A Heightmap Model for Efficient 3D Reconstruction from Street-Level Video

David Gallup[1], Jan-Michael Frahm[1]
[1] University of North Carolina
Department of Computer Science
{gallup,jmf}@cs.unc.edu

Marc Pollefeys[12]
[2] ETH Zuerich
Department of Computer Science
{marc.pollefeys}@inf.ethz.ch

Abstract

This paper introduces a fast approach for automatic dense large scale 3D urban reconstruction from video. The presented system uses a novel multi-view depthmap fusion algorithm where the surface is represented by a heightmap. Whereas most other systems attempt to produce true 3D surfaces, our simplified model can be called a 2.5D representation. While this model seems to be a more natural fit to aerial and satellite data, we have found it to also be a powerful representation for ground-level reconstructions. It has the advantage of producing purely vertical facades, and it also yields a continuous surface without holes. Compared to more general 3D reconstruction methods, our algorithm is more efficient, uses less memory, and produces more compact models at the expense of losing some detail. Our GPU implementation can compute a 200×200 *heightmap from 64 depthmaps in just 92 milliseconds. We demonstrate our system on a variety of challenging ground-level datasets including large buildings, residential houses, and store front facades obtaining clean, complete, compact, and visually pleasing 3D models.*

1. Introduction

Automatic large-scale 3D reconstruction of urban environments from ground reconnaissance video or active sensors like LiDAR is a very active research topic on the intersection of computer vision and computer graphics [2, 3, 7, 15, 13]. The applications of these techniques are very broad from augmenting maps like in Google Earth or Microsoft Bing Maps, civil and military planning, to entertainment. In this work, we focus on reconstructions from ground reconnaissance video since ground reconnaissance data is easier and cheaper to acquire as well as the video cameras are significantly less expensive than active sensors like LiDAR. Additionally, ground-level reconstructions can be used to compliment existing reconstructions from aerial and satellite platforms or manually created models for example Google Sketchup models, providing greater detail from a pedestrian perspective.

One important aspect of most current research efforts is computational efficiency to enable the modeling of wide-area urban environments such as entire cities. The data sets resulting from data collection of these areas are massive. Even a small town may require millions of frames of video just to capture the major streets. The reconstruction algorithms deployed must be fast in order to finish processing in a reasonable amount of time. Additionally, the generated models need to be compact in order to efficiently store, transmit, and render them.

To address these needs, we present a novel multi-view depthmap fusion algorithm deployed as a post processing step on the outputs of systems like for example [7]. The processing with our fusion results in a heightmap model for the scene, *i.e*. for every point defined over a horizontal grid, our algorithm estimates a single height value. Whereas most other systems attempt to produce true 3D surfaces, our simplified model is a 2.5D representation. This simplified model leads to an efficient algorithm on one hand and to a simplified model representation on the other hand meeting the requirements for simplification and reconstruction efficiency.

Similarly to volumetric reconstruction approaches in our method a 2D horizontal grid is defined over the region of interest. To avoid the explosion in memory of the volumetric approaches, we define for every 2D grid cell, a height value computed to minimize the amount of free space below and the amount of filled space above the value. Free and filled space votes are accumulated from the viewing rays of all depthmaps that intersect the vertical column in space defined by the 2D grid cell and the up-vector. Then a polygonal mesh is generated from the heightmap, and texture maps are generated from the images. Facades of buildings are of particular interest in urban modeling. In our heightmap representation, facades appear as large depth gradients between the roof tops and the ground below. These height discontinuities are detected with a threshold and strictly vertical polygons are generated to connect the ground and roof. See Figure 1 for an overview.

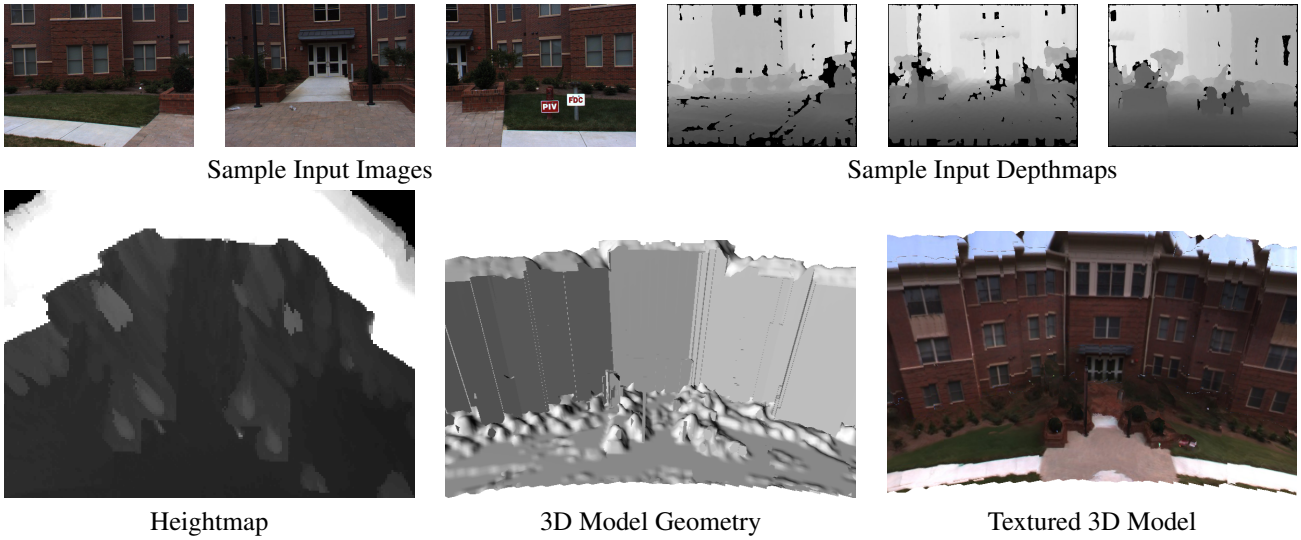


Figure 1. Our novel depthmap fusion approach uses a vertical heightmap model to reconstruct a textured 3D mesh. Our simplified model cannot represent overhanging surfaces, but it is well-suited for vertical facades, and delivers a compact and complete model without holes.

One limitation of our model compared to a general geometry representation is that it does not accommodate overhanging surfaces such as eves, inset windows, and larger trees. However, it has the advantage of producing purely vertical facades, and it also yields a continuous surface without holes. Thus our algorithm presents a trade-off compared to more general 3D reconstruction methods. It is more efficient, robust, and produces more compact models at the expense of losing some detail.

By design the proposed method is well-suited for processing large amounts of video naturally fusing all votes within the entire vertical column to support the height value decision. Hence it obtains robust and visually pleasing results even without any additional regularization. Thus the height value at every grid cell can be computed independently enabling parallelization. Our GPU implementation can compute a 200×200 heightmap in just 92 milliseconds.

We demonstrate the quality of the modeling of our of our approach for a variety of challenging ground-level datasets including large buildings, residential houses, and store front facades. Even though some details cannot be captured by our model, the resulting reconstructions are clean, complete, and compact.

2. Related Work

There are two parts to a 3D reconstruction from video system. The first part is the camera motion estimation from the video frames commonly called structure from motion or sparse reconstruction. The second part is the so called dense reconstruction, which obtains a dense scene geometry using the known camera poses and the video frames. In some approaches [7] the robustness and drift of the sparse estimation is improved through additional sensors such as INS and GPS, which also remove the ambiguity in scale inherent in structure from motion, and provide an absolute coordinate system. The dense subproblem refers to performing stereo matching, depthmap fusion, or other means to generate a 3D model as for example LiDAR [3]. This paper focuses on the dense subproblem, and can be viewed as a restricted depthmap fusion algorithm.

There is a wealth of 3D reconstruction methods that address the dense subproblem. A taxonomy of stereo algorithms is given by Sharstein and Szeliski [9]. Seitz *et al*. [10] give an overview of multi-view stereo for object-centered scenes. 3D reconstruction from video has been addressed in Pollefeys *et al*. [8]. Pollefeys *et al*. use uncalibrated hand-held video as input and obtained reconstructions of hundreds of frames, but could not handle wide-area scenes. The system presented by Pollefeys *et al*. [7] was designed to process wide-area scenes. The resulting 3D models are general 3D surfaces represented as texture-mapped polygonal meshes coming with all the problems like holes in homogneous areas, in windows and slightly inaccurate geometry on facades deviating from the true planar geometry causing visually disturbing artifacts. In contrast, our system aims to fit a simple heightmap model to provide greater efficiency, robustness, compactness and a water tight surface delivering a visually more pleasing model.

There are several recent approaches deploying simplified geometries [1, 2, 4, 11]. Cornelis *et al*. [1] used a simple U-shaped ruled surface model to efficiently produce compact street models. To enhance the appearance of cars and pedestrians not modeled through the ruled surface model Cornelis *et al*. extended the approach to detect and replace those

through explicit template models [2]. While our approach also determines a simplified geometry the heightmap model is far less restrictive than the ruled surface model and can model most urban objects and scenes to a large degree. Our approach naturally models, cars, pedestrians, lamp posts, and bushes within the heightmap framework. Furukawa *et al*. [4] uses a very specific Manhattan-world model, where all planes must be orthogonal, and Sinha *et al*. [11] uses a general piecewise planar model. Non-planar surfaces are not handled well and are either reconstructed with a staircase appearance or are flattened to nearby planes. The approach proposed in this paper is far less limited than the approaches [4, 11] since the heightmap is able to model general geometry except for overhanging surfaces.

Our technique operates conceptually over the occupancy grid of the scene (though this is never explicitly required during the computation). Other depthmap fusion techniques, such as Zach *et al*. [14], also use an occupancy grid for depthmap fusion but require the occupancy grid to be present leading to limitations on the model resolution. Whereas these other methods recover a general 3D surface from the present occupancy grid, our method simplifies the fusion problem by recovering only a heightmap. This allows our method to be much more efficient in terms of processing time and especially memory.

Several methods have been aimed directly at modeling buildings from street-side imagery. Xiao *et al*. [13] present an automatic method that learns the appearance of buildings, segments them in the images, and reconstructs them as flat rectilinear surfaces. The modeling framework of Xiao *et al*. does not attempt to geometrically represent other scene parts like vegetation typically present in many urban scenes. Our method also targets building facades captured from street-level video for which the heightmap model is ideal. The heightmap also other objects typically present in urban scenes like cars, lamp posts, and mail boxes to the degree that they have no overhanging structure.

3. Heightmap based Stereo Fusion

Our method aims at the dense 3D urban reconstruction. The inputs to our method are one or more video sequences, the estimated camera poses and the intrinsic calibration for every frame, a depthmap for every frame, and an estimate of the world's vertical or gravity direction. Camera parameters can be recovered with Structure from Motion (SfM) techniques [8] as well as inertial sensor data and GPS measurements [7]. Depthmaps can be computed robustly and efficiently using GPU-accelerated multi-view stereo [5]. The vertical or gravity direction can be easily obtained from the inertial sensors or from the vertical vanishing point in each image [12]. The output of our method is a textured 3D polygonal mesh.

(a) (b) (c)

Figure 2. (a) Heightmap computed without axis alignment. (b) Heightmap computed with axes aligned to dominant surface normal. Dark red indicates complete blocks where a CUDA kernel can diverge with an early exit. (c) Bright red indicates detected heightmap discontinuities.

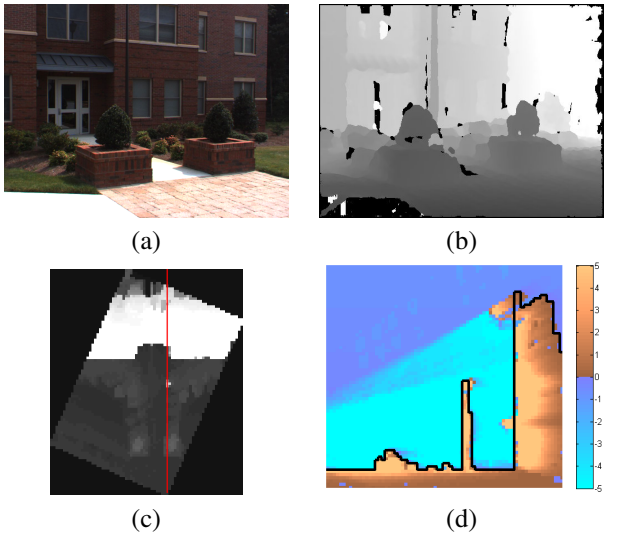
3.1. Single Heightmap fusion

This section describes the proposed method for reconstructing a local subset of video frames. The fusion of multiple of these for large-scale reconstruction is described in Section 3.2.

Step 1: Grid Alignment. The heightmap grid is first defined with respect to one of the input cameras, called the reference view. According to the desired spatial resolution of the computed model the size and resolution of the heightmap are defined by variables x_{min}, x_{max}, y_{min}, y_{max}, Δx, and Δy. Similarly, the expected height range and height resolution are defined by z_{min}, z_{max}, and Δz. Please note that while x_{min}, x_{max}, y_{min}, y_{max}, Δx, and Δy directly influence the memory consumption of our computation, the height range and the height resolution do not increase the memory consumption of the algorithm (except for temporary local storage).

Facades and other vertical surfaces will appear as discontinuities in our heightmap. To avoid staircase discontinuities, we align the grid's x and y axes to the dominant surface normal. This can be done with a surface normal voting procedure. Normals are computed from the depth gradients in the reference view. Aligning the x and y axes requires only a rotation about the vertical direction. Each normal is projected to the $x-y$ plane, and the angle to the x axis is computed. Votes are accumulated in a histogram, and the angle with the most votes is chosen. The grid axes are then rotated to align with the chosen angle. See Figure 2a-b.

With the heightmap grid defined, all the views in the video sequence that include the volume $[x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$ or parts of it in their viewing frustum are loaded and their estimated scene depths will be used in the subsequent steps.



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

Figure 3. (a) Sample input image. (b) Sample input depthmap. (c) Computed heightmap. (d) Slice of the occupancy volume corresponding to the vertical red line on the heightmap. Votes for every voxel indicate full (positve) or empty (negative). The black line shows the resulting height values for the slice.

Step 2: Heightmap Computation. The next step is to compute the height value for every cell in the heightmap. To ensure parallel computability each cell in the height map is computed independently. The vertical column corresponding to the cell is divided into voxels ranging from z_{min} to z_{max} at Δz intervals. Each voxel v is projected into every depthmap, and a vote ϕ_p for for each depth pixel p is computed as follows:

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

where ρ_p and ρ_v are the depth or range values for the pixel and the voxel with respect to the depthmap's camera. Thus each pixel votes *empty* if it's depth value is beyond the voxel, and *full* if it's depth value is in front of the voxel. λ_{empty} controls the relative weighting between empty and full votes. In the $\rho_v \geq \rho_p$ (full) case, the vote falls off to reflect the uncertainty of knowing what is *behind* an observed surface. The mean vote ϕ_v is stored in each voxel and represents a confidence of being either full (positive) or empty (negative). The height value z for the column is then chosen to minimize

$$c_z = \sum_{v_z > z} \phi_v - \sum_{v_z < z} \phi_v \quad (2)$$

where v_z is the z coordinate of the center of each voxel. Thus z is chosen so that most of the voxels above it are emtpy and most voxels below it are full. Figure 3 shows the accumulated votes and final height values for a slice of the volume of interest.

Step 3: Mesh Generation. Once a heightmap is computed, the next step is to create a polygonal mesh. This step is used to enforce the planar representation of facades and walls in general urban scenes. These structures will represent height discontinuities in our representation. Hence we detect these discontinuities between neighboring height values by thresholding of the absolute height difference with λ_{disc} . To model our domain knowledge of facades and walls corresponding to these discontinuities we generate planes to span them. An example of the meshing result is shown in Figure 2c.

Step 4: Texture Mapping. Each single heightmap represents the geometry measured by all views that include the volume $[x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$ or parts of it in their viewing frustum as explained in step 1. Hence we need to generate a texture as a composite of all images observing this scene geometry as generally no single image observes the full geometry. Furthermore the texture generation needs to be robust to occlusion.

Our method generates initially blank texture maps with texture coordinates assigned to every vertex in the mesh. Once the texture map has been layed out on the surface, the 3D point corresponding to each pixel can be computed. That point is projected into all the images, and the color of each point is stored in an array. The texel color is computed simply as the median value of the array for each color channel. We experimented with other texture map generation methods, such as that of [13], but found this method to be faster and more robust to occlusion.

Steps 1-4 are repeated for every reference view desired to represent the scene. These are typically chosen in a way to cover the entire scene with moderate overlap. Hence the next step is to compute the model combining all the resulting separate models.

3.2. Fusion of Multiple Heightmaps

The fusion of multiple heightmaps corresponding to the different reference views starts with overlaying the meshed models for all separate heightmaps. Note that the reference frame is initially used only to determine the location of the heightmap–all views contribute equally to the solution. In order to eliminate redundant computation, we mask out any cells of the current heightmap that fall within the previously computed heightmap. The selection of the view for which a heightmap is computed is dependent on the camera motion and is performed dynamically. It ensures to compute heightmaps frequently enough to avoid gaps in the reconstruction and keeps the heightmaps spaced far enough apart so that there are a sufficient number of new cells to compute to maximize computational throughput. This is especially important for the parallel implementation, where

Parameters	
$[x_{min}, x_{max}]$	$[-5m, 5m]$
$[y_{min}, y_{max}]$	$[5m, 20m]$
$[z_{min}, z_{max}]$	$[-3m, 15m]$
$\Delta x, \Delta y, \Delta z$	$20cm$
λ_{empty}	0.5
σ	1

Figure 4. We use these same parameters for all our experiments. Note that the scale of our scene is known due to the GPS data.

there needs to be enough new cells to keep the GPU hardware fully utilized.

We have implemented the depthmap fusion (Step 2 in Section 3.1) and texture map generation (Step 4 in Section 3.1) steps on the GPU using NVIDIA's CUDA platform [6]. While our method can be parallelized using any technology, we found CUDA to have a number of advantages. First, CUDA provides shared memory which multiple threads can access. We use this to our advantage by having a different thread compute the votes for each voxel in a vertical column, writing the results to an array in shared memory. After the vote accumulation, one of the threads is assigned the task of looping over the array to select the height value, which minimizes equation 2. Second, CUDA allows for divergent branching at a block level. (Each block is composed of multiple threads, see [6] for details.) This allows for greater efficiency when all cells assigned to a block have been masked out due to overlapping the previous heightmap. In that case, the block of threads can terminate quickly, freeing up resources for the next block. Figure 2b shows the blocks of a heightmap layout that can take advantage of this early exit divergence.

4. Results

In this section we describe the evaluation of our algorithm on video data captured from a vehicle-mounted multi-camera array. The array consisted of four Point Grey Flea2 color cameras with a resolution of 1024x768 pixels. Three of the cameras were aimed horizontally at 50, 90, and 130 degrees to the driving direction, and one was aimed at 90 degrees horizontally and 30 degrees vertically to the driving direction. Each camera had roughly a 40 degree horizontal and 30 degree vertical field of view, giving the array a combined 120 degree horizontal field of view and 60 degree vertical field of view. The capture system was augmented by an Applanix POS-LV system, which combines inertia sensors and GPS.

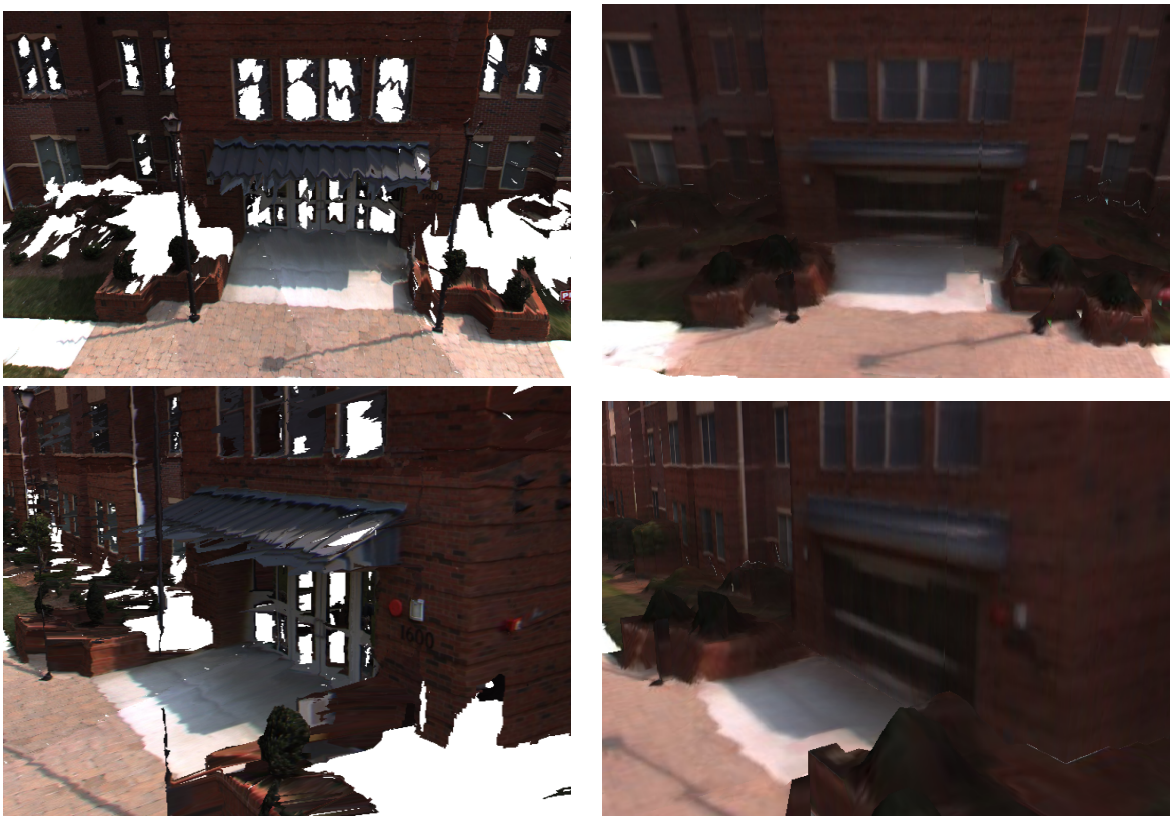
Stereo depthmaps are computed using the real-time GPU-accelerated multi-view planesweep method of Kim *et al*. [5]. It outputs depth maps for each video frame along with the camer a poses. Then the images, depthmaps, and camera poses are fed into our system to produce the 3D models. We use the same set of parameters, shown in Figure 4 for all our experiments. Our method was executed on a commodity PC: Intel Pentium 4 3.2 Ghz CPU, 1.5 GB RAM, Nvidia GeForce GTX 285 GPU. On this system, our method takes 92 ms to generate the heightmap, 203 ms to generate the mesh, and 696 ms to generate the texture map, for a total of 983 ms per reference view. Since only a subset of views (roughly every 20th) is used as reference view (see Section 3.2 for details), our method effectively runs at 20.3 Hz for the test scenes used.

Furthermore, our system, using the parameters in Figure 4, produces highly compact 3D models. For every linear meter driven by the capture vehicle, our 3D models require on average only 7500 triangles and 2.3 kilobytes of JPEG compressed texture maps. (These sizes grow linearly with the Δx and Δy parameters.) Thus our method is ideal for reconstructing wide-area urban environments, such as entire cities.

Our system was tested on a variety of different types of scenes, including streets with large buildings, residential houses, and store front facades. We have produced textured polygonal meshes directly from the depthmap data, and compared them to the models produced by our system in Figure 5. Our method performs quite well for surfaces that fit our heightmap model, especially facades, which are of primary interest for city modeling. The depthmap models in general have more detail, and better handle overhanging roofs and trees. However, our models have the advantage of a clean and simple geometry providing a visually more pleasing appearance. Our heightmap model also produces a continuous surface without holes whereas the holes in the depthmap reconstruction are due to areas of the scene which are never observed. In some sense our algorithm may be hallucinating these surfaces, and indeed the textures for the surfaces originate from views which don't actually see them. Nevertheless the completeness of the surface is an advantage. The holes of the general geometry model along with artifacts around windows cause significant visual disturbances when viewing the model. In contrast our algorithm forces the geometry of these artifacts into a planar scene determined by the surrounding surfaces. This regularizes a lot of the artifacts away.

5. Conclusion

In this paper we introduced an efficient depth map fusion for the large-scale 3D reconstruction of urban scenes. The novel method fuses multiple depthmaps into a height map representation ensuring a continuous surface. These models produce in many occasions visually more pleasing results than the state of the art methods obtaining general scene geometry. A limitation of the proposed method is that our approach cannot model overhanging surfaces. How-



Original Depthmap Model Our Heightmap Model

Figure 5. We have compared our method against the 3D models triangulated from the input depthmaps. These depthmap models generally have more details but also have large errors, and missing regions due to occlusion or specularity (*e.g.* windows). Our models cannot capture the overhanging porch roof for example, but they are clean and complete.

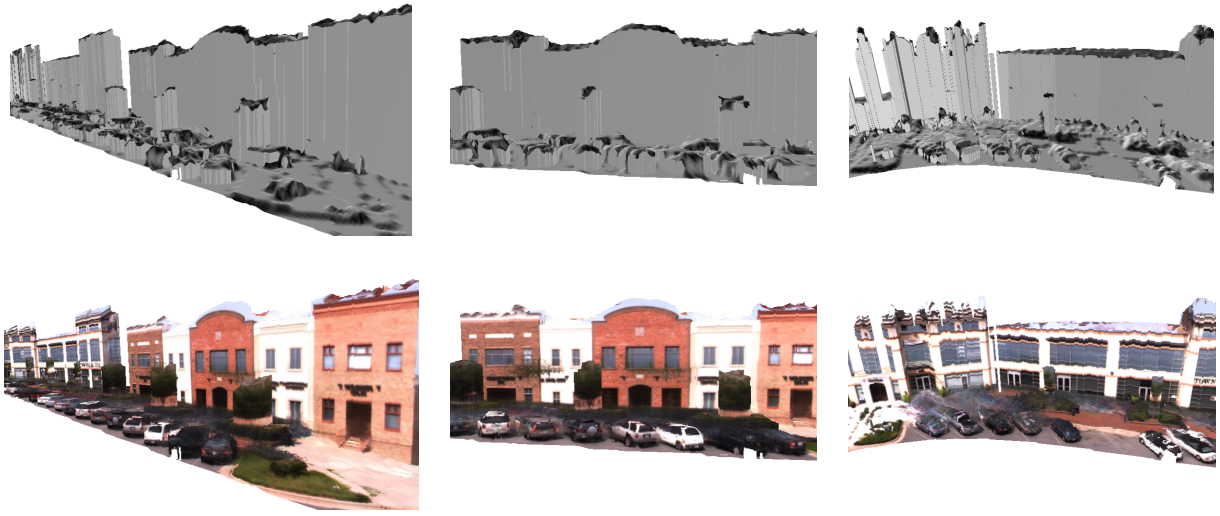


Figure 6. Untextured and textured 3D models produced by our system. This challenging scene features many reflective cars and glass store-front facades.

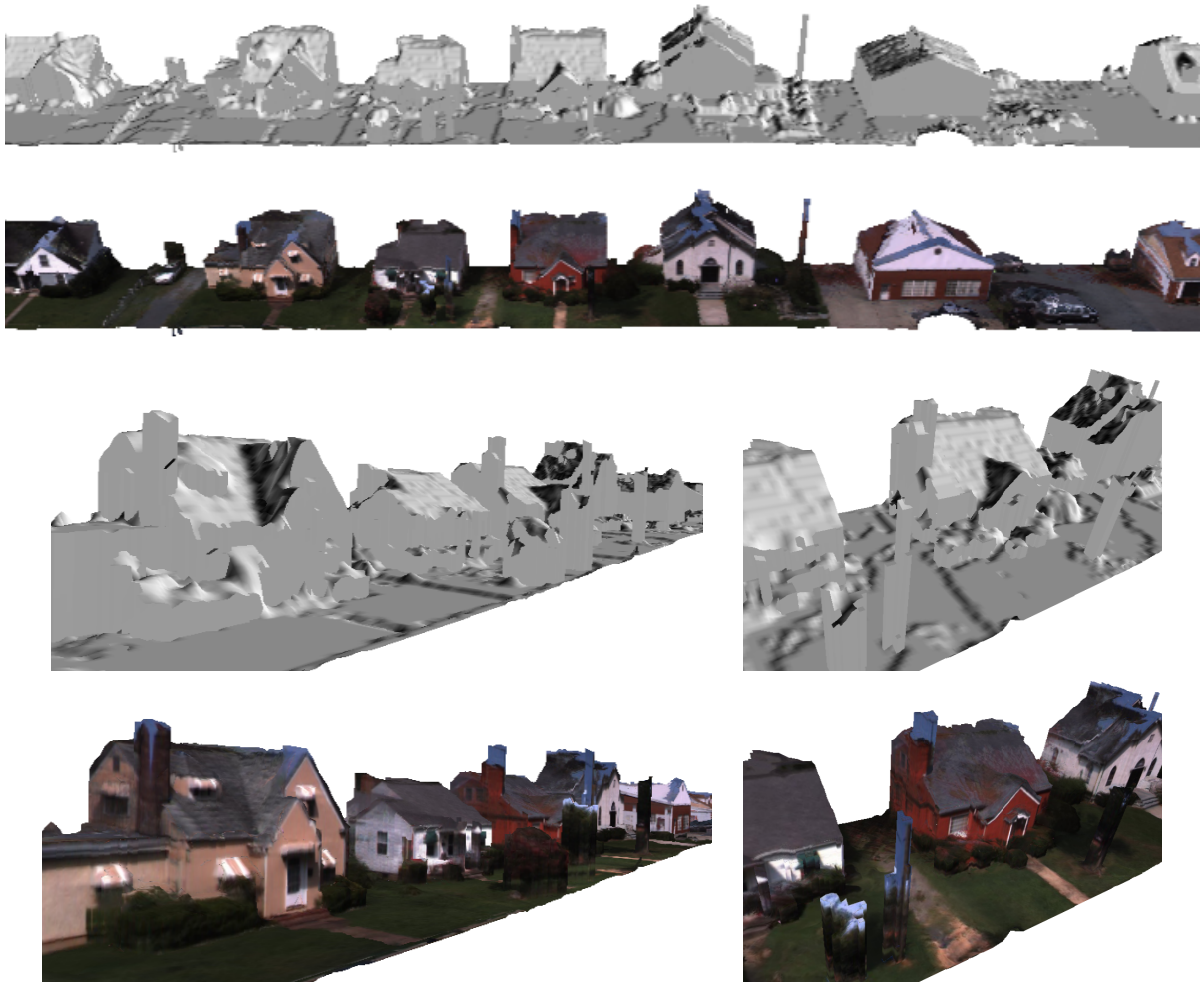


Figure 7. Untextured and textured 3D models of a residential scene reconstructed by our algorithm.

ever, our model is more general than previous approaches that applied simplified geometry [1]. Our approach can for example represent the vegetation like bushes frequently observed in urban scenes. The heightmap model also leads to an efficient and scalable algorithm which produces compact, clean, and complete surface geometry.

Acknowledgements: This work was funded by the David and Lucille Packard Foundation Fellowship and the Department of Energy under Award DE-FG52-08NA28778.

References

[1] N. Cornelis, K. Cornelis, and L. Van Gool. Fast compact city modeling for navigation pre-visualization. In *Computer Vision and Pattern Recognition (CVPR)*, 2006. 2, 7

[2] N. Cornelis, B. Leibe, K. Cornelis, and L. V. Gool. 3d urban scene modeling integrating recognition and reconstruction. *International Journal of Computer Vision (IJCV)*, 2008. 1, 2, 3

[3] C. Früh, S. Jain, and A. Zakohr. Data processing algorithms for generating textured 3d building facade meshes from laser scans and camera images. *International Journal of Computer Vision (IJCV)*, 2005. 1, 2

[4] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Manhatten-world stereo. In *In Proceedings IEEE CVPR*, 2009. 2, 3

[5] S. Kim, D. Gallup, J. Frahm, A. Akbarzadeh, Q. Yang, R. Yang, D. Nister, and M. Pollefeys. Gain adaptive real-time stereo streaming. In *International Conference on Computer Vision Systems (ICVS)*, 2007. 3, 5

[6] Nvidia. Cuda. http://www.nvidia.com/cuda. 5

[7] M. Pollefeys and *et al*. Detailed real-time urban 3d reconstruction from video. *Int. Journal of Computer Vision (IJCV)*, 2008. 1, 2, 3

[8] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision (IJCV)*, 2004. 2, 3

[9] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. Journal of Computer Vision (IJCV)*, 2002. 2

[10] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer Vision and Pattern Recognition (CVPR)*, 2006. 2

[11] S. N. Sinha, D. Steedly, and R. Szeliski. Piecewise planar stereo for image-based rendering. In *In Proceedings IEEE ICCV*, 2009. 2, 3

[12] S. Teller. Automated urban model acquisition: Project rationale and status. In *Image Understanding Workshop*, pages 455–462, 1998. 3

[13] J. Xiao and L. Quan. Image-based street-side city modeling. In *SIGGRAPH Asia*, 2009. 1, 3, 4

[14] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust tv-l1 range image integration. In *International Conference on Computer Vision (ICCV)*, 2007. 3

[15] L. Zebedin, J. Bauer, K. Karner, and H. Bischof. Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In *European Conference on Computer Vision (ECCV)*, 2008. 1