Eurographics Italian Chapter Conference (2006)
G. Gallo and S. Battiato and F. Stanco (Editors)

Recovering 3D architectural information from dense digital models of buildings

A. Spinelli, F. Ganovelli, C. Montani, R. Scopigno †

Abstract

In recent years the progress of 3D scanning technologies and the consequent growing commercialization of scanners opened a large spectrum of opportunities for many professionals. In particular, architects and engineers may access to a digital model of a building without having to model it using a CAD software. On the other hand, there are two important differences between the digitized and the handcrafted model. The first is the absence of interpretation. The digitized model is merely a set of polygons that describe, possibly in a very accurate manner, the scanned object. It does not provide the user with any other information about what a surface is (a wall, a window, an arch etc.) that, conversely, can be incorporated during the editing in a CAD session. The second difference is excess of realism. In the digitized models the are no planar walls, no right angles, no straight edges, simply because they are not, at the millimetric scale. Unfortunately, if a model must be used in a FEM simulation, for example, a CAD like model would be required. This paper describes an application framework and some techniques that have been implemented to help a non computer-graphics user in handling digital models of buildings acquired using 3D scanning. The techniques permit to visualize efficiently the models independently from their size, recover 3D information (measurements, sections, . . .), extract geometric features and fit high level geometric primitives.

Categories and Subject Descriptors (according to ACM CCS): I.4.8 [Scene Analysis]: Range data
Keywords: 3D scanning, feature extraction, data fitting.

1. Introduction and previous work

The increasing availability of 3D range scanning devices, the development of software more and more efficient and user-friendly for the creation and manipulation of complex 3D digital models and, last but not least, the drop of the scanning technology costs are the main reasons of the recent fast proliferation of scanning campaigns for the acquisition of the shape of real objects.

Along with other application fields, 3D range scanning is more and more used in architecture. However, the output of 3D scanning cannot in general be used as it is, because of the following characteristics:

- the size of the 3D dataset is often very large and this limits the importing of the data into the more used and known CAD systems;
- the 3D range scanning data, organized in clouds of points or triangle meshes, describe the surface of the investigated object but nothing of the available information id directly useable by a professional. The data do not present edges, nor project lines, nor geometric features, nor high level geometric primitives;
- the architectural design is based on the use of exact and measurable geometric primitives. It's somewhat paradoxical to observe that the *excess of realism* of the 3D range data represents a limitation for efficient measurements: no planar walls in the digital model, no right angles, no straight edges, and so on.

For these reasons, the application software we are developing at the Visual Computing Lab and which is summarily described in this paper tries to give a complete answer to these open problems and to fill the gap between the architectural 3D range scanning and an efficient use of the data by the professionals.

The tackled problems are not new and there exist many different solutions in the scientific literature. However, the

† Visual Computing Laboratory, ISTI-CNR, Pisa (I) email: name.surname@isti.cnr.it

existing solutions do not face the problem in an integrated manner but they try to solve one or more aspects of the problem itself.

Let us examine some of the main aspects.

The 3D range scanning of buildings, archeological sites, or large industrial implants is generally carried out by means of time-of-flight laser scanning devices. The time-of-flight technology [CM02] permits to acquire in reasonable times (about 1000 samples per second) and good accuracy (6-8 millimeters with 8-10 millimeters of spatial resolution) the description of the surface of buildings. The typical working distance (device-target) is between 20 and 100 meters.

The 3D range scanning is not the only available technology for the recovering of 3D information of buildings [BSZF99, SB03]. The classic photogrammetric approach can be extended by means of model-based reconstruction methods in order to obtain the detailed reconstruction from close-range images. The 3D points obtained through image matching are generally segmented into a coarse polyhedral model with robust regression algorithm, then the geometry of the model is refined with predefined shape templates in order to automatically recover a CAD-like model of the building surface. Reprojection of the 3D shape templates is used to optimally fit their parameters to the image information.

In order to make the use of the 3D model more efficient and to reduce the amount of information to be handled, geometric simplification techniques can be applied [CMS97, Lue01, LT99]. Geometric simplification algorithms aim at removing redundant information and replacing sets of (quasi) planar adjacent triangles with larger ones. This ensures the lightening of the model but not the characterization of the interesting geometric features.

A radical improvement towards an efficient use of 3D architectural range data can be only obtained by means of feature extraction techniques.

However, feature extraction within an irregular 3D point cloud or triangle mesh from 3D scanning is difficult, mainly because it is not obvious if we are in front of a change of the curvature corresponding to the separation lines between two adjacent features rather than local noise which is a common drawback in the time-of-flight laser range scanning. Features extraction techniques can be broadly classified into two large groups. The first is the automatic features extraction from the 3D triangle mesh or cloud of points (in practice the automatic location of the lines separating homogeneous regions of the surface): these techniques usually rely on on discrete local curvature operators to detect features [OBS04, WB01, HHW05] and all the effort is somewhat spent to make the algorithm insensitive to the noise. We devised an ad hoc strategy based on variational shape approximation [CSAD04]. This approach finds a partition of the mesh in k planar regions by a region growing algorithm. At the end of the process, the lines separating the different regions characterize ridges and valleys on the model, i.e. the features searched for.

The second group is the fitting to the 3D data of high level geometric primitives. In this case, the user chooses the primitive to be fitted (arches, cylinders, boxes, etc.). Methods exist for the least-squares fitting of spheres, cylinders, cones, and tori to 3D point data, and their application within a segmentation framework [MLM01, Bes88]. We developed a software architecture to support the fitting of generic primitives with a plug in system, using point sampling of the primitive to fit and non linear minimization.

2. Our Framework

As mentioned in the introduction, the tool we provide in out application can be grouped into two groups: **measurements and utilities** and **reverse engineering**.

In this Section we will give details about the most important aspects of the implemented software tools.

Before proceeding further it has to be underlined that a spatial indexing data structure is used to carry out most of the operations performed on the model. A 3D indexing data structure is initialized at the mesh loading time. Each element of the structure, a regular 3D grid, holds references to faces of the mesh standing on it. The data structure allows to speed up the location of the primitives to be used in the different geometric operations.

2.1. Measurements and Utilities

Among the facilities the program puts at user's disposal sections and high resolution rendering have to be pointed out. In order to compute the **sections** between a plane and the 3D model the user can directly use the analytic expression of the plane normal and a distance from the origin of the coordinate axes or she/he can, more simply, select three points on the model: the sections can be obtained for equidistant parallel planes as well. The vectorial *edgemesh* data structures, corresponding to the sections, can be exported in multiple common formats. Some examples of multiple parallel sections are shown in Figure 1.

A common and diffuse need in architecture is the generation of **snapshots of the scene** (prospects, views) to be printed on large format output devices (plotters). Our system includes the generation of high resolution snapshots. The resolution is not limited to the screen resolution because the image plane defined by the user is automatically subdivided into parts by means of a regular grid. For each grid element, the scene is rendered at the maximum resolution available for the frame buffer in use. The final image is built by simply collocating the partial views in the right position. Moreover, the user can easily select the appropriate projection type.

This tool allows the user to **select regions** on the model.

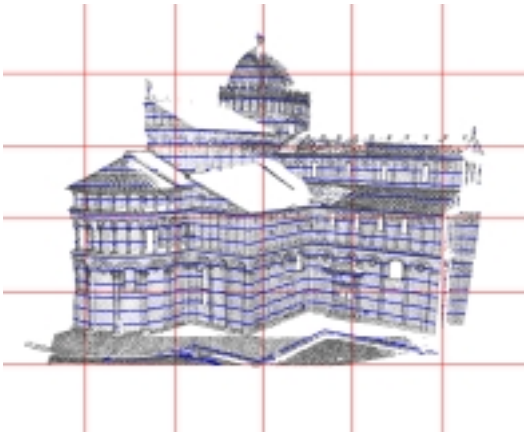

Figure 1: *Multiple sections between horizontal parallel planes and the model of the Dome of Pisa*

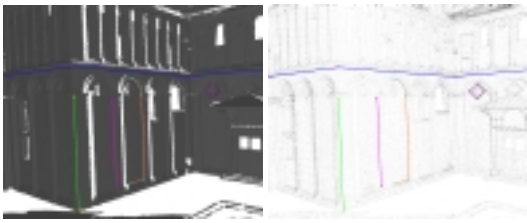

Figure 2: *Drawing 3D multiple segments on the digital model (top left). The selections are meshes themselves (top right)*

Although the task is apparently simple, it has to be noted that the solutions exploiting the mesh connectivity to *grow* a set of faces do not work well for scanned meshes, since they typically presents holes, duplicated vertices or faces, or generally non manifold situations. Furthermore, since our framework uses multiresolution, the representation (in term of set of triangles) dinamically changes .

So, instead of navigating the mesh, we create new segments and areas that are fit over it. A new segment is created by clicking over two points on the surface. The algorithm creates a segment from one point to the other, then add a new point in the middle of the segment and project it over the mesh, splitting the segment in two parts. This step is iterated until the chain of segments is closer than a given tolerance to the mesh (see Figure 2).

A new surface is created by clicking a sequence of at least three points. Then a closed polygon is generated by defining a segment between every pair of consecutive points and applying the fitting algorithm previously explained. The polygon is then triangulated to form a new mesh that we use as active contour to be fit over the mesh [KWT88]. We associate an energy value to this mesh, given by the sum of a

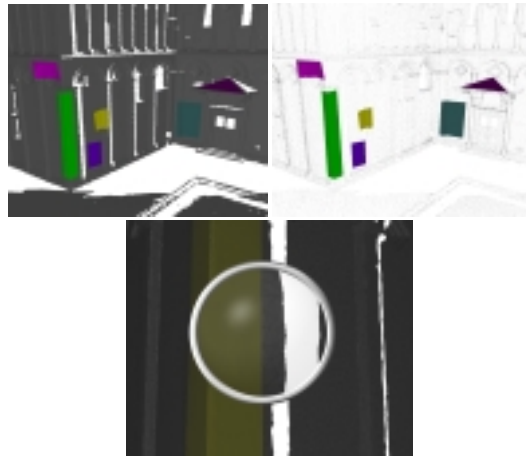

Figure 3: *Drawing 3D multiple closed regions on the digital model (top left). The selections are meshes themselves (top right). Detail of the selection (bottom).*

term that tries to preserve the shape of the triangles and a term that attracts the mesh vertices to the surface, and iteratively minimize this energy. We control the accuracy of the mesh by refining it until the approximation error is greater than a given threshold (see Figure 3).

Both the lines and the surface can be exported in a CAD application (e.g. in DXF format).

The importance of this tool is really high. The simple drawing of lines on the model permits the user to design directly in 3D starting from the acquired state of the structure. Locating regions in 3D allows the measurement of particular regions of the building which are not necessarily delimited by edges of the model; a typical example are the parts of the external surface to be cleaned or restored.

2.2. Reverse Engineering

The **automatic recovering of edges** is an important component in the reverse engineering process. The recognized features can be directly exported in a CAD application software. The method we implemented in our system adopts a region growing approach and consists of two main modules:
region growing which identifies uniform regions of the model;
edge detection which attends to the location of edges as the result of the intersection between the interpolating planes of two adjacent regions.

The first module subdivides the model into *uniform* regions; each region R is characterized by a plane Γ_R and it contains one or more adjacent faces of the model. Each face of R belongs to Γ within a user defined ε tolerance.

At the beginning k seed-triangles are randomly placed on

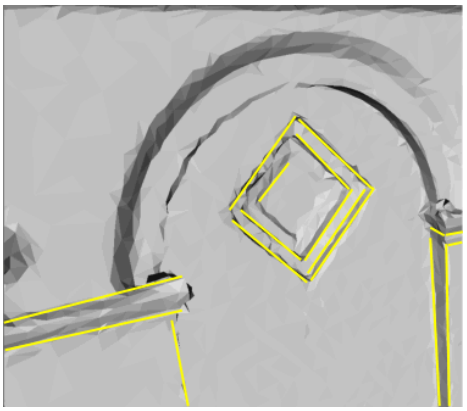

Figure 4: *Automatic recovering of edges in a 3D digital model*

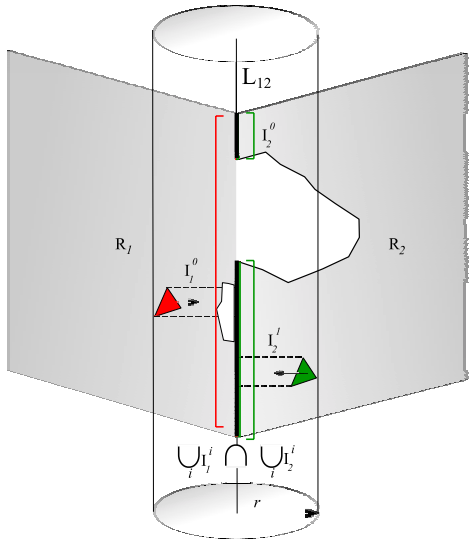

Figure 5: *Feature extraction. Red and green intervals are the projection of triangles in R_1 and R_2, respectively. The thick line if the edge returned.*

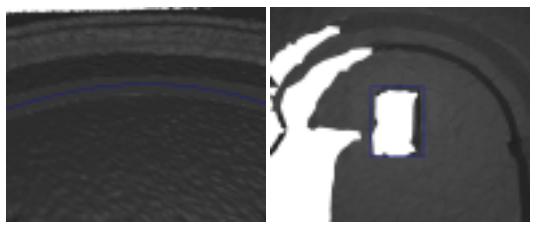

Figure 6: *Automatic fitting of high level geometric primitives. Fitting a roman arch (left) and a window (right).*

the surface, then the regions expand by associating faces adjacent to their border, picking at each step the face creating the smallest approximation error. When all the faces have been associated to the regions, every region R is fit with a new plane Γ_R and every seed is redefined as the face of the corresponding region with the smallest error with respect to the new plane. The region growing approach proceeds by visiting the 3D model in a topological way and binding new faces to the appropriate regions. We extended the algorithm as suggested in [CSAD04] by placing a new seed on the region with the greatest error, until the error is lower than a user defined threshold.

The next step of the algorithm is the edge detection. In general terms, the characteristic planes R_1 and R_2 of two adjacent regions are intersected to locate the line L_{12} which the edge belongs to (see Figure 5). Practically, the line needs to be limited in order to pick out the edge(s) on it. To do so, the extents of the faces of the regions R_1 are first projected onto the line. Each projected extent generates an interval I_1^i on L_{12}; the union of all the intervals $\bigcup_i I_1^i$ represents the contribution of R_1 to the searched edge(s). A similar job on R_2 brings to the set of intervals $\bigcup_j I_2^j$. The intersection of the contributing intervals $(\bigcup_i I_1^i) \bigcap (\bigcup_j I_2^j)$ identifies the edge(s) we are looking for. This approach ensure the removal of *false positive* in the edge recovering algorithm. It has also to be underlined that the projection of the extents of the faces of the regions onto the line is limited to the faces close to the line itself. This limitation avoids the interference with far faces not really contributing to the edge formation.

In order to exploit the architectural nature of our data, we endowed our feature extraction algorithm with a simple query management system. The user is able to define the edges she/he is interested in by simply expressing angles between adjacent regions or tolerances. An example of automatic edge detection result is shown in Figure 4.

Another important aspect of the reverse engineering process in our application is **the automatic fitting of high level primitives** to the 3D data. The least squares fitting of primitives is not a new approach. The relevant aspect of our solution is the ease for the user to extend the capability of the application and to implement new primitives fitting tools. This subsystem is composed by three modules: the *GUI* , the *PrimitiveType* and the *Least Square Engine* (please refer to Figure7). The GUI and the Least Square modules are fixed and hard coded, while the Primitive module implements a specific primitive type.
The module *Primitive* exports the initialization data required for place the primitive in a starting position toward the GUI, which return, after the interaction with the user, the data required. For example for the Arch are required three points, two at the basement and one on the top of the Arch.
The *PrimitiveType* is also responsible for providing a sampling of the primitive to the *Least Square Engine* module, which performs the minimization and return the values for

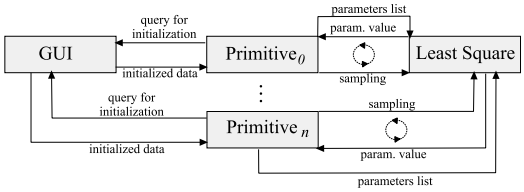

Figure 7: *Scheme of the fitting subsystem.*

the parameters. The basic organization in plug-ins makes it easy to write the new simple rules for the identification of other primitives belonging to existing classes or other classes and to expand the availability of fitting tools, since that adding a new kind of primitive only requires to implement the module *PrimitiveType* respecting the interface with the other two components. At present, the system is able to detect roman arches and generic windows (an example in Figure 6).

3. Results

We have sketched the tools we provide in our application for recovering 3D information from an architectural model.

All the operations explained in this paper are interactive, with the possible exception of the partitioning of the model in planar regions, which can use even some minutes in a model with a million triangles. However the partitioning needs to be done only once per model and it can be carried out in an unattended manner.

4. Future work

The future development of our system for the recovering of useful 3D information from range scanning data is oriented towards two main direction: (a) the automatic fitting of new families of geometric primitives and (b) the *regularization* of the data for a CAD environment.

Because we are mainly interested in buildings with some cultural or artistic relevance, we are going to expand the families or classes of objects to be automatically recognized: columns, capitals, roses, and so on.

The second aspect we intend to deal with in the next future is the so called *regularization* of the recovered information for an effective use in a CAD system. By regularization of the data we mean the possibility for the user to fix *architectural* requirements. Simple needs as, for example, walls of a building to be vertical or the windows on a wall to be at same height with respect the floor are not easily obtained by processing range scanning data. The tools we are going to design will help the user of our application in this direction.

Acknowledgments

This work was mainly supported by the CONTURA Project, a curiosity driven internal project of ISTI-CNR, Pisa (I).

References

[Bes88] BESL P. J.: *Surfaces in range image understanding*. Springer-Verlag New York, Inc., New York, NY, USA, 1988.

[BSZF99] BAILLARD C., SCHMID C., ZISSERMAN A., FITZGIBBON A.: Automatic line matching and 3d reconstruction of buildings from multiple views. In *ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery* (Munich, 1999), pp. 69–80.

[CM02] COLOMBO L., MARANA B.: 3d building models using laser scanning. *GIM - Geomatics Info Magazine 16*, 5 (2002), 32–35.

[CMS97] CIGNONI P., MONTANI C., SCOPIGNO R.: A comparison of mesh simplification algorithms, June 1997.

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM Trans. Graph. 23*, 3 (2004), 905–914.

[HHW05] HILDEBRANDT K., HILTHIER K., WARDETZKY M.: Smooth feature lines on surface meshes. In *Symposium on Geometry Processing* (2005), pp. 85–90.

[KWT88] KASS M., WITKIN A., TERZOPOULOS D.: Snakes: Active contour models. *International Journal of Computer Vision 1*, 4 (1988), 321–331.

[LT99] LINDSTROM P., TURK G.: Evaluation of memoryless simplification. *IEEE Transactions on Visualization and Computer Graphics 5*, 2 (1999), 98–115.

[Lue01] LUEBKE D. P.: A developer's survey of polygonal simplification algorithms. *IEEE Comput. Graph. Appl. 21*, 3 (2001), 24–35.

[MLM01] MARSHALL D., LUKACS G., MARTIN R.: Robust segmentation of primitives from range data in the presence of geometric degeneracy. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 3 (2001), 304–314.

[OBS04] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph. 23*, 3 (2004), 609–612.

[SB03] SCHINDLER K., BAUER J.: A model-based method for building reconstruction. In *First IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis* (2003), pp. 74–82.

[WB01] WATANABE K., BELYAEV A. G.: Detection of salient curvature features on polygonal surfaces. *Comput. Graph. Forum 20*, 3 (2001).