Sec. IV for details. The values in the SD maps and the error maps (E, C, and C^d) have the same scale. To visualize the errors, the SD maps are very dark.

C. Two-Scale Image Sampling

A dense set of neighboring images has less chances of a surface point being occluded or projected outside the view (*e.g.*, $\boldsymbol{p}_3$ and $\boldsymbol{p}_4$ in Fig. 3) but might provide narrow baselines leading to imprecise depth estimate. We increase triangulation angles by introducing a second set that samples the frames more sparsely and more widely-spread. Hereby we can refine the depth range at the dense scale and determine depth more reliably at the sparse scale (see Sec. III-D).

To this end, we sample the video frames into a dense subset $\boldsymbol{I}^d$, and then further sample $\boldsymbol{I}^d$ into a sparse subset $\boldsymbol{I}^s$ with a four times wider spacing (*i.e.*, $I_i^s \in \boldsymbol{I}^s$ corresponds to $I_{4i}^d \in \boldsymbol{I}^d$). Our depth estimation is performed on $\boldsymbol{I}^s$. Our goal is to guarantee that the sparse secondary image sampling of each primary image features approximately the same change in view direction of about 90° in total. This means, each pair of neighboring dense samples should have a viewing angle no smaller than $\frac{90^\circ}{4\times100}$. This is achieved using the visibility constraints of the SfM points from [4]. In the following, we introduce how we obtain subpixel depth precision via iterative depth range refinement on both image subsets.

D. Subpixel-Level Depth Sweeping

We get subpixel precision by iteratively testing 1024 depth hypotheses within the gradually refined depth range based on the score profile (visualized in Fig. 3), until convergence or bad matching is reached. This process is done per pixel.

For an edge pixel in the i^{th} sparse image sample, we initialize its depth range using the depth values of the nearest and farthest visible SfM points in that view. We first refine the depth range *in one pass* by referring to the *dense* images $\boldsymbol{N}(4i, \boldsymbol{I}^d)$ and then *iteratively* refine it by using the *sparse* samples $\boldsymbol{N}(i, \boldsymbol{I}^s)$ (see (1)). Particularly, in the t^{th} iteration, we first determine the optimal depth $\hat{d}$ within the current depth range. If $\hat{d}$ generates only a few measurable matches, *i.e.*, $n_m < 10$ in (6), we stop sweeping and leave the pixel assigned no depth. Otherwise, let the corresponding aggregated score of $\hat{d}$ be S_t^Σ. If $|\,\mathrm{S}_t^\Sigma - \mathrm{S}_{t-1}^\Sigma\,| < 0.001$, the sweeping is converged and this pixel is assigned $\hat{d}$. Otherwise, we update the depth range with the minimum and maximum of all the hypotheses whose aggregated score is high enough, *i.e.*, higher than $0.9\,\mathrm{S}_t^\Sigma$.

Depth refinement. The pixel-level operation probably results in outliers due to insufficient contrast, aliasing artifact, or reflection. To detect the unreliable depth, we shift the pixel projections in all sparse-scale secondary images along the epipolar lines by one pixel and calculate a new aggregated score. If this value is higher or close to the score of the depth estimate, the optimal depth is ambiguous and thus removed. We use 0.02 as the threshold. We also eliminate the isolated points with large 3D distance from neighborhoods and then perform a 7×7 median filter for smoothness.

E. Depth Standard Deviation

We calculate the standard deviation, SD, of an edge depth to measure its imprecision, which will be reused for adjusting tolerances when depth inconsistencies are searched in homogeneous areas (Sec. IV-B). The score profile of a confident depth should have a sharp peak (*e.g.*, pixel $\boldsymbol{p}_2$ in Fig. 3). Thus, we define the SD of the depth estimate d as

$$\sigma_{sd} = \sqrt{\sum_k (d - d_k)^2\,\mathrm{S}_k^\Sigma / 1024}. \tag{7}$$

S_k^Σ is the aggregated score of the k^{th} depth d_k after the first depth range refinement pass. Each SD map is also smoothed by a 7×7 median filter.

IV. Homogeneous Area Depth Estimation

In many scenes, object edges already capture the main characteristic structures of the scene. Thus we propose to handle remaining homogeneous areas by diffusing those sparse depth estimates, assuming flat surfaces in between. In addition to completeness, we also favor visibility consistency of the interpolated surfaces.

For this purpose we adopt the workflow depicted in Fig. 4, that begins by diffusing each edge depth map $D_i^e \in \boldsymbol{D}^e$ to a complete depth map D^d (Sec. IV-A). In Sec. IV-B the wrong interpolants occluding the real surfaces are invalidated from D^d to create a corresponding valid depth map $D_i^v \in \boldsymbol{D}^v$. In Sec. IV-C, we improve the completeness and consistency of

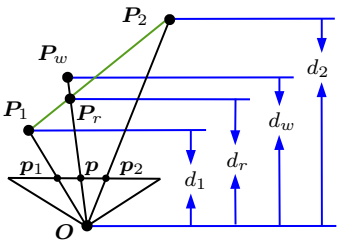


Figure 5. Depth diffusion from pixels $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ to the mid-pixel $\boldsymbol{p}$. Point $\boldsymbol{P}_w$ is the projected interpolant using the classic isotropic method, which defines the diffused depth as $d_w = \frac{1}{2}(d_1 + d_2)$. $\boldsymbol{P}_r$ is the assumed real point on the principle surface (green), and d_r is the corresponding real depth. $\boldsymbol{O}$ is the camera center.

the final depth map $D_i^f \in \boldsymbol{D}^f$ by propagating the front-most valid depth values over views.

A. Depth Diffusion

This section first investigates the problem of the classic diffusion scheme on depth data [16]. Next, we propose a modified method to correct surfaces perspectively.

1) Classic diffusion: The classic approach performs iterative convolution with a 4-point stencil. Let the diffused result of edge depth map D_i^e in the t^{th} iteration be D_t^d, and $D_0^d \leftarrow D_i^e$ for initialization. For a pixel $\boldsymbol{p} = (x, y)$, the diffusion problem is solved with the Jacobi iterations

$$D_{t+1}^d(\boldsymbol{p}) = \frac{1}{\delta^\Sigma} \sum_{(u,v)} \delta(D_t^d(u,v)) D_t^d(u,v) \tag{8}$$

if $\boldsymbol{p}$ has no edge depth, and $D_{t+1}^d(\boldsymbol{p}) = D_t^d(\boldsymbol{p})$ otherwise to preserve the depth discontinuity. $(u,v) \in \{(x-w,0),(x+w,0),(0,y-w),(0,y+w)\}$ and w is the stencil size. The indicator function $\delta(\cdot)$ is evaluated to 1 for a known depth and 0 for others. $\delta^\Sigma = \sum_{(u,v)} \delta(D_t^d(u,v))$ is used for normalization. If no available depth is found for diffusion, *i.e.*, $\delta^\Sigma = 0$, the result at $\boldsymbol{p}$ remains unchanged.

Fig. 5 illustrates the horizontal diffusion for $\boldsymbol{p}$ from two pixels $\boldsymbol{p}_1 = (x-w, y)$ and $\boldsymbol{p}_2 = (x+w, y)$ with depth values d_1 and d_2, respectively. By using the isotropic diffusion in (8), $\boldsymbol{p}$ would be assigned the average depth $d_w = \frac{1}{2}(d_1 + d_2)$. However, due to the perspective distortion the corresponding 3D interpolant $\boldsymbol{P}_w$ lies behind the point $\boldsymbol{P}_r$, which is on a flat surface connecting the 3D points of $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, *i.e.*, $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$. This is because the 2D pixel grid is not exactly projected to an uniform grid on the slanted surface. A solution to this problem is adopting an anisotropic strategy.

2) Perspective diffusion: Instead of finding a weighting function, we employ a simpler method by extending (8) to the viewing ray space. We note in Fig. 5 that, since $\boldsymbol{P}_r$ is the intersection of the viewing ray $\overrightarrow{\boldsymbol{O}\boldsymbol{P}_r}$ and the 3D line $\overline{\boldsymbol{P}_1\boldsymbol{P}_2}$, both can be calculated, $\boldsymbol{p}$ can get its depth $\mathrm{G}(\boldsymbol{p}_1, \boldsymbol{p}_2, D_t^d) := d_r$ by back-projecting $\boldsymbol{P}_r$ onto the image plane.

Therefore, we first formulate this perspective strategy to allow for that the initial depths are scattered. Then we incorporate the interpolants respectively obtained in horizontal and vertical directions into the 2D diffusion task.

In order to assign $\boldsymbol{p}$ the only available depth if one of $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ has no estimate, we include the indicator function $\delta(\cdot)$ to denote the horizontally diffused depth as

$$\begin{aligned} d_x = {} & \delta_1\delta_2\,\mathrm{G}(\boldsymbol{p}_1, \boldsymbol{p}_2, D_t^d) \\ & + (1-\delta_1\delta_2)(\delta_1 D_t^d(\boldsymbol{p}_1) + \delta_2 D_t^d(\boldsymbol{p}_2)), \end{aligned} \tag{9}$$

where $\delta_1 = \delta(D_t^d(\boldsymbol{p}_1))$ and $\delta_2 = \delta(D_t^d(\boldsymbol{p}_2))$.

The vertically diffused value d_y is calculated similarly by using $\boldsymbol{p}_1 = (x, y-w)$ and $\boldsymbol{p}_2 = (x, y+w)$. We incorporate these two depths for each non-edge pixel $\boldsymbol{p}$ using

$$D_{t+1}^d(\boldsymbol{p}) = (\delta(d_x) d_x + \delta(d_y) d_y)/\delta^\Sigma \tag{10}$$

with $\delta^\Sigma = \delta(d_x) + \delta(d_y)$ for normalization.

Thereby, a small stencil size w might slow down the depth transportation. We use the stencil-shrinking solver [22] by initializing w for each interpolated pixel with the Euclidean distance from the nearest edge pixel.

Fig. 6 indicates that our perspective diffusion recovers flat surfaces while the classic method produces slight oscillations on large homogeneous surfaces. Videos for more convincing comparisons can be found on our project webpage.

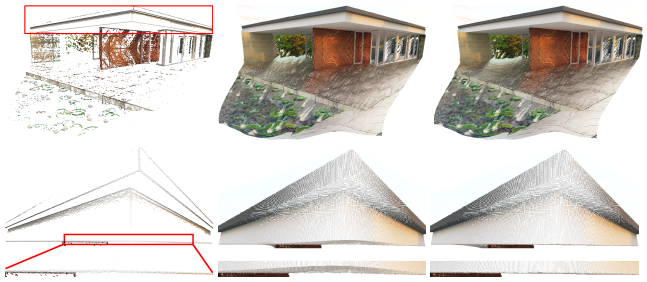


Figure 6. Points reconstructed by depth diffusion. The bottom of the roof is viewed parallel in the second row and the bounded region is enlarged. The classic method produces slight oscillations while ours obtains flat surfaces.

B. Depth Invalidation

Depth discontinuities with homogeneous areas in the background exhibit the *occlusion problem*, as illustrated in Figs. 2 and 7. Filling unreconstructed areas by diffusing the known edge depth (Sec. IV-A) leads to interpolating between fore- and background edges. This introduces incorrect interpolants which would occlude real surfaces. We solve these visibility conflicts in a sparse-to-dense manner (see Figs. 4 and 7). In each complete depth map, Sec. IV-B1 first presents how we distinguish a set of definitely invalid depth values. Then we make these sparse invalid pixels grow to larger areas as described in Sec. IV-B2.

1) Local depth invalidation: We find invalid interpolants in the diffused depth map D^d by checking for consistence with the edge depth calculated in other views. This process generates an error map E.

In particular, let us consider an interpolated pixel $\boldsymbol{p}$ in D^d as shown in Fig. 7. We back-project its corresponding point $\boldsymbol{P}_w$ to the edge depth map D_j^e of another view j. Let the pixel projection be $\boldsymbol{q}$ and the projected depth be d_j. If the depth estimate of $\boldsymbol{q}$ is available, it means that an edge ($\boldsymbol{P}_r$

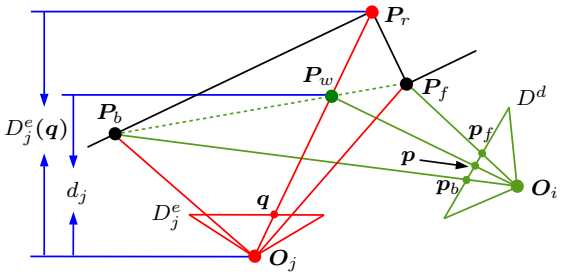


Figure 7. Occlusion problem and our solution. For the view at O_i, diffusing the depths of the foreground edge P_f and the background texture edge P_b produces the interpolants (dashed line) occluding the real surfaces. We use the detected edge P_r at pixel q in another view (at O_j) to invalidate the interpolant P_w and then grow the invalid area towards P_b and P_f.

in Fig. 7) is reconstructed at q. Since it is impossible to see an edge behind a surface in the same view, the considered depth $D^d(p)$ is theoretically invalid in case $d_j < D_j^e(q)$. For more robustness to the accuracy of D_j^e, we respect the depth imprecision in the corresponding edge SD map $S_j^e \in S^e$ (see Fig. 4) by comparing with a shortened depth:

$$d_j < D_j^e(q) - 3S_j^e(q). \tag{11}$$

If we find enough edges in other views that disagree with the interpolant, *i.e.*, (11) holds for $n_o \geq 3$ views, we define $D^d(p)$ as a bad depth and store the average depth difference:

$$E(p) = \frac{1}{n_o}\sum_j (D_j^e(q) - d_j). \tag{12}$$

2) Invalid area growing: Because the edge depths are sparse, the above process only removes a few isolated depth areas (see Fig. 4, E). These information has to be spread to all remaining pixels that most likely have bad interpolants as well (*e.g.*, in Fig. 7, p should grow towards pixels p_b and p_f to invalidate the whole surface between points P_b and P_f). Our thought is that, for each pixel with uncertain validity in the diffused depth map D^d, we can calculate two SD values, one *expected* only based on the measured S_i^e including SD values at edges, and the other *approximated* based on the calculated E including invalidation errors at a few interpolated pixels. If the latter SD value is larger, the interpolant is indicated invalid.

Hereby, as shown in Fig. 4, we first calculate the expected SD map S^d by diffusing S_i^e. On the other hand, a combined map C is calculated by summing up the values in S_i^e and E, which is subsequently also diffused to get the approximated SD map C^d. To maintain consistent data transportation between the depth diffusion (Sec. IV-A) and these two SD diffusions, we use the same number of iterations. If $C^d(p) > 1.2S^d(p)$, we label $D^d(p)$ incorrect by setting $M(p) = 0$, M being a validity mask initialized by $M(p) \leftarrow 1$. We use M to remove those invalid interpolants, producing the corresponding valid depth map $D_i^v \in D^v$.

Figs. 2 and 8 show the reconstructions before and after invalidation. A failure would occur when some low-contrast foreground edges fail to be reconstructed in one view (see Fig. 8). This case would generate sunken surfaces *behind* the real surfaces. These wrong areas cannot be invalidated because they are deeper than the corresponding edges correctly recovered in other views, *i.e.*, (11) fails. This problem, together with the holes reliably created for all occlusion problems, will be solved by the information from other views as introduced in Sec. IV-C.

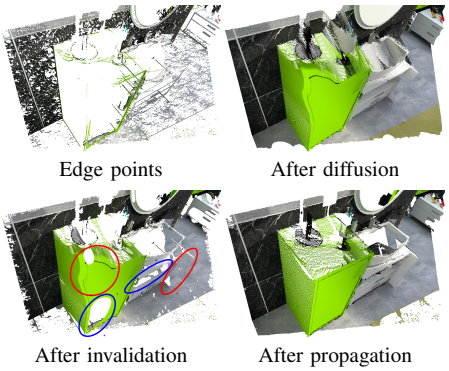


Figure 8. Points produced for one view. The depth propagation corrects the sunken surfaces (red) due to lost edge points (see the top row) and fills the holes (blue) arising from invalidation.

Figure 9. Image samples of the Pabellon, Boxes, and Building datasets.

C. Depth Propagation

Object edges probably have view-dependent image contrast (*i.e.*, thus an unreconstructed edge in one view might be recovered in another image) and different views reconstruct the depth discontinuities at various places, so both depth values and holes in the valid depth maps D^v are not consistent. Because all interpolants occluding the real surfaces have been invalidated and the final depth maps D^f should contain the front-most surfaces, we improve each depth map by taking the *closest* depth projected from other views. Trivial propagation might also project background surfaces into the holes. So we only fill holes with the data which agree with the hole boundary by thresholding the normalized color differences along the edge of the propagated area. Fig. 8 shows that the propagated data effectively overwrite the sunken surfaces and increase the completeness.

V. Experimental Results

A. Datasets and Compared Algorithms

We have tested our approach on four datasets with large homogeneous surfaces and arbitrary camera trajectories: *Bathroom* (Fig. 4), *Pabellon*, *Boxes*, and *Building* (Fig. 9). The first two are synthetic with ground truth, and the others real-world. Each type includes both indoor and outdoor scenes. All images have resolution of 1920×1080. We compare our results with other depth-map-based methods: [20] (*BAI*), [19] (*WEI*), [2] (*KIM*), and [5] (*ENG*). We also evaluate the depth propagation (Sec. IV-C) by testing our

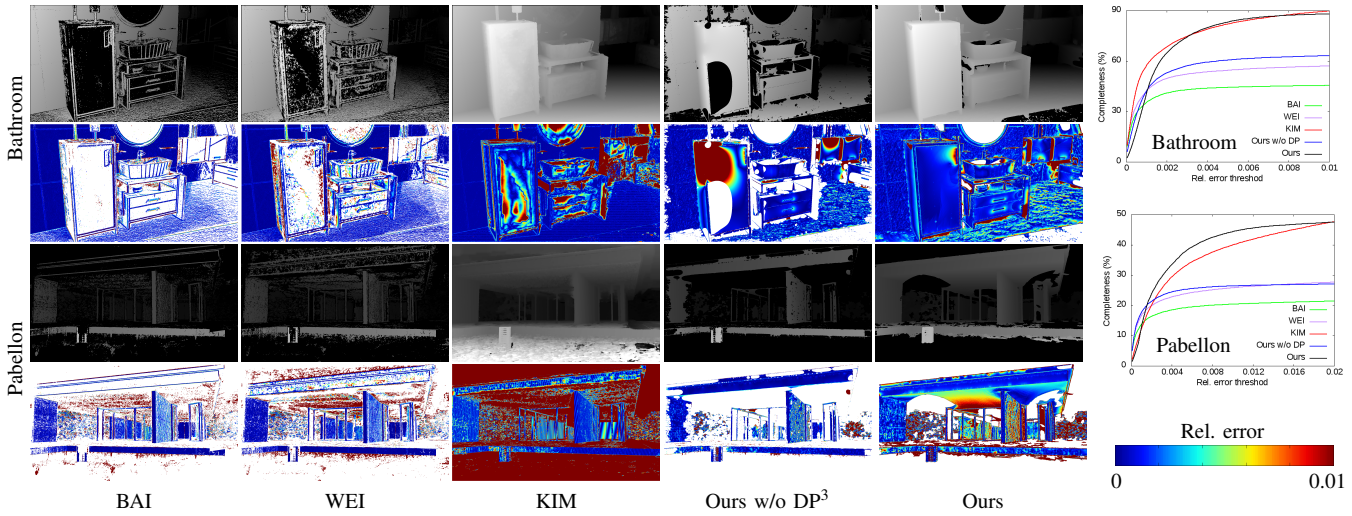


Figure 10. Comparisons against the ground truth. For each scene, we present the depth maps (the first row), the relative error maps (the second row, the white pixels having no estimate or no ground truth data), and the curves (right) measuring how much of the estimated depth has a smaller relative error than a given threshold. Our approach only recovers the reliable surfaces, and invalidates the noisy or inconsistent reconstructions (see Fig. 11).

pipeline without this step.[3] Our experiments were run on NVIDIA GeForce GTX 680 (KIM) and Titan (others) GPUs.

The classic MVS algorithms BAI and WEI use PatchMatch on a few secondary images, region growing, and post-checking to remove inconsistent outliers. WEI incorporates cross-view depth filtering for high consistency.

KIM performs ray-oriented depth sweeping at edges like ours, but only probing 256 hypotheses in single pass. The homogeneous surfaces are recovered by detecting edges in gradually downscaled images. We used the new results with 1024 depth values from the authors for fair evaluation.

The recent SLAM system ENG only produces semi-dense depth maps and its depth quality relies largely on the calculated camera poses. We obtained bad camera tracking on the Pabellon and Boxes scenes using this method. Because the dimensions of the images input to its source code must be multiples of 16, we cropped the images into 1920×1072 and thus cannot compare its results to the ground truth.

B. Parameter Settings

Since we sparsely sample the secondary images for about 90° movement around the scene, experiments demonstrated that using 100 secondary images together with two-scale image sampling provides just enough angular resolution on the surface points for reconstruction.

256 depth values are sufficient for visualizing depth maps as in [2]. But we need more robust estimates for surface invalidation. We use 1024 hypotheses to avoid skipping the correct estimates in the depth-sweeping process. We found that this setting best compromises on quality and speed.

In (2), we set $\sigma_c = 8$ for synthetic scenes and 15 for real-world, *i.e.*, a larger value leads to better noise-robustness. In case 2) of our score aggregation, we utilized a stricter score threshold s_1 for dense image samples to greatly refine the initial depth range and a looser threshold s_2 at sparse scale. Specifically, we used $s_1 = 0.8$ and $s_2 = 0.4$ for synthetic scenes but smaller values $s_1 = 0.4$ and $s_2 = 0.2$ for real-world to have higher noise-tolerance. In (4), we set $\alpha = 0.05$. With more noisy score profiles due to image noise, increasing it can improve the depth completeness.

[3] In this section, we use w/o for without and DP for depth propagation.

Measurement (Bathroom)	BAI	WEI	KIM	Ours w/o DP[3]	Ours
Completeness (%) ↑	47.499	60.368	100.000	70.221	92.503
Mean Rel. Error ($\times 10^{-3}$) ↓	4.467	4.150	9.867	3.695	3.275
Measurement (Pabellon)	**BAI**	**WEI**	**KIM**	**Ours w/o DP**	**Ours**
Completeness (%) ↑	26.886	35.697	100.000	28.221	56.255
Mean Rel. Error ($\times 10^{-3}$ ↓	69.616	91.890	216.955	7.248	34.065

Table I
STATISTICAL COMPARISONS OF THE RESULTS PRESENTED IN FIG. 10. THE ARROWS INDICATES PREFERRED DIRECTIONS.

C. Evaluation

Synthetic scenes. Fig. 10 presents the depth map comparisons using the ground truth, and the completeness curves for varying relative error thresholds. Table I lists the quantitative evaluation in terms of completeness and mean relative error. It can be seen that, BAI and WEI both eliminate most of homogeneous surfaces by removing all inconsistent outliers, although WEI improves the depth consistency. KIM reconstructs complete scenes but generating discrete artifacts and wrong geometries in textureless areas. In contrary, we only aim to reconstruct the data which is captured and do not guess where the surfaces are most likely. The depth propagation introduces both correct (*e.g.*, the refrigerator corner in Bathroom) and wrong (*e.g.*, the roof in Pabellon) surfaces, but it significantly improves the completeness and we still yield the highest accuracy (see Table I). Fig. 11 shows how bad the KIM's points are which we do not recover, and that ENG only reconstructs noisy edges.

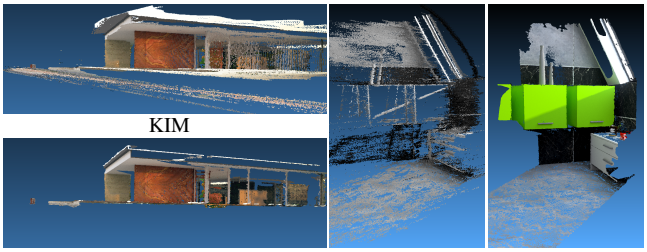


Figure 11. Comparisons of the reconstructed points on synthetic scenes. ENG's points exhibit no color due to grayscale input of the source code.

Method	Bathroom	Pabellon	Boxes	Building
BAI	68.12	63.28	71.29	75.92
WEI	34.02	35.14	32.84	35.30
KIM (256)	33.72	33.27	31.74	35.77
KIM (1024)	121.55	117.12	119.76	125.39
Ours	10.21	12.53	11.67	11.07

Table II RUNTIME (SEC.) FOR CALCULATING ONE DEPTH MAP. THE TIME OF KIM TESTING 256 AND 1024 DEPTHS ARE BOTH LISTED.

Real-world scenes. We compare the real-world reconstructions in Fig. 1. Although the captured images suffer from noise and compression artifact, we can still recover dense and smooth surfaces without generating wrong interpolants (*e.g.*, the surface of the table recovered by KIM occludes the bottom of the background boxes) or noisy points (*e.g.*, due to the reflection on the windows of the building).

Runtime. Table II provides the GPU runtimes for reconstructing one view. We achieve the fastest reconstruction even compared with the KIM testing 256 depth hypotheses. The reason is that, we only sweep depth values on edges and our robust score calculation avoids the mean-shift iterations employed by KIM, although we iteratively test 1024 depths. Moreover, our diffusion-based technique is more parallelizable than the region growing strategy of BAI and WEI.

VI. CONCLUSION

We proposed an efficient approach for dense recovery of textureless surfaces from videos. We have discovered that the depth estimates at object edges are sufficient to infer dense reasonable surfaces for the enclosed areas which are assumed to be smooth. Our edge depth calculation is investigated in a novel depth-view space and the incorporated techniques are robust to unstructured camera trajectories. Homogeneous surfaces are reconstructed by diffusing the scattered depth while occlusions are solved effectively. The results demonstrate the superior performance of our algorithm in terms of completeness, accuracy, and consistency. Our methods for edge or homogeneous area depth estimation can be plugged into other reconstruction techniques.

REFERENCES

[1] Middlebury Multi-View Stereo Evaluation. http://vision.middlebury.edu/mview/eval/.

[2] C. Kim, H. Zimmer, Y. Pritch, A. Sorkine-Hornung, and M. Gross. Scene reconstruction from high spatio-angular resolution light fields. *ACM Trans. Graph.*, 32:73:1-12, 2013.

[3] A. Kundu, Y. Li, F. Dellaert, F. Li, and J. M. Rehg. Joint semantic segmentation and 3d reconstruction from monocular video. In *ECCV*, pp.703-718, 2014.

[4] B. Resch, H. P. A. Lensch, O. Wang, M. Pollefeys, and A. Sorkine-Hornung. Scalable structure from motion for densely sampled videos. In *CVPR*, pp.3936-3944, 2015.

[5] J. Engel, T. Schops, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *ECCV*, 2014.

[6] K. Kolev, P. Tanskanen, P. Speciale, and M. Pollefeys. Turning mobile phones into 3D scanners. In *CVPR*, 2014.

[7] Z. Kang and G. Medioni. Progressive 3D model acquisition with a commodity hand-held camera. In *WACV*, 2015.

[8] M. Pizzoli, C. Forster, and D. Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time. In *ICRA*, pp.2609-2616, 2014.

[9] R. A. Newcombe and A. J. Davison. Live Dense reconstruction with a single moving camera. In *CVPR*, 2010.

[10] D. Chekhlov, A. P. Gee, A. Calway, and W. Mayol-Cuevas. Ninja on a Plane: Automatic discovery of physical planes for augmented reality using visual SLAM. In *ISMAR*, 2007.

[11] A. P. Gee, D. Chekhlov, W. Mayol-Cuevas, and A. Calway. Discovering planes and collapsing the state space in visual SLAM. In *BMVC*, 2007.

[12] J. Martinez-Carranza and A. Calway. Unifying planar and point mapping in monocular SLAM. In *BMVC*, 2010.

[13] G. Zhang, J. Jia, T. T. Wong, and H. Bao. Consistent depth maps recovery from a video sequence. *IEEE TPAMI*, 31(6):974-988, 2009.

[14] F. Yu and D. Gallup. 3d reconstruction from accidental motion. In *CVPR*, pp.3986-3993, 2014.

[15] Z. Kang and G. Medioni. Fast dense 3D reconstruction using an adaptive multiscale discrete-continuous variational method. In *WACV*, pp.53-60, 2014.

[16] N. Stefanoski, C. Bal, M. Lang, O. Wang, and A. Smolic. Depth estimation and depth enhancement by diffusion of depth features. In *ICIP*, pp.1247-1251, 2013.

[17] R. Rzeszutek and D. Androutsos. Efficient automatic depth estimation for video. In *DSP*, pp.1-6, 2013.

[18] R. Rzeszutek and D. Androutsos. A Framework for estimating relative depth in video. *CVIU*, 133:15-29, 2015.

[19] J. Wei, B. Resch, and H. P. A. Lensch. Multi-view depth map estimation With cross-view consistency. In *BMVC*, 2014.

[20] C. Bailer, M. Finckh, and H. P. A. Lensch. Scale robust multi view stereo. In *ECCV*, 2012.

[21] C. Hoppe, M. Klopschitz, M. Donoser, and H. Bischof. Incremental Surface Extraction from Sparse Structure-from-Motion Point Clouds. In *BMVC*, 2013.

[22] S. Jeschke, D. Cline, and P. Wonka. A GPU Laplacian solver for diffusion curves and Poisson image editing. *ACM Trans. Graph.*, 28(5):116:1-116:8, 2009.