Real-Time Visibility-Based Fusion of Depth Maps

Conference Paper *in* Proceedings / IEEE International Conference on Computer Vision. IEEE International Conference on Computer Vision · January 2007

DOI: 10.1109/ICCV.2007.4408984 · Source: DBLP

CITATIONS
280

READS
1,165

8 authors, including:

Paul Merrell
Google Inc.
17 PUBLICATIONS **1,864** CITATIONS
SEE PROFILE

Jan-Michael Frahm
University of North Carolina at Chapel Hill
145 PUBLICATIONS **9,150** CITATIONS
SEE PROFILE

Marc Pollefeys
ETH Zurich
610 PUBLICATIONS **27,805** CITATIONS
SEE PROFILE

Real-Time Visibility-Based Fusion of Depth Maps

Paul Merrell[1], Amir Akbarzadeh[2], Liang Wang[2], Philippos Mordohai[1], Jan-Michael Frahm[1],
Ruigang Yang[2], David Nistér[2], and Marc Pollefeys[1]
[1]Department of Computer Science
University of North Carolina, Chapel Hill, USA
[2] Center for Visualization and Virtual Environments
University of Kentucky, Lexington, USA

Abstract

We present a viewpoint-based approach for the quick fusion of multiple stereo depth maps. Our method selects depth estimates for each pixel that minimize violations of visibility constraints and thus remove errors and inconsistencies from the depth maps to produce a consistent surface. We advocate a two-stage process in which the first stage generates potentially noisy, overlapping depth maps from a set of calibrated images and the second stage fuses these depth maps to obtain an integrated surface with higher accuracy, suppressed noise, and reduced redundancy. We show that by dividing the processing into two stages we are able to achieve a very high throughput because we are able to use a computationally cheap stereo algorithm and because this architecture is amenable to hardware-accelerated (GPU) implementations. A rigorous formulation based on the notion of stability of a depth estimate is presented first. It aims to determine the validity of a depth estimate by rendering multiple depth maps into the reference view as well as rendering the reference depth map into the other views in order to detect occlusions and free-space violations. We also present an approximate alternative formulation that selects and validates only one hypothesis based on confidence. Both formulations enable us to perform video-based reconstruction at up to 25 frames per second. We show results on the Multi-View Stereo Evaluation benchmark datasets and several outdoors video sequences. Extensive quantitative analysis is performed using an accurately surveyed model of a real building as ground truth.

1. Introduction

The problem of 3D reconstruction from video is a very important topic in computer vision. It has received renewed attention recently due to applications such as Google Earth and Microsoft Virtual Earth which have been introduced for delivering effective visualizations of large scale models based on aerial and satellite imagery. This has sparked an interest in ground-based models as the next logical step in creating a more effective city visualization tool. Visualizations from ground imagery are possible in the form of panoramic mosaics [21, 16] or simple geometric models [2] which require less data to construct, but limit the user's ability to freely navigate the environment. For unconstrained navigation, accurate and detailed 3D models are needed. Video-based 3D reconstruction can deliver this type of content, but only once it is capable of overcoming the challenge of processing massive amounts of imagery. Typical multiple-view reconstruction algorithms that process all images simultaneously are unable to create models such as the one in Fig. 1 which is made from 170,000 frames.

Figure 1. Aerial View and Detailed Views of reconstructed models from a 170,000 frame video

In this paper, we investigate the reconstruction of accurate 3D models from video. We employ a two-stage approach designed to achieve high processing speeds. In the first stage, depth maps are quickly reconstructed from calibrated images. Due to the speed of the algorithm, we expect that the raw stereo depth maps contain many errors and that they do not completely agree with one another. These conflicts and errors are identified and resolved in the fusion stage. In this step, a set of depth maps from neighboring camera positions are combined into a single depth map, the *fused* depth map, for one of the views. Then consecutive fused depth maps are merged together. The final result is a fused surface represented by a mesh and a set of images for texture-mapping. By decoupling the problem into a stereo reconstruction stage followed a fusion stage, we are able to use fast algorithms that can be easily ported to a

programmable Graphics Processing Unit (GPU). Since it is impossible to process thousands of images simultaneously, processing is performed in a sliding window.

Another benefit of the fusion step is that it produces a compact representation of the data since the number of fused depth maps that are outputted is a small fraction of the original number of depth maps. Much of the information in the original depth maps is redundant since many nearby viewpoints observe the same surface. After fusion, the surface is constructed by detecting any overlap between consecutive fused depth maps and merging the overlapping surfaces. A compact representation is especially important for long video sequences since it allows the end-user to visualize very large models.

2. Related Work

Large scale reconstruction systems typically generate partial reconstructions which are then merged. Conflicts and errors in these partial reconstructions are identified and resolved during the merging process. Surface fusion has received considerable attention in the literature mostly for data produced by range finders, where the noise level and the fraction of outliers is typically lower than what is encountered using passive sensors. Multiple-view reconstruction methods based only on images have also been thoroughly investigated [19], but many of them are limited to single objects and can not be applied to large scale scenes due to computation and memory requirements.

Turk and Levoy [22] proposed a method for registering and merging two triangular meshes. They remove any overlapping parts of the meshes, connect the mesh boundaries and then update the positions of the vertices. Soucy and Laurendeau [20] introduced a similar algorithm which first updates the positions of the vertices and then connects them to form the triangular mesh. A different approach was presented by Curless and Levoy [3] who employ a volumetric representation of the space and compute a cumulative weighted distance function from the depth estimates. This signed distance function is an implicit representation of the surface. A volumetric approach that explicitly takes into account boundaries and holes was published by Hilton et al. [8]. Wheeler et al. [23] adapted the method of [3] to only consider potential surfaces in voxels that are supported by some consensus, instead of just one range image, to increase its robustness to outliers. An online algorithm that uses a structured light sensor was introduced by Rusinkiewicz et al. [17]. It can merge partial reconstructions in the form of point clouds in real-time by quantizing the space in voxels and aggregating the information in each voxel. A slower, more accurate algorithm was also described.

Among the first approaches for passive data was that of Fua [4] who adopted a particle based representation. The positions and orientations of the particles are initialized from the depth estimates and modulated according to an image-based cost function and smoothness. Koch et al. [10] first establish binocular pixel correspondences and then propagate them to more images. When a match is consistent with a new camera, the camera is added to the chain that supports the match. The position of the point is updated using the wider baseline, reducing the sensitivity to noise. Narayanan et al. [13] compute depth maps using multi-baseline stereo and merge them to produce viewpoint-based visible surface models. Holes due to occlusion are filled in from nearby depth maps. Hernández et al. [7] compute the probability that a point in a volumetric grid is visible from depth maps and then segment the volume into foreground and background using graph cuts.

Koch et al. [11] also presented a volumetric approach for fusion. Given depth maps for all images, the depth estimates for all pixels are projected in the voxelized 3D space. Each depth estimate votes for a voxel probabilistically and the surfaces are extracted by thresholding. Sato et al. [18] also advocated a volumetric method based on voting. Each depth estimate votes not only for the most likely surface but also for the presence of free space between the camera and the surface. Morency et al. [12] operate on a linked voxel space to rapidly produce triangulations. Information is fused in each voxel while connectivity information is maintained and updated in order to produce the final meshes. Goesele et al. [6] presented a two-stage approach which merges depth maps produced by a simple algorithm. Normalized cross-correlation is computed for each depth estimate between the reference view and several target views. The depth estimates are rejected if the cross-correlation is not large enough for at least two target views. The remaining depth estimates are used for surface reconstruction using the technique of [3].

3. Plane-Sweeping Stereo

In the first of the two processing stages, depth maps are computed from a set of images captured by a moving camera with known pose, using plane-sweeping stereo [1, 5, 24]. Plane-sweeping was chosen primarily because it can be computed efficiently on the GPU, but since the two stages are designed to operate independently, any other stereo matching algorithm could have been used instead.

The depth maps are computed using the plane-sweeping technique described in [5]. Each depth map is computed for the central image in a set of typically 5 to 11 images. At each pixel, several depth hypotheses are tested in the form of planes. For each plane hypothesis, the depth for a pixel is computed by intersecting the ray emanating from the pixel with the hypothesized plane. All images are projected onto the plane, via a set of homographies, and a cost for the hypothesized depth is calculated based on how well the images match on the plane. Here, we use absolute inten-

sity difference as the cost. The set of images is divided in two halves, one preceding and one following the reference image. The sum of the absolute differences between each half of the images and the reference view, projected onto the plane, is calculated in square windows. The minimum of the two sums is the cost of the depth hypothesis [9]. This scheme is effective against occlusions, since in general the visibility of a pixel does not change more than once during an image sequence. The depth of each pixel is estimated to be the depth d_0 with the lowest cost. Each pixel is processed independently which allows non-planar surfaces to be reconstructed.

The depth with the lowest cost may not be the true depth due to noise, occlusion, lack of texture, surfaces that are not aligned with the plane direction, and many other factors. Parts of the image with little texture are especially difficult to accurately reconstruct using stereo. A measurement of the confidence of each depth estimate is important. To estimate the confidence, we use the following heuristic. We assume the cost is perturbed by Gaussian noise. Let $c(\mathbf{x}, d)$ be the matching cost for depth d at pixel $\mathbf{x}$. We wish to estimate the likelihood that the true depth, d_o, does not have the lowest cost after the cost is perturbed. This likelihood is proportional to: $e^{-(c(\mathbf{x},d)-c(\mathbf{x},d_0))^2/\sigma^2}$ for some σ that depends on the strength of the noise. The confidence $C(\mathbf{x})$ is defined as the inverse of the sum of these probabilities for all possible depths:

$$C(\mathbf{x}) = \left(\sum_{d \neq d_0} e^{-(c(\mathbf{x},d)-c(\mathbf{x},d_0))^2/\sigma^2} \right)^{-1} \quad (1)$$

This equation produces a high confidence when the cost has a single sharp minimum. The confidence is low when the cost has a shallow minimum or several low minima.

4. Visibility-Based Depth Map Fusion

Due to the speed of the stereo algorithm, the raw stereo depth maps contain errors and do not completely agree with each other. These conflicts and errors are identified and resolved in the fusion stage. In this step, a set of depth maps from neighboring camera positions are combined into a single fused depth map for one of the views. From the fused depth maps we show how to construct a triangulated surface in Sec. 4.3.

The input to the fusion step is a set of N depth maps denoted by $D_1(\mathbf{x}), D_2(\mathbf{x}), \ldots, D_N(\mathbf{x})$ which record the estimated depth of each pixel of the N images. Each depth map has an associated confidence map labeled $C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_N(\mathbf{x})$ computed according to (1). One of the viewpoints, typically the central one, is selected as the reference viewpoint. We seek a depth estimate for each pixel of the reference view. The current estimate of the 3D point seen at pixel $\mathbf{x}$ of the reference view is called $\hat{F}(\mathbf{x})$. $R_i(\mathbf{X})$ is the distance between the center of projection of viewpoint i and the 3D point $\mathbf{X}$. We define the term $\hat{f}(\mathbf{x}) \equiv R_{ref}(\hat{F}(\mathbf{x}))$ which is the distance of the current depth estimate $\hat{F}(\mathbf{x})$ for the reference camera.

The first step of fusion is to render each depth map into the reference view. When multiple depth values project onto the same pixel, the nearest depth is kept. Let D_i^{ref} be the depth map D_i rendered into the reference view and C_i^{ref} be the confidence map rendered in the reference view. Given a 3D point $\mathbf{X}$, we need a notation to describe the value of the depth map D_i at the location where $\mathbf{X}$ projects into view i. Let $P_i(\mathbf{X})$ be the image coordinates of the 3D point $\mathbf{X}$ projected into view i. To simplify the notation, we define the term $D_i(\mathbf{X}) \equiv D_i(P_i(\mathbf{X}))$. $D_i(\mathbf{X})$ is likely to be different from $R_i(\mathbf{X})$ which is the distance between $\mathbf{X}$ and the camera center.

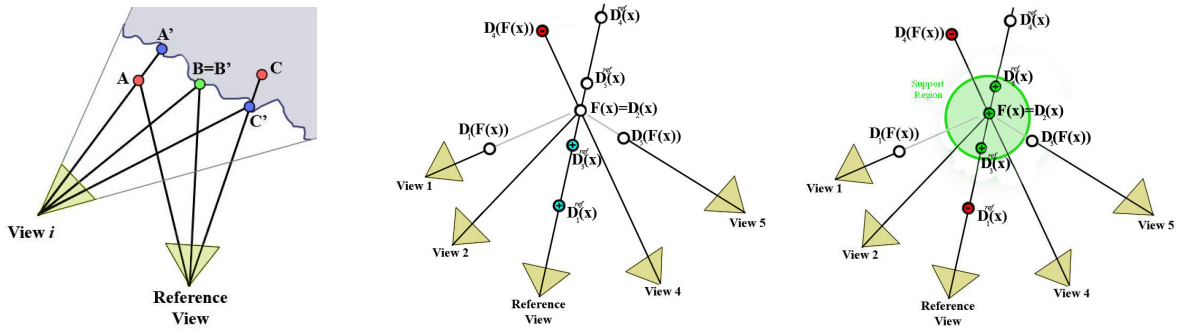
Our approach considers three types of visibility relationships between hypothesized depths in the reference view and computed depths in the other views. These relations are illustrated in Fig. 2(a). The point A' observed in view i is behind the point A observed in the reference view. There is a conflict between the measurement and the hypothesized depth since view i would not be able to observe A' if there truly was a surface at A. We say that A violates the free space of A'. This occurs when $R_i(A) < D_i(A)$.

In Fig. 2(a), B' is in agreement with B since they are in the same location. In practice, we define points B and B' as being in agreement when $\frac{|R_{ref}(B)-R_{ref}(B')|}{R_{ref}(B)} < \epsilon$.

The point C' observed in view i is in front of the point C observed in the reference view. There is a conflict between these two measurements since it would be impossible to observe C if there truly was a surface at C'. We say that C' occludes C. This occurs when $D_i^{ref}(\mathbf{C}') < \hat{f}(\mathbf{C}) = D_{ref}(\mathbf{C})$.

Note that operations for a pixel are not performed on a single ray, but on rays from all cameras. Occlusions are defined on the rays of the reference view, but free space violations are defined on the rays of the other depth maps. The reverse depth relations (such as A behind A' or C in front of C') do not represent visibility conflicts.

The raw stereo depth maps give different estimates of the depth at a given pixel in the reference view. We first present a method that tests each of these estimates and selects the most likely candidate by exhaustively considering all occlusions and free-space constraints. We then present an alternative approach that selects a likely candidate upfront based on the confidence and then verifies that this estimate agrees with most of the remaining data. The type of computations required in both approaches are quite similar. Most of the computation time is spent rendering a depth map seen in one viewpoint into another viewpoint. These computations can be performed efficiently on the GPU.



(a) Visibility relations between points (b) Stability calculation (c) Support estimation

Figure 2. (a) Visibility relations between points. The point A' seen in view i has its free space violated by A seen in the reference view. B' supports B. C seen in the reference view is occluded by C'. (b) Stability Calculation. In this example, there are two occlusions which raise stability and one free-space violations which lowers it. The stability is +1. (c) Support calculation. Three measurements are close to the current estimate and add support to it. Outside the support region, there is one occlusion and one free-space violation which lower the support.

Figure 3. A stereo depth map for a dataset from [19], the fused depth map using stability-based fusion and the final confidence map (black corresponds to maximum confidence).

4.1. Algorithm 1: Stability-Based Fusion

If a depth map occludes a depth hypothesis $\hat{F}(\mathbf{x})$, this indicates that the hypothesis is too far away from the reference view. If the current depth hypothesis violates a free-space constraint, this indicates the hypothesis is too close to the reference view. The stability of a point $S(\mathbf{x})$ is defined as the number of depth maps that occlude $\hat{F}(\mathbf{x})$ minus the number of free-space violations. Stability measures the balance between these two types of visibility violations. A point is stable if the stability is greater than or equal to zero. If the stability is negative, then most of the depth maps indicate that $\hat{F}(\mathbf{x})$ is too close to the camera to be correct. If the stability is positive then at least half of the depth maps indicate that $\hat{F}(\mathbf{x})$ is far enough away from the reference camera. Stability generally increases as the point moves further away from the camera. The final fused depth is selected to be the closest depth to the camera for which stability is non-negative. This depth is not the median depth along the viewing ray since free-space violations are defined on rays that do not come from the reference view. This depth is balanced in the sense that the amount of evidence that indicates it is too close is equal to the amount of evidence that indicates it is too far away.

With this goal in mind, we construct an algorithm to find the closest stable depth. To begin, all of the depth maps are rendered into the reference view. In the example shown in Fig. 2(b), five depth maps are rendered into the reference view. The closest depth is selected as the initial estimate. In the example, the closest depth is $D_1^{ref}(\mathbf{x})$ and so its stability is evaluated first. The point is tested against each depth map to determine if the depth map occludes it or if it violates the depth map's free space. If the depth estimate is found to be unstable, we move onto the next closest depth. Since there are N possible choices, the proper depth estimate is guaranteed to be found after $N-1$ iterations. The total number of depth map renderings is bound by $O(N^2)$. In the example, the closest two depths $D_1^{ref}(\mathbf{x})$ and $D_3^{ref}(\mathbf{x})$ were tested first. Figure 2(b) shows the test being performed on the third closest depth $D_2^{ref}(\mathbf{x})$. A free-space violation and two occlusions are found and thus the stability is positive. In this example, $D_2^{ref}(\mathbf{x})$ is the closest stable depth.

The final step is to compute a confidence value for the estimated depth. The distance to the selected depth $R_i(\hat{F}(\mathbf{x}))$ is compared with the estimate in depth map i given by $D_i(\hat{F}(\mathbf{x}))$. If these values are within ϵ, the depth map supports the final estimate. The confidences of all the estimates that support the selected estimate are added. The resulting fused confidence map is passed on to the mesh construction module. An example of an input depth map and the fused depth map and confidence map can be seen in Fig. 3.

4.2. Algorithm 2: Confidence-Based Fusion

Stability-based fusion tests up to $N-1$ different depth hypotheses. In practice, most of these depth hypotheses are close to one another, since the true surface is likely to be visible and correctly reconstructed in several depth maps. Instead of testing so many depth estimates, an alternative approach is to combine multiple close depth estimates into a single estimate and then perform only one test. Because

there is only one hypothesis to test, there are only $O(N)$ renderings to compute. This approach is typically faster than stability-based fusion which tests $N-1$ hypotheses and computes $O(N^2)$ renderings, but the early commitment may introduce additional errors.

Combining Consistent Estimates Confidence-based fusion also begins by rendering all the depth maps into the reference view. The depth estimate with the highest confidence is selected as the initial estimate for each pixel. At each pixel $\mathbf{x}$, we keep track of two quantities which are updated iteratively: the current depth estimate and its level of support. Let $\hat{f}_0(\mathbf{x})$ and $\hat{C}_0(\mathbf{x})$ be the initial depth estimate and its confidence value. $\hat{f}_k(\mathbf{x})$ and $\hat{C}_k(\mathbf{x})$ are the depth estimate and its support at iteration k, while $\hat{F}(\mathbf{x})$ is the corresponding 3D point.

If another depth map $D_i^{ref}(\mathbf{x})$ produces a depth estimate within ϵ of the initial depth estimate $\hat{f}_0(\mathbf{x})$, it is very likely that the two viewpoints have both correctly reconstructed the same surface. In the example of Fig. 2(c), the estimates $D_3(\hat{F}(\mathbf{x}))$ and $D_5(\hat{F}(\mathbf{x}))$ are close to the initial estimate. These close observations are averaged into a single estimate. Each observation is weighted by its confidence according to the following equations:

$$\hat{f}_{k+1}(\mathbf{x}) = \frac{\hat{f}_k(\mathbf{x})\hat{C}_k(\mathbf{x}) + D_i^{ref}(\mathbf{x})C_i(\mathbf{x})}{\hat{C}_k(\mathbf{x}) + C_i(\mathbf{x})} \tag{2}$$

$$\hat{C}_{k+1}(\mathbf{x}) = \hat{C}_k(\mathbf{x}) + C_i(\mathbf{x}) \tag{3}$$

The result is a combined depth estimate $\hat{f}_k(\mathbf{x})$ at each pixel of the reference image and a support level $\hat{C}_k(\mathbf{x})$ measuring how well the depth maps agree with the depth estimate. The next step is to find how many of the depth maps contradict $\hat{f}_k(\mathbf{x})$ in order to verify its correctness.

Conflict Detection The total amount of support for each depth estimate must be above the threshold C_{thres} or else it is discarded as an outlier and is not processed any further. The remaining points are checked using visibility constraints. Figure 2(c) shows that $D_1(\hat{F}(\mathbf{x}))$ and $D_3(\hat{F}(\mathbf{x}))$ occlude $\hat{F}(\mathbf{x})$. However, $D_3(\hat{F}(\mathbf{x}))$ is close enough (within ϵ) to $\hat{F}(\mathbf{x})$ to be within its support region and so this occlusion does not count against the current estimate. $D_1(\hat{F}(\mathbf{x}))$ is occluding $\hat{F}(\mathbf{x})$ outside the support region and thus contradicts the current estimate. When such an occlusion takes place the support of the current estimate is decreased by:

$$\hat{C}_{k+1}(\mathbf{x}) = \hat{C}_k(\mathbf{x}) - C_i^{ref}(\mathbf{x}) \tag{4}$$

When a free-space violation occurs outside the support region, as occurs with the depth $D_4(\hat{F}(\mathbf{x}))$ in Fig. 2(c), the confidence of the conflicting depth estimate is subtracted from the support according to:

$$\hat{C}_{k+1}(\mathbf{x}) = \hat{C}_k(\mathbf{x}) - C_i(P_i(\hat{F}(\mathbf{x}))) \tag{5}$$

We have now added the confidence of all the depth maps that support the current depth estimate and subtracted the confidence of all those that contradict it. If the support is positive, the majority of the evidence supports the depth estimate and it is kept. If the support is negative, the depth estimate is discarded as an outlier. The fused depth map at this stage contains estimates with high confidence and holes where the estimates have been rejected.

Hole filling After discarding the outliers, there are many holes in the fused depth map. In practice, the depth maps of most real-world scenes are piecewise smooth and we assume that any small missing parts of the depth map are most likely to have a depth close to their neighbors. To fill in the gaps, we find all inliers within a $w \times w$ window centered at the pixel we wish to estimate. If there are enough inliers to make a good estimate, we assign the median of the inliers as the depth of the pixel. If there are only a few neighboring inliers, the depth map is left blank. Essentially, this is a median filter that ignores the outliers. In the final step, a median filter with a smaller window w_s is used to smooth out the inliers.

4.3. Surface Reconstruction

We have presented two algorithms for generating fused depth maps. Consecutive fused depth maps partially overlap one another. It is likely that these overlapping surfaces will not be aligned perfectly. The desired output of our system is a smooth and consistent model of the scene. To this end, consistency between consecutive fused depth maps is enforced in the final model. Each fused depth map is compared with the previous fused depth map as it is being generated. If a new estimate violates the free space of the previous fused depth maps, the new estimate is rejected. If a new depth estimate is within ϵ of the previous fused depth map, the two estimates are merged into one vertex which is generated only once in the output. Thus redundancy is removed along with any gaps in the model where two representations of the same surface are not connected. More than one previous fused depth map should be kept in memory to properly handle surfaces that disappear and become visible again. In most cases, two previous fused depth maps are sufficient.

After duplicate surface representations have been merged, a mesh is constructed taking into account the corresponding confidence map to suppress any remaining outliers. By using the image plane as a reference both for geometry and for appearance, we can construct a triangular mesh very quickly. We employ a multi-resolution quadtree algorithm in order to minimize the number of triangles while maintaining geometric accuracy as in [14]. We use

a top-down approach rather than a bottom-up approach to lower the number of triangles that need to be processed. Starting from a coarse resolution, we form triangles and test if they correspond to nonplanar parts of the depth map, if they bridge depth discontinuities or if points with low confidence (below C_{thres}) are included within them. If any of these events occur, the quad, which is formed out of two adjacent triangles, is subdivided. The process is repeated on the subdivided quads up to the finest resolution.

We use the following simple planarity test proposed in [15] for each vertex of each triangle:

$$\left| \frac{z_{-1} - z_0}{z_{-1}} - \frac{z_0 - z_1}{z_1} \right| < t. \quad (6)$$

Where z_0 is the z-coordinate, in the camera coordinate system, of the vertex being tested and t is a threshold. z_{-1} and z_1 are the z-coordinates of the two neighboring vertices of the current vertex on an image row. (The distance between the corresponding pixels of two neighboring vertices is equal to the size of the quad's edges.) The same test is repeated along an image column. If either the vertical or the horizontal tests fails for any of the vertices of the triangle, the triangle is not part of a planar surface and so the quad is subdivided. For these tests, we have found that 3D coordinates are more effective than disparity values. Since we do not require a manifold mesh and are interested in fast processing speeds, we do not maintain a restricted quad-tree [14].

5. Results

Our methods were tested on videos of urban environments and on the Multi-View Stereo Evaluation dataset (http://vision.middlebury.edu/mview/) [19]. On the urban datasets, the plane-sweeping stereo algorithm used 48 planes and matched 7 images to obtain each stereo depth map. Every 17 frames, 17 stereo depth maps were used to produce a fused depth map. Using these settings and stability-based fusion a processing rate of 23 frames per second can be achieved on a high-end personal computer with an NVidia GeForce 8800 GPU. If confidence-based fusion is used instead, processing speed reaches 25 frames per second. On the Multi-View dataset, the plane-sweeping step used 94 planes and matched five images. Every five frames, 15 stereo depth maps were fused together. On both datasets, the following parameters were used: $\epsilon = 0.05, \sigma = 120, w = 8$ pixels, $w_s = 4$ pixels, and $C_{thres} = 5$. The image sizes are 512×384 for the urban videos and 640×480 for the Multi-View dataset.

To evaluate the ability of our method to reconstruct urban environments, a 3,000 frame video of the exterior of a Firestone store was captured with two cameras to obtain a more complete reconstruction. One of the cameras was pointed horizontally and the other was tilted up 30^o. The videos from each camera were processed separately. The Firestone building was surveyed to an accuracy of 6 mm and the reconstructed model (Fig. 4(a)) was directly compared with the surveyed model (Fig. 4(b) inset). There are several objects such as parked cars that are visible in the video, but were not surveyed. The ground which slopes away from the building also was not surveyed. Objects included in the video that were not surveyed were manually removed from the evaluation. To measure the accuracy of each reconstructed vertex, the distance from the vertex to the nearest triangle of the ground truth model is calculated. The error measurements for each part of a reconstructed model are displayed in Fig. 4(b). We also evaluated the completeness of the reconstruction which measures how much of the building was reconstructed and is defined similar to the completeness measurement in [19]. Sample points are chosen at random on the surface of the ground truth model with a density of 50 sample points per square meter of surface area. The distance from each sample point to the nearest reconstructed point is measured. A visualization of these distances is shown for one of the reconstructions in Fig. 4(c).

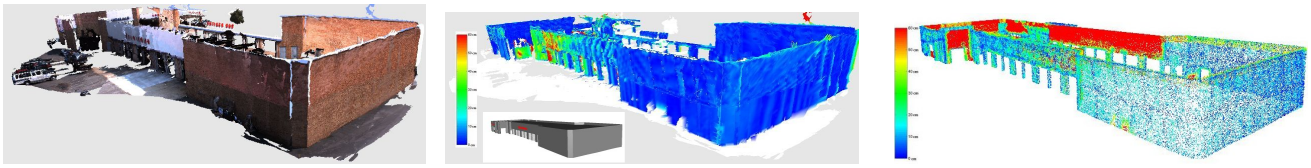
We performed a quantitative evaluation between raw stereo depth maps and the results of the fusion algorithms. For the results labeled as *stereo-reference*, we evaluate the raw depth maps from each of the fusion reference views. For the results labeled *stereo-exhaustive*, we evaluated the depth maps from *all* images as the representation of the scene. We also implemented the real-time algorithm of Rusinkiewicz et al. [17] using a grid of $5cm$ voxels. This algorithm produces a point cloud from which surface reconstruction is not trivial. Table 1 contains the median and mean error values for each method as well as the completeness achieved on the Firestone building. Volumetric methods that handle outliers such as [23] could be applied in our settings, but they would not be real-time.

The results show that the raw stereo depth maps are highly inaccurate and contain many large outliers. Confidence-based and stability-based fusion improve the accuracy of the reconstruction, but lose some completeness, since some of the points removed as outliers might have been near the surface. Depth estimates created in the hole-filling stage of confidence-based fusion are not guaranteed to satisfy visibility constraints, but are typically reasonable resulting in increased completeness at the expense of some accuracy as shown in the results. The real-time technique of Rusinkiewicz et al. [17] does not improve accuracy and does not reduce the number of points in the model effectively without a significant loss of resolution.

Our methods were also used on several videos of urban environments using the same camera configuration and settings. The reconstructed models are shown in Fig. 6 and 7. A very long 170,000 frame video was used to reconstruct a

Fusion Method	Stereo-exhaustive	Stereo-reference	Rusinkiewicz	Confidence	Stability
Median Error(cm)	4.87	4.19	12.9	2.60	2.19
Mean Error(cm)	40.61	39.20	25.6	6.60	4.79
Completeness	94%	83%	84%	73%	66%
Number of Vertices	9,074,377	544,558	4,106,755	539,807	214,940

Table 1. Accuracy and Completeness for different fusion methods using the default parameters (both cameras).



(a) Reconstructed Model using Confidence-based fusion

(b) Accuracy evaluation. White indicates unsurveyed areas. (Inset) Surveyed Model

(c) Completeness of the Firestone building. Red areas mostly correspond to unobserved or untextured areas.

Figure 4. Firestone Building Accuracy and Completeness Evaluation. (b,c) Blue, green and red indicate errors of $0cm$, $30cm$ and $60{+}cm$, respectively. Please view on a color display.

large model shown in Fig. 1. We achieve processing rates of 25 frames per second for typical settings. Even though our approach is viewpoint-based, we are still able to handle multiple depth layers (see Fig. 6, 7, and supplemental video) due to the redundancy in the depth maps.

The two algorithms were also evaluated on the Multi-View Stereo Evaluation benchmark dataset [19]. This evaluation provides metrics on the accuracy and completeness of the reconstruction. The accuracy metric is the distance d such that 90% of the reconstructed surface is within d from the ground truth surface. Completeness is the percentage of ground truth within 1.25 mm of the model. The results of the evaluation on the ring datasets for both objects, as well as the total runtime using our GPU implementation are given in Table 2. The fusion process only for stability-based fusion on the "TempleRing" dataset took 153 seconds implemented on the CPU and 16 seconds when run on the GPU. The fusion process for the confidence-based algorithm on the same dataset took 40 seconds on the CPU and 14 on the GPU. The remaining time was spent reading in the data, constructing silhouettes, and computing depth maps. Images of the reconstructed models are shown in Fig. 5.

	Accuracy	Completeness	Time
TEMPLE			
Stability	0.76 mm	85.2%	22 sec
Confidence	0.83 mm	88.0%	19 sec
DINO			
Stability	0.73 mm	73.1%	28 sec
Confidence	0.84 mm	83.1%	21 sec

Table 2. Quantitative evaluation for the ring datasets of the Multi-view Stereo Evaluation

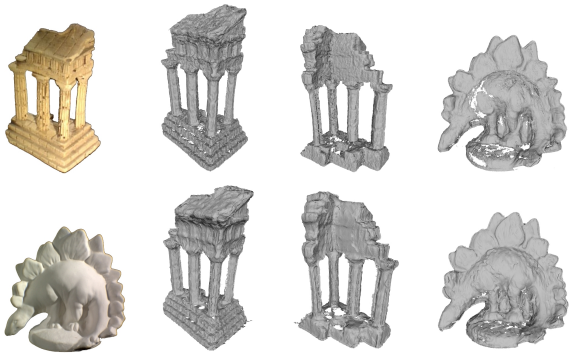


Figure 5. Results on the Multi-View Stereo Evaluation data set. First row shows Stability-based fusion, second row shows Confidence-based fusion. Computing times ranged from 19 to 28 seconds.

Figure 6. Reconstructed buildings using stability-based fusion

6. Conclusion

We have presented a fast, visibility-based approach for reconstructing 3D shape from video in a two-stage process. Two alternative algorithms were presented for fusing the depth maps. In both cases, occlusions and free-space viola-

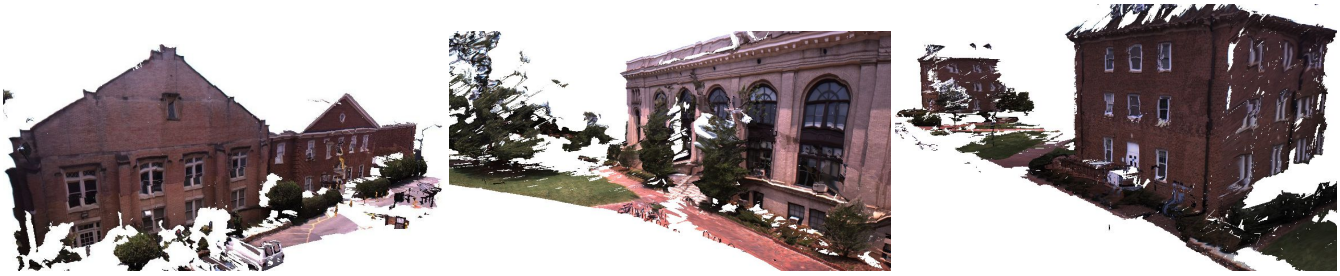


Figure 7. Reconstructed buildings using confidence-based fusion

tions are used to guide the search for a plausible depth for each pixel. Stability-based fusion is more robust and produces slightly more accurate results. Since its complexity scales quadratically with the number of input depth maps, it becomes significantly slower than confidence-based fusion as the number of depth maps increases. Confidence-based fusion is a greedy algorithm since it makes an initial commitment to the most likely candidate. Its linear complexity allows for real-time performance on larger input sets.

Our methods are suitable for large scale reconstructions because they can process large amounts of data in pipeline mode at rates near real time. The accuracy of the reconstruction was evaluated using a surveyed model of a real building and was found to be accurate within a few centimeters. We also participated in the Multi-View Stereo Evaluation obtaining satisfactory results with at least an order of magnitude faster processing than the state of the art. Our future work will focus on improving the visual quality of the models by detecting features such as planar surfaces and straight lines and ensuring they are represented as such.

Acknowledgments: Supported by the DTO under the VACE program and by DARPA under the UrbanScape project. Approved for Public Release, Distribution Unlimited.

Project Website: http://cs.unc.edu/Research/urbanscape/

References

[1] R. Collins. A space-sweep approach to true multi-image matching. In *CVPR*, pages 358–363, 1996.

[2] N. Cornelis, K. Cornelis, and L. Van Gool. Fast compact city modeling for navigation pre-visualization. In *CVPR*, 2006.

[3] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *SIGGRAPH*, 30:303–312, 1996.

[4] P. Fua. From multiple stereo views to multiple 3-d surfaces. *IJCV*, 24(1):19–35, 1997.

[5] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *CVPR*, 2007.

[6] M. Goesele, B. Curless, and S. M. Seitz. Multi-view stereo revisited. In *CVPR*, pages 2402–2409, 2006.

[7] C. Hernández, G. Vogiatzis, and R. Cipolla. Probabilistic visibility for multi-view stereo. In *CVPR*, 2007.

[8] A. Hilton, A. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In *CVPR*, pages 117–126, 1996.

[9] S. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. In *CVPR*, pages 103–110, 2001.

[10] R. Koch, M. Pollefeys, and L. Van Gool. Multi viewpoint stereo from uncalibrated video sequences. In *ECCV*, volume I, pages 55–71, 1998.

[11] R. Koch, M. Pollefeys, and L. Van Gool. Robust calibration and 3d geometric modeling from large collections of uncalibrated images. In *DAGM*, pages 413–420, 1999.

[12] L. Morency, A. Rahimi, and T. Darrell. Fast 3d model acquisition from stereo images. In *3DPVT*, pages 172–176, 2002.

[13] P. Narayanan, P. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *ICCV*, pages 3–10, 1998.

[14] R. Pajarola. Overview of quadtree-based terrain triangulation and visualization. Technical report, 2002.

[15] R. Pajarola, Y. Meng, and M. Sainz. Fast depth-image meshing and warping. Technical report, 2002.

[16] A. Román, G. Garg, and M. Levoy. Interactive design of multi-perspective images for visualizing urban landscapes. In *IEEE Visualization*, pages 537–544, 2004.

[17] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. *SIGGRAPH*, 21(3):438–446, 2002.

[18] T. Sato, M. Kanbara, N. Yokoya, and H. Takemura. Dense 3-d reconstruction of an outdoor scene by hundreds-baseline stereo using a hand-held video camera. *IJCV*, 47(1-3):119–129, 2002.

[19] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, pages 519–528, 2006.

[20] M. Soucy and D. Laurendeau. A general surface approach to the integration of a set of range views. *PAMI*, 17(4):344–358, 1995.

[21] S. Teller, M. Antone, Z. Bodnar, M. Bosse, S. Coorg, M. Jethwa, and N. Master. Calibrated, registered images of an extended urban area. *IJCV*, 53(1):93–107, June 2003.

[22] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *SIGGRAPH*, pages 311–318, 1994.

[23] M. Wheeler, Y. Sato, and K. Ikeuchi. Consensus surfaces for modeling 3d objects from multiple range images. In *ICCV*, pages 917–924, 1998.

[24] R. Yang and M. Pollefeys. A versatile stereo implementation on commodity graphics hardware. *Journal of Real-Time Imaging*, 11(1):7–18, 2005.