Densification of Semi-Dense Reconstructions for Novel View Generation of Live Scenes

Domagoj Baričević Tobias Höllerer Matthew Turk
University of California, Santa Barbara
{domagoj, holl, mturk}@cs.ucsb.edu

Abstract

In this paper, we consider the problem of rendering novel views of a live unprepared scene from video input, important to many application scenarios (such as telepresence and remote collaboration). We present an optimization approach to improving incomplete scene reconstructions captured in real time with a single moving monocular camera. We take semi-dense depth maps and convert them into a dense scene model, suitable for rendering plausible novel views of the scene using conventional image-based rendering. Our implementation densifies depth maps at the rate they are generated, and enables us to generate novel views of live scenes with no pre-capture or preprocessing. In evaluations comparing with other approaches, our method performs well even on difficult scenes, and results in higher-quality novel views.

1. Introduction

In recent years there has been significant progress in the field of monocular scene reconstruction, resulting in efficient real-time algorithms able to run on mobile hardware. However, most reconstruction algorithms are not able to produce fully dense accurate scene models. Instead, the current state-of-the-art approaches typically create semi-dense (or sparse) models. That is, they reconstruct the high-texture areas while leaving large gaps in low-texture regions. Some algorithms can produce dense models, but at the cost of efficiency and scalability, and often still fail at untextured areas. While semi-dense models are sufficient for some applications (*e.g.*, pose tracking), there are many use cases that require complete models (*e.g.*, novel view generation). Given the efficiency of semi-dense reconstruction, a question arises if it is possible to turn semi-dense models into dense reconstructions in a live, online fashion.

In this paper, we present a method for improving scene reconstructions captured in real time. Specifically, we propose an optimization approach to creating a dense scene model from a collection of semi-dense depth maps. We assume a scenario with a moving monocular camera from which keyframes are periodically captured. In addition to the keyframe images, we assume that keyframes have known relative poses and semi-dense depth maps. This is consistent with what can be achieved using current state-of-the-art SLAM, with no pre-capture or preprocessing. From the output of such a system, we generate a dense scene model suitable for creating plausible novel views of the scene using conventional image-based rendering methods.

Our focus is on application scenarios that require the ability to generate a novel view of a *live* scene, where it is not desirable (or possible) to capture and process the scene beforehand. Some examples of these are telepresence [18], remote collaboration using Augmented Reality [31], and user-perspective views [3]. In these cases, the scene modelling needs to be efficient but still provide a good base for the image-based rendering. Addressing such scenarios are the motivation behind our densification method. We take semi-dense keyframes and densify them sufficiently fast to keep up with the rate that new keyframes are added, enabling real-time novel view generation.

2. Related Work

Densifying incomplete scene geometry into dense models has been previously demonstrated. Hawe *et al.* [15] showed that it is possible to reconstruct an accurate dense disparity map from only 5% of the disparity measurements, using compressive sensing. The approach depends on having accurate measurements for particular areas, but these areas are mostly near edges where reconstruction algorithms perform best. Shan *et al.* [29] used an energy minimization approach to densify individual semi-dense depth maps as part of their method for improving multi-view stereo reconstructions. These depth maps were then fed back to an approach that computed the scene model using a variation of PMVS [12] and Poisson surface reconstruction, an offline process not suitable for real-time applications.

Many real-time methods have been proposed that capture sparse or semi-dense scene geometry; primarily aimed

978-1-5090-4822-9/17 $31.00 © 2017 IEEE
DOI 10.1109/WACV.2017.99

at tracking, visual odometry, and SLAM. Engel *et al*. [10] proposed a visual odometry approach based on semi-dense stereo, which was later extended into a SLAM system (LSD-SLAM) [9]. LSD-SLAM models the scene as a collection of keyframes with semi-dense depth maps. These maps are fairly noisy, and the different keyframes do not necessarily fully agree on a consistent scene model (due to noise, error, and occlusion). However, the approach is very efficient and has been shown to work on a mobile phone.

Real-time dense reconstruction with a single moving camera was presented in [22]. The approach was based on using PTAM [20] for tracking and the initial scene model; the scene is then reconstructed by deforming a dense 3D mesh while satisfying image correspondences. In DTAM [24] the tracking uses whole image alignment, while the scene is modeled as per-keyframe depth maps computed by building and optimizing cost volumes. MonoFusion [26] is based on KinectFusion [19, 23] but works with standard color cameras, instead of a depth sensor. The depth data is fused into a volumetric model that defines the implicit surface of the scene. It uses stereo matching with the live frame and selected keyframes to compute the depth maps used in the fusion. As these method rely on satisfying photometric constraints, they can be sensitive to difficult image conditions. Also, these methods typically need to integrate information from every frame in the input video. By contrast, our approach is primarily focused on satisfying geometric constraints and works with sparse collections of frames.

There is a distinction between IBR methods that work directly and solely with the input images, and those that use some form of scene geometry. The work presented in this paper belongs to the latter category. Most IBR novel view generation methods rely on some form of geometric constraint: an explicit 3D model, depth maps, some kind of geometric proxy, image correspondences, etc. One of the earliest works in this area is by Chen and Williams [7], who introduced a view interpolation method that depends on precomputed image correspondences derived from range data. The Lumigraph by Gortler *et al*. [14] is another highly influential work. It used approximate geometry, a visual hull computed from the silhouettes of the object, with the silhouettes extracted using a blue-screen technique. Buehler *et al*. [4] extended this concept, generalized a number of IBR methods and defined a set of desirable properties for IBR. Among the key features of the approach is that the input images can be from arbitrary viewpoints.

Schirmacher *et al*. [27] presented a depth map based version of the lumigraph approach, and a system that used stereo with a camera array to render novel views of a dynamic scene. Hofsetz *et al*. [16] presented a IBR method that is based on pre-reconstructed depth maps, but also considers the uncertainty of the per-pixel depth estimates; the novel views are generated by splatting Gaussian kernels. Zitnick and Kang [34] developed a stereo algorithm intended specifically for IBR, noting that accurate depth maps are not necessary as long as the final image was plausible. Eisemann *et al*. [8] addressed the ghosting/blurring rendering artifacts in IBR caused by inaccurate geometry.

Recently, IBR methods have been proposed that rely on multi-view stereo [28] or structure-from-motion to build a model of the scene. However, MVS and SfM are computationally expensive so these IBR methods require substantial preprocessing, and are not suitable for live scenes. Goesele *et al*. [13] presented a method for generating transitions between images in a photo collection of a landmark (similar to [30]). Chaurasia *et al*. [6] presented a method that attempts to minimize distortions that can occur in interpolated images by considering object silhouettes at depth discontinuities. In follow-up work [5], instead of segmenting objects and warping the entire image, the inputs are oversegmented into superpixels and then the superpixels are locally warped.

3. Approach

Our goal is to densify semi-dense scene geometry and produce a dense scene reconstruction suitable for novel view generation using image-based rendering with a single monocular moving camera. From this camera, keyframes are captured periodically. Each keyframe has a known relative pose to other keyframes (at least to the preceding keyframe) and a semi-dense depth map with reasonably accurate depth information along the most prominent edges.

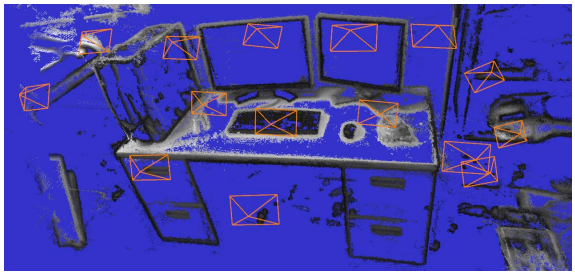

Figure 1. Visualization of input keyframes showing camera positions (orange frusta) and point cloud created from semi-dense depth maps. The blue regions are background (no captured points).

3.1. Optimization framework

At a high level, we treat the task of densifying the input semi-dense model as an optimization problem. We define an energy function whose minimum represents our desired solution, a dense model of the scene.

Our input consists of a set of keyframes, which we note as $\boldsymbol{K}$. Each keyframe $k \in \boldsymbol{K}$ is defined by an input image $\boldsymbol{i}_k$, an input semi-dense depth map $\boldsymbol{s}_k$, and a set of known poses $\boldsymbol{M}_{ki}$ of other keyframes $i \neq k$ relative to k.

Our objective is to compute a dense scene model. We define this model as a set of dense depth maps, with a dense

depth map computed for each input keyframe. We note this set as $\boldsymbol{D}$, and the dense depth map for keyframe k as $\boldsymbol{d}_k$. We formalize the desired properties of the final scene model with an energy function $F(\boldsymbol{D})$. This function is defined so that a ground truth model would be at or near the global minimum. It is a weighted sum of three basic terms: per-keyframe input deviation (E), per-keyframe smoothnesses (G_2), and mutual consistency between keyframes (R):

$$F(\boldsymbol{D}) = \alpha \cdot E(\boldsymbol{D}) + \beta_2 \cdot G_2(\boldsymbol{D}) + \gamma \cdot R(\boldsymbol{D}) \quad (1)$$

All three terms are functions of $\boldsymbol{D}$ (and only $\boldsymbol{D}$). For the purposes of minimizing F, only $\boldsymbol{D}$ and terms that are functions of $\boldsymbol{D}$ (or depth maps in $\boldsymbol{D}$) are variable; all other terms below are considered constant. The parameters α, β_2, and γ control the relative influence of each term. While these can be adjusted, they are constants for the optimization.

In the notation below, although images and depth maps are 2D arrays, they are noted as vectors. Their derivatives are computed with consideration of their 2D nature, but are also noted as vectors. We use "$\cdot$" to denote both scalar multiplication and the element-wise product of two vectors.

From the input images $\boldsymbol{i}_k$, we define $\boldsymbol{\sigma}_k$ and $\boldsymbol{\tau}_k$ – per-pixel weights computed from the partial derivatives of $\boldsymbol{i}_k$. From $\boldsymbol{s}_k$, we derive per-pixel weights $\boldsymbol{\epsilon}_k$ (see below). For completeness and generality, we also define per-keyframe weights w_k and pair-wise weights w_{ij} that control the influence of each keyframe or keyframe pair (currently just kept set to 1). All these weights assume values in the range $[0, 1]$.

3.1.1 Input deviation

The deviation term E represents the weighted sum of the input deviations from all the keyframes in the scene model. The deviation for each keyframe is the difference between the final scene model and the input depth data. The difference is a sum of weighted per-pixel differences, with the weight derived from the input depth data (*e.g.*, 1 if the pixel is in the semi-dense area, 0 otherwise). (We note the weighted sum as the L^2 norm of the element-wise product of the vector of weights and the vector of differences.)

$$E(\boldsymbol{D}) = \sum_{k \in \boldsymbol{K}} w_k \cdot \|\boldsymbol{\epsilon}_k \cdot (\boldsymbol{d}_k - \boldsymbol{s}_k)\| \quad (2)$$

This term is responsible for enforcing the known depth data. That is, it ensures that the final scene model remains mostly consistent with the input depths and does not drift too far from them. For the case of perfect input depths and ground truth final solution, this term would be exactly zero.

3.1.2 Smoothness

The smoothness term G_2 represents the desired property of biasing the scene model toward smooth surfaces, in order to suppress noisiness. Since most real-world scenes are piecewise smooth, this should provide a better overall reconstruction. As with the error term, this term is also the weighted sum of per-keyframe values. The per-keyframe smoothness is the sum of per-pixel values, computed from the partial derivatives of the depth map, with separate weights for the horizontal and vertical component. These weights are computed from the input keyframe image.

$$G_2(\boldsymbol{D}) = \sum_{k \in \boldsymbol{K}} w_k \cdot \left(\left\| \boldsymbol{\sigma}_k \cdot \partial_x^2 \boldsymbol{d}_k \right\| + \left\| \boldsymbol{\tau}_k \cdot \partial_y^2 \boldsymbol{d}_k \right\| \right) \quad (3)$$

$$\boldsymbol{\sigma}_k(x,y) = e^{-\frac{|\partial_x \boldsymbol{i}_k(x,y)|}{2}}, \boldsymbol{\tau}_k(x,y) = e^{-\frac{|\partial_y \boldsymbol{i}_k(x,y)|}{2}} \quad (4)$$

In general, the value of this term will not be exactly zero for a ground truth scene model, because a real scene can have non-smooth surfaces. Nevertheless, this term should be small for good reconstructions and large for bad ones.

3.1.3 Dense consistency

The final term R is the mutual consistency between the depth maps of the individual keyframes. Without this term, it is possible for each keyframe to have an optimal solution for its own depth map (satisfying the other two terms), while conflicting with the solution of another keyframe. This can happen because each keyframe has a different view of the overall scene, but these views cover overlapping areas. In any reasonable scene model, if two keyframes see the same point in space, they should agree on where that point is. Otherwise, the reconstruction is inconsistent and therefore incorrect. This term helps enforce the consistency of the reconstruction. Ideally, the value of this term should be zero, as there should be no inconsistencies.

To compute the consistency term R we consider the projections of each keyframe's depth map into all the other keyframes. That is, for a keyframe i there is a dense depth map $\boldsymbol{d}_i$ that defines that keyframe's model of the scene. This model can be projected into the point-of-view of another keyframe k, since the relative pose $\boldsymbol{M}_{ki}$ between the two keyframes is known from the view-graph. We note the general projection function as $\mathcal{P}(\boldsymbol{d}, \boldsymbol{M})$, and a specific instance as $\mathcal{P}_{ki}(\boldsymbol{d}_i) \equiv \mathcal{P}(\boldsymbol{d}_i, \boldsymbol{M}_{ki})$. The projected depth map is noted as $\boldsymbol{p}_{ki}$, and the area of keyframe k that keyframe i projects to is noted as $\mathit{\Pi}_{ki}$. If the two keyframes are mutually consistent, then the difference between the depth map of keyframe k and the projection $\boldsymbol{p}_{ki}$ within the area $\mathit{\Pi}_{ki}$ should be small.

The full consistency term is the weighted sum of all the pair-wise consistencies. Each pair-wise term is computed as the weighted per-pixel difference between the target keyframe's depth map $\boldsymbol{d}_k$ and the projected depth map $\boldsymbol{p}_{ki}$. The per-pixel weights $\boldsymbol{\pi}_{ki}$ are derived from the projection $\mathcal{P}_{ki}$ of keyframe i to keyframe k, (*e.g.*, 1 if the pixel is

in Π_{ki}, 0 otherwise). This is notably different from the per-pixel weights in the other terms; those are constant values derived from the input data. Here, the per-pixel weights are themselves functions of the scene model $\boldsymbol{D}$. However, they can be considered constant with respect to $\boldsymbol{d}_k$, as they are not dependent on it.

$$R(\boldsymbol{D}) = \sum_{\substack{i,j \in \boldsymbol{K} \\ i \neq j}} w_{ij} \cdot r_{ij}(\boldsymbol{d}_i, \boldsymbol{d}_j) \tag{5}$$

$$r_{ij}(\boldsymbol{d}_i, \boldsymbol{d}_j) = \left\| \boldsymbol{\pi}_{ij} \cdot \left(\boldsymbol{d}_i - \boldsymbol{p}_{ij} \right) \right\| \tag{6}$$

$$\{\boldsymbol{\pi}_{ij}, \boldsymbol{p}_{ij}\} = \mathcal{P}_{ij}(\boldsymbol{d}_j) \tag{7}$$

Ideally, if two keyframes are mutually consistent, then r_{ki} will be zero. In practice, due to occlusion issues and ambiguities with defining the projection function $\mathcal{P}$ (*e.g.*, how to handle depth discontinuities), this difference will be non-zero. However the value will be small for consistent keyframes and large for inconsistent ones. It should be noted that this difference is asymmetric; that is: $r_{ki} \neq r_{ik}$. This is primarily because there is no one-to-one mapping between Π_{ki} and Π_{ik}. Indeed it is possible for one area to be completely empty, while the other is completely full. This is because one keyframe may observe an occluder that hides parts of the scene that the other keyframe sees.

3.2. Additional terms

In principle, the above energy function is sufficient for finding a solution and creating a dense scene model. However, in practice some additional considerations need to be made in order to get good results with real world data. These include better handling of noisy input data and initializing the depth maps before the optimization.

3.2.1 Input consistency

The input deviation term described above is necessary to enforce that the densified model fits the input data. However, if the input semi-dense depth map is noisy it can make the densified model noisy as well. Furthermore, the dense consistency term will spread the noise between keyframes. Since real algorithms (*e.g.*, LSD-SLAM) are quite noisy, we need to address this issue.

Fortunately, this noise can be reduced by considering other keyframes' semi-dense input. Instead of just considering deviation from the keyframe's own input depth, we can consider the deviation from all the other keyframes' input as well. Each densified depth map should be constrained by all the input depth data, not just the data from one keyframe. This would provide more than one depth sample per pixel and allow for a better estimate of the true depth, reducing the impact of the noisiness in the input. Note that this is different from the dense consistency, as that constraint is only concerned with mutual consistency between dense depth maps and does not consider the input data at all.

We therefore define an input consistency term. It is similar in structure to the dense consistency term, but has a very different purpose and effect, and is meant to replace the input deviation term. As with dense consistency, we project depth data from one keyframe to another. However, we only project the semi-dense input depth map, not the current dense depth map. We define a projection function $\mathcal{S}(\boldsymbol{s}, \boldsymbol{M})$, with a specific instance noted as $\mathcal{S}_{ki}(\boldsymbol{s}_i) \equiv \mathcal{S}(\boldsymbol{s}_i, \boldsymbol{M}_{ki})$. The projected semi-dense depth map is noted as $\boldsymbol{s}_{ki}$, and the per-pixel weights as $\boldsymbol{\rho}_{ij}$. Note that unlike their dense consistency counterparts, both $\boldsymbol{s}_{ki}$ and $\boldsymbol{\rho}_{ij}$ are constant values, derived from the input data.

$$E_g(\boldsymbol{D}) = \sum_{i,j \in \boldsymbol{K}} w_{ij} \cdot \left\| \boldsymbol{\rho}_{ij} \cdot (\boldsymbol{d}_i - \boldsymbol{s}_{ij}) \right\| \tag{8}$$

$$\{\boldsymbol{\rho}_{ij}, \boldsymbol{s}_{ij}\} = \mathcal{S}_{ij}(\boldsymbol{s}_j) \tag{9}$$

3.2.2 Flatness

The smoothness term is the key to turning semi-dense inputs into dense depth maps. Its purpose is to fit a piecewise smooth surface to the semi-dense input data, and it is the term responsible for filling in the gaps of the depth map. As described above, it is based on the second-order derivative of the individual dense depth maps. Although the second-order is the appropriate choice when creating a global dense model consistent across multiple keyframes, it can cause issues when densifying a single keyframe (*e.g.*, for initialization). This is because the only constraints to the shape of the surface are the keyframe's semi-dense input depths and the smoothness requirement. There are hard constraints in the regions with depth data, but the surface has a lot of freedom regarding its shape in the gap regions (areas where there is no input depth data) as there is no constraint on the surface area or orientation. Since the smoothness term discourages discontinuities, it is possible to get a surface that is very smooth and closely conforms to the semi-dense input regions, but has abnormally large curved shapes in the gap regions. Since the overall optimization also includes the dense consistency term, this is not an issue when densifying multiple keyframes together into a consistent scene model, as the dense consistency creates constraints in the gap regions. However, the problem with abnormal surfaces happens when densifying a single keyframe, as there are no consistency constraints in that case.

Since it is sometimes necessary to densify a single keyframe, we need another constraint that will also fill the gaps but not have the same problem. In order to enable reasonable densification of individual keyframes, we define a "flatness" term, based on the first-order depth derivative. The use of a first-order derivative mitigates the problems

with the smoothness term. The first-order derivative encourages surfaces that are not just generally smooth, but also planar and facing the camera. This results in better initial depth maps, with the gap regions filled in with smaller and flatter surfaces.

$$G_1(\boldsymbol{D}) = \sum_{k \in \boldsymbol{K}} w_k \cdot (\|\boldsymbol{\sigma}_k \cdot \partial_x \boldsymbol{d}_k\| + \|\boldsymbol{\tau}_k \cdot \partial_y \boldsymbol{d}_k\|) \quad (10)$$

Note that while encouraging front-facing surfaces helps with densifiying a single depth map, it cannot be used for the mutually consistent global model, because each keyframe would pull the surface towards itself.

3.3. Energy minimization

The key to our approach is in defining energy terms whose minima correspond to the dense scene model that we want to obtain. The objective is to find the minimum of the energy function, and thus the dense scene model. These energy terms are over very high-dimensional spaces, and also feature some non-linearities. In order to find a solution efficiently we need to address these issues.

The first issue is with the dense consistency term. This term includes the use of projected depth maps $\boldsymbol{p}_{ij}$ produced by a projection function $\mathcal{P}$. This function depends on the current scene model, features non-linearities, and is best implemented as a rendering step. This makes it very difficult to use directly to solve for the scene model $\boldsymbol{D}$. However, for each keyframe k the projected depth maps $\boldsymbol{p}_{ki}$ are constant with respect to $\boldsymbol{d}_k$. Therefore each depth map $\boldsymbol{d}_k$ can be directly optimized separately. The overall scene model $\boldsymbol{D}$ can then be optimized iteratively.

Next, we note that each of the energy function terms described now fall into one of two categories: they either minimize the difference to some constant value, or they minimize a derivative defined as a simple linear equation over two or three neighbouring pixels. Therefore, we can redefine this problem as a large overdetermined system of linear equations for which we want a least-squares solution. Furthermore, this is a very sparse system. This type of system of equations can be very efficiently solved.

3.4. Densification process

Here, we outline the steps necessary to turn a collection of input semi-dense depth maps into a dense scene model.

The first step in the process of densifying a new keyframe is the initialization. This produces an initial dense depth map for the keyframe, created by only considering the data from the new input keyframe. The objective for the initialization is to get a dense depth map that contains the input semi-dense data, but fills in the missing information with a reasonable guess, by fitting a piecewise smooth surface. We accomplish this by defining a simple energy function over the keyframe's depth map, and finding its minimum. The energy function we optimize consists of only two terms: the semi-dense input deviation term and the flatness term.

The next step in the process is to generate the projected semi-dense depth maps needed by the input consistency term. For each source keyframe we create a semi-dense mesh from their input depth map and project it to the target keyframe. However, we do not simply project the semi-dense regions, as this can result in projecting geometry that would not be visible by the target keyframe because it is occluded by surfaces that should exist in the target's gap regions. Instead, we allow for the surfaces belonging to the gap regions to exclude the surfaces from the input regions that they would occlude.

Finally, we optimize for the main energy function. This function uses the projected semi-dense input consistency term, the smoothness term, and the dense consistency term:

$$F(\boldsymbol{D}) = \alpha_g \cdot E_g(\boldsymbol{D}) + \beta_2 \cdot G_2(\boldsymbol{D}) + \gamma \cdot R(\boldsymbol{D}) \quad (11)$$

The optimization is iterated. During an iteration, all the keyframe depth maps are considered static, while updated dense depth maps are being computed. The first step in each iteration is to generate the projected depth maps. Next, the individual depth maps are all optimized separately. At the end of each iteration, the keyframes' depth maps are updated with the newly optimized depth maps, and then the process is repeated. This keeps the process of optimizing a keyframe simple, while allowing all the depth maps to converge to a solution that is consistent between keyframes. In practice, just two iterations seem to be sufficient.

4. Implementation

We implemented our live system using a modular approach, separating it into three main components: the semi-dense source, the scene densifier, and the application. These parts run as separate processes communicating over a network connection. This design provides a great deal of flexibility in choice of input data or target application, and in how the processing is distributed. For example, the input data can be coming from a mobile device or aerial drone, the densification can be done on a server, while the target application could be running on an end-user's laptop.

The densifier implements the densification process and runs the energy minimization. We use the gradient descent method, which iteratively converges to the solution that minimizes the energy function. Given the sparsity and predictable pattern of the coefficients in the system, we can define an expression for the energy function's gradient that allows us to compute the partial derivative for a pixel. Each pixel is only dependent on itself and eight of its neighbours (two in each cardinal direction), and the expression easily scales with the number of keyframes as those constraints simply sum together. This optimization approach is inher-

ently parallel, and we exploit this by implementing a GPU-based solver, using OpenGL compute shaders. The full optimization problem scales quadratically with the number of keyframes, which is impractical. To maintain performance, the optimization needs to only consider a fixed number of keyframes at a time (*e.g.*, only use the 5 latest keyframes). The system can process a new keyframe in about one second, which is sufficient to keep up with the rate that keyframes are generated by the SLAM system, enabling real-time applications.

We use LSD-SLAM as the source of the input keyframes with semi-dense depth maps. Those are received by the densifier and the application, the densifier processes the keyframes and sends densified depth maps to the application. Our application takes the input keyframes and the dense depth maps and uses them to render the scene, using an approach similar to Unstructured Lumigraph Rendering [4]. Each keyframe is converted into a per-keyframe mesh. For a given desired camera view we select a number of keyframes with the closest matching views. We render their individual meshes, and blend them together to form the final image. LSD-SLAM's preferred resolution is 640×480, so we use the same for all depth maps and images.

5. Results

The densification method described above produces dense depth maps that are good approximations of the scene geometry. From these, we are able to render novel views of the scene that are generally very convincing. Figure 2 shows example results of our densification and novel view rendering for the Couch scene. We evaluate our method by comparing our results with those of representative methods from two other approaches.

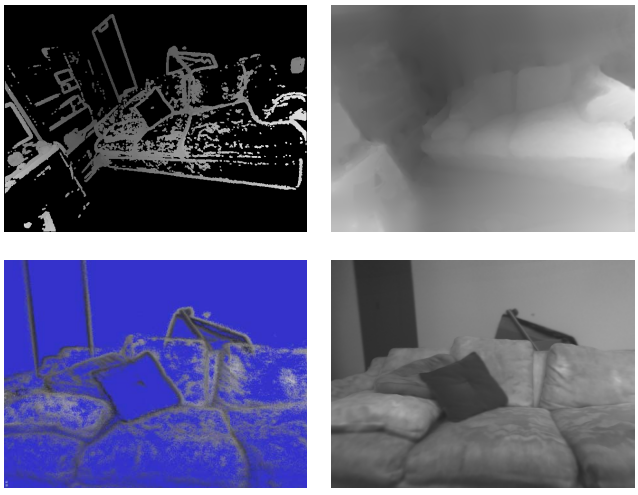

Figure 2. Results for Couch scene. Top row: input depth map (left), densified depth map (right). Bottom row: input point cloud (left), rendered novel view (right).

5.1. Evaluation

We run all methods on a dataset consisting of six different scenes (see left-most column in Figure 4) featuring different conditions. These include characteristics of indoor scenes, outdoor scenes, strong lighting, thin geometry, surface detail, textureless areas, and specular surfaces.

The first method for comparison is based on creating meshes out of point cloud data. The methods in this category (*e.g.*, [17], [21],[33]) take as input a (sparse) point cloud with camera poses from which the points have been observed. The first step is to perform a Delaunay tetrahedralization of the point cloud. After this, a two-label graph-cut is used to label each tetrahedron as either being free (empty space) or occupied (inside an solid object). A triangle that is a shared face between tetrahedra with different labels is considered to be a surface triangle. The collection of these triangles represent the surface mesh of the reconstructed scene. We implemented a version of this approach based on the real-time method of Hoppe *et al.* [17], and used this for our experiments. The output from LSD-SLAM was used as the source of the point cloud data.

The other method we compare to is based on dense real-time multi-view stereo. This approach takes as input a video sequence, and then performs multi-view stereo between frames (typically a reference frame and the current frame). For our experiments, we compare with REMODE by Pizzoli et al. [25], using the authors' implementation [1]. This code uses SVO [11, 2] (a visual odometry method from the same authors) for the pose estimation. However, we found SVO to be unreliable when run on our datasets (due to lack of texture), so we instead used LSD-SLAM to provide the camera poses. We did not modify REMODE to do this, since REMODE gets input from SVO via a node in the ROS framework. As LSD-SLAM also uses ROS, we simply modified LSD-SLAM to provide the same node in order to mimic SVO. This provided much more stable camera tracking and allowed REMODE to work on our dataset. Note that the semi-dense depth-maps from LSD-SLAM are not used by REMODE; only the camera poses are used.

Figure 3 shows a comparison of the reconstructions for the Couch and Desk scenes. For each scene we show: the ground truth reconstruction captured with a depth sensor, a mesh representing a dense depth map produced by our method, the mesh produced by Hoppe *et al.*, and a mesh representing a depth map produced by REMODE. As can be seen, our results are considerably better than the results of the other methods. We produce reasonably accurate smooth results, capture important detail, and handle untextured areas well. The tetraherdalization and mesh extraction approach results in coarse low-polygon reconstructions, and is sensitive to noise and outliers in the input point cloud. The dense multi-view stereo approach produces generally noisy results, and fails at low-texture areas.

In Figure 4 we show a comparison of novel views rendered using the reconstruction results of the different methods, against a ground truth reference. We use the same rendering approach for all methods, based on the unstructured lumigraph approach. When generating an image, we select eight keyframes as sources for the lumigraph rendering. Normally, we select the keyframes closest to the desired novel view. Here, in order to more faithfully represent the results obtained for novel views not in the camera path, we exclude frames that are in the immediate neighborhood of the reference frame (but still select eight frames total). As can be seen in Figure 4, the novel view renderings based on our method are generally of higher quality than the renderings based on the other methods.

We further compare the novel view rendering with a quantitative evaluation. For each of the scenes in our dataset we generate novel views from fifteen different camera poses and compute the Structural Similarity (SSIM) score [32] for each of the poses. Figure 5 shows the average SSIM scores across all the views per scene. Again, the results show our method outperforming the other two.

5.2. Limitations

Our approach has some limitations. It requires semi-dense inputs that are of reasonable quality. While it is capable of handling noise, large outliers can be problematic. If the input has extreme outliers that cannot be detected, then the densified models will suffer. We also assume that the inputs have fairly accurate pose estimates; large errors in the pose can results in distorted models. Our densified models can have inaccuracies in homogenous regions, due to the semi-dense input not providing any constraints in those areas. Fortunately, the homogeneity also means that the errors are generally not very objectionable when rendering, but they can become a greater issue when the homogenous area has a significant surface edge. Often, these regions correspond to planar areas, so it may be possible to remove these errors with a plane-fitting post-processing step.

6. Conclusion

We have presented an optimization approach to creating a dense scene model from a semi-dense reconstruction captured in real-time. Our approach is to densify the semi-dense depth maps produced by a state-of-the-art SLAM system, producing dense depth maps that are good approximations of the scene geometry. Our implementation can densify a semi-dense depth map fast enough to keep up with the source reconstruction, thus enabling real-time use. We use the produced dense scene model to create convincing novel views of the scene using image-based rendering. With this, we enable application scenarios that require the ability to show novel views of a live scene.

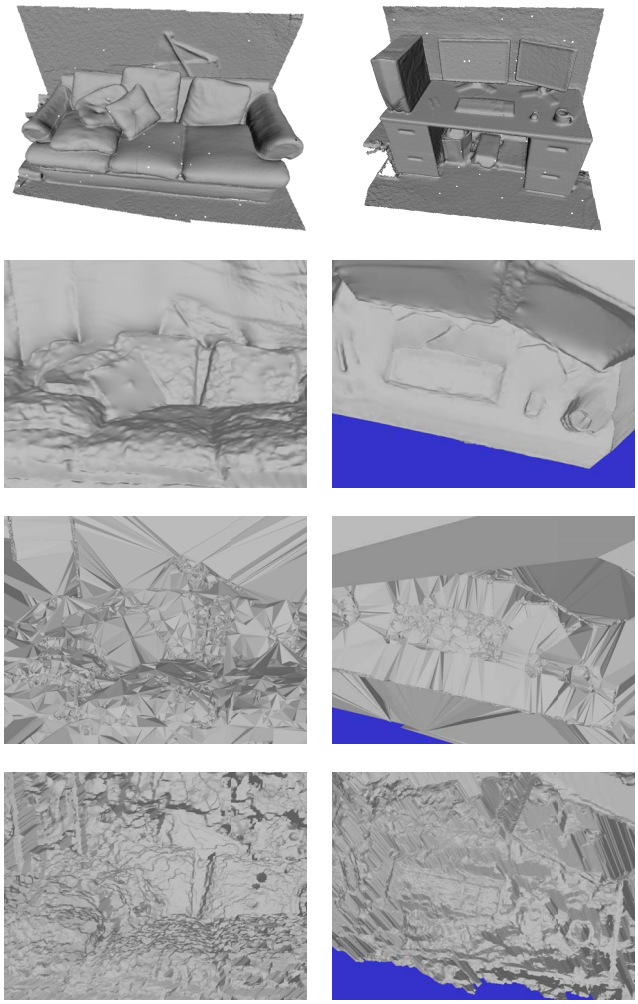

Figure 3. Comparison of reconstructions for Couch (left) and Desk (right) scenes. Top-to-bottom: ground truth, our method, Hoppe *et al.*, REMODE.

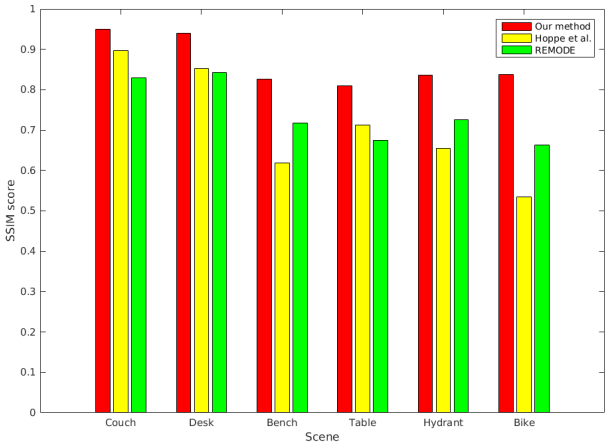

Figure 5. Average SSIM scores of novel views rendered using scene models produced by the different methods.

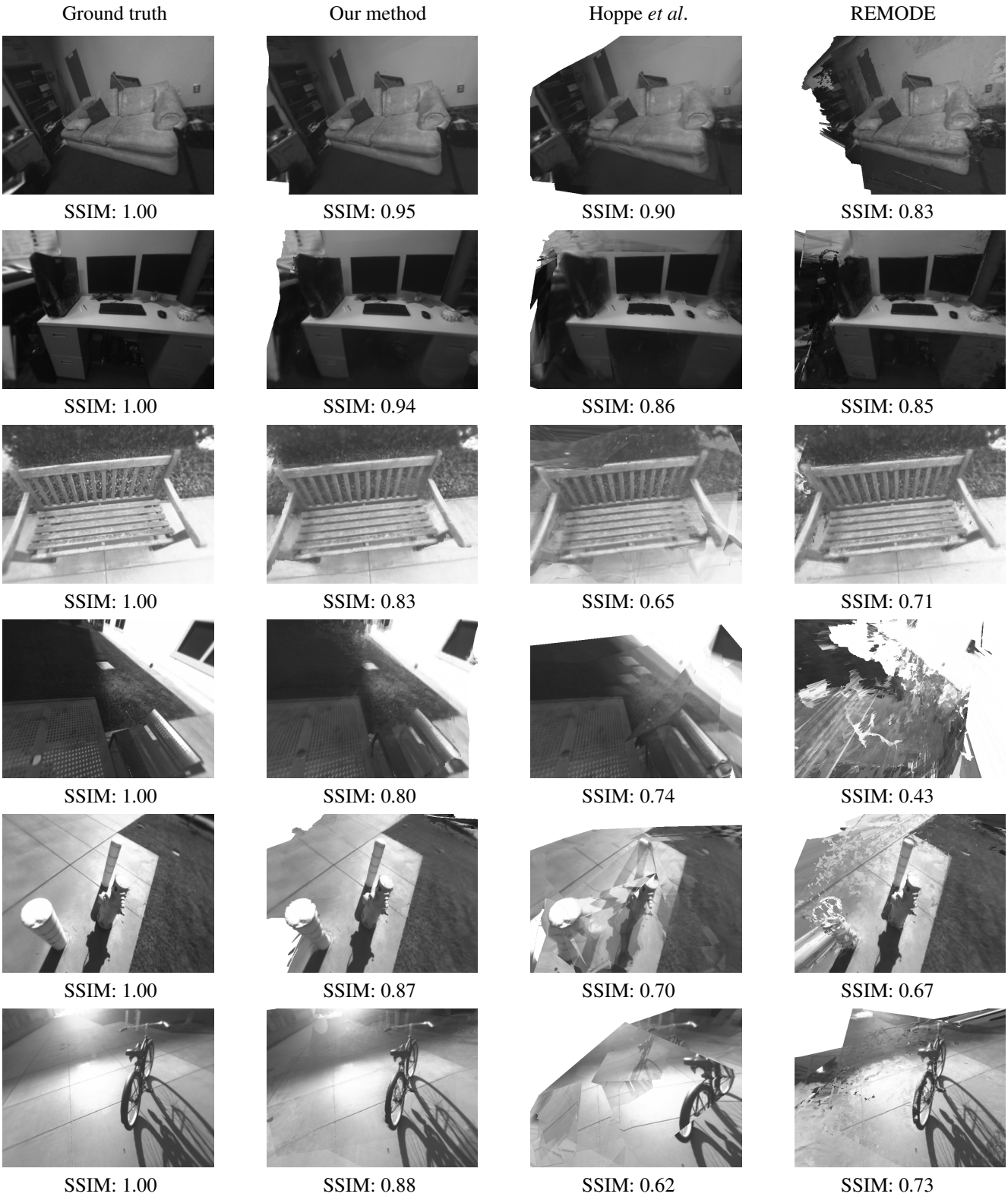

Figure 4. Comparison of rendered novel views using models from different methods (columns) for the six scene conditions (rows). Left-to-right: ground truth, our method, Hoppe *et al*., REMODE.

References

[1] https://github.com/uzh-rpg/rpg_open_remode. 6

[2] https://github.com/pizzoli/rpg_svo. 6

[3] D. Baričević, T. Höllerer, P. Sen, and M. Turk. User-perspective Augmented Reality Magic Lens from Gradients. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, VRST '14, pages 87–96, New York, NY, USA, 2014. ACM. 1

[4] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured Lumigraph Rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 425–432, New York, NY, USA, 2001. ACM. 2, 6

[5] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis. Depth Synthesis and Local Warps for Plausible Image-based Navigation. *ACM Trans. Graph.*, 32(3):30:1–30:12, July 2013. 2

[6] G. Chaurasia, O. Sorkine, and G. Drettakis. Silhouette-Aware Warping for Image-Based Rendering. *Comput. Graph. Forum*, 30(4):1223–1232, 2011. 2

[7] S. E. Chen and L. Williams. View Interpolation for Image Synthesis. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 279–288, New York, NY, USA, 1993. ACM. 2

[8] M. Eisemann, B. de Decker, M. A. Magnor, P. Bekaert, E. de Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating Textures. *Comput. Graph. Forum*, 27(2):409–418, 2008. 2

[9] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 8690 of *Lecture Notes in Computer Science*, pages 834–849. Springer, 2014. 2

[10] J. Engel, J. Sturm, and D. Cremers. Semi-dense Visual Odometry for a Monocular Camera. In *2013 IEEE International Conference on Computer Vision (ICCV)*, pages 1449–1456. IEEE, 2013. 2

[11] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, May 2014. 6

[12] Y. Furukawa and J. Ponce. Accurate, Dense, and Robust Multiview Stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1362–1376, 2010. 1

[13] M. Goesele, J. Ackermann, S. Fuhrmann, C. Haubold, R. Klowsky, D. Steedly, and R. Szeliski. Ambient Point Clouds for View Interpolation. *ACM Trans. Graph.*, 29(4):95:1–95:6, July 2010. 2

[14] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The Lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 43–54, New York, NY, USA, 1996. ACM. 2

[15] S. Hawe, M. Kleinsteuber, and K. Diepold. Dense disparity maps from sparse disparity measurements. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 2126–2133, Nov 2011. 1

[16] C. Hofsetz, K. C. Ng, G. Chen, P. McGuinness, N. Max, and Y. Liu. Image-Based Rendering of Range Data with Estimated Depth Uncertainty. *IEEE Computer Graphics and Applications*, 24(4):34–42, 2004. 2

[17] C. Hoppe, M. Klopschitz, M. Donoser, and H. Bischof. Incremental Surface Extraction from Sparse Structure-from-Motion Point Clouds. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2013. 6

[18] T. Ishikawa, K. Yamazawa, and N. Yokoya. Real-time generation of novel views of a dynamic scene using morphing and visual hull. In *IEEE International Conference on Image Processing 2005*, volume 1, pages I–1013–16, Sept 2005. 1

[19] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. A. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. J. Davison, and A. W. Fitzgibbon. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, pages 559–568. ACM, 2011. 2

[20] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, pages 1–10, Washington, DC, USA, 2007. IEEE Computer Society. 2

[21] P. Labatut, J. P. Pons, and R. Keriven. Efficient Multi-View Reconstruction of Large-Scale Scenes using Interest Points, Delaunay Triangulation and Graph Cuts. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Oct 2007. 6

[22] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1498–1505. IEEE Computer Society, 2010. 2

[23] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *2011 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136. IEEE Computer Society, 2011. 2

[24] R. A. Newcombe, S. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 2320–2327. IEEE, 2011. 2

[25] M. Pizzoli, C. Forster, and D. Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2609–2616, May 2014. 6

[26] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche. MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 83–88, Oct 2013. 2

[27] H. Schirmacher, M. Li, and H. Seidel. On-the-fly processing of generalized lumigraphs. *Comput. Graph. Forum*, 20(3):165–174, 2001. 2

[28] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 519–528, 2006. 2

[29] Q. Shan, B. Curless, Y. Furukawa, C. Hernandez, and S. Seitz. Occluding Contours for Multi-view Stereo. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4002–4009, June 2014. 1

[30] N. Snavely, S. M. Seitz, and R. Szeliski. Photo Tourism: Exploring Photo Collections in 3D. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 835–846, New York, NY, USA, 2006. ACM. 2

[31] M. Tait and M. Billinghurst. The effect of view independence in a collaborative ar system. *Comput. Supported Coop. Work*, 24(6):563–589, Dec. 2015. 1

[32] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004. 7

[33] S. Yu and M. Lhuillier. Incremental Reconstruction of Manifold Surface from Sparse Visual Mapping. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*, pages 293–300, Oct 2012. 6

[34] C. L. Zitnick and S. B. Kang. Stereo for Image-Based Rendering using Image Over-Segmentation. *International Journal of Computer Vision*, 75(1):49–65, 2007. 2