A Fast Algorithm for Making Mesh Models from Multiple-view Range Data

Roth, Gerhard; Wibowo, E.

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC-CNRC

*A Fast Algorithm for Making Mesh Models from Multiple-view Range Data **

Roth, G., and Wibowo, E.
October 1996

* published in the Proceedings of the DND/CSA Robotics and Knowledge Based Systems Workshop. St.Hubert, Québec, Canada. October 15-18, 1996. pp. 120-128. NRC-39179

Canada

A Fast Algorithm for Making Mesh-Models from Multiple-View Range Data

Gerhard Roth and Eko Wibowo

Visual Information Technology Group
Institute for Information Technology
Building M 50, Montreal Road
National Research Council of Canada
Ottawa, Canada K1A 0R6

Abstract

In this paper we present a way of integrating a number of different views taken by a rangefinder in order to create a single surface model. This model consists of a mesh of triangular planar patches which can be easily and efficiently rendered on graphics hardware. Our method is based on the marching cubes algorithm which was created for rendering volumetric medical data. Our approach is efficient, incremental and relatively simple to implement. We demonstrate its application on a number of range views of an office environment.

1 Introduction

This paper presents a way of building a geometric model from multiple-views taken by a range sensor. Such sensors, which are becoming more widely available are able to directly capture the geometry of an object. They do this by scanning a laser beam to obtain the distance to the objects struck by the beam. At NRCC we have developed a triangulation based laser-rangefinder which has been adapted to a wide variety of applications [1]. Different versions of the sensor can obtain data over varying distances (from a few centimetres to a few meters), and at different rates (from tens of thousands to million points per second). One of the main applications of such sensors is to build a geometric model [2].

To accomplish this task an object or environment is scanned by the sensor from a number of different viewpoints. Usually between five and fifty viewpoints are sufficient depending on the complexity of the scene. The result is a set of range images, where each image is a grid of range points. Each range point contains the three dimensional co- ordinates of a surface point relative to the origin of the rangefinder at the current sensor position. The result of this process is a large number of three-dimensional data points.

Two things are essential in order to make use of the data. The first thing that needs to be done is to register all these individual range views so that the data points are in a single co-ordinate frame. This registration process can be done mechanically, by moving the range sensor using an accurate positioning device, or from the data itself, by aligning the images so that the overlapped regions have minimal error. Both methods are being used, and have been shown to be adequate [3, 4]. Therefore in this paper we will assume that all the views have already been properly registered.

The second thing that needs to be done is to integrate all this data into some type of geometric model that is useful. Here, the choice of model depends on the application. One of the NRCC rangefinders is being mounted on a remotely operated vehicle or manipulator which will be used to explore an unstructured environment. The only interaction the operator will have with this world will be through the data obtained by the range sensor. The main requirement of the geometric model will be to display the range data in such a way the features of the environment are clear.

The most logical type of model that can be used for this purpose is a triangular mesh [5]. This consists of a large number of triangular facets that can be displayed efficiently on any graphics system. The input to our model building process is a set of registered range views, and the output should be a single triangular mesh which can be rendered on a graphics system.

Since there are many views, there is considerable overlap between the range data from different views. In a non-redundant model these overlapping portions of the data are described by a single geometric primitive (ie. a single triangular face), instead of multiple primitives (ie. multiple faces). Any algorithm that builds a geometric model from multi-view range data should create a non-redundant mesh, or else the final mesh model will be much too large.

Creating a non-redundant model is the first requirement of our mesh creation algorithm. There are number of algorithms in the literature that are able to create non-redundant meshes from multi-view range data [6, 7, 8, 9]. However, our application also has other requirements. The operator will explore the environment by obtaining successive range views. Therefore the mesh- creation algorithm must be able to incremently update the current mesh model as new views are obtained. It can not require that all the range views be available before the mesh-building process begins. This implies that the algorithm must be able to deal with incomplete data, since the incremental acquisition process means that there may be holes, or missing portions of the range data. It must be reasonably efficient in the sense that it should not take much longer to make a mesh from a range view than the range data acquisition process, which is no more than a minute per view. It is also the case that the accuracy of the range data varies considerably with the distance to the object. Therefore the resulting mesh should be more detailed in regions where the sensor data is more accurate, and less detailed elsewhere. Besides visualization, the range data will also be used to avoid collisions when the sensor is moved in the remote environment. Therefore it would be an advantage if the underlying data structure for our mesh algorithm also be useful in collision avoidance applications.

There is currently no algorithm in the literature that meets all these goals. The closest is one that is based on the marching cubes algorithm [7, 10]. However, it falls short in a number of areas, especially the speed of execution. What we have done is to go back to the original marching cubes algorithm and create a mesh algorithm that is specifically tailored to our application. While it shares some of the characteristics of previous mesh creation algorithms, it has some significant differences and improvements.

In this paper we will describe our mesh creation algorithm which takes a set of registered range views, and outputs a single triangular mesh. The paper is organized as follows. The next section presents the algorithm, data structure and some modifications we have made to the marching cubes algorithm. This will be followed by a number of examples of the algorithm in operation. Finally we will have some conclusions and discussions of future work.

2 Mesh Creation Using Marching Cubes

The marching cubes algorithm was created to render volumetric-based medical imagery. Such imagery consists of a set of three dimensional voxels, which contain data from medical sensors such as CAT scans. The algorithm takes its name from the fact that it marches around the voxels and produces a set of triangles for each voxel element. The final result is a triangular mesh that can be rendered on a graphics system.

In order to use this algorithm we must first convert our range data into voxel form. At first glance, a voxel representation may appear to require a large amount of storage. However, we make use of the fact that range data is sampled only on the surface of an object. This means that only a fraction of the total number of possible voxels are actually occupied. Therefore the amount of memory required for the voxel representation is greatly reduced when the input source is range data instead of medical imagery, where all the voxels are occupied.

The first thing that must be done is to compute the dimensions of the voxel grid. For now, we will assume that the grid dimensions have been set. Later in the paper we will discuss an approach that performs this computation directly from the range data. The resulting voxel grid has a specified size in each of the x, y and z directions. Our grid data structure consists of a two dimensional array of pointers where each pointer contains a linked list of voxels. The array is indexed in the x and y direction, and each linked list contains all the voxels in the z direction that are currently occupied. A voxel is occupied if it contains any range data points. All the range data points for a voxel are themselves placed in a linked list associated with the voxel. For any 3d range point it is a simple matter to find the voxel in which it falls. Therefore for an entire range image, we can efficiently add range points to this volumetric data structure. Since there are range points only on the surface of an object, the voxel map is relatively sparse, and the storage requirements are reasonable.

The mesh creation algorithm is actually very simple. All the range data points are placed in their associated voxel. Then each occupied voxel is processed to produce a small number of triangles that describe the local surface defined by these range points. This is followed by a cleanup operation in which gaps between neighbouring triangles from each voxel are bridged, and where isolated triangles are removed. The result is a triangular mesh that describes the surface of the object at a given resolution, the resolution of the voxel grid.

The most important step is efficiently finding the mesh triangles for each voxel. In each of the occupied voxels a plane is fit to the range data points to produce a first order approximation to the local surface geometry, which is called the tangent plane. This plane is then intersected with the boundaries of that voxel. This intersection produces a small planar patch, which is then divided into a small number of triangles. This process can be made very efficient because the topological possibilities which result from the intersections are clearly limited. We know that there are always between one and four of the voxel's vertices on either side of the voxel plane. Once this number of vertices is known, then the edges of the voxel that intersect the tangent plane are completely defined. Therefore the triangulation of the tangent plane can be made very efficient by using a table lookup which is indexed by the relationship of the voxel vertices to the tangent plane [7, 10].

One issue which has not been discussed is how to choose the direction of the surface normals for each of the triangular faces. A simple way that this can be done is to note the location of the laser source at the current rangefinder position for the given point. Because range data is obtained by scanning the laser beam this implies that each local surface normal must point towards the laser source. A simple dot product test can therefore be used to choose the appropriate direction for the normal of each triangular mesh patch.

We have not yet described how we choose the voxel resolution. If this resolution is too low, then the grid is too course, and if it is too high then the grid has missing portions. While this grid resolution can be set manually, we have found that the following automatic method works well. We simply choose a subset of the input points and find the average distance of each point to its closest neighbour. By setting the voxel grid to a value slightly more than this voxel distance we can be reasonably certain that each voxel contains enough points to define a tangent plane. This method is used in the examples in the next section, and it works well.

3 Experiments

In this section we will show an example where we take three range views of a room, and make a single non-redundant triangular mesh from these views. The three range images are shown in Figure 1, with the intensity value being proportional to the distance from the sensor. This means the lighter parts of the shaded image are closer to the sensor. Each range consists of two hundered and fifty thousand points for a total of seven hundred and fifty thousand points, which is a considerable amount of data. Our example has three views, but there are many applications where more views are necessary.

The algorithm reads the three views, automatically computes a resolution for the required grid, and computes the triangular mesh. The mesh is displayed from two different viewpoints using a simulated light source in Figure 2. This is a high resolution mesh, however it only contains seventy thousand mesh vertices, which is a reduction of an order of magnitude in the amount of data. The close up demonstrates that the mesh is non-redundant since there is only a single triangular mesh element in the overlapping regions of the three images. The total time taken to create the mesh is about three minutes on an SGI Indy workstation, which is reasonable. The total number of triangular meshes varies with the resolution of the voxel grid. In Figure 3, we created another mesh at coarser resolution, which was selected by hand. This mesh contains only seven thousand vertices, which is a reduction in two orders of magnitude over the original data. However, it is still a relatively faithful replication.

In Figure 4 we show a close up of an individual at a terminal. In Part (a) of the figure we see the shaded version of the triangular mesh, while in Part (b) we see the actual triangles.

4 Discussion and Conclusions

Let us consider the requirements for the mesh algorithm that have been stated at the beginning of the paper. Our approach is certainly incremental, efficient, and is able to deal with holes and missing data. Because it uses a voxel grid as the underlying data structure it can also be easily integrated with a number of collision avoidance/detection algorithm that use this data structure. One important advantage of the voxel grid approach is that there is no need to keep the original range data points, once the tangent plane for each occupied voxel is computed. This means the storage requirements depend only on the number of occupied voxels, and not the number of range data points. This is an important difference of this approach from others non-redundant mesh algorithms. It means that we can make meshes from a very large number of views, and as long as the voxel grid dimensions do not change, the storage requirements will not increase.

Currently our algorithm does not satisfactorily deal with range data of different accuracies. We know that the the range data that has been obtained with the sensor at different standoff distances has different accuracies. However, currently we can only create the mesh at single level of voxel resolution. We would like to be able to create the underlying voxel grid at a number of simultaneous different resolutions. Creating such a multi-resolution grid is one of our current areas of research.

References

[1] M. Rioux, "Laser rangefinders based on synchronized scanning," *Applied Optics*, vol. 23, pp. 3837–3844, 1985.

[2] W. Grimson and J. Mundy, "Computer vision applications," *Communications of the ACM*, vol. 37, pp. 45–51, Mar. 1994.

[3] H. Gagnon, M. Soucy, R. Bergevin, and D. Laurendeau, "Registration of multiple range views for automatic 3-d model building," in *Proceedings of IEEE Computer Vision and Pattern Recognition Conference*, (Seattle, Washington), pp. 581–586, June 1994.

[4] Y. Chen and G. Medioni, "Object modelling by registraion of multiple range images," *Image and Vision Computing*, vol. 10, pp. 145–155, Apr. 1992.

[5] J. D. Foley and A. V. Dam, *Fundamentals of Interactive Computer Graphics.* Reading, Mass.: Addison-Wesley, 1982.

[6] M. Soucy and D. Laurendeau, "Multi-resolution surface modelling from multiple-range views," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, (Champlaign, Illinois), pp. 348–353, 1992.

[7] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized data points," in *Computer Graphics 26: Siggraph'92 Conference Proceedings*, vol. 26, pp. 71–78, July 1992.

[8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh optimization," in *Computer Graphics: Siggraph '93 Proceedings*, pp. 19–25, 1993.

[9] Y. Chen and G. Medioni, "Surface description of complex objects from multiple range images," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (Seattle, Washington), June 1994.

[10] W. E. Lorenen and H. E. Cline, "Marching cubes: a high resolution 3d surface reconstruction algorithm," in *Computer Graphics: Siggraph'87 Conference Proceedings*, vol. 21, pp. 163–169, July 1987.

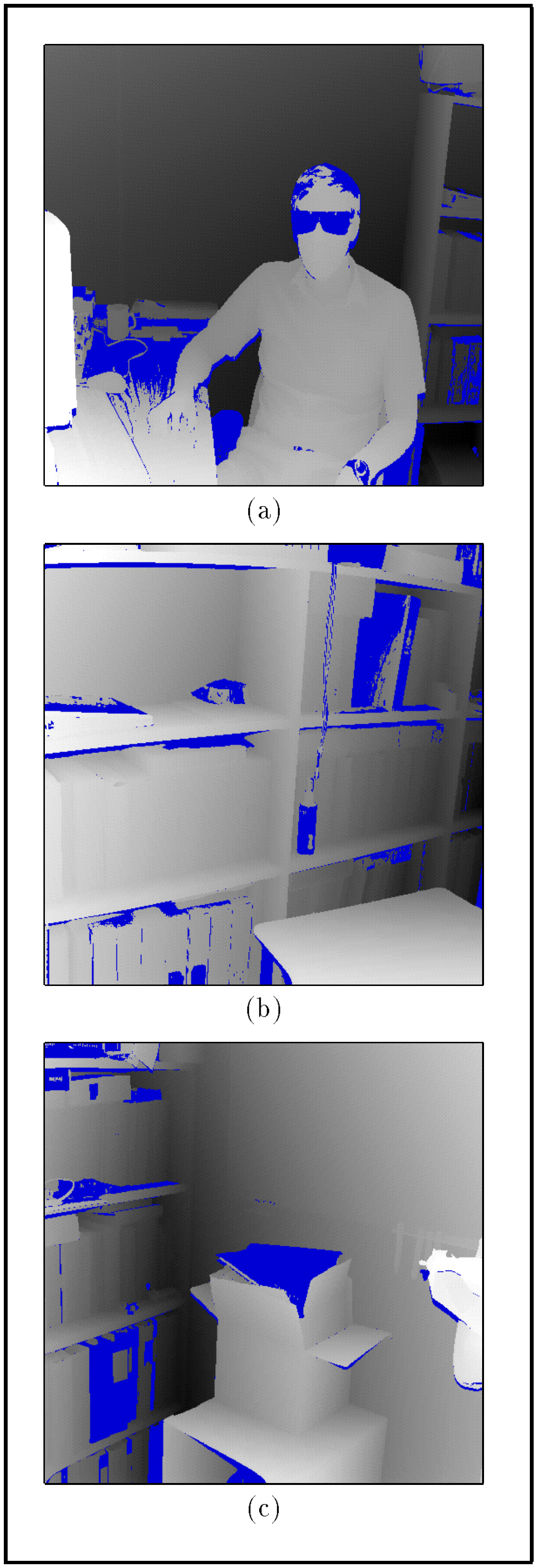
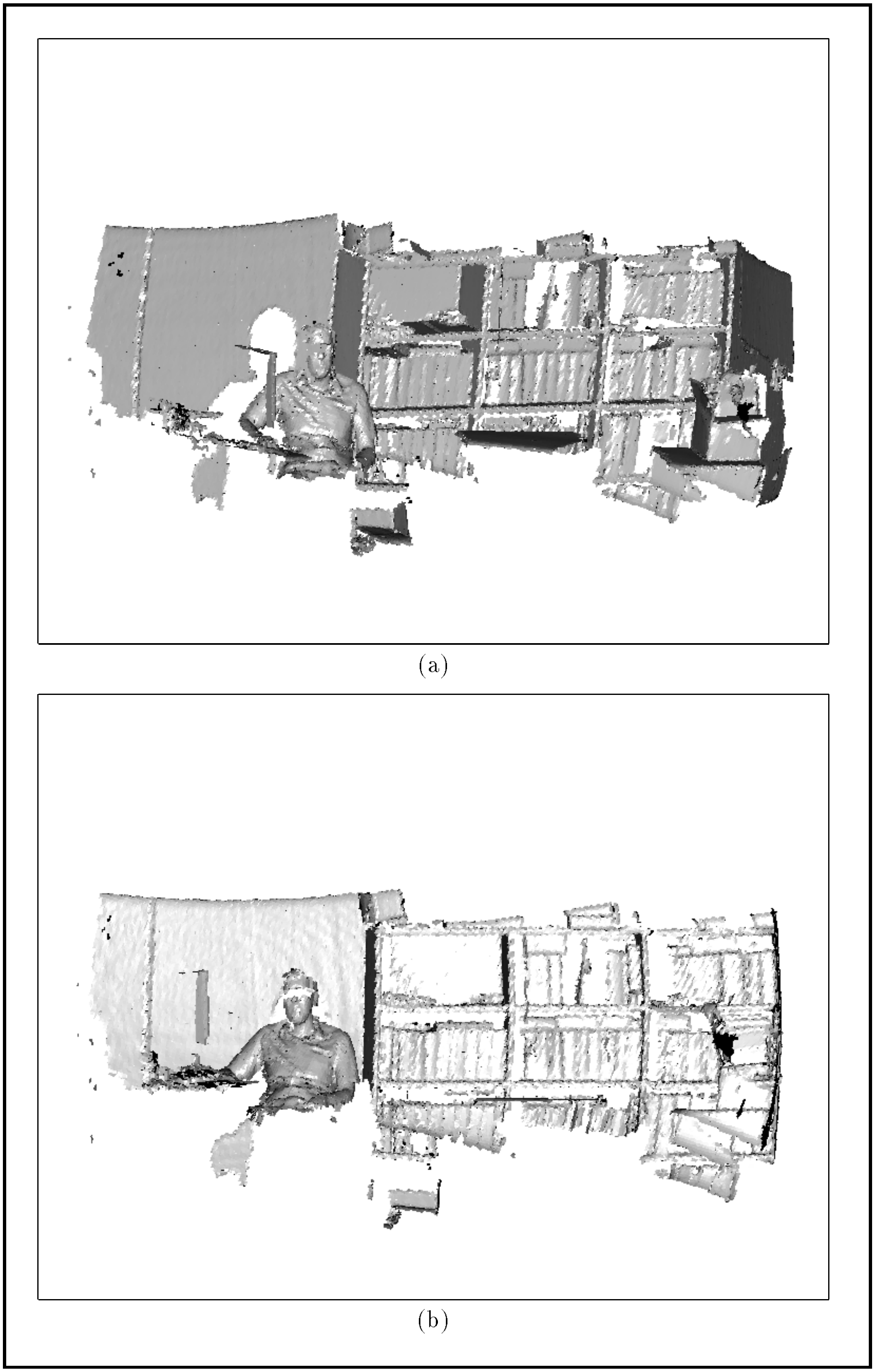


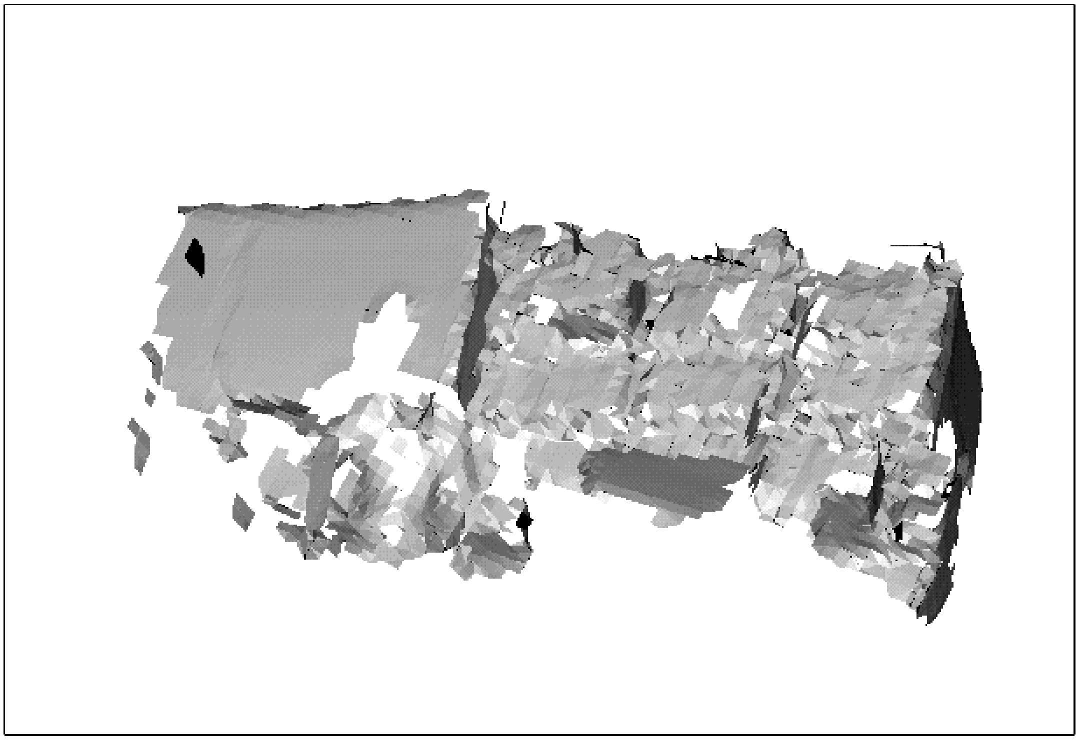
Figure 1: Three range views of a person at a terminal, and some bookshelves.



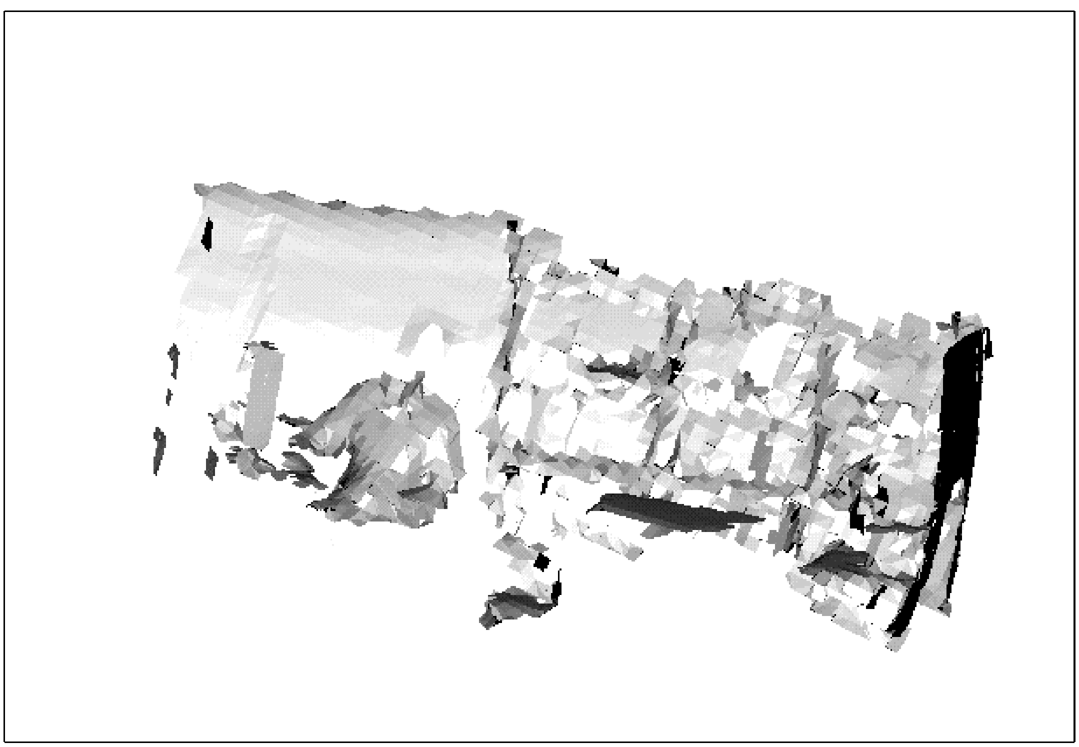
(a)

(b)

Figure 2: Two different viewpoints of the high-resolution triangular mesh model using synthetic light source.



(a)



(b)

Figure 3: Two different viewpoints of the low-resolution triangular mesh model using synthetic light source.

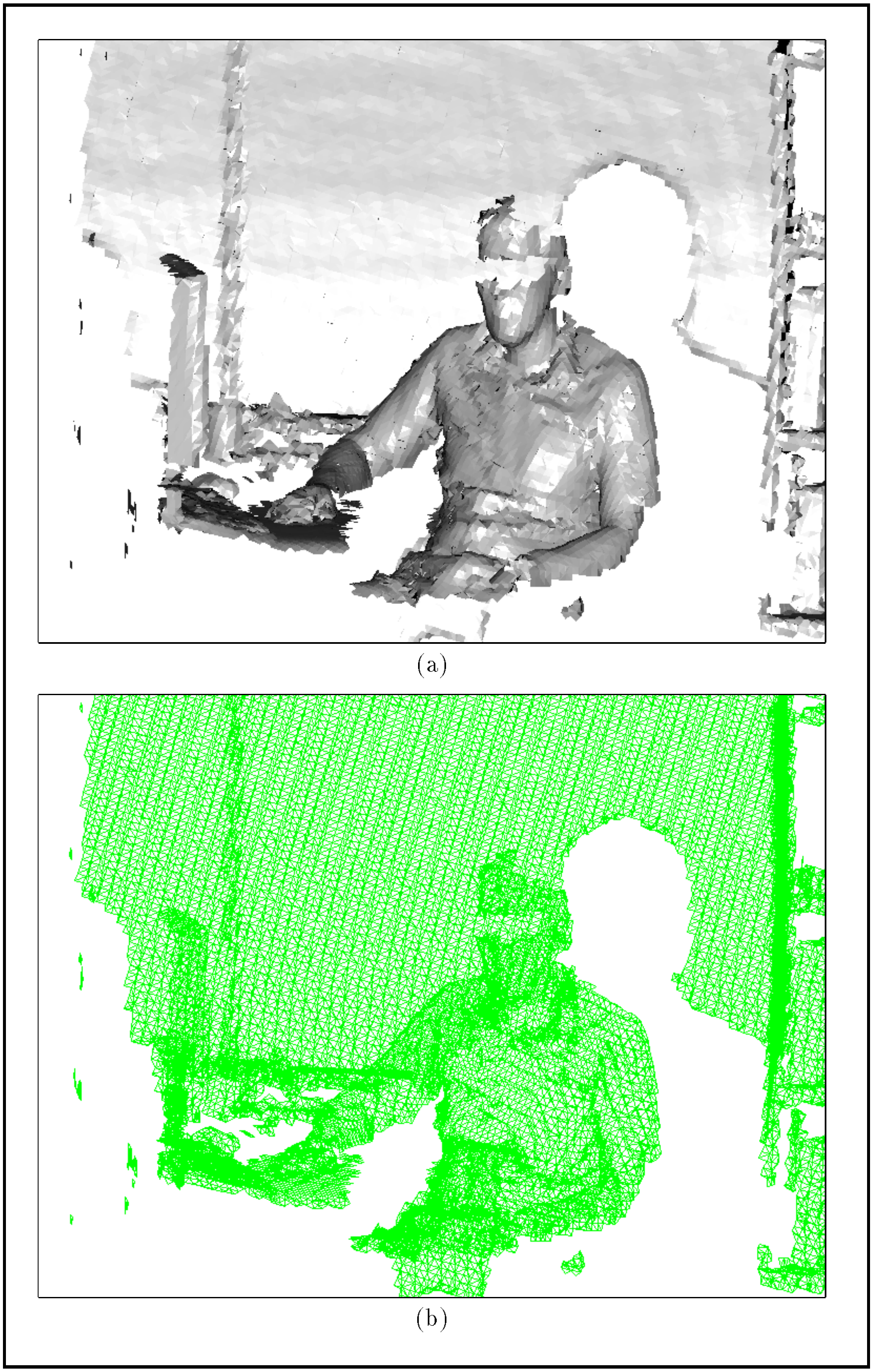


Figure 4: Two close up views of the person at the terminal (a) The shaded person (b) The actual triangles that make up the mesh model.