(b, s)-BCSL : Structured Light Color Boundary Coding for 3D Photography

Asla M. Sá, Paulo Cezar P. Carvalho, Luiz Velho

IMPA, Instituto de Matemática Pura e Aplicada
Estrada Dona Castorina 110, 22460-320 Rio de Janeiro, Brazil
{asla|pcezar|lvelho}@visgraf.impa.br

Abstract

In this paper we present (b, s)-BCSL, a new coded structured light method for 3D photography. It employs a sequence of complementary color stripe patterns to define correspondences within a camera/projector shape acquisition system. Unique codes are associated with the boundaries of successive stripes in time. The use of color fully exploits current imaging hardware and makes possible an effective trade-off between spatial and temporal coherence. Furthermore, alternating complementary colors completely eliminates photometric restrictions.

1 Introduction

Shape from structured light is an active stereo vision technique. Measurement of depth values is carried out with a system that resembles a two-camera stereo system, except that a projection unit is used instead of the second camera. A very simple technique to achieve depth information with the help of structured light is to scan a scene with a projected laser plane and detect the location of the reflected stripe in the camera image. Assuming that the projected laser can be seen by the camera, and both are calibrated, the depth information can be computed by triangulation using the known correspondences.

In order to get dense range information, the laser plane has to be moved in the scene. In an attempt to alleviate the problem of capturing only one depth line per image, a slide containing multiple stripes is projected onto the scene. To distinguish between different stripes they must be coded appropriately, in such a way that their location in the projector can be identified. This type of encoding scheme is called *coded structured light* (CSL).

There is a natural analogy between coded structured light and a digital communication system [7]. The projector coordinates are encoded by the slide patterns and transmitted through the scene. At each point of the camera image, a noisy transmission is received and needs to be decoded. The transmission channel is the object's surface and the transmitted message is the encoded position of the projector pixel. Considering this analogy, two main issues have to be studied:

1. Limitations of the transmission channel, related to surface reflectance properties.
2. How to encode projector pixels, which will restrict the class of objects suitable to be scanned.

In this paper, we propose a new coded structured light scheme based on time-varying complementary color stripe boundaries.

The original contributions of our work include:

- A stripe boundary encoding that, through the use of color supports high resolution depth triangulation, and, at the same time, guarantees the existence of stripe boundaries.
- An scheme that employs a sequence of complementary color patterns to completely eliminate the restrictions on reflectivity properties of the scene.

Our scheme is very robust to image noise and can reliably detect both stripe projected color and stripe boundaries with subpixel accuracy. Moreover, it is also able to recover surface reflectance using time integration of color.

In summary, these unique characteristics provide an excellent trade-off between spatial, temporal and photometric coherence for structured light coding.

The paper is organized as follows. Section 2 reviews the literature of CSL methods and discusses how our approach is related to previous research in the area. Section 3 describes the design of our (b, s)-BCSL coding scheme. Section 4 presents the mechanisms to minimize restrictions imposed on the scene to be scanned. Section 5 shows some results obtained with an experimental 3D photography setup using our scheme. Finally, section 6 concludes with directions for future work.

2 Previous work

The research in structured light coding can be divided in two distinct periods. Early work started with the investigation of active vision systems and, more recently, a new impulse to the area was provided by 3D-photography applications.

Classical research of CSL methods was done in 80's and 90's [2, 10, 13, 15]. A good survey of the work developed in this period can be found in [1]. The flavor of this work was to empirically create light patterns to capture the geometry of scanned 3D objects.

Three main approaches were used in the design of projected light coding systems: spatial chromatic modulation; spatial intensity modulation, and temporal intensity modulation.

Chromatic modulation imposes restrictions on the allowable colors in the scene. When spatial codification is used, local continuity of the scene is necessary for recovering the transmitted code. Conversely, temporal codification restricts motion of the scene.

A characterization of CSL methods in terms of the underlying assumptions about reflectance, spatial and temporal coherence of the scene is given in [6].

In recent years CSL systems have been revisited by many researchers with the motivation to find a theoretical basis that could provide a better understanding of known methods and make possible to develop new improved systems [11, 12, 17].

Such a shift of paradigm can be seen in the work of Hall-Holt and Rusinkiewicz [6], who proposed a CSL scheme based on stripe boundaries. In their scheme, the codes are associated with pairs of stripes, instead of with the stripes themselves as in traditional methods. Boundary coding has several advantages: it gives higher spatial precision and requires less slides (better temporal coherence).

In order to allow the greatest possible variations in scene reflectance, the scheme of Hall-Holt and Rusinkiewicz is based on black and white stripes. This option leads to an undesirable problem: "ghost" boundaries must be allowed (i.e., black to black and white to white transitions). Their scheme is also complicated by restrictions on boundary transitions because of the decoding algorithm, which is augmented by an additional step that solves the matching problem.

The most recent work done by Zhang, Curless and Seitz [17] uses a dynamic programming technique to compute an optimal surface given a projected pattern and the observed image. The camera and projector correspondence is obtained up to pixel resolution and a post-processing step is carried out to achieve sub-pixel accuracy.

The concept of using color boundary codification is present in [17] but the option to use a one-shot codification implies in considering a sub-sequence of consecutive stripes to guarantee uniqueness of codification with the desired resolution. Augmenting the size of the basis used in codification complicates the decodification step. The price of adopting a one-shot codification is that requirements on spatial coherence cannot achieve its minimum, and some information will be lost due to discontinuities in the scene. Some considerations to reduce color misalignment and account for color crosstalk are done in designing and processing correspondences, but the problem is not robustly solved, as shown in their results. To increase resolution and robustness, the same pattern is shifted and projected, leading to a spacetime analysis of the color stripe pattern.

Our coding scheme adopts a stripe boundary time coding as in [6], but uses color to augment the codification basis. In this way, we avoid the need of ghost boundaries. The larger basis makes possible to achieve higher resolution and better precision, as discussed in [4]. This basis also makes the decodification more efficient, consisting essentially of a table lookup [8]. We propose a procedure that solves the reflectance restrictions imposed by the traditional color coding methods, such as in [17]. Our scheme employs alternating complementary colors. This makes the use of colored stripes non-restrictive, in any sense, to reflective properties of the scene.

3 The (b, s)-BCSL Scheme

In designing a code for structuring light, the main parameters to be considered are the number of different stripe colors (i.e., the base length b of the code) and the number s of slides projected at each step of geometry acquisition. We call the resulting coding, a (b, s) CSL scheme and leads to a total of b^s different patterns. A larger number of patterns can be obtained if one codes the transitions between stripes, rather than the stripes themselves. For a boundary-coded scheme (i.e. (b, s)-BCSL), the number of different patterns is $[b(b-1)]^s$ (we assume that two successive stripes may not have the same color, to avoid ghost boundaries).

Increasing the number of colors and slides allows using a larger number of stripes, thus increasing the resolution of the acquired geometry. However, distinguishing the stripes becomes harder and the restrictions become more severe. The goal in designing a BCSL scheme is to be able to provide satisfactory resolution, without making stripe recognition too complex and without imposing strong coherence constraints on the scene.

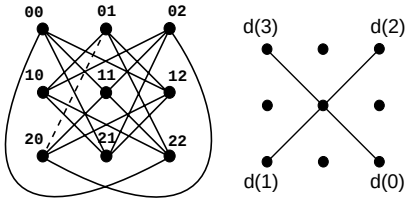
Schemes having $s = 1$ are purely spatial codings, imposing no restrictions on object movement. For $s > 1$, we have spatial-temporal codings. Among these, the case $s = 2$ reduces the need for time coherence to a minimum, namely that objects in the scene move slowly enough so that the displacement between the two captures is smaller than the projected stripe.

In our scheme, we only use the primary colors R, G and B and the corresponding complementary colors C, M and Y, respectively. This is equivalent to using a binary code (0 or 1) for each color channel. We avoid using Black (0 in all channels), since black stripes may be confused with shadow areas. For symmetry reasons, we do not use White, either. Thus, the maximum number of different colors is 6. Using $b = 6$ and $s = 2$, leads to 900 coded boundaries for (6, 2)-BCSL. This is more than what can be processed by most cameras or projectors.

3.1 Coding

We turn now to the problem of generating a particular sequence of b-colored stripes for each of the s slides. This sequence should be generated in such a way that, given the stripe transitions at some pixel for the s slides, we can efficiently recover the position of the corresponding projected boundary.

The problem of generating a set of stripe sequences can be modeled as a the problem of finding a eulerian path in a suitable graph G. G has b^s vertices, each corresponding to a possible assignment of the b colors at a given position for each of the s slides. For instance, for $b = 3$ and $s = 2$, G has 9 vertices, each labeled by a 2-digit number in base b, as shown in figure 1(a). For example, vertex 01 corresponds to projecting color 0 in the first slide and color 1 in the second, at a given stripe position.



(a) graph (3,2) (b) neighborhood

Figure 1: (3,2)-BCSL Encoding.

The edges of G correspond to the possible transitions for two consecutive stripe positions. The forbidden transitions are those that repeat the same color at the same slide, in order to disallow ghost boundaries. For instance, in 1(a), there is not an edge connecting vertex 01 to vertex 02, since that would mean that two consecutive stripes in the first slide would use color 0. On the other hand, there is an edge connecting vertex 20 to vertex 01. This situation is illustrated in figure 2 and 3: at the same border position, we go from color 2 to color 0 in the first slide and from color 0 to color 1 in the second.

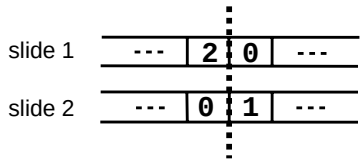


Figure 2: Example of boundary code for dashed edge in Figure 1(a).

For the $(3, 2)$ case, the neighborhood structure is shown in figure1(b), with each vertex having 4 possible neighbors. For the general (b, s) scheme, there are $(b-1)^s$ possible neighbors for each vertex, leading to a regular graph where each of the b^s vertices has degree $(b-1)^s$. It is more appropriate, however, to think of G as a directed graph where each vertex has $(b-1)^s$ incoming arcs and $(b-1)^s$ outgoing arcs, since the same pair of vertices correspond to two distinct transitions, one for each ordering.

Possible color stripe schemes correspond to paths with no repeated edges (meaning that each multi-slide transition occurs only once) in the directed graph G. The maximum number of stripes is achieved for a eulerian path, i.e., a path that visits once each edge of G. This path certainly exists since the every vertex in G has even degree and G is connected (for $b \geq 3$) ([16]).

In fact, there is a very large number of different Eulerian paths in G, and an optimization problem can be formulated to search for the best path according to some desired criteria. The computation of optimal paths is an open and beautiful problem from the coding theory point of view, and several heuristics have been proposed to tackle it. Horn et al. [7] formulated the problem as one of designing optimal signals for digital communication. They consider more important to encode with very distinct codewords points which are spatially distant, than to encode neighbor points with similar codewords. They use this criterion to evaluate the path quality. We could also adopt an image processing perspective, using information about the photometric properties of the scene to be scanned as the criteria to generate an adaptive best code.

In most cases, there is no need to use the complete Eulerian path, since it suffices to use a path of length equal to the maximum resolution handled by the cameras or projectors used.

In the present work we generate the complete path, given a base b and the desired number of slides s, without considering any optimization criteria. To achieve the desired number of stripes, we simply truncate the path.

The encoding algorithm is described below:

Input: A connected digraph G with $d_{in}(v) = d_{out}(v)$ for all v in $V(G)$.

Procedure BCSL_Code(G)

Step 1: Choose an initial vertex v_0 in $V(G)$. Construct an spanning in-tree[1] T starting from v_0 using a breadth-first search algorithm.

Step2: Specify an arbitrary ordering of the edges that leave each vertex v, with the restriction that the edge in T has to be the last one.

Step3: Construct an eulerian circuit starting from v_0 as follows: whenever u is the current vertex, exit along the next unused edge in the ordering specified for edges leaving u, where $\delta_{in}(v)$ is the number of in-edges of v in graph G, and $\delta_{out}(v)$ is the number of its out-edges.

Figure 3 shows a particular (3,2)-BCSL generated by this algorithm; this stripe coloring scheme is used in the first example presented in Section 5.

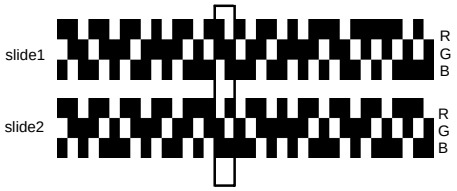


Figure 3: (3,2)-BCSL using R, G, B as base. The outlined boundary has code $20|01$ (see figure 2).

3.2 Decoding

We now consider the problem of recovering a border position, given the color transitions at each slide. This is equivalent to find the position of a given edge in the eulerian path found in the previous algorithm.

We adopt a decoding algorithm presented in [8] to obtain the position of a identified codeword. In our case, to identify the codeword means to find the colors on both sides of the projected boundary that are reflected by the object. A specific image processing method, described in section 4, is used to achieve this goal.

[1] An in-tree starting from v_0, is a tree consisting of a unique path from each vertex u in V(G) to v_0.

The decoding algorithm employs a decoding table that allows computing, in constant time, the projector position of a given stripe boundary found in the images. This table is shown in 1 for the $(3, 2)$ case. Each row of the table corresponds to a vertex v of G (i.e., to a stripe color assignment for all slides), ordered according to its expansion in base b. Each column corresponds to one of its neighbors, ordered according to the pattern shown in figure 1(b). Each one of the neighbors can be conveniently expressed by means of arithmetic operations modulo-b, exploiting the regularity of the adjacency relationships, as shown in detail in [8].

The entry in a given row and column of the table gives the position, in the eulerian path, of the arc corresponding to the transition from the vertex associated with the row to the neighbor associated with the column. For instance the entries 0, 3, 6 and 9 in the first row mean that vertex $v_0 = 00$ occurs four times in the path, at those positions. The corresponding transitions are to neighbors d_0, d_1, d_2 and d_3, which are vertices 11, 12, 21 and 22.

Suppose now that a given transition, from vertex v_i to vertex v_j has been detected. To find its position in the path, it suffices to determine the location of v_j in the neighborhood of v_i and recover the entry for that column in the row corresponding to v_0. For instance, suppose that the transition illustrated in Figure 2, from codeword 20 to 01, has been detected. Since 20 expands to 6 in base 3, the corresponding row is $V(6)$. Subtracting the digits of 01 and 20 module-3 yields 11. To find the position of 01 in the neighborhood of 20 it suffices to subtract 1 from each digit and expand the resulting pattern in base 2. In this case, we find 00 = 0, meaning that 11 is the d_0 neighbor of 20. Since the table entry at the $(V(6), d(0))$ position is 16, this means that the found transition corresponds to the 16th boundary, as marked in Table 1 and in Figure 3.

vertices	d(0)	d(1)	d(2)	d(3)
V(0)	0	3	6	9
V(1)	14	17	19	11
V(2)	28	34	22	24
V(3)	26	29	18	21
V(4)	1	31	33	35
V(5)	15	4	8	13
V(6)	**16**	23	32	12
V(7)	27	5	7	25
V(8)	2	10	20	30

Table 1: Decoding table for (3,2)-BCSL.

4 Uncompromising Use of Color

Considering the proposed codification and the decoding algorithm discussed in the previous section, the problem of corresponding camera pixels to projector pixels is reduced to an image processing task, responsible to identify in camera images the transitions and colors of the projected stripes.

As we have adopted a vertical stripe boundary coding, scan lines can be treated independently.

4.1 Detecting Stripe Boundaries

The crucial step for range image accuracy is the measurement of stripe transition positions in the camera image.

It is desirable to determine the stripe edge position with subpixel accuracy. A known method to obtain sub-pixel accuracy on boundary position consists of projecting the positive and the negative slides of the same pattern.

In this methodology the position of the stripe edge P is computed as the intersection between the line AB and the line EF of positive and negative patterns respectively. (See Figure 4.)

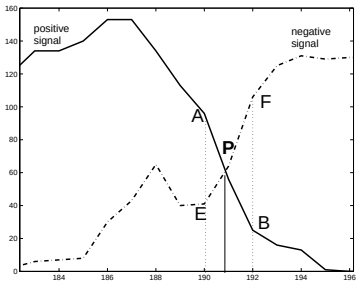


Figure 4: Boundary from complementary patterns.

The above technique assumes that the considered sampling points lie in the linear part of the edge profile. This is true in general, except for very narrow edges whose width is less than two pixels.

The linear interpolation technique with inverse patterns is also very robust against slightly different width of black and white stripes. This phenomenon is caused by a small isolating gap between adjacent stripes in the LCD array.

Trobina [14] did a comparison of the techniques for detecting stripe position and concluded that linear interpolation with inverse pattern is more accurate and robust. The study considered only black and white patterns (one channel). We extend this technique for color patterns (three channels) and exploit the information redundancy to make it more robust with respect to projected shadows.

Shadow regions are viewed by the camera, but are not illuminated by the projector. These areas are characterized by having a very small brightness difference in both positive and negative slides. For these areas, the stripe pattern is not processed and no range values are obtained.

Our boundary detection technique is based on pairs of complementary color stripes. Two subsequent slides are projected with stripes of complementary colors. Each color channel is processed as in the one-channel inverse pattern edge detection, except that we merge boundaries of distinct channels which are within 2 pixel distance.

Shadow regions are detected by analyzing the magnitude and derivative of the complementary color difference image. A point is in shadow if the absolute value in the three channels of the difference image are all bellow the threshold, and the derivative in the horizontal direction is large.

4.2 Recovering Color

The intensity p of a projector light beam is scattered from the object surface into a camera pixel. If the camera characteristic is linear, the sensor clips intensity at a maximum value. The digitized intensity per channel is given by:

$$\begin{aligned} I_R &= \min(u_R + r_R p_R, I_{R_{\max}}) \\ I_G &= \min(u_G + r_G p_G, I_{G_{\max}}) \\ I_B &= \min(u_B + r_B p_B, I_{B_{\max}}) \end{aligned}$$

where (u_R, u_G, u_B) is the ambient light component, (r_R, r_G, r_B) is the local intensity transfer factor mainly determined by local surface properties and (p_R, p_G, p_B) is the projector intensity for each channel [9]. Supposing that $I_{R_{\max}}$, $I_{G_{\max}}$ and $I_{B_{\max}}$ are never achieved we have: $(I_R, I_G, I_B) = (u_R + r_R p_R, u_G + r_G p_G, u_B + r_B p_B)$.

We can estimate parameters u and r if we fix projector, sensor and object in relative positions, and produce sequential projected patterns varying p. As mentioned previously, our option was to project two complementary slides, that is, if $p_i = 0$ on first slide then $p_i = 1$ on second. We have:

$$I_i = \begin{cases} u_i & \text{when } p_i = 0 \\ u_i + r_i & \text{when } p_i = 1 \end{cases}$$

If we just take the maximum value per pixel for each channel between the complementary slides, it will be equivalent to recovering the value of each pixel as if it were illuminated with white light coming from the projector, that is , $p = (1, 1, 1)$. Equivalently, if we take the minimum value per pixel for each channel, we are recovering the ambient light, that is, $p = (0, 0, 0)$.

One problem in the use of color coding is the cross-talk between the RGB sensors [3]. In that respect, color fidelity can be improved by a color correction pre-processing step that takes into account the response of the projector-camera system.

The traditional use of color in coding restricts the object surface reflectivity, because we would not want to modify the projected color when acquiring the surface. By projecting complementary slides, the reflectivity restrictions are eliminated.

From the signal transmission point of view we are introducing redundancy on the transmitted message replicating it on complementary slide. This is a good procedure as it reduces the probability of errors and the codeword read can be checked to assure the correctness of the received message.

Note that, the use of complementary colors allows us to recover both the stripe and scene colors using only two slides.

The considerations above are valid for all pixels, except at stripe boundaries. At those locations, we recover color by interpolating the information from neighbor pixels at both sides of the boundary.

5 Experimental results

Now we demonstrate the application of our (b, s)-BCSL scheme with some examples.

We start with an overview of the method using a virtual acquisition pipeline in which the camera–projector setup is simulated with BMRT [5]. The model in this example is the Stanford Bunny. Color Plates 1(a) to (d) show the input images for the two slide pairs of a (3,2) code. Plate 1(e) shows the computed stripe boundaries. Plates 1(f) and (g) show the stripe codes for slide pairs (a)(b) and (c)(d), respectively. Plate 1(h) shows the recovered texture.

We continue with results generated using a real acquisition setup, consisting of a FujiPix 2400Z digital camera and a Sony VPL-CS10 projector. The acquired images have a resolution of 1024×768 in JPEG format. Three objects with different characteristics were used in the following examples.

The first object is a Ganesh statue, which has a detailed geometry and homogeneous surface properties, as can be seen in Figure 5(a). We used three (b, s) codes with increasing resolution: $(3, 2)$, $(4, 2)$ and $(6, 2)$. The stripe width is 18 pixels for the (3,2) code, 10 pixels for the (4,2) code and 5 pixels for the (6,2) code. The (b, s) slides were generated with 600×480 pixels. Note that the maximum resolution of our codification was achieved only for the (3,2) code, and we could achieve the 5 pixel resolution using the (4,2) code (we used the (6,2) code just to exemplify its use).

Figures 5(b), (c) and (d) show the computed stripes for the (3,2), (4,2) and (6,2) codes, respectively, and Figures 5(f), (g) and (h) show the corresponding stripe boundaries. Figure 5(e) shows the mask for background and shadow areas.

The second object is a furry frog which has a fuzzy geometry. Figures 6(a) and (b) show one of the slide pairs. Figure 6(c) shows the stripe boundaries and Figure 6(d) shows the recovered texture.

The third object is a Babuska doll. It has a simple shape, a complex painted texture and also a highly reflective surface. Color Plates 2(a) and (b) show one of the slide pairs, Color Plate 2(c) shows the stripe codes and Color Plate 2(d) shows the recovered texture.

6 Current Work

We are working on an extension of the method that can process slow moving objects. In this way, we will be able to generalize the $(n, 2)$ version of our codification to dynamic scenes. The main idea is to motion blurr the frames, perform tracking of the stripe boundaries and warp the images accordingly.

We can also formulate a one shot version of (b, s)-BCSL. In this case, instead of projecting stripe coded patterns, we generate a checkerboard pattern alternating vertically the 2 temporal patterns. A vertical "ghosts" restriction has to be formulated. In this one shot version, vertical pixels are not independent and the image processing phase has to be revised.

References

[1] J. Batlle, E. Mouaddib and J. Salvi, "Recent Progress in Coded Structured Light as a Technique to Solve the Correspondence Problem: A Survey", *Pattern Recognition 31(7)*, pp. 963-982, 1998.

[2] K.L.Boyer and A.C.Kak, "Color-encoded structured light for rapid active ranging", *IEEE Trans. Pattern Anal. Mach. Intell. 9(1)*, pp. 14-28, 1987.

[3] D.Caspi, N.Kiryati, and J.Shamir. ''Range imaging with adaptive color structured light", *IEEE Trans. on Pattern Analysis and Machine Intelligence, 20(5)* 1998

[4] P.M.Griffin, L.S.Narasimhan and S.R.Yee, "Generation of uniquely encoded light patterns for range data acquisition" *Pattern Recognition 25(6)*, pp. 609-616, 1992.

[5] L. Gritz and J. Hahn", "BMRT: A global illumination implementation of the RenderMan standard", *Journal of Graphics Tools*, 1(3), pp. 29-47, 1996.

[6] O.Hall-Holt and S.Rusinkiewicz, "Stripe boundary codes for real-time structured-light range scanning of moving objects", *ICCV*, pp. 13-19, 2001.

[7] E.Horn and N.Kiryati, "Toward optimal structured light patterns", *Image and Vision Computing 17(2)*, pp. 87-97, 1999.

[8] Y.C.Hsieh, "Decoding structured light patterns for three-dimensional imaging systems", *Pattern Recognition 34(2)*, pp. 343-349, 2001.

[9] R.W.Malz, "3D sensors for high-performance surface measurement in reverse engineering" *Handbook of computer vision and applications vol. 1, cap 20*

[10] J.L.Posadamer and M.D.Altschuler, "Surface measurement by space-encoded projected beam systems", *Comput. Graphics Image Process. 18*, pp. 1-17, 1982.

[11] C.Rocchini, P.Cignoni, C.Montani, P.Pigni and R.Scopigno, "Allow cost 3D scanner based on structured light", *EUROGRAPHICS 2001 20(3)*, pp. 299-308, 2001.

[12] S.Rusinkiewicz, O.Hall-Holt and M.Levoy, "Real-time 3D model acquisition" *SIGGRAPH 2002*.

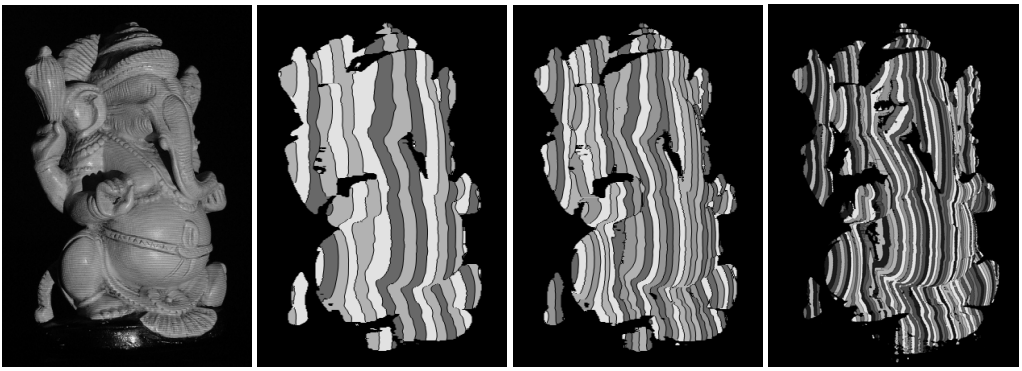
[13] J.Tajima and M.Iwakawa, "3-D data acquisition by rainbow range finder", *Proc. Int. Conf. on Pattern Recognition*, pp 309-313, 1990.

[14] M.Trobina, "Error model of a coded-light range sensor", *Technical Report BIWI-TR-164*, Communication Technology Laboratory, Image Science Group, ETH Zirich, 1995.

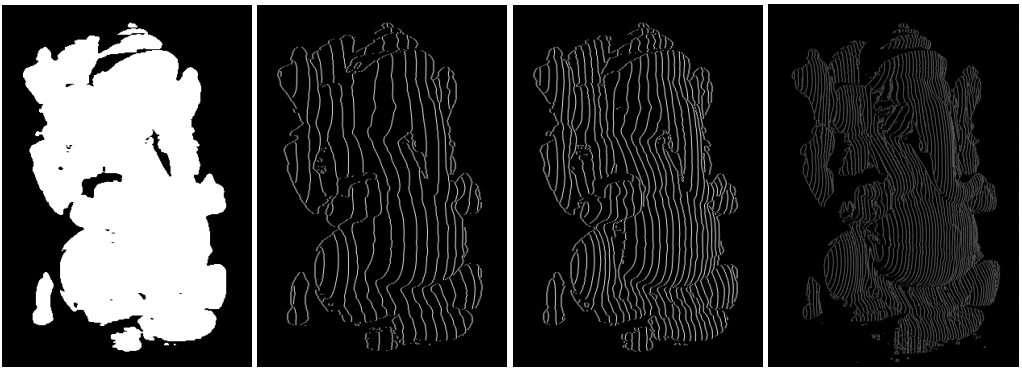
[15] P.Vuylsteke and A.Oosterlinck, "Range image acquisition with a single binary-encoded light pattern", *IEEE Trans. Pattern Anal. Mach. Intell. 12(2)*, pp. 148-164, 1990.

[16] D.B.West, "Introduction to graph theory", Prentice Hall 1996

[17] L.Zhang, B.Curless and S.M.Seitz, "Rapid Shape Acquisition using color structured light and multi-pass dynamic programming", *Proc. Symposium on 3D Data Processing Visualization and Transmission (3DPVT)*, 2002,

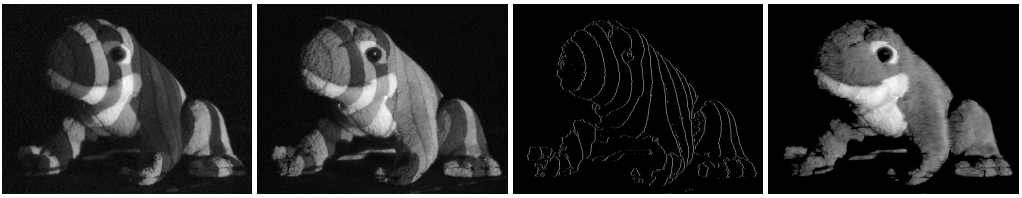


(a) original (b) $(3, 2)$ code (c) $(4, 2)$ code (d) $(6, 2)$ code



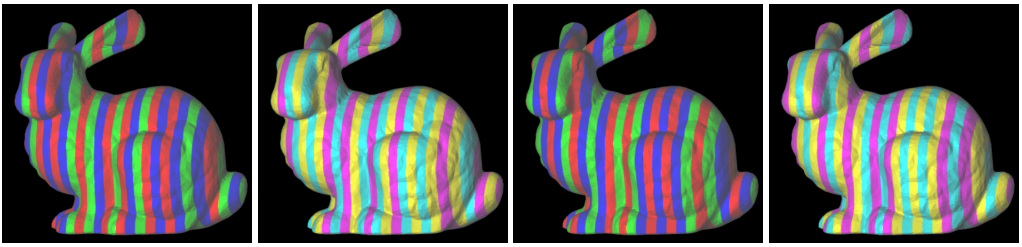
(e) mask (f) $(3, 2)$ edges (g) $(4, 2)$ edges (h) $(6, 2)$ edges

Figure 5: Ganesh.



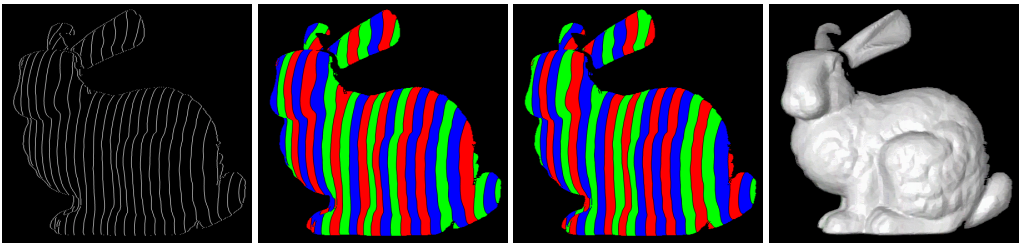
(a) slide i (b) slide i+1 (c) boundaries (d) texture

Figure 6: Frog.



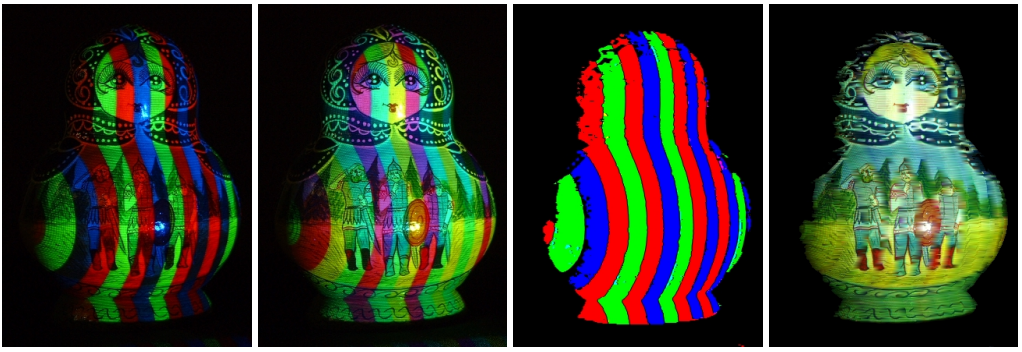
(a) Slide i (b) Slide i+1 (c) Slide i+2 (d) Slide i+3

(Slide pair A) (Slide pair B)



(e) Boundaries (f) Stripe code A (g) Stripe code B (h) Texture

Color Plate 1: Bunny.



(a) Slide i (b) Slide i+1 (c) Stripe code (d) Texture

Color Plate 2: Babuska.