A Cross-Platform Open Source 3D Object Reconstruction System using a Laser Line Projector

Vaibhav Bajpai and Vladislav Perelman
School of Engineering and Computer Science
Campus Ring 1, Jacobs University Bremen
{v.bajpai, v.perelman}@jacobs-university.de

***Abstract*—This paper introduces a cross-platform low-cost system for 3D object reconstruction using a projection-based laser scanner. It uses contact-free measurement techniques for 3D object reconstruction and fast surface registration using Iterative Closest Point (ICP) [1]. The only hardware requirements are a simple hand-held laser line projector, a calibration rig and a standard camera. The camera is initially calibrated using Zhang's camera calibration method so that its external and internal parameters are known. The visible intersection with the known background is used to find the 3D pose of the laser plane. This laser plane is used to triangulate new 3D point coordinates of the object's surface. The point clouds obtained are processed using the "3DTK - The 3D Toolkit" [2] which includes an automatic high-accurate registration process and a fast 3D viewer.**

I. Introduction

A projection of a real-world object to an image annihilates its depth information from a geometric point of view. This is because the 3D points in the same viewing direction inherently yield a single 2D point in the image. 3D object scanning and reconstruction is a technique to acquire enough information in order to regenerate the 3D shape of such a real-world object. This is achieved by taking multiple scans of the object from different angles to subsequently register them in a common coordinate system. The generated 3D models are widely used for medical diagnosis, archaelogical analysis, industrial design and production and in the entertainment industry.

3D scanning techniques can be broadly divided into two categories depending on whether or not the scanning system is in contact with the target object. The contact-free techniques themselves can be either active or passive. Active contact-free techniques emit a radiation to calculate the deviation from the object, while passive techniques use the visible light itself for such an estimation. The authors in [3] provide a solid review of the most important 3D laser scanning methods developed in the last 20 years.

This paper uses the active contact-free technique by using a hand-held projector to emit a laser line. The recovery of the object surface is done by triangulating the laser with the rays that were projected back to the camera. We use the real-time self-calibration approach [4] to eliminate the need of an external sensor to track the position of the scanner. The pipeline initially begins with the data acquisition using an inexpensive web camera that captures multiple runs of a laser sweeping across an object as shown in Fig. 1. Camera calibration using a chessboard pattern is performed next, that helps estimate the internal parameters of the camera to establish a mapping between its natural units to the units in the physical world. We also use the chessboard patterns as our reference double-frames to calculate the extrinsic parameters of the camera. These parameters are later used to calculate the pose[1] of the laser plane.

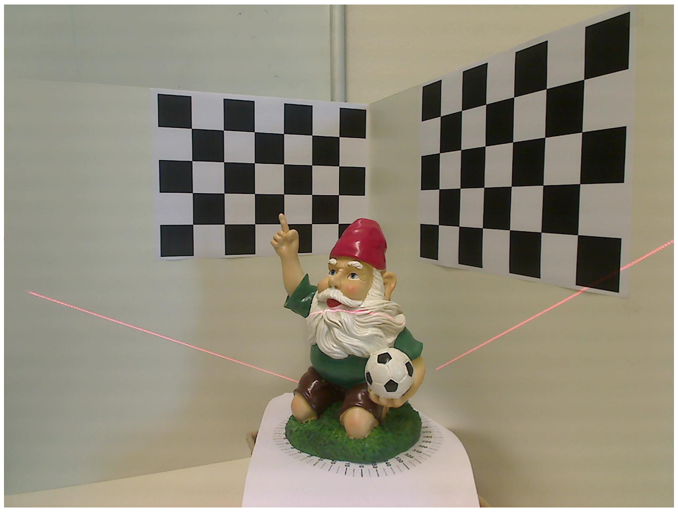
Fig. 1. Data Acquisition Setup

The quality of the final 3D shape depends largely on how accurately the laser lines and the object points are identified. We use a reference image without a laser line to calculate an image difference with each frame of the laser sweeping the object as used in [4]. The difference image is then smoothened and filtered with a red color threshold to remove the outliers. Hough transform on the result helps isolate the two laser lines on each side of the object from the object points.

The extrinsic parameters of the camera calculated using the chessboard patterns are used with the laser pixels from the hough lines to calculate the laser plane equation. The laser plane equation is finally intersected with the object points to determine the 3D surface points (point cloud) of the objects. The point clouds from each scan (with a different orientation of the object) are registered using ICP [1]. We use Simultaneous Localization and Mapping (SLAM) from 3D Toolkit (3DTK) [2] for automatic scan registration.

[1]combination of position and orientation

II. Related Work

Stereophotogrammetry [5], [6] is a technique to generate 3D models using a collection of complementary 2D images of an object taken at different angles It uses perspective methods and illumination rules to recover basic geometric models of the photographed scenes. The hardware requirements of the setup are usually two digital cameras, however the quality of the digitized 3D models is generally low.

Structured Light is an active contact-free technique where a known pattern is projected onto the surface and a separate perspective is used to observe the resulting deformed pattern. The 2D deformed image of the pattern acquired is used to extract the 3D information of the surface. Microsoft Kinect [2] internally uses this technique to compute a depth of the object by projecting an infrared laser. The major limitation of the Kinect is its inability to work well in outdoor environments.

The time-of-flight scanning approach [7] is also an active contact-free technique where the camera measures the round trip time of the light from the object surface to determine its distance. Such an approach can capture depth information over a long-range and is fairly quick in operation. Unfortunately the equipments are expensive and the accuracy is not high.

Contact-free triangulation based techniques to generate 3D models of real-world objects have been known for more than decade [8], [9]. The traditional systems however, use high-precision expensive actuators for rotating/translating the laser plane and the object. They also depend on external sensors to track the position of the scanner.

The underpinnings of our 3D object reconstruction system are largely inspired from The David Laser Scanner [4] project. It is a software package that uses the triangulation technique, however with a self-calibration method for the hand-held laser plane to keep the cost to a minimum. It is a generalization of [10] which instead uses four visual intersection points of laser with a double reference frame to calibrate the laser. The software package although originally free, is now available at a price of €299. In addition, it can only run on Windows since currently it is dependent on the .NET framework. We in this paper present a free alternative to the David Laser Scanner. It is written in C++ and is based on OpenCV and the "3DTK - The 3D Toolkit" [2] making it widely-available as a cross-platform solution.

III. Approach

The complete pipeline starting from capturing the video of the laser sweeping across an object to the end result of visualizing the 3D model of reconstructed point cloud is primarily divided into five major steps as shown in Fig. 2.

A. Data Acquisition

We use an inexpensive web camera to capture multiple runs of a hand-held laser sweeping across the object as shown in Fig. 1. Since the videos are stored in a raw format which cannot be directly processed by OpenCV [11], we use

[2]http://www.xbox.com/en-us/kinect

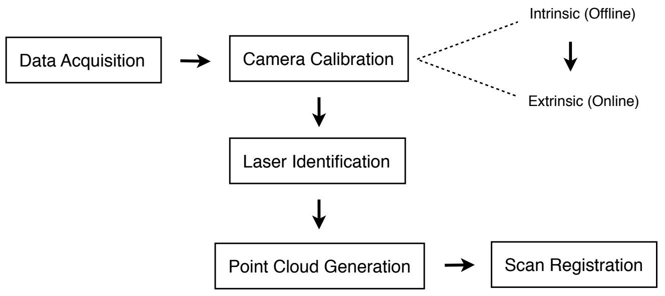

Fig. 2. Software Pipeline

`mplayer` to extract individual frames at the rate of 5 frames per second.

```
$ mplayer -demuxer rawvideo \
          -rawvideo fps=5:w=1600:h=1200:yuy2 \
          -vo pnm:ppm $FILE
```

These frames are read into memory by calling the OpenCV routine `cvLoadImage()` with a `CV_LOAD_IMAGE_UNCHANGED` flag. The routine allocates an image data structure and returns a pointer to a struct of type `IplImage`

B. Camera Calibration

The first step in the process of reconstructing the 3D geometry of the object is to establish a mathematical relationship between the natural units of the camera with the physical units of the 3D world. We use camera calibration to estimate the internal parameters of the camera and its distortion coefficients. The geometry is described in terms of camera's optical center and focal length of the camera.

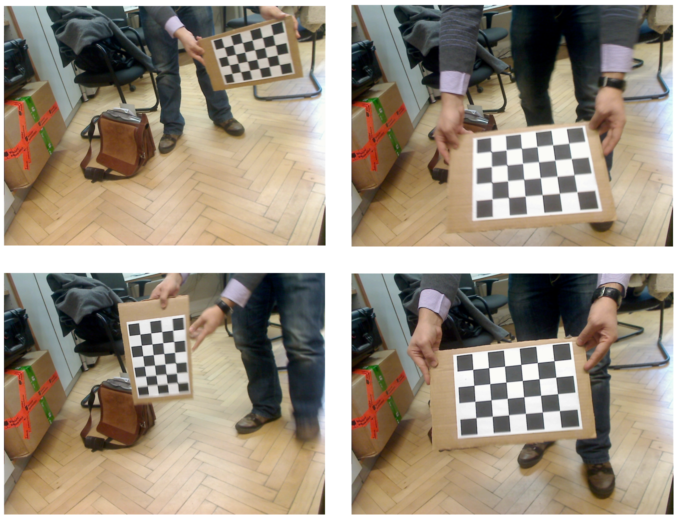

Fig. 3. Calculating the Camera's Intrinsic Parameters

We use OpenCV camera calibration routines and a planar chessboard pattern as our calibration object. OpenCV uses Zhang's method [12] to calculate the focal lengths and offsets.

However it uses Brown's method [13] to calculate the distortion coefficients. The calibration pattern is rotated and translated to provide multiple views in order to get precise information about the intrinsic parameters of the camera as shown in Fig. 3. The OpenCV routine `cvFindChessboardCorners()` is used to locate the corners and once we have enough corners from multiple images, we use `cvCalibrateCamera2()` to get the intrinsic matrix A as shown in Eq. 1.

$$s \times \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \cdot \begin{bmatrix} R \mid T \end{bmatrix} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{1}$$

$$\text{where } A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

The intrinsic matrix A is later used to describe the pose of the object being scanned by the laser relative to the coordinate system of the camera. In order to determine this pose on both sides of the target object, the patterns are masked to allow individual calculation as shown in Fig. 4. The parameters represented by $\begin{bmatrix} R \mid T \end{bmatrix}$ are then separately calculated for both the sides by calling the OpenCV routine `cvFindExtrinsicCameraParams2()`. R and T here refer to the rotation matrix and the translation vector respectively.

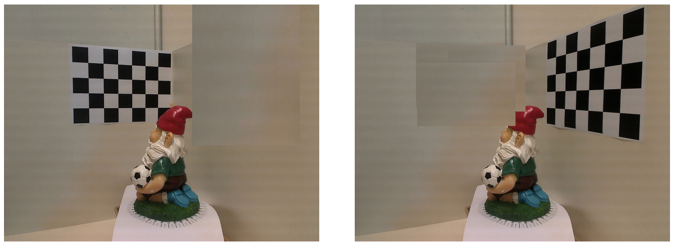
(a) $R_1 \mid T_1$ (b) $R_2 \mid T_2$

Fig. 4. Calculating the Camera's Extrinsic Parameters

C. Identification of 2D Laser Lines and Object Points

We use OpenCV routine `cvAbsDiff()` to calculate the image difference of the laser image from the reference image using Eq. 2. The resulted image difference is shown in Fig. 5

$$Z = X - Y \tag{2}$$

where X is the laser image in figure 5(a) and

Y is the reference image in figure 5(b) and

Z is the difference image in figure 5(c)

To reduce the noise in the difference image, we use the OpenCV routine `cvSmooth()` to convolve the image with a Gaussian kernel function. It not only helps to remove the camera artifacts but also reduces the information content in the image. In order to remove all outliers and keep just the pixels representing the red laser line, we use a pre-defined threshold value for the intensity of the red pixels. In order to restrict this thresholding only along the red channel, we use `cvSplit()` to split the three-channel (R,G,B) difference image into separate one-channel image planes. We use `cvGet2D()` and `cvSet2D()` to work on the scalar values of the pixels.

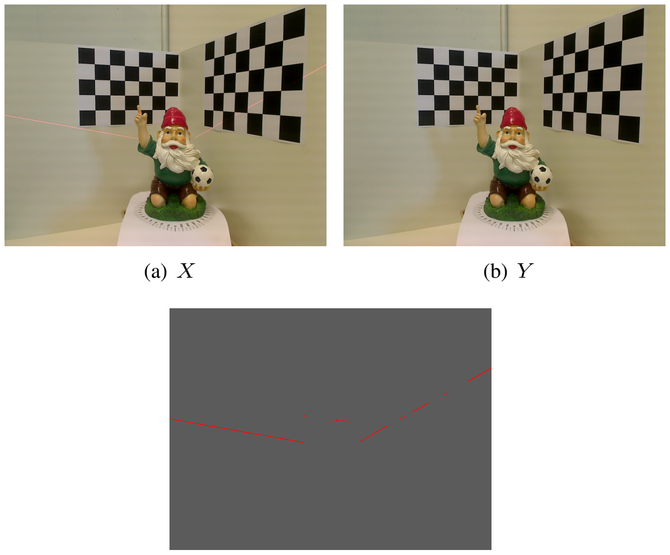
(a) X (b) Y (c) Z

Fig. 5. Using Image Difference to Find the Laser

We use the Probabilistic Progressive Hough Transform (PPHT) [14], [15] using the OpenCV routine `cvHoughLines2()` to detect the laser lines on both sides of the target object. The line end points of each line thus obtained are used to draw the line using `cvLine()` as shown in Fig. 6. The difference image is initially passed through an edge detection phase, since hough transform not only expects a gray-scale image as input but the input is also treated as binary information where the non-zero points are edge points of the image. Therefore, we use the OpenCV routine `cvCvtColor` to convert the RGB difference image to gray scale and `cvCanny()` to perform the Canny Edge Detection [16] before PPHT. The two hough line equations on either side of the object are used to find the laser line points, while the points not identified as part of the hough line are taken as target object points.

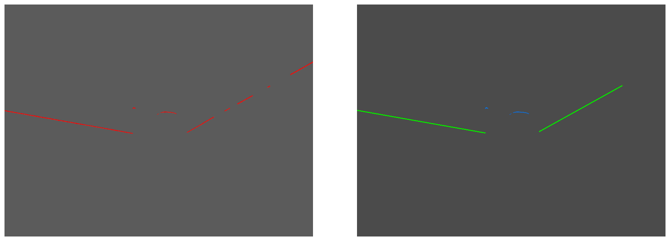
(a) Before Transformation (b) After Transformation

Fig. 6. Hough Transformation

D. Point Cloud Generation

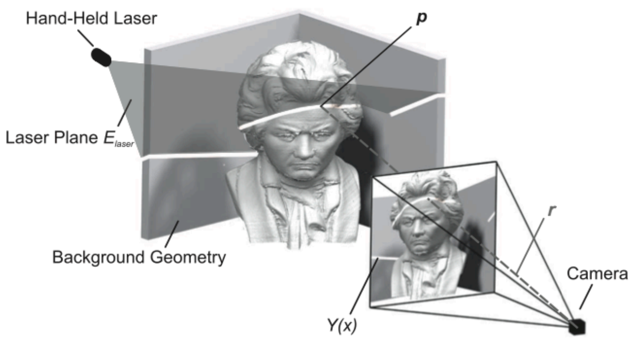

Fig. 7. Laser Triangulation [4]

The first step is to use the calculated camera extrinsics $(R \mid T)$ for each side of the target object along with the intrinsic parameters (A) to transform each laser pixel (P_c) into 3D laser surface points (P_w) using Eq. 3. The laser pixel points are expressed in the homogenous coordinate system.

$$P_w = \underbrace{s \cdot R^{-1} \cdot A^{-1} \cdot P_c}_{\overrightarrow{b}} - \underbrace{R^{-1} \cdot T}_{\overrightarrow{a}} \tag{3}$$

$$\text{where } P_c = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}, \overrightarrow{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}, \overrightarrow{b} = \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix}$$
$$\text{and } s = \frac{a_z}{b_z}$$

Next, in order to bring all the laser surface points into a common coordinate system, we transform all the 3D laser points from the right side of the target object to the coordinate system of the left side using Eq. 4

$$P_l = R_1^{-1} \times P_r - R_1^{-1} T_1 \tag{4}$$
$$\text{where } P_r = \left[R_2 \mid T_2\right] \times P_w$$

With all the 3D laser surface points in a common coordinate system, we randomly choose 3 points to generate the laser plane equation. Using the coefficients of this equation, we define the normal to the plane $(\overrightarrow{N})$ using Eq. 5.

$$E_x + F_y + G_z + H = 0 \tag{5}$$
$$\text{where } \overrightarrow{N} = \begin{pmatrix} E \\ F \\ G \end{pmatrix}$$

The last step is to use the target object pixels (P_c) and intersect them with the laser plane equation represented by the normal $(\overrightarrow{N})$ to obtain the 3D surface points of the target object (P_w) as shown in Fig. 7 using Eq. 6

$$P_w = s \times R^{-1} \times A^{-1} \times P_c - R^{-1} \times T \tag{6}$$
$$\text{where } s = \frac{\overrightarrow{N} \times \overrightarrow{a} - D}{\overrightarrow{b} \times \overrightarrow{N}} \text{ and } P_c = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

We also use the OpenCV routine `cvGet2D()` to retrieve the RGB color information for each object pixel and map it to the calculated corresponding 3D object surface point. We use the reference image as the source for retrieving the original color information since that image does not have a laser line sweeping the target object.

E. Point Cloud Processing and Registration

We use ICP [1] to register the two point clouds from different scans into a common coordinate system using Eq. 7. The algorithm requires an initial starting guess of relative poses $(x, y, z, \theta_x, \theta_y, \theta_z)$ to compute the rotation and translation to fit the 3D geometrical models together. Since our system did not have an odometer, we set the initial pose to 0 and let it extrapolate further.

$$E(R,t) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \|m_i - (Rd_j + t)\|^2$$
$$\begin{aligned} \text{where } N_m &= \text{number of points in model set } M \\ N_d &= \text{number of points in data set } D \\ w_{i,j} &= 1, \text{ if } m_i \text{ is closest to } d_j \\ w_{i,j} &= 0, \text{ otherwise} \end{aligned} \tag{7}$$

We use 6D SLAM from 3DTK [2] for automatic scan registration that uses cached kd-trees for fast iterative ICP match [17]. The `slam6D` component produces a `frames` file that is used by the `show` component to visualize the 3D model of the target object along with its color information.

IV. Experimental Results

The imaging system in addition to the source directory, expects a reference image without the laser stripe and two images of the individual background patterns used for calibration as input. The program saves the point cloud thus obtained in a destination directory which is used by the 3DTK components for processing and visualization. The results obtained from the `show` program are shown in Fig. 8.

A discernible amount of noise is evident in the final result. However, considering the quality and the price of the hardware used for the data acquisition these results can be deemed satisfactory. The gaps in the point cloud are due to the fast movement of the laser ray over the object. They can be overcome by reducing the speed of the laser and thereby producing larger number of image frames. The final step of the software pipeline, namely scan registration using the ICP method did not yield good results. One possible explanation is that the rotation angle between the two points from different scans turned out to be too large making it hard for the SLAM component in 3DTK to converge.

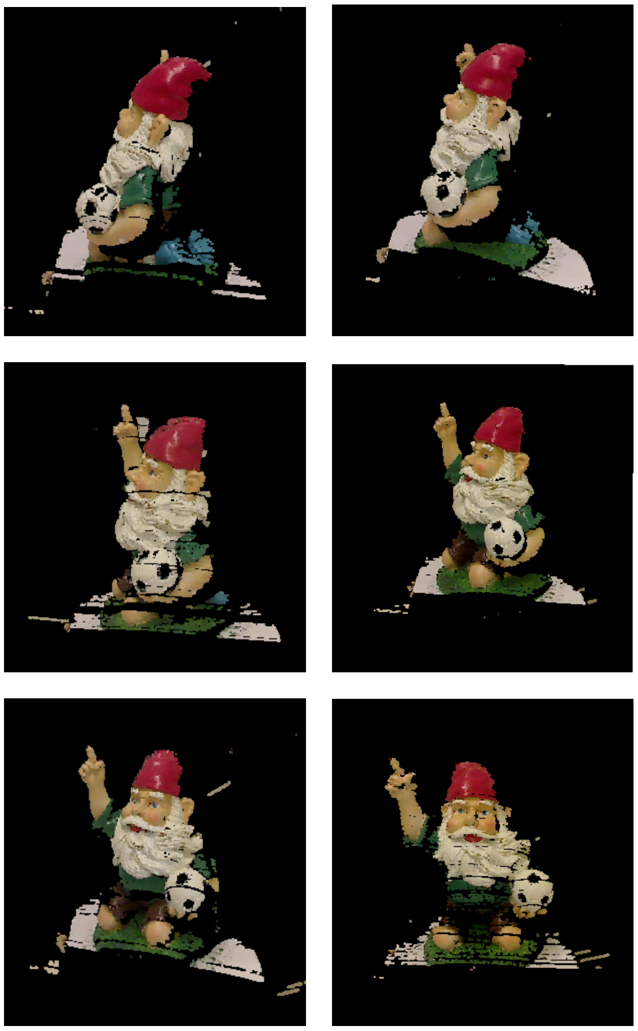

Fig. 8. Results

V. Future Work and Conclusion

The process of data acquisition, point cloud generation and scan registrations are currently performed offline. A foreseeable future work item is to make it real time to allow the results to be viewed as they are being processed. It will allow the user to not only adjust the speed of laser sweep, but also get immediate feedback on the frequency of swipes required to avoid any gaps in the final 3D model. In addition to the implementation, a performance evaluation with a larger dataset and a study comparing the results with the David Laser Scanner also needs to be done.

We presented our implementation of a 3D object reconstruction system using a hand-held laser line projector and a web camera. This work provides a free, open-source, and cross-platform alternative to the David Laser Scanner. It uses contact-free triangulation with self-calibration of the laser plane to generate the 3D object surface points. The point clouds obtained from each scan are registered using SLAM from 3DTK and viewed using its fast viewer.

References

[1] Besl, P., McKay, H.: A Method for Registration of 3-D Shapes. Pattern Analysis and Machine Intelligence, IEEE Transactions on **14**(2) (feb 1992) 239 –256

[2] Automation Group (Jacobs University Bremen) and Knowledge-Based Systems Group (University of Osnabrück): 3DTK - The 3D Toolkit. http://slam6d.sourceforge.net/ [Online; accessed 6-May-2012].

[3] Canada, C., F, B.: Review of 20 Years of Range Sensor Development * (2004)

[4] Winkelbach, S., Molkenstruck, S., Wahl, F.M.: Low-Cost Laser Range Scanner and Fast Surface Registration Approach. In: Proceedings of the 28th conference on Pattern Recognition. DAGM'06, Berlin, Heidelberg, Springer-Verlag (2006) 718–728

[5] Li, Y., Shum, H.Y., Tang, C.K., Szeliski, R.: Stereo reconstruction from multiperspective panoramas. IEEE Trans. Pattern Anal. Mach. Intell. **26**(1) (January 2004) 45–62

[6] Debevec, P.E., Taylor, C.J., Malik, J.: Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. SIGGRAPH '96, New York, NY, USA, ACM (1996) 11–20

[7] Nielsen, T., Bormann, F., Wolbeck, S., Spiecker, H., Burrows, M.D., Andresen, P.: Time-of-flight analysis of light pulses with a temporal resolution of 100 ps. Review of Scientific Instruments **67**(5) (1996) 1721–1724

[8] Beraldin, J.A., Blais, F., Cournoyer, L., Rioux, M., El-Hakim, S., Rodella, R., Bernier, F., Harrison, N.: Digital 3d imaging system for rapid response on remote sites. In: 3-D Digital Imaging and Modeling, 1999. Proceedings. Second International Conference on. (1999) 34 –43

[9] Andreetto, M., Brusco, N., Cortelazzo, G.: Automatic 3d modeling of textured cultural heritage objects. Image Processing, IEEE Transactions on **13**(3) (march 2004) 354 –369

[10] Zagorchev, L., Goshtasby, A.: A paintbrush laser range scanner. Comput. Vis. Image Underst. **101**(2) (February 2006) 65–86

[11] Bradski, G., Kaehler, A.: Learning OpenCV: Computer Vision with the OpenCV Library. Software that sees. O'Reilly (2008)

[12] Zhang, Z.: A Flexible New Technique for Camera Calibration. Pattern Analysis and Machine Intelligence, IEEE Transactions on **22**(11) (nov 2000) 1330 – 1334

[13] Brown, D.C.: Close-Range Camera Calibration. Photogrammetric Engineering **37**(8) (1971) 855–866

[14] Kiryati, N., Eldar, Y., Bruckstein, A.M.: A Probabilistic Hough Transform. Pattern Recogn. **24**(4) (February 1991) 303–316

[15] Matas, J., Galambos, C., Kittler, J.: Robust Detection of Lines using the Progressive Probabilistic Hough Transform. Comput. Vis. Image Underst. **78**(1) (April 2000) 119–137

[16] Canny, J.: A Computational Approach to Edge Detection. IEEE Trans. Pattern Anal. Mach. Intell. **8**(6) (June 1986) 679–698

[17] Nuchter, A., Lingemann, K., Hertzberg, J.: Cached k-d Tree Search for ICP Algorithms. In: Proceedings of the Sixth International Conference on 3-D Digital Imaging and Modeling. 3DIM '07, Washington, DC, USA, IEEE Computer Society (2007) 419–426