A Camera-Projector System for Real-Time 3D Video

Marcelo Bernardes Vieira Luiz Velho Asla Sá Paulo Cezar Carvalho

IMPA - Instituto de Matemática Pura e Aplicada
Est. Dona Castorina, 110, Rio de Janeiro, Brazil

Abstract

In this paper, we describe a real-time 3D video system that is based on active stereo. Combining a NTSC standard camera/projector equipment and a suitable color code, the geometric and photometric information of a scene is robustly obtained in 30fps. Our main motivation to develop this system is to create a platform for investigating the issues that will be posed by the next generation of digital video and how it can shape up new media.

1. Introduction

The development of the initial generations of digital video focused on technological foundations, system issues and standards. In this respect, the result was a significant improvement of evolutionary aspects, such as image resolution. But the data type remained the same – i.e., color and sound information.

In our vision, the next generation of digital video will bring truly revolutionary innovations by the incorporation of new types of data into the media. Depth information is certainly the most natural candidate of data type to augment digital video. Not only it is consonant with the human perceptual system, but it also facilitates scene analysis by computers to extract information at higher levels.

In this paper, we describe the first phase of the Fourth Generation Video, a project to investigate the next digital video format. Our system consists of an acquisition device, data processing and visualization modules. It generates 3D video in real-time (30fps) from a wide range of scenes.

The acquisition device employs active stereo and is composed of calibrated, synchronized video camera and projector. The data processing module extracts depth information from structured light code. The visualization module renders 3D video using the geometry induced by the color code used. In this paper, we emphasize the 3D video capture aspects of the system using a camera/projector setup.

Section 2 is an overview of related works. In Section 3, we present the (b,s)-BCSL color code which is the base for active correspondence between camera and projector image spaces. The hardware requirements, configuration and calibration for using with (6,2)-BCSL code are presented at Section 4. The details about how video stream is generated, captured and processed in our system are given in Section 5. Results are given in Section 6.

2. Related work

We will call 3D video a time-varying image with color + depth information. The first challenge for 3D video is the design and development of an acquisition device.

In general, 3D acquisition methods in computer vision depend strongly on correspondence and calibration. These methods can be classified based on what type of input data is used and how correspondences are obtained for depth triangulation.

The obvious choice for 3D acquisition would be a system based on a pair of cameras and the use of passive stereo methods. However, fully general stereo is an ill-posed inverse problem which is very hard to solve - and real-time requirements make matters even worse. The literature on passive stereo methods is very extensive and for this reason, we will restrict the discussion here to real-time systems. Most of the proposed algorithms cannot perform in real-time without some kind of hardware acceleration. In this context, a recent trend is to take advantage of programmable GPU's, [8]. Another option is to use multiple fixed cameras and scene analysis to obtain a background-foreground decomposition of the scene. This is the basis for visual hull and photo hull methods [4, 1]

An alternative to passive stereo algorithms is a system based on camera/projector and active stereo. This option has the advantage of simpler and robust constrained stereo algorithms, but the price is that a pattern of light has to be projected on the scene. Recent work in this area investigates different configurations of cameras and projectors [9, 3]. In general, the pattern of light projected defines the capturing features. The method proposed in [5] uses a self-adaptive, one-shot pattern for real-time 3D capture at 20fps. Using simpler methods, our system provides 3D video at 30fps.

Our system is also based on a camera/projector pair and active stereo. The acquisition device is built with off-the-shelf NTSC video equipment. This has many advantages,

such as good cost-performance, ease of synchronization, compatibility and many options of distribution channels. The current configuration of our system is similar to the one proposed in [6]. However, our system is more efficient due to the (6,2)-BCSL color code. It reduces the number of images needed for geometric and photometric detection allowing a 3D video stream with 30fps.

3 The (b, s)-BCSL Scheme

In designing a color code for structuring light, the main parameters to be considered are the number of different colors, i.e., the base length b of the color code and the number s of slides projected at each step of geometry acquisition. A larger number of codewords can be obtained if one codes the transitions between stripes, rather than the stripes themselves. For a boundary-coded scheme, assuming that s slides and b colors are used in coding, the number of different codewords is $[b(b-1)]^s$ (we assume that two successive stripes may not have the same color, to avoid ghost boundaries). We will refer to the resulting code as (b, s)-BCSL, as proposed in [7].

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. Increasing the number of slides allows using a larger number of stripes, thus increasing the resolution that can be obtained using the code. Augmenting the color basis b also increases the possible number of stripes. However, distinguishing the stripes in the decoding phase becomes harder and the restrictions on the scene's reflective properties become more severe.

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 edges of G correspond to the possible transitions for two consecutive stripe positions. In order to disallow ghost boundaries, the forbidden transitions are those that repeat the same color at the same slide.

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. The graph G used to generate the (3,2)-BCSL encoding is shown in Figure 1a.

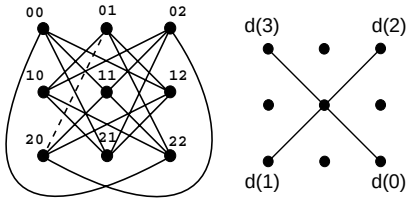


Figure 1: (3,2)-BCSL graph and it's neighboring scheme. G has 9 vertices, each labeled by a 2-digit number in base b

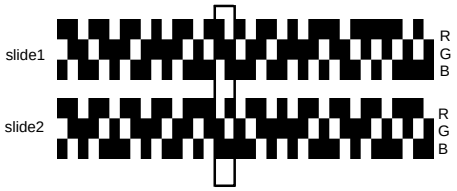


Figure 2: (3,2)-BCSL using R, G, B as base. The outlined boundary has code $20|01$ (see figure 3).

For the $(3, 2)$ case, the neighborhood structure is shown in Figure 1b, 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.

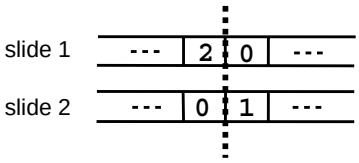


Figure 3: Example of boundary code for dashed edge in Figure 1a. At the same border position, we go from color 2 to color 0 in the first slide and from color 0 to color 1 in the second.

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 every vertex in G has even degree and G is connected (for $b \geq 3$). Figure 2 shows a particular (3,2)-BCSL. The

edge connecting vertex 20 to vertex 01 is highlighted, as well as in Figures 1 and 3.

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. We could 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 camera or projector used.

3.2 Decoding

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

Our 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 Table 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), represented in base b. Each column corresponds to one of its neighbors, ordered according to the pattern shown in Figure 1b. 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 [2].

vertices	d(0)	d(1)	d(2)	d(3)
V(00)	0	3	6	9
V(01)	14	17	19	11
V(02)	28	34	22	24
V(10)	26	29	18	21
V(11)	1	31	33	35
V(12)	15	4	8	13
V(20)	**16**	23	32	12
V(21)	27	5	7	25
V(22)	2	10	20	30

Table 1: Decoding table for (3,2)-BCSL. The table entry at the $(V(20), d(0))$ position is 16, this means that the found transition corresponds to the 16th boundary, highlighted in Figure 2

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 example, the arc that begins at vertex 11 and ends at vertex 02, which is neighbor $d(2)$ of 11 (see fig. 1), is the 33rd arc on path, and the 33rd stripe transition in the pattern.

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. To find out the column to be inspected we have to know which neighbor has the given color, which can be achieved using module-b operations.

Considering the proposed coding and the decoding algorithm presented, the problem of corresponding camera pixels to projector pixels is reduced to an image processing task, responsible for identifying in camera images the transitions and colors of the projected stripes.

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

4. System features

We propose the combination of NTSC standard video with (b,s)-BCSL code to obtain a real-time 3D video stream. The camera/projector settings should be specific for this code in order to obtain better colorimetric and geometric reconstruction.

This section presents color code generation, camera/projector requirements and calibration for using with (6,2)-BCSL code.

4.1. (b,s)-BCSL color pattern generation

Color patterns of (b,s)-BCSL code can be generated using a standard accelerated graphics card with vertical synchronization control and video output.

Using analog video signal for color pattern projection makes the synchronization easier with off-the-shelf equipments. Another advantage is that it can be recorded in standard video media such as tapes. The color pattern is then reproduced by the corresponding player.

However, analog video signal is very restrictive and noisy for our purposes. First, the maximum resolution is about 640×240 pixels by field, limiting considerably the number of stripes that can be projected. Much of color resolution is lost since standard video is designed to have small color bandwidth. An even harder problem is the loss in definition near the stripe boundary transitions, this implies in a noisy depth estimation since it depends on a precise transition detection.

4.2. Projector requirements

A requirement to use complementary colored patterns, as (b,s)-BCSL, is that the projector must be capable to switch fast between two any different colors. Not all projectors have this feature. LCD projectors, for example, have long

latency in switching colors. Thus, they are unable to reproduce stripe colors in large areas of the image. Projectors that filter the input video stream can also give undesired results with (b,s)-BCSL color stripes as, for instance, some models designed for entertainment that try to enhance colors between video fields. Models with DLP (Digital Light Processing) technology with no video filtering are recommended in our case.

Projector contrast ratio should be as large as possible since we need to robustly detect color transitions at stripe boundaries. The projector brightness depend on the application. It will define how strong and far the color pattern will be imposed to the objects. In our set-up, brightness of 1100 lumens is the minimum to capture indoor scenes.

4.3. Camera requirements

Camera / projector synchronization is necessary to guarantee that a given slide will be projected by the projector and captured by the camera at the same time. This can be achieved using a camera that have a *genlock* pin, in a way that a slide video signal is sent simultaneously to the projector and to the camera genlock. It forces the video field capture to begin exactly when the slide field projection is started. Thus, the camera sensor will be exposed to slide colors during all field projection time.

Reconstruction is better performed in well exposed areas of the captured image. A camera with automatic exposure is recommended in order to make the system flexible to arbitrary materials, depths, illumination and motion.

The 3D capture system works only where camera and projector frustums intersect. Adjustable lenses are highly recommended in order to augment the workspace configuration flexibility.

4.4. Color calibration

In general, the camera sensor response curve and the projector reproduction response curve have quite distinct behavior. It means that realizable colors have different coordinates in both color systems. Furthermore, each material composing the scene process the projector and ambient lights differently which makes an optimal calibration method complex.

The key for correspondence using (b,s)-BCSL code is to know where a stripe transition occur and what are the colors in both sides. Thus, a transformation between projector and camera color spaces must be performed. Given a color in captured image, we need to know exactly what color was projected. A simple linear transformation gives quite good approximations on the form:

$$[r_p\ g_p\ b_p]^T = \mathbf{M}\ \cdot\ [r_c\ g_c\ b_c]^T, \qquad (1)$$

where $[r_p\ \ g_p\ \ b_p]^T$ is the estimated projected color, $[r_c\ g_c\ b_c]^T$ is the captured color and $\mathbf{M}$ is a 3×3 matrix which transforms the image color space to the projector color space restricted to the colors used for coding. Note that this transformation can be performed very fast, which is essential for real-time systems.

The calibration procedure starts by projecting each color S_i, $i = 1, 2, ..., s+2$, where s is the number of colors in the code, onto a white, non-specular target and capturing the resulting image I_i. The images for black ($\mathrm{S}_{s+1} = [0\ 0\ 0]^T$) and white ($\mathrm{S}_{s+2} = [1\ 1\ 1]^T$) projections are also obtained. No ambient light should be present at this stage. The camera exposure must be fixed and its response must coincide with the projected black and white range.

A response color C_i, $i = 1, 2, ..., s+2$, can be computed from the captured images $\mathbf{I}_i$ by averaging the pixels reflected by the white target.

Thus, the color calibration matrix M should minimize the error in the system

$$[\mathrm{S}_1^T\ \mathrm{S}_2^T\ ...\ \mathrm{S}_{s+2}^T]\ =\ \mathbf{M} \cdot [\mathrm{C}_1^T\ \mathrm{C}_2^T\ ...\ \mathrm{C}_{s+2}^T], \qquad (2)$$

and can be computed using least-squares method. The camera response for black and white colors give a lower and a upper bound in this minimization.

Note that if **M** were an affine transformation, the translation vector would take the ambient light into account. As a result, the calibration driven by Equation 1 would be too restrictive since a translation on color vectors $[r_c\ g_c\ b_c]^T$ would always be applied. A linear transformation keeps the origin at $[0\ 0\ 0]^T$. As a result, Equation 1 is less sensible to ambient illumination changes.

5. Stream processing pipeline

Our current system is composed by: a projector Infocus LP-70; a video camera Sony Hyper HAD; a PC 1.9Ghz with a Nvidia GForce3 TI based graphics card for pattern generation; a PC 2.4Ghz with a Pinnacle Studio DC10plus capture card to process the input frames from camera (Fig. 4).

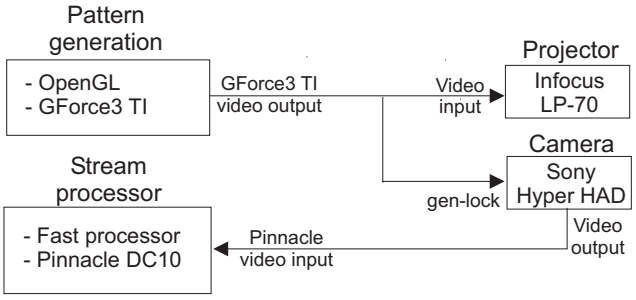


Figure 4: Diagram of the 3D capture system hardware.

One could use a single computer to generate the color patterns, capture video and process the stream. However, the high processing load can result in frame dropping. The

next subsections present in detail how the video stream is processed in our system.

5.1. Projecting and capturing color patterns

Our goal is to capture scene colors and geometry projecting as few color pattern slides as possible. Two slides is the minimum using (b,s)-BCSL code.

The use of complementary patterns is required to robustly detect stripe transitions and colors. The robustness of color detection is augmented if we use only primary colors, though we limit our code to use the six primary colors that can be unambiguously detected through zero-crossing detection: red, green, blue, cyan, magenta and yellow.

This two assumptions leads us to the use of (6,2)-BCSL code which features two slides of a maximum of 900 stripes. The maximum number of stripes is much more than we need using analog video. The current implementation allows the user to choose the total number of stripes and where it begins within the 900 stripe sequence. We will refer to the two pattern color slides as $\mathbf{P}^1$ and $\mathbf{P}^2$. The respective color complementary images are $\mathbf{P}^{1'}$ and $\mathbf{P}^{2'}$.

Each 640×480 video frame in NTSC standard is composed by two interlaced 640×240 fields. Each field is exposed/captured in $1/59.54s$.

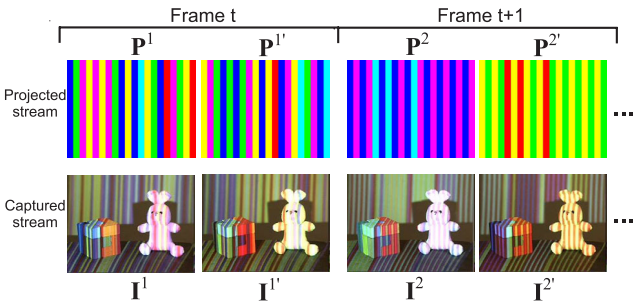


Figure 5: The sequence of color pattern frames and the captured images as a result of their projection onto a scene.

The camera/projector synchronization guarantees that one projected frame will correspond to one captured frame. Thus, the color patterns $\mathbf{P}^{1,2}$ can be coded with their corresponding complements $\mathbf{P}^{1',2'}$ as fields in a single frame. The resulting projected video stream is then $(\mathbf{P}^1_t, \mathbf{P}^{1'}_t)$, $(\mathbf{P}^2_{t+1}, \mathbf{P}^{2'}_{t+1})$, $(\mathbf{P}^1_{t+2}, \mathbf{P}^{1'}_{t+2})$, and so on.

The frame captured at instant t is referred to as $(\mathbf{I}_t, \mathbf{I}'_t)$ or, when the corresponding projected pattern $k \in \{1, 2\}$ is known, as $(\mathbf{I}^k_t, \mathbf{I}^{k'}_t)$. The projected and captured streams are illustrated in Figure 5.

5.2. Detecting stripe boundaries and colors

The projection of the complementary color patterns is the key for finding where stripe transitions occur and what colors were projected. The detection starts by low-pass filtering each input field $\mathbf{I}_t$ and $\mathbf{I}'_t$, at instant t, in order to reduce noise.

(a) (b)

Figure 6: Color calibration example. (a) Input field. (b) Calibrated image.

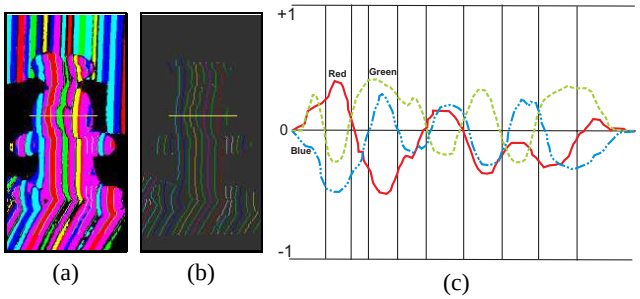
The projected color stripes are clearly observable as shown in Fig. 6a. However, as the projected colors are modulated by the ambient illumination, the materials reflective properties and the camera sensors, a direct processing of the input frames would not be robust.

The camera/projector color matching procedure reduces considerably this problem (Section 4.4). The calibration proposed transforms an arbitrary color into another color which is closer to the colors used for projection. The calibrated images used for detection are defined as

$$\mathbf{C}_t(i,j) = \mathbf{M} \cdot \mathbf{I}_t(i,j) \;\text{ and }\; \mathbf{C}'_t(i,j) = \mathbf{M} \cdot \mathbf{I}'_t(i,j), \quad (3)$$

where $\mathbf{M}$ is the calibration matrix (Eq. 2) and (i, j) are pixel coordinates. Figure 6b shows a calibrated image. Note that the projected colors are better if compared to Figure 6a.

Images should be normalized in order to make the process less sensible to threshold values. In our implementation, the normalization range is defined by zero and by the global maximum intensity obtained by checking all channels.



(a) (b) (c)

Figure 7: Color and boundary detection from the input fields difference. (a) Projected color code. (b) Stripe transitions. (c) Profile of the difference image on bunny's nose. Vertical lines are the detected stripe transitions.

5.2.1 Detecting stripe boundaries

Let $\mathbf{D}_t^R$, $\mathbf{D}_t^G$ and $\mathbf{D}_t^B$ be the color channels of the calibrated images' difference $\mathbf{D}_t = \mathbf{C}_t - \mathbf{C}_t^{'}$.

Since complementary colors are projected at the same position, the boundaries are given by the locations P satisfying

$$\mathbf{D}_t^R(\mathrm{P}) = 0 \text{ or } \mathbf{D}_t^G(\mathrm{P}) = 0 \text{ or } \mathbf{D}_t^B(\mathrm{P}) = 0,$$

i.e., zero-crossings of at least one color channel of the difference image(Fig. 7b,c). However, this general form is highly subject to noise. In order to reduce false detections, the zero-crossings should be examined only in the X direction and have high slope, i.e., high local intensity variation. Thus, for a fixed row j the set of stripe boundaries $\mathbf{SB}_t$ is the set of subpixel locations (i, j) for which

$$\mathbf{D}_t(i, j) = 0 \text{ and } |\mathbf{D}_t(i - \delta, j) - \mathbf{D}_t(i + \delta, j)| \geq ts, \quad (4)$$

for at least one color channel, where δ determines neighbors in both sides and ts is the slope threshold. The δ parameter is usually small and depends on the number of stripes used and how precise the transition is in the image. For analog video, 3-6 pixels are adequate for masking-off the noisy region around the boundary.

The threshold ts can be fixed in 5 if the image is normalized after calibration. This procedure is very robust in detecting stripe boundaries (Fig. 7c).

5.2.2 Detecting projected colors

The projected colors image $\mathbf{PC}_t$ can be recovered by verifying the sign of the difference in each color channel [7]:

- $\mathbf{PC}_t^R(i, j) = 0$, if $\mathbf{D}_t^R(i, j)$ is positive, 1 otherwise;
- $\mathbf{PC}_t^G(i, j) = 0$, if $\mathbf{D}_t^G(i, j)$ is positive, 1 otherwise;
- $\mathbf{PC}_t^B(i, j) = 0$, if $\mathbf{D}_t^B(i, j)$ is positive, 1 otherwise.

Of course, not all objects are well illuminated by the projector, due to material properties, ambient illumination, occlusion, distance from projector, etc. To avoid errors in decoding, only pixels having at least two channels with absolute difference greater than a threshold tc are used. Pixels of $\mathbf{PC}_t$ not satisfying this condition or having white value $[1\ 1\ 1]^T$ are set to black $[0\ 0\ 0]^T$ and thus invalidated (Fig. 7a). In our system, the tc threshold was fixed to 20.

For a stripe boundary $\mathrm{P} = (i, j) \in \mathbf{SB}_t$, the left and right stripe colors are

$$\mathrm{P}_l = \mathbf{PC}_t(i - \delta, j) \text{ and } \mathrm{P}_r = \mathbf{PC}_t(i + \delta, j), \quad (5)$$

respectively. The displacement δ is the same used to find boundaries (Eq. 4). Note that P_l and P_r must be two distinct possible projected colors. Otherwise, another pair of colors should be searched beyond δ. If there are no valid colors, then boundary P is removed.

5.2.3 Detecting dropped frames

Some input frames may be dropped due to delays in the capture system. This will result in two consecutive frames with the same projected pattern. Thus, the camera/projector correspondence using these frames is not possible.

If $\mathbf{PC}_t(\mathrm{P}) = \mathbf{PC}_{t+1}(\mathrm{P})$ for a large number of valid pixels P, then the same color pattern was used. It means that some frames were dropped in projection or capture streams. The latest frame $\mathbf{PC}_{t+1}$ should be kept for correspondence with next frame. This is a very robust method to always correlate frames with different patterns in the pipeline.

5.3. Camera/projector correspondence

Camera/projector correspondence is achieved using the stripe boundaries and projected colors of two consecutive frames. The first step is to correlate all stripe boundaries $\mathrm{P} \in \mathbf{SB}_t$ detected in frame t with the boundaries $\mathrm{Q} \in \mathbf{SB}_{t-1}$ detected in frame $t - 1$.

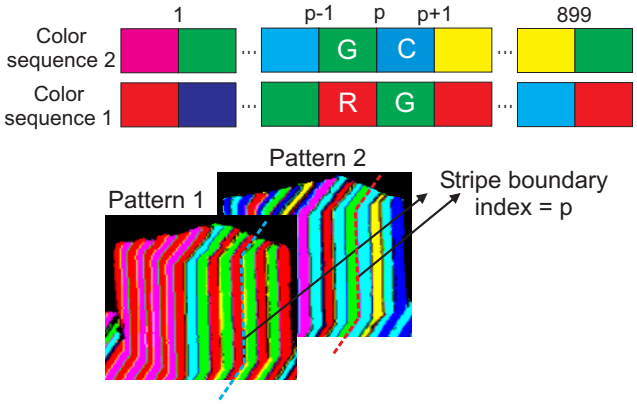


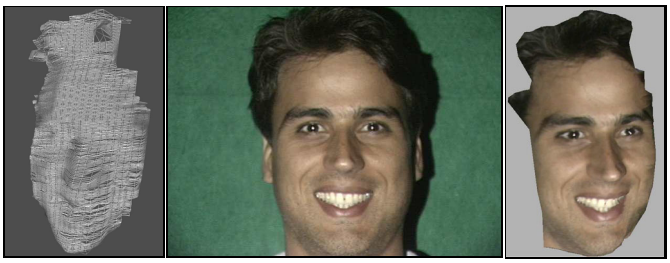
Figure 8: Decoding stripe transitions using (6,2)-BCSL.

At this point, a projected stripe position $p_t(\mathrm{P})$ is assigned to each boundary P in camera image space. If there is no matching Q, P must be kept for next frame correlation.

We need to find the nearest point Q which, combined with P, gives a valid (6,2)-BCSL stripe position by decoding the color tuples $(\mathrm{P}_l, \mathrm{P}_r, \mathrm{Q}_l, \mathrm{Q}_r)$ or $(\mathrm{Q}_l, \mathrm{Q}_r, \mathrm{P}_l, \mathrm{P}_r)$ (Fig. 8). Each frame is decoded using the tuple which gives more valid stripe positions. This tuple can be easily predicted from the previous one.

Note that the decoding color order depends on the last projected pattern sequence: $(\mathbf{P}_{t-1}^1, \mathbf{P}_{t-1}^{1'})$, $(\mathbf{P}_t^2, \mathbf{P}_t^{2'})$ or $(\mathbf{P}_{t-1}^2, \mathbf{P}_{t-1}^{2'})$ $(\mathbf{P}_t^1, \mathbf{P}_t^{1'})$.

The boundary Q is searched in a 7×7 neighborhood of P. Surprisingly, this is sufficient even for reasonably fast motion. The reason is that, while objects move, the stripe remains stationary. Since high discontinuities in depth are unlikely to occur in most objects, the P and Q boundaries are likely to be near each other.



(a) (b) (c)

Figure 9: Virtual scene composition. (a) Geometry. (b) Texture. (c) Composed scene.

5.4. Colorimetry and geometry reconstruction

Having the boundaries P $\in$ $\mathbf{SB}_t$ and their estimated project stripe position $p_t(\mathrm{P})$, it is possible to obtain the real 3D point in camera reference system. This is done using the camera and projector intrinsic parameters and the rigid transformation that match their coordinate systems.

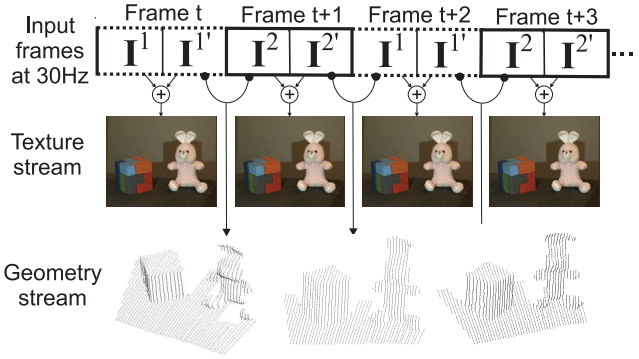


Figure 10: Input video frames and the texture and geometry output streams. The output rate is also 30fps.

The colorimetry can be retrieved by simply adding both input fields since the sum of complementary colors result in white. However, this operation depends on the quality of the projecting and capturing system. In our system, the analog video corrupts colors around stripe boundaries. As a result, the colors in that regions are not well reconstructed.

Note that the patterns are interleaved in time (Section 5.1). In this scheme, each input frame results in one color image, by adding the fields, and is correlated to the previous frame to obtain geometry (Fig. 10). Thus, our pipeline can reconstruct both colorimetry and geometry at 30fps using NTSC.

5.5. Quality tradeoffs

The pipeline proposed is not the unique possible combination of video stream and (6,2)-BCSL code. Other schemes can be implemented depending on the application. However, no scheme can give at same time high quality texture, geometry and frame rate. Thus, there is a tradeoff that must be balanced for each application. The following schemes illustrate this fact:

- great texture at 30fps and poor geometry at 60fps: this is possible by projecting the following sequence as video fields: color pattern 1, full white, and color pattern 2. The white field is used for detecting zero-crossings and projected colors. The difference between the full white and the color patterns give the missing complementary slide. However, this is not robust to ambient light variation, resulting in low geometric precision;

- excellent texture at 10Hz and excellent geometry at 20Hz: complementary patterns are projected as proposed but the texture is not obtained by adding fields. Instead, one full white frame is projected. It gives high quality texture and geometry but their frame rates are low and unbalanced. Fast motion may be a problem.

Hybrid schemes are also possible using the above projection strategies. The pipeline proposed gives reasonable textures and high quality geometry both at 30fps. This is the best compromise for having good geometry and texture quality, interframe registration and correspondence with motion.

6. Results

The geometry and texture delivered by the proposed pipeline are well registered. Figure 9 shows a recovered mesh combined to the respective texture to obtain a virtual 3D view. This model was reconstructed with 90 stripes. The connectivity is given by the nearest boundaries in image space.

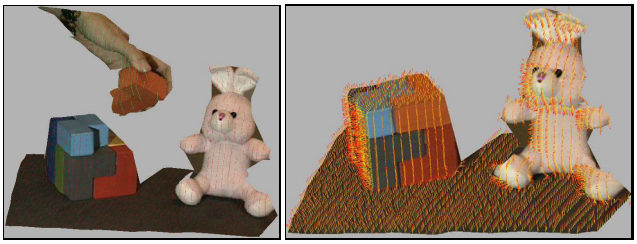
Figure 11 shows a small scale reconstruction example projecting 70 stripes over a scene composed by a $10 \times 10 \times 10$cm cube and a bunny. The smooth curvatures of bunny's body are accurately recovered. The normals showed in Figure 11b were obtained using nearest neighbors in 3D.

Medium scale reconstruction is illustrated by Figure 12. This example shows the fine movements of the mouth and

face detected with 70 stripes. Face 3D information can be used in many application such as security systems, recognition methods, biometry, etc.

The sequence in Figure 13 shows a subject walking in front of a plane. It shows the performance of our system with a complex scenario and reasonably fast motion. The large scale reconstruction was made with 67 stripes.

The complete geometry and texture sequences of these examples can be downloaded, together with a special purpose viewer, from `www.impa.br/~mbvieira/video4d`.



(a) (b)

Figure 11: Bunny and box reconstruction. (a) Detected points. (b) Inferred normals.

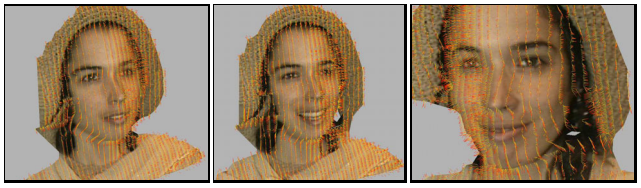


Figure 12: Face and mouth motion.

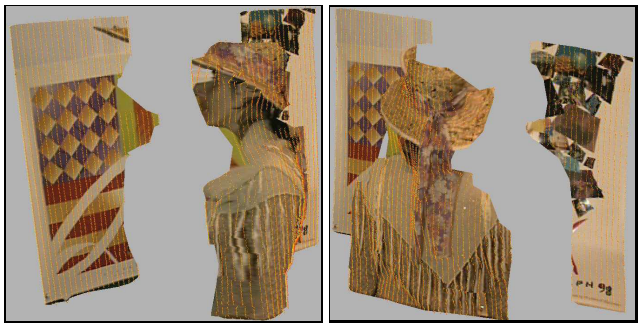


Figure 13: Walking in front of a plane.

7. Summary and Conclusions

In conclusion, we reported in this paper the design of a camera/projector system for real-time 3D video. It provides both color and geometry of reasonably moving objects at 30fps. As usual with camera/projector systems, the main limitation concerns highly specular objects. The system addresses the prospective needs for a next generation digital video system and is the first phase of the project on Fourth Generation Video.

Our development points toward an object based audio-visual content and it contemplates the augmented 3D scene content obtained by incorporating geometry information into the video stream. The second phase of the project will exploit the features of the system in practical applications, such as face detection, recognition and tracking and 3D scene reconstruction, among others.

References

[1] Markus Gross, Stephan Würmlin, Martin Naef, Edouard Lamboray, Christian Spagno, Andreas Kunz, Esther Koller-Meier, Thomas Svoboda, Luc Van Gool, Silke Lang, Kai Strehlke, Andrew Vande Moere, and Oliver Staadt. blue-c: A spatially immersive display and 3d video portal for telepresence. *ACM Transactions on Graphics*, 22(3):819–827, July 2003.

[2] Hsieh. Decoding structured light patterns for three-dimensional imaging systems. *Pattern Recognition*, 34(2):343–349, 2001.

[3] Peisen S. Huang, Chengping Zhang, and Fu-Pen Chiang. High-speed 3-d shape measurement based on digital fringe projection. *Optical Engineering*, 42(1):163–168, 2003.

[4] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 369–374, July 2000.

[5] T. P.Koninckx, A. Griesser, and L. Van Gool. Real-time range scanning of deformable surfaces by adaptively coded structured light. In *Fourth International Conference on 3-D Digital Imaging and Modeling - 3DIM03*, pages 293–300, October 2003.

[6] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3d model acquisition. *ACM Transactions on Graphics*, 21(3):438–446, July 2002.

[7] Asla Sa, Paulo Cezar Carvalho, and Luiz Velho. (b, s)-bcsl: Structured light color boundary coding for 3d photography. In *Proceedings of 7th International Fall Workshop on Vision, Modeling, and Visualization*, 2002.

[8] Ruigang Yang and Marc Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *2003 Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, pages 211–220, June 2003.

[9] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Spacetime faces: high resolution capture for modeling and animation. *ACM Transactions on Graphics*, 23(3):548–558, August 2004.