Real-Time Plane-Sweeping Stereo with Multiple Sweeping Directions

David Gallup[1], Jan-Michael Frahm[1], Philippos Mordohai[1], Qingxiong Yang[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

Recent research has focused on systems for obtaining automatic 3D reconstructions of urban environments from video acquired at street level. These systems record enormous amounts of video; therefore a key component is a stereo matcher which can process this data at speeds comparable to the recording frame rate. Furthermore, urban environments are unique in that they exhibit mostly planar surfaces. These surfaces, which are often imaged at oblique angles, pose a challenge for many window-based stereo matchers which suffer in the presence of slanted surfaces. We present a multi-view plane-sweep-based stereo algorithm which correctly handles slanted surfaces and runs in real-time using the graphics processing unit (GPU). Our algorithm consists of (1) identifying the scene's principle plane orientations, (2) estimating depth by performing a plane-sweep for each direction, (3) combining the results of each sweep. The latter can optionally be performed using graph cuts. Additionally, by incorporating priors on the locations of planes in the scene, we can increase the quality of the reconstruction and reduce computation time, especially for uniform textureless surfaces. We demonstrate our algorithm on a variety of scenes and show the improved accuracy obtained by accounting for slanted surfaces.

1. Introduction

Reconstructions of buildings in 3D from aerial or satellite imagery has long been a topic of research in computer vision and photogrammetry. The success of such research can be seen in applications such as Google Earth and Microsoft Virtual Earth, which now offer 3D visualizations of several cities. However, such visualizations lack ground-level realism, due mostly to the point of view of the imagery. A different approach is to generate visualizations in the form of panoramas [17, 13] which require less data to be constructed but also limit the user's ability to freely navigate the environment. Recent research has focused on systems for obtaining automatic 3D reconstructions of urban environments from video acquired at street level [16, 14, 7].

Urban environments are unique in that they exhibit mostly planar surfaces. A typical image, for example, may contain a ground plane, and multiple facade planes intersecting at right angles. Many systems aim to reconstruct such imagery using sparse techniques, which examine point or line correspondences to deduce the location of the planes in the scene [18, 7]. Special measures are then taken to account for trees, cars, or other urban clutter which does not fit the planar model. Alternatively, approaches such as [1, 12] employ general 3D from video techniques, using stereo to provide a dense depth estimation, or depthmap, for every frame of the video. In this paper, we favor a hybrid approach where we specialize our algorithm to be particularly efficient on typical urban scenes, while preserving the ability to perform reconstructions of general 3D shape.

Due to the enormous amount of video data required to reconstruct entire cities, the stereo matcher needs to operate at speeds comparable to the recording rate. Furthermore, the ground and facades in urban environments are often imaged at oblique angles. This poses a problem for many window-based stereo algorithms, which typically assume surfaces are parallel to the image plane (fronto-parallel). We present a multi-view plane-sweep-based stereo algorithm which correctly handles slanted surfaces and runs in real-time using the graphics processing unit (GPU). Plane-sweeping is ideal for efficient implementation due to its simplicity and parallelizability [6, 19]. The primary operation of the algorithm, rendering images onto planes, is an operation at which the GPU is particularly adept.

Like other window-based stereo algorithms, plane-sweeping typically assumes surfaces are fronto-parallel. However, slanted surfaces can be correctly handled by sweeping a plane which has a matching surface normal through space. In our approach, we perform multiple plane-sweeps, where each plane-sweep is intended to reconstruct planar surfaces having a particular normal. Our algorithm consists of three steps. First, we identify the surface normals of the ground and facade planes by analysis of 3D points obtained through sparse structure from motion. Second, we perform a plane-sweep for each surface normal, resulting in multiple depth candidates for each pixel in the final depthmap. Third, we select the best depth/normal combination for each pixel using a simple best-cost approach or, optionally, a more advanced three-label graph cut which takes smoothness and integrability into account.

Additionally, we incorporate priors obtained from sparse point correspondences into our depth estimation. This aids in areas with little texture and produces a smoother result. We can also significantly reduce computation time by sweeping planes only in those regions with high prior probability according to the sparse data. Finally, we evaluate our algorithm on several scenes, and demonstrate the accuracy gained over the basic plane-sweeping algorithm.

2. Previous Work

The plane-sweeping algorithm was introduced by Collins [6] as a way to perform matching across multiple images simultaneously without the need for rectification. The approach was originally targeted at the reconstruction of sparse features. Yang and Pollefeys [19] implemented the plane-sweeping stereo algorithm on the GPU, achieving real-time dense depth estimation. Although plane-sweeping is a simple approach, it produces good results for sufficiently textured, unoccluded surfaces. More sophisticated algorithms based on graph cuts [10] and belief propagation [15] offer improved results for textureless areas and occlusion handling, but at a heavy performance penalty.

An approach for reconstructing buildings was described by Werner and Zisserman [18] who use sparse point and line correspondences to discover the ground and facade planes. When there is insufficient information to locate a plane, they sweep a hypothesized plane through space to determine the position which best matches the images. We also use sparse features to obtain information about the scene's planar structure. However, rather than estimating the location of a sparse set of planes, we seek to compute a depth estimate for every pixel in the reference view. Our algorithm is therefore able to reconstruct objects such as trees and cars that do not fit the planar model.

Several stereo algorithms explicitly handle slanted surfaces. Burt *et al.* [5] advocates pre-warping the images to a reference plane, such as the ground, before performing binocular stereo. In this way, they achieve greater accuracy as well faster computation due to the reduced disparity range. Our approach essentially achieves this pre-warping by adjusting our sweeping plane to be parallel to the expected planar surfaces in the scene. Birchfield and Tomasi [2] cast the problem of stereo as image segmentation followed by the estimation of affine transformations between corresponding segments. Processing iterates between segmentation and affine parameter estimation for each segment using graph cuts [4]. The energy function does not favor constant disparity surfaces, but accounts for affine warping and thus slanted surfaces. Ogale and Aloimonos [11] point out that if scene surfaces exhibit horizontal slant, then M pixels on an epipolar line necessarily correspond to N pixels in the other image. Therefore, requiring a one-to-one correspondence per pixel results in labeling $|M - N|$ pixels as occluded. These pixels that are interleaved with matched pixels, however, are visible in both images, just not at integer coordinate positions. An algorithm based on dynamic programming is proposed to obtain correspondences between segments of scanlines rather than pixels.

Zabulis and Daniilidis [20] explicitly addressed non-fronto-parallel surfaces by performing correlation on 3D planes instead of the image plane. Thus, correlation kernels can be aligned with the scene surfaces but the dimensionality of the search space is increased from 1D (disparity) to 3D (disparity and two rotation angles). In this paper we present methods for aligning the correlation windows with the surfaces without exhaustive search for urban scenes. Zabulis *et al.* [21] replaced the planes in space sweeping stereo with spheres. They argue that correlation on spherical sectors along the direction of camera rays is geometrically more accurate since the surfaces where correlation takes place are always orthogonal to the viewing ray.

Recently, Cornelis *et al.* [7] presented a system for real-time city modeling that employs a very simple model for the geometry of the world. Specifically, they assume that the scenes consist of three ruled surfaces: the ground and two facades orthogonal to it. A stereo algorithm that matches vertical lines across two calibrated cameras is used to reconstruct the facades, while objects that are not consistent with the facades or the ground are suppressed. Our algorithm on the other hand is able to handle arbitrary 3D shape. We take advantage of urban scenes, yielding higher quality and speed, but will perform no worse on general scenes.

3. Plane-sweeping Stereo

In this section we outline the basic plane-sweeping algorithm. For more details we refer readers to [19]. Plane-sweeping stereo tests a family of plane hypotheses and records for each pixel in a reference view the best plane as scored by some dissimilarity measure. The algorithm works with any number of cameras, and images need not be rectified. The inputs to the algorithm are M 3D-planes for the depth tests, $N+1$ images at different camera positions (we assume images have been corrected for radial distortion), and their respective camera projection matrices P_k:

$$P_k = K_k[R_k^T \ \ -R_k^T C_k] \text{ with } k = 1, \ldots, N, \qquad (1)$$

where K_k is the camera calibration matrix, and R_k, C_k are the rotation and translation of camera P_k with respect to the reference camera P_{ref}. The reference camera is assumed to be the origin of the coordinate system. Accordingly its projection matrix is $P_{ref} = K_{ref} \begin{bmatrix} I_{3\times3} & 0 \end{bmatrix}$. The family of depth planes Π_m with $m = 1, \ldots, M$ is defined in the coordinate frame of the reference view by:

$$\Pi_m = \begin{bmatrix} n_m^T & -d_m \end{bmatrix} \text{ for } m = 1, \ldots, M \qquad (2)$$

where n_m is the unit length normal of the plane and d_m is the distance of the plane to the origin namely the center of the reference camera. (For a fronto-parallel sweep, $n_m^T = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$.) The depths d_m of the planes Π_m fall within the interval $[d_{near}, d_{far}]$. It is best to space the planes to account for the sampling (pixels) in the images. This is discussed in detail in Section 4.2.

In order to test the plane hypothesis Π_m for a given pixel (x, y) in the reference view I_{ref}, the pixel is projected into the other images $k = 1, \ldots, N$. The mapping from the image plane of the reference camera P_{ref} to the image plane of the camera P_k is a planar mapping, and can therefore be described by the homography H_{Π_m, P_k} induced by the plane Π_m. This homography H_{Π_m, P_k} is:

$$H_{\Pi_m, P_k} = K_k \left(R_k^T + \frac{R_k^T C_k n_m{}^T}{d_m} \right) K_{ref}^{-1}. \qquad (3)$$

The location (x_k, y_k) in image I_k of the mapped pixel (x, y) of the reference view is computed by:

$$\begin{aligned} \begin{bmatrix} \tilde{x} & \tilde{y} & \tilde{w} \end{bmatrix}^T &= H_{\Pi_m, P_k} \begin{bmatrix} x & y & 1 \end{bmatrix}^T \\ x_k = \tilde{x}/\tilde{w}, & \qquad y_k = \tilde{y}/\tilde{w}. \end{aligned} \qquad (4)$$

If the plane intersects the surface projected to pixel (x, y) in the reference view, the colors of $I_k(x_k, y_k)$ and $I_{ref}(x, y)$ should be similar assuming Lambertian surfaces.

We use the absolute difference of intensities as the dissimilarity measure. The measurement at one pixel is in general very sensitive to noise. To reduce the sensitivity several measurements in the neighborhood of the pixel are combined. Typically this is done by summing the measurements in a rectangular window W centered at the pixel (x, y). A cost metric can then be defined as a function of the pixel location (x, y) in the reference view, and the plane Π_m by

$$\begin{aligned} C(x, y, \Pi_k) = & \sum_{k=0}^{N-1} \sum_{(i,j)\in W} |I_{ref}(x - i, y - j) \\ & - \beta_k^{ref} I_k^\star(x - i, y - j)|, \end{aligned} \qquad (5)$$

where $I_k^\star$ is the image I_k warped by the homography H_{Π_m, P_k} and β_k^{ref} corresponds to the gain ratio between image k and the reference. The ability to compensate for gain changes is essential to handle outdoor scenes where the brightness range often far exceeds the dynamic range of the camera. In our GPU implementation the gain compensation does not have a measurable effect on speed.

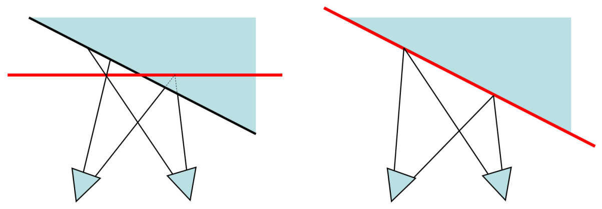
Figure 1. Implications of cost aggregation over a window. *Left*: Slanted surfaces with fronto-parallel plane-sweeping. Not all points over the window are in correspondence. *Right*: Surface-aligned sweeping plane to handle slanted surfaces correctly.

3.1. Extracting the Depthmap from the Cost

Once the cost function for all pixels has been computed the depth map may be extracted. The first step is to select the best plane at each pixel in the reference view. This may simply be the plane of minimum cost, also called best-cost or winner-takes-all, defined as follows

$$\tilde{\Pi}(x, y) = \underset{\Pi_m}{\operatorname{argmin}}\, C(x, y, \Pi_m). \qquad (6)$$

For a given plane Π_m at pixel (x, y), the depth can be computed by finding the intersection of Π_m and the ray through the pixel's center. This is given by

$$Z_m(x, y) = \frac{-d_m}{\begin{bmatrix} x & y & 1 \end{bmatrix} K_{ref}^{-T} n_m}. \qquad (7)$$

More sophisticated approaches based on global optimization select $\tilde{\Pi}$ that minimizes C but also enforces smoothness between neighboring pixels. Such methods give improved results but are computationally expensive.

We adopt a computationally less expensive solution proposed by Kang *et al.* [9]. For each pixel we compute the cost for each plane using the left and right subset of the cameras and select the minimum. This scheme is very effective against occlusions, since typically the visibility of a pixel changes at most once in a sequence of images.

3.2. Implications of Cost Aggregation

To minimize C at a given pixel (x, y), the plane $\Pi_m = [n_m^T \quad d_m]$ should intersect not only the surface imaged at (x, y), but at all the pixels in the neighborhood window W centered at (x, y) as well. Assuming a locally planar surface, this is accomplished when n_m aligns with the surface normal. Misalignment of n_m and the surface normal potentially leads to errors in the computed depth map, depending on the size of the window, the degree of misalignment, and the surface texture (see Figure 1).

Plane-sweeping stereo, as well as many other stereo algorithms, traditionally approximate the surfaces in the world by assuming that they are piece-wise fronto-parallel. However, many environments, such as cities, contain surfaces which can appear at highly oblique angles. For these surfaces the neighboring pixels cannot be mapped with the same fronto-parallel plane. Accordingly, the cost aggregation within a window around the pixel of interest is perturbed by artifacts due to perspective distortion. To overcome this limitation our approach samples not only the depth of the plane hypothesis it also samples their orientation. This is explained in more detail in the Section 4.

4. Multiple Sweeping Directions

We extend plane-sweeping stereo to account for non-fronto-parallel surfaces in the scene. A trivial extension would be to sample the hemisphere of all visible surface normals and sweep planes in each direction. This would lead to a large number of plane hypotheses that need to be tested, as in [20]. We propose a more efficient approach to keep the algorithm in real-time that performs multiple plane sweeps where the sweeping directions are aligned to the expected surface normals of the scene. This results in multiple depthmaps which we then combine using best-cost or, optionally, a graph cut method.

4.1. Identifying Sweeping Directions

Instead of exhaustively sampling the set of potential surface orientations, we can identify a much smaller set of likely surface normals either by application-specific heuristics or by examining the scene's sparse structure.

In many applications, images are captured by video or still cameras which are either hand-held or mounted on a land-based vehicle. Camera motion can be recovered either by structure from motion or from GPS/INS sensors. The motion of such cameras is generally constrained to be parallel to the ground plane, especially for vehicle-mounted, but typically also for hand-held cameras.

Additionally, a scene's planar structure can be determined by sparse features such as lines and points. This is especially true in urban environments where by examining lines in a single image, vanishing points can be recovered which in turn can be combined to give estimates of plane normals. In applications that use structure from motion, 3D point or line features are recovered as well as the camera poses. These features are available, but many algorithms do not utilize them. Many techniques have been explored to recover planar surfaces from point and line correspondences and vanishing points [18, 3, 14, 8].

We present an effective technique for recovering planar structure in urban environments using 3D point features obtained from structure from motion. We first find the vertical direction or gravity vector. This is either given by an INS system or can be computed from vanishing points. Since most facades are vertical, the vanishing point corresponding to the gravity vector is quite prominent in urban scenes. By assuming the ground plane has zero slope in the direction perpendicular to the computed camera motion, we can obtain a good estimate for the ground plane normal as

$$G = \frac{(V \times M) \times M}{\|(V \times M) \times M\|} \tag{8}$$

where V is the gravity vector, and M is the camera motion direction. Note that obtaining the ground normal is particularly important, since the ground is imaged at a small angle.

To compute the facade normals, we assume that they are perpendicular to the gravity vector, and are therefore determined by a rotation about the gravity vector. By assuming the facades are orthogonal to each other, only one rotation determines the facade normals. We recover the remaining rotation of the facades as follows. We first compute the orthogonal projection of each 3D point in the direction of gravity to obtain a set of 2D points. Note that 3D points on a common vertical facade will project to a line. We then evenly sample the space of in plane rotations between 0 and 90 degrees, and then test each rotation. For each rotation $R = \left[\begin{array}{cc} u & v \end{array}\right]^T$, we rotate the set of 2D points, and construct two histograms H_u and H_v. Each bin in H_u (resp. H_v) counts the number of points with a similar u (resp. v) component. We then compute the entropy of each histogram, and finally select the rotation which has the lowest sum of entropies. As shown in Figure 2, entropy will be minimized when points are aligned in directions u and v.

4.2. Plane Selection

Once the sweeping directions have been computed, we generate a family of planes for each. Referring to equation (2), each family is parameterized by the distance of the plane to the origin d_m. The range $[d_{near}, d_{far}]$ can be determined either by examining the points obtained from structure from motion or by applying useful heuristics. For example, in outdoor environments, it is usually not useful for the ground plane family to extend above the camera center. The spacing of the planes in the range can be uniform, as in [20]. However, it is best to place the planes to account

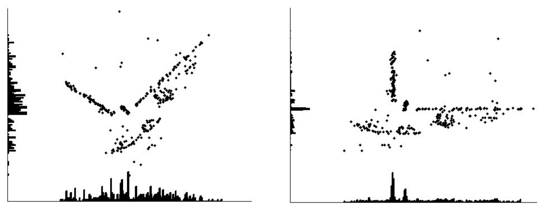
(a) Arbitrary rotation (b) Minimum entropy

Figure 2. Minimum entropy direction optimization. The points on the facades are obtained from structure from motion and projected in the direction of gravity. The points are then rotated into the basis formed by u and v and histograms are generated. The histograms of (a) have more entropy than those of (b). (b) corresponds to the correct surface normals.

for image sampling (pixels). Ideally, when comparing the respective image warpings induced by consecutive planes, the amount of pixel motion should be less than or equal to one. This is particularly important when matching surfaces that exhibit high-frequency texture.

We define the disparity change between two planes Π_m and Π_{m+1} to be the maximum displacement over all pixels in all images.

$$\Delta D(\Pi_m, \Pi_{m+1}) = \max_{k=1,\ldots,N} \max_{(x,y)\in I_k} \sqrt{(x_k^m - x_k^{m+1})^2 + (y_k^m - y_k^{m+1})^2} \quad (9)$$

where (x_k^m, y_k^m) (resp. (x_k^{m+1}, y_k^{m+1})) are obtained by applying the homography H_{Π_m,P_k} (resp. H_{Π_{m+1},P_k}) as in equation (4). To avoid orientation inversions we avoid using planes that intersect the convex hull of the camera centers. In general it is then sufficient to measure the displacements only in the cameras most distant from the reference view. Furthermore it is not necessary to compute the displacement for every pixel. The greatest displacement will occur at the boundaries of the image. Thus, to compute the disparity, we warp the polygon defined by the boundaries of image I_k into the reference view by applying the planar homography H_{Π_m,P_k}. We then clip the polygon in the reference view, and compute the displacement of the vertices of the clipped polygon. As only planes that do not intersect the convex hull of the camera center are used, the polygon warped with $H_{\Pi_{m+1},P_k} H^{-1}_{\Pi_m,P_k}$ is guaranteed to remain convex, and thus the maximal disparity is bound by the maximum displacement of its vertices. The family of planes is then constructed so that the disparity change of consecutive planes is less than or equal to one.

4.3. Incorporating Plane Priors

The minimum-entropy histograms computed in Section 4.1 also indicate the location of the facades. They can be used as a prior in a Bayesian formulation when selecting $\tilde{\Pi}$. The posterior probability of a plane Π_m at pixel (x, y) is

$$P(\Pi_m|C(x,y)) = \frac{P(C(x,y)|\Pi_m)P(\Pi_m)}{P(C(x,y))} \quad (10)$$

where $P(\Pi_m)$ is the prior probability of the surface being located at plane Π_m, and $P(C(x, y)|\Pi_m)$ indicates the likelihood of the correct plane having matching cost $C(x, y)$. $P(C(x, y))$ is the marginal likelihood of the cost. The prior is obtained by sampling the normalized histogram at the location of the plane. For a plane Π_m chosen from sweeping direction u, the location in the histogram H_u is given by the depth component of the plane d_m. The prior is

$$P(\Pi_m) = \frac{H_u(d_m)}{\sum_i H_u(i)}. \quad (11)$$

The cost likelihood depends on image noise, camera pose error, alignment of the plane normal n_m and the surface normal, as well as on the surface texture. This is extremely difficult to model correctly. Instead we choose an exponential distribution:

$$P(C(x,y)|\Pi_m) = e^{\frac{-C(x,y)}{\sigma}} \quad (12)$$

where σ is determined empirically. The exponential is self-similar, and so it makes no assumptions about the minimum matching cost, which is often difficult to predetermine.

Since we are only interested in the maximum likelihood solution we ignore $P(C(x, y))$ and modify the plane selection equation (6) as follows:

$$\tilde{\Pi}(x,y) = \underset{\Pi_m}{\operatorname{argmax}}\, e^{\frac{-C(x,y)}{\sigma}} P(\Pi_m). \quad (13)$$

Maximizing this likelihood is equivalent to minimizing the negative logarithm of the likelihood. Therefore

$$\begin{aligned} \tilde{\Pi}(x,y) &= \underset{\Pi_m}{\operatorname{argmin}} -\log e^{\frac{-C(x,y)}{\sigma}} P(\Pi_m) \\ &= \underset{\Pi_m}{\operatorname{argmin}} \{C(x,y) - \sigma \log P(\Pi_m)\}. \quad (14) \end{aligned}$$

Surfaces with little or no texture will exhibit a low matching cost over a range of planes, the minimum of which may be determined more by noise than by true correspondence. The prior distribution for the depth $P(\Pi_m)$ helps to eliminate such ambiguities and produce a smoother surface. The implementation of equation (14) comes at little additional cost and contributes significantly to the results.

We can also use the prior to significantly reduce our computation time by not testing plane hypotheses with a low prior probability. Typically a scene requires hundreds of planes for each sweeping direction to adequately sample the disparity range. While our algorithm is able to compute this many plane hypotheses at several Hz, we have found that we can obtain quality reconstructions almost an order of magnitude faster by testing only a few dozen of planes. The selected planes are those with the highest prior probability. This is only effective by sweeping in multiple directions. For example, if the sweeping direction were not aligned to the ground and were instead fronto-parallel, it would require many plane hypotheses to reconstruct the ground. However, having determined the ground's surface normal and predicted its location from the prior, we can reconstruct it with only a few plane hypotheses.

4.4. Selecting Depths from Multiple Sweeps

Once the plane sweeps have been performed and the best-cost solutions are selected for each sweep, the remaining task is to determine which sweeping direction at each pixel is correct. Selecting the best-cost solution already

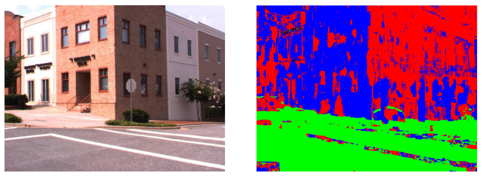

Figure 3. Best-cost direction selection. The labeling on the right indicates the selected sweeping direction. Ground points are labeled green, and facade points are labeled red or blue. The changes in surface normal (blue amidst a sea of red and vice versa) are clearly errors and suggest further optimization is possible.

produces very good results, and most pixels are assigned the correct surface normal. However, it is evident from Figure 3 that due to noise some pixels are assigned incorrect normals. The immediate observation is that many of the errors are small regions of erroneous pixels embedded in a sea of correctly assigned pixels. This suggests minimizing an energy function which penalizes for abrupt changes of the surface normal within a small neighborhood. Also, the correct depths should minimizes the distance between the depths of neighboring pixels. We formulate an energy which encodes the matching cost, surface normal smoothness, and depth smoothness, and minimize it using graph cuts [4].

Typically, graph cut stereo algorithms compute the solution over a range of depth values. For the scenes in which we are interested, this would require hundreds of labels to account for the disparity range. Our energy function can be minimized efficiently since the solution is chosen from typically only a handful of sweeping directions.

Although our algorithm can be generalized to any number of sweeping directions, for simplicity the graph cut is defined in terms of three labels but can be simply extended to multiple labels. We define the set of labels $L = \{l_g, l_{f_1}, l_{f_2}\}$, with one label for each sweeping direction. Each direction also has an associated surface normal n_l and best cost image

$$\tilde{C}_l(x, y) = C\left(x, y, \tilde{\Pi}_l(x, y)\right) - \sigma \log P(\tilde{\Pi}_l(x, y)). \quad (15)$$

We define the energy function:

$$E = \sum_{(x,y)\in I} E_{data} + \lambda_1 \sum_{(x,y),(x',y')\in N} E_{smooth} + \lambda_2 \sum_{(x,y)\in I} E_{int} \quad (16)$$

where λ_1 and λ_2 adjust the relative magnitude for each penalty and $(x, y), (x', y') \in N$ indicates that (x, y) and (x', y') are 4-neighbors. The term E_{data} simply refers to the matching cost of a particular label.

$$E_{data}(x, y) = \tilde{C}_l(x, y) \quad (17)$$

The term E_{smooth} penalizes neighboring pixels for having different surface normals according to the Potts model [4].

$$E_{smooth}(x, y) = \delta(l \neq l') \quad (18)$$

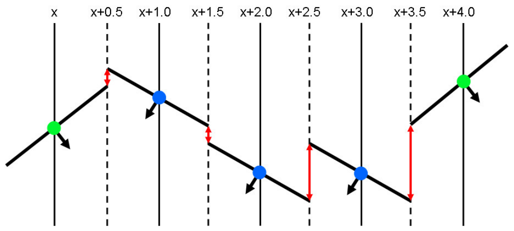

Figure 4. Integrability penalty for horizontal neighbors. Using the surface normals given by the labeling, the surfaces are extended to the midpoint $(x + 0.5)$. The penalty is then defined as a function of the distance between the surfaces along the line $x + 0.5$.

where $\delta(\cdot)$ is 1 when its argument is true and 0 otherwise.

The integrability penalty E_{int} penalizes for depth discontinuities in the surface. In other stereo algorithms, this is typically defined as the absolute or squared distance between the depths of neighboring pixels. In our algorithm not only do we have the depth at each pixel, but we have the surface normal as well. We can therefore approximate the surface at each pixel as a planar patch. For neighboring pixels, rather than compare the depths at the pixel centers, we extrapolate each surface to the midpoint between the two pixels and compare the depths of the surfaces at this point.

Without loss of generality we define the integrability penalty for horizontal neighbors (x, y) and $(x + 1, y)$ with labels l and l'. Let the plane at each pixel be $\tilde{\Pi}_l$ and $\tilde{\Pi}_{l'}$. We find the intersection point of each plane and the ray passing through the pixel $(x + 0.5, y)$ as in equation (7). The integrability penalty is then defined as

$$E_{int}(x, y) = \min(|Z_l(x+0.5, y) - Z_{l'}(x'-0.5, y')|, E_{int}^{max}) \quad (19)$$

where $Z_l(x+0.5, y)$ is the intersection of the ray defined by pixel $(x+0.5, y)$ and $\tilde{\Pi}_l$, and similarly $Z_{l'}(x+0.5, y)$ is the intersection with $\tilde{\Pi}_{l'}$. This assumes the pixel with label l is to the left of the pixel with label l'. The penalty saturates at value E_{int}^{max} so as not to overly penalize true discontinuities in the surface. The integrability penalty can be defined similarly for vertical neighbors. Figure 4 illustrates the integrability penalty. This penalty has the important property that it incurs no penalty for slanted surfaces, which do not have constant depth, as long as the depth gradient coincides with the estimated surface normal.

5. Results

We first demonstrate our algorithm on several video sequences which were captured by a camera mounted on a vehicle moving through streets in an urban environment. The vehicle is equipped with a GPS/INS system. We compute the three sweeping directions as described in Section 4.1, and then compute depthmaps from 11 grayscale images with 512x384 resolution. We processed the data on a PC with an NVIDIA GeForce 8800 GTX graphics card.

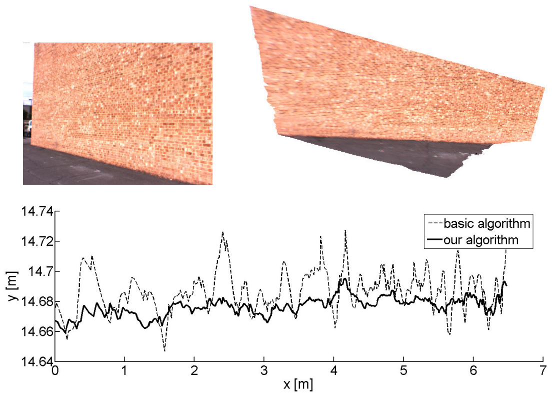

Figure 5. Comparison of the basic fronto-parallel plane sweep and sweeping in multiple directions. *Top left*: the original viewpoint. *Top right*: The depthmap triangulated into a polygonal mesh viewed from above. *Bottom*: A cross-section of the surface. Note the scale on the axes. We measured the standard deviation of the surface from the best-fit line. This was 1.31 cm for the fronto-parallel sweep and 0.61 cm for our algorithm.

In the first experiment, we compare our algorithm with the basic fronto-parallel sweep. The scene is a flat brick wall which is viewed obliquely. We reconstruct the scene using 144 plane hypotheses for both algorithms. For our algorithm we selected the depths from the multiple sweeping directions using best-cost, and achieved a processing rate of at 6.33 Hz. Figure 5 compares a scanline of the depthmaps resulting from the fronto-parallel sweep algorithm and our algorithm. The surface from our algorithm is much smoother and better approximates the planar facade. Note here that for both algorithms we compute sub-pixel matches by parabolic fitting of the best cost. Despite this fact, the fronto-parallel sweep is unable to compute a smooth surface. Since only a percentage of the points in the aggregation window actually have the correct depth, the matching costs to which the polynomial is fit are less indicative of the depth of the surface.

In Figure 6 we demonstrate the graph-cut-based depth selection from Section 4.4. Selecting the best-cost sweeping directions already produces a surface normal labeling which is quite accurate. However, small errors remain. The graph cut is able to eliminate the incorrectly labeled regions and straighten out the interface between the two facades. The cross-sections in Figure 6 shows that by selecting the correct surface normals labels the computed depthmap is improved. This graph cut, although unsuitable for real-time, is quite efficient and takes less than 2 seconds to compute.

In Figure 7, we demonstrate the ability to compute accurate depthmaps with only a small number of plane hypotheses. By testing only the planes with highest prior probability, we produced quality reconstructions with just 48 plane hypotheses per frame. This increases the speed of our algorithm to 33.7 Hz. Although we assume the presence of planes in the scene, our algorithm is a general stereo matcher, and we are still able to reconstruct non-planar objects such as the bushes.

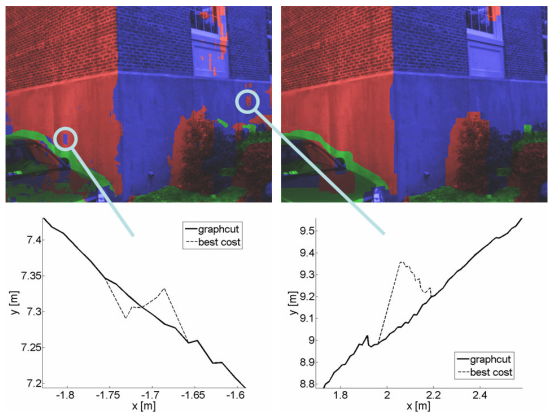

Figure 6. *Top*: The best-cost sweeping direction labeling (left) and the labeling after the graph cut (right) Several mislabeled pixels are corrected by the graph cut. *Bottom*: The improvement in accuracy after the graph cut. This shows the incorrectly labeled pixels were indeed errors in depth. **This figure is best viewed in color.**

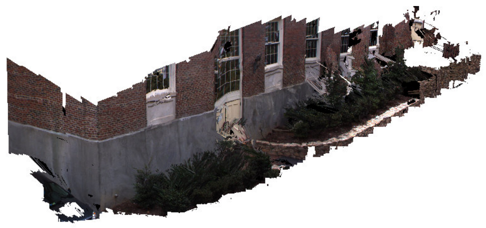

Figure 7. The reconstruction produced by our algorithm from several hundred frames of video. By testing only plane hypotheses with high prior probability, the reconstruction was achieved with only 48 plane hypotheses per frame at a speed of 20.0 Hz.

Finally, we present the reconstruction of a scene captured by a hand-held camera. We computed the direction of the gravity vector from vertical vanishing points and computed the sweeping directions as described in Section 4.1. The reconstruction of the scene is shown in Figure 8. By rendering the scene with a synthetic light source we can see that without aligning the sweeping plane to the surface normals, the reconstructed surface is quite bumpy. Our algorithm produces a reconstruction which is much smoother.

6. Conclusion

We have presented a real-time multi-view stereo algorithm based on plane-sweeping which correctly handles slanted surfaces. It is very effective in real-world applications, such as video-based reconstruction of large urban areas. This is due to the algorithm being inherently fast and to the proper selection of parameters to achieve high quality reconstruction with a limited number of plane hypotheses. We utilize sparse point correspondences, which are avail-

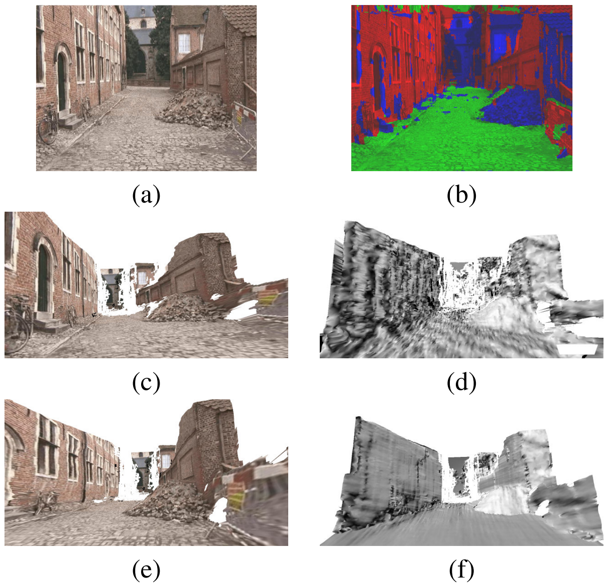

Figure 8. *(a)*: Frame of video captured by a handheld camera. *(b)*: The best-cost labeling of sweeping direction. *(c) and (e)*: Novel views of the reconstructed scene. *(d) and (f)*: The reconstructed scene rendered as a polygonal mesh with a light source. (d) is from the basic plane sweeping algorithm, and (f) is from our algorithm. The lighting demonstrates a much smoother surface is obtained by aligning the sweeping plane to the surface normals. **This figure is best viewed in color.**

able from the structure from motion estimation, to obtain an estimate of the normals of the planar surfaces in the scene. Then we perform plane-sweeps along multiple directions and combine the results according to best-cost or, optionally, a simple three-label graph cut. Furthermore, by incorporating plane priors given by the sparse point correspondences, we can reduce computation time and noise in ambiguous parts of the scene. Comparisons with ground truth show a clear improvement over the basic fronto-parallel plane-sweeping algorithm.

Acknowledgements: Partially Supported by the DTO under the VACE program. Partially supported by DARPA under the UrbanScape project, lead by Geo-Spatial Technologies Information Division of SAIC. Approved for Public Release, Distribution Unlimited.

References

[1] A. Akbarzadeh *et al*. Towards urban 3d reconstruction from video. In *Int. Symp. on 3D Data, Processing, Visualization and Transmission*, 2006.

[2] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *Int. Conf. on Computer Vision*, pages 489–495, 1999.

[3] M. Bosse, R. Rikoski, J. Leonard, and S. Teller. Vanishing points and 3d lines from omnidirectional video. *The Visual Computer*, 19(6):417–430, 2003.

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[5] P. Burt, L. Wixson, and G. Salgian. Electronically directed ”focal” stereo. In *Int. Conf. on Computer Vision*, pages 94–101. IEEE Computer Society, 1995.

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

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

[8] D. Hoiem, A. Efros, and M. Hebert. Putting objects in perspective. In *Proc. of Computer Vision and Pattern Recognition*, pages 2137–2144, 2006.

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

[10] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions via graph cuts. In *Int. Conf. on Computer Vision*, pages 508–515, 2001.

[11] A. Ogale and Y. Aloimonos. Stereo correspondence with slanted surfaces: Critical implications of horizontal slant. In *Proc. of CVPR*, pages I: 568–573, 2004.

[12] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a handheld camera. *Int. Journ. of Computer Vision*, 59:207–232, 2004.

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

[14] G. Schindler and F. Dellaert. Atlanta world: an expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In *Proc. of CVPR*, pages 203–209, 2004.

[15] J. Sun, Y. Li, S. Kang, and H. Shum. Symmetric stereo matching for occlusion handling. In *Proc. of CVPR*, pages II: 399–406, 2005.

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

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

[18] T. Werner and A. Zisserman. New techniques for automated architectural reconstruction from photographs. In *European Conf. on Computer Vision*, pages 541–555, 2002.

[19] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *Proc. of CVPR*, pages I: 211–217, 2003.

[20] X. Zabulis and K. Daniilidis. Multi-camera reconstruction based on surface normal estimation and best viewpoint selection. In *3DPVT*, 2004.

[21] X. Zabulis, G. Kordelas, K. Mueller, and A. Smolic. Increasing the accuracy of the space-sweeping approach to stereo reconstruction, using spherical backprojection surfaces. In *Int. Conf. on Image Processing*, 2006.