Urban Site Modeling From LiDAR

Suya You, Jinhui Hu, Ulrich Neumann, and Pamela Fox

Integrated Media Systems Center
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781

Abstract. This paper presents a complete modeling system that extracts complex building structures with irregular shapes and surfaces. Our modeling approach is based on the use of airborne LiDAR which offers a fast and effective way to acquire models for a large urban environment. To verify and refine the reconstructed ragged model, we present a primitive-based model refinement approach that requires minor user assistance. Given the limited user input, the system automatically segments the building boundary, does the model refinement, and assembles the complete building model. By adapting a set of appropriate geometric primitives and fitting strategies, the system can model a range of complex buildings with irregular shapes. We demonstrate this system's ability to model a variety of complex buildings rapidly and accurately from LiDAR data of the entire USC campus.

1 Introduction

While current sensing and modeling technologies offer many methods suitable for modeling a single or a small number of objects, an accurate large-scale urban model still remains costly and difficult to produce, requiring enormous effort, skill, and time, which results in painfully slow evolution of such visual databases [1]. This problem is the main impetus for our work. One of our objectives is the rapid and reliable creation of 3D models of large-scale urban environments such as city models. Our approach is based on the use of airborne LiDAR (Light Detection and Ranging), which offers a fast and effective way to acquire models for a large environment. In urban areas, LiDAR also provides useful approximations for urban features and buildings. However, sample-rate limitations and measurement noise obscures small details and occlusions from vegetation and overhangs lead to the data voids in many areas. So, another objective of our work is to refine the acquired models to be geometrically accurate within all local details, rather than in a global average sense.

Over years, a wealth of research has appeared to address the urban site modeling problem from photogrammetry or from laser sensing data. For example, Elaksher et al. [6] proposed a system for the reconstruction of planar rooftop building wire-frames from LIDAR data. Coorg and Teller [5] constructed a large set of 3D building models by using spherical mosaics produced from accurately calibrated ground view cameras. The method can be applied to model a relatively large site area, but it is limited to simple shape buildings and does not capture the roof structure. Lee and Nevatia [2] presented a method of integrating aerial and ground view images for urban site modeling. Zhao [3] and Seresht [4] developed methods for extracting

buildings by combining color aerial images with DEM (Digital Elevation Model). CYBERCITY [13] is a commercial software package for structuring 3D objects. There are also other similar approaches or systems that use single sensor data or integrated multiple sensors, but these implementations are limited to a set of simple building elements or combinations.

In this paper we present a complete modeling system (Fig.1) that can extract a variety of complex building structures with irregular shapes and surfaces. Our approach is based on the use of airborne LiDAR data. To verify and refine the reconstructed geometry model, we present a primitive-based modeling approach that requires only minor user assistance. We have used the system to model a variety of complex buildings from LiDAR data of the entire USC campus. The results indicate that our system is suitable for producing large-scale urban models at modest cost.

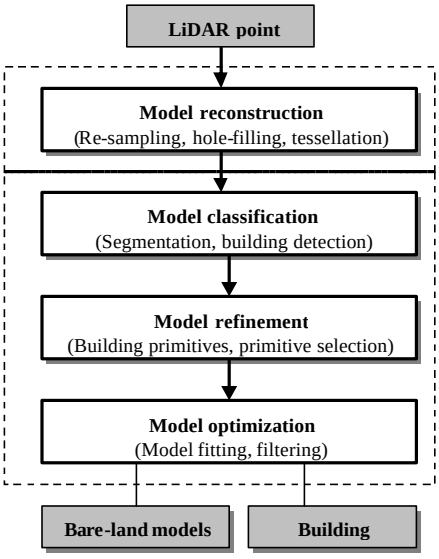

Fig. 1. Algorithmic structure and work flow of our modeling system

3 Model Reconstruction From LiDAR Data

A LiDAR sensor system permits an aircraft flyover to quickly collect a height field for a large environment with an accuracy of centimeters in height and sub-meter in ground position (typical) [14]. Multiple passes of the aircraft are merged to ensure good coverage. Due to its advantages as an active technique for reliable 3D determination, LiDAR has become a rather important information source for generating high quality 3D digital surface models.

With the cooperation with Airborne1 Inc. [14], we acquired the LiDAR model of the entire USC campus and surrounding University Park area. The end result is a cloud of 3D point samples registered to a world coordinate system (ATM - Airborne Topographics Mapper). We project and re-sample the points onto a regular grid (user defined resolution) to produce a height field or range image suitable for tessellation.

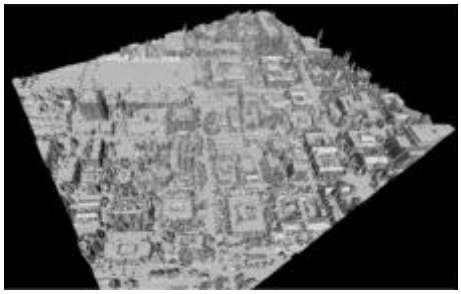

Fig.2. Reconstructed 3D mesh model of entire USC campus

Due to laser occlusions and the nature of the material being scanned, there are lots of holes in the range image without height measurements. We perform the hole-filling operation by directly interpolating the depth values in the range image in order to preserve the geometric topology of model. To preserve the edge information, we utilize an adaptive-weighting neighborhood interpolation. The interpolation weights are determined by an inverse function of the distance between the neighbor points and the point to be interpolated. The window size of interpolation is adaptive to the surface-hole size. When the surface-hole size is only a few points, a small window is used that contains the close neighbors for weighing interpolation. For the large holes, the window size is increased to ensure sufficient points for interpolation.

Triangle meshes are used as the 3D geometric representation as they are easily converted to other geometric representations; many level-of-detail techniques use triangle meshes; photometric information is easily added with texture projections; and graphics hardware supports fast rendering of triangle meshes [11]. We have tested several tessellation methods including the closest-neighbor triangulation and Delaunay triangulation, and found that the Delaunay triangulation appears to be superior to preserve the topology and connectivity information of the original data.

The whole processing of model reconstruction is fully automatic. The system allows a user to select any portion of the input data to reconstruct a 3D mesh model under defined re-sample resolution. Once the parameters of data size and re-sample resolution are set, the system automatically performs the steps to process the 3D point cloud and outputs the reconstructed 3D mesh model in VRML format. Fig. 2 shows the reconstructed model of the entire USC campus and surrounding University Park area at original sample resolution.

4 Urban Model Classification

To extract the buildings from the reconstructed 3D mesh model, the points of the mesh model have to be classified according to if they belong to terrain, building, or something else. In our system, we classify the original LiDAR model into two categories: buildings, and bare-land. The building subset is a collection of the building models represented as the parametric forms, while the bare-land subset is the reconstructed 3D mesh model with the buildings removed.

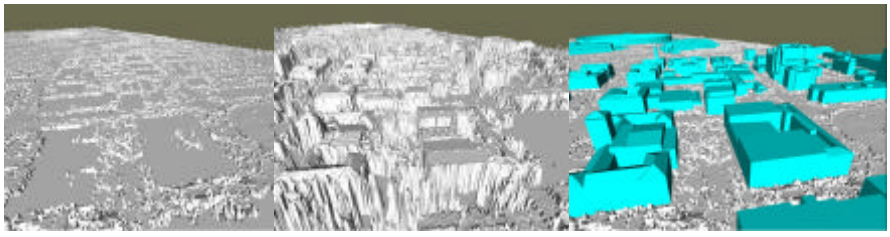

Fig. 3. Classifying the LiDAR model as two categories: (left) bare-land, and (middle) buildings. The extracted buildings are very rough that there are many artifacts remained around buildings. The initial classification has to be refined in order to remove the undesired areas to improve its utility and visualization value (right)

The classification approach is conducted based on an obvious fact: the objects, which have the height above a certain value, must be either vegetation or buildings. So, by applying a height threshold to the reconstructed 3D mesh data, we can create an approximate building mask. The mask is applied to filter all the mesh points, and only those masked points are extracted as building points. Fig. 3 illustrates the results of applying the approach to classify the USC campus mesh mode as the bare-land (Fig. 3 left) and the building areas (Fig.3 middle). As we can see, the extracted building subset is very rough that there are many artifacts remained around buildings. The initial classification has to be further refined in order to remove the undesired areas. Our strategy is to use an accurate geometry model to fit the building mesh data to produce a constrained CG building model. Once we obtain the refined building models with accurate geometry, we can easily remove those artifacts from the initial classification by combining the geometry shape cues. Fig. 3 (right) illustrates the accurate classification of the bare-land and the buildings embedded in the land.

5 Model Optimization and Refinement

Our model refinement is a primitive-based approach. We divide a complex building into several basic building primitives and model them using a parametric representation. As the models from constructive solid geometry allow the composition of complex models from basic primitives that are represented as parametric models, our approach is quite general. Also, as the type of primitive is not limited, may contain objects with curved surfaces, so the flexibility of model combinations is very high, hence we can model a range of complex buildings with irregular shapes and surfaces by combining appropriate geometry primitives and fitting strategies.

5.1 Building Primitives

Based on the shape of a building roof (flat-roof, slope-roof, dome-roof, gable-roof, etc.), we classify a building section into one of several groups, and for each group we define a set of appropriate geometry primitives, including the standard CG primitives such as a plane, slope, cube, polyhedron, wedge, cylinder, and sphere, etc., and high-order surface primitives such as ellipsoids, and superquadrics. These geometry primitives are the basic units used for building construction. They also can be combined with each other to form more complex new primitives. Fig. 4 illustrates a set of building primitives and their relationships defined for modeling a complex building.

A high-order surface primitive is useful to model irregular shapes and surfaces, such as classical dome-roof buildings and a coliseum or arena. Superquadrics are a family of parametric shapes that are mathematically defined as an extension of non-linear general quadric surfaces, and have the capability of describing a wide variety of irregular shapes with a

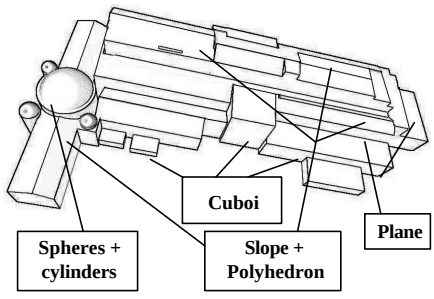

Fig. 4. Geometry primitives used for representing a building model

small number of parameters [9]. In our work, we use them as a general form to describe all the nonlinear high-order primitives, as defined in (1).

$$r(h,w) = \begin{bmatrix} a_1 \cos^{e_1} h \cos^{e_2} w \\ a_2 \cos^{e_1} h \sin^{e_2} w \\ a_3 \sin^{e_1} h \end{bmatrix} \quad \begin{matrix} -p/2 \le h \le p/2 \\ -p \le w < p \end{matrix} \tag{1}$$

where e_1 and e_2 are the deformation parameters that control the shape of primitive, and parameters a_1, a_2 and a_3 defines the primitive size in x, y and z direction, respectively. By selecting different combination of these parameters, superquadric can model a wide variety of irregular shapes, and also many standard CG primitives as well.

Once defined, each building primitive is represented as a parametric model. The parametric model describes the primitive by a small but fixed set of variable parameters. The number of parameters needing to be specified depends on the properties of each primitives and the knowledge assumed for the model fitting. For example, a generic plane in 3D space can be represented as

$$z = ax + by + c \tag{2}$$

which has 3 parameters need to be estimated. However, in the case of slope-roof fitting, we may reduce the parameters from 3 to 2 by setting either parameter *a* or *b* to zero. This is based on the observation that if a building's orientation is nearly parallel to the *x* or *y* axis of a defined world-coordinates, then either parameter *a* or *b* will be close to zero for most buildings, ie. the roof of a building usually has only one slope along the *x* or *y* direction. We use the term "zero x/y slope" to indicate the constraint for slope-roof fitting. Similar constraints can also be established for other primitives, which are based on our observation and applications. We will details those in the following sections.

5.2 Primitive Selection

As we classify the building sections into several groups, in which appropriate building primitives are defined, we have to provide the system the information to indicate the selected building section and associated type. We term the building section being processed as the "Element of Interest" (EOI) that is a square area roughly bounding the building section. Currently, the EOI information needs to be provided by a user in that only few mouse-clicks are required. Once the user input is provided, the system automatically segments the building borders and surface points, and uses the indicated building primitive to fit the building mesh model. The amount of user interaction depends on the type of building primitives associate with the group. For example, the cube primitive is determined by two points and an orientation. So, this primitive fitting needs only two user mouse-clicks to indicate two diagonal-points on the roof surface. In most cases, 2 or 3 user mouse-clicks are sufficient. The constraints established for primitives also reduce the amount of user interaction.

5.3 Primitive Fitting

The following describes the most commonly used building primitives, including their mathematical parametric representations, and examples of building fitting. The detailed algorithm procedures for each primitive are also described.

Plane Primitives. The flat-roof is a typical roof type of man-made buildings, which can be modeled using the plane-primitive group, including 3D plane, cuboids, polyhedron, and the combined primitives such as hollow-cuboids. They all share the same property that the depth surface can be described by equation (2). A 3D plane primitive is usually determined by two reference points and an orientation. If we align the building's orientation to the global direction that is defined in our working coordinates, we can reduce the specified parameters to 2, i.e. each plane is specified by two diagonal-points.

After the user indicates (mouse clicks) the two reference points, the system automatically estimates all corners of the building roof based on the global direction. The estimated corner points are then used for detecting the roof edges using a depth discontinuity constraint. We proposed an improved 8-neighbors connectivity algorithm to detect building edges. First, we used the geometry connectivity information of Delaunay reconstruction to track the connected edge points. Only those edges that lie along the Delaunay triangulation are accepted as the possible edge points. Second, we utilized a depth filter to constrain the detected edges. The depth filter is applied to all the possible edge points, and only those points having similar depth values as that of the defined reference points are passed as correct edge points. Once the roof borders have been extracted, we parameterize them using least-square fitting, and then the roof corners are refined again based on the fitted roof borders.

Plane depth fitting is performed on all the surface points inside the roof border. The depth discontinuity constraint is used for surface segmentation. We opted not to use surface-normal information due to its sensitivity to noise. Our experiments show that the depth discontinuity constraint performs well for surface segmentation. After segmenting the surface points, the plane least-square fitting is applied to the depth values of those points, and the best fitting is the height of the refined surface.

Slope Primitive. Slope is a special case of the plane with non-zero horizontal or vertical normal direction. Similar to the plane primitive, a sloped roof with rectangular edges is also extracted with two reference points using the plane fitting method. The depth fitting for sloped surfaces, however, is more complex. A 3D plane defined in (2) has 3 parameters to be estimated, where the two parameters a, b, represent two slopes in the x and y directions, and the parameter c is an offset. But we can reduce the parameters from 3 to 2 based on the "zero x/y slope" constraint. In the case of a building does not meet the condition, we perform an orientation alignment to orient the building to the reference direction. The least-square method is also used for parameter estimation, using all the surface points inside the detected roof borders.

We observe the fact that most roofs of real buildings usually have two symmetric slopes. To facilitate this structure, we combine the two connected slope primitives to form a new primitive: *roof*. In this case, three reference points (rather than four if we model the two slopes separately) are need for parameter estimations: two on the slope edges, and one on the roof ridge. The surface points of the two symmetric slopes are segmented using the above method. The least-square fitting is performed on the depth values of the segmented surface points for each of the two slope primitives. The

accurate roof ridge is computed based on the intersection of the two modeled slope planes.

Cylinder Primitive. Surface fitting of a generic cylinder is a nonlinear optimization problem. However, we observe the fact that most cylinder primitives in buildings have an axis perpendicular to the ground. Based on this constraint, we can eliminate the rotation parameter from the estimate to simplify the primitive as a vertical cylinder for circle-roofs.

The roof extraction and surface segmentation are similar as the plane case using the depth discontinuity constraint. Two concentric circles are defined for segmentation: the inner circle for roof border detection, and the outer circle for surface point segmentation. Three parameters are required for specifying the concentric circles: one for the circle center and two for the radius. To guarantee there are enough surface points for accurate segmentation and model fitting, the defined circles should cover all the possible surface points on the rooftop.

To achieve an accurate boundary reconstruction from the ragged mesh data, we defined two filters to refine the detected edges: a depth filter constraining the edge points having similar depth values as that of the defined center, and a distance filter constraining the edge points to being inside of the estimated circle. The depth filter is similar as the one we applied for plane primitives, but using the circle center's depth value as a filtering threshold. The distance filtering is a recursive procedure. Using the detected edge points, we first fit them to the circle model to obtain initial estimates of the circle center and radius. We then use these initial estimates to filter the detected edges. Any edge points whose distance to the circle center is less than a threshold will pass the filtering. The distance deviation is used as a filtering threshold. After the distance filtering, the refined edge points are used recursively to estimate a new border parameter.

Sphere Primitive. The dome-roof is a very popular roof type in classical buildings. A simple dome-roof can be modeled as a sphere shape, but more complicated tones may need high-order surfaces to represent them. Similar to the cylinder primitive, the surface of a sphere is also a quadric surface.

To detect the roof border and surface points, two reference values are needed: one for dome-roof center and another one for roof size. To guarantee enough surface points for accurate segmentation and fitting, the defined area should cover all the possible points on the roof surface. Since the section-projection of a sphere is a circle, the methods for sphere-roof detection and surface segmentation are almost the same as those we used for the cylinder primitive, except for not using the depth filtering as we don't have the sphere center in 3D space.

The model fitting is performed on all the segmented spherical surface points. As in the cylinder case, the distance constraint has also been recursively used to achieve an accurate model reconstruction. The sphere primitive can also be combined with other type primitives. The most popular usage is the sphere-cylinder combination. Another use of sphere primitives is their use as a preprocessing step for high-order surface fitting. High-order surface fitting normally is a non-linear problem. Appropriate selection of initial estimates is vital to guarantee a convergence to an optimal solution.

High-order Primitives. The standard CG primitives have limited capability to model complex objects. One of the innovative features of our system is it supports high-order modeling primitives to facilitate irregular building structures. Superquadrics are a family of parametric shapes that are mathematically defined as an extension of non-linear generic quadric surfaces. They have the capability of describing a wide variety of irregular shapes with a small number of parameters. We use the superquadric as a general form to describe all the nonlinear high-order primitives.

As an example of applying the high-order primitives to model irregular building shapes, we describe the steps to model the Los Angeles Arena with an ellipsoid primitive. The ellipsoid is a special case of superquadrics with the deformable parameters $e_1 = 1, \quad e_2 = 1$.

1. Object segmentation

The region-growing approach [12] is employed to segment the irregular object from its background. Given a seed-point, the algorithm automatically segments the seeded region based on a defined growing rule. In our implementation, the surface normal and depth information are used to supervise the growing procedure.

2. Initial surface fitting

To guarantee a converged optimal solution, an appropriate initial value is required for the Levenberg-Marquardt (LM) algorithm [10]. A sphere primitive fitting is used for system initialization.

3. *High-order surface filling*

Once initialized, the system fits the ellipsoid primitive to the segmented surface points using LM algorithm. In this Arena modeling example, the algorithm needs 606 iterations to converge to the correct solution. Fig. 5 shows the final fitting result. The refined model appears to represent the LiDAR data very well although we have yet to survey the actual building and compare its measures to our refined model dimensions.

6 System Evaluation

We have implemented a prototyped modeling system. Based on our on-hand LiDAR dataset, we have modeled the entire USC campus and surrounding University Park area including the Coliseum, LA Arena, museums, and gardens (Fig. 5). The system has been tested using a range of different building structures. For example, Fig. 6 shows the model of the LA Natural History Museum across the street from USC campus. Note the inclusion of slanted roof segments and domes makes this facility very complicated to model. Our system allows users to create the building models in a few minutes by selecting a few points in the LiDAR data.

Due to lack of actual building measurements, quantitative evaluation of the modeling accuracy is difficult. In our work, we use two strategies to evaluate our modeling system. The first strategy is to verify the reconstructed model dimensions by embedding the model in the original LiDAR data. As the LiDAR data is physically acquired from the real world, it appears to represent the real structures very well. We

have used the strategy to evaluate the reconstructed accuracy with every primitive we proposed in the system.

The second strategy is to use imagery geo-referencing to verify the accuracy of the geometry model. We evaluated our system using several imagery sources including aerial photographs, ground image/video captured with high-resolution digital cameras, and terrain maps. By projecting those images onto the geometry models, we can immediately observe the errors resulting from model reconstruction. This strategy allows us to evaluate the accuracy of very fine building structures.

Acknowledgment

This work was supported by a Multidisciplinary University Research Initiative (MRUI) on "Next Generation 4-D Distributed Modeling and Visualization". We thank the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, for their support and facilities. We thank Airborne1 Inc. for providing us with the USC campus LiDAR data. Our thanks also go to HP, NVidia, Intel, and Microsoft for their supports.

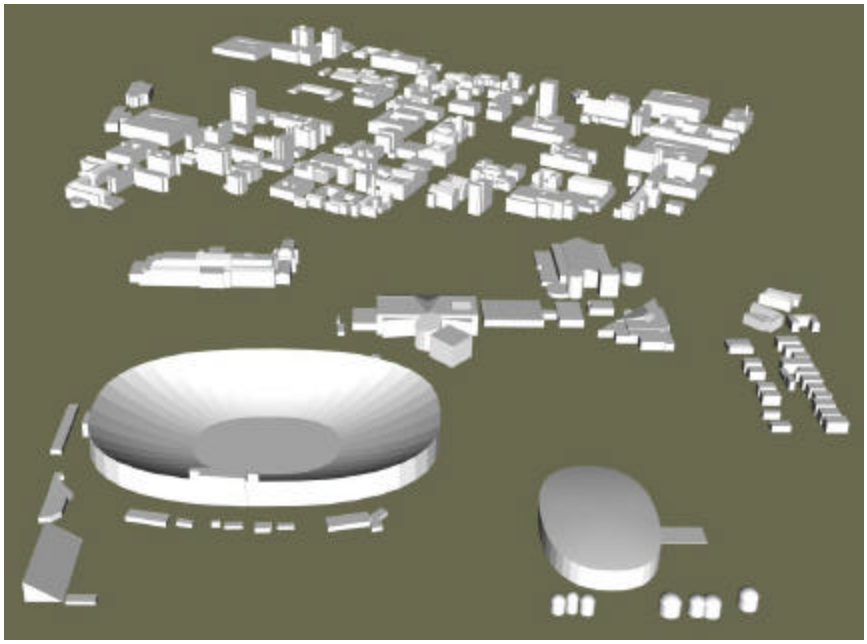

Fig.5. Complete refined models of USC campus and surrounding University Park area. The model was created in two days by author using the proposed system.

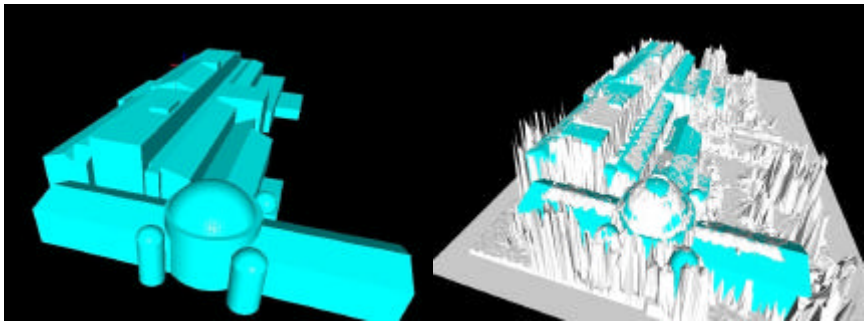

Fig.6. System performance evaluation by embedding the refined model in the original LiDAR data: (left) refined model of LA Natural History Museum, and (right) the refined model is embedded in the original mesh data. The refined model appears to represent the LiDAR data very well.

References

1. W. Ribarsky, T. Wasilewski, and N. Faust, "From Urban Terrain Models to Visible Cities", *IEEE Computer Graphics & Applications*, Vol. 22, No. 4, 2002.
2. S. C. Lee, S. K. Jung and R. Nevatia, "Automatic Integration of Façade Textures into 3D Building Modelings with Projective Geometry Based Line Clustering", *EUROGRAPHIC'02*, 2002.
3. Zhao, B. and Trinder, J., "Integrated Approach Based Automatic Building Extraction", *19th ISPRS Congress*, Book 3B, pp. 1026-1032, Amsterdam.
4. Seresht, M. and Azizi, A., "Automatic Building Recognition from Digital Aerial Images", *19th ISPRS Congress*, Book 3B, pp. 792-798, Amsterdam.
5. Coorg, S. and S. Teller, "Extracting Textured Vertical Facades from Controlled Close-Range Imagery", *CVPR*, 1999, pp. 625-632.
6. A. Elaksher, J. Bethel, "Building Extraction Using Lidar Data", *ASPRS-ACSM Annual Conference and FIG XXII Congress*, April 22-26, 2002.
7. Haala, N., C. Brenner, *"Generation of 3D city models from airborne laser scanning data"*, the 3rd EARSEL Workshop on LIDAR Remote Sensing on Land and Sea, Tallinn, 105-112,1997
8. A. Fruh and A. Zakhor, *"3D Model Generation for Cites Using Aerial Photographs and Ground Level Laser Scans"*, IEEE Conference on Computer Vision and Pattern Recognition, *2001*.
9. Suya Y., Ulrich N., *Automatic Object Modeling for 3D virtual Environment*, Proceedings of NMBIA, pp.21-26, 1998.
10. William, H., Saul, A., William, T., and Brian, P., *"Numerical Recipes in C", Cambridge University Press, 1992.*
11. Foley, J. D., Van Dam, A., Feiner, S. K. and Hughes, J. F., "Computer Graphics: principles and practice", Addison-Wesley, Reading, Massachusetts, 1990.
12. Richard E.W. and Rafael C.G., "Digital Image Processing", Prentice Hall PTR, 2nd Edition.
13. *http://www.cybercity.tv*
14. *http://www.airborne1.com*