# Building 3D Environment Models for Mobile Robots Using Time-Of-Flight (TOF) Laser Scanner

Sašo Koceski[1] and Nataša Koceska[2]

*Abstract* – **In this paper, we present system and an intuitive method to generate visually convincing 3D models of indoor environments from data collected using TOF laser scanner. The method allows line and surface detection and mesh representation of the input data. The method shows accurate 3D environment representation useful for different applications in mobile robotics.**

*Keywords* – **TOF Laser Scanner, Environment modelling, Mobile robots, 3D mesh, 3D triangulation.**

## I. INTRODUCTION

Since standard computers allow efficient processing and visualization of three-dimensional data, the interest of using 3D graphics has increased in numerous fields. Car design, architecture, or the modern movie industry are merely a few examples of fields which are unimaginable nowadays without the use of 3D graphics. The increasing need for rapid characterization and quantification of complex environments has created challenges for data analysis. Mobile systems with 3D laser scanners that automatically perform multiple steps such as scanning, gaging and autonomous driving have the potential to greatly advance the field of environment representation. 3D information available in real-time enables autonomous robots to navigate in unknown environments, e.g., in the field of inspection and rescue robotics.

Autonomous navigation of a mobile robot requires basic capabilities for sensing the environment in order to avoid obstacles and move in a safe way. The problem of the reconstruction can be expressed as a procedure of learning the topology from the data set and reduction of the data set cardinality. We address the problem via the analysis and realistic representation of an unknown indoor environment. Our objective is to build environment models from the real measured data of building interiors using intuitive methods and low-cost data acquisition systems.

Some groups have attempted to build 3D volumetric representations of environments with 2D laser range finders. Thrun et al. [1],[2] and Früh and Zakhor [3] use two 2D laser range finder for acquiring 3D data. One laser scanner is mounted horizontally and one is mounted vertically. The latter one grabs a vertical scan line which is transformed into 3D points using the current robot pose. But the accuracy in these cases was not satisfactory and also the costs were high.

Our approach is based on 2D Time-Of-Flight (TOF) laser scanner and extended by a low cost rotation module based on a servo command. Combining such an extension with a set of fast algorithms has resulted in good environment representation system.

[1]Sašo Koceski is with the Applied Mechanics Laboratory, DIMEG, University of L'Aquila, 67040 Roio Poggio (AQ), Italy, E-mail: koceski@yahoo.com

[2]Nataša Koceska is with the Applied Mechanics Laboratory, DIMEG, University of L'Aquila, 67040 Roio Poggio (AQ), Italy, E-mail: njankova@yahoo.com

## II. DATA ACQUISITION SYSTEM

### A. Main components and connections

The main component of the system is the sensor for data acquisition Laser Radar (LADAR) LD-OEM1000. It is placed in one aluminum carrying construction and attached to the mount with 1 Degree Of Freedom (DOF), so that it can be rotated. On the top of the mount, berth for camera is foreseen (Fig. 1). The rotational axis is horizontal. Annotation: An alternative approach is to rotate the LADAR around the vertical axis. Throughout this paper we will only discuss the approach based on a horizontal rotation, but all presented algorithms can be used in the same way. The differences between both approaches are the orientation of the apex angle and the effects of dynamic objects moving through the scene, e.g. persons. Using vertical scanning, a perturbation either appears with low probability within a few scans making them useless for further data processing, or does not appear at all.

Fig.1. Photo of the created system

For the rotation of the scanner's movable element, a servo command (Hitec HSC 5955TG) was chosen. In order to move the servo command, a PPM (Pulse Position Modulation) was used. It is connected to the main processing unit (Standard

PC) via PCI NI 6221 Data Acquisition board (DAQ). For the LADAR, CAN data interface was chosen (instead of a RS-232 which is the standard interface), because it enables high grabbing speed, up to 1 MBit/s, and it can be easily connected via low cost CAN to USB adaptor to an USB port.

*B. Working principle*

The LADAR measures its environment in two-dimensional polar coordinates. When a measuring beam strikes an object, the position is determined with regard to the distance and direction. The measured data can be transferred in real time to the connected computer for evaluation.

The scan is carried out in a 360° sector. The LADAR ranges on bright; natural surfaces (a white house wall, for example) are approx. 100m. (Fig. 2).

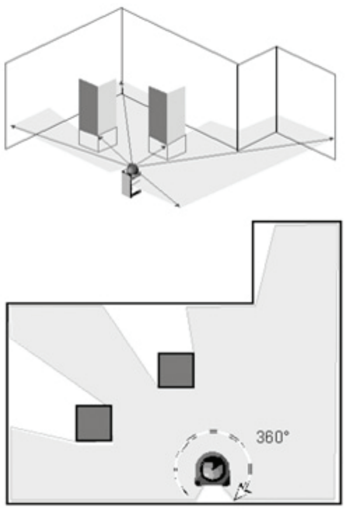


Fig.2. Scanning the environment

The distance to the object is calculated from the propagation time required by the light from the point at which the beam is emitted until the reflection is received by the sensor. The scanner head rotates with a frequency of 5 to 20 Hz (programmable). A laser pulse is emitted in accordance with a variable angle step, thereby triggering a distance measurement. The maximum angular resolution is 0.125°. This is set by the angle encoder at 5,760 steps. The angular resolution can be selected as an integral multiple of 0.125°.

*C. Data acquisition and interpretation*

Working principle and the given setup determine an intrinsic order of the acquired data. The data coming from the 2D LADAR is ordered counterclockwise. In addition the 2D scans (scanned planes) are ordered due to the rotation. While obtaining horizontal profiles LADAR is sending CAN data packets, with different useful information, among which a pairs of two by two values for distance and corresponding angle. The distance value is represented by a 16-bit binary value with a resolution (step width) of 3.9 mm (1/256 m). The angle is also represented by a 16-bit binary value with a resolution of 1/16°. Having in mind the vertical rotation of the system (the value for the azimuth angle) as well as the dimension of the structural components of the system (Fig. 3), these values are subsequently transformed into spherical coordinates according to the Eqs. (1) and (2).

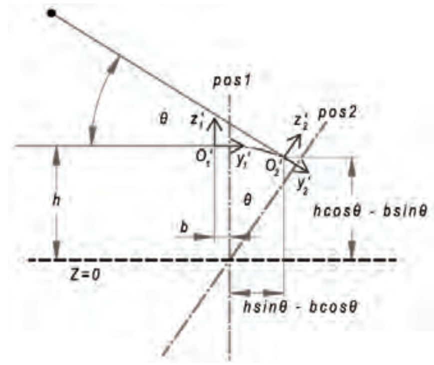


Fig.3. Coordinates transformation

$$\begin{cases} x' = r \cdot \sin\alpha \\ y' = r \cdot \cos\alpha \end{cases} \tag{1}$$

$$\begin{cases} X = -r \cdot \sin\alpha \\ Y = -r \cdot \cos\alpha \cdot \cos\vartheta - h \cdot \sin\vartheta + b \cdot \cos\vartheta \\ Z = -r \cdot \cos\alpha \cdot \sin\vartheta + h \cdot \cos\vartheta + b \cdot \sin\vartheta \end{cases} \tag{2}$$

*D. Pre-processing algorithms*

Different pre-processing algorithms for line and surface detection were applied to the data. First algorithm is a simple straightforward matching algorithm running in *O(n)* (n is the number of points), with small constants. The algorithm implements a simple length comparison. The data of the LADAR (points $a_0,a_1,\ldots a_n$) is ordered counterclockwise so that one comparison per point is sufficient. We assume that the points $a_i,\ldots a_j$ are already on a line. For $a_{j+1}$ we have to check if the condition in Eq. (3) is satisfied.

$$\frac{\left\| a_i, a_{j+1} \right\|}{\sum_{t=i}^{j} \left\| a_t, a_{t+1} \right\|} < \varepsilon(j) \tag{3}$$

To obtain better results this algorithm runs on preprocessed data. Data points located close together are joined so the distance from one point to the next point is almost the same. This process minimizes the fluctuations within the data and reduces the points to be processed. Hence this algorithm runs very fast, but the quality of the lines is limited.

The quality could be increased by additional analysis (using Hough transformation for example) of the photos obtained with camera during the scanning process.

After line detection is done the data is converted into 3D. Based on the detected lines, the following algorithm tries to detect surfaces in the 3-dimensional scene.

Scanning a plane surface, the line detection algorithm will return a sequence of lines in successive 2D scans approximate the shape of this surface. The task is to recognize such structures within the 3D data input and to concatenate these independent lines to one single surface.

The surface detection algorithm proceeds the following steps:

1. The first set of lines - coming from the very first 2D scan - is stored.
2. Every other line is being checked with the set of stored lines. If a matching line is found, these two lines are transformed into a surface.
3. If no such matching line exists, the line may be an extension of an already found surface. In this case, the new line is matching with the top line of a surface. This top line is being replaced by the new line, resulting in an enlarged surface.
4. Otherwise the line is stored as a stand-alone line in the set mentioned above.

To achieve real time capabilities, the algorithm makes use of the characteristics of the data as it comes from the LADAR, i.e. it is the order by the scanned planes. Therefore the lines are sorted throughout the whole scene (with regard to their location within the virtual scene) due to their inherited order. Thus an efficient local search can be realized.

Two criteria have to be fulfilled in order to match lines: On one hand the endpoints of the matching line must be within a ε-area around the corresponding points of the given line. On the other hand the angle between the two lines has to be smaller than a given value. The second constraint is necessary for correct classification of short lines, since they fulfill the distance criterion very easily. These algorithms enables that the robot or a user gets much information about objects in the scenery right during the scan, which is essential for path planning and collision avoiding during the movement of mobile robots inside indoor environments.

*E. Post-processing algorithms*

Post-processing algorithms were used to create a 3D mesh representation of the room that will give more qualitative information for safe navigation of mobile robots. Despite of the prior algorithms, this step requires information about the whole scene and has to be done after the scan process is finished.

This procedure is based on the Delaunay triangulation algorithm that typically projects the data set onto a plane for finding the most probable connections between close vertices and re-locates in space the connected vertices at the end creating a triangular mesh.

At any stage of the triangulation process one has an existing triangular mesh and sample point to add to that mesh. The process is initiated by generating a supertriangle, an artificial triangle which encompasses all points. At the end of the triangulation process any triangles which share edges with the supertriangle are deleted from the triangle list:

1. All the triangles whose circumcircle encloses to point to be added are identified, the outside edges of those triangles form an enclosing polygon. (The circumcircle of a triangle is the circle which has the three vertices of the triangle lying on its circumference).
2. The triangles in the enclosing polygon are deleted and new triangles are formed between the point to be added and each outside edge of the enclosing polygon.
3. After each point is added there is a net gain of two triangles. Thus the total number of triangles is twice the number of sample points. (This includes the supertriangles, when the triangles sharing edges with the supertriangle are deleted at the end the exact number of triangles will be less than twice the number of vertices, the exact number depends on the sample point disturbation). The triangulation algorithm may be described in pseudo-code as follows:

**subroutine triangular**
*input:* vertex list
*output:* triangle list

```
  initialize the triangle list
  determine the supertriangle
  add supertriangle vertices to the end of the vertex list
  add the supertriangle to the triangle list
  for each sample point in the vertex list
     initialize the edge buffer
     for each triangle currently in the triangle list
       calculate the triangle circumcircle center and radius
       if the point lies in the triangle circumcircle then
            add the three triangle edges to the edge buffer
            remove the triangle from the triangle list
       endif
     endfor
     delete all doubly specified edges from the edge buffer
     this leaves the edge of the enclosing polygon only
     add the triangle list all triangles formed between the
     point and the edges of the enclosing polygon
  endfor
  remove any triangles from the triangle list that use the
  supertriangle vertices
  remove the supertriangle vertices from the vertex list
```
**end**

The above can be refined in a number of ways to make it more efficient. The most significant improvement is to pre-sort the sample points by one coordinate, the coordinate used should be one with the greatest range of samples. If the X axis is used for pre-sorting then as soon as the x component of the distance from the current point to the circumcircle centre is greater than the circumcircle radius, that triangle need never to be considered for later points. With the above improvement the algorithm presented here should increase with the number of point as approximately $O(N^{1.5})$.

The algorithm does not require a large amount of internal storage. The algorithm only requires one internal array and that is a logical array of flags for identifying those triangles that no longer need to be considered.

## III. Experimental results

The scan shown on Fig. 4 which represents indoor of our laboratory in the form of cloud of points, was obtained by our system, with the following settings: Resolution - 0,25° x 0,25°; Frequency – 20 Hz. The scan covers an area of 180(h)x60(v) degrees and includes in total 157976 points. Wired model of the entire scene after triangulation contains 307095 triangles.

Details of one part of the same scene in the form of wired mesh is presented on Fig. 5.

Standard PC with AMD Athlon 64 Processor 2.41GHz and 1GB RAM was used. The software based on the above algorithm was developed in C++ using the Computational Geometry Algorithms Library (CGAL) and VTK (The Visualization Toolkit) which are freely available data processing and visualization libraries.

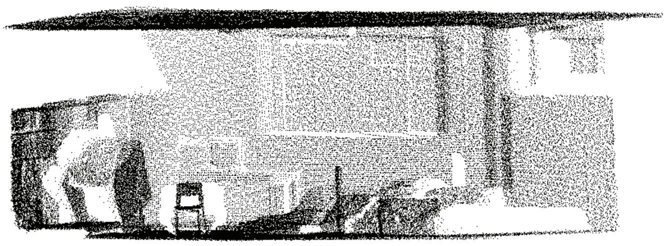

Fig. 4. Point cloud representation of indoor environment

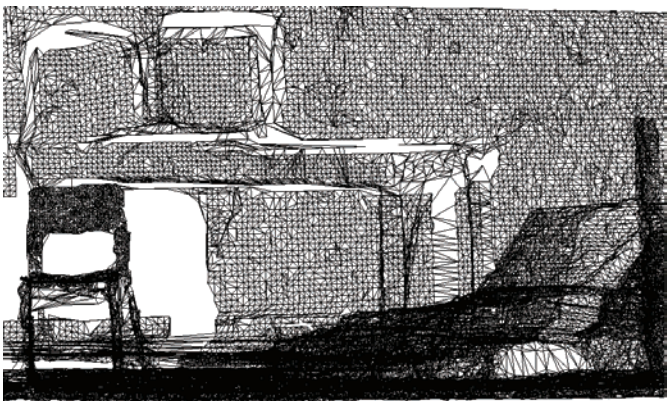

Fig. 5. Details of the environment in the form of wired mesh

## IV. Conclusion

In this paper low-cost, precise and reliable 3D sensor and methods for environment modeling are presented. With the proposed approach the mobile robot navigation and recognition could be significantly improved. The 3D sensor is built on base of 2D TOF laser scanner, which senses the environment contactless, without the necessity of landmarks. The implemented software, based on presented algorithms, give accurate mesh representation of input data, where particular objects could be easily identified.

## V. Future work

The future work from one side regards hardware modifications and supplements of the system. First of all, replacement of the current servo command with step motor that could be connected to the PC via CAN bus or USB connection is foreseen. This way we will avoid the usage of the DAQ card and will reduce the cost of the solution. The usage of the camera for photo capturing is also planned. This would lead to efficient texture mapping of the scanned environments and objects. On the other side improvement of the current algorithms and algorithms for terrain maps creation and feature extraction from the scans is planned to be implemented.

## References


[1] S. Thrun, D. Fox, W. Burgard, A real-time algorithm for mobile robot mapping with application to multi-robot and 3D mapping, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'00), San Francisco, CA, April 2000.

[2] D. Hähnel, W. Burgard, S. Thrun, Learning compact 3D models of indoor and outdoor environments with a mobile robot, in: Proceedings of the Fourth European Workshop on Advanced Mobile Robots (EUROBOT'01), Lund, Sweden, September 2001.

[3] C. Früh, A. Zakhor, 3D model generation for cities using aerial photographs and ground level laser scans, in: Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR'01), Kauai, Hawaii, December 2001.

[4] G. Guidi, J.-A. Beraldin, C. Atzeni, "High accuracy 3D modeling of Cultural Heritage: the digitizing of Donatello's Maddalena", IEEE Transactions on Image Processing, Vol. 13-3, 2004, pp. 370-380.

[5] Terzopoulos, D., The Computation of Visible Surface Representation. IEEE Transactions on PAMI, Vol. 10, No 4, 1988.

[6] G. Guidi, J.-A. Beraldin, S. Ciofi, C. Atzeni, "Fusion of range camera and photogrammetry: a systematic procedure for improving 3D models metric accuracy", IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics. Vol. 33-4, 2003, pp. 667- 676.

[7] VTK (www.kitware.com/vtk.html)

[8] CGAL (www.cgal.org)