Editors: Lawrence Rosenblum and Michael Macedonia

From Urban Terrain Models to Visible Cities

William Ribarsky, Tony Wasilewski, and Nickolas Faust

Georgia Institute of Technology

In the novel *Invisible Cities* by Italo Calvino, Marco Polo, returning from his travels, encounters Kublai Khan. It's the end of the day, near the end of the great Khan's life, and perhaps at the end of his empire. Marco Polo tells of the cities he visited while traveling the far-flung empire. Each one is fantastic and is described in a short fable, so the whole story takes on the quality of a folktale constructed around a series of adventures. The reader can't be sure whether these cities actually exist or whether they are an illusion in Marco Polo's or the Khan's mind.

We are now faced with the possibility and, in some cases, the results of acquiring accurate digital representations of our cities. But these cities will be just as invisible as Marco Polo's unless we meet some fundamental challenges. The first challenge is to take data from multiple sources, which are often accurate but incomplete, and weave them together into comprehensive models. Because of the size and extent of the data that we can now obtain, this modeling task is daunting and must be accomplished in a semiautomated manner. Once we have comprehensive models, and especially if we can build them rapidly and extend them at will, the next question is what to do with them. Thus, the second challenge is making the models visible. In particular, they must be made interactively visible so users can explore, inspect, and analyze them.

In this article, we discuss the nature of the acquired data and how we're beginning to meet these challenges and produce visually navigable models. We've developed 3D City, a semiautomated system that supports human intervention at key points to meet the challenge of constructing complete and extended urban models from several data sources. We've already built virtual environments (VEs) for urban planning and emergency response using 3D City.

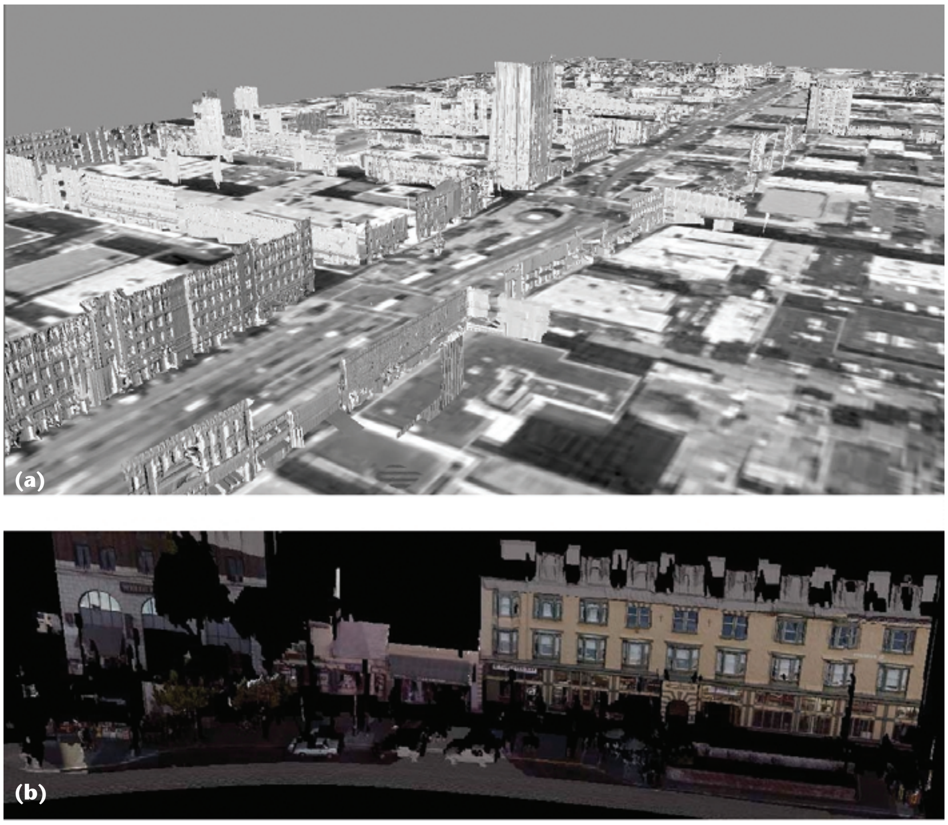
Once we've met the challenges of urban construction and visualization, possible applications include education, urban planning, emergency response, tourism and entertainment, military operations, traffic management, construction (especially large-scale projects), various geolocated and mobile services, citizen–government relations (when complex civic projects are vetted), and games based on real locations. For example, large-scale urban projects now require coordinated efforts by affected neighborhood groups; the business community; and city, state, and federal governments. Collecting data and modeling a city as it is creates a foundation for inserting new structures, bridges, and roads. We can provide different constituencies with interactive visualizations of different model designs. Vehicle and pedestrian traffic models can be applied to the street layouts and then visualized in the city environment, helping with overall planning and with meeting government requirements (such as those from the US Environmental Protection Agency). The results will be faster, less expensive, and better planning and construction. With the appropriate apparatus for collecting and then inserting new data into existing data collections, we'll be able to always keep an urban database up to date. Any change in a building facade, move of a lamppost, or removal of a tree can be recorded.

Urban data acquisition

There are now several ways to acquire accurate location, height, and appearance information for 3D urban features such as buildings, bridges, roads, sidewalks, and trees. We can use geocorrected imagery from aerial photography and satellites at high resolution (down to a foot or less). There are ground-based close-up images of buildings, groups of buildings, or other urban features. Many of these are freely available on the Web. Airborne laser range-finding systems, such as Lidar, permit an airplane to quickly collect a high-resolution height field for a small city in just a few hours. Synthetic aperture radar (SAR) can capture similar information. Now ground-based systems using cameras, or laser range finding with calibrated cameras,[1] can collect copious amounts of accurately located, ground-level 3D detail and appearance information.

The data provided by these technologies are typically incomplete, or they provide some parts at higher detail than others. The aerial and satellite methods can provide accurate locations and footprints; they also offer height information that ranges from good to highly accurate. For example, Lidar provides heights at the resolution of inches for samples that can be as close as a foot apart. However, the sides of buildings, especially taller buildings, have much less detail. The ground-based methods can provide highly accurate 3D details on building facades and associated appearance information (from calibrated cameras). However, they often will provide less detail for the upper floors of taller build-

0272-1716/02/$17.00 © 2002 IEEE


1 (a) Overview of Berkeley, California, facade mesh. (b) Detail from mesh with textures. (Courtesy of Avideh Zakhor.)

ings (see Figure 1[1] for an example). In addition, some techniques that capture fairly complete building facades haven't been applied to extended models of cityscapes. In all methods, trees or other buildings (or cars and pedestrians in ground-based techniques) can occlude parts of the acquired model. Thus, we must use a combination of methods to derive complete models, and we must often augment these with techniques for filling holes where no acquired data exist.

Semiautomated urban construction

Our system lets us more rapidly build urban databases than with manual methods. In an initial test, we experienced more than a factor of 10 speed-up in using the semiautomated system 3D City rather than the previous manual methods for constructing a model of downtown Atlanta. The current version of our system is faster, has more capabilities, and has been used to build several other urban databases.

Our system uses imagery from various sources. Users first compile one or more overhead photos of the urban area to be modeled. If close-up oblique images are available for selected buildings, users have the option of using a close-range photogrammetry tool to create more detailed 3D models. Other buildings can be created from one or more overhead images. Users select roof corners, base corners, and shadow extents (see Figure 2), and they create a building with a flat-topped textured roof at the appropriate height (given the date and time a photo was taken). Building sides can be textured using texture images loaded into a texture library or extracted from perspective-distorted images obtained from ground-level or oblique photos. Users then export the building models into VGIS, a real-time 3D geospatial visualizer that places the urban terrain in its global setting. Here, we perform final placement and adjustment of the buildings and insert them into the VGIS database, after which users can explore and interact with the created urban terrain.

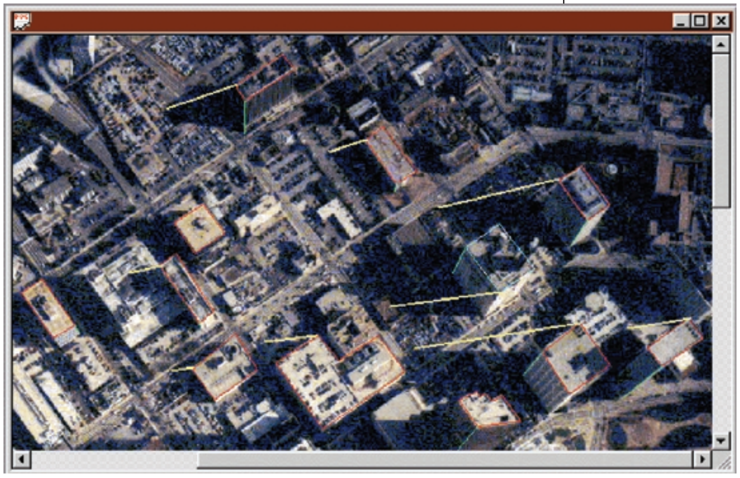
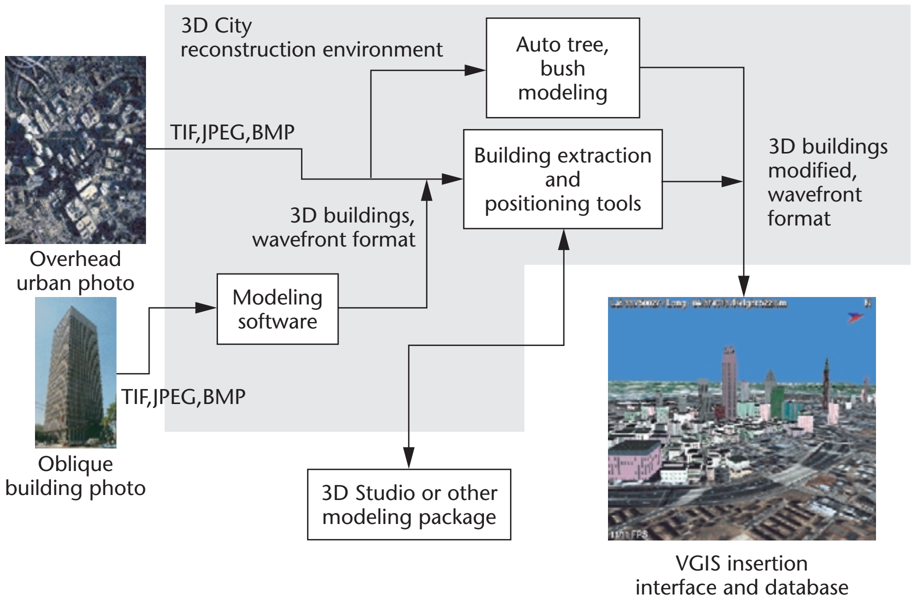
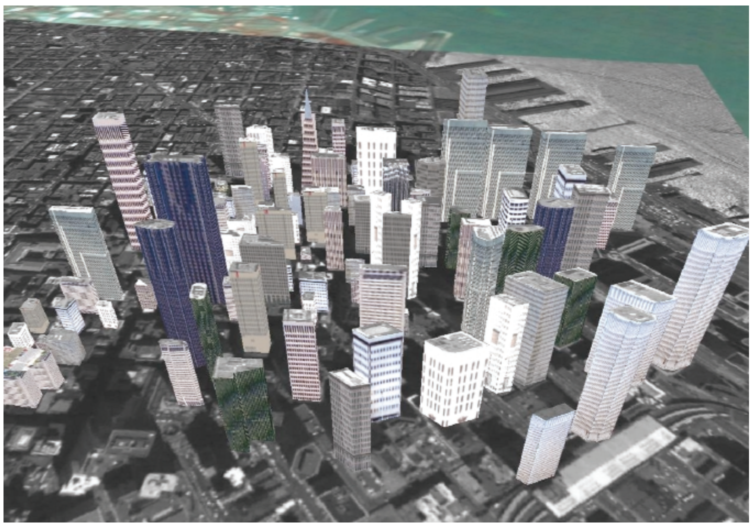

2 Building selection, anchoring, and shadow selection.

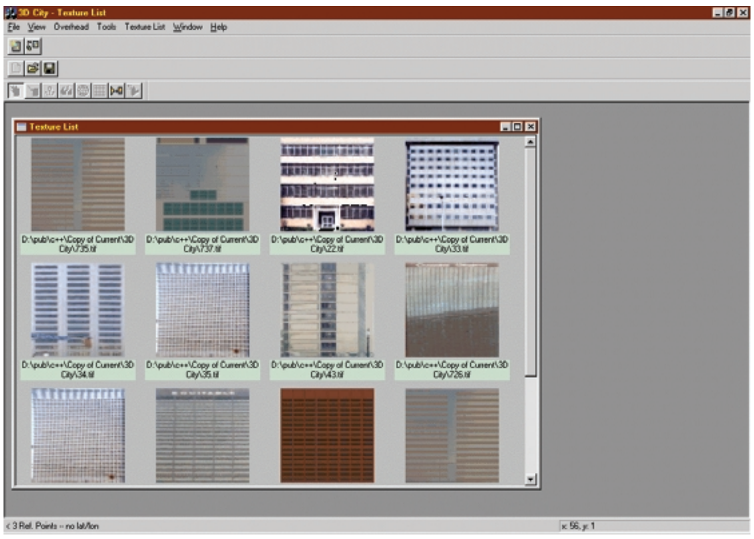
Figure 3 (next page) shows our system's overall environment. The items within the dotted box are part of the modeling system, which runs on a desktop PC. Users can bring into the system orthorectified overhead images and oblique images in various formats. Footprint, position, and height information are extracted from the overhead images, as Figure 2 shows.


3D City reconstruction environment
Overhead urban photo
TIF,JPEG,BMP
Auto tree, bush modeling
Building extraction and positioning tools
3D buildings, wavefront format
3D buildings modified, wavefront format
Modeling software
Oblique building photo
TIF,JPEG,BMP
3D Studio or other modeling package
VGIS insertion interface and database

3 Overview of the 3D City modeling environment.



4 Oblique view of San Francisco model with Transamerica Building and other landmarks.



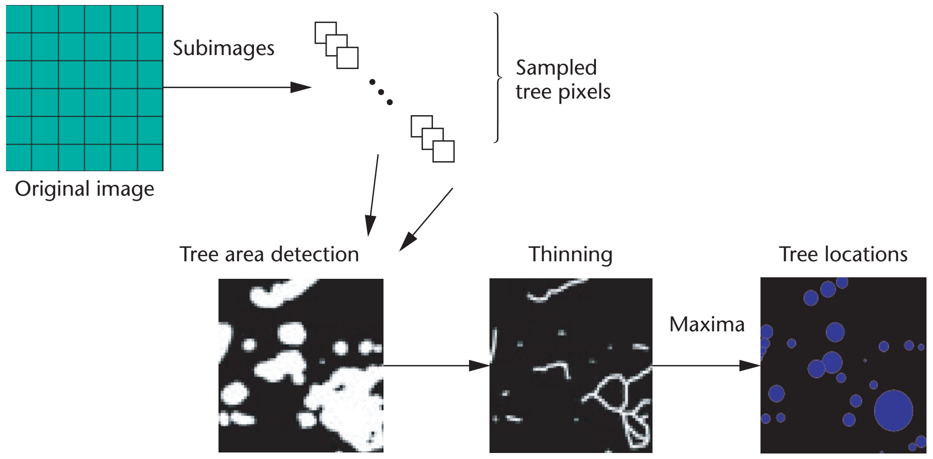
D:\pub\c++\Copy of Current\3D City\735.tif
D:\pub\c++\Copy of Current\3D City\737.tif
D:\pub\c++\Copy of Current\3D City\22.tif
D:\pub\c++\Copy of Current\3D City\33.tif
D:\pub\c++\Copy of Current\3D City\34.tif
D:\pub\c++\Copy of Current\3D City\35.tif
D:\pub\c++\Copy of Current\3D City\43.tif
D:\pub\c++\Copy of Current\3D City\726.tif

< 3 Ref. Points -- no lat/lon
x: 56, y: 1

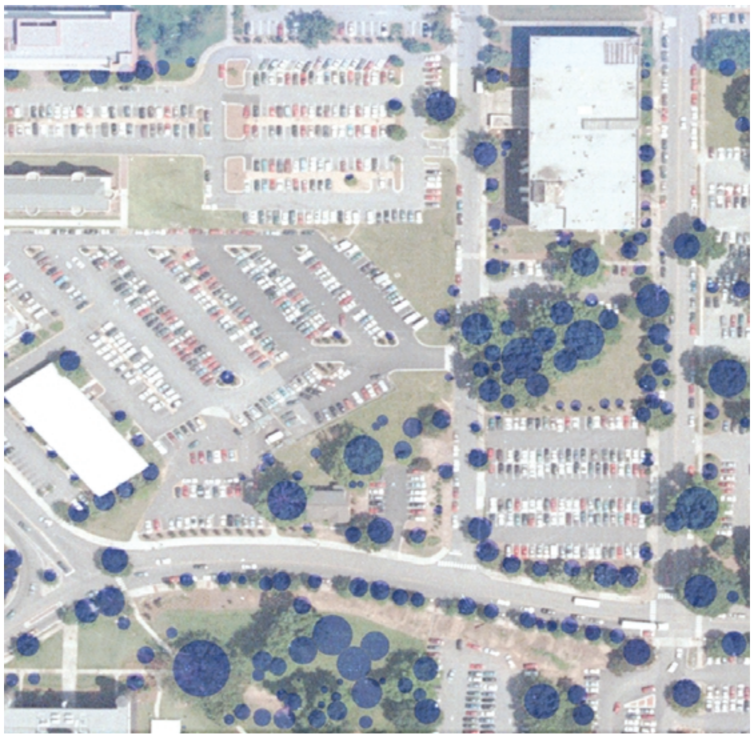
5 Texture library thumbnail window.

Sometimes shadow information is insufficient, so we can additionally resolve heights by referring to buildings of known height, counting floors, or using other factors.

We introduced photogrammetry software (indicated by the modeling software module in Figure 3) in the pipeline to provide more detailed models of landmark buildings, such as the Transamerica tower in the San Francisco model (see Figure 4). Oblique images from multiple angles provide data for the detailed model, which we position and scale using the techniques in Figure 2. A texture library (see Figure 5) provides enriched detail for building facades. After users select a texture quadrilateral from an oblique image, the software performs a projective mapping to a rectangular texture and stores it in the library. The projective mapping removes most of the perspective distortion and makes the texture look as it would from an orthogonal view. We can then choose textures in the library and apply them to selected parts of the building facades. This semiautomated method provides both particular textures (from photos of the building being modeled) and generalized textures—for example, brick fronts for filling in missing appearance information. A building roof library provides various premodeled roof structures that can be attached to the tops of buildings by



6 Process for tree and shrub identification and modeling.



7 Aerial image with modeled trees.

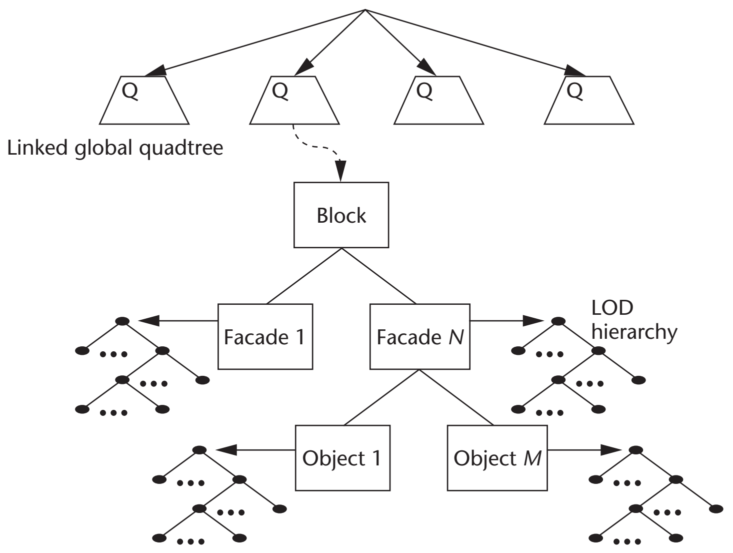
user selection to provide more realistic roofs. After modeling, items are fed into the VGIS system (lower right in Figure 3) for final positioning with respect to the rest of the cityscape, terrain imagery and elevation data, and features such as roads. The system then transforms the final city model into the VGIS internal format so that it's available for interactive visual navigation.

A separate module in Figure 3 extracts tree and shrub information from overhead images. Figure 6 illustrates the entire process schematically. The system splits a geolocated and geocorrected image into smaller (512 × 512 pixel) images. Users select tree areas by choosing tree masks—the white tree areas on the black background in Figure 6. We create the masks with standard image-manipulation software. Given this training data, the system samples pixels and extracts average color values. The software uses alternating stages of color selection and blurring to choose areas likely to be trees. Once the system has tree areas, it needs to derive locations for individual trees. A thinning technique is used to find "skeletons" representing the essential topology of the tree areas but located in the center of tree groups (where the trunks are likely to be). Tree group width is preserved in this thinning step. Tree group width maxima are used as tree position indicators. Overlapping trees are culled, and tree coordinates are derived from pixel coordinates in the geolocated subimages. Finally, 3D tree objects (height scaled according to width and colored based on sample tree statistics) are placed at these coordinates and shipped to VGIS for insertion. The system performs this process in a series of automated steps.

This approach has several advantages. After the training step the procedure is automatic, although small adjustments of the tree models may be desirable at the end. Exact tree trunk positions aren't extracted (hard to do anyway for dense clusters of trees), rather likely centers for a given canopy are found. With information about average foliage width for a given tree species and size, we can automatically generate the likely number of tree trunks and their positions. Figure 7 shows an example of the automated tree detection for a true-color aerial photograph. We plan to extend this algorithm by looking at the frequency information embedded in the tree textures. This will make the algorithm more accurate and will permit it to distinguish between different types of trees and between trees and shrubs.

Automated construction of detailed, interactive urban environments

The swelling flow of urban data acquisition makes us want to go beyond semiautomated methods. A complete



8 Hierarchical structure of 3D urban block geometry.

method would have to handle a much larger range of scales, from small 3D details (on the order of inches) on the side of a building to whole cityscapes. To attack this problem, we recently teamed with Avideh Zakhor and her group at the University of California, Berkeley. Figure 1 shows some facades acquired with their automated tools.[1] Two major problems exist. These models are generated without the means for interactive visualization as they grow in size and, as discussed earlier, they are incomplete. We've initially attacked the latter problem by inserting simplified versions of the models, with facade outlines and appearance information converted to textures, into the semiautomated methods we explained in the last section. We can then complete the models using the semiautomated tools.

To attack the problem of interactive visualization of scalably large urban data, we developed a structure with global geospatial reach.[2,3] In this method, static 3D objects (such as urban data) are inserted into a forest of quadtrees that span the Earth. A related quadtree structure organizes multiresolution terrain elevation and image data. The static object quadtree descends to the level of urban blocks. Traversal provides efficient first-order culling of urban data and sets paging priorities for bringing urban data into memory (what is in view, what is close by and may be in view soon, and what isn't likely to be in view soon). A filled quadnode contains the center of a 3D block bounding box; the block footprint can extend beyond the node's boundary. Thus, neighbor nodes out of the view frustum can potentially contain objects that extend into the view frustum. We chose a quadnode level appropriate for a block footprint of about 50 meters on a side, the dimensions of a typical city block. If a typical building is the size of a house (10 meters on a side), we would have to consider no more than tens of buildings on average for each block. We can adjust this footprint for different settings (urban versus suburban), but we haven't done this so far.

Recently we've considered the construction and interactive visualization of multiresolution block models. (See http://www.gvu.gatech.edu/datavis/research/research.html for more information and references.) A multiresolution organization and view-dependent visualization approach is necessary to achieve interactive navigation of extended, highly detailed models. Working with the Berkeley group, we take block-sized chunks of the urban facade textured mesh. We organize these into the block hierarchy in Figure 8. In this method, the block is divided into a series of facades rather than individual buildings. In dense urban environments, individual buildings are often joined together and may even be hard to distinguish as individuals. The facade structure can include free-standing buildings (as a set of facades) and related objects such as lampposts or sidewalks, which are child objects for neighboring facades (see Figure 8). Such objects will typically be part of the acquired data, as Figure 1 shows. There are numberous ways to create the multiresolution facade hierarchy; we use the bounding sphere hierarchy developed for Qsplat.[4] The sphere hierarchy is set up by starting from the bounding box for the overall model and recursively splitting vertices along the longest axis to find the children's bounding boxes (and bounding spheres). The Qsplat hierarchy was originally used for splat-based rendering. Our approach has the advantage of supporting both splat-based (point) and polygonal rendering. Trees, for example, might be best rendered using splats, but large, flat areas of facades are best rendered with polygons.

The bounding-sphere hierarchy is a fast way to organize 3D vertices on each facade. At the highest level of detail (LOD), each vertex is represented by a unique sphere. This is a leaf node. Parent nodes hold the average position of their children, and their bounding spheres encompass the child-bounding spheres. For splat-based rendering, a splat is associated with each node. For polygon-based rendering, a simplification process must be constructed that connects all the LODs. We follow the bounding-sphere hierarchy to create a simplification process because this procedure is fast and seems to produce reasonably good results. However, the procedure needs more thorough evaluation. We introduce special boundary vertices that the mthod doesn't merge during simplification (although other vertices can be merged to them). These are the final vertices and connect the facade hierarchy with the textured building structure we described in the last section. Using these results, we're now setting up a view-dependent rendering structure for interactive navigation of these highly detailed urban environments within our global geospatial system.

Current directions

We continue to work on the automated construction and organization of extended, detailed models so that they can be used for interactive visual navigation. As we begin to address even larger models (of the size of an urban downtown, for example, with thousands of buildings), we must go beyond textured geometry methods. We can combine these methods with image-based rendering (for example, to depict more distant details in a city) and develop a data organization to handle the combination of methods. Ultimately, this data structure must have a temporal component so that we can efficiently collect and store changes in the urban environment over time.

We believe that our geometric simplification methods can also be applied to the extended CAD models of cities developed by architects, construction engineers, city planners, and so forth. Up until to now, this huge resource (for example, models for much of Manhattan) has been inaccessible to interactive visualization and visual query approaches because its models are so different and so much more detailed than the models in interactive visualization. ■

Acknowledgments

The Office of Naval Research supported this work under Award No. N00014-97-1-0898. In addition, the US Department of Defense Multidisciplinary University Research Initiative (MURI) program through the Army Research Office supported this work under Grant DAAD19-00-1-0352.

References

1. C. Früh and A. Zakhor, “Fast 3D Model Generation in Urban Environments,” *Proc. Int’l Conf. Multisensor Fusion and Integration for Intelligent Systems 2001*, 2001, pp. 165-170.
2. D. Davis et al., “Real-Time Visualization of Scalably Large Collections of Heterogeneous Objects,” *Proc. IEEE Visualization 99*, Report GIT-GVU-99-14, ACM Press, New York, 1999, pp. 437-440.
3. N. Faust and W. Ribarsky, “Development of Tools for Construction of Urban Databases and Their Efficient Visualization,” *Modeling and Visualizing the Digital Earth*, M. Abdelguerfi, ed., Kluwer, Amsterdam, 2001.
4. S. Rusinkiewicz and M. Levoy, “Qsplat: A Multiresolution Point Rendering System for Large Meshes,” *Computer Graphics* (Proc. Siggraph 2000), ACM Press, New York, 2000, pp. 343-352.

Readers may contact Bill Ribarsky at the GVU Center, College of Computing, Georgia Inst. of Tech., Atlanta, GA 30332-0280, email ribarsky@cc.gatech.edu.

Readers may contact the department editors by email at rosenblu@ait.nrl.navy.mil or Michael_Macedonia@ stricom.army.mil.