Interactive Visualization of a 15 Million-Point Scene, Captured With a Structured Light Point Scanner

by

Jonathan C. A. Schmid

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Thesis Committee

Camillo J. Taylor, University of Pennsylvania (Advisor)
Konstantinos Daniilidis, University of Pennsylvania
Norman I. Badler, University of Pennsylvania

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Master of Science in Engineering

2010

Camillo J. Taylor,
Supervisor of Thesis

Konstantinos Daniilidis
Graduate Group Chairperson

Acknowledgments

Special thanks to Professor C. J. Taylor for tireless instruction and guidance in Computer Vision math. And also for contributing such cool figurines for the scan. Stay tuned for more scans...

Thanks also to my committee members: Professor Norman Badler and Professor Kostas Daniilidis for agreeing to participate and help oversee the defense presentation.

Sincerely,

Jon Schmid

Contents

Abstract

Acknowledgments

Introduction

Scene Capture of 3D Point Data

- Structured Light
- Per-Pixel Gray Coding
- Triangulation: Range Images from Disparity

Visualization and Point Rendering

- Point Based Rendering (PBR)
- Real-World Point Size
- Speed Optimizations

Towards Practical Scene Capture

- Multiple Views: (Filling Data Gaps)
- Reflectance Capture: Simulated Lighting and Shading
- Fitting More in VRAM
  - Computed SIMD Shading Functions
  - Range Image Compression

List of Figures

1) Macro photography and other photographic disciplines present visual forms outside everyday experience.

2)

2.1) View into our captured virtual scene from along the center rig-camera optical axis.

2.2) View into our captured virtual scene with camera position translated to the right of center axis and view angle rotated towards scene center.

2.3) View into our virtual scene with camera position translated up and to the right of the center axis.

3) Digital Michelangelo Project: Scanning Michelangelo's 5m *David* sculpture.

4) Conceptual sketch of light triangulation between projector and camera.

5) Examples of Gray code targets for projector calibration and pixel tracking.

6)

6.1) Point Rendering of object demonstrating static point size of 1 pixel.

6.2) Point Rendering of object demonstrating static point size of 5 pixels.

7) Plot of Frames per Second (FPS) vs. millions of point primitives rendered in Glsl visualization using 1.5Ghz CPU draw thread.

8) View into scene, at rear with respect to the physical camera orientation and position. Large, visible holes in the surfaces are mostly projector occlusions from foreground objects.

9) Diagram of multiple scan-rig view configurations captured over time by progressively orbiting the camera rig around the subject.

Abstract

Interactive Visualization of a 15 Million-Point Scene,
Captured With a Structured Light Point Scannner

Jonathan C. Schmid
Camillo J. Taylor, Advisor

This thesis investigates the visualization of real world scenes captured and rendered as a three-dimensional point cloud. Dense color and range imagery of real world objects are scanned by a low-cost rig. The rig is made up of a 15 megapixel consumer camera and an off-the-shelf digital projector (*1024 x 768 DLP*). The captured scene data can be explored by a remote viewer who navigates it by controlling the virtual camera.

This thesis focuses on using computer vision procedures such as structured light to gather color and range data from the rig, and then use commodity programmable graphics hardware to interactively visualize flying through the captured space. The most likely subjects for future platforms of capture rig technology would include room interiors and fragile archeological objects or museum pieces.

Introduction

At present, objects and places on the internet are usually depicted with digital photographs or panoramic photo imagery. But viewers of these remote spaces cannot easily acquire a rich spatial understanding, because all possible views of a photograph or panorama can only convey a single perspective viewpoint. Some interactive panorama viewer tools synthesize a user-controllable sequence of images to increase immersion, but these have nearly always allowed camera control of only the view angle.

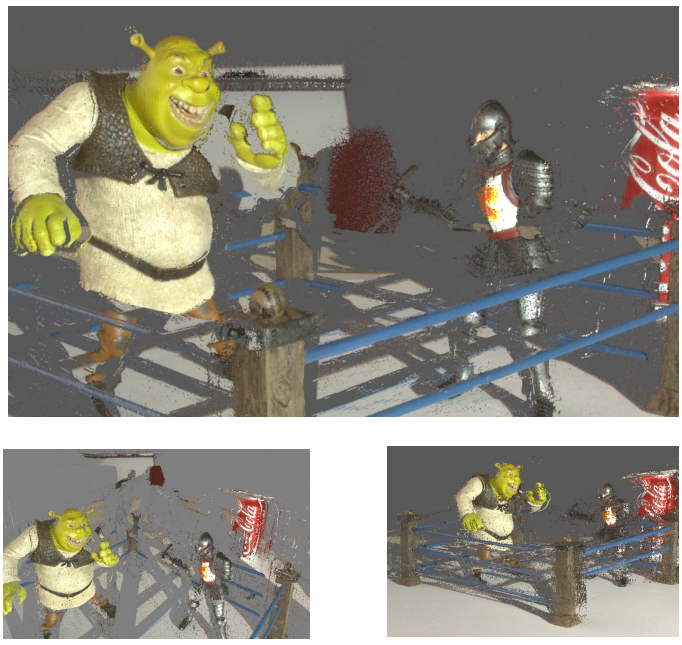
A user experiencing a digital scene can smoothly change the virtual camera's location within the scene in addition to varying the usual view angles. Control of a visualization camera is usually constrained only by simulation design. This contrasts sharply with the familiar human experience of viewing a proximal scene from one's own body under rigid kinematic constraints. Scanners will thus allow simulation users to experience complete viewing freedom within novel scenes from the real world.

Figure 1, Closeup image of bee pollen requiring macro-photography.

The 3D scene data can be created many different ways. The plastic Shrek figurines scanned in this thesis were captured by a rig we assembled from equipment readily available in many office facilities. A personal computer coordinated the capture process using our custom software, a standard 15 megapixel color camera, and an off-the-shelf Digital Light Processing (DLP) projector for displaying structured-light onto scene objects. The DLP is an array of tiny actuated mirror switches that has lead to affordable projectors. The 1024x768 DLP was the highest commonly available projector resolution at the start of this project, but 1280x800 models are now starting to enter a similar relative price range.

The figures (2.1, 2.2, and 2.3) below illustrate three different views of our scanned scene rendered from point primitives using GL Shading Language (GLSL). Some objects in this raw scan show many artifacts from range-errors, but the central figures are fairly well depicted, except for holes due to the single captured view, which cannot sample all geometry and leads to occlusions in expected areas.

Three rendered views from separate virtual camera positions.

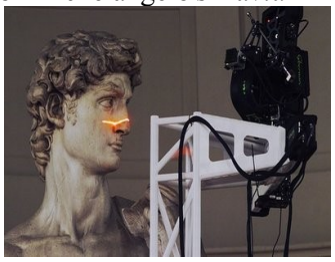


There are many well known 3D scanning efforts led by other computer vision researchers. Many of these groups have the goal of digitizing and preserving important objects and artifacts. The *Digital Michelangelo Project* is a famous example. It began in the early 1990s, organized by Stanford University Professor Marc Levoy. The *DMP* makes use of both laser-stripe and range-triangulation systems in order to push the state of the art in scanning, capturing marble statues and other carvings.

> The production phase of the project, from September of 1998 to June of 1999, consisted of scanning [Michelangelo's artworks]. For this purpose, Professors Marc Levoy and Brian Curless, Dr. Kari Pulli, and 7 graduate and 14 undergraduate students ... relocated to Italy for periods ranging from a few weeks to a year.
> … We scanned 10 statues: the David, the four Unfinished Slaves, and St. Matthew … the four allegorical statues (Night, Day, Dawn, and Dusk) in the Medici Chapels, ... and the architectural settings of both museums (Levoy, 0002).

The Digital Michelangelo Project's large-scale scanning effort such as in the 5 meter David sculpture scan depicted below in Figure 3, coupled with a substantial scholarly audience motivated commission of specialized equipment and use of substantial human labor. The scanning effort in this thesis, in contrast, focuses on demonstrating the interactive display of smaller scenes that we have scanned with readily available capture equipment. Our visualization techniques and high-resolution color approach demonstrate configurations that can be incorporated into the approaches of groups who seek to scan valued artifacts.

Stanford Digital Michelangelo Project,
Scan of Michelangelo's *David*



Scene Capture of 3D Point Data

Structured Light

Our rig projector shines a set of known striped image patterns onto the scene during the capture process, a technique known as structured light. The technique includes many approaches with differing types and patterns of light used. Our rig could be classified as an active, non-contact, triangulation-based sensor. Active denotes that the projector emits light into the scene, and non-contact refers to the absence of mechanical contact between scanner and object surfaces.

Due to the camera's large megapixel image size, our rig can make only partial use of parallel measurement while capturing each single-view. In consequence, the use of our multiple structured light images to compute a range image restricts the scene's appearance between the first exposure and the completion of the entire image set. The result is that global temporal coherence of the image set requires that both scene geometry and reflectance remain constant during any measurement sequence.

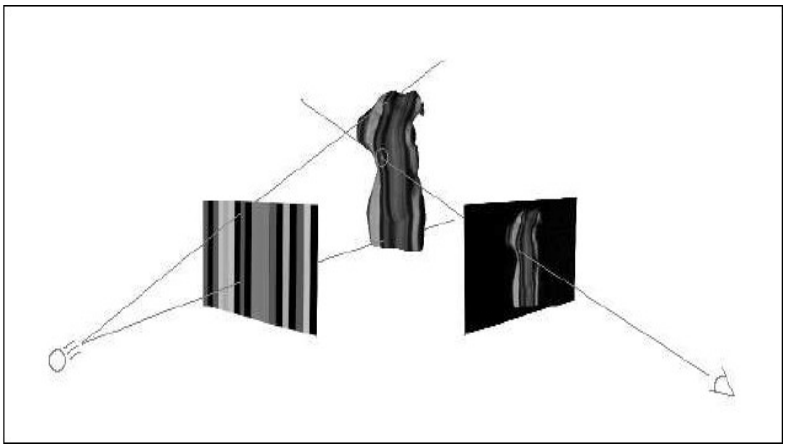


Figure 4: Conceptual sketch of projected patterns (left) as viewed by camera (right) on object.

The optical properties of scanned objects also affect rig performance. Excessively shiny objects with bright specular highlights may cause incorrect stripe decoding for nearby camera pixels. Highly reflective, refractive, or transparent materials are likely to divert light from its expected route in the scene, violating basic assumptions about ray-triangulation necessary to calculate range.

Per-Pixel Gray Coding

Gray codes are a binary coding scheme originated by Frank Gray of Bell Labs in his 1954 patent. They are used here to provide a mapping that addresses all pixels in the 15 megapixel camera image while it images light from a projector with only about 0.8 megapixels to use as binary outputs shining light onto scene geometry. A purely binary output of projector pixels is used, as opposed to varying color or gray-levels, because it was found to be reliably detectable and threshold-able when used in unknown scenes.

Camera calibration provides the camera matrix C, which describes the ideal pinhole camera projective transform. The calibration camera model parameters allow simultaneous calibration of both camera and projector relative to the scene through a process of imaging the projector's light in the camera view. Detection of the camera-projector point pairs through gray codes constitute sufficient means to automatically generate the necessary projector calibration data from the pairs.

Any 6 or more correctly corresponding camera-projector point pairs are sufficient to constrain projection matrix: P_c. However, some of the reportedly corresponding

points may be less accurate than others. Shiny or reflective objects may also generate some pairs in violation of the simple ray-triangulation assumptions. Thus we calculate many linear solutions to P_c, and employ non-linear minimization. We iterate over solutions i, looking for P that minimizes the sum:

$$\sum_i \left((x_i, y_i) - P(X_i, Y_i, Z_i) \right)^2$$ (Trister, 19 - 20).

The number of images we must gather over time within separate exposures is determined primarily by the number of bits needed for gray coding. Distinct column-coded (horizontal) and row-coded (vertical) components form a pixel's address. A set of patterns is first displayed in order to gather images that constrain the horizontal codes. A separate set of patterns constrains the vertical codes: [See Figure 5]. Lastly, two additional non-coded, uniform-exposure frames are recorded. The first of these reference images activates all projector pixels to 100% output, and the other activates no projector pixels. These two uniform-light images assist in thresholding the coded readings to classify pixels of the structured light patterns as *on, off, or invalid*.

Range Images from Disparity

A stereo pair of calibrated cameras is able to determine ranges into the scene for any pixel which one camera can correctly locate a correspondence along a one dimensional epipolar line in the other camera's image. Our rig uses the same triangulation geometry to determine pixel ranges, although coded structured-light addressing eliminates the need for correspondence search at every pixel.

With the camera and projector matrices C and P_c, range image formation begins by defining an infinite ray from the world space location of the pinhole camera's origin through each pixel's world space location on the image plane. As shown in the previous triangulation illustration, each camera ray will intersect the projection of a stripe-line. The stripe sweeps out into space forming a stripe-plane when imaged in the direction of the ray. The particular stripe captured in a given intersection measurement is known because of the gray code mapping which assigns one horizontal and one vertical stripe to each pixel in the image plane. The intersection for "reasonable" objects typically occurs within a small tolerance, on the order of millimeters, of the point imaged by the camera.

Visualization and Point Rendering

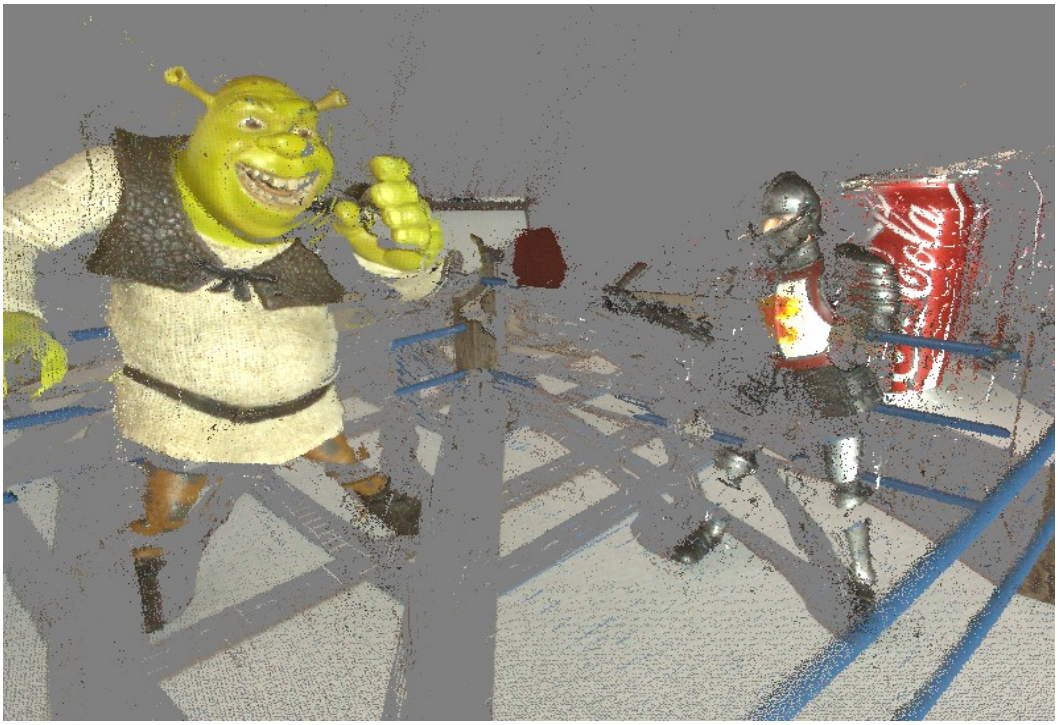
Point Based Rendering (PBR)

Point based rendering refers to visualization software that employs clouds of point-primitives for rendering. Calculation of *Point-Feature Sizes* within the previous section showed how the eventual screen space point-sizes are calculated when rendering the cloud of point-primitives at every frame. Using a properly calculated point-size based on both the scanner and virtual-camera conditions is very important for quality surface reconstruction without holes. Point size must be calculated every frame when moving the virtual camera.

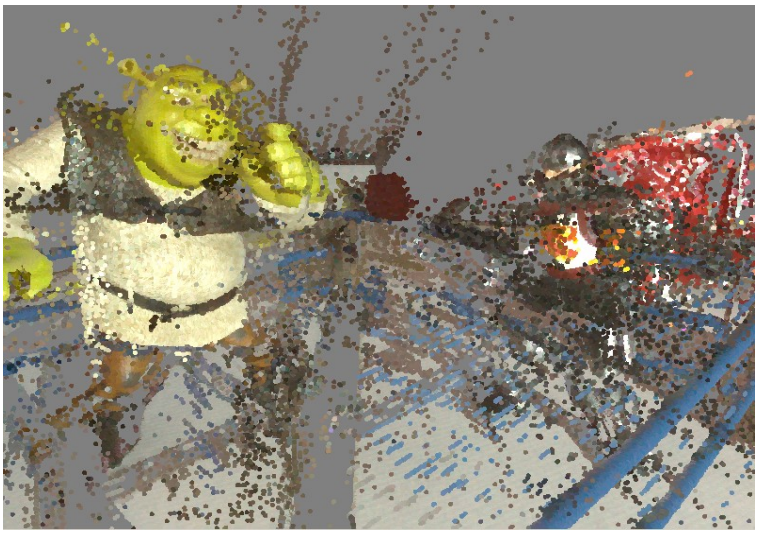
The following figures 2.1 and 2.2 illustrate the result of incorrectly using a static point size in screen-space that will likely be too big or too small for the optimal screen

height of a given pixel in an arbitrary camera position. The large Figure (2.1) at top on page 2, showed a view with correct point sizes. The following figures show the effect of static splat sizes of 1px, and 5px, which are likely to be incorrect for many splats.

1-Pixel Static Splats. Many holes in Shrek's hand at left where blue / gray background is visible.



5-Pixel Static Splats. Holes are mostly filled, but noise is emphasized and much color texture is lost.



Real-World Point Size

Run time virtual-camera control by the user necessitates that the actual rendered screen height of samples be calculated on a just-in-time basis during the rendering of each output frame. Point samples have an intrinsic world space height δ_h, which is constant and known when the scan data are processed. Therefore, part of the calculations that will contribute to the sample's eventual screen height can be precomputed and stored in a single float field for each image pixel.

The object's world space height can be calculated from the scan depth Zs, and the focal length of the scan camera Fs: $\delta_h = (Zs / Fs)$. The symbol Fv represents the focal length of the virtual camera, which is defined by its field of view. If virtual camera lens parameters are not under user control as in our demo software, then Fv is constant throughout execution.

The complete formula for projected sample screen height, H_v, can be written as:

$$H_v = (Fv / Fs) * (Zs / Zv)$$

The Zv term depends on virtual camera position and is addressed at run-time. The other scalars are combined in a single float scalar and passed into the vertex shader through a texture coordinate attribute of the current point primitive. The value of $(Fv * Zs / Fs)$ is read in the vertex shader and divided by Zv, yielding the desired H_v. This final value is used to set the point-size vertex shader variable controlling circular 'splat' size, (a screen-projected circle) in the glsl pipeline stage: 'VERTEX_PROGRAM_POINT_SIZE_ARB'.

Accelerated Rendering

Naïve rendering of all point primitives directly from an OpenGl application side for-loop was observed to achieve a frame-rate of just under 1 frame-per-second drawing the scene's roughly 7 million valid point samples. We noted that the 8800GTX graphics card was faster under this naïve scheme than our prototype Matlab implementation, but not interactive enough for user camera control.

Since the point-feature data are static throughout the visualization, we tested drawing the point cloud from a compiled display list. OpenGl was able to move the display list data into server-side memory because it fit easily into available VRAM. This relieved transferring the point cloud over PCI-E bus during every frame. The speedup was substantial, resulting in an improved frame-rate of approximately 8.5fps on the 8800 card.

Note that on a card such as the *8800GTX*, the CPU is still involved in dispatching the display list into the pipeline at the start of every frame, although at least it does not have to copy all point cloud data over the bus. For large data sets that fit in VRAM, this can yield a speedup, but overall performance is still usually influenced by CPU clock speed. We observed a 50% reduction in overall application frame rate when executing on a dual core 1.5Ghz P4, as opposed to a faster dual-core 2.9Ghz P4. This slowdown occurred despite the 1.5Gz machine's newer *GTX 260* card with many more GPU cores than the *8800GTX* card. See [Figure 7] for a table of average point-rendering frame-rates observed with our application using a GeForce 9400 card and 1.5Ghz P4 CPU under different point cloud sizes.

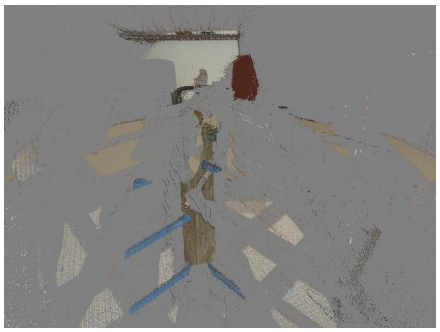
According to Nvidia and the OpenGl 4 specification, cards in the recent 'Fermi' series support OpenGl 4 capabilities and are able to render large numbers of point-features from a *CUDA* or *OpenCL* calculated vertex buffer object (VBO) without any per-frame interaction from the CPU. The speedup for a large point cloud data set like this one could be substantial if the need for high CPU clock speed was avoided when rendering huge numbers of primitives each frame.

Towards Practical Scene Capture

The thesis demonstration scan was imaged from a single projector position and a single camera position. To improve modeling accuracy, one could move the scanner rig in two different ways between each data-set batch of 26 photos. Moving the camera component would allow collection of data from multiple views, whereas moving the projector would allow scene imaging under different lighting angles.

If we move the rig's camera between data sets, many views with complimentary range data can be merged to fill occlusions, increase spatial resolution, and increase the scene's overall extent. Figure 8 shows a view at the far back of the scene with respect to the scanner camera. Only data from additional views can correct the substantial areas missing from scene surfaces.

Figure 8: Occlusions at rear of scene.



If we instead move the rig's projector, we gain data that allows us to simulate how the object should appear under novel lighting conditions. Varying the angle of the projector yields additional color samples for each camera-pixel. This might allow investigation and approximation of a Bi-Directional Reflectance Function (BDRF) for most or all scene points. Since the projector is the dominant directional light source in the scene, we

can study the change in observed brightness under varied lighting to probe a particular material's reflectance.

Multiple data sets with many redundant samples gathered under varying projector positions could allow enough BDRF samples to detect scene materials. Such information could perhaps allow automatic shading function capture and assignment within the scene. Massive savings in video memory usage could be gained if even a portion of scene point-samples instead stored a shader classification and permitted shading by OpenGl Shading Language (Glsl). All shader-assigned pixels could save several bits per pixel by avoiding storage of a per-point color sample.

Final screen pixel-color determination is made in the Glsl pipeline's fragment shader. It is unlikely that we will be able to initially assign shading functions to every scene point due to uncertainties, thus some form of smooth-point interpolation would help to blend raw-captured point samples with the shader computed samples in a seamless and believable way. A point interpolation algorithm as reported by Zhang and Pajarola elaborates how to compute fragment output color **c**(f) as a weighted sum of colors from splats that overlap a given pixel. The weighted contribution W_i to the pixel varies with distance between "the fragment's intersection **f**(i) with the splat plane of point p_i in object space (Zhang, 2)."

Sum of Weighted Averaged Splats:

$$\mathbf{c}(f) = \frac{\sum_i w_i(\mathbf{f}_i) \cdot \mathbf{c}_i}{\sum_i w_i(\mathbf{f}_i)},$$

Determining the shader code and its assignments to the many scene points would undoubtedly require massive amounts of computation, and much more memory than our raw range and color data set. Fortunately this analysis could take place offline with respect to the simulation. It could be performed at the same time as the sub-pixel interpolation we used to improve range estimates beyond the native 1024x768 pixel projector resolution. Offline enhanced-range estimates and per-point shader assignments would both be baked into the simulation data-file for use by the simulation run-time.

Range Compression

The number of views will remain limited by the size of VRAM on today's commodity video cards, but compression of the basic data may help to gain as many views as possible on given hardware. Compression of raw range data may be possible during storage, although most visual domain compression algorithms were not optimized for compressing range samples. Instead, specialized range-image compression techniques must be used to reduce the size of the raw range data as much as possible without distorting it (Kerber et al., 1-2).

Range image studies have revealed that significant compressible structure does exist within range imagery, even if it differs significantly from that found in color images. In an investigation of the statistics describing natural range images, Huang, Lee, and Mumford conclude that range images have simpler features than their optical image counterparts, at least after transforming range samples via logarithm.

They point out fundamental patch features:

> The world can be broken down into piecewise smooth regions (in, for example, range or intensity) that depend little on each other (5).

And suggest hope for reasonable feature segmentation:

> The concept of “objects" is often better determined in terms of changes in range, than in terms of changes in intensity, color, texture, lighting etc (5).

Thus, the right compression technique may be able to save significant space while storing the range-image in VRAM without sacrificing many subtle geometric features encoded by the original measurements.

There are now many common algorithms for direct range-image compression, and the 3D point sets they imply. These include fractal methods, geometric and topological compression schemes, range-image quadtrees, 3D octrees, implicit sphere trees, wavelets, and a variety of other transforms. Applications like ours that use range-scanner rigs to produce large scale color and range imagery can select from among these specialized approaches in order to preserve VRAM without overly degrading range performance or especially introducing major decompression delays in the rendering pipeline. For end users, more 3D data will fill the occluded holes and extend the overall size of captured scenes.

Figure 5: Example patterns for encoding vertical pixel position via gray codes.

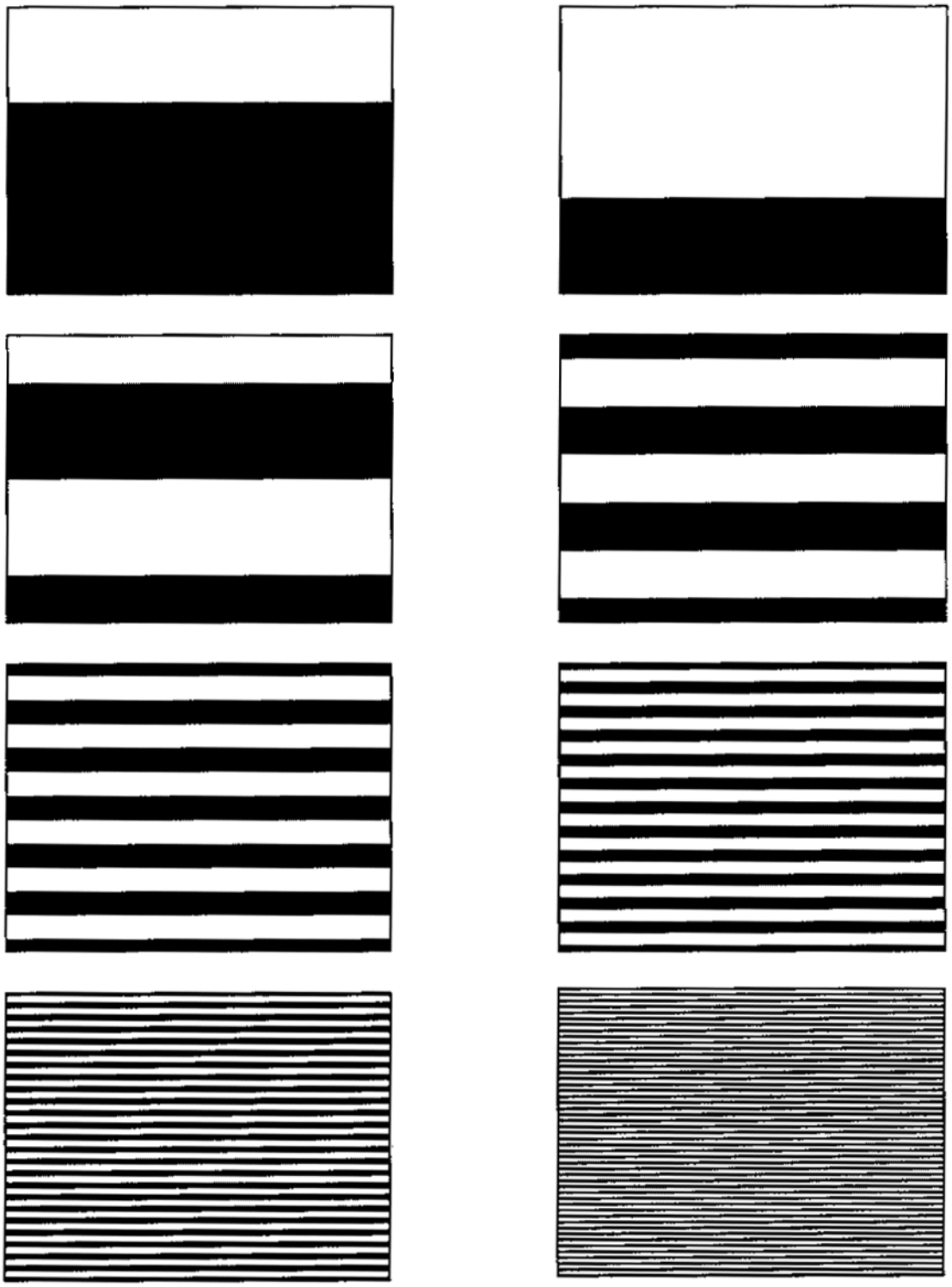


Figure 9: Diagram Showing Multiple Camera Views During a Structured Light Projective Scan (Levoy, *99)*.

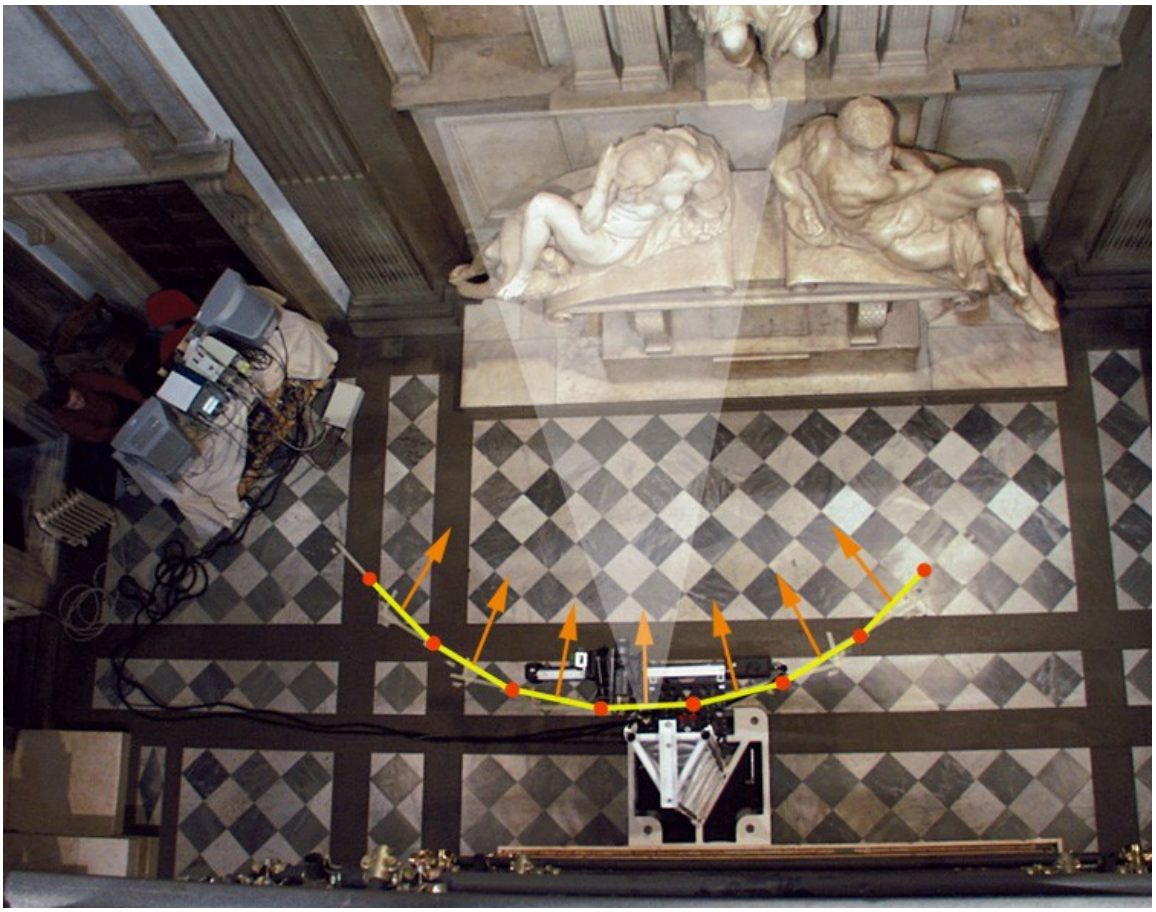
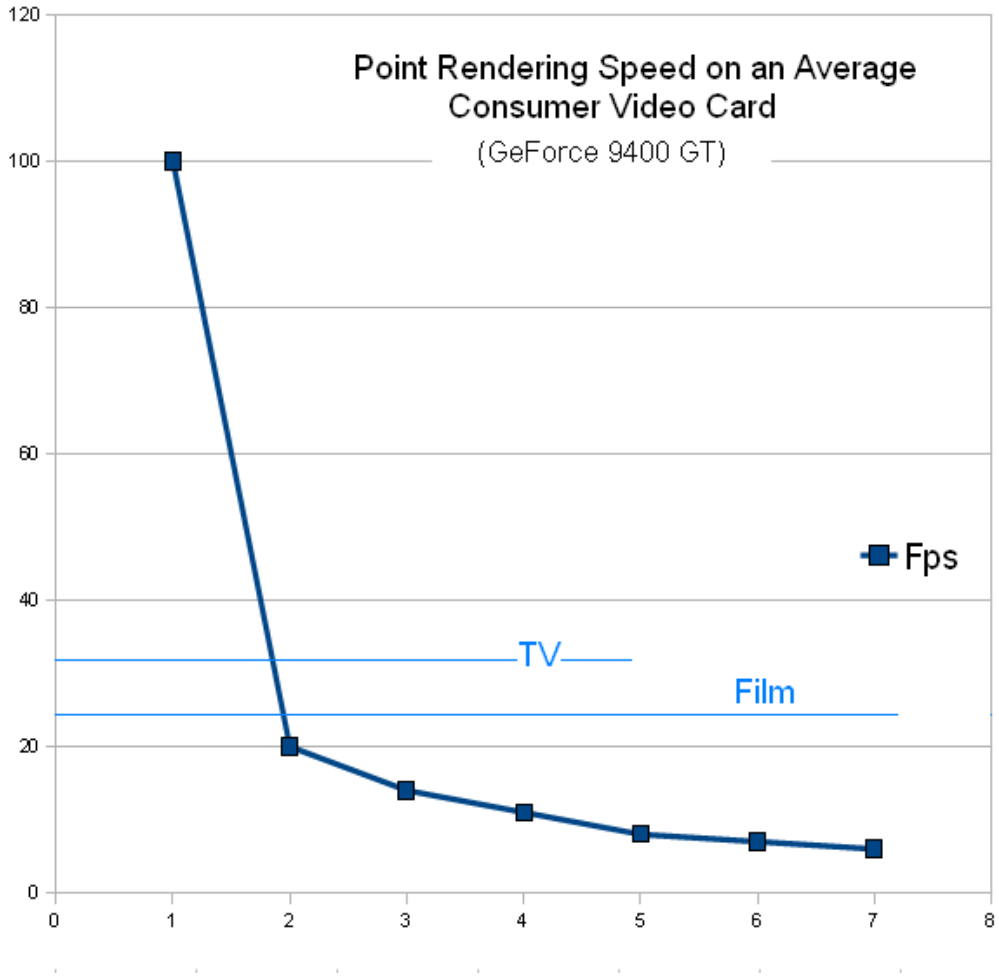


Figure 7: Point rendering performance of our application under differing numbers of input point primitives.

M (Pts)	Fps
1	100
2	20
3	14
4	11
5	8
6	7
7	6



References

Huang, J., Lee, A. B., and Mumford, D., “Statistics of Range Images.” Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2000), pp. 324-332.

Kerber, J., Belyaev, A., and Seidel, H., “Feature Preserving Depth Compression of Range Images.” Proceedings of the 23rd Spring Conference on Computer Graphics (2007), pp. 110-114.

Levoy, M., “The Digital Michelangelo Project.” Proceedings of the Second International Conference on 3D Digital Imaging and Modeling, Ottawa, Canada, October 5-8, 1999.

Trister, Andrew D. “Reconstruction of Three-Dimensional Space Using Structured Light”. MD-PhD diss., University of Pennsylvania, 2002.

Zhang Y., Pajarola R., “Single-Pass Point Rendering and Transparent Shading.” Proceedings of the Eurographics / IEEE VGTC Symposium on Point-Based Graphics ’06 (2006), pp. 37–48.