Appears in the SPIE Visual Communications and Image Processing Conference, San Jose, CA, vol. 5308, pp. 850—861, 20-22 January 2004.

Creating Interactive 3-D Media with Projector-Camera Systems

Nelson L. Chang*
Hewlett-Packard Laboratories, 1501 Page Mill Road, MS 1203, Palo Alto, CA 94304 USA

ABSTRACT

This paper presents a practical framework for creating and visualizing interactive 3-D media using a system of uncalibrated projector-cameras. The proposed solution uses light patterns that temporally encode the projector's coordinate system to solve the traditionally challenging multiframe correspondence problem by straightforward decoding instead of computational multiframe optimization. Two sets of coded light patterns (black/white stripes and colored 2x2 blocks, both of varying spatial resolutions) are presented and compared. The resulting correspondences are directly used as a compelling form of interactive 3-D media through described techniques including three-frame view synthesis, multiframe view synthesis using multiple three-frame groupings, and even single-camera view interpolation. It is shown that adapting the rendering order of the correspondences with respect to the projector's coordinate system ensures the correct visibility for the synthesized views. Experimental results demonstrate that the framework works well for various real-world scenes, even including those with multiple objects and textured surfaces. The framework, along with the resulting correspondences, also has implications in many other computer vision and image processing applications, especially those that require multiframe correspondences.

Keywords: Interactive 3-D media, projector-camera systems, structured light scanning, dense correspondences, view synthesis, view interpolation, multiframe, automatic, real-time

1. INTRODUCTION

Interactive 3-D media are becoming increasingly important as a means of communication and visualization. This observation is clearly evident as more and more online vendors include some form of visual interaction of their products to increase sales. The notion of 3-D media covers any visually compelling content that conveys the impression of 3-D shape to the end user. On traditional 2-D displays, the sense of 3-D is achieved by the user's interaction with the media and the perceived motion parallax cues from the updated display.

Today's 3-D media can be roughly organized into three major categories of increasing interactivity and functionality. The first category consists of strictly image-based representations such as stereoscopic image pairs, panoramic images, and "rotatable" 360º objects. It is very simple to create this type of 3-D media. However, such a medium provides the user with restricted interactivity since only the original viewpoints can be recovered.

A second category starts with a set of images and extrapolates information to synthesize additional views not originally captured. There has been a lot of research in this category in areas including view interpolation, view morphing, and other image-based rendering techniques. With these approaches, one can generate a modest range of virtual viewpoints, which provides the user with more interactivity than the images alone. Without computing true 3-D shape, these approaches still require a fair amount of computation to derive some sense of geometry.

The final group focuses on recovering actual 3-D models. One could estimate accurate 3-D shape (e.g. dense depth maps, voxels, geometric surfaces) to develop a highly flexible and interactive representation of the scene. However, these approaches are often computational and slow, sometimes requiring considerable human intervention to achieve reasonable results. Truly realistic and sophisticated models can take up to several man-months to create. Moreover, accurate 3-D measurement may not be necessary since the end goal is ultimately an image.

* nlachang@hpl.hp.com; phone +1.650.857.5563; http://www.hpl.hp.com/personal/Nelson_Chang

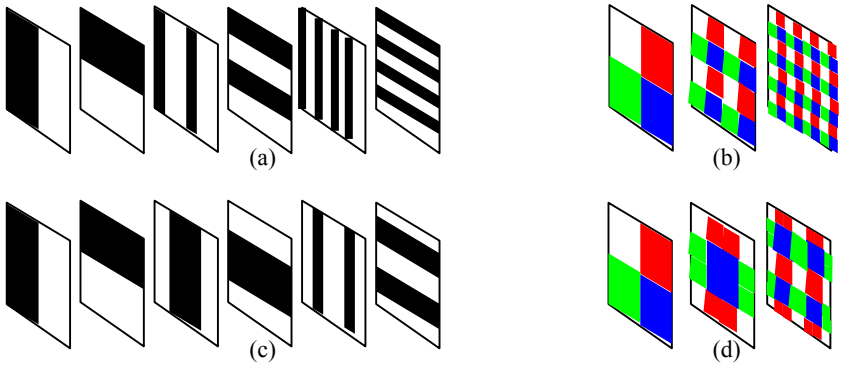


Fig. 1. Example of different temporally encoded light patterns used to represent a 8x8 projector grid: (a) two sets of 1-D base-2 (black and white) stripes; (b) 2-D base-4 (white, red, green, blue) blocks; (c) Gray coded version of (a) referred to as "Stripes"; and (d) reflected and shifted version of (b) referred to as "Blocks".

Of paramount importance to the formation of these interactive 3-D media, as well as many computer vision and image processing applications, is establishing dense correspondences. The traditionally difficult correspondence problem involves determining the 2-D mapping relating points across multiple coordinate systems. The conventional method[6,4] for solving this problem is to use image information directly by exploiting color consistency among the camera images. This passive approach works well for scenes that are distinctly textured and have consistent textures across all the images. The approach has difficulty when the scene is more uniform in color or when the illumination varies across the images. Typically, the approach reliably produces only a sparse set of correspondences, especially in the presence of hard-to-identify occluding boundaries. Furthermore, matching algorithms are often geared for establishing correspondences only two frames at a time. Thus, solving the multiframe correspondence problem for K cameras requires $\frac{1}{2}K(K-1)$ dual-frame matches.

Instead of using passive techniques to solve the correspondence problem, an alternative approach is to use more active techniques. This paper proposes a flexible framework for automatically computing dense correspondences for static 3-D scenes. The framework is designed to work for projector-camera systems without requiring the time-consuming step of accurate calibration. An uncalibrated projector-camera system is particularly beneficial because the imaging components can be arbitrarily placed and quickly relocated.

The proposed framework efficiently determines dense correspondences across any number of imaging components by straightforward decoding instead of computational multiframe optimization. As detailed in Section 2, it uses a series of light patterns that temporally encode the projector's 2-D coordinate system; the paper focuses on two specific sets of light patterns. The resulting correspondences may aid in the creation of 3-D media and, more importantly, may be used directly as a form of interactive 3-D media. Section 3 describes how these correspondences lead to compelling view synthesis. The proposed framework is tested on various real-world sequences and the experimental results are presented in Section 4. Finally, Section 5 summarizes the advantages of the proposed framework, especially as a practical solution for capturing and visualizing interactive 3-D media.

2. USING TEMPORALLY ENCODED LIGHT PATTERNS

To address some of the above issues, this section proposes using temporally encoded light patterns for directly recovering the correspondence mapping between an uncalibrated projector-camera pair. The so-called LUMA approach encodes the 2-D coordinates of the projector's coordinate system (referred to as projector space), wherein each of the wxh connected rectangular regions is assigned a unique temporal code.[3] The projector displays each light pattern in succession onto the 3-D scene of interest and the camera captures the resulting image. Because the codes are temporal in nature, one can decode the projected images at each pixel independently to obtain immediately the mapping between the camera's coordinate system (camera space) and projector space. A refined coarse-to-fine analysis is subsequently performed to resolve any matching ambiguity and to obtain a one-to-one mapping between the two coordinate systems.

Figure 1 shows examples of different sets of temporally encoded light patterns discussed in this paper. Figure 1(a) consists of $\log_2(w)$ spatially varying vertically striped patterns to encode the columns of the projector space and the

$\log_2(h)$ spatially varying horizontally striped patterns to encode the rows of the projector space. These patterns are simply the appropriate bitplane of the binary representation of the row and column indices, respectively. Extending to 2-D, Figure 1(b) uses four colors (white, red, green, blue) in a repeating 2x2 arrangement to form a set of $\log_4(w*h)$ blocked patterns of varying spatial resolutions. Note that this set requires only half as many patterns as the binary set for the same projector space resolution. Figure 1(c) (herein referred to as Stripes) shows the same set of patterns as Figure 1(a) using a Gray coding to reduce the patterns' spatial frequency and improve error resiliency at the pattern transitions.[15] Similarly, through reflection and shifting, Figure 1(d) (herein referred to as Blocks) presents the 2-D Gray coded equivalent of Figure 1(b). This encoding again reduces spatial frequency and ensures that the codes of neighboring pixels differ at one and only one symbol.

This proposed approach for a single projector-camera pair easily extends to handle multiple imaging components. The correspondence mapping is then determined between each component with respect to a reference component. Because all the correspondence information is defined with respect to this reference component, one automatically obtains the correspondence mapping between any pair of the components. Moreover, the computation to obtain the correspondence mapping for each additional component grows linearly with the number of components. Without loss of generality, it is assumed that the imaging system consists of one projector and one or more cameras, where the projector serves as the reference component.

The LUMA approach consists of several detailed steps. The following notation will be used in the remainder of the paper. Suppose there are $K+1$ coordinate systems (CS) in the system, where the projector space is defined as the 0^{th} coordinate system and the K cameras are indexed 1 through K. Let lowercase boldface quantities such as $\boldsymbol{p}$ represent a 2-D point (p_u,p_v) in local coordinates. Let capital boldface quantities such as $\boldsymbol{P}$ represent a 3-D vector (p_x,p_y,p_z) in the global coordinates. Assume the light patterns are formed from R different color bases (e.g. two for the binary striped patterns and four for the colored blocked patterns shown in Figure 1). Suppose there are a total of N light patterns to be displayed in sequence, indexed 0 through $N-1$. Then, define image function $I_k(\boldsymbol{p};n)=\boldsymbol{C}$ as the three-dimensional color vector $\boldsymbol{C}$ of CS k at point $\boldsymbol{p}$ corresponding to light pattern n. Note that $I_0(\cdot;n)$ represents the actual light pattern n defined in the projector space. Denote $V_k(\boldsymbol{p})$ to be the indicator function at point $\boldsymbol{p}$ in camera k. A point $\boldsymbol{p}$ in camera k is defined to be *valid* if and only if $V_k(\boldsymbol{p})=1$. Denote $S_k(\boldsymbol{p};n)$ to be the unknown symbol sequence function corresponding to pattern n at point $\boldsymbol{p}$ in camera k. Note that the decoded symbol sequence is given by the set of symbols $\{S_k(\boldsymbol{p};0), S_k(\boldsymbol{p};1),..., S_k(\boldsymbol{p};N-1)\}$, each symbol consisting of $\log_2(R)$ bits. Define the mapping function $M_{ij}(\boldsymbol{p})=\boldsymbol{q}$ for point $\boldsymbol{p}$ defined in CS i as the corresponding point $\boldsymbol{q}$ defined in CS j. Note that these mappings are bi-directional, i.e. if $M_{ij}(\boldsymbol{p})=\boldsymbol{q}$, then $M_{ji}(\boldsymbol{q})=\boldsymbol{p}$. Also, if points in two CS's map to the same point in a third CS (i.e. $M_{ik}(\boldsymbol{p})=M_{jk}(\boldsymbol{r})=\boldsymbol{q}$), then $M_{ij}(\boldsymbol{p})=M_{kj}(M_{ik}(\boldsymbol{p}))=\boldsymbol{r}$.

The proposed solution to the multiframe correspondence problem is then equivalent to the following: project a series of light patterns $I_0(\cdot;n)$ with $n=0...N-1$, use the captured images $I_k(\cdot;n)$ to compute the symbol sequence functions $S_k(\boldsymbol{p};n)$ for each camera $k=1...K$, and determine the mappings $M_{ij}(\boldsymbol{p})$ for all valid points $\boldsymbol{p}$ and any pair of cameras i and j. The mappings may be determined for each camera independently with respect to projector space since from above

$$M_{ij}(\boldsymbol{p})=M_{0j}(M_{i0}(\boldsymbol{p})) \quad (1)$$

This observation is the key to the efficiency of LUMA. The algorithm to solve the multiframe correspondence problem is as follows

1. Project and capture reference patterns. There are R reference patterns, $I_0(\cdot;0)$ to $I_0(\cdot;R-1)$, corresponding to each of the R possible colors displayed in full. The captured patterns $I_k(\boldsymbol{p};0)$ to $I_k(\boldsymbol{p};R-1)$ provide the approximate mean color vector for each allowable symbol respectively at image pixel $\boldsymbol{p}$ in camera k. Because of illumination variations and different reflectance properties across the object, the mean color vectors will vary across image pixels in every camera.
2. (Optional) Capture the color information of the 3-D scene from all cameras in ambient lighting. These images serve as the color texture maps in the final representation. If black is one of the color bases and there is sufficient ambient light in the scene, then the captured image of its reference pattern can instead serve as the color information for camera k, making this step unnecessary.
3. Determine validity maps for every camera. For every pixel $\boldsymbol{p}$ in camera k, set $V_k(\boldsymbol{p})=1$ if the intersymbol color variance is below some global threshold and 0 otherwise. The invalid pixels correspond to points that lie

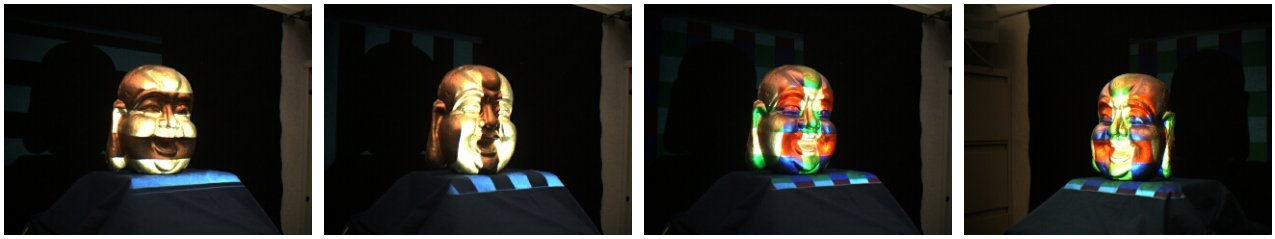


Fig. 2. Examples of captured light patterns for Stripes and Blocks from the "Buddha" image quartet.

outside the projected space or that do not offer enough discrimination among the different symbols (e.g. regions of black in the scene absorb the light).

4. Create the series of light patterns to be projected. The total number of patterns is N including the above reference patterns. These patterns correspond to $I_0(\cdot;m)$, where m varies from R to $N-1$ with increasing spatial frequency.
5. In succession, project each coded light pattern onto the 3-D scene and capture the result from all cameras. In other words, project $I_0(\cdot;m)$ and capture $I_k(\cdot;m)$ for camera k, where m varies from R to $N-1$. It is important to ensure that there is proper synchronization between the projector and the cameras. Figure 2 shows captured images $I_1(\cdot;6)$ and $I_1(\cdot;7)$ from Stripes and $I_1(\cdot;3)$ and $I_4(\cdot;3)$ from Blocks for the "Buddha" image quartet described in Section 4.
6. For every light pattern, decode the symbol at each valid pixel in every image. Let $S_k(\mathbf{p};m)$ be the decoded symbol taking on value 0 thru R. Then, $S_k(\mathbf{p};m)$ = arg min $(|I_k(\mathbf{p};m) - I_k(\mathbf{p}; S_k(\mathbf{p};m))|)$ for all valid points $\mathbf{p}$ in camera k, where $m=R,\ldots,N-1$. Every valid pixel is assigned the symbol corresponding to the smallest absolute difference between the perceived color from the captured light pattern and each of the symbol mean colors.
7. Perform coarse-to-fine analysis to extract and interpolate imaged corner points at finest resolution of projector space. Define $B_k(\mathbf{q})$ to be the binary map for camera k corresponding to location $\mathbf{q}$ in projector space at the finest resolution, initially all set to zero. A projector space location $\mathbf{q}$ is said to be *marked* if and only if $B_k(\mathbf{q})=1$. For a given resolution level l, one follows these substeps for every camera k:
    a. *Convert symbol sequences of each image point to the corresponding projector space rectangle at the current resolution level*. For all valid points $\mathbf{p}$, the corresponding column c and row r in the 2^{l+1}x2^{l+1} projector space is derived from the decoded symbols $S_k(\mathbf{p};s)$, for $s=0,1,\ldots,l$ based on the encoding scheme used. Hence, $M_{k0}(\mathbf{p})=(c,r)$.
    b. *Locate imaged corner points corresponding to unmarked corner points in projector space*. Suppose valid point $\mathbf{p}$ in camera k maps to unmarked point $\mathbf{q}$ in projector space. Then, $\mathbf{p}$ is an imaged corner candidate if there are image points within a 5x5 neighborhood that map to at least three of $\mathbf{q}$'s neighbors in projector space. The motivation is to use the projector space connectivity to overcome possible decoding errors due to specularities and aliasing. Imaged corners are found by spatially clustering imaged corner candidates together and computing their subpixel averages. Set $B_k(\mathbf{q})=1$ for all corner points $\mathbf{q}$ at the current resolution level.
    c. *Interpolate remaining unmarked corners in projector space at the current resolution level*. The matches to unmarked corners are bilinearly interpolated from matches to the corners' marked nearest neighbors, assuming there are a sufficient number of such neighbors.
    d. *Increment l and repeat steps a-c for all finer resolution levels*. The result is a dense mapping $M_{0k}(\cdot)$ of corner points in projector space to their corresponding matches in camera k.
8. Validate rectangles in projector space. For every point (c,r) in projector space, the rectangle with vertices $\{(c,r),(c+1,r),(c+1,r+1),(c,r+1)\}$ is valid if and only if all of its vertices are marked and they correspond to valid points in camera k.
9. Warp texture map from every camera to projector space. For every valid rectangle in projector space, bilinear interpolation of the points in camera k corresponding to the rectangle vertices is used to fill in the appropriate color information. This warping automatically results in a parallax-corrected view of the scene from the projector's viewpoint.

In the end, one obtains the dense correspondence mapping $M_{k0}(\cdot)$ between any camera k and the projector space, and these mapping may be combined as described above to give the correspondence mapping $M_{ij}(\mathbf{p})=M_{0j}(M_{i0}(\mathbf{p}))$ between any pair of cameras i and j for all valid points $\mathbf{p}$.

It is important to contrast LUMA with existing structured light scanning approaches. Structured light scanning algorithms recover 3-D shape information primarily by 3-D ray-plane intersections.[1] Le Moigne and Waxman[12] project an uncoded grid pattern with landmark dots, an approach that requires a labeling process and a particular camera configuration. Boyer and Kak[2] use a single color pattern to handle only neutrally colored scenes. Other approaches[7,9] use a small set of sophisticated binary patterns to capture moving but mainly color-neutral objects. All of these approaches, as well as those[14,15] that have a similar temporal encoding to LUMA, encode only one dimension (the column indices) of projector space and no point correspondences are consequently determined. Moreover, they require a calibrated projector-camera pair in order to operate successfully.

In contrast, LUMA directly solves for dense correspondences rather than 3-D measurements. As such, it does not require the exact 3-D coordinate transformation between the projector and camera, and it places no explicit constraints on their configuration. It is also not restricted to solitary color-neutral objects; it may be applied to arbitrary 3-D scenes consisting of multiple textured objects. Moreover, LUMA supports any number of projectors and/or cameras in an efficient framework that automatically determines visibility and occlusions across all the components without any additional computation.

3. CREATING INTERACTIVE 3-D MEDIA

The multiframe correspondence mapping obtained by LUMA can help produce different forms of interactive 3-D media. Certainly, one can easily convert the correspondences into 3-D shape information with a calibrated projector-camera system through triangulation using a mesh of triangular patches in a rectangular topology.[3] A complete 3-D model may then be formed by integrating and fusing multiple meshes together. Other structured light approaches[1,7,9] offer similar methods for computing 3-D shape with a calibrated set up.

Alternatively, the correspondences may be used directly as a form of interactive 3-D media, even with an uncalibrated imaging system. One simple method is to perform view interpolation between any pair of images to synthesize intermediate viewpoints that were not originally captured. An intermediate view between two basis images is constructed by examining projector space for points corresponding to the basis images and then computing the weighted average of the pixel information (both location and color). In this formulation, only scene points that are visible by the projector and the cameras (i.e. scene points that lie in the intersection of the projector and cameras' visibility spaces) may be properly interpolated. A single parameter then dictates the relative weights of the basis views. This method gives the user limited interaction and manipulation of a 3-D scene without having to perform any calibration. It should be clear that this interpolation provides a smooth transition between the basis images in a manner similar to image morphing. The key difference is that parallax effects are properly handled through the use of the correspondence mapping. It is worth noting that any of the camera's correspondences can be warped to produce a synthetic view from the projector's viewpoint to make the projector into a virtual camera. In this case, the color information is appropriately transferred from the camera image according to the correspondence mapping. In addition, one can synthesize the view between just a single camera and the projector's view using view interpolation; Section 4 shows an example of this single-frame view interpolation.

To expand the range of synthesizable views, one can instead perform interactive view synthesis with three basis images at a time. In this case, the user controls the desired view through two parameters (α,β) that specify the appropriate

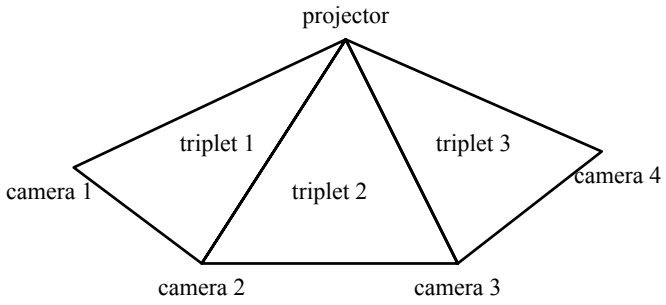


Fig. 3. Example of grouping projector and four cameras into three connected triplets for view synthesis.

weighting of the three basis images; an intuitive 2-D triangular user interface to specify (α,β) was shown previously.[3] Suppose a given scene point $\boldsymbol{P_i}$ corresponds to the points (u_{i1},v_{i1}), (u_{i2},v_{i2}), and (u_{i3},v_{i3}) in the three images respectively. Let (r_{i1}, g_{i1}, b_{i1}), (r_{i2}, g_{i2}, b_{i2}), and (r_{i3}, g_{i3}, b_{i3}) be the color values at each of these points. Then, the scene point appears at the point (u_i,v_i) with color (r_i, g_i, b_i) is given by

$$\begin{bmatrix} u_i \\ v_i \\ r_i \\ g_i \\ b_i \end{bmatrix} = \begin{bmatrix} u_{i1} & u_{i2} & u_{i3} \\ v_{i1} & v_{i2} & v_{i3} \\ r_{i1} & r_{i2} & r_{i3} \\ g_{i1} & g_{i2} & g_{i3} \\ b_{i1} & b_{i2} & b_{i3} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \underbrace{\begin{bmatrix} u_{i3} & u_{i1}-u_{i3} & u_{i2}-u_{i3} \\ v_{i3} & v_{i1}-v_{i3} & v_{i2}-v_{i3} \\ r_{i3} & r_{i1}-r_{i3} & r_{i2}-r_{i3} \\ g_{i3} & g_{i1}-g_{i3} & g_{i2}-g_{i3} \\ b_{i3} & b_{i1}-b_{i3} & b_{i2}-b_{i3} \end{bmatrix}}_{G_i(\alpha,\beta)} \begin{bmatrix} 1 \\ \alpha \\ \beta \end{bmatrix} \tag{2}$$

where $\gamma=1-\alpha-\beta$. In this case, the entire set of views is compactly described by a 5x3 matrix $G_i(\alpha,\beta)$ for every scene point. Note that one of the basis images could just as easily be the synthetic viewpoint of the projector, thereby giving an additional degree of freedom for view synthesis using just two cameras.[3]

The above computation may be sped up quantizing α and β such that only integer and bitwise operations appear in Equation (2). In particular, the parameter space is discretized to reduce the number of allowable views. The real parameter interval [0,1] is remapped to the integral interval $[0,W]$, where $W=2^w$ is a power of two. Let a=round(αW) and b=round(βW) be the new integral view parameters. Then, Equation (2) may be reduced as follows:

$$\begin{bmatrix} u_i \\ v_i \\ r_i \\ g_i \\ b_i \end{bmatrix} = G_i(\alpha,\beta) \begin{bmatrix} 1 \\ \alpha \\ \beta \end{bmatrix} = \frac{1}{W} G_i(\alpha,\beta) \begin{bmatrix} W \\ aW \\ bW \end{bmatrix} \approx \left(\begin{bmatrix} u_{i3} \\ v_{i3} \\ r_{i3} \\ g_{i3} \\ b_{i3} \end{bmatrix} << w + a \begin{bmatrix} u_{i1}-u_{i3} \\ v_{i1}-v_{i3} \\ r_{i1}-r_{i3} \\ g_{i1}-g_{i3} \\ b_{i1}-b_{i3} \end{bmatrix} + b \begin{bmatrix} u_{i2}-u_{i3} \\ v_{i2}-v_{i3} \\ r_{i2}-r_{i3} \\ g_{i2}-g_{i3} \\ b_{i2}-b_{i3} \end{bmatrix} \right) >> w \tag{3}$$

where '<<' and '>>' refer to bit shifting left and right, respectively. Thus, floating point multiplication is reduced to fast bit shifting and all computations are performed in integer math. The computed quantities are obviously approximations to the actual values due to round-off errors.

It is straightforward to use Equations (2) or (3) to provide even greater flexibility when the projector-camera system consists of three or more cameras. The cameras are grouped into connected triplets where one of the cameras or even the synthetic projector view always serves as one basis image, with the other two basis images determined by the user. Figure 3 offers an example for a projector and four cameras. A simple heuristic is to assume an ordering to the cameras and allow the user to control a single parameter that determines which neighboring pair of cameras will be used. Then, the data for the projector and the two chosen cameras will be used in the above three-frame view synthesis.

An important issue to consider for view synthesis is the proper rendering of multiple surfaces. It is possible for the desired view (α,β) to contain surfaces that occlude one another; occlusions occur when two different triplets $\{(u_{i1},v_{i1})$, (u_{i2},v_{i2}), and $(u_{i3},v_{i3})\}$ and $\{(u'_{i1},v'_{i1})$, (u'_{i2},v'_{i2}), and $(u'_{i3},v'_{i3})\}$ both equate to the same (u_i,v_i) in Equation (2). To resolve this ambiguity, one should select the triplet corresponding to the scene point that is closest to the synthetic camera. However, depth information is not easily determined without knowing how the three views are related to one another in 3-D space. When performing strict view interpolation (α and β both lie inside [0,1]) for scenes that obey the monotonicity assumption[16], occlusions are not a problem since by definition only points in common to all basis views are visible. However, they do pose a potential challenge for more general scenes and for extrapolated views (when α and/or β lie outside [0,1]).

Instead of estimating depth, an alternative is to simply reorder and render the data back-to-front to ensure correct visibility. A careful analysis of using the above formulation with LUMA leads to the following observation:

Observation: *The rendering order of the new view* (α,β) *depends only on the view's projection in projector space.*

With LUMA, the projector space contains all the correspondence information with respect to all the cameras. A correspondence match implies that the associated scene point is visible to both the projector and that specific camera. As discussed before, if the same point in projector space corresponds to points in two or more cameras, then by transitivity all of these points refer to the same physical point in the scene. Assuming no transparent or translucent objects in the scene, then each point in projector space corresponds to exactly one scene point and that scene point is the closest to the projector's viewpoint. One can then regard synthesizing a new view as merely a reprojection of the projector's viewpoint. By projecting the new viewpoint's center of projection onto the projector's coordinate system, one can use McMillan's algorithm[11] to determine the correct rendering order based on the location of this epipole in projector space.

To determine the rendering order, one first computes the fundamental matrix F between the projector space and the desired view given by

$$\begin{bmatrix} x_i & y_i & 1 \end{bmatrix} F \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} x_i & y_i & 1 \end{bmatrix} F \begin{bmatrix} u_{i3} & u_{i1} - u_{i3} & u_{i2} - u_{i3} \\ v_{i3} & v_{i1} - v_{i3} & v_{i2} - v_{i3} \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ \alpha \\ \beta \end{bmatrix} = 0 \qquad (4)$$

where (x_i,y_i) is the projector space point that corresponds to scene point $\boldsymbol{P_i}$. F may be solved through eigenanalysis or a least squares technique that solves for nine different Fs, each with one parameter held constant, and selects the F corresponding to the minimum error in Equation (4). Normalizing the correspondences and iteratively discarding outliers help further improve the results. The epipole e in projector space lies in the nullspace of F^T or $F^Te=0$. Because of noise, e may be obtained by determining the eigenvector corresponding to the smallest eigenvalue of F^T. Assuming that the synthetic camera lies in front of the projector, the epipole's location in projector space determines which of the nine rendering orders to select.[11] Note that F is a function of (α,β) so one could quantize the allowable range of these parameters, precompute F, and form a lookup table to speed up computation. Also, another speed up is to use only a fraction of the thousands of correspondence matches in computing F in Equation (4).

The above results are consolidated into a complete two-pass rendering algorithm that enables users to synthesize new views at interactive rates. Only points visible to all the pertinent basis images are considered; this filtering is straightforward with respect to LUMA's projector space. Given the desired view (α,β), the first pass determines the correct rendering order as described above, traverses the projector space array in that order, and then computes Equation (3) for every valid point. The result is a visually compelling and real-time approximation of the desired viewpoint. The second pass further refines the results by treating the warped points as vertices of quadrilateral patches and rendering the interior using a traditional scanline algorithm. Small holes and gaps are thereby filled in, resulting in an improved synthesized view.

The proposed solution to interactive view synthesis is more flexible and less complex than many image interpolation algorithms. It provides a two-pass approach to allow the users to interactively manipulate the scene in real-time. It also supports an arbitrary number of cameras unlike dual-frame approaches.[5,10,17] Moreover, the proposed solution gives the user an additional degree of freedom as compared to other parameterized approaches.[8] Unlike these previous approaches, the formulation correctly determines visibility for the resulting synthesized view.

Pollard et al.[13] demonstrate a related technique for synthesizing views from a trinocular set up (without a projector). The epipolar geometry is first estimated between every pair of cameras in the set up. Edge points are matched across all three cameras and then transferred to the synthetic viewpoint to form an edge sketch. The two viewing parameters are defined as the cascade of two 1-D interpolation functions. In contrast to a rendering order, visibility is determined by a disparity heuristic. Instead of computing dense correspondences, a raster-based algorithm approximates the results by filling in scan lines between edge intersections, leading to potential rendering errors for weak boundaries in the scene.

4. EXPERIMENTAL RESULTS

Using real-world sequences, one can demonstrate how LUMA efficiently computes dense correspondences to create some compelling synthesized views. The imaging system consists of a single 1.1 GHz Pentium III notebook computer, four widely-spaced Point Grey Research's Dragonfly cameras (640x480, 15 fps) all connected to a single Firewire/IEEE-1394 bus, and one DLP light projector display at XGA (1024x768) resolution; an example of the system is shown in

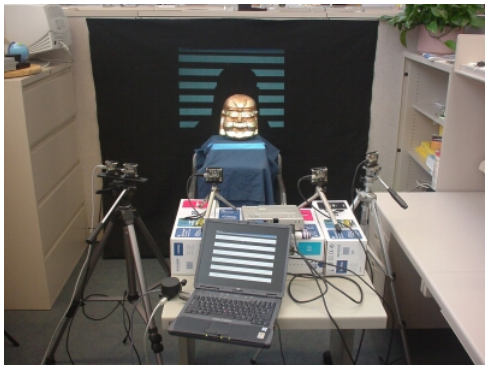


Fig. 4. The experimental LUMA system uses a DLP projector and up to four Firewire cameras connected to a notebook computer, shown here with one of the Stripes patterns illuminating the scene.

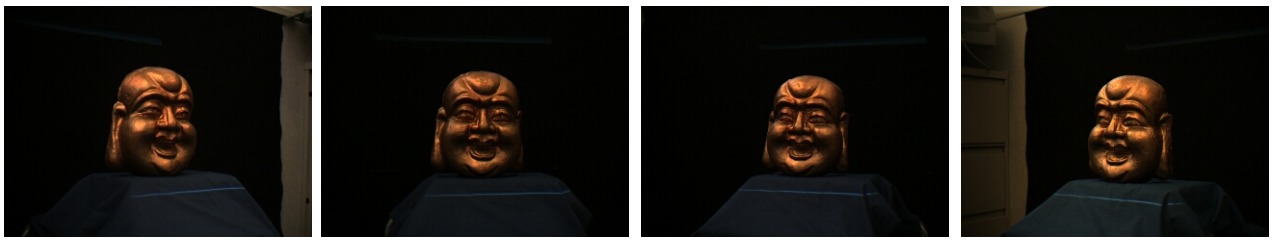


Fig. 5. The "Buddha" image quartet.

Figure 4. The computer serves to control the cameras and the projector, and it runs custom capture and visualization software written in Microsoft MFC Visual C++ and OpenGL. While LUMA supports any number of cameras, bandwidth limitations on the Firewire bus constrains the number to four VGA cameras. The cameras capture both color and correspondence information simultaneously in ambient light.

In the following experiments, the 1024x1024 projector space is encoded by the Stripes patterns with 22 binary light patterns (including two reference patterns) and by the Blocks patterns with a total of 14 multicolored light patterns. Because of projector-camera latency, the complete system is currently limited to about 5 fps to ensure proper synchronization. The capturing process thus takes between three and five seconds, and the coarse-to-fine analysis takes about a minute for four cameras.

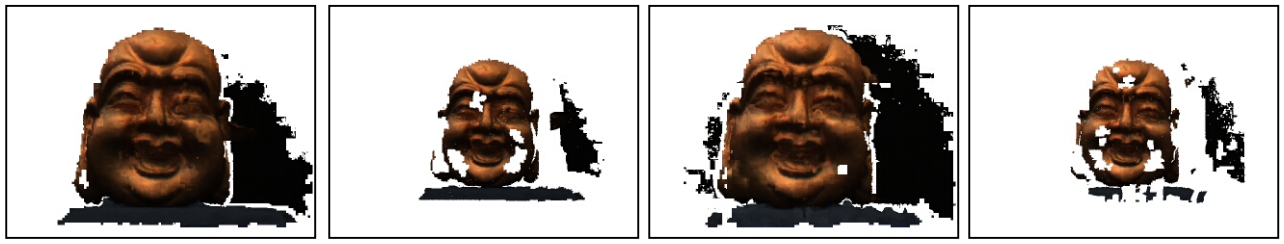
The first example uses four cameras to capture a Buddha mask in the "Buddha" image quartet shown in Figure 5. LUMA automatically determines the dense correspondence mapping with respect to each camera and the projector. Using the black/white Stripes pattern set, the correspondences may be warped to the projector's viewpoint to synthesize a virtual camera image displayed in Figure 6(a). Note that this viewpoint is dramatically different from the four captured images. The regions in white correspond to points which are not visible to the projector or the cameras or which do not offer enough discrimination between the projected color bases. The viewpoint fuses the information captured by all four cameras into a good representation and even correctly maps part of the black cloth behind the mask as well as the supporting structure underneath the mask. The projector and the four cameras form three connected triplets for view synthesis, where the projector is the apex and a neighboring pair of cameras make up the base, similar to the configuration shown in Figure 3. The interactive viewing application allows the user to naturally and smoothly transition between each triplet in real-time. Figures 6(b)-(d) show representative examples of synthesized views from each of the three triplets and demonstrate the wide range of synthesizable views. Figure 6(b) is a bilinear interpolation between the first and second captured images alone. Figure 6(c) comes from an equal weighting of the images in triplet 2 (projector's image, second captured image, and third captured image). Figure 6(d) interpolates the images in triplet 3 by (0.12,0.20). The entire facial structure moves quite naturally and is especially dramatic when seen live. One could render only the points visible by all four cameras and the projector, however there would be very few scene points in common due to the wide baseline. Using the connected triplets ensures that more points will overlap and thus appear in the synthesized images.

(a) (b) (c) (d)

Fig. 6. "Buddha" image quartet using Stripes: (a) estimated projector's viewpoint; (b) synthesized view (0.5,0.5) from triplet 1; (c) synthesized view (0.33,0.33) from triplet 2; and (d) synthesized view (0.12,0.20) from triplet 3.

(a) (b) (c) (d)

Fig. 7. "Buddha" image quartet using Blocks: (a) estimated projector's viewpoint; (b) synthesized view (0.5,0.5) from triplet 1; (c) synthesized view (0.33,0.33) from triplet 2; and (d) synthesized view (0.12,0.20) from triplet 3.



(a) (b) (c) (d)

Fig. 8. "Buddha" image quartet: (a) estimated projector's viewpoint and (b) synthesized view (0.33,0.33) from triplet 2 using Stripes without Gray coding ; (c) estimated projector's viewpoint and (d) synthesized view (0.33,0.33) from triplet 2 using Blocks without Gray coding.

Under the same configuration, one can change the coding scheme and use the colorful Blocks pattern set instead to solve for the correspondence mapping. The resulting projector's viewpoint is shown in Figure 7(a). It is very similar in quality to the one obtained with the Stripes patterns with more of the background cloth included. Figures 7(b)-(d) show the same synthesized views from each of the three triplets as before. In the synthesized views, there are more holes primarily due to decoding problems. Overall, the results are quite reasonable especially given that they are achieved with fewer patterns than Stripes.

Figure 8 presents results using a non-Gray coded version of the Stripes and Blocks patterns shown in Figures 1(a) and (b), respectively. By comparison, the images are clearly much worse than their Gray-coded counterparts. The final projector's viewpoint becomes much more degraded as well. There are a number of holes that arise because of decoding errors especially near pattern transitions. Gray coding the patterns clearly improves the overall results by reducing the finest spatial frequency used and also ensuring that only one color symbol changes across pattern transitions.

Another application of the LUMA system is to perform view interpolation from just a single projector-camera pair. As before, the correspondence mapping is used to create a synthetic projector view and then linear view interpolation is applied to the original image and the projector view. Figure 9 presents an example of a large globe captured using the Stripes pattern set. By simple interpolation, one can smoothly transition between the original viewpoint and the projector's viewpoint. Note that the globe appears to rotate as a function of the view synthesis parameter as expected for

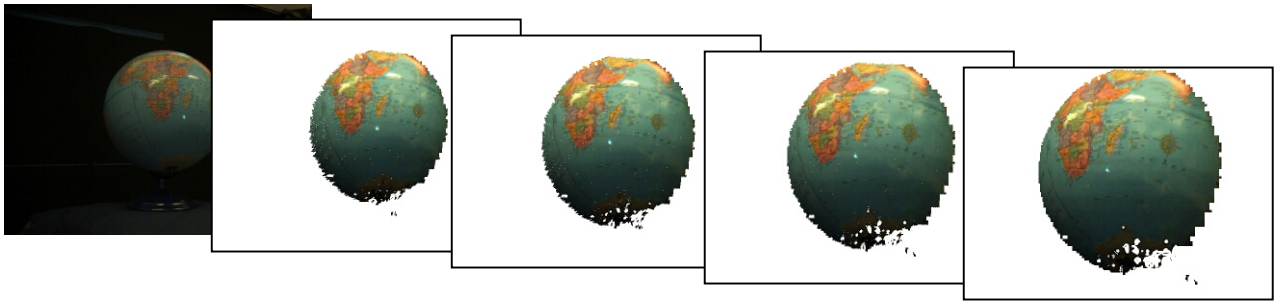


Fig. 9. "Globe" single image and view interpolation results for α=0.2,0.4,0.6,1.0 using Stripes.

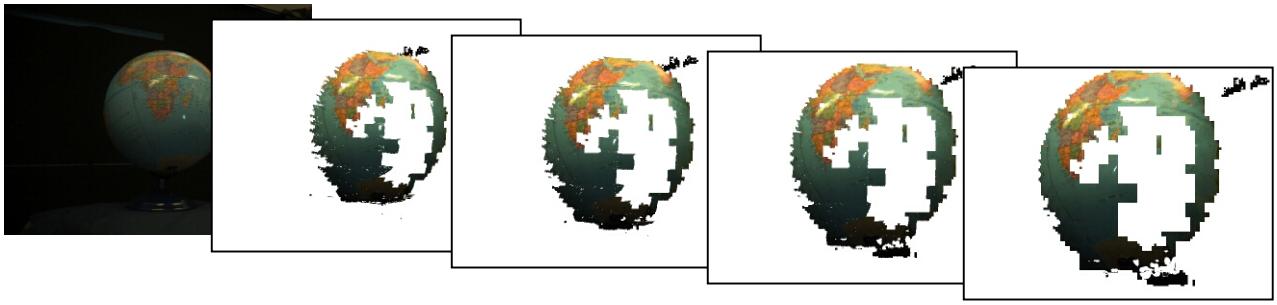


Fig. 10. "Globe" single image and view interpolation results for α=0.2,0.4,0.6,1.0 using Blocks.

such a scene. Also, LUMA has automatically separated the globe cleanly from the background cloth. In contrast, using the Blocks pattern set, the results shown in Figure 10 are much worse with the colored patterns as compared to the black/white patterns. The colored patterns have particular difficulty with the large blue region of the globe resulting in reduced decoding effectiveness and yielding a large hole in the synthesized images.

The final example consists of a complicated scene captured with two cameras and a projector presented in Figure 11. The scene, called "Objects," includes a metal box on the left, a miniature globe to the right, and a highly textured poster in the background. Using the Stripes patterns generates the results shown in Figure 12. The projector's viewpoint in Figure 12(a) is quite good considering the complexity of the scene. The box, globe, and background are captured fairly nicely. Even a small portion of the background poster, seen below and to the left of the globe, is also properly handled. Some of the small decoding holes occur in regions where the intersymbol threshold was found to be too low. Specularities and interreflections that lead to decoding errors are detected and eliminated, also producing holes. Figures 12(b)-(d) offer three different examples of synthesized views using the proposed view synthesis technique with only two original images. Despite some minor decoding artifacts in the background, the results demonstrate an impressive range of allowable motion, and the different structures are recovered quite well. LUMA extrapolates views when one or both of the view synthesis parameters fall outside the [0,1] interval, as shown in Figures 12(c) and (d). Moreover, it correctly resolves the correspondence mapping even in the presence of multiple objects with occlusions and textured surfaces.

To ensure correct visibility especially with multiple surfaces in the scene, the rendering order is modified according to the projected epipole in projector space. The fundamental matrix is computed using the techniques described in Section 3 for each synthesized view with an overall reprojection error of less than 0.002. In Figure 12(c), the epipole is found to be at (+7956,-1201), resulting in a left-to-right and bottom-to-top rendering order with respect to the projector space. Likewise in Figure 12(d) the epipole occurs at (-4082,-582) leading to a right-to-left and bottom-to-top ordering. For comparison, a traditional left-to-right and top-to-bottom ordering for both of these images yields the results in Figure 13. The circled regions mark areas where multiple surfaces occur and the background surface is incorrectly revealed. Using an adaptive rendering order as shown in Figures 12(c) and (d) guarantees correct visibility results.

Figure 14 shows the results of "Objects" captured using the multi-colored Blocks patterns. The colored light patterns introduce a few more decoding errors throughout the scene as compared to the black/white ones, but overall do a reasonable job at recovering the scene especially the background poster. These patterns result in adequate synthesized views that still convey the sense of 3-D in the scene.

Fig. 11. The "Objects" image pair.

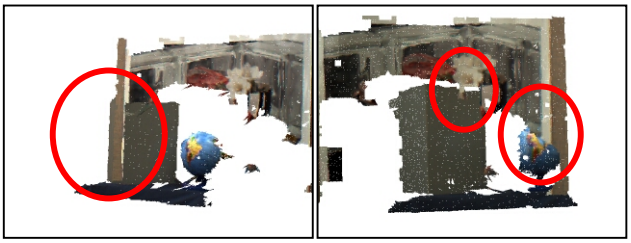
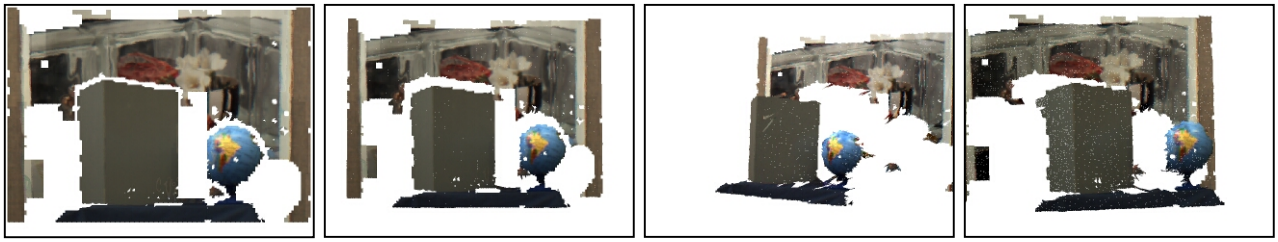


Fig. 13. "Objects" image pair using Stripes: results of using traditional scanning order for rendering Figures 13(c)-(d).



(a) (b) (c) (d)

Fig. 12. "Objects" image pair using Stripes: (a) estimated projector's viewpoint; (b) synthesized view (0.35,0.08); (c) synthesized view (-0.63,1.34); and (d) synthesized view (0.89,-0.49).

(a) (b) (c) (d)

Fig. 14. "Objects" image pair using Blocks: (a) estimated projector's viewpoint; (b) synthesized view (0.35,0.08); (c) synthesized view (-0.63,1.34); and (d) synthesized view (0.89,-0.49).

5. CONCLUSIONS

This paper presented an entire framework for creating and visualizing interactive 3-D media using a system of uncalibrated projector-cameras. The proposed solution uses temporally encoded light patterns to solve the traditionally challenging multiframe correspondence problem in a computationally efficient manner. The resulting correspondences can directly be used as a form of interactive 3-D media through view interpolation, three-frame view synthesis, multiframe view synthesis using piecewise connected triplets, and even single-camera view interpolation. Correct visibility for the rendered scene is automatically determined with respect to the projector's coordinate system. Experimental results with real-world data suggest that, unlike many structured light scanning approaches, the proposed framework works well for multiple objects and textured surfaces.

The paper also compared two different sets of temporally encoded light patterns. The black/white striped patterns produced the best results overall and are effective for a wide variety of scenes. The multi-colored blocked patterns produced reasonable results with fewer patterns, and hence can capture the scene in a shorter time than with the striped ones. However, they are much more sensitive to noise and appear to work better for color-neutral scenes. It is also clear that patterns with Gray coding and its 2-D extension outperform their non-coded counterparts especially at pattern transitions.

One of the main limitations of these sets of patterns is that the scene must remain static for the duration of the series of patterns (about five seconds). Future work consists of reducing the set of patterns to a much smaller number in order to

capture dynamically changing scenes with an uncalibrated set up. Also, adaptive coded patterns are being investigated to reduce the errors due to spatial aliasing.

It is important to stress that the LUMA framework and resulting multiframe correspondences are useful for more than just facilitating the creation of interactive 3-D media. For projector-camera applications, LUMA gives a more general point-to-point mapping between the camera(s) and the projector without having to model the nonlinear lens distortions in either. In the presence of a dominant planar surface in the scene, one can use the resulting correspondences to solve for the homography. For multi-camera applications, LUMA introduces a projector to simplify problems traditionally difficult using passive techniques alone. For example, multiframe correspondences with LUMA become very straightforward to compute without ambiguity since everything is defined with respect to projector space. Furthermore, the same correspondences can also be used to solve for epipolar geometry between any pair of cameras or even for full calibration. Used as a preprocessing step or primary technique, LUMA can simplify many computer vision and image processing problems.

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*, vol. 31, no. 7, pp. 963—982, 1998.
2. K. Boyer and A. Kak, "Color-Encoded Structured Light for Rapid Active Ranging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 10, pp. 14—28, 1987.
3. N. L. Chang, "Efficient Dense Correspondences using Temporally Encoded Light Patterns," *Proceedings of the IEEE International Workshop on Projector-Camera Systems*, Nice, France, October 2003.
4. N. L. Chang and A. Zakhor, "Constructing a Multivalued Representation for View Synthesis," *International Journal of Computer Vision*, vol. 45, no. 2, pp. 157—190, November 2001.
5. S. E. Chen and L. Williams, "View Interpolation for Image Synthesis," *Proceedings of SIGGRAPH,* New York, New York, pp. 279—288, August 1993.
6. O. D. Faugeras, *Three-Dimensional Computer Vision*, MIT Press, Cambridge, MA 1994.
7. O. Hall-Holt and S. Rusinkiewicz, "Stripe Boundary Codes for Real-Time Structured-Light Range Scanning of Moving Objects," *Proceedings of the International Conference of Computer Vision*, Vancouver, Canada, pp. 359—366, July 2001.
8. M. E. Hansard and B. F. Buxton, "Parametric View Synthesis," *Proceedings of the European Conference on Computer Vision*, vol. 1, pp. 191—202, 2000.
9. T. Jaeggli, T. P. Koninckx, and L. Van Gool, "Online 3D Acquisition and Model Integration," *Proceedings of the IEEE International Workshop on Projector-Camera Systems*, Nice, France, October 2003.
10. S. Laveau and O. Faugeras, "3-D Scene Representation as a Collection of Images and Fundamental Matrices," *INRIA Technical Report 2205*, February 1994.
11. L. McMillan, "Computing Visibility Without Depth," *UNC Chapel Hill Technical Report TR-95-047*, 1995.
12. J. J. Le Moigne and A. M. Waxman, "Structured Light Patterns for Robot Mobility," *IEEE Journal of Robotics and Automation*, vol. 4, no. 5, pp. 541—546, 1988.
13. S. Pollard, S. Hayes, M. Pilu, and A. Lorusso, "Automatically Synthesizing Virtual Viewpoints by Trinocular Image Interpolation," *Hewlett-Packard Labs Technical Report HPL-98-05*, January 1998.
14. J. L. Posdamer and M. D. Altschuler, "Surface Measurement by Space Encoded Projected Beam Systems," *Computer Graphics and Image Processing*, vol. 18, no. 1, pp. 1—17, January 1982.
15. K. Sato and S. Inokuchi, "Three-Dimensional Surface Measurement by Space Encoding Range Imaging," *Journal of Robotic Systems*, vol. 2, no. 1, pp. 27—39, Spring 1985.
16. S. Seitz and C. Dyer, "Physically-valid View Synthesis by Image Interpolation," *Proceedings of the IEEE Workshop on Representations of Visual Scenes*, Cambridge, MA, pp. 18—25, 1995.
17. S. Seitz and C. Dyer, "View Morphing," Proceedings of SIGGRAPH, New Orleans, LA, pp. 21—30, August 1996.