3DUNDERWORLD-SLS: An Open-Source Structured-Light Scanning System for Rapid Geometry Acquisition

Kyriakos Herakleous and Charalambos Poullis[1]

Immersive and Creative Technologies Lab,
Cyprus University of Technology

June 26, 2014

[1]charalambos@poullis.org

Abstract

Recently, there has been an increase in the demand of virtual 3D objects representing real-life objects. A plethora of methods and systems have already been proposed for the acquisition of the geometry of real-life objects ranging from those which employ active sensor technology, passive sensor technology or a combination of various techniques.

In this paper we present the development of a 3D scanning system which is based on the principle of structured-light, without having particular requirements for specialized equipment. We discuss the intrinsic details and inherent difficulties of structured-light scanning techniques and present our solutions. Finally, we introduce our open-source scanning system "3DUNDERWORLD-SLS" which implements the proposed techniques. We have performed extensive testing with a wide range of models and report the results. Furthermore, we present a comprehensive evaluation of the system and a comparison with a high-end commercial 3D scanner.

0.1 Introduction

In recent years, there is an increasing demand for 3D models and in particular 3D models representing/replicating real-world objects. Of the many available techniques ([21], [4])([1],[7]), Structured-light Scanning (SLS)([8]) systems have emerged as the most cost-effective and accurate method to capture the 3D geometry and appearance of a real object.

SLS systems employ active-sensors such as projectors and laser emitters to project light of a known structure (pattern). The scanning process involves the projection of a series of these known patterns on the object being scanned, while capturing the scene from one or more cameras. The projected pattern will be distorted according to the geometry of the object being scanned. The geometry of the object can then be computed by identifying correspondences between the pixels in the images captured by the camera(s). In order to be able to identify these pixel correspondences, the projected pattern is encoded such that every pixel in the projected pattern can be uniquely identified in the images captured by the cameras. This provides an efficient and very accurate method of mapping corresponding pixels between the images captured by the cameras. Finally, using this mapping between corresponding image pixels, it is possible to calculate their accurate 3D positions.

Many variants of SLS systems have already been proposed each one tailored to a particular task. Although the theory behind the SLS systems is well documented and understood, there are still many issues one has to consider when developing or using SLS systems, which are currently lacking documentation. In this paper, we present all the intrinsic details, limitations and solutions one has to consider when involved with the design, development and application of SLS systems. Moreover, we introduce the open-source scannins system "3DUNDERWORLD-SLS" and present the results of extensive testing.

The paper is organized as follows: Section 0.2 provides a brief overview of state-of-the-art in the area of 3D scanning, and Section 0.3 gives an overview of the scanning system. The different schemes for the encoding of the patterns are presented in Section 0.4 and the calibration of the cameras is described in Section 0.5. Section 0.6 describes the data acquisition. The captured images are decoded as explained in Section 0.7 and the reconstructed points are calculated as described in Section 0.8. Section 0.9 describes how the points can be finally converted to a mesh. In Section 0.10 we report on the results of our extensive testing and in Section 0.11 we provide a comprehensive evaluation of the scanning system.

Figure 1: System Overview

0.2 Related Work

A plethora of work has been done in the area of scanning and in particular structured-light scanning. Below we present a brief overview on the state-of-the-art in the area of

scanning systems:

Microsoft's Kinect[22] is currently attracting the attention of researchers in the area. The device employs an infrared camera and pattern projector system to enable 3D vision in order to recognize people in the scene. Many uses of this device have already been proposed for fast 3D geometry acquisition, even for full body scanning as reported in [5] [24] for 3D avatar building and 3D human animation purposes.

There are also uses of the device combined with other techniques for higher resolution results, such as the system proposed in [23] which integrates traditional stereo matching with Kinect's information to take the advantages of both. In [6] a multi-camera system for dense 3D Point cloud computation is proposed. The system uses 5 cameras and a Kinect device. In this case the Kinect device is not directly used for the 3D acquisition, but rather as a light source where the IR projector embeds features in the scene which the 4 infrared sensitive cameras are observing while the fifth is used for movement registration. Similarly, in [20] a camera flash based projector is presented for use in stereo camera systems (with two cameras) to improve stereo matching.

Nowadays, due to the improvement of computational systems, 3D geometry of real scenes [15] [16] [18] [14] or objects [19] [17] can find uses for Virtual and Augmented Reality applications even for home applications. KinectFusion [10] is a system that allows 3D reconstructions through a moving Kinect device and in addition it can be used for augmenting real scene based on its reconstructed geometry.

In [11] the authors propose a system which enables users to interact with surface particles in real world with the use of a camera, a projector and an infrared pen. The proposed system employs SLS 3D scanning technology in order to reconstruct the 3D geometry of the scene in order appropriately augment the surface particles with the projector. In [3] the use of digitization of archaeological findings is discussed for the development of integrated virtual exhibitions.

0.3 System Overview

Figure 1 shows the system overview. Firstly, given as input the resolution of the projector, a sequence of patterns is encoded. Two of the most effective ways of encoding this type of information are presented in Section 0.4. Secondly, the cameras-projector system is geometrically calibrated and the intrinsic and extrinsic parameters are calculated as described in Section 0.5. Thirdly, the sequence of patterns is projected on the object and images are captured by the cameras for each pattern in the sequence, as described in Section 0.6. Finally, the captured images are decoded in order to derive a pixel-to-pixel mapping between the images captured by the cameras as explained in Section 0.7 and is then used to reconstruct a pointcloud of the object using triangulation as described in Section 0.8.

0.4 Encoding in Temporal Domain

The first step of the process is the encoding of the information into patterns in the temporal domain. This involves the generation of encoded patterns which when projected

in sequence on the object being scanned they allow the unique identification of each of the projector's pixels. In order to achieve this, the encoding of the information is performed in the temporal domain i.e. the encoding is a function of time and all encoded patterns in the sequence are required in order to uniquely identify each pixel.

There are several schemes for encoding information into sequences of patterns, the most popular being the Binary-code and Gray-code encoding which are performed in the temporal domain. In both cases, the information about the two image axes X, Y is encoded separately into a different pattern. A projector P with resolution P_{res_x}, P_{res_y}, will result in $N_{cols} = \lceil \log_2(P_{res_x}) \rceil$ encoded patterns representing the columns, and in $N_{rows} = \lceil \log_2(P_{res_y}) \rceil$ encoded patterns representing the rows. For example a projector with resolution $1024x768$ will result in $N_{cols} = 10$ and $N_{rows} = 10$ i.e. a total of 20 patterns.

0.4.1 Binary Encoding

In the binary encoding, the decimal number D_{c_i} corresponding to each column c_i where the index $i \in P_{res_x}$ is converted to its binary form B_{c_i}. Each individual bit $b^k_{c_i}$ where $k \in [1, |B_{c_i}|]$, of the binary form B_{c_i} is then marked in the k^{th} pattern in the sequence, with black color if its value is 0 or white color if its value is 1. Note, that the length of the binary form B_{c_i} equals the number of patterns N_{cols}. Similarly, this process is repeated for each row r_j where $j \in P_{res_y}$.

An example of the process is shown in Figure 2 where the binary form of a single pixel p at location $(209, 660)$ is encoded into two sequences of $1x1$ patterns in temporal domain; one sequence representing the column and the other the row information. The column sequence represents the binary form of the column's decimal number i.e. $B_{c_p} = 1000101100$ and the row sequence represents the binary form of the row's decimal number i.e. $B_{r_p} = 0010100101$. As it can be seen, each pattern corresponds to a bit in the binary pattern and is represented with either the color black or white.

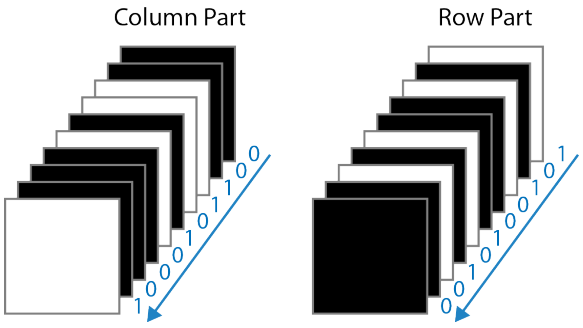


Figure 2: Encoding of a single pixel p at location $(209, 660)$ is encoded into two sequences of $1x1$ patterns in temporal domain; one sequence representing the column and the other the row information.

By iteratively performing the process for every pixel $p_{(x,y)}$ in the projector where $x \in res_x, y \in res_y$, two sequences of 2D patterns are produced representing all the columns and all the rows respectively. Figure 3 shows an example of the final

sequences for the columns and rows. The patterns are shown in the order they are projected as indicated by the time arrow i.e the first pattern to be projected corresponds to the most significant bit and appears as the last one in the diagram, followed by the remaining. As time progresses the frequency of the pattern increases until it reaches the highest frequency which represents the least significant bit in binary pattern.

The result is a sequence of binary encoded patterns where each pattern represents one bit of the binary sequence. This produces images containing vertical stripes in the case of the columns, and horizontal stripes in the case of the rows.

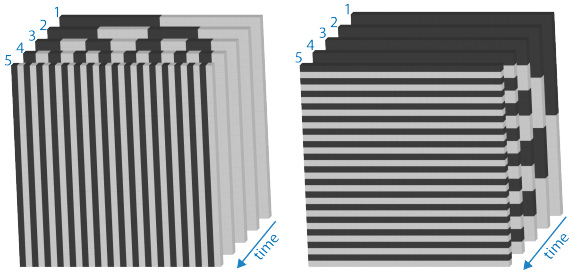


Figure 3: Encoding the information into a sequence of patterns in temporal domain.

0.4.2 Gray-code Encoding

The Gray-code encoding works in a similar way as the binary encoding previously described, however it ensures that there is only one bit different between consecutive patterns. Figure 4 shows a comparison between Gray and Binary encoding for the columns of an 8x8 area. As it is evident, in the case of Gray codes there is a maximum of one bit difference between any two consecutive encoding of values. Figure 6b shows a sequence of encoded patterns using Gray-codes, corresponding to the same area of 8x8 as in Figure 6a.

Decimal Value	Gray-code	Binary code
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111

Figure 4: Comparison between Gray codes and binary codes. In the case of Gray codes there is only one bit difference between any two consecutive encodings.

Gray codes can be calculated by first computing the Binary representation of a number and then converting it as follows: copy the most significant bit as is, and then for the remaining bits (taking one bit at a time), replace with the result of an XOR operation of the current bit with the previous bit of higher significance in the binary form. An

example is shown in Figure 5a for the computation of the Gray code for the decimal number 93.

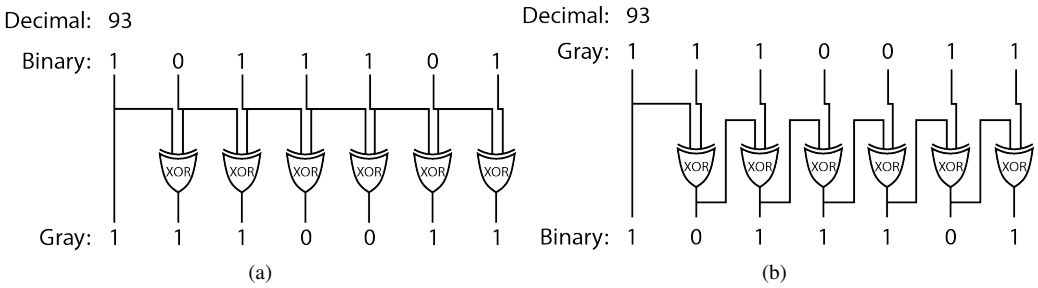


Figure 5: (a) Encoding of decimal number 93. The Binary Code representing the decimal number 93 is converted into Gray Code. (b) Conversion of Gray Code to Binary Code.

Similarly, the conversion of a Gray code to the corresponding Binary code is as follows: copy the most significant bit as is and for the remaining bits (taking one bit at a time), replace with the result of the XOR between the current bit in the Gray code and the previous bit of higher significance in the Binary code. Figure 5b shows the computation of the Binary code from the Gray code representing the decimal number 93.

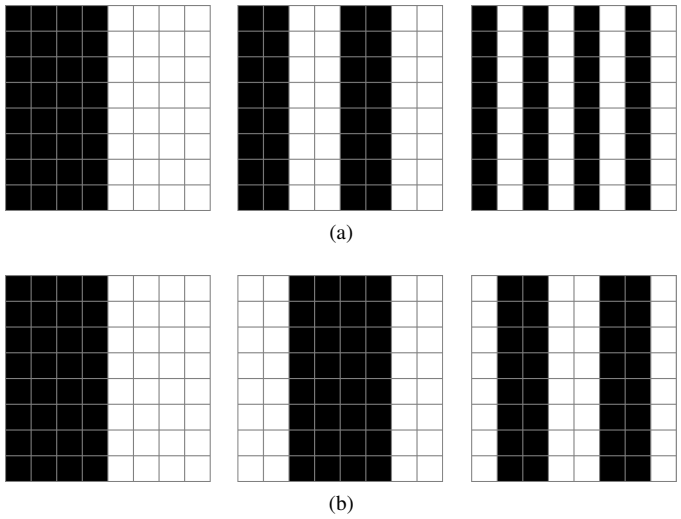


Figure 6: (a) Binary encoding of the columns for an area of 8x8 pixels. (b) Gray encoding corresponding to the same columns of the area in (a).

Although the two schemes presented above result in similar patterns, the Gray encoding is considered as a better alternative to the Binary Codes. When using Gray codes the value of the least significant bit changes after two consecutive stripes, in contrast to the Binary codes where the value changes at every pattern. This can be seen in Figure 6a, 6b where the pattern using Gray encoding corresponding to the least significant bit, contains stripes with thickness of two pixels, as opposed to the pattern using Binary encoding corresponding to the least significant bit which contains stripes with thickness

of a single pixel. Larger stripe thickness is preferable since it can considerably reduce unwanted effects such as color bleeding in the projection or/and in the captured images.

0.4.3 Implementation Issues

The generation of the two sequences of pattern is performed using Gray encoding as described above. However, there are two important implementation issues to consider:

- Firstly, the choice of the colors to be used in the patterns. The patterns can have any two colors, although traditionally black and white has been used. In any case, each pixel must have an intensity value which will be used as a threshold to determine the value of the pixel i.e. 1 or 0. Therefore, it is imperative to take this into account when choosing the two colors and choose colors which will result in large intensity differences. A poor choice on colors, can otherwise jeopardize the entire process by failing to distinguish a pixel's value later on in the process. For this reason and to overcome this limitation, it is preferable to project a pattern followed by a projection of its color-inverted pattern. Inverted pattern images are images with the same structure as the original but with inverted colors. This provides an effective method for easily determining the intensity value of each pixel when it is lit (highest value) and when it is not lit (lowest value). The threshold for the intensity value of a pixel p with a highest value p_h and a lowest value p_l can then be computed as the average $\tau_p = (p_h - p_l)/2$.
- Secondly, identifying shadow regions. The cameras observe the object from different angles than the projector, hence quite often there will be a viewing areas which lies in shadow regions. Thus, it is preferable that pixels falling under a shadow region are removed at the early stages of the process. This can be achieved by projecting a white and then black image on the object and capturing the images. By evaluating the intensity value of the pixels in the images, one can determine which pixels fall under a shadow region by identifying cases where the intensity values in the two captured images are very similar. A "shadow mask" can then be created which leaves only the pixels which do not fall under shadow regions. This can considerably reduce computational processing time. Section 0.7.1 explains how to compute the shadow mask.

Thus, the final projection sequence contains the column and row pattern sequences encoded using Gray-code, their inverted patterns, as well as two images of solid colors, one for each used color. The patterns are projected in sequence as follows: first the two solid color images, then interchangeably the column and its inverted sequence, followed by interchangeably the row and its inverted sequence.

0.5 Camera Calibration

When considering a camera, the image is formed when light rays being reflected in the scene, pass through the camera lens, through the aperture and interact with the photo-sensitive light sensor. If there had been no light sensor present, then the light rays would converge at a single point called the center of projection as shown in Figure 7. A projector can be thought of as an inverted camera. The light rays start from a point

i.e. light bulb inside the projector, then go through mirrors and/or lenses and finally interact with the scene.

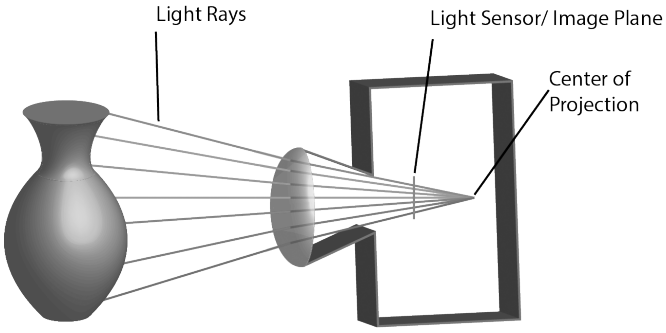


Figure 7: Image formation. Light rays being reflected in the scene, pass through the camera lens, then through the aperture and interact with the photo-sensitive light sensor.

In both cases (camera, projector), one can relate a light ray to a particular pixel if the geometry of the system at the time the image was taken is known. The geometry is defined in terms of a set of intrinsic and extrinsic parameters which can be computed by performing a geometric calibration.

To calibrate the cameras, an object with known geometric characteristics is used; usually a flat board containing a printed checker pattern. By taking images of the board at different positions and orientations as explained in [25] we can compute the intrinsic and extrinsic parameters which specify the camera matrix C in Equation 1,

$$C = \underbrace{\begin{bmatrix} \alpha & -\alpha cot(\theta) & u_0 \\ 0 & \frac{\beta}{sin(\theta)} & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{intrinsic} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}}_{extrinsic} \tag{1}$$

where $\alpha = kf_x$, $\beta = kf_y$, (f_x, f_y) is the focal length on the x and y axis respectively, θ is the skew angle, u_0, v_0 is the principal point on the x and y axis respectively and r_{1-3}, t_{x-z} determine the camera's rotation and translation relative to the world.

Given a minimum of four 2D to 3D correspondences specified interactively by the operator the camera extrinsic and intrinsic parameters can be accurately estimated. The camera pose estimation is performed using a non-linear Levenberg-Marquardt optimization [12, 13] which minimizes the error function E_{C_k} for each camera C_k,

$$E_{C_k} = \frac{1}{n} \sum_{i=0}^{n} \sqrt{(I_x^i - P_x^i)^2 + (I_y^i - P_y^i)^2)} \tag{2}$$

where I^i is the ith image point, P^i is the projection of the ith 3D world point and n is the number of 2D to 3D correspondences.

In addition to the camera matrix C as defined in Equation 1, the intrinsic parameters include the distortion coefficients; three coefficients for the radial distortion and two coefficients for the tangential distortion.

Although, theoretically the camera matrix C can be computed from a single image of the flat board, it is recommended that 10-20 images of the flat board at different orientations and positions are used.

Multi-Camera System Extrinsics: In order to calculate the extrinsic parameters of each camera, its intrinsic ones must be already calculated and then using one picture, where the origin of the world is specified, the rotation and translation based on the world origin can be estimated. To achieve this, all cameras in the system must capture the scene at the same time, in order to take the same position and orientation of the board,as shown in Figure 8. In this way the same world origin can be used in order to relate the motion between the cameras i.e. the rotation and translation, as explained later in the Section 0.8.

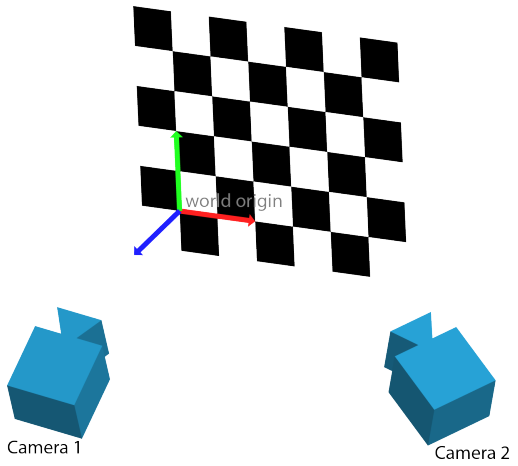


Figure 8: Multiple Camera Calibration. For each orientation and position of the calibration board, images are captured from all cameras. This allows the use of a common reference point i.e. world origin.

0.5.1 Implementation Issues

- Once the calibration is performed, there should be no movement to any part of the system otherwise a re-calibration will be needed.
- The camera calibration in 3DUNDERWORLD-SLS is implemented using OpenCV's calibration functions. Another well-known and established calibration toolbox is the Camera Calibration Toolbox for MatLab [2].

0.6 Acquisition

The acquisition is a relatively straight forward process. Each image of the sequence is projected on the object and every camera in the system captures an image. However, there are several things to consider:

- The object should remain static during the acquisition.

- The cameras' and the projector's settings (exposure, brightness etc.) should be adjusted according to the lighting in scene. Indirect light reaching the scene from other sources should be eliminated if possible in order to achieve better results.
- If the acquisition process is automated then it is recommended that there is a "forced-delay" after each image capture until the cameras confirm that the last image was captured and stored successfully. This will reduce and/or eliminate unwanted artifacts which may appear because the projector delayed the projection of a pattern or the camera delayed the capturing an image.

An acquisition set consists of the images captured by each camera for each pattern in the sequence. However, from each acquisition set, only the parts of the scene which are visible to all the cameras can be reconstructed (Figure 9), thus more scans may be required to obtain all sides of the object. This can be achieved by changing the object's position and orientation between each acquisition set while ensuring that no changes occur in the cameras and projector.

(a) (b) (c)

Figure 9: Result of a single scan. (a) and (b) show the two camera views and (c) the reconstructed mesh. In same cases more scans are needed in order to obtain the desired area.

0.7 Decoding of Captured Images

Following the acquisition of the data, the next step is the decoding of the captured images. This involves the calculation of the shadow masks and the decoding of the patterns.

0.7.1 Computing the shadow mask

The goal of this process is to determine which pixels in a camera's image fall under shadow regions. This involves comparing each pixel's intensity values between the two first projections: the black and white images. Pixels whose intensity values in the two captured images corresponding to the black image projection and white image

projection is large, are considered as a valid pixels. If the difference is small then the pixel is marked as "in-shadow". Figure 10 shows an example of a shadow mask.

(a) (b)

Figure 10: Shadow mask example. (a) Image captured by the camera. (b) Shadow mask - all black pixels are removed from subsequent processing.

0.7.2 Decoding the patterns

The cameras capture images of the encoded projected patterns. The next step is to decode each pixel in the captured images into their corresponding decimal number representing the column and the row. This provides a mapping between the pixels in the cameras i.e. pixels in the captured images which correspond to the same projector pixel as shown in Figure 11.

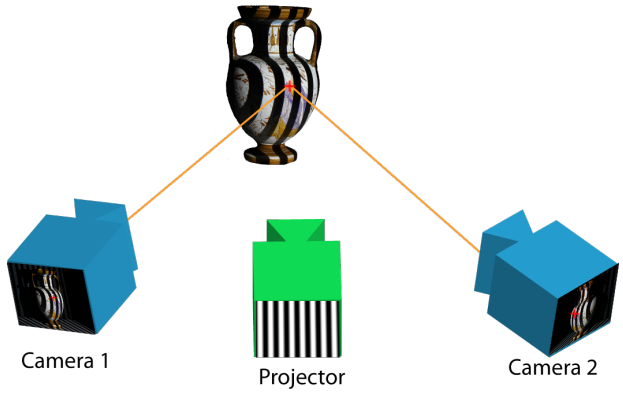


Figure 11: A 3D point is viewed by two different cameras. The decoding aims at deriving a mapping between the pixels of the two cameras, corresponding to the same 3D point.

More specifically, the decoding is performed as follows:

- Determine whether a pixel p is lit or not (1 or 0) in the images capturing the projected sequence of patterns encoding the columns.
- Calculate its binary form B_p.
- Convert the binary form B_p into the equivalent decimal number x.

Similarly, the process is repeated for the images capturing the projected sequence of patterns encoding the rows and results in a decimal number y. Thus, (x, y) are the image coordinates of the projector's pixel corresponding to the pixel being decoded in a camera. By repeating the process for all the cameras, we can map the pixels not only to the projector's pixels but rather to the other cameras viewing the object. It should be noted that a camera pixel may be mapped to more than one pixels of another camera due to differences between the camera and projector resolutions.

0.8 Reconstruction

The decoded captured images result in a set of a many-to-many mappings between the pixels of the different cameras. Next, by triangulating the rays corresponding to each pair, a 3D point is computed at their intersection. The projection of this point falls onto the mapped pixels in the different cameras.

0.8.1 Pixel-to-Ray

A camera ray is a straight line in 3D Euclidean space that starts from the camera's center of projection (Figure 7), intersects the image plane and extends outward to the scene. The ray can be defined with a single point of the ray and the ray's direction vector.

In order to compute the direction vector two ray's points are needed. The first point is the camera's center of projection $q_{center}^{camera} = (0, 0, 0)$. The second point is the point corresponding to the pixel from which the ray passes through. That point can be computed by first finding the undistorted pixel's position, using the distortion coefficients computed during camera calibration, and then "unprojecting" the pixel into 3D space. This is achieved by multiplying the inverse of the camera matrix C with the pixel's coordinates as given by the Equation 3,

$$q_{pixel}^{camera} = C^{-1} \times \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3}$$

Therefore, both q_{center}^{camera} and $q_{pixel}^{ccamera}$ are in the camera's local coordinates frame and they must be converted to world coordinates using the extrinsic parameters computed during the camera calibration as follows,

$$q^{world} \quad = \quad R^{-1} \times q^{camera} - R^{-1} \times T \tag{4}$$

where R is the rotation matrix and T is the translation vector of the camera relative to the world origin shown in Figure 8.

Thus, the ray corresponding to pixel p is defined as $R_p^{world} =< p_{center}^{world}, \vec{v}_{dir}^{world} >$ in world coordinates, where $\vec{v}_{dir}^{world} = p_{pixel}^{world} - p_{center}^{world}$.

0.8.2 Ray Triangulation

Given one pixel in image I_1 and its corresponding pixel in image I_2, two rays are formed as described above, and their intersection is computed. In practice, the triangulation of two rays in 3D space can be considered as an ill-posed problem since quite often the two rays do not intersect 'exactly' but rather pass by one another in close proximity.

In order to overcome this limitation, the segment perpendicular to the two rays with the shorted distance is computed, and the middle point of the segment is considered to be the intersection point. Figure 12 shows an example where the segment ab is the shortest line connecting the two rays A and B. The mid-point p is considered to be the intersection point.

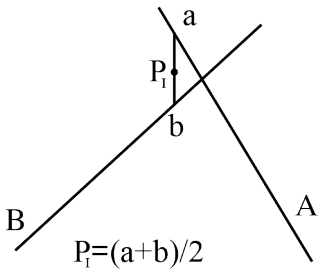


Figure 12: Ray intersection. In cases where two rays do not intersect 'exactly', the intersection point is considered to be the midpoint of the shortest segment connecting the two rays.

Computation of the Intersection Point: Consider two lines in 3D space A and B passing through points p and q, with direction vectors $\vec{u}$ and $\vec{v}$ respectively, and let the two closest points on the lines be a and b, as defined in Equation 5 and Equation 6, where s and t are scalar values.

$$a = p + s * \vec{u} \tag{5}$$

$$b = q + t * \vec{v} \tag{6}$$

The segment connecting a and b is perpendicular to the lines, hence the dot product of their vectors is equal to 0 as follows,

$$(a - b).\vec{u} = 0 \tag{7}$$

$$(a - b).\vec{v} = 0 \tag{8}$$

and with Equations 5 and 6, the Equations 7 and 8 become,

$$\vec{w}.\vec{u} + s * \vec{u}.\vec{u} - t * \vec{v}.\vec{u} = 0 \tag{9}$$

$$\vec{w}.\vec{v} + s * \vec{v}.\vec{u} - t * \vec{v}.\vec{v} = 0 \tag{10}$$

where

$$\vec{w} = p - q \tag{11}$$

From Equation 9 and Equation 10 it follows that,

$$s = \frac{\vec{w}.\vec{u} * \vec{v}.\vec{v} - \vec{v}.\vec{u} * \vec{w}.\vec{v}}{\vec{v}.\vec{u} * \vec{v}.\vec{u} - \vec{v}.\vec{v} * \vec{u}.\vec{u}} \tag{12}$$

$$t = \frac{\vec{v}.\vec{u} * \vec{w}.\vec{u} - \vec{u}.\vec{u} * \vec{w}.\vec{v}}{\vec{v}.\vec{u} * \vec{v}.\vec{u} - \vec{v}.\vec{v} * \vec{u}.\vec{u}} \tag{13}$$

The intersection point is the average of the 2 end-points of the segment as follows,

$$P_i = \frac{(p + s * \vec{u}) + (q + t * \vec{v})}{2} \tag{14}$$

Thus, for all ray-pairs the intersections are computed as above in order to avoid any possible problems with non-intersecting rays.

0.8.3 Implementation Issues

There are several limitations with the chosen triangulation method that should be taken into account. Firstly, although two rays (as defined in our system) should not be parallel, it is a good practice to check for near-parallel conditions as well. In order to to so, it is suggested that a near zero condition is checked on the denominator of the Equations 12 and 13.

Another important note is the fact that in practice the mapping between pixels is not one-to-one but rather many-to-many i.e. an area of pixels in one image maps onto another area of pixels in another image. This is due to the fact that the cameras and projector resolutions are different. In situations like these it is suggested that the average of the intersection points is taken as the final intersection point. Averaging the resulting intersection points not only reduces the amount of generated geometry by removing duplicates bu also results in smoother geometry.

Finally, the linear triangulation described in 0.8.2 is the simplest triangulation method and not necessarily the best one. In [9] the triangulation problem is discussed extensively, and the authors propose an alternative method which outperforms the linear triangulation discussed above. We have found through extensive testing that linear triangulation performs very well; at least in our case.

0.9 Point Cloud to Mesh

Thus far, we have described how to generate a cloud of 3D points in Euclidean space that represents the scanned scene. In many cases this type of output is not easily usable, since it is difficult to manipulate. For this reason, we may first need to convert it to a 3D mesh, where the points (vertices) are interconnected (with edges) to each other producing faces. Many methods already exist with various complexities and resulting output qualities. In this paper a quick conversion method will be described which takes advantage of the spatial relation between the 3D points and the projector's pixels.

As already mentioned, each point in the cloud corresponds to a projector's pixel; in fact each point is representing the area in the scene that was lit by the related projectors' pixel. Based on this fact, areas that are lit by neighbouring pixels are next to each other,

2nd	1st	2nd
1st	p	1st
2nd	1st	2nd

Figure 13: Neighbouring pixels. 1st level neighbours of a pixel p are the previous, next, above and below pixels. 2nd level are the diagonal pixels.

and similarly the points relating to those pixels are neighbours as well. Neighbours of a pixel p are considered the 8 surrounding pixels, where the ones placed in the same row or column with p are 1st level neighbours, while the diagonals are 2nd level neighbours as it can be seen in Figure 13.

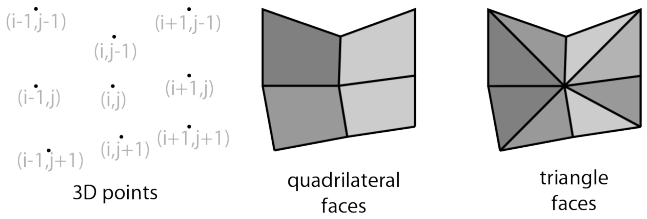


Figure 14: Conversion of point-cloud to mesh. 1st level connections result to a quadrilateral faces. 1st and 2nd level result to triangular faces

Points of neighbouring pixels are iteratively connected to each other forming faces. Most commonly, faces can be either triangular or quadrilateral. In order to form triangular faces both 1st and 2nd level neighbours are interconnected, in contrast to quadrilateral forming where only 1st level neighbour connections are needed as shown in Figure 14. Meshes consisting of triangular faces have higher complexity than the quadrilateral ones, however they often appear to form smoother surfaces.

The described method can introduce some artifacts in the output mesh in cases where some pixels are not visible by more than one camera in which case holes may appear in the mesh, or cases where neighbouring pixels are projected on different surfaces in the scene resulting connections between them. However, this method is generic and quite fast since it is performed in projector's image space and works very well for most of the scene's area. Possible hole problems can be filled by another scanning and possible connections between wrong faces which are seldom found can be easily removed manually. In addition, it can accommodate extra information such as the color of the mesh and it is even suitable for cases where the scanned object is decorated with structured color motives or other images.

0.10 Experimental Results

0.10.1 Apparatus

In this work we focus on a system consisting of two digital cameras with manual (fixed) focus mode (as opposed to autofocus) and a DLP projector. [1] It is important that the projector's pixels are clearly captured in the images taken by the cameras, thus it is preferable to use a camera which has higher resolution than the resolution of the projection. This however will not affect the resolution of the final 3D model since the reconstruction takes place in projector's space i.e. it will have the resolution of the projector. The process of capturing the images can be done either automatically using a connected computer, or manually. In the latter case, it is imperative that the setup remains static. Even a slight movement can result in misalignment and severe reconstruction errors.

In the open-source scanning system 3DUNDERWORLD-SLS v3.x, we employ two Canon SLR cameras EOS-1D Mark IV with a resolution of 4896x3264, and an in-Focus IN110 portable projector with a resolution of 1024x768. All three devices are connected to a portable computer which runs the software for the scanning process. Each of the encoded images (total of 42 for a resolution of 1024x768), generated as previously explained, are projected on the object and at the same time each connected camera captures a photo of the scene. At the end of the scanning process these captured images will be processed in order to compute the 3D geometry of the scene.

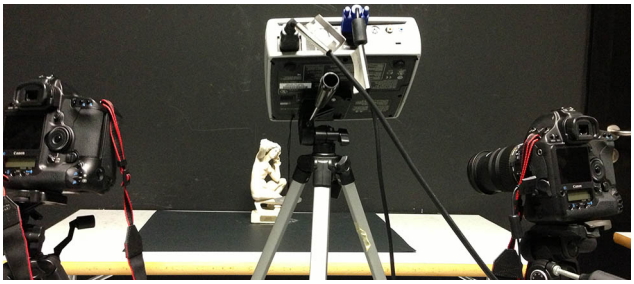


Figure 15: Example of two cameras setup, with the cameras left and right of the projector.

0.10.2 Setup

As this technique is very general, there are several acceptable ways to position the imaging devices. The positioning is subjected to the size of the scanned object, the number of the cameras used and the type of their lenses. Generally it is good practice to have the cameras in non-parallel positions in order to get smoother results. Note however that as the angle between the cameras increases the areas of the object that can be reconstructed decreases since the area visible to **all** cameras reduces. For this reason, we suggest to position the projector in the middle of the cameras as shown in Figure 15.

[1] 3DUNDERWORLD-SLS v1.0 requires one web camera.
3DUNDERWORLD-SLS v2.x requires one Canon SLR camera.
3DUNDERWORLD-SLS v3.x requires two or more Canon SLR cameras.

Model	Real	Recon.	Size	Pts	Tri.
Aphrodite			$88cm\times$ $20cm\times$ $20cm$	602685	1205406
Aztec			$22.5cm\times$ $10.5cm\times$ $6cm$	540745	1054972
Caryatid			$24.5cm\times$ $5.5cm\times$ $5.5cm$	659777	1319544
Aphrodite			$25.5cm\times$ $19cm\times$ $14.5cm$	2028162	4056320

Table 1: Videos of the reconstructed models can be found at: `http://www.vimeo.com/TheICTLab`

Another important aspect is to adjust the optics to the object's size. For better results, the projector can be set in a way such that most of the area being lit lies on the object; the more 'lit' pixels project on the object the higher the resolution of the resulting pointcloud. The same rule of thumb applies to the cameras too: it is crucial to ensure that cameras are in close proximity such that the projection is visible down to the pixel level. Additionally, the more camera pixels capture one projector's pixel the smoother the reconstructed model will be.

After the placement of the imaging devices, it is important to set the focus on the object's surface in order to have a sharp projection and sharp captured images. Before starting the scanning process the system must be calibrated.

0.10.3 Scanning & Reconstruction

The proposed algorithms and implemented techniques were extensively tested and the results are reported. All results shown below were generated with the developed open-source scanning system 3DUNDERWORLD-SLS v3.1. Several objects were scanned and information about their structural size in real-life, the number of points and the number of triangles of the reconstructed models are presented in Table 2 and Table 3.

Figure 19 shows a render of a genuine replica of an amphora (right) next to a photo of the original (left). The amphora has dimensions $18cmx10cmx10cm$ and the resulting object contains 1276766 points and 2553536 faces.

0.11 Evaluation

The evaluation of the proposed technique is performed by measuring the following four evaluation metrics: linearity, orthogonality, sampling rate, accuracy.

(a) (b)

Figure 16: Reconstructed model of a genuine replica of an amphora. (a) Photo of the object. (b) A render of the reconstructed model.

0.11.1 Linearity metric

A perfectly flat plane Π is scanned and a plane $\Pi_{fitted} = \langle \alpha, \beta, \gamma, \delta \rangle$ with normal $N_{\Pi_{fitted}} = \langle n_x, n_y, n_z \rangle$ is fitted on the resulting $\aleph$ 3D points. The plane fitting is perfomed using RANSAC and the average error E_{avg} and average $RMSE$ is computed as follows,

$$E_{avg} = \sum_{i=0}^{\aleph} |\delta - (\alpha \times \aleph_i^x + \beta \times \aleph_i^y + \gamma \times \aleph_i^z)|/||\aleph|| \tag{15}$$

$$RMSE = \sqrt{\frac{\sum_{i=0}^{\aleph} \{\delta - (\alpha \times \aleph_i^x + \beta \times \aleph_i^y + \gamma \times \aleph_i^z)\}^2}{||\aleph||}} \tag{16}$$

We scan several different planar surfaces shown in Table 2 and perform plane fitting using RANSAC. This metric is measured as the distance of all points from the fitted surface in terms of the average error E_{avg} and the root-mean-square error $RMSE$. As it can be seen from the reported measured the error is minuscule when considering the total number of points. It should be noted that for the box example, the three orthogonal planes were measured separately.

0.11.2 Orthogonality metric

A set of three perpendicular planes were scanned and three planes Π_1, Π_2, Π_3 are fitted respectively using RANSAC. The orthogonality metric is defined as the magnitude of the three dimensional vector containing the measured angles between the three planes in terms of the dot product as follows,

$$E_{ortho} = \langle dot(N^1_{\Pi_{fitted}}, N^2_{\Pi_{fitted}}), \\ dot(N^1_{\Pi_{fitted}}, N^3_{\Pi_{fitted}}), dot(N^2_{\Pi_{fitted}}, N^3_{\Pi_{fitted}}) \rangle \tag{17}$$

Object	Left Image	Right image	Linearity		Points
			E_{avg}	$RMSE$	$\aleph$
Flat Plane Π_1			0.0046794643	0.0083896662	490625
Flat Plane Π_2			0.0109797063	0.0186112309	359659
Flat Plane Π_3			0.002567169	0.004390225	540160
Box Plane Π^1_{Box}			0.0012361568	0.0015234051	59210
Box Plane Π^2_{Box}			0.0064742412	0.0090599699	24980
Box Plane Π^3_{Box}			0.0264965185	0.0352513109	24464

Table 2: Linearity metric. This metric is measured by fitting planes using RANSAC to the data and measuring the distance of all points from the fitted surface in terms of the average error E_{avg} and the root-mean-square error $RMSE$.

We scan the box object containing orthogonal planes shown in Table 2. For each of the three planes a linear surface is fitted using RANSAC. This metric is measured as the angle formed between the three planes as shown in Table 3. As it is evident from the reported results the resulting planes are perpendicular (up to at least the fourth decimal point).

$\perp$	Box Plane Π^1_{Box}	Box Plane Π^2_{Box}	Box Plane Π^3_{Box}
Box Plane Π^1_{Box} (long)	-	$89.9997272431°$	$89.9995961527°$
Box Plane Π^2_{Box} (top)	$89.9997272431°$	-	$89.9978673427°$
Box Plane Π^3_{Box} (side)	$89.9995961527°$	$89.9978673427°$	-

Table 3: Orthogonality metric. This metric is measured by fitting planes using RANSAC to the three sides of the box shown in Table 2 and measuring the angle formed between them.

0.11.3 Accuracy

An object containing a ruler of length L_{orig} is scanned. The size of ruler is measured in the reconstructed model as L_{scan}. The accuracy is defined as the absolute difference between the two measurements,

$$E_{acc} = |L_{orig} - L_{scan}| \tag{18}$$

All calibration parameters are calculated in millimeters, hence the measuring units for accuracy is also millimeters.

We scan a planar object which contains a imprinted ruler on its surface. We measure several distances between points on the ruler on the reconstructed model and compute the average accuracy as shown in the Table 4.

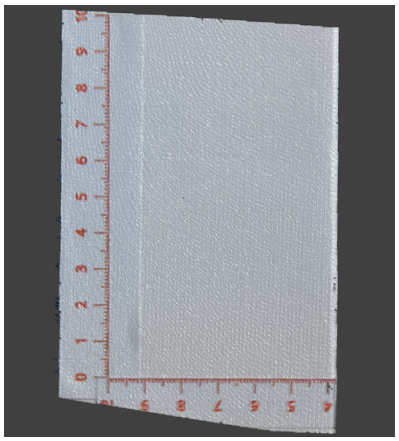
	E_{acc} = 0.0251 E_{acc} = 0.0827 E_{acc} = 0.0259 E_{acc} = 0.0166 E_{acc} = 0.0716 E_{acc} = 0.04335 E_{acc} = 0.0533 E_{acc} = 0.1223 E_{acc} = 0.003 E_{acc} = 0.0361 E_{acc} = 0.0413 E_{acc} = 0.1124 E_{acc} = 0.0397 E_{acc} = 0.0174 E_{acc} = 0.1091 E_{acc} = 0.0298 E_{acc} = 0.0933 E_{acc} = 0.0163 E_{acc} = 0.0153 E_{acc} = 0.0855
$E_{avg} = \sum_{i=0}^{20} \frac{E_{acc}^{i}}{20}$	0.0520025

Table 4: Accuracy metric. This metric is measured by taking multiple measurements on a reconstructed planar object with an imprinted ruler on its surface.

0.11.4 Sampling rate

The sampling rate is computed by selecting an area(patch) of known dimensions $(width, height)$ in the reconstructed model and measuring the number of points contained. We scanned a planar object containing an imprinted ruler on its surface. We manually crop multiple small patches from the reconstructed object and measure the corresponding point and face densities, an example of this procedure is shown in Figure 17. We report the individual point densities per squared centimeter (cm^2) and the average point density per

squared centimeter (cm^2) in Table 5. As it can be also be seen in Figure 18 the mean face density per cm^2 is 2835.6 faces and the mean point density per cm^2 is 1458.3. It should be noted that all faces are triangles for these experiments.

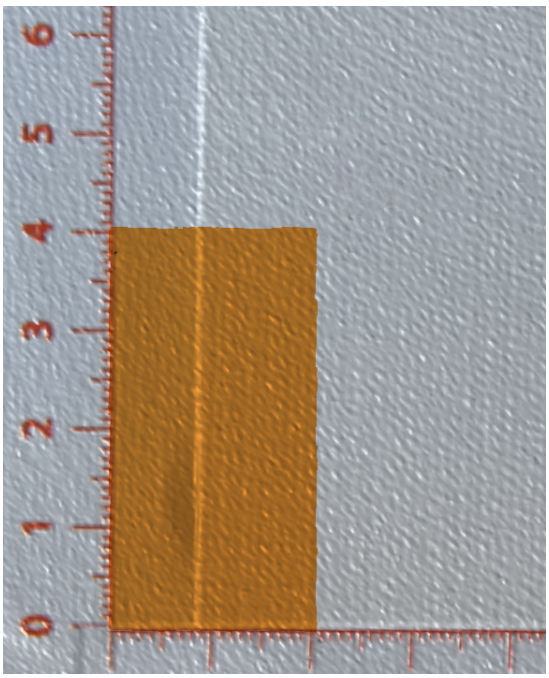


Figure 17: Example of a density measurement on the reconstructed object(rendered). Multiple areas are selected on a marked reconstructed object and the number of points and number of faces(triangles) are counted in order to compute the point and face density per squared centimetre.

0.11.5 Comparison to a High-End Commercial 3D Scanner

The proposed algorithms and implemented techniques were evaluated against a high-end commercial laser scanner. Figure 19a shows part of a mesh scanned with a high-end commercial 3D scanner, namely ZScanner 700CX. The laser scanner is hand held and it was used exactly as recommended in its manual with the highest possible resolution. During the the scanning process we scanned the part shown from **multiple** directions in order to capture as much as possible of the curved surface. The same part of the object is shown in Figure 19b but this time it was scanned using 3DUNDERWORLD-SLS v3.1. It should be noted that this is the results of a **single scan**.

We applied hole-filling on both meshes with the same settings (i.e. of size less than 50 edges) and both meshes are shown with Gouraud shading. The mesh produced by the commercial solution contained 107888 faces formed by 54870 points, and the mesh produced by our solution contained 55068 faces formed by 28014 points. A possible explanation for the significant difference in the number of points and faces is the fact that the laser scanner works in "sweeps" therefore, as the scanner moves more points are added to the reconstructed result. When using SLS the same occurs when

Width (cm)	**Height** (cm)	Point density	Face density
1	1	1588	2995
2	2	5968	11594
3	5	21129	41551
3	3	12864	25211
4	4	22354	44020
1	2	3117	5955
5	3	20959	41197
2	1	3060	5838
3	2	8700	16952
2	3	8740	17024
5	5	34759	68648
5	2	14295	27952
2	5	14308	28004
Mean point density per cm^2: 1458.3			
Mean face density per cm^2: 2835.6			

Table 5: Sampling rate. Multiple patches of difference sizes are taken from the reconstructed model shown in Table 4 and their corresponding point and face densities are measured.

combining multiple scans. Although the number of points and number of faces of the high-end 3D scanner surpass those of the 3DUNDERWORLD-SLS v3.1 it is evident that the 3DUNDERWORLD-SLS v3.1. outperforms the commercial solution in the level of detail captured and the amount of information captured in a single scan.

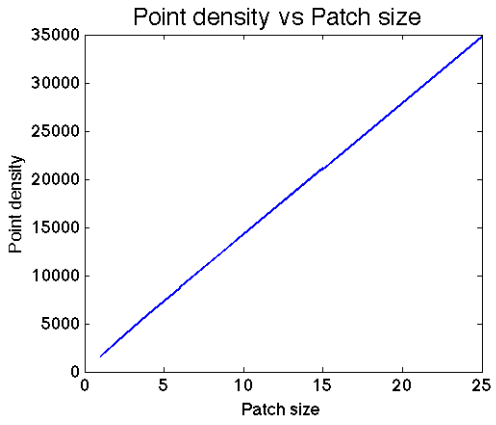
0.12 Conclusion

Although the theory behind the SLS systems is well documented and understood, there are still many issues one has to consider when developing or using SLS systems, which are currently lacking documentation. Many variants of SLS systems have already been proposed however, each one is tailored to a particular task. In this paper, we have presented all possible limitations, difficulties and solutions that one has to consider when involved in the design, development or use of SLS systems.

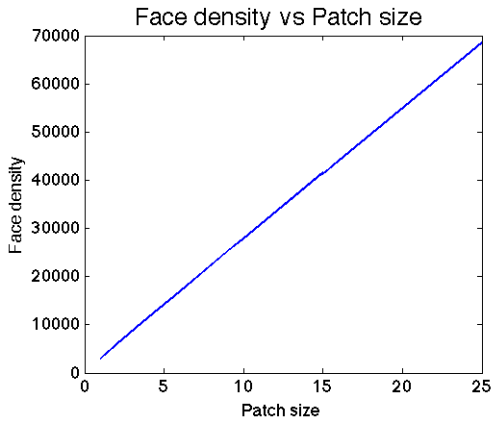
Furthermore, we have introduced the general-purpose, open-source 3DUNDERWORLD-SLS software and reported on the results of our extensive testing. Moreover, we have evaluated the proposed system on four different evaluation metrics and compared it to a high-end commercial 3D scanner. The produced results are of considerable high-fidelity and the system is currently being used as a documentation device in archaeological sites.

Acknowledgment

This work was supported by EC FP7 Marie Curie IRG-268256 - "Rapid Scanning and Automatic 3D Reconstruction of Underwater Sites" - 3DUNDERWORLD – `http:`



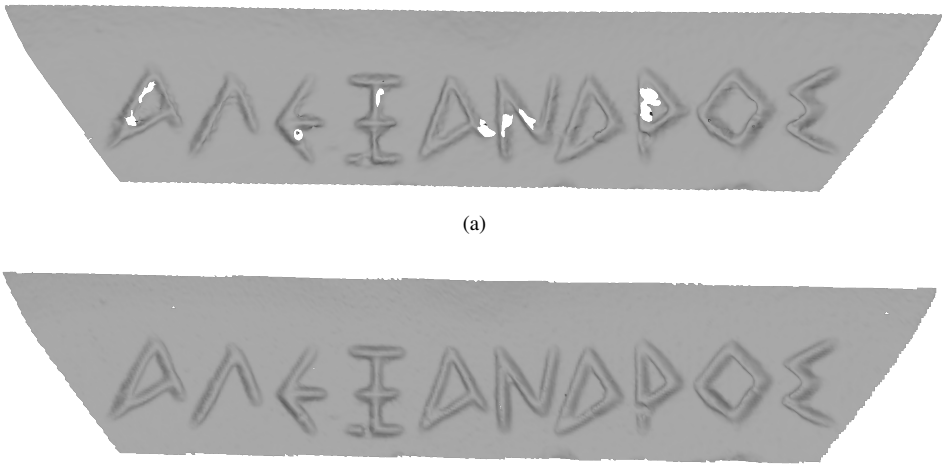
(a)



(b)

Figure 18: The point and face density for difference patch sizes. **Mean point density per cm^2: 1458.3**. **Mean face density per cm^2: 2835.6**

`//www.3dunderworld.org`. The software and other documentation can be downloaded from `http://www.3dunderworld.org/software`.



(a)

(b)

Figure 19: The same part of an object reconstructed with (a) ZScanner 700CX configured to the highest resolution and scanned from multiple directions and (b) the proposed system using a **single scan**. Both meshes were hole-filled using the same settings and are presented with the same Gouraud shading.

Bibliography

[1] Max Bajracharya, Jeremy Ma, Andrew Howard, and Larry Matthies. Real-time 3d stereo mapping in complex dynamic environments. In *International Conference on Robotics and Automation-Semantic Mapping, Perception, and Exploration (SPME) Workshop*, 2012.

[2] Jean-Yves Bouguet. Camera calibration toolbox for matlab. 2004.

[3] Fabio Bruno, Stefano Bruno, Giovanna De Sensi, Maria-Laura Luchi, Stefania Mancuso, and Maurizio Muzzupappa. From 3d reconstruction to virtual reality: A complete methodology for digital archaeological exhibition. *Journal of Cultural Heritage*, 11(1):42–49, 2010.

[4] Yan Cui, Sebastian Schuon, Derek Chan, Sebastian Thrun, and Christian Theobalt. 3d shape scanning with a time-of-flight camera. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1173–1180. IEEE, 2010.

[5] Yan Cui and Didier Stricker. 3d body scanning with one kinect. In *2nd International Conference on 3D Body Scanning Technologies*, pages 121–129, 2011.

[6] Dieter Fritsch, Mohammed Abdel-Wahab, Alessandro Cefalu, and Konrad Wenzel. Photogrammetric point cloud collection with multi-camera systems. In *Progress in Cultural Heritage Preservation*, pages 11–20. Springer, 2012.

[7] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 963–968. IEEE, 2011.

[8] Mohit Gupta, Amit Agrawal, Ashok Veeraraghavan, and Srinivasa G Narasimhan. Structured light 3d scanning in the presence of global illumination. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 713–720. IEEE, 2011.

[9] Richard I Hartley and Peter Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

[10] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[11] Brett R Jones, Rajinder Sodhi, Roy H Campbell, Guy Garnett, and Brian P Bailey. Build your world and play in it: Interacting with surface particles on complex objects. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*, pages 165–174. IEEE, 2010.

[12] K. Levenberg. A method for the solution of certain non-linear probelms in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[13] D. W. Marquardt. An algorithm for least-squares estimation of non-linear parameters. *Journal of the Society of Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[14] Charalambos Poullis. A framework for automatic modeling from point cloud data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(11):2563–2575, 2013.

[15] Charalambos Poullis, Andrew Gardner, and Paul Debevec. Photogrammetric modeling and image-based rendering for rapid virtual environment creation. Technical report, DTIC Document, 2004.

[16] Charalambos Poullis and Suya You. Automatic creation of massive virtual cities. In *Virtual Reality Conference, 2009. VR 2009. IEEE*, pages 199–202. IEEE, 2009.

[17] Charalambos Poullis and Suya You. Delineation and geometric modeling of road networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(2):165–181, 2010.

[18] Charalambos Poullis and Suya You. 3d reconstruction of urban areas. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2011 International Conference on*, pages 33–40. IEEE, 2011.

[19] Charalambos Poullis, Suya You, and Ulrich Neumann. A vision-based system for automatic detection and extraction of road networks. In *Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on*, pages 1–8. IEEE, 2008.

[20] MV Rohith, Gowri Somanath, Debra Hess Norris, Jennifer Jae Gutierrez, and Chandra Kambhamettu. A camera flash based projector system for true scale metric reconstruction. In *Applications of Computer Vision (WACV), 2009 Workshop on*, pages 1–8. IEEE, 2009.

[21] Sebastian Schuon, Christian Theobalt, James Davis, and Sebastian Thrun. High-quality scanning using time-of-flight depth superresolution. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–7. IEEE, 2008.

[22] Jan Smisek, Michal Jancosek, and Tomas Pajdla. 3d with kinect. In *Consumer Depth Cameras for Computer Vision*, pages 3–25. Springer, 2013.

[23] Gowri Somanath, Scott Cohen, Brian Price, and Chandra Kambhamettu. Stereo+ kinect for high resolution stereo correspondences. In *3DTV-Conference, 2013 International Conference on*, pages 9–16. IEEE, 2013.

[24] Jing Tong, Jin Zhou, Ligang Liu, Zhigeng Pan, and Hao Yan. Scanning 3d full human bodies using kinects. *Visualization and Computer Graphics, IEEE Transactions on*, 18(4):643–650, 2012.

[25] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.