Extraction of a Projected Pattern in a Single Image using Deep Learning

by

H M Ata-E-Rabbi

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

The appearance of an image projected by a projector onto an arbitrary surface has the potential to appear differently depending on several factors, e.g., the properties of the surface being projected on, the colors of the light projected by the projector and the colors and the optical properties of the surface. Recovering the original colors of the projected image and hence, the scene structure using a Structured Light (SL) system are challenged by the aforementioned factors. One of the key aspects of doing 3D reconstruction using SL systems is to detect the colors of the elements of the projected pattern in the image captured by the camera which contains important information needed for decoding the pattern to reconstruct the 3D shape of the object. However, this can be a challenging task when the colors of the projected lights interact with the colors of objects in the scene making the detection of the original colors difficult or when the surface absorbs most of the projected light making even the detection of the shapes of the elements of projected pattern difficult, if not impossible. Many off-the-shelf SL devices like the Microsoft Kinect use infrared (IR) cameras and projectors to get around the issue of color blending. But IR devices are not as readily available as normal RGB cameras and projectors which are usually the devices used in cost-efficient do-it-yourself custom SL systems. In this thesis, we explore the problem of pattern extraction for SL systems. We first focus on extracting the dots of the Kinect pattern, without any regard for the colors of the dots as color information is not used for decoding this pattern, by posing the problem of pattern extraction

as a pixel classification problem and propose a simple encoder-decoder based model that can surpass the classification accuracy of a straightforward image processing approach. Because obtaining ground truth to train such models is expensive and also because many RGB SL systems developed so far use many different kinds of patterns, we next propose a method, inspired by the ideas from recent advances in scene decomposition, to extract the elements of any pattern with original colors in an unsupervised manner at the cost of longer processing time. Our experiments show that this approach works best with dense patterns, but has trouble extracting fine details accurately required for sparse patterns like that of the Kinect's.

Acknowledgements

I am grateful to my supervisor, Professor Herb Yang, for his continuous guidance and assistance throughout the research and my exam committee. I would also like to thank Huawei and NSERC for partially funding this project.

Finally, I am forever grateful to my parents and sister for always being there for me and to baby Shayan who was a constant source of joy and happiness making the pandemic a little more bearable.

Contents

List of Tables

List of Figures

Glossary

Bidirectional Reflectance Distribution Function (BRDF)

Convolutional Neural Network (CNN)

infrared (IR)

Maximum a Posteriori (MAP)

Peak Signal-to-Noise Ratio (PSNR)

Red Green Blue (RGB)

Single-Shot Structured Light (SSSL)

Structural Similarity Index Measure (SSIM)

Structured Light (SL)

Chapter 1

Introduction

One of the effective methods of 3D reconstruction of a scene is to utilize a Structured Light (SL) system. An SL system uses a projector to project light onto an object. Based on the projected pattern, the 3D shape of the object can be determined. However, one major issue in an SL system is the extraction of the projected pattern, which may appear differently from the original pattern because of the interaction of the projected light with the scene object. SL systems are similar to stereo systems, but with one key difference - one of the two cameras in a typical stereo system is replaced by a projector in an SL system. The projector in an SL system projects a single pattern or a sequence of patterns on the scene being reconstructed and the camera in the system captures images of the scene for each of the projections. Some advantages of SL systems include their high accuracy and applicability to textureless surfaces. SL reconstruction algorithms analyze the spatial distortion of the pattern and also the change in the pattern over time if a sequence of patterns is projected to establish the correspondence between the pixels of the image captured by the camera and that of the image of the pattern projected by the projector. Once the correspondence is established, the SL system can, for all intents and purposes, be considered as a stereo system and standard stereo algorithms may be applied to do 3D reconstruction.

The success of an SL system largely depends on detecting and isolating the pattern that is projected onto the scene so that it can be matched with the actual image of the pattern used by the projector to establish correspondence.

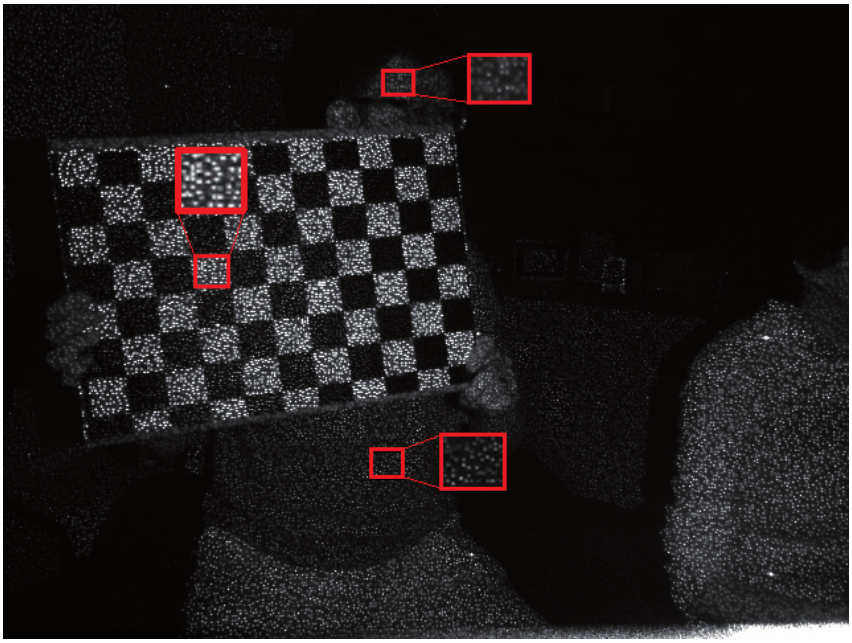
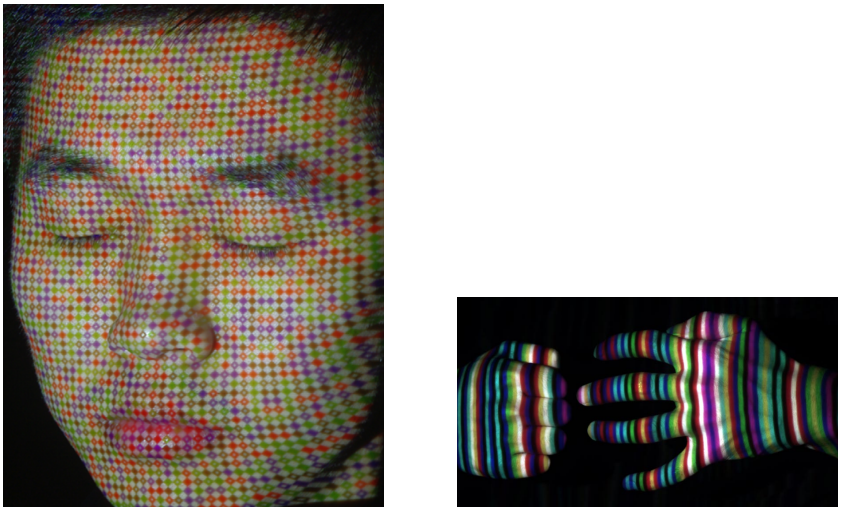


Figure 1.1: An image captured by the Microsoft Kinect [58]. The dots in the projected infrared pattern have minimal interaction with surfaces of different colors and materials and are clearly visible.

One of the challenges of detecting a pattern in a scene is to recover the colors of the pattern itself in the captured image because the colors of the light projected by the projector blend with the scene colors to produce composite colors that may be different from the colors that are projected. One of the most popular SL 3D reconstruction consumer devices, the Microsot Kinect [58], attempts to avoid this problem by using infrared (IR) projectors and cameras because the projected IR light does not interact with the scene colors making the pattern easily distinguishable. Figure 1.1 shows an image captured by the Microsoft Kinect. The dots in the projected pattern are clearly visible in the captured IR image.

While IR devices are an option, they are not as widespread and cheap as regular Red Green Blue (RGB) cameras and projectors. So, RGB cameras and projectors are still preferable for cost-efficient custom SL systems. There are previous works of using RGB patterns in SL systems, but these methods perform best in controlled environments where the projected patterns are clearly visible and easily distinguishable. Figure 1.2 shows some examples of



(a) (b)

Figure 1.2: Some examples of projected patterns on real scenes using RGB patterns. (a) is from [33] and (b) is from [57].

experimental setups of methods using RGB patterns where the illumination is carefully controlled to maximize the visibility of the projected patterns and surfaces being scanned cause minimal color-blending. While some of these methods do have some error-correction mechanism like assuming uniform reflectance, using local neighbourhood information, defining empirical methods, assuming constant reflectance in each of the three color bands etc. to account for small undetected pattern sections, complicated interactions between the the projected pattern and the scene colors still pose a problem [13] which is one of the main reasons why modern off-the-shelf depth sensing devices like the Microsoft Kinect use IR cameras and projectors. Successfully addressing this problem should make the use of normal RGB cameras and projectors more effective and feasible for cost-efficient do-it-yourself SL systems which is the main motivation of our work.

In this thesis, we investigate the problem of pattern extraction for RGB SL systems. Despite its importance, there is no existing published work that is focused on this problem. We first propose a method for extracting the Microsoft Kinect pattern by posing the problem of pattern extraction as a pixel classification task and training an encoder-decoder based model which can achieve better classification accuracy than a straightforward image processing

approach. Due to the lack of a publicly available dataset, we have developed a script to generate realistic physically-accurate synthetic data using the Mitsuba 2 renderer [41] to train the model.

Next, we go beyond the Kinect pattern, as there are many existing SL methods with their own unique patterns which may be more desirable in a do-it-yourself system due to the ease of pattern generation or the implementation of the decoding process, and propose a method for extracting any projected pattern from images. There have been many works in recent times such as [1], [5], [12], [17], [31] that decompose a scene into its constituent components - reflectance, shading, lighting, textures, albedo etc. Inspired by the success of these approaches, we treat the projection of a pattern in the scene image as a component of the scene and focus on extracting this additional component. While decomposing a scene into components is an ill-posed problem and usually requires labelled data for training, we show that with only two images of the same scene, one with the projector turned on and one off, we can extract the pattern component using an approach similar to that of the recent scene decomposition methods in an unsupervised manner which can overcome the difficult task of obtaining ground truth data for training the model in the first approach at the expense of a longer processing time for the optimization process to converge.

The organization of this thesis is as follows. We briefly discuss SL systems and various scene components and also present some foundational information as well as recent works done in these areas in chapter 2 and chapter 3 respectively. Then, chapter 4 describes the two new methods that we propose along with the experimental results. Finally, in chapter 5 we summarize our contributions and discuss possible future research opportunities.

Chapter 2

SL Systems

An SL system is used to do 3D reconstruction of a scene similar to that of a stereo camera system. However, unlike a stereo system, one of the cameras in an SL system is replaced by a projector which is used to project a single or a sequence of known patterns onto the scene. SL systems that project only one pattern are called single-shot systems while the ones that project a sequence of patterns are known as multi-shot systems. Figure 2.1 shows the setup of a typical SL system. SL systems have the advantages of being more accurate and better than normal stereo systems at scanning textureless surfaces as the projected pattern itself provides the texture. However, the projected pattern can also be a disadvantage when scanning textured or colored objects as the projected light will mix with the scene colors making the pattern detection process harder. An example of color-blending is shown in figure 2.2.

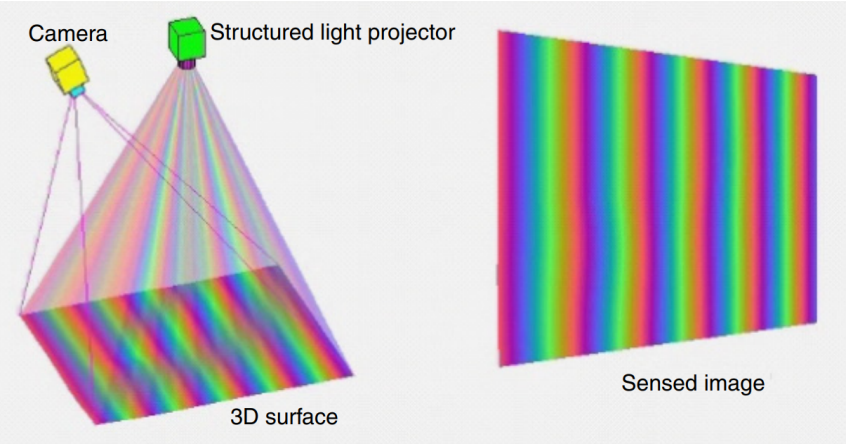


Figure 2.1: A typical SL system [18].

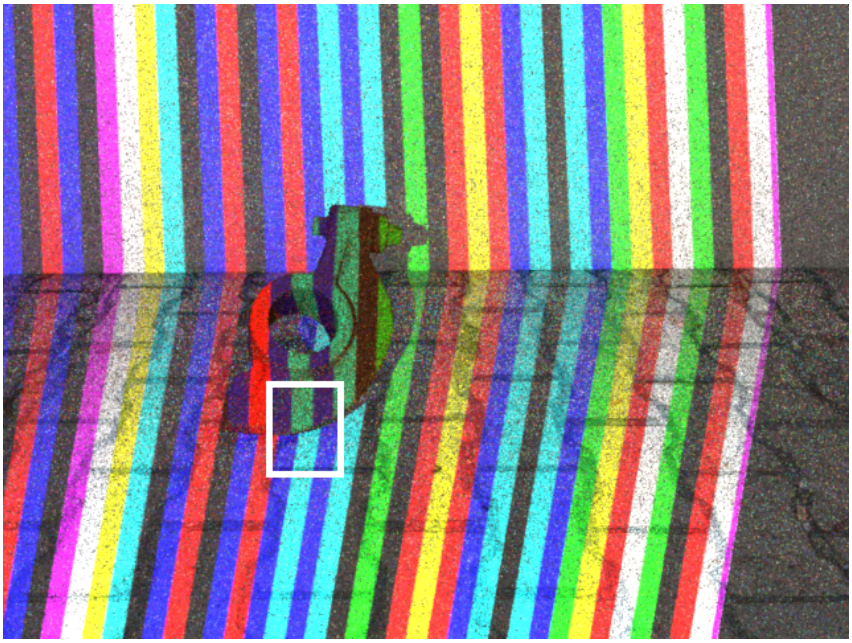


Figure 2.2: An example of color-blending. Inset shows an instance where the projected bright sky-blue colored stripe mixed with the color of the object resulting in an emerald color.

The camera in the system captures image(s) of the scene with the projection(s) from the projector which are then analyzed to find pixels in the projected pattern(s) that correspond to as many pixels in the captured images as possible. The naive approach of searching the entire projected image for each pixel in the captured image to find its correspondence is intractable especially in case of high-resolution images. The complexity of finding correspondences can be reduced to searching along just one row of pixels instead of a full 2D image, thus making the process much more efficient using the theories of epipolar geometry which is illustrated in figure 2.3. O_L and O_R are the optical centers of the two devices used in a stereo system, the two blue planes are the image planes corresponding to the two devices, X is a point of interest in the scene and X_L and X_R are the images of point X in the left and the right images, respectively. The line segment O_LO_R connecting the two optical centers intersects the left image plane at e_L and the right image plane at e_R which are called the epipolar points. The plane O_LXO_R intersects the two image planes at lines that are known as the epipolar lines, one of which

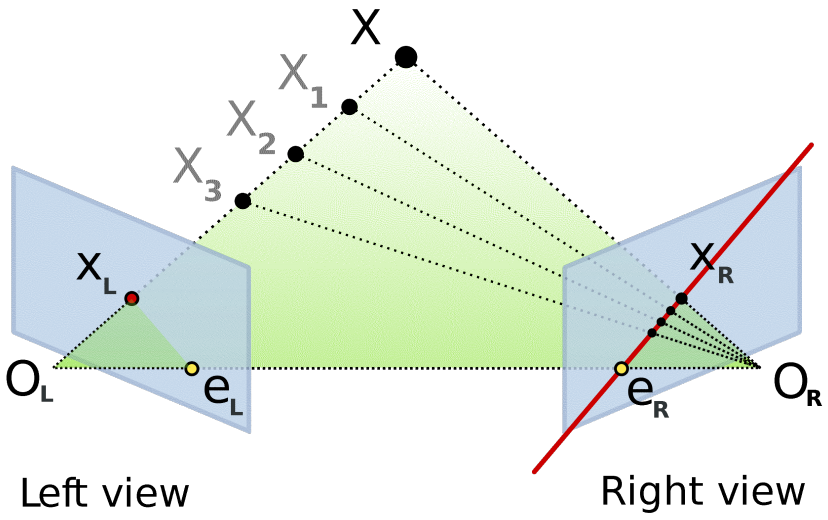


Figure 2.3: Epipolar geometry [43].

is shown as the red line in figure 2.3. Pixel X_R in the right image is called the corresponding pixel of pixel X_L in the left image because they are both images of the same scene point X. Epipolar lines constrain the search for corresponding pixels because, as can be seen from the image, the corresponding pixel X_R for pixel X_L lies on an epipolar line and vice versa.

The process of modifying two images so that the epipolar lines of the image planes "line up" with each other is called rectification. Rectification simplifies the task of finding the correspondence for a pixel in an image in a stereo system by ensuring that the corresponding pixel lies in the same row in the other image. This also simplifies the design of SL patterns as the same feature for encoding points in a scene can be repeated across rows without giving rise to any ambiguity. Many different algorithms for rectification have been proposed over the years [15], [16], [22]. Some rectification algorithms may require calibration which refers to the process of finding out the camera parameters that describe the camera's focal length, principal points, pose and location in the world among other camera features and in the case of stereo calibration, the relation between the two cameras in a stereo system. Again, calibration is a well-researched field with many different algorithms available [22]. In an SL system, the projector is modeled similar to a camera but

with the direction of the rays reversed [39] which makes traditional calibration algorithms applicable to SL systems as well.

Given a rectified SL system, the depth z of a pixel in the captured image can be computed from the displacement d in the pixel space between that pixel and its corresponding pixel in the projected image, the distance b, called baseline, between the center of the camera and that of the projector and the focal length f of the camera as,

$$z = \frac{bf}{d}.$$

2.1 Related Works

The design of SL patterns and their associated decoding methods require careful consideration of important aspects such as the colors used in the pattern, the number of patterns used to encode a scene, structure of the pattern elements etc. Here we discuss some works that demonstrate various design choices made for building SL systems over the years.

2.1.1 Grayscale Patterns

Some of the earliest works on SL 3D reconstruction [48], [49], [54] utilize binary coded black and white patterns to assign a code to each point in the scene. Because the pattern is binary, each point can be encoded and decoded in a robust manner, even in the case of color-shift due to noise or color-blending, by estimating the direct illumination component corresponding to the projector and global illumination component corresponding to ambient light of a scene using multiple patterns with thin stripes [40]. However, approaches using binary coded patterns require a sequence of images to be captured with each image containing a unique pattern making these approaches unsuitable for moving objects.

Single-Shot Structured Light (SSSL) systems benefit from the fact that they need only one image for doing 3D reconstruction. However, this advantage comes at the cost of requiring precise detection of the projected pattern. One

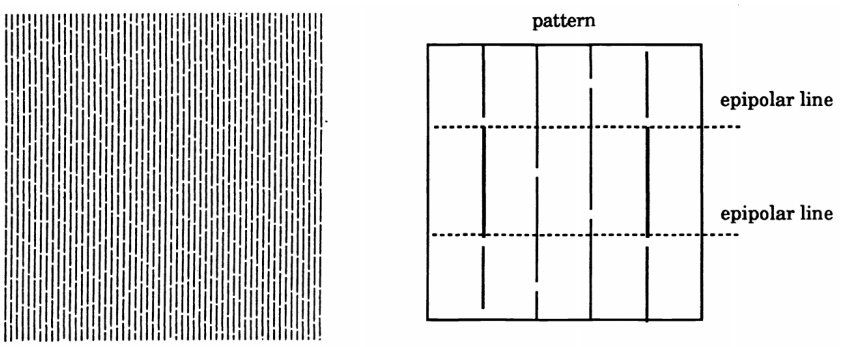


Figure 2.4: The image on the left is the slit pattern used in [38]. The right image shows examples of epipolar lines that have endpoints of multiple slits on them.

of the earliest works in this category is the method proposed by Maruyama and Abe [38] that uses a pattern with vertical slits with slit cut into many short line segments of random lengths. Stereo correspondence is established between the endpoints of slit-segments in the captured image and those in the projected image along the epipolar lines. Since two or more slit-segments may have their endpoints on the same epipolar line, not all epipolar lines can uniquely identify a single slit-segment. Figure 2.4 shows the pattern used in the method proposed by Maruyama and Abe and some epipolar lines containing endpoints of mutiple slit segments. The algorithm first establishes correspondences by matching slit-segments along those epipolar lines that have only one segment endpoint on them. Then, for other epipolar lines that have multiple slit-segments on them, either by design or because of false endpoint detection due to occlusion or noise, the method uses information from neighboring slit-segments to resolve ambiguities and propagate correspondences. While this approach allows the method to account for small local inconsistencies, it is not suitable for correcting errors when a large portion of the pattern is undetectable due to occlusion or color-blending because it only uses the nearest 6 neighbours of any particular endpoint, spanning a small local area, to resolve ambiguity. Moreover, the method uses simple binarization and edge-thinning techniques to detect the line segments which cannot recover slit-segments in the presence of color-blending.

Durdle *et al.* [11] proposed a method that uses three different intensity

levels - white (W), black (B) and gray (G) - to build a pattern as BWG, WBG, WGB, GWB, GBW, BGW where B=bbbb, W=wwww, G=ggggg, b=one pixel of intensity level 0, g=one pixel of intensity level 127, w=one pixel of intensity level 255. This sequence of 72 pixels is repeated vertically 9 times and the last 8 pixels are discarded to create a column of 640 pixels and then the column is repeated horizontally 480 times to create the final 640x480 pattern. The decoding process takes advantage of the uniqueness window property of the pattern (i.e. each 72 or more pixels in a column of the captured image must contain the selected sequence of colors that were used to build the projected pattern) and utilizes template matching to find the start of the pattern in each column of the captured image and then a second template matching is applied to find subcodes like BWG, WBG and GBW to refine the decoding. Because this method is designed to be applied on the human trunk, the authors only report results of experiments done on human skin where the pattern is easily detectable.

2.1.2 1D Color Patterns

The first method to use colored-stripes to build patterns was proposed by Boyer and Kak [8]. The pattern used in this method is formed using multiple subpatterns each of which contains N colored-stripes where the colors are chosen from a set of size γ. The authors propose a distance metric on the subpatterns defined as the number of same colored-stripes in the same position in two subpatterns and show that for a given threshold distance of d, the set of all possible subpatterns that are at least d apart from each other has an upper limit of $\gamma(\gamma-1)^{N-1-d}$ on its maximum size. An algorithm is proposed to generate all possible such patterns but the number of patterns found by the algorithm is much less than the upper bound. Subpatterns are then chosen from the generated set as required and concatenated to create the final pattern. The decoding of the algorithm follows a "crystal growing" scheme where for each subpattern in the projected pattern, the captured image is scanned from left to right and all matching positions are saved in subpattern-specific lists. These matches then act as the "seed crystals" where for each subpattern and

for each match for that subpattern, the captured image is scanned from the start of the match towards the left and the starting position of the match is decreased as long as there is a match between the projected pattern and the captured image. The same procedure is repeated from the endpoint of the matches towards the right. This is termed as "crystal growing". Finally, the authors propose an algorithm that does "crystal fitting" to produce the final matching result. The algorithm keeps two lists - one that contains all the crystals grown in the previous step sorted by their lengths and a second list that keeps the final chosen crystals sorted by their ordinal positions. The algorithm iterates until there is no more crystal left in the first list. In each iteration, the algorithm removes the first crystal in the first list, which has the largest length of all the crystals in the first list, and puts that in the second list and sorts the second list according to the ordinal positions of its crystals. It then removes all the crystals that are in the first list that are grown from the same seed as the chosen crystal. Finally, the crystals in the first list are trimmed so that none of them has any overlap with any of the crystals in the second list. The color assigned to each camera pixel is red if the red channel in the input image pixel has the maximum value, blue if the blue channel in the input image pixel has the maximum value or green otherwise. Such a simple scheme does not work for color-blending that can change pixel colors significantly.

Another early work that uses colored-stripes as the pattern is by Hugli and Maitre [24]. The sequence of colors chosen to build the coding scheme in the pattern is based on the theories of random sequences [47] which state that the maximum length of an N-unique sequence, a sequence for which all subsequences of length N are unique, containing elements from a set of Q colors is Q^N. However, for structured light patterns, in order to make stripe detection easier, no two neighbouring stripes should have the same color. This constraint reduces the maximum length of N-unique patterns that do not have the same color in two consecutive positions in the sequence to $Q(Q-1)^{N-1}$. Hugli and Maitre do not provide any proof of the existence for such N-unique patterns. But they use an algorithm to generate such patterns and verify their

validity for various values of N and Q and do not find any failure cases for the different values of N and Q that are tested. The decoding process begins by assigning an index to each of the detected color subsequences. The detected color subsequences are divided into two categories - disturbed and undisturbed. An undisturbed color subsequence is a subsequence whose elements have a true one-to-one mapping with only one subsequence in the projected pattern. A disturbed colored subsequence may also have a one-to-one mapping with a subsequence in the projected pattern due to occluded or undetected colors in the captured image or they may not have any mapping if they have invalid color sequences that do not appear in any subsequences in the projected pattern. The authors prove that an N-unique pattern is also an $N + 1$-unique pattern and use this principle to filter out as many disturbed subsequences as possible because longer subsequences have a lower probability of producing a valid color sequence. As the subsequences index applies to all of its constituent elements, each stripe in the pattern gets assigned at most N different candidate indices. To select an index from the candidate indices, a rule-based algorithm is suggested that first assigns to each stripe-index a pair of two values - the length of the segment indicated by the index and the number of elements that is assigned to that segment. Based on these two values, several different rules are defined to select a final index for each of the stripes which establishes the required correspondence with the projected pattern. While this approach has some mechanisms to account for occluded colors, the authors mention that they do not have any way of correcting incorrectly detected colors.

The method from Caspi *et al.* [10] proposes a scene-adaptive SL multi-shot system that uses colored pattern instead of multiple grayscale patterns as used in typical multi-shot systems to encode the points in a scene. This method has three parameters out of which the user has to specify at least two. The parameters are the number of columns in each pattern, the number of patterns to project and a noise immunity parameter. The method first takes an image of the scene without any projection and then another image is taken with pure white light projected from the projector. Then the method calculates the missing parameter if the user provided only two parameters and

generates the required number of patterns using a novel algorithm that uses the noise immunity parameter and the two images taken of the scene. These two images are also used to calculate the scene reflectivity and thus handle color-blending. While this is better than the other methods which require scenes with not much color variation, they still make the assumption that the reflectivity of the scene is approximately constant within each of the red, green and blue color bands which does not hold in complex scenes. The reflectivity calculation also requires careful calibration of the camera and the projector to find their color responses and in case of the projector the transformation from the projection instruction, the colors of the digital image that is sent to the projector, to the actual colors of the light that is transmitted by the projector.

Similar to the work by Hugli and Maitre, Zhang *et al.* [57] use a pattern with multiple colored-stripes. The authors argue that using a pattern with smoothly varying intensities is not ideal for SL systems because the color-blending between the projected pattern's color and the scene's color can affect the variations significantly and so they use piecewise-constant illumination pattern. However, unlike [24], the sequence of colors chosen for the stripes in this work is selected according to de Bruijn sequences [14]. First a representation of the projected pattern is defined where each edge between any two consecutive stripes is expressed as a list of 3 numbers - one number each for the three color channels. For each channel, if at the edge the intensity rises in the next stripe, it is assigned +1. If the intensity remains the same in both the stripes around an edge, it is assigned 0. Otherwise, the edge is assigned a value of -1. A representation like this allows for 27 unique edges but the edge (0,0,0) is discarded because it does not represent any valid transition between two stripes of different colors. Since the number of unique valid edges is only 26, for dense reconstruction one way to create a pattern is to repeat the same sequence of edges as much as necessary. This poses a problem, however, because when an object is larger than the width of the 26 edges, the same object will have multiple repetitions of the same pattern on it making the decoding process ambiguous as there are multiple valid matchings. To get around this issue, the authors resort to using de Bruijn sequences to take advantage of

their unique-window property which means that each subsequence of size N in these sequences is unique within the entire sequence. While creating the pattern, aside from the unique-window property, they also add an additional restriction that no two consecutive stripes can have the same color. Then an algorithm is presented to generate a sequence with these restrictions that first maps the colors to base-2 numbers and then cleverly applies the XOR operation. The pattern used for the experiments has 5 different colored-stripes and a unique-window size of 3 which results in a pattern with 125 stripes. While the edges in the projected pattern are represented by lists having the discrete values -1, 0 and 1, the edges in the captured images are instead assigned lists each containing three fractional values - the actual difference in the color values in each of the three channels between two consecutive patterns around each edge. To match these fractional values representing the edges in the captured image with the discrete values representing edges in the projected pattern, a scoring function with tunable parameters is proposed. Finally, a dynamic programming approach that utilizes the scoring function to do the actual matching is proposed. While the previous works used only a single-pass dynamic programming, the authors argue that a single-pass approach assumes monotonicity (i.e. the order of projected transitions and detected edges is the same) which is violated by occlusions and surface discontinuities. To address this issue, they propose a multi-pass algorithm that applies the original dynamic programming algorithm multiple times - each pass operating on the remaining unmatched edge-pairs from the previous pass until there is no more matches to be made. The method proposed in this work addresses mismatches due to occlusions but the authors mention that their method still has the restriction that the surface reflectance should not change the reflected colors too much. While the parameters of the scoring function can be tuned to address color-blending, the parameters have to be adjusted manually for each different scene making the process tedious.

2.1.3 2D Color Patterns

Another form of patterns used in structured light systems is a 2D grid pattern with intersection horizontal and vertical lines. One of the more recent works that makes use of such a pattern is by Ulusoy *et al.* [53] where the pattern is created using intersecting horizontal blue lines and vertical red lines. The horizontal and vertical lines are differentiated from each other using channel-wise color thresholding. The decoding process first detects stripes in the captured images and then uses them to construct a 2D graph encoding the position of the intersection points. Due to occlusion, discontinuities, noise or failure of image processing to detect the stripes, there may be multiple disconnected 2D graphs that are recovered from the captured images. The decoding process is applied to each connected graph independently of each other. The authors show that grid patterns have the advantage of reducing the correspondence-finding problem to finding just a single correspondence. Once a correspondence for an intersection point is determined, using the epipolar constraint it is possible to find the correspondences for all of its neighbouring intersection points. These neighbouring points can then propagate correspondences to their neighbours recursively until all of the intersection points in a connected graph have been assigned their correspondences. The initial correspondence for an intersection point in a graph can have multiple candidate correspondences because all the points in the projected image belonging to the same epipolar lines are viable options. This gives rise to multiple candidate solutions to the correspondence-finding problem for a particular connected graph. For an intersection point u in the captured image and one of its candidate corresponding points s in the projected image, the actual corresponding point in the projected image may not necessarily be located exactly at s due to noise in measurement, error in calibration or in image processing. Ulusoy *et al.* assume that the true corresponding point is one of the four nearest neighbours of s in the projected image and define a cost function as the minimum Euclidean distance of a nearest neighbor of s to the epipolar line of u. The solution that minimizes the sum of costs for all the intersection points is

chosen as the true solution. They show that a regularly spaced grid causes the function to be minimized to have many local minima close to the global minima. To avoid this issue, they propose creating an irregularly spaced grid where both the vertical and horizontal spacings between the stripes follow a De Bruijn sequence and show that this has a much better robustness to selecting the wrong solution. While this method can work even in the case of undetectable intersection points due to color-blending as it will simply skip processing problematic areas, its accuracy can be improved by some preprocessing step that addresses color-blending and extracts the projected pattern as much as possible so that more intersection points are available for processing.

Lin *et al.* [33] use both color and geometrical features to create a pattern. The pattern consists of rhombus-shaped elements arranged in a 2D grid in a way such that any two adjacent elements are touching each other at a single point. The authors justify the usage of rhombic elements by stating that rhombic elements arranged in the way that they propose allow for two adjacent elements to have the same color and the local symmetry property provides robustness to perspective projection and distortion due to surface discontinuity and also makes decoding the pattern easier. There are two types of elements - one type which has geometrical features inlaid in it (a smaller white rhombus in their proposed pattern) and another type which is a solid rhombus with uniform color. Each type of element can have one of 4 different colors giving rise to 8 different unique elements which make up the complete pattern. Using the principles of pseudorandom arrays [36], the authors use these 8 distinct elements to create a 65x63 grid pattern with a unique window property of 2x2 which allows successful decoding of any local area of the pattern if only 4 elements are successfully detected (i.e. have their types and colors determined). The decoding process then focuses on detecting the 2x2 windows in the pattern. It starts by first detecting all the intersection points of the adjacent rhombus-shaped elements in the pattern. The intersection points are detected by converting the image to grayscale and then applying a cross-mask to all the candidate intersection points in the image detected by some tradi-

tional key-point detection algorithm like SURF [6]. The authors empirically set a threshold value to filter out most of the candidate points. Finally, circular regions centered around the remaining candidate points are extracted and then each of these regions is assigned a score, taking advantage of the local symmetry property, by comparing with its own 180° flipped image. Again a threshold value is used to select the final intersection points. After the intersection points have been detected, a topological map of all the detected points is built to facilitate finding neighbouring points easier. The final step is to actually detect the types and colors of the two rhombic elements associated with each intersection points. The types of elements are determined by comparing border pixel values with central pixel values. The elements that have white rhombi inlaid in them will have larger pixel values around the central area in the corresponding grayscale image than the border pixels. This determines the type of the elements. Next a revised HSV model-based color representation method is proposed as well as empirical values for detecting each color in the new representation. Some constraints based on epipolar geometry are also proposed to provide additional robustness to noise. While the proposed color detection mechanism with the suggested empirical values works for surfaces with not much texture variation like the human skin, the values for detecting colors are scene-specific and require manual tuning for each new scene.

2.2 Summary

In this chapter, we have briefly discussed the basics of SL systems and also some works SL demonstrating various design techniques of these systems. Grayscale patterns can be reliably detected in the captured images if the number of shades of gray used to build a pattern is small (e.g. 2 or 3) and the shades are sufficiently distinct from each other. However, using a small number of different shades of gray severely restricts the amount of available information that can be used to encode a scene with just a single pattern. Thus, to perform 3D reconstruction with high spatial resolution, a sequence of grayscale patterns is required to encode scenes which is why these patterns

are more prevalent in multi-shot SL systems. Using different colors instead of just different shades of gray to build patterns allows for more information to be encoded in scenes with just a single pattern making a colored-pattern more suitable for single-shot SL systems. 1D stripe patterns have the advantage of being simpler to design and decode than their 2D grid counterparts. However, 2D patterns have the advantage of being able to utilize neighbourhood information to correct erroneous decoding to a degree that may occur due to occlusion or color-blending.

While many of these methods have some mechanism to address the color-blending issue, the simplifying assumptions they make can be violated in complex scenes and so it is still a major problem for most of the methods. Thus, methods for extracting patterns in the presence of color-blending can be beneficial for any RGB SL system.

Chapter 3

Scene Decomposition

The scene decomposition task involves breaking down an image of a scene into its constituent components. There are many ways a scene can be decomposed and different works focus on different types of decomposition. Popular choices of desired scene components are albedo, normals, depth, lighting and roughness. Albedo is a measure of diffuse reflection that is given by the ratio of reflected light to incident light. A surface's normal defines the direction the surface is facing which is commonly used to represent the geometry of a scene. Lighting describes the light sources in a scene which is vital to provide shading of a scene, on which different components of a scene depend. Lighting in a scene can be modeled by different representations such as environment maps [7], spherical harmonics [9], spherical Gaussian lobes [20] etc. Components like albedo and roughness are usually specified in functions known as the Bidirectional Reflectance Distribution Function (BRDF) to model diffuse and specular reflections. Figure 3.1 shows an example scene and its different components.

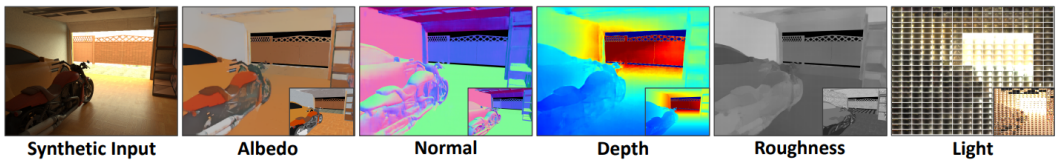


Figure 3.1: An example scene and its associated components [31].

3.1 Related Works

Earlier works in scene decomposition rely mostly on defining rules based on observing how different components behave under various scenarios that occur in the natural world and making simplifying assumptions that make optimization tractable. Later, with the advent of deep learning, the methods take advantage of physically-based rendering engines to create large datasets that accurately resemble real-world scenarios to train deep learning models. Here we give a brief overview of some of the methods proposed for scene decomposition. First, we present works that focus on simpler scenes consisting of just a single object which constitute some of the earlier works in this field. Next, we present works that are applicable to general scenes which represent the current research trend in scene decomposition.

3.1.1 Single-Object Scene Decomposition

Barrow and Tenenbaum [4] proposed one of the first works on scene decomposition in which they present a method to calculate depths, normals, incident lights and reflectances for all the points in a scene given an image of the scene. They make some assumptions about the nature of the scene to simplify the problem such as surfaces being mostly continuous and so having constant reflectances and continuous depths and normals except at edges corresponding to surface boundaries. They also assume that the incident illumination changes smoothly except at shadow or surface boundaries and that straight edges correspond to planar surfaces and ellipses correspond to circles viewed from oblique angles in man-made environments. They then create a simple experimental environment where these assumptions hold and based on the assumptions, rules are defined to classify image regions as various types of surfaces such as curved, planar, shadowed, non-shadowed etc. In the same vain, edges are classified into several types such as object boundaries, shadow boundaries etc. by checking the image intensities on both sides of the edges and across the edges and also by doing "tangency tests" to check whether reflectances derived at several points on an edge are consistent to provide in-

formation on occluding objects. They also compile all the various types of surfaces, intensities and edges to construct a catalog which is used to calculate the intrinsic properties of input images. For an input image, first the edges are detected and assigned types according to the catalog created from the simplified experimental environment. Then, intrinsic properties for the edges are assigned according to the catalog and finally these values are propagated throughout the image using continuity assumptions. This approach works for very simple scenes due to the oversimplification of the problem domain. The authors discuss some strategies to relax some of the assumptions made so far but note that the method is still not suitable for complex real-world scenes.

One of the more recent works on single-image scene decomposition is proposed by Lombardi and Nishino [35] where images of objects of known shape taken under natural unknown illumination are decomposed into scene reflectances and illuminations. They argue that the spatial layout and the color distribution of natural illuminations are structured and provide cues for reflectance estimation. The method represents the natural illumination as a 2D illumination map where the y-axis corresponds to the inclination angles and the x-axis corresponds to the azimuth angles in the spherical coordinate system. This representation allows flexible discretization of the scene illumination and thus controls the level of details. As Huang and Mumford [23] pointed out that the distribution of gradients of natural images has a heavy tail, Lombardi and Nishino [35] model the 2D illumination map as a hyper-Laplacian. To constrain the space of possible illuminations, they point out the fact that scene reflectance attenuates the incident illumination and empirically show that because the reflectance blurs the incident illumination, the entropy of the reflected radiance increases. This observation led them to constrain the estimated illumination to have minimum entropy so that the entropy of the reflected illumination matches that of the captured images. To model the space of reflectance, they make the assumption that an isotropic BRDF sufficiently represents the majority of different reflectances observed in the natural world and that the space of reflectance is modelled with the Directional Statistics BRDF [42] to make the model of reflectance adaptable to data. Then they

provide a Bayesian formulation for the image formation process relating the reflectance, the shape and the illumination in the scene allowing for the Maximum a Posteriori (MAP) optimization to jointly estimate the scene reflectance and illumination.

The method proposed by Oxholm and Nishino [45] jointly estimates the reflectance and the geometry of an object. They show that it is possible to extract the scene geometry and reflectance from the real-world with unconstrained, natural and known illumination in some cases and for other cases the problem space is not constrained enough to do geometry and reflectance extraction. The proposed method can leverage favourable lighting conditions to do accurate reflectance and geometry estimation if possible. Because this is an ill-posed problem, the authors adopt a probabilistic framework to optimize the geometry and the reflectance of a scene jointly by defining a Factorial Markov Random Field [27] with latent variables, associated with each other by their joint contribution on the observed appearance layer, for representing the reflectance and the object geometry. The usage of a probabilistic approach also helps integrate some prior knowledge in the optimization process and reduces the search space. These priors constrain orientations of surfaces according to the observed appearance of pixels, use occlusion boundaries to constrain surface orientations near boundary regions, make sure that the estimated gradient is consistent with the observed image gradients, penalize sharp changes in geometry and adopt a zero-mean multivariate Gaussian prior to model reflectance. The optimization is done using an expectation maximization framework where the reflectance and the geometry are estimated in turns.

Georgoulis *et al.* [19] propose a method that models the foreground object in an image as a highly imperfect mirror and attempt to estimate the environment map from multiple reflectance maps of the foreground object assuming that it is not perfectly diffuse. The method first uses previous works to extract multiple reflectance maps - one each for the 100 different possible materials that the method is designed to work with - and segments the foreground and the background in the input image. Then, a Convolutional Neural Network

(CNN) is used to estimate the environment map. The CNN has three components - an independent encoder to encode the background, several encoders that share parameters to encode the multiple reflectance maps and a decoder that takes the concatenated encodings from the encoders to produce the final environment map.

Another prominent method for scene decomposition is proposed by Barron and Malik [3] where a scene is decomposed into shapes, reflectances and illuminations. They define an image formation model using these three components and pose the estimation of these components as optimizing a function relating priors on these components under the constraint of the image formation model. They then provide several hand-crafted priors for each of these components. For reflectance, they assume piecewise constancy which is modeled by minimizing local variation of log-reflectance of images. They also minimize the number of colors required to form an image by minimizing the global entropy of log-reflectance. Finally, they give preference to some colors over others. For shapes, they assume that objects are smooth by minimizing the variation of mean curvature and by assuming that objects have the same probability of facing any direction which reduces to a flatness prior and that the normals near boundaries of objects are more likely to be facing towards the boundary edge. They also propose another optional prior on the shapes that models uncertain and noisy measurements by the measuring device. They use a spherical-harmonic model of illumination and the prior they impose on the illumination model is that it follows a multivariate Gaussian distribution which they find to be a good approximation. Finally, they propose an optimization technique generalized from multigrid methods [52] to take into account the priors they defined.

The method proposed by Li *et al.* [32] estimates the diffuse albedo, specular roughness, surface normal and depth of a scene from a single image of the scene taken under natural illumination and flash. A bright light source like the flash of a camera removes shadows and makes it easier to observe high frequency specularities. But unlike most previous works, the authors use a deep learning approach to achieve the task. They created a synthetic dataset

with objects of procedurally generated shapes and complex reflectance that resemble real world scenarios closely which they used to train their proposed deep learning model with cascaded structure that can reason about the inputs at multiple scales to extract important features. The first part of the proposed model is an encoder-decoder architecture that produces an initial estimate of the four desired scene components. The decoders share the features extracted by the encoder. Next, the authors attempt to estimate the global illumination of the scene. Because traditional radiosity methods are expensive and difficult to evaluate and are not always differentiable, they again use a deep learning approach to estimate global illumination. They use an analytical rendering layer to estimate direct illumination or the first-bounce image from the estimated scene components from before. Then they use another encoder-decoder model that takes the first bounce image along with the estimated scene components to produce the second-bounce image. They repeat this with a final encoder-decoder model to produce the third-bounce image. The total global illumination is then calculated as these three bounce images which the authors found to produce good enough approximations of the ground truth. Finally, they use a series of encoder-decoder models arranged in a cascade structure that each take the outputs of its previous model as inputs to produce successively more refined outputs.

3.1.2 General Scene Decomposition

Here we discuss some works that apply to general scenes that have occlusions and spatially-varying illuminations, not only scenes with single objects. One such work is by Barron and Malik [2] that takes a single RGB image along with a rough estimate of the scene depth measured with devices like the Microsoft Kinect and produces an improved depth map along with scene normals, reflectance, shading and illumination. The proposed method is an extension of their earlier work [3] to allow scene decomposition of natural scenes. As such, the optimization function [3] is extended so that the priors are defined over multiple depthmaps corresponding to multiple shapes instead of just the one shape as was done previously [3] and similarly the priors over illumination are

extended to multiple illuminations. The effect of different shapes and illuminations on each pixel is modeled by two probability maps - one for the shapes and one for the illuminations - that are used to extend the previously defined priors for one instance of shape and illumination to many. The probability maps are computed from a special embedding of the input RGB image that is computed from the eigenvectors of the normalized Laplacian of a graph corresponding to the input image in a manner similar to the work of Maji *et al.* [37]. The authors found that such embedding is superior to using plain superpixel embeddings in their application. The optimization process is similar to that of their earlier work [3].

A method for object insertion in a scene was proposed by Karsch *et al.* [26] where the user could simply drag and drop objects in images and the proposed method would place the objects correctly in 3D and apply appropriate lightings. To achieve this, the proposed method estimates the scene geometry, illumination, albedo and camera parameters using an algorithm that can estimate visible and invisible illuminations as well as a depth estimation algorithm that takes the scene geometry into consideration. The depth estimation algorithm adopts a label transfer approach that does not require any assumption on scene composition. They created a large database of RGB images that also include a depthmap for each scene. Given an input image, they use GIST [44] to extract features from the image and compare the features with the precomputed features of each of the images in their database. The top 7 matched candidate images with their depthmaps are selected from the database and warped using SIFT Flow [34] to establish pixel-to-pixel correspondences with the input image. Then an optimization function is defined that is minimized to produce the depthmap estimate for the input image. The optimization function includes a term that is the weighted sum of the difference between the warped depthmaps and the estimated depthmap as well as the difference between their gradients in the horizontal and the vertical direction to make sure that the estimated depthmap is similar to the warped candidate depthmaps. To infuse geometric constraints in the optimization process, they adopt ideas from the the work by Lee *et al.* [29] and include terms to enforce a Manhat-

tan scene structure, to constrain the orientations of planar surfaces and to encourage the estimation to be smoothly-varying. The method proposed by Lee *et al.* also estimates three vanishing points in a scene which the authors of this method use to estimate the camera parameters using well-established calibration algorithms [22]. The scene reflectance is estimated using the color Retinex algorithm proposed by Grosse *et al.* [21] which projects an input image into its brightness subspace and chromaticity subspace, which is the null space of the brightness subspace, and then thresholds these two images to assign the reflectance of each pixel as the brightness subspace projection or zero. For estimating visible illumination, they use the SUN360 dataset where they first segment the images into superpixels and extract features from each of the superpixels to train a binary classifier that classifies whether or not a superpixel is emitting or reflecting a significant amount of light. To estimate out-of-view illumination, they use the intuition that two images that appear similar should have similar illuminations. They annotate the SUN360 panaroma dataset [56] to include light positions and distances and propose an algorithm to match the input image with the images in the dataset to estimate out-of-view illumination.

The work proposed by Li and Snavely [30] attempts to address full scene decomposition by training deep learning models. Using the Mitsuba renderer [25], they create a physically-based synthetic dataset of 20,000 images with associated ground truth designed to resemble realistic images closely with high sample counts that takes 6 months to create more than 10000 images with a cluster of 10 machines. They also tonemap the high dynamic range images from the renderer to get low dynamic range images that resemble images of real cameras by finding the 90^{th} percentile intensity value and mapping that to 0.8 of the low dynamic range value and then applying gamma correction and clipping to make sure that the final values are within the range [0, 1]. The proposed model architecture is an encoder-decoder architecture with one encoder and two decoders - one for the shading and the other for the reflectance. The loss functions for training the model include the mean squared errors of the predicted reflectance and shading and the mean squared errors of their

gradients. These loss functions are applied to multiple scale outputs taken from the various stages of the decoders. The authors also make use of a reconstruction loss that is the mean squared error of the image reconstructed from shading and reflectance prediction. They show that the proposed model trained on synthetic data can surpass the accuracy of state-of-the art methods trained on real data and that augmenting the training with real data improves the performance further.

Other deep learning approaches for scene decomposition include the method by Sengupta *et al.* [50]. They propose a model which can be trained to render synthetic images with complex effects and thus can be fine-tuned for real data in a self-supervised manner using reconstruction loss after training the model on synthetic data. The proposed model has an encoder-decoder architecture that first estimates scene albedo, normals and environmental lighting. Albedo and normals predictions are supervised with ground truth data from the synthetic dataset. Ground truth environmental lighting cannot be used for supervision because only parts of the environment map are visible through doors or windows. To provide a form of weak supervision, they train a separate residual block based model that can approximate the ground truth environment lighting by first training the model to estimate direct indoor illumination supervised using the images generated by a direct renderer that uses ground truth albedo, normals and randomly sampled indoor lights. Then the model is fine-tuned using a reconstruction loss between raytraced images and images produced by the direct renderer from ground truth albedo and normals and lighting prediction from the model. Fine-tuning the model for real data can be done using a simple reconstruction loss between real images and the images generated by the direct renderer using the model outputs. However, a direct renderer cannot model complex effects such as inter-reflections, cast shadows or near-field illumination. To address this problem, they propose a separate encoder-decoder based model that learns to estimate the residual image which is added to the image from the direct renderer to generate the final image which then can be used in the reconstruction loss. This model is trained using synthetic data and then kept fixed when fine-tuning the whole model for

real data. Although this approach can model many complex effects found in real-world indoor scenes, it does not account for spatially varying lighting.

A method for general indoor scene decomposition that produces scene geometry, spatially varying lighting and complex spatially varing BRDF was proposed by Li *et al.* [31]. They designed a custom renderer that can render physically-based realistic images faster than currently available off-the-shelf renderer like Mitsuba and used that to generate a dataset containing more than 70,000 images and associated ground truth. This new dataset is created by taking an existing dataset [51] that does not have realistic textures or reflectance model and by improving them with semantically correct tileable texture and also by replacing the unrealistic specular reflectance with a more realistic spatially varying BRDF model. Their proposed model is a cascaded encoder-decoder architecture similar to that proposed by Li *et al.* [32] which has a shared encoder and 4 separate decoders - one each for estimating the scene albedo, roughness, normals and depth. The outputs of the first model is successively refined by the later model in the cascade. To estimate spatially varying lighting, they model the environment illumination as the sum of multiple spherical Gaussian lobes that can approximate lighting with a smaller number of parameters than environment maps or spherical harmonics. They propose a separate encoder-decoder model that takes the estimations from the first model as well as the original input image and produces the parameters of each of the Gaussian lobes. We use a similar architecture as that proposed in this work for extracting the projected pattern in an SL system. However, as we are not concerned with accurately estimating the various scene components such as normals, albedo, depths or lighting as doing so is intractable with an unsupervised extraction method that we desire, we make many simplifying assumptions in the proposed method that still produces reasonable results. If supervision with ground truth data were possible, our additions for handling projector lighting could easily be combined with ideas from this method making it a physically-accurate pattern extracting method.

3.2 Summary

In this chapter, we have discussed some of the different methods for scene decomposition which is the task of decomposing a scene into its constituent components like normal, albedo, reflectance, depth, lighting etc. Earlier works were mostly concerned with simpler scenes with single objects which was later improved upon to handle more general scenes. More recently, deep learning-based approaches have made it possible to avoid many simplifying assumptions and handcrafted rules used by previous optimization-based methods by learning complex scene features from large datasets. We use a model with the same architecture as MGNet0 [31], a recently proposed deep learning based supervised model, with the projector added as a separate component of the scene in addition to the other already existing scene components the model estimates and optimize it in an unsupervised manner. The modular nature of the model would make it straightforward to create a physically-accurate pattern extraction method by training the model in a supervised manner if training data were available.

Chapter 4

Extraction of SL Patterns using Deep Learning

In this chapter, we present two new methods for extracting SL patterns from RGB images. Section 4.1 details a method for extracting dot patterns used in many commercial IR-based depth sensing devices. Then in section 4.2, we describe a method that can extract any RGB pattern.

4.1 Extracting the Kinect Pattern

One of the most widely-used patterns for SSSL is the sparse dot pattern shown in figure 4.1 which is used in Microsoft Kinect family of devices. All of the commercial mainstream devices work with IR images where the limited interaction of the projected IR light with the surface textures in the scene makes it relatively easy to detect the projected pattern's dots in captured images of IR cameras.

But when the pattern projected is composed of colors in the visible spectrum (e.g. projected with a normal off-the-shelf RGB projector), the projected light interacts with the scene colors which can make detecting the projected pattern in the captured RGB images quite challenging in some scenarios. In this section, we demonstrate that a simple encoder-decoder based pixel classification model can be used to detect such dot patterns in RGB images with high accuracy. As such, applying the proposed model as a preprocessing step can allow building Kinect-like SL systems with low-cost RGB cameras and

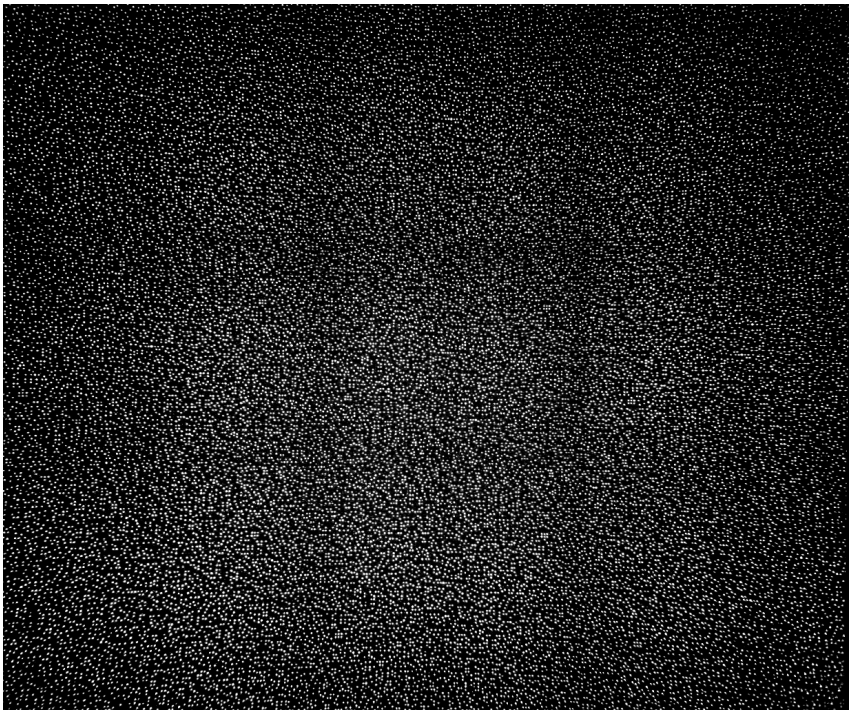


Figure 4.1: The original Microsoft Kinect pattern.

projectors.

4.1.1 Model Architecture

The pixel classifier model has a U-Net architecture comprised of an encoder, followed by a bottleneck and finally a decoder that is connected with the encoder by layer-wise skip connections. More specifically, for two inputs I_{off} and I_{on} that are the images of the same scene with the projector turned off and on respectively, an encoder E, a bottleneck B and a decoder D, we have,

$$
\begin{aligned}
E_{out} &= E(I_{off}, I_{on}; \theta_E), \\
B_{out} &= B(E_{out}; \theta_B), \\
D_{out} &= D(B_{out}; \theta_D),
\end{aligned}
$$

where E_{out}, B_{out}, D_{out} are the outputs of the encoder, the bottleneck and the decoder and θ_E, θ_B and θ_D are their parameters, respectively. Figure 4.2 visualizes the model architecture.

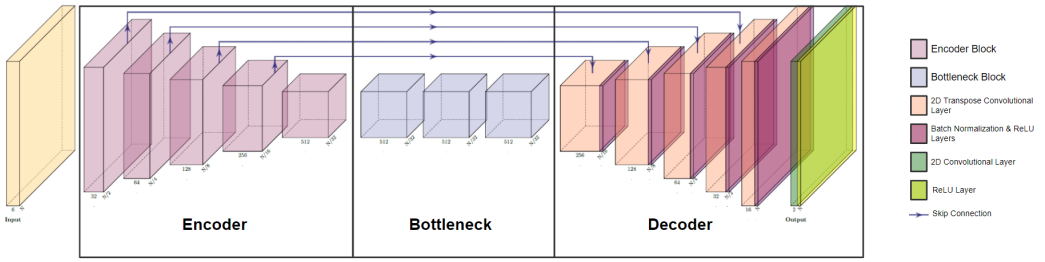


Figure 4.2: The U-Net architecture for extracting the dot pattern.

Encoder

The encoder consists of five residual blocks that each halves the spatial resolution of its input. The first block takes as input two images - one image of the scene with the projector turned off and one image of the same scene from the same viewpoint with the projector turned on projecting the Kinect pattern. It outputs 32 channels while the subsequent blocks double the number of output channels from their respective previous blocks. Each of these residual blocks contains two convolutional layers.

The first convolutional layer has a 3x3 kernel, a stride of 2 to half the spatial resolution of the input and the same number of output channels as the block's desired number of output channels. It is followed by a batch normalization layer and a ReLU activation layer. The second convolutional layer has a 3x3 kernel, a stride size of 1 to preserve spatial resolution and the same number of output channels as its input. It is again followed by a batch normalization layer and a linear activation layer.

The input is passed through the first convolutional layer the output of which is then passed through the second convolutional layer. The original input and the output of the second convolutional layer are then added together to produce the final output of the whole block. Figure 4.3 shows an encoder block.

Bottleneck

The bottleneck takes the output of the encoder and performs 2D convolutions while preserving the spatial resolution and the number of channels.

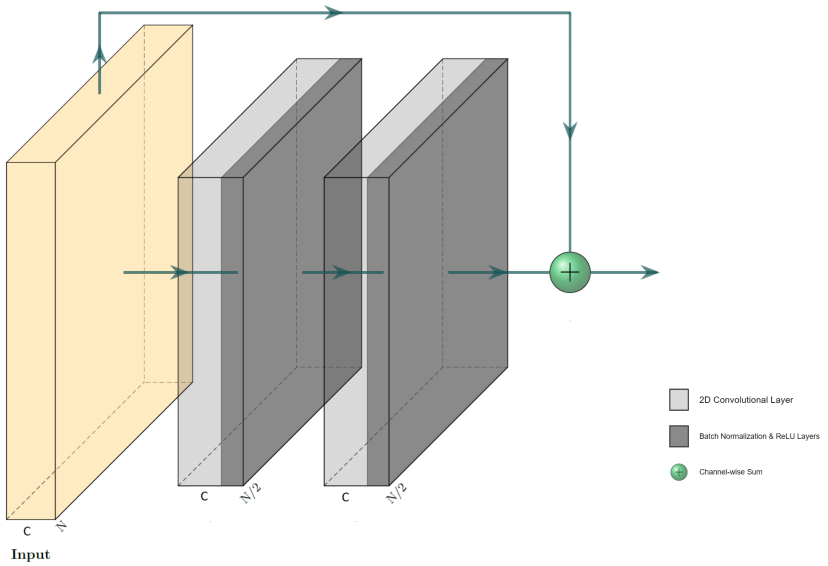


Figure 4.3: The encoder block.

It consists of three residual blocks each of which contains two convolutional layers. These layers are similar to the two layers of the residual blocks in the encoder, however both of these layers have a stride of 1 to preserve spatial resolution. Also, unlike the layers in the encoder's residual blocks, the layers in the second and the third bottleneck blocks perform 3-dilated and 5-dilated convolutions respectively to increase the receptive field of the model. Figure 4.4 shows a decoder block.

Decoder

The decoder takes the output of the bottleneck and upsamples the output back to the original spatial resolution through a series of upsampling blocks. It consists of five upsampling blocks, one each for the five downsampling residual blocks in the encoder, that doubles the spatial resolution and halves the number of channels and one final block to produce the final classification output.

Each upsampling block consists of a transpose convolutional layer, followed by a batch normalization layer and a ReLU activation layer. The output of each upsampling block is concatenated with the output of its corresponding downsampling block in the encoder and then fed to the next upsampling block. The final block takes the output of the last upsampling block and passes it

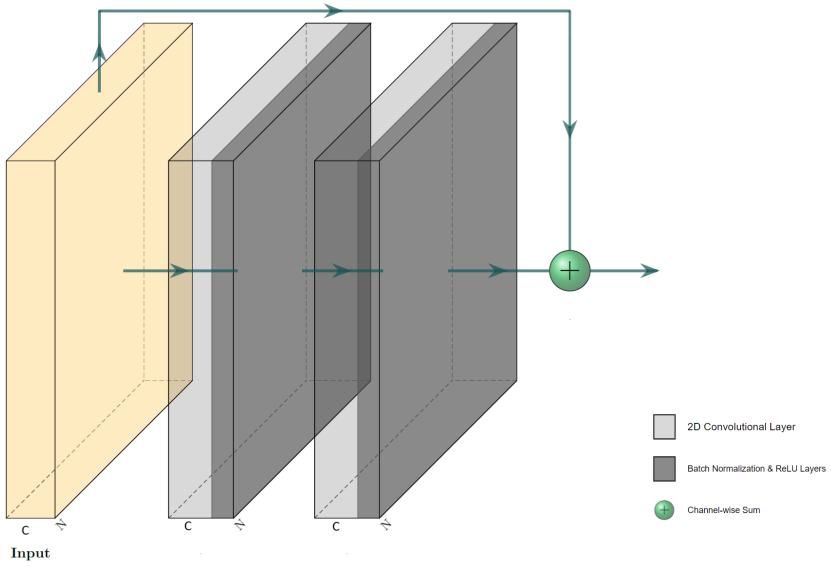


Figure 4.4: The bottleneck block.

though a 2D convolutional layer with a kernel size of 3x3, stride of 1 and a channel count of 2. This output is then passed through a batch normalization layer and a ReLU activation layer to produce classification logits.

4.1.2 Dataset

We have created a synthetic dataset using the Mitsuba 2 renderer [41] to train the proposed model. The dataset consists of images of scenes where objects are placed in an enclosed room in random poses and orientations. The textures on the walls of the room and the objects are randomly selected from a pool of 56 different textures and the object models are selected from a pool of 31 different models. The room has a ceiling light whose position and brightness are randomised for each image render. Figure 4.5 shows some of the textures and object models used for generating the dataset.

Instead of creating a dataset with a fixed number of images, we take advantage of the Mitsuba 2 renderer's capability of seamless integration with PyTorch [46]. During training, the renderer is queried in real-time to generate image renders of randomised scenes along with the corresponding ground truths. To generate each render, an SL rig, consisting of a projector on the left and a camera on the right, is placed in a random pose in the scene pointing

Figure 4.5: Some textures (top row) and object models (bottom row) used to create the dataset.

at the objects and then two images are captured with the camera - one with the projector turned off and one with the projector turned on projecting the Kinect dot pattern. For each render, each of the dots in the projected pattern is assigned a random color. Using the associated ground truth depth-map, we calculate the positions of the dots in the images captured by the camera to classify pixels in two categories - pattern pixels corresponding to dots in the projected pattern and background pixels. The baseline of the SL system is also randomised for each render to be in the 5cm to 20cm range. We use 39 of the 56 textures and 25 of 31 object models while generating the training data and use the rest for generating the test data. We purposefully keep the number of samples per pixel parameter of the Mitsuba 2 renderer to a very low value of 8 to achieve fast rendering as well as to generate noisy images to simulate noise that is likely to be present in real world images, especially in low-light conditions.

4.1.3 Training

The model is trained in an end-to-end manner. We use the Adam optimizer [28] with a starting learning rate of 0.001 and train the model on a total of 6000 images. After 2000 and 4000 images, we set the learning rate to 0.0001 and 0.00001 respectively. On a machine with 32GB RAM, Intel Core i7 6900k CPU and an Nvidia Titan RTX GPU, the training, including rendering all the data required to train the model, takes about 5 hours. The model classifies

each pixel as belonging either to a dot or to a background pixel. We use the binary cross-entropy loss as the loss function to train the model. More specifically, for the ground truth label y_i of pixel i, which is 1 for a dot pixel and 0 otherwise, and the model-predicted probability $p(y_i)$ of pixel i being a dot pixel, the loss L is calculated by taking the average across all N pixels as,

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(p(y_i)) + (1 - y_i)\log(1 - p(y_i)).$$

The training updates the model parameters θ_E, θ_B and θ_D using gradient descent by calculating the gradients $\frac{\partial L}{\partial \theta_E}$, $\frac{\partial L}{\partial \theta_B}$ and $\frac{\partial L}{\partial \theta_D}$ with the backpropagation technique.

4.1.4 Results

We generate 100 images using our test data generation method with associated ground truths to measure the effectiveness of the proposed method. The classification accuracy of our trained model on this test set is 96.3%. We also calculate the accuracy of a straightforward image processing approach where we first subtract the image of the scene with the projector off from the image of the same scene with the projector on and then apply thresholding to produce a binary image. A linear search on threshold values was performed for the 100 generated images and the threshold value of 40 was found out to have the highest average accuracy which was selected to threshold all the images. The classification accuracy of this approach on the same test set is 93.1%. Figure 4.6 shows some of the classification results of both methods.

Figure 4.7 shows the accuracy map of the proposed method as well as the image processing method. The black pixels in the accuracy maps signify incorrect classification. The proposed method performs better in challenging areas where the visibility of the dots is low (e.g. around the junction of the wall and the floor or the long dark stripes on the floor that do not reflect the projected dots very well) evidenced by lower amount of black pixels implying that the proposed method has learnt to reason about the structure of the pattern to provide better classification results.

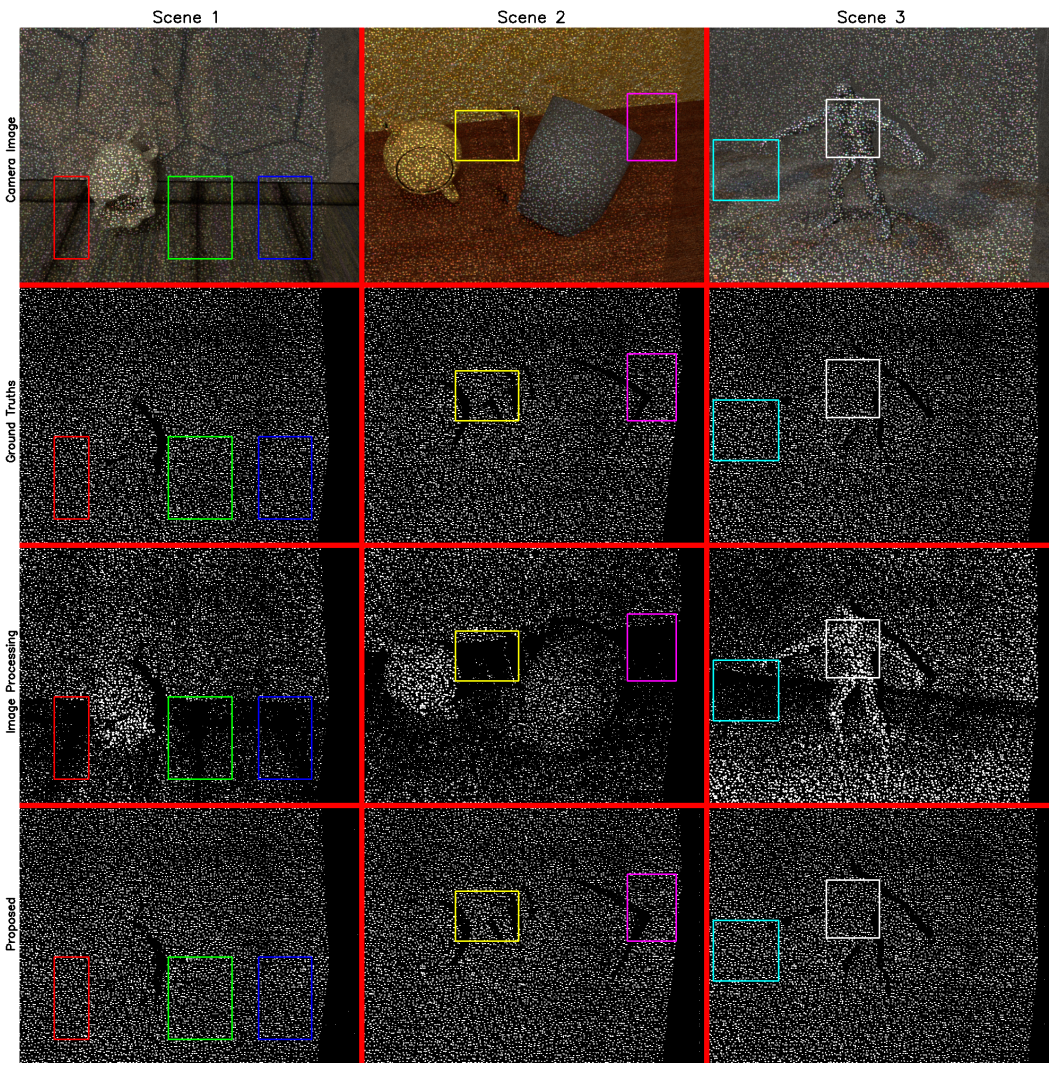
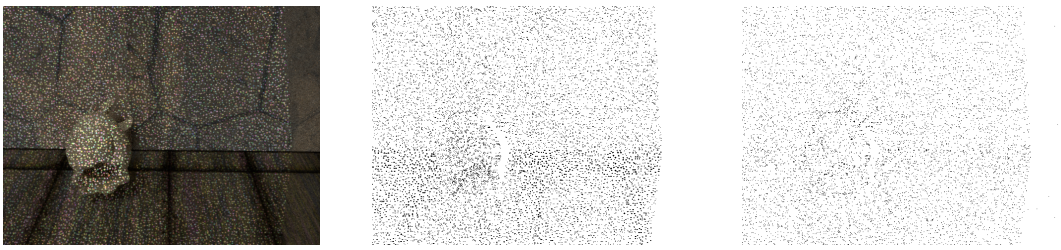


Figure 4.6: Some classification results of the image processing method and the proposed method. The first row contains images of three different scenes, the second row contains the respective ground truth classifications and the third and the fourth rows show the outputs of the image processing method and that of the proposed method respectively. Insets, except the white one, highlight some of the difficult regions where the visibility of the projected pattern is low. The proposed method can still detect dots in these regions while the straightforward image processing solution cannot. The white inset in scene 3 shows a region where thresholding causes false dot detection in the image processing solution indicating that just lowering the threshold to detect more dots in darker regions will cause problems in other areas.



(a) (b) (c)

Figure 4.7: (a) Image of a test scene with associated accuracy maps of (b) the image processing method and (c) the proposed method. The accuracy map of the image processing method has more black pixels, signifying incorrect classification, than that of the proposed method indicating better classification accuracy of the proposed method.

4.2 Extracting General Patterns

The method in section 4.1 works for a very particular type of pattern and requires training data for supervised training. But obtaining high-quality and accurate training data for real-world applications is costly. Moreover, this method may require domain adaptation for every different environment. In this section, we propose a general unsupervised iterative method that works with any pattern. This method relaxes the requirement of obtaining training data and performing domain adaptation for different scenes at the cost of more processing time for the optimization process to converge.

4.2.1 Model Architecture

We adapt the ideas from scene decomposition to break a scene down to different components including the pattern we are interested in extracting. Similar to that proposed by Li *et al.* [31], the model has one shared encoder $\mathbf{E}$ and multiple decoders for estimating different components of the scene. Figure 4.8 shows the architecture of the model.

The proposed model has one decoder $\mathbf{D_R}$ to estimate the scene reflectivity $\mathbf{R}$, one decoder $\mathbf{D_N}$ to estimate the scene normals $\mathbf{N}$, one decoder $\mathbf{D_D}$ to estimate the scene depth $\mathbf{D}$ and one decoder $\mathbf{D_P}$ to estimate the distorted pattern $\mathbf{P}$ as it appears in the captured image. The encoder $\mathbf{E}$ takes as input two images of the same scene - one with the projector off ($\mathbf{I_{off}}$) and one with

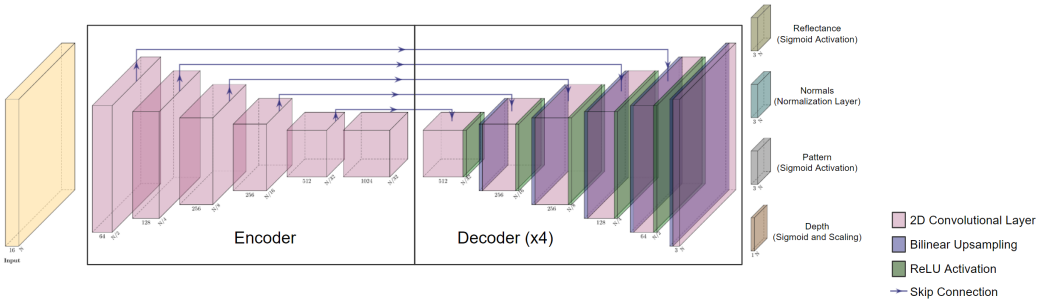


Figure 4.8: The model architecture for decoding general dense patterns.

the projector on projecting the desired pattern ($\mathbf{I_{on}}$) - as well as $\mathbf{R}$, $\mathbf{N}$, $\mathbf{D}$ and $\mathbf{P}$. Each decoder takes the output of the shared encoder $\mathbf{E}$ and estimates the component it is responsible for. More specifically,

$$
\begin{aligned}
X &= E(I_{off}, I_{on}, R, N, D, P; \theta_E), \\
R &= D_R(X; \theta_R), \\
N &= D_N(X; \theta_N), \\
D &= D_D(X; \theta_D), \\
P &= D_P(X; \theta_P),
\end{aligned}
$$

where $\theta_{\mathbf{E}}$, $\theta_{\mathbf{R}}$, $\theta_{\mathbf{N}}$, $\theta_{\mathbf{D}}$ and $\theta_{\mathbf{P}}$ are the parameters of $\mathbf{E}$, $\mathbf{D_R}$, $\mathbf{D_N}$, $\mathbf{D_D}$ and $\mathbf{D_P}$, respectively.

Encoder

We use the same architecture for the encoder as in MGNet0 [31] with slight alterations to account for using different inputs. Along with the two images $\mathbf{I_{off}}$ and $\mathbf{I_{on}}$, the encoder takes as input the scene reflectance $\mathbf{R}$, the scene normal map $\mathbf{N}$, the scene depthmap $\mathbf{D}$ and the extracted pattern $\mathbf{P}$. The encoder consists of six 2D convolutional layers that each doubles the number of its input channels, except for the first one which outputs 64 channels and the fourth one which preserves the number of channels, while halving the input spatial resolution except for the last one which preserves the spatial resolution. The first five convolutional layers use 4×4 kernels while the last one uses a 3×3 kernel.

Decoders

Following the work of Li *et al.* [31], we use a similar architecture for all the four separate decoders and only change their output layers to account for different output types. Aside from the output layer, each decoder has six blocks. The first block takes the output of the encoder and halves the number of channels while preserving the spatial resolution. The next 5 blocks upscale their corresponding inputs to double the spatial resolution using bilinear upsampling and then apply a 2D convolutional layer to half the number of input channels, except the third block which preserves the number of input channels. The first five blocks all have a ReLU activation layer following their corresponding convolutional layers. The blocks in the decoder are connected to their corresponding spatially equivalent layers in the encoder using skip connections. The output of the decoders $\mathbf{D_R}$ and $\mathbf{D_P}$ are passed through a sigmoid activation layer to constrain their outputs to be in the $[\mathbf{0}, \mathbf{1}]$ range. The output of the decoder $\mathbf{D_N}$ is normalized so that the output normals have unit length. Finally, the output of the decoder $\mathbf{D_D}$ is passed through a sigmoid layer and then scaled to the range $[\mathbf{0.2}, \mathbf{2.0}]$ corresponding to a minimum depth of 20cm and a maximum depth of 2m. We found that constraining the outputs of the different decoders as described helps the optimization process to converge faster.

4.2.2 Differentiable Rendering

We use the estimations made by the model to render a synthetic image in a fully differentiable manner that is compared with the actual image captured by the camera to define a loss function. When rendering a synthetic image, we assume that the projector of SL rig is located at the origin projecting towards the positive z-axis. We begin by calculating the 3D coordinates of points corresponding to each pixel in the camera image. For a pixel $\mathbf{p}$, the camera origin $\mathbf{r_0}$ and the direction of the ray $\mathbf{r_p}$ from the camera origin $\mathbf{r_0}$ through the pixel $\mathbf{p}$, we calculate the 3D coordinates $\mathbf{c_p}$ of the world point corresponding to pixel $\mathbf{p}$ as,

$$\mathbf{c_p} = \mathbf{r_0} + \mathbf{r_p} \times d_p,$$

where d_p is the estimated depth of pixel $\mathbf{p}$. Then, for each pixel $\mathbf{p}$, we calculate the light falloff effect l_{f_p} as

$$l_{f_p} = \frac{1}{c_{p_z}^2} \times (-\hat{\mathbf{l}}_\mathbf{p} \cdot \hat{\mathbf{z}}),$$

where c_{p_z} is the z-coordinate of the 3D point $\mathbf{c_p}$, $\hat{\mathbf{l}}_\mathbf{p}$ is a unit vector pointing from the world point $\mathbf{c_p}$ to the projector center and $\hat{\mathbf{z}}$ is the unit vector along the z-axis. Next, for each pixel $\mathbf{p}$, we calculate the light contribution $\mathbf{l_p}$ from the projector as

$$\mathbf{l_p} = \mathbf{P_p} \times l_{f_p},$$

where $\mathbf{P_p}$ is the output for pixel $\mathbf{p}$ of the decoder $\mathbf{D_P}$. The final color in the synthetic image $\mathbf{I_{s_p}}$ corresponding to pixel $\mathbf{p}$ is calculated as

$$\mathbf{I_{s_p}} = \mathbf{I_{off_p}} + \mathbf{R_p} \times \mathbf{l_p} \times \mathbf{N_p} \cdot \hat{\mathbf{l}}_\mathbf{p},$$

where $\mathbf{I_{off_p}}$ is the value of pixel $\mathbf{p}$ in the image of the scene captured with the projector turned off, $\mathbf{R_p}$ is the reflectance for pixel $\mathbf{p}$ calculated from the output of the decoder $\mathbf{D_R}$ and $\mathbf{N_p}$ is the output of the decoder $\mathbf{D_N}$ for pixel $\mathbf{p}$. We assume that we are dealing with a scene with only diffuse reflections and so do not estimate environment maps to handle complex reflections that may occur in the presence of specularity. We also assume that $\mathbf{I_{off_p}}$ is formed as,

$$\mathbf{I_{off_p}} = \mathbf{R_p} \times \overline{\mathbf{L_p}} \times a_p, \tag{4.1}$$

where, $\overline{\mathbf{L_p}}$ is a single light source approximating all the light sources present in the scene and a_p is a light attenuation term accounting for the position and

the distance of the single approximating light source. Equation 4.1 can be rewritten to get,

$$\mathbf{R_p} = \mathbf{I_{off_p}} \times \sigma_\mathbf{p},$$

where, $\sigma_\mathbf{p} = \frac{1}{\overline{\mathbf{L_p}} \times a_p}$. The output of the reflectance decoder $\mathbf{D_R}$ is actually $\sigma_\mathbf{p}$ which is multiplied with $\mathbf{I_{off_p}}$ to produce the reflectance of pixel $\mathbf{p}$. We found that this approximation, as well as the utilization of the light falloff effect which constrains the range of the predicted depth, are crucial for a faster convergence of the optimization process than without using the approximation and the constraint.

4.2.3 Dataset

We use the same real-time data generation method to create the dataset that was detailed in section 4.1. The only difference is that we use two more patterns in addition to the Kinect pattern to generate the data (see figure 4.9).

4.2.4 Training and testing

The proposed method uses unsupervised training for each scene. For each scene, we randomly initialize the reflectance $\mathbf{R}$, the normal map $\mathbf{N}$, the depthmap $\mathbf{D}$ and the extracted pattern $\mathbf{P}$ and feed these along with $\mathbf{I_{off}}$ and $\mathbf{I_{on}}$ to the model to start the training. In the subsequent iterations, we feed the decoder outputs $\mathbf{R}$, $\mathbf{N}$, $\mathbf{D}$ and $\mathbf{P}$ along with $\mathbf{I_{off}}$ and $\mathbf{I_{on}}$ to the model which is trained for 1200 iterations. The Adam optimizer is used to optimize the encoder and all the decoders. We found that the optimization process is highly sensitive to the learning rates of the encoder and the different decoders of the model. We experimented with different combinations of the learning rates and observed that setting the learning rates of the decoders $\mathbf{D_D}$ and $\mathbf{D_N}$ to 0.01 and the learning rates of the encoder $\mathbf{E}$ and the other decoders ($\mathbf{D_R}$ and $\mathbf{D_P}$) to 0.001 result in successful convergence of the optimization process. We use the mean squared error of the synthetic reconstructed image $\mathbf{I_s}$ and the

image captured by the camera as the loss function to guide the optimization process. More specifically, the loss L is defined as,

$$L = \frac{1}{N}\sum_{i=1}^{N}(I_{on_i} - I_{s_i})^2,$$

where, N is the total number of pixels. The optimization process updates $\theta_{\mathbf{E}}$, $\theta_{\mathbf{R}}$, $\theta_{\mathbf{N}}$, $\theta_{\mathbf{D}}$, and $\theta_{\mathbf{P}}$ which are the parameters of $\mathbf{E}$, $\mathbf{D_R}$, $\mathbf{D_N}$, $\mathbf{D_D}$ and $\mathbf{D_P}$, respectively, by calculating $\frac{\partial \mathbf{L}}{\partial \theta_{\mathbf{E}}}$, $\frac{\partial \mathbf{L}}{\partial \theta_{\mathbf{R}}}$, $\frac{\partial \mathbf{L}}{\partial \theta_{\mathbf{N}}}$, $\frac{\partial \mathbf{L}}{\partial \theta_{\mathbf{D}}}$ and $\frac{\partial \mathbf{L}}{\partial \theta_{\mathbf{P}}}$ using the backpropagation technique in each iteration.

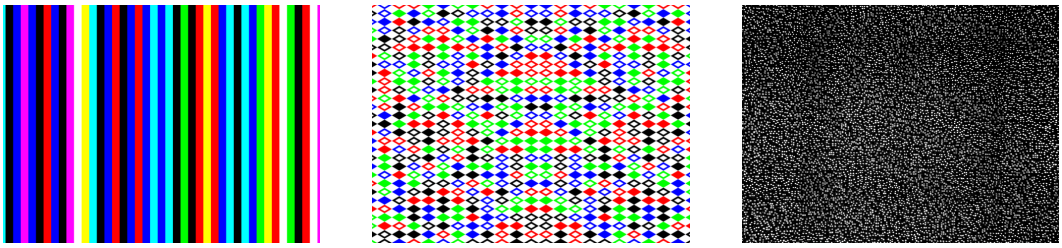
4.2.5 Results

We generate 100 images each for three different patterns using our data generation method. The three different patterns are a vertical stripe pattern created according to de Bruijn sequences [14], the diamond pattern [33] and the Microsoft Kinect pattern which are shown in figure 4.9. We report the mean Peak Signal-to-Noise Ratio (PSNR), which is an objective quantification of the error between the extracted pattern and the ground truth, across the test set in table 4.1. A higher PSNR value corresponds to a lower error and PSNR values above 20 are considered good for many applications. However, higher PSNR values do not always correspond to more visually pleasing outputs. To quantify the perceived similarity between two images, another metric known as the Structural Similarity Index Measure (SSIM) [55] is used which has a range from 0 to 1 with 1 signifying an identical pair of images. We also report the mean SSIM values in table 4.1.

Pattern	PSNR	SSIM
de Bruijn	17.39	0.76
Diamond pattern [33]	16.11	0.72
MS Kinect	13.46	0.57

Table 4.1: Quantitative evaluation of the pattern extraction method for three different patterns using unsupervised training.

Next we present a couple of examples of the extracted de Bruijn patterns to analyze the effectiveness of the proposed method on scenes with complex

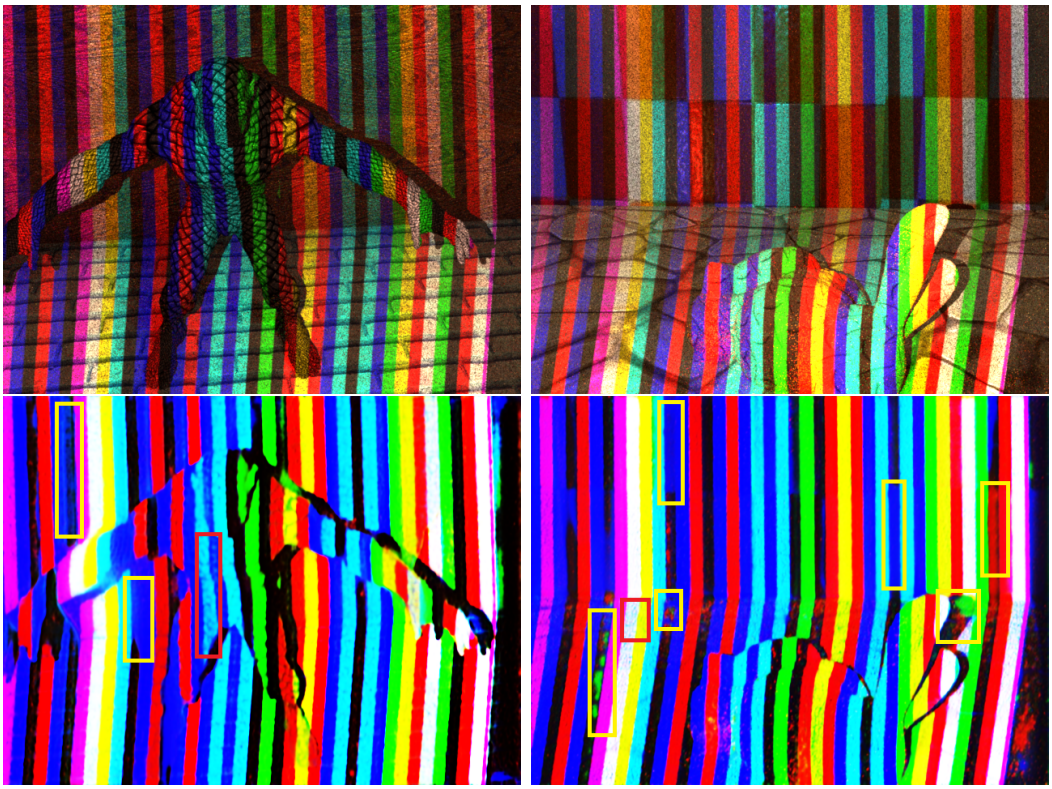


(a) (b) (c)

Figure 4.9: (a) The de Bruijn pattern, (b) the diamond pattern [33] and (c) the MS Kinect pattern.

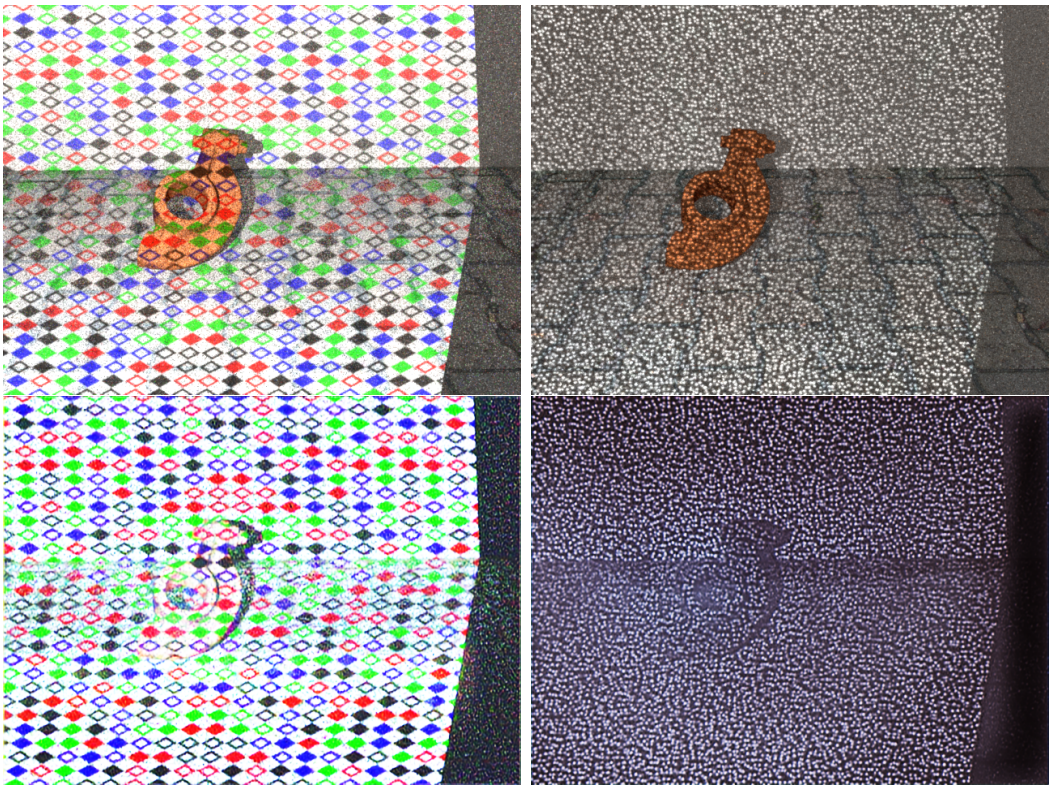
textures and objects of different shapes in figure 4.10. While the method can successfully extract the projected patterns from the camera images with their original colors, some influence of the scene colors still remains in the extracted pattern. Moreover, there is a significant amount of noise present in areas where the pattern is not visible enough, e.g. near the junction of the floor and the back wall where the pattern in almost faded out or in regions occluded from the point of view of the projector, e.g. in the area behind the object where there is no projection, or in areas corresponding to the black-colored pixels of the pattern which may require pattern-specific post-processing to clean up.

Next we try extracting different patterns. Figure 4.11 shows the extraction results for the diamond pattern [33] (figure 4.9b) and the Microsoft Kinect pattern (figure 4.9c). We see that the results are similar to that using the de Bruijn pattern. Areas with low pattern visibility, occluded areas or areas corresponding to black pixels in the projected pattern have a significant amount of noise whereas the other areas are comparatively cleaner. However, the dots in the extracted Kinect pattern do not have the same colors as the ones in the actual projected pattern. Moreover, applying binary thresholding to the extracted Kinect pattern to produce a binary image and comparing that with the ground truth classification shows that this method has a classification accuracy of only 39% for the Kinect pattern. This suggests that this method may not be suitable for sparse patterns like the Kinect pattern as the method is very sensitive to scene noise and cannot recover fine details required for sparse patterns.



(a) (b)

Figure 4.10: Two examples of extracted de Bruijn patterns. Top row shows the captured images and the bottom row the extracted patterns. The yellow insets demonstrate the difficulty of the model to deal with no light projection corresponding to black pixels in the projected pattern or occlusions. There are also regions marked by the red insets where influence of the background texture still remains in the extracted patterns.



(a) (b)

Figure 4.11: Extraction results for (a) the diamond pattern [33] and (b) the Microsoft Kinect pattern. Top row shows the captured images and the bottom row the extracted patterns. The extracted patterns are more noisy in areas where the patterns are less visible, e.g. the areas around the junctions of the wall and the floor.

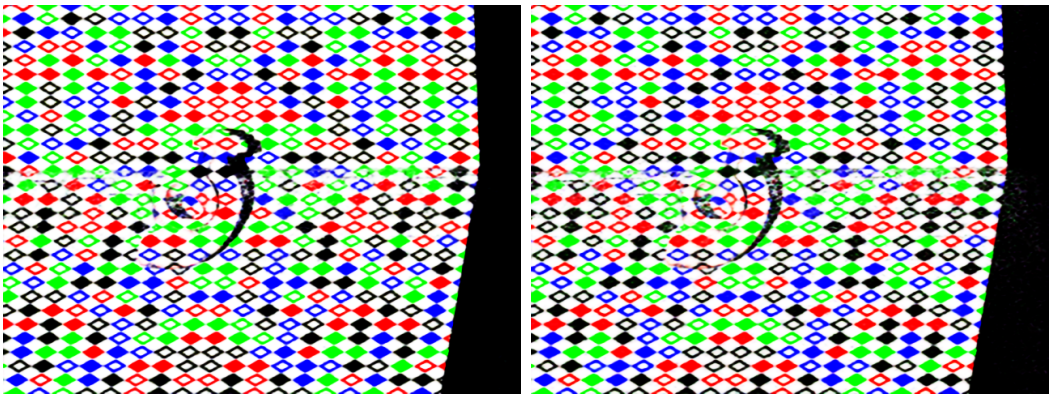
4.2.6 Partial Supervision

We also ran some additional experiments where we trained the proposed model in a supervised manner by defining a mean-squared error loss between the estimated pattern and the ground truth pattern for the diamond pattern [33] with 2000 images. The diamond pattern [33] was chosen specifically as it has a significant amount of white regions which easily highlight residual background effects in the extracted patterns. We conducted these experiments to understand the effects of the unsupervised loss better by studying how it improves upon pure supervised training. In one experiment, we only used the supervised loss and in another, we also kept the unsupervised loss. The ground truth normal, depth or reflectance were not used in the training. Table 4.2 presents the quantitative results.

Loss	PSNR	SSIM
Supervised only	20.64	0.89
Supervised & unsupervised	22.79	0.92

Table 4.2: Quantitative evaluation of the pattern extraction method with partial supervision using the diamond pattern [33].

We can see that just with supervision data alone, without any notion of a model of the projector from the unsupervised loss, the results are good. But adding the unsupervised loss improves the results by helping the model to focus on learning only the additive light from the projector which eliminates the residual background textures in the extracted patterns as can be seen in figure 4.12. Figure 4.13 shows some examples of the extracted patterns using the model trained with both losses.



(a) (b)

Figure 4.12: Pattern extraction results for the same scene of (a) the model trained with both the supervised and the unsupervised losses and (b) the model trained with only the supervised loss. The pattern extracted by the model trained with only the supervised loss exhibits more noise due to residual background texture in challenging areas such as the area around the junction of the wall and the floor.

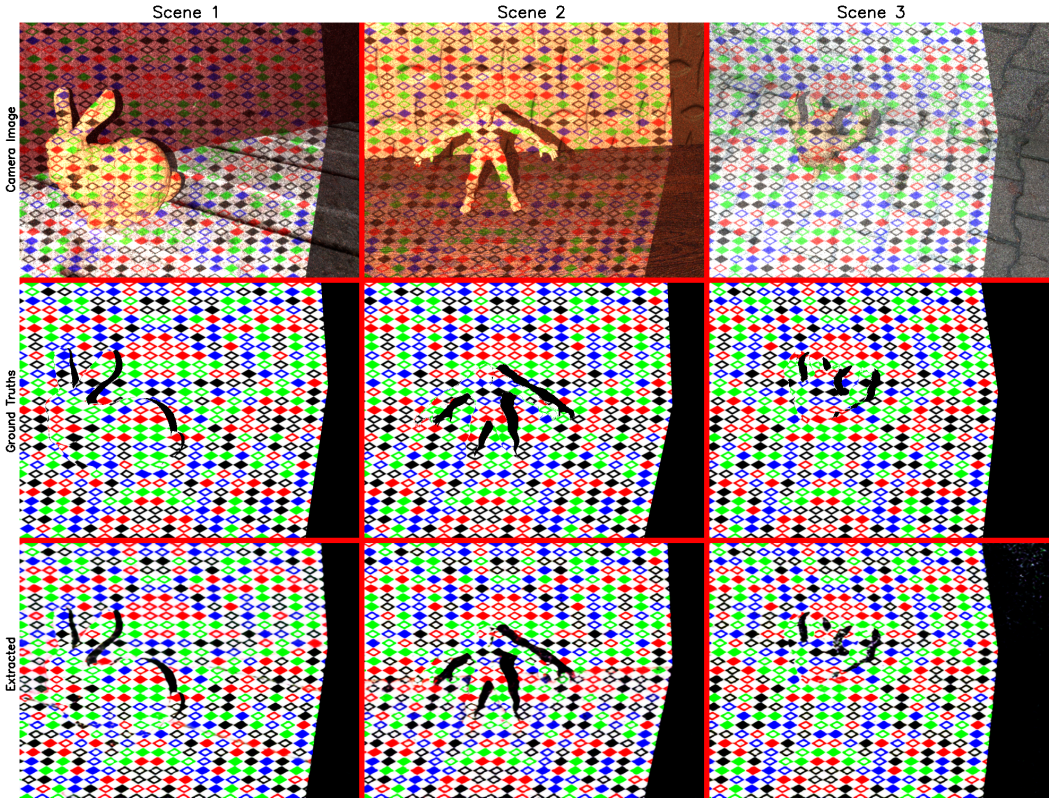


Figure 4.13: Some examples of extracted patterns using a trained model. The extracted patterns closely match the ground truths.

Chapter 5

Conclusion

In this thesis, we propose two novel methods for extracting patterns used in SL systems. The first method, specific to the Microsoft Kinect pattern, trains a U-Net model to classify the dots in the projected pattern. Experiments on synthetic data show that our model can exceed the accuracy of a straightforward image processing approach, especially in challenging areas in a scene where the visibility of the dots in the projected pattern is low. Then we propose another method that can theoretically extract any pattern from images captured by the camera in a SL system without any training at the cost of a longer processing time. The proposed model follows the same approach as one of the state-of-the-art deep learning scene decomposition methods by adding a model of the projection from a projector to the image formation process. As accurately estimating all the different components of a scene such as reflectance, normals, depth etc. is intractable in an unsupervised method, several simplifying assumptions are made to constrain the solution space as much as possible and to only focus on extracting the pattern. Our results show that the proposed method is reasonably good at extracting the pattern in areas where the visibility of the pattern is good with minor influence of the background texture present in the extracted pattern. Areas with low pattern visibility or where there is no light from the projector at all, e.g. black pixels in the projected pattern or areas occluded from the projector's view, are the most problematic areas where the patterns extracted by our method show significant errors.

As was demonstrated by Li and Snavely [30], deep learning models trained on a high-quality physically accurate dataset can outperform models trained on real data even in real-world scenes. Then, Li *et al.* [31] propose an improved method of generating a synthetic dataset that resembles real-world scenarios more closely than all other existing datasets and achieve state-of-the-art performance by training a model with the proposed dataset. Inspired by the success of these approaches, we believe that the next step for pattern extraction methods is to build a physically-accurate synthetic dataset that closely resemble real-world scenarios for SL systems to allow for training deep learning models that extract the projected pattern. We have shown that training with just partial supervision can already produce very good results for diffuse scenes. A model trained on a complete dataset with full supervision should be able to handle complex cases such as occlusions, inter-reflections, specularity etc. as evidenced from the recent success of deep learning models trained on physically-accurate synthetic datasets on the scene decomposition task even for real-world scenes.

Although there is a scarcity of works focused on pattern extraction, it is an important task for building improved SL systems using RGB cameras and projectors. We believe that the ideas from recent data-driven approaches in scene decomposition have the potential of addressing many failure cases of SL systems such as color-blending or specularity. While we do not take advantage of data in this work, our proposed method for extracting general patterns is additive and can easily be adapted to augment supervised physically-accurate scene decomposition methods to create faster and more accurate pattern extractors.

References

[1] J. T. Barron and J. Malik, "Intrinsic scene properties from a single rgb-d image," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 17–24. DOI: 10.1109/CVPR.2013.10.

[2] ——, "Intrinsic scene properties from a single rgb-d image," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 17–24. DOI: 10.1109/CVPR.2013.10.

[3] ——, "Shape, illumination, and reflectance from shading," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 8, pp. 1670–1687, 2015. DOI: 10.1109/TPAMI.2014.2377712.

[4] H. G. Barrow and J. Tenenbaum, "Recovering intrinsic scene characteristics from images," 1978.

[5] A. S. Baslamisli, H.-A. Le, and T. Gevers, "Cnn based learning using reflection and retinex models for intrinsic image decomposition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6674–6683. DOI: 10.1109/CVPR.2018.00698.

[6] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417, ISBN: 978-3-540-33833-8.

[7] J. Blinn, "Texture and reflection in computer generated images," *Communications of the ACM*, vol. 19, no. 10, Oct. 1976. [Online]. Available: https://www.microsoft.com/en-us/research/publication/texture-and-reflection-in-computer-generated-images/.

[8] K. L. Boyer and A. C. Kak, "Color-encoded structured light for rapid active ranging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 1, pp. 14–28, 1987. DOI: 10.1109/TPAMI.1987.4767869.

[9] B. Cabral, N. Max, and R. Springmeyer, "Bidirectional reflection functions from surface bump maps," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '87, New York, NY, USA: Association for Computing Machinery, 1987, pp. 273–281, ISBN: 0897912276. DOI: 10.1145/37401.37434. [Online]. Available: https://doi.org/10.1145/37401.37434.

[10] D. Caspi, N. Kiryati, and J. Shamir, "Range imaging with adaptive color structured light," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 5, pp. 470–480, 1998. DOI: 10.1109/34.682177.

[11] N. Durdle, J. Thayyoor, and V. Raso, "An improved structured light technique for surface reconstruction of the human trunk," in *Conference Proceedings. IEEE Canadian Conference on Electrical and Computer Engineering (Cat. No.98TH8341)*, vol. 2, 1998, 874–877 vol.2. DOI: 10.1109/CCECE.1998.685637.

[12] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15, USA: IEEE Computer Society, 2015, pp. 2650–2658, ISBN: 9781467383912. DOI: 10.1109/ICCV.2015.304. [Online]. Available: https://doi.org/10.1109/ICCV.2015.304.

[13] D. Fofi, T. Sliwa, and Y. Voisin, "A comparative survey on invisible structured light," *SPIE Electronic Imaging-Machine Vision Applications in Industrial Inspection XII, San José, USA*, vol. 5303, pp. 90–97, May 2004. DOI: 10.1117/12.525369.

[14] H. Fredricksen, "The lexicographically least de bruijn cycle," *Journal of Combinatorial Theory*, vol. 9, no. 1, pp. 1–5, 1970, ISSN: 0021-9800. DOI: https://doi.org/10.1016/S0021-9800(70)80050-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021980070800503.

[15] A. Fusiello and L. Irsara, "Quasi-euclidean epipolar rectification of uncalibrated images," *Mach. Vis. Appl.*, vol. 22, pp. 663–670, Jul. 2011. DOI: 10.1007/s00138-010-0270-3.

[16] A. Fusiello, E. Trucco, and A. Verri, "A compact algorithm for rectification of stereo pairs," vol. 12, Oct. 2000. DOI: 10.1007/s001380050120.

[17] M.-A. Gardner, K. Sunkavalli, E. Yumer, X. Shen, E. Gambaretto, C. Gagné, and J.-F. Lalonde, "Learning to predict indoor illumination from a single image," *ACM Trans. Graph.*, vol. 36, no. 6, Nov. 2017, ISSN: 0730-0301. DOI: 10.1145/3130800.3130891. [Online]. Available: https://doi.org/10.1145/3130800.3130891.

[18] J. Geng, "Structured-light 3d surface imaging: A tutorial," *Adv. Opt. Photon.*, vol. 3, no. 2, pp. 128–160, Jun. 2011. DOI: 10.1364/AOP.3.000128. [Online]. Available: http://aop.osa.org/abstract.cfm?URI=aop-3-2-128.

[19] S. Georgoulis, K. Rematas, T. Ritschel, M. Fritz, T. Tuytelaars, and L. Van Gool, "What is around the camera?" In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5180–5188. DOI: 10.1109/ICCV.2017.553.

[20] P. Green, J. Kautz, W. Matusik, and F. Durand, "View-dependent precomputed light transport using nonlinear gaussian function approximations," in *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, ser. I3D '06, Redwood City, California: Association for Computing Machinery, 2006, pp. 7–14, ISBN: 159593295X. DOI: 10.1145/1111411.1111413. [Online]. Available: https://doi.org/10.1145/1111411.1111413.

[21] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman, "Ground truth dataset and baseline evaluations for intrinsic image algorithms," in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 2335–2342. DOI: 10.1109/ICCV.2009.5459428.

[22] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. USA: Cambridge University Press, 2003, ISBN: 0521540518.

[23] J. Huang and D. Mumford, "Statistics of natural images and models," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 1, 1999, 541–547 Vol. 1. DOI: 10.1109/CVPR.1999.786990.

[24] H. Hugli and G. Maitre, "Generation And Use Of Color Pseudo Random Sequences For Coding Structured Light In Active Ranging," in *Industrial Inspection*, D. W. Braggins, Ed., International Society for Optics and Photonics, vol. 1010, SPIE, 1989, pp. 75–82. [Online]. Available: https://doi.org/10.1117/12.949215.

[25] W. Jakob, *Mitsuba renderer*, http://www.mitsuba-renderer.org, 2010.

[26] K. Karsch, K. Sunkavalli, S. Hadap, N. Carr, H. Jin, R. Fonte, M. Sittig, and D. Forsyth, "Automatic scene inference for 3d object compositing," *ACM Trans. Graph.*, vol. 33, no. 3, Jun. 2014, ISSN: 0730-0301. DOI: 10.1145/2602146. [Online]. Available: https://doi.org/10.1145/2602146.

[27] J. Kim and R. Zabih, "Factorial markov random fields," in *Computer Vision - ECCV 2002, 7th European Conference on Computer Vision, Copenhagen, Denmark, May 28-31, 2002, Proceedings, Part III*, A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, Eds., ser. Lecture Notes in Computer Science, vol. 2352, Springer, 2002, pp. 321–334. DOI: 10.1007/3-540-47977-5_21. [Online]. Available: https://doi.org/10.1007/3-540-47977-5%5C_21.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980.

[29] D. C. Lee, M. Hebert, and T. Kanade, "Geometric reasoning for single image structure recovery," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 2136–2143. DOI: 10.1109/CVPR.2009.5206872.

[30] Z. Li and N. Snavely, "Cgintrinsics: Better intrinsic image decomposition through physically-based rendering," in *European Conference on Computer Vision (ECCV)*, 2018.

[31] Z. Li, M. Shafiei, R. Ramamoorthi, K. Sunkavalli, and M. Chandraker, "Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2475–2484.

[32] Z. Li, Z. Xu, R. Ramamoorthi, K. Sunkavalli, and M. Chandraker, "Learning to reconstruct shape and spatially-varying reflectance from a single image," *ACM Trans. Graph.*, vol. 37, no. 6, Dec. 2018, ISSN: 0730-0301. DOI: 10.1145/3272127.3275055. [Online]. Available: https://doi.org/10.1145/3272127.3275055.

[33] H. Lin, L. Nie, and Z. Song, "A single-shot structured light means by encoding both color and geometrical features," *Pattern Recognition*, vol. 54, pp. 178–189, 2016, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2015.12.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320315004720.

[34] C. Liu, J. Yuen, and A. Torralba, "Sift flow: Dense correspondence across scenes and its applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 978–994, 2011. DOI: 10.1109/TPAMI.2010.147.

[35] S. Lombardi and K. Nishino, "Reflectance and natural illumination from a single image," in *ECCV (6)*, 2012, pp. 582–595. [Online]. Available: https://doi.org/10.1007/978-3-642-33783-3_42.

[36] F. MacWilliams and N. Sloane, "Pseudo-random sequences and arrays," *Proceedings of the IEEE*, vol. 64, no. 12, pp. 1715–1729, 1976. DOI: 10.1109/PROC.1976.10411.

[37] S. Maji, N. K. Vishnoi, and J. Malik, "Biased normalized cuts," in *CVPR 2011*, 2011, pp. 2057–2064. DOI: 10.1109/CVPR.2011.5995630.

[38] M. Maruyama and S. Abe, "Range sensing by projecting multiple slits with random cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp. 647–651, 1993. DOI: 10.1109/34.216735.

[39] D. Moreno and G. Taubin, "Simple, accurate, and robust projector-camera calibration," in *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*, 2012, pp. 464–471. DOI: 10.1109/3DIMPVT.2012.77.

[40] S. K. Nayar, G. Krishnan, M. D. Grossberg, and R. Raskar, "Fast separation of direct and global components of a scene using high frequency illumination," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 935–944, Jul. 2006, ISSN: 0730-0301. DOI: 10.1145/1141911.1141977. [Online]. Available: https://doi.org/10.1145/1141911.1141977.

[41] M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob, "Mitsuba 2: A retargetable forward and inverse renderer," *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, vol. 38, no. 6, Dec. 2019. DOI: 10.1145/3355089.3356498.

[42] K. Nishino, "Directional statistics brdf model," in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 476–483. DOI: 10.1109/ICCV.2009.5459255.

[43] Nordmann, Arne. (2007). "Epipolar geometry." [Online; accessed 13-July-2021], [Online]. Available: https://en.wikipedia.org/wiki/File:Epipolar_geometry.svg.

[44] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, pp. 145–175, 2001.

[45] G. Oxholm and K. Nishino, "Shape and reflectance from natural illumination," in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 528–541, ISBN: 978-3-642-33718-5.

[46] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[47] W. Peterson, W. Weldon, W. Peterson, E. Weldon, E. Weldon, and E. Weldon, *Error-correcting Codes*. Cambridge, MA, 1972, ISBN: 9780262160391. [Online]. Available: https://books.google.ca/books?id=5kfwlFeklx0C.

[48] J. Posdamer and M. Altschuler, "Surface measurement by space-encoded projected beam systems," *Computer Graphics and Image Processing*, vol. 18, no. 1, pp. 1–17, 1982, ISSN: 0146-664X. DOI: https://doi.org/10.1016/0146-664X(82)90096-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0146664X8290096X.

[49] K. Sato, "Range imaging system utilizing nematic liquid crystal mask," 1987.

[50] S. Sengupta, J. Gu, K. Kim, G. Liu, D. W. Jacobs, and J. Kautz, "Neural inverse rendering of an indoor scene from a single image," in *International Conference on Computer Vision (ICCV)*, 2019.

[51] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 190–198. DOI: 10.1109/CVPR.2017.28.

[52] D. Terzopoulos, "Image analysis using multigrid relaxation methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 2, pp. 129–139, 1986. DOI: 10.1109/TPAMI.1986.4767767.

[53] A. O. Ulusoy, F. Calakli, and G. Taubin, "One-shot scanning using de bruijn spaced grids," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, 2009, pp. 1786–1792. DOI: 10.1109/ICCVW.2009.5457499.

[54] R. Valkenburg and A. McIvor, "Accurate 3d measurement using a structured light system," *Image and Vision Computing*, vol. 16, no. 2, pp. 99–110, 1998, ISSN: 0262-8856. DOI: https://doi.org/10.1016/S0262-8856(97)00053-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S026288569700053X.

[55] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. DOI: 10.1109/TIP.2003.819861.

[56] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba, "Recognizing scene viewpoint using panoramic place representation," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2695–2702. DOI: 10.1109/CVPR.2012.6247991.

[57] L. Zhang, B. Curless, and S. M. Seitz, "Rapid shape acquisition using color structured light and multi-pass dynamic programming," in *The 1st IEEE International Symposium on 3D Data Processing, Visualization, and Transmission*, Padova, Italy, Jun. 2002, pp. 24–36.

[58] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, Apr. 2012, ISSN: 1070-986X. DOI: 10.1109/MMUL.2012.24. [Online]. Available: https://doi.org/10.1109/MMUL.2012.24.