Stripe Boundary Codes for Real-Time Structured-Light Range Scanning of Moving Objects

Olaf Hall-Holt Szymon Rusinkiewicz
Stanford University†

Abstract

We present a novel approach to real-time structured light range scanning. After an analysis of the underlying assumptions of existing structured light techniques, we derive a new set of illumination patterns based on coding the boundaries between projected stripes. These stripe boundary codes allow range scanning of moving objects, with only modest assumptions about scene continuity and reflectance. We describe an implementation that integrates these new codes with real-time algorithms for tracking stripe boundaries and determining depths. Our system uses a standard video camera and DLP projector, and produces dense range images at 60 Hz with 100 μm accuracy over a 10 cm working volume. As an application, we demonstrate the creation of complete models of rigid objects: the objects are rotated in front of the scanner by hand, and successive range images are automatically aligned.

1 Introduction

The automated computer acquisition of three-dimensional shape is becoming increasingly important commercially, with applications in fields such as computer graphics, virtual and augmented reality, robot navigation, and manufacturing. For many of these applications, it is desirable to obtain the 3D geometry of moving objects in real time. In addition, in many cases 3D scanning is practical only if costs are low, precluding the use of specialized hardware.

Range scanning may be broadly divided into active and passive methods. Active range scanning methods (based on time of flight [3DV Systems], depth from defocus [Nayar 96], or projected-light triangulation) typically perform well in the absence of scene texture, are computationally inexpensive and robust, and return accurate, densely-sampled range data. Of the active range scanning schemes, those based on triangulation may be used with the largest range of scene sizes and have the lowest hardware costs, especially given the increasing capabilities and declining prices of computer-controlled projectors and video cameras. Comparatively little research has been done, however, on applying triangulation methods to scenes containing moving or deforming objects. Previously-studied systems that allow object motion during scanning either use custom-designed hardware [Gruss 92], make strong continuity assumptions about the scene, or are variants of passive vision methods that use a projected texture to aid in solving the correspondence problem.

In this paper, we introduce a triangulation-based range scanning system for moving scenes that returns dense range images in real time (60 Hz). The system uses a new kind of illumination pattern, based on time-coding the boundaries between projected stripes. The boundaries are tracked from frame to frame, permitting the determination of depths even in the presence of moving objects in the scene.

We begin by presenting a classification of structured-light scanning methods according to the reflectance, spatial, and temporal coherence assumptions they make. We show how previous approaches fit in this taxonomy, and present a new set of assumptions that leads to a system optimized for moving scenes. Next, we derive the stripe boundary codes used by our system and describe the boundary tracking and decoding algorithms used by our prototype implementation. We present results from this system, and show how the scanner may be combined with automatic alignment to allow the rapid creation of complete models of rigid objects. Finally, we discuss limitations of our current system and propose some future enhancements to make the system less sensitive to the presence of scene texture and depth discontinuities.

2 Structured-Light Code Taxonomy

The assumptions made in designing a structured light system are often stated only implicitly, despite their importance. We characterize the space of structured light methods in terms of the underlying assumptions they make about the reflectance, space, and time coherence of the scene. We then argue that there exists a relatively unexplored, practical combination of assumptions that leads to a new class of structured light scanning methods.

†Stanford Computer Graphics Lab
Gates Building 3B
Stanford University
Stanford, CA 94305
{olaf,smr}@graphics.stanford.edu

0-7695-1143-0/01 $10.00 © 2001 IEEE

2.1 Background

We shall consider the space of scanners incorporating a single illumination source and a single detector (typically a CCD camera), where depth is obtained by triangulation [Posdamer 82, Besl 88]. The differences among these systems lie in the methods they use to form correspondences between camera pixels and locations in the projected pattern. Without loss of generality, we refer to these pattern locations as "projector pixels." To obtain correct depths, each method must make certain assumptions about how the scene transforms the projected light into the camera image. In abstract terms, we may think of the scene as a function that transforms projector pixels into camera pixels, and we may consider the restrictions each scanning method places on this transfer function.

One class of scanners uses color coding to determine the correspondence between camera and projector pixels. A pattern consisting, for example, of colored dots [Davies 96] is projected onto the scene, then the color observed at a location is used to determine the camera-projector correspondence. This type of method assumes that the scene does not modify the colors from the projector to the camera, thus restricting the allowable transfer functions to those that do not modify colors. Effectively, this makes a *reflectivity* assumption about the scene.

Another class of scanners encodes information into a pattern in some neighborhood of projector pixels. For example, Boyer and Kak describe a system in which location in the projector image is identified by a color coding of adjacent stripes [Boyer 87]. The correspondence between camera pixels and projector location can therefore be made only if the scene "transfer function" preserves spatial neighborhoods. We will call this a *spatial coherence* assumption. Note that for this system, violation of the coherence assumptions may cause small areas of incorrect depths around the discontinuities. Thus, the system assumes *local* scene coherence. In contrast, there exist systems for which the spatial coherence assumption must be valid *globally*. For example, Proesmans et. al. describe a system in which a grid pattern is projected onto the scene, and the grid lines visible in the camera image are followed to determine correspondences between points in the camera and projector images [Proesmans 96]. In this case, a single visible surface is assumed, since the system relies on counting grid lines.

A third way of encoding information into the projector signal is to group pixels over time rather than space. For example, in the system of Sato and Inokuchi [Sato 87], the projector pixels are turned on and off over time, so that when a camera pixel records a particular on and off intensity pattern, the corresponding projector pixel can be identified. A popular choice for this on-off pattern is a set of Gray codes [Bitner 76], though other approaches using black-and-white [Gartner 96], gray-level [Horn 99], or color [Caspi 96] stripes, or swept laser stripes or dots [Rioux 94], have been examined. The common underlying assumption is *temporal coherence* of the scene, namely that neighborhoods of pixels can be identified over time. Note that all of the methods mentioned here make a global temporal coherence assumption (i.e., that the scene is static); we will discuss what it means to make a local temporal coherence assumption in the following subsection.

The design space for projected light illumination patterns can therefore be described in terms of reflectance, spatial coherence, and temporal coherence assumptions. The strength of a spatial coherence assumption can be measured by the number of pixels involved: if the patterns to be identified in a given camera image require a minimum of n pixels, then the smallest identifiable features in the scene must occupy at least n camera pixels. The reflectance assumption impacts the range of colors permitted in the scene, as well as the frequency of textures.

Our goal in designing an illumination pattern is to enable the scanning of moving scenes, with small feature sizes and largely unknown reflectance properties. Stated in terms of the above assumptions, this means that we would like to simultaneously minimize the spatial coherence, reflectance, and temporal coherence assumptions required by our system. This goal leads to the adoption of the following assumptions:

- Most of the time, two horizontally adjacent camera pixels will see the same surface.
- Most of the time, the reflectance of two horizontally adjacent pixels is similar.
- Most of the time, projected features that persist for n frames can be used in making correlations. In our implementation, we use $n = 4$.

The last of these assumptions has not to our knowledge been used in the same way before, so we will now look at the implications of motion for structured light systems, in particular explaining this last assumption more fully.

2.2 Space-time Coherence

In order to formalize the notion of temporal coherence, as well as to define a metric of the strength of such assumptions (as we did for spatial coherence), we consider the appearance of a scene illuminated by a sequence of patterns in (four-dimensional) space-time. We begin by examining the simplified case of two-dimensional range scanning (i.e., working in a plane), with time as a third dimension.

Let us consider a scene with objects moving in the plane. By stacking up a series of snapshots of the scene at different times, we see that the objects trace out volumes in three-dimensional space-time (Figure 1). A time-varying light pattern projected onto the objects can then be visualized as a two-dimensional light pattern projected onto these volumes, and a series of (one-dimensional) camera images

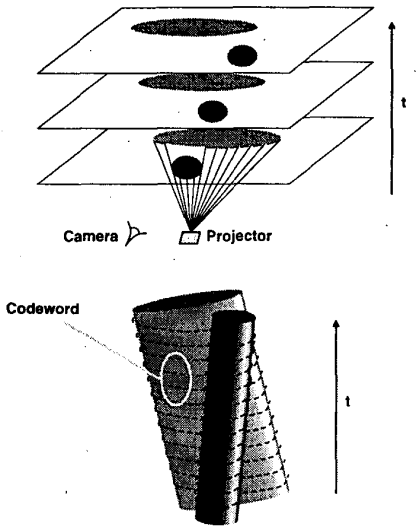

Figure 1: A 2D (planar) scanning application examined in 3D space-time. At top, we show a scene being scanned at three points in time. At bottom, we show how the trajectory of each object may be thought of as defining a solid in space-time. A sequence of 1D projected patterns becomes a 2D pattern projected onto the objects in space-time, and local neighborhoods (defining codewords) have both spatial and temporal extent. The relation between 3D scanning and 4D space-time is analogous.

of the illuminated objects may be thought of as a 2D picture of the volumes. Thus, there is a direct correspondence between moving-object, multiple-pattern range scanning in the plane and single-frame correspondence methods (sometimes called one-shot methods) in 3D.

Given this analogy, we may apply some concepts of 3D one-shot methods to the case of 2D moving-object range scanning. For example, 3D one-shot methods often use light patterns in which two-dimensional neighborhoods specify codes. In the 2D moving-object case, such neighborhoods have both spatial and temporal extent, but the idea of communicating a code via these two-dimensional neighborhoods can still be used. Similarly, an analogy may be formed between following a continuous grid line (in 3D grid-based methods) and tracking a feature across several frames (in the 2D case). The same analogies may be made between moving objects in 3D and a single pattern in 4D.

Thus, we see that the notions of temporal and spatial coherence may be combined into a single concept of coherence in space-time. We may then classify particular spatio-temporal coherence assumptions according to the extent of space-time neighborhoods over which the moving scene is assumed to be continuous, and may therefore convey codes. Thus, our assumption of temporal coherence over 4 frames means that we are assuming neighborhoods of eight pixels to form correspondences in space-time: two in each frame. This implies the ability to *track* the movement of this pair of pixels from one frame to the next.

2.3 Derivation of Stripe Boundary Codes

The assumptions given above limit the set of scenes that our system will be able to scan. Making these assumptions explicit, in our case, also provides a set of tools for designing an illumination pattern. We proceed with a sort of worst-case analysis: if the scene behaves arbitrarily badly, except where constrained by the above assumptions, what approach to forming correspondences will still work?

- Since our spatial coherence assumption involves only horizontally adjacent pixels, pixels on different scan lines of the camera may not have anything in common, and must be treated independently. Due to the epipolar constraint, the independence of scan lines of the camera implies independence of rows of pixels on the projector, and therefore we can design the projector illumination pattern independently for different horizontal rows. In particular, we use the same illumination pattern for each row, which forms a stripe pattern.
- In order to allow the greatest possible variation in scene reflectances, our system uses only black and white stripes.
- Since at most two pixels of a given frame can be used together to infer correspondences, the only projected feature that we can reliably identify is the boundary between two stripes. Thus, we will use a *stripe boundary code* to convey information between projector and camera.

Focusing on stripe boundaries has several advantages. For example, if the stripes on either side of a stripe boundary can each be assigned n different codes over time, then roughly n^2 distinct stripe boundaries can be identified in a camera image, even if no scene feature is as wide as a stripe. Also, under appropriate assumptions on the smoothness of the scene geometry and texture, the stripe boundary may be located with sub-pixel accuracy, thus increasing the ultimate accuracy of the scanning system.

At this point the design of an illumination pattern has been considerably constrained. The next section describes a method for generating stripe boundary codes that can incorporate a variety of additional design criteria.

3 Designing a Stripe Boundary Code

Designing a stripe boundary code involves assigning a color (black or white) to each stripe at each point in time, such that each stripe boundary has a unique code (consisting of the black/white illumination history on both sides of the boundary) over the sequence of four frames. To maximize spatial resolution, we would like the pattern to contain as many boundaries as possible. In order to generate such patterns, we form an analogy between stripe boundary codes and paths in a graph, and search for a maximal-length path that satisfies certain conditions.

Consider a graph in which each node represents the black/white illumination of a single stripe over four frames, and all pairs of nodes are connected with edges. Each edge represents a (time-varying) stripe boundary, since it lies between two (time-varying) stripes. This graph has $2^4 = 16$ nodes and 120 (undirected) edges. A path through this graph that traverses each edge at most once in each direction corresponds to a stripe boundary code, since it specifies the assignment of colors to stripes over time, with the property that the code of each boundary is unique.

We may place additional restrictions on the stripe boundary codes by modifying the graph and adding requirements on how it is traversed. First, we avoid any stripe boundaries that do not vary over time, since they will be indistinguishable (from the point of view of the camera) from texture boundaries. This restriction is manifested in the graph by deleting the edge between the two time-invariant color progressions, namely the edge between all black (0000) and all white (1111).

A second, more sweeping restriction arises as follows. In the complete graph, we allowed "boundaries" between any two stripes, including two stripes of the same color. Such a boundary cannot be seen in a camera image, so we call it a "ghost" boundary. In our case, if we eliminated all ghost boundaries, we would eliminate all but eight edges in the graph, which would lead to very short solution paths.

In order to allow some ghost boundaries, there must be a way to identify them implicitly in camera images. We enable such a determination with the help of the following restriction: we number the stripe boundaries according to their position along a projector row, from 1 to m, and restrict the locations of ghost boundaries to odd-numbered positions in frames 1 and 3 and even-numbered positions in frames 2 and 4. Thus, a given stripe boundary will be visible at least every other frame, and in any frame there will be at most one ghost between any two visible boundaries.

To embed this restriction in the graph, we distinguish edges according to the ghost boundaries they contain: an edge is dotted if there are no ghost boundaries in frames 1 and 3, and solid if there are no ghost boundaries in frames 2 and 4. Edges that have no ghost boundaries in any frame are dashed, and any remaining edges are deleted from the graph, since they will not be used in any path. This restriction reduces the number of (undirected) edges to 55, resulting in the graph of Figure 2. We then constrain our search to paths in which the types of traversed links alternate between solid and dotted (where dashed is understood to stand for either).

Since the problem as stated has many solutions, we may impose additional conditions:

- We would like the effect of errors (i.e., misidentifying a stripe boundary) to be as large as possible, so that outliers are easier to identify and filter away. (This is the opposite of the strategy adopted by [Jiang 94], in which the aim is to minimize the size of errors.)
- We look for codes in which the distribution of stripes of widths 1 and 2 (or, equivalently, the distribution of ghosts) is as uniform as possible within each frame.

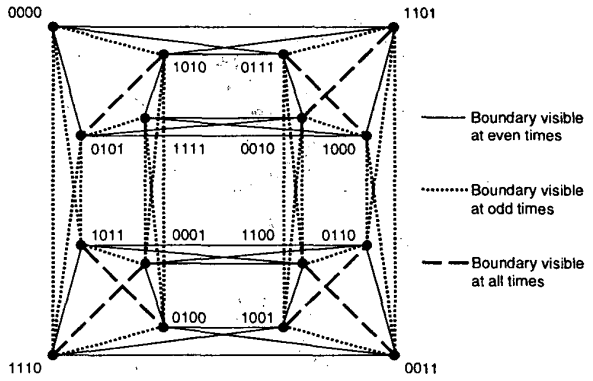

Figure 2: A graph used to determine a set of stripe boundary codes. We look for a traversal of edges in this graph (with the understanding that each edge may be traversed once in each direction), with the added constraint that solid and dotted edges must alternate along the path. A dashed edge may be substituted for either solid or dotted. As mentioned in the text, the edge between 0000 and 1111 is missing: we disallow this stripe boundary as a valid code, since it is too easy to confuse with static texture.

A maximal-length code will traverse each of the 55 edges of the graph twice (once in each direction), so will have $2 \cdot 55 = 110$ stripe boundaries (and 111 stripes). Since we only need to find the code once, we may use a brute-force algorithm to search for paths in the graph. An example of such a code is shown in Figure 3.

4 Implementation

We have implemented a prototype range scanning system based on the stripe boundary codes described above. In our system, a DLP projector cycles through the illumination patterns at 60 Hz., and a standard NTSC video camera is used to capture images. The video is digitized and processed in real time, and the system generates a new range image every $^1/_{60}$ sec.

Our algorithm for depth extraction consists of segmenting each video field into illuminated and unilluminated regions (i.e., "black" and "white"), finding stripe boundaries, matching these boundaries to those on the previous field, and using information about the illumination history of each boundary to determine the plane in space to which it corresponds. Depth is then obtained via ray-plane triangulation.

We now discuss the tradeoffs made in implementing each stage of this pipeline on current hardware, as well as possible extensions to these algorithms to make them more robust as hardware capabilities increase.

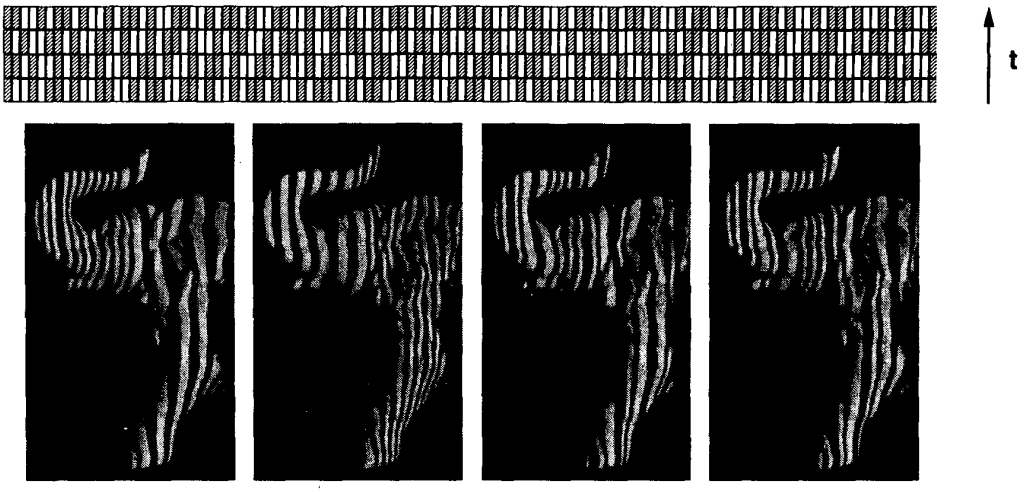

Figure 3: At top, a four-frame sequence of projected patterns satisfying the conditions in Section 3. Each pattern has $2 \cdot 55 = 110$ stripe boundaries (111 stripes), with the property that each stripe boundary has a unique code (consisting of the black/white illumination history on either side of the boundary over the sequence of four frames). At bottom, a sequence of video frames of an elephant figurine illuminated with a stripe boundary code.

4.1 Segmentation Algorithm

Given the assumption of local coherence, we may assume that whenever we see a strong gradient in image intensity it is due to a projected stripe boundary. Therefore, we have implemented a segmentation algorithm that finds large gradients in intensity along camera image scanlines, and assumes that these are the locations of stripe boundaries. For scenes without high-frequency textures, we have found this method to be effective and robust, while still running in real time. In particular, we have found this method less sensitive to variations in reflectivity and changes in ambient illumination than simple threshold-based segmentation.

4.2 Stripe Matching Algorithm

Since our approach relies on time-coding the boundaries between stripes, a critical part of our algorithm is matching the boundaries visible in each frame to those in previous frames. This is a nontrivial problem for two reasons. First, the boundaries move from frame to frame, potentially with large velocities. Second, the fact that our code contains "ghost" boundaries means that not all boundaries are visible in each frame.

It is the presence of ghosts (i.e., the inferred black-black and white-white stripe "boundaries") that distinguishes our stripe matching problem from the traditional feature tracking literature. To make the problem tractable, we must use the constraints presented in Section 3, namely that there may be at most one ghost between each pair of visible stripe boundaries, and that ghost must match to a visible stripe boundary in the previous and following frames. These conditions limit the possible matches and allow us to determine, in many cases, whether certain boundaries should match to

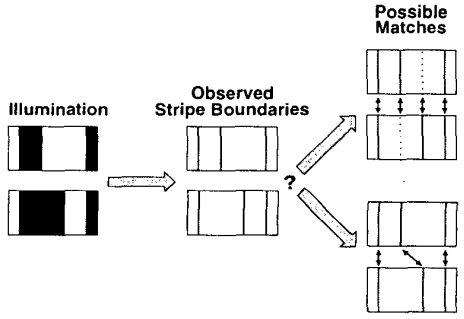

Figure 4: Matching stripe boundaries becomes difficult in the presence of "ghosts" (i.e., boundaries that could not be observed). Here, we show one possible ambiguity, namely whether the center stripe boundaries in the two frames should match to each other or should match to "ghosts" in the other frame.

other visible boundaries or to ghosts. Even these conditions, however, are not enough to disambiguate the situation shown in Figure 4. The two possibilities of having the center stripes match to each other and having them match to ghosts in the other frame are both allowed by the constraints mentioned above.

Although there is a large literature on tracking algorithms that could potentially be adapted to our application, including multiple-hypothesis methods [Reid 79] and methods that use velocities [Brown 97], most of these approaches are too slow for real-time implementation. Therefore, we currently implement only a simple matching algorithm that hypothesizes all possible locations of ghosts and matches each visible boundary to the closest stripe or hypothesized ghost in the previous frame. As discussed later, this places a constraint on the maximum allowable velocity of stripes, hence limiting the speed at which objects in

the scene can move. We anticipate that future systems will incorporate better matching heuristics, permitting correct stripe matching in the presence of greater frame-to-frame motion.

4.3 Decoding Algorithm

Once we have matched the stripe boundaries in one frame to those in the previous frame, we propagate the illumination history (i.e., the color of the stripes on either side of the boundary over the past four frames) of the old boundaries to the new ones. If we have seen and successfully tracked this boundary for at least four frames, this history identifies it uniquely. The boundary remains identified at every frame thereafter, since the four-frame illumination history contains all four patterns.

Given a stripe boundary identification, we determine the plane in space to which the boundary corresponds. We then find the intersection of that plane with the ray corresponding to the camera position at which the boundary was observed; this determines the 3D location of a point on the object being scanned. An important difference between our approach and traditional projected-stripe systems based on Gray codes is that this scheme only gives us depth values at stripe boundaries. These depths, however, are very accurate: we triangulate with an exact plane (the stripe boundary), rather than a wedge formed by two planes (the stripe itself). For smooth surfaces without high-frequency texture, we may perform sub-pixel estimation of the location of the stripe boundaries to further reduce depth errors (Figure 5).

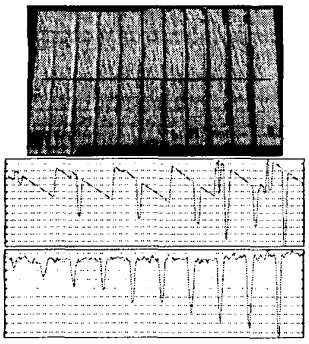

Figure 5: Comparison of decoding accuracy with and without sub-pixel estimation of stripe boundaries. At top, a rendering of a scanned target with grooves ranging from 0.1 mm to 1.0 mm in depth. At center, the depth profile of a slice through the target, if sub-pixel estimation of stripe boundaries is not used. At bottom, the corresponding depth profile when sub-pixel estimation is used.

5 Results

We have tested our implementation on a scene consisting of an elephant figurine, approximately 10 cm in size, rotated by hand at approximately 1 cm/sec. In Figure 6, we compare our stripe boundary code system to a conventional Gray code-based system for this scene. Note that because the elephant was moving, the Gray code method gave erroneous results in regions in which stripes moved from frame to frame. In contrast, our system produced correct results in the moving-object case. Some statistics about our implementation are shown in Table 1, and statistics about the test scene are shown in Table 2.

Table 1: Statistics about our range scanner implementation.

Frame size	200 x 200 (potentially 640 x 240)
Rate	60 Hz
Working Volume	6 cm x 12 cm x 16 cm
Sample Spacing	1.25 – 2.5 mm (X) x 0.75 mm (Y)
Accuracy	< 100 μm (Z)
Computer System	SGI Onyx2, 250 MHz MIPS R10000

Table 2: Statistics about our "elephant" test scene.

Total frames	16850
Total scanning time	4.7 min.
Avg. samples per frame	3680
Total range samples	62 M
Frames used for reconstruction	644
Samples used for reconstruction	1.3 M

5.1 Automatic Frame-to-frame Alignment

Although our implementation can be used for scanning non-rigid (deforming) moving objects, if the objects are rigid we may extend the system to automatically produce complete 3D models instead of just range images. The key is that because we gather data continuously, successive range images of a rigid object moving at modest speeds are close to each other. We can thus align consecutive range images, thereby recovering the relative motion of the object and range scanner. The algorithm we use to do this is iterative closest points (ICP): for each sample in a range image, we find the closest sample in the other range image and apply a rigid-body transformation that minimizes the distances between corresponding points [Chen 91, Besl 92]. This algorithm is iterated until convergence.

The classic problem with ICP algorithms has been the presence of a large number of local minima, leading to incorrect solutions unless a good initial estimate of the relative alignment is available. Because our range images are taken so close together, however, we know that the relative transformation between any two consecutive range images will be small. Thus, frame-to-frame ICP can be performed completely automatically, with little danger of converging to incorrect local minima unless the object being scanned moves completely out of the field of view or contains pathological geometric degeneracies.

Figure 7 shows 644 range images of the elephant, obtained by moving the elephant in front of the scanner by hand. Because our system acquires data at a high rate, this represents only 4% of the frames gathered over approximately five minutes of scanning. The scans were automatically aligned using frame-to-frame ICP, then additional cor-

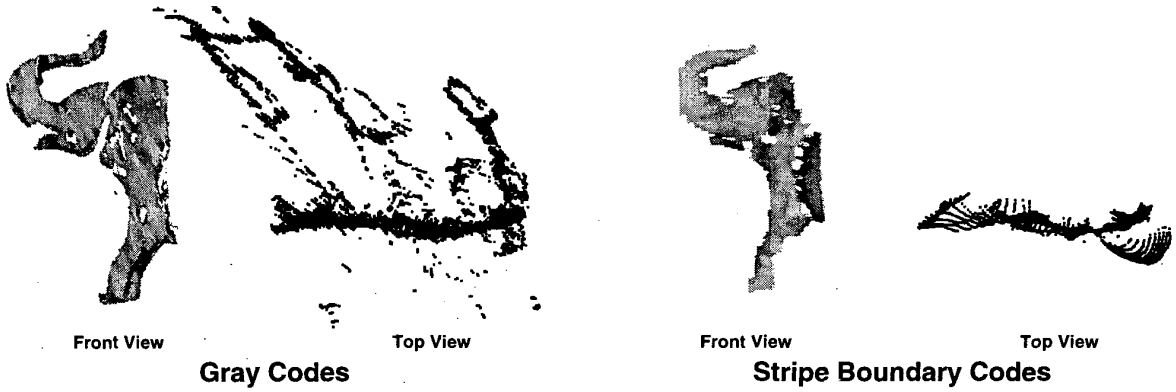

Figure 6: Comparison of range data acquired using Gray codes and stripe boundary codes on a moving scene consisting of an elephant figurine moving over a black background (some raw video frames are shown in Figure 3). At left, a scene moving at approximately one pixel per frame, scanned using Gray codes. Note the presence of incorrect geometry, especially near discontinuities. At right, the same scene scanned using our codes. Note that despite the presence of motion we obtain correct geometry.

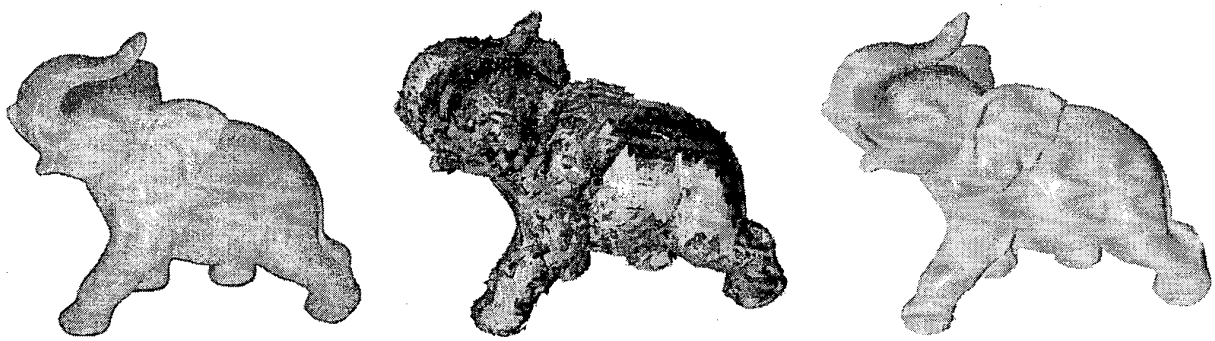

Figure 7: At left, a photograph of an elephant figurine. At center, 644 range images of the elephant obtained using our system by manually moving the elephant in front of the scanner. The individual scans are color coded, and have been automatically aligned using an ICP algorithm. At right, a unified model of the elephant, obtained by combining the scans with the VRIP algorithm.

respondences were computed and an optimal global registration among all the scans was found using a relaxation technique [Pulli 99]. Finally, the scans were merged into a single mesh using the VRIP algorithm [Curless 96]. The alignment and merging are currently performed offline; for this example, they took approximately 30 minutes.

5.2 Limitations

Although we have demonstrated a system capable of real-time range scanning, our implementation has several limitations on its applicability:

Texture: As mentioned in Section 4.1, we currently assume that scene texture varies slowly in order to segment illuminated and unilluminated regions. If the scene does contain step-edges in texture, our segmentation algorithm may report false positives (as well as some false negatives). Even though we have designed our code such that static stripe boundaries correspond to illegal codewords, our system may obtain incorrect geometry in the presence of moving texture.

Structured-light methods for static scenes often compensate for texture by adding an additional all-white frame. This allows them to determine the reflectance seen at each pixel, and use this reflectance estimate during segmentation of the remaining frames. Such an approach could also be implemented in the moving scene case, though it would require frame-to-frame tracking of the reflectance at each point. In addition, the presence of extra all-white frames would reduce the rate at which depths are returned. An alternative solution might be based on simultaneous acquisition at multiple wavelengths; this would not reduce the rate of capture.

Silhouettes: Another aspect of our current system that could be made more robust is the handling of silhouette edges and disocclusion of geometry from behind silhouettes. Currently, we must wait at least four frames before we have enough data to identify a new stripe, since the code is four frames long. Shortening this delay would be possible by assuming greater temporal coherence. For example, we could attempt to identify a new stripe as soon as it appears by *looking ahead* three frames. Such a look-ahead scheme

could also be used improve robustness, by providing additional hints of whether a given stripe was misidentified (i.e., it is likely that there was an error in tracking if the observed code at a stripe is not the same as the codes four frames ago and four frames into the future). Introducing such an algorithm based on look-ahead, however, would not only introduce additional latency into the system, which might be undesirable in certain applications, but also reduce the ability to identify short-lived features (specifically, those that appear then disappear again between four and eight frames later).

Object Motion: Because our prototype implementation was designed to run in real time on present hardware, we were limited to simple algorithms for stripe boundary matching and decoding. In practice, this requires objects in the scene to move relatively slowly (between one-fourth and one-half stripe-width per frame) in order for boundaries to be matched correctly. For our prototype, this corresponds to a constraint that objects move a maximum of approximately 10% of our working volume per second. We anticipate that future systems could incorporate more sophisticated tracking algorithms to allow for greater object speeds.

Intrusiveness: For many potential applications of a range scanning system (in particular, any applications involving people), it is distracting to have visible flashing lights. Such applications would require making the illumination patterns imperceptible, either by projecting them in the infrared or by using a time-multiplexed light cancellation technique [Raskar 98].

6 Conclusion

We have presented an analysis of structured-light scanning in terms of reflectance, spatial, and temporal coherence assumptions. Based on this examination, we have derived a new set of illumination codes optimized for moving scenes, and implemented a prototype system capable of obtaining range images at video rates. We have shown that this system may be used to obtain complete models of rigid objects rapidly and automatically, without the need for calibrated object motion.

In addition to addressing the limitations mentioned in Section 5.2, future work on this system might focus on extensions to high-speed cameras, multiple cameras and projectors, and automated methods for calibrating the scanner. For the case of rigid objects, we are currently working on methods to perform alignment and merging in real time. This would have the advantage of giving the user instant feedback about the location of unscanned areas in the model, allowing for easier determination of when the object has been scanned completely.

Because our system uses off-the-shelf components and is computationally inexpensive, it permits a variety of potential applications in such fields as tele-immersion or robot guidance. In addition, since moving the scanner is equivalent to moving the scene, making the scanner portable would permit real-time digitization of buildings, rooms, or movie sets.

References

[**3DV Systems**] 3DV Systems, Inc. "ZCam," Web page: http://www.3dvsystems.com/

[**Boyer 87**] Boyer, K. L. and Kak, A. C. "Color-Encoded Structured Light for Rapid Active Ranging," *Trans. PAMI*, Vol. 9, No. 1, 1987.

[**Besl 88**] Besl, P. "Active Optical Range Imaging Sensors," *Machine Vision and Applications*, Vol. 1, 1988.

[**Besl 92**] Besl, P. and McKay, N. "A Method for Registration of 3-D Shapes," *Trans. PAMI*, Vol. 14, No. 2, Feb. 1992.

[**Bitner 76**] Bitner, J. R., Erlich, G, and Reingold, E. M. "Efficient Generation of the Binary Reflected Gray Code and its Applications," *CACM*, Vol. 19, No. 9, 1976.

[**Brown 97**] Brown, R. G. and Hwang, P. Y. C. *Introduction to Random Signals and Applied Kalman Filtering*, 3 ed., John Wiley & Sons, 1997.

[**Caspi 96**] Caspi, D. and Kiryari, N. "Range Imaging with Adaptive Color Structured Light," *Trans. PAMI*, Vol. 20, No. 5, 1996.

[**Chen 91**] Chen, Y. and Medioni, G. "Object Modeling by Registration of Multiple Range Images," *Proc. IEEE Conf. on Robotics and Automation*, 1991.

[**Curless 96**] Curless, B. and Levoy, M. "A Volumetric Method for Building Complex Models from Range Images," *Proc. SIGGRAPH*, 1996.

[**Davies 96**] Davies, C. and Nixon, M. "Sensing Surface Discontinuities via Coloured Spots," *Proc. IWISP*, 1996.

[**Gartner 96**] Gartner, H., Lehle, P., and Tiziani, H. "New, Highly Efficient, Binary Codes for Structured Light Methods," *SPIE*, Vol. 2599, 1996.

[**Gruss 92**] Gruss, A., Tada, S., and Kanade, T. "A VLSI Smart Sensor for Fast Range Imaging," *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, 1992.

[**Horn 99**] Horn, E. and Hiryati, N. "Toward Optimal Structured Light Patterns," *Image and Vision Computing*, Vol. 17, 1999.

[**Jiang 94**] Jiang, X. and Bunke, H. "Range Data Acquisition by Coded Structured Light: Error Characteristic of Binary and Gray Projection Code," *SPIE*, Vol. 2252, 1994.

[**Nayar 96**] Nayar, S. K., Watanabe, M., and Noguchi, M. "Real-Time Focus Range Sensor," *Trans. PAMI*, Vol. 18, No. 12, 1996.

[**Posdamer 82**] Posdamer, J. L. and Altschuler, M. D. "Surface Measurement by Space-encoded Projected Beam Systems," *Computer Graphics and Image Processing*, Vol. 18, 1982.

[**Proesmans 96**] Proesmans, M., Van Gool, L., and Oosterlinck, A. "One-Shot Active 3D Shape Acquisition," *Proc. ICPR*, 1996.

[**Pulli 99**] Pulli, K. "Multiview Registration for Large Data Sets," *Proc. 3DIM*, 1999.

[**Raskar 98**] Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L., and Fuchs, H. "The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays," *Proc. SIGGRAPH*, 1998.

[**Reid 79**] Reid, D. "An Algorithm for Tracking Multiple Targets," *Trans. Auto. Control*, Vol. 24, No. 6, 1979.

[**Rioux 94**] Rioux, M. "Digital 3-D Imaging: Theory and Applications," *SPIE*, Vol. 2350, 1994.

[**Sato 87**] Sato, K. and Inokuchi, S. "Range-Imaging System Utilizing Nematic Liquid Crystal Mask," *Proc. ICCV*, 1987.