Single-Shot Direct Block Address Encoding for Learning Screen Geometry

by

Sina Farsangi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Masters of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2021

Author's Declaration

Statement of Contributions

I am the first author of the conference papers below and have contributed to the implementation, experimentation, and writing of all the papers. Some of the content presented in Chapters 5 and 6 has been documented in these papers.

1. Farsangi S, Naiel M, Lamm M, Fieguth P, "Rectification Based Single-Shot Structured Light for Accurate and Dense 3D Reconstruction", *6th annual Conference on Vision and Intelligent Systems*, CVIS 2020 (Recipient of Best Paper Award).

2. Farsangi S, Naiel M, Lamm M, Fieguth P, "Efficient Direct Block Address Encoding for Single-Shot based 3D Reconstruction", *Society for Information Display Conference*, SID 2021.

Abstract

3D surface reconstruction has many applications in different domains such as projection mapping, virtual reality, robot navigation, human computer interaction and manufacturing inspection, to name a few. Among different methods of 3D reconstruction, structured light is widely used as it is comparatively cheap and accessible and solves the main problem of traditional stereo vision systems which is finding accurate pixel correspondences between two or multiple views. Structured light techniques can be most fundamentally categorized in terms of the number of projected images over time, whether a single image (single-shot) or multiple images (multi-shot). Multi-shot structured light methods take advantage of multiple images that are projected sequentially over time, allowing simple encoding / decoding of projector pixel addresses. In contrast, single-shot structured light is preferred in contexts of dynamically moving cameras, projectors or surfaces, and in scenarios where short projection time is important.

In this thesis, a new framework for designing single-shot structured light images using tag embedding, called Direct Block Address Encoding, is presented which, unlike previous methods, results in efficient encoding, decoding and 3D reconstruction. Also, error detection and correction mechanisms are designed to detect pixel codewords with errors and find their correspondences in the projector image. In addition, the relationship between different design parameters (alphabet size, encoding Scheme, tag size, block size) are derived to cover projectors with different resolutions.

Experimental results demonstrate that the proposed scheme is capable of obtaining projector-camera pixel correspondences at higher speed in comparison with previous tag embedding methods, allowing for learning screen geometry from a single shot with high resolution projectors and dynamic cameras and projectors.The proposed Direct Block Address Encoding scheme offers 2-3 times speed up for 3D reconstruction and 5-6 times speed up for encoding/decoding stages due to not requiring a look-up table and/or an exhaustive search, something not achieved with other methods.

Acknowledgements

I would like to thank my supervisor Prof. Paul Fieguth for his constant support during my masters studies. He set an incredible example for me as a researcher, teacher and mentor. Thank you for all the guidance and support during my studies.

I would like to thank both Prof. Clausi and Prof. Wang for serving as readers of my Masters thesis.

I would like to thank Dr. Mohamed Naiel for his inputs and suggestions during my research, and his warm support. Thanks for you efforts.

I would like to thank my friends and roommates, Eric, Nani and Bo; and my other friends Hossein, Kasra and Martins for making my time in Waterloo enjoyable. I would also like to thank the members of the Vision and Image Processing Lab.

In addition, I would like to thank Mark Lamm and Christie Digital for providing me an excellent internship opportunity through which I have been able to experience industry oriented research. Finally, I want to thank my parents, my fiancee and my brother for their continuous support and encouragement throughout my years of study.

Dedication

This is dedicated to my loved ones.

Table of Contents

List of Figures

List of Tables

Chapter 1

Motivation

3D surface reconstruction techniques learn 3D geometry of surfaces widely used in a variety of applications such as projection mapping [4], virtual reality [5] and industrial inspection [6], to name a few. A taxonomy of different 3D reconstruction techniques are shown in Figure 1.1 [5]. These techniques can be classified in to passive and active techniques, where in passive methods no active illumination is used for 3D reconstruction. One of the most popular methods in this category is stereo vision [7,8]. As shown in Figure 1.1, in this method, two or more cameras are used to capture images from a target surface from different views. Then, pixel correspondences are obtained between these views and by assuming that the physical properties of the cameras (principal point, focal length, etc) and pose between the cameras are known, 3D reconstruction is performed using a process called triangulation [9]. However, if the target surface does not have a rich surface, the obtained pixel correspondences and 3D reconstructed surface will be inaccurate, therefore, limiting the usage of stereo vision in certain scenarios.

On the other hand, active 3D reconstruction methods use a source of illumination to perform 3D reconstruction. Two of the most widely used methods in this category are Time-Of-Flight (TOF) [10] and structured light methods (Figure 1.1) [3]. TOF methods illuminate the target surface with a laser and collect the reflected light from the target surface. Then, the surface depth is calculated using the delay between the emission and collection of light and speed of light in the environment [10].

Finally, structured light methods [5,6,11] solve the problem of obtaining accurate pixel correspondences in stereo vision by replacing one of the cameras by a projector. A structured light system usually consists of a camera-projector pair. In this method, one or more images, called the structured light images, are projected onto the target surface.

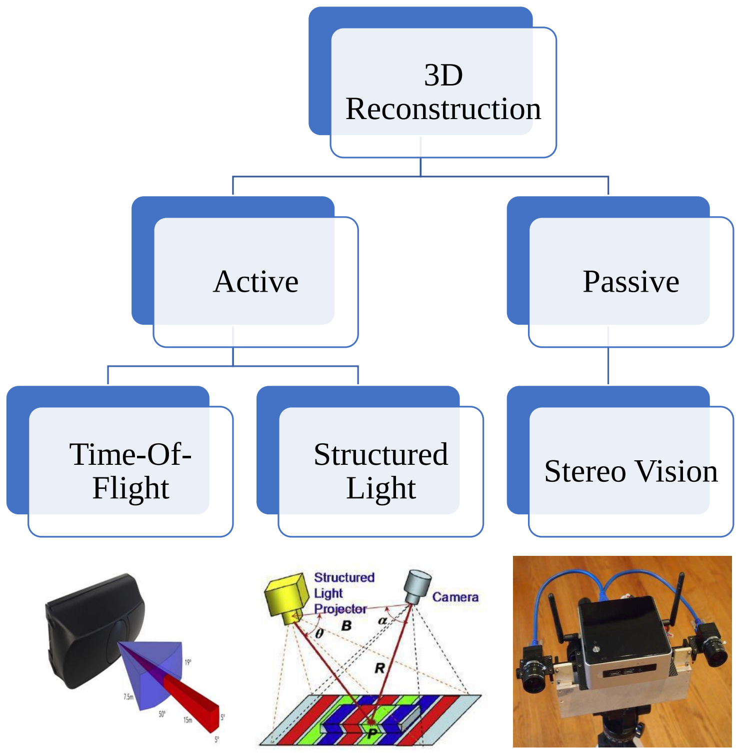


Figure 1.1: Taxonomy of 3D reconstruction methods. Images from [1, 2, 3]

Structured light images are designed such that they encode the location information of the projector pixels (as code-words). After image projection, the camera captures images from the illuminated surface. Then, by decoding the code-words in the camera image, pixel correspondences are obtained between the camera and the projector, therefore solving the problem of finding pixel correspondences for surfaces without rich texture. Finally, assuming the calibration parameters are already obtained, 3D reconstruction is performed using triangulation [5, 9].

The choice of the reconstruction method is dependent on the requirements of each scenario. One of the important applications of 3D reconstruction is projection mapping. Projection mapping is a projection technique, where videos are projected using projectors on irregular objects in a way to make immersive visualizations [12]. Some examples of projection mapping created by Christie Digital Systems are shown in Figure 1.2. In order to create a good visualization, the 3D geometry of the target surface is required. Therefore, using structured light is suitable since a projector is needed for projection mapping and at

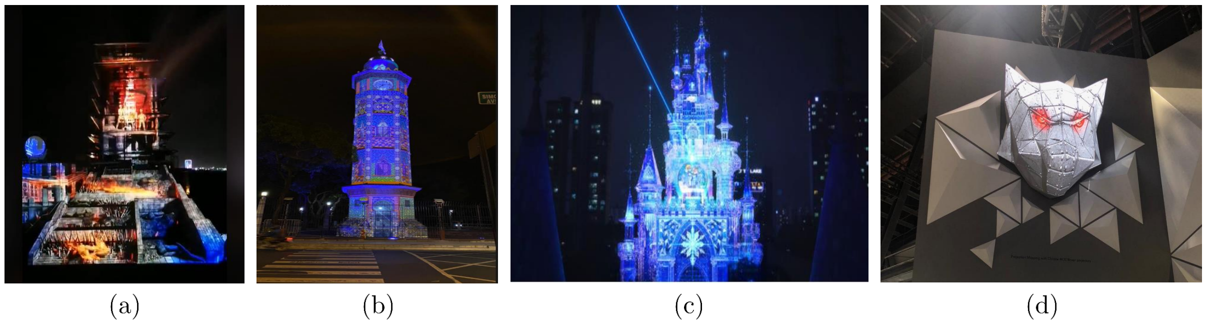


Figure 1.2: Application of projection mapping used by Christie Digital Systems in (a) Lanzhou's Yellow River Tower, China, (b) The Clock Tower, Guayaquil, Ecuador, (c) Lotte World's Magic Castle, South Korea and (d) Wolf-head projection mapping, Christie Digital lab. Images taken from the Christie Digital Linked-in web page [13].

the same time the 3D geometry of the target surface is required. In this thesis, the focus is on the application of 3D reconstruction in projection mapping and therefore, structured light based methods.

As mentioned previously in this chapter, the focus in this thesis is on 3D reconstruction using structured light. Structured light methods can be most fundamentally categorized in terms of the number of projected images over time, whether a single image (single-shot) or multiple images (multi-shot) [5, 6, 11]. Multi-shot structured light methods take advantage of multiple images that are projected sequentially over time, allowing simple encoding/decoding of projector pixel addresses [14, 15, 16, 17, 18]. In contrast, single-shot structured light is preferred in contexts of dynamically moving cameras, projectors or surfaces, and in scenarios where short projection time is important [19,20,21,22,23,24,25,26,27,28,29]. In principle, both techniques can be used for learning 3D screen geometry needed for projection mapping applications. In this thesis, the usage of single-shot structured light for 3D reconstruction is studied. Among different types of single-shot methods, tag embedding methods are widely used where tags with different colors, shapes and/or texture properties are used to represent an encoded structured light image [19, 20, 21, 22, 23, 25, 26, 27, 28, 29]. The objective of this thesis is to present a new framework for single-shot 3D reconstruction using tag embedding which offers a high robustness to errors in tag classification as long as efficient encoding (structured light image construction) and decoding (obtaining camera-projector pixel correspondences) in comparison with previously proposed methods making it suitable for 3D reconstruction using high resolution projectors.

The remainder of this thesis is structured as follows: First, in chapter 2, a review is presented on structured light techniques. Then, in chapter 3, the thesis problem formulation is presented. Chapter 4 discusses tag design for robust classification and chapters 5 and 6 present direct block address encoding, a new efficient method to preform 3D reconstruction using single-shot structured light.

Chapter 2

Structured Light Literature Review

As mentioned in Chapter 1, the focus in this thesis is on 3D reconstruction using structured light. First, in Section 2.1, it is explained what information is needed to perform 3D reconstruction using a structured light system. Then, in Sections 2.2 and 2.3, a summary on important structured light methods in the literature is presented.

2.1 3D reconstruction using structured light

Stereo vision [7, 8] and structured light [5, 6, 11] perform 3D reconstruction through the same process of triangulation [5, 9]. In order to understand how triangulation works, it is necessary to have an understanding of multi-view geometry. First, the process of performing triangulation for stereo vision is explained. Then, the same concept is easily extendable to a structured light system.

In stereo vision [7, 8], multiple cameras are used which capture images from different views of the same scene as shown in Figure 2.1. For the sake of simplicity, the stereo vision setup in Figure 2.1 is constrained to a camera pair. Given a pair of cameras C_1 and C_2, the goal is to reconstruct the 3D geometry of a target surface in the scene, in other words, the 3D coordinates of points on the surface. Assuming a point on the target surface with coordinates $p = (x, y, z)$ in the 3D world coordinate system where the origin of the world coordinate system is denoted by $O = (0, 0, 0)$ (usually the projection center, where all light rays enter the camera, of one of the cameras). The goal of triangulation is to find p, the location of the given point in the 3D world coordinate system. As shown in Figure 2.1, each 3D point like p has a projection on the image planes of cameras C_1 and C_2, which their

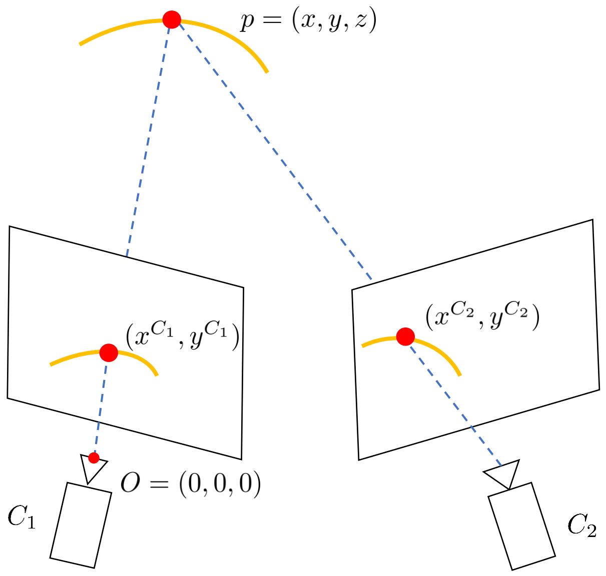


Figure 2.1: An example of a stereo vision setup where two cameras, C_1 and C_2 view the same curved surface (orange) from two different angles.

coordinates in C_1 and C_2 image planes are shown by (x^{C_1}, y^{C_1}) and (x^{C_2}, y^{C_2}), respectively. Any pair of pixel coordinates in C_1 and C_2 image planes like (x^{C_1}, y^{C_1}) and (x^{C_2}, y^{C_2}) that correspond to the same 3D point is called a pixel correspondence between C_1 and C_2.

Now, assuming that N pixel correspondences are obtained between C_1 and C_2, the set, denoted by $\mathcal{C}$, can be shown as:

$$\mathcal{C} = [(x_1^{C_1}, y_1^{C_1}, x_1^{C_2}, y_1^{C_2}), (x_2^{C_1}, y_2^{C_1}, x_2^{C_2}, y_2^{C_2}), ..., (x_N^{C_1}, y_N^{C_1}, x_N^{C_2}, y_N^{C_2})] \tag{2.1}$$

where $C_k = (x_k^{C_1}, y_k^{C_1}, x_k^{C_2}, y_k^{C_2})$ denotes the coordinates of the kth pixel correspondence between C_1 and C_2 $(k \in 1, 2, ..., N)$.

For each camera, a calibration matrix [9], $K_{3\times3}$, of size 3×3 is defined which includes parameters that are related to the physical properties of each camera, such as the principle point, the focal length, aspect ratio and lens distortion parameters. As these parameters are related to the physics of the camera, they are called the intrinsic parameters of the camera. In contrast, the pose between cameras C_1 and C_2 give rise to the extrinsic parameters which are defined by the rotation matrix, $R_{4\times4}$, and the translation matrix, $\mathcal{T}_{3\times1}$, between C_1 and C_2 [9]. The process of obtaining the intrinsic parameters of each camera and the extrinsic parameters between a camera pair is called camera pair calibration [9, 30], and a camera

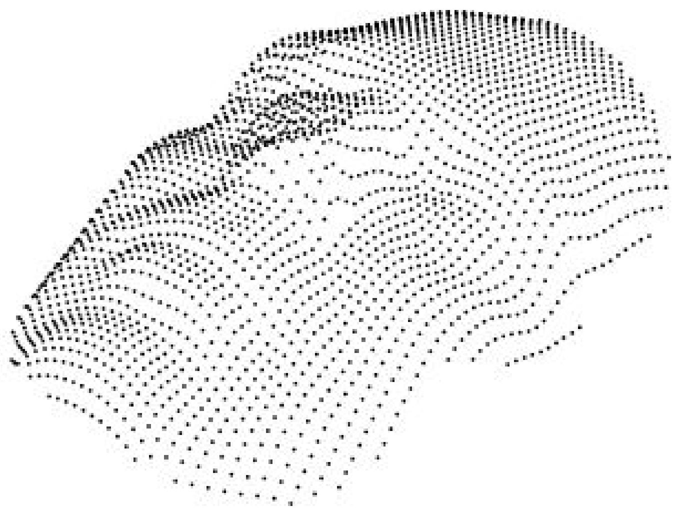


Figure 2.2: An example of a 3D face represented as a point cloud [31].

pair for which their intrinsic and extrinsic parameters are known is called a calibrated camera pair.

Based on triangulation [9], for the kth pair pixel correspondence between C_1 and C_2, the 3D coordinates of the point that $\mathcal{C}_k$ corresponds to, $p_k = (x_k, y_k, z_k)$, can be obtained using the camera calibration matrices, K_1 and K_2, and the rotation matrix, R, and translation matrix, $\mathcal{T}$, between C_1 and C_2. This process can be shown as:

$$p_k = \tau(K_1, K_2, R, \mathcal{T}, \mathcal{C}_k) \tag{2.2}$$

where τ is the triangulation operator. By applying τ on all of pixel correspondences in $\mathcal{C}$, a set of 3D coordinates, $\mathcal{P}$, can be obtained:

$$\mathcal{P} = [(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_N, y_N, z_N)] = \tau(K_1, K_2, R, \mathcal{T}, \mathcal{C}) \tag{2.3}$$

Therefore, the output of triangulation is a set of 3D coordinates of the target surface denoted by $\mathcal{P}$. This set is also referred as the point cloud of the reconstructed surface which is a good tool for visualizing the reconstruction result. An example of a 3D reconstructed point cloud is shown in Figure 2.2.

In this section, the exact mathematical formulation of triangulation is not presented since it is not the main focus in this thesis. One can find more details on triangulation in [9].

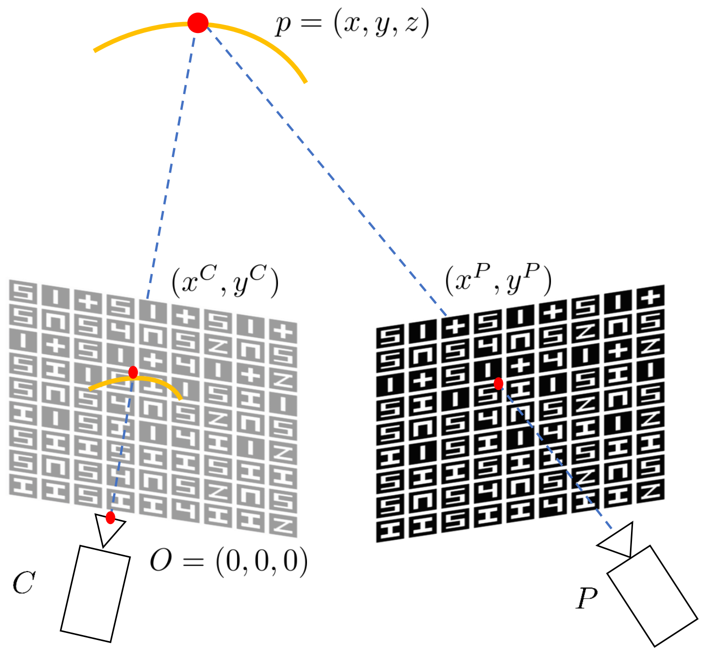


Figure 2.3: An example of a structured light setup where one camera, C, and a projector,P, are used to reconstruct the target surface (orange). The projector projects one or multiple structured light images and the camera captures images from the projected scene. The correspondences are obtained by decoding the pixels from the camera and matching them to the pixels in the projector space with the same code-word (in this case, the same neighborhood of symbols).

The conclusion of Equation 2.3 is that by having the intrinsic and extrinsic parameters of C_1 and C_2 and the set of pixel correspondences in the image planes of C_1 and C_2, 3D reconstruction can be performed using triangulation.

One of the main drawbacks of using stereo vision systems is that finding pixel correspondences between the cameras is not easy and subject to error if the target surface is texture-less which is quite common [5]. In order to solve this problem, in structured light systems [5,6,11], as shown in Figure 2.3, one of the cameras in Figure 2.1 is replaced by a projector, P. In this new setup, the projector projects images (also called structured light images) onto the target surface and the camera captures images from the projected scene. If multiple images are projected onto the target surface, it is called multi-shot structured light [14,15,16,17,18] and if a single image is projected, it is called single-shot structured light [19,20,21,22,23,24,25,26,27,28,29]. The structured light images are designed such that information of projector pixel locations, (x^P, y^P), are uniquely encoded (are given code-words) in the images. As shown in Figure 2.3, the pixel in the structured light image at location (x^P, y^P) is projected onto the point p in the 3D world coordinate system. At the same time, the camera projects this point to its image plane at location (x^C, y^C) by capturing an image from the projected scene. (x^P, y^P) and (x^C, y^C) correspond to the same 3D point, therefore, this pair is called a camera-projector pixel correspondence.

After structured light image construction, the camera captures an image or a set of images of the projected scene. Then, each pixel in the camera image is decoded (their code-words are found) and, by matching them to the pixel in the projector space with the same code-word, pixel correspondences between the projector and the camera can be obtained. Assuming N pair of pixel correspondences are obtained between C and P, this set of camera-projector pixel correspondences can be written similarly to Equation 2.1 as

$$\mathcal{C} = [(x_1^C, y_1^C, x_1^P, y_1^P), (x_2^C, y_2^C, x_2^P, y_2^P), ..., (x_N^C, y_N^C, x_N^P, y_N^P)] \tag{2.4}$$

Also, the intrinsic parameters for a camera-projector pair and the extrinsic parameters of this pair can be defined. The process of obtaining these parameters for any camera-projector pair is called camera-projector calibration [18, 32, 33, 34]. Based on triangulation and similar to Equation 2.3, 3D surface reconstruction can be performed using camera-projector pixel correspondences and intrinsic and extrinsic parameters of C and P as

$$\mathcal{P} = [(x_1, y_1, z_1), (x_1, y_1, z_1), ..., (x_N, y_N, z_N)] = \tau(K_C, K_P, R, \mathcal{T}, \mathcal{C}) \tag{2.5}$$

where K_C is the calibration matrix for the camera, K_P is the projector calibration matrix and the rest of the parameters are the same as defined in Equation 2.3.

Based on Equation 2.5, in order to perform 3D reconstruction using structured light, two types of information are needed: The camera-projector calibration parameters including the intrinsics and the extrinsics, and the pixel correspondences between the projector and the camera. Several methods in the literature have been proposed to perform camera-projector calibration. In this thesis, it is assumed that calibration is done beforehand using one such calibration method [18, 32, 33, 34]. By having the calibration parameters, the problem of performing 3D reconstruction using triangulation from Equation 2.5 boils down to finding the camera-projector pixel correspondences. The process of obtaining pixel correspondences using structured light is highly dependent on the method that is used to encode projector pixel locations into structured light images. Therefore, structured light image construction is of high importance and different methods have been proposed in the literature based on their approach to encode projector pixel locations and obtain camera-projector pixel correspondences [14, 15, 16, 17, 18, 21, 22, 23, 24, 25, 26, 27, 28, 29]. Structured light methods can be most fundamentally categorized in terms of the number of projected images over time, whether multiple images (multi-shot) [14, 15, 16, 17, 18] or a single image (single-shot) [21, 22, 23, 24, 25, 26, 27, 28, 29] is projected. In the rest of this chapter, a summary of each of these methods is presented.

2.2 Multi-shot structured light

Multi-shot methods [14, 15, 16, 17, 18], also known as temporal coding techniques, find the pixel correspondences between the projector and the camera by sequentially projecting multiple images over time on the target surface. These methods have the advantage of reaching pixel level accuracy and dense 3D reconstruction, meaning that all of the pixel locations in the projector space can be encoded uniquely using the sequence of structured light images. However, these methods usually require a long projection time and are not applicable to scenarios where moving cameras, projectors and surfaces are used. Different multi-shot methods have been proposed based on how projector pixels are encoded. Some of the most widely used multi-shot methods are based on simple binary coding [14] and Gray coding [15]. In the following, a summary of each of these methods is presented.

2.2.1 Simple binary coded structured light

Given a projector with a resolution of $N_1 \times N_2$ pixels, in a simple binary coded structured light method [14], the column and rows of each pixel location in the projector space is encoded using simple binary coding where a set of N_c structured light images are used to encode the column index of pixels and a set of N_r structured light images are used to encode the row indices. For the sake of simplicity, the process of creating binary coded structured light images for encoding column addresses is explained first. Then, the same process is extendable to the rows.

In order to encode the projector column indices, each column address in the projector space is encoded by converting the column addresses to their binary representations with N_c digits. Then, the qth structured light image carries the value of the qth ($q \in 1, 2, ..., N_c$) digit of the binary representation of each projector column index. In other words, each column in the qth structured light image is white if the qth digit of the column index binary representation is 1 and it is black if the qth digit of its index binary representation is 0.

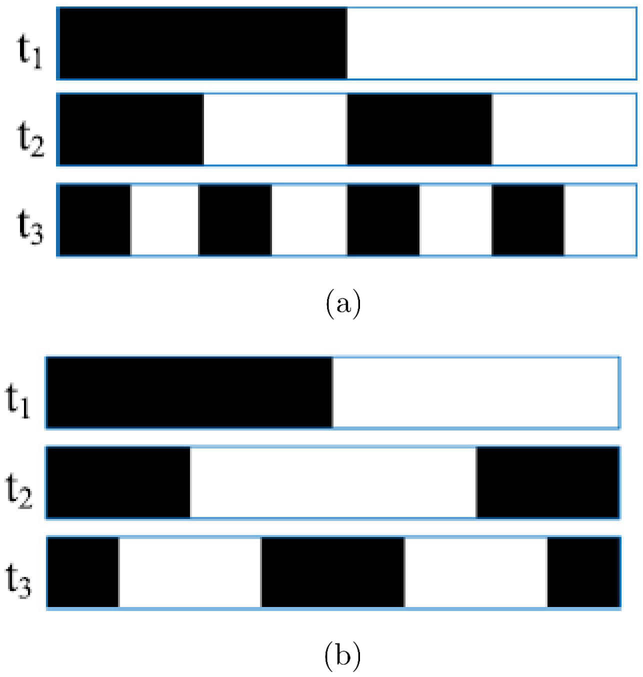


Figure 2.4: Illustration of (a) simple binary coded structured light, (b) Gray coded structured light. In this case $N_2 = 8$ projector columns are encoded using $N_c = \lceil \log_2 N_2 \rceil = 3$ structured light images that are projected sequentially at times t_1, t_2 and t_3.

The number of structured light images needed to encode N_2 column indices in a binary fashion is $N_c = \lceil \log_2 N_2 \rceil$.

Similarly, the row indices of the projector can be encoded using a sequence of $N_r = \lceil \log_2 N_1 \rceil$ structured light images in the same way. After creating the structured light images, first, the set of N_c images that encode the column indices are projected sequentially onto the target surface and camera images are captured from the projected scene. Then, in the decoding stage, each pixel in each of the captured camera images from the projected scene is classified as either black (0) or white (1), meaning that either it was projected by a black or white projector pixel. This helps to obtain the code-word for each pixel location in the camera and to find its corresponding column in the projector space. By performing the same process for the rows, the pixel correspondences between the camera and the projector can be obtained. An example of this method shown in Figure 2.4 (a).

2.2.2 Gray coded structured light

A common problem while using simple binary coded structured light are errors that may occur due to problems in camera sampling. In other words, the camera may not be able to capture the transition in pixel intensity from black to white or vice-versa, causing errors in obtaining the correct code-words for each projector row or column. In order to decrease this error, the authors in [15, 18, 35], take inspiration of the well-known Gray coding in communications theory [36], and use Gray codes instead of simple binary codes to encode the projector column and row indices. An example of this is shown in Figure 2.4 (b). The main advantage of Gray code is that each code-word is different from its previous and next code-word only in one digit. Therefore, if there are errors due to camera-sampling, the error in decoding will be constrained to one column or row which is not necessarily the case for simple binary coded structured light [14].

2.3 Single-shot Structured light

Unlike multi-shot structured light methods [14,15,16,17,18], single-shot methods [21,23,24,26,27,29,37,38] use a single structured light image in order to perform 3D reconstruction. These methods have the advantage of having a short projection time, making it suitable for scenarios where the target surface or the camera is moving. However, these methods [21,23,24,26,27,29,37,38] usually encode a sparse set of pixel locations in the projector space, resulting in sparser point clouds in comparison with multi-shot methods [14,15,16,17,18].

The location information of projector pixels can be encoded in different ways in one structured light image. Based on this, different single-shot structured light methods have been proposed in the literature [21,23,24,26,27,29,37,38] of which a summary of the most important methods is presented in the following section.

2.3.1 Single-shot structured light using tag embedding

Single-shot structured light methods in [21,23,26,27,29,37,38] use a set of usually square tags which have different colors or shapes that are embedded in the structured light image, so that each tag and the tags in its neighborhood create a pattern (code-word) that is unique in the entire structured light image. In particular, these methods have utilized pseudo-random arrays for structured light image encoding [21, 23, 26, 27, 29, 37, 38]. A pseudo-random array with an alphabet size of K, is an array which its element values can

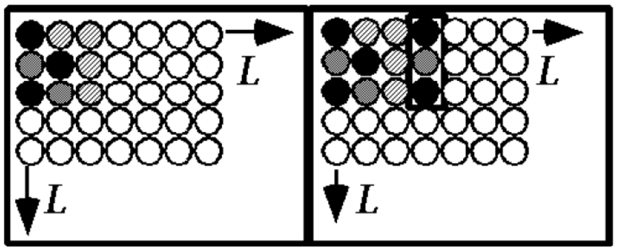
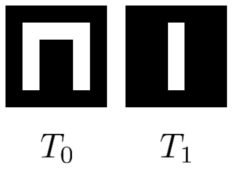


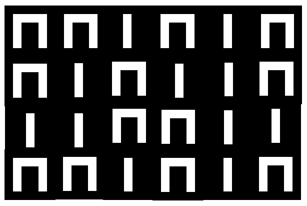
Figure 2.5: Example of creating a pseudo-random array using the brute force algorithm in [21] where an alphabet size of $K = 3$ and block property of 3×3 is used. The matrix elements are represented here by black dot, gray dot and hatched dot. Image from [21]

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

(a)



(b)



(c)

Figure 2.6: Structured light image construction using pseudo-random arrays. (a) An example of a pseudo-random array with an alphabet size of 2 and window property of 2×2. (b) The tags used to create the structured light image. (c) The structured light image based on the pseudo-random array of (a).

be between 0 to $K - 1$ and the array has a window property of $w_1 \times w_2$, meaning that all overlapping blocks of size $w_1 \times w_2$ are unique in the entire array. Pseudo-random arrays can be created using a brute force algorithm to ensure the uniqueness of overlapping blocks [21]. In order to give an understanding of this algorithm, an example is shown in Figure 2.5. Assume that it is desired to create a pseudo-random array of size $\bar{N}_1 \times \bar{N}_2 = 5 \times 7$ with an alphabet size of $K = 3$ and window property of $w \times w = 3 \times 3$. The algorithm starts by creating an array of size $\bar{N}_2 \times \bar{N}_1$ with empty elements which are shown by white dots. Then, the top left of the array is filled with a random 3×3 window using the alphabet shown by black dot, grey dot and hatched dots. Then, as shown in the top right section of Figure 2.5, the next elements of the array are filled randomly. When a new 3×3 block is created, the new block is compared with the previously created blocks to ensure its

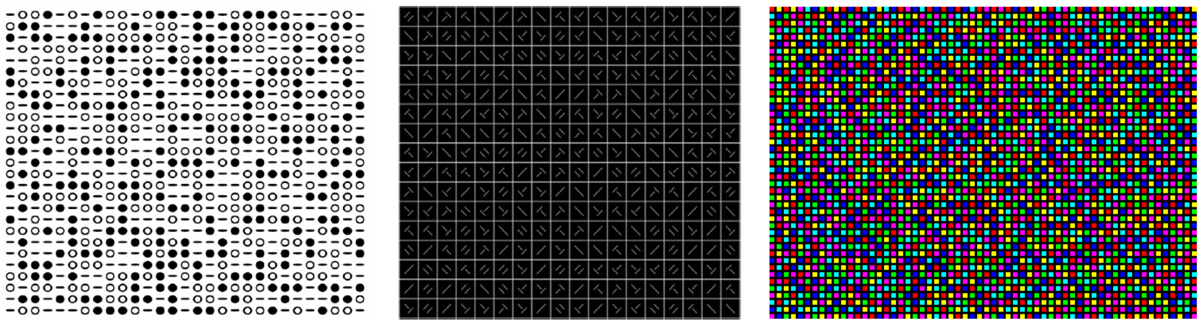

(a) (b) (c)

Figure 2.7: Examples of structured light images created by utilizing pseudo-random arrays. (a) A structured light image from [23] where an alphabet size of $K = 3$ binary tags and a block property of 3×3 is used. (b) Structured light image [27] where an alphabet size of $K = 8$ binary tags and a block property of 2×2 is used. (c) Structured light image of in [29] where an alphabet size of $K = 6$ colored tags and a block property of 3×3 is used. Zoom in for finer details.

uniqueness in the entire array. This is called the uniqueness test. If the block is unique, then the algorithm proceeds with filling the next column. The algorithm terminates when the creation of the pseudo-random array with the desired size is completed.

After creating a pseudo-random array with an alphabet size of K and a particular window property, the structured light image is created using K different types of 2D tags, which differ in their color or shape, by embedding the tags in the same order of the pseudo-random array elements in the structured light image. An example of a pseudo-random array with an alphabet size of $K = 2$ and a window property of $w_1 \times w_2 = 2 \times 2$ is shown in Figure 2.6 (a). After creating this pseudo-random array, two tags, T_0 and T_1, as shown in Figure 2.6 (b) are used to create the structured light image in Figure 2.6 (c).

After structured light image construction, the structured light image is projected onto the target surface. Then, a camera image is captured from the projected scene. In the decoding stage (obtaining pixel correspondences), the unique blocks of $w_1 \times w_2$ tags are detected in the camera image and matched to a block of tags in the structured light image by performing a search in the structured light image. After matching blocks in the camera and projector, different feature points can be extracted from the tags in the detected blocks, both in the camera and projector image, in order to obtain camera-projector pixel correspondences. For example, the center pixels of a pair of corresponding tags in the camera and the projector can be chosen as camera-projector pixel correspondences.

Several papers exist in the literature that use pseudo-random arrays in order to create single-shot structured light images and perform 3D reconstruction [21,23,26,27,29,37,38]. For example, different papers have used different pseudo-random arrays with different sizes and different alphabet sizes which vary in their shapes and colors. Some of these structured light images that are created using pseudo-random arrays are shown in Figure 2.7. In terms of the extracted feature points, the methods in [21,23], use the center pixels of tags as feature points for obtaining pixel correspondences, the authors in [27] use white grids between the tags and design a grid detector which can detect the grid intersection in the camera image and use them as feature points. Also [29] designed detectors to detect the corners of the tags and use them as feature points. In summary, single-shot structured light images based on tag embedding are widely used since they are robust to noise and illumination conditions in the scene when reconstructing comparatively smooth surfaces.

2.4 Conclusion

In this chapter, a review on structured light was presented. More specifically, the requirements for performing 3D reconstruction using structured light was explained. A summary of different structured light methods including multi-shot and single-shot methods was presented. In the next chapter, the problem formulation for this thesis is presented.

Chapter 3

Problem Formulation

At a high level, this thesis aims to present an efficient (with respect to memory and run time) and robust 3D reconstruction (with respect to tag classification) technique based on single-shot structured light that can be scaled-up to high resolution projectors and multi-projector settings. As expressed by Equation 2.5, using triangulation [9], the set of 3D coordinates of the target surface which is presented as a point cloud, $\mathcal{P}$, can be obtained using camera-projector calibration parameters and camera-projector pixel correspondences. By the assumption that the calibration parameters of the projector and the camera (K_p, K_c) and the extrinsic parameters $(R, \mathcal{T})$ are already obtained using one of the methods in [18, 32, 33, 34], the problem of performing 3D reconstruction using structured light boils down to finding the set of pixel correspondences between the projector and the camera, $\mathcal{C}$.

As discussed in Section 2.3, single-shot structured light methods based on tag embedding (Section 2.3.1) [21, 23, 26, 27, 29, 37, 38] are of high interest since they are comparatively robust to noise and illumination conditions in the scene and have a short projection time. Therefore, the single-shot structured light method used in this thesis is restricted to methods based on tag embedding. In this chapter, a detailed overview of tag embedding methods [21, 23, 26, 27, 29, 37, 38] and related notations are presented. Then, the areas of focus in this thesis are discussed that will be the topics in the subsequent chapters.

3.1 Overview of tag embedding methods

A detailed overview for the process of performing 3D reconstruction using tag embedding methods [21, 23, 26, 27, 29, 37, 38] is shown in Figure 3.1. These methods start by

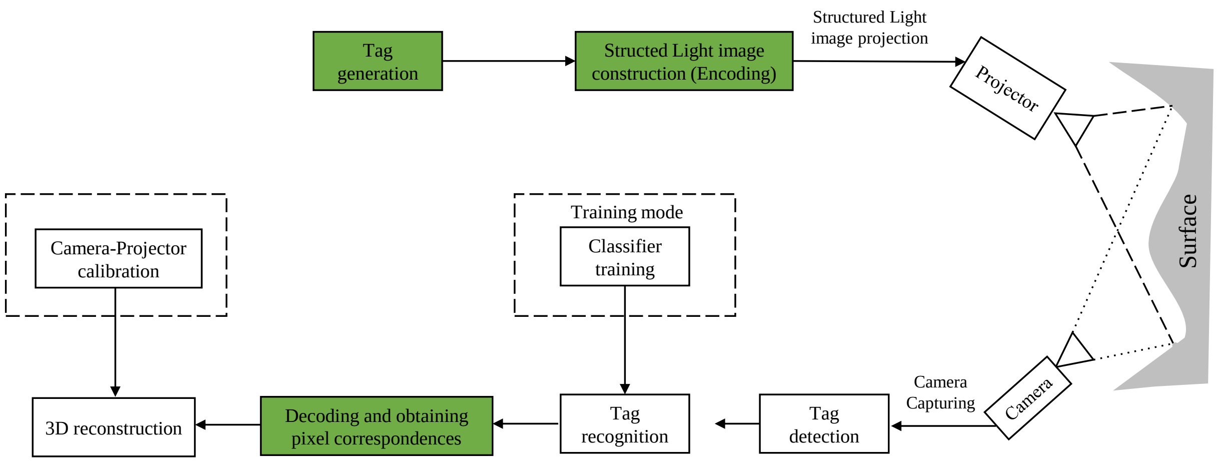

Tag generation
Structed Light image construction (Encoding)
Structured Light image projection
Projector
Surface
Camera
Camera Capturing
Tag detection
Tag recognition
Training mode
Classifier training
Camera-Projector calibration
Decoding and obtaining pixel correspondences
3D reconstruction

Figure 3.1: An overview block diagram of single-shot structured light using tag embedding. The areas of focus in this thesis are highlighted in green. Tag design is discussed in Chapter 4. Structured light image construction and obtaining pixel correspondences is the topic of Chapters 5 and 6.

generating a set of K different tags, also called the alphabet. Each tag, denoted by T_z ($z \in 0, 1, ..., K-1$), corresponds to its label, z and vice versa. Generally, these tags can be binary (white and black) or have different colors as previously shown in Figure 2.7. However, non-binary tags may not be robust to surface color and may cause problems in tag detection and classification stages. Therefore, in this thesis, binary tags are used in the process of single-shot structured light. After creating the tags, T_z, the structured light image, I_P, should be constructed. As mentioned in Section 2.3.1, the tags in the alphabet should be embedded in the structured light image in a particular order so that each tag and the tags in its neighborhood create unique patterns (code-word) that can eventually help to find camera-projector pixel correspondences.

Assuming the structured light image, I_P, has a resolution of $N_1 \times N_2$ pixels which is the same as the projector resolution, the structured light image is partitioned into $\bar{N}_1 \times \bar{N}_2$ cells as

$$\text{Structured light image}: \quad I_P = \begin{bmatrix} c_{0,0} & \cdots & c_{0,\bar{N}_2-1} \\ \vdots & \ddots & \vdots \\ c_{\bar{N}_1-1,0} & \cdots & c_{\bar{N}_1-1,\bar{N}_2-1} \end{bmatrix} \tag{3.1}$$

where $c_{i,j}$ is the (ij)th cell. Then, to complete the structured light image construction, the tags in the alphabet, T, should be embedded in the cells. In order to determine which tags from the alphabet should be embedded in each cell of I_P, an array of size $\bar{N}_1 \times \bar{N}_2$ is defined as

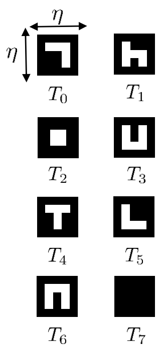
$$\text{Tag label array}: \quad \bar{I}_P = \begin{bmatrix} \bar{l}_{0,0} & \cdots & \bar{l}_{0,\bar{N}_2-1} \\ \vdots & \ddots & \vdots \\ \bar{l}_{\bar{N}_1-1,0} & \cdots & \bar{l}_{\bar{N}_1-1,\bar{N}_2-1} \end{bmatrix} \tag{3.2}$$

where $\bar{I}_P$ is the tag label array of size $\bar{N}_1 \times \bar{N}_2$ and $\bar{l}_{i,j}$ denotes the label corresponding to the embedded tag in $c_{i,j}$ from Equation 3.1. The tag label array determines the order in which different types of tags are embedded in the structured light image, I_P. The tag label array should be designed in a way that each tag and tags in its neighborhood create a pattern that it is unique in the entire array. For example, the methods in [21, 23, 26, 27, 29, 37, 38] use the concept of pseudo-random arrays and uniqueness of sliding blocks (Section 2.3.1) to create the tag label array. It is important to note that any method which ensures uniqueness of patterns created by tag labels in $\bar{I}_P$, can be used for structured light image construction and this is not restricted to pseudo-random arrays.

After creating $\bar{I}_P$ and I_P, the structured light image is projected onto the target surface and a camera image, I_C, is captured. The next step is tag detection. This is usually done using a thresholding method and connected components analysis [27, 39]. The output of this stage is a set of bounding boxes around each tag in I_C. After detecting the tags in the camera image, the tags are classified using a classifier. In other words, each detected tag in the camera image is assigned a label. The classifier is usually trained offline using a machine learning algorithm [27, 40] and before starting the process of 3D reconstruction. After tag classification, for each tag in the camera image and its neighborhood, a unique code-word is identified and matched to the unique code-word in the structured light image to find camera-projector pixel correspondences. Also, it is assumed that the camera and the projector are calibrated offline, using one of the methods in [18, 32, 33, 34]. Finally, 3D reconstruction can be performed using the obtained pixel correspondences and the camera-projector calibration parameters. An example of the explained pipeline is shown in Figure 3.2 where a pseudo-random array is utilized to create the structured light image and perform 3D reconstruction.

3.2 Thesis direction

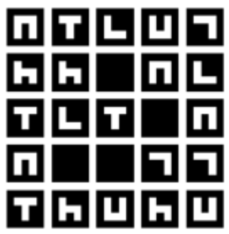
The areas of focus in this thesis are shown in the green boxes of Figure 3.1. In the following sections a summary of each these areas is presented.



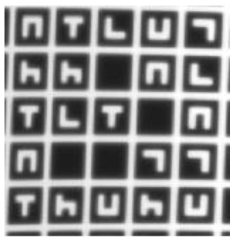
(a)

$$\begin{bmatrix} 6 & 4 & 5 & 3 & 0 \\ 1 & 1 & 7 & 6 & 5 \\ 4 & 5 & 4 & 7 & 6 \\ 6 & 7 & 7 & 0 & 0 \\ 4 & 1 & 3 & 1 & 3 \end{bmatrix}$$

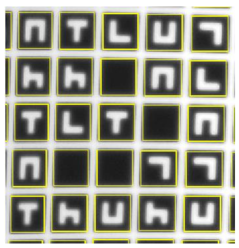
(b)



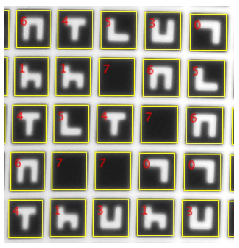
(c)



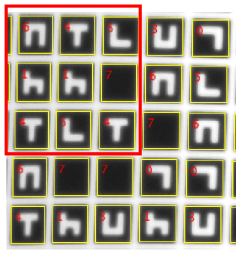
(d)



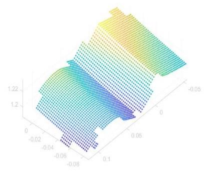
(e)



(f)



(g)



(h)

Figure 3.2: An example of different steps of performing 3D reconstruction using tag embedding methods. In this case, a pseudo-random array is used to create the structured light image. (a) Tag generation. An alphabet size of $K = 8$ binary tags is used. (b) A sample of the constructed tag label array, $\bar{I}_P$. In this case, $\bar{I}_P$ is a pseudo-random array with an alphabet of $K = 8$ and window property of 3×3. (c) Structured light image , I_P, constructed by embedding the tags in cells using the tag label array. Each cell includes a tag, a black margin and a white grid for tag separation as used in [27]. (d) A sample of the captured camera image from the projected scene, I_C. (e) Tag detection. The detected tags are shown using yellow boxes around each detected tag (f) Tag classification. The classification output which is the predicted label for each tag, is shown using red numbers in each bounding box. (g) Block detection and obtaining pixel correspondences. A 3×3 block of tags is detected. The code-word of this block is easily obtained using the labels of the tags in the block. The detected block is then matched to a block in the tag label array to find its location in the structured light image. By continuing the same process for all of the blocks in the camera image, camera-projector pixel correspondences can be obtained. (h) Assuming that the calibration parameters of the camera and the projector are given, 3D reconstruction can be performed using triangulation.

3.2.1 Tag design

As shown in Figure 3.1 and Figure 3.2, one important step in performing 3D reconstruction using tag embedding is tag classification. In order to obtain a set of accurate pixel correspondences, it is necessary to have a high performing tag classifier. One important factor that can affect the performance of tag classification is the choice of the tags used in the structured light image. The tags should be as distinguishable as possible in the camera image and under environment conditions like tag shear and blur in the camera image. As shown in Figure 3.1, one area of study in this thesis is the process of binary tag generation. Given an alphabet size of K and tag sizes of $\eta \times \eta$ pixels, it is investigated how steps for automatically designing distinguishable binary tags are taken. This topic will be covered in Chapter 4.

3.2.2 Encoding and decoding

The common practice for structured light methods using tag embedding is to utilize the concept of pseudo-random arrays [21,23,26,27,29,37,38] as shown in Figure 3.2. However, this can be computational and memory inefficient in terms of the encoding (structured light image construction) and decoding (obtaining pixel correspondences) as the projector resolution and number of embedded tags increase.

Regarding the encoding cost for pseudo-random arrays, as mentioned in Section 2.3.1, pseudo-random arrays are created using a brute force algorithm [21]. The described algorithm can be easily extended to other pseudo-random arrays for different $\bar{N}_1 \times \bar{N}_2$, K and w. The main computation cost for the described algorithm comes from the uniqueness test when a new block is completed. The complexity of this uniqueness test for a pseudo-random array of size $\bar{N} \times \bar{N}$ is $\mathcal{O}(\bar{N}^2)$ as mentioned in [41].

Also, regarding the decoding cost for methods based on pseudo-random arrays, as mentioned previously in Section 2.3.1 and as shown in Figure 2.6 (g), in order to obtain the pixel correspondences between the projector and the camera, blocks of size $w \times w$ should be detected. After block detection, the code-word for each block is obtained using the classification labels of the tags inside each block. Then, the correspondences are obtained by matching each block code-word in the camera image to a block in the tag label array. This is done by performing a search in the tag label array and comparing with the detected blocks in the camera image. The complexity of this search for a pseudo-random array of size $\bar{N} \times \bar{N}$ is also $\mathcal{O}(\bar{N}^2)$ as mentioned in [41]. However, one can use a Look-Up Table (LUT) to store the block code-word locations in the tag label array. This will eliminate the search in the tag label array but at the cost of memory for storing all the possible code-words which is dependent on the alphabet size, K, and block size $w \times w$ and has a space complexity of $\mathcal{O}(K^{w^2})$ which may not be practical for all scenarios.

As the projector resolution, $N_1 \times N_2$ increases, the number of the embedded tags, $\bar{N}_1 \times \bar{N}_2$, needed to cover the projector resolution and the required alphabet size, K, increases. Therefore, the computation and memory requirements of using a pseudo-random array for single-shot structured light in the encoding and decoding stages as mentioned above, will increase further. In addition to efficiency, robustness is also a requirement of the decoding scheme. In other words, errors can happen during tag classification and in the absence of an error detection mechanism, detected blocks in the camera image may incorrectly be matched to blocks in the structured light image, resulting in errors in camera-projector pixel correspondences and 3D reconstruction. As shown in Figure 3.1, one point of focus in this thesis will be on proposing a new method for encoding tag locations in the structured light image with an error detection mechanism which results in a more efficient and robust way of encoding, decoding and obtaining pixel correspondences, resulting in a reduction of the computation cost in comparison with methods based on pseudo-random arrays [21, 23, 26, 27, 29, 37, 38]. Also, the error detection mechanism makes the method robust to tag misclassifications in the decoding stage. Details of the proposed method and how to design single-shot structured light images using the proposed scheme is presented in Chapters 5 and 6.

Chapter 4

Tag Design

As shown previously in Figures 3.1 and 3.2, in order to construct a structured light image using tag embedding, we use binary (white/black) square tags where each tag represents a label. Also, in order to obtain pixel correspondences between a projector and camera pair, it is necessary to detect and classify the tags in the camera image. Therefore, for an accurate reconstruction, it is important to have a comparatively high classification accuracy. The lower the classification error, the higher the accuracy in pixel correspondences and 3D reconstruction. Tag misclassifications are caused by degradation of the tags in the camera image due to illumination conditions and blurriness coming from the camera and the projector. Also, the tags may be skewed according to the geometry of the surface and cause error during classification. As a result, having a classifier that can easily distinguish between different types of tags is necessary. An example of tag degradation in the camera image is shown in Figure 4.1.

One of the factors that is important to take into consideration, is the design of the used tags themselves. By intuition, the tags should be fairly distinguishable for the classifier to learn how to classify the tags with accuracy. Although the tags can be designed such that they look distinguishable to the human eye, we would like to explore whether we can objectively design a set of distinguishable binary tags for different alphabet sizes that result in a comparatively high tag classification accuracy in the camera image. For this purpose, first, the problem of tag design and concept of tag distinguishability needs to be formulated.

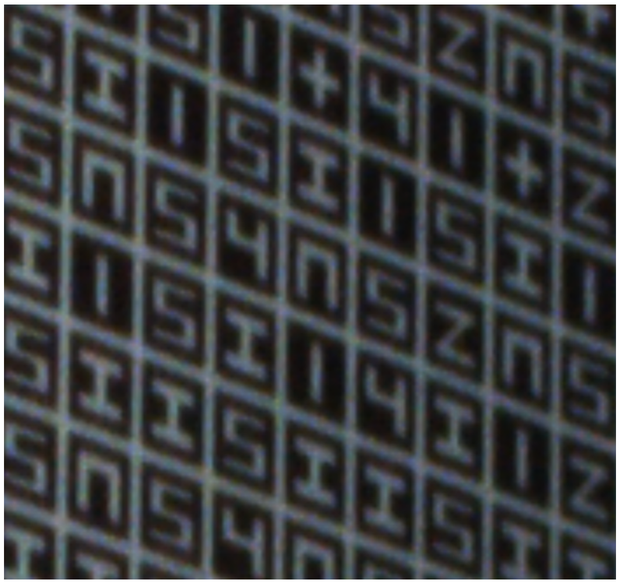


Figure 4.1: Blurriness and skewness observed in the camera image.

4.1 Tag design problem formulation

As shown earlier in Figure 3.2, tags are embedded in a set of cells in the structured light image. These cells consist of three components: A square tag symbol made of white and black pixel, a black margin, and a white grid which is used to separate the tags [27]. The problem that we want to solve is to find the most K distinguishable tags , $T = \{T_1, T_2, ..., T_K\}$, each of size $\eta \times \eta$ pixels. In the following, steps are taken to formulate an optimization problem for finding the set of optimal tags. The number of the available binary tags with size $\eta \times \eta$ pixels can be written as

$$N_T = 2^{\eta^2} \tag{4.1}$$

where N_T is the number of the available tags. The problem is to take K symbols out of N_T available tags and find the most distinguishable set. The number of combinations of K tags can be obtained as

$$N_C = \binom{N_T}{K} \tag{4.2}$$

where N_C is the number of available combinations. In order to find the set of most distinguishable tags, we can use an optimization method to maximize the tag classification accuracy as a function of the set of tags as

Table 4.1: Values for the number of available tags, N_T, and the number of available combinations, N_C, with different values of tag size, η, and alphabet size, K.

η	N_T	N_C			
		$K = 3$	$K = 4$	$K = 6$	$K = 8$
3	512	22238720	2.82e+09	2.43e+13	1.11e+17
4	65536	4.69e+13	7.69e+17	1.10e+26	8.44e+33
6	6.87e+10	5.40e+31	9.28e+41	1.46e+62	1.23e+82

$$\underset{T}{\operatorname{argmax}} \quad \text{Accuracy}(T) \tag{4.3}$$

However, using the classification accuracy as the loss function for optimization is costly and not a good choice since it is time consuming to project the tags on a target surface and train a classifier for each set of tags. The ideal case is to define a loss function, $L(T)$, that leads to less computation and does not require to project the tags on the target surface. Also, the defined loss function should be monotonic with the actual classification accuracy. In other words, we need a loss function that behaves similar to the classification accuracy, so that it represents the behaviour of the actual tag classifier. Then, the optimization problem to be solved can be rewritten:

$$\underset{T}{\operatorname{argmax}} \quad L(T) \tag{4.4}$$

By finding a set of tags that maximizes $L(T)$, the found set of tags should also result in a maximum or near to maximum value for Accuracy(T). Now, the main question remains is how the loss function $L(T)$ can be defined which also represents the tag classification accuracy? Intuitively, the effects seen in the camera image such as blurriness and shear should be somehow taken into account while defining the loss function. In other words, the defined loss function should quantify distinguishablity of tags under blur and shear. In this study, we assume the tags are projected onto a smooth and planar surface, so that tags are without any shear. We only take into account the blurriness added to the tags in the camera image while defining the loss function.

In order to model the tag blurriness in the camera image, we use Gaussian blur which is a popular method for adding blur to images. The Gaussian blur is added to a binary tag by the convolution between each tag image and the Gaussian function, $G(x, y)$:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.5}$$

where σ is the standard deviation of the Gaussian distribution and it determines the level of blurriness added to the binary tags. We also define a standard deviation vector, $\sigma' = [\sigma_1, \sigma_2, ..., \sigma_m]$, which includes a set of m standard deviations that are used to add blurriness to the tags. Then, for a given set of K tags, $T = \{T_1, T_2, ..., T_K\}$, blur is added in each of the levels in σ', resulting in a set of blurred tags for each blur level, denoted by $T_\sigma = \{T_{1,\sigma}, T_{2,\sigma}, ..., T_{K,\sigma}\}$. The idea is to use the distinguishability between blurred version of tags as the criterion of distinguishability between tags. Therefore, for each level of blurriness, $\sigma \in [\sigma_1, \sigma_2, ..., \sigma_m]$, a distance matrix, D_σ, is defined which each of its elements is the L_1 distance between each pair of blurred tags. More specifically, this distance matrix is defined as:

$$D_\sigma : \quad d_{m,n} = \begin{cases} \text{N.A}, & \text{if} \quad m = n \\ ||T_{\sigma,m} - T_{\sigma,n}||, & \text{if} \quad m \neq n \end{cases} \tag{4.6}$$

where D_σ is of size $K \times K$ and $d_{m,n}$ is the (m, n)th element of D_σ and is the L_1 distance between the mth and nth tag in the blurred set of tags, T_σ. In case $m = n$, the value of the output is Not Assigned (N.A) because the distance between the same tags is not of interest. By intuition, it is ideal that the elements of D_σ have high values and they should be maximized to increase the distinguishabilty between tags in different blur levels. Hence, we define the loss function such that it maximizes the minimum off-diagonal element of each distance matrix, D_σ. In this case, the loss function can be written as:

$$L(T) = \sum_\sigma \min\big(D_\sigma(T)\big) \tag{4.7}$$

By maximizing $L(T)$, the optimal or near optimal set of tags that are distinguishable based on the defined criterion is obtained. However, an important and challenging remaining step is to determine whether the defined loss function and the tag classification accuracy are monotonic with respect to each other. In the next section, experiments are performed to obtain optimal tags using the proposed loss function and then, it is checked if the proposed loss function behaves the same way as the tag classifier.

4.2 Experiments

In order to test the proposed scheme, as an example we use a tag size of $\eta \times \eta = 6 \times 6$ pixels and an alphabet size of $K = 8$. As shown in Table 4.1, the number of available tags,

Table 4.2: Values of available tags, N_T, and available tag combinations, N_C, for different η and K after applying the connectivity and thickness constraints.

η	N_T	N_C			
		$K = 3$	$K = 4$	$K = 6$	$K = 8$
6	913	1.2642e+8	2.8762e+10	7.9130e+14	1.1612e+19

N_T, and the possible combinations of tags, N_C, is very high which some of them are of interest. In order to overcome this problem, we first try to reduce the number of available tags, N_T, by applying constraints to the used tags. The constrains that we use are defined as follows:

1. The white pixels in each of the tags should be connected to each other.
2. Each part of the tags with white pixels should have a thickness of two pixels. In other words, we want to ensure that white pixels are thick to ensure good tag projections.

By applying these constraints, the values of available tags, N_T, and available tag combinations, N_C, can be updated as shown in Table 4.2. The next step is to optimize the defined loss function, $L(T)$. One can naively perform an exhaustive search on all of the tag combinations and calculate the optimum value. However, the very big number of the available combinations as shown in table 4.2 makes it impractical for some scenarios to use this solution. Also, the loss function, $L(T)$, is not continuous or differentiable. As a result, one of the optimization methods that can be suitable for our problem is based on random sampling for optimization [42,43,44]. In other words, in each iteration, a set of K tags are randomly sampled and the value for the loss function, $L(T)$, is calculated. At the end of the maximum iteration numbers (for example, 10^9 iterations), the set that has resulted in the highest value for the loss function is taken as the optimization solution. By using this approach, the optimal or near to optimal set of tags can be obtained. The result of this optimization approach is shown in Figure 4.2.

Besides obtaining the set of tags using the optimization approach, we have to check if the proposed loss function is monotonic with the tag classification accuracy. For this purpose, we first choose a set of 40 tags from the available N_T tags with size of 6×6 pixels. Then, these tags are projected onto three different flat surfaces (paper, wood and metal) and camera images are captured from the projected scenes. An example of these projections on three different surfaces is shown in Figure 4.3. Next, a set of $K = 8$ tags are chosen for 100 times and for each set, the classification accuracy is found by training Support Vector Machine (SVM) classifiers [45] on projected tags of the first surface and

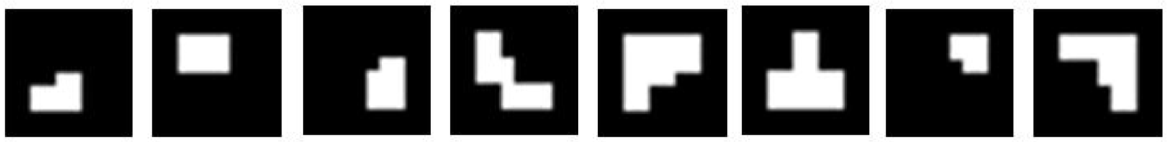


Figure 4.2: The set of tags obtained using the optimization approach and the proposed loss function.

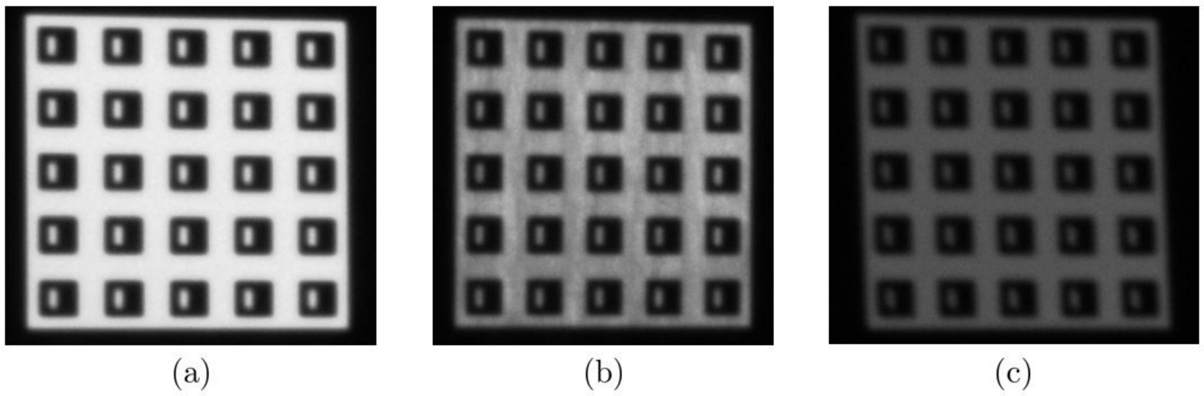


Figure 4.3: An example of the projected tags on three different flat surfaces: (a) Paper, (b) Wood, and (c) Metal

testing the classifier on the two other surfaces. Also, the loss function in Equation 4.7 is calculated for all of the same chosen 100 set of tags. Finally, the values for loss function, $L(T)$ are plotted against the values of the tag classification accuracy, Accuracy(T). The result of this plotting is shown in Figure 4.4. It can be seen that the loss function and the accuracy are not monotonic, an indication of the fact that the proposed loss function can not be a representation of the tag classification accuracy. **This also reveals the challenge of finding a function that behaves in the same way as the classifier does.** It is worth noting that other configurations of defining the loss function was tested which also did not result in a monotonic function with the classification accuracy.

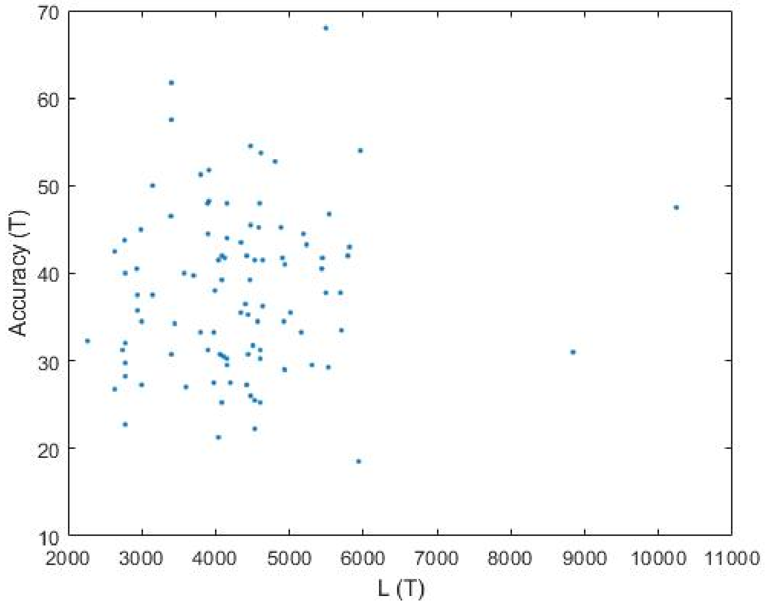


Figure 4.4: $L(T)$ vs. Accuracy(T) for a set of 100 tags which each set includes $K = 8$ tags with size of 6×6 pixels. The measurement of Accuracy(T) is based on real images and learned classifiers, whereas $L(T)$ is based on simulation/assumed loss function.

4.3 Conclusion

In this chapter, tag design for single-shot structured light image reconstruction was formulated as an optimization problem. The most challenging part of this formulation is to define a loss function that takes into account the effects seen in the camera image caused by the projector, the target surface and the camera itself. In addition, the behaviour of the classifier can also add to the challenge as it is not really known what the classifier is learning and it is usually treated as a black box. In the following chapters, we empirically choose the set of tags that result in a comparatively high tag classification accuracy.

Chapter 5

Direct Block Address Encoding

As mentioned in Section 3.2.2, the use of pseudo-random arrays [21, 23, 26, 27, 29, 37, 38] has a computation and memory cost in the encoding (structured light image construction) and decoding (obtaining pixel correspondences) stages. In order to reduce this cost, a new framework for single-shot 3D reconstruction based on tag embedding is proposed, where the structured light image is partitioned into a number of non-overlapping blocks that each represent a code-word for directly encoding the block column and row addresses resulting in an efficient computation for both constructing the structured light images and obtaining the projector-camera pixel correspondences. As will be shown, unlike previous methods [21, 23, 26, 27, 29, 37, 38], the proposed scheme requires no additional search cost in the the projector space nor the storing of any Look Up Table (LUT), leading to reductions in computational complexity and storage requirements, particularly attractive for very high resolution displays.

5.1 Methodology

An overview of the proposed scheme is shown in Figure 5.1. It is assumed that a set of K distinguishable binary tags, T, is given. The proposed method simplifies the structured light image construction and decoding schemes by proposing to partition the structured light image, I_P, and the tag label array, $\bar{I}_P$, into a number of unique non-overlapping blocks (Figure 5.1 (a) to (c)) as opposed to methods based on pseudo-random arrays [21, 23, 26, 27, 29, 37, 38] where uniqueness of overlapping blocks are used. More specifically, the structured light image is partitioned into $\hat{N}_1 \times \hat{N}_2$ unique blocks as:

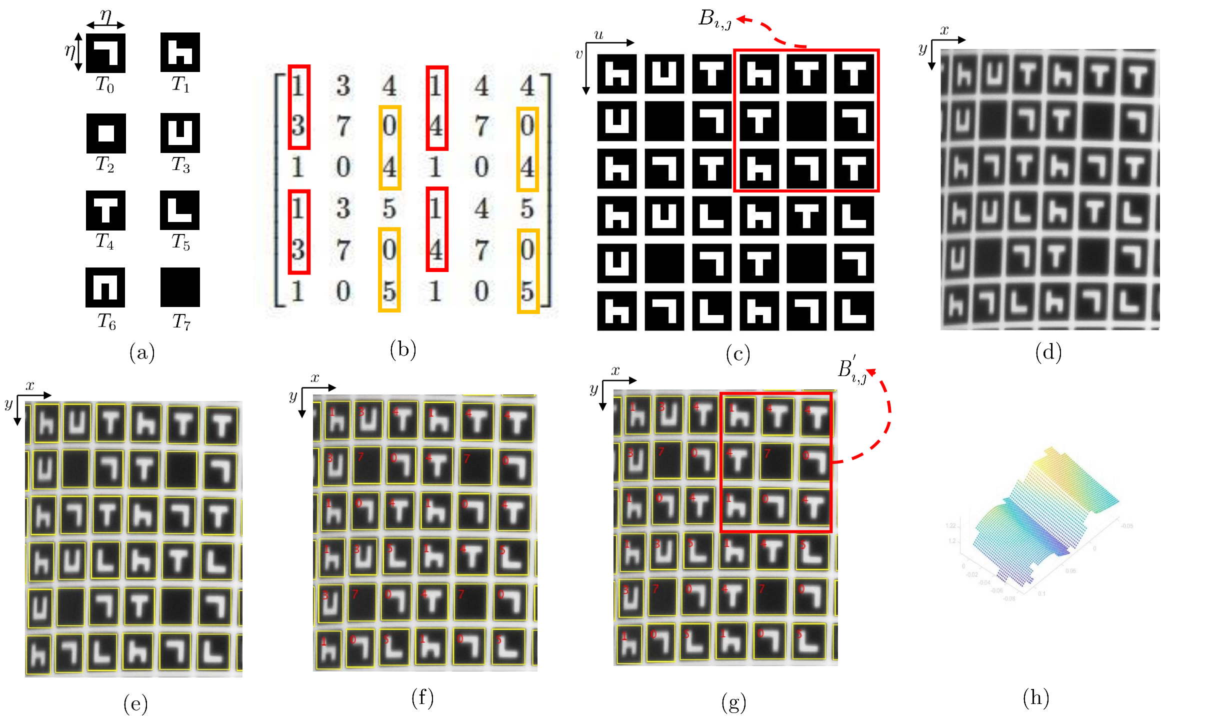


Figure 5.1: Example of different steps of performing 3D reconstruction using direct block address encoding. (a) Tag generation: An alphabet size of $K = 8$ binary tags of size $\eta \times \eta$ pixels is used. (b) A sample of the constructed tag label array, $\bar{I}_P$, which consists of 3×3 non-overlapping blocks where a marker tag (label 7) is in the middle of each block. The surrounding elements are used to encode the row and column indices of each block. Red and orange rectangles indicate encoded row and column indices in each block (c) Structured light image, I_P, constructed using the tag label array by embedding the tags in cells of size $p \times p$ pixels. (d) A sample of the captured camera image from the projected scene, I_C. (e) Tag detection: The detected tags are shown using yellow boxes around each detected tag (f) Tag classification: The classification output which is the predicted label for each tag, is shown using red numbers in each bounding box. (g) Block detection and obtaining pixel correspondences. A 3×3 block of tags is detected (in red). The code-word and address of this block is easily obtained using the labels of the tags in the block. The pixel correspondences are obtained directly using the row and column index and there is no need for a search or using an LUT as opposed to pseudo-random methods (h) Assuming that the camera and projector are calibrated, 3D reconstruction is performed using triangulation.

$$I_P = \begin{bmatrix} B_{0,0} & \cdots & B_{0,\hat{N}_2-1} \\ \vdots & \ddots & \vdots \\ B_{\hat{N}_1-1,0} & \cdots & B_{\hat{N}_1-1,\hat{N}_2-1} \end{bmatrix} \tag{5.1}$$

where $B_{\imath,\jmath}$ is the $\imath\jmath$th block in the structured light image, $\imath \in 0, 1, ..., \hat{N}_1$, $\jmath \in 0, 1, ..., \hat{N}_2$ and each block includes $w \times w$ tags. As a result, the number of embedded tags in the vertical direction, $\bar{N}_1$, is equal to $\bar{N}_1 = w\hat{N}_1$. Similarly, $\bar{N}_2 = w\hat{N}_2$.

In addition, the number of blocks that can be embedded in the vertical direction inside the structured light image is a function of the projector resolution $N_1 \times N_2$, cell size (tag size + black margin + white grid) $p \times p$, and the block size $w \times w$, and can be derived as:

$$\hat{N}_1 = \left\lfloor \frac{N_1}{wp} \right\rfloor \tag{5.2}$$

The same can be written for the number of the blocks in the horizontal direction, $\hat{N}_2$. As a result of Equation 5.1, the tag label array which includes tag label orders in the structured light image, $\bar{I}_P$, is an array of size $\bar{N}_1 \times \bar{N}_2$ which consists of $\hat{N}_1 \times \hat{N}_2$ blocks of size $w \times w$. This can be written as:

$$\bar{I}_P = \begin{bmatrix} \bar{B}_{0,0} & \cdots & \bar{B}_{0,\hat{N}_2-1} \\ \vdots & \ddots & \vdots \\ \bar{B}_{\hat{N}_1-1,0} & \cdots & \bar{B}_{\hat{N}_1-1,\hat{N}_2-1} \end{bmatrix} \tag{5.3}$$

where $\bar{B}_{\imath,\jmath}$ is the $\imath\jmath$th block in the tag label array of size $w \times w$. Each block in the tag label array, $\bar{B}_{\imath,\jmath}$, is obtained by encoding the block indices, $\imath$ and $\jmath$, in each block. Therefore, the proposed method is called direct block address encoding. In the following section, it will be explained how each block is obtained that will also allow encoding the block indices, $\imath$ and $\jmath$, into a unique matrix representation.

5.1.1 Block construction

For each pair of projector block indices $(\imath, \jmath)$ in the structured light image, I_P, the proposed method uses $w \times w$ tags in that block, $B_{\imath,\jmath}$, to encode the block indices. For this purpose, first one tag type is embedded in the middle of every block that is unique, a so-called marker tag, in order to know the origin for each block in the camera image (Figure 5.1 (b) and (c)).

The remaining $w^2 - 1$ tag locations and the remaining $K - 1$ tag types will then be used to encode the $\imath\jmath$th block indices and their associated error detection digits. The inclusion of error detection is motivated by challenges affecting the captured imagery, such as illumination conditions, surface geometry, and/or camera blurriness that may cause tag misclassifications and error in obtaining pixel-correspondences. Several encoding schemes exist in the literature that can be used for adding error detection digits [36]. In principle, one block index, $(\imath, \jmath)$, could be coded jointly in all $w^2 - 1$ digits (a codeword of length $w^2 - 1$ digits), however, it is simplest to consider coding each of $\imath, \jmath$ individually, each using $(w^2 - 1)/2$ digits.

Let the number of digits in base $K - 1$ for address encoding per block in each direction be d, and their corresponding number of control digits per block be $n \leq d$. As a result, one can relate the number of tags in each block, and number of encoding and control digits as follows:

$$2(d + n) + 1 = w^2 \tag{5.4}$$

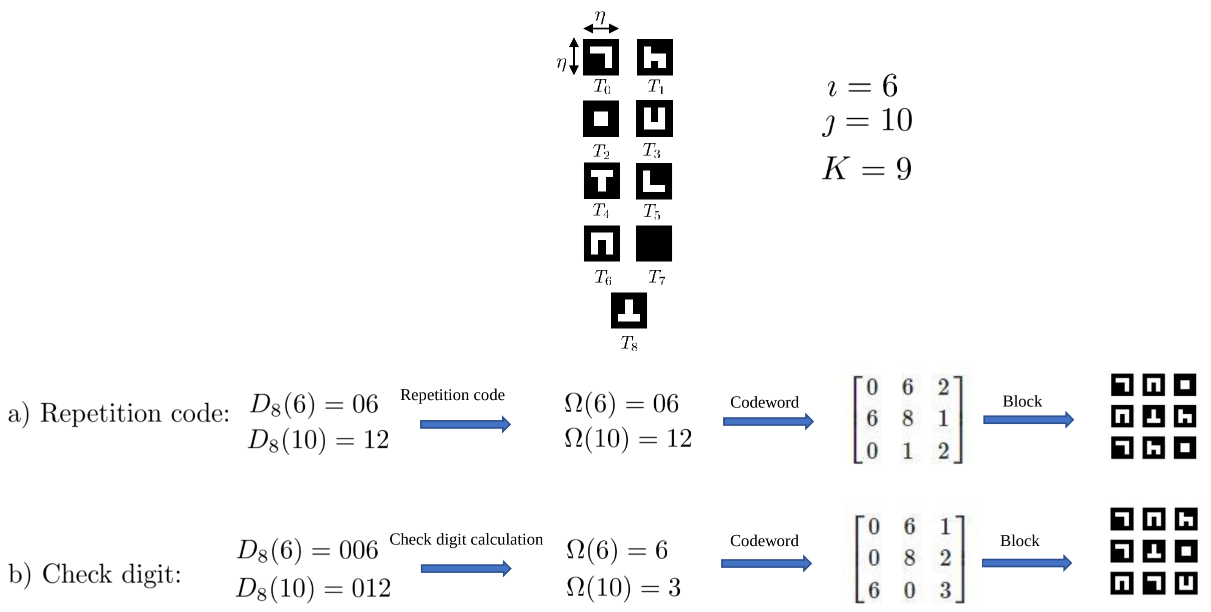
The column and row indices, $\imath$ $\jmath$, can be encoded by converting the indices into base $K - 1$ with d digits and by adding n control digits for the purpose of error detection. Now, the code-word of the $\imath\jmath$th block, $c_{\imath,\jmath}$, can be generated as follows:

$$c_{\imath,\jmath} = [D_{K-1}(\imath), \Omega_{K-1}(\imath), k^*, D_{K-1}(\jmath), \Omega_{K-1}(\jmath)] \tag{5.5}$$

where $c_{\imath,\jmath}$ is a column vector of size $w^2 \times 1$, $0 \leq \imath, \leq \hat{N}_1 - 1$, $0 \leq \jmath \leq \hat{N}_2 - 1$, as well as $D_a(b)$ is a function that converts a number, b, from base 10 to base a with d digits, $\Omega(\cdot)$ is an error detection function, and k^* is the digit corresponding to the marker tag for reconstructing the partitioned blocks. Next, the labels within the $\imath, \jmath$th block in $\bar{I}_P$ in Equation (5.3), $\bar{B}_{\imath,\jmath}$, are obtained by representing $c_{\imath,\jmath}$ in a w^2 matrix format. By continuing this process for each block address, the construction of the tag label array, $\bar{I}_P$, and the structured light image, I_P, using direct block address encoding is completed. As the proposed method encodes the block addresses directly, there is no need for a uniqueness test as it is required for methods based on pseudo-random arrays [21, 23, 26, 27, 29, 37, 38] (Section 3.2.2), resulting in a significant reduction of computation for the encoding stage (structured light image construction).

5.1.2 Projector Address Encoding

As mentioned previously in Section 5.1.1, due to the challenges affecting the camera image, such as difficult illumination conditions, surface geometry and/or camera blurriness,



$$\begin{bmatrix} 0 & 6 & 2 \\ 6 & 8 & 1 \\ 0 & 1 & 2 \end{bmatrix} \qquad \begin{bmatrix} 0 & 6 & 1 \\ 0 & 8 & 2 \\ 6 & 0 & 3 \end{bmatrix}$$

Figure 5.2: A numerical example for block address encoding using $K = 9$ tags, at $\jmath = 10$ and $\imath = 6$. In this figure, two solutions have been used for encoding the block indices: (a) Repetition code and, (b) Check digit.

errors can occur during tag classification, and thus in the pixel correspondences and 3D reconstruction stages [21]. Therefore, adding error detection capabilities is necessary for an accurate reconstruction. Different encoding schemes can be used to encode the column and row indices. The choice of the encoding scheme affects the number of address and control digits, d and n. In this section, some encoding methods are introduced that can be used for direct block address encoding.

One-time Repetition Code

This address encoder simply repeats one time the address digits, D, to create redundant control digits, Ω. Therefore, $d = n$ [36, 46]. This encoder can be defined as:

$$\Omega_{K-1}(\imath) = D_{K-1}(\imath) \tag{5.6}$$

In the decoding stage, if any of the control digits is different from its corresponding address digit, an error is detected. An example of block address encoding using one time repetition code is shown in Figure 5.2 (a).

Also, as an extension to the introduced repetition code, one can use this type of coding with a location offset for more robustness to tag miscalssifications. As the repetition code repeats the tags in the block, in case the classifier has a local error in classifying one particular type of tag, it is possible that the decoder will be unable to detect the errors that will result in incorrect correspondences. In other words, first, the codeword is created as the same way expressed in Figure 5.2 (a) and then, a circular location offset is added to each element of the codeword (except for the marker tag). For example, for the 3rd element of the codeword is added by 2 and then reduced by K to prevent codeword elements exceed $K-1$. In the decoding stage, the offset can be calculated based on the tag location in the camera block. Then, Equation 5.6 can be used to check if errors in classification exist or not.

Check Digit

Let us assume that the alphabet size, including the marker tag, for constructing the encoded blocks of the structured light image is K. Therefore $K-1$ different tags remain for encoding the block addresses in each direction. The check digit add one or more digits from 0 to $K-1$ to a given encoded address in order to detect errors if a predetermined relation between the message digits, also called the check equation, is satisfied or not [47, 48]. Several versions of check digit schemes have been proposed with different check equations [47, 48]. As an example, one method is to calculate the check digit as the modulo $K-1$ of the summation of message digits as [47]:

$$\Omega_{K-1}(\imath) = \mathbf{mod}(\sum D_{K-1}(\imath), K-1) \tag{5.7}$$

where $\mathbf{mod}(a, b)$ is an operator that returns the remainder of dividing a by b. Then, in the decoding stage the digits of the received message are checked to see if they satisfy Equation 5.7. An example of address encoding using the scheme in Equation 5.7 is shown in Figure 5.2 (b).

5.1.3 Choice of K

Until now, the values of the alphabet size, K, cell size in pixels, $p \times p$, were chosen somewhat arbitrarily. However, for a given projector with a resolution of $N_1 \times N_2$ pixels, we wish to fill the projector space with as many blocks as possible. In order to fulfill this requirement using block address encoding, the number of available encoding addresses in the direction of the larger projector image dimension, normally the x direction (N_2), needs to be at least as large as the number of block addresses in that direction, thus:

$$\underbrace{(K-1)^d}_{\text{Maximum No. of Encoding Addresses}} \geq \underbrace{\frac{N_2}{wp}}_{\text{No. of Projector Blocks}} \tag{5.8}$$

where d is the number of available digits to encode blocks in the x-direction, which is determined by the chosen encoding scheme. The minimum alphabet size, $K_{\min}$, can then be found as

$$K_{\min} = \left\lceil \left(\frac{N_2}{wp}\right)^{1/d} \right\rceil + 1 \tag{5.9}$$

where $\lceil \cdot \rceil$ is the ceiling operator. $K_{\min}$ determines a lower bound on the required alphabet size to cover the projector space; clearly Equation (5.9) can be used to find K for different projector resolutions, cell sizes and window sizes.

5.1.4 Projector-Camera Pixel Correspondences

After generating the structured light image, I_P, using direct block address encoding, the image is projected onto a target surface and a corresponding image, I_C, is captured by a camera as shown in Figure 5.1 (d). Given I_C, the goal is to obtain the pixel correspondences as shown in Figure 2.3 between the projector and camera.

The first step in this process is to detect and recognize the tags in the camera image, where global thresholding and connected component analysis can be used for tag detection (Figure 5.1 (e)) and multiple class support vector machine (SVM) as used in [38] for tag recognition (Figure 5.1 (e)), respectively. Next, the non-overlapping blocks in the camera image, $B'_{\tilde{\imath},\tilde{\jmath}}$, which are unique can be recovered from I_C by first attempting to identify the marker tags (Figure 5.1 (g)), and then sampling a window of size $w \times w$ centered around each marker tag. After finding each block, its code-word is reconstructed and if no error is detected, it is decoded to obtain the predicted block indices, $\tilde{\imath}, \tilde{\jmath}$. By obtaining the decoded block address, $(\tilde{\imath}, \tilde{\jmath})$, the location of the block is obtained in the projector image and pixel correspondences between the matching tags can be obtained.

As a result, unlike existing pseudo-random array based methods [21,23,26,27,29,37,38], the proposed direct block address encoding here requires neither any additional search in the projector space nor any sort of LUT, and the pixel correspondences are obtained directly once the code-words in the camera captured image are decoded which is a significant computation and required memory reduction in comparison with methods based on pseudo-random arrays which was explained in Section 3.2.2.

5.2 Experimental Results

The choice of different parameters such as tag size, cell size, block size, alphabet size and encoding scheme has a direct effect on the overall structured light reconstruction performance. The ideal case is to have many pixel correspondences (smaller tag size), an easier classification task (smaller alphabet size), an encoding scheme which is robust to tag misclassifications, and a fast run time. However, there is a trade-off in choosing the design parameters to meet these goals.

In this section, experiments for single-shot image construction are presented using different encoding schemes introduced in Section 5.1.2, projector resolutions, tag sizes and block sizes. The encoding schemes are compared in terms of the minimum required alphabet size, error detection rate and the run time. In the second part of the experiments, 3D reconstruction is performed using the proposed direct block address encoding scheme. Finally, the proposed scheme is compared to methods based on pseudo-random arrays [21, 23, 26, 27, 29, 37].

5.2.1 Comparing Address Encoding Schemes

Minimum Alphabet Size

In this section, the efficiency of different encoding schemes, Repetition Code (RC) as shown in Figure 5.2 (a) and Check Digit (CD) as shown in Figure 5.2 (c), in terms of the minimum alphabet size is assessed. This is important because an encoding scheme that requires a low alphabet size will have less uncertainty in the tag classification step. Equation 5.9 determines that the minimum alphabet size, K_{min}, is a function of the cell size p, projector resolution in bigger dimension N_2, block size in terms of number of tags, w, and the number of digits used for block address encoding, d, which is a function of the used encoding scheme. Based on Equation 5.4, d can be expressed in terms of w and n as $d = (\frac{w^2-1}{2} - n)$. For RC, $n = \frac{w^2-1}{4}$ and for CD, $n = 1$, where n is the number of added control digits. From Equations 5.4 and 5.9, the minimum alphabet size, $K_{\min}$, can be derived as:

Table 5.1: Comparing the number of blocks, minimum alphabet size (K), and the encoding and decoding times for Check Digit (CD) and Repetition Code (RC) schemes at different projector resolutions, block and cell sizes, where the best is denoted by bold.

Projector Resolution	Block Size $\hat{n}_1 \times \hat{n}_2$	Tag Size $\eta \times \eta$	Cell size $p \times p$	Block num. $\hat{N}_1 \times \hat{N}_2$	K_{min} as defined in (5.9)		Encoding Time (s)		Decoding Time (s)	
					CD	RC	CD	RC	CD	RC
WXGA (800×1280)	3×3	6×6	12×12	22×35	**5**	7	0.056	**0.051**	**0.051**	**0.051**
		12×12	24×24	11×17	**4**	6	0.034	**0.030**	**0.026**	0.028
	5×5	6×6	12×12	13×21	**3**	**3**	0.064	**0.044**	0.048	**0.041**
		12×12	24×24	6×10	**3**	**3**	0.030	**0.025**	**0.023**	**0.023**
4k (2160×4096)	3×3	6×6	12×12	60×113	**6**	12	0.212	**0.175**	0.212	**0.194**
		12×12	24×24	30×56	**5**	9	0.085	**0.078**	**0.078**	0.079
	5×5	6×6	12×12	36×68	**3**	4	0.194	**0.159**	**0.168**	**0.168**
		12×12	24×24	18×34	**3**	**3**	0.083	**0.072**	**0.066**	0.067
8k (4320×7680)	3×3	6×6	12×12	120×213	**7**	16	0.635	**0.547**	0.671	**0.653**
		12×12	24×24	60×106	**6**	12	0.212	**0.168**	**0.194**	**0.194**
	5×5	6×6	12×12	72×128	**3**	4	0.688	**0.547**	0.741	**0.653**
		12×12	24×24	36×64	**3**	**3**	0.194	**0.153**	0.159	**0.157**

$$K_{\text{min}} = \left\lceil \left(\frac{N_2}{w\eta}\right)^{-(\frac{\omega^2-1}{2}-n)} \right\rceil + 1 \tag{5.10}$$

In order to assess the efficiency of different encoding schemes, Equation 5.10 is used to obtain K_{min} for the two encoding schemes discussed in Section 5.1.2 at three projector resolutions (WXGA, 4K, 8K), two tag sizes (6×6 and 12×12 pixels), two cell sizes (12×12 and 24×24 pixels), and two block sizes (3×3 and 5×5). As can be seen in Table 5.1, CD offers the lowest minimum alphabet size. In addition, these two encoding schemes allow reducing the required alphabets when a larger block size is selected.

Running Time

We also compare the run time of encoding and decoding stages for each encoding scheme. For this purpose, tag label arrays (Equation 5.3) are created for different encoding schemes (RC and CD), projector resolutions, block sizes and tag sizes. The average time (1000 iterations) to construct the tag label arrays is reported as encoding time in Table 5.1. In addition, for each of the cases in Table 5.1, the average time to decode the codewords of the blocks in the tag label arrays and obtain the block addresses is measured and reported as the decoding time.

Table 5.2: Error Detection Rate (EDR) for the Repetition Code (RC) snd Check Digit (CD) method in the presence of ξ random number of errors per block for a WXGA projector, where alphabet size $K = 7$, cell size $p = 12$, tag size $\eta = 6$, block size $\hat{n} = 3$ is used and the number of blocks is equal to $\hat{N}_1 \times \hat{N}_2 = 22 \times 35$. Note that RC detects all blocks with odd number of errors.

	EDR							
ξ	1	2	3	4	5	6	7	8
RC	1.000	0.956	1.000	0.999	1.000	0.998	1.000	1.000
CD	1.000	0.778	0.889	0.978	0.963	0.9258	0.9424	1.000

As shown in Table 5.1, generally RC provides the best encoding/decoding run time. However, the difference between the two schemes (CD and RC) is small such that both are plausible to be used in fast implementations.

Error Detection Rate

In this section, we compare the error detection capability of the presented block address encoding schemes. For this experiment, we create tag label arrays for each of the two mentioned encoding methods (CD and RC) using a projector of resolution 800×1280 pixels (WXGA projector), blocks of 3×3 cells, each cell of 12×12 pixels including tags of size 6×6 pixels, and for an alphabet size of $K = 7$. In this case, the tag label array consists of 22×35 blocks. Assuming the middle tag is classified correctly, tag misclassification can occur in each of the eight tags in the neighborhood of the marker tag. Therefore, for this experiment, for each block in the tag label array and number of possible misclassifications, random errors are added at random locations for 1000 times. Next, the average Error Detection Rate (EDR) is calculated as the ratio of the number of detected errors to the total number errors added.

The results for comparing the two considered encoding schemes in terms of the EDR at different ratios of corrupted digits (misclassified tags) per block are shown in Table 5.2. From this table, RC obtains a higher EDR at all block noise levels. On the other hand, CD method offers poorer error detection capability but maintains the smallest alphabet size, as shown in Table 5.1. This method makes the tag classification task easier in comparison with other methods, however, this would be at the cost of lower error detection capability.

Table 5.3: The four experimented scenarios for performing 3D reconstruction using the proposed block address encoding where a WXGA projector is used. The block sizes are of size 3×3 and 5×5, and the used cell sizes are of size 12×12 and 24×24 pixels. The alphabet size is derived using Equation 5.9.

Projector Resolution	Block Size $\hat{n}_1 \times \hat{n}_2$	Tag Size $\eta \times \eta$	Cell Size $p \times p$	Block num. $\hat{N}_1 \times \hat{N}_2$	$K_{\min}$ as defined in (5.9) RC
WXGA (800×1280)	3×3	6×6	12×12	22×35	7
		12×12	24×24	11×17	6
	5×5	6×6	12×12	13×21	3
		12×12	24×24	6×10	3

Conclusion

In this section, various experiments were performed to evaluate the performance of different encoding schemes. The performance of each of these schemes was measured in terms of the minimum alphabet size, number of pixel correspondences which is proportional to the number of embedded tags, EDR and running time as shown in Table 5.2 and Table 5.1. The choice of different parameters totally depends on the requirements of the problem to solve. If a high error detection rate is required and having a number of incorrect correspondences is not tolerable, then, using RC is preferred. However, this will come at the cost of a higher required alphabet size. If using a block size of 3×3, Table 5.1 indicates that as the projector resolution increases, the required alphabet size increases rapidly. In this case, the tag size should be adjusted such that the tags are comparatively distinguishable in the camera image, resulting in a good classification accuracy. However, increasing the tag size will result in fewer number of correspondences. Therefore, if the high alphabet size makes the classification problem hard, it would be desirable to use a block size of 5×5 since it requires a lower alphabet size.

5.2.2 3D Reconstruction

In this section, experiments of performing 3D reconstruction using the proposed block address encoding are presented. The experiments are performed using a WXGA projector with a resolution of 800×1280 pixels and a Point Grey Flea camera with a resolution of 2048×2448 pixels. For the experiments, the camera and the projector are calibrated as proposed in [18].

In order to obtain a high robustness to potential tag classification errors, Repetition Code (RC) [36, 46] is used as the encoding scheme. Also two tag sizes (6×6 pixels and 12×12 pixels), two cell sizes (12×12 pixels and 24×24 pixels), and two block sizes (3×3 and 5×5) are used for the sake of comparison. Based on Equation 5.9 and Table 5.1, each of the above mentioned scenarios would require a different minimum alphabet size that may have an effect on the tag classification accuracy. In order to assess whether the chosen tag size for the required alphabet size is big enough, the structured light image should be created and then projected onto the target surface. If the tags are fairly distinguishable in the camera image, then, the process can proceed to perform 3D reconstruction. Otherwise, either the tag size/cell size or the block size should be increased. A summary of the performed experiments is shown in Table 5.3.

The tags and a sample of the constructed structured light images are shown in Figure 5.3. Each cell in the structured light image is consists of three components. First, a square tag which is a symbol made of white and black pixels. Second, a black margin which is used to separate the tags inside the structured light image. Third, a white grid, as used in [27], which helps to detect the tags. In the case of using cells of size 12×12 pixels, the cell consists of a 6×6 pixel square tag, 2 pixel black margin and a gird of 1 pixel around the symbol. This would be doubled for cells of size 24×24 pixels.

After structured light image construction using the proposed direct block address encoding, the structured light images should be projected onto the target surfaces to perform 3D reconstruction. A curved-zigzag surface is used for this purpose. Also, a sample of the projected structured light images are shown in Figures 5.4 (a)-(d). Then, for each of the test scenarios, the tags are first detected using global thresholding and connected components analysis as used in [27,38]. After tag detection, tag classification is performed using a trained SVM classifier [40]. In order to train the classifier, tags are projected onto the target surface and then, used as training examples.

After tag detection and classification, the marker tag (middle tag in each block) and unique non-overlapping blocks are detected. Then, the codeword for each block is decoded and if no errors are detected in the block, its address in the structured light image and pixel correspondences between the projector and the camera tags are obtained as centroids of corresponding tags. Finally, by obtaining the pixel correspondences and using the projector-camera calibration parameters, 3D reconstruction is performed using triangulation (Figure 5.1).

For all test cases, all of the tags on the target surface are classified correctly, indicating that a 12×12 pixel cell size (6×6 pixel tag size) is big enough to reach a high classification

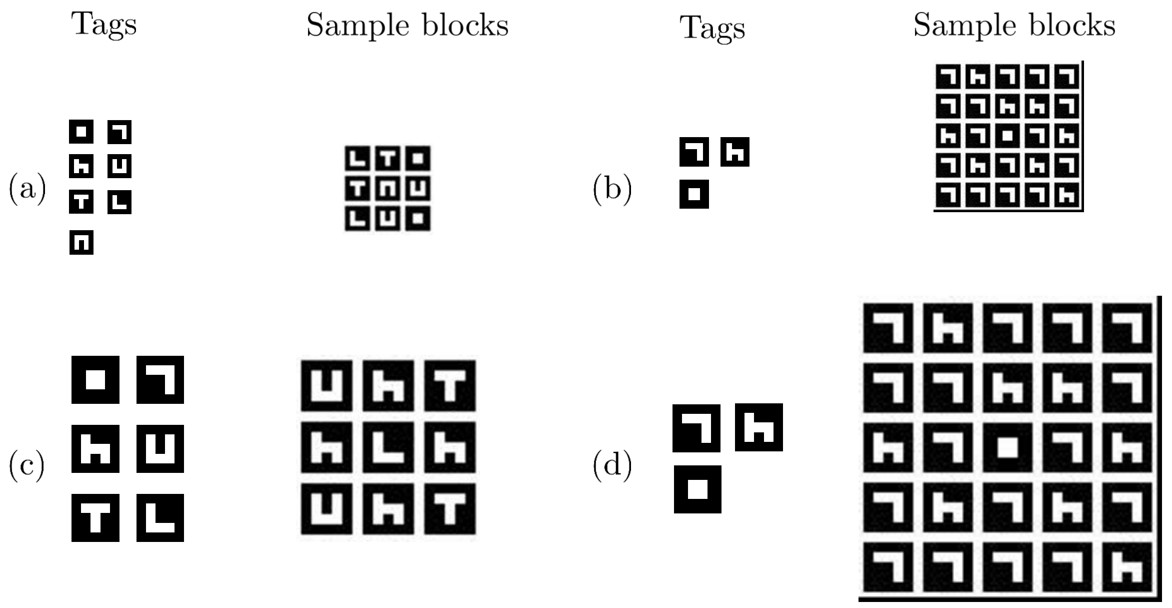


Figure 5.3: The used tags and one sample block of the constructed structured light images to perform 3D reconstruction. In all cases, a repetition code is used. (a) Tag size $\eta = 6 \times 6$ pixels, block size $\omega = 3 \times 3$ and alphabet size $K = 7$, (b) tag size $\eta = 6 \times 6$ pixels, block size $\omega = 5 \times 5$ and alphabet size $K = 3$, (c) tag size $\eta = 12 \times 12$ pixels, block size $\omega = 3 \times 3$ and alphabet size $K = 6$, (d) tag size $\eta = 12 \times 12$ pixels, block size $\omega = 5 \times 5$ and alphabet size $K = 3$.

accuracy for our setup. Its worth noting that this will change based on the configuration (distance between camera and surface, distance between projector and surface, projector type, etc). The results of performing 3D Reconstruction for all of the test scenarios is shown in Figure 5.5. Also, the number of detected tags and the number of obtained pixel correspondences are shown in Table 5.4. It can be seen that using a lower tag size/cell size results in a denser set of pixel correspondences and denser point cloud (more complete reconstruction) since a larger number of tags are embedded in the structured light image. Also, comparing the different block sizes, as shown in Table 5.4 and Figure 5.5, it can be seen that using a smaller block size results in a denser set of pixel correspondences and point cloud. The reason behind this observation is that a set of tags in the surface borders exist that are not part of a complete block. Therefore, pixel correspondences are not obtained for these tags which results in a loss of pixel correspondences from the surface border. We call these tags the unassociated set of tags, which will be further discussed in Chapter 6. When using a block size of 5 × 5 tags, it is more probable to have a higher

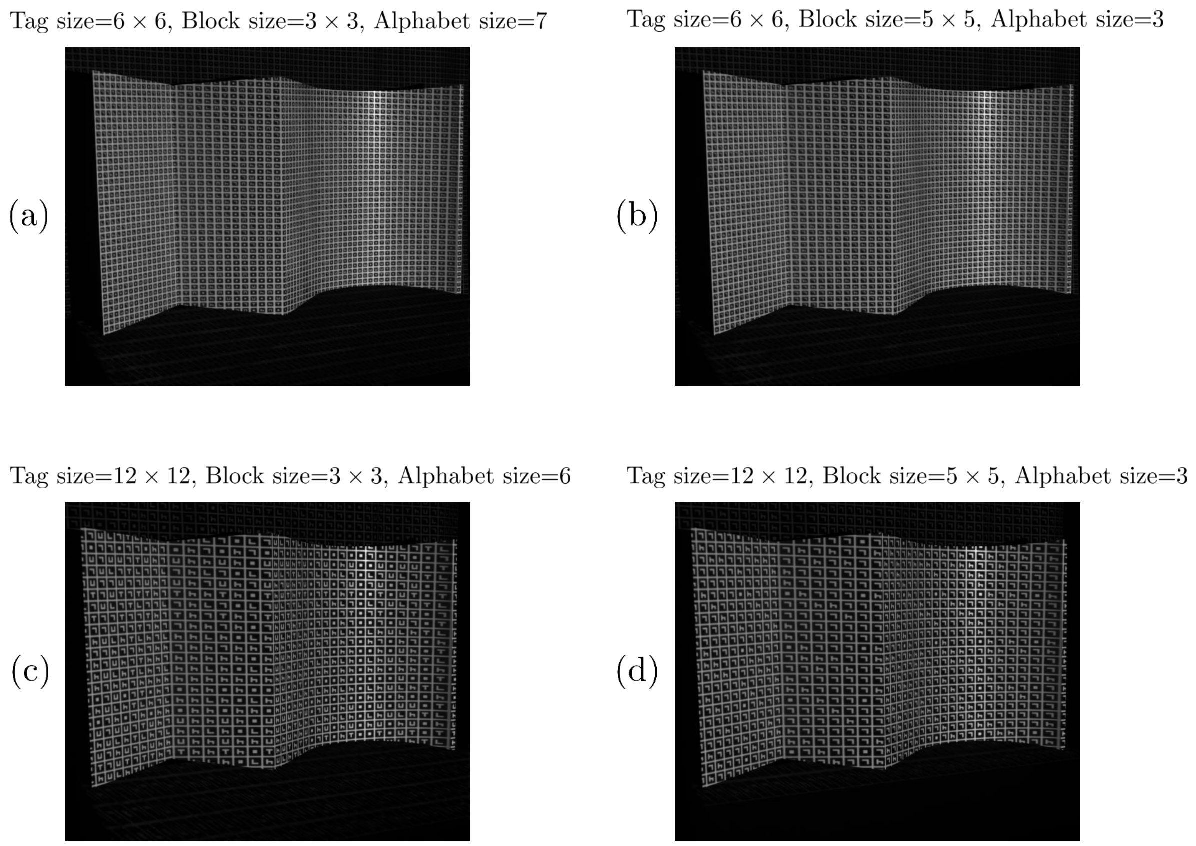
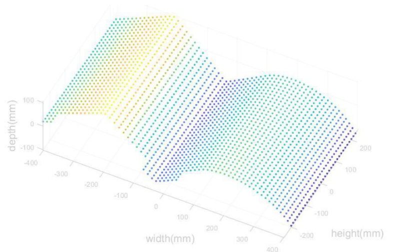

Tag size=6 × 6, Block size=3 × 3, Alphabet size=7
(a)
Tag size=6 × 6, Block size=5 × 5, Alphabet size=3
(b)
Tag size=12 × 12, Block size=3 × 3, Alphabet size=6
(c)
Tag size=12 × 12, Block size=5 × 5, Alphabet size=3
(d)

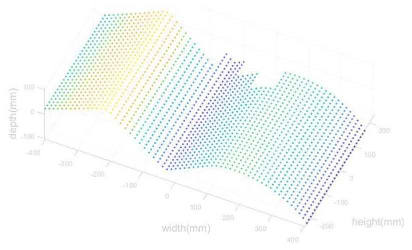
Figure 5.4: Captured camera images of the projected structured light images on the target surface

number of unassociated tags in the border which explains the difference in the. number of pixel correspondences when using 3×3 blocks.

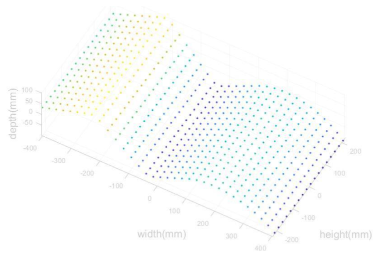
(a) Tag size=6 × 6, Block size=3 × 3, Alphabet size=7



(b) Tag size=6 × 6, Block size=5 × 5, Alphabet size=3



(c) Tag size=12 × 12, Block size=3 × 3, Alphabet size=6



(d) Tag size=12 × 12, Block size=5 × 5, Alphabet size=3

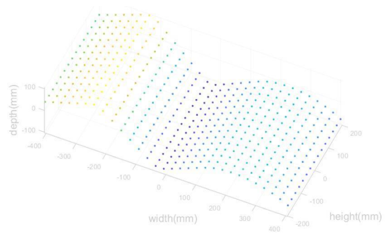


Figure 5.5: The reconstructed point cloud for all test scenarios which is obtained using triangulation. A smaller tag and block size results in a more dense point cloud.

5.3 Comparison with pseudo-random arrays

5.3.1 3D reconstruction

As mentioned in Section 5.1, the proposed direct block address encoding eliminates the computation for uniqueness tests in pseudo-random encoding (structured light image construction) and the need for performing an exhaustive search or using a Look-Up Table (LUT) in the decoding (obtaining pixel correspondences) stage, since the proposed approach creates non-overlapping blocks which carry the encoded block address. In order to have a further comparison between the proposed method and methods based on pseudo-random arrays [21, 23, 26, 27, 29, 37], in this section a set of 3D reconstruction experiments is performed using both methods. For a fair comparison, the same tag size, cell size, block size, alphabet size and tags should be used. For this purpose, a cell size of 24×24 pixels, tag size of 12×12 pixels, and an alphabet size of $K = 6$ is used and a structured light image is constructed using direct block address encoding as shown in Figure 5.3 (c). For the pseudo-random array, we follow the method in [21] and create a pseudo-random

Table 5.4: Comparison between the test scenarios in terms of the detected tags and the number of obtained pixel correspondences.

Block Size	Tag Size	Cell Size	Number of detected tags	Number of pixel correspondences
3×3	6×6	12×12	2290	2016
	12×12	24×24	734	594
5×5	6×6	12×12	2290	1800
	12×12	24×24	730	450

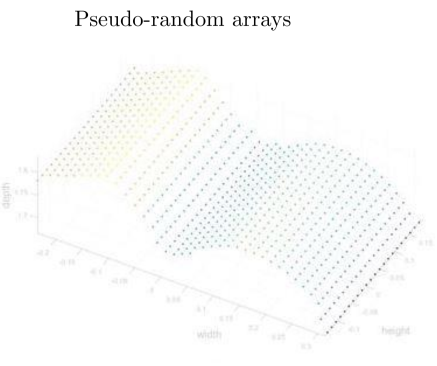
Table 5.5: Comparison between the Pseudo-Random Array (PRA) method and the proposed direct block address encoding in terms of the detected tags and the number of obtained pixel correspondences.

Method	Number of detected tags	Number of pixel correspondences
PRA with LUT	734	730
PRA with search	734	730
Proposed	734	594

array with the same number of embedded tags (33×51 tags) with a block property of 3×3 and alphabet size of $K = 6$ and the same tag and cell sizes as used for direct block address encoding. Then, as shown previously in Figure 3.2, 3D surface reconstruction is performed one time without using a Look-Up Table (LUT) which requires an exhaustive search in the projector image (PRA with search), and another time by storing a Look-Up Table in the encoding stage and then, using it for obtaining the pixel correspondences in the decoding stage (PRA with LUT). The reconstructed point cloud and the number of obtained pixel correspondences are compared with the 3D reconstruction performed using direct block address encoding and shown in Figure 5.6 and Table 5.5. It can be seen that pseudo-random arrays result in a higher number of pixel correspondences and a denser point cloud and obtain pixel correspondences for almost all of the detected tags. This is because pseudo-random arrays leverage overlapping blocks and therefore, obtain more correspondences from border tags in comparison with direct bock address encoding. Next, we compare both methods in terms of their run time and memory requirements.

5.3.2 Run time and memory requirements

The most significant rationale of the proposed block address encoding method was to improve the complexity in terms of computational run time and memory requirements compared to that of pseudo-random array based methods, such as that in [21,23,26,27,29,



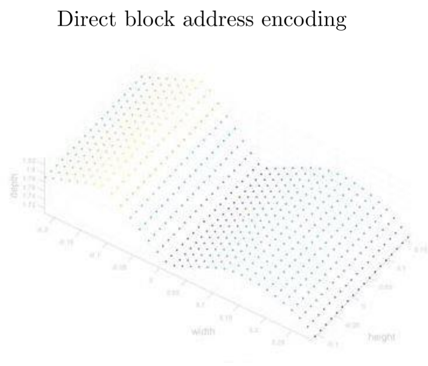


Figure 5.6: The reconstructed point clouds using pseudo-random arrays and direct block address encoding. Pseudo-random arrays results in a denser point cloud because of missing correspondences in the surface borders.

37]. Table 5.6 shows a comparison in terms of run times for the proposed method and that of the performed 3D reconstruction using pseudo-random arrays in the previous section. The results show that the proposed method offers an overall speed-up of 2-3 times faster than that of the pseudo-random array methods, which is attributed to the highly efficient encoding and decoding stages, since no uniqueness test/search is required during those stages. Indeed, the encoding/decoding portions are in total 26 times faster than the search based pseudo-random array method, where the overall increase is more modest because of the computational time associated with tag detection and classification which all methods have in common. Future work can be done to reduce the detection and classification time for faster implementations.

It is, of course, possible to use a Look-Up Table to reduce the computation time for pseudo-random array based approaches. As shown in Table 5.6, using a Look-Up Table increases the encoding run time in comparison with search based methods since it requires to create a Look-Up Table. However, the run time in the decoding stage decreases with a factor of 3 times. Nevertheless, for the size of problems considered here, based on high-resolution projectors, the associated Look-Up table would require a significant amount of memory, an amount which is impractical in many scenarios as shown in Table 5.7. In the case of using an alphabet size of $K = 6$, the required memory is 161MB as mentioned in Table 5.7. In contrast, the proposed direct block address encoding method has the advantage of requiring neither the computational time for a search nor the memory storage for a large Look-Up table. Also, the encoding/decoding portions of the proposed direct

Table 5.6: Comparing the average run time (seconds) during the 3D reconstruction stages for the proposed Direct Block Address Encoding scheme and the Pseudo-Random Array (PRA) method.

Processing Step / Requirements	PRA with search	PRA with LUT	Proposed
Encoding	3.28	3.30	0.03
Det. and Class.	2.68	2.68	2.68
Pixel Corr (Decoding).	2.01	0.66	0.17
Total	7.97	6.64	**2.88**
Speed-up over PRA with search			$\times$**2.77**
Speed-up over PRA with LUT			$\times$**2.31**

block address encoding are in total 20 times faster than the LUT based pseudo-random array method and at the same time does not require using a Look-Up Table to obtain camera-projector pixel correspondences.

5.4 Conclusion

In this chapter, a new method called direct block address encoding was proposed for more efficient structured light encoding and obtaining pixel correspondences by leveraging non-overlapping blocks which have the column and row addresses encoded inside of them. Different design parameters and their effects on the structured light system performance was discussed. Although the results suggest a significant improvement in the time complexity in encoding (tag label array construction), decoding (obtaining pixel correspondences), and memory requirements, the proposed method has the problem of existing unassociated tags in the surface borders which is a problem in those cases where having a more complete reconstruction is important. In the next chapter, the cause of having unassociated tags is discussed and methods are proposed to obtain pixel correspondences for those tags.

Table 5.7: The required memory of Look-Up Tables (LUT) for different ω and K for pseudo-random based methods. As the projector resolution increases, a higher alphabet size should be used that results in a higher memory requirement. As mentioned in Section 3.2.2, this LUT storage has a space complexity of $\mathcal{O}(K^{\omega^2})$.

Block size $\omega \times \omega$	Alphabet size K	LUT size
3×3	2	8KB
	4	4MB
	6	161MB
	8	2GB
5×5	2	536MB
	4	>8GB
	6	>8GB
	8	>8GB

Chapter 6

Second-level Correspondences

In the previous chapter, direct block address encoding was presented which reduces the computational and memory costs in the encoding and decoding stages of single-shot structured light 3D reconstruction. However, it is expected to have a number of detected tags in the camera image which have no corresponding projector tag. These tags are the set of unassociated tags and their existence is caused by one of the following reasons:

1. Incorrect Blocks: For a given recovered block in the camera image, $B'_{i,j}$, as mentioned in Section 5.1.1, an error detection scheme is applied to detect possible errors that may occur during tag recognition in the camera image. If an error in the decoding stage is detected and there is no ability for the decoder mechanism to correct this error, the tags within this incorrect block will not be associated.

2. Incomplete Blocks: At the boundaries of the surface, incomplete blocks may exist, resulting in tags that have no correspondences in the projector image.

As mentioned in Section 5.2.2, in case it is required to have a more complete 3D reconstruction using direct block address encoding, it is important to obtain correspondences for unassociated tags. In this chapter, a methodology is presented to obtain camera-projector pixel correspondences for a set of unassociated tags, also called second-level correspondences, by leveraging the already obtained pixel correspondences and detected and classified tags.

6.1 Problem formulation

In this section, the problem of obtaining second-level correspondences is formulated. Given a set of N_d detected tags in the camera image, a set of N_a tags are associated with a tag in the structured light image using the proposed direct block address encoding method in Chapter 5, called the set of associated tags. Also, a set of N_u tags exist that are not associated to a tag in the structured light image called the set of unassociated tags ($N_d = N_u + N_a$). In addition, the tag index in the camera image is denoted by $\bar{c}$ where $\bar{c} \in 1, 2, ..., N_d$. The set of associated tag indices in the camera image is shown by $\mathcal{A}^C$ and the set of unassociated tag indices in the camera image is denoted by $\mathcal{U}^C$. The same in the projector image (structured light image) is shown by $\mathcal{A}^P$ and $\mathcal{U}^P$. Moreover, a vector f, called the tag correspondence vector, of size N_d is defined which each of its elements include the corresponding tag indices in the structured light image (projector image). In other words, if $\bar{c} \in \mathcal{A}^C$, $f_{\bar{c}}$ (The $\bar{c}$th element of f) is equal to the corresponding structured light tag index and if $\bar{c} \in \mathcal{U}^C$, $f_{\bar{c}} = \text{NULL}$. In this chapter, the goal is to find the complete tag correspondence vector, including the correspondences for unassociated tags, $f_{\bar{c}}$ where $\bar{c} \in \mathcal{U}^C$.

The problem of obtaining second-level correspondences between a projector and camera pair can be formulated as an optimization problem. As mentioned earlier, the aim of solving the second level correspondences is to find $f_{\bar{c}} \in \mathcal{U}^P$ for $\bar{c} \in \mathcal{U}^C$. One approach to solve for the second-level correspondences is to define a loss function, $L(f, \mathcal{U}^C)$, which declares its confidence in candidate tag correspondence vector, f, including candidate correspondences for unassociated tags in the camera image. The loss function, $L(f, \mathcal{U}^C)$, can be written as a summation of confidences in individual candidates for each unassociated tag:

$$L(f, \mathcal{U}^C) = \sum_{\bar{c} \in \mathcal{U}^C} L'(f, \bar{c}) \tag{6.1}$$

where $L(f, \mathcal{U}^C)$ is the loss function related to the confidence in the candidate tag correspondences of unassociated tags, and $L'(f, \bar{c})$ is the confidence in the candidate correspondence of unassociated tag with index $\bar{c}$. In this section, we first formulate $L'(f, \bar{c})$ and then, the final loss function is easily obtained using Equation 6.1. To have a better understanding of how to formulate $L'(f, \bar{c})$, an unassociated tag in the camera image with index $\bar{c} \in \mathcal{U}^C$ is shown in Figure 6.1 (a). For each unassociated tag with index $\bar{c} \in \mathcal{U}^C$, the goal is to quantify the confidence in a candidate corresponding tag with index $f_{\bar{c}} \in \mathcal{U}^P$ in the projector image.

For this purpose, we leverage the tags in the neighborhood of the unassociated tag (Figure 6.1 (a)) and the neighborhood of its candidate corresponding tag in Figure 6.1 (c).

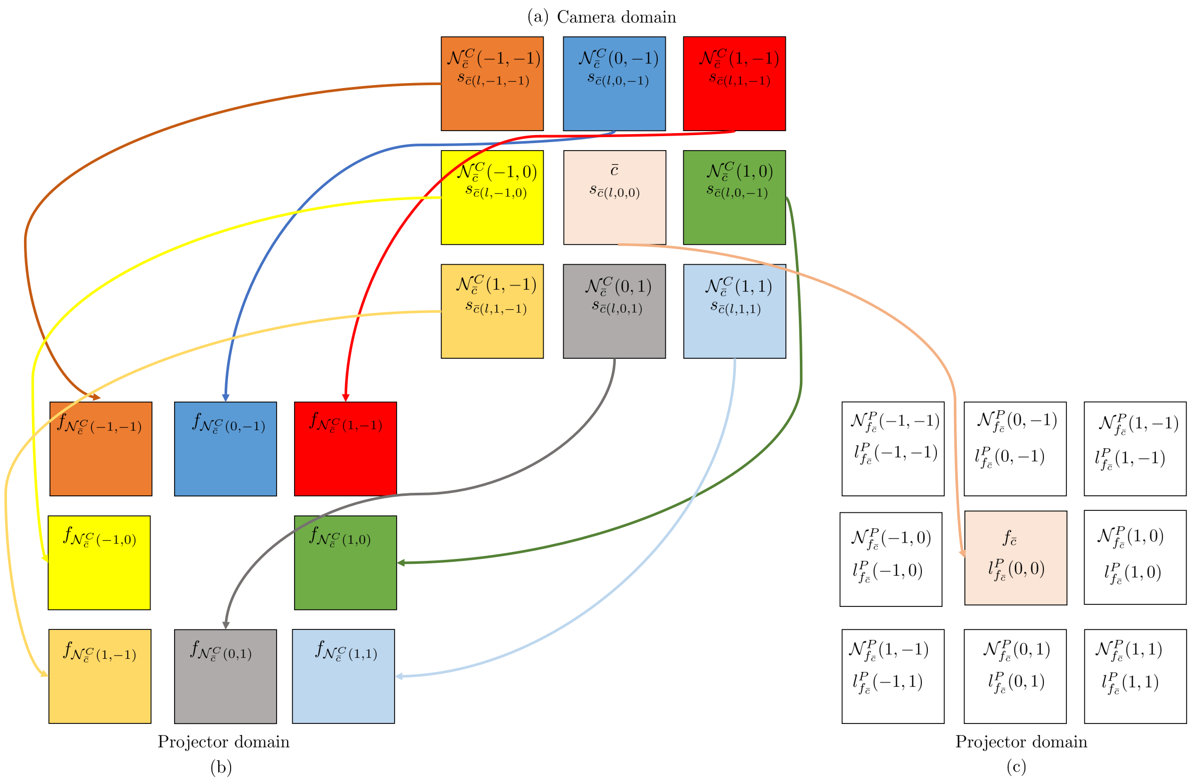

(a) Camera domain
$\mathcal{N}_{\bar{c}}^{C}(-1,-1)$ $s_{\bar{c}(l,-1,-1)}$
$\mathcal{N}_{\bar{c}}^{C}(0,-1)$ $s_{\bar{c}(l,0,-1)}$
$\mathcal{N}_{\bar{c}}^{C}(1,-1)$ $s_{\bar{c}(l,1,-1)}$
$\mathcal{N}_{\bar{c}}^{C}(-1,0)$ $s_{\bar{c}(l,-1,0)}$
$\bar{c}$ $s_{\bar{c}(l,0,0)}$
$\mathcal{N}_{\bar{c}}^{C}(1,0)$ $s_{\bar{c}(l,0,-1)}$
$\mathcal{N}_{\bar{c}}^{C}(1,-1)$ $s_{\bar{c}(l,1,-1)}$
$\mathcal{N}_{\bar{c}}^{C}(0,1)$ $s_{\bar{c}(l,0,1)}$
$\mathcal{N}_{\bar{c}}^{C}(1,1)$ $s_{\bar{c}(l,1,1)}$
$f_{\mathcal{N}_{\bar{c}}^{C}(-1,-1)}$
$f_{\mathcal{N}_{\bar{c}}^{C}(0,-1)}$
$f_{\mathcal{N}_{\bar{c}}^{C}(1,-1)}$
$f_{\mathcal{N}_{\bar{c}}^{C}(-1,0)}$
$f_{\mathcal{N}_{\bar{c}}^{C}(1,0)}$
$f_{\mathcal{N}_{\bar{c}}^{C}(1,-1)}$
$f_{\mathcal{N}_{\bar{c}}^{C}(0,1)}$
$f_{\mathcal{N}_{\bar{c}}^{C}(1,1)}$
Projector domain
(b)
$\mathcal{N}_{f_{\bar{c}}}^{P}(-1,-1)$ $l_{f_{\bar{c}}}^{P}(-1,-1)$
$\mathcal{N}_{f_{\bar{c}}}^{P}(0,-1)$ $l_{f_{\bar{c}}}^{P}(0,-1)$
$\mathcal{N}_{f_{\bar{c}}}^{P}(1,-1)$ $l_{f_{\bar{c}}}^{P}(1,-1)$
$\mathcal{N}_{f_{\bar{c}}}^{P}(-1,0)$ $l_{f_{\bar{c}}}^{P}(-1,0)$
$f_{\bar{c}}$ $l_{f_{\bar{c}}}^{P}(0,0)$
$\mathcal{N}_{f_{\bar{c}}}^{P}(1,0)$ $l_{f_{\bar{c}}}^{P}(1,0)$
$\mathcal{N}_{f_{\bar{c}}}^{P}(1,-1)$ $l_{f_{\bar{c}}}^{P}(-1,1)$
$\mathcal{N}_{f_{\bar{c}}}^{P}(0,1)$ $l_{f_{\bar{c}}}^{P}(0,1)$
$\mathcal{N}_{f_{\bar{c}}}^{P}(1,1)$ $l_{f_{\bar{c}}}^{P}(1,1)$
Projector domain
(c)

Figure 6.1: Second level correspondence problem formulation. **(a)** The unassociated tag with index $\bar{c}$ and tags in its neighborhood with index $\mathcal{N}_{\bar{c}}^{C}(i,j)$ and tag classification score $s_{\bar{c}(l,i,j)}$, in the camera image. **(b)** The corresponding tag indices in the structured light image, $f_{\mathcal{N}_{\bar{c}}^{C}(i,j)}$, for the tags in the neighborhood of the $\bar{c}$th unassociated tag. The squares with the same colors in (a) and (b) denote corresponding tags. **(c)** The predicted corresponding tag in the structured light image, $f_{\bar{c}}$ and the tags in its neighborhood, $\mathcal{N}_{f_{\bar{c}}}^{P}(i,j)$, with labels $l_{f_{\bar{c}}}^{P}(i,j)$. The squares with the same colors in (a) and (c) denote corresponding tags.

Specifically, a 3×3 neighborhood of tags is defined around each unassociated tag in the camera image as shown in Figure 6.1 (a). The indices of these tags is denoted by $\mathcal{N}_{\bar{c}}^{C}(i,j)$ which is defined as the index of the tag in the camera image with a distance of (i,j) tags from the unassociated tag. Similarly, $\mathcal{N}_{f_{\bar{c}}}^{P}(i,j)$, is defined as the index of the tag in the structured light image (Projector image) with a distance of (i,j) tags from tag with index $f_{\bar{c}}$ (Candidate corresponding tag) which is shown in Figure 6.1 (c).

Each of the tags in the neighborhood, $\mathcal{N}_{\bar{c}}^{C}(i,j)$, (Figure 6.1 (a)) are already classified and have a classification score for each of the possible predicted labels, which is denoted by $s_{\bar{c}}\big(l, i, j\big)$ where $l \in \{0, 1, 2, ..., K-1\}$, and K is the alphabet size. Furthermore, for some of the tags in the neighborhood, $\mathcal{N}_{\bar{c}}^{C}(i, j)$ (Figure 6.1 (a)), tag correspondences are already obtained in the structured light image which its index is denoted by $f_{\mathcal{N}_{\bar{c}}^{C}(i,j)}$ as shown in Figure 6.1 (b). If any of the tags in the neighborhood of the $\bar{c}$th camera tag in Figure 6.1 (a) belong to the set of unassociated tags, $f_{\mathcal{N}_{\bar{c}}^{C}(i,j)} = \text{NULL}$. Also, the neighborhood of the candidate corresponding tag, $\mathcal{N}_{f_{\bar{c}}}^{P}(i, j)$, is shown in Figure 6.1 (c) where the labels of the tags in this neighborhood are shown by $l_{f_{\bar{c}}}^{P}(i, j)$.

One term of $L'(f, \bar{c})$ can be obtained based on the fact that the label of the candidate corresponding tag and its neighbors in Figure 6.1 (c), $l_{f_{\bar{c}}}^{P}(i, j)$, should be as similar as possible to the labels of the tags in the unassociated tag neighborhood in Figure 6.1 (a). This is equivalent to $s_{\bar{c}}\big(l, i, j\big)$ having a maximum value when $l = l_{f_{\bar{c}}}^{P}(i, j)$. Therefore, for each pair of (i, j), $L'(f, \bar{c})$ should be minimized when $s_{\bar{c}}\big(l, i, j\big)$ is maximized. Therefore, the negative of the score, $-s_{\bar{c}}\big(l, i, j\big)$, should be minimized. Also, some of the tags in the neighborhood of the unassociated tag in the camera image may not exist. For example, the tags in the surface border do not have a complete 3×3 tags around them. In this case, $\mathcal{N}_{\bar{c}}^{C}(i, j) = \text{NULL}$, and therefore, there is no score to calculate. In order to model this, we can use a delta function $\delta(a, b)$ which is a function that is equal to 0 if $a \neq b$ and it is equal to 1 if $a = b$. Finally, the first term of the loss function, $L'_1(f, \bar{c})$ can be written as:

$$L'_{1}(f, \bar{c}) = -\underbrace{\sum_{i=-1}^{1}\sum_{j=-1}^{1}\Bigg(1 - \delta\Big(\mathcal{N}_{\bar{c}}^{C}(i, j), \text{NULL}\Big)\Bigg) \cdot s_{\bar{c}}\Big(l_{f_{\bar{c}}}^{P}(i, j), i, j\Big)}_{\text{Comparing the tag labels, (a) and (c)}} \tag{6.2}$$

In addition to matching labels, the corresponding tags of the unassociated tag neighbors, $f_{\mathcal{N}_{\bar{c}}^{C}(i,j)}$, shown in Figure 6.1 (b), should have the same tag index as the tags in the neighborhood of the candidate corresponding tag as shown in Figure 6.1 (c). Hence, the value of the defined loss function should decrease as more tag indices in Figure 6.1 (b) match with more tags in the neighborhood of the candidate tag correspondence in Figure 6.1 (c). Therefore, for each pair of (i, j), it is desirable that $\mathcal{N}_{f_{\bar{c}}}^{P}(i, j)$ and $f_{\mathcal{N}_{\bar{c}}^{C}(i,j)}$ are the same. This can be modeled using the δ function. Also, some of the tags in the unassociated tag neighborhood, $\mathcal{N}_{\bar{c}}^{C}(i, j)$, may not have correspondences in the structured light image. We also can use the delta function to model this to check if the correspondences for tags in the neighborhood of the unassociated tag exist or not. Therefore, second term of $L'_{2}(f, \bar{c})$ can be written as:

$$L'_2(f,\bar{c}) = \underbrace{-\sum_{i=-1}^{1}\sum_{j=-1}^{1}\left(1-\delta\left(f_{\mathcal{N}_{\bar{c}}^{C}(i,j)},\mathrm{NULL}\right)\right)\cdot\delta\left(\mathcal{N}_{f_{\bar{c}}}^{P}(i,j),f_{\mathcal{N}_{\bar{c}}^{C}(i,j)}\right)}_{\text{Comparing tag indices, (b) and (c)}} \tag{6.3}$$

Finally, the confidence for the candidate correspondence for the each unassociated tag , $f_{\bar{c}}$, can be measured as the summation of $L'_1(f,\bar{c})$ and $L'_2(f,\bar{c})$:

$$\begin{aligned} L'(f,\bar{c}) = & \underbrace{-\sum_{i=-1}^{1}\sum_{j=-1}^{1}\left(1-\delta\left(\mathcal{N}_{\bar{c}}^{C}(i,j),\mathrm{NULL}\right)\right)\cdot s_{\bar{c}}\left(l_{f_{\bar{c}}}^{P}(i,j),i,j\right)}_{\text{Comparing the tag labels, (a) and (c)}} \\ & \underbrace{-\sum_{i=-1}^{1}\sum_{j=-1}^{1}\left(1-\delta\left(f_{\mathcal{N}_{\bar{c}}^{C}(i,j)},\mathrm{NULL}\right)\right)\cdot\delta\left(\mathcal{N}_{f_{\bar{c}}}^{P}(i,j),f_{\mathcal{N}_{\bar{c}}^{C}(i,j)}\right)}_{\text{Comparing tag indices, (b) and (c)}} \end{aligned} \tag{6.4}$$

where the first term compares the tag labels in the Figure 6.1 (a) and (b). The second term in Equation 6.4 compares the tag indices in Figures 6.1 (b) and (c). In summary, the lower L' is, the higher is our confidence for declaring tag with index $f_{\bar{c}}$ in the structured light image as the corresponding tag. Finally, the loss function, $L(f,\mathcal{U}^C)$, can be obtained as mentioned in Equation 6.1. In order to obtain the second level correspondences for the set of unassociated tags, the lost function, $L(f,\mathcal{U}^C)$, defined in Equation 6.1 should be minimized. Different approaches can be used to minimize $L(f)$. In the following, three approaches to obtain the second level correspondences, are explained.

6.1.1 Optimization using simulated annealing

Simulated annealing [49, 50] is an optimization technique which takes inspiration from the concept of annealing in metallurgy and it is based on random sampling techniques [42, 43, 44]. Simulated annealing allows for more exploration in the search space by introducing a temperature parameter, θ_k, where k is the iteration number. The algorithm is designed such that it allows accepting inferior candidate points with a probability which is a function of the temperature in each iteration. In the early iterations, the temperature has a high value resulting in accepting more inferior points, allowing for more exploration in the search space (global search). As the algorithm proceeds, the temperature decreases using a temperature scheduler which is a descending function and changes the temperature in each iteration as a function of the iteration number. This makes the algorithm to search in a more local area as the temperature decreases. Some examples of possible temperature schedulers are $\theta_k = \frac{\theta_0}{\log(k+1)}$ and $\theta_k = \frac{\theta_0}{(k+1)}$ where θ_0 is an initial temperature.

This technique is predominantly used for finding global optimum for combinatorial optimization problems which consist of a configuration (solution) set and a loss function [50]. The configuration set is a set of m configurations of the target system that need to be optimized which is denoted by $f = (f_1, f_2, ..., f_m)$. The goal is to minimize a loss function $L(f) : f \longrightarrow \mathbb{R}$ and find an optimum set, f^*, which minimizes the loss function. When solving for the second level correspondences, f is the tag correspondence vector as introduced in Section 6.1. Also, m is equal to the number of detected tags in the camera image, N_d.

The pseudo-code of simulated annealing for obtaining the second level correspondences in shown is Algorithm 1. One important step in each iteration of this optimization is to sample a new configuration set at the neighborhood of the current configuration set, f_{curr}. This can be done using Gibbs sampling [51]. For this purpose, first one of the configurations with location $f' \in 1, 2, ..., m$ in the configuration set should be chosen at random in order to sample neighbors by changing its value. Then, as shown in Algorithm 1, the state value set, F, which includes all possible states for the f'th configuration is calculated. This set includes all configuration sets that can be created by replacing the f'th configuration with all possible alternative values. After calculating F and θ_k, Gibbs sampling should be performed in order to sample a new value for the configuration set from the state values. The pseudo-code of one simple Gibbs sampling is shown in Algorithm 2. Gibbs sampling takes a sample from the marginal probability distribution/state values (F) with a certain probability which is a function of the temperature in each iteration, θ_i. After sampling a new sample, the best configuration set is updated. The same procedure continues until the maximum number of iterations is reached.

6.1.2 Region growing approach by iterating over unassociated projector tags

The introduced simulated annealing method in the previous section optimizes the tag correspondences as a vector which is denoted by f. However, one can optimize each of the elements of the tag correspondence vector, f, sequentially rather than doing it simultaneously. By intuition, the unassocaited tags that have a lower distance to associated camera tags, have more information available in their neighborhood (more associated tags). Therefore, in order to solve for the second level correspondences, one can iterate over the set

Algorithm 1: Simulated Annealing with Gibbs sampling

```
   Input: Unassociated set of camera tags $\mathcal{U}^C$
   Input: Initial temperature $\theta_0$
   Input: Loss function $L()$
   Output: Optimum set $f^*$
1  Initialize configuration set $f_{\text{curr}} \leftarrow f_{\text{first}}$
2  Initial optimum set $f^* \leftarrow f_{\text{curr}}$
3  for $k$=1 to $iter_{max}$ do
4      Select a random configuration location   $f' \leftarrow \text{Random}()$
5      Obtain all possible state values   $F \leftarrow \text{StateValue}(f')$
6      Obtaining the temperature using the scheduler   $\theta_k \leftarrow \text{Scheduler}(k, \theta_0)$
7      Gibbs sampling   $f_{\text{curr}} \leftarrow \text{SingleGibbs}(\theta_k, F, f_{\text{curr}})$
8      Update $f^*$   if $L(f_{curr}, \mathcal{U}^C) < L(f^*, \mathcal{U}^C)$ then $f^* \leftarrow f_{\text{curr}}$ ;
9  end for
```

of unassociated tags, $\mathcal{U}^C$, and then, for each unassociated tag, the loss function introduced in Equation 6.4, $L'(f, \bar{c})$, can be used to evaluate each of the unassociated tags in the projector image, $\mathcal{U}^P$, as a candidate corresponding tag. Next, the unassociated tag in the projector image that results in the minimum value for $L^{'}(f, \bar{c})$ is taken as the corresponding tag. The same procedure is repeated for all of the unassociated tags in the camera image. A pseudo-code of this approach is shown in Algorithm 3. This approach is called region growing since the nearest unassociated tags are used as seeds points for obtaining the second level correspondences.

6.1.3 Region growing approach using the nearest associated tags

Similar to the approach proposed in Algorithm 3, we can first sort unassociated tags in the camera image based on having least distance to associated camera tags. Then, we iterate on the sorted unassociated tags, $\mathcal{U}^C_{\text{sort}}$, and find the index of its nearest associated camera tag, $\hat{c}$. Then, by localizing the nearest associated tag with respect to the unassociated tag, the tag correspondences can be easily obtained. For example, assume that the nearest associated tag is localized in the left hand side of the unassociated tag. As a result, the corresponding tag of the associated tag also locates in the left hand side of the corresponding tag of the unassociated tag. Since the correspondence of the associated tag is already known, the tag correspondence for the unassociated tag, $f_{\bar{c}}$, can be easily obtained. The pseudo-code of this approach is shown in Algorithm 4.

Algorithm 2: SingleGibbs Function

```
    Input: Temperature $\theta$
    Input: State Values $F$
    Input: Current Configuration set $f_{\text{curr}}$
    Output: Updated configuration set $f_{\text{curr}}$
1   $h \leftarrow []$
2   for $j \in$ size$(F)$ do
3       $h(j) \leftarrow \exp\Big(L\big(F(j)\big)/\theta\Big)$        Test all possible state values
4   end for
5   $s \leftarrow$ sum$(h)$                     Compute marginal partition function
6   $r \leftarrow$ UniformRandom()
7   $p \leftarrow 0$
8   for $j \in$ size$(F)$ do
9       $p \leftarrow p + h(j)/s$          Sample state $F(j)$ from marginal distribution
10      if   $\mathrm{r} \le p$   then   $\mathrm{f}_{\text{curr}} \leftarrow F(j)$ return
11  end for
```

6.2 Experiments

In this section, experiments are performed to compare different proposed methods for obtaining second level correspondences in Section 6.1. Second level correspondences are obtained using each of these methods and they are compared together in terms of correspondence accuracy and run time. Finally, the results are compared with methods based on pseudo-random arrays.

6.2.1 Comparing different methods of obtaining second level correspondences

In order to compare the proposed methods for obtaining second level correspondences in Section 6.1, we use a structured light image constructed using the proposed direct block address encoding method of Chapter 5. We perform 3D reconstruction of a curve-zigzag surface as shown in Figure 5.4. The used projector is WXGA (800 × 1280 pixels) and the camera has a resolution of 2048 × 2448 pixels. Also, the structured light image is constructed using a cell size of 24×24 pixels, block size of 3×3 and alphabet size of $K = 6$ as shown previously in Figure 5.3 (c).

Algorithm 3: Region growing by iterating over unassociated projector tags

```
   Input: Tag correspondence vector $f$
   Input: Unassociated tag indices in camera image $\mathcal{U}^C$
   Input: Associated tag indices in the camera image $\mathcal{A}^C$
   Input: Unassociated tag indices in the structured light image $\mathcal{U}^P$
   Output: Optimum tag correspondence vector $f$
1  $\mathcal{U}^C_{\text{sort}} \leftarrow \text{sort}(\mathcal{U}^C, \mathcal{A}^C)$ Sort camera unassociated tags
2  Loss $\leftarrow \{\}$
3  for $\bar{c} \in \mathcal{U}^C_{sort}$ do
4      for $\hat{c} \in \mathcal{U}^P$ do
5          $f'$=$f$ Test the loss function for each unassociated ptojector tag
6          $f'_{\bar{c}} = \hat{c}$
7          Loss $\leftarrow \{\text{Loss}, L'(f', \bar{c})\}$
8      end for
9      $f_{\bar{c}} = \{\hat{c} | L'(f', \bar{c}) = \min(\text{Loss})\}$ Assign tag correspondence
10     $\mathcal{U}^P \leftarrow$ Update($\mathcal{U}^P, f_{\bar{c}}$) Update the set of unassociated projector tags
11 end for
```

After creating the structured light image, the image is projected onto the target surface and a camera image is captured. Then, the proposed direct block address encoding is used to find the pixel correspondences between the projector and the camera and perform 3D reconstruction as shown in Figure 6.2 (a). As mentioned earlier in the beginning of this chapter, while performing 3D reconstruction using direct block address encoding, tags in the camera image may exist with no correspondences for two reasons. First, incomplete blocks at surface border; Second, blocks that have tag misclassifications. In the case of Figure 6.2 (a), the tags are all classified correctly and therefore the missing correspondences only lie in the surface borders. In order to simulate missing tag correspondences caused by possible tag misclassifications, we add tag misclassifications at random blocks. This will cause the decoding scheme to detect errors in these blocks and hence, no correspondences will be found for them. These misclassfications are added in different degrees: 0% (no blocks with miscalssifications), 10%, 20% and 50% percent of blocks in the camera image. The effect of this is shown in the reconstructed point clouds shown in Figure 6.2 (a)-(d).

The proposed methods in Section 6.1 are used to find the second level correspondences for each of the scenarios in Figure 6.2. Table 6.1 compares the performance of different methods for each test scenario. As shown in Table 6.1, all methods are able to find the correct correspondences for all unassociated tags in all test scenarios. However, the run time

Algorithm 4: Region growing using the nearest associated tags

```
  Input: Tag correspondence vector $f$
  Input: Unassociated tag indices in camera image $\mathcal{U}^C$
  Input: Associated tag indices in the camera image $\mathcal{A}^C$
  Output: Optimum set $f^*$
1 Sort camera unassociated tags $\mathcal{U}^C_{\text{sort}} \leftarrow \text{sort}(\mathcal{U}^C, \mathcal{A}^C)$
2 for $\bar{c} \in \mathcal{U}^C_{sort}$ do
3     Find the nearest associated tag $\hat{c} = \text{MinDistance}(\bar{c}, \mathcal{A}^C)$
4     Localizing the associated tag and find the tag correspondence
        $f_{\bar{c}} = \text{Localization}(\bar{c}, \hat{c}, f)$
5     Update the set of asscoiated camera tags $\mathcal{A}^C \leftarrow$ Update$(\mathcal{A}^C, f_{\bar{c}})$
6 end for
```

for simulated annealing is much higher than the other two methods. This run time may be reduced if other alternative temperature schedulers are used. However, any such reduction is not expected to be that significant to result in a lower run time than other methods since methods based on random sampling are normally computationally expensive. Also, as the results in Table 6.1 suggest, region growing based methods result in a reasonable run time, where region growing based on the nearest unassociated tags (RG-N), discussed in Algorithm 4, outperforms the iterative based region growing (RG-I), discussed in Algorithm 3. In the next section, the results of using direct block address encoding with region growing is compared with 3D reconstruction using pseudo-random arrays. In addition, the reconstructed 3D point cloud by obtaining the second level correspondences is shown in Figure 6.2 (e). It can be seen that by obtaining the second level correspondences, a denser and more complete reconstruction is obtained.

6.2.2 Direct block address encoding vs. pseudo-random arrays

In Section 5.3, we compared the performance of the proposed direct block address encoding with the pseudo-random arrays method [21,23,26,27,29,37,38]. The results in Tables 5.5 and 5.6 indicated that although direct block address encoding results in a significant complexity reduction and memory requirements, the number of obtained correspondences is lower than the number of obtained correspondences from using pseudo-random arrays which is caused by the existence of incomplete blocks at surface borders. Earlier, in this chapter, we introduced methods to remedy this problem by obtaining second level correspondences. Similar to Section 5.3, we compare direct block address encoding with the

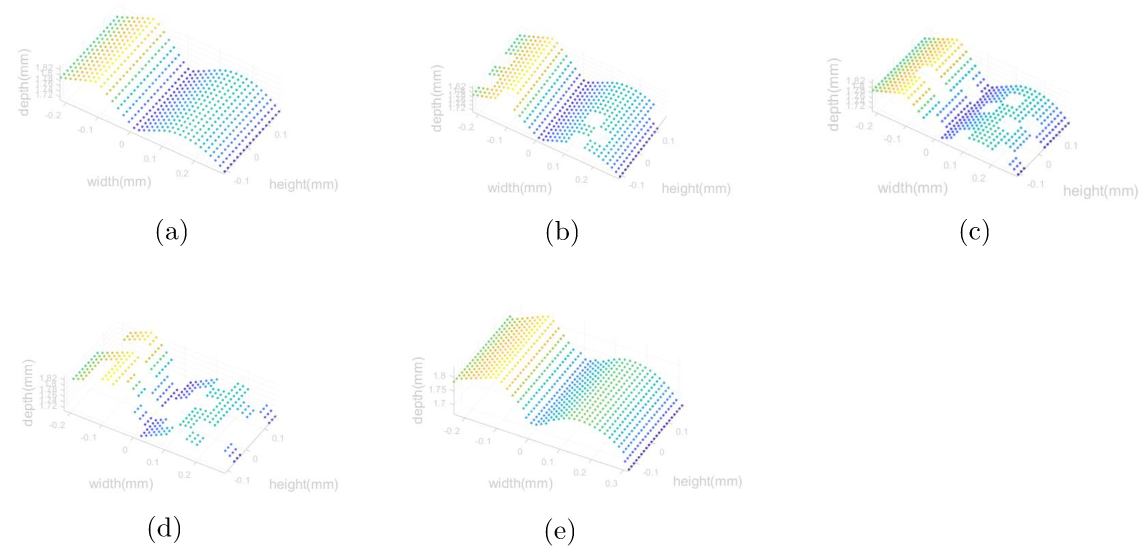


Figure 6.2: Reconstructed point clouds with missing tag correspondences caused by tag misclassifications and incomplete blocks in the surface border : (a) 0% (b) 10% (c) 20% (d) 50% (e) The 3D reconstructed point cloud by obtaining the second level correspondences.

Table 6.1: Comparison between the proposed methods, Simulated Annealing (SA), Region Growing by iterating over unassociated projector tags (RG-I) and Region Growing using nearest associated tags (RG-N) for obtaining second level correspondences in terms of the correspondence accuracy (correctly obtained second level correspondences) and run time for different test scenarios with different percentage of blocks with misclassifications, number of camera associated tags (N_a), number of unassociated camera tags (N_u).

Percentage of misclassified blocks	N_a	N_u	Accuracy			Run time		
			SA	RG-I	RG-N	SA	RG-I	RG-N
0%	594	140	140/140	140/140	140/140	3.0h	7.56s	0.68s
10%	558	176	176/176	176/176	176/176	5.5h	12.66s	0.87s
20%	477	257	257/257	257/257	257/257	7h	23.45s	1.12s
50%	288	446	603/603	603/603	603/603	14h	47.66s	2.15s

pseudo-random arrays method, this time by adding the second level correspondences obtained by region growing using nearest associated tags (RG-N) to our proposed method. The results of this comparison is shown in Tables 6.2 and 6.3. As shown in Table 6.2, the proposed method with region growing outperforms the pseudo-random method by 2-3 time speed-up in total and 5-6 time speed up in total for encoding /decoding stages. Also, the proposed method has the advantage of not requiring using a Look-Up Table as opposed to pseudo-random array method as shown previously in Table 5.7.

Table 6.2: Comparison between the Pseudo-Random Array (PRA) method and the proposed direct block address encoding in terms of the detected tags and the number of obtained pixel correspondences.

Method	Num. of detected tags	Num. of pixel corr.
PRA with search	734	730
PRA with LUT	734	730
Proposed without Second level correspondences	734	594
Proposed with RG	734	734

Table 6.3: Comparing the average run time (seconds) during the encoding / decoding stages for the proposed scheme and the pseudo-random array (PRA) method.

Processing Step / Requirements	PRA with search	PRA with LUT	Proposed *without* Second level correspondences	Proposed *with* RG
Encoding	3.28	3.30	0.03	0.03
Det. and Class.	2.68	2.68	2.68	2.68
Pixel Corr.	2.01	0.66	0.17	0.17
Region Growing	-	-	-	0.68
Total	7.97	6.64	2.88	3.56
Speed-up over PRA with search			$\times 2.77$	$\times 2.24$
Speed-up over PRA with LUT			$\times 2.31$	$\times 1.87$

Chapter 7

Conclusion

This chapter provides a summary of the work presented in this thesis along with a discussion of potential future directions.

7.1 Summary

This thesis aimed at proposing a new framework for performing single-shot structured light using tag embedding which is more robust to tag misclassifications and more efficient in terms of time and memory complexity in comparison with previously proposed pseudo-random arrays. For this purpose, first, in Chapter 4, it was investigated whether we can design a set of optimal tags that are distinguishable which also result in a comparatively high classification accuracy. However, challenges will rise when defining a loss function that is monotonic with actual tag classification accuracy. Then, in Chapter 5, direct block address encoding was proposed which as opposed to pseudo-random arrays, leverage non-overlapping blocks which have the column and rows addresses of the blocks encoded inside them. Also, the error detection schemes such as repetition code and check digit can be used to encode the row and column address to increase the robustness of the decoding scheme to tag misclassifications. In addition, the choice of different design parameters such as cell size, tag size, block size, projector size, alphabet size and their effects on the overall system performance was investigated. Next, in Chapter 6, several methods for obtaining the second level correspondences was proposed to solve the problem of unassociated tags in surface borders, and blocks with detected tag misclasssfications. The results both in Chapters 5 and 6 indicate that the proposed method reduces the computation of performing uniqueness tests during structured light image construction and the computation of performing an exhaustive search in the projector image or the need of using a look-up table to find pixel correspondences between a camera-projector pair. More specifically, the experimental results of performing 3D reconstruction on a target surface indicate that this method decrease the run time of performing encoding/decoding and the whole 3D reconstruction pipeline almost by 5-6 times and 2-3 times in comparison with pseudo-random arrays. The significance of the proposed scheme will be more when using high resolution projectors or multi-projectors since the computation for constructing the structured light image and the computation/memory requirements for obtaining the pixel correspondences will increase further.

7.2 Future work

Based on the proposed method, several new paths for research can be identified. First, in this thesis, we used the centroid of each tag in the camera and projector images as the feature point used to find the pixel correspondences. Future work can be done to increase the density of the pixel correspondences and the reconstructed point cloud by detecting more feature points such as the gird intersections and the tag corners in the camera and projector images. Second, the usage of the proposed scheme for performing projector-camera calibration can be investigated. The method in [18] leverages Gray coding multi-shot structured light to perform camera-projector calibration. However, the effect of using a single-shot method is unknown. Also, using a single-shot method for calibration can be useful since it reduces the projection time significantly, specifically when a hand-held camera is used. Finally, the application of the proposed scheme for multi-projector setups can be investigated where target surfaces with big areas are used.

References

[1] https://www.photonics.com/Products/Time-of-Flight_Camera/pr64381.

[2] http://pfrommer.us/stereovision.

[3] J. Geng, "Structured-light 3D surface imaging: A tutorial," *Adv. Opt. Photon.*, vol. 3, pp. 128–160, Jun 2011.

[4] J. Chen, T. Yamamoto, T. Aoyama, T. Takaki, and I. Ishii, "Real-time projection mapping using high-frame-rate structured light 3d vision," *SICE Journal of Control, Measurement, and System Integration*, vol. 8, no. 4, pp. 265–272, 2015.

[5] S. Zhang, "High-speed 3D shape measurement with structured light methods: A review," *Opt. and Lasers in Eng.*, vol. 106, pp. 119 – 131, 2018.

[6] J. Salvi, J. Pagès, and J. Batlle, "Pattern codification strategies in structured light systems," *Pattern Recognition*, vol. 37, no. 4, pp. 827 – 849, 2004.

[7] U. R. Dhond and J. Aggarwal, "Structure from stereo-a review," *IEEE Trans. Syst. Man Cybern.*, vol. 19, pp. 1489–1510, 1989.

[8] N. Lazaros, G. C. Sirakoulis, and A. Gasteratos, "Review of stereo vision algorithms: from software to hardware," *International Journal of Optomechatronics*, vol. 2, no. 4, pp. 435–462, 2008.

[9] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* USA: Cambridge University Press, 2 ed., 2003.

[10] M. Hansard, S. Lee, O. Choi, and R. P. Horaud, *Time-of-flight cameras: principles, methods and applications.* Springer Science & Business Media, 2012.

[11] J. Geng, "Structured-light 3D surface imaging: A tutorial," *Adv. Opt. Photon.*, vol. 3, no. 2, pp. 128–160, 2011.

[12] C. Siegl, M. Colaianni, M. Stamminger, and F. Bauer, "Adaptive stray-light compensation in dynamic multi-projection mapping," *Computational Visual Media*, vol. 3, no. 3, pp. 263–271, 2017.

[13] https://www.linkedin.com/company/christie-digital-systems/posts/?feedView=all.

[14] J. Posdamer and M. Altschuler, "Surface measurement by space-encoded projected beam systems," *Comput. Graphics and Image Process.*, vol. 18, no. 1, pp. 1 – 17, 1982.

[15] S. Inokuchi, "Range-imaging system for 3D object recognition," in *Proc. 7th Int. Conf. Pattern Recognition*, pp. 806–808, 1984.

[16] B. Carrihill and R. Hummel, "Experiments with the intensity ratio depth sensor," *Comput. Vision, Graphics, and Image Process.*, vol. 32, no. 3, pp. 337 – 358, 1985.

[17] V. Srinivasan, H. C. Liu, and M. Halioua, "Automated phase-measuring profilometry: A phase mapping approach," *Appl. Opt.*, vol. 24, pp. 185–188, Jan 1985.

[18] F. Li, H. Sekkati, J. Deglint, C. Scharfenberger, M. Lamm, D. A. Clausi, J. S. Zelek, and A. Wong, "Simultaneous projector-camera self-calibration for three-dimensional reconstruction and projection mapping," *IEEE Trans. on Comput. Imaging*, vol. 3, pp. 74–83, 2017.

[19] T. Etzion, "Constructions for perfect maps and pseudorandom arrays," *IEEE Trans. on Inform. Theory*, vol. 34, no. 5, pp. 1308–1316, 1988.

[20] C. J. Mitchell, "Aperiodic and semi-periodic perfect maps," *IEEE Trans. on Inform. Theory*, vol. 41, no. 1, pp. 88–95, 1995.

[21] R. A. Morano, C. Ozturk, R. Conn, S. Dubin, S. Zietz, and J. Nissano, "Structured light using pseudorandom codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 322–327, 1998.

[22] C. Tomasi, "Coded-array technique for obtaining depth and other position information of an observed object," May 1 2007. US Patent 7,212,663.

[23] C. Albitar, P. Graebling, and C. Doignon, "Design of a monochromatic pattern for a robust structured light coding," in *Proc. IEEE Int. Conf. on Image Process. (ICIP)*, pp. VI – 529–VI – 532, 2007.

[24] R. Sagawa, Y. Ota, Y. Yagi, R. Furukawa, N. Asada, and H. Kawasaki, "Dense 3D reconstruction method using a single pattern for fast moving object," in *Proc. IEEE Int. Conf. on Comput. Vision (ICCV)*, pp. 1779–1786, 2009.

[25] X. Maurice, P. Graebling, and C. Doignon, "Real-time structured light patterns coding with subperfect submaps," in *Proc. Real-Time Image and Video Process.*, vol. 7724, p. 77240E, SPIE, 2010.

[26] X. Maurice, P. Graebling, and C. Doignon, "Epipolar based structured light pattern design for 3-D reconstruction of moving surfaces," in *Proc. IEEE Int. Conf. on Robot. Autom.*, pp. 5301–5308, 2011.

[27] S. Tang, X. Zhang, Z. Song, L. Song, and H. Zeng, "Robust pattern decoding in shape-coded structured light," *Optics and Lasers in Eng.*, vol. 96, pp. 50 – 62, 2017.

[28] S. Tang, X. Zhang, Z. Song, H. Jiang, and L. Nie, "Three-dimensional surface reconstruction via a robust binary shape-coded structured light method," *Optical Eng.*, vol. 56, no. 1, pp. 1 – 14, 2017.

[29] K. Yang, Z. Ling, J. Li, X. Gao, L. Xie, and Z. Bai, "Color M-array shape reconstruction of using grid points and center points," in *Proc. Int. Conf. on Inform. Optics and Photonics (CIOP)*, vol. 11209, pp. 659 – 668, 2019.

[30] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[31] T. Fabry, D. Smeets, and D. Vandermeulen, *Surface representations for 3D face recognition.* 04 2010.

[32] X. Zhang, Z. Zhang, and W. Cheng, "Iterative projector calibration using multi-frequency phase-shifting method," in *2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pp. 1–6, 2015.

[33] D. Han, A. Chimienti, and G. Menga, "Accuracy improvement in structured light system calibration using plane based residual error compensation," in *European Workshop on Visual Information Processing (EUVIP)*, pp. 154–159, 2013.

[34] D. Moreno and G. Taubin, “Simple, accurate, and robust projector-camera calibration,” in *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*, pp. 464–471, 2012.

[35] Daesik Kim, Moonwook Ryu, and Sukhan Lee, “Antipodal gray codes for structured light,” in *2008 IEEE International Conference on Robotics and Automation*, pp. 3016–3021, 2008.

[36] A. Grami, *Introduction to Digital Communications.* Elsevier Science, 2015.

[37] Z. Song, S. Tang, F. Gu, C. Shi, and J. Feng, “Doe-based structured-light method for accurate 3d sensing,” *Optics and Lasers in Engineering*, vol. 120, pp. 21–30, 2019.

[38] S. Farsangi, M. A. Naiel, M. Lamm, and P. Fieguth, “Rectification based single-shot structured light for accurate and dense 3D reconstruction,” in *Proc. Conf. Vision and Intell. Systems (CVIS)*, 2020.

[39] K. Sadatsharifi, M. Naiel, M. Lamm, and P. Fieguth, “Locally adaptive thresholding for single-shot structured light patterns,” *Journal of Computational Vision and Imaging Systems*, vol. 6, pp. 1–3, Jan. 2021.

[40] C. Bishop, *Pattern Recognition and Machine Learning*, vol. 16, pp. 140–155. 01 2006.

[41] X. Maurice, P. Graebling, and C. Doignon, “Real-time structured light coding for adaptive patterns,” *J. Real-Time Image Process.*, p. 169–178, June 2013.

[42] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice.* New York: Springer New York, 2001.

[43] J. E. Handschin, “Monte carlo techniques for prediction and filtering of non-linear stochastic processes,” *Automatica*, vol. 6, p. 555–563, July 1970.

[44] N. de Freitas, C. Andrieu, P. Højen-Sørensen, M. Niranjan, and A. Gee, *Sequential Monte Carlo Methods for Neural Networks.* New York: Springer New York, 2001.

[45] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, pp. 273–297, 2004.

[46] M. Williard, “Introduction to redundancy coding,” *IEEE Transactions on Vehicular Technology*, vol. 27, no. 3, pp. 86–98, 1978.

[47] J. A. Gallian, “Error detection methods,” *ACM Comput. Surv.*, vol. 28, p. 504–517, Sept. 1996.

[48] Y. Chen, M. Niemenmaa, A. J. H. Vinck, and D. Gligoroski, “On some properties of a check digit system,” in *2012 IEEE International Symposium on Information Theory Proceedings*, pp. 1563–1567, 2012.

[49] S. Geman and D. Geman, “Stochastic relaxation, gibbs distributions, and the bayesian restoration of images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 721–741, 1984.

[50] P. Grabusts, J. Musatovs, and V. Golenkov, “The application of simulated annealing method for optimal route detection between objects,” *Procedia Computer Science*, vol. 149, pp. 95–101, 2019.

[51] C. Robert and G. Casella, “Monte carlo statistical methods,” in *Springer Texts in Statistics*, 2004.

[52] X. Maurice, P. Graebling, and C. Doignon, “A pattern framework driven by the hamming distance for structured light-based reconstruction with a single image,” in *Proc. Comput. Vision and Pattern Recog. (CVPR)*, pp. 2497–2504, 2011.

[53] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms.* Cambridge University Press, 2005.

[54] D. D. Hung, “3D scene modelling by sinusoid encoded illumination,” *Image and Vision Computing*, vol. 11, no. 5, pp. 251 – 256, 1993.

[55] X. Hu, M. Naiel, Z. Azimifar, I. Ben Daya, M. Lamm, and P. Fieguth, “Text enhancement in projected imagery,” *Journal of Comput. Vision and Imaging Systems*, vol. 4, p. 3, Dec. 2018.

[56] D. Bradley and G. Roth, “Adaptive thresholding using the integral image,” *J. Graphics Tools*, vol. 12, pp. 13–21, 01 2007.

[57] C. Luo, Z. Mo, Y. Zhou, and J. Wang, “Multi-view shape reconstruction based on m-array image decoding with adaptive local window,” in *2018 ICMA*, pp. 894–899, 2018.

[58] Weidong Hu, Mingying Gong, Yanhui Hong, Lifeng Sun, and Shiqiang Yang, “High-resolution 3d reconstruction for complex color scenes with structured light,” in *2014*

IEEE International Conference on Multimedia and Expo Workshops (ICMEW), pp. 1–6, 2014.

[59] Željko Santoši, I. Budak, V. Stojaković, M. Šokac, and ore Vukelić, "Evaluation of synthetically generated patterns for image-based 3d reconstruction of texture-less objects," *Measurement*, vol. 147, p. 106883, 2019.

[60] X. He and Q. Kemao, "A comparison of n-ary simple code and n-ary gray code phase unwrapping in high-speed fringe projection profilometry," *Optics and Lasers in Engineering*, vol. 128, p. 106046, 2020.

[61] M. Gupta and S. Nayar, "Micro Phase Shifting," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, Jun 2012.

[62] P. Gupta, "Gray code composite pattern structured light illumination," 2007.