

Modeling and Simulation of Short Range 3D Triangulation-Based Laser Scanning System

Theodor Borangiu, Anamaria Dogar, Alexandru Dumitrache

Abstract: In this paper, a simulation environment for a short range 3D laser scanning system that uses triangulation is presented. The simulation is used for integrating a laser scanning probe that uses a line laser and two cameras, with a vertical articulated robotic arm with 6 degrees of freedom and a rotary table. The optical subsystem is simulated using POV-Ray ray tracing software, and the image processing for triangulation, together with the robotic arm kinematics and the user interface, are implemented in MATLAB.

Keywords: 3D Laser Scanning, Simulation, Ray Tracing.

1 Introduction

This work is part of a bigger project, whose goal is to integrate a short range 3D laser scanning probe with a vertical articulated robotic arm with 6 degrees of freedom and a rotary table. The laser probe is able to measure distances from 70 to 250 millimetres, achieving 30 μm accuracy. The robotic arm will move around the workpiece being scanned by using computer-generated adaptive scanning paths, which are computed in real-time while the scanner is discovering the workpiece features. The scanned 3D models will be then reproduced on a CNC milling machine. The simulator presented in this paper is designed to be a development tool and test bench for the adaptive scanning algorithms which will be developed.

2 Motivation

Using a simulation environment for designing the scanning algorithms has several advantages:

- The possibility of collisions between the robotic arm, laser probe and workpiece is eliminated. As the laser probe is an expensive device, collision avoidance is a very important point to consider;
- The system can be analyzed in ideal conditions, with no surface reflections, external light sources or perturbations in the measurements;
- The parameters of the scanning system components, like camera location, focal length, optical sensor resolution, laser beam width, rotary table size and location, can be freely changed, and the influence of these changes can be analyzed thoroughly;
- Scanning of objects with complex shapes and different surface properties, that may not be physically available, is possible.

There are also a number of disadvantages, the biggest one being the computational power involved for accurate simulations in real-time, and the second one being the difficulty for simulating the less-than-ideal conditions of the real system. In the authors' opinion, the advantages outweigh the shortcomings involved here.

3 Problem description

The laser scanning system that is to be simulated consists of three main components:

- The robotic arm
- The laser probe
- The rotary table

For the robotic arm, the simulator should allow displaying the robot in any user-defined position. The user should be able to control either the joint angles for each articulation, or the Cartesian position together with the orientation given by YZZ' Euler angles. The user should also be able to specify the tool transformation.

For the laser probe, the software should simulate the interaction of the laser beam with any user-defined workpiece, having various surface properties. The two cameras which are integrated into the laser probe should also be simulated, and the image which would be captured by them should be displayed to the user. Furthermore, the laser beam should be detected in the images from the two cameras, and the triangulation equations should be applied to them in order to compute a point cloud. The point clouds obtained from simulating the scanning process from different viewing angles should be transformed into a fixed coordinate system, that will be attached to the workpiece being scanned.

As for the rotary table, its rotation have to be simulated, and the workpiece that sits on the table should rotate synchronously with the table. Behaviors such as inertial movement due to fast table movements do not have to be simulated; the workpiece should be considered attached to the table.

4 Proposed Solution

The first decision is to choose which software environments are suitable for development of this simulator. Since it is needed to model the interaction between the laser beam and an arbitrary surface, a possible solution is to compute the intersections between the laser rays and the given object. This is not a trivial task, and there are many 3D renderers that use the *ray tracing* method, for which there is a good description in [3]. Examples of ray tracing software include the free POV-Ray, Rayshade, Radiance, and the commercial software, such as 3D Studio Max, Maya or Catia.

For this simulator, POV-Ray was chosen for modeling the laser beam. POV-Ray uses a *Scene Description Language* [4], which is used to describe the objects, lights and cameras that interact in a virtual environment by means of ASCII-based input files. This capability allows easy interfacing with many programming environments. Furthermore, there is a command-line version of POV-Ray, that simplifies integration of POV-Ray with other software programs.

For the computational side of the simulator, which involves 3D matrix multiplications and trigonometric operations, the programming environment was chosen to be Matlab. Its advantages include an interpreted language with syntax oriented for matrix operations and mathematical functions, also having an Image Processing Toolbox and built-in 2D/3D graphics capabilities.

4.1 Robot arm and rotary table modeling

For modeling the robot arm, there are two main points to consider: computing positions and orientations for every link of the robot, including direct and inverse kinematics, and displaying the robot to the user in a given configuration.

The World coordinate system used here, $X_0Y_0Z_0$, is right-handed, with the X_0Y_0 plane being horizontal, X_0 axis pointing forward, Z_0 axis pointing upwards, and the origin being at the base of the robotic arm. The rotary table has the same reference frame as the workpiece, $X_RY_RZ_R$, and the Z_R axis points in the same direction as Z_0 .

Direct and Inverse Kinematics

The direct kinematics for the robot arm function is obtained by using the Denavit-Hartenberg [1] convention. The first step is to assign individual reference frames to each link from the kinematic chain, which includes the six robot arms and the laser probe (Fig. 1(a)). The direct kinematics function is the product of the individual homogeneous transformations [2] for each link $i = \overline{1..6}$. An individual matrix, called T_i^{i-1} , is the transformation from the $(i-1)^{th}$ link reference frame to the i^{th} link reference frame. The 0^{th} link is the robot base, and the 7^{th} link is the laser probe. T_L^6 is the transformation from the 6^{th} joint (robot flange) to the laser probe reference frame, and it is a constant matrix, since the laser probe is rigidly attached to the robot. Knowing the the Denavit-Hartenberg parameters a_i , d_i , α_i and θ_i for each joint $i = \overline{1..6}$, with θ_i being the joint variables, the individual transformations T_i^{i-1} can be written:

$$T_i^{i-1} = \mathcal{R}_Z(\theta_i) \mathcal{T}(a_i, 0, d_i) \mathcal{R}_X(\alpha_i) \quad (1)$$

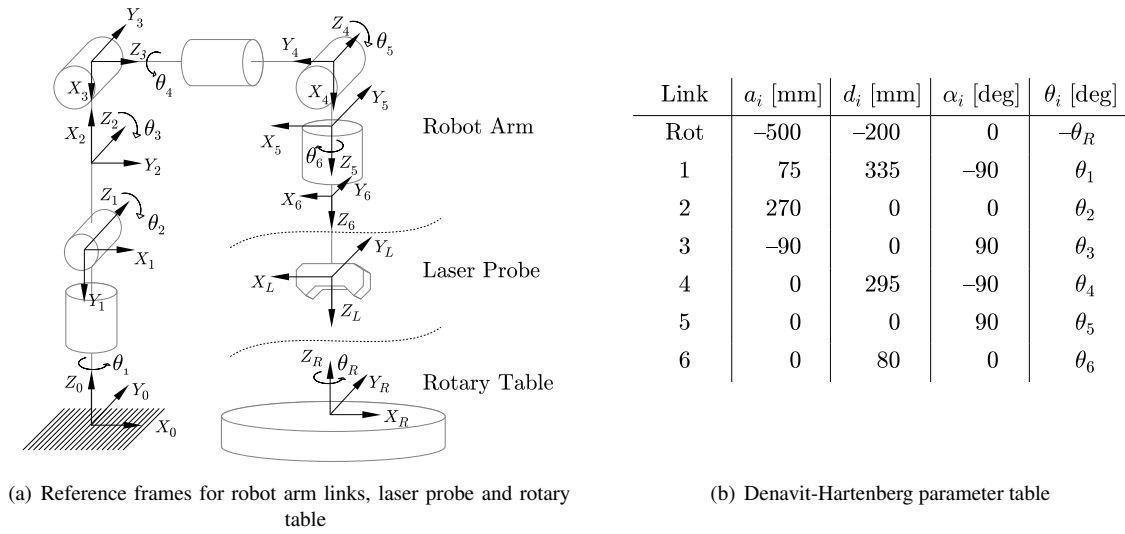


Figure 1: Reference frame assignment and Denavit-Hartenberg parameter table

where $\mathcal{T}(x, y, z)$ is the homogeneous translation, and $\mathcal{R}_A(\phi)$ is the homogeneous rotation around axis A with angle ϕ . All the distances are expressed in millimetres, and all the angles are expressed in radians.

The direct kinematics transforms are 4×4 matrices:

$$T_{DK}^* = T_6^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 \quad (2)$$

$$T_{DK} = T_7^0 = T_{DK}^* T_L^6 \quad (3)$$

The matrix T_{DK}^* may be used for transforming any object attached to the robot flange, in this case, the laser probe. The matrix T_{DK} expresses the position of the laser reference frame, with respect to the robot base, and will be used to convert the laser probe measurements into a unique coordinate system.

The inverse kinematics was implemented using Peter Corke's Robotic Toolbox, function `ikine`, which uses the pseudo-inverse of jacobian method [6]. The simulator is also able to connect to an existing robot controller via TCP/IP and use its internal inverse kinematics routine, which has been found to have a slightly different behavior, depending on the initial estimation.

Considering the rotary table as a 7-th link of the kinematic chain, and moving the reference frame to the center of the table $X_R Y_R Z_R$, the table is modelled as a new link, applied before the 6 links of the robot (Fig. 1(b)), and the transformations required to convert from the robot base reference frame to the rotary table reference frame and viceversa are:

$$T_0^R = \mathcal{R}_Z(-\theta_R) \mathcal{T}(a_R, 0, d_R) = \mathcal{R}_Z(-\theta_R) \mathcal{T}(-500, 0, -200) \quad (4)$$

$$T_R^0 = (T_0^R)^{-1} = \mathcal{T}(-a_R, 0, -d_R) \mathcal{R}_Z(\theta_R) = \mathcal{T}(500, 0, 200) \mathcal{R}_Z(\theta_R) \quad (5)$$

Rendering

The robot is displayed by rendering it using POV-Ray, from a triangle mesh model obtained from the CAD model for the robot arm. For the robot arm used in this project, the CAD model is available for download from the manufacturer's web site [5]. Each link of the robotic arm is modeled as a triangle mesh, and can be freely moved in the scene. Using the joint values computed from Inverse Kinematics routine, the links are placed in the correct position using the individual transformations T_i^{i-1} from Eq. 1, and the scene is rendered, with the result being shown in Fig. 4(a) from Section 5. A similar approach was used for rendering the laser probe and the rotary table, by creating POV-Ray models and applying the transformations T_{DK}^* and T_R^0 from Eq. 2 and Eq. 4.

4.2 Laser probe modeling

The reference frame of the laser probe is $X_L Y_L Z_L$, from Fig. 1(a), and it will be called XYZ in this section in order to simplify the notations.

Laser beam modeling

The laser probe emits a laser beam focused into a plane, such as when it intersects a surface, it casts a line or a curve. This laser beam may be modelled as a point light source, which is constrained to pass to a narrow opening (Fig. 2). In the example from Fig. 2(b), the laser rays start from origin, is projected in the positive direction of the Z axis, the plane of the laser rays is YZ and the narrow opening has the dimensions L (large edge) and W (small edge) and is located at a distance D from the origin.

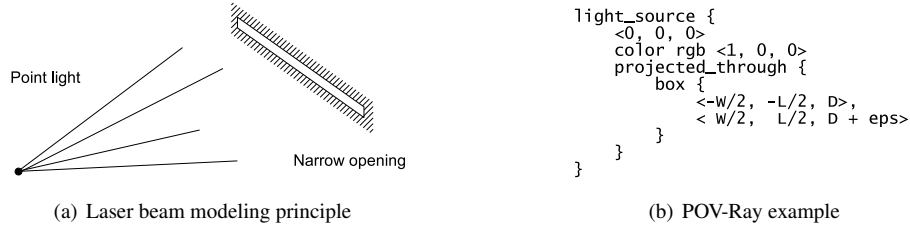


Figure 2: Laser beam modeling using POV-Ray

Camera modeling and triangulation

The cameras used in the laser probe are modeled as two standard perspective cameras, which may be implemented in POV-Ray by entering their parameters such as position, orientation and focal length. In the following text, only one of the two cameras will be described, as the other one is identical and symmetrical to the first one.

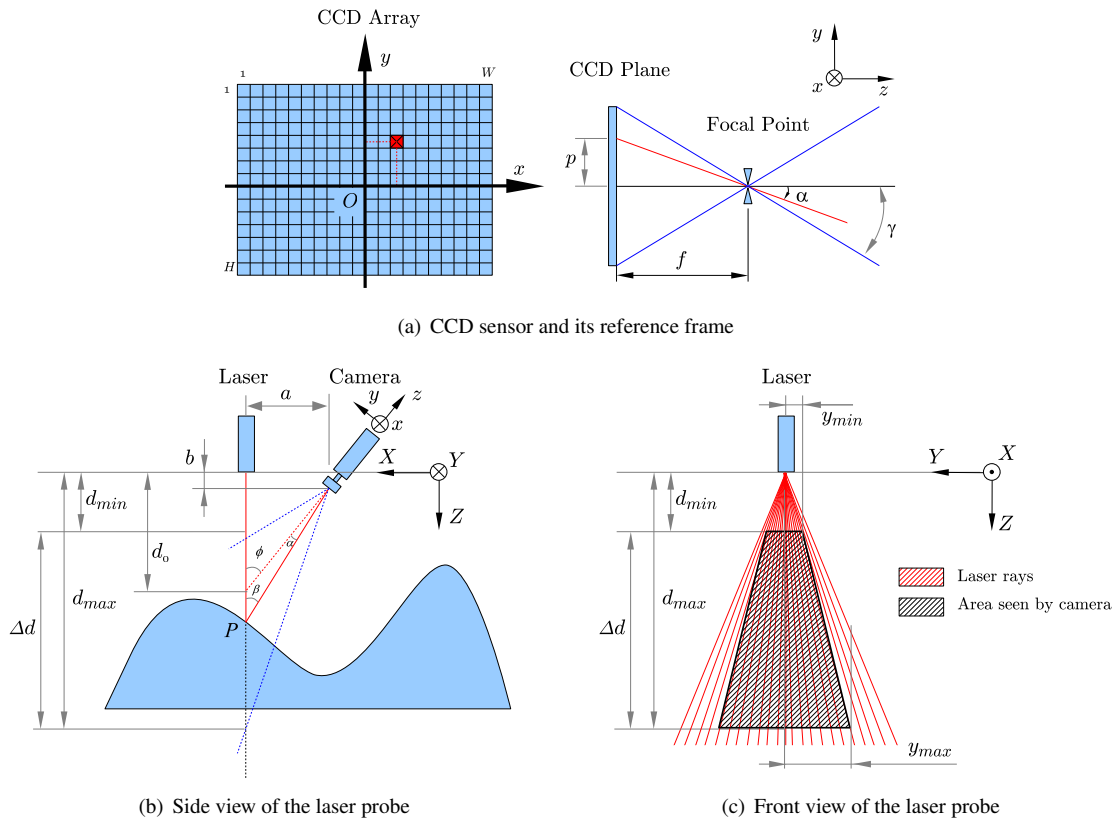


Figure 3: Triangulation

Let XYZ be the reference frame of the laser probe (Fig. 3(b) and (c)), and let xyz be the reference frame of the CCD array from the camera (Fig. 3(a) and 3(b)). Referring to Fig. 3(b), the camera position and orientation with

respect to the laser device is given by the parameters a , b and ϕ .

Using these notations, let $P = (P_X; P_Y; P_Z)$ the point of reflection of a laser ray, in the XYZ reference frame, and let $p = (p_x; p_y)$ be the coordinate of the pixel at which the ray was detected on the CCD matrix, in xy reference frame. Knowing the pixel coordinates p , the location of the point P can be expressed using the triangulation equations (6):

$$P_X = 0 \quad P_Y = \frac{a}{f \sin\left(\phi - \arctan\frac{p_y}{f}\right)} p_x \quad P_Z = \frac{a}{f \tan\left(\phi - \arctan\frac{p_y}{f}\right)} + b \quad (6)$$

where $f = \frac{H}{2 \tan \gamma}$ if the unit length is considered to be 1 pixel, i.e. the distance between two adjacent pixels on the CCD array.

The area in the plane determined by the laser rays, i.e. YZ plane in Fig. 3(c), is a trapezoid, and its limits are z_{min} , z_{max} , y_{min} and y_{max} , whose expressions are given in Eq. 7. The scanning range is thus given by z_{min} and z_{max} , and the length of the laser line L_L that is effectively being analyzed is dependent of the distance of the scanned object with respect to the laser probe, and varies from $L_{Lmin} = 2 y_{min}$ when the workpiece is close to the probe, to $L_{Lmax} = 2 y_{max}$ when the workpiece is far from the laser probe.

$$\begin{aligned} y_{min} &= \frac{a \tan \gamma}{\sin(\phi + \gamma)} & z_{min} &= \frac{a}{\tan(\phi + \gamma)} + b \\ y_{max} &= \frac{a \tan \gamma}{\sin(\phi - \gamma)} & z_{max} &= \frac{a}{\tan(\phi - \gamma)} + b \end{aligned} \quad (7)$$

Creating the point cloud

There are three main steps in computing a point cloud by processing the image obtained from the simulated camera. In the first place, one needs to identify the laser beam in the image. The simplest and fastest method is to use a gray workpiece and a red laser, and apply a thresholding operation on the saturation component of the image. The second step is to apply Eq. 6 to every pixel on the laser beam, obtaining a cloud point in the laser probe's local reference frame. The third step is to transform the coordinates of the points in an reference frame that is attached to the workpiece. The homogeneous transformation required here is computed by first expressing the point cloud coordinates in robot's World reference frame, and then expressing them in the rotary table's reference frame, using the matrices defined in Eq. 3 and Eq. 4:

$$T_{PC} = T_R^0 T_{DK} \quad (8)$$

A more advanced image processing strategy would be able to achieve sub-pixel accuracy, and may also address the reflections, ambient light and other error sources.

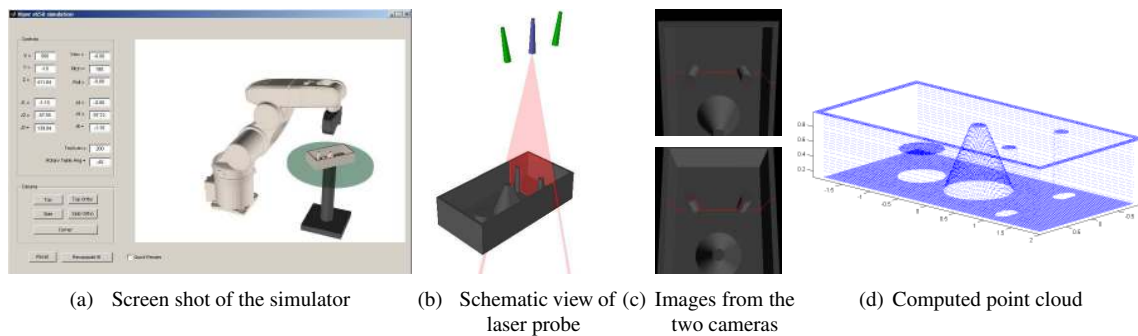
5 Simulation example

The screen shot of the simulation software is displayed in Fig. 4(a). The user can control either the position and orientation of the robot in Cartesian mode, relative to either robot base or rotary table, or the individual joint angles. The software may simulate a continuous movement of the laser probe over the workpiece, compute the images that would be seen by the two cameras, and generate a point cloud from processing them (Fig. 4(b)-(d)).

6 Summary and Conclusions

A computer simulator of a 3D laser scanning system was presented, along with the theoretical aspects involved. The software simulates the kinematics of the robot arm moving synchronously with a rotary table, and the interaction of the laser probe, which uses a laser beam and two cameras, with a virtual workpiece. A point cloud representing the workpiece can be generated by processing the rendered images. Other types of robot arms may be simulated by providing their Denavit-Hartenberg parameters and a 3D mesh model for displaying.

The bottleneck in the simulation is the rendering speed, which is currently performed by ray tracing. Therefore, real-time simulations are not currently possible. On an Intel Pentium 4-M processor at 2.0 GHz, the simulator is able to render and process around 3 frames per second by simulating a 160×160 sensor, while the speed of a real laser probe may be of the order of 100 frames per second, as described in the technical specifications.



(a) Screen shot of the simulator

(b) Schematic view of the laser probe

(c) Images from the two cameras

(d) Computed point cloud

Figure 4: Laser probe simulator

References

- [1] Mark W. Spong et.al., *Robot Modeling and Control*, John Wiley and Sons, Inc., pp. 71-83, 2005
- [2] Tom Davis, *Homogeneous coordinates and computer graphics*, 2001
- [3] Andrew S. Glassner, *An Introduction to Ray Tracing*, Morgan Kaufmann Publishers, Inc., 1989.
- [4] Persistence of Vision Raytracer Pty. Ltd., *POV-Ray Online Documentation*
- [5] Adept Technology, Inc. Web Site, www.adept.com
- [6] P.I. Corke, A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, Vol. 3, No. 1, pp. 24-32, March 1996.

Theodor Borangiu, Anamaria Dogar, Alexandru Dumitrache
University Politehnica of Bucharest
Department of Automatic Control and Industrial Informatics
Splaiul Independentei 313, Bucharest, Romania
E-mail: borangiu@cimr.pub.ro, dogar@cimr.pub.ro, alex@cimr.pub.ro