Data Processing Algorithms for Generating Textured 3D Building Façade Meshes From Laser Scans and Camera Images

Christian Früh and Avideh Zakhor

Video and Image Processing Lab
Dept. of Electrical Engineering and Computer Sciences
University of California, Berkeley

In this paper, we develop a set of data processing algorithms for generating textured façade meshes of cities from a series of vertical 2D surface scans, obtained by a laser scanner while driving on public roads under normal traffic conditions. These processing steps are needed to cope with imperfections and non-idealities inherent in laser scanning systems such as occlusions and reflections from glass surfaces. The driven path is cut into easy-to-handle quasi-linear segments with approximately straight driving direction. The vertical scans in each segment are recorded in a sequential topological order, and are transformed into a depth image; if available, additional 3D data from other modalities such as stereo vision is also taken into consideration. Dominant building structures are detected in the depth images, and points are classified into foreground and background layers. Large holes in the background layer, caused by occlusion from foreground layer objects, are filled in by planar or horizontal interpolation. The depth image is further processed by removing isolated points and filling remaining small holes, to obtain a textured surface mesh. We apply the above steps to a large set of data with several million 3D points, and show photorealistic texture mapped 3D models.

Keywords: 3D city model, depth image, occlusion, image restauration, urban simulation

I. INTRODUCTION

Three-dimensional models of urban environments are useful in a variety of applications such as urban planning, training and simulation for urban terrorism scenarios, and virtual reality. Currently, the standard technique for creating large scale city models in an automated or semi-automated way, is to use stereo vision approaches on aerial or satellite images [5, 9, 14]. In recent years, advances in resolution and accuracy of airborne laser scanners have also rendered them suitable for the generation of reasonable models [8, 10]. Both approaches have the disadvantage that their resolution is only in the range of 1 to 2 feet, and more importantly, they can only capture the roofs of the buildings but not the facades. This essential disadvantage prohibits their use in photo realistic walk or drive-through applications. There have been various attempts to acquire the complementary ground-level data necessary to complete existing airborne models, either using stereo vision [3] or 3D laser scanners [11]. But as data has to be acquired in a slow stop-and-go fashion, these approaches do not scale to more than few buildings.

In previous work, we have developed a method capable of rapidly acquiring 3D geometry and texture data for an entire city at the ground level by using fast 2D laser scanners and a digital camera [6, 7]. The data acquisition system is mounted on a truck, moving at normal speeds on public roads, collecting data to be processed offline. This approach has the advantage that data can be acquired continuously, rather than in a stop-and-go fashion, and is therefore extremely fast. Relative position changes are computed with centimeter accuracy by matching successive horizontal laser scans against each other; global position is determined by additional use of aerial photos and digital roadmaps [7]. As a result, façade scan points are registered with the aerial photos or airborne laser scans, facilitating subsequent fusion with models derived from airborne data.

In this paper, our goal is to create a detailed, textured 3D façade mesh to represent the building walls at the highest level of detail. As there are many erroneous scan points, e.g. due to glass surfaces, and foreground objects partially occluding the desired buildings, the generation of a facade mesh is not straightforward. A simple triangulation of the raw scan points by connecting neighboring points whose distance is below a threshold value, does not result in an acceptable reconstruction of the street scenery, as shown in Figures 1(a) and 1(b). Even though the 3D structure can be easily recognized when viewed from a viewpoint near the original acquisition position as in Figure 1(a), the mesh appears noisy due to several reasons; first, there are holes and erroneous vertices due to reflections off the glass on windows; second, there are many pieces of geometry "floating in the air", corresponding to partially captured objects and measurement errors. The mesh appears to be even more problematic when viewed from other viewpoints such as the one shown in Figure 1(b); this is because in this case the large holes in the building facades caused by occluding foreground objects, such as cars and trees, become visible. Furthermore, since the laser scan only captures the frontal view of foreground objects, they become almost unrecognizable when viewed sideways. As

we drive by a street only once, it is not possible to use additional scans from other viewpoints to fill in gaps caused by occlusions, as is done in [2, 11]. Rather, we have to reconstruct occluded areas by only using cues from neighboring scan points; as such, there has been little work to solve this problem [12].

Figure 1: Triangulated raw points; (a) front view; (b) side view.

In this paper, we propose a class of data processing techniques to create visually appealing façade meshes by removing noisy foreground objects and filling holes in the building facades. Our objectives are robustness and efficiency with regards to processing time, in order to ensure scalability to the enormous amount of data resulting from a city scan. The outline of this paper is as follows: In section II we introduce our data acquisition system and position estimation; sections III and IV discuss path splitting and depth image generation schemes. We describe our strategy to transform the raw scans into a visually appealing façade mesh in sections V through VII, automatic texture mapping in section VIII, and the experimental results in section IX.

II. Data Acquisition and Position Estimation

As described in [6], we have developed a data acquisition system consisting of two Sick LMS 2D laser scanners, and a digital color camera with a wide angle lens. As seen in Figure 2, this is mounted on a rack of approximately 3.6 meters height on top of a truck, in order to obtain measurements that are not obstructed by pedestrians and cars. The scanners have a 180^0 field of view with a resolution of 1^0, a range of 80 meters and an accuracy of ±6 centimeters. Both 2D scanners are facing the same side of the street. One is mounted horizontally and is used for position estimation. The estimation is based on scan-to-scan matching and global correction algorithms using aerial photos, and is described in detail in [7]. The other scanner is mounted vertically with the scanning plane orthogonal to the driving direction, and scans the buildings and street scenery as the truck drives by. The camera is oriented in the same direction as the scanners, with its center of projection approximately in the intersection line of the two scanning planes. All three devices are synchronized with each other using hardware-generated signals, and their coordinate systems are registered with respect to each other.

Figure 2: Truck with data acquisition equipment.

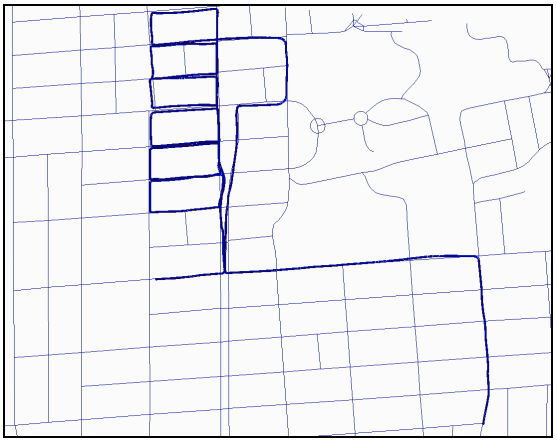


Figure 3: Driven path overlaid with road map.

We introduce a Cartesian world coordinate system [x,y,z] where x,y is the ground plane and z points into the sky. Assuming that the city streets are flat, the position of the truck can be described by the two coordinates x,y and an orientation angle θ of the truck coordinate system. Using the localization methods in [6, 7], current speed, position and orientation of the truck can be estimated for each scan. Thus, the entire "capture" path of the acquisition truck can be reconstructed, as shown in Figure 3, together with a long series of vertical 2D scans, associated with scanner

position. To partially compensate for the unpredictable, non-uniform motion of the truck, the scan series is subsampled such that the spacing between successive scans is roughly equidistant. For each vertical scan S_n used for the 3D reconstruction, there is a tuple (x_n, y_n, θ_n) which describes position and orientation of the scanner in the world coordinate system during acquisition. Furthermore, let $s_{n,\upsilon}$ be the distance measurement on a point in scan S_n with azimuth angle υ. Then, $d_{n,\upsilon}=\cos(\upsilon)\cdot s_{n,\upsilon}$ is the depth value of this point with respect to the scanner, i.e. its orthogonal projection into the ground plane, as shown in Figure 4.

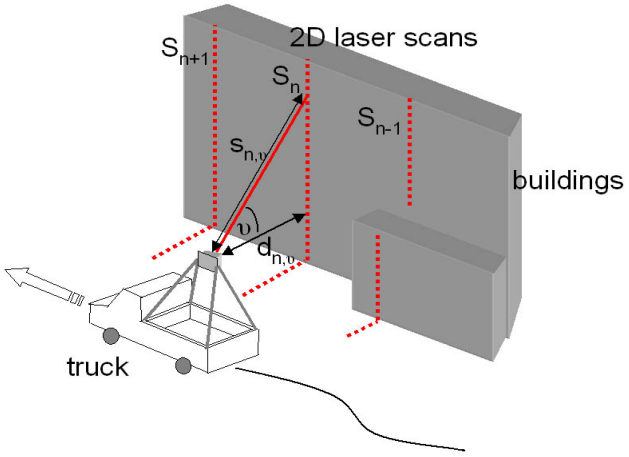


Figure 4: Scanning setup.

III. SEGMENTATION OF THE DRIVING PATH INTO QUASI LINEAR SEGMENTS

The captured data during a twenty minute drive consists of tens of thousands of scan columns. Since successive scans in time correspond to spatially close points, e.g. a building or a side of a street block, it is computationally advantageous not to process the entire data as one block, rather to split it into smaller segments to be processed separately. We impose the constraints that (a) path segments have low curvature, and (b) scan columns have a regular grid structure. This allows us to readily identify the neighbors to right, left, above and below for each point, and, as seen later, is essential for the generation of a depth image and segmentation operations.

Scan points for each truck position are obtained as we drive by the streets. During straight segments, the spatial order of the 2D scan rows is identical to the temporal order of the scans, forming a regular topology. Unfortunately, this order of scan points can be reversed during turns towards the scanner side of the car. Figures 5(a) and (b) show the scanning setup during such a turn, with scan planes indicated by the two dotted rays. During the two vertical scans, the truck performs not only a translation but also a rotation, making the scanner look slightly backwards during the second scan. If the targeted object is close enough, as shown in Figure 5(a), the spatial order of scan points 1 and 2 is still the same as the temporal order of the scans; however, if the object is further away than a critical distance d_{crit}, the spatial order of the two scan points is reversed, as shown in Figure 5(b).

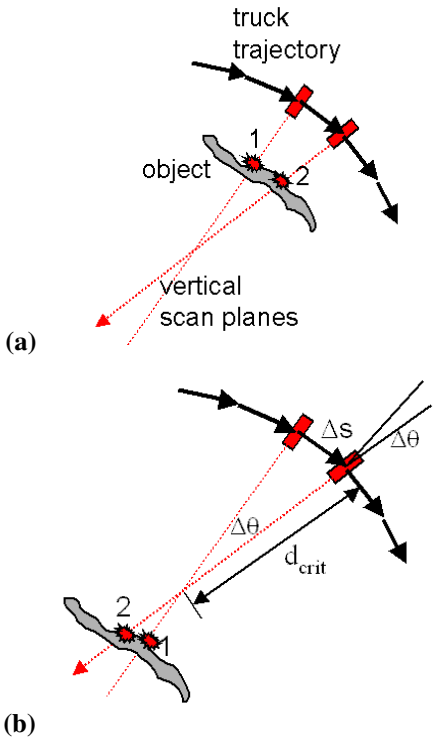


Figure 5: Scan geometry during a turn,
(a) normal scan order for closer objects;
(b) reversed scan order for further objects.

For a given truck translation of Δs, and a rotation $\Delta\theta$ between successive scans, the critical distance can be computed as

$$d_{crit} = \frac{\Delta s}{\sin(\Delta\theta)}.$$

Thus, d_{crit} is the distance at which the second scanning plane intersects with the first scanning plane. For a particular scan point, the order with its predecessors should be reversed if its depth $d_{n,\upsilon}$ exceeds d_{crit}; this means that its geometric location is somewhere in between points of previous scans. The effect of such order reversal can be seen in the marked area in Figure 6. At the corner, the ground and the building walls are scanned twice, first from a direct view and then from an oblique angle, and hence with significantly lower accuracy. For the oblique points, the scans are out of order, destroying the regular topology between neighboring scan points.

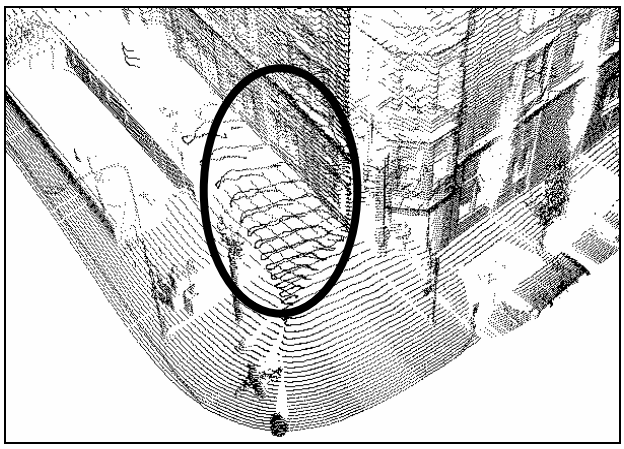


Figure 6: Scan points with reversed order.

Since the "out of order" scans obtained in these scenarios correspond to points that have already been captured by "in order" scans, and are therefore redundant, our approach is to discard them and use only "in order" scans. For typical values of displacement, turning angle, and distance of structures from our driving path, this occurs only in scans of turns with significant angular changes. By removing these "turn" scans and splitting the path at the "turning points", we obtain path segments with low curvature that can be considered as locally quasi-linear, and can therefore be conveniently processed as depth images, as described in the following section. In addition, to ensure that these segments are not too large for further processing, we subdivide them if they are larger then a certain size; specifically, in segments that are longer than 100 meters, we identify vertical scans that have the fewest scan points above street level, corresponding to empty regions in space, and segment at these locations. Furthermore, we detect redundant path segments for areas captured multiple times due to multiple drive bys, and only use one of them for reconstruction purposes. Figures 7(a) and 7(b) show an example of an original path, and the resulting path segments overlaid on a road map, respectively. The small lines perpendicular to the driving path indicate the scanning plane of the vertical scanner for each position.

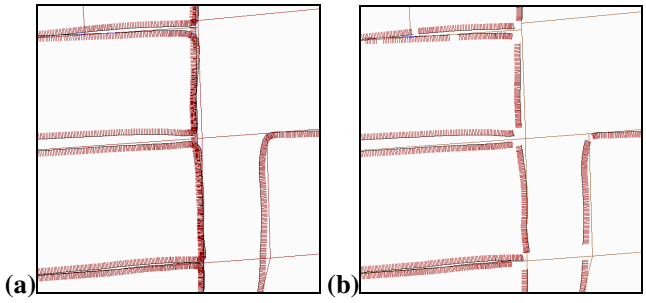
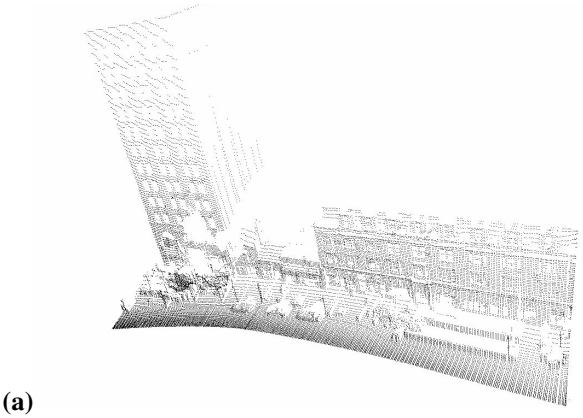


Figure 7: Driven path, (a) before segmentation; (b) after segmentation into quasi-linear segments.

IV. CONVERTING PATH SEGMENTS INTO DEPTH IMAGES

In the previous section, we create path segments that are guaranteed to contain no scan pairs with permuted horizontal order. As the vertical order is inherent to the scan itself, all scan points of a segment form a 3D scan grid with regular, quadrilateral topology. This 3D scan grid allows us to transform the scan points into a 2.5D representation, i.e. a depth image where each pixel represents a scan point, and the gray value for each pixel is proportional to the depth of the scan point. The advantage of a depth image is its intuitively easy interpretation, and the increased processing speed the 2D domain provides. However, most operations that are performed on the depth image can be done just as well on the 3D point grid directly, only not as conveniently.

A depth image is typically used for representing the data from 3D scanners. Even though the way the depth value is assigned to each pixel is dependent on the specific scanner, in most cases it is the distance between scan point and scanner origin, or its cosine with respect to the ground plane. As we expect mainly vertical structures, we choose the latter option and use the depth $d_{n,\upsilon} = \cos(\upsilon) \cdot s_{n,\upsilon}$ rather than the distance $s_{n,\upsilon}$, so that the depth image is basically a tilted height field. The advantage is that in this case points that lie on a vertical line, e.g. a building wall, have the same depth value, and are hence easy to detect and group. Note that our depth image differs from one that would be obtained from a normal 3D scanner, as it does not have a single center from which the scan points are measured; instead, there are different centers for each individual vertical column along the path segment. The obtained depth image is neither a polar nor a parallel projection; it resembles most to a cylindrical projection. Due to non-uniform driving speed and non-linear driving direction, these centers are in general not on a line, but on an arbitrary shaped, though low-curvature curve, and the spacing between them is not exactly uniform. Because of this, strictly speaking the grid position only specifies the topological order of the depth pixels, and not the exact 3D point coordinates. However, as topology and depth value are a good approximation for the exact 3D coordinates, especially within a small neighborhood, we choose to apply our data processing algorithms to the depth image, thereby facilitating use of standard image processing techniques such as region growing. Moreover, the actual 3D vertex coordinates are still kept and used for 3D operations such as plane fitting. Figure 8(a) shows an example of the 3D vertices of a scan grid, and Figure 8(b) shows its corresponding depth image, with a gray scale proportional to $d_{n,\upsilon}$.



(a)

(b)

Figure 8: Scan grid representations; (a) 3D vertices; (b) depth image.

V. Properties of City Laser Scans

In this section, we briefly describe properties of scans taken in a city environment, resulting from the physics of a laser scanner as an active device measuring time-of-flight of light rays. It is essential to understand these properties and the resulting imperfections in distance measurement, since at times they lead to scan points that appear to be in contradiction with human eye perception or a camera. As the goal of our modeling approach is to generate a photo realistic model, we are interested in reconstructing what the human eye or a camera would observe while moving around in the city. As such, we discuss the discrepancies between these two different sensing modalities in this section.

a) Discrepancies due to different resolution

The beam divergence of the laser scanner is about 15 milliradians (mrad) and the spacing, hence the angular resolution, is about 17 mrad. As such, this is much lower than the resolution of the camera image with about 2.1 mrad in the center and 1.4 mrad at the image borders. Therefore, small or thin objects, such as cables, fences, street signs, light posts and tree branches, are clearly visible in the camera image, but only partially captured in the scan. Hence they appear as "floating" vertices, as seen in the depth image in Figure 9.

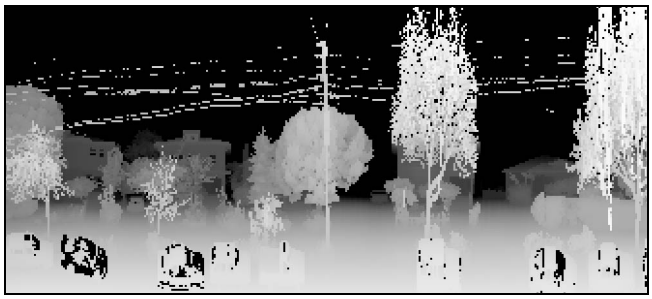


Figure 9: "Floating" vertices.

b) Discrepancies due to the measurement physics

Camera and eye are passive sensors, capturing light from an external source; this is in contrast with a laser scanner, which is an active sensor, and uses light that it emits itself. This results in substantial differences in measurement of reflecting and semitransparent surfaces, which are in form of windows and glass fronts frequently present in urban environments. Typically, there is at least 4% of the light reflected at a single glass/air transition, so a total of at least 8 % per window; if the window has a reflective coating, this can be larger. The camera typically sees a reflection of the sky or a nearby building on the window, often distorted or merged with objects behind the glass. Although most image processing algorithms would fail in this situation, the human brain is quite capable of identifying windows. In contrast, depending on the window reflectance, the laser beam is either entirely reflected, most times in a different direction from the laser itself, resulting in no distance value, or is transmitted through the glass. In the latter case, if it hits a lambertian surface as shown in Figure 10, the backscattered light travels again through the glass. The resulting surface reflections on the glass only weaken the laser beam intensity, eventually below the detection limit, but do not otherwise necessarily affect the distance measurement. To the laser, the window is quasi non-existent, and the measurement point is generally not on the window surface, unless the surface is orthogonal to the beam. In case of multi-reflections, the situation becomes even worse as the measured distance is almost random.

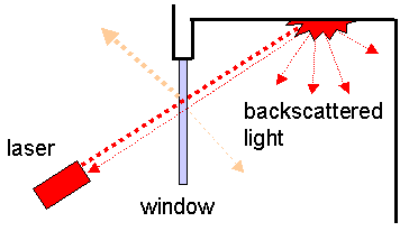


Figure 10: Laser measurement in case of a glass window

c) Discrepancies due to different scan and viewpoints

Laser and camera are both limited in that they can only detect the first visible/backscattering object along a measurement direction and as such cannot deal with occlusions. If there is an object in the foreground, such as a tree in front of a building, the laser cannot capture what is behind it; hence, generating a mesh from the obtained scan points results in a hole in the building. We refer to this type of mesh hole as *occlusion hole*. As the laser scan points resemble a cylindrical projection, but rendering is parallel or perspective, in presence of occlusions, it is impossible to reconstruct the original view without any holes, even for the viewpoints from which data was acquired. This is a special property of our fast 2D data acquisition method. An interesting fact is that the wide-angle camera images captured simultaneously with the scans often contain parts of the background invisible to the laser. These could in potentially be either used to fill in geometry using stereo techniques, or to verify the validity of the filled in geometry obtained from using interpolation techniques.

For a photo realistic model, we need to devise techniques for detecting discrepancies between the two modalities, removing invalid scan points, and filling in holes, either due to occlusion or due to unpredictable surface properties; we will describe our approaches to these problems in the following sections.

VI. MULTI-LAYER REPRESENTATION

To ensure that the facade model looks reasonable from every viewpoint, it is necessary to complete the geometry for the building facades. As our facades are not only manifolds, but also resemble a height field, it is possible to introduce a representation based of multiple depth layers for the street scenery, similar to the one proposed in [1]. Each depth layer is a scan grid, and the scan points of the original grid are assigned to exactly one of the layers. If at a certain grid location there is a point in a foreground layer, this location is empty in all layers behind it and needs to be filled in.

Even though the concept can be applied to an arbitrary number of layers, for our problem it is sufficient to generate only two layers, a foreground and a background. To assign a scan point to either one of the two layers we make the following assumptions about our environment: Main structures, i.e. buildings, are usually (a) vertical, and (b) extend over several feet in horizontal dimension. For each vertical scan n corresponding to a column in the depth image, we define the main depth as the depth value that occurs most frequently, as shown in Figure 11. The scan vertices corresponding to the main depth lie on a vertical line, and the first assumption suggests that this is a main structure, such as a building, or perhaps other vertical objects, such as a street light or a tree trunk. With the second assumption, we filter out the latter class of vertical objects. More specifically, our processing steps can be described as follows:

We sort all depth values $s_{n,\upsilon}$ for each column n of the depth image into a histogram as shown in Figures 11(a) and (b), and detect the peak value and its corresponding depth. Applying this to all scans results in a 2D histogram as shown in Figure 12, and an individual main depth value estimate for each scan. According to the second assumption, isolated outliers are removed by applying a median filter on these main depth values across the scans, and a final depth value is assigned to each column n. We define a "split" depth, γ_n, for each column n, and set it to the first local minimum of the histogram occurring immediately before main depth, i.e. with a depth value smaller than the main depth. Taking the first minimum in the distribution instead of the main value itself has the advantage that points clearly belonging to foreground layers are splits off, whereas overhanging parts of buildings, for which the depth is slightly smaller than the main depth, are kept in the main layer where they logically belong to, as shown in Figure 11.

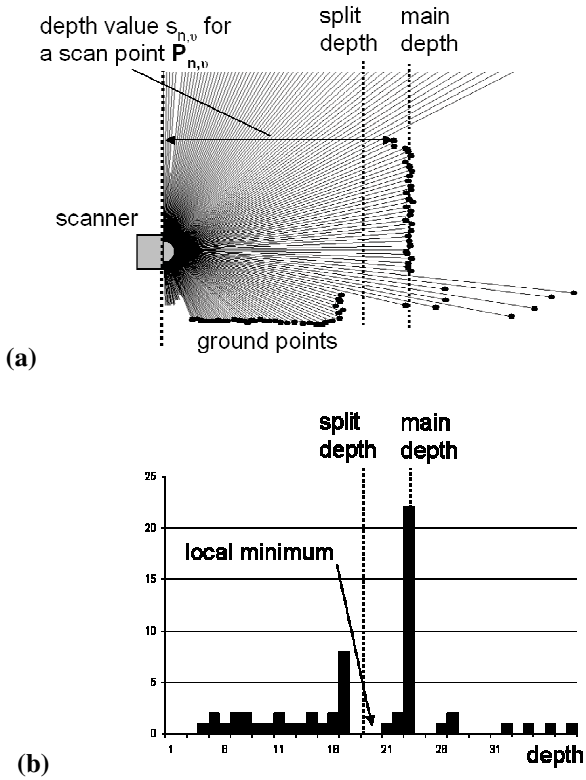


Figure 11: Main depth computation for a single scan n; (a) laser scan with rays indicating the laser beams and dots at the end the corresponding scan points; (b) computed depth histogram.

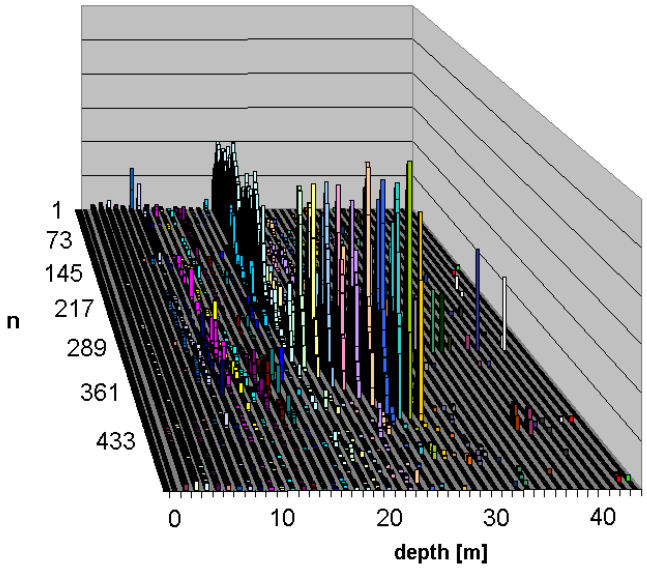
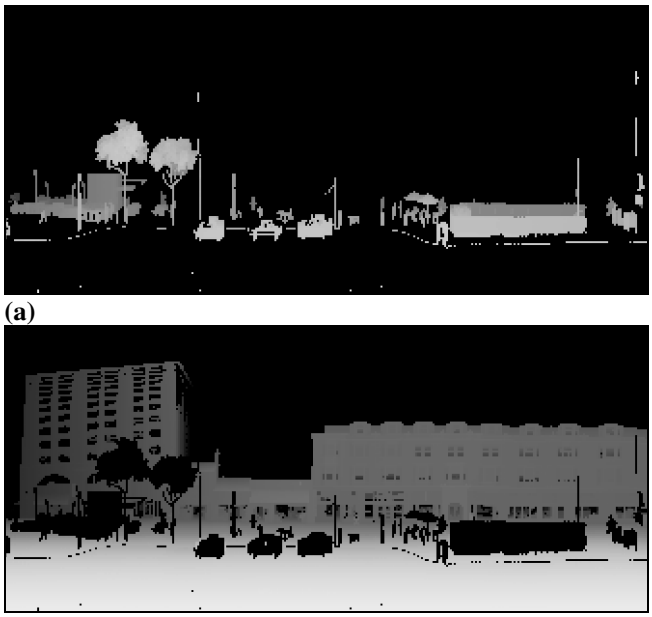


Figure 12: Two-dimensional histogram for all scans.

A point can be identified as a ground point if its z coordinate has a small value and its neighbors in the same scan column have a similarly low z value. We prefer to include the ground in our models, and as such, assign ground points also to the background layer. Therefore, we perform the layer split by assigning a scan point $P_{n,\upsilon}$ to the background layer, if $s_{n,\upsilon} > \gamma_n$ or $P_{n,\upsilon}$ is a ground point, and otherwise, to the foreground layer. Figure 13 shows an example for the resulting foreground and background layers.



(a)

(b)

Figure 13: (a) Foreground layer; (b) background layer.

Since the steps described in this section assume the presence of vertical buildings, they cannot be expected to work for segments that are dominated by trees; this also applies to the processing steps we introduce in the following sections. As our goal is to reconstruct buildings, path segments can be left unprocessed and included "as is" in the city model, if they do not contain any structure. A characteristic of a tree area is its fractal geometry, resulting in a large variance among adjacent depth values, or even more characteristic, many significant vector direction changes for the edges between connected mesh vertices. We define a coefficient for the fractal nature of a segment by counting vertices with direction changes greater than a specific angle, e.g. twenty degrees, and dividing them by the total number of vertices. If this coefficient is large, the segment is most likely a tree area and should not be made subject to the processing steps described in this section. This is for example the case for the segment shown in Figure 9.

After layer splitting, all grid locations occupied in the foreground layer are empty in the background layer. The vertical laser does not capture any occluded geometry, and in the next section we will describe an approach for filling these empty grid locations based on neighboring pixels. However, in our data acquisition system there are 3D vertices available from other sources, such as stereo vision and the horizontal scanner used for navigation. Thus, it is conceivable to use these additional information to fill some in the depth layers. Our approach to doing so is as follows:

Given a set of 3D vertices V_i obtained from a different modality, determine the closest scan direction for each vertex and hence the grid location (n,υ) it should be assigned to. As shown in Figure 14, each V_i is assigned to the vertical scanning plane, S_n, with the smallest Euclidean distance, corresponding to column n in the depth image. Using simple trigonometry, the scanning angle under which this vertex appears in the scanning plane, and hence the depth image row υ, can be computed, as well as the depth $d_{n,\upsilon}$ of the pixel.

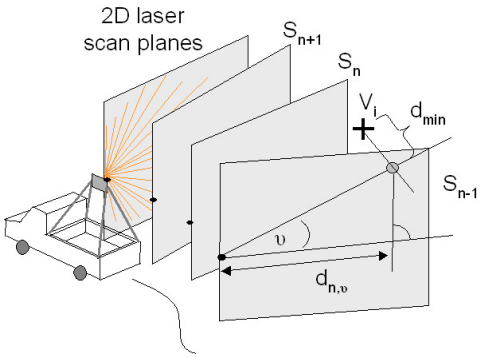


Figure 14: Sorting additional points into the layers.

We can now use these additional vertices to fill in the holes. To begin with, all vertices that do not belong to background holes are discarded. If there is exactly one vertex falling onto a grid location, its depth is directly assigned to that grid location; for situations with multiple vertices, median depth value for this location is chosen. Figure 15 shows the background layer from Figure 13(b) after sorting in 3D vertices from stereo vision and horizontal laser scans. As seen, some holes can be entirely filled in, and the size of others becomes smaller, e.g. the holes due to trees in the tall building on the left side. Note that this intermediate step is optional and depends on the availability of additional 3D data.

Figure 15: Background layer after sorting in additional points from other modalities.

VII. Background Layer Postprocessing and Mesh Generation

In this section, we will describe a strategy to remove erroneous scan points, and to fill in holes in the background layer. There exists a variety of successful hole filling approaches, for example based on fusing multiple scans taken from different positions [2]. Most previous work on hole filling in the literature has been focused on reverse engineering applications, in which a 3D model of an object is obtained from multiple laser scans taken from different locations and orientations. Since these existing hole filling approaches are not applicable to our experimental setup, our approach is to estimate the actual geometry based on the surrounding environment and reasonable heuristics. One cannot expect this estimate to be accurate in *all* possible cases, rather to lead to an acceptable result in *most* cases, thus reducing the amount of further manual interventions and postprocessing drastically. Additionally, the estimated geometry could be made subject to further verification steps, such as consistency checks, by applying stereo vision techniques to the intensity images captured by the camera.

Our data typically exhibits the following characteristics:

- Occlusion holes, such as those caused by a tree, are large and can extend over substantial parts of a building.
- A significant number of scan points surrounding a hole may be erroneous due to glass surfaces.
- In general, a spline surface filling is unsuitable, as building structures are usually piecewise planar with sharp discontinuities.
- The size of data set resulting from a city scan is huge, and therefore the processing time per hole should be kept to a minimum.

Based on the above observations, we propose the following steps for data completion:

1. Detecting and removing erroneous scan points in the background layer

We assume that erroneous scan points are due to glass surfaces, i.e. the laser measured either an internal wall/object, or a completely random distance due to multi-reflections. Either way, the depth of the scan points measured through the glass is substantially greater than the depth of the building wall, and hence these points are candidates for removal. Since glass windows are usually framed by the wall, we remove the candidate points only if they are embedded among a number of scan points at main depth. An example of the effect of this step can be seen by comparing the windows of the original image in Figure 16(a) with the processed background layer in Figure 16(b).

2. Segmenting the occluding foreground layer into objects

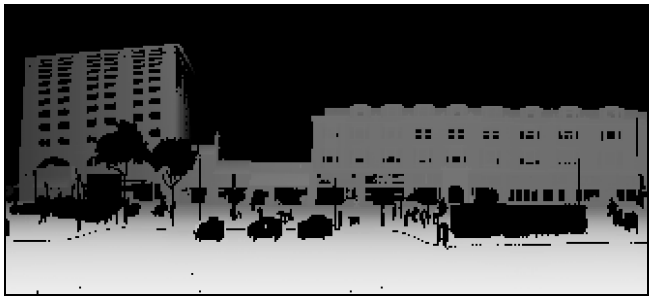
In order to determine holes in the background layer caused by occlusion, we segment the occluding foreground layer into objects and project segmentation onto the background layer. This way, holes can be filled in one "object" at a time, rather than all at the same time; this approach has the advantage that more localized hole filling algorithms are more likely to result in visually pleasing models than global ones. We segment the foreground layer by taking a random seed point that does not yet belong to a region, and applying a region growing algorithm that iteratively adds neighboring pixels if their depth discontinuity or their local curvature is small enough. This is repeated until all pixels are assigned to a region, and the result is a region map as shown in Figure 16(c). For each foreground region, we determine boundary points on the background layer; these are all the valid pixels in the background layer that are close to hole pixels caused by the occluding object.

3. Filling occlusion holes in the background layer for each region

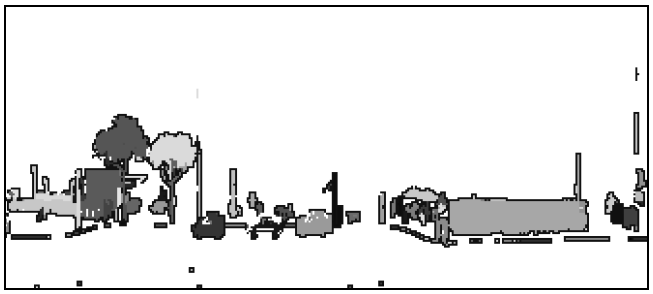
As the foreground objects are located in front of main structures and in most cases stand on the ground, they occlude not only parts of a building, but also parts of the ground. Specifically, an occlusion hole caused by a low object, such as a car, with a large distance to the main structure behind it, is typically located only in the ground and not in the main structure. This is because the laser scanner is mounted on top of a rack, and as such has a top down view of the car. As a plane is a good approximation to the ground, we fill in the ground section of an occlusion hole by the ground plane. Therefore, for each depth image

column, i.e. each scan, we compute the intersection point between the line through the main depth scan points and the line through ground scan points. The angle υ'_n at which this point appears in the scan marks the virtual boundary between ground part and structure part of the scan; we fill in structure points above and ground points below this boundary differently.

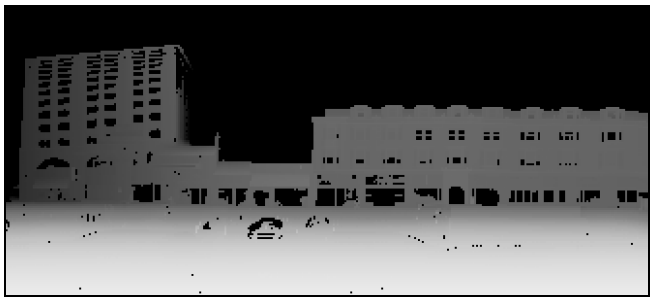
(a) Initial depth image.



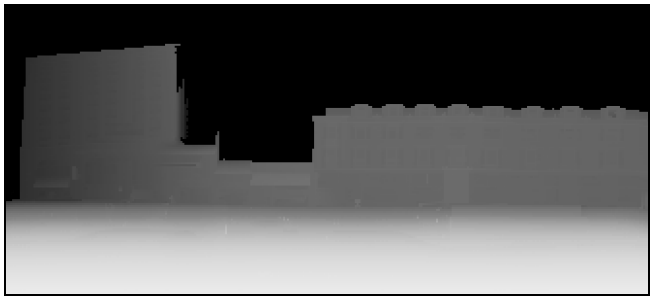
(b) Background layer after removing invalid scan points.



(c) Foreground layer segmented.



(d) Occlusion holes filled.



(e) Final background layer after filling remaining holes.

Figure 16: Processing steps of depth image.

Applying a RANSAC algorithm, we find the plane with the maximum consensus, i.e. maximum number of ground boundary points on it, as the optimal ground plane for that local neighborhood. Each hole pixel with $\upsilon < \upsilon'_n$ is then filled in with a depth value according to this plane. It is possible to apply the same technique for the structure hole pixels, i.e. the pixels with $\upsilon > \upsilon'_n$, by finding the optimal plane through the structure boundary points and filling in the hole pixels accordingly. However, we have found that in contrast to the ground, surrounding building pixels do not often lie on a plane. Instead, there are discontinuities due to occluded boundaries and building features such as marquees or lintels, in most cases extending horizontally across the building. Therefore, rather than filling holes with a plane, we fill in structure holes line by line horizontally, in such a way that the depth value at each pixel is the linear interpolation between the closest right and left structure boundary point, if they both exist; otherwise no value is filled in. In a second phase, a similar interpolation is done vertically, using the already filled in points as valid boundary points. This method is not only simple and therefore computationally efficient, it also takes into account the surrounding horizontal features of the building in the interpolation. The resulting background layer is shown in Figure 16(d).

4. Postprocessing the background layer

The resulting depth image and the corresponding 3D vertices can be improved by removing scan points that remain isolated, and by filling small holes surrounded by geometry using linear interpolation between neighboring depth pixels. The final background layer after applying all processing steps is shown in Figure 16(e).

In order to create a mesh, each depth pixel can be transformed back into a 3D vertex, and each vertex $P_{n,\upsilon}$ is connected to a depth image neighbor $P_{n+\Delta n,\upsilon+\Delta\upsilon}$ if

$$|s_{n+\Delta n,\upsilon+\Delta\upsilon} - s_{n,\upsilon}| < s_{\max} \quad \text{or if}$$

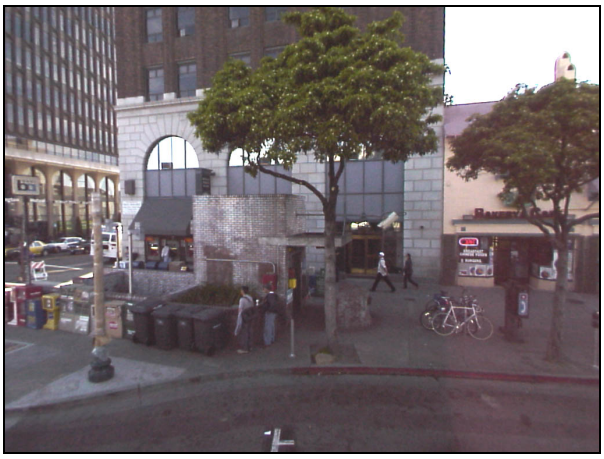
$$\cos\varphi > \cos\varphi_{\max}$$

with

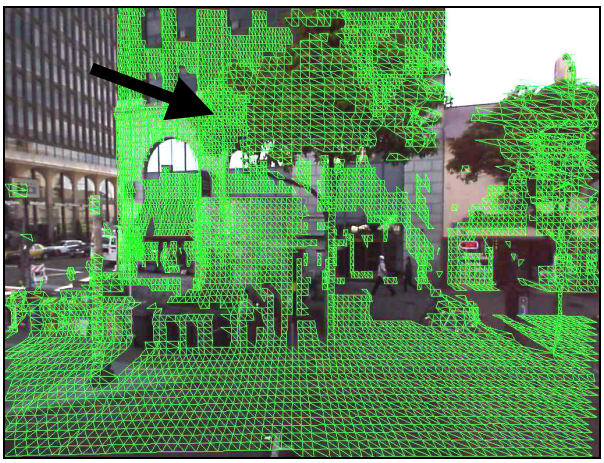
$$\cos\varphi = \frac{(\vec{P}_{n-\Delta n,\upsilon-\Delta\upsilon} - \vec{P}_{n,\upsilon}) \cdot (\vec{P}_{n,\upsilon} - \vec{P}_{n+\Delta n,\upsilon+\Delta\upsilon})}{|\vec{P}_{n-\Delta n,\upsilon-\Delta\upsilon} - \vec{P}_{n,\upsilon}| \cdot |\vec{P}_{n,\upsilon} - \vec{P}_{n+\Delta n,\upsilon+\Delta\upsilon}|}.$$

Intuitively, neighbors are connected if their depth difference does not exceed a threshold s_{max} or the local angle between neighboring points is smaller than threshold angle φ_{max}. The second criteria is intended to connect neighboring points that are on a line, even if their depth difference exceeds

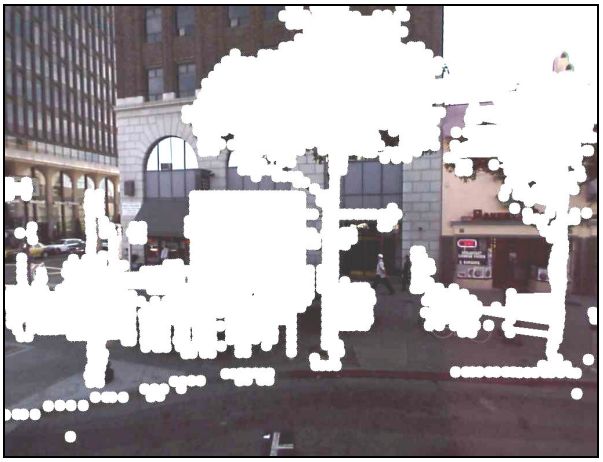
s_{max}. The resulting quadrilateral mesh is split into triangles, and mesh simplification tools such as Qslim [4] or VTK decimation can be applied to reduce the number of triangles.



(a) Camera Image.



(b) Mesh triangles projected into the image, with some foreground and background triangles projecting to the same image area (arrow).



(c) Foreground objects marked white.

Figure 17: Mesh triangles projected into camera images.

VIII. Automatic Texture Mapping

As photorealism cannot be achieved by using geometry alone, and requires color texture as well, our data acquisition system includes a digital color camera with a wide angle lens. The camera is synchronized with the two laser scanners, and is calibrated against the laser scanners' coordinate system, and hence, the camera positions can be computed for all images. After calibrating the camera and removing the lens distortion in the images, each 3D vertex can be mapped to its corresponding pixel in an intensity image by a simple projective transformation. As the 3D mesh triangles are small compared to their distance to the camera, perspective distortions within a triangle can be neglected, and each mesh triangle can be mapped to a triangle in the picture by applying the projective transformation to the three corner points.

As described in section V, camera and laser scanner have different viewpoints during data acquisition, and in most camera pictures at least some mesh triangles of the background layer are occluded by foreground objects; this is particularly true for triangles that consist of filled-in points. An example of this is shown in Figure 17(b), where triangles of the tree and the building project to the same areas in the image. Although the pixel location of the projected background triangles is correct, the corresponding texture triangles are incorrect, and merely correspond to the foreground objects.

However, after splitting the scan points to the two layers, the foreground geometry is readily identified, and both foreground scan points and triangles can be marked in each camera picture, as shown in Fig 17(c) with white color. In order to select specific camera images to be used for texture mapping of a specific mesh triangle of the background layer, for each picture containing a triangle in its field of view, we determine whether any of the triangle's corner points maps onto a white marked pixel; if this is the case, the picture cannot be used for texturing the triangle. For the particular wide angle lens we use and typical building topologies, a background layer point is usually in the field of view of about 10 to 20 pictures. In most situations, we can use multiple images, and as such, we choose the most direct view to texture map the triangle. However, if foreground object and building façade are too close, some façade triangles might not be visible in any picture and hence cannot be texture mapped at all. A possible solution for this case is to apply a texture synthesis algorithm in order to create artificial texture. We are currently investigating this approach.

IX. Results

We drove our equipped truck on a 6769 meters long path in downtown Berkeley, starting from Blake street through Telegraph avenue, and in loops around the downtown

blocks. During this 24-minute-drive, we captured 107,082 vertical scans, consisting of 14,973,064 scan points. Applying the described path splitting techniques, we cut this path into 73 segments, as shown in Figure 18 overlaid with a road map. There is no need for further manual cutting, even at Shattuck Avenue, where the "Manhattan geometry" of Berkeley is not preserved. For each of the 73 segments, we generate two meshes for comparison: the first mesh is obtained directly from the raw scans, and the second one from the depth image to which we have applied the postprocessing steps described in previous sections. The processing time for the millions of scan points of the entire path is about 2 hours on a 2 GHz Pentium 4 PC.

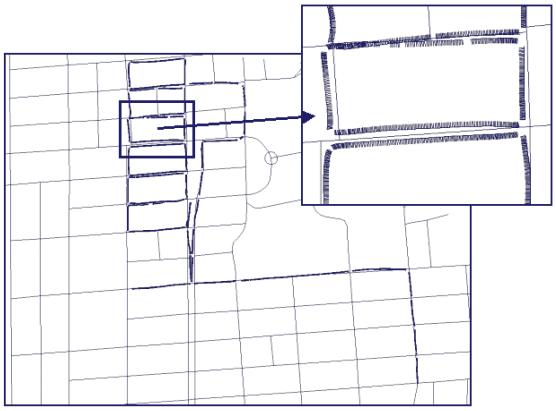


Figure 18: Entire path after split in quasi-linear segments.

For 12 out of the 73 segments, additional 3D vertices derived from stereo vision techniques are available, and hence, sorting in these 3D points into the layers based on section VI does fill some of the holes. For these specific holes, we have compared the results based on stereo vision vertices with those based on interpolation alone as described in section VII, and have found no substantial difference; often the interpolated mesh vertices appear to be slightly more accurate, as they are less noisy than the stereo vision based vertices. Figure 20(a) shows an example before processing, and Figure 20(b) shows the tree holes completely filled in by stereo vision vertices. As seen, the outline of the original holes can still be recognized in Figure 20(b), whereas the points generated by interpolation alone are almost indistinguishable from the surrounding geometry, as seen in Figure 20(c).

Significantly better	35	48 %
Better	17	23 %
Same	15	21 %
Worse	5	7 %
Significantly worse	1	1 %
Total	*73*	*100%*

Table 1: Subjective comparison of the processed mesh vs. the original mesh for all 73 segments.

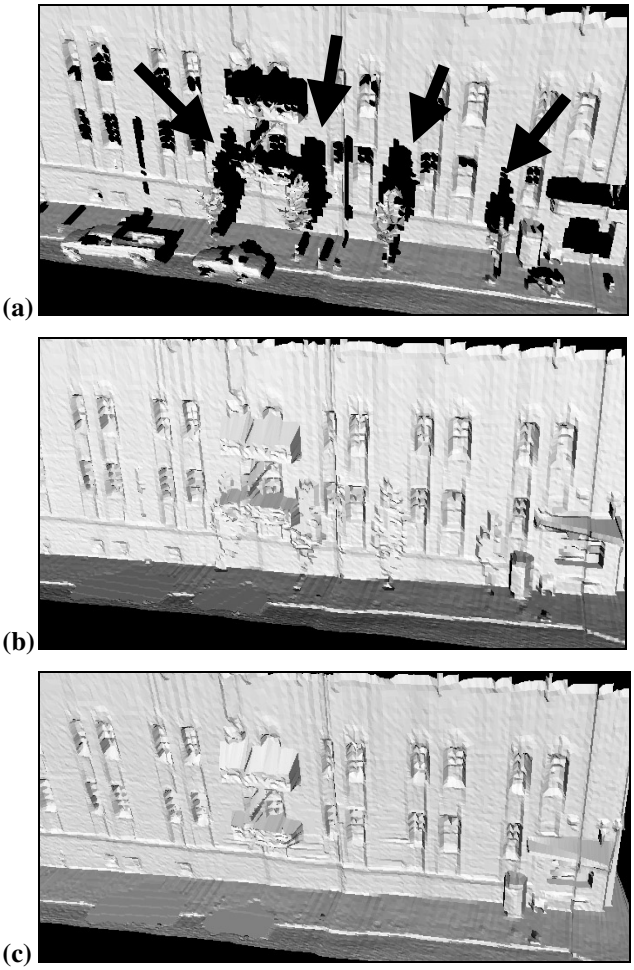


Figure 19: Hole filling (a) original mesh with holes behind occluding trees; (b) filled by sorting in additional 3D points using stereo vision; (c) filled by using the interpolation techniques of section VII.

We have found our approach to work well in the downtown areas, where there are clear building structures and few trees. However, in residential areas, where the buildings are often almost completely hidden behind trees, it is difficult to accurately estimate the geometry. As we do not have the ground truth to compare with, and as our main concern is the visual quality of the generated model, we have manually inspected the results and subjectively determined the degree to which the proposed postprocessing procedures have improved the visual appearance. The evaluation results for all 73 segments before and after postprocessing techniques described in this paper are shown in Table 1; the postprocessing does not utilize auxiliary 3D vertices from horizontal laser scanner or the camera. Even though 8 % of all processed segments appear visually worse than the original, the overall quality of the façade models is significantly improved. The important downtown segments are in most cases ready to use and do not require further manual intervention.

The few problematic segments all occur in residential areas, consisting mainly of trees. The tree detection algorithm described in section VI classifies ten segments as "critical" in that too many trees are present; all six problematic segments corresponding to "worse" and "significantly worse" rows in Table 1 are among them, yet none of the improved segments in rows 1 and 2 are detected as critical. This is significant because it shows that (a) all problematic segments correspond to regions with a large number of trees, and (b) they can be successfully detected and hence not be subjected to the proposed steps. Table 2 shows the evaluation results if only non-critical segments are processed. As seen, the postprocessing steps described in this paper together with the tree detection algorithm improve over 80% of the segments, and never result in degradations for any of the segments.

Significantly better	35	56 %
Better	17	27 %
Same	11	17 %
Worse	0	0 %
Significantly worse	0	0 %
Total	*63*	*100%*

Table 2: Subjective comparison of the processed mesh vs. the original mesh for the segments automatically classified as non-tree-areas.

In Figure 20 we show before and after examples, and the corresponding classifications according to Tables 1 and 2. As seen, except for pair "f", the proposed postprocessing steps result in visually pleasing models. Pair f in Figure 20 is classified by our tree detection algorithm as critical, and hence, should be left "as is" rather than processed. We have further reduced all 73 meshes using the Qslim mesh simplification tool [4] to create multiple levels of details. This enables us to render the façades for the entire path with a standard VRML viewer as shown in Figure 21. For better visualization, these facades are superimposed over an aerial image.

Note that the evaluation of our technique is based on comparing the non-textured geometry; in a comparison in which both models are texture mapped, the processed mesh is more likely to be visually superior to the original. Texture distracts the human eye from geometry imperfections such as those introduced by hole filling algorithms. In Figure 22 we compare a textured mesh of a city block without and with processing. As seen, the visual difference between the two meshes is striking, for the reasons described above. Note the façade area occluded by the two trees on the left side of the original mesh has been filled in, and texture mapped using camera views as described in section VIII. A few triangles are not visible in any camera image and are therefore left untextured. We plan to subject these to a texture synthesis algorithm in a near future.

X. Conclusions and Future Work

We have proposed a strategy to create building facade meshes from large laser surface scans. Future work will focus on using color and texture cues to verify filled-in geometry. Additionally, foreground objects could be classified and replaced by appropriate generics; the ground-level façade meshes can also be merged with rooftop models obtained from aerial view.

XI. Acknowledgement

This work was sponsored by Army Research Office under contract DAAD19-00-1-0352. We wish to thank Sick, Inc. for their support. We also wish to thank John Flynn for providing the stereo vision results.

XII. References

[1] N.L. Chang and A. Zakhor, "A Multivalued Representation for View Synthesis", Proc. Int'l Conference on Image Processing, Kobe, Japan, 1999, vol. 2, pp. 505-509

[2] B. Curless and M. Levoy, "A volumetric method for building complex models from range images", SIGGRAPH, New Orleans, 1996, p. 303-312

[3] A. Dick, P. Torr, S. Ruffle, and R. Cipolla, "Combining Single View Recognition and Multiple View Stereo for Architectural Scenes", International Conference on Computer Vision, Vancouver, Canada, 2001, p. 268-74

[4] M. Garland and P. Heckbert, "Surface Simplification Using Quadric Error Metrics", SIGGRAPH '97, Los Angeles, 1997, p. 209-216

[5] D. Frere, J. Vandekerckhove, T. Moons, and L. Van Gool, "Automatic modelling and 3D reconstruction of urban buildings from aerial imagery", IEEE International Geoscience and Remote Sensing Symposium Proceedings, Seattle, 1998, p.2593-6

[6] C. Frueh and A. Zakhor, "Fast 3D model generation in urban environments", IEEE Conf. on Multisensor Fusion and Integration for Intelligent Systems, Baden-Baden, Germany, 2001, p. 165-170

[7] C. Frueh and A. Zakhor, "3D model generation of cities using aerial photographs and ground level laser scans", Computer Vision and Pattern Recognition, Hawaii, USA, 2001, p. II-31-8, vol.2. 2

[8] N. Haala and C. Brenner, "Generation of 3D city models from airborne laser scanning data", Proc. EARSEL workshop on LIDAR remote sensing on land and sea, Tallin, Esonia, 1997, p.105-112

[9] Z.Kim, A. Huertas, and R. Nevatia, "Automatic description of Buildings with complex rooftops from multiple images", Computer Vision and Pattern Recognition, Kauai, 2001, p. 272-279

[10] H.-G. Maas, "The suitability of airborne laser scanner data for automatic 3D object reconstruction", 3. Int'l Workshop on Automatic Extraction of Man-Made Objects, Ascona, Switzerland, 2001

[11] I. Stamos and P.E. Allen, "3-D model construction using range and image data." Computer Vision and Pattern Recognition, Hilton Head Island, 2000, p.531-6

[12] F. Stulp, F. Dell'Acqua, and R. B. Fisher, "Reconstruction of surfaces behind occlusions in range images", Proc. 3rd Int. Conf. on 3-D Digital Imaging and Modeling, Montreal, Canada, 2001, p. 232-239

[13] S. Thrun, W. Burgard, and D. Fox, "A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping", Proc. of International Conference on Robotics and Automation, San Francisco, 2000, p..321-8, vol. 1. 4

[14] C. Vestri and F. Devernay, "Using Robust Methods for Automatic extraction of buildings", Computer Vision and Pattern Recognition, Hawaii, USA, 2001, p. I-133-8, vol.1. 2

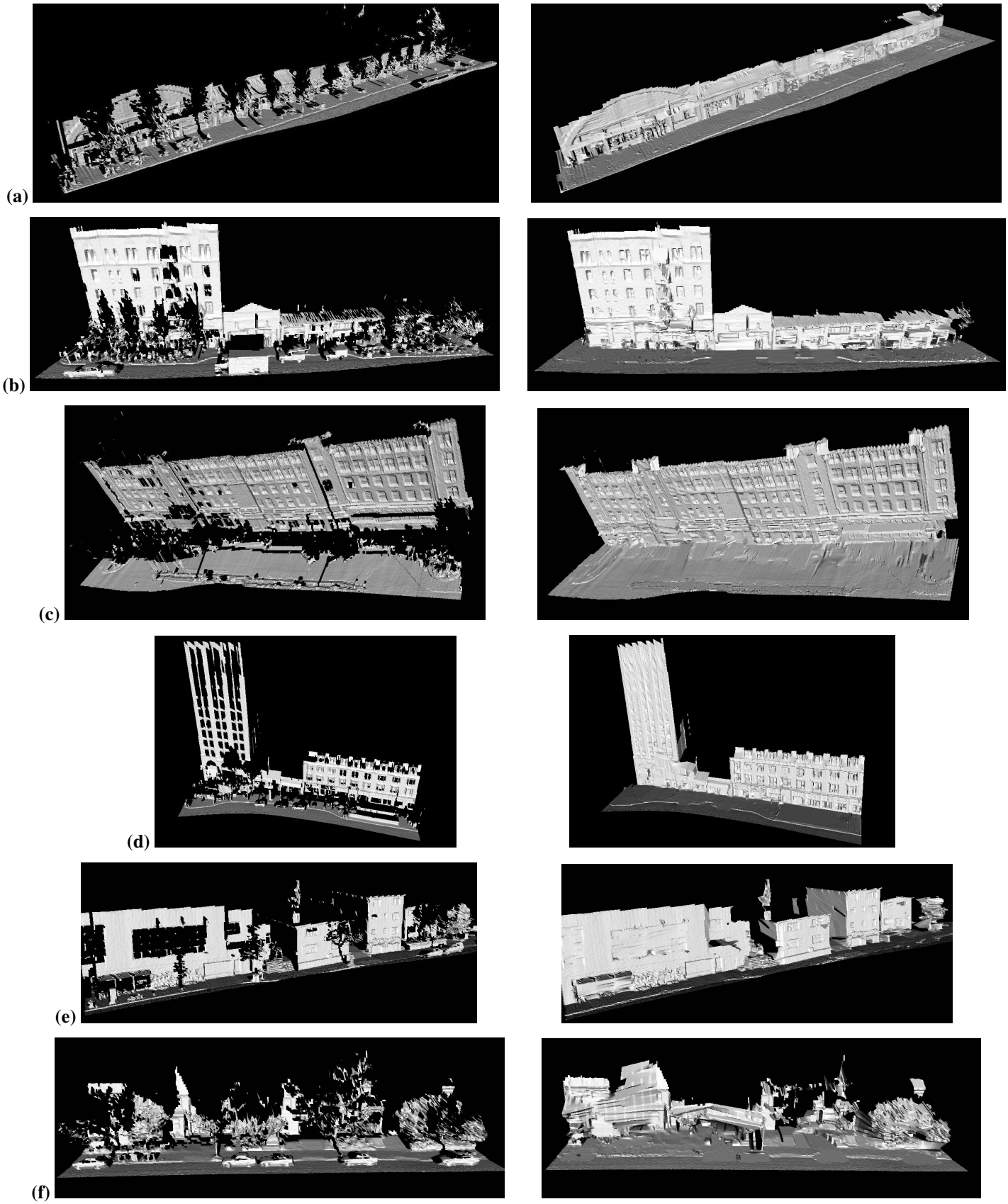


Figure 20: Generated meshes, left side original, right side after the proposed foreground removal and hole filling procedure. The classification for the visual impression is "significantly better" for the first four image pairs, "better" for pair e and "worse" for pair f.

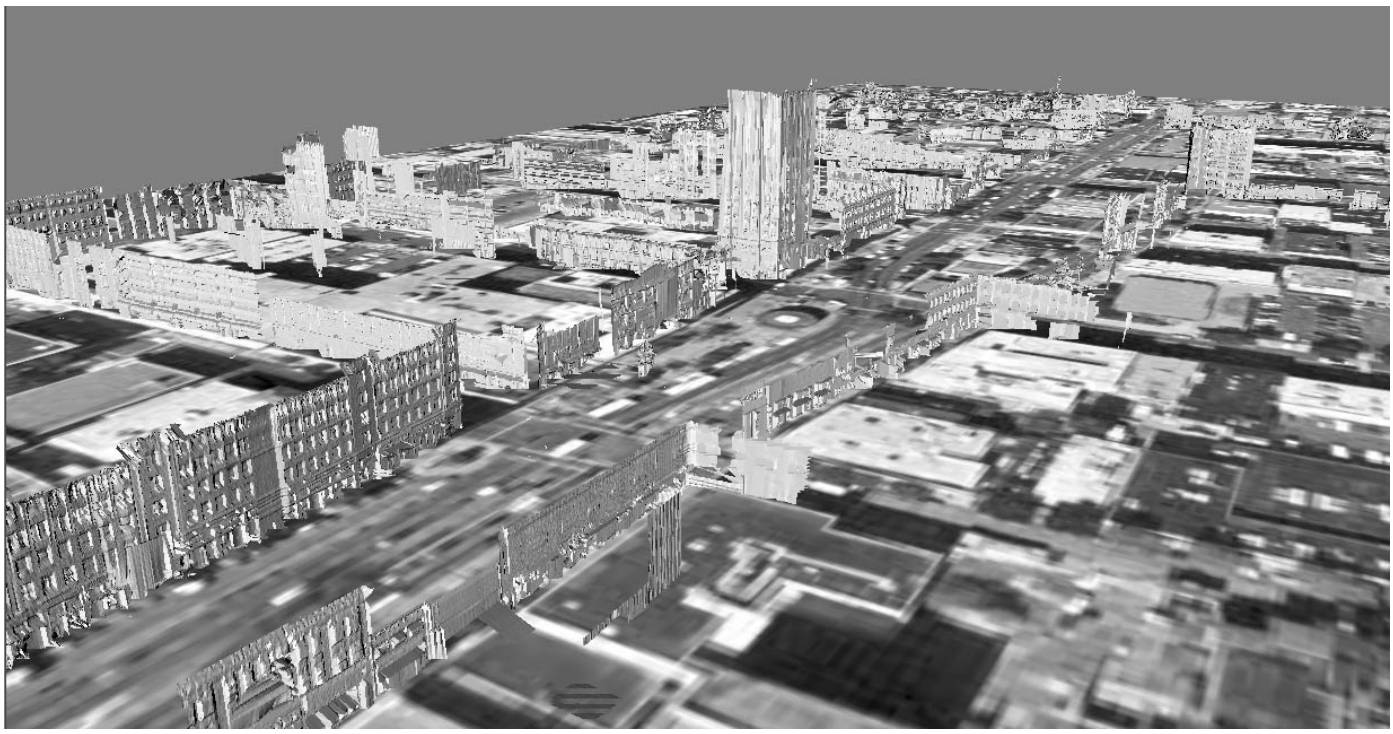


Figure 21: Non-textured facade model for the entire path, overlaid on top of an aerial photo.

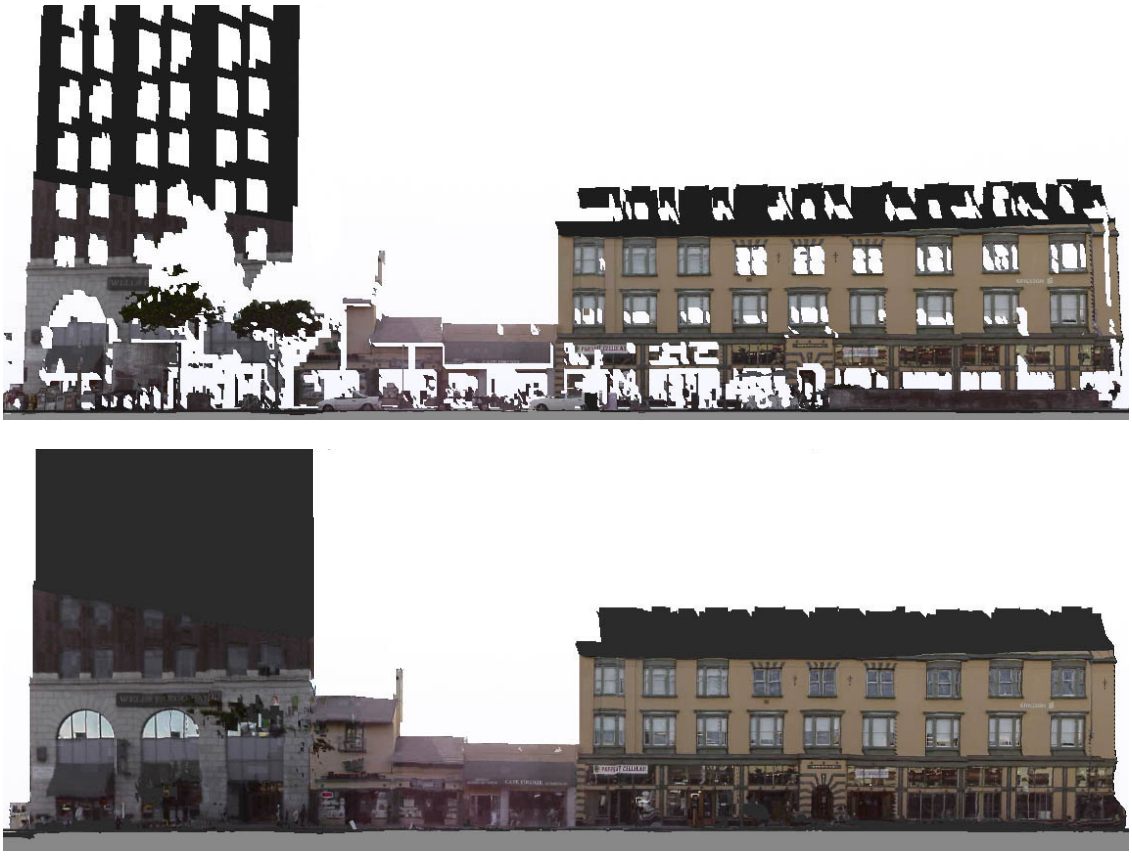


Figure 22: Textured facade mesh without (top) and with (bottom) processing.