Fast Compact City Modeling for Navigation Pre-Visualization

Nico Cornelis, Kurt Cornelis and Luc Van Gool
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10
B-3001 Heverlee, Belgium

Abstract

Nowadays, GPS-based car navigation systems mainly use speech and aerial views of simplified road maps to guide drivers to their destination. However, drivers often experience difficulties in linking the simple 2D aerial map with the visual impression that they get from the real environment, which is inherently ground-level based. Therefore, supplying realistically textured 3D city models at ground-level proves very useful for pre-visualizing an upcoming traffic situation. Because this pre-visualization can be rendered from the expected future viewpoints of the driver, the latter will more easily understand the required maneuver. 3D city models can be reconstructed from the imagery recorded by surveying vehicles. The vastness of image material gathered by these vehicles, however, puts extreme demands on vision algorithms to ensure their practical usability. Algorithms need to be as fast as possible and should result in compact, memory efficient 3D city models for future ease of distribution and visualization. For the considered application, these are not contradictory demands. Simplified geometry assumptions can speed up vision algorithms while automatically guaranteeing compact geometry models. We present a novel city modeling framework which builds upon this philosophy to create 3D content at high speed which could allow for pre-visualization of any conceivable traffic situation by car navigation modules.

1. Introduction

Today, the main assistance modes offered by GPS-based car navigation modules are speech and/or a display of a very simplified aerial representation describing the upcoming traffic situation. Navigation mistakes often arise due to the difficulty of interpreting this information correctly in the context of the real visual environment. We aim at simplifying this interpretation by offering a pre-visualization of a required traffic maneuver, by rendering a virtual trajectory through a realistically texture-mapped 3D model of the environment. The work has been carried out in consultation with an industrial navigation content provider. Figure 1 demonstrates how this ground-level based pre-visualization requires less effort to interpret.

Figure 1. Left: Sometimes it is difficult to relate the aerial map, offered by the car navigation system, to the real environment. Right: A ground-level based pre-visualization of the required maneuver avoids these difficulties.

Texture-mapped 3D city models can be extracted from the imagery collected by survey vehicles. These are equipped with cameras, GPS/INS units, odometry sensors, etc., and drive around daily to record new city data which can aid car navigation. For our envisioned application, when a survey vehicle performed in the past the exact maneuver currently requested by the driver, the playback of the recorded survey image sequence would suffice to illustrate the maneuver. However, the number of possible traffic maneuvers is so enormous that pre-recording and storing such demonstration sequences is practically impossible. Reconstructing 3D city models from the survey sequences and rendering virtual trajectories through them offers a more memory friendly and flexible solution.

The extracted 3D models must be as simple as possible to keep storage requirements low and to render them in real-time on car navigation systems. Furthermore, the time needed to extract these models from the survey sequences should also be low to ensure practical usability in light of

0-7695-2646-2/06 $20.00 (c) 2006 IEEE

the vast extent of image material gathered by survey vehicles. In this paper, we present a city modeling framework which can run in real-time, thereby offering the possibility of online processing while the survey vehicle is recording. A realistically textured, compact 3D model of the recorded scene can already be available when the survey vehicle returns to its home base.

2. Previous Work

City modeling used to be mainly performed on aerial images. Building types and locations were manually indicated or recognized using computer vision algorithms, and Digital Elevation Maps supplied by airborne laser scanners [5, 6, 7, 10, 15, 16, 17]. Much could already be accomplished with the resulting models, however, they usually lacked a realistic impression at ground level since building facades could not be textured from aerial imagery. Today, we find laser scanners mounted on mobile survey platforms gathering 3D depths and textures for building facades throughout cities [3, 4, 9, 12, 13] filling the gaps where aerial imagery could not reach. Furthermore, mobile reconstruction systems based on passive 3D vision algorithms are emerging.

The results of laser systems are very detailed and impressive. These models could be used as is or be simplified to save on memory. To this day, however, laser-equipped survey vehicles are sparse and vast amounts of data has already been gathered by survey vehicles using video-streams annotated with GPS/INS measurements in order to geo-reference them. Vision algorithms are the key to extracting 3D information from these video-streams. Most computer vision city modeling algorithms appearing today try to extract detailed 3D from video-streams using state-of-the-art dense reconstruction algorithms which are not practically usable due to their computational cost.

Recently, real-time dense reconstruction algorithms have been developed that make use of the graphics hardware such as [1], which is based upon the work of [18]. Due to their local optimization approach, however, they lack a more global constraint which is needed to disambiguate between multiple possible matches in the case of repeating patterns such as facades. Also, the integration of the resulting depth maps into one consistent model would introduce a time demanding integration step. Keeping the final application of the 3D model in mind, the necessary level of detail is low and suggests vision algorithms which use this to gain speed.

3 The City Modeling Framework

Two survey video-streams recorded by a stereo-head and annotated with GPS/INS measurements, form the input to the proposed framework. First, a Structure-from-Motion (SfM) algorithm computes a camera pose for each image. Subsequently, these poses are used to generate a compact 3D city model with textures extracted from the image material. Due to the extent of the inner-workings of the framework and the limited space available here, we refrain from explaining in detail the workings of well-known algorithms such as Structure-from-Motion pipelines [8] and dense stereo [11], but limit the discussion to the specific changes which were made to allow for high processing speeds and a compact 3D model representation.

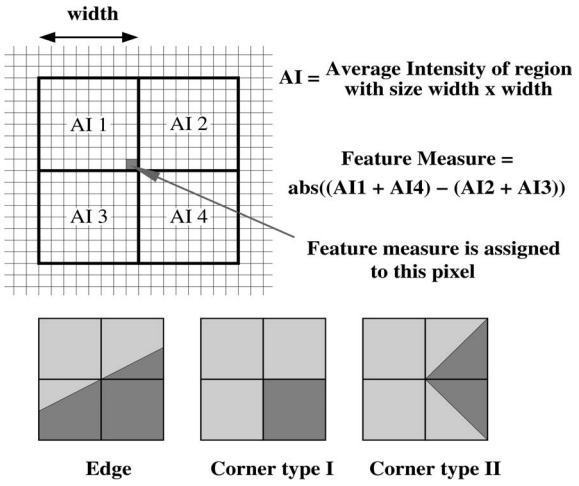

Figure 2. The measure used to detect image features.

3.1 Structure-from-Motion

This component computes a camera pose for each image as fast as possible. Our assumption is that the stereo-head's camera internals and relative pose are pre-calibrated. Therefore, computing the camera poses for only one video-stream suffices to determine the poses of both streams. Furthermore, we only process the green channel instead of all three color bands during SfM. The computationally most expensive step is feature *tracking* or *matching*. We developed a real-time feature matcher which extracts image feature points by finding local maxima of a very simple feature measure, see Figure 2. For straight edges the measure value is low. Corners of type I result in a high measure value while corners of type II have a low value. In city survey sequences, type I corners are more prevalent than type II due to the building architecture. Furthermore, in our scenario, corners of type I do not change over time into corners of type II because the camera typically does not rotate around the optical axis.

Both its simplicity and the fact that the order in which image data is used for its computation corresponds to the way image data is laid out in computer memory (Integral Images benefit from this too [14]), lead to a fast, single image pass, feature extraction algorithm which takes advantage of image caching. The extracted features are matched between consecutive images based on a fast sum of absolute intensity differences.

A classic SfM pipeline is followed. First, two images are used to determine an initial framework in which 3D points will be reconstructed. For each new incoming image, we first determine correspondences with already reconstructed features of the previous image. They allow to compute the camera pose of the new image. Subsequently, we use epipolar geometry to speed up the computation of correspondences with features of the previous image which had not yet been assigned a 3D point. A feature track is first reconstructed when the number of images through which it was tracked exceeds a threshold. Sufficient baseline between images is guaranteed by only accepting a new image when the GPS or odometry signals a sufficient movement. The 3D triangulation is performed by finding the midpoint of the shortest line segment which connects the lines of sight of the start and the end of the track. For each new processed image, 3D points are refined by re-triangulation using the only start and the new end of the feature track. During this refinement step we remember for each track the largest angle subtended by the lines of sight. Refinement is only accepted when the current triangulation angle is closer to 90 degrees than the previously stored value, to avoid a decrease in reconstruction accuracy.

To avoid memory congestion during SfM calculations the length of a feature track is limited. This limits the history of the SfM algorithm and thus allows us to write out systematic blocks of computed camera poses and 3D points to the computer hard-disk in order to free up the active RAM memory. Although limiting the track length will lead to a faster error buildup and faster drift in camera poses, this is not so disastrous since the GPS/INS image annotations help us to overcome drift by registering the final result without drift in a common world frame.

To save time a bundle adjustment routine is running in parallel with the main SfM algorithm. While the latter determines the camera pose for each new incoming image, the former refines the camera poses and 3D feature points which were 1) already written out to the hard disk and 2) registered in the world frame using GPS/INS data. The bundle adjustment routine is written so that it is iteratively performed on consecutive blocks of camera poses and 3D points in order to be able to load the data into memory and avoid congestion.

3.2 Facade Reconstruction

Reconstruction of building facades by means of passive stereo techniques is a very difficult problem. Many techniques have already been developed to compute dense disparity maps from stereo images. However, due to their computational complexity, they are not suited for processing vast amounts of data and for creating highly compact output. Therefore, we have developed an adapted dense stereo algorithm which incorporates the assumption of simple output geometry, namely that building facades are approximately all ruled surfaces parallel to the direction of gravity $\vec{g}$. This allows us to gain speed.

3.2.1 Stereo Camera Rectification

SfM resulted in camera parameters for each incoming stereo image pair. The vector $\vec{g}$ can be found by looking at the vanishing points in the images. Without loss of generality, we will assume here that the baseline of each stereo set is perpendicular to $\vec{g}$, enabling us to rectify each stereo pair such that the up direction of the rectified cameras equals $\vec{g}$. In practice, the rectification is not done as a preprocessing step but is performed implicitly by making use of the projective texture mapping capabilities of the graphics card (GPU) when performing texture lookups.

3.2.2 Similarity Measure

Once a stereo pair is transformed into a standard stereo setup with image size $w \times h$ ($columns \times rows$), we can define a discrete disparity search range $[0, d_{max}]$. Because the up direction after rectification equals $\vec{g}$, it can be shown that for fixed values of $x \in [0, w-1]$ and $d \in [0, d_{max}]$, the corresponding 3D points for all $y \in [0, h-1]$ form a straight line parallel to $\vec{g}$.

Using the assumption that facades are also parallel to $\vec{g}$, we can derive a robust line-based similarity measure by summing the per pixel similarity values along the y direction in image space. This results in a two dimensional similarity map $\mathbf{S}$ of size $w \times d_{max}$ where:

$$\mathbf{S}_{x,d} = \sum_{y=0}^{h-1} \min(SSD_{max}, SSD_{x,y,d}) \qquad (1)$$

with

$$SSD_{x,y,d} = SSD(imright_{x,y}, imleft_{x+d,y}) \qquad (2)$$

where SSD_{max} is a saturation value of the Sum of Squared Differences, introduced to limit the influence of possible outliers, and $imright$ and $imleft$ are the rectified images. The left and middle images in Figure 3 illustrate the use of the similarity measure, while the right image shows the similarity map $\mathbf{S}$, computed for that stereo pair.

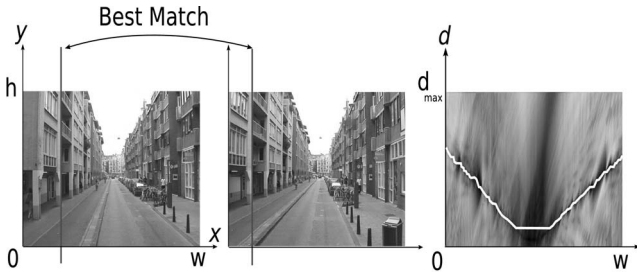

Figure 3. Left and middle: Rectified stereo pair with example of a best match, based on the similarity measure. Right: Similarity map with minimal cost path (white line).

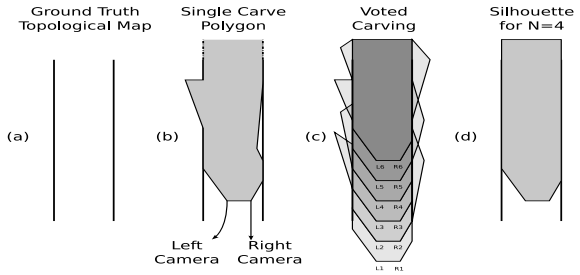

Figure 4. (a) Example of a ground truth topological map. (b) Polygon extracted for a single stereo set. (c) Voted carving. (d) Topological map from silhouette extraction.

3.2.3 Line Selection

As noted earlier, each entry in **S** corresponds to a 3D line parallel to $\vec{g}$. So by selecting a d value for each x and interconnecting them, one is able to reconstruct a ruled surface of the facades. Selecting for each x the disparity d where $\mathbf{S}_{x,d}$ is minimal, would result in a fair amount of artifacts, due to occlusions etc. Looking at Figure 3 , however, one can see that it is reasonable to apply a more global constraint, namely the ordering constraint, imposing that vertical lines in the left image and their corresponding lines in the right image should appear in the same order. This constraint can be implemented efficiently with dynamic programming which extracts a minimal cost path from **S** that satisfies the ordering constraint, shown in Figure 3.

3.3 Topological Map Generation

Since all ruled surfaces extracted from all stereo pairs are parallel to $\vec{g}$, it is possible to create a topological map by applying an orthogonal projection along the $\vec{g}$ vector. The 3D ruled surfaces now become 2D curves on the topological map. By adding the left and right camera centers, we can create a closed curve, outlining a single polygon for each stereo pair. As Figure 4 shows, these polygons can still contain some errors.

Each polygon can be interpreted as a carving area, meaning that the 2D area covered by the polygon has been marked as empty by the corresponding stereo set. In order to create an integrated topological map, one could carve out the polygons of each stereo pair and detect the silhouette of the total carved area. This process, however, is prone to errors introduced by a single incorrect polygon since it can never be recovered by other stereo pairs. Therefore we apply a voting based carving algorithm. After the topological map is initialized to a value of zero, the area covered by each projected polygon is incremented by one. Finally, only the area with a value greater than a threshold N is carved and the corresponding silhouette is extracted, see Figure 4. This results in a more robust integration algorithm.

3.4 Road Reconstruction

Facade reconstruction using passive techniques is a difficult task. Looking at the visual imagery, road reconstruction is even more so. Due to the homogeneous texture of a road, one can only determine its position by using edge information of road markings, if available at all. The correct detection of these edges however, is also tedious due to the likely presence of false edges, eg. a parked car creates false edges which lay outside the road plane. Hence, using passive techniques for road reconstruction is extremely limited. Therefore we introduce a method which interacts with the SfM in order to determine the road position.

The stereo cameras are mounted onto the survey vehicle in fixed positions. This allows us to determine the positions where the two front wheels touch the road, relative to the camera coordinate frames. As the SfM algorithm tracks the positions and orientations of the cameras, it also tracks the positions of these contact points implicitly. This results in a sparse sampling of points on the road as the vehicle moves forward. For each stereo set, the left and right contact points are connected to form a line segment which gets elongated in both directions until intersection with the facade ruled surfaces. By interconnecting the resulting line segments of consecutive stereo sets we are able to reconstruct another ruled surface, representing the road.

Looking at Figure 3, one can see that the line-based similarity measure would perform better if the integration along the y direction is limited to the facades. The reconstructed ruled surface for the road makes this possible. It allows us to discard similarity values $SSD_{x,y,d}$ for which the corresponding 3D point is positioned beneath the road.

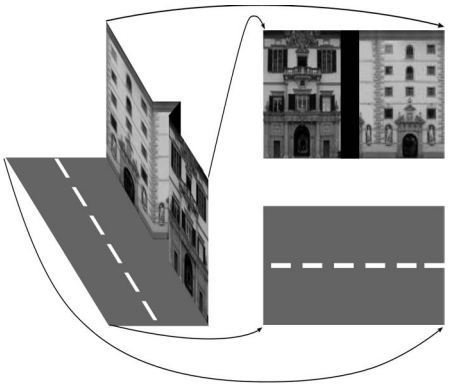

Figure 5. Each 3D line is mapped onto a column in the texture map.

Figure 6. Left: Original images. Right: Renderings of the reconstructed models from the same point of view.

3.5 Texture Generation

Once the 3D facade and road models have been constructed, we can apply textures to them by backprojecting the images onto the models. Because the facade and road models consist of line segments, we can initialize 2D textures where each vertical line in a texture corresponds to a 3D line segment, as shown in Figure 5. To ensure that images do not backproject on occluded 3D structures, we apply a commonly used technique in computer graphics, called shadow mapping.

The road and facade textures are initialized as black and the images are processed in the same order in which they were recorded. If an image is the first to be applied to a certain line segment and passes the visibility test, the black color is replaced by the projected image. If a previous processed image already projected onto the line segment, the resulting color C_{new} is a linear combination of the previously stored color C_{old} and the current texture color C_{tex} according to the equation $C_{new} = (1-B)\times C_{old} + B \times C_{tex}$ where the blending factor B is 0.5 or greater. The constraint on B ensures that newly applied images, which are closer to the model and are therefore more detailed, have a larger weight than previous images.

Because the model is composed of ruled surfaces for both the facades and the roads, we can make sure that line segments which are close in 3D space are also close in the texture map. In contrast to a random placement of the line segments in the textures, this avoids introducing high frequencies in the resulting textures and allows us to apply compression techniques such as jpeg2000, resulting in a highly compact representation of a textured 3D model.

	Image Size	Disparity Range	# Stereo Pairs	SfM (fps)	Bundle (fps)	Map Generation (pairs/s)	Texturing (pairs/s)	Data Storage	Travel Distance
City 1	360x288	64	1175	33.7	30.1	26.5	30.3	712 KB	± 500m
City 2	384x288	64	290	29.6	26.8	26.3	30.2	473 KB	± 400m
CPU				X	X				
GPU						X	X		

Table 1. Timing results.

4 Experiments

We tested two stereo sequences. Table 1 shows the statistics and timing results of each specific sequence. Note that real-time processing at 25 frames per second can be achieved by batching and pipelining the different operations. The SfM, bundling, map generation and texture application each form an independent stage in the pipeline and can therefore be executed on four computers in parallel. Also note that we can make use of both the CPUs and GPUs.

Comparisons between the original images and renderings of the reconstructed models from the same viewpoint are shown in Figure 6. The topological map generation is illustrated in Figure 7. Because the simple geometry assumptions are never perfectly met in real-life it makes no sense to investigate the reconstruction accuracy of the resulting 3D models. As our goal is to supply simple models for realistic pre-visualization of traffic situations the only measure used to judge the quality of the result is the subjective measure of realism.

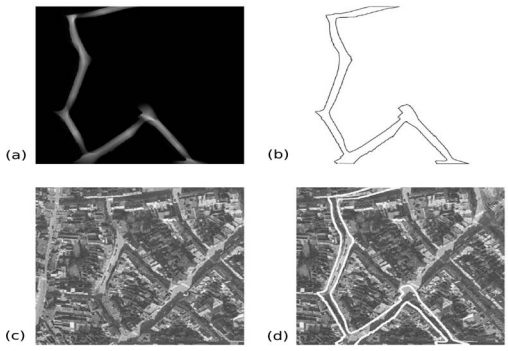

Figure 7. (a) Topological map generation by voted carving. (b) Extracted silhouette. (c) Aerial view of the corresponding area. (d) Silhouette mapped onto aerial image.

5 Conclusion

We presented a fast city modeling framework which delivers compact 3D representations to be used for easy distribution and pre-visualization of traffic situations in consumer navigation tools. It builds on a speed-optimized SfM algorithm to determine camera poses for a stereo survey sequence. Computation time is kept low thanks to the precalibrated stereo rig, the use of monochrome video signals and a simple feature extraction algorithm. Subsequently, the resulting camera poses are used by an adapted dense stereo algorithm. The latter uses the graphics card as implementation platform and incorporates the assumptions of simple output geometry. The results are compact and can be retrieved at high speed making it realistic for this city modeling framework to be unleashed on large image databases covering entire cities.

Future work will mainly focus on two separate problems. On the one hand, because the simple geometry assumptions will never completely fit the true 3D geometry of the scene, a clever way of merging textures from different images is needed to remove artifacts from the final texture. On the other hand, we can use the simplified model as a starting point for determining a more detailed 3D model when accurate 3D measurements are required.

A further increase in reconstruction accuracy and realism can be achieved by mounting additional cameras on the survey vehicles and by using view dependent texture mapping [2].

6 Acknowledgements

We wish to acknowledge the support of the European Commission project DIRAC, the K.U.Leuven Research Fund GOA, content provider TeleAtlas for the use of their data and Wim Moreau for constructing the stereo rig.

References

[1] N. Cornelis and L. V. Gool. Real-time connectivity constrained depth map computation using programmable graphics hardware. In *CVPR '05 - Volume 1*, pages 1099–1104, 2005.

[2] P. E. Debevec, Y. Yu, and G. D. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. *Eurographics Rendering Workshop*, pages 105–116, June 1998.

[3] C. Frueh, S. Jain, and A. Zakhor. Data processing algorithms for generating textured 3d building facade meshes from laser scans and camera images. *IJCV*, 61:159–184, February 2005.

[4] C. Frueh and A. Zakhor. 3d model generation for cities using aerial photographs and ground level laser scans. *CVPR, Kauai, USA*, 2.2:31–38, 2001.

[5] A. Gruen. Automation in building reconstruction, 1997. In Fritsch/Hobbie, editor, Proc. Photogrammetric Week97, Stuttgart.

[6] N. Haala and C. Brenner. Fast production of virtual reality city models, 1998. IAPRS.

[7] N. Haala, C. Brenner, and C. Stätter. An integrated system for urban model generation. *ISPRS, Cambridge*, pages 96–103, 1998.

[8] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2000.

[9] J. Hu, S. You, and U. Neumann. Approaches to large-scale urban modeling. *IEEE Computer Graphics and Applications*, 23(6):62–69, 2003.

[10] H.-G. Maas. The suitability of airborne laser scanner data for automatic 3d object reconstruction. *Third International Workshop on Automatic Extraction of Man-Made Objects from Aerial and Space Images*, pages 10–15, 2001.

[11] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47:7–42, April-June 2002.

[12] I. Stamos and P. K. Allen. 3-d model construction using range and image data. *CVPR*, I:531–536, June 2000.

[13] Y. Sun, J. K. Paik, A. Koschan, and M. A. Abidi. 3d reconstruction of indoor and outdoor scenes using a mobile range scanner. *ICPR*, III:653–656, 2002.

[14] O. Veksler. Fast variable window for stereo correspondence using integral images. *CVPR*, pages 556–564, 2003.

[15] C. Vestri and F. Devernay. Using robust methods for automatic extraction of buildings. *CVPR*, pages 133–138, 2001.

[16] G. Vosselman and S. Dijkman. 3d building model reconstruction from point clouds and ground plans. *IAPRS*, XXXIV-3/W4:22–24, October 2001.

[17] M. Wolf. Photogrammetric data capture and calculation for 3d city models. *Photogram-metric Week .99*, pages 305–312, 1999.

[18] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *CVPR*, 2003.