Efficient and Scalable Depthmap Fusion

Enliang Zheng
http://www.cs.unc.edu/~ezheng
Enrique Dunn
http://www.cs.unc.edu/~dunn
Rahul Raguram
http://www.cs.unc.edu/~rraguram
Jan-Michael Frahm
http://www.cs.unc.edu/~jmf

Department of Computer Science
University of North Carolina
Chapel Hill, NC USA

Abstract

The estimation of a complete 3D model from a set of depthmaps is a data intensive task aimed at mitigating measurement noise in the input data by leveraging the inherent redundancy in overlapping multi-view observations. In this paper we propose an efficient depthmap fusion approach that reduces the memory complexity associated with volumetric scene representations. By virtue of reducing the memory footprint we are able to process an increased reconstruction volume with greater spatial resolution. Our approach also improves upon state of the art fusion techniques by approaching the problem in an incremental online setting instead of batch mode processing. In this way, are able to handle an arbitrary number of input images at high pixel resolution and facilitate a streaming 3D processing pipeline. Experiments demonstrate the effectiveness of our proposal both at 3D modeling from internet-scale crowd source data as well as close-range 3D modeling from high resolution video streams.

1 Introduction

In recent years, there has been an explosion of activity in the area of automatic large-scale 3D reconstruction of urban environments, both from video as well as internet photo collections [1, 2, 4, 6, 17, 19, 23, 27]. Recent work has shown that traditional structure from motion techniques can be scaled up to operate on massive datasets, containing hundreds of thousands [1, 19], or even millions of images [4]. However, while much of this work has focused on building sparse 3D models (or "point clouds") from this imagery, there have been relatively fewer efforts [4, 6] focused on recovering complete *dense* 3D geometry from these large scale image collections. Indeed, while exciting progress has been made on the structure from motion problem (recovery of camera pose and sparse scene geometry), the last step in the 3D reconstruction pipeline – the recovery of dense, textured 3D models – has a number of research challenges that still remain open. While excellent results have been achieved on smaller scale problems [20, 21], operating at *internet-scale* is still a particularly challenging problem given the sheer amount of data as well its intrinsic variability.

Depthmap fusion is a standard approach for generating accurate and robust 3D scene representations using structure from motion pipelines. Fusion methods based on determining an implicit surface corresponding to the level set of a volumetric occupancy function have shown to be effective at generating 3D models for arbitrary objects. Nevertheless, the use of a discretized volumetric representation requiring cubic storage complexity hinders the applicability of these methods in terms of the magnitude and resolution of the volume of interest. Moreover, fusion methods striving for an optimal estimate of the implicit surface, typically analyze input imagery in batch mode, which again limits these methods in the total number of images that can be processed, as well as their corresponding pixel resolution. Such limitations are typically mitigated by finding a trade-off among these competing factors and the quality of the generated model. This work addresses these issues through the development of an incremental depthmap fusion module with quadratic storage complexity, enabling high-resolution large-scale modeling.

2 Related Work

In recent years there has been a tremendous effort to obtain robust depth estimation techniques for street side video [2, 3, 18] as well as for photo-collections [5, 7, 22]. Pollefeys et al. [18]obtain general scene geometry through a frame based dense depth map fusion. Hence the fusion is limited to a small set of nearby depthmaps not allowing to incorporate a large variety of viewpoints as our proposed fusion. Cornelis *et al.* [3] constraints the scene to consist of a ruled surface model by extracting for each frame a model of a slice of ground plane and two orthogonal slices of the facade. The different frame models are then combined into the final ruled surface model. Accordingly, the obtained geometry neglects any surface detail or object not on the facade plane. Similarly, Furukawa *et al.* [5] constraint the scene geometry to coincide with three orthogonal scene plane directions suppressing any general scene geometry not complying with those orthogonal directions. Sinha *et al.* [22] only constrains the scene to consist of piecewise planar segments allowing to model non-orthogonal planes but still suppressing non-planar scene segments. This was broadened by Gallup *et al.* [7] to a reconstruction only enforcing planarity where the scene geometry was planar but modeling other scene parts as general geometry. Their formulation is based on a computationally expensive Markov-Random-Field optimization. Our proposed method is allowing general scene geometry providing superior accuracy than [3, 5, 22], while being significantly more computationally efficient than [7].

To estimate the 3D model exploiting the redundancy of many depth estimates of the same structure Merrell *et al.* [15] developed fusion method for depthmaps from street side video. This method was highly computationally efficient but still produced surface models with respect to a single view, which were then concatenated to obtain the 3D scene model. Estimating 3D models from photo collections was first tackled by Goesele *et al.* [10] through a seed growing carefully selecting compatible views during the growing. Recently, Furukawa *et al.* [6] and Gallup *et al.* [8, 9] developed methods to tackle large scale photo collections of thousands of registered images for 3D model extraction from photo collections. Furukawa *et al.* [6] again deploy a careful selection of compatible images to obtain the 3D model from 6304 images within about 28 days on a single computer. Our method in contrast is significantly more efficient in the model estimation. Zach *et al.* [29, 30] use a signed distance field regularized by an L_1 total variation energy functional to obtain a volumetric scene representation, which has limited scalability due to its cubic memory complexity.

Some recent depthmap fusion proposals [8, 9] have mitigated the cubic storage complexity of volumetric representations by decoupling occupancy correlations in two of the three volume dimensions. In this way, such methods enable concurrent and independent space occupancy estimation along disjoint 1D neighborhoods of the volumetric function. Such representations can be used to estimate a *heightmap* with respect to an arbitrarily defined *ground plane* in the scene. In practice, the ground plane may be selected to be orthogonal to the principal surface directions on the scene (e.g. in urban modeling where vertical facades are predominant) to provide an implicit regularization of the obtained 3D model. Our proposed approach is most similar to the approach of Gallup *et al.* [9] and its extension in work [8], which deploys a probability occupancy grid known from robotics [14, 16] to compute volume occupancy through Bayesian inference. To improve the visual quality of the results it uses a regularization in the height direction through a limited number of transitions between empty and occupied volume segments. This approach is very computationally efficient and avoids the typically cubic memory complexity of volumetric approaches [12] for a small sets of depthmaps (<128 depthmaps) . The approach [9] was further extended by Haene *et al.* [11] for indoor environment to regularize the obtained surfaces through total variation. The major drawback of these approaches is the high storage requirement associated with batch processing. Our proposed extension of these methods overcomes this limitation allowing an unlimited number of depthmaps to be fused and allowing updating existing fused models.

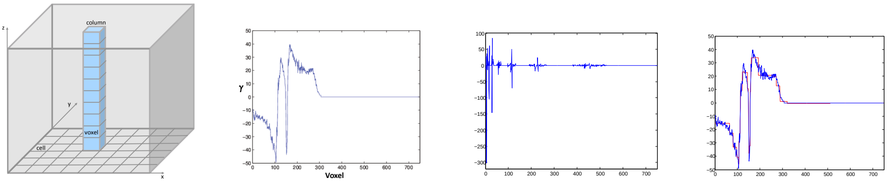
3 An Efficient Heightmap Representation

Our proposal builds upon recently proposed heightmap representations [8, 9] and introduces a wavelet based compression mechanism ÆŠ every column in the heightmap in order to attain a reduced parametric representation of each column. The heightmap fusion method takes a set of depthmaps as input, along with external and internal camera parameters. From the external camera parameters, we can determine a ground plane for the scene [25], which serves as the lower $x-y$ boundary of the fusion volume. Following this, the $x-y$-plane (ground plane) is partitioned into cells, where the x, y size of the cell matches the desired resolution of the 3D-model. For each cell we can define a column representing the volume above the cell (see Fig. 1a). One of the computational advantages of the original heightmap fusion algorithm, in contrast to earlier volumetric fusion methods, is that the fusion is solved independently within each column, allowing for easy parallelization of the fusion process.

In order to estimate the voxel occupancy function across an entire column, each voxel is backprojected to the set of input depthmaps and occupancy votes based on visibility constraints are aggregated across all views. Denoting the projected point p in the depthmap, a vote γ_p is calculated for every camera:

$$\gamma_p = \begin{cases} \lambda_{empty} & \text{if } \rho_v < \rho_p \\ -e^{\frac{|\rho_v-\rho_p|}{\sigma}} & \text{if } \rho_v \geq \rho_p \end{cases} \qquad (1)$$

with ρ_p being the depth value at pixel p, ρ_v representing the distance between the voxel and camera center and the variance σ controlling the surface thickness. If the voxel is in front of the surface seen in the depthmap, ρ_p will be larger than ρ_v. Then, the vote is for the voxel being empty, by casting a positive constant λ_{empty} vote. If the voxel is occluded by the surface measured in the depthmap, with ρ_p smaller than ρ_v , the casted vote decreases exponentially with the distance to the surface. Each voxel is initialized with occupancy 0, and the votes



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

Figure 1: (a) Volume defined by *x*-*y* ground plane and a single column volume divided into voxels. (b) Plot of the occupancy function values for 750 voxels in a given column. (c) Coefficients after applying the wavelet transform to the occupancy function. (d) The blue and red curves correspond, respectively, to the original and the recovered signal using the 20 most significant wavelet coefficients.

from the different cameras are then accumulated. A column of voxels with accumulated votes from 63 images is shown in Fig. 1 (b). Positive values mean the corresponding voxel is likely to be empty, and negative values indicate the voxel is occupied.

Once the occupancy function value of all voxels of a given column has been updated, the heightmap fusion divides the column into occupied and empty segments. The mechanisms by which the voxel labeling is performed will be presented in the following section. As presented in [9], voxel occupancy estimates may be considered simply as transient data during the voxel labeling procedure. Accordingly, voxel occupancy values are estimated, analyzed and discarded after each column has been processed. This is an intuitive approach for batch processing where all the viewing rays passing through a given voxel are readily available. Online processing in the other hand, requires us to cycle through each voxel for each incoming depthmap image. To perform such an update, the full cost volume would need to be stored, as the heightmap itself does not contain sufficient information for performing the update. This would revert the quadratic memory complexity of the heightmap fusion back to a cubic storage complexity, as is the case for traditional volumetric methods.

Our proposal builds on the insight that due to the characteristics of the volumetric occupancy data, we can use wavelet-based compression techniques to dramatically reduce the memory requirements, by compressing the data corresponding to each column of the cost volume. A related but different work [28] applies wavelets to efficiently represent and update occupancy grids for range sensors. In the Subsection 3.1, we introduce the Haar wavelet transform, and describe how our compression and decompression of the per column occupancy are performed.

3.1 Wavelet-based compression of the cost volume

There are two main observations that we leverage: (a) in the basic heightmap formulation, columns of the 3D volume are processed independently of each other and (b) when considering a single column (see Fig. 1 (b)), it is typically the case that the votes change smoothly within a segment, with sharp transitions between adjacent segments. These sharp transitions correspond to the change from occupied to empty space in the physical world. Accordingly, wavelet based compression techniques are very well suited for modeling such volumetric data, as the transitional intervals of the input data signal can be accurately represented using a small number of wavelet coefficients [26].

We have found that for our application the Haar wavelet provides a suitable tradeoff between implementation efficiency and data fidelity. In general, one of the limitations of the Haar wavelet is that it is not continuous, and thus, not differentiable. However, it is also precisely this property that provides an advantage when analyzing signals that contains sharp transitions [13]. For the Haar wavelet, the scaling function and all the wavelets are orthonormal, and span a vector space. In this paper, when we refer to a wavelet basis, this includes both the scaling function and the set of wavelets. Given the original signal and the orthonormal basis, the coefficients of the transform can be calculated by convolving the signal with the wavelet basis. In practice, however, we build the wavelet transformation through a lifting scheme [24], which allows us to perform an *in-place* computation.

We now broadly overview the computational framework for wavelet transformation. Consider a signal S_n of length 2^n. For each pair of data points S_{2l} and S_{2l+1}, $l = 1, 2, ... 2^{n-1}$, we apply the following operation and denote the results by A and D :

$$A_{2l} = \frac{S_{2l} + S_{2l+1}}{\sqrt{2}}, \; D_{2l} = \frac{S_{2l} - S_{2l+1}}{\sqrt{2}} \tag{2}$$

In order to achieve in-place calculation, Eq. 2 is changed to

$$D_{2l} = \frac{S_{2l} + S_{2l+1} - 2S_{2l+1}}{\sqrt{2}} = \frac{S_{2l} + S_{2l+1}}{\sqrt{2}} - \sqrt{2}S_{2l+1} = A_{2l} - \sqrt{2}S_{2l+1} \tag{3}$$

In other words, we can apply Eq. (2), and store A_{2l} in the location for S_{2l}. In turn, this value is further used when computing Eq. (3), and we can store D_{2l} in the location for S_{2l+1}.

In brief, the above operations recursively decompose the original signal into an approximation part A and a detail part D, where the approximation part A can be thought of as the low frequency component of the signal, while the detail part D may be thought of as providing the finer, high-frequency components. By recursively applying this procedure to the each obtained A approximation, we can obtain a complete set of detail D coefficients after $n-1$ recursions. The approximation part of the final step, together with all the detailed coefficients are the final output of the wavelet transform. The inverse wavelet transformation is done in a similar way, but in the reverse direction.

It can be seen from the discussion above that the wavelet transform produces as many coefficients as there are elements in the original signal. However, the power of the wavelet transform comes from the fact that following the transform, the information (or energy) is statistically concentrated in just a few coefficients (see Fig. 1c). Thus, some signal $f(x)$ can be expressed as: $f(x) = \sum_{i=1}^{m} c_i u_i(x)$ where u_i, $i = 1, 2...m$ form the wavelet basis. The coefficients are sorted so that $|c_{\sigma(1)}| \geq |c_{\sigma(2)}|..., \geq |c_{\sigma(m)}|$, where $\sigma(1), \sigma(2)..., \sigma(m)$ is a permutation of $1, 2..., m$. Given some desired compression rate, the smallest $m - \tilde{m}$ coefficients are set to 0. The decompression step proceeds as follows: the coefficients $c_{\sigma(1)}, c_{\sigma(2)}, ..., c_{\sigma(\tilde{m})}$ are used to recover the signal: $\tilde{f}(x) = \sum_{i=1}^{\tilde{m}} c_{\sigma(i)} u_{\sigma(i)}(x)$

The red curve in Fig. 1(d) shows the recovered signal for a given volume column using the 20 most significant coefficients of the wavelet transform, representing the original input signal comprising of 750 samples. It can be observed that while there are marginal local errors in the smooth parts of the signal as a compression artifact, the sharp transitions are well modeled and can be very accurately reconstructed.

Applying wavelet compression to the occupancy function volume reduces the memory complexity from $O(n^3)$ to $O(kn^2)$, given that we can approximate each column in the volume using a constant number k of wavelet coefficients ($k = 30$ in all our experiments). The required k for recovering the correct occupancy information is significantly smaller than n,

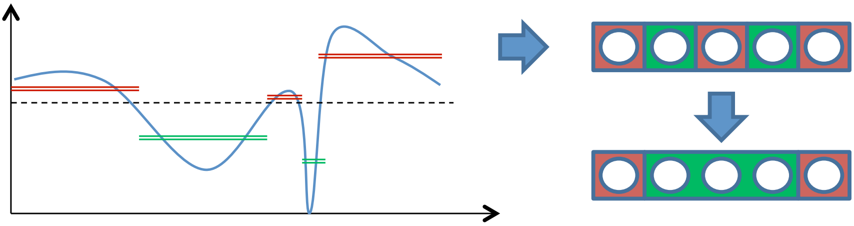


Figure 2: Voxel occupancy function simplification. The occupancy profile for a given column is segmented into intervals of homogenous classification. Each interval is considered a segment and is assigned the average function value over the interval. In turn, each segment is considered an atomic element to be assigned a labeled during multi-layer assignment.

and it does not grow proportionally to the number of voxels in a column. An intuitive and direct illustration is given below. Given two different column representations Col_1 and Col_2 placed at the same position, but with Col_1 having twice the number of voxels than Col_2, the occupancy signal of these two columns will have similar overall trend. The smooth part of occupancy in Col_1 will be more likely the linear upsample of the corresponding part in Col_2. Hence, for the smoothing part of the signal, doubling the number of voxels only increases the number of detail coefficients D with small magnitude (based on Equation (2)). The required k is closely related to the expected number of transitions in the signal. There is a computational overhead involved in performing the compression and decompression procedure. However, we note that 1D wavelet coefficient estimation can be performed *in-place* and is amenable to parallelization. Moreover, this overhead is negligible compared to the computational cost of our dynamic programming voxel segmentation introduced in Section 4.

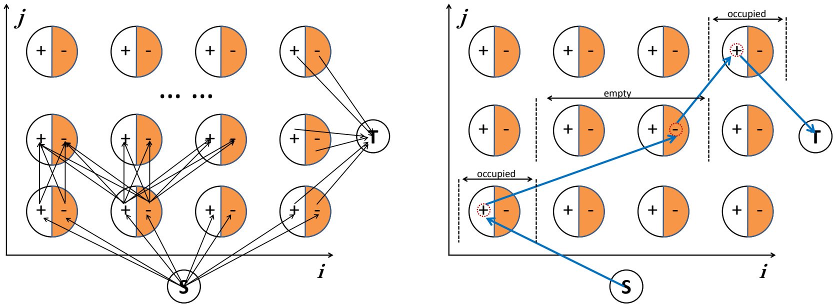
4 Incremental Multi-Layer Estimation

The goal of our multi-layer heightmap estimation module is to transform the input occupancy function values for all the voxels belonging to a given column into an output binary segmented set of occupancy layers. We model the associated segmentation problem as a *directed acyclic graph* traversal problem and efficiently find the optimal solution through dynamic programming (DP). Our multi-layer approach differs from [9] in the following:

1. We deploy a general graph structure that is not restricted in regard to the parity of the number of layers nor in the topology of the assigned layers. In [9] strong assumptions on the observed scene needed to be made to define the graph structure. Accordingly, to model either indoor or outdoor scenes the author recommends defining a priori an odd or even maximum number of layers.

2. We perform the dynamic programming in a reduced search space. We identify and quantify segments of homogeneous occupancy classification and use them as atomic elements to be labeled by the DP procedure (see Figure 2). The importance of this pre-processing is that it enables more efficient processing at finer voxel resolutions while mitigating the memory requirements of DP search.

The computational framework for our multi-layer estimation is described as follows. Once the updated occupancy function values for all the pixels in a given column have been



(a) (b)

Figure 3: (a) The graph models the problem of fitting the layers onto the segments. (b) Example path from source S to sink T, with 4 segments. The blue path depicts the first and last segment are configured as an occupied layer while the second and third as empty.

extracted, we identify the zero-crossings of the corresponding function profile. N zero-crossings split the column into $N+1$ segments. The first type of segments include voxels with accumulated votes bigger or equal to 0, while the other type only contains voxels with negative votes. The average vote within a segment is used to represent the segment's state, denoted as Γ_i for segment i. This is directly proportional to the typically used sum of the occupancies of the segment [8, 9].

The next step is to fit the layers onto the segments with state Γ. To fit the layers onto segments, we cast the optimization as a graph traversal optimization. As is illustrated in Fig. 3(a), directed edges are inserted from every node in one layer to the nodes in the next. There are four edges ($+ \rightarrow -$, $+ \rightarrow +$, $- \rightarrow +$, $- \rightarrow -$) connecting every two nodes. In this graph, the height of the node array (in j direction) equals the user define maximum number of layers. The width of the node array (in i direction) equals the number of segments.

One path from the source node S to the sink node T represents one configuration of layers. Let $V(j,i,s)$ denote the node at row j, column i in the node array with sign $s \in \{+,-\}$ as shown in Fig. 3(a). If the edge from $V(j,i_1,s_1)$ to $V(j+1,i_2,s_2)$ is part of the shortest path from source S to sink T, then all the nodes from i_1+1 to i_2 are part of the same segment defining a transition layer. The occupancy of the transition layer is determined by s_2. If s_2 is $+$, it means the layer is occupied, and vice versa. Fig. 3(b) gives a concrete example. The cost of the directed edge from node $V(j,i_1)$ to node $V(j+1,i_2)$ is defined as:

$$Cost(V(j,i_1,s_1),V(j+1,i_2,s_2)) = \begin{cases} \sum_{k=i_1+1}^{i_2} \Gamma_k & \text{if } i_1 < i_2 \text{ and } s_2 = + \\ -\sum_{k=i_1+1}^{i_2} \Gamma_k & \text{if } i_1 < i_2 \text{ and } s_2 = - \\ +\infty & \text{if } i_1 \geq i_2 \end{cases} \tag{4}$$

Here the cost equals infinity if $i_1 \geq i_2$, as the layer is not allowed to go back. Γ_k is the average occupancy vote representing the occupancy of segment k. Based on Equation 2, positive Γ_k means the segment is likely to be empty, so the negative factor in the second Equation of 4 encourages the edge to be part of the shortest path, which further means this segment is likely to be configured as part of an empty layer. The edges connecting the sink node T and node array have no weight. Edges connecting the source node S and node array have an edge weight similar to Equation 4: $Cost(S,V(1,i)) = \alpha * \sum_{k=1}^{i} \Gamma_k$.

While the upper bound on the number of layers is an input parameter, the algorithm determines on a per column basis the optimal number of representative layers. Most columns in the heightmap will not need n layers. Hence, if not removed, the extra layers will be free to fit the noise in the measurements. To mitigate this issue we introduce a layer penalty C corresponding to the Bayesian Information Criterion as used in [9]. So Equation 4 becomes:

$$Cost(V(j,i_1,s_1),V(j+1,i_2,s_2)) = \begin{cases} \sum_{k=i_1+1}^{i_2} \Gamma_k & \text{if } i_1 < i_2,\, s_2 = +,\, s_1 = + \\ \sum_{k=i_1+1}^{i_2} \Gamma_k + C & \text{if } i_1 < i_2,\, s_2 = +,\, s_1 = - \\ -\sum_{k=i_1+1}^{i_2} \Gamma_k & \text{if } i_1 < i_2,\, s_2 = -,\, s_1 = - \\ -\sum_{k=i_1+1}^{i_2} \Gamma_k + C & \text{if } i_1 < i_2,\, s_2 = -,\, s_1 = + \\ +\infty & \text{if } i_1 \geq i_2 \end{cases} \quad (5)$$

Where we add a cost each time one new layer is generated, i.e s_2 has different sign with s_1. Please note that our proposed graph has redundancy, i.e. some paths will have the same cost and the same layer configurations, hence it still finds the optimal layer configuration.

Finding the optimal segments defining the layers is equivalent to finding the shortest path from source node S to sink node T. The problem can be computed with dynamic programming. In order to efficiently find the shortest path to the nodes in layer j, the shortest path and cost from S to every node in layer $j-1$ should be saved.

5 Experiments

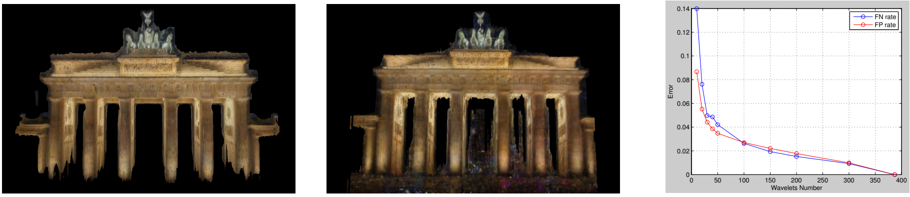
In our experiment, camera poses and depthmaps are computed using the pipeline of [4]. The number of images and size of the volume for each model are depicted in Table 1. For the parameters of the depthmap fusion, λ_{empty} is 0.7, $sigma$ is 0.1, the layer cost C is 20, and the maximum number of layers for each column is set 9 for all experiments. We compared our depthmap fusion approach against the one proposed in [9]. Fig. 4(a) (b) illustrate an improved robustness to noise provided by our method. Additional results are depicted in Figure 5. The approach in [9] needs to first spatially partition the model and cluster input depthmaps into sets of 128 images to be able to mitigate batch storage requirements. We can see how our online method benefits from being able to use all available information for every voxel leading to a robust 3D reconstruction. We also uilized the output of a SfM pipeline for close range scene reconstruction. We utilized a 16MP DSLR camera to obtain a total of 100 images (randObject dataset). Figure 6 depicts the sequential process of incrementally building the 3D model through online depthmap fusion.

We quantitatively evaluate our online method by comparing the final model with one computed without the procedure of compression and decompression, which is taken as ground truth (See Fig. 4(c)). All the parameters are set the same except the number of wavelets. 150k Columns are randomly sampled from volumes, with each having 387 voxels. Rate of false negatives and rate of false positives are computed: $FN\_rate = \frac{\text{number of FN}}{T}$, $FP\_rate = \frac{\text{number of FP}}{T}$, in which T is the total number of occupied voxels within the ground truth. Using 30 wavelets typically yields visually similar results to the ground truth.

Our depthmap fusion runs at 33 Hz on one NVIDIA Tesla C2050/C2070 graphics card. Computing a 9-layer heightmap on a volume of 100x100x100 given one image takes 24.9 ms, within which the procedure of compression and decompression only takes 1.6 ms.

Model	# of images	# of columns	# of voxels per column
Brandenburg Gate	4545	1000k.	512
Reichstag	2638	4000k	512
Berlin Dome	2681	500k	512
randObject	100	1440k	1000

Table 1: dataset used in experiment



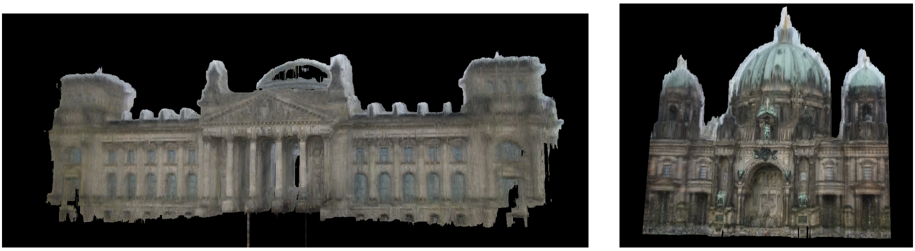
(a) Our Results (b) Results from [9] (c) Quantitative evluation

Figure 4: (a) and (b) are comparative results for crowd sourced data. Both images show 3D models of the Brandenburg Gate. (c) shows voxel errors for different number of wavelets.

6 Discussion and Future Work

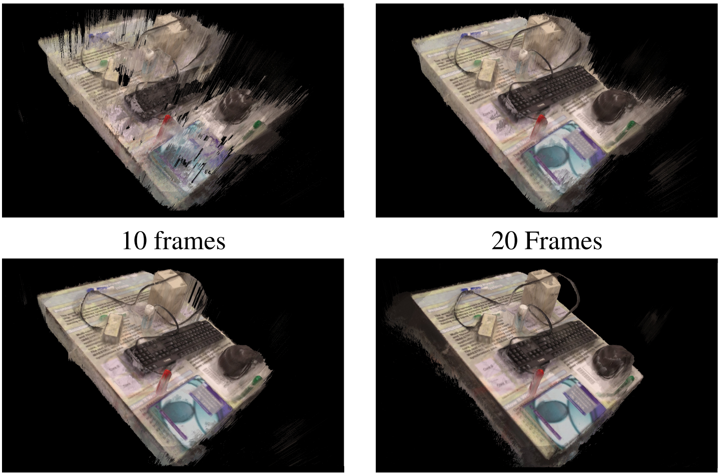
We have proposed an efficient and scalable heightmap based depth fusion framework. Our proposal reduces memory requirement through the incorporation of an efficient data compression technique and a reformulation of the binary segmentation problem associated with multi-layer heightmap estimation. Our incremental (online) fusion framework is well suited for 3D modeling from both asynchronous or streaming input sources. Extensions to our framework include exploring the use of compression scopes greater than a single volume column. Clearly the tradeoff between compression rate and signal reconstruction is a critical issue to study. However, a derived research path along these lines is to investigate mechanism to incorporate implicit spatial regularization to wavelet based compression/decompresion of the volumetric occupancy function. Moreover, additional efficiencies may be achieved by performing volume updating directly on the wavelet domain.

Acknowledgement. We acknowledge the support of NSF grant IIS-0916829, the Department of Energy under Award DE-FG52-08NA28778 and the US Navy SPAWAR Center.



Reichstag Berliner Dom

Figure 5: Additional 3D models from crowd sourced data.



10 frames

20 Frames

40 Frames

100 Frames

Figure 6: Incremental improvement of 3D model by online depthmap fusion.

References

[1] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building Rome in a day. In *ICCV*, 2009.

[2] N. Cornelis, B. Leibe, K. Cornelis, and L. Van Gool. 3d urban scene modeling integrating recognition and reconstruction. *IJCV*, 2008.

[3] Nico Cornelis, Kurt Cornelis, and Luc Van Gool. Fast compact city modeling for navigation pre-visualization. In *Computer Vision and Pattern Recognition (CVPR)*, 2006.

[4] Jan-Michael Frahm, Pierre Fite-Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, and Marc Pollefeys. Building rome on a cloudless day. In *ECCV*, 2010.

[5] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Manhatten-world stereo. In *In Proceedings IEEE CVPR*, 2009.

[6] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Towards internet-scale multi-view stereo. In *CVPR*, 2010.

[7] David Gallup, Jan-Michael Frahm, and Marc Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *CVPR*, 2010.

[8] David Gallup, Jan-Michael Frahm, and Marc Pollefeys. A heightmap model for efficient 3d reconstruction from street-level video. In *3DPVT*, 2010.

[9] David Gallup, Marc Pollefeys, and Jan-Michael Frahm. 3d reconstruction using an n-layer heightmap. In *DAGM*, 2010.

[10] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. Multi-view stereo for community photo collections. In *ICCV*, 2007.

[11] C. Haene, C. Zach, J. Lim, A. Ranganathan, and M. Pollefeys. Stereo Depth Map Fusion for Robot Navigation. In *Proceedings IROS*, 2011.

[12] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38:199–218, 2000. ISSN 0920-5691. URL http://dx.doi.org/10.1023/A:1008191222954. 10.1023/A:1008191222954.

[13] Stephane Mallat. A wavelet tour of signal processing, 1998.

[14] D. Margaritis and S. Thrun. Learning to locate an object in 3d space from a sequence of camera images. In *ICML*, 1998.

[15] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nister, and M. Pollefeys. Real-Time Visibility-Based Fusion of Depth Maps. In *Proceedings of International Conf. on Computer Vision*, 2007.

[16] K. Pathak, A. Birk, J. Poppinga, and S. Schwertfeger. 3d forward sensor modeling and application to occupancy grid based sensor fusion. In *IROS*, 2007.

[17] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3d reconstruction from video. *Int. Journal of Computer Vision (IJCV)*, 2008.

[18] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59:207–232, 2004. ISSN 0920-5691.

[19] Rahul Raguram, Changchang Wu, Jan-Michael Frahm, and Svetlana Lazebnik. Modeling and recognition of landmark image collections using iconic scene graphs. *International Journal of Computer Vision*, 95(3):213–239, 2011.

[20] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. Journal of Computer Vision (IJCV)*, 2002.

[21] Steve Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer Vision and Pattern Recognition (CVPR)*, 2006.

[22] S. N. Sinha, D. Steedly, and R. Szeliski. Piecewise planar stereo for image-based rendering. In *In Proceedings IEEE ICCV*, 2009.

[23] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH*, pages 835–846, 2006.

[24] Wim Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Appl. Comput. Harmon. Anal.*, 1996.

[25] Richard Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Computer Vision*, 2:1–104, 2006.

[26] Wikipedia. Wavelet transform, 2011. URL "http://en.wikipedia.org//Wavelet_transform".

[27] Jianxiong Xiao and Long Quan. Image-based street-side city modeling. In *Siggraph Asia*, 2009.

[28] Manuel Yguel, Christopher Tay, Meng Keat, Christophe Braillon, Christian Laugier, and Olivier Aycard.

[29] C. Zach. Fast and high quality fusion of depth maps. In *in Proc. International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT)*, 2008.

[30] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust TV-L1 range image integration. In *in Proc. IEEE International Conference on Computer Vision (ICCV)*, 2007.