Efficient Dense Correspondences using Temporally Encoded Light Patterns

Nelson L. Chang
Imaging Technology Department, Hewlett-Packard Laboratories
1501 Page Mill Road MS 1203, Palo Alto, CA 94304 USA
nelson.chang@hp.com

Abstract

Establishing reliable dense correspondences is crucial for many 3-D and projector-related applications. This paper proposes using temporally encoded patterns to directly obtain the correspondence mapping between a projector and a camera without any searching or calibration. The technique naturally extends to efficiently solve the difficult multiframe correspondence problem across any number of cameras and/or projectors. Furthermore, it automatically determines visibility across all cameras in the system and scales linearly in computation with the number of cameras. Experimental results demonstrate the effectiveness of the proposed technique for a variety of applications.

1. Introduction

Projector-camera systems have become increasingly popular in recent years to address a wide range of applications. These systems typically take advantage of the feedback achieved with the camera capturing the controllable projected display. This feedback mechanism allows the system to automatically establish the geometric relationship between the projector's coordinate system (projector space) and the camera's coordinate system (camera space) in a process of self-calibration. Integral to the calibration process is solving the correspondence problem, i.e. relating the 2-D coordinates in one frame to those in a second one.

A single projector-camera pair can drive many applications including 3-D shape recovery [1], automatic keystoning correction [13], and even interactive control of presentations [14]. Instead of modeling the full 3-D coordinate transformation, many of these applications assume the presence of a reference planar surface such as a projection screen. Projector space is then related to camera space by a simple 3x3 homography. The homography encapsulates the correspondence mapping for points that lie within the imaged planar surface and works well in many cases. However, this simple model does not accurately handle lens distortions with the camera and/or projector. Moreover, it does not generalize for variations in the planar surface and for points in the scene that do not lie on the planar surface.

This paper proposes an approach to directly recover the true correspondence mapping between a projector and camera for an arbitrary 3-D scene. The approach uses temporally encoded light patterns to quickly identify points in common and automatically establish the dense point mapping between projector space and camera space. In addition, the paper demonstrates how this fundamental approach may be used to improve—and even enable—a number of applications including interactive view synthesis. Moreover, the approach serves as the basis for a multi-camera and/or multi-projector framework to quickly solve the multiframe correspondence problem.

2. LUMA for Projector-Camera Pair

The proposed approach, called LUMA, simultaneously encodes the *w* columns and the *h* rows of projector space, thereby providing a unique temporal code for the *w*x*h* 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. The patterns are the bitplanes of the binary representation of the row and column indices, respectively. Figure 1 shows an example of using LUMA to encode rectangle (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 the 4x4 projector space shown in Figure 2. 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

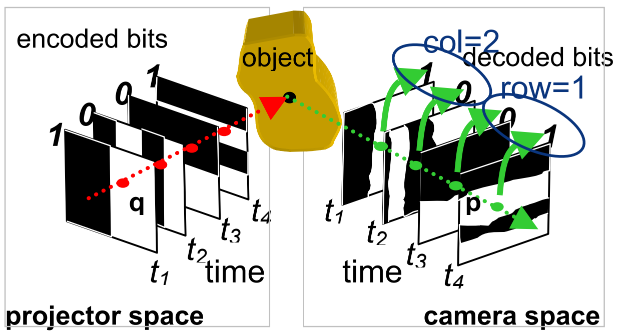

Figure 1. *With LUMA, one simply decodes the captured light patterns to identify corresponding points in projector space.*

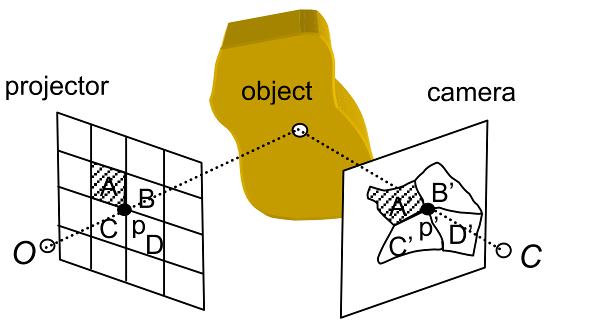

Figure 2. *The connectedness of projector space regions may be used to pinpoint correspondences.*

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 3 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.

With these observations in mind, recovery of the correspondence mapping is straightforward with LUMA [3]. The algorithm to solve dense correspondences between a projector and camera is as follows:

1. Capture the color information of the 3-D scene. This image serves as the color texture map in the final representation.
2. Project and capture reference patterns. An ambient light ("all black") pattern and a full illumination ("all white") pattern are used to determine approximate mean color vectors for the "black" (0) and "white" (1) symbols respectively at every image pixel. Because of illumination variations and different reflectance properties across the scene, the mean color vectors may vary across the image pixels.
3. Determine the validity map. Comparing the captured reference patterns identifies the valid pixels. 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).
4. In succession, project each coded light pattern onto the 3-D scene and capture the result. The k-th pattern of the $\log_2(h)$ horizontally striped patterns that encode the rows of projector space is derived by taking the k-th bit of the row index at every pixel location. The other $\log_2(w)$ patterns to encode the columns are similarly derived. The spatially varying binary patterns are further Gray coded to reduce spatial frequency [11].
5. For every light pattern, decode the symbol at each valid pixel. 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. 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.
6. Perform coarse-to-fine analysis to extract and interpolate imaged corner points at the finest resolution of projector space. 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 4, 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.

LUMA has a number of benefits for a projector-camera pair. It provides an automatic method for estimating dense correspondences between projector space and camera space. These correspondences may be used directly as the mapping for general 3-D scenes or to solve for the optimal homography for scenes with a dominant planar surface. LUMA also identifies the scene points common to both the projector and camera.

It is important to highlight the differences between LUMA and similar structured light scanning (SLS) techniques [9,1,10]. These SLS techniques use binary light patterns to temporally encode only the column indices of projector space. They proceed to compute 3-D

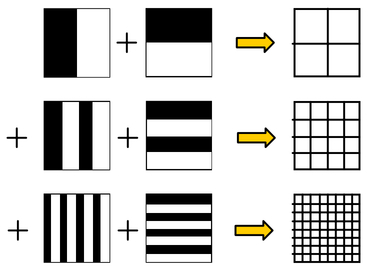
Figure 3. *Projector space rectangles decrease in area with additional finer resolution light patterns.*

information through ray-plane intersections of the identified column of light. With these techniques, no explicit point correspondences are established. Moreover the exact 3-D coordinate transformation between the projector and camera is required. In contrast, LUMA solves directly for correspondences rather than 3-D shape information, thereby enabling many image-based and even multiframe applications. As such, LUMA requires no calibration and places no constraints on the physical locations of the projector and camera. While LUMA employs more patterns than [10] and thus does not handle moving objects, it can be modified to tradeoff capture speed with spatial resolution due to the patterns' hierarchical nature.

In addition, SLS approaches typically use only textureless, uniformly white, and non-specular surfaces. The reason is that certain surfaces (e.g. light-absorbing, specular, or highly reflective) make it more difficult to properly discriminate and decode the projected symbols. LUMA has been designed to minimize the effect of these errors and imposes no explicit restriction on the type of 3-D scenes it can handle.

3. Using Multiple Cameras and/or Projectors

Because of the problem formulation, LUMA may be naturally extended to handle systems with multiple cameras. As described above, LUMA produces the correspondence mapping between a camera and the projector space. 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 a common projector space, resulting in the implicit correspondence mapping between any pair of cameras.

LUMA efficiently solves the difficult multiframe correspondence problem for a static 3-D scene. Through active projection, it overcomes the inherent problems with passive image-based approaches like matching algorithms, stereo, and optic flow [5,2]. Specifically, LUMA does not rely on distinct and consistent textures, nor does it produce spurious results for uniformly colored objects. LUMA also supports any number of cameras

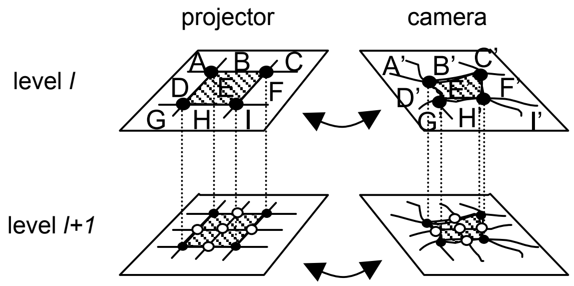

Figure 4. *Analyzing corners at one resolution level helps to locate corners at a finer level.*

and scales linearly in computation with the number, in contrast to many algorithms that require ½*K(K-1)* pairwise matches for *K* cameras. In addition to simplifying the capture process, LUMA significantly reduces computation by defining the possibly multiframe correspondence mappings with respect to a common space.

Furthermore, LUMA automatically computes correspondence, occlusions, and visibility among all cameras without additional computation. Figure 5 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.

The remainder of the paper focuses on the multiple-camera case. It should be clear that by duality LUMA also solves the multiframe correspondence problem for systems with multiple projectors and a single camera. In this case, the correspondences are defined with respect to the common camera space. Likewise, the multi-camera and multi-projector case can be treated by redefining the entire multiframe correspondence mapping with respect to a common camera or projector space.

4. Experimental Results

The following sections demonstrate the applicability and flexibility of LUMA to a number of projector-camera applications including interactive view synthesis, camera

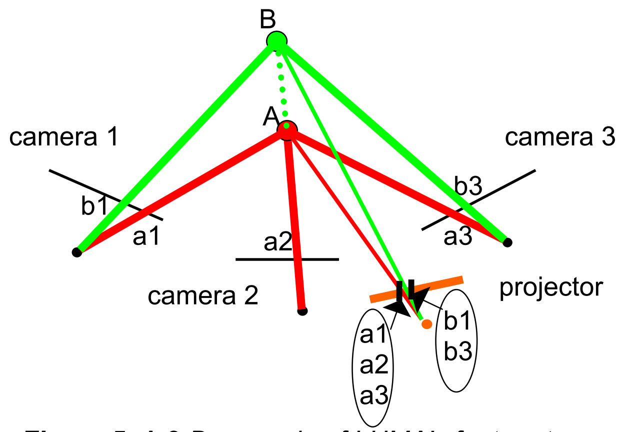

Figure 5. *A 2-D example of LUMA's feature to automatically determine correspondence and visibility across multiple cameras.*

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 6. 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 in ambient light.

In the present implementation, 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 operates at about 5 fps to ensure proper synchronization. The capturing process thus takes less than five seconds, and the analysis process in Section 2 requires less than a minute for three cameras.

The following example gives an idea of the performance of LUMA. A textured planar surface (poster on a cubicle wall) and a two-planar sided foreground object (box) are captured by two cameras, shown as the "Planes" image pair in Figure 7. The images differ by a large average pixel disparity of 150. The first image is warped according to the estimated correspondence vectors to obtain a prediction of the second image. The predicted frame difference is then used to assess the quality of the correspondence results.

LUMA automatically estimates the correspondence mapping between each camera and the projector, thereby implicitly obtaining the correspondence mapping between the two images. The estimated correspondences lead to an accurate predicted second image and absolute frame difference as seen in Figure 8. The brighter regions in the

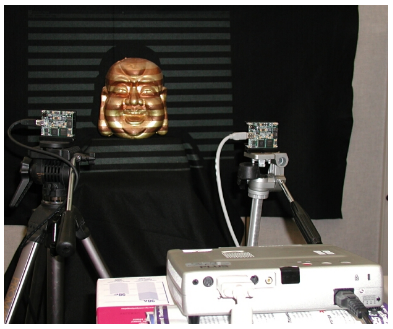

Figure 6. *LUMA uses a projector and one or more cameras.*

latter are due primarily to differences in the cameras' color balances and view dependent lighting effects.

As seen from the results, LUMA automatically finds the points in the scene common to both images, especially important when the projected patterns are not entirely visible. It also performs well even for highly textured scenes. Unlike homography-based systems, LUMA makes no assumption about the scene and hence can automatically identify the correct mapping for scenes with multiple objects. Note that the foreground box is correctly separated from the background poster.

For comparison, a simple bi-directional correlation-based matching algorithm [2] uses the "Planes" image pair directly to produce the results in Figure 9. To help with matching, LUMA is used to first determine the epipolar geometry as described in Section 6. The resulting prediction is visibly much worse, especially in non-textured regions. While basic in nature, this algorithm does highlight common issues found in many image-based matching algorithms, such as the aperture problem, spurious matches due to low-textured or periodic regions, and difficulties with occlusions and discontinuities in the scene. Furthermore, matching algorithms are generally much more computational than an active scheme like LUMA.

5. Application: Interactive View Synthesis

One application of LUMA is to render virtual views of a captured scene to provide the user with an interactive and responsive system, the so-called *interactive view synthesis* problem. 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 parallax-corrected view interpolation 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. Without loss of generality, only three anchor images are considered for simplicity.

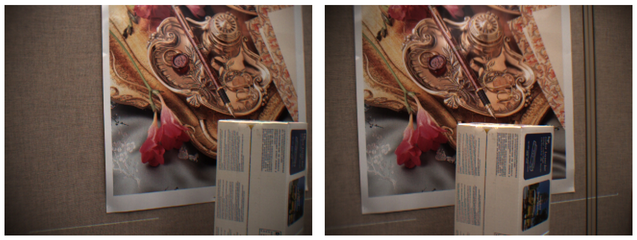

Figure 7. *The "Planes" image pair.*

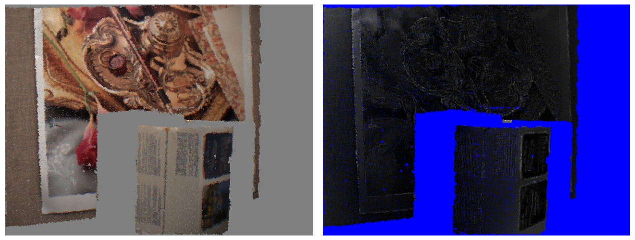

Figure 8. *Prediction and absolute frame difference using correspondences estimated with LUMA.*

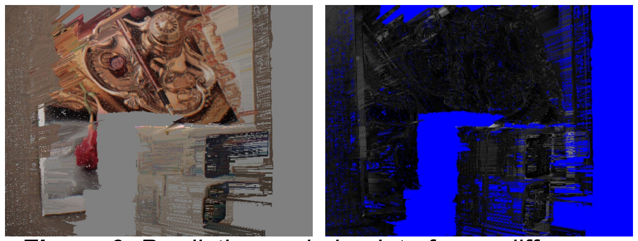

Figure 9. *Prediction and absolute frame difference using correspondences estimated with matching*

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.

[1] The virtual view corresponds to a physically valid in-between view with parallel affine cameras [12], or nearly so, with small rotation [8].

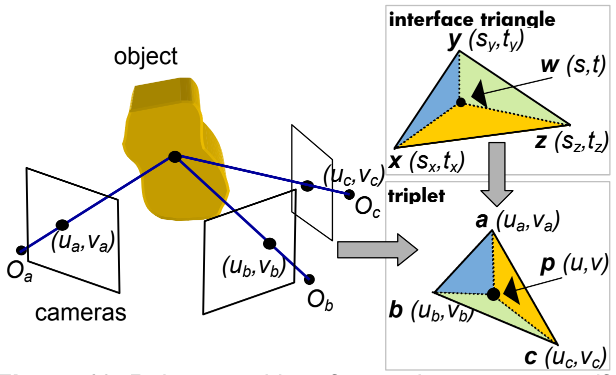

Figure 10. *Point **w** and interface polygon **xyz** specify the new coordinates **p** of a scene point associated with the triplet (**a**,**b**,**c**).*

2. Define user-specified point $\boldsymbol{w}=(s,t)$ with respect to $\Delta\boldsymbol{xyz}$.
3. Determine barycentric coordinates (α,β,γ) corresponding to the weights for vertices $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$:
   a. Compute signed areas of subtriangles formed by vertices and $\boldsymbol{w}$, i.e. SA($\boldsymbol{x},\boldsymbol{y},\boldsymbol{w}$), SA($\boldsymbol{y},\boldsymbol{z},\boldsymbol{w}$), SA($\boldsymbol{z},\boldsymbol{x},\boldsymbol{w}$), where for vertices $\boldsymbol{x}=(s_x,t_x)$, $\boldsymbol{y}=(s_y,t_y)$, $\boldsymbol{z}=(s_z,t_z)$,
   $$SA(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \tfrac{1}{2}((t_y-t_x)s_z+(s_x-s_y)t_z+(s_yt_x-s_xt_y))$$
   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 $\boldsymbol{p}=(u,v)$ 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 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 [3].

The proposed solution to interactive view synthesis is more flexible and less complex than traditional image interpolation approaches [4,7,8]. 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 [6], it is a parameterized view synthesis

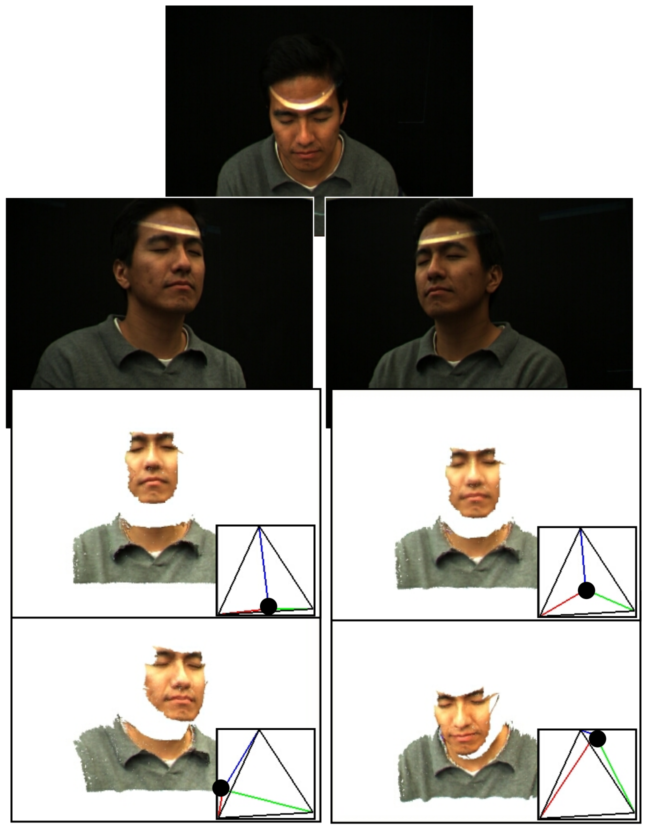

***Figure 11.** "Face" image triplet: view synthesis results.*

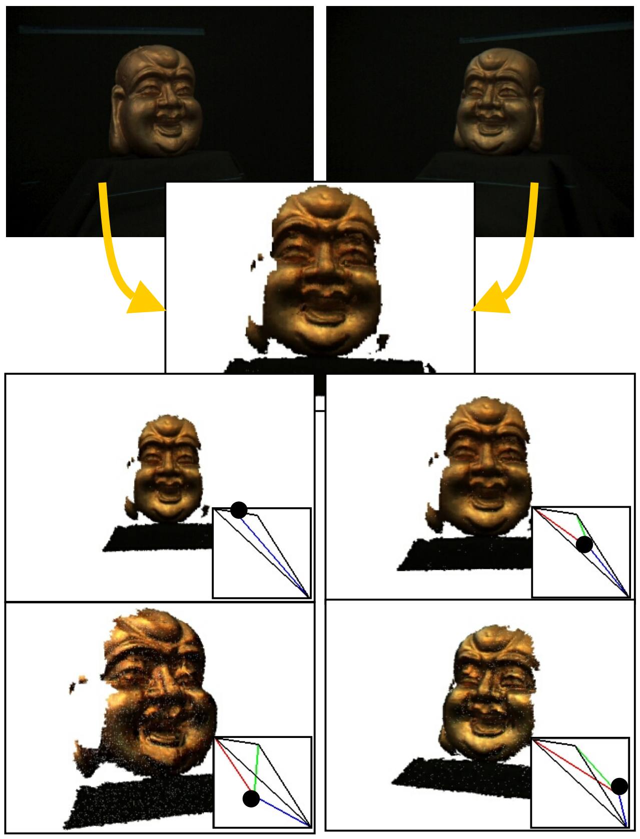

***Figure 12.** "Buddha" image pair: estimated projector's viewpoint and view synthesis results.*

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 anchor 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. The top three images serve as the anchor images to produce the synthesized views shown in the bottom set. The interface triangle, located in the lower right corner of each subimage, shows the user's viewpoint as a dark circle and its relative distance to each camera or vertex. The upper left subimage is a view interpolation between the left and right cameras alone. The next subimage shows the result of interpolating using all three cameras. The bottommost two subimages 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.

The real power of the proposed framework can be demonstrated in the following examples using only two cameras. Figure 12 shows a Buddha mask in the "Buddha" image pair while Figure 13 shows a colorful plant in the "Plant" image pair. Each image pair is used to form the projector's synthesized viewpoint with scaled dimensions shown as the upper center image. The bottom set shows typical synthesized views. With the original two images alone, one can perform only view interpolation like the upper left subimage. However, using the projector's viewpoint as a third anchor image, one can create a broader set of views with more motion parallax as seen in the other three synthesized views.

6. Application: Camera Calibration

The LUMA technology can also be used as a front end for weak or full camera calibration. It is particularly

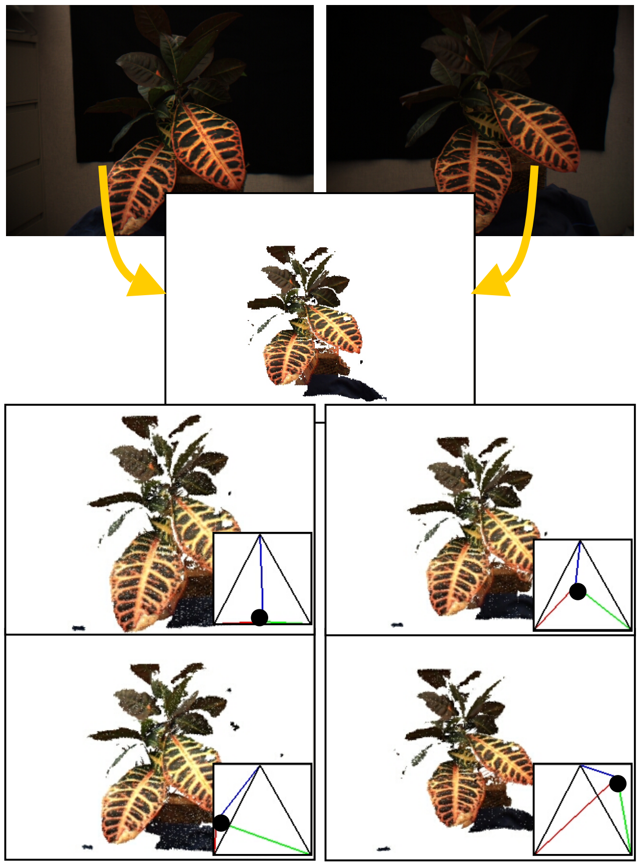
***Figure 13.** "Plant" image pair: estimated projector's viewpoint and view synthesis results.*

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 the images, one can apply LUMA to automatically solve for correspondences among all cameras. 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.

For weak calibration between a pair of cameras, one simply uses the dense correspondences to solve for the fundamental matrix, hence the epipolar geometry, with traditional techniques such as linear least squares or eigenanalysis [15]. LUMA can thus act as a first step for view synthesis techniques like [7,12] that require epipolar geometry.

For full calibration across a number of cameras, a featureless fronto-parallel planar surface (e.g. projection screen, whiteboard, box) serves as an adequate target. Arbitrary world coordinates are assigned to each position in projector space, and the information is fed into

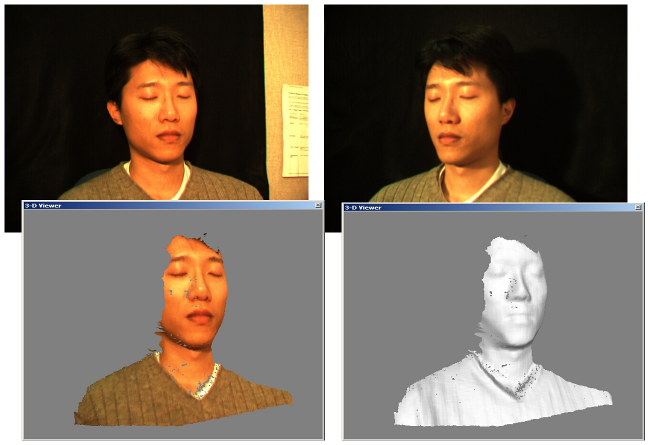
***Figure 14.** "Face2" image pair: reconstructed textured and untextured 3-D models.*

standard nonlinear optimization techniques like [16] to generate the calibration parameters.[2]

7. Application: 3-D Shape Recovery

With a calibrated set up[3], 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 using the ray-plane intersection technique described in Section 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. 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.

[2] 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.

[3] Active calibration from Section 6 is used to obtain dense correspondences for a featureless planar target, and these correspondences are fed into Zhang's calibration algorithm [16].

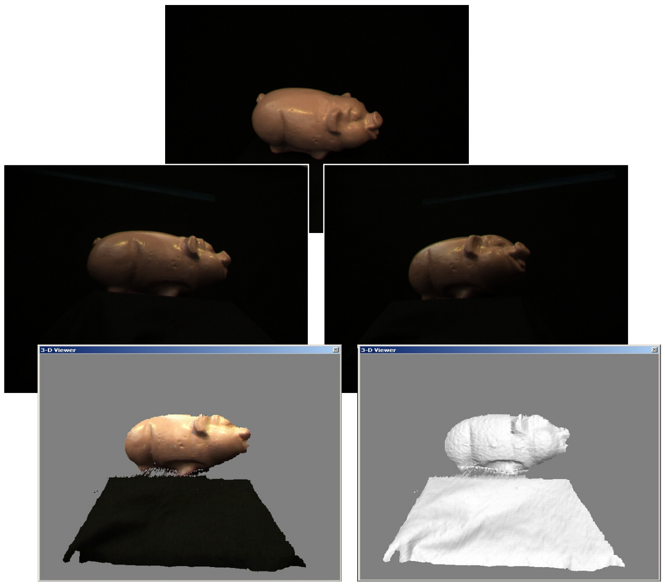

Figure 15. *"Pig" image triplet: reconstructed textured and untextured 3-D models.*

The LUMA system has been used to capture the 3-D shape of various real-world objects. One example is a human face captured with two cameras. As shown in Figure 14, the so-called "Face2" image pair consists of a human subject. The bottom row 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 15, the cameras are organized in triangular formation. The resulting 3-D model with and without texture map can be found in the bottom row. 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.

8. Conclusions

This paper has focused on solving the traditionally difficult problem of establishing dense correspondence across multiple coordinate systems. The proposed solution uses an efficient and robust active illumination technique that replaces computational multiframe optimization by simple decoding. The approach scales well with the number of cameras in the system. As a side benefit, the projector can also serve as an additional virtual camera, thus widening the range of synthesizable views. Future work includes developing more sophisticated coded light patterns for faster correspondence capture as well as exploring stitching-type applications. As demonstrated, LUMA is a flexible technique for a variety of projector-camera and 3-D applications, especially those involving multiple cameras and/or multiple projectors.

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," *Patt. Recog.*, 31(7), pp. 963—982, 1998.

[2] N.L. Chang, *Depth-Based Representations of Three-Dimensional Scenes for View Synthesis*, Ph.D. Thesis, EECS Dept, University of California at Berkeley, 1999.

[3] N.L. Chang, "Interactive 3-D Media with Structured Light Scanning," *Hewlett-Packard Labs Technical Report HPL-2003-112*, May 2003.

[4] S.E. Chen and L. Williams, "View Interpolation for Image Sythesis, *SIGGRAPH'93*, pp. 279—288.

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

[6] M.E. Hansard and B.F. Buxton, "Parametric View-Synthesis," *ECCV'00*, pp. 191—202.

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

[8] 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.

[9] J.L. Posdamer and M.D. Altschuler, "Surface Measurement by Space Encoded Projected Beam Systems," *CGIP*, 18(1), pp. 1—17, January 1982.

[10] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-Time 3-D Model Acquisition," *SIGGRAPH'02*.

[11] K. Sato and S. Inokuchi, "Three-Dimensional Surface Measurement by Space Encoding Range Imaging," *J. Robo. Sys.*, 2(1), pp. 27—39, Spring 1985.

[12] S. Seitz and C. Dyer, "View Morphing," *SIGGRAPH'96*.

[13] R. Sukthankar, R. Stockton, and M. Mullin, "Automatic Keystone Correction for Camera-assisted Presentation Interfaces," *Intl Conf. Multimedia Interfaces*, 2000.

[14] R. Sukthankar, R. Stockton, and M. Mullin, "Smarter Presentations: Exploiting Homography in Camera-Projectors Systems," *ICCV'01*.

[15] G. Xu and Z. Zhang, *Epipolar Geometry in Stereo, Motion, and Object Recognition*, Kluwer, Dordrecht, 1996.

[16] Z. Zhang, "A Flexible New Technique for Camera Calibration," *PAMI*, 22(11), pp. 1330—1334, November 2000.