Interactive 3-D Media with Structured Light Scanning

Nelson L. Chang
Imaging Systems Laboratory
HP Laboratories Palo Alto
HPL-2003-112
May 29th , 2003*

E-mail: nelson.chang@hp.com

interactive 3-D media, structured light scanning, multiframe correspondence estimation, 3-D shape recovery, interactive view synthesis, view interpolation, active calibration

The core to many 3-D related applications is establishing reliable dense correspondences among the multiple cameras used in a given imaging system. This paper proposes a structured light scanning technique for directly solving the difficult multiframe correspondence problem across any number of cameras without any searching or calibration. The technique consists of temporally encoding positional information of projector space and capturing the projected result with one or more fixed cameras. In contrast to traditional image-based approaches, the proposed technique does not rely on consistent textures across frames. Furthermore, it automatically determines visibility across all cameras in the system and scales linearly in computation with the number of cameras.

The proposed technique forms the core of a complete framework to synthesize new views of a given 3-D scene at interactive rates. An efficient interpolation method using barycentric coordinates renders new views at interactive rates from the computed correspondences. The light projector is shown to serve as an additional viewpoint for interpolation, thereby expanding the range of synthesizable views. The paper also creates an intuitive user interface for specifying the desired virtual view in the absence of any 3-D geometry . Experimental results using real-world data demonstrate the effectiveness of the proposed framework for creating interactive 3-D media.

* Internal Accession Date Only
Approved for External Publication

1. Introduction

Interactive 3-D media is becoming increasingly important as a means of communication and visualization. Photorealistic content like "rotatable" objects and panoramic images transmitted over the Internet provide the end user with limited interaction and give a sense of the 3-D nature of the modeled object or scene. Such content helps markets like e-commerce and commercial real estate by making the product of interest appear more realistic and tangible to the customer.

To develop an effective framework for interactive 3-D media, a number of issues need to be addressed:

- *How to create realistic looking and parallax-corrected virtual views?*
- *How to quickly and robustly capture a 3-D scene to automatically produce interactive media?*
- *How to render these views at interactive rates?*
- *How to represent an intuitive interface for the user to specify views?*

A traditional approach is to estimate 3-D models and then reproject the results to create new views [Faugeras, 1994]. This approach is often computational and slow, sometimes requiring considerable human intervention to achieve reasonable results. Moreover, accurate 3-D measurement may not be necessary since the end goal is ultimately an image.

More recently, image-based rendering (IBR) techniques have focused on using images directly for synthesizing new views. Chen and Williams [1993] and Seitz and Dyer [1996] interpolate two basis images to synthesize new views. Hansard and Buxton [2000] estimate a parametric function used to interpolate two views. Laveau and Faugeras [1994] exploit constraints of weakly calibrated image pairs. Avidan and Shashua [1998] use trifocal tensors for view synthesis. Pollard et al. [1998] match edges in three views and then interpolate. These IBR techniques perform well at the first issue of synthesizing good-looking views. However, they assume dense correspondences have already been established, and in some case, use complex rendering to synthesize new views offline.

In contrast, this paper addresses the problem of reliably computing dense correspondences for a static 3-D scene across any number of images in an efficient manner and without requiring calibration. Instead of using image information alone, an active structured light scanning technique is proposed to solve the difficult multiframe correspondence problem. To simplify computation, correspondences are first established with respect to the light projector's coordinate system, herein referred to as projector space, consisting of a rectangular grid with *w*x*h* connected rectangular regions. The resulting correspondences may be used to create interactive 3-D media, either directly for view synthesis or with calibration information for recovering 3-D shape.

2. Related Work

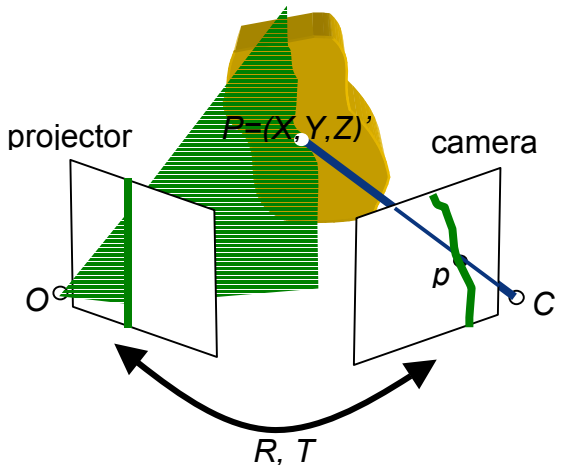

Figure 1. *Structured light scanning intersects a given 3-D ray with its corresponding projected light plane to estimate shape.*

The proposed algorithm solves the difficult multiframe correspondence problem using structured light scanning. The following subsections discusses related work in both of these fields.

2.1. Multiframe Correspondence Estimation

Solving the correspondence problem is a classic problem in computer vision and image processing literature. It is central to many 3-D related applications including stereopsis, 3-D shape recovery, camera calibration, motion estimation, view synthesis, and others. Simply stated, the problem is to find the mapping relating points in one coordinate system to those in a second (or more) coordinate system.

The traditional method for solving the correspondence problem is to use image information directly. The correspondence mapping is determined for every pixel by examining its neighborhood and exploiting color consistency. One optimizes some objective function (e.g. maximize correlation or minimize some error metric) for a given point in one image to find its corresponding match in all the other images [Faugeras, 1994]. Typical approaches include matching algorithms, stereo algorithms, and optic flow [Chang, 1999].

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 produces only a sparse set of correspondences reliably. 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.

2.2. Structured Light Scanning

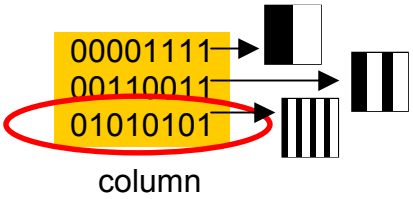

Figure 2. *An eight-column example illustrating the formation of light patterns to encode each column's position.*

Structured light scanning algorithms traditionally use a calibrated camera-projector pair to recover 3-D shape information [Batlle et al., 1998]. A single active column in projector space results in a plane of light. The active plane of light is projected onto the scene of interest and the resulting contour is imaged by the camera. Then, for any image pixel ***p*** on the contour, 3-D point ***P****=(X,Y,Z)'* is found by intersecting the 3-D ray passing through ***p*** and the 3-D plane equation; an example is shown in Figure 1.

Instead of specifying each plane individually, one can specify the planes in parallel using a set of temporally encoded patterns [Posdamer and Altschuler, 1982]. One can form light patterns simply by considering the bitplanes of the binary representation of the column indices. An eight-column example is shown in Figure 2. Each column consists of a three-bit binary representation. Three light patterns are formed by taking each bitplane horizontally.

The spatially varying light patterns form a temporal encoding of the columns in projector space. Every light pattern, consisting of binary vertical stripes of varying widths, is projected in succession. The camera decodes the projected patterns and builds up the appropriate bit sequence at every pixel location. Hence, given a pixel location in the camera, one can instantly determine the corresponding column in projector space, and then calculate the corresponding 3-D point. Note one achieves greater 3-D accuracy by considering the transitions between adjacent columns.

The above approach works well for arbitrary static scenes with a calibrated setup. There have been many approaches to further improve on the design of the projected light patterns and reduce the total number used [Batlle et al., 1998]. Le Moigne and Waxman [1988] project an uncoded grid pattern with landmark dots, an approach that requires a labeling process and a particular camera configuration. Boyer and Kak [1987] use a single color pattern to handle only neutrally colored scenes. Other approaches like [Hall-Holt and Rusinkiewicz, 2001; Devernay et al, 2002] use sophisticated binary patterns and matching algorithms to capture dynamic but not very textured scenes.

3. Proposed Solution: LUMA

Instead of requiring camera-projector calibration, this section proposes using structured light scanning for recovering the correspondence mapping between a camera and the projector directly. Unlike typical structured light scanning approaches, the so-called LUMA technology

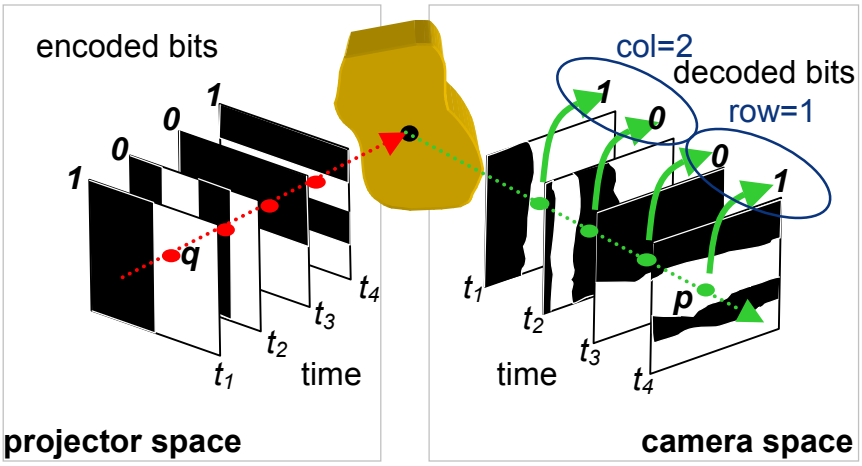

Figure 3. *With LUMA, one simply decodes the captured light patterns to identify corresponding points in projector space.*

encodes not only the *w* columns of projector space, but also the *h* rows, thereby providing a unique temporal code for the *wxh* connected rectangular regions in projector space. It accomplishes this task by using a set of horizontally striped and vertically striped binary light patterns, both of varying widths. Without loss of generality, it is assumed that *w=h*, thus only $2*\log_2(w)$ total light patterns are needed. Figure 3 shows an example of using LUMA to encode location (2,1) in a 4x4 projector space.

There are a couple of observations to make about these coded light patterns. First, the patterns encode only the mapping to a particular projector space rectangle and not to specific points within the rectangle. Consider a 4x4 projector space as shown in Figure 4. After decoding, the set of image points A' is assigned to rectangle A in projector space, however the exact point-to-point mapping remains unclear. Instead of mapping interior points, one can exploit the connectedness of the projector space rectangles and map the corner points that border any four neighboring projector space rectangles. For example, the corner point ***p*** that borders A,B,C,D corresponds to the image point that borders A',B',C',D', or in other words, the so-called imaged corner point ***p'***. Imaged corner points may be found more easily and accurately than matches to points within any projector space rectangle.

In addition, the coded light patterns exhibit a natural hierarchical spatial resolution ordering, which dictates the size of the projector space rectangles. The patterns are ordered from coarse to fine, and each associated pair of vertical-horizontal patterns at the same scale subdivides the projector space by two in both directions. Figure 5 shows an example of six patterns that encode an 8x8 projector space. Using the coarsest two patterns alone results in only a 2x2 projector space. Adding the next pair of patterns increases the projector space to be 4x4, with every rectangle's area reduced by a fourth, and likewise for the third pair of patterns.

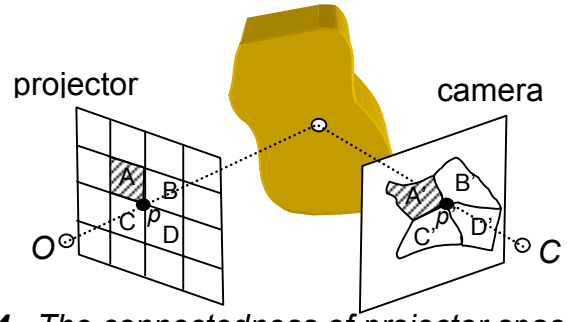

Figure 4. *The connectedness of projector space rectangles may be used to pinpoint correspondences of corner points.*

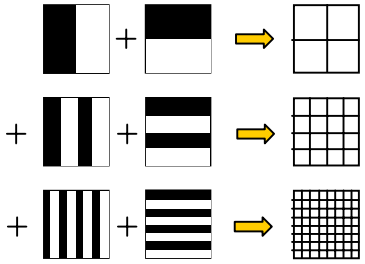

Figure 5. *Projector space rectangles decrease in area with additional finer resolution light patterns.*

With these observations in mind, recovery of the correspondence mapping is straightforward with LUMA. The projector displays the coded patterns in sequence onto the scene of interest. For each light pattern, the camera captures the projected pattern and then decodes the symbol at every pixel location. After all the patterns have been projected, the pixel locations in the camera consist of a decoded bit sequence that specifies a particular rectangle in projector space.

In contrast to previous work [Chang, 2002], LUMA proceeds to use the corners of connected projector space rectangles to unambiguously and more accurately establish the correspondence mapping. Since it may be difficult to locate every corner's match at the projector space resolution, LUMA finds each corner's match at the coarsest possible resolution and interpolates to finer resolutions where necessary. In Figure 6, it is easier to find the four imaged corners (A',B',D',E' is one such corner) at resolution level *l* than at level *l+1*. Other corner points at level *l+1* are found directly or interpolated from these four imaged corners. In the end, subpixel estimates of the imaged corners at the finest projector space resolution are established. The mapping can be similarly established for any number of cameras in the system. Thus, LUMA obtains an accurate correspondence mapping from every camera to projector space, resulting in the implicit correspondence mapping among any pair of cameras.

The LUMA technology has a number of benefits. It provides an automatic method for estimating dense correspondences between any camera and projector space. Unlike image-based approaches, LUMA does not rely on distinct and consistent textures, nor does it produce spurious results for uniformly colored objects. Because of the problem

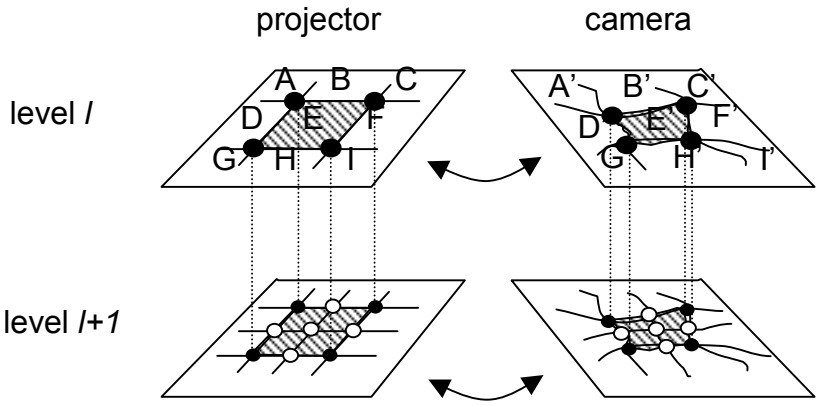

Figure 6. *Analyzing corners at one resolution level helps to locate corners at a finer level.*

formulation, LUMA supports any number of cameras, efficiently solves the difficult multiframe correspondence problem for a static 3-D scene, determines visibility and occlusions across all cameras, and scales linearly with the number of cameras. LUMA uses the same cameras to capture color and shape. Because it does not require calibration, there are no restrictions on the physical locations of the camera(s) and the projector. Moreover, there are no restrictions on the type of 3-D scenes it can handle since it uses robust binary patterns. The following section describes the technology in greater detail.

4. LUMA System Details

The LUMA system consists of several steps that are detailed in this section. 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. 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 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 bit sequence function corresponding to pattern *n* at point $\boldsymbol{p}$ in camera *k*. Note that the decoded bit sequence is given by the set of bits $\{S_k(\boldsymbol{p};0), S_k(\boldsymbol{p};1),\ldots, S_k(\boldsymbol{p};N-1)\}$. 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 multiframe correspondence problem is then equivalent to the following: project a series of light patterns $I_0(\cdot;n)$ and use the captured images $I_k(\cdot;n)$ to compute the bit sequence functions $S_k(\boldsymbol{p};n)$ 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

$$M_{ij}(\mathbf{p})=M_{0j}(M_{i0}(\mathbf{p}))$$

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

1. 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.
2. Project and capture reference patterns. An ambient-only ("all black") pattern and a full-illumination ("all white") pattern correspond to the first two patterns projected $I_0(\cdot;0)$ and $I_0(\cdot;1)$, respectively. The captured patterns $I_k(\mathbf{p};0)$ and $I_k(\mathbf{p};1)$ provide the approximate mean color vector for the "black" (0) and "white" (1) symbols respectively at image pixel $\mathbf{p}$ in camera *k*. Because of illumination variations and different reflectance properties across the object, the mean color vectors may vary across image pixels in every camera. Note that $I_k(\cdot;0)$ can also serve as the color information in step 1 for camera *k* as long as there is sufficient ambient light in the scene.
3. Create the series of light patterns to be projected. This series includes the $\log_2(w)$ 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. The number of patterns *N* is then $\log_2(w)+\log_2(h)+2$, including the two reference patterns. These patterns correspond to $I_0(\cdot;m)$, where *m* varies from *2* to *N-1* as the pattern width decreases for each vertical and horizontal pattern pair. The patterns are Gray coded to reduce spatial frequency [Sato and Inokuchi, 1985].
4. Determine validity maps for every camera. For every pixel $\mathbf{p}$ in camera *k*, $V_k(\mathbf{p})=1$ if $|I_k(\mathbf{p};0)-I_k(\mathbf{p};1)|>T$ and 0 otherwise, where *T* is some preset global threshold. The invalid pixels correspond to points that lie outside the projected space or that do not offer enough discrimination between the "black" and "white" symbols (e.g. regions of black in the scene absorb the light).
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 *2* to *N-1*. It is important to ensure that there is proper synchronization between the projector and the cameras. Figure 7 shows captured images $I_1(\cdot;14)$ and $I_1(\cdot;15)$ for the "Cap" image pair described in Section 8.
6. For every light pattern, decode the symbol at each valid pixel in every image. Let $S_k(\mathbf{p};m)$ be the decoded bit taking on value 0 or 1. 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 $m=2,\ldots,N-1$. Every valid pixel is assigned the symbol (0 or 1) 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

Figure 7. *Captured light patterns m=14 and m=15 from camera 1 for the "Cap" image pair.*

zero. A projector space location $\boldsymbol{q}$ is said to be *marked* if and only if $B_k(\boldsymbol{q})=1$. For a given resolution level l, one follows these substeps for every camera k:

a. *Convert bit sequences of each image point to the corresponding projector space rectangle at the current resolution level*. For all valid points $\boldsymbol{p}$, the corresponding column c in the 2^{l+1}x2^{l+1} projector space is simply the concatenation of the decoded bits $S_k(\boldsymbol{p};2*s+2)$, and the corresponding row r is the concatenation of the remaining bits $S_k(\boldsymbol{p};2*s+3)$, for $s=0,1,\ldots,l$. Hence, $M_{k0}(\boldsymbol{p})=(c,r)$.

b. *Locate imaged corner points corresponding to unmarked corner points in projector space*. Suppose valid point $\boldsymbol{p}$ in camera k maps to unmarked point $\boldsymbol{q}$ in projector space. Then, $\boldsymbol{p}$ is an imaged corner candidate if there are image points within a 5x5 neighborhood that map to at least three of $\boldsymbol{q}$'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 $\boldsymbol{q}$ 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 resolution levels* $l<\log_2(w)$. 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

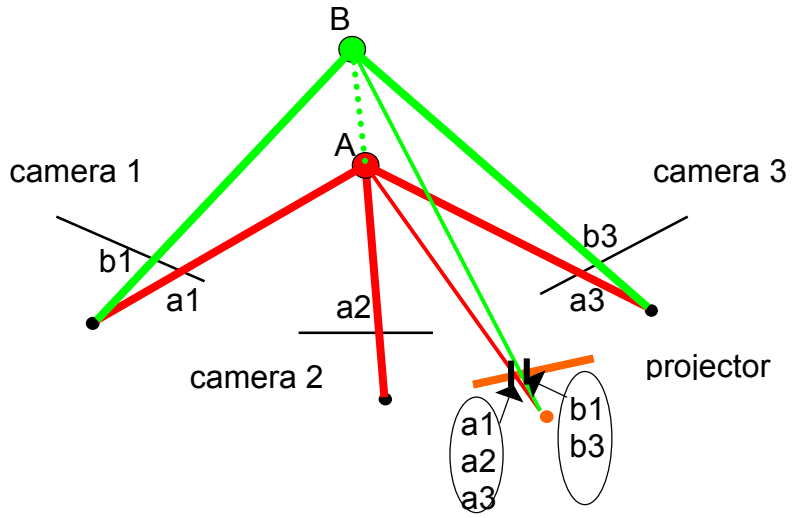

Figure 8. *A 2-D example of LUMA's feature to automatically determine correspondence and visibility across multiple cameras.*

automatically results in a parallax-corrected view of the scene from the projector's viewpoint.

In the end, one obtains the 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 $\mathbf{p}$.

The approach automatically computes correspondence, occlusions, and visibility among all cameras without additional computation. Figure 8 depicts a three-camera 2-D example with scene points A and B. Decoding each camera in succession, one finds various points in each image and their corresponding match in projector space (points *a1*, *a2*, and *a3* all map to the first location in projector space, whereas *b1* and *b3* map to the second location). No points in camera 2 are found to decode to the second location since B is occluded from view. One can easily build up an array in projector space that keeps track of the original image points and traverse this array to determine correspondences. The first location in projector space contains image points *a1*, *a2*, and *a3*, suggesting that (a) all three cameras can see this particular point A, and (b) the three image points all correspond to one another. The second location in the projector space contains image points *b1* and *b3* implying that (a) cameras 1 and 3 can view this point B, (b) B must be occluded in camera 2, and (c) only b1 and b3 correspond.

It is worth noting that the above algorithm may be extended to handle multi-colored patterns. Instead of using binary black-white patterns, one could just as easily employ multiple color bases (e.g. white, red, green, blue), thereby reducing the overall number of patterns needed. In preliminary tests, it has been found that black and white patterns appear to be more robust for a wider range of 3-D objects and scenes.

Figure 9. *The experimental LUMA system uses a projector and one or more cameras.*

5. Experimental Results

LUMA can directly contribute to many applications that require one or more cameras. It is used to compute dense multiframe correspondences for various real-world scenes and to create compelling 3-D media. In addition to simplifying the capture process, LUMA significantly reduces computation by defining the correspondence mappings with respect to the common projector space. The following sections demonstrate the applicability and flexibility of LUMA to a number of applications including interactive view synthesis, camera calibration, and 3-D shape recovery.

In all cases, the imaging system consists of a single 1.1 GHz Pentium III notebook computer, a number of Point Grey Research's Dragonfly cameras (640x480, 30 fps) all connected to a single Firewire/IEEE-1394 bus, and one DLP light projector displaying at XGA (1024x768) resolution; an example of the LUMA system is shown in Figure 9. The computer serves to control the cameras and the projector. The computer runs custom capture and visualization software written in Microsoft MFC Visual C++ and OpenGL. The following results use only two or three cameras, but LUMA easily supports any number of cameras limited only by the bus throughput. The cameras capture both color and correspondence information simultaneously in ambient light.

In the present implementation, the scene is assumed to be stationary and only a single 3-D mesh of the scene is captured. Only 22 light patterns (including the two reference patterns) are needed to encode the 1024x1024 projector space. Because of projector-camera latency, the complete system is currently limited to about 5 fps to ensure proper synchronization. The capturing process thus takes less than five seconds, and the analysis process described in Section 4 requires less than a minute.

6. Application: Interactive View Synthesis

The multiframe correspondence mapping obtained by LUMA may be directly used as a form of interactive 3-D media. Using the images captured at every camera and the associated correspondence mapping, one can easily perform view interpolation between any pair of images to synthesize viewpoints that were not originally captured. It gives the user limited interaction and manipulation of a 3-D scene without having to perform any calibration.

An intermediate view between two anchor images is constructed by examining projector space for points corresponding to the anchor images and then computing the weighted average of the pixel information (both location and color). A parameter specifies the relative distance away from the anchor views. It should be clear that this interpolation provides a smooth transition between the anchor 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. In this formulation, only scene points that are visible in the anchor images (i.e. scene points that lie in the intersection of the cameras' visibility spaces) may be properly interpolated.

One can proceed to render virtual views of the captured scene given a set of images and corresponding dense correspondences. Instead of merely performing view interpolation to recover intermediate views, the given LUMA data may also be used for extrapolating views outside the set of basis images. Moreover, the views should be generated at interactive rates to provide a responsive system for the user. The section details the proposed solution to this so-called *interactive view synthesis* problem, and, without loss of generality, it considers only three images for simplicity.

Given the triplet corresponding to the same scene point visible in all three images, one needs a way to predict the coordinates of the scene point in a virtual view. A natural choice is to extend conventional linear interpolation to three images. To this end, barycentric coordinates are used to weight the triplet to come up with the new location, where the weights sum to one and can be negative. The same weights are then used on all such triplets to form a consistent virtual view.[1]

This representation leads to a natural interface for specifying new views, especially important in the absence of any 3-D geometry. As shown in Figure 10, the so-called *interface triangle* gives an abstract 2-D representation of the arrangement of the cameras. The user controls a point with respect to the interface triangle defined in the plane. The barycentric coordinates of the point are then used to weight all triplets to synthesize the appropriate image. The steps of the complete algorithm are as follows:

1. Construct interface triangle $\Delta\boldsymbol{xyz}$: Let $\boldsymbol{x}=(0,0)$. Define $\boldsymbol{y}-\boldsymbol{x}$ to be the median of correspondence differences between cameras 2 and 1, and likewise, $\boldsymbol{z}-\boldsymbol{y}$ for cameras 3 and 2**.**
2. Define user-specified point $\boldsymbol{w}=(s,t)$ with respect to $\Delta\boldsymbol{xyz}$.

[1] The virtual view corresponds to a physically valid in-between view with parallel affine cameras [Seitz and Dyer, 1996], or nearly so, with small rotation [Pollard et al., 1998].

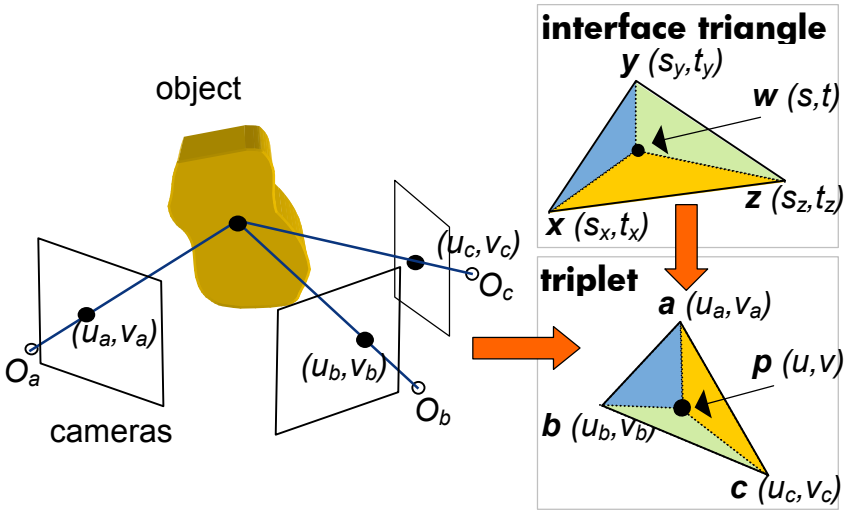

Figure 10. *Point **w** and interface polygon **xyz** specify the new coordinates **p** of a scene point associated with the triplet (**a**,**b**,**c**).*

3. Determine barycentric coordinates (α,β,γ) corresponding respectively to the weights for vertices ***x***, ***y***, ***z***:
   a. Compute signed areas of subtriangles formed by vertices and ***w***, i.e. SA(***x***,***y***,***w***), SA(***y***,***z***,***w***), SA(***z***,***x***,***w***), where for vertices ***x***=(s_x,t_x), ***y***=(s_y,t_y), ***z***=(s_z,t_z),
   $$SA(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \tfrac{1}{2}((t_y-t_x)s_z+(s_x-s_y)t_z+(s_yt_x-s_xt_y))$$
   Note that it is positive if the vertices are oriented clockwise and negative otherwise.
   b. Calculate (possibly negative) weights α, β, γ based on relative subtriangle areas, such that
   $$\alpha = SA(\boldsymbol{y},\boldsymbol{z},\boldsymbol{w}) / SA(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$$
   $$\beta = SA(\boldsymbol{z},\boldsymbol{x},\boldsymbol{w}) / SA(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$$
   $$\gamma = SA(\boldsymbol{x},\boldsymbol{y},\boldsymbol{w}) / SA(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = 1 - \alpha - \beta$$
4. For every triplet $(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})$ of corresponding image coordinates, use a weighted combination to compute the new position ***p***=(u,v) relative to $\Delta\boldsymbol{abc}$, i.e.
   $$u = \alpha\, u_a + \beta\, u_b + \gamma\, u_c$$
   $$v = \alpha\, v_a + \beta\, v_b + \gamma\, v_c$$
   Note that the new color vector is similarly interpolated.

The algorithm automatically performs three-image view interpolation for interior points of the interface triangle, reduces to pairwise view interpolation along the perimeter, and, in fact, executes view extrapolation for exterior points, all without any calibration.

The algorithm runs in real-time after quantizing α and β such that only integer and bitwise operations are used in the interpolation equation. In particular, the parameter space is discretized to reduce the number of allowable views. Each real parameter interval [0,1] is remapped to the integral interval [0,N], where $N=2^n$ is a power of two. Let a=(int)(α N) and b=(int)(β N) be the new integral view parameters. Then, the generic interpolation expression quantities p_a, p_b, p_c may be reduced as follows:

$$p = \alpha\, p_1 + \beta\, p_2 + \gamma\, p_3 = p_3 + \alpha\,(p_1 - p_3) + \beta\,(p_2 - p_3)$$
$$= (N\, p_3 + \alpha\, N\,(p_1 - p_3) + \beta\, N\,(p_2 - p_3)) / N$$
$$\approx ((p_3 \ll N) + a\,(p_1 - p_3) + b\,(p_2 - p_3)) \gg n$$

where '«' and '»' refer to the C/C++ operators for 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.

To handle occlusions without knowing depth, one can analyze correspondence vectors or use McMillan's rendering algorithm to ensure the correct order of points [McMillan, 1995]. Alternatively, one can calibrate from correspondences to estimate depth information and then use z-buffering.

The proposed solution to interactive view synthesis is more flexible and less complex than traditional image interpolation approaches. It works well for interpolating and even extrapolating views with three images, and it can support more than three images by arranging the layout of images into connected triangles. In contrast to [Hansard and Buxton, 2000], it is a parameterized view synthesis technique that offers a second degree of freedom in selecting the virtual view.

There are additional benefits when used in conjunction with LUMA for obtaining correspondences. By construction, all correspondence information is defined with respect to projector space. This fact allows LUMA to quickly identify the scene points visible by all cameras, leading to much faster rendering. Furthermore, the projector itself may be viewed as a virtual camera by warping image information back to projector space and synthesizing the projector's viewpoint. This new image serves as an additional basis image for view synthesis, thereby creating a way to interpolate views from even a single camera.

Using real-world sequences, one can use LUMA to create some compelling synthesized views. A custom interactive viewer renders the appropriate view in real-time based on the user's input, thereby simulating manipulation of the captured object. The first example in Figure 11 uses three cameras to capture a human subject in the "Face" triplet. These three images serve as the basis images to produce the synthesized views shown in Figure 12. In this example as well as the subsequent ones, the dark regions that appear correspond to portions of the dropcloth behind and possibly supporting the captured object; note that LUMA properly determines the correspondences for these points as well. The interface triangle, located in the lower right corner of each image, shows the user's viewpoint as a dark circle and its relative distance to each camera or vertex. The upper left image is a view interpolation between the left and right cameras alone. The upper right image shows the result of interpolating using all three cameras. The bottom two images are extrapolated views that go beyond the interface triangle. The subject's face and shirt are captured well and appear to move naturally. The neck does not appear in the results since it is occluded in the top camera. Because of the wide separation among cameras, the resulting views are quite dramatic especially when seen live.

Figure 11. *The "Face" image triplet.*

The system also works well for scenes consisting of multiple objects. Figure 13 shows a three-camera example consisting of a bowling pin, banana, and globe in the "Mixed" triplet. Again, using the three original views as basis images, the interactive viewer generates the synthesized views found in Figure 14. The correspondences are obtained automatically and without any explicit segmentation. The resulting views are reasonable and the objects appear to move in a consistent manner. The small gaps in the globe arise because of sampling issues and may be filled in by using more cameras or better arranged ones.

The real power of the proposed framework can be demonstrated in the last example. Figure 15 shows a Buddah mask captured by only two cameras in the "Buddah" image pair. The pair are used to form the projector's synthesized viewpoint with scaled dimensions shown as the lower image of the image triplet. Figure 16 shows typical synthesized views. With the original two images, one can perform only view interpolation like the upper left image. However, using the projector's viewpoint as a third basis image, one can create a broader set of views with more motion parallax as seen in the other three synthesized views.

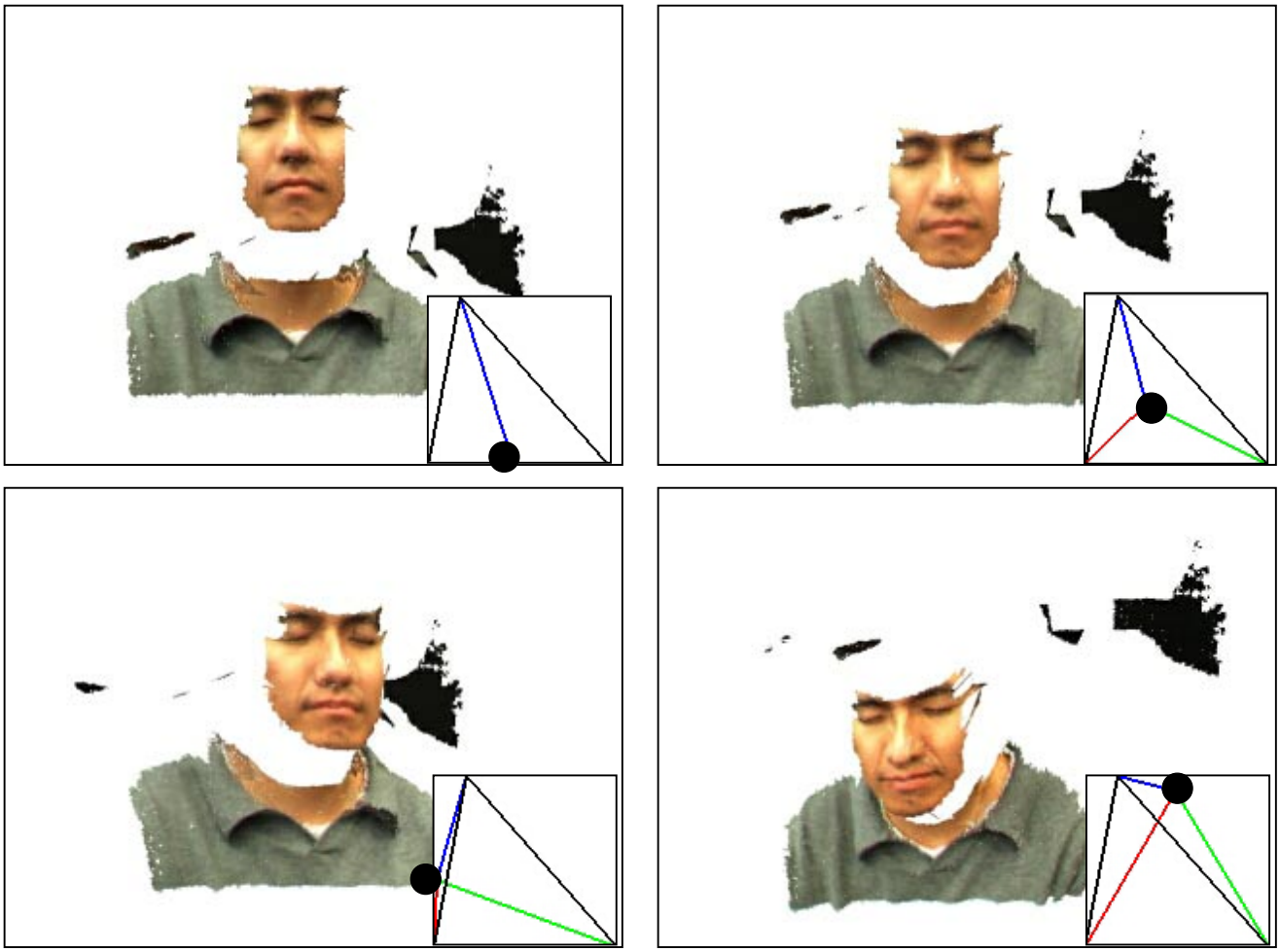

Figure 12. *View synthesis results with the "Face" image triplet.*

It should be clear from this last example that LUMA can be used to generate even stereoscopic image pairs from a single camera. One simply needs to place the projector in the appropriate location next to the camera to simulate a binocular set up. LUMA proceeds to solve the dense correspondence mapping between the camera frame and the projector space. The correspondence mapping is then used to synthesize the image from the projector's viewpoint. Thus, the image captured by the camera and the synthetic projector's viewpoint together form a stereo pair. To achieve a lower cost solution, a modified patterned flash could be used instead of the projector to encode the positional information of the projector's space.

7. Application: Camera Calibration

The LUMA technology can also be used as a front end for active camera calibration. It is particularly useful for systems consisting of more than one camera. Instead of using a patterned calibration target and various image processing techniques to establish feature correspondences across all the images, one can apply LUMA to automatically solve for

Figure 13. *The "Mixed" image triplet.*

correspondences among all cameras. A featureless rigid planar surface (e.g. projection screen, whiteboard, box) serves as an adequate target. Because LUMA automatically determines occlusions and visibility across all the images, only the points that are visible in all images are used for calibration. The number of such points is considerably larger than that obtained with passive techniques. Arbitrary world coordinates are assigned to each position in projector space, and the information is fed into standard nonlinear optimization techniques to generate the calibration parameters [Tsai, 1987; Zhang, 2000]. It is worth noting that the light source is assumed to have negligible projection distortions (e.g. nonlinear lens or illumination distortion); one should account for these parameters for a complete calibration process.

8. Application: 3-D Shape Recovery

With a calibrated set up[2], one can easily convert the correspondence mapping obtained by LUMA into 3-D information. If the projector's location is known, 3-D shape can be computed

[2] Active calibration from Section 7 is used to obtain dense correspondences for a featureless planar target, and these correspondences are fed into Zhang's calibration algorithm [2000].

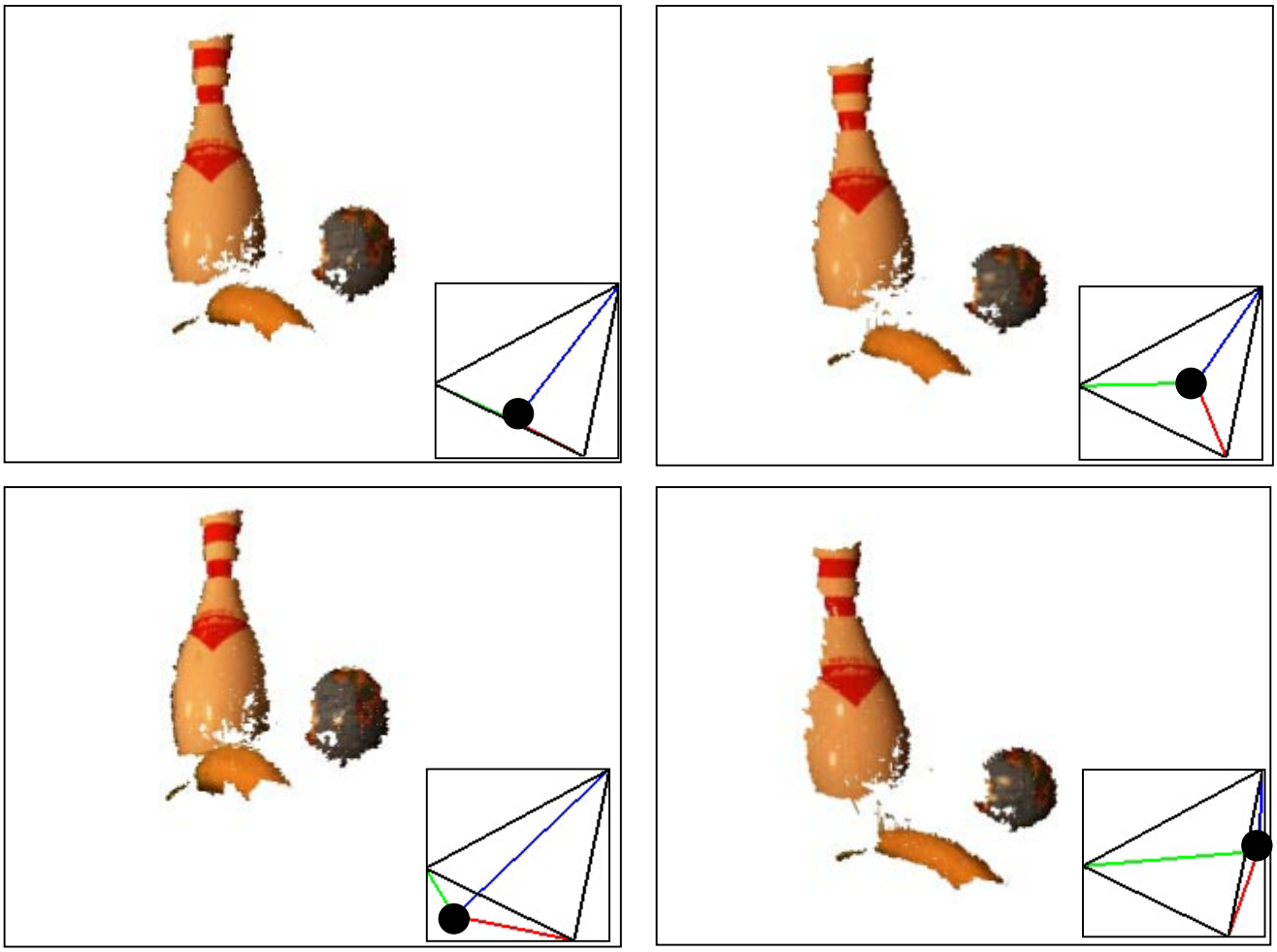

Figure 14. *View synthesis results with the "Mixed" image triplet.*

using the ray-plane intersection technique described in Section 2.2. However, using LUMA with multiple cameras, the projector's location does not ever have to be known, and the projector can in fact be arbitrarily placed. One can instead use least squares triangulation to compute the local 3-D coordinates for a particular scene point given at least two corresponding image points referring to the scene point, The result is a cloud of 3-D points that can be defined with respect to projector space coordinates or one of the cameras' coordinate systems.

Because of the regular rectangular structure of either coordinate system, one can form a 3-D mesh through these points and tessellate neighboring points together. A triangle patch may be formed by examining neighboring points in the rectangular grid, where each vertex of the patch consists of the 3-D coordinates and color information. Note that some patches are not added to the model, including those whose vertices correspond to points not visible in all the cameras and whose vertices transcend depth boundaries. In the end, one obtains a coherent 3-D mesh (possibly piecewise) model of the scene corresponding to points common to all cameras.

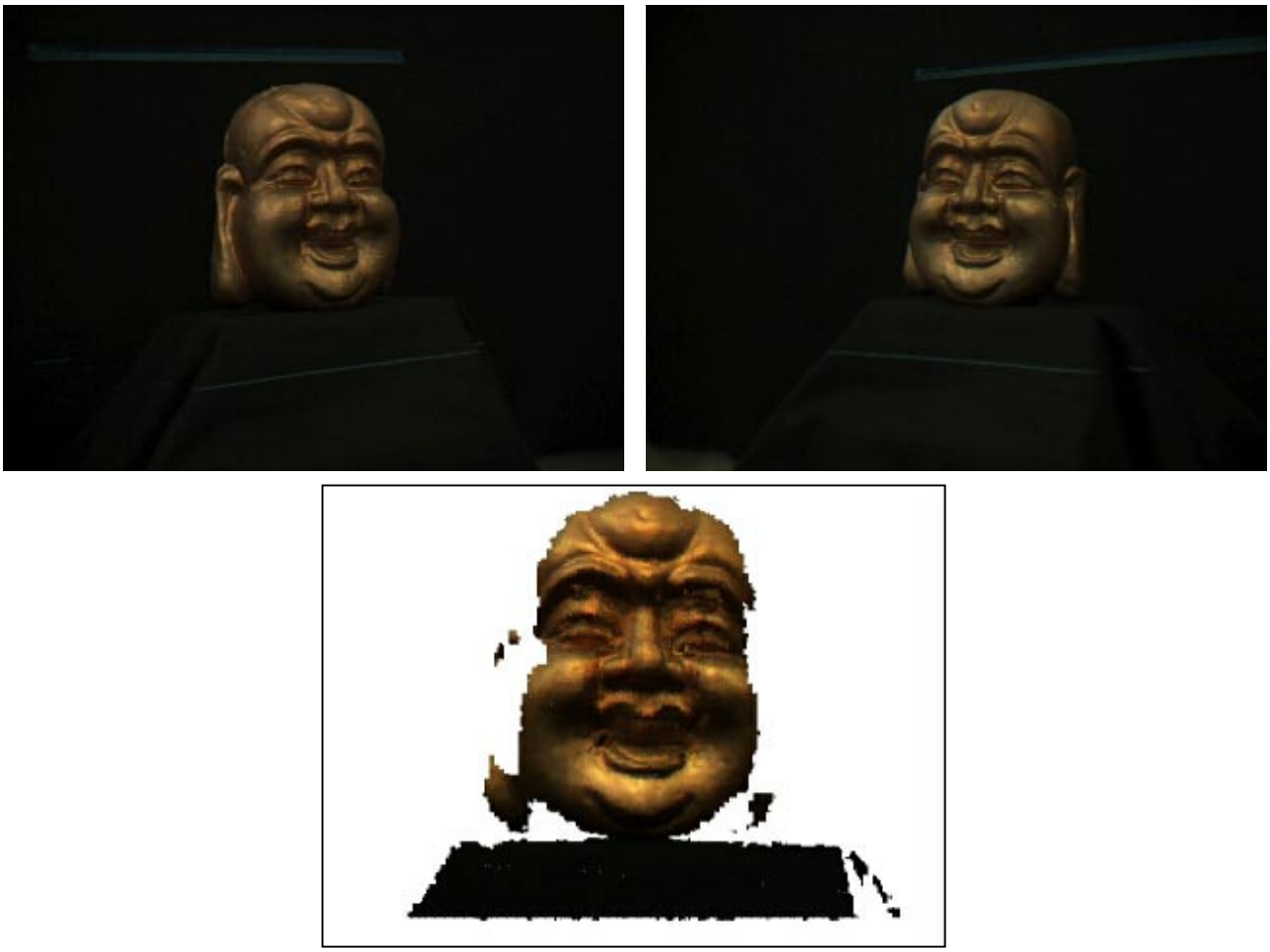

Figure 15. *The "Buddah" image pair and the estimated projector's viewpoint.*

The LUMA system has been used to capture the 3-D shape of various real-world objects. The first example is the "Cap" image pair shown in Figure 17. The set of coded light patterns are projected onto a baseball cap, and two cameras independently decode the patterns to obtain the dense mapping with respect to projector space. As described in the previous section, this mapping may be used directly for view synthesis to allow the user to seemingly manipulate the cap to a limited extent. With calibration parameters, however, one can obtain a true 3-D model for the cap and freely view the object at an arbitrary angle as shown in Figure 18. Notice that the resulting model is smooth and captures the shape quite nicely. The jaggedness along the edge of the model corresponds to points not visible in both cameras.

A second example comes from the "Mixed" image pair shown in Figure 13, where only the bottom two frames are considered as basis images. Figure 19 demonstrates that without any segmentation LUMA can also recover 3-D shape for scenes featuring multiple objects quite well. The small gap in the globe occurs because of the reflection of the ambient light source in the right image.

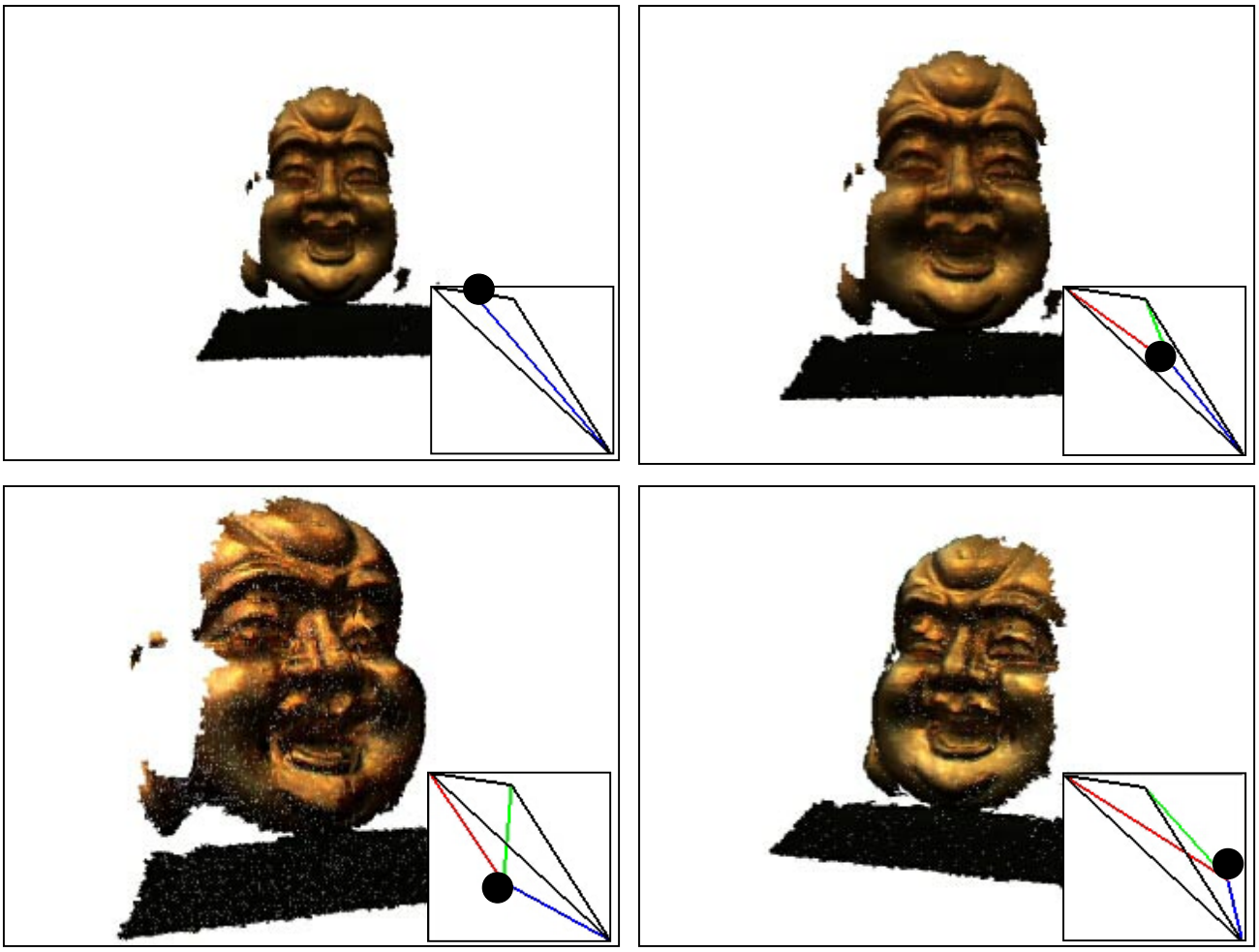

Figure 16. *View synthesis results with the "Buddah" image pair.*

Another example is a human face captured with two cameras. As shown in Figure 20, the so-called "Face2" image pair consists of a human subject. Figure 21 presents the recovered 3-D model with and without the texture map. Note that the human facial structure, including the subject's nose and Adam's apple, has been automatically computed, and even ripples in the clothing have been recovered.

In the final example, a ceramic pink pig is captured by three cameras. As shown in Figure 22, the cameras are organized in triangular formation. LUMA can easily handle the additional cameras since the algorithm computes the correspondence mapping between a given camera and projector space independently of any other camera. The resulting 3-D model with and without texture map can be found in Figure 23. The quality of the reconstructed surface, including the pig's ear and hindquarters, is quite good. It is worth noting that uniformly colored objects like this pig tend to cause image-based approaches to fail because of the lack of distinct texture. Note too that LUMA is able to recover the subtle shape contours of the dropcloth covering the support structure of the pig.

Figure 17. *The "Cap" image pair.*

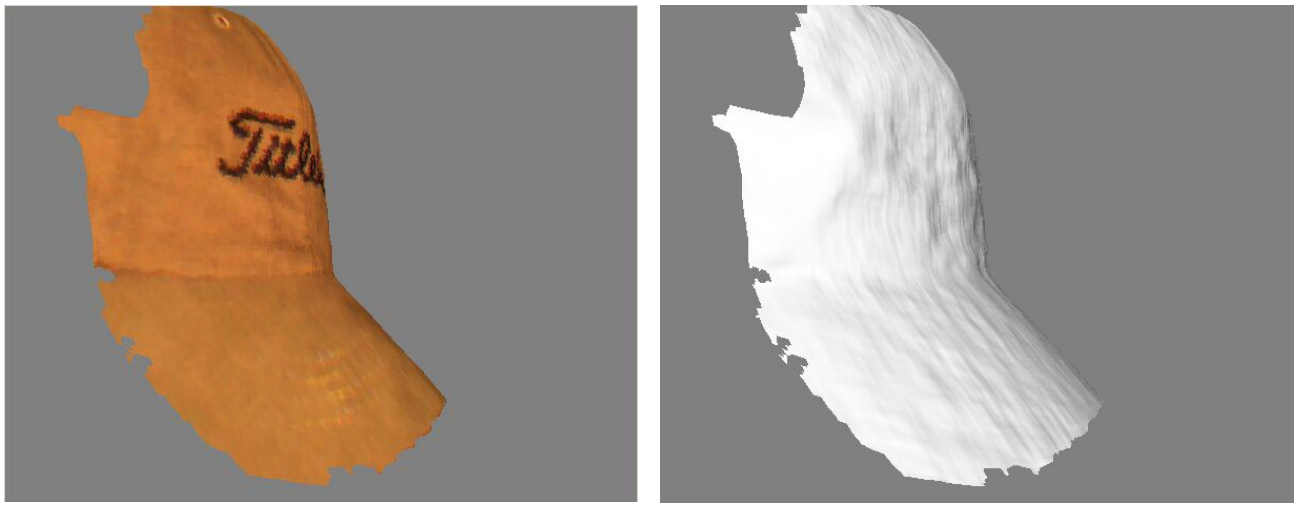

Figure 18. *Reconstructed textured and untextured 3-D model for "Cap".*

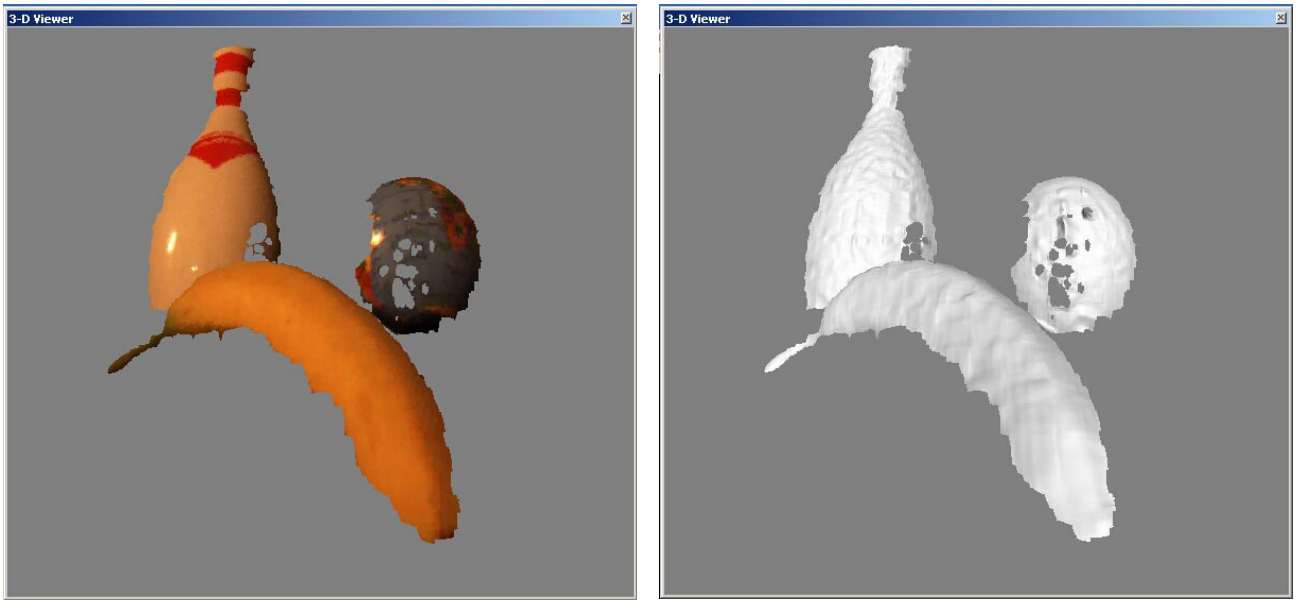

Figure 19. *Reconstructed textured and untextured 3-D model for "Mixed".*

Figure 20. *The "Face2" image pair.*

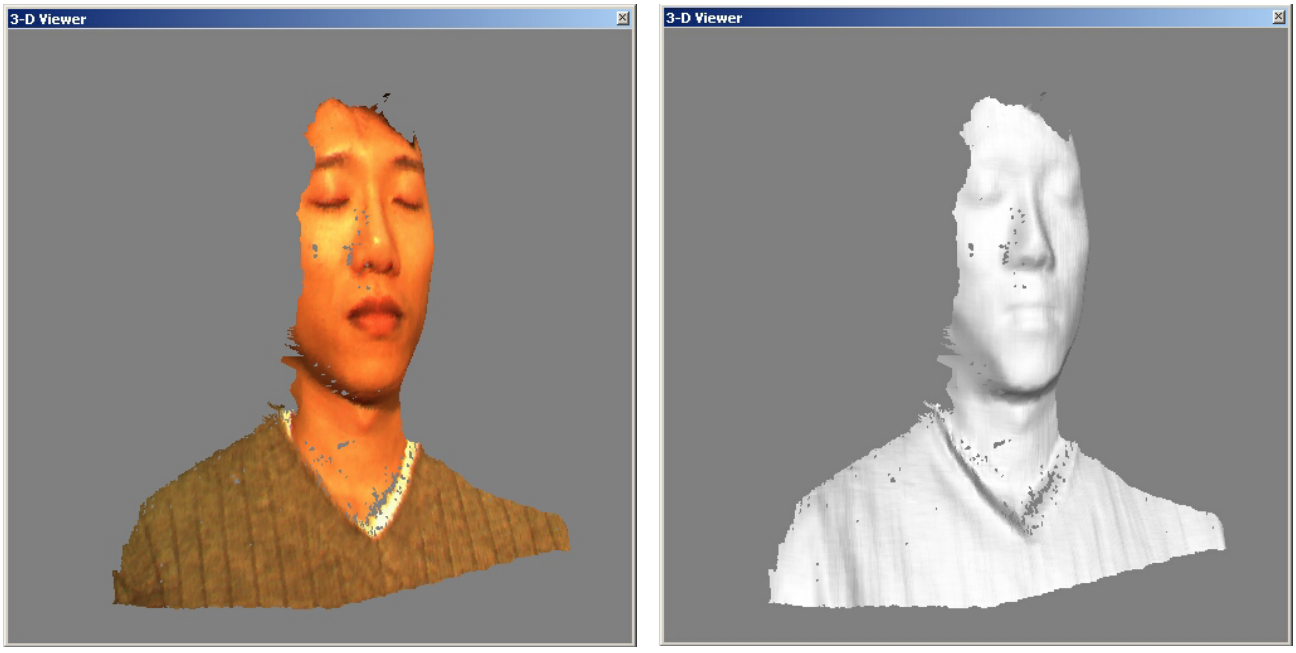

Figure 21. *Reconstructed textured and untextured 3-D model for "Face2".*

Figure 22. *The "Pig" image triplet.*

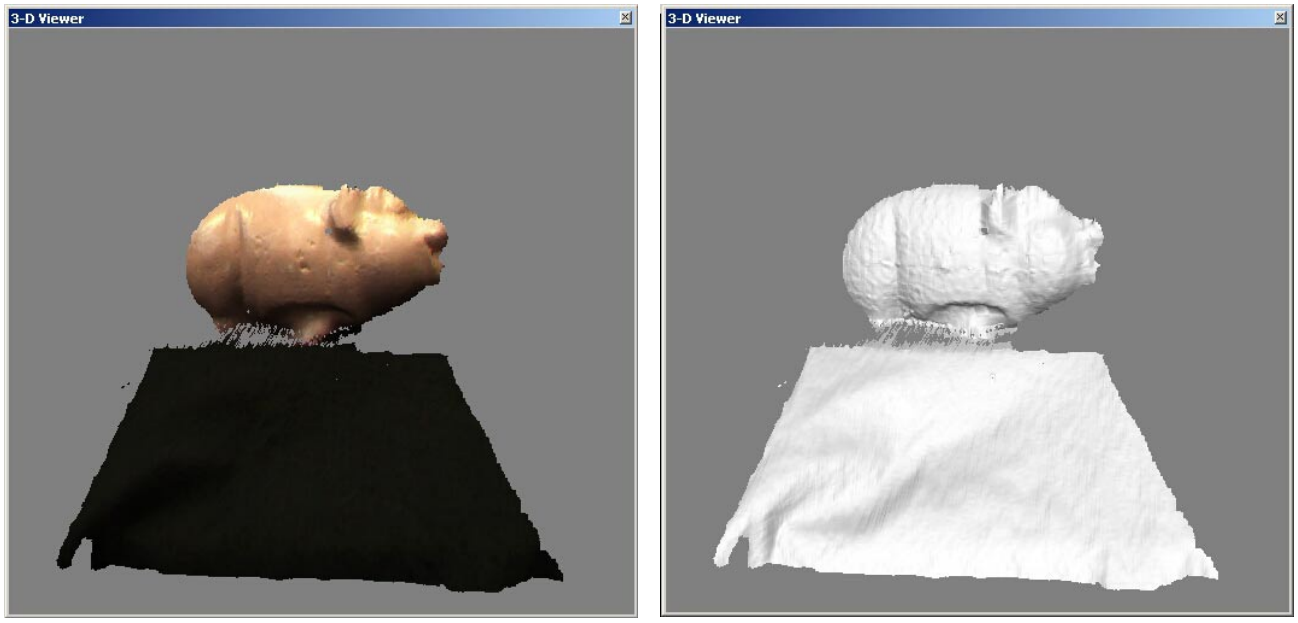

Figure 23. *Reconstructed textured and untextured 3-D model for "Pig".*

9. Conclusions and Future Work

This paper has focused on solving the traditionally difficult problem of establishing dense correspondence across multiple images. The proposed solution uses an efficient and robust structured light scanning algorithm that replaces computational multiframe optimization by simple decoding. The resulting dense correspondence mappings can be directly used as a form of 3-D media with limited interaction (i.e. for interactive view synthesis) or, with calibration information, can compute 3-D models. The approach scales well with the number of cameras in the system. It also offers a good tradeoff between much slower image-based techniques and much more expensive laser scanning solutions for 3-D shape recovery. Experimental results with real-world data suggest the system to be effective for fast creation of visually interactive 3-D media. As a side benefit, the projector can also serve as an additional virtual camera, thus widening the range of synthesizable views.

There are many future directions to pursue. Two primary issues include improving accuracy and speed. Shape accuracy can be improved with multiple acquisitions and higher resolution cameras. Different setups, including larger number of cameras, may simplify scanning of arbitrary 3-D scenes, and lead to complete shape and lighting recovery. Speed can be improved by reducing camera-projector latency and redesigning the coded patterns. These improvements may result in real-time capture of dynamic 3-D scenes. As demonstrated, LUMA is a flexible and powerful technique for a variety of 3-D applications and it may help with many others involving multiple cameras.

References

[Avidan and Shashua, 1998] S. Avidan and A. Shashua, “Novel View Synthesis by Cascading Trilinear Tensors,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 4, pp. 293—306, 1998.

[Batlle et al., 1998] 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.

[Boyer and Kak, 1987] 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.

[Chang, 1999] N.L. Chang, *Depth-Based Representations of Three-Dimensional Scenes for View Synthesis*, Ph.D. Thesis, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 1999.

[Chang, 2002] N.L. Chang, “A Multi-Camera and Projected Light System for Fast Efficient 3-D Media Creation,” *Hewlett-Packard Laboratories Technical Report HPL-2002-306*, October 2002.

[Chen and Williams, 1993] S.E. Chen and L. Williams, “View Interpolation for Image Sythesis, *Proceedings of SIGGRAPH*, New York, New York, pp. 279—288, August 1993.

[Devernay et al., 2002] F. Devernay, O. Bantiche, and E. Coste-Manière, “Structured Light on Dynamic Scenes using Standard Stereoscopy Algorithms,” *INRIA Technical Report 4477*, June 2002.

[Faugeras, 1994] O.D. Faugeras, *Three-Dimensional Computer Vision*, MIT Press, Cambridge, MA, 1994.

[Hall-Holt and Rusinkiewicz, 2001] 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.

[Hansard and Buxton, 2000] M.E. Hansard and B.F. Buxton, “Parametric View-Synthesis,” *Proceedings of the European Conference on Computer Vision*, vol. 1, pp. 191—202, 2000.

[Laveau and Faugeras, 1994] S. Laveau and O. Faugeras, “3-D Scene Representation as a Collection of Images and Fundamental Matrices,” *INRIA Tech Report 2205*, February 1994.

[Le Moigne and Waxman, 1988] 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—548, 1988.

[McMillan, 1995] L. McMillan, “Computing Visibility without Depth,” *UNC Chapel Hill Tech Report TR-95-047*, 1995.

[Pollard et al., 1998] S. Pollard, S. Hayes, M. Pilu, and A. Lorusso, “Automatically Synthesising Virtual Viewpoints by Trinocular Image Interpolation,” *Hewlett-Packard Labs Technical Report HPL-98-05*, January 1998.

[Posdamer and Altschuler, 1982] 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.

[Sato and Inokuchi, 1985] 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.

[Seitz and Dyer, 1996] S. Seitz and C. Dyer, “View Morphing,” *Proceedings of SIGGRAPH*, pp. 21—30, August 1996.

[Tsai, 1987] R.Y. Tsai, “A Versatile Camera Calibration Technique for High-Accuracy 3-D Machine Vision Metrology using Off-the-Shelf TV Camera and Lenses,” *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 4, pp.323—344, August 1987.

[Zhang, 2000] Z. Zhang, “A Flexible New Technique for Camera Calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330—1334, November 2000.