

3D reconstruction of objects in the presence of
strong inter reflection or subsurface scattering
using structured light

Michael Hobbel
ICA-3350312
Utrecht University
supervised by prof. Remco Veltkamp

October 19, 2015

Abstract

In this thesis a new structured light method is proposed. It is a line shifting algorithm in combination with error correction based on scene continuity. The method initially gathers a large number of potential camera to projector correspondences. This is done by sweeping both a horizontal (row) and a vertical (column) scan line over the scene with a projector. This process is captured by a web cam on highly light sensitive settings. A depth consistency check is then performed to prune candidates that must have been lit indirectly because depths from potential matching rows match none of the depth from potential matching columns. The remaining candidates are scored on their combined scene continuity using dynamic programming with a cost function that rewards smoothness. To reduce the effect of subsurface scattering, adjacent candidates are grouped together and judged by their center. Additionally it is shown that multiple sweep lines can be run at once as the depth consistency check in combination with dynamic programming can solve most ambiguity. The proposed method was implemented in C++ and was tested on different scenes with indirect light against a structured light method specifically designed to handle indirect light. While the proposed method is a lot slower, it did outperformed the other method on resolution and accuracy and may therefore fill a niche of slow, but very cheap and highly accurate structured light methods.

Contents

1	Introduction	4
1.1	Importance of 3D reconstruction	4
1.2	Objectives	5
1.3	Contribution	6
2	Related work	7
2.1	Direct codification	7
2.2	Time multiplexing	8
2.3	Spatial neighborhood	9
2.4	Indirect light	9
3	Background knowledge	11
3.1	Camera Calibration	13
3.1.1	Pinhole camera	13
3.1.2	Extrinsic parameters	14
3.1.3	World to camera coordinate	14
3.1.4	Intrinsic parameters	14
3.1.5	Transforming camera to image coordinates	15
3.1.6	Transforming world to image coordinates	16
3.1.7	Calibration using a plane	16
3.1.8	Maximum likelihood	21
3.1.9	Distortion	21
3.2	Projector calibration	22
3.2.1	Plane orientation	23
3.3	Projector camera correspondence	24
3.3.1	Intersection method	25
3.3.2	Time multiplexing	25
3.3.3	More robust decoding	27

3.4	Intersections	29
3.4.1	Ray-plane intersection	29
3.4.2	Ray-ray intersection	30
3.4.3	Differences	33
3.5	Drawbacks binary decoding	33
3.5.1	Errors	34
3.6	Inter Reflectance	37
3.7	Subsurface scattering	40
3.8	Best of both worlds	43
4	Approach	46
4.1	Finding the candidates	47
4.2	Dynamic programming	48
4.3	Cost function	51
4.4	Reducing inter reflectance	55
4.5	Reducing subsurface scattering	58
4.6	Speeding up the process	61
5	Experimentation	62
5.1	Experimental design	62
5.2	Implementation	62
5.3	Experimental set up	63
5.3.1	Camera-Projector calibration	63
5.3.2	Capturing images	64
5.3.3	Camera intensity settings	65
5.3.4	Scan line decoding	65
5.3.5	Cost function	65
5.4	Experimental Results	65
5.4.1	Generic objects	66
5.4.2	Objects with strong inter reflectance	78
5.4.3	Objects with strong subsurface scattering	84
5.5	Evaluation	95
6	Conclusion and Future Work	98
6.1	Conclusions	98
6.2	Future work	99

Chapter 1

Introduction

1.1 Importance of 3D reconstruction

3D models are used nearly everywhere from the entertainment to the medical industry. However, due to a recent rise of consumer grade 3D printers, there exists a greater need for a wide variety of high quality printable 3D models. While it is possible to create such models manually using 3D modeling software, it is very time consuming and highly skilled work. Creating them using 3D scanning technology is therefore preferable. Although a multitude of such scanning techniques exist, each has its own benefits and drawbacks, with great variations in speed, quality and costs.

Active contact scanners use an arm to actively trace the surface of the object. This has the disadvantage that it can damage or modify the scene because it has to maintain contact, for that reason it is also relatively slow. Non-contact active scanners on the other hand emit light or some other type of radiation and reconstruct the model based on the object's interaction with the radiation. The two main types of non-contact active scanners are time-of-flight, which calculates the depth by timing how long it takes for the emitted light to bounce onto the object and back, and structured light, which calculates the depth based on the level of distortion of a known emitted light pattern.

Time-of-flight has the advantage that it can be used for (very) long distance scanning, but has a relatively low resolution and accuracy. Structured light on the other hand, works on a much smaller distance with a much higher resolution and accuracy. For the purpose of generating high quality 3D mod-

els, used by 3D printers, structured light therefore seems to be the obvious choice. Additionally, the equipment required to perform a structured light 3D scan may consist of only a single light source (typically a projector) and a single camera, which is inexpensive and therefore easily available to the average consumer.

1.2 Objectives

Structured light based methods calculate depth based on the distortion of known emitted light patterns. A general assumption used for this is that light that hits the surface of an object is scattered only diffusely. This means, that when a light ray hits an object, its light is scattered equally strongly in all directions away from the object's surface. That spot on the surface will therefore appear brighter to an observer. This is what we will call direct light. A lot of materials will approximately follow this assumption. In those cases, the distortion of the emitted patterns is observed accurately, and the resulting depth reconstruction of a scene is therefore also accurate.

However, there are also materials that do not follow this assumption. Metals, for example, will reflect a large portion of the direct light into a single direction based on the angle of the incoming light and the surface normal. This reflected light, can then hit a different part of the object's surface, where this effect can happen again. Because only a small portion of the light is scattered diffusely, this makes it very hard to observe the correct direct light. More importantly, if light is reflected towards an observer after the first contact with the object, then an indirectly lit part of the object will appear brighter, which may lead to an incorrect assumption about the distortion of an emitted light pattern. This effect is called inter reflection, and while it is strongest for highly reflective materials, such as metals, it can occur on a smaller scale for most materials. Inter reflection is a long range or global effect, as light can easily be reflected over larger distances inside a scene.

A different example of indirect lighting is subsurface scattering. Here, rather than light scattering on the surface of an object, it will travel into the object. After entering the object, the light will scatter in random directions and will eventually either leave the object at a different point, or be absorbed by the material. This may also lead to incorrect assumptions about the distortion

of an emitted light pattern. Unlike inter reflections, this is a however a very short range or local effect, as it is observed stronger near the point where the light hit the object directly.

Objectives: To solve the problem of indirect light for the relatively cheap but high quality method of structured light.

This thesis will only deal with the reconstruction of depth from a single point of view. Typically, multiple views are combined to create a model. The process of stitching multiple views together or any post processing past the depth reconstruction is not within the scope of this thesis.

1.3 Contribution

The contributions of this thesis are a complete framework that handles difficult scenes with indirect light in the form of inter reflections and subsurface scattering well. This is done by first collecting a large number of potential camera to projector correspondence candidates by running a line sweep algorithm. This is done using a camera set up to be highly light sensitive settings. The best candidates are then picked using dynamic programming in combination with a cost function that rewards scene continuity. Additionally, candidates are pruned based on the idea that any depth that is the result of a correspondence between a camera pixel and a projector row, must always be supported by a projector column resulting in the same depth, in cases where it is direct light.

Chapter 2

Related work

Structured light works by finding a correspondence between each camera pixel and the projector which allows the calculation of an intersection point: a 3D point in the scene. In the area of structured light several different methods have been proposed. Salvi et al.[1], wrote an excellent overview of structured light techniques. Most techniques fall into one out of three different structured light families that each will get a short explanation.

2.1 Direct codification

Direct codification techniques attempt to pack the identification code of a projector column number into a single pixel. If the pixel, and therefore its identifying code is observed somewhere, then a correspondence is found. This is most commonly done using different shades of gray or colors. Carrihill and Hummel [3], created a wedge of increasing gray values. Each projector column gets its own shade of gray and whenever that shade is observed by the camera, the corresponding projector column is immediately known. Because the albedo and the effects of ambient light may vary within the scene, an image unlit by the projector must be subtracted first. However, while this method is very fast, because the differences between different colors, and therefore corresponding columns, are very small, it is also highly inaccurate and prone to noise.

2.2 Time multiplexing

Time multiplexing techniques make use of the fact that data from multiple images, taken at different times, can be combined if both the scene and the scanning equipment remain static. Of the structured light techniques this results in 3D models of the highest resolution and accuracy. Because of the relatively high number of images, it however has a high acquisition time and scenes have to remain still for the entire acquisition period.

The simplest time multiplexing techniques are binary code based. The projected images consist solely of black and white pixels. By projecting a unique code of black and white pixels for each projector column using multiple images, a correspondence between each observing (camera) pixel and each projector column can be established. The first and simplest binary code based algorithm was made by Posdamer [5]. Inokuchi et al. [6] modified this coding scheme to a Gray coding scheme (named after Frank Gray, not the color) to make the Hamming distance between consecutive column numbers one. This made the scheme less susceptible to noise. Many different coding schemes have been proposed since then. Each with its own advantages and disadvantages. Several binary coding techniques will be explained in more detail in the background knowledge chapter of this thesis.

To greatly reduce the number of required images, some time multiplexing techniques use different shades of gray or even color, such as the n-ary technique proposed by Caspi et al.[7]. The extra colors add additional information to each projected image which greatly reduces the required number of acquired images. However, because different points of the scene may react differently to different projected colors, using additional colors may also reduce the accuracy of the method.

Finally, there are time multiplexing techniques that simply shift a simple pattern slowly over the entire scan range. The advantage of using shifted patterns is that they locally have a very high precision at even sub-pixel levels. However the downside is that to shift the patterns over an entire scan range, a very high number of images is required. By periodically repeating the pattern, while solving the ambiguity created by the periodic repetition, the best of both worlds can be achieved. Guhring [4] for example, shifted a periodically repeated single white projector column and solved the arising

ambiguity with a small number of binary coding based images. This results in a line sweep algorithm that was inspired by laser line scanning techniques.

Similarly, Bergman[8] used gray intensities of sinusoid patterns and also solved the ambiguity using binary coding based images.

2.3 Spatial neighborhood

Unlike time multiplexing techniques, spatial neighborhood usually use only a single image. Identifying information for a point in the image is based on the information of neighboring pixels. Because only a single image is used, dynamically moving scenes can be scanned. However deformations and occlusions of patterns because of depth differences in a scene, will often cause incorrect reconstructions. Additionally, the decoding process is more complex and takes longer compared to other techniques. The acquisition time is a lot lower because only a single image is used. However, because there is only little data space to work with, the resolution and accuracy are relatively low compared to time multiplexing.

Microsoft Kinect for instance uses an infrared laser which is then split into unique speckle patterns. Research by Khoshelham and Elberink [9] suggests that while the Kinect is very fast, the point density and accuracy is a lot lower than some of the other methods.

2.4 Indirect light

Most research is focused on further improving the accuracy and resolution of techniques or further speeding them up. Despite that, nearly all structured light techniques suffer under indirect light.

Nayar et al.[10] showed how high frequency patterns can be used to separate the indirect light and the direct light. Attempts to apply this to structured light however run into problems when the direct component is too weak or noisy because of strong indirect components. Chen et al.[11] used polarized light to separate the direct light from the indirect light using phase shifting. They concluded that while the polarization is effective against the indirect light of subsurface scattering, it also reduces the direct light, which makes reconstruction at near grazing angles not possible. Gupta et al.[2] used four different binary codification techniques with different strengths and weak-

nesses to different types of indirect light. If two separate binary codes agree for a single pixel, then their answer is assumed to be the correct answer to the situation. For our experiments, we will compare our proposed method to the combined code ensemble and individual binary patterns by Gupta et al.[2].

Chapter 3

Background knowledge

In this chapter the theory required to understand the process of structured light is covered. Structured light works by knowing the location and orientation of a projector and a camera. By projecting a known pattern onto a scene, a correspondence between projector and camera pixels can be established. Once this is done, the depth of a point on the scene can be recovered by equating the mathematical representations of the camera and projector pixels. Because of the triangular shape between the projector, camera and the intersection of their pixel rays, this technique is known as triangulation.

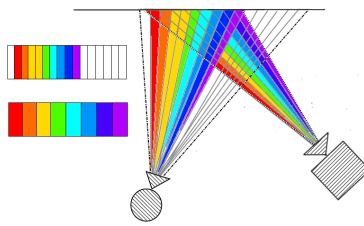


Figure 3.1: A sample structured light setup in 2D. The projector on the right projects the image on the bottom left onto a wall. The camera in the center captures this wall in the image on the top left. Each color corresponds to a projector column. By observing a (reflected) color in a camera pixel, it is established that the camera pixel intersects with the projector column corresponding to that color, which makes calculating the depth of the intersection possible.

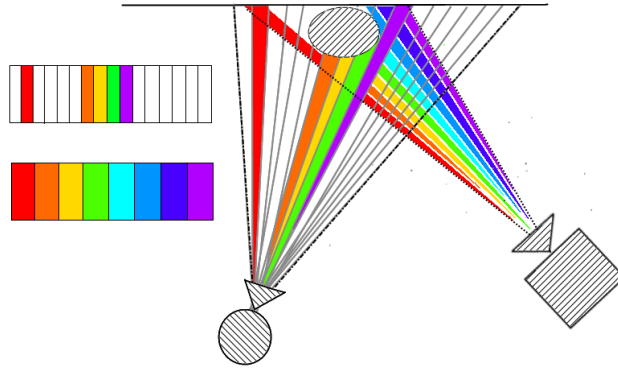


Figure 3.2: Another example of the same structured light setup. The image on the bottom left is projected onto the scene. The observed decoding distorted by the object is shown as observed by the camera in the top left.

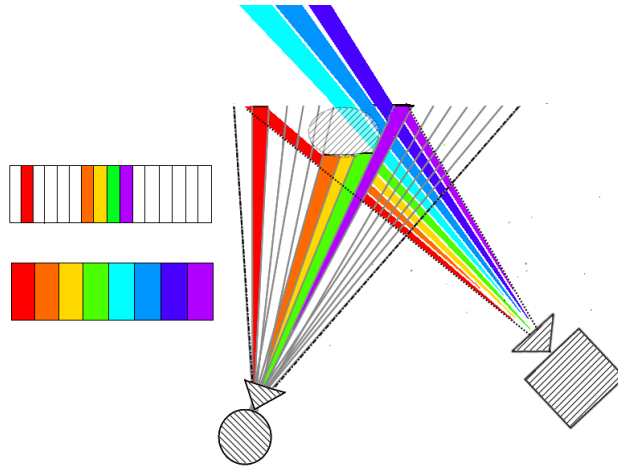


Figure 3.3: The observed decoding from the previous image is used to reconstruct the depth. The original object is shown semi transparent on top of the reconstruction. The reconstructed depth corresponds to the intersection of the corresponding camera and projector pixel ray and the resolution of the reconstruction therefore depends on the resolution of both the projector and camera.

3.1 Camera Calibration

The first step of the triangulation is extracting the required parameters of both the camera the projector, needed to create the mathematical rays corresponding to their pixels. This is known as calibrating the system. Multiple methods of calibration have been proposed. The relatively easy to implement method by Zhang[12] will be discussed here, this allows a user to calibrate the 3D system using a 2D pattern, commonly a printed checkerboard.

3.1.1 Pinhole camera

Pinhole cameras are by far the most popular cameras, mostly because they are relatively simple and cheap. However, especially cheaper models do have the drawback that they often suffer from distortion. This distortion affects the image position relative to the mathematical expected true image position of objects captured by the camera. The effect is usually circularly symmetric and gets stronger further away from the center of the image. This type of distortion is therefore called radial distortion. Zhang's method calibrates a camera using only a known pattern (commonly a chessboard) to find the intrinsic parameters, the extrinsic parameters and the first two orders of radial distortion. The objective of the calibration process is to find the parameters that affect how the camera projects the 3D world onto a 2D image plane. Once it is known how the 2D image plane is affected by these parameters, then the inverse can be used to calculate positions in the 3D world. The camera model consists of the intrinsic camera parameters K and the extrinsic parameters R (rotation) and T (translation) which combined form the homography (H) which can be used to transform a point represented by 3D world coordinates to an image coordinate.

Three different coordinate systems are used to describe the same point. The image coordinate system consists of two coordinates and relates to pixel locations in the image. The camera coordinate system consists of three coordinates and describes locations relative to the camera location. And finally, the world coordinate system, which also consists of three coordinates and describes locations relative to the world origin. In the next sections it will be shown how to transform from one coordinate system to another and how this helps extract the camera parameters.

3.1.2 Extrinsic parameters

The extrinsic parameters consist of the 3x3 rotation matrix R denoting the camera's orientation in 3D:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = [r_1 \quad r_2 \quad r_3]$$

where r_i is the i th column of R . The three columns of R form an orthonormal basis in R^3 , which makes them unit length and also means the dot product between each 2 of them is 0.

The translation matrix 1x3 T denotes the camera's translation in 3D:

$$T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

The extrinsic parameters are often concatenated into a single 4x3 matrix:

$$[R \ T] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} = [r_1 \quad r_2 \quad r_3 \quad t]$$

The extrinsic parameters affect how the camera transforms points from coordinates centered around the world's origin, to coordinates centered around the camera.

3.1.3 World to camera coordinate

Transforming from world coordinates to camera coordinates:

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} (x * r_{11} + y * r_{12} + z * r_{13}) + t_x \\ (x * r_{21} + y * r_{22} + z * r_{23}) + t_y \\ (x * r_{31} + y * r_{32} + z * r_{33}) + t_z \end{bmatrix}$$

3.1.4 Intrinsic parameters

Next, the intrinsic parameters, K , are how a camera transforms points from a coordinate system centered around the camera, to a coordinate centered

around the image. $K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$

here f_x and f_y are the camera's focal length and (c_x, c_y) the optical center of the camera in pixels. This matrix is also known as the camera matrix. Once it has been computed for a camera it can be reused regardless of camera orientation or translation. If the resolution of the camera is either scaled up or down, then the focal length as well as the optical center are scaled along by the same factor.

3.1.5 Transforming camera to image coordinates

Transforming from camera coordinates to image coordinates is done by multiplying the camera matrix K with camera based coordinates:

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} f_x * x_c + c_x * z_c \\ f_y * y_c + c_y * z_c \\ z_c \end{bmatrix}$$

Because image coordinates only have two meaningful coordinates (x, y) , the transformation from a point in camera coordinates to image coordinates results in a ray through the point, rather than a point. Homogeneous coordinates are therefore used to allow this transformation. The other two coordinates are divided by the 3rd coordinate z_c . If $z_c = 0$ then the point is at infinity.

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = z_c \begin{bmatrix} f_x * x_c / z_c + c_x \\ f_y * y_c / z_c + c_y \\ 1 \end{bmatrix} \sim \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

The \sim used here denotes that the two sides are now equal up to a non zero scalar value.

Transforming from image coordinates to a camera coordinates now becomes:

$$K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

3.1.6 Transforming world to image coordinates

Combining the transformation from world to camera coordinates and from camera coordinates to image coordinates gives:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}$$

Or in short:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim K[R \ T] \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}$$

H is defined as the homography, $H = K[R \ T]$ which projects world coordinates to image coordinates.

3.1.7 Calibration using a plane

Now that the reference frames have been established, a calibration plane can be used to find the values of H that map the observed points in image coordinates in an image to known points in world coordinates.

To do this we first take pictures of a calibration plane with some well distinguishable points with known measurable distances between them. A common choice for this is the corner crossings of a checkerboard pattern as these are easy to detect automatically. The center of the world coordinates is chosen to coincide with the first image point. Since all other points lie on the same calibration plane, the entire plane is chosen to have a depth of 0 which sets all points (Z_W) on it to 0. The Z_X and Z_Y of a point is set to the real world distance to the chosen first point, which is measured. H is now readily computed for the given image; the location of points is known in both image coordinates (detected) and in world coordinates (measured). H only maps one to the other. Because of noise in the data however, computing H exactly is often not possible, but there are multiple ways to estimate H .

The real calibration problem however, is extracting K, R, T from H . To do this, the fact that all points have $Z_W = 0$ is exploited.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim K \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ 0 \\ 1 \end{bmatrix}$$

Because the column r_3 is only multiplied by 0s it has no effect on the results and we can therefore omit it:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ 1 \end{bmatrix}$$

This also reduces H to a 3x3 matrix

$$K[r_1 r_2 t] = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} = [h_1 \quad h_2 \quad h_3]$$

where

$$h_i = \begin{bmatrix} H_{1i} \\ H_{2i} \\ H_{3i} \end{bmatrix}$$

Next, two constraints are added to R , because it needs to be an orthonormal basis in R^3 .

The first of the two constraints is that as r_1 and r_2 are orthogonal their inner product is 0.

$$r_1 r_2 = 0$$

rewriting both in terms of K and h

$$h_1 = K r_1$$

$$h_2 = K r_2$$

$$r_1 = K^{-1}h_1$$

$$r_2 = K^{-1}h_2$$

Transposing either one of them to get the inner product.

$$h_1^T K^{-T} K^{-1} h_2 = 0 \tag{3.1}$$

The second is that r_1 and r_2 are both unit vectors and therefore:

$$\|r_1\| = \|r_2\| = 1$$

per definition of vector length:

$$\sqrt{r_1 \cdot r_1} = \sqrt{r_2 \cdot r_2}$$

$$r_1 \cdot r_1 = r_2 \cdot r_2$$

using the notation of r_1, r_2 in terms of h and K

$$h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2$$

$$h_1^T K^{-T} K^{-1} h_1 - h_2^T K^{-T} K^{-1} h_2 = 0 \tag{3.2}$$

Both equation 3.1 and 3.2 need to be satisfied simultaneously when we compute the values of K, R, T .

Simplifying these constraints further we use $B = K^{-T} K^{-1}$

$$B = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$$

B is symmetric in the diagonal and positive definite. Once B is computed, K can be computed using Cholesky factorization.

Additionally, as $B_{12} = B_{21}, B_{13} = B_{31}, B_{23} = B_{32}$ we can reduce B to

$$b = \begin{bmatrix} B_{11} \\ B_{12} \\ B_{13} \\ B_{22} \\ B_{23} \\ B_{33} \end{bmatrix}$$

Because of the symmetry in B , B and therefore b , have only 6 degrees of freedom. To take advantage of these reduced degrees of freedom we rewrite the constraints to a better suited form

$$h_i^T B h_j = v_{ij}^T b$$

for

$$v_{ij} = \begin{bmatrix} h_{1i}h_{1j} \\ h_{1i}h_{2j} + h_{2i}h_{1j} \\ h_{1i}h_{3j} + h_{3i}h_{1j} \\ h_{2i}h_{2j} \\ h_{2i}h_{3j} + h_{3i}h_{2j} \\ h_{3i}h_{3j} \end{bmatrix}$$

To prevent the reader the hassle of working through several tedious matrix multiplications to confirm this equation, the proof is given below:

$$h_i^T \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} \begin{bmatrix} h_{1j} \\ h_{2j} \\ h_{3j} \end{bmatrix} = v_{ij}^T \begin{bmatrix} B_{11} \\ B_{12} \\ B_{13} \\ B_{22} \\ B_{23} \\ B_{33} \end{bmatrix}$$

$$h_i^T \begin{bmatrix} B_{11}h_{1j} + B_{12}h_{2j} + B_{13}h_{3j} \\ B_{21}h_{1j} + B_{22}h_{2j} + B_{23}h_{3j} \\ B_{31}h_{1j} + B_{32}h_{2j} + B_{33}h_{3j} \end{bmatrix} = \begin{aligned} & B_{11}h_{1i}h_{1j} + \\ & B_{12}(h_{1i}h_{2j} + h_{2i}h_{1j}) + \\ & B_{13}(h_{1i}h_{3j} + h_{3i}h_{1j}) + \\ & B_{22}(h_{2i}h_{2j}) + \\ & B_{23}(h_{2i}h_{3j} + h_{3i}h_{2j}) + \\ & B_{33}(h_{3i}h_{3j}) \end{aligned}$$

$$\begin{aligned} & B_{11}h_{1j}h_{1i} + B_{12}h_{2j}h_{1i} + B_{13}h_{3j}h_{1i} + \\ & B_{21}h_{1j}h_{2i} + B_{22}h_{2j}h_{2i} + B_{23}h_{3j}h_{2i} + \\ & B_{31}h_{1j}h_{3i} + B_{32}h_{2j}h_{3i} + B_{33}h_{3j}h_{3i} \end{aligned} = \begin{aligned} & B_{11}h_{1i}h_{1j} + \\ & B_{12}(h_{1i}h_{2j} + h_{2i}h_{1j}) + \\ & B_{13}(h_{1i}h_{3j} + h_{3i}h_{1j}) + \\ & B_{22}(h_{2i}h_{2j}) + \\ & B_{23}(h_{2i}h_{3j} + h_{3i}h_{2j}) + \\ & B_{33}(h_{3i}h_{3j}) \end{aligned}$$

Finally we see if we rearrange the terms, that if we take into account that on the right side the symmetrical terms appears once but are twice multiplied, that both sides are equal.

This means that the two constraints can now be expressed as:

$$Vb = 0$$

for

$$V = \begin{bmatrix} v_{12}^T \\ v_{11}^T - v_{22}^T \end{bmatrix}$$

Each image used to calibrated the camera gives two equations subject to these constraints. These equations are stacked on top of each other in V . V thus becomes a $2n \times 6$ matrix, where n is the number of images used.

Because b has six degrees of freedom at least three images are required to get six linear equations required to solve the system. If more equations are present, because images are usually corrupted by noise there is no exact solution to the system of equations. Least squares minimization can be used to solve the equations.

After solving the system for b , K is computed from b using Cholesky factorization. From K each of the other parameters can then be calculated for each of the images:

$$r_1 = K^{-1}h_1$$

$$r_2 = K^{-1}h_2$$

$$r_3 = r_1 \times r_2$$

$$t = K^{-1}h_3$$

Because of noise in the data and using least squares optimization it is unlikely that R is a proper rotation matrix (orthogonal basis in R^3). Furthermore minimizing these algebraic distances is not physically meaningful. It does however provide a good initial starting guess that can be refined further.

3.1.8 Maximum likelihood

For each of the n calibration images found before, m points are observed in image coordinates. Each of these m points needs to correspond to a known world coordinate on the $Z_W = 0$ plane. The function $m(K, R_i, t_i, M_j)$ maps a point M_j in world coordinates to a point in image coordinates, according to the parameters K , R_i and t_i . Here K is the same for all images, but the values of R_i and t_i are specific for each of the n images; after all in each case the calibration plane has its own orientation and translation.

By taking the squared difference between the m known points in image coordinates and the mapped points from the known world coordinates to image coordinates according to the estimated parameters, the best parameters can be estimated. The closer their difference is to 0, the better the parameters fit. This function is known as the maximum likelihood function as it maximizes the likelihood the parameter values are correct given the known evidence.

$$\sum_{i=1}^n \sum_{j=1}^m |m_{ij} - m(K, R_i, t_i, M_j)|^2$$

Here m_{ij} is the j th point in the i th image in image coordinates, $m(K, R_i, t_i, M_j)$ is the projection of M_j , the j th point in world coordinates, to image coordinates using K , and R_i and t_i specific of the i th image. Here R_i can be parametrized by only 3 parameters using Rodrigues rotation formula, nevertheless as K , R_i and t_i are unknown this is a non-linear optimization problem that can be solved using the Levenberg-Marquardt algorithm. This however requires an initial guess, for which the closed form solution in the previous section is used.

3.1.9 Distortion

Because of the image distortion of pinhole cameras, the parameter estimation needs to be further refined for accurate calibration results.

Zhang only takes into account the first 2 radial components as the benefits of using more components is negligible while it can introduce numerical instability.

The distorted image coordinates are calculated by adding the radial components to the undistorted image coordinates:

$$\begin{aligned}\tilde{u} &= u + (u - u_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \\ \tilde{v} &= v + (v - v_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]\end{aligned}$$

Here \tilde{u} , \tilde{v} are the distorted image coordinates, u , v are the undistorted ideal coordinates following the pinhole model based on the first 2 order radial components k_1 and k_2 , and u_0 , v_0 are the center of the image. The farther a point lies from the optical center of the camera, or the farther the corresponding pixel lies from the center of the image, the more it is affected by both the first order and second order radial distortion.

rewriting the distortion model to a matrix notation yields:

$$\begin{bmatrix} \tilde{u} - u \\ \tilde{v} - v \end{bmatrix} = \begin{bmatrix} (u - u_0)(x^2 + y^2) & (u - u_0)(x^2 + y^2)^2 \\ (v - v_0)(x^2 + y^2) & (v - v_0)(x^2 + y^2)^2 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$$

$$d = Dk$$

After first using the maximum likelihood function to estimate all other parameters other than the distortion, we can use the pinhole model to calculate u , v , u_0 , v_0 , x , y while the distorted \tilde{u} , \tilde{v} can be observed. This thus gives two equations for every point in every image, making D a $2mn \times 2$ matrix. Solving this system of equations can be done using least squares optimization.

Once k_1 , k_2 have been estimated the maximum likelihood function is expanded to include the distortion and can then be solved using the Levenberg-Marquardt algorithm using the parameters found by the previous maximum likelihood function and the estimates of k_1 and k_2 as an initial guess:

$$\sum_{i=1}^n \sum_{j=1}^m |m_{ij} - m(K, k_1, k_2 R_i, t_i, M_j)|^2$$

3.2 Projector calibration

The projector can be thought of as an inverse camera; whereas a camera projects the 3D scene onto a 2D plane, the projector projects a 2D plane onto a 3D scene. The 3D scene can be controlled by projecting known image points onto a plane of a known orientation and then capturing these

points using a calibrated camera. These camera image points can then be converted to world coordinates using the camera calibration. Because the projector image points are known corresponding to these world coordinates, the homography H_{proj} can be estimated, just like was done with the camera, followed by the computation of the intrinsic parameters K_{proj} , extrinsic parameters $[R_{proj}T_{proj}]$ in the same way it was done before with the camera.

3.2.1 Plane orientation

A good way to find the orientation of the calibration plane is to print a known image onto a plane used for calibration, such as a checkerboard. By first calibrating the camera using this image the extrinsic parameters $R_{cam}T_{cam}$ are found.

The calibration plane is then expressed by converting to the general formulation:

$$ax + by + cz + d = 0$$

a, b, c and d are known parameters for the plane, (x, y, z) is a point on the plane if the equation holds.

By projecting an additional calibration image onto the plane using the projector and then capture it using the calibrated camera, the world coordinates corresponding to these projector image points can be retrieved up to a non zero unknown scalar value by using the camera homography. Recall:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim H_{cam} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} H_{cam}^{-1} \sim \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} H_{cam}^{-1} = s \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}$$

s is defined the scalar value required to scale the unscaled world coordinates to the correct ones. We know that the ray corresponding to these world coordinates intersects the calibration plane. Computing s and therefore the exact coordinates is therefore only a matter of computing the intersection point between the ray and the plane. Thus, the ray is filled into the plane equation:

$$s(aX_W + bY_W + cZ_W) + d = 0$$

As s is the only unknown in the equation, it can be computed readily. Scaling the unscaled coordinates by s gives the correct world coordinates. This is done for each of the m points in each of the n images used for calibration. The homography H_{proj} , intrinsic and extrinsic parameters and distortion parameters for the projector are then calibrated in the exact same way we did for the camera.

3.3 Projector camera correspondence

With the camera projector system calibrated the depth of any combination between camera pixels a projector pixels can be calculated. The difficulty is finding the correct correspondence between each camera and projector pixel. In the images 3.1, 3.2 and 3.3 a color encoding was shown. While such color encodings allow correspondence decoding between many projector and camera pixels within a single image, the results are highly affected by the color and material of the scanned scene. While very fast, this technique therefore only has a limited precision. To create something much more robust to many material problems, a single color (white) could be used. Nevertheless, regardless of the how the column numbers are identified and encoded, the examples shown in the images apply ever the same. With the use of only a single color, the very simplest way to identify the camera projector correspondence is to only light a single projector pixel and then check which camera pixel is brightened.

The obvious problem with this is that this would require as many pictures as there are pixels on the projector to get a full scan (width times height). Assuming a not unreasonable 1024x768 projector resolution, with a camera capturing 4 pictures per second, this would take over 2 days.

Another difficulty arises when camera pixels brighten up for multiple different projector pixels. In addition, it is also possible that a much larger

than feasible number of camera pixels light up for a single projector pixel, this is likely an indication that there is a lot of indirect lighting affecting the scene, which will give incorrect decoding results.

3.3.1 Intersection method

The first thing that can significantly speed up the scanning process is noticing how the depth is constructed after the correspondence is found (more details on intersection depth construction in the next section). This is done by triangulation: for each camera pixel a mathematical ray-representation is known, and a second equation coming from the projector is required to get an intersection point between the two.

While using a single projector pixel works well for depth calculation, as it gives a ray-ray intersection, theoretically, a plane-representation, where the projector ray lies entirely on the plane, would give the same intersection point. In the next section some small practical objections are brought up to this, however the depth difference is minor and will considerably speed up the process.

This means that the projector can project an entire plane, rather than a single pixel ray. The easiest way to project a plane with the projector rather than a single pixel, is to light up a single row, or column in the projected image.

This also means that using rows (or columns) will reduce the number of pictures required for a full scan to just the projector height (or width). Assuming the same 1024x768 projector resolution, with a camera capturing 4 pictures per second, the time required is now reduced to a little over 4 minutes using columns, and a little over 3 minutes for rows.

As mathematically there is no real difference between using either rows or columns, when referring to either one of them later in this thesis, the other is also automatically inferred.

3.3.2 Time multiplexing

As neither the camera nor the projector moves during the capturing process, each pixel for both the projector and camera always corresponds to the same location in the scene. Therefore, multiple images can be combined to decode the correspondence between an individual camera pixel and the projector. The idea of combining images taken at different times, rather than encoding

them into a single image is known as time multiplexing. This has the advantage over single image techniques that it can have a higher resolution and accuracy as there is more data space to work with. This however does have the obvious disadvantages that the scene cannot move, that it takes more acquisition time and a lot more data space.

One of the more time efficient time multiplexing techniques is the idea of storing a binary encoding for each pixel using multiple images. The encoding of a projector column number can be thought of as a binary string, where each of the images corresponds to a single bit. Because each pixel corresponds to the same spot in 3D, multiple image bits can be combined to form a binary column number. If a pixel is "turned on" in an image, then the bit is set to 1. If not the bit is set to 0. By combining all the image bits of a single camera pixel, the corresponding projector column number can be decoded as a binary number

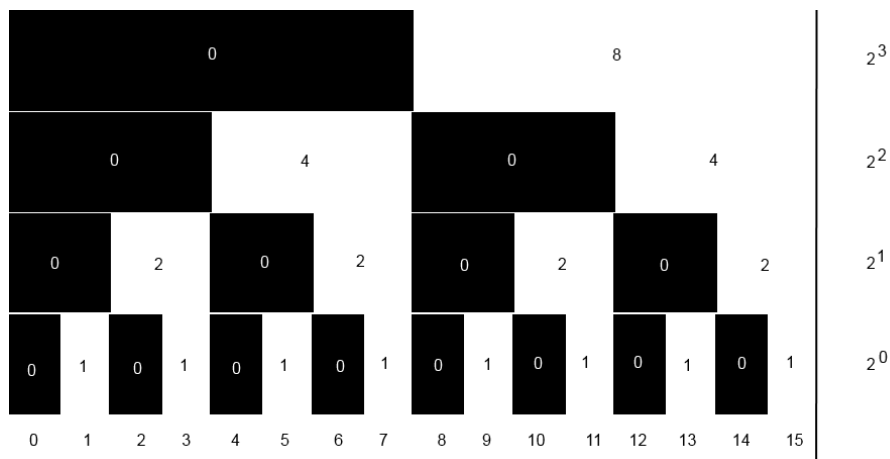


Figure 3.4: first 4 images of a binary projection method. 2^4 different columns can be distinguished. By reading the image from top to bottom the binary encoding of a column can be determined. Whenever the column is white for the image, its corresponding bit is set to 1, if it's black it is set to 0.

Or in a non-image form:

image number																
1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
column number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Because the method is binary each image helps distinguish a factor 2 more columns. The maximum number of distinguishable columns is therefore 2^n , where n is the number of captured images. Using the same projector using 1024x768 resolution it would require only $\lceil \log_2 1024 \rceil = 10$ pictures for columns or $\lceil \log_2 768 \rceil = 10$ pictures for rows.

In order to determine whether a pixel is lit or unlit in each image, an additional two images are taken: one where none of the projector’s pixels are lit, and another where all of the projector’s pixels are lit. By taking the difference between the fully- and unlit image, an initial mask can be created to take out any background pixels that will not receive light as well as pixels that get full saturation even without the added light of the projector. For all the other images the unlit image is also subtracted. If a pixel’s remaining intensity is still over a threshold value, then the projector is considered lit. Otherwise it is considered unlit.

3.3.3 More robust decoding

To improve the reliability of decoding results, each projector image may also be inversed (black where white, white where black), in order to make a clearer distinction between the lit and unlit parts of the pattern. In the rest of this thesis, the distinction will be made between ”regular” patterns and the ”inversed” pattern, which as a pair are used for a single encoding bit.

Additionally, while capturing either just the columns or just the rows would be sufficient, capturing both will improve precision and gives the option of rejecting pixels when the resulting depths between the two are too far apart and therefore likely faulty.

Algorithm 1 Decode

```
1: procedure DECODINGINITIALIZATION(unlitImage, litImage, partial-
   LitImages)
2:   mask  $\leftarrow$  litImage - unlitImage > backgroundThreshold
3:   for each image in partialLitImages do
4:     differenceImages  $\leftarrow$  image - unlitImage
5:   decode(differenceImages, mask)
6:
7: procedure DECODE(cameraImages, mask)
8:   i  $\leftarrow$  number(cameraImage) - 1
9:   for each image in cameraImages do
10:    columnValue  $\leftarrow 2^i$ 
11:    for each pixel in image and not in mask do
12:      if pixel.intensity  $\geq$  threshold then
13:        decoding[p]  $\leftarrow$  decoding[p] + columnValue
   i  $\leftarrow$  i - 1
```

Algorithm 2 DecodeMoreRobust

```
1: procedure DECODINGINITIALIZATION(unlitImage, litImage, imgsNor-
   mal, imgsInversed)
2:   mask  $\leftarrow$  litImage - unlitImage > backgroundThreshold
3:   for each img1 in imgsNormal and img2 in imgsInversed do
4:     differenceImages  $\leftarrow$  image1 - image2
5:   decode(differenceImages, mask)
```

3.4 Intersections

With the system calibrated and a way to find correspondences between the camera and the projector in the scene, this section will show how to calculate the depth using that correspondence. Depending on the decoding process used, different methods apply.

3.4.1 Ray-plane intersection

As noted in a previous section, using columns or rows mathematically corresponds to projecting a plane of light onto the scene. Observing a specific column (or row) at a specific pixel, therefore corresponds to the intersection between a plane and a ray. The plane will be expressed in intrinsic form

$$ax + by + cz + d = 0$$

for which we will define the plane's normal $n = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ giving

$$n \cdot p + d = 0$$

where p is a point in world coordinates.
and the ray in parametric form:

$$R(t) = q + tv$$

where q is a point on the ray, the camera's origin, v a vector denoting the direction of the line, and t the distance moved along the ray. As it is of the parametric form any and every point on this ray is described by letting t move from 0 to infinity.

As any intersection point lies on both the ray and plane and must therefore satisfy both equations, the ray equation can be filled into the plane equation:

$$n \cdot (q + tv) + d = 0$$

As every parameter but t is known, solving for t will find the intersection. Rewriting for t gives us:

$$n \cdot q + n \cdot (tv) + d = 0$$

$$\begin{aligned}
n \cdot q + t(n \cdot v) + d &= 0 \\
t(n \cdot v) &= -(n \cdot q + d) \\
t &= -\frac{n \cdot q + d}{n \cdot v}
\end{aligned}$$

Feeding t back into the ray equation gives the world coordinate where the intersection occurred; a point on the scene. While this method omits some degenerate cases, in practice these degenerate cases cannot occur with the setup.

If not just rows or just columns were used, but both of them, then both intersection points should ideally be the same. However in practice there is always going to be some difference due to imperfect calibration, decoding errors, or a simple limit in resolution. If the two intersection points however lie farther than a distance threshold apart, then both points are rejected and the pixel will have missing depth information. If the two points do lie within the threshold of each other, then the average of the two is used for the depth.

3.4.2 Ray-ray intersection

If both rows and columns are used, then a specific projector pixel can be pinpointed, which means that the point on the scene is the intersection of the camera pixel ray and the projector pixel ray. It is however highly likely, for the same reasons given for the plane-ray intersection method, that the two rays do not intersect exactly. In this case the closest point between the two rays is used.

Given both ray equations in parametric form:

$$R_1(t) = q + tv$$

$$R_2(s) = p + su$$

$R_1(t)$, returns a point in world coordinates t times the orientation of the ray, v , away from its origin q , which in this case is the camera's origin. Likewise $R_2(s)$, does the same thing for a ray coming from the projector's origin.

Since we are interested in the smallest distance between the rays (preferably the intersection), we will define the distance as the length of the displacement vector d between the rays at time s and t :

$$d(s, t) = R_1(t) - R_2(s)$$

$$d(s, t) = (q + tv) - (p + su)$$

The smallest distance between a ray and point is equal to the distance along a line perpendicular to the ray, through the point. Similarly, the smallest distance between the two rays R_1 , R_2 , is smallest where a line perpendicular to both R_1 and R_2 that intersects both of them.

d_0 is defined as the minimum displacement vector between the 2 rays. Because of its perpendicularity to both R_1 and R_2 the following constraints can be added to d_0 :

$$v \cdot d_0 = 0$$

$$u \cdot d_0 = 0$$

The dot product of two vectors is only 0 if they are perpendicular to each other.

Because d_0 still follows the general displacement equation $d(s, t)$ it can be filled into the constraints:

$$u \cdot ((q + tv) - (p + su)) = 0$$

$$v \cdot ((q + tv) - (p + su)) = 0$$

reworking these equations to move the unknowns s and t out of them:

first equation:

$$u \cdot q + u \cdot (tv) - u \cdot p + u \cdot (su) = 0$$

$$u \cdot (tv) - u \cdot (su) = -u \cdot q + u \cdot p$$

$$t(u \cdot v) - s(u \cdot u) = -u \cdot (q - p)$$

second equation:

$$v \cdot q + v \cdot (tv) - v \cdot p + v \cdot (su) = 0$$

$$v \cdot (tv) - v \cdot (su) = -v \cdot q + v \cdot p$$

$$t(v \cdot v) - s(v \cdot u) = -v \cdot (q - p)$$

Substituting the inner products for simplicity:

$$a = v \cdot v$$

$$b = v \cdot u = u \cdot v$$

$$c = u \cdot u$$

$$d = v \cdot (q - p)$$

$$e = u \cdot (q - p)$$

gives:

$$bt - cs = -e$$

$$at - bs = -d$$

rewriting the second equation for t :

$$t = \frac{bs - d}{a}$$

substituting t into the first equation and solving for s :

$$bt - cs = -e$$

$$b \frac{bs - d}{a} - cs = -e$$

$$b(bs - d) - acs = -ae$$

$$b^2s - bd - acs = -ae$$

$$(b^2 - ac)s = -ae + bd$$

$$s = \frac{-ae + bd}{(b^2 - ac)}$$

substituting s back into t gives us:

$$t = \frac{-be + cd}{(b^2 - ac)}$$

Whenever the denominator $(b^2 - ac)$ equals 0, the 2 equations are dependent and the 2 lines are parallel. In practice with the way the projector and camera are set up this cannot happen. The solutions found for s and t are fed back into $R_1(t)$ respectively $R_2(s)$ which give a world coordinate each, averaging the 2 gives the reconstructed scene point.

3.4.3 Differences

In the case of ray-plane intersection, the assumption that each row (or column) is a perfect mathematical plane does not take into account any distortion that the projector may add to the plane. As such, results will be less precise than when ray-ray intersection is used. For ray-ray intersection each projector ray is modeled according to the calibrated projector distortion and has the potential to be more precise on an individual basis, whereas for ray-plane this precision is removed by fitting a plane closely to these rays.

The downside of ray-ray intersection on the other hand is that if either the column or row is decoded incorrectly, then the used projector ray is also incorrect. Whereas this can be spotted in the case of a ray-plane intersection by comparing the differences in depths between columns and rows, thus possibly resulting in the rejection of the depth, this cannot be done using just ray-ray intersections.

3.5 Drawbacks binary decoding

While in theory the method is quite straightforward, in practice labeling each pixel with the correct encoding is a difficult problem.

Thus far we have assumed that a pixel besides some ambient light will only receive direct light from the projector. The incoming light for a specific camera pixel that is lit directly under a pattern when capturing the pattern and its inverse can be modeled like this:

$$\begin{aligned} \text{incoming}_{reg} &= \text{ambient} + \text{direct}_{reg} \\ \text{incoming}_{inv} &= \text{ambient} + \text{direct}_{inv} \end{aligned}$$

Taking the difference between the images cancels out the ambient component, leaving only the direct components to determine whether the pixel was lit.

Unfortunately, in practice cameras suffer from noise, projectors suffer from defocus and most materials have some reflecting or translucent component resulting in indirect light transference throughout the scene. Updating the model to also take this indirect light into account from the other pattern gives the following equations:

$$\begin{aligned}
\text{incoming}_{reg} &= \text{ambient} + \text{direct}_{reg} + \text{indirect}_{reg} \\
\text{incoming}_{inv} &= \text{ambient} + \text{direct}_{inv} + \text{indirect}_{inv} \\
\text{difference} &= (\text{direct}_{reg} + \text{indirect}_{reg}) - (\text{indirect}_{inv} + \text{indirect}_{inv})
\end{aligned}$$

Taking the difference between the two still cancels out the ambient light. However, as the indirect component gets stronger in a scene, the direct light will get weaker as the light is converted to indirect light. In the extreme case, no direct component is present at all making the binary function to determine whether the pixel was lit:

$$\text{indirect}_{reg} > \text{indirect}_{inv}$$

Because the effects of indirect light cannot be predicted reliably, the encoding can swing either way. Therefore, the weaker the direct light in a scene, the less reliable the decoding. In the next sections we will look at how different types of projector patterns are affected by different types of indirect lighting.

3.5.1 Errors

In the decoding section we could see that each image contributes differently to the column number, which also affects the severity of the error whenever a pixel is incorrectly labeled in a single image.

For the binary pictures a pixel labeled incorrectly in the i th image out of a total of n images, contributes to a decoding error of 2^{n-i} .

As will be seen in the next sections, some of the patterns are more susceptible to certain kinds of indirect light and therefore errors. In particular the frequency determines how well a pattern behaves for a type of material. High frequency patterns, with a lot of black to white crossings have narrow stripes and are less susceptible to inter reflectance, but more to subsurface scattering. Whereas low frequency patterns with few black to white crossing and broader stripes are the exact opposite.

In the case of the binary pattern, whenever strong subsurface scattering occurs, the high frequency patterns that correspond to the least significant bit of the column encoding, fail. They will therefore not contribute in a useful matter to the encoding, effectively reducing the resolution of the reconstruction.

However, whenever strong inter reflectance occurs, the low frequency patterns that correspond to the most significant bit of the column encoding fail.

They will therefore contribute to a significant depth error.

Alternatives to the binary codes can be created to minimize the effect of particular indirect light effects. One popular encoding is the Gray encoding (named after Frank Gray, not the color). Gray encoding patterns have the property that two subsequent numbered column encodings differ by a maximum of one bit (one image), this opens up the potential to do error correction in the case where very deviating encoding numbers are found next to each other.

column number	binary encoding	Gray encoding
0	0000	1111
1	0001	1110
2	0010	1100
3	0011	1101
4	0100	1001
5	0101	1000
6	0110	1010
7	0111	1011
8	1000	0011
9	1001	0010
10	1010	0000
11	1011	0001
12	1100	0101
13	1101	0100
14	1110	0110
15	1111	0111

A new encoding pattern can be made by switching 2 binary encodings. By doing this in a structured way specific properties can be added to the new encoding pattern. To decode the new pattern a table such as the above can be created to map an observed pattern, in this case the Gray encoding, to the actual column number. The algorithm to decode columns using a specific type is given below. However if a single image bit is incorrectly decoded without error correction, the effects can be even more devastating than for the binary encoding. For example for the Gray encoding, the difference between the very first and last column number is solely the first bit, and getting it wrong therefore results in a massive depth error.

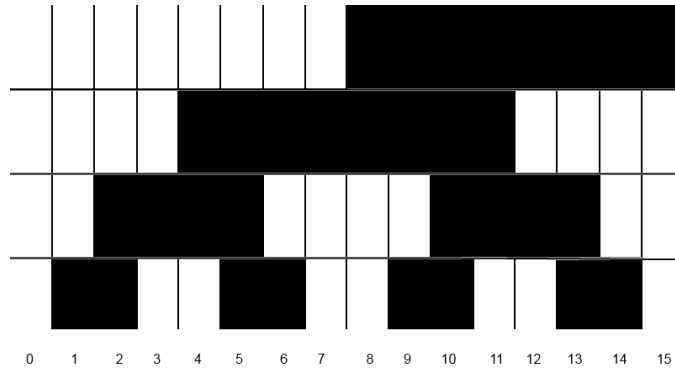


Figure 3.5: first four images of the Gray encoding. the same number of 2^4 different columns can be distinguished. Adjacent columns differ only by one bit, this makes error detection and potentially error correction easier.

Algorithm 3 Decode specific pattern

```

1: procedure CREATETRANSLATIONTABLE(projectorImages)
   decodingTable  $\leftarrow$  new int[ $2^{\text{number}(\text{projectorImages})}$ ]
2:   for  $x \leftarrow 0$  to projectorImages.width do
3:     observedCode  $\leftarrow 0$ 
4:     for  $i \leftarrow 0$  to number(projectorImages) - 1 do
5:       if (projectorImages[i].intensity( x, 0) > 0 ) then
6:         observedCode  $\leftarrow$  observedCode +  $2^i$ 
7:       decodingTable[observedCode]  $\leftarrow x$ 
8:   return decodingTable
9:
10: procedure DECODE(cameraImages, decodingTable)
11:   for each Pixel p in baseImage do
12:     if p.intensity  $\geq$  threshold then
13:       observedCode  $\leftarrow 0$ 
14:       for  $i \leftarrow 0$  to numImages - 1 do
15:         if cameraImages[i].intensity(p)  $\geq$  threshold then
16:           observedCode  $\leftarrow$  observedCode +  $2^i$ 
17:         decodedColumns[p]  $\leftarrow$  decodingTable[observedCode]
18:   return decodedColumns

```

3.6 Inter Reflectance

One of the main sources of indirect light is inter reflectance. In particular concave objects made out of reflective materials such as metals suffer a lot from the effect. This happens often to the point where (almost) no direct light can be observed for some directly lit pixels. The angle to the surface normal of a reflecting object determines the angle at which the projected light is reflected away. Because objects have locally similar surface normals that usually change smoothly, two points close together on an object are likely to reflect light roughly into the same direction, hitting the same spot. For decoding this means that a single pixel can easily receive only the direct light from the projector when it is under a particular pattern, while receiving indirect light from a number of pixels in close proximity of each other under a different non-direct light pattern, thus getting an incorrect bit assignment.

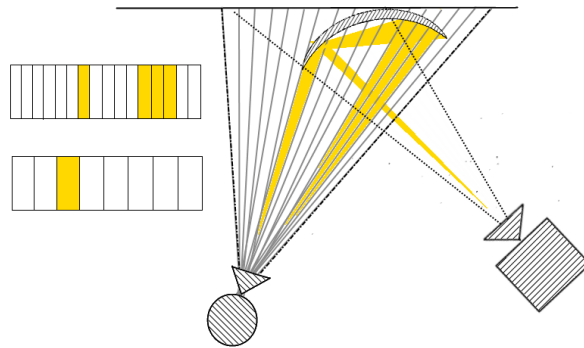


Figure 3.6: Effects of inter reflection for a projector pattern with the column encoded as a single lit column pattern shown on the bottom left. The concave object reflects the direct light from the pattern onto different parts of the scene, which are then captured and incorrectly decoded by the camera in the image at the top left.

Assume there is a scene point captured by pixel p that is under direct light from a particular (regular) pattern. The inverse of this pattern does not light p directly. It however does light a patch of other pixels, all with normals that make them reflect light indirectly onto p . Apart from the direct light p receives from the regular pattern, it receives no indirect light.

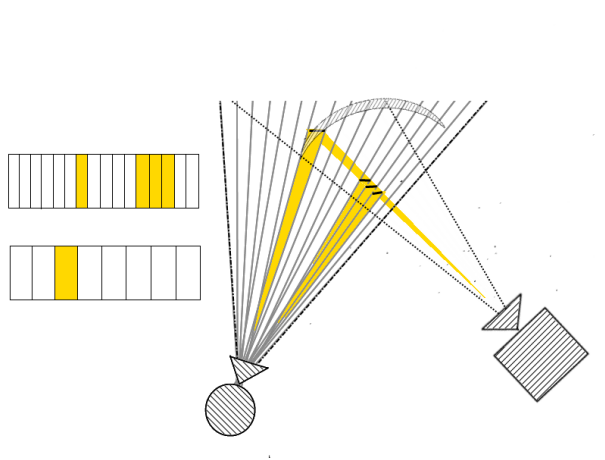


Figure 3.7: The decoding captured in the previous image is used to reconstruct the scene. The incorrectly labeled pixels result in reconstructed depths that do not correspond to any depth in the scene, at the location where the two supposedly corresponding rays intersect.

The binary function used to decode the pattern was:

$$\text{direct}_{reg} + \text{indirect}_{reg} > \text{direct}_{inv} + \text{indirect}_{inv}$$

as p receives no indirect_{reg} light, the term can be removed:

$$\text{direct}_{reg} - \text{direct}_{inv} > + \text{indirect}_{inv}$$

Now whenever the indirect light is stronger than the difference between the direct light in both images, the image is decoded incorrectly. Because there is only one source of direct light, while there can be multiple of indirect light, this situation can easily happen for concave objects.

The problem can however be solved by moving half the indirect light from the inversed pattern to the regular pattern and the other way around. This would lead to the following formula:

$$\text{direct}_{reg} + \frac{1}{2} \text{indirect}_{inv} + \frac{1}{2} \text{indirect}_{reg} > \text{direct}_{inv} + \frac{1}{2} \text{indirect}_{inv} + \frac{1}{2} \text{indirect}_{reg}$$

with both sides having the same amount of indirect light from both patterns, they cancel each other out, once again leading to the formula without indirect light:

$$\text{direct}_{reg} > \text{direct}_{inv}$$

The closest thing possible to doing this is by inverting half the pixels of each stripe to reduce the reflected light to roughly half the original amount. This can be done by performing an XOR operation with a high frequency pattern on a set of projector patterns.

XOR	0	1
0	0	1
1	1	0

Below is shown how a four image binary encoding is XOR-ed with an image of stripe width two, to create a new four image encoding of a higher frequency.

Binary encoding																
	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Width 2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
XOR																
	0	0	1	1	0	0	1	1	1	0	0	1	1	0	0	0
	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

Because an XOR on a pattern with itself results in a 0 pattern, it is taken out and replaced by a pattern with one pixel wide stripes:

final encoding																
	0	0	1	1	0	0	1	1	1	1	0	0	1	1	0	0
	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
observed encoding	0	3	14	13	4	7	10	9	8	11	6	5	12	15	2	1
corresponding column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The resulting patterns are of at least the same, or a higher frequency. The encoding has the same distinguishable column resolution in the same number of images as before, but the individual image patterns are now more robust to inter reflections.

3.7 Subsurface scattering

When light hits the surface of an object, apart from scattering diffusely or reflecting, it may also enter the material. This effect is called subsurface scattering and is present in (partially) translucent materials such as wax, milk and most organic materials. Photons enter the surface where light hits an object directly. From there on they will interact with the material and scatter in different directions, randomly, with a probability distribution depending by the material. Eventually a photon is either absorbed by the material, or it resurfaces out of the material again at a different location into a different direction than at which it entered. Because photons that stay inside an object eventually end up absorbed, there is only a limited distance a photon can reach inside the material, this gives the subsurface effect also a limited range. Due to the random scatter directions photons are more concentrated closer to the point of entry, and thus more of them will resurface closer to the point of entry, resulting in a stronger intensity.

This makes subsurface scattering a very local effect that affects nearby points in the scene.

Assume we have five adjacent pixels o, p, q, r, s in that order. For a given projector pattern o, q and r are lit directly. For the inversed pattern only p and r are lit directly. Because of subsurface scattering each pixel is also affected by indirect light from pixels directly adjacent to it.

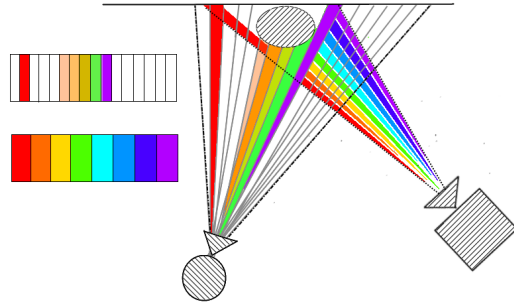


Figure 3.8: Effects of subsurface scattering; with the columns encoded as a single color. Colors and therefore columns get mixed when the light scatters inside the object. The camera observes multiple colors at certain pixels. Any of the columns that are in the observed mix could be the correct match. For the binary encoding, the mixing of light particularly hurts the high frequency pattern which correspond to the lowest column number bits. This therefore mostly affects the depth resolution. For any of the other time multiplexing patterns, the effects are unpredictable, as the influence of a single incorrect image on the decoded column number varies per image.

affects	o	p	q	r	s
o	directly	indirectly			
p	indirectly	directly	indirectly		
q		indirectly	directly	indirectly	
r			indirectly	directly	indirectly
s				indirectly	directly

In each row: the directly lit pixel. In each column: whether another pixel is affected by this.

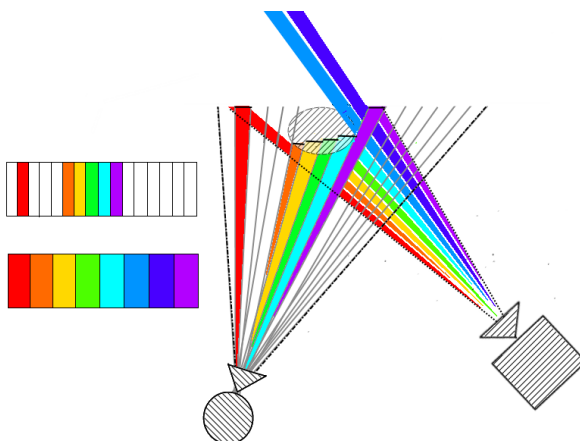


Figure 3.9: A potential reconstruction of the encoding shown above. Because of the mixture of column encodings, a solution needs to be picked somehow. In this case the chosen encoding does not match the scanned object and large parts of the object are therefore shaved off.

regular	o	p	q	r	s
o	directly	indirectly			
q		indirectly	directly	indirectly	
s				indirectly	directly
inversed	o	p	q	r	s
p	indirectly	directly	indirectly		
r			indirectly	directly	indirectly

In each row: whether a pixel is lit under the particular image. In each column: whether another pixel is affected by this. For pixel q this shows that under the regular pattern it only receives its own direct light. However whenever it is under the inversed pattern it will receive indirect light from both p and r! This makes this pattern very susceptible to subsurface scattering.

Now assume that the same five adjacent pixels o, p, q, r, s are affected by a different pattern for which p, q and r are lit simultaneously. For the inversed pattern only o and s are lit. The subsurface scattering still only affects pixels directly adjacent.

regular	o	p	q	r	s
p	indirectly	directly	indirectly	indirectly directly	indirectly
q		indirectly	directly		
r			indirectly		
inversed	o	p	q	r	s
o	directly	indirectly		indirectly	directly
s					

Now whenever the regular pattern is used pixel q receives its own light and the indirect light from p and r. When the inversed pattern is used pixel q receives no light at all, making it a lot less susceptible to subsurface scattering.

For pixels q and r whenever the regular pattern is used they receive their own light and the indirect light from q. When the inversed pattern is used they receive the indirect light from o or s. Assuming the indirect light from all pixels is roughly equal, this should still offer some robustness for this level of subsurface scattering.

As subsurface scattering gets stronger and affects more pixels, the effects become more apparent: high frequency patterns become less and less useful, which in the case of binary encodings reduces the resolution of recovered depth map.

3.8 Best of both worlds

As shown in the previous sections, subsurface scattering and inter reflectance benefit and suffer from opposite types of patterns. With a priori knowledge of a scene it is possible to design an encoding out of only high frequency or low frequency patterns to combat either of the indirect light problems. But without a priori knowledge, or in a scene that suffers from both problems, results might only worsen using a specific pattern. Gupta et al.[?] handle this problem by using four separate encodings: the robust Gray encoding, one encoding with only low frequency patterns that maximizes the width of its minimum stripes (MaxMinSW), and 2 encodings with only high frequency patterns, created by using the XOR technique on the Gray encoding patterns, once with a four width stripe pattern (XOR04) and once with a two width stripe pattern (XOR02). Each encoding consists of 10 regular patterns and their inverses totaling 80 pictures. First each encoding is decoded separately, then a consistency check is done; if two or more decodings match, the encod-

ing is considered correct. If none match, then a pixel receives no encoding. In the experiments section this method is compared to the proposed method.

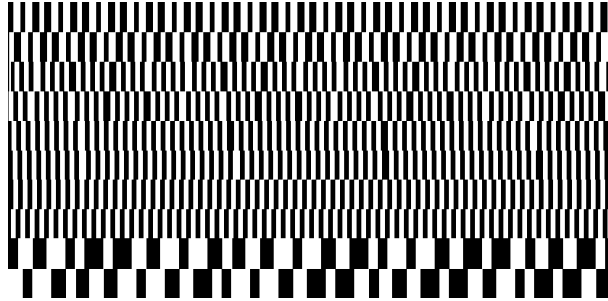


Figure 3.10: Pattern that maximizes the minimum stripe width to maximize the effectiveness versus subsurface scattering.

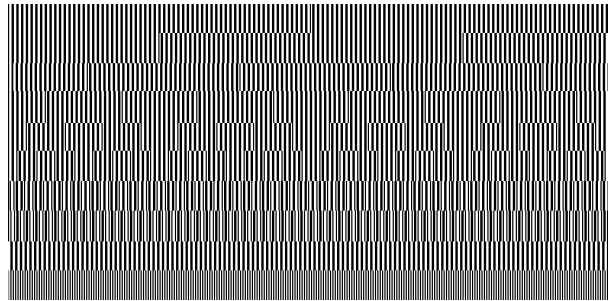


Figure 3.11: The XOR04 high frequency pattern that was created by performing an XOR image operation using the binary pattern of stripe width 4 on the Gray encodings. This should perform well under the effects of inter reflectance

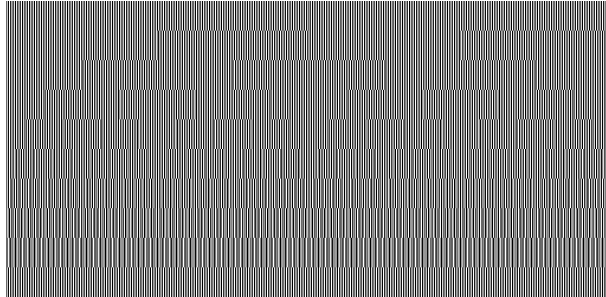


Figure 3.12: The XOR02 High frequency pattern that was created by performing an XOR image operation using the binary pattern of stripe width 2 on the Gray encodings. This should perform well under the effects of inter reflectance

Chapter 4

Approach

Our goal is to make a method that achieves high accuracy at an affordable price, even for difficult scenes with strong indirect lighting. Therefore a time multiplexing method seems most suitable. It has the highest resolution as well as accuracy, while the equipment is relatively low priced. Single shot methods, always give something up to achieve the high density of information. Either resolution is reduced through using resolution for spatial encoding, or accuracy is reduced by using color intensity, which is much more susceptible to various light effects. By using time multiplexing, multiple images can be used to verify or disprove column numbers.

Additionally, the direct column encoding techniques explained in chapter 3 are too definitive, if the decoded column is incorrect, there simply are no alternatives. The approach will therefore first gather potential candidate projector column numbers for each camera pixel and then combine the most likely combination of all of them together to decode the entire scene.

Contrary to most structured light methods that attempt to separate and remove indirect light effects from the captured images, the main focus of this approach is error correction; rather than trying to reduce the light influence directly, the camera sensitivity to light is turned up as much as possible. This makes it possible to detect very weak direct light on objects where this is often not possible with other methods. The effects of indirect light are then handled with consistency checks across the scene.

4.1 Finding the candidates

To improve the chance of correct column decoding, each column will be decoded in a single image. The dependence of a column encoding on multiple images as seen in chapter 3, while faster, has a lot more potential for errors. Each projector image therefore consists of only a single lit column. If the camera captures a pixel that lights up beyond a threshold intensity value for a projector image, then the pixel in combination with the column corresponding to the image is considered a candidate match.

Because it is important that the "correct" match lights up beyond the threshold value for each pixel, camera settings need to be adjusted to be extra sensitive to light. While this yields many more incorrect candidates, the rest of the approach will attempt to remove these incorrect candidates as much as possible.

The most important consistency check, is the one for depth. In addition to running a column scan line across the scene, the same thing is also done for rows. Any pixel that corresponds to a part of the scene that is hit directly by light, should have the same calculated depth for both a row and a column. If this is not the case, then either is incorrect and the match could not have happened. This will prove to be a very useful consistency check.

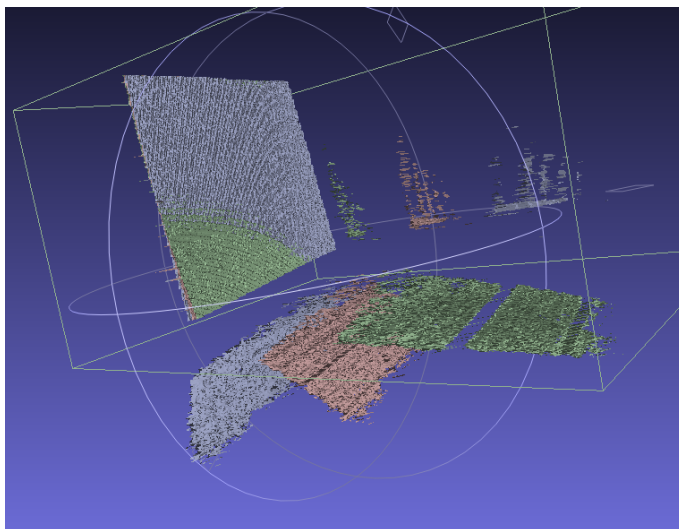


Figure 4.1: A simple scene: A wall is scanned using both columns (green) and rows (gray). Their average is shown in red. Overlapping parts have the same depth as they are lit directly by both. The non-overlapping parts are the result of indirect light and therefore have different depths for columns and rows.

However, running a second scan line also results in having to take projector width plus projector height pictures. Luckily, the second row scan line, can in fact help reduce the acquisition time. This will be shown in section 4.6.

4.2 Dynamic programming

The goal of the decoding process is to assign each pixel its correct column and row. This is done by using dynamic programming to find the best assignments for the entire scene rather than just for individual pixels.

First to reduce complexity, the potential matches are split per image row. Each row will be decoded individually and then later stitched together.

The decoding problem is best modeled as a graph. In this graph each candidate camera-projector match corresponds to a node. A directed edge between two nodes in the graph corresponds to the likeliness that both candidates are both the correct match based on solely each other.

In this graph, a path of multiple edges corresponds to a decoding, using all the matches that correspond to the traversed nodes on the path.

The goal of finding a correct decoding for an image row, thus corresponds to finding a path from a node with the very first pixel id, to a node with the last pixel id.

Such a path should not contain the same pixel id twice, as this would give no answer as to what column and row correspond to that pixel. To achieve this, edges between nodes only exist from nodes with lower pixel id's, to nodes with a higher pixel id. Paths can therefore only move forward and skipping nodes with some pixel id means that they do not get decoded.

This however still leaves a very large number of potential paths. By introducing a constraint that the scene does not contain any holes, the number of paths can be restricted a lot. While this seems very restrictive, in a semi controlled scene where we have a solid background such as a wall, it is not as rigid as it appears.

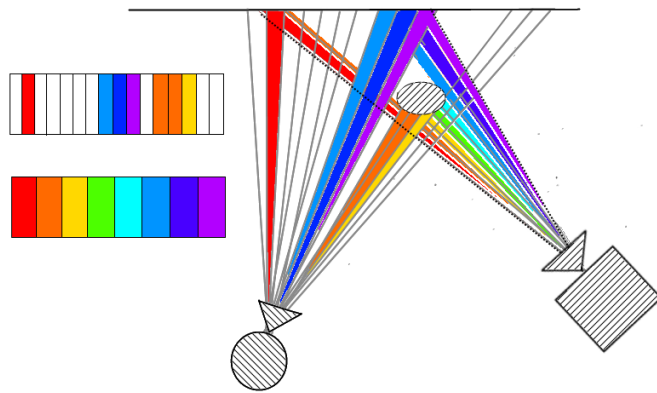


Figure 4.2: An example of a scene with a hole in it. Each color corresponds to a single column, the column color encoding can be replaced by any other type of encoding for the same results.

In figure 4.1, the large hole between the object and the wall results in light passing behind the object. The problem with this is that the order in which the camera observes column numbers is completely unpredictable.

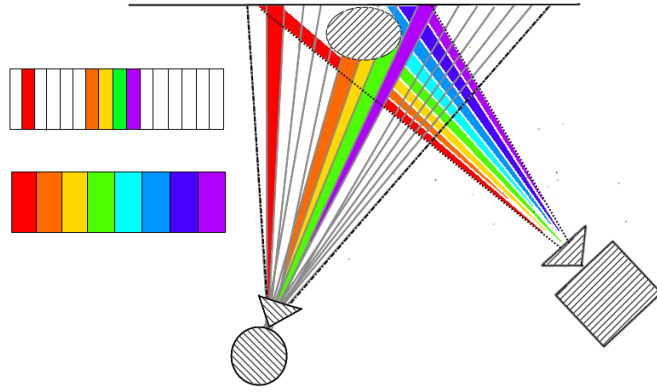


Figure 4.3: An example of a scene without holes. The column numbers are now decoded strictly in a monotonously ascending order.

With the introduced constraint, column and row numbers must now strictly ascend (or descent depending on the placement of the projector relatively to the camera). By using the background as a part of the scene, potential holes can be prevented by placing the object close enough to it to prevent light from passing behind the object in the camera image.

This also means that any edges from nodes with a higher projector column number to nodes with a lower column number cannot exist.

The remaining edges are scored on the likeliness of the connected nodes, and therefore corresponding candidate matches, coexisting in the same scene decoding. This scoring function is explained in detail in the next section.

In addition to the candidate match nodes, a special starting and ending node are added to the graph. These nodes do not correspond to any pixel, but are merely added for convenience. The starting node is connected to every other node, and every candidate match node is connected to the ending node.

Each node in the graph keeps track of what was the lowest cost of the path beginning in the starting node and ending in itself and what was the previous node on this path.

Initially all lowest costs are infinite as none of the nodes have not been explored and there thus exists no lowest cost path from the starting node towards them. The exception is of course the starting node, which has a cost

of 0. By exploring all the starting nodes' edges, all nodes are discovered, which updates their minimum cost path.

Next, the nodes with lowest pixel number explore all their edges and update the lowest cost paths of the nodes that lead to them. Because there exist only edges to nodes with a higher pixel id, whenever a node is exploring its edges, the path leading from the starting node to itself is already final. By exploring the nodes in ascending pixel order thus leads to the minimum cost path in the ending node.

The algorithm for this is straightforward:

Algorithm 4 Decoding

```
1: procedure GRAPH::SOLVEDYNAMICPROGRAMMING
2:   for each Node n1 in nodes do
3:     for each Edge e in n1.edges do
4:       if e.n2.lowestCost > n1.lowestCost + e.cost then
5:         e.n2.lowestCost ← n1.lowestCost + e.cost
6:         e.n2.previousNode ← n1
       return endNode
```

4.3 Cost function

The most important part of the dynamic programming process is the cost function. A bad cost function will lead to bad decodings. To get a better idea of what a good cost function looks like, we will look at some images to see how depth affects the column number:

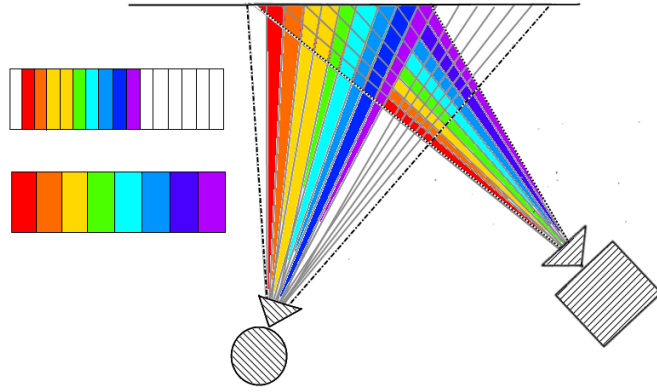


Figure 4.4: An example with no depth changes. As expected, the columns simply change gradually, no gaps in pixel id exist, nor are there any gaps in column number.

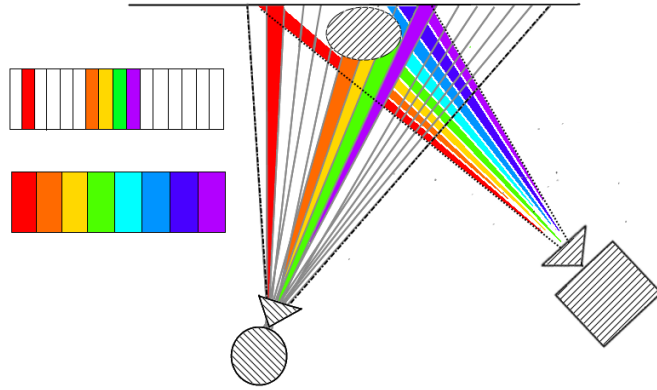


Figure 4.5: An example with some bigger depth changes. From left to right in the observed camera image several things can be observed. Initially the first, red, column is seen, followed by a pixel gap, then the orange, second to fourth columns can be observed, all changing gradually. Then suddenly the 8th column appears without the 5th, 6th and 7th column appearing anywhere in the scene. If we observe how this corresponds with the depth changes, then we can see the following relationship. The higher the column changes over a pixel difference, the more positive a change in depth and the lower the change in column over pixel distance the more negative the change in depth. At extremes, this results in either a gap in pixel ids or a gap in column numbers. Continuity in either one of the two is therefore (likely) preserved when the other is disrupted.

This idea that continuity is preserved, results in an initial, simple but very effective cost function:

Algorithm 5 Simple cost function

```

1: procedure NODE::COST(NEIGHBOR)
2:   cost  $\leftarrow$  -initialBonus
3:   return cost;

```

Each match simply has a negative cost. Because edges are only forward, from a lower pixel number to a higher one in combination with an ever

increasing column number, the minimum cost path will be a no constraint violating, decoding of the scene. This is done with as many matches as possible, as each match "improves" the solution path. However, since some matches are worse than others, the function needs to be refined to take this into account:

Algorithm 6 Better cost function

```

1: procedure NODE::COST(NEIGHBOR)
2:   colDifferenceproj ← Neighbor.colproj - this.colproj
3:   cost ← - initialBonus
4:   cost ← cost + colDifferenceproj2
5:   return cost;

```

Now whenever there is a choice between 2 valid paths of an equal number of matches, the path with the lowest sum of squared column differences is preferred. This is the path with the smoothest differences between columns. This is easily demonstrated by:

$$\left(\frac{1}{2}x\right)^2 + \left(\frac{1}{2}x\right)^2 < 0^2 + x^2$$

$$\frac{1}{2}x^2 < x^2$$

For $x > 0$, with x as the column difference, the cost of splitting x evenly over 2 matches is always lower than when it is split unevenly. This benefits a path with smooth column transitions over multiple matches.

This also means that whenever the squared difference in column number outweighs the initial cost bonus, a match may be skipped entirely. As this also affects the cost path with other matches, an entire potential section may be skipped.

So far we have only considered the 1 dimensional case, where we solve a single image row at a time. However, solving each image row completely individually without consideration of other rows may lead to some very patchy results when all rows are stitched back together. Therefore we must also consider some continuity among the rows:

Algorithm 7 2 dimensional continuous

```
1: procedure NODE::COST(NEIGHBOR)
2:   colDifferenceproj ← Neighbor.colproj - this.colproj
3:   rowDifferenceproj ← Neighbor.rowproj - Neighbor.lastFoundRowproj
4:   rowDifferencecam ← Neighbor.rowcam - Neighbor.lastFoundRowcam
5:
6:   cost ← - initialBonus
7:   cost ← cost + colDifferenceproj2
8:   if rowDifferencecam < thresholdDistance then
9:     cost ← cost + α rowDifferenceproj2
10:  return cost;
```

With α as a chosen parameter, the higher α the more punishing a difference between the decodings of this row compared to the last is. The threshold for camera row difference is in place because of the rigid image row combination nature, if one individual row incorrectly labels a way off pixel, then unlike inside while solving matches in a row, this cannot be corrected and may cause errors that are propagated to other rows. Therefore the threshold may prevent the continuous build up of holes across the entire image after a single incorrect pixel.

To conclude this section: The cost of each edge tends to be negative, which means that any matches are added to the path as long as they do not violate a possible decoding and improve the solution path. Because of the way edges are added, no resulting path can ever violate the constraints. In the case where pixels have multiple matches, a path with smooth column and row changes is preferred because the cost function punishes rough transitions.

4.4 Reducing inter reflectance

This section will show how the the ideas seen in the previous sections will help handle the effects of inter reflection:

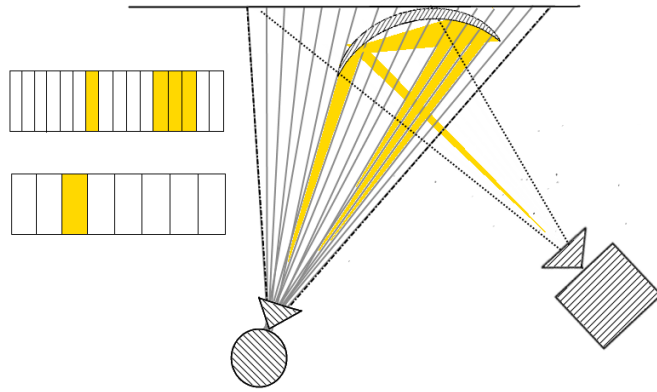


Figure 4.6: An example of inter reflection for a one column scan line projection. Due to the concave nature of the object, the single column is reflected to a different part where it is also observed.

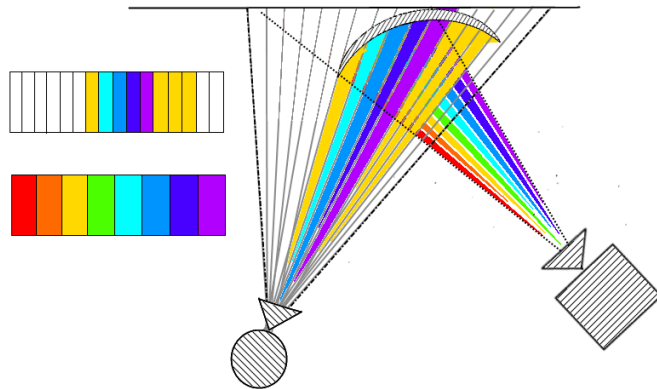


Figure 4.7: By projecting the other columns as well, the combination of observed columns is observed in the camera image on the top left. The inter reflection shown in the previous image is still present.

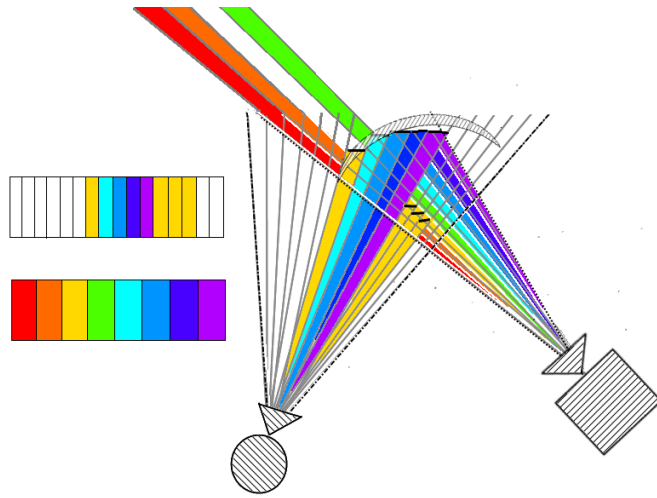


Figure 4.8: The observed decoding found in the previous image and the resulting depth, reconstructed on top of the transparent original object. The reconstructed object now clearly violates the no hole rule, presented in the previous section. This shows that the decoding has to be incorrect, which is indeed the case as the decoded column numbers are not monotonously ascending. Enforcing the monotonously ascending constraint thus helps dealing with inter reflections, as inter reflections will often violate this constraint.

However, there are situations where even this cannot help solve the problem of inter reflections. A depth consistency check can however help with a number of those.

Assume there is a spot p in a scanned scene, whenever light hits p directly, it will reflect light onto a different spot in the scene, q . p and q lie on the same horizontal plane; projector rows light p and q directly simultaneously (while q is also lit indirectly through p).

However, whenever a projector column is projected, whenever p is lit directly, q is lit indirectly through p 's reflection. But when q is lit directly, only q is lit.

If we were to reconstruct the scene, the depth for p would be straightforward: p is only lit by a single projector row and a single projector column, and given a good calibration, both give the same depth.

However for the depth of q this is more difficult: while q is only lit by a single projector row, it is lit by 2 projector columns. Once directly, and once indirectly. By calculating the depths, the indirectly lit candidate can

be ruled out, as any directly lit candidate will have an equal depth for both columns and rows.

While this was a simple case with p and q in the same horizontal plane, when this is not the case reflected indirectly lit spots (almost) always have different depths for columns and rows. This is because the horizontal and vertical displacement of the reflected spot compared to the directly lit spot is independent of the other, the resulting depths are therefore also independent and usually different .

For the remaining cases where indirectly lit spots are still supported by both a column and a row number, there is still the cost function to remove spots that affect the smoothness of the scene.

This does mean that a good calibration is of vital importance, if the calibration is not precise enough, correct candidates may be rejected, while incorrect candidates may be allowed. Because a perfect calibration is unlikely, the depth difference between row and column needs to differ by less than a chosen depth threshold.

4.5 Reducing subsurface scattering

As we have seen in chapter 3, subsurface scattering acts locally, the main problem with the striped projector patterns is that neighboring stripes affect the stripes next to them. By using the single scan line projections this problem is removed because there simply are no nearby stripes any more.

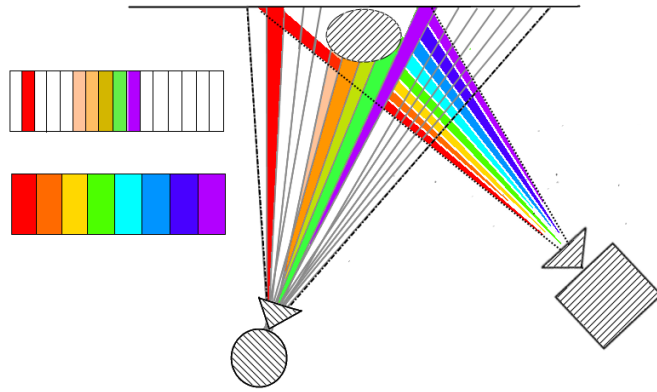


Figure 4.9: The projected columns are locally smeared over the object. This is observed as a mixture of columns by the camera. Choosing which column belongs to each camera pixel is therefore a difficult problem.

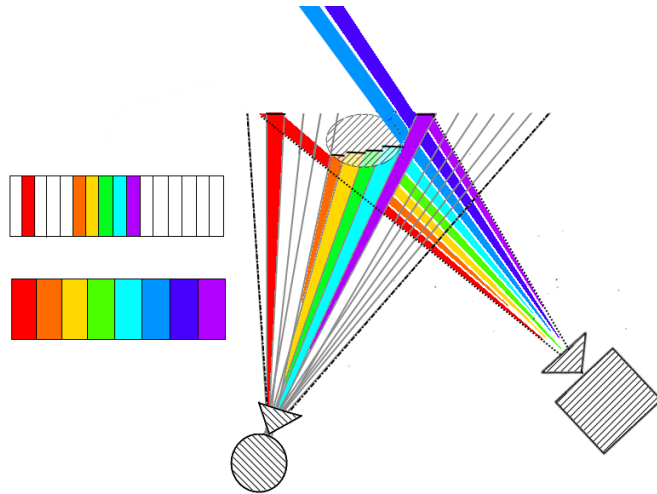


Figure 4.10: A potential decoding as might be chosen by the cost function is reconstructed. Because the cost function craves smooth continuity the object is "shaved" to be as continuous as possible.

In addition, the depth consistency check does very little, as nearby in-

correct column or row matches are likely to be within the threshold depth difference between the two.

It is therefore, that rather than looking at matching pixel-column pairs and pixel-row pairs, pixel-light ranges are used. Whenever a pixel is hit by consecutive columns (or rows), the consecutive matches are grouped together. Candidate matches are gathered in the same way done before, except when a pixel already has other potential matches, the match may be grouped with the other matches into a single pixel-light range object.

Because on average an equal amount of light will spread in all directions, the center of consecutive columns is a crude but reasonable estimate of the center of the correct direct light hit. More sophisticated methods can be devised based on intensity values to find the peak brightness and therefore the center of the direct light. This is for example used by [4] to reach sub-pixel precision in a scan line based system. However, intensity values are often polluted by other sources of indirect light and may therefore introduce even more uncertainty for those difficult to construct objects that this method aims to handle.

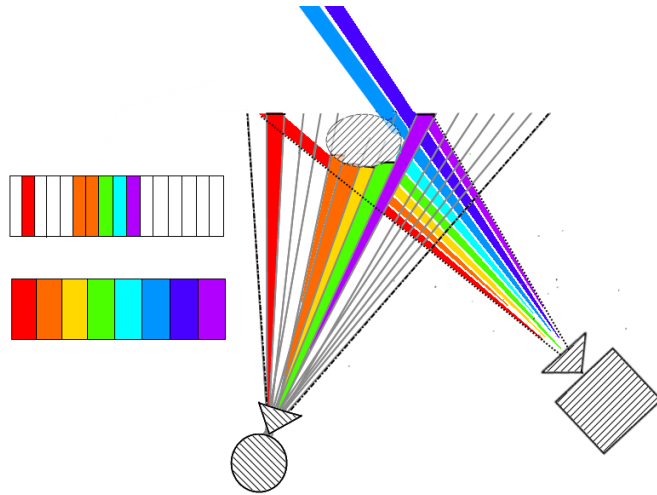


Figure 4.11: A potential reconstruction using the middle column idea for the decoding.

This changes nothing for the cost function, as the center of the pixel-light range object is simply used as the candidate match, while the other potential matches inside the pixel light range object are disregarded.

4.6 Speeding up the process

At this point there are still projector width plus projector height images required. Aside from the time it would take to project and capture all these images, it has a tremendous memory cost. It would therefore be highly beneficial to reduce the number of images that need to be taken, without completely undoing all the benefits that we got from projecting only a single column. Because of the the depth consistency check between rows and columns, it is possible to do this by projecting multiple columns per image, as long as they are spaced far enough apart.

n simultaneous projector rows are used per image, each spaced an equal projector height divided by n apart. Likewise, m simultaneous projector columns are used, each spaced an equal projector width divided by m apart.

Whenever the same camera pixel brightens for both a row image and a column image, then for each of the potential n row matches, the resulting depth is compared to the depth of each of the potential m column matches. Only if two depths match within a threshold difference, then it is a valid candidate match for the corresponding row and column numbers.

Using multiple scan lines cuts down the number of required images as per row only projector height divided by n images are required and per column only projector width divided by m images. The downside is that by having multiple rows or columns in each image, it is a lot more likely that a false positive gets supported by the depth of both a row and a column.

While the dynamic programming process will sort out most false positives, it is slowed down considerably if there are a lot of candidates. Additionally, if a large region is supported by two different pairs of rows and columns, each supporting different depths for the region, then the region may get a correct depth reconstruction relatively to itself, but relatively to the rest of the scene will still be incorrect.

Using additional rows or columns per image will greatly decrease the acquisition time, but may result in an increase in decoding time and accuracy. A good balance between the benefits and drawbacks of using different values of n and m needs to be found.

Chapter 5

Experimentation

5.1 Experimental design

In this section experimental results of the proposed approach on different scenes, each with specific difficulties, are compared to the code ensemble by Gupta et al.[2] consisting of four different encodings, shown at the end of chapter 3, each designed to handle difficult scenes well.

The first few scenes consist of several objects with no specific difficulty. This is done to compare the base accuracy as well as required computation time. Then a metal bowl with a lot of inter reflection is used, as well as an identical metal bowl, painted to make it opaque to show the ability to handle inter reflections. Finally a glass object filled with different concentrations of diluted milk that has a high level of subsurface scattering is used to show how the approach fares under those conditions.

5.2 Implementation

The implementation used is written in C++ and built upon the openCV [14] framework. As a base the standard structured light implementation by Lanman [15] was used. This base implementation contains code to calibrate both the projector and camera using a printed checkerboard pattern and a calibration plane. It has automated the process of projecting and capturing standard binary patterns as well as decoding them and converting them to depths using either ray-plane or ray-ray intersections.

For these experiments the framework was enhanced to decode any specific

decoding patterns (in this case the Gray, XOR04, XOR02 and MaxMinWidth encodings), as well as the suggested line scanning patterns and the code to decode them using a per row and column, dynamic programming approach. As the approach relies on a good calibration, some additions have been made to improve the calibration results. Additionally, each step: calibration, capturing the scene, decoding the captured images, converting decodings to depths, has been made modular to allow for easier experimentation with each step. Finally, the resulting pixel-depths are converted to a .ply 3D model triangulating neighboring vertices if they are within a threshold distance to visualize the results.

5.3 Experimental set up

5.3.1 Camera-Projector calibration

To calibrate the system a web cam, a projector, a calibration plane and a printed checkerboard pattern attached to the calibration plane are required. For our experiments we used the Logitech HD Webcam C270 which captures at a 640x480 resolution and offers software options to adjust its brightness and light sensitivity. The Sony VPL-CX21 projector was used, with a resolution of 1024x768 and the ability to auto-focus. For the checkerboard an A3 format paper was used with a 9x7 printed checkerboard which was taped to a large wooden panel, used as the calibration plane.

The camera and the projector are placed in a way that they both point towards a wall right behind where captured scenes will be placed. To reduce the influence of ambient light, the light in the room is reduced to a bare minimum; lamps turned off, closed curtains. Between the camera and projector some horizontal and vertical displacement is required to make the triangulation for both rows and columns more robust to calibration errors. For the web cam any automatic adjustment settings are turned off to prevent adaption to the general low light environment in between frames. Additionally, images are in black and white to reduce data size. For the projector the auto focus is used once to focus onto the wall.

Now the semi-automated calibration process is started. The calibration plane with the attached checkerboard is placed in full view of the camera in an orientation where it will not move. The web cam continuously takes pictures and runs a procedure to detect the crossing corners between checkerboard

fields using build in openCV functions [14]. If the number of checkerboard fields matches the 9x7 attached board, the projector will project its own checkerboard onto the calibration scene. By taking the difference between the frame captured before the projector was turned on and the frame after the projector was turned on, we get only the checkerboard projected by the projector. On this projected checkerboard the corners are detected using the same function used for the physical checkerboard.

If both checkerboards are detected and confirmed by the user the calibration plane is placed in a new orientation and the process is repeated. Good calibration results are typically acquired using 10+ different orientations, in the case of the experiments we used 15. Because the physical and projected checkerboard are assumed to be on the same (perfect) plane and the calibration will suffer under imperfections, it is imperative that the plane does not move between the frame before- and after the projection of the checkerboard. To achieve this clamps were used to keep the board in place.

In addition, tools were added to inspect the used calibration images. In case of movement of the calibration plane, noise, or anomalies in finding the corresponding checkerboard fields between the 2 checkerboards, images can be suppressed to greatly improve calibration results.

5.3.2 Capturing images

Once the system has been calibrated the next modular step is performed: capturing the scene. The desired projected patterns are loaded. After selecting an appropriate brightness for both the camera and the projector, each pattern is projected onto the scene and subsequently captured by the web cam. This process is fully automated and is run for both rows and columns. While the web cam and projector have a much higher potential frames per second, roughly 4 frames are captured each second to reduce the odds of some projected frames being captured incorrectly.

In the case of using the proposed scan line technique, the user may additionally choose the number of scan lines used for both columns as well as rows.

Additionally, for each set of images the fully lit as well as the unlit image is also captured to find the pixels that are unaffected by the projected patterns. The images are then saved for further use. All images are saved and loaded in black and white, with pixels having an 8-bit intensity value (256 shades of gray).

5.3.3 Camera intensity settings

For the experiments 2 different camera settings are determined: one medium light sensitive setting, where the binary patterns seem to behave well and a high sensitive setting, where the binary patterns hardly function, but single scan lines still show up on very difficult materials.

5.3.4 Scan line decoding

For the decoding process the dynamic programming process explained in 4.2 is used. An intensity threshold of 10 was used to select candidates for pixel-column and pixel-row candidates. A depth difference threshold of 100 was used to combine potential pixel-row and pixel-column candidates into a single candidate.

Subsequent candidate matches are connected by edges if the candidate has less than a maximum of 100 edges with a maximum pixel difference between the candidates of 200. If the candidate has less than 5 edges however, it will ignore this upper pixel distance limit and allow candidates of any pixel distance until this lower limit is reached. Finally any matches within a pixel distance of 25 are added regardless of the maximum of 100 edges.

5.3.5 Cost function

For the cost function the following parameters were chosen: The initial cost bonus is -200 and α is 0.3.

5.4 Experimental Results

In this section, the results of the suggested approach are presented and compared to the code ensemble by Gupta et al.[2]. Each of the results is presented as an untextured point cloud. Results are untextured to not create any false illusions of shape through shaded colors present in photographs. However to enhance the perception of depth and shape in each image, the xray rendering technique is used in the program Meshlab [16].

5.4.1 Generic objects

The first object is a suitcase, see figure 5.1. It is scanned multiple times using different numbers of scan lines to demonstrate the effect on the required time, as well as the precision of each of the scans.



Figure 5.1: The suitcase can be considered a relatively easy object, with smooth depth transitions. Most of its difficulty comes from the several dark parts that reflect a low amount of light. In addition it has reflective metal edges.

Results

First the results of the suggested approach are presented, using only a single column and row scanline.

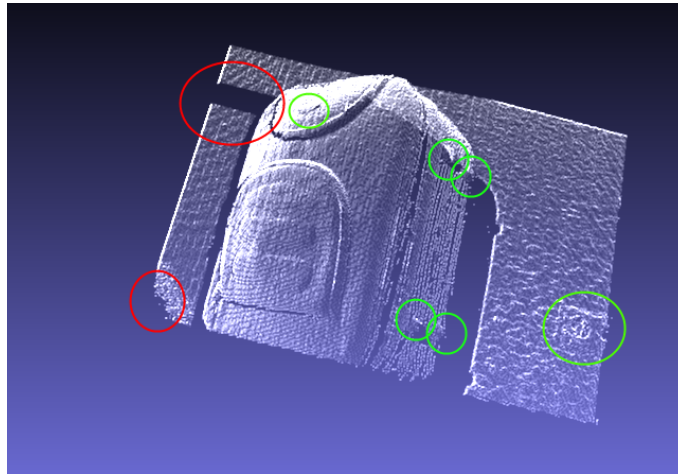


Figure 5.2: Marked in green circles are some properly reconstructed minor details. Marked in red circles are errors (all in the background of the scene).

The error on the top left is caused by a diagonal occlusion; the cost function assumes that an occlusion in a row results in a depth change in that same row. Column number continuity is maintained in that case. In this case however, the occluding shadow is cast diagonally. The column number thus still increases suddenly, resulting in a vast increase in costs, without finding enough compensation in negative costs. This results in part of the background being rejected in favor of a void.

At the bottom left, this error is simply caused by an imperfect calibration, the "correct match" is rejected, because the correct row and column depth are still far apart.

Comparison to code ensemble

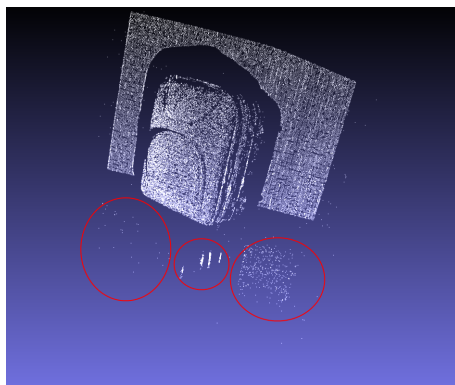


Figure 5.3: Results according to the original code ensemble, consisting of 80 images. While the results are accurate and reasonably noise free, they are also quite sparse. The minor details are therefore much less noticeable. In addition, reflections on the floor have incorrectly been decoded resulting in faulty depths.

Comparison of each individual encoding

Each individual projected pattern in the code ensemble is shown below. For each of the patterns both the rows and columns are used to do a depth consistency check.

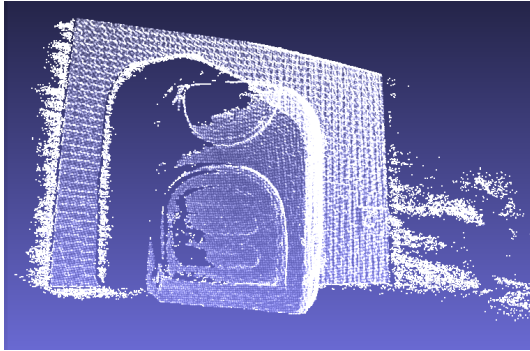


Figure 5.4: Gray encoding

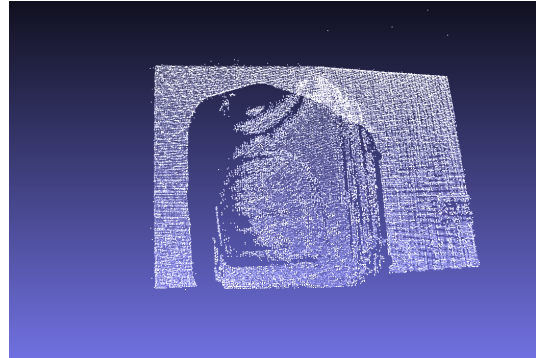


Figure 5.5: XOR04



Figure 5.6: XOR02

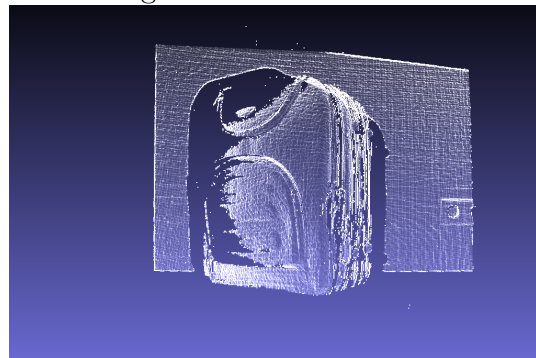


Figure 5.7: Maximum minimum width

The gray encoding, while dense and accurate, has a lot of noise near the edges. The XOR04, is accurate but sparse, this makes details hard to spot. The XOR02, fails nearly completely, as the high frequency patterns are very easily mislabeled. The maximum minimum width pattern is nearly perfect, both accurate and dense.

In general, it seems like high frequency patterns do not do well, this is possibly a problem with the low resolution of the camera; multiple projector pixels seem to correspond to the same camera pixel which causes problems with detecting direct light.

Multiple scanlines

Next we will look at the effect of using multiple scan lines at the same time, both accuracy and speed wise.

In the table below are the number of images required when using different numbers of scan lines using a 1024x768 projector.

On the horizontal axis, the number of simultaneous column scan lines and on the vertical axis the number of simultaneous row scan lines:

	1	2	4	8
1	1792	1280	1024	896
2	1408	896	640	512
4	1216	704	448	320
8	1120	608	352	224

With the acquisition process at a rate of 6 frames per second this results in the following acquisition times in seconds:

	1	2	4	8
1	299	213	170	149
2	235	149	107	85
4	203	117	75	53
8	187	101	59	37

Followed by these decoding times:

	1	2	4	8
1	179	216	372	471
2	177	147	218	249
4	129	125	185	566
8	159	143	476	684

Summing the 2 together to get the total time:

	1	2	4	8
1	478	439	542	620
2	412	296	325	334
4	332	242	260	619
8	346	244	535	721

While the acquisition time decreases more quickly with an increase of column numbers, as the width of the projector is bigger than its height, the decoding and total processing time increase by a whole lot more. This is mostly because the heaviest computation step, the dynamic programming, works on a per row basis. Where most of the columns need to be matched, increasing

the number of potential column numbers seems to increase the number of potential minimum cost paths by more than an increase in row scan lines does.

In contrast, the code ensemble's pictures are taken in 15 seconds (30, if rows are captured as well) and takes 3 seconds to decode.

Multiple scan lines accuracy

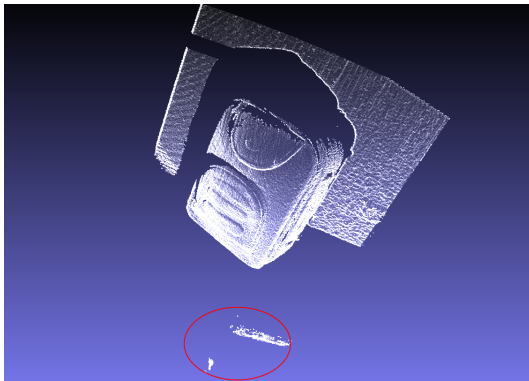


Figure 5.8: 2 columns, 2 rows

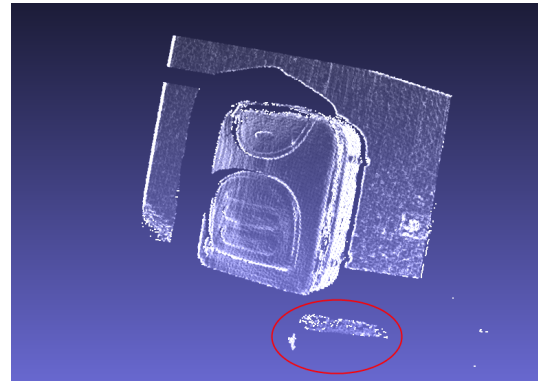


Figure 5.9: 2 columns, 4 rows

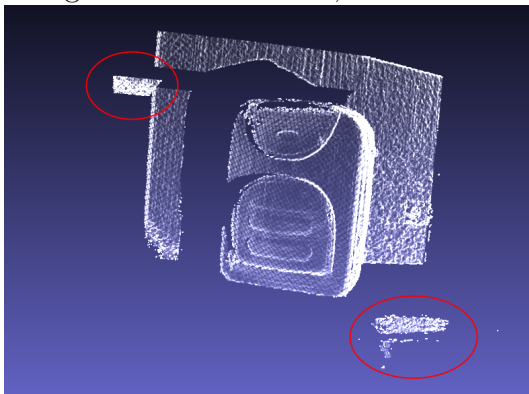


Figure 5.10: 2 columns, 8 rows

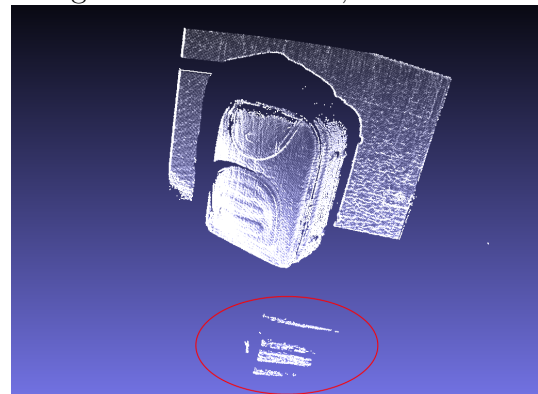


Figure 5.11: 4 columns, 4 rows

The accuracy of the constructed depth is largely unaffected for most pixels. Each of the results show a mostly precise and relatively noise free reconstruction of the object, where all the minor details remain visible. The pixels that are affected most, are pixels that do not have an alternative projector match, such as the reflection on the floor. While the reflection is initially

ruled out as indirect right because there is no depth consistency between any pair of rows and columns. This suddenly changes when the number of scan lines is increased. Because there are no direct light alternatives, the cost function stands no chance and happily matches the reflections as if it were direct light of the projector.

The next scene contains pillows, a teddy bear and a stuffed animal dog. Difficulties are the black parts of the dog that reflect very little light, the subsurface scattering of the pillows and animals in combination with some fine details and the reflective floor.

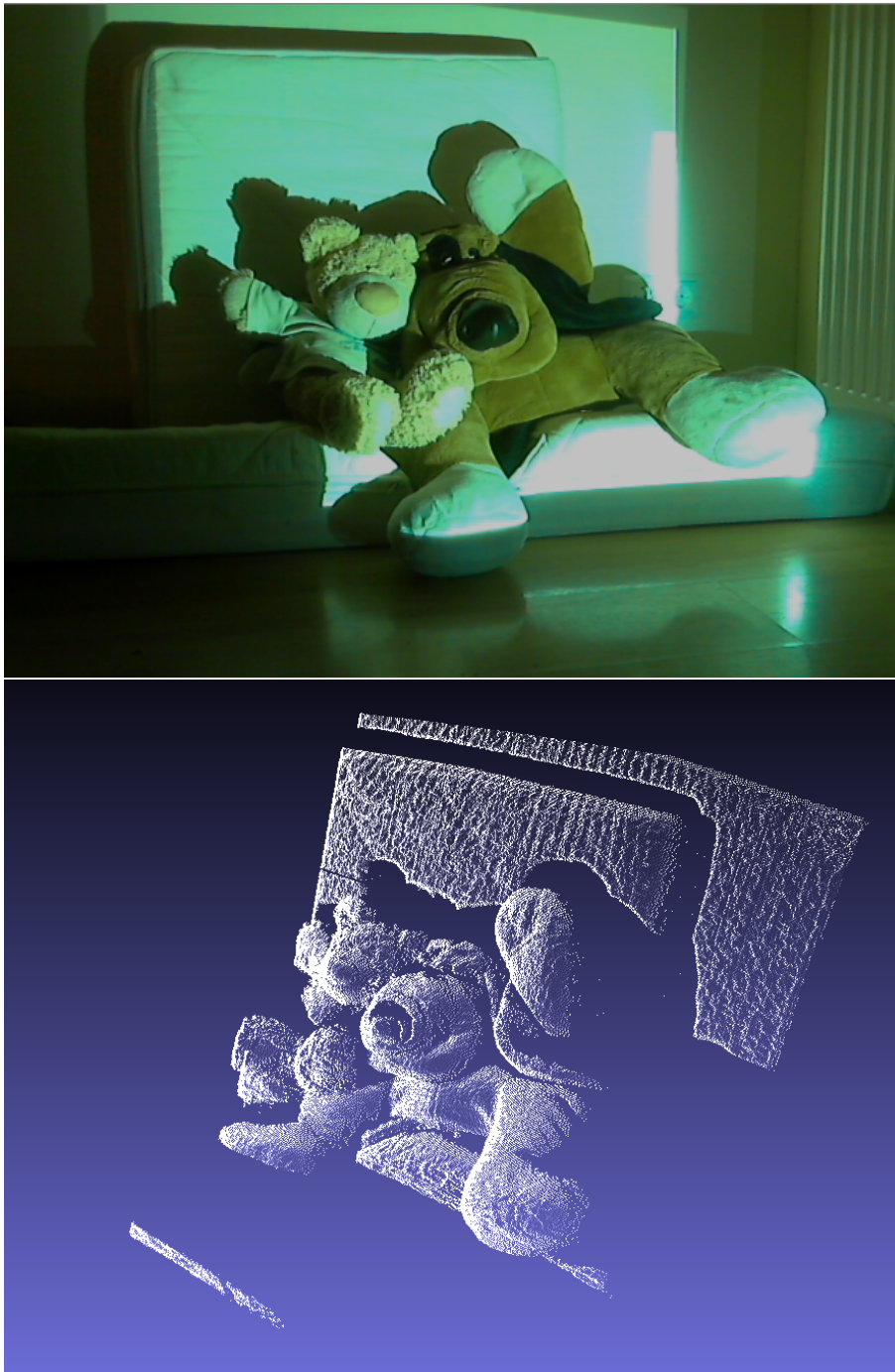


Figure 5.12: Our proposed approach using 2 columns, 4 rows. 704 images: 2 minutes acquisition time, 3 minutes decoding time.

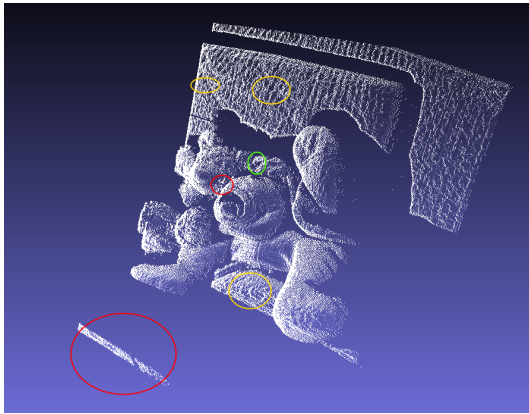


Figure 5.13: Our proposed approach using 2 columns, 4 rows. Yellow markers indicate small details that are reconstructed, but not as sharply as should have been. Bottom red marker is the floor's reflection. Middle red marker is incorrect depth reconstruction in between the stuffed animals' heads due to inter reflectance. Green marker is the good reconstruction of black parts of the dog.

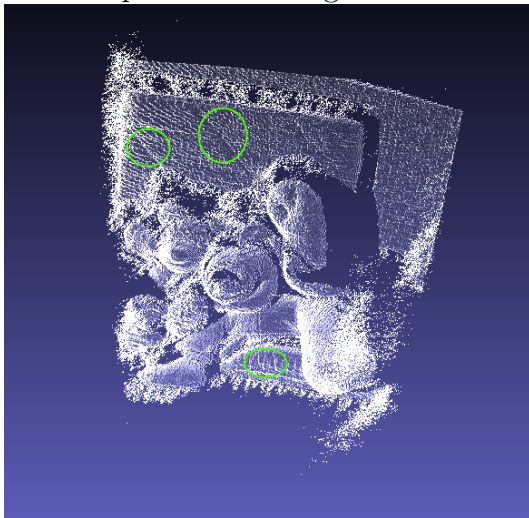


Figure 5.15: Gray codes. While some of the details have been reconstructed nicely, it is very noisy around the edges.

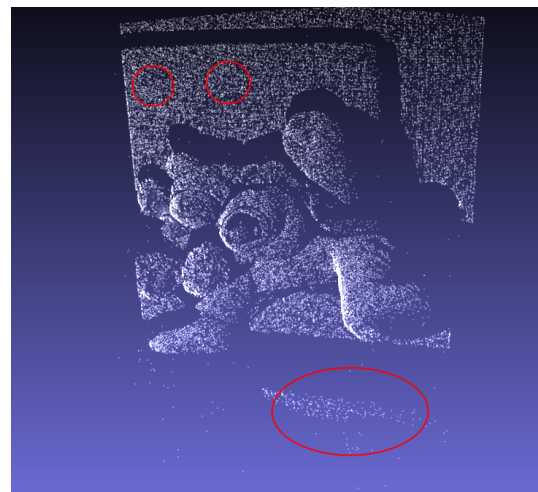


Figure 5.14: Gupta et al. code ensemble, 80 images, only columns. Clean, accurate reconstruction. However, very sparse, resulting in the loss of details where the top red markers are. Black parts on the dog only partially reconstructed. Again there is incorrectly reconstructed depth at the reflection of the floor.

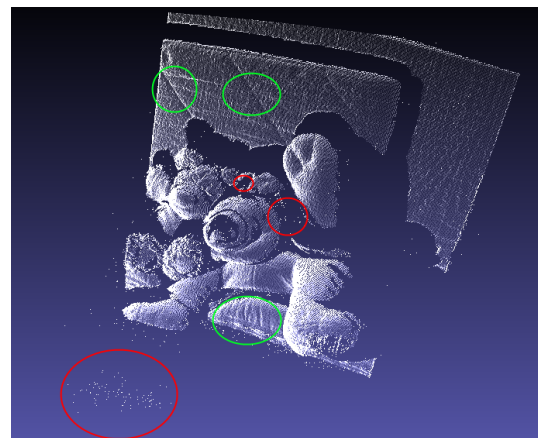


Figure 5.16: MaxiMinWidth codes. Once again excellent reconstruction, a bit noisy, but the fine details have all been reconstructed very sharply. Black portions of the dog are omitted.

The next scene consists of an ornamental metal plate with very fine roughly 1 millimeter thick decorative markings. Difficulties are the general reflectiveness of the plate and the very fine decorations.

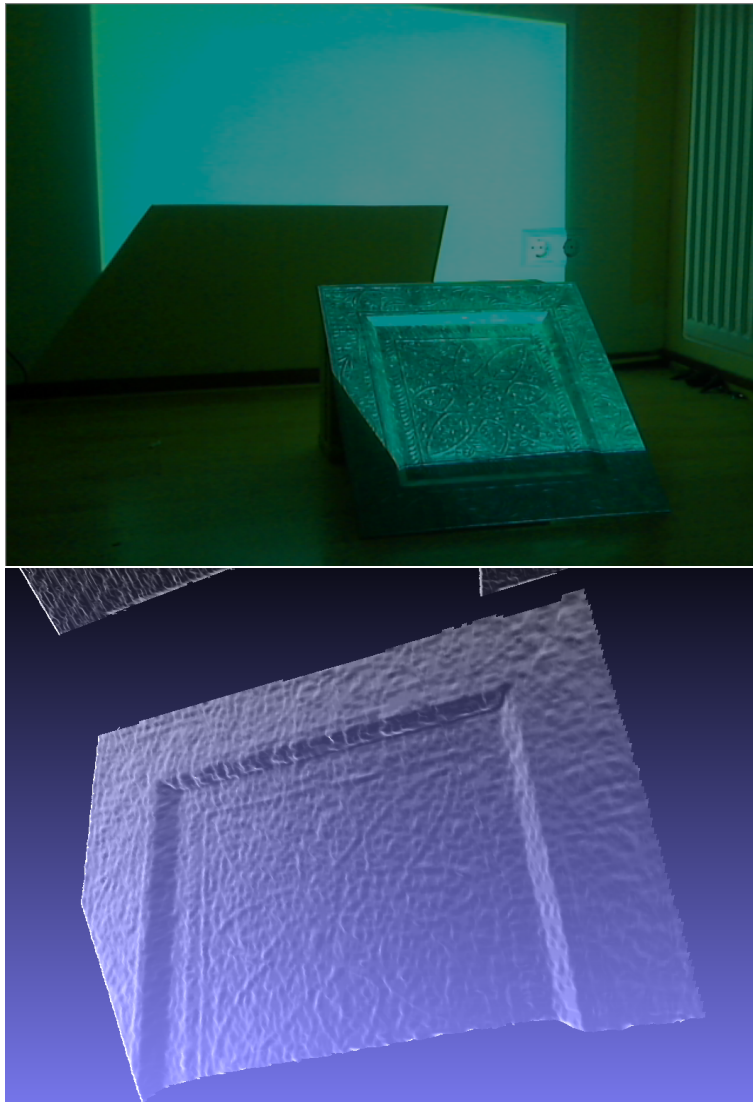


Figure 5.17: Approach using 2 columns, 4 rows. 704 images: 2 minutes acquisition time, 2.5 minutes decoding time.

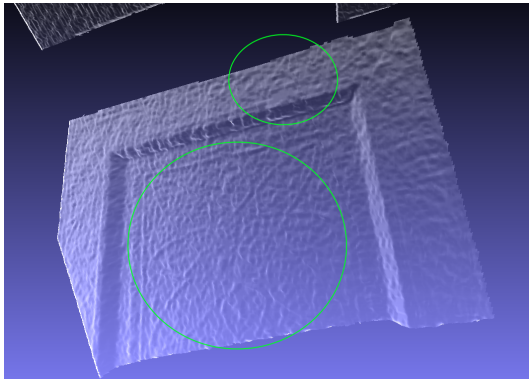


Figure 5.18: Approach using 2 columns, 4 rows. Green markers show an excellent reconstruction of the ornamental platter, especially the thicker center decorations, but even some of the very thin edge decorations are reconstructed.

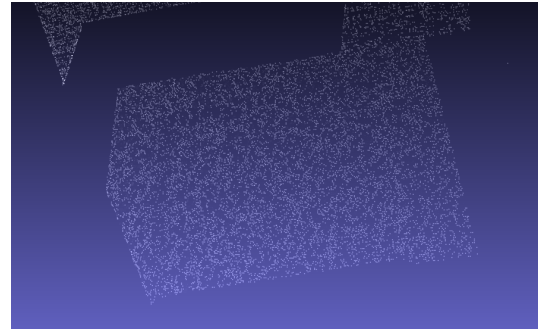


Figure 5.19: Code ensemble, 80 images, only columns. While the shape of the plate shines through, the results are so sparse that none of the decorations show at all.

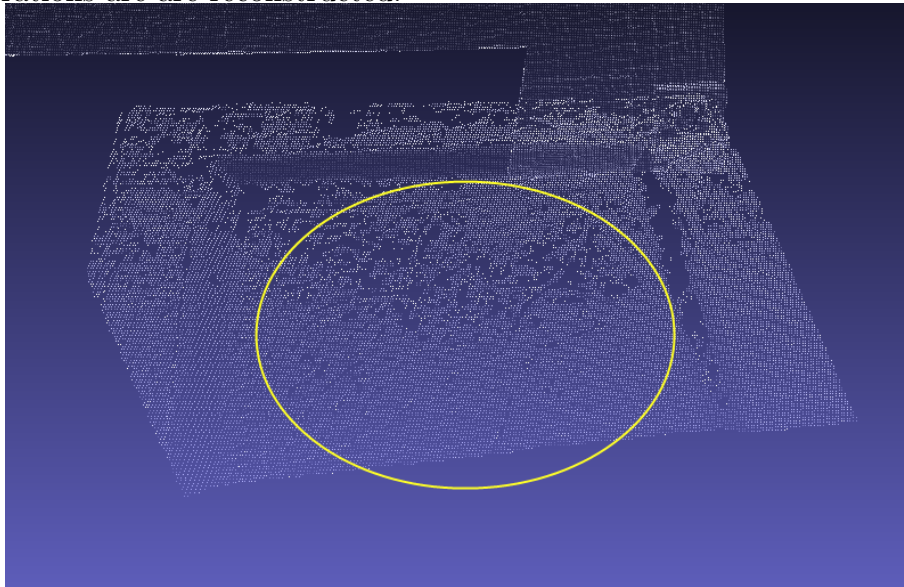


Figure 5.20: Maximum minimum width pattern. The center decorations are visible. However the plate has numerous holes because of reflections and none of the edge decorations show. Other patterns did even worse.

The proposed method handles each of the general scenes very well. Each of the reconstructions is highly accurate and very dense. Even difficult parts that the code ensemble did not handle very well are properly reconstructed.

5.4.2 Objects with strong inter reflectance

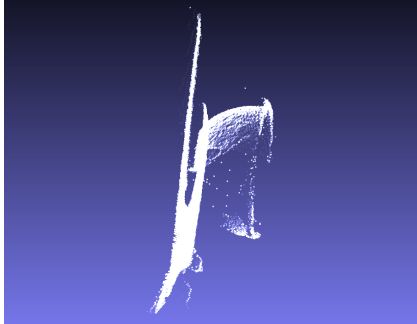
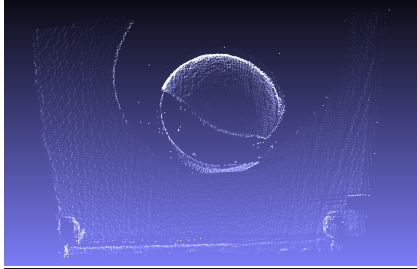
To see how the proposed method truly fares under the influence of inter reflectance, two identical metal bowls with very strong inter reflective properties are used. One is painted white to make it opaque and easier to scan. This bowl will be used as a base measure to compare the metal bowl to. Both bowls are glued to a flat piece of wood to make putting the two bowls in different orientations easier. Unless specified otherwise, the proposed method uses two simultaneous columns and 4 simultaneous rows.

Orientation 1

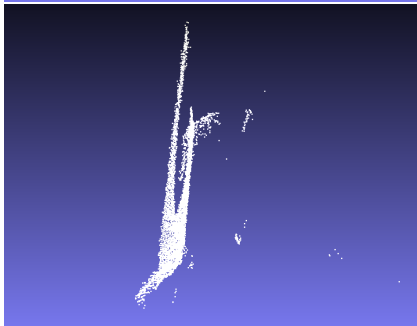
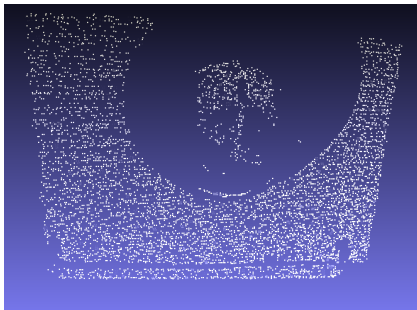


Figure 5.21: The painted and unpainted bowl in the same orientation side by side. Even with the projector not projecting any intentional light, the unpainted bowl already shows a lot of specular light.

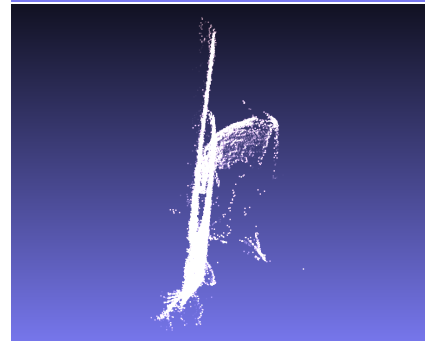
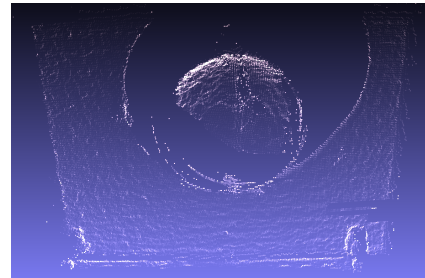
Displaying the back and side view of all reconstructions:



Painted bowl.



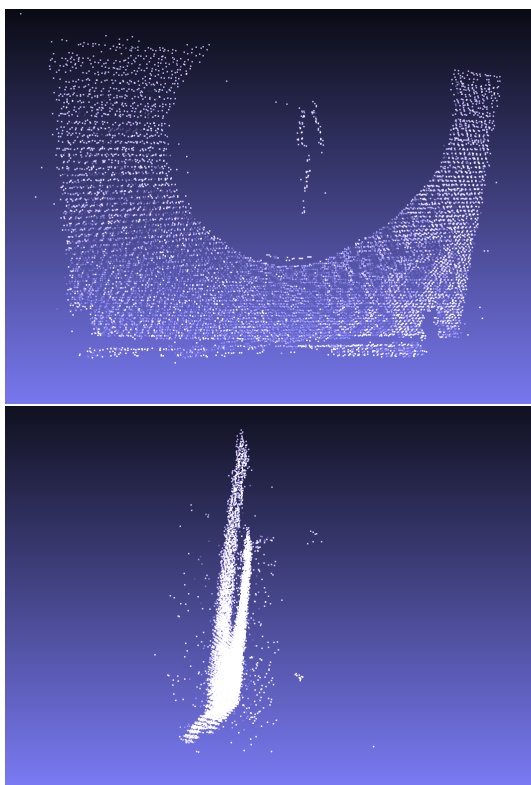
The code ensemble on unpainted metal bowl.



The proposed method on unpainted metal bowl.

The proposed method on the metal bowl shows a clear and dense reconstruction very similar to the painted bowl. Most pixels have gotten a correct depth. Only minor depth mistakes near the most specular parts of the inside of the bowl have been made.

The code ensemble has reconstructed only very little of the inter reflecting portions of the bowl. The model is very sparse in general, but seems to be at least accurate for the parts that it did reconstruct.



XOR04.



XOR02.

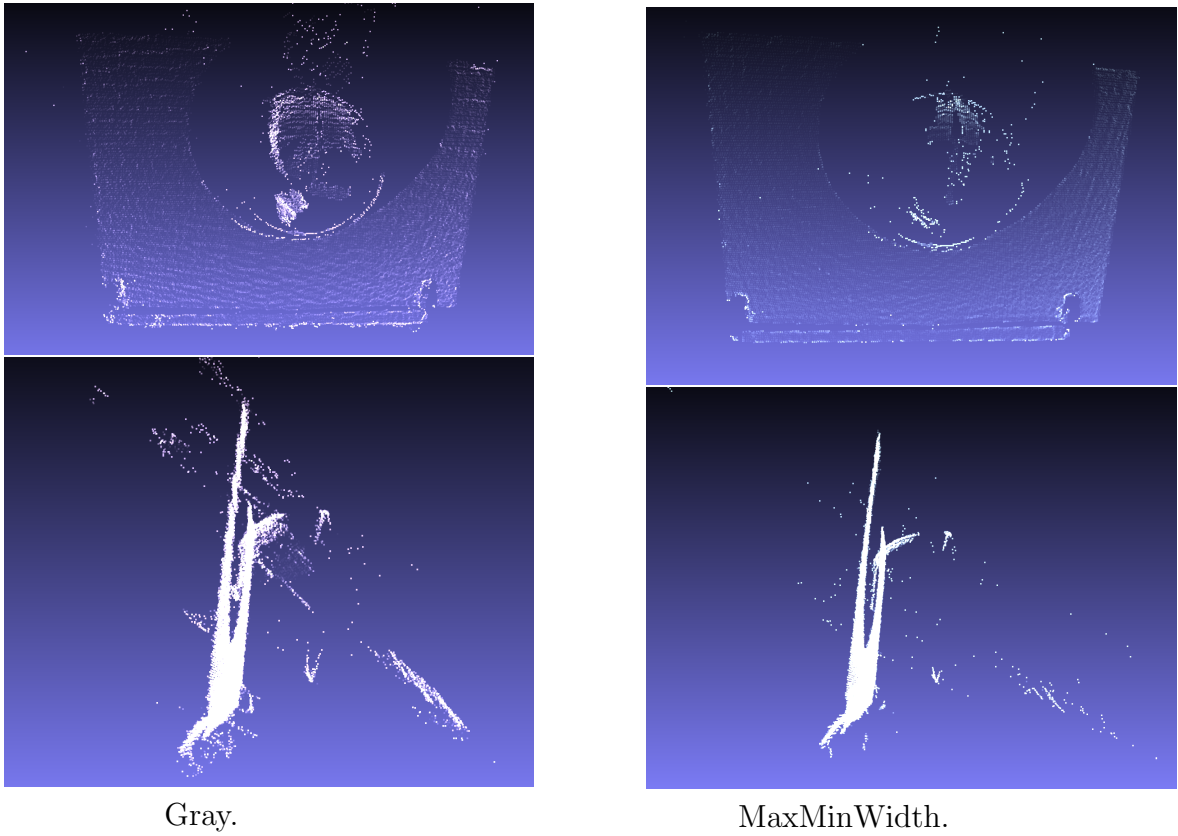


Figure 5.22: Each of the individual encodings used by the code ensemble on the unpainted metal bowl.

Despite being created specifically to handle inter reflectance, XOR02 reconstructs nothing at all. XOR04 barely does any better. The Gray encoding and MaxMinWidth, reconstruct a small portion of the bowl. As expected they however cannot handle the inter reflections.

Orientation 2

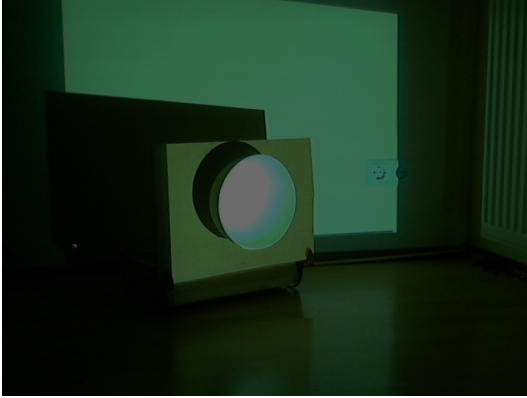


Figure 5.23: Painted bowl as a base for comparison.



Figure 5.24: Unpainted bowl.

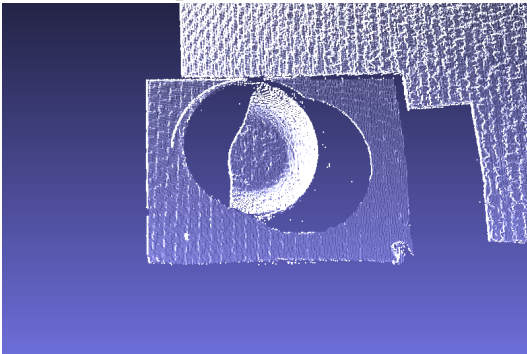


Figure 5.25: Top view painted bowl, constructed using proposed method.

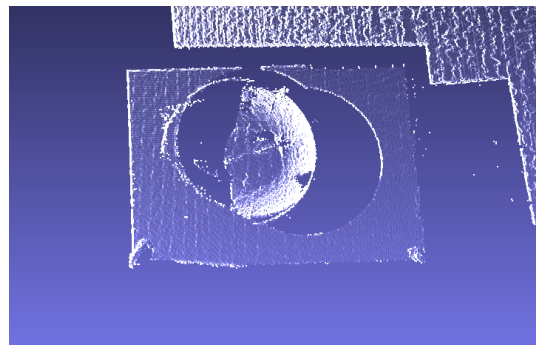


Figure 5.26: Top view unpainted metal bowl, constructed using proposed method.

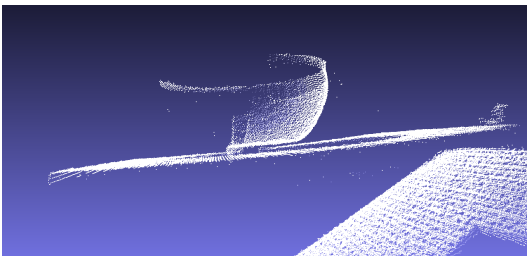


Figure 5.27: Side view painted bowl, constructed using proposed method.

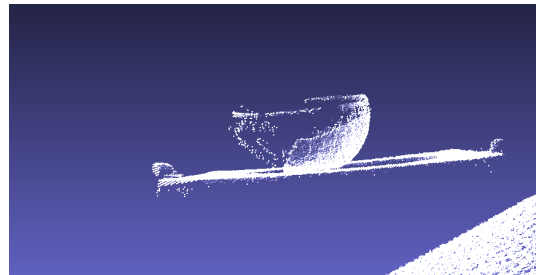


Figure 5.28: Side view unpainted metal bowl, constructed using proposed method.

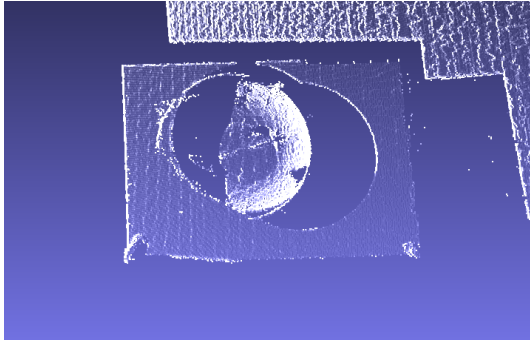


Figure 5.29: Only one column and one row scan line. Results are very good, nearly every pixel is matched perfectly.

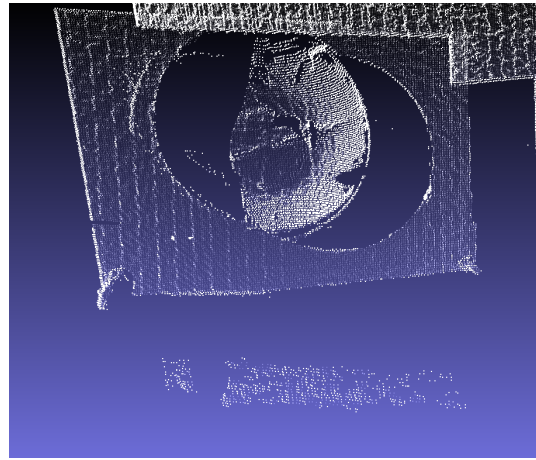


Figure 5.30: two column, four row scan lines. The results are very similar to the one column one row scan line results. The only difference is that the reflection on the floor is incorrectly reconstructed.

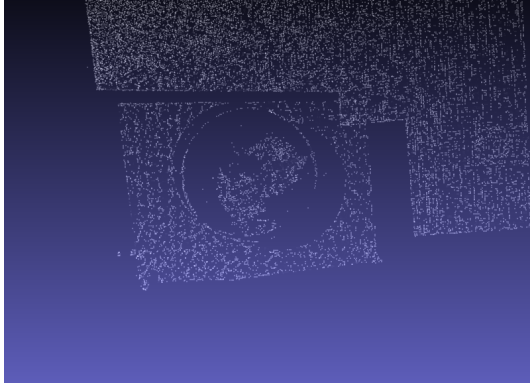


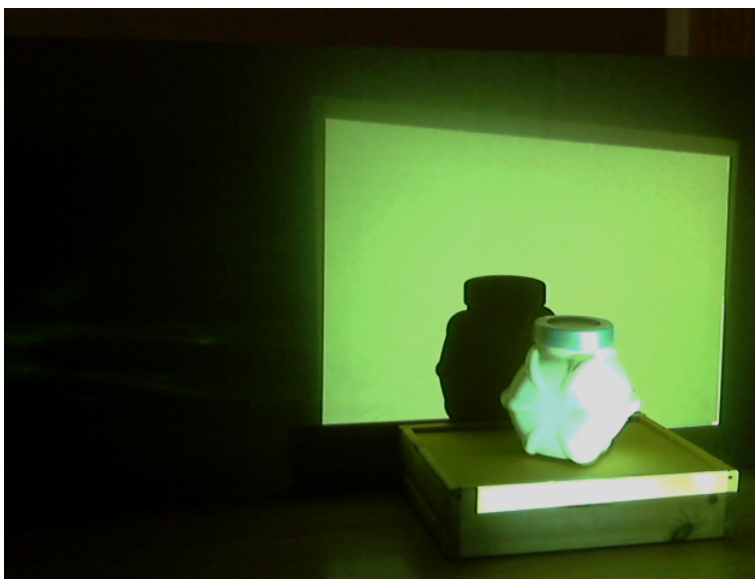
Figure 5.31: Code ensemble, again very poor handling of the inter reflection. Only a very small part of the bowl is reconstructed.

In both orientations, it is clear that our proposed method handles even strong inter reflection very well. Only the parts of the bowl that are consistently specularly lit do not get a depth reconstruction which also affects the parts

directly next to those. The code ensemble on the other hand seems to handle inter reflections very poorly as neither of the high frequency patterns seems to work against them.

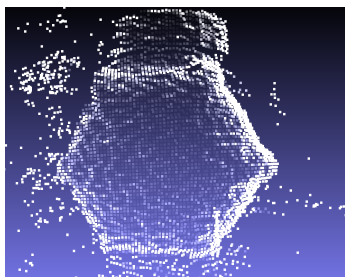
5.4.3 Objects with strong subsurface scattering

In this section we will fill a somewhat interestingly shaped glass object with different concentrations of milk and water and see how the approach handles this compared to the 4 coding schemes used by the code ensemble. For our proposed method, we will again use two simultaneous columns and four simultaneous row scan lines.

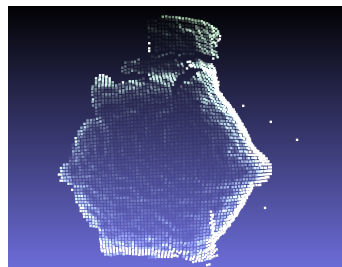


Initially we start with a concentration of 100 % milk and 0% water. Then we water the milk down to 80, 60, 40,20 ,10 and finally 5%*milk*.

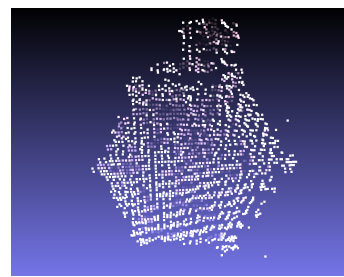
100% milk, 0% water



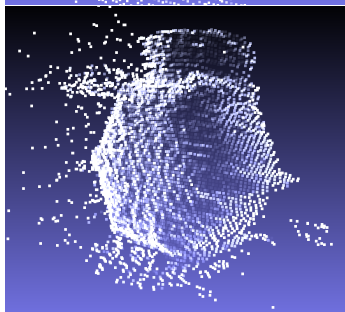
Gray.



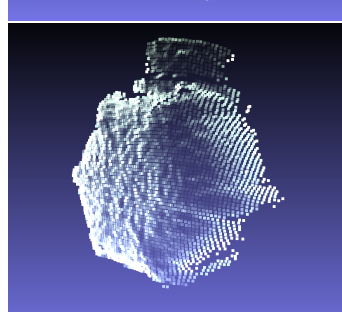
MaxMinWidth.



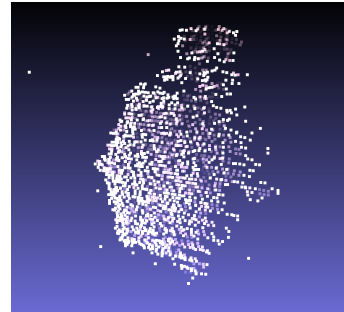
XOR04.



XOR02.



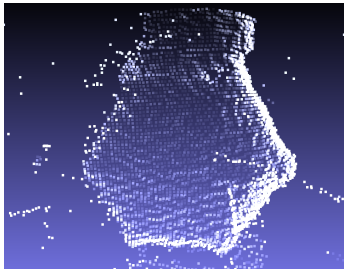
Code ensemble



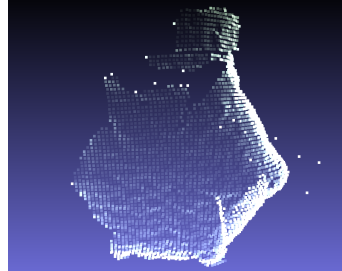
Our method.

The high frequency patterns perform very poorly. The XOR02 pattern has captured nothing but noise and will be omitted for the rest of the results. The Gray encoding and the MaxMinWidth encoding both capture the object well, although the Gray encoding suffers from light that shines through the object onto the wall. The code ensemble while seeming accurate, is very sparse. Our method, with the exception of a specular part on the lid, has captured the object very cleanly and accurately.

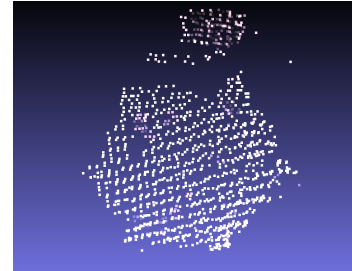
80% milk, 20% water



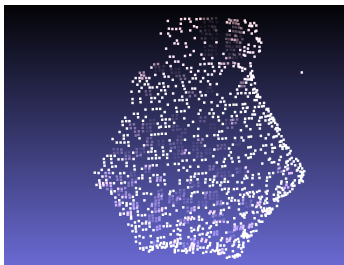
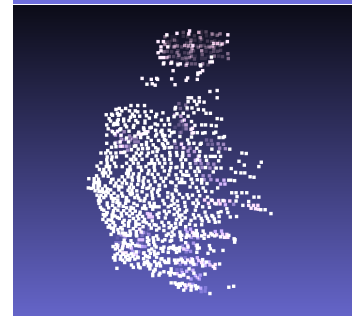
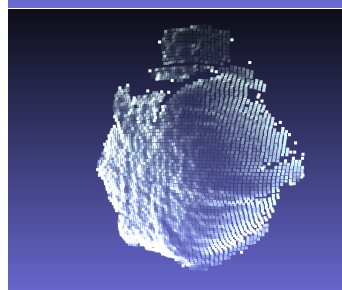
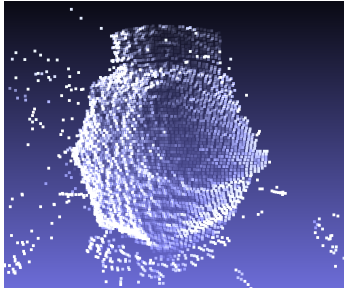
Gray.



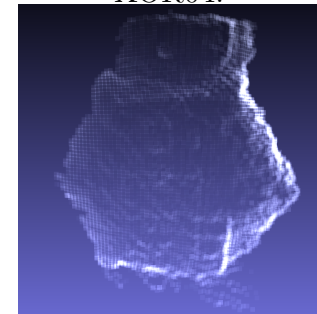
MaxMinWidth.



XOR04.



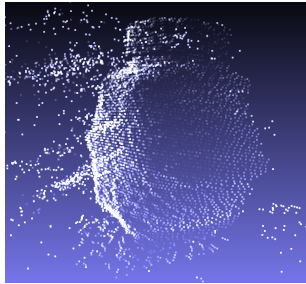
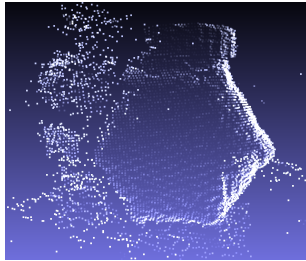
Code ensemble



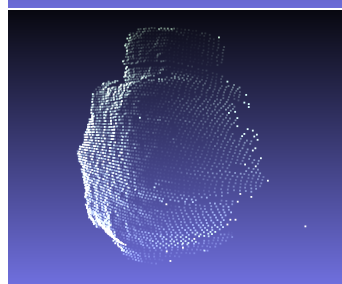
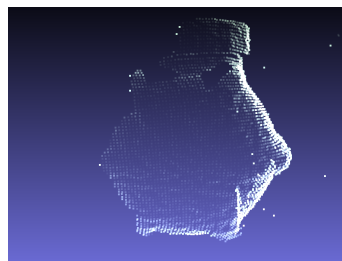
Our method.

Compared to the pure milk, little has changed. The MaxMinWidth starts to show some holes a bit below the lid.

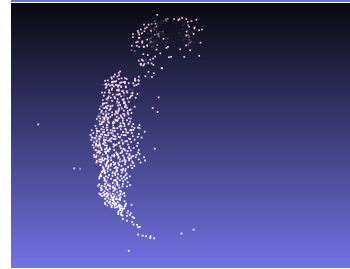
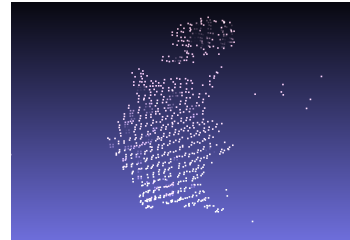
60% milk, 40% water



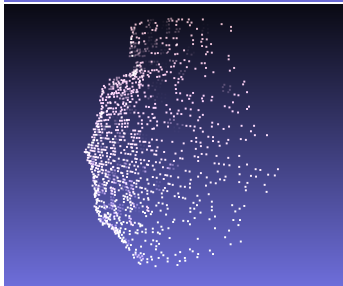
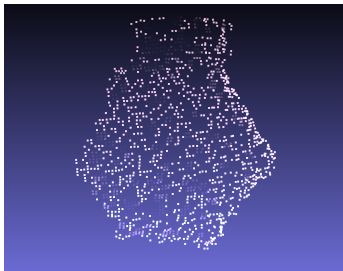
Gray.



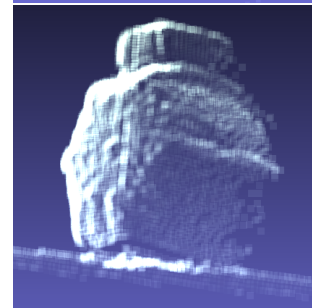
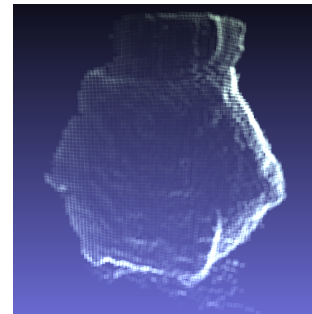
MaxMinWidth.



XOR04.



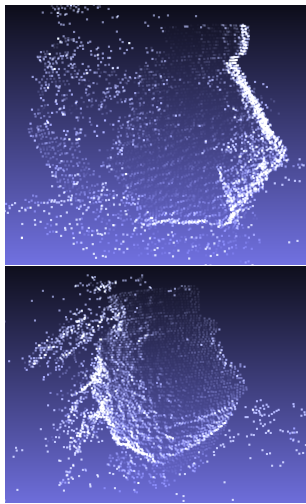
Code ensemble



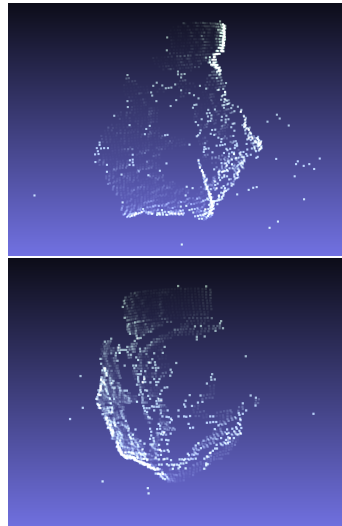
Our method.

As the milk-water solution gets thinner, more light shines through the milk and increases the noise for the Gray encoding. Meanwhile, the XOR04 pattern is getting sparse.

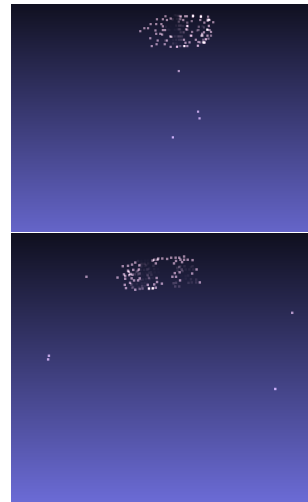
40% milk, 60% water



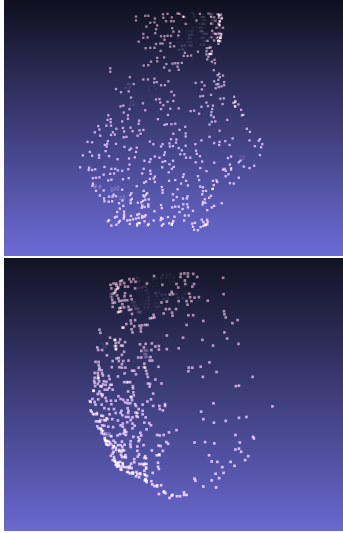
Gray.



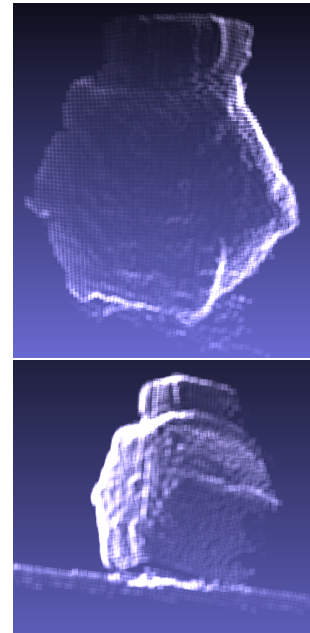
MaxMinWidth.



XOR04.



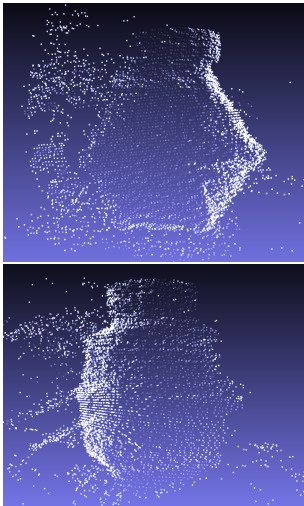
Code ensemble



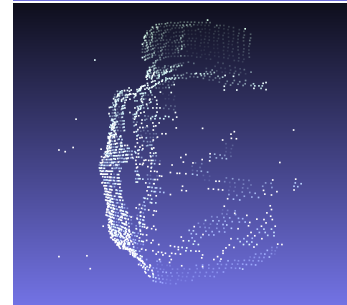
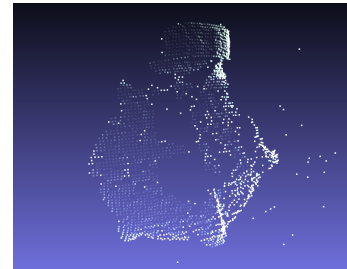
Our method.

The XOR04 pattern is now reduced to only the lid and will be omitted for the rest of the experiments. The Gray encoding is now showing extreme levels of noise. The MaxMinWidth is showing an increasing number of holes. Because the code ensemble can only be as good as its two best coding schemes, it is now closely tied to the MaxMinWidth increasingly worse performance. Meanwhile, our proposed approach appears to be unaffected thus far. Only the specular spot on the lid has a small depth error.

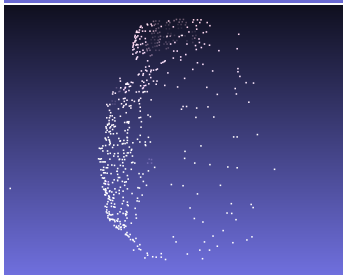
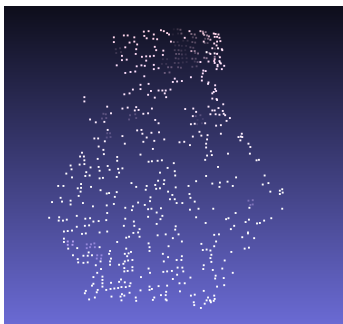
20% milk, 80% water



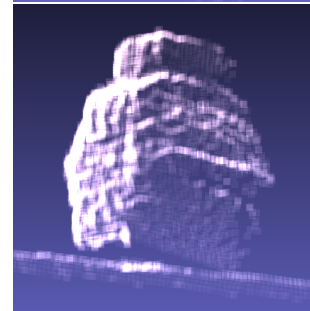
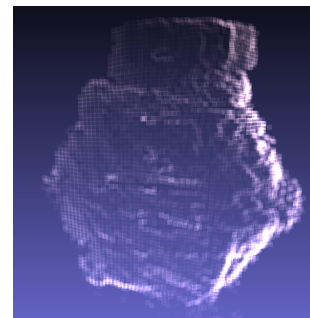
Gray.



MaxMinWidth.



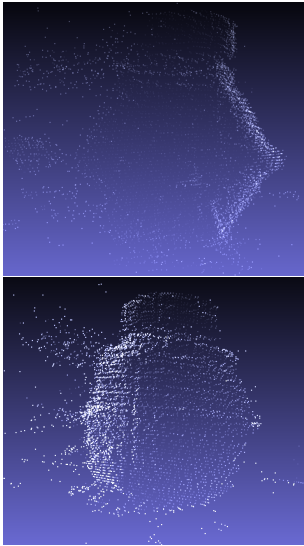
Code ensemble



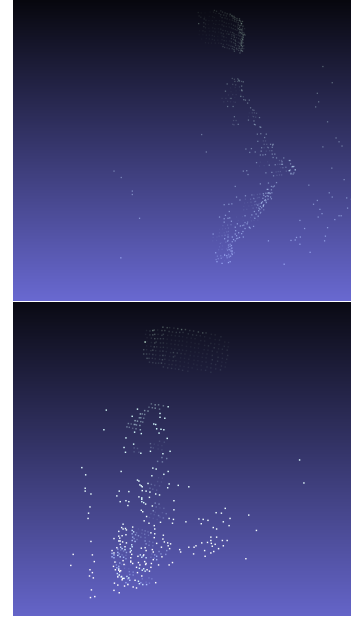
Our method.

Finally, our proposed method is also starting to break down. The front of the object is starting to get rougher and is showing some tears.

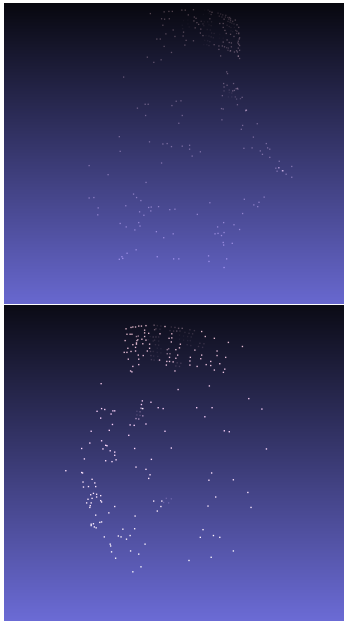
10% milk, 90% water



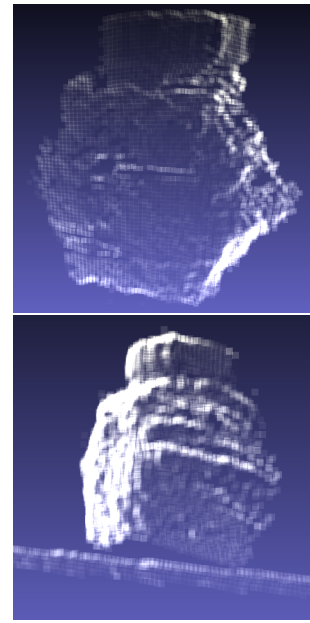
Gray.



MaxMinWidth.



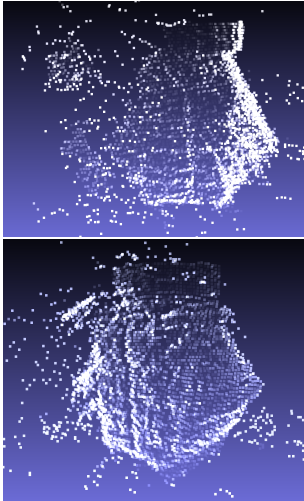
Code ensemble



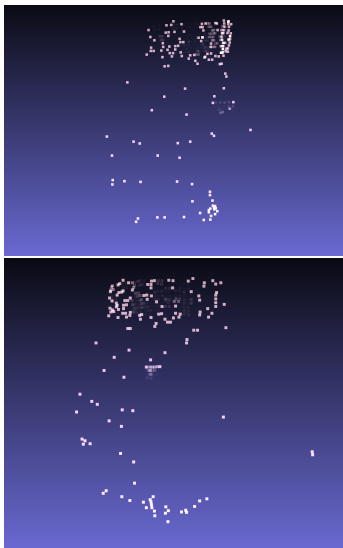
Our method.

At this point the MaxMinWidth pattern reconstructs only a very small portion of the object. The Gray encoding still recovers most of the object, albeit with an incredible amount of noise. The code ensemble's reconstruction is now incredibly thin and can easily be safely called a failed attempt. Our proposed method, while being very rough at the front, still captures the shape of the object quite accurately, without too much noise.

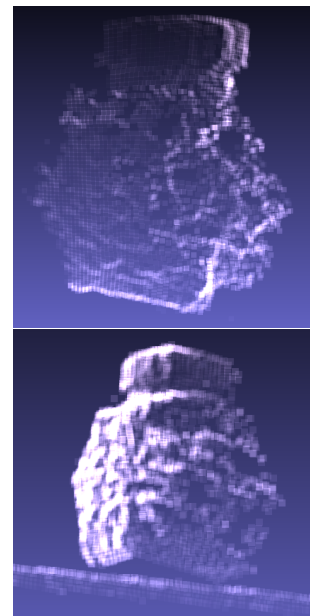
5% milk, 95% water



Gray.



Code ensemble



Our method.

The Gray encoding, while the jar is still somewhat showing through, is entirely covered in noise. The code ensemble, fails to construct anything beyond

the lid. Our proposed method, has cut off a part of the side of the jar.

5.5 Evaluation

Our approach under high sensitivity to light, handles both easy and difficult scenes with inter reflection or subsurface scattering very well. A moderate increase in multiple scan lines affects the accuracy of results only a little, while improving the speed considerably. Inter reflections can often be solved very well, even the parts that are affected most by inter reflections are reconstructed accurately. Subsurface scattering is somewhat more tricky, while the effects are mitigated by the approach, very sharp edges are rounded nonetheless.

Unfortunately due to the high difficulty of achieving a ground truth for most scenes, it is difficult to compare the methods using meaningful numbers. Comparing the number of reconstructed points has little meaning without knowing their accuracy. This is particularly a problem because we want to assess the quality of reconstructions in the presence of strong indirect light. Not being able to provide numbers because of a lack of a ground truth is not an uncommon problem in 3D reconstruction research, which other papers, such as [2] [11] solve by displaying the results, while marking the interesting or incorrect parts.

A visual comparison between the actual real scene, our proposed approach and the code ensemble does show some significant differences. The visual results have been summarized in the following table:

Low indirect light	Proposed	Ensemble	Gray	MaxMinWidth	XOR04	XOR02
Shape	+++	+++	+++	+++	+	-
Accuracy	+++	+++	++	+++	+	+
Density	+++	0	+++	+++	0	--
Noise	+++	+++	+	++	+++	+++
Inter reflection	Proposed	Ensemble	Gray	MaxMinWidth	XOR04	XOR02
Shape	++	-	-	-	--	----
Accuracy	++	++	--	--	++	++
Density	+++	0	+++	+++	--	----
Noise	++	++	-	-	+++	+++
Subsurface scattering	Proposed	Ensemble	Gray	MaxMinWidth	XOR04	XOR02
Shape	++	++	++	+	-	----
Accuracy	++	++	++	++	+	+
Density	+++	0	+++	++	--	----
Noise	+++	+++	--	++	0	0

In this table, for each of the three experiment types, the six different coding schemes have been assigned a score between +++ (completely perfect) and --- (as bad as possible) scored in the following four categories:

1. Shape, how well the model resembles the scanned object. This includes what part of the object is reconstructed out of what was potentially visible.
2. Accuracy, how accurate the parts that are reconstructed are.
3. Density, how dense the reconstruction of the model is.
4. Noise, the levels of noise present in the reconstruction.

The score in one category may affect the score in a different category. If only a small section of an object is reconstructed (low shape score), it will likely have a higher accuracy for the part that it did reconstruct.

The models created by our approach are very dense and accurately reconstruct even tiny details. In situations with a lot of inter reflections, even most of the difficult parts of the scene are reconstructed accurately. Up

until a considerable level of subsurface scattering models are reconstructed accurately.

The code ensemble on the other hand, creates sparse models that often lack tiny details. In situations with a lot of inter reflections most of the difficult parts are omitted rather than constructed. The code ensemble performs reasonable under a considerable level of subsurface scattering.

Interestingly enough, the code ensemble itself requires a decent portion of this evaluation section. The high frequency patterns failed very badly during the experiments. This is potentially because multiple projector columns light the same camera pixel. This makes it possible that the direct light of both the regular and the inversed pattern light a pixel equally, making the decoding process for the pixel impossible. While this is a hardware related problem, it is also a drawback of the method, as it would mean that the projector must be set up to have a resolution as high or lower than the camera, which may be less than the highest potential resolution given the equipment.

Another problem with multiple projector columns hitting the same pixel is that the code ensemble needs two of the four image pattern sets to agree on the decoding. When multiple columns hit the same pixel, different sets may come up with different columns that are both correct for that pixel. However, because the sets do not agree exactly with each other, they will both be rejected. Rather than using an exact match for the column numbers and getting the sparse models that were observed, using a small threshold for whether sets agree would increase the density at the cost of nearly no accuracy.

Finally, while the code ensemble pretends to handle subsurface scattering well, it can never do this better than just the Gray encodings. In cases with subsurface scattering it can be easily assumed (and observed) that both high frequency XOR codes will not give a correct decoding, therefore to get a correct encoding, the Gray and the MaxMinWidth encoding would need to agree. However, if both agree, then this makes the ensemble no better than solely the Gray encoding.

Chapter 6

Conclusion and Future Work

6.1 Conclusions

The experimental results show a clear improved accuracy over the code ensemble by Gupta et al [2] and even individual coding schemes specifically designed to handle inter reflections and subsurface scattering. Both less pixels are decoded incorrectly, and less pixels are incorrectly rejected.

This however does come with a massive increase in acquisition and computation time. One other downside is that because the method chooses a best match largely based on scene continuity, it handles narrow, large depth disparities such as sharp edges poorly. If an object has a depth disparity that is rejected by the method, it can therefore only be fully recovered by combining multiple views of different sides of the object.

While the acquisition time can be reduced by increasing the number of scan lines, for a moderate increase, this results in only a slight reduction of accuracy. For larger number of scan lines, especially for both rows and columns, the decoding time increases significantly due to the increase in viable candidate matches. In addition, with the increase in viable candidates, alternative and incorrect, however more continuous decoding paths might present itself that lead to depth errors.

Using four simultaneous row and two simultaneous column scan lines, good accuracy was achieved. With the total process taking four to seven minutes for most scenes.

The approach reconstruct objects with good accuracy, even if they are

made of difficult materials, at a very low cost. It may therefore fill a niche of low cost 3D scanning with high accuracy, even in difficult situations, as long as computational time matters little.

6.2 Future work

The most important future improvement, would be an improvement in the used calibration method: a lot of the accuracy of handling the difficult problems depends on the depth based rejection between columns and rows. Therefore, the better the calibration is, the lower the depth difference threshold can be and with a lower threshold, less candidates make it to the computational time extensive dynamic programming part of the process. This results in both an increase in accuracy and a reduction in decoding time.

Another improvement would be the way of dealing with subsurface scattering. The used system is too crude and will only mitigate the effects of subsurface scattering. Improving the way of finding the center of consecutive columns hitting the same pixel will improve the sharpness of results. Additionally, if the center can be found beyond an integer column number, then sub pixel accuracy can be achieved.

One other problem is saturated pixels in the case when no light is emitted. These pixels receive the same amount of light whether they are directly lit or not and will therefore cause problems in combination with the cost function, as they appear to be occluded. These saturated pixels are caused because the projector always emits a small amount of light which is specularly reflected strongest where the angle between the projector and the surface normal equals the angle between the surface normal and the camera. Because both the location of the projector and the equation corresponding to each camera pixel is known, a good estimate of the depth and surface normal for these saturated pixels should be possible. This would both increase the number of recovered pixels as well as the accuracy of nearby pixels due to continuity constraints.

Part of this problem is due to the way the implementation makes use of the projector; the projector is used as a second monitor that displays a maximized window using openCV and Windows. However this adds an impossible to hide white border around the projector screen. This border constantly lights part of the scene and may therefore appear as a specular reflection. The border therefore both degenerates results and makes solutions

to genuinely saturated pixels due to the material more difficult.

To reduce the acquisition time further, more simultaneous scan lines could be used, this would however require a stronger method of solving ambiguity. Using some binary codification patterns, like some other phase shifting methods have done [4] [8] is a possibility. However using them directly could cause problems with indirect light effects, in particular inter reflection. Using a median or average depth over a larger section might help handle this however.

Bibliography

- [1] J. Salvi, J. Pags, J. Batlle, Pattern codification strategies in structured light systems, *Pattern Recognition* 37, 2004, 827-849.
- [2] M.Gupta, A. Agrawal, A. Veeraraghavan, S. Narasimhan, A Practical Approach to 3D Scanning in the Presence of Interreflections, 2012, *Subsurface Scattering and Defocus*, Columbia University Academic Commons.
- [3] B. Carrhill, R. Hummel, Experiments with the intensity, ratio depth sensor, in: *Computer Vision, Graphics and Image Processing*, Vol. 32, Academic Press, New York, 1985, pp. 337358.
- [4] J. Guhring, Dense 3-d surface acquisition by structured light using of-the-shelf components, *Videometrics Opt. Meth. 3DShape Meas.* 4309 (2001) 220231.
- [5] J.L. Posdamer, M.D. Altschuler, Surface measurement by space-encoded projected beam systems, *Comput. Graph.Image Process.* 18 (1) (1982) 117.
- [6] S. Inokuchi, K. Sato, F. Matsuda, Range imaging system for 3-D object recognition, in: *Proceedings of the International Conference on Pattern Recognition*, 1984, pp. 806 808.
- [7] D. Caspi, N. Kiryati, J. Shamir, Range imaging with adaptive color structured light, *Pattern Anal. Mach. Intell.* 20 (5), 1998, 470480.
- [8] D. Bergmann, New approach for automatic surface reconstruction with coded light, in: T.F. Schenck (Ed.), *Proceedings of Remote Sensing and Reconstruction for Three-Dimensional Objects and Scenes*, Vol. 2572, SPIE, San Diego, CA, 1995, pp. 29

- [9] K. Khoshelham, S.O Elberink, Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. Sensors 2012, 12, 1437-1454.
- [10] S. K. Nayar, G. Krishnan, M. D. Grossberg, and R. Raskar. Fast separation of direct and global components of a scene using high frequency illumination. ACM Trans. Graph., 25(3), 2006.
- [11] T. Chen,H. Lensch, C. Fuchs,H. Seidel, Polarization and Phase-Shifting for 3D Scanning of Translucent Objects, Computer Vision and Pattern Recognition, 2007. CVPR '07, 1 - 8
- [12] Z. Zhang. A flexible new technique for camera calibration, Pattern Analysis and Machine Intelligence, IEEE Transactions on (Volume:22 , Issue: 11), 2000, pp 1330 - 1334
- [13] G. Falcao, N. Hurtos, J. Massich, D. Fofi, "Projector-Camera Calibration Toolbox", <http://code.google.com/p/procamcalib>, 2009.
- [14] OpenCV Computer Vision library (c++), <http://opencv.org/>
- [15] Douglas Lanman, "Structured Light for 3D scanning", Brown University, <http://mesh.brown.edu/byo3d/source.html>, 2009
- [16] Meshlab, <http://meshlab.sourceforge.net/>